Facultad
de
Ciencias

APLICACIÓN MÓVIL PARA MONITORIZAR TRASTORNOS DE MOVILIDAD

(Mobile application to monitor mobility disorders)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERIA INFORMATICA

Autor: Pablo Nodar Campos

Director: Rafael Duque Medina

Julio – 2017

Resumen

En la actualidad existen una serie de enfermedades (Alzheimer, Parkinson, etc.) que afectan a gran parte de la población mundial, las cuales provocan ciertos trastornos de movilidad en las personas que las padecen. En este contexto este proyecto propone el empleo de un Smartphone acompañando al paciente y analizando sus movimientos, de manera que este dispositivo queda totalmente integrado con la persona, sin ser percibido como un objeto diferenciado. Este fenómeno se conoce como computación ubicua.

Con esta finalidad se construye una aplicación que utiliza el acelerómetro del Smartphone para recoger los datos de las aceleraciones que experimenta el usuario en los tres ejes: movimientos sobre el eje X, sobre el eje Y, y sobre el eje Z. Esto es una gran ventaja para el personal médico, ya que con la recogida de datos se puede entender mucho mejor los patrones de movimiento de los pacientes, así como realizar un estudio detallado de cómo influyen estas patologías en la movilidad del usuario.

Para finalizar, esta aplicación ha incluido funcionalidades para enviar los datos recopilados por correo electrónico y consultar los datos almacenados con respecto a las enfermedades, pacientes y cuidadores.

Palabras clave: Trastornos de movilidad, aplicación móvil, servicios web, acelerómetro, computación ubicua

Abstract

Nowadays there are a lot of diseases affecting the world's population, which cause certain mobility disorders in these people (Alzheimer, Parkinson, etc.). In this context, this project proposes the use of a Smartphone accompanying the patient and analyzing their movements, so this device is totally integrated with the person, without being perceived as a differentiated object. This phenomenon is known as ubiquitous computing.

To accomplish the aim of the application it will use the accelerometer of the Smartphone to collect the data of the accelerations that the user experiences in the three axes: movements on the X axis, on the Y axis, and on the Z axis. This is a great Advantage for the medical staff, since with the collection of data the movement patterns of patients can be analyzed in a better way, as well as a detailed study of how these pathologies influence the mobility of the user.

To conclude, this application has included features to send the data collected by email and query stored data regarding diseases, patients and caregivers.

Keywords: mobility disorders, accelerometer, mobile application, web service, pervasive computing

Índice

Índice de Figuras

1 Introducción

En este capítulo se encontrarán los principales objetivos del proyecto, así como los alicientes para la realización del mismo. El capítulo se compone de tres secciones. En la primera sección se analizan las motivaciones de este trabajo y el contexto tecnológico en el que se desarrolla. Se plantea la manera de monitorizar los trastornos de movilidad que forman parte de la sintomatología de distintas enfermedades (trastornos relacionados con Alzheimer, epilepsia, etc.) empleando el acelerómetro de los actuales teléfonos móviles. En la segunda sección se describen los objetivos que tiene el trabajo, entre los que se encuentran el de monitorizar la información suministrada por el acelerómetro, identificar los patrones de movimiento del usuario, enviar dichos datos a través de un correo electrónico a la persona encargada de dicho paciente para poder analizar el comportamiento del paciente y consultar el estado del paciente de manera que la aplicación nos diga si el paciente esta leve, moderado o grave. Finalmente, en la última sección se muestra la estructura que tendrá el resto de este documento.

1.1. Motivación y contexto tecnológico

En la actualidad, los dispositivos móviles son capaces de realizar múltiples operaciones y cálculos de manera muy rápida y sencilla. Dichos dispositivos contienen una serie de componentes que permiten llevar a cabo distintas funciones que pueden resultar realmente útiles para un determinado perfil de usuario. En este caso nos vamos a centrar en un sensor que poseen todos los dispositivos móviles en la actualidad, el acelerómetro.

El acelerómetro de un dispositivo móvil es algo parecido a un chip, de tamaño muy reducido. Se compone básicamente de un condensador que está integrado por dos placas metálicas fijas y un material dieléctrico (un aislante) entre ambas. De esta manera, cuando ejercemos una aceleración, es decir, un cambio de velocidad sobre el dispositivo, la capacidad del condensador cambiará, debido a que el material dieléctrico dejará mayor o menor espacio con las placas entre las que se encuentra, y así, midiendo los cambios de carga, podremos identificar los movimientos bruscos que se puedan dar en el dispositivo [1].

Los trastornos de movilidad son desordenes que provocan en los enfermos movimientos descoordinados, movimientos erráticos, como por ejemplo cuando un enfermo de alzhéimer pierde la memoria, o movimientos bruscos y violentos. Cuando una persona enferma tiene una crisis, estos movimientos se agudizan. Partiendo de la base de que existen enfermedades, como el Alzheimer o la epilepsia, que provocan los mencionados trastornos de movilidad, el objetivo es emplear el acelerómetro del móvil para poder controlar estos desordenes. Gracias al acelerómetro, es posible controlar los movimientos que se realizan en el terminal y, de esta manera, controlar también los movimientos del usuario que porta dicho terminal para analizar la evolución de estos trastornos de movilidad.

1.2. Objetivos

El objetivo principal de este proyecto es diseñar y desarrollar una aplicación para dispositivos móviles, que permita la monitorización de pacientes con trastornos de movilidad. Este objetivo global se puede concretar en los siguientes subobjetivos:

- Monitorizar los datos registrados por el acelerómetro del teléfono móvil del usuario.
    - Visualizar los datos recopilados mediante el acelerómetro.
    - Almacenar los datos del acelerómetro en un fichero.
    - Enviar los datos recopilados por correo electrónico para su posterior estudio.
- Almacenar los distintos datos recopilados por el terminal en una base de datos.
- Consultar mediante un servicio web el estado de la enfermedad del paciente.

1.3. Estructura

Este documento contiene 5 capítulos adicionales, con el siguiente contenido:

- **Capítulo 2: Material y métodos**, se describirán las herramientas, tecnologías y metodologías que se emplean a lo largo del trabajo.
- **Capítulo 3: Análisis de requisitos**, se definirán los requisitos funcionales y no funcionales.
- **Capítulo 4: Diseño e implementación**, se expondrá como se ha diseñado y desarrollado la aplicación.
- **Capítulo 5: Evaluación y pruebas**, se mostrarán las pruebas realizadas para la verificación de la aplicación desarrollada.
- **Capítulo 6: Conclusiones y trabajos futuros**, se detallarán las conclusiones sacadas tras la realización del proyecto. Además, se desarrollarán las posibilidades que este proyecto abre a trabajos futuros.

2 Material y Métodos

En este capítulo se incluyen dos apartados. En el primer apartado se desarrollan las herramientas y tecnologías que se emplean en el proyecto, en el segundo la metodología que se seguirá, en el tercer aparatado se presenta un dataset con datos de movilidad de pacientes que se utilizó para generar la aplicación.

2.1 Herramientas y Tecnologías

En este apartado se describirá el *framework* usado para el desarrollo de la aplicación móvil, *Android Studio*, así como otras herramientas y tecnologías que se usen durante el desarrollo del proyecto.

2.1.1 Android Studio

Android Studio es el *IDE (Integrated Development Environment)* oficial para el desarrollo de aplicaciones *Android*. Este *framework* ofrece un desarrollo sencillo e intuitivo con las múltiples características que ofrece a la hora de diseñar la aplicación e implementar la misma [2]. Las características más importantes son las siguientes:

- Renderización en tiempo real.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en Gradle.
- Refactorización especifica de Android y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear.

Anteriormente el *framework* utilizado era *Eclipse,* pero en los últimos meses, *Android Studio* ha cogido fuerza y ha conseguido superar a *Eclipse* y fue anunciado como IDE oficial para el desarrollo de aplicaciones *Android* por *Google*.

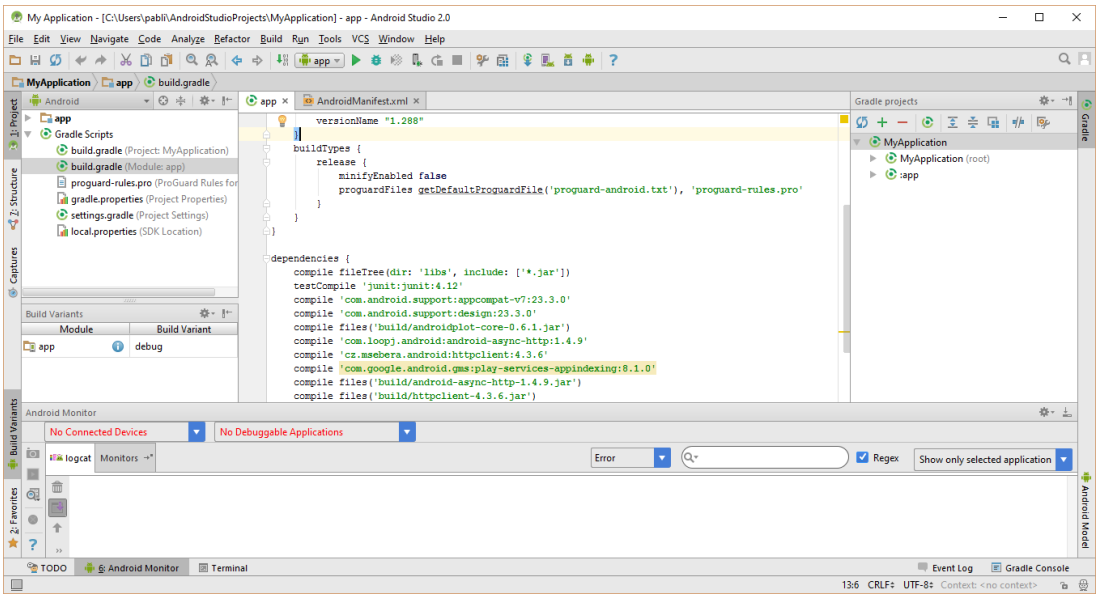


Figura 1 . *Android Studio*

2.1.2 SQLite

SQLite [3] es un gestor de base de datos de código abierto bastante popular, que utiliza el lenguaje *Structured Query Language* (SQL). A diferencia de otros gestores de base de datos, SQLite lee y escribe directamente en un archivo generado. Gracias a esto, se genera un archivo local dentro del dispositivo móvil y el acceso a la base de datos es relativamente sencillo. Esto sumando a la clase ya implementada en Android Studio *SQLiteOpenHelper* la experiencia a la hora de gestionar bases de datos es más sencilla y funcional.

2.1.3 DBDesigner

DBDesigner [4] es un software libre para el diseño visual de bases de datos que integra el diseño, modelado, creación y mantenimiento de bases de datos en un entorno individual y sin fisuras. El uso de este software para este proyecto se limita al diseño del diagrama del modelo relacional de la base de datos.

2.1.4 XML

XML (Extensible Markup Language) es un lenguaje de etiquetado muy simple pero muy importante a la hora del intercambio de datos. El uso principal que se le da a este lenguaje es el del almacenamiento de datos mediante etiquetas, de manera que los datos sean fáciles de identificar y perfectamente legibles. En este proyecto se empleará este lenguaje para desarrollar las interfaces de las distintas actividades que componen la aplicación móvil[5].

