



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

MedsScheduler: Aplicación móvil para gestión de pautas
de medicamentos

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Bermejo Canet, José Ramón

Tutor/a: Andrés Martínez, David de

CURSO ACADÉMICO: 2023/2024

Resumen

Un problema común entre muchas personas, especialmente las de tercera edad, es recordar cuándo les corresponde tomar la dosis de un medicamento determinado. Es frecuente que se olvide tomar la dosis indicada y, por consiguiente, la salud del paciente se ve perjudicada de algún modo. Por otro lado, el perfil de paciente que más medicamentos consume es el de una persona mayor, ya que, debido a su longevidad, su salud se ve resentida. Además, es muy frecuente en este rango de edad encontrar problemas como la pérdida de visión, de memoria y de facultades cognitivas que dificultan el correcto seguimiento de las pautas indicadas. Es por ello que este trabajo está enfocado a remediar este problema mediante una aplicación móvil. Se trata de una aplicación cuyo objetivo es permitir a los usuarios establecer alarmas según las pautas de medicamentos que tiene recetados por su médico. Así, el paciente no debe preocuparse por si le falla la memoria, ya que, al establecer una alarma en su teléfono, será éste quien le recuerde la toma de la medicina. Asimismo, se ofrece una serie de opciones para mejorar la accesibilidad como por ejemplo aumentar el tamaño de la letra de la aplicación para facilitar el visionado de los contenidos. Por otro lado, la aplicación ofrece un sistema de roles paciente/médico, el cual permite gestión de recetas y consulta de citas entre otros. Además, se ofrecen consejos para fomentar una vida saludable entre todos los usuarios de la aplicación. Para el desarrollo de este proyecto se usa *Kotlin* como lenguaje de programación principal y tecnologías más sofisticadas como *Firebase*, además de la documentación y las guías de desarrollo en *Android* donde se detallan las prácticas más recomendadas por Google.

Palabras clave: Pauta; Receta; Medicamento; Médico; Salud; Notificación; Firebase; Android; Smartphone; Móvil; Seguimiento; Kotlin;

Abstract

A common problem among many people, especially the elderly, is remembering when they are due to take the dose of a certain medication. It is common to forget to take the indicated dose and, consequently, the patient's health is harmed in some way. In addition, the profile of the patient who consumes the most medications is that of an elderly person, since, due to their longevity, their health suffers. Furthermore, it is very common in this age range to encounter problems such as loss of vision, memory and cognitive abilities that make it difficult to follow the indicated guidelines correctly. That's why this study is focused on solving this problem through a mobile application. This is an application whose objective is to allow users to set alarms according to the medication guidelines prescribed by their doctor. Thus, the patient does not have to worry about his memory failing, since, by setting an alarm on his phone, it will remind him to take the medicine. Likewise, a series of options are offered to improve accessibility, such as increasing the font size of the application to facilitate viewing the content. Moreover, the application offers a patient/doctor role system, which allows prescription management and appointment consultation, among others. In addition, tips are offered to promote a healthy lifestyle among all users. For the development of this project, Kotlin is used as the main programming language and more sophisticated technologies such as *Firebase*, in addition to documentation and *Android* development guides that detail the most recommended practices by Google.

Key words: Instruction; Prescription; Medicine; Doctor; Health; Notification; Firebase; Android; Smartphone; Mobile; Tracking; Kotlin;

Resum

Un problema comú entre moltes persones, especialment en les de tercera edat, és recordar quan els correspon prendre la dosi d'un medicament determinat. És freqüent que s'oblidi prendre la dosi indicada i que després la salut del pacient es veja perjudicada d'alguna forma. És per això que este treball està enfocat a oferir un remei mitjançant una aplicació mòbil. Es tracta d'una aplicació que té com a objectiu permetre als usuaris establir alarmes d'acord amb pautes de medicació que tenen receptades pels seus metges. Així, el pacient no s'ha de preocupar per si falla la seua memòria, ja que establint una alarma al seu telèfon, serà este qui li recorde que s'ha de prendre la medicina. Per altre costat, l'aplicació oferix un sistema de rols pacient/metge, el qual permet la gestió de receptes i consulta de cites entre altres. A més, s'oferixen consells per a promoure un estil de vida saludable entre tots els usuaris de l'aplicació. Per al desenvolupament d'este projecte s'utilitza *Kotlin* com a llenguatge de programació principal i tecnologies més sofisticades com *Firebase*, a més de la documentació i les guies de desenvolupament en *Android* on es descriuen les pràctiques més recomanades per Google.

Paraules clau: Pauta; Recepta; Medicament; Metge; Salut; Notificació; Firebase; Android; Smartphone; Mòbil; Seguiment; Kotlin;

