TÍTOL: Gestión material quirúrgico hospitalario. Dispositivos Móviles

AUTOR: David Palazón Bermúdez

TITULACIÓ: Ingenieria técnica en informàtica de Gestión

DIRECTOR: DAUDÉ VENTURA, JORGE

DEPARTAMENT: LSI

DATA: Junio 2014
Este proyecto de final de carrera consiste en el desarrollo de una aplicación para la gestión comercial del sector Médico. Concretamente centrado en las peticiones de material quirúrgico.

Para poder clasificar las diferentes transacciones se han establecido conceptos que nos permitan clasificarlo como Doctores, Hospitales, Tipos de cirugía y Mutuas.

En función de estas características comunes posteriormente se podrán establecer conjuntos de transacciones de las cuales obtener múltiples estadísticas y/o gráficos.

Para la generación de la app partimos de unas premisas.
- Tenía que ser Rápida
- Tenía que funcionar desconectada.
- Tenía que poder volcar los datos a un sistema central
- Tenía que ser multiplataforma IOS/ANDROID.

Paraules clau (màxim 10):

<table>
<thead>
<tr>
<th>ANDROID</th>
<th>MVC</th>
<th>PHP</th>
<th>TABLET</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOS</td>
<td>APP</td>
<td>MOVIL</td>
<td>MYSQL</td>
</tr>
<tr>
<td>ACTIONSCRIPT</td>
<td>SQLite</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Contenido

1. Introducción al proyecto ........................................................................................................... 5
   1.1 Introducción ......................................................................................................................... 5
   1.2 Motivación del Proyecto .................................................................................................... 5

2. Descripción del Proyecto ......................................................................................................... 7
   2.1 Descripción General ........................................................................................................... 7
   2.2 Gestión de doctores ........................................................................................................... 7
   2.3 Gestión de Hospitales ....................................................................................................... 8
   2.4 Gestión de Tipos de cirugía ............................................................................................... 8
   2.5 Gestión de Mutuas ............................................................................................................. 8
   2.6 Gestión de Solicitudes ....................................................................................................... 8
   2.7 Necesidades Técnicas ...................................................................................................... 9
   2.8 Funcionamiento del Negocio ........................................................................................... 9

3. Evaluación de la tecnología .................................................................................................. 12
   3.1 Lenguajes de programación ............................................................................................. 12
       3.1.1 Apache Flex ............................................................................................................... 12
   3.2 Sistema gestor de bases de datos ................................................................................... 13

4. Análisis y especificación ...................................................................................................... 14
   4.1 Análisis de Requerimientos ............................................................................................ 14
   4.2 Requisitos funcionales .................................................................................................... 14

5. Modelo Conceptual ............................................................................................................. 15

6. Casos de Uso ....................................................................................................................... 16
   6.1 Casos de Uso de Solicitud .............................................................................................. 16
       CU1. Crear Solicitud ............................................................................................................. 16
       CU2. Modificar Solicitud .................................................................................................... 16
       CU1. Consultar Solicitud .................................................................................................... 16
   6.2 Casos de uso de Doctor ................................................................................................... 17
       CU1. Crear Doctor .............................................................................................................. 17
       CU1. Modificar Doctor ....................................................................................................... 18
       CU1. Consultar Doctor ....................................................................................................... 18
   6.3 Casos de uso de Hospital ................................................................................................ 19
       CU1. Crear Hospital ............................................................................................................ 19
       CU1. Modificar Hospital .................................................................................................... 19
       CU1. Consultar Hospital ................................................................................................. 20
   6.4 Casos de uso de Mutua ................................................................................................... 20
       CU1. Crear Mutua ................................................................................................................ 20
       CU1. Modificar Mutua ....................................................................................................... 20
       CU1. Consultar Mutua ....................................................................................................... 21
   6.5 Casos de uso de Tipo de cirugía ...................................................................................... 21
       CU1. Crear Tipo Cirugía ..................................................................................................... 21
       CU1. Modificar Tipo Cirugía ............................................................................................ 22
       CU1. Consultar cirugía ....................................................................................................... 22
   6.6 Casos de Uso de Usuario ............................................................................................... 23

7. Diagramas de secuencia ...................................................................................................... 24
   7.1 Solicitudes ....................................................................................................................... 24
       7.1.1 Alta Solicitud .............................................................................................................. 24
       7.1.2 Modificación Solicitud ............................................................................................. 25
       7.1.3 Consultar Solicitud ................................................................................................. 25
7.1.4 Listado Solicitudes.................................................................26
7.2 Hospital ..................................................................................27
  7.2.1 Alta Hospital .................................................................27
  7.2.2 Consulta de Hospital ....................................................27
  7.2.3 Modificar Hospital .........................................................28
  7.2.4 Listado Hospital .............................................................28
7.3 Casos de uso Mutuas ..............................................................29
  7.3.1 Listado Mutuas ..............................................................29
  7.3.2 Añadir Mutua .................................................................29
  7.3.3 Modificar Mutua .............................................................30
7.4 Casos de uso de doctores ......................................................31
  7.4.1 Listado de Doctores .......................................................31
  7.4.2 Añadir Doctores ...........................................................31
  7.4.3 Modificar Doctor ..........................................................32
  7.4.4 Consultar Doctor ..........................................................32
7.5 Caso de Uso Tipo cirugía .......................................................33
  7.5.1 Listado Tipos de Cirugía ..................................................33
  7.5.2 Añadir nuevo tipo de cirugía ..........................................33
  7.5.3 Modificar tipo de cirugía ................................................34
  7.5.4 Consultar tipo de cirugía ................................................34
7.6 Caso de uso login .................................................................35
  7.6.1 Login con Conexión .......................................................35
8. Diseño de Interfaz gráfica .......................................................36
  8.2 MadComponents ..............................................................37
    8.2.1 Definición de pantallas con XML con la librería MadComponents .....................37
    Pantalla de Login .................................................................38
9. Modelo de componentes .......................................................41
10. Diseño de la capa de gestión de datos .....................................46
  10.1 Diseño de la base de datos .............................................46
  10.2 Definición de las tablas ...................................................46
11. Implementación y test ..........................................................49
  11.1 Metodología Empleada ...................................................49
  11.2 Implementación .............................................................49
  11.3 Testing ..............................................................................50
12. Conclusiones y trabajos Futuros ...........................................51
  12.1 Conclusiones .................................................................51
  12.2 Trabajos futuros .............................................................51
13. Bibliografía ............................................................................52
Apéndices ..................................................................................53
  Manual de instalación ..........................................................53
1. Introducción al proyecto

1.1 Introducción

Durante las últimas décadas hemos podido observar como la informática pasaba de una mesa a nuestro bolsillo. Actualmente llevamos dispositivos móviles que compiten en prestaciones con muchos equipos de sobremesa.

Se podría afirmar que nuestros móviles hacen de todo y además permiten realizar llamadas telefónicas.

Estos dispositivos han sustituido nuestros anticuadas PDAS y en algunos casos incluso los ordenadores de sobremesa y portátiles.

Por este motivo se ha convertido en una extensión de la oficina para aquellas personas que estamos fuera de ella mucho tiempo.

1.2 Motivación del Proyecto.

A continuación procederé a explicar cuáles han sido mis motivaciones personales para realizar este proyecto. Las motivaciones han sido dos completamente diferentes por un lado he tenido una motivación personal , siempre he sido muy curiosos en todo lo referente a las tecnologías y desde muchos años sigo todo lo que ha sido el movimiento open source. Apasionado de Linux desde la versión Slackware 1.0.

Hace unos años, me sumergí en el mundo de adobe con un proyecto en el que tuve que desarrollar un back-end para una aplicación hecha en Flash. Y en aquel momento ya despertó mi interés. Un entorno multiplataforma, pero estaba muy verde y no tenía un entorno de programación como tal. Pero con el tiempo pude observar que todo esto cambiaba y se creaba un entorno para separar Diseñadores y programadores. Y desde ese momento se podía desarrollar aplicaciones sin tener que entrar en las app’s de diseño. Pero los precios eran prohibitivos.

Durante ese tiempo me metí en el mundo de la consultoría SAP y eso dejó muy reducido mi tiempo libre para conocer y experimentar.

Después de muchos años en el mundo de la consultoría he pasado al cliente final. Concretamente a una empresa dedicada a la comercialización de material quirúrgico (Prótesis) de alta tecnología.

Y es por el día a día que me planteo iniciar este proyecto. Las razones han sido básicamente dos:

Motivación Personal. Era una buena excusa para conocer el mundo de la programación móvil y que te paguen el tiempo invertido.

Motivación Funcional. Desde hace un tiempo intentamos cubrir la necesidad de recoger las cirugías que hemos de atender con la máxima información, actualmente se reciben por mail y por fax, y tenerla ordenada de forma que sea fácil de clasificar y localizar posteriormente. Para resolver este problema se encargó la realización de una app en Filemaker, que aunque llevamos más de un año en “pruebas” no hemos podido poner en explotación.

Las causas son varias pero las más importantes son:
- La instalación no es la estándar de una app
- El desarrollo solo es para IOS formato Tablet (IPAD) y no todo el personal comercial tiene uno. Se les regala cuando llevan un año y han demostrado su valía.
- La parte servidor necesita un equipo especial (Cloud OSX).
- Las licencias para accesos concurrentes al servidor son prohibitivas.

