UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universitat Politècnica de València

Desarrollo de una aplicación para evaluar y detectar el deterioro cognitivo

Proyecto Final de Carrera

Grado en Ingeniería Informática

Autor: Héctor Badenes Tur

Director: José Alberto Conejero Casares

Directora experimental: Patricia Villacampa Fernández

27 de junio de 2017

Resumen

CorApp (Cognitive Rate App) es una aplicación de evaluación y recogida de datos destinada a ayudar a los equipos dedicados a la investigación clínica, concretamente en el ámbito de la detección del deterioro cognitivo. Esta evaluación se llevará a cabo mediante dos métodos: la utilización de cuestionarios, que se utilizarán para realizar un seguimiento físico y psicológico de los pacientes, y la realización de una prueba especializada, el ACE-III.

El ACE-III es una herramienta de detección cognitiva recomendada para uso de profesionales de la salud e investigadores en pacientes mayores de 50 años con sospecha de demencia.

La propuesta consiste en el desarrollo de una aplicación Android para la recolección de datos sobre los pacientes que están siendo estudiados mediante las herramientas mencionadas. La aplicación se encuentra principalmente orientada a profesionales de la salud, tanto psicólogos como enfermeros, que se encargarán de interpretar los datos obtenidos, aunque también podrán utilizarla los pacientes que podrán contestar a los cuestionarios desde sus casas, evitando así la necesidad de desplazarse al centro médico para ser evaluados.

La aplicación debe ser capaz de dar de alta pacientes en el sistema; crear, evaluar y almacenar los resultados de los cuestionarios; y realizar la evaluación del ACE-III. También ha de ser capaz de medir los tiempos de cada prueba realizada, funcionar en modo offline y compartir los datos a un medio donde resulten más manejables sin la necesidad de usar esta herramienta.

Para realizar el correcto manejo de los datos será necesario el desarrollo de un servicio web que almacene los datos recogidos y sincronice los dispositivos, este será desarrollado con el lenguaje PHP y utilizará una base de datos MySQL.

La aplicación se desarrollará para el sistema operativo Android 5.0 mediante el uso del IDE oficial, Android Studio.

Palabras clave: ACE, Deterioro Cognitivo, Android, Encuesta, PHP, Psicología.

Tabla de contenidos

1. Introducción

1.1. Entorno

Todos los estudios modernos apuntan a que la actividad social y la mente son los factores clave para alcanzar el "envejecimiento activo", es decir el proceso de optimización de las oportunidades de salud, participación y seguridad con el fin de mejorar la calidad de vida a medida que las personas envejecen. La disminución de la capacidad cognitiva está fuertemente relacionada con el estilo de vida, así como el compromiso social, la estimulación cognitiva, la nutrición y la actividad física.

Durante el envejecimiento, todas las personas desarrollan cierto grado de deterioro cognitivo. Este declive natural puede ser acelerado por desuso, enfermedad, factores psicológicos y sociales. Del lado opuesto, la descomposición cognitiva es también responsable del aislamiento social y de la depresión.

Por todas estas razones una gran cantidad de organizaciones, tanto gubernamentales como privadas, están poniendo el foco en la detección prematura del deterioro cognitivo. Para ello están desarrollando nuevas técnicas de detección como el ACE-III y otros tipos de cuestionarios.

En este contexto la cantidad de información generada puede crecer desmesuradamente, cada prueba se realiza en papel y los resultados tienen que traspasarse a algún formato digital como ‘Excel’ o ‘Access’ para poder gestionarlos correctamente y poder también compartirlos con la comunidad científica. Esta actividad ocupa gran parte del tiempo de trabajo de los investigadores y, además, propicia la aparición de errores en los datos debidos a la copia manual.

Otra problemática con la que se deben enfrentar los profesionales es la medida de tiempos. Para ellos el tiempo que tarda un paciente en realizar una determinada prueba otorga información valiosísima, pero el único medio del que disponen para realizar la medición es utilizando un cronometro, esto distrae y angustia a los pacientes, provocando que se pongan nerviosos y les cueste más realizar las actividades que en otros contextos. El resultado es la obtención de unos datos poco fiables, ya que no se puede cuantificar hasta que punto se han visto los pacientes influenciados por el entorno.

En este es el ámbito se enmarca el proyecto, hay una necesidad de herramientas especializadas que ayuden a los investigadores a realizar tareas

que para ellos ya son rutinarias y en las que desempeñan un tiempo que podrían dedicar a analizar resultados y extraer conclusiones.

1.2. Objetivos

Desarrollo de una aplicación Android que nos permita realizar los cuestionarios necesarios para evaluar el deterioro cognitivo en mayores de cincuenta años.

En primer lugar, si se pretende que el proyecto sea útil para una gran cantidad de profesionales y sus pacientes este deberá ser desarrollado en una plataforma de fácil acceso para todo el mundo, que sea portable, con la que preferiblemente ya estén habituados de antemano y que este muy extendida. Actualmente en España un 78,17% de los dispositivos móviles usan el sistema operativo Android [Stat], su gran disponibilidad unida al bajo coste de los dispositivos y su implantación en el mercado hacen de esta la opción más recomendable para llevar a cabo nuestro proyecto.

Por otro lado, se pretende que la aplicación almacene los datos en nube, permitiéndonos operar con los mismos datos, aunque cambiemos de dispositivo, haciendo posible que un equipo pueda trabajar simultáneamente desde distintos dispositivos con los mismos datos. Otro motivo por el que nos es indispensable el almacenamiento en nube es la necesidad de que los propios pacientes puedan contestar desde sus casas a los cuestionarios que les propongan los evaluadores. Esta característica, que es una gran ventaja para los investigadores y pacientes, nos supone un problema técnico. La aplicación será usada en el ámbito hospitalario, muchas veces desde smartphones y tabletas sin conexión a la red de datos, que dependerán, por tanto, de la disponibilidad de la conexión wi-fi. Como se prevé que la red falle muy frecuentemente la aplicación deberá disponer de una base de datos local que le permita operar independientemente en caso de desconexión, una vez recupere la conexión esta deberá iniciar la sincronización con el servidor.

1.3. Descripción

El sistema seleccionado es una aplicación desarrollada para las versiones del sistema operativo Android desde la 5.0 en adelante. Dicha aplicación hará uso

de la interacción con un servicio web PHP que se encargará de la sincronización y almacenamiento de los datos.

La aplicación se divide en los siguientes bloques funcionales:

- **Gestión de Encuestas.** Las encuestas se podrán crear dinámicamente mediante una herramienta integrada en la aplicación. Las encuestas dispondrán de las cuatro funciones básicas CRUD.
- **Gestión de Pacientes.** El evaluador podrá gestionar los datos de los pacientes que serán evaluados. Los pacientes también dispondrán de las cuatro funciones básicas CRUD.
- **Evaluación.** Los pacientes podrán ser evaluados mediante las encuestas creadas o mediante la prueba estandarizada ACE-III. La evaluación de las encuestas la pueden llevar a cabo tanto el evaluador como el paciente, pero la evaluación del ACE-III siempre deberá realizarse por el evaluador que controlará la prueba.
- **Resultados.** Los evaluadores podrán consultar los resultados de todas las pruebas realizadas por los pacientes en la propia aplicación.
- **Exportación.** Los datos podrán ser exportados a un medio que resulte más fácil de compartir y donde se puedan visualizar mediante herramientas más estandarizadas.

La diferenciación de las funciones de la aplicación en función con el rol del usuario (Evaluador o Paciente) ara necesaria la creación de un sistema de autentificación mediante usuario y contraseña, según el tipo de usuario la misma aplicación cambiará su comportamiento.

A través de estas actividades se pretende poder realizar un seguimiento completo sobre el paciente, se registrarán datos físicos sobre el paciente, estilo de vida, evaluaciones, etc. También se pretende conseguir familiarizar a los usuarios, que serán siempre personas mayores, con las nuevas tecnologías, por ello resulta aún más interesante la opción de que ellos mismos resuelvan las encuestas desde sus dispositivos.

2. Estudio estratégico

2.1. Introducción

En este capítulo se trata de situar el sistema a desarrollar en el contexto en el que se enmarca en la actualidad. En primer lugar, veremos una introducción teórica y la situación legal en la que se encuentra. Más adelante trataremos de reflejar el estado actual del mercado en el que se enmarca nuestro sistema mediante la búsqueda de sistemas similares y el análisis de sus distintas características. Esto nos permitirá hallar aquellas áreas conceptuales que estén menos trabajadas, para finalmente elaborar una lista de aquellas que resultan más interesantes para la aplicación.

2.2. Marco teórico

Para entender completamente el propósito de la aplicación resulta indispensable conocer en que consiste la prueba ACE-II y cuál es su importancia.

El Addenbrooke's Cognitive Examination (ACE) es un test de cribado para el diagnóstico de demencia. Tras varias revisiones, actualmente se utiliza la tercera versión del test (ACE-III) [Ace1], Este test ha sido traducido y adaptado al español [Ace2].

Herramientas como ésta son de gran actualidad ya que existe una necesidad de contar con herramientas de cribado ágiles y fiables que permitan detectar la demencia en alguno de sus estadios iniciales por los especialistas en medicina general. Además, estas herramientas tienen la utilidad de poder incorporarse a evaluaciones más complejas de los individuos con el fin de buscar relaciones de causalidad con otros síntomas y enfermedades. Este test evalúa la relación entre la fluidez verbal y el lenguaje con respecto a la orientación y el recuerdo diferido. Permite además detectar la demencia frontotemporal con respecto a la enfermedad de Alzheimer.

No obstante, hasta la fecha no existen estudios que incluyan el tiempo invertido por el paciente en la cumplimentación del test, y sobre todo en cada una de las dimensiones. Con el desarrollo de esta aplicación se desea introducir esta información como criterio a tener en cuenta a la hora de modular los resultados del test administrado. Es conocido que el estudio de los tiempos de respuesta permiten, por ejemplo, realizar cribados preliminares a la hora de diagnosticar

el TDAH (Trastorno de Deficit de Atención e Hiperactividad) [ACE3], así como el estudio del deterioro cognitivo [ACE4].

Para entender cómo se realiza la evaluación del ACE-III debemos saber que está compuesto por un conjunto de preguntas clasificadas en cinco grandes áreas: Atención, Memoria, Fluencia, Lenguaje y Visuoespacial.

Para evaluar el ACE-III se valorará cada una de las preguntas según el criterio que esta indique y su puntuación se sumará al total de su área. Finalmente se presentará el resultado total, comprendido entre cero y cien, y el resultado parcial de las distintas áreas. También se mostrará el resultado individual de cada una de las preguntas, así como el tiempo que se ha tardado en contestar.

2.3. Sustitutos potenciales

Este tipo de pruebas, por su formato, se realizan habitualmente mediante herramientas muy básicas. Tan solo con lápiz, papel y un cronometro el evaluador ya sería capaz de realizar el estudio sin la necesidad de desarrollar ningún sistema alternativo. Esto podría levantar reticencias entre evaluadores y pacientes que, acostumbrados, gracias al modelo educativo actual, al uso constante de estos medios se resistan al cambio y se muestren reacios a utilizar el nuevo sistema.

2.4. Sistemas similares

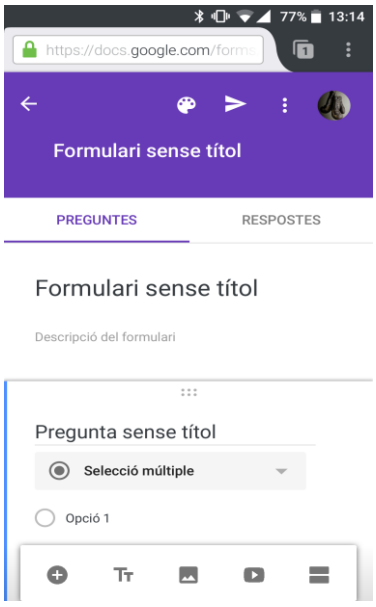
A lo largo de los años, la investigación clínica ha ido evolucionando intentando adaptarse al crecimiento tecnológico que ha imperado en estos últimos años. La tecnología ha pasado de ser algo ajeno para la mayoría de personas a ser algo cotidiano. Es por ello que se antoja necesaria la introducción de las TIC dentro de los laboratorios, hospitales y centros de salud.

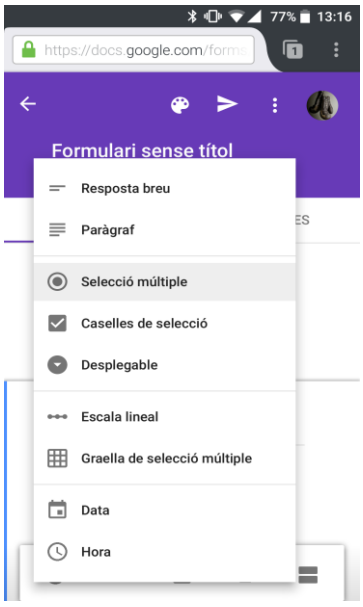
A pesar del crecimiento tecnológico en todos los ámbitos no existen aún herramientas digitales específicas para realizar la prueba ACE-III, esto es debido a que se trata de una técnica novedosa. Por otra parte, si que existen herramientas y aplicaciones capaces de generar encuestas y formularios con los que se podría realizar la evaluación del paciente. En este punto vamos a analizar algunas de las herramientas disponibles para nuestra plataforma, tanto apps disponibles en Google Play como herramientas web. Dado que la idea

principal del sistema es que sea de descarga gratuita se ha optado por analizar las versiones gratuitas de las aplicaciones elegidas

2.4.1. Google Forms

Se trata de una herramienta 100% gratuita desarrollada por Google para la creación y resolución de encuestas. Forms se trata de una aplicación web que podemos usar desde el navegador de nuestro dispositivo móvil o desde el pc. Es muy personalizable, permite cambiar colores, textos, insertar imágenes y logos, etc. Las encuestas creadas se pueden compartir con los interesados enviando el link que genera a través del email o cualquier otra plataforma de mensajería, y estas se pueden resolver sin la necesidad de utilizar ningún software específico. Muestra los resultados tanto individualmente como estadísticamente sobre el conjunto de encuestados. Al tratarse de una herramienta web no es posible utilizarla en modo offline.



