3. DESARROLLO DEL PFC

Como ya se ha explicado lo que deseamos es implementar dos aplicaciones, un cliente y un servidor. Ambas aplicaciones se integrarán en un sistema de información cuya finalidad es proporcionar un servicio de recomendaciones para itinerarios de trasporte intermodal.

1) La aplicación cliente actuará como intermediario entre el usuario final y el servidor de recomendaciones. Resumidamente, las funciones del cliente serán:
2) Recibir las peticiones de los usuarios a través de un formulario web o mediante sms (telefonía móvil).
3) Establecer la comunicación con la aplicación servidor y enviarle los parámetros necesarios para calcular las recomendaciones (el cálculo de dichas recomendaciones no es ámbito del presente proyecto).
4) Recibir las recomendaciones que le devolverá la aplicación servidor y presentarlas al usuario final. Si el usuario hizo la petición por formulario web recibirá la/s recomendación/es mediante página web. Si, por el contrario, realizó la petición por SMS, recibirá la/s recomendación/es por sms.

La aplicación servidor será igualmente un intermediario entre la aplicación cliente y otra aplicación que será la que realmente calcule las recomendaciones. Abreviadamente, las funciones del servidor serán:

1) Permanecer a la espera de recibir peticiones de un cliente.
2) Recibir los parámetros de una petición mediante el protocolo de mensajes definido. A partir de dichos parámetros se llevará a cabo el cálculo de la/s recomendación/es.
3) Entregar los parámetros a la aplicación que hace el cálculo de recomendaciones.
4) Responder a la aplicación cliente con las recomendaciones calculadas mediante el protocolo de mensajes definido.

3.1. DEFINICIÓN DEL PROTOCOLO DE COMUNICACIÓN ENTRE CLIENTE Y SERVIDOR.

Como ya se ha explicado es necesario implementar un protocolo de comunicación entre un proceso cliente y un proceso servidor, dichos procesos podrán localizarse en la misma máquina o en máquinas diferentes. Mediante el protocolo que se desea definir, ambos procesos intercambiarán la información necesaria para proporcionar un servicio de recomendaciones de itinerarios para el transporte público de pasajeros.

Por lo tanto, el objetivo es que el proceso cliente establezca una comunicación sobre TCP/IP con el proceso servidor. Una vez establecida la comunicación el cliente enviará una serie de parámetros al servidor, el cual realizará el cálculo de recomendaciones con dichos parámetros. El servidor responderá a la petición del cliente con las recomendaciones calculadas. Todo esta comunicación se realizará a través de un protocolo definido para tal fin, y que marcará la pauta de las acciones a llevar a cabo por los procesos cliente y servidor.

3.1.1. Mensajes del Protocolo CLIENTE-SERVIDOR.

- P: mensaje de petición inicial. Es el primer mensaje enviado por el cliente hacia el servidor para indicarle que desea iniciar un servicio de cálculo de recomendaciones de itinerario.
- V: mensaje de asentimiento positivo. Es el mensaje con el que el servidor responderá a una petición inicial de recomendación, cuando esté dispuesto a realizar el servicio.
- N: mensaje de asentimiento negativo. Es el mensaje con el que el servidor responderá a una petición inicial de recomendación, cuando no esté dispuesto a realizar el servicio. Las causas pueden ser que el servidor esté saturado u otras que le impidan en ese momento realizar el cálculo de recomendaciones.
- R: mensaje de réplica o reply. Es el mensaje con el que el cliente responderá siempre tras

recibir un mensaje del servidor. Con este mensaje se consigue sincronizar las acciones del cliente y el servidor, ya que tras enviar un mensaje el servidor siempre se mantendrá a la espera de un reply para continuar con su cometido.

- OK: mensaje de parámetros correctos. Es el mensaje con el que el servidor indicará al cliente que los parámetros recibidos para el cálculo de la recomendación son correctos.
- NOK: mensaje de parámetros no correctos. Es el mensaje con el que el servidor indicará al cliente que los parámetros recibidos para el cálculo de la recomendación no son correctos.
- NI: mensaje de nuevo intento de cálculo de recomendación. Es el mensaje con el que el servidor indicará al cliente que el cálculo de la recomendación ha fallado y va ha realizar un nuevo intento de cálculo.
- IMP: mensaje de cálculo de recomendación imposible. Es el mensaje con el que el servidor indicará al cliente que el cálculo de la recomendación ha fallado y es imposible realizar un nuevo intento de cálculo.
- B: mensaje de fin del servicio o bye: Es el mensaje con el que el servidor indicará al cliente que el cálculo de la recomendación se ha realizado y ya ha terminado de enviarle los mensajes que contienen la recomendación. En el caso de que el servidor reciba unos parámetros correctos pero no encuentre recomendaciones para esa petición, el servidor lo indicará al cliente mediante dos mensajes: uno denominado EMPTY seguido de otro que indique la causa (ejemplo: no hay plazas disponibles).
- Mensajes de información: son los mensajes que pueden ser enviados en los dos sentidos de la comunicación. Contendrán los parámetros necesarios para el cálculo de recomendaciones, las recomendaciones calculadas, EMPTY y la causa por la que no se encontraron recomendaciones.

3.1.2.Diagrama de paso de mensajes del protocolo CLIENTE-SERVIDOR.

3.1.2.1. Cálculo de una Recomendación con éxito.

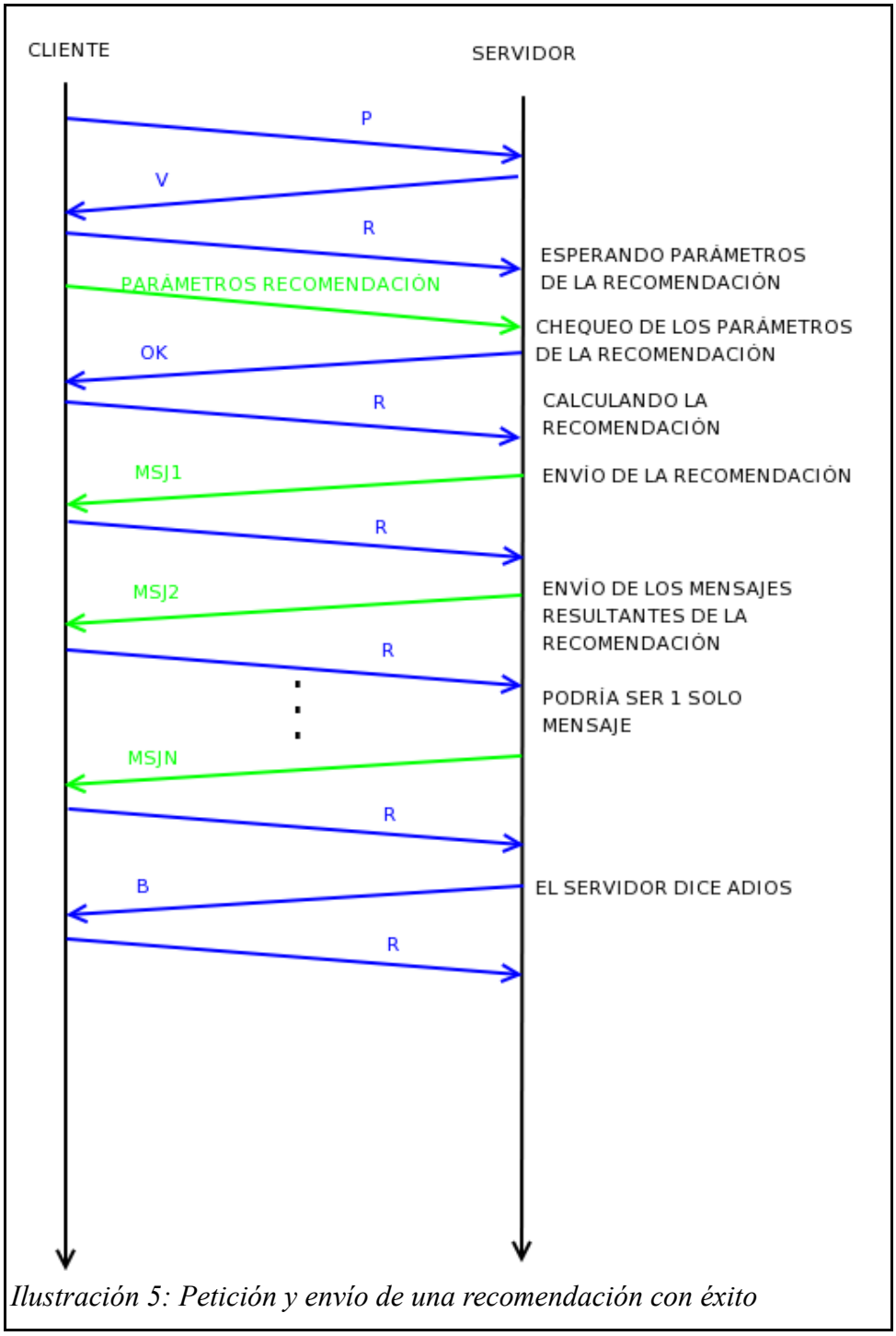

Ilustración 5: Petición y envío de una recomendación con éxito

3.1.2.2. Cálculo de una recomendación con éxito tras un nuevo intento.

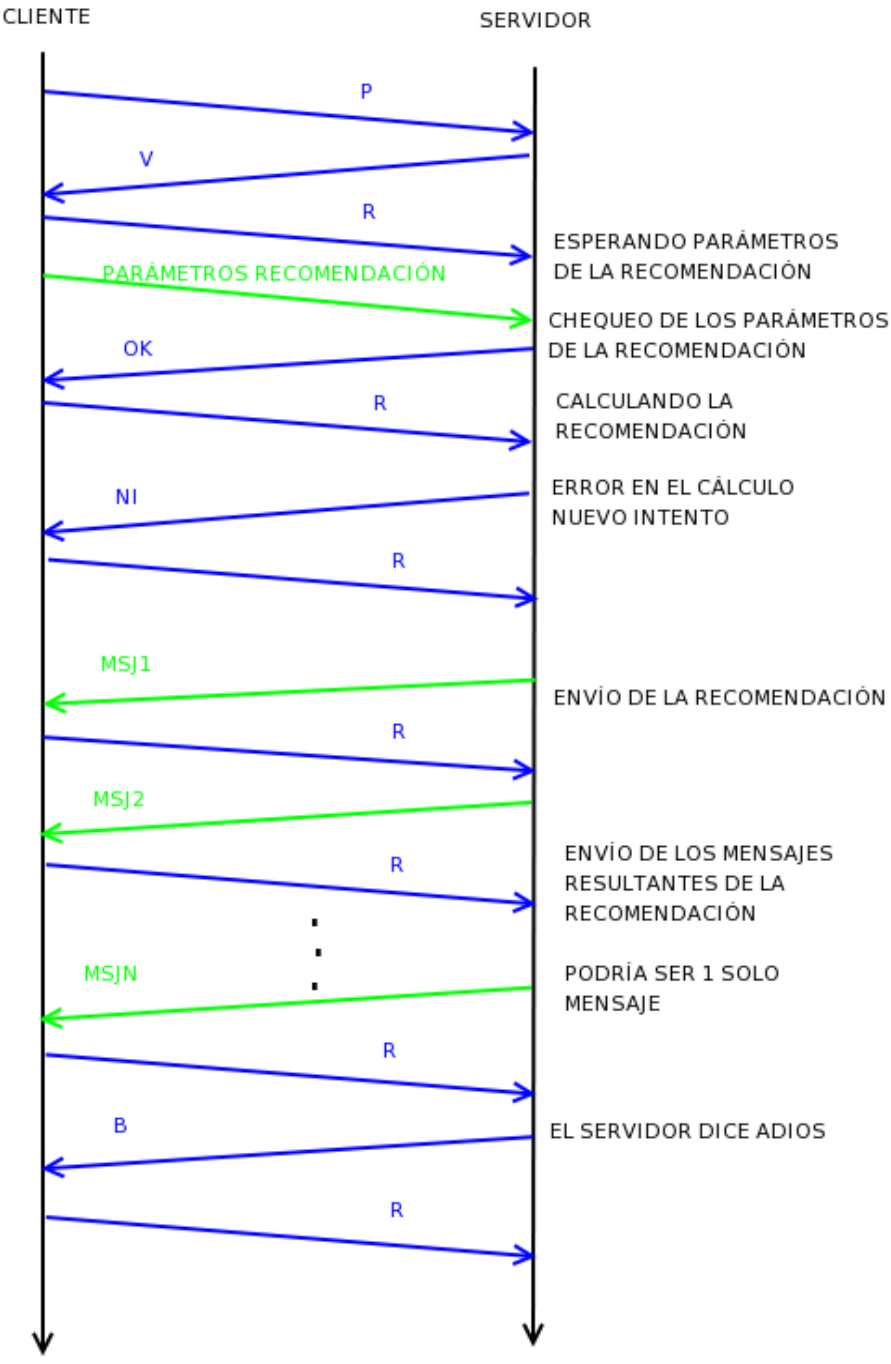

Ilustración 6: Cálculo de recomendación tras nuevo intento

3.1.2.3. Imposible calcular la recomendación.

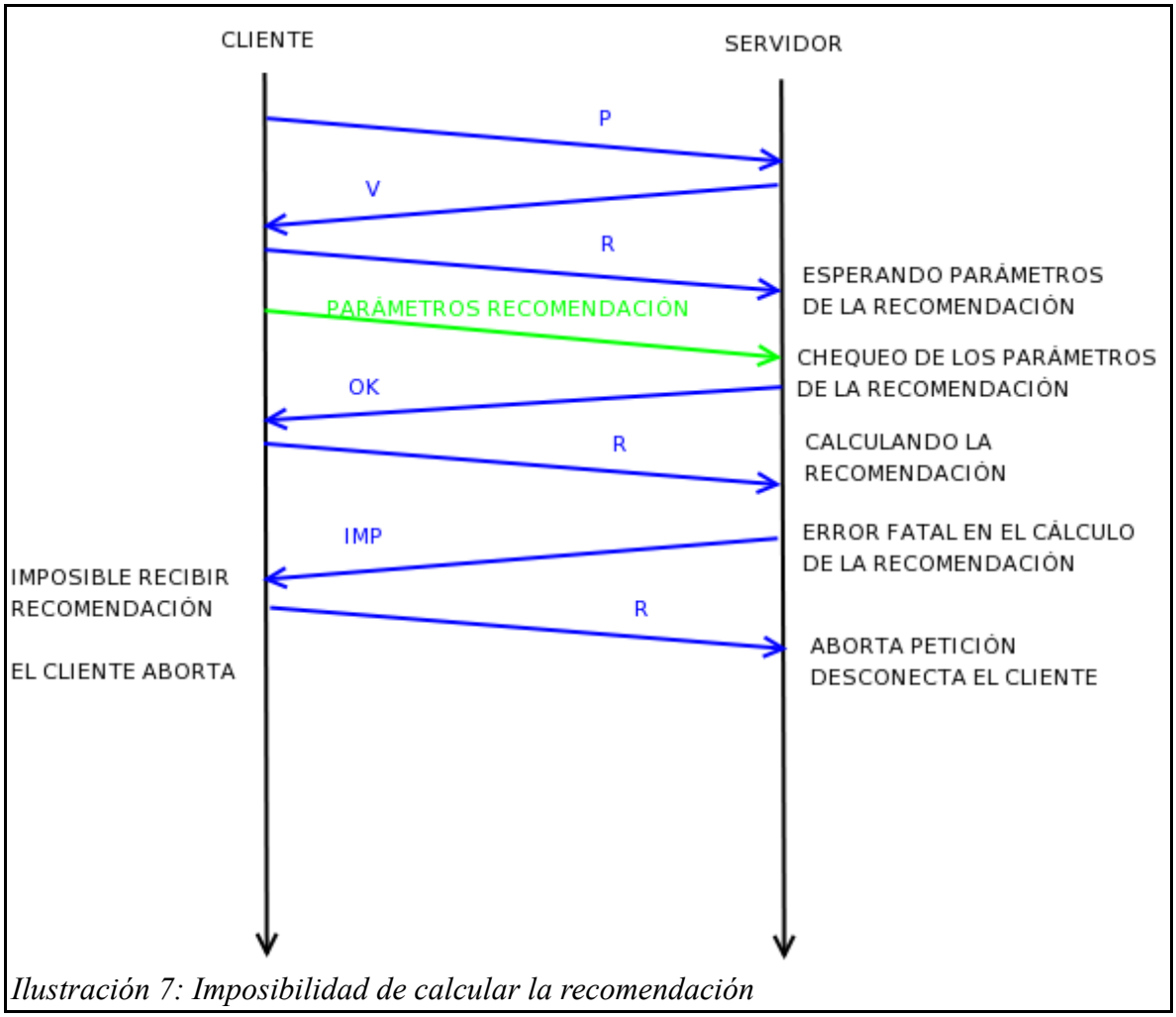

Ilustración 7: Imposibilidad de calcular la recomendación

En este caso el CLIENTE envía todos los parámetros correctamente pero debido a un fallo interno el SERVIDOR no puede calcular la recomendación y le indica al CLIENTE que es imposible calcular un recomendación.

