4. DISEÑO E IMPLEMENTACIÓN

Como venimos señalado a lo largo de esta memoria, cuando un usuario accede a un Sistema de Información, su acceso debe ser controlado por distintos agentes, que aseguran que el uso de la información presente en el sistema se haga sólo por usuarios autorizados.

Para que posteriormente pueda comprobarse qué usuarios han accedido al sistema, en qué situaciones y qué acciones han llevado a cabo dentro del mismo se precisan nuevos agentes. Estas funciones recaerán en un servidor encargado de recopilar y controlar la base de datos que almacenará la información.

La base de datos diseñada en este proyecto recibe el nombre de "Registro de Acceso" y en su implementación se ha tenido en cuenta todas las indicaciones aparecidas en la legislación actual referente al tratamiento de ficheros automatizados que contengan datos de carácter personal.

Para el diseño del Servidor de Registro de Acceso se han tenido en cuenta los artículos mostrados en la siguiente legislación:

- Ley 5/1992, de 29 de Octubre, de regulación del tratamiento automatizado de datos de carácter personal (LORTAD). Legislación derogada por la Ley 15/1999, pero que muestra las primeras líneas de actuación al tratamiento de datos de carácter personal.

- Real Decreto 994/1999, de 11 de Junio, por el que se aprueba el reglamento de medidas de seguridad de los ficheros automatizados que contengan datos de carácter personal. Cataloga los datos de carácter personal y señala las mínimas medidas de seguridad que deben tener los Sistemas de Información en función de la categoría de datos que contengan.

- Ley 15/1999, de 13 de Diciembre, de protección de datos de carácter personal (LOPD). En ella se recoge los procedimientos de obtención de datos y los derechos y responsabilidades de los interesados.

- Ley 41/2002, de 14 de Noviembre, básica, reguladora de la autonomía del paciente y de derechos y obligaciones en materia de información y documentación clínica. Legislación encaminada a remarcar los derechos de los pacientes y obligaciones de los profesionales sanitarios en cuanto al documento de historia clínica se refiere.

Los artículos de la nombrada legislación que hayan motivado la inclusión de algún aspecto o elemento al diseño del servidor o de la base de datos serán destacados en esta memoria.

4.1. Base de Datos "Registro de Acceso".-

La base de datos "Registro de Acceso" será la encargada de almacenar la información referente a los accesos que se produzcan en el dominio de seguridad. La creación de esta base de datos está basada, especialmente, en el Real Decreto 994/1999.

Según Real Decreto 994/1999 se establecen varios niveles de seguridad en cuanto a la protección de los datos se refiere. A los relacionados con la salud les corresponde el nivel más alto. Esto conlleva que debe cumplir las exigencias de los niveles anteriores, bajo y medio , mas unas condiciones especiales. Los niveles de seguridad que se describen en el Real Decreto tienen la condición de mínimos exigibles.

A continuación se presenta el cuadro resumen, proporcionado por la Agencia Española de Protección de Datos, de las medidas de seguridad introducidas por el Real Decreto 994/1999.

AGENCIA ESPAÑOLA DE PROTECCIÓN DE DATOS

CUADRO RESUMEN MEDIDAS DE SEGURIDAD

Reglamento de medidas de seguridad de los ficheros que contengan datos de carácter personal (RD 994/1999)

Nivel básico: Ficheros que contengan datos de carácter personal.
Nivel medio: Ficheros que contengan datos relativos a la comisión de infracciones administrativas o penales, Hacienda Pública, servicios financieros y los que se rijan por el artículo 29 de la LOPD (prestación de servicios de solvencia y crédito).
Nivel alto: Ficheros que contengan datos de ideología, religión, creencias, origen racial, salud o vida sexual así como los recabados para fines policiales sin consentimiento de las personas afectadas.

	NIVEL BÁSICO	NIVEL MEDIO	NIVEL ALTO
DOCUMENTO DE SEGURIDAD	- Ambito de aplicación. - Medidas, normas, procedimientos reglas y estándares de seguridad. - Funciones y obligaciones del personal. - Estructura y descripción de ficheros y sistemas de información. - Procedimiento de notificación, gestión y respuesta ante incidencias. - Proced. realización copias de respaldo y recuperación de datos.	- Identificación del responsable de seguridad. - Control periódico del cumplimiento del documento. - Medidas a adoptar en caso de reutilización o desecho de soportes.	
PERSONAL	- Funciones y obligaciones claramente definidas y documentadas. - Difusión entre el personal, de las normas que les afecten y de las consecuencias por incumplimiento.		
INCIDENCIAS	- Registrar tipo de incidencia, momento en que se ha producido, persona que la notificia, persona a la que se comunica y efectos derivados.	- Registrar realización de procedimientos de recuperación de los datos, persona que lo ejecuta, datos restaurados y grabados manualmente. - Autorización por escrito del responsable del fichero para su recuperación.	
IDENTIFICACIÓN Y AUTENTICACIÓN	- Relación actualizada de usuarios y accesos autorizados. - Procedimientos de identificación y autenticación. - Criterios de accesos. - Procedimientos de asignación y gestión de contraseñas y periodicidad con que se cambian. - Almacenamiento ininteligible de contraseñas activas.	- Se establecerá el mecanismos que permita la identificación de forma inequívoca y personalizada de todo usuario y la verificación de que está autorizado. - Límite de intentos reiterados de acceso no autorizado.	
CONTROL DE ACCESO	- Cada usuario accederá únicamente a los datos y recursos necesarios para el desarrollo de sus funciones. - Mecanismos que eviten el acceso a datos o recursos con derechos distintos de los autorizados. - Concesión de permisos de acceso sólo por personal autorizado.	- Control de acceso físico a los locales donde se encuenetren ubicados los sistemas de información.	
GESTIÓN DE SOPORTES	- Identificar el tipo de información que contienen. - Inventario. - Almacenamiento con acceso restringido. - Salida de soportes autorizada por el responsable del fichero.	- Registro de entrada y salida de soportes. - Medidas para impedir la recuperación posterior de información de un soporte que vaya a ser desechado o reutilizado. - Medidas que impidan la recuperación indebida de la información almacenada en un soporte que vaya a salir como consecuencia de operaciones de mantenimiento.	- Cifrado de datos en la distribución de soportes.
COPIAS DE RESPALDO	- Verificar la definición y aplicación de los procedimientos de copias y recuperación. - Garantizar la reconstrucción de los datos en el estado en que se encontraban en el momento de producirse la pérdida o destrucción. - Copia de respaldo, al menos semanal.		- Copia de respaldo y procedimientos de recuperación en lugar diferente del que se encuentren los equipos.
RESPONSABLE		- Uno o varios nombrados por el responsable del fichero. - Encargado de coordinar y controlar las medidas del documento. - No supone delegación de responsabilidad del responsable del fichero.	
PRUEBAS		- Solo se realizarán si se asegura el nivel de seguridad correspondiente al tipo de fichero tratado.	
AUDITORIA		- Bianual, interna o externa. - Adecuación de las medidas y controles. - Deficiencias y propuestas correctoras. - Análisis del responsable de seguridad y conclusiones al responsable del fichero. - Adopción de las medidas correctoras adecuadas.	
REGISTRO DE ACCESOS			- Registrar usuario, hora, fichero, tipo acceso y registro accedido. - Control del responsable de seguridad. Informe mensual. - Conservación 2 años.
TELECOMUNICACIONES			- Transmisión de datos cifrada.

- Los niveles son acumulativos y tienen la condición de mínimos exigibles.
- Los accesos a través de redes de telecomunicaciones deben garantizar un nivel de seguridad equivalente al de los accesos en modo local.
- La ejecución de trabajos fuera de los locales de la ubicación del fichero debe ser expresamente autorizada por el responsable del fichero y garantizar el nivel de seguridad.
- Los ficheros temporales deberán cumplir el nivel de seguridad correspondiente y serán borrados una vez que hayan dejado de ser necesarios.
- Los ficheros de nivel básico que contengan datos que permitan obtener una evaluación de la personalidad del individuo deberan garantizar, además de las medidas de nivel básico, las de nivel medio relativas a auditoria, identificación y autenticación, control de acceso físico y gestión de soportes.

Figura 4.1. Cuadro resumen medidas de seguridad del Real Decreto 994/1999.

Cumpliendo rigurosamente lo señalado en el Real Decreto 994/1999, diseñamos las tablas y campos de la base de datos “Registro de Acceso”.

4.1.1. Tablas.-

Se han creado quince tablas de las cuales seis pueden considerarse como principales, seis como auxiliares y tres dedicadas al control de usuarios del Servidor de Registro de Acceso.

Las tablas principales reciben el nombre de "registro_de_..." y en ellas se almacenan los datos sobre accesos, incidencias e intercambios de información generados por los usuarios del Sistema de Información. Digamos que estas tablas serán las que almacenan todos los datos dinámicos que se producen en el sistema.

Las tablas auxiliares recogen la descripción de los datos introducidos en las tablas principales y son por tanto tablas que facilitan el almacenaje de la información. Los valores de estas tablas podrán ser modificados para un ajuste más preciso.

Por último las tablas dedicadas al control de usuarios del Servidor de Registro de Acceso recogerán la identificación de los usuarios autorizados, sus credenciales y la descripción de los recursos que ofrece el servidor.

Vemos en la siguiente figura los nombres proporcionados a las tablas de la base de datos "Registro de Acceso".

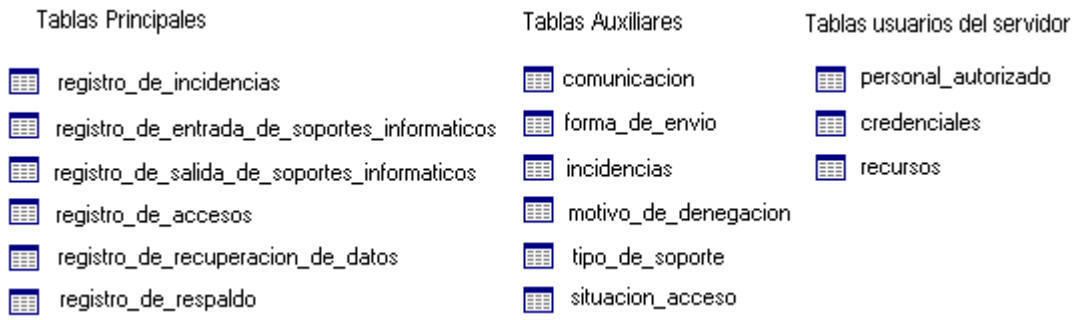

Figura 4.2. Tablas de la base de datos "Registro de Acceso".

4.1.1.1. Tabla: "registro_de_incidencias".-

Según se describe en el artículo 2 del Real Decreto 994/1999, se entiende por incidencia "*cualquier anomalía que afecte o pudiera afectar a la seguridad de los datos".* Por tanto el registro de incidencias recogerá aquellas incidencias relacionadas exclusivamente con la seguridad de los datos almacenados en los recursos. Así fallos en el servidor, en el browser, etc., que no representen una amenaza para los ficheros no serán registrados.

Según se describe en el artículo 10 del Real Decreto 994/1999 este registro debe incluir:

a. Tipo de incidencia.
b. Fecha y hora del instante de la incidencia.
c. Identificación de la persona que realiza la notificación.
d. A quién se le comunica la incidencia.
e. Posibles efectos derivados de la misma.

Además como se señala en el artículo 21 del Real Decreto 994/1999, deberán considerarse los procedimientos realizados de recuperación de datos indicando:

f. Persona que ejecutó el proceso.
g. Qué datos se restauraron
h. Datos grabados manualmente.
i. Autorización por escrito del responsable del fichero.

Para cumplir con estas exigencias se han diseñado las tablas “registro_de_incidencias” y “registro_de_recuperación_datos”.

La definición, en PostgreSQL, de la tabla “registro_de_incidencias” se muestra a continuación. En ella, como en todas las tablas diseñadas, se muestra una breve descripción de la función que desempeña cada campo y se diferencia si su diseño responde a las exigencias recogidas en la legislación o a la aportación del proyectante.

```
CREATE TABLE registro_de_incidencias
(
  numero_incidencia serial NOT NULL,
  codigo_incidencia char(5),
  fecha date,
  hora time,
  notificante varchar(50),
  codigo_comunicacion char(5),
  responsable_seguridad varchar(50),

  CONSTRAINT registro_de_incidencias_pkey PRIMARY KEY (numero_incidencia),
  CONSTRAINT registro_de_incidencias_codigo_comunicacion_fkey FOREIGN KEY (codigo_comunicacion)
  REFERENCES comunicacion (codigo_comunicacion) ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT registro_de_incidencias_codigo_incidencia_fkey FOREIGN KEY (codigo_incidencia)
  REFERENCES incidencias (codigo_incidencia) ON UPDATE NO ACTION ON DELETE NO ACTION
)

WITHOUT OIDS;
ALTER TABLE registro_de_incidencias OWNER TO postgres;

COMMENT ON COLUMN registro_de_incidencias.numero_incidencia IS
 'Valor añadido. Enumera las incidencias de seguridad que se producen.';

COMMENT ON COLUMN registro_de_incidencias.codigo_incidencia IS
 'Valor añadido. Recogerá el código del tipo de incidencia que se ha producido.';

COMMENT ON COLUMN registro_de_incidencias.fecha IS
 'Real Decreto 994/1999. Articulo 10. Recibirá la fecha en el momento de producirse una incidencia.';

COMMENT ON COLUMN registro_de_incidencias.hora IS
 'Real Decreto 994/1999. Articulo 10. Recibirá la hora en el momento de producirse una incidencia.';

COMMENT ON COLUMN registro_de_incidencias.notificante IS
 'Real Decreto 994/1999. Articulo 10. Recogerá quien que notifica la incidencia.';

COMMENT ON COLUMN registro_de_incidencias.codigo_comunicacion IS
 'Valor Añadido. Recogerá el código de las distintas posibilidades de comunicación de la incidencia.';

COMMENT ON COLUMN registro_de_incidencias.responsable_seguridad IS
 'Valor añadido. Persona que por ley se responsabiliza de la seguridad de los ficheros.';
```

Como se deduce de la definición, la tabla “registro_de_incidencias” está relacionada con las tablas auxiliares “incidencias” y “comunicación” mediante los campos “codigo_incidencia” y ”codigo_comunicación” respectivamente.

