UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS

INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas y Automática

Área de Telemática

PROYECTO FIN DE CARRERA

APLICACIONES WEB PARA EL ACCESO A INTERNET A TRAVÉS DE LA RED GSM USANDO HTTP Y WAP

Autor: **Aurelio Mariscal Ortiz**
Tutor: **Teresa Ariza Gómez**
Tutor en la empresa: **Julio Jesús Sánchez García**

Marzo, 2001

A todos aquellos que me quieren y saben que yo les quiero y muy especialmente a aquellos que saben que son mis amigos y sin cuyo cariño y apoyo no hubiera podido llegar a donde he llegado.

A mis extraordinarios compañeros de trabajo, sin cuyo apoyo jamás podría haber llevado a buen puerto la realización de este proyecto.

MEMORIA

Índice de Contenidos

1. INTRODUCCION

Este primer capítulo pretende servir como una introducción general al proyecto que se va a desarrollar. Antes de plantear el problema que se pretende resolver, debemos conocer el entorno para el cual se desarrolla esta aplicación y, de esta forma, saber en todo momento cual es el objetivo final que se pretende alcanzar a medida que se vayan exponiendo los pasos seguidos en la evaluación del problema y posterior búsqueda de soluciones. Se dará una descripción general de la arquitectura del sistema en el cual se engloba nuestro trabajo, ofreciendo una descripción suficiente tanto del hardware como del software, así como las relaciones entre ellos y la comunicación con aquellos elementos externos al sistema con los que interactúa (se trata de un sistema que para realizar sus funciones necesita determinada información obtenida por elementos externos a nuestro sistema). Una vez tengamos una visión general del sistema con el que se trabaja, se planteará el problema a resolver y que es objeto de la elaboración de este proyecto.

1.1 Descripción general del sistema

El sistema es una aplicación que lleva a cabo la gestión, evaluación y asignación de los recursos humanos correspondientes a personal técnico de una empresa de telefonía. Se trata de una aplicación de **Workforce Management** que permite realizar, en tiempo real, un seguimiento continuo de las actuaciones que requieren los servicios solicitados por los clientes y de las incidencias ocurridas en planta relacionadas con las afecciones del servicio. Permite llevar un control de todas las actividades que se generen y que requieran una actuación (conjunto de acciones a realizar por un grupo técnico con el fin de reparar o suministrar un servicio solicitado por un cliente). Dentro de estas actividades se incluyen aquellas relativas a mantenimiento preventivo (son aquellas actividades programadas por la empresa para garantizar un funcionamiento óptimo de toda su infraestructura a largo plazo) y a la diagnosis y resolución de averías y reclamaciones de clientes (comunicación por parte de un sistema externo de una incidencia o fallo de un servicio, es decir, la causa o motivo por el cual se provoca la perdida de operatividad de un servicio). Toda petición del cliente o incidencia va a provocar la puesta en marcha de las actuaciones necesarias para la instalación o el restablecimiento del servicio afectado. Este sistema optimiza y agiliza el tiempo de resolución de la actuación, y los costes y recursos necesarios para la misma.

En sucesivas versiones de la aplicación general sobre la que funciona este sistema se ha conseguido desarrollar un algoritmo muy potente el cual tiene como finalidad determinar en cada momento que grupo técnico es el más indicado para llevar a cabo la

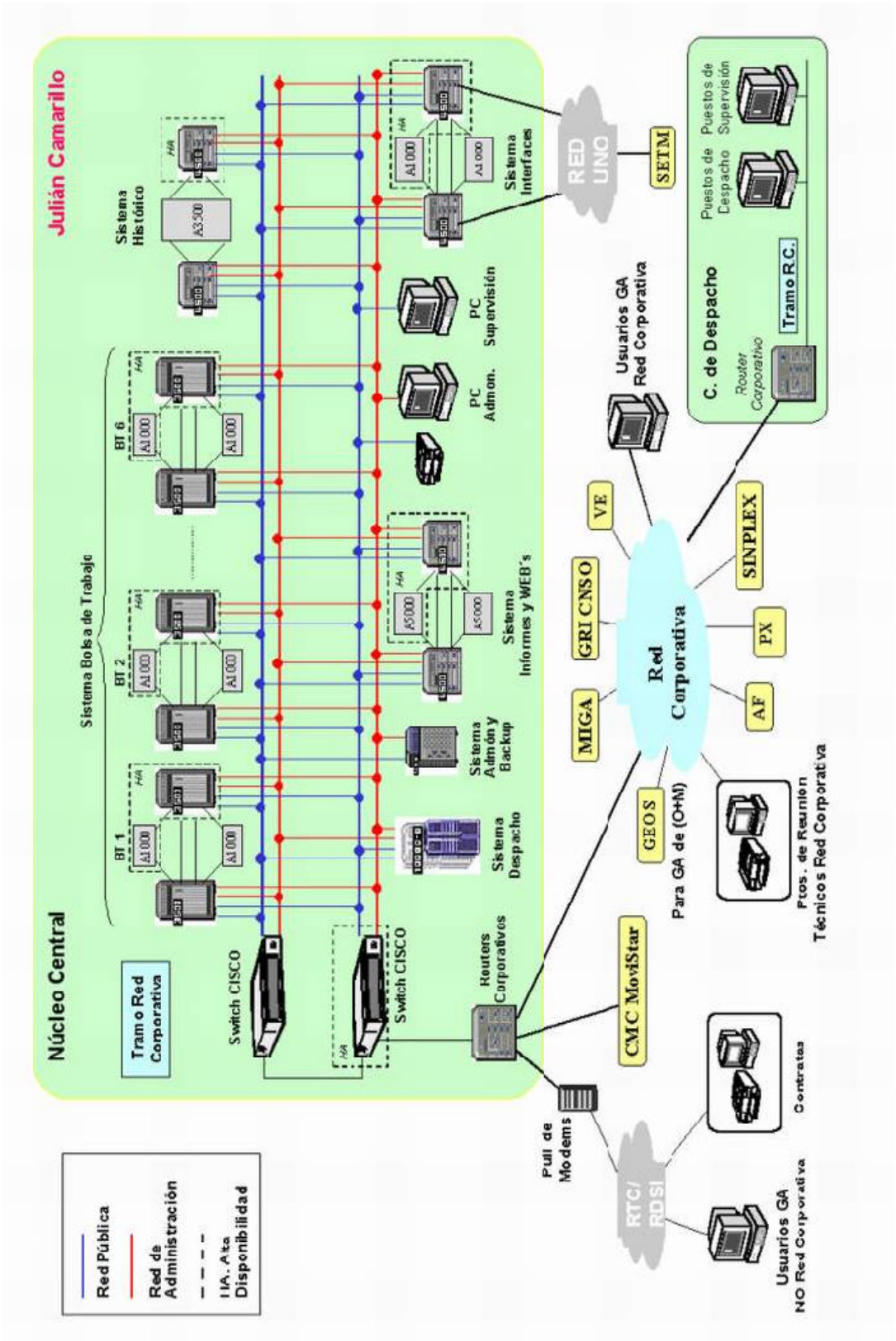
resolución de esa actuación (por proximidad geográfica, cualificación de los integrantes del grupo para resolver el problema, situación laboral actual, y otros muchos factores) y le hace llegar a dicho grupo técnico toda la información que estos necesitan referente al trabajo a realizar (domicilio del abonado, cita previa con el mismo, descripción breve del problema,...). Este algoritmo de planificación tiene como objetivo principal la propuesta de planes de trabajo para conjuntos amplios de grupos técnicos, considerando las actuaciones previstas dentro de un horizonte temporal flexible (típicamente de varios días). Debido a la naturaleza dinámica del despacho de actuaciones (mientras los técnicos están ocupados con la resolución de las actuaciones que tienen asignadas en cada momento siguen entrando actuaciones en el sistema debidas a las reclamaciones de los clientes), este objetivo es complementario a atender todas las variaciones que se puedan producir en los planes de trabajo, debido a la llegada de nuevas actuaciones, retrasos o adelantos en la resolución de actuaciones planificadas, etc...

El algoritmo, cuando se termina de ejecutar, dará una lista de actuaciones que cada técnico puede realizar (planificación para cada uno), permitiendo al despachador asignar manualmente qué actuación de todas las propuestas le asigna a cada uno. También es posible asignar prioridades a la resolución de las actuaciones de forma que, al tener una lista ordenada, no sea necesaria le presencia física del despachador, encargándose nuestro sistema de asignar automáticamente la siguiente actuación por orden de prioridad asignada a un técnico cuando este haya terminado con la que tenía asignada anteriormente.

El algoritmo recibe como datos de entrada un conjunto de actuaciones y un conjunto de grupos técnicos, junto con una serie de parámetros generales de configuración, y produce como resultado una planificación del trabajo de los grupos técnicos a lo largo del horizonte de planificación especificado. Se ejecuta como un proceso independiente, servidor de planificaciones, al que uno o más Despachadores Automáticos del sistema pueden realizar peticiones de planificación. El número de actuaciones y grupos técnicos que puede tratar es, en principio, ilimitado, estando su capacidad y rendimiento condicionado por las características del hardware sobre el que se ejecuta.

En la página siguiente, en la **Figura 1**, podemos ver la arquitectura del sistema **Gestor de Actuaciones**, para el cual ha sido desarrollado este proyecto. Podemos observar la estructura del núcleo central del sistema así como el hardware que lo constituye. También puede apreciarse el conjunto de interacciones que se produce con este núcleo durante el desarrollo de la actividad normal del sistema.

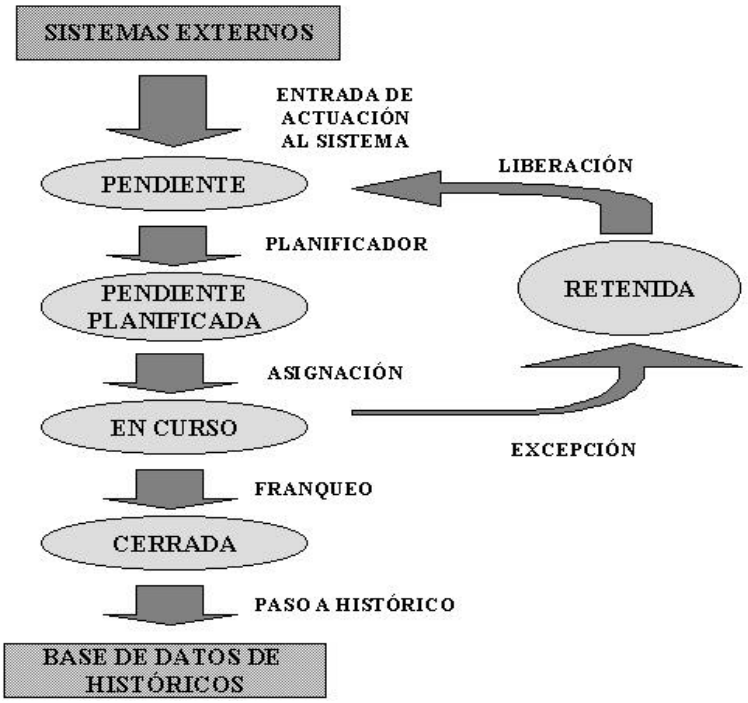
Figura 1.- Arquitectura del sistema Gestor de Actuaciones.



Se trata de un sistema real situado en Julián Camarillo (Madrid), propiedad de **Telefónica España S.A.** y desarrollado por **Telefónica Investigación y Desarrollo**. En la **Figura1** puede apreciarse la interacción del núcleo del sistema con otros sistemas externos de los cuales obtiene la información que necesita para desarrollar su trabajo (usa interfaces desarrollados para posibilitar esta comunicación y esta se produce a través de la red corporativa). También observamos la forma de comunicarse de este núcleo con aquellas partes del sistema que se encuentran fuera del núcleo de este (**RDSI**, **RTC**, **CMC de Movistar**,...). También han sido desarrolladas interfaces específicas del núcleo del sistema con estos otros componentes del mismo y que van a posibilitar los accesos remotos al mismo. Todas aquellas partes del sistema que interese conocer para la comprensión de este proyecto serán explicadas, con la profundidad que sea necesaria, a medida que vayamos avanzando en la elaboración de este documento.

Las actuaciones tienen un ciclo de vida que comienza cuando son dadas de alta en el sistema y termina cuando se dan por finalizadas. Durante su existencia, se van a realizar sobre ellas ciertas operaciones, "**procesos**", que van a provocar la evolución de la actuación a través de sus "**estados**" posibles, comenzando por el estado de pendiente, y finalizando, una vez resuelta la actuación, en el de cerrada. En la **Figura 2** podemos ver el ciclo de vida de una actuación.

Figura 2.- Diagrama de estados del ciclo de vida de una actuación.



Cuando una actuación entra en el sistema proveniente de un sistema externo pasa al estado de "**pendiente**". Una actuación estará "**pendiente planificada**" cuando haya sido planificada y esté esperando a que el grupo técnico al que esté asignada quede libre. Una actuación estará "**en curso**" cuando haya comenzado su proceso de resolución por parte de un grupo técnico. Cuando este queda libre se procederá a la asignación de la siguiente (bien manual o automáticamente). Una vez resuelto el problema por parte del grupo técnico enviará al sistema toda la información que este necesite para el buen funcionamiento del mismo (actuaciones resueltas que salen del sistema, disponibilidad del grupo técnico a partir de ese momento, apuntes para contabilidad, información para la elaboración posterior de informes,...). Los técnicos también podrán informar solo de la evolución de la actividad. Una vez resuelta la actuación por parte del grupo técnico y tras la comprobación por parte del sistema de la coherencia de los datos que este último ha reportado una vez finalizada su actividad, la actuación pasará al estado de "**cerrada**", quedando liberado el grupo técnico en cuestión.

Un caso especial que queda representado en la **Figura 2** es cuando la actuación pasa al estado **"retenida"**. Esto tiene lugar cuando existen circunstancias especiales que impiden un ciclo de vida normal de la actuación (averías masivas debidas a fallos de algún elemento de red que desencadenan numerosas reclamaciones por parte de los clientes, imprevistos que impiden la resolución de una actuación por un técnico,...). Se dice que se ha producido una excepción. El técnico informa de la incidencia acontecida y la actuación queda retenida. El sistema tomará las medidas necesarias y, una vez solucionado el problema, liberará estas actuaciones, las cuales volverán al sistema para su resolución. Muchas de ellas (el caso más significativo sería el de aquellas reclamaciones llegadas al sistema por un fallo de red) pasarán rapidamente al estado de cerradas tras la comprobación por parte del técnico de que su existencia era debida a esas circunstancias especiales y que, después de las medidas adoptadas, han quedado automáticamente resueltas. Otras serán resueltas por técnicos preparados para los imprevistos que habían surgido.

En el último paso vemos que las actuaciones salen del sistema y pasan a una base de datos de históricos para cualquier tipo de uso futuro.

Como grupo técnico se entenderá un grupo formado por personal técnico cualificado para la resolución de las actuaciones. Un grupo técnico puede estar formado por varias personas o por una sola. Cada grupo técnico tendrá una serie de características como puede ser el perfil técnico (características técnicas que definen la cualificación de un técnico y que le habilitan o no para la resolución de determinados trabajos), la disponibilidad (jornada laboral, vacaciones, bajas temporales, horario de comidas, etc…), su ámbito de acción, etc..., que dependerá de las características individuales de cada uno de los integrantes del grupo.

El sistema permite realizar, en tiempo real, un seguimiento continuo de las actuaciones que requieren los servicios solicitados por los clientes. Se pretende que el proceso **"petición del cliente – actuación – resolución de la petición"** sea optimizado en

coste, tiempo y recursos, de manera que se produzca una mejora en la respuesta en dichas reclamaciones. En último extremo, el sistema proporciona métodos para que este proceso sea totalmente **automático**.

El sistema interacciona con los sistemas específicos del Área Comercial encargados de la atención al cliente. De los sistemas del área comercial se obtiene información sobre los requerimientos (reclamaciones u órdenes de servicio que requieran una actuación) del cliente en forma de boletines. Proporciona las interfaces necesarias con los sistemas de operación y supervisión de red. Por medio de estas interfaces se obtienen datos en tiempo real sobre elementos de red de transmisión y conmutación afectados por una incidencia.

El sistema se puede integrar con los sistemas de recogida de reclamaciones e incidencias preexistentes en cada una de las redes de telecomunicaciones. La integración se consigue con el diseño de interfaces específicas con los sistemas existentes o bien con el conjunto de funcionalidades que soporta el núcleo del sistema. Del resultado de esta integración no traumática con los sistemas de gestión preexistentes se obtiene un alto aprovechamiento y reutilización de las inversiones ya efectuadas, que se ve facilitada con la previsión de un plan de migración específico para cada uno de estos sistemas.

Dentro del sistema, una actuación tiene asociado un **boletín**, que es un formato electrónico que contiene todos los datos necesarios para identificar las actividades a realizar: tipo de trabajo, localización, cliente, equipo y material necesario para llevar a cabo la actuación, etc.. El sistema dispone de las capacidades suficientes para llevar a cabo la gestión de dichos boletines, tanto manualmente como de forma automática, gestión de cambios de estado, gestión de histórico de actuaciones cerradas, etc...

Cada vez que se produzca en cualquiera de los sistemas externos con los que se comunica el nuestro la necesidad de poner en marcha una actuación (por la venta de un equipo, la solicitud de un servicio, el aviso de una avería, una alarma en una cabina, etc...), la interfaz entre nuestro sistema y el sistema externo correspondiente creará de forma automática su boletín de actuación e introducirá en la **Bolsa de Trabajo** dicha actuación como pendiente de planificar, comprobando con anterioridad la consistencia de los datos introducidos en el boletín (la Bolsa de Trabajo almacena información sobre la actividad del sistema, es decir, de todos aquellos eventos que se considere necesario registrar, generarán una entrada en el registro de actividad en la que se indica los datos más relevantes de dicho evento).

El sistema también permite que el usuario cree de forma manual nuevos boletines de actuación (de averías e instalaciones o de mantenimiento preventivo). Los nuevos boletines creados formarán parte de una carpeta de actuaciones ya existente, o que haya sido creada de forma expresa. Durante el proceso de creación el usuario debe cumplimentar una serie de campos de forma asistida que permitan al sistema el

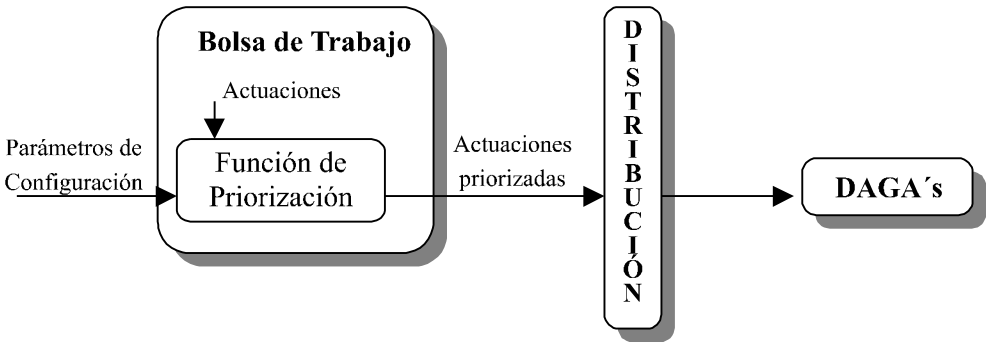
tratamiento posterior de dicha actuación hasta su distribución a una de las Posiciones de Despacho.

Al sistema **Gestor de Actuaciones** (nombre con el que se conoce realmente) le llegan de forma simultanea un gran número de actuaciones, las cuales debe planificar. Para llevar a cabo esta planificación, calcula y asigna a cada actuación una prioridad.

El cálculo de esta prioridad permite otorgar a todas las actuaciones pendientes que existan en el sistema un número que indica su prioridad frente al resto de las actuaciones. De este modo, ante la entrada simultánea en el sistema de varias actuaciones, el sistema da preferencia a la realización de algunas de ellas, generalmente aquellas con carácter más urgente (prioridad baja). La prioridad permite decidir el orden de resolución de las actuaciones cuando compiten por los recursos.

Nuestro sistema incorpora un algoritmo para el cálculo de las prioridades de las actuaciones, de forma que los parámetros que definen su comportamiento son configurables, permitiendo al usuario del sistema adaptarlo a sus necesidades. Para ello, se determinan todos los parámetros de una actuación que se considera afectan a la prioridad de realización de la actuación. Este conjunto de parámetros constituye la entrada al **Proceso de Priorización**. Este proceso proporciona a su salida un **Código de Prioridad** asociado a cada actuación. En la **Figura 3** vemos representado este proceso de priorización.

Figura 3.- Proceso de priorización.

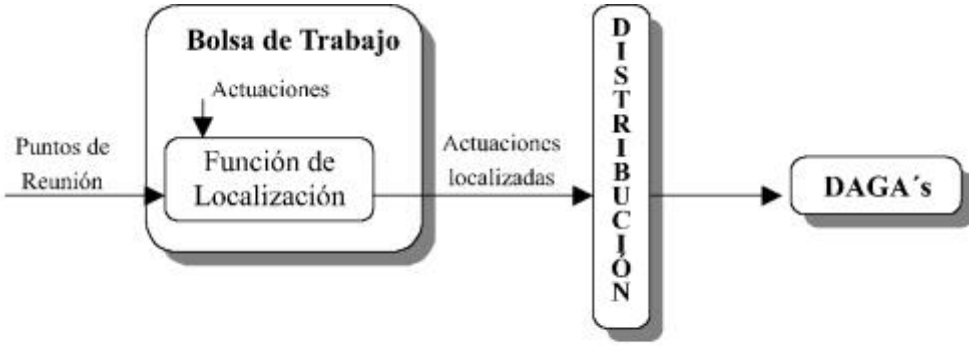


En la **Figura 3** vemos como las actuaciones priorizadas pasan al **algoritmo de distribución**. Este último las distribuye entre los distintos técnicos y su planificación es enviada a los **DAGA's** (**Despachadores Automáticos del Gestor de Actuaciones**) que son los encargados de hacérselas llegar a los técnicos.

Cada criterio de asignación de prioridad puede configurarse para que afecte en mayor o menor medida a la hora de hacer el cálculo de la prioridad. Esa configuración la realiza la persona que opera el sistema atribuyendo un peso a todos los valores posibles de cada criterio de asignación de prioridad, que utiliza la función de asignación de prioridad de la Bolsa de Trabajo. Mediante esta funcionalidad se ofrece al usuario la posibilidad de priorizar las actuaciones que llegan al sistema, según el esquema de prioridades que desee el usuario responsable del mismo.

Para que nuestro sistema Gestor de Actuaciones pueda realizar una óptima asignación de actuaciones a grupos técnicos debe conocer la situación geográfica de varios elementos, las actuaciones, los grupos técnicos y los **puntos de reunión** (lugar donde se reúnen los técnicos de un grupo para comenzar la jornada laboral). Este proceso de localización se realiza antes de la distribución de actuaciones, para poder garantizar que lleguen las actuaciones con toda la información que necesita el algoritmo de planificación. En la **Figura 4** se observa la participación del proceso de localización en la distribución de actuaciones.

Figura 4.- Proceso de localización.



Cualquier actuación que llegue al sistema se localiza geográficamente mediante una coordenada **UTM**. La localización de una actuación es crucial, tanto para la distribución de las actuaciones entre centros de despacho como para la asignación de las mismas a grupos técnicos. Las coordenadas que se utilizarán para localizar una actuación son, por este orden de preferencia:

- **Las coordenadas explícitas que definen la localización de la actuación en función del par caja terminal / central.**

  Dada una central y una caja terminal, el sistema calcula las coordenadas UTM correspondientes a la actuación.

- **Las coordenadas de la central.**

  Si la actuación puede asociarse a una central pero no a una caja terminal.

- **Las coordenadas de la localidad.**

  En el caso de localidades pequeñas (poca población y extensión), y si el sistema no encontrase, a pesar de todo, la coordenada geográfica de la central, tratará de localizar la actuación mediante la coordenada UTM de la localidad en la que se desarrolle la actuación.

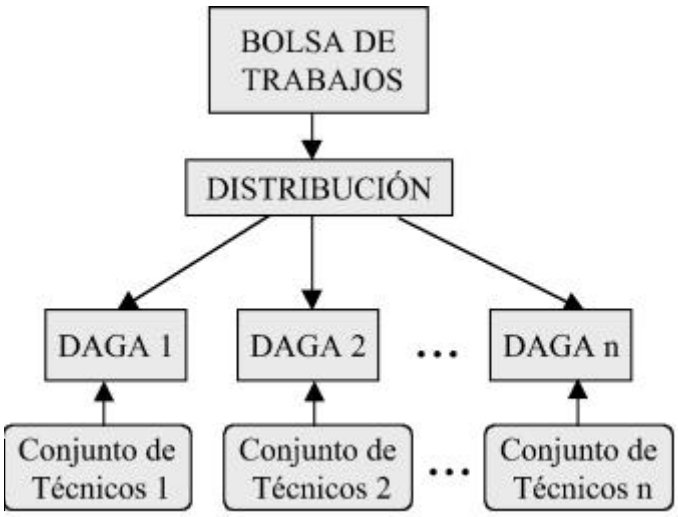
El sistema también contempla la posibilidad de modificar / corregir la localización de una actuación manualmente, asignando unas nuevas coordenadas UTM a la actuación. No se permite en cambio la definición directa de la coordenadas UTM de la actuación, sino que se permite que la localización de la actuación sea igual a la de una caja terminal, una central, una localidad o la de la última actuación realizada por un grupo concreto.

La localización de los puntos de reunión es fundamental para calcular los tiempos y costes del desplazamiento para la primera actuación del día y para el retorno al punto de reunión tras la última actuación, aunque esto puede no ser completamente necesario (material preparado del día anterior, o al finalizar la última, desplazamiento directamente al domicilio del técnico). Además, puede ser necesario en ocasiones pasar por el punto de reunión entre dos actuaciones para conseguir material y (o) documentación.

La localización de los grupos técnicos es la de su punto de reunión, al comienzo de su jornada laboral, y la de la última actuación realizada durante el transcurso de la jornada.

La funcionalidad de Distribución de actuaciones consiste en arrancar los procesos que llevarán a cabo la asignación de dichas actuaciones a los grupos técnicos. Al proceso de distribución, como ya se ha indicado en los apartados anteriores, le deben llegar las actuaciones convenientemente priorizadas y localizadas. En la **Figura 5** se representa este proceso de distribución.

Figura 5.- Proceso de distribución de actuaciones.



Para la asignación de actuaciones a los técnicos el algoritmo tiene en cuenta numerosísimos factores, además de la proximidad geográfica y de la cualificación de los técnicos para determinados trabajos, tales como el tipo de vehículo del que dispone el técnico para desplazarse hasta el lugar donde se ha producido la incidencia.

La distribución de actuaciones se puede realizar de forma manual o mediante un proceso automático, para lo cual deben existir unos criterios de distribución que gobiernen dicho proceso.

La persona que opere el sistema, y que tenga los permisos adecuados, podrá configurar la tabla de distribución en la cual se establecerán los criterios a seguir por el proceso automático en la distribución de las actuaciones.

Una vez establecidos dichos criterios, el sistema puede realizar de forma automática, utilizando la tabla de distribución en vigor y los propios valores que para cada actuación presenten los campos del boletín, la distribución de las actuaciones "pendientes" de la Bolsa de Trabajo a los grupos técnicos que se consideren adecuados. Una de las principales fuentes de ahorro de costes que proporciona el sistema es la eliminación de la figura del "despachador", que era el que hasta ahora desarrollaba este trabajo y que, por tanto, podrá ser empleado en lo sucesivo para otras cosas.

El sistema también ofrece la capacidad de poder realizar la distribución de las actuaciones a los grupos técnicos de forma manual.

El sistema es capaz de retener todas aquellas actuaciones que estén relacionadas con una incidencia de servicio grave (averías masivas de planta exterior o trabajos programados) debido a la imposibilidad de proseguir con la resolución de dicha actuación hasta que no se solucione la incidencia de servicio grave. Se dispone de una tabla en la que se encuentran reflejadas todas las causas posibles de retención automática y la información necesaria para localizar las actuaciones relacionadas con ellas. La retención de actuaciones se puede hacer de forma automática debido a la existencia de una incidencia de servicio grave o manualmente con la generación de un boletín de actuación correspondiente a una avería masiva.

Una vez que la incidencia de servicio grave es resuelta, el sistema recibe la notificación de dicha resolución para que a continuación libere todas las actuaciones que habían sido retenidas por ese motivo. Las actuaciones o averías que no hubiesen sido provocadas por la incidencia de servicio grave, son liberadas y devueltas a la Bolsa de Trabajo con el estado de "pendiente" para que vuelvan a ser planificadas. Las actuaciones que fueron provocadas por la incidencia de servicio grave son franqueadas y cerradas.

El sistema mantiene una carpeta en la que se incluirá todas aquellas actuaciones que requieran un tratamiento especial, por haberse producido algún problema en los procesos normales de tratamiento de dichas actuaciones. Además, el sistema notifica la excepción al supervisor de despacho de forma automática. Esta carpeta es común para todas las actuaciones en excepción, sean éstas del tipo que sean. La gestión de excepciones se utiliza como mecanismo de seguridad y control ante la falta o incorrección de determinada información necesaria para el tratamiento de la actuación en curso. El sistema enviará al supervisor la notificación correspondiente y paralizará la actuación actual hasta que éste resuelva la excepción.

El sistema implementa la funcionalidad de almacenar por separado aquellas actuaciones para las que haya transcurrido cierto tiempo desde su fecha de cierre, 7 días por defecto. Cada noche se realiza la comprobación del tiempo transcurrido desde la fecha de cierre de cada actuación y aquellas actuaciones que lleven cerradas más tiempo del indicado, son enviadas a unas tablas separadas con objeto de liberar memoria del sistema y almacenar el histórico de las actuaciones cerradas.

Aparte de estas facilidades, la posibilidad de acceso a estas bases de datos con herramientas de usuario final permite obtener completos informes y estadísticas de forma automática. Mediante esta funcionalidad, la persona que opere el sistema puede definir, a la hora de realizar la consulta, una serie de criterios que le permitan discriminar un subconjunto de carpetas o de boletines de actuaciones del total de carpetas o boletines existentes. La persona que opere el sistema puede realizar una consulta tan selectiva como desee combinando diferentes criterios de filtrado. Esta funcionalidad va a consistir en dotar al sistema de la posibilidad de almacenar durante un determinado tiempo las actuaciones cerradas, con el objetivo de recurrir a ellas si hace falta debido a una posible reclamación de un cliente o para la realización de informes.

La gestión de esta funcionalidad va a consistir esencialmente en:

- **Trasvase de actuaciones.**

  Consiste en el paso de todas aquellas actuaciones cerradas desde hace más de 48 horas desde la máquina de la Bolsa de Trabajo a la máquina de Históricos. Esto va a suponer una mejora en las prestaciones de la Bolsa de Trabajo puesto que va liberar espacio en disco. El paso de las actuaciones cerradas a la máquina de Histórico se lleva a cabo por la noche, de forma automática, para no interferir en el funcionamiento del sistema

- **Consulta de actuaciones.**

  Las consultas al Histórico proporciona al usuario la posibilidad de generar distintos tipos de consultas creando filtros con la casi totalidad de los parámetros que caracterizan a una actuación, como pueden ser la fecha de comienzo y finalización, tipo de actuación, código, etc... .

- **Borrado de actuaciones.**

  La máquina de Históricos tiene unos límites de almacenamiento calculados para dar cabida alrededor de unos doce millones de actuaciones, que vienen a ser el número de actuaciones estimado para un año. Por tanto, transcurrido un año desde que se haya cerrado una actuación, ésta se borrará de la máquina de Históricos, quedando definitivamente eliminada del sistema Gestor de Actuaciones.

El despacho de actuaciones es una de las funcionalidades de mayor importancia dentro del sistema, ya que como su nombre indica, en él va a residir la responsabilidad de asignar o despachar las actuaciones pendientes de resolver procedentes de la Bolsa de Trabajo a los grupos técnicos que se encargarán de su resolución.

Las funciones que se deberán realizar durante la distribución de las actuaciones que hayan entrado al sistema serán, básicamente:

- **la planificación de actuaciones:** propuestas de asignación de las actuaciones pendientes de realización entre los grupos de técnicos existentes.

- **el despacho de actuaciones.**

- **el seguimiento:** debe hacerse un seguimiento de las actuaciones durante el tiempo de vida de estas en el sistema y hasta la finalización de las mismas por parte de los grupos técnicos a los que se encuentran asignadas.

El Sistema Gestor de Actuaciones deberá realizar la asignación de las actuaciones que le lleguen (procedentes de los distintos sistemas corporativos) a los técnicos o grupos técnicos más adecuados para su resolución.

Para decidir qué grupos técnicos son los más idóneos para resolver dichas actuaciones se tendrán en cuenta, entre otros, los siguientes factores:

- **Perfiles de los técnicos.**
- **Localización geográfica de las actuaciones.**
- **Turnos de trabajo.**
- **Calendario laboral.**
- **Sectores asignados a grupos técnicos.**

El proceso de optimización de la asignación de las actuaciones a los grupos técnicos es bastante costoso computacionalmente, por lo que para mejorar el rendimiento del sistema se establece la siguiente estrategia de planificación:

- **Planificación predictiva:** mediante intervención directa de la persona que opera el sistema: este algoritmo realiza una planificación total de todas las actuaciones pendientes y todos los grupos técnicos durante un horizonte de planificación previamente definido.
- **Planificación reactiva:** este mecanismo se aplica con objeto de adaptar la planificación existente (obtenida mediante el algoritmo de planificación predictiva) a los nuevos eventos surgidos en el sistema.

El algoritmo de planificación se puede considerar como el corazón de la funcionalidad de Despacho. Va a ser el encargado de realizar las planificaciones partiendo de unos

datos de entrada que son las actuaciones pendientes de asignación, y los grupos técnicos disponibles en el sistema. Para que el algoritmo desempeñe su objetivo de una forma óptima se ha de configurar, para lo cual existen en el sistema Gestor de Actuaciones unos parámetros de configuración. Otro punto a señalar es la existencia de dos tipos de planificación, la planificación predictiva y la planificación reactiva.

Este algoritmo se basará en lo siguiente:

- **Parámetros de Configuración:** Proporcionan al usuario la posibilidad de ajustar el algoritmo de planificación a los requerimientos del sistema. Este dispone de los mecanismos necesarios para consultar y modificar de una manera sencilla y unificada los siguientes parámetros:

  - ✓ **Parámetros del técnico:** Disponibilidad, horario de comida, costes de la hora extra, coste por inactividad, etc...
  - ✓ **Velocidad para cada tipo de carretera:** urbanas, locales, autónomas, nacionales, autopistas, circunvalaciones, velocidad de campo a través, todas ellas expresadas en Km./h.
  - ✓ **Parámetros de la actuación:** duración, bloqueo en ventana, bloqueo en técnico, incremento por retraso e incremento por actuación no planificada.
  - ✓ **Parámetros de la planificación:** horizonte de planificación, offset por cambio de sector e incremento duración en otro sector, etc.
  - ✓ **Parámetros del algoritmo predictivo:** máximo número de cambio por iteración, máximo número de cambios exitosos, máximo número de iteraciones, umbral de fluctuación, etc...

- **Planificación Predictiva:** Realiza una optimización de los planes de trabajo de los grupos técnicos de acuerdo con las actuaciones de las que se tiene constancia en un momento determinado. Su objetivo es proponer un plan de trabajo dentro de un horizonte de tiempo razonable configurable, por ejemplo, 72 horas, que permita a la persona encargada del despacho de actuaciones tener el grueso del trabajo organizado con

suficiente antelación. El resultado de esta planificación podrá ser modificado con posterioridad si durante el período de vida de las actuaciones dentro del sistema se produce la entrada de nuevos datos que den como resultado la posibilidad de una nueva iteración del algoritmo y, por tanto, una nueva planificación que optimice el empleo de los recursos humanos disonibles.

- **Planificación Reactiva:** Proporciona una ayuda en tiempo real para la resolución de excepciones que dificultan o impiden la cumplimentación de la planificación a largo plazo. Su objetivo se concreta en determinar el mejor grupo técnico para la realización de una nueva actuación o, en general, de una o más actuaciones que, por cualquier motivo, no se han considerado en la última planificación realizada.

- **Duración de la actuación:** las actuaciones suelen ser de larga duración, generalmente más de cuatro horas. Esto implica que muchas de ellas serán realizadas durante varios días.

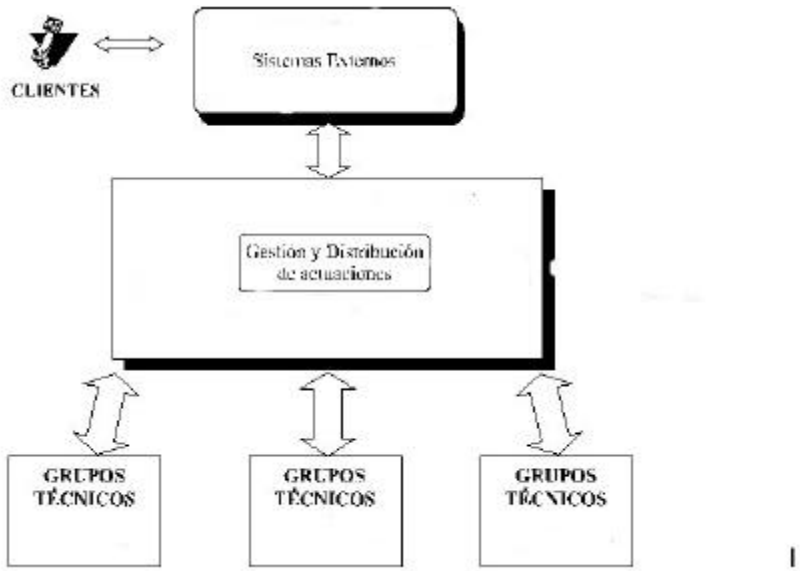
- **Disponibilidad de recursos:** nunca debe haber recursos humanos libres (sin asignación) en el sistema. Existen una serie de trabajos planificados a largo plazo que se pueden ir adelantando en función de que no exista otra actuación o trabajo más urgente pendiente. Asimismo, las actuaciones se deben asignar a los grupos técnicos con independencia de que no se comience la realización de la actuación durante el día actual.

- **Vida de la actuación:** siempre que sea posible, una actuación será realizada en su totalidad por el grupo que la comience, con independencia de las interrupciones que puedan existir.

1.2 Entorno operacional del sistema

El entorno operacional del sistema, entendido como el conjunto de sistemas con los que interactúa y recursos de que dispone, se muestra en la **Figura 6**.

Figura 6.- Entorno operacional del sistema.



Como puede apreciarse, forman parte de este entorno distintos sistemas externos de los cuales nuestro sistema obtiene la información necesaria para su funcionamiento, como la información de localización de la actuación, o las órdenes de poner en marcha las actuaciones para solucionar las peticiones de los clientes. Una vez gestionadas las actuaciones, el sistema las distribuye a los técnicos ubicados por los territorios para que lleven a cabo su resolución. Con el sistema también es posible la comunicación de **ejecutivos** vía WEB (**Web de ejecutivos)** para la petición de informes y estadísticas y de esa forma realizar un seguimiento de la operatividad y eficacia en la realización de las actuaciones (**Objetivos Corporativos**).

Aquí es donde aparece el **CMC**. El **CMC** (**Centro de Mensajes Cortos**) es la forma mediante la cual el sistema se comunica con sus técnicos mediante telefonía móvil cuando éstos ya han abandonado sus puntos de reunión (los puntos de reunión serán aquellos lugares donde los técnicos se reúnan para comenzar la jornada laboral) para transmitirles las actuaciones que deben realizar o cualquier otra información relativa a actuaciones que ya estén en curso. La comunicación se realiza utilizando la funcionalidad de mensajes cortos que ofrecen los teléfonos móviles digitales. Estos son utilizados por los técnicos propios para la recepción de actuaciones asignadas automáticamente a ellos y el envío de franqueos. Para que esta transferencia de información pueda llevarse a cabo se crea un canal de comunicación mediante el intercambio de mensajes cortos entre el sistema, el **Centro de Mensajes Cortos de**

Movistar (CMC) y los terminales móviles. El estudio de la forma de comunicarse entre el sistema y los técnicos y las propuesta de soluciones alternativas es la base de la realización de este proyecto, por lo cual, a medida que vayamos avanzando, se irá profundizando cada vez más en el tema.

1.3 Web de técnicos

El objetivo del proyecto realizado se centra en esta parte del sistema. El **Web de Técnicos** es una aplicación WEB, englobada en el total del sistema, que permite recuperar información, basada en unos criterios de consulta específicos, acerca de los boletines de actuaciones almacenados en la base de datos del sistema. A su vez, los técnicos propios tienen la posibilidad de franquear o cumplimentar las actuaciones que hayan realizado. Una actuación está **franqueada o cumplimentada** cuando se ha establecido o restablecido el servicio solicitado por el cliente. Esto implica la comunicación con el sistema comercial correspondiente y la próxima finalización de la actuación. En este estado sólo resta cerrar la actuación informando de la actividad para que quede constancia de todo lo realizado por el técnico en la base de datos del sistema (tiempo empleado en la resolución, material utilizado, kilómetros desplazados hasta el domicilio del abonado,...). Una actuación se considerará **cerrada** cuando toda la información remitida por los técnicos al sistema haya sido supervisada y validada por este, evitando de esta forma cualquier tipo de inconsistencia en la base de datos del sistema. Una vez cerrada, los datos relativos a esa actuación se almacenarán para posteriores consultas de históricos y generación de informes, pero la actuación en si, considerada como actividad a realizar por los técnicos, saldrá del sistema al ser considerada como resuelta. Una actuación también podrá llegar a este estado a través de la intervención de cualquier usuario del sistema autorizado para ello, para resolver los casos especiales en los cuales dicha actuación deba de salir del sistema sin la intervención especifica de ningún técnico.

La información a la que tienen acceso sus usuarios (técnicos), es aquella relacionada con las actuaciones de sus respectivos grupos técnicos.

La comunicación con los técnicos es un elemento muy importante dentro de las funcionalidades del sistema puesto que ellos son los responsables de la resolución de las actuaciones. Por tanto el sistema debe incorporar los mecanismos necesarios para que la comunicación con ellos se realice de una forma fluida. La comunicación se venía realizando vía telefónica convencional (para lo cual era necesaria la presencia de un despachador que intercambiara información con los técnicos), para los técnicos que no dispongan de móvil GSM, y vía GSM (mensajería corta de los teléfonos móviles digitales), que aporta un notable avance en la productividad del sistema al reducir numerosas tareas que no aportaban valor al proceso, permitiendo tanto la asignación de

actuaciones a los técnicos como el franqueo de las actuaciones de una forma mucho más directa con el sistema. Este último procedimiento de comunicación permite la automatización total del proceso **recepción – asignación - resolución de actuaciones**

El objeto del proyecto que se ha desarrollado es la inclusión en el sistema de nuevas formas de comunicación con los técnicos vía **Internet**. Esto es posible gracias a las nuevas tecnologías surgidas que permiten el acceso directo al Web de Técnicos en Internet usando la red **GSM**. Estas nuevas formas de comunicación plantean numerosos problemas debido a que GSM es una red que no ha sido diseñada para el tráfico de datos y, sobre todo, a que dispone de un ancho de banda muy limitado. Pero la automatización total de la funcionalidad del sistema hace que sea rentable emplear recursos en el desarrollo de esta aplicación debido, principalmente, a la posibilidad de desaparición de la figura del despachador y el consiguiente ahorro para la empresa.

Desde los llamados puntos de reunión (son aquellos lugares donde los técnicos se reúnen para comenzar la jornada laboral y donde disponen de un punto de acceso al Web de Técnicos a través de la red telefónica básica) los técnicos pueden comenzar la jornada laboral conociendo la planificación que existe para ellos de las actuaciones y, sobre todo, la primera actuación de la que han de ocuparse. La comunicación entre el sistema y los técnicos cuando estos han abandonado esos puntos de reunión se realiza a través de la funcionalidad de mensajería corta de los teléfonos móviles digitales. Esta forma de comunicación con los técnicos plantea numerosos problemas que hacen que sea poco eficiente como más adelante se expondrá.

Las funcionalidades que ofrece el sistema Gestor de Actuaciones respecto al servicio dado por el Web de Técnicos se pueden resumir en los siguientes puntos:

- **Control de acceso y de usuarios del servidor Web:** El sistema realizará un control de acceso mediante el requerimiento del nombre de usuario y de la palabra clave correspondiente. El sistema informará al usuario si su identificación ha sido correcta por medio de un mensaje de bienvenida, o si por el contrario, el intento de acceso ha sido denegado.

  Una vez dentro, el sistema determina los permisos que tiene del usuario para acceder a determinadas páginas **html**, que variarán de unos usuarios a otros.

- **Configuración del servidor Web:** El sistema dispone de una serie de parámetros configurables los cuales controlarán:

  - ✓ **La asignación de actuaciones por parte de los usuarios (manual o automática).**

Por defecto este parámetro estará en automático. Indicará que el propio sistema asignará la siguiente actuación que se encuentre planificada para el usuario. Este proceso será ejecutado cada vez que el usuario realice el franqueo (averías) o cumplimentación (instalaciones) de una actuación, de tal forma que el sistema ejecutará el franqueo o cumplimentación y seguidamente asignará la siguiente actuación pendiente planificada. El proceso de asignación automática será iniciado cuando el técnico haya culminado la actuación que posee actualmente en curso, es decir, cuando realice el franqueo o cumplimentación de la actuación. En este caso, al usuario no se le mostrará la operación de **auto-asignación**.

Cuando esté configurado como manual, el usuario podrá auto-asignarse actuaciones que se encuentren en estado planificadas, por lo que se le mostrará dentro de sus operaciones la de **auto-asignar**. También se debe configurar el tipo de usuario al cual se le da permiso de auto-asignación. Los usuarios que pueden tener este permiso son los grupos técnicos y los técnicos individuales, pudiendo ambos a la vez realizar esta operación dependiendo de los requerimientos del servicio.

- ✓ **Las actuaciones planificadas que el usuario podrá observar.**

  Los usuarios del sistema siempre podrán observar la actuación que se encuentre actualmente en curso, y N actuaciones planificadas, siendo **N>=0**. Por defecto, el parámetro estará a 0, pudiendo ver únicamente la actuación que tiene en curso, siempre y cuando el parámetro de asignación esté en estado automático. En el caso en que el parámetro de asignación esté en estado manual, la N deberá estar por defecto a uno ya que al menos debe poder ver la siguiente actuación que tiene pendiente planificada para poder auto-asignársela.

- **Gestión de las actuaciones por parte de los usuarios (técnicos y contratas), a través del Web:** El sistema permite realizar ciertas operaciones a sus usuarios según qué tipo de usuario sean y según el estado de los parámetros de configuración:

- ✓ **Pueden realizar la operación de auto-asignar.**

  De acuerdo al parámetro configurable del sistema que les permitirá o no realizar esta operación. Así, si este parámetro está activo el usuario tiene permisos para realizarla, de estar inactivo no le aparecerá dentro de la lista de operaciones que puede realizar.

- ✓ **Puede ver el detalle o el resumen de las averías e instalaciones que le sean asignadas.**

  Dependiendo de la información en la que esté interesado puede visualizar solo los datos más importantes de una actuación o, por el contrario, tener una visión completa de los mismos.

- ✓ **Puede franquear y cumplimentar actuaciones, siempre y cuando estas estén en curso.**

  Solo se podrá realizar un franqueo por actuación. También podrá generar **Informes de Actividad**, que son los mecanismos mediante los cuales los técnicos reflejan la evolución y el estado de una actuación, y que son reportados al sistema.

- **Generación de Informes:** Cuando una actuación es franqueada, se realiza automáticamente un informe de actividad que también podrá ser ejecutado por el usuario cuando lo considere pertinente.

1.4 Arquitectura software

El producto software de este sistema constituye una aplicación distribuida compleja, basada en el sistema operativo **UNIX** y **Microsoft Windows**, estando formada por componentes que ofrecen la funcionalidad mencionada en párrafos anteriores. De acuerdo con el modelo de implementación que se describe en la sección siguiente, los componentes indicados son, mayoritariamente, procesos **UNIX** y aplicaciones **MS-Windows**.

Se presenta a continuación una visión de conjunto del software que constituye la aplicación. En adelante nos referiremos al **Sistema Gestor de Actuaciones** como **GA**.

Vamos a distinguir los siguientes módulos:

- **Núcleo GA:** Constituye el elemento principal del sistema. Este componente está implementado en su mayor parte a partir del producto software **ARS** (**Action Request System**) de la empresa **Remedy Corp**., el cual ofrece facilidades de gestión de la Bolsa de Trabajo, gestión de los recursos humanos y gestión de actividad. Además formando parte del Núcleo GA, fuera de **ARS**, se encuentra el algoritmo de planificación implementado en **C++**. También y gracias al producto **Ilog** y mediante diagramas **Gantt** se permite visualizar de una forma intuitiva la planificación de los trabajos y los técnicos asignados a cada actuación.

- **Sistema de Interfaces:** También aparece, como elemento diferenciado, el módulo software que se encarga de las interfaces del GA con los distintos sistemas Comerciales (**Sistema de Interfaces**).

- **Interfaz SETM:** El sistema Gestor de Actuaciones ofrece un módulo de interfaz específico para comunicarse con el sistema del Área Comercial **SETM** (este módulo aparece a parte del resto de las interfaces por las peculiares características de antigüedad que presenta el sistema **SETM**).

- **Informes:** El GA dispone de una base de datos independiente que posibilita la generación de informes con la herramienta **Bussiness Objects**.

- **WEB:** También el GA dispone de software **WEB**. El **Web de ejecutivos** permite obtener información en tiempo real del estado de los boletines de actuaciones a los directivos de la empresa autorizados para ello. El **Web de Técnicos** permite diversas funciones para los empleados y entre ellas la de franquear y cumplimentar actuaciones.

En capítulos posteriores se describe con más detalle los aspectos más importantes de la arquitectura software del Sistema Gestor de Actuaciones.

1.5 Modelo de Implementación

Debido a la diferente naturaleza de los elementos que conforman el Gestor de Actuaciones, este sistema emplea varios modelos de implementación. Algunos de dichos elementos constituyen productos comerciales que ha sido necesario integrar o completar con desarrollos adicionales, empleándose en tales casos las facilidades de programación que ofrecían éstos.

En algunos casos, la implementación sigue el paradigma del diseño y programación estructurada, con llamadas a funciones que exportan otros componentes, o mediante mecanismos generales de comunicación entre procesos (RPCs, colas, ...).

Básicamente, los modelos de implementación que se han empleado en el desarrollo del sistema Gestor de Actuaciones han dado lugar a procesos que se comunican de diferente manera a la hora solicitar y ofrecer servicios. Entre éstos, cabe destacar:

- **Comunicación vía APIs.**

  Este tipo de mecanismos se emplea para invocar servicios o funciones de otros procesos. En el GA se hace un uso extensivo de este mecanismo en la comunicación con los componentes **Gestor de Esquemas** y **Notificador** del Núcleo GA.

- **Comunicación vía ficheros**.

  Este procedimiento se emplea sobre todo en la comunicación con los sistemas comerciales de acceso a clientes, tales como **AF**, **MIGA**, **SINPLEX** y **PX**.

- **Comunicación vía DSO.**

  Este mecanismo de comunicación es empleado por la interfaz con el sistema **GRI** (**Gestor de Reclamaciones e Incidencias**), que se apoya en **RPC** y entre las bolsas de trabajo.

- **Comunicación vía CGIs.**

  Mediante este procedimiento se comunica a determinados procesos de la funcionalidad de **Consultas e Informes** la información que se solicita desde los navegadores **WEB**, que acceden al servidor a través del protocolo **HTTP**. Es este modelo el que más nos interesa debido a

su relación directa con el proyecto elaborado. En el acceso al Web de Técnicos empleando terminales móviles, la generación dinámica de las páginas "**html**" y "**wml**" optimizadas para dichos terminales, las cuales contendrán la información solicitada por los técnicos, se realizará mediante la ejecución en el servidor de los correspondientes **CGIs,** los cuales efectuarán los accesos a la base de datos del sistema para introducir o extraer información y tendrán como salida el código "**html**" o "**wml**" correspondiente a cada caso particular.

Todas las comunicaciones relacionadas con **X.25**, **Red de Área Loca**l **Ethernet**, **RTC/RDSI** y **punto a punto** se resuelven mediante el uso de software comercial.

El nivel **RPCs** ofrece a los niveles superiores un mecanismo de comunicación basado en llamadas remotas a procedimientos. Este nivel es empleado tanto por el producto **ARS** de **Remedy Co.**, como por los procesos que implementan la interfaz del GA con los sistemas del Área Comercial. El nivel **SQL*Net** proporciona facilidades a los procesos de aplicación para el acceso a bases de datos remotas **ORACLE**.

El nivel **HTTP** permite el acceso mediante visualizadores Web a determinada información ejecutiva que mantiene el GA. Los niveles inferiores que tienen que ver con **X.25** son empleados para comunicar los sistemas **Hosts**, proveedores de información, con el Núcleo Central GA. Por otro lado, los niveles inferiores de **Ethernet** se emplean para comunicar los distintos elementos que forman parte de la instalación central del GA (routers, impresoras, servidores del GA,..). Por último, los niveles **PPP** y **RTC** se utilizan para comunicar vía **RTC/RDSI** ciertos Clientes GA con el Núcleo Central GA.

La mayor parte de las interacciones entre los componentes que conforman la funcionalidad del sistema siguen los modelos conocidos como **cliente/servidor** y **Web**. De acuerdo con el modelo cliente-servidor las aplicaciones se organizan de forma que el servidor contiene la parte de funcionalidad que debe ser compartida por diversos usuarios, mientras que en el cliente permanece sólo la particular de cada usuario. En cuanto a las comunicaciones vía Web los usuarios tendrán instalado un navegador en sus puestos mediante el cual accederán a los servidores WEB correspondientes.

Los clientes realizan, generalmente, funciones tales como:

- **Manejo de la interfaz de usuario.**
- **Captura y validación de los datos de entrada.**
- **Generación de consultas e informes sobre bases de datos remotas.**

Por su parte, los servidores realizan, entre otras, las siguientes funciones:

- **Ejecución y gestión de la lógica de la aplicación.**
- **Control de accesos concurrentes a bases de datos compartidas.**
- **Gestión de periféricos compartidos.**
- **Interacción con otros sistemas.**

En el caso de nuestro sistema, la lógica de los procedimientos de gestión de boletines, los procesos de interfaz con sistemas externos y procesos auxiliares Unix se ejecutan en el Núcleo Central del mismo.

Los procedimientos de interfaz con los distintos Sistemas Corporativos se ejecutan en el modulo de Interfaces.

Posteriormente se describirán con más detalle los aspectos más importantes de la arquitectura software en la que se basa el sistema.

1.6 Bases de datos

La mayor parte de la funcionalidad ofrecida por el sistema se apoya en el mantenimiento de grandes cantidades de información relativa a averías, instalaciones y trabajos programados (todo englobado bajo el nombre general de actuaciones), así como los datos auxiliares necesarios para su gestión (disponibilidad de técnicos, cualificación de cada técnico, situación geográfica de los técnicos, etc...). Este hecho obliga a contar con una adecuada infraestructura para el almacenamiento, actualización y recuperación de dicha información que, en nuestro caso, se ubica principalmente en el Núcleo Central. Allí, se encuentra instalado un sistema de gestión de bases de datos relacionales de la firma ***ORACLE***.

1.7 Procesos del Sistema Histórico

El sistema de **Históricos** es la parte dentro de nuestro sistema encargada del almacenamiento de los históricos de las actuaciones, entendiendo como tales a toda aquella información relacionada con las actuaciones que lleven cerradas en el sistema más de un tiempo determinado.

Para proceder a la extracción y duplicación de esta información se utiliza la herramienta **DSO** (**Distributed Server Option**), propia de **ARS**, y orientada a la copia de esquemas entre servidores distribuidos.

El proceso va a consistir en comprobar diariamente, por medio de un procedimiento automático, qué actuaciones llevan cerradas en el sistema más de un tiempo determinado, de forma que todas que cumplan dicha condición serán trasladadas al sistema de Históricos, con toda su información relacionada. El tiempo mínimo para que una actuación cerrada pase al Histórico es configurable y suele ser de 24 horas.

Este sistema de Históricos se crea para disponer de un lugar donde se encuentre almacenada toda la información relativa a actuaciones y que puede ser de utilidad en el futuro para atender posibles reclamaciones de los clientes o para la generación de informes y estadísticas. Además, al ser almacenada dicha información en un módulo independiente del sistema, se consigue ir liberando a este de mucha de la carga que soporta y así conseguir un rendimiento mayor del mismo.

La base de datos en la que se almacena el Histórico es una **BD Oracle** que por encima va a llevar, en la capa inmediatamente superior, **ARS** (servidor de aplicaciones).

1.8 Procesos de Backup

Como en todo sistema, el hacer copias de seguridad de los datos es un tema de vital importancia para asegurar el correcto funcionamiento del mismo ante posibles perdidas de información. El sistema Gestor de Actuaciones dispone de un sistema de Backup gestionado por un software comercial adecuado y que será el encargado de realizar las copias de seguridad, extrayendo de la base de datos del sistema toda la información considerada fundamental para poder restablecer posteriormente el funcionamiento correcto sin pérdida de datos.

El proceso de extracción de datos se realizará de forma que no suponga una disminución de rendimiento del sistema al tener que acceder a las bases de datos del mismo. Para

ello el Núcleo Central del GA se encargará de lanzar unas "**macros"** que realizan estas funciones de backup y que estarán programadas para ejecutarse a aquellas horas de menor carga del sistema, es decir, en horario nocturno, en el cual apenas existen accesos de usuarios al sistema ni entrada de nuevas actuaciones al mismo.

2. PLANTEAMIENTO DEL PROBLEMA Y OBJETIVOS

Es objeto de este capítulo el presentar el problema que se pretende resolver con la elaboración de este proyecto. Se intenta dar una visión del problema desde el punto de vista de la eficiencia del sistema y plantear unos objetivos que lleven a la eliminación del problema en cuestión.

2.1 Eficiencia actual en la comunicación con los técnicos

Como ya se ha comentado antes, este proyecto se apoya en un sistema ya existente de optimización en la resolución de averias e instalaciones telefónicas por parte de personal técnico cualificado. Más concretamente, el proyecto se centra en el acceso a la información existente relativa a actuaciones por parte del personal técnico para obtener los datos que le son necesarios para la realización de su trabajo y el posterior envío al sistema de los datos referentes al trabajo realizado para que este pueda dar por finalizada esa actuación y así poder gestionar con eficiencia los recursos humanos disponibles.

Esta comunicación entre sistema y técnicos se hacía, en su origen, a través de unos operadores (personal físico de la empresa) que se conocían como **despachadores**. Estos despachadores eran los encargados de analizar los resultados que ofrecía el proceso de planificación del sistema y, en base a estos, ponerse en contacto con los técnicos correspondientes para comunicarles todo aquello referente a la actuación que debían llevar a cabo. Estos mismos técnicos, una vez realizado su trabajo, se ponían en contacto con los despachadores para comunicarles las incidencias y todo lo que estuviera relacionado con la resolución de la actuación (material utilizado, kilómetros desplazados, tiempo empleado,...). Todo este proceso se producía a través de comunicaciones telefónicas de voz entre el despachador y los técnicos. Los técnicos tenían el inconveniente de tener que tomar nota de los datos sobre la actuación que le comunicaba el despachador y, estos, a su vez, debían tomar nota de toda la información de resolución de la actuación que les comunicaban los técnicos. Además, los despachadores debían ocuparse también de introducir todos estos datos manualmente en el sistema. Por tanto, el despachador era una figura imprescindible en todo este proceso.

Posteriormente se pensó en dotar al sistema de las herramientas necesarias para que todo este proceso fuese automático y se pudiese prescindir de la figura del despachador, con lo cual estos recursos humanos de la empresa iban a poder ser utilizados para otras funciones.

Para ello, se debía dar al sistema la posibilidad de poderse comunicar directamente con los técnicos, sin ningún intermediario y, a su vez, la posibilidad a los técnicos de comunicarse directamente con el sistema. La solución para esto se encontró en los mensajes de texto de los terminales móviles digitales que operan a través de la red GSM.

Para que esta idea pudiera llevarse a cabo debía crearse una **interfaz** entre el sistema **Gestor de Actuaciones** y el **Centro de Mensajes Cortos** de **Movistar**.

Una vez creada esta interfaz, llegamos a que la comunicación con los técnicos se produce en la actualidad a través del servicio de mensajería corta de telefonía móvil. Cuando a un técnico se le asigna una actuación, recibe en su teléfono móvil un aviso de mensaje recibido. En este primer mensaje el sistema le envía los datos sobre la avería que tiene que reparar o sobre la instalación que debe realizar (titular, teléfono, domicilio, población, provincia, central, situación de cajas terminales, cita previa con el cliente, tipo de aviso recibido, posibles equipos a instalar, observaciones…). Podemos observar que la cantidad de información que es necesario hacer llegar al técnico es elevada por lo que, debido a la limitada capacidad de los mensajes de texto, no será posible incluir toda la información en un solo mensaje. Este es el primer problema que surge al usar esta forma de comunicación con los técnicos. Será necesario el envío de un elevado número de mensajes al móvil del técnico y éste deberá disponer de la suficiente capacidad en su memoria interna para almacenarlos y así poder consultarlos cuando lo necesite.

Los técnicos usan terminales móviles cuyas tarjetas **SIM** han sido reprogramadas para que ofrezcan a sus usuarios (técnicos) dos funcionalidades:

- **Capacidad de almacenamiento de datos.**

  Las tarjetas llevan un pequeño módulo de memoria que posibilite a los técnicos almacenar toda la información sobre la actuación que deben resolver y que les envía el sistema. Además deben disponer de capacidad suficiente de memoria para almacenar los datos que necesiten para crear los menús de selección que necesitan los técnicos para franquear las averías.

- **Capacidad de procesamiento de datos.**

  Disponen tanbién de un pequeño microprocesador integrado en la tarjeta SIM suficiente para ejecutar una pequeña aplicación que realiza dos tareas diferentes:

  - ✓ **Decodificación.**

    Por un lado recoje todos los mensajes que llegan al terminal móvil con información relativa a la actuación y la almacena en un formato visualizable para el técnico.

  - ✓ **Codificación.**

    Por otro lado crea una serie de mensajes de texto a partir de los datos que introduce el técnico a mano, rellenando formularios o seleccionando en menús y los envía al sistema para su procesamiento.

Una vez el técnico ha reparado la avería o instalado el equipo correspondiente deberá enviar al sistema todos los datos que este necesita para dar por finalizada esa actuación (datos de desplazamiento, tiempo empleado, equipo utilizado, etc…) y así poder asignarle al técnico la siguiente actuación que tenga planificada. Parte de estos datos los debe introducir el técnico manualmente y el resto deberá seleccionarlo de unos menús existentes. Como ya se ha comentado, esto obliga a programar la tarjeta SIM de los teléfonos móviles de todos los técnicos para que dispongan de una aplicación que genere los mensajes de texto con los datos correspondientes que deben enviar al sistema y para que dispongan de una base de datos con los posibles equipos utilizados o instalados de forma que dicha aplicación pueda generar los menús donde el técnico realizará la selección los mismos. Aquí surge el segundo problema existente y, también, el más grave. Cualquier cambio que se decida realizar sobre esta aplicación (cambio en los datos requeridos por el sistema para la gestión de las actuaciones, variación en la capacidad de transporte de los mensajes de texto, cambios en el código de la aplicación para ofrecer nuevas funcionalidades, etc…), así como todos aquellos cambios que el desarrollo tecnológico produce con cierta frecuencia sobre la base de datos de posibles equipos utilizados o instalados por el personal técnico, obligan a recoger las tarjetas SIM de los teléfonos móviles de todos los técnicos y a reprogramarlas.