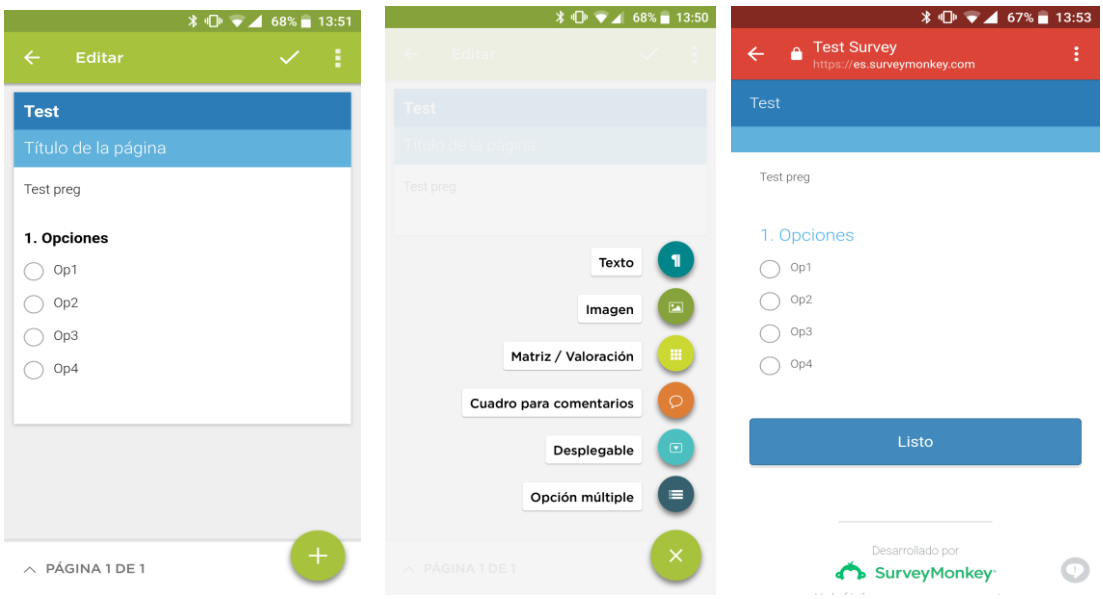


2.4.2. SurveyMonkey

SurveyMonkey es una aplicación Android desarrollada por el estudio SurveyMonkey con características muy similares a las de Google Forms, se

puede encontrar en Google Play y es una de las aplicaciones más populares de su tipo. La aplicación permite la creación de encuestas, la personalización de colores, inserción de imágenes y el análisis tanto individual como colectivo de las respuestas. Aunque se trate de una aplicación nativa de Android solo lo es para la gestión de las encuestas, ya que su resolución se realizara a través de la web. Las encuestas se enviarán, por tanto, mediante un link que genera la aplicación y que tendremos que distribuir mediante las herramientas de mensajería habituales, y al igual que Forms será necesaria la conexión a internet para resolverlas.

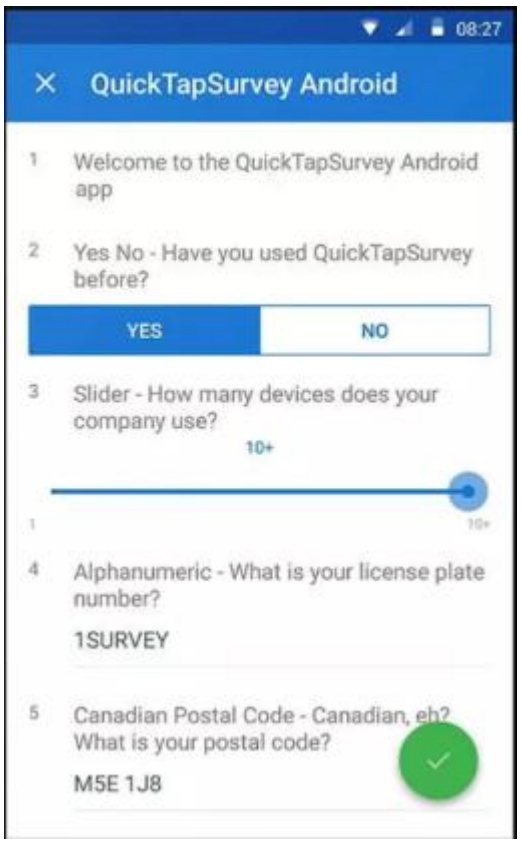
Esta aplicación solo es gratuita en modo Básico, limitándonos a encuestas con un máximo de 10 preguntas y 100 respuestas por encuesta. Si nuestras necesidades son mayores será necesario contratar uno de sus planes.



2.4.3. QuickTapSurvey

Esta es otra herramienta que podemos encontrar en Google Play, está desarrollada por TabbleDabble Inc. Esta aplicación, al contrario que las anteriores, no ofrece la posibilidad de gestionar las encuestas, la gestión se realiza íntegramente desde su plataforma web mientras que su resolución se llevará a cabo mediante la aplicación. A consecuencia de esto para la gestión será necesaria la conexión a internet mientras que las evaluaciones se podrán realizar offline.

QuickTapSurvey es capaz de exportar los resultados a Excel, CVS, y Salesforce.

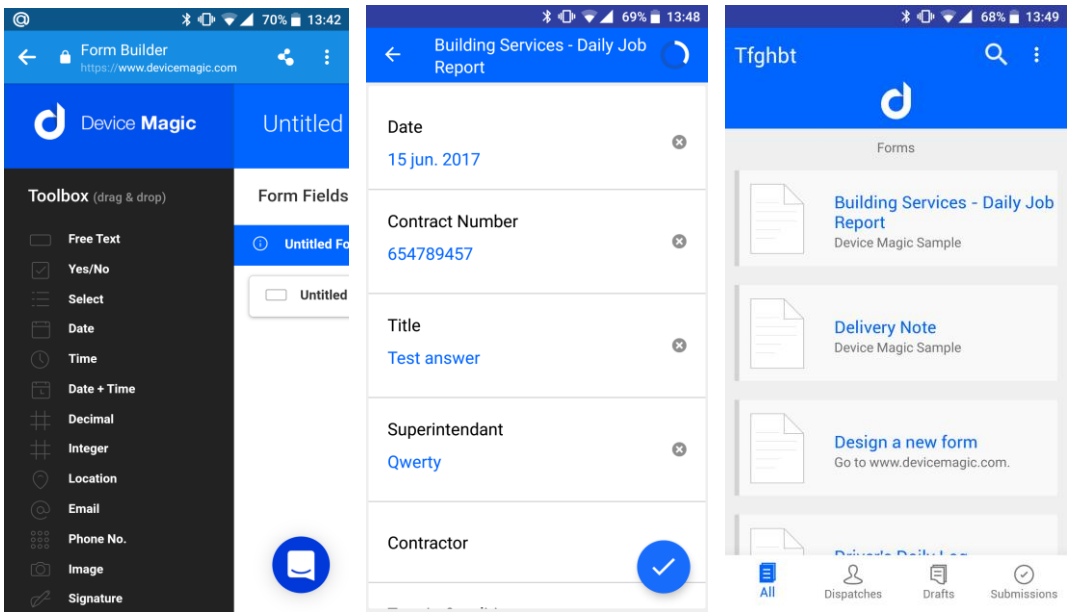


2.4.3. Forms

Forms es una aplicación disponible en Google Play desarrollada por Device Magic Inc, sus características son muy similares a las de QuickTapSurvey, toda la gestión se desarrolla en la web mientras que las evaluaciones se realizan en el dispositivo. Al igual que QuickTapSurvey tan solo la evaluación se puede realizar offline. Cabe destacar que la aplicación web donde debemos realizar la gestión no se encuentra bien optimizada para pantallas de smartphones, en este entorno su usabilidad es deficiente y requeriríamos de una tableta o pc para realizar nuestro trabajo.

Forms es capaz de exportar los resultados a PDF, Word y Excel.

Esta aplicación tampoco cuenta con versiones gratuitas, aunque su plan básico si cuenta con un periodo de prueba. La limitación de sus planes no reside en el número de encuestas sino con las funciones disponibles.



2.5. Análisis

Tras un estudio detallado de las aplicaciones propuestas en el punto anterior, realizamos ahora un análisis comparativo entre ellas tomando como referencia los aspectos que resultan más interesantes para nuestros objetivos. De esta manera, podremos encontrar más fácilmente aquellos aspectos que resulta interesante trabajar en nuestra. La cuantificación del cumplimiento o no cumplimiento de las características por parte de los sistemas se ha realizado mediante la siguiente escala:

Un punto (1): para aquellas características que no se cumplen.

Dos puntos (2): para aquellas características que se cumplen parcialmente o de manera indirecta mediante otra herramienta.

Tres puntos (3): para aquellas características que se cumplen totalmente en la aplicación.

Las características analizadas serán el cumplimiento de los bloques funcionales deseados para la aplicación Gestión de encuestas, Gestión de

pacientes, Evaluación, Resultados y Exportación. En el bloque de gestión de pacientes buscamos cualquier forma de definir usuarios encuestados, no es necesario que sea algo tan especifico, y en el apartado de Evaluación nos referiremos solo a la parte de las encuestas, ya que habíamos determinado previamente que no existen herramientas que evalúen el ACE-III.

	Gestión de encuestas	Gestión de pacientes	Evaluación	Resultados	Exportación
Google Forms	(2)	(1)	(2)	(2)	(2)
SurveyMonkey	(3)	(1)	(2)	(3)	(1)
QuickTapSurvey	(2)	(1)	(3)	(2)	(2)
Forms	(2)	(1)	(3)	(2)	(2)

Tabla 2.1: Análisis de Características

Lo más destacable de la tabla es que ninguna de estas aplicaciones ofrece soporte para gestionar listas de evaluados, ya que están pensadas para evaluaciones más generales y no para realizar seguimientos, esto ya nos ofrece un carácter diferenciador sobre todas ellas. También cabe destacar que ninguna de estas herramientas proporciona todas sus características en la aplicación, mostrado todas, de un modo u otro, dependencia de aplicaciones web externas. Esto nos deja una oportunidad de destacar entre los otros sistemas analizados puesto que ninguno cumple las características deseadas en el ámbito de la realización de encuestas. Si además de todo esto añadimos que ninguna de estas herramientas ofrece soporte para medir los tiempos que tarda el evaluado en contestar y que nuestra aplicación será la única que ofrezca la posibilidad de realizar el test ACE-III sin la necesidad de tener que confeccionarlo como si se tratase de una encuesta mas, el resultado es una aplicación única, muy especializada en el ámbito que nos atañe y sin ningún rival de las mismas características.

2.6. Síntesis

Una vez hemos hecho el análisis de las distintas aplicaciones y observando la tabla expuesta, podemos sacar algunas conclusiones sobre los aspectos que nuestra aplicación debería tratar con más o con menos profundidad.

En cuanto a la funcionalidad podemos deducir que los aspectos clave para el éxito del proyecto van a ser dos. Primero la gestión de los pacientes/evaluados, ningún competidor tiene este aspecto desarrollado y ofrece al evaluador una mejor organización de los resultados, esto se traduce en un mejor seguimiento del paciente y portante una reducción considerable en el tiempo que este dedica a realizar estas tareas. Segundo la integración de la prueba ACE-III, esto es vital para los investigadores en el campo de la detección del deterioro cognitivo y ninguna aplicación actual lo ofrece.

Otro aspecto a tener en cuenta es el coste de las aplicaciones, como hemos visto la mayoría no ofrecen planes gratuitos o estos son muy limitados, nuestra aplicación, en cambio, no supondrá ningún coste para los equipos que van a usarla en sus ensayos clínicos.

2.7. Conclusión

En este capítulo hemos realizado un estudio con la intención de destacar las características a destacar y encontrar nuestras fortalezas respecto al mercado. Para ello hemos elegido un conjunto de sistemas similares al nuestro que ya se encuentran entre los más populares en nuestra plataforma. Esto nos ha ayudado a descubrir algunos aspectos a incluir en nuestra aplicación además de los que habíamos decidido previamente, aumentando así el alcance del sistema a desarrollar.

La realización de este estudio nos ha permitido perfilar más la aplicación. De esta manera podremos lograr que la siguiente especificación de requisitos sea lo más concreta posible, puesto que ya sabemos con certeza que aspectos queremos trabajar y, por tanto, los requisitos que debe cumplir la aplicación para conseguir nuestros objetivos.

3. Especificación de requisitos

3.1. Introducción

En este apartado se trata de reflejar con la máxima fidelidad cuales van a ser las funcionalidades y el comportamiento del sistema. En primer lugar, reflejaremos los posibles usuarios del sistema. En segundo lugar, desarrollaremos la descripción de los requisitos del sistema, tratando de englobar todos aquellos requisitos (funcionales y no funcionales) que puedan resultar importantes a la hora del desarrollo y la implantación de la aplicación.

Para la captación de los requisitos hemos contado con la ayuda de miembros del departamento de psicología de la universidad de Valencia.

Este apartado se ha redactado en base al estándar IEEE 830, que es uno de los más extendidos en cuanto a este ámbito se refiere.