2.1.5 Tomcat

Apache Tomcat es un software libre basado en *Java Servlets* [6]. Básicamente es utilizado para *hostear* el servicio web en un ordenador de manera que accediendo a la IP de nuestro ordenador podremos conectarnos sin problemas al servicio web y consultar la información deseada.

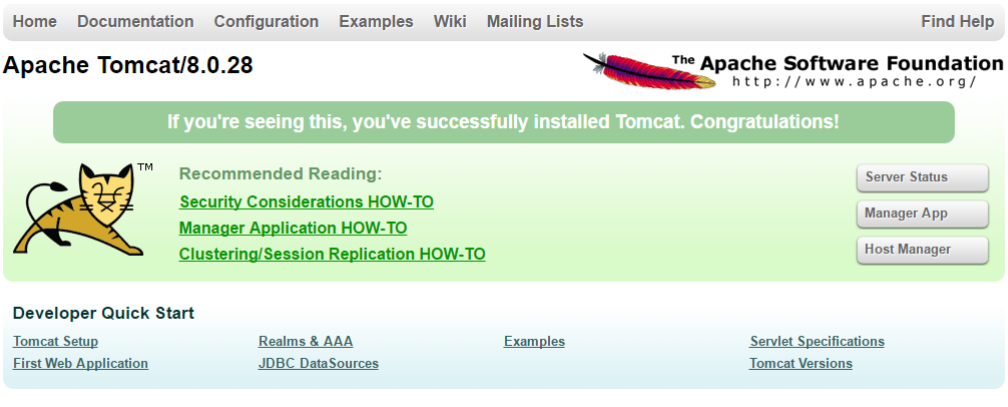


Figura 2 . *Tomcat*

2.1.6 SoapUI

SoapUI es la herramienta de prueba de API más utilizada actualmente y es de código abierto. Esta herramienta, ofrece pruebas funcionales para servicios web, así como pruebas REST para dichos servicios. En este proyecto se utiliza esta herramienta para probar el servicio web desarrollado [7].

Figura 3 . Logo *SoapUI*[13]

2.1.7 Eclipse

Eclipse es un IDE de *Java*[8]. Este *framework* es el más famoso y usado para la programación en *java* así como en otros lenguajes de programación, debido a su versatilidad y a la capacidad de adaptarse a cualquier lenguaje añadiendo un simple *add-on* al *framework*. En este proyecto se empleará para el desarrollo del servicio web sobre el que la aplicación realizará una consulta.

2.1.8 AnyPoint Studio

AnyPoint Studio[17] es un framework utilizado para la programación en distintos lenguajes. Principalmente se utiliza para el diseño y prueba de aplicaciones para la tecnología *Mule*. El empleo de este framework para el proyecto, es principalmente la facilidad que ofrece a la hora de desarrollar servicios web, en este caso uno de tipo *SOAP*, permitiendo al usuario abstraerse de configuraciones avanzadas, siendo necesario solamente la implementación lógica y una configuración muy básica.

2.2 *Metodología*

Como en todo proyecto software, es imprescindible seguir una metodología para definir los diferentes pasos a la hora de desarrollar el trabajo. Para esto se ha elegido seguir la metodología iterativa incremental (figura 4).

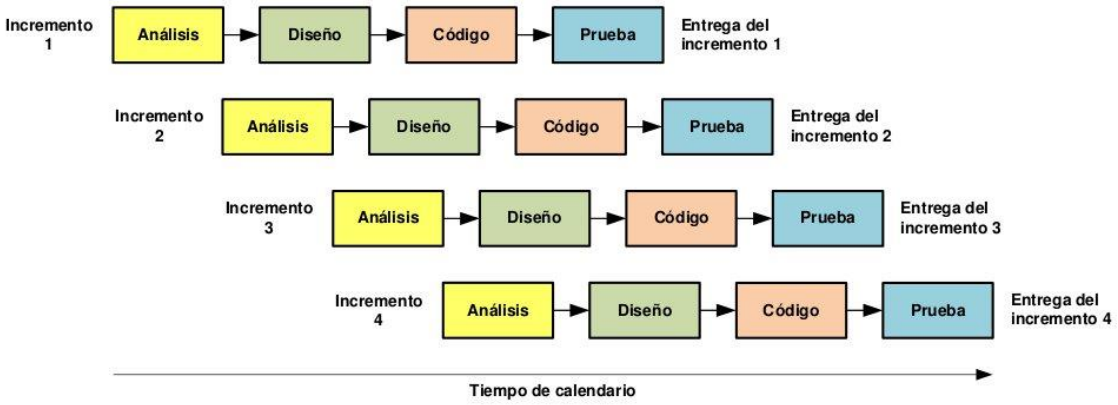


Figura 4 . Metodología iterativa e incremental*[13]*

Siguiendo esta metodología, se deberán ir analizando los diferentes requisitos para poder diseñar la aplicación y desarrollarla. Una vez hecho esto tendremos una primera versión, la cual será probada y de la cual aparecerán nuevos requisitos que se deberán volver a analizar y realizar el trabajo anterior de manera iterativa para ir incrementando poco a poco la aplicación, de manera que satisfaga todos los requisitos encontrados.

2.3 *Dataset de un caso de estudio en pacientes de Alzheimer*

Para la realización de este proyecto se ha empleado un dataset [16] donde se monitoriza mediante el acelerómetro de un Smartphone comercial la actividad de 35 pacientes con Alzheimer, entre los cuales se distingue el estado de la enfermedad del paciente como leve, moderado y avanzado de acuerdo al criterio médico. Finalmente, los autores aplicaron distintas técnicas de aprendizaje automático para tratar de estimar el estado de la enfermedad a partir de los datos registrados por el acelerómetro. Así los autores generar un árbol de decisión para estimar el estadio de la enfermedad (leve, moderada o grave) a partir de los datos recopilados por el acelerómetro. El árbol decisión subyacente será utilizado en este trabajo para implementar una funcionalidad predictora del estadio de la enfermedad a partir de la información recopilada por el acelerómetro.

.

3 Análisis de Requisitos

A continuación, se explicarán los requisitos identificados a la hora de realizar el proyecto. Primero se desarrollarán los requisitos funcionales y posteriormente los no funcionales

3.1 Requisitos Funcionales

En esta parte se desarrollarán los requisitos funcionales identificados para el desarrollo de la aplicación. En dicha aplicación se encuentran dos roles principales. El rol de paciente, el cual no tiene ninguna función más que el uso de la aplicación, y el rol de persona responsable. Dicho rol es el encargado de proporcionar los datos personales y de poder ver los datos recogidos por la aplicación. A continuación, se pasará a detallar los distintos requisitos funcionales identificados, los cuales se han identificado siguiendo la metodología iterativa e incremental, de manera que, si en la primera iteración se han obviado algunos requisitos, estos han podido ser reconocidos en posteriores iteraciones.

Identificador	Descripción	Iteración
RF01	El usuario deberá registrar los datos del paciente	Iteración 01
RF02	El usuario podrá modificar los datos del paciente	Iteración 01
RF03	El usuario podrá iniciar sesión dentro de la aplicación	Iteración 01
RF04	El usuario podrá modificar el teléfono de contacto del paciente	Iteración 01
RF05	El usuario podrá ver los datos del acelerómetro.	Iteración 02
RF06	El usuario podrá elegir el idioma de la aplicación	Iteración 03
RF07	El usuario podrá registrar a un nuevo cuidador	Iteración 04
RF08	El usuario podrá consultar las enfermedades almacenadas en la base de datos	Iteración 04
RF09	El usuario podrá enviar por correo los datos almacenados por el monitor	Iteración 04
RF10	El usuario podrá consultar los cuidadores almacenados en la base de datos	Iteración 04
RF11	El usuario podrá consultar el estado de la enfermedad del paciente	Iteración 05

Tabla 1 . Especificación de requisitos funcionales

Basándonos en los requisitos funcionales obtenidos tras la metodología iterativa e incremental implementada, hemos llegado a obtener 10 casos de uso que serán necesarios para satisfacer los requisitos encontrados. Dado que la aplicación se usará para controlar a pacientes y estos no interactúan con el sistema, el sistema tendrá un solo actor, una "Persona Responsable" o "Cuidador".

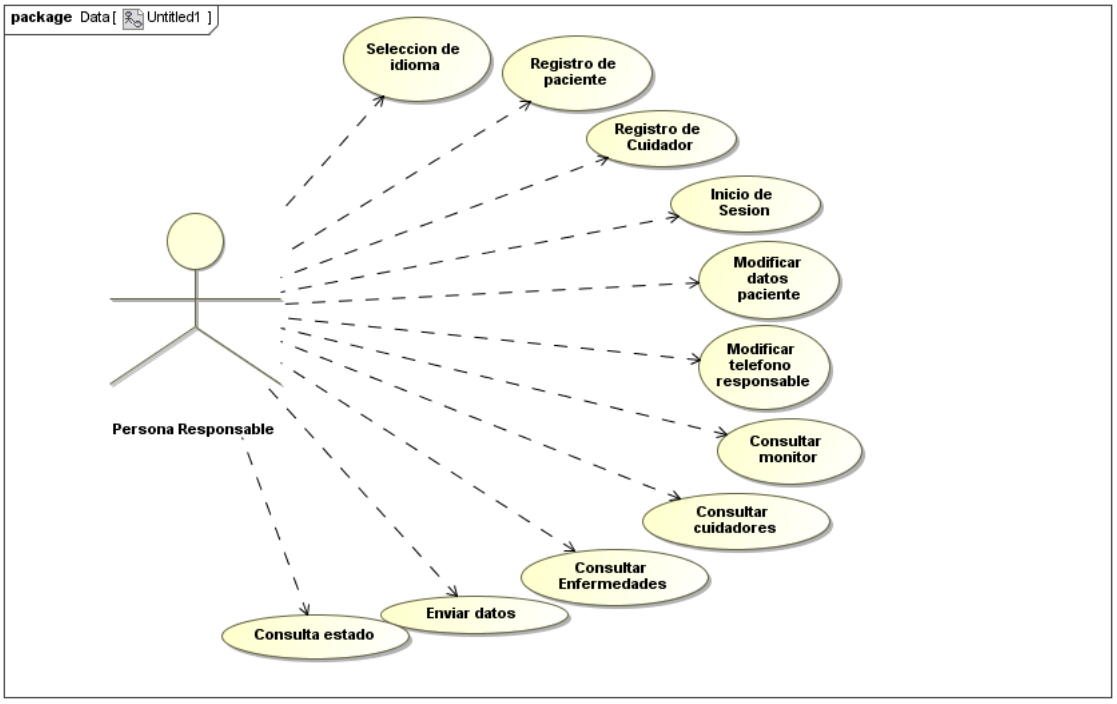


Figura 6 . Especificación de los casos de uso

3.1.1 Casos de uso

Debido a que el usuario paciente no realiza ninguna acción con la aplicación, todos los casos de uso identificados a raíz de los requisitos funcionales son para el rol de "Persona Responsable".

