
# SOFTWARE LIBRE DE GESTIÓN WEB DE UNA CONSULTA.

## Gestión de consultas de traumatología

**Autores**
Javier Moscardó Marichalar
Pablo Pozuelo Mayordomo
Guillermo Truan Cano

**Director del proyecto**
Luis Garmendia Salvador

**Proyecto de Sistemas Informáticos, Facultad de Informática, Universidad Complutense de Madrid**

**Curso 2010-2011**


GGT
Gestor de Clínicas
de Traumatología

# Índice

# 1. Introducción

El objetivo de este proyecto es el desarrollo de una aplicación que nos permita realizar la gestión de una clínica de traumatología de manera eficiente utilizando los nuevos medios que las bases de datos actuales nos brindan. En la actualidad el sistema tradicional de agenda y fichas ha quedado obsoleto y cualquier clínica verá incrementada su capacidad de gestión con el uso de esta aplicación.

La aplicación facilita tanto la gestión de la consulta para los servicios de control de pacientes, consultas y quirófanos como la actividad propia de los empleados que pueden consultar la planificación de sus agendas desde cualquier punto con acceso a internet, sabiendo así cual es la carga de trabajo que tienen para cada día, algo de especial utilidad para los quirófanos.

Los pacientes también se verán directamente beneficiados de ventajas que nuestra aplicación proporciona, estos podrán tener acceso a la información relacionada con sus consultas e intervenciones desde cualquier lugar con acceso a internet, pudiendo también pedir cita.

Esta herramienta es flexible, abierta y ampliable de modo que se mejora la gestión, el acceso, la interactividad y la utilidad de la gran cantidad de información de la que se dispone en un negocio de estas características ya que está adaptada para cualquier tipo de usuario (con conocimientos avanzados de informática o sin ellos) además de mejorar la experiencia del paciente.

# 2. Lista de palabras clave

- Spring
- MySQL
- PHP
- Java
- Clínica
- JavaScript
- Ajax
- SVN
- JDBC
- Apache

# 3. Integrantes del proyecto y metodología de trabajo

El proyecto ha sido llevado a cabo por un grupo compuesto por tres alumnos:

Javier MoscardóMarichalar

Pablo Pozuelo Mayordomo

Guillermo Truan Cano

Y un profesor que ha hecho de director del proyecto:

Luis Garmendia Salvador

## 3.1.Jefe del equipo

Tratándose de un grupo reducido no ha existido la figura de un jefe de proyecto como tal si no que los propios miembros del grupo han ido repartiéndose esta tarea según se ha ido considerando necesario.

## 3.2.Equipo de desarrollo

El trabajo se ha realizado de forma descentralizada y por consenso entre las partes involucradas, siendo así repartido el trabajo entre los miembros del grupo para la realización de tareas concretas .

La toma de decisiones importantes, el tratamiento de los problemas surgidos y la resolución de los mismos se han llevado a cabo entre todos los miembros del grupo.

## 3.3.Seguimiento y reuniones

Las reuniones entre los alumnos que forman el grupo se han llevado tanto de manera presencial como virtual, haciendo uso de las instalaciones de la facultad de informática de la Universidad Complutense de Madrid y de los medios telemáticos actuales, como Skype.

Las reuniones con el director del proyecto han tenido lugar en el despacho 435 de la facultad de informática de la UCM.

## 3.4.Comunicación entre el grupo

La comunicación con el director ha sido básicamente a través de intercambio de correos electrónicos.

La comunicación entre los alumnos en cambio ha sido llevada de manera distinta, también se ha hecho uso del correo electrónico pero se ha dado más importancia al uso de la utilidad de Google, llamada Google sites.

## 3.5.Gestión de archivos

La gestión de los archivos que han ido poco a poco componiendo el proyecto ha sido mediante:

### 3.5.1. Google

Google proporciona diversos servicios y aplicaciones web que colaboran al trabajo de grupos en red en desarrollo de proyectos. Todos estos servicios se comparten a través de la cuenta de correo electrónico de Gmail.

#### *3.5.1.1. Code*

Alojamiento de proyectos y repositorios de código libre. Además de alojar el proyecto tiene una API bastante interesante, en la cuál puedes ver el árbol de carpetas del proyecto, un gestor de versiones con el cual puedes comparar distintas versiones y ver los cambios, volver a una versión anterior del proyecto.

#### *3.5.1.2. Groups*

Servicio de comunicación para el grupo, con foro de debate y alojamiento de archivos de proyecto

#### 3.5.1.3. Calendar

El calendario compartido entre los participantes permite fijar fechas de reuniones con alertas y visualización de próximos eventos.

#### 3.5.1.4. Docs

A través de GoogleDocs mantenemos en línea varios documento de texto que podemos modificar los integrantes del proyecto online. Sin necesidad de instalar programas de ofimática.

#### 3.5.1.5. Sites

Sitio web de creación con plantillas, el cual integra otros servicios de Google como son GoogleDocs, GoogleCalendar. En el sitio web se introduce un historial de reuniones, listado de tareas pendientes, vista del calendario común a los miembros del grupo con fechas de reuniones., blog para comentar tecnologías o noticias que pueden ayudar a mejorar la aplicación.

### 3.5.2. Dropbox

Dropbox es un Servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre computadoras y compartir archivos y carpetas con otros.

El cliente de Dropbox permite a los usuarios dejar cualquier archivo en una carpeta designada. Ese archivo es sincronizado en la nube y en todas las demás computadores del cliente de Dropbox. Los archivos en la carpeta de Dropbox pueden entonces ser compartidos con otros usuarios de Dropbox o ser accedidos desde la página Web de Dropbox. Asimismo, los usuarios pueden grabar archivos manualmente por medio de un navegador web.

# 4. Autorización a la UCM

Los autores de este proyecto, Javier MoscardóMarichalar, Pablo Pozuelo Mayordomo y Guillermo Truan Cano autorizamos a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos (no comerciales) tanto la memoria como el código y el prototipo desarrollado.

Fdo. Javier MoscardóMarichalar

Fdo. Pablo Pozuelo Mayordomo

Fdo. Guillermo Truan Cano

# 5. Descripción del Proyecto

Nuestro proyecto consta de dos partes diferenciadas, la aplicación web y la aplicación de sobremesa, ambas gestionan una base de datos en la que se almacena toda la información que la aplicación necesita.

La aplicación web implementa técnicas modernas de gestión de bases de datos como MySQL , PHP o JavaScript y la aplicación de escritorio esta desarrollada en Java e implementa tambien tecnicás de gestión de bases de datos como MySQL o Spring.

De una forma sencilla y con una interfaz amigable el cliente de nuestra herramienta tiene acceso a todo el contenido de su clínica, ya sea tanto a las fichas de los pacientes como al histórico de consultas y quirófanos, pasando por la gestión de sus empleados.

Nuestra aplicación proporciona un servicio que se puede considerar básico y prioritario en una clínica actual y que sin embargo no está tan extendido como cabría esperar, son muchas la clínicas que actualmente trabajan con herramientas muy rudimentarias incluso para la gestión de las fichas de los pacientes por no hablar de la gestión de consultas, citas y partes de quirófanos.

Nosotros venimos a cubrir esta necesidad proponiendo una herramienta ágil, amigable y de gran utilidad, que engloba todas las soluciones que una clínica actual demanda, ofrecemos a nuestros clientes la posibilidad de gestionar desde una misma interfaz todos los aspectos de una clínica de traumatología moderna, es decir proponemos:

- Un sistema de almacenamiento de las fichas de los pacientes que nos permita tener rápido acceso tanto a sus datos personales como a sus datos clínicos.
- Una rápida gestión de horarios de consultas y citas.
- Un eficiente sistema de gestión de quirófanos y listas de espera.
- Una gestión rápida de los empleados de la clínica.

# 6. Aim of the project

The aim of this project is to develop an application that allows us to make the management of a traumatology clinic efficiently using the new tools that the current databases provides. At present the traditional calendar and files is already obsolete and any clinic will increase its managing capacity by using this application.

Application facilitates both the management consulting services for patient such as monitoring dates and surgeries as the activity itself of employees who can consult their schedules from anywhere with Internet access, and knowing which is the load of work they have scheduled for any day, something particularly useful for surgeries.

Patients also will directly benefit from advantages that our application provides, they may have access to information relating to their dates and actions from anywhere with Internet access and they will be also able to arrange a new date.

This tool is flexible, scalable and open so as to improve management, access, interactivity and usefulness of the vast amount of information that is available in a business of this nature because it is suited for any type of user (with advanced computer knowledge or without them). In addition it improves the patient experience

# 7. Requisitos. Casos de Uso

En este capítulo se describen detalladamente los requisitos de una clínica de traumatología, estos han sido desarrollados a partir de reuniones que tuvimos con el Dr. Truan, jefe del Servicio de Cirugía Ortopédica y Traumatología del Hospital Universitario Madrid Monteprincipe en los que se trataron las necesidades que debía cubrir un software como el nuestro.

A continuación se describen los requisitos de aplcaciones de escritorio comunes, para médicos, administrador y auxiliares administrativos.

## 7.1.CASOS DE USO DE LA APLICACIÓN DE ESCRITORIO.

### 7.1.1. CASOS DE USO COMUNES A TODOS LOS ROLES.

| | |
|---|---|
| **[CU-01]** | **Dar de alta paciente** |
| **Objetivo en contexto** | Describe el proceso de añadir un paciente a la clínica. |
| **Entradas** | Datos relativos al nuevo paciente. |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |
| **Salidas** | El nuevo paciente pasa a formar parte de la lista de pacientes en la ventana “Lista |

| | |
|---|---|
| | de pacientes” |
| **Post-condición si éxito** | El paciente se incluye en la base de la aplicación. |
| **Post-condición si fallo** | El paciente no se incluye en la base de la aplicación. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de pacientes.<br>3. Selecciona opción Nuevo paciente<br>4. Muestra pantalla de inserción de datos<br>Del paciente.<br>5. El sistema comprueba que estén especificados los datos necesarios. Si error S1. Si no, envía los datos<br>6. El sistema accede a la estructura de datos<br>empleado. Si no puede acceder, a S2. |
| **Secuencia alternativa** | S1. Campos incompletos o falta de datos. Se solicita al<br>Auxiliar administrativo que introduzca los datos de nuevo.<br>S2. Mensaje de error porque no se puede |

| | |
|---|---|
| | acceder a la Base de Datos. |
| **[CU-02]** | **Modificar datos del paciente** |
| **Objetivo en contexto** | Describe el proceso de modificar los datos de un paciente de la clínica. |
| **Entradas** | Datos relativos al paciente. |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |
| **Salidas** | Los datos son modificados en la lista de pacientes y se muestran en la pestaña del paciente |
| **Post-condición si éxito** | Los datos del paciente se modifican en la base de la aplicación |
| **Post-condición si fallo** | Los datos del paciente no se modifican en la base de la aplicación |
| **Actores** | GUI del Administrador o del médico o del auxiliar, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de pacientes. |

| | |
|---|---|
| | 3. Selecciona opción Modificar paciente.<br><br>4. Muestra la pantalla de inserción de datos del paciente.<br><br>5. El sistema comprueba que estén especificados los datos necesarios. Si error S1. Si no, envía los datos<br><br>6. El sistema accede a la estructura de datos<br><br>paciente. Si no puede acceder, a S2. |
| **Secuencia alternativa** | S1. Campos incompletos o falta de datos. Se solicita al usuario que introduzca los datos de nuevo.<br><br>S2. Mensaje de error porque no se puede acceder a la Base de Datos. |

| | |
|---|---|
| **[CU-03]** | **Dar cita** |
| **Objetivo en contexto** | Describe el proceso de añadir una cita para un paciente. |
| **Entradas** | Datos relativos a la cita (médico, paciente, fecha, horario y tipo de consulta). |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |

| | |
|---|---|
| **Salidas** | No hay mensaje de éxito, el paciente pasa formar parte del parte de consultas para el día seleccionado y el médico elegido. |
| **Post-condición si éxito** | La cita se incluye en la base de la aplicación. |
| **Post-condición si fallo** | La cita no se incluye en la base de la aplicación. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de empleados.<br>3. Selecciona un empleado de la lista de empleados.<br>4. Muestra pantalla de inserción de datos de la consulta.<br>5. El sistema comprueba que estén especificados los datos necesarios. Si error S1. Si no, envía los datos<br>6. El sistema accede a la estructura de datos de las citas. Si no puede acceder, a S2. |

| | |
|---|---|
| **Secuencia alternativa** | S1. Campos incompletos o falta de datos. Se solicita al<br><br>administrador que introduzca los datos de nuevo.<br><br>S2. Mensaje de error porque no se puede acceder a la Base<br><br>de Datos. |

| | |
|---|---|
| **[CU-04]** | **Consultar histórico general de intervenciones** |
| **Objetivo en contexto** | Describe el proceso de consultar el histórico de consultas de la aplicación. |
| **Entradas** | |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |
| **Salidas** | Se muestra la consulta realizada, el listado con las intervenciones de una fecha seleccionada. |
| **Post-condición si éxito** | Se muestra la consulta realizada. |
| **Post-condición si fallo** | No Se muestra la consulta realizada. |

| | |
|---|---|
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña intervenciones.<br>2. El sistema muestra la pantalla de intervenciones.<br>3. Selecciona una fecha determinada.<br>4. Muestra los datos de las intervenciones para la fecha seleccionada. |
| **Secuencia alternativa** | |
| **Restricciones** | En caso de ser el auxiliar administrativo quien realiza la acción solo tendrá acceso al listado del parte de quirófano y no los detalles del mismo |

| | |
|---|---|
| **[CU-05]** | **Consultar histórico de consultas de un paciente.** |
| **Objetivo en contexto** | Describe el proceso de consultar el histórico de consultas asociadas a un paciente de la aplicación. |
| **Entradas** | |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |

| | |
|---|---|
| **Salidas** | Se muestra la consulta realizada, el listado con las intervenciones de una fecha seleccionada |
| **Post-condición si éxito** | Se muestra la consulta realizada. |
| **Post-condición si fallo** | No se muestra la consulta realizada. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de pacientes.<br>3. Selecciona un paciente determinado.<br>4. Muestra la ficha del paciente en la que se inluyen los datos de las intervenciones para la fecha seleccionada. |
| **Secuencia alternativa** | |
| **Restricciones** | En caso de ser el auxiliar administrativo quien realiza la acción solo tendrá acceso al listado del parte de consultas y no los detalles del mismo. En caso de ser el médico, solo puede ver los listados de sus consultas. |

| | |
|---|---|
| **[CU-06]** | **Cambiar la contraseña del propio usuario** |

| | |
|---|---|
| **Objetivo en contexto** | Describe el proceso de modificar la contraseña de acceso al sistema de un empleado |
| **Entradas** | |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |
| **Salidas** | Se muestra un mensaje de éxito. |
| **Post-condición si éxito** | La contraseña del usuario queda modificada. |
| **Post-condición si fallo** | No se cambia la contraseña del usuario. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar el menú archivo.<br>2. Pulsar sobre “Cambiar contraseña”.<br>3. Introducir la contraseña actual.<br>4. Introducir la nueva contraseña.<br>5. Confirmar la nueva contraseña. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-07]** | **Consultar datos del paciente** |
| **Objetivo en contexto** | Describe el proceso de consultar los datos de un paciente de la clínica. |
| **Entradas** | |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |
| **Salidas** | Se muestran los datos del paciente consultado |
| **Post-condición si éxito** | |
| **Post-condición si fallo** | |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de pacientes.<br>3. Selecciona un paciente.<br>4. pinchar sobre él.<br>5. Se abre la pestaña del paciente dónde se pueden ver los datos personales del |

| | |
|---|---|
| **Secuencia alternativa** | paciente. |

| | |
|---|---|
| **[CU-08]** | **Dar de baja paciente** |
| **Objetivo en contexto** | Describe el proceso de eliminar un paciente a la clínica. |
| **Entradas** | |
| **Precondiciones** | El usuario que realiza la acción debe haber hecho login en la aplicación |
| **Salidas** | |
| **Post-condición si éxito** | El paciente se elimina del listado de pacientes. |
| **Post-condición si fallo** | El paciente no se elimina del listado de pacientes. |

| | |
|---|---|
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de pacientes.<br>3. Seleccionar al paciente a eliminar.<br>4. Pulsar botón derecho.<br>5. Pulsar sobre eliminar.<br>6. Confirmar que se desea eliminar. |
| **Secuencia alternativa** | |

### 7.1.2. CASOS DE USO RELATIVOS AL ADMINISTRADOR

El administrador puede realizar cualquier acción dentro de la aplicación que pueda realizar cualquier otro rol. Por otro lado él y sólo él puede realizar las siguientes.

- Registrar un empleado en la aplicación
- Dar de baja un empleado en la aplicación
- Asignar horarios de consulta
- Asignar horarios de intervenciones
- Consultar datos personales de un empleado

| **[CU-01]** | **Registrar un empleado en la aplicación** |
|---|---|
| **Objetivo en contexto** | Describe el proceso de añadir un empleado a la clínica. |
| **Entradas** | Datos relativos al nuevo empleado. |
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que haber hecho login. |
| **Salidas** | Mensaje de éxito o fallo. (No hay mensaje de éxito, solo de error, si se introduce correctamente aparece en la ventana general en el listado según sea Medico, ATS o auxiliar administrativo. |
| **Post-condición si éxito** | El empleado se incluye en la base de la aplicación. |
| **Post-condición si fallo** | El empleado no se incluye en la base de la |

| | |
|---|---|
| | aplicación. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña empleados.<br>2. El sistema muestra la pantalla de empleados.<br>3. Selecciona opción Nuevo empleado<br>4. Muestra pantalla de inserción de datos<br>empleado.<br>4.b) Selección del rol del empleado: Médico, ATS, Auxiliar administrativo.<br>5. El sistema comprueba que estén especificados<br>necesarios. Si error S1. Si no, envía los datos<br>6. El sistema accede a la estructura de datos<br>empleado. Si no puede acceder, a S2. |
| **Secuencia alternativa** | S1. Campos incompletos o falta de datos. Se solicita al<br>administrador que introduzca los datos de nuevo.<br>S2. Mensaje de error porque no se puede acceder a la Base de Datos. |

| | |
|---|---|
| **[CU-02]** | **Dar de baja un empleado en la aplicación** |
| **Objetivo en contexto** | Describe el proceso de quitar un empleado de la clínica. |
| **Entradas** | Identificador del empleado a eliminar |
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que<br>haber hecho login. El empleado debe de existir en la aplicación |
| **Salidas** | Mensaje de éxito o fallo. |
| **Post-condición si éxito** | El empleado se marca como borrado en la base de la aplicación. |
| **Post-condición si fallo** | El empleado no se marca como borrado en la base de la aplicación. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña empleados.<br>2. El sistema muestra la pantalla de empleados. |

| | |
|---|---|
| **Secuencia alternativa** | 3. Seleccionar el empleado a eliminar<br>4. Selecciona opción Eliminar empleado<br>5. Confirmar que se desea eliminar el empleado |

| | |
|---|---|
| **[CU-03]** | **Asignar horarios consulta** |
| **Objetivo en contexto** | Describe el proceso por el que el el administrador asigna horarios de consulta a un medico. |
| **Entradas** | Datos relativos a las fechas que se quieren agregar y al médico al que se le van a aplicar |
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que<br>haber hecho login. |
| **Salidas** | Mensaje de éxito o fallo. (No hay mensaje de éxito, solo de error), si se introduce correctamente aparece en la ventana de parte de consultas. |

| | |
|---|---|
| **Post-condición si éxito** | El horario queda asignado al médico seleccionado. |
| **Post-condición si fallo** | El horario no se le asigna al médico seleccionado. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña empleados.<br>2. El sistema muestra la pantalla de empleados.<br>3. Selecciona opción Asignar horarios consultas.<br>4. Muestra pantalla de inserción horarios para empleados.<br>5. Seleccionar el médico, el turno y la fecha.<br>6. Si no hay error se muestra la pantalla de inserción de horarios para empleados con el médico que habíamos seleccionado y la fecha, si hubo error S1. |
| **Secuencia alternativa** | S1. El turno seleccionado para el médico coincide con su horario de intervenciones en alguno de los días, se nos muestra este error. |

| | |
|---|---|
| **[CU-04]** | **Asignar horarios intervenciones** |
| **Objetivo en contexto** | Describe el proceso por el que el administrador asigna turnos de quirófano a un médico. |
| **Entradas** | Datos relativos a las fechas y turnos que se quieren asignar a un médico, así como el médico al que se le quiere asignar dicho turno. |
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que<br>haber hecho login. |
| **Salidas** | Mensaje de éxito o fallo. (No hay mensaje de éxito, solo de error), si se introduce correctamente aparece en la ventana de parte de intervenciones. |
| **Post-condición si éxito** | El turno de quirófano queda asignado al médico seleccionado. |
| **Post-condición si fallo** | El turno de quirófano no queda asignado al médico seleccionado. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |

| | |
|---|---|
| **Secuencia normal** | 1. Seleccionar la pestaña empleados.<br>2. El sistema muestra la pantalla de empleados.<br>3. Selecciona opción Asignar horarios intervenciones.<br>4. Muestra pantalla de inserción de horarios de intervención.<br>5. Seleccionar el médico, el turno y la fecha.<br>6. Si no hay error se muestra la pantalla de inserción de intervenciones con el médico que habíamos seleccionado y la fecha, si hubo error S1. |
| **Secuencia alternativa** | S1. El turno seleccionado para el médico coincide con su horario de consultas en alguno de los días, se nos muestra este error. |

| | |
|---|---|
| **[CU-05]** | **Consultar datos personales de un empleado** |
| **Objetivo en contexto** | Describe el proceso de consultar los datos un empleado de la clínica. |
| **Entradas** | Identificador del empleado a consultar |
| **Precondiciones** | El administrador es el que realiza la |

| | |
|---|---|
| | operación y por tanto tiene que<br>haber hecho login. El empleado debe de existir en la aplicación |
| **Salidas** | |
| **Post-condición si éxito** | Se muestran los datos personales de un empleado. |
| **Post-condición si fallo** | No se muestran los datos personales de un empleado. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña empleados.<br>2. El sistema muestra la pantalla de empleados.<br>3. Seleccionar el empleado a consultar.<br>4. Se muestra la ficha del empleado. |
| **Secuencia alternativa** | |

### 7.1.3. CASOS DE USO RELATIVOS A LOS MÉDICOS

- Cambiar fecha intervención
- Eliminar intervencion
- Crear una nueva consulta para un paciente
- Modificar una consulta para un paciente
- Añadir antecedentes clínicos a un paciente.
- Consultar Lista de espera (pacientes pendientes de entregar pruebas).
- Programar una intervención
- Modificar una intervención
- Apuntar entrega de pruebas de un paciente para una intervención ya programada.
- Modificar parte de quirófano.
- Apuntar una intervención como ya creada.

| **[CU-01]** | **Cambiar fecha intervención** |
|---|---|
| **Objetivo en contexto** | Describe el proceso de cambiar la fecha de una intervención ya programada. |
| **Entradas** | |
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que haber hecho login. |
| **Salidas** | |
| **Post-condición si éxito** | El empleado se incluye en la base de la aplicación. |

| | |
|---|---|
| **Post-condición si fallo** | El empleado no se incluye en la base de la aplicación. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña quirofano.<br>2. El sistema muestra la pantalla de quirofanos.<br>3. Seleccionar la intervención buscando por fecha.<br>4. Pinchar sobre editar intervención<br>5. Se abre la ventana de datos de la intervención<br>6. Pinchando sobre el recuadro de la fecha se podrá modificar |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-02]** | **Eliminar intervencion** |
| **Objetivo en contexto** | Describe el proceso de eliminar una intervencion un empleado a la clínica. |
| **Entradas** | |

| | |
|---|---|
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que<br>haber hecho login. |
| **Salidas** | |
| **Post-condición si éxito** | Se elimina una intervención |
| **Post-condición si fallo** | Se mantiene la intervención en la base de datos |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña quirofanos.<br>2. El sistema muestra la pantalla de quirofanos.<br>3. Selecciona una fecha.<br>4. Muestra las intervenciones para esa fecha.<br>5.Haciendo clic con el botón derecho se muestra la opción de eliminar.<br>6. Pinchando sobre eliminar aparece un mensaje de confirmación, si se pulsa “SI” S1 si no S2. |

| | |
|---|---|
| **Secuencia alternativa** | S1. Se elimina la intervención.<br>S2. No se elimina la intervención. |

| | |
|---|---|
| **[CU-03]** | **Crear una nueva consulta para un paciente** |
| **Objetivo en contexto** | Describe el proceso añadir una nueva consulta a un paciente citado. |
| **Entradas** | Datos relativos a la nueva consulta. |
| **Precondiciones** | El médico es el que realiza la operación y por tanto tiene que haber hecho login. |
| **Salidas** | |
| **Post-condición si éxito** | Se añade la nueva consulta a la ficha del paciente. |
| **Post-condición si fallo** | No se añade la consulta a la ficha del paciente. |
| **Actores** | GUI del médico, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña consultas.<br>2. El sistema muestra la pantalla de consultas. |

| | |
|---|---|
| **Secuencia alternativa** | 3. Selecciona un paciente determinado dentro de una fecha de citas valida.<br>4. Muestra los datos del paciente.<br>5. Pinchar sobre el botón añadir consulta.<br>6. Rellenar los datos de la misma. |

| | |
|---|---|
| **[CU-04]** | **Modificar una consulta para un paciente** |
| **Objetivo en contexto** | Describe el proceso modificar una consulta de un paciente. |
| **Entradas** | Datos relativos a la consulta. |
| **Precondiciones** | El médico es el que realiza la operación y por tanto tiene que haber hecho login. |
| **Salidas** | |
| **Post-condición si éxito** | Se añaden las modificaciones sobre la consulta en la ficha del paciente. |
| **Post-condición si fallo** | No se añaden las modificaciones sobre la consulta en la ficha del paciente. |

| | |
|---|---|
| **Actores** | GUI del médico, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña consultas.<br>2. El sistema muestra la pantalla de consultas.<br>3. Selecciona un paciente determinado dentro de una fecha de citas valida.<br>4. Muestra los datos del paciente.<br>5. Pinchar sobre la consulta que se quiere modificar.<br>6. Modificar los datos de la consulta. |
| **Secuencia alternativa** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de pacientes.<br>3. Selecciona un paciente determinado.<br>4. Muestra los datos del paciente.<br>5. Pinchar sobre la consulta que se quiere modificar.<br>6. Modificar los datos de la consulta. |

| | |
|---|---|
| **[CU-05]** | **Añadir antecedentes clínicos a un paciente.** |

| | |
|---|---|
| **Objetivo en contexto** | Describe el proceso de añadir antecedentes clínicos a un paciente. |
| **Entradas** | Datos relativos a los antecedentes del paciente. |
| **Precondiciones** | El médico es el que realiza la operación y por tanto tiene que haber hecho login. |
| **Salidas** | |
| **Post-condición si éxito** | Se añaden las modificaciones sobre la ficha del paciente. |
| **Post-condición si fallo** | No se añaden las modificaciones sobre la ficha del paciente. |
| **Actores** | GUI del médico, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña consultas.<br>2. El sistema muestra la pantalla de consultas.<br>3. Selecciona un paciente determinado dentro de una fecha de citas valida.<br>4. Muestra los datos del paciente.<br>5. Pinchar sobre el botón “añadir antecedentes” |

| | |
|---|---|
| | 6. Introducir los detalles. |
| **Secuencia alternativa** | 1. Seleccionar la pestaña pacientes.<br>2. El sistema muestra la pantalla de pacientes.<br>3. Selecciona un paciente determinado.<br>4. Muestra los datos del paciente.<br>5. Pinchar sobre el botón “añadir antecedentes”<br>6. Introducir los detalles. |

| | |
|---|---|
| **[CU-06]** | **Consultar Lista de espera (pacientes pendientes de entregar pruebas).** |
| **Objetivo en contexto** | Describe el proceso de consultar el listado de pacientes pendientes de entregar las pruebas. |
| **Entradas** | |
| **Precondiciones** | El médico es el que realiza la acción y debe haber hecho login en la aplicación |
| **Salidas** | Se muestra la consulta realizada, el listado con los pacientes pendientes de entregar las pruebas. |

| | |
|---|---|
| **Post-condición si éxito** | Se muestra la consulta realizada. |
| **Post-condición si fallo** | No Se muestra la consulta realizada. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña intervenciones.<br>2. El sistema muestra la pantalla de intervenciones.<br>3. pulsar sobre el botón “muestra en espera”.<br>4. Se abre un listado con los detalles solicitados. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-07]** | **Programar una intervención** |
| **Objetivo en contexto** | Describe el proceso de programar una intervención. |
| **Entradas** | Datos relativos a la inbtervención |
| **Precondiciones** | El médico es el que realiza la acción y |

| | |
|---|---|
| | debe haber hecho login en la aplicación |
| **Salidas** | |
| **Post-condición si éxito** | Se añade la intervención a la base de datos. |
| **Post-condición si fallo** | No se añade la intervención a la base de datos. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña consultas.<br>2. Elegir al paciente que tenemos citado en el parte de consultas.<br>3. Pulsar sobre el botón “añadir intevención”.<br>4. Rellenar los campos requeridos y pulsar aceptar |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-08]** | **Modificar una intervención** |

| | |
|---|---|
| **Objetivo en contexto** | Describe el proceso de modificar los detalles de una intervención |
| **Entradas** | Datos a modificar de la intervención. |
| **Precondiciones** | El médico es el que realiza la acción y debe haber hecho login en la aplicación |
| **Salidas** | |
| **Post-condición si éxito** | Se modifican los datos de la intervención en la base de datos. |
| **Post-condición si fallo** | No se modifican los datos de la intervención en la base de datos. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña intervenciones.<br>2. El sistema muestra la pantalla de intervenciones.<br>3. Seleccionar una fecha<br>4. Elegir la intervención a modificar.<br>5. Pinchar sobre el botón “modificar intervención”<br>6. Cambiar los datos que se quieren modificar. |

| | |
|---|---|
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-09]** | **Apuntar entrega de pruebas de un paciente para una intervención ya programada.** |
| **Objetivo en contexto** | Describe el proceso marcar una intervención ya programada con pruebas ya entregadas |
| **Entradas** | |
| **Precondiciones** | El médico es el que realiza la acción y debe haber hecho login en la aplicación |
| **Salidas** | |
| **Post-condición si éxito** | Se marca en la base de datos como “entregadas las pruebas” y se le añade la fecha actual como la fecha en la que fueron entregadas. |
| **Post-condición si fallo** | No se marca en la base de datos como “entregadas las pruebas” y se le añade la fecha actual como la fecha en la que fueron entregadas. |

| | |
|---|---|
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña consultas.<br>2. Elegir al paciente que tenemos citado en el parte de consultas.<br>3. Pulsar sobre la intervención en la ficha del paciente<br>4. Se abre una pantalla en la que se muestran los detalles de la intervención.<br>5. Marcar la casilla de verificación indicando que el paciente ha entregado las pruebas. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-10]** | **Modificar parte de quirófano.** |
| **Objetivo en contexto** | Describe el proceso de modificar cualquier aspecto del parte de quirófano. |
| **Entradas** | |

| | |
|---|---|
| **Precondiciones** | El médico es el que realiza la acción y debe haber hecho login en la aplicación |
| **Salidas** | |
| **Post-condición si éxito** | Se modifica el parte de quirófano. |
| **Post-condición si fallo** | No se modifica el parte de quirófano. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña intervenciones.<br>2. El sistema muestra la pantalla de intervenciones.<br>3. Seleccionar la intervención que se desea eliminar.<br>4. Pulsar sobre ella con el botón derecho y selccionar la opción “eliminar”.<br>5. Confirmar que se desea eliminar. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-10]** | **Apuntar una intervención como ya realizada.** |
| **Objetivo en contexto** | Describe el proceso hacer constar en la aplicación que una intervención ya ha sido realizada. |
| **Entradas** | |
| **Precondiciones** | El médico es el que realiza la acción y debe haber hecho login en la aplicación |
| **Salidas** | |
| **Post-condición si éxito** | Se marca en la base de datos una intervención como ya realizada y se actualizan los datos de la misma. |
| **Post-condición si fallo** | No se marca en la base de datos una intervención como ya realizada y se actualizan los datos de la misma. |
| **Actores** | GUI, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña intervenciones.<br>2. El sistema muestra la pantalla de intervenciones.<br>3. Seleccionar la intervención que se desea marcar como ya realizada. |

| | |
|---|---|
| **Secuencia alternativa** | 4. Pulsar el botón “Editar intervención”.<br><br>5. Rellenar los detalles de la intervención y marcar la casilla de verificación “intervenido” |

### 7.1.4. CASOS DE USO RELATIVOS AL AUXILIAR ADMINISTRATIVO

- Consultar histórico general de consultas
- Consultar histórico general de intervenciones

| | |
|---|---|
| **[CU-01]** | **Consultar histórico general de consultas.** |
| **Objetivo en contexto** | Describe el proceso de consultar el histórico de consultas de la aplicación. |
| **Entradas** | |
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que haber hecho login. |
| **Salidas** | Se muestra la consulta realizada, el listado con las consultas de una fecha seleccionada para un médico seleccionado. |
| **Post-condición si éxito** | Se muestra la consulta. |
| **Post-condición si fallo** | No se muestra la consulta. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |

| | |
|---|---|
| **Secuencia normal** | 1. Seleccionar la pestaña consultas.<br>2. El sistema muestra la pantalla de consultas.<br>3. Selecciona una fecha determinada.<br>4.Selecciona un médico determinado.<br>5. Muestra los datos de las consultas para la fecha seleccionada. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CU-09]** | **Consultar histórico general de intervenciones** |
| **Objetivo en contexto** | Describe el proceso de consultar el histórico de intervenciones de la aplicación. |
| **Entradas** | |
| **Precondiciones** | El administrador es el que realiza la operación y por tanto tiene que haber hecho login. |
| **Salidas** | Se muestra la consulta realizada, el listado con las intervenciones de una fecha |

| | |
|---|---|
| | seleccionada. |
| **Post-condición si éxito** | Se muestra la consulta realizada. |
| **Post-condición si fallo** | No se muestra la consulta realizada. |
| **Actores** | GUI del Administrador, Servidor, Base de Datos |
| **Secuencia normal** | 1. Seleccionar la pestaña intervenciones.<br>2. El sistema muestra la pantalla de intervenciones.<br>3. Selecciona una fecha determinada.<br>4. Muestra los datos de las intervenciones para la fecha seleccionada. |
| **Secuencia alternativa** | |

## 7.2.CASOS DE USO DE LA APLICACIÓN WEB.

La aplicación web da servicio tanto a médicos como a pacientes, estos pueden acceder a ella a través de la dirección http://consulta2011.sytes.net/web/ una vez dentro deberán de loguearse como usuarios ya registrados o registrase si no lo han hecho previamente. Los médicos no pueden registrarse, deberán hacerlo en la clínica.

A continuación se describen losrequisitos de la aplicación web para médicos y pacientes.

### 7.2.1. CASOS DE USO COMUNES A LOS DOS ACTORES (MÉDICOS Y PACIENTES).

- Recuper contraseña
- Ver datos personales
- Modificar datos de usuario

| | |
|---|---|
| **[CUW-01]** | **Recuperar contraseña.** |
| **Objetivo en contexto** | Describe la acción de recuperación de la contraseña de acceso a los servicios web. |
| **Entradas** | Nombre de usuario y dni del usuario. |
| **Precondiciones** | El usuario debe estar dado de alta previamente en la web. |
| **Salidas** | Se envía un correo electrónico a la cuenta de correo con la que se registro el usuario. |
| **Post-condición si éxito** | Se genera una nueva contraseña para el usuario que la solicito. |

| | |
|---|---|
| **Post-condición si fallo** | No se genera una nueva contraseña para el usuario que la solicito. |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el icono de recuperación de contraseña.<br>2. Rellenar los datos del formulario de recuperación de contraseña.<br>3. Recuperar de la cuenta de correo la nueva contraseña. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CUW-02]** | **Ver datos personales.** |
| **Objetivo en contexto** | Describe la acción de consultar los datos personales del usuario. |
| **Entradas** | |
| **Precondiciones** | El usuario debe logueado en la web. |

| | |
|---|---|
| **Salidas** | Formulario con los datos personales del usuario. |
| **Post-condición si éxito** | Se muestran los datos personales del usuario. |
| **Post-condición si fallo** | No se muestran los datos personales del usuario. |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “ver datos personales” del menú principal.<br>2. Se muestran los datos personales. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CUW-03]** | **Modificar los datos de usuario.** |
| **Objetivo en contexto** | Describe la acción de modificar los datos usuario. |
| **Entradas** | |

| | |
|---|---|
| **Precondiciones** | El usuario debe logueado en la web. |
| **Salidas** | Formulario con los datos personales del usuario. |
| **Post-condición si éxito** | Se modifican los datos de usuario. |
| **Post-condición si fallo** | No se modifican los datos de usuario. |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “modificar datos de usuario” del menú principal.<br>2. Se muestra un formulario para modificar la contraseña.<br>3. Introducir los datos solicitados. |
| **Secuencia alternativa** | |

### 7.2.2. CASOS DE USO A LOS MÉDICOS

- Ver su calendario de consultas
- Ver su calendario de intervenciones

| | |
|---|---|
| **[CUW-01]** | **Ver calendario de consultas.** |
| **Objetivo en contexto** | Describe la acción de mostrar los detalles de sus días de consulta. |
| **Entradas** | |
| **Precondiciones** | El usuario debe estar logueado en la web con un perfil de médico. |
| **Salidas** | Formulario con los datos de las consultas de una fecha determinada. |
| **Post-condición si éxito** | Se muestran los detalles de las consultas de la fecha seleccionada. |
| **Post-condición si fallo** | No se muestran los detalles de las consultas de la fecha seleccionada. |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “mi horario de consultas” del menú principal. |

| | |
|---|---|
| **Secuencia alternativa** | 2. Seleccionar una fecha.<br>3. Seleccionar un turno. |

| | |
|---|---|
| **[CUW-01]** | **Ver calendario de intervenciones.** |
| **Objetivo en contexto** | Describe la acción de mostrar los detalles de sus días de quirófano. |
| **Entradas** | |
| **Precondiciones** | El usuario debe estar logueado en la web con un perfil de médico. |
| **Salidas** | Formulario con los datos de las intervenciones de una fecha determinada. |
| **Post-condición si éxito** | Se muestran los detalles de las intervenciones de la fecha seleccionada. |
| **Post-condición si fallo** | No se muestran los detalles de las intervenciones de la fecha seleccionada. |

| | |
|---|---|
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “mi horario de intervenciones” del menú principal.<br>2. Seleccionar una fecha. |
| **Secuencia alternativa** | |

### 7.2.3. CASOS DE USO RELATIVOS A LOS PACIENTES

- Pedir cita
- Consultar detalles de la próxima intervención.
- Consultar detalles de la próxima consulta.

| | |
|---|---|
| **[CUW-01]** | **Pedir cita.** |
| **Objetivo en contexto** | Describe la acción de pedir cita en la consulta. |
| **Entradas** | |
| **Precondiciones** | El usuario debe estar logueado en la web. |

| | |
|---|---|
| **Salidas** | Mensaje de confirmación de cita programada. |
| **Post-condición si éxito** | El usuario queda citado en la aplicación para un médico, una fecha y una hora determinados. |
| **Post-condición si fallo** | El usuario no queda citado en la aplicación para un médico, una fecha y una hora determinados. |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “pedir cita” del menú principal.<br>2. Seleccionar un médico, una fecha y un Horario.<br>3. Elegir una hora entre las que constan como “libres”.<br>4. Confirmar la cita. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CUW-02]** | **Consultar detalles de la próxima consulta.** |

| | |
|---|---|
| **Objetivo en contexto** | Describe la acción de consultar los detalles de la próxima cita. |
| **Entradas** | |
| **Precondiciones** | El usuario debe estar logueado en la web. |
| **Salidas** | Se muestran los detalles de la próxima cita. |
| **Post-condición si éxito** | Se muestran los detalles de la próxima cita. |
| **Post-condición si fallo** | |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “próxima cita”.<br>2. Se muestra un desplegable con los detalles de la misma. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CUW-03]** | **Consultar histórico de sus consultas.** |

| | |
|---|---|
| **Objetivo en contexto** | Describe la acción de consultar los detalles de sus consultas ya pasadas. |
| **Entradas** | |
| **Precondiciones** | El usuario debe estar logueado en la web. |
| **Salidas** | Se muestran los detalles de las consultas ya pasadas. |
| **Post-condición si éxito** | Se muestran los detalles de las consultas ya pasadas. |
| **Post-condición si fallo** | |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “útima consulta”.<br>2. Se muestra un desplegable con los detalles de la mismas. |
| **Secuencia alternativa** | |

| | |
|---|---|
| **[CUW-04]** | **Consultar detalles de la próxima intervención.** |

| | |
|---|---|
| **Objetivo en contexto** | Describe la acción de consultar los detalles de la próxima intervención. |
| **Entradas** | |
| **Precondiciones** | El usuario debe estar logueado en la web. |
| **Salidas** | Se muestran los detalles de la próxima intervención. |
| **Post-condición si éxito** | Se muestran los detalles de la próxima intervención. |
| **Post-condición si fallo** | |
| **Actores** | GUI de la web, Servidor, Base de Datos |
| **Secuencia normal** | 1. Pinchar sobre el botón “próxima intervención”.<br>2. Se muestra un desplegable con los detalles de la misma. |
| **Secuencia alternativa** | |

# 8. Arquitectura, diseño e implementación

## 8.1.INTRODUCCIÓN

En el desarrollo de las distintas aplicaciones (Web y Java) hemos tenido en cuenta las últimas tecnologías de desarrollo de páginas web, lenguajes de programación y servicios que ofrecía el mercado y se ajustaban a nuestras necesidades.