Índices y glosarios

Tabla de contenido

Resumen	1
Abstract.....	1
Resum	2
Índices y glosarios	3
Índice de figuras.....	5
Índice de tablas.....	7
1. Introducción.....	8
1.1. Motivación	8
1.2. Objetivos	8
1.3. Impacto esperado.....	9
1.4. Metodología.....	9
2. Estado de la cuestión	12
2.1. Descripción de aplicaciones similares	12
2.1.1. Medisafe.....	12
2.1.2. MyTherapy	16
2.1.3. TOM	20
2.2. Análisis comparativo de las aplicaciones estudiadas	27
2.3. Conclusiones.....	29
3. Análisis del problema	30
3.1. Especificación de requisitos	30
3.2. Tabla “MoSCoW”	31
3.3. Modelo de dominio	32
3.4. Diagrama de casos de uso.....	34
3.5. Análisis de la seguridad.....	37
3.6. Análisis del marco legal y ético	38
3.6.1. Análisis de la protección de datos.....	38
3.6.2. Propiedad intelectual.....	39
3.7. Análisis de riesgos.....	39
3.8. Identificación y análisis de soluciones posibles	40
3.8.1. <i>AlarmManager</i> vs. <i>Cloud Messaging</i>	40
3.8.2. <i>Cloud Firestore</i> vs. <i>Realtime Database</i>	41
3.9. Solución propuesta.....	42
3.10. Plan de trabajo	43
3.10.1. Sprint 0.....	44
3.10.2. Sprint 1.....	44
3.10.3. Sprint 2.....	45
3.10.4. Sprint 3.....	45
3.10.5. Sprint 4.....	46
3.10.6. Sprint 5.....	46
4. Diseño de la solución	47

4.1.	Arquitectura del sistema	47
4.2.	Capa de presentación	51
4.3.	Diseño detallado.....	60
4.4.	Tecnología utilizada.....	63
4.4.1.	Android Studio.....	63
4.4.2.	Kotlin	64
4.4.3.	Firebase	65
4.4.3.1.	Firebase Authentication	65
4.4.3.2.	Cloud Firestore.....	66
4.5.	ML Kit Translator	67
4.6.	Hilt.....	67
4.7.	Espresso	68
4.8.	JUnit	69
4.9.	Mantenimiento y Gestión de Versiones	69
5.	Desarrollo de la solución propuesta	71
5.1.	Implementación de la creación de alarmas	71
5.2.	Modelo de calidad empleado.....	73
5.3.	Heurísticas de Nielsen.....	74
5.4.	Refactoring y análisis estático del código fuente.....	75
6.	Pruebas.....	76
6.1.	Pruebas unitarias	76
6.2.	Pruebas de integración.....	76
6.3.	Pruebas de usabilidad	77
6.4.	Pruebas de accesibilidad.....	79
6.5.	Pruebas de regresión	81
7.	Resultados obtenidos.....	82
8.	Conclusiones.....	89
8.1.	Relación del trabajo desarrollado con los estudios cursados	90
8.2.	Trabajos futuros.....	91
9.	Referencias	92
10.	Anexos	95
10.1.	Reflexión sobre la relación del TFG con los Objetivos de Desarrollo Sostenible (ODS)	95
10.2.	Glosario	96
10.3.	Descripción detallada de los casos de uso.....	97
10.4.	Interfaces de usuario de la aplicación	112

Índice de figuras

Figura 1.- Ilustración gráfica de un Product Backlog.....	11
Figura 2.- Logotipo de Medisafe	12
Figura 3.- Recopilación inicial de: a) nombre y apellidos del usuario, b) género del usuario, c) edad del usuario	13
Figura 4.- Recopilación de datos de la primera alarma como: a) nombre del medicamento, b) frecuencia de administración, c) hora de la dosis.....	14
Figura 5.- Interfaz de selección de datos opcionales a agregar en Medisafe	15
Figura 6.- Menú principal de Medisafe	15
Figura 7.- Logotipo de MyTherapy	16
Figura 8.- Creación de alarma en MyTherapy con las siguientes interfaces: a) selección de medicamento, b) selección de frecuencia de administración, c) selección de dosis y hora de administración del medicamento	17
Figura 9.- Interfaz de recordatorio de relleno de existencias en MyTherapy	18
Figura 10.- Interfaz de indicación del motivo en MyTherapy.....	18
Figura 11.- Menú principal de MyTherapy.....	19
Figura 12.- Ejemplo de notificación de dosis en MyTherapy.....	19
Figura 13.- Logotipo de TOM	20
Figura 14.- Interfaz de adición de medicamentos en TOM	21
Figura 15.- Creación de alarma en TOM, donde: a) se selecciona un medicamento, b) se selecciona una frecuencia de administración, c) indicación de duración del tratamiento, d) selección de hora y dosis de notificación.....	23
Figura 16.- Finalización de la creación de alarma en la que: a) se ofrece mantener un control de las unidades restantes del medicamento, b) asignar observación de la administración en cuanto a las comidas diarias, c) resumen de la información de la alar	23
Figura 17.- Interfaz de adición de fotografía del envase del medicamento en TOM	24
Figura 18.- Menú principal de TOM.....	25
Figura 19.- Notificación de toma de medicamento en TOM	25
Figura 20.- Interfaz de dosis pendientes para el día actual en TOM.....	26
Figura 21.- Interfaz de históricos de pautas en TOM	26
Figura 22.- Diagrama que representa el modelo de dominio del problema a resolver.....	33
Figura 23.- Jerarquía de los actores presentes en la aplicación	35
Figura 24.- Casos de uso del usuario no autenticado en la aplicación	36
Figura 25.- Casos de uso del paciente en la aplicación.....	36
Figura 26.- Casos de uso del médico en la aplicación.....	37
Figura 27.- Diagrama de la arquitectura de la aplicación. Imagen adaptada de “Guía de arquitectura de apps – Arquitectura de app recomendada” de la página web Android Developers	48
Figura 28.- Elementos contenidos dentro de la capa de presentación de la arquitectura. Imagen adaptada de “Arquitectura de app recomendada – Capa de la IU” de la página web Android Developers	49
Figura 29.- Diagrama que muestra el funcionamiento del flujo unidireccional de datos en la arquitectura de la aplicación. Imagen adaptada del sitio "Capa de la IU - Contenedores de estado" de la web Android Developers.....	49
Figura 30.- Capa de dominio en la arquitectura de la aplicación. . Imagen adaptada de “Arquitectura de app recomendada – Capa de dominio” de la página web Android Developers.....	50
Figura 31.- Repositorios y fuentes de datos en la capa de persistencia de la arquitectura. Imagen adaptada de “Arquitectura de app recomendada – Capa de datos” de la página web Android Developers.....	51
Figura 32.- Bocetos de las interfaces de usuario: a) Inicio de sesión b) Cierre de sesión y c) Registro de nuevos usuarios.....	52
Figura 33.- Bocetos de las interfaces de usuario: a) Consejos de salud b) Recordar pautas de medicamentos c) Localización de farmacias cercanas - Mapa y d) Localización de farmacias cercanas - Detalle de ubicación.....	54
Figura 34.- Bocetos de las interfaces de usuario: a) Selección de dosis b) Selección de frecuencia y c) Selección de hora de inicio	55
Figura 35.- Bocetos de las interfaces de usuario: a) Indicación de cumplimiento de pauta b) Registro de medicamentos c) Reservar citas médicas d) Selección de tipo de dosis	57
Figura 36.- Bocetos de las interfaces de usuario: a) Roles paciente/médico - Interfaz de paciente b) Roles paciente/médico - Interfaz de médico y c) Interacciones medicamentosas	58
Figura 37.- Mapa de navegación entre las interfaces de usuario	59
Figura 38.- Logotipo de MedsScheduler	59
Figura 39.- Estructura de directorios en los que se organiza el código fuente de este proyecto software..	60
Figura 40.- Ejemplo de implementación en el paquete data	61
Figura 41.- Contenido del paquete di.....	61
Figura 42.- Contenido del paquete domain.model	62
Figura 43.- Contenido del paquete receivers	62