Caso de uso: Elegir Idioma
Precondiciones:
Postcondiciones: • La aplicación se ejecutará en el idioma seleccionado
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Selecciona el idioma 3. La aplicación se ejecuta en el idioma seleccionado
Flujo alternativo:

Tabla 2 . Caso de uso: Elegir Idioma

Caso de uso: Registro Paciente
Precondiciones:
Postcondiciones: • El paciente debe tener todos sus datos personales registrados
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Selecciona el idioma deseado 3. Selecciona la opción Registro paciente 4. Ingresa los datos propicios 5. Selecciona la opción guardar 6. Los datos se almacenan correctamente
Flujo alternativo: A partir del punto 5: 1. La persona responsable no ingresa los datos correctamente 2. La aplicación muestra un mensaje de error y vuelve a la actividad anterior

Tabla 3 . Caso de uso: Registro Paciente

Caso de uso: Registro Cuidador
Precondiciones:
Postcondiciones: • Un nuevo cuidador es dado de alta en el sistema
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Selecciona el idioma deseado 3. Selecciona la opción Registro cuidador 4. Ingresa los datos propicios 5. Selecciona la opción guardar 6. Los datos quedan almacenados
Flujo alternativo: A partir del punto 5: 1. La persona responsable no ingresa los datos correctamente 2. La aplicación muestra un mensaje de error y vuelve a la actividad anterior

Tabla 4 . Caso de uso: Registro Cuidador

Caso de uso: Consultar Enfermedades
Precondiciones:
Postcondiciones:
Escenario de éxito: 1. La persona responsable ejecuta la aplicación 2. Selecciona el idioma deseado 3. Selecciona la opción “Enfermedades” 4. El sistema muestra las enfermedades registradas
Flujo alternativo:

Tabla 5 . Caso de uso: Consultar Enfermedades

Caso de uso: Consultar Cuidadores
Precondiciones:
Postcondiciones:
Escenario de éxito: 1. La persona responsable ejecuta la aplicación 2. Selecciona el idioma deseado 3. Selecciona la opción “Cuidadores” 4. El sistema muestra las enfermedades registradas
Flujo alternativo:

Tabla 6 . Caso de uso: Consultar Cuidadores

Caso de uso: Inicio Sesión
Precondiciones:
Postcondiciones: • El usuario queda logueado en el sistema y puede acceder a nuevas opciones
Escenario de éxito: 1. La persona responsable ejecuta la aplicación 2. Selecciona el idioma deseado 3. Ingresa los datos para el inicio de sesión 4. Selecciona la opción aceptar 5. El usuario entra en el sistema
Flujo alternativo: A partir del punto 3: 1. Los datos introducidos por el usuario no son los correctos 2. El sistema muestra un error 3. Los campos rellenados por el usuario son vaciados

Tabla 7 . Caso de uso: Inicio Sesión

Caso de uso: Registro de teléfono de contacto
Precondiciones: • El Cuidador debe estar registrado en el sistema • El Paciente debe estar registrado en el sistema • El Paciente debe tener asociado dicho cuidador
Postcondiciones: • El paciente tendrá el teléfono de contacto asignado
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Ingresa en el sistema con su usuario y el del paciente 3. Selecciona la opción teléfono de contacto 4. Ingresa los datos propicios 5. Selecciona la opción guardar 6. Los datos se almacenan correctamente
Flujo alternativo: A partir del punto 2: 1. El cuidador no le corresponde al paciente 2. Se muestra un mensaje de error A partir del punto 4: 3. El móvil no tiene acceso a internet 4. La aplicación muestra un mensaje de error 5. Los datos no se almacenan

Tabla 8 . Caso de uso: Registro de teléfono de contacto

Caso de uso: Modificar datos paciente
Precondiciones: • El Cuidador debe estar registrado en el sistema • El Paciente debe estar registrado en el sistema • El Paciente debe tener asociado dicho cuidador
Postcondiciones: • El paciente tendrá los datos modificados
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Ingresa en el sistema con su usuario y el del paciente 3. Selecciona la opción datos personales 4. Ingresa los datos propicios 5. Selecciona la opción guardar 6. Los datos se almacenan correctamente
Flujo alternativo: A partir del punto 2: 1. El cuidador no le corresponde al paciente 2. Se muestra un mensaje de error A partir del punto 4: 3. El móvil no tiene acceso a internet 4. La aplicación muestra un mensaje de error 5. Los datos no se almacenan

Tabla 9 . Caso de uso: Modificar datos paciente

Caso de uso: Consultar monitor
Precondiciones: • El Cuidador debe estar registrado en el sistema • El Paciente debe estar registrado en el sistema • El Paciente debe tener asociado dicho cuidador
Postcondiciones:
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Selecciona el idioma deseado 3. Ingresa en el sistema 4. Selecciona la opción monitor 5. La aplicación muestra en vivo los datos que va recogiendo el acelerómetro
Flujo alternativo:

Tabla 10 . Caso de uso: Consultar monitor

Caso de uso: Enviar datos por correo
Precondiciones: • El Cuidador debe estar registrado en el sistema • El Paciente debe estar registrado en el sistema • El Paciente debe tener asociado dicho cuidador
Postcondiciones:
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Selecciona el idioma deseado 3. Ingresa en el sistema 4. Selecciona la opción monitor 5. Se ingresa el email al que se quiere enviar la información 6. Selecciona la opción enviar 7. La aplicación abre la aplicación *GMAIL* para enviar el archivo generado por la aplicación
Flujo alternativo:

Tabla 11 . Caso de uso: Enviar datos por correo

Caso de uso: Consultar estado
Precondiciones: • El Cuidador debe estar registrado en el sistema • El Paciente debe estar registrado en el sistema • El Paciente debe tener asociado dicho cuidador • El terminal debe tener acceso a internet • El servicio web debe estar activo
Postcondiciones:
Escenario de éxito: 1. La persona responsable del paciente ejecuta la aplicación 2. Selecciona el idioma deseado 3. Ingresa en el sistema 4. Selecciona la opción monitor 5. Selecciona la opción estado paciente 6. La aplicación muestra la información devuelta por el servicio web.
Flujo alternativo:

Tabla 12 . Caso de uso: Consultar estado

3.2 Requisitos no Funcionales

Los requisitos no funcionales son condiciones que el sistema deberá cumplir en distintos ámbitos para definir las características del funcionamiento de la aplicación.

3.2.1 Usabilidad

La usabilidad [9] es quizá, uno de los factores más importantes dentro de la calidad de un software, y por ello es interesante contar con metodologías para medir dicho factor.

Según la ISO (International Organization for Standardization) [10] se define la usabilidad como la efectividad, la eficiencia y la satisfacción que un usuario tiene para conseguir sus objetivos en un entorno especifico. Por lo tanto, para nuestra aplicación será necesario que el usuario pueda monitorizar los movimientos de una manera sencilla y efectiva, así como satisfacer la necesidad de poder emplear los datos recogidos de alguna manera.

3.2.2 Compatibilidad con dispositivos

Debido al amplio catálogo de dispositivos que hay actualmente, la aplicación se desarrollará teniendo en cuenta que el usuario que utilice la aplicación puede tener desde un dispositivo con *Android* de última generación hasta un *Smartphone* más sencillo.

3.2.3 Conexión a internet

Ya que la aplicación se conecta con un servicio web para consultar información del paciente y envía por correo electrónico la información recogida del paciente, será necesario que el terminal disponga de datos móviles para una conexión a internet. Si el dispositivo no dispone de dichos datos la aplicación seguirá funcionando, pero las características antes mencionadas no estarán disponibles.

3.2.4 Necesidades hardware

Dado que el uso que se le da a la aplicación requiere del acelerómetro que tenga el dispositivo móvil, será necesario que el dispositivo posea este sensor, bien sea un *Smartphone*, una *Tablet* o cualquier otro dispositivo con un sistema operativo *Android*.

4 Diseño e implementación

En este capítulo se desarrollan los procesos de diseño de la aplicación y su implementación. El diseño de un sistema de software consiste en proponer distintas técnicas y procesos con suficiente detalle para permitir su implementación física. El proceso de implementación en cambio, consiste en aplicar dichas técnicas para desarrollar el sistema software. A continuación, se desarrollarán las técnicas empleadas y la implementación llevada a cabo.

4.1 Diseño arquitectónico

Figura 7 . Arquitectura en tres capas[15]

El diseño arquitectónico que se desarrollará en este proyecto es el diseño en tres capas: capa de presentación, capa de negocio y capa de datos.

- Capa de presentación: Esta capa es la que interactúa con el usuario. Da información al usuario y la recibe de este para que el usuario pueda interactuar con la aplicación. Esta capa se comunica con la capa de negocio.
- Capa de negocio: Esta capa es la intermediaria entre la capa de presentación (lo que el usuario ve) y la capa de datos (donde se encuentran todos los datos). Su principal función es la de recoger los datos de la capa de datos para mostrarlos en la capa de presentación, así como recoger datos de la capa de presentación para almacenarlos en la capa de datos.
- Capa de datos: En esta capa es donde se encuentran los datos almacenados y es la encargada de acceder a ellos. Está formada por un gestor de base de datos que realiza todo el almacenamiento de datos y recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

4.2 Diseño e implementación de la aplicación

A continuación, se va a desarrollar los pasos seguidos para el diseño y la implementación de la aplicación *Android.* Para ello se explicarán en tres secciones distintas (una por cada capa) como se ha llevado a cabo.

4.2.1 Diseño de la capa de datos

En este apartado, se desarrollará el diseño e implementación de la capa de datos a través del diseño relacional expuesto en la figura 10 donde se observan las relaciones entre tablas.

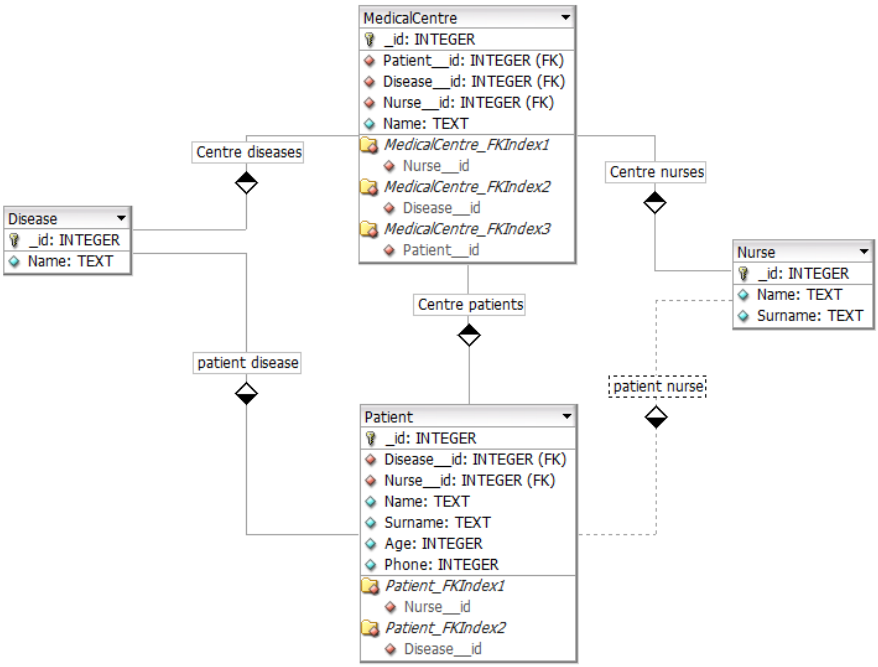


Figura 8 . Diseño BD

Como podemos ver en el diagrama, la tabla PATIENT es la tabla principal, la cual contiene los datos de interés del paciente, así como dos claves externas para relacionar el paciente con la enfermedad que tiene y con el cuidador asociado. La tabla DISEASE es la tabla que contiene la información de las enfermedades, de las cuales simplemente necesitamos saber el nombre. Para finalizar la tabla NURSE contiene el nombre y el apellido del cuidador.