Un último problema, pero igual de importante que los anteriores, es que el servicio de mensajería corta de texto que ofrecen los operadores de telefonía móvil digital es un **servicio no fiable**. No existe un acuse de recibo por parte de estos terminales de la recepción del mensaje, por lo cual el Centro de Mensajes Cortos podría comunicar a

nuestro sistema, a través de la interfaz creada, el envio de los mensajes y estos no ser recibidos, lo cual crearía una situación de inconsistencia en el sistema. Este último se quedaría esperando alguna comunicación por parte del técnico sobre el desarrollo de su actividad en la resolución de la actuación enviada, y este último estaría esperando una comunicación por parte del sistema de su próxima tarea, con lo cual se violaría una de las premisas principales seguidas a la hora de desarrollar esta aplicación, la cual consistía en evitar a toda costa que los recursos humanos de la empresa estuvieran parados.

De todo lo anterior se deduce que la eficiencia del actual sistema de comunicación con el personal técnico es baja.

Como una primera solución se desarrolla un **Web de Técnicos** a través del cual los técnicos puedan comunicarse con el sistema usando Internet. Las funcionalidades que obtenemos con las páginas WEB creadas para ello ya han sido comentadas con anterioridad. Mediante este Web es posible una conexión **on-line** con el sistema y de esta forma poder disponer en todo momento y en **tiempo real** de todas las actualizaciones que se efectúen sobre la base de datos del sistema, ya sea referentes a actuaciones asignadas a técnicos o referentes a posibles equipos instalados o material utilizado. Las ventajas de una conexión on-line con la base de datos del sistema se analizarán más adelante en profundidad.

Esta primera solución nos resuelve los problemas que se han comentado pero nos origina uno todavía más importante si cabe, el de la comunicación entre el sistema y los técnicos cuando estos últimos no disponen en susproximidades de una estación de trabajo con un navegador que les permita la conexión al sistema a través de la red Internet. Como vía de comunicación con el sistema ofrece muchas posibilidades para gestión y creación de informes y estadísticas por parte de ejecutivos, pero como solución para nuestro problema de comunicación con los técnicos es claramente insuficiente.

2.2 Objetivos

Se plantean posibles soluciones encontradas a los problemas existentes y la eficiencia de las mismas, para así poder llegar a la conclusión de cual es la que mejor se adapta a las características de nuestro sistema.

El objetivo que nos marcamos consiste en un cambio del proceso de comunicación del sistema con los técnicos. De todo lo analizado en la sección anterior se desprende que la solución ideal a nuestro problema saldría de crear un **Web de Técnicos** "**móvil**". Se mantendrán las páginas WEB creadas con anterioridad para cuando se desee acceder al sistema con herramientas poderosas de navegación y procesamiento de datos que

podemos encontrar en un **PC** (páginas WEB desarrolladas en código **html**, **javascript**,...). Pero para una solución efectiva de nuestro problema se usarán las nuevas tecnologías emergentes en telefonía móvil digital, las cuales nos proporcionan recursos para acceder a **Internet** a través de la red de telefonía móvil **GSM**. Esto nos va a obligar a desarrollar un Web de Técnicos "**paralelo**", con páginas WEB optimizadas para acceder a ellas usando terminales móviles de última generación que, aunque ofrecen esta posibilidad, están muy limitados en cuanto a potencia del **navegador** que llevan integrado y en cuanto a capacidad de procesamiento de datos del microprocesador que traen también integrado.

Usando estas nuevas tecnologías será posible establecer conexiones en '**tiempo real**" con las bases de datos del sistema de tal forma que quedarán resueltos nuestros problemas de comunicación con los técnicos al poder disponer estos últimos "**on-line**" de toda la información que necesiten correctamente actualizada. Cualquier cambio que se produzca en la base de datos del sistema será efectivo en el teléfono móvil del técnico en el instante en que dicho técnico solicite la correspondiente página WEB a nuestro servidor.

3. CONCEPTOS TEÓRICOS

En este capítulo se hace una pequeña introducción teórica a todos aquellos aspectos del sistema que tienen especial interés en la elaboración del proyecto (herramientas comerciales, estándares internacionales,...).

3.1 Herramienta ARS

La herramienta **ARS** (**Action Request System**) de **Remedy Corp.** es el componente que soporta la mayor parte de la funcionalidad que ofrece el Núcleo del Sistema Gestor de Actuaciones (gestión de la Bolsa de Trabajo, de recursos humanos, de actividades, ...). ARS es un sistema de aplicaciones (servidor de aplicaciones) con arquitectura **cliente/ servidor** y comunicaciones **TCP/IP**, que se apoya para realizar sus funciones en una base de datos o conjunto de bases de datos distribuidas. Los servidores de **ARS** pueden ejecutarse sobre las plataformas **UNIX** de los suministradores más comunes (**HP**, **Sun**, **IBM**, ...) y sobre **WINDOWS NT** (**Intel**, **Digital**, ...), mientras que los clientes de **ARS** pueden ejecutarse sobre estaciones de trabajo de estos mismos suministradores y sobre plataformas PC. Asimismo, **ARS** puede emplear gestores de bases de datos relacionales de múltiples suministradores (en nuestro caso concreto de la marca **Oracle**), o en su defecto, también puede utilizar ficheros de texto con estructura plana.

ARS también puede definirse como un **servidor de Workflow**. Esto es así ya que es la herramienta encargada de supervisar el ciclo de vida de una actuación en el sistema (**flujo de trabajo**). Seguirá los pasos previamente definidos para posibilitar que, desde la entrada de la actuación en el sistema, esta siga una evolución coherente a través de los posibles estados en los que pueda encontrarse y este proceso finalice con la salida de la actuación (cierre de la misma), después de las correspondientes actualizaciones que le sea necesario realizar sobre la base de datos del sistema.

ARS es una aplicación cliente/servidor fácilmente adaptable a numerosas situaciones, lo cual da la posibilidad a los profesionales de los sistemas de información de automatizar un amplio rango de procesos de **soporte al negocio** (operaciones internas). Esta herramienta está específicamente diseñada para consolidar los procesos que mantienen a la infraestructura de una organización operando a niveles óptimos de rendimiento.

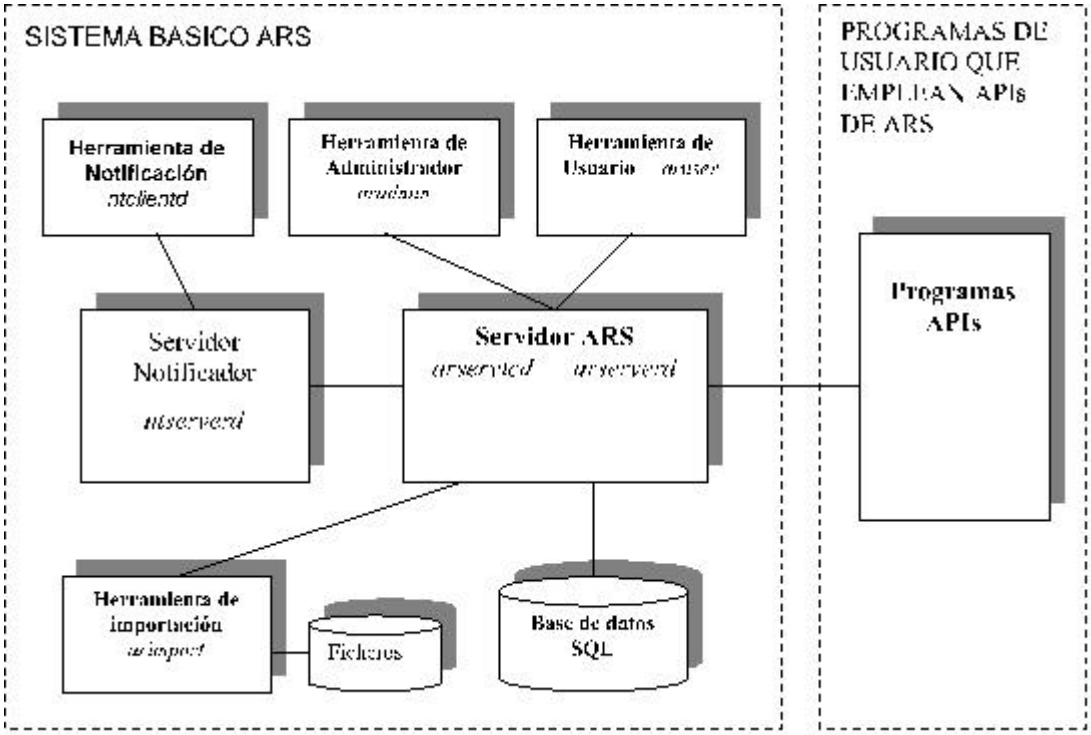
Uno de las principales características de **ARS** es que es rápidamente adaptable a las características particulares de cada organización individual con sus propias necesidades de negocio. Nos ofrece un conjunto muy completo de **APIs** (posibilidades de

comunicación e intercambio de datos con el resto de herramientas y aplicaciones software que conforman el sistema completo).

Una **AR** (**Action Request**) es una información que describe un evento simple o un incidente, como podrían ser un problema determinado o una solicitud de servicio. Cada una de las solicitudes de acción representan una entrada simple en una base de datos de un sistema de AR. Las actividades de estos sistemas incluyen la actualización de esquemas, registro de entrada de solicitudes de acción, consultas a la base de datos del sistema para supervisar el progreso de una determinada petición, así como la resolución de dichas solicitudes. Como ya se ha comentado, una posible definición de **ARS** que se ajusta muy bien a las funciones que lleva a cabo dentro de un sistema, es la de **Servidor de Workflow,** ya que su objetivo final es la de realizar un seguimiento de flujos de trabajo (en este caso ciclos de vida de las actuaciones en el sistema).

En la **Figura 7** siguiente podemos ver como se organiza una aplicación de usuario que hace uso de esta herramienta.

Figura 7.- Herramienta ARS.



En su contexto más general, **ARS** facilita la organización y seguimiento de la información y de los procesos que forman parte de un flujo de trabajo. En el caso de nuestro sistema, ARS es el componente que permite organizar y seguir el proceso de resolución de las averías e instalaciones.

El sistema **ARS** incluye en su configuración básica sobre plataforma **Unix**, tal como muestra la figura anterior, los siguientes tipos de programas y procesos:

- **Servidor ARS.**

  Este es el elemento sobre el que recae la responsabilidad de controlar los diferentes procesos que están definidos para realizar una tarea. Entre sus funciones está el vigilar que las actividades que componen la tarea se realizan en el orden correcto y por las personas o instancias interesadas, así como el que se use la información necesaria en cada una de sus fases.

  El servidor **ARS** va a estar constituido, principalmente, por dos procesos:

  - ✓ **arservtcd:** es el que está encargado de gestionar los aspectos dinámicos de los procesos.
  - ✓ **arserverd**: se encarga de todos los aspectos relacionados con la información (datos).

- **Herramienta de Administración.**

  Este elemento permite a ciertos usuarios autorizados realizar las labores propias de administración de **ARS** (creación de nuevos formatos de boletines, creación de nuevos usuarios, gestión de permisos, ...). La herramienta de Administración está constituida por el proceso **aradmin**.

- **Herramienta de Usuario.**

  Este elemento es el que emplean de manera habitual los usuarios de **ARS**. Mediante él introducen información en el sistema (boletines) y consultan la información que éstos contienen. La herramienta de Usuario está compuesta por el proceso **aruser**.

- **Herramienta de Notificación**

  Esta herramienta implementa un mecanismo de **notificación** de **eventos** y **mensajes**:

  - ✓ **eventos:** los eventos están asociados, de manera general, a sucesos que ocurren durante la realización de todo tipo de tareas.
  - ✓ **mensajes:** los mensajes constituyen una forma de comunicación entre las instancias involucradas en dichas tareas.

  La herramienta de **Notificación** está compuesta por dos procesos, "**ntserverd**" y "**ntclientd**":

  - ✓ **ntserverd:** es el proceso servidor de notificaciones. Se ejecuta en la misma máquina en la que éste funcionando un servidor **ARS**.
  - ✓ **ntclientd:** es el proceso que se encarga de recibir y presentar las notificaciones. Puede ejecutarse en la misma máquina en la que esté funcionando una herramienta de Usuario, o bien de manera independiente en cualquier otra máquina.

- **Herramienta de Importación**

  Esta herramienta, constituida por el proceso "**arimport**", permite a sus usuarios, generalmente los administradores del sistema, importar dentro del servidor **ARS** datos de boletines. Dichos datos se encontrarán habitualmente en formato de fichero normalizado **ARS**, generado mediante la herramienta de Administración durante los procedimientos de copia de seguridad o como copia procedente de otro sistema **ARS**. También es posible la introducción de datos a partir de ficheros con formato de texto.

La **Figura 7** recoge asimismo las aplicaciones que pueden crearse en torno a **ARS**. Dichas aplicaciones interaccionan con el servidor de **ARS** mediante la utilización de las **APIs** que ofrece esta herramienta. La funcionalidad de **ARS** que está accesible a través de estas **APIs** es equivalente a la funcionalidad que proporcionan los comandos de las herramientas de Usuario y Administración.

La herramienta **ARS** permite también mejorar las prestaciones del servidor **ARS** mediante una opción multiproceso (**Multi-Process Server Option**), que no viene incluida en la configuración básica y para la que se requiere un contrato de licencia adicional. Mediante esta opción, la funcionalidad que implementa el proceso "**arservd**" se divide entre un conjunto de procesos especializados en tareas particulares.

Los procesos que constituyen el servidor **ARS** con opción multiproceso son:

- **arservtcd.**

  Este proceso, que también está presente en la configuración básica, se encarga de gestionar el resto de procesos del servidor, así como de equilibrar sus niveles de carga.

- **Servidor de administración.**

  Este proceso puede invocar cualquier operación que se lleve a cabo dentro de **ARS**. En concreto, el servidor de administración ejecuta todas las operaciones que tienen que ver con la reestructuración de esquemas (formatos de boletines), garantizando la **serialización** e **integridad** de todas las operaciones de reestructuración. Este proceso se encarga también de gestionar las peticiones que proceden de las herramientas de Usuario de **ARS** de versiones anteriores a la versión **ARS 2.0**.

- **Servidores rápidos.**

  Estos servidores gestionan todas las operaciones rápidas de **ARS**, tales como las peticiones de introducción de boletines nuevos, consultas, etc.

- **Servidores de listados.**

  Estos procesos están encargados de servir aquellas peticiones relacionadas con la exportación de datos. Dependiendo del tamaño de la solicitud serán operaciones mas o menos rápidas (volumen de datos).

- **Servidor de escalaciones.**

  Este proceso gestiona aquellos eventos que tienen que ver con sucesos relativos a intervalos de tiempo.

- **Servidores privados.**

  Estos procesos están dedicados a determinados usuarios. El proceso "arservtcd" no controla estos procesos, por lo que necesitan ser arrancados y detenidos de manera manual por los administradores del sistema.

Además de los procesos anteriores, existen otros procesos asociados a una capacidad de **ARS** relacionada con la conectividad de varios servidores **ARS**. Dichos procesos, que se recogen bajo la denominación de Opción de Servidores Distribuidos (**Distributed Server Option**), permiten ciertos tipos de transparencia de ubicación y replicación de los boletines en los diferentes servidores **ARS** de un sistema **multiservidor**.

3.2 Interfaz con el Centro de Mensajes Cortos

Nuestro Sistema Gestor de Actuaciones presenta numerosas interfaces con muchos otros sistemas externos, ya que es un sistema que se basa para realizar su función en la información que recibe proveniente de los sistemas comerciales de acceso a clientes. Introducimos brevemente la interfaz con el **CMC** por ser la que nos interesa por estar relacionada con el proyecto del que es objeto esta memoria.

La interfaz entre el **Sistema GA** y el **Centro de Mensajes Cortos** (**CMC**) permite el poder realizar la distribución automática de actuaciones pendientes de forma simultánea a diversos técnicos, que recibirán la información vía mensajes cortos de texto en sus terminales móviles.

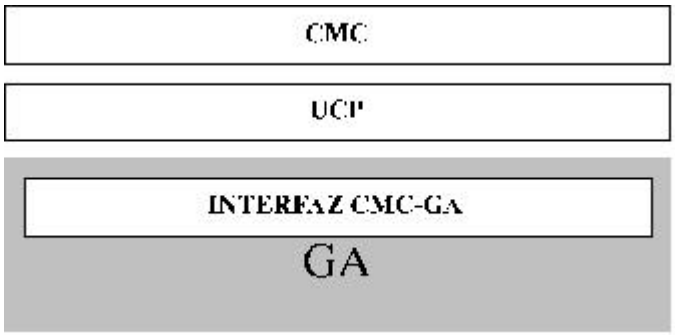
Está basada y desarrollada sobre la ya existente y estandarizada interfaz entre **Centros de Mensajes Cortos** (**SMSC**) y **sistemas informáticos**, con protocolo **UCP** (**Universal Computer Protocol**). A partir de esta, se desarrolla una interfaz propia para nuestra aplicación que tenga en cuenta las características particulares de nuestro sistema, como podría ser la de la necesidad de poder realizar el envío de varios mensajes cortos consecutivos a los técnicos para que estos puedan recibir toda la información que les es necesaria en la resolución de la actuación que se les ha asignado. Esta interfaz también debe ser capaz de interpretar los mensajes cortos de texto que provienen del terminal móvil de un técnico y que han sido generados por la aplicación que se ha programado en la tarjeta SIM de los mismos. Debe poder extraer la información que allí va contenida y entregársela a nuestro sistema en un formato tal que este pueda entenderlo.

La comunicación física se establece mediante una línea punto a punto entre el **Sistema GA** y el **CMC** (requisito "hardware" necesario para una óptima comunicación con los

técnicos, ya que este es un punto crítico en nuestro sistema y debemos procurar que dicha comunicación se produzca con el menor número de errores posible). El tipo de linea punto a punto que se establezca dependerá en gran medida de los recursos disponibles.

La arquitectura de conexión se presenta en la **Figura 8** :

Figura 8.- Arquitectura de conexión CMC-GA



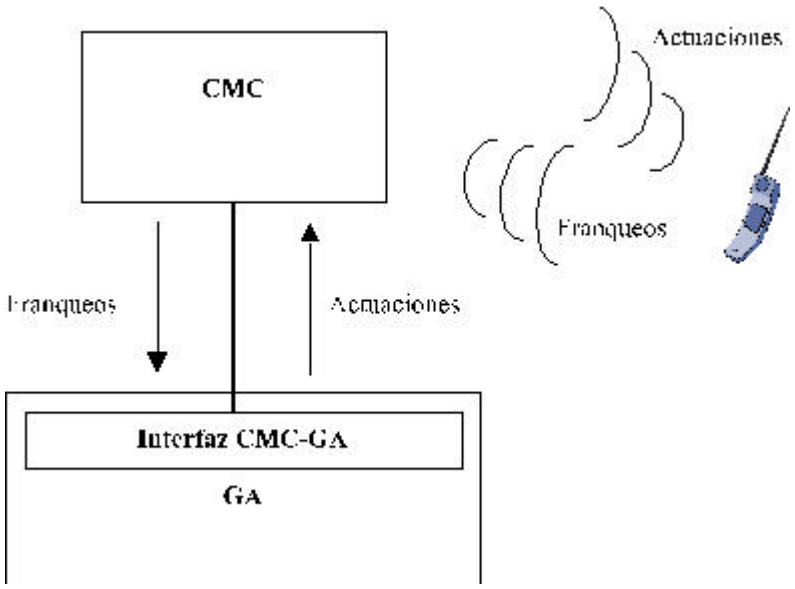
La transferencia entre ambos sistemas está basada en el intercambio de mensajes. Debido a las limitaciones dadas por el **SMS**, el mensaje se divide en cuatro mensajes cortos (longitud limitada de los mensajes de texto). Cuando el **CMC** recibe la información relativa a las actuaciones, en forma de estos mensajes cortos, notifica al sistema GA el estado de la transmisión. Cuando la transmisión ha sido correcta, se distribuye a su destinatario que la recibirá en el teléfono móvil. Posteriormente, el **CMC** también será el responsable de recoger los mensajes que los técnicos envíen tras la realización de su trabajo, y comunicárselo al Sistema GA.

El servicio de **mensajeria corta** de la telefonía móvil digital es un **servicio no fiable**, de tal forma que, aunque nuestro sistema haya recibido notificación por parte del **CMC** de que la transmisión se ha llevado a cabo correctamente, los mensajes puede que no lleguen finalmente a su destino con lo cual el sistema se quedaría esperando una comunicación por parte del técnico relativa a la avería que le ha sido asignada, y tal comunicación no se llevaría a cabo ya que dicho técnico estaría esperando la comunicación de los datos relativos a su próxima actuación por parte del sistema. Como vemos, aparecerían inconsistencias en el sistema relativas a determinadas averías y técnicos las cuales producirían **excepciones** que habrían de ser resueltas manualmente. Como parte del objetivo del desarrollo de este sistema es la automatización completa del proceso de resolución de actuaciones relativas a averías, debemos buscar una forma

más eficiente de comunicación con los técnicos, para lo cual es objeto la realización de este proyecto. La comunicación con los técnicos usando conexiones en tiempo real entre estos y el sistema (conexiones al sistema usando la red **Internet** y la red **GSM**) evita el problema anterior, ya que cuando la comunicación no sea posible no creará inconsistencias en el sistema, simplemente habrá que volver a intentarlo el número de veces que sea necesario hasta que dicha comunicación se produzca. Veremos que dependiendo de la solución que adoptemos podremos suprimir o no (junto con los mensajes cortos que transportan la información relativa a la actuación) el mensaje corto que realiza la función de **notificación** (o alarma) de nuevos datos a los técnicos. Será posible la eliminación de todos los mensajes cortos usados hasta ahora en aquellos casos en los que podamos hacer uso de un sistema de notificaciones alternativo al usado por nuestro sistema.

La **Figura 9** presenta el flujo de información entre los sistemas involucrados en esta comunicación.

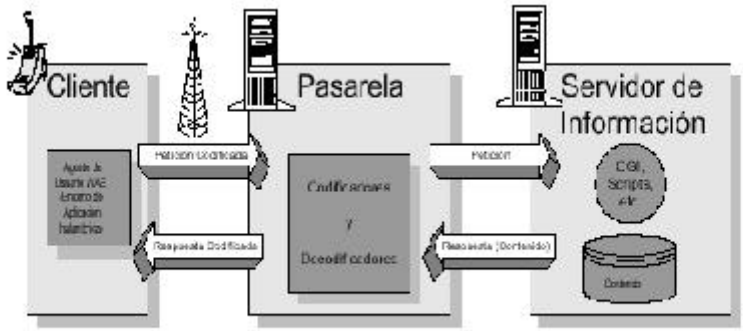
Figura 9.- Flujo de información entre terminales móviles, CMC y GA.



3.3 El Protocolo Inalámbrico de Aplicaciones (WAP)

El **Protocolo de Aplicaciones Inalámbricas** surge como la combinación de dos tecnologías de amplio crecimiento y difusión durante los últimos años: Las **Comunicaciones Inalámbricas** e **Internet**. Mas allá de la posibilidad de acceder a los servicios de información contenidos en Internet, el protocolo pretende proveer de servicios avanzados adicionales como, por ejemplo, el desvío de llamadas inteligente, en el cual se proporcione una interfaz al usuario en el cual se le pregunte la acción que desea realizar: aceptar la llamada, desviarla a otra persona, desviarla a un buzón vocal, etc. Para ello, se parte de una arquitectura basada en la arquitectura definida para el **World Wide Web** (**WWW**), pero adaptada a los nuevos requisitos del sistema. En la **Figura 10** se muestra el esquema de la arquitectura **WAP**.

Figura 10.- Arquitectura WAP.



De esta forma, en el terminal inalámbrico existiría un "**micronavegador**" (se pretende que este micronavegador actúe de interfaz con el usuario de la misma forma que lo hacen los navegadores estándar) encargado de la coordinación con la pasarela, a la cual la realiza peticiones de información que son adecuadamente tratadas y redirigidas al servidor de información adecuado. Una vez procesada la petición de información en el servidor, se envía esta información a la pasarela que de nuevo procesa adecuadamente para enviarlo al terminal inalámbrico.

Para conseguir consistencia en la comunicación entre el terminal móvil y los servidores de red que proporcionan la información, **WAP** define un conjunto de componentes estándar:

- **Un modelo de nombres estándar.**

  Se utilizan las **URIs** (**Universal/Uniform Resource Identifier** ó **Identificador Uniforme/Universal de Recurso**) definidas en WWW para identificar los recursos locales del dispositivo (tales como funciones de control de llamada) y las **URLs** (también definidas en el WWW, **Universal/Uniform Resource Location** ó **Localización Universal/Uniforme de Recurso**) para identificar el contenido WAP en los servidores de información.

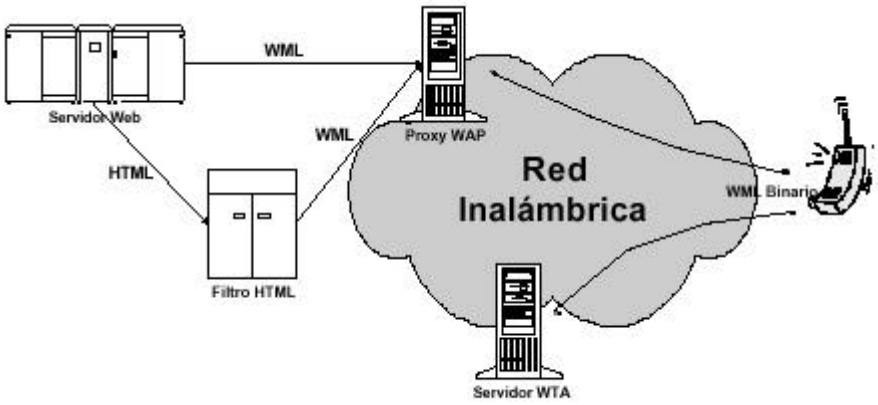
- **Un formato de contenido estándar, basado en la tecnologíaWWW.**

- **Unos protocolos de comunicación estándares.**

Estos protocolos son los que van a posibilitar la comunicación del micronavegador del terminal móvil con el servidor Web en red.

Veamos ahora un modelo global de funcionamiento de este sistema en la **Figura 11**.

Figura 11.- Arquitectura general de una red basada en el protocolo WAP.



En el ejemplo de la figura, nuestro terminal móvil tiene dos posibilidades de conexión: a un **proxy WAP**, o a un **servidor WTA** (**Wireless Telephony Application** ó **Aplicación de Telefonía Inalámbrica**).

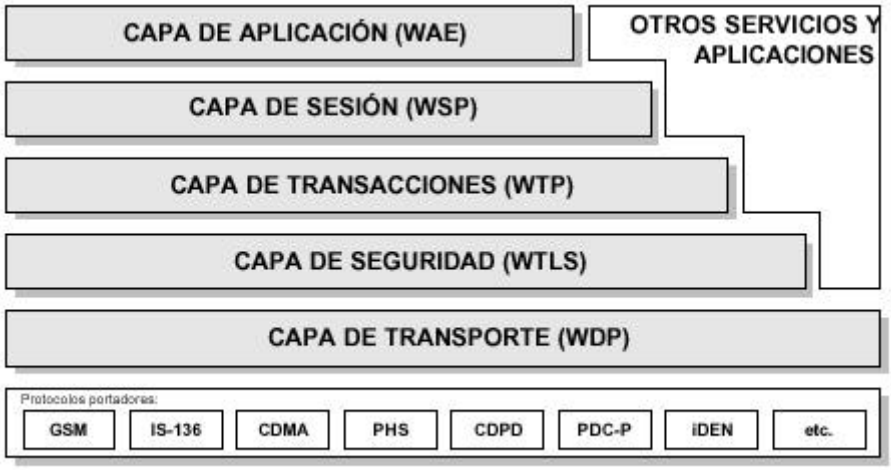
El primero de ellos, el **proxy WAP**, traduce las peticiones WAP a peticiones WEB (HTTP), de forma que el cliente WAP (el terminal inalámbrico) pueda realizar peticiones de información al servidor WEB. Adicionalmente, este proxy codifica las respuestas del servidor WEB en un formato binario compacto, que es interpretable por el cliente. Por tanto, este proxy WAP va a ser el encargado de traducir las peticiones del cliente móvil que usan el protocolo WAP a peticiones del protocolo estándar de Internet, el protocolo HTTP, para que puedan ser enviadas al servidor correspondiente a través de la red Internet. Las respuestas que producen dichos servidores también hacen uso del protocolo HTTP, con lo cual serán dirigidas al proxy WAP para que realice la conversión de protocolo y curse los datos de la respuesta al terminal móvil que lo hubiese solicitado, usando un protocolo capaz de ser entendido por los navegadores de que disponen estos terminales.

El segundo de ellos, el **Servidor WTA,** está pensado para proporcionar acceso WAP a las facilidades proporcionadas por la infraestructura de telecomunicaciones del proveedor de conexiones de red. Este servidor será capaz de entender las peticiones en el protocolo WAP y proporcionar, de esta forma, los servicios solicitados. Al no ser ya peticiones que deban usar el protocolo http, no es necesaria la aparición del proxy WAP que era el encargado de realizar las conversiones de protocolo.

3.3.1 Componentes de la arquitectura WAP

Una vez introducido el sistema, vamos a ver la arquitectura que le da consistencia. La arquitectura WAP está pensada para proporcionar un "**entorno escalable y extensible para el desarrollo de aplicaciones para dispositivos de comunicación móvil**". Para ello, se define una estructura en capas, en la cual cada capa es accesible por la capa superior así como por otros servicios y aplicaciones a través de un conjunto de interfaces muy bien definidos y especificados. Este esquema de capas de la arquitectura WAP la podemos ver en la **Figura12**. La comunicación entre entidades de la misma capa se realizará utilizando protocolos diferentes para cada una de las capas (WAP no es en si mismo un protocolo, representa a una **torre de protocolos** completa que posibilita la comunicación entre entidades de la misma capa a distintosniveles).

Figura 12.- Capas distintas de la arquitectura WAP.



Hagamos un recorrido por estas capas de forma breve, antes de pasar a analizarlas con más profundidad.

3.3.1.1 CAPA DE APLICACIÓN (WAE , Wireless Application Environment ó Entorno Inalámbrico de Aplicación)

El **Entorno Inalámbrico de Aplicación** (**WAE**) es un entorno de aplicación de propósito general basado en la combinación del World Wide Web y tecnologías de Comunicaciones Móviles.

Este entorno incluye un micronavegador, del cual ya hemos hablado anteriormente, que posee las siguientes funcionalidades como características más importantes:

- **Lenguaje WML.**

  Un lenguaje denominado **WML** (**Wireless Markup Language**) similar al HTML, pero optimizado para su uso en terminales móviles.

- **Lenguaje de Script.**

  Un lenguaje denominado **WMLScript**, similar al JavaScript (esto es, un lenguaje para su uso en forma de **Script**).

- **Formatos especiales.**

  Un conjunto de formatos de contenido, que son un conjunto de formatos de datos bien definidos entre los que se encuentran imágenes, entradas en la agenda de teléfonos e información de calendario.

3.3.1.2 CAPA DE SESIÓN (WSP, Wireless Session Protocol ó Protocolo Inalámbrico de Sesión)

El **Protocolo Inalámbrico de Sesión** (**WSP**) proporciona a la Capa de Aplicación de WAP interfaz con dos servicios de sesión:

- **Servicio orientado a conexión:** servicio que funciona por encima de la Capa de Transacciones.
- **Servicio no orientado a conexión:** servicio que funciona por encima de la Capa de Transporte (y que proporciona servicio de **datagramas seguro** o servicio de **datagramas no seguro**).

Actualmente, esta capa consiste en servicios adaptados a aplicaciones basadas en la navegación Web, proporcionando las siguientes funcionalidades:

- **Semántica y funcionalidades del HTTP/1.1 en una codificación compacta.**
- **Negociación de las características del Protocolo.**
- **Suspensión de la Sesión y reanudación de la misma con cambio de sesión.**

3.3.1.3 CAPA DE TRANSACCIONES (WTP, Wireless Transaction Protocol ó Protocolo Inalámbrico de Transacción)

El **Protocolo Inalámbrico de Transacción** (**WTP**) funciona por encima de un servicio de **datagramas**, tanto seguros como no seguros, proporcionando las siguientes funcionalidades:

- **Transacciones síncronas:** tres clases de servicio de transacciones síncronas:
  - ✓ **Peticiones inseguras de un solo camino.**
  - ✓ **Peticiones seguras de un solo camino.**
  - ✓ **Transacciones seguras de dos caminos (petición-respuesta).**
- **Seguridad:** seguridad usuario-a-usuario opcional.
- **Transacciones asíncronas.**

3.3.1.4 CAPA DE SEGURIDAD (WTLS, Wireless Transport Layer Security ó Capa Inalámbrica de Seguridad de Transporte)

La **Capa Inalámbrica de Seguridad de Transporte** (**WTLS**) es un protocolo basado en el estándar **SSL**, utilizado en el entorno Web para la proporción de seguridad en la realización de todo tipo de transferencias de datos. Este protocolo ha sido especialmente diseñado para los protocolos de transporte de WAP y, por tanto, está optimizado para ser utilizado en canales de comunicación de banda estrecha. Para este protocolo se han definido las siguientes características:

- **Integridad de los datos.**

Este protocolo asegura que los datos intercambiados entre el terminal y un servidor de aplicaciones no ha sido modificada y no es información corrupta.

- **Privacidad de los datos.**

  Este protocolo asegura que la información intercambiada entre el terminal y un servidor de aplicaciones no puede ser entendida por terceras partes que puedan interceptar el flujo de datos, se trata de encriptar la información.

- **Autentificación.**

  Este protocolo contiene servicios para establecer la autenticidad del terminal y del servidor de aplicaciones.

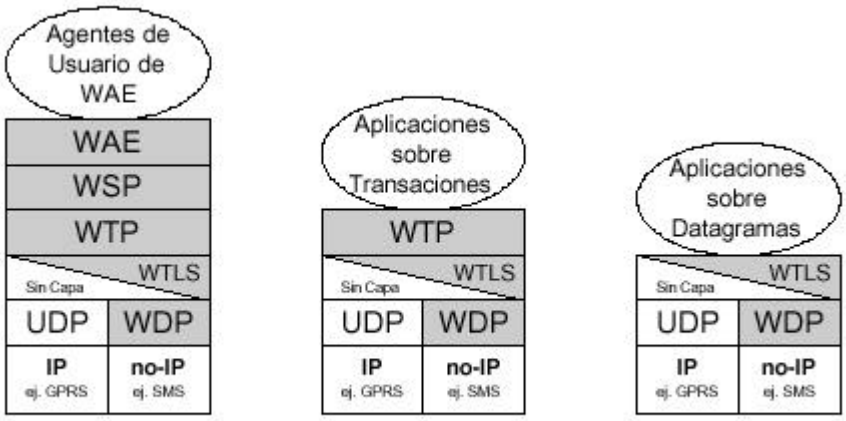
Adicionalmente, el **WTLS** puede ser utilizado para la realización de comunicación segura entre terminales, por ejemplo en el caso de operaciones de comercio electrónico entre terminales móviles.

3.3.1.5 CAPA DE TRANSPORTE (WDP, Wireless Datagram Protocol ó Protocolo Inalámbrico de Datagramas)

El **Protocolo Inalámbrico de Datagramas** (**WDP**) proporciona un servicio fiable a los protocolos de las capas superiores de WAP y permite la comunicación, de forma transparente para los usuarios de esta capa, sobre los protocolos portadores válidos. Debido a que este protocolo proporciona un interfaz común a los protocolos de las capas superiores, las capas de Seguridad, Sesión y Aplicación pueden trabajar independientemente de la red inalámbrica que dé soporte al sistema. Se trata de ofrecer un servicio fiable de transporte a las capas superiores independientemente de los distintos tipos de servicio ofrecidos por las capas inferiores.

Antes de pasar a estudiar en más profundidad cada uno de estos protocolos que hemos ido mencionando, vamos a ver tres ejemplos de interconexión de estas capas en la **Figura 13**.

Figura 13.- Ejemplos de interconexión entre capas en tecnología WAP.



Así pues, dependiendo de la aplicación en cuestión, la comunicación se realizará con una determinada capa de la estructura de WAP.

3.3.2 El entorno inalámbrico de aplicaciones (WAE)

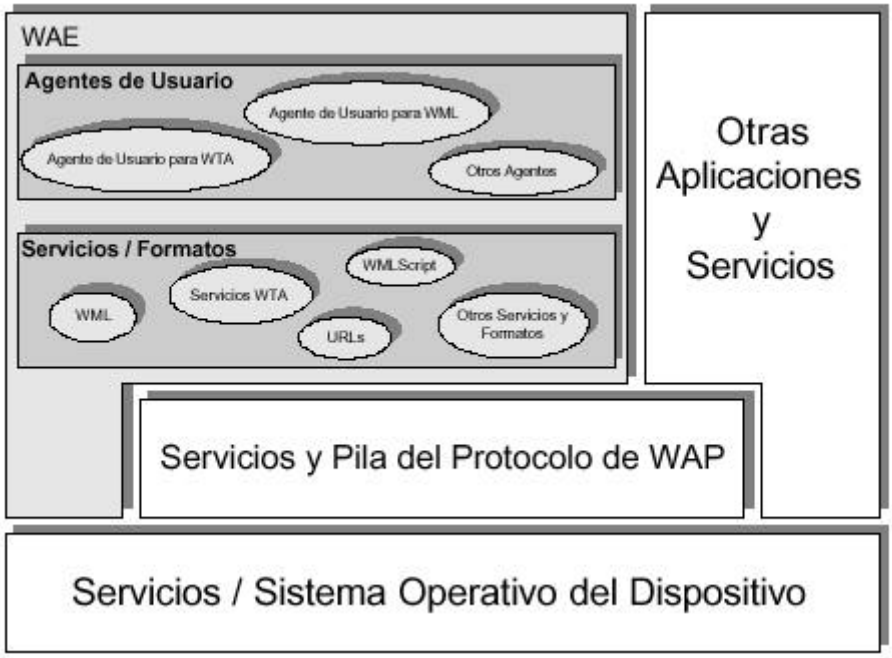
El objetivo del **Entorno Inalámbrico de Aplicaciones** es construir un entorno de aplicación de propósito general, basado fundamentalmente en la filosofía y tecnología del **World Wide Web** (WWW). Principalmente, se pretende establecer un entorno que permita a los operadores y proveedores de servicios construir aplicaciones y servicios que puedan utilizarse en una amplia variedad de plataformas inalámbricas de forma útil y eficiente.

De esta forma, la arquitectura del **Entorno Inalámbrico de Aplicaciones** (en adelante **WAE**) está enfocado principalmente sobre los aspectos del cliente de la arquitectura del sistema de **WAP**, esto es, de los puntos relacionados con los agentes de usuario (Un agente de usuario es todo aquel software o dispositivo que interpreta un contenido, p. e. **WML**. Esto incluye navegadores de texto, navegadores de voz, sistemas de búsqueda, etc.). Esto es debido a que la parte que más interesa de la arquitectura es aquella que afecta principalmente a los terminales móviles, esto es, a aquellos puntos en los cuales van a estar ejecutándose los diversos agentes de usuario.

Si volvemos sobre la **Figura 10**, vemos que entre los agentes de usuario localizados en el cliente (en el terminal móvil) y los servidores de información se define un nuevo elemento: **Las Pasarelas**. Su función es codificar y decodificar la información intercambiada con el cliente, para así minimizar la cantidad de datos radiados, así como minimizar el proceso de la información por parte del cliente. Estas pasarelas serán gestionadas por los operadores de telefonía móvil que ofrezcan a sus clientes el servicio WAP (cada operador gestionará sus pasarelas).

Basándonos en esta arquitectura, vamos a profundizar un poco más en los componentes de este **Entorno Inalámbrico de Aplicación**. Una primera visión de cómo estaría formado un cliente particular de este entorno la encontramos representada en la **Figura 14**.

Figura 14.- Componentes del cliente de WAE.



Tal y como podemos observar en la **Figura 14**, se divide en dos partes, dos capas lógicas:

- **Agentes de usuario.**

  Los Agentes de Usuario incluyen aquellos elementos como navegadores, agendas telefónicas, editores de mensajes, etc.

- **Servicios y Formatos.**

  Los Servicios y Formatos incluyen todos aquellos elementos y formatos comunes, accesibles a los Agentes de Usuario, tales como WML, WMLScript, formatos de imagen, etc.

Como se puede ver en la Figura, dentro de **WAE** se separan Servicios de Agentes de Usuario, lo que proporciona flexibilidad para combinar varios servicios dentro de un único Agente de Usuario, o para distribuir los servicios entre varios Agentes de Usuario.

Los dos Agentes de Usuario más importantes son:

- **Agente de Usuario para WML.**

  El Agente de Usuario para WML es el Agente de Usuario fundamental en la arquitectura del Entorno Inalámbrico de Aplicación. A pesar de su importancia, este Agente de Usuario no está definido formalmente dentro de esta arquitectura, ya que sus características y capacidades se dejan en manos de los encargados de su implementación. El único requisito de funcionalidad que debe cumplir este Agente de Usuario, es el proporcionar un sistema intérprete a los lenguajes WML y WMLScript, de forma que se permita la navegación desde el terminal móvil.

- **Agente de Usuario para WTA.**

  El Agente de Usuario para WTA permite a los autores acceder e interactuar con las características de los teléfonos móviles (p.e. Control de Llamada), así como otras aplicaciones supuestas en los teléfonos, tales como agendas de teléfono y aplicaciones de calendario.

3.3.3 El protocolo inalámbrico de sesión

El **Protocolo Inalámbrico de Sesión** constituye la capa que se sitúa por debajo de la capa de Aplicación siguiendo la torre de protocolos. Esta capa se comunicará con la de aplicación y, situada por debajo de ella en esta torre, con la de transporte. Haciendo uso de esta comunicación entre capas proporcionará la capacidad necesaria para:

- **Establecimiento de la conexión.**

  Establecer una conexión fiable entre el cliente y el servidor, y liberar esta conexión de una forma ordenada.

- **Negociación.**

  Ponerse de acuerdo en un nivel común de funcionalidades del protocolo, a través de la negociación de las posibilidades. Se producirá un proceso de negociación a la baja.

- **Intercambio y codificación.**

  Intercambiar contenido entre el cliente y el servidor utilizando codificación compacta.

- **Suspensión y recuperación.**

  Suspender y recuperar la sesión.

Hoy por hoy, este protocolo ha sido definido únicamente para el caso de la navegación, definiéndose como **WSP/B** (**Wireless Session Protocol – Browsing**) . Esta implementación está realizada para el establecimiento de una conexión sobre la base de un protocolo compatible con **HTTP 1.1**. De esta forma, se han definido un conjunto de **primitivas de servicio** (una primitiva de servicio representa el intercambio lógico de información entre la capa de Sesión y capas adyacentes) para permitir la comunicación entre la capa de sesión integrada dentro del equipo cliente y la capa de sesión integrada en el equipo servidor. Estas primitivas, junto con una pequeña descripción de las mismas, puede verse en la **Tabla 1**.

Tabla 1.- Primitivas de la capa de sesión.

Nombre de la primitiva	Descripción
S-Connect	Esta primitiva se utiliza para iniciar el establecimiento de la conexión, y para la notificación de su éxito
S-Disconnect	Esta primitiva se utiliza para desconectar una sesión, y para notificar al usuario de una sesión que esa sesión no se puede establecer, que ha sido desconectada
S-Suspend	Esta primitiva se utiliza para solicitar la suspensión de la sesión
S-Resume	Esta primitiva se utiliza para solicitar que se recupere la sesión utilizando para las direcciones el nuevo identificador de punto de acceso de servicio.
S-Exception	Esta primitiva se utiliza para notificar aquellos eventos que no están asignados a una transacción en particular, ni provocan la desconexión o suspensión de la sesión.
S-MethodInvoke	Esta primitiva se utiliza para solicitar una operación que deba ser ejecutada en el servidor.
S-MethodResult	Esta primitiva se utiliza para devolver una respuesta a una petición de operación.
S-MethodAbort	Esta primitiva se utiliza para abortar una solicitud de ejecución de operación, que no haya sido aún completada.
S-Push	Esta primitiva se utiliza para enviar información no solicitada desde el servidor, dentro del contexto de una sesión de forma y sin confirmación.
S-ConfirmedPush	Esta primitiva realiza las mismas funciones que la anterior, pero con confirmación.
S-PushAbort	Esta primitiva se utiliza para anular una primitiva anterior del tipo *S-Push* o *S-ConfirmedPush.*

Adicionalmente, existen cuatro tipos de cada una de estas primitivas, tal y como puede verse en la **Tabla 2**.

Tabla 2.- Tipos de primitiva de servicio.

Tipo	Abreviación	Descripción
Request	req	Se utiliza cuando una capa superior solicita un servicio de la capa inmediatamente inferior
Indication	ind	Una capa que solicita un servicio utiliza este tipo de primitiva para notificar a la capa inmediatamente superior de las actividades relacionadas con su par, o con el proveedor del servicio
Response	res	Este tipo de primitiva se utiliza para reconocer la recepción de la primitiva de tipo *Indication* de la capa inmediatamente inferior
Confirm	cnf	La capa que proporciona el servicio requerido utiliza este tipo de primitiva para notificar que la actividad ha sido completada satisfactoriamente.

Por último, reseñar que cada una de estas primitivas está perfectamente definida dentro de la especificación, tanto desde el punto de vista del diagrama de tiempos en el que se tienen que invocar las primitivas, como desde el punto de vista de los parámetros intercambiados.

3.3.4 El protocolo inalámbrico de transacción

El **Protocolo Inalámbrico de Transacción** se establece para proporcionar los servicios necesarios que soporten aplicaciones de "**navegación**" (del tipo **petición/respuesta**). Es a este dúo **petición/respuesta**, lo que vamos a denominar como **transacción**. Este protocolo se sitúa por encima del **Protocolo Inalámbrico de Datagramas** (que es el correspondiente a la capa de transporte) y, de forma opcional, de la **Capa Inalámbrica de Seguridad de Transporte** (en función del tipo de aplicación se usará o no este último), que serán estudiados posteriormente.

Las características de este protocolo son:

- Proporciona tres clases de servicios de transacción:

  - ✓ **Clase 0:** mensaje de solicitud no seguro, sin mensaje de resultado.
  - ✓ **Clase 1:** mensaje de solicitud seguro, sin mensaje de resultado.
  - ✓ **Clase 2:** mensaje de solicitud seguro, con, exactamente, un mensaje de resultado seguro.

- La seguridad se consigue a través del uso de identificadores únicos de transacción, asentimientos, eliminación de duplicados y retransmisiones.
- Seguridad opcional usuario a usuario.
- De forma opcional, el último asentimiento de la transacción puede contener algún tipo de información adicional relacionada con la transacción, como medidas de prestaciones, etc.
- Se proporcionan mecanismos para minimizar el número de transacciones que se reenvían como resultado de paquetes duplicados.
- Se permiten las transacciones asíncronas.

Al igual que en el protocolo anterior, en la **Tabla 3** vamos a ver las primitivas de servicio correspondientes a la capa de transacción.

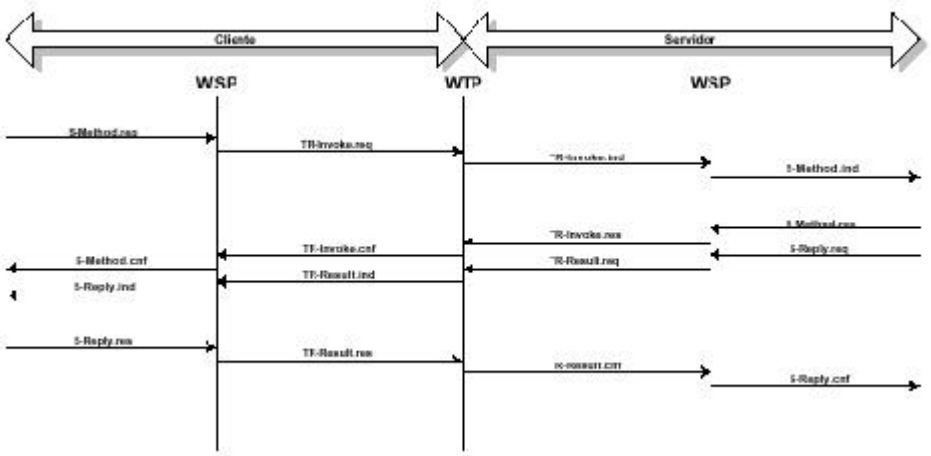
Tabla 3.- Primitivas de Servicio de transacción.

Nombre de la primitiva	Descripción
TR-Invoke	Esta primitiva se utiliza para iniciar una nueva transacción.
TR-Result	Esta primitiva se utiliza para devolver el resultado de transacción iniciada anteriormente
TR-Abort	Esta primitiva se utiliza para abortar una transacción existente

Las primitivas que aparecen en la **Tabla 3** son las que sustentan la comunicación entre dos capas de transacciones situadas en dos equipos distintos.

Vemos en la **Figura 15** la concatenación de **Primitivas de Servicio de Sesión y de Transacción** para el caso de una **petición-respuesta**.

Figura 15.- Intercambio de primitivas entre capa de Sesión y Transacción.



Para finalizar, vamos a detallar un poco más las principales características de este protocolo:

- **Transferencia de Mensajes.**

  Dentro de este protocolo se distinguen dos tipos de mensajes: mensajes de datos y mensajes de control. Los mensajes de datos transportan únicamente datos de usuario, mientras que los mensajes de control se utilizan para los asentimientos, informes de error, etc. pero sin transportar datos de usuario.

- **Retransmisión hasta el asentimiento.**

  Esta característica se utiliza para la transferencia fiable de datos desde un proveedor **WTP** a otro, en caso que haya pérdida de paquetes. A modo de comentario, dejar claro que para reducir lo máximo posible el número de paquetes que se transmiten, este protocolo utiliza asentimiento explícito siempre que sea posible.

- **Asentimiento de usuario.**

  El Asentimiento de Usuario permite al usuario de este protocolo,confirmar cada mensaje recibido por el proveedor **WTP**.

- **Información en el Último Asentimiento.**

  Se permite, así pues, enviar información en el último, y únicamente en el último, asentimiento de una transacción. De esta forma, se puede enviar, por ejemplo, información del rendimiento proporcionado por el sistema durante la transacción realizada, etc.

- **Concatenación y Separación.**

  Podemos definir **concatenación** como el proceso de trasmitir múltiples **Unidades de Datos del Protocolo** (**PDU**, **Protocol Data Unit**) de **WTP** en una **Unidad de Datos del Servicio** (**SDU**, **Service Data Unit**) de la red portadora. Por el contrario, **separación** es el proceso de separar múltiples **PDUs** de un único **SDU** (esto es, el proceso inverso al anterior). El objetivo de estos sistemas es proveer eficiencia en la transmisión inalámbrica, al requerirse un menor número de transmisiones.

- **Transacciones Asíncronas.**

  Para un correcto funcionamiento del protocolo, múltiples transacciones deben ser procesadas de forma asíncrona, debe ser capaz de iniciar múltiples transacciones antes que reciba la respuesta a la primera transacción.

- **Identificador de la Transacción.**

  Cada transacción está identificada de forma única por los pares de direcciones de los **sockets** (dirección fuente, puerto fuente, dirección destino y puerto destino) y por el **Identificador de Transacción** (**TID**, **Transaction Identifier**), el cual se incrementa para cada una de las transacciones iniciadas. Este número es de 16 bits, utilizándose el bit de mayor orden para indicar la dirección.

- **Segmentación y re-ensamblado. (opcional)**

  Si la longitud del mensaje supera la **Unidad Máxima de Transferencia** (**MTU**, **Maximum Transfer Unit**), el mensaje puede ser segmentado por el **WTP** y enviado en múltiples paquetes. Cuando esta operación se realiza, estos paquetes pueden ser enviados y asentidos en grupos. De esta forma, el emisor puede realizar control de

flujo cambiando el tamaño de los grupos de mensajes dependiendo de las características de la red.

3.3.5 La capa inalámbrica de seguridad de transporte (WTLS)

La Capa Inalámbrica de Seguridad de Transporte (en adelante **WTLS**), constituye una capa modular, que depende del nivel de seguridad requerido por una determinada aplicación. Esta capa proporciona a las capas de nivel superior de **WAP** de una interfaz de servicio de transporte seguro, que lo resguarde de una interfaz de transporte inferior. Al igual que hemos hecho en los protocolos anteriores, en la **Tabla 4** vamos a ver las primitivas de servicio.

Tabla 4.- Primitivas de servicio de capa de seguridad.

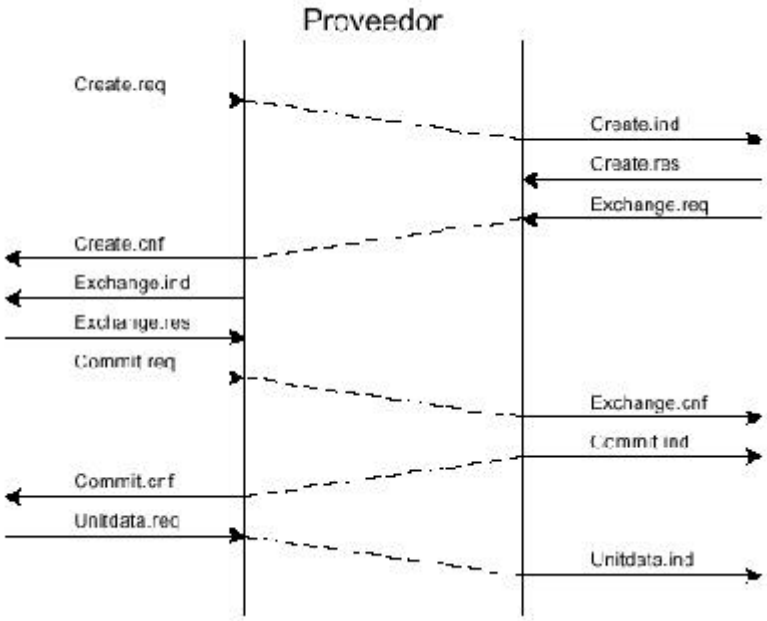
Nombre de la primitiva	Descripción
SEC-Unitdata	Esta primitiva se utiliza para intercambiar datos de usuario entre los dos participantes. Sólo puede ser invocada cuando existe previamente una conexión segura entre las direcciones de transporte de los dos participantes.
SEC-Create	Esta primitiva se utiliza para iniciar el establecimiento de una conexión segura.
SEC-Exchange	Esta primitiva se utiliza en la creación de una conexión segura si el servidor desea utilizar autentificación de clave pública o intercambio de claves con el cliente.
SEC-Commit	Esta primitiva se inicia cuando el *handshake*[20] se completa y cualquiera de los equipos participantes solicita cambiar a un nuevo estado de conexión negociado.
SEC-Terminate	Esta primitiva se utiliza para finalizar la conexión.
SEC-Exception	Esta primitiva se utiliza para informar al otro extremo sobre las alertas de nivel de aviso.
SEC-Create-Request	Esta primitiva se utiliza por el servidor para solicitar al cliente que inicie un nuevo *handshake*.

Todas las primitivas que estamos viendo pueden ser de cuatro tipos, tal y como se indicó en la **Tabla 2**. Estas primitivas son las que sustentan la comunicación entre dos capas situadas en dos equipos distintos, cuando nos encontramos en el nivel de la capa de seguridad.

El principal objetivo de esta capa es proporcionar privacidad, integridad de datos y autentificación entre dos aplicaciones que se comuniquen (intenta ofrecer un determinado nivel de seguridad en la transferencia de información entre los extremos de la conexión). Adicionalmente, la **WTLS** proporciona una interfaz para el manejo de conexiones seguras.

Hemos hablado anteriormente del proceso de establecimiento de una sesión segura o **handshake**. En la **Figura 16** podemos ver este intercambio de primitivas.

Figura 16.- Secuencia de primitivas para el establecimiento de una sesión segura.

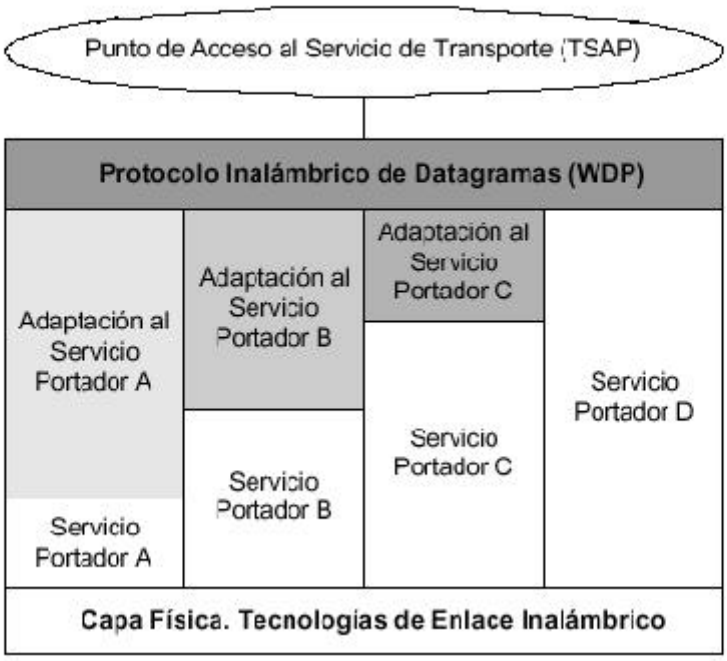


3.3.6 El protocolo inalámbrico de datagramas (WDP)

El Protocolo Inalámbrico de Datagramas (en adelante **WDP**, **Wireless Datagram Protocol**) ofrece un servicio consistente al protocolo (Seguridad, Transacción y Sesión) de la capa superior de **WAP**, comunicándose de forma transparente sobre uno de los servicios portadores disponibles. Este protocolo ofrece servicios a los protocolos superiores del estilo a direccionamiento por numero de puerto, segmentación y re-ensamblado opcional y detección de errores opcional, de forma que se permite a las aplicaciones de usuario funcionar de forma transparente sobre distintos servicios portadores disponibles. Estas aplicaciones de usuario no tienen porqué conocer los distintos servicios portadores que son ofrecidos por las capas inferiores y que se encuentran accesibles en los puntos de acceso al servicio. Simplemente deben conocer que existe la posibilidad del transporte de datos apoyándose en las capas inferiores, pero será completamente transparente para ellos la forma en que dichas capas inferiores realicen el transporte real de los datos entre los extremos de la conexión.

Para hacer esto, se plantea una arquitectura de protocolo como el que se muestra en la **Figura 17**.

Figura 17.- Arquitectura del Protocolo Inalámbrico de Datagramas.



Al igual que hemos hecho en los protocolos anteriores, en la **Tabla 5** vamos a ver las primitivas de servicio que se utilizan en este protocolo, las cuales podrán ser de cuatro tipos diferentes, para indicación (**indication**), solicitud (**request**), respuesta (**response)** y confirmación (**confirm)**.

Tabla 5.- Primitivas de servicio de la capa de datagramas.

Nombre de la primitiva	Descripción
T-DUnitdata	Esta primitiva es la utilizada para transmitir datos como datagramas. No requiere que exista una conexión para establecerse.
T-DError	Esta primitiva se utiliza para proporcionar información a la capa superior cuando ocurre un error que pueda influenciar en el servicio requerido.

Todo lo que hemos visto sobre el protocolo de aplicaciones inalámbricas ha sido desde un punto de vista general, englobando cualquier tipo de arquitectura que se pretenda montar sobre redes móviles, ya que dicho protocolo ha sido creado para optimizar la transferencia de datos sobre este tipo de redes, las cuales tienen una seria limitación para esta clase de servicio en el ancho de banda del que disponen (espectro radioeléctrico limitado).

Por último, vamos a ver en las figuras siguientes, la arquitectura de este protocolo dentro de la arquitectura global de **WAP**, para el caso de utilizarse **GSM** como servicio portador (**Figura 18** y **Figura 19**), que es el protocolo que más nos puede interesar por su amplia implantación en los sistemas de comunicaciones móviles telefónicas existentes hoy en día. La arquitectura para otros tipos de redes móviles es análoga a esta. También se representa, en la **Figura 20**, la arquitectura considerando **CDMA** (**Acceso Múltiple por División en el Código**).

Figura 18.- WDP sobre GSM SMS.

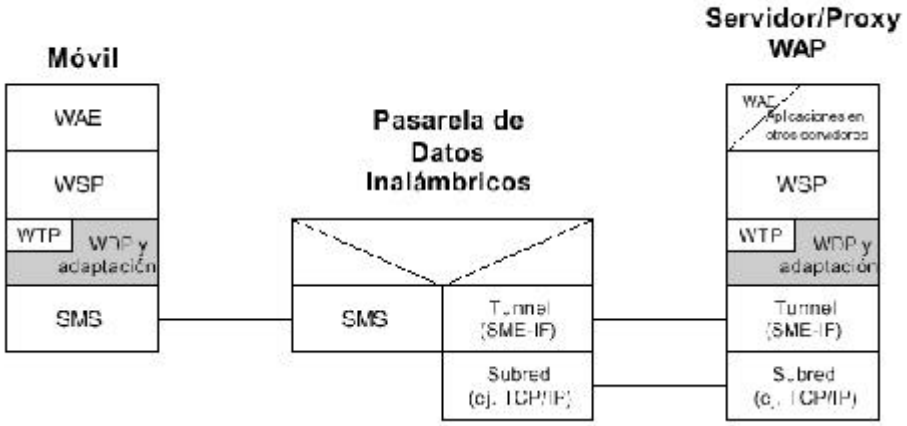


Figura 19.- WDP sobre GSM Canal de Datos de Circuitos Conmutados.