Como va a ir apareciendo a lo largo de esta memoria, las tablas principales se relacionan con las tablas auxiliares a partir de campos con nombre “codigo_...”. La misión de estos códigos es la de optimizar la cantidad de datos almacenados en la base de datos, además de proporcionar una organización sobre la información que puede recibir nuestro servidor. Será siempre más eficiente que se nos comunique que se ha producido la “I-600”, que almacenar el texto “Se ha producido una incidencia de recuperación de datos”.

La tabla “incidencias” asigna a cada código de incidencia establecido por el administrador del servidor, la descripción de la de incidencia y de los posibles efectos que de ella pudieran surgir. La definición, en PostgreSQL, de la tabla “incidencias” es la siguiente:

```
CREATE TABLE incidencias
(
  codigo_incidencia char(5) NOT NULL,
  descripcion varchar(200) NOT NULL,
  efectos varchar(200) NOT NULL,

  CONSTRAINT incidencias_pkey PRIMARY KEY (codigo_incidencia)
)

WITHOUT OIDS;
ALTER TABLE incidencias OWNER TO postgres;

COMMENT ON COLUMN incidencias.codigo_incidencia IS
 'Valor añadido. Recogerá el código del tipo de incidencia que se ha producido.';

COMMENT ON COLUMN incidencias.descripcion IS
 'Real Decreto 994/1999. Articulo 10. Describe el tipo de incidencia que se ha producido.';

COMMENT ON COLUMN incidencias.efectos IS
 'Real Decreto 994/1999. Articulo 10. Recogerá los efectos que deriven de la incidencia que se ha producido.';
```

La tabla “registro_de_incidencias” presenta otra relación con la tabla “comunicación” por medio del campo “codigo_comunicacion”. En la tabla “comunicacion” se asigna a un código la persona o módulo al que se comunica la incidencia. Su definición es la siguiente:

```
CREATE TABLE comunicacion
(
  codigo_comunicacion char(5) NOT NULL,
  descripcion varchar(200) NOT NULL,

 CONSTRAINT comunicacion_pkey PRIMARY KEY (codigo_comunicacion)
)

WITHOUT OIDS;
ALTER TABLE comunicacion OWNER TO postgres;

COMMENT ON COLUMN comunicacion.codigo_comunicacion IS
'Valor Añadido. Recogerá el código de las distintas posibilidades de comunicación de la incidencia.';

COMMENT ON COLUMN comunicacion.descripcion IS
```

```
'Real Decreto 994/1999. Articulo 10. Recogerá a quien se le comunica  la incidencia.';
```

En el artículo 21 del Real Decreto 994/1999, se señala que debe considerarse en el registro de incidencias los procedimientos realizados de recuperación de datos. Para tener en cuenta este artículo se diseño la tabla "registro_de_recuperacion_de_datos" cuyos campos complementan a los de la tabla "registro_de_incidencias" cuando el código de incidencia comunicado es el correspondiente a la recuperación de datos. La relación entre ambas tablas se realiza por medio del campo "numero_incidencia".

La definición, en PostgreSQL, de esta tabla es la siguiente:

```
CREATE TABLE registro_de_recuperacion_de_datos
(
  numero_incidencia serial NOT NULL,
  id_usuario varchar(50),
  datos text,
  datos_manuales text,
  autorizada bool,

  CONSTRAINT registro_de_recuperacion_de_datos_numero_incidencia_fkey FOREIGN KEY (numero_incidencia)
  REFERENCES registro_de_incidencias (numero_incidencia) ON UPDATE NO ACTION ON DELETE NO ACTION
)

WITHOUT OIDS;
ALTER TABLE registro_de_recuperacion_de_datos OWNER TO postgres;

COMMENT ON COLUMN registro_de_recuperacion_de_datos.numero_incidencia IS
 'Valor añadido. Enumera las incidencias de seguridad que se producen.';

COMMENT ON COLUMN registro_de_recuperacion_de_datos.id_usuario IS
 'Real Decreto 994/1999. Articulo 21. Persona que ejecuto el proceso de recuperación de datos.';

COMMENT ON COLUMN registro_de_recuperacion_de_datos.datos
IS 'Real Decreto 994/1999. Articulo 21. Datos restaurados.';

COMMENT ON COLUMN registro_de_recuperacion_de_datos.datos_manuales IS
 'Real Decreto 994/1999. Articulo 21. Datos grabados manualmente en el proceso de recuperación.';

COMMENT ON COLUMN registro_de_recuperacion_de_datos.autorizada IS
'Real Decreto 994/1999. Articulo 21. Para la ejecución de los procedimientos de recuperación de datos es necesaria
la autorización por escrito del responsable del fichero.';
```

En el siguiente gráfico vemos las relaciones de la tabla "registro_de_incidencias".

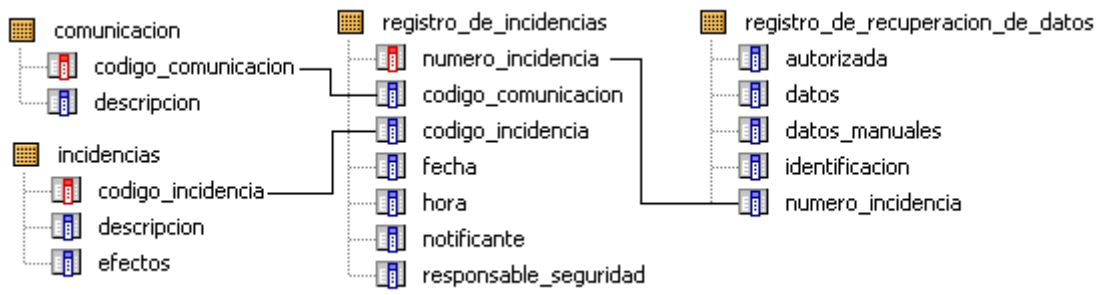

Figura 4.3. Relaciones de la tabla "registro_de_incidencias".

4.1.1.2. Tabla: “registro_de_entrada_de_soportes_informáticos”.-

Según se describe en el artículo 20 del Real Decreto 994/1999 deberá establecerse un sistema de registro de entrada de soportes informáticos, que permita directa o indirectamente conocer:

a) Tipo de soporte informático.
b) Fecha y hora.
c) Emisor.
d) Número de soportes.
e) Tipo de Información que contiene.
f) Forma de envío.
g) Persona responsable de recepción.
h) Autorización de la persona responsable de recepción.

Para cumplir con estas exigencias se ha diseñado la tabla “registro_de_entrada_de_soportes_ informaticos”, la cual estará relacionada, siguiendo las mismas pautas vistas en las tablas anteriores, con las tablas auxiliares “forma_de_envio” y “tipo_de_soporte”.

```
CREATE TABLE registro_de_entrada_de_soportes_informaticos
(
  numero_entrada serial NOT NULL,
  codigo_soporte char(5),
  responsable_seguridad varchar(50),
  emisor varchar(50),
  receptor varchar(50),
  autorizacion_escrita bool,
  fecha date,
  hora time,
  numero_soportes numeric,
  codigo_envio char(5),

CONSTRAINT registro_de_entrada_de_soportes_informaticos_pkey PRIMARY KEY (numero_entrada),
CONSTRAINT registro_de_entrada_de_soportes_informatico_codigo_soporte_fkeyFOREIGN KEY (codigo_soporte)
REFERENCES tipo_de_soporte (codigo_soporte) ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT  registro_de_entrada_de_soportes_informaticos_codigo_envio_fkey  FOREIGN  KEY  (codigo_envio)
REFERENCES forma_de_envio (codigo_envio) ON UPDATE NO ACTION ON DELETE NO ACTION
)

WITHOUT OIDS;
ALTER TABLE registro_de_entrada_de_soportes_informaticos OWNER TO postgres;

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.numero_entrada IS
 'Valor añadido. Enumera las entradas de soportes que se producen.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.codigo_soporte IS
 'Valor añadido. Recogerá el código del tipo de soporte informático recibido.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.responsable_seguridad IS
 'Valor añadido. Persona que por ley se responsabiliza de la seguridad de los ficheros.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.emisor IS
 'Real Decreto 994/1999. Artículo 20. Identifica al emisor de la información.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.receptor IS
 'Real Decreto 994/1999. Artículo 20. Identifica al receptor de la información.';
```

```
COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.autorizacion_escrita IS
 'Real Decreto 994/1999. Artículo 20. Recoge si el receptor posee la autorización.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.fecha IS
 'Real Decreto 994/1999. Articulo 20. Recibirá la fecha en el momento de producirse la entrada del soporte.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.hora IS
 'Real Decreto 994/1999. Articulo 20. Recibirá la hora en el momento de producirse la entrada del soporte.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.numero_soportes IS
 'Real Decreto 994/1999. Artículo 20. Recoge el número de soportes recibidos.';

COMMENT ON COLUMN registro_de_entrada_de_soportes_informaticos.codigo_envio IS
 'Valor añadido. Recogerá el código de envío que se relaciona con las distintas formas de envio de los ficheros.';
```

En la definición de la tabla “registro_de_entrada_de_soportes_informaticos” aparece la relación con la tabla auxiliar “forma_de_envio”. El campo “codigo_envio” recoge el código que se relaciona con las distintas formas de envío de ficheros.

La tabla “forma_de_envio” asigna a cada código un tipo de envío. La definición de la tabla se muestra a continuación:

```
CREATE TABLE forma_de_envio
(
  codigo_envio char(5) NOT NULL,
  descripcion varchar(200) NOT NULL,
  CONSTRAINT forma_de_envio_pkey PRIMARY KEY (codigo_envio)
)

WITHOUT OIDS;
ALTER TABLE forma_de_envio OWNER TO postgres;

COMMENT ON COLUMN forma_de_envio.codigo_envio IS
'Valor añadido. Recogerá el código de envío que se relaciona con las distintas formas de envío de los ficheros.';

COMMENT ON COLUMN forma_de_envio.descripcion IS
 'Real Decreto 994/1999. Artículo 20. Describe la forma de envío de la información.';
```

La tabla “tipo_de_soporte” se diseña para recoger todos los tipos de soportes que pueden ser transferidos o enviados en el Sistema de Información. Esta tabla se encuentra relacionada con la tabla “registro_de_entrada_de_soportes_informaticos” por medio del campo “codigo_soporte”. Su definición, en PostgreSQL, es la siguiente:

```
CREATE TABLE tipo_de_soporte
(
  codigo_soporte char(5) NOT NULL,
  descripcion varchar(200) NOT NULL,
  CONSTRAINT tipo_de_soporte_pkey PRIMARY KEY (codigo_soporte)
)
WITHOUT OIDS;
ALTER TABLE tipo_de_soporte OWNER TO postgres;

COMMENT ON COLUMN tipo_de_soporte.codigo_soporte IS
 'Valor añadido. Recogerá el código del tipo de soporte.';

COMMENT ON COLUMN tipo_de_soporte.descripcion IS
```

'Real Decreto 994/1999. Articulo 20. Describe el tipo de soporte.';

En la figura 4.4 podemos observar la relación existente entre las tablas descritas.

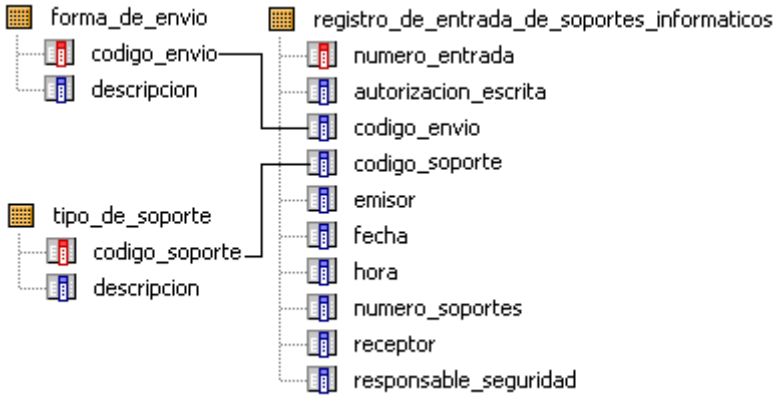

Figura 4.4. Relaciones de la tabla "registro_de_entrada_de_soportes_informaticos".

4.1.1.3. Tabla: "registro_de_salida_de_soportes_informáticos".-

En el artículo 20 del Real Decreto 994/1999 se indica que deberá establecerse un sistema de registro de salida de soportes informáticos, que permita directa o indirectamente conocer:

a) Tipo de soporte informático.
b) Fecha y hora.
c) Receptor.
d) Número de soportes.
e) Tipo de Información que contiene.
f) Forma de envío.
g) Persona responsable de emisión.
h) Autorización de la persona responsable de emisión.

Con el fin de cumplir con estas exigencias se ha diseñado la tabla "registro_de_salida_de_soportes_informaticos", cuyos campos y relaciones son similares a los señalados en la tabla "registro_de_entrada_de_soportes_informaticos" . Aunque se podría pensar en un vínculo entre los datos almacenados en ambas tablas, la verdad es que ambos procesos de intercambio de información son independientes y por lo tanto la introducción de los datos se hace de forma autónoma.