De todo este cumulo de circunstancias y por las peticiones de los comerciales unos 50 sobre la posibilidad de correr en Android me planteé como proyecto personal fuera de mi horaria laboral de crear una alternativa y cuando fuera equiparable presentarla a Dirección comercial.
2. Descripción del Proyecto

2.1 Descripción General

Este proyecto consiste en la creación de una app para facilitar la notificación por parte del comercial de toda la información relacionado con la solicitud para realizar una cirugía.

Esta información incluye los datos necesarios para poder gestionar logísticamente el envío y recepción del material y por otro lado los datos que posteriormente nos pide el cliente para dejarnos emitir un albarán y poder facturaciones. Esto es sumamente importante en el caso de los hospitales privados que utilizan el mínimo detalle para demorar la emisión del albarán y retrasar el pago.

- Lugar
- Hora y doctor que realizará la cirugía para que el personal de logística pueda preparar los Kits necesarios y servirlos con la antelación necesaria.

A nivel Facturación los datos que se recogerán serán:

- Datos como son el Paciente, DNI, Teléfono, cirugía Pública o Privada y en caso de ser a través de una Mutua los datos de autorización y número de póliza.

La aplicación tendrá que permitir que el comercial informe de un doctor nuevo en cualquier momento y le tendrá que permitir enviar emails o documentación a los responsables de una cirugía.

La app se tendrá que poder distribuir fácilmente, tendrá que ser susceptible de realizarse cambios con la mayor facilidad posible y tiene que estar libre de cualquier carga de licencias.

Como actualmente el sistema está realizado en una aplicación propietaria que corre solo en entornos móviles IOS, la app tiene que cubrir el máximo de sistemas móviles.

La app tiene que cubrir básicamente con la generación de las solicitudes, consulta y búsqueda rápida. Y permitir la edición de aquellos maestros que sean funcionalmente necesarios. Como en determinadas ocasiones puede que haya datos que no estén informados en el sistema tiene que permitir modificar los Maestros. En nuestro caso los Maestros serán los doctores que realizan las cirugías, los diferentes tipos de cirugía, los centros donde se realizan y las mutuas que pagarán las cirugías.

El objetivo de este proyecto es montar las bases para realizar una aplicación que tenga una comunicación bidireccional con el sistema de gestión que no consuma licencias de ningún tipo y que sea altamente modificable.

2.2 Gestión de doctores.

El doctor sobre todo en todas las cirugías privadas es muy importante porque de la impresión que tenga de nosotros y nuestro material dependerá que se realicen más o menos cirugías.

Se gestionarán las altas y modificaciones de los doctores. Las bajas no se han implementado puesto que por decisión corporativa interesa tener los datos de todos los doctores con los que hemos trabajado. Estos solo se podrán eliminar desde la aplicación de gestión ERP.

Para dar de alta a un doctor se le solicitará los siguientes datos: NIF, nombre, apellidos, teléfono, móvil, correo electrónico, número de colegiado y especialidad. El sistema permitirá realizar anotaciones guardando la fecha de la nota pero no se podrá eliminar.

Se realizará la validación del NIF y del email. Además el sistema controlará que no haya duplicidad de
doctores a través del NIF. No podemos utilizar el código de colegiado porque van en función de la provincia y la especialidad. En caso de tener dos especialidades el doctor tendrá dos números de colegiado.

2.3 Gestión de Hospitales.

El hospital es una entidad importante del negocio pero tiene dos roles diferentes en función de si es privado o público. En caso de ser privado a nivel de gestión solo nos interesa porque es donde se ha de entregar el material pero el que pagará las facturas será el Paciente en caso de ser cirugía 100%Particular o la Mutua. En caso de ser público Hospital y Cliente suelen ser el mismo.

Se permite dar de alta y modificar los datos de un hospital pero al igual que en la gestión de doctores no se permite eliminar un hospital.

Para dar de alta un hospital se tendrá que informar del Nombre, Teléfono de la persona de Compras, email de compras, dirección, código ERP y NIF.

En cualquier momento se pueden modificar los datos del hospital.

Al igual que en el resto de módulos se controlará que los campos NIF y correo sean válidos.

2.4 Gestión de Tipos de cirugía

Los tipos de cirugía en nuestro caso están ligados a la familia. Por este motivo y como tiene implicaciones contables estos datos vienen del sistema de Gestión (ERP). La información que tendremos en la app es código y descripción.

2.5 Gestión de Mutuas.

La Mutua es un dato importante y que se actualizado muy poco, por este motivo el sistema lo recibirá desde la aplicación de Gestión siendo imposible modificarlo desde la app. Se podrán consultar los datos en todo momento.

2.6 Gestión de Solicitudes.

La transacción principal que conforma el negocio es la solicitud y en el caso del comercial básicamente se podría considerar un pedido. Los datos a informar al dar de alta una solicitud son los siguientes:

Datos de la cirugía. Fecha de la cirugía, Doctor, Hospital, Tipos de Cirugía.
Datos del Paciente: Nombre, Apellidos, DNI, Teléfono, Email.
Datos de facturación: Privado/Publico, Particular/Mutua, Nº Póliza, Nº Autorización

Notas: Se permiten realizar anotaciones (normalmente para el departamento logístico) pero no se pueden modificar una vez grabados. En la nota quedara una marca de tiempo para saber siempre en qué momento se indicó.

En cualquier momento el usuario podrá modificar la solicitud siempre que su estado no sea procesado.

Una solicitud no se podrá borrar nunca, se le pude cambiar el estado indicando que se ha anulado y el motivo (texto libre).

Desde la misma solicitud el sistema permitirá llamar al Doctor, al Hospital, al Gestor de la mutua o
2.7 Necesidades Técnicas

A nivel técnico los requerimientos son los siguientes.

La app tendrá que ser capaz de volcar y recoger los datos de una base de datos externa, esa base de datos será utilizada por el ERP también pero no será la que contenga toda la información del negocio.

En muchas ocasiones en las zonas donde se reúnen los comerciales con los doctores para tomar nota de sus necesidades no hay cobertura de ningún tipo de red de datos ya sea móvil o wiki. Por este motivo la aplicación tiene que ser desconectada.

Tiene que ser fácilmente modificable. Se han de poder realizar cambios en los diferentes módulos e incluso sustituirlos completamente o añadir nuevos sin que eso afecte al resto. Tiene que ser multiplataforma. No tiene que tener precio de licencia por usuario.

Se plantea en formato multiplataforma formato móvil porque todos los comerciales tienen un móvil de empresa o phablets. Se les da la opción de pagar un diferencial si quieres un iPhone haciendo que el parque tenga las dos tecnologías (IOS y Android).

2.8 Funcionamiento del Negocio.

Para entender más la problemática pasaremos a explicar brevemente el funcionamiento del negocio. Tal como he comentado anteriormente estamos tratando con una empresa dedicada al suministro de material quirúrgico de alta tecnología a hospitales públicos y a Privados.

Ha diferencia de un negocio de distribución habitual en las que el material que se entrega es el solicitado por el cliente en las empresas de estas características el cliente solo expresa su deseo de realizar un tipo o varios de cirugía en una fecha concreta.

Como las medidas reales no se conocen hasta el momento en que se ha abierto al paciente en quirófano la empresa no puede determinar el tipo de material a servir, estando obligada a entregar a los hospitales todo el material necesario para realizar la cirugía para cualquier paciente (diferenciando entre adultos y niños). Esto implica que se pueden entregar cientos de referencias contenidas en un embalaje esterilizable (esto es lo que denominamos Kit) y del que se consumirán solo un número pequeño de referencias.

Durante la cirugía una persona de las que está en quirófano va anotando el material empleado y sus números de lote. Las etiquetas del material si la llevan se pueden pegar en el parte. Posteriormente esta información llega por fax, foto de móvil o escaneado. Este documento contendrá el registro del material empleado, lotes y datos del paciente. Este último dato es importante en caso del privado, pues un requisito para emitir un albarán es que conste parte de esta información.

La problemática radica en que cada Kit puede tener precios de más de 30000 Euros de precio medio de coste.

Hay que tener un nivel alto de calidad de servicio (casi del 96%) porque no tenerlo implicaría cirugías que no se podrían hacer, para cumplir con este nivel se tiene que recurrir a una de las siguientes
opciones:

- **Tener exceso de stock.** Contradictorio para la empresa por la cantidad de inmovilizado que además es perecedero (mucho material tiene caducidad).
- **Tener una gestión logística muy complicada con alta rotación del material.**

En nuestro caso usamos el segundo método, esto provoca que el gran peso y responsabilidad del buen servicio al cliente recae en el departamento logístico (compras y suministro). Para que el departamento pueda planificar correctamente la entrega y recogidas es muy importante tener las fechas de cirugía y los datos del hospital.

