Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

Aplicación cliente destinada a obtener una diagnosis consensuada entre varios médicos.

Andrés Cenci Alonso

Director: Simone Balocco
Realitzat a : Departament de Matemàtica
Aplicada i Anàlisi. UB
Barcelona, 8 de enero de 2014

Indice

Summary

This paper in front of you is the resulting report and explanation of a final degree work. The project discussed here is to create a client application that belongs to the field of telemedicine. It aims to facilitate the doctors the decision making in cases of doubtful diagnosis.

The execution process can be summarized in three steps:
first user, ie a doctor, creates a question about a diagnosis which is not completely secure.
Since then, the other users can give their opinion on the matter and thus contribute to solving the doubt. Finally, the user who created the question decides to close it, and the system shows what it estimates that is the correct answer. The estimation is based in responses that others have given.

This work belongs to a wider project, that consists in three parts:
The client application, that we will discuss here.
A server and data base, that will be explained in another document.
And a third part, in charge of the decision making process. This part has not been finished because the final team was only of two members.

This document contains three basic sections:
The first one is an introduction that contains the explanation of the problem and the planned solution in detail, a resume of the technologies used to implement that solution and the time planification that we have followed.
The second part, focuses on the design of the client application and the most important details of the implementation. This explanation is done starting with the major functionalities of the application and then explaining how they spread through the different classes implemented.
The third part consists in a guide for users. This section shows a complete example of execution, so the user can follow step by step in a walk-through by all the functionalities of the application.

It has also been created an annex containing useful information about the client implementation (like the full class diagram or the javadoc) and a glossary of some technical terms that do constant reference.

Introducción

2.1.- Motivación.

La motivación de este proyecto viene dada por dos motivos.
El primero es mi gran interés personal en el desarrollo para las nuevas plataformas como Android. Veo en estas plataformas grandes posibilidades laborales y un buena proyección de futuro, además, considero que una buena experiencia con este tipo de plataformas proporciona una gran oportunidad para emprender proyectos propios.
El segundo es la posibilidad de participar en la competición de aplicaciones del *Mobile World Congres*. Creo que puede ser una buena experiencia, puesto que nos permitiría medir nuestras cualidades y la calidad de nuestro trabajo con otros estudiantes y equipos a nivel mundial.
Todo ello, combinado con la necesidad de realizar el mejor trabajo de final de grado posible, hace que esta elección se convierta en un gran oportunidad de obtener una experiencia de trabajo muy valiosa.

2.2.- Explicación del proyecto.

En nuestra experiencia como informáticos, hemos descubierto que las preguntas y ayudas entre colegas son un recurso muy válido y recomendable, además de útil. Por tanto, creemos que esa experiencia debería poder exportarse al ámbito de la medicina.
En definitiva, este proyecto se plantea como una aplicación de tele-medicina, que intenta facilitar a un médico la toma de decisiones en los casos en que se presentan dudas sobre un determinado diagnostico.

Esta aplicación permitiría a sus usuarios generar consultas, es decir, preguntas o cuestiones sobre casos de diagnostico en los que no se encuentran absolutamente seguros. Este uso no es absolutamente exclusivo, puesto que también podría usarse como un medio de responder dudas conceptuales o cuestiones de índole más genérica.
Concretamente, una vez que un usuario ha generado una cuestión, se les presentará a los demás usuarios para que puedan responder o dar su opinión al respecto. El sistema tendría entonces que almacenar la información de todas esas respuestas y elegir una de las posibles como válida y correcta.

El funcionamiento podría resumirse en los siguientes pasos:

1. Un usuario crea una pregunta.
2. Los demás usuarios dan su opinión.
3. El sistema valora las opiniones y decide cual es la que considera más correcta.

En el siguiente punto profundizaremos mejor en el planteamiento de la solución que hemos pensado para este problema.

2.3.- Solución planteada.

En primer lugar, dadas las funcionalidades de la aplicación planteada en el apartado anterior, pensamos que son necesarios tres módulos independientes que se encarguen de diferentes aspectos del proyecto.
Esos tres módulos son:

1. Una *aplicación cliente*, con la que los usuarios interactúan.
2. Un *servidor y base de datos*, que permite comunicar a los usuarios entre sí.
3. Un programa de *toma de decisiones* encargado de valorar las opiniones de los médicos y decidir cual de ellas es la más correcta.

A continuación, entraremos en los detalles de las responsabilidades y funcionalidades de cada uno de los módulos independientes mencionados anteriormente, y de esta manera, completaremos la explicación conceptual de la solución que hemos planteado.

La *aplicación cliente* se encargará de:

- Facilitar un formulario mediante el cual sea posible generar una nueva cuestión.
- Mostrar listas con lasa consultas que hemos generado, las consultas que podemos contestar y las consultas que ya han sido resueltas.
- Mostrar detalles de cada una de las consultas en cada una de las secciones referidas antes.
- Permitir al usuario cerrar un consulta.
- Permitir al usuario contestar una consulta.
- Mantener una copia de la información en el disco local.
- Implementar un sistema que permita enviar y recibir mensajes al servidor.
- Implementar un sistema de control de errores en tres niveles:
    1. Errores en la ejecución de un proceso en el servidor.
    2. Errores en la conexión y/o transmisión de datos del servidor.
    3. Errores en la ejecución de alguna funcionalidad local.

El *servidor y base de datos* gestionará:

- La base de datos mantendrá toda la información necesaria para el correcto funcionamiento tanto del servidor como del cliente.
- El servidor será el encargado de recibir e interpretar las peticiones del cliente para:
    1. Añadir nuevas consultas a la base de datos.
    2. Devolver la información que el cliente necesite.
    3. Permitir modificar las tablas de la base de datos cuando sea necesario.
- Implementar un sistema de control de errores.
- Mantener un protocolo de conexión con el cliente.

El programa de *toma de decisiones,* se encargará de:

- Mantener un indicador de confianza asociado a cada médico.
- Decidir cual de las posibles respuestas es correcta, cuando el usuario decide cerrar una de las consultas que había creado.

La siguiente figura (figura 2.3.1), muestra de forma esquemática, la distribución de los tres módulos del proyecto y sus conexiones.

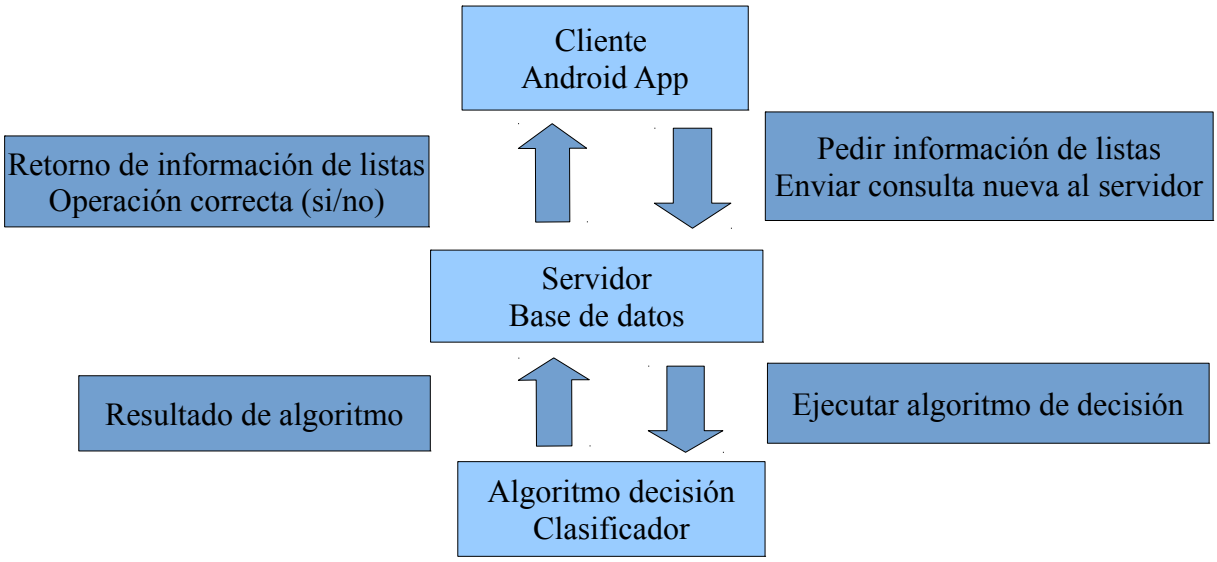


Figura 2.3.1: Esquema de la distribución de los tres módulos del proyecto y sus conexiones.

Finalmente, aclarar que, dado que nuestro equipo de trabajo está formado por dos integrantes, hemos asumido las responsabilidades sobre la aplicación cliente y el servidor – base de datos. El tercer modulo, el programa de toma de decisiones, no ha podido implementarse. Por tanto, creemos que podría realizarse en un futuro, mediante otro proyecto.

En el siguiente punto, explicaremos las tecnologías que cada uno ha usado para solventar su parte del problema. En este caso la aplicación cliente.

2.4.- Tecnología utilizada.

En este punto especificaremos las tecnologías que han sido empleadas para implementar y probar la aplicación cliente.

La aplicación se ha realizado para el S.O. Android, puesto que, además de ser el más difundido, las herramientas para el desarrollo en el mismo son totalmente libres y gratuitas. Se ha desarrollado pensando en la compatibilidad a partir de la versión mínima del SDK 11.

Para mejorar la experiencia del usuario, se han utilizado dos librerías externas: *ActionBarSherlock* y *SlidingMenu*. A continuación entraremos en detalles sobre las funcionalidades proporcionadas por cada una.

ActionBarSherlock

Esta librería es una extensión de la librería de soporte oficial, está diseñada para facilitar el uso del 'action bar' a través de diferentes versiones del S.O. La librería usará automáticamente la barra nativa si es apropiado o, encaso contrario, usará una implementación personalizada. Gracias a esto el diseño e implementación del 'action bar' se simplifica en gran medida. En las secciones 3.4, 3.5, 3.6 y 3.7 señalaremos como y donde se utiliza esta librería.

SlidingMenu

Es una librería que nos permite crear menús deslizantes fácilmente. Estos menús imitan los que vemos en aplicaciones como Google+, YouTube o Facebook. Al implementarla, por tanto, conseguimos un diseño y navegación acorde a las aplicaciones más populares. Esta librería puede configurarse para interaccionar con la anterior. En las secciones 3.4, 3.5, 3.6 y 3.7 señalaremos como y donde se utiliza.

Las herramientas utilizadas para el desarrollo, han sido: *eclipse SDK* con el plug-in de desarrollo *android adt*

Para realizar las pruebas y tests de la aplicación cliente se han usado los siguientes dispositivos:

Tabletas

Motorola XOOM 2

- Pantalla de 10,1 pulgadas
- Procesador dual-core a 1.2 Ghz
- 1 GB de RAM
- Android 3.2

Samsung Galaxy Tab 2

- Pantalla de 10.1 pulgadas
- Procesador dual-core a 1 Ghz
- 1 GB de RAM
- Android 4.1.2

Teléfonos móviles

Samsung Galaxy S I

- Pantalla de 4 pulgadas
- Procesador a 1 Ghz
- 512MB de RAM
- Android 4.0

Samsung Galaxy S III

- Pantalla de 4.8 pulgadas
- Procesador cuad-core a 1.4 Ghz
- 1GB de RAM
- Android 4.1.2

En las pruebas realizadas con estos dispositivos, no se han encontrado fallos específicos relativos al aparato. El funcionamiento ha sido igual de filudo en todos los dispositivos y todas las interfaces diseñadas se han visualizado correctamente.

2.5.- Etapas del proyecto.

En este apartado describiremos la planificación general del proyecto. Nos ocuparemos primero de la planificación conjunta y coordinación de los dos proyectos (cliente y servidor). A continuación, repasaremos en detalle la planificación individual de las tareas de este proyecto (cliente).

2.5.1.- Diagrama de Gantt teórico común.

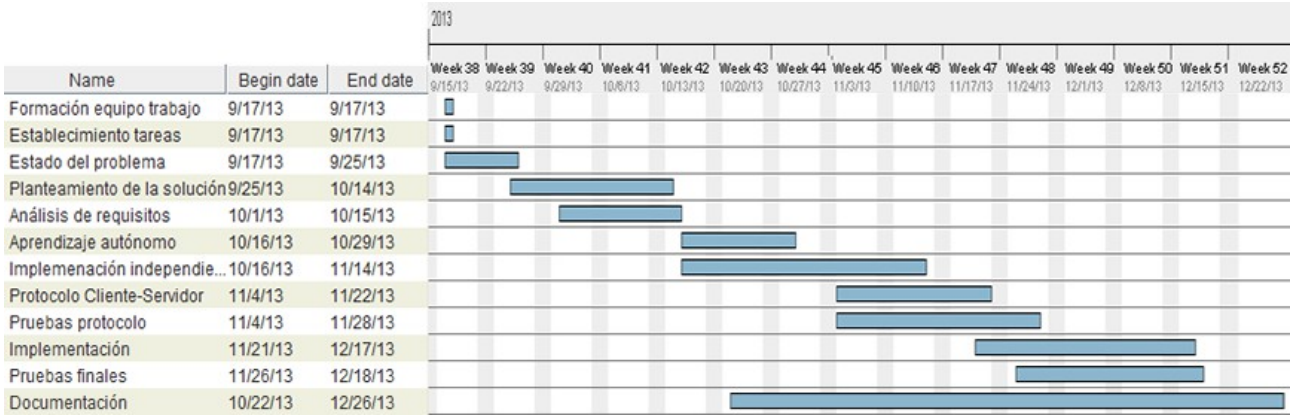
En la siguiente figura (2.5.1.1), tenemos representadas las tareas del proyecto en detalle. Podemos agruparlas en tres grupos: *implementación*, *pruebas* y *tareas accesorias*.

Las *tareas accesorias*, incluyen la formación del equipo de trabajo, la distribución de competencias, el análisis del problema, planteamiento de la solución y el análisis de requisitos. Todas estas tareas se planificaron como lo más esencial y lo primero a realizar.

Implementación, incluye las dos fases de implementación previstas. La primera consistiría en la implementación de aquellas funciones que son propias de los proyectos individuales. La segunda se refiere a las funcionalidades que son interdependientes entre los dos proyectos, comunicación, envío y recepción de datos, etc. (detallaremos todos estos aspectos en el punto 2.5.3). Estas tareas ocuparían la mayor cantidad de tiempo.

Las *pruebas*, se refieren a los periodos de tiempo empleados en testar las funcionalidades implementadas antes y corregir los fallos encontrados.

Como podemos ver, las tareas accesorias debían realizarse en primer lugar, seguidas de una primera fase de implementación y sus pruebas correspondientes. Finalmente, un proceso de integración de las dos partes y sus pruebas. La documentación se había pensado como un proceso continuo paralelo al desarrollo del proyecto.

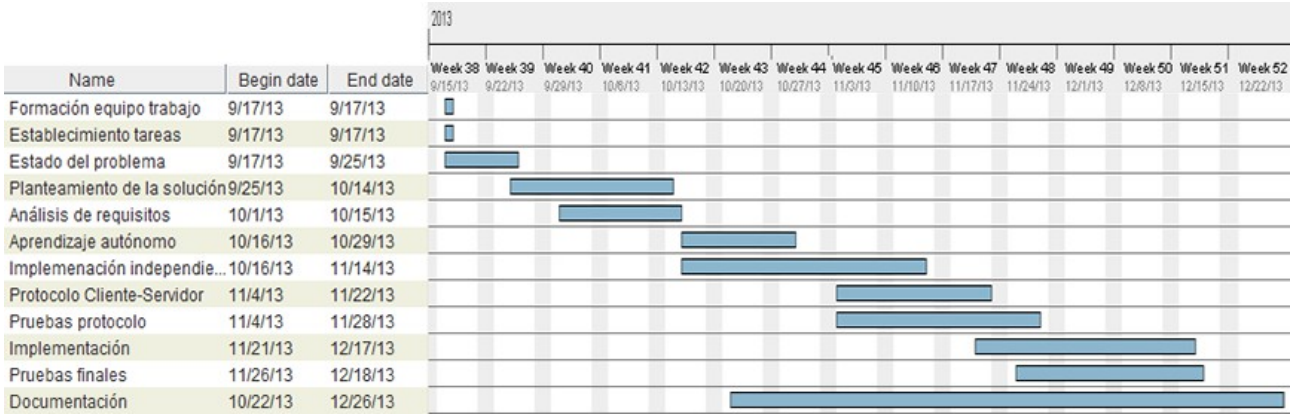


Name	Begin date	End date
Formación equipo trabajo	9/17/13	9/17/13
Establecimiento tareas	9/17/13	9/17/13
Estado del problema	9/17/13	9/25/13
Planteamiento de la solución	9/25/13	10/14/13
Análisis de requisitos	10/1/13	10/15/13
Aprendizaje autónomo	10/16/13	10/29/13
Implemenación independie...	10/16/13	11/14/13
Protocolo Cliente-Servidor	11/4/13	11/22/13
Pruebas protocolo	11/4/13	11/28/13
Implementación	11/21/13	12/17/13
Pruebas finales	11/26/13	12/18/13
Documentación	10/22/13	12/26/13

Figura 2.5.1.1: Diagrama de Gantt con la planificación inicial de las tareas del proyecto.