## 8.2.BASE DE DATOS

La Base de Datos se encuentra físicamente en el servidor, el cual tiene instalado el PhpMyAdmin que nos ha facilitado la creación y manipulación de la misma.

La tecnología de almacenamiento escogida es InnoDB que, a diferencia de MyISAM, sí nos permite trabajar con procedimientos almacenados así como mantener la integridad referencial por medio de claves foráneas. El cotejamiento de la misma será UTF8.

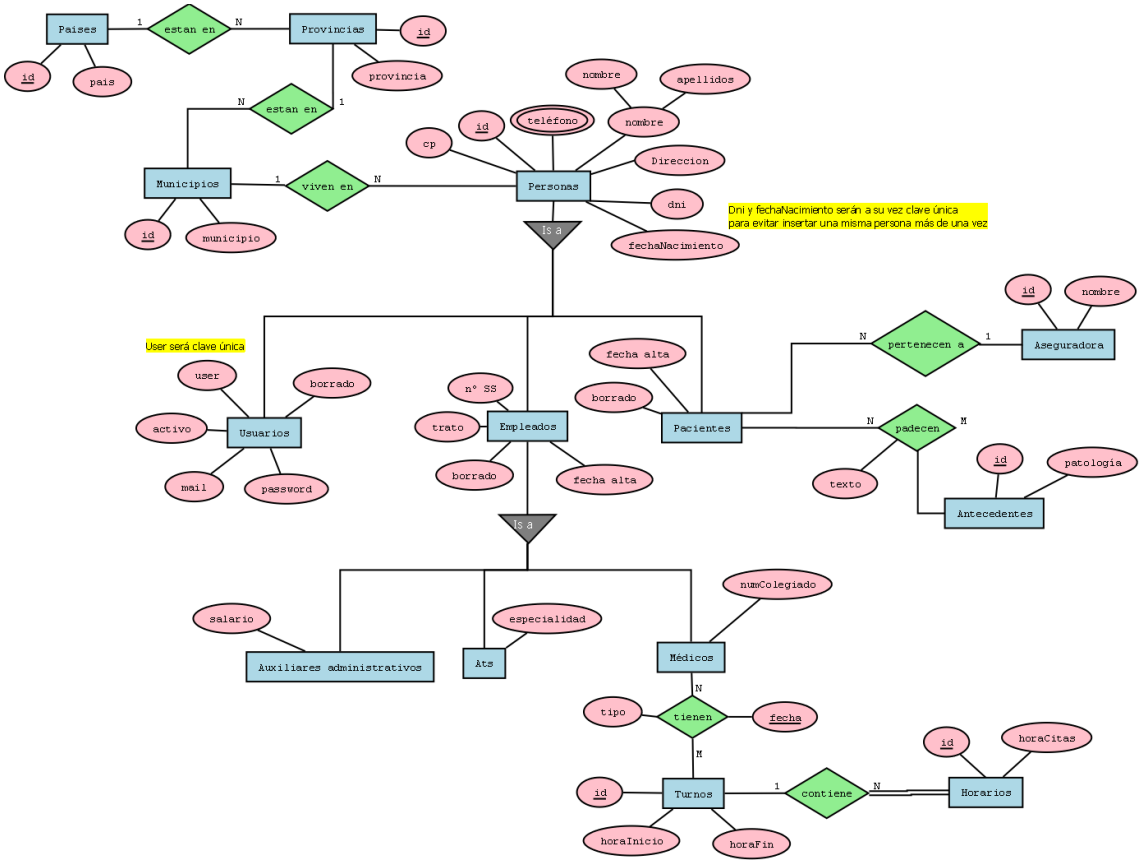
La base de datos que se presenta a continuación tiene el propósito de almacenar todos aquellos datos que son relevantes, sin redundancias, para una clínica de traumatología.

### 8.2.1. MODELO E/R

A lo largo del desarrollo del proyecto, los cambios en los requisitos se han producido sin previo aviso, viéndose la base de datos modificada en todo momento. Finalmente recogemos el modelo resultante dividido en 3 partes claramente diferenciadas como son: actores, consultas e intervenciones.

#### *8.2.1.1. ACTORES*

Como vemos tenemos una entidad Personas de las que heredan Usuarios, Empleados y Pacientes. De la entidad Empleados heredan Ats, Auxiliares Administrativos y Médicos.

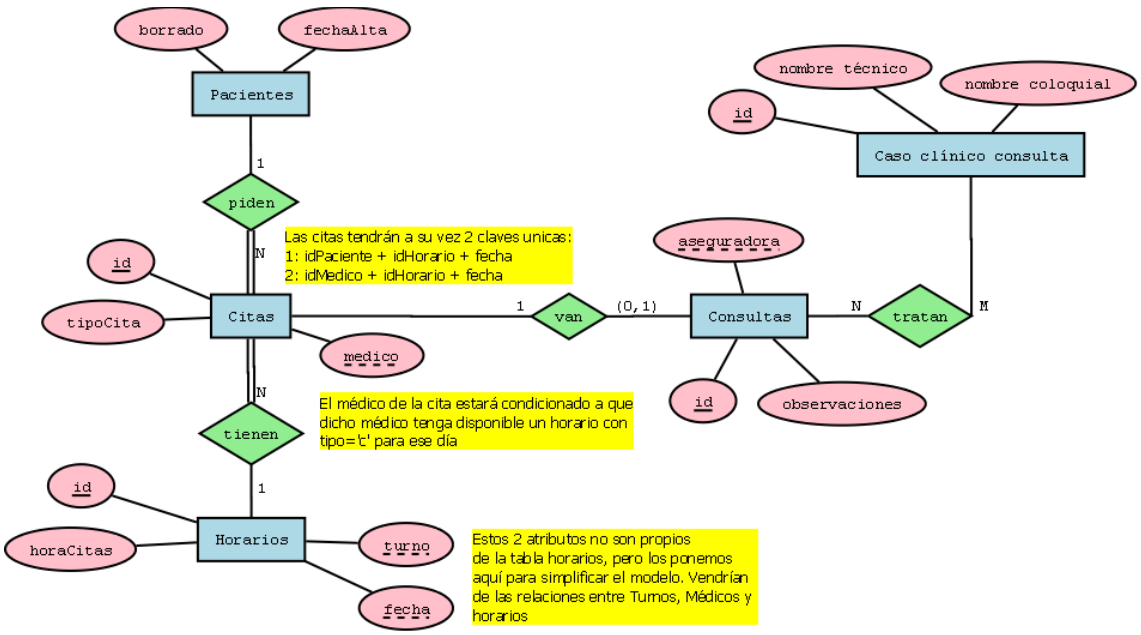



En las entidades Usuarios, Empleados y Pacientes almacenamos el atributo 'borrado' que nos permitirá hacer un borrado lógico de esa persona. Lo mantenemos en las 3 entidades porque puede darse el caso de que se requiera borrar a un empleado que a su vez era paciente de la clínica, por lo que solo marcaremos el *flag*de borrado a 1 en la tabla empleados quedando sin borrar como paciente.

Las personas serán de un municipio, éste a su vez de una provincia y ésta de un país. Cada paciente tendrá una lista de antecedentes y una compañía de seguros asociada. Cada médico tendrá un turno asignado (de momento *mañana/tarde* pero flexible ante cualquier ampliación). Cada turno tendrá asociado un conjunto de horarios que serán los correspondientes a los horarios de las citas. La relación entre turnos y médicos nos dará la fecha y el tipo, siendo éste de tipo consulta o de tipo intervención. El tipo no forma parte de la clave primaria para que no pueda haber un mismo médico que el mismo día tenga consultas e intervenciones en el mismo turno.

#### *8.2.1.2. CONSULTAS*

Cada consulta está asociada a una cita. Dicha cita tendrá asociado un paciente, un médico y un horario. El médico de la cita estará condicionado a que dicho médico tenga un turno de tipo consulta para esa fecha. La participación de citas en las relaciones 'piden' y 'tienen' es total porque no puede haber citas sin paciente o sin horario.

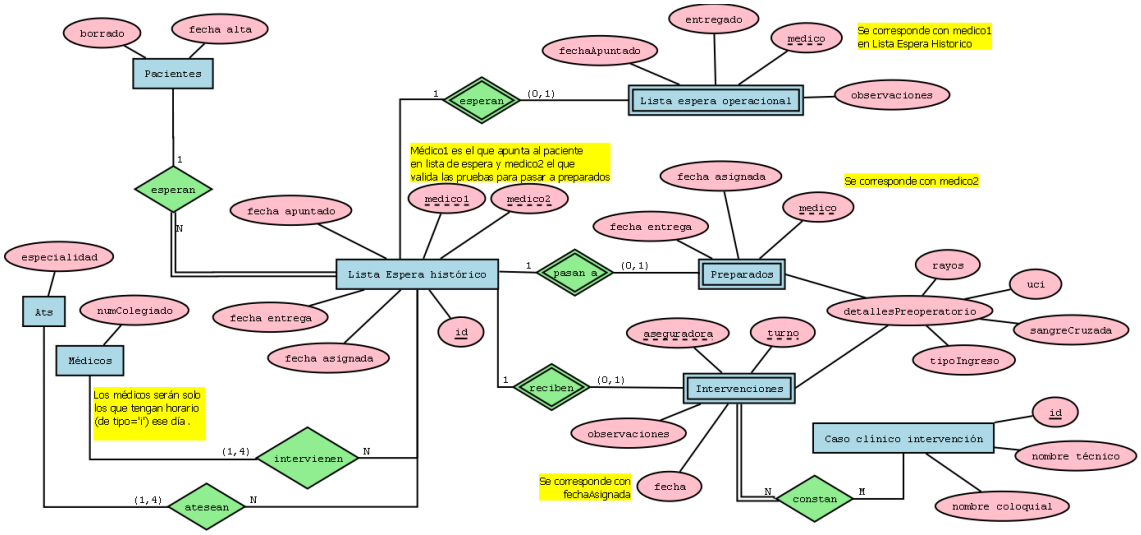



Por otra parte vemos que guardamos la aseguradora en cada consulta. Habitualmente se rellenará con la aseguradora que tenga asociado dicho paciente (tal y como vimos en el modelo ER relativo a los actores), pero haciéndolo así nos permitirá saber con exactitud qué compañía cubre cada consulta independientemente de que el paciente cambie o no. A su vez, una consulta, además de las observaciones, contendrá también una lista de casos clínicos que trata.

#### *8.2.1.3. INTERVENCIONES*

Las entidades ListaEsperaOperacional y Preparados serán tablas operacionales que solo contendrán los registros correspondientes a ListaEsperaHistórico que aún no hayan sido intervenidos, una vez intervenidos se borrarán quedando solo en ListaEsperaHistórico e Intervenciones.

Estará en ListaEsperaOperacional mientras no haya entregado las pruebas y estará en Preparados una vez el médico valide las pruebas del preoperatorio. Las consultas SQL (mientras sea posible) irán contra estas tablas operacionales.




Como podemos observar a partir del modelo, cada intervención tendrá 1 paciente y de 1 a 4 ats y médicos. Los médicos estarán condicionados a que tengan para ese día en ese turno horario de tipo intervención.

Análogamente a lo que vimos para Consultas en el apartado anterior, guardaremos la aseguradora para la Intervención. De la misma manera que antes cada intervención constará de una lista de casos clínicos y de un campo de observaciones.

ListaEsperaOperacional, Preparados e Intervenciones serían entidades débiles puesto que su clave primaria es la PK de ListaEsperaHistorico (ID).

### 8.2.2. MODELO RELACIONAL

Del anterior modelo ER obtendremos el siguiente modelo relacional. Distinguiremos 3 apartados: creación de tablas, índices y procedimientos almacenados.

#### *8.2.2.1. CREACIÓN DE TABLAS*

Nota de la implementación utilizaremos el prefijo 'r_' para nombrar a las tablas que procedan de una relación del modelo ER y el prefijo 't_' para las tablas originadas por una entidad.

Pasamos a mostrar el DDL de las tablas en orden alfabético. Para cada tabla mostraremos comentadas las relaciones que existan para sus atributos indexados. En el apartado siguiente veremos cómo se añaden las claves foráneas a estos índices.

```
--
-- Estructura de tabla para la tabla `r_atesean`
--
CREATE TABLE IF NOT EXISTS`r_atesean` (
`idAts` int(10) unsigned NOT NULL,
  `idIntervencion` int(10) unsigned NOT NULL,
PRIMARYKEY(`idAts`,`idIntervencion`),
KEY`idIntervencion` (`idIntervencion`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `r_atesean`:
--   `idAts`
```

```
--       `t_ats` -> `idAts`
--   `idIntervencion`
--       `t_listaesperahistorico` -> `id`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `r_constan`
--
CREATETABLEIFNOTEXISTS`r_constan` (
`idCaso` int(10) unsigned NOT NULL,
  `idIntervencion` int(10) unsigned NOT NULL,
PRIMARYKEY(`idCaso`,`idIntervencion`),
KEY`idIntervencion` (`idIntervencion`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `r_constan`:
--   `idCaso`
--       `t_casosclinicosintervenciones` -> `idCaso`
--   `idIntervencion`
--       `t_listaesperahistorico` -> `id`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `r_horariosmedicos`
--
CREATE TABLE IF NOT EXISTS`r_horariosmedicos` (
`idTurno` tinyint(3) unsigned NOT NULL,
  `idMedico` int(10) unsigned NOT NULL,
  `fecha` dateNOT NULL,
  `tipo` enum('c','i') NOT NULL,
PRIMARYKEY(`idTurno`,`idMedico`,`fecha`),
KEY`idMedico` (`idMedico`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `r_horariosmedicos`:
--   `idTurno`
--       `t_turnos` -> `idTurno`
--   `idMedico`
--       `t_medicos` -> `idMedico`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `r_intervienen`
--
CREATE TABLE IF NOT EXISTS`r_intervienen` (
`idMedico` int(10) unsigned NOT NULL,
  `idIntervencion` int(10) unsigned NOT NULL,
PRIMARYKEY(`idMedico`,`idIntervencion`),
KEY`idIntervencion` (`idIntervencion`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `r_intervienen`:
--   `idMedico`
--       `t_medicos` -> `idMedico`
--   `idIntervencion`
```

```
--       `t_listaesperahistorico` -> `id`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `r_padecen`
--
CREATE TABLE IF NOT EXISTS`r_padecen` (
`idPaciente` int(10) unsigned NOT NULL,
  `idAntecedente` smallint(5) unsigned NOT NULL,
`texto` textNOT NULL,
PRIMARYKEY(`idPaciente`,`idAntecedente`),
KEY`idAntecedente` (`idAntecedente`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `r_padecen`:
--   `idPaciente`
--       `t_pacientes` -> `idPaciente`
--   `idAntecedente`
--       `t_antecedentes` -> `idAntecedente`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `r_roles`
--
CREATE TABLE IF NOT EXISTS`r_roles` (
`idUsuario` int(10) unsigned NOT NULL,
  `idRol` tinyint(3) unsigned NOT NULL,
PRIMARY KEY(`idUsuario`,`idRol`),
KEY`idRol` (`idRol`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `r_roles`:
--   `idUsuario`
--       `t_usuarios` -> `idUsuario`
--   `idRol`
--       `t_rol` -> `idRol`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_antecedentes`
--
CREATE TABLE IF NOT EXISTS`t_antecedentes` (
`idAntecedente` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
`patologia` varchar(100) NOT NULL,
PRIMARYKEY(`idAntecedente`)
) ENGINE=InnoDB  DEFAULTCHARSET=utf8;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_aseguradora`
--
CREATE TABLE IF NOT EXISTS`t_aseguradora` (
`idAseguradora` tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
`nombre` varchar(50) NOT NULL,
PRIMARY KEY(`idAseguradora`)
```

```
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_ats`
--
CREATE TABLE IF NOT EXISTS`t_ats` (
`idAts` int(10) unsigned NOT NULL,
  `especialidad` varchar(100) NOT NULL,
  `borrado` tinyint(1) unsigned NOT NULL,
PRIMARY KEY(`idAts`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_ats`:
--   `idAts`
--       `t_empleados` -> `idEmpleado`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_auxiliaresadministrativos`
--
CREATE TABLE IF NOT EXISTS`t_auxiliaresadministrativos` (
`idAuxAdmin` int(10) unsigned NOT NULL,
  `salario` float(6,2) unsigned NOT NULL,
  `borrado` tinyint(3) unsigned NOT NULL,
PRIMARY KEY(`idAuxAdmin`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_auxiliaresadministrativos`:
--   `idAuxAdmin`
--       `t_empleados` -> `idEmpleado`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_casosclinicosconsultas`
--
CREATE TABLE IF NOT EXISTS`t_casosclinicosconsultas` (
`idCaso` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`nombreTecnico` varchar(100) NOT NULL,
  `nombreColoquial` varchar(100) NOT NULL,
PRIMARY KEY(`idCaso`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_casosclinicosintervenciones`
--
CREATE TABLE IF NOT EXISTS`t_casosclinicosintervenciones` (
`idCaso` int(10) unsigned NOT NULL,
`nombreTecnico` varchar(100) NOT NULL,
  `nombreColoquial` varchar(100) NOT NULL,
PRIMARY KEY(`idCaso`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

-- --------------------------------------------------------
```

```
--
-- Estructura de tabla para la tabla `t_citas`
--
CREATE TABLE IF NOTEXISTS `t_citas` (
`idCita` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `idPaciente` int(10) unsigned NOT NULL,
  `idMedico` int(10) unsigned NOT NULL,
  `fecha` dateNOT NULL,
  `idHorario` tinyint(3) unsigned NOT NULL,
`tipoCita` enum('Urgencias','Online','Tlf','Presencial') NOT NULL,
PRIMARY KEY(`idCita`),
UNIQUE KEY`idMedico_2` (`idMedico`,`fecha`,`idHorario`),
UNIQUE KEY`idPaciente_2` (`idPaciente`,`fecha`,`idHorario`),
KEY`idPaciente` (`idPaciente`),
KEY`idMedico` (`idMedico`),
KEY`fecha` (`fecha`),
KEY`idHorario` (`idHorario`)
) ENGINE=InnoDB  DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_citas`:
--   `idPaciente`
--       `t_pacientes` -> `idPaciente`
--   `idMedico`
--       `t_medicos` -> `idMedico`
--   `idHorario`
--       `t_horarios` -> `idHorario`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_consultas`
--
CREATE TABLE IF NOT EXISTS`t_consultas` (
`idConsulta` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `idAseguradora` tinyint(3) unsigned NOT NULL,
`observaciones` textNOT NULL,
PRIMARY KEY(`idConsulta`),
KEY`idAseguradora` (`idAseguradora`)
) ENGINE=InnoDB  DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_consultas`:
--   `idAseguradora`
--       `t_aseguradora` -> `idAseguradora`
--   `idConsulta`
--       `t_citas` -> `idCita`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_empleados`
--
CREATE TABLE IF NOT EXISTS`t_empleados` (
`idEmpleado` int(10) unsigned NOT NULL,
  `numSS` char(12) NOT NULL,
  `fechaAlta` dateNOT NULL,
  `trato` enum('Dr.','Dra.','Don','Doña') NOT NULL DEFAULT 'Dr.',
PRIMARY KEY(`idEmpleado`),
UNIQUE KEY`numSS` (`numSS`)
```

```
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_empleados`:
--   `idEmpleado`
--       `t_personas` -> `idPersona`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_horarios`
--
CREATE TABLE IF NOT EXISTS`t_horarios` (
`idHorario` tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
  `horaCitas` timeNOT NULL,
  `idTurno` tinyint(3) unsigned NOT NULL,
PRIMARYKEY(`idHorario`),
KEY`idTurno` (`idTurno`)
) ENGINE=InnoDB  DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_horarios`:
--   `idTurno`
--       `t_turnos` -> `idTurno`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_intervenciones`
--
CREATE TABLE IF NOT EXISTS`t_intervenciones` (
`idIntervencion` int(10) unsigned NOT NULL,
  `idAseguradora` tinyint(3) unsigned NOT NULL,
  `fecha` dateNOT NULL,
  `idTurno` tinyint(3) unsigned NOT NULL,
  `observaciones` varchar(255) NOT NULL,
  `hospital` enum('a','h','i') NOT NULL,
  `rayos` tinyint(1) NOT NULL,
  `uci` tinyint(1) NOT NULL,
`sc` tinyint(1) NOT NULL,
PRIMARYKEY(`idIntervencion`),
KEY`idAseguradora` (`idAseguradora`),
KEY`idTurno` (`idTurno`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_intervenciones`:
--   `idAseguradora`
--       `t_aseguradora` -> `idAseguradora`
--   `idIntervencion`
--       `t_listaesperahistorico` -> `id`
--   `idTurno`
--       `t_turnos` -> `idTurno`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_listaesperahistorico`
--
CREATE TABLE IF NOT EXISTS`t_listaesperahistorico` (
`id` int(10) unsigned NOT NULL,
```

```
  `idPaciente` int(10) unsigned NOT NULL,
`fechaApuntado` dateNOT NULL,
  `fechaEntrega` dateNOT NULL,
  `fechaAsignada` dateNOT NULL,
  `medico1` int(10) unsigned NOT NULL COMMENT 'médico que apunta',
`medico2` int(10) unsigned NOT NULL COMMENT 'médico que confirma las pruebas',
PRIMARYKEY(`id`),
KEY`idPaciente` (`idPaciente`),
KEY`medico1` (`medico1`),
KEY`medico2` (`medico2`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_listaesperahistorico`:
--   `idPaciente`
--       `t_pacientes` -> `idPaciente`
--   `medico1`
--       `t_medicos` -> `idMedico`
--   `medico2`
--       `t_medicos` -> `idMedico`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_listaesperaoperacional`
--
CREATE TABLE IF NOT EXISTS`t_listaesperaoperacional` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `idPaciente` int(10) unsigned NOT NULL,
  `fecha` dateNOT NULL,
  `observaciones` varchar(255) NOT NULL,
  `medico` int(10) unsigned NOT NULL,
  `entregado` tinyint(1) unsigned NOT NULL,
PRIMARYKEY(`id`),
KEY`medico` (`medico`),
KEY`idPaciente` (`idPaciente`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_listaesperaoperacional`:
--   `idPaciente`
--       `t_pacientes` -> `idPaciente`
--   `medico`
--       `t_medicos` -> `idMedico`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_medicos`
--
CREATE TABLE IF NOT EXISTS`t_medicos` (
`idMedico` int(10) unsigned NOT NULL,
  `numColegiado` char(9) NOT NULL,
  `borrado` tinyint(1) unsigned NOT NULL,
PRIMARY KEY(`idMedico`),
UNIQUE KEY`numColegiado` (`numColegiado`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_medicos`:
--   `idMedico`
```

```
--       `t_empleados` -> `idEmpleado`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_municipios`
--
CREATE TABLE IF NOT EXISTS`t_municipios` (
`idMunicipio` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `idProvincia` smallint(5) unsigned NOT NULL,
`municipio` varchar(100) NOT NULL,
PRIMARYKEY(`idMunicipio`),
KEY`idProvincia` (`idProvincia`)
) ENGINE=InnoDB  DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_municipios`:
--   `idProvincia`
--       `t_provincias` -> `idProvincia`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_pacientes`
--
CREATE TABLE IF NOT EXISTS`t_pacientes` (
`idPaciente` int(10) unsigned NOT NULL,
  `fechaAlta` dateNOT NULL,
  `borrado` tinyint(1) unsigned NOT NULL,
  `idAseguradora` tinyint(3) unsigned NOT NULL,
PRIMARYKEY(`idPaciente`),
KEY`idAseguradora` (`idAseguradora`)
) ENGINE=InnoDB DEFAULTCHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_pacientes`:
--   `idPaciente`
--       `t_personas` -> `idPersona`
--   `idAseguradora`
--       `t_aseguradora` -> `idAseguradora`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_paises`
--
CREATE TABLE IF NOT EXISTS`t_paises` (
`idPais` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
`pais` varchar(100) NOT NULL,
PRIMARYKEY(`idPais`)
) ENGINE=InnoDB  DEFAULTCHARSET=utf8;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_personas`
--
CREATE TABLE IF NOT EXISTS `t_personas` (
`idPersona` int(10) unsigned NOT NULL AUTO_INCREMENT,
`nombre` varchar(50) NOT NULL,
  `apellidos` varchar(100) NOT NULL,
```

```
`dni` char(9) NOT NULL,
  `fechaNac` date NOT NULL,
  `direccion` varchar(250) NOT NULL,
  `idMunicipio` int(10) unsigned NOT NULL,
`cp` char(5) NOT NULL,
PRIMARY KEY (`idPersona`),
UNIQUE KEY `dni_fecha` (`dni`,`fechaNac`),
KEY `idMunicipio` (`idMunicipio`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_personas`:
--   `idMunicipio`
--       `t_municipios` -> `idMunicipio`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_preparados`
--
CREATE TABLE IF NOT EXISTS`t_preparados` (
`idIntervencion` int(10) unsigned NOT NULL,
  `idPaciente` int(10) unsigned NOT NULL,
  `idAseguradora` tinyint(3) unsigned NOT NULL,
  `idMedico` int(10) unsigned NOT NULL,
`fechaEntrega` dateNOT NULL,
  `fechaAsignada` dateNOT NULL,
`idTurno` tinyint(3) unsigned NOT NULL,
  `hospital` enum('a','h','i') NOT NULL COMMENT 'Hospital de dia, ambulante,
ingreso',
  `rayos` tinyint(1) NOT NULL COMMENT 'Rayos X',
  `uci` tinyint(1) NOT NULL,
  `sc` tinyint(1) NOT NULL COMMENT 'Sangre cruzada',
PRIMARY KEY(`idIntervencion`),
KEY`idPaciente` (`idPaciente`),
KEY`idMedico` (`idMedico`),
KEY`idTurno` (`idTurno`),
KEY`idAseguradora` (`idAseguradora`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_preparados`:
--   `idPaciente`
--       `t_pacientes` -> `idPaciente`
--   `idIntervencion`
--       `t_listaesperahistorico` -> `id`
--   `idMedico`
--       `t_medicos` -> `idMedico`
--   `idTurno`
--       `t_turnos` -> `idTurno`
--   `idAseguradora`
--       `t_aseguradora` -> `idAseguradora`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_provincias`
--
CREATE TABLE IF NOT EXISTS`t_provincias` (
`idProvincia` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
  `provincia` varchar(100) NOT NULL,
```

```
  `pais` smallint(5) unsigned NOT NULL,
PRIMARY KEY(`idProvincia`),
KEY `pais` (`pais`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_provincias`:
--   `pais`
--       `t_paises` -> `idPais`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_rol`
--
CREATE TABLE IF NOT EXISTS `t_rol` (
`idRol` tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
`rol` varchar(30) NOT NULL,
PRIMARY KEY (`idRol`)
) ENGINE=InnoDB  DEFAULTCHARSET=utf8;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_telefonos`
--
CREATE TABLE IF NOTEXISTS `t_telefonos` (
`idPersona` int(10) unsigned NOT NULL,
`telefono` char(9) NOT NULL,
PRIMARY KEY(`idPersona`,`telefono`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_telefonos`:
--   `idPersona`
--       `t_personas` -> `idPersona`

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_turnos`
--
CREATE TABLE IF NOT EXISTS`t_turnos` (
`idTurno` tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
  `horaInicio` time NOT NULL,
  `horaFin` time NOT NULL,
PRIMARY KEY(`idTurno`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--
-- Estructura de tabla para la tabla `t_usuarios`
--
CREATE TABLE IF NOT EXISTS`t_usuarios` (
`idUsuario` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user` varchar(50) NOT NULL,
  `mail` varchar(80) NOT NULL,
  `password` char(40) NOT NULL COMMENT `Password encriptado con SHA1`,
  `activo` tinyint(1) unsigned NOT NULL,
  `borrado` tinyint(1) unsigned NOT NULL,
```

```
PRIMARY KEY(`idUsuario`),
UNIQUE KEY`user` (`user`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

--
-- RELACIONES PARA LA TABLA `t_usuarios`:
--   `idUsuario`
--       `t_personas` -> `idPersona`
```

Fig. Creación de tablas

#### *8.2.2.2. ÍNDICES*

Añadimos las claves foráneas para todos los índices que fueron definidos previamente. En todos los casos borraremos y actualizaremos en cascada.
Aparece ordenado alfabéticamente por el nombre de la tabla a la que pertenecen dichos índices.

```
--
-- Filtros para la tabla `r_atesean`
--
ALTER TABLE`r_atesean`
ADD CONSTRAINT`r_atesean_ibfk_1` FOREIGN KEY(`idAts`) REFERENCES`t_ats` (`idAts`) ON
DELETE CASCADE ON UPDATE CASCADE,
ADDCONSTRAINT`r_atesean_ibfk_2` FOREIGNKEY(`idIntervencion`)
REFERENCES`t_listaesperahistorico` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `r_constan`
--
ALTER TABLE `r_constan`
ADD CONSTRAINT `r_constan_ibfk_2` FOREIGN KEY (`idCaso`) REFERENCES
`t_casosclinicosintervenciones` (`idCaso`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `r_constan_ibfk_3` FOREIGN KEY (`idIntervencion`) REFERENCES
`t_listaesperahistorico` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `r_horariosmedicos`
--
ALTER TABLE`r_horariosmedicos`
ADD CONSTRAINT`r_horariosmedicos_ibfk_1` FOREIGN KEY(`idTurno`) REFERENCES`t_turnos`
(`idTurno`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`r_horariosmedicos_ibfk_2` FOREIGN KEY(`idMedico`)
REFERENCES`t_medicos` (`idMedico`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `r_intervienen`
--
ALTER TABLE`r_intervienen`
ADD CONSTRAINT`r_intervienen_ibfk_1` FOREIGN KEY(`idMedico`) REFERENCES `t_medicos`
(`idMedico`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`r_intervienen_ibfk_2` FOREIGN KEY(`idIntervencion`)
REFERENCES`t_listaesperahistorico` (`id`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `r_padecen`
--
```

```
ALTERTABLE`r_padecen`
ADD CONSTRAINT`r_padecen_ibfk_1` FOREIGN KEY(`idPaciente`) REFERENCES`t_pacientes`
(`idPaciente`) ON DELETE CASCADE ON UPDATE CASCADE,
ADDCONSTRAINT`r_padecen_ibfk_2` FOREIGN KEY(`idAntecedente`)
REFERENCES`t_antecedentes` (`idAntecedente`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `r_roles`
--
ALTER TABLE`r_roles`
ADD CONSTRAINT`r_roles_ibfk_1` FOREIGN KEY(`idUsuario`) REFERENCES`t_usuarios`
(`idUsuario`) ON DELETE CASCADE ON UPDATE CASCADE,
ADDCONSTRAINT`r_roles_ibfk_2` FOREIGNKEY(`idRol`) REFERENCES`t_rol` (`idRol`) ON
DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_ats`
--
ALTER TABLE`t_ats`
ADD CONSTRAINT`t_ats_ibfk_1` FOREIGN KEY(`idAts`) REFERENCES`t_empleados`
(`idEmpleado`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_auxiliaresadministrativos`
--
ALTER TABLE`t_auxiliaresadministrativos`
ADD CONSTRAINT`t_auxiliaresadministrativos_ibfk_1` FOREIGN KEY(`idAuxAdmin`)
REFERENCES `t_empleados` (`idEmpleado`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_citas`
--
ALTER TABLE`t_citas`
ADD CONSTRAINT `t_citas_ibfk_1` FOREIGN KEY(`idPaciente`) REFERENCES`t_pacientes`
(`idPaciente`) ON DELETE CASCADE ON UPDATE CASCADE,
ADDCONSTRAINT`t_citas_ibfk_2` FOREIGNKEY(`idMedico`) REFERENCES`t_medicos`
(`idMedico`) ON DELETE CASCADE ON UPDATE CASCADE,
ADDCONSTRAINT`t_citas_ibfk_3` FOREIGN KEY(`idHorario`) REFERENCES`t_horarios`
(`idHorario`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_consultas`
--
ALTER TABLE`t_consultas`
ADD CONSTRAINT`t_consultas_ibfk_4` FOREIGN KEY(`idAseguradora`) REFERENCES
`t_aseguradora` (`idAseguradora`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`t_consultas_ibfk_5` FOREIGN KEY(`idConsulta`) REFERENCES `t_citas`
(`idCita`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_empleados`
--
ALTER TABLE `t_empleados`
ADD CONSTRAINT `t_empleados_ibfk_1` FOREIGN KEY (`idEmpleado`) REFERENCES
`t_personas` (`idPersona`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_horarios`
--
ALTER TABLE `t_horarios`
```

```
ADD CONSTRAINT `t_horarios_ibfk_1` FOREIGN KEY (`idTurno`) REFERENCES `t_turnos`
(`idTurno`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_intervenciones`
--
ALTER TABLE`t_intervenciones`
ADD CONSTRAINT`t_intervenciones_ibfk_2` FOREIGN KEY (`idAseguradora`) REFERENCES
`t_aseguradora` (`idAseguradora`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`t_intervenciones_ibfk_3` FOREIGN KEY (`idIntervencion`)
REFERENCES`t_listaesperahistorico` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`t_intervenciones_ibfk_4` FOREIGN KEY(`idTurno`) REFERENCES`t_turnos`
(`idTurno`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_listaesperahistorico`
--
ALTER TABLE `t_listaesperahistorico`
ADD CONSTRAINT `t_listaesperahistorico_ibfk_1` FOREIGN KEY (`idPaciente`) REFERENCES
`t_pacientes` (`idPaciente`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `t_listaesperahistorico_ibfk_2` FOREIGN KEY (`medico1`) REFERENCES
`t_medicos` (`idMedico`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `t_listaesperahistorico_ibfk_3` FOREIGN KEY (`medico2`) REFERENCES
`t_medicos` (`idMedico`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_listaesperaoperacional`
--
ALTER TABLE`t_listaesperaoperacional`
ADD CONSTRAINT`t_listaesperaoperacional_ibfk_1` FOREIGN KEY(`idPaciente`)
REFERENCES`t_pacientes` (`idPaciente`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`t_listaesperaoperacional_ibfk_2` FOREIGN KEY(`medico`) REFERENCES
`t_medicos` (`idMedico`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_medicos`
--
ALTER TABLE`t_medicos`
ADD CONSTRAINT`t_medicos_ibfk_1` FOREIGN KEY(`idMedico`) REFERENCES `t_empleados`
(`idEmpleado`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_municipios`
--
ALTER TABLE `t_municipios`
ADD CONSTRAINT `t_municipios_ibfk_1` FOREIGN KEY (`idProvincia`) REFERENCES
`t_provincias` (`idProvincia`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_pacientes`
--
ALTER TABLE `t_pacientes`
ADD CONSTRAINT `t_pacientes_ibfk_1` FOREIGN KEY (`idPaciente`) REFERENCES
`t_personas` (`idPersona`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `t_pacientes_ibfk_2` FOREIGN KEY (`idAseguradora`) REFERENCES
`t_aseguradora` (`idAseguradora`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_personas`
--
ALTER TABLE `t_personas`
```

```
ADD CONSTRAINT `t_personas_ibfk_1` FOREIGN KEY (`idMunicipio`) REFERENCES
`t_municipios` (`idMunicipio`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_preparados`
--
ALTERTABLE`t_preparados`
ADD CONSTRAINT`t_preparados_ibfk_2` FOREIGNKEY(`idPaciente`) REFERENCES`t_pacientes`
(`idPaciente`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`t_preparados_ibfk_3` FOREIGN KEY(`idIntervencion`) REFERENCES
`t_listaesperahistorico` (`id`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT`t_preparados_ibfk_4` FOREIGN KEY(`idMedico`) REFERENCES`t_medicos`
(`idMedico`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`t_preparados_ibfk_5` FOREIGN KEY(`idTurno`) REFERENCES `t_turnos`
(`idTurno`) ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT`t_preparados_ibfk_6` FOREIGN KEY(`idAseguradora`) REFERENCES
`t_aseguradora` (`idAseguradora`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_provincias`
--
ALTERTABLE`t_provincias`
ADDCONSTRAINT`t_provincias_ibfk_1` FOREIGNKEY(`pais`) REFERENCES`t_paises` (`idPais`)
ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_telefonos`
--
ALTERTABLE`t_telefonos`
ADDCONSTRAINT`t_telefonos_ibfk_1` FOREIGNKEY(`idPersona`) REFERENCES`t_personas`
(`idPersona`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Filtros para la tabla `t_usuarios`
--
ALTER TABLE`t_usuarios`
ADDCONSTRAINT`t_usuarios_ibfk_1` FOREIGNKEY(`idUsuario`) REFERENCES`t_personas`
(`idPersona`) ON DELETE CASCADE ON UPDATE CASCADE
```

Fig. Índices

#### 8.2.2.3. PROCEDIMIENTOS ALMACENADOS

Para algunos de los procesos tenemos almacenados en la Base de Datos varios *Stored Procedures* que llamaremos desde la aplicación Java y la aplicación Web. Estos procedimientos, como veremos, reducen el tráfico de red, favorecen el SQL estático, centralizan la lógica en la base de datos facilitando el mantenimiento y simplifican el código de las GUIs..