En la planificación entran muchos parámetros pero algunos de los más destacados son los siguientes:

- **La población del hospital.** Esta condiciona el tiempo de anticipación con el que hemos de planificar la entrega del material y también el tiempo que tardará en volver para enviarlo a otro hospital (a veces se envía directamente). En nuestro caso en transportes habituales trabajamos con los siguientes valores. Si estamos en Cataluña tendremos que enviar el material con 1 día de antelación y uno de recogida antes de poder utilizar el material en otra cirugía. Resto de provincias 2 días y para canarias 3. En nuestro que tenemos filiales en Portugal, Italia y Alemania puede ser que tengamos Kits especialmente caros o especiales y que nos lo pidan prestados. Esto implica que tenemos ese material bloqueado y no podemos realizar cirugías de ese tipo mientras estén en las filiales.
- **Días Laborables Locales y Estatales.**
- **Anulaciones de última hora.** Puede ser que la cirugía se anule/atrase porque el paciente tiene fiebre, está enfermo o se ha muerto circunstancias en las que el material se tiene que gestionar para recuperarlo lo antes posible.
- **Múltiples Cirugías.** A veces en el mismo centro se pueden hacer muchas cirugías del mismo tipo.
- **Solicitudes Especiales.** En ocasiones el doctor es un poco “caprichoso” y pide otro material que no forma parte del mismo kit y se le ha de entregar suelto.
- **Hospital sin departamento esterilización.** Si el hospital no tiene departamento de esterilización el material tiene que volver a nuestras instalaciones para pasar la esterilización (4 horas).

A nivel de compras tenemos otro problema “*No se puede planificar con antelación*” porque “no tenemos pedidos en firme” que nos indiquen los futuros consumos. Esto comporta que no podemos hacer una planificación de las necesidades, ni una previsión de la demanda. En nuestro caso estamos utilizando una herramienta de previsión de la demanda según tendencias llamado Lockad realizado por un equipo de estadísticos de una universidad Francesa.

A todo esto se nos suma la problemática de las artimañas de los comerciales o doctores que planifican cirugías para días antes de la que realmente tienen para asegurarse el material. Esto provoca que se entregue el material y luego la anulen. De esta forma el material está en el hospital para la cirugía real que se hace varios días después. Los comentarios que actualmente envían por email a veces implican cambios de planificación y a veces llegan con unas horas de antelación obligando a transportes especiales o a entregas en mano haciendo viajes en avión por parte del personal del almacén.

Hasta este momento parece que el problema lo tenemos en el departamento de logística pero no es completamente cierto. Como comentaba se realizaba la cirugía y se enviaba un parte de cirugía por parte del hospital.

Con este parte se registra el consumo generando un albarán. En ocasiones este parte no coincide con el material que falta del kit y hay que entablar conversaciones con el cliente normalmente obligando a que
cuadre con sus datos. Antes de generar el albarán se ha de enviar un documento proforma con el material que vamos a albaranar. En ese documento según el cliente (públicos no hay demasiados problemas) se ha de hacer constar datos del paciente Nombre, Nº Póliza, Nº Autorización, Fecha Cirugía,… Si todo es correcto entonces se nos permite albaranar y luego generar factura. La falta de alguno de estos datos puede acarrear complicadas gestiones, llamadas de teléfono y favores que pueden hacer que se demore el albarán varios meses sin tener en cuenta que luego la factura puede tener una media de 120 días para cobrarla.

En ocasiones el doctor realiza operaciones con material que le gusta (privado) pero que la mutua no tiene autorizado, en estos casos se tendría que hacer presupuesto previo si en el sistema no tenemos con anterioridad que esa cirugía se hace a través de la mutua X no podemos darnos cuenta de esta problemática y generar oferta antes de la cirugía. Por este motivo en ocasiones quedan facturas que no se cobrarán nunca.
3. Evaluación de la tecnología.

3.1 Lenguajes de programación.

Todos los que nos sumergimos en el mundo de la programación de dispositivos móviles nos encontramos siempre con una decisión importante. ¿Y ahora con qué programo?

Cuando intentamos desarrollar para entornos móviles nos encontramos que inicialmente según para que plataforma queramos desarrollar tenemos que elegir.

Para dispositivos IOS todo nos conduce al entorno del propio fabricante, que tiene un coste ridículo, lo regalan con el sistema operativo pero que implica comprar un equipo APPLE. Este entorno nos da todo el material necesario, la documentación y el entorno de programación. El lenguaje de programación es en este caso objective–C. Es un lenguaje orientado objetos basado en C.

Para Android tenemos la alternativa de google, este está basado en Java y un SDK propietario que suministra todas las herramientas necesarias al igual que APPLE. El coste de este es gratuito y como ventaja es que cualquier sistema operativo que corra java y tengo entorno nativo para desarrollar en java nos servirá.

Pero si queremos que una misma app funcione nos preguntamos qué opciones tenemos. En este caso tenemos que recurrir a software que cumpla el siguiente paradigma “write once, run everywhere”. Software que se programa en un único idioma y que el compilador es capaz de generar código para diferentes maquinas.

Lo cierto es que existen unas cuantas alternativas muy buenas y profesionales:

- .NET (Proyecto Mono)
- Appcelerator Titanium
- PhoneGap
- Apache Flex
- Haxe.

En mi caso impuse dos condiciones que el lenguaje se pareciese a alguno de los que ya conocía o que tuviese muchas similitudes y que no fuese necesario un entorno Apple para desarrollar para esta plataforma.

Esto nos dejaba dos opciones. Haxe y Apache Flex los dos basados en el lenguaje de programación de Adobe ActionScript. Al final por tiempo en el mercado y documentación opte la segunda opción.

En el backend se montaran servicios simples sobre la plataforma APACHE junto a la librería AMFPHP. La decisión es esta por ser uno de los más utilizados en los servidores de hosting dando más versatilidad de despliegue.

3.1.1 Apache Flex

Apache Flex antes llamado Adobe Flex es un Kit de desarrollo de Software (SDK) que permite realizar aplicaciones Ricas de internet (RIA’S) con el lenguaje ActionScript. Esto genera apps que corren en la plataforma de Adoba Flash.

Inicialmente fue desarrollado por Macromedia y luego adquirido por Adobe. En diciembre del 2011 lo donaron a la Apache Software Foundation y ahora consta como proyecto de nivel superior.
La última versión fue liberada bajo la licencia 2 de Apache.

El lenguaje Actionscript es muy similar al lenguaje Java y C++. A continuación cito algunas de sus características:

- Estructurado
- Orientado a Objetos
- Multiplataforma y Libre
- Gestión de Memoria
- Programación Modular y reutilizable
- Tiene una extensión del ECMAScript que añade XML como un tipo de datos nativo del lenguaje.

Mediante librerías específicas para el dispositivo ANE, podemos acceder a prácticamente todas sus características.

3.2 Sistema gestor de bases de datos

A nivel de base de datos en nuestro caso al tener que ser local para que pudiese ser autónomo solo teníamos una opción SQLite.

SQLite es un sistema de gestión de bases de datos relacional que cumple el ACID, esto implica que implanta los medios suficientes para poder establecer un sistema transaccional. No tiene una parte servidor sino que se integra dentro de la aplicación como si fuese parte de ella reduciendo la carga de memoria y teniendo un mayor rendimiento. La base de datos está contenida en un único fichero. Para cumplir con el modelo transaccional lo que realiza es un bloqueo del fichero completo en cada transacción. En la versión que utilizo (la versión 3) soporta ficheros de 2 Terabytes e inclusión de ficheros Blob.

Ventajas de utilizar la base de datos local:

- Para su uso no es indispensable disponer de conexión
- SQLite es muy ligero y veloz.
- Aunque la capacidad de datos es mucho más limitada, en nuestro caso es muy superior a la que dispondremos en el dispositivo móvil.

En el Back-End,

En el backend utilizaremos el servidor más utilizado habitualmente, en nuestro caso Mysql. Porque eso nos permitirá tener un mayor abanico de equipos donde desplegarlo sin necesitar infraestructuras complejas.
4. Análisis y especificación

4.1 Análisis de Requerimientos.

A partir de la descripción general del proyecto se inicia el análisis de requerimientos. Estos requerimientos serán los que determinen que y como se resuelven los distintos casos de uso en el momento de realizar la especificación Los requisitos definidos son los siguientes.

4.2 Requisitos funcionales

- **R1: Transacción principal (Solicitud)**
  - R1.1 El sistema tiene que permitir que el actor pueda dar de alta nuevas solicitudes
  - R1.2. El sistema tiene que permitir consular cualquier transacción existente que el actor haya creado previamente.
  - R1.3 El sistema tiene que permitir que el actor modifique las características de sus transacciones siempre que no se haya enviado
  - R1.4 El sistema tiene que permitir que el actor anule (no está permitido borrarlas)

- **R2: Doctores**
  - R2.1 El sistema tiene que permitir que el actor pueda dar de alta nuevos doctores.
  - R2.2 El sistema tiene que permitir que el actor pueda modificar los datos de cualquier doctor.
  - R2.3 El sistema tiene que permitir consultar los datos de cualquier doctor.

- **R3: Tipos de cirugía**
  - R3.1 El sistema tiene que permitir que el actor consulte cualquier tipo de cirugía.

- **R4: Mutuas.**
  - R4.1 El sistema tiene que permitir que el actor consulte cualquier mutua.

- **R5: Fichero (Se dejan para una segunda fase)**
  - R5.1 El sistema tiene que permitir generar documentos en formato CSV y enviarlos por email

- **R6: Estadísticas. (Se dejan para una segunda fase)**
  - R6.1 El sistema ha de crear informes de las cirugías realizadas en un hospital
  - R6.2 El sistema ha de crear informes de las cirugías según tipo de cirugía.
  - R6.3 El sistema ha de crear informes de las cirugías según la mutua
  - R6.4 El sistema ha de crear informes de las cirugías según el Doctor.