3.2. Identificación de interesados

Una identificación de los stakeholders incorrecta puede llevar a incumplir los objetivos del proyecto debido a los riesgos no identificados, la falta de información, o el retrabajo debido a entregables que no cumplen con las expectativas.

Podemos realizar una división de estos en cuatro grupos diferentes: los usuarios que serán evaluados (los pacientes), los familiares de los pacientes (Al ser pacientes de avanzada edad podemos prever que serán ayudados en muchos casos), el personal médico encargado del estudio (los evaluadores) y aquellos que forman parte del proyecto (programadores futuros).

1. **Pacientes:** Personas de avanzada edad (se prevé que se trate de personas en edad de jubilación) que serán el objeto de las evaluaciones, estas podrán utilizar la aplicación directamente en el caso de que el evaluador encuentre conveniente enviarles alguna encuesta.

2. **Familiares:** Formado por el entorno más cercano del paciente. Como se pretende que los pacientes usen la aplicación desde sus casas se espera que las personas de su entorno les ayuden a adaptarse con las nuevas tecnologías en los casos que estos no lo hayan hecho aún.
3. **Evaluadores:** Personal sanitario que realizará el uso completo de la aplicación, realizará todas las gestiones sobre pacientes y encuestas, podrá realizar evaluaciones o enviar encuesta a los pacientes, realizará la evaluación del ACE-III y podrá visualizar y exportar los resultados.
4. **Investigadores:** Profesionales de la salud que se encargarán de realizar sus estudios a partir de la información recogida mediante la aplicación.
5. **Programadores futuros:** formado por los posibles desarrolladores que en un futuro puedan reutilizar o complementar la aplicación añadiendo más módulos o extendiéndola a plataformas web.

Resulta importante destacar que para el éxito del proyecto este debe convencer sobre todo a los evaluadores, ya que serán estos los que realicen su uso más intenso y los que tienen la capacidad de decisión para usar esta herramienta o cualquier otra.

3.3. Catálogo de requisitos del sistema

Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los desarrolladores diseñar un sistema que satisfaga estos requisitos, y que permita realizar las pruebas que demuestren si el sistema satisface, o no, los requisitos. Todo requisito aquí especificado describirá los comportamientos externos del sistema, perceptibles por parte de los usuarios, operadores y otros sistemas.

3.3.1. Objetivos y alcance del sistema

El sistema tiene como objetivo simplificar y agilizar el proceso de evaluación y seguimiento de pacientes es estudios de deterioro cognitivo.

La aplicación consta de cuatro módulos que corresponden a los cinco bloques funcionales excepto el bloque de Exportación que quedará distribuido en los distintos módulos:

- **Gestión de Encuestas.** La aplicación contendrá un creador de encuestas donde se podrá indicar el nombre, la descripción y las preguntas que contiene, también se podrá ver una vista previa de la encuesta final. Las encuestas dispondrán de las cuatro funciones básicas CRUD. Podrán exportarse listados de encuestas.

- **Gestión de Pacientes.** El evaluador podrá gestionar los datos de los pacientes que serán evaluados. Los pacientes también dispondrán de las cuatro funciones básicas CRUD. Podrán exportarse listados de pacientes.

- **Evaluación.** Los pacientes podrán ser evaluados mediante las encuestas creadas o mediante la prueba estandarizada ACE-III. La evaluación de las encuestas la pueden llevar a cabo tanto el evaluador como el paciente, pero la evaluación del ACE-III siempre deberá realizarse por el evaluador que controlará la realización de la prueba. Otra función muy importante que realizará este módulo es la recogida de tiempos, en las encuestas se medirá el tiempo total que se tarda en resolverla y en la prueba ACE-III se mide el tiempo que se tarda en cada una de las preguntas individualmente.

- **Resultados.** Los evaluadores podrán consultar los resultados de todas las pruebas realizadas por los pacientes tanto si han sido realizadas por el evaluador como si la ha realizado el paciente. Los resultados podrán ser exportados a Excel de esta manera nos resultarán mucho más fácil de compartir y visualizar.

Por otra parte, tendremos que desarrollar un servicio web que apoye a la aplicación, esta tendrá las siguientes funciones:

- **Login.** El servidor será capaz de autentificar usuarios.

- **Sincronizar datos:** Almacenara y sincronizara con los dispositivos todos los datos generados en la aplicación (Pacientes, Encuestas y Resultados).

- **Enviar encuestas:** Se encargará de enviar a los pacientes las encestas que los evaluadores les asignen.

3.3.2.Requisitos funcionales

Código: F1	**Nombre:** Login	
Resumen: Autentificar al usuario contra el servidor y sincronizar sus datos si procede.		
Entradas	**Proceso**	**Salidas**
1. Inicio de la aplicación.	1. El usuario, paciente o evaluador, introduce su usuario y contraseña en los campos de texto. 2. Pulsa el botón entrar.	1. Si las credenciales introducidas pertenecen a un evaluador sincroniza todos los datos del sistema y entra al menú principal de evaluador. 2. Si las credenciales introducidas pertenecen a un paciente sincroniza sus datos y entra al menú principal de paciente. 3. Si las credenciales no son correctas muestra un mensaje de error.

Tabla 3.1: Funcionalidad 1 Login

Código: F2	**Nombre:** Alta paciente	
Resumen: Crear un nuevo paciente en el sistema.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Pacientes’ accedemos al módulo de pacientes.	1. Pulsamos el botón flotante ‘+’. 2. Introducimos los datos del nuevo paciente 3. Pulsamos el botón ‘Guardar’.	1. Muestra un mensaje indicando si todo está correcto o si hay algún error.

Tabla 3.2: Funcionalidad 2 Alta paciente

Código: F3	**Nombre:** Editar paciente	
Resumen: Modificar los datos de un paciente existente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Pacientes’ accedemos al módulo de pacientes.	1. En la lista pulsamos sobre el paciente deseado. 2. Modificamos los datos del paciente. 3. Pulsamos el botón ‘Guardar’.	1. Muestra un mensaje indicando si todo está correcto o si hay algún error.

Tabla 3.3: Funcionalidad 3 Editar paciente

Código: F4	**Nombre:** Eliminar paciente	
Resumen: Eliminar los datos de un paciente existente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Pacientes’ accedemos al módulo de pacientes.	1. En la lista mantenemos pulsado sobre el paciente deseado hasta que aparezca el menú flotante. 2. Seleccionamos la opción ‘Eliminar’	1. El paciente eliminado desaparecerá de la lista.

Tabla 3.4: Funcionalidad 4 Eliminar paciente

Código: F5	**Nombre:** Alta encuesta	
Resumen: Crear una nueva encuesta en el sistema.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Encuestas’ accedemos al módulo de encuestas.	1. Pulsamos el botón flotante ‘+’. 2. Introducimos los datos del nuevo paciente 3. En la pestaña ‘PREGUNTAS’ introducimos las preguntas que deseamos. 3. En la pestaña ‘DATOS pulsamos el botón ‘Guardar’.	1. Muestra un mensaje indicando si todo está correcto o si hay algún error.

Tabla 3.5: Funcionalidad 5 Alta encuesta

Código: F6	**Nombre:** Editar encuesta	
Resumen: Modificar los datos de una encuesta existente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Encuestas’ accedemos al módulo de encuestas.	1. En la lista pulsamos sobre la encuesta deseada. 2. Modificamos los datos de la encuesta. 3. En la pestaña ‘PREGUNTAS’ modificamos las preguntas de la encuesta. 3. Pulsamos el botón ‘Guardar’.	1. Muestra un mensaje indicando si todo está correcto o si hay algún error.

Tabla 3.6: Funcionalidad 6 Editar encuesta

Código: F7	**Nombre:** Eliminar encuesta	
Resumen: Eliminar los datos de una encuesta existente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Encuestas’ accedemos al módulo de encuestas.	1. En la lista mantenemos pulsado sobre la encuesta deseada hasta que aparezca el menú flotante. 2. Seleccionamos la opción ‘Eliminar’	1. La encuesta eliminada desaparecerá de la lista.

Tabla 3.7: Funcionalidad 7 Eliminar encuesta

Código: F8	**Nombre:** Evaluar encuesta paciente	
Resumen: Realizamos la evaluación de una de las encuestas sobre un paciente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Evaluar’ accedemos al módulo de evaluación.	1. En la lista seleccionamos el paciente deseado. 2. Seleccionamos ‘Encuesta’. 3. En la lista pulsamos sobre la encuesta deseada. 4. Rellenamos los datos de la encuesta. 5. Pulsamos el botón ‘Finalizar’	1. Se cerrará la ventana de la encuesta y se guardarán los resultados. 2. Junto al resultado se guardará la estructura de la encuesta, así si la encuesta cambia o se elimina no nos afectará en nada y podremos consultar las respuestas igualmente.

Tabla 3.8: Funcionalidad 8 Evaluar encuesta paciente

Código: F9	**Nombre:** Evaluar ACE-III paciente	
Resumen: Realizamos la evaluación del test ACE-III sobre un paciente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Evaluar’ accedemos al módulo de evaluación.	1. En la lista seleccionamos el paciente deseado. 2. Seleccionamos ‘ACE’. 3. Evaluamos tal y como está descrito en pantalla al paciente y pulsamos el botón ‘Siguiente’ 4. Repetimos hasta haber finalizado todas las secciones.	1. Se cerrará la ventana de la encuesta y se guardarán los resultados. 2. Se mostrarán los resultados de la evaluación actual.

Tabla 3.9: Funcionalidad 9 Evaluar ACE-III paciente

Código: F10	**Nombre:** Enviar encuesta paciente	
Resumen: Enviamos una de las encuestas a un paciente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Evaluar’ accedemos al módulo de evaluación.	1. En la lista seleccionamos el paciente deseado. 2. Seleccionamos ‘Enviar Encuesta’. 3. En la lista pulsamos sobre la encuesta deseada.	1. Se cerrará la ventana actual y se enviará la encuesta.

Tabla 3.10: Funcionalidad 10 Enviar encuesta paciente

Código: F11	**Nombre:** Resultados paciente	
Resumen: Consultamos los resultados de las evaluaciones de un paciente.		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Resultados’ accedemos al módulo de resultados.	1. En la lista seleccionamos el paciente deseado. 2. Podemos cambiar entre las dos pestañas (ACE y ENCUESTAS) para ver las distintas evaluaciones. 3.Haciendo clic en uno de los resultados podemos ver sus detalles.	1. Listado de los resultados del ACE y de las encuestas. 2. Si hemos pulsado sobre una evaluación ACE veremos los detalles. 3. Si hemos pulsado sobre una encuesta podremos verla tal y como se contestó.

Tabla 3.11: Funcionalidad 11 Resultados paciente

Código: F12	**Nombre:** Exportar Resultados	
Resumen: Exportar los resultados ACE o de encuestas a un documento Excel		
Entradas	**Proceso**	**Salidas**
1. Pulsando el botón ‘Resultados’ accedemos al módulo de resultados.	1. En la lista seleccionamos el paciente deseado. 2. Pulsamos el botón de opciones situado a la derecha de la barra superior. 3. Seleccionamos una de las opciones en función de que queramos exportar.	1. Se abrirá el menú de Android donde podremos elegir que hacer con el documento generado.

Tabla 3.12: Funcionalidad 12 Exportar Resultados

Código: F13	**Nombre:** Resolver Encuesta Paciente	
Resumen:		
Entradas	**Proceso**	**Salidas**
1. Iniciar la aplicación identificándose como paciente.	1. En la lista pulsamos sobre la encuesta deseada. 2. Rellenamos los datos de la encuesta. 3. Pulsamos el botón 'Finalizar'.	1. La encuesta que acabamos de resolver desaparecerá de la lista.

Tabla 3.13: Funcionalidad 13 Resolver Encuesta Paciente

3.3.3. Suposiciones y dependencias

1. **Suposiciones**

   a) Se asume que los requisitos en este documento son estables.

   b) Se asume que la prueba estándar ACE-III no va a cambiar por el momento.

   c) Se asume que las encuestas pueden cambiar en cualquier momento al largo de su vida, incluso si ya existen evaluaciones de esta.

   d) Se asume que los pacientes que van a ser estudiados disponen de smartphones o tabletas Android.

   e) Se asume que los centros de salud y hospitales que utilizarán la aplicación tienen acceso a dispositivos Android.

2. **Dependencias**

   a) El sistema dependerá de la librería Apache POI [Apa] que utilizaremos para realizar la exportación a Excel.

3.3.4. Requisitos de usuario y tecnológicos

1. **Requisitos de usuario**

   a) Como parte de la aplicación va a ser usada por los pacientes, que se les espera poco habituados a los smartphones, las interfaces deben ser muy fáciles e intuitivas, aunque casi siempre será utilizada por personal cualificado.

2. **Requisitos tecnológicos**

   a) El sistema se desarrolla como una aplicación Android y un servicio web.

   b) El sistema operativo sobre el cual se instalará la aplicación deberá ser Android 5.0 o superior.

   c) El servicio deberá desplegarse sobre un servidor apache que cuente con PHP 5 y MySql.

3.3.5. Requisitos de interfaces

a) **Interfaces de usuario.** La interfaz de usuario debe ser sencilla e intuitiva, y cumplir con los patrones establecidos por Google ‘Material Design’[Mat].

3.4. Conclusión

En este apartado del documento hemos analizado los usuarios participantes del sistema, los objetivos y el alcance, los requisitos funcionales y no funcionales, las suposiciones y dependencias. A través de los requisitos funcionales se ha hecho un acercamiento mucho más certero a las diversas

funcionalidades que el sistema debe ofrecer y las suposiciones que debemos tener en cuenta.