Figura 44.- Paquetes contenidos dentro del paquete ui, donde se muestra un ejemplo de Fragment y ViewModel	62
Figura 45.- Contenido del paquete utils	63
Figura 46.- Logotipo de Android Studio.....	63
Figura 47.- Logotipo de Kotlin	64
Figura 48.- Logotipo de Firebase	65
Figura 49.- Logotipo de ML Kit.....	67
Figura 50.- Logotipo del framework Espresso	68
Figura 51.- Logotipo de Git	70
Figura 52.- Primera fase del proceso de creación de una alarma.....	72
Figura 53.- Segunda fase del proceso de creación de alarmas	73
Figura 54.- Icono de la aplicación "Test de Accesibilidad"	79
Figura 55.- Funcionamiento de la aplicación "Test de Accesibilidad" en la que: a) Se resalta el elemento cuya accesibilidad podría mejorarse, b) Se muestra el detalle de la sugerencia en cuestión.....	80
Figura 56.- Funcionalidades de la aplicación: a) Registro de nuevos usuarios, b) Selección de médico, c) Cierre de sesión, d) Inicio de sesión.....	85
Figura 57.- Funcionalidades de la aplicación: a) Consejo de salud, b) Selección de medicamento, c) Selección de presentación del medicamento, d) Selección de dosis, e) Selección de hora y fecha de inicio y f) Notificación de la alarma.....	86
Figura 58.- Funcionalidades de la aplicación: a) Listado de alarmas, b) Detalle de alarma, c) Selección de fecha, d) Selección de hora, e) Listado de citas médicas	87
Figura 59.- Funcionalidades de la aplicación: a) Diálogo de otorgación de permisos de localización y b) Muestra de datos de una farmacia.....	87
Figura 60.- Funcionalidades de la aplicación: a) Sección de medicinas y b) Formulario de registro de nuevas medicinas	88
Figura 61.- Logotipo de la Agenda 2030	95
Figura 62.- Interfaces de usuario de la aplicación: a) listado de alarmas establecidas, b) Listado de citas médicas, c) Localización de farmacias cercanas.....	113
Figura 63.- Interfaces de usuario de la aplicación: a) Selección de medicamento, b) Selección de la presentación del medicamento, c) Selección de la dosis, d) Selección de la hora de inicio y la frecuencia	114
Figura 64.- Interfaces de usuario de la aplicación a) consejo de salud, b) detalle de una alarma, c) selección de fecha de cita médica, d) selección de hora de cita médica	115
Figura 65.- Interfaces de usuario de la aplicación: a) inicio de sesión, b) formulario de registro, c) selección de médico, d) formulario de registro de nuevos medicamentos.....	116