- **R7: Usuarios**
  - R6.1: El sistema tiene que tener un sistema de validación para entrar dentro de la app.
5. Modelo Conceptual

A continuación podemos ver el modelo conceptual del sistema que muestra los conceptos del proyecto, las relaciones que existen entre ellos y que tipo de relación tienen:
6. Casos de Uso

6.1 Casos de Uso de Solicitud.

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Crear Solicitud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>Permite crear una solicitud nueva en el sistema</td>
</tr>
<tr>
<td>Actores:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Usuario Logeado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• El usuario tiene que estar logeado.</td>
</tr>
<tr>
<td></td>
<td>• El Doctor que realiza la cirugía tiene que existir en la app</td>
</tr>
<tr>
<td></td>
<td>• El Hospital tiene que existir en la app</td>
</tr>
<tr>
<td></td>
<td>• El Tipo de cirugía tiene que existir en la app</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. El actor pulsa el botón para dar de alta una nueva solicitud</td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra una vista donde informar de los datos de la cirugía.</td>
</tr>
<tr>
<td></td>
<td>3. El actor introduce los datos de la cirugía</td>
</tr>
<tr>
<td></td>
<td>4. El sistema valida los datos y los almacena</td>
</tr>
<tr>
<td>Flujo Alternativo:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.</td>
</tr>
<tr>
<td></td>
<td>a. La fecha de cirugía no es posterior a la de creación+1</td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra un mensaje y permite corregir los datos.</td>
</tr>
<tr>
<td>PostCondiciones:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>La solicitud se ha registrado en el sistema.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU2. Modificar Solicitud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>El sistema permite elegir una solicitud existente para poder modificarla</td>
</tr>
<tr>
<td>Actores:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Usuario Logeado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• El usuario tiene que estar logeado.</td>
</tr>
<tr>
<td></td>
<td>• El Doctor que realiza la cirugía tiene que existir en la app</td>
</tr>
<tr>
<td></td>
<td>• El Hospital tiene que existir en la app</td>
</tr>
<tr>
<td></td>
<td>• El Tipo de cirugía tiene que existir en la app.</td>
</tr>
<tr>
<td></td>
<td>• La solicitud tiene que existir en el sistema.</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1. El actor selecciona una solicitud para modificarla dentro de un listado.</td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra una vista con los datos de la solicitud.</td>
</tr>
<tr>
<td></td>
<td>3. El actor introduce los datos de la cirugía</td>
</tr>
<tr>
<td></td>
<td>4. El sistema valida los datos y los almacena</td>
</tr>
<tr>
<td>Flujo Alternativo:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra una vista que solo permite modificar las Notas</td>
</tr>
<tr>
<td>PostCondiciones:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• La solicitud se ha modificado en el sistema.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Consultar Solicitud</th>
</tr>
</thead>
</table>
Autor: David Palazón

Descripción:
El sistema permite elegir una solicitud existente para poder consultarla.

Actores:
- Usuario Logado

Precondiciones:
- El usuario tiene que estar logrado.
- El Doctor que realiza la cirugía tiene que existir en la app.
- El Hospital tiene que existir en la app.
- El Tipo de cirugía tiene que existir en la app.
- La solicitud tiene que existir en el sistema.

Flujo Normal:
- El actor selecciona una solicitud para visualizarla dentro de un listado.
- El sistema muestra una vista con los datos de la solicitud.

PostCondiciones:
- La solicitud permanece igual que estaba inicialmente.

6.2 Casos de uso de Doctor

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Crear Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
</tbody>
</table>

Descripción:
Permite crear un doctor nuevo en el sistema.

Actores:
- Usuario Logado

Precondiciones:
- El usuario tiene que estar logrado.

Flujo Normal:
1. El actor pulsa el botón para dar de alta nuevo doctor.
2. El sistema muestra una vista donde informar de los datos del doctor.
3. El actor introduce los datos de la doctor.
4. El sistema valida los datos y los almacena.

Flujo Alternativo
4. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.
   a. El email informado no es válido.
   b. El DNI informado no es válido.
   c. El doctor ya existe en el sistema.
5. El sistema muestra un mensaje y permite corregir los datos.

PostCondiciones:
- El doctor se ha registrado en el sistema.
<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Modificar Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>El sistema permite elegir un doctor existente y modificarlo</td>
</tr>
<tr>
<td>Actores:</td>
<td>• Usuario Logado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td>• El usuario tiene que estar logrado.</td>
</tr>
<tr>
<td></td>
<td>• El Doctor tiene que existir en el sistema</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td>1. El actor selecciona un doctor para modificarlo dentro de un listado.</td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra una vista con los datos del doctor.</td>
</tr>
<tr>
<td></td>
<td>3. El actor modifica los datos del Doctor</td>
</tr>
<tr>
<td></td>
<td>4. El sistema valida los datos y los almacena</td>
</tr>
<tr>
<td>Flujo Alternativo.</td>
<td>4. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.</td>
</tr>
<tr>
<td></td>
<td>a. El email informado no es válido.</td>
</tr>
<tr>
<td></td>
<td>b. El DNI informado no es válido.</td>
</tr>
<tr>
<td></td>
<td>c. El doctor ya existe en el sistema.</td>
</tr>
<tr>
<td>PostCondiciones:</td>
<td>• La solicitud se ha modificado en el sistema.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Consultar Doctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>El sistema permite elegir una doctor existente para poder consultarla</td>
</tr>
<tr>
<td>Actores:</td>
<td>• Usuario Logado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td>• El usuario tiene que estar logrado.</td>
</tr>
<tr>
<td></td>
<td>• El doctor tiene que existir en el sistema.</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td>1. El actor selecciona un doctor para visualizarla dentro de un listado.</td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra una vista con los datos del doctor.</td>
</tr>
<tr>
<td>PostCondiciones:</td>
<td>• El doctor permanece igual que estaba inicialmente.</td>
</tr>
<tr>
<td>Notas:</td>
<td>El sistema permitirá llamar o enviar email con los datos visualizados.</td>
</tr>
</tbody>
</table>
### 6.3 Casos de uso de Hospital

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Crear Hospital</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>Permite crear un hospital nuevo en el sistema</td>
</tr>
<tr>
<td>Actores:</td>
<td>• Usuario Logado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td>• El usuario tiene que estar logrado.</td>
</tr>
</tbody>
</table>
| Flujo Normal:   | 1. El actor pulsa el botón para dar de alta nuevo hospital  
|                 | 2. El sistema muestra una vista donde informar de los datos del hospital.  
|                 | 3. El actor introduce los datos del Hospital  
|                 | 4. El sistema valida los datos y los almacena                              |
| Flujo Alternativo| 4. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.  
|                 | a. El hospital ya existe en el sistema.  
|                 | 5. El sistema muestra un mensaje y permite corregir los datos.             |
| PostCondiciones:| El usuario se ha registrado en el sistema.                                  |

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Modificar Hospital</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>El sistema permite elegir un hospital existente y modificarlo</td>
</tr>
<tr>
<td>Actores:</td>
<td>• Usuario Logado</td>
</tr>
</tbody>
</table>
| Precondiciones: | • El usuario tiene que estar logrado.   
|                 | • El Hospital tiene que existir en el sistema                                  |
| Flujo Normal:   | 1. El actor selecciona un hospital para modificarlo dentro de un listado.  
|                 | 2. El sistema muestra una vista con los datos del hospital.  
|                 | 3. El actor modifica los datos del hospital  
|                 | 4. El sistema valida los datos y los almacena                                  |
| Flujo Alternativo| 1. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.  
|                 | o El email informado no es válido.  
|                 | o El hospital ya existe en el sistema.                                         |
| PostCondiciones:| • El hospital se ha modificado en el sistema.                                  |
###Nombre | CU1. Consultar Hospital  
Autor: David Palazón  
Descripción: El sistema permite elegir una hospital existente para poder consultarla  
Actores:  
- Usuario Logado  
Precondiciones:  
- El usuario tiene que estar logrado.  
- El hospital tiene que existir en el sistema.  
Flujo Normal:  
1. El actor selecciona un hospital para visualizarla dentro de un listado.  
2. El sistema muestra una vista con los datos del doctor.  
PostCondiciones:  
- El hospital permanece igual que estaba inicialmente.  
Notas: El sistema permitirá llamar o enviar email con los datos visualizados.

###6.4 Casos de uso de Mutua

####Nombre | CU1. Crear Mutua  
Autor: David Palazón  
Descripción: Permite crear una mutua nueva en el sistema  
Actores:  
- Usuario Logado  
Precondiciones:  
- El usuario tiene que estar logrado.  
Flujo Normal:  
2. El actor pulsa el botón para dar de alta nueva Mutua  
3. El sistema muestra una vista donde informar de los datos de la Mutua.  
4. El actor introduce los datos de la Mutua  
5. El sistema valida los datos y los almacena  
Flujo Alternativo  
1. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.  
   a. La mutua ya existe en el sistema.  
2. El sistema muestra un mensaje y permite corregir los datos.  
PostCondiciones: La Mutua se ha registrado en el sistema.

####Nombre | CU1. Modificar Mutua  
Autor: David Palazón  
Descripción: El sistema permite elegir una Mutua existente y modificarla  
Actores:  
 Usuario Logado  
Precondiciones:  
- El usuario tiene que estar logrado.  
- La mutua tiene que existir en el sistema  
Flujo Normal:  
1. El actor pulsa el botón para modificar una Mutua.  
2. El sistema muestra una vista con los datos de la Mutua.  
3. El actor modifica los datos de la Mutua.
4. El sistema valida los datos y los almacena

Flujo Alternativo:
4. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.
   a. El doctor ya existe en el sistema.