Pasamos a listar los procesos junto a su procedimiento almacenado.

##### 8.2.2.3.1. Insertar un Médico

El procedimiento recibirá todos los parámetros necesarios como parámetros de entrada. Insertará una nueva entrada en la tabla **t_personas**. Una vez insertado recuperará el *idPersona* que es un valor declarado como auto incrementable e insertará los correspondientes registros en las tablas **t_empleados**, **t_medicos**, **t_usuarios** y **r_roles.** El rol correspondiente al médico es el 3.

Observar que siempre que insertamos en la tabla **t_usuarios**encriptaremos el password con la función SHA1 que nos devolverá una cadena de 40 caracteres.

```
CREATE PROCEDURE`insertaMedico`(IN nombre VARCHAR(50), IN apellidos VARCHAR(100), IN
_dni CHAR(9), IN _fechaNac DATE, IN dir VARCHAR(250), IN municipio INT(10), IN cp
CHAR(5),IN numSS char(12),IN numColegiado CHAR(9),IN usuario VARCHAR(50),IN mail
VARCHAR(80),IN pass VARCHAR(18),IN trato VARCHAR(10))
BEGIN
DECLAREid INT;

       INSERTINTOt_personas
VALUES(NULL,nombre,apellidos,_dni,_fechaNac,dir,municipio,cp);
       SELECTidPersona FROMt_personas WHEREdni=_dni ANDfechaNac=_fechaNac INTOid;
       INSERTINTOt_empleados VALUES(id,numSS,now(),trato);
       INSERTINTOt_medicos VALUES(id,numColegiado,0);
       INSERTINTOt_usuarios VALUES(id,usuario,mail,sha1(pass),1,0);
       INSERTINTOr_roles VALUES(id,3);

END
```

Fig. insertaMédico

#### *8.2.2.3.2. Insertar un Auxiliar Administrativo*

El procedimiento tiene una estructura similar al anterior, pero en este caso se inserta en **t_auxiliaresAdministrativos** en lugar de en t_medicos. El rol en este caso será el 2.

```
CREATE PROCEDURE`insertaAuxAdmin`(IN nombre VARCHAR(50), IN apellidos VARCHAR(100),
IN _dni CHAR(9), IN _fechaNac DATE, IN dir VARCHAR(250), IN municipio INT(10), IN cp
CHAR(5),IN numSS char(12),IN salario FLOAT(6,2),IN usuario VARCHAR(50),IN mail
VARCHAR(80),IN pass VARCHAR(18),IN trato VARCHAR(10))
BEGIN
DECLAREid INT;

       INSERT INTOt_personas VALUES
(NULL,nombre,apellidos,_dni,_fechaNac,dir,municipio,cp);
       SELECT idPersona FROM t_personas WHERE dni=_dni AND fechaNac=_fechaNac INTO
id;
       INSERT INTOt_empleados VALUES(id,numSS,now(),trato);
       INSERT INTOt_auxiliaresAdministrativos VALUES(id,salario,0);
       INSERT INTOt_usuarios VALUES(id,usuario,mail,sha1(pass),1,0);
       INSERT INTOr_roles VALUES(id,2);

END
```

Fig. insertaAuxAdmin

#### 8.2.2.3.3. Insertar un ATS

De manera análoga a los anteriores solo que insertando en **t_ats** y asignándole el rol correspondiente que es el 4.

```
CREATE PROCEDURE`insertaAts`(IN nombre VARCHAR(50), IN apellidos VARCHAR(100), IN
_dni CHAR(9), IN _fechaNac DATE, IN dir VARCHAR(250), IN municipio INT(10), IN cp
CHAR(5),IN numSS char(12),IN especialidad VARCHAR(100),IN usuario VARCHAR(50),IN mail
VARCHAR(80),IN pass VARCHAR(18),IN trato VARCHAR(10))
BEGIN
DECLAREid INT;

	INSERTINTOt_personas
VALUES(NULL,nombre,apellidos,_dni,_fechaNac,dir,municipio,cp);
	SELECTidPersona FROMt_personas WHEREdni=_dni AND fechaNac=_fechaNac INTOid;
	INSERTINTOt_empleados VALUES(id,numSS,now(),trato);
	INSERTINTOt_ats VALUES(id,especialidad,0);
	INSERTINTOt_usuarios VALUES(id,usuario,mail,sha1(pass),1,0);
	INSERTINTOr_roles VALUES(id,4);

END
```

Fig. inseraAts

#### 8.2.2.3.4. Insertar un Paciente

Tendremos 2 posibles casos y, por tanto 2 posibles procedimientos.

A. Insertar un paciente que no sea empleado.

En este caso procederemos de manera equivalente a las anteriores. Insertaremos en las tablas **t_personas**, **t_pacientes**, **t_usuarios** y **r_roles**. El rol en este caso es el 5.

```
CREATEPROCEDURE`insertaPaciente`(IN nombre VARCHAR(50), IN apellidos
VARCHAR(100), IN _dni CHAR(9), IN _fechaNac DATE, IN dir VARCHAR(250), IN
municipio INT(10), IN cp CHAR(5),IN aseguradora TINYINT(3),IN usuario
VARCHAR(50),IN mail VARCHAR(80),IN pass VARCHAR(18))
BEGIN
DECLAREid INT;
DECLAREexisteP TINYINT(1);
DECLAREexisteU TINYINT(1);
SELECTcount(*) FROMt_personas WHEREdni=_dni ANDfechaNac=_fechaNac INTOexisteP;
SELECTcount(*) FROMt_usuarios WHEREuser=usuario INTOexisteU;

IFexisteP=0 ANDexisteU=0 THEN
	INSERTINTOt_personas
VALUES(NULL,nombre,apellidos,_dni,_fechaNac,dir,municipio,cp);
	SELECTidPersona FROMt_personas WHEREdni=_dni ANDfechaNac=_fechaNac
INTOid;
	INSERT INTOt_pacientes VALUES(id,now(),0,aseguradora);
	INSERT INTOt_usuarios VALUES(id,usuario,mail,sha1(pass),0,0);
	INSERT INTOr_roles VALUES(id,5);
END IF;
END
```

Fig. insertaPaciente no empleado

B. Insertar un paciente que ya era empleado.

Como ya tenemos asignado un registro en **t_personas** asociado a esta persona simplemente insertaremos en la tabla **t_pacientes** y en **r_roles** con rol igual a 5. Este procedimiento solo recibirá como parámetros el idPersona y la aseguradora del paciente.

```
CREATE PROCEDURE`insertaPacienteEmpleado`(IN id INT,IN aseguradora TINYINT(3))
BEGIN
INSERT INTOt_pacientes VALUES(id,now(),0,aseguradora);
INSERTINTOr_roles VALUES(id,5);
END
```

Fig. insertaPacienteEmpleado

#### 8.2.2.3.5. Rellenar turnos de Consultas

Recibe como parámetros el idMédico, el año, el mes, el día de la semana (del 1->domingo al 7->sábado) y el turno y se inserta (para todo el mes) en la tabla **r_horariosMedicos** una nueva entrada de **tipo='c'** es decir, de tipo consulta.

```
CREATE PROCEDURE`rellenaFechas`(IN medico INT,IN anno CHAR(4),IN mes CHAR(2),IN dia
INT,IN turno TINYINT(3))
BEGIN
DECLAREfecha DATE;
SETfecha=CONCAT(anno,'-',mes,'-01');
WHILEmonth(fecha)=mes DO
        IFdia=dayofweek(fecha) THEN
        INSERTINTOr_horariosMedicos VALUES(turno,medico,fecha,'c');
        ENDIF;
        SETfecha = adddate(fecha,1);
ENDWHILE;
END
```

Fig. rellenaFechas

#### 8.2.2.3.6. Rellenar turnos de Intervenciones

Este procedimiento es análogo al anterior pero con **tipo ='i'**.

```
CREATE PROCEDURE`rellenaFechasIntervenciones`(IN medico INT,IN anno CHAR(4),IN mes
CHAR(2),IN dia INT,IN turno TINYINT(3))
BEGIN
DECLAREfecha DATE;
SETfecha=CONCAT(anno,'-',mes,'-01');

WHILEmonth(fecha)=mes DO
        IFdia=dayofweek(fecha) THEN
        INSERTINTOr_horariosMedicos VALUES(turno,medico,fecha,'i');
        ENDIF;
        SETfecha = adddate(fecha,1);
ENDWHILE;
END
```

Fig. rellenaFechasIntervención

#### 8.2.2.3.7. Borrar turnos de Consultas

Dada una fecha, un médico y un turno borrará todos los registros de la tabla **r_horariosMedicos** de **tipo='c'** de ese mes.

```
CREATE PROCEDURE borraFechas(IN medico INT,IN fec DATE,IN turn TINYINT(3))
BEGIN
DECLAREfech DATE;
DECLAREdia INT;
DECLAREmes INT;
DECLAREanno INT;
SETmes = month(fec);
SETanno = year(fec);
SETfech=CONCAT(anno,'-',mes,'-01');
SETdia = dayofweek(fec);
WHILEmonth(fech)=mes DO
	IFdia=dayofweek(fech) THEN
	DELETEFROMr_horariosMedicos WHEREidTurno = turn ANDfecha = fech ANDidMedico =
medico ANDtipo = 'c';
	ENDIF;
	SETfech = adddate(fech,1);
ENDWHILE;
END
```

Fig. borraFechasConsultas

#### 8.2.2.3.8. Borrar turnos de Intervenciones

Al igual que el anterior, dada una fecha, un médico y un turno borrará todos los registros de la tabla **r_horariosMedicos** de **tipo='i'** de ese mes.

```
CREATE PROCEDURE borraFechasInt(IN medico INT,IN fec DATE,IN turn TINYINT(3))
BEGIN
DECLAREfech DATE;
DECLAREdia INT;
DECLAREmes INT;
DECLAREanno INT;
SETmes = month(fec);
SETanno = year(fec);
SETfech=CONCAT(anno,'-',mes,'-01');
SETdia = dayofweek(fec);
WHILEmonth(fech)=mes DO
	IFdia=dayofweek(fech) THEN
	DELETEFROMr_horariosMedicos WHEREidTurno = turn ANDfecha = fech ANDidMedico =
medico ANDtipo = 'i';
	ENDIF;
	SETfech = adddate(fech,1);
ENDWHILE;
END
```

Fig. borraFechasIntervenciones

#### 8.2.2.3.9. Pedir Cita

Tras comprobar que sea posible la solicitud de cita (el médico solicitado no tiene otra consulta el mismo día a la misma hora y el paciente que la solicita no tiene ninguna otra consulta para ese día a esa hora) insertará una nueva entrada en la tabla **t_citas**.

```
CREATE PROCEDURE`pideCita`(IN paciente INT,IN medico INT,IN _fecha DATE,IN horario
TINYINT(3),tipo VARCHAR(30))
BEGIN
DECLAREcuentaP INT;
DECLAREcuentaM INT;
SELECTcount(*) FROMt_citas WHEREidMedico=medico ANDidHorario=horario ANDfecha=_fecha
INTOcuentaM;
SELECTcount(*) FROMt_citas WHEREidPaciente=paciente ANDidHorario=horario
ANDfecha=_fecha INTOcuentaP;
IFcuentaM=0 ANDcuentaP=0 THEN
	INSERTINTOt_citas VALUES(NULL,paciente,medico,_fecha,horario,tipo);
ENDIF;
END
```

Fig. pideCita

#### *8.2.2.3.10. Apuntar en lista de Espera de Intervención*

Lo único que hace es guardar una nueva entrada en **t_listaEsperaOperacional**. Como el id es auto incrementable solo necesitaremos pasar como parámetros paciente, médico, observaciones y fecha (que habitualmente llamaremos con la función now()).

```
CREATE PROCEDURE`creaIntervencion`(IN idPaciente INT, IN _fecha DATE, IN
observaciones VARCHAR(255), IN idMedico INT)
BEGIN
	INSERTINTOt_listaEsperaOperacional
VALUES(NULL,idPaciente,_fecha,observaciones,idMedico,0);
END
```

Fig. creaIntervención

#### *8.2.2.3.11. Validar pruebas de Intervención*

Una vez el paciente entrega las pruebas del preoperatorio y el médico las valida se llamará a este procedimiento. Se actualiza el campo **entregado=1**en **t_listaEsperaOperacional**. Se recupera el id de dicha tabla y se inserta un registro en **t_listaEsperaHistorico** y en **t_preparados** con ese mismo id. El parámetro **fechaEnt** habitualmente será llamado con la función now().

```
CREATE PROCEDURE`entregaPruebas`(IN idInterv INT,IN fechaEnt DATE,IN fechaAsig DATE,
IN medico2 INT,
	IN idAseguradora TINYINT(3), IN idTurno TINYINT(3), IN hosp
ENUM('a','h','i'),IN rx TINYINT(1),IN ucii TINYINT(1),IN sangre TINYINT(1))
BEGIN
DECLAREexiste INT;
DECLAREidPac INT;
DECLAREfechaAp DATE;
DECLAREobs VARCHAR(255);
DECLAREmedico1 INT;
SELECTcount(*) FROMt_listaEsperaOperacional WHEREid=idInterv ANDentregado = 0
INTOexiste;
IFexiste != 0 THEN
	UPDATEt_listaEsperaOperacional SETentregado = 1 WHEREid=idInterv;
	SELECTidPaciente, fecha, observaciones, medico INTOidPac, fechaAp, obs,
medico1 FROMt_listaEsperaOperacional WHEREid=idInterv;
	INSERTINTOt_listaEsperaHistorico VALUES(idInterv,idPac, fechaAp, fechaEnt,
fechaAsig, medico1, medico2);
```

```
        -- AL haber entregado las pruebas se inserta en la tabla de preparados con los
-- detalles del preoperatorio.
              INSERTINTOt_preparados VALUES(idInterv,idPac, idAseguradora, medico2,
fechaEnt, fechaAsig, idTurno, hosp, rx, ucii, sangre);

END IF;
END
```

Fig. entregaPruebas

#### *8.2.2.3.12. Completar Intervención*

Cuando el paciente sea intervenido se llamará a este procedimiento que almacenará la intervención en **t_intervenciones**. Posteriormente se borrarán las entradas correspondientes en las tablas **t_listaEsperaOperacional** y **t_preparados**.

```
CREATE PROCEDURE`intervenido`(IN idInterv INT,IN fechaAsig DATE, IN observ
VARCHAR(255))
BEGIN
DECLAREexiste INT;
DECLAREidAseg TINYINT(3);
DECLAREidTur TINYINT(3);
DECLAREhosp ENUM('a','h','i');
DECLARErx TINYINT(1);
DECLAREucii TINYINT(1);
DECLAREsangre TINYINT(1);
SELECTcount(*) as cuentas FROMt_preparados WHEREidIntervencion=idInterv INTOexiste;
IFexiste != 0 THEN
        SELECTidAseguradora, idTurno, hospital, rayos, uci, sc INTO
                idAseg, idTur, hosp, rx, ucii, sangre FROMt_preparados
WHEREidIntervencion=idInterv;
        INSERTINTOt_intervenciones VALUES(idInterv, idAseg, fechaAsig, idTur, observ,
hosp, rx, ucii, sangre);
        DELETEFROMt_preparados WHEREidIntervencion = idInterv;
        DELETEFROMt_listaEsperaOperacional WHEREid = idInterv;
ENDIF;
END
```

Fig. intervenido

### 8.2.3. Normalización BD

Una vez tenemos el modelo relacional, comprobamos como efectivamente nuestra base de datos está normalizada.

- **Primera Forma Normal**: la cumple puesto que todos los atributos son atómicos, es decir no existen atributos multivalorados ni compuestos.
- **Segunda Forma Normal**: puesto que todos los atributos que no forman parte de la clave primaria depende funcional y completamente de cada clave, podemos afirmar que está en Segunda Forma Normal.
- **Tercera Forma Normal**: un esquema está en tercera forma normal si satisface la segunda forma normal y ninguno de los atributos que no forman parte de la clave primaria depende transitivamente de la clave primaria. En nuestro caso, como se cumple está condición en todas las tablas, podemos afirmar que nuestro diseño está en Tercera Forma Normal.

- **Forma Normal de Boyce-Codd**: puesto que en todas las dependencias funcionales no triviales de la forma X →Y, X es superclave, podemos asegurar que también está en Forma Normal de Boyce-Codd

### 8.2.4. MYSQL

MySQL es un sistema gestor de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Dado que los computadores son muy buenos manejando grandes cantidades de información, los gestores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones. MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL (GNU General Public License) para definir qué puede hacer y que no puede hacer con el software en diferentes situaciones.

MySQL es muy utilizado en aplicaciones Web como MediaWiki o Drupal, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación Web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones Web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Otra de las características de MySQL es su gran número de usuarios (cerca de seis millones), esto es debido principalmente a tres motivos: su licencia open source, por lo cual es un producto gratuito, su compatibilidad con casi todos de los lenguajes de programación actuales y por ser un producto multiplataforma, es decir, puede ser montada en cualquier versión de Linux, Windows o Mac.

#### *8.2.4.1. Características de la versión 5.5.8 de MySQL*

- Un amplio subconjunto de ANSI SQL 99, y varias extensiones.
- Soporte a multiplataforma
- Procedimientos almacenados
- Triggers
- Cursors
- Vistas actualizables
- Soporte a VARCHAR
- INFORMATION_SCHEMA

- Modo Strict
- Soporte X/Open XA de transacciones distribuidas; transacción en dos fases como parte de esto, utilizando el motor InnoDB de Oracle
- Motores de almacenamiento independientes (MyISAM para lecturas rápidas,
- InnoDB para transacciones e integridad referencial)
- Transacciones con los motores de almacenamiento InnoDB, BDB Y Cluster; puntos derecuperación(savepoints) con InnoDB
- Soporte para SSL
- Query caching
- Sub-SELECTs (o SELECTs anidados)
- Replication with one master per slave, many slaves per master, no automatic
- Support for multiple masters per slave.
- Indexing y buscando campos de texto completos usando el motor de almacenamiento MyISAM
- Embedded database library
- Soporte completo para Unicode
- Conforme a las reglas ACID usando los motores InnoDB, BDB y Cluster
- Shared-nothing clustering through MySQL Cluster

#### *8.2.4.2. Características adicionales*

- Usa GNU Automake, Autoconf, y Libtool para portabilidad
- Uso de multihilos mediante hilos del kernel.
- Usa tablas en disco b-tree para búsquedas rápidas con compresión de índice
- Tablas hash en memoria temporales· Tablas hash en memoria temporales
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL Completo soporte para operadores y funciones en cláusulas select y where. Completo soporte para cláusulas group by y order by, soporte de funciones de agrupación
- Seguridad: ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.
- Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).107
- Los clientes se conectan al servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows se pueden conectar usando named pipes y en sistemas Unix usando ficheros socket Unix.
- En MySQL 5.0, los clientes y servidores Windows se pueden conectar usando memoria compartida.
- MySQL contiene su propio paquete de pruebas de rendimiento proporcionado con el código fuente de la distribución de MySQL.

### 8.2.4.3. MYSQL Workbench

MySQL Workbench es una herramienta visual y multiplataforma para el diseño de bases de datos desarrollada por MySQL. MySQL Workbench. Está disponible como una herramienta GUI para Windows, Linux, y OS X.

Esta herramienta ahorra bastante tiempo a la hora de diseñar bases de datos en MySQL, y, una vez finalizado el diseño, se puede obtener el código del Lenguaje de Definición de Datos en formato sql para poder usarla de inmediato.

### 8.2.4.4. PhpMyAdmin 3.3.9

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas webs, utilizando Internet. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 50 idiomas. Se encuentra disponible bajo la licencia GPL.

Este proyecto se encuentra vigente desde el año 1998, siendo el mejor evaluado en la comunidad de descargas de SourceForge.net como la descarga del mes de diciembre del 2002. Como esta herramienta corre en máquinas con Servidores Webs y Soporte de PHP y MySQL, la tecnología utilizada ha ido variando durante su desarrollo.

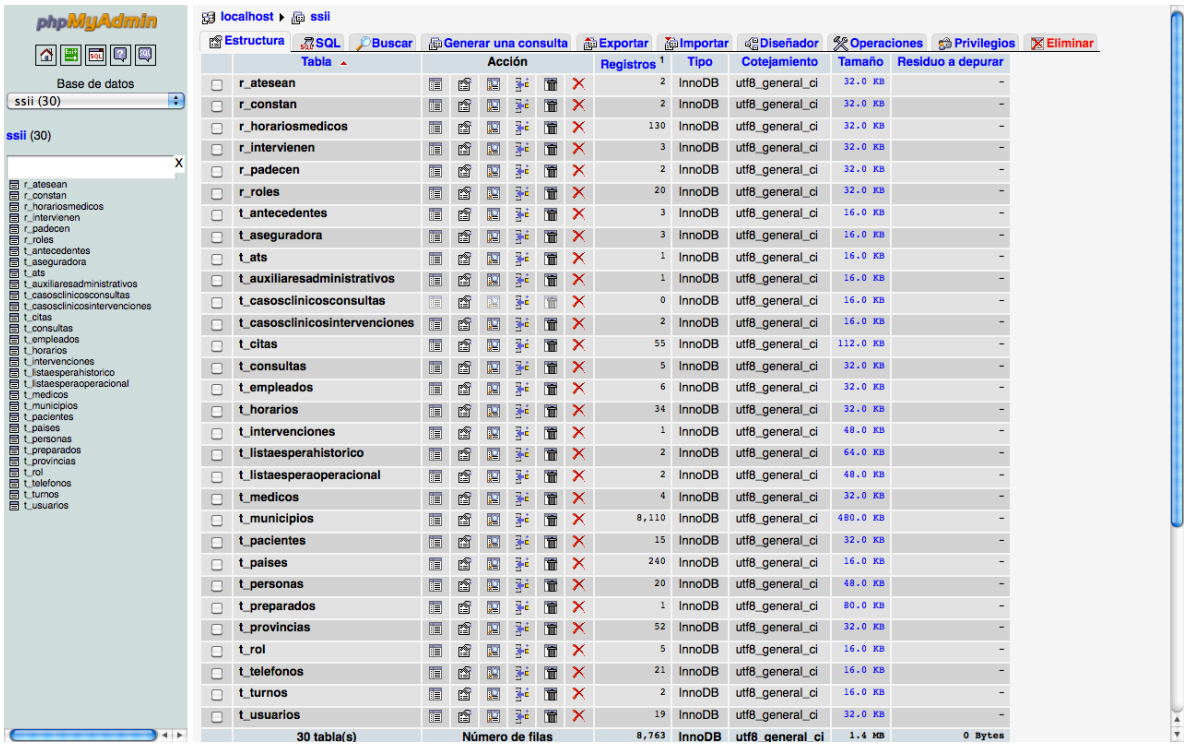



Fig 8.2.4.4.1 Herramiento web de administración de MySQL

## 8.3.APLICACIÓN WEB

La estructura de la aplicación consta de dos páginas claramente diferenciadas por su URL.

- Página de registro http://consulta2011.sytes.net/web/registro.php
- Página principal http://consulta2011.sytes.net/web/index.php

El resto de contenidos se cargarán de forma dinámica mediante AJAX, por lo que la navegación en la página principal se hará desde la misma URL siendo todos los parámetros de navegación ocultos para el usuario (POST). Esto mejorará la seguridad del sistema.

La estructura de archivos de la aplicación Web está dividida en carpetas:

- **css/** carpeta que contiene todas la hojas de estilos (*.css) de la aplicación.
- **img/** carpeta que contiene las imágenes de la aplicación.
- **includes/** carpeta que contiene los diferentes módulos escritos en PHP que se 'incluyen' en la web de forma estática (antes de que la página sea cargada).
- **js/** carpeta que contiene los scripts y las librerías de JavaScript (jQuery).
- **lib/** carpeta que contiene las librerías PHP de la aplicación. En nuestro caso solo contiene la librería Swift Mailer.
- **src/** carpeta que contiene todos los archivos PHP de los que hace uso la aplicación de forma dinámica (AJAX).

La aplicación Web está escrita en **XHTML 1.1** siguiendo los estándares de código que marca la World Wide Web Consortium(***W3C***).

Además de este lenguaje de marcado, haremos uso de otros lenguajes de programación. En este punto habrá que distinguir entre los lenguajes del lado del servidor (**PHP**) y lenguajes del lado del cliente (JavaScript →**jQuery** en nuestro caso).

### 8.3.1. XHTML

**XHTML**, acrónimo en inglés de *e**X**tensible **H**yper**t**ext **M**arkup **L**anguage* (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. En su versión 1.0, XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del *WorldWideWebConsortium* de lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas. La versión 1.1 es similar, pero parte a la especificación en módulos. En sucesivas versiones la W3C planea romper con los tags clásicos traídos de HTML.

### 8.3.2. PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor

(server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+, también puede ser embebido dentro de código HTML/XHTML.

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdof en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber unaespecificación formal. Publicado bajo la PHP License, la Free Software Foundationconsidera esta licencia como software libre.

PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, aunque el número de sitios en PHP ha declinado desde agosto de 2005. Es también el módulo Apache más popular entre las computadoras que utilizan Apache como servidor web.La versión más reciente de PHP es la 5.3.6 (17 de marzo de 2011).

#### *8.3.2.1. Características de PHP 5.3.5*

El 13 de julio de 2004, fue lanzado PHP 5, utilizando el motor Zend Engine 2.0 (o Zend Engine 2). La versión más reciente de PHP es la 5.3.6 (17 de marzo de 2011), que incluye todas las ventajas que provee el nuevo Zend Engine 2 como:

- Mejor soporte para la Programación Orientada a Objetos, que en versiones anteriores era extremadamente rudimentario, con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.
- Mejor soporte a XML ( XPath, DOM, etc. ).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Manejo de excepciones.
- Mejoras con la implementación con Oracle.

Aún se siguen publicando versiones de la rama 5.2.X, siendo publicada la versión 5.2.14 el 22 de julio de 2010, aunque la mayoría son actualizaciones de seguridad

Está previsto el lanzamiento en breve de la rama 6 de PHP. Cuando se lance esta nueva versión quedarán solo dos ramas activas en desarrollo (PHP 5 y 6), pues se abandonó el desarrollo y soporte de PHP 4 el 13 de Julio de 2007.

#### *8.3.2.2. Extensión Mysqli.*

La extensión ***mysqli***, o como a veces se le conoce, la extensión de MySQL *mejorada*, se desarrolló para aprovechar las nuevas funcionalidades encontradas en los sistemas MySQL con versión 4.1.3 o posterior. La extensión *mysqli* viene incluida en las versiones PHP 5 y posteriores.

La extensión *mysqli* contiene numerosos beneficios, siendo estas las mejoras principales respecto a la extensión *mysql*:

- Interfaz orientada a objetos
- Soporte para Declaraciones Preparadas (*Prepared Statements*)
- Soporte para Múltiples Declaraciones
- Soporte para Transacciones
- Mejoradas las opciones de depuración
- Soporte para servidor empotrado

Además de la interfaz orientada a objetos, esta extensión también proporciona una interfaz procedural.

La extensión *mysqli* está desarrollada mediante el framework de extensiones de PHP. Su código fuente se ubica en el directorio *ext/mysqli*.

Haremos uso de esa extensión para conectar con la BD utilizando la interfaz orientada a objetos.

#### *8.3.2.3. Swift Mailer +SMTP GMail.*

**SwiftMailer** es una librería escrita en PHP 5 que nos permite configurar y gestionar el envío de correos electrónicos mediante nuestras aplicaciones PHP. Ofrece una interfaz orientada a objetos para facilitar su configuración. Soporta envíos mediante SMTP, sendmail, postfix o cualquier método personalizado de transporte que implementemos.

Algunas de **sus principales características son:**

- conectividad persistente para mejorar el rendimiento;
- **soporte para internacionalización** (i18n);
- carga balanceada;
- **SSL & TLS Support – para servidores Gmail**;
- permite incluir imágenes y otros tipos de archivos;
- soporte total para MIME 1.0;
- procesos por lotes;
- **envía adjuntos de cualquier tamañano** incluso con el límite de 8MB en la memoria de PHP;
- soporte para múltiples adjuntos;
- permite establecer la prioridad del mensaje;
- monitor del ancho de banda incluído;

Utilizaremos la conexión mediante **SMTP** (Simple Mail Transfer Protocol), quenos facilita el envío de mails desde GMail sin hacer uso de un servidor de correo independiente como pudiera ser Mercury/32 del que disponíamos en nuestro servidor.

### 8.3.3. jQuery

**jQuery** es una biblioteca o framework de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC. La versión actual es la 1.6.1 (12 de mayo de 2011).

**jQuery** es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos.

jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

#### *8.3.3.1. AJAX*

**Ajax**, acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante *XMLHttpRequest*, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

Haremos uso de esta técnica mediante jQuery utilizando las funciones que nos brindan esta posibilidad como son **$.post()** y **$.get().** Usaremos una u otra dependiendo de si los parámetros viajan vía 'POST' o vía 'GET'. Será la forma de conectar el código jQuery con el código del servidor de PHP.

#### *8.3.3.2. jQuery UI*

jQuery UI es un complemento de jQuery que permite implementar componentes diversos para generar interfaces de usuario en páginas web, además de otras funcionalidades básicas para crear aplicaciones web enriquecidas. Como su propio nombre indica, está basado en el popular

framework Javascript y podemos encontrar links, explicaciones, así como demos y descargas a partir del sitio web oficial de jQuery.

Su uso ha sido de especial importancia a la hora de utilizar sus widgets (pestañas, acordeón, etc.) y sobre todo a la hora de incluir selectores de fechas.

También nos ha simplificado el CSS de la web, ya que te permite utilizar temas personalizados para dichos widgets, asi como dar formato a los botones y otros componentes del DOM.

#### *8.3.3.3. Plugin Validation*

Es un plugin que nos permite validar los formularios de la web desde el lado del cliente de una forma sencilla. Permite opciones de personalización y ampliación. En nuestro caso hemos añadido un validador de contraseñas que obligará al usuario a registrarse con contraseñas de 8 o más caracteres siendo obligatoria la presencia de letras y números. También hemos añadido la función para verificar que el DNI sea correcto, a fin de evitar SPAM o registros masivos.

Este plugin fue desarrollado por Jörn Zaefferer, miembro del equipo de jQuery y ha sido constantemente revisado a medida que evolucionaba jQuery. La versión actual es la 1.8.0 (marzo 2011).

## 8.4.APLICACIÓN JAVA

Para la aplicación Java hemos utilizado las siguientes tecnologías:

### 8.4.1. J2EE

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de Programación (parte de la plataforma Java) para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; no obstante sin un estándar de ISO o ECMA.
Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Esto permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable

con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

Para el diseño de las GUI se han utilizado las librerías java.awt y javax.swing. La primera de ellas se adhiere al sistema operativo donde se ejecute utilizando sus componentes y aspecto gráfico. Con la segunda se pueden modificar los temas(LooksAndFeel) consiguiendo personalizar nuestra GUI. Con la mezcla de ambos se mantiene el aspecto del sistema nativo con la personalización de nuestra aplicación.

### 8.4.2. J2SE

Java Platform, Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la plataforma Java. La plataforma Java 2, Enterprise Edition incluye todas las clases en el Java SE, además de algunas de las cuales son útiles para programas que se ejecutan en servidores sobre workstations.

### 8.4.3. JAVA.AWT

La Abstract Window Toolkit contiene rutinas para soportar operaciones básicas GUI y utiliza ventanas básicas desde el sistema nativo subyacente.

La idea original de la biblioteca AWT de utilizar componentes nativos, con lo que adopta un estilo más consistente en todas las plataformas. La biblioteca Swing, por otro lado, está codificada enteramente en Java y frecuentemente se la acusa de no brindar una experiencia idéntica a la de una aplicación nativa. Sin embargo, el precio a pagar por esa mejora es la dependencia (anivel de aspecto visual y no de interfaz de programación) de la aplicación resultante del  sistema operativo sobre el cual se ejecuta.

### 8.4.4. JAVAX.SWING

Swing es una colección de rutinas que se construyen sobre java.awt para suministrar un toolkit de widgets independiente de la plataforma. Swing usa las rutinas de dibujado 2D para renderizar los componentes de interfaz de usuario en lugar de confiar en el soporte GUI nativo subyacente del Sistema operativo.

Swing es un sistema muy rico por sí mismo, soportando pluggable looks and feels (PLAFs) para que los controles (widgets) en la GUI puedan imitar a aquellos del sistema nativo subyacente. Los patrones de diseño impregnan el sistema, especialmente una modificación del patrón modelo-vista-controlador, el cual afloja el acoplamiento entre función y apariencia. Una

inconsistencia es que (para J2SE 1.3) las fuentes son dibujadas por el sistema nativo subyacente, limitando la portabilidad de texto. Mejoras, tales como usar fuentes de mapas de bits, existen. En general, las layouts(disposiciones de elementos) se usan y mantienen los elementos dentro de una GUI consistente a través de distintas plataformas.

### 8.4.5. SPRING FRAMEWORK

Con el fin de trabajar con datos de una base de datos, es necesario obtener una conexión a la base de datos. Un DataSource es parte de la especificación JDBC y puede ser visto como una fábrica de conexiones generalizada. Permite que un contenedor o un marco pueda ocultar la agrupación de conexiones y las cuestiones de gestión de transacciones del código de la aplicación. Como desarrollador, no se necesita saber más detalles acerca de cómo conectarse a la base de datos, lo cuál es responsabilidad del administrador que establece el data source. Lo más probable es que el programador cumpla ambas funciones al mismo tiempo: desarrollando y testeando el código. Pero no necesariamente tiene que saber comoestá configurado el data source.

Spring nos sirve para simplificar diversas funciones de los proyectos que podemos crear en la plataforma Eclipse o NetBeans, de tipo Maven, WebFlow, Java y una larga lista.

En nuestro caso queremos conectarnos a una base de datos MySql a través del driver jdbc.

Para ello debemos crear un archivo de configuración en XML "**Bean.xml**" en el que están las librerías necesarias y los parámetros de conexión tanto el driver a utilizar, la ruta de la base de datos, usuario y contraseña entre otros parámetros.

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:property-placeholder location="jdbc.properties"/>

        <beanid="dataSource"class="org.apache.commons.dbcp.BasicDataSource">
```

```
        <property name="driverClassName"value="${driverClassName}"/>
        <propertyname="url"value="${url}"/>
        <property name="username"value="${username}"/>
        <property name="password"value="${password}"/>
    </bean>
    <bean id="jdbcTemplate"class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource"ref="dataSource"/>
    </bean>
</beans>
```

Fig. "Bean.xml"

En otro archivo "**jdbc.properties**" alternativo pasamos por referencia los parámetros de conexión definidos para nuestro caso

```
driverClassName=com.mysql.jdbc.Driver
url=jdbc:mysql://consulta2011.sytes.net:3306/ssii
username=user
password=contraseña
```

Fig. "jdbc.properties"

Una vez tenemos nuestros datos de conexión configurados sólo tenemos que crear un DataSource e instanciar a nuestra "Bean" creada para la conexión.