3.1.2.4. Parámetros incorrectos para el cálculo de la recomendación.

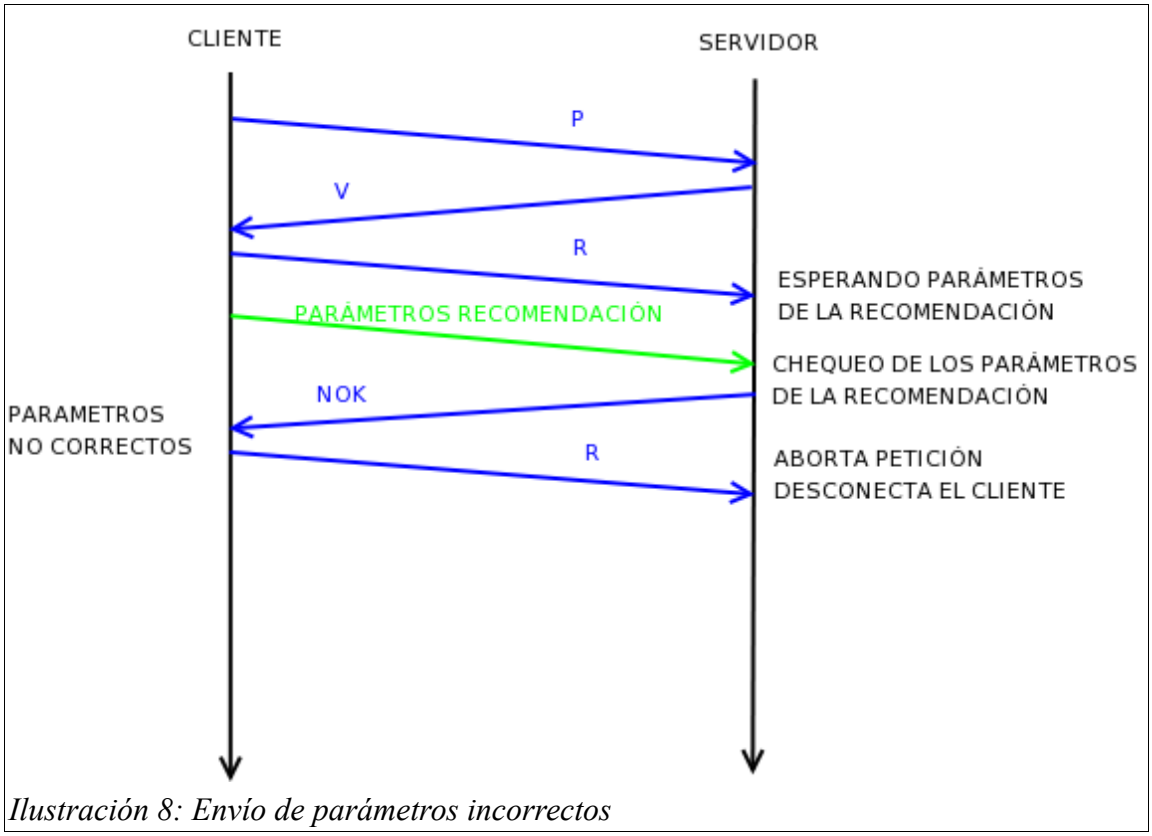

Ilustración 8: Envío de parámetros incorrectos

3.1.2.5. Imposible calcular la recomendación por rechazo del SERVIDOR.

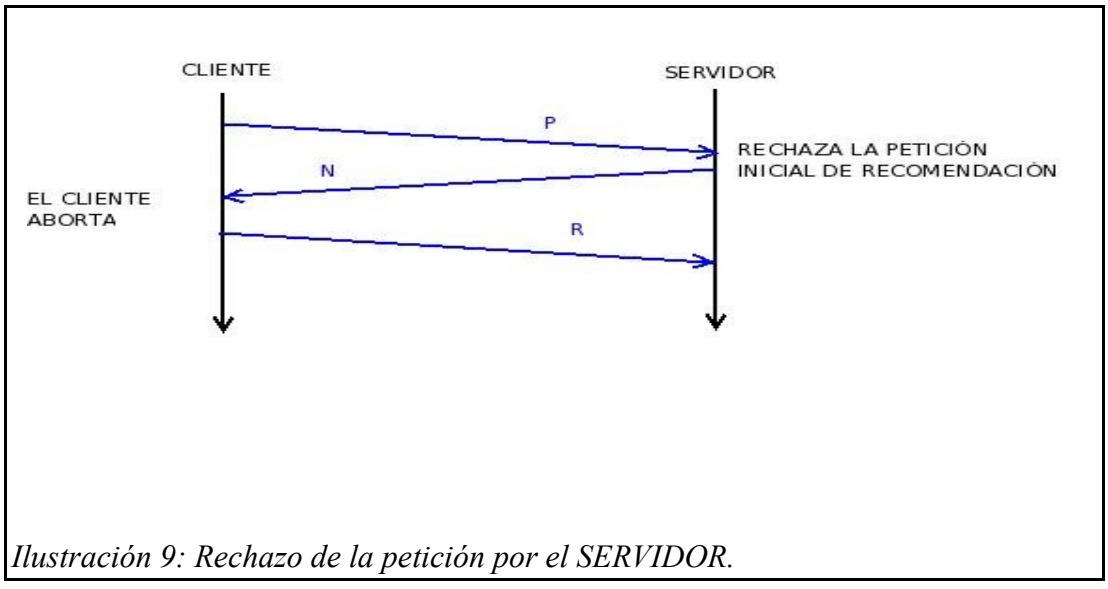

Ilustración 9: Rechazo de la petición por el SERVIDOR.

3.2. DISEÑO DE LA APLICACIÓN CLIENTE.

3.2.1. Diagrama de flujo de la aplicación CLIENTE.

- T_ACK: tiempo en segundos que permanece a la espera de un asentimiento tras el envío de una petición inicial.
- T_RC: tiempo en segundos que permanece a la espera de una respuesta tras el envío de los parámetros necesarios para el cálculo de las recomendaciones. Esta respuesta será OK o NOK.
- T_MSJ: tiempo en segundos que permanece a la espera de un mensaje procedente del servidor de recomendaciones. Estos mensajes serán las recomendaciones calculadas, la causa por la que no se obtuvo una recomendación o BYE.
- MAX_ESP: número máximo de veces que espera un asentimiento tras enviar una petición inicial.
- MAX_RC: número máximo de veces que espera una respuesta tras enviar los parámetros para el cálculo de la recomendación.
- V: asentimiento positivo.
- N: asentimiento negativo.
- CONT_PI: contador de intentos de espera de asentimiento.
- CONT_RC: contador de intentos de recibir respuesta tras enviar los parámetros.
- BYE: mensaje que marca el final de la conversación con el servidor de recomendaciones.
- MSJ: serán las recomendaciones calculadas, la causa por la que no se obtuvo una recomendación o BYE.

Este diagrama de flujo define la estructura de la aplicación cliente. Dicho servidor ha sido implementado en lenguaje C a partir de este diagrama de flujo. El código fuente del mismo se ha adjuntado en los anexos para su consulta. Dicho código fuente está comentado de forma exhaustiva y detallada para que sea fácil la comprensión del mismo y se sepa que es lo que realiza cada sentencia y función definida.

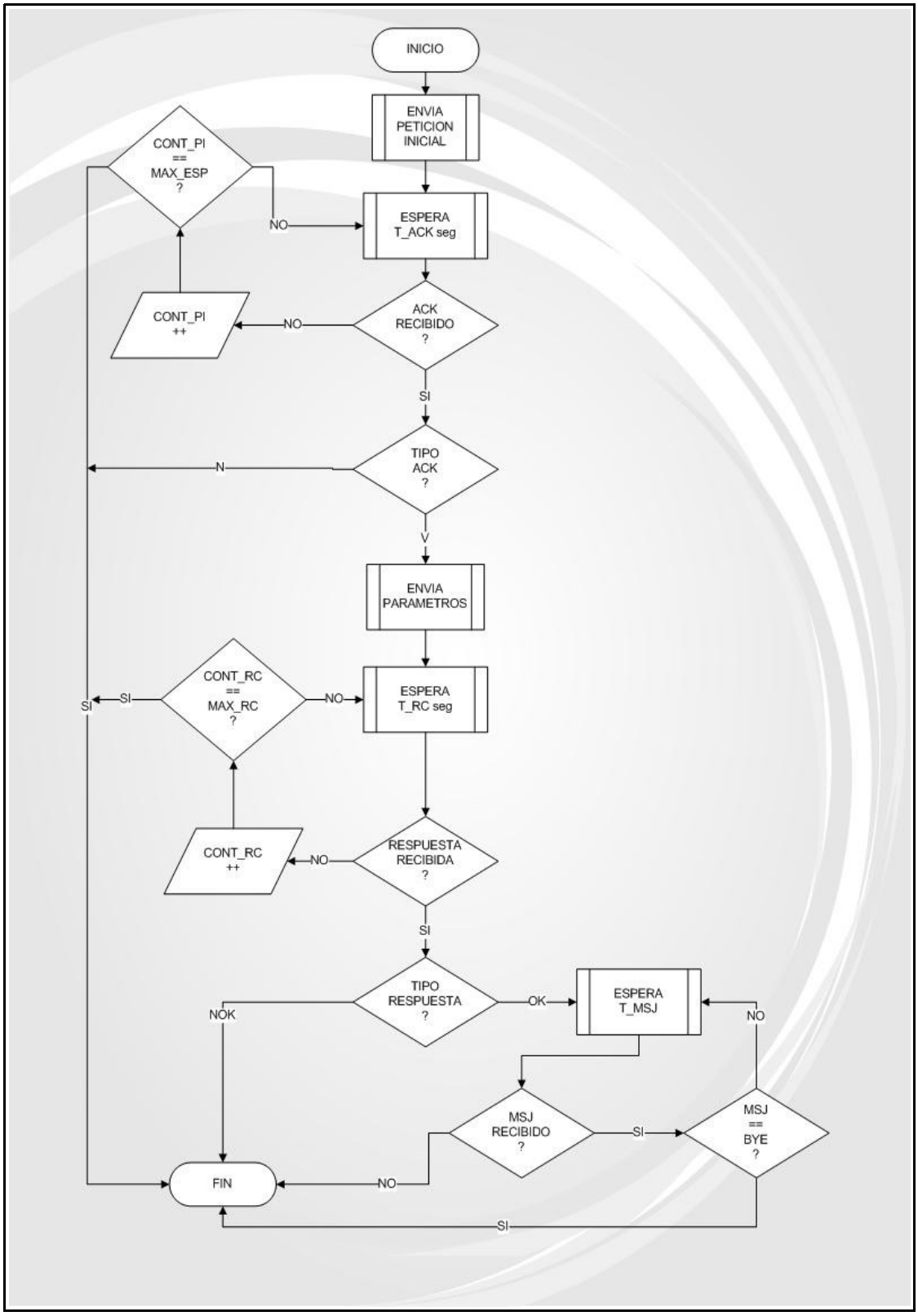

Ilustración 11: Diagrama de flujo de la aplicación CLIENTE.

3.3. DISEÑO DE LA APLICACIÓN SERVIDOR.

3.3.1. Diagrama de flujo de la aplicación SERVIDOR.

- T_RC: tiempo en segundos que permanece a la espera de los parámetros para el cálculo de la recomendación.
- MSJ_PI: mensaje de petición inicial para el cálculo de una recomendación.
- CONT_NO_RC: contador del número de veces que se esperan los parámetros para el cálculo de la recomendación y no se reciben.
- CONT_PR: contador del número de veces que se intenta hacer el cálculo de la recomendación y no se consigue.
- MAX_NO_RC: cota máxima para CONT_NO_RC.
- MAX_PR: cota máxima para CONT_PR.
- ACK_N: asentimiento negativo. No acepta la petición inicial porque el estado en el que se encuentra el servidor de recomendaciones no lo permite.
- ACK_V: asentimiento positivo. Acepta la petición inicial porque el estado del servidor de recomendaciones lo permite.
- MSJ_RC_OK: mensaje que indica que los parámetros para el cálculo de la recomendación son correctos y por lo tanto se procesa la petición.
- MSJ_RC_NO_OK: mensaje que indica que los parámetros para el cálculo de la recomendación no son correctos y por lo tanto no se procesa la petición.
- ESPERA MSJ_PI: función que mantiene el servidor a la espera de una petición inicial para el cálculo de una recomendación.
- COMPRUEBA ESTADO: función que comprueba si el servidor de recomendaciones puede aceptar una nueva petición de recomendación.
- COMPRUEBA MSJ_RC: función que comprueba que los parámetros para el cálculo de una

recomendación son correctos o no.

- PROCESA MSJ_RC: función que realiza el cálculo de la recomendación. Si el procesado se realizó con éxito o no se lo indica al proceso cliente.
- ENVÍA RESULTADO: función que se encarga de enviar al cliente el/los mensaje/s que forman la recomendación final. Se enviarán una secuencia de mensajes y para indicar el final un mensaje BYE.

Este diagrama de flujo define la estructura de la aplicación servidor. Dicho servidor ha sido implementado en lenguaje C a partir de este diagrama de flujo. El código fuente del mismo se ha adjuntado en los anexos para su consulta. Dicho código fuente está comentado de forma exhaustiva y detallada para que sea fácil la comprensión del mismo y se sepa que es lo que realiza cada sentencia y función definida.

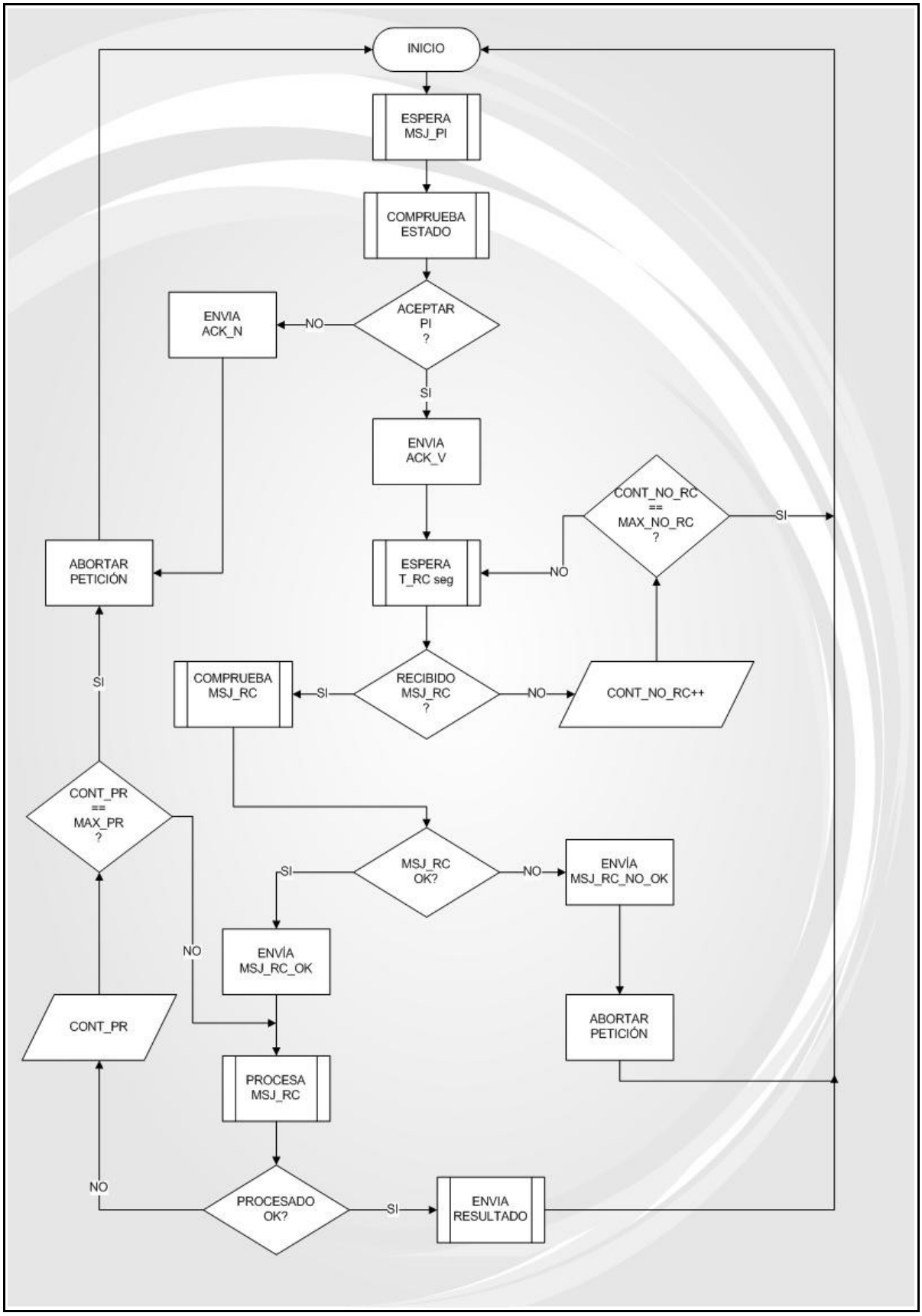

Ilustración 12: Diagrama de flujo de la aplicación SERVIDOR

3.4. INSTALACIÓN DEL ENTORNO DE DESARROLLO.

3.4.1. Instalación de Linux Mandriva.

Al introducir el CD/DVD, aparecerá la pantalla siguiente, en la que podemos seleccionar el idioma, y la configuración de la pantalla pulsando F2 y F3 respectivamente. Para acceder, se marca la opción Instalar Mandriva Linux 2008 Spring [17].