La definición de la tabla, en PostgreSQL, es la siguiente:

```
CREATE TABLE registro_de_salida_de_soportes_informaticos
(
  numero_salida serial NOT NULL,
  codigo_soporte char(5),
  responsable_seguridad varchar(50),
  receptor varchar(50),
  emisor varchar(50),
  autorizacion_escrita bool,
  fecha date,
  hora time,
```

```
  numero_soportes numeric,
  codigo_envio char(5),

CONSTRAINT registro_de_salida_de_soportes_informaticos_pkey PRIMARY KEY (numero_salida),
CONSTRAINT  registro_de_salida_de_soportes_informaticos_codigo_envio_fkey  FOREIGN  KEY  (codigo_envio)
REFERENCES forma_de_envio (codigo_envio) ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT registro_de_salida_de_soportes_informaticos_codigo_soporte_fkey FOREIGN KEY (codigo_soporte)
REFERENCES tipo_de_soporte (codigo_soporte) ON UPDATE NO ACTION ON DELETE NO ACTION
)

WITHOUT OIDS;
ALTER TABLE registro_de_salida_de_soportes_informaticos OWNER TO postgres;

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.numero_salida IS
 'Valor añadido. Enumera las salidas de soportes que se producen.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.codigo_soporte IS
'Valor añadido. Recogerá el código del tipo de soporte informático enviado.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.responsable_seguridad IS
 'Valor añadido. Persona que por ley se responsabiliza de la seguridad de los ficheros.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.receptor IS
 'Real Decreto 994/1999. Artículo 20. Identifica al receptor de la información.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.emisor IS
'Real Decreto 994/1999. Artículo 20. Identifica al emisor de la información.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.autorizacion_escrita IS
 'Real Decreto 994/1999. Artículo 20. Recoge sie el emisor posee la autorización.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.fecha IS
 'Real Decreto 994/1999. Articulo 20. Recibirá la fecha en el momento de producirse la salida del soporte.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.hora IS
'Real Decreto 994/1999. Articulo 20. Recibirá la hora en el momento de producirse la salida del soporte.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.numero_soportes IS
 'Real Decreto 994/1999. Artículo 20. Recoge el número de soportes enviados.';

COMMENT ON COLUMN registro_de_salida_de_soportes_informaticos.codigo_envio IS
 'Valor añadido. Recogerá el código de envío que se relaciona con las distintas formas de envio de los ficheros.';
```

En la figura 4.5 podemos observar las relaciones de la tabla descrita.

Figura 4.5. Relaciones de la tabla “registro_de_salida_de_soportes_informaticos”.

4.1.1.4. Tabla: “registro_de_accesos”.-

En el artículo 24 del Real Decreto 994/1999 se recoge que se deberá establecer un sistema de registro de acceso donde se guardaran como mínimo:

a) Identificación del usuario.
b) Fecha y hora.
c) Fichero accedido.
d) Tipo de acceso.
e) Acceso autorizado o denegado. (En el caso que haya sido autorizado será preciso guardar la información que permita identificar el registro accedido).

Con el fin de dar solución a estas exigencias se ha diseñado la tabla “registro_de_accesos” cuya definición se muestra a continuación:

```
CREATE TABLE registro_de_accesos
(
  numero_acceso serial NOT NULL,
  id_usuario varchar(50),
  id_host varchar(50),
  credenciales varchar(50),
  responsable_seguridad varchar(50),
  fecha date,
  hora time,
  autorizado bool,
  fichero_accedido varchar(50),
  propiedades_fichero_accedido varchar(50),
  codigo_denegacion char(5),
  id_recurso varchar(50),
  url varchar(500),
  copia_respaldo bool,
  tipo_acceso varchar(50),
  numero_acceso_fichero numeric,
  codigo_acceso char(5),
  despersonalizacion bool,

  CONSTRAINT registro_de_accesos_pkey PRIMARY KEY (numero_acceso),
  CONSTRAINT registro_de_accesos_codigo_acceso_fkey FOREIGN KEY (codigo_acceso)
  REFERENCES situacion_acceso (codigo_acceso) ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT registro_de_accesos_codigo_denegacion_fkey FOREIGN KEY (codigo_denegacion)
  REFERENCES motivo_de_denegacion (codigo_denegacion) ON UPDATE NO ACTION ON DELETE NO ACTION
)

WITHOUT OIDS;
ALTER TABLE registro_de_accesos OWNER TO postgres;

COMMENT ON COLUMN registro_de_accesos.numero_acceso IS
 'Valor añadido. Enumera los accesos que se producen.';

COMMENT ON COLUMN registro_de_accesos.id_usuario IS
 'Real Decreto 994/1999. Articulo 24. Recogerá la identificación del usuario.';

COMMENT ON COLUMN registro_de_accesos.id_host IS
 'Valor añadido. Recogerá el Host del cual se accede al fichero.';
```

```
COMMENT ON COLUMN registro_de_accesos.credenciales IS
 'Valor añadido. Recogerá los permisos que posee el usuario en el momento de acceso al fichero.';

COMMENT ON COLUMN registro_de_accesos.responsable_seguridad IS
 'Valor añadido. Persona que por ley se responsabiliza de la seguridad de los ficheros.';

COMMENT ON COLUMN registro_de_accesos.fecha IS
 'Real Decreto 994/1999. Articulo 24. Recibirá la fecha en el momento de producirse el acceso.';

COMMENT ON COLUMN registro_de_accesos.hora IS
 'Real Decreto 994/1999. Articulo 24. Recibirá la hora en el momento de producirse el acceso.';

COMMENT ON COLUMN registro_de_accesos.autorizado IS
 'Real Decreto 994/1999. Articulo 24. Recogerá si el acceso al fichero ha sido autorizado.';

COMMENT ON COLUMN registro_de_accesos.fichero_accedido IS
 'Real Decreto 994/1999. Articulo 24. Recogerá el nombre del fichero accedido.';

COMMENT ON COLUMN registro_de_accesos.propiedades_fichero_accedido IS
'Real Decreto 994/1999. Articulo 24. Recogerá información sobre el fichero accedido.';

COMMENT ON COLUMN registro_de_accesos.codigo_denegacion IS
'Valor añadido. Recogerá el código del tipo de denegación de acceso al fichero.';

COMMENT ON COLUMN registro_de_accesos.id_recurso IS
 'Valor añadido motivado por el Real Decreto 994/1999. Articulo 24. Recogerá el nombre del  recurso al que se
accede.';

COMMENT ON COLUMN registro_de_accesos.url IS
 'Valor añadido motivado por el Real Decreto 994/1999. Articulo 24. Recogerá el URL del  fichero al que se accede.';

COMMENT ON COLUMN registro_de_accesos.copia_respaldo IS
'Valor añadido. Indica si en el momento de acceso del fichero existe una copia de respaldo del mismo.';

COMMENT ON COLUMN registro_de_accesos.tipo_acceso IS
 'Real Decreto 994/1999. Articulo 24. Recogerá el tipo de acceso al fichero (lectura, escritura,...).';

COMMENT ON COLUMN registro_de_accesos.numero_acceso_fichero IS
 'Valor añadido. Enumera los accesos al fichero.';

COMMENT ON COLUMN registro_de_accesos.codigo_acceso IS
 'Valor añadido. Recogerá el código de la situación en que se accede al recurso.';

COMMENT ON COLUMN registro_de_accesos.despersonalizacion IS
'Valor añadido. Señala si se despersonalizó el fichero accedido.';
```

Como se observa en los comentarios asociados a los campos de la tabla definida existen muchos marcados como “valor añadido”. El registro de acceso, como su nombre indica, va a ser el registro más importante en que se basa nuestro proyecto. Se han añadido campos que facilitan la identificación del usuario y la situación en la que accede al Sistema de Información.

En la ley 41/2002 se autoriza la comunicación de los datos de la historia clínica, aun sin el consentimiento del interesado, en los supuestos siguientes: cesión a las autoridades judiciales, situaciones de urgencia, estudios epidemiológicos, actuaciones de salud pública, investigación y docencia (previa despersonalización de los datos), así como ejercicio de funciones de inspección, evaluación , acreditación y planificación.

En el Real Decreto 994/1999 se recoge continuamente el tratamiento de copia de respaldo de ficheros como medida de seguridad. El campo “copia_respaldo” recoge si existe o no copia de respaldo de un archivo accedido . En el caso que exista copia de respaldo, en la tabla “registro_de_respaldo” se registrará la fecha de ésta, con el fin de poder posteriormente comprobar las modificaciones realizadas al fichero desde que se realizó la copia.

La tabla “registro_de_respaldo” es definida, en PostgreSQL, con el siguiente código:

```
CREATE TABLE registro_de_respaldo
(
  numero_acceso serial NOT NULL,
  fecha_copia_respaldo date,

CONSTRAINT registro_de_respaldo_numero_acceso_fkey FOREIGN KEY (numero_acceso)
REFERENCES registro_de_accesos (numero_acceso) ON UPDATE NO ACTION ON DELETE NO ACTION
)

WITHOUT OIDS;
ALTER TABLE registro_de_respaldo OWNER TO postgres;

COMMENT ON COLUMN registro_de_respaldo.numero_acceso IS
 'Valor añadido. Enumera los accesos que se producen.';

COMMENT ON COLUMN registro_de_respaldo.fecha_copia_respaldo IS
 'Valor añadido. Recogerá la fecha de la copia de respaldo.';
```

La tabla “registro_de_acceso” presenta relaciones con las tablas auxiliares “situacion_acceso” y “motivo_de_denegacion”.

En la tabla “situacion_acceso” se recoge en qué circunstancias el usuario accede a los datos almacenados en los recursos. La definición de la tabla es la siguiente:

```
CREATE TABLE situacion_acceso
(
  codigo_acceso char(5) NOT NULL,
  descripcion varchar(200) NOT NULL,

  CONSTRAINT situacion_acceso_pkey PRIMARY KEY (codigo_acceso)
)

WITHOUT OIDS;
ALTER TABLE situacion_acceso OWNER TO postgres;


COMMENT ON COLUMN situacion_acceso.codigo_acceso IS
 'Valor añadido. Recogerá el código que será relacionado con una situación de acceso al recurso.';

COMMENT ON COLUMN situacion_acceso.descripcion IS
 'Ley 41/2002. Describe en que contexto se accedio al recurso (judicial, profesional, investigación, docencia, el paciente, ...)';
```

La tabla auxiliar “motivos_de_denegación” recoge los motivos por los que no se permite el acceso a un fichero. Su descripción, en PostgreSQL, responde al siguiente código:

```
CREATE TABLE motivo_de_denegacion
(
  codigo_denegacion char(5) NOT NULL,
  descripcion varchar(200) NOT NULL,

  CONSTRAINT motivo_de_denegacion_pkey PRIMARY KEY (codigo_denegacion)
)

WITHOUT OIDS;
ALTER TABLE motivo_de_denegacion OWNER TO postgres;

COMMENT ON COLUMN motivo_de_denegacion.codigo_denegacion IS
 'Valor añadido. Recogerá el código del tipo de denegación de acceso al fichero.';

COMMENT ON COLUMN motivo_de_denegacion.descripcion IS
 'Valor añadido. El motivo por el cual no se ha autorizado el acceso.';
```

Con el fin de mostrar mejor el diseño de la base de datos, se muestra a continuación la figura 4.6 con las relaciones de la tabla “registro_de_accesos”.

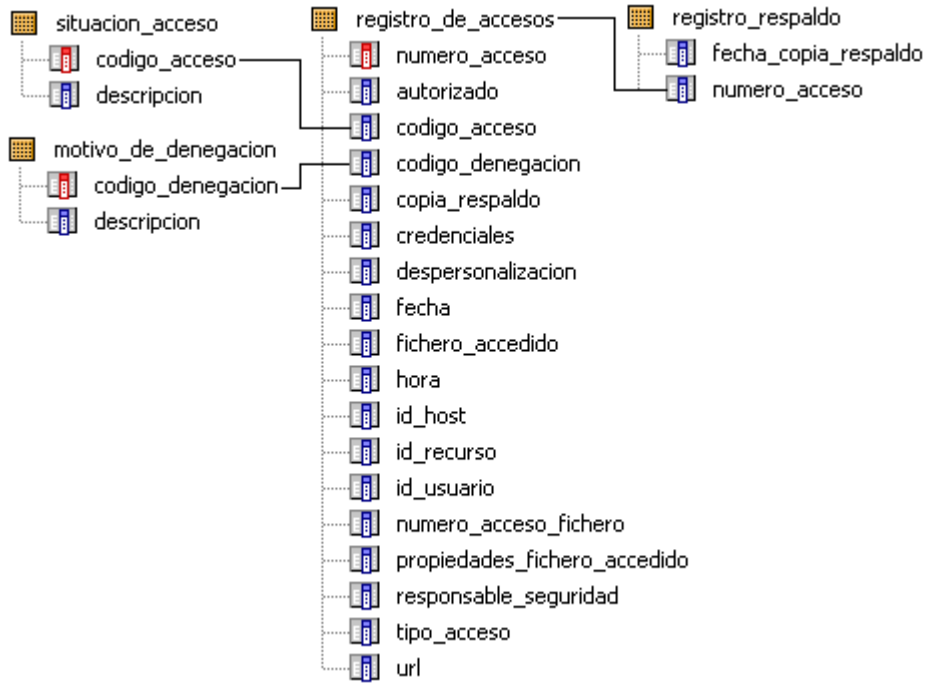

Figura 4.6. Relaciones de la tabla “registro_de_accesos”.

4.1.1.5. Tablas dedicadas a los usuarios del servidor.-

Como indicamos al comienzo de este apartado la base de datos “Registro de Acceso” presenta también tres tablas dedicadas al control de los usuarios del servidor. Estas tres tablas almacenan los usuarios que pueden beneficiarse de los servicios del servidor, los permisos que le son concedidos y los recursos que pueden ser accedidos.