Para implementar esta base de datos se ha empleado el lenguaje SQLite, ya que Android posee la clase *SQLiteOpenHelper*() que nos permite gestionar de una manera más sencilla las bases de datos desde la aplicación. Además, para generar el diagrama del diseño relacional hemos utilizado el software libre *DBDesigner.*

4.2.2 Diseño e implementación de la capa de negocio

En este apartado se explica el diseño e implementación de la capa de negocio. A continuación, en la Figura 11 se muestra cómo en la clase *Centro* se almacenan los datos correspondientes a los pacientes, cuidadores y enfermedades que se tratan en dicho centro. Un objeto de esta clase se utilizará para obtener la información del paciente y mostrarla correctamente en cada una de las pantallas de la aplicación.

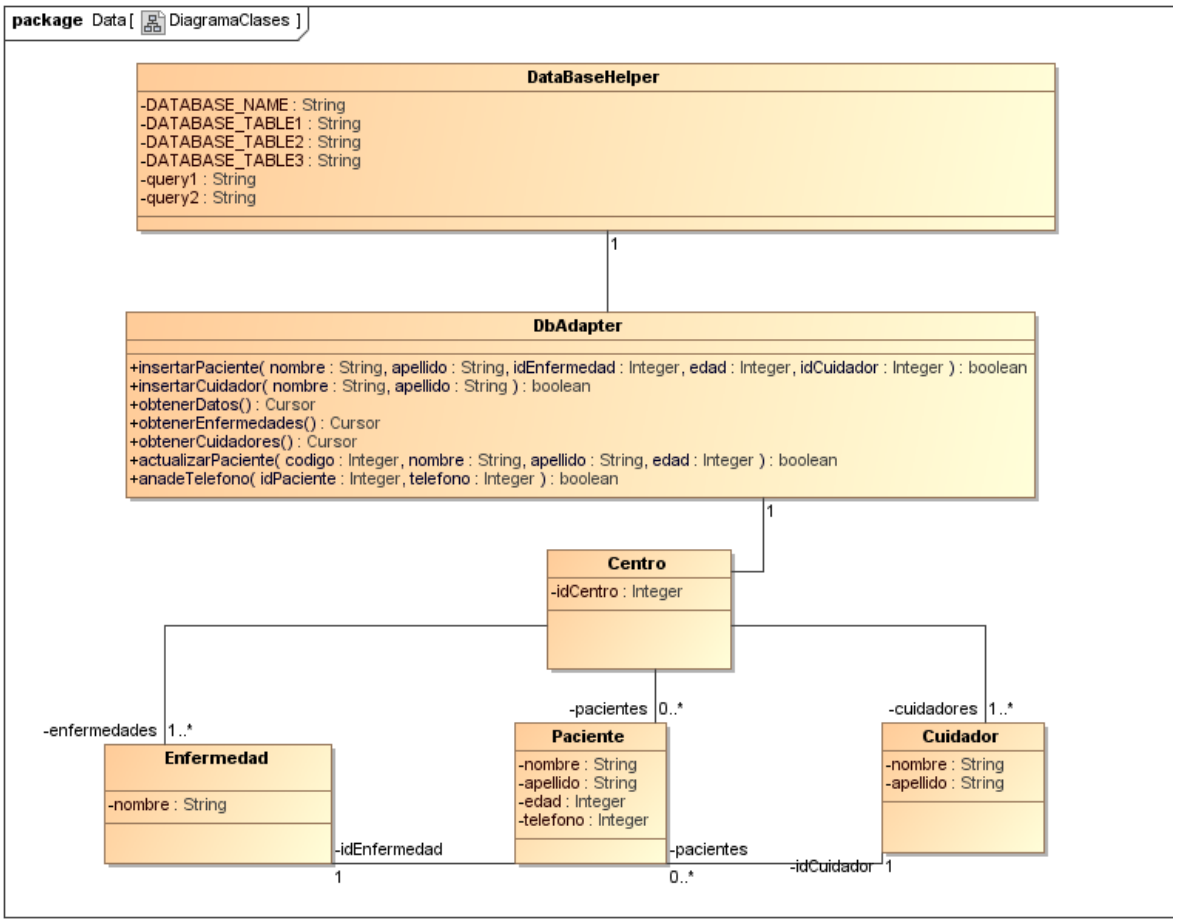


Figura 9 . Diagrama de clases

A continuación, se explica la implementación de la capa de negocio y diversas funcionalidades de la aplicación como serían la conexión con la base de datos, la manera en que se leen los datos del acelerómetro y estos se almacenan y envían.

4.2.3 Conectando con la base de datos

Para realizar las conexiones entre la base de datos y la aplicación se han implementado dos clases para poder realizarlo de manera exitosa. La primera es la clase *DbAdapter* la cual contiene métodos para la actualización de la base de datos y la conexión de la misma. Por otro lado, tenemos la clase *DataBaseHelper* que se encuentra dentro de la clase *DbAdapter* de manera que cuando se ejecuta la aplicación por primera vez, si no hay una base de datos creada se creará.

Dicha base de datos se encuentra almacenada de manera local en el teléfono móvil, lo que facilita la creación y modificación de la misma, así como un acceso directo sin depender de terceros y debido a su pequeño tamaño no habrá problema a la hora del almacenamiento. La creación y conexión a la base de datos se realiza de la siguiente manera:

- Dentro de la clase *DbAdapter* tenemos la conexión con la base de datos, de manera que si es la primera vez que ejecutamos el método se creará la base de datos en el dispositivo, y si por el contrario ya se ha conectado con la base de datos, la aplicación verificara si debe actualizar la base de datos o no.

```
public SQLiteDatabase open() throws SQLException {
    // Crea un objeto asistente de base de datos de clase SqLiteHelper.
    dbHelper = new DataBaseHelper(contexto);
    // Abre la base de datos en modo escritura (lectura permitida).
    db = dbHelper.getWritableDatabase();
    // Devuelve el objeto de tipo SQLiteDatabase.
    return db;
}

/**
 * Cierra la base de datos.
 */
public void close() { dbHelper.close(); }
```

Figura 10 . Conexión con la base de datos

- Al crear un objeto de la clase *DataBaseHelper* se ejecuta el método *onCreate()* de manera que se ejecutan una serie de sentencias SQLite para la creación de la base de datos.

```
public void onCreate(SQLiteDatabase db) {
    // Se ejecutan las sentencias de creacion de las tablas,
    // asi como las consultas para añadir datos por defecto.
    db.execSQL(TABLE_CREATE1);
    db.execSQL(TABLE_CREATE2);
    db.execSQL(TABLE_CREATE3);
    db.execSQL(query8);
    db.execSQL(query9);
    db.execSQL(query6);
    db.execSQL(query7);
    db.execSQL(query4);
    db.execSQL(query5);
    Log.e("TABLE_CREATE1", "" +
            TABLE_CREATE1);
    Log.e("TABLE_CREATE2", "" +
            TABLE_CREATE2);
    Log.e("TABLE_CREATE3", "" +
            TABLE_CREATE3);
```

Figura 11 . Creación de la base de datos

Teniendo en cuenta esto, para poder escribir y leer datos desde las clases de la aplicación haremos lo siguiente:

4.2.3.1 Escritura en la base de datos

El ejemplo que se muestra a continuación es el método *add()* dentro de la clase *RegistraPaciente* mediante el cual se añade un nuevo paciente a la aplicación. En primer lugar, la aplicación ejecuta el método antes nombrado al rellenar los datos dentro de la aplicación y pulsar sobre el botón *Save*.

```
    <Button
        android:text="Save"
        android:layout_x="7dp"
        android:id="@+id/Guardar"
        android:onClick="add"
        android:layout_y="230dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</AbsoluteLayout>
```

Figura 12 . Botón que inicia la escritura

Como podemos observar en la imagen, con el evento *onClick* se ejecutará el método *add()*, el cual funciona de la siguiente manera

```
public void add(View view){
    // Declaramos variables
    String nombre,apellido;
    int edad,enfermedad,cuidador;
    EditText nombret,apellidot,edadt,enfermedadt,cuidadort;

    nombret = (EditText) findViewById(R.id.Nombre);
    apellidot = (EditText) findViewById(R.id.Apellido);
    edadt = (EditText) findViewById(R.id.Edad);
    enfermedadt = (EditText) findViewById(R.id.Enfer);
    cuidadort = (EditText) findViewById(R.id.Cuidador);

    //Obtenemos Texto de los TextEdit
    nombre=nombret.getText().toString();
    apellido=apellidot.getText().toString();
    edad=Integer.parseInt(edadt.getText().toString());
    enfermedad=Integer.parseInt(enfermedadt.getText().toString());
    cuidador=Integer.parseInt(cuidadort.getText().toString());

    db = new DbAdapter(getApplicationContext());
    db.open();
    if(db.insertarPaciente(nombre,apellido,enfermedad,edad,cuidador)==true) {


        Cursor cursor = db.obtenerDatos();
        cursor.moveToFirst();
```

Figura 13 . Método para escribir en la base de datos

Como se puede observar, dentro del método se crea un objeto de la clase *DbAdapter* para poder llamar al método *insertarPaciente()*, método el cual hace lo siguiente.

```
public boolean insertarPaciente(String nombre, String apellido,int enfermedad,int edad,int cuidador){
    boolean resultado=false;

    try{
        String query="INSERT INTO Paciente(Name, Surname, Disease, Age, Nurse) VALUES('"+nombre+"','"+apellido+"','"+
                enfermedad+"','"+edad+"','"+cuidador+"')";

        db.execSQL(query);
        resultado=true;
        return resultado;
    }
    catch (Exception e){
        resultado=false;
        return resultado;
    }
}
```

Figura 14 . Método que ejecuta comando SQLite para escritura

Con los datos que se le pasa al llamar a este método dentro de la clase *DbAdapter* lo que se hace es ejecutar una consulta sobre la base de datos, de manera que en este ejemplo se ejecuta una sentencia *INSERT INTO* dentro de la tabla *Paciente*. Como ya vimos en el apartado anterior el objeto nombrado *db* sobre el que se llama al método *execSQL(query)* es un objeto de la clase *DataBaseHelper* que es el encargado de crear la base de datos y sobre el que se ejecutan las consultas. Dicho método está ya implementado por la clase *SQLiteDataBase* ya implementada en *Android Studio*.