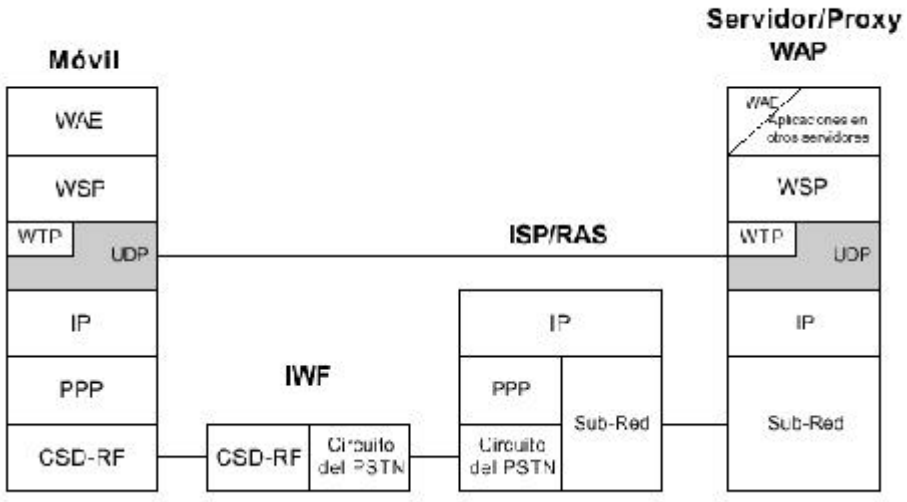
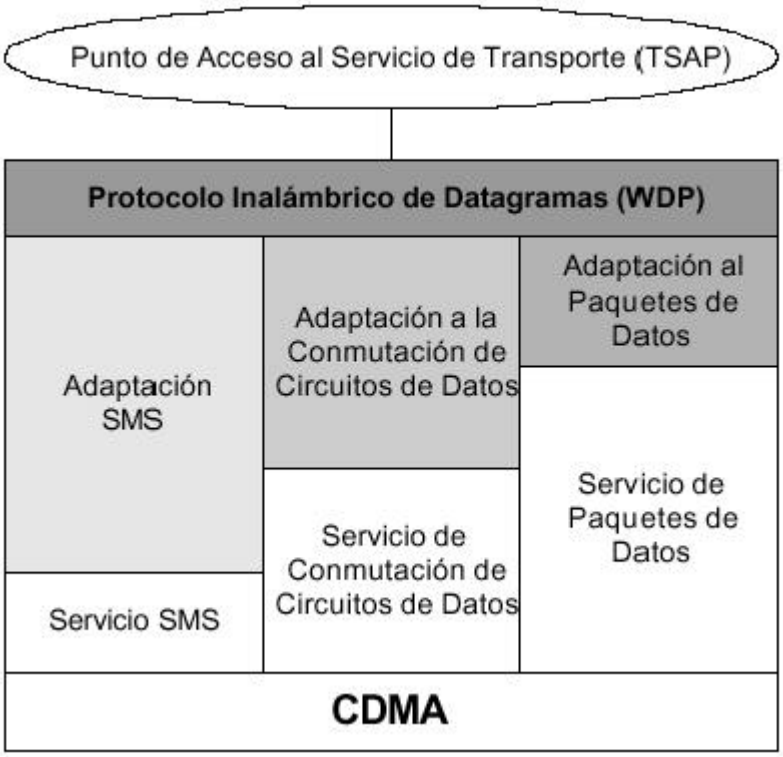


Figura 20.- WDP sobre servicios portadores CDMA.



*3.4 Oracle PRO*C/C++*

Pro*C es el lenguaje usado para realizar consultas a una base de datos Oracle dentro de un programa escrito en lenguaje de programación **C++**.

El precompilador de **Pro*C/C++** es el encargado de convertir las sentencias **SQL** (mediante las cuales se realizan las consultas a las tablas) embebidas en el programa **C** y **C++** en código **C** estándar. Cuando se precompila el código, el resultado es un programa en **C** o **C++** que se puede compilar posteriormente y usarse para construir aplicaciones que accedan a una base de datos **Oracle**.

Para el acceso a la base de datos, se usa un lenguaje de consulta de alto nivel conocido como **Lenguaje de Consulta Estructurado (Structured Query Languaje) o SQL**. El precompilador de **Pro*C/C++** convierte las sentencias **SQL** en comandos **EXEC**

SQL dentro de sentencias escritas en **C**, de tal forma que el resultado obtenido puede ser compilado por un compilador de **C/C++**.

El precompilador de **Pro*C/C++** permite crear aplicaciones que accedan a una base de datos **Oracle** cuando las prioridades sean un desarrollo rápido de dichas aplicaciones y la compatibilidad con otros sistemas.

3.5 Introducción a las Bases de Datos de Oracle

3.5.1 Introducción

El servidor de **Oracle** es un **sistema de gestión de Bases de Datos Relacionales**. Un servidor de Base de Datos resuelve los problemas de gestión de la información, ya que realiza la gestión de una gran cantidad de datos en un entorno multiusuario en el cual muchos usuarios acceden concurrentemente a los mismos datos. Todo ello debe ser correctamente gestionado para obtener un rendimiento alto de las aplicaciones. Un servidor de Bases de Datos debe también prevenir contra accesos no autorizados y dar soluciones eficientes para los fallos en el tratamiento de los datos.

Oracle nos da soluciones eficientes para entornos con arquitecturas que siguen el modelo **cliente/servidor** (**procesos distribuidos**). Para aprovechar todas las ventajas de un sistema de computadores que trabajan en red, **Oracle** permite dividir el procesamiento de datos entre el servidor de Base de Datos y los programas de aplicación cliente. La máquina sobre la que se monta el sistema de gestión de la Base de Datos controla todas las tareas del servidor, mientras que las estaciones de trabajo que funcionan como clientes de la aplicación se encargan de la interpretación y visualización de los datos.

Oracle soporta las Bases de Datos más grandes, las cuales pueden contener terabytes de datos. También soporta un amplio número de usuarios concurrentes ejecutando una amplia variedad de aplicaciones de Base de Datos que operan sobre los mismos datos.

Las Bases de Datos **Oracle** no interrumpen su funcionamiento durante la relización de copias de seguridad o por caidas parciales del sistema.

Oracle se adhiere a los estándares aceptados por la industria en todo aquello que se relaciona con lenguajes de acceso a los datos, sistemas operativos, interfaces de usuario y protocolos de comunicación de red. Es un sistema abierto que protege las inversiones de los usuarios. También soporta el **Protocolo de Gestión de Red Simple** (**SNMP**)

estándar para gestión de sistemas. Este protocolo permite a los administradores gestionar sistemas heterogéneos con una única interfaz de administración.

Como protección contra accesos no autorizados a la Base de Datos y el uso de esta, **Oracle** presenta características de seguridad que limitan el acceso y la monitorización de los datos. Son características que facilitan la gestión de los diseños más complejos de acceso a los datos.

Oracle garantiza la integridad de los datos siguiendo las reglas que dictan los estándares para considerar los datos como aceptables. Esto reduce el coste de codificar y gestionar consultas en muchas aplicaciones que operan sobre Bases de Datos.

Otra característica importante es la portabilidad. El software de **Oracle** trabaja bajo diferentes sistemas operativos, lo que hace que las aplicaciones desarrolladas para **Oracle** puedan ser portadas a cualquier otro sistema operativo con pequeñas o ninguna modificación.

En entornos de red, para aplicaciones distribuidas, **Oracle** combina los datos físicamente localizados en diferentes computadores en una **Base de Datos lógica**, a la cual pueden acceder todos los usuarios de la red. Los sistemas distribuidos tienen el mismo grado de transparencia de usuario y de consistencia de los datos que los sistemas no distribuidos y disfrutan de las ventajas de la gestión de Bases de Datos locales.

También se ofrece heterogeneidad en el sentido de que se permite a los usuarios el acceso a algunas Bases de Datos no Oracle con total transparencia para estos.

El software de **Oracle** permite la "réplica" de grupos de tablas a múltiples sitios. Soporta la "réplica" de los cambios a nivel de esquema y a nivel de datos a todos esos sitios.

3.5.2 Arquitectura de Oracle

3.5.2.1 El servidor de Oracle

Se trata de un **sistema de gestión de Bases de Datos Relacionales** que proporciona una gestión de la información integrada, comprensiva y abierta. Un servidor de **Oracle** está formado por una **Base de Datos Oracle** y una **instancia del servidor de Oracle**. La combinación de los procesos que se arrancan al crear la Base de Datos y el area de memoria utilizada para mostrar la información solicitada por los usuarios es lo que se conoce como una **instancia** de Oracle. Dicha instancia tiene dos tipos de procesos:

- **Procesos de usuario:** ejecutan el código de un programa de aplicación.
- **Procesos Oracle:** son los procesos del servidor que realizan su trabajo para los procesos de usuario y los procesos secundarios que realizan las labores de mantenimiento para el servidor Oracle.

Cuando múltiples usuarios se conectan a la Base de Datos para acceder a los mismos datos físicos se crea una instancia por cada usuario, lo cual nos va a proporcionar un aumento del rendimiento.

3.5.2.2 Estructura lógica de la Base de Datos

Una **Base de Datos Oracle** es una colección de datos que se trata como una unidad. El propósito de una Base de Datos es almacenar información y relacionarla para suministrar posteriormente aquella información solicitada mediante consultas. La base de datos tiene estructuras lógicas y físicas. Ambos tipos de estructura son independientes, por lo cual el almacenamiento físico de los datos puede ser gestionado sin afectar al acceso a las estructuras de almacenamiento lógicas.

Dentro de las estructuras lógicas de las bases de datos Oracle encontramos **tablespaces, esquemas, bloques de datos y segmentos.**

Una base de datos se divide en unidades de almacenamiento lógicas llamadas **tablespaces (**división lógica dentro de la BD. Esta zona lógica tendrá su ubicación física y es en esas zonas donde los usuarios guardan sus datos). Estas agrupan juntas diversas estructuras lógicas relacionadas. Comúnmente agrupan todos los objetos de una aplicación para simplificar algunas operaciones administrativas. Podemos decir que:

- **Base de Datos**: estructura lógica de **tablespaces** + estructura física de discos y ficheros (estática).
- **Instancia**: BD + un conjunto de procesos background y una estructura de memoria (dinámico).

Tenemos lo siguiente:

- Cada base de datos está dividida logicamente en una o más **tablespaces**.
- Uno o más ficheros de datos (**datafiles**) se crean explicitamente por cada tablespace para almacenar fisicamente los datos de todas las estructuras lógicas presentes en una tablespace.
- La combinación de los tamaños de los archivos de datos de una tablespace es la capacidad de almacenamiento total de la tablespace (por defecto, SYSTEM tablespace tiene 2 MB de capacidad de almacenamiento mientras que USERS tablespace tiene 4 MB).
- La combinación de la capacidad de almacenamiento de las tablespaces de una base de datos es la capacidad de almacenamiento total de la base de datos (por defecto, 6 MB).

Los tablespace podemos dividirlos en:

- **Online (accesible):** los usuarios pueden acceder a la información.
- **Offline (no accesible):** una parte de la base de datos estará indisponible mientras que se podrá acceder normalmente al resto. Esto facilita la realización de muchas tareas de administración.

Un **esquema** es una colección de objetos de la base de datos. Los objetos esquema son las estructuras de datos que se refieren directamente a los datos de la base de datos. Los objetos esquema incluyen estructuras como tablas , vistas, secuencias, indices,... No hay ninguna relación entre un tablespace y un esquema; objetos del mismo esquema pueden encontrarse en tablespaces diferentes, y un tablespace puede incluir objetos de diferentes esquemas.

Llegados a este punto vamos a enumerar las tablas de la base de datos de nuestro sistema a las cuales accedemos para consultarlas y extraer o imtroducir información. Explicamos brevemente el contenido de cada una de estas tablas para una mejor comprensión posteriormente de las sentencias SQL que se ejecutan:

- **GA_TECNICOS:** tabla que contiene toda la información relativa a los técnicos propios de la empresa (cualificación, situación laboral, localización, estado, ...).

- **GA_AVERIAS:** tabla que contiene toda la información relativa a las averías que han entrado en el sistema Gestor de Actuaciones.

- **GA_PLANIFICACION:** tabla que contiene toda la información relativa a la planificación realizada por el sistema sobre la resolución de cada actuación concreta por un técnico en concreto y en un determinado momento.

- **GA_ORDENESDESERVICIO:** tabla que contiene toda la información relativa a las ordenes de servicio (servicios de instalación) que han entrado en el sistema Gestor de Actuaciones.

- **GA_DATOSEQUIPOSSERVICIOSAAV:** tabla que contiene toda la información relativa a posibles equipos y materiales utilizados en la resolución de una avería.

- **GA_DATOSEQUIPOSSERVICIOSAOS:** tabla que contiene toda la información relativa a posibles equipos y materiales utilizados en una determinada instalación en el domicilio de un cliente.

- **GA_DATOSLINEASAOS:** tabla que contiene toda la información relativa a posibles equipos y materiales utilizados en una determinada instalación referente a los distintos tipos de líneas que han podido instalarse. Los datos que hacen referencia a los equipos terminales en los domicilios particulares de los clientes se encuentran en la tabla anterior.

Oracle permite un control más fino del uso de espacio en disco a través de estructuras de almacenamiento lógico denominadas **bloques de datos**, **extensiones** y **segmentos**:

- **bloques:** en el nivel más fino de granularidad, los datos de las bases de datos Oracle se almacenan en bloques de datos. Un bloque de datos corresponde a un número específico de bytes de espacio físico en disco. El tamaño de un bloque es especificado durante la creación de la base de datos. Una base de datos usa y asigna el espacio libre de memoria por medio de bloques de datos.

- **extensiónes:** el siguiente nivel de espacio lógico en las bases de datos se llama extensión. Una extensión es un número específico de bloques de datos contiguos, obtenidos en una única asignación y utilizados para almacenar un tipo específico de información.

- **segmentos:** el siguiente nivel de almacenamiento lógico es conocido como segmento. Un segmento es un conjunto de extensiones asignadas a una determinada estructura lógica (segmento de datos, segmento de índices, segmento de roolback, segmento temporal,...).

Oracle asigna dinámicamente espacio de memoria cuando las extensiones de un segmento están siendo usadas. Así pues, cuando las extensiones existentes de un segmento están llenas, Oracle asigna otra extensión a dicho segmento cuando sea necesario. Como las asignaciones de extensiones se realizan a medida que se van necesitando, las extensiones de un segmento podrían no estar localizadas contiguamente en disco. Se admite cualquier tipo de posible asignación dinámica de memoria.

3.5.2.3 Estructura física de la Base de Datos

Todas las bases de datos Oracle tienen uno o más **datafiles** físicos. Un datafile es un fichero de datos que tiene existencia física. Estos datafiles contienen todos los datos de la base de datos. Los datos de las estructuras lógicas de la base de datos, como tablas e índices, son físicamente almacenados en los ficheros de datos asignados para una determinada base de datos. Las características de estos ficheros de datos son:

- Cada fichero de datos puede ser asociado con una única base de datos.

- Podemos establecer determinadas características de los ficheros de datos que les permitan extenderse automáticamente cuando la base de datos lo requiera.

- Uno o más ficheros de datos forman una unidad lógica de la base de datos conocida como tablespace.

Los datos de un archivo de datos se leen, cuando es necesario, durante las operaciones normales de la base de datos y son almacenados en la memoria caché de Oracle. Si un usuario quiere acceder a los datos de una tabla y la información solicitada no se

encuentra en la memoria caché, dichos datos se leen de los archivos de datos apropiados y se almacenan en memoria. Los nuevos datos o la modificación de los ya existentes no es necesario escribirlos inmediatamente a un fichero de datos. Para reducir la cantidad de accesos a disco y aumentar el rendimiento, los datos son almacenados en memoria y, posteriormente, son llevados de una sola vez a los archivos de datos apropiados, operación que es llevada a cabo por un proceso de Oracle denominado **DBWn**.

3.5.2.4 Ficheros de control

Todas las bases de datos Oracle tienen un fichero de control. Este fichero contiene entradas que especifican la estructura física de la base de datos. Entre otros, contiene los siguientes tipos de información:

- **Nombre de la base de datos.**
- **Nombre y localización de ficheros de datos.**
- **Time stamp de creación de la base datos.**

Cada vez que se crea una instancia de una base de datos Oracle, su fichero de control se usa para identificar la base de datos y los ficheros que se deben abrir para poder realizar operaciones con ella. Si la estructura física de la base de datos es alterada (por ejemplo, con la creación de un nuevo fichero de datos), Oracle modifica automáticamente el fichero de control para reflejar los cambios.

3.5.3 Structured Query Language (SQL)

SQL es el lenguaje de programación que define y manipula la base de datos. Las bases de datos SQL son bases de datos relacionales; esto significa simplemente que los datos se almacenan en función de un conjunto de relaciones. Una base de datos puede tener una o más tablas. Cada tabla tiene columnas y filas. Se puede definir y manipular datos en una tabla usando sentencias SQL. Como una parte de SQL encontramos el **lenguaje de definición de datos** (**DDL**), el cual incluye sentencias para crear y alterar bases de

datos y tablas. También se incluye dentro del SQL el **lenguaje de manipulación de datos** (**DML**) incluye sentencias para actualizar, borrar y obtener datos de una tabla. Todo ello no son más que múltiples sentencias para operar con las bases de datos.

3.5.4 Creación de usuarios

Cuando creamos un user: (como DBA):

Create user <nom_user> identified by <passwd>

default tablespace <ts_datos>

temporary tablespace <ts_temp>;

- **default tablespace**: indicamos el almacenamiento por defecto para ese usuario, es decir, que cuando dicho usuario cree cualquier objeto se almacene el tablespace indicado.
- **temporary tablespace**: indicamos el tablespace que van a usar todos lo objetos de dicho usuario cuando necesiten espacio temporal.

Para acceder a información acerca de usuarios se puede acceder a cualquiera de las siguientes tablas:

- **DBA_USERS**
- **ALL_USERS**
- **USER_USERS**

Pero, además de crear un usuario le tenemos que dar privilegios para que pueda hacer cosas como conectarse, crear objetos, consultar tablas del diccionario o consumir recursos.

grant connect, resource to <nombre_user>;

De ésta manera si el usuario crea un objeto, por ejemplo:

create table x(....);

No se produce ningún error ya que el usuario tiene privilegios para crear objetos.

3.5.5 Correspondencia entre tablespaces y espacio físico

Cada tablespace al crearse, por lo menos se le debe asignar 1 fichero de SO llamado datafile. Cuando un usuario crea un objeto en el tablespace, se crea en el tablespace un segmento y en el datafile se crea una extensión (**initial**). Esta extensión es un conjunto de bloques de Oracle contiguos en disco. Cuando se empieza a meter datos en las tablas, se empiezan a rellenar los bloques de la extensión. Cuando se acaba el espacio asignado a su extensión (asociada a su segmento) se le intenta dar otra extensión (**next**). Si lo consigue el usuario no se entera de la operación. Hay veces que el sistema no puede asignarle otra extensión porque no queda espacio en disco. Entonces tendrá que añadir otro datafile al tablespace. Si llega otro usuario, también se creará una extensión en el mismo datafile. Esto es un problema ya que en cada datafile puede haber una mezcla de muchas cosas, por lo que si el datafile se pierde o se corrompe se pueden llegar a perder muchos objetos o volverse inconsistentes (ya que les falta un trozo). Esto puede forzar a hacer una recuperación incluso de toda la BD.

3.5.6 Creación de un Tablespace

Para crear un tablespace se hace lo siguiente:

create tablespace <ts_name> datafile '/users/.../x_0.dbf' size 6k default storage (initial 10k next 10k);

Para información acerca de los tablespaces consultaremos las tablas:

DBA_TABLESPACES

Y para consultar información acerca de los datafiles:

DBA_DATA_FILES

3.5.7 Arranque y Parada de una Base de Datos ORACLE

Para ver con que **BD**, mejor dicho, instancia de **BD**, estoy trabajando, hay que consultar una variable de entorno de **Unix** (variable del **regedit** en W95 y NT) que se llama **ORACLE_SID**.

echo $ORACLE_SID

Para cambiar su valor:

ORACLE_SID=prueba

Para que el cambio sea efectivo (del modo anterior no afecta a los procesos hijos):

export ORACLE_SID

ARRANQUE

Se realiza mediante el comando **startup**. Consta de 3 fases:

- **Arrancar la instancia de Oracle.**

  Se lee el fichero de inicialización de parámetros que es el init<sid>.ora. Este fichero se lee de un directorio hijo que tiene que ser siempre el mismo ya que Oracle lo busca automáticamente (en éste caso Oracle_home/dbs). En el init.ora nos encontramos con una entrada, ifile, que apunta a un fichero de configuración (config<sid>.ora) que contiene los parámetros más o menos fijos de la BD. En general, son parámetros que definen lo que es una instancia de BD.

- **Montar la BD.**

  Leer los ficheros de control, asociar toda la estructura de ficheros a la instancia que ya se ha arrancado, y además se hace una comprobación.

- **Abrir la BD.**

Problemas que pueden aparecer cuando intentamos arrancar la base de datos:

- **Número máximo de ficheros abiertos dentro de la maquina.**

  Existe un parámetro de la máquina que indica el número de ficheros abiertos que puede soportar (la máquina). Puede que al abrir la BD se sobrepase ése número, por lo que no nos deja abrirla.

  Solución: aumentar el número máximo de ficheros abiertos. Después hay que recompilar el kernel y volver a arrancar la máquina (esto lo suele hacer el administrador de SO).

- **Número máximo de procesos dentro de la máquina.**

  Ocurre lo mismo que en el punto anterior pero con procesos.

- **Número máximo de semáforos que se pueden usar en la máquina (ssmax).**

  Por cada proceso, se necesitan 1 o 2 semáforos, entonces tenemos el mismo problema que en los puntos anteriores.

- **Tamaño de memoria asignada a Oracle.**

  Ocurre cuando asignamos demasiada memoria a Oracle, ya que en la maquina se estarán ejecutando más cosas y no las dejamos memoria suficiente.

Parámetros del **startup**:

- **nomount:** sólo arranca la instancia. Sirve para solucionar determinados problemas con procesos o memoria.
- **mount:** arranca la instancia y monta la BD. Sirve para solucionar problemas con ficheros (no abre los ficheros, sólo los relaciona).
- **open:** es igual al startup sin parámetros.
- **force:** fuerza a arrancar una BD (aunque ya esté arrancada).

Si estamos en "estado" **nomount**, para pasar a montar la instancia:

alter database mount;

Si estamos en "estado" **mount**, para pasar a abrir la BD:

alter database open;

También existe **alter database nomount** por si queremos desmontar la BD a mano. Si quiero que Oracle busque en otro directorio el fichero de inicializacion, se especifica el path completo en el startup:

startup pfile = /users/.../init.ora <parametros>

SHUTDOWN (Parada)

Ejecuta las tres fases del **startup** pero en orden inverso.

Parámetros del **shutdown** (sin parámetros, **shutdown** no tira abajo la BD hasta que todo el mundo se desconecta):

- **immediate:** lo primero que hace es cerrar todas las conexiones de usuarios, y después hace el proceso normal de **shutdown**.
- **abort:** tira la instancia de BD directamente (ni cierra ficheros, ni desmonta la BD). Por lo tanto, no escribe nada a ficheros y no se podrá hacer una recuperación automática de los ficheros de **redolog**. Se usa cuando no se puede usar ninguno de los demás.

El **startup** y el **shutdown** escriben en unos **ficheros de alerta** (**log**). Puede que no se pueda abrir o cerrar la BD porque no puede escribir en los ficheros de alerta (porque hay muchos, por ejemplo).

3.5.8 Memoria

Existen 2 tipos de memoria en Oracle:

- **Db block buffer cache:** caché de bloques de datos (su tamaño se determina mediante el parámetro de inicialización DB_BLOCK_BUFFERS).

- **Caché de librería o de SQL compartido (Shared SQL area) o SQL//PL/SQL:** aquí se guarda las sentencias ejecutadas, más el plan de ejecución (análisis de la sentencia + camino elegido por el optimizador). (su tamaño viene determinado por el parámetro de inicialización **SHARED_POOL_SIZE**).

3.5.9 Análisis de rendimiento de sentencias SQL

La herramienta a usar es el **SQL Trace**, y gracias a la utilidad **TKPROF** se consigue un fichero legible.

Cada sentencia tiene 3 partes:

- **parse(1):** el optimizador de Oracle, lo primero que hace es comprobar si el texto exacto está en la caché de SQL. Si está no hay que realizar el plan de ejecución. Si no está:

  - ✓ Mira la sintaxis de la sentencia para ver si está bien escrita.
  - ✓ Cuando está bien, comprueba a qué objeto hace referencia la sentencia y según la estructura del objeto elige un camino.
  - ✓ Una vez hecho todo, escribe todo lo que ha hecho en el área de SQL compartido.

- **Parse(2):** según los manuales, recupera los bloques de disco, pero en realidad, sólo da valores cuando se producen inserts (o sea, que será al revés, mira escrituras).
- **fetch:** trae los bloques de disco y lee de memoria.

Parámetros:

- **count:** hace referencia a una zona intermedia que se va rellenando según recuperamos filas. Por defecto, ése valor es igual a 14 (es decir, el buffer contiene espacio para 14 filas de la tabla).
- **rows:** nos da el número de filas devueltas. Entonces, si una select devuelve 22 filas, el valor de count = 2, ya que 14*2>22, por lo tanto el buffer sólo se llena 2 veces para devolver todas las filas.
- **cpu:** tiempo de cpu consumido para realizar la sentencia en seg. (depende de la máquina y del número de usuarios conectados)
- **elapsed:** tiempo de cpu + tiempo de E/S en segundos. (depende de la máquina y del número de usuarios conectados)
- **disk:** cuantos bloques se han traído de disco.
- **query:** aciertos a memoria (bloques). Pero cuando hace lecturas a disco, también las incluye. Menos en una ocasión, que es cuando se lee toda la tabla de disco (disk>query por 1 bloque), query > disk siempre.

Existe en Oracle, además del optimizador basado en reglas, el optimizador basado en costes. Este optimizador se basa en las estadísticas generadas (si hay estadísticas, Oracle elige el basado en costes). Dependiendo de la aplicación con la cual se trabaje se opta por uno u otro.

3.5.10 Accesos en ORACLE

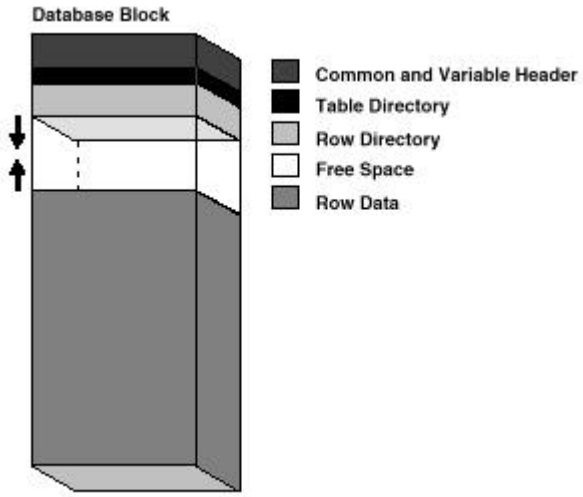
Existen 3 formas de localizar una fila en Oracle:

- **por ROWID (la más rápida)**
- **por índice**
- **por recorrido completo de tabla.**

La cláusula **distinct** hay que evitarla siempre que se pueda porque crea una tabla temporal y retarda bastante.

3.5.11 Formato de bloque de ORACLE

Figura 21.- Estructura de bloque en Oracle.



En la **Figura 21** tenemos la estructura de bloque para Un sistema de gestión de bases de datos de la firma Oracle. En esta figura vamos a destacar algunos aspectos:

- **Directorio tablas:** si la tabla no es **clustered** sólo hay 1 entrada para 1 tabla y si el bloque pertenece a una tabla **clustered**, habrá más de una entrada para más de 1 tabla.
- **Directorio filas:** es fijado por los parámetros **inittrans** (cuantas transacciones pueden estar dentro del bloque) y **maxtrans** (máximo de transacciones).

3.5.12 Chained rows y Migrated rows

Chained rows

Cuando no tenemos suficiente espacio en un bloque para **updates**, se produce una fila encadenada o migrada. Es decir, cuando no tiene espacio libre lo coge de 1 bloque que si lo tenga (pero de su zona de inserciones). Entonces en la fila original se guarda un puntero a donde se encuentra el campo migrado (mejor dicho, la parte del campo o el campo entero que se ha migrado).

El problema que genera éste efecto es un problema de rendimiento porque para una fila tenemos que ir accediendo a varios bloques, con lo que ello supone.

Migrated rows

Este fenómeno se produce cuando la **update** (es cada nueva actualización realizada sobre los datos de una tabla) es tal que se tiene que llevar toda la fila a un nuevo bloque. Entonces en la ubicación original sólo queda el puntero al nuevo bloque.

3.6 SQL (Introducción a las consultas)

Las consultas son operaciones que se realizan sobre los datos de una base de datos. Estas operaciones pueden ser de diversos tipos:

- **Consultas de selección de datos:** permiten recuperar los datos almacenados en las tablas en el formato y orden adecuados. Además permiten filtrar y agrupar la información. El resultado de estas consultas consiste en una tabla "virtual": una tabla que físicamente no ocupa espacio (porque trabaja sobre los datos originales de las tablas sobre las que se define), pero que permite ser manejada de la misma forma que una tabla real.

- **Consultas de inserción de datos:** permiten agregar registros a una tabla.
- **Consultas de modificación:** permiten modificar los valores de los campos de los registros de una tabla.
- **Consultas de borrado:** permiten eliminar registros de una tabla.
- **Consultas de creación de tabla:** permiten crear nuevas tablas cuyos campos y registros se obtienen a partir de los almacenados en otras tablas.

Dentro de las consultas de selección podemos resaltar algunos grupos importantes:

- **Consultas de selección simple:** permite filtrar tuplas y añadir o descartar campos de los registros. Se utilizan para crear "vistas": consultas que se utilizan como tablas para restringir el acceso a los datos a determinados usuarios.
- **Consultas de unión:** permiten relacionar los datos de distintas tablas a través de campos clave.
- **Consultas de agrupamiento:** permiten obtener resultados estadísticos de conjuntos de registros, como medias de un campo, totales, etc.

Las consultas a los datos de las bases de datos se hacen a través de los denominados "**lenguajes de consulta**". El más utilizado de este tipo de lenguajes es el **SQL** (**Standard Query Language**). SQL es un estándar que aparece en multitud de sistemas sin variaciones, y es el lenguaje de consultas comúnmente utilizado para tratar con bases de datos Oracle.

3.6.1 Introducción al SQL

Una consulta SQL está compuesta por una instrucción SQL que define esa consulta. Se trata de un comando que puede ocupar cuantas líneas de texto se desee, terminado en punto y coma (;). SQL es un lenguaje sencillo e intuitivo: las consultas se asemejan al lenguaje natural.

Existen algunas palabras reservadas, como en cualquier lenguaje: **SELECT**, **INSERT**, **DELETE**, **UPDATE**, **SET**, **WHERE**, **IN**, **DISTICT**, **GROUP**, **ORDER**, **BY**, etc.

3.6.1.1 Consultas de selección simple

La consulta más simple posible consiste en la selección de campos y registros de una tabla. Se identifican los campos que nos interesan y una condición que deben cumplir los registros seleccionados. El resultado es una tabla que es un subconjunto de la original.

El formato genérico de este tipo de consultas es:

SELECT <lista de campos> FROM <tabla> WHERE <condición>;

Esta instrucción recupera ciertos campos de los registros de una tabla que verifican una condición. La cláusula WHERE es opcional. Si se omite, se seleccionan todos los registros (se supone que la condición es siempre verdadera).

SELECT <lista de campos> FROM <tabla>;

Si nos interesan todos los campos podemos utilizar el símbolo ***** para identificar a la lista completa:

SELECT * FROM <tabla> WHERE <condición>;

Si no, podemos especificar varios campos identificándolos por sus nombres y separándolos por comas (,).

SELECT campo1, campo2, ..., campoN FROM <tabla> WHERE <condición>;

Vamos a ver algún ejemplo de uso de este tipo de consultas que podemos encontrar en el código desarrollado para este proyecto.

Para obtener información relativa a actuaciones asignadas a un técnico (ya sea de aquella en la que está trabajando en ese momento o de las que tiene planificadas para él y, por lo tanto, pendientes de realizar) se realiza la siguiente consulta a la base de datos del sistema:

```
SELECT  entry_id_actuacion, telefono_n_os, estado,
        area_planta_tipo_mercado, fecha_inicio_planificada,
        fecha_inicio_planificada + duracion_remanente
FROM ga_planificacion
WHERE estado IN ('En Curso', 'Pendiente');
```

Otra consulta que nos encontramos es aquella de la cual obtenemos el **número de matrícula** del técnico que se está conectando al sistema usando el número de teléfono del terminal móvil que usa para conectarse (dicho número de teléfono se encuentra asociado, en la base de datos del sistema, a ese técnico concreto). La consulta sería la siguiente:

```
SELECT  nummatchar, literalrealizador
INTO :matricula,:tecnico
FROM ga_realizador
WHERE telfcontactogrupo = :telefmovil;
```

Una vez tenemos identificado al técnico, podemos identificar la actuación en la que está trabajando actualmente dicho técnico (**entry_id_actuacion** *)*, usando para ello la siguiente consulta:

SELECT entry_id_actuacion

INTO :entry_id_actuacion

FROM ga_planificacion

WHERE (estado = 'En Curso') AND (tecnico_grupo = :tecnico);

Es posible consultar, desde una base de datos, una tabla que pertenezca a otra base de datos. En este caso utilizaremos la sintaxis:

SELECT <lista de campos> FROM <tabla> IN <base de datos>;

La cláusula IN permite especificar otra base de datos como origen de la tabla.

Esta estructura permite también acceder a datos que se encuentren almacenados en otras bases de datos que no sean del mismo tipo que la que nosotros hemos utilizado, siempre y cuando Oracle se encuentre correctamente instalado y configurado.

3.6.1.2 Adición de campos

Podemos generar consultas en las que aparezcan nuevos campos. En tal caso podemos utilizar la sintaxis "<expresión> AS <nombre campo>" para cada columna añadida como si se tratara de un campo más de la tabla:

SELECT <lista campos>, <expresión> AS <nombre campo> FROM <tabla> WHERE <condición>;

3.6.1.3 Operadores y expresiones

Las expresiones en SQL son semejantes a las utilizadas en la mayoría de los lenguajes. Vemos, en la **Tabla 6,** un resumen de los operadores usados en SQL que coinciden con los usados en otros muchos lenguajes.

Tabla 6.- Operadores comunes usados en SQL.

Operador	Significado	Operador	Significado
+	Suma aritmética	"	Delimitador de cadenas
-	Resta aritmética	&	Concatenación de cadenas
*	Producto aritmético	=	Comparador igual
/	División aritmética	<>	Comparador distinto
mod	Módulo	>	Comparador mayor
AND	AND lógico	<	Comparador menor
OR	OR lógico	>=	Comparador mayor o igual
NOT	Negación lógica	<=	Comparador menor o igual
XOR	OR exclusivo lógico	( )	Delimitadores de precedencia

Sin embargo podemos destacar, en la **Tabla 7**, algunos operadores propios de SQL.

Tabla 7.- Operadores propios de SQL.

Operador	Significado
IS NULL	Comparador con valor nulo. Indica si un campo se ha dejado en blanco.

Operador	Significado
IS NOT NULL	Comparador con valor no nulo. Indica si un campo contiene un valor, y no se ha dejado en blanco.
LIKE	Comparador de semejanza. Permite realizar una comparación de cadenas utilizando caracteres comodines: ? = Un carácter cualquiera * = Cualquier combinación de caracteres (incluido ningún carácter)
BETWEEN...AND	Comparador de pertenencia a rango.
[ ]	Delimitadores de identificadores. Sirven para delimitar los nombres de objetos (campos, tablas, etc.) cuando éstos incluyen espacios.

3.6.1.4 Valores repetidos

Una consulta de selección puede recuperar tuplas idénticas. Para evitar obtener tuplas repetidas, podemos utilizar el modificador **DISTINCT**:

SELECT DISTINCT <lista campos> FROM <tabla>;

Ahora la consulta no devolverá tuplas repetidas. Existe otro modificador, **DISTINCTROW**. A diferencia del anterior, **DISTINCTROW** no tiene en cuenta tuplas que estén completamente duplicadas en la tabla de origen (y no sólo para los campos seleccionados).

3.6.1.5 Ordenación de registros

SQL permite especificar que las tuplas seleccionadas se muestren ordenadas por alguno o algunos de los campos seleccionados, ascendente o descendentemente. Para ello se dispone de la palabra reservada ORDER BY, con el siguiente formato:

SELECT <lista de campos seleccionados> FROM <tabla> WHERE <condición> ORDER BY <lista de campos para ordenar>;

La lista de campos para ordenar debe ser un subconjunto de la lista de campos seleccionados. Para especificar un orden inverso (decreciente) se emplea la cláusula **DESC** que puede ser incluida tras el nombre del campo por el que se ordena de forma descendente. De la misma forma la cláusula **ASC** ordena de forma ascendente, aunque no es necesario especificarla, ya que es la opción por defecto.

Como ejemplo de esto vemos la consulta extraida anteriormente del código fuente mediante la cual se obtenía información de las actuaciones asignadas a un técnico, la cual habíamos mostramos incompleta y que ahora completamos un poco más añadiéndole la parte referida a la ordenación:

```
SELECT  entry_id_actuacion, telefono_n_os, estado,
        area_planta_tipo_mercado, fecha_inicio_planificada,
        fecha_inicio_planificada + duracion_remanente
FROM  ga_planificacion
WHERE estado IN ('En Curso', 'Pendiente');
ORDER BY estado, fecha_inicio_planificada;
```

3.6.1.6 Agrupamiento de datos

SQL permite definir consultas en la que se ofrecen tuplas que se obtengan como resultado del agrupamiento de varias tuplas. Por ejemplo, valor promedio de un campo, máximo, mínimo, cuenta, etc.

Para este tipo de consultas se proporcionan los operadores que se relacionan en la **Tabla 8**, que se denominan **funciones de agregado**.

Tabla 8.- Funciones de agregado.

Operador	Significado
COUNT(<campo>)	Número de tuplas seleccionadas (excepto las que contienen valor nulo para el campo). Si <campo> es una lista de campos (separados por &) o *, la tupla se cuenta si alguno de los campos que intervienen es no nulo.
SUM(<campo>)	Suma del conjunto de valores contenidos en el campo especificado. Las tuplas con valor nulo no se cuentan.
AVG(<campo>)	Media aritmética del conjunto de valores contenidos en el campo especificado. Las tuplas con valor nulo no se cuentan.
MAX(<campo>)	Valor máximo del conjunto de valores contenidos en el campo especificado. Las tuplas con valor nulo no se cuentan.
MIN(<campo>)	Valor mínimo del conjunto de valores contenidos en el campo especificado. Las tuplas con valor nulo no se cuentan.

El formato de este tipo de consultas es:

SELECT COUNT/SUM/AVG/MAX/MIN (<campo>) AS <nombre> FROM <tabla> WHERE <condición>;

Se pueden incluir varias funciones de agregado en la misma consulta.

En todas las consultas vistas hasta ahora, las funciones de agregado se aplican sobre el conjunto total de registros de una tabla (excepto lo que no cumplen la cláusula WHERE, que son descartados), y el resultado de tales consultas es un único valor. SQL permite crear grupos de registros sobre los cuales aplicar las funciones de agregado, de manera que el resultado es un conjunto de tuplas para cada una de las cuales se ha calculado el valor agregado. Los grupos se componen de varios registros que contienen el mismo valor para un campo o conjunto de campos. El formato es:

SELECT <agregado> AS <nombre> FROM <tabla> WHERE <condición>GROUP BY <lista de campos>;

De esta forma, para cada valor distinto de la <lista de campos> suministrada, se calcula la función de agregado correspondiente, sólo con el conjunto de registros con dicho valor en los campos (los registros que no verifiquen la condición WHERE no se tienen en cuenta).

El agrupamiento de filas impone limitaciones obvias sobre los campos que pueden ser seleccionados, de manera que sólo pueden obtenerse campos resultado de una función de agregado o la combinación de campos que aparezca en la cláusula GROUP BY, y nunca otros campos de la tabla de origen.

3.6.1.7 Filtrado de tuplas de salida

En estas consultas puede aparecer una condición WHERE que permite descartar las tuplas que no deben ser tenidas en cuenta a la hora de calcular las funciones de agregado. Sin embargo WHERE no permite descartar tuplas utilizando como condición el resultado de la función de agregado. La cláusula WHERE no puede contener

funciones de agregado. Para este cometido existe otra cláusula semejante a WHERE, **HAVING**, que tiene el siguiente formato:

SELECT <agregado> AS <nombre> FROM <tabla> WHERE <condición> GROUP BY <lista de campos> HAVING <condición de agregado>;

En resumen: WHERE selecciona las tuplas que intervienen para calcular las funciones de agregado y HAVING selecciona las tuplas que se muestran teniendo en cuenta los resultados de las funciones de agregado.

En todos los casos, la cláusula **ORDER BY** puede ser incluida. Evidentemente esta cláusula afectará únicamente al orden en que se muestran las tuplas resultado, y no al cálculo de las funciones de agregado. Los campos por los cuales puede efectuarse la ordenación sólo pueden ser aquéllos susceptibles de ser también mostrados, es decir, que los campos admisibles en la cláusula ORDER BY son los mismos que sean admisibles en la cláusula SELECT: funciones de agregado y la combinación de campos que aparezca en GROUP BY.

Recordemos el formato de una instrucción SQL de selección con todas las opciones vistas hasta ahora:

SELECT <lista de campos> FROM <tabla> WHERE <condición> GROUP BY <lista de campos>HAVING <condición de agregado> ORDER BY <lista de campos>;

3.6.1.8 Consultas sobre múltiples tablas

Todas las consultas estudiadas hasta el momento se basan en seleccionar tuplas y campos sobre los datos almacenados en una única tabla. SQL también permite obtener resultados a través de la combinación de múltiples tablas. La forma de hacerlo es a través del **enlace o unión** (**join**) de varias tablas a través de claves externas (**claves ajenas**, **foreign keys**). Una clave externa es un campo o conjunto de campos que hacen referencia a otro campos o conjunto de campos de otra tabla. Esta relación habitualmente se establece entre uno o varios campos de una tabla y la clave principal de otra tabla, y la mayoría de las veces va a guardar relación directa con las políticas de integridad referencial definidas.

Producto cartesiano

El origen de las consultas basadas en múltiples tablas es la operación de producto cartesiano, que consiste en una consulta para la que se generan tuplas resultado de todas las combinaciones de los registros de las tablas implicadas.

La forma de obtener una consulta de producto cartesiano es especificando el nombre de las tablas implicadas en la cláusula FROM:

SELECT <lista de campos> FROM <tabla1>, <tabla2>, ...<tablaN>;

El resto de cláusulas estudiadas hasta ahora (WHERE, ORDER BY, GROUP BY, HAVING...) siguen siendo válidas y utilizan el mismo formato. Las listas de campos válidos son ahora cualquiera de los de las tablas utilizadas, como si se tratara de una única tabla en la que existen todos los campos de todas las tablas. Puesto que es posible que existan campos con el mismo nombre en las diferentes tablas, a la hora de nombrar los campos será necesario especificar a qué tabla pertenecen con el formato "<tabla>.<campo>".

Las consultas de producto cartesiano como fin último son poco habituales. Por lo general el producto cartesiano se utiliza como medio para obtener consultas que relacionan varias tablas a partir de claves externas.

Unión (join)

Aunque esta forma de enlazar tablas es correcta, existe otro mecanismo más adecuado para enlazar tablas a través de sus claves externas. Se trata de la operación de unión (join).

La operación de unión básicamente obtiene el mismo resultado que un producto cartesiano filtrado para que sólo se muestren las tuplas en las que coincida la clave externa (condición de join). La diferencia es que se va a emplear una cláusula específica para definir la operación, en lugar de la genérica WHERE, lo que permitirá al SGDB identificar el tipo de operación y proporcionar algunas ventajas sobre el resultado (que veremos más adelante).

La sintaxis para una operación de unión es:

SELECT <lista de campos> FROM <tabla1> INNER JOIN <tabla2> ON <tabla1>.<campo1>=<tabla2>.<campo2>;

Esta es la unión equiparable al producto cartesiano filtrado como:

SELECT <lista de campos> FROM <tabla1>, <tabla2>WHERE <tabla1>.<campo1> = <tabla2>.<campo2>;

En general para cualquier número de tablas, la unión se realiza mediante anidamiento de uniones. La sintaxis para tres tablas es:

SELECT <lista de campos> FROM <tabla1> INNER JOIN (<tabla2> INNER JOIN <tabla3> ON <tabla2>.<campo2> =<tabla3>.<campo3>) ON <tabla1>.<campo1> = <tabla2>. <campo2> ;

Y para N tablas:

SELECT <lista de campos> FROM <tabla1> INNER JOIN (<tabla2> INNER JOIN(...<tablaN-1>INNER JOIN<tablaN>ON <tablaN-1>. <campoN-1> = <tablaN>.<campoN>...) ON <tabla2>.<campo2> = <tabla3>.<campo3>) ON <tabla1>.<campo1> = <tabla2>.<campo2>;

3.6.1.9 Consultas de inserción

Las consultas de inserción permiten añadir registros a una tabla. para este tipo de consultas se requiere:

- **Una tabla a la que añadir los datos.**
- **Una consulta de selección de la que obtener los datos que se añaden, o bien una lista de los valores a insertar.**

El formato SQL de una consulta de inserción de datos utilizando una consulta de selección como origen de los datos es:

INSERT INTO <tabla destino> (<lista campos destino>)SELECT <lista campos origen>FROM <tabla origen>;

La lista de campos destino es una lista de campos separados por comas; la lista de campos origen es una lista al estilo de la empleada en una consulta de selección cualquiera. Cada campo de la lista de origen debe corresponderse con otro en la lista de destino, en el mismo orden, de manera que las tuplas obtenidas en la consulta se añaden a la tabla de destino. Los campos no especificados serán llenados con los valores por defecto, a menos que no tengan ningún valor predeterminado, en cuyo caso quedarán vacíos (con valores nulos).

La parte de la consulta de selección puede contener todas las opciones estudiadas: agrupamiento, funciones de agregado, ordenamiento de tuplas, condiciones de filtrado, etc.

Para añadir datos a una tabla sin utilizar otra tabla o consulta como origen de datos, se puede utilizar la siguiente sintaxis:

INSERT INTO <tabla destino> (<lista campos destino>)VALUES <lista campos origen>;

Como en el caso anterior, debe existir una correspondencia y compatibilidad exacta entre la lista de campos de origen y la lista de campos de destino (correspondencia uno a uno). De no ser así se pasaría a un estado de inconsistencia y el resultado sería un error.

3.6.1.10 Consultas de creación de tabla

Este tipo de consultas son idénticas a las de inserción excepto por que la tabla de destino de los datos especificada no existe, y se crea en el momento de ejecutar la consulta. Si la tabla existiese, estas sentencias únicamente insertarían los nuevos datos en ella.

3.6.1.11 Consultas de actualización

Las consultas de actualización de datos permiten modificar los datos almacenados en una tabla. Se trata de modificar los valores de determinados campos en los registros que cumplan una determinada condición. La sintaxis de este tipo de consultas es:

UPDATE <tabla>SET <campo> = <nuevo valor>, <campo> = <nuevovalor>, <campo> = <nuevo valor>WHERE <condición>;

3.6.1.12 Consultas de borrado

Las consultas de actualización de datos permiten eliminar tuplas de una tabla de forma selectiva. Esta selección se consigue señalando aquellos registros los registros que cumplan una determinada condición. Se trata de una actualización a un valor que indique la ausencia de datos. La sintaxis de este tipo de consultas es:

DELETE [<tabla>.*] FROM tabla WHERE <condición>;

Las consultas de borrado no permiten borrar campos; sólo tuplas completas, es decir, borrar todos los campos de una entrada de la tabla. Por eso la parte <tabla>.* es opcional. Para eliminar el valor de los campos debe utilizarse las consultas de actualización. Con ello se consigue el objetivo perseguido actualizando el valor de los campos a NULL.

Si no se especifica ninguna condición dentro de las sentencias que forman las consultas de borrado, se eliminan todas las tuplas que existan en la tabla. No se elimina la tabla, ya que la estructura va a seguir existiendo, aunque no contenga ningún registro.

3.6.1.13 Consultas anidadas

Se permite el anidamiento de consultas. La forma habitual de utilizar este mecanismo es emplear el resultado de una consulta para seleccionar valores de otra. La **subconsulta** se encierra entre paréntesis. Se pueden anidar tantas consultas como se quiera. Las cláusulas que permiten enlazar la consulta principal y la subconsulta son las siguientes:

- **Cualquier comparador (>, <, =, etc...).**

  En este caso, la subconsulta debe proporcionar un resultado único con el que realizar la comparación.

- **Cualquier comparador seguido de ALL, ANY o SOME.**

  En este caso, la subconsulta puede proporcionar múltiples tuplas como resultados.

  - ✓ **ALL:** se seleccionan en la consulta principal sólo los registros que verifiquen la comparación con todas las tuplas seleccionadas en la subconsulta.
  - ✓ **ANY:** se seleccionan en la consulta principal sólo los registros que verifiquen la comparación con alguna de las tuplas seleccionadas en la subconsulta.
  - ✓ **SOME:** es idéntico a ANY.

- **El nombre un campo + IN.**

  En este caso la subconsulta puede proporcionar múltiples tuplas como resultados, y se seleccionan en la consulta principal los registros para los que el valor del campo aparezca también en le resultado de la subconsulta. Por tanto, el efecto es el de una doble consulta. Es equivalente a utilizar "= ANY". Se puede utilizar NOT IN para conseguir el efecto contrario, equivalente a "<> ALL".

- **La cláusula EXISTS.**

  El resultado de la consulta puede proporcionar múltiples tuplas. La condición evaluada es que en la subconsulta se recupere alguna tupla (EXISTS) o no se recupere ninguna tupla (NOT EXISTS).

Para ver un ejemplo de anidamiento de consultas, volvemos a la consulta de obtención de información de actuaciones, que ya habíamos ordenado, y vamos a mostrarla tal y como se necuentra recogida en el código fuente, donde existe un anidamiento:

```
SELECT  entry_id_actuacion, telefono_n_os, estado,
        area_planta_tipo_mercado, fecha_inicio_planificada,
        fecha_inicio_planificada + duracion_remanente
FROM ga_planificacion
WHERE estado IN ('En Curso', 'Pendiente') AND
      fecha_inicio_planificada IS NOT NULL AND
      ( tecnico_grupo = ( SELECT literalrealizador
                          FROM ga_realizador
                          WHERE nummatchar =:     matricula) OR
      tecnico_grupo = ( SELECT gr.literalrealizador
                          FROM ga_realizador gr, ga_realizador te
                          WHERE te.nummatchar = :matricula AND
                          gr.codigogrupo = te.grupotecnico))
ORDER BY estado, fecha_inicio_planificada;
```

A veces es necesario utilizar los valores de los campos de la consulta principal en la subconsulta. En tal caso es necesario identificar la tabla del campo consultado utilizado un nombre y **AS**.

3.6.1.14 Consultas de tabla de referencias cruzadas

Las consultas de tabla de referencias cruzadas permiten crear un tipo de tabla en el que tanto los títulos de fila como los de columna se obtienen a partir de los datos de una tabla.

3.6.1.15 Consultas específicas de SQL

Hay dos tipos de consultas específicas de SQL: de concatenación de datos y de definición de datos.

Consultas de concatenación de tablas

Este tipo de consultas se denominan "**de unión**", aunque las llamaremos "**de concatenación**" para no confundirlas con las de **JOIN** (que ya hemos denominado "de unión").

Las consultas de concatenación de tablas permiten obtener una tabla a partir de los datos de varias, pero no como se hace en el producto cartesiano, sino al final de la tabla, como si se añadiran los datos de las demás tablas a los que ya hay en la primera.

La sintaxis es:

SELECT <lista de campos>FROM <tabla 1>UNION [ALL] SELECT <lista de campos>FROM <tabla 2>;

La cláusula opcional **ALL** permite obtener registros duplicados: si se omite no aparecen y si se especifica, se mostrarán sólo valores únicos. Cada consulta de concatenación debe devolver el mismo número de campos, y en el mismo orden. Se necesita que los campos correspondientes tengan tipos de datos compatibles (que se puedan convertir entre sí). Si los nombres de los campos correspondientes no coinciden o deben ser cambiados, hay que utilizar la cláusula **AS** de forma similar en todos los SELECT.

La cláusula **ORDER BY** debe especificarse al final de la consulta, afecta a la consulta completa y sólo puede aplicarse sobre campos mostrados en la selección. El resto de cláusulas (WHERE, GROUP BY, etc.) pertenecen a cada SELECT y se especifican como en cualquier otra consulta.

Si existen más de dos tablas concatenadas, el criterio **UNION** o **UNION ALL** utilizado será el último especificado.

Consultas de definición de datos

Las consultas de definición de datos se utilizan para crear tablas, modificar definiciones de tablas, borrar tablas, crear índices y borrar índices. Ya hemos estudiado otras consultas de creación de tablas. Sin embargo este tipo de consultas permite crear tablas vacías, haciendo una especificación precisa de las características de la tabla.

3.6.1.16 Modificación y acceso a los datos de una consulta. Vistas.

La utilización de consultas en bases de datos persigue dos objetivos:

- **La obtención de resultados sobre los datos almacenados.**

  Es lo que hemos visto hasta ahora. Mediante consultas a la base de datos conseguimos extraer toda aquella información en la que estemos interesados.

- **La generación de vistas.**

Las vistas son consultas de selección que se utilizan como si se tratara de tablas. De forma transparente al usuario, las vistas muestran el contenido de una tabla con un formato, orden y contenido adecuado a las necesidades del usuario.

Una consulta se puede presentar a casi todos los efectos de la misma forma que una tabla. Se pueden hacer consultas sobre consultas, añadir , modificar o eliminar datos

sobre las presentación del resultado de una consulta, crear formularios e informes sobre consultas (en vez de tablas), etc.

Si embargo existe una limitación: determinadas operaciones no se permiten sobre determinadas consultas empleadas como vistas. Esta limitación está impuesta por la posibilidad o imposibilidad de que se inserte de forma automática los valores a los que no se tiene acceso mediante la vista.

Es importante reseñar que las tuplas que se añadan pueden no verificar la condición WHERE o la cláusula ORDER BY. Sin embargo esto no representa una falta a ninguna regla de integridad y es perfectamente legal. La próxima vez que se reconsulte la vista, las tuplas que no verifiquen la condición no volverán a aparecer, pero estarán en la tabla original.

Los siguientes tipos de consultas pueden funcionar como vistas:

- **Consultas de selección simple, con o sin condición WHERE, que añadan o filtren campos, con cualquier tipo de orden.**
- **Consultas de unión utilizando INNER JOIN.**

Los siguientes tipos de consultas no pueden funcionar como vistas debido a la imposibilidad de reconocer la ubicación de los datos en la tabla original a partir de los de la vista:

- **Consultas de unión basadas en producto cartesiano (y filtradas con WHERE) que no utilizan INNER JOIN.**
- **Consultas con funciones de agregado (utilizando GROUP BY).**
- **Consultas de concatenación de tablas.**
- **Consultas que no sean de selección: inserción, borrado, modificación, definición de datos, etc.**

4. ASPECTOS DE IMPLEMENTACIÓN

En esta parte del documento se describe el entorno para el cual se ha desarrollado el proyecto. Se da una relación del hardware y el software empleado tanto en la aplicación de la que es objeto el proyecto, como en el sistema general en el cual se encuentra englobada. También se analizará los ámbitos de uso de esta aplicación así como las herramientas utilizadas en el desarrollo de la misma.

4.1 Descripción del entorno de trabajo

Como ya se ha comentado con anterioridad el objeto de este proyecto es la creación de páginas WEB para el sistema general de forma tal que los técnicos de la empresa de telefonía puedan acceder a ellas a través de sus teléfonos móviles.

Lo primero que necesitamos para el desarrollo de nuestra aplicación es un servidor WEB. El que se ha utilizado en este caso concreto es el **Netscape Enterprise Server 3.62.**

Para el desarrollo del software necesario se ha hecho uso de un PC con microprocesador **Pentium III a 450 MHz**. El sistema operativo usado en este PC ha sido **Windows NT** (Microsoft).

La máquina sobre la que montamos la Base de Datos del sistema es de la marca **Sun Microsystem (E10000)** y presenta las siguientes características :

- **1 x E10000 con 1 único dominio.**
- **40 CPUs 366 MHz.**
- **20 GB de RAM.**
- **2 dispositivos D1000, con 4 unidades de 4.2 GB cada uno.**

Esta máquina correrá bajo sistema operativo **Unix (Solaris)**.

De igual forma, el software de **Netscape** para el servidor WEB se ejecutará sobre máquinas de este tipo, utilizando para ello la versión comercial para **Solaris**.

La Base de datos del sistema sobre la que trabajamos es de la marca **Oracle**, y más concretamente la **Oracle 8.0**.

Como cliente del sistema general **Gestor de Actuaciones** tenemos una aplicación gráfica (denominada Gestor de Actuaciones) desarrollada sobre **ARS 3.2**, haciendo uso del API que viene suministrado con esta herramienta (**servidor de Workflow**).

Para el desarrollo de páginas **WEB** a las cuales podamos acceder vía **GSM** usando terminales móviles se ha hecho uso de un editor de texto normal (más concretamente del **vi**, que viene integrado con el sistema operativo **Unix**). En muchos casos y, para facilitar la modificación de los ficheros que contienen el código fuente y que se encuentran en la máquina donde se ejecuta el servidor, se ha utilizado un editor de texto denominado **Ultraedit**, el cual nos permite el poder realizar transferencias de ficheros (**FTP**) con relativa facilidad. Esto es así ya que nuestra aplicación se basa en la ejecución en el servidor de **CGIs** que usan como lenguaje de programación **C++**, por lo que resulta cómodo usar una herramienta que nos permita modificar fácilmente los ficheros de código fuente en el servidor, donde se compilarán y se linkarán con las librerías existentes para crear los ejecutables que habrá de lanzar el servidor atendiendo a las peticiones que le lleguen. Estos ejecutables simplemente darán como salida las páginas WEB solicitadas por los navegadores de los clientes.

Para poder visualizar las páginas WEB creadas necesitamos acceder al servidor desde nuestro cliente a través de un navegador. Para cargar páginas WEB desde el PC cliente usamos **Internet Explorer 5.0** y **Netscape Navigator 4.7**. Para visualizar las páginas WEB desde terminales móviles necesitamos que estos dispongan de un navegador. En este punto debemos de distinguir entre dos casos diferentes, según el tipo de páginas WEB que se pretenda visualizar:

- **HTML:** Estas páginas WEB han sido desarrolladas para terminales móviles que accedan a Internet usando el protocolo **HTTP**. Para este caso el terminal utilizado ha sido el **Nokia 9110 Communicator**.

- **WML:** Estas otras páginas WEB se han desarrollado para aquellos terminales móviles que acceden a Internet usando el protocolo **WAP** (**Protocolo de aplicaciones inalámbricas**). En este caso se han de usar terminales móviles de última generación que vienen con un navegador especial que soporta este protocolo. El navegador que incorporan los terminales es distinto de unos a otros y depende de cada fabricante en concreto. Hasta el momento existen dos navegadores que son capaces de interpretar el protocolo WAP, uno propiedad de **Nokia** y otro propiedad de **Phone.com**. Se ha hecho uso de numerosos terminales móviles con estas características en el desarrollo del proyecto, los cuales se mencionan a continuación:

- ✓ **Alcatel One Touch**
- ✓ **Motorola Talkabout**
- ✓ **Siemens C-35**
- ✓ **Nokia 7110**
- ✓ **Ericsson T-28**
- ✓ **Motorola Timeport**

Además de los terminales móviles comentados, se han utilizado unos simuladores creados por las empresas suministradoras de los navegadores y que simulan el comportamiento de los terminales, para que los usuarios puedan desarrollar sus aplicaciones y observar como se verían las páginas**wml** al visualizarlas en las pantallas de dichos terminales, sin necesidad de establecer una conexión de datos a través de la red GSM. Estos simuladores también permiten simular los retrasos que introducen las distintas redes, según como vaya a ejecutarse la aplicación.

Se han usado dos simuladores, propiedad de las empresas desarrolladoras de los dos navegadores existentes hasta la fecha:

- **Nokia WAP ToolKit.**
- **UP.SDK (Unwired Planet es el navegador de Phone.com).**

Es importante destacar que en el caso de acceso a páginas **WEB** haciendo uso del protocolo **WAP**, es necesario que este servicio sea suministrado por el operador de telefonía móvil, para lo cual necesita disponer de un **Gateway WAP** que realice las conversiones de protocolo, como más adelante se ve con detenimiento. En este caso particular se ha utilizado **Movistar** como operador de telefonía móvil, haciendo uso del servicio **WAP** que dicha operadora puso en funcionamiento en Agosto de 2000.

5. SOLUCIONES PROPUESTAS.

Vamos a plantear en este capítulo soluciones al problema expuesto anteriormente que implementen un proceso eficiente de comunicación con los técnicos.

Se presenta a continuación una exposición breve de aquellos aspectos que se habían determinado como causas de una baja eficiencia del proceso actual de comunicación:

- **Tráfico elevado de mensajes cortos.**

  Vimos como el intercambio de información entre el sistema y los técnicos exigía un número elevado de mensajes cortos que obligaban a dotar a los terminales móviles de una capacidad de memoria suficiente para almacenarlos y de una capacidad de procesamiento (además de una aplicación que gestione estos mensajes) también adecuada.

- **Reprogramación de las tarjetas SIM de los terminales móviles.**

  Los cambios que se produzcan sobre la funcionalidad de la aplicación, materiales y accesorios nuevos que deban aparecer en los menús, etc..., obligan a recoger las tarjetas SIM de todos los móviles de los técnicos y a reprogramarlas.

- **Servicio NO FIABLE de mensajes cortos de texto.**

  El servicio de mensajería corta ofrecido por los operadores de telefonía móvil digital es un servicio no fiable que no garantiza la entrega de los mensajes a sus destinatarios.

Para evitar estos graves problemas que habían surgido se plantea como solución el acceso de los técnicos al Web de Técnicos a través de la red GSM. Para ello se van a utilizar determinados terminales móviles los cuales presentan una nueva funcionalidad que es la posibilidad de acceso a Internet.

De esta forma no va a ser necesaria la programación de ninguna aplicación en la memoria de los móviles sino que es suficiente con que estos dispongan de un navegador que les permita acceder al servidor para obtener la información.

Tampoco será necesario que estos terminales dispongan de memoria de almacenamiento ya que cualquier información que necesiten podrán obtenerla mediante una conexión con el servidor del sistema. Aunque no es necesaria si es recomendable disponer de cierta capacidad de almacenamiento, de forma que los técnicos puedan almacenar toda la información que necesiten para realizar su trabajo de forma "local" y no tengan que estar continuamente conectándose a la base de datos del sistema. Esto es así porque un número elevado de conexiones de datos con el servidor del sistema implican un aumento del gasto telefónico no compensado por la adquisición de terminales móviles más económicos con poca capacidad de almacenamiento (en este punto debe recordarse que nuestro objetivo principal es la optimización de los recursos para reducir al mínimo los costes). También suponen una carga innecesaria del sistema.

El problema surgido por la caracterización no fiable del servicio de mensajes cortos también quedaría subsanado por dos motivos fundamentales :

- El técnico recibiría una notificación de que le ha sido asignada una nueva actuación, tras lo cual debería establecer una conexión con el servidor del sistema para descargarse los datos relativos a su nueva ocupación. Esta conexión con el servidor serviría a este último para certificar que el técnico a recibido el último envio de trabajo.

- En la conexión establecida con el servidor por el técnico en la cual este introduce los datos necesarios en el sistema para la finalización de la actuación que esté llevando a cabo, se introducirán datos a mano y seleccionando de un menú como partes de un formulario. Cuando toda los campos requeridos hayan sido rellenados el sistema deberá comprobar la coherencia de los datos y no enviará una notificación de aceptación hasta que este último proceso no haya finalizado. El técnico no podrá dar por finalizada la actuación hasta que no haya recibido esta última notificación del sistema.

Con esto quedaría subsanado el problema de una posible inconsistencia del sistema debido a pérdidas de mensajes cortos.

Todo esto que se ha comentado, a su vez, facilita la actualización de la base de datos ya que cualquier cambio que se haga sobre esta estará disponible en tiempo real para cualquier técnico que acceda al Web. Ya no es necesaria una aplicación programada en la memoria del teléfono que cree los menus de selección y genere los mensajes de texto. Existirán accesos a la base de datos a través de Internet para obtener la información necesaria (posibles equipos instalados, materiales utilizados,...), información que se le suministrará a los técnicos mediante menús de selección, y los datos que son requeridos al técnico para el franqueo y cierre de la actuación serán enviados por este al servidor

mediante una selección en dichos menús y entrada manual de datos, todo ello a través de formularios.