PostCondiciones:
- La solicitud se ha modificado en el sistema.

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Consultar Mutua</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>El sistema permite elegir una mutua existente para poder consultarla</td>
</tr>
<tr>
<td>Actores:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Usuario Logado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- El usuario tiene que estar logrado.</td>
</tr>
<tr>
<td></td>
<td>- El doctor tiene que existir en el sistema.</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td></td>
</tr>
<tr>
<td>1. El actor selecciona una mutua para visualizarla dentro de un listado.</td>
<td></td>
</tr>
<tr>
<td>2. El sistema muestra una vista con los datos de la mutua.</td>
<td></td>
</tr>
<tr>
<td>PostCondiciones:</td>
<td></td>
</tr>
<tr>
<td>2. La mutua permanece igual que estaba inicialmente.</td>
<td></td>
</tr>
<tr>
<td>Notas:</td>
<td>El sistema permitirá llamar o enviar email con los datos visualizados.</td>
</tr>
</tbody>
</table>

### 6.5 Casos de uso de Tipo de cirugía

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Crear Tipo Cirugía</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>Permite crear un nuevo tipo de cirugía en el sistema</td>
</tr>
<tr>
<td>Actores:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Usuario Logado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- El usuario tiene que estar logrado.</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td></td>
</tr>
<tr>
<td>1. El actor pulsa el botón para dar de alta nuevo Tipo Cirugía</td>
<td></td>
</tr>
<tr>
<td>2. El sistema muestra una vista donde informar de los datos del nuevo tipo de cirugía</td>
<td></td>
</tr>
<tr>
<td>3. El actor introduce los datos del nuevo tipo de cirugía</td>
<td></td>
</tr>
<tr>
<td>4. El sistema valida los datos y los almacena</td>
<td></td>
</tr>
<tr>
<td>Flujo Alternativo:</td>
<td></td>
</tr>
<tr>
<td>4. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.</td>
<td></td>
</tr>
</tbody>
</table>
   a. El tipo cirugía ya existe en el sistema. |
   b. El sistema muestra un mensaje y permite corregir los datos. |
<p>| PostCondiciones:     |                         |
|                      | - El tipo cirugía se ha registrado en el sistema. |</p>
<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Modificar Tipo Cirugía</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>El sistema permite elegir una cirugía existente y modificarla</td>
</tr>
<tr>
<td>Actores:</td>
<td>• Usuario Logado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td>• El usuario tiene que estar logrado.</td>
</tr>
<tr>
<td></td>
<td>• La mutua tiene que existir en el sistema</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td>1. El actor pulsa el botón para modificar una cirugía.</td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra una vista con los datos de la cirugía.</td>
</tr>
<tr>
<td></td>
<td>3. El actor modifica los datos de la cirugía</td>
</tr>
<tr>
<td></td>
<td>4. El sistema valida los datos y los almacena</td>
</tr>
<tr>
<td>Flujo Alternativo:</td>
<td>4. El sistema valida los datos y no se cumple alguna de las situaciones siguientes.</td>
</tr>
<tr>
<td></td>
<td>a. La cirugía ya existe en el sistema.</td>
</tr>
<tr>
<td>PostCondiciones:</td>
<td>La cirugía se ha modificado en el sistema.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Consultar cirugía</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
<tr>
<td>Descripción:</td>
<td>El sistema permite elegir una cirugía existente para poder consultarla</td>
</tr>
<tr>
<td>Actores:</td>
<td>• Usuario Logado</td>
</tr>
<tr>
<td>Precondiciones:</td>
<td>• El usuario tiene que estar logrado.</td>
</tr>
<tr>
<td></td>
<td>• La cirugía tiene que existir en el sistema.</td>
</tr>
<tr>
<td>Flujo Normal:</td>
<td>1. El actor selecciona una cirugía para visualizarla dentro de un listado.</td>
</tr>
<tr>
<td></td>
<td>2. El sistema muestra una vista con los datos de la cirugía.</td>
</tr>
<tr>
<td>PostCondiciones:</td>
<td>• La cirugía permanece igual que estaba inicialmente.</td>
</tr>
<tr>
<td>Notas:</td>
<td>El sistema permitirá llamar o enviar email con los datos visualizados.</td>
</tr>
</tbody>
</table>
6.6 Casos de Uso de Usuario

<table>
<thead>
<tr>
<th>Nombre</th>
<th>CU1. Login de Usuario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autor:</td>
<td>David Palazón</td>
</tr>
</tbody>
</table>

Descripción:
El sistema verifica en base de datos externa en caso de existir conexión y en la interna en caso de no tener conexión que el usuario y la contraseña estén autorizados.

Aceros:
- Usuario no logrado

Precondiciones:
- El usuario no logrado.
- El usuario no está bloqueado.

Flujo Normal:
1. El sistema muestra ventana para que el usuario informe de los datos de conexión.
2. El actor introduce los datos
3. El sistema valida los datos con el servidor remoto
4. El sistema registra los datos remotos en el servidor local.

Flujo Alternativo (Datos Erróneos)
4. El sistema valida los datos y los datos de conexión no existen o son erróneos
5. El sistema incrementa el contador de intentos.
6. Si el número de reintentos supera los permitidos bloquea el usuario.
7. Muestra mensaje y da la oportunidad de introducir los datos nuevos.

PostCondiciones:
- El usuario está logrado y entra en el sistema.
7. Diagramas de secuencia.

Los diagramas de secuencia junto a los diagramas de comunicación nos permiten definir cómo interactúan los objetos de una aplicación y sus actores.

Los de secuencia se encargan de mostrar la interacción de los objetos en el tiempo. En los diagramas paso a especificar un diagrama completo y no está limitado a la interacción del actor y el sistema. También quedará patente que se ha tomado la decisión de utilizar el patrón de diseño MVC.

7.1 Solicitudes.

7.1.1 Alta Solicitud.
7.1.2 Modificación Solicitud.

7.1.3 Consultar Solicitud.
7.1.4 Listado Solicitudes.
7.2 Hospital

7.2.1 Alta Hospital

7.2.2 Consulta de Hospital
7.2.3 Modificar Hospital

- **Actor**: ModificarHospital
- **Hospital Edit View**: SetKey, GetHospital, PushView, Edit Campos, Click save button
- **Controller**: Save, Save(Info), PopView

7.2.4 Listado Hospital

- **Actor**: ListadoSolicitudes, Necesita Refresco
- **Listado Hospitales View**: SetList, GetList
- **Controller**: Necesita Refresco
- **Hospital Model**: Necesita Refresco, LoadSolicitudes
- **Hospital Service**: Necesita Refresco
7.3 Casos de uso Mutuas

7.3.1 Listado Mutuas.

7.3.2 Añadir Mutua
7.3.3 Modificar Mutua
7.4 Casos de uso de doctores

7.4.1 Listado de Doctores

7.4.2 Añadir Doctores
7.4.3 Modificar Doctor

7.4.4 Consultar Doctor
7.5 Caso de Uso Tipo cirugía

7.5.1 Listado Tipos de Cirugía

7.5.2 Añadir nuevo tipo de cirugía
7.5.3 Modificar tipo de cirugía

7.5.4 Consultar tipo de cirugía
7.6 Caso de uso login

7.6.1 Login con Conexión
8. Diseño de Interfaz gráfica

La elaboración de la interfaz de usuario ha sido de los puntos, sin duda, más conflictivos del proyecto.

Mi experiencia ha sido siempre sobre entornos que no tenían resoluciones tan dispares, aplicaciones de escritorio donde el framework utilizado (SAP) te da la estética completa haciendo que como programador te desprecueses de este punto...

El entorno móvil es muy similar al entorno web y requiere de muchas técnicas muy similares. Y es un entorno del que he huido porque mis dotes como diseñador gráfico son prácticamente nulas.

El lenguaje Flex que utilizo en el proyecto suele utilizar un lenguaje descriptivo llamado MXML, este lenguaje permite mediante tags muy parecidos al XML o al HTML definir elementos del entorno gráfico e incluso heredarlos y modificarlos. No se centra solo en este punto sino que permite también definir timers, fuentes de datos, conexiones remotas, etc.

Otra posibilidad es usar actionScript e incluso mezclárselos.

En las últimas versiones se han creado unos componentes nuevos llamados SPARK especialmente diseñados para tener un mejor rendimiento en entornos móviles.

Al inicio del proyecto, se realizaron los diseños de pantalla con este entorno, pero a la hora de la verdad me encontré con una desagradable sorpresa que explicaré a continuación.