La creación de tabla “personal_autorizado” es motivada por el artículo 12 del Real Decreto 994/1999. Es descrita en PostgreSQL del siguiente modo:

```
CREATE TABLE personal_autorizado
(
  login varchar(20) NOT NULL,
  "password" char(9) NOT NULL,
  id_usuario varchar(50) NOT NULL,
  codigo_credenciales char(6),

  CONSTRAINT personal_autorizado_pkey PRIMARY KEY (login),
  CONSTRAINT personal_autorizado_codigo_credenciales_fkey FOREIGN KEY (codigo_credenciales)
  REFERENCES credenciales (codigo_credenciales) ON UPDATE NO ACTION ON DELETE NO ACTION
)

WITHOUT OIDS;
ALTER TABLE personal_autorizado OWNER TO postgres;

COMMENT ON COLUMN personal_autorizado.login IS
'Valor añadido derivado del Real Decreto 994/1999. Articulo 12. Alias del usuario autorizado en el Servidor de Registro de Acceso.';

COMMENT ON COLUMN personal_autorizado."password" IS
 'Valor añadido derivado del Real Decreto 994/1999. Articulo 12. Contraseña del usuario autorizado en el Servidor de Registro de Acceso.';

COMMENT ON COLUMN personal_autorizado.id_usuario IS
 'Valor añadido derivado del Real Decreto 994/1999. Articulo 12. Identificación del usuario autorizado en el Servidor de Registro de Acceso';


COMMENT ON COLUMN personal_autorizado.codigo_credenciales IS
 'Valor añadido. Recogerá un código que representa los permisos del personal autorizado en el Servidor de Registro de Acceso';
```

La tabla credenciales actúa como tabla complementaria de la anterior y su definición es la siguiente:

```
CREATE TABLE credenciales
(
  codigo_credenciales char(6) NOT NULL,
  descripcion varchar(200) NOT NULL,
  recursos varchar(2000) NOT NULL,

  CONSTRAINT credenciales_pkey PRIMARY KEY (codigo_credenciales)
)

WITHOUT OIDS;
```

```
ALTER TABLE credenciales OWNER TO postgres;


COMMENT ON COLUMN credenciales.codigo_credenciales IS
 'Valor añadido. Recogerá el código que representa los permisos del personal autorizado en el Servidor de Registro de Acceso';

COMMENT ON COLUMN credenciales.descripcion IS
 'Real Decreto 994/1999. Articulo 12. Recogerá la descripción de los permisos en el Servidor de Registro de Acceso.';

COMMENT ON COLUMN credenciales.recursos IS
 'Real Decreto 994/1999. Articulo 12. Recogerá los valores numéricos que identifican los recursos en el Servidor de Registro de Acceso.';
```

En el campo “recursos” se almacenarán valores numéricos que identificaran las acciones que puede realizar el usuario que posea la credencial definida. Estas acciones son descritas con ayuda de la tabla “recursos”.

```
CREATE TABLE recursos
(
  recurso numeric NOT NULL,
  descripcion varchar(200) NOT NULL,

  CONSTRAINT recursos_pkey PRIMARY KEY (recurso)
)

WITHOUT OIDS;
ALTER TABLE recursos OWNER TO postgres;

COMMENT ON COLUMN recursos.recurso IS
 'Valor añadido. Recogerá el número con el que se representa el recurso del Servidor de Registro de Acceso';

COMMENT ON COLUMN recursos.descripcion IS
 'Valor añadido. Descripción del recurso';
```

Las relaciones entre las tablas dedicadas a los usuarios del servidor se presentan en la figura 4.7.

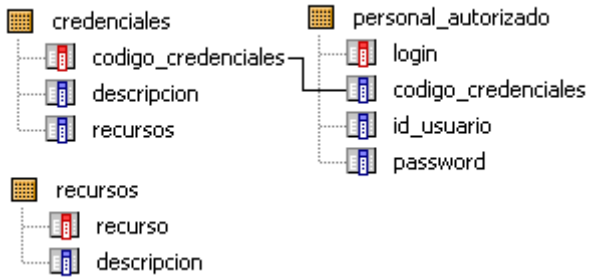

Figura 4.7. Relaciones de las tablas dedicadas a los usuarios del servidor.

4.2. Servidor de Registro de Acceso.-

Una vez que el usuario sea identificado y autentificado, haya obtenido sus credenciales, se hayan identificado los recursos y se haya comprobado que el usuario tiene acceso a alguno de ellos, deberán ser registradas las acciones que realice, transparentemente a él, en la base de datos “Registro de Acceso”. Para ello los agentes implicados en el acceso establecerán comunicación con el Servidor de Registro de Acceso proporcionándole la información necesaria.

El nuevo esquema del Sistema de Información Sanitario presentado en la figura 2.1 introduciendo el Servidor de Registro de Acceso y la base de datos “Registro de Acceso” es el siguiente:

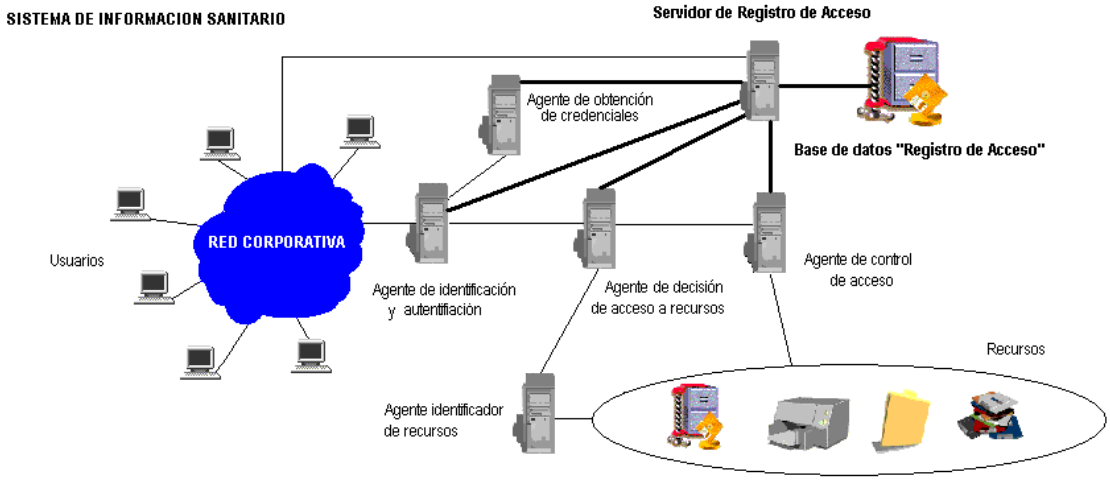

Figura 4.8. Arquitectura de seguridad en el acceso en un Sistema de Información Sanitario con el Servidor de Registro de Acceso y la base de datos “Registro de Acceso”.

Como se observa en la figura, la base de datos sólo podrá ser accedida desde el servidor, quien controlará la entrada y la salida de información.

El Servidor de Registro de Acceso está diseñado utilizando clases Java que actúan bajo el servidor de aplicaciones Tomcat. Estas clases se distribuyen en dos “packages” o paquetes: ardb (Access Register DateBase) y ars (Access Register Service).

4.2.1. Definición de los paquetes.-

4.2.1.1. Paquete “ardb”.-

En este paquete se implementan las funciones necesarias para el diseño y la comunicación con la base de datos “Registro de Acceso”. Una vez generada una base de datos vacía, en PostgreSQL Database Server 8.0, con el nombre “Registro de Acceso”, será el Servidor de Registro de Acceso quien diseñará su estructura con las sentencias SQL definidas en este paquete.

El paquete “ardb” presenta las siguientes clases:

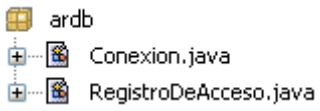

Figura 4.9. Clases definidas en el paquete “ardb”.

Se muestra a continuación una breve descripción, extraída del Javadoc que acompaña a la documentación, de las clases que forman el paquete.

Class Summary	
Conexion	Clase utilizada por el Servidor de Registro de Acceso para establecer y cerrar la conexión con la base de datos "Registro de Acceso" ubicada en PostgreSQL Database Server 8.0.
RegistroDeAcceso	Crea, comprueba, actualiza y consulta la base de datos "Registro de Acceso" ubicada en PostgreSQL Database Server 8.0 Las tablas y campos de la base de datos "Registro de Acceso" han sido diseñadas con el fin de proporcionar una solución práctica a las exigencias, en cuanto a materia de seguridad, que deben reunir los ficheros automátizados que contengan datos de carácter personal relativos a la salud.

4.2.1.2. Paquete “ars”.-

Este paquete es el núcleo del Servidor de Registro de Acceso. Presenta métodos que reciben datos de otros agentes, funciones que almacenan la información en la base de datos, procedimientos para consultar y generar informes y clases para modificar los valores de las tablas auxiliares y dedicadas a usuarios del servidor. Las clases que forman el paquete “ars” se muestra a continuación:

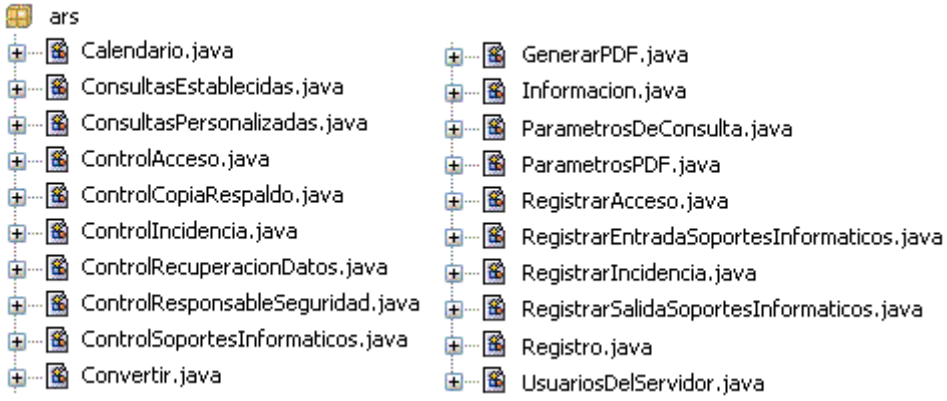

Figura 4.10. Clases definidas en el paquete “ars”.

Se muestra a continuación una breve descripción de las clases que forman el paquete.

Class Summary	
Calendario	Clase utilizada por el Servidor de Registro de Acceso para obtener la fecha y la hora en el instante que se precise.
ConsultasEstablecidas	Clase utilizada por el servidor de Registro de Acceso para obtener información de la base de datos "Registro de Acceso" con consultas establecidas por el diseñador.
ConsultasPersonalizadas	Clase utilizada por el Servidor de Registro de Acceso para obtener información de la base de datos "Registro de Acceso" con consultas personalizadas por el programador.
ControlAcceso	Clase que almacena información sobre el acceso del usuario al Sistema de Información.
ControlCopiaRespaldo	Clase que almacena información relacionada con la existencia de ficheros con copia de respaldo en los recursos presentes en el Sistema de Información.
ControlIncidencia	Clase que almacena información sobre las incidencias de seguridad que pueden darse en el acceso del usuario al Sistema de Información.
ControlRecuperacionDatos	Clase que almacena información sobre la recuperación de datos que pueda darse en alguno de los recursos controlados en el Sistema de Información.
ControlResponsableSeguridad	Clase que almacena el nombre del Responsable de Seguridad definido en el artículo 16 del Real Decreto 994/1999.
ControlSoportesInformaticos	Clase que almacena información sobre la salida o entrada de soportes informáticos en alguno de los recursos controlados en el Sistema de Información.
Convertir	Clase utilitaria que facilita la conversión de tipos de datos simples.
GenerarPDF	Clase que genera un informe en PDF con la consulta almacenada en los campos de la clase ParametrosPDF.java.
Informacion	Clase que actualiza los campos de las tablas, auxiliares y dedicadas a usuarios del servidor, de la base de datos "Registro de Acceso".
ParametrosDeConsulta	Clase utilizada para almacenar la información, existente en los distintos campos de las tablas de la base de datos, cuando se realiza una consulta.
ParametrosPDF	Clase que almacena la información necesaria para generar el informe PDF.
RegistrarAcceso	Clase que ingresa en tabla "registro_de_accesos" información sobre el acceso de los usuarios a los recursos del Sistema de Información.
RegistrarEntradaSoportesInformaticos	Clase que introduce en la tabla "registro_de_entrada_de_soportes_informaticos" información sobre la entrada de soportes al Sistema de Información.
RegistrarIncidencia	Clase que ingresa en tabla "registro_de_incidencias" información sobre las incidencias de seguridad ocurridas durante el acceso del usuario al Sistema de Información.
RegistrarSalidaSoportesInformaticos	Clase que introduce en la tabla "registro_de_salida_de_soportes_informaticos" información sobre la salida de soportes del Sistema de Información.
Registro	Clase que recoge variables comunes para las clases RegistrarAcceso.java, RegistrarIncidencia.java, RegistrarEntradaSoportesInformáticos.java y RegistrarSalidaSoportesInformáticos que almacenan en la base de datos "Registro de Acceso" los datos perteneciente al control de seguridad aplicado a los usuarios que acceden al Sistema de Información.
UsuariosDelServidor	Clase utilizada por el Servidor de Registro de Acceso para comprobar los datos y permisos de los usuarios que acceden a él.

4.2.2. Descripción de clases.-

Se describen a continuación las clases que forman la implementación del Servidor de Registro de Acceso. Para un mayor estudio, se recomienda al lector la consulta del Javadoc que acompaña la documentación y la consulta del código desarrollado expuesto en el apartado pliego de condiciones.