2.5.2.- Diagrama de Gantt real común.

En la figura siguiente (2.5.2.1) podemos ver los resultados de la evolución del proyecto. Las tareas iniciales: formación del equipo, repartición de tareas, análisis del problema, se llevaron a cabo sin inconvenientes; el análisis de requisitos se hizo casi en paralelo al planteamiento de la solución. Toda esta fase nos ocupó, aproximadamente, el primer mes de trabajo. Las fases de implementación, pruebas e integración ocuparon un periodo más largo del previsto inicialmente. Para solventar este problema, todas esas tareas se realizaron de manera casi simultánea, esto solo fue posible gracias a la buena coordinación y comunicación entre los responsables de las dos partes. Finalmente, la documentación se realizó de manera continua durante el desarrollo del proyecto.

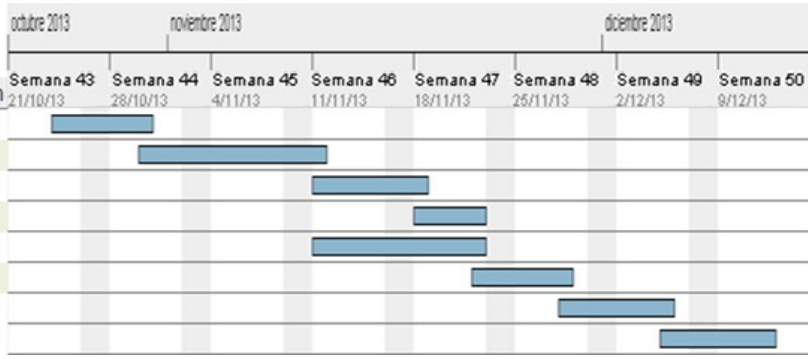


Name	Begin date	End date
Formación equipo trabajo	9/17/13	9/17/13
Establecimiento tareas	9/17/13	9/17/13
Estado del problema	9/17/13	9/25/13
Planteamiento de la solución	9/25/13	10/14/13
Análisis de requisitos	10/1/13	10/15/13
Aprendizaje autónomo	10/16/13	10/29/13
Implemenación independie...	10/16/13	11/14/13
Protocolo Cliente-Servidor	11/4/13	11/22/13
Pruebas protocolo	11/4/13	11/28/13
Implementación	11/21/13	12/17/13
Pruebas finales	11/26/13	12/18/13
Documentación	10/22/13	12/26/13

Figura 2.5.2.1: Diagrama de Gantt con los resultados reales del desarrollo del proyecto.

2.5.3.- Diagrama de Gantt teórico individual.

En los puntos anteriores hemos hablado de la coordinación entre las dos partes y de algunas tareas generales. A continuación, detallaremos la planificación de las tareas especificas de este proyecto.
La implementación se planteó en dos fases: primero una implementación y test de las funciones de la aplicación independientes del servidor. Segundo las funciones dependientes del servidor y la integración de las dos partes. Entre esas dos fases, era necesario definir el protocolo concreto de comunicación, incluyendo: mensajes, respuestas, datos enviados, composición de *JSON*, etc..
En el siguiente gráfico (2.5.3.1) podemos apreciar como se planteó inicialmente el coste temporal de cada tarea:



Nombre	Fecha de in...	Fecha de fin
Diseño grafico interfaz	24/10/13	30/10/13
Logica interfaz	30/10/13	11/11/13
Logica modelo datos	11/11/13	18/11/13
Pruebas y correccion errores	18/11/13	22/11/13
Protocolo Cliente-Servidor	11/11/13	22/11/13
Comunicacion con servidor	22/11/13	28/11/13
Parseo JSON	28/11/13	5/12/13
Pruebas finales	5/12/13	12/12/13

Figura 2.5.3.1: Diagrama con la planificación de las tareas individuales del proyecto.

2.5.4.- Diagrama de Gantt real individual.

Para terminar esta sección, veremos la evolución real de las tareas planteadas en el punto anterior.
La implementación de la lógica de control de la interfaz gráfica ocupo una parte de tiempo mayor de la esperada, esto ocasionó que se superpusiera a la implementación del modelo de datos. El primer periodo de pruebas se alargó algunos días más de lo previsto por errores en el tratamiento de datos y tablas de información locales. La especificación del protocolo de comunicación se hizo según lo planeado y sin contratiempos, gracias a la buena comunicación y coordinación entre las dos partes. La segunda fase de implementación, incluyendo la lógica que controla la comunicación cliente – servidor, el manejo de J*SON*, etc. se realizó, aproximadamente, según lo planeado. Finalmente, la segunda fase de pruebas y correcciones se extendió bastante más de lo planeado, debido a errores en el parseo de ficheros *JSON* y otras imprecisiones en la implementación de lo acordado en la especificación del protocolo.
La siguiente figura (2.5.4.1), muestra la evolución de las tareas mencionadas antes.

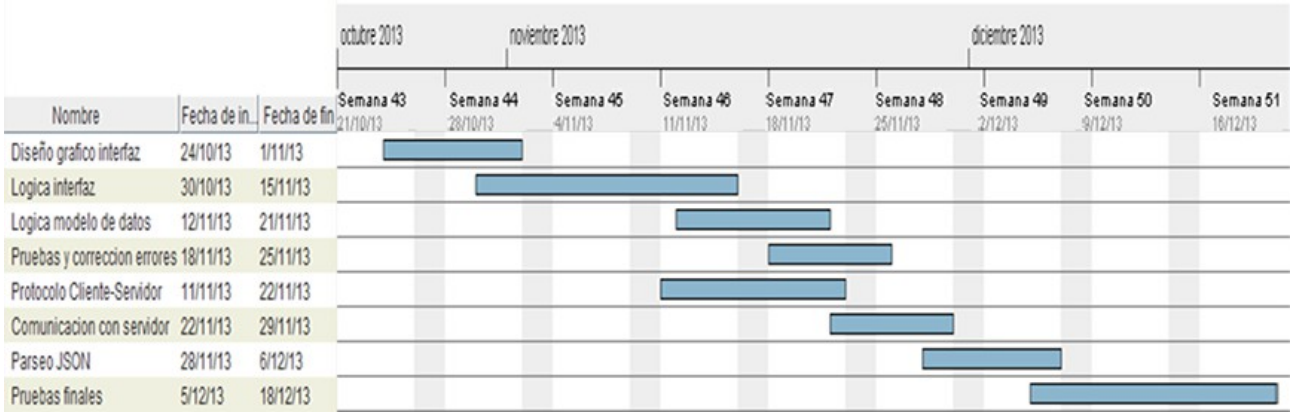


Figura 2.5.4.1: Diagrama con los resultados de la evolución del proyecto.

2.6.- Introducción a las funcionalidades y diseño de la aplicación.

El objetivo fundamental de la aplicación es poder generar una consulta, y que otros usuarios puedan dar su opinión al respecto. Una consulta consta de: titulo, descripción, posibles respuestas y posible imagen.
Con esa premisa, tenemos las siguientes funcionalidades básicas:

1. Crear una consulta, monitorizarla y cerrarla cuando queramos.
2. Volver a consultar las consultas que hemos cerrado.
3. Responder a consultas creadas por otros usuarios.

En las sección tres de esta memoria (implementación) se ilustraran las funcionalidades en detalle.

La interfaz de la aplicación está estructurada en tres componentes básicos (figura 2.5.1):

1. Un menú lateral ('menu fragment') que nos permite seleccionar entre las tres opciones principales. Estas son:
    - La opción 'inbox'. Nos mostrará una lista de consultas pendientes de contestar.
    - 'in progress', nos mostrará una lista de consultas que hemos creado para que otros den su opinión.
    - Y la opción 'historic', que nos muestra un registro de las consultas que hemos hecho en el pasado.

2. El segundo componente es la barra superior ('action bar'). Las acciones que nos permite ejecutar varían dependiendo del contexto de la aplicación. Las principales son:
    - Mostrar u ocultar el menú lateral.
    - Actualizar la información en pantalla con nuevos contenidos del servidor.
    - Enviar una petición o consulta al servidor.

3. El tercer componente es el espacio mayoritario de pantalla ('content fragment'). La información que veremos en esta zona, varia según la opciones que seleccionemos en el menú lateral o la barra superior. Algunas posibilidades son:
    - El formulario para añadir una consulta nueva.
    - Una de las listas mencionadas anteriormente: 'inbox', 'in progress' o 'historic'.
    - Detalles de una consulta en concreto.

La interfaz se adapta al tamaño de pantalla del dispositivo. De manera que si la ejecutamos en un teléfono móvil, el 'menu fragment' quedará oculto a la vista (figura 2.5.2) por el 'content fragment', para mostrarlo se deberá pulsar el botón 'home' del 'action bar' o arrastrar la pantalla de izquierda a derecha (figura 2.5.3). Si ejecutamos en un dispositivo de mayor pantalla, el 'menu fragment' se mostrará permanentemente a la izquierda del 'content fragment' (figura 2.5.1).

Figura 2.5.1: Interfaz en una tableta de 10.1 pulgadas. 'Content fragment' y 'Menu fragment' son visibles al mismo tiempo.

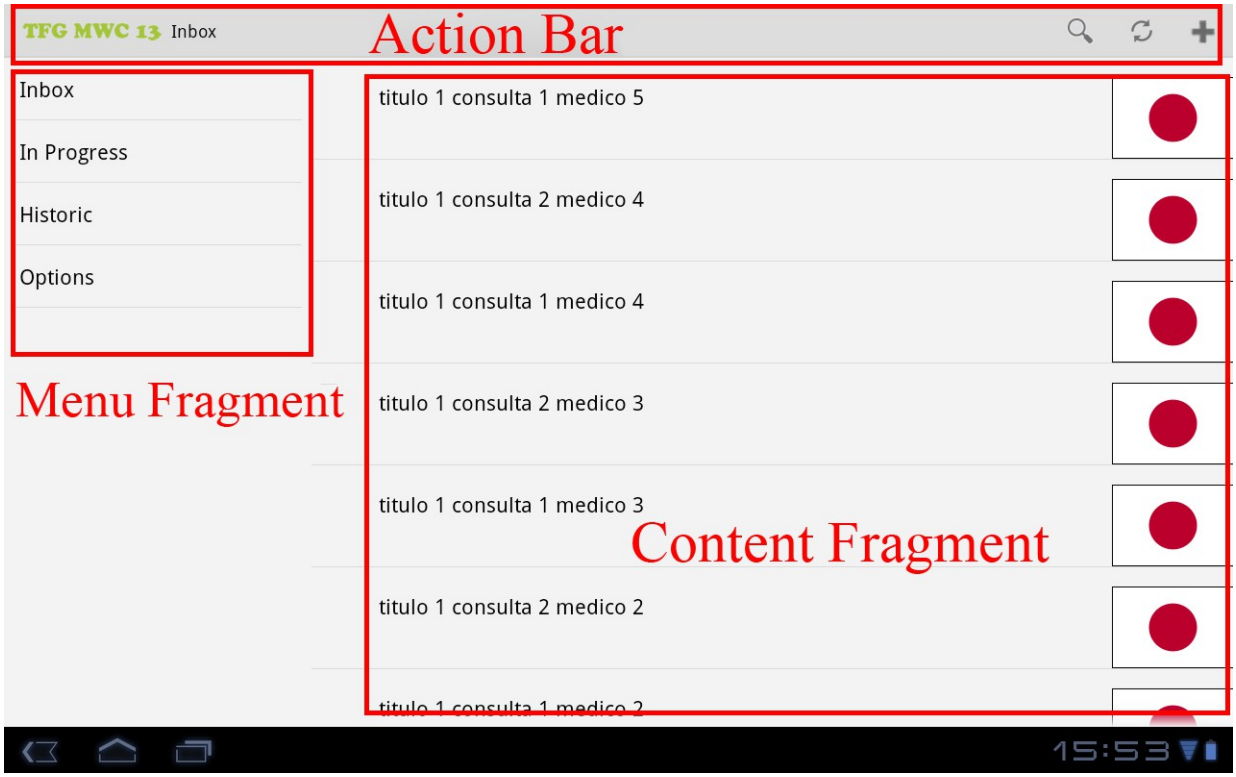


Figura 2.5.2: Interfaz en un teléfono móvil de 4,8 pulgadas. 'Content fragment' es visible mientras que 'Menu fragment' permanece oculto.

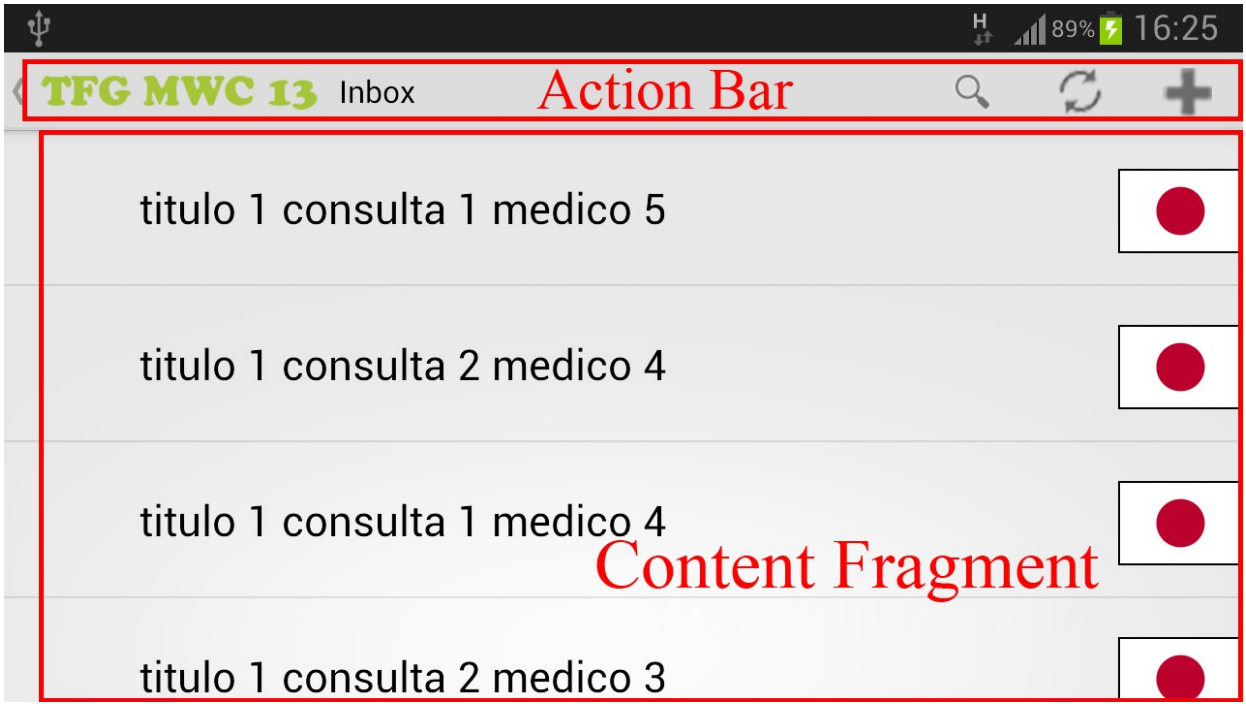
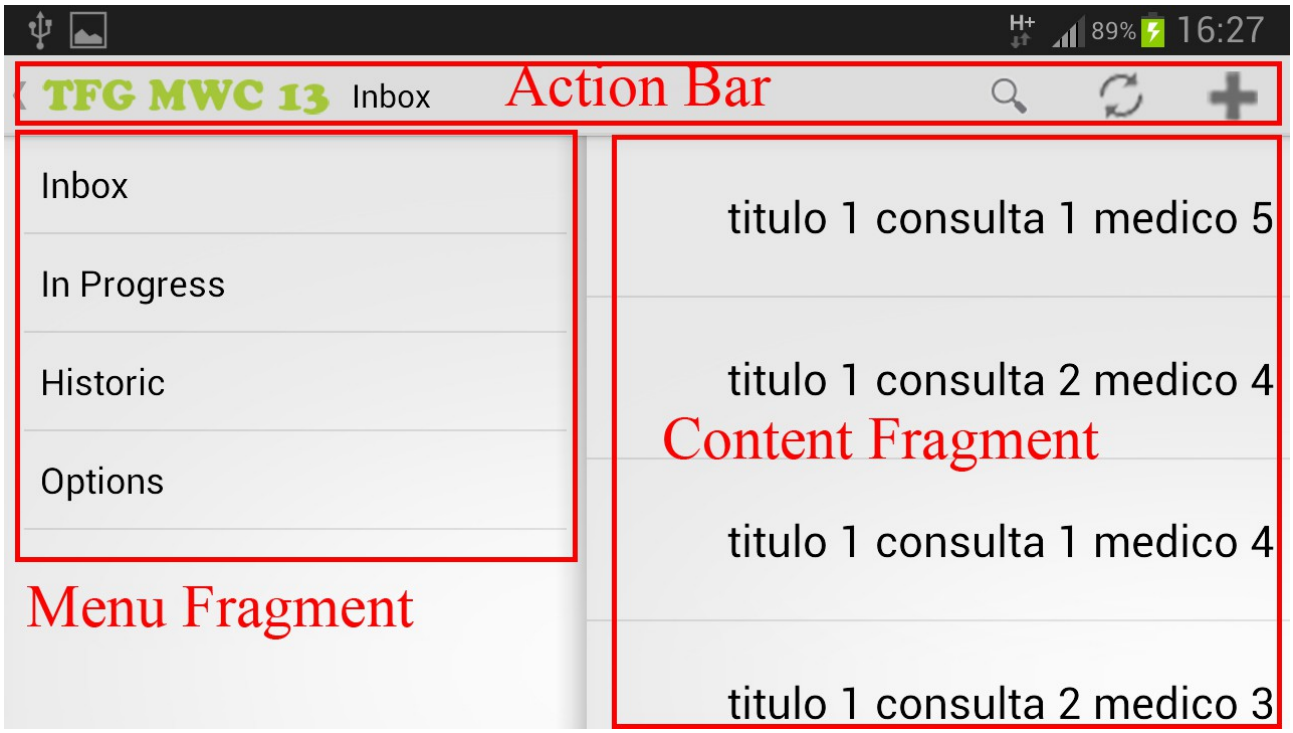