El estudio de estos terminales móviles arroja una serie de conclusiones que recomiendan el diseño de páginas **WEB** optimizadas para ellos (cada uno de los teléfonos móviles presenta un navegador particular con características diferentes entre unos y otros). La creación de dichas páginas, junto con el uso de los lenguajes de programación **HTML**, **WML** y **Java**, implica el desorrollo en código **C++** de los **CGIs** que se ejecutarán en el servidor **WEB** y que controlarán el acceso a la base de datos **Oracle** para obtener la información necesaria o grabar los datos suministrados por los técnicos vía **WEB**. Todos estos accesos a la base de datos se realizan mediante funciones en lenguaje de programación **ProC** con **SQL** embebido. El usar **ProC** nos posibilita crear estas consultas **SQL** dentro de programas escritos en **C++**. Además de las consultas e inserciones a la base de datos obtendremos de los **CGIs** como resultado de la ejecución de los mismos las correspondientes páginas **WEB** a cada caso particular, páginas que contendrán la información necesaria para la creación de los formularios de selección o para la visualización de los datos solicitados por los técnicos. Estas páginas **WEB** creadas en el servidor son suministradas a los navegadores de los terminales móviles cuando estos las soliciten.

Existen limitaciones de ancho de banda de la red GSM que ralentizan la transmisión de cantidades importantes de datos. Llegados a este punto debe comentarse que el espectro radioeléctrico es un recurso limitado y compartido por numerosos servicios, lo que le deja a GSM una pequeña porción del mismo. Esto obliga a considerar a GSM como una red no optimizada para el tráfico de datos (no está pensada para ello), aunque un uso correcto de la misma nos ofrece suficientes recursos para dar una solución a nuestros problemas.

Otra característica que se debe tener en cuenta son las particularidades propia del navegador que presenta cada uno de los terminales móviles. La funcionalidad que ofrecen a sus usuarios es prácticamente la misma aunque su funcionamiento interno y capacidad de procesamiento varía mucho de unos a otros, lo que obliga al desarrollo de un Web de Técnicos (páginas WEB) optimizado para cada caso particular.

Se proponen dos soluciones, ambás factibles como veremos, cuya diferencia fundamental radica en el protocolo usado por los terminales móviles para el acceso a Internet:

- **Acceso a Internet usando el protocolo HTTP (Hypertext Transfer Protocol).**
- **Acceso a Internet usando el protocolo WAP (Wireless Aplication Protocol).**

5.1 Acceso a Internet usando el protocolo HTTP (Hypertext Transfer Protocol)

Para el acceso a Internet con un terminal móvil usando el protocolo HTTP usamos el Nokia 9110 Communicator, el cual presenta, entre otras, esta funcionalidad.

5.1.1 Nokia 9110 Communicator

Este terminal móvil presenta la posibilidad de acceso directo a Internet a través de la red GSM usando para ello el protocolo HTTP (es lo común en el mundo de Internet usando para la creación de páginas el lenguaje HTML). Este protocolo es el protocolo de comunicaciones estándar en Internet y, por tanto, no está optimizado para la transferencia de información usando la red GSM. Esta red presenta una limitación importante de ancho de banda lo que hace que el volumen de información que se debe transmitir sea un factor crítico. El uso del protocolo HTTP junto con el elevado precio de estos terminales móviles son los dos factores que van a condicionar nuestra decisión y nos hará decidirnos por la segunda solución que se posteriormente se propone.

El HTML básico para el Nokia 9110 Communicator debe basarse en HTML 3.2. Deben tenerse en cuenta algunas excepciones y limitaciones a la hora del diseño, entre las cuales destacamos las relativas a los comandos no soportados por el navegador del 9110 y las correspondientes a las limitaciones propias de la pantalla de visualización (tamaño de la misma). El navegador de este terminal no es estricto y simplemente desecha todos los comandos que no es capaz de interpretar, no intenta interpretarlos.

5.1.1.1 El navegador del terminal móvil Nokia 9110 Communicator

Este terminal móvil presenta un navegador (**browser**) propietario de **Nokia** que se denomina "**Nokia-Communicator-WWW-Browser/3.0 (Geos 3.0 Nokia-9110)**".

El navegador del Nokia 9110 no entiende Java ni Javascript, lo que nos obliga a cambiar todos aquellos scripts en Java que existen en el Web de Técnicos y sustituirlos por código que se ejecute en el servidor (no en el cliente). Mantendremos el código ya existente para el Web de Técnicos pero se introducirá una versión optimizada para el

9110 que se ejecutará cuando se acceda desde el mismo. En esta nueva versión no se ejecutarán **scripts** en el cliente como ya se ha comentado y cualquier nueva operación que haya de realizarse exigirá la carga de una nueva página desde el servidor.

La autentificación (insertar un nombre y una clave en un cuadro de diálogo) también es soportada por el navegador y puede usarse. Cuando son requeridos por segunda vez los datos de autentificación en el mismo dominio, una aplicación ofrece automaticamente el nombre de usuario y la clave obtenidos en la primera petición. Si esta falla el cuadro de diálogo se abre con el nombre de usuario por defecto. El proceso de autorización es transparente al usuario.

Deben evitarse formularios y diálogos que requieran mucha entrada de datos ya que puede hacerse engorroso para el usuario.

Para obtener mejores resultados es recomendable introducir enlaces a páginas diseñadas especificamente para el navegador del Nokia 9110 Communicator, no para otros navegadores. El mecanismo de acceso a una u otra página podrá basarse en la información de identificación del navegador que es requerida por el protocolo HTTP.

Las páginas deben optimizarse con el nivel apropiado de HTML, gráficos y número de enlaces para cada navegador. Cuando son diseñadas con propiedad, estas páginas son muy útiles y pueden usarse como plantillas para la creación de otras páginas de similares características. Cambios que se produzcan en el navegador pueden ser tenidos en cuenta con cambios en la plantilla mejor que con cambios en todas las páginas individuales creadas para el navegador.

Para asegurar que los usuarios reciben páginas que son compatibles con el Nokia 9110 debemos usar un **script** que chequee que navegador ó que usuario es el que solicita el documento (el servidor identifica el browser del 9110) y devolver las páginas optimizadas para el 9110.

5.1.1.2 Guía de diseño

A continuación vamos a presentar una breve guía de diseño cuyo objetivo es dar facilidades para la creación de páginas WEB optimizadas para el Nokia 9110. Estas consideraciones deberían tenerse en cuenta a la hora del diseño con lenguaje HTML de las páginas. Cuando se estime pertinente, se comentan las características que presenta el navegador del terminal para así poder ser consciente de sus limitaciones.

- ***HTML en general.***

  El HTML básico para el Nokia 9110 Communicator debe basarse en HTML 3.2. Además debe tenerse en cuenta que las limitaciones en el navegador nos impondrán restricciones a la hora de usar toda la potencia del lenguaje HTML. Entre ellas cabría destacar las relativas a los comandos no soportados por el navegador del 9110 y las correspondientes a las limitaciones propias de la pantalla de visualización (tamaño de la misma). Así mismo, no siempre será posible el posicionamiento de los objetos de la forma que se defina en el código fuente. El navegador del 9110 no es estricto y no intentará interpretar aquellos comandos HTML que no soporte.

  La pantalla y, por tanto, el diseño gráfico para el Nokia 9110 son radicalmente diferentes a los correspondientes realizados para una aplicación WEB a la que se acceda desde un PC. Debido a esto, para maximizar las propiedades de ambos casos, las versiones de las páginas WEB deberían crearse optimizadas para cada caso.

  Debería evitarse todo aquello que no sea imprescindible y que tienda a ralentizar la adquisición de los datos por parte del terminal móvil (comentarios HTML, excesivos espacios en blanco, comandos no utilizables, gráficos complejos,…).

- **Pantalla del Nokia 9110 Communicator.**

  El tamaño de la pantalla del Nokia 9110 Communicator es de 200×640 pixels. La representación en pantalla se realiza usando una escala de 16 tonos de grises. El area habilitada para representar el contenido de páginas Web es 200×540 pixels, ya que una parte de la pantalla se utiliza para las opciones del navegador y de la ventana. Esto plantea una limitación importante a la hora de presentar la información debido al reducido tamaño del display.

- **Texto.**

  El tamaño de texto estándar es de 12 puntos. El navegador soporta tanto texto con espaciado simple como texto con espaciado variado. También se permite texto subrayado, en negrita y la letra cursiva. Es posible utilizar además más de un tipo de letra distinto. El navegador también soporta todos los niveles de cabeceras.

  Ahora bien, existen algunas consideraciones que se deben tener en cuenta. El texto debería ser formateado para que todo aquello que se debiera leer primero aparezca en la primera pantalla de la página. Existe un scroll vertical pero para una navegación cómoda por la red conviene

hacer uso del mismo en la menor medida posible. No es posible decir lo mismo del scroll horizontal debido a que este no se nos presenta como una opción. No debemos hacer uso del subrayado como forma de enfatizar algunas puntos del texto ya que los enlaces a otras páginas representados por texto aparecen en pantalla como texto subrayado y puede ser motivo de confusión para el usuario. (Las palabras subrayadas aparecerían como enlaces para el usuario). El texto preformateado (comando <PRE>) no se debe usar ya que el usuario no puede redimensionar la ventana del navegador. Aproximadamente caben 75 caracteres en una sola linea cuando se utiliza el tamaño básico de texto (depende de las letras que aparezcan en el texto ya que no todas ocupan igual). Cuando visualizamos texto preformateado, aquellas líneas que son demasiado largas para las dimensiones de la pantalla continuan en la siguiente linea descolocando al resto que viene después, razón por la cual se debe evitar su uso en la medida de lo posible.

Uno de los cambios más importantes que se han acometido es el de modificar el boletín de actuación ya existente. Este boletin es creado por AF (sistema comercial de acceso a clientes) y en el Web de Técnicos que existe se muestra sin modificación por pantalla. Como se dijo anteriormente, el texto preformateado debía evitarse con el 9110 debido a que si las líneas no cabían completas en pantalla el navegador automaticamente baja a la siguiente línea lo que no hubiera cabido en la anterior. Esto da como resultado un boletín practicamente ilegible en la pantalla del Nokia 9110. Para evitar esto se decide hacer el acceso a la base de datos y leer aquellos campos que posteriormente se quieran mostrar por pantalla. Por tanto, para la versión optimizada para el 9110, sustituimos el boletin de actuación ya existente por la enumeración en pantalla de aquellos campos de dicho boletín que hayamos decidido mostrar.

- **Tablas.**

  El navegador del Nokia 9110 soporta tablas. Existe un botón que permite abrir un visualizador independiente para la correcta visualización de la tabla. Esta vista de la tabla es **scrollable** tanto horizontalmente como verticalmente. Contiene un cursor que se puede mover para habilitar la selección y copia de los contenidos de la tabla. Esta visión independiente de la tabla está disponible cuando existe una tabla en la página WEB que hayamos cargado. Si la tabla contiene elementos que son formularios, imágenes complejas ó campos de texto que exceden la longitud máxima permitida en la tabla entonces no podremos disponer de este visualizador independiente.

Cuando cargamos una página WEB, las tablas no se visualizan correctamente ya que el navegador las muestra en pantalla de forma que los elementos no están encolumnados (los fallos en la visualización de la tabla serán más o menos importantes dependiendo de varios factores, entre ellos de la longitud individual de cada elemento. Esta es la principal razón que nos mueve a diseñar nuestras páginas sin tablas, de tal forma que evitamos al usuario la tarea engorrosa de tener que visualizar la tabla independientemente de los datos cargados con la página WEB.

- **Frames.**

  El navegador del 9110 también soporta **frames** pero es desaconsejable su uso pues le crea problemas a los usuarios.

- **Scroll.**

  El usuario puede moverse a través de la pantalla así como seleccionar el enlace adecuado utilizando las teclas de **scroll** en el Nokia 9110 Communicator. Las teclas de **scroll** mueven al navegador a la próxima línea o al próximo enlace. La parte de la página que se visualiza también cambia en concordancia. La tecla "**Chr**" junto con las teclas de **scroll** pueden usarse conjuntamente para moverse una pantalla completa de una vez. También son interesantes las combinaciones de teclas "**Chr – H**" para ir al comienzo de la página así como "**Chr – E**" para ir al final de la página. No existe ningún cursor para un scroll continuado a través de la página. Como el movimiento libre a través de una página está restringido, los enlaces deben ser cuidadosamente situados. Páginas jerarquizadas y enlazadas permiten una navegación más sencilla para el usuario que una única página grande.

- **Jerarquía.**

  No es recomendable un elevado número de enlaces en una misma página a menos que se trate de un menú de elección. Tampoco es recomendable un elevado número de niveles jerarquicos y, si son necesarios, se debe garantizar una navegación fácil entre los niveles. Se debe dar al usuario una forma fácil de volver al subnivel anterior así como a la página principal. Las páginas complejas dificutan la navegación además de ralentizar el acceso a ellas por parte del navegador.

- **Gráficos.**

  Los atributos gráficos del Nokia 9110 Communicator deben ser tenidos en cuenta cuando se incorporan imágenes. Además del tamaño y de la

resolución de la pantalla también debemos tener en cuenta la velocidad de comunicación del navegador, la velocidad del protocolo HTTP y el uso de mapas de imágenes.

Es altamente recomendable el uso del atributo "**ALT**" para especificar una cadena de texto que el navegador pueda visualizar alternativamente a la imagen correspondiente. Presenta una opción mediante la cual, si se intenta visualizar una imagen de elevada complejidad, el navegador no la descarga y, en su defecto, presenta en pantalla un indicativo de que ahí debe ir una imagen. El usuario que quiera visualizar la imagen solo debe seleccionar este indicativo de imagen y la misma se le descargará individualmente en una pantalla nueva.

Se deben volver a diseñar aquellas imágenes que tengan más de ocho tonalidades de grises. También se prefieren grandes áreas rellenas de un mismo color a muchas pequeñas áreas de colores variados. Las imágenes se pueden cargar utilizando determinados tipos de compresión y ello es muy útil para mejorar la tasa de transferencia. Una imagen externa no debería exceder el tamaño 570 $\times$ 200 pixels, ya que esta ocuparía la pantalla completa. Si la imagen es descargada por el navegador de una página WEB, esta no debería exceder 540 $\times$ 200 pixels, ya que en este caso la pantalla queda reducida por las opciones del navegador. Si una imagen excede el tamaño máximo se escala hasta que tenga unas dimensiones adecuadas para ser representadas en pantalla (1:2, 1:3, etc…). Las dimensiones verticales de la imagen no son importantes ya que se dispone de un scroll vertical.

Los formatos para imágenes **GIF**, **JPG**, y **UPF** son soportados y podrán visualizarse aquellas imágenes con dicho formato. La compresión progresiva **JPEG** no se debería usar.

El tamaño de las imágenes no debería exceder 10 Kbytes. Como los usuarios pueden seleccionar la opción de no cargar automáticamente las imágenes deberían ser avisados en aquellos casos en que estemos ofreciendo páginas muy saturadas de imágenes.

Los GIFs animados no son soportados (solo se verá la primera imagen de la serie).

En cualquier caso, y siempre buscando el diseño optimizado de páginas WEB para el Nokia 9110 Communicator, es recomendable no usar imágenes y sustituir estas por otro tipo de elementos que permitan la creación de páginas fáciles de interpretar y que sean visualizadas con la mayor rapidez posible por el navegador.

- **Consideraciones sobre velocidad de transferencia de datos.**

  La velocidad de transferencia de datos del Nokia 9110 Communicator es de 9600 bit/s (ó de 14400 bit/s si es soportada por la red GSM). En condiciones normales no debería llevar más de 10 segundos el buscar y visualizar una página WEB típica optimizada para el navegador del 9110.

- **Enlaces.**

  Todos los enlaces se visualizan como texto subrayado. Debe evitarse el texto subrayado para enfatizar para no crear confusión al usuario. El comando <**EM**> puede producir un efecto de subrayado para el texto enfatizado. Los enlaces deben situarse de forma que sean claramente distinguibles por el usuario. Por ejemplo, dos palabras cortas situadas una junto a la otra podrían no aparecer como enlaces separados para el usuario. Aquellos enlaces que van semánticamente juntos deberían localizarse generalmente de forma que el usuario no tuviera que hacer un scroll vertical a la siguiente pantalla para poder acceder al siguiente enlace. Los enlaces que juntos forman un conjunto de elecciones, de las cuales el usuario debe seleccionar una, deberían ser visibles al mismo tiempo en la pantalla. La probabilidad de que una de las opciones no sea tenida en cuenta se disminuye en gran medida si se agrupan convenientemente los enlaces.

  El usuario puede mantener una lista de "bookmarks" personales en el Nokia 9110.

- **Memoria Caché.**

  El navegador del Nokia 9110 presenta un sistema de **caché**. La **memoria caché** almacena los documentos que son buscados y las páginas que son obtenidas por el navegador. Esta caché tiene un tamaño máximo de **300 Kb**. Cuando la caché está llena y se accede a nuevos documentos se borran los documentos más antiguos para dejar sitio a estos. Son almacenados en la caché todos los documentos buscados excepto los documentos seguros y los documentos a los cuales se accede y tienen la cabecera HTTP "**Cache-Control : no-cache**" ó "**Pragma : no-cache**". La visualización de la página en la pantalla a veces puede variar dependiendo de si el documento es obtenido o no de la memoria caché. Esto debe ser tenido en cuenta a la hora de preparar el aspecto de la página ya que, aunque en el código se especifique que se quiere visualizar una distribución determinada por pantalla, realmente y dependiendo de donde haya cargado el navegador la página se visualizará o no lo que se había previsto. En concreto, cuando el

navegador del Nokia 9110 Communicator carga la página del servidor (sin usar la caché) parece insertar un "**EOL**" antes de una lista de selección con lo cual desplaza a esta a la siguiente línea. Esto, sin embargo, no ocurre cuando el navegador carga la página de la memoria caché, caso en el cual el aspecto de la página en pantalla es el mismo que habíamos definido en el código.

- **Otras consideraciones**

  - ✓ La autentificación (insertar un nombre y una clave en un cuadro de diálogo) también es soportada por el navegador y puede usarse. Cuando los datos de autentificación son requeridos por segunda vez en el mismo dominio, una aplicación ofrece automaticamente el nombre de usuario y la clave obtenidos en la primera petición. Si esta falla el cuadro de diálogo se abre con el nombre de usuario por defecto. El proceso de autorización es transparente al usuario.

  - ✓ Deben evitarse formularios y diálogos que requieran mucha entrada de datos ya que puede hacerse engorroso para el usuario.

  - ✓ Para obtener mejores resultados es recomendable introducir enlaces a páginas diseñadas especificamente para el navegador del Nokia 9110 Communicator. El mecanismo de acceso a una u otra página podrá basarse en la información de identificación del navegador que es requerida por el protocolo HTTP.

5.1.2 Modificaciones sobre el software existente

El objetivo de este apartado es proporcionar una referencia adecuada sobre la estructura de las páginas WEB creadas para el acceso al Web de técnicos usando el Nokia 9110 Communicator. Se pretende reflejar los cambios realizados sobre el software ya existente para adaptarlo a las nuevas exigencias. De igual forma se indican las limitaciones más importantes detectadas en el uso del navegador incorporado al 9110 y las soluciones adoptadas para solventar las dificultades que se fueron presentando.

Una vez determinado el aspecto que queremos que tengan las páginas del Web de Técnicos optimizadas para el Nokia 9110 Communicator tenemos que proceder a modificar el software ya existente para poder acceder a estas páginas desde el navegador del 9110. A continuación se enumeran los cambios que se han considerado necesarios para este fín y se mostrarán las soluciones adoptadas para así facilitar una posterior integración de este software en versiones posteriores del Web de Técnicos.

5.1.2.1 Modificaciones generales

En las modificaciones realizadas sobre los códigos que controlan la creación de las páginas que se han de visualizar en pantalla (**cgi**$_s$) se ha procurado seguir siempre el mismo esquema de cambios para así facilitar la posterior integración del código en versiones posteriores. En la medida de lo posible se ha intentado no modificar las librerias y trabajar con las ya existentes. En alguna ocasión ha habido que añadir alguna nueva función a estas librerias para realizar determinadas actividades exclusivas para el **Web del 9110**, que basicamente realizan las mismas operaciones pero con datos distintos (por ejemplo, los accesos a la base de datos para obtener los campos que se quieran mostrar por pantalla). Se ha introducido en todos los **cgi**$_s$ la definición de las constantes y variables que han sido necesarias pero todo ello siguiendo una estructura definida.

Hemos mantenido también el código que ya existía sin modificar y, para ello, en todos los casos, se ha duplicado el código para mantener el antiguo y realizar los cambios oportunos sobre aquello que va a ser la versión optimizada para el 9110. Así mismo, y en todos los **cgi**$_s$, se ha añadido una función al final que será la encargada de mostrar los mensajes de error correspondientes cuando es llamada con uno u otro valor de sus parámetros. Todo el tratamiento de errores se ha hecho de la misma forma y por ello en la sección de definiciones de cada uno de los **cgi**$_s$ deberán definirse los tipos de errores que puedan aparecer y el código correspondiente asignado que se le pasará como parámetro a la función.

En todos los códigos, comenzamos con un pequeño control de errores para determinar si el usuario conectado está o no autorizado para entrar en el **Web de Técnicos** y, a continuación, añadimos código para decidir que parte del **cgi** se va a ejecutar y que dependerá del tipo de usuario que se haya conectado (si en el nombre de usuario remoto conectado aparece la cadena de dígitos "**9110**" se ejecutará la parte de código de nueva creación que ha sido diseñada de forma óptimizada para el navegador del Nokia 9110). Esta parte de código de nueva creación aparece como una función independiente que en cada **cgi** lleva el nombre **main9110**.

Para ver como se hace esto, a continuación se muestran las líneas de código que van al comienzo del programa principal y que determinan que solo se ejecute la parte correspondiente al 9110 en caso de que deba ser así :

```
// Codigo 9110
 remoteuser = getenv ("REMOTE_USER");
 if (remoteuser == NULL)
  return error9110 (ERR_VAR_REMOTEUSER);
  serverurl = getenv ("SERVER_URL");
  if (serverurl == NULL)
      return error9110 (ERR_VAR_SERVERURL);
  // Primero se lee el fichero de configuracion
  if (leewebconf())
      return error9110 (ERR_CONFIGURACION);
  // Si no es un usuario 9110 camino normal
  if (strstr (remoteuser, "9110"))
      return main9110 ();
  // Fin de codigo 9110
```

También debemos de tener en cuenta en cada **cgi** el incluir los ficheros de cabeceras que sean necesarios y que no hayan sido incluidos con anterioridad. Tanto la función principal cuando se trata del 9110 "**main9110**", como la función para errores "**error9110**", aparecen como funciones normales dentro de cada **cgi** y, como tales, deben tener definido su prototipo antes del cuerpo principal del programa.

Hemos introducido las modificaciones generales que se han hecho sobre todos los **cgi**$_s$ y dado una idea sobre cual ha sido el esquema seguido. A continuación veremos cada **cgi** independientemente y los cambios realizados sobre él. Así mismo y en su caso se darán a conocer las posibles modificaciones que haya sido necesario realizar sobre las librerías.

5.1.2.2 inicio9110.cgi

Este primer **cgi** es completamente nuevo. Se encarga de visualizar la primera página del Web de Técnicos cuando se accede ya sea como usuario normal ó como usuario particular del Nokia 9110 (tiene su tratamiento de errores para el caso de que se trate de un usuario no autorizado).

En primer lugar se comprueba si en el nombre de usuario aparece la cadena 9110. Si no es así se sigue el camino normal que existía hasta ahora y se visualiza en pantalla la primera página del Web que ya había creado. Si aparece dicha cadena, entonces tendrá que cargarse el Web optimizado para el 9110 y este se ha diseñado completamente configurable, con lo cual la primera página que se visualize va a depender de la configuración que hayamos elegido. Dicha configuración dependerá de lo especificado en el fichero de configuración "**webconf.txt**".

En el código aparece lo siguiente :

" Pragma : no-cache\n "

Esto se pone para que el navegador siempre obtenga la página del servidor y no de la memoria caché del terminal móvil. Con ello conseguimos que los posibles cambios en la configuración siempre sean tenidos en cuenta.

Una vez comprobada la autenticidad y validez del usuario se visualizará una página u otra dependiendo de la configuración existente. En principio, y siguiendo las especificaciones dadas para el Web de Técnicos vía terminal móvil Nokia 9110, la configuración existente hará que se visualize en pantalla el detalle de la actuación en curso para ese técnico o, en caso de no tener ninguna en curso, un mensaje de advertencia sobre dicha situación. Como ya se ha dicho, esto es completamente reconfigurable y, entre otras cosas, se contempla la posibilidad de cargar como primera página una lista de todas las actuaciones planificadas para ese técnico (o de un número de ellas si esta lista ha sido limitada).

Tras el código principal se contempla una función de tratamiento de los errores para sacar el mensaje de advertencia correspondiente al error que se haya producido y que se indica a dicha función a través de un parámetro.

Finalmente observamos la existencia de una función de nueva creación para visualizar en pantalla una página que nos de un mensaje de advertencia en el caso de que el

usuario no tenga ninguna actuación asignada. En este caso se le da la opción al usuario de consultar de nuevo por si ha habido cambios o de ver la lista de actuaciones planificadas (todo esto a través de enlaces).

5.1.2.3 detalle.cgi

La finalidad de este **cgi** es acceder a la base de datos para enviar al terminal móvil del técnico una página WEB donde se encuentren reflejados los detalles de la actuación asignada al técnico.

Como siempre, al principio del código de cada **cgi**, se comprueba la autenticidad del usuario y si tiene permiso o no de acceso al WEB. Si está autorizado se mira si se trata de un usuario que accede a través del terminal móvil Nokia 9110, en cuyo caso se ejecuta la parte de código orientada a la creación de una página optimizada para el comunicador.

Antes, cuando se visualizaba el detalle de una actuación, lo que se hacía era sacar en pantalla el boletín de la actuación suministrado por **AF** (sistema del Area Comercial de aceso a cliente). Sin ninguna modificación, usando para ello el comando de HTML **<PRE>**. En el caso del 9110 ya vimos los problemas que esto planteaba debido fundamentalmente a la limitación de la pantalla del comunicador y a la inexistencia de un **scroll** horizontal. Para corregir esto se han creado dos nuevas funciones especificas para el código del 9110 y que van a realizar el acceso a la base de datos (utilizando para ello código **proC**) obteniendo de ella aquellos campos que se ha decidido mostrar por pantalla como el nuevo boletín de actuación. Una vez obtenidos esos campos se visualizan en pantalla en un orden y con una leyenda que hemos predeterminado. Estas funciones de nueva creación se han añadido a la librería de funciones existente para SQL denominada "**sql.pc**" y sus correspondientes prototipos al fichero de cabeceras "**auxsql.def**". A continuación vemos las dos funciones creadas a tal efecto (y la definición de los cursores necesarios para la selección). Como vemos existirá una para el caso de acceso a base de datos para obtención de campos del boletín de una avería y otra para la obtención de los campos correspondientes al boletín de una orden de servicio (instalación):

AVERIA

EXEC SQL DECLARE det9110aver CURSOR FOR

SELECT esav.equipos_servicios

FROM ga_datosequiposserviciosaav esav, ga_averias av

```
 WHERE esav.id_datos_equip = av.id_datos_equip AND

        av.entry_id = :actuacion;


int detalle9110av(char *act, char *doc, char *est, char *telef, char *grupo, char *com,char *tit, char *domic, char *poblac, char *provinc, char *tip_client, char *cent,char *sint, long *c_p, long *fh_recl, char *tip_av,char *a_plant, char *grul, char *p_a_r, char *denom, char *p_ent, char *p_sal, char *c_term, char *res_prueb, char *ob_mec, char *sit_c_term, char *sit_c_conex, char *asig_de, char ***eqser,  int *neqser, char *marca, int flag_contrata)

{

int tam=15,i=0;

char **eq;

conecta();

CHECKSQL

CTOSQL(actuacion,act);

/* Solo se debe llamar  if(*act=='A') &&  if !(flag_contrata) */

/* Primero se obtienen los detalles de la averia GA_Averias */

EXEC SQL

SELECT

a.documento, a.estado, p.telefono_n_os, p.tecnico_grupo, NVL(a.comentario, ' '),NVL(a.titular,' '), NVL(a.domicilio,' '), NVL(a.poblacion,''), NVL(a.provincia,' '),NVL(a.tipo_cliente,' '), NVL(a.central,' '), NVL(a.sintoma,' '), NVL(a.cita_previa,0),NVL(a.fechahora_reclamacion,0), NVL (a.tipo_de_aviso,''), NVL(a.area_de_planta,' '),NVL(a.grupo, ' '), NVL(a.par,' '), NVL(a.denominacion,''), NVL(a.par_entrada,' '),NVL(a.par_de_salida,' '), NVL(a.caja_terminal,' '), NVL(a.resultado_prueba,' '),NVL(a.observaciones_mecanico,' '), NVL(a.situacion_caja_terminal,' '),NVL(a.situacion_caja_conexion,' '), NVL(a.asignacion_de,' ')

INTO
```

```
:documento, :estado, :telefono, :gtecnico, :comentario, :titular,
:domicilio, :poblacion, :provincia, :tipo_cliente, :cen_tral, :sintoma,
:cit_prev, :fh_reclamacion, :tipo_de_aviso, :area_de_planta, :grupol,
:par, :denominacion, :par_entrada, :par_salida, :caja_terminal,
:result_prueba, :observ_mecan, :situac_caja_term,
:situac_caja_conex, :asignacion_de

FROM ga_averias a, ga_planificacion p

WHERE a.entry_id = p.entry_id_actuacion AND (estado =est)

SQLTOC(telefono,telef);

SQLTOC(gtecnico,grupo);

SQLTOC(comentario,com); SQLTOC(marca_modif,marca);

/* Variables 9110 */

SQLTOC(titular,tit);

SQLTOC(domicilio,domic);

SQLTOC(poblacion,poblac);

SQLTOC(provincia,provinc);

SQLTOC(tipo_cliente,tip_client);

SQLTOC(cen_tral,cent);

SQLTOC(sintoma,sint);

*c_p=cit_prev;

*fh_recl=fh_reclamacion;

SQLTOC(tipo_de_aviso,tip_av);

SQLTOC(area_de_planta,a_plant);

SQLTOC(grupol,grul);

SQLTOC(par,p_a_r);

SQLTOC(denominacion,denom);

SQLTOC(par_entrada,p_ent);
```

```
SQLTOC(par_salida,p_sal);
SQLTOC(caja_terminal,c_term);
SQLTOC(result_prueba,res_prueb);
SQLTOC(observ_mecan,ob_mec);
SQLTOC(situac_caja_term,sit_c_term);
SQLTOC(situac_caja_conex,sit_c_conex);
SQLTOC(asignacion_de,asig_de);
/* Ahora detalles de equipos asociados a la averia*/
eq=malloc(sizeof(char *)*tam);
EXEC SQL OPEN det9110aver;
CHECKSQL
EXEC SQL FETCH det9110aver INTO :eqservicio;
while(sqlca.sqlcode==0) {
eqservicio.arr[eqservicio.len]='\0';
eq[i++]=strdup((char*)eqservicio.arr);
if(i==tam){
tam+=10;
eq=realloc(eq,tam*sizeof(char*));
}
EXEC SQL FETCH det9110aver INTO :eqservicio;
}
EXEC SQL CLOSE det9110aver;
if(sqlca.sqlcode<0) return sqlca.sqlcode;
*eqser=eq;
*neqser=i;
```

/ Solo se asigna un valor a "doc" cuando no ha habido ningun problema, ya*

*que en "detalle.C" se comprueban los errores mirando esta variable */*

SQLTOC(documento,doc)

return 0;

}

ORDENES DE SERVICIO

EXEC SQL DECLARE det9110os1 CURSOR FOR

SELECT NVL(es.operacion,' '), NVL(es.equipos_servicios,' '),

NVL(es.unidades,' '),NVL(es.prolongacion,' ')

FROM ga_datosequiposserviciosaos es, ga_ordendeservicio os

WHERE es.id_datos_equip = os.id_datos_equip AND os.entry_id = :actuacion;

EXEC SQL DECLARE det9110os2 CURSOR FOR

SELECT

NVL(li.operacion2,' '), NVL(li.linea,' '), NVL(li.central,' '),

NVL(li.telefono_linea,' '), NVL(li.asign,' '), NVL(li.grupo,' '),

NVL(li.par,' '), NVL(li.caja,' '), NVL(li.domicilio,' ')

FROM ga_datoslineasaos li, ga_ordendeservicio os

WHERE li.id_datos_lineas = os.id_actuacion AND

os.entry_id = :actuacion;

*int detalle9110os(char *act, char *doc, char *est, char *n_os, char *grupo, char *com, char *tit, char *domic, char *poblac, char *provinc, char *efect, char *cent,char *cl_os, long *fh_creac, long*

```
*fh_form, char *ent, char *n_solic, char *clas_abon, char *tipo_serv,
char *procedenc, char *tipo_trab, char *clav_esp, char *observac,
struct datosEquiposOS **eqos, int *neqos, struct datosLineasOS
**lin_os, int *nlineas, char *marca, int flag_contrata)

{

int tam=15,i=0;

struct datosEquiposOS *eq;

struct datosLineasOS *lin;

conecta();

CHECKSQL

CTOSQL(actuacion,act);

/* Solo se debe llamar  if(*act=='O') &&  if !(flag_contrata) */

/* Primero se obtienen los detalles de la orden de servicio
GA_OrdenDeServicio */

EXEC SQL

SELECT

os.documento, os.estado, os.n_orden_de_servicio,
NVL(p.tecnico_grupo,' '), NVL(os.comentario, ' '), NVL(os.titular,' '),
NVL(os.domicilio,' '), NVL(os.poblacion,' '), NVL(os.provincia,' '),
NVL(os.efectividad,' '), NVL(os.central,' '), NVL(os.clase_os,' ')
NVL(os.fecha_hora_de_creacion,0),

NVL(os.fecha_hora_de_formulacion,0), NVL(os.entidad,' '),
NVL(os.n_de_solicitud,' '),

NVL(os.clase_de_abono,' '), NVL(os.tipo_de_servicio,' '),
NVL(os.procedencia,' '),

NVL(os.tipo_de_trabajo,' '), NVL(os.claves_especiales,' '),
NVL(os.observaciones,' ')

INTO

:documento, :estado, :n_orden_serv, :gtecnico, :comentario,
:titular, :domicilio, :poblacion, :provincia, :efectividad, :cen_tral,
:clase_os, :fechahora_creac, :fechahora_formul, :entidad,
```

```
:n_solicitud, :clase_abono, :tipo_servicio, :procedencia,
:tipo_trabajo, :claves_espec, :observaciones

FROM ga_ordendeservicio os, ga_planificacion p

WHERE os.entry_id = p.entry_id_actuacion AND

os.entry_id = :actuacion;

CHECKSQL

SQLTOC(estado,est)

SQLTOC(n_orden_serv,n_os)

SQLTOC(gtecnico,grupo);

SQLTOC(comentario,com); /*!!*/

SQLTOC(marca_modif,marca);

/* Variables 9110 */

SQLTOC(titular,tit);

SQLTOC(domicilio,domic);

SQLTOC(poblacion,poblac);

SQLTOC(provincia,provinc);

SQLTOC(efectividad,efect);

SQLTOC(cen_tral,cent);

SQLTOC(clase_os,cl_os);

*fh_creac=fechahora_creac;

*fh_form=fechahora_formul;

SQLTOC(entidad,ent);

SQLTOC(n_solicitud,n_solic);

SQLTOC(clase_abono,clas_abon);

SQLTOC(tipo_servicio,tipo_serv);
```

```
SQLTOC(procedencia,procedenc);

SQLTOC(tipo_trabajo,tipo_trab);

SQLTOC(claves_espec,clav_esp);

SQLTOC(observaciones,observac);

/* Ahora detalles de equipos asociados a la averia*/

eq=malloc(sizeof(struct datosEquiposOS)*tam);

EXEC SQL OPEN det9110os1;

CHECKSQL

EXEC SQL FETCH det9110os1 INTO :operacion, :eqservicio,
:unidades, :prolongacion;

while(sqlca.sqlcode==0) {

operacion.arr[operacion.len]='\0';

eqservicio.arr[eqservicio.len]='\0';

unidades.arr[unidades.len]='\0';

prolongacion.arr[prolongacion.len]='\0';

strcpy (eq[i].operac, (char *) operacion.arr);

strcpy (eq[i].equipo, (char *) eqservicio.arr);

strcpy (eq[i].unidades, (char *) unidades.arr);

strcpy (eq[i].prolong, (char *) prolongacion.arr);

i++;

if(i==tam){

tam+=10;

eq=realloc(eq,tam*sizeof(struct datosEquiposOS));

}

EXEC SQL FETCH det9110os1 INTO :operacion, :eqservicio,
:unidades, :prolongacion;
```

```
}

EXEC SQL CLOSE det9110os1;

if(sqlca.sqlcode<0) return sqlca.sqlcode;

*eqos = eq;

*neqos = i;

/* Ahora detalles de líneas asociadas a la averia */

tam=15; i=0;

lin=malloc(sizeof(struct datosLineasOS)*tam);

EXEC SQL OPEN det9110os2;

CHECKSQL

EXEC SQL FETCH det9110os2 INTO :operacion2, :linea, :li_central,
:telefono_linea, :asign, :li_grupo, :li_par, :caja, :li_domicilio;

while(sqlca.sqlcode==0) {

operacion2.arr[operacion2.len]='\0';

linea.arr[linea.len]='\0';

li_central.arr[li_central.len]='\0';

telefono_linea.arr[telefono_linea.len]='\0';

asign.arr[asign.len]='\0';

li_grupo.arr[li_grupo.len]='\0';

li_par.arr[li_par.len]='\0';

caja.arr[caja.len]='\0';

li_domicilio.arr[li_domicilio.len]='\0';

strcpy (lin[i].operac2, (char *) operacion2.arr);

strcpy (lin[i].linea, (char *) linea.arr);

strcpy (lin[i].central, (char *) li_central.arr);
```

```
strcpy (lin[i].telefono_linea, (char *) telefono_linea.arr);

strcpy (lin[i].asign, (char *) asign.arr);

strcpy (lin[i].grupo, (char *) li_grupo.arr);

strcpy (lin[i].par, (char *) li_par.arr);

strcpy (lin[i].caja, (char *) caja.arr);

strcpy (lin[i].domicilio, (char *) li_domicilio.arr);

i++;

if(i==tam){

tam+=10;

lin=realloc(lin,tam*sizeof(struct datosLineasOS));

}

EXEC SQL FETCH det9110os2 INTO :operacion2, :linea, :li_central
:telefono_linea, :asign, :li_grupo, :li_par, :caja, :li_domicilio;

}

EXEC SQL CLOSE det9110os2;

if(sqlca.sqlcode<0) return sqlca.sqlcode;

*eqos = eq;

*neqos = i;

*lin_os = lin;

*nlineas = i;

/* Solo se asigna un valor a "doc" cuando no ha habido ningun
problema, ya que en "detalle.C" se comprueban los errores mirando
esta variable */

SQLTOC(documento,doc);

return 0;

}
```

Al principio del código representado, tras la definición del primer cursor para consultar datos de equipos en la tabla de equipos posibles para averías, aparece definida la función "**detalle9110av**", cuya cabecera se reproduce a continuación:

*int detalle9110av(char *act, char *doc, char *est, char *telef, char *grupo, char *com, char *tit, char *domic, char *poblac, char *provinc, char *tip_client, char *cent,char *sint, long *c_p, long *fh_recl, char *tip_av,char *a_plant, char *grul, char *p_a_r, char *denom, char *p_ent, char *p_sal, char *c_term, char *res_prueb, char *ob_mec, char *sit_c_term, char *sit_c_conex, char *asig_de, char ***eqser, int *neqser, char *marca, int flag_contrata)*

Observamos en dicha cabecera la presencia de un puntero triple, cuya existencia vamos a justificar a continuación.

La clave de todo esto está en que cuando hacemos las consultas de equipos a la base de datos para, posteriormente, crear los menús de selección que le aparecerán en pantalla al técnico, no sabemos de antemano la longitud de las cadenas de texto que nos va a devolver la base de datos (estos equipos forman parte del boletín de actuación que se entrega al técnico, para que este se haga una idea de lo que puede necesitar para le realización de su trabajo). Antes de la realización de este proyecto, en el **Web** se mostraba el boletín tal cual está almacenado en los sistemas corporativos, practicamente como un fichero de texto, pero para los terminales móviles no es posible mostrar este texto "**preformateado**" ya que excede de las dimensiones del **display** del que estos disponen y, por tanto, los equipos relacionados con cada actuación hay que extraerlos, uno por uno, de la base de datos).

Pues bién, si queremos obtener el nombre de un equipo del cual desconozco su longitud hago lo siguiente:

- **declaro puntero a carácter (char *eq1).**
- **le paso como parámetro a la función la dirección de memoria de este puntero (&eq1), donde escribiremos la dirección de memoria de la cadena de texto cuando la memoria necesaria haya sido asignada dinámicamente.**
- **la función obtiene este parámetro (dirección) como un doble puntero (char **eq2).**

- **una vez leido el equipo (como char * eqleido) copiamos la dirección del equipo leido en la dirección de memoria correcta (*eq2=eqleido).**
- **por la correspondencia de parámetros habremos escrito esta dirección en la direción de memoria de nuestro puntero original (&eq1.).**

En el caso que nos ocupa, no leemos un equipo de la base de datos, sino una lista de equipos de la cual no sabemos cuanto va a ocuparnos y cuya memoria se asignará dinámicamente dentro de la función. Por lo tanto, sigo el mismo procedimiento pero partiendo de un doble puntero.

- **declaro puntero (lista) a puntero (equipo) a carácter (char**eqser).**
- **paso la dirección del mismo como parámetro a la función (&eqser).**
- **la función obtiene este parámetro (dirección) como un triple puntero (char ***eqser).**
- **una vez leida la lista de equipos (como char ** eq) copiamos la dirección de la lista de equipos leidos en la dirección de memoria correcta (*eqser=eq).**
- **igual que antes, por la correspondencia de parámetros, habremos escrito esta dirección en la dirección de memoria de nuestro puntero original, en este caso un puntero a lista (&eq1).**

Una vez obtenidos los campos, la visualización del boletín de actuación dependerá de si es una avería o una orden de servicio (instalación). Cuando obtenemos estos campos de la base de datos, alguno de ellos presenta espacios en blanco innecesarios, tanto al final como intermedios, que debemos eliminar para que dichos campos quepan en pantalla (recordar que no se dispone de **scroll horizontal**). Para los espacios en blanco al final del campo ya teníamos una función que los eliminaba, "**RecortarBlancos**". Para los espacios en blanco intermedios hemos creado una y la hemos llamado "**RecortarBlancosIntermedios**". El código de esta función es el siguiente :

```
// Función para recortar blancos intermedios usada en main9110
void RecortarBlancosIntermedios (char *cadena)
 {
  int aux1,aux2;
  for (int i=0; i < strlen(cadena);i++)
        if (cadena[i]==' ' && cadena[i+1]==' ')
         {
          aux1=i+1;
          aux2=i+2;
          while (aux1 < strlen(cadena))
            {
              cadena[aux1]=cadena[aux2];
               aux1++;
              aux2++;
            };
          i--;
         }
}
```

En la página de presentación del boletín de actuación aparecen varios enlaces. Hay dos que aparecen siempre, "**Inicio**" que es un enlace al "**inicio9110.cgi**" para mostrar por pantalla el detalle de la actuación en curso, y "**Lista**" para visualizar la lista de las actuaciones planificadas para el técnico. Por último, "**Asignar**". Este último enlace no va a aparecer siempre. Para que aparezca el detalle que estemos viendo no puede ser el de la actuación en curso ya que esta está ya asignada, sino el de una de las actuaciones pendientes. Ahora bien, aunque sea una actuación pendiente el usuario podrá asignársela o no, dependiendo del valor de una de las entradas del fichero de configuración "**webconf.txt**". Tal y como se configuraba el Web antiguo existía una sola entrada de dicho fichero que permitía la asignación o no a ese técnico. Dicha entrada era

"**webconf.asigna**" y si estaba a "**1**" permitía la asignación y si estaba a '**0**" no la permitía. Por tanto, dependiendo del valor de dicha entrada aparecía o no el enlace correspondiente al cgi "**asignar.cgi**". Para el caso del Nokia 9110 se ha hecho una pequeña modificación. Esta entrada del fichero de configuración se ha desdoblado en dos, denominadas "**webconf.asignaMAN**" y "**webconf.asignaAUTO**" que permitirá la configuración independiente de la asignación automática y de la asignación manual por parte del usuario. En ambos casos, si el valor de la entrada correspondiente del fichero de configuración está a "**1**" indicará que se encuentra habilitada dicha asignación y deshabilitada en caso de que esté a "**0**". Por tanto, en el caso del enlace "**Asignar**" en el detalle de una actuación que está pendiente, dicho enlace aparecerá en caso de que la entrada del fichero de configuración "**webconf.asignaMAN**" se encuentre a "**1**" y no aparecerá en caso de que dicha entrada esté a "**0**". Este cambio en la entrada del fichero de configuración también hemos tenido que reflejarlo en otros sitios. Se ha cambiado en la parte correspondiente al franqueo y a la cumplimentación (instalación), como veremos posteriormente (después de franquear ó cumplimentar puede que se configure para asignación automática de una nueva actuación). También se han hecho cambios en el fichero **webconfig.c** para definir la nueva estructura usada para leer esta nueva configuración. Así mismo, dentro de este mismo fichero, se ha modificado la rutina que lee la configuración. Estos dos cambios quedan como sigue :

```
struct {
  char loginDB[20], passwdDB[20], servDB[20],
  loginAR[20], passwdAR[20], servAR[20];
  int asignaMAN, asignaAUTO, tecN, prN, min_fr;
  } webConf = {"","","","","","",0,0,0,0};
int leewebconf()
{
  char buf[100];
  FILE *f=fopen(FWEBCONF,"r");
  if(f==NULL) return -100;
  while(fgets(buf,100,f)!=NULL){
  if(!strncmp("ASIGNAAUTO",buf,10))
  webConf.asignaAUTO=atoi(strchr(buf,'=')+1);
```

```
else if(!strncmp("ASIGNAMAN",buf,9))
   webConf.asignaMAN=  atoi(strchr(buf,'=')+1);
else if(!strncmp("NTEC",buf,4))
    webConf.tecN=atoi(strchr(buf,'=')+1);
else if(!strncmp("NPR",buf,3))
    webConf.prN=atoi(strchr(buf,'=')+1);
else if(!strncmp("LOGDB",buf,5)){
   strcpy(webConf.loginDB,strchr(buf,'=')+1);
   webConf.loginDB[strlen(webConf.loginDB) - 1]='\0';
    }
else if(!strncmp("LOGAR",buf,5)){
    strcpy(webConf.loginAR,strchr(buf,'=')+1);
    webConf.loginAR[strlen(webConf.loginAR) -1] ='\0';
    }
else if(!strncmp("PASSWDDB",buf,8)){
strcpy(webConf.passwdDB,strchr(buf,'=')+1);
webConf.passwdDB[strlen(webConf.passwdDB)-1]='\0';
    }
else                                  if(!strncmp("PASSWDAR",buf,8)){
strcpy(webConf.passwdAR,strchr(buf,'=')+1);
webConf.passwdAR[strlen(webConf.passwdAR)-1]='\0';
    }
else if(!strncmp("SERVDB",buf,6)){
    strcpy(webConf.servDB,strchr(buf,'=')+1);
    webConf.servDB[strlen(webConf.servDB)-1]='\0';
```

```
    }
else if(!strncmp("SERVAR",buf,6)){
    strcpy(webConf.servAR,strchr(buf,'=')+1);
    webConf.servAR[strlen(webConf.servAR)-1]='\0';
    }
 else if(!strncmp("TFRANQUEO",buf,9))
      webConf.min_fr=atoi(strchr(buf,'=')+1);
fclose(f);
return 0;
  }
```

Este cambio también se tiene que reflejar en "**webconf.h**".

En el detalle de la actuación en curso aparecerá un enlace para franquear ó cumplimentar dependiendo del tipo de actuación. Si es franqueo el enlace llevará al primer **cgi** de los pasos de franqueo y si es cumplimentación al primer **cgi** para la cumplimentación.

5.1.2.4 tecact.cgi

Este **cgi** es el encargado de listar un breve resumen de todas las actuaciones planificadas para un técnico.

En este **cgi**, al igual que en el resto, lo primero que se hace en el cuerpo principal del programa es identificar al usuario conectado al **WEB** y, en función del tipo que sea, se ejecutará la parte del código que ya existía para un usuario normal ó la parte nueva de código optimizada para el 9110 si se trata de un usuario que accede al WEB a través de este tipo de terminal móvil (una vez se ha hecho un pequeño control de errores y determinado que el usuario está autorizado a acceder al **WEB**).

Una vez dentro del código de nueva creación, lo primero que observamos en este **cgi** es una línea de código para que esta página no se obtenga nunca de la memoria caché del terminal móvil y siempre sea cargada de nuevo desde el servidor. Esto se hace para que

la página que se muestre por pantalla registre siempre los últimos cambios que se hayan efectuado sobre la planificación de las actuaciones para el técnico (franqueo de actuaciones, nueva asignación, etc…) y no sea necesario hacer uso de una opción especifica del terminal móvil para cargar de nuevo la página (es posible que no nos demos cuenta de posibles cambios si el navegador obtiene los datos de la página de la memoria caché). La línea de código que realiza esta función es la siguiente :

```
// Para evitar el uso de cache en los cgi
cout << "Pragma: no-cache" << endl;
```

Lo siguiente es una comprobación de que el usuario tiene perfil de técnico que serán los únicos que tendrán acceso al Web vía terminal móvil Nokia 9110. Se continúa con la creación de una cabecera específica para el 9110. El código que, a partir del número de matrícula del técnico, nos va a determinar cuales son las actuaciones que tiene pendientes ese técnico permanece prácticamente invariable. Lo que si ha cambiado es la forma de representar la información obtenida en pantalla y que ahora, para optimizarlo para el navegador del 9110, aparece sin tablas ni imágenes. Lo que se ha hecho es sacar la información general de cada actuación como una pequeña lista y así una detrás de otra.

En el nuevo **Web para el 9110** se accede directamente al detalle de la actuación en curso. En la página que muestra el detalle de la actuación se ha añadido un enlace a este **cgi** si el usuario desea ver la lista de actuaciones pendientes.

Si queremos optimizar el diseño para el Nokia 9110, se han sustituido los enlaces, que eran **gifs** y no se podían visualizar bien en la pantalla del Nokia 9110 ya que este no tiene tanta capacidad de procesamiento, por palabras subrayadas que siempre seguirán el mismo patrón (forma de distribuir los enlaces en pantalla). Un enlace particular que aparece en esta página está representado por "**Reintentar**" y solo aparecerá, junto a un mensaje aclaratorio, en caso de que el técnico no tenga actuaciones pendientes. Cuando se seleccione este enlace se volverá a cargar la página desde el servidor y así poder ver si se ha producido alguna modificación (se le ha planificado alguna actuación al técnico).

Al final y, siguiendo el mismo estilo de diseño para el **WEB del 9110**, tendremos la función para crear la página de error con el mensaje específico en función del tipo de error que se haya dado.

5.1.2.5 asigna.cgi

Este **cgi** es el que se encarga de la asignación, cuando corresponda, de una actuación a un técnico. Se ha modificado al principio, como en todos, para comprobar la validez del usuario, su autorización y si se conecta o no vía terminal móvil. La función modificada para el 9110 sigue haciendo lo mismo que antes y lo único que cambia es el formato de los mensajes que se muestran por pantalla (cuando se ha llegado hasta aquí es porque ya se ha comprobado la posibilidad de asignación manual). Siguiendo nuestro criterio se introduce, finalmente, una función de tratamiento de errores que distingue los diferentes casos en función de un parámetro que se le pasa.

5.1.2.6 listadet.cgi

La finalidad de este **cgi** es presentar por pantalla, uno a continuación de otro, todos los boletines de actuación correspondientes a actuaciones planificadas para un técnico en concreto. Se visualizan en orden por fecha de planificación siendo la que está en curso la primera en aparecer. Basicamente consiste en hacer lo mismo que se hacía en el "**detalle.cgi**", pero ahora se repite en un bucle que termina cuando ya no existen más actuaciones planificadas cuyo boletín no se haya representado ó se llegue al límite definido en el fichero de configuración. Por tanto, tras las comprobaciones de autenticidad que se hacen en todos los **cgi$_s$** y la identificación del usuario como conectado a través del terminal móvil Nokia 9110, se pasa a sacar en pantalla la lista de los boletines de actuación correspondientes a ese técnico, operación para la cual se hace uso de las funciones creadas para acceso a la base de datos y que habíamos llamado "**detalle9110av**" y "**detalle9110os**" (según se trate de una avería o una orden de servicio).

Al final, como en todos hasta ahora, podemos observar la función creada para el tratamiento de errores.

5.1.2.7 franqueo1.cgi

Se trata del primero de una colección de cinco **cgi$_s$** que se van a encargar del franqueo y del consiguiente cierre de la actuación. Su misión consiste en crear las páginas **WEB** que muestren a los técnicos los menús de selección y en acceder a la base de datos para

introducir en esta los datos que les son suministrados por los técnicos mediante formularios de inserción.

Se comienza, al igual que siempre con las comprobaciones y determinación de si el usuario accede a través de un terminal móvil Nokia 9110 para, si es así, ejecutar la parte de código creada a tal efecto. Como esto se hace para todos los **cgi**$_s$ de franqueo, en lo sucesivo no se mencionará más.

En estos **cgi**$_s$ de franqueo la novedad principal es que la operación se realiza en cinco pasos. Esto tiene que ser así debido a la incapacidad del navegador del Nokia 9110 de comprender código **Javascript**. Esto hace que todos aquellos **scripts** de **Java** que ya existían y que se corrían en el cliente deban ser sustituidos por alguna otra solución equivalente. La solución adoptada ha sido ir rellenando los campos del formulario uno por uno y, en cada paso se vuelve a cargar una nueva página del servidor donde ya aparecerán reflejados y de forma fija los campos del formulario rellenados en pasos anteriores. Esto es así hasta que se han completado todos los datos de franqueo, en cuyo caso se realizará el franqueo de la actuación.

Este **cgi** de franqueo muestra por pantalla un formulario sencillo con los primeros datos a rellenar por el usuario. Además aparecen enlaces para cancelar la operación o para continuar con ella. Este último enlace hará que se ejecute el segundo **cgi** de franqueo y se deberán introducir los siguientes datos de franqueo. El botón de "**submit**" del formulario será el que enlace con el siguiente **cgi** y se produzca el traspaso de parámetros.

Los datos que se van introduciendo en los sucesivos pasos de franqueo se recogen siempre en el formulario, ya sean como campos de entrada de datos o como campos ocultos. Por tanto, en la navegación entre los diferentes **cgi**$_s$, estos campos se deben ir pasando de unos a otros como parámetros. En cada **cgi** se leen todos estos parámetros en una clase creada para tal efecto. El problema que surge es que el constructor de la clase destroza la variable de entorno. Para evitar este problema lo que se debe hacer es una copia de dicha variable de entorno para construir la clase. Lo vemos en el código siguiente, el cual es un fragmento del código desarrollado para nuestra aplicación:

```
// strdup porque el constructor de parametros destroza la variable de
//entorno

Parámetros par(strdup(getenv("QUERY_STRING")));
```

Existe algún caso en el cual, en la creación de la clase, se destroza la variable de entorno. Posteriormente, si se produce un error, se llama a la función de tratamiento de errores la cual crea un enlace al **cgi** "**detalle.cgi**" para continuar desde ese punto una vez

se ha dado el mensaje de error. Pues bien, para la creación de ese enlace no se pueden obtener los parámetros necesarios de la variable de entorno ya que ha sido destrozada. Para evitar este problema se creará un nuevo parámetro que puede o no pasarse a la función (por defecto estará a **NULL**). Dentro de la función de error se comprobará el valor de este parámetro de tal forma que si está a **NULL** es porque no se le ha pasado como parámetro y eso indicará que la variable de entorno no se ha destrozado aún y que es posible leer los parámetros de forma normal. Por el contrario, si a este parámetro se le ha dado algún valor, se utilizará el mismo para la creación del enlace.

Al final del código, como siempre, aparece la función creada para el tratamiento de errores.

5.1.2.8 franqueo2.cgi

No nos vamos a detener mucho en los siguientes pasos de franqueo ya que son análogos al primero, con la diferencia de que a medida que avanzamos en el franqueo aparecen nuevos campos del formulario a rellenar por el usuario y los datos ya introducidos aparecen como fijados.

5.1.2.9 franqueo3.cgi

Este cgi es prácticamente análogo al anterior y la razón de su existencia es el ir avanzando en los diferentes pasos de franqueo.

5.1.2.10 franqueo4.cgi

Como siempre se empieza por un pequeño control de usuario y determinación de si se trata de un usuario del Nokia 9110. Si es así se ejecuta el código creado a tal efecto. Se fijan los datos introducidos en pasos anteriores y se visualizan los campos que quedan por rellenar para completar el franqueo. El botón de **submit** "**Continuar**" de los anteriores pasos se sustituye por "**Franquear**" para advertir al usuario de que si todos los datos son correctos y se selecciona este pulsa este botón el franqueo se completará y la actuación pasará al estado cerrada.

5.1.2.11 franquear.cgi

Este **cgi** es el único diferente en el proceso de franqueo. No carga una página con el correspondiente formulario para que el usuario tenga que introducir datos. Su función es la de comprobar que se han seguido correctamente todos los pasos de franqueo y dar las instrucciones oportunas para que el franqueo se complete correctamente. Si algo falla cargará una página con un mensaje de error de acuerdo con el error cometido.

Tras los controles de usuario de rigor se determinará, en su caso, la ejecución del código para el caso del 9110. Si la avería se ha franqueado correctamente se da el oportuno mensaje y, si la entrada del fichero de configuración "**asignaAUTO**" está puesta a "**1**" se asignará automáticamente al técnico la siguiente actuación planificada. En caso contrario se franqueará la que está en curso y pasará a estar cerrada sin ninguna asignación automática para el técnico. Este cambio en el fichero de configuración ya ha sido comentado anteriormente.

Al final del código, siguiendo con nuestro estilo, aparece la función creada para el tratamiento de los posibles errores que puedan producirse.

5.1.2.12 cumplimform.cgi

Se trata del primer **cgi** en el proceso de cumplimentación de instalaciones. En este caso, a diferencia del caso de franqueo, existirán menos pasos (pudiendo ser sólo uno). Este **cgi** comienza con el correspondiente control de usuario. Cuando se determina que el usuario accede a través de un terminal móvil 9110, ejecuta el código correspondiente (página cargada optimizada para visualizarla con el Nokia 9110 Communicator).

Las funciones que realizan los **cgi**$_s$ de cumplimentación de instalaciones son idénticas a las realizadas por los de franqueo de averías que hemos visto hasta ahora, con la única diferencia de tipos de datos diferentes usados en unos y otros.

Las tres primeras listas de selección han cambiado respecto a lo que ya había. Antes se ejecutaba un pequeño **script** de **Java** de tal forma que cuando se seleccionaba una opción de una de las listas de selección automaticamente se seleccionaban las opciones correspondientes de las otras dos listas (están relacionadas las tres listas). Como ya sabemos el **Javascript** es algo no soportado por este navegador y, por lo tanto, lo comentado anteriormente no podrá llevarse a cabo en nuestro caso. Aparecerán las tres listas de selección pero habrá que elegir la opción correcta en los tres casos, sin que una de ellas ya determine automáticamente las otras dos.

La cumplimentación con incidencia también presenta una pequeña modificación. Cuando se cumplimentaba con incidencia debía seleccionarse la causa de la misma. Podía dejarse en blanco o seleccionar solo una de las opciones, nunca ambas simultáneamente. Esto se conseguía con **Javascript**, por lo cual se ha tenido que hacer de otra forma. La opción elegida es la de situar tres botones que son "**mutuamente excluyentes**", es decir, que solo podrá estar seleccionado uno de ellos. El botón que aparecerá seleccionado por defecto será el de "**ninguna**".

También aparece dentro del formulario un botón para seleccionar si se quiere o no cerrar la actuación. Si no se cierra, lo que se generará será un **informe de actividad** sobre la orden de servicio en curso.

Finalmente aparece la posibilidad de cumplimentar con modificación. Será un **cgi** distinto el que deba ejecutarse en este caso, luego será necesario una pequeña comprobación al comienzo del siguiente **cgi** (**cumplim.cgi**) para determinar si debe o no ejecutarse el otro **cgi** (**cumplimMod.cgi**). Esto es así porque solo es posible incluir un botón de **submit** en el formulario y aquí tenemos dos posibles **cgi**$_s$ a ejecutarse. Anteriormente eran **scripts** de **Java** los que se ejecutaban en el cliente cuando se pulsaban determinados botones, y estos **scripts** eran los que cambiaban la acción del **submit** del formulario. Esto ya sabemos que no puede hacerse con el navegador del 9110, por lo cual la opción de que el **submit** del formulario llame siempre al mismo **cgi** y este compruebe al principio si es o no necesario que se ejecute el otro **cgi**, parece una buena alternativa. En el caso de cumplimentar con modificación lo seleccionamos con un botón e introduciremos en su campo correspondiente el patrón de búsqueda en la base de datos. Este patrón de búsqueda también ha sido modificado en su definición para que sea un poco mayor. La razón de hacer esto es una particularidad del navegador del Nokia 9110 que hace que listas de selección largas descoloquen la posición de los elementos en la pantalla. Para evitar esto que da una mala visualización de la página se debe hacer un filtrado más exhaustivo de los equipos para que las listas presenten menor número de elementos. Para conseguir eso hemos aumentado el tamaño del campo patrón y, además, hemos cambiado la forma de hacer el SELECT en la función correspondiente del fichero "**sql.pc**". Lo que hemos hecho ha sido modificar el SELECT para que busque la cadena patrón en cualquier posición dentro del nombre del equipo mientras que antes solo buscaba los equipos que comenzaran por ese patrón. Ahora es posible incluso buscar aquellos equipos que tengan en su nombre más de una cadena, siendo estas cadenas las especificadas entre signos "**%**" en el patrón. Por tanto se debe tener en cuenta esta modificación sobre una función del fichero "**sql.pc**".

En estos **cgi**$_s$ de cumplimentación también seguimos nuestro patrón de tratamiento de errores y para ello tenemos al final del código la función correspondiente.

5.1.2.13 cumplim.cgi

Este **cgi** es el encargado de introducir los datos en la base de datos y de devolver un mensaje de aceptación si todo ha transcurrido con normalidad y los datos son coherentes. Este mensaje de aceptación dependerá de si se ha seleccionado la opción de cerrar la actuación, o solo se ha generado un informe de actividad. En este último caso no se le asigna una nueva actuación al técnico debido a que todavía no se ha cerrado la actual.

Este **cgi** comienza con la comprobación de autenticidad del técnico y con la determinación de si se trata de uno conectado a través del terminal móvil 9110.