Índice de tablas

Tabla 1.- Tabla comparativa de las alternativas analizadas	27
Tabla 2.- Tabla de priorización de requisitos mediante MoSCoW	31
Tabla 3.- Resumen de características de Cloud Firestore y Realtime Database	41
Tabla 4.- Planificación de los sprints que conforman el proyecto.....	43
Tabla 5.- Colores usados en los temas claro y oscuro de la aplicación	60
Tabla 6.- Resultados del cuestionario de usabilidad de la aplicación.....	77
Tabla 7.- Resumen de las pruebas realizadas para evaluar la calidad del software y los resultados obtenidos	82
Tabla 8.- Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).....	95
Tabla 9.- CU_01 - Inicio de sesión con usuario y contraseña	97
Tabla 10.- CU_02 - Cierre de sesión.....	98
Tabla 11.- CU_03 - Listar alarmas establecidas.....	98
Tabla 12.- CU_04 - Registrarse.....	98
Tabla 13.- CU_05 - Visualizar consejo de salud.....	99
Tabla 14.- CU_06 - Borrar alarma	100
Tabla 15.- CU_07 - Crear alerta de pauta	100
Tabla 16.- CU_08 - Asignar medicamento	101
Tabla 17.- CU_09 - Asignar dosis.....	102
Tabla 18.- CU_10 - Asignar periodo de pauta	102
Tabla 19.- CU_11 - Reservar cita	102
Tabla 20.- CU_12 - Comprobar farmacias cercanas.....	103
Tabla 21.- CU_13 - Registrar nuevo medicamento	104
Tabla 22.- CU_14 - Seleccionar idioma de uso	104
Tabla 23.- CU_15 - Seleccionar médico.....	105
Tabla 24.- CU_16 - Consultar citas	105
Tabla 25.- CU_17 - Seleccionar fecha y hora	106
Tabla 26.- CU_18 - Cancelar cita	106
Tabla 27.- CU_19 - Indicar efectos secundarios	107
Tabla 28.- CU_20 - Establecer nombre del medicamento.....	107
Tabla 29.- CU_21 - Cancelar cita	108
Tabla 30.- CU_22 - Registrar síntoma.....	108
Tabla 31.-CU_23 - Visualizar artículo sobre salud	109
Tabla 32.- CU_24 - Indicar prescripción médica necesaria	109
Tabla 33.- CU_25 - Escanear código de barras	110
Tabla 34.- CU_26 - Crear receta	110
Tabla 35.- CU_27 - Asignar medicamento a paciente	111
Tabla 36.- CU_28 - Comprobar disponibilidad	111

1. Introducción

1.1. Motivación

En la actualidad, cada vez es más común encontrar personas de edad avanzada que consumen algún tipo de medicamento. Para ser más precisos, datos del año 2016 indican que el 88,9% de las personas mayores de 65 años en España consume algún medicamento, yéndose incluso al 93,4% en personas mayores de 75 años. Por otro lado, el 94% de las personas que padecen algún tipo de enfermedad crónicas están polimedicadas [1]. Esto añade complejidad al manejo terapéutico, con una mayor dificultad para un correcto cumplimiento de la terapia, con un riesgo mayor de aparición de efectos secundarios e interacciones entre diferentes principios activos y con posibilidades más altas de errores en la toma o administración de los medicamentos previamente prescritos. Además, en 2022 el número de personas mayores de 65 años en España es de nueve millones, una cifra que representa casi un 20% de la población [2]. Todo esto también se corrobora con el siguiente estudio en el que se indica que casi el 90% de las personas mayores toman al menos un fármaco recetado, casi el 80% toma al menos dos regularmente y un 36% que toma cinco medicamentos prescritos distintos [3]. Es aquí donde entra el mundo de la tecnología para ayudar a las personas que deben seguir unas pautas en la medicación a conseguirlo de forma adecuada.

A continuación, se motiva la realización de este proyecto desde un lado más personal. Desde hace mucho tiempo, el síndrome de *Alport* ha sido un tema de conversación muy recurrente dentro de mi propio ámbito familiar. Se trata de una enfermedad genética que afecta en mayor medida al correcto funcionamiento de los riñones, el oído y la vista entre otros [4]. Muchos de los pacientes que la padecen ven condicionada su vida diaria ya que deben someterse a controles médicos constantemente, cuidar su alimentación, administrarse numerosos medicamentos e incluso a someterse a procesos de diálisis.

Sumado a esto, es muy común ver a personas de mi entorno con esta enfermedad olvidándose de tomar la dosis de un medicamento determinado cuando les corresponde. Todo esto es lo que me ha motivado a afrontar el desarrollo de una solución orientada no solo a pacientes que padecen esta enfermedad, sino también a un público más general, incluyendo a las personas de edad avanzada.

En definitiva, dados los datos mostrados anteriormente y lo visto a lo largo de muchos años en mi círculo más cercano, son las razones por las que he decidido afrontar este desafío y así ayudar al mayor número de personas posibles a facilitarles el cuidado de su salud.

1.2. Objetivos

Los objetivos que se pretenden alcanzar con este trabajo son los siguientes:

- Facilitar la comunicación entre pacientes y personal médico a través de un sistema común.
- Proporcionar un sistema de recordatorios de administración de medicamentos.
- Asegurar la facilidad de uso del sistema mediante opciones de accesibilidad.
- Fomentar un estilo de vida saludable entre los usuarios.

Para ello, se va a desarrollar una aplicación para dispositivos móviles que funcionen sobre el sistema operativo Android que permita al usuario gestionar sus propias alarmas de toma de medicamentos, mantener una interacción directa con profesionales de la salud y adoptar un estilo de vida saludable mediante la aplicación de los consejos que se proporcionen en dicho sistema.

1.3. Impacto esperado

La aplicación que se propone pretende ayudar al usuario final en las facetas descritas a continuación.

En primer lugar, el usuario contará con una solución que le recordará cuando le corresponde tomar una dosis de algún medicamento determinado, y a su vez, llevar un control de las dosis que lleva cumplimentadas. Este control ayudará al usuario a la hora de mantener su salud en buen estado, ya que será menos probable el hecho de olvidar la toma de alguna de estas dosis o tomar varias dosis de forma involuntaria.