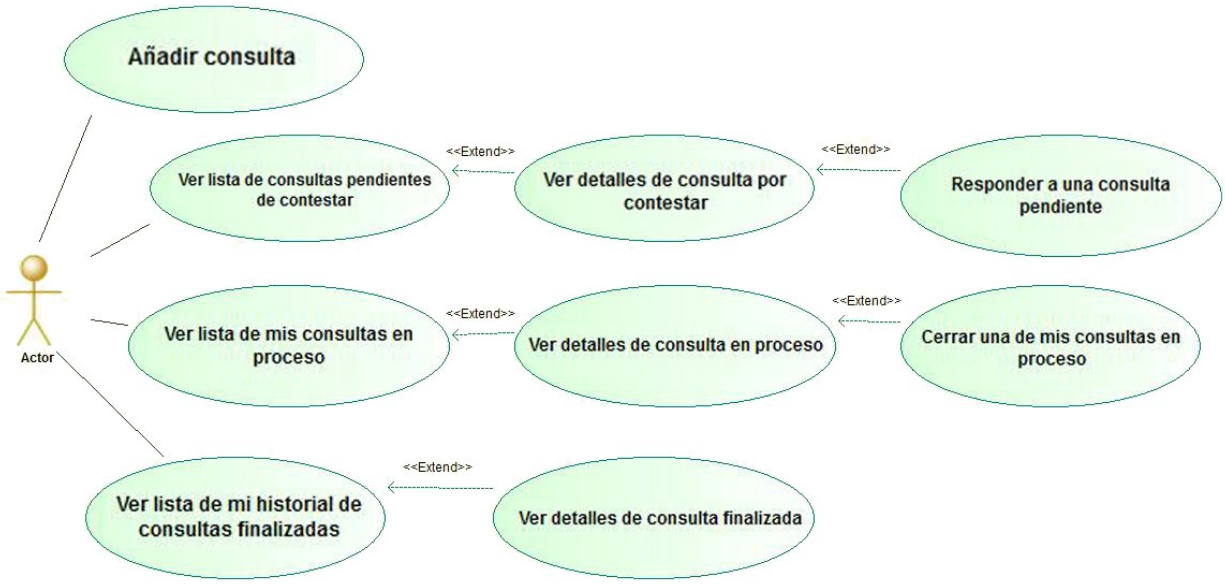
Figura 2.5.3: 'Menu fragment' desplegado desde el lateral izquierdo.



Implementación

3.1.- Diagrama de casos de uso de la aplicación:

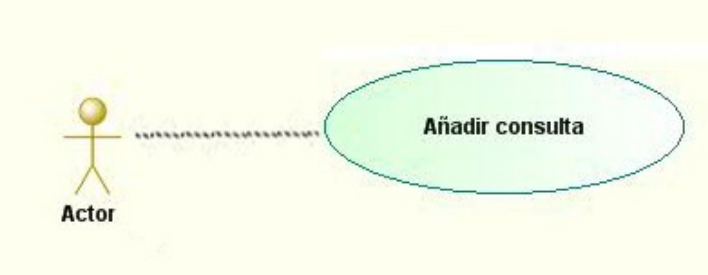
Figura 3.1.1: Diagrama de casos de uso de la aplicación



Tal como hemos mencionado en la sección 2.5, la aplicación tiene tres funcionalidades básicas: añadir consulta, responder a una de otro usuario y consultar el histórico. A continuación analizaremos las condiciones, requisitos y excepciones de cada uno. En la secciones 3.4, 3.5, 3.6 y 3.7 explicaremos la implementación en detalle.

3.2.- Casos de uso textuales:

3.2.1.- Añadir consulta.



Caso de uso: Añadir consulta.

Actor: Usuario.

Descripción: El usuario creará una consulta nueva que se subirá al servidor para que otros usuarios contribuyan a resolverla dando su opinión.

Requisitos:

1. Pulsar el botón 'añadir' en la barra superior.
2. Completar todos los campos obligatorios: titulo de la pregunta, descripción de la misma y dos posibles respuestas.
   De manera opcional, el usuario puede:
   - Añadir tantas respuestas extra como quiera.
   - Adjuntar una imagen que se encuentre en su dispositivo y contribuya a resolver la pregunta.
3. Pulsar el botón de 'enviar' en la barra superior.

Excepciones:

1. El usuario no introduce todos los campos obligatorios: la aplicación mostrará un mensaje de error en el que se indica cual es el campo erróneo. No se hacen cambios en el servidor.
2. El dispositivo tiene algún problema de conexión a Internet o el servidor no funciona correctamente: la aplicación nos mostrará un mensaje de error con información relacionada al problema que ha ocurrido. No se hacen cambios en el servidor.

3.2.2.- Ver lista de consultas pendientes de contestar.

Caso de uso: Ver lista pendientes de contestar.

Actor: Usuario.

Descripción: El usuario accede a la lista de consultas que puede contestar. La lista se genera a partir de contenidos descargados del servidor y se almacena una copia en el disco local. Si la lista no tuviera ninguna entrada (no tenemos ninguna consulta que contestar), se mostraría una imagen indicando que no hay contenidos para mostrar.

Requisitos:

1. Pulsar el botón que muestra el menú lateral o arrastrar la pantalla de izquierda a derecha. para mostrarlo.
2. Seleccionar la primera opción de la lista del menú 'inbox'.

Excepciones:

1. Si hay algún problema en la conexión con el servidor o en la transferencia de datos, se cargará la lista local. En cualquier caso, si la lista que vamos a mostrar está vacía, se mostrará una imagen para indicar que no hay contenidos disponibles.

3.2.3.- Ver detalles de consulta por contestar.

Caso de uso: Ver detalles de consulta por contestar.

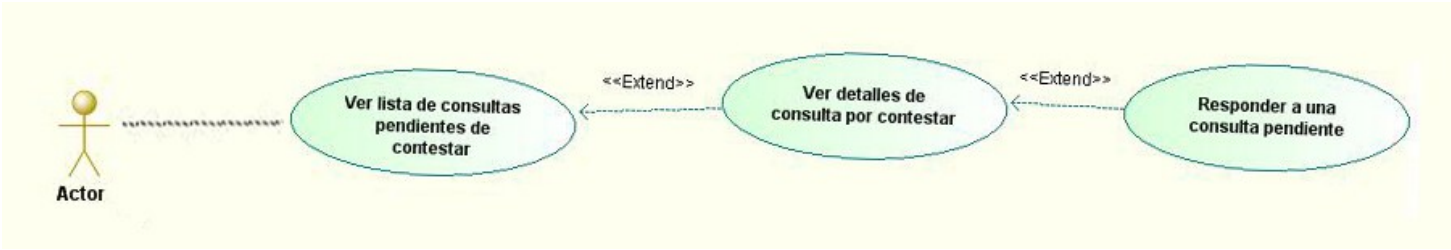
Actor: Usuario.

Descripción: El usuario examina una consulta en detalle. Se mostrará el título de la pregunta, la descripción de la misma, las posibles respuestas y la imagen si la hubiera.

Requisitos:

1. El usuario está viendo la lista de consultas que puede contestar (caso de uso 2).
2. Seleccionar una consulta de la lista.

3.2.4.- Responder a una consulta pendiente.



Caso de uso: Responder a una consulta pendiente.

Actor: Usuario.

Descripción: El usuario responde a una consulta. La respuesta será enviada al servidor y la consulta se borrará de la lista de pendientes del usuario (ya la ha contestado por tanto no la necesita seguir viendo).

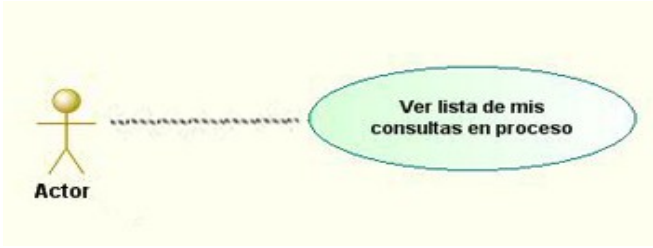
Requisitos:

1. El usuario está viendo los detalles de una consulta que puede contestar (caso de uso 3).
2. Seleccionar una posible respuesta.
3. Pulsar el botón 'contestar' en la barra superior.

Excepciones:

1. Si hay algún problema en la conexión con el servidor o en la transferencia de datos, se mostrará un mensaje de error con información del fallo. No se hacen cambios en el servidor ni locales.

3.2.5.- Ver lista de mis consultas en proceso.



Caso de uso: Ver lista de mis consultas en proceso.

Actor: Usuario.

Descripción: El usuario accede a la lista de consultas que ha creado y publicado (caso de uso 1). La lista se genera a partir de contenidos descargados del servidor y se almacena una copia en el disco local. Si la lista no tuviera ninguna entrada (no tenemos ninguna consulta que contestar), se mostraría una imagen indicando que no hay contenidos para mostrar.

Requisitos:

1. Pulsar el botón que muestra el menú lateral o arrastrar la pantalla de izquierda a derecha. para mostrarlo.
2. Seleccionar la primera opción de la lista del menú 'in progress'.

Excepciones:

1. Si hay algún problema en la conexión con el servidor o en la transferencia de datos, se cargará la lista local. En cualquier caso, si la lista que vamos a mostrar está vacía, se mostrará una imagen para indicar que no hay contenidos disponibles.

3.2.6.- Ver detalles de una de mis consultas en proceso.

Caso de uso: Ver detalles de consulta por contestar.

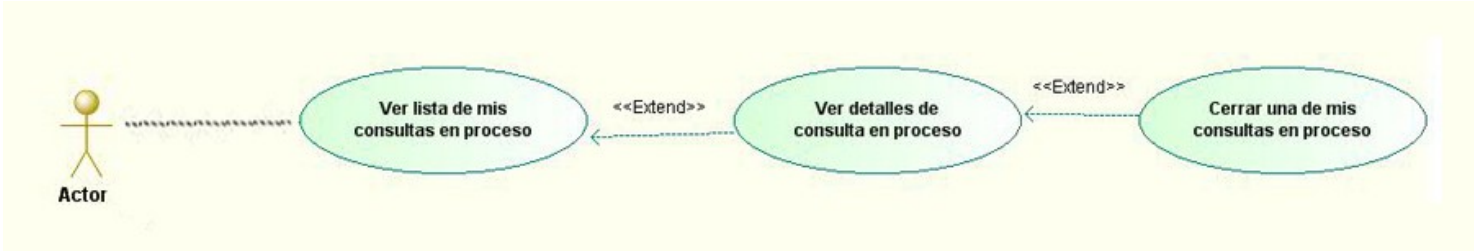
Actor: Usuario.

Descripción: El usuario monitoriza una consulta que ha creado previamente. Se mostrará el título de la pregunta, la descripción de la misma, las posibles respuestas (cada una con un contador que indica cuantas veces ha sido seleccionada por otros usuarios) y la imagen si la hubiera.

Requisitos:

1. El usuario está viendo la lista de sus consultas en proceso (caso de uso 5).
2. Seleccionar una consulta de la lista.

3.2.7.- Cerra una consulta en proceso.



Caso de uso: Cerra una de mis consultas en proceso.

Actor: Usuario.

Descripción: El usuario puede cerrar una consulta en cualquier momento. Una vez cerrada, la consulta será eliminada de la lista 'in progress' y pasará a la lista 'historic'.

Requisitos:

1. El usuario está viendo los detalles de una de sus consultas (caso de uso 6).
2. Pulsar el botón 'cerrar consulta' en la barra superior.

Excepciones:

1. Si hay algún problema en la conexión con el servidor o en la transferencia de datos, se mostrará un mensaje de error con información del error. No se harán cambios locales ni el en el servidor.

3.2.8.- Ver lista de consultas cerradas.

Caso de uso: Ver histórico (lista de mis consultas cerradas).

Actor: Usuario.

Descripción: Cuando el usuario cierra una consulta que había abierto, esta pasa al histórico. Los detalles de la pregunta siguen estando disponibles para volver a consultarlos. La lista que se muestra se genera a partir de datos obtenidos del servidor, también se mantiene una copia local.

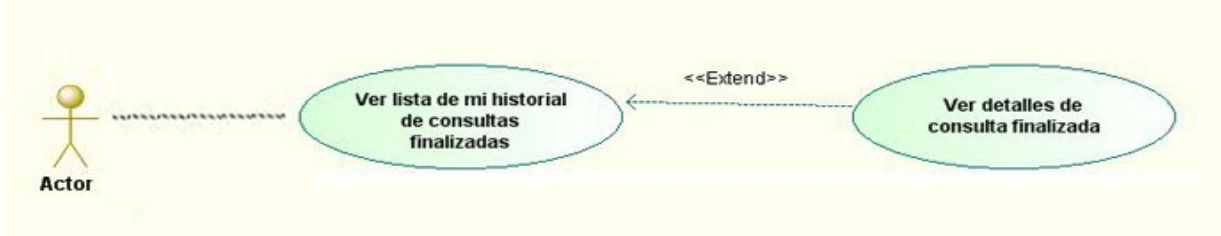
Requisitos:

1. Pulsar el botón que muestra el menú lateral o arrastrar la pantalla de izquierda a derecha. para mostrarlo.
2. Seleccionar la primera opción de la lista del menú 'in progress'.