Una vez comienza a ejecutarse el código correspondiente al 9110 debemos leer los parámetros y ver si se trata o no de una cumplimentación con modificación. Si así fuese deberíamos llamar al **cgi** que cumplimenta con modificación pero ideando alguna forma de pasarle todos estos parámetros necesarios para la cumplimentación (un **cgi** llama a otro y debe pasarle a este último todos aquellos parámetros obtenidos por el primero a través del formulario). La forma elegida de hacerlo ha sido con la lectura de los parámetros del fichero estándar de entrada y su posterior escritura, antes de la llamada al **cgi**, en el fichero estándar de salida. Lo podemos ver a continuación :

```
nread2 = 0;
while ((nread1 = read (0, cadena1, 1000)) != 0)
  {
        fprintf (in, "Leidos %d\n", nread1);
        memcpy (cadena2+nread2, cadena1, nread1);
        nread2 += nread1;
   };
 fprintf (in, "LeidosFin %d\n", nread2);
 pipe (&fdPipe[0]);
close (0);
dup (fdPipe[0]);
```

```
nwrite =write (fdPipe[1], cadena2, nread2);

fprintf (in, "Escritos %d\n", nwrite);

close (fdPipe[1]);

if (strstr (cadena2, "MOD=s")) // Si flag de modificacion activado

execl ("cumplimMod.cgi", "cumplimMod.cgi",   NULL);
```

Una vez hecha la comprobación este **cgi** debe determinar si los datos de la cumplimentación son correctos o no. Si lo fueran deberá generar el informe de actividad correspondiente si no está seleccionada la opción de cerrar ó cumplimentar definitivamente la orden de servicio si dicha opción está seleccionada. Si se cierra la actuación, dependiendo de los valores del fichero de configuración, podrá o no asignársele una nueva actuación al técnico.

Cuando este **cgi** genera los mensajes correspondientes (de error si se ha producido o de conformidad si la operación solicitada se ha realizado correctamente) aparecerá un enlace a diferentes **cgi**$_s$ dependiendo de lo que hubiese sido solicitado. En el caso de un error o de la generación de un informe de actividad, el enlace "**Continuar**" nos llevará al detalla de la orden de servicio que está en curso. En el caso de cumplimentación con cierre el enlace continuar nos llevará al inicio **(inicio9110.cgi**) para determinar si el técnico tiene o no una nueva actuación asignada.

Al final del código tenemos la función para el tratamiento de errores.

5.1.2.14 cumplimMod.cgi

Este es el **cgi** que se ejecutará si el usuario selecciona una cumplimentación con modificación. Para ello en el paso anterior ha debido de seleccionar esta opción e introducir el patrón correspondiente. Aquí, con este patrón, se hará un acceso a la base de datos para obtener los equipos que contienen dicho patrón. Una vez obtenidos se mostrarán como listas de selección de un formulario cuyo botón de **submit** hará que se ejecute el **cgi** "**cumplim.cgi**".

En este formulario también aparecerá el botón de cerrar, el cual estará seleccionado o no por defecto dependiendo de si lo hemos o no seleccionado en el paso anterior. Si ahora cambiásemos de opinión no tendríamos más que cambiar el valor de este botón hasta dejarlo como quisiéramos.

Cuando se llama al "**cumplim.cgi**" se le ha cambiado el valor al campo de cumplimentación con modificación de forma que dicho **cgi** pueda generar el informe de actividad correspondiente o cumplimentar la orden de servicio.

5.1.2.15 defines.h

Es importante también tener en cuenta que se han introducido numerosas definiciones nuevas, con lo cual el fichero "**defines.h**" también a sido modificado (este fichero es incluido siempre al comienzo de todos los demás y contiene la definición de todas las constantes que se usan posteriormente en el código fuente del programa).

5.1.2.16 Resumen de modificaciones realizadas

A continuación mostramos se muestra un breve resumen de todos los ficheros que han sido modificados para la generación de páginas optimizadas para el Nokia9110 Communicator, así como las modificaciones principales llevadas a cabo sobre cada uno de ellos. Comenzamos con una enumeración de las modificaciones generales que aparecen en todos los ficheros:

- Duplicación del código para mantener el antiguo y realizar cambios sobre el nuevo.
- Comprobación al comienzo de cada **cgi** de la autenticidad y autorización del usuario.
- Comprobación en todos los **cgi**$_s$ de si el usuario conectado lo hace o no a través del terminal móvil Nokia9110 para ejecutar la parte de código correspondiente.
- Introducción al final del código de cada **cgi** de una función para el tratamiento de los errores.
- Definición de tipos de errores y asignación del código correspondiente para pasárselo como parámetro a la función de tratamiento de error.

Estas serían las modificaciones generales para todos los **cgi$_s$**. A continuación, y a modo de resumen, se muestra la **Tabla 9**, donde aparecen aquellos ficheros que se han modificado para la integración del código que permita la creación de páginas optimizadas para el 9110 así como las modificaciones más importantes.

Tabla 9.- Modificaciones para conseguir páginas WEB optimizadas para el terminal móvil Nokia 9110 Communicator.

Ficheros	Modificaciones
Inicio9110.C	Fichero de nueva creación. Decide cual es la primera página que se carga cuando se accede al Web.
Detalle.C	Cambia el acceso a base de datos para obtener los campos del boletín. Función nueva para recortar espacios en blanco sobrantes en medio del campo. Cambia la forma de hacer la asignación manual y automática.
sql.pc	Aparecen funciones de nueva creación para el acceso a los campos del boletín: "detalle9110av" y "detalle9110os". Modificación sobre el patrón de búsqueda (para hacer el SELECT) en la cumplimentación con modificación.
auxsql.h	Deben aparecer los prototipos de las nuevas funciones creadas.
webconfig.C	Cambios en la estructura creada para leer el fichero de configuración debido al cambio en el proceso de asignación.
webconf.h	Los cambios del punto anterior deben definirse también aquí.

listadet.C	Nuevas funciones para acceso a los campos del boletín. Nueva función para recortar blancos intermedios.
tecact.C	Cambio de la tabla por listas para representar la lista de actuaciones planificadas para el técnico. Código para evitar el uso de la memoria caché por parte del browser del 9110.
asigna.C	Las modificaciones que aparecen aquí se limitan a la forma de los mensajes de confirmación y de error.
franqueo1.C	Forma del formulario de franqueo. Eliminación del código Javascript y su sustitución por un franqueo en diversos pasos (repetidas cargas de páginas desde el servidor).
franqueo2.C	Forma del formulario de franqueo. Eliminación del código Javascript y su sustitución por un franqueo en diversos pasos (repetidas cargas de páginas desde el servidor).
franqueo3.C	Forma del formulario de franqueo. Eliminación del código Javascript y su sustitución por un franqueo en diversos pasos (repetidas cargas de páginas desde el servidor).
franqueo4.C	Forma del formulario de franqueo. Eliminación del código Javascript y su sustitución por un franqueo en diversos pasos (repetidas cargas de páginas desde el servidor).
franquear.C	Cambia el mecanismo de asignación automática tras el franqueo de una avería. Los mensajes de error y de confirmación del franqueo cambian su formato.

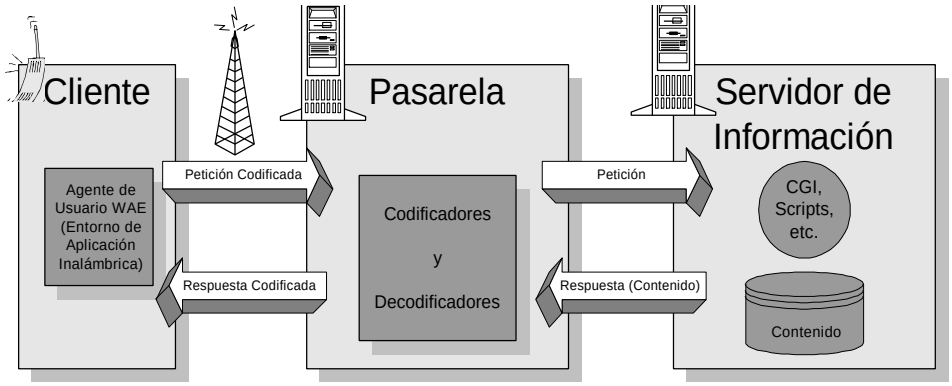
cumplimform.C	Forma del formulario de cumplimentación. Eliminación del código Javascript y su sustitución por una cumplimentación en diversos pasos Cambia la cumplimentación con incidencia. Cambia la cumplimentación con modificación. Eliminación de la cumplimentación de material ("cumplimMat.cgi" se queda como estaba, no se crea versión nueva para el 9110).
cumplim.C	Lee parámetros, si la cumplimentación es con modificación ejecuta el cgi "cumplimMod.cgi", y si no termina su propia ejecución. Los mensajes de error y los de confirmación cambian su formato.
cumplimMod.C	Selecciona los equipos de la base de datos según patrón de modificación. Llama a "cumplim.cgi" para finalizar la cumplimentación.
defines.h	Fichero modificado por la aparición de numerosas definiciones nuevas.

Es importante comentar que, a pesar de todas las modificaciones introducidas para conseguir conexiones **on-line** con la base de datos del sistema, nos es imposible eliminar la interfaz creada con el **Centro de Mensajes Cortos** de **Movistar**. Esto es debido a que es necesario que el técnico reciba al menos un mensaje de alerta que le indique que tiene que conectarse al servidor del sistema ya que se le ha asignado una nueva actuación. Si esto no se hiciese, el sistema no tendría ninguna posibilidad de comunicarle este evento al técnico, a menos que este decidiera conectarse al sistema por su iniciativa propia.

5.2 Acceso a Internet usando el protocolo WAP (Wireless Aplication Protocol)

Se trata de una nueva arquitectura que emplea un nuevo protocolo para las conexiones de datos a través de la red **GSM** (realmente es un protocolo desarrollado para cualquier tipo de red móvil). La arquitectura utilizada para el acceso a servidores WEB en Internet es la mostrada en la **Figura 22**.

Figura 22.- Arquitectura WAP para accesos a servidores WEB en Internet.



Es la segunda solución propuesta y, como se deduce de las distintas pruebas realizacas, la más eficiente.

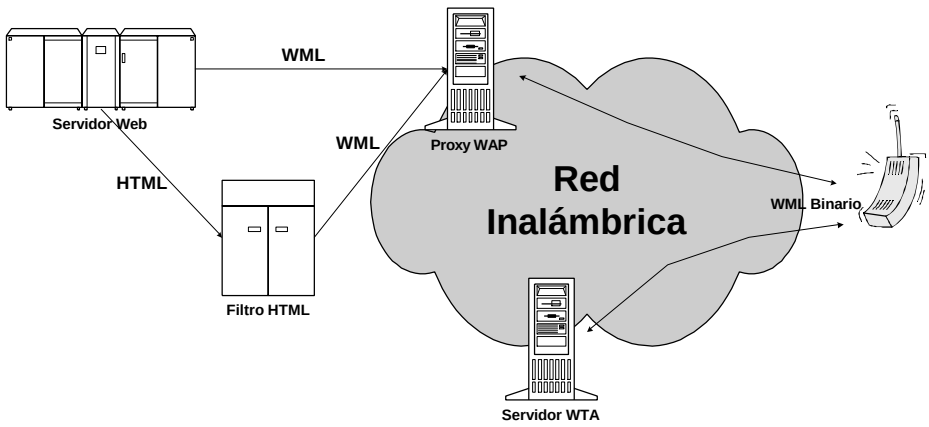
Este otro grupo de teléfonos móviles se engloba dentro de un grupo de terminales móviles de última generación que también acceden a la información de Internet pero lo hacen usando un protocolo diferente, más novedoso, el protocolo **WAP**. El Protocolo de Aplicaciones Inalámbricas surge como la combinación de dos tecnologías de amplio crecimiento y difusión durante los últimos años: las **Comunicaciones Inalámbricas** e **Internet**. Se parte de una arquitectura basada en la arquitectura definida para el **World Wide Web** (WWW), pero adaptada a los nuevos requisitos del sistema.

Para hacer esto el terminal **WAP** necesita páginas escritas en lenguaje **WML**. Hay tres formas de conseguir esto :

- Desarrollando un **Web** particular con páginas escritas en código WML (la conversión de protocolos y la identificación de usuarios dados de alta en el servicio de Wap dado por el operador de telefonía móvil correspondiente la hace el **Gateway**).
- Introduciendo un **Gateway** que haga la conversión de HTML a WML (además de la necesaria conversión de protocolos e identificación de usuarios dados de alta).
- Usando servidores que sean capaces de entender las peticiones de los navegadores que usan el protocolo WAP.

Las distintas formas de acceso a la información es posible apreciarlas en la**Figura 23.**

Figura 23.- Distintas formas de acceso a Internet desde un terminal móvil.



5.2.1 Características técnicas

El protocolo **WAP** si es un protocolo optimizado para comunicación de datos a través de la red **GSM**. Utiliza una compresión binaria de datos que reduce en gran medida el volumen de información que debe transferirse entre los terminales móviles y el servidor lo que permite que se adapte perfectamente a la limitación de ancho de banda de la red **GSM**. Sin embargo el uso de este protocolo obliga a la aparición de un **proxy WAP** ya que el protocolo estándar en **Internet** es el protocolo **HTTP** y debe de existir algo que realice el cambio de protocolo de forma que la petición realizada desde el móvil pueda ser entendida por el servidor. Algo análogo ocurre en el proceso inverso en el cual la respuesta dada por el servidor lleva una cabecera de protocolo **HTTP** y debe ser traducida a una cabecera de protocolo **WAP**. La prestación de servicios **WAP** por parte de los operadores de telefonía móvil obliga a estos a habilitar dicho proxy. El navegador que presentan los terminales **WAP** solo es capaz de representar la páginas **Web** codificadas en lenguaje **WML**, lo cual obliga a diseñar un **WEB** optimizado para nuestra aplicación con páginas codificadas en **WML**. Este lenguaje también ha sido optimizado para un óptimo aprovechamiento del ancho de banda de la red **GSM** y presenta características particulares. Existe la posibilidad de un **proxy WAP** que relice, además de la conversión de protocolo, una conversión de código de **HTML** a **WML** y viceversa lo que puede evitar el tener que diseñar obligatoriamente unas páginas **Web** especificas en **WML**. Esto no se utiliza en la solución que presentamos a nuestro problema por tres razones fundamentales:

- El código HTML creado para páginas WEB que se han de visualizar en un PC o en la pantalla del terminal móvil Nokia 9110 Communicator no está optimizado para aplicaciones WAP ya que existen muchas diferencias entre los terminales (distinto tamaño de las pantallas, soportan distintos tipos de gráficos y distinto tamaño de los mismos, diferencia entre los comandos entendidos por los distintos navegadores…). Esto obligaría a volver a diseñar las páginas Web llegando a una solución de compromiso de forma que el código HTML fuese entendido por todos los navegadores y se pudiera crear la página Web en todos los casos. Pero ahora esta página Web no estaría optimizada para ninguno de los casos y es posible que no sea una solución válida para la empresa que desarrolla la aplicación. Además, puesto que habría que volver a diseñar las páginas, el único factor que podría impedir el desarrollo de la aplicación usando WML sería la capacidad de memoria del servidor de aplicaciones ya que se debería

disponer de varias versiones diferentes (HTML, WML,…) y ese no suele ser un factor crítico.

- Aún en el caso de que se decidiese crear páginas optimizadas para el servicio WAP en HTML, usando un proxy WAP que realizara la conversión de código de HTML a WML, tampoco sería posible que dichas páginas fuesen óptimas debido a que los proxys que realizan las conversiones de código están poco desarrollados y solo pueden realizar conversiones básicas. De esta forma se perdería gran parte de la potencia que ofrece el lenguaje de programación WML.

- En la aplicación que se está desarrollando es un factor crítico el tiempo de conexión a la red ya que se producirá un elevado número de conexiones a la red para acceder a la base de datos (el personal técnico

- es muy numeroso) durante las cuales se está gastando dinero hasta la finalización de las mismas. Esto hace que se prefiera el diseño de las páginas directamente en WML de forma que no se emplee tiempo innecesario en la conversión de código (el proceso de franqueo y cierre de una actuación requiere interactuar con el servidor y cargar un elevado número de páginas, por lo que el tiempo empleado en la conversión de código sería elevado).

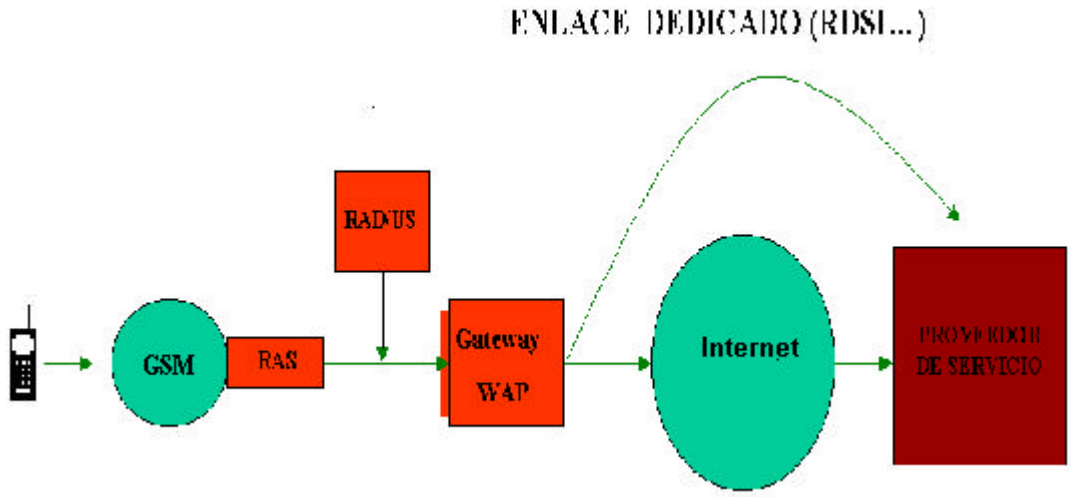
El aprovechamiento óptimo del reducido ancho de banda debido a la transferencia minimizada de datos (lo que da lugar a una mayor velocidad en la transferencia de información) es el motivo principal para la elección de esta segunda solución como la más indicada.

Como un punto negativo debemos señalar que los terminales WAP presentan una pantalla de visualización reducida, por lo cual no disponen de toda la capacidad de representación de páginas WEB de que dispone el terminal móvil Nokia 9110 Communicator. Esto hace que el 9110 pueda ofrecer páginas WEB mucho más elaboradas. Sin embargo el elevado coste de este terminal frente al coste de los terminales WAP, mucho más reducido, hace que esta capacidad de visualización quede en un segundo plano para nuestra aplicación (existe un elevado numero de técnicos que implicaría una inversión muy elevada de dinero en la compra de un terminal móvil Nokia 9110 Communicator para cada técnico).

5.2.2 Arquitectura hardware

En este apartado vamos a mostrar la arquitectura hardware necesaria para el correcto funcionamiento de nuestra aplicación. En la **Figura 24** se muestra dicha arquitectura.

Figura 24.- Arquitectura hardware.



En la **Figura 24** puede verse como el terminal móvil establece una llamada de datos a través de la red **GSM** y esta es encaminada hasta el **proxy WAP**. Una vez allí, se produce la conversión de protocolo y las peticiones **HTTP** son enviadas, a través de la red **Internet**, al servidor **WEB** correspondiente. SE observa que una alternativa a esto es utilizar un enlace dedicado entre el **Gateway WAP** y los servidores de nuestro sistema para, de esa forma, evitar todos los retrasos que introduce la saturación de **Internet**, retrasos que pueden llegar a ser muy elevados.

Aunque se ha explicado para una aplicación particular que se ha desarrollado con objeto de este proyecto y para un sistema particular, la arquitectura aquí representada es la arquitectura general para todo tipo de aplicaciones de datos que usen redes inalámbricas (redes móviles).

5.2.3 Modificaciones realizadas sobre el Web de Técnicos

En este caso la modificación es completa, ya que se crea un Web de Técnicos completamente nuevo con páginas WEB escritas en lenguaje WML. Esto es así porque debemos configurar a nuestro servidor WEB para que sepa que las peticiones que le lleguen por un puerto específico se refieren a páginas escritas en WML. Seguirán siendo peticiones HTTP, pero el servidor sabrá que lo que le solicitan son páginas WML. De esta forma el responderá también con cabecera HTTP pero enviará páginas HTML con una subcabecera WAP para la posterior conversión de protocolo en el proxy WAP.

Partiendo de esto ya podemos crear una página de inicio estática en WML para las peticiones al servidor por ese puerto.

Como primera modificación, y muy importante, destacamos que al estar usando distintos protocolos vamos a necesitar distintas variables de entorno para crear nuestra aplicación WEB, ya que los navegadores son muy diferentes a los que usan el protocolo HTTP. Cada navegador introduce en sus cabeceras distinto tipo de información en las peticiones que realizan a los servidores WEB, información que se encuentra localizada en las llamadas **variables de entorno,** variables que van a ser, en general, distintas para cada protocolo (algunas coinciden y otras no).

Todas las modificaciones adicionales que deben hacerse en las páginas WEB del sistema son prácticamente las mismas que las realizadas en el caso del Nokia 9110, pero teniendo en cuenta la diferencia fundamental de que mientras que en el caso anterior las páginas eran escritas en HTML, en este el lenguaje que se usará es el WML.

Vamos a exponer a continuación las modificaciones más importantes realizadas para el diseño de este nuevo WEB.

5.2.3.1 start.wml

Se trata de la página **WML** de inicio. Es la página que el servidor **WEB** enviará al navegador del terminal móvil **WAP** cuando este realice una petición al servidor por el puerto adecuado (en nuestro caso será el puerto 8080). La dirección que debemos introducir en el navegador del teléfono móvil es:

http:// 195.235.93.137:8080

Al recibir esta petición, el servidor devuelve la página WML "**start.wml"** (en el software de administración del servidor existen unos parámetros de configuración para indicar todo ello).

A continuación presentamos la cabecera WAP que va al inicio de esta página WML y que deberá ir en el resto de páginas WML que el servidor devuelve como resultado de ejecutar cada **cgi**:

```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"

"http://www.wapforum.org/DTD/wml_1.1.xml">
```

Esta cabecerea indica que se trata de un documento WML 1.1, así como la dirección donde es posible encontrar el archivo de **Tipo de Documento**.

Tenemos un problema con la memoria caché de los terminales móviles. Este consiste en que las páginas WML cargadas por dichos terminales son cacheadas en memoria durante un determinado periodo de tiempo. Esto implica que cuando realicemos una petición a nuestro servidor WEB, primero se comprobará si la página solicitada se encuentra cacheada en memoria y, si es así, no se efectuará una nueva descarga desde el servidor, se visualizará en pantalla la que ya teníamos en caché. Esto es un problema ya que los cambios y actualizaciones realizados sobre la información que debe llegar a los técnicos no estaría disponible realmente para ellos si las páginas donde se deben reflejar los cambios se encuentran cacheadas en memoria.

Para evitar esto vamos a obligar a los terminales a descargarse la página del servidor cada vez que los técnicos lo soliciten. Para ello debemos indicar de alguna forma a los navegadores de los móviles que no busquen en memoria caché las posibles páginas WEB almacenadas, sino que las soliciten directamente al servidor.

Esto se va a conseguir mediante las etiquetas **<meta>**, las cuales van incluidas dentro de las páginas WML e indican operaciones a realizar sobre la configuración de funcionamiento de los terminales móviles.

Dentro de todas las páginas WML observamos la inclusión de esta etiqueta, que para el caso que nos ocupa definen una operación sobre la memoria caché del terminal móvil :

```
<head>
<meta http-equiv="Cache-Control" content="max-age=0"/>
</head>
```

Vemos como se define una operación sobre la memoria caché, en la cual estamos indicando que la vida de las páginas WML en la caché sea de **0 seg** mediante **content="max-age=0"**. Con esto tenemos resuelto el problema de la memoria caché del telefono.

La estructura de las páginas WML es bastante simple debido a la limitada capacidad de visualización del display de los terminales móviles que soportan el protocolo WAP. Una página WML va incluida en una estructura denominada **deck**, la cual está delimitada por las etiquetas **<wml>** y **</wml>**. Estas se encuentran divididas en unas estructuras que se conocen como **cards**, igualmente delimitadas por otro par de etiquetas, **<card>** y **</card>**. Una página WML está constituida por una o varias **cards** y todas ellas son descargadas de una sola vez en la correspondiente petición de página al servidor. La diferencia está ahora en que la página WML completa no se muestra en pantalla, sino que solo se mostrará la primera de las **cards**. Estas cards pueden ocupar un tamaño superior al del display, con lo cual será necesario un scroll. Para pasar de unas **cards** a otras tendremos enlaces entre ellas, pero la particularidad ahora es que todas ellas se encuentran en la memoria local del terminal móvil y no es necesario hacer nuevas peticiones al servidor. Estas nuevas peticiones solo se realizan en el caso de que lo que se solicite si sea una nueva página WML.

Por último observamos en esta página la inclusión de una pequeña imagen predefinida. La inclusión de esta imagen no va a ralentizar en exceso nuestro aplicación ya que se trata de una imagen optimizada para visualizarla en la pantalla de estos terminales, la cual es seleccionada entre un grupo ya predefinido que se encuentra en memoria local del teléfono.

5.2.3.2 detalleWap.cgi

Este va a ser el primer **cgi**, al cual se accede desde la página de inicio. Se va a encargar de acceder a la base de datos para obtener los datos relativos a la actuación asignada al técnico de que se trate. Para hacer esto debe ser capaz de identificar de alguna forma qué técnico es el que está solicitando la información. Se consigue gracias al **Gateway**

WAP del operador de telefonía móvil. Para realizar la petición HTTP al servidor WEB de nuestro sistema, el terminal móvil, usando WAP, primero tiene que autentificarse en el **Gateway** para que se compruebe si está dado de alta como usuario del servicio WAP. Durante este proceso, el proxy obtiene el número del terminal que realiza la petición de conexión de datos y lo introduce en una variable de entorno al realizar la conversión de protocolo. Esta variable de entorno se denomina **HTTP_X_UP_SUBNO.** Cuando el servidor ejecute el **cgi** correspondiente, este leerá esta variable de entorno para saber que técnico es el que solicita la información y así poder acceder a la base de datos a por la información adecuada.

Un **cgi** de nuestro servidor dará como respuesta la página WML correspondiente con la información solicitada. Las cabeceras que deben incluir estas páginas y que exige el protocolo WAP son siempre las mismas y, por ello, en el desarrollo de nuestro proyecto hemos utilizada unas librerías que nos son suministradas en los entornos de desarrollo. Para hacer uso de ellas debemos incluir en nuestro código fuente de los **cgi**$_s$ la siguiente línea para incluir el fichero de cabeceras:

```
#include "digest.h"
```

Estas librerías las incluimos en la parte que dedicamos a **código fuente**, y comentamos a continuación las características más importantes de las mismas.

Ya hemos dicho que la función principal de las librerías es crear estas cabeceras y todo aquello que sea común para todas las páginas WML y que su inclusión pueda ser automática. El resto del código WML (individual para cada página) debemos crearlo independientemente para cada caso.

Podemos observar en estas librerías la inclusión de un nuevo elemento que se denomina **digest**. Este elemento representa el volumen máximo de información que se descarga desde el servidor en cada petición a este. Este volumen máximo de transferencia de datos se encuentra limitado a **1492 bytes**. Un **digest** puede estar formado únicamente por un **deck** (página WML), pero puede incluir, además, por la definición de otro tipo de funciones que deben ser realizadas en el terminal móvil y que deben ser enviadas a estos junto con el código WML de las páginas, tales como operaciones de memoria y caché o alertas (este último caso se comentará posteriormente por su relación con nuestro proyecto). Se deberá incluir código en los **cgi**$_s$ para crear estos **digests** y, posteriormente, hacer uso de las distintas funciones de las librerías para ir creando los elementos que van a conformar un **digest** en particular. El código fuente incluido en un **cgi** para crear un **digest** es el siguiente:

```
Digest d;
d = DigestConstruct();
sprintf(texto,"<wml>"
"<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/></head>"
 "<card>"
"<do type=\"accept\" label=\"Inicio\">"
"<go href=\"%s\"/>"
"</do>"
"<p align=\"center\">"
"<br/>"
"<b>Tec: </b><i>%s</i><br/>"
 "<b>SIN ACTUACIONES</b>"
  "</p></card></wml>",url,tec);….
 DigestAddDeck(d, "",texto, NULL);
 output = DigestSerialize(d, &outputLength);
OutputBuffer(output,outputLength);
free(output);
DigestDestruct(d);
```

Vemos como los datos son serializados y enviados y, posteriormente, liberamos la memoria de la máquina utilizada por el servidor para crear el **digest**.

5.2.3.3 franqueoWap.cgi

En esta sección vamos a englobar a todos los **cgi**$_s$ que realizan el franqueo ya que la función que desempeñan todos es la misma cambiando únicamente la información que procesan.

Para el franqueo de una actuación es necesario que el sistema pueda determinar de que actuación se trata y qué técnico se está encargando de la misma. En el **cgi** anterior habíamos conseguido identificar al técnico a partir del número del terminal telefónico desde el que se conectaba. Una vez identificado este el sistema solo tenía que hacer una consulta a la base de datos para obtener el identificador de la actuación que tenía asignada. A partir de aquí se hacían las consultas pertinentes para obtener los datos relativos a la actuación. Cuando comenzamos el franqueo, el proceso es al mismo pero, en vez de los datos relativos a la avería en cuestión, lo que se le envía al técnico son formularios con menús de selección donde aparezcan los posibles equipos instalados y materiales utilizados para esa actuación. Como los menús subsiguientes dependerán de la elección hecha por el técnico en el menú actual, debemos crear una serie de **cgi**$_s$ de franqueo que, a partir de los datos recibidos de la selección en el menú anterior, cree las páginas WML con los formularios correctos para el siguiente menú de selección. Cada vez que se realiza un **submit** en un formulario le estamos enviando al **cgi** siguiente (el cual estará determinado por el enlace creado con el **submit**) una lista de parámetros que irá en la cabecera de la correspondiente petición al servidor. En cada **cgi** leeremos todos estos parámetros y los iremos almacenando en variables locales. Cuando se va a crear la salida del **cgi**, es decir, el siguiente formulario del proceso de franqueo, todas estas variables se introducen como campos ocultos para, de esta forma, tener toda la información disponible en cada paso de franqueo. Los campos ocultos son parte del formulario aunque el usuario (técnico) no los ve y, por ello, al hacer el **submit** de ese formulario, también se pasarán al sistema como parámetros en la cabecera de la petición.

En el último paso de franqueo solo nos quedaría acceder a la base de datos para insertar la información introducida por el técnico. Si todo va bien, y la información introducida es coherente, el sistema responderá enviando al técnico una página WML de aceptación.

5.2.3.4 Notificaciones

El proceso de **notificaciones** es bastante importante en nuestro sistema. Gracias a él el técnico recibe un mensaje corto de texto en su móvil en el que se le comunica que se le ha asignado una nueva actuación y que debe conectarse al sistema. Pretendemos seguir dando este servicio de notificación pero eliminando la interfaz creada con el **Centro de**

Mensajes Cortos de **Movistar**. Esta interfaz no podía ser eliminada ya que, aunque la comunicación con los técnicos ya no hacía uso de ella, si seguía siendo necesaria para el proceso de notificación.

En tecnología WAP se nos ofrece un nuevo servicio que es el de **Alertas** o **Notificaciones**. Este servicio es el que nosotros necesitamos ya que su función es la de notificar a terminales móviles que se ha producido determinados eventos. Podemos configurar nuestro sistema para que, al darse el evento, se le haga llegar una alerta a un terminal móvil concreto. Nuestro sistema se comunicaría con el **proxy WAP** del operador de telefonía móvil que nos suministrara el servicio, y este sería el encargado de que el móvil recibiera la alerta. Como habíamos visto anteriormente, dentro del volumen de información que se le envía de una vez a cada terminal (**digest**) iban incluidas, además del código WML y otros servicios, las alertas. Estas alertas no son más que mensajes de texto que se visualizarían automáticamente en la pantalla del móvil nada más recibir el **digest**. En este mensaje de texto se le informaría brevemente de que determinado evento se ha producido y que debe conectarse al sistema. En nuestro caso se le informaría al técnico de que una nueva actuación le ha sido asignada y que debe conectarse para obtener más información. También le aparecerá un enlace con la dirección correcta del **Web de Técnicos** para que pueda conectarse directamente.

6. CONCLUSIONES

En este capítulo vamos a resumir un poco todo lo visto y, teniendo en cuenta toda la información obtenida, se ofrecerá un conjunto de conclusiones que la realización del proyecto nos ha proporcionado.

Partíamos del problema de un ineficiente proceso de comunicación con los técnicos y decidíamos hacer uso de las nuevas tecnologías móviles para conexiones **on-line,** a través de GSM e Internet, de nuestros técnicos con el sistema.

Proponíamos dos soluciones que se diferenciaban, además de en otra multitud de pequeños detalles, en el protocolo utilizado por los terminales móviles para realizar las conexiones de datos con la base de datos del sistema.

Las pruebas realizadas arrojan que ambas formas de comunicación son eficientes y con ambas se logran los objetivos perseguidos, sobre todo el de la automatización completa del proceso **recepción de actuación – asignación a técnico - resolución de la misma**

La conexión de datos con nuestro servidor WEB usando protocolo HTTP es bastante lenta ya que es un protocolo que no ha sido pensado para su uso en GSM y no utiliza ninguna compresión de datos que minimice el volumen de información a transmitir. Esto, junto con el elevado precio de un terminal móvil Nokia 9110 Communicator frente al resto, nos hace decantarnos en nuestro análisis por las posibilidades que ofrece la tecnología WAP, si bien no todo va a ser ventajas de esta.

El navegador del Nokia 9110 tampoco soporta **Javascript**. Esto obliga a que, cualquier procesamiento de los datos, por pequeño que sea, se deba realizar en el servidor, lo que requiere un elevado número de conexiones con este y la transferencia de un elevado número de datos lo que, debido a las especiales características de lentitud de estas transferencias, da como resultado una baja eficiencia de nuestra aplicación.

La ventaja más importante presentada por el Nokia 9110 es el tamaño de su display. Es bastante mayor que las pantallas disponibles en todos los terminales WAP actuales. Esto nos permite el poder diseñar páginas WEB mucho más elaborada y con un contenido de información mucho mayor, aunque su excesiva lentitud minimiza estas cualidades.

Las conexiones de datos a través de terminales que soportan el protocolo de aplicaciones inalámbricas (WAP) van a ser mucho más rápidas. El reducido tamaño del display nos obliga, aún teniendo la opción de un scroll vertical, a la división de una página WML en numerosas unidades menores de datos, las **cards**, y a crear enlaces entre ellas. El paso de unas **cards** a otras no introduce lentitud ya que estas se encuentran almacenadas todas en la memoria local del teléfono y, sin embargo, nos permite una mayor estructuración de la información. Pero esto último puede ser

perjudicial si creamos una jerarquía muy engorrosa en la cual nuestros usuarios terminen desubicándose.

El lenguaje WML, al ser un lenguaje creado especialmente para los navegadores que vienen integrados con los terminales WAP, no permite crear páginas WEB tan elaboradas como con HTML (la filosofía de WAP es conseguir conexiones de datos rápidas, por lo cual no tiene sentido la existencia de determinados **tags** que existen en HTML y que con WML no sería posible utilizarlos). Si aparecen, por el contrario, nuevos **tags** que permiten el tratamiento de estructuras de datos propias de WAP.

Los terminales WAP todavía no soportan lenguajes de **script** lo que nos obliga a numerosas peticiones al servidor, que es el que realiza el procesamiento de los datos. La próxima aparición de **WMLscript** y de terminales móviles con navegadores que lo soporten nos va a posibilitar la ejecución de pequeñas tareas de procesamiento de datos en el cliente (terminal móvil), con lo cual se conseguirá reducir el numero de accesos al servidor y la consiguiente ganancia de rapidez de la aplicación.

También se ha comentado la existencia del servicio de notificaciones usando WAP dado por los operadores de telefonía digital, en nuestro caso Movistar. Este servicio nos permite eliminar completamente de nuestro sistema la interfaz con el Centro de Mensajes Cortos. Ya se ha comentado la poca fiabilidad de este servicio a la hora de automatizar completamente nuestro sistema, luego la aparición de una alternativa al uso del mismo nos hace decantarnos por la tecnología WAP en nuestra elección.

Los terminales móviles WAP se comercializan a un precio inferior al que tiene el Nokia 9110 Communicator lo que, si pensamos en una empresa como la que estamos tratando con una plantilla muy numerosa de técnicos, nos hace decidirnos por los teléfonos WAP como la mejor opción.

Es un buen momento para reflexionar un poco sobre la aparición de nuevas tecnologías móviles y la adaptación a ellas de las aplicaciones WEB desarrolladas sobre WAP.

Empezamos hablando un poco de la tecnología **GPRS**. Los nuevos servicios basados en GPRS representan un avance muy significativo en la rapidez y economía de la transmisión de datos desde un teléfono móvil y un paso ineludible hacia los sistemas de **3ª Generación** o **UMTS**.

Con GPRS, los usuarios podrán estar permanentemente conectados. No se factura por tiempo, sino por la cantidad de información enviada y recibida (medida en kbit/s). Al tratarse de una nueva tecnología, es necesario diseñar una nueva red de conmutación superpuesta a la red convencional de GSM, y cuya diferencia fundamental es que funciona como conmutación de paquetes y no como conmutación de circuitos. Así, mientras que en GSM una llamada de datos tiene que tener asignado un canal en exclusividad cuando realmente el 90% del tiempo no lo está usando, con GPRS se usan paquetes, a los cuales se le pone la dirección de destino y se envían a través del primer

canal que esté libre, sin utilizarlo en exclusividad, con lo cual el aprovechamiento del recurso radioeléctrico va a ser mucho mayor.

Con GPRS la velocidad de transmisión es muy superior a la actual. Mientras que en GSM los datos circulan a 9,6 kbits, GPRS proporciona velocidades de hasta 50 kbits, dependiendo del tipo de terminal. Actualmente, los terminales disponibles permiten una velocidad de transmisión de 20 kbits, aunque los fabricantes se han comprometido a alcanzar en breve los 50 kbits.

Pero todo esto es transparente a las aplicaciones WEB sobre WAP. Estas relizan conexiones de datos con los servidores de aplicaciones y el modo en que se transporten esos datos no va a influir negativamente. Todo lo contrario, con GPRS, las aplicaciones WAP serán más rápidas y se optimizarán en coste (más reducido al no tener que establecer conexión).

La **3ª Generación** o **UMTS** se basa en GPRS al que se añade un ancho de banda mayor para las comunicaciones móviles, con lo cual se mejorará aún más el rendimiento de las transmisiones de datos y, por tanto, esta aplicación desarrollada sobre WAP seguirá siendo igual de funcional y mucho más optimizada en su rendimiento.

7. BIBLIOGRAFÍA

1.- Oracle error messages and codes manual. Version 6.0.

Dimmick, Shelly

ORACLE

2.- SQL Getting Started

IBM

3.- Data over Wireless Networks

Held, Gilbert

MCGRAW-HILL

4.- Wireless Internet : Applications, Technology & Player Strategies

Maynard, Steve

ARC GROUP

5.- HTTP Pocket Reference

Wong, Clinton

O'REILLY

6.- C++ A GUIDE FOR C PROGRAMMERS

Hekmatpour, Sharam

PRENTICE HALL OF AUSTRALIA PTY, LTD.

7.- C++ Manual de Referencia

Schildt, Herbert

MCGRAW HILL; INTERAMERICANA DE ESPAÑA, S.A.U.

8.- Action Request System: Getting Started Guide

Febrero 2000

REMEDY CORPORATION

9.- Action Request System: User's Guide

Febrero 2000

REMEDY CORPORATION

REFERENCIAS

1.- Wireless Datagram Protocol Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

2.- Wireless Transaction Protocol Especification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

3.- Wireless Transport Layer Security Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

4.- Wireless Session Protocol Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

5.- Wireless Application Environment Overview

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

6.- Wireless Application Environment Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

7.- Wireless Application Protocol Architecture Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

8.- Wireless Markup Language Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

9.- Wireless Telephony Application Interface Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

10.- Wireless Telephony Application Specification

WAP Forum, 30-Abril-98

URL: http://www.wapforum.com/

11.- WAP Binary XML Content Format

WAP Forum, 30-Abril-98

WAP Forum, http://www.wapforum.com

12.- Nokia 9000i Communicator : Internet and E-mail Access Possibilities.

13.- Nokia 9000i Communicator : Quick Guide.

14.- Nokia 9110 Communicator : WWW Browser Style Guide.

PLIEGO DE CONDICIONES

1. CONDICIONANTES TÉCNICOS

Se describen a continuación los requisitos técnicos que van a ser necesarios para el desarrollo e implantación de aplicaciones WEB desarrolladas para el acceso a Internet mediante estos terminales.

Recursos materiales

- La máquina de Informes y Web, debido a su escasa variación por la incorporación de nuevos territorios y nuevas aplicaciones , se presenta de forma única para todo el territorio nacional. Utilizará el hardware existente para realizar la labor de servidor WEB, formado por:

  - **Una SUN E4500 con 6 CPU´s a 250 MHz, 2 MB. de RAM y 2 HD internos de 4.2 GB.**
  - **Un A5000 de 8 x 9.1 GB.**

- Para la instalación de aplicaciones en esta máquina usamos una estación de trabajo **SUN SparcStation 10**, que funciona bajo sistema operativo **Solaris 3.1**.

- Para el desarrollo de código y fuente y compilación se precisa un **PC** con sistema operativo **Windows 95**, **98**, **2000** o **Windows NT**, con procesador **Pentium III 450 Mhz**, **128 MB** de memoria **RAM** y tarjeta de red **Ethernet**.

- Línea **RDSI** para acceso al servidor de aplicaciones desde el exterior.

- Terminal móvil **Nokia 9110 Communicator**.

- Terminales móviles WAP. Los utilizados en este caso han sido :

  - ✓ **Alcatel One Touch**
  - ✓ **Motorola Talkabout**
  - ✓ **Siemens C-35**

- ✓ **Nokia 7110**
- ✓ **Ericsson T-28**
- ✓ **Motorola Timeport**

Recursos lógicos (software)

- Sistemas operativos **Solaris** y **Windows 9x/NT**.
- Compilador de **C++** y precompilador de **ProC**.
- Librerías desarrolladas para crear páginas **WEB** con **HTM**L y **WML**.
- Sistema **Gestor de Actuaciones 3.1**.
- **Netscape Enterprise Server 3.62.**
- Sistema de gestión de base de datos relacionales **Oracle 8.0.**
- Servidor de Workflow **ARS 3.2.**
- Simuladores para aplicaciones inalámbricas:

  - **Nokia WAP ToolKit.**
  - **UP.SDK (Unwired Planet es el navegador de Phone.com).**

PLANOS

PÁGINAS WEB HTML

inicio9110.cgi

```
/* ------------------------------------------------------------------
------- */
//      Fichero: inicio9110.C
//
//  Descripcion: Proceso cgi que sera utilizado como pagina home del
//  Web y que
//  tendra como objetivo decidir entre conexiones de brower 9110
//  o de Browser Windows.
//
/* ------------------------------------------------------------------
------- */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <defines.h>
#include <enlace.h>
extern "C" {
#include <webconf.h>
#include <auxldap.h>
#include <auxsql.h>
          }
#define ERR_VAR_REMOTEUSER          -1
#define ERR_VAR_SERVERURL           -2
#define ERR_PERFIL_USUARIO          -3
#define ERR_CONFIGURACION           -4
#define ERR_ORACLE                  -5

int paginaSinActuaciones9110 (char *nmat);
int error9110 (int codError);

int main ()
{
  int tu;
  char tec[50], pr[50], co[50];
  char cadena[500];
  char entry_id[20];
  char *remoteuser;
  char *serverurl;
  remoteuser = getenv ("REMOTE_USER");
  if (remoteuser == NULL)
    return error9110 (ERR_VAR_REMOTEUSER);
  serverurl = getenv ("SERVER_URL");
  if (serverurl == NULL)
    return error9110 (ERR_VAR_SERVERURL);

  // Si no es un usuario 9110 camino normal
  if (strstr (remoteuser, "9110") == NULL)
    printf("Location: %s%s\n\n", serverurl, PAG_INDICE);

  tu = infoLDAP(tec,pr,co);
```

```
  switch(tu) {
    case LDAP_TECNICO:
    case LDAP_TECPR:
    case LDAP_TECCO: break;
    default:
       return error9110 (ERR_PERFIL_USUARIO);
  }

  // Primero se lee el fichero de configuracion
  if (leewebconf())
    return error9110 (ERR_CONFIGURACION);

  printf ("Pragma: no-cache\n");

  // Si no puedes ver las planificadas y solo la que esta En Curso
  //if (webConf.tecN == 0)
  {
     // Se busca si tiene alguna actuacion En Curso
     int ret = algunaEnCurso9110 (tec, entry_id);

     if (ret == 1403) // Ningun registro
       return paginaSinActuaciones9110 (tec);

     if (ret == 0)
     {
       // Si tiene una actuacion asignada
       /*
       sprintf (cadena, "/cgi-bin/detalle.cgi?ACT=%s?USER=TEC",
entry_id);
       printf("Location: %s%s\n\n", serverurl, cadena);
       */
       char query_string[100];
       sprintf (query_string, "QUERY_STRING=ACT=%s?USER=TEC",
entry_id);
       putenv (query_string);
       execl ("detalle.cgi", "detalle.cgi", NULL);
     }
     else
     {
        return error9110 (ERR_ORACLE);
     };
  }
  /*else
     printf("Location: %s%s\n\n", serverurl, PAG_FRAMETEC); */

  return 0;
}



int error9110 (int codError)
{
  printf ("Pragma: no-cache\n");
  printf ("Content-type: text/html\n\n");
  printf ("<TITLE>WEB-GA: ERROR DE ACCESO</TITLE>\n");
  printf ("<P><FONT SIZE=+1>");
  printf ("<I>Se ha producido un error al acceder al Web del
GA:</I>\n");
  printf ("<UL>\n");
```

```
  if (codError == ERR_VAR_REMOTEUSER)
  {
    printf ("<LI>Variable <B>REMOTE_USER</B> no definida.\n");
    printf ("<LI>Posible control de acceso no activado para el
servidor WEB.\n");
  }


  if (codError == ERR_VAR_SERVERURL)
  {
    printf ("<LI>Variable <B>SERVER_URL</B> no definida.\n");
    printf ("<LI>Posible ejecucion del cgi manual fuera del entorno
WEB.\n");
  }

  if (codError == ERR_PERFIL_USUARIO)
  {
    printf ("<LI>Usuario <B>%s</B> no autorizado.\n", getenv
("REMOTE_USER"));
    printf ("<LI>Posible usuario sin perfil de tecnico o sin numero de
matrícula.\n");
    printf ("<LI>Posible Servidor LDAP parado o no accesible.\n");
  }

  if (codError == ERR_CONFIGURACION)
  {
    printf ("<LI>Fichero de <B>configuración</B> no accesible o con
formato incorrecto.\n");
  }

  if (codError == ERR_ORACLE)
  {
    printf ("<LI>Error de acceso a la base de datos.\n");
    printf ("<LI>Mensaje Oracle: <B>%s</B>\n", errorSQL ());
  }
  printf ("</UL>\n");
  printf ("</FONT>");
  printf ("<P><CENTER><A href=../cgi-
bin/inicio9110.cgi>Reintentar</A></CENTER>");
  return codError;
}

int paginaSinActuaciones9110 (char *nmat)
{
  printf ("Pragma: no-cache\n");
  printf ("Content-type: text/html\n\n");
  printf ("<TITLE>WEB-GA: SIN ACTUACIONES</TITLE>\n");
  printf ("<P>Usuario: <B>%s</B><P>", getenv ("REMOTE_USER"));
  printf ("<P><FONT SIZE=+1><CENTER>");
  printf ("EL TECNICO <B>%s</B> NO TIENE ACTUACION ASIGNADA", nmat);
  printf ("</CENTER></FONT>");
  printf ("<P><CENTER><A href=../cgi-bin/inicio9110.cgi>Consultar</A>
| "
          "<A href=../cgi-bin/tecact.cgi>Lista</A></CENTER>");

  return 0;
}
```

detalle.cgi

```
/* <MESA:01:@(#):MettnadJnj1M:ga2:1.5:990819084431:etna:1 34
38571:MESA> */
//--------------------------------------------------------------------
-------
//
// Proyecto:                    GA
//
// Grupo:                       Tecnico-Contratas
//
// Fichero:                     detalle.C
//
// Autor:
//
// Fecha creacion:              21/01/19999
//
// Fecha ultima modificacion:
//
//------------------------------ DESCRIPCION -----------------------
-------
//
//
//
//------------------------------ RESTRICCIONES ---------------------
-------
//
//------------------------------ COMENTARIOS -----------------------
-------
//
//--------------------------------------------------------------------
-------
//                (c) Telefonica Investigacion y Desarrollo
//--------------------------------------------------------------------
-------


#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <parametros.h>
extern "C" {
#include <webconf.h>
#include <auxldap.h>
#include <auxsql.h>
#include "auxars.h"     //9110
          }

#define TITULO "Detalle"
#define TIT_9110AV "GA-Técnico: Avería"            //9110
#define TIT_9110OS "GA-Técnico: Orden de Servicio" //9110
#define TIT9110ERROR "WEB-GA: ERROR EN DETALLE"    //9110

#define ERR_VAR_REMOTEUSER          -1
```

```
#define ERR_VAR_SERVERURL            -2
#define ERR_PERFIL_USUARIO           -3
#define ERR_CONFIGURACION            -4
#define ERR_ORACLE                   -5

// Prototipo de funcion principal para browser Nokia 9110
int main9110 (); //9110
int error9110 (int codError); //Error con 9110

// Funcion para recortar los ultimos caracteres no visibles
void RecortarBlancos (char *cadena)
{
  for (int i=strlen(cadena)-1;
       (cadena[i]==' ' || cadena[i]=='\n' || cadena[i]=='\r' ||
cadena[i]=='\n');
       i--)
     cadena[i] = '\0';
}

// Función para recortar blancos intermedios usada en main9110

void RecortarBlancosIntermedios (char *cadena)
 {
  int aux1,aux2;
  for (int i=0;
       i < strlen(cadena);
       i++)
         if (cadena[i]==' ' && cadena[i+1]==' ')
           {
            aux1=i+1;
            aux2=i+2;
            while (aux1 < strlen(cadena))
               {
                 cadena[aux1]=cadena[aux2];
                 aux1++;
                 aux2++;
               };
            i--;
           }
 }

int main  ()
{

  char tecnico[TAM_TECNICO+1],
       ptoReunion[TAM_PUNTO_REUNION+1],
       contrata[TAM_CONTRATA+1],
       doc[TAM_DOC+1],
       //!!
       com[TAM_COM+1],
       marca[TAM_DESC_MARCA+1],
       //!!
       estado[TAM_ESTADO+1],
       telef[TAM_TELEFONO+1],
       grupo[TAM_GRUPO+1],
       cadena[500],
       cadena2[500],
       *user = NULL,
       *act = NULL,   // Entry-id de la actuacion
```

```
        *var = NULL,
        *gif,
        *temaAyuda,
        *atras ;
  Parametros *parametros ;
  Enlace     *enlace ;
  char *remoteuser; //9110
  char *serverurl;  //9110

  // Codigo 9110
  remoteuser = getenv ("REMOTE_USER");
  if (remoteuser == NULL)
    return error9110 (ERR_VAR_REMOTEUSER);

  serverurl = getenv ("SERVER_URL");
  if (serverurl == NULL)
    return error9110 (ERR_VAR_SERVERURL);

  // Primero se lee el fichero de configuracion
  if (leewebconf())
    return error9110 (ERR_CONFIGURACION);

  // Si no es un usuario 9110 camino normal
  if (strstr (remoteuser, "9110"))
    return main9110 ();
  // Fin de codigo 9110

  Salida salida (&cout) ; //9110

   /* APV:V1.2.7:18/05/99 */
  int flag_contrata = 0;
  int marcada=0;
  var = getenv ("QUERY_STRING") ;
  parametros = new Parametros (var) ;
  user = parametros->DameValor (CAMPO_USER) ;
  act = parametros->DameValor (CAMPO_ACT) ;
  if (act[0] == 'O') gif = TIT_DETALLEOS ;
  else gif = TIT_DETALLEAV ;
  if (infoLDAP (tecnico, ptoReunion, contrata) > 0) {
  strcpy (doc, "") ;
  strcpy (com, "");
  strcpy (marca, "");

    /* APV:V1.2.7:18/05/99 */
    flag_contrata = !strcmp (user, USER_CO);

    detalle (act, doc, estado, telef, grupo, com, marca,
             flag_contrata) ; //!!

    marcada = (!strcmp (user, USER_CO)) && (strcmp (marca,
"NO_MARCADO") != 0);

    if (!strcmp (estado, "Pendiente")) {
      if (act[0] == 'A') {
        if (!strcmp (user, USER_TEC)) temaAyuda = AYUDA_DET_AV_TEC ;
        else temaAyuda = AYUDA_DET_AV_PR ;
      }
      else {
        if (!strcmp (user, USER_TEC)) temaAyuda = AYUDA_DET_OS_TEC ;
```

```
        else temaAyuda = AYUDA_DET_OS_PR ;
      }
    }
    else {
      if (!strcmp (user, USER_CO)) temaAyuda = AYUDA_FRANQUEO_CON ;
      else {
        if (act[0] == 'A') {
        if (!strcmp (user, USER_TEC)) temaAyuda = AYUDA_FRANQUEO_TEC ;
          else temaAyuda = AYUDA_FRANQUEO_PR ;
        }
        else {
         if (!strcmp (user, USER_TEC)) temaAyuda = AYUDA_CUMPLIM_TEC ;
          else temaAyuda = AYUDA_CUMPLIM_PR ;
        }
      }
    }

    if (!strcmp (user, USER_CO)) atras= PAG_FRAMECO ;
    else if (!strcmp (user, USER_TEC)) atras = PAG_FRAMETEC ;
    else if (!strcmp (user, USER_PR)) atras = PAG_FRAMEPR ;
    salida.CrearCabecera (TITULO, temaAyuda, gif, NULL) ;
    salida.InsertarTextoNeutro ("<CENTER>") ;
    salida.InsertarLineaHoriz () ;
    if (!strcmp (user, USER_TEC)) {
      sprintf (cadena,"<B>T&eacute;cnico:  <FONT COLOR=\"red\">
%s</FONT></B>",
               tecnico) ;
    }
    else
      if (!strcmp (user, USER_PR)) {
        sprintf (cadena,
         "<B>Punto de reuni&oacute;n: <FONT COLOR=\"red\">
%s</FONT></B>",
                 ptoReunion) ;
      }
      else {
        sprintf (cadena,
            "<B>C&oacute;digo de contrata: <FONT COLOR=\"red\">
%s</FONT></B>",
        contrata) ;
      }
    salida.InsertarTextoNeutro (cadena) ;
    salida.InsertarLineaHoriz () ;
    salida.InsertarTextoNeutro ("</CENTER>") ;
    salida.AbrirTabla ("100%", NULL, -1 , 0, 0, NULL) ;
    salida.InicioFilaTabla () ;
    if (act[0] == 'O') {
      sprintf (cadena, "<B>Orden de servicio n&uacute;mero: </B><FONT
COLOR=\"red\"> %s</FONT>", telef) ;
      salida.InsertarEltoTabla (cadena) ;
    }
    else {
      strcpy (cadena2, "");
      if (marcada)
        sprintf (cadena2, "<IMG SRC=\"%s\" ALIGN=LEFT>\n", GIF_MARCA);

      sprintf (cadena, "%s<B>Aver&iacute;a de tel&eacute;fono:
</B><FONT COLOR=\"red\"> %s</FONT>",
               cadena2, telef) ;
```

```
      if (marcada) // Para averias de contratas marcadas
      {
         enlace = new Enlace (CGI_DESMARCAR);
         enlace->InsertarCampo (CAMPO_ACT, act) ;
         enlace->InsertarCampo (CAMPO_USER, user) ;
         enlace->CodificaURL ();
         enlace->InsertarCampoFin ();
         sprintf (cadena2, "<BR><B>Marca: </B>"
                           "<FONT COLOR=\"red\">%s"
                           "<A HREF=%s>(Quitar marca)</A>"
                           "</FONT>", marca, enlace->DameCadena ());
         strcat (cadena, cadena2);
      }
      salida.InsertarEltoTabla (cadena ) ;
    }

    salida.InicioEltoTabla (2) ;

    salida.InsertarGifSensible (BOT_ANTERIOR, "Atr&aacute;s", "atras",
                                atras, NULL);

    if (!strcmp (estado, "Pendiente")) {
      if (strcmp (user, "CO")) {
        if (webConf.asignaMAN == 1) {
          enlace = new Enlace (CGI_ASIGNAR) ;
          enlace->InsertarCampo (CAMPO_ACT, act) ;
          enlace->InsertarCampo (CAMPO_USER, user) ;
          enlace->InsertarCampo (CAMPO_GRUPOS, grupo) ;
          enlace->CodificaURL () ;
          enlace->InsertarCampoFin () ;
          salida.InsertarGifSensible (BOT_ASIGNAR,
"Asignar","asignar",
          enlace->DameCadena (), NULL) ;
          delete enlace ;

        }
      }
    }
    else {
      if (!strcmp (user, "CO")) {
        if ( (!strcmp (estado, "En Curso")) && (!marcada) ) {
          enlace = new Enlace (CGI_FRANQUEAR1) ;
          enlace->InsertarCampo (CAMPO_ACT, act) ;
          enlace->InsertarCampo (CAMPO_USER, user) ;
          enlace->CodificaURL () ;
          enlace->InsertarCampoFin () ;
          salida.InsertarGifSensible (BOT_FRANQUEAR,
"Franquear","franquear",
          enlace->DameCadena (), NULL) ;
          delete enlace ;

          /* ?? */
#ifdef V202
          enlace = new Enlace (CGI_CODTEC1) ;
          enlace->InsertarCampo (CAMPO_ACT, act) ;
          enlace->InsertarCampo (CAMPO_USER, user) ;
          enlace->CodificaURL () ;
          enlace->InsertarCampoFin () ;
```

```
          salida.InsertarGifSensible (BOT_ASIGNAR,"Asignar","codtec",
          enlace->DameCadena (), NULL);
          delete enlace ;
#endif
          /* ?? */
        }
      }
      else {
        if (act[0] == 'A') {
          enlace = new Enlace (CGI_FRANQUEAR1) ;
          enlace->InsertarCampo (CAMPO_ACT, act) ;
          enlace->InsertarCampo (CAMPO_USER, user) ;
          enlace->CodificaURL () ;
          enlace->InsertarCampoFin () ;
          salida.InsertarGifSensible (BOT_FRANQUEAR,
"Franquear","franquear",
          enlace->DameCadena (), NULL) ;
          delete enlace ;
        }
        else {
          enlace = new Enlace (CGI_CUMPLIM_FORM) ;
          enlace->InsertarCampo (CAMPO_ACT, act) ;
          enlace->InsertarCampo (CAMPO_USER, user) ;
          enlace->CodificaURL () ;
          enlace->InsertarCampoFin () ;
          salida.InsertarGifSensible (BOT_CUMPLIM,
"Cumplimentar","cumplim",
          enlace->DameCadena (), NULL) ;
          delete enlace ;
        }
      }
    }
    salida.FinEltoTabla () ;
    salida.FinFilaTabla () ;
    salida.CerrarTabla () ;
    salida.InsertarLineaHoriz () ;

    if (strlen (doc) > 0) {

      RecortarBlancos (doc);
      cout << "<PRE>" << doc << "</PRE>" << endl ;
      cout << "<B>Comentario:</B>";
      cout << "<PRE>" << com << "</PRE>" << endl;
    }
    else {
      char *menError ;
      menError = errorSQL () ;
      salida.InsertarTexto (menError) ;
    }
  }
  else {

    salida.CrearCabecera (TITULO, AYUDA_INDICE, gif, NULL) ;
    salida.InsertarTexto ("Usuario no autorizado") ;
  }
}


// Funcion principal para browser Nokia 9110
```

```
int main9110  ()
{

  char tecnico[TAM_TECNICO+1],
       ptoReunion[TAM_PUNTO_REUNION+1],
       contrata[TAM_CONTRATA+1],
       doc[TAM_DOC+1],

       // Nokia 9110
       tit[TAM_TIT+1],
       domic[TAM_DOMIC+1],
       poblac[TAM_POBLAC+1],
       provinc[TAM_PROV+1],
       tip_client[TAM_TIP_CLIENT+1],
       cent[TAM_CENT+1],
       sint[TAM_SINTOMA+1];

  long c_p,
       fh_recl,
       fh_creac,
       fh_form;

  char tip_av[TAM_TIP_AVIS+1],
       a_plant[TAM_AREA_PLANTA+1],
       grul[TAM_GRUPOL+1],
       p_a_r[TAM_PAR+1],
       denom[TAM_DENOMINAC+1],
       p_ent[TAM_PAR_ENTRADA+1],
       p_sal[TAM_PAR_SALIDA+1],
       c_term[TAM_CAJA_TERM+1],
       res_prueb[TAM_RESULT_PRUEB+1],
       ob_mec[TAM_OBSERVAC_MEC+1],
       sit_c_term[TAM_SITUAC_CAJA_TERM+1],
       sit_c_conex[TAM_SITUAC_CAJA_CONEX+1],
       asig_de[TAM_ASIGNAC_DE+1],
       n_os[TAM_N_OS+1],
       efect[TAM_EFECT+1],
       cl_os[TAM_CLAS_OS+1],
       ent[TAM_ENTIDAD+1],
       n_solic[TAM_N_SOLIC+1],
       clas_abon[TAM_CLAS_ABON+1],
       tipo_serv[TAM_TIP_SERV+1],
       procedenc[TAM_PROCEDENC+1],
       tipo_trab[TAM_TIP_TRAB+1],
       clav_esp[TAM_CLAV_ESP+1],
       observac[TAM_OBSERVAC+1];

  char **eqser;
  int  neqser;
  struct datosEquiposOS *eqos;
  int  neqos;
  struct datosLineasOS *lin_os;
  int  nlineas;

       //!!

  char com[TAM_COM+1],
       marca[TAM_DESC_MARCA+1],
       //!!
```

```
        estado[TAM_ESTADO+1],
        telef[TAM_TELEFONO+1],
        grupo[TAM_GRUPO+1],
        cadena[500],
        cadena2[500],
        *user = NULL,
        *act = NULL,   // Entry-id de la actuacion
        *var = NULL,
        *gif,
        *temaAyuda;

  Parametros *parametros ;
  Enlace     *enlace ;
  struct tm *tiempo ;
  int tu;

  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  /* APV:V1.2.7:18/05/99 */
  int flag_contrata = 0;

  var = getenv ("QUERY_STRING") ;
  parametros = new Parametros (var) ;
  user = parametros->DameValor (CAMPO_USER) ;
  act = parametros->DameValor (CAMPO_ACT) ;

  if (act[0] == 'O') gif = TIT_9110OS ;
  else gif = TIT_9110AV ;

  tu = infoLDAP(tecnico, ptoReunion, contrata);

  // Solo si tiene perfil de tecnico
  if (tu == LDAP_TECNICO || tu == LDAP_TECPR || tu == LDAP_TECCO) {
  strcpy (doc, "") ;
    //!!
    strcpy (com, "");
    strcpy (marca, "");

    /* APV:V1.2.7:18/05/99 */
    flag_contrata = !strcmp (user, USER_CO);

  if (conecta() != 0) return error9110(ERR_ORACLE);
  if (act[0] == 'O')
    detalle9110os(act, doc, estado, n_os, grupo, com, tit, domic,
poblac, provinc, efect,
                  cent, cl_os, &fh_creac, &fh_form, ent, n_solic,
clas_abon, tipo_serv,
                  procedenc, tipo_trab, clav_esp, observac, &eqos,
&neqos,
                  &lin_os, &nlineas,
                  marca, flag_contrata); /*!!*/
  else
    detalle9110av (act, doc, estado, telef, grupo, com, tit, domic,
poblac, provinc,
                   tip_client, cent, sint, &c_p, &fh_recl, tip_av,
a_plant, grul,
                   p_a_r, denom, p_ent, p_sal, c_term, res_prueb,
ob_mec, sit_c_term,
```

```
                    sit_c_conex, asig_de, &eqser, &neqser,
                    marca, flag_contrata); /*!!*/

if (strlen (doc) > 0) {
    Salida salida (&cout) ;
    salida.CrearCabecera (gif);
    salida.InsertarNuevaLinea();
    sprintf (cadena, "<B>Técnico: </B>%s", tecnico) ;
    salida.InsertarTextoNeutro (cadena);
    salida.InsertarTextoNeutro ("<CENTER>") ;
    enlace = new Enlace (CGI_IND9110) ;
    enlace->CodificaURL () ;
    enlace->InsertarCampoFin () ;
    salida.InsertarEnlace (enlace->DameCadena(), "Inicio");
    delete enlace ;
    salida.InsertarTextoNeutro (" | ") ;
    enlace = new Enlace (CGI_TEC_ACT) ;
    enlace->CodificaURL () ;
    enlace->InsertarCampoFin () ;
    salida.InsertarEnlace (enlace->DameCadena(), "Lista");
    delete enlace ;
    if (!strcmp (estado, "Pendiente")) {
      if (webConf.asignaMAN == 1) {
        salida.InsertarTextoNeutro (" | ") ;
        enlace = new Enlace (CGI_ASIGNAR) ;
        enlace->InsertarCampo (CAMPO_ACT, act) ;
        enlace->InsertarCampo (CAMPO_USER, user) ;
        enlace->InsertarCampo (CAMPO_GRUPOS, grupo) ;
        enlace->CodificaURL () ;
        enlace->InsertarCampoFin () ;
        salida.InsertarEnlace ( enlace->DameCadena (),"Asignar") ;
        delete enlace ;
        }
      }
     else {
        salida.InsertarTextoNeutro (" | ") ;
        if (act[0] == 'A') {
        enlace = new Enlace (CGI_FRANQUEAR1) ;
        enlace->InsertarCampo (CAMPO_ACT, act) ;
        enlace->InsertarCampo (CAMPO_USER, user) ;
        enlace->CodificaURL () ;
        enlace->InsertarCampoFin () ;
        salida.InsertarEnlace (enlace->DameCadena (), "Franquear") ;
        delete enlace ;
      }
      else {
       enlace = new Enlace (CGI_CUMPLIM_FORM) ;
       enlace->InsertarCampo (CAMPO_ACT, act) ;
       enlace->InsertarCampo (CAMPO_USER, user) ;
       enlace->CodificaURL () ;
       enlace->InsertarCampoFin () ;
       salida.InsertarEnlace (enlace->DameCadena (),"Cumplimentar");
        delete enlace ;
       }
     }
   salida.InsertarTextoNeutro ("</CENTER>") ;
   salida.InsertarLineaHoriz();

   if (act[0] == 'O') {
```

```
       sprintf (cadena, "<B>Orden de servicio n&uacute;mero:
</B>%s<BR>",n_os ) ;
       salida.InsertarTextoNeutro ("<FONT SIZE='+1'>") ;
       salida.InsertarTextoNeutro (cadena) ;
       sprintf (cadena, "<B>Estado: </B>%s<BR>",estado) ;
       salida.InsertarTextoNeutro (cadena) ;
       salida.InsertarTextoNeutro ("</FONT>") ;
       }
     else {
       sprintf (cadena,"<B>Aver&iacute;a de tel&eacute;fono:
</B>%s<BR>",telef) ;
       salida.InsertarTextoNeutro ("<FONT SIZE='+1'>") ;
       salida.InsertarTextoNeutro (cadena) ;
       sprintf (cadena, "<B>Estado: </B>%s<BR>",estado) ;
       salida.InsertarTextoNeutro (cadena) ;
       salida.InsertarTextoNeutro ("</FONT>") ;
     }

     // Datos de Averias para 9110
     if (act[0] == 'A') {
       salida.InsertarTextoNeutro("<PRE><FONT SIZE=+1>");
       sprintf (cadena,"<B>Titul:</B>%s\n",tit);
       salida.InsertarTextoNeutro(cadena);
       RecortarBlancosIntermedios (domic);
       sprintf (cadena,"<B>Domic:</B>%s\n",domic);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Pobla:</B>%s (%s)\n",poblac,provinc);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>TiCli:</B>%s\n",tip_client);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Areap:</B>%s",a_plant);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"  <B>Centr:</B>%s\n",cent);
       salida.InsertarTextoNeutro(cadena);
       tiempo = localtime (&fh_recl) ;
       sprintf (cadena,"<B>FeRec:</B>%02d/%02d/%4d %02d:%02d",
       tiempo->tm_mday, tiempo->tm_mon+1, tiempo->tm_year+1900,
       tiempo->tm_hour, tiempo->tm_min) ;
       salida.InsertarTextoNeutro (cadena);
       if (c_p > 0)
       {
         tiempo = localtime (&c_p);
         sprintf (cadena,"  <B>Citap:</B>%02d/%02d/%4d %02d:%02d",
         tiempo->tm_mday, tiempo->tm_mon+1, tiempo- >tm_year+1900,
         tiempo->tm_hour, tiempo->tm_min);
         salida.InsertarTextoNeutro (cadena) ;
       }
       salida.InsertarTextoNeutro ("\n") ;

       sprintf (cadena,"<B>  Grupo  Par  Denomi Paren Parsa
Cajter</B>\n");
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena, "  %s %s %s %s %s %s \n",
                grul, p_a_r, denom, p_ent, p_sal, c_term);
       salida.InsertarTextoNeutro(cadena);

       RecortarBlancosIntermedios (sit_c_term);
       sprintf (cadena,"<B>SitCT:</B>%s\n",sit_c_term);
       salida.InsertarTextoNeutro(cadena);
```

```
        RecortarBlancosIntermedios (sit_c_conex);
        sprintf (cadena,"<B>SitCC:</B>%s\n",sit_c_conex);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Asign:</B>%s\n",asig_de);
        salida.InsertarTextoNeutro(cadena);

        for (int j=0; j<neqser; j++)
        {
          sprintf (cadena,"<B> Eq%2d:</B>%s\n", j+1, eqser[j]);
          salida.InsertarTextoNeutro(cadena);
        };

        sprintf (cadena,"<B>RePru:</B>%s\n",res_prueb);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Sínto:</B>%s\n",sint);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Aviso:</B>%s\n",tip_av);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Obser:</B>%s\n",ob_mec);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Comen:</B>%s\n",com);
        salida.InsertarTextoNeutro(cadena);
        salida.InsertarTextoNeutro("</FONT></PRE>");
      }

      else { // Caso de ordenes de servicio

        salida.InsertarTextoNeutro("<PRE><FONT SIZE='+1'>");
        sprintf (cadena,"<B>Titul:</B>%s\n",tit);
        salida.InsertarTextoNeutro(cadena);
        RecortarBlancosIntermedios (domic);
        sprintf (cadena,"<B>Domic:</B>%s\n",domic);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Pobla:</B>%s (%s)\n",poblac,provinc);
        salida.InsertarTextoNeutro(cadena);
        tiempo = localtime (&fh_creac);
        sprintf (cadena,"<B>FeCre:</B>%02d/%02d/%4d %02d:%02d",
        tiempo->tm_mday, tiempo->tm_mon+1, tiempo->tm_year+1900,
        tiempo->tm_hour, tiempo->tm_min);
        salida.InsertarTextoNeutro (cadena);
        tiempo = localtime (&fh_form) ;
        sprintf (cadena,"<B>  FeFor:</B>%02d/%02d/%4d %02d:%02d\n",
        tiempo->tm_mday, tiempo->tm_mon+1, tiempo->tm_year+1900,
        tiempo->tm_hour, tiempo->tm_min) ;
        salida.InsertarTextoNeutro (cadena) ;
        sprintf (cadena,"<B>ClaOS:</B>%s\n",cl_os);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Centr:</B>%s",cent);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"  <B>Entid:</B>%s",ent);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"  <B>NuSol:</B>%s\n",n_solic);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>ClAbo:</B>%s\n",clas_abon);
        salida.InsertarTextoNeutro(cadena);
 sprintf (cadena,"<B>Traba:</B>%s/%s/\n%s\n",tipo_serv, procedenc,
tipo_trab);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>ClEsp:</B>%s\n",clav_esp);
```

```
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Obser:</B>%s\n",observac);
        salida.InsertarTextoNeutro(cadena);
        sprintf (cadena,"<B>Comen:</B>%s\n",com);
        salida.InsertarTextoNeutro(cadena);
        salida.InsertarTextoNeutro("<B>DATOS EQUIPOS:</B>\n");

        for (int j=0; j<neqos; j++)
        {
          sprintf (cadena,"<B> Eq%2d:</B>\n",j+1);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena,"   <B>Ope:</B>%s <B>Uni:</B>%s
<B>Prol:</B>%s\n",
          eqos[j].operac, eqos[j].unidades,eqos[j].prolong);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena,"   <B>Equ:</B>%s\n", eqos[j].equipo);
          salida.InsertarTextoNeutro(cadena);
        };

        salida.InsertarTextoNeutro("<B>DATOS LINEAS:</B>\n");
        for (int k=0; k<nlineas; k++)
        {
          sprintf (cadena," <B>Li%2d:</B>\n",k+1);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena,"   <B>Ope:</B>%s",lin_os[k].operac2);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena," <B>Lin:</B>%s",lin_os[k].linea);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena," <B>Cen:</B>%s\n",lin_os[k].central);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena,"   <B>Tel:</B>%s",lin_os[k].telefono_linea);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena," <B>Asi:</B>%s\n",lin_os[k].asign);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena,"   <B>Gru:</B>%s",lin_os[k].grupo);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena," <B>Par:</B>%s",lin_os[k].par);
          salida.InsertarTextoNeutro(cadena);
          sprintf (cadena," <B>Caj:</B>%s\n",lin_os[k].caja);
          salida.InsertarTextoNeutro(cadena);
          RecortarBlancosIntermedios (lin_os[k].domicilio);
          sprintf (cadena,"   <B>Dom:</B>%s\n",lin_os[k].domicilio);
          salida.InsertarTextoNeutro(cadena);
        };

     salida.InsertarTextoNeutro("</FONT></PRE>");
     } // Fin de if os
    }
    else return error9110(ERR_ORACLE);
   }
  else return error9110(ERR_PERFIL_USUARIO);
  return 0;
}


int error9110 (int codError) //Error con 9110
{
  char cadena[200];
  Enlace *enlace ;
```

```
  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  Salida salida (&cout) ;
  salida.CrearCabecera (TIT9110ERROR);
  salida.InsertarTextoNeutro ("<P> <FONT SIZE=+1>");
  salida.InsertarTextoNeutro ("<I>Se ha producido un error al obtener
el detalle de una actuación:</I>\n");
  salida.InsertarTextoNeutro ("<UL>\n");

  if (codError == ERR_VAR_REMOTEUSER)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>REMOTE_USER</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible control de acceso no
activado para el servidor WEB.\n");
  };

  if (codError == ERR_VAR_SERVERURL)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>SERVER_URL</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible ejecucion del cgi manual
fuera del entorno WEB.\n");
  };

  if (codError == ERR_PERFIL_USUARIO)
  {
    sprintf (cadena, "<LI>Usuario <B>%s</B> no autorizado.\n", getenv
("REMOTE_USER"));
    salida.InsertarTextoNeutro (cadena);
    salida.InsertarTextoNeutro ("<LI>Posible usuario sin perfil de
técnico o sin numero de matrícula.\n");
    salida.InsertarTextoNeutro ("<LI>Posible Servidor LDAP parado o no
accesible.\n");
  }

  if (codError == ERR_CONFIGURACION)
  {
    salida.InsertarTextoNeutro ("<LI>Fichero de <B>configuración</B>
no accesible o con formato incorrecto.\n");
  }

  if (codError == ERR_ORACLE)
  {
    salida.InsertarTextoNeutro ("<LI>Error de acceso a la base de
datos.\n");
    sprintf (cadena, "<LI>Mensaje Oracle: <B>%s</B>\n", errorSQL ());
    salida.InsertarTextoNeutro (cadena);
  };

  salida.InsertarTextoNeutro ("<CENTER><P>");

  // Enlace Continuar
  enlace = new Enlace (CGI_IND9110) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Continuar");
```

```
  salida.InsertarTextoNeutro ("</CENTER>");
  return codError;
}
```