```
Public class Dao implements DaoDataSource {

    privateJdbcTemplatejdbc;
    publicDao() {
```

```
        ApplicationContext context= newClassPathXmlApplicationContext("Bean.xml");

        jdbc = context.getBean(JdbcTemplate.class);

        }
}
```

Fig. Dao implements DaoDataSource

Ahora en nuestra variable del tipo JdbcTemplate "jdbc" tenemos cargada la conexión y podemos usar las distintos procesos que nos permite los jdbctemplate a través del framework de Spring.

**Uso**

Existen distintos tipo de procesos:

Consultas: Para todo tipo que pueda devolver la cláusula SELECT

queryForObject (devuelve un objetodel tipo que declaremos en la clase mapRow)

```
public Tpaciente damePaciente(int id) {

Tpaciente p = new Tpaciente();

    String sql;

    String t1 = "";

    String t2 = "";

try {

sql = "SELECT p.IdPersona, p.nombre, p.apellidos, p.dni, p.fechaNac, p.direccion,
p.cp,m.municipio, pr.provincia,pa.pais, a.nombre, pac.fechaAlta ";

sql = sql + "FROM t_personas p,t_municipios m,t_provincias pr,t_paises pa, t_pacientes pac,
t_aseguradora a ";

sql = sql + "WHERE idPersona= ? And idPaciente = ?";

sql = sql + "AND p.idPersona=pac.idPaciente AND a.idAseguradora = pac.idAseguradora ";

        sql = sql + "AND p.idMunicipio=m.idMunicipio AND pr.idProvincia=m.idProvincia AND
pa.idPais=pr.Pais ";
```

```
        sql = sql + "LIMIT 1; ";
p = (Tpaciente) jdbc.queryForObject(sql, new RowMapper() {


public Tpaciente mapRow(ResultSet rs, int rowNum) throws SQLException {
Tpaciente paciente = new Tpaciente();
                paciente.setId(rs.getInt(1));
                paciente.setNombre(rs.getString(2));
                paciente.setApellidos(rs.getString(3));
                paciente.setDni(rs.getString(4));
                paciente.setFechaNacimiento(rs.getString(5));
                paciente.setDireccion(rs.getString(6));
                paciente.setCp(rs.getInt(7));
                paciente.setMunicipio(rs.getString(8));
                paciente.setProvincia(rs.getString(9));
                paciente.setPais(rs.getString(10));
                paciente.setAseguradora(rs.getString(11));
                paciente.setFechaAlta(rs.getString(12));
return paciente;
            }
        }, new Object[]{id, id});
      } catch (Exception e) {
System.out.println(e.getMessage());
}
return p;    }
```

Fig. queryForObject

queryForInt (devuelve un entero)

```
sql = "SELECT count(*) as cuenta ";
sql = sql + "FROM r_padecen ";
sql = sql + "WHERE idPaciente = ? AND idAntecedente= ?";
existe = jdbc.queryForInt(sql, new Object[]{p.getId(), a.getId()});
```

fig. queryForInt

query (devuelve una Lista de objetos del tipo que declaremos en la clase mapRow)

```
public Object[][] damePacientes() {
    List lista = null;
Object[][] list = null;
    String sql;
try {
        sql = "SELECT p.IdPersona,p.apellidos, p.nombre, p.dni ";
        sql = sql + "FROM t_personas p, t_pacientes pac ";
        sql = sql + "WHERE p.idPersona = pac.idPaciente ";
        sql = sql + "AND pac.borrado = 0";
lista = jdbc.query(sql, new RowMapper() {

public Tpersona mapRow(ResultSet rs, int rowNum) throws SQLException {
Tpersona paciente = new Tpaciente();
            paciente.setId(rs.getInt(1));
            paciente.setApellidos(rs.getString(2));
```

```
                paciente.setNombre(rs.getString(3));
paciente.setDni(rs.getString(4));
return paciente;
            }
        });
int i = 0;
list = new Object[lista.size()][4];
while (!lista.isEmpty()) {

list[i][0] = ((Tpersona) lista.get(0)).getId();
list[i][1] = ((Tpersona) lista.get(0)).getNombre();
list[i][2] = ((Tpersona) lista.get(0)).getApellidos();
list[i][3] = ((Tpersona) lista.get(0)).getDni();
lista.remove(0);
i++;
        }

    } catch (Exception e) {
System.out.println(e.getMessage());
}
    return list;
  }
```

Fig. query

Ejecución: Sirve para ejecutar ordenes del tipo INSERT INTO, UPDATE o DELETE.

**-INSERT INTO**

//Guardamos los nuevos tlf2

```
        sql = "INSERT INTO t_telefonos (telefono, idPersona) VALUES ( ? , ? )";
vars = new Object[]{pac.getTlf2(), pac.getId()};
jdbc.update(sql, vars);
```

Fig. Insert into

**-UPDATE**

```
public Boolean borrarAts(int id) {
      String sql;
      Boolean error = false;
try {
sql = "UPDATE t_ats SET borrado = 1 WHERE idAts = ?";
jdbc.update(sql, new Object[]{new Integer(id)});
borraUsuario(id);
      } catch (Exception e) {
System.out.println(e.getMessage());
error = true;
      }
return !error;
   }
```

Fig. Update

**- DELETE**

```
public Boolean borrarCita(int id) {
      String sql;
      Boolean error = false;
try {
```

```
sql = "DELETE FROM t_citas WHERE idCita = ?";

jdbc.update(sql, new Object[]{new Integer(id)});

      } catch (Exception e) {

System.out.println("borrarCita Exception: " + e.getMessage());

error = true;

      }      return !error;    }
```

Fig. Delete

### 8.4.6. JavaMail

JavaMail es una expansión de Java que facilita el envío y recepción de e-mail desde código java.

Características:

JavaMail implementa el protocolo SMTP (Simple Mail Transfer Protocol) así como los distintos tipos de conexión con servidores de correo -TLS, SSL, autentificación con usuario y password, etc.

JavaMail no se incluye en la JDK ni en la JRE, sino que debe conseguirse como un paquete externo. Debe, además, descargarse adicionalmente elJavaBeans Activation Framework en caso de usar una JDK inferior a la versión 6.

Clases y configuración de JavaMail para el envío de mensajes

Para este ejemplo sencillo, hay tres clases de JavaMail implicadas:

**Session**: De alguna forma representa la conexión con el servidor gmail de correo. Hay que obtenerla pasándole los parámetros de configuración para el servidor de correo -gmail en nuestro caso-.

**Transport**: Clase para el envío de mensajes. Se obtiene llamando al método **getTransport()** de la clase Session.

**MimeMessage**: El mensaje.

### 8.4.7. Diagramas UML

Esquema de la composición de la aplicación de Java. Se subdivide en dos capas: vista ( GUI) y procesos (src).

La capa vista consta de las pestañas principales y de un sub-paquete con las interfaces gráficas de las ventanas emergentes.

La capa procesos se subdivide en dos diferenciadas por ofrecer la funcionalidad de la aplicación y la segunda por proveer las acciones sobre la base de datos. La primera compuesta del paquete actores y casos. Y la segunda es el paquete procesos.

#### *8.4.7.1. Diagrama Clases GCT*

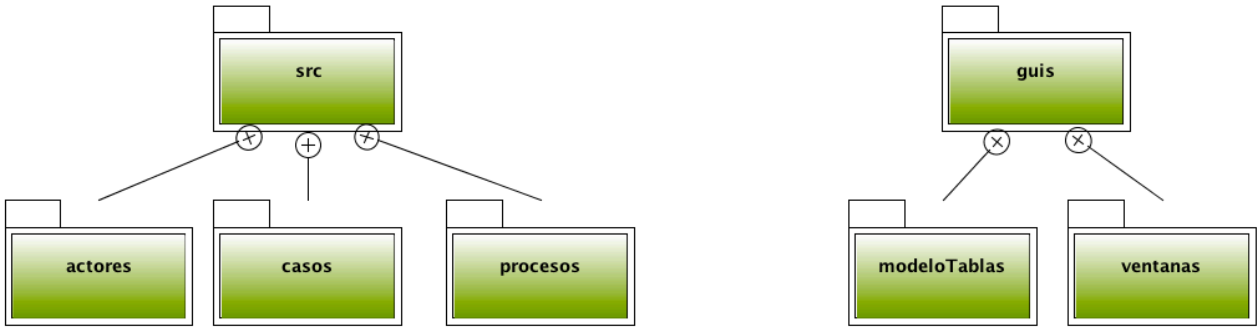



Fig. 8.4.7.1. Diagrama Clases GCT

La capa vista consta de las pestañas principales y de un sub-paquete con las interfaces gráficas de las ventanas emergentes.

#### 8.4.7.1.1. Diagrama Clases GUIS

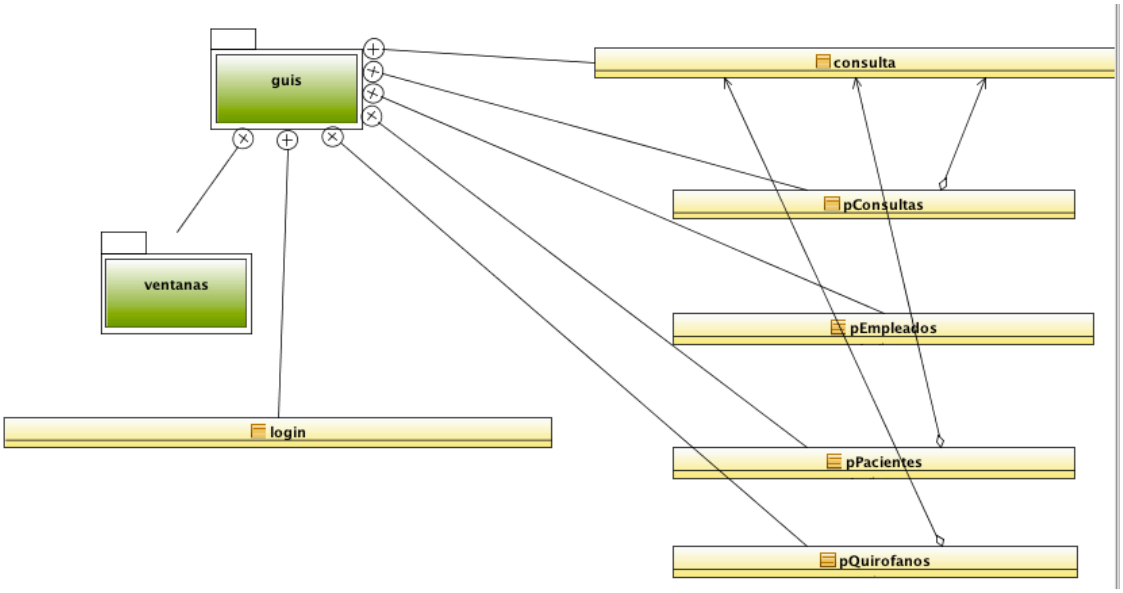



Fig. 8.4.7.1.1. Diagrama Clases GUIS

#### 8.4.7.1.2. Clase consulta

Esta clase implementa la ventana principal de la aplicación con todos sus componentes y acciones.

private JScrollPane jScrollPane11
private JScrollPane jScrollPane12
private JScrollPane jScrollPane13
private JScrollPane jScrollPane14
private JScrollPane jScrollPane15
private JScrollPane jScrollPane7
private JScrollPane jScrollPane8
private JScrollPane jScrollPane9
private Separator jSeparator1
private JTabbedPane jTabbedPane1
private JTabbedPane jTabbedPane2
private JTabbedPane jTabbedPane3
private JMenuBar menuBar

*Operations*

public consulta( )
public consulta( DaoDataSource dao )
public Image getIconImage( )
private void cargaModelosTablas( )
public DaoDataSource getDao( )
public void setDao( DaoDataSource dao )
public Tpersona getUser( )
public void setUser( Tpersona user )
public void setPermisos( )
private void bloqueaEmpleados( )
private void initComponents( )
private void jMenuexitActionPerformed( ActionEvent evt )
private void jConsProgramarInterActionPerformed( ActionEvent evt )
private void jConsModifDatosPacienteActionPerformed( ActionEvent evt )
private void jConsNuevaConsActionPerformed( ActionEvent evt )
private void jConsMedicoActionPerformed( ActionEvent evt )
private void jPacApellido3ActionPerformed( ActionEvent evt )
private void jQuiroBotonListaEsperaActionPerformed( ActionEvent evt )
private void jQuiroBotonEliminarIntervencionActionPerformed( ActionEvent evt )
private void jQuiroBotonEditarIntervencionActionPerformed( ActionEvent evt )
private void jQuiroFechaActionPerformed( ActionEvent evt )
private void jEmplNuevoEmplActionPerformed( ActionEvent evt )
private void jEmplModEmpleadoActionPerformed( ActionEvent evt )
private void jEmplEliminarEmpleadoActionPerformed( ActionEvent evt )
private void jConsListadoMaÒanaMouseClicked( MouseEvent evt )
private void jEmplListMedicosMouseClicked( MouseEvent evt )
private void jConsListadoTardeMouseClicked( MouseEvent evt )
private void jEmplListInstrumentistasMouseClicked( MouseEvent evt )
private void jEmplListAdministrativosMouseClicked( MouseEvent evt )
private void formComponentShown( ComponentEvent evt )
public void hideCargando( )
public void showCargando( )
private void jConsultasComponentShown( ComponentEvent evt )
private void jPacNuevoPacActionPerformed( ActionEvent evt )
private void jPacModPacActionPerformed( ActionEvent evt )

```
private JPanel jEmpleados
private JTable jIntListadoMaÒana
private JTable jIntListadoTarde
private JLabel jLabel1
private JLabel jLabel16
private JLabel jLabel36
private JLabel jLabel39
private JLabel jLabel40
private JMenuItem jMenuCambiarPass
private JMenuItem jMenuCerrarSesion
private JMenuItem jMenuItem1
private JMenuItem jMenuReconectar
private JMenuItem jMenuexit
private JPanel jPacAcciones
private JTextField jPacApellido3
private JButton jPacBotonBuscar
private JTextField jPacCampoBusqueda
private JButton jPacDarCita
private JTabbedPane jPacDer
private JPanel jPacIzq
private JTable jPacListadoPacientes
private JButton jPacModPac
private JButton jPacNuevoPac
private JRadioButton jPacOpcBusqDni
private JRadioButton jPacOpcBusqNombre
private JPanel jPacientes
private JPanel jPacientes1
private JPanel jPanel10
private JPanel jPanel11
private JPanel jPanel12
private JPanel jPanel13
private JPanel jPanel5
private JPanel jPanel7
private JPanel jPanel8
private JTabbedPane jPanelPrincipal
private jPopupMenu1
private jPopupMenu2
private jPopupMenu3
private jPopupMenu4
private JPanel jQuiroAcciones
private JButton jQuiroBotonEditarIntervencion
private JButton jQuiroBotonEliminarIntervencion
private JButton jQuiroBotonListaEspera
private JTextField jQuiroFecha
private JPanel jQuiroIzq
private JTabbedPane jQuirofDer
private JPanel jQuirofanos
```

```
private void jConsultasComponentShown( ComponentEvent evt )
private void jPacNuevoPacActionPerformed( ActionEvent evt )
private void jPacModPacActionPerformed( ActionEvent evt )
private void jPacDarCitaActionPerformed( ActionEvent evt )
private void jPacCampoBusquedaActionPerformed( ActionEvent evt )
private void jPacOpcBusqNombreActionPerformed( ActionEvent evt )
private void jEmplAsignaHorarioActionPerformed( ActionEvent evt )
private void jConsBotonAddAntecedActionPerformed( ActionEvent evt )
private void jEmplAsignaHorario2ActionPerformed( ActionEvent evt )
private void jIntListadoTardeMouseClicked( MouseEvent evt )
private void jIntListadoMaÒanaMouseClicked( MouseEvent evt )
private void jMenuItem1ActionPerformed( ActionEvent evt )
private void jPacListadoPacientesMouseClicked( MouseEvent evt )
private void jPacListadoPacientesMousePressed( MouseEvent evt )
private void jMenuCerrarSesionActionPerformed( ActionEvent evt )
private void jMenuCambiarPassActionPerformed( ActionEvent evt )
private void jPacBotonBuscarActionPerformed( ActionEvent evt )
private void jMenuReconectarActionPerformed( ActionEvent evt )
private void itemModificarActionPerformed( ActionEvent evt )
private void aboutMenuItemActionPerformed( ActionEvent evt )
private void itemModificar1ActionPerformed( ActionEvent evt )
private void itemModificar2ActionPerformed( ActionEvent evt )
private void itemModificar3ActionPerformed( ActionEvent evt )
private void selectFecha( ComponentEvent evt )
private void selectFechaQuiro( ComponentEvent evt )
private void cargaFormatosFecha( )
private String dameFechaSQL( )
private String dameFechaEU( )
private void muestraSelectorFecha( )
private void cargaActions( )
public void cargaDatos( )
public void cargaConsultas( )
private void cargaIntervenciones( )
private void cargaListaMedicos( )
public void cargaEmpleados( )
private void cargaPacientes( )
private void cargaMedicos( )
private void cargaAts( )
private void cargaAuxiliares( )
private void cargaConsultasMaÒana( )
private void cargaConsultasTarde( )
private void cargaIntervencionesMaÒana( )
private void cargaIntervencionesTarde( )
private void ocultaColIntervenciones( )
private void ocultaColPacientes( )
private void ocultaColEmpleados( )
private void ocultaColConsultas( )
private void busquedaPacienteDNI( )
private void busquedaPacienteNombre( )
```


Fig. 8.4.7.1.2. Clase consulta

#### *8.4.7.1.3. Clase login*

Esta clase se utiliza para mostrar el formulario inicial solicitando usuario y contraseña para acceder a la aplicación.


Fig. 8.4.7.1.2. Clase consulta

#### 8.4.7.1.4. Clase pConsultas

Esta clase contiene los atributos y métodos para la pestaña del panel principal que muestra información de consultas.

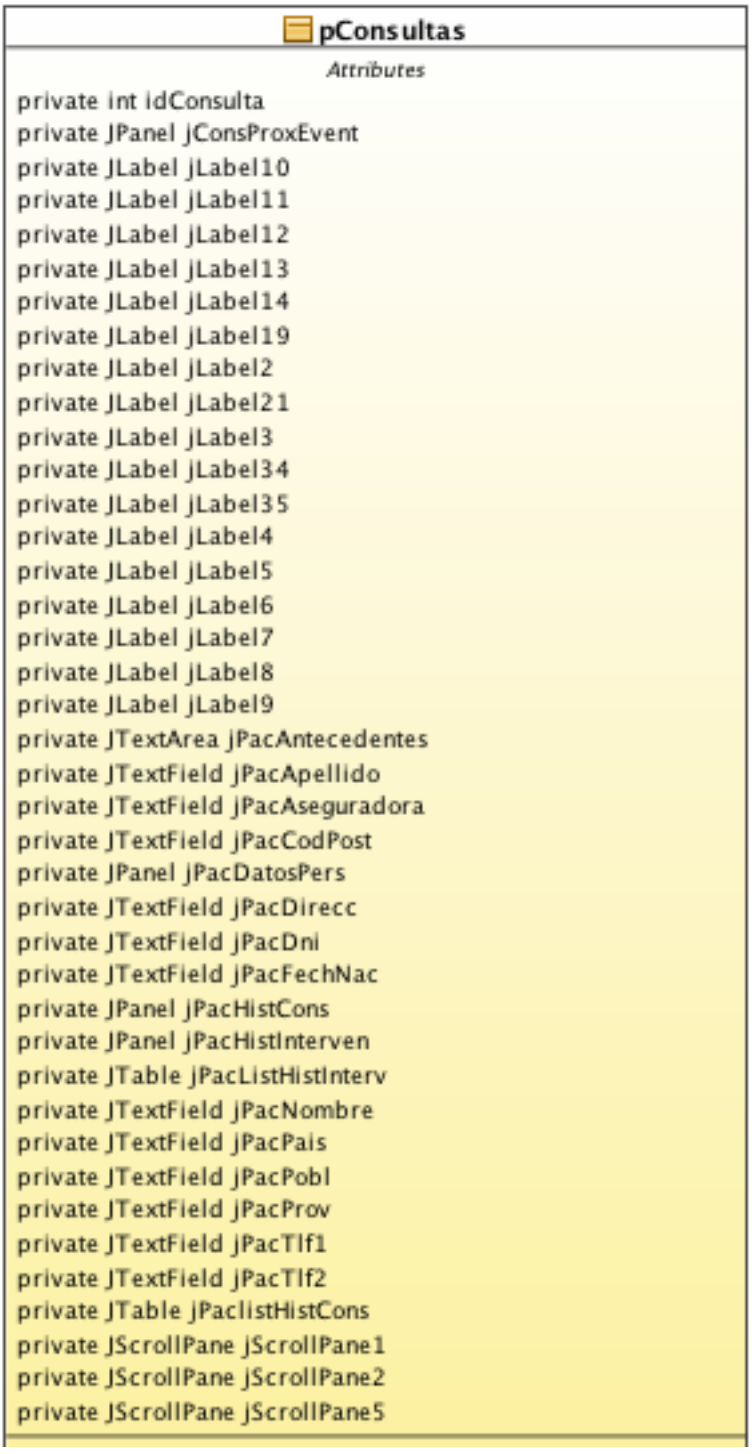



Fig . 8.4.7.1.4. Clase pConsultas

#### *8.4.7.1.5. Clase pEmpleados*

Esta clase contiene los atributos y métodos para la pestaña del panel principal que muestra información de empleados.

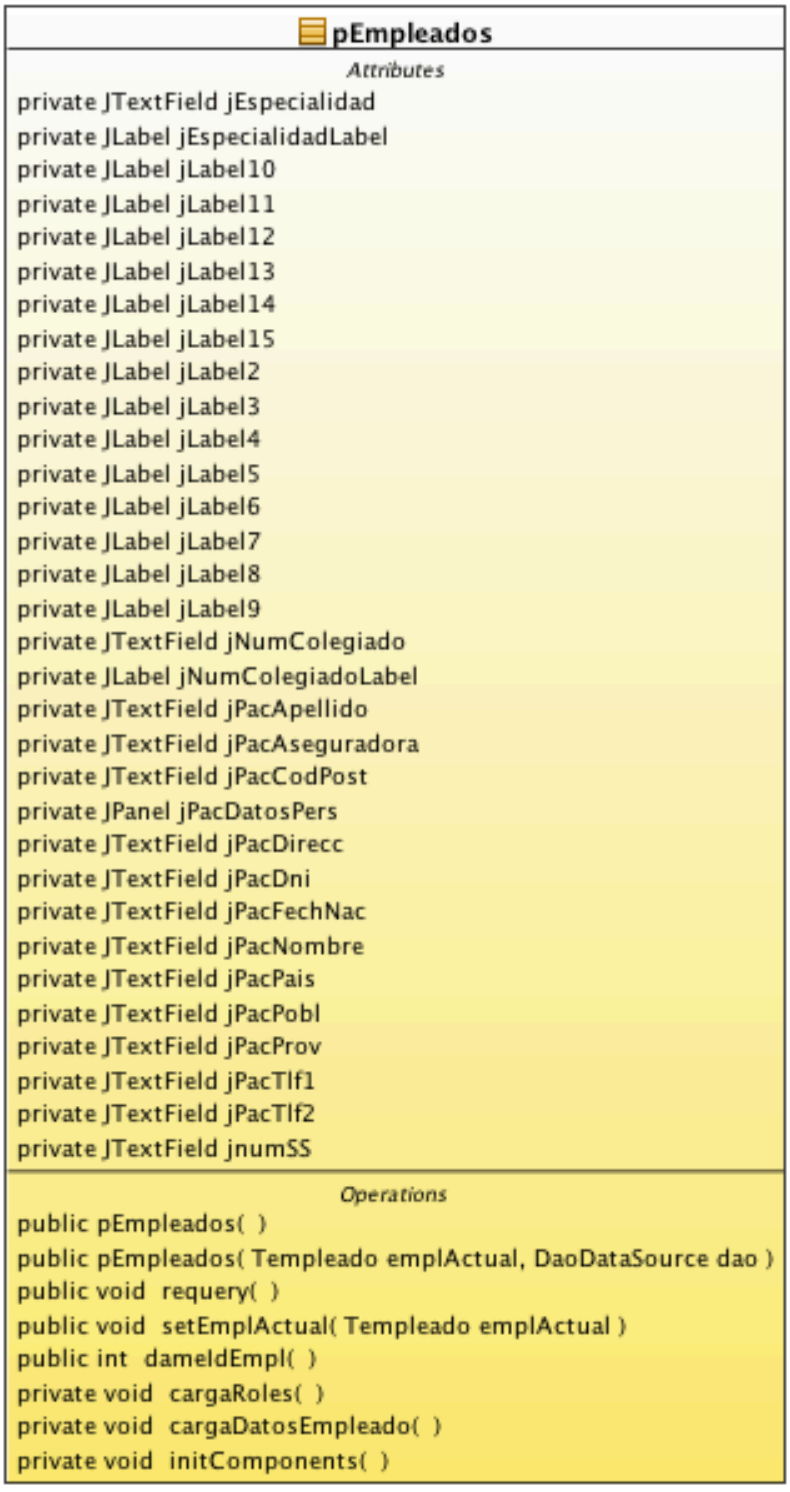
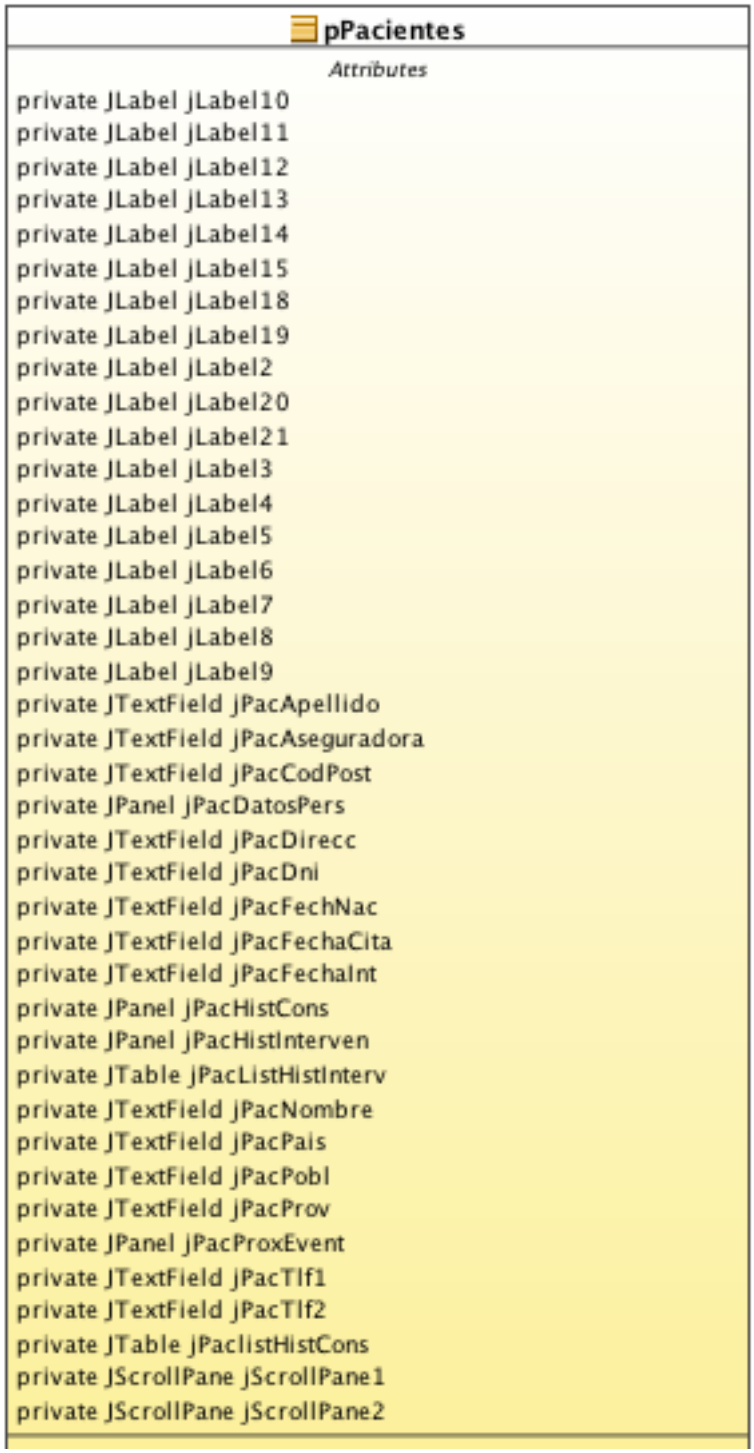



Fig. 8.4.7.1.5. Clase pEmpleados

#### *8.4.7.1.6. Clase pPacientes*

Esta clase contiene los atributos y métodos para la pestaña del panel principal que muestra información de pacientes.



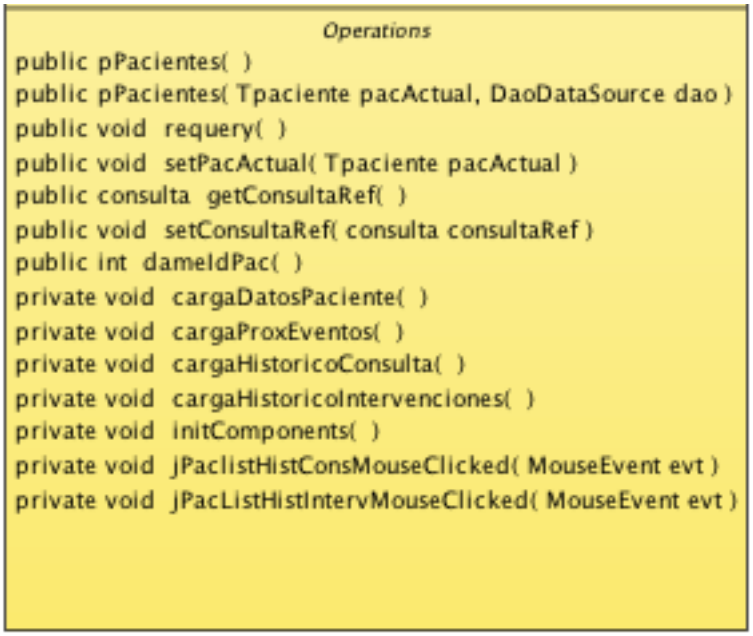



Fig. 8.4.7.1.6. Clase pPacientes

#### 8.4.7.1.7. Clase PQuirofanos

Esta clase contiene los atributos y métodos para la pestaña del panel principal que muestra información de intervenciones.

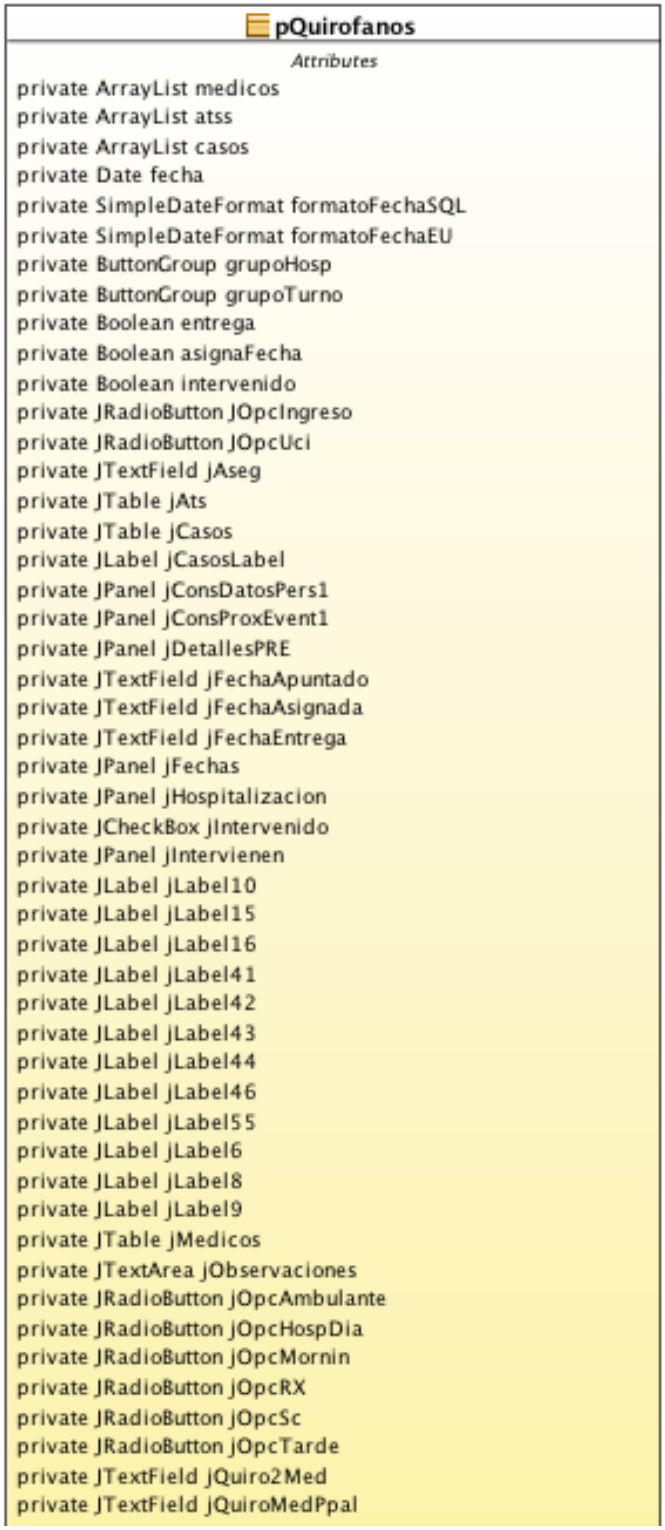



Fig. 8.4.7.1.7. Clase PQuirofanos

#### *8.4.7.1.8. Diagrama Clases Ventanas*

Este sub-paquete de GUIS contiene las ventanas emergentes de la aplicación.

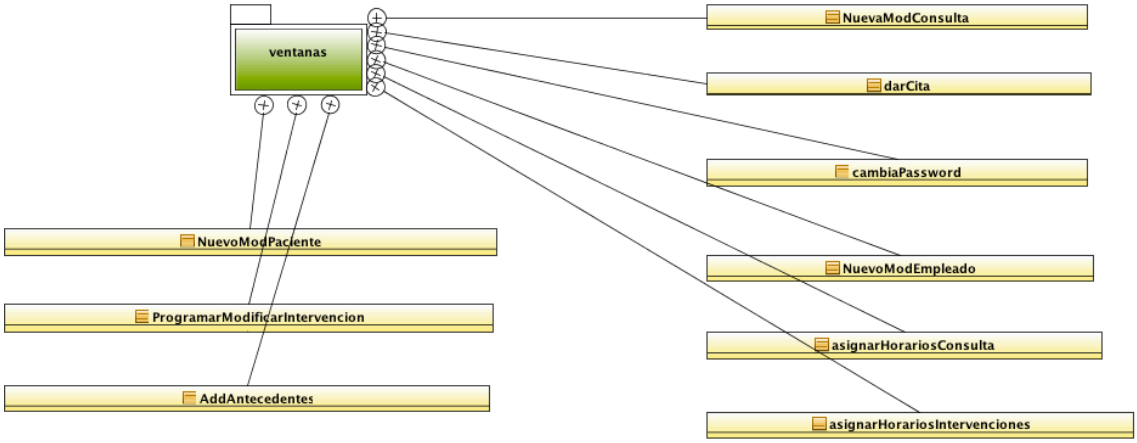



Fig. 8.4.7.1.8. Diagrama Clases Ventanas

#### *8.4.7.1.8.1. Clase AddAntecedentes*

Esta clase muestra los antecedentes del paciente.


Fig. 8.4.7.1.8.1. Clase AddAntecedentes

#### *8.4.7.1.8.2. Clase asignarHorariosConsulta*

Esta clase se utiliza para que el administrador asigne horarios de consulta a sus médicos empleados.

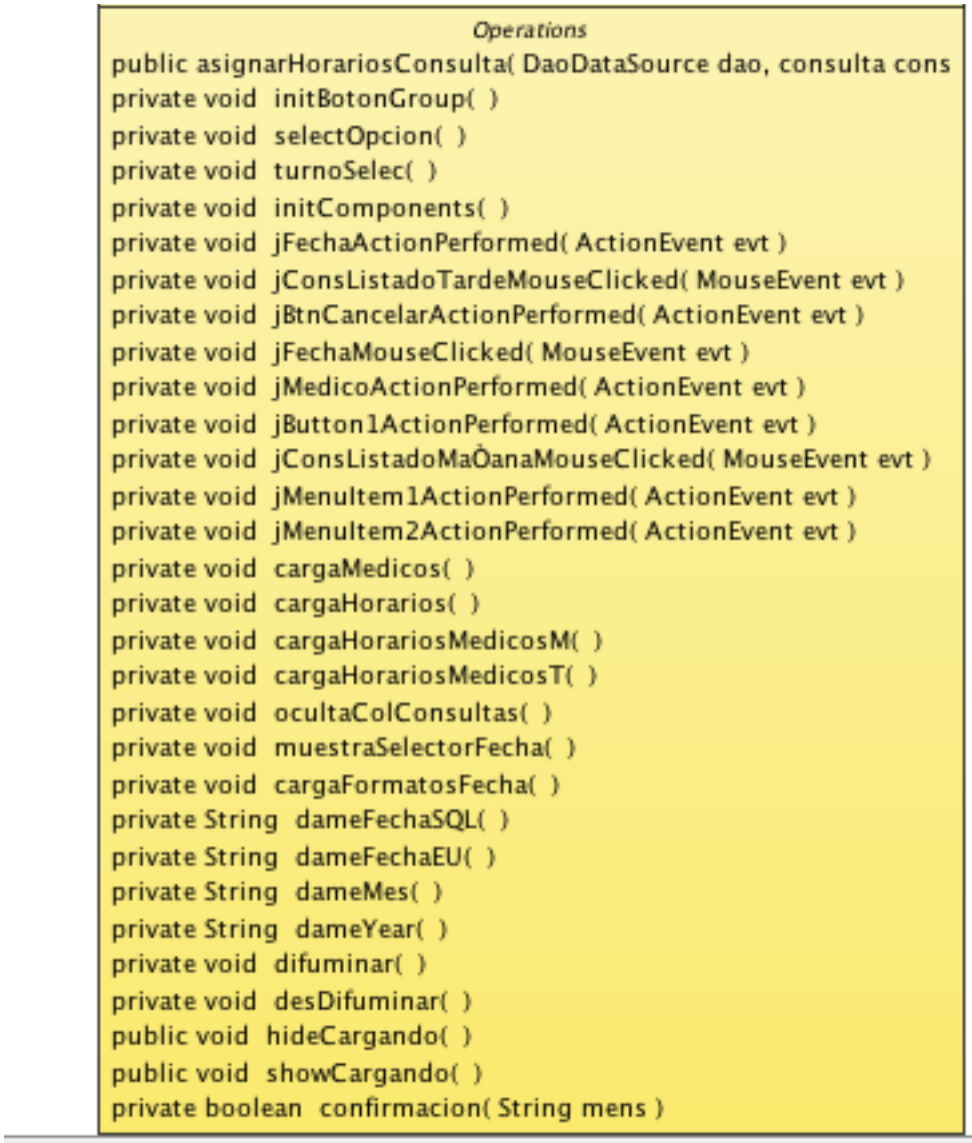
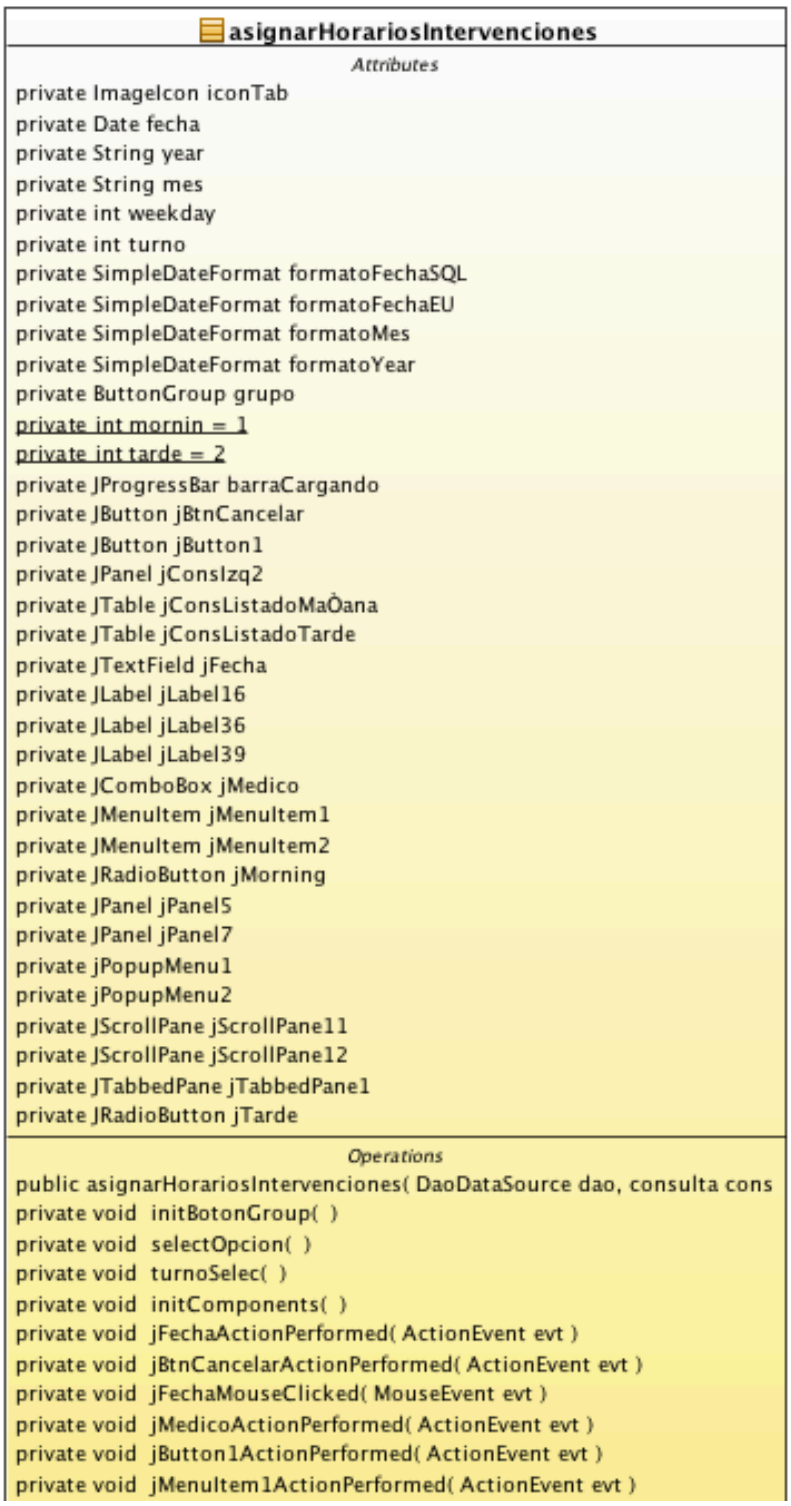



Fig. 8.4.7.1.8.2. Clase asignarHorariosConsulta

#### *8.4.7.1.8.3. Clase asignarHorariosIntervenciones*

sta clase se utiliza para que el administrador asigne horarios de intervenciones a sus médicos empleados.