Excepciones:

1. Si hay algún problema en la conexión con el servidor o en la transferencia de datos, se cargará la lista local. En cualquier caso, si la lista que vamos a mostrar está vacía, se mostrará una imagen para indicar que no hay contenidos disponibles.

3.2.9.- Ver detalles de una consulta del histórico.



Caso de uso: Ver detalles de consulta por contestar.

Actor: Usuario.

Descripción: El usuario vuelve a consultar los detalles de una consulta que ha cerrado previamente. Se mostrará el título de la pregunta, la descripción de la misma, las posibles respuestas (cada una con un contador que indica cuantas veces ha sido seleccionada por otros usuarios) y la imagen si la hubiera.

Requisitos:

1. El usuario está viendo su histórico de consultas (caso de uso 8).
2. Seleccionar una consulta de la lista.

3.3.- Estructura de clases simplificada y ficheros *.xml* relacionados.

En esta sección describiremos, a grandes rasgos, el contenido de las clases mas importantes desarrolladas en el proyecto y los ficheros *.xml* con los que se relacionan. La figura 3.3.1 muestra estas relaciones de manera esquemática.

Main Activity.java es la actividad principal, se encarga de que la interfaz se adapte a diferentes tamaños de pantalla, contiene la lógica para el manejo de los fragmentos intercambiables del centro de la pantalla y también para el menú lateral. Contiene los siguientes ficheros *.xml*:

1. *main_menu.xml* describe el aspecto visual de la barra superior. Esta barra siempre se mantiene en la pantalla e incluye un botón de refresco, un botón de búsqueda, uno de nueva entrada y el botón que muestra el menú lateral (también puede mostrarse arrastrando desde el borde izquierdo).
2. *responsive_content_frame.xml*, este archivo tiene dos versiones. Si ejecutamos en un dispositivo de gran pantalla, contendrá dos marcos, uno para el contenido en la parte central de la pantalla y otro para el menú lateral. Si ejecutamos en un teléfono móvil, el *.xml* contendrá unicamente el marco para el contenido, el menú lateral se situará en otro marco totalmente dependiente (en el fichero *side_menu_frame.xml*), y permanecerá oculto hasta que el usuario lo revele.

MenuFragment.java es la clase que describe el comportamiento del fragmento que usamos como menú lateral (por tanto, este fragmento es responsabilidad de *MainActivity*) y su aspecto se encuentra en *side_menu.xml* (describe como será la lista en la que se encontrarán las entradas del menú). Al pulsar en uno de los elementos del menú, esta clase indicará a *MainActivity* que haga los cambios de fragmentos pertinentes.

Add.java es la actividad que nos permite crear nuevas consultas. Se ejecuta al pulsar uno de los botones de la barra de acciones de *MainActivity*. Al iniciar se nos mostrará un formulario donde podemos rellenar la información de la nueva consulta (título, descripción, respuestas, imagen) y un botón de envío en la barra superior. Esta actividad tiene dos *.xml* relacionados:

1. *main_menu_add.xml* que contiene los elementos de la barra superior asociados a esta actividad (botón home y enviar).
2. *add_activity.xml* que contiene los widgets que componen el formulario.

InBoxFragment.java es un fragmento (dependiente de MainActivity) que contiene una lista donde se mostrará la información (obtenida del servidor) relativa a las consultas en las que el usuario puede participar. La clase *Server.java* se encarga de la transferencia de la información y la lista que veremos en pantalla se construye a partir de la clase *BaseListViewAdapter.java*, hablaremos de estas clases más abajo en esta sección. El aspecto de este fragmento está descripto en *inbox_fragment.xml*. Al pulsar en una de las entradas de la lista, se lanzará la actividad *InBoxSingleItemActivity*.

InBoxSingleItemActivity.java es una actividad que nos permite contestar a una consulta creada por otro usuario. Se nos mostrarán los detalles de la consulta: título, descripción, respuestas e imagen (si la hubiere). Las respuestas se nos presentarán en forma de botones excluyentes, de manera que solo podremos seleccionar uno, también podemos pulsar en la imagen para verla en pantalla completa (*ImageViewActivity*). Contiene los siguientes *.xml*:

1. *main_menu_add.xml* que describe los elementos de la barra superior. En este caso, contiene el botón 'home' y un botón para enviar la opción seleccionada.
2. *base_acitivity.xml*, este fichero contiene los *widgets* que conforman el formulario de consulta.

InProgressFragment.java es la clase que contiene el fragmento en el que veremos una lista de las consultas que nosotros hemos creado. El comportamiento de la lista se encuentra en la clase *BaseListViewAdapter* y los contenidos que se mostrarán, se obtienen del servidor con ayuda de la clase *Server.java*. Si pulsamos en un elemento de esta lista se lanzará la actividad *InProgressSingleItemActivity*, donde veremos los detalles de nuestra consulta. El aspecto de este fragmento se describe en *inprogress_fragment.xml.*

InProgressSingleItemActivity.java es la actividad que nos permite monitorizar una de nuestras consultas. Se nos mostrará toda la información de la consulta, un botón que nos permite cerrarla y también podremos pulsar en la imagen para verla en pantalla completa (*ImageViewActivity*). Contiene dos *.xml*:

1. *main_menu_close.xml* describe los elementos de la barra superior (el botón 'home' y otro para cerrar la consulta).
2. *base_acitivity.xml*, este fichero contiene los *widgets* que conforman el formulario de consulta.

HistoricFragment.java clase que contiene el fragmento del registro histórico. Esta lista se genera a partir de contenidos descargados de *Server.java* y la lista se configura con la ayuda de *BaseListViewAdapter*. Si pulsamos en un elemento de la lista, se lanzará una nueva actividad (*HistoricSingleItemAcitivity*) en la que veremos esa consulta en detalle. El aspecto de este fragmento se describe en historic*_fragment.xml.*

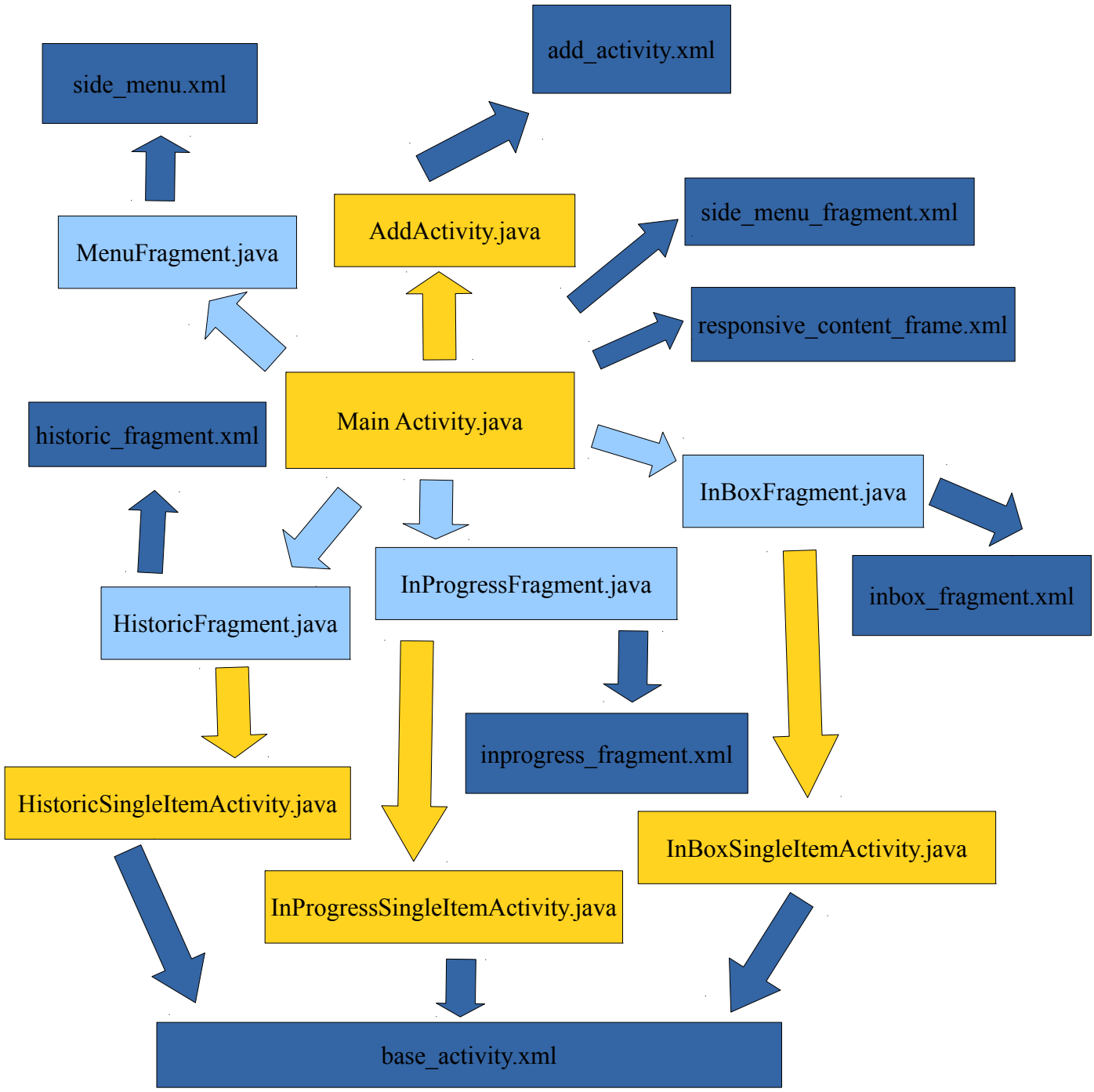
HistoricSingleItemActivity.java es la actividad en la que veremos los detalles de una consulta en la sección de histórico. Los detalles incluyen: título, descripción, respuestas e imagen. La consulta que estamos viendo ya ha sido cerrada, por tanto, esta actividad no nos permitirá ninguna opción extra, solo podremos pulsar sobre la imagen para verla en pantalla completa. La interfaz de la actividad está descripta en el fichero *base_activity.xml.*

Server.java es una clase auxiliar con funciones que nos permiten enviar y recibir datos al servidor, hacer el control de errores, establecer *time outs* para las conexiones, generar mensajes de error, etc. Son funciones estáticas puesto que no dependen de unas instancias en concreto, están diseñadas de manera genérica.

BaseListViewAdapter.java es una clase que contiene un adaptador de lista genérico, que usamos para construir las listas mencionadas antes ('inbox', 'inprogress', 'historic'). Cada elemento de la lista contiene: título y una visualización de la imagen. El aspecto del conjunto de la lista está descripto en el fichero *listview_item.xml.*

ImageViewActivity.java es el fichero que contiene la actividad auxiliar que gestiona la vista de la imagen en pantalla completa. Se nos mostrará la imagen en pantalla completa y la barra superior se ocultará a la vista automáticamente tras tres segundos. Para restaurarla tendremos que pulsar en cualquier lugar de la pantalla, aunque volverá a ocultarse tras tres segundos.

Figura 3.3.1: clases java y ficheros .xml relacionados.
Nota: los cuadros de color amarillo representan 'activities', los de color azul claro representan 'fragments' y los de color azul oscuro 'GUI'.



3.4.- Añadir consulta.

Esta es la funcionalidad más básica de la aplicación. Los pasos necesarios para añadir una consulta están explícitamente documentados en el capitulo cuatro, a partir de esta captura.
Mientras estemos en la actividad principal *MainActivity.java*, tendremos siempre a nuestra disposición unos botones en la barra de acciones de la parte superior. Estos botones nos permiten varias funcionalidades y todos ellos gestionados por un listener, *onOptionsItemSelected.* En este caso, la función manejará el evento creando una nueva actividad, *AddActivity.java,* que nos mostrará en pantalla el formulario con la información que necesitamos completar para crear la consulta. Si pulsamos el botón que nos permite añadir la imagen a la consulta, se creará una nueva actividad que nos permitirá elegir una imagen de la galería, cuando esta actividad auxiliar termine se ejecutará la función *onActivityResult,* que se encargará de mostrar una previsualización de la imagen en pantalla. Si pulsamos el botón de añadir respuestas extra se ejecutará la función *getNewAnswer,* esta función mostrará en pantalla una ventana de dialogo con un recuadro de texto y los botones 'aceptar' y 'cancelar'. La actividad en la que ahora nos encontramos (*AddActivity.java*) tiene su propia barra de acciones en la parte superior, por lo tanto también tiene su propia función listener (*onOptionsItemSelected*) para los botones que allí se encuentran. Al pulsar el botón de envío de la nueva consulta, esta función se encargará de crear una tarea asíncrona, llamada *UploadNewEntry.* La tarea asíncrona comprobará que todos los campos del formulario son correctos, creará un *string* con todos los campos de la consulta y llamará a la función estática *SendToServer* de la clase *Server*.
La función *SendToServer* recibe dos *strings* como parámetros, el primero indica la URL del servidor y el segundo es la lista de parámetros que pasaremos mediante el GET. La lista de parámetros ha sido creada antes en la tarea asíncrona *UploadNewEntry*. La función intentará conectar con el servidor y hará el GET. Si todo ha funcionado correctamente, retornará un *string* 'si' y se nos mostrará en pantalla un mensaje de confirmación. En este punto la actividad *AddActivity.java* no tiene nada más que hacer y terminará, devolviendo el control a *MainAcitivity.java.*
AddActivity.java es una actividad con resultado, por tanto, cuando esta acaba causa que se ejecute la función *onActivityResult* de la actividad que la ha llamado. Esta funcion se encargará de mostrarnos la pantalla de la aplicación donde podemos monitorizar nuestras consultas y la que acabamos de crear, se mostrará en primer lugar.

Todo este proceso puede fallar en tres módulos diferentes:

1. Errores en la aplicación local. Por ejemplo, al no completar todos los campos del formulario.
2. Errores en la comunicación con el servidor. Por ejemplo: no tenemos Internet, el servidor está caído, el servidor no contesta antes de un determinado *time out.*
3. Errores de ejecución en el servidor. Si esto ocurriera, el servidor nos contestaría con un mensaje de error en lugar de con el 'sí' mencionado antes.

El siguiente esquema representa, de manera simplificada, el flujo del proceso:

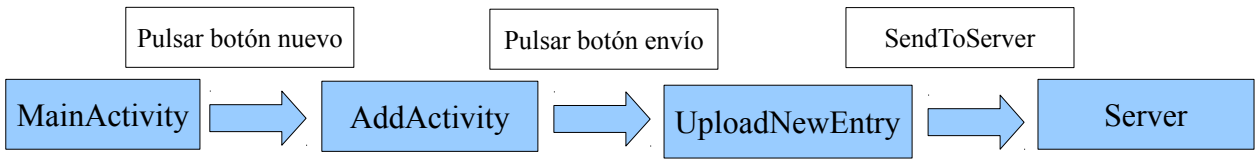


Figura 3.4.1: Esquema de creación de nueva consulta.

3.5.- Responder a una consulta.

Para responder a una consulta creada por otro usuario, necesitamos que la lista visible sea 'inbox' y pulsar en un elemento de la lista (el que queramos contestar). Todo el proceso de respuesta se encuentra ilustrado en la sección cuatro (partiendo de esta captura).
Si la lista 'inbox' es visible, quiere decir que el fragmento que la contiene es el activo (en este caso el fragmento *InBoxFragment.java,* que hereda de *BaseFragment*). La primera vez que el fragmento se muestra, se actualizará con contenidos descargados desde el servidor, si queremos podemos actualizarlo manualmente pulsando el botón 'actualizar' en la barra superior.
Cuando pulsemos en un elemento de la lista, el fragmento lanzará una nueva actividad, *InBoxSingleItemActivity.java.* Esta actividad, que hereda de *BaseSingleItemActivity.java*, nos mostrará en pantalla todos los detalles de la consulta: titulo, descripción, posibles respuestas y la imagen. La función *onCreate* se encargará de que las respuestas se muestren como radio botones, de manera que solo podremos mantener seleccionado uno de ellos. Si pulsamos en la imagen, la función *imageClick* en la clase *BaseSingleItemActivity.java* se encargará de mostrarnos la imagen en pantalla completa.
Una vez que hayamos marcado nuestra respuesta preferida, pulsamos el botón de 'responder' en la barra superior y el listener (*onOptionItemSelected*) se encarga de crear los dos *strings* necesarios para enviar el mensaje al servidor (URL del servidor y parámetros para el GET) y de crear una tarea asíncrona que gestiona el proceso de envío. Esta tarea asíncrona, *SendToServerAsync,* ejecuta la función estática *SendToServer* de la función *Server.java* y muestra un mensaje de confirmación o de error dependiendo del resultado de la conexión, el envío y la respuesta. Si todo ha funcionado, la actividad indica que ha terminado correctamente en la función *onProcessFinished* y acaba su ejecución. La actividad que la ha llamado (*MainActivity.java*, mediante el fragmento 'inbox') ejecuta entonces la función *onActivityResult* (en *InBoxFragment.java*). Esta función se encarga de: eliminar la consulta contestada de la lista 'inbox', puesto que, si ya la hemos contestado no necesitamos seguirla viendo y de actualizar automáticamente el fragmento (para consultar si hay nuevas preguntas que responder).