4.2.2.1. Clase Conexion.java.-

Clase utilizada por el Servidor de Registro de Acceso para establecer y cerrar la conexión con la base de datos "Registro de Acceso" ubicada en PostgreSQL Database Server 8.0. Para que exista comunicación se hacen necesarias las extensiones JDBC.

La clase Conexión.java presenta los siguientes métodos:

Method Summary	
static void	**CerrarConexion**(java.sql.Connection conexion) Cierra la conexión con la base de datos referenciada en el argumento objeto tipo Connection.
static java.sql.Connection	**EstablecerConexion**() Establece la conexión con la base de datos "Registro de Acceso".

4.2.2.2. Clase RegistroDeAcceso.java.-

Crea, comprueba, actualiza y consulta la base de datos "Registro de Acceso" ubicada en PostgreSQL Database Server 8.0 Las tablas y campos de la base de datos "Registro de Acceso" han sido diseñadas con el fin de proporcionar una solución práctica a las exigencias, en cuanto a materia de seguridad, que deben reunir los ficheros automatizados que contengan datos de carácter personal relativos a la salud. Motivado por el Real Decreto 994/1999, la Ley 41/2002, y la Ley Orgánica 15/1999.

La clase RegistroDeAcceso.java define lo siguientes métodos:

Method Summary	
static boolean	**ActualizarBaseDeDatos**(java.sql.Connection conexion, java.lang.String sql) Actualiza la base de datos referenciada en conexion, con la secuencia SQL pasada como argumento.
static boolean	**ComprobarBaseDeDatos**() Compueba si la base de datos "Registro de Acceso" ha sido creada.
static java.sql.ResultSet	**ConsultarBaseDeDatos**(java.sql.Connection conexion, java.lang.String sql) Consulta la base de datos referenciada en conexion con la sentencia SQL pasada como argumento, devolviendo los resultados en el objeto ResulSet rs.
static void	**CrearTablasBaseDeDatos**() Diseña la base de datos "Registro de Acceso".

Los métodos del paquete han sido definidos como “static” con el fin de simplificar el desarrollo del cliente. De este modo, no se tendrá que crear nuevos objetos cada vez que se desee utilizar algún método definido en las clases.

4.2.2.3. Clase Calendario.java.-

Clase utilizada por el Servidor de Registro de Acceso para obtener la fecha y la hora en el instante que se precise. Con funciones de las clases java.util.Calendar y java.util.GregorianCalendar creamos el formato de fecha y hora establecido por PostgreSQL Database Server 8.0 para los campos tipo date y time.

Method Summary	
java.lang.String	**ObtenerFecha**() Método que devuelve la fecha actual en formato mm/dd/aaaa.
java.lang.String	**ObtenerHora**() Método que devuelve la hora actual en formato hh:mm:ss.

4.2.2.4. Clase ConsultasEstablecidas.java.-

Clase utilizada por el servidor de Registro de Acceso para obtener información de la base de datos "Registro de Acceso" por medio de consultas establecidas por el diseñador. Toda consulta establecida utilizada para informar a los usuarios del servidor, así como aquellas diseñadas para servicios de gestión y mantenimiento de la base de datos, se encuentran en los métodos de esta clase. Las consultas a la base de datos "Registro de Acceso" están ordenadas numéricamente, de modo que la forma de elegir una u otra dependerá del valor del parámetro tipo “int” Elección, que se pasa como argumento. Una vez realizada la consulta, se prepara el formato del resultado para facilitar su gestión por el cliente.

Method Summary	
static java.util.Vector	**CrearVectorConsulta**(java.sql.ResultSet rs) Método que convierte el resultado tipo ResultSet de las consultas a un Vector cuyos elementos son parámetros de la clase ParametrosDeConsulta.java
static java.util.Vector	**Dinamicas**(java.lang.String Fecha_Inferior, java.lang.String Fecha_Superior, int Eleccion) Clase que obtiene información de los registros principales de la base de datos para unas fechas concretas.
static java.util.Vector	**Estaticas**(int Eleccion) Método que ejecuta una u otra consulta en función del valor de Eleccion que se pasa como argumento.
static int	**NumeroDeFichero**(java.lang.String Nombre_Fichero) Método que consulta el campo "fichero_accedido" de la tabla "registro_de_accesos" para obtener el número de veces que se ha consultado un mismo fichero.
static int	**NumeroDeRegistro**(int Opcion) Método que devuelve el número de registro próximo para las tablas principales de la base de datos.
static java.util.Vector	**RetornarNombreCampos**(java.lang.String Sql) Método que devuelve un Vector con los nombres de los campos consultados por la secuencia SQL pasada como parámetro.
static java.lang.String	**TextoCodigoIncidencia**() Método que devuelve el campo "descripcion" de la tabla "incidencias" relacionado con el "codigo_incidencia" pasado internamente al método por la clase ControlIncidencia.java.
static java.lang.String	**TextoConsulta**(int Eleccion) Método que devuelve el texto "Consulta número: ?"
static java.sql.ResultSet	**UsuariosDelServidor**(int Eleccion) Método que consulta las tablas "personal_autorizado" y "credenciales" para comprobación de usuarios y permisos en el Servidor de Registro de Acceso.

4.2.2.5. Clase ConsultasPersonalizadas.java.-

Clase utilizada por el Servidor de Registro de Acceso para obtener información de la base de datos "Registro de Acceso" con consultas personalizadas por el programador. En esta clase se recogen métodos que permiten personalizar las consultas a la base de datos "Registro de Acceso".

La forma de crear una nueva consulta se basa en la formación dinámica de sentencias SQL. El formato SQL de consulta permitido responde al código SQL: "SELECT (tablas.campos) FROM (tablas) WHERE (condiciones SQL) ORDER BY (campos);".

En la clase de definen los siguientes campos:

Field Summary	
private static java.lang.String	**Campo** Objeto tipo String con el nombre del campo de la tabla a utilizar.
static java.util.Vector	**CamposSeleccionados** Objeto tipo Vector con los campos seleccionados para realizar la consulta.
private static java.lang.String	**Condicion** Objeto tipo String con la condición SQL que debe cumplir la consulta.
static java.util.Vector	**CondicionesSeleccionadas** Objeto tipo Vector con las condiciones impuestas por el usuario a la consulta.
private static java.sql.Connection	**conexion** Objeto tipo Connection con la conexión a la base de datos "Registro de Acceso".
static java.util.Vector	**ConsultasAlmacenadas** Objeto tipo Vector que almacena las sentencias SQL pertenecientes a las nuevas consultas.
private static java.sql.DatabaseMetaData	**dmd** Objeto tipo DatabaseMetaData que obtiene las características de la base de datos.
(package private) static int	**EleccionConsulta** Campo que indica el número de consulta a registrar.
static java.util.Vector	**NombreConsultasAlmacenadas** Objeto tipo Vector que almacena el nombre de las nuevas consultas.
static java.util.Vector	**NumeroEleccionConsultasAlmacenadas** Objeto tipo Vector que almacena el número correspondiente a la consulta almacenada.

private static java.lang.String	**Orden** Objeto tipo String con la descripción del orden que debe cumplir la consulta.
static java.util.Vector	**OrdenSeleccionadas** Objeto tipo Vector con los campos que marcan el orden.
private static java.sql.ResultSet	**rs** Objeto tipo ResultSet con el resultado de la consulta a la base de datos.
private static java.sql.ResultSetMetaData	**rsmd** Objeto tipo ResultSetMetaData que obtiene las características del "rs".
private static java.lang.String	**Sql** Objeto tipo String con la sentencia SQL a ejecutar.
private static java.lang.String	**Tabla** Objeto tipo String con el nombre de la tabla a utilizar.
static java.util.Vector	**TablasSeleccionadas** Objeto tipo Vector con las tablas seleccionadas para realizar la consulta.

ConsultasPersonalizadas.java presenta los siguientes métodos:

Method Summary	
static void	**BorrarCampos**() Método que borra los Vectores CamposSeleccionados y TablasSeleccionadas.
static void	**BorrarCamposCondicionesOrden**() Método que borra todos los Vectores relativos a la creación de una nueva consulta.
static void	**BorrarCondiciones**() Método que borra el Vector CondicionesSeleccionadas.
static void	**BorrarConsultasAlmacenadas**() Borra los Vectores NombreConsultasAlmacenadas, ConsultasAlmacenadas y NumeroEleccionConsultasAlmacenadas.
static void	**BorrarLaConsultaX**(java.lang.String SQL) Borra la consulta cuya sentencia SQL es pasada como argumento.
static void	**BorrarOrden**() Método que borra el Vector OrdenSeleccionadas.
static void	**BorrarUltimaCondicion**() Método que borra la última condición introducida.
static void	**BorrarUltimoCampo**() Método que borra el último campo introducido.
static void	**BorrarUltimoOrden**() Método que borra el último campo selecionado para ordenar la consulta.
static void	**ClearCampo**() Borra las variables Campo y Tabla.
static void	**ClearCondicion**() Borra la variable Condicion.

static void	**ClearOrden**() Borra la variable Orden.
static java.lang.String	**CrearConsultaSQL**(java.lang.String NombreConsulta) Método que crea la sentencia SQL de consulta con información introducida en CamposSeleccionados, TablasSeleccionadas, CondicionesSeleccionadas y OrdenSeleccionadas.
static java.lang.String	**DevolverConsultaAlmacenada**(java.lang.String NombreConsulta) Método que devuelve la sentencia SQL que corresponde al nombre de la consulta pasada como argumento.
static java.lang.String	**DevolverNombreConsultaAlmacenada**(java.lang.String SQL) Método que devuelve el nombre de la consulta cuya sentencia SQL es pasada como argumento.
static int	**DevolverNumeroEleccion**(java.lang.String SQL) Método que devuelve el número asociado a la consulta cuya sentencia SQL es pasada como argumento.
static java.lang.String	**getCampo**() Devuelve Campo
static java.lang.String	**getCondicion**() Devuelve Condición.
static java.lang.String	**getOrden**() Devuelve Orden.
static java.lang.String	**getTabla**() Devuelve Tabla.
static void	**InsertarCampo**() Método que inserta en los Vectores CamposSeleccionados y TablasSeleccionadas el campo seleccionado.

static void	**InsertarCondicion**() Método que inserta en el Vector CondicionesSeleccionadas la condicion SQL configurada por el usuario.
static void	**InsertarConsultaAlmacenada**(java.lang.String NombreConsulta, java.lang.String Sql) Inserta en los vectores NombreConsultasAlmacenadas, ConsultasAlmacenadas y NumeroEleccionConsultasAlmacenadas el nombre, la sentencia SQL y el número asociado a la nueva consulta.
static void	**InsertarOrden**() Método que inserta en el vector OrdenSeleccionadas los campos por los que se quiere ordenar la consulta.
static java.util.Vector	**MostrarConsultaSQL**(java.lang.String SQL) Método que recibe como argumento una sentencia SQL de consulta, guarda los datos necesarios para el informe PDF y retorna el resultado asociado, en un Vector cuyos elementos son parámetros de la clase ParametrosDeConsulta.java.
static java.util.Vector	**MostrarNombresCampos**(int Eleccion) Método que devuelve un Vector con los nombres de los campos de la tabla seleccionada de la base de datos "Registro de Acceso".
static java.lang.String	**MostrarNombreTabla**(int Eleccion) Método que devuelve un objeto tipo String con el nombre de la tabla de la base de datos "Registro de Acceso" seleccionada con Eleccion.
static void	**setCampo**(java.lang.String AuxCampo) Método que guarda el nombre del campo seleccionado.
static void	**setCondicion**(java.lang.String AuxCondicion) Método que almacena la condición SQL seleccionada.

static void	**setOrden**(java.lang.String AuxOrden) Método que guarda el nombre del campo por el cual se pretende ordenar la consulta.
static void	**setTabla**(java.lang.String AuxTabla) Método que guarda el nombre de la tabla seleccionada.

4.2.2.6.Clase ControlAcceso.java.-

Clase que almacena información sobre el acceso del usuario al Sistema de Información. La clase ControlAcceso.java recoge los datos a insertar en la tabla “registro_de_accesos” . Los campos que reciben la información son los siguientes:

Field Summary	
(package private) static boolean	**Autorizado** Campo que recoge si el acceso del usuario al recurso fue autorizado.
(package private) static java.lang.String	**CodigoAcceso** El código que recoge en que situación accedio el usuario al fichero.
(package private) static java.lang.String	**CodigoDenegado** Campo que recoge el código por el cual no se pudo acceder al fichero.
(package private) static boolean	**CopiaRespaldo** Campo que señala si existe una copia de respaldo para el fichero accedido.
(package private) static java.lang.String	**Credenciales** Credenciales del usuario al acceder al sistema.
(package private) static boolean	**Despersonalizacion** Campo que recoge si se despersonalizó el fichero antes del acceso del usuario.
(package private) static java.lang.String	**FicheroAccedido** Fichero accedido por el usuario.
(package private) static java.lang.String	**IdHost** Identificación del ordenador utilizado por el usuario para acceder al sistema.
(package private) static java.lang.String	**IdRecurso** Identificación del recurso al que accede el usuario.
(package private) static java.lang.String	**IdUsuario** Identificación del usuario.
(package private) static java.lang.String	**PropiedadesFicheroAccedido** Propiedades del fichero accesiddo por le usuario.
(package private) static java.lang.String	**TipoAcceso** Recoge el tipo de acceso del usuario al fichero.
(package private) static java.lang.String	**URL** URL del fichero accedido por el usuario.