4.2.3.2 Lectura de la base de datos

A continuación, mostraremos el método *enfermedades()*, el cual está dentro de la clase *mainMenu*. Como podemos observar en la siguiente imagen, el método es llamado por la opción *onClick* que se encuentra en un botón dentro del layout correspondiente a la clase anteriormente mencionada.

```
<Button
    android:layout_width="140dp"
    android:layout_height="50dp"
    android:text="Diseases"
    android:id="@+id/button3"
    android:layout_x="5dp"
    android:layout_y="250dp"
    android:onClick="enfermedades" />
```

Figura 15 . Botón que inicia el cambio de actividad

Básicamente este método permite el paso de la actividad principal a la siguiente actividad correspondiente a la clase *Diseases*, el modo para pasar a la siguiente actividad es bastante sencillo:

```
public void enfermedades(View view) {
    Intent i = new Intent(this, Diseases.class);
    startActivity(i);
}
```

Figura 16 . Método que lanza la nueva actividad

Una vez se pasa a la siguiente actividad se ejecuta directamente el método que realiza la consulta a la base de datos dentro del método *onCreate()* de la clase

```
public class Diseases extends AppCompatActivity {

    private DbAdapter db;
    TextView id1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.content_diseases);
        id1= (TextView) findViewById(R.id.tb1);
        obten();
    }
    public void obten(){
        db = new DbAdapter(getApplicationContext());
        db.open();
            Cursor cursor = db.obtenerEnfermedades();
            cursor.moveToFirst();
            while (!cursor.isAfterLast()) {
              Log.e("ID: ",cursor.getString(cursor.getColumnIndex("_id")));
                Log.e("Enfer",cursor.getString(cursor.getColumnIndex("Name")));
              id1.append("ID: "+cursor.getString(cursor.getColumnIndex("_id")));
               id1.append("    "+cursor.getString(cursor.getColumnIndex("Name"))+"\n");
               cursor.moveToNext();
            }
            cursor.close();

            db.close();
    }
}
```

Figura 17 . Clase que realiza la lectura de la base de datos

Como se puede observar, en el método *obten()*, llamado dentro del *onCreate()*, realiza la conexión a la base de datos, posteriormente recorre el cursor devuelto por el método *obtenerEnfermedades()*, y va mostrando los resultados que se encuentran en dicho cursor. Por último, se cierra la conexión con la base de datos. El método *obtenerEnfermedades()*, el cual se encuentra dentro de la clase *DbAdapter*, lo único que hace es una consulta sobre la base de datos y devolverlo sobre un objeto de la clase *Cursor*.

```
public Cursor obtenerPacientes(){
    return db.rawQuery("SELECT * FROM "+DataBaseHelper.DATABASE_TABLE3,null);

}
```

Figura 18 . Método que ejecuta comando SQLite para lectura

4.2.4 Servicio web

Para realizar la consulta acerca del estado del paciente, se ha desarrollado un servicio web en *java* utilizando el framework *AnyPointStudio*. Un servicio web es una tecnología que se emplea para el intercambio de datos entre aplicaciones, en el caso de este proyecto se intercambiará datos entre la aplicación cliente y la aplicación servidor. Dicho servicio web, corresponde a un servicio tipo *SOAP*, al cual se le manda una petición con una serie de valores. En este, el servicio web recibe como parámetros los valores máximos, mínimos y la media de los tres ejes que recoge el acelerómetro del dispositivo (eje x, eje y, eje z). Con dichos datos, el servicio hace uso del árbol de decisión introducido en 2.3 para estimar el estado actual del paciente con Alzheimer.

A parte de esto, también ha sido necesaria la instalación del software libre *Apache Tomcat* para poder *hostear* dicho servicio web y poder acceder a él de manera remota desde la aplicación.

web.xml | MovementControl.java

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" :
  <display-name>MovementControl</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>MovementControl</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>MovementControl</servlet-name>
    <url-pattern>/mov/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Figura 19 . Servicio web

4.3 Capa de presentación

A continuación, se va a pasar a explicar cómo se ha diseñado la capa de presentación, a parte de la implementación de la misma. Para completar la explicación se adjuntarán capturas de pantalla de cada actividad.

4.3.1 Diseño de la capa de presentación

En este apartado se mostrará un diagrama para ayudar al entendimiento de la aplicación y de todas las actividades a las que el usuario tiene acceso. De esta manera la capa de presentación se explica casi por si sola.

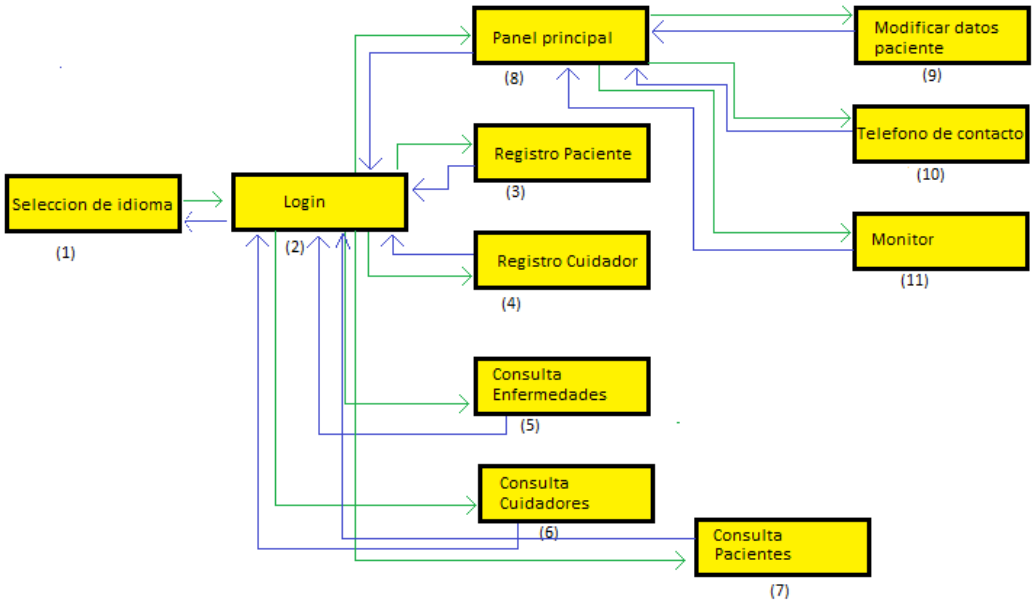


Figura 20 . Diagrama de las actividades de la aplicación

Como se ha mencionado anteriormente, la aplicación se ha desarrollado mediante el *framework Android Studio* y cada pantalla, por así decirlo, es denominado *Actividad* dentro del mismo *framework*. Como se puede observar en el diagrama anterior, el proyecto consta de 11 actividades las cuales pueden acceder a otras actividades dentro de un cierto orden como podemos observar en las flechas de dicho diagrama.

4.3.2 Implementación de la capa de presentación

A continuación, se explicará cómo funciona cada actividad que compone el proyecto, para poder hacernos una idea de cómo funciona la aplicación exactamente.

4.3.3 Selección de idioma

La primera actividad que el usuario encuentra al ejecutar la aplicación es la selección de idioma. Esta actividad es simple, pero marcará el posterior trascurso de la aplicación, es decir dependiendo del idioma seleccionado (español o inglés) la aplicación mostrará la información en uno u otro idioma.

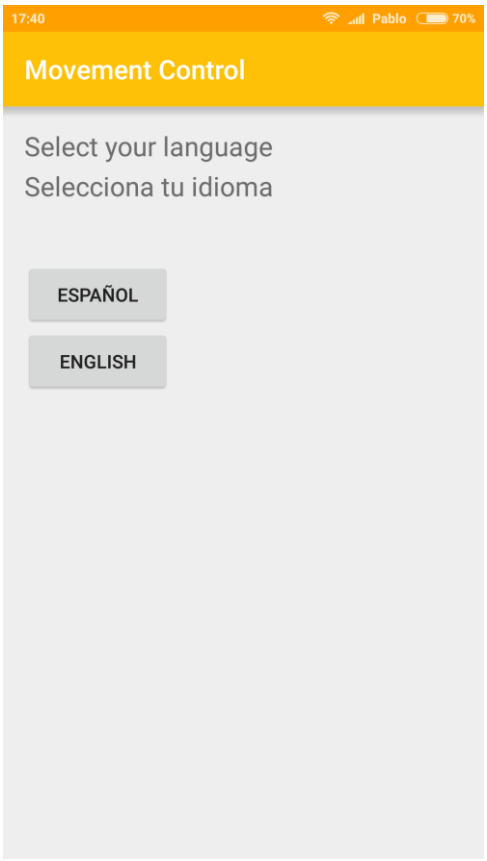


Figura 21 . Selección de idioma

Como podemos observar dependiendo del idioma seleccionado la aplicación se mostrará en uno u otro idioma, lo cual resulta interesante ya que para un mismo paciente puede consultarse la información independientemente de tu lengua materna.

4.3.4 Login dentro de la aplicación

Una vez hemos decidido el idioma en él se quiere que la aplicación se muestre, la siguiente actividad nos muestra dos campos en los que ingresar el id del cuidador y el id del paciente, así como un menú de 6 botones: Aceptar, Reg. Paciente, Enfermedades, Reg. Cuidador, Cuidadores y Pacientes.

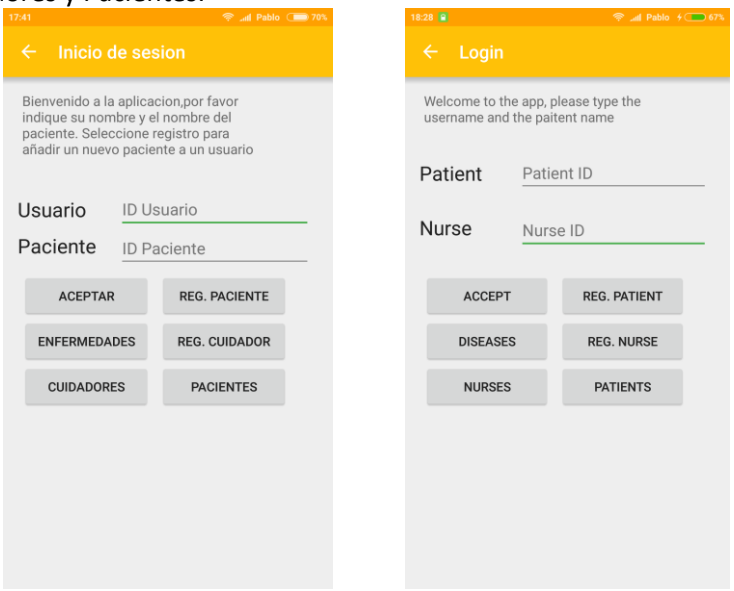


Figura 22 . Login dentro de la aplicación

Cada botón de ese menú nos llevara a una actividad específica, a excepción del botón Aceptar, el cual en caso de ingresar unos datos incorrectos no cambiara de actividad y nos mostrara un mensaje de error como se muestra a continuación

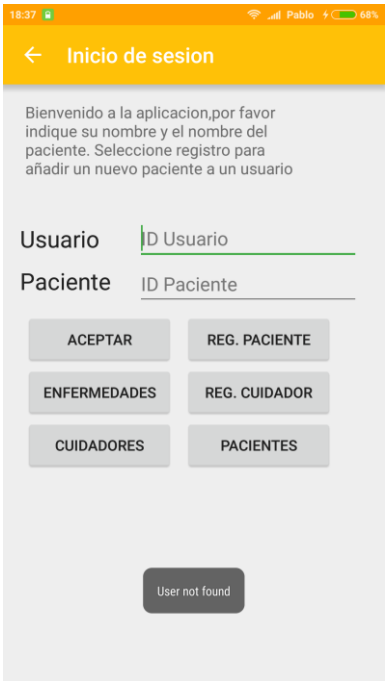


Figura 23 . Error al iniciar sesión

4.3.5 Registro de un paciente

Si se selecciona el botón Reg. Paciente la aplicación pasa a la siguiente actividad "Registro Paciente". Dentro de esta actividad, se podrá dar de alta a un nuevo paciente, de manera que ingresando ciertos datos el paciente quede almacenado en la base de datos

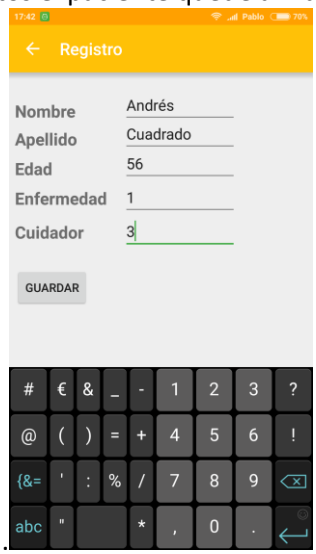


Figura 24 . Registro de un paciente

Una vez se acaba de ingresar los datos, se pulsa sobre el botón guardar, de esta manera se ejecuta el método encargado de recoger los datos y almacenarlos en la base de datos mediante una serie de sentencias para una base de datos SQLite.