Una vez terminado este capítulo, nos encontramos con un sistema totalmente detallado y perfilado, lo cual nos permite dar paso al siguiente apartado, en el que trataremos de realizar un acercamiento más formal al funcionamiento del sistema que nos facilite el desarrollo del mismo.

4. Diseño del sistema

4.1. Introducción

En este capítulo vamos a hacer uso de la especificación de requisitos para realizar de una manera más precisa un acercamiento al funcionamiento del sistema y a su diseño. Para ello, en primer lugar, mediante esquemas conceptuales vamos a representar las funcionalidades del sistema. A continuación, presentaremos el prototipo de las distintas pantallas de la aplicación, con lo que conseguiremos hacernos una idea de cuál será el aspecto final de la interfaz de usuario. Este diseño no tiene porque ser definitivo, puesto que a la hora de implementar la aplicación pueden surgir problemas que obliguen a cambiarlo. Finalmente, mediante el uso de diagramas vamos a tratar de reflejar de manera formal el proceso de las funcionalidades del sistema reflejadas en el capítulo anterior.

4.2. Especificación conceptual

El propósito del diseño conceptual es realizar una aproximación más visual al funcionamiento del sistema, tratando de clarificar lo expuesto en el punto 3.3.3. de este documento. Así pues, mediante el uso de esquemas conceptuales se tratará de representar aquellas funcionalidades cuyo proceso pueda resultar más complicado de entender. Los diseños de la GUI expuestos en estos esquemas son aproximaciones hechas con el objetivo de representar la idea de la manera más exacta posible, pero no deben tomarse como diseños definitivos. En este caso, se ha decidido trabajar con los esquemas conceptuales correspondientes a las actividades principales de la aplicación. El resto de funcionalidades resulta lo suficientemente sencillo como para no ser necesaria su representación mediante el diseño conceptual.

En primer lugar, vamos a observar en un esquema gráfico (Figura 4.1) las distintas partes del sistema y como interactuarán entre ellas.

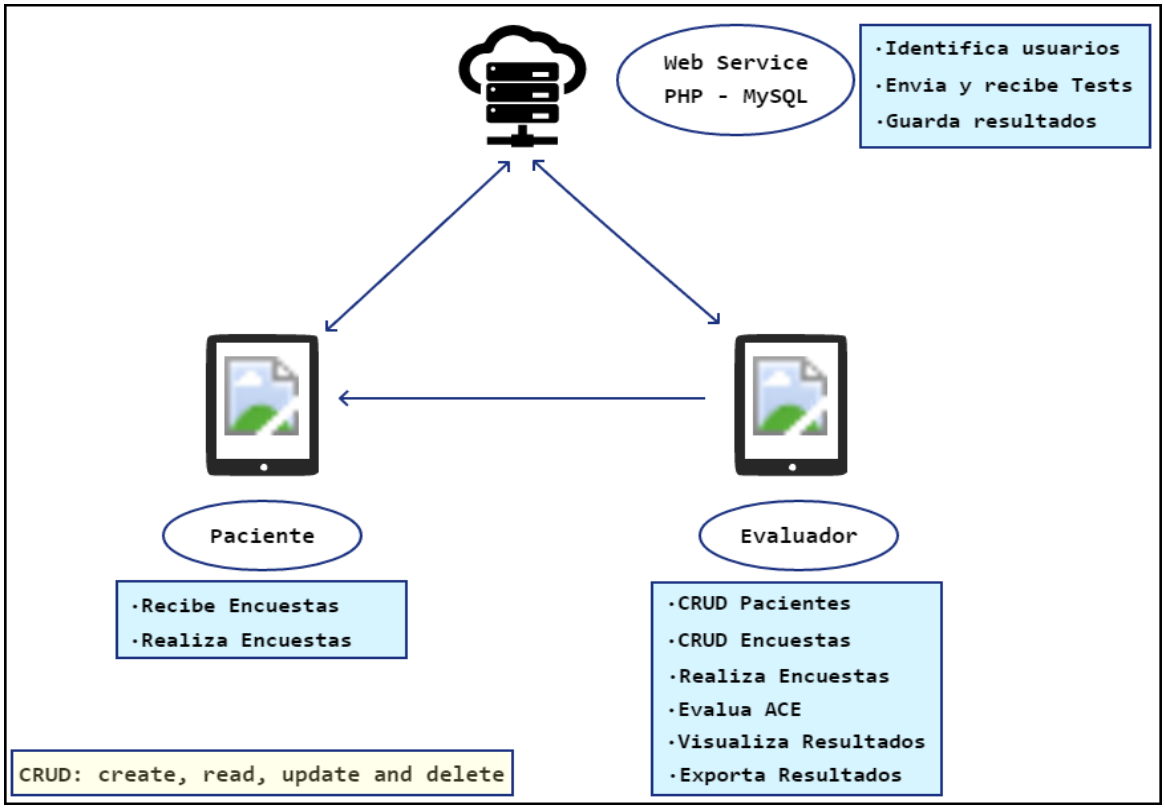


Figura 4.1 Resumen del Sistema

Como hemos podido observar en la figura 4.1 el sistema consistirá en un servicio web programado en PHP que contendrá una base de datos MySQL, este se comunicará con la aplicación Android que tendrá dos modos de funcionamiento, el modo Evaluador y el modo Paciente.

En la figura 4.2 podemos observar el proceso de creación de una encuesta. En el módulo de encuestas pulsamos el botón flotante '+' para crear una nueva, esto nos abrirá la pantalla de creación de encuestas, en la primera pestaña 'Datos' introducimos el nombre y una descripción opcional, en la segunda pestaña 'Preguntas' introducimos las preguntas que deseamos. Para guardar nuestros resultados nos situamos en la pestaña 'Datos' pulsamos el botón 'Guardar'.

Para agregar una pregunta nueva pulsaremos botón flotante '+' y en la pantalla que aparecerá seleccionamos un tipo de pregunta, esto nos conducirá a un formulario donde tendremos que introducir diferentes datos en función del tipo escogido.

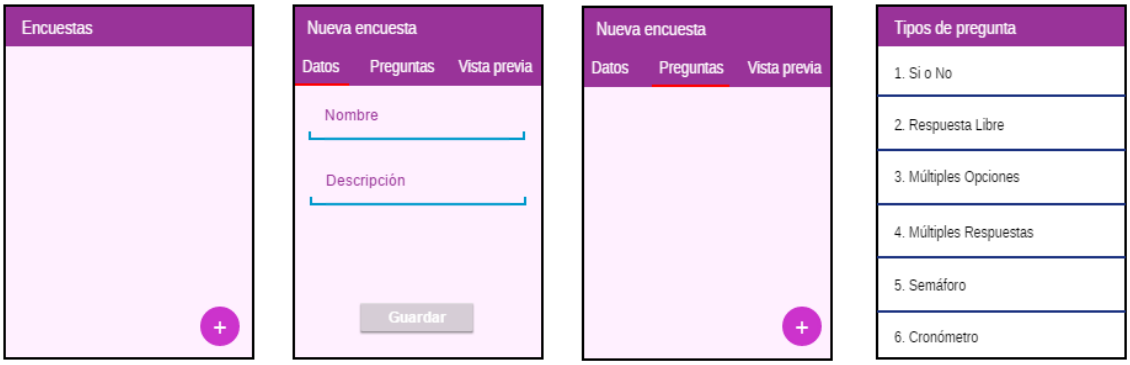


Figura 4.2 Creación de encuestas

En la figura 4.3 podemos observar cómo se consultan los resultados de la prueba ACE-III. En el módulo de resultados seleccionamos al paciente deseado, se abrirá la pantalla de resultados de este paciente, en la pestaña 'ACE' vemos todas las evaluaciones de este tipo, si nos interesa una en concreto podemos ver sus detalles pulsándola.

Figura 4.3 Consulta de resultados ACE

4.3. Especificación formal

4.3.1. Presentación

A continuación, presentaremos unos diseños con los que trataremos de hacer un acercamiento formal a la interfaz que presentará la aplicación. Vamos a presentar una versión simplificada del diseño que implementaremos para conseguir tener una mayor flexibilidad en la fase de desarrollo, reflejando solamente los elementos más importantes.

Debido a la edad de los pacientes, su parte de la aplicación será muy simple y con pocos elementos, mientras que en la parte del evaluador este tendrá de una mayor cantidad de elementos en la interfaz para que pueda realizar su trabajo satisfactoriamente.

En cuanto a la orientación, se permitirá el uso de la aplicación tanto en horizontal como en vertical. Esta decisión se ha tomado pensando en la comodidad de los usuarios, ya que algunos pueden tener preferencias distintas en este aspecto.

Para empezar, podemos ver en la figura 4.4 la pantalla de login de la aplicación junto a los dos menús principales, el de pacientes a la derecha y el de evaluadores a la izquierda. Como podemos observar el menú de pacientes es mucho más simple, esto se debe a que la única función que tienen disponible es la resolución de las encuestas que les aparecen listadas. Por otro lado, el menú de evaluadores presenta cuatro botones correspondientes a los cuatro módulos de la aplicación.

Figura 4.4 Pantallas principales

En la figura 4.5 podemos observar el módulo de pacientes con todas sus interfaces. Como vemos en la primera pantalla el elemento principal es una lista con los nombres de todos los pacientes registrados en el sistema, también aparecen un buscador de pacientes en la parte superior y un botón de agregar en la inferior. Tanto si pulsamos sobre un paciente existente para editarlo, como si pulsamos en el botón flotante para crear uno nuevo, esta acción nos conducirá a la pantalla de datos del paciente donde podremos manipular su información. Finalmente, en la pantalla de datos aparecerán diversos campos de texto que debemos rellenar y un botón de guardado.

Figura 4.5 Módulo pacientes

En la figura 4.6 observamos el módulo de encuestas. La primera pantalla es muy parecida a la de pacientes y su funcionalidad es idéntica, la única diferencia de que en este caso se ha decidido prescindir del buscador. La pantalla de datos de encuesta es, posiblemente, la pantalla con mayor complejidad de la aplicación. Esta pantalla presenta tres pestañas, la primera (Datos) contiene la información básica de la encuesta, la segunda (Preguntas) contiene una lista ordenada de las preguntas que contiene la encuesta y la tercera (Vista previa) ofrece una vista de la encuesta tal y como se presentará en el momento de la evaluación. Cabe recordar que el funcionamiento de este módulo está explicado en el apartado 4.2. En la Figura 4.7 podemos ver un ejemplo de encuesta con los seis tipos posibles de pregunta.

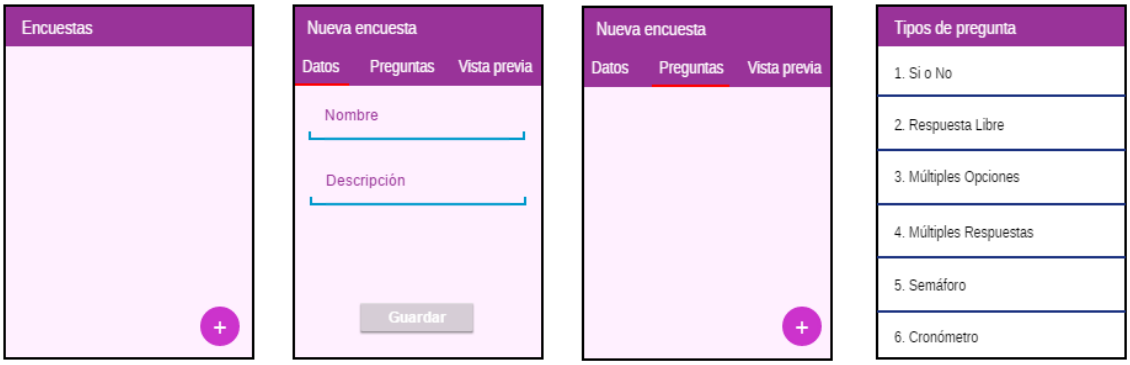


Figura 4.6 Módulo encuestas

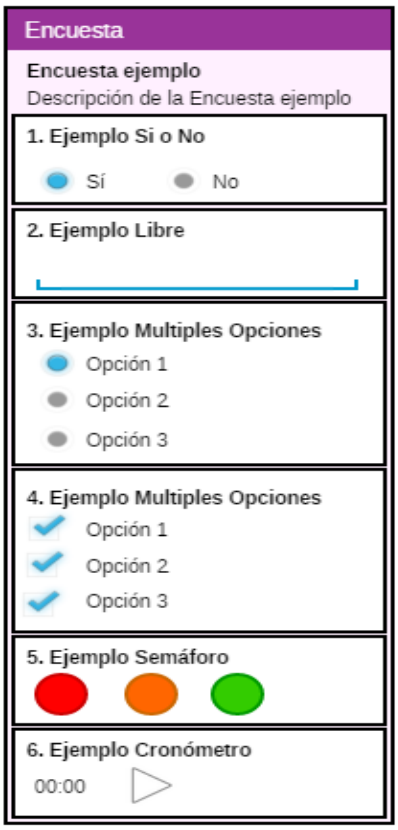


Figura 4.7 Encuesta ejemplo

En la figura 4.8 vemos las interfaces correspondientes al módulo de evaluación. La primera pantalla es idéntica a la primera del módulo de pacientes con la diferencia de que no existe el botón de agregar en este caso. Una vez seleccionado un paciente aparecerá su pantalla de evaluación, esta es una interfaz muy simple que nos muestra algunos datos relevantes del paciente y nos presenta tres botones. El primer botón nos conducirá a la evaluación del ACE, el segundo nos presentará una lista de encuestas donde elegiremos cual evaluaremos y el tercero nos presentará la misma lista que en el caso anterior, pero en vez de evaluar la encuesta nosotros esta será enviada al paciente.