```
private void  jMenuItem2ActionPerformed( ActionEvent evt )
private void  cargaMedicos( )
private void  cargaHorarios( )
private void  cargaHorariosIntervM( )
private void  cargaHorariosIntervT( )
private void  ocultaColConsultas( )
private void  muestraSelectorFecha( )
private void  cargaFormatosFecha( )
private String  dameFechaSQL( )
private String  dameFechaEU( )
private String  dameMes( )
private String  dameYear( )
private void  difuminar( )
private void  desDifuminar( )
public void  hideCargando( )
public void  showCargando( )
private boolean  confirmacion( String mens )
```

Fig. 8.4.7.1.8.3. Clase asignarHorariosIntervenciones

#### *8.4.7.1.8.4. Clase darCita*

Esta clase sirve para dar Citas a los pacientes.

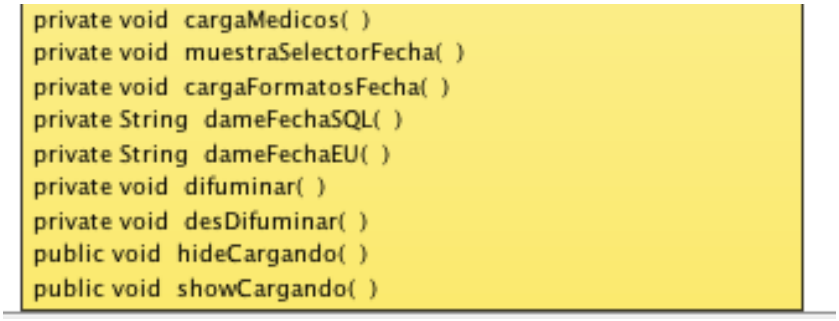
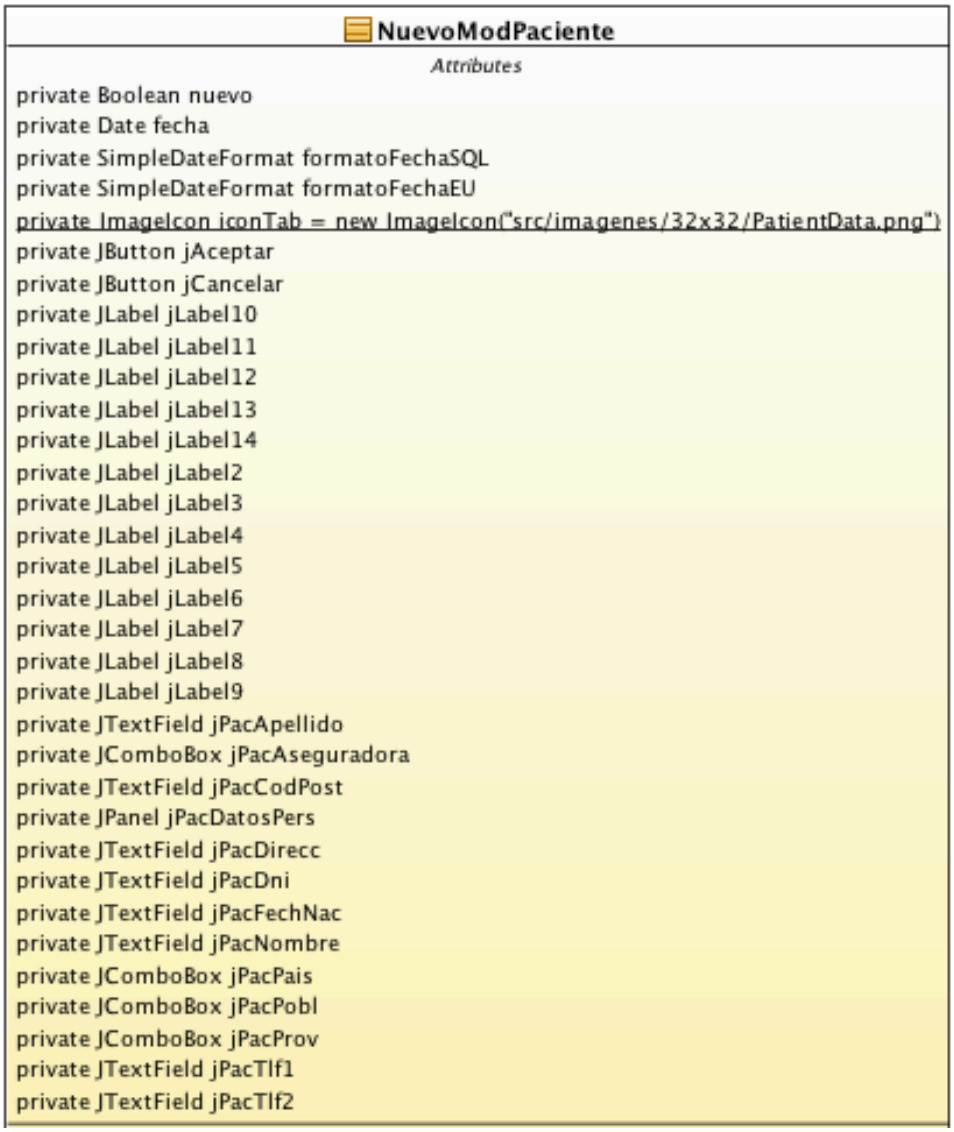



Fig. 8.4.7.1.8.4. Clase darCita

### *8.4.7.1.8.5. Clase NuevoModPaciente*

Esta clase se usa para modificar un paciente existente o dar de alta uno nuevo.



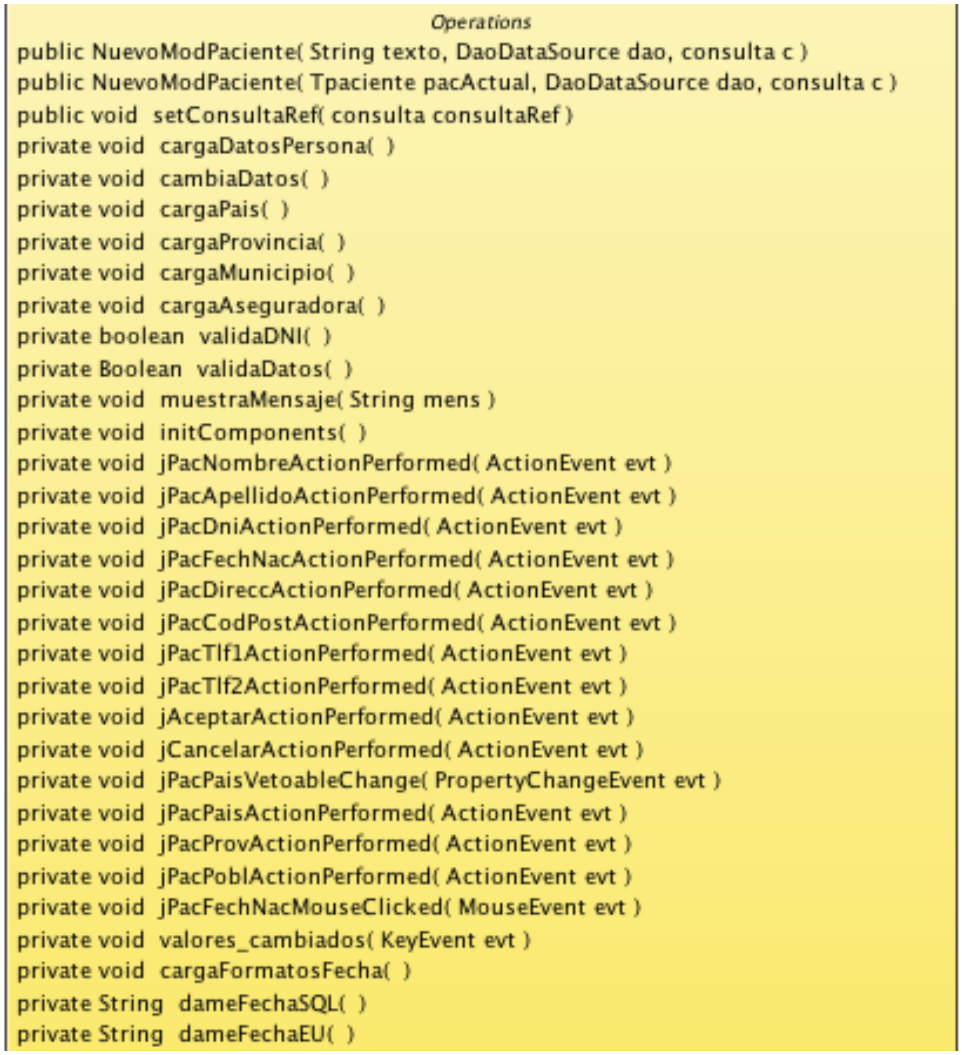
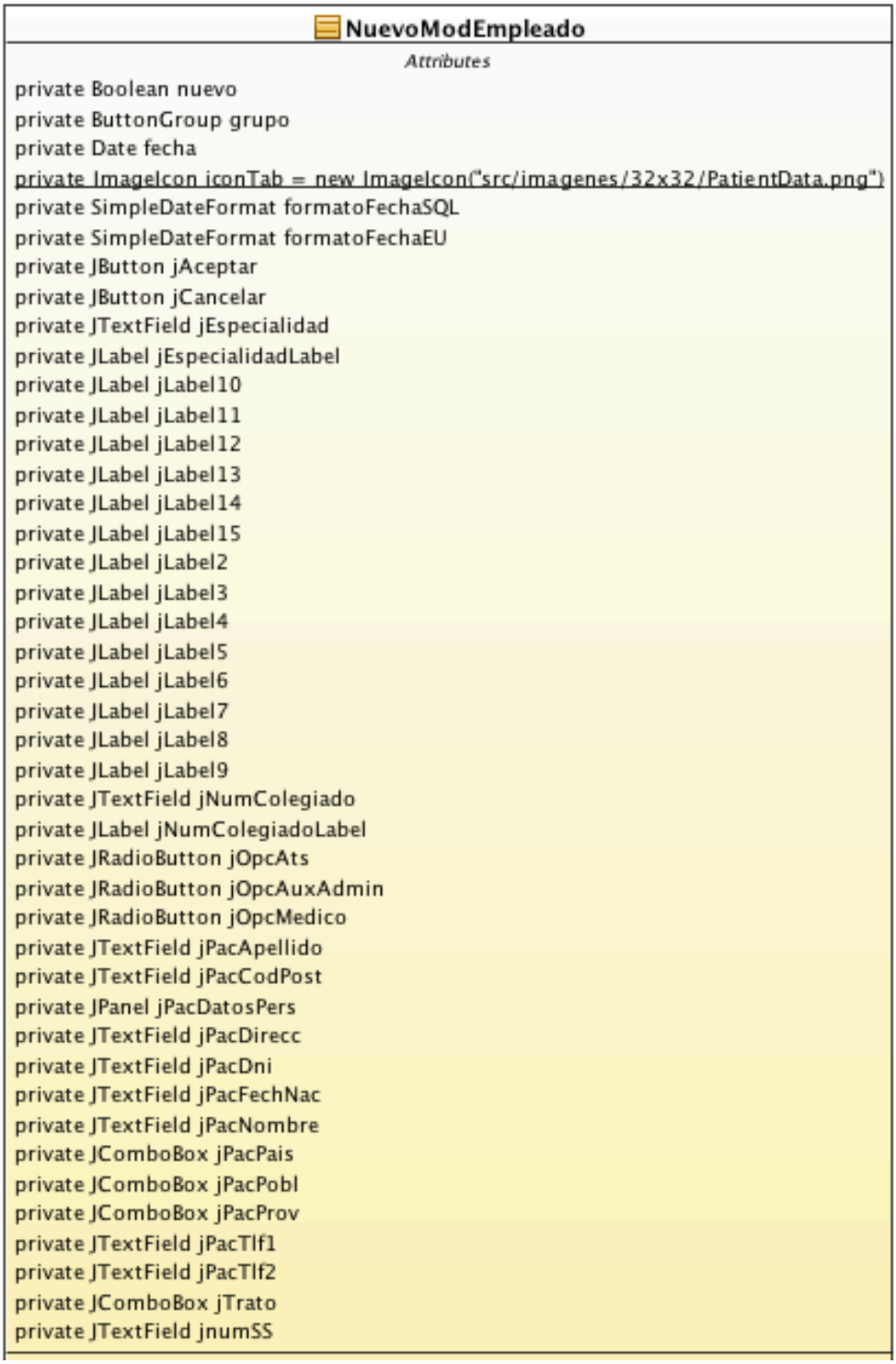



Fig. 8.4.7.1.8.5. Clase NuevoModPaciente

#### *8.4.7.1.8.6. Clase NuevoModEmpleado*

Esta clase se usa para modificar un empleado existente o dar de alta uno nuevo.



*Operations*

public NuevoModEmpleado( DaoDataSource dao, consulta c )
public NuevoModEmpleado( Templeado emplActual, DaoDataSource dao, consulta c )
public void setConsultaRef( consulta consultaRef )
private void cargaRoles( )
private void initBotonGroup( )
private void bloqueaGrupo( )
private void selectOpcion( )
private void cargaDatosPersona( )
private void cambiaDatos( )
private void cargaPais( )
private void cargaProvincia( )
private void cargaMunicipio( )
private void cargaTrato( )
private void setTratoResto( )
private void setTratoMedico( )
private boolean validaDNI( )
private Boolean validaDatos( )
private void muestraMensaje( String mens )
private void initComponents( )
private void jPacNombreActionPerformed( ActionEvent evt )
private void jPacApellidoActionPerformed( ActionEvent evt )
private void jPacDniActionPerformed( ActionEvent evt )
private void jPacFechNacActionPerformed( ActionEvent evt )
private void jPacDireccActionPerformed( ActionEvent evt )
private void jPacCodPostActionPerformed( ActionEvent evt )
private void jPacTlf1ActionPerformed( ActionEvent evt )
private void jPacTlf2ActionPerformed( ActionEvent evt )
private void jAceptarActionPerformed( ActionEvent evt )
private void jCancelarActionPerformed( ActionEvent evt )
private void jPacPaisVetoableChange( PropertyChangeEvent evt )
private void jPacPaisActionPerformed( ActionEvent evt )
private void jPacProvActionPerformed( ActionEvent evt )
private void jPacPoblActionPerformed( ActionEvent evt )
private void jnumSSActionPerformed( ActionEvent evt )
private void jPacFechNacMouseClicked( MouseEvent evt )
private void jNumColegiadoActionPerformed( ActionEvent evt )
private void jEspecialidadActionPerformed( ActionEvent evt )
private void jOpcMedicoActionPerformed( ActionEvent evt )
private void jOpcAtsActionPerformed( ActionEvent evt )
private void jOpcAuxAdminActionPerformed( ActionEvent evt )
private void cargaFormatosFecha( )
private String dameFechaSQL( )
private String dameFechaEU( )

Fig. 8.4.7.1.8.6. Clase NuevoModEmpleado

#### *8.4.7.1.8.7. Clase NuevoModConsulta*

Esta clase se usa para modificar una consulta de un paciente existente o dar crear una nueva.

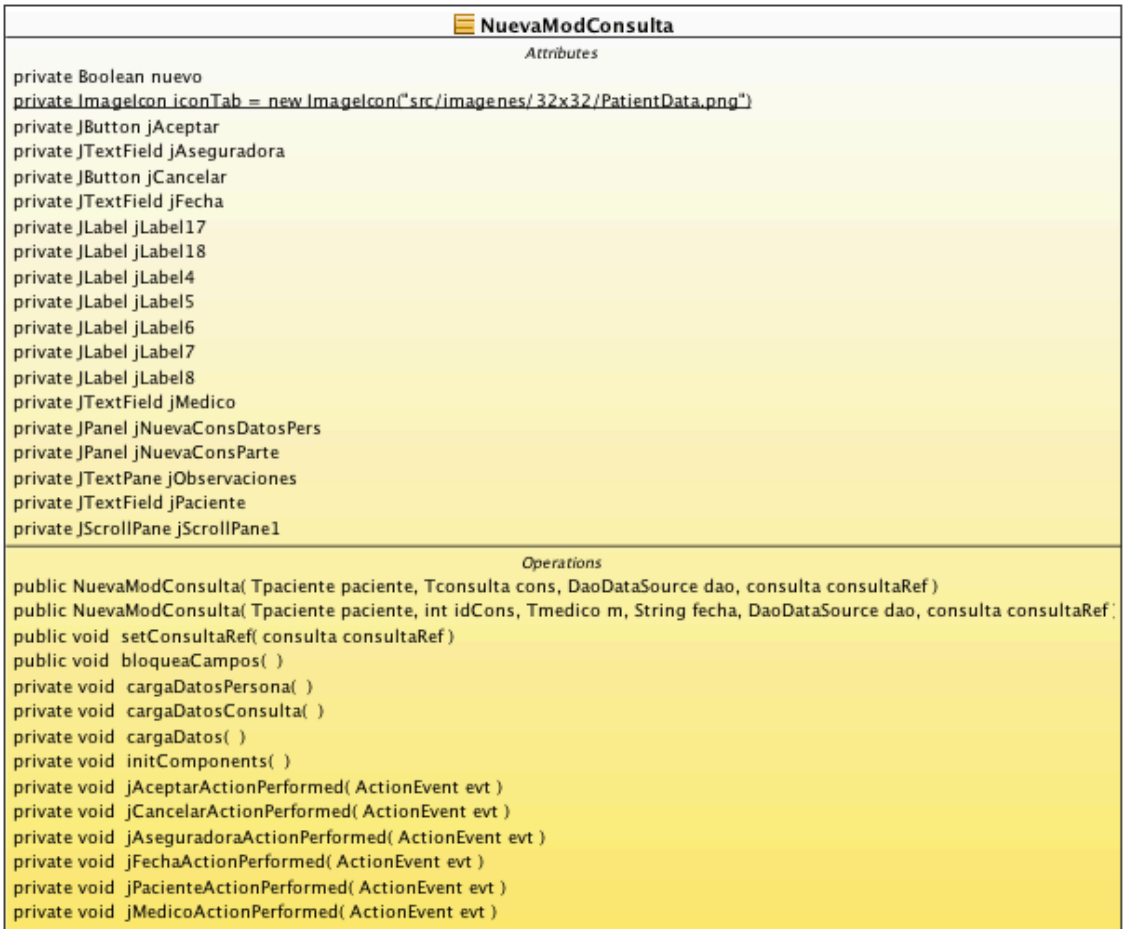



Fig. 8.4.7.1.8.7. Clase NuevoModConsulta

#### *8.4.7.1.8.8. Clase ProgramarModIntervencion*

Esta clase se usa para modificar una intervención de un paciente existente o dar crear una nueva.

private JLabel jLabel17
private JLabel jLabel4
private JLabel jLabel5
private JLabel jLabel6
private JLabel jLabel7
private JLabel jLabel8
private JLabel jLabel9
private JTable jMedicos
private JMenuItem jMenuItem1
private JMenuItem jMenuItem2
private JMenuItem jMenuItem3
private JPanel jNuevaConsDatosPers
private JPanel jNuevaConsParte
private JTextArea jObservaciones
private JRadioButton jOpcAmbulante
private JRadioButton jOpcHospDia
private JRadioButton jOpcMornin
private JRadioButton jOpcRX
private JRadioButton jOpcSc
private JRadioButton jOpcTarde
private JTextField jPaciente
private jPopupMenu1
private jPopupMenu2
private jPopupMenu3
private JScrollPane jScrollPane12
private JScrollPane jScrollPane13
private JScrollPane jScrollPane14
private JScrollPane jScrollPane3
private JScrollPane jScrollPane4
private JTextArea jTextArea1

*Operations*

public ProgramarModificarIntervencion( Tpaciente p, Tmedico m, DaoDataSource dao )
public ProgramarModificarIntervencion( Tpaciente paciente, Tmedico m, Tintervencion i, DaoDataSource dao )
public void vengoDeConsultas( )
private void buttonGroup( )
private void ocultaCampos( )
public void soloConsulta( )
private void cargaListas( )
public void setConsultaRef( consulta consultaRef )
private void cargaDatosPersona( )
private void cargaDatosIntervencion( )
private void cambiarDatos( )
private void cargaMedicos( )
private void cargaAts( )
private void cargaCasos( )
private void muestraSelectorFecha( )

```
private String  dameFechaSQL( )
private String  dameFechaEU( )
private void  recargaCasos( )
private void  recargaMedicos( )
private void  recargaAts( )
private void  recargaFechas( )
private void  recargaDetallesPre( )
private void  recargaDetallesHosp( )
private void  agregaCaso( )
private void  cargaCasosTabla( )
private void  pasaCasosTabla( Object datos[0..*,0..*] )
private void  agregaMedico( )
private void  cargaMedicoTabla( )
private void  pasaMedicoTabla( Object datos[0..*,0..*] )
private void  agregaAts( )
private void  cargaAtsTabla( )
private void  pasaAtsTabla( Object datos[0..*,0..*] )
private void  actualizarIntervencion( )
private void  initComponents( )
private void  jAceptarActionPerformed( ActionEvent evt )
private void  jCancelarActionPerformed( ActionEvent evt )
private void  jFechaAsignadaActionPerformed( ActionEvent evt )
private void  jEntregaActionPerformed( ActionEvent evt )
private void  jBotonNewCasoActionPerformed( ActionEvent evt )
private void  jBotonAddCasoActionPerformed( ActionEvent evt )
private void  jBotonAddMedicoActionPerformed( ActionEvent evt )
private void  jBotonAddAtsActionPerformed( ActionEvent evt )
private void  jFechaAsignadaMouseClicked( MouseEvent evt )
private void  jIntervenidoActionPerformed( ActionEvent evt )
private void  jMenuItem1ActionPerformed( ActionEvent evt )
private void  jMenuItem2ActionPerformed( ActionEvent evt )
private void  jMenuItem3ActionPerformed( ActionEvent evt )
```

Fig. 8.4.7.1.8.8. Clase ProgramarModIntervencion

#### 8.4.7.1.8.9. Clase cambiaPassword

Para cambiar la contraseña del usuario logueado en el sistema actualmente.

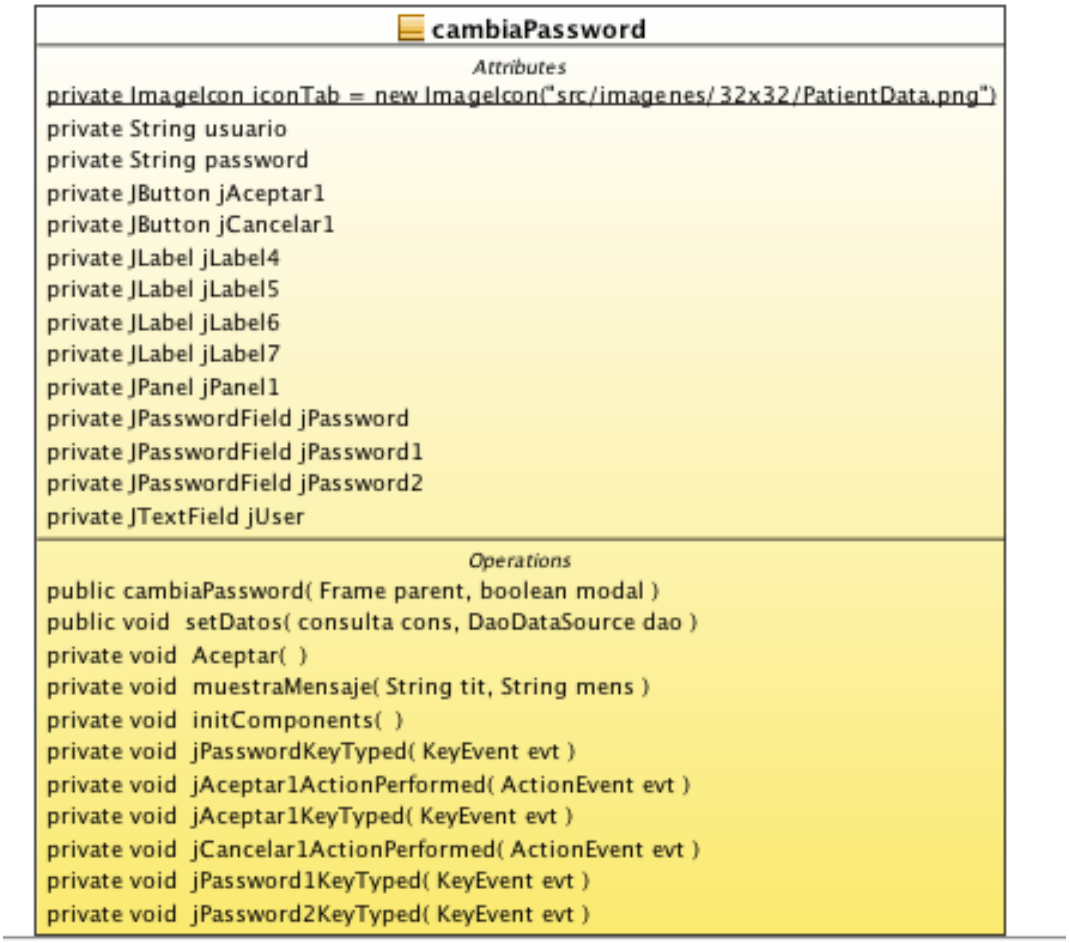



Fig. 8.4.7.1.8.9. Clase cambiaPassword

### 8.4.7.2. Diagrama Clases SRC

La capa procesos se subdivide en dos diferenciadas por ofrecer la funcionalidad de la aplicación y la segunda por proveer las acciones sobre la base de datos. La primera compuesta del paquete actores y casos. Y la segunda es el paquete procesos.

#### 8.4.7.2.1. Diagrama Clases SRC

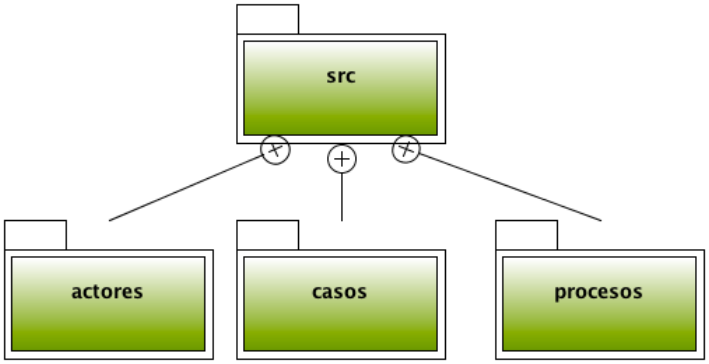



Fig. 8.4.7.2.1 Diagrama Clases SRC

#### 8.4.7.2.2. Diagrama Clases Actores

Los actores que participan activa o pasivamente en la aplicación.

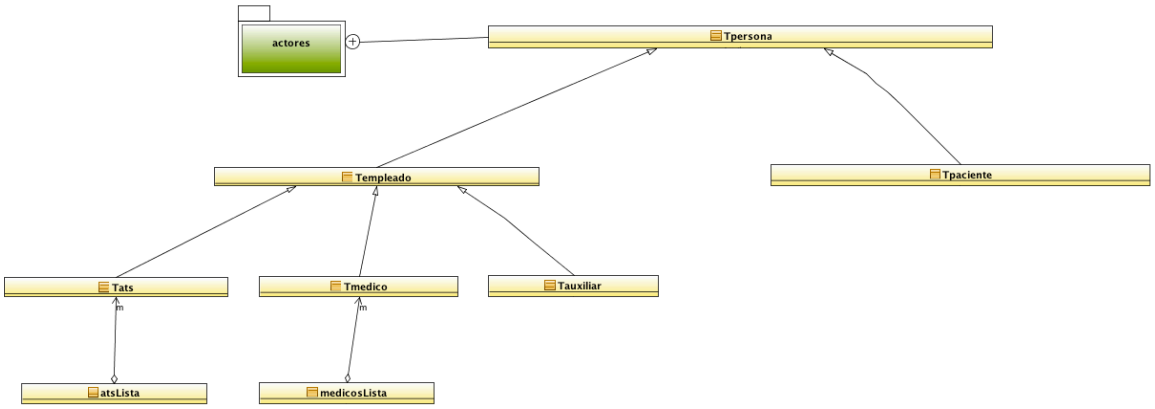



Fig. 8.4.7.2.2. Diagrama Clases Actores

#### *8.4.7.2.2.1. Clase TPersona*

Clase principal de la cual heredan el resto. Contiene los atributos y métodos propios.


Fig. 8.4.7.2.2.1. Clase TPersona

#### 8.4.7.2.2.2. Clase TPaciente

Clase que hereda de Tpersona. Contiene los atributos y métodos propios.

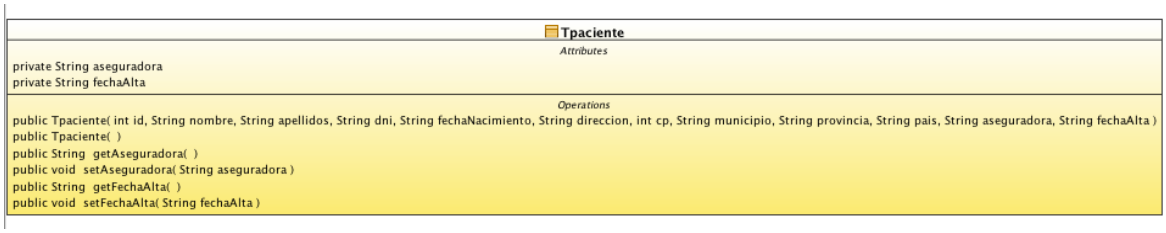



Fig. 8.4.7.2.2.2. Clase TPaciente

#### 8.4.7.2.2.3. Clase TEmpleado

Clase que hereda de Tpersona. Contiene los atributos y métodos propios. De aquí heredan el resto de empleados.

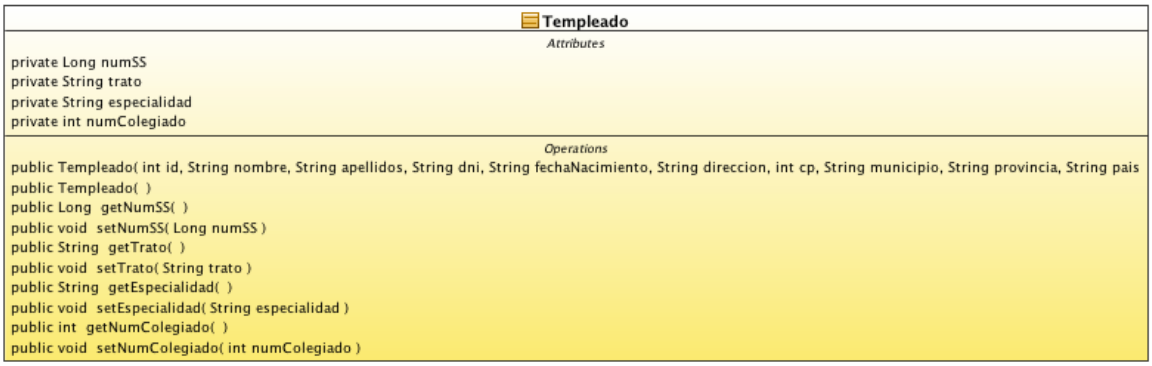



Fig. 8.4.7.2.2.3. Clase TEmpleado

#### 8.4.7.2.2.4. Clase Tmedico

Clase que hereda de Templeado. Contiene los atributos y métodos propios.

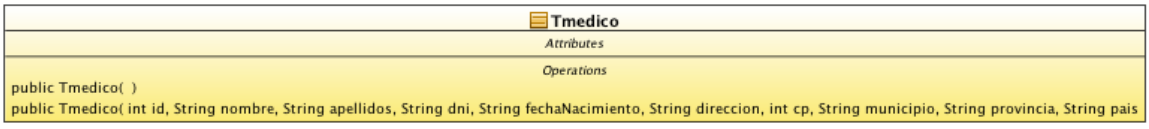



Fig. 8.4.7.2.2.4. Clase Tmedico

#### 8.4.7.2.2.5. Clase Tats

Clase que hereda de Templeado. Contiene los atributos y métodos propios.

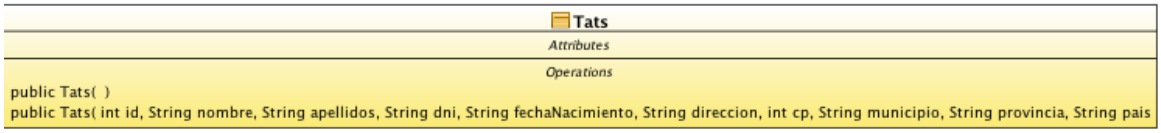


Fig. 8.4.7.2.2.5. Clase Tats

#### *8.4.7.2.2.6. Clase Tauxiliar*

Clase que hereda de Templeado. Contiene los atributos y métodos propios.

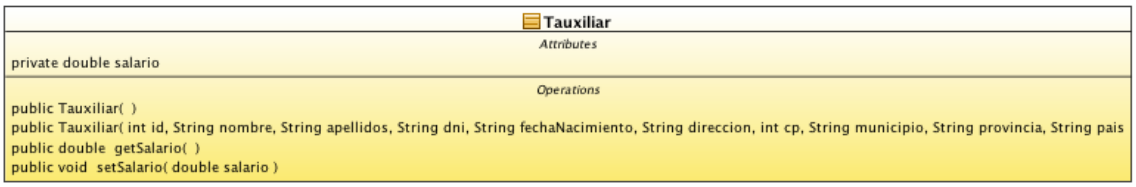



Fig. 8.4.7.2.2.6. Clase Tauxiliar

#### *8.4.7.2.2.7. Clase atsLista*

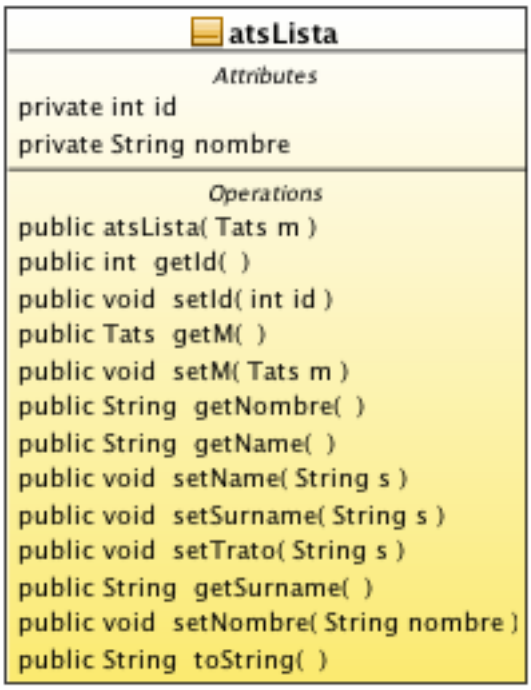



Fig. 8.4.7.2.2.7. Clase atsLista

#### 8.4.7.2.2.8. Clase medicosLista

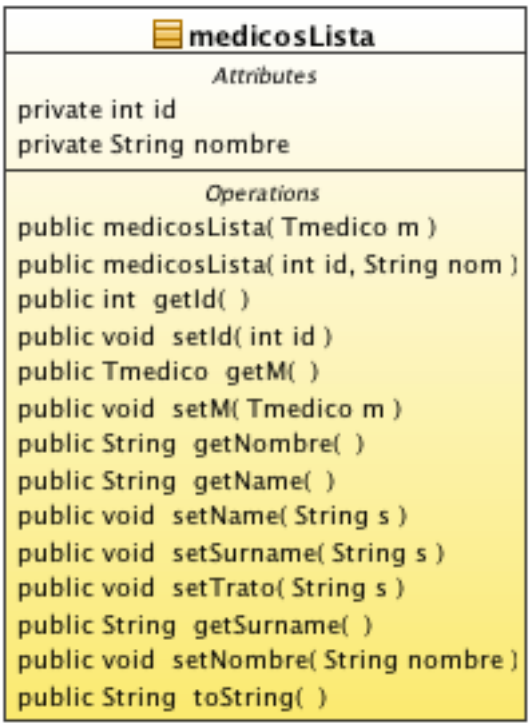



Fig. 8.4.7.2.2.8. Clase medicosLista

### 8.4.7.2.3. Diagrama Clases Casos

Los diferentes objectos que manejamos en la aplicación.

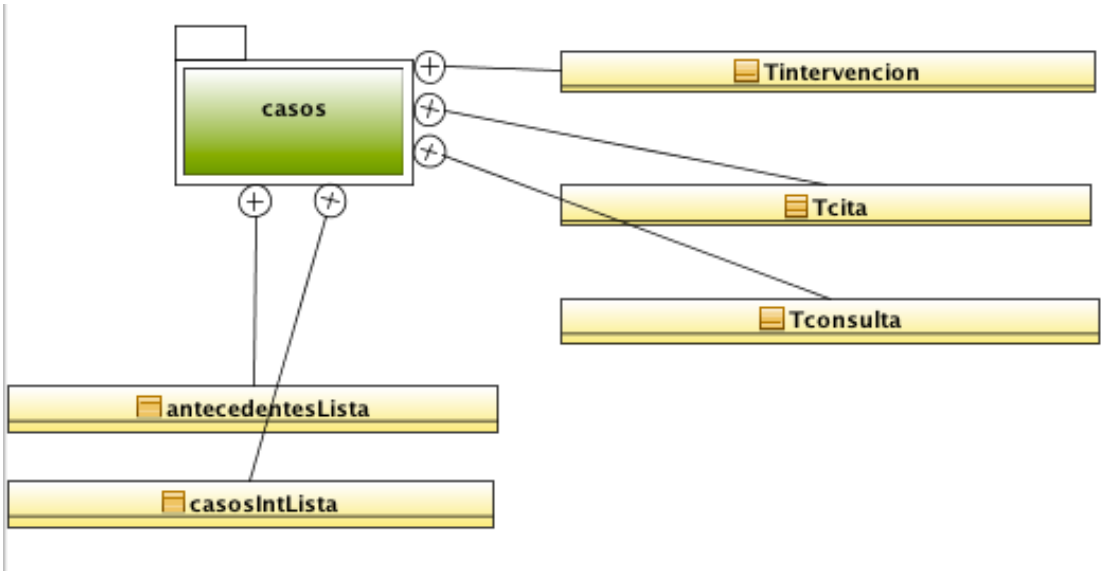



Fig. 8.4.7.2.3. Diagrama Clases Casos

#### *8.4.7.2.3.1. Clase Tcita*

Contiene los atributos y métodos necesarios para implementar una cita.

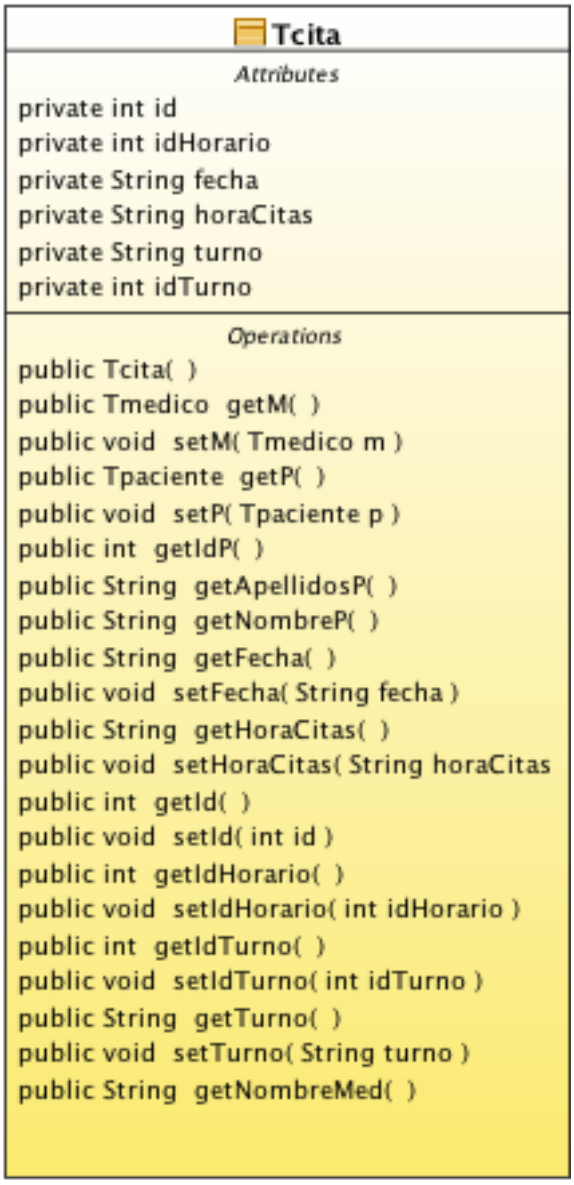



Fig. 8.4.7.2.3.1. Clase Tcita

#### *8.4.7.2.3.2. Clase Tconsulta*

Contiene los atributos y métodos necesarios para implementar una consulta.

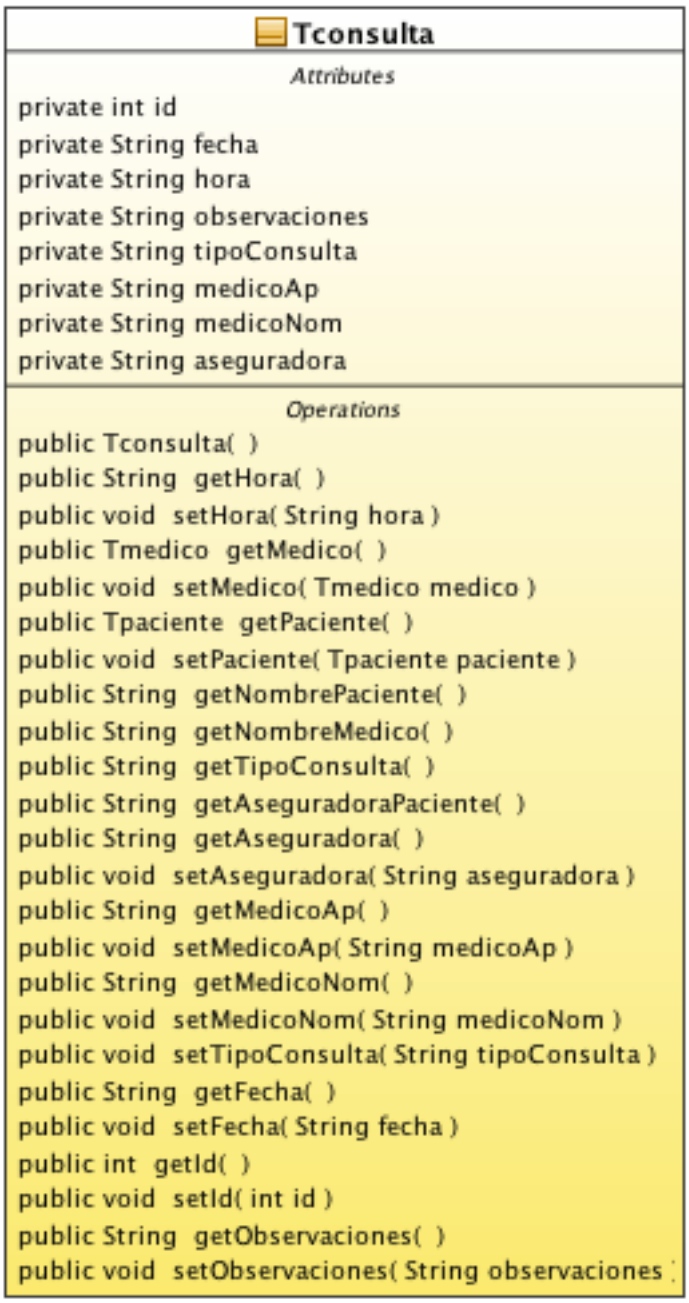
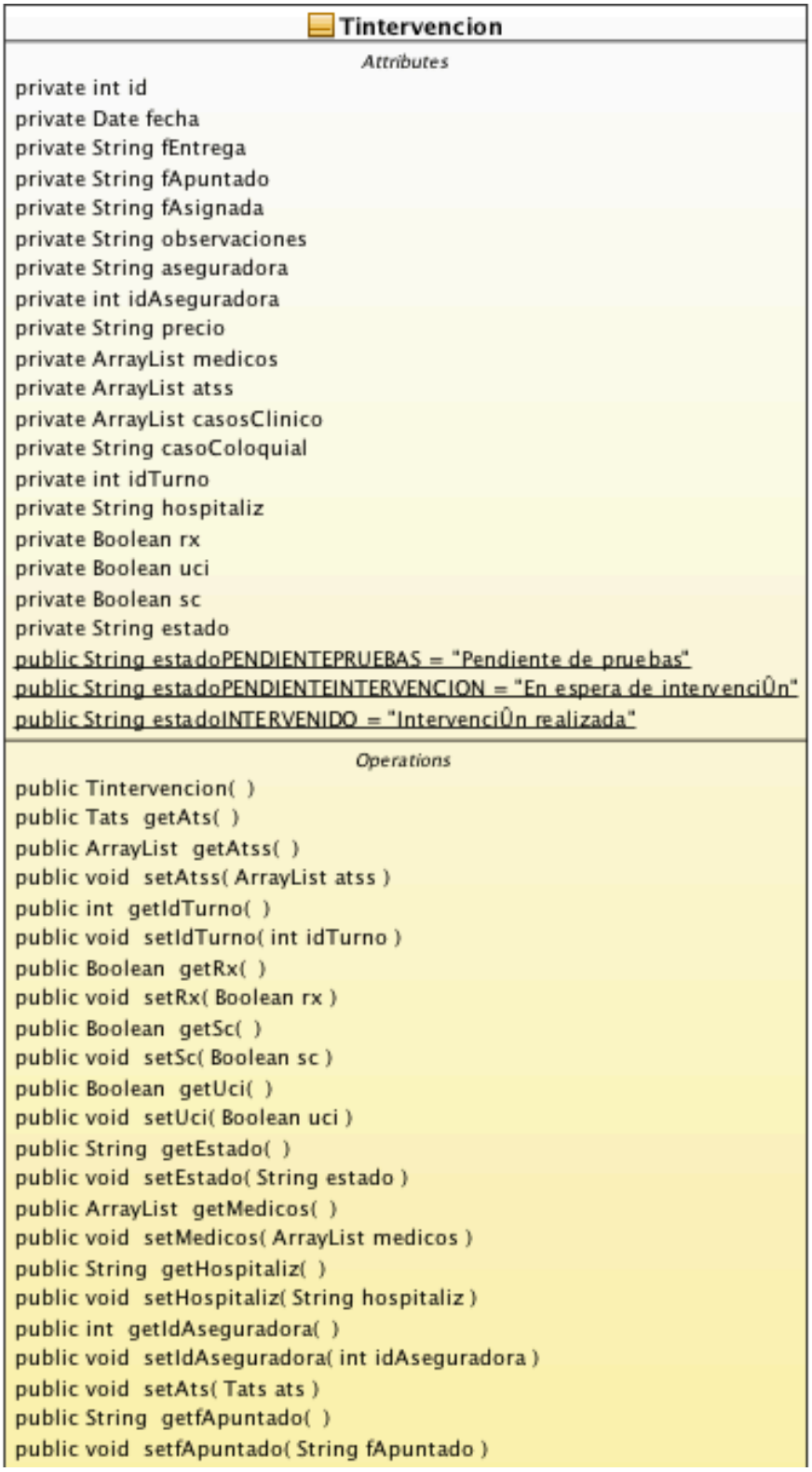



Fig. 8.4.7.2.3.2. Clase Tconsulta

#### *8.4.7.2.3.3. Clase Tintervencion*

Contiene los atributos y métodos necesarios para implementar una intervención.