4.3.6 Registro cuidador

Si en el menú principal se selecciona la opción Reg. Cuidador, la aplicación pasara a la actividad correspondiente, permitiendo al usuario de la aplicación registrar un nuevo cuidador.

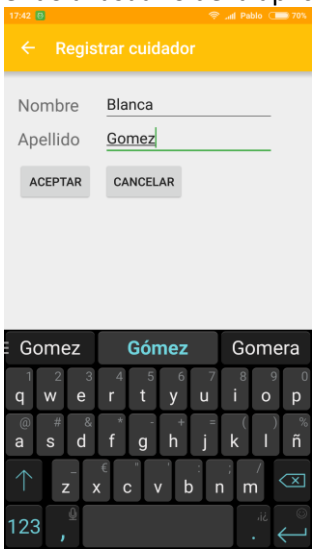


Figura 25 . Registro cuidador

Al ingresar los datos necesarios (nombre y apellido) se podrá seleccionar la opción Aceptar, para ejecutar los métodos necesarios para guardar los datos en la base de datos. La id se genera automáticamente de manera que el usuario no tiene que estar pendiente de cuál fue el último id empleado por un cuidador.

4.3.7 Consulta enfermedades

Al seleccionar el botón Enfermedades, la aplicación pasará a la Actividad Consulta enfermedades, en la cual se muestran simplemente las distintas enfermedades que soporta la aplicación.

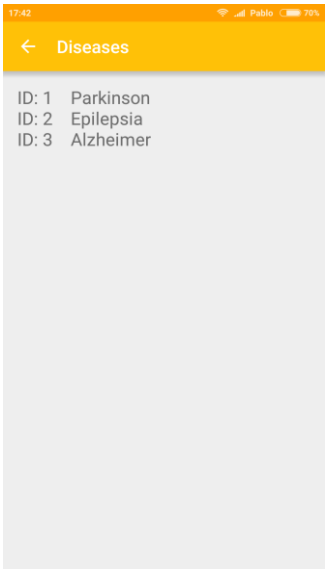


Figura 26 . Enfermedades almacenadas

Como se puede observar es una Actividad simple que hace una consulta sobre la base de datos de manera que se muestra el id de la enfermedad, así como el nombre de dicha enfermedad.

4.3.8 Consulta cuidadores

De manera análoga al punto anterior, cuando se selecciona la opción Cuidadores, la aplicación avanza a la actividad correspondiente en la cual se muestran los cuidadores registrados dentro de la aplicación.

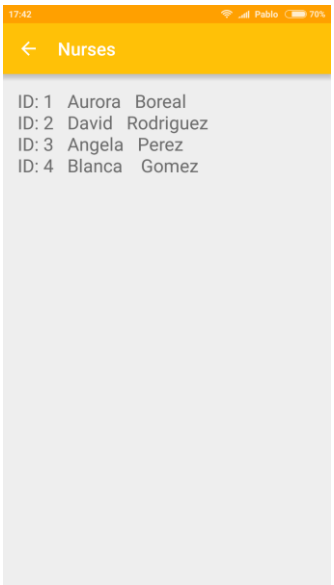


Figura 27 . Cuidadores almacenados

Como ocurre en la anterior consulta, se trata de una actividad sencilla en la cual se muestra el id de cada cuidador junto a su nombre y apellido

4.3.9 Consulta Pacientes

Al igual que en los puntos *4.3.7* y *4.3.8* se trata de una actividad simple de consulta a la cual se accede a través de la opción Pacientes.

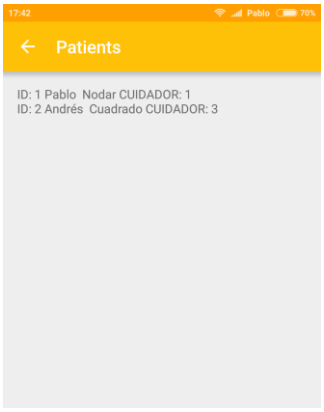


Figura 28 . Pacientes almacenados

Se observa que es una Actividad que realiza una consulta sobre la base de datos para mostrarnos los datos de los pacientes registrados en la aplicación, de manera que se muestra el id, el nombre, el apellido y la id del cuidador responsable. Cabe destacar que esta consulta resulta especialmente útil ya que para ingresar en la aplicación se necesitará saber el id del paciente que se quiere monitorizar y el id del cuidador asociado a ese paciente.

4.3.10 Panel principal

Una vez hemos ingresado los datos correctamente en la aplicación (recordemos que debemos ingresar la id del paciente y la id del cuidador asociado a este) y hemos presionado el botón aceptar, la aplicación avanzara a la siguiente actividad, el Panel principal.

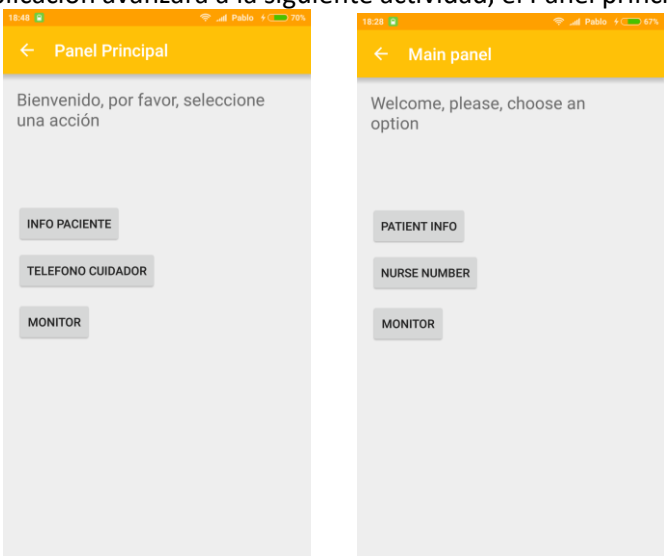


Figura 29 . Panel principal

Como se muestra en la imagen anterior, el panel principal se compone por un mensaje simple de bienvenida, así como un menú compuesto de tres botones distintos que harán que la aplicación avance a otras tres actividades distintas: Modificar datos paciente, Teléfono de contacto y Monitor

4.3.11 Modificar datos del paciente

Si se selecciona la opción Info Paciente, la aplicación iniciará la nueva actividad Mostrar datos paciente. Dentro de esta actividad se podrán modificar ciertos datos del paciente como son el nombre, el apellido y la edad, de esta manera se pueden corregir ciertos errores de escritura a la hora de ingresar un nuevo paciente, así como actualizar la edad de un paciente.

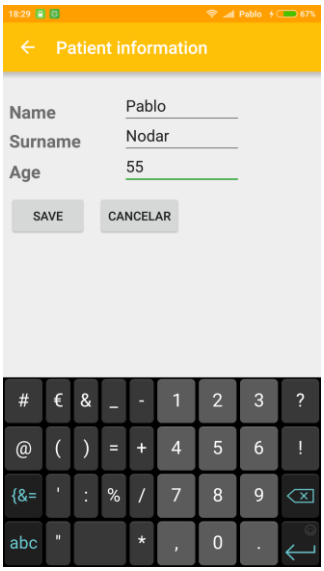


Figura 30 . Modificar datos del paciente

4.3.12 Teléfono de contacto

Si, una vez dentro del panel principal, se selecciona la opción Tel. Cuidador la aplicación iniciara la actividad correspondiente.

Figura 31 . Teléfono de contacto

Como se puede apreciar esta actividad es bastante simple, su utilidad es simplemente la de añadir un teléfono de contacto, normalmente el teléfono del cuidador, para tener un teléfono al cual avisar en caso de necesidad.

4.3.13 Monitor

Como última actividad tenemos la actividad Monitor. A esta se accede a través de la opción monitor, dentro de la opción que hay en la actividad Panel principal. Esta actividad parece muy simple, pero en realidad es la que activa el funcionamiento esencial de la aplicación. Una vez hemos entrado en esta actividad se empiezan a recoger los datos del acelerómetro de manera que, a partir de ese momento, todos los datos se empiezan a guardar en un archivo txt que podrá ser enviado por correo desde dentro de la misma actividad.

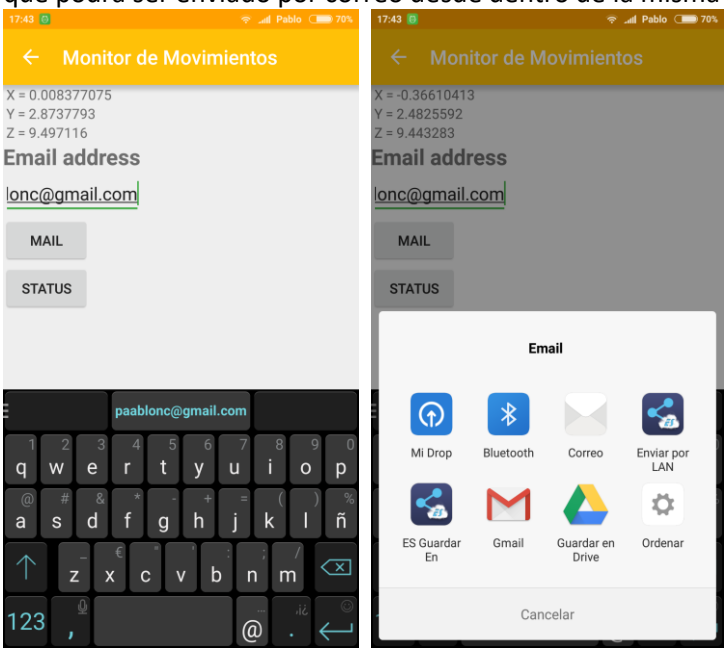


Figura 32 . Envío por correo

Como se puede observar, existe también la opción de consultar el estado del paciente, el cual se obtiene a través del botón Estado Paciente, el cual realiza una conexión al servicio web y muestra un mensaje con información del estado del paciente. Dependiendo del estado de este paciente (Leve, moderado o grave) el mensaje se mostrará de un color u otro (Verde, azul y rojo respectivamente).