tecact.cgi

```
/* <MESA:01:@(#):MettnadJnj5g:ga2:1.2:990326154401:etna:1 34
60216:MESA> */
//--------------------------------------------------------------------
-------
//
// Proyecto:                    GA
//
// Grupo:                       Tecnico-Contratas
//
// Fichero:                     tecact.C
//
// Autor:
//
// Fecha creacion:              19/01/19999
//
// Fecha ultima modificacion:
//
//------------------------------- DESCRIPCION ----------------------
-------
//
//
//
//------------------------------ RESTRICCIONES ---------------------
-------
//
//------------------------------- COMENTARIOS ----------------------
-------
//
//--------------------------------------------------------------------
-------
//                (c) Telefonica Investigacion y Desarrollo
//--------------------------------------------------------------------
-------


#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
extern "C" {
#include <webconf.h>
#include <auxldap.h>
#include <auxsql.h>
#include "auxars.h"      //9110
          }

#define TITULO     "T&eacute;cnico"
```

```
#define TITULO9110 "GA-Técnico: Actuaciones"       //9110
#define TIT9110ERROR "WEB-GA: ERROR DE BUSQUEDA"           //9110
#define ERR_VAR_REMOTEUSER           -1
#define ERR_VAR_SERVERURL            -2
#define ERR_PERFIL_USUARIO           -3
#define ERR_CONFIGURACION            -4
#define ERR_ORACLE                   -5

// Prototipo de funcion principal para browser Nokia 9110
int main9110 (); //9110
int error9110 (int codError); //Error con 9110

int main  ()
{
  char telefono[TAM_TELEFONO+1],
       tecnico[TAM_TECNICO+1],
       ptoReunion[TAM_PUNTO_REUNION+1],
       contrata[TAM_CONTRATA+1],
       actuacion[TAM_ACTUACION+1],
       estado[TAM_ESTADO+1],
       aptm[TAM_APLANTA_TMERCADO+1],
       cadena[100],
       seguir = 0 ;
  int  contPend = 0,
       contAct = 0 ;
  long feInicio,
       feFin ;
  Enlace    *enlace ;
  struct tm *tiempo ;
  char *remoteuser; //9110
  char *serverurl;  //9110

  // Codigo 9110
  remoteuser = getenv ("REMOTE_USER");
  if (remoteuser == NULL)
    return error9110 (ERR_VAR_REMOTEUSER);

  serverurl = getenv ("SERVER_URL");
  if (serverurl == NULL)
    return error9110 (ERR_VAR_SERVERURL);

  // Primero se lee el fichero de configuracion
  if (leewebconf())
    return error9110 (ERR_CONFIGURACION);

  // Si no es un usuario 9110 camino normal
  if (strstr (remoteuser, "9110"))
    return main9110 ();
  // Fin de codigo 9110
  Salida salida (&cout) ;
  salida.CrearCabecera (TITULO, AYUDA_TABLA_TEC, GIF_TECNICO, NULL,
                        FUNCIONES_JS) ;

  if (infoLDAP (tecnico, ptoReunion, contrata) > 0) {
    salida.AbrirTabla (NULL, NULL, -1, 2, 0 , NULL) ;
    salida.InicioFilaTabla () ;
    sprintf (cadena,
             "<B>T&eacute;cnico:   <FONT COLOR=\"red\">%s</FONT></B>",
             tecnico) ;
```

```
    salida.InsertarEltoNegritaTabla (cadena) ;
    salida.FinFilaTabla () ;
    salida.CerrarTabla () ;
    salida.InsertarTexto ("<CENTER>") ;
    salida.AbrirTabla ("745", NULL, 0, 0, 0, NULL) ;
    salida.InicioFilaTabla () ;
    salida.InicioEltoTabla (1) ;
    sprintf (cadena, "\"javascript:parent.location='%s';\"",
             PAG_INDICE ) ;
    salida.InsertarGifSensible (BOT_ANTERIOR, "Atr&aacute;s", "atras",
                                cadena, NULL) ;
    salida.FinEltoTabla () ;
    salida.InicioEltoTabla (1) ;
    enlace = new Enlace (CGI_LIST_DET) ;
    enlace->InsertarCampo (CAMPO_USER, "TEC") ;
    enlace->CodificaURL () ;
    enlace->InsertarCampoFin () ;
    salida.InsertarGifSensible (BOT_LIST_DET, "Lista de detalles",
"lista",
    enlace->DameCadena (), NULL) ;
    delete enlace ;
    salida.FinEltoTabla () ;
    salida.InicioEltoTabla (1) ;
    salida.InsertarGifSensible (BOT_CONSULT, "Consultar", "consultar",
             "\"javascript:window.location.reload(true);\"",NULL) ;
    salida.FinEltoTabla () ;
    salida.FinFilaTabla () ;
    salida.CerrarTabla () ;
    salida.InsertarLineaHoriz () ;
      if (!tecactIni (tecnico)) {
      seguir = 1 ;
      while (seguir) {
        if (tecactSig (actuacion, telefono, estado, aptm, &feInicio,
&feFin)) {
          seguir = 0 ;
        }
        else {
          /* Parche APV */
          if (!strcmp (estado, "Pendiente")) {
            contPend ++;
            if (contPend > webConf.tecN)
            {
              seguir = 0;
              continue;
            }
          };
          contAct ++ ;
          if (contAct == 1) {
            salida.AbrirTabla ("745", NULL, 1, 1,0, COLOR_TABLA) ;
            salida.InicioFilaTabla (COLOR_TABLA_LETRA) ;
            salida.InsertarEltoNegritaTabla ("Tipo", 1,
                                             COLOR_TABLA_FONDO) ;
            salida.InsertarEltoNegritaTabla ("Tel&eacute;f/Nº
OS",1,COLOR_TABLA_FONDO);
            salida.InsertarEltoNegritaTabla ("Estado",
1,COLOR_TABLA_FONDO) ;
            salida.InsertarEltoNegritaTabla ("&Aacute;rea/Mercado",
1,COLOR_TABLA_FONDO );
```

```
            salida.InsertarEltoNegritaTabla ("Fecha inicio",
1,COLOR_TABLA_FONDO) ;
            salida.InsertarEltoNegritaTabla ("Fecha
fin",1,COLOR_TABLA_FONDO) ;
            salida.FinFilaTabla () ;
          }
          salida.InicioFilaTabla () ;
          strcpy (cadena, actuacion) ;
          if (cadena[0] == 'A') cadena[1] = 'V' ;
          else cadena[1] = 'S' ;
          cadena[2] = '\0' ;
          salida.InsertarEltoTabla (cadena, 1) ;
          salida.InicioEltoTabla (1) ;
          enlace = new Enlace (CGI_DET_ACT) ;
          enlace->InsertarCampo (CAMPO_ACT, actuacion) ;
          enlace->InsertarCampo (CAMPO_USER, "TEC") ;
          enlace->CodificaURL () ;
          enlace->InsertarCampoFin () ;
          salida.InsertarEnlace (enlace->DameCadena (), telefono) ;
          delete enlace ;
          salida.FinEltoTabla () ;
          salida.InsertarEltoTabla (estado, 1) ;
          salida.InsertarEltoTabla (aptm, 1) ;
          tiempo = localtime (&feInicio) ;
          sprintf (cadena, "%02d/%02d/%4d %02d:%02d",
                   tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                  tiempo->tm_hour, tiempo->tm_min) ;

          salida.InsertarEltoTabla (cadena, 1) ;
          tiempo = localtime (&feFin) ;
          sprintf (cadena, "%02d/%02d/%4d %02d:%02d",
                   tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                   tiempo->tm_hour, tiempo->tm_min) ;
          salida.InsertarEltoTabla (cadena, 1) ;
          salida.FinFilaTabla () ;
        }
      } // while
      if (contAct)
        salida.CerrarTabla () ;
      else salida.InsertarTexto ("<CENTER><FONT COLOR=red>NO HAY
ACTUACIONES PARA ESTE T&Eacute;CNICO</FONT></CENTER>");
    }
    else {
      char *menError ;
      menError = errorSQL () ;
      salida.InsertarTexto (menError) ;
    }
  }
  else {
    salida.InsertarTexto ("Usuario no autorizado") ;
  };
}


// Funcion principal para browser Nokia 9110
int main9110 ()
{
```

```
  char telefono[TAM_TELEFONO+1],
       tecnico[TAM_TECNICO+1],
       ptoReunion[TAM_PUNTO_REUNION+1],
       contrata[TAM_CONTRATA+1],
       actuacion[TAM_ACTUACION+1],
       estado[TAM_ESTADO+1],
       aptm[TAM_APLANTA_TMERCADO+1],
       cadena[100],
       seguir = 0 ;
  int  contPend = 0,
       contAct = 0 ;
  long feInicio,
       feFin ;
  Enlace    *enlace ;
  struct tm *tiempo ;
  int tu;
  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;
  tu = infoLDAP(tecnico, ptoReunion, contrata);
  // Solo si tiene perfil de tecnico
  if (tu == LDAP_TECNICO || tu == LDAP_TECPR || tu == LDAP_TECCO) {
    if (!tecactIni (tecnico)) {
      seguir = 1 ;
      Salida salida (&cout) ;
      salida.CrearCabecera (TITULO9110) ;
      salida.InsertarNuevaLinea () ;
      while (seguir) {
        if (tecactSig (actuacion, telefono, estado, aptm, &feInicio,
&feFin))
          seguir = 0 ;
        else {
          /* Parche APV */
          if (!strcmp (estado, "Pendiente")) {
            contPend ++;
            if (contPend > webConf.tecN)
            {
              seguir = 0;
              continue;
            }
          };
          contAct ++ ;
          if (contAct == 1) {
            sprintf (cadena, "<B>Técnico: </B>%s", tecnico) ;
            salida.InsertarTextoNeutro (cadena);
            salida.InsertarTextoNeutro ("<CENTER>") ;
            salida.InsertarEnlace (CGI_IND9110, "Inicio") ;
            salida.InsertarTextoNeutro (" | ") ;
            enlace = new Enlace (CGI_LIST_DET) ;
            enlace->InsertarCampo (CAMPO_USER, "TEC") ;
            enlace->CodificaURL () ;
            enlace->InsertarCampoFin () ;
            salida.InsertarEnlace (enlace->DameCadena(), "Listar
detalles");
            delete enlace ;
            salida.InsertarTextoNeutro (" | ") ;
            salida.InsertarEnlace (CGI_TEC_ACT, "Consultar") ;
            salida.InsertarTextoNeutro ("</CENTER>") ;
            salida.InsertarLineaHoriz () ;
```

```
          }
          if (actuacion[0] == 'A') strcpy (cadena,"<FONT
SIZE='+1'><B>Avería Tfno: </B>");
          else strcpy (cadena, "<FONT SIZE='+1'><B>Orden de servicio
Nº: </B>");
          salida.InsertarTextoNeutro (cadena);
          enlace = new Enlace (CGI_DET_ACT) ;
          enlace->InsertarCampo (CAMPO_ACT, actuacion) ;
          enlace->InsertarCampo (CAMPO_USER, "TEC") ;
          enlace->CodificaURL () ;
          enlace->InsertarCampoFin () ;
          salida.InsertarEnlace (enlace->DameCadena (), telefono) ;
          delete enlace ;
          salida.InsertarTextoNeutro ("</FONT><UL>");
          sprintf(cadena,"<LI><B>Estado: </B>%s",estado);
          salida.InsertarTextoNeutro (cadena) ;
           if (actuacion[0] == 'A') sprintf (cadena, "<LI><B>Area de
planta: </B>%s",aptm);
          else sprintf (cadena, "<LI><B>Tipo de mercado:
</B>%s",aptm);
          salida.InsertarTextoNeutro (cadena);
           tiempo = localtime (&feInicio) ;
          sprintf (cadena, "<LI><B>Fecha inicio: </B>%02d/%02d/%4d
%02d:%02d",
                   tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                   tiempo->tm_hour, tiempo->tm_min) ;
          salida.InsertarTextoNeutro (cadena) ;
          tiempo = localtime (&feFin) ;
          sprintf (cadena, "<LI><B>Fecha fin: </B>%02d/%02d/%4d
%02d:%02d</UL>",
                   tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                   tiempo->tm_hour, tiempo->tm_min) ;
          salida.InsertarTextoNeutro (cadena) ;
        }
      } // while
      if (!contAct) {
        // Enlace Continuar
        salida.InsertarTextoNeutro ("<CENTER>") ;
        enlace = new Enlace (CGI_TEC_ACT) ;
        enlace->CodificaURL ();
        enlace->InsertarCampoFin ();
        salida.InsertarEnlace (enlace->DameCadena(), "Reintentar");
        salida.InsertarTextoNeutro ("</CENTER>");
        salida.InsertarLineaHoriz () ;
        salida.InsertarTextoNeutro ("<FONT SIZE='+1'><B><CENTER>NO HAY
ACTUACIONES PARA ESTE T&Eacute;CNICO</CENTER></B></FONT>");
        }
    }
    else return error9110(ERR_ORACLE);
  }
   else return error9110(ERR_PERFIL_USUARIO);
  return 0;
}
int error9110 (int codError) //Error con 9110
{
  char cadena[200];
  Enlace *enlace ;
```

```
  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  Salida salida (&cout) ;
  salida.CrearCabecera (TIT9110ERROR);
  salida.InsertarTextoNeutro ("<P> <FONT SIZE=+1>");
  salida.InsertarTextoNeutro ("<I>Se ha producido un error al buscar
actuación:</I>\n");
  salida.InsertarTextoNeutro ("<UL>\n");

  if (codError == ERR_VAR_REMOTEUSER)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>REMOTE_USER</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible control de acceso no
activado para el servidor WEB.\n");
  };

  if (codError == ERR_VAR_SERVERURL)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>SERVER_URL</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible ejecucion del cgi manual
fuera del entorno WEB.\n");
  };

  if (codError == ERR_PERFIL_USUARIO)
  {
    sprintf (cadena, "<LI>Usuario <B>%s</B> no autorizado.\n", getenv
("REMOTE_USER"));
    salida.InsertarTextoNeutro (cadena);
    salida.InsertarTextoNeutro ("<LI>Posible usuario sin perfil de
técnico o sin numero de matrícula.\n");
    salida.InsertarTextoNeutro ("<LI>Posible Servidor LDAP parado o no
accesible.\n");
  }

  if (codError == ERR_CONFIGURACION)
  {
    salida.InsertarTextoNeutro ("<LI>Fichero de <B>configuración</B>
no accesible o con formato incorrecto.\n");
  }

  if (codError == ERR_ORACLE)
  {
    salida.InsertarTextoNeutro ("<LI>Error de acceso a la base de
datos.\n");
    sprintf (cadena, "<LI>Mensaje Oracle: <B>%s</B>\n", errorSQL ());
    salida.InsertarTextoNeutro (cadena);
  };

  salida.InsertarTextoNeutro ("<CENTER><P>");

  // Enlace Continuar
  enlace = new Enlace (CGI_TEC_ACT) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Reintentar");
```

```
  salida.InsertarTextoNeutro ("</CENTER>");
  return codError;
}
```

listadet.cgi

```
/* <MESA:01:@(#):MettnadJnj2M:ga2:1.4:990622102914:etna:1 34
49002:MESA> */
//--------------------------------------------------------------------
-------
//
// Proyecto:                    GA
//
// Grupo:                       Tecnico-Contratas
//
// Fichero:                     listadet.C
//
// Autor:
//
// Fecha creacion:              19/01/19999
//
// Fecha ultima modificacion:
//
//------------------------------- DESCRIPCION -----------------------
-------
//
//
//
//------------------------------ RESTRICCIONES ----------------------
-------
//
//------------------------------- COMENTARIOS -----------------------
-------
//
//--------------------------------------------------------------------
-------
//                (c) Telefonica Investigacion y Desarrollo
//--------------------------------------------------------------------
-------


#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <entrada.h>
#include <parametros.h>
extern "C" {
#include <webconf.h>
#include <auxldap.h>
#include <auxsql.h>
#include "auxars.h"         //9110
        }
#define TITULO "Lista de detalles"
#define TITULO9110 "GA-Detalle de actuaciones"     //9110
```

```
#define TIT9110ERROR "WEB-GA: ERROR EN DETALLES"    //9110
#define ERR_VAR_REMOTEUSER           -1
#define ERR_VAR_SERVERURL            -2
#define ERR_PERFIL_USUARIO           -3
#define ERR_CONFIGURACION            -4
#define ERR_ORACLE                   -5


// Prototipo de funcion principal para browser Nokia 9110
int main9110 (); //9110
int error9110 (int codError); //Error con 9110

// Funcion para recortar los ultimos caracteres no visibles
void RecortarBlancos (char *cadena)
{
  for (int i=strlen(cadena)-1;
       (cadena[i]==' ' || cadena[i]=='\n' || cadena[i]=='\r' ||
cadena[i]=='\n');
       i--)
     cadena[i] = '\0';
}

// Función para recortar blancos intermedios usada en 9110

void RecortarBlancosIntermedios (char *cadena)
 {
  int aux1,aux2;
  for (int i=0;
       i < strlen(cadena);
       i++)
         if (cadena[i]==' ' && cadena[i+1]==' ')
           {
            aux1=i+1;
            aux2=i+2;
            while (aux1 < strlen(cadena))
               {
                 cadena[aux1]=cadena[aux2];
                 aux1++;
                 aux2++;
               };
            i--;
           }

 }

int main  ()
{

  char tecnico[TAM_TECNICO+1],
       ptoReunion[TAM_PUNTO_REUNION+1],
       contrata[TAM_CONTRATA+1],
       cadena[100],
       cadena2[100],
       doc[TAM_DOC+1],
       //!!
       com[TAM_COM+1],
       marca[TAM_DESC_MARCA+1],
       //!!
       act[TAM_ACTUACION+1],
```

```
     telef[TAM_TELEFONO+1],
     estado[TAM_ESTADO+1],
     grupoS[TAM_GRUPO+1],
     *todosGrupos,
     *cadEnCur = NULL,
     *cadPlanif = NULL,
     *cadAver = NULL,
     *cadOs = NULL,
     *grupo = NULL,
     *user = NULL,
     *var = NULL,
     *temaAyuda ;
int  planif = 1,
     encur = 1,
     aver = 1,
     os = 1,
     error = 0 ;
Entrada *entrada ;
Parametros *parametros ;

char *cond_todosEstados = NULL,
     *cond_estado = NULL;

int contPend = 0;
int marcada = 0;

char *remoteuser; //9110
char *serverurl;  //9110

// Codigo 9110
remoteuser = getenv ("REMOTE_USER");
if (remoteuser == NULL)
  return error9110 (ERR_VAR_REMOTEUSER);

serverurl = getenv ("SERVER_URL");
if (serverurl == NULL)
  return error9110 (ERR_VAR_SERVERURL);

// Primero se lee el fichero de configuracion
if (leewebconf())
  return error9110 (ERR_CONFIGURACION);

// Si no es un usuario 9110 camino normal
if (strstr (remoteuser, "9110"))
  return main9110 ();
// Fin de codigo 9110
Salida salida (&cout) ; //9110
var = getenv ("QUERY_STRING") ;
if (!var) {
  entrada = new Entrada (&cin) ;
  user = entrada->DameValor (CAMPO_USER) ;
  todosGrupos = entrada->DameValor (CAMPO_TODOS_GRUPOS) ;
  cadEnCur = entrada->DameValor (CAMPO_EN_CURSO) ;
  cadPlanif = entrada->DameValor (CAMPO_PLANIFICADA) ;
  cadAver = entrada->DameValor (CAMPO_AVERIA) ;
  cadOs = entrada->DameValor (CAMPO_INSTALACIONES) ;
  grupo       = entrada->DameValor (CAMPO_GRUPOS) ;

  /* APV:V1.2.7:18/05/99 */
```

```
    cond_todosEstados = entrada->DameValor (CAMPO_CO_TODEST) ;
    cond_estado = entrada->DameValor (CAMPO_CO_ESTADOS) ;
  }
  else {
    parametros = new Parametros (var) ;
    user = parametros->DameValor (CAMPO_USER) ;
    todosGrupos = parametros->DameValor (CAMPO_TODOS_GRUPOS) ;
    cadEnCur = parametros->DameValor (CAMPO_EN_CURSO) ;
    cadPlanif = parametros->DameValor (CAMPO_PLANIFICADA) ;
    cadAver = parametros->DameValor (CAMPO_AVERIA) ;
    cadOs = parametros->DameValor (CAMPO_INSTALACIONES) ;
    grupo       = parametros->DameValor (CAMPO_GRUPOS) ;

    /* APV:V1.2.7:18/05/99 */
    cond_todosEstados = parametros->DameValor (CAMPO_CO_TODEST) ;
    cond_estado = parametros->DameValor (CAMPO_CO_ESTADOS) ;
  }
    if (todosGrupos) {
    if (!strcmp (todosGrupos, "s")) grupo = NULL ;
    if (!cadEnCur ) encur = 0 ;
    if (!cadPlanif ) planif = 0 ;
    if (!cadAver ) aver = 0 ;
    if (!cadOs ) os = 0 ;
  }

  /* APV:V1.2.7:18/05/99 */
  if (cond_todosEstados)
    if (!strcmp (cond_todosEstados, "s")) cond_estado = NULL;
  if (!strcmp (user, "TEC")) temaAyuda = AYUDA_LIST_DET_TEC ;
  else
    if (!strcmp (user, "CO")) temaAyuda = AYUDA_LIST_DET_CON ;
    else  temaAyuda = AYUDA_LIST_DET_PR ;
  salida.CrearCabecera (TITULO, temaAyuda, GIF_LIST_DET, NULL) ;
  if (infoLDAP (tecnico, ptoReunion, contrata) > 0) {
    strcpy (com, "");
    strcpy (marca, "");
    salida.InsertarLineaHoriz () ;
    salida.InsertarTextoNeutro ("<CENTER>") ;
    if (!strcmp (user, "TEC"))
      sprintf (cadena, "<B>T&eacute;cnico: <FONT COLOR=
red>%s</FONT></B>",
               tecnico) ;
    else {
      if (!strcmp (user, "CO"))
      sprintf (cadena,
            "<B>C&oacute;digo de contrata: <FONT COLOR=
red>%s</FONT></B>",
             contrata) ;
      else
      sprintf (cadena,
         "<B>Punto de reuni&oacute;n: <FONT COLOR= red>%s</FONT></B>",
             ptoReunion) ;
    }

    salida.InsertarTextoNeutro (cadena) ;
    salida.InsertarLineaHoriz () ;
    salida.InsertarTextoNeutro ("</CENTER>") ;
    salida.AbrirTabla ("100%", NULL) ;
    salida.InicioFilaTabla () ;
```

```
sprintf (cadena,"<B>Lista de boletines de actuaciones</B>") ;
salida.InsertarEltoTabla (cadena) ;
salida.InicioEltoTabla (2) ;
salida.InsertarGifSensible (BOT_ANTERIOR, "Atr&aacute;s", "atras",
                    "\"javascript:atras('');\"",
                    "\"atras('');\"") ;
salida.FinEltoTabla () ;
salida.FinFilaTabla () ;
salida.CerrarTabla () ;
salida.InsertarLineaHoriz () ;
contPend = 0;
if (!strcmp (user, "TEC")) {
  if (!teclistadetIni (tecnico))
  while (!teclistadetSig (doc, act, telef, estado, com)) { //!!

      /* Parche APV */
      if (!strcmp (estado, "Pendiente")) {
        contPend ++;
        if (contPend > webConf.tecN)
          continue;
      };
    salida.AbrirTabla () ;
    salida.InicioFilaTabla () ;
    if (act[0] == 'O')
      sprintf (cadena, "Orden de servicio n&uacute;mero:") ;
    else
      sprintf (cadena,   "Aver&iacute;a de tel&eacute;fono:") ;
    salida.InsertarEltoNegritaTabla (cadena, 2) ;
    sprintf (cadena, "<FONT COLOR='red'>%s</FONT>",  telef) ;
    salida.InsertarEltoTabla (cadena) ;
    salida.FinFilaTabla () ;
    salida.InicioFilaTabla () ;
    salida.InsertarEltoNegritaTabla ("Estado:", 2) ;
    sprintf (cadena, "<FONT COLOR='red'>%s</FONT>",  estado)  ;
    salida.InsertarEltoTabla (cadena) ;
    salida.FinFilaTabla () ;
    salida.CerrarTabla () ;
    salida.InsertarLineaHoriz () ;

      //!!
      RecortarBlancos (doc);
      //!!

      cout << "<PRE>" << doc << "</PRE>" << endl ;

      //!!
      cout << "<B>Comentario:</B>";
      cout << "<PRE>" << com << "</PRE>" << endl;
      //!!

    salida.InsertarLineaHoriz () ;
  }
  else error = 1 ;
}
else {
  if (!strcmp (user, "CO")) {
  if (!colistadetIni (contrata, cond_estado))
    while (!colistadetSig (doc, telef, estado, com, marca)) { //!!
```

```
            marcada = (!strcmp (user, USER_CO)) && (strcmp (marca,
"NO_MARCADO") != 0);

            salida.AbrirTabla () ;

            // APV: Para marcadas (sin utilizar clase) revisar !!
            salida.InicioFilaTabla () ;

            strcpy (cadena2, "");
            if (marcada)
              sprintf (cadena2, "<IMG SRC=\"%s\" ALIGN=LEFT>\n",
GIF_MARCA);
            sprintf (cadena, "%s<B>Aver&iacute;a de
tel&eacute;fono:</B>", cadena2);
            salida.InsertarTextoNeutro ("<TD ALIGN=RIGHT
VALIGN=TOP>\n");
            salida.InsertarTextoNeutro (cadena);
            salida.InsertarTextoNeutro ("</TD>");
            salida.InsertarTextoNeutro ("<TD VALIGN=TOP>\n");
          sprintf (cadena,   "<FONT COLOR='red'>%s</FONT>", telef) ;
            salida.InsertarTextoNeutro (cadena);
            salida.InsertarTextoNeutro ("</TD>");
            salida.FinFilaTabla () ;
            if (marcada)
            {
              sprintf (cadena, "<BR><B>Marca:</B> %s\n"
                               "<A HREF=\"SSS\">(Quitar marca)</A>",
marca);
              salida.InicioFilaTabla ();
            salida.InsertarEltoNegritaTabla ("Marca:", 2);
              sprintf (cadena, "<FONT COLOR='red'>%s</FONT>", marca) ;
              salida.InsertarEltoTabla (cadena) ;
            salida.FinFilaTabla ();
            }
          salida.InicioFilaTabla () ;
          salida.InsertarEltoNegritaTabla ("Estado:", 2) ;
          sprintf (cadena, "<FONT COLOR='red'>%s</FONT>", estado) ;
          salida.InsertarEltoTabla (cadena) ;
          salida.FinFilaTabla () ;
          salida.CerrarTabla () ;
          salida.InsertarLineaHoriz () ;

            //!!
            RecortarBlancos (doc);
            //!!
            cout << "<PRE>" << doc << "</PRE>" << endl ;
            //!!
            cout << "<B>Comentario:</B>";
            cout << "<PRE>" << com << "</PRE>" << endl;
            //!!
            salida.InsertarLineaHoriz();
        }
      else error = 1 ;
      }

      else
      if (!prlistadetIni (ptoReunion, planif, encur, aver, os, grupo))
        while (!prlistadetSig (doc, act, telef, estado, grupoS, com))
{ //!!
```

```
          /* Parche APV */
          if (!strcmp (estado, "Pendiente")) {
            contPend ++;
            if (contPend > webConf.prN)
              continue;
          };

        salida.AbrirTabla () ;
        salida.InicioFilaTabla () ;
        if (act[0] == 'O')
          sprintf (cadena, "Orden de servicio n&uacute;mero:") ;
        else
          sprintf (cadena, "Aver&iacute;a de tel&eacute;fono:") ;
        salida.InsertarEltoNegritaTabla (cadena, 2) ;
        sprintf (cadena,   "<FONT COLOR='red'>%s</FONT>",  telef) ;
        salida.InsertarEltoTabla (cadena) ;
        salida.FinFilaTabla () ;
        salida.InicioFilaTabla () ;
        salida.InsertarEltoNegritaTabla ("Estado:", 2) ;
        sprintf (cadena, "<FONT COLOR='red'>%s</FONT>", estado) ;
        salida.InsertarEltoTabla (cadena) ;
        salida.FinFilaTabla () ;
        salida.InicioFilaTabla () ;
        salida.InsertarEltoNegritaTabla ("Realizador:", 2) ;
        sprintf (cadena, "<FONT COLOR='red'>%s</FONT>", grupoS) ;
        salida.InsertarEltoTabla (cadena) ;
        salida.FinFilaTabla () ;
        salida.CerrarTabla () ;
        salida.InsertarLineaHoriz () ;

          //!!
          RecortarBlancos (doc);
          //!!
          cout << "<PRE>" << doc << "</PRE>" << endl ;
          //!!
          cout << "<B>Comentario:</B>";
          cout << "<PRE>" << com << "</PRE>" << endl;
          //!!

        salida.InsertarLineaHoriz () ;
        }
      else error = 1 ;
    }

    if (error) {
      char *menError ;
      menError = errorSQL () ;
      salida.InsertarTexto (menError) ;
    }
  }
  else {
    salida.InsertarTexto ("Usuario no autorizado") ;
  }
}

// Funcion principal para browser Nokia 9110
int main9110  ()
{
```

```
char tecnico[TAM_TECNICO+1],
     ptoReunion[TAM_PUNTO_REUNION+1],
     contrata[TAM_CONTRATA+1],
     cadena[100],
     doc[TAM_DOC+1],
     //!!
     com[TAM_COM+1],
     //!!
     act[TAM_ACTUACION+1],
     telef[TAM_TELEFONO+1],
     estado[TAM_ESTADO+1],
     grupoS[TAM_GRUPO+1],
     n_os[TAM_N_OS+1],
     grpo[TAM_GRUPO+1],
     tit[TAM_TIT+1],
     domic[TAM_DOMIC+1],
     poblac[TAM_POBLAC+1],
     provinc[TAM_PROV+1],
     efect[TAM_EFECT+1],
     cl_os[TAM_CLAS_OS+1],
     ent[TAM_ENTIDAD+1],
     n_solic[TAM_N_SOLIC+1],
     clas_abon[TAM_CLAS_ABON+1],
     tipo_serv[TAM_TIP_SERV+1],
     procedenc[TAM_PROCEDENC+1],
     tipo_trab[TAM_TIP_TRAB+1],
     clav_esp[TAM_CLAV_ESP+1],
     observac[TAM_OBSERVAC+1],
     marca[TAM_DESC_MARCA+1],
     // averías
     tip_client[TAM_TIP_CLIENT+1],
     cent[TAM_CENT+1],
     sint[TAM_SINTOMA+1],
     tip_av[TAM_TIP_AVIS+1],
     a_plant[TAM_AREA_PLANTA+1],
     grul[TAM_GRUPOL+1],
     p_a_r[TAM_PAR+1],
     denom[TAM_DENOMINAC+1],
     p_ent[TAM_PAR_ENTRADA+1],
     p_sal[TAM_PAR_SALIDA+1],
     c_term[TAM_CAJA_TERM+1],
     res_prueb[TAM_RESULT_PRUEB+1],
     ob_mec[TAM_OBSERVAC_MEC+1],
     sit_c_term[TAM_SITUAC_CAJA_TERM+1],
     sit_c_conex[TAM_SITUAC_CAJA_CONEX+1],
     asig_de[TAM_ASIGNAC_DE+1],
     *todosGrupos,
     *cadEnCur = NULL,
     *cadPlanif = NULL,
     *cadAver = NULL,
     *cadOs = NULL,
     *grupo = NULL,
     *user = NULL,
     *var = NULL,
     *temaAyuda ;
int  planif = 1,
     encur = 1,
     aver = 1,
```

```
     os = 1;
long c_p,
     fh_recl,
     fh_creac,
     fh_form;

char **eqser;
int  neqser;
struct datosEquiposOS *eqos;
int  neqos;
struct datosLineasOS *lin_os;
int  nlineas;

Entrada *entrada ;
Parametros *parametros ;
Enlace     *enlace ;
struct tm *tiempo ;

char *cond_todosEstados = NULL,
     *cond_estado = NULL;
int contPend = 0;
int tu;

var = getenv ("QUERY_STRING") ;
if (!var) {
  entrada = new Entrada (&cin) ;
  user = entrada->DameValor (CAMPO_USER) ;
  todosGrupos = entrada->DameValor (CAMPO_TODOS_GRUPOS) ;
  cadEnCur = entrada->DameValor (CAMPO_EN_CURSO) ;
  cadPlanif = entrada->DameValor (CAMPO_PLANIFICADA) ;
  cadAver = entrada->DameValor (CAMPO_AVERIA) ;
  cadOs = entrada->DameValor (CAMPO_INSTALACIONES) ;
  grupo       = entrada->DameValor (CAMPO_GRUPOS) ;

  /* APV:V1.2.7:18/05/99 */
  cond_todosEstados = entrada->DameValor (CAMPO_CO_TODEST) ;
  cond_estado = entrada->DameValor (CAMPO_CO_ESTADOS) ;
  }
else {
  parametros = new Parametros (var) ;
  user = parametros->DameValor (CAMPO_USER) ;
  todosGrupos = parametros->DameValor (CAMPO_TODOS_GRUPOS) ;
  cadEnCur = parametros->DameValor (CAMPO_EN_CURSO) ;
  cadPlanif = parametros->DameValor (CAMPO_PLANIFICADA) ;
  cadAver = parametros->DameValor (CAMPO_AVERIA) ;
  cadOs = parametros->DameValor (CAMPO_INSTALACIONES) ;
  grupo       = parametros->DameValor (CAMPO_GRUPOS) ;

  /* APV:V1.2.7:18/05/99 */
  cond_todosEstados = parametros->DameValor (CAMPO_CO_TODEST) ;
  cond_estado = parametros->DameValor (CAMPO_CO_ESTADOS) ;
  }
if (todosGrupos) {
  if (!strcmp (todosGrupos, "s")) grupo = NULL ;
  if (!cadEnCur ) encur = 0 ;
  if (!cadPlanif ) planif = 0 ;
  if (!cadAver ) aver = 0 ;
  if (!cadOs ) os = 0 ;
  }
```

```
  /* APV:V1.2.7:18/05/99 */
  if (cond_todosEstados)
    if (!strcmp (cond_todosEstados, "s")) cond_estado = NULL;

  tu = infoLDAP(tecnico, ptoReunion, contrata);

  // Solo si tiene perfil de tecnico
  if (tu == LDAP_TECNICO || tu == LDAP_TECPR || tu == LDAP_TECCO) {

    if (!teclistadetIni (tecnico)) {

      // Para evitar el uso de cache en los cgi
      cout << "Pragma: no-cache" << endl;
      Salida salida (&cout) ;
      salida.CrearCabecera (TITULO9110) ;
      strcpy (com, "");
      salida.InsertarNuevaLinea();
      sprintf (cadena, "<B>Técnico: </B>%s",tecnico) ;
      salida.InsertarTextoNeutro (cadena);
      salida.InsertarTextoNeutro ("<CENTER>") ;
      enlace = new Enlace (CGI_TEC_ACT) ;
      enlace->InsertarCampo (CAMPO_USER, "TEC") ;
      enlace->CodificaURL () ;
      enlace->InsertarCampoFin () ;
      salida.InsertarEnlace (enlace->DameCadena(), "Atrás");
      delete enlace ;
      salida.InsertarTextoNeutro ("</CENTER>") ;
      salida.InsertarLineaHoriz() ;
      contPend = 0;
      while (!teclistadetSig (doc, act, telef, estado, com)) { //!!

        /* Parche APV */
        if (!strcmp (estado, "Pendiente")) {
          contPend ++;
          if (contPend > webConf.tecN)
            continue;
          };

      int flag_contrata = 0;
      flag_contrata = !strcmp (user, USER_CO);
      if (act[0] == 'O')
             detalle9110os(act, doc, estado, n_os, grpo, com, tit,
domic, poblac,
                           provinc, efect,cent, cl_os, &fh_creac,
&fh_form, ent, n_solic,
                           clas_abon, tipo_serv, procedenc, tipo_trab,
clav_esp,
                           observac, &eqos, &neqos, &lin_os, &nlineas,
                           marca, flag_contrata); /*!!*/
        else
             detalle9110av (act, doc, estado, telef, grpo, com, tit,
domic, poblac,
                            provinc, tip_client, cent, sint, &c_p,
&fh_recl, tip_av,
                            a_plant, grul, p_a_r, denom, p_ent, p_sal,
c_term, res_prueb,
                            ob_mec, sit_c_term,sit_c_conex, asig_de,
&eqser, &neqser,
```

```
                            marca, flag_contrata); /*!!*/
     if (act[0] == 'O') {
       sprintf (cadena, "<B>Orden de servicio n&uacute;mero:
</B>%s<BR>",n_os ) ;
       salida.InsertarTextoNeutro ("<FONT SIZE='+1'>") ;
       salida.InsertarTextoNeutro (cadena) ;
       sprintf (cadena, "<B>Estado: </B>%s<BR>",estado) ;
       salida.InsertarTextoNeutro (cadena) ;
       salida.InsertarTextoNeutro ("</FONT>") ;
       }
     else {
       sprintf (cadena,"<B>Aver&iacute;a de tel&eacute;fono:
</B>%s<BR>",telef) ;
       salida.InsertarTextoNeutro ("<FONT SIZE='+1'>") ;
       salida.InsertarTextoNeutro (cadena) ;
       sprintf (cadena, "<B>Estado: </B>%s<BR>",estado) ;
       salida.InsertarTextoNeutro (cadena) ;
       salida.InsertarTextoNeutro ("</FONT>") ;
     }
     // salida.InsertarLineaHoriz () ;
     // Datos de Averias para 9110
     if (act[0] == 'A') {
       salida.InsertarTextoNeutro("<PRE><FONT SIZE=+1>");
       sprintf (cadena,"<B>Titul:</B>%s\n",tit);
       salida.InsertarTextoNeutro(cadena);
       RecortarBlancosIntermedios (domic);
       sprintf (cadena,"<B>Domic:</B>%s\n",domic);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Pobla:</B>%s (%s)\n",poblac,provinc);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>TiCli:</B>%s\n",tip_client);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Areap:</B>%s",a_plant);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"  <B>Centr:</B>%s\n",cent);
       salida.InsertarTextoNeutro(cadena);
       tiempo = localtime (&fh_recl) ;
       sprintf (cadena,"<B>FeRec:</B>%02d/%02d/%4d %02d:%02d",
                    tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                    tiempo->tm_hour, tiempo->tm_min) ;
       salida.InsertarTextoNeutro (cadena);
       if (c_p > 0)
       {
         tiempo = localtime (&c_p);
         sprintf (cadena,"  <B>Citap:</B>%02d/%02d/%4d %02d:%02d",
                  tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                  tiempo->tm_hour, tiempo->tm_min);
         salida.InsertarTextoNeutro (cadena) ;
       }
       salida.InsertarTextoNeutro ("\n") ;
       sprintf (cadena,"<B>  Grupo  Par  Denomi Paren Parsa
Cajter</B>\n");
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena, "  %s %s %s %s %s %s \n",
                grul, p_a_r, denom, p_ent, p_sal, c_term);
       salida.InsertarTextoNeutro(cadena);
       RecortarBlancosIntermedios (sit_c_term);
```

```
       sprintf (cadena,"<B>SitCT:</B>%s\n",sit_c_term);
       salida.InsertarTextoNeutro(cadena);
       RecortarBlancosIntermedios (sit_c_conex);
       sprintf (cadena,"<B>SitCC:</B>%s\n",sit_c_conex);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Asign:</B>%s\n",asig_de);
       salida.InsertarTextoNeutro(cadena);
       for (int j=0; j<neqser; j++)
       {
         sprintf (cadena,"<B> Eq%2d:</B>%s\n", j+1, eqser[j]);
         salida.InsertarTextoNeutro(cadena);
       };
       sprintf (cadena,"<B>RePru:</B>%s\n",res_prueb);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Sínto:</B>%s\n",sint);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Aviso:</B>%s\n",tip_av);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Obser:</B>%s\n",ob_mec);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Comen:</B>%s\n",com);
       salida.InsertarTextoNeutro(cadena);
       salida.InsertarTextoNeutro("</FONT></PRE>");
     }
     else { // Caso de ordenes de servicio
       salida.InsertarTextoNeutro("<PRE><FONT SIZE='+1'>");
       sprintf (cadena,"<B>Titul:</B>%s\n",tit);
       salida.InsertarTextoNeutro(cadena);
       RecortarBlancosIntermedios (domic);
       sprintf (cadena,"<B>Domic:</B>%s\n",domic);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Pobla:</B>%s (%s)\n",poblac,provinc);
       salida.InsertarTextoNeutro(cadena);
       tiempo = localtime (&fh_creac);
       sprintf (cadena,"<B>FeCre:</B>%02d/%02d/%4d %02d:%02d",
                tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                tiempo->tm_hour, tiempo->tm_min);
       salida.InsertarTextoNeutro (cadena);
       tiempo = localtime (&fh_form) ;
       sprintf (cadena,"<B>  FeFor:</B>%02d/%02d/%4d %02d:%02d\n",
                   tiempo->tm_mday, tiempo->tm_mon+1, tiempo-
>tm_year+1900,
                   tiempo->tm_hour, tiempo->tm_min) ;
       salida.InsertarTextoNeutro (cadena) ;
       sprintf (cadena,"<B>ClaOS:</B>%s\n",cl_os);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Centr:</B>%s",cent);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"  <B>Entid:</B>%s",ent);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"  <B>NuSol:</B>%s\n",n_solic);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>ClAbo:</B>%s\n",clas_abon);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Traba:</B>%s/%s/\n      %s\n",tipo_serv,
procedenc, tipo_trab);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>ClEsp:</B>%s\n",clav_esp);
```

```
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Obser:</B>%s\n",observac);
       salida.InsertarTextoNeutro(cadena);
       sprintf (cadena,"<B>Comen:</B>%s\n",com);
       salida.InsertarTextoNeutro(cadena);
       salida.InsertarTextoNeutro("<B>DATOS EQUIPOS:</B>\n");
       for (int j=0; j<neqos; j++)
       {
         sprintf (cadena,"<B> Eq%2d:</B>\n",j+1);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena,"   <B>Ope:</B>%s <B>Uni:</B>%s
<B>Prol:</B>%s\n",
                  eqos[j].operac, eqos[j].unidades,eqos[j].prolong);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena,"   <B>Equ:</B>%s\n", eqos[j].equipo);
         salida.InsertarTextoNeutro(cadena);
       };
       salida.InsertarTextoNeutro("<B>DATOS LINEAS:</B>\n");
       for (int k=0; k<nlineas; k++)
       {
         sprintf (cadena," <B>Li%2d:</B>\n",k+1);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena,"   <B>Ope:</B>%s",lin_os[k].operac2);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena," <B>Lin:</B>%s",lin_os[k].linea);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena," <B>Cen:</B>%s\n",lin_os[k].central);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena,"   <B>Tel:</B>%s",lin_os[k].telefono_linea);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena," <B>Asi:</B>%s\n",lin_os[k].asign);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena,"   <B>Gru:</B>%s",lin_os[k].grupo);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena," <B>Par:</B>%s",lin_os[k].par);
         salida.InsertarTextoNeutro(cadena);
         sprintf (cadena," <B>Caj:</B>%s\n",lin_os[k].caja);
         salida.InsertarTextoNeutro(cadena);
         RecortarBlancosIntermedios (lin_os[k].domicilio);
         sprintf (cadena,"   <B>Dom:</B>%s\n",lin_os[k].domicilio);
         salida.InsertarTextoNeutro(cadena);
       };
     salida.InsertarTextoNeutro("</FONT></PRE>");
     } // Fin de if os
    salida.InsertarLineaHoriz () ;
    } //while
   }
   else return error9110(ERR_ORACLE);
  }
  else return error9110(ERR_PERFIL_USUARIO);

  return 0;
}


int error9110 (int codError) //Error con 9110
{
  char cadena[200];
  Enlace *enlace ;
```

```
  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  Salida salida (&cout) ;
  salida.CrearCabecera (TIT9110ERROR);
  salida.InsertarTextoNeutro ("<P> <FONT SIZE=+1>");
  salida.InsertarTextoNeutro ("<I>Se ha producido un error al obtener
el detalle de las actuaciones:</I>\n");
  salida.InsertarTextoNeutro ("<UL>\n");

  if (codError == ERR_VAR_REMOTEUSER)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>REMOTE_USER</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible control de acceso no
activado para el servidor WEB.\n");
  };

  if (codError == ERR_VAR_SERVERURL)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>SERVER_URL</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible ejecucion del cgi manual
fuera del entorno WEB.\n");
  };

  if (codError == ERR_PERFIL_USUARIO)
  {
    sprintf (cadena, "<LI>Usuario <B>%s</B> no autorizado.\n", getenv
("REMOTE_USER"));
    salida.InsertarTextoNeutro (cadena);
    salida.InsertarTextoNeutro ("<LI>Posible usuario sin perfil de
técnico o sin numero de matrícula.\n");
    salida.InsertarTextoNeutro ("<LI>Posible Servidor LDAP parado o no
accesible.\n");
  }

  if (codError == ERR_CONFIGURACION)
  {
    salida.InsertarTextoNeutro ("<LI>Fichero de <B>configuración</B>
no accesible o con formato incorrecto.\n");
  }

  if (codError == ERR_ORACLE)
  {
    salida.InsertarTextoNeutro ("<LI>Error de acceso a la base de
datos.\n");
    sprintf (cadena, "<LI>Mensaje Oracle: <B>%s</B>\n", errorSQL ());
    salida.InsertarTextoNeutro (cadena);
  };

  salida.InsertarTextoNeutro ("<CENTER><P>");

  // Enlace Continuar
  enlace = new Enlace (CGI_IND9110) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Continuar");
```

```
  salida.InsertarTextoNeutro ("</CENTER>");
  return codError;
}
```