En Flex una de sus funcionalidades más potentes es la facilidad de generar componentes que llaman itemRenderer que permite mostrar elementos de lista personalizados. Por poner un ejemplo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<!-- containers\spark\myComponents\MySimpleCustomItemRenderer.mxml -->
    xmlns:mx="library://ns.adobe.com/flex/mx"
    xmlns:s="library://ns.adobe.com/flex/spark">
    <s:states>
        <s:State name="normal"/>
        <s:State name="hovered"/>
    </s:states>
    <s:Label id="labelDisplay"
        verticalCenter="0"
        left="3" right="3" top="6" bottom="4"
        fontSize.hovered='14' fontStyle.hovered="italic"/>
</s:ItemRenderer>
```

Esto se introduce dentro de un componente lists que iteracciona por los elementos de una colección y da una bonita lista.

Después de llevar unas jornadas realizando el diseño y desarrollo en entorno de pruebas (emulador flash) decidido traspasarlo a un dispositivo real.

En ese momento me llevo la sorpresa, el desempeño (rendimiento) dejaba mucho que desear. Después de investigar un poco fuera del entorno del “fabricante” mirando en foros me encuentro que ciertamente es algo habitual.
Según la documentación encontrada en estos casos te obliga a programar a más bajo nivel y codificar 100% en Actionscript.

Durante mi búsqueda encontré las recomendaciones de mucha gente en la utilización de una librería que suministraba un nuevo set de componentes hechos desde cero optimizados para dispositivos móviles. Esta librería es open source y tenía disponible todos los fuentes. Además tenía una comunidad muy activa.

Los tres motivos que me hicieron decantarme por esta librería.

- Muy ligera, las apps quedaban reducidas en muchos megas. Con el framework base de Flex 200 Mb una app y la equivalente con este framework 70 mb.
- Componentes muy rápidos y a los que se les podía definir capas de visualización (SKINS)
- Las pantallas aunque estábamos trabajando en puro actionscript se podian definir mediante definiciones de pantalla XML.
- Era responsive, excepto en ocasiones muy concretas no era necesario definir un formato para cada tamaño de pantalla.

8.2 MadComponents.

Lo cierto es que la librería te deja bastante impresionado, el único problema es que el desarrollador es una única persona. Pero debido a la carga de trabajo al final decidió poner el código y cederlas en opensource. Por otro lado creo una comunidad para que los usuarios pudiesen comunicarse y compartir experiencias porque el ya no podía responder a tantas cuestiones. La librería suministra un set reducido de componentes que permiten realizar prácticamente cualquier desarrollo móvil.

En mi caso se complicó un poco más al querer programar con el patrón de diseño MVC puesto que la librería no cargaba dinámicamente las vistas. Esto me obligo a usar un método no documentado para inyectar los componentes una vez la aplicación había cargado.

8.2.1 Definición de pantallas con XML con la librería MadComponents.

Como ya comentaba anteriormente la librería da un juego reducido de componentes, pero más que suficientes para cualquier aplicación. La definición se realiza con una variable XML donde se carga la definición de las diferentes pantallas. Y el problema radica en este punto, al querer implantar el Patrón de Diseño MVC y usando la librería Robotlegs nos encontrábamos que al no ser estándar el sistema no podía detectar cuando se creaba una vista para poder inyectar sus dependencias al mediador.

Para ello al final tuve que usar una extensión de la librería no muy documentada que permite extenderla la librería con nuevos componentes del lenguaje estandard. Esto lo utilizamos para crear componentes con nuestras pantallas y al extenderlas el sistema es capaz de realizar la inyección.

A continuación muestro algunas de las pantallas de la aplicación y sus definiciones XML en MadComponents.
Pantalla de Login

Entrar

Login Demo

¿No recuerdas password?
9. Modelo de componentes

El modelo seguido en el proyecto está basado en un patrón de diseño modelo vista controlador. Concretamente basado en el modelo suministrado por la librería Robotlegs que corresponde al modelo MVCS donde la S se refiere a servicio.

En este patrón la capa de modelo se divide en dos. En modelo y servicio donde modelo se encarga de tratar la colección de datos (Es un repositorio) y el de servicio se encarga de toda la interacción con el exterior.

En la parte de la vista también se diferencia, todas las vistas tienen una clase especial llamada mediador. Este modelo permite desacoplar el tratamiento de los datos de cómo se persisten.

Listado de componentes.

A continuación paso a detallar los diferentes componentes del sistema y sus métodos y visibilidades. En la practica la aplicación se ha realizado utilizando inyección de dependencias basada en Interfaces de esa forma fijábamos un contrato a cumplir, pero como se implementará no afecta al sistema. Esto es buena práctica para poder realizar cambios rápidamente y ayuda a los test unitarios.

En nuestro caso hay una clase especial que hace de gestor de eventos, en nuestro caso es el controlador (ApplicationMediator).

<table>
<thead>
<tr>
<th>DoctorVO</th>
<th></th>
</tr>
</thead>
</table>
| **Atributos** |  + DOCTOR:String  
|   |  + NOMBRE:String  
|   |  + PRIMER_APELLIDO:String  
|   |  + SEGUNDO_APELLIDO:String  
|   |  + TELEFONO:String  
|   |  + MOVIL:String  
|   |  + EMAIL:String  
|   |  + DNI:String  
|   |  + NUMERO_COLEGIADO:String  
|   |  + ESPECIALIDAD:String  
|   |  + NOTAS:String  |
| **Métodos** |  + Update(doctor:DoctorVO):void  
|   |  + NOMBRECOMPLETO():String  
|   |  +toArray():Array  |

<table>
<thead>
<tr>
<th>HospitalVO</th>
<th></th>
</tr>
</thead>
</table>
| **Atributos** |  + idHospital:int  
|   |  + ERPCODE:String  
|   |  + Hospital:String  
|   |  + Ciudad/String  
|   |  + EMail:String  
|   |  + Telefono:String  
|   |  + IdProvincia:int  |
| **Métodos** |  + Update(HospitalVO):void  
|   |  +toArray():Array  |

<table>
<thead>
<tr>
<th>MutuaVO</th>
<th></th>
</tr>
</thead>
</table>
| **Atributos** |  idMutua:String  
|   |  Mutua:String  |
| **Métodos** |  + Update(MutuaVO):void  
<p>|   |  +toArray():Array  |</p>
<table>
<thead>
<tr>
<th>Atributos</th>
<th>Solicitud</th>
</tr>
</thead>
<tbody>
<tr>
<td>+nId:int</td>
<td></td>
</tr>
<tr>
<td>+IdSolicitud:int</td>
<td></td>
</tr>
<tr>
<td>+IdHospital:int</td>
<td></td>
</tr>
<tr>
<td>+NombreHospital: String</td>
<td></td>
</tr>
<tr>
<td>+NombreSolicitud: String</td>
<td></td>
</tr>
<tr>
<td>+ID_DOCTOR:String</td>
<td></td>
</tr>
<tr>
<td>+NombreDoctor: String</td>
<td></td>
</tr>
<tr>
<td>+FechaCreacion: String</td>
<td></td>
</tr>
<tr>
<td>+FechaCirugia: String</td>
<td></td>
</tr>
<tr>
<td>+IdEstado: String</td>
<td></td>
</tr>
<tr>
<td>+idUser:int</td>
<td></td>
</tr>
<tr>
<td>+Tipo_centro:String +Particular:Boolean</td>
<td></td>
</tr>
<tr>
<td>+IdMutua: String</td>
<td></td>
</tr>
<tr>
<td>+Mutua: String</td>
<td></td>
</tr>
<tr>
<td>+Nombre: String</td>
<td></td>
</tr>
<tr>
<td>+Primer_Apellido: String</td>
<td></td>
</tr>
<tr>
<td>+Segundo_Apellido: String</td>
<td></td>
</tr>
<tr>
<td>+Autorizacion: String</td>
<td></td>
</tr>
<tr>
<td>+NumeroPoliza: String</td>
<td></td>
</tr>
<tr>
<td>+Telefono: String</td>
<td></td>
</tr>
<tr>
<td>+idCirugia1:int</td>
<td></td>
</tr>
<tr>
<td>+idCirugia2:int</td>
<td></td>
</tr>
<tr>
<td>+idCirugia3:int</td>
<td></td>
</tr>
<tr>
<td>+DescCirugia: String</td>
<td></td>
</tr>
<tr>
<td>+DescCirugia2: String</td>
<td></td>
</tr>
<tr>
<td>+DescCirugia3: String</td>
<td></td>
</tr>
<tr>
<td>+Comentarios: String</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Métodos</th>
<th>Solicitud</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Update(SolicitudVO): void</td>
<td></td>
</tr>
<tr>
<td>+ toArray(): Array</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Atributos</th>
<th>TipoCirugiaVO</th>
</tr>
</thead>
<tbody>
<tr>
<td>+idCirugia:int</td>
<td></td>
</tr>
<tr>
<td>+Cirugia: String</td>
<td></td>
</tr>
<tr>
<td>+CostePromedio: Number</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Métodos</th>
<th>TipoCirugiaVO</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Update(SolicitudVO): void</td>
<td></td>
</tr>
<tr>
<td>+ toArray(): Array</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Atributos</th>
<th>UsuarioVO</th>
</tr>
</thead>
<tbody>
<tr>
<td>+IDUSUARIO:String</td>
<td></td>
</tr>
<tr>
<td>+LOGIN: String</td>
<td></td>
</tr>
<tr>
<td>+PASSWORD: String</td>
<td></td>
</tr>
<tr>
<td>+LOGINSERROR:int</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Métodos</th>
<th>UsuarioVO</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Update(SolicitudVO): void</td>
<td></td>
</tr>
<tr>
<td>+ toArray(): Array</td>
<td></td>
</tr>
</tbody>
</table>
### DoctorModel

**Properties**
- onSelected():Signal
- onDeleted():Signal
- list():ArrayCollection
- NeedLoad():Boolean
- selected():DoctorVO
- lastload():Date

**Methods**
- create():DoctorVO
- remove(DoctorVO):void
- GetDoctor(String):DoctorVO

### HospitalModel

**Properties**
- onSelected():Signal
- onDeleted():Signal