Estos campos serán tratados en los siguientes métodos:

Method Summary	
static void	**Clear**() Borra todos los parámetros del control de acceso.
static boolean	**getAutorizado**() Devuelve Autorizado
static java.lang.String	**getCodigoAcceso**() Devuelve CodigoAcceso.
static java.lang.String	**getCodigoDenegado**() Devuelve CodigoDenegado.
static boolean	**getCopiaRespaldo**() Devuelve CopiaRespaldo.
static java.lang.String	**getCredenciales**() Devuelve Credenciales.
static boolean	**getDespersonalizacion**() Devuelve Despersonalizacion.

static java.lang.String	**getFicheroAccedido**() Devuelve FicheroAccedido.
static java.lang.String	**getIdHost**() Devuelve IdHost.
static java.lang.String	**getIdRecurso**() Devuelve IdRecurso,
static java.lang.String	**getIdUsuario**() Devuelve IdUsuario.
static java.lang.String	**getPropiedadesFicheroAccedido**() Devuelve PropiedadesFicheroAccedido
static java.lang.String	**getTipoAcceso**() Devuelve TipoAcceso.
static java.lang.String	**getURL**() Devuelve URL
static void	**setAutorizado**(boolean AuxAutorizado) Almacena Autorizado.
static void	**setCodigoAcceso**(java.lang.String AuxCodigoAcceso) Almacena CodigoAcceso.
static void	**setCodigoDenegado**(java.lang.String AuxCodigoDenegado) Almacena CodigoDenegado.
static void	**setCopiaRespaldo**(boolean AuxCopiaRespaldo) Almacena CopiaRespaldo.

static void	**setCredenciales**(java.lang.String AuxCredenciales) Almacena Credenciales.
static void	**setDespersonalizacion**(boolean AuxDespersonalizacion) Almacena Despersonalizacion.
static void	**setFicheroAccedido**(java.lang.String AuxFicheroAccedido) Almacena FicheroAccedido.
static void	**setIdHost**(java.lang.String AuxIdHost) Almacena IdHost.
static void	**setIdRecurso**(java.lang.String AuxIdRecurso) Almacena IdRecurso.
static void	**setIdUsuario**(java.lang.String AuxIdUsuario) Almacena IdUsuario.
static void	**setPropiedadesFicheroAccedido**(java.lang.String AuxPropiedadesFicheroAccedido) Almacena PropiedadesFicheroAccedido
static void	**setTipoAcceso**(java.lang.String AuxTipoAcceso) Almacena TipoAcceso.
static void	**setURL**(java.lang.String AuxURL) Almacena URL

4.2.2.7. Clase ControlCopiaRespaldo.java.-

Clase que almacena información relacionada con la existencia de ficheros con copia de respaldo en los recursos presentes en el Sistema de Información. Define el campo:

Field Summary	
(package private) static java.lang.String	**FechaCopiaRespaldo** Fecha de la copia de respaldo del fichero accedido en formato mm/dd/aaaa.

Presenta los siguientes métodos de acceso:

Method Summary	
static void	**Clear**() Borra todos los parámetros del control de copia de respaldo.
static java.lang.String	**getFechaCopiaRespaldo**() Devuelve FechaCopiaRespaldo.
static void	**setFechaCopiaRespaldo**(java.lang.String AuxFechaCopiaRespaldo) Almacena FechaCopiaRespaldo.

4.2.2.8. Clase ControlIncidencia.java.-

Clase que almacena información sobre las incidencias de seguridad que pueden darse en el acceso del usuario al Sistema de Información. Los campos descritos en esta clase son:

Field Summary	
(package private) static java.lang.String	**CodigoComunicacion** Código de la comunicacion de la incidencia.
(package private) static java.lang.String	**CodigoIncidencia** Código de la incidencia de seguridad detectada en el acceso.
(package private) static java.lang.String	**Notificante** Persona o sistema que notifica la incidencia.

La clase presenta los siguientes métodos:

Method Summary	
static void	**Clear**() Borra todos los parámetros del control de incidencia.
static java.lang.String	**getCodigoComunicacion**() Devuelve CodigoComunicacion.
static java.lang.String	**getCodigoIncidencia**() Devuelve CodigoIncidencia.
static java.lang.String	**getNotificante**() Devuelve Notificante.
static void	**setCodigoComunicacion**(java.lang.String AuxCodigoComunicacion) Almacena CodigoComunicacion.
static void	**setCodigoIncidencia**(java.lang.String AuxCodigoIncidencia) Almacena CodigoIncidencia.
static void	**setNotificante**(java.lang.String AuxNotificante) Almacena Notificante.

4.2.2.9. Clase ControlRecuperacionDatos.java.-

Clase que almacena información sobre la recuperación de datos que pueda darse en alguno de los recursos controlados en el Sistema de Información. Los campos que guardan la información son los siguientes:

Field Summary	
(package private) static java.lang.String	**Datos** Datos que han sido recuperados en el proceso.
(package private) static java.lang.String	**DatosManuales** Datos que han sido recuperados manualmente en el proceso.
(package private) static boolean	**RecuperacionAutorizada** Campo que recoge si la recuperación de datos ha sido autorizada.

La clase presenta los siguientes métodos:

Method Summary	
static void	**Clear**() Borra todos los parámetros del control de recuperación de datos.
static java.lang.String	**getDatos**() Devuelve Datos.
static java.lang.String	**getDatosManuales**() Devuelve DatosManuales.
static boolean	**getRecuperacionAutorizada**() Devuelve RecuperacionAutorizada.
static void	**setDatos**(java.lang.String AuxDatos) Almacena Datos.
static void	**setDatosManuales**(java.lang.String AuxDatosManuales) Almacena DatosManuales.
static void	**setRecuperacionAutorizada**(boolean AuxRecuperacionAutorizada) Almacena RecuperacionAutorizada.

4.2.2.10. Clase ControlResponsableSeguridad.java.-

Clase que almacena el nombre del Responsable de Seguridad; definido en el artículo 16 del Real Decreto 994/1999. El campo que recoge la información es:

Field Summary	
(package private) static java.lang.String	**ResponsableSeguridad** Nombre del Responsable de Seguridad.

Los métodos de acceso son:

Method Summary	
static void	**Clear**() Borra todos los parámetros del control del Responsable de Seguridad.
static java.lang.String	**getResponsableSeguridad**() Devuelve ResponsableSeguridad.
static void	**setResponsableSeguridad**(java.lang.String AuxResponsableSeguridad) Almacena ResponsableSeguridad.

4.2.2.11. Clase ControlSoportesInformaticos.java.-

Clase que almacena información sobre la salida o entrada de soportes informáticos en alguno de los recursos controlados en el Sistema de Información.

Presenta los siguientes campos:

Field Summary	
(package private) static boolean	**AutorizacionEscrita** Variable que recoge si existe una autorización escrita que autorice la entrada o la salida de los soportes seleccionados.
(package private) static java.lang.String	**CodigoEnvio** Código de la forma de envío de la información.
(package private) static java.lang.String	**CodigoSoporte** Código del tipo de soporte informático
(package private) static java.lang.String	**Emisor** Emisor de la información.
(package private) static int	**NumeroSoportes** Número de soportes que entran o salen del sistema.
(package private) static java.lang.String	**Receptor** Receptor de la información.

Junto a los siguientes métodos de acceso:

Method Summary	
static void	**Clear**() Borra todos los parámetros del control de sopostes informáticos.
static boolean	**getAutorizacionEscrita**() Devuelve AutorizacionEscrita.
static java.lang.String	**getCodigoEnvio**() Devuelve CodigoEnvio.
static java.lang.String	**getCodigoSoporte**() Devuelve CodigoSoporte.
static java.lang.String	**getEmisor**() Devuelve Emisor.
static int	**getNumeroSoportes**() Devuelve el número de soportes.
static java.lang.String	**getReceptor**() Devuelve Receptor.
static void	**setAutorizacionEscrita**(boolean AuxAutorizacionEscrita) Recoge AutorizacionEscrita.
static void	**setCodigoEnvio**(java.lang.String AuxCodigoEnvio) Recoge CodigoEnvio.
static void	**setCodigoSoporte**(java.lang.String AuxCodigoSoporte) Recoge el código del soporte informático.
static void	**setEmisor**(java.lang.String AuxEmisor) Recoge Emisor.
static void	**setNumeroSoportes**(int AuxNumeroSoportes) Recoge CodigoSoporte.
static void	**setReceptor**(java.lang.String AuxReceptor) Recoge Receptor.

4.2.2.12. Clase Convertir.java.-

Clase utilitaria que facilita la conversión de tipos de datos simples. La clase Convertir.java presenta tres métodos para transformar objetos; String que representen valores numéricos a variables tipo int, variables char que representen valores numéricos a variables tipo int y String que representen valores tipo boolean, a variables tipo boolean.

Method Summary	
static int	**Char_Int**(char AuxChar) Método que convierte a variable tipo int, el parámetro tipo Char pasado como argumento.
static boolean	**String_Boolean**(java.lang.String FALSE_TRUE) Método que convierte a variable tipo boolean, el String pasado como argumento.
static int	**String_Int**(java.lang.String AuxString) Método que convierte a variable tipo int, el String pasado como argumento.

4.2.2.13. Clase GenerarPDF.java.-

Clase que genera un informe en PDF con la consulta almacenada en los campos de la clase ParametrosPDF.java. El informe presenta un encabezado referente al proyecto, la fecha y la hora en que se crea, el número de informe creado y las tablas con la consulta seleccionada. El archivo será conocido como Informe.pdf y será guardado temporalmente dentro del servidor hasta que se produzca otro informe.

En la clase encontramos los siguientes métodos:

Method Summary	
static java.lang.String	**CambiarFormato**(java.lang.String AuxParametro) Método que transforma el formato de valores consultados a la base de datos que no presentan un significado muy intuitivo.
static void	**generarPDF**() Método que recoge de la clase ParametrosPDF.java información sobre la consulta selecionada y crea un informe PDF con los valores asociados a la consulta.

4.2.2.14. Clase Informacion.java.-

Clase que actualiza los campos de las tablas, auxiliares y dedicadas a usuarios del servidor, de la base de datos "Registro de Acceso". Presenta métodos para modificar, ampliar o eliminar la información recogida en estas tablas, así como el método que inicializa la información que contienen.

Define los siguientes métodos:

Method Summary	
static java.lang.String	**Actualizacion**(java.lang.String Sql) Método que ejecuta la actualización seleccionada en los métodos de esta clase.
static java.lang.String	**Actualizar**(int Eleccion, java.lang.String Codigo, java.lang.String Descripcion) Método que actualiza los valores de los campos de las tablas: comunicacion, forma_de_envio, motivo_de_denegacion, situaccion_acceso, tipo_de_soporte y recursos.

static java.lang.String	**Actualizar**(int Eleccion, java.lang.String Codigo, java.lang.String Descripcion, java.lang.String Recurso_Efectos) Método que actualiza los valores de los campos de las tablas: credenciales e incidencias.
static java.lang.String	**Actualizar**(java.lang.String Login, java.lang.String Password, java.lang.String IdUsuario) Método que actualiza los valores de los campos de la tabla personal_autorizado.
static java.lang.String	**Borrar**(int Eleccion, java.lang.String Codigo_Login) Método que elimina registros presentes en las tablas, auxiliares y dedicadas a usuarios del servidor, de la base de datos "Registro de Acceso".
static void	**Inicial**() Inserta información inicial en las tablas, auxiliares y dedicadas a usuarios del servidor, necesaria para el funcionamiento originario del servidor.
static java.lang.String	**Insertar**(int Eleccion, java.lang.String Codigo, java.lang.String Descripcion) Método que inserta nueva información en las tablas: comunicacion, forma_de_envio, motivo_de_denegacion, situaccion_acceso, tipo_de_soporte y recursos.
static java.lang.String	**Insertar**(int Eleccion, java.lang.String Codigo, java.lang.String Descripcion, java.lang.String Recurso_Efectos) Método que inserta nueva información en las tablas credenciales e incidencias.
static java.lang.String	**Insertar**(java.lang.String Login, java.lang.String Password, java.lang.String IdUsuario, java.lang.String Credenciales) Método que inserta nueva información en la tabla personal autorizado.
static java.lang.String	**TransformarTresCifras**(java.lang.String AuxCodigo) Método que convierte el código seleccionado por el usuario y en un código de 3 cifras, a insertar dentro de la sentencia SQL de actualización.

4.2.2.15. Clase ParametrosDeConsulta.java.-

Clase utilizada para almacenar la información, existente en los distintos campos de las tablas de la base de datos, cuando se realiza una consulta. Para simplificar la tarea de representación de datos al cliente se le devuelve un Vector donde cada elemento es un objeto de tipo ParámetrosDeConsulta.java. En cada elemento del Vector se recoge una fila de la consulta realizada. El número máximo de campos a consultar está limitado a 18 (número de campos en la tabla “registro_de_accesos”).

En esta clase se presentan 18 campos tipo String descritos como Parametro1, Parametro2, ..., Parametro18 junto a sus métodos de acceso. También existe un método de inicialización de los campos y un método que transforma el valor devuelto por la base de datos a un formato más inteligible.

4.2.2.16. Clase ParametrosPDF.java.-

Clase que almacena información necesaria para generar el informe PDF. Los campos que recogen la información son:

Field Summary	
(package private) static java.util.Vector	**NombresCabecera** Objeto tipo Vector donde cada elemento se corresponde con el nombre de cabecera que se desea aplicar a cada campo para su representación.
(package private) static java.util.Vector	**NombresCampos** Objeto tipo Vector donde cada elemento se corresponde con el nombre del campo a consultar.
(package private) static javax.servlet.http.HttpSession	**Sesion** Objeto tipo HttpSession que recoje las caracteristicas de la sesión del usuario.
(package private) static java.lang.String	**Sql** Objeto tipo String con la sentencia SQL a ejecutar.