```
public String  getfAsignada( )
public void  setfAsignada( String fAsignada )
public String  getfEntrega( )
public void  setfEntrega( String fEntrega )
public String  getCasoColoquial( )
public void  setCasoColoquial( String casoColoquial )
public ArrayList  getCasosClinico( )
public void  setCasosClinico( ArrayList casosClinico )
public String  getMedicoPreparaNombre( )
public String  getMedicoEntregaNombre( )
public Tmedico  getM1( )
public void  setM1( Tmedico m1 )
public Tmedico  getM2( )
public void  setM2( Tmedico m2 )
public Tmedico  getmPrograma( )
public void  setmPrograma( Tmedico mPrograma )
public Tpaciente  getPaciente( )
public void  setPaciente( Tpaciente paciente )
public String  getNombrePaciente( )
public String  getAseguradora( )
public void  setAseguradora( String aseguradora )
public Date  getFecha( )
public void  setFecha( Date fecha )
public int  getId( )
public void  setId( int id )
public String  getObservaciones( )
public void  setObservaciones( String observaciones )
public String  getPrecio( )
public void  setPrecio( String precio )
```

Fig. 8.4.7.2.3.3. Clase Tintervencion

#### *8.4.7.2.4. Diagrama Clases Procesos*

Este paquete contiene la interfaz que luego implemente la clase Dao para todas las transacciones con la base de datos.

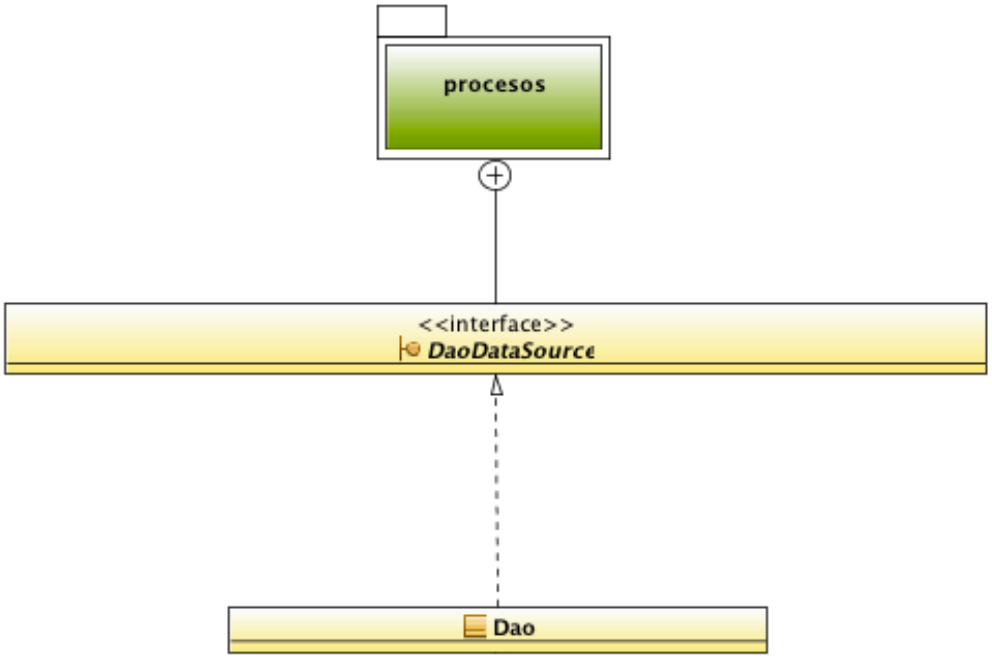
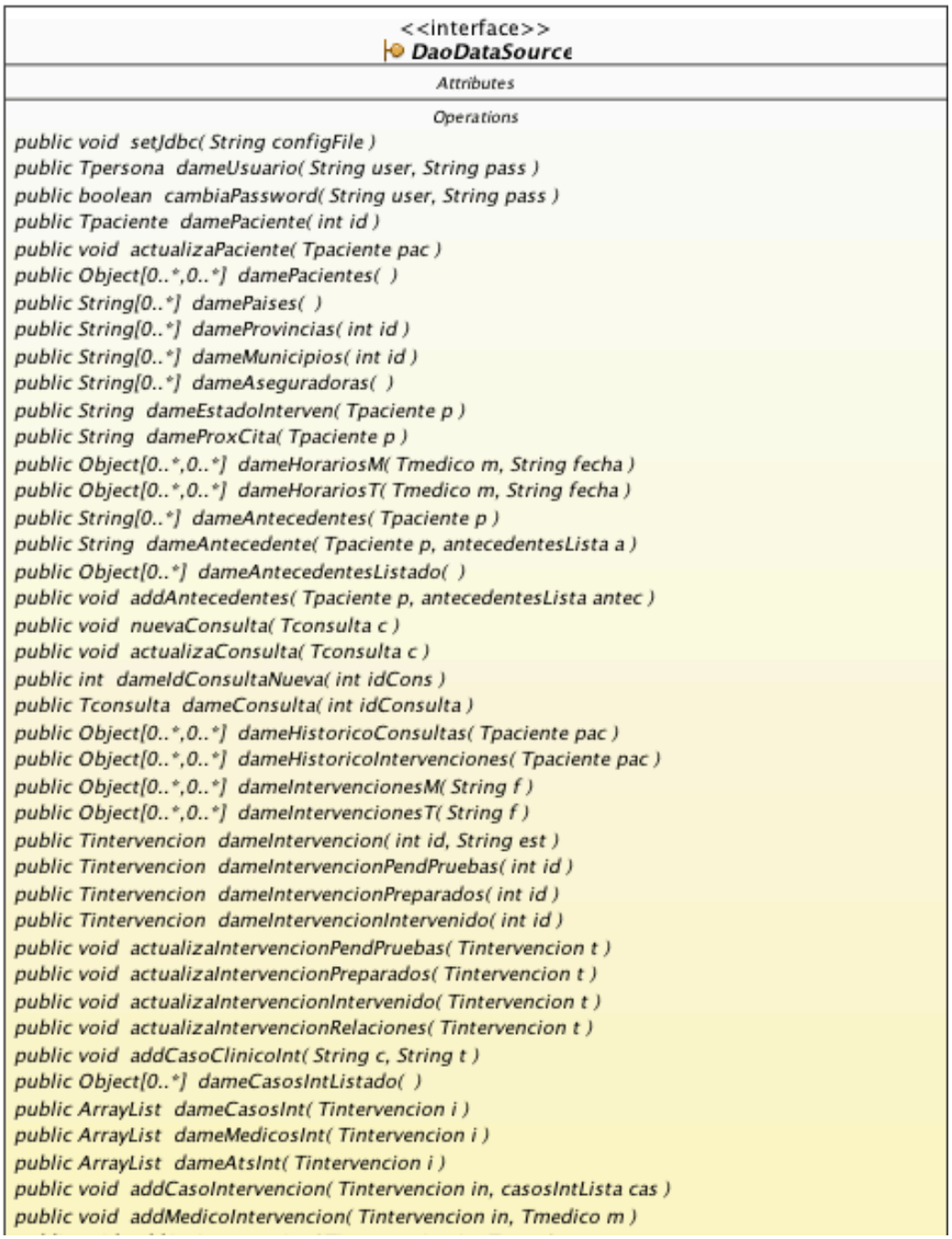



Fig. 8.4.7.2.4. Diagrama Clases Procesos

#### 8.4.7.2.4.1. Interfaz DaoDataSource

En esta interfaz se muestran los métodos de transacciones con la base de datos que usamos en la aplicación.

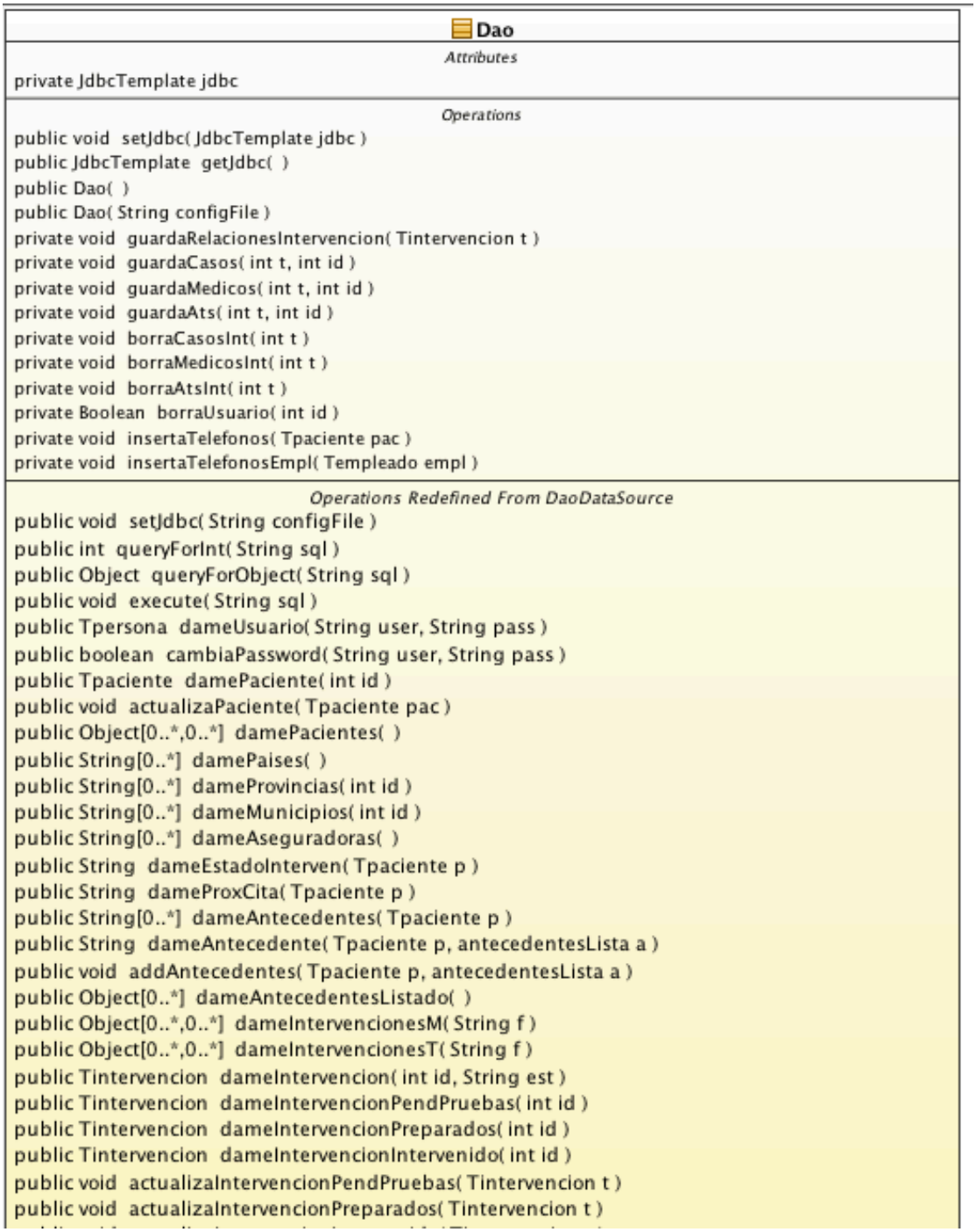


```
public void  addAtsIntervencion( Tintervencion in, Tats a )
public boolean  getRx( Tintervencion interv )
public boolean  getUci( Tintervencion interv )
public boolean  getSc( Tintervencion interv )
public boolean  getRxIntervencion( Tintervencion interv )
public boolean  getUciIntervencion( Tintervencion interv )
public boolean  getScIntervencion( Tintervencion interv )
public Object[0..*,0..*]  dameListadoIntervenciones( String fecha, int turno )
public Object[0..*,0..*]  dameMedicos( )
public Object[0..*]  dameMedicosListado( )
public Object[0..*]  dameAtsListado( )
public Object[0..*,0..*]  dameATS( )
public Object[0..*,0..*]  dameAuxiliares( )
public void  eliminaEmpleado( int id )
public Templeado  dameEmpleado( int id )
public Templeado  dameEmpleadoMedico( int id )
public Templeado  dameEmpleadoAts( int id )
public Templeado  dameEmpleadoAuxAdmin( int id )
public void  actualizaEmpleado( Templeado empl )
public void  actualizaEmpleadoMedico( Templeado empl )
public void  actualizaEmpleadoAts( Templeado empl )
public void  actualizaEmpleadoAuxAdmin( Templeado empl )
public Object[0..*,0..*]  dameHorariosMedM( String fecha )
public Object[0..*,0..*]  dameHorariosMedT( String fecha )
public Object[0..*,0..*]  dameHorariosMedIntM( String fecha )
public Object[0..*,0..*]  dameHorariosMedIntT( String fecha )
public Boolean  insertaPac( Tpaciente paciente )
public Boolean  darCita( Tpaciente paciente, int idMedico, String fecha, int idHorario, String tipo )
public Boolean  insertaMedico( Templeado m )
public Boolean  insertaAts( Templeado a )
public Boolean  insertaAuxAdmin( Templeado a )
public Boolean  rellenaFecha( Tmedico m, String year, String mes, int day, int turno )
public Boolean  rellenaFechaIntervenciones( Tmedico m, String year, String mes, int day, int turno )
public Boolean  crealntervencion( Tintervencion interv )
public Boolean  entregaPruebas( Tintervencion interv, int rx, int uci, int sc )
public Boolean  intervenido( Tintervencion interv )
public Boolean  borraFecha( int m, String f, int turno )
public Boolean  borraFechaIntervenciones( int m, String f, int turno )
public Boolean  borrarCita( int id )
public Boolean  borrarPaciente( int id )
public Boolean  borrarMedico( int id )
public Boolean  borrarAts( int id )
public Boolean  borrarAuxAdmin( int id )
public int  queryForInt( String sql )
public Object  queryForObject( String sql )
public void  execute( String sql )
public void  inserta( )
public void  update( )
public void  delete( )
```

Fig. 8.4.7.2.4.1. Interfaz DaoDataSource

#### 8.4.7.2.4.2. Clase Dao

Esta clase implementa la interfaz DaoDataSource con todos los métodos y sus sentencias sql, call procedures para realizar transacciones con la base de datos.



```
public void  actualizaIntervencionIntervenido( Tintervencion t )
public void  actualizaIntervencionRelaciones( Tintervencion t )
public Object[0..*]  dameCasosIntListado( )
public void  addCasoClinicoInt( String c, String t )
public ArrayList  dameCasosInt( Tintervencion interv )
public ArrayList  dameMedicosInt( Tintervencion interv )
public ArrayList  dameAtsInt( Tintervencion interv )
public void  addCasoIntervencion( Tintervencion in, casosIntLista cas )
public void  addMedicoIntervencion( Tintervencion in, Tmedico m )
public void  addAtsIntervencion( Tintervencion in, Tats a )
public boolean  getRx( Tintervencion interv )
public boolean  getUci( Tintervencion interv )
public boolean  getSc( Tintervencion interv )
public boolean  getRxIntervencion( Tintervencion interv )
public boolean  getUciIntervencion( Tintervencion interv )
public boolean  getScIntervencion( Tintervencion interv )
public Object[0..*,0..*]  dameListadoIntervenciones( String fecha, int turno )
public Object[0..*,0..*]  dameMedicos( )
public Object[0..*]  dameMedicosListado( )
public Object[0..*]  dameAtsListado( )
public Object[0..*,0..*]  dameATS( )
public Object[0..*,0..*]  dameAuxiliares( )
public void  eliminaEmpleado( int id )
public Templeado  dameEmpleado( int id )
public Templeado  dameEmpleadoMedico( int id )
public Templeado  dameEmpleadoAts( int id )
public Templeado  dameEmpleadoAuxAdmin( int id )
public void  actualizaEmpleado( Templeado empl )
public void  actualizaEmpleadoMedico( Templeado empl )
public void  actualizaEmpleadoAts( Templeado empl )
public void  actualizaEmpleadoAuxAdmin( Templeado empl )
public Object[0..*,0..*]  dameHorariosMedM( String fecha )
public Object[0..*,0..*]  dameHorariosMedT( String fecha )
public Object[0..*,0..*]  dameHorariosMedIntM( String fecha )
public Object[0..*,0..*]  dameHorariosMedIntT( String fecha )
public Object[0..*,0..*]  dameHorariosM( Tmedico m, String fecha )
public Object[0..*,0..*]  dameHorariosT( Tmedico m, String fecha )
public void  nuevaConsulta( Tconsulta c )
public void  actualizaConsulta( Tconsulta c )
public int  dameIdConsultaNueva( int idCons )
public Tconsulta  dameConsulta( int id )
public Object[0..*,0..*]  dameHistoricoConsultas( Tpaciente pac )
public Object[0..*,0..*]  dameHistoricoIntervenciones( Tpaciente pac )
public Boolean  insertaPac( Tpaciente p )
public Boolean  darCita( Tpaciente p, int idMedico, String fecha, int idHorario, String tipo )
```