- list():ArrayCollection
- NeedLoad():Boolean
- selected():HospitalVO
- lastload():Date

**Methods**
- create():HospitalVO
- remove(HospitalVO):void
- GetDoctor(String):DoctorVO

### MutaModel

**Properties**
- onSelected():Signal
- onDeleted():Signal
- list():ArrayCollection
- NeedLoad():Boolean
- selected():MutuaVO
- lastload():Date

**Methods**
- create():MutuaVO
- remove(MutuaVO):void
- GetMutua(String):MutuaVO

### SolicitudModel

**Properties**
- onSelected():Signal
- onDeleted():Signal
- list():ArrayCollection
- NeedLoad():Boolean
- selected():SolicitudVO
- lastload():Date

**Methods**
- create():SolicitudVO
- remove(SolicitudVO):void
- GetMutua(String):SolicitudVO

### TiposdeCirugiaModel

**Properties**
- onSelected():Signal
- onDeleted():Signal
- list():ArrayCollection
- NeedLoad():Boolean
- selected():TiposCirugiaVO
- lastload():Date

**Methods**
- create():TiposCirugiaVO
- remove(TiposCirugiaVO):void
- GetTIpoCirugia(String):TiposCirugiaVO

### UsuarioModel

**Properties**
- onSelected():Signal
- onDeleted():Signal
- list():ArrayCollection
### DoctorService

**Atributos**
- log:ILogger
- ADD_DOCTOR_SQL: String
- UPDATE_DOCTOR_SQL: String
- LOAD_DOCTOR_SQL: String
- LOAD_DOCTORES_SQL: String
- DELETE_Doctor_SQL: String
- Model: DoctorModel
- sqlRunner: SQLRunner
- nuevoDoctor: DoctorModel

**Properties**
- onDeleted(): Signal
- onLoaded(): Signal
- onSaved(): Signal
- onError(): Signal

**Methods**
- load(): void
- save(DoctorVO): void
- remove(DoctorVO): void

### MutuaService

**Atributos**
- log:ILogger
- ADD_MUTUA_SQL: String
- UPDATE_MUTUA_SQL: String
- LOAD_MUTUA_SQL: String
- LOAD_MUTUAS_SQL: String
- DELETE_Mutua_SQL: String
- Model: MutuaModel
- sqlRunner: SQLRunner
- nuevoMutua: MutuaModel

**Properties**
- onDeleted(): Signal
- onLoaded(): Signal
- onSaved(): Signal
- onError(): Signal

**Methods**
- load(): void
- save(MutuaVO): void
- remove(MutuaVO): void

### HospitalService

**Atributos**
- log:ILogger
- ADD_HOSPITAL_SQL: String
- UPDATE_HOSPITAL_SQL: String
- LOAD_HOSPITAL_SQL: String
- LOAD_HOSPITALES_SQL: String
- DELETE_Hospital_SQL: String
- Model: HospitalModel
- sqlRunner: SQLRunner
- nuevoHospital: HospitalModel

**Propiedades**
- onDeleted(): Signal
- onLoaded(): Signal
- onSaved(): Signal
- onError(): Signal

**Métodos**
- load(): void
- save(HospitalVO): void
- remove(HospitalVO): void
| TipoCirugiaService | Atributos | log:ILogger  
| | | ADD_TIPOCIRUGIA_SQL:String  
| | | UPDATE_TIPOCIRUGIA_SQL:String  
| | | LOAD_TIPOCIRUGIA_SQL:String  
| | | LOAD_TIPOSCIRUGIA_SQL:String  
| | | DELETE_TipoCirugia.SQL:String  
| | | Model:TipoCirugiaModel  
| | | sqlRunner:SQLRunner  
| | | nuevoTipoCirugia:TipoCirugiaModel  
| | Propiedades | onDeleted():Signal  
| | | onLoaded():Signal  
| | | onSaved():Signal  
| | | onError():Signal  
| | Métodos | load():void  
| | | save(TipoCirugiaVO):void  
| | | remove(TipoCirugiaVO):void |
10. Diseño de la capa de gestión de datos

10.1 Diseño de la base de datos.

Una de las particularidades de las bases de datos SQLite es que tiene muy limitados los tipos de datos a salvaguardar, además de que aunque se definen el tipo de columna, sea el tipo del valor a guardar el que prevalece. Aunque la base de datos puede intentar hacer conversiones de tipo dando errores.

- Los valores que he utilizado son:
  - Integer
  - Real
  - Text
  - Blob
  - Date (aunque lo tiene, debido a problemas con su tratamiento) he optado por usar enteros y guardar los datos como numérico con formato yyyymmdd.

10.2 Definición de las tablas

<table>
<thead>
<tr>
<th>Tabla</th>
<th>Descripción</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cirugías</td>
<td>Almacena las descripciones de los tipos de cirugías disponibles</td>
</tr>
<tr>
<td>Doctor</td>
<td>Guarda los datos de los doctores disponibles</td>
</tr>
<tr>
<td>Hospitales</td>
<td>Guarda los Hospitales con los que trabajamos</td>
</tr>
<tr>
<td>Mutuas</td>
<td>Guarda la relación de mutuas</td>
</tr>
<tr>
<td>Solicitudes</td>
<td>Guarda los datos de las cirugías que se han realizado o se van a realizar</td>
</tr>
<tr>
<td>Usuario</td>
<td>Guarda la lista de usuarios del sistema.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tabla: Cirugías</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>idCirugia</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Cirugia</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>CostePromedio</td>
<td>NUMERIC(10,2)</td>
</tr>
<tr>
<td>MD5</td>
<td>VARCHAR(32)</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DELETED</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>idCirugia</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tabla: Doctor</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_DOCTOR</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>DOCTOR</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>NOMBRE</td>
<td>VARCHAR(100)</td>
</tr>
<tr>
<td>PRIMER_APELLIDO</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>SEGUNDO_APELLIDO</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>TELEFONO</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>MOVIL</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>DNI</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>NUMERO_COLEGIADO</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>ESPECIALIDAD</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>NOTAS</td>
<td>TEXT</td>
</tr>
<tr>
<td>EMAIL</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>MD5</td>
<td>VARCHAR(32)</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DELETED</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>ID_DOCTOR</td>
</tr>
<tr>
<td>Tabla: Hospital</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------</td>
</tr>
<tr>
<td>idHospital</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>Hospital</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>Ciudad</td>
<td>VARCHAR(100)</td>
</tr>
<tr>
<td>Provincia</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>TELEFONO</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>TELEFONO</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>EMAIL</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>MD5</td>
<td>VARCHAR(32)</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DELETED</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>idHospital</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tabla: Mutua</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>idMutua</td>
<td>VARCHAR(20)</td>
</tr>
<tr>
<td>Mutua</td>
<td>VARCHAR(200)</td>
</tr>
<tr>
<td>MD5</td>
<td>VARCHAR(32)</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DELETED</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>idMutua</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tabla:Solicitud</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>nId</td>
<td>INTEGER</td>
</tr>
<tr>
<td>IdSolicitud</td>
<td>INTEGER</td>
</tr>
<tr>
<td>IdHospital</td>
<td>VARCHAR( 10 ),</td>
</tr>
<tr>
<td>ID_DOCTOR</td>
<td>VARCHAR( 12 )</td>
</tr>
<tr>
<td>FechaCreacion</td>
<td>VARCHAR( 10 )</td>
</tr>
<tr>
<td>IdEstado</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>IdUser</td>
<td>INTEGER</td>
</tr>
<tr>
<td>tipo_centro</td>
<td>BOOLEAN( 1 )</td>
</tr>
<tr>
<td>Particular</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>idMutua</td>
<td>VARCHAR( 10 )</td>
</tr>
<tr>
<td>NumeroPoliza</td>
<td>VARCHAR( 20 ),</td>
</tr>
<tr>
<td>Nombre</td>
<td>VARCHAR( 200 )</td>
</tr>
<tr>
<td>Telefono</td>
<td>VARCHAR( 100 )</td>
</tr>
<tr>
<td>idCirugia1</td>
<td>VARCHAR( 20 ),</td>
</tr>
<tr>
<td>idCirugia2</td>
<td>VARCHAR( 20 ),</td>
</tr>
<tr>
<td>idCirugia3</td>
<td>VARCHAR( 20 ),</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
</tr>
<tr>
<td>MD5</td>
<td>VARCHAR( 32 ),</td>
</tr>
<tr>
<td>DELETED</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>FechaCirugia</td>
<td>VARCHAR,</td>
</tr>
<tr>
<td>Comentarios</td>
<td>VARCHAR( 500 )</td>
</tr>
<tr>
<td>Primer_Apellido</td>
<td>VARCHAR( 200 )</td>
</tr>
<tr>
<td>Segundo_Apellido</td>
<td>VARCHAR( 200 )</td>
</tr>
<tr>
<td>Autorizacion</td>
<td>VARCHAR( 200 )</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>nId</td>
</tr>
<tr>
<td>FOREIGN KEY</td>
<td>ID_DOCTOR,MUTUAS,HOSPITAL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tabla:Usuario</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>idUsuario</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Login</td>
<td>VARCHAR( 200 ),</td>
</tr>
<tr>
<td>Password</td>
<td>VARCHAR( 200 )</td>
</tr>
<tr>
<td>VERSION</td>
<td>INTEGER</td>
</tr>
<tr>
<td>MD5</td>
<td>VARCHAR( 32 ),</td>
</tr>
<tr>
<td>DELETED</td>
<td>BOOLEAN</td>
</tr>
<tr>
<td>PRIMARY KEY</td>
<td>idUsuario</td>
</tr>
</tbody>
</table>
11. Implementación y test

11.1 Metodología Empleada.

Debido al carácter documentado que necesita un proyecto de final de carrera se ha utilizado una metodología clásica. Que en la realidad no suele utilizarse demasiado por los siguientes motivos.

Se pasa mucho tiempo planificando con lo que para clientes pequeños el coste de horas se dispara sin tener resultados visibles. Muchas veces los requisitos varían con el tiempo y el usuario no suele ser capaz de transmitirte realmente toda su problemática.