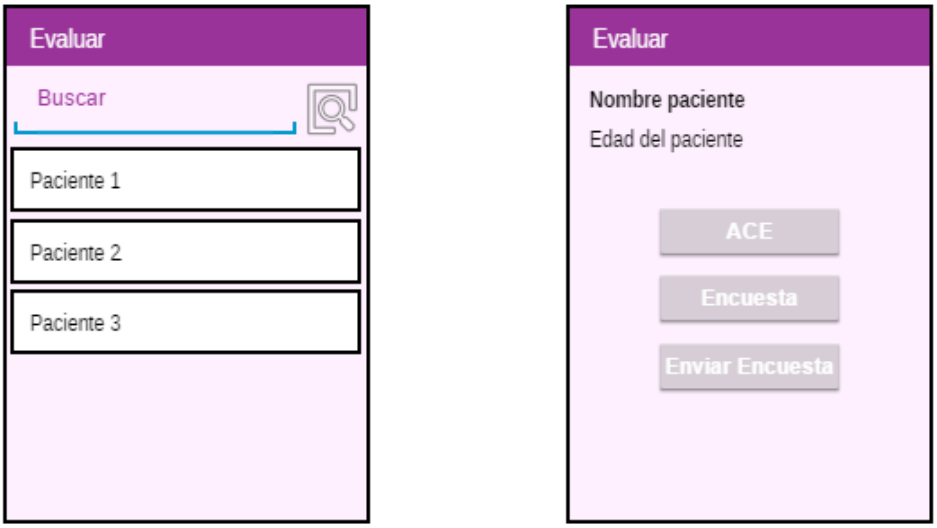


Figura 4.8 Módulo evaluación

Por último, en la figura 4.9 observamos el módulo de resultados. El igual que en el módulo de evaluación en la primera pantalla nos aparece un listado de pacientes. Al hacer clic en un paciente aparece una pantalla con dos pestañas, en la primera observamos los resultados de todos los ACE del paciente y en la segunda todas las encuestas que este ha resuelto. Si pulsamos cualquiera de los resultados podremos ver los detalles de este.

Figura 4.9 Módulo resultados

4.3.2. Servicio Web

Ahora que tenemos bien definida la aplicación es el momento de definir los procesos que debe realizar el servicio. Este será esencial para que nuestra aplicación sea multidispositivo ya que nos permitirá compartir datos entre los distintitos usuarios. Este servicio también podrá ser utilizado si en un futuro exportamos la aplicación a otra plataforma.

Para realizar el diseño de las comunicaciones seguiremos el estilo REST [Rest] que hace uso de cuatro operaciones básica **POST**, **GET**, **PUT** y **DELETE.**

a) Login: El servicio recibirá un usuario y contraseña, comprobará que el par existe en la base de datos y contestara si es correcto o no.

b) Get pacientes: El servicio enviará al dispositivo una lista con todos los pacientes.

c) Get paciente id: El servicio recibirá un Id de paciente y si existe en la BBDD devolverá todos sus datos.

d) Put pacientes: El servicio recibirá una lista de pacientes e insertará todos los que no existan en la BBDD.

e) Update pacientes: El servicio recibirá una lista de pacientes y actualizará todos aquellos que existan en la BBDD.

f) Delete paciente: dado un id de paciente el servicio lo eliminara de la BBDD

g) Get encuestas: El servicio enviará al dispositivo una lista con todas las encuestas.

h) Get encuesta id: El servicio recibirá un Id de encuesta y si existe en la BBDD devolverá todos sus datos.

i) Put encuestas: El servicio recibirá una lista de encuestas e insertará todas las que no existan en la BBDD.

j) Update pacientes: El servicio recibirá una lista de encuestas y actualizará todas aquellas que existan en la BBDD.

k) Delete encuesta: dado un id de encuesta el servicio la eliminara de la BBDD

l) Get resultados encuesta: El servicio enviara una lista de resultados de encuesta.

m) **Get resultados encuesta paciente:** Dado un id de paciente el servicio enviara una lista de sus resultados de encuesta.

n) **Put resultados encuesta:** El servicio recibirá una lista de resultados de encuestas y los insertará en la BBDD.

o) **Get resultados ACE:** El servicio enviara una lista de resultados de pruebas ACE.

p) **Get resultados ACE paciente:** Dado un id de paciente el servicio enviara una lista de sus resultados del ACE.

q) **Put resultados ACE:** El servicio recibirá una lista de resultados de ACE y los insertará en la BBDD.

r) **Get encuestas pendientes:** El servicio enviará al dispositivo una lista con todas las encuestas pendientes.

s) **Get encuestas pendientes id:** El servicio recibirá un Id de un paciente y si existe en la BBDD devolverá todas sus encuestas pendientes.

t) **Put encuestas pendientes:** El servicio recibirá una lista de encuestas pendientes e insertará todas las que no existan en la BBDD.

4.4. Conclusión

A lo largo de este capítulo se ha realizado una de las fases más importantes en el proceso de un proyecto software, el diseño del sistema. En primer lugar, se han presentado los esquemas conceptuales que permiten entender el funcionamiento de la aplicación, en segundo lugar, se ha mostrado un prototipo de las interfaces que permite realizar todas las funcionalidades y por último se ha decidido cuáles serán las funciones que nuestro servicio web debe realizar.

Tras la especificación de requisitos y el diseño del sistema, tenemos ahora el sistema a desarrollar totalmente definido y perfilado. Esta tarea resulta totalmente indispensable de cara a afrontar la fase de desarrollo del sistema y nos ahorrará mucho tiempo a la hora de implementar el código de la aplicación.

5. Implementación

5.1. Introducción

En este capítulo trataremos de presentar el resultado de la aplicación que hemos desarrollado a partir de las definiciones previas. Para ello, vamos a analizar la aplicación en dos aspectos, el diseño final de la interfaz y la implementación de las funcionalidades. En cuanto a la interfaz, trataremos de observar cual es la apariencia definitiva de la aplicación y compararla con aquella que previamente diseñamos. En cuanto a las funcionalidades, analizaremos el código más importante que nos ha permitido llevar a cabo la implementación de los procesos que reflejamos en el capítulo anterior.

5.2. Presentación

En este punto vamos a mostrar cual es el estado actual de la interfaz mediante capturas de pantalla sobre la aplicación real. De esta manera podremos compararla con el diseño original, lo cual nos permitirá mejorar en este aspecto de cara a proyectos futuros. Al igual que en el punto 4.3.1. presentaremos las interfaces agrupadas por módulos.

Primero veremos, en la figura 5.1, las interfaces correspondientes al inicio de la aplicación, el login y los menús de evaluador y paciente.

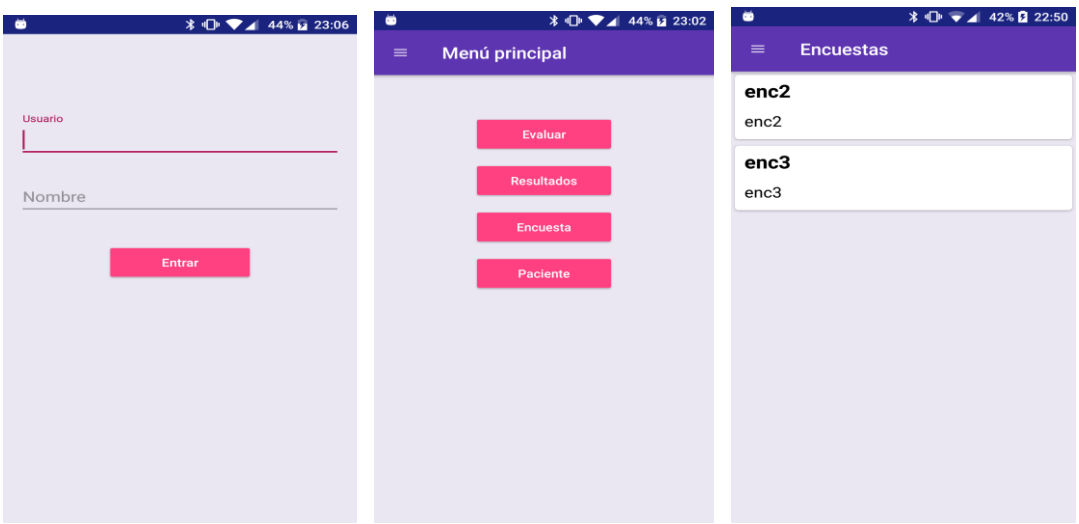


Figura 5.1 Pantallas principales

En la figura 5.2 podemos ver las interfaces correspondientes al módulo de pacientes.

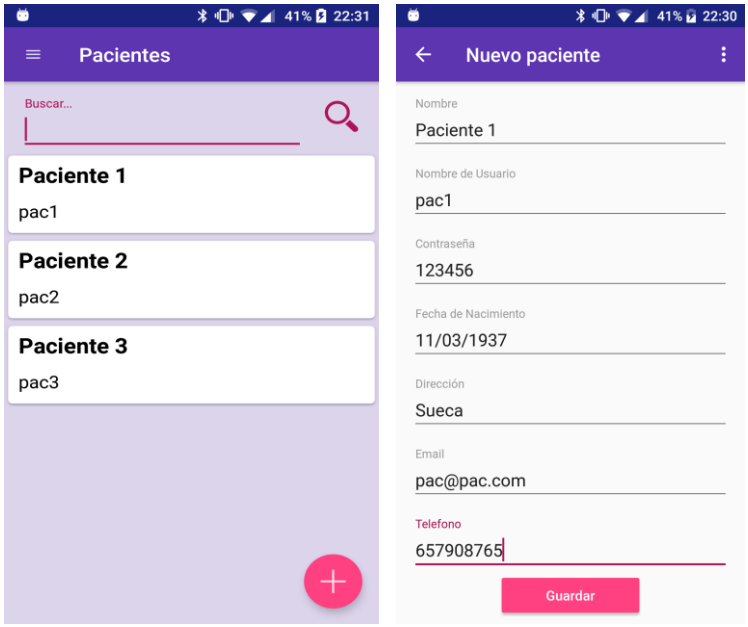


Figura 5.2 Módulo pacientes

En las figuras 5.3 y 5.4 podemos ver las interfaces correspondientes al módulo de encuestas.

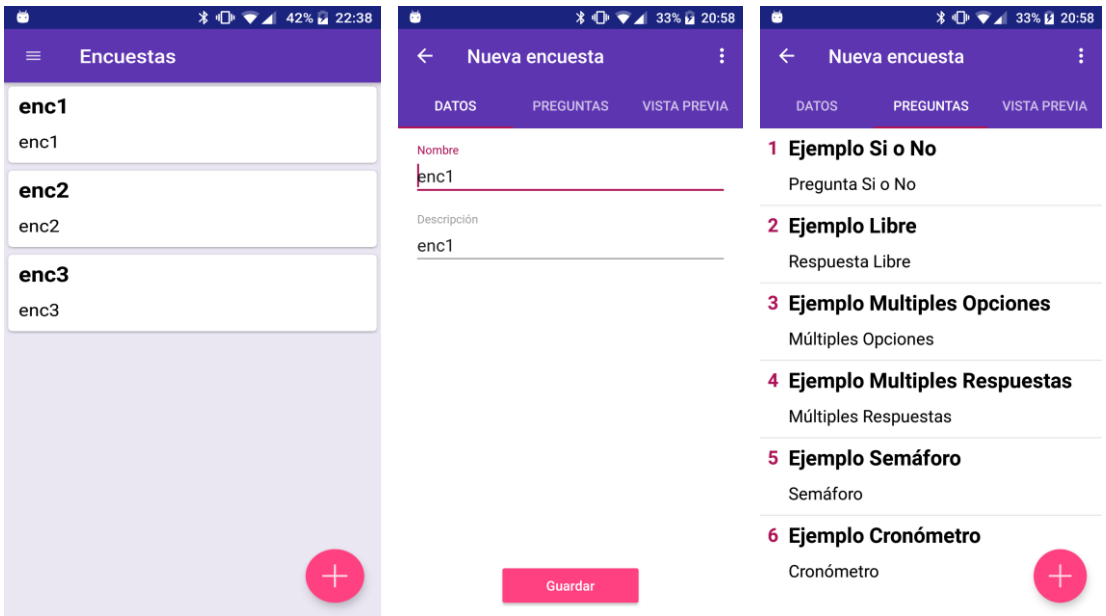


Figura 5.3 Módulo encuestas

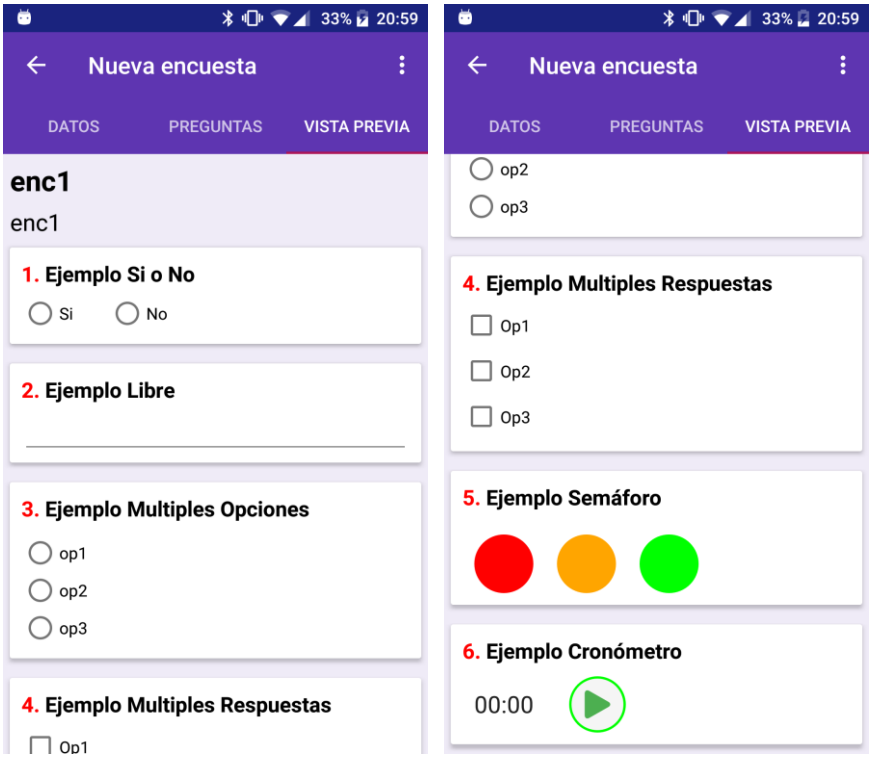


Figura 5.4 Ejemplo encuesta

En la figura 5.5 podemos ver las interfaces correspondientes al módulo de evaluación.