Presenta los siguientes métodos de acceso:

Method Summary	
static void	**Clear**() Borra todos los parámetros para formar el informe PDF
static java.util.Vector	**getNombresCabecera**() Devuelve NombresCabecera.
static java.util.Vector	**getNombresCampos**() Devuelve NombresCampos.
static javax.servlet.http.HttpSession	**getSesion**() Devuelve Sesion.
static java.lang.String	**getSql**() Devuelve Sql.
static void	**setNombresCabecera**(java.util.Vector AuxNombresCabecera) Almacena NombresCabecera.
static void	**setNombresCampos**(java.util.Vector AuxNombresCampos) Almacena NombresCampos.
static void	**setSesion**(javax.servlet.http.HttpSession AuxSesion) Almacena Sesion.
static void	**setSql**(java.lang.String AuxSql) Almacena Sql.

4.2.2.17. Clase RegistrarAcceso.java.-

Clase que inserta en tabla "registro_de_accesos" información sobre el acceso de los usuarios a los recursos del Sistema de Información.

Define el siguiente método:

Method Summary	
static java.lang.String	**Registrar**() Método que accede a la tabla "registros_de_accesos" e inserta un nuevo registro con los valores pasados a las clases ControlAcceso.java, ControlResponsableSeguridad.java y ControlCopiaRespaldo.java.

4.2.2.18. Clase RegistrarEntradaSoportesInformáticos.java.-

Clase que introduce en la tabla "registro_de_entrada_de_soportes_informaticos" información sobre la entrada de soportes al Sistema de Información.

Define el siguiente método:

Method Summary	
static java.lang.String	**Registrar**() Método que accede a la tabla "registro_de_entrada_de_soportes_informaticos" e inserta un nuevo registro con los valores pasados al servidor en la clase ControlSoportesInformaticos.java.

4.2.2.19. Clase RegistrarIncidencia.java.-

Clase que inserta en tabla "registro_de_incidencias" información sobre las incidencias de seguridad ocurridas durante el acceso del usuario al Sistema de Información.

Define el siguiente método:

Method Summary	
static java.lang.String	**Registrar**() Método que accede a la tabla "registro_de_incidencias" e inserta un nuevo registro con los valores pasados al servidor en las clases ControlIncidencia.java y ControlRecuperacionDatos.java.

4.2.2.20. Clase RegistrarSalidaSoportesInformáticos.java.-

Clase que introduce en la tabla "registro_de_salida_de_soportes_informaticos" información sobre la salida de soportes del Sistema de Información.

Define el siguiente método:

Method Summary	
static java.lang.String	**Registrar**() Método que accede a la tabla "registro_de_salida_de_soportes_informaticos" e inserta un nuevo registro con los valores pasados al servidor en la clase ControlSoportesInformaticos.java.

4.2.2.21. Clase Registro.java.-

Recoge variables comunes para las clases RegistrarAcceso.java, RegistrarIncidencia.java, RegistrarEntradaSoportesInformaticos.java y RegistrarSalidaSoportesInformaticos.java que almacenan en la base de datos "Registro de Acceso" los datos perteneciente al control de seguridad aplicado a los usuarios que acceden al Sistema de Información.

Presenta los siguientes campos:

Field Summary	
(package private) static java.sql.Connection	**conexion** Objeto tipo Connection con el que se establece la conexión la base de datos "Registro de Acceso".
(package private) static java.lang.String	**Fecha** Objeto tipo String con la fecha actual en la que se produce un nuevo registro.
(package private) static java.lang.String	**Hora** Objeto tipo String con la hora actual en la que se produce un nuevo registro.
(package private) static java.lang.String	**Mensaje** Objeto tipo String con el mensaje de confirmación o de negación del nuevo registro.
(package private) static java.lang.String	**ResponsableSeguridad** Objeto tipo String con el nombre del Responsable de Seguridad.
(package private) static java.lang.String	**Sql** Objeto tipo String con la sentencia SQL a ejecutar.

4.2.2.22. Clase UsuariosDelServidor.java.-

Clase utilizada por el Servidor de Registro de Acceso para comprobar los datos y permisos de los usuarios que acceden a él.

Presenta los siguientes métodos:

Method Summary	
static boolean	**ComprobarAcceso**(java.lang.String IdUsuario, int Recurso) Método que comprueba si el usuario del servidor tiene permiso para acceder a un determinado recurso.
static boolean	**ComprobarLoginPassword**(java.lang.String Login, java.lang.String Password) Método que comprueba el Login y el Password del usuario del servidor.
static boolean	**ComprobarUsuario**(java.lang.String IdUsuario) Método que comprueba si el usuario está registrado para usar el servidor.

4.3. Cliente del Servidor de Registro de Acceso.-

Con el fin de mostrar el funcionamiento del Servidor de Registro de Acceso se ha desarrollado un cliente con la capacidad de representar el acceso de distintos usuarios al Sistema de Información representado en la figura 4.11. Los usuarios accederán a los distintos recursos en función de sus credenciales, y su acceso provocará una respuesta por parte del servidor. Uno de estos usuarios será el "administrador" del servidor y podrá utilizar y configurar todos los parámetros y servicios que ofrece.

Implementamos al cliente con JSP y clases Java. Utilizamos JSP para el diseño de la interfaz de usuario, debido a la facilidad para crear Web dinámicas y clases Java para realizar la transferencia de información entre la aplicación Web y el Servidor de Registro de Acceso.

En el diseño del cliente se ha tenido muy en cuenta la arquitectura presentada por los Sistemas de Información Sanitarios.

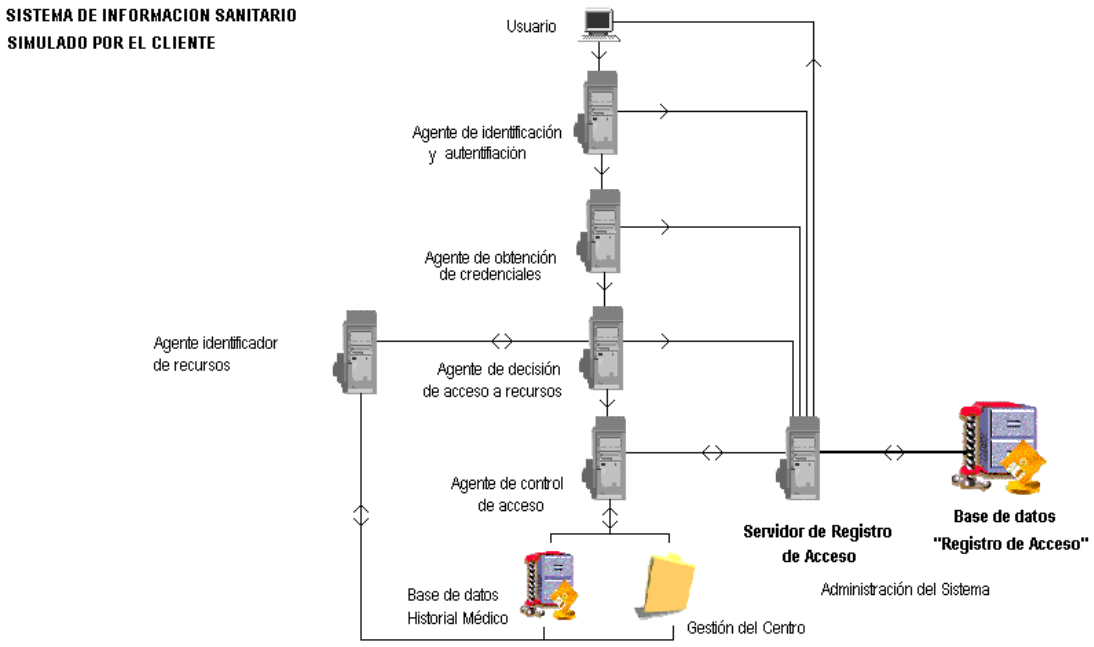

Figura 4.11. Sistema de Información Sanitario simulado por el cliente.

En el cliente se han simulado, simplemente para dar una visión más general del funcionamiento y del entorno del servidor, agentes encargados de identificar y autentificar, obtener credenciales y decidir y controlar el acceso de los usuarios. La implementación de cada uno de estos agentes necesitaría de un desarrollo especifico.

Mostramos a continuación los archivos JSP que implementan los distintos elementos de la arquitectura de la figura 4.11.

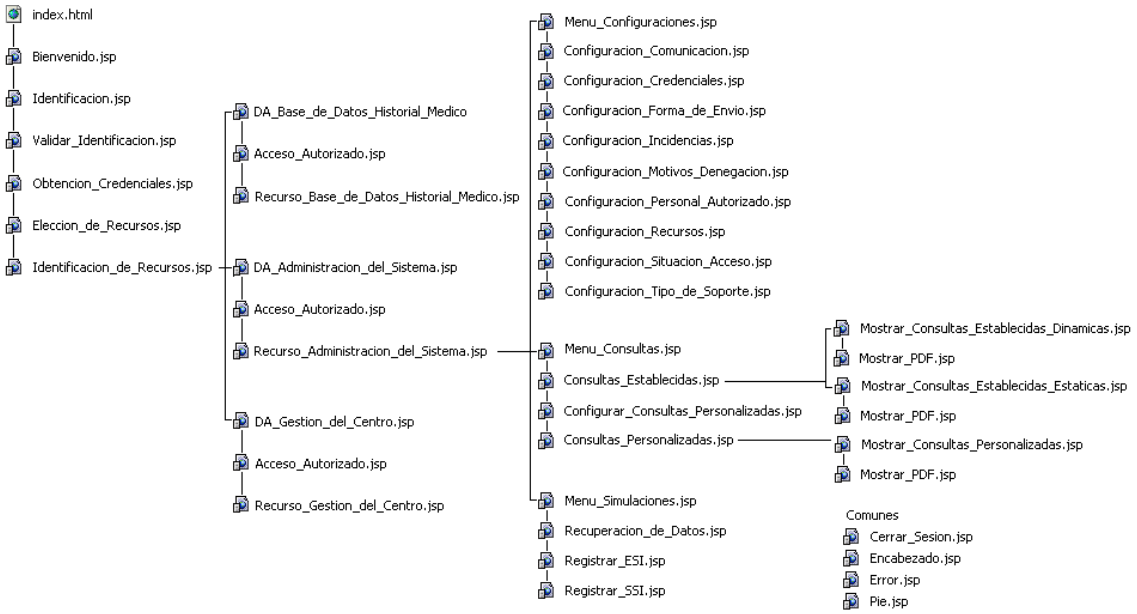

Figura 4.12. Archivos JSP que implementan el cliente.

El acceso por parte de los usuarios a los recursos simulados deberá producirse de forman secuencial como se deduce de las figuras 4.11 y 4.12. En la figura 4.12 se destacan los enlaces principales entre los archivos JSP que forman el cliente.

Pasamos a describir brevemente las funciones que desempeñan los archivos JSP presentados en la figura 4.12.

Archivos JSP que implementan el cliente	Breve Descripción
index.html	Página de bienvenida del cliente del Servidor de Registro de Acceso.
Bienvenido.jsp	JSP que da la bienvenida al Servidor de Registro de Acceso. Controla que la base de datos "Registro de Acceso" haya sido inicializada, creando las tablas necesarias antes de cualquier uso. En el caso que no se haya creado la base de datos se presentará un mensaje en pantalla.
Identificacion.jsp	JSP que forma parte del módulo que simula el agente de identificación y autentificación. Este JSP pide el login y la clave del usuario y la pasa al fichero Validar_Identificacion.jsp.
Validar_Identificacion.jsp	JSP que forma parte del módulo que simula el agente de identificación y autentificación. Este JSP recibe un identificador de usuario y una clave y lo compara con las existentes en el archivo Usuarios_Registrados.dat informando al servidor del resultado de la búsqueda si no ha habido éxito. Si el usuario es identificado se mostrará por pantalla el nombre completo del usuario.
Obtencion_Credenciales.jsp	JSP que simula el agente de obtención de credenciales. Según la identificación del usuario obtendremos unas credenciales que especifican las características del usuario en el Sistema.
Eleccion_de_Recursos.jsp	Este JSP simula el agente de decisión de acceso a recursos. En este JSP se muestras los distintos recursos del Sistema de Información simulado.
Identificacion_de_Recursos.jsp	JSP, incluido en Elección_de_Recursos.jsp, que recoge las propiedades de los recursos del Sistema de Información simulado.
DA_Base_de_Datos_Historial_Medico.jsp	Este JSP presenta el fallo del agente de decisión de acceso sobre el acceso al recurso base de datos Historial Medico.
Recurso_Base_de_Datos_Historial_Medico.jsp	Este JSP presenta la página de inicio del recurso Base de Datos Historial Médico.
DA_Administracion_del_Sistema.jsp	Este JSP presenta el fallo del agente de decisión de acceso sobre el acceso al recurso Administración del Sistema.
Recurso_Administracion_del_Sistema.jsp	Este JSP presenta la página de inicio del recurso Administración del Sistema.