Ilustración 13: Instalación Linux Mandriva.

Una vez a cargado algunos de los dispositivos y la aplicación de instalación en la memoria (lo cual se lleva a cabo automáticamente), se entra en el entorno de instalación (el cual es aparte de muy bueno gráficamente, muy sencillo de usar, características que definen a Mandriva). La primera opción que aparece es la del idioma, dónde seleccionamos aquel que queremos. Una vez aceptado, aparecerá el Acuerdo de Licencia (GPL), no estaría mal leérselo, aunque sea por curiosidad, pero

para seguir con la instalación es necesario aceptarlo.

La siguiente tarea será elegir el idioma de teclado, que por defecto marca el que se ha elegido como idioma.

Ilustración 14: Instalación Linux Mandriva. Idioma

Una vez pasadas las primeras pantallas, llega la primera tarea delicada, no por su dificultad, sino por las opciones que tiene; se trata de particionar el disco duro. Si no se es un instalador experimentado o recomendado es que la partición del disco la efectúe el instalador de Mandriva el cual lo hace a la perfección creando las necesarias: 1 swap, 1 para el directorio raíz (/) y 1 para el directorio home (/home) que es donde se guardará la configuración de los usuarios. Es muy importante que si se crea a mano, se separe el /home, pues si por algún motivo es necesario

reinstalar Mandriva, se conservará toda la configuración del escritorio y los programas guardados. Otra ventaja es a la hora de instalar una nueva versión, pues así solo no se toca la partición dónde está el directorio /home y ya está instalado el sistema.

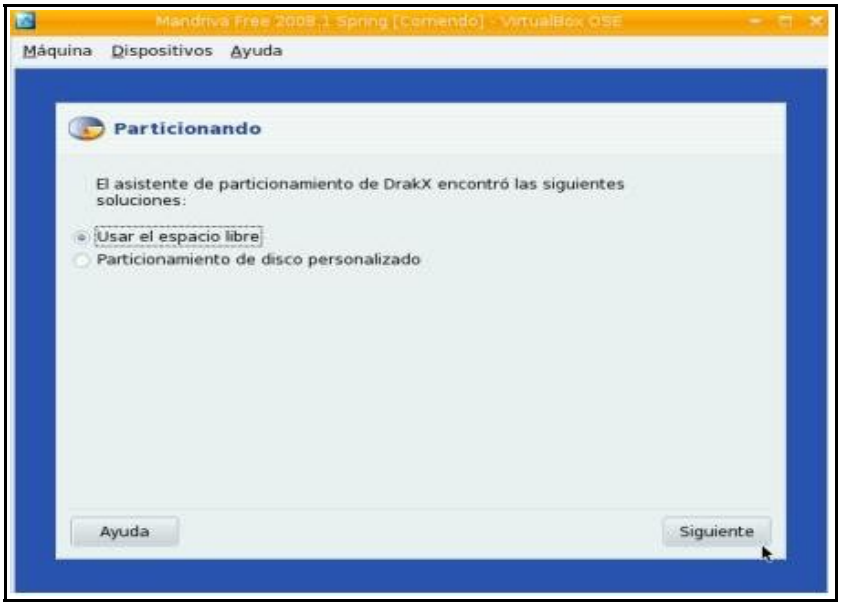

Ilustración 15: Instalación Linux Mandriva. Particionado

Una vez creadas y formateadas las particiones, se pregunta dónde se desea instalar los paquetes, en este caso y obviando que se dispone de conexión a Internet, lo hará desde el CD/DVD, por lo que se responde Ninguno cuando pregunte por soportes suplementarios a configurar. Los del CD/DVD los toma automáticamente.

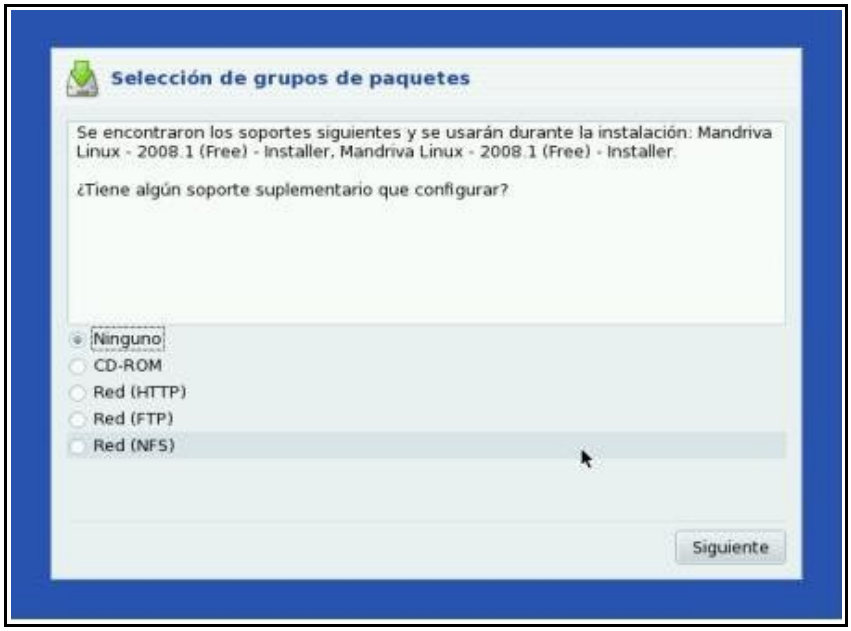

Ilustración 16: Instalación Linux Mandriva. Fuente de Instalación.

Aquí llega la segunda tarea delicada: la selección de paquetes (o programas) que se quieren instalar. Para tener el entorno KDE personalizado por Mandriva, se pulsa Instalación personalizada, de otra forma instala los entornos por defecto KDE o Gnome, según el caso. Y ahora después se mostrarán las dos variantes existentes para instalar los paquetes.

Ilustración 17: Instalación Linux Mandriva. Selección de paquetes.

Variante A: Se puede optar por hacerlo de forma sencilla, pulsando los tipos de programas que se desean instalar y apretando siguiente. Para esta opción hay que dejar sin marcar la casilla de "Selección de paquetes individuales". Una vez pulsamos siguiente, el instalador inicia la ardua tarea de instalación, que solo llevará unos 10 minutos escasos.

Ilustración 18: Instalación Linux Mandriva. Selección de paquetes A

Variante B: Esta variante es para usuarios más experimentados y que tienen la necesidad de instalar algunos programas de forma especial, como puedan ser MySQL, Apache, OpenSSH, etc. Para este caso sí se marca la opción "Selección de paquetes individuales". Una vez se pulsa siguiente, aparecerá esta pantalla para seleccionar los paquetes uno por uno.

Ilustración 19: Instalación Linux Mandriva. Administrar Software

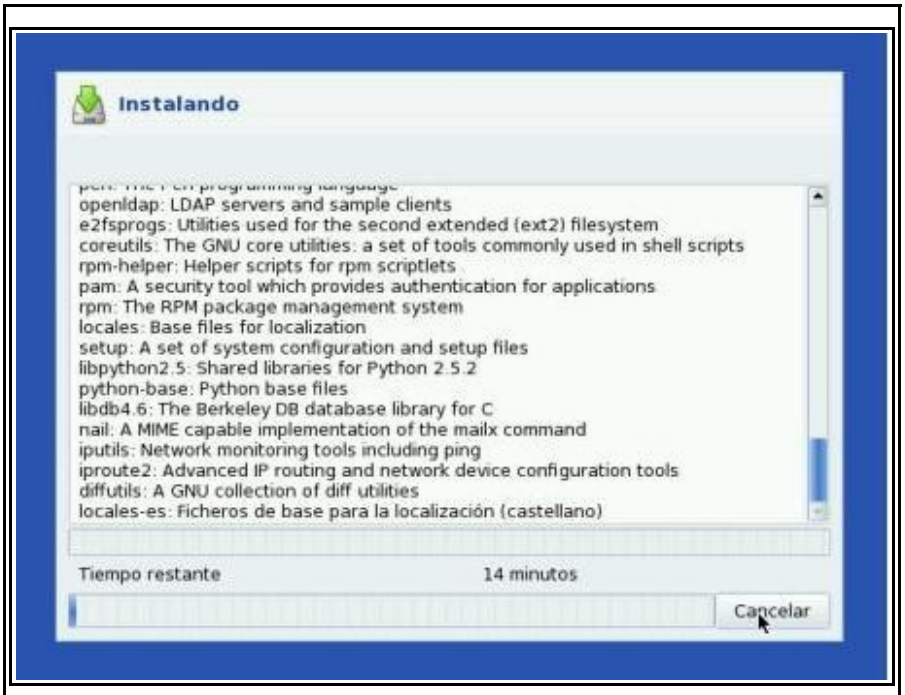

Ilustración 20: Instalación Linux Mandriva. Administrar Software

Una vez pasados los 10 minutos, se puede volver a mirar la pantalla, donde se realizarán unas instalaciones de paquetes, unos 10 aproximadamente, y se configurará automáticamente el gestor de arranque (por defecto GRUB en modo gráfico). Acabado de configurar el cargador, que pueden ser unos 2 o 3 minutos, se continua configurando la instalación. Ahora toca los usuarios y la contraseña de root. Es muy importante que no olvidar ni revelar la contraseña de root, pues es la de superusuario, aquel que puede configurar las opciones. Por lo demás, sólo se tiene que configurar un usuario por defecto con su nombre y apellidos, nombre del ordenador en la red, y contraseñas.

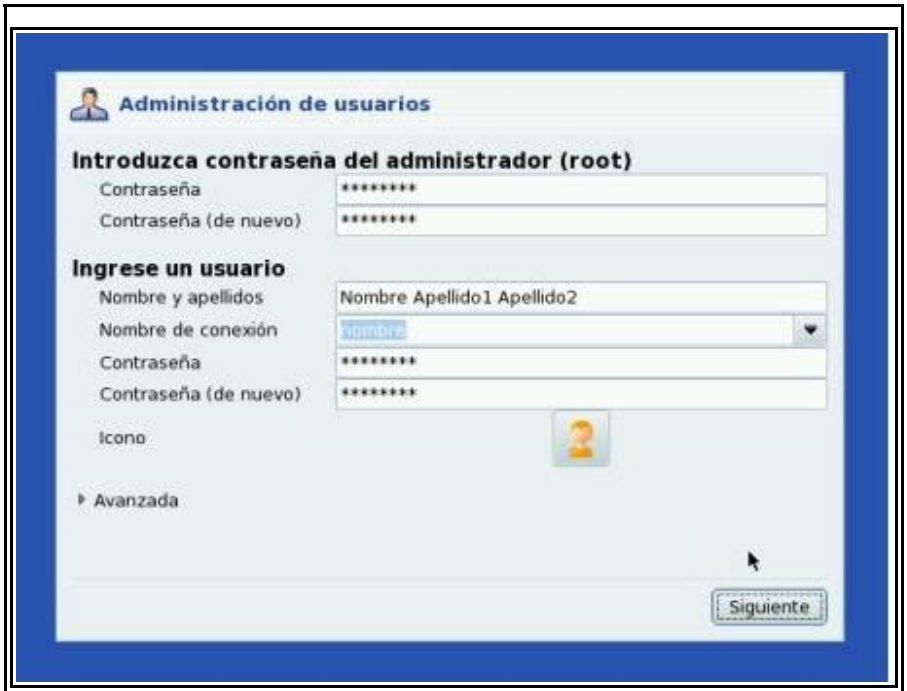

Ilustración 21: Instalación Linux Mandriva. Creación de usuarios

Una vez se acepta la creación de usuarios, es momento de seleccionar la configuración del monitor, aunque la que se pone por defecto debería estar bien.

Ilustración 22: Instalación Linux Mandriva. Configurar hardware

A continuación aparece un resumen de la configuración hasta ahora. Se debe observar que esta todo, o casi todo bien configurado, porque habrá una cosa importante que no: la conexión a Internet o a la red. Se procede a configurarla buscando la opción Redes e Internet y pulsando en Configurar.

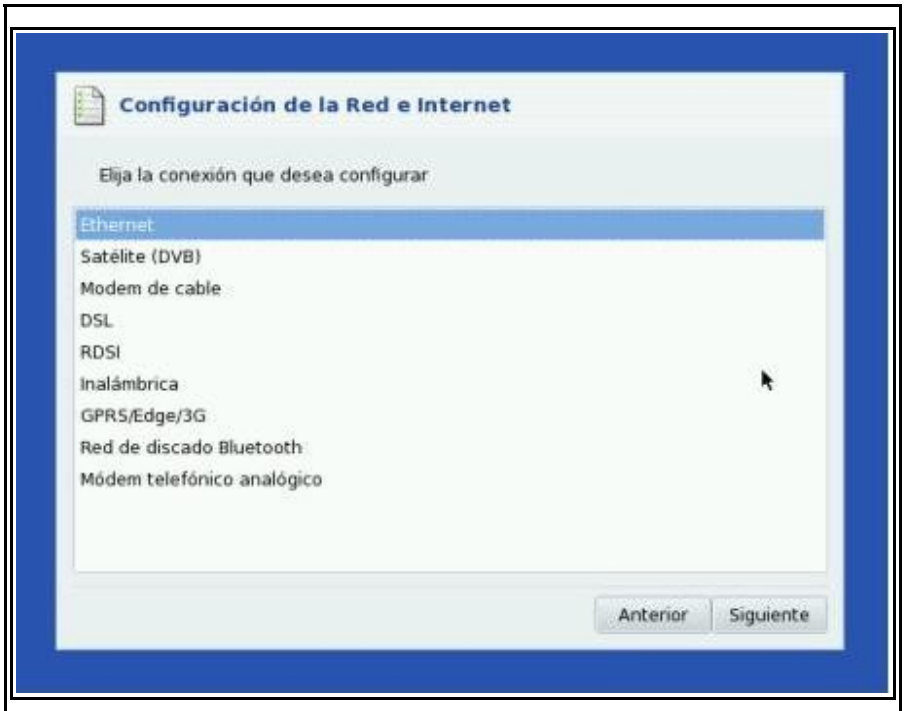

Ilustración 23: Instalación Linux Mandriva. Configurar red

Si la conexión es por cable (par trenzado), se escoge Ethernet, para otro caso, se elige:

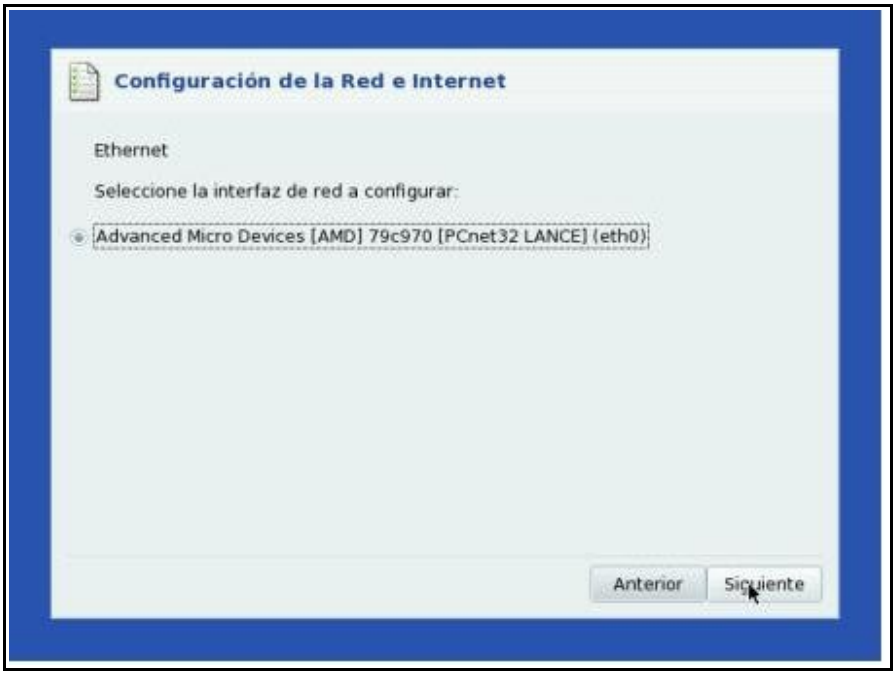

Ilustración 24: Instalación Linux Mandriva. Configurar red

Se selecciona el dispositivo a configurar (lo reconocerá automáticamente)