Por otro lado, el hecho de añadir funcionalidades enfocadas a ayudar a los usuarios a implantar un estilo de vida saludable también está muy ligado al objetivo de ayudar al usuario a mantenerse sano, tal y como se ha comentado anteriormente.

Por último, se pretende que la aplicación sea de fácil uso para cualquier usuario, de tal modo que la aplicación sea utilizada incluso para personas de edad avanzada que puedan experimentar cierto grado de deterioro cognitivo.

Todas estas ventajas están directamente relacionadas con el Objetivo de Desarrollo Sostenible nº 3, relacionado con la salud y el bienestar de toda la ciudadanía.

1.4. Metodología

La metodología utilizada para este proyecto es conocida como **Scrum** y se trata de una metodología ágil.

El objetivo de las metodologías ágiles es proporcionar cada corto período de tiempo una versión incrementada del software al cliente para mejorar su satisfacción. Estas metodologías toman la flexibilidad y el trabajo en equipo como bases para ofrecer una mejora constante del producto. Por lo general, el desarrollo ágil implica el equipo de desarrollo autoorganizado y los representantes de la empresa se reúnen continuamente durante el ciclo de vida del software. La metodología ágil permite al equipo de desarrollo adaptarse a los posibles cambios o imprevistos que puedan surgir a lo largo del ciclo de vida [5].

Todas estas metodologías están construidas de acuerdo con el manifiesto ágil, publicado en 2001 [6]. Se trata de un conjunto de ideas que describen una nueva metodología de trabajo, con el objeto de reemplazar el enfoque rígido de los proyectos convencionales. Esta forma de desarrollar proyectos (software u otros) consta de cuatro puntos:

- **Valorar a los miembros del equipo y sus respectivas iteraciones por delante de los procesos y herramientas empleados:** A pesar de que todos los recursos utilizados para desarrollar un trabajo son importantes, nada es capaz de sustituir a las personas, a las que hay que poner en primer plano. No obstante, esto no significa que haya que trabajar sin la ayuda de procesos.
- **Valorar el software por encima de la documentación:** Se ha estado valorando equitativamente tanto el hecho de documentar el trabajo como el MVP. Es un gran error restar atención al producto en cuestión. Todo lo demás es secundario. Esto no significa abandonar completamente la documentación. Se propone documentar solo lo imprescindible y paralelamente a la construcción del producto.

- **Valorar la colaboración con los *stakeholders* por encima de la negociación de un contrato:** La forma más productiva de sacar adelante un trabajo es estableciendo un consenso con quien lo encarga. Sin embargo, la atención se ha centrado en la formación de un contrato que solamente puede servir para enfrentar al cliente y al equipo, cuando realmente tienen intereses comunes. No obstante, esto no significa que no se deba establecer unos puntos en común para alcanzar el entendimiento entre ambas partes.
- **Estimar la capacidad de cambio y adaptación por delante de cualquier estrategia:** Se trata de aceptar la incertidumbre como una parte básica e indispensable del proyecto, por lo que la capacidad de adaptarse y la flexibilidad ante los posibles cambios se convierten en virtudes. El seguimiento riguroso de una hoja de ruta conduce al fracaso si no se puede modificar el rumbo ante los inevitables cambios que van surgiendo. Pero esto no significa que se haya que trabajar con la ausencia de un plan.

De acuerdo con los principios del manifiesto ágil descritos anteriormente, Scrum emplea un enfoque mediante iteraciones e incremental con el objeto de controlar el riesgo y mejorar la capacidad de previsión. Además, involucra personas que disponen de todas las habilidades necesarias para hacer el trabajo o adquirir tales habilidades según sea necesario [7].

A lo largo de un proyecto en el que se aplica una metodología *Scrum*, los incrementos del producto se van sucediendo a través de *sprints*. Un *sprint* es un período de tiempo de longitud fija, de un mes aproximadamente, en el que se trata de implementar todos los aspectos acordados para el *sprint actual* y de alcanzar los objetivos propuestos para éste. Antes del comienzo de cada *sprint*, el equipo de desarrollo y el *product owner* se reúnen para acordar qué elementos de trabajo pendiente se van a realizar durante su transcurso. Esta reunión se conoce como *sprint planning*. Antes de finalizar cada *Sprint*, se realiza el *sprint review*, una reunión en la cual tanto todos los miembros del equipo se reúnen para inspeccionar el resultado del *Sprint* que está a punto de acabar y determinar futuras adaptaciones o mejoras. Por último, el equipo se reúne en una *Sprint Retrospective* (Retrospectiva del *Sprint*), en la que se inspecciona el desempeño del equipo a lo largo del *sprint* anterior y se planifican formas para aumentar la calidad y la eficacia.

El trabajo pendiente del producto se ordena según la prioridad que el *product owner* y el resto del equipo pactan previamente y se almacena todo en el *product backlog* (pila del producto). Es la única fuente de trabajo que el equipo de *Scrum* va a gestionar. Todos y cada uno de los elementos contenidos se consideran listos para su selección en un *sprint planning*. Por último, cabe destacar que su contenido puede variar y ser modificado a lo largo del transcurso del proyecto, ya que todos sus elementos compiten por ser seleccionados para los *sprints* siguientes, tal como se muestra en la Figura 1.