DA_Gestion_de_Centro.jsp	Este JSP presenta el fallo del agente de decisión de acceso sobre el acceso al recurso Gestión del Centro.
Recurso_Gestion_de_Centro.jsp	Este JSP presenta la página de inicio del recurso Gestión del Centro.
Acceso_Autorizado.jsp	Presenta en pantalla la autorización del acceso. Incluido en los JSP "DA_..."
Menu_Configuraciones.jsp	Este JSP presenta las opciones de configuración de las tablas, auxiliares y dedicadas a usuarios del servidor, de la base de datos "Registro de Acceso".
Configuracion_Comunicacion.jsp	Este JSP permite modificar los campos de la tabla "comunicacion" de la base de datos "Registro de Acceso".
Configuracion_Credenciales.jsp	Este JSP permite modificar los campos de la tabla "credenciales" de la base de datos "Registro de Acceso".
Configuracion_Forma_de_Envio.jsp	Este JSP permite modificar los campos de la tabla "forma_de_envio" de la base de datos "Registro de Acceso".
Configuracion_Incidencias.jsp	Este JSP permite modificar los campos de la tabla "incidencias" de la base de datos "Registro de Acceso".
Configuracion_Motivos_Denegacion.jsp	Este JSP permite modificar los campos de la tabla "motivos_de_denegacion" de la base de datos "Registro de Acceso".
Configuracion_Personal_Autorizado.jsp	Este JSP permite modificar los campos de la tabla "personal_autorizado" de la base de datos "Registro de Acceso".
Configuracion_Recursos.jsp	Este JSP permite modificar los campos de la tabla "recursos" de la base de datos "Registro de Acceso".
Configuracion_Situaccion_Acceso.jsp	Este JSP permite modificar los campos de la tabla "situacion_acceso" de la base de datos "Registro de Acceso".
Configuracion_Tipo_de_Soporte.jsp	Este JSP permite modificar los campos de la tabla "tipo_de_soporte" de la base de datos "Registro de Acceso".
Menu_Consultas.jsp	Este JSP presenta las opciones de consulta de la base de datos "Registro de Acceso".
Configurar_Consultas_Personalizadas.jsp	Este JSP muestra un asistente que permite crear consultas personalizadas a la base de datos “Registro de Acceso”.
Consultas_Establecidas.jsp	Este JSP muestra las consultas diseñadas por el programador.
Consultas_Personalizadas.jsp	Este JSP muestra las consultas personalizadas creadas con el asistente Configurar_Consultas_Personalizadas.jsp.
Mostrar_Consultas_Establecidas_Estaticas.jsp	Este JSP muestra el resultado de las consultas estáticas diseñadas por el programador.
Mostrar_Consultas_Establecidas_Dinamicas.jsp	Este JSP muestra el resultado de las consultas dinámicas diseñadas por el programador.
Mostrar_Consultas_Personalizadas.jsp	Este JSP muestra el resultado de las consultas personalizadas.
Mostrar_PDF.jsp	Este JSP muestra en pantalla el informe PDF.
Menu_Simulaciones.jsp	Este JSP presenta las opciones de simulación que verifican el funcionamiento del servidor.
Recuperacion_de_Datos.jsp	Este JSP simula la recuperación de datos almacenados.
Registrar_ESI.jsp	Este JSP simula la entrada de soportes informáticos al Sistema de Información Sanitario.
Registrar_SSI.jsp	Este JSP simula la salida de soportes informáticos al Sistema de Información Sanitario.
Encabezado.jsp	JSP con el encabezado de las páginas que forman el cliente.
Pie.jsp	JSP que recoge el pie de las páginas que forman la aplicación.
Error.jsp	JSP que se muestra en caso que exista alguna excepción en el transcurso de la ejecución del cliente.
Cerrar_Sesion.jsp	JSP que inicializa los parámetros de control de acceso y redirecciona al usuario del cliente a la página de bienvenida.

Tabla 4.1. Breve descripción de los archivos JSP que implementan el cliente.

4.3.1. Definición de los paquetes.-

Como hemos señalado, en la implementación del cliente se utilizan clases Java. Estas han sido empaquetadas en los paquetes “cliente” y “recurso”.

4.3.1.1. Paquete “cliente”.-

En este paquete se encuentran clases utilizadas por el cliente para simular los agentes del Sistema de Información que actúan en el acceso del usuario. Presenta las siguientes clases:

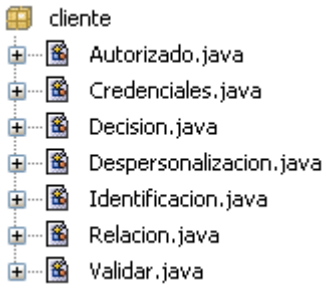

Figura 4.13. Clases definidas en el paquete “cliente”.

Se muestra a continuación una breve descripción de las clases.

Class Summary	
Autorizado	Clase que almacena la información necesaria para el agente de decisión de acceso.
Credenciales	Clase que almacena la información necesaria para el agente de obtención de credenciales.
Decision	Clase que simula simplemente las funciones del agente de decisión de acceso.
Despersonalizacion	Clase que asocia la situación, en la que se produce el acceso, con la despersonalización de ficheros.
Identificacion	Clase que almacena información necesaria para el eagente de identificación y autentificación.
Relacion	Clase que relaciona los códigos de incidencia con los códigos de comunicación.
Validar	Clase que busca en los archivos Usuarios_Registrados.dat y Credenciales.dat los datos de identificación proporcionados por el usuario.

4.3.1.2. Paquete “recurso”.-

El paquete “recurso” presenta las clases dedicadas a la descripción de los recursos del Sistema de Información. Para el diseño de estas clases se han considerado algunas de las recomendaciones “Resource Access Decision Facility Specification” del proyecto CORBAmed desarrollado por el OMG. El “package” presenta las siguientes clases:

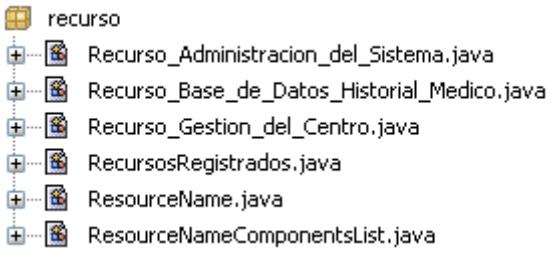

Figura 4.14. Clases definidas en el paquete “recurso”.

Se muestra a continuación una breve descripción de las clases.

Class Summary	
Recurso_Administracion_del_Sistema	Clase que describe el Recurso Administración del Sistema.
Recurso_Base_de_Datos_Historial_Medico	Clase que describe el Recurso Base de Datos Historial Médico.
Recurso_Gestion_del_Centro	Clase que describe el Recurso Gestión del Centro.
RecursosRegistrados	Clase que presenta una clasificación de los recursos del Sistema de Información.
ResourceName	La norma Resource Access Decision (RAD) del proyecto CORBAmed señala que el recurso se describe utilizando un ResourceName o "nombre de recurso" formado por un identificador único de la autoridad de nombrado (ResourceNamingAuthority) y una secuencia de componentes (ResourceNameComponent), donde cada componente esta formado por una pareja nombre-valor.
ResourceNameComponentsList	Clase que almacena la secuencia de componentes que caracterizan al recurso.

4.3.2. Definición de las clases.-

4.3.2.1. Clase Autorizado.java.-

Clase que almacena la información necesaria para el agente de decisión de acceso.

Define el campo:

Field Summary	
(package private) static boolean	**Autorizado** Campo que recoge la decisión del agente de decisión de acceso.

Y recoge los métodos de acceso:

Method Summary	
static void	**Clear**() Borra Autorizado.
static boolean	**getAutorizado**() Devuelve Autorizado.
static void	**setAutorizado**(boolean Autorizacion) Almacena Autorizado.

4.3.2.2. Clase Credenciales.java.-

Clase que almacena la información necesaria para el agente de obtención de credenciales.

Define el campo:

Field Summary	
(package private) static java.lang.String	**Credenciales** Credenciales del usuario que accede al Sistema de Información.

Y recoge los métodos de acceso:

Method Summary	
static void	**Clear**() Borra Credenciales.
static java.lang.String	**getCredenciales**() Devuelve Credenciales.
static void	**setCredenciales**(java.lang.String AuxCredenciales) Almacena Credenciales.

4.3.2.3. Clase Decision.java.-

Clase que simula simplemente las funciones del agente de decisión de acceso. Actualiza el campo "Autorizado" de la clase Autorizado.java si el usuario posee las credenciales necesarias para acceder al recurso seleccionado.

Define el siguiente método:

Method Summary	
static void	**Acceso**(java.lang.String Credenciales, java.lang.String IdRecurso) Método que comprueba las credenciales de un usuario para acceder a un recurso.

4.3.2.4. Clase Despersonalizacion.java.-

Clase que asocia la situación, en la que se produce el acceso, con la despersonalización de ficheros.

Presenta el siguiente método:

Method Summary	
static void	**Acceso**(java.lang.String Codigo_Acceso) Método que recoge la situación, en la que se produce el acceso, y establece si los ficheros consultados fueron despersonalizados.

4.3.2.5. Clase Identificacion.java.-

Clase que almacena información necesaria para el agente de identificación y autentificación.

Presenta los siguientes campos:

Field Summary	
(package private) static java.lang.String	**IdUsuario** Objeto tipo String con la identificación del usuario.
(package private) static java.lang.String	**Password** Objeto tipo String con la clave del usuario.

Y define los siguientes métodos de acceso:

Method Summary	
static void	**Clear**() Borra los parámetros de identificación.
static java.lang.String	**getIdUsuario**() Devuelve IdUsuario.
static java.lang.String	**getPassword**() Devuelve Password.
static void	**setIdUsuario**(java.lang.String AuxIdUsuario) Almacena IdUsuario.
static void	**setPassword**(java.lang.String AuxPassword) Almacena Password.

4.3.2.6. Clase Relacion.java.-

Clase que relaciona los códigos de incidencia con los códigos de comunicación. Cuando se produce una incidencia de seguridad en el sistema, lo habitual es que según el tipo de incidencia generada se informe de una u otra manera.

Presenta el siguiente método:

Method Summary	
static java.lang.String	**Incidencia_Comunicacion**() Método que aplica un código de comunicación a un determinado código de incidencia.

4.3.2.7. Clase Validar.java.-

Clase que busca en los archivos Usuarios_Registrados.dat y Credenciales.dat los datos de identificación proporcionados por el usuario.

Presenta los siguientes métodos:

Method Summary	
static void	**Credenciales**(java.lang.String IdUsuario) Método que busca en el archivo Credenciales.dat las crendenciales del usuario identificado por el agente de identificación y autentificación.
static void	**Identificacion**(java.lang.String Login, java.lang.String Password) Método que busca en el archivo Usuarios_Registrados.dat el login y el password del usuario que intenta acceder al Sistema de Información.

4.3.2.8. Clase Recurso_Admistración_de_Sistema.java.-

Clase que describe el recurso Administración del Sistema. Presenta el siguiente método:

Method Summary	
static ResourceName	**Descripcion**() Método donde se describen los parámetros necesarios para identificar y utilizar el recurso.

4.3.2.9. Clase Recurso_Base_de_Datos_Historial_Medico.java.-

Clase que describe el recurso Base de Datos Historial Médico. Presenta el siguiente método:

Method Summary	
static ResourceName	**Descripcion**() Método donde se describen los parámetros necesarios para identificar y utilizar el recurso.

4.3.2.10. Clase Recurso_Gestion_del_centro.java.-

Clase que describe el recurso Gestión del Centro. Presenta el siguiente método:

Method Summary	
static ResourceName	**Descripcion**() Método donde se describen los parámetros necesarios para identificar y utilizar el recurso.

4.3.2.11. Clase RecursosRegistrados.java.-

Clase que presenta una clasificación de los recursos del Sistema de Información. Cada recurso es identificado por un entero. Para seleccionar la descripción de alguno de los recursos se pasa como argumento el entero que lo identifica. La asignación es la siguiente:

1- Administración del Sistema.
2- Base de Datos Historial Médico.
3- Gestión del Centro.

El método devuelve un objeto de tipo "ResourceName" que caracteriza al recurso.

En la clase se define el siguiente método:

Method Summary	
static ResourceName	**Eleccion**(int Opcion) Método que devuelve las propiedades del recurso seleccionado por el argumento.

4.3.2.12. Clase ResourceName.java.-

La norma Resource Access Decision (RAD) del proyecto CORBAmed señala que el recurso se describe utilizando un ResourceName o "nombre de recurso" formado por un identificador único de la autoridad de nombrado (ResourceNamingAuthority) y una secuencia de componentes (ResourceNameComponent), donde cada componente está formado por una pareja nombre - valor. Esta clase forma objetos tipo ResourceName que describen los recursos del Sistema de Información.

Se definen los siguientes campos:

Field Summary	
(package private) java.lang.String	**ResourceLink** Link, si existe, de la página principal del recurso.
(package private) ResourceNameComponentsList	**ResourceNameComponent** Secuencia de componentes del recurso.
(package private) java.lang.String	**ResourceNamingAuthority** Identificador único de la autoridad de nombrado.

La clase presenta los siguientes métodos de acceso:

Method Summary	
java.lang.String	**getResourceLink**() Devuelve ResourceLink.
ResourceNameComponentsList	**getResourceNameComponent**() Devuelve ResourceNameComponent.
java.lang.String	**getResourceNamingAuthority**() Devuelve ResourceNamingAuthority
void	**setResourceName**(java.lang.String AuxResourceNamingAuthority, java.lang.String AuxResourceLink, ResourceNameComponentsList AuxResourceNameComponent) Almacena ResourceNamingAuthority, ResourceLink y ResourceNameComponent.

4.3.2.13. Clase ResourceNameComponetsList.java-

Clase que almacena la secuencia de componentes que caracterizan al recurso.

Se podrían añadir en esta clase más campos para la descripción del recurso. En esta especificación del cliente los campos recogidos son:

Field Summary	
(package private) java.lang.String	**Descripcion** Breve descripcion del recurso.
(package private) java.lang.String	**TipoRecurso** Tipo del recurso.

Estos campos presentan los siguientes métodos de acceso:

Method Summary	
java.lang.String	**getDescripcion**() Devuelve Descripcion.
java.lang.String	**getTipoRecurso**() Devuelve TipoRecurso.
void	**setDescripcion**(java.lang.String AuxDescripcion) Almacena Descripcion.
void	**setTipoRecurso**(java.lang.String AuxTipoRecurso) Almacena TipoRecurso.