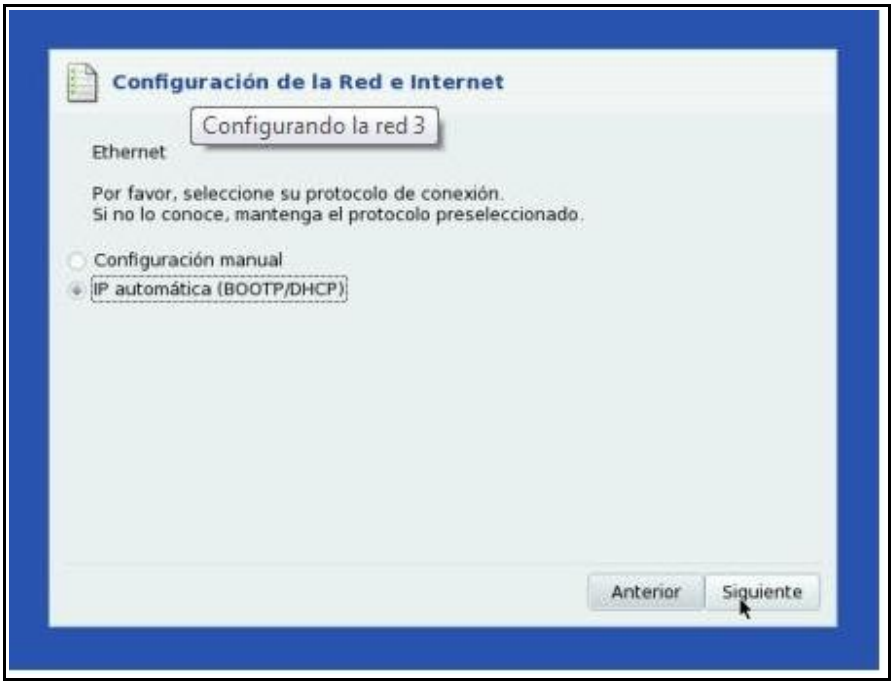

Ilustración 25: Instalación Linux Mandriva. Configurar red

Se elige que asigne una IP manual, esto es lo mejor cuando la conexión a Internet es a través de un router. Él asignará la IP

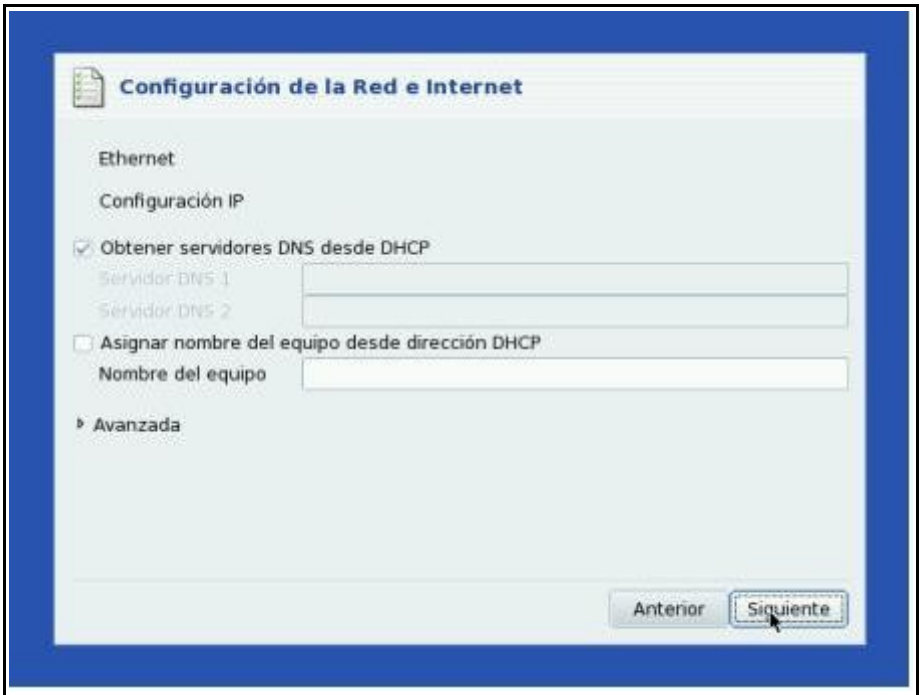

Ilustración 26: Instalación Linux Mandriva. Configurar red

Al igual que antes, se elige que el router se encargue de los servidores DNS.

Ilustración 27: Instalación Linux Mandriva. Configurar red

Se marcan las dos opciones de gestión de la interfaz de red

Ilustración 28: Instalación Linux Mandriva. Configurar red

Estando en el resumen, se pulsa Siguiente y el instalador pregunta si se desea instalar las actualizaciones en caso de haberlas. Se marca que No y por último se reinicia.. Una vez se arrancar el ordenador, aparecerá el menú de arranque, se pulso la opción por defecto, y se espera unos 45 segundos hasta la pantalla de autenticación. Una vez en esa pantalla se indica el usuario y la

Ilustración 29: Instalación Linux Mandriva. Login

contraseña.

Recordar que hay que volver a poner en primera opción el arranque desde el Disco Duro en la bios y sacar el CD/DVD.

Ilustración 30: Instalación Linux Mandriva. Escritorio

3.4.1.1. Configuración de los repositorios

Lo primero que hay que hacer una vez instalado Mandriva, es configurar los repositorios, tanto los des software libre, como los de software privativo, pues tienen algunas aplicaciones bastante interesantes. Los repositorios son almacenes de paquetes (programas, librerías, etc.). Hay tanto repositorios oficiales (con todos los paquetes de la distribución, así como sus actualizaciones de seguridad y bugs), como repositorios con software que Mandriva no puede incluir en los oficiales (el llamado PLF, que contiene codecs multimedia propietarios, programas libres pero con restricciones legales, etc). Para entrar en el Centro de Control de Mandriva, se puede acceder desde Menú –> Herramientas –> Herramientas del sistema–> Configurar su computadora, o desde el

tercer icono empezando por la izquierda al lado del botón Menú. Nos aparecerá una pantalla donde se pide la contraseña de root.

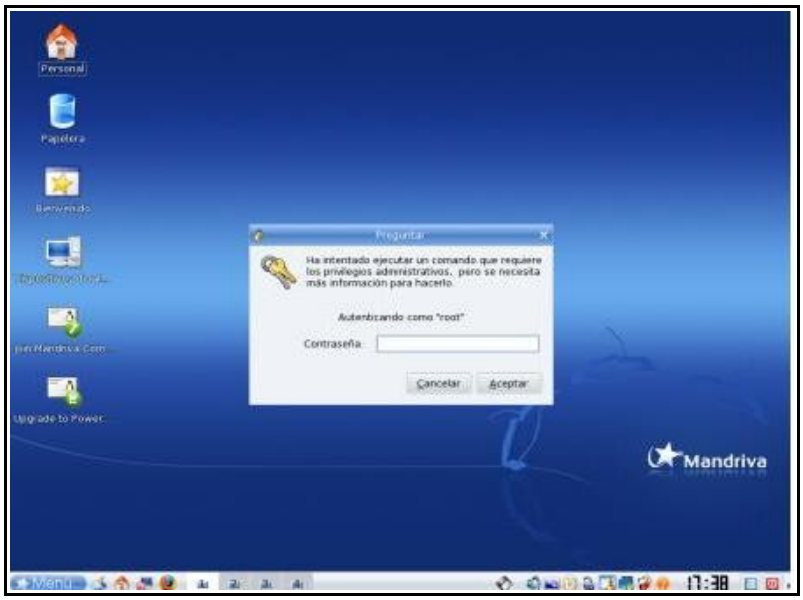

Ilustración 31: Instalación Linux Mandriva. Configurar repositorios

Una vez dentro, aparecemos en la pestaña de Administración de software. Aquí se tiene que pulsar la opción Configurar los soportes de paquetes.

Ilustración 32: Instalación Linux Mandriva. Configurar repositorios

Se seleccionan los repositorios de los CD o DVD.

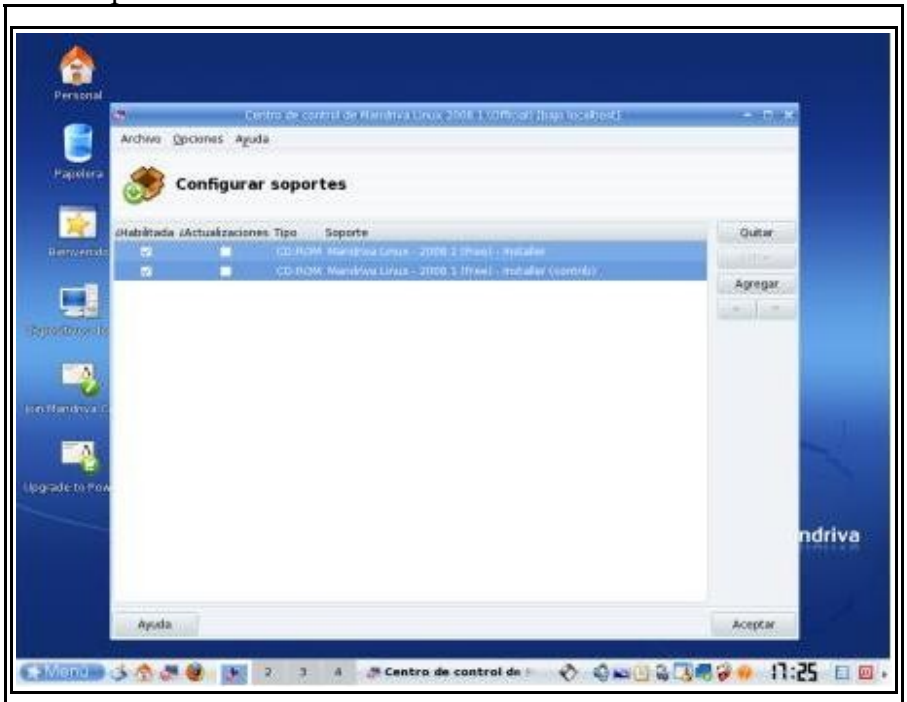

Ilustración 33: Instalación Linux Mandriva. Configurar repositorios

Una vez eliminados estos repositorios, se pulsa en agregar y aparecerá la pregunta de si se desea configurar ahora los repositorios, y se pulsa que Sí, lo que permite acceder a la siguiente pantalla, donde se selecciona el Conjunto completo de fuentes y de dónde se quiere hacer las descargas.

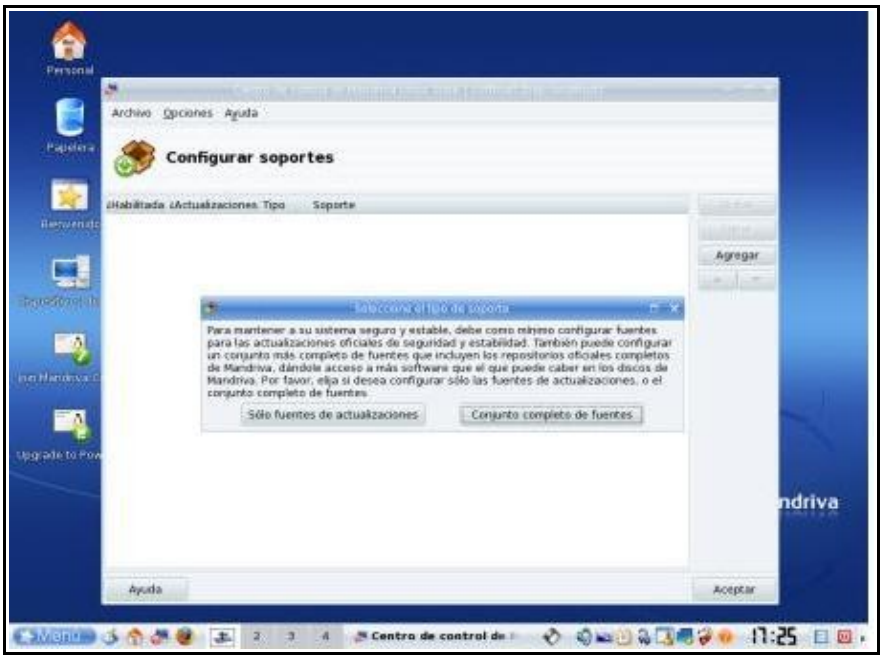

Ilustración 34: Instalación Linux Mandriva. Configurar repositorios

Ilustración 35: Instalación Linux Mandriva. Configurar repositorios

Se acepta y ya se tienen los repositorios instalados. Ahora es posible instalar todo el software que se necesite desde el Centro de Control de Mandriva, en la misma pestaña que se ha configurado los repositorios, pero en la opción Instalar y quitar software. La pantalla para instalar tiene la siguiente forma, y sólo es necesario buscar el programa deseado y pulsar Aplicar. Del resto se ocupa

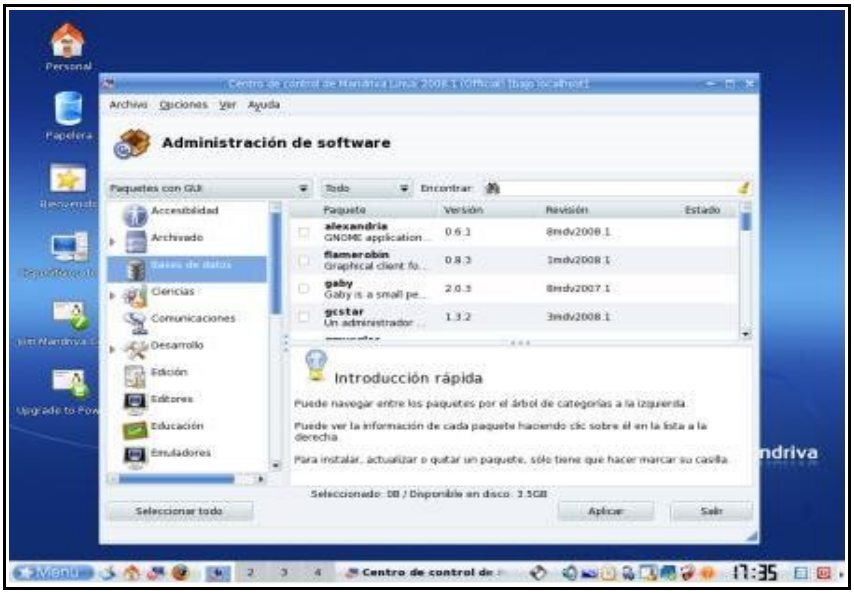

Ilustración 36: Instalación Linux Mandriva. Configurar repositorios

Mandriva, así de fácil.

3.4.2. Instalación de XAMPP.

Los pasos a seguir para instalar XAMPP [10] en nuestro sistema Linux son muy simples:

- Descargar el paquete desde la web oficial: http://www.apachefriends.org/en/xampplinux.html.
- Abrir una consola de Linux y registrarse como root con el comando su.
- Extraer los archivos en el directorio /opt: tar xvfz xampp-linux-1.6.8a.tar.gz -C /opt . A partir de este momento el paquete está instalado en el directorio /opt/lampp.
- Para iniciar XAMPP ejecutamos el siguiente comando:

  /opt/lampp/lampp start
- Y seguidamente aparecerá en la consola lo siguiente:
  - Starting XAMPP 1.6.8a...
  - LAMPP: Starting Apache...
  - LAMPP: Starting MySQL...
  - LAMPP started.A continuación se comprueba que funciona la interfaz web de XAMPP.
  - Para ello se abre el navegador y se teclea: http://localhost

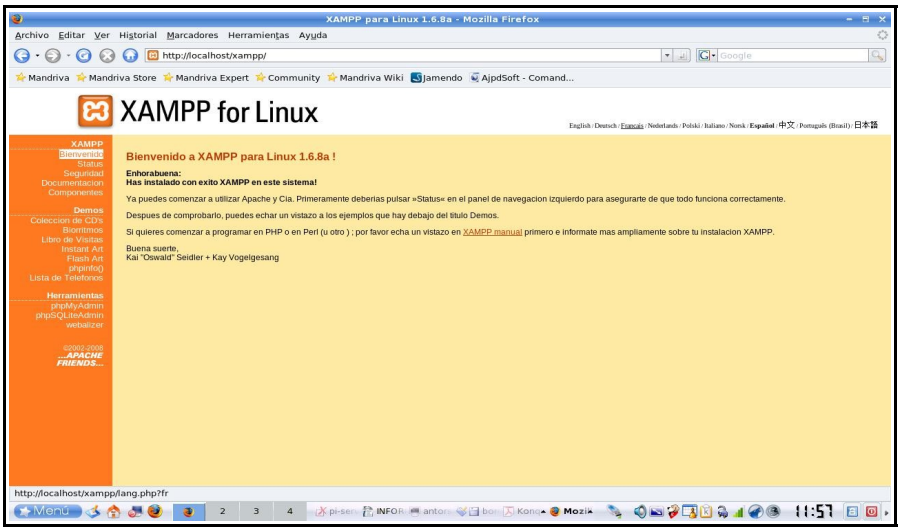

Ilustración 37: Interfaz web de XAMPP

Tal como ya se ha dicho este entorno presenta una serie de vulnerabilidades que a continuación listamos:

- El administrador root de MySQL no tiene password.
- El demonio MySQL es accesible desde la red.
- ProFTPD usa el password "lampp" para el usuario "nobody".
- PhpMyAdmin es accesible desde la red.
- Los ejemplos son accesibles desde la red.
- MySQL y Apache se ejecutan bajo el mismo usuario "nobody".
- Para resolver estas vulnerabilidades se puede ejecutar el siguiente comando: /opt/lampp/lampp security
- Es posible ver una lista completa de los comandos disponibles ejecutando: /opt/lampp/lampp help
- Si se quiere desinstalar XAMPP ejecutar:

  rm -rf /opt/lampp

3.4.3. Instalación de gSOAP.

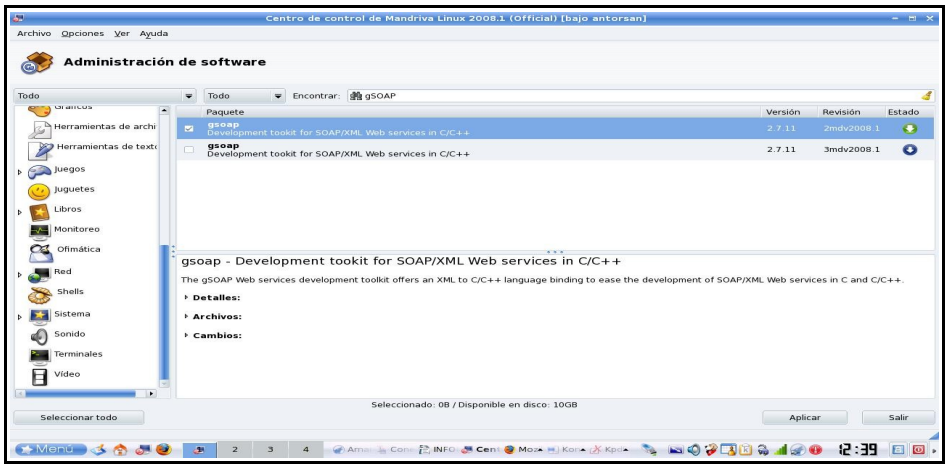

Ilustración 38: Instalación de gSOAP

La instalación en Linux Mandriva es muy sencilla, basta con ir al gestor de instalación y

desinstalación de software. Una vez en allí buscar gSOAP y pulsar el botón de instalar.