```
public Boolean  insertaMedico( Templeado empl )
public Boolean  insertaAts( Templeado empl )
public Boolean  insertaAuxAdmin( Templeado empl )
public Boolean  rellenaFecha( Tmedico m, String year, String mes, int day, int turno )
public Boolean  rellenaFechaIntervenciones( Tmedico m, String year, String mes, int day, int turno )
public Boolean  crealntervencion( Tintervencion interv )
public Boolean  entregaPruebas( Tintervencion interv, int rx, int uci, int sc )
public Boolean  intervenido( Tintervencion interv )
public Boolean  borraFecha( int m, String f, int turno )
public Boolean  borraFechaIntervenciones( int m, String f, int turno )
public Boolean  borrarCita( int id )
public Boolean  borrarPaciente( int id )
public Boolean  borrarMedico( int id )
public Boolean  borrarAts( int id )
public Boolean  borrarAuxAdmin( int id )
public void  inserta( )
public void  update( )
public void  delete( )
```

Fig. 8.4.7.2.4.2. Clase Dao

## 8.5.Otras tecnologías

Para el desarrollo del proyecto han sido necesarias las siguientes tecnologías. El repositorio en Google Code para poder mantener un control de versiones del desarrollo del proyecto Java. Los programas de conexión al repositorio. Y Las nuevas tecnologías de Google como Calendar, Sites o Groups que nos han permitido comunicarnos en todo momento, concertar citas o tener una web donde mantenernos al día de los progresos del desarrollo, búsquedas de información, debates y discusiones .

### 8.5.1. SVN subversion (repositorio)

### 8.5.2. Subclipse

Plugin de Eclipse que permite la comunicación con un repositorio y su tratamiento directamente con el programa Eclipse para la creación y  modificación de proyectos.

### 8.5.3. TortoiseSVN

Aplicación que se integra en la interfaz de Windows y permite descargar a local las carpetas y archivos en local  contenidos en el repositorio. Es muy popular debido a su fácil utilización y la sencilla interfaz gráfica.

### 8.5.4. Google

Google proporciona diversos servicios y aplicaciones web que colaboran al trabajo de grupos en red en desarrollo de proyectos. Todos estos servicios se comparten a través de la cuenta de correo electrónico de Gmail.

#### *8.5.4.1. Code*

Alojamiento de proyectos y repositorios de código libre. Además de alojar el proyecto tiene una API bastante interesante, en la cuál puedes ver el árbol de carpetas del proyecto, un gestor de versiones con el cual puedes comparar distintas versiones y ver los cambios, volver a una versión anterior del proyecto.

#### *8.5.4.2. Groups*

Servicio de comunicación para el grupo, con foro de debate y alojamiento de archivos de proyecto.

#### *8.5.4.3. Calendar*

El calendario compartido entre los participantes permite fijar fechas de reuniones con alertas y visualización de próximos eventos.

#### *8.5.4.4. Docs*

A través de GoogleDocs mantenemos en línea varios documento de texto que podemos modificar los integrantes del proyecto online. Sin necesidad de instalar programas de ofimática.

#### *8.5.4.5. Sites*

Sitio web de creación con plantillas, el cual integra otros servicios de Google como son GoogleDocs, GoogleCalendar. En el sitio web se introduce un historial de reuniones, listado de tareas pendientes, vista del calendario común a los miembros del grupo con fechas de reuniones., blog para comentar tecnologías o noticias que pueden ayudar a mejorar la aplicación.

### 8.5.5. XAMPP. Servidor

La Base de Datos se encuentra físicamente en el servidor creado por nosotros gracias a la herramienta XAMPP que integra Apache, MySQL, Filezilla ,Mercury Mail Transport System, Tomcat, Filezilla entre otros servicios.

XAMPP es una compilación de software libre (comparable a una distribución Linux), es gratuito y libre para ser copiado conforme los términos de la licencia GNU General Public License. Sin embargo, sólo la compilación de XAMPP está publicada bajo la licencia GPL..

En nuestro caso usamos Apache para el servidor, MySql para la gestión de la base da datos, Filezilla como cliente/servidor FTP para las transacciones de archivos desde cualquier lugar, PhpMyAdmin para manejar la creación de la base de datos, sus tablas, procedures, relaciones, etc. y Mercury para el envío de correos electrónicos para la confirmación de alta de un usuario.

Para su conexión en internet debíamos asignar un dirección web asociada a su IP. En este caso el lugar de instalación del servidor tenía IP dinámica y por tanto fue necesario recurrir a la ayuda de los servicios que provee no-ip.com. Una web que provee un software gratuito el cual asocia una dirección web estática a tu ip dinámica, con una frecuencia de refresco en el escaneo de la IP para registrar cambios mantener la conexión siempre estable. Losparámetros de conexión los configuramos son sencillos utilizando ambos programas.

# 9. Estimación de costes

Para hacer el estudio de costes de toda la aplicación se han de considerar los siguientes elementos:

1. Sueldos de los empleados
2. Recursos informáticos
3. Licencias de software
4. Instalaciones
5. Mantenimiento mensual
6. Seguro
7. Enfoque de mercado

## 9.1.Sueldos de los empleados:

Se estima que cada uno de los tres empleados trabaja una media de 10 horas semanales cobrando 10 € a la hora

10 horas semanales * 10 €/h *3 empleados = 300 €/semanales -> 1200€/mensuales.

## 9.2.Recursos informáticos:

Se calcula que se han de invertir en torno a los 5000€ en material informático, es decir, ordenadores, unidades de almacenamiento, consumibles, etc…

## 9.3.Licencias de software:

Teniendo en cuenta que el software se instalará en una clínica entera, en principio no se requiere ninguna licencia adicional.

## 9.4.Instalaciones:

Para este apartado se ha supuesto que el equipo trabajaba en un local alquilado y se han añadido los costes de mantenimiento de la red, equipos y la luz.

Luz, equipos, red: 75 €/mes

Alquiler de local: 500 €/mes

## 9.5.Mantenimiento mensual:

El mantenimiento incluye el soporte software durante el primer año, no se incluye ningún tipo de soporte sobre los dispositivos físicos de la clínica ya que nosotros sólo vendemos el software el gasto en hardaware corre por cuenta del cliente.

## 9.6.Seguro:

Se ha estimado también la contratación de un seguro, tanto del local como de todos los equipos de que disponemos, su valor sería de:

Precio del seguro: 200 € mensuales.

## 9.7.Estimación inicial:

Para calcular el total, antes de empezar con el proyecto hacemos una estimación de costes, por ello, sumando todo lo presupuestado previamente obtendremos que el gasto propio de un mes sería:

Gasto mensual en empleados + instalaciones + seguro.

1200€ + 575€+ 200€ = 1975€ mensuales

Una técnica muy utilizada para estimar los costes de un proyecto como dice la profesora Marcela Varas en los contenidos de la asignatura Gestión de Proyectos de Desarrollo de Sofware que imparte en la facultad de ingeniería de la Universidad de Concepción, consiste en realizar una estimación optimista (EO), otra más probable (EMP) y una pesimista (EP), y asignarle una probabilidad a cada una, obteniendo así nuestra estimación mediante:

E=EO *Po + EMP*Pmp + EP*Pp

En nuestro caso estimamos que lo mínimo que nos llevaría serían 5 meses con probabilidad de un 10%, y lo máximo 12 meses con probabilidad de un 30%, y seguramente tardaremos 10 meses. Por otro lado, lo mínimo que noscostará será 10.000€ con probabilidad 15%, lo máximo 25.000€ con probabilidad 20%, y lo más probable es que el costo sea de 20.000€. Con estos datos podemos obtener dos estimaciones:

Tiempo = 5 *0.1 + 12 * 0.3 + 10 * 0.6 =0.5 + 3.6 +6 = 10.1 meses
Coste = 10000 * 0.15 + 25000*0.2 + 20000 * 0.65 = 1500 + 5000 + 13000 = 19500€

## 9.8.Coste final:

Finalmente el proyecto nos llevó 9 meses con lo que la estimación fue bastante acertada en cuanto al tiempo. En cuanto al coste la estimación también fue acertada ya que si a la estimación mensual que habíamos hecho de unos 1975€ le sumamos la inversión en recursos informáticos que habíamos fijado en 5000€ tendríamos:

1975€/mes * 9 meses= 17775€ (cifra próxima a la presupuestada).

+ 5000€ (recursos informáticos) = 22775€

A nuestra "empresa" la primera versión del proyecto le costaría estos 22775€ pero cada cliente nuevo que tengamos no nos conllevará este gasto.

Esto quiere decir que si este proyecto lo vamos a vender a un único cliente a parte de la remuneración de nuestros honorarios como programodores buscaríamos un porcentaje de beneficio neto sobre el producto, por ejemplo un 60%.

El 60% de 22775€ son 13665€ con lo que venderíamos nuestro proyecto por unos 36440€ + 18% de IVA serían 42999.2€.

En cambio si se pretende que nuestra empresa venda este producto de forma continua buscando clientes nuevos, teniendo en cuenta que el gasto será muy inferior al inicial a partir de ahora se podría buscar un beneficio de un 10% sobre cada venta, por lo que tendríamos el siguiente precio por el producto.

10% de 22775€ serían 2277.5€ con lo que cada venta serían 25052.5€ + 18% de IVA serían 29561.95€

## 9.9.Enfoque del mercado

La venta del producto se enfoca en PYMES nacionales, ofreciendo un producto barato en comparación con otras propuestas y sobre todo innovador, hecho a la medida de una clientela que carece de una alternativa. La mayor parte del software que existe en esta rama del mercado es software hecho desde cero para un cliente particular, nosotros proponemos un software que se adapta a cualquier clínica de traumatología, pero que a la vez es suficientemente flexible para poder ser adaptado a una clínica de cualquier especialidad, ya que básicamente el funcionamiento es el mismo para cualquiera de ellas.

Teniendo esto último en cuenta y que una versión nueva para un cliente distinto con necesidades distintas nos llevaría mucho menos trabajo ya que ahora sabemos bien cómo hacerlo, nos permitiría rebajar cerca del 15% el precio final del producto.

# 10. Una historia para contar la funcionalidad de la aplicación

Una clínica de traumatología necesita tener al día los datos relacionados con sus pacientes y consultas, por ello, tras sondear el mercado decide que la opción que mejor se ajusta a sus necesidades es nuestro software, un software que satisface todas las necesidades que su consulta demanda y por un precio muy competitivo. GCT es una aplicación hecha a medida del cliente y el cliente rápidamente se da cuenta.

Una vez obtenido el software, el administrador de la clínica comienza a darle uso y se le instala el software en todos los ordenadores de la clínica.

El administrador es único y como tal debe ser él quien dé de alta a los demás empleados, para ello debe ejecutar la aplicación y loguearse como el administrador.


Fig.1. Login

Una vez logueado como administrador, este tiene acceso a la pestaña de empleados, en donde puede gestionar las altas y bajas de empleados.

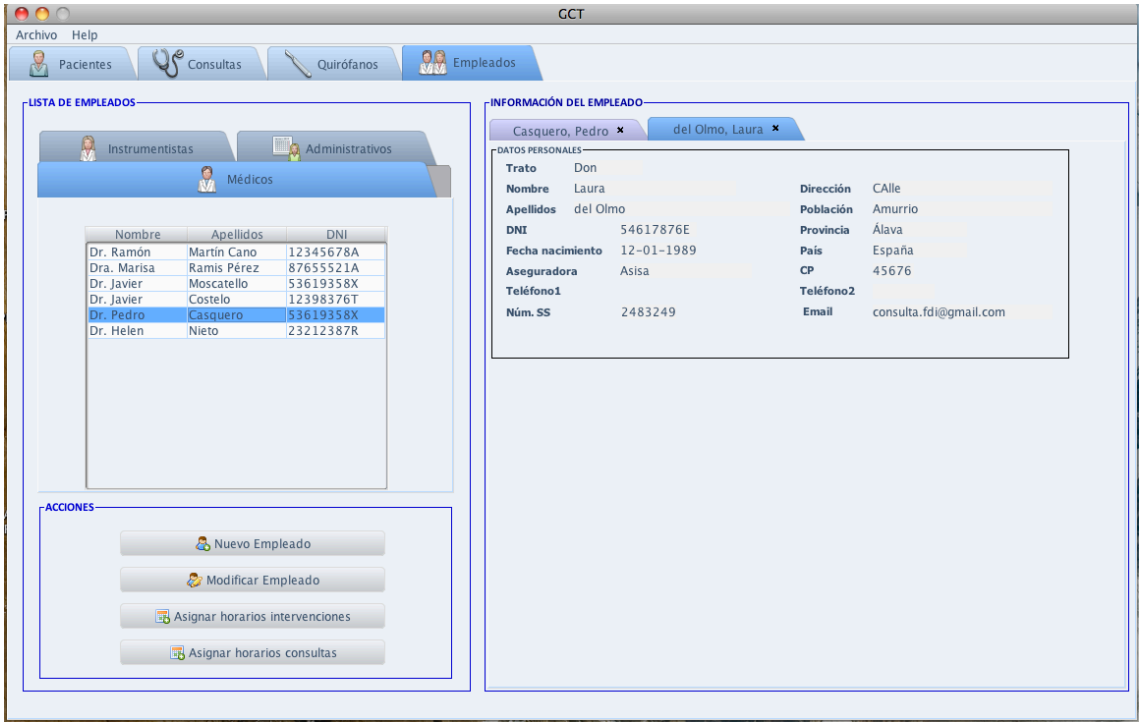



Fig.2. Información empleados.

Se le mostrarán en la parte izquierda de la aplicación tres pestañas en las que podrá seleccionar cada una de las tres categorías de empleados que hay en la clínica, situándose sobre cada una de ella, se mostrarán los empleados actuales de la categoría seleccionada. En el caso de la figura 2 se ha seleccionado la pestaña de médicos, pinchando sobre el nombre del médico que quiere consultar se abre en la parte derecha una pestaña con los datos asociados a este, podrá seleccionar tantos empleados como quiera que sus fichas se irán abriendo en pestañas en la parte derecha de la aplicación pudiendo cerrarlas en cualquier momento pinchando sobre el (x) de arriba a la derecha.

El administrador puede realizar una serie de acciones sobre el mismo, tales como asignarle horarios de consulta a los médicos.

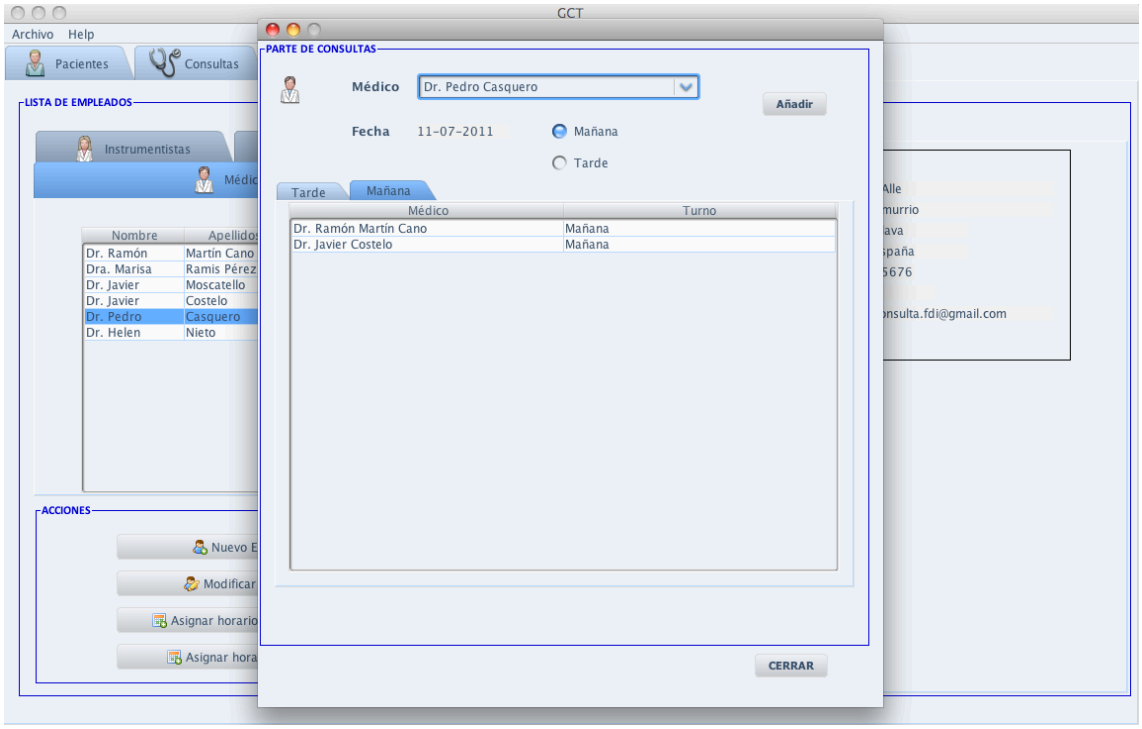



Fig.3. Asignar turnos de consultas

Al pulsar sobre el botón “asignar horarios consultas” se le muestra una ventana como la de la figura 3, en la que selecciona el médico al que le quiere asignar un horario, elige la fecha y el turno y pulsa el botón de añadir, en caso de que dicho medico tenga asignado un horario de quirófano en ese mismo día y turno se mostrará un mensaje de error en el que se le advertirá de que esto sucede, en caso de que sea una fecha y un turno en el que el médico no tiene asignado ningún horario le quedaría asignado.

Si lo que el administrador quiere es asignar un nuevo horario de intervenciones a un médico lo que hará será pinchar sobre el botón “asignar horario intervenciones” y realizará un proceso análogo al anterior (figura 4).

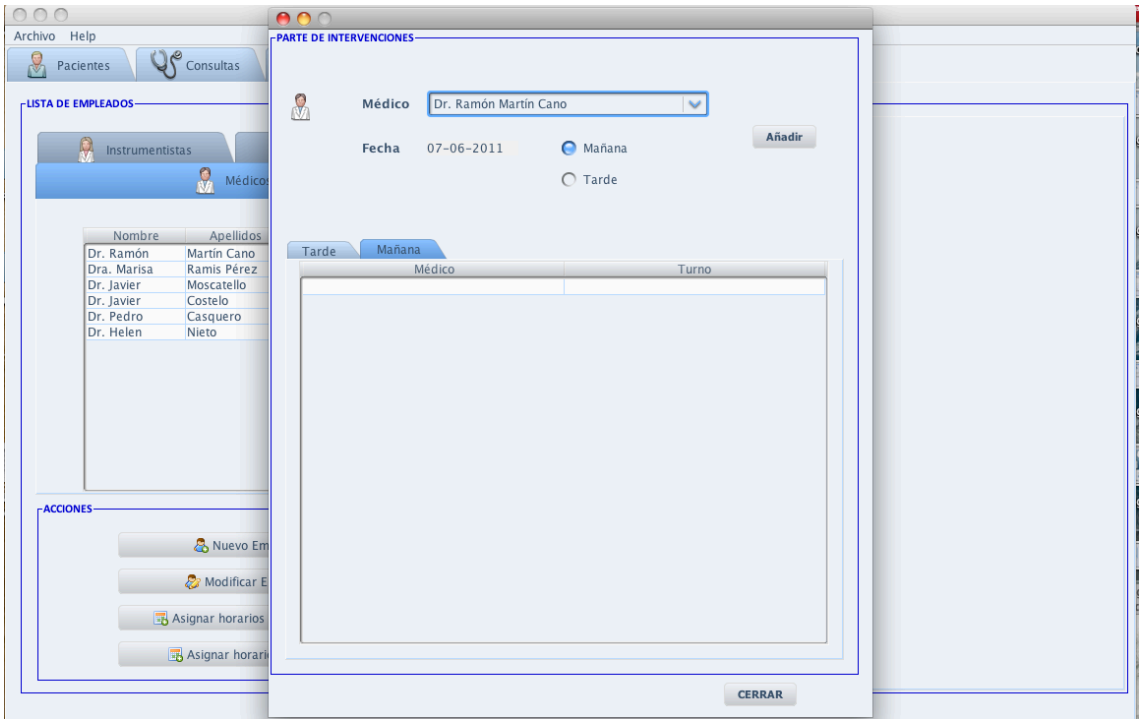



Fig.4. Asignar turnos de intervenciones

La clínica cuenta con un nuevo empleado, con lo que el administrador necesita darlo de alta, para ello lo único que tendría que hacer sería pinchar sobre el botón “nuevo empleado” y rellenar los datos del mismo marcando el tipo de empleado del que se trata (figura 5).

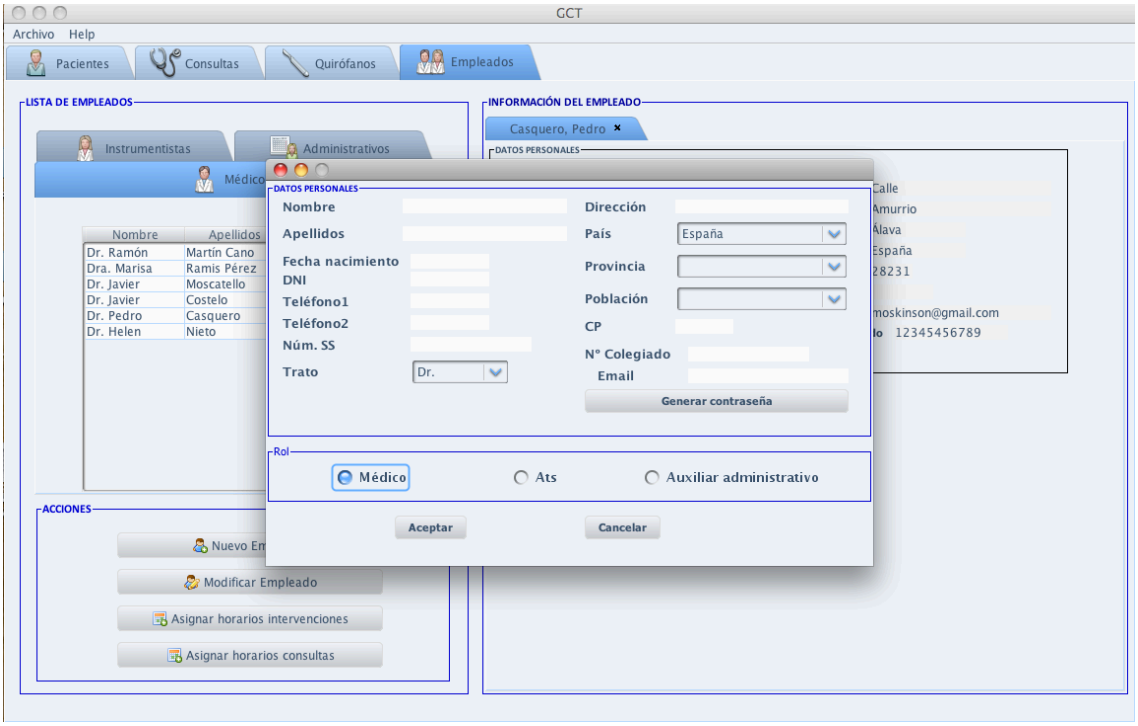



Fig.5. Nuevo empleado

En cambio si se deja de contar con los servicios de un empleado, el administrador deberá darlo de baja, para ello lo único que deberá hacer es seleccionar dicho empleado y pichar sobre el botón “eliminar empleado” y confirmar que desea eliminar dicho empleado.

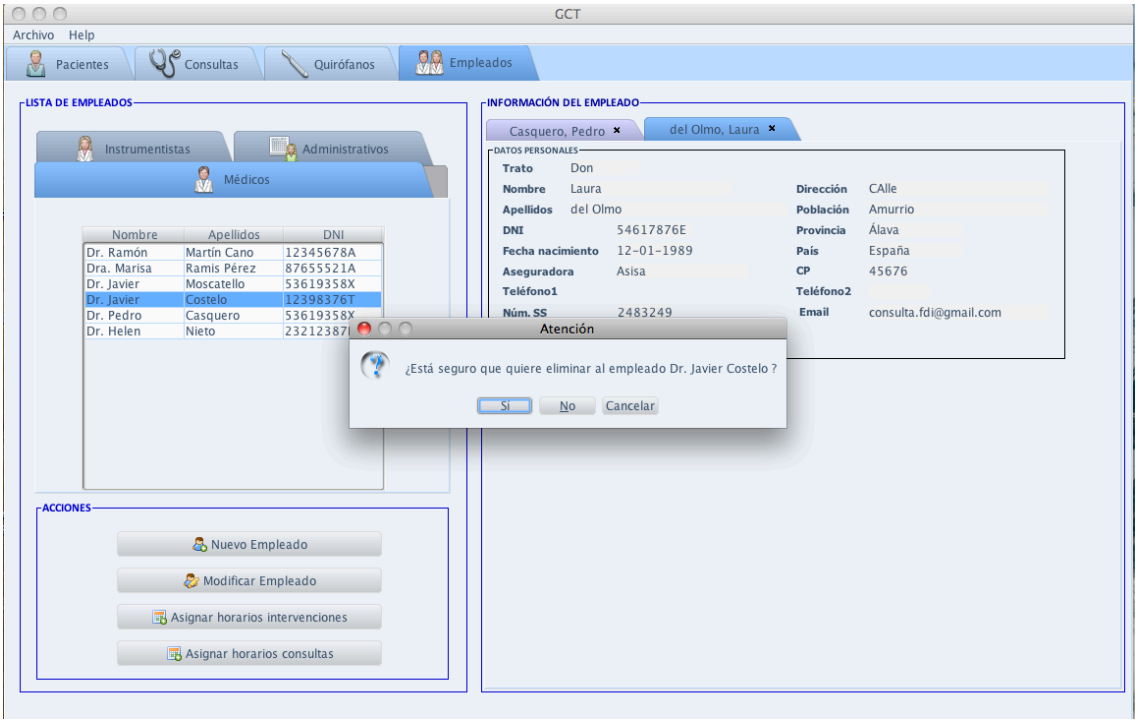



Fig.6. Eliminar empleado

El administrador ya ha dado de alta a los empleados en la aplicación y estos ya la usan a diario, tanto los auxiliares administrativos como los médicos.

Para un auxiliar administrativo la adquisición de nuestra aplicación es una gran noticia, facilita de manera notoria su trabajo, veamos una secuencia de hechos cotidianos en el trabajo del auxiliar para ilustrarlo.

Lunes por la mañana, empieza una nueva semana y Eva la auxiliar administrativa que trabaja por las mañanas en la clínica se sienta en su mesa para empezar con su labor, arranca sesión y abre la aplicación logueandose con su perfil.


Fig.7. Login auxiliar administrativo

Una vez que Eva se ha logueado, la aplicación se abre con la vista propia de un perfil de auxiliar administrativo y Eva está lista para empezar a trabajar, pronto llama un paciente que quiere pedir cita, Eva lo tiene fácil no tiene mas que consultar al paciente si ya ha venido previamente a la consulta o si se trata de la primera vez que acude a la clínica, el paciente contesta que no es la primera vez, que ya ha venido antes, en ese caso Eva le pregunta su nombre o su dni y busca la ficha del paciente en la pestaña de pacientes.

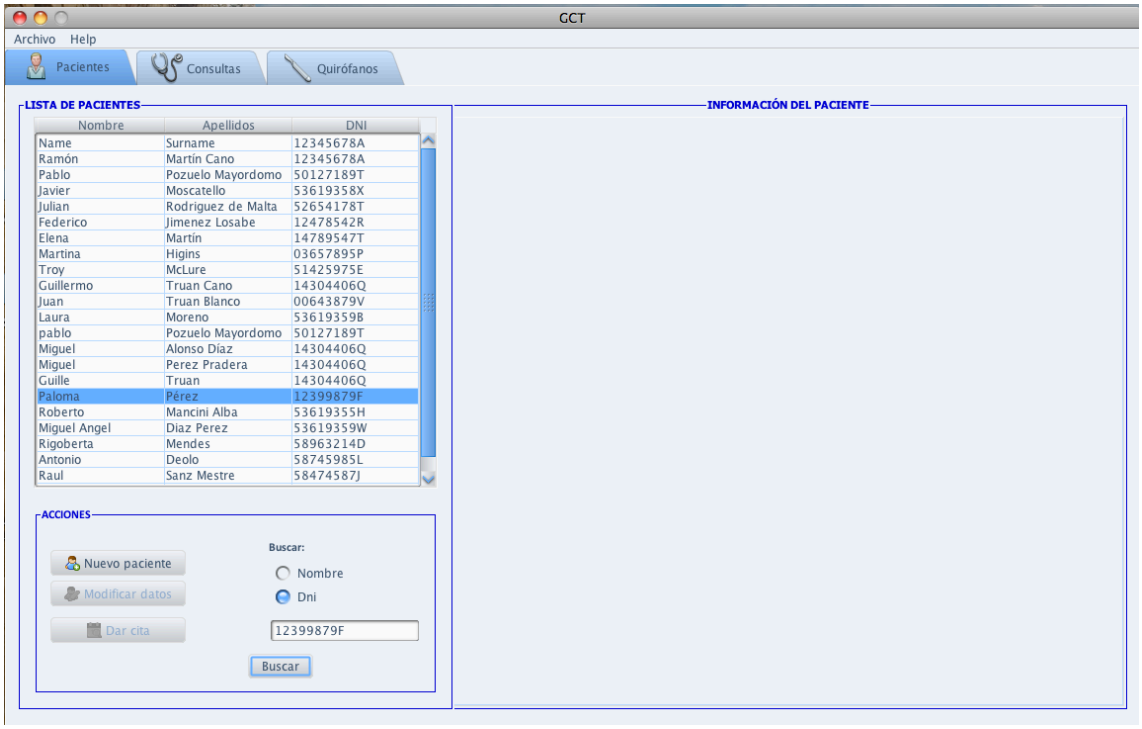



Fig.8. Buscar paciente

Eva ya ha localizado al paciente que efectivamente está ya en la base de datos, consulta su ficha y ve un pequeño resumen de la misma, en la que puede ver las ultimas consultas que ha tenido, sus ultimas intervenciones, sus datos personales y la fecha de la próxima consulta y el estado de su próxima intervención en caso de que tenga alguna programada. En este caso Eva ve que no tiene ninguna programada y le pregunta para cuando quiere consulta, el paciente contesta que la quiere para una fecha determinada y con un médico determinado, entonces Eva consulta en la aplicación si ese día el médico que el paciente quiere tiene consulta o no, en caso de que así sea, se mira la disponibilidad de horarios y tras preguntar al paciente se acuerda que sea a las 11.00, hora que el médico tiene libre.

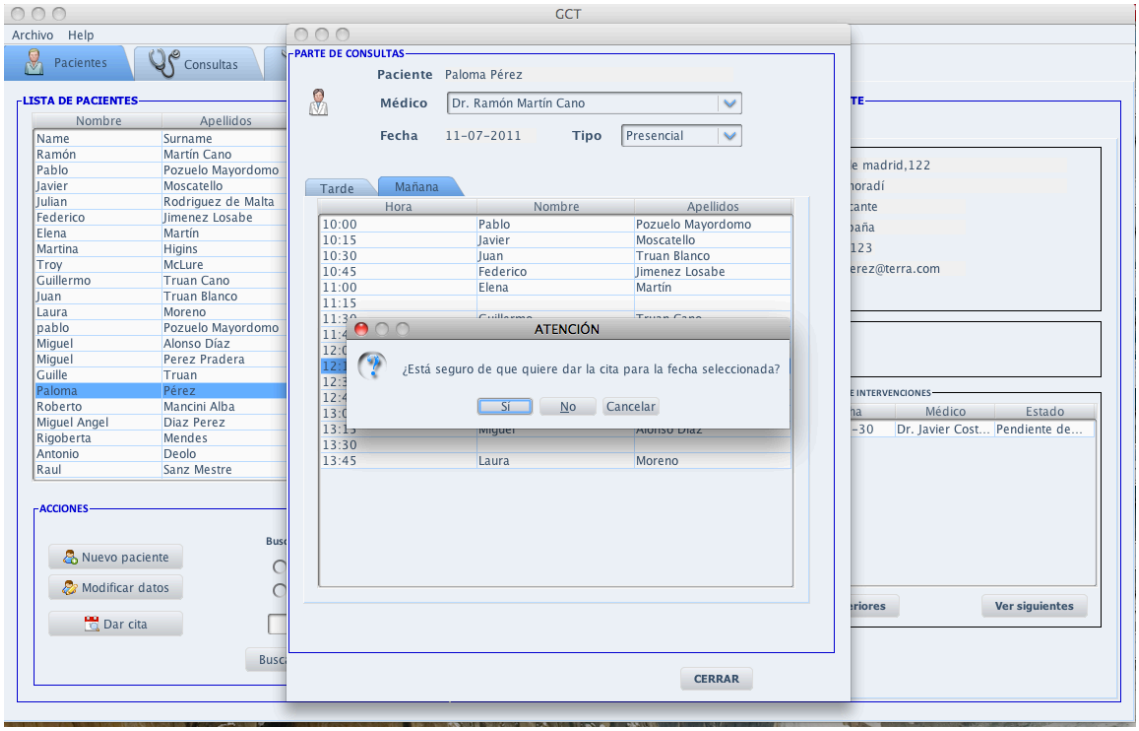



Fig.9. Dar cita

El paciente ya tiene su cita programada y Eva puede ver en la ficha del mismo que esta ha sido adjudicada ya que ahora consta como la próxima intervención en la ficha resumen del paciente.

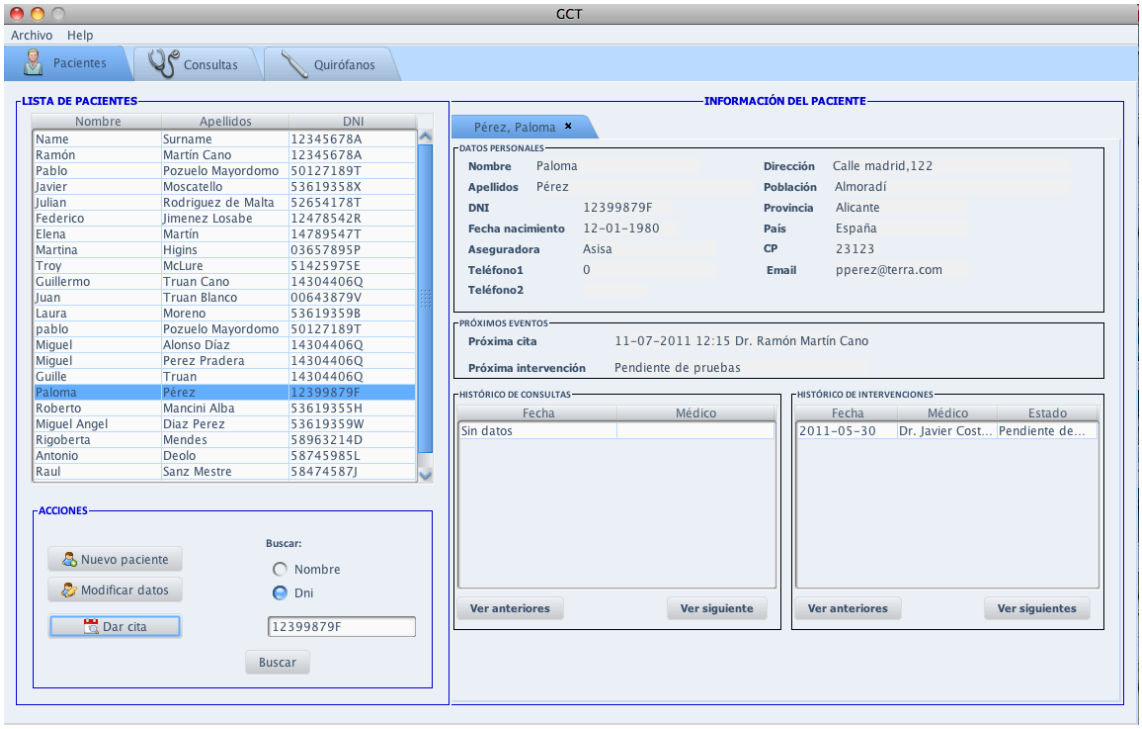



Fig.10. Estado próxima cita

La mañana continua y acude a la consulta un nuevo paciente que no había venido previamente, en este caso Eva lo que hace es crear una nueva ficha para dicho paciente, para ello le pregunta sus datos personales y tras hablar con él y ver que día y/o que médico le interesan mas, se le asigna una cita, su historial en nuestra clínica comienza en este momento, su ficha ya se ha creado y poco a poco se irá rellenando.

El teléfono de Eva vuelve a sonar, se trata de un paciente que dice que tenía cita para la semana siguiente, pero que no recuerda la fecha exacta, en este caso Eva lo tiene fácil, le pregunta al paciente por su nombre o dni y le busca en la base de datos, rápidamente le encuentra y consulta su ficha, en ella ve, en el apartado de “próxima cita” que efectivamente el paciente tiene cita para la semana siguiente y le recuerda al paciente los detalles de la

misma.

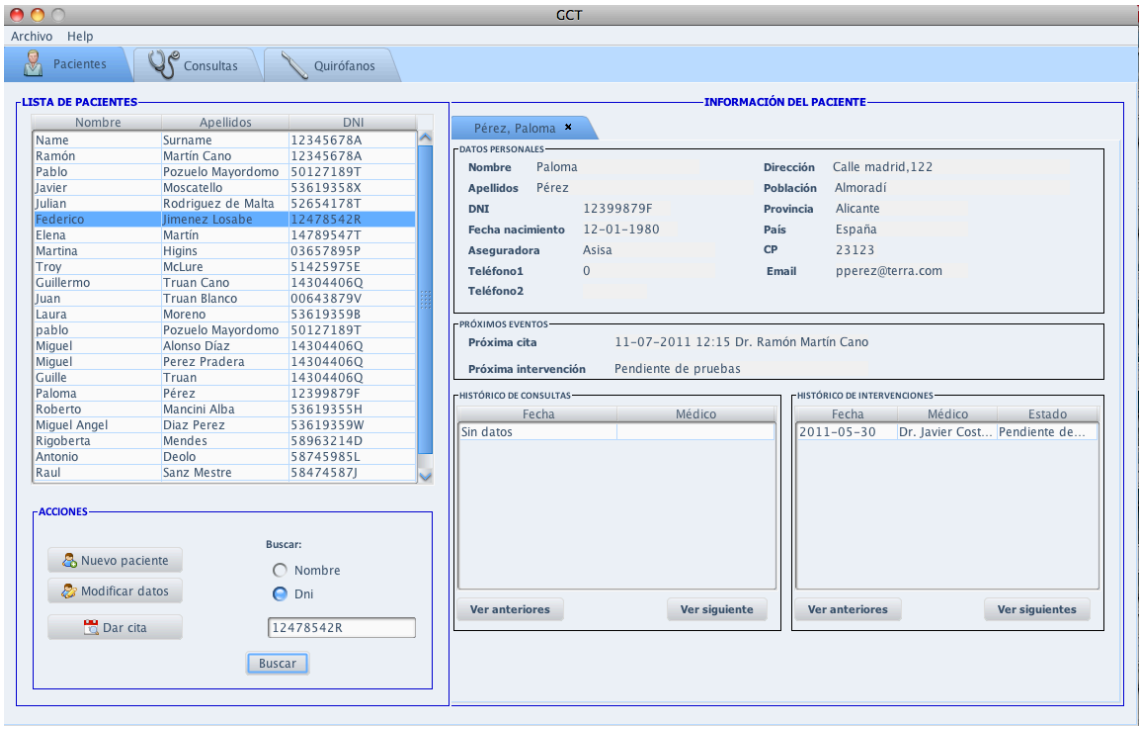



Fig.11. Búsqueda de paciente

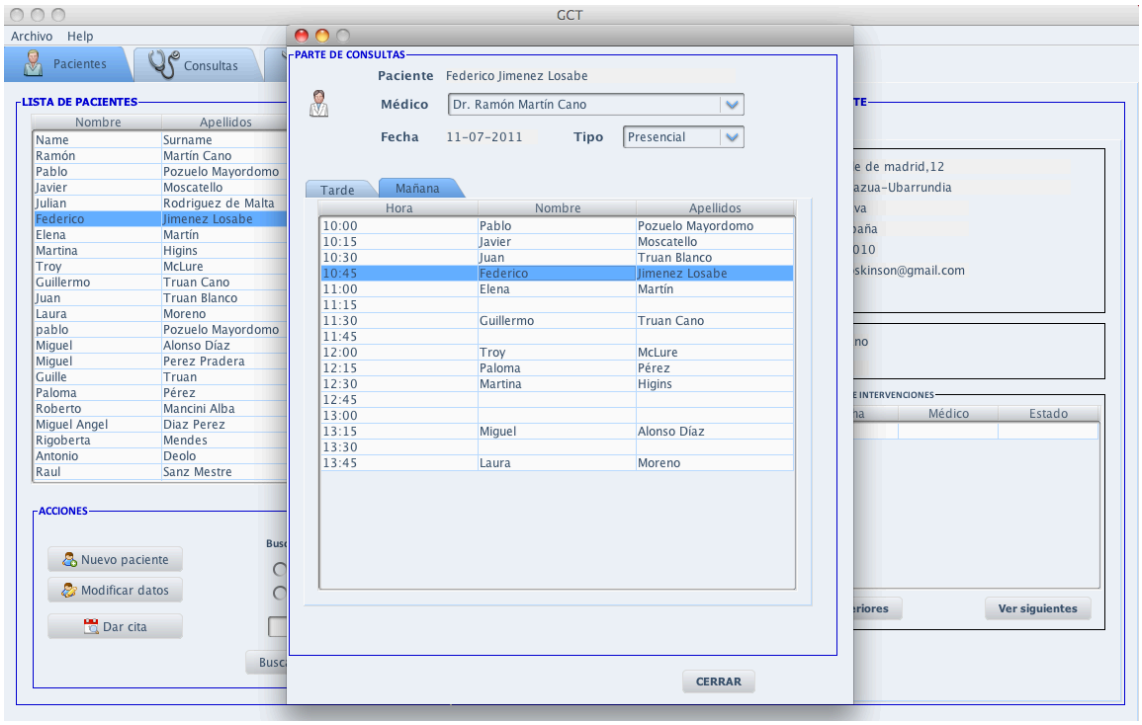



Fig.12. Dar cita

Más tarde llama un paciente advirtiendo a Eva de que tenía cita para el día siguiente pero que no va a poder asistir, Eva le comenta que no es problema y consulta la ficha del paciente, ve que en efecto este tenía una cita para el día siguiente, entonces Eva mira el médico que se le había asignado a dicho paciente y se va a la pestaña “consultas” en donde selecciona al médico y la fecha en la que el paciente tenía programada la cita y la cancela(figura), Eva le agradece que haya avisado de que no va a asistir finalmente y le pregunta si quiere cita para otro momento con el mismo doctor o incluso con otro si es que así lo prefiere, el paciente contesta que si, que le gustaría poder asisitir el mismo día pero de la semana siguiente entonces Eva consulta el parte de consultas de ese día para ese médico y le dice al paciente que podría ser a las horas que constan como libres en la aplicación, el paciente elige una de estas horas libres y se le asigna un nueva cita.

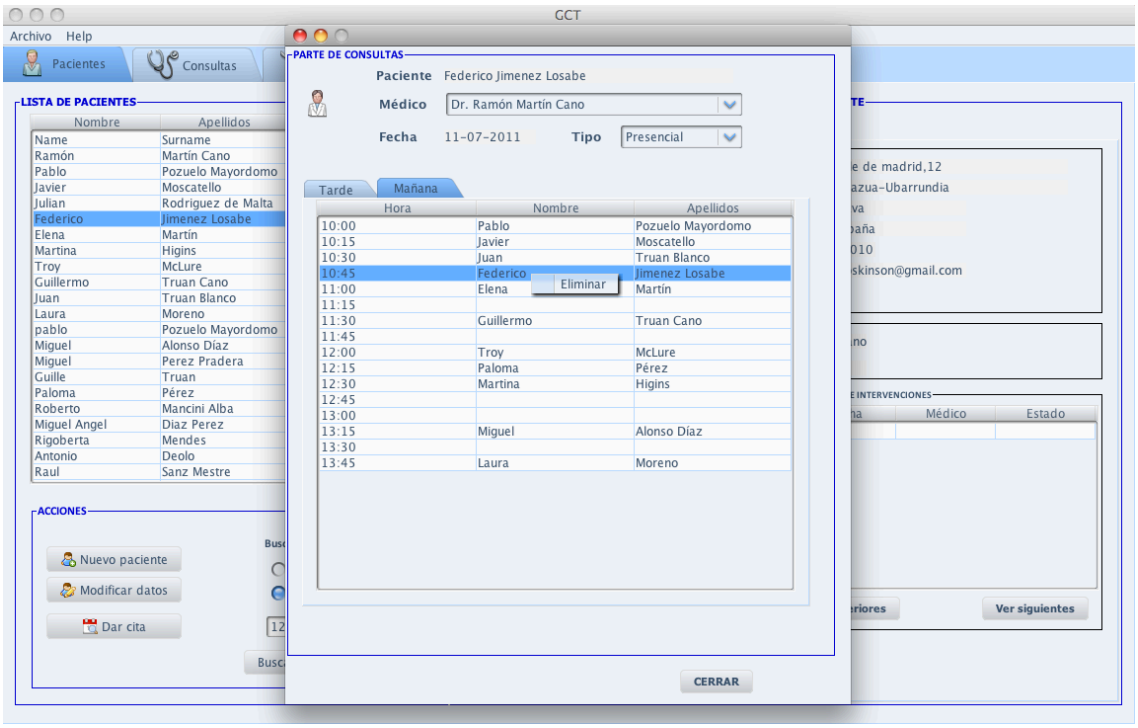



Fig.13. Cancelar cita

Durante la mañana se suceden las llamadas de pacientes pidiendo citas y poco a poco el parte de consultas se va llenando y el volumen de datos en la aplicación va creciendo.

Son cerca de las 14.00 y el turno de Eva va llegando a su fin, cuando un nuevo paciente llama para confirmar la fecha de la operación a la que va a ser sometido ya que no recuerda bien qué día era, Eva no tiene mas que consultar la ficha del paciente y mirar el apartado “próxima intervención” y así rápidamente resolver la duda del paciente.

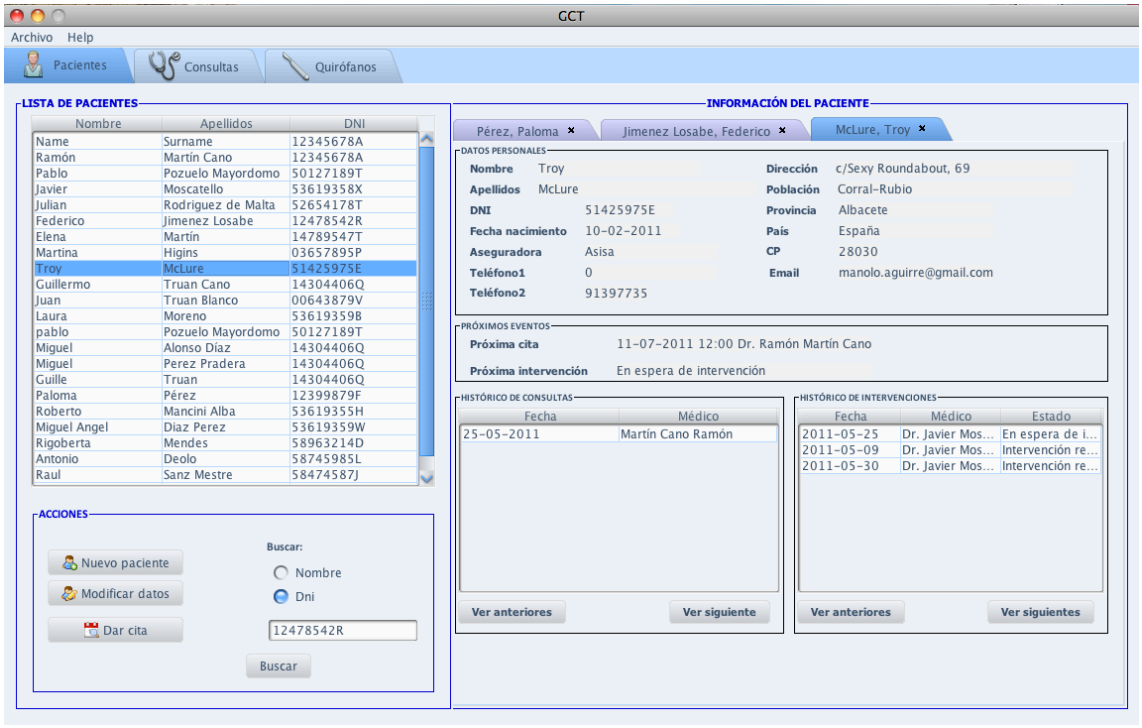



Fig.14. Estado próxima intervención

El turno de Eva ha terminado y viene Marisa la auxiliar administrativa que tiene turno de tarde, Eva cierra su sesión en la aplicación (Fig. cerrar sesión) y Marisa abre la suya, cuando suena el teléfono, se trata del jefe, que en este momento no tiene conexión a internet y por tanto no tiene acceso a la parte web de la aplicación pero necesita saber como está el parte de quirófano para el día siguiente, entonces Marisa rápidamente accede a la pestaña de "quirófanos" y ve lo que hay programado para ese día, se lo comenta al jefe y este ya está informado y puede planificar el día de trabajo.