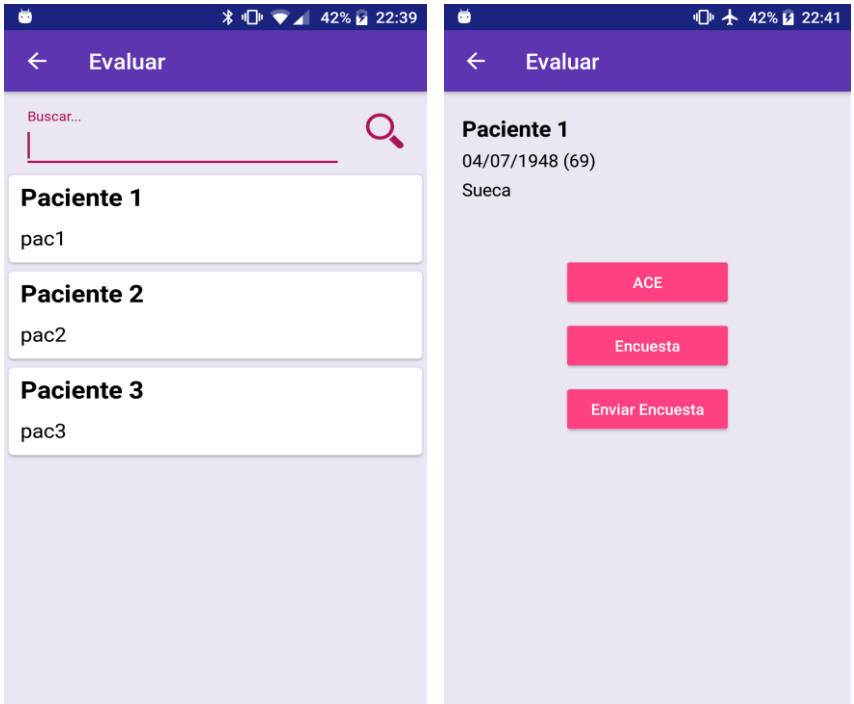


Figura 5.5 Módulo evaluación

Por último, en las figuras 5.6 y 5.7 podemos ver las interfaces correspondientes al módulo de resultados. En este módulo es importante destacar que en la barra superior a la derecha aparece un botón con tres puntos, si lo pulsamos nos aparecerá un menú flotante con dos opciones, 'ListadoACE' y 'ListadoEncuestas', al seleccionar cualquiera de las dos la aplicación exportará los datos a PDF y nos ofrecerá la opción de compartir el documento generado mediante alguna de las aplicaciones instaladas en el dispositivo.

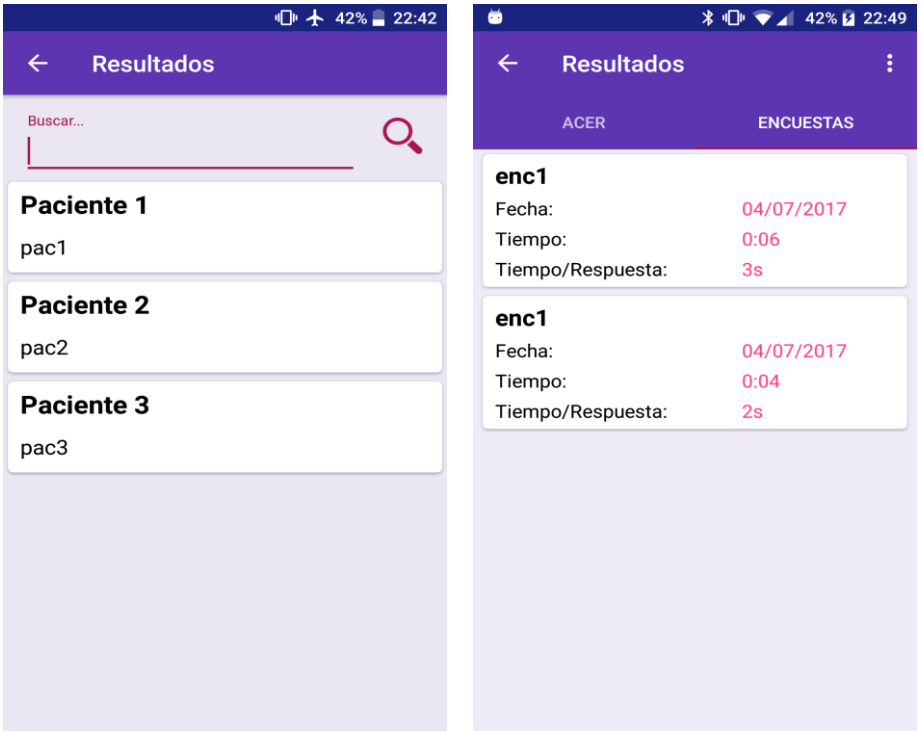


Figura 5.6 Módulo resultados (Encuestas)

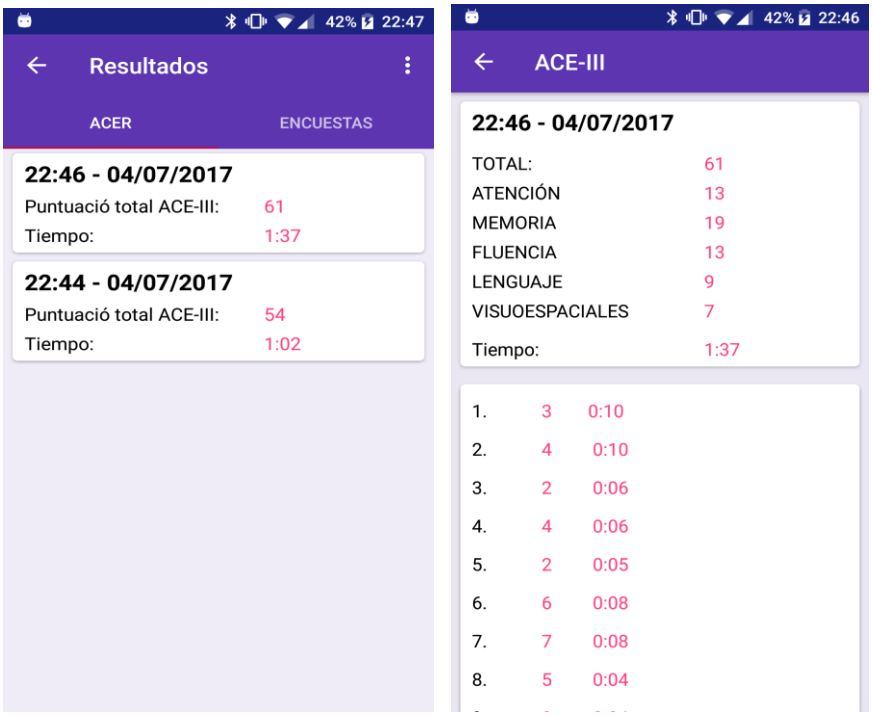


Figura 5.7 Módulo resultados (ACE)

5.3. Negocio

En esta sección vamos a exponer las partes del código que puedan resultar más significativas o interesantes para entender de qué manera se ha llevado a cabo la implementación de los procesos de las distintas actividades. Para esta implementación se ha partido de los diseños vistos en el capítulo anterior y se ha llevado a cabo su desarrollo en lenguaje Java para la aplicación y PHP para el servicio. Además, adjuntar el código necesario para entender el funcionamiento de las actividades, también vamos a comentar este en la medida de lo posible para que su interpretación se pueda realizar de la manera más sencilla posible.

5.3.1. Base de Datos

Antes de adentrarnos en el código de la base de datos conviene tener clara su estructura, la cual podemos observar en la figura 5.8. Como podemos ver los elementos centrales en la BBDD son las tablas Encuesta y Paciente, pues todas las acciones que realicemos estarán relacionadas con ellos. También observamos que diversas tablas tienen un campo llamado *estado,* este campo

nos será de utilidad en la sincronización pues nos marcará si el registro es nuevo, si se ha modificado o si ya está sincronizado.

Cuando observemos el esquema tenemos que tener en cuenta que los campos que aparecen en son las llaves primarias de cada tabla.

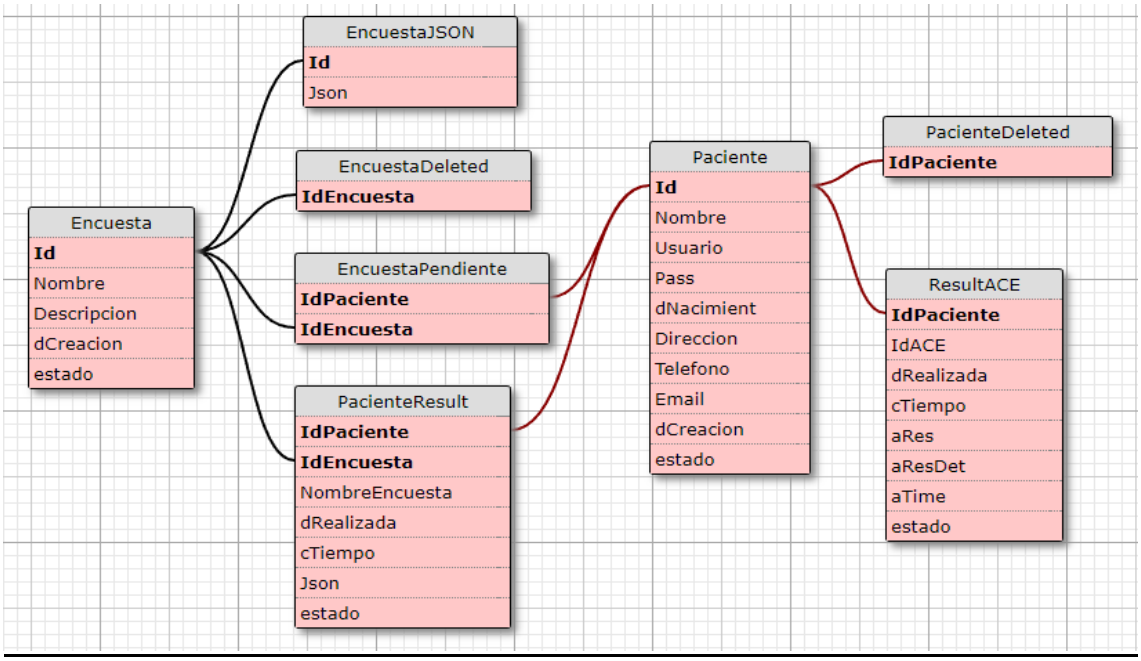


Figura 5.8 Esquema BBDD

Como hemos visto en el esquema existen ocho tablas en nuestra base de datos, vamos ahora a explicar qué función cumplen cada una.

- **Encuesta y EncuestaJSON:** Estas dos tablas funcionan como una sola, se encargan de almacenar toda la información de las Encuestas definidas en la aplicación. El motivo de la existencia de la tabla EncuestaJSON es aligerar la tabla encuesta, de esta manera optimizamos el proceso de búsqueda y listado de tablas, ya que el peso del JSON almacenado supera significativamente al resto de los datos.

- **EncuestaDeleted:** Aquí almacenamos el Id de las encuestas que hemos eliminado localmente, cuando realicemos la sincronización con el servicio le indicaremos a este que elimine los registros que aquí le indicamos.

- **EncuestaPendiente:** Esta tabla nos servirá para indicar las encuestas que deben resolver los pacientes en sus dispositivos.

- **Paciente:** Almacenara los pacientes registrados en el sistema.

- **PacienteResult:** Contiene los resultados de las encuestas que han resuelto los pacientes. En este caso hemos decidido no separa el JSON en otra tabla, ya que la optimización de esta tabla no resulta tan importante.
- **PacienteDeleted:** Al igual que EncuestaDeleted indica los registros que el servicio ha de eliminar de su BBDD.
- **ResultACE:** Aquí encontramos los resultados de las evaluaciones ACE.

A continuación vamos a ver el código correspondiente a la creación de las bases de datos tanto de la aplicación Android como del servicio, ya que esto nos resultará fundamental para entender el conjunto del sistema.