Una vez instalado ya es posible utilizar todas las funcionalidades de gSOAP. Entre ellas la que a nosotros nos interesa es la de generar código C a partir de un descriptor del servicio WDSL. El siguiente paso es abrir un terminal de consola para generar el código C que necesitamos. Para ello, se ejecuta los siguiente [4, 21]:

$ wsdl2h -c -o fichero.h fichero.wsdl

Donde la opción -c es para generar código C puro, -o para indicar el nombre del fichero de cabecera que se generará a partir del fichero descriptor del servicio, fichero.wsdl.

$ wsdl2h -c -o envios_minerva.h http://193.147.165.85:9002/minerva/envios.asmx?WSDL

A partir del fichero de cabecera generado, envios_minerva.h, el compilador gSOAP creará los "STUBS" para desarrollar las aplicaciones cliente o servidor. Para ello ejecutar:

$ soapcpp2 envios_minerva.h

De esta forma obtenemos los ficheros fuente para implementar el Servicio Web completo, tanto el lado cliente como el lado servidor. Si se quiere solo el código C necesario para el lado cliente:

$ soapcpp2 -C envios_minerva.h

Los ficheros generados se muestran más abajo en la ilustración

Lo ficheros que utilizaremos para compilar nuestra interfaz SOAP serán:

- soapH.h: necesario obligatoriamente para usar los "STUBS" generados.
- soapStub.h: definiciones de estructuras y declaración de las funciones propias del lado cliente y el servidor.
- EnviosSoap.nsmap: necesario para definir los espacios de nombres.
- soapC.c: código fuente que contiene las funciones necesarias para el mapeo entre C y XML.
- soapClient.c: código fuente que contiene las funciones específicas del lado cliente. Entre

ellas la que se emplea para enviar los SMS-MT a la plataforma Minerva -RedBox: SOAP_FMAC5 int SOAP_FMAC6 soap_call___ns1__enviosms(struct soap *soap, const char *soap_endpoint, const char *soap_action, struct _ns1__enviosms *ns1__enviosms, struct _ns1__enviosmsResponse *ns1__enviosmsResponse).

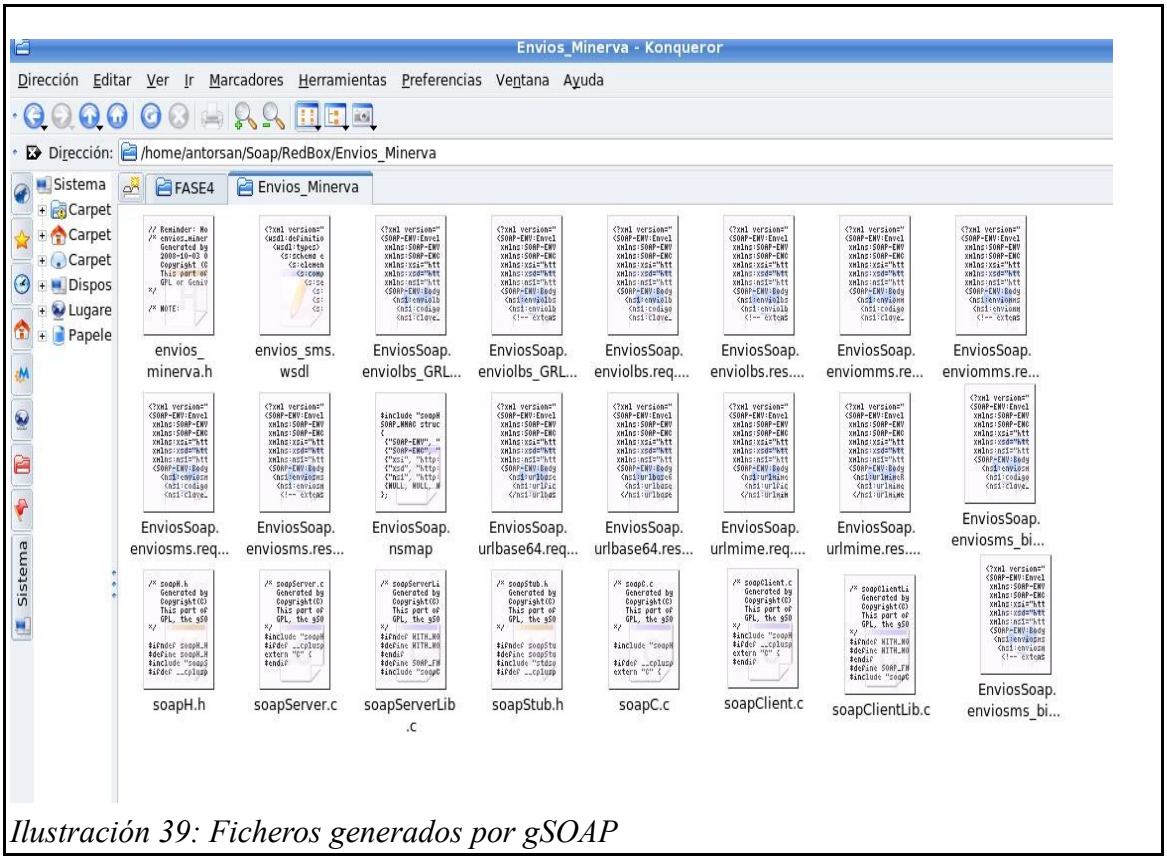

Ilustración 39: Ficheros generados por gSOAP

3.5. IMPLEMENTACIÓN DE LA APLICACIÓN CLIENTE.

Una vez definidas las características de la aplicación cliente, lo único que queda es implementar dicha aplicación. Para ello se emplea el lenguaje de programación C, que proporciona todas las funcionalidades necesarias y una gran flexibilidad. De hecho, es posible asegurar que con C se puede hacer todo, en lo que se refiere al ámbito de programar aplicaciones. La aplicación cliente será un script CGI que comenzará a ejecutarse cuando se lleve a cabo una petición al servidor donde se aloje. Por un lado, dichas peticiones como ya se ha explicado podrán provenir de otra página web, en concreto un formulario codificado en código HTML. O bien, tendrán su origen en un

dispositivo móvil que enviará un SMS a través de la red GSM o UMTS. En ambos casos, la aplicación cliente recibirá una serie de parámetros a través del método POST. Estos parámetros serán analizados y la aplicación sabrá entonces si se trata de una petición vía WEB o vía SMS.

Llegados a este punto se procede a explicar el funcionamiento de las dos interfaces que interaccionan con la aplicación cliente, en los apartados siguientes

- La interfaz WEB que está basada en un formulario en HTML que realiza peticiones HTTP al servidor que a su vez invoca al script CGI.
- La interfaz SMS que está basada en un SERVICIO WEB que se implementa mediante la plataforma Minerva Red-Box.

3.5.1. Implementación de la interfaz web.

La interfaz Web es la que permite a la aplicación CLIENTE atender peticiones de un cliente a través de Internet. Para ello la aplicación cliente se implementa en forma de script CGI y queda instalada en un directorio del servidor web dedicado para tal fin (en Apache es **cgi-bin**). De forma simplificada el proceso es el siguiente:

Se dispone de un servidor web, el cual ofrece un formulario web para realizar peticiones de recomendación, dicho formulario se encuentra en el directorio de documentos públicos de nuestro servidor (en el caso de Apache es **htdocs**). El usuario de Internet realiza una petición http para que el servidor le muestre en el navegador el formulario de peticiones web, rellena el formulario y al pulsar el botón de enviar los datos son capturados por la aplicación CLIENTE. Tras capturar los datos, la aplicación CLIENTE establece el dialogo con la aplicación SERVIDOR que en función de los parámetros que reciba, ofrece una determinada respuesta a la aplicación CLIENTE, que nuevamente envía los resultados al usuario final mediante una página web. Todo esto se muestra en la siguiente ilustración.

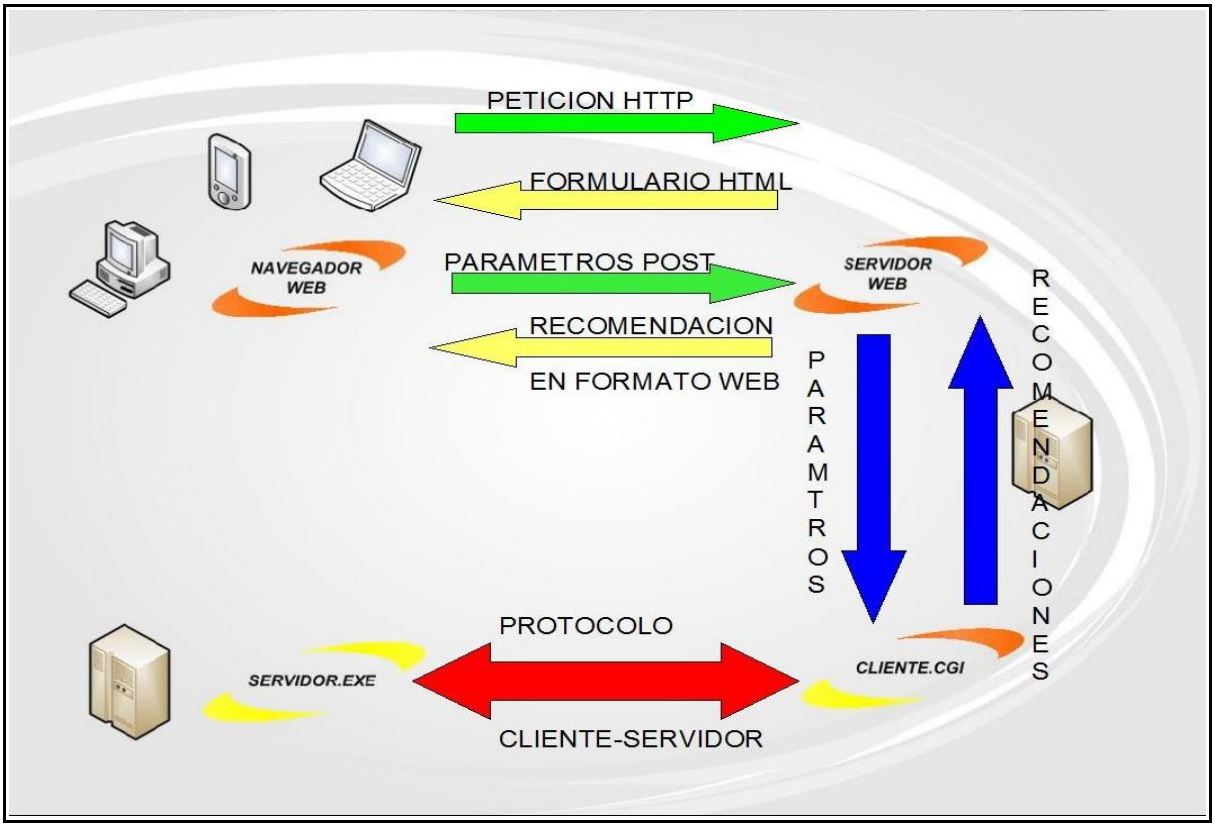

Ilustración 40: Servicio de recomendación a través de Internet

3.5.1.1. El formulario.

Los formularios son posiblemente la herramienta más utilizada en Internet para obtener datos e información acerca de la gente que navega nuestro sitio. La idea de los formularios es recolectar información online en la interacción con el usuario y luego ejecutar una determinada acción con la misma, que podría ser por ejemplo, quizás el caso más utilizado, un formulario de venta que el usuario completa y luego es enviado vía email al vendedor en forma encriptada. En nuestro caso, la información recolectada por el formulario se le pasará a la aplicación CLIENTE.

En el código HTML [7, 8, 9] sólo se emplean cinco etiquetas para definir todos los elementos interactivos que un formulario puede presentar: **<FORM>**, **<INPUT>**, **<SELECT>**, **<OPTION>** y **<TEXTAREA>**. Estas etiquetas, junto a un número de modificadores o atributos, son suficientes

para indicarle al navegador cliente cómo debe recolectar la información, cómo debe manejarla una vez recolectada y cómo debe interactuar con el servidor.

3.5.1.1.1. Etiqueta FORM.

Todo formulario debe estar encerrado entre el par de etiquetas **<FORM>** y **</FORM>**, y debe ser ubicado en el cuerpo de cualquier documento HTML, es decir, entre el par de etiquetas **<BODY>** y **</BODY>**. Esta etiqueta **<FORM>** presenta tres atributos posibles:

- **ACTION** el valor de este parámetro es la URL del programa o guión en el Servidor Web utilizado para procesar la información recolectada.
- **METHOD** puede asumir el valor **GET** o el valor **POST**, y definen la manera en la cual los datos son transferidos al servidor.
- **ENCTYPE** este atributo está reservado para que la información viaje en forma encriptada a través de Internet.

Los dos primeros atributos de la tabla son de uso obligatorio para cualquier formulario que generemos, ya que establecen dónde enviar la información y cómo enviarla. El atributo o parámetro **ENCTYPE** es optativo y no es realmente importante.

3.5.1.1.2. Etiqueta INPUT.

La etiqueta **<INPUT>** es la segunda etiqueta más importante de los formularios. Se la puede definir como una etiqueta multifunción, ya que con la misma podemos crear "push buttons", "radio buttons", "check boxes", y simples recuadros para ingresar texto. Posee ocho posibles parámetros: **ALIGN**, **CHECKED**, **MAXLENGTH**, **NAME**, **SIZE**, **SRC**, **TYPE**, y **VALUE**. Pero normalmente solo se utilizan a lo sumo solo cinco. Los atributos cruciales para toda etiqueta **<INPUT>** son **NAME**, que asocia un nombre con cada variable ingresada; y **TYPE**, que puede asumir los valores **TEXT**, **PASSWORD**, **CHECKBOX**, **RADIO**, **SUBMIT**, **RESET**, **IMAGE**, and **HIDDEN** (pueden ser necesarios para algunos scripts, lo que hacen es comunicar cierta

información al servidor acerca de cómo manipular los datos manteniéndose ocultos a la vista de los usuarios); de acuerdo al tipo de elemento que querríamos representar.

3.5.1.1.3. Etiqueta SELECT.

Este par de etiquetas define una lista de elementos de los cuales el usuario debe seleccionar uno o varios, de acuerdo a los atributos que especifiquemos.

3.5.1.1.4. Etiqueta OPTION.

Con esta etiqueta definimos cada elemento de las listas designadas con el par **<SELECT>** y **</SELECT>**.

3.5.1.1.5. Etiqueta TEXTAREA.

Este par de etiquetas nos permiten definir un área de dimensiones arbitrarias que funciona como una suerte de editor, donde el usuario puede ingresar texto.

3.5.1.1.6. El atributo ACTION.

Como dijimos anteriormente todo formulario debe comenzar con la etiqueta **<FORM>** y finalizar con **</FORM>**. El parámetro **ACTION** define qué es lo que debemos hacer con la información obtenida. La mayoría de los formularios en Internet recolectan información del usuario y la envían por correo electrónico hacia algún destino. Para realizar esta tarea existen dos formas de hacerlo, una es: **<FORM ACTION="mailto:quantum@mundo21.com" METHOD=POST>/p>**; y la otra es hacer uso de algún script o guión **CGI (Comon Gateway Interface)** que procesa los datos y los reenvía hacia donde le indicamos.

Un **script CGI** es un pequeño programa escrito en general en lenguaje PERL o C (muy populares en ambientes Unix) alojado en los servidores web que facilitan el intercambio y la transferencia de información entre el servidor y el cliente.