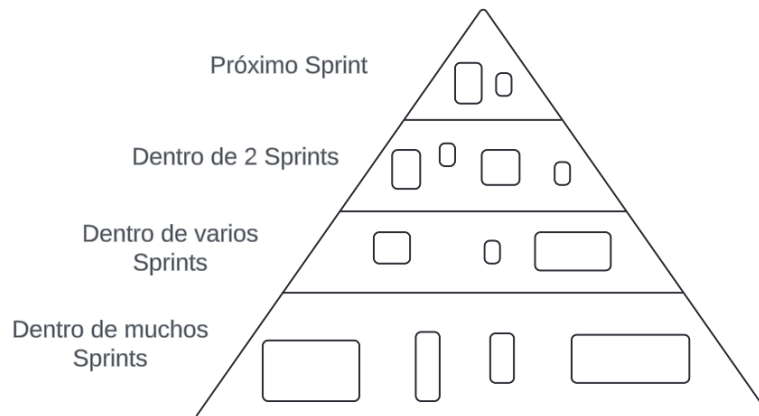


Figura 1.- Ilustración gráfica de un Product Backlog

En un equipo *Scrum*, tal como se ha mencionado anteriormente, existen varios perfiles de componente (todos ellos necesarios) que desempeñan funciones distintas dentro del equipo. A continuación, se describen los roles presentes en un equipo *Scrum* [7]:

- **Scrum Master:** Su principal responsabilidad es establecer la metodología tal como se define en las guías de *Scrum* y de lograr la efectividad del equipo. Para ello, ayuda a todos y cada uno de los integrantes del equipo a comprender la teoría y la práctica del método y a mejorar su desempeño dentro del marco de *Scrum*.
- **Product Owner (Propietario del producto):** Se encarga de maximizar el valor del producto resultante del proceso de desarrollo. Por otro lado, también es el encargado de la gestión eficaz del *product backlog*. También puede representar en el trabajo pendiente del producto las necesidades de un gran número de personas o departamentos afectados por el resultado del proyecto.
- **Desarrollador:** Son aquellas personas que se comprometen a generar un incremento del producto en cada *sprint*. Cabe destacar que dentro del equipo no existe ningún tipo de jerarquía y que todos y cada uno de los integrantes tiene todas las capacidades necesarias para crear valor en todos y cada uno de los *sprints realizados*.

En definitiva, se opta por una metodología ágil, en concreto por *Scrum*, ya que ofrece la posibilidad de obtener un incremento del producto por cada *Sprint* realizado a lo largo del proyecto y permite responder rápidamente a cambios en la especificación de requisitos del producto final y a posibles imprevistos que puedan surgir durante su desarrollo. Por consiguiente, se podrá obtener un MVP al final del proyecto.

2. Estado de la cuestión

Tal como se ha descrito en la sección 1.2, las características principales que se busca en una aplicación de este tipo son:

- Recordar pautas de medicamentos.
- Establecer pautas de medicamentos.
- Establecer dosis a pautas existentes.
- Ajustes personalizables para usuarios con discapacidades auditivas o visuales.
- Crear un sistema con roles diferenciados de paciente y médico.
- Registrar nuevos medicamentos.
- Implementar un sistema de seguimiento de las pautas existentes.
- Mostrar a los usuarios consejos para motivar a los usuarios a instaurar un estilo de vida saludable.

2.1. Descripción de aplicaciones similares

Para llevar a cabo este análisis, se han seleccionado tres aplicaciones para dispositivos móviles con valoraciones muy positivas por parte de los usuarios que realizan funciones similares a las que se pretende implementar en este proyecto. Estas aplicaciones son *Medisafe*, *MyTherapy* y *TOM*. A continuación, se describirán todas y cada una de estas aplicaciones seleccionadas y finalmente se compararán con el objeto de encontrar alguna característica que permita añadir algún requisito que diferencie al resto de aplicaciones.

2.1.1. Medisafe



Figura 2.- Logotipo de Medisafe

*Medisafe*¹ (cuyo logotipo se muestra en la Figura 2) es una aplicación disponible las plataformas móviles Android y iOS que ha sido desarrollada por la compañía con el mismo nombre.

Nada más ejecutar la aplicación por vez primera, se solicita al usuario una recopilación de sus datos tales como nombre y apellidos, género y fecha de nacimiento (Figura 3).