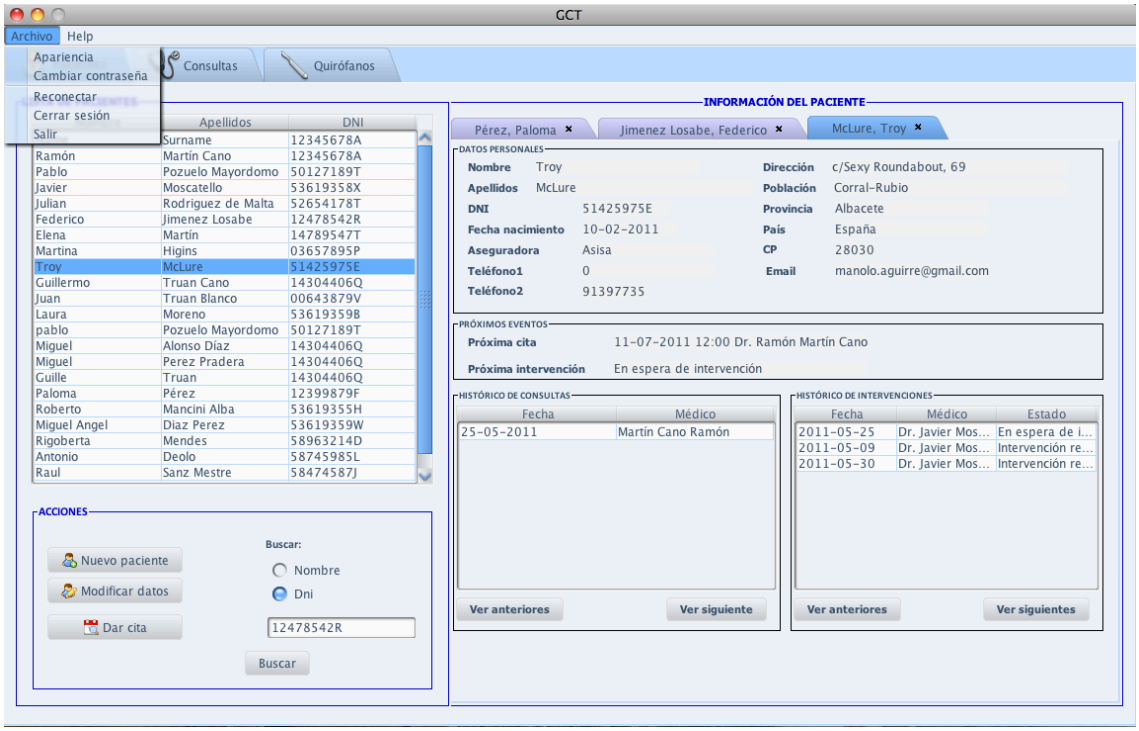



Fig.15. Cerrar sesión

Es domingo y el doctor Ramón Martín Cano, sabe que el día siguiente se prevé duro, tiene quirófano por la mañana y consulta por la tarde, pero le gustaría buscar un hueco para comer con su mujer, sabe que tiene que estar en la consulta a las 16.00, pero no recuerda muy bien cómo está de cargado el parte de quirófano para ese lunes, entonces como él ya está registrado en la parte web de la aplicación entra en ella y consulta el parte de quirófano para esa fecha y comprueba que la mañana esta muy cargada y que seguramente no tenga tiempo para comer con su mujer como le habría gustado, ya que está logueado dentro de la aplicación decide consultar el estado de la consulta también para ver cuantos pacientes están citados y sobre todo para comprobar que tal y como suponía mañana está citado un paciente cuya evolución le tiene especialmente preocupado.


Fig.16. Listado de consultas a través de la web.

El lunes ha llegado y el doctor Ramón Martín Cano entra en quirófano listo para comenzar con el trabajo, en el quirófano hay un ordenador en el cual está la aplicación abierta y uno de los médicos que hoy están en quirófano ya se ha logueado. En la pestaña “quirófanos” seleccionan la fecha de hoy y ven todo el parte del día, pinchando sobre el primer paciente que consta en el parte se abre a la derecha su ficha correspondiente en la que el doctor Ramón Martín Cano puede ver los detalles de la intervención, una vez se ha realizado la intervención el doctor Ramón Martín Cano pincha en “editar intervención” y procede a rellenar los campos que queden por rellenar tales como las observaciones propias de la intervención o los empleados que la han llevado a cabo, tanto médicos como ATS y finalmente marca al paciente como ya intervenido, quedando así todos los detalles de la intervención ya realizada recogidos en la

aplicación.

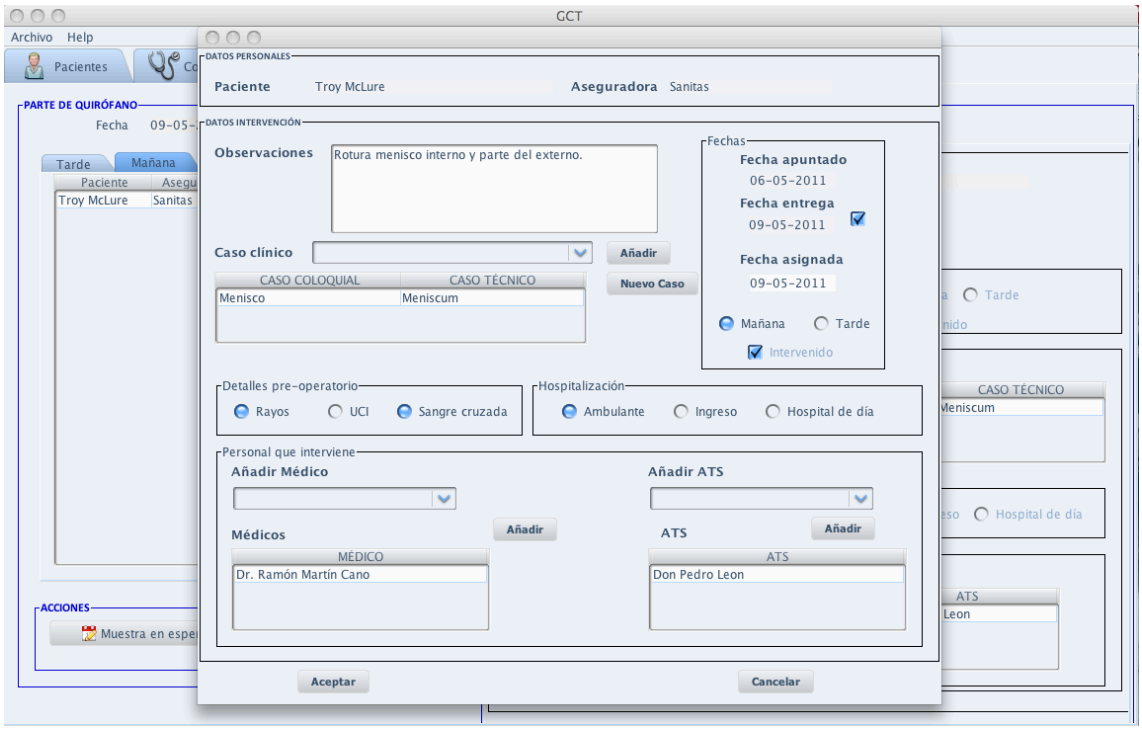



Fig.17. Editar intervención

Mientras tanto la mañana continua y la doctora Marisa Ramis Pérez está pasando consulta, ella también tiene la aplicación abierta y esta logueada con su perfil, tiene abierta la pestaña “consultas” y el día de hoy seleccionado, se le muestra una lista con todos los pacientes (captura) que tiene citados, se acaba de marchar un paciente y es el turno del siguiente, este entra y la doctora le saluda y le pregunta su nombre, el paciente responde y ella comprueba de que se trata de su siguiente paciente. El paciente acude porque según dice fue operado unos meses atrás en nuestra clínica y todo va bien pero de vez en cuando siente algún dolor en determinadas posiciones, la doctora comprueba en su ficha que así es que el paciente fue operado por el doctor Martín Cano (fig. histórico intervenciones) y tras ver los detalles de la intervención determina que se tratan de secuelas propias de la intervención y que poco a poco irán remitiendo, apunta los detalles de la consulta pulsando sobre el botón “nueva consulta” y emplaza al paciente a volver pasados unos meses para ver como evoluciona.

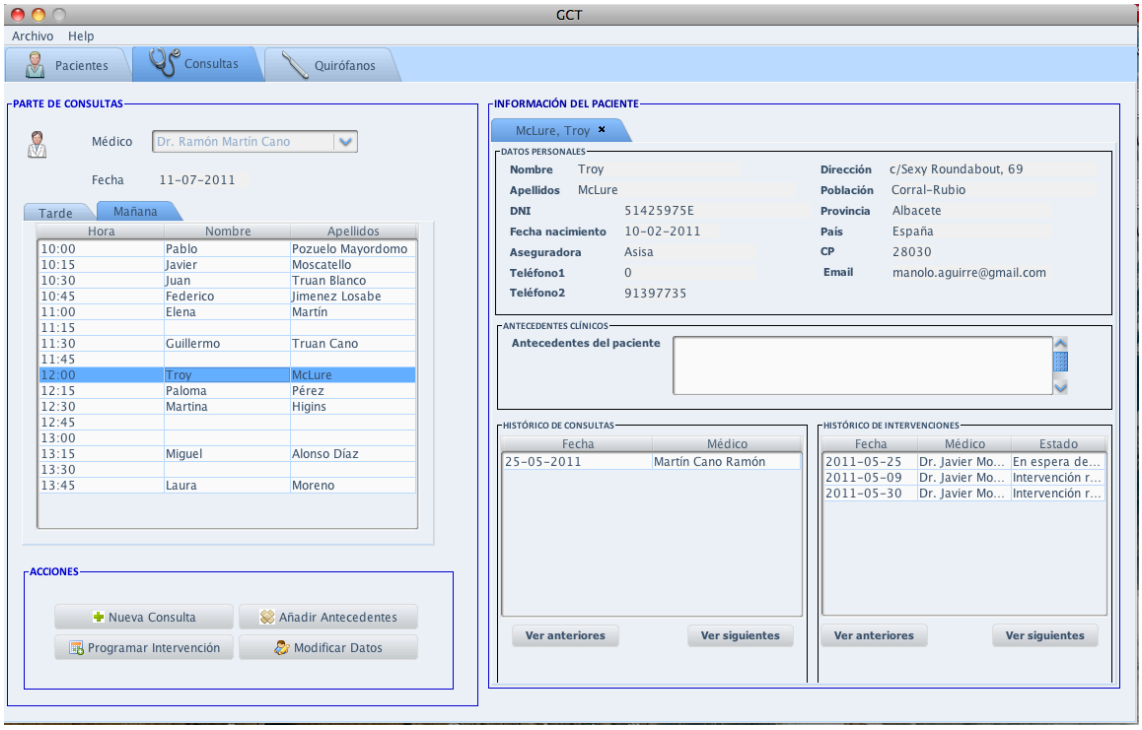



Fig.18. Histórico intervenciones y consultas para un médico en una fecha.

El siguiente paciente entra en la consulta, se trata de un nuevo paciente, es la primera vez que acude a nuestra clínica y la doctora lo ve en su ficha, por lo que antes de explorarle le pregunta por sus antecedentes, si toma alguna medicación o si sufre alguna alergia para apuntarlo en su ficha, el paciente contesta que toma una medicación tras un infarto que tuvo, la doctora pulsa sobre el botón “añadir antecedentes” y lo hace constar en su ficha (Fig. antecedentes). El paciente acude por un dolor en el muslo que la doctora rápidamente identifica como una pequeña rotura fibrilar y le recomienda reposo durante unos días, lo apunta en la ficha y se despide del paciente.

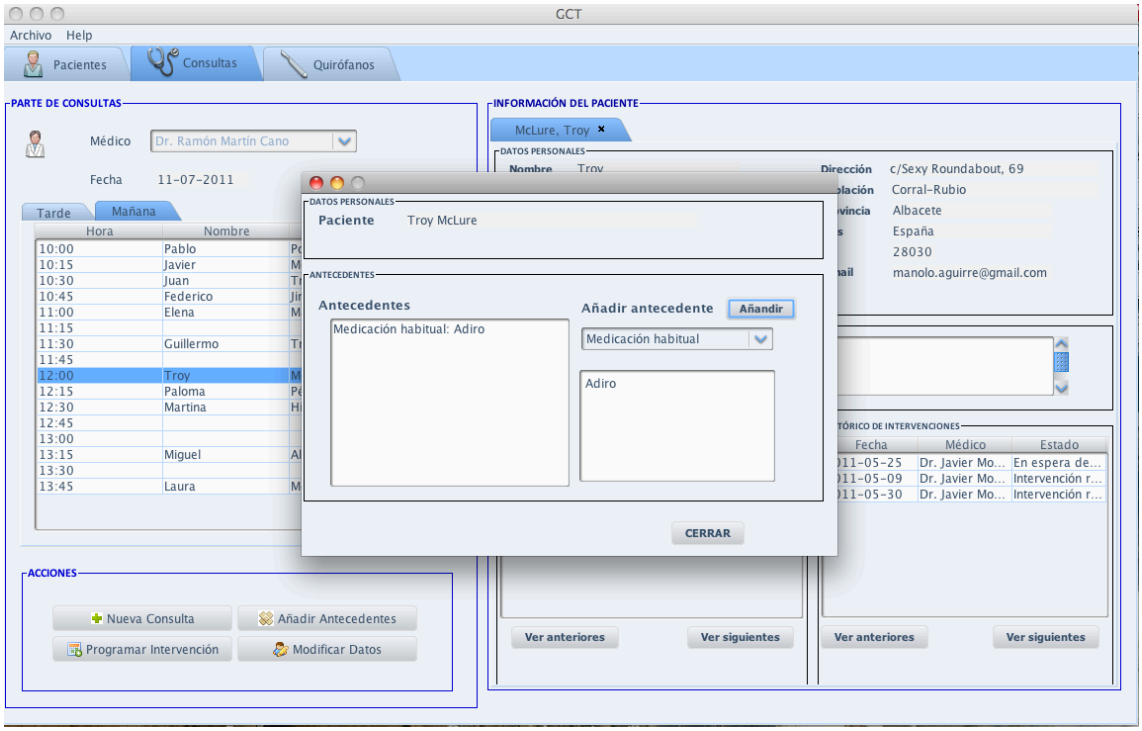



Fig.19.Ficha de Antecedentes

El siguiente paciente entra en la consulta y tras preguntarle su nombre la doctora comprueba que no se trata de su siguiente paciente si no de el que tenía citado después, da por hecho que el paciente cuyo turno era el actual finalmente no ha asistido y procede a pinchar sobre el nombre del paciente que tiene delante abriéndose así la ficha del paciente, ahora la doctora tiene delante de ella toda la ficha del paciente que tenemos recogida en nuestra clínica.

El paciente asegura que tiene un fuerte dolor en la rodilla, la doctora tras explorarle determina que es probable que tenga el menisco dañado y que tal vez haya que operarle, apunta esto en la consulta de hoy pinchando sobre el botón “nueva consulta” y manda al paciente que se haga una resonancia magnética y una radiografía y que cuando lo tenga pida cita otra vez, el paciente así lo hace.

Ha pasado una semana desde que el paciente vino por primera vez y hoy vuelve a acudir a la consulta ya con las pruebas que la doctora le había pedido, éste entra en la consulta y así se lo hace saber a la doctora que no obstante consulta en la aplicación la ficha del paciente y ve que así es. Tras consultar las pruebas la doctora se da cuenta de que finalmente va a tener que ser operado, apunta la incidencia de la consulta y comenta con el paciente que necesitará ser operado a lo que el paciente contesta que siendo así le gustaría que fuese lo más rápido posible.

La doctora entonces se dispone a programar la intervención, para ello pincha sobre el botón “programar intervención” poniéndolo en espera a falta de que el paciente realice las pruebas necesarias antes de ser operado y ordena al paciente que pida cita para ellas y que después vuelva con los resultados a la consulta, el paciente así lo hace y vuelve a los pocos días con los resultado que en este caso son positivos, entonces la doctora programa la fecha de la intervención marcando el dia de hoy como el dia de entrega de las pruebas y le asigna una fecha a la intervención rellenando los campos de observaciones y tipo de intervención a la que ha de ser sometido. El paciente ya tiene fecha para ser operado y se marcha a su casa en espera de que llegue el día de su intervención.

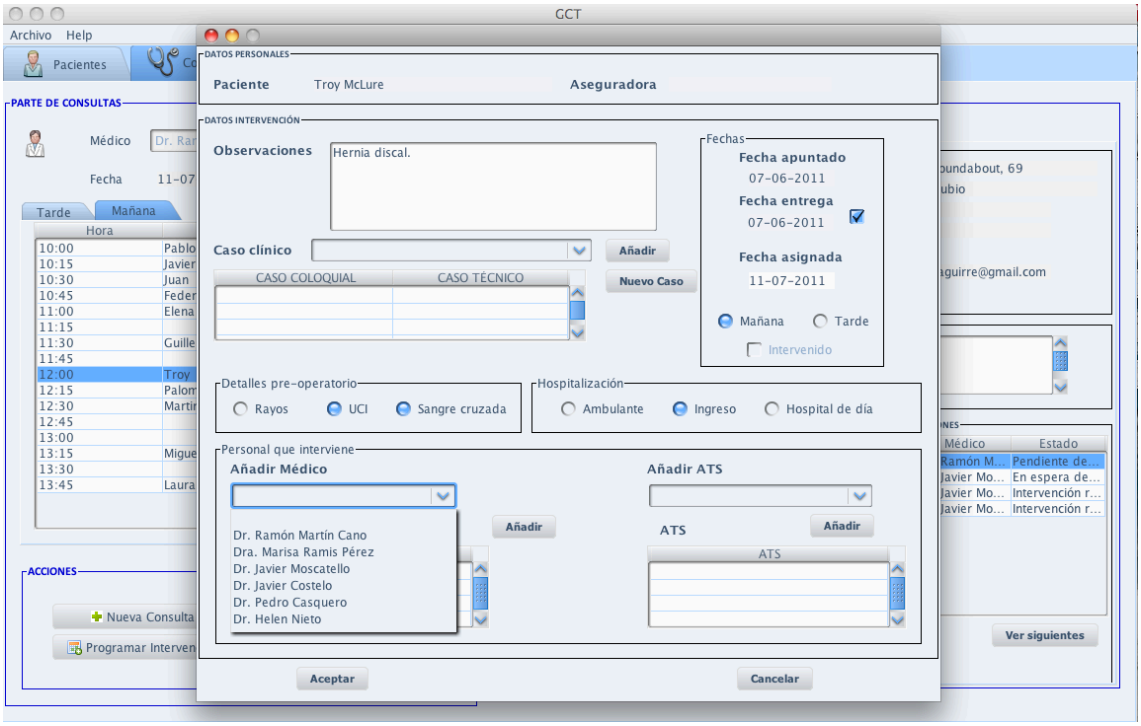



Fig.20. Programar intervención.

Ilustraemos ahora el uso de la parte web de la aplicación para los pacientes.

Miguel ha entrado en la página web de la clínica advertido por un compañero de trabajo de la posibilidad que esta ofrece de registrase en línea y tener acceso tanto a la solicitud de citas como a la consulta de la ficha personal que la clínica tiene de él.

Para ello, una vez dentro pincha sobre el botón “registrarse” abriéndose un formulario que debe rellenar, en él rellena por un lado sus datos personales y por otro sus datos como usuario de nuestro sistema.


Fig.21. Formulario de registro.

Tras rellenar correctamente todos los campos y marcar que ha leído y acepta la política de privacidad de nuestra web se le manda al correo que nos ha proporcionado un link de confirmación, el cual debe clicar para poder ser dado de alta en el sistema.

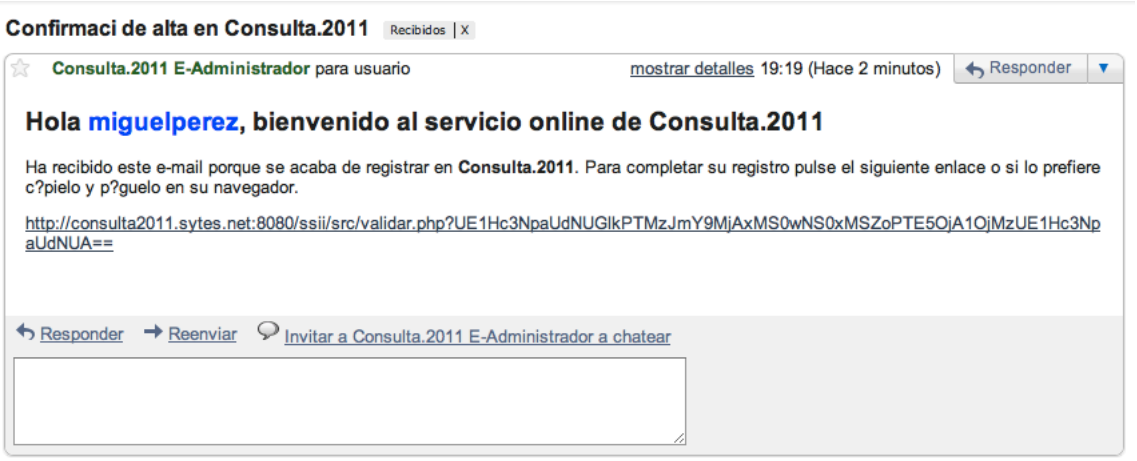



Fig.22. e-Mail de confirmación.

Tras clicar sobre el link se le redirecciona a una página en la que se le advierte de que su cuenta ha sido activada.


Fig.23. Aviso de cuenta activada correctamente.

Una vez que Miguel ya se ha registrado ya tiene acceso a nuestro sistema y hace uso de él, en la página web hace login como usuario registrado con los datos que él ha elegido.

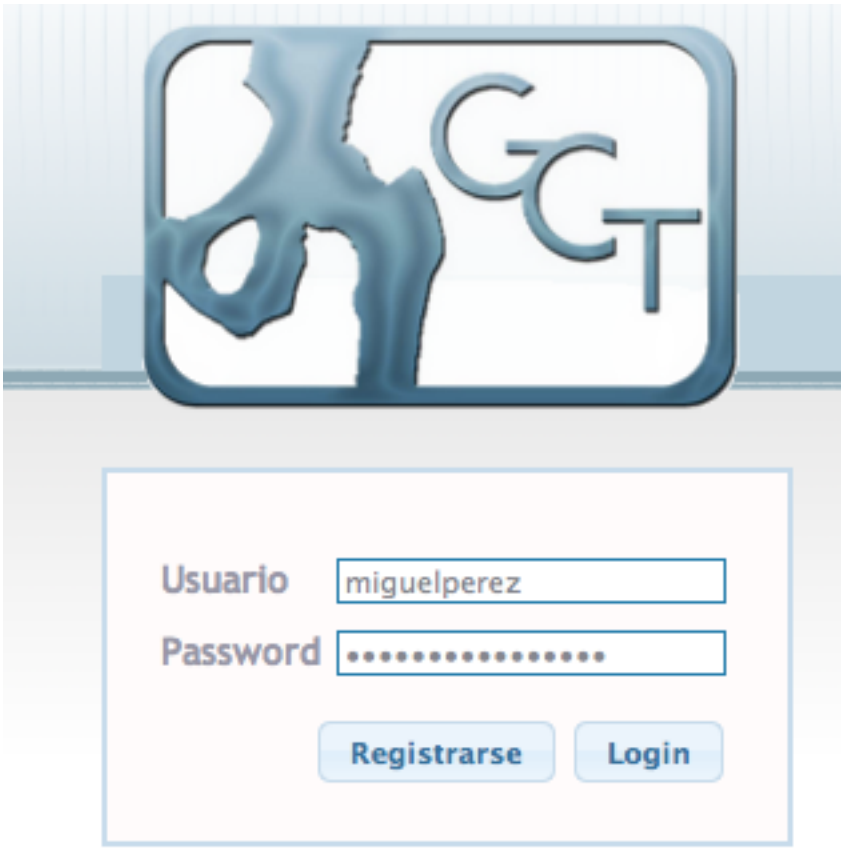



Fig.24. Login de usuario.

Una vez dentro tiene acceso a todas las posibilidades que nuestra web le propone.


Fig.25. Pantalla de inicio de usuario web.

Miguel lo que quiere es pedir consulta, para ello pincha sobre el botón “pedir cita” del menú de la izquierda y se le abre un submenú en el que puede elegir tanto medico como turno de mañana o tarde y la fecha en la que quiere ser atendido, en este caso como Miguel nunca ha estado en la consulta no tiene ningún tipo de preferencia deja seleccionado “cualquier médico” y marca el turno de mañana y selecciona una fecha, a la hora de seleccionar fecha solo podrá elegir entre los días que haya consulta y que la pase el médico seleccionado, en este caso cualquiera.


Fig.26. Pedir cita via web, selección de médico, turno y fecha.

Una vez seleccionado todo se le abre automáticamente un listado de las horas libres que tiene para ese día con el médico que ha seleccionado.

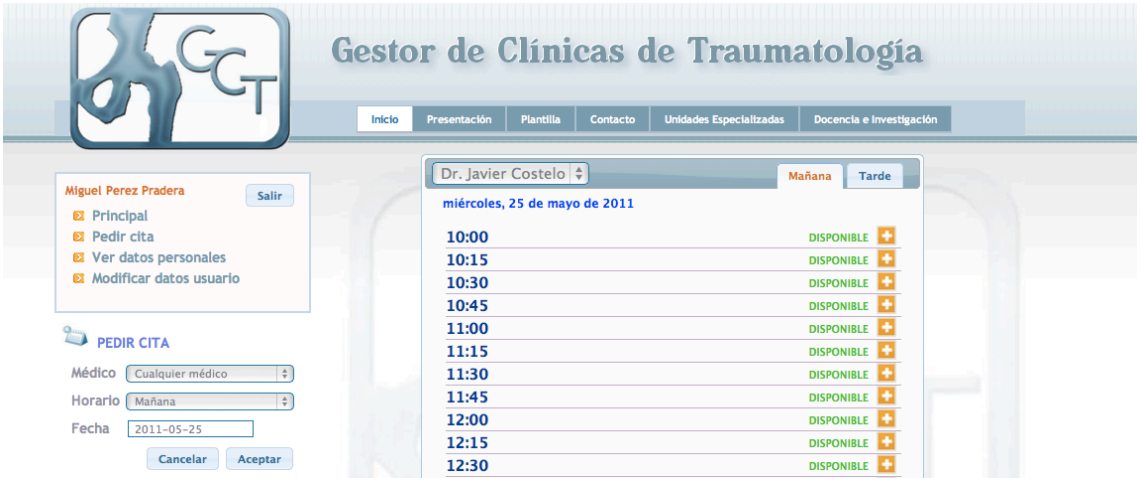



Fig.27. Pedir cita via web, selección de horario.

Miguel elige una que se le pedirá confirmar y automáticamente quedará asignada a él.


Fig.28. Pedir cita via web, confirmación de datos.

En la pantalla principal podrá consultar la fecha de su próxima consulta, así como el histórico de las mismas y el estado de su próxima intervención en caso de que tenga alguna programada.

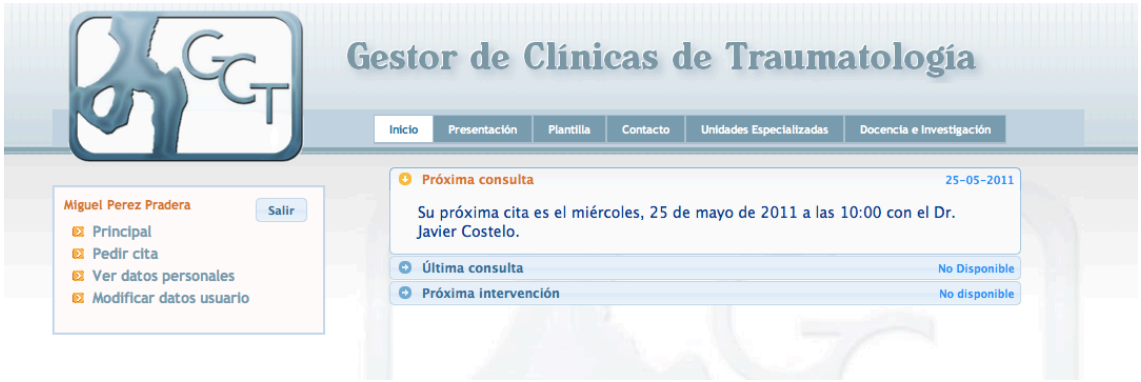



Fig.29. Consultar próxima consulta via web.

En cualquier momento Miguel podrá modificar sus datos personales pinchando sobre la opción correspondiente en el menú lateral.


Fig. 29. Modificar datos personales via web.

Asi como sus datos de usuario, de tal manera que puede cambiar su contraseña de acceso.


Fig. 30. Modificar datos de acceso a la aplicación web.

Otro usuario que ya ha hecho uso de la aplicación como es el caso de Pablo, puede ver en el apartado “ultima consulta” un resumen de sus últimas consultas en nuestra clínica


Fig.31. Consultar últimas consultas.

# 11. La aplicación web

## 11.1. Descripción

La aplicación Web permite a los pacientes consultar directamente su historial médico desde cualquier lugar, asi como las últimas intervenciones que se han realizado.

Actualmente la página Web se encuentra alojada en el dominio Consulta2011.sytes.net , correspondiente a una clínica hipotética. Si una clínica se hace con nuestro sistema deberá comprar un dominio en el cual alojar la aplicación web de nuestro sistema.

## 11.2. Funcionalidad detallada

Tal como se ha adelantado en el apartado anterior, la funcionalidad completa que ofrece la página Web para cada uno de los pacientes es la siguiente:

**-Registro**: Un paciente puede registrarse en nuestra aplicación web rellanando sus datos personales y así poder hacer uso de las ventajas de la aplicación web.

**-Log-in:** Cada paciente dispondrá de un identificador de usuario y una contraseña para entrar enla página Web. El usuario ha tenido que darse de alta en el sistema o bien a través de la propia página web o en la consulta.

-**Datos personales**: El paciente puede consultar sus datos personales en la web y modificar sus datos de acceso a la misma. Si desea modificar sus datos personales, esto debe hacerse a través de la aplicación de la consulta, es decir deberá ponerse en contacto con la clínica para solicitar que le sean modificados.

-**Pedir cita**: Se le da al paciente la opción de elegir un médico y un turno y en función de la elección del paciente se le muestran las fechas en las que podría tener consulta según esos criterios. Una vez seleccionada la fecha se despliega la agenda de ese día y turno pudiendo elegir el paciente cualquiera de las horas que queden libres.

-**Próxima consulta**: Muestra al paciente la fecha, hora y médico de la próxima consulta que tiene programada.

-**Última consulta**: Muestra al paciente un registro de las últimas consultas que hemos tenido, con la fecha, la hora y le médico que le atendió.

-**Próxima consulta**: Muestra al paciente la fecha y el médico de la próxima intervención que tiene programada.

## 11.3. Ley de Protección de Datos

La página Web se encuentra realizada en PHP 5, lo cual nos ha permitido una conexión directa con la Base de Datos, utilizando la extensión mysqli, disponible a partir de esta versión de PHP. El uso de la extensión mysqli nos facilita el control de los posibles ataques que pueda recibir la web a través de inyección SQL.

Utilizamos la función de encriptación SHA1(), propia de MySQL, para encriptar el password del usuario. El uso de esta función nos proporciona un elevado nivel de seguridad sobre la contraseña. Además, obligaremos al usuario a utilizar contraseñas con un alto índice de complejidad para protegernos contra ataques sobrer este tipo de encriptado. Esta metodología de encriptado no permite la recuperación de la contraseña siendo así necesario generar una nueva en caso de que el usuario haya olvidado la misma.

Para el resto de información confidencial que maneja la web utilizamos un tipo de encriptado bidireccional.

Nuestro sistema se encuentra dentro de la protección de nivel alto, según laLOPD, puesto que contenemos datos personales y/o información médica privada. Portanto, nuestro sistema ha tenido que cumplir las siguientes restricciones.

A) Distribución de soportes (art.23): La distribución de los soportes que contengan datosde carácter personal se realizará cifrando dichos datos o bien utilizando cualquier otromecanismo que garantice que dicha información no sea ininteligible ni manipuladadurante su transporte.

B) Registro de Accesos (art.24).

Esta medida impuesta por el Reglamento es la que conlleva más problemastécnicos y económicos para su implantación en las empresas, dado que han deconfigurarse las

aplicaciones destinadas al tratamiento de los datos para queguarden y almacenen un gran volumen de datos.

Establece el Reglamento que, de cada acceso, se guardarán como mínimo:

a) La identificación del usuario,

b) Fecha y la hora en que se realizó el acceso,

c) Fichero accedido,

d) Tipo de acceso: autorizado o denegado,

e) Y en el caso que el acceso haya sido autorizado, será preciso guardar lainformación que permita identificar el registro accedido.

C) Copias de respaldo y recuperación (art.25).

El Reglamento establece para los ficheros de datos de nivel alto que deberáconservarse una copia de respaldo y de los procedimientos de recuperación de losdatos en un lugar diferente a aquél en que se encuentran los equipos informáticosque los tratan cumpliendo, en todo caso, las medidas de seguridad exigidas.

D) Transmisión de datos por redes de telecomunicaciones (art.26).

El reglamento, finalmente, establece que la transmisión de datos de carácterpersonal a través de redes de telecomunicaciones se realizará cifrando dichosdatos o bien utilizando cualquier otro mecanismo que garantice que lainformación no sea inteligible ni manipulada por terceros.

# 12. Conclusiones

El desarrollo del proyecto de Sistemas Informáticos nos ha servido para enfrentarnos a un proyecto real, en un mercado real y con un cliente real.

GCT está hecho a la medida de un cliente, un cliente exigente a la hora de ver facilitada su labor a través de una herramienta de software, un cliente que no ha encontrado una solución informática que se ajuste a las necesidades de su negocio y que ha recurrido a nosotros para poder obtenerla.

Precisamente el haber tenido un cliente real, consideramos que ha sido una experiencia muy positiva, nos ha permitido enfrentarnos a casos reales, casos que nos ocurrirán en el día a día en un futuro si nos dedicamos a desarrollar software para un cliente. Hemos participado en un proyecto en todas sus fasesde principio a fin, desde la búsqueda de los requisitos que el cliente exige a través de las entrevistas con el mismo pasando por las múltiples variaciones del diseño que nos ha llevado a hacer tras cada una de las entrevistas de seguimiento del proyecto, hasta una presentación final y su puesta en marcha tras una fase de pruebas.

También hemos visto interesante el haber reunido en un mismo proyecto diferentes conocimientos que hemos ido asimilando durante estos años de universidad y haberlos tenido que ampliar para llevar a cabo el proyecto (gestión de bases de datos, desarrollo de aplicaciones Java, aplicaciones web, ingeniería del software, etc)

El uso de nuestros conocimientos en conjunto con la capacidad que existe actualmente en la web para buscar recursos ha sido la clave para poder llevar a cabo nuestro proyecto, el haber utilizado software libre nos ha permitido realizar nuestro proyecto, el uso de éste junto a las enormes cantidades de información que lo rodean en internet ha sido un aspecto clave en el desarrollo de GCT ya que buena parte del tiempo lo hemos invertido en el estudio de esta información.

## 12.1. Estado actual del proyecto y futuras líneas de desarrollo

Actualmente nuestra aplicación da un servicio de gestión completa a cualquier clínica de traumatología, permitiéndole una gestión rápida y eficiente tanto de las fichas de los pacientes como de sus consultas e intervenciones.

También proporciona a los pacientes el acceso web a la consulta de sus citas y la posibilidad de pedir cita online.

Como posibles líneas de desarrollo de la aplicación encontramos principalmente dos:

- Adaptar nuestro software a cualquier tipo de clínica médica, ya que las diferencias entre especialidades no son demasiado relevantes.

- Ajustarnos más a las necesidades de cada cliente especifico, haciendo un software dinamico que nos permita adaptarnos a cada cliente de forma mas especializada.

# 13. Bibliografía

## 13.1. Referencias web

**Spring**:

[1]. http://static.springsource.org/spring/docs/2.0.x/reference/jdbc.html

[2]. http://www.springhispano.org

[3]. http://www.springdeveloper.com

[4]. http://static.springsource.org/spring/docs/2.0.x/reference/jdbc.html

**Tortoise SVN**

[5]. http://tortoisesvn.net

Java

[6]. http://www.oracle.com/technetwork/java/javamail/index.html

PHP

[7]. http://www.php.net/

[8]. http://swiftmailer.org/

jQuery

[9]. http://jquery.com/

[10]. http://jqueryui.com/

[11]. http://bassistance.de/jquery-plugins/jquery-plugin-validation/

XAMPP

[12]. http://www.apachefriends.org/es/xampp.html

MySQL

[13]. http://dev.mysql.com/doc/refman/5.0/es/index.html

## 13.2. Referencias bibliográficas

[1]. Converse, Tim. Park, Joyce. Morgan, Clark. PHP5 and MySQL BIBLE. Wiley Publishing, Inc. 2004

[2]. Walls, Craig. Spring. Manning Publications Co..2008

[3]. Wellman, Dan. JQuery 1.7. Ediciones Anaya(Packt Pub). 2009.

[4]. Silberschatz, Abraham. Fundamentos de Bases de Datos. Editorial McGraw –Hill, 2006

# 14. Agradecimientos

A Eva Guillén y Moncho Domínguez por su colaboración en el diseño del logo.

A Ángel del Blanco Aguado por su ayuda y consejos en el uso del Spring Framework.

A nuestros familiares y amigos que nos han apoyado en todo momento.

Al Dr. Truan, jefe del Servicio de Cirugía Ortopédica y Traumatología del Hospital Universitario Madrid Monteprincipe.