Como en la funcionalidad tratada en el punto anterior, tenemos un tratamiento de errores en tres posibles niveles:

1. Errores en la aplicación local. Por ejemplo, al no completar todos los campos del formulario.
2. Errores en la comunicación con el servidor. Por ejemplo: no tenemos Internet, el servidor está caído, el servidor no contesta antes de un determinado *time out.*
3. Errores de ejecución en el servidor. Si esto ocurriera, el servidor nos contestaría con un mensaje de error en lugar de con el 'sí' mencionado antes.

El siguiente esquema representa el orden de llamadas del proceso:

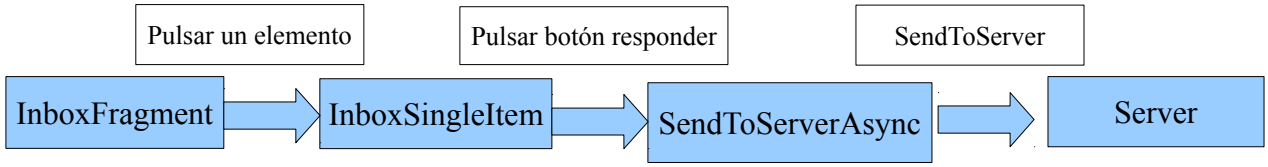


Figura 3.5.1: Esquema del flujo de llamadas en el proceso de respuesta a una consulta.

3.6.- Monitorizar y cerrar una consulta propia.

Nuestras consultas se muestran en forma de lista, en el fragmento 'in progress'. Para ver una de nuestras consultas en detalle, debemos pulsar en un elemento de la lista. El proceso completo se muestra gráficamente en el capítulo cuatro (en esta captura). El fragmento encargado de esta lista es el *InProgressFragment.java,* al pulsar en uno de los elementos, se lanza una nueva actividad del tipo *InProgressSingleItemActivity.java.* Esta actividad muestra en pantalla los detalles de la consulta, junto a un botón 'cerrar consulta'. Adjunto a cada una de las respuestas, se muestra un contador de la cantidad de veces que esa respuesta ha sido elegida (todo esto es responsabilidad de la función *onCreate*). Cuando el usuario decide que puede cerrar la consulta y pulsa el botón de la barra superior, la función *onOptionsItemSelected* se encarga de generar dos *strings:* uno con la URL del servidor y el segundo con los parámetros que se usarán más tarde en el GET. Una vez creados los *strings*, se crea un tarea asíncrona que se encarga de gestionar el proceso de envío al servidor, el proceso se resume en tres pasos consecutivos:

1. Mostrar un mensaje de aviso de que la tarea ha empezado.
2. Ejecutar la función estática *SendToServer* de la clase *Server*. Esta función se encarga de conectar con es servidor, hacer el GET y retornar un código de error.
   Si todo ha funcionado correctamente, el servidor habrá pasado la consulta que cerramos a su lista 'historic', ahora pertenece a esa lista y no a 'in progress'.
3. Mostrar un mensaje de final correcto o de error.

En este caso, a diferencia del punto 3.5, los errores pueden producirse en dos niveles:

1. Errores en la comunicación con el servidor. Por ejemplo: no tenemos Internet, el servidor está caído, el servidor no contesta antes de un determinado *time out.*
2. Errores de ejecución en el servidor. Si esto ocurriera, el servidor nos contestaría con un mensaje de error en caso contrario nos enviaría un 'sí'.

Una vez ejecutados los tres pasos mencionados antes, esta actividad (*InProgressSingleItemActivity*) puede terminar. Cuando la actividad termina, se ejecuta la función *onActivityResult.* Esta función es muy importante puesto que se encarga de mantener la coherencia entre las tablas locales y las que mantiene el servidor. Concretamente, lo que hace la función es:

1. Eliminar de la lista 'in progress' la consulta que acabamos de cerrar.
2. Forzar a que la próxima vez que abramos la lista del registro histórico ('historic fragment') se produzca una actualización desde el servidor. Esto permitirá que la consulta cerrada se descargue como nueva entrada en esa lista.

El siguiente esquema representa el orden de llamadas del proceso:

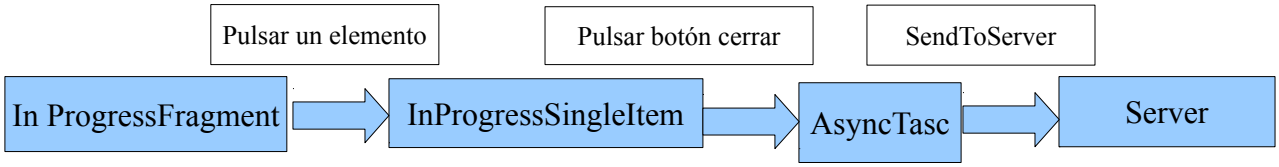


Figura 3.6.1: Esquema del flujo de llamadas en el proceso de cerrar una consulta.

3.7.- Consultar el registro histórico.

Consultar el registro histórico consiste en volver a revisar los detalles de una consulta, una vez que ha sido cerrada. Los pasos para hacerlo se muestran en el capitulo cuatro, a partir de esta captura. El proceso es similar al descripto en las secciones anteriores. Cuando el fragmento 'historic' está activo, nos muestra la lista de nuestras consultas que ya hemos cerrado. Entonces, al pulsar en un elemento, se lanza la actividad *HistoricSingleItemActivity.java.* Esta actividad se encargará de mostrar en pantalla los detalles de esa consulta tal como estaba cuando fue cerrada. Es decir que veremos todos los campos: titulo, descripción, imagen y respuestas (con un contador de las veces que fue seleccionada por otros usuarios), todo esto es competencia de la función *onCreate*.
En este caso, la actividad no nos proporciona la posibilidad de hacer más acciones, puesto que la consulta ya ha sido cerrada. Es por esto que la actividad no proporciona ningún resultado, ni tiene conexiones al servidor. Una vez que pulsemos el botón de menú, volveremos a la lista del fragmento 'historic'.

La siguiente figura muestra, esquemáticamente, el resumen del proceso:

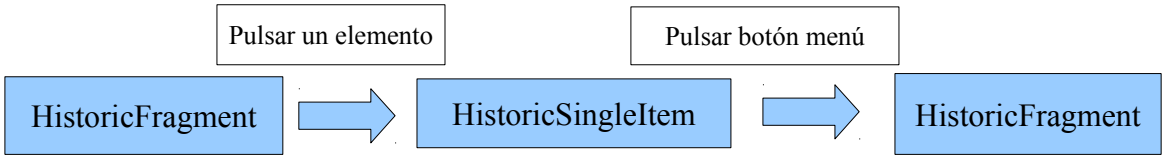


Figura 3.7.1: Esquema de llamadas para ver una consulta del registro histórico.

Manual de usuario

A continuación mostraremos un ejemplo de ejecución de la aplicación, en el que se ilustrarán todas las funcionalidades descriptas en las secciones anteriores. Para esto, usaremos capturas de pantalla de las interfaces tanto para teléfonos móviles, como para tabletas.

4.1.- Manual de usuario (interfaz teléfonos móviles).

Las imágenes mostradas a continuación pertenecen a la aplicación del usuario 1, excepto donde se indique lo contrario.

1.- Al ejecutar la aplicación, lo primero que veremos será un cuadro de dialogo que nos informará de que hay una actualización automática en proceso (figura 4.1.1). Al finalizar esta actualización, veremos la lista de consultas que podemos contestar (figura 4.1.2).

Figura 4.1.1: cuadro de dialogo que veremos durante una actualización.

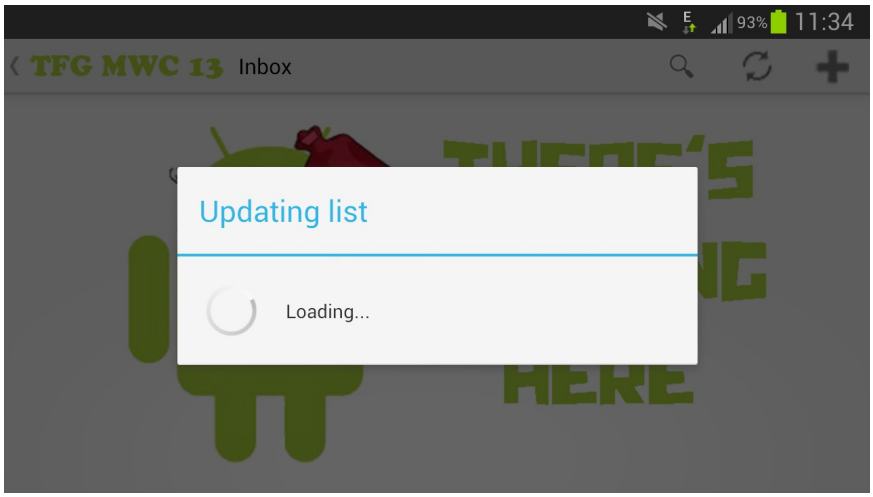
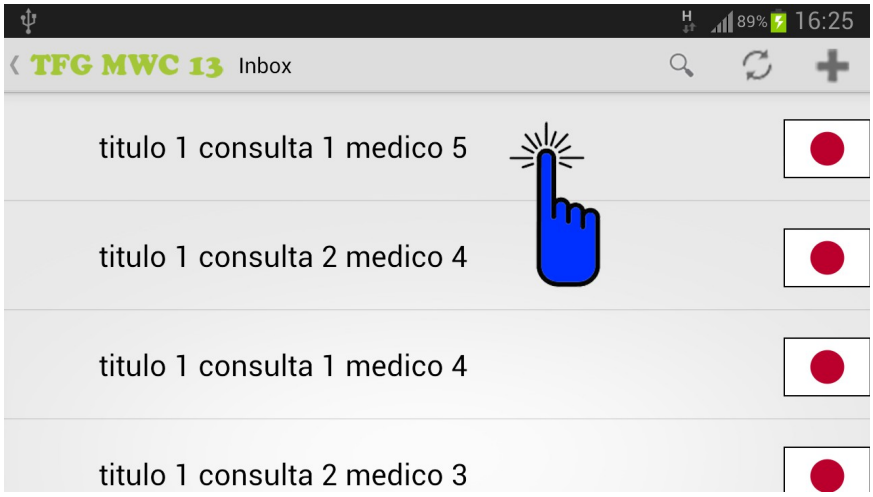


Figura 4.1.2: vista de la lista de consultas que podemos contestar ('inbox').



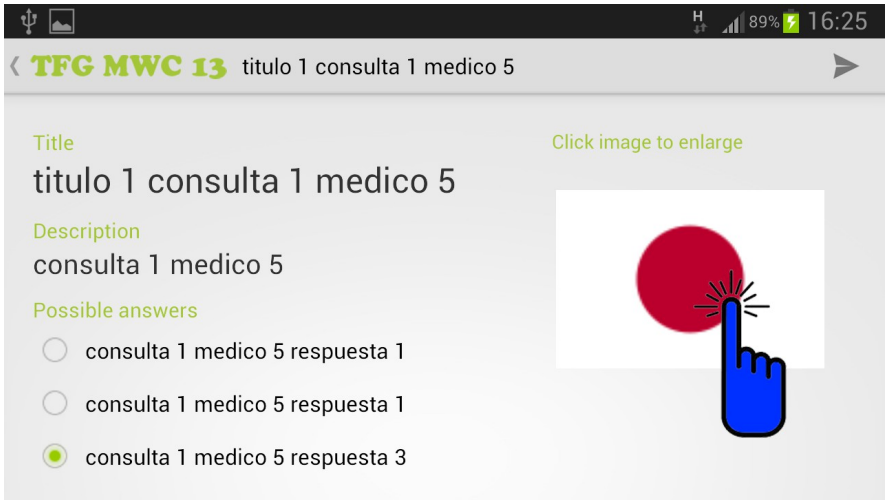


Figura 4.1.3: detalles de una pregunta en nuestro 'inbox'.

2.- Si pulsamos en un elemento de la lista, veremos los detalles de esa pregunta. Podremos seleccionar una de las respuestas y enviarla, ver la imagen en pantalla completa o volver atrás. Pulsamos sobre la imagen

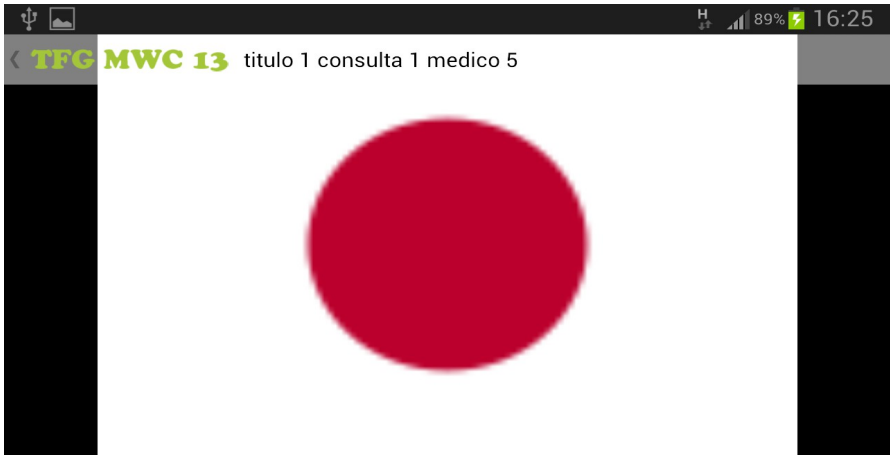


Figura 4.1.4: Vista de la imagen en pantalla completa.

3.- Vemos la imagen en pantalla completa. Después de tres segundos, la barra superior se ocultara automáticamente.

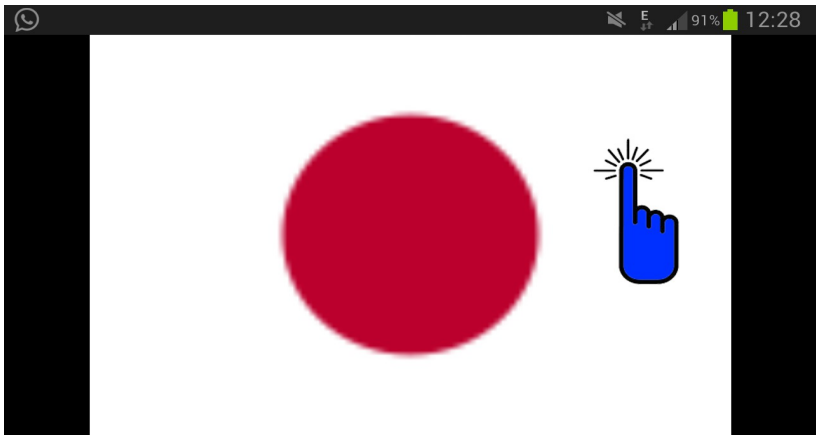


Figura 4.1.5: Vista de la imagen en pantalla completa, la barra superior está oculta.

4.- Para volver a ver la barra superior, tendremos que pulsar en cualquier lugar de la pantalla.

Figura 4.1.6: Pulsar el logo para volver atrás.

5.- Podemos pulsar el logo de la aplicación para volver atrás (volveremos a la vista de los detalles de la pregunta).