franqueo1.cgi

```
/* <MESA:01:@(#):MettnadJnj90:ga2:1.6:990819084438:etna:1 34
41644:MESA> */
//  Alejandro Secades Gomez
//  franqueo1.cgi

#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "CopiaHtml.h"
#include "parametros.h"
#include "defines.h"
#include <enlace.h>
#include <salida.h>

extern "C" {
#include "webconf.h"
#include "auxsql.h"
#include "auxldap.h"
#include "auxars.h"     //9110
}

#define TEC 0
#define PR 1
#define CO 2
#define HTMLFRANQTEC    "../franqueo.html"
#define HTMLFRANQCO     "../franqueoco.html"
//!!
#define MANUAL          'B'
#define NORMAL          'A'
#define CAT_MANUAL      "9"
//!!
#define TITULO9110 "GA-Franqueo"                          //9110
#define TIT9110ERROR "WEB-GA: ERROR DE FRANQUEO"          //9110
#define ERR_VAR_REMOTEUSER           -1
#define ERR_VAR_SERVERURL            -2
#define ERR_PERFIL_USUARIO           -3
#define ERR_CONFIGURACION            -4
#define ERR_ORACLE                   -5
#define ERR_EQLIN_NOTFOUND           -6

inline void error() {
  cout << "Location: " << getenv("SERVER_URL") << "/errorfranq.html?"
       << "1: " << errorSQL() << "\n\n";
  exit(0);
}

inline void titulo(char *t, int negrita=0) {
  if(negrita) cout << "<B>";
  cout << t;
  if(negrita) cout << "</B>";
}
```

```
#define vacio(C,n) sal.readonly(C,"",n);
int tipousuario(char *user, char *mat);
// Prototipo de funcion principal para browser Nokia 9110
int main9110 (); //9110
int tipousuario9110(char *user, char *mat); // 9110
int error9110 (int codigoerror); //Error con 9110

int main()
{
  char *av, *user, **mats, mat[20], **eqlin;
  int nm, neq,
    tipou;  // 0 tec, 1 pr, 2 con
  long *cats;

  // ??
  Enlace *enlace;

  //!!
//La variable manual se utiliza para distinguir cuando la seleccion de
//un equipo/servicio se realiza de manera manual o a traves de una
//lista
  // desplegable
  int manual=0;
  char miav[100];
  //!!

  char *remoteuser; //9110
  char *serverurl;  //9110

  // Codigo 9110
  remoteuser = getenv ("REMOTE_USER");
  if (remoteuser == NULL)
    return error9110 (ERR_VAR_REMOTEUSER);

  serverurl = getenv ("SERVER_URL");
  if (serverurl == NULL)
    return error9110 (ERR_VAR_SERVERURL);

  // Primero se lee el fichero de configuracion
  if (leewebconf())
    return error9110 (ERR_CONFIGURACION);

  // Si no es un usuario 9110 camino normal
  if (strstr (remoteuser, "9110"))
    return main9110 ();
    printf("Location: %s%s\n\n", serverurl, PAG_INDICE);
  // Fin de codigo 9110

  Parametros par(getenv("QUERY_STRING"));

  // recoger parametros
  av=par.DameValor(CAMPO_ACT);
  user=par.DameValor(CAMPO_USER);

  //!!
  // Para distinguir el acceso al cgi de manera manual se utiliza (de
manera
  // provisional) el primer caracter del entry-id, de modo que tendra
'A' si
```

```
  // si accede con la lista y 'B' si se accede de manera manual
  if (av[0] == MANUAL)
  {
    manual = 1;
    av[0] = NORMAL; //Se vuelve a dejar bien para funcionamiento
correcto
  }
  //!!
  // distinguir tipo user
  tipou=tipousuario(user,mat);
  // consultas
  if(conecta()) error();
  if((neq=lfranqeqser(av,&eqlin,&cats))<=0) error();
  if(tipou==PR)
    if((nm=lprmatgrupo(av,&mats))<=0) error();
  if(tipou==CO)
    if(cofranq(av,mat)) error();

  // generar salida
  CopiaHtml sal((tipou==CO)?HTMLFRANQCO:HTMLFRANQTEC);
  sal.noreadonly(CAMPO_EQLIN);
  sal.copia();          // javascript
  cout << "var categorias=new makeArray(" << neq+1 << ");\n";
  cout << "categorias[1]=1;\n";
  for(int i=0;i<neq;i++)
  cout << "categorias[" << (i+2) << "]=" << cats[i] << ";\n";
  sal.copia();          // ayuda
  switch(tipou) {
  case CO:  cout << AYUDA_FRANQUEO_CON; break;
  case TEC: cout << AYUDA_FRANQUEO_TEC; break;
  case PR:  cout << AYUDA_FRANQUEO_PR; break;
  }
  sal.copia();          // destino del submit
  cout << "franqueo2.cgi";
  sal.copia();
  //!!
  // De manera provisional se utiliza un valor de categoria especial
("9") para
  // asociado a un equipo introducido de manera manual para que pueda
distinguirlo
  // el cgi del paso posterior de franqueo
  if (manual)
  {
     sal.input(CAMPO_CATEGORIA,CAT_MANUAL, 5, "HIDDEN");
     sal.input(CAMPO_MAN,CAT_MANUAL, 5, "HIDDEN");
  }
  else
  {
  //!!
  sal.input(CAMPO_CATEGORIA,"V",5,"HIDDEN");
  // ??
  sal.input(CAMPO_MAN,"V",5,"HIDDEN");
  // ??
  }
  sal.input(CAMPO_ACT,av,15,"HIDDEN");
  sal.input(CAMPO_USER,user,3,"HIDDEN");
  sal.copia();
  if(tipou!=PR) cout << "Seleccione el <B>Equipo/l&iacute;nea:</B>";
```

```
  else cout << "Seleccione el <B>Eq./l&iacute;nea y
matr&iacute;cula:</B>";
  // ??
  sal.copia();
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  cout << enlace->DameCadena ();
  // ??
  sal.copia();          // matricula
  titulo((tipou==CO)?"Nº de contrata:":"Nº
matr&iacute;cula:",(tipou==PR)?1:0);
  sal.copia();
  if(tipou==PR)
    sal.seleccion(CAMPO_MAT,mats,nm);
  else
    sal.readonly(CAMPO_MAT,mat,10);
  /* if(tipou!=CO) { */
    sal.copia();        // nmf
    titulo("N&uacute;m. modelo fact:");
    sal.copia();
    vacio(CAMPO_NMF,7);
  /* }  */
  // ???
  if(tipou==CO) {
    sal.copia();
    titulo("Fecha-Hora Franqueo:");
    sal.copia();
    vacio(CAMPO_FHF,16);
  }
  // ??
  sal.copia();          //eq/linea
  titulo("Equipo/L&iacute;nea:  ",1);
  //!!
  if (!manual)
  {
    // Se inserta enlace para acceso al cgi manual
    strcpy (miav, av);
    miav[0]=MANUAL;
    cout << "<A HREF=" << CGI_FRANQUEAR1
         << "?ACT=" << miav
         << "?USER=" << user
         << ">(Manual)"
         <<  "</A>" ;
  }
  else
    cout << "<A HREF=" << CGI_FRANQUEAR1
         << "?ACT=" << av
         << "?USER=" << user
         << ">(Lista)"
         << "</A>" ;
  //!!

  sal.copia();
  //!!
  if (manual)
     sal.input(CAMPO_EQLIN,"Editar equipo/linea",30,"TEXT","");
```

```
  else
  //!!
  //??
  sal.seleccion(CAMPO_EQLIN,"envia(2)",eqlin,neq);
  sal.copia();          // eq inst
  titulo("Equipo Instalado:");
  sal.copia();
  vacio(CAMPO_EQINS,18);
  sal.copia();
  titulo("Clase de Aver&iacute;a:");
  sal.copia();
  vacio(CAMPO_CLASEAV,18);
  sal.copia();
  titulo("Nombre de Aver&iacute;a:");
  sal.copia();
  vacio(CAMPO_NOMBREAV,18);

  if(tipou!=CO) {
    sal.copia();        // horas
    titulo("Horas:");
    sal.copia();
    vacio(CAMPO_DHORAS,4);
    sal.copia();        // minutos
    titulo("Minutos:");
    sal.copia();
    vacio(CAMPO_DMIN,4);
    sal.copia();        // kilometros
    titulo("Kil&oacute;metros:");
    sal.copia();
    vacio(CAMPO_DKM,5);
  }

  sal.copia();          //resto
  return 0;
}

int tipousuario(char *user, char *mat)
{
  int r;
  char aux1[20], aux2[20], m[20];
  if(!strcmp(user,USER_TEC)) {
    r=TEC;
    infoLDAP(mat,aux1,aux2);
  }
  else if(!strcmp(user,USER_CO))
    r=CO;
  else if(!strcmp(user,USER_PR))
    r=PR;
  else error();
  return r;
}



// Funcion principal para browser Nokia 9110
int main9110()
{
  char *av, *user, **mats, mat[20], **eqlin;
```

```
  int nm, neq,
    tipou;  // 0 tec, 1 pr, 2 con
  long *cats;
  char cadena[200];
  char ptoReunion[100], contrata[100];
  int tu;
  // ??
  Enlace *enlace;
  //!!
  // La variable manual se utiliza para distinguir cuando la seleccion
de
  // un equipo/servicio se realiza de manera manual o a traves de una
lista
  // desplegable
  int manual=0;
  char miav[100];

  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  //!!
  Parametros par(strdup(getenv("QUERY_STRING"))); // recoger
parametros
  av=par.DameValor(CAMPO_ACT);
  user=par.DameValor(CAMPO_USER);

  //!!
  // Para distinguir el acceso al cgi de manera manual se utiliza (de
manera
  // provisional) el primer caracter del entry-id, de modo que tendra
'A' si
  // si accede con la lista y 'B' si se accede de manera manual

  if (av[0] == MANUAL)
    {
      manual = 1;
      av[0] = NORMAL; //Se vuelve a dejar bien para funcionamiento
correcto
    }
  //!!
  // distinguir tipo user
  tu = infoLDAP(mat, ptoReunion, contrata);

  // Error si no tiene perfil de tecnico
  if (tu != LDAP_TECNICO && tu != LDAP_TECPR && tu != LDAP_TECCO)
     return error9110 (ERR_PERFIL_USUARIO);

  if (strcmp(user,USER_TEC) != 0)
     return error9110 (ERR_PERFIL_USUARIO);

  tipou=TEC;

  // consultas
  if(conecta()) return error9110(ERR_ORACLE);
  neq=lfranqeqser(av,&eqlin,&cats);
  if (neq < 0) return error9110(ERR_ORACLE);
  if (neq == 0) return error9110(ERR_EQLIN_NOTFOUND);

  // generar salida
```

```
  Salida salida (&cout) ;
  salida.AbrirForm ("/cgi-bin/franqueo2.cgi");
  salida.CrearCabecera(TITULO9110) ;
  salida.InsertarTextoNeutro ("<CENTER>");

  // Enlace Cancelar
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Cancelar");
  salida.InsertarTextoNeutro ("</CENTER>");
  sprintf (cadena," Nº matr: <I>%s</I><BR>", mat);
  salida.InsertarTextoNeutro(cadena);
  salida.InsertarTextoNeutro(" <B>Equ/Lín: </B>");
  if (manual) salida.InsertarEntradaForm (CAMPO_EQLIN, "Editar
equipo/linea", 30);
  else salida.InsertarSelectForm ( CAMPO_EQLIN, neq, eqlin, eqlin,
eqlin[0] );
  salida.InsertarNuevaLinea();
  if (!manual)
    { // Se inserta enlace para acceso al cgi manual
      strcpy (miav, av);
      miav[0]=MANUAL;
      cout << "  <A HREF=" << CGI_FRANQUEAR1 << "?ACT=" <<
miav << "?USER=" << user << ">(Selección Manual)" <<  "</A>" ;
    }
  else cout << "  <A HREF=" << CGI_FRANQUEAR1 << "?ACT=" <<
av << "?USER=" << user << ">(Selección Lista)" << "</A>";


  //!!
  // De manera provisional se utiliza un valor de categoria especial
("9") para
  // asociado a un equipo introducido de manera manual para que pueda
distinguirlo
  // el cgi del paso posterior de franqueo

  if (manual)
    {
      salida.InsertarCampoOcultoForm (CAMPO_CATEGORIA, CAT_MANUAL);
      salida.InsertarCampoOcultoForm (CAMPO_MAN, CAT_MANUAL);
    }
  else {
     salida.InsertarCampoOcultoForm (CAMPO_CATEGORIA, "V");
     salida.InsertarCampoOcultoForm (CAMPO_MAN, "V");
    }
  salida.InsertarCampoOcultoForm (CAMPO_ACT, av);
  salida.InsertarCampoOcultoForm (CAMPO_USER, user);
  // Campo numero de matricula  solo lectura (se mantiene campo oculto
por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)
  salida.InsertarCampoOcultoForm (CAMPO_MAT, mat);

  // Numero de modelo facturable oculto por si acaso
  salida.InsertarCampoOcultoForm (CAMPO_NMF, "");
  salida.InsertarCampoOcultoForm (CAMPO_EQINS, "");
```

```
  salida.InsertarCampoOcultoForm (CAMPO_CLASEAV, "");
  salida.InsertarCampoOcultoForm (CAMPO_NOMBREAV, "");
  salida.InsertarCampoOcultoForm (CAMPO_DHORAS, "");
  salida.InsertarCampoOcultoForm (CAMPO_DMIN, "");
  salida.InsertarCampoOcultoForm (CAMPO_DKM, "");
  salida.InsertarTextoNeutro ("<CENTER>");
  salida.InsertarBotonSubmitForm ("Siguiente");
  salida.InsertarTextoNeutro ("<CENTER>");
  salida.CerrarForm();
  return 0;
}

int error9110 (int codError) //Error con 9110
{
  char cadena[200];
  Enlace *enlace ;
  char *av;
  char *user;

  Salida salida (&cout) ;
  salida.CrearCabecera (TIT9110ERROR);
  salida.InsertarTextoNeutro ("<P> <FONT SIZE=+1>");
  salida.InsertarTextoNeutro ("<I>Se ha producido un error en el
primer paso de franqueo:</I>\n");
  salida.InsertarTextoNeutro ("<UL>\n");

  if (codError == ERR_VAR_REMOTEUSER)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>REMOTE_USER</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible control de acceso no
activado para el servidor WEB.\n");
  };
  if (codError == ERR_VAR_SERVERURL)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>SERVER_URL</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible ejecucion del cgi manual
fuera del entorno WEB.\n");
  };
  if (codError == ERR_PERFIL_USUARIO)
  {
    sprintf (cadena, "<LI>Usuario <B>%s</B> no autorizado.\n", getenv
("REMOTE_USER"));
    salida.InsertarTextoNeutro (cadena);
    salida.InsertarTextoNeutro ("<LI>Posible usuario sin perfil de
técnico o sin numero de matrícula.\n");
    salida.InsertarTextoNeutro ("<LI>Posible Servidor LDAP parado o no
accesible.\n");
  }
  if (codError == ERR_CONFIGURACION)
  {
    salida.InsertarTextoNeutro ("<LI>Fichero de <B>configuracion</B>
no accesible o con formato incorrecto.\n");
  }

  if (codError == ERR_ORACLE)
  {
```

```
    salida.InsertarTextoNeutro ("<LI>Error de acceso a la base de
datos.\n");
    sprintf (cadena, "<LI>Mensaje Oracle: <B>%s</B>\n", errorSQL ());
    salida.InsertarTextoNeutro (cadena);
  };

  if (codError == ERR_EQLIN_NOTFOUND)
  {
    salida.InsertarTextoNeutro ("<LI>No se encontraron equipos/lineas
asociados a la avería.\n");
    salida.InsertarTextoNeutro ("<LI>Posible avería sin equipos.");
    salida.InsertarTextoNeutro ("<LI>Posibles equipos de avería no
catalagodos en GA.");
  };
   salida.InsertarTextoNeutro ("<CENTER><P>");

  // strdup porque el constructor de parametros destroza la variable
de entorno
  Parametros par(strdup(getenv("QUERY_STRING")));

  // recoger parametros
  av=par.DameValor(CAMPO_ACT);
  user=par.DameValor(CAMPO_USER);
  // Enlace Continuar
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Continuar");
  salida.InsertarTextoNeutro ("</CENTER>");
  return codError;
}
```

franqueo2.cgi

```
/* <MESA:01:@(#):MettnadJnj9M:ga2:1.6:990819084440:etna:1 34
40620:MESA> */
//  Alejandro Secades Gomez
//  franqueo1.cgi

#include <stdlib.h>
#include <string.h>
#include "CopiaHtml.h"
#include "entrada.h"
#include "defines.h"
#include "enlace.h"
#include "salida.h" //9110

extern "C" {
#include "auxsql.h"
#include "auxldap.h"
}

#define TEC 0
#define PR 1
#define CO 2
```

```
#define HTMLFRANQTEC    "../franqueo.html"
#define HTMLFRANQCO     "../franqueoco.html"
#define CAT_MANUAL  '9'
#define MANUAL "9"
#define TITULO9110 "GA-Franqueo"                    //9110
#define TIT9110ERROR "WEB-GA: ERROR DE FRANQUEO"          //9110
#define ERR_VAR_REMOTEUSER           -1
#define ERR_VAR_SERVERURL            -2
#define ERR_PERFIL_USUARIO           -3
#define ERR_CONFIGURACION            -4
#define ERR_ORACLE                   -5
#define ERR_EQLIN_NOTFOUND           -6

inline void error(int i) {
  cout << "Location: " << getenv("SERVER_URL") << "/errorfranq.html?"
       << "2: " << i << '-' << errorSQL() << "\n\n";
  exit(0);
}

inline void titulo(char *t, int negrita=0) {
  if(negrita) cout << "<B>";
  cout << t;
  if(negrita) cout << "</B>";
}

#define vacio(C,n) sal.readonly(C,"",n);

int tipousuario(char *user);

// Prototipo de funcion principal para browser Nokia 9110
int main9110 (); //9110
int tipousuario9110(char *user); // 9110
int error9110 (int codigoerror); //Error con 9110

int main()
{
  char *av, *user, *cat, *mat, *eqlin, **claseav;
  // ??
  char **mats, **eqlins, *man;
  int nm, neq;
  long *cats;
  // ??
  int nclaseav, tipou;  // 0 tec, 1 pr, 2 con
  Entrada *par;
  Enlace *enlace;
  char *remoteuser; //9110
  char *serverurl;  //9110

  // Codigo 9110
  remoteuser = getenv ("REMOTE_USER");
  if (remoteuser == NULL)
    return error9110 (ERR_VAR_REMOTEUSER);



  serverurl = getenv ("SERVER_URL");
  if (serverurl == NULL)
    return error9110 (ERR_VAR_SERVERURL);
```

```
  // Primero se lee el fichero de configuracion
  if (leewebconf())
    return error9110 (ERR_CONFIGURACION);

  // Si no es un usuario 9110 camino normal
  if (strstr (remoteuser, "9110"))
    return main9110 ();
    printf("Location: %s%s\n\n", serverurl, PAG_INDICE);
  // Fin de codigo 9110

  // recoger parametros
  par=new Entrada(&cin);

  av=par->DameValor(CAMPO_ACT);
  user=par->DameValor(CAMPO_USER);
  man=par->DameValor(CAMPO_MAN);
  cat=par->DameValor(CAMPO_CATEGORIA);
  mat=par->DameValor(CAMPO_MAT);
  eqlin=par->DameValor(CAMPO_EQLIN);

  //!!
  int manual=0, nmanual=0;
  manual = (cat[0] == CAT_MANUAL);

  // En el caso de acceso manual, es necesario calcular la categoria
del
  // equipo introducido manualmente ya que no se calculo en cgi
anterior
  if (manual)
  {
    if((nmanual=conecta())!=0) error(nmanual);
    if((nmanual=catManual(eqlin,cat))!=0) error(nmanual);
  }

  // distinguir tipo user
  tipou=tipousuario(user);

  // consultas
  if((nclaseav=conecta())!=0) error(nclaseav);
  if((nclaseav=lfranqclav(atol(cat),&claseav))<=0) error(nclaseav);

  // ??
  if((neq=lfranqeqser(av,&eqlins,&cats))<=0) error(neq);
  if(tipou==PR)
    if((nm=lprmatgrupo(av,&mats))<=0) error(nm);
  // if(tipou==CO)
    // if(cofranq(av,mat)) error();
  // ??

  // generar salida
  CopiaHtml sal((tipou==CO)?HTMLFRANQCO:HTMLFRANQTEC);     // salida
html
  sal.noreadonly(CAMPO_CLASEAV);

  sal.copia();          // javascript, no hay nada
  // ??
  //cout << " ";
  cout << "var categorias=new makeArray(" << neq << ");\n";
  for(int i=0;i<neq;i++)
```

```
  cout << "categorias[" << (i+1) << "]=" << cats[i] << ";\n";
// ??

sal.copia();          // ayuda
switch(tipou) {
case CO:  cout << AYUDA_FRANQUEO_CON; break;
case TEC: cout << AYUDA_FRANQUEO_TEC; break;
case PR:  cout << AYUDA_FRANQUEO_PR; break;
}

sal.copia();          // action del form
cout << "franqueo3.cgi";

sal.copia();          // campos hidden
sal.input(CAMPO_CATEGORIA,cat,5,"HIDDEN");
if (manual)
  sal.input(CAMPO_MAN,MANUAL,5,"HIDDEN");
else
  sal.input(CAMPO_MAN,"V",5,"HIDDEN");
sal.input(CAMPO_ACT,av,15,"HIDDEN");
sal.input(CAMPO_USER,user,3,"HIDDEN");
sal.copia();
cout << "Seleccione la <B>clase de aver&iacute;a:</B>";

// ??
sal.copia();
enlace = new Enlace (CGI_DET_ACT) ;
enlace->InsertarCampo (CAMPO_ACT, av) ;
enlace->InsertarCampo (CAMPO_USER, user) ;
enlace->CodificaURL ();
enlace->InsertarCampoFin ();
cout << enlace->DameCadena ();
// ??

sal.copia();          // matricula
titulo((tipou==CO)?"Nº de contrata:":"Nº matr&iacute;cula:");
sal.copia();
// ??
// sal.readonly(CAMPO_MAT,mat,10);
if(tipou==PR)
  sal.seleccion(CAMPO_MAT,"envia(2);",mat,mats,nm);//mat);
else
  sal.readonly(CAMPO_MAT,mat,10);
// ??

/* if(tipou!=CO) { */
  sal.copia();        // nmf
  titulo("N&uacute;m. modelo fact:");
  sal.copia();
  vacio(CAMPO_NMF,7);
/* } */




// ???
if(tipou==CO) {
   sal.copia();
   titulo("Fecha-Hora Franqueo:");
```

```
     sal.copia();
     vacio(CAMPO_FHF,16);
  }
  // ???

  sal.copia();          //eq/linea
  titulo("Equipo/L&iacute;nea:");
  sal.copia();
  // ??
  // sal.readonly(CAMPO_EQLIN,eqlin,35);
  if(manual)
    sal.readonly(CAMPO_EQLIN,eqlin,35);
  else
    sal.seleccion(CAMPO_EQLIN,"envia(2)",eqlin,eqlins,neq);
  sal.copia();          // eq inst
  titulo("Equipo Instalado:");
  sal.copia();
  vacio(CAMPO_EQINS,25);
  sal.copia();
  titulo("Clase de Aver&iacute;a:",1);
  sal.copia();
  //??

sal.seleccion(CAMPO_CLASEAV,"javascript:envia(3);",claseav,nclaseav);

  sal.copia();
  titulo("Nombre de Aver&iacute;a:");
  sal.copia();
  vacio(CAMPO_NOMBREAV,25);

  if(tipou!=CO) {
    sal.copia();        // horas
    titulo("Horas:");
    sal.copia();
    vacio(CAMPO_DHORAS,4);

    sal.copia();        // minutos
    titulo("Minutos:");
    sal.copia();
    vacio(CAMPO_DMIN,4);

    sal.copia();        // kilometros
    titulo("Kil&oacute;metros:");
    sal.copia();
    vacio(CAMPO_DKM,5);
  }

  sal.copia();          //resto

  return 0;
}



int tipousuario(char *user)
{
  if(!strcmp(user,USER_TEC))
    return TEC;
```

```
  else if(!strcmp(user,USER_CO))
    return CO;
  else if(!strcmp(user,USER_PR))
    return PR;
  else error(-10001);
}

//Funcion principal para browser Nokia 9110
int main9110 ()
{
  char *av, *user, *cat, *mat, *eqlin, **claseav;
  // ??
  char **mats, **eqlins, *man;
  int nm, neq;
  long *cats;
  // ??
  int nclaseav, tipou;  // 0 tec, 1 pr, 2 con
  Entrada *par;
  Enlace *enlace;
  char cadena[200];

  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  // recoger parametros
  par=new Entrada(&cin);
  av=par->DameValor(CAMPO_ACT);
  user=par->DameValor(CAMPO_USER);
  man=par->DameValor(CAMPO_MAN);
  cat=par->DameValor(CAMPO_CATEGORIA);
  mat=par->DameValor(CAMPO_MAT);
  eqlin=par->DameValor(CAMPO_EQLIN);

  //!!
  int manual=0, nmanual=0;
  manual = (cat[0] == CAT_MANUAL);

  // Para distinguir el tipo de usuario
  tu = infoLDAP(mat, ptoReunion, contrata);

  // Error si no tiene perfil de tecnico
  if (tu != LDAP_TECNICO && tu != LDAP_TECPR && tu != LDAP_TECCO)
     return error9110 (ERR_PERFIL_USUARIO);

  if (strcmp(user,USER_TEC) != 0)
     return error9110 (ERR_PERFIL_USUARIO);

  tipou=TEC;
  // Fin de tratamiento del tipo de usuario

  // En el caso de acceso manual, es necesario calcular la categoria
// del
  // equipo introducido manualmente ya que no se calculo en cgi
//anterior

  if((nmanual=conecta())!=0) return error9110(ERR_ORACLE);
  nmanual = catManual(eqlin,cat))!=0) error9110(nmanual);
  if (nmanual == 1403) return error9110(ERR_EQLIN_NOTFOUND);
  if (nmanual != 0) return error9110(ERR_ORACLE);
```

```
  // consultas
  if((nclaseav=conecta())!=0) error9110(ERR_ORACLE);
  nclaseav = lfranqclav(atol(cat),&claseav))
  if (nclaseav == 0) return error9110(ERR_EQLIN_NOTFOUND);
  if (nclaseav != 0) return error9110
  if((nclaseav=lfranqclav(atol(cat),&claseav))<=0) error9110(1);

  // ??
  if((neq=lfranqeqser(av,&eqlins,&cats))<=0) error9110(1);
  if(tipou==PR)
  if((nm=lprmatgrupo(av,&mats))<=0) error9110(1);
  // if(tipou==CO)
    // if(cofranq(av,mat)) error9110(1);
  // ??

  // generar salida
  salida salida (&cout) ;
  salida.AbrirForm ("/cgi-bin/franqueo3.cgi");
  salida.CrearCabecera (TITULO9110) ;
  salida.InsertarTextoNeutro ("<CENTER>");

  // Enlace Cancelar
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Cancelar");
  salida.InsertarTextoNeutro ("</CENTER>");

  // Campo numero de matricula  solo lectura (se mantiene campo oculto
por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)

  sprintf (cadena," Nº matr: <I>%s</I><BR>", mat);
  salida.InsertarTextoNeutro(cadena);

  // Numero de modelo facturable oculto por si acaso
  // Campo Equipo/Linea (se mantiene campo oculto por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)
  sprintf (cadena, " Equ/Lin: <I>%s</I><BR>", eqlin);
  salida.InsertarTextoNeutro(cadena);
  salida.InsertarTextoNeutro("<B> ClaseAv: </B>");
  salida.InsertarSelectForm ( CAMPO_CLASEAV, nclaseav, claseav,
claseav, claseav[0]);

  salida.InsertarCampoOcultoForm (CAMPO_CATEGORIA, cat);
  salida.InsertarCampoOcultoForm (CAMPO_MAN, MANUAL);
  salida.InsertarCampoOcultoForm (CAMPO_ACT, av);
  salida.InsertarCampoOcultoForm (CAMPO_USER, user);
  salida.InsertarCampoOcultoForm (CAMPO_NMF, "");
  salida.InsertarCampoOcultoForm (CAMPO_MAT, mat);
  salida.InsertarCampoOcultoForm (CAMPO_EQLIN, eqlin);
  salida.InsertarCampoOcultoForm (CAMPO_EQINS, "");
  salida.InsertarCampoOcultoForm (CAMPO_NOMBREAV, "");
  salida.InsertarCampoOcultoForm (CAMPO_DHORAS, "");
```

```
  salida.InsertarCampoOcultoForm (CAMPO_DMIN, "");
  salida.InsertarCampoOcultoForm (CAMPO_DKM, "");
  salida.InsertarNuevaLinea ();
  salida.InsertarTextoNeutro ("<CENTER>");
  salida.InsertarBotonSubmitForm ("Siguiente");
  salida.InsertarTextoNeutro ("<CENTER>");
  salida.CerrarForm();
  return 0;
  }

int error9110 (int codigoerror=0) //Error con 9110
  {
  char msg_error[100];
  char cadena[200];
  Salida salida (&cout) ;
  salida.CrearCabecera ("Error durante Franqueo") ;
  if (codigoerror == 10) {
    salida.InsertarNuevaLinea();
    salida.InsertarTextoNeutro("<FONT
SIZE='+1'><B>  Variable REMOTE_USER no
definida.</B></FONT>");
    }
  else {
    strcpy (msg_error, errorSQL());
    salida.InsertarNuevaLinea();
    sprintf (cadena, "<FONT SIZE='+1'><B>  Error.
</B><BR>");
    salida.InsertarTextoNeutro (cadena);
    if (strstr (msg_error, "no data found")) {
      salida.InsertarNuevaLinea();
      salida.InsertarTextoNeutro  ("  Falta
informaci&oacute;n en la Base de Datos: <BR>");
      switch(codigoerror) {
        case 1:
salida.InsertarTextoNeutro("  Equipo/Linea");break;
        case 2: salida.InsertarTextoNeutro("  Equipo/Linea o
Clase de aver&iacute;a"); break;
        case 3: salida.InsertarTextoNeutro("  Nombre de
Aver&iacute;a/Equipo a instalar"); break;
        default : break;
        }
      salida.InsertarTextoNeutro("</FONT>");
      }
    else {
         if (strstr (msg_error, "SQL OK") == NULL)
           {
           salida.InsertarTextoNeutro("<FONT SIZE='+1'>");
           sprintf (cadena,"%s", msg_error);
           salida.InsertarTextoNeutro("  <B>Mensaje de error
de ORACLE: </B></FONT><BR>");
           salida.InsertarTextoNeutro  (cadena);
           }
         else salida.InsertarTextoNeutro("  <B>Compruebe que
usa correctamente el sistema.</B>");
         }
    }
  exit (-1);
  return 0;
}
```

```
int tipousuario9110 (char *user) // 9110
{
  if(!strcmp(user,USER_TEC))
    return TEC;
  else error9110(-10001);
}

int error9110 (int codError) //Error con 9110
{
  char cadena[200];
  Enlace *enlace ;
  char *av;
  char *user;

  Salida salida (&cout) ;
  salida.CrearCabecera (TIT9110ERROR);
  salida.InsertarTextoNeutro ("<P> <FONT SIZE=+1>");
  salida.InsertarTextoNeutro ("<I>Se ha producido un error en el
segundo paso de franqueo:</I>\n");
  salida.InsertarTextoNeutro ("<UL>\n");

  if (codError == ERR_VAR_REMOTEUSER)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>REMOTE_USER</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible control de acceso no
activado para el servidor WEB.\n");
  };

  if (codError == ERR_VAR_SERVERURL)
  {
    salida.InsertarTextoNeutro ("<LI>Variable <B>SERVER_URL</B> no
definida.\n");
    salida.InsertarTextoNeutro ("<LI>Posible ejecucion del cgi manual
fuera del entorno WEB.\n");
  };

  if (codError == ERR_PERFIL_USUARIO)
  {
    sprintf (cadena, "<LI>Usuario <B>%s</B> no autorizado.\n", getenv
("REMOTE_USER"));
    salida.InsertarTextoNeutro (cadena);
    salida.InsertarTextoNeutro ("<LI>Posible usuario sin perfil de
técnico o sin numero de matrícula.\n");
    salida.InsertarTextoNeutro ("<LI>Posible Servidor LDAP parado o no
accesible.\n");
  }

  if (codError == ERR_CONFIGURACION)
  {
    salida.InsertarTextoNeutro ("<LI>Fichero de <B>configuracion</B>
no accesible o con formato incorrecto.\n");
  }

  if (codError == ERR_ORACLE)
  {
    salida.InsertarTextoNeutro ("<LI>Error de acceso a la base de
datos.\n");
```

```
    sprintf (cadena, "<LI>Mensaje Oracle: <B>%s</B>\n", errorSQL ());
    salida.InsertarTextoNeutro (cadena);
  };

  if (codError == ERR_EQLIN_NOTFOUND)
  {
    salida.InsertarTextoNeutro ("<LI>Posible equipo seleccionado no
catalogado en GA.\n");
  };
   salida.InsertarTextoNeutro ("<CENTER><P>");

  // strdup porque el constructor de parametros destroza la variable
de entorno
  Parametros par(strdup(getenv("QUERY_STRING")));

  // recoger parametros
  av=par.DameValor(CAMPO_ACT);
  user=par.DameValor(CAMPO_USER);

  // Enlace Continuar
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Continuar");
  salida.InsertarTextoNeutro ("</CENTER>");
  return codError;
}
```

franqueo3.cgi

```
/* <MESA:01:@(#):MettnadJnjaw:ga2:1.5:990819084443:etna:1 34
50860:MESA> */
//  Alejandro Secades Gomez
//  franqueo3.cgi

#include <stdlib.h>
#include <string.h>
#include "CopiaHtml.h"
#include "entrada.h"
#include "defines.h"
#include "enlace.h"
#include "salida.h"  //9110
extern "C" {
#include "auxsql.h"
#include "auxldap.h"
}
#define TEC 0
#define PR 1
#define CO 2

#define EQUIPO '0'
#define LINEA '1'
#define CAT_MANUAL      '9'
#define MANUAL "9"
```

```
#define HTMLFRANQTEC    "../franqueo.html"
#define HTMLFRANQCO     "../franqueoco.html"
#define TITULO9110 "GA-Franqueo"                    //9110
#define ERR_VAR_REMOTUSER 1                         //9110

inline void error(int i) {
  //  cout << "Content-type: text/plain\n\n";

  cout << "Location: " << getenv("SERVER_URL") << "/errorfranq.html?"
       << "3: " << i << "-" << ((i==-10001)?"no data
found":errorSQL()) << "\n\n";
  exit(0);
}

inline void titulo(char *t, int negrita=0) {
  if(negrita) cout << "<B>";
  cout << t;
  if(negrita) cout << "</B>";
}

#define vacio(C,n) sal.readonly(C,"",n);

int tipousuario(char *user);

// Prototipo de funcion principal para browser Nokia 9110
int main9110 (); //9110
int tipousuario9110(char *user); // 9110
int error9110 (int codigoerror); //Error con 9110


int main()
{
  char *av, *user, *cat, *mat, *eqlin, *claseav, **nomav;
  // ??
  char **mats, **eqlins, **claseavs, *man;
  int nm, neq, nclaseav;
  long *cats;
  // ??
  int nnomav, tipou;    // 0 tec, 1 pr, 2 con
  Entrada *par;
  Enlace *enlace;
  char *remoteuser; //9110

  // Veamos quién se conecta al WEB
  remoteuser=getenv("REMOTE_USER");
//9110
  if (remoteuser == NULL) return error9110(ERR_VAR_REMOTUSER);
//9110
  if (strstr(remoteuser,"9110")) return main9110 ();      //9110

  // recoger parametros
  par=new Entrada(&cin);

  av=par->DameValor(CAMPO_ACT);
  user=par->DameValor(CAMPO_USER);
  man=par->DameValor(CAMPO_MAN);
  cat=par->DameValor(CAMPO_CATEGORIA);
  mat=par->DameValor(CAMPO_MAT);
  eqlin=par->DameValor(CAMPO_EQLIN);
```

```
  claseav=par->DameValor(CAMPO_CLASEAV);

  // distinguir tipo user
  tipou=tipousuario(user);

  // consultas
  if((nnomav=conecta())) error(nnomav);
  //cout << "Content-type: text/plain\n\n";

  switch(*cat) {
  case LINEA:
    if((nnomav=lfranqnomav(eqlin,claseav,&nomav))<=0) error(nnomav);
    break;
  case EQUIPO:
    if((nnomav=lfranqeqins(eqlin,&nomav))<=0) error(nnomav);
    break;
  default: error(-10001);
  }
  //  cout << nnomav << endl;

  // ??
  if((nclaseav=lfranqclav(atol(cat),&claseavs))<=0) error(nclaseav);
  if((neq=lfranqeqser(av,&eqlins,&cats))<=0) error(neq);
  if(tipou==PR)
  if((nm=lprmatgrupo(av,&mats))<=0) error(nm);
  // ??

  CopiaHtml sal((tipou==CO)?HTMLFRANQCO:HTMLFRANQTEC);      // salida
html
  sal.noreadonly((*cat==EQUIPO)?CAMPO_EQINS:CAMPO_NOMBREAV);
  // generar salida

  sal.copia();          // javascript, no hay nada
  // ??
  // cout << " ";
  cout << "var categorias= new makeArray(" << neq << ");\n";
  for(int i=0;i<neq;i++)
    cout << "categorias[" << (i+1) << "]=" << cats[i] << ";\n";
  // ??

  sal.copia();          // ayuda
  switch(tipou) {
  case CO:  cout << AYUDA_FRANQUEO_CON; break;
  case TEC: cout << AYUDA_FRANQUEO_TEC; break;
  case PR:  cout << AYUDA_FRANQUEO_PR; break;
  }

  sal.copia();          // action del form
  cout << "franqueo4.cgi";
  sal.copia();          // campos hidden
  sal.input(CAMPO_CATEGORIA,cat,5,"HIDDEN");
  if(man[0] == CAT_MANUAL)
    sal.input(CAMPO_MAN,MANUAL,5,"HIDDEN");
  else
    sal.input(CAMPO_MAN,"V",5,"HIDDEN");
  sal.input(CAMPO_ACT,av,15,"HIDDEN");
  sal.input(CAMPO_USER,user,3,"HIDDEN");
  sal.copia();
  if(*cat==EQUIPO) cout << "Seleccione el <B>equipo instalado:</B>";
```

```
else cout << "Seleccione el <B>nombre de la aver&iacute;a:</B>";

// ??
sal.copia();
enlace = new Enlace (CGI_DET_ACT) ;
enlace->InsertarCampo (CAMPO_ACT, av) ;
enlace->InsertarCampo (CAMPO_USER, user) ;
enlace->CodificaURL ();
enlace->InsertarCampoFin ();
cout << enlace->DameCadena ();
// ??

sal.copia();          // matricula
titulo((tipou==CO)?"Nº de contrata:":"Nº matr&iacute;cula:");
sal.copia();

// ??
// sal.readonly(CAMPO_MAT,mat,10);
if(tipou==PR)
  sal.seleccion(CAMPO_MAT,"envia(3);",mat,mats,nm);
else
  sal.readonly(CAMPO_MAT,mat,10);

/* if(tipou!=CO) { */
  sal.copia();        // nmf
  titulo("N&uacute;m. modelo fact:");
  sal.copia();
  vacio(CAMPO_NMF,7);
/* } */

// ???
if(tipou==CO) {
  sal.copia();
  titulo("Fecha-Hora Franqueo:");
  sal.copia();
  vacio(CAMPO_FHF,16);
}
// ???

sal.copia();          //eq/linea
titulo("Equipo/L&iacute;nea:");
sal.copia();
// ??
// sal.readonly(CAMPO_EQLIN,eqlin,(*cat==EQUIPO)?18:35);
if(man[0] == CAT_MANUAL)
  sal.readonly(CAMPO_EQLIN,eqlin,(*cat==EQUIPO)?18:35);
else
  sal.seleccion(CAMPO_EQLIN,"envia(2)",eqlin,eqlins,neq);
sal.copia();          // eq inst
if(*cat==EQUIPO) titulo("Equipo Instalado:",(*cat==EQUIPO)?1:0);
else cout << " ";
sal.copia();
//??
if(*cat==EQUIPO) sal.seleccion(CAMPO_EQINS,"envia(4)",nomav,nnomav);
else cout << " ";//vacio(CAMPO_EQINS,30);

sal.copia();
titulo("Clase de Aver&iacute;a:");
sal.copia();
```

```
  // ??
  // sal.readonly(CAMPO_CLASEAV,claseav,(*cat==LINEA)?20:30);

sal.seleccion(CAMPO_CLASEAV,"javascript:envia(3);",claseav,claseavs,nc
laseav);

  sal.copia();
  if(*cat==LINEA) titulo("Nombre de
Aver&iacute;a:",(*cat==LINEA)?1:0);
  else cout << " ";
  sal.copia();
  if(*cat==LINEA)
sal.seleccion(CAMPO_NOMBREAV,"envia(4)",nomav,nnomav);
  else cout << " ";//vacio(CAMPO_NOMBREAV,20);

  if(tipou!=CO) {
    sal.copia();        // horas
    titulo("Horas:");
    sal.copia();
    vacio(CAMPO_DHORAS,4);

    sal.copia();        // minutos
    titulo("Minutos:");
    sal.copia();
    vacio(CAMPO_DMIN,4);

    sal.copia();        // kilometros
    titulo("Kil&oacute;metros:");
    sal.copia();
    vacio(CAMPO_DKM,5);
  }

  sal.copia();          //resto
  return 0;
}

int tipousuario(char *user)
{
  if(!strcmp(user,USER_TEC))
    return TEC;
  else if(!strcmp(user,USER_CO))
    return CO;
  else if(!strcmp(user,USER_PR))
    return PR;
  else error(-10002);
}



//Funcion principal para browser Nokia 9110
int main9110()
{
  char *av, *user, *cat, *mat, *eqlin, *claseav, **nomav;
  // ??
  char **mats, **eqlins, **claseavs, *man;
  int nm, neq, nclaseav;
  long *cats;
  // ??
  int nnomav, tipou;    // 0 tec, 1 pr, 2 con
```

```
  Entrada *par;
  Enlace *enlace;
  char cadena[100];

  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  // recoger parametros
  par=new Entrada(&cin);

  av=par->DameValor(CAMPO_ACT);
  user=par->DameValor(CAMPO_USER);
  man=par->DameValor(CAMPO_MAN);
  cat=par->DameValor(CAMPO_CATEGORIA);
  mat=par->DameValor(CAMPO_MAT);
  eqlin=par->DameValor(CAMPO_EQLIN);
  claseav=par->DameValor(CAMPO_CLASEAV);

  // distinguir tipo user
  tipou=tipousuario9110(user);

  // consultas
  if((nnomav=conecta())) error9110(nnomav);
  //cout << "Content-type: text/plain\n\n";

  switch(*cat) {
  case LINEA:
    if((nnomav=lfranqnomav(eqlin,claseav,&nomav))<=0)
error9110(nnomav);
    break;
  case EQUIPO:
    if((nnomav=lfranqeqins(eqlin,&nomav))<=0) error9110(nnomav);
    break;
  default: error9110(-10001);
  }
  //  cout << nnomav << endl;

  // ??
  if((nclaseav=lfranqclav(atol(cat),&claseavs))<=0)
error9110(nclaseav);
  if((neq=lfranqeqser(av,&eqlins,&cats))<=0) error9110(neq);
  if(tipou==PR)
    if((nm=lprmatgrupo(av,&mats))<=0) error9110(nm);
  // ??

  // generar salida

  Salida salida (&cout) ;
  salida.AbrirForm ("/cgi-bin/franqueo4.cgi");
  salida.CrearCabecera (TITULO9110) ;
  salida.InsertarTextoNeutro ("<CENTER>");

  // Enlace Cancelar
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  salida.InsertarEnlace (enlace->DameCadena(), "Cancelar");
```

```
  salida.InsertarTextoNeutro ("</CENTER>");

  // Campo numero de matricula  solo lectura (se mantiene campo oculto
//por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
//9110)

  sprintf (cadena, " Nº matr: <I>%s</I><BR>", mat);
  salida.InsertarTextoNeutro(cadena);

  // Numero de modelo facturable oculto por si acaso
  // Campo Equipo/Linea (se mantiene campo oculto por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
//9110)

  sprintf (cadena, " Equ/Lin: <I>%s</I><BR>", eqlin);
  salida.InsertarTextoNeutro(cadena);

  // Campo Clase de avería (se mantiene campo oculto por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)

  sprintf (cadena, " ClaseAv: <I>%s</I><BR>", claseav);
  salida.InsertarTextoNeutro(cadena);
  if(*cat==EQUIPO) {
  salida.InsertarTextoNeutro(" <B>EqInsta: </B>");
  salida.InsertarSelectForm ( CAMPO_EQINS, nnomav, nomav, nomav,
nomav[0]);
     }
  else {
      salida.InsertarTextoNeutro(" <B>NomAver: </B>");
      //salida.InsertarNuevaLinea();
      salida.InsertarSelectForm ( CAMPO_NOMBREAV, nnomav, nomav,
nomav, nomav[0]);
     }
  salida.InsertarCampoOcultoForm (CAMPO_CATEGORIA, cat);
  salida.InsertarCampoOcultoForm (CAMPO_MAN, MANUAL);
  salida.InsertarCampoOcultoForm (CAMPO_ACT, av);
  salida.InsertarCampoOcultoForm (CAMPO_USER, user);
  salida.InsertarCampoOcultoForm (CAMPO_MAT, mat);
  salida.InsertarCampoOcultoForm (CAMPO_NMF, "");
  salida.InsertarCampoOcultoForm (CAMPO_EQLIN, eqlin);
  salida.InsertarCampoOcultoForm (CAMPO_CLASEAV, claseav);
  salida.InsertarCampoOcultoForm (CAMPO_NOMBREAV, "");
  salida.InsertarCampoOcultoForm (CAMPO_EQINS, "");
  salida.InsertarCampoOcultoForm (CAMPO_DHORAS, "");
  salida.InsertarCampoOcultoForm (CAMPO_DMIN, "");
  salida.InsertarCampoOcultoForm (CAMPO_DKM, "");

  salida.InsertarNuevaLinea ();

  salida.InsertarTextoNeutro ("<CENTER>");
  salida.InsertarBotonSubmitForm ("Siguiente");
  salida.InsertarTextoNeutro ("<CENTER>");

  salida.CerrarForm();

  return 0;
```

```
}

int error9110 (int codigoerror=0) //Error con 9110
  {
  char msg_error[100];
  char cadena[200];
  Salida salida (&cout) ;

  salida.CrearCabecera ("Error durante Franqueo") ;
  if (codigoerror == 10) {
    salida.InsertarNuevaLinea();
    salida.InsertarTextoNeutro("<FONT
SIZE='+1'><B>  Variable REMOTE_USER no
definida.</B></FONT>");
    }
  else {
    strcpy (msg_error, errorSQL());
    salida.InsertarNuevaLinea();
    sprintf (cadena, "<FONT SIZE='+1'><B>  Error.
</B><BR>");
    salida.InsertarTextoNeutro (cadena);
    if (strstr (msg_error, "no data found")) {
      salida.InsertarNuevaLinea();
      salida.InsertarTextoNeutro  ("  Falta
informaci&oacute;n en la Base de Datos: <BR>");
      switch(codigoerror) {
        case 1:
salida.InsertarTextoNeutro("  Equipo/Linea");break;
        case 2: salida.InsertarTextoNeutro("  Equipo/Linea o
Clase de aver&iacute;a"); break;
        case 3: salida.InsertarTextoNeutro("  Nombre de
Aver&iacute;a/Equipo a instalar"); break;
        default : break;
        }
      salida.InsertarTextoNeutro("</FONT>");
      }
    else {
         if (strstr (msg_error, "SQL OK") == NULL)
           {
           salida.InsertarTextoNeutro("<FONT SIZE='+1'>");
           sprintf (cadena,"%s", msg_error);
           salida.InsertarTextoNeutro("  <B>Mensaje de error
de ORACLE: </B></FONT><BR>");
           salida.InsertarTextoNeutro  (cadena);
           }
         else salida.InsertarTextoNeutro("  <B>Compruebe que
usa correctamente el sistema.</B>");
         }
    }
  exit (-1);
  return 0;
}
int tipousuario9110 (char *user) // 9110
{
  if(!strcmp(user,USER_TEC))
    return TEC;
  else error9110(-10002);
}
```

franqueo4.cgi

```
/* <MESA:01:@(#):MettnadJnjbg:ga2:1.4:990819084445:etna:1 34
43179:MESA> */
//  Alejandro Secades Gomez
//  franqueo4.cgi

#include <stdlib.h>
#include <string.h>
#include <floatingpoint.h>
#include <time.h>
#include "CopiaHtml.h"
#include "entrada.h"
#include "defines.h"
#include "enlace.h"
#include "salida.h"  //9110

extern "C" {
#include "webconf.h"
#include "auxsql.h"
#include "auxldap.h"
}

#define TEC 0
#define PR 1
#define CO 2

#define EQUIPO 0
#define LINEA 1

#define EQUIP '0'
#define LIN '1'

#define CAT_MANUAL      '9'
#define MANUAL          "9"

#define HTMLFRANQTEC2   "../franqueo2.html"
#define HTMLFRANQCO2    "../franqueoco2.html"

#define LONG_CAMPO_NMF 7      //9110
#define LONG_CAMPOS_TIEMPO 3  //9110
#define LONG_CAMPO_DISTKM 5  //9110

#define TITULO9110 "GA-Franqueo"                    //9110
#define ERR_VAR_REMOTUSER 1                         //9110

inline void error(int i) {
  cout << "Location: " << getenv("SERVER_URL") << "/errorfranq.html?"
       << i << "&" << errorSQL() << "\n\n";
  exit(0);
}

inline void titulo(char *t, int negrita=0) {
  if(negrita) cout << "<B>";
  cout << t;
  if(negrita) cout << "</B>";
}
```

```
#define vacio(C,n) sal.readonly(C,"",n);

int tipousuario(char *user);

// Prototipo de funcion principal para browser Nokia 9110
int main9110 (); //9110
int tipousuario9110(char *user); // 9110
int error9110 (int codigoerror); //Error con 9110


int main()
{
  char *av, *user, *cat, *mat, *eqlin, *claseav, *nomav, *eqins,
buf[100], tiempo_sistema[16+1], tiempo_reclamacion[16+1];
  // ??
  char **mats, **eqlins, **claseavs, **nomavs, *man;
  int nm, neq, nclaseav, nnnomav;
  long *cats;
  // ??
  int nnomav, tipou;          // 0 tec, 1 pr, 2 con
  long seg=3600,metros=1000, icat, i;
  // ???
  long fechaSistema;
  long fechaReclamacion;
  struct tm *tiempo;
  char cadena[100];
  // ???
  Entrada *par;
  Enlace *enlace;

   char *remoteuser; //9110

  // Veamos quién se conecta al WEB
  remoteuser=getenv("REMOTE_USER");
//9110
  if (remoteuser == NULL) return error9110(ERR_VAR_REMOTUSER);
//9110
  if (strstr(remoteuser,"9110")) return main9110 ();      //9110

  // recoger parametros
  par=new Entrada(&cin);

  av=par->DameValor(CAMPO_ACT);
  user=par->DameValor(CAMPO_USER);
  man=par->DameValor(CAMPO_MAN);
  cat=par->DameValor(CAMPO_CATEGORIA);
  mat=par->DameValor(CAMPO_MAT);
  eqlin=par->DameValor(CAMPO_EQLIN);
  claseav=par->DameValor(CAMPO_CLASEAV);
  nomav=par->DameValor(CAMPO_NOMBREAV);
  eqins=par->DameValor(CAMPO_EQINS);

  icat=atol(cat);
  // distinguir tipo user
  tipou=tipousuario(user);

  // consultas
  if((i=conecta())!=0) error(i);
```

```
  if(tipou!=CO)
  {
    if((i=datosdesplazam(av,&seg,&metros))!=0) error(i);
  }
  // ???
  else
  {
    if(leewebconf()) {
      cout << "<FONT COLOR='red'>";
      cout << "ERROR FATAL: Imposible cargar configuracion";
      cout << "</FONT>";
      return 0;
    }

    fechaSistema = time(NULL);
    tiempo = localtime(&fechaSistema);
    sprintf(tiempo_sistema, "%02d/%02d/%4d %02d:%02d", tiempo-
>tm_mday, tiempo->tm_mon+1, tiempo->tm_year+1900, tiempo->tm_hour,
tiempo->tm_min);

    if(FHReclam(av,&fechaReclamacion)) {
      cout << "<FONT COLOR='red'>";
      cout << "ERROR FATAL: No se puede consultar la base de datos";
      cout << "Pongase en contacto con su administrador";
      cout << "</FONT>";
      return 0;
    }

    tiempo = localtime(&fechaReclamacion);
    sprintf(tiempo_reclamacion, "%02d/%02d/%4d %02d:%02d", tiempo-
>tm_mday, tiempo->tm_mon+1, tiempo->tm_year+1900, tiempo->tm_hour,
tiempo->tm_min);

  }
  // ???


  // ??
  switch(*cat) {
  case LIN:
     if((nnnomav=lfranqnomav(eqlin,claseav,&nomavs))<=0)
error(nnnomav);
     break;
  case EQUIP:
     if((nnnomav=lfranqeqins(eqlin,&nomavs))<=0) error(nnnomav);
     break;
  default: error(-10001);
  }

  if((nclaseav=lfranqclav(atol(cat),&claseavs))<=0) error(nclaseav);

  if((neq=lfranqeqser(av,&eqlins,&cats))<=0) error(neq);
  if(tipou==PR)
    if((nm=lprmatgrupo(av,&mats))<=0) error(nm);
  // ??


  // generar salida
```

```
  CopiaHtml sal((tipou==CO)?HTMLFRANQCO2:HTMLFRANQTEC2);    // salida
html
  sal.noreadonly(CAMPO_DKM);

  sal.copia();          // javascript, no hay nada
  // ??
  // cout << " ";
  cout << "var categorias=new makeArray(" << neq<< ");\n";
  for(i=0;i<neq;i++)
    cout << "categorias[" << (i+1) << "]=" << cats[i] << ";\n";
  // ??

  // ???
  if(tipou==CO)
  {
    sal.copia();
    cout << webConf.min_fr;
    sal.copia();
    cout << tiempo_sistema;
    sal.copia();
    cout << tiempo_reclamacion;
  }
  // ???

  sal.copia();
  switch(tipou) {
  case CO:  cout << AYUDA_FRANQUEO_CON; break;
  case TEC: cout << AYUDA_FRANQUEO_TEC; break;
  case PR:  cout << AYUDA_FRANQUEO_PR; break;
  }

  sal.copia();          // action del form
  cout << "franquear.cgi";

  sal.copia();          // campos hidden
  sal.input(CAMPO_CATEGORIA,cat,5,"HIDDEN");
  if(man[0]==CAT_MANUAL)
     sal.input(CAMPO_MAN,MANUAL,5,"HIDDEN");
  else
     sal.input(CAMPO_MAN,"V",5,"HIDDEN");
  sal.input(CAMPO_ACT,av,15,"HIDDEN");
  sal.input(CAMPO_USER,user,3,"HIDDEN");
  if (tipou == CO)
     sal.input(CAMPO_FHSIS, tiempo_sistema,9,"HIDDEN");

  // ??
  sal.copia();
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
  cout << enlace->DameCadena ();
  // ??

  sal.copia();          // matricula
  titulo((tipou==CO)?"Nº de contrata:":"Nº matr&iacute;cula:");
  sal.copia();
```

```
if (tipou == CO)
  sal.readonly2(CAMPO_MAT,mat,10);
// ??
// sal.readonly2(CAMPO_MAT,mat,10);
else if (tipou == PR)
  sal.seleccion(CAMPO_MAT,"envia(4);",mat,mats,nm);
// ??
else
  sal.readonly(CAMPO_MAT,mat,10);

/* if(tipou!=CO) { */
  sal.copia();        // nmf
  titulo("N&uacute;m. modelo fact:",1);
  sal.copia();
  sal.input(CAMPO_NMF,"",7);
/* } */

// ???
if(tipou==CO) {
  sal.copia();
  titulo("Fecha-Hora Franqueo:",1);
  sal.copia();
  sal.input(CAMPO_FHF,tiempo_sistema,16);
}
// ???

sal.copia();          //eq/linea
titulo("Equipo/L&iacute;nea:");
sal.copia();

/* ??
if(tipou == CO)
  sal.readonly2(CAMPO_EQLIN,eqlin,35);
else
  sal.readonly(CAMPO_EQLIN,eqlin,35); */
if(man[0] == CAT_MANUAL)
  sal.readonly2(CAMPO_EQLIN,eqlin,35);
else
  sal.seleccion(CAMPO_EQLIN,"envia(2)",eqlin,eqlins,neq);
// ??

sal.copia();          // eq inst
if(icat==EQUIPO) titulo("Equipo Instalado:");
else cout << " ";
sal.copia();
if(icat==EQUIPO)
{
 /* ??
 if(tipou == CO)
   sal.readonly2(CAMPO_EQINS,eqins,35);
 else
   sal.readonly(CAMPO_EQINS,eqins,35); */
 sal.seleccion(CAMPO_EQINS,"envia(4)",eqins,nomavs,nnnomav);
 // ??
}
else cout << " ";

sal.copia();
titulo("Clase de Aver&iacute;a:");
```

```
  sal.copia();
  /* ??
  if(tipou == CO)
    sal.readonly2(CAMPO_CLASEAV,claseav,18);
  else
    sal.readonly(CAMPO_CLASEAV,claseav,18); */
  sal.seleccion(CAMPO_CLASEAV,"envia(3);",claseav,claseavs,nclaseav);
  // ??


  sal.copia();
  if(icat==LINEA) titulo("Nombre de Aver&iacute;a:");
  else cout << " ";
  sal.copia();
  if(icat==LINEA)
  {
   /* ??
   if(tipou == CO)
     sal.readonly2(CAMPO_NOMBREAV,nomav,35);
   else
     sal.readonly(CAMPO_NOMBREAV,nomav,35); */
   sal.seleccion(CAMPO_NOMBREAV,"envia(4)",nomav,nomavs,nnnomav);
   // ??
  }
  else cout << " ";

  if(tipou!=CO) {
    sal.copia();        // horas
    titulo("Horas:",1);
    sal.copia();
    sal.input(CAMPO_DHORAS,lltostr((seg/60)/60,buf),4, "TEXT",
              "onChange='esEntero(document.forms[0].DHORAS, true);'");


    sal.copia();        // minutos
    titulo("Minutos:",1);
    sal.copia();
    sal.input(CAMPO_DMIN,lltostr((seg/60)%60,buf),4, "TEXT",
              "onChange='esEntero(document.forms[0].DMIN, true);'");

    sal.copia();        // kilometros
    titulo("Kil&oacute;metros:",1);
    sal.copia();
    sprintf(buf,"%.3f",metros/1000.0);
    sal.input(CAMPO_DKM,buf,6, "TEXT",
              "onChange='esReal(document.forms[0].DKM, true);'");
  }

  sal.copia();          //resto

  return 0;
}

int tipousuario(char *user)
{
  if(!strcmp(user,USER_TEC))
    return TEC;
  else if(!strcmp(user,USER_CO))
    return CO;
```

```
  else if(!strcmp(user,USER_PR))
    return PR;
  else error(-10001);
}




//Funcion principal para browser Nokia 9110
int main9110() {
  char *av, *user, *cat, *mat, *eqlin, *claseav, *nomav, *eqins,
buf[100], tiempo_sistema[16+1], tiempo_reclamacion[16+1];
  // ??
  char **mats, **eqlins, **claseavs, **nomavs, *man;
  int nm, neq, nclaseav, nnnomav;
  long *cats;
  // ??
  int nnomav, tipou;          // 0 tec, 1 pr, 2 con
  long seg=3600,metros=1000, icat, i;
  // ???
  long fechaSistema;
  long fechaReclamacion;
  struct tm *tiempo;
  char cadena[100];
  // ???
  Entrada *par;
  Enlace *enlace;

  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  // recoger parametros
  par=new Entrada(&cin);

  av=par->DameValor(CAMPO_ACT);
  user=par->DameValor(CAMPO_USER);
  man=par->DameValor(CAMPO_MAN);
  cat=par->DameValor(CAMPO_CATEGORIA);
  mat=par->DameValor(CAMPO_MAT);
  eqlin=par->DameValor(CAMPO_EQLIN);
  claseav=par->DameValor(CAMPO_CLASEAV);
  nomav=par->DameValor(CAMPO_NOMBREAV);
  eqins=par->DameValor(CAMPO_EQINS);

  icat=atol(cat);
  // distinguir tipo user
  tipou=tipousuario9110(user);

  // consultas
  if((i=conecta())!=0) error9110(i);
  if(tipou!=CO)
  {
    if((i=datosdesplazam(av,&seg,&metros))!=0) error9110(i);
  }
  // ???
  else
  {
    if(leewebconf()) {
```

```
        cout << "ERROR FATAL: Imposible cargar configuracion";
      return 0;
      }
    fechaSistema = time(NULL);
    tiempo = localtime(&fechaSistema);
    sprintf(tiempo_sistema, "%02d/%02d/%4d %02d:%02d", tiempo-
>tm_mday, tiempo->tm_mon+1, tiempo->tm_year+1900, tiempo->tm_hour,
tiempo->tm_min);

    if(FHReclam(av,&fechaReclamacion)) {
      cout << "ERROR FATAL: No se puede consultar la base de datos";
      cout << "Pongase en contacto con su administrador";
      return 0;
    }

    tiempo = localtime(&fechaReclamacion);
    sprintf(tiempo_reclamacion, "%02d/%02d/%4d %02d:%02d", tiempo-
>tm_mday, tiempo->tm_mon+1, tiempo->tm_year+1900, tiempo->tm_hour,
tiempo->tm_min);

  }
  // ???


  // ??
  switch(*cat) {
  case LIN:
     if((nnnomav=lfranqnomav(eqlin,claseav,&nomavs))<=0)
error9110(nnnomav);
     break;
  case EQUIP:
     if((nnnomav=lfranqeqins(eqlin,&nomavs))<=0) error9110(nnnomav);
     break;
  default: error(-10001);
  }

  if((nclaseav=lfranqclav(atol(cat),&claseavs))<=0)
error9110(nclaseav);

  if((neq=lfranqeqser(av,&eqlins,&cats))<=0) error9110(neq);
  if(tipou==PR)
    if((nm=lprmatgrupo(av,&mats))<=0) error9110(nm);
  // ??


  // generar salida


  Salida salida (&cout) ;
  salida.AbrirForm ("/cgi-bin/franquear.cgi");
  salida.CrearCabecera (TITULO9110) ;
  salida.InsertarTextoNeutro ("<CENTER>");

  // Enlace Cancelar
  enlace = new Enlace (CGI_DET_ACT) ;
  enlace->InsertarCampo (CAMPO_ACT, av) ;
  enlace->InsertarCampo (CAMPO_USER, user) ;
  enlace->CodificaURL ();
  enlace->InsertarCampoFin ();
```

```
  salida.InsertarEnlace (enlace->DameCadena(), "Cancelar");

  salida.InsertarTextoNeutro("</CENTER>");

  // Campo numero de matricula  solo lectura (se mantiene campo oculto
por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)
  sprintf (cadena, " Nº matr: <I>%s</I>", mat);
  salida.InsertarTextoNeutro(cadena);
  salida.InsertarNuevaLinea();

  // Campo Equipo/Linea (se mantiene campo oculto por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)
  sprintf (cadena, " Equ/Lin: <I>%s</I>", eqlin);
  salida.InsertarTextoNeutro(cadena);
  salida.InsertarNuevaLinea();

  // Campo Clase de avería (se mantiene campo oculto por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)
  sprintf (cadena, " ClaseAv: <I>%s</I>", claseav);
  salida.InsertarTextoNeutro(cadena);
  salida.InsertarNuevaLinea();

  // Campo Nombre de avería y Campo Equipo instalado (se mantiene
campo oculto por si acaso se
  // necesita en el siguiente cgi y por compatibilidad con version sin
9110)


  if(*cat==EQUIP)  sprintf (cadena, " EqInsta: <I>%s</I>",
eqins);
  else  sprintf (cadena, " NomAver: <I>%s</I>", nomav);
  salida.InsertarTextoNeutro(cadena);
  salida.InsertarNuevaLinea();

  salida.InsertarTextoNeutro(" <B>ModFact: </B>");

salida.InsertarEntradaForm(CAMPO_NMF,"",LONG_CAMPO_NMF,LONG_CAMPO_NMF,
"");
  salida.InsertarNuevaLinea();

  salida.InsertarTextoNeutro(" DESPLAZAMIENTO");
  salida.InsertarNuevaLinea();
  salida.InsertarTextoNeutro(" <B>Horas: </B>");

salida.InsertarEntradaForm(CAMPO_DHORAS,"0",LONG_CAMPOS_TIEMPO,LONG_CA
MPOS_TIEMPO,"");

  salida.InsertarTextoNeutro("<B>Minutos: </B>");

salida.InsertarEntradaForm(CAMPO_DMIN,"0",LONG_CAMPOS_TIEMPO,LONG_CAMP
OS_TIEMPO,"");

  salida.InsertarTextoNeutro("<B>Kilómetros: </B>");
```

```
salida.InsertarEntradaForm(CAMPO_DKM,"0.000",LONG_CAMPO_DISTKM,LONG_CA
MPO_DISTKM,"");


  salida.InsertarCampoOcultoForm (CAMPO_CATEGORIA, cat);
  salida.InsertarCampoOcultoForm (CAMPO_MAN, MANUAL);
  salida.InsertarCampoOcultoForm (CAMPO_ACT, av);
  salida.InsertarCampoOcultoForm (CAMPO_USER, user);
  salida.InsertarCampoOcultoForm (CAMPO_MAT, mat);
  salida.InsertarCampoOcultoForm (CAMPO_EQLIN, eqlin);
  salida.InsertarCampoOcultoForm (CAMPO_CLASEAV, claseav);
  salida.InsertarCampoOcultoForm (CAMPO_EQINS, eqins);
  salida.InsertarCampoOcultoForm (CAMPO_NOMBREAV, nomav);


  salida.InsertarTextoNeutro ("<CENTER>");
  salida.InsertarBotonSubmitForm ("Franquear");
  salida.InsertarTextoNeutro ("<CENTER>");

  salida.CerrarForm();

  return 0;

}


int error9110 (int codigoerror=0) //Error con 9110
  {
  char msg_error[100];
  char cadena[200];
  Salida salida (&cout) ;

  salida.CrearCabecera ("Error durante Franqueo") ;
  if (codigoerror == 10) {
    salida.InsertarNuevaLinea();
    salida.InsertarTextoNeutro("<FONT
SIZE='+1'><B>  Variable REMOTE_USER no
definida.</B></FONT>");
    }
  else {
    strcpy (msg_error, errorSQL());
    salida.InsertarNuevaLinea();
    sprintf (cadena, "<FONT SIZE='+1'><B>  Error.
</B><BR>");
    salida.InsertarTextoNeutro (cadena);
    if (strstr (msg_error, "no data found")) {
      salida.InsertarNuevaLinea();
      salida.InsertarTextoNeutro  ("  Falta
informaci&oacute;n en la Base de Datos: <BR>");
      switch(codigoerror) {
        case 1:
salida.InsertarTextoNeutro("  Equipo/Linea");break;
        case 2: salida.InsertarTextoNeutro("  Equipo/Linea o
Clase de aver&iacute;a"); break;
        case 3: salida.InsertarTextoNeutro("  Nombre de
Aver&iacute;a/Equipo a instalar"); break;
        default : break;
        }
```

```
      salida.InsertarTextoNeutro("</FONT>");
      }
    else {
         if (strstr (msg_error, "SQL OK") == NULL)
           {
           salida.InsertarTextoNeutro("<FONT SIZE='+1'>");
           sprintf (cadena,"%s", msg_error);
           salida.InsertarTextoNeutro("  <B>Mensaje de error
de ORACLE: </B></FONT><BR>");
           salida.InsertarTextoNeutro  (cadena);
           }
         else salida.InsertarTextoNeutro("  <B>Compruebe que
usa correctamente el sistema.</B>");
         }
    }
  exit (-1);
  return 0;
}

int tipousuario9110 (char *user) // 9110
{
  if(!strcmp(user,USER_TEC))
    return TEC;
  else error9110(-10001);
}
```