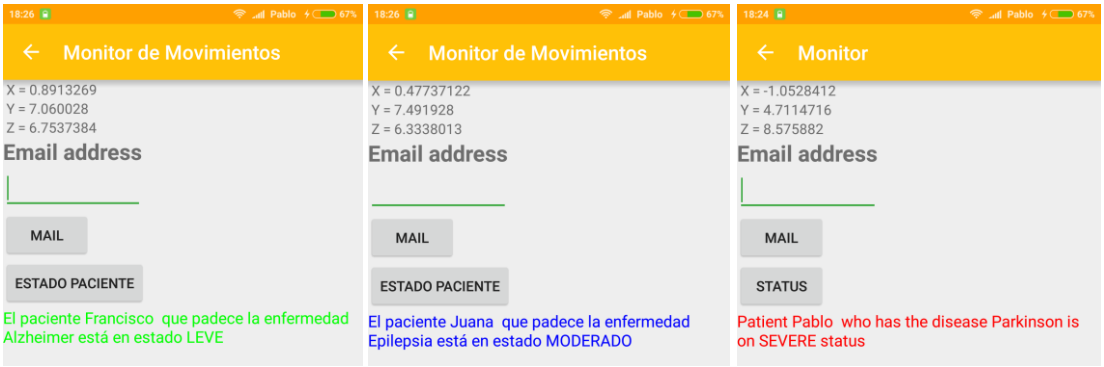


Figura 33 . Consulta del estado del paciente

5 Evaluación y pruebas

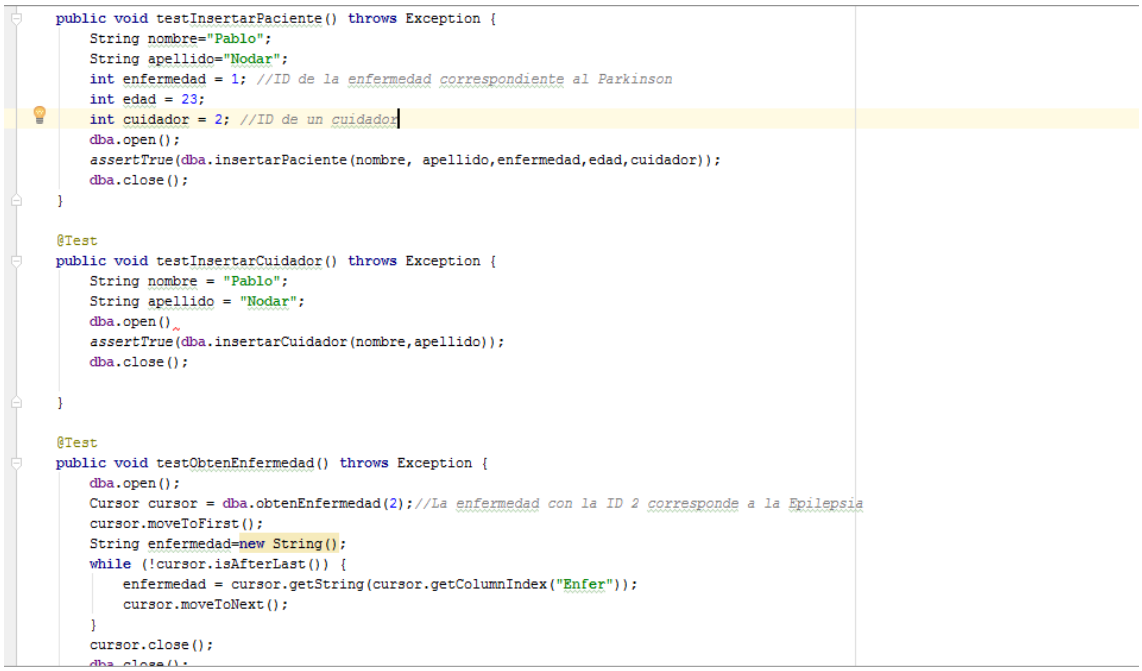
A la hora de desarrollar cualquier sistema software es necesario realizar una serie de pruebas, ya que constantemente surgen errores y fallos que deben ser abordados para que el sistema funcione correctamente y como se espera. Para ello es importante realizar una serie de pruebas como pueden ser las siguientes:

- Pruebas unitarias: Las pruebas unitarias son una forma común de realizar pruebas en programas, como los programas orientados a objetos. Estas pruebas son pequeños programas a los que se les da una entrada, y mediante asertos, se controlan los datos obtenidos [11].
- Pruebas de integración: Las pruebas de integración se utilizan una vez superadas con éxito las pruebas unitarias, probando todos los elementos unitarios juntos para buscar algún error no detectado con anterioridad [12].
- Pruebas de sistema: Las pruebas de sistema se realizan tras realizar con éxito las pruebas de integración. Estas pruebas se realizan para encontrar errores en el sistema a la hora de interactuar con su entorno. En este proyecto se ha realizado al probar el funcionamiento de la aplicación con, el que en este caso es su entorno, una base de datos local [12].
- Pruebas de validación: Las pruebas de validación son las últimas pruebas a realizar. Para ello se verifica que el sistema software cumple con los requisitos especificados. En este proyecto dichas pruebas se han realizado con la colaboración de personas a las cuales se les ha dado a probar la aplicación durante un periodo de tiempo [12]

5.1 *Pruebas realizadas en la aplicación*

De entre todas las pruebas que se han realizado en la aplicación, en este apartado se procederá a explicar las pruebas realizadas sobre la clase que conecta directamente con la base de datos, ya que el resto de clases contienen métodos que recogen los datos directamente de la aplicación y no tienen parámetros. Para ello se ha empleado la clase *JUnit* implementada en el framework *Android Studio*, lo cual ha facilitado el trabajo a la hora de hacer las pruebas de los métodos.

El procedimiento a seguir ha sido bastante simple, para crear la clase de prueba basta con estar en la clase que se desea probar y pulsar Ctrl+Shift+T dentro de la clase de prueba para que aparezca el menú en cual podremos crear directamente el test para dicha clase. Una vez creada la clase de prueba, lo que se ha hecho ha sido crear un atributo privado de la clase a probar, en nuestro caso *DbAdapter* y probar la clase en distintos métodos test, que prueban los métodos de la clase a probar por separado.



```
public void testInsertarPaciente() throws Exception {
    String nombre="Pablo";
    String apellido="Nodar";
    int enfermedad = 1; //ID de la enfermedad correspondiente al Parkinson
    int edad = 23;
    int cuidador = 2; //ID de un cuidador
    dba.open();
    assertTrue(dba.insertarPaciente(nombre, apellido,enfermedad,edad,cuidador));
    dba.close();
}

@Test
public void testInsertarCuidador() throws Exception {
    String nombre = "Pablo";
    String apellido = "Nodar";
    dba.open()
    assertTrue(dba.insertarCuidador(nombre,apellido));
    dba.close();

}

@Test
public void testObtenEnfermedad() throws Exception {
    dba.open();
    Cursor cursor = dba.obtenEnfermedad(2);//La enfermedad con la ID 2 corresponde a la Epilepsia
    cursor.moveToFirst();
    String enfermedad=new String();
    while (!cursor.isAfterLast()) {
        enfermedad = cursor.getString(cursor.getColumnIndex("Enfer"));
        cursor.moveToNext();
    }
    cursor.close();
    dba.close();
```

Figura 34 . Clase de prueba unitaria

Como se puede observar en la imagen anterior, las pruebas se realizan mediante las instrucciones *assertTrue()*. Dichas instrucciones se basan en un valor booleano, de manera que, si el resultado es *true*, el test saldrá bien, y si es false, el test fallará. Para los dos primeros métodos se probó a insertar un paciente y un cuidador, ambos sin ningún fallo, de manera que los resultados dieron positivo. Para el tercer método, se comprobó que la enfermedad con la *ID* 2 era la correcta, y también concluyó de manera exitosa.

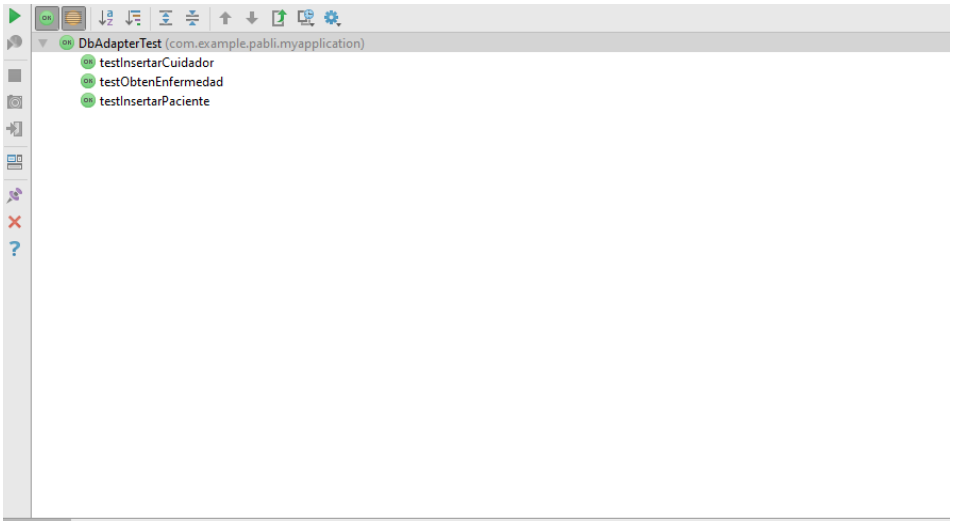


Figura 35 . Ejemplo de pruebas exitosas

Para comprobar que dichos métodos estuvieran correctamente, se modificó el primer método de manera que la id del cuidador que se iba a asociar al paciente no coincidiese con ningún cuidador registrado en el sistema.

```
@Test
public void testInsertarPaciente() throws Exception {
    String nombre="Pablo";
    String apellido="Nodar";
    int enfermedad = 1; //ID de la enfermedad correspondiente al Parkinson
    int edad = 23;
    int cuidador = 33; //ID de un cuidador que no existe
    dba.open();
    assertTrue(dba.insertarPaciente(nombre, apellido,enfermedad,edad,cuidador));
    dba.close();
}
```

Figura 36 . Método de prueba erróneo

Una vez hecho esto, se volvió a ejecutar el test, de manera que diera el resultado esperado: fallo en el primer método y éxito en los otros dos.

Figura 37 . Ejemplo de prueba fallida

Una vez realizado este proceso y ya con la aplicación desarrollada al completo y funcionando, se instaló la aplicación en dos dispositivos de dos personas distintas para comprobar que no hubiera fallos de diseño que el programador no fuera capaz de identificar. De este modo se identificaron ciertos problemas de interfaz, por ejemplo, los campos de ID cuidador e ID paciente estaban colocados de distinta forma en la interfaz en ingles con respecto a la interfaz en español. Otro problema que se identifico fue que, al registrar un paciente dentro de la interfaz en español, esta pasaba a la interfaz inglesa, cambiando por completo el flujo del sistema.

5.2 *Pruebas en el servicio web*

Para probar el correcto funcionamiento de todo el sistema, se hicieron también pruebas en el servicio web. Para ello se utilizó el framework *SoapUI*. Dentro de este framework se nos da la posibilidad de agregar la dirección en la que se aloja nuestro servicio web, y añadiendo los parámetros ver como este funciona. Un ejemplo de ello es la imagen que se muestra a continuación.