Android:

Para poder manejar fácilmente el acceso a datos se han centralizado todas las funciones que hacen uso de la BBDD en una misma clase. Esta clase que llamaremos *DbHelper* heredará las propiedades y métodos de la clase nativa de Android *SQLiteOpenHelper*.

```
public class DbHelper extends SQLiteOpenHelper {
```

El método expuesto *onCreate* se lanzará cuando se instancie la clase *DbHelper,* este se encargará de crear las tablas necesarias si no existen aún.

```
public void onCreate(SQLiteDatabase db) {
    //db.execSQL(SQL_CREATE_ENTRIES);
    db.execSQL("CREATE TABLE Encuesta (Id TEXT PRIMARY KEY, Nombre TEXT,
Descripcion TEXT, dCreacion TEXT, estado TEXT)");

    db.execSQL("CREATE TABLE EncuestaJSON (Id TEXT PRIMARY KEY, Json BLOB)");

    db.execSQL("CREATE TABLE EncuestaDeleted (IdEncuesta TEXT)");

    db.execSQL("CREATE TABLE EncuestaPendiente (IdPaciente TEXT, IdEncuesta TEXT,
estado TEXT)");

    db.execSQL("CREATE TABLE Paciente (Id TEXT PRIMARY KEY, Nombre TEXT,
Usuario TEXT, Pass TEXT, dNacimiento TEXT, Direccion TEXT, Telefono TEXT, Email
TEXT, dCreacion TEXT, estado TEXT, UNIQUE(Usuario))");

    db.execSQL("CREATE TABLE PacienteResult (IdPaciente TEXT, IdEncuesta,
NombreEncuesta TEXT, dRealizada TEXT, cTiempo TEXT, Json TEXT, estado TEXT)");
```

```
    db.execSQL("CREATE TABLE PacienteDeleted (IdPaciente TEXT)");

    db.execSQL("CREATE TABLE ResultACE (IdPaciente TEXT, IdACE TEXT PRIMARY
KEY, dRealizada TEXT, cTiempo TEXT, aRes TEXT,  aResDet TEXT, aTime TEXT, estado
TEXT)");
}
```

PHP:

El servicio contará con una clase llamada *ConexionBD* que se encargará de la conexión con la BBDD MySql. Para asegurarnos de que solo existe una conexión entre el servicio y la base de datos emplearemos un patrón Singleton [Sing].

```
private static $db = null;

final private function __construct()
{
    try {
        // Crear nueva conexión PDO
        self::obtenerBD();
    } catch (PDOException $e) {
        // Manejo de excepciones
    }
}

/** Retorna en la única instancia de la clase
 * @return ConexionBD|null
 */
public static function obtenerInstancia()
{
    if (self::$db === null) {
        self::$db = new self();
    }
    self::$db->IniciaBD();
    return self::$db;
}
```

Esta clase se conectará mediante el método *obtenerBD*.

```
public function obtenerBD()
{
    if (self::$pdo == null) {
        self::$pdo = new PDO(
            'mysql:dbname=' . BASE_DE_DATOS .
            ';host=' . NOMBRE_HOST . ";",
            USUARIO,
            CONTRASENA,
            array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8")
        );
```

```
        // Habilitar excepciones
        self::$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }

    return self::$pdo;
}
```

Siempre que se realice una llamada al servicio este, a través de su archivo principal (*index.php),* se encargará de la creación de las tablas necesarias utilizando método *IniciaBD.* Como podremos observar en el código para realizar la creación utilizara el comando 'CREATE TABLE IF NOT EXISTS' , de este modo si la tabla ya existe no hará nada.

```
public static function IniciaBD()
{
    if(!isset(self::$pdo)){

        $var = new ConexionBD();
        $var->obtenerBD();
    }

    $comando = "CREATE TABLE IF NOT EXISTS encuesta (
                Id varchar(100) NOT NULL,
                Nombre varchar(50) NOT NULL,
                Descripcion varchar(1024),
                dCreacion varchar(20) NOT NULL,
                Json TEXT,
                PRIMARY KEY (Id))";
    $sentencia = self::$pdo->prepare($comando);
    $sentencia->execute();

    $comando = "CREATE TABLE IF NOT EXISTS encuestapendiente (
                IdPaciente varchar(100) NOT NULL,
                IdEncuesta varchar(100) NOT NULL)";
    $sentencia = self::$pdo->prepare($comando);
    $sentencia->execute();

    $comando = "CREATE TABLE IF NOT EXISTS paciente (
                Id varchar(100) NOT NULL,
                Nombre varchar(100)UNIQUE,
                Usuario varchar(100) NOT NULL UNIQUE,
                Pass varchar(100) NOT NULL,
                dNacimiento varchar(20),
                Direccion varchar(100),
                Telefono varchar(20),
                Email varchar(100),
                dCreacion varchar(20),
                PRIMARY KEY (Id))";
    $sentencia = self::$pdo->prepare($comando);
    $sentencia->execute();

    $comando = "CREATE TABLE IF NOT EXISTS pacienteresult (
                IdPaciente varchar(100) NOT NULL,
                IdEncuesta varchar(100) NOT NULL,
```

```
            NombreEncuesta varchar(100) NOT NULL,
            dRealizada varchar(100) NOT NULL,
            cTiempo varchar(20) NOT NULL,
            Json TEXT NOT NULL)";
    $sentencia = self::$pdo->prepare($comando);
    $sentencia->execute();

    $comando = "CREATE TABLE IF NOT EXISTS resultace (
            IdPaciente varchar(100) NOT NULL,
            IdACE varchar(100) NOT NULL,
            dRealizada varchar(20) NOT NULL,
            cTiempo varchar(20) NOT NULL,
            aRes varchar(200) NOT NULL,
            aResDet varchar(200) NOT NULL,
            aTime varchar(300) NOT NULL)";
    $sentencia = self::$pdo->prepare($comando);
    $sentencia->execute();

}
```

Como podemos observar en el código las tablas del servicio son las mismas que en la aplicación a excepción de tres que faltan.

- **EncuestaDeleted y PacienteDeleted:** En el servicio no son necesarias, ya que aquí eliminaremos directamente los registros.
- **EncuestaJSON:** Como el servicio solo se encarga de almacenar las encuestas no es necesario separar el JSON en una tabla distinta.

5.3.2. Sincronización

A continuación, expondremos las funciones más importantes en lo referente a la sincronización entre la aplicación y el servicio.

Para diseñar la comunicación del sistema se ha seguido el estilo REST [Rest] que hace uso de las cuatro operaciones básica **POST**, **GET**, **PUT** y **DELETE**.

Android:

Para lograr la comunicación desde la aplicación hemos creado la clase *RestCliente* que contendrá las llamadas de las cuatro operaciones. Esta clase heredara las propiedades de *AsyncTask* porque Android no nos permite realizar peticiones asíncronas, como son las peticiones web, desde su hilo principal.

```
public class RestCliente extends AsyncTask<String, Void, String> {


@Override
protected String doInBackground(String[] params) {

    switch (params[0]){
        case "get": return GET(params[1]);
        case "post": return POST(params[1],params[2]);
        case "put": return PUT(params[1],params[2]);
        case "delete": return DELETE(params[1]);
        default: return "Error Param";
    }
}
```

Hemos sobrescrito el método de *AsyncTask doInBackground* para que cuando lancemos la ejecución de la tarea este evalúe cuál de los cuatro métodos le hemos indicado y lance el proceso correspondiente.

Como los procesos de los cuatro métodos son muy similares vamos a mostrar solo el de GET. Lo más destacable es el uso de la clase *HttpURLConnection* que será la encargada de realizar las distintas operaciones de comunicación, así como el uso de las clases *InputStream* y *BufferedReader* Para obtener la respuesta del servicio.

```
@NonNull
private String GET(String cURL) {

    StringBuffer oBuffer = new StringBuffer("");
    try{
        // Envío la petición al servicio
        URL url = new URL(cURL);
        HttpURLConnection connection = (HttpURLConnection)url.openConnection();
        connection.setRequestProperty("Content-Type", "application/json; charset=utf-8");
        connection.setRequestMethod("GET");
        connection.connect();

        //Recojo la respuesta del servicio
        InputStream inputStream = connection.getInputStream();

        BufferedReader rd = new BufferedReader(new InputStreamReader(inputStream));

        String line;
        while ((line = rd.readLine()) != null) {
            oBuffer.append(line);
        }
        Log.i("Response Get",oBuffer.toString());
    }
    catch (IOException e) {
        // Writing exception to log
        e.printStackTrace();
    }
    return oBuffer.toString();
}
```

Aunque *RestCliente* se encargue de las peticiones esta no se encarga de organizar la sincronización, la clase *Sincronizador* será la encargada de esta tarea.

Sincronizador hará de intermediaria entre la base de datos local y *RestCliente,* de esta forma conseguimos que desde cualquier punto de la aplicación que deseemos sincronizar algo solo tendremos que llamar a un método de esta clase y olvidarnos del resto de detalles de la conexión. Para ilustrar como funciona vamos a mostrar dos métodos.

El método SubirDatos se encarga de subir todos los datos de la base de datos local del evaluador.

```
private void SubirDatos() {

    DbHelper oDb = new DbHelper(oContext);

    //---------- Pacientes ----------//
    ArrayList<Paciente> aPacN = oDb.getPacNuevos();
    ArrayList<Paciente> aPacM = oDb.getPacModificados();

    if (SubirPacNuevos(aPacN)) {
       oDb.setSincPac(aPacN);
    }

    if (SubirPacModificados(aPacM)) {
       oDb.setSincPac(aPacM);
    }

    //---------- Encuestas ----------//
    ArrayList<Encuesta> aEncN = oDb.getEncNuevas();
    ArrayList<Encuesta> aEncM = oDb.getEncModificadas();

    if (SubirEncNuevas(aEncN)) {
       oDb.setSincEnc(aEncN);
    }

    if (SubirEncModificadas(aEncM)) {
       oDb.setSincEnc(aEncM);
    }

    //---------- Encuestas Result ----------//
    String[][] aResN = oDb.getResultEncuestaNueva();
    if (aResN != null) {

       if (SubirResEncNuevas(aResN)) {
          oDb.setSincResEnc(aResN);
       }
    }

    //---------- ACE Result ----------//
    ACE[] aAceN = oDb.getAceNuevas();
    if (aAceN != null) {

       if (SubirResAceNuevas(aAceN)) {
```

```
                oDb.setSincAce(aAceN);
            }
        }

        //---------- Encuestas Pendientes ----------//
        String[][] aEncPen = oDb.getEncuestasPendientes();

        if (aEncPen != null) {
            SubirEncPendientes(aEncPen))
        }
    }
```

El método *SubirPacNuevos* subirá la lista de pacientes que el evaluador ha agregado desde la última sincronización.

```
@NonNull
private Boolean SubirPacNuevos(ArrayList<Paciente> aPacN) {

    if (aPacN.size() == 0) {
        return true;
    }

    try {
        JSONArray aJson = new JSONArray();

        for (int i = 0; i < aPacN.size(); i++) {
            JSONObject oJson = new JSONObject();
            oJson.put("Id", aPacN.get(i).getcId());
            oJson.put("Usuario", aPacN.get(i).getcUsuario());
            oJson.put("Nombre", aPacN.get(i).getcNombre());
            oJson.put("Pass", aPacN.get(i).getcContraseña());
            oJson.put("dNacimiento", aPacN.get(i).getdNacimiento());
            oJson.put("Direccion", aPacN.get(i).getcDireccion());
            oJson.put("Telefono", aPacN.get(i).getcTelefono());
            oJson.put("Email", aPacN.get(i).getcEmail());
            oJson.put("dCreacion", aPacN.get(i).getdCreacion());

            aJson.put(oJson);
        }

        String[] aParams = {"post", "http://hecbatutfg.esy.es/v1/pacientes/registrarlista",
aJson.toString()};
        String cRes;
        try {
            RestCliente oRest = new RestCliente();
            cRes = oRest.execute(aParams).get();
        } catch (Exception e) {
            return false;
        }

        if (cRes.equals("Error")) return false;

        return true;
    } catch (JSONException e) {
        return false;
    }
}
```

Una cosa muy importante que podemos observar en el código de *SubirPacNuevos* es que el cuerpo de a petición lo construimos en formato JSON. Podríamos haber escogido XML como formato para las comunicaciones, pero nos decantamos por JSON porque este resulta más ligero y su manejo en PHP resulta mucho más fácil.

PHP:

Toda petición que llegue al servicio será dirigida al archivo *index.php*, en el código de este se recogen los parámetros que hemos recibido y se redirige a la clase correspondiente.

```
// Extraer segmento de la url
if (isset($_GET['PATH_INFO']))
    $peticion = explode('/', $_GET['PATH_INFO']);
else
    throw new ExcepcionApi(ESTADO_URL_INCORRECTA, utf8_encode("No se reconoce
la petición"));

// Inicio BD

// Obtener recurso
$recurso = array_shift($peticion);
$recursos_existentes = array('pacientes', 'encuestas', 'pacienteresult', 'resultace',);

// Comprobar si existe el recurso
if (!in_array($recurso, $recursos_existentes)) {
    throw new ExcepcionApi(ESTADO_EXISTENCIA_RECURSO,
        "No se reconoce el recurso al que intentas acceder");
}

$metodo = strtolower($_SERVER['REQUEST_METHOD']);

// Filtrar método
switch ($metodo) {
    case 'get':
        // Procesar método get
        switch ($recurso) {
            case 'pacientes':
                $vista->imprimir(pacientes::get($peticion));
                break;
            case 'pacienteresult':
                $vista->imprimir(pacienteResult::get($peticion));
                break;
            case 'encuestas':
                $vista->imprimir(encuestas::get($peticion));
                break;
            case 'resultace':
                $vista->imprimir(resultace::get($peticion));
                break;
        }
        break;

    case 'post':
```

```
        // Procesar método post
        switch ($recurso) {
            case 'pacientes':
                $vista->imprimir(pacientes::post($peticion));
                break;
            case 'pacienteresult':
                $vista->imprimir(pacienteResult::post($peticion));
                break;
            case 'encuestas':
                $vista->imprimir(encuestas::post($peticion));
                break;
            case 'resultace':
                $vista->imprimir(resultace::post($peticion));
                break;
        }
        break;
    case 'put':
        // Procesar método put
        switch ($recurso) {
            case 'pacientes':
                $vista->imprimir(pacientes::put($peticion));
                break;
            case 'pacienteresult':
                $vista->imprimir(pacienteResult::put($peticion));
                break;
            case 'encuestas':
                $vista->imprimir(encuestas::put($peticion));
                break;
            case 'resultace':
                $vista->imprimir(resultace::put($peticion));
                break;
        }
        break;

    case 'delete':
        // Procesar método delete
        switch ($recurso) {
            case 'pacientes':
                $vista->imprimir(pacientes::delete($peticion));
                break;
            case 'pacienteresult':
                $vista->imprimir(pacienteResult::delete($peticion));
                break;
            case 'encuestas':
                $vista->imprimir(encuestas::delete($peticion));
                break;
            case 'resultace':
                $vista->imprimir(resultace::delete($peticion));
                break;
        }
        break;
    default:
        // Método no aceptado
}
```

Como en la parte Android hemos visto como subimos la lista de pacientes nuevos aquí vamos a mostrar como el servicio recibe y almacena esta lista.