Por este motivo son más interesantes las metodologías circulares Cruz, Agile,… Donde se preparan Spring marcando hitos a corto plazo y se van entregando prototipos donde el cliente está involucrado detectando de forma temprana cualquier error.

El diseño que se ha empleado en la generación del prototipo está orientada a este tipo de desarrollo, se ha intentado que la aplicación este altamente desacoplada utilizando inyección de dependencias con lo que sería muy fácil realizar test de pruebas unitarios con alguna herramienta. También he implementado la inversión de control. Metodología más adecuada para la TDD.

La comunicación entre los diferentes módulos se hacen mediante Señales y Commands. Los niveles superiores no tienen conocimiento de los inferiores.

11.2 Implementación.

A nivel de desarrollo se han utilizado dos herramientas, inicialmente se probó con una herramienta OpenSource que funcionaba correctamente al principio pero que a medida que el proyecto empezó a crecer se volvió más instable, cada Señal es una clase, las vistas son 2 ficheros y el modelo son 2 ficheros por entidad, El emulador se cerraba inesperadamente, devolvía errores que no eran tales, etc…. Esto me invito después de un periodo de pruebas a adquir un software de la marca JetBrains que para programadores particulares tiene un precio muy asequible ronda los 179 € y te permite trabajar con una herramienta profesional.

Tal como he comentado para la implementación del MVC y la inversión de Control e Inversión de Dependencias he utilizado la librería hecha en actionscript puro Robotlegs v2. Esto me ha obligado ha ser muy estructurado, pero ha provocado que el proyecto a nivel de ficheros se vuelva monstruoso. Si además sumamos que se toma la decisión de implementar la inyección de componentes mediante los interfaces significa que para clase se ha generado un fichero de interface.

La estructura del proyecto es la que paso a detallar a continuación.

- **Carpeta Contex:** Aquí se definen los ficheros correspondientes al contexto, en la versión 2, son básicamente fichero de configuración donde indicamos que dependencias tenemos que inyectar, las señales a tratar y que comandos se encargaran de realizarlas.
- **Carpeta Controllers:** Esta carpeta tiene dos subcarteras la correspondiente a las Señales y los comandos. No he utilizado el sistema de eventos estándar puesto que eran más complejos para incluirlos en los interfaces.
- **Carpeta Model:** Es donde tendremos las clases responsables de tratar con los datos. El modelo de robotlegs difiere un poco del estándar (no es necesario usarlo) puesto que separa el modelo de datos. En modelos (son nuestras colecciones, que solo saben cómo trabajar con los datos, devolverlos, filtrarlos, etc. y por otro lado tenemos la capa de servicios que es la encargada de persistirlos ya sea mediante accesos a base de datos o servicios remotos. Estos se encargan de actualizar el modelo si hay algún cambio.
- **Carpeta View:** Aquí tendremos todas las clases encargada de la capa de visual. Por un lado
tenemos las vistas propiamente dichas que se encargarán solo de visualizar y validar los datos de pantalla que no necesiten acceder a datos almacenados. Y luego tendremos los mediators (mediadores) que se encargarán de comunicar la vista con el modelo de datos lanzando eventos.

A nivel de la interfaz gráfica la idea como ya se ha comentado anteriormente era utilizar los componentes que daba el framework de Flex y sus ficheros de definición mxml., pero debido al problema de rendimiento se ha tenido que hacer una programación a más bajo nivel y pasar a programar en actionscript.

Por suerte la librería MadCommand que es opensource mejora rendimiento y tamaño de la aplicación que en los dispositivos moviles es realmente importante. Además no perdíamos la facilidad de diseño porque utiliza definiciones en XML para la definición de las vistas

Para la interfaz gráfica la librería MadCommands, que además suministra una librería para la elaboración de gráficos.

El framework actionscript utilizado ha sido Apache Flex 4.12.1

11.3 Testing

Durante la implementación han surgido muchísimos problemas, desarrollar apps para IOS en Windows no es un lecho de rosas. Pero lo cierto es que la experiencia mejoro al cambiar la herramienta. La herramienta tiene una característica muy interesante y es que se integra con la aplicación FlexUnit para realizar pruebas unitarias.

En mi caso solo he podido realizar pruebas unitarias de la capa de datos. Testear la interface es muy complejo así que lo tienes que hacer en el proceso de pruebas de integración.

Otro problema importante que me he encontrado ha sido que a veces en las maquinas virtuales de test no tienen el mismo comportamiento que en el dispositivo móvil y cosas que te funcionan correctamente en el pc luego no corren bien en el móvil.

De esto puedo sacar la conclusión de que si tienes el dispositivo de destino es mejor probarlo directamente. El proceso es más lento pero más fiable.
12. Conclusiones y trabajos Futuros.

12.1 Conclusiones.

Personalmente me siento muy satisfecho porque después de muchas horas invertidas he logrado solucionar problemas que me parecían realmente grandes escollos.

En referencia a los objetivos que me marqué en un principio quizás fueron demasiado altos y esto me ha ido desgastando al ver que no obtenía los resultados deseados. Tal como incurren las metodologías modernas lo mejor es marcarte pequeños hitos para ver un resultado pronto y así ganar confianza.

Que puedo decir, pues que he sentido lo que hacía tiempo que no sentía, esa desesperación de no encontrar una solución y tener que investigar horas para sacar algo en claro, básicamente volver al punto de partida, no tener experiencia en un producto y tener que tirar de la documentación.

Pero aparte de eso lo cierto es que estoy realmente satisfecho, he aprendido un lenguaje nuevo, he asimilado nuevos patrones de diseño y me veo capaz de empezar a desarrollar en dispositivos móviles.

Respecto a problemas remarcables en el proyecto la verdad es que he tenido cuatro bastante grandes para probarlos.

- El primero era la cantidad de tiempo que tenía que invertir para encontrar la solución a un problema. Como siempre con el software libre hay cientos de sitios donde te dan una solución que no funciona. Incluso a veces he tenido que deshacer parte del trabajo.
- La falta de experiencia en el producto. A la hora de programar me costaba encontrar soluciones a problemas ya conocidos que en los lenguajes que uso habitualmente puedo resolver rápidamente.
- La división del producto en Adobe y Apache hace que cierta documentación no sea del todo fiable.
- EL seguimiento del proyecto. He coincidido con el arranque de dos empresas de la sociedad en que trabajo y un cambio de partner. Esto me ha reducido el tiempo muchísimo.

Por la parte del diseño, lo cierto es que será mejor que deje eso en manos de los profesionales como he hecho hasta ahora.

12.2 Trabajos futuros.

Puntos a mejorar para que la aplicación fuese más completa.

- Mejorar la sincronización. Sería interesante poner un sistema más complejo de sincronización para poder tener una comunicación directa con el ERP de la organización.
- Versión de escritorio Aprovechando la separación de la capa visual y que el entorno es multiplataforma se podría realizar una versión para escritorio.
- Hacer cambios en algunas tablas y modificar las vistas. Hay campos que se podrían pasar a tablas para poder aumentar la cantidad de información. Ejemplos múltiples fotos, mas cirugías por solicitud.
- Añadir informes o estadísticas que el usuario pueda necesitar.

- Pascal Roques. UML 2 par la pratique, Études de cas et exercices corriges. 5ª Edición 2006.
- MADCOMPONENTS. https://code.google.com/p/mad-components/
- Guía de diseño IOS7. http://ivomynttinen.com/blog/the-ios-7-design-cheat-sheet/
- IntelliJdea http://www.jetbrains.com/idea/buy/
- RiaActive http://riactive.mx/2011/02/10/robotlegs-101-accion/
- Flexenespannol http://www.flexenespannol.com/
Apéndices.

Manual de instalación

Para poder realizar la compilación son necesarios tal como se indicaba en el apartado de implantación el entorno de desarrollo de JetBrains IntelliJ IDEA o en su defecto FlashDevelop. Una vez tenemos el entorno será necesario bajar las librerías y frameworks necesarios.


Las librerías utilizadas se pueden bajar desde aquí, pero ya están en el mismo proyecto.

Los componentes de MADCOMPONENTS se descargan desde la web https://code.google.com/p/mad-components/. Para su correcto funcionamiento los tuve que compilar con la versión de Flex que iba a utilizar en el proyecto.

La librería de inyección de dependencias la podemos descargar desde http://www.robotlegs.org/ descargar la versión 2.5.1


Una vez los tenemos bastara con copiarlos e indicarle en el proyecto File → Proyect Structure → IOS el provisionning Key y el Keystore file.

Ahora ya podemos compilar he instalarlo en el móvil.

Si lo queremos ejecutar en el emulador nos podemos ahorrar estos pasos.