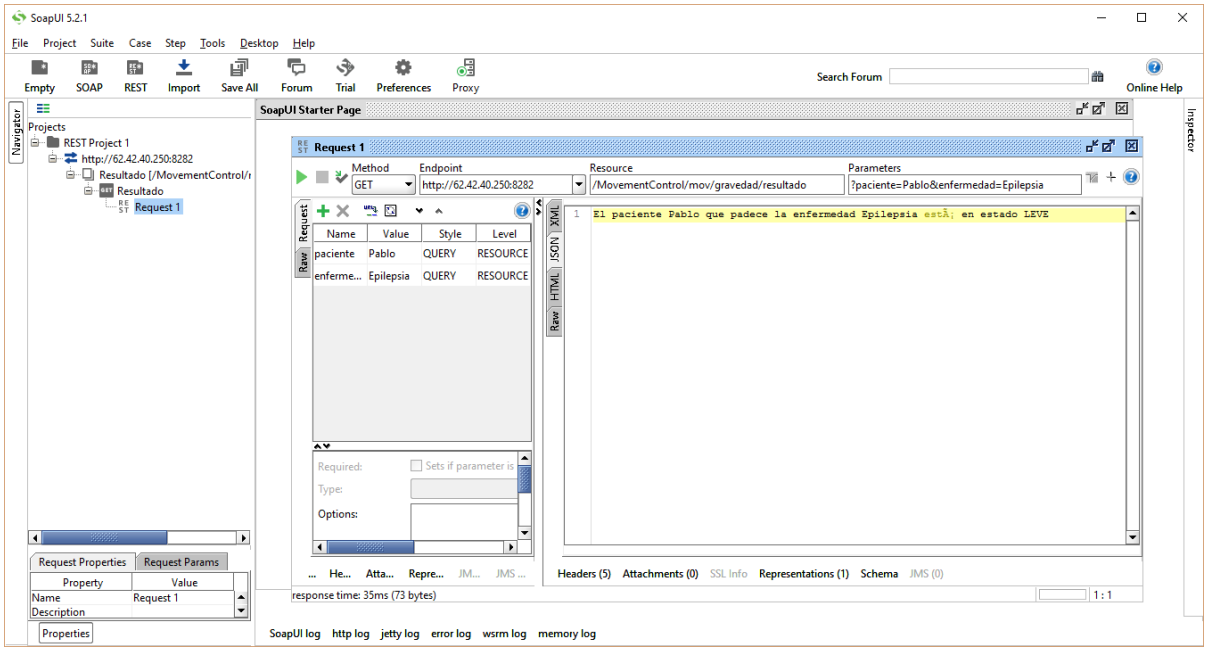


Figura 38 . Ejemplo prueba SoapUI

6 Conclusiones

Una vez terminado el proyecto es necesario saber si se han conseguido los objetivos establecidos desde un principio y como se han conseguido alcanzar estos. Además, es importante mencionar también las distintas funcionalidades que en un futuro se pueden realizar teniendo como base este proyecto.

6.1 Grado de consecución de objetivos

En el inicio de este proyecto se propusieron una serie de objetivos que debían ser cumplidos por el sistema a desarrollar. Para ello se establecieron subobjetivos relacionados al objetivo principal, para cumplir y complementar el objetivo. Dicho objetivo principal era el de crear una aplicación *Android* capaz de monitorizar los movimientos de personas con trastornos de movilidad.
A continuación, se mostrarán dichos objetivos y el desarrollo de los mismos:

I. Monitorizar los datos registrados por el acelerómetro del teléfono móvil del usuario.
- A. Visualizar gráficamente los datos recopilados mediante el acelerómetro.
- B. Almacenar los datos recopilados en un fichero.
- C. Enviar los datos recopilados por correo electrónico para su posterior estudio.

Este es el objetivo principal de la aplicación, y el porqué del desarrollo de la misma. La idea principal era captar de alguna manera los cambios en los movimientos de un paciente para poder recopilar información acerca de las distintas enfermedades con trastornos de movimiento.

Para ello se diseñó una aplicación que, utilizando el acelerómetro del terminal, fuera capaz de mostrar los cambios en los movimientos, guardar estos datos de alguna manera y una vez almacenados poder enviar dichos datos por correo para un futuro estudio de los mismos.

El primer paso fue recoger los datos del acelerómetro. Para ello, utilizando el framework *Android Studio* se utilizaron una serie de métodos capaces de escuchar los movimientos del acelerómetro, de esta manera al seleccionar la opción destinada a la monitorización se empezarían a recopilar los datos. Una vez con esto, lo siguiente fue ir actualizando la interfaz para que se vieran los cambios dentro de la aplicación y así poder visualizar dichos datos.

Una vez conseguido esto, la siguiente propuesta era la de almacenar toda la información dentro del dispositivo móvil. Para ello, se desarrolló una serie de métodos para poder generar un archivo de texto dentro del teléfono y poder escribir los datos recogidos y mostrados dentro de dicho archivo. Para ello fue necesario dar permisos sobre el almacenamiento del dispositivo, ya que de lo contrario el archivo se genera dentro de la memoria interna del terminal y quedaría inaccesible para el usuario, ya que la única forma de acceder seria mediante permisos de superusuario. Una vez proporcionados los permisos dentro del *Android Manifest*, se desarrolló el método capaz de generar el archivo y escribir dentro del mismo.

Para concluir con el último subobjetivo, se implementó una opción para poder enviar el archivo anteriormente generado por correo electrónico o similar. Dentro de la aplicación, se implementó un *EditText* de manera que el usuario podrá escribir el correo al que desea enviar el archivo y posteriormente seleccionar la opción de enviar para completar el proceso.

II. Tener un sistema capaz de almacenar en base de datos distintos datos en un terminal.

Para poder cumplir este objetivo era imprescindible desarrollar una base de datos. Una de las principales dudas que surgieron a la hora de desarrollar el sistema de almacenamiento fue la elección de un software para ello. De entre todas las opciones, finalmente se decidió emplear *SQLite*. Se eligió este software por su simplicidad y por la facilidad de manejo dentro de la aplicación Android, así como la posibilidad de tener la base de datos de manera local en un archivo dentro del terminal.

Una vez implementada la base de datos, se pudo almacenar con facilidad todos los datos y además permitió que se pudieran añadir más datos desde dentro de la propia aplicación, así como consultar dichos datos. Para ello se desarrollaron distintos métodos y clases que permitieron la conexión con la base de datos sin necesidad de añadir ninguna extensión al framework, ya que *Android Studio* tiene implementadas una serie de funcionalidades que permiten la conexión con las bases de datos del tipo *SQLite.*

III. Consultar el estado de la enfermedad de un paciente para saber cómo avanza la enfermedad del mismo.

Para cumplir este objetivo se implementó el árbol de decisión y se desarrolló un servicio web en *java,* utilizando el framework *Anypoint Studio* y hosteando este servicio en un ordenador mediante el uso del software *Apache Tomcat*. Este servicio web, espera que se le pasen una serie de parámetros, mediante una petición de tipo *SOAP*. Para ello, mientras el dispositivo lee los datos del acelerómetro, se van almacenando los valores máximos, mínimos y la media para cada uno de los ejes que se detectan. De este modo se obtienen los valores necesarios para poder realizar la conexión con el servicio web y este realizará toda la lógica interna implementando el árbol de decisión generado. La respuesta del servicio web será en formato *XML* que posteriormente será tratada por la aplicación Android para mostrar el estado en el que se encuentra el paciente.

6.2 Trabajos futuros

Como experiencia personal, este proyecto me ha sido de gran ayuda, ya que he aprendido a desarrollar aplicaciones en lenguaje Android, cosa que nunca antes había hecho, he aprendido a desarrollar servicios web y he puesto en práctica los conocimientos adquiridos durante la carrera, como por ejemplo la conexión de una aplicación con una base de datos y el diseño e implementación de un proyecto. De cara al futuro este trabajo me da cierta experiencia para poder entender un poco como sería la realización de un proyecto en el mundo laboral.

Por otro lado, este proyecto abre la posibilidad a nuevas funcionalidades que serían de cierto interés para un mejor uso de la aplicación. Entre ellas podemos destacar las siguientes:

- Utilizando el teléfono de contacto del paciente, enviar un mensaje de alerta cuando un paciente sufra una crisis
- Almacenar los datos recogidos en la nube, por ejemplo, *Dropbox*, para facilitar el acceso a dichos datos.

Por ultimo destacar la posibilidad de adaptar la aplicación a otros sistemas operativos comúnmente utilizados por dispositivos móviles, como podrían ser *IOS* o *Windows Phone*.

7 Bibliografía

[1] http://www.livescience.com/40102-accelerometers.html - Ryan Goodrich. Accelerometers: What They Are & How They Work. Última visita 06/09/2016

[2] http://developer.android.com/intl/es/tools/studio/index.html - Página oficial *Android* para desarrolladores. Última visita 28/02/2016

[3] https://www.sqlite.org/about.html - Última visita 29/2/2016 - Página oficial *SQLite.* Última visita 12/06/2016

[4] http://www.fabforce.net/dbdesigner4/ - Página oficial del software *DBDesigner*. Última visita 12/06/2016

[5] http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML - W3C. Guía breve de tecnologías XML. Última visita 29/2/2016

[6]http://tomcat.apache.org/ - Página oficial *Apache Tomcat*. Última visita 30/08/2016

[7] https://www.soapui.org/open-source.html - Página oficial *SoapUI*. What is SoapUI. Última visita 31/08/2016

[8] https://eclipse.org/home/index.php - Página oficial *Eclipse*. What is Eclipse. Última visita 31/08/2016

[9] http://secyt.unpa.edu.ar/journal/index.php/ICTUNPA/article/view/71 - Juan Gabriel Enriquez, Sandra Isabel Casas. Usabilidad en aplicaciones móviles. Última visita 07/06/2016

[10] https://www.w3.org/2002/Talks/0104-usabilityprocess/slide3-0.html - W3C. Usability - ISO 9241 definition. Última visita 12/06/2016

[11] https://www.google.com/patents/US7587636 - Nikolai Tillmann, Wolfgang Grieskamp, Wolfram Schulte. Unit test generalization

[12] Macario, P. U; Beatriz, P. L.; Pedro, R. M. (2012). Técnicas combinatorias y de mutación para testing de sistemas software.

[13] https://www.soapui.org/soapui/media/images/stories/homepage/soapui_logo.png - Logo SoapUI. Última visita 31/08/2016

[14] http://osc.co.cr/wp-content/uploads/2011/06/incremental.jpeg - Imagen metodología iterativa e incremental. Última visita 26/02/2016

[15] http://2.bp.blogspot.com/-bnVPkcNuXPU/VLQ2-0a_VFI/AAAAAAAAA88/Yowntg2qEls/s1600/programaci%C3%B3n-3-capas-incanatoit.PNG - Imagen arquitectura en tres capas. Última visita 25/06/2016

[16] Detecting Human Movement Patterns through Data Provided by Accelerometers. A case study regarding Alzheimer´s desease – Departamento de Matemáticas, Estadística y Computación. – Rafael Duque, Alicia Nieto-Reyes, Carlos Martínez y José Luis Montaña

[17] https://docs.mulesoft.com/anypoint-studio/v/6/ - Página oficial *Anypoint Studio, MuleSoft*. Última visita 25/6/2017.