¹ <https://www.medisafe.com/>

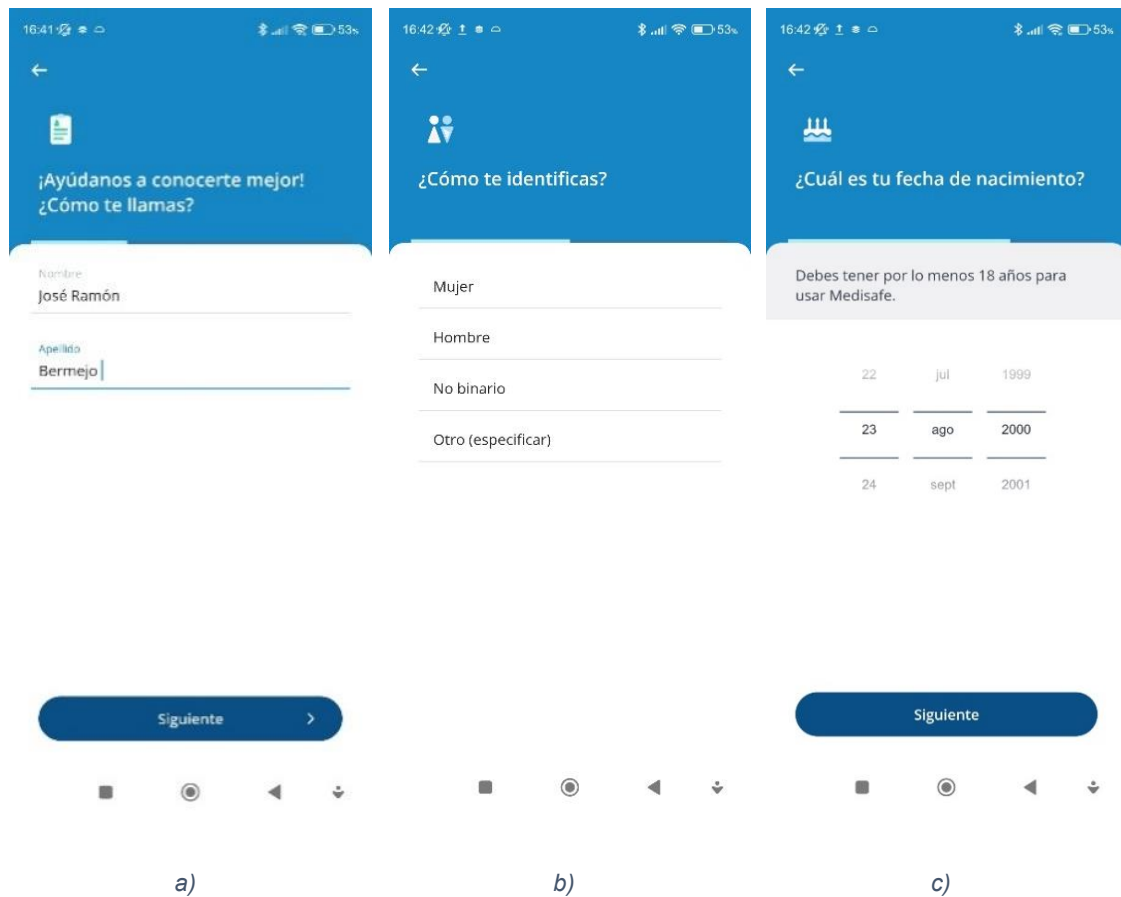


Figura 3.- Recopilación inicial de: a) nombre y apellidos del usuario, b) género del usuario, c) edad del usuario

Una vez aportados estos datos, la aplicación guía al usuario para crear su primer recordatorio de pauta. Para ello, debe registrar el medicamento en cuestión aportando su nombre, indicar la presentación en la que viene el medicamento (pastilla, inyección, gotas, polvo, etc....), la frecuencia con la que es administrado el medicamento y la hora a la que debe tomar la dosis (Figura 4).

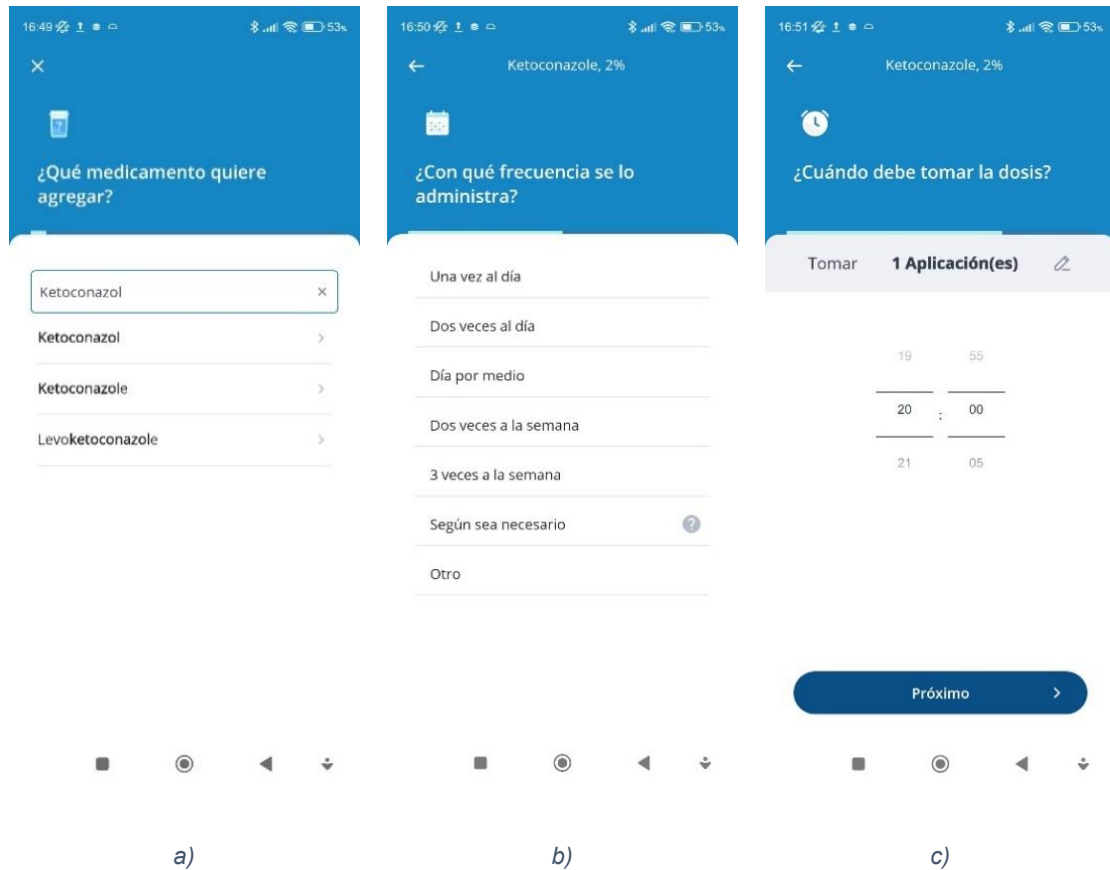


Figura 4.- Recopilación de datos de la primera alarma como: a) nombre del medicamento, b) frecuencia de administración, c) hora de la dosis