¿Cuál es la diferencia entre ambos métodos? El primer caso es el más sencillo y no utiliza ningún script, pero a su vez es menos eficaz, ya que en la mayoría de los casos no funcionará. Mientras que el segundo, si bien un poco más complicado, asegura casi un 100% de efectividad y una gran variedad de opciones para hacerlo, pues existen miles de scripts para diferentes tipos de formularios. Si quisiéramos utilizar un script **CGI**, es necesario averiguar si el servidor web que aloja nuestro sitio posee un directorio con scripts cgi, comúnmente conocidos como **/cgi-bin** (pues aloja binarios). Además, los campos a utilizar deben ser aceptados por el script, es decir el script está diseñado para manejar cierto número y tipo de variables, es por eso que se debe configurar el formulario acorde al script o guión seleccionado.

3.5.1.2. Construcción de los bloques elementales de un formulario.

3.5.1.2.1. Casillas de texto.

<FORM>
<INPUT TYPE="text" NAME="nombre" VALUE="Antonio">
</FORM>

Es posible indicar el tamaño máximo de la casilla con **SIZE**. El valor predeterminado para el tamaño de un **INPUT** es **SIZE**=20.

<FORM>
<INPUT TYPE="text" NAME="nombre" VALUE="Antonio" SIZE=10>
</FORM>SIZE=10>

También se puede indicar la cantidad de caracteres a ingresar por el usuario con **MAXLENGTH**. Para los casos en los que se quiere introducir un password se empleará: **TYPE=PASSWORD.**

<FORM>
<INPUT TYPE="password" NAME="clave">

</FORM>

3.5.1.2.2. Radio Buttons.

<FORM>
<INPUT TYPE="radio" NAME="preferencia" VALUE="1">PRECIO
*
<INPUT TYPE="radio" NAME="preferencia" VALUE="2">TIEMPO*
*
<INPUT TYPE="radio" NAME="preferencia" VALUE="3">PRECIO Y TIEMPO*
</FORM>

El **NAME** es el mismo para los tres Radio Buttons, pues son tres opciones para el mismo **campo NAME**="preferencia". En este caso el **VALUE** ya está definido, solo se debe esperar la selección del usuario, por ejemplo si marcase la primera se procesaría la siguiente información: preferencia=2. Con el parámetro **CHECKED** se indica al navegador cual de todas las opciones debe aparecer marcada por defecto.

3.5.1.2.3. Check Boxes.

A diferencia de los Radio Buttons, los Check Boxes mantienen el mismo valor para el parámetro **VALUE**, y cambian el valor de **NAME**. Por ejemplo:

<FORM>
<INPUT TYPE="checkbox" NAME="tren" VALUE="si">Tren
*
<INPUT TYPE="checkbox" NAME="avión" VALUE="si">Avión*
*
<INPUT TYPE="checkbox" NAME="barco" VALUE="si"> Barco*
*
<INPUT TYPE="checkbox" NAME="bus" VALUE="si"> Bus*
</FORM>

Al igual que con los Radio Buttons es posible hacer uso del parámetro **CHECKED** para marcar alguna por defecto.

3.5.1.2.4. Listas Desplegables.

<FORM>
<SELECT NAME="ciudades">
<OPTION VALUE="0">Ciudad 1

```
<OPTION VALUE="1">Ciudad 2
<OPTION VALUE="2">Ciudad 3
<OPTION VALUE="3">Ciudad 4
<OPTION VALUE="4">Ciudad 5
</SELECT>
</FORM>
```

La opción por defecto de una lista de este tipo es la primera **<OPTION>** declarada. Si se quiere marcar otra opción por defecto lo usar el parámetro **SELECTED**.

3.5.1.2.5. Listas Desplegadas.

```
<FORM>
<SELECT NAME="equipos" SIZE=5>
<OPTION VALUE="0">Ciudad 1
<OPTION VALUE="1">Ciudad 2
<OPTION VALUE="2">Ciudad 3
<OPTION VALUE="3">Ciudad 4
<OPTION VALUE="4">Ciudad 5
</SELECT>
</FORM>
```

El valor del parámetro **SIZE** indica exactamente cuántos elementos desplegar simultáneamente. Si el valor del **SIZE** es menor al número total de elementos de la lista, automáticamente aparecerá un barra de desplazamiento sobre el lado derecho de la misma. Al igual que las lista desplegables se puede indicar un elemento seleccionado por defecto mediante el parámetro **SELECTED**.

3.5.1.2.6. Área de Texto.

Para generar un área de texto donde el usuario pueda escribir libremente un mensaje, un comentario o sugerencia, se debe utilizar la etiqueta **<TEXTAREA>** de la siguiente manera:

```
<FORM>
<TEXTAREA NAME="SUGERENCIAS">
</TEXTAREA>
</FORM>
```

Controlamos el tamaño del área de texto con los parámetros **ROWS** y **COLS**, cuyos valores representan el número de filas y columnas respectivamente que ocupará en pantalla dicha área Para entenderlo mejor, **ROWS** sería la altura y **COLS** la anchura.

```
<FORM>
<TEXTAREA NAME="SUGERENCIAS" ROWS=6 COLS=50>
</TEXTAREA>
</FORM>
```

3.5.1.2.7. Botones Submit y Reset.

```
<FORM>
<INPUT TYPE="submit" VALUE="Enviar Datos">
<INPUT TYPE="reset" VALUE="Borrar Datos">
</FORM>
```

Para ver el código fuente del formulario empleado, consultar los anexos. Además en la ilustración 36 que aparece en la siguiente página se muestra como el navegador muestra dicho formulario.

3.5.1.3. Programación del CGI.

El CGI [5, 6]se ejecuta en el servidor, por lo tanto, se puede programar en cualquier lenguaje que permita crear ejecutables para el sistema operativo del servidor. Dicho lenguaje de programación debe cumplir una serie de requisitos:

- Leer datos de la entrada estándar.
- Acceder a las variables de entorno.
- Escribir en la salida estándar.
- El lenguaje puede ser compilado o interpretado.

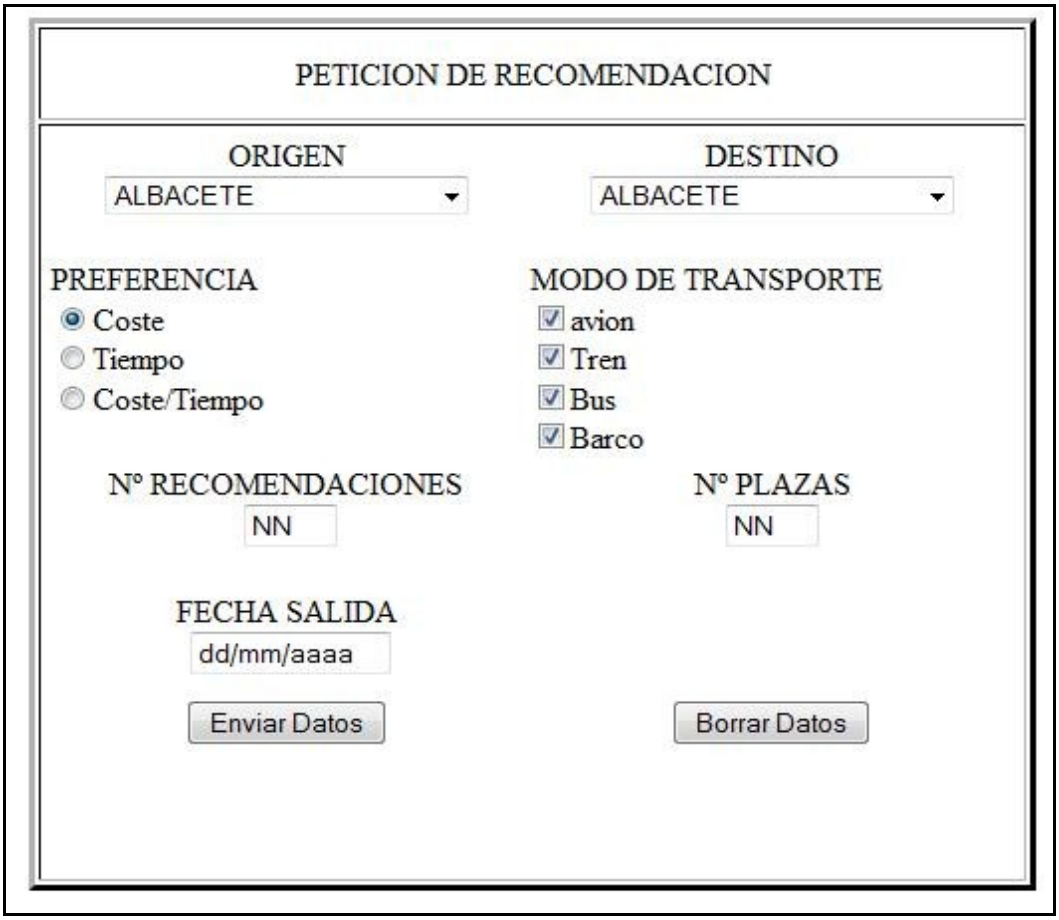

Ilustración 41: Formulario de petición web

3.5.1.3.1. Intercambio de información entre el Servidor y un CGI.

Un programa CGI recibe la información desde un servidor web de cuatro formas diferentes:

- A través de la línea de comandos.
- A través de la URL (QUERY_STRING).
- A través de la entrada estándar (stdin).
- A través de información de ruta (PATH_INFO).

El programa **CGI** tiene que saber como va a recibir la información, ya que en cada caso tiene

que actuar de forma diferente. Los métodos más populares son a través de la URL (método **GET**) y a través de la entrada estándar (método **POST**).

3.5.1.3.2. Tratamiento de Formularios.

Los métodos que permiten el tratamiento de formularios son **GET** y **POST**. Pero para poder procesar los formularios correctamente es necesario saber el formato con el que el navegador web codifica automáticamente los datos introducidos en el formulario y enviados al servidor web. Los datos introducidos en un formulario se envían al programa **CGI** como una cadena de caracteres con pares ordenados en la forma nombre=valor, separados por un signo ***&***. Los espacios serán pasados como un signo más (+) y los caracteres especiales (incluyendo el espacio) en la forma ***%valor*** hexadecimal.

Por tanto, el programa **CGI** lo primero que debe realizar es la descodificación de la entrada:

- Tiene que separar las distintas parejas ***nombre=valor***. Para lo cual hay que dividir los datos recibidos cada vez que se encuentre un "***&***". No hay peligro de confundirse con la entrada del usuario, ya que si un usuario introduce un ampersand, se envía codificado como "***%26***".
- Una vez que se tienen las distintas parejas, se separan en nombre y valor sabiendo que están separados por el "=". No hay peligro de confundirse con la entrada del usuario, ya que si un usuario introduce un signo igual, se envía codificado como "***%3D***".
- Los signos "+" se sustituyen por espacios en blanco y se interpretan los posibles códigos hexadecimales que puedan aparecer codificados en la forma "***%*****", donde ****** son los códigos hexadecimales. No hay peligro de confundirse con la entrada del usuario, ya que si un usuario introduce un signo porcentaje, se envía codificado como "***%25***".

El CGI puede recibir el contenido de un formulario de dos maneras básicas:

- Mediante la variable de entorno **QUERY_STRING**.
- Mediante la entrada estándar en cuyo caso la variable de entorno **CONTENT_LENGTH**

indicará el largo del string que se debe leer.

Cuál de las dos formas se usa dependerá del método que se haya seleccionado en el parámetro method de la etiqueta form del documento HTML. Para saber de cuál método fue usado se deberá consultar la variable de ambiente **REQUEST_METHOD** que contendrá **GET** o **POST** según sea el caso.

Las principales diferencias entre los métodos **GET** y **POST** son las siguientes:

GET	POST
El CGI lo recibe por medio de una variable de entorno lo que significa una restricción en el tamaño de los datos.	El CGI lo recibe por medio de la entrada estandard (como si fuera teclado) lo que significa virtualmente recibir cualquier tamaño de datos.
El navegador web envía los datos visiblemente haciendo una llamada de la forma: .../cgi-bin/cgi.pl?campo=algo Es la forma como se llaman cgis desde fuera de un formulario, por ejemplo los contadores de visitas son habitualmente llamados de la forma: <img src="Count.cgi?df=cgimaster2"> siendo el resultado:	El navegador web envia los datos directamente al CGI por lo cual no se ven en el navegador web (ideal para campos ocultos)
Al hacer un reload (recarga) de la página el navegador web pasará los parámetros automáticamente.	Al hacer un reload (recarga) de la página el navegador web pasará los parámetros previa confirmación

Ilustración 42: Comparativa GET vs POST

3.5.1.3.3. Variables de entorno CGI.

SERVER_SOFTWARE Formato:nombre/ versión	El nombre y versión del servidor de información (Web Server) que está atendiendo al visitante y que es el padre del proceso actual (CGI). Ejemplo: Apache/2.2.2

SERVER_NAME	El nombre del host (computador donde está el CGI), su nombre en el DNS (algo.pais), o la dirección IP del mismo si el CGI ha sido llamado con este. Ejemplo: comunidad.ok.cl
GATEWAY_INTERFACE Formato: CGI/revisión	La revisión de la especificación para CGIs con la que cumple el servidor. Ejemplo: CGI/1.1
SERVER_PROTOCOL Formato: protocolo/revisión	El nombre y revisión del protocolo de información con que se ha generado la llamada al CGI. Ejemplo: HTTP/1.1
SERVER_PORT	El número de la puerta con que fue hecha la llamada al CGI.
REQUEST_METHOD	El método con que fue hecha la llamada. Para el protocolo HTTP contiene "GET", "HEAD", "POST", etc.
PATH_INFO	Información del path extendida. Los scripts (CGIs) pueden ser llamados usando su path virtual seguido de información extra. Esta información es enviada como PATH_INFO. La misma debe ser decodificada por el servidor si viene desde una URL antes de ser pasada al CGI. Pruebe: ../docs/variables_cgi.shtml/extra/
PATH_TRANSLATED	El servidor entrega una versión traducida del PATH_INFO, que toma el path y hace los traducciones correspondientes, entregando al CGI la dirección correspondiente en el sistema de archivos en que está alojado.
SCRIPT_NAME	El path virtual con que el CGI está siendo ejecutado. Usado para autoreferenciarse. Ejemplo: /cgi-bin/cliente.cgi
QUERY_STRING	La Información que sigue al ? en las llamadas a un CGI. Esta es la información de consulta o parámetros, no está decodificada y es labor del CGI hacerlo. Pruebe: ../docs/variables_cgi.shtml?p1=algo&p2=xxx
REMOTE_HOST	El nombre del host que hace la solicitud. Si el servidor no tiene la

	información, entrega NULL (none). Muchos servidores no lo entregan porque averiguar el nombre del visitante supone una carga extra.
REMOTE_ADDR	La dirección IP del host que hace la solicitud. Varios equipos pueden compartir la misma IP si están detrás de un proxy. En ese caso puede estar seteada la variable HTTP_X_FORWARDED_FOR Ejemplo: 79.144.51.11
AUTH_TYPE	Si el servidor soporta autenticación de usuario y el CGI está protegido, contiene el protocolo especifico de autenticación usado para validar al usuario.
REMOTE_USER	Si el servidor soporta autenticación de usuario y el CGI está protegido, contiene nombre del usuario que está accediendo al cgi.
REMOTE_IDENT	Si el servidor Web soporta identificación RFC 931, esta variable entrega el nombre del usuario remoto tomado desde el servidor.
CONTENT_TYPE	Para llamadas que llevan información agregada, como en el caso de las llamada con POST o PUT, contiene el tipo de los datos.
CONTENT_LENGTH	El largo de los datos enviados por el cliente.

3.5.1.3.4. Acceso a las variables desde C.

Para tener acceso al contenido de las variables de entorno en C [3, 22] se puede emplear la función **getenv**:

- NOMBRE: getenv - obtiene una variable de ambiente.
- SINOPSIS:

  #include <stdlib.h>

  char *getenv(const char *nombre).
- DESCRIPCIÓN: la función **getenv()** busca en la lista de ambiente una cadena de caracteres que concuerde con la apuntada por nombre. Las cadenas son de la forma **nombre = valor.**
- VALOR DEVUELTO: la función **getenv()** devuelve un puntero al valor correspondiente del ambiente, o NULL si no hay concordancia.

En los anexos es posible consultar un ejemplo de como capturar las variables de entorno.

3.5.2. Implementación de la interfaz SMS.

Para que nuestra aplicación cliente pueda recibir peticiones de recomendación a través de un SMS, empleará la interfaz SOAP proporcionada por la plataforma Minerva-RedBox. Como ya se ha explicado la plataforma Minerva-RedBox [20] permite el envío y recepción de mensajes SMS y MMS, así como el envío de peticiones de localización LBS y la recepción de las respuestas asociadas. No obstante, en nuestro proyecto solo se ha implementado el canal SMS, tanto para la recepción de peticiones como para enviar las respuestas asociadas a dicha petición. Dichas respuestas serán las recomendaciones o posibles mensajes de error.