franquear.cgi

```
/* <MESA:01:@(#):MettnadJnj8g:ga2:1.6:990819084436:etna:1 34
23725:MESA> */
//-------------------------------------------------------------------------
-------
//
// Proyecto:                    GA
//
// Grupo:                       Tecnico-Contratas
//
// Fichero:                     franqueo.C
//
// Autor:
//
// Fecha creacion:              21/01/19999
//
// Fecha ultima modificacion:
//
//------------------------------ DESCRIPCION ------------------------
-------
//
//
//
//------------------------------ RESTRICCIONES ------------------------
-------
//
//------------------------------ COMENTARIOS ------------------------
-------
//
```

```
//-------------------------------------------------------------------------
-------
//                (c) Telefonica Investigacion y Desarrollo
//-------------------------------------------------------------------------
-------


#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <fstream.h>
#include <ctype.h>

#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <entrada.h>

extern "C" {
#include "webconf.h"
#include "auxars.h"
#include "auxsql.h"
}

#define TITULO      "Franquear averia"
#define ESTEC 0
#define ESPR 1
#define ESCO 2

#define EQUIPO 0
#define LINEA 1
#define TITULO9110 "GA-Franqueo"                    //9110
#define ERR_VAR_REMOTUSER 1                         //9110

long fechaeq(char *fecha);

// Prototipo de funcion principal para browser Nokia 9110
  int main9110 (); //9110
  int error9110 (int codigoerror);

int main()
{

  char *var,
       *user = NULL,
       *act = NULL,
       *mat, *cat,
       *eqlin, *claseav, *eqins=NULL, *nomav=NULL,
       *nmf,
       *fhfranqueo,
       *fhsistema,
       *atras;
  long horas, min, icat;
  long fechaSistema, fechaFranqueo;
  double km;
  int tipo, i;          // 0 tecnico, 1 punto de reunion, 2 contrata

  Entrada *parametros ;
  Enlace     *enlace ;
```

```
   char *remoteuser; //9110

  // Veamos quién se conecta al WEB
  remoteuser=getenv("REMOTE_USER");
//9110
  if (remoteuser == NULL) return error9110(ERR_VAR_REMOTUSER);
//9110
  if (strstr(remoteuser,"9110")) return main9110 ();
//9110

  Salida salida (&cout) ;

  parametros = new Entrada(&cin) ;

  user = parametros->DameValor (CAMPO_USER);
  if(!strcmp(user,USER_TEC)) tipo=ESTEC;
  else if(!strcmp(user,USER_CO)) tipo=ESCO;
  else tipo=ESPR;

  act = parametros->DameValor(CAMPO_ACT);
  mat=parametros->DameValor(CAMPO_MAT);
  eqlin=parametros->DameValor(CAMPO_EQLIN);
  cat=parametros->DameValor(CAMPO_CATEGORIA);
  icat=atol(cat);
  claseav=parametros->DameValor(CAMPO_CLASEAV);
  if(icat==EQUIPO) eqins=parametros->DameValor(CAMPO_EQINS);
  else nomav=parametros->DameValor(CAMPO_NOMBREAV);

  nmf=parametros->DameValor(CAMPO_NMF);

  if(!(tipo==ESCO)) {
    // nmf=parametros->DameValor(CAMPO_NMF);
    horas=atol(parametros->DameValor(CAMPO_DHORAS));
    min=atol(parametros->DameValor(CAMPO_DMIN));
    km=atof(parametros->DameValor(CAMPO_DKM));
  }
  else
  {
    fhsistema=parametros->DameValor(CAMPO_FHSIS);
    fhfranqueo=parametros->DameValor(CAMPO_FHF);
    fechaSistema = fechaeq(fhsistema);
    fechaFranqueo = fechaeq(fhfranqueo);
  }
  salida.CrearCabecera(TITULO,(tipo==ESTEC)? AYUDA_FRANQUEO_TEC:

((tipo==ESCO)?AYUDA_FRANQUEO_CON:AYUDA_FRANQUEO_PR),
                   TIT_FRANQUEO, NULL);

  salida.InsertarTexto ("<CENTER><FONT SIZE=4>") ;

  if(leewebconf() || !isdigit(*mat)) {
    salida.InsertarTexto ("<FONT COLOR='red'>") ;
    salida.InsertarTexto("ERROR FATAL: Imposible cargar
configuracion");
    salida.InsertarTexto ("</FONT>") ;
    salida.InsertarTexto(
```

```
                  "Pongase en contacto con el administrador web o
pruebe");
    salida.InsertarTexto (
                   "a utilizar correctamente el web, siguiendo el
manual");
  }
  else {
    if(tipo==ESCO){

i=franqueoCoARS(webConf.loginAR,webConf.passwdAR,webConf.servAR,
                  act,mat,eqlin,claseav,eqins,nomav,fechaSistema,
                  fechaFranqueo,(isalpha(*nmf))?nmf:NULL);
      //if(i==0)

//i=FranqueoActCo(webConf.loginAR,webConf.passwdAR,webConf.servAR,act)
;
    }
    else
i=franqueoTecARS(webConf.loginAR,webConf.passwdAR,webConf.servAR,
                    act,mat,eqlin,claseav,eqins,nomav,
                    horas,min,km,(isalpha(*nmf))?nmf:NULL);
    if(i==0) {
      char nueva_act[16], ngrupo[16];
      if(webConf.asigna==0 && sigActAsig(mat,act,nueva_act,ngrupo)==1)
      if(grupoLibre(ngrupo))

asignarARS(webConf.loginAR,webConf.passwdAR,webConf.servAR,nueva_act,n
grupo);
    }
    switch(i) {
    case 0:
      salida.InsertarTexto("Se ha franqueado correctamente la
aver&iacute;a");
      break;
    case 1:
      salida.InsertarTexto(
            "Se ha franqueado la aver&iacute;a pero el sistema ha
devuelto") ;
      salida.InsertarTexto ("el siguiente mensaje de advertencia:");
      salida.InsertarTexto ("<FONT COLOR='red'>") ;
      salida.InsertarTexto(errorARS(1));  // de momento con uno vale
      salida.InsertarTexto ("</FONT>") ;
      salida.InsertarNuevaLinea();
      salida.InsertarTexto("P&oacute;ngase en contacto con el
administrador ARS");
      break;
    default:
      salida.InsertarTexto ("<FONT COLOR='red'>") ;
      salida.InsertarTexto(
                  "Error de ARS: No se ha podido franquear la
aver&iacute;a.");
      salida.InsertarTexto ("</FONT>") ;
      salida.InsertarTexto("Mensaje de error:");
      salida.InsertarTexto ("<FONT COLOR='red'>") ;
      salida.InsertarNuevaLinea();
      salida.InsertarTexto(errorARS(1));  // de momento con uno vale
      salida.InsertarTexto ("</FONT>") ;
      salida.InsertarNuevaLinea();
```

```
      salida.InsertarTexto("P&oacute;ngase en contacto con el
administrador ARS");
    }
  }

  salida.InsertarTexto ("</FONT>") ;
  switch(tipo) {
  case ESTEC: atras=CGI_TEC_ACT; break;
  case ESPR: atras=PAG_FRAMEPR; break;
  case ESCO: atras=PAG_FRAMECO;
  }

  salida.InsertarGifSensible(BOT_CONTINUAR,"Continuar","atras",atras,
NULL);
  return 0;
}


long fechaeq(char *fecha)
{

char *fecha_aux, *fecha_aux2;

struct tm tiempo;
long resultado = 0;

int dia, mes, anno, hora, minutos, segundos;

dia = (int) strtol(fecha, &fecha_aux, 10);
mes = ((int) strtol(fecha_aux+1, &fecha_aux2, 10)) - 1 ;
anno = ((int) strtol(fecha_aux2+1, &fecha_aux, 10)) - 1900;
hora = (int) strtol(fecha_aux+1, &fecha_aux2, 10);
minutos = (int) strtol(fecha_aux2+1, &fecha_aux, 10);
segundos =0;

tiempo.tm_mday = dia;
tiempo.tm_mon = mes ;
tiempo.tm_year = anno ;
tiempo.tm_hour = hora ;
tiempo.tm_min = minutos ;
tiempo.tm_sec = segundos;
tiempo.tm_isdst = 1;

resultado = mktime (&tiempo);

if (tiempo.tm_hour != hora)
{
  tiempo.tm_mday = dia;
  tiempo.tm_mon = mes ;
  tiempo.tm_year = anno ;
  tiempo.tm_hour = hora ;
  tiempo.tm_min = minutos ;
  tiempo.tm_sec = segundos;
  tiempo.tm_isdst = 0;
  resultado = mktime (&tiempo);
}

return resultado;
```

```
}



//Funcion principal para browser Nokia 9110
int main9110() {
  char *var,
       *user = NULL,
       *act = NULL,
       *mat, *cat,
       *eqlin, *claseav, *eqins=NULL, *nomav=NULL,
       *nmf,
       *fhfranqueo,
       *fhsistema,
       *atras;
  long horas, min, icat;
  long fechaSistema, fechaFranqueo;
  double km;
  int tipo, i;          // 0 tecnico, 1 punto de reunion, 2 contrata
  char cadena[100];

  Entrada *parametros ;
  Enlace     *enlace ;

  // Para evitar el uso de cache en los cgi
  cout << "Pragma: no-cache" << endl;

  Salida salida (&cout) ;

  parametros = new Entrada(&cin) ;
  user = parametros->DameValor (CAMPO_USER);
  act = parametros->DameValor(CAMPO_ACT);
  mat=parametros->DameValor(CAMPO_MAT);
  eqlin=parametros->DameValor(CAMPO_EQLIN);
  cat=parametros->DameValor(CAMPO_CATEGORIA);
  icat=atol(cat);
  claseav=parametros->DameValor(CAMPO_CLASEAV);
  if(icat==EQUIPO) eqins=parametros->DameValor(CAMPO_EQINS);
  else nomav=parametros->DameValor(CAMPO_NOMBREAV);

  nmf=parametros->DameValor(CAMPO_NMF);
  horas=atol(parametros->DameValor(CAMPO_DHORAS));
  min=atol(parametros->DameValor(CAMPO_DMIN));
  km=atof(parametros->DameValor(CAMPO_DKM));

  //generar salida

  salida.CrearCabecera ("GA-Franqueo") ;
  salida.InsertarNuevaLinea();
  salida.InsertarTextoNeutro ("<CENTER>");

  // Enlace Continuar
    enlace = new Enlace (CGI_TEC_ACT) ;
    enlace->InsertarCampo (CAMPO_USER, "TEC") ;
    enlace->CodificaURL () ;
    enlace->InsertarCampoFin () ;
```

```
    salida.InsertarEnlace (enlace->DameCadena(), "Continuar");
    delete enlace ;
    salida.InsertarTextoNeutro ("</CENTER>");
    salida.InsertarNuevaLinea();


  if(leewebconf() || !isdigit(*mat)) {
    salida.InsertarTextoNeutro("<FONT SIZE='+1'>  <B>ERROR
FATAL: </B>Imposible cargar configuracion.<BR>");
    salida.InsertarTextoNeutro("  Pongase en contacto con el
administrador web o pruebe");
    salida.InsertarTextoNeutro("a utilizar correctamente el web,
siguiendo el manual.</FONT>");
  }
  else
i=franqueoTecARS(webConf.loginAR,webConf.passwdAR,webConf.servAR,
                   act,mat,eqlin,claseav,eqins,nomav,
                   horas,min,km,(isalpha(*nmf))?nmf:NULL);
  if(i==0) {
      char nueva_act[16], ngrupo[16];
      if(webConf.asigna==0 && sigActAsig(mat,act,nueva_act,ngrupo)==1)
      if(grupoLibre(ngrupo))

asignarARS(webConf.loginAR,webConf.passwdAR,webConf.servAR,nueva_act,n
grupo);
    }
  switch(i) {
    case 0:
      salida.InsertarTextoNeutro("  <FONT SIZE='+1'><B>Se ha
franqueado correctamente la aver&iacute;a.</B></FONT>");
      break;
    case 1:
      salida.InsertarTextoNeutro("<B>  Se ha franqueado la
aver&iacute;a pero el sistema ha devuelto") ;
      salida.InsertarTextoNeutro ("el siguiente mensaje de
advertencia: </B><BR>");
      salida.InsertarTextoNeutro(errorARS(1));  // de momento con uno
vale
      salida.InsertarNuevaLinea();
      salida.InsertarTexto("  P&oacute;ngase en contacto con
el administrador ARS.");
      break;
    default:
      salida.InsertarTextoNeutro("<B>  Error de ARS: </B>No
se ha podido franquear la aver&iacute;a.<BR>");
      salida.InsertarTextoNeutro("  Mensaje de error:");
      salida.InsertarNuevaLinea();
      salida.InsertarTexto(errorARS(1));  // de momento con uno vale
      salida.InsertarNuevaLinea();
      salida.InsertarTexto("  P&oacute;ngase en contacto con
el administrador ARS.");
    }
  return 0;
}

int error9110 (int codigoerror=0) //Error con 9110
  {
  char msg_error[100];
  char cadena[200];
```

```
  Salida salida (&cout) ;

  salida.CrearCabecera ("Error durante Franqueo") ;
  if (codigoerror == 10) {
    salida.InsertarNuevaLinea();
    salida.InsertarTextoNeutro("<FONT
SIZE='+1'><B>  Variable REMOTE_USER no
definida.</B></FONT>");
    }
  else {
    strcpy (msg_error, errorSQL());
    salida.InsertarNuevaLinea();
    sprintf (cadena, "<FONT SIZE='+1'><B>  Error.
</B><BR>");
    salida.InsertarTextoNeutro (cadena);
    if (strstr (msg_error, "no data found")) {
      salida.InsertarNuevaLinea();
      salida.InsertarTextoNeutro  ("  Falta
informaci&oacute;n en la Base de Datos: <BR>");
      switch(codigoerror) {
        case 1:
salida.InsertarTextoNeutro("  Equipo/Linea");break;
        case 2: salida.InsertarTextoNeutro("  Equipo/Linea o
Clase de aver&iacute;a"); break;
        case 3: salida.InsertarTextoNeutro("  Nombre de
Aver&iacute;a/Equipo a instalar"); break;
        default : break;
        }
      salida.InsertarTextoNeutro("</FONT>");
      }
    else {
         if (strstr (msg_error, "SQL OK") == NULL)
           {
           salida.InsertarTextoNeutro("<FONT SIZE='+1'>");
           sprintf (cadena,"%s", msg_error);
           salida.InsertarTextoNeutro("  <B>Mensaje de error
de ORACLE: </B></FONT><BR>");
           salida.InsertarTextoNeutro  (cadena);
           }
         else salida.InsertarTextoNeutro("  <B>Compruebe que
usa correctamente el sistema.</B>");
         }
    }
  exit (-1);
  return 0;
}
```

PÁGINAS WEB WML

start.wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
```

```
                        "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <head>
    <meta http-equiv="Cache-Control" content="max-age=0"/>
  </head>
  <card id="main">
    <do type="accept" label="Sig">
      <go href="/wap-cgi/detalleWap.cgi?TFNOMOV=649470888" />
    </do>
    <p align="center"><b><i>GA</i></b></p>
    <p align="center"><b><i>Gestor de Actividad</i></b></p>
    <p align="center"><img alt="telef" src="" localsrc="phone1" /></p>
  </card>
</wml>
```

detalleWap.cgi

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <parametros.h>

extern "C" {
#include <webconf.h>
#include <auxldap.h>
#include <auxsql.h>
#include "auxars.h"
#include "digest.h"
        }

#define ERROR_CONFIGURACION 1
#define ERROR_ORACLE        2
#define ERROR_ARS           3

int errorWap (int codigoerror,char *texto);

void OutputBuffer (char *buffer, int length)
{
      int i;

      i=0;
      while (i < length) {
            printf("%c",*(buffer+i));
            i++;
      }
}

main ()
{
        Digest d;
        char *output;
        int outputLength, ret;
      char texto[5000];
        char *telef=NULL;
```

```
        char *var=NULL;
        char url[500];
        char tec[30],actuac[20];
        Parametros *parametros ;
        Enlace     *enlace ;
        Actuacion act;

        d = DigestConstruct();
        var = getenv ("QUERY_STRING");
        parametros = new Parametros (var);
        telef = parametros->DameValor (CAMPO_TFNO_MOVIL);
        if (leewebconf()) errorWap(ERROR_CONFIGURACION,texto);
        else {
          if (conecta()) errorWap(ERROR_ORACLE,texto);
          else {
            ret = algunaEnCursoWap (telef,tec,actuac);
            if (ret == 1403) {
              enlace = new Enlace (INICIO);
              sprintf(url,"%s",enlace->DameCadena ());
              delete enlace ;
              sprintf(texto,"<wml>"
                            "<head><meta http-equiv=\"Cache-
Control\" content=\"max-age=0\"/></head>"
                            "<card>"
                              "<do type=\"accept\"
label=\"Inicio\">"
                                "<go href=\"%s\"/>"
                              "</do>"
                              "<p align=\"center\">"
                              "<br/>"
                              "<b>Tec: </b><i>%s</i><br/>"
                              "<b>SIN ACTUACIONES</b>"
                              "</p></card></wml>",url,tec);
                           }
            else {
              ret = ObtenerMensajes (telef,&act);
              if (ret == SQL_OK) {
                enlace = new Enlace (CGI_FRANQUEOWAP1);
                enlace->InsertarCampo (CAMPO_ACT, act.entry_id);
                enlace->InsertarCampo (CAMPO_USER, act.tecnico);
                enlace->InsertarCampo (CAMPO_TFNO, act.telefono);
                enlace->CodificaURL();
                sprintf(url,"%s",enlace->DameCadena ());
                delete enlace ;
                sprintf (texto,"<wml>"
                               "<head><meta http-equiv=\"Cache-
Control\" content=\"max-age=0\"/></head>"
                         "<card id=\"card\">"
                           "<do type=\"options\" label=\"Franq\">"
                                 "<go href=\"%s\"/>"
                               "</do>"
                          "<p mode=\"nowrap\">"
                             "<b><i>DATOS AVERIA:</i></b>"
                             "<select ivalue=\"1\">"
                               "<option onpick=\"#card1\">Datos
Asignacion</option>"
                               "<option onpick=\"#card2\">Datos
Cliente I</option>"
```

```
                                    "<option onpick=\"#card3\">Datos
Cliente II</option>"
                                      "<option
onpick=\"#card4\">Observaciones</option>"
                                  "</select>"
                                "</p>"
                              "</card>"
                              "<card id=\"card1\">"
                                        "<do type=\"accept\">"
                                          "<go href=\"#choice\"/>"
                                        "</do>"
                                        "<do type=\"options\"
label=\"Back\">"
                                          "<prev/>"
                                        "</do>"
                                "<p mode=\"nowrap\">"
                                   "<b><i>DATOS ASIGNACION:</i></b>"
                                   "<select name=\"choice\" ivalue=\"1\">"
                                           "<option value=\"%s\">Tipo
Bucle</option>"
                                          "<option
value=\"%s\">Grupo</option>"
                                          "<option
value=\"%s\">Par</option>"
                                   "<option value=\"%s\">Pto
Interconexion</option>"
                                   "<option value=\"%s\">Par
Entrada</option>"
                                            "<option value=\"%s\">Par
Salida</option>"
                                   "<option value=\"%s\">Caja
Terminal</option>"
                                   "<option value=\"%s\">Sit. Caja
Terminal</option>"
                                   "<option value=\"%s\">Sit. Caja
Conexion</option>"
                                   "</select>"
                                "</p>"
                              "</card>"
                                     "<card id=\"card2\">"
                                        "<do type=\"accept\">"
                                          "<go href=\"#choice\"/>"
                                        "</do>"
                                        "<do type=\"options\"
label=\"Back\">"
                                          "<prev/>"
                                        "</do>"
                                        "<p>"
                                          "<b><i>DAT CLIENTE I:</i></b>"
                                          "<select name=\"choice\"
ivalue=\"1\">"
                                            "<option
value=\"%s\">Titular</option>"
                                            "<option
value=\"%s\">Domicilio</option>"
                                            "<option
value=\"%s\">Poblacion</option>"
                                          "</select>"
                                        "</p>"
```

```
                        "</card>"
                        "<card id=\"card3\">"
                                "<do type=\"accept\">"
                                  "<go href=\"#choice\"/>"
                                "</do>"
                                "<do type=\"options\"
label=\"Back\">"
                                  "<prev/>"
                                "</do>"
                                "<p mode=\"nowrap\">"
                                   "<b><i>DAT CLIENTE II:</i></b>"
                                   "<select name=\"choice\"
ivalue=\"1\">"
                                     "<option
value=\"%s\">Sintoma</option>"
                                     "<option
value=\"%s\">Telefono</option>"
                                     "<option
value=\"%s\">Central</option>"
                                     "<option
value=\"%s\">Resultado Prueba</option>"
                                     "<option value=\"%s\">Tipo
Cliente</option>"
                                     "<option value=\"%s\">Reg.
Comercial</option>"
                                     "<option value=\"%s\">Fecha
Cita</option>"
                                     "<option value=\"%s\">Hora
Cita</option>"
                                     "<option
value=\"%s/%s\">TP.EM/N.EM</option>"
                                   "</select>"
                                "</p>"
                        "</card>"
                                "<card id=\"card4\">"
                                  "<do type=\"accept\">"
                                    "<go href=\"#choice\"/>"
                                  "</do>"
                                  "<do type=\"options\"
label=\"Back\">"
                                    "<prev/>"
                                  "</do>"
                                  "<p>"
                                     "<b><i>OBSERVACIONES:</i></b>"
                              "<select name=\"choice\" ivalue=\"1\">"
                                       "<option value=\"%s\">Tipo
Aviso</option>"
                                       "<option value=\"%s\">Obs.
mecanico</option>"
                                "<option
onpick=\"#equipos\">Equipos/Serv.</option>"
                                     "</select>"
                             "</p>"
                                 "</card>"
                        "<card id=\"choice\">"
                                "<do type=\"accept\"
label=\"Back\">"
                                  "<prev/>"
                                "</do>"
```

```
                              "<p>"
                                "$(choice)"
                              "</p>"
                            "</card>"
                            "<card id=\"equipos\">"
                                      "<do type=\"accept\"
label=\"Back\">"
                                        "<prev/>"
                                      "</do>"
                                      "<p mode=\"nowrap\">"
                                        "%s"
                                      "</p>"
                                    "</card>"
                          "</wml>",url,act.bucle, act.grupo, act.par,
act.pto_int,
                         act.par_ent, act.par_sal, act.caja_term,
act.sit_caja_term,
                         act.sit_conexion, act.titular, act.domicilio,
act.poblacion,
                         act.sintoma, act.telefono, act.central,
act.res_prueba,
                         act.tipo_cli, act.reg_comer, act.fecha_cita,
act.hora_cita,
                         act.TP_EM, act.N_EM, act.tipo_aviso,
act.observ_mec,
                                 act.equipos_serv);
                             }
           else errorWap(ERROR_ARS,texto);
          }
        }
        }
    DigestAddDeck(d, "",texto, NULL);
    output = DigestSerialize(d, &outputLength);

    OutputBuffer(output,outputLength);

    /*
    The caller of DigestSerialize is responsible for deleting the
output buffer
    after using it
    */
    free(output);

    /* Delete the digest object */
    DigestDestruct(d);
}

int errorWap (int codigoerror,char *texto)

{
  Enlace     *enlace ;
  char url[500];

  enlace = new Enlace (INICIO) ;
  sprintf(url,"%s",enlace->DameCadena ());
  delete enlace ;
  switch(codigoerror) {
             case ERROR_CONFIGURACION:
                 sprintf(texto,"<wml>"
```

```
                        "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                        "</head>"
                        "<card>"
                        "<do type=\"accept\" label=\"Inicio\">"
                          "<go href=\"%s\"/>"
                        "</do>"
                        "<p align=\"center\">"
                        "Fichero de <b>configuracion</b>"
                        " no accesible o con formato incorrecto.<br/>"
                        "</p></card></wml>",url);
                break;
            case ERROR_ORACLE:
                sprintf(texto,"<wml>"
                        "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                        "</head>"
                        "<card>"
                        "<do type=\"accept\" label=\"Inicio\">"
                          "<go href=\"%s\"/>"
                        "</do>"
                        "<p align=\"center\">"
                        "Error de acceso a la base de datos.<br/>"
                        "<b>Mensaje Oracle:</b><br/>"
                        "%s</p></card></wml>",url,errorSQL ());
                break;
            case ERROR_ARS:
                sprintf(texto,"<wml>"
                        "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                        "</head>"
                        "<card>"
                        "<do type=\"accept\" label=\"Inicio\">"
                          "<go href=\"%s\"/>"
                        "</do>"
                        "<p align=\"center\">"
                        "Error de acceso a ARS.<br/>"
                        "<b>Mensaje ARS:</b></p>"
                        "<p
align=\"left\">%s</p></card></wml>",url,errorARS(1));
                break;
            default: sprintf(texto,"<wml>"
                        "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                        "</head>"
                        "<card>"
                        "<do type=\"accept\" label=\"Inicio\">"
                          "<go href=\"%s\"/>"
                        "</do>"
                        "<p align=\"center\">"
                        "<b>ERROR</b>"
                        "</p></card></wml>",url);
          }
  return 0;
}
```

franqueoWap1.cgi

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <parametros.h>

extern "C" {
#include <webconf.h>
#include <auxldap.h>
#include <auxsql.h>
#include "auxars.h"
#include "digest.h"
          }

#define ERROR_CONFIGURACION 1
#define ERROR_ORACLE        2
#define ERR_EQLIN_NOTFOUND  3

int errorWap (int codigoerror,char *texto);

void OutputBuffer (char *buffer, int length)
{
      int i;

      i=0;
      while (i < length) {
            printf("%c",*(buffer+i));
            i++;
      }
}


main ()
{
   Digest d;
   char *output;
   int outputLength;
   char texto[2000];
   char url[500],
        cadena[500];
   char *act=NULL;
   char *tec=NULL,
        *telef=NULL;
   char *var=NULL;

   Parametros *parametros ;
   Enlace     *enlace ;

   char **eqlin;
   char **claseav;
   long *cats;
   char cat[10];
   int neq,nclaseav,ncat,i,z;

    /* Create the Digest object */
    d = DigestConstruct();
```

```
    /*
    Add any needed sub parts to the digest
    In this case, add the deck to the displayed when this digest is
requested
    and add the Alert to send to the phone
    */

   var = getenv ("QUERY_STRING") ;
   parametros = new Parametros (var) ;
   act = parametros->DameValor (CAMPO_ACT) ;
   tec = parametros->DameValor (CAMPO_USER) ;
   telef = parametros->DameValor (CAMPO_TFNO) ;
   if (leewebconf()) errorWap(ERROR_CONFIGURACION,texto);
   else {
     if (conecta()) errorWap(ERROR_ORACLE,texto);
     else {
          neq=lfranqeqser(act,&eqlin,&cats);
          sprintf(texto,"<wml>"
                "<head>"
                "<meta http-equiv=\"Cache-Control\" content=\"max-
age=0\"/>"
                "</head>"
                "<card id=\"main\">"
            "<p mode=\"nowrap\">"
            "<b>Tec: </b><i>%s</i><br/>"
                "<b>Av: </b><i>%s</i><br/>"
            "<b>Equipo/Linea:</b>"
            "<select name=\"equiplin\">",tec,telef);
          for ( i=0 ; i<neq ; i++) {
                sprintf(cadena,"<option value=\"%s\"
onpick=\"#card%d\">%s</option>",eqlin[i],i,eqlin[i]);
                strcat (texto,cadena);
              }
        sprintf(cadena,"</select></p></card>");
        strcat (texto,cadena);
        for ( i=0 ; i<neq ; i++) {
            ncat=catManual(eqlin[i],cat);
            nclaseav = lfranqclav(atol(cat),&claseav);
            enlace = new Enlace ();
              enlace->InsertarCampo (CAMPO_ACT, act);
              enlace->InsertarCampo (CAMPO_USER,tec);
              enlace->InsertarCampo (CAMPO_TFNO,telef);
              enlace->InsertarCampo (CAMPO_EQLIN,eqlin[i]);
              enlace->InsertarCampo (CAMPO_CATEGORIA,cat);
              enlace->InsertarCampo (CAMPO_CLASEAV,"$(claseav)");
              enlace->CodificaURL();
              sprintf(url,"%s%s",CGI_FRANQUEOWAP2,enlace->DameCadena
());
              delete enlace;
            sprintf(cadena,"<card id=\"card%d\"><p mode=\"nowrap\">"
                    "<do type=\"accept\" label=\"Sig\">"
                      "<go href=\"%s\"/>"
                      "</do>"
                    "<b>Clase Av.:</b>"
                    "<select name=\"claseav\">",i,url);
            strcat (texto,cadena);
            for ( z=0 ; z<nclaseav ; z++) {
```

```
                    sprintf(cadena,"<option
value=\"%s\">%s</option>",claseav[z],claseav[z]);
                strcat (texto,cadena);
               }
             sprintf(cadena,"</select></p></card>");
               strcat (texto,cadena);
              }
            sprintf(cadena,"</wml>");
            strcat (texto,cadena);
           }
         }
    DigestAddDeck(d, "",texto, NULL);
    output = DigestSerialize(d, &outputLength);

    OutputBuffer(output,outputLength);

    /*
    The caller of DigestSerialize is responsible for deleting the
output buffer
    after using it
    */
    free(output);

    /* Delete the digest object */
    DigestDestruct(d);
}


int errorWap (int codigoerror,char *texto)

{
  Enlace     *enlace ;
  char url[500];

  enlace = new Enlace (INICIO) ;
  sprintf(url,"%s",enlace->DameCadena ());
  delete enlace ;
  switch(codigoerror) {
             case ERROR_CONFIGURACION:
                 sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "Fichero de <b>configuracion</b>"
                       " no accesible o con formato incorrecto.<br/>"
                       "</p></card></wml>",url);
                 break;
             case ERROR_ORACLE:
                 sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
```

```
                        "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "Error de acceso a la base de datos.<br/>"
                       "<b>Mensaje Oracle:</b><br/>"
                       "%s</p></card></wml>",url,errorSQL ());
                break;
            case ERR_EQLIN_NOTFOUND:
                sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p>"
                       "<b>-No se encontraron equipos/lineas asociados
a la averia.</b><br/>"
                       "<b>-Posible averia sin equipos.</b><br/>"
                       "<b>-Posibles equipos de averia no catalagodos
en GA.</b><br/>"
                       "</p></card></wml>",url);
                break;
            default: sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "<b>ERROR</b>"
                       "</p></card></wml>",url);
        }
  return 0;
}
```

franqueoWap2.cgi

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <parametros.h>

extern "C" {
#include <webconf.h>
#include <auxldap.h>
```

```
#include <auxsql.h>
#include "auxars.h"
#include "digest.h"
          }

#define EQUIPO '0'
#define LINEA '1'

#define ERROR_CONFIGURACION 1
#define ERROR_ORACLE        2
#define ERR_EQLIN_NOTFOUND  3

int errorWap (int codigoerror,char *texto);

void OutputBuffer (char *buffer, int length)
{
      int i;

      i=0;
      while (i < length) {
            printf("%c",*(buffer+i));
            i++;
      }
}

main ()
{
    Digest d;
    char *output;
    int outputLength;
    char texto[2000],
         url[500],
         cadena[500];
    char *act=NULL,
         *tec=NULL,
         *telef=NULL,
         *var=NULL;

   Parametros *parametros ;
   Enlace     *enlace ;

   char *cat, *eqlin, *claseav, **nomav;
   int nnomav=0,i;
   char literal[25];

   /* Create the Digest object */
    d = DigestConstruct();

    /*
    Add any needed sub parts to the digest
    In this case, add the deck to the displayed when this digest is
requested
    and add the Alert to send to the phone
    */

    var = getenv ("QUERY_STRING");
    parametros = new Parametros (var) ;
    act = parametros->DameValor (CAMPO_ACT) ;
    tec = parametros->DameValor (CAMPO_USER) ;
```

```
    telef = parametros->DameValor (CAMPO_TFNO) ;
    eqlin = parametros->DameValor (CAMPO_EQLIN) ;
    claseav = parametros->DameValor (CAMPO_CLASEAV) ;
    cat = parametros->DameValor (CAMPO_CATEGORIA) ;

    if(leewebconf()) errorWap(ERROR_CONFIGURACION,texto);
   else {
     if(conecta()) errorWap(ERROR_ORACLE,texto);
     else {
         switch(*cat) {
             case LINEA:
                 nnomav=lfranqnomav(eqlin,claseav,&nomav);
                 strcpy(literal,"Nombre averia");
                 break;
             case EQUIPO:
                 nnomav=lfranqeqins(eqlin,&nomav);
                 strcpy(literal,"Eq. instalado");
                 break;
            }
         if (nnomav == 0) errorWap(ERR_EQLIN_NOTFOUND,texto);
         else {
           if (nnomav < 0) errorWap(ERROR_ORACLE,texto);
           else {
             enlace = new Enlace ();
             enlace->InsertarCampo (CAMPO_ACT, act);
             enlace->InsertarCampo (CAMPO_USER,tec);
             enlace->InsertarCampo (CAMPO_TFNO,telef);
             enlace->InsertarCampo (CAMPO_EQLIN,eqlin);
             enlace->InsertarCampo (CAMPO_CLASEAV,claseav);
             enlace->InsertarCampo (CAMPO_CATEGORIA,cat);
             enlace->InsertarCampo (CAMPO_NOMBREAV,"$(nomav)");
             enlace->CodificaURL();
             sprintf(url,"%s%s",CGI_FRANQUEOWAP3,enlace->DameCadena
());
             delete enlace ;
             sprintf(texto,"<wml>"
             "<head><meta http-equiv=\"Cache-Control\" content=\"max-
age=0\"/></head>"
             "<card>"
             "<do type=\"accept\" label=\"Sig\">"
              "<go href=\"%s\">"
              "</go>"
             "</do>"
             "<p>"
             "<b>Tec: </b><i>%s</i><br/>"
             "<b>Av: </b><i>%s</i><br/>"
             "<b>Equipo/Linea: </b><i>%s</i><br/>"
             "<b>Clase Av.: </b><i>%s</i><br/>"
             "<b>%s:</b></p>"
             "<p mode=\"nowrap\">"
             "<select
name=\"nomav\">",url,tec,telef,eqlin,claseav,literal);
             for ( i=0 ; i<nnomav ; i++) {
             sprintf(cadena,"<option
value=\"%s\">%s</option>",nomav[i],nomav[i]);
             strcat (texto,cadena);
             }
             sprintf(cadena,"</select></p></card></wml>");
             strcat (texto,cadena);
```

```
            }
          }
        }
      }
    DigestAddDeck(d, "",texto, NULL);
    output = DigestSerialize(d, &outputLength);

    OutputBuffer(output,outputLength);

    /*
    The caller of DigestSerialize is responsible for deleting the
output buffer
    after using it
    */
    free(output);

    /* Delete the digest object */
    DigestDestruct(d);
}

int errorWap (int codigoerror,char *texto)

{
  Enlace     *enlace ;
  char url[500];

  enlace = new Enlace (INICIO) ;
  sprintf(url,"%s",enlace->DameCadena ());
  delete enlace ;
  switch(codigoerror) {
             case ERROR_CONFIGURACION:
                 sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "Fichero de <b>configuracion</b>"
                       " no accesible o con formato incorrecto.<br/>"
                       "</p></card></wml>",url);
                 break;
             case ERROR_ORACLE:
                 sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "Error de acceso a la base de datos.<br/>"
                       "<b>Mensaje Oracle:</b><br/>"
                       "%s</p></card></wml>",url,errorSQL ());
                 break;
             case ERR_EQLIN_NOTFOUND:
```

```
                sprintf(texto,"<wml>"
                      "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                      "</head>"
                      "<card>"
                      "<do type=\"accept\" label=\"Inicio\">"
                        "<go href=\"%s\"/>"
                      "</do>"
                      "<p>"
                      "<b>Falta de datos</b> de franqueo en GA.<br/>"
                      "-Posible falta de equipos instalados
relacionados.<br/>"
                      "-Posible falta de nombres de averias
relacionadas.<br/>"
                      "</p></card></wml>",url);
                break;
             default: sprintf(texto,"<wml>"
                      "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                      "</head>"
                      "<card>"
                      "<do type=\"accept\" label=\"Inicio\">"
                        "<go href=\"%s\"/>"
                      "</do>"
                      "<p align=\"center\">"
                      "<b>ERROR</b>"
                      "</p></card></wml>",url);
         }
  return 0;
}
```

franqueoWap3.cgi

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <parametros.h>

extern "C" {
#include <webconf.h>
#include <auxldap.h>
#include <auxsql.h>
#include "auxars.h"
#include "digest.h"
           }

#define EQUIPO '0'
#define LINEA '1'

#define ERROR_CONFIGURACION 1
#define ERROR_ORACLE        2

int errorWap (int codigoerror,char *texto);
```

```
void OutputBuffer (char *buffer, int length)
{
      int i;

      i=0;
      while (i < length) {
            printf("%c",*(buffer+i));
            i++;
      }
}

main ()
{
   Digest d;
   char *output;
   int outputLength;
   char texto[2000],
        cadena[500],
        url[500];
   char *act=NULL,
        *tec=NULL,
        *telef=NULL,
        *var=NULL;

   Parametros *parametros ;
   Enlace     *enlace ;

   char *cat,*eqlin, *claseav,*nomav;
   char literal[25],buf[100];
   long seg=3600,metros=1000;
   int  i;

   /* Create the Digest object */
    d = DigestConstruct();

    /*
    Add any needed sub parts to the digest
    In this case, add the deck to the displayed when this digest is
requested
    and add the Alert to send to the phone
    */

    var = getenv ("QUERY_STRING");
    parametros = new Parametros (var) ;
    act = parametros->DameValor (CAMPO_ACT) ;
    tec = parametros->DameValor (CAMPO_USER) ;
    telef = parametros->DameValor (CAMPO_TFNO) ;
    eqlin = parametros->DameValor (CAMPO_EQLIN) ;
    claseav = parametros->DameValor (CAMPO_CLASEAV) ;
    nomav = parametros->DameValor (CAMPO_NOMBREAV) ;
    cat = parametros->DameValor (CAMPO_CATEGORIA) ;

    if(leewebconf()) errorWap(ERROR_CONFIGURACION,texto);
    else {
     if(conecta()) errorWap(ERROR_ORACLE,texto);
     else {
       switch(*cat) {
          case LINEA:
            strcpy(literal,"Nombre averia");
```

```
            break;
          case EQUIPO:
            strcpy(literal,"Eq. instalado");
            break;
          }
       if((i=datosdesplazam(act,&seg,&metros))!=0)
errorWap(ERROR_ORACLE,texto);
       else {
         enlace = new Enlace ();
         enlace->InsertarCampo (CAMPO_ACT,act);
         enlace->InsertarCampo (CAMPO_USER,tec);
         enlace->InsertarCampo (CAMPO_TFNO,telef);
         enlace->InsertarCampo (CAMPO_EQLIN,eqlin);
         enlace->InsertarCampo (CAMPO_CLASEAV,claseav);
         switch(*cat) {
             case LINEA:
                 enlace->InsertarCampo (CAMPO_EQINS,"");
                 enlace->InsertarCampo (CAMPO_NOMBREAV,nomav);
                 break;
             case EQUIPO:
                 enlace->InsertarCampo (CAMPO_EQINS,nomav);
                 enlace->InsertarCampo (CAMPO_NOMBREAV,"");
                 break;
             }
         sprintf(cadena,"%.3f",metros/1000.0);
         enlace->InsertarCampo (CAMPO_CATEGORIA,cat);
         enlace->InsertarCampo (CAMPO_NMF,"$(nmf)");
         enlace->InsertarCampo (CAMPO_DHORAS,"$(horas)");
         enlace->InsertarCampo (CAMPO_DMIN,"$(minutos)");
         enlace->InsertarCampo (CAMPO_DKM,"$(Km)");
         enlace->CodificaURL();
         sprintf(url,"%s%s",CGI_FRANQUEARWAP,enlace->DameCadena ());
         delete enlace;
         sprintf(texto,"<wml>"
               "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/></head>"
               "<card id=\"card1\">"
               "<do type=\"accept\" label=\"Sig\">"
                 "<go href=\"#card2\">"
                 "</go>"
               "</do>"
               "<p>"
               "<b>Tec: </b><i>%s</i><br/>"
               "<b>Av: </b><i>%s</i><br/>"
               "<b>Equipo/Linea: </b><i>%s</i><br/>"
               "<b>Clase Av.: </b><i>%s</i><br/>"
               "<b>%s: </b><i>%s</i><br/></p>"
               "</card>"
               "<card id=\"card2\" title=\"Datos Franqueo:\"
newcontext=\"true\" "
               "ordered=\"false\">"
               "<do type=\"accept\" label=\"Franq\">"
                 "<go href=\"%s\">"
                 "</go>"
               "</do>"
               "<p>"
               "<b>NumModFac:</b>"
               "<input title=\"NumModFac\" name=\"nmf\" type=\"text\"
value=\"\" "
```

```
                "format=\"AANNNNN\" emptyok=\"false\"/>"
                "<b>Horas:</b>"
                "<input title=\"Horas\" name=\"horas\" type=\"text\"
value=\"%s\" "
                "format=\"N2N\" emptyok=\"false\"/>"
                "<b>Minutos:</b>"
                "<input title=\"Minutos\" name=\"minutos\"
type=\"text\" value=\"%s\" "
                "format=\"N2N\" emptyok=\"false\"/>"
                "<b>Kilometros:</b>"
                "<input title=\"Km\" name=\"Km\" type=\"text\"
value=\"%s\" "
                "format=\"N4N\" emptyok=\"false\"/>"
                "</p>"

"</card></wml>",tec,telef,eqlin,claseav,literal,nomav,url,

lltostr((seg/60)/60,buf),lltostr((seg/60)%60,buf),cadena);
            }
          }
        }
    DigestAddDeck(d, "",texto, NULL);
    output = DigestSerialize(d, &outputLength);

    OutputBuffer(output,outputLength);

    /*
    The caller of DigestSerialize is responsible for deleting the
output buffer
    after using it
    */
    free(output);

    /* Delete the digest object */
    DigestDestruct(d);
}


int errorWap (int codigoerror,char *texto)

{
  Enlace     *enlace ;
  char url[500];

  enlace = new Enlace (INICIO) ;
  sprintf(url,"%s",enlace->DameCadena ());
  delete enlace ;
  switch(codigoerror) {
             case ERROR_CONFIGURACION:
                 sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "Fichero de <b>configuracion</b>"
```

```
                        " no accesible o con formato incorrecto.<br/>"
                        "</p></card></wml>",url);
                break;
            case ERROR_ORACLE:
                sprintf(texto,"<wml>"
                        "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                        "</head>"
                        "<card>"
                        "<do type=\"accept\" label=\"Inicio\">"
                          "<go href=\"%s\"/>"
                        "</do>"
                        "<p align=\"center\">"
                        "Error de acceso a la base de datos.<br/>"
                        "<b>Mensaje Oracle:</b><br/>"
                        "%s</p></card></wml>",url,errorSQL ());
                break;
            default: sprintf(texto,"<wml>"
                        "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                        "</head>"
                        "<card>"
                        "<do type=\"accept\" label=\"Inicio\">"
                          "<go href=\"%s\"/>"
                        "</do>"
                        "<p align=\"center\">"
                        "<b>ERROR</b>"
                        "</p></card></wml>",url);
        }
  return 0;
}
```

franquearWap.cgi

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <defines.h>
#include <salida.h>
#include <enlace.h>
#include <parametros.h>
#include <fstream.h>
#include <ctype.h>

extern "C" {
#include <webconf.h>
```

```
#include <auxldap.h>
#include <auxsql.h>
#include "auxars.h"
#include "digest.h"
          }

#define EQUIPO 0
#define LINEA 1

#define ERROR_CONFIGURACION 1
#define ERROR_ORACLE        2
#define ERROR_ARS           3

int errorWap (int codigoerror,char *texto);

void OutputBuffer (char *buffer, int length)
{
      int i;

      i=0;
      while (i < length) {
            printf("%c",*(buffer+i));
            i++;
      }
}

main ()
{
    Digest d;
    char *output;
    int outputLength;
    char texto[2000],cadena[500];
    char url[500];
    char *var=NULL,
         *tec=NULL,
         *act=NULL,
         *cat=NULL,
         *eqlin,*claseav, *eqins=NULL, *nomav=NULL,
         *nmf;
    int  i;
    long horas, min, icat;
    double km;

    Parametros *parametros ;
    Enlace     *enlace ;

    /* Create the Digest object */
    d = DigestConstruct();

    /*
    Add any needed sub parts to the digest
    In this case, add the deck to the displayed when this digest is
requested
    and add the Alert to send to the phone
    */

    var = getenv ("QUERY_STRING");
    parametros = new Parametros (var) ;
    act = parametros->DameValor (CAMPO_ACT) ;
```

```
    tec = parametros->DameValor (CAMPO_USER) ;
    cat = parametros->DameValor (CAMPO_CATEGORIA) ;
    icat=atol(cat);
    eqlin = parametros->DameValor (CAMPO_EQLIN) ;
    claseav = parametros->DameValor (CAMPO_CLASEAV) ;
    if(icat==EQUIPO) eqins=parametros->DameValor(CAMPO_EQINS);
    else nomav=parametros->DameValor(CAMPO_NOMBREAV);
    nmf=parametros->DameValor(CAMPO_NMF);
    horas=atol(parametros->DameValor(CAMPO_DHORAS));
    min=atol(parametros->DameValor(CAMPO_DMIN));
    km=atof(parametros->DameValor(CAMPO_DKM));

    if(leewebconf() || !isdigit(*tec))
errorWap(ERROR_CONFIGURACION,texto);
    else {

i=franqueoTecARS(webConf.loginAR,webConf.passwdAR,webConf.servAR,
                         act,tec,eqlin,claseav,eqins,nomav,
                         horas,min,km,(isalpha(*nmf))?nmf:NULL);
         if ((i != 0) && (i != 1)) errorWap(ERROR_ARS,texto);
         else {
             enlace = new Enlace (INICIO) ;
             sprintf(url,"%s",enlace->DameCadena ());
             delete enlace ;
             sprintf(texto,"<wml>"
             "<head><meta http-equiv=\"Cache-Control\" content=\"max-
age=0\"/></head>"
             "<card>"
             "<do type=\"accept\" label=\"Inicio\">"
               "<go href=\"%s\"/>"
             "</do>"
             "<p align=\"center\"><b><i>Franqueo
Correcto</i></b></p>",url);
             switch(i) {
               case 0: // Franqueada correctamente
                 break;
               case 1: // Franqueada correctamente con warning
                 sprintf(cadena,"<p align=\"left\">"
                         "<b>Advertencia:</b><br/>"
                         "%s</p>",errorARS(1));
                 strcat (texto,cadena);
                 break;
               };
             sprintf(cadena,"</card></wml>");
             strcat (texto,cadena);
          }
       }
    DigestAddDeck(d, "",texto, NULL);
    output = DigestSerialize(d, &outputLength);

    OutputBuffer(output,outputLength);

    /*
    The caller of DigestSerialize is responsible for deleting the
output buffer
    after using it
    */
    free(output);
```

```
    /* Delete the digest object */
    DigestDestruct(d);
}


int errorWap (int codigoerror,char *texto)

{
  Enlace     *enlace ;
  char url[500];

  enlace = new Enlace (INICIO) ;
  sprintf(url,"%s",enlace->DameCadena ());
  delete enlace ;
  switch(codigoerror) {
             case ERROR_CONFIGURACION:
                 sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "Fichero de <b>configuracion</b>"
                       " no accesible o con formato incorrecto.<br/>"
                       "</p></card></wml>",url);
                 break;
             case ERROR_ARS:
                 sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "Error de acceso a ARS.<br/>"
                       "<b>Mensaje ARS:</b></p>"
                       "<p
align=\"left\">%s</p></card></wml>",url,errorARS(1));
                 break;
             default: sprintf(texto,"<wml>"
                       "<head><meta http-equiv=\"Cache-Control\"
content=\"max-age=0\"/>"
                       "</head>"
                       "<card>"
                       "<do type=\"accept\" label=\"Inicio\">"
                         "<go href=\"%s\"/>"
                       "</do>"
                       "<p align=\"center\">"
                       "<b>ERROR</b>"
                       "</p></card></wml>",url);
         }
  return 0; }
```

digest.C (librerías suministradas con el simulador UP.SDK propiedad de Phone.com).

```
/*************************************************************************
******
*
*   (c) Copyright 1995-1999, Phone.com, Inc.  All Rights Reserved.
*
*
*   Subject to the terms and conditions of the SDK LICENSE AGREEMENT,
*   Phone.com, Inc. hereby grants you a license to use the UP.SDK(TM)
*   software and its related documentation.
*
*   The following are trademarks or registered trademarks of
Phone.com,
*   Inc. All Rights Reserved: Powered-By UP(TM), Powered-By UP
logo((TM)),
*   Visual Voicemail(TM), Phone.com(TM) name, Phone.com logo((TM)),
*   uplanet.com(R), UP.Access(TM), UP.Admin(TM), UP.Alert(TM),
*   UP.Application(TM), UP.Bookmarks(TM), UP.Browser(TM), UP.Cast(TM),
*   UP.Dev(TM), UP.Developer(TM), UP.Device(TM), UP.Directory(TM),
*   UP.Encrypt(TM), UP.Fax(TM), UP.HomePage(TM), UP.Link(TM),
UP.Mail(TM),
*   UP.Market(TM), UP.Monitor(TM), UP.Notify(TM), UP.Organizer(TM),
*   UP.Pager(TM), UP.Phone(R), UP.PIM(TM), UP.SDK(TM),
UP.Simulator(TM),
*   UP.Secure(TM), UP.Transact(TM), UP.University(TM), UP.Web(TM),
*   and any other term carrying the "UP." prefix. All other brand,
*   company and product names are used for identification purposes
only and
*   may be trademarks that are the sole property of their respective
owners.
*
*************************************************************************
*****/

/*************************************************************************
*
* Digest
*
* This file implements functions to create a Digest and add
* different sub parts (such as a Deck, Alert, a Cache Invalidation
request)
* to the Digest and to create a single character stream of the Digest
* as an ASCII MIME message for sending as part of a HTTP response.
*
*************************************************************************/

#include <stdlib.h>
#include <string.h>
#include "strbuf.h"
#include "msgenty.h"
#include "digest.h"

const static char* CHARSET_KEY = "charset=";
const static char* CONTENTTYPE_CHARSET = "";
/*static char* CONTENTTYPE_CHARSET = "charset=utf-8";*/
```

```
const static char* DTD_WML = "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD
WML 1.1//EN\" \"http://www.wapforum.org/DTD/wml_1.1.xml\">";
const static char* DTD_HDML = "";

const static char* CONTENTTYPE_WML = "text/x-wap.wml";
const static char* CONTENTTYPE_HDML = "text/x-hdml";

int DECKTYPE;

struct _digest
{
      MultipartMsg m_multipartMsg;
            /* This multi part Msg object will hold the Digest
components */
};

/* Private function declarations */
char *GetContentType(char* charset);
const char *GetDTD();

/*********************************************************************
*
* DigestConstruct
*
* Creates a Digest structure and initializes the multipart to be of
type
* multipart/mixed
*
*********************************************************************/
Digest DigestConstruct()
{
      Digest d = (Digest)malloc(sizeof(struct _digest));
      d->m_multipartMsg = MultipartMsgConstruct();
      MultipartMsgAddHeader(d->m_multipartMsg, "Content-type",
"multipart/mixed");

      return d;
}

/*********************************************************************
*
* DigestDestruct
*
* Deletes the Digest structure
*
*********************************************************************/
void DigestDestruct(Digest d)
{
      if (!d)
            return;

      MultipartMsgDestruct(d->m_multipartMsg);
      free(d);
}

/*********************************************************************
*
* DigestAddCacheOp
*
```

```
* Adds a Cache operation msg entity to the digest
*
*****************************************************************/
void DigestAddCacheOp(Digest d, char* operation, char* url)
{
      StringBuf body;
      StringBuf encodedString;
      MsgEntity cacheOpSubPart;

      if (!d || !operation)
            return;

      body = StringBufConstruct();

      StringBufAssignString(body,"OpCode=");
      encodedString = StringBufHTTPEscapeString(operation);
      StringBufAppendStringBuf(body,encodedString);
      StringBufDestruct(encodedString);

      StringBufAppendString(body,"&Operand=");
      encodedString = StringBufHTTPEscapeString(url);
      StringBufAppendStringBuf(body,encodedString);
      StringBufDestruct(encodedString);

      StringBufAppendString(body,"\n");

      cacheOpSubPart = MsgEntityConstruct();
      MsgEntityAddHeader(cacheOpSubPart, "Content-type",
"application/x-up-cacheop");
      MsgEntitySetBody(cacheOpSubPart, StringBufToChar(body));
      MultipartMsgAddSubPart(d->m_multipartMsg, cacheOpSubPart);

      StringBufDestruct(body);

}

/*****************************************************************
*
* DigestAddDeck
*
* Adds the HDML deck to the Digest
*
*****************************************************************/
void DigestAddDeck(Digest d, char* url, char* deck, char* charset)
{
      MsgEntity deckSubPart;
      char * contentType;
      char *dtd;
      char *output;

      if (!d || !deck)
            return;

      dtd = (char *)GetDTD();
      output = (char*)malloc(strlen(deck) + strlen(dtd) + 1 + 1);
      strcpy(output,dtd);
      strcat(output,"\n");
      strcat(output,deck);
```

```
	contentType = GetContentType(charset);

	deckSubPart = MsgEntityConstruct();
	MsgEntityAddHeader(deckSubPart, "Content-type", contentType);
	MsgEntityAddHeader(deckSubPart, "Content-location", url);
	MsgEntitySetBody(deckSubPart, output);

	MultipartMsgAddSubPart(d->m_multipartMsg, deckSubPart);

	free(contentType);
	free(output);

}

/*********************************************************************
*
* DigestAddImage
*
* Adds an image to the Digest
*
*********************************************************************/
void DigestAddImage(Digest d, char * type, char* url, char* image, int
imageSize)
{
	MsgEntity imageSubPart;

	if (!d || !type || !url)
		return;

	imageSubPart = MsgEntityConstruct();
	MsgEntityAddHeader(imageSubPart, "Content-type", type);
	MsgEntityAddHeader(imageSubPart, "Content-location", url);
	MsgEntitySetBinaryBody(imageSubPart, image, imageSize);

	MultipartMsgAddSubPart(d->m_multipartMsg, imageSubPart);

}

/*********************************************************************
*
* DigestAddCookie
*
* Adds a cookie in HTTP header format to the Digest
* Replaces the header if one already exists with the same name
*
*********************************************************************/
void DigestAddCookie(Digest d, char* cookie)
{
	if (!d)
		return;

	MultipartMsgAddHeader(d->m_multipartMsg, "Set-Cookie", cookie);
}

/*********************************************************************
*
* DigestAddHeader
*
* Adds the HTTP header to the Digest
```

```
* Replaces the header if one already exists with the same name
*
*****************************************************************/
void DigestAddHeader(Digest d, char* name, char* value)
{
      if (!d)
            return;

      MultipartMsgAddHeader(d->m_multipartMsg, name, value);
}

/*****************************************************************
*
* DigestAddAlert
*
* Add an alert to the Digest
*
*****************************************************************/
void DigestAddAlert(Digest d, char* alertType, char* URL, char* title,
char* charset)
{
      StringBuf body;
      StringBuf encodedString;
      MsgEntity alertSubPart;
      char contentType[255];

      if (!d || !URL || !title)
            return;

      body = StringBufConstruct();

      StringBufAssignString(body,"Type=");
      encodedString = StringBufHTTPEscapeString(alertType);
      StringBufAppendStringBuf(body,encodedString);
      StringBufDestruct(encodedString);

      StringBufAppendString(body,"&Title=");
      encodedString = StringBufHTTPEscapeString(title);
      StringBufAppendStringBuf(body,encodedString);
      StringBufDestruct(encodedString);

      StringBufAppendString(body,"&URL=");
      encodedString = StringBufHTTPEscapeString(URL);
      StringBufAppendStringBuf(body,encodedString);
      StringBufDestruct(encodedString);

      StringBufAppendString(body,"\n");

      strcpy(contentType, "application/x-up-alert; ");
      if (charset && strlen(charset) > 0)
      {
            /* Add the key if user doesn't pass it in. */
            if (strncasecmp(charset, CHARSET_KEY, strlen(CHARSET_KEY))
!= 0)
                  strcat(contentType, CHARSET_KEY);

            strcat(contentType, charset);
      }
      else
```

```
            strcat(contentType, CONTENTTYPE_CHARSET);

      alertSubPart = MsgEntityConstruct();
      MsgEntityAddHeader(alertSubPart, "Content-type", contentType);
      MsgEntitySetBody(alertSubPart, StringBufToChar(body));
      MultipartMsgAddSubPart(d->m_multipartMsg, alertSubPart);

      StringBufDestruct(body);

}

/*****************************************************************
*
* DigestSerialize
*
* Serialize the digest structure to a character stream. Caller
* is responsible for freeing the returned char pointer
*
*****************************************************************/
char *DigestSerialize(Digest d, int* outputLength)
{

      if (!d)
            return "";

      return MultipartMsgSerialize(d->m_multipartMsg, outputLength);
}

/*****************************************************************
*
* Private functions to retrieve the content type and DTD to use in the
decks
*
* Serialize the digest structure to a character stream. Caller
* is responsible for freeing the returned char pointer
*
*****************************************************************/
char *GetContentType(char* charset)
{
      char *contentType  = (char *)malloc(255);
      char *charsetToUse = (char *)malloc(255);

      charsetToUse[0] = 0;

      /* Copy the charset and format it properly */
      if (charset && strlen(charset) > 0)
      {
            /* Add the key if user doesn't prepend it. */
            if (strncasecmp(charset, CHARSET_KEY, strlen(CHARSET_KEY))
!= 0)
                  strcpy(charsetToUse, CHARSET_KEY);

            strcat(charsetToUse, charset);
      }
      else
            strcpy(charsetToUse, CONTENTTYPE_CHARSET);

      if (DECKTYPE != DECKTYPE_HDML)
            strcpy(contentType, CONTENTTYPE_WML);
```

```
	else
		strcpy(contentType, CONTENTTYPE_HDML);

	if (strlen(charsetToUse) > 0) {
		strcat(contentType, ";");
		strcat(contentType, charsetToUse);
	}

	free(charsetToUse);
	return contentType;
}

const char *GetDTD()
{
	if (DECKTYPE != DECKTYPE_HDML)
		return DTD_WML;
	else
		return DTD_HDML;
}
```

FICHERO DE CONFIGURACION

webconf.txt **(archivo de texto para configuración).**

```
ASIGNAAUTO=0
ASIGNAMAN=0
NTEC=10
NPR=10
LOGDB=aradmin
PASSWDDB=ar#admin#
SERVDB=GASICILIA
LOGAR=Demo
PASSWDAR=ga
SERVAR=sicilia
TFRANQUEO=1440
```

PRESUPUESTO

1. PRESUPUESTO

PRELIMINARES

- El presupuesto total del proyecto es la suma del presupuesto de contrata más el I.V.A. correspondiente y los honorarios facultativos. A su vez, el presupuesto de contrata consta del presupuesto de ejecución material, los gastos generales y financieros y el beneficio industrial.
- Para la realización del proyecto ha sido necesaria la participación de dos profesionales, un Ingeniero Superior de Telecomunicación y un mecanógrafo.
- La duración del proyecto se estima en 15 meses, incluyendo la mecanografía y el tratamiento de textos.
- Todas las cantidades presentes en este documento están expresadas en pesetas.

PRESUPUESTO DE EJECUCIÓN MATERIAL

COSTE DE LA MANO DE OBRA

RELACIÓN DE SALARIOS

- Ingeniero Superior de Telecomunicación

Sueldo mensual 226.300

Sueldo diario 7.300

- Mecanógrafo

Sueldo mensual 93.300

Sueldo diario 3000

RELACIÓN DE OBLIGACIONES SOCIALES

Vacaciones anuales retribuidas	7.50%
Indemnizaciones de despido	1.70%
Seguro de accidentes	2.80%
Subsidio de vejez	2.00%
Subsidio familiar	2.00%
Abono de días festivos	15.20%
Días de enfermedad	1.00%
Cuota sindical	2.70%
Plus de cargas familiares	4.20%
Gratificaciones extraordinarias	20.00%
Plus de carestía de vida	9.50%
Paro tecnológico	17.20%
Otros conceptos	5.00%
Total	**90,80%**

IMPORTE TOTAL DE LOS SALARIOS

- Ingeniero Superior de Telecomunicación

Salario diario	7.300
Cargas sociales	6.628
Salario total por día	13.928

Días empleados	400
Salario global	**5.571.200**

- Mecanógrafo

Salario diario	3.000
Cargas sociales	2.724
Salario total por día	5.724
Días empleados	30
Salario global	**171.720**

COSTE TOTAL DE LA MANO DE OBRA 5.742.920

COSTE DE LOS MATERIALES

GASTOS DE AMORTIZACIÓN

Se considerará una amortización del 12.5 % anual, tanto para los componentes hardware como para las licencias del software utilizado.

- Equipos hardware

Una máquina Sun E10000	80.000.000
Una estación de trabajo Sun Sparc 10	800.000
Una tarjeta de red Ethernet	6.000
Un ordenador personal PC	150.000

Nokia 9110 Communicator	120.000
Terminal móvil WAP	50.000
Días de uso	400
Impresora Láser	100.000
Días de uso	30
Amortización	**11.870.000**

- Licencias de software

S. O. Unís (Solaris)	30.000
Windows NT	15.000
Netscape Enterprise Server 3.62	60.000
ARS 3.2	250.000
Base de datos Oracle 8.0	4.000.000
Días de uso	400
Microsoft Word 2000	15.000
Días de uso	30
Amortización	**597.600**

IMPORTE TOTAL DE LOS MATERIALES

Diskettes	2.000
CDs	1.000
Material de oficina	25.000

Gastos de amortización	12.467.600
Factura de Teléfono móvil	70.000
COSTE TOTAL DE LOS MATERIALES	**12.565.600**

COSTE TOTAL DE EJECUCIÓN MATERIAL

Es la suma de del coste de la mano de obra más el coste de los materiales

Coste de la mano de obra	5.742.920
Coste de los materiales	12.565.600
COSTE TOTAL DE EJECUCIÓN MATERIAL	**18.308.520**

PRESUPUESTO DE CONTRATA

Esta formado por el presupuesto de ejecución material, más los gastos generales y el beneficio industrial.

Presupuesto de ejecución material	18.308.520
Gastos generales y financieros (16% de 18.308.520)	2.929.363
Beneficio industrial (6% de 18.308.520)	1.098.511
PRESUPUESTO DE CONTRATA	**22.336.394**

HONORARIOS FACULTATIVOS

Se aplica el coeficiente del 7% reducido en 0.8 en el tramo de 1 a 6 millones y en 0.65 en el tramo por encima de 6 millones.

- Tramo 0-1 millón

Coeficiente	0.07
Base imponible	1.000.000
Reducción	1
Total	**70000**

- Tramo 1-6 millones

Coeficiente	0.07
Base imponible	5.000.000
Reducción	0.8
Total	**280.000**

- Tramo +6 millones

Coeficiente	0.07
Base imponible	16.336.394
Reducción	0.65
Total	**743.305**

TOTAL DE HONORARIOS FACULTATIVOS: 1.093.305

PRESUPUESTO TOTAL

Presupuesto de contrata	22.336.394
I.V.A. (16%)	3.573.823
Honorarios facultativos	1.093.305
Total	**27.003.522**

El presupuesto total asciende a VEINTISIETE MILLONES TRES MIL QUINIENTAS VEINTIDÓS (27.003.522) PESETAS.

EL INGENIERO AUTOR DEL PROYECTO

Fdo: Aurelio Mariscal Ortiz

MADRID, a 14 de Marzo de 2001