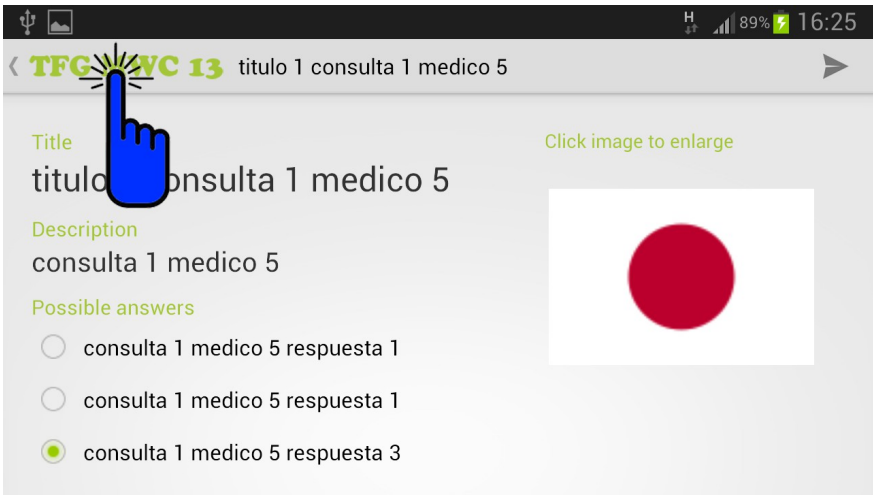


Figura 4.1.7: Detalles de la consulta.

6.- Veremos otra vez los detalles de la consulta. Si pulsamos el logo de la aplicación una vez más, volveremos a la vista de la lista 'inbox'.

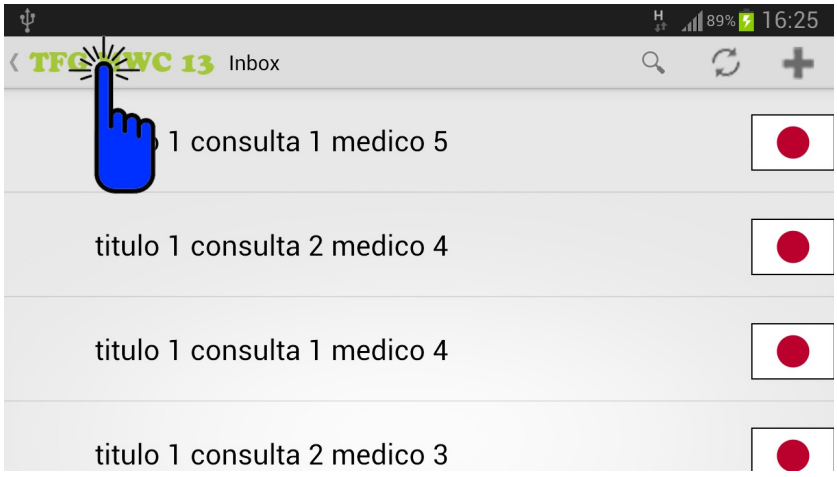
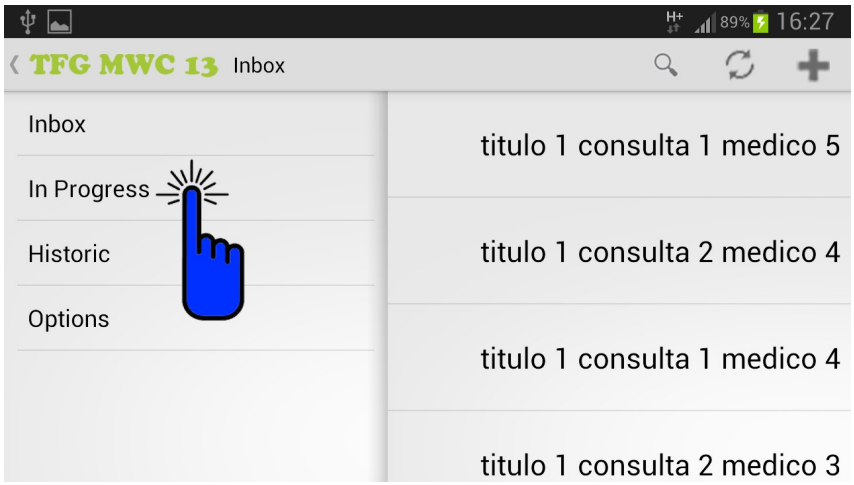


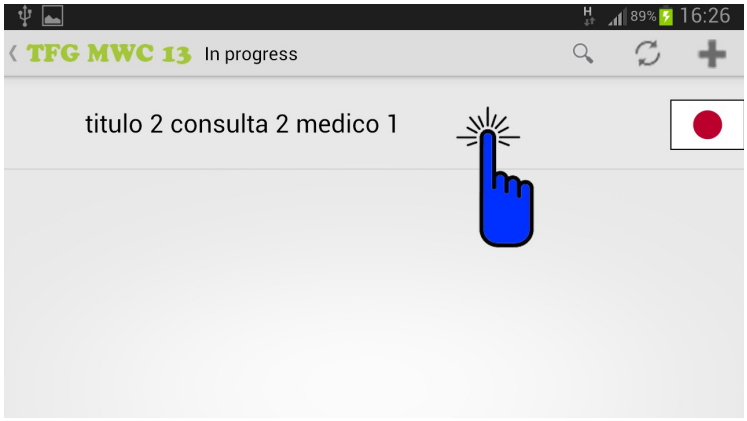
Figura 4.1.8: Vista de la lista 'inbox'.

7.- Vista de la lista 'inbox'. Para revelar el menú lateral, podemos pulsar el logo o arrastrar la ventana de izquierda a derecha.



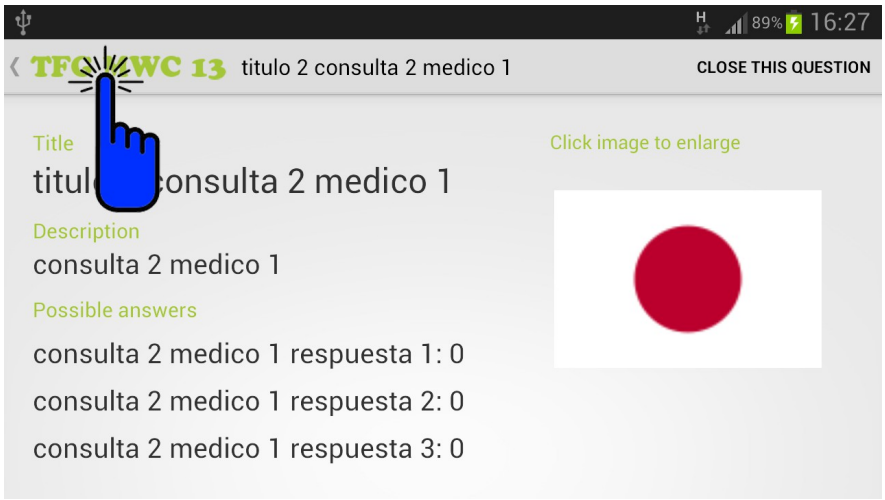
8.- Vemos el menú lateral. Podemos pulsar cualquiera de las opciones y se modificará el fragmento de contenido de la aplicación.

Figura 4.1.9: Vista del menú lateral y posibles interacciones.



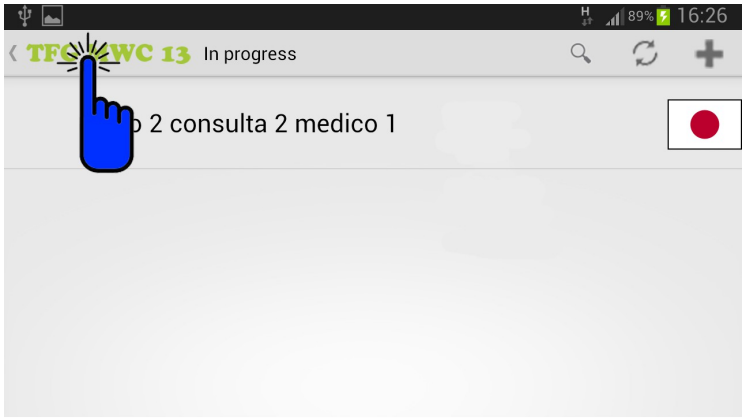
9.- Al pulsar la opción 'in progress', veremos la lista de nuestras consultas. Estas son las consultas que hemos abierto para que otros usuarios nos den su opinión. Si pulsamos en un elemento de la lista, veremos los detalles de la pregunta, incluida información relativa a cuantos usuarios han votado por cada posible respuesta.

Figura 4.1.10: Vista de la lista 'in progress'.



10.- Al monitorizar una de nuestras consultas, se nos muestra toda la información relativa a la pregunta, incluidos contadores do votos para cada posible respuesta, y se nos presenta la opción de cerrarla y darla por terminada. Pulsamos el logo de la aplicación para volver atrás.

Figura 4.1.11: Vista de la pantalla de monitorización de una de nuestras consultas.

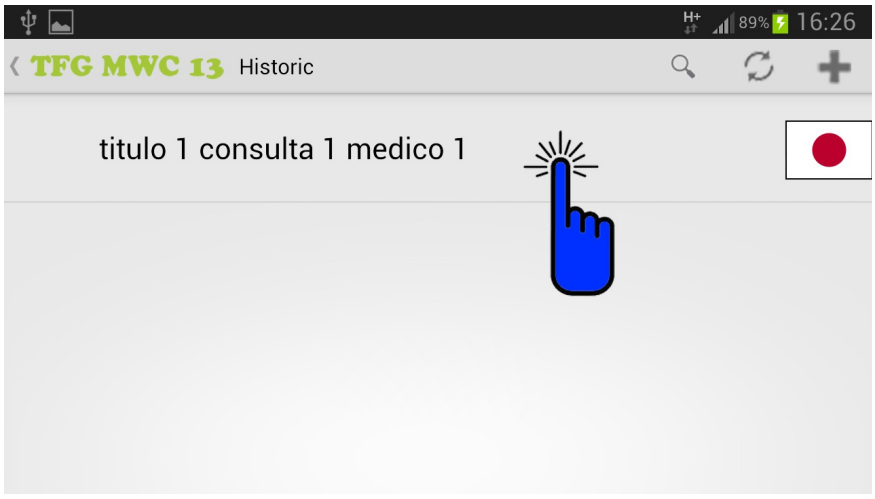


11.- Volvemos a la vista de la lista de nuestras consultas, pulsamos el logo una vez más para mostrar el menú.

Figura 4.1.12: Lista de nuestras consultas en progreso.

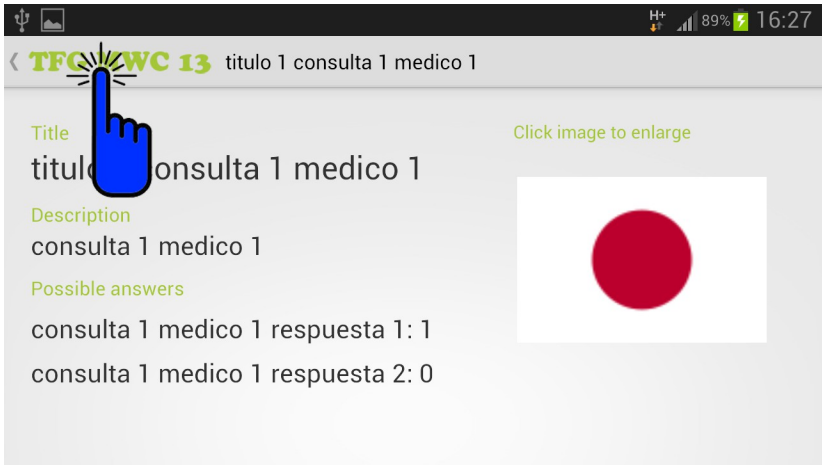
12.- Una vez en el menú elegimos la opción 'historic'.

Figura 4.1.13: Seleccionamos la opción 'historic'.



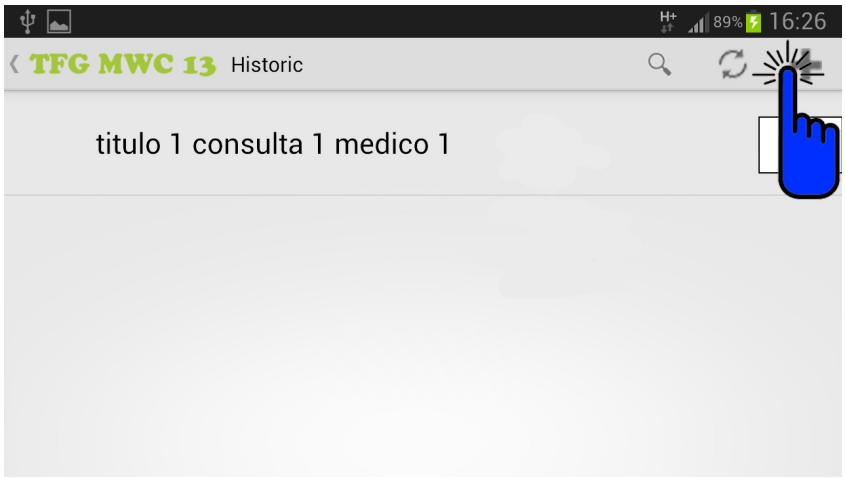
13.- Veremos nuestro registro de consultas cerradas. Seleccionamos una para ver los detalles de la misma.

Figura 4.1.14: Vista de la lista 'historic'.



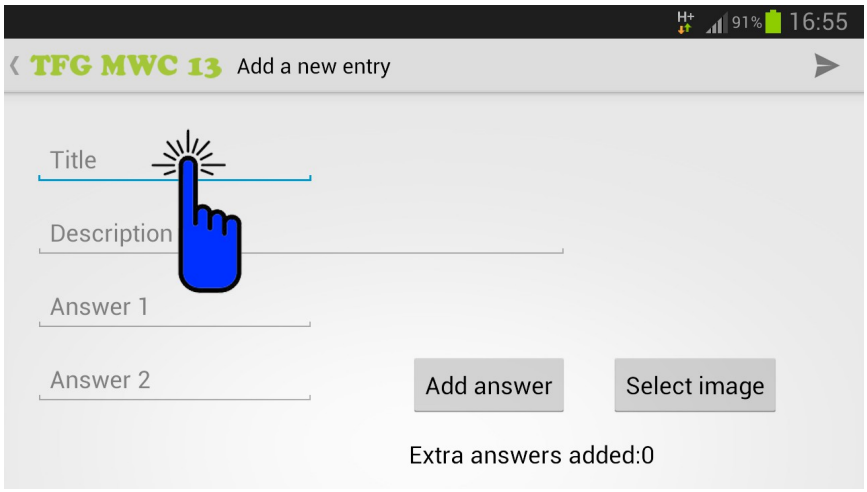
14.- Vemos los detalles de la consulta. Cuando fue cerrada la respuesta 1 tenia un voto y la 2 ninguno. Pulsamos el logo de la aplicación para volver atrás.

Figura 4.1.15: Detalles de una consulta en el registro histórico.



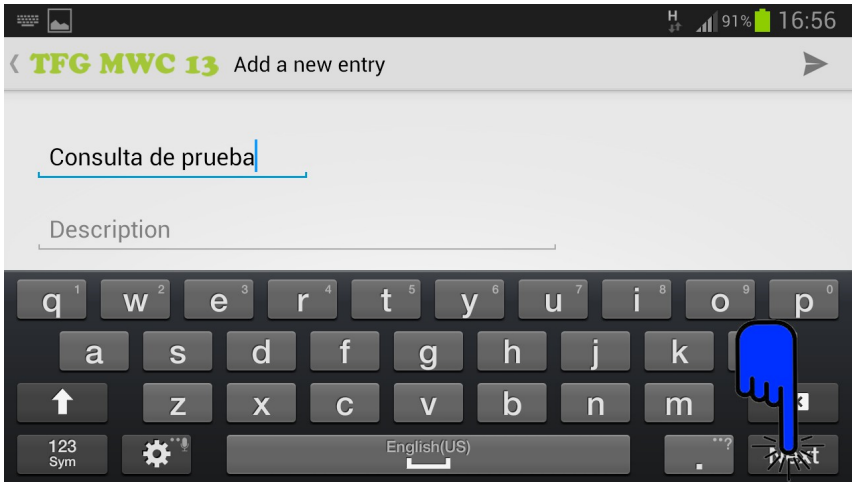
15.- Hasta ahora hemos repasado las funcionalidades de consulta de contenido, ya sea creado por nosotros o por otros usuarios. Ahora pasaremos a crea contenido nuevo. Pulsamos el botón 'add' pata añadir una nueva consulta.