SERVICIO	DESCRIPCIÓN	URL MINERVA
enviosms	Permite el **envío de un SMS** a un número de teléfono determinado	http://193.147.165.85:9002/minerva/envios.asmx/enviosms
enviomms	Permite el **envío de un MMS** a un número de teléfono determinado, **con un máximo de 10 adjuntos**. No es necesario rellenar todos los adjuntos, sólo los que se necesite enviar	http://193.147.165.85:9002/minerva/envios.asmx/enviomms
urlmime	**Detecta el tipo MIME de un fichero presente en la URL que se suministra**. Usa la extensión del fichero	http://193.147.165.85:9002/minerva/envios.asmx/urlmime
urlbase64	**Convierte un fichero** presente en la URL que se suministra **a una cadena codificada en base64**	http://193.147.165.85:9002/minerva/envios.asmx/urlbase64
enviolbs	Permite enviar una petición para **localizar la posición** (geolocalización) **de un terminal móvil**; las coordenadas devueltas vendrán en formato geográfico (latitud/longitud) o UTM	http://193.147.165.85:9002/minerva/envios.asmx/enviolbs
enviolbs_GRLR	Permite enviar una petición de geolocalización inversa, para **obtener una dirección** (calle, ciudad, etc...) **a partir de unas coordenadas** en formato geográfico (latitud/longitud) o UTM	http://193.147.165.85:9002/minerva/envios.asmx/enviolbs_GRLR

Ilustración 43: Servicios de Minerva-RedBox

Las peticiones SMS serán enviadas desde la plataforma Minerva-RedBox hacia nuestra aplicación utilizando SOAP. El esquema de dicho proceso es el siguiente.

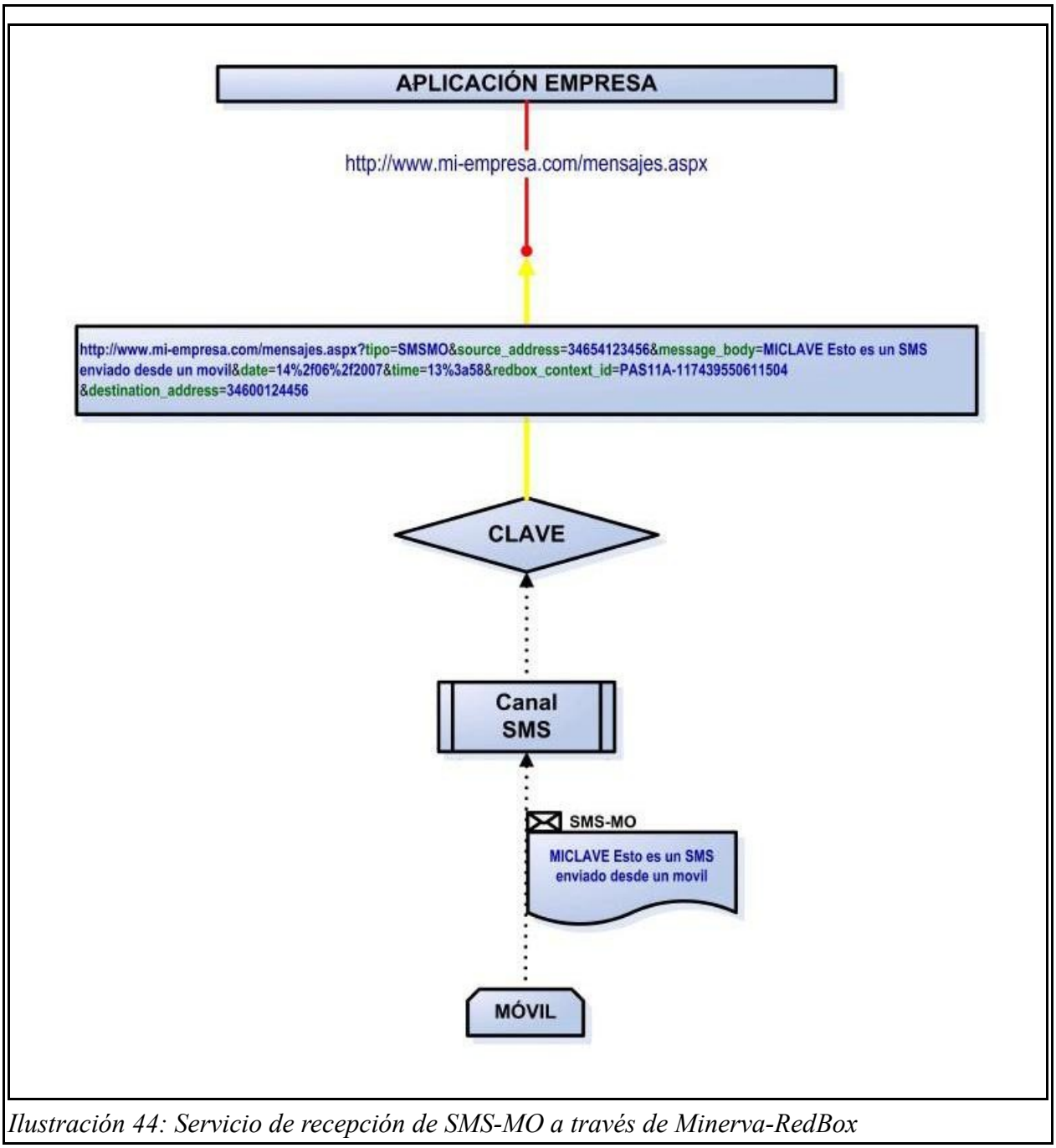

Ilustración 44: Servicio de recepción de SMS-MO a través de Minerva-RedBox

Dado que el servicio que ofrece Minerva dispone de un único número de teléfono, la única forma que se tiene de poder redirigir correctamente a una u otro sistema de información los distintos mensajes entrantes es a través del uso de claves.

Dependiendo de dicha palabra clave (1ª del SMS recibido), invocará la URL de la aplicación que la tenga registrada. La aplicación recibirá en dicha URL una notificación de tipo SMSMO en el caso de SMS entrantes.

Puede observarse que, en el caso del esquema, la notificación se ha enviado por HTTP-GET; es decir, los datos asociados a la notificación se pasan como parte de la query al invocar la URL asociada. También es posible hacerlo mediante el método POST, que es el caso de nuestra aplicación cliente.

Los parámetros que han sido registrados en la plataforma Minerva-RedBox para asociar las peticiones SMS a nuestro sistema de recomendaciones se muestran a continuación.

DATOS DE LA CUENTA DE MINERVA-REDBOX:	
Nombre persona contacto:	(
Email persona contacto	g
Código autentificación servicios web:	t
Clave autentificación servicios web:	I

Ilustración 46: Registro de cuenta en Minerva-RedBox

DATOS DE LA APLICACIÓN DE LA EMPRESA:	
Tecnología SMS	
Código	SMS
Método notificación :	HTTP-POST [X] HTTP-GET [ ]
URL recepción	

Ilustración 47: Método de notificación del canal SMS

CREAVIAJE
INFOVIAJE
MODIFICAVIAJE

Ilustración 46: Palabras clave del servicio SMS de Minerva-RedBox

Los parámetros de los que constará el SMS-MO serán los siguientes:

tipo	"SMSMO"
redbox_context_id	Identificador de Red Box asociado a la petición
source_address	Número del abonado que ha remitido el SMS
destination_address	Número corto del subservicio al que se envió el SMS
message_body	Texto remitido por el terminal móvil
date	Día en el que se ha remitido el SMS
Time	Hora a la que se ha remitido el SMS

Ilustración 48: Parámetros característicos del SMS-MO

Una vez conocidos estos parámetros sólo nos queda introducirlos en nuestro código fuente. Para ello se declaran como constantes simbólicas en nuestro fichero de cabecera principal, "funciones.h". Las funciones necesarias para implementar la interfaz en lenguaje C utilizando SOAP no las tenemos que codificar ya que nos las proporciona GSOAP, empleado para ello el fichero descriptor del Servicio Web especifico de Minerva-RedBox. Dicho fichero como ya se mostró anteriormente es el http://193.147.165.85:9002/minerva/envios.asmx?WSDL. Así mismo los ficheros de cabecera que contienen las declaraciones necesarias para emplear las funciones son "soapH.h" y "EnviosSoap.nsmap". Para tener un mayor conocimiento de las funciones lo recomendable es consultar el manual de referencia de GSOAP, "gSOAP 2.7.11 User Guide" y el código fuente generado por GSOAP.

Hasta ahora hemos hablado de como recibir SMSs, pero nuestra aplicación cliente también enviará SMSs como respuesta a dichas peticiones. Estas respuestas serán la recomendación resultante de la petición o bien un mensaje de error.

Para enviar SMS desde nuestra aplicación habrá que realizar una petición SOAP a una

determinada URL. Dicha petición SOAP constará de un fichero XML que será enviado mediante POST y el cual contendrá una serie de parámetros específicos requeridos por la plataforma Minerva-RedBox.

En el momento de realizar cualquier petición, Minerva-RedBox responderá con un XML, indicando si se ha aceptado la petición (ACK) o si ésta no se ha aceptado (NACK). Las razones por las que no se acepte una petición (NACK) pueden ser varias; entre las más comunes destacamos: el XML que se ha enviado no está bien formado; los datos de autenticación (identificador de empresa/contraseña) que se han enviado no son correctos; el canal de envío indicado no existe, o no es el de la tecnología de la solicitud; la empresa no dispone de créditos para realizar el envío.

La URL para el envío del SMS es URL: http://193.147.165.85:9002/minerva/envios.asmx/enviosms. Y los parámetros se muestran a continuación.

codigo_cuenta	Texto	Código de la cuenta que solicita el envío
clave_cuenta	Texto	Clave de la cuenta que solicita el envío
codigo_canal	Texto	Código del canal por el que se quiere realizar el envío
numero_destino	Texto	Número de tfno. al que se desea enviar el SMS
contenido_sms	Texto / Tamaño máximo: 160 caracteres	Contenido del SMS
permanencia	Numérico	Define el periodo de tiempo que el centro de mensajes de la operadora conserva el SMS intentando la entrega a su destinatario (permanencia). 0: 5 Minutos 1: 15 minutos 2: 1 hora 3:12 horas 4: 1 día 5: 2 días 6:Máximo (1 semana)
fecha_hora_envio	Texto / dd/mm/yyyy hh:mm:ss	Fecha a la que se desea que se envíe el mensaje
Emergente	Texto / "true" ó "false"	Indica si el SMS será de tipo *flash*
Acuse	Texto / "true" ó "false"	Indica si se solicita acuse de recibo

Ilustración 49: Parámetros característicos de un SMS-MT

No es necesario conocer los ficheros XML que intervienen tanto en el envío como en la recepción de un SMS, ya que toda esta complejidad queda oculta al programador gracias a las funciones que genera GSOAP. Por tanto, nosotros sólo debemos preocuparnos por utilizar las funciones adecuadas. Toda la complejidad de SOAP queda simplificada al empleo de ciertas funciones en C. Para aquel que desee conocer las plantillas XML me remito a la documentación de formación proporcionada por Minerva-RedBox y que se adjunta junto con esta memoria, o los anexos.

El esquema que define el envío de un SMS hacia Minerva-RedBox es el siguiente.

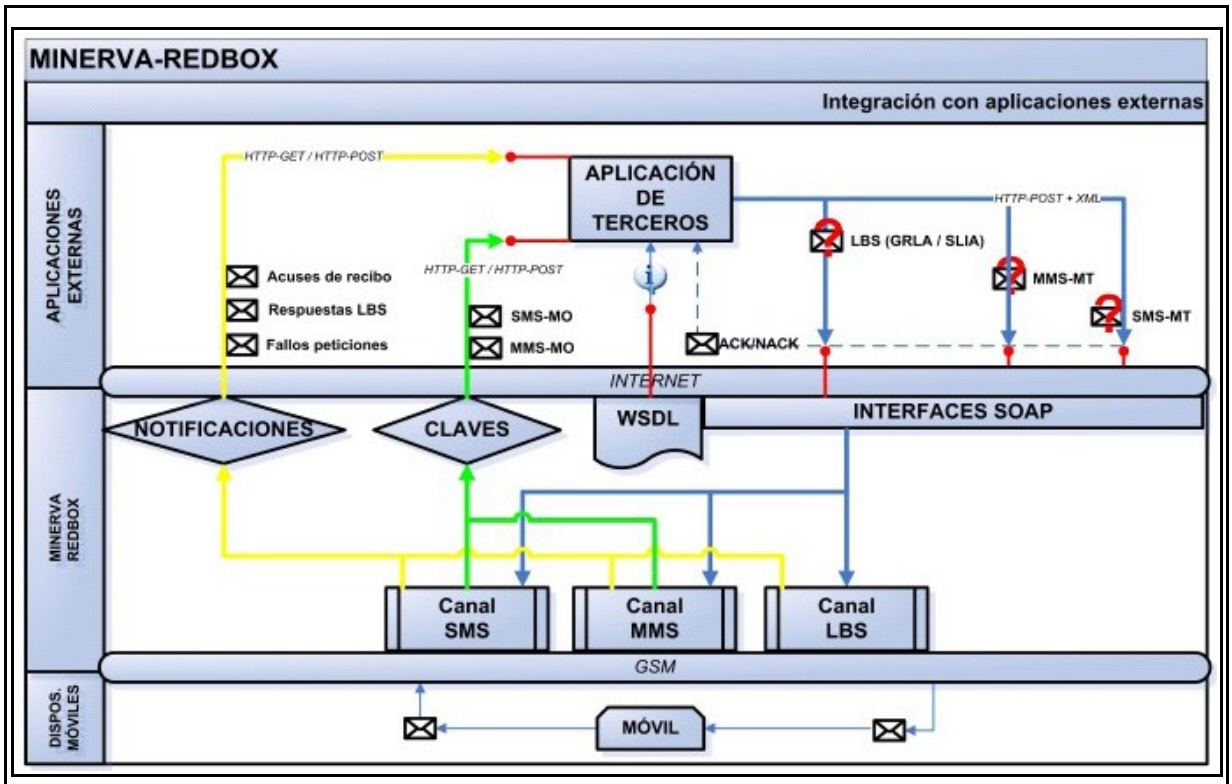

Ilustración 50: Flujo de comunicación desde/hacia un móvil usando Minerva-RedBox

Por consiguiente, para que nuestra aplicación cliente pueda enviar SMSs hacia Minerva-RedBox tan sólo tendrá que hacer uso de las funciones en C que proporciona GSOAP. La función fundamental para ello es "***soap_call___ns1__enviosms(soap,NULL,NULL,envio,respuesta)***".

Evidentemente esta función tiene que ir acompañada de otras funciones y sentencias de C. Para

entender su uso correcto se adjunta en los anexos un ejemplo de código fuente en C, con sus respectivas explicaciones. Tras entender dicho ejemplo se estará preparado para afrontar el código fuente de nuestra aplicación cliente.

3.5.3. Implementación de la comunicación con la aplicación SERVIDOR.

Para lograr la comunicación entre la aplicación CLIENTE y la aplicación SERVIDOR emplearemos la API de red de C. Dicha API nos proporciona un conjunto de funciones, llamadas al sistema y estructuras de datos que usadas correctamente nos permiten comunicar dos procesos entre sí, independientemente de que estén en máquinas diferentes o no. Al mismo tiempo esta API permite el uso de diferentes protocolos de comunicaciones, pero a nosotros sólo nos interesan las comunicaciones entre procesos remotos mediante TCP/IP, que es el protocolo que nuestras aplicaciones utilizarán para intercambiar mensajes entre sí.

Desde una perspectiva generalista y simplificadora los pasos que debe seguir cualquier aplicación cliente para establecer una comunicación con un servidor son los siguientes:

1) Crear un socket.
2) Crear una dirección de destino para el servidor.
3) Conectar con el servidor.
4) Enviar petición de servicio.
5) Recibir resultado del servicio.
6) Cerrar la conexión.

Con el fin de explicar como emplear la API de red de C [1] y ver como funciona, se ha incluido en los anexos una explicación detallada de sus principales funciones, así como el código fuente de un cliente y un servidor de eco. Con la asimilación de estos conceptos se tendrá una base sólida para afrontar el código fuente tanto de la aplicación CLIENTE como del SERVIDOR. Al mismo tiempo servirá como cimiento para crear otras aplicaciones que necesiten comunicarse de forma remota.

3.5.4. Respuesta de la aplicación CLIENTE al usuario final.

Llegados a este punto conocemos como le llegan al cliente las peticiones iniciales de recomendaciones. Bien mediante el empleo de un formulario web, o bien mediante un SMS a través de la plataforma Minerva-RedBox. También sabemos como el cliente se comunica con el servidor para enviarle los parámetros de la petición inicial de recomendación y como recibe los resultados proporcionados por el servidor.

Lo que aún no se ha explicado es cómo el usuario final que realizó la petición inicial recibe el resultado, que el CLIENTE a obtenido a partir del SERVIDOR. Habrá que distinguir nuevamente entre si la petición inicial se realizó vía formulario web, o si por el contrario, se realizó por SMS mediante el uso de Minerva-RedBox.

3.5.4.1. Respuesta al formulario Web.

Si la petición se realizó por formulario, la respuesta se ofrece igualmente a través de una página web donde aparecen las diferentes recomendaciones ofrecidas por el servidor en el caso de que las haya. Esto se consigue incrustando el código HTML que define la web en el código C de la aplicación cliente, que no es más que un script CGI.