El índice nos habrá redirigida al método post de la clase pacientes tal y como hemos visto en el código anterior

```
public static function post($peticion)
{
    if ($peticion[0] == 'registrar') {
        return self::registrar();
    } else if ($peticion[0] == 'login') {
        return self::loguear();
    } else if ($peticion[0] == 'registrarlista') {
        return self::registrarLista();
    } else {
        throw new ExcepcionApi(self::ESTADO_URL_INCORRECTA, "Url mal formada", 400);
    }
}
```

La única función del método *post* es redirigir la petición a otro método que se pueda encargar de ella, en este caso será *registrarLista.*

```
private static function registrarLista()
{
    $cuerpo = file_get_contents('php://input');
    $lista = json_decode($cuerpo);

    foreach ($lista as $paciente) {

        $resultado = self::crear($paciente);

        switch ($resultado) {
            case self::ESTADO_CREACION_EXITOSA:
                http_response_code(200);
                break;
            case self::ESTADO_CREACION_FALLIDA:
                http_response_code(400);
                throw new ExcepcionApi(self::ESTADO_CREACION_FALLIDA, "Ha ocurrido un
error");
                break;
            default:
        }
    }

    return
        [
            "estado" => self::ESTADO_CREACION_EXITOSA,
            "mensaje" => utf8_encode("¡Registro con exito!")
        ];

}
```

Este método extraerá el cuerpo de la petición que contendrá una lista de pacientes, recorrerá la lista y por cada elemento llamará al método *crear* que insertará el paciente en la BBDD.

```
private static function crear($datosPaciente)
{
    $id         = $datosPaciente->Id;
    $idUsuario  = $datosPaciente->Usuario;
    $nombre     = $datosPaciente->Nombre;
    $contrasena = $datosPaciente->Pass;
    $nacimiento = $datosPaciente->dNacimiento;
    $direccion  = $datosPaciente->Direccion;
    $telefono   = $datosPaciente->Telefono;
    $correo     = $datosPaciente->Email;
    $creacion   = $datosPaciente->dCreacion;

    try {
        $pdo = ConexionBD::obtenerInstancia()->obtenerBD();

        // Sentencia INSERT
        $comando = "INSERT INTO " . self::NOMBRE_TABLA . " ( " .
            self::ID . "," .
            self::ID_USUARIO . "," .
            self::NOMBRE . "," .
            self::CONTRASENA . "," .
            self::NACIMIENTO . "," .
            self::DIRECCION . "," .
            self::TELEFONO . "," .
            self::CORREO . "," .
            self::CREACION . ")" .
            " VALUES(?,?,?,?,?,?,?,?,?)";

        $sentencia = $pdo->prepare($comando);

        $sentencia->bindParam(1, $id);
        $sentencia->bindParam(2, $idUsuario);
        $sentencia->bindParam(3, $nombre);
        $sentencia->bindParam(4, $contrasena);
        $sentencia->bindParam(5, $nacimiento);
        $sentencia->bindParam(6, $direccion);
        $sentencia->bindParam(7, $telefono);
        $sentencia->bindParam(8, $correo);
        $sentencia->bindParam(9, $creacion);

        $resultado = $sentencia->execute();

        if ($resultado) {
            return self::ESTADO_CREACION_EXITOSA;
        } else {
            return self::ESTADO_CREACION_FALLIDA;
        }
    } catch (PDOException $e) {
        http_response_code(400);
        throw new ExcepcionApi(self::ESTADO_ERROR_BD, $e->getMessage());
    }
}
```

5.3.3. Encuesta

La clase *Encuesta* es una de las más importantes de la aplicación porque contiene la estructura y la información de las evaluaciones. Pese a ser tan importante la mayor parte de su código resulta muy sencillo, lo más destacable es la creación del JSON que utilizaremos para guardar la estructura.

En este método insertaremos el nombre y la descripción de la encuesta dentro del JSON y, a continuación, le indicaremos a cada pregunta que nos dé su JSON para agregarlo a la estructura.

```
public JSONObject makeJson() {

    try {
        // Creamos el JSON de la ecuesta con su nombre y descripción
        Json = new JSONObject();
        Json.put("nombre", Nombre);
        Json.put("descripcion", Descripcion);

        JSONArray aPregJSON = new JSONArray();

        // Obtenemos el JSON de cada pregunta
        for (int i = 0; i < aPreguntas.size(); i++) {

            aPregJSON.put(aPreguntas.get(i).makeJson());
        }

        // Agregamos la lista de preguntas al JSON de encuesta
        Json.put("preguntas", aPregJSON);
        return Json;

    } catch (JSONException e) {
        return null;
    }
}
```

5.3.4. EncuestaActual

En algunos ámbitos el uso de la clase *Encuesta* puede resultar complicado ya que la información tiene que compartirse entre distintas pantallas.

Para simplificar el uso de *Encuesta* la hemos encapsulado dentro de otra clase *EncuestaActual*. Esta tan solo contendrá un objeto *Encuesta* en su interior, el

beneficio lo obtenemos en la forma que la creamos, al igual que en *ConexionBD* hemos aplicado un patrón singleton [Sing]. Siempre que queramos usar una encuesta la crearemos a través de *EncuestaActual,* así, aunque cambiemos de actividad los datos de la encuesta se encontrarán accesibles en esta variable 'global'.

En el siguiente código se puede observar la implementación del patrón en Java.

```
private static EncuestaActual instance = null;
private Encuesta encuesta;

// Patrón Singleton
protected EncuestaActual() {
   encuesta = new Encuesta();
}


// Creamos una nueva instancia de EncuestaActual si esta no existe aún
public static EncuestaActual getInstance() {
   if(instance == null) {
      instance = new EncuestaActual();
   }
   return instance;
}
```

5.4. Conclusión

A lo largo de este capítulo hemos observado como la fase de diseño es necesaria para realizar la fase de desarrollo de una manera eficiente, ya que al tener todo claramente definido con anterioridad podemos dedicarnos exclusivamente a producir el código. Sin embargo, pueden aparecer factores que provoquen cambios en la implementación tanto de la interfaz como de las funcionalidades del sistema. Cabe decir que en este caso el número de variaciones ha sido bajo, y la mayoría han sido de cara a mejorar la usabilidad y la experiencia de usuario. Una vez se ha desarrollado completamente el sistema, es el momento realizar una valoración global del proyecto, donde vamos a reflejar los aspectos más significativos.

6. Conclusiones

6.1. Problemas y soluciones

Teníamos previsto realizar pruebas de usabilidad con usuarios reales de la aplicación al final del desarrollo, pero como esto no ha sido posible por la falta de tiempo se ha tenido en cuenta el criterio de la directora experimental que ha sido la que ha determinado la adecuación de este aspecto a los requisitos originales.

El problema más significativo en el desarrollo de este proyecto ha sido subestimar el tiempo de desarrollo del servicio web. Esta era la primera vez que desarrollábamos con el lenguaje PHP y además él también era el primer servicio RESTful, se había considerado originalmente que el coste de aprender PHP sería mucho menor que el coste real, posponiendo el inicio de las fases de desarrollo.

Otro problema significativo fue el retraso en la fase de diseño del sistema, ya que para clarificar las especificaciones y requisitos fue necesario realizar varias sesiones con la directora experimental.

Finalmente, como consecuencia de los retrasos, a pesar de que las interfaces en el modo Tablet funcionan correctamente y está completa la usabilidad, quedaría optimizar su usabilidad convenientemente para facilitar la experiencia del usuario con este tipo de dispositivo.

6.2. Trabajo futuro

Aunque el proyecto está finalizado y el sistema se encuentra listo para usarse en un entorno real siempre quedan cosas pendientes, no porque las acordáramos y no las hayamos realizado, sino porque inicialmente quedaron fuera del alcance del proyecto.

En lo referente a la funcionalidad del sistema sería muy conveniente implementar un módulo de evaluadores. Crearíamos un usuario administrador que tendría el control total de la aplicación y se encargaría de dar de alta en el sistema los evaluadores que van a utilizar el sistema, estos podrían compartir los pacientes, encuestas y resultados, pero quedaría registrado quien ha

realizado cada acción. También se podría implementar un sistema de permisos para limitar que acciones puede realizar cada evaluador.

Otro módulo que resultaría interesante implementar sería uno referente a la comunicación entre el paciente y el evaluador. En este módulo los evaluadores podrían enviar notificaciones Push [Pmsg] a los pacientes, o incluso se podría implementar un chat que permitiera la comunicación directa.

Fuera del ámbito de las ampliaciones de funcionalidad una ampliación del sistema muy valiosa sería la creación de una aplicación web capaz de realizar todas las gestiones que ahora deben realizarse desde la aplicación, esto otorgaría una gran comodidad a los evaluadores que podrían elegir trabajar desde cualquier plataforma.

Finalmente cabe destacar que también sería muy útil ampliar la exportación de los datos añadiendo una mayor variedad de formatos como Access, CVS, Word o PDF.

6.3. Aportaciones

Se espera que este proyecto sea una gran aportación para los profesionales de este campo, de hecho, ya está planeada su implementación en el departamento de la directora experimental Patricia. Sus opiniones serán muy valiosas para depurar posibles errores y de cara al aprendizaje personal.

A nivel personal este proyecto me ha permitido desarrollar mis habilidades y me ha otorgado experiencia en algunos campos donde carecía de ella.

En primer lugar, la captura de requisitos y definición del alcance con un interesado real, aunque no es un cliente, ya que el sistema se va a implementar sin coste alguno para el departamento, si que se ha tenido que negociar el alcance del proyecto utilizando el tiempo disponible como coste. La experiencia obtenida mediante está interacción es muy importante de cara a desarrollarse en el campo de la ingeniería del software.

En segundo lugar, hay que mencionar el desarrollo del servicio en PHP, un lenguaje que nunca había utilizado y que ahora gracias a este proyecto conozco. No hay ninguna duda de que esto resulta de gran ayuda para el desarrollo profesional y amplia mis oportunidades laborales.

Por último, pero no menos importante, cabe destacar la experiencia obtenida en el desarrollo de aplicaciones Android. Aunque este es un entorno en el que ya contaba con bastante experiencia la creación de controles dinámicamente

para crear las encuestas a partir de un JSON y la creación de un cliente RESTful han sido dos desafíos a los que nunca me había enfrentado. Podemos afirmar claramente que mi dominio sobre el sistema Android se ha incrementado significativamente.

Abreviaturas

ACE-III: Addenbrooke's Cognitive Examination-III

BBDD: Base de datos

IDE: Integrated Developement Environment

JSON: JavaScript Object Notation

Bibliografía clínica

[ACE1] Hsieh, S., Schubert, S., Hoon, C., Mioshi, E., Hodges J.R.. Validation of the Addenbrooke's Cognitive Examination III in Frontotemporal Dementia and Alzheimer's Disease. *Dement. Geriatr. Cogn. Disord.* 36, 242-250 (2013).

[ACE2] Matías-Guiu J.A., Fernández de Bobadilla R., et al. Validation of the Spanish version of Addenbrooke's Cognitive Examination III for diagnosing dementia. *Neurología* 30, no.9, 545-551 (2015). doi:10.1016/j.nrl.2014.05.0041

[ACE3] Gu, S.L., Gau, S.S., Tzang, S.W., Hsu, W.Y. The ex-Gaussian distribution of reaction times in adolescents with attention-deficit/hyperactivity disorder. *Res Dev Disabil.* 34, no. 11, 3709-3719 (2013). doi: 10.1016/j.ridd.2013.07.025

[ACE4] Moret-Tatay, C., Lemus-Zúñiga, L.G., Tortosa, D.A., Gamermann, D., Vázquez-Martínez, A., Navarro-Pardo, E., Conejero, J.A. Age slowing down in detection and visual discrimination under varying presentation times. *Scand. J. Psychol.* En prensa 2017 doi: 10.1111/sjop.12372

Bibliografía tecnológica

[Apa] Apache POI: https://poi.apache.org/ (Consultado el 09/07/2017)

[Mat] Material Design para Android: https://developer.android.com/design/material/index.html (Consultado el 09/07/2017)

[Pmsg] Firebase: Notificaciones Push https://firebase.google.com/docs/cloud-messaging/ (Consultado el 09/07/2017)

[Rest] TutorialsPoint: RESTful https://www.tutorialspoint.com/restful/restful_introduction.htm (Consultado el 09/07/2017)

[Sing] OO Design: Patrón Singleton http://www.oodesign.com/singleton-pattern.html (Consultado el 09/07/2017)

[Stat] StatCounter: Mercado de dispositivos Android. http://gs.statcounter.com/os-market-share/mobile/spain (Consultado el 09/07/2017)