Figura 4.1.16: Pulsar el botón 'add'.



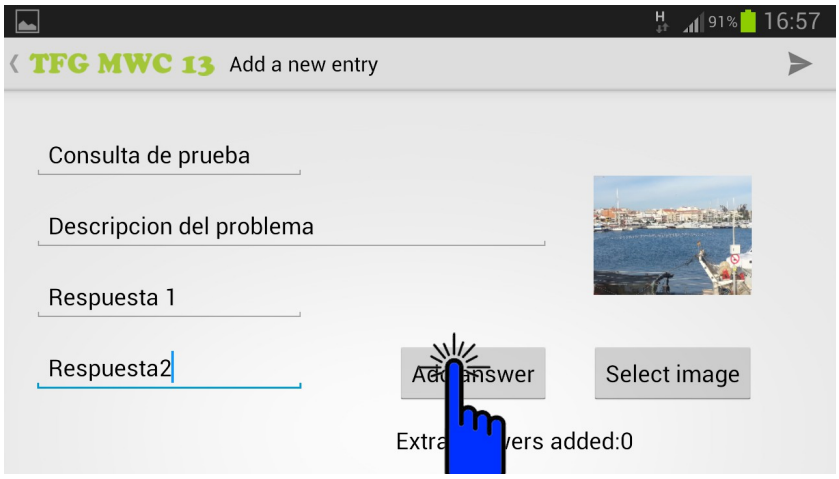
16.- Vemos el formulario para añadir una consulta. Primero pulsamos en el recuadro de 'Title' para que se nos muestre el teclado.

Figura 4.1.17: Vista del formulario de nueva consulta.



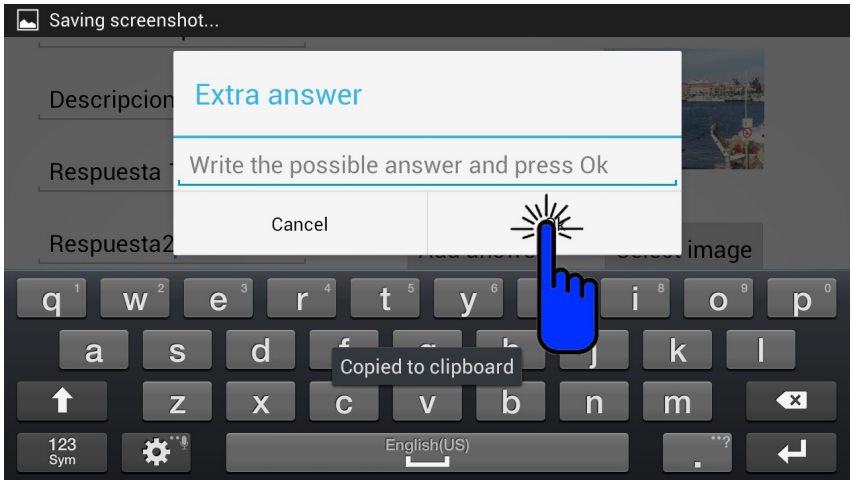
17.- Pulsamos el botón 'next ' en la parte inferior derecha del teclado para pasar al siguiente campo.

Figura 4.1.18: Pulsar botón 'next' para cambiar el foco de un campo al siguiente.



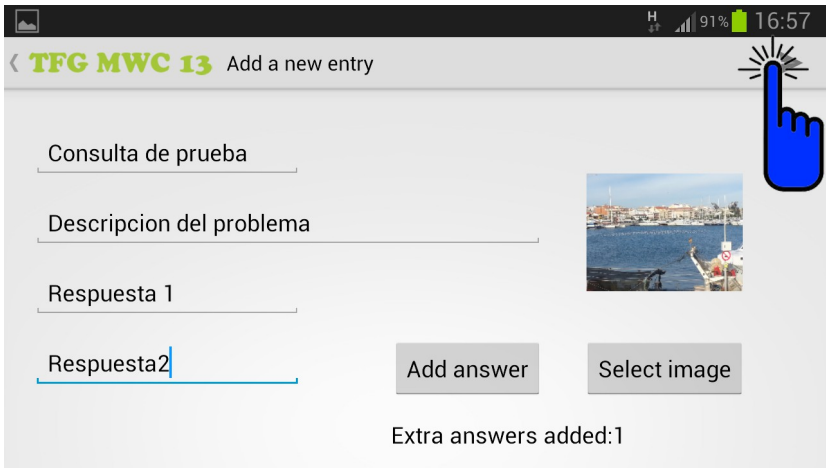
18.- Una vez completados todos los campos obligatorios, podemos añadir mas respuestas pulsando el botón 'add answer'.

Figura 4.1.19: Pulsar el botón 'add answer' para ver el formulario de respuestas extras.



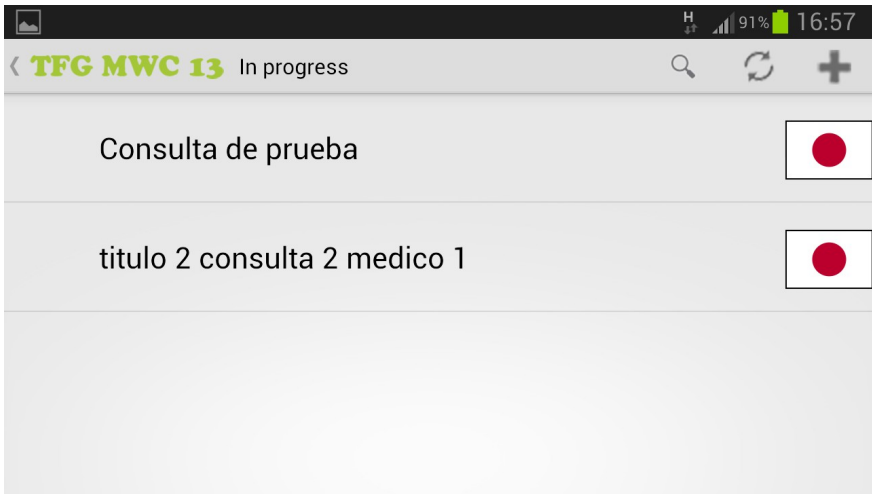
19.- El teclado se mostrará automáticamente, una vez escrita la respuesta pulsamos el botón 'OK'.

Figura 4.1.20: Pulsar 'OK' para añadir la respuesta.



20.- Una vez que hemos completado los campos obligatorios (titulo, descripción y dos respuestas) y añadido la imagen y las respuestas extras que consideremos oportunas, pulsamos el botón 'send' en la parte derecha de la barra superior.

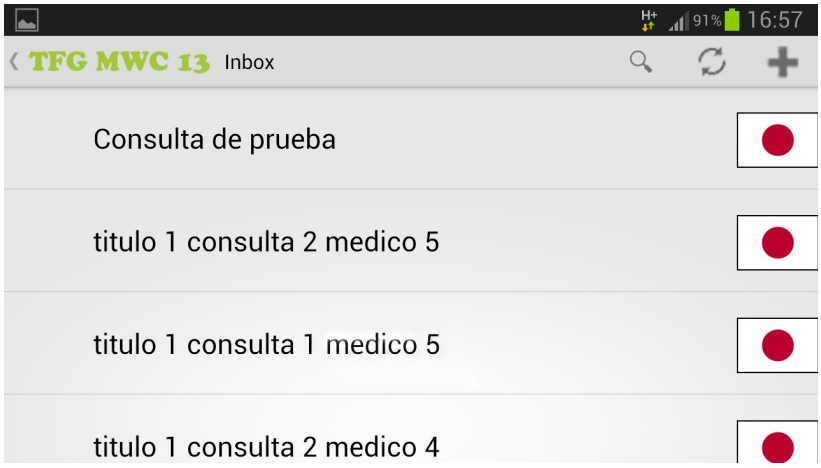
Figura 4.1.21: Todos los datos de la consulta están completos, pulsar el botón 'send'.



21.- Cuando el proceso de enviar la consulta al servidor finalice correctamente, se nos mostrará automáticamente nuestra lista de consultas en proceso. Vemos que la nueva consulta está en el primer lugar de la lista.

Figura 4.1.22: Vemos la lista 'in progress', la nueva consulta figura en la primera posición.

A continuación mostraremos una captura de la aplicación perteneciente a otro usuario (usuario 2):



22.- Vemos que, el usuario 2, puede contestar a la pregunta creada anteriormente (por el usuario 1). La figura 5.1.23 muestra la lista 'inbox' del usuario 2, donde la consulta 'consulta de prueba' sale en primer lugar.

Figura 4.1.23: La consulta creada por el usuario 1, aparece en la lista 'inbox' del usuario 2.

Finalmente dos detalles relacionados con las funcionalidades que hemos visto anteriormente.

23.- Si una de las listas de la aplicación ('in box', 'in progress' o 'historic') no contiene ningún elemento, se nos mostrará una imagen con un mensaje.

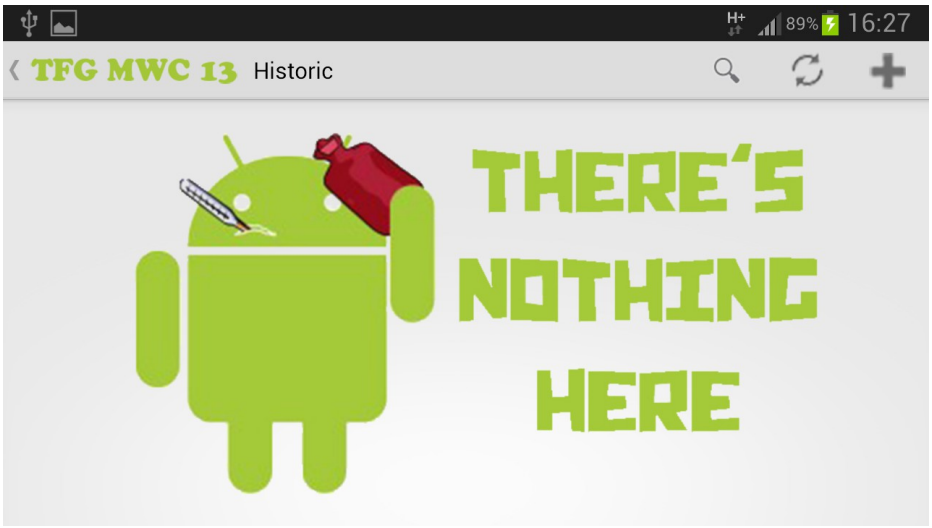
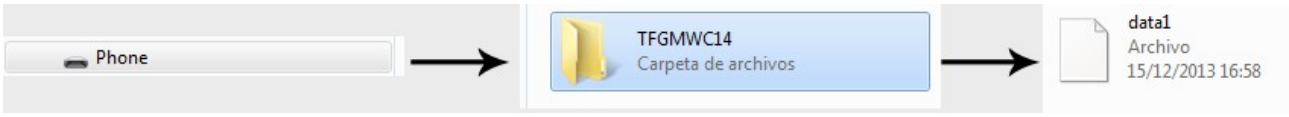


Figura 4.1.24: Imagen que veremos si una de las listas está vaciá.

24.- Todas las listas de la aplicación ('in box', 'in progress' o 'historic') se obtienen a través del servidor, al mismo tiempo se guarda una copia local. Por tanto la aplicación crea y mantiene tres ficheros en el disco local: data1, data2 y data3. Estos ficheros contienen la información de las tablas, si el usuario los borra manualmente o si resultaran corruptos por cualquier motivo, esto no causará ningún daño a la integridad de los datos.

Figura 4.1.25: Muestra el path (phone/TFGMWC14/data1) a uno de los ficheros locales que la aplicación crea automáticamente, en este caso el fichero data1.



4.2.- Aspectos interesantes en la interfaz de tabletas.

En esta sección veremos las principales diferencias de la interfaz de tabletas, con respecto a la de teléfonos móviles, vista antes.

1.- Gracias al mayor tamaño de pantalla del que disponemos en este tipo de dispositivos, en esta interfaz, los tres componentes básicos descriptos en la sección 2.6, se nos mostrarán en todo momento. En la figura 4.2.1 podemos ver la lista 'inbox' en el 'content fragment', al mismo tiempo que el 'action bar' es visible en la parte superior y el 'menu fragment' en el lateral. Las figuras 4.2.2 y 4.2.3 muestran las visualizaciones de las listas 'in progress' e 'historic'.

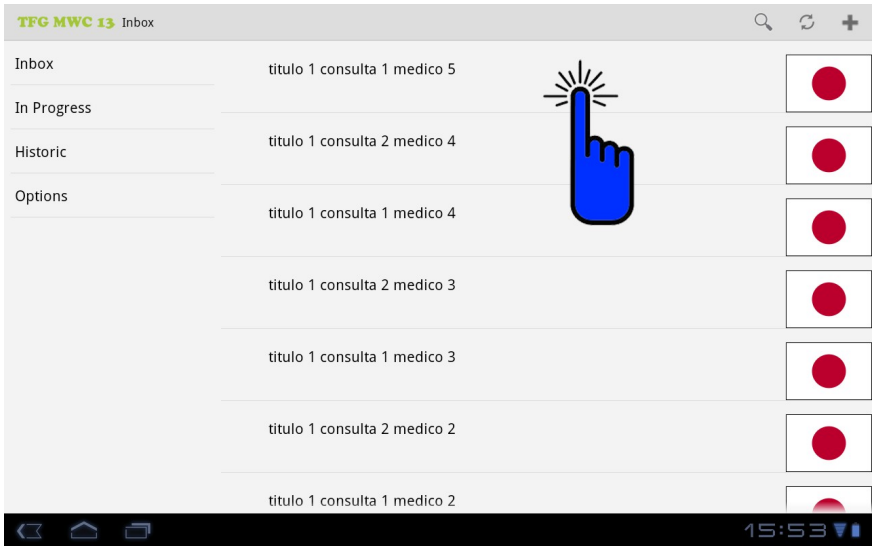


Figura 4.2.1: Vista de un la lista en la interfaz de tabletas, en este caso, la lista 'inbox' .

2.- En la vista de una consulta en detalle, se nos muestra toda la información de la pregunta y la imagen ocupa más proporción de pantalla. podemos verla en pantalla completa, si pulsamos sobre ella.

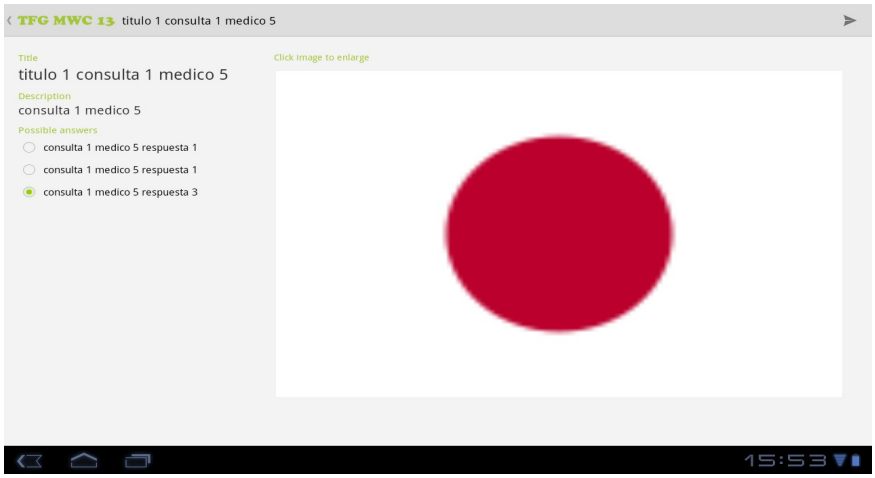


Figura 4.2.2: Vista de los detalles de una consulta en la interfaz de tabletas.

Conclusiones

Mi valoración final del resultado y el desarrollo del proyecto es muy positiva, esta valoración viene dada por tres aspectos complementarios: el trabajo en equipo, una temática interesante y una valiosa experiencia personal y de aprendizaje.