Incrustar código HTML en código C es muy sencillo y sólo tenemos que colocarlo en la salida estándar mediante el uso masivo de la función fprint o printf. Esto es así porque al ejecutarse el script CGI (nuestra aplicación cliente) todo lo que se envía por salida estándar es interpretado por el navegador web. De esta forma el navegador web es capaz de interpretar el código HTML que le envía el cliente y construir así la web de respuesta con las recomendaciones.

Un ejemplo sencillo de página web en HTML incrustada en C sería:

#include<stdio.h>

```
int main(){

   printf("Content-Type: text/html\n\n");
   printf("<html>Esto es una página web incrustada en código C</html>\n");

   return( 0 );

}
```

Compilando este código fuente en C debemos obtener un ejecutable con extensión ".**cgi**".

Por ejemplo:

```
$> gcc micodigoC.c -o miscript.cgi
```

Después se colocaría "miscript.cgi" en la carpeta del servidor adecuada para la ejecución de CGIs, y al ejecutarlo desde un navegador web este interpretaría la siguiente código HTML:

```
<html>Esto es una página web incrustada en código C</html>
```

Por tanto, esta es la técnica que empleamos para ofrecer las recomendaciones ofrecidas por el servidor. Para ver en detalle lo que realiza la aplicación cliente hay que consultar el código fuente en los anexos. Dicho código fuente viene comentado al detalle por lo que es fácil de comprender lo que se está haciendo en cada momento.

En la siguiente ilustración se muestra la respuesta que recibe el usuario final que ha realizado la petición de recomendación a través de un formulario web a través de Internet. Como se puede ver se ofrecen diferentes recomendaciones para el itinerario Sevilla-Madrid.

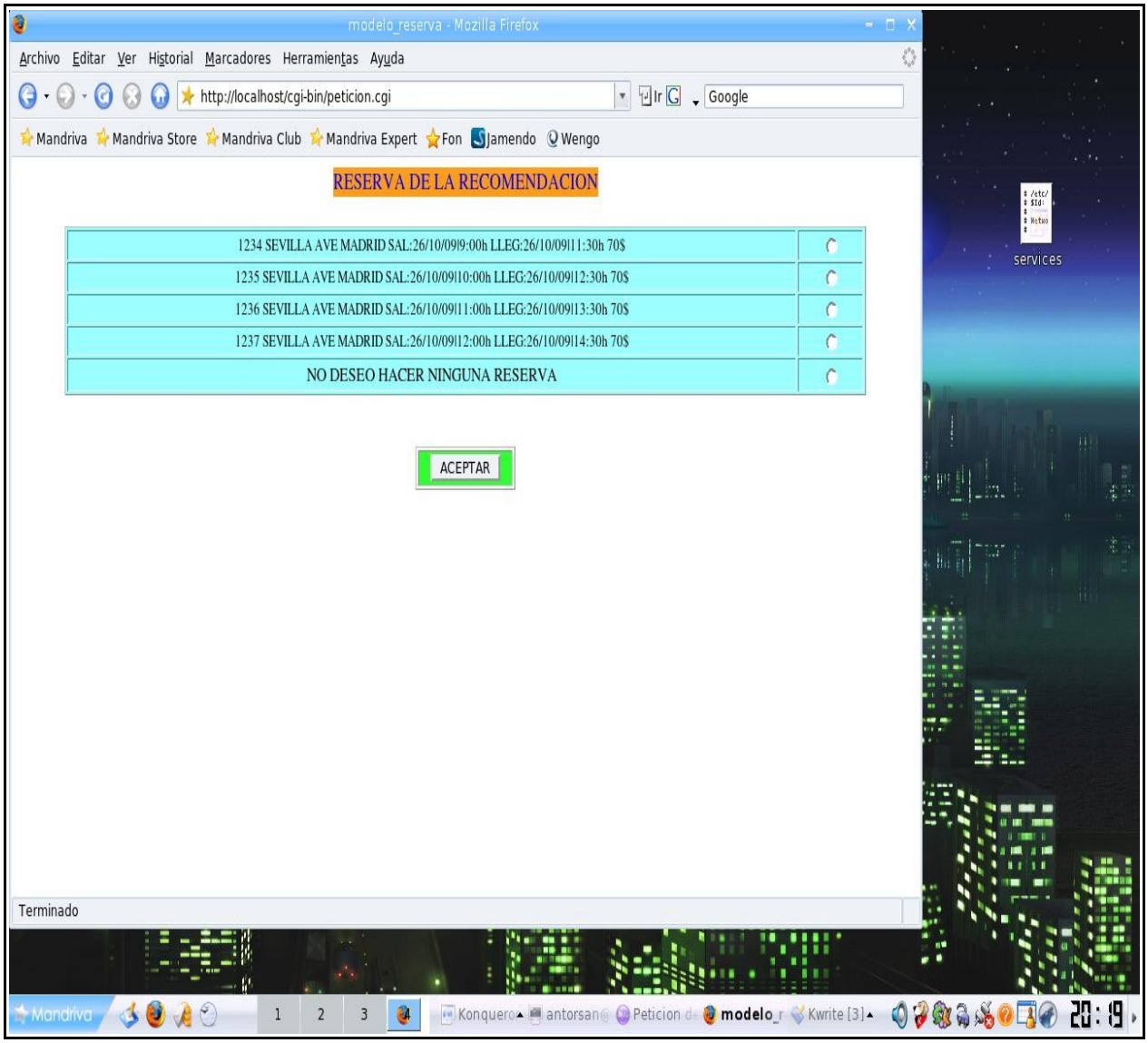

Ilustración 51: Respuesta a la petición de recomendación por la interfaz web

3.5.4.2. Respuesta al SMS.

Si la petición inicial se realizó por SMS a través de Minerva-RedBox, la respuesta se realiza mediante el envío de uno o varios SMS al mismo número de teléfono. Ello se consigue mediante el uso de funciones de C proporcionadas por gSOAP, y que ya se ha explicado con anterioridad. Nuevamente se ha de consultar el código fuente para ver los detalles y comprender paso a paso lo que se hace.

3.6. IMPLEMENTACIÓN DE LA APLICACIÓN SERVIDOR.

La aplicación SERVIDOR está implementada igualmente en lenguaje C. Esta aplicación no calcula las recomendaciones a partir de los parámetros que le envía la aplicación CLIENTE, ya que este aspecto queda fuera del ámbito de este proyecto. Esta aplicación está diseñada e implementada para que pueda comunicarse con el CLIENTE mediante el protocolo de mensajes definidos.

El SERVIDOR está implementado para ser disertado como un módulo más en la aplicación que realmente calcule las recomendaciones. Actúa como una mera interfaz de comunicación que implementa el protocolo definido, recibiendo los parámetros necesarios para el cálculo de la recomendación y pasándoselos a al módulo que realice los cálculos de recomendaciones. A continuación, recibiría las recomendaciones calculadas en el caso de que las haya (podría ocurrir que no existiesen recomendaciones) o la correspondiente señal de error, en el caso de no poder calcular las recomendaciones. Todas las posibilidades están recogidas en los diferentes mensajes que se definen en el protocolo implementado. Y por último, enviaría las recomendaciones a la aplicación cliente o un mensaje indicando el por qué no se pudo calcular la recomendación. Para entender estos mensajes se recomienda ver el apartado donde aparecen los diagramas de paso de mensajes (1.1.2 Diagrama de paso de mensajes del protocolo CLIENTE-SERVIDOR).

En nuestro caso para poder realizar las simulaciones de recepción y envío de las recomendaciones el servidor ha sido implementado para que realice lo siguiente:

1. Al ejecutarse se le indicará por argumentos en la linea de comandos el puerto por el que ofrece el servicio, seguido de un “1” si queremos simular el caso en el que los parámetros enviados son correctos, otro número que simula el número de intentos que se realizarán para el "cálculo" de las recomendaciones, y por último, un número que simula el estado en el que se encontraría la aplicación que debería calcular las simulaciones.

Ej: Simulación de una petición de recomendación con éxito: server.exe 65535 1 0 1

Ej: Simulación de una petición de recomendación con éxito tras dos intentos de cálculo de la recomendación: server.exe 65535 1 2 1

Ej: Simulación de una petición de recomendación que recibe un asentimiento negativo por parte del servidor.

2. Una vez arranca el servidor este permanece a la escucha de peticiones. Las peticiones son aceptadas en cola ,es decir, no hay multihilo. Si no escucha nada, finaliza el proceso y debe ser arrancado de nuevo.
3. Al establecer una comunicación con el cliente lo indica por la pantalla de la consola. Y también imprime por pantalla todo el dialogo con el cliente. De esta forma se puede seguir todo el intercambio de mensajes del protocolo. Como no se lleva a cabo un cálculo de las recomendaciones, la aplicación lee las recomendaciones de un fichero que las contiene y que fue creado para poder llevar a cabo las simulaciones. Dicho fichero se colocará en el mismo directorio donde se encuentra la aplicación servidor bajo el nombre de "Recomendaciones.txt". El contenido de este fichero serán las recomendaciones que queremos enviar al cliente.
4. Al finalizar la comunicación con un cliente vuelve a esperar más peticiones.

Estos 4 pasos describen a grandes rasgos la forma de actuar de la aplicación servidor, pero para entender todos los casos y pasos que lleva a cabo es obligatorio revisar el código fuente de dicha aplicación, que se adjunta en los anexos.

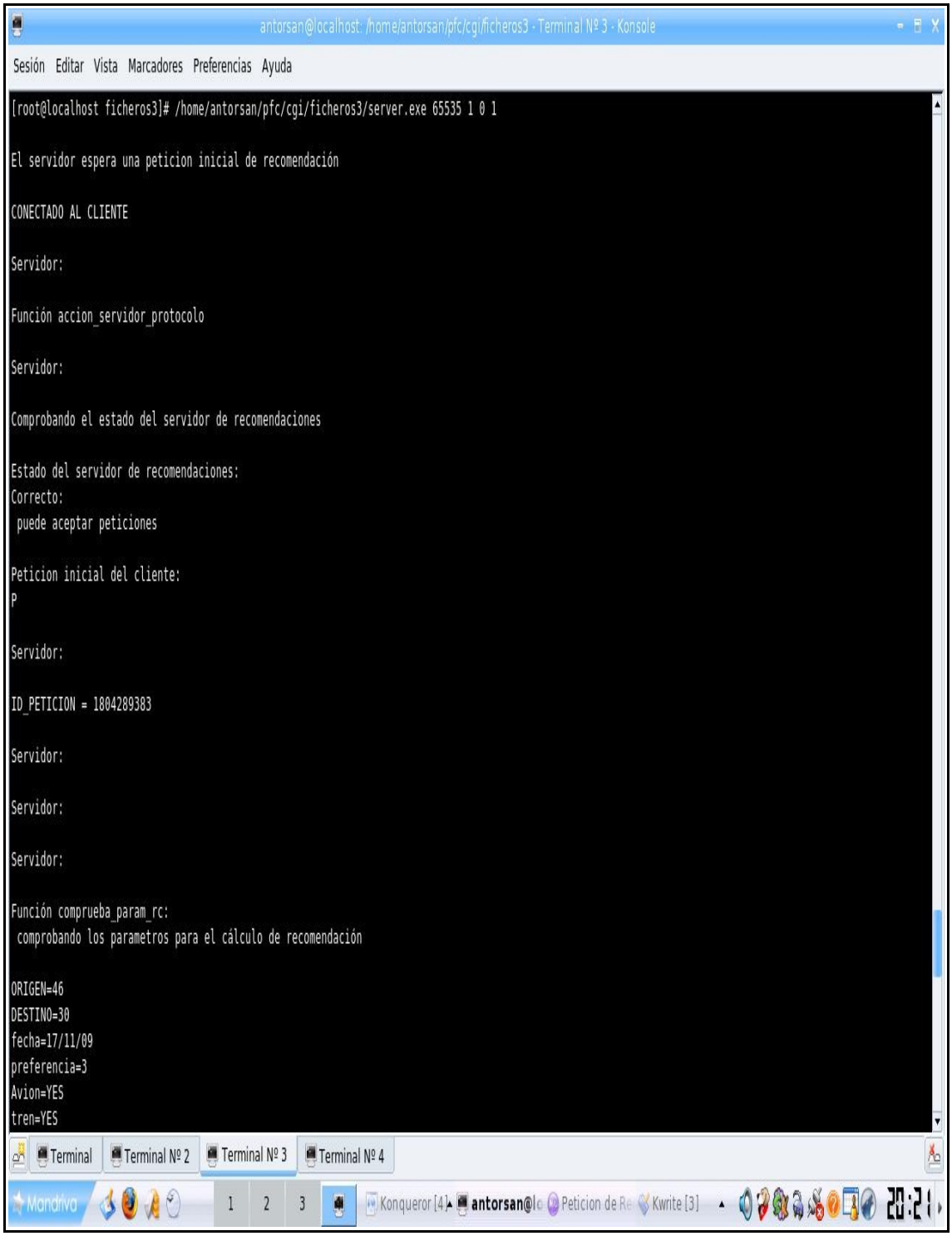

Ilustración 52: Aplicación SERVIDOR aceptando una petición de recomendación

A continuación se muestra una serie de capturas de pantalla que muestran por consola la forma de proceder de la aplicación SERVIDOR, ya sea para una petición por SMS o a través de Internet.

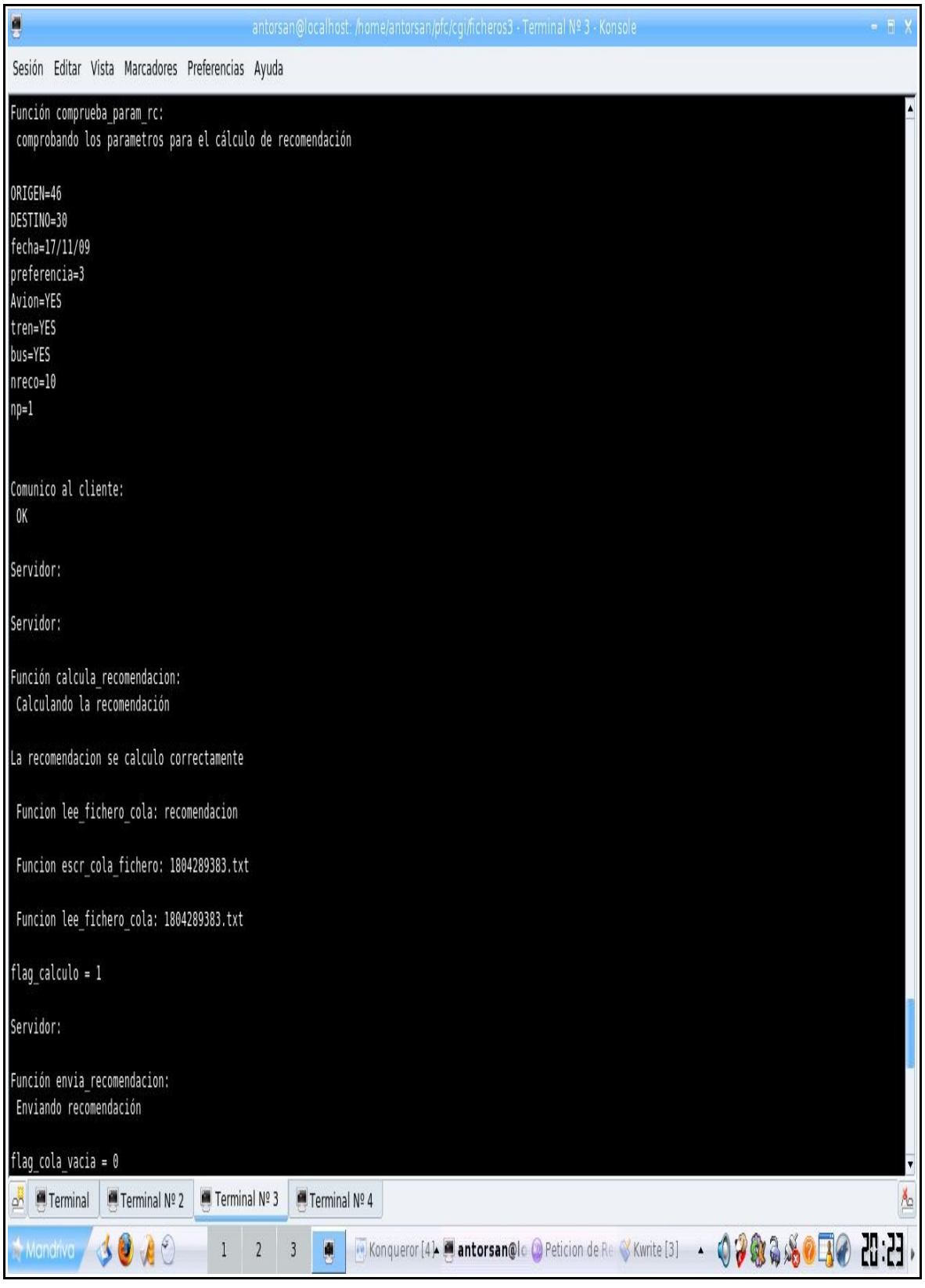

Ilustración 53: Aplicación SERVIDOR calculando recomendaciones

Ilustración 54: Aplicación SERVIDOR enviando recomendaciones al CLIENTE