Opcionalmente, se puede asignar un icono con el que identificar al medicamento, agregar instrucciones adicionales al medicamento sobre su administración, establecer recordatorio de recarga o establecer una duración determinada a la que el usuario se debe someter a la medicación (Figura 5).

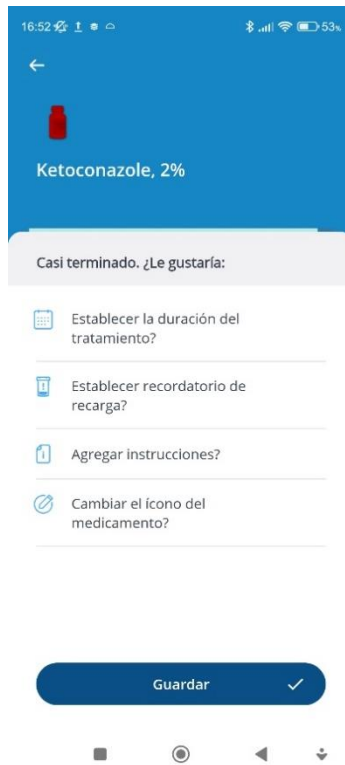


Figura 5.- Interfaz de selección de datos opcionales a agregar en Medisafe

Una vez dada de alta la primera alerta, el usuario accede a la página principal de la aplicación, tal como se muestra en la Figura 6.

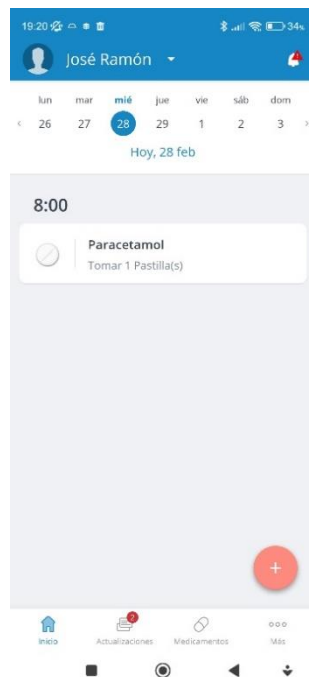


Figura 6.- Menú principal de Medisafe

Tal como se muestra en la anterior figura (Figura 6), se muestra en la barra superior un calendario en el que podemos seleccionar un día determinado y comprobar todas las dosis que debemos tomar ese día en función de la hora. Las dosis se muestran en forma de lista. Al pulsar en una dosis determinada de la lista mostrada, el usuario puede reprogramar la dosis a otra fecha u hora distinta, omitir la dosis o indicar que se ha administrado la dosis.

Por otra parte, la aplicación *Medisafe* ofrece otras funcionalidades como:

- Gestionar una lista de profesionales de la salud del paciente.
- Solicitar citas a un profesional determinado.
- Tomar apuntes relacionados con los problemas de salud, síntomas y cualquier información médica importante.
- Crear seguimientos de salud relacionado con algún síntoma determinado (dolor de cabeza, secreción nasal, dolor de pecho, dolor abdominal, etc....).
- Elaborar informes de cumplimiento de las pautas registradas en el sistema.
- Añadir familiares o amigos que le pueden ayudar a recordar al usuario la administración de un medicamento.
- Compartir las pautas existentes con dichos familiares o amigos.
- Añadir miembros de una unidad familiar, permitiendo al usuario poder gestionar las pautas de sus seres más allegados.

Sin embargo, la aplicación no ofrece ningún soporte para personas con dificultades visuales, a pesar de que se puede agrandar el tamaño de la fuente en el dispositivo, puede resultar no ser suficiente para el paciente. Por otro lado, la aplicación no ayuda a los usuarios a mantener un estilo de vida saludable. Es importante mantener hábitos de vida saludable para prevenir futuros problemas de salud.

2.1.2. MyTherapy



Figura 7.- Logotipo de MyTherapy

*MyTherapy*² (cuyo logotipo se muestra en la Figura 7) es una aplicación similar a la propuesta, desarrollada por la compañía del mismo nombre, disponible tanto para Android como para iOS. Este software cuenta con cierto prestigio al ser utilizado por grandes empresas del sector sanitario y de las ciencias de la salud como Bayer, Novartis, Pfizer y Merck.

En la primera ejecución de la aplicación que realiza el usuario, se le pide que busque un medicamento concreto. Una vez seleccionado el medicamento, la aplicación solicita la frecuencia con la que dicho medicamento es administrado por el paciente. Finalmente, se solicita cuando el usuario desea tomar la dosis indicada. Todos estos pasos se muestran tal y como aparecen en la Figura 8.

² <https://www.mytherapyapp.com/es>