Destacar que, aunque en el transcurso de la carrera se realizan muchos trabajos prácticos en equipo, ninguno es de la complejidad de lo que aquí hemos intentado. Esto nos ha impuesto la necesidad de trabajar colaborando estrechamente de muchas maneras, como por ejemplo: llevando a cabo reuniones semanales, resolviendo dudas mutuamente, revisando continuamente algunos aspectos de nuestros trabajos para que se ajustaran a lo implementado por el compañero, e incluso, dando nuestra opinión en aspectos de diseño relativos al proyecto de la otra parte. Si consideramos que el resultado final ha sido satisfactorio, ello se debe en gran medida a nuestra disposición a colaborar y trabajar en favor del bien conjunto del proyecto, por encima de nuestras preferencias personales.

En cuanto a la temática que hemos elegido, creo que la tele-medicina es un campo con muy buena proyección de futuro y, aunque nuestra aplicación tiene muchas limitaciones, creo que puede ampliarse en gran medida con nuevas funcionalidades (como se comenta en el anexo 6.2). La ampliación más prioritaria seria la implementación del tercer modulo del proyecto, que como se indica en el apartado 2.3 no ha podido realizarse.
En cuanto a la plataforma elegida para el desarrollo de mi parte (Android), comentar que me parece una gran herramienta a nivel general. Sin embargo tiene un aspecto negativo, el hecho de que los primeros pasos pueden resultar complicados y frustrantes debido a la gran cantidad de nuevos conceptos a asimilar: actividades, fragmentos, ciclos de vida, funcionamiento de la librería de soporte oficial y otras extras, etc. El glosario del anexo 6.1 aclara el significado de algunos de estos términos.

Creo que la experiencia obtenida durante los tres meses de trabajo ha sido muy valiosa, puesto que hemos enfocado el proyecto desde un punto de vista profesional. Hemos teniendo en cuenta que algunas funcionalidades no podían incluirse dada su complejidad, o que, otras que creíamos terminadas y cerradas podían reconstruirse para mejorarlas notablemente (por ejemplo la navegación en la GUI). La perspectiva del concurso de aplicaciones del *Mobile World Congress* ha influido mucho en intentar llevar a cabo el proyecto de la mejor manera posible y en plantear un proyecto fácilmente ampliable con funcionalidades mucho mas compejas.

Finalmente destacar el gran trabajo de mi compañero Sergi, así como su ayuda y opinión en algunas decisiones relativas al diseño. Y destacar también la aportación de nuestro tutor, Simone Balocco, con su ayuda tanto en los planteamientos iniciales del problema y la solución como en la coordinación del trabajo semanal.

Anexo

6.1.- Glosario.

Sistema Operativo: SO, frecuentemente OS, del inglés *Operating System,* es un programa o conjunto de programas que en un sistema informático gestiona los recursos de hardware y provee servicios a los programas de aplicación, ejecutándose en modo privilegiado respecto de los restantes y anteriores próximos y viceversa.

Linux: Es un núcleo libre de sistema operativo (también suele referirse al núcleo como *kernel*) basado en *Unix*. Es uno de los principales ejemplos de software libre y de código abierto. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo. El desarrollo del día a día tiene lugar en la *Linux Kernel Mailing List Archive*

Android: es un sistema operativo basado en Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, inicialmente desarrollado por Android, Inc. Google respaldó económicamente y más tarde compró esta empresa en 2005. Android fue presentado en 2007 junto a la fundación del Open Handset Alliance: un consorcio de compañías de hardware, software y telecomunicaciones para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.

Java: Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

Librería: En informática, una librería (del ingles *library*) es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

Cliente: Es una aplicación información o un computador que consume un servicio remoto en otro computador, conocido como servidor, normalmente a través de una red de telecomunicaciones.

Servidor: Una aplicación informática o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes. Algunos servicios habituales son los servicios de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Este es el significado original del término. Es posible que un ordenador cumpla simultáneamente las funciones de cliente y de servidor.

Base de datos: Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. Existen programas denominados sistemas gestores de bases de datos, abreviado DBMS, que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos DBMS, así como su utilización y administración, se estudian dentro del ámbito de la informática.

JSON: Acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

Activity: Una actividad es una cosa simple que el usuario puede hacer. Casi todas las actividades interactúan con el usuario, por tanto la clase de la Actividad se encarga de crear una ventana en la que se situará la interfaz gráfica con la función setContentView(View). Aunque las actividades se presentan al usuario como ventanas en pantalla completa, también pueden ser usadas de otras maneras: ventanas flotantes o incrustadas en otra actividad.

Fragment: Un Fragmento representa un comportamiento o una porción de interfaz de usuario de una actividad. Múltiples fragmentos pueden combinarse en una única actividad para construir una interfaz de múltiples paneles o pueden rehusarse en actividades diferentes. Se puede pensar en un fragmento como una sección modular de una actividad, que tiene su propio ciclo de vida, recibe su propio input, y puede ser añadido o eliminado mientras la actividad funciona.

GUI: La **interfaz gráfica de usuario**, conocida también como **GUI** (del ingles *graphical user interface*) es un programa informático que actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador.

Widget: Es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o *Widget Engine*. Entre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

Action Bar: La barra de acciones es una característica de la ventana que identifica la posición del usuario dentro de la aplicación y provee de acciones y herramientas de navegación. Usar una barra de acciones ofrece a los usuarios una interfaz familiar entre aplicaciones que el sistema adapta a diferentes configuraciones de ventana.

Time out: Se utiliza en algunos programas para indicar el tiempo máximo que debe esperar el mismo para conectarse antes de abortar una tarea. Si se conecta y se mantiene activa la misma, el comando se desactiva.

6.2.- Posibles extensiones.

Añadir sistema de toma de decisiones:

La extensión básica del proyecto es añadirle el sistema de toma de decisiones, teóricamente debía haber sido un clasificador-recomendador que utilizando unos valores de confianza almacenados en la base de datos de los médicos, debería dar una respuesta más fiable en base a qué médicos la habían respondido, y a su vez modificar estos valores en base a si fueron o no correctas la soluciones brindadas por éstos.

Organizar las paginas *php* en una sola con un switch-case*:*

Actualmente, las funcionalidades del servidor están relacionadas con una página *php*, organizándolas de esta manera es mas fácil de modificar y visualizar, pero una posible mejora se basaría en simplemente tener una sola página *php* y que contenga un switch-case en el cual cada caso corresponde a una funcionalidad diferente, sería pues, obligatorio por parte del cliente, indicar al servidor que funcionalidad quiere llamar.

Implementar otros clientes para navegador y/o para ordenador:

Otra posible extensión, esta vez sería únicamente del cliente, sería extenderlo para diferentes plataformas, por ejemplo, diseñar el cliente en *html* para navegadores, o hacer un programa ejecutable para ordenadores, dependiendo del sistema operativo que éstos usen.

Implementar un proceso independiente que nos notifique cambios:

Esta extensión consistiría en implementar un proceso que siempre estaría activo, es decir que, este proceso seria independiente de la aplicación y seguiría ejecutando aun cuando la aplicación fuera cerrada. Su función seria consultar al servidor periódicamente y avisar al usuario cuando hay cambios: una nueva consulta que responder, una respuesta a una de nuestras consultas en progreso, etc.

Añadir comentarios a las respuestas:

Una opción interesante sería modificar la funcionalidad de responder a una consulta del cliente, de tal manera que se permitiera elegir una de las respuestas disponibles y poder dar la opción a añadir un pequeño comentario, por ejemplo, en un caso en el que se duda entre una serie de medicaciones a recetar, poder añadir como comentario la dosis de ésta.

Añadir la posibilidad de rechazar un pregunta:

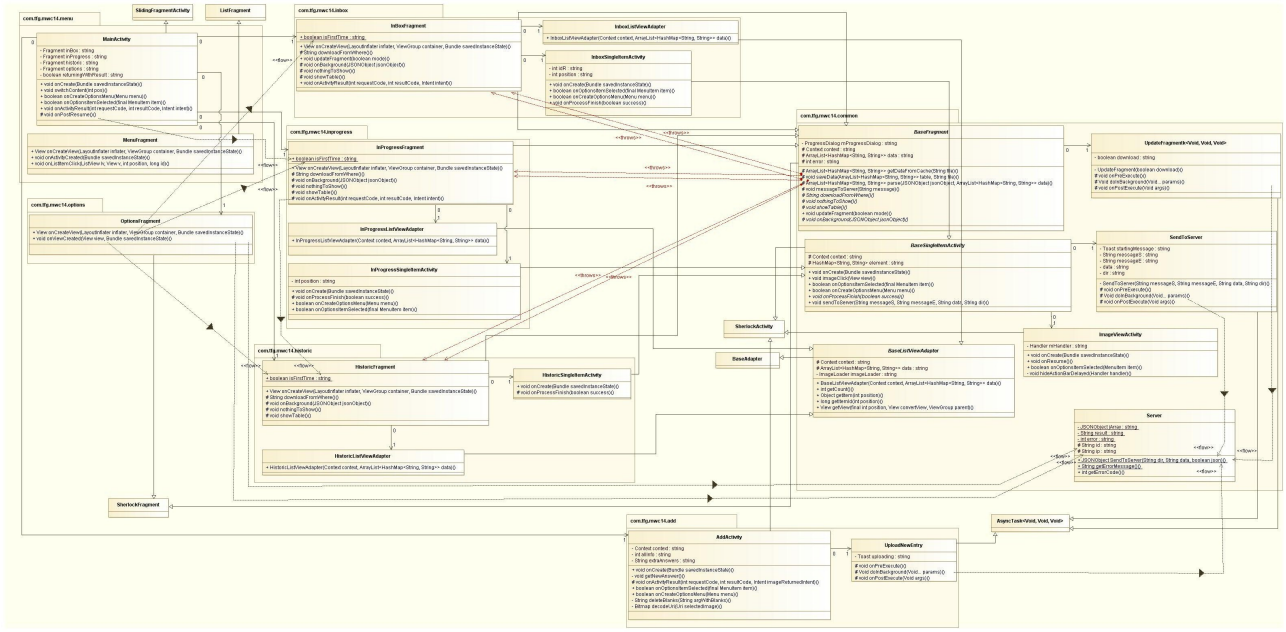
En rigor, esta opción ya es posible. Basta con no contestar las preguntas que no nos interesan y esperar a que el usuario que las ha creado las cierre.
La extensión consistiría en crear una respuesta por defecto para todas las preguntas. Esta respuesta sería del tipo: 'no responder' e indicaría al servidor que ese usuario no quiere (o no puede) responder a esa determinada pregunta.

Añadir la posibilidad de cierre automático de una pregunta:

Actualmente, las preguntas se cierran cuando el usuario que las ha creado lo indica explícitamente. La intencional consistiría en añadir la posibilidad de establecer una fecha máxima para el cierre automático de la consulta, esta opción se presentaría en forma de calendario en el formulario de creación de consulta nueva.

6.3.- Diagrama de clases.

La siguiente imagen contiene el diagrama de clases de la aplicación. Es un diagrama completo que muestra las relaciones entre clases, sus funciones y atributos.
Para verlo en resolución completa, consultar la página:
https://dl.dropboxusercontent.com/u/32678271/tfgmwc14%20Class%20diagram.jpg



6.4.- Descargas.

El **instalador** de la aplicación puede descargarse desde el siguiente link:
https://dl.dropboxusercontent.com/u/32678271/TFGMWC14.apk

Para descargar y configurar el **código fuente** del proyecto, es necesario seguir algunos pasos:

1. Descargar la ultima versión de Eclipse ADT:
   - http://developer.android.com/sdk/index.html

2. Descargar las librerías *ActionBarSherlock* y *SlidingMenu* de las siguientes direcciones y extraerlos en una carpeta:
   - https://dl.dropboxusercontent.com/u/32678271/AbsLib.rar
   - https://dl.dropboxusercontent.com/u/32678271/SlidingMenuLib.rar

3. Descargar el código del cliente desde el siguiente enlace y extraerlo en una carpeta:
   - https://dl.dropboxusercontent.com/u/32678271/TFGMWC14.rar

4. Para configurar *ActionBarSherlock* es necesario abrir Eclipse ADT y pulsar:
   4.1. 'File' → 'New' → 'Other' → Seleccionar la carpeta 'Android' → 'Android Project from Existing Code' → 'Next' → 'Browse' → Seleccionar la carpeta extraída de *ActionBarSherlock* → Marcar 'Copy projects into workspace' → 'Finish'.
   4.2. Con esto se copiaran los ficheros a nuestra carpeta *workspace* y se creara un nuevo proyecto en nuestro ADT. Pulsar sobre el proyecto creado con el botón derecho y seleccionar: 'Properties' → 'Android' → Asegurarse de que la opción 'Is Library' está marcada y de que la opción 'Project Build Target' coincide con lo definido en el *AndroidManifest.xml* en el atributo `android:targetSdkVersion`.

5. Configurar la librería *SlidingMenu:*
   5.1. Repetir el proceso 4.1, seleccionado la carpeta extraída de *SlidingMenu.*
   5.2. Repetir el proceso 4.2, comprobando que la librería añadida en el paso anterior figura como dependencia.

6. Repetir el proceso una vez más para el código del cliente:
   6.1. Repetir el proceso 4.1, seleccionado la carpeta extraída de *TFGMWC14.*
   6.2. Repetir el proceso 4.2, comprobando que las librerías añadidas en los pasos anteriores figuran como dependencias.

6.5.- Protocolo y JSON.

En esta sección repasaremos los detalles del protocolo de comunicación y el formato de los JSON.

Detalles del protocolo:

variables :

IP = **numero**
idM = id del médico. **String**
idC = id de la consulta. **String**
idR = id de la respuesta. **String**
tit = titulo de la consulta. **String**
descr = descripción de la consulta. **String**
resps = posibles respuestas. Formato: **resp1|resp2|resp3|**

Llamadas a páginas php:

Consulta nueva:

- args: "idMedico=" + idM + "&descripcion=" + descr + "&titulo=" + tit + "&resps=" + resps
- url : http://MIIP:1289/xampp/TFGETASTERISCO/guardar_consulta.php?" + args
- retorno: "si" o "mensaje de error"

Recibir lista consultas por resolver:

- args : "idMedico=" + idM
- url : http://MIIP:1289/xampp/TFGETASTERISCO/mostrar_consultas_resolver.php?" + args
- retorno: JSON con tags:

idConsulta;
titulo;
descr;
respuestas = array();
idRespuestas = array();
contadores = array();

Responder una consulta:

args : "idMedico=" + idM + "&idConsulta=" + idC + "&idRespuesta=" + idR
url : http://MIIP:1289/xampp/TFGETASTERISCO/resolver_consulta.php?" + args
retorno : "si" o "mensaje de error"

Recibir lista de consultas propias:

args : "idMedico=idM"
url : http://MIIP:1289/xampp/TFGETASTERISCO/mostrar_consultas_propias.php?" + args
retorno : JSON con tags:
idConsulta;
titulo;
descr;
respuestas = array();
idRespuestas = array();
contadores = array();

Cerrar una consulta abierta:

args : "idMedico=" + idM + "&idConsulta=" + idC
url : http://MIIP:1289/xampp/TFGETASTERISCO/cerrar_consulta.php?" + args
retorno : "si" o "mensaje de error"

Recibir lista de histórico:

args : "idMedico=idM"
url : http://MIIP:1289/xampp/TFGETASTERISCO/mostrar_historico.php?" + args
retorno : JSON con tags:
idConsulta;
titulo;
descr;
respuestas = array();
idRespuestas = array();
contadores = array();

Ejemplo de JSON:

A continuación tenemos un ejemplo del formato de los JSON.

```
{
"consultas":                                    // una lista global con todas las consultas
        [
                {                               // primera consulta de la lista
                 "idConsulta":1,                // identificador de la consulta
                "titulo":"blabla",              // titulo
                 "desc":"le duele mucho",       // descripción
                "respuestas":                   // lista de respuestas
                        [
                                {               // primera respuesta
                                "respuesta":"resp1"
                                },
                                {               // segunda respuesta
                                "respuesta":"resp2"
                                }
                        ],
                "idRespuestas":                 // lista con los identificadores de las respuestas
                        [
                                {
                                "idRespuesta":"1"       // primer id
                                },
                                {
                                "idRespuesta":"2"       // segundo id
                                }
                        ],
                "contadores":                   // lista con los contadores de cada respuesta
                        [
                                {
                                "contador":"3"          // primer contador
                                },
                                {
                                "contador":"0"          // segundo contador
                                }
                        ],
                "img":"url imagen "             // url de la imagen
                 }
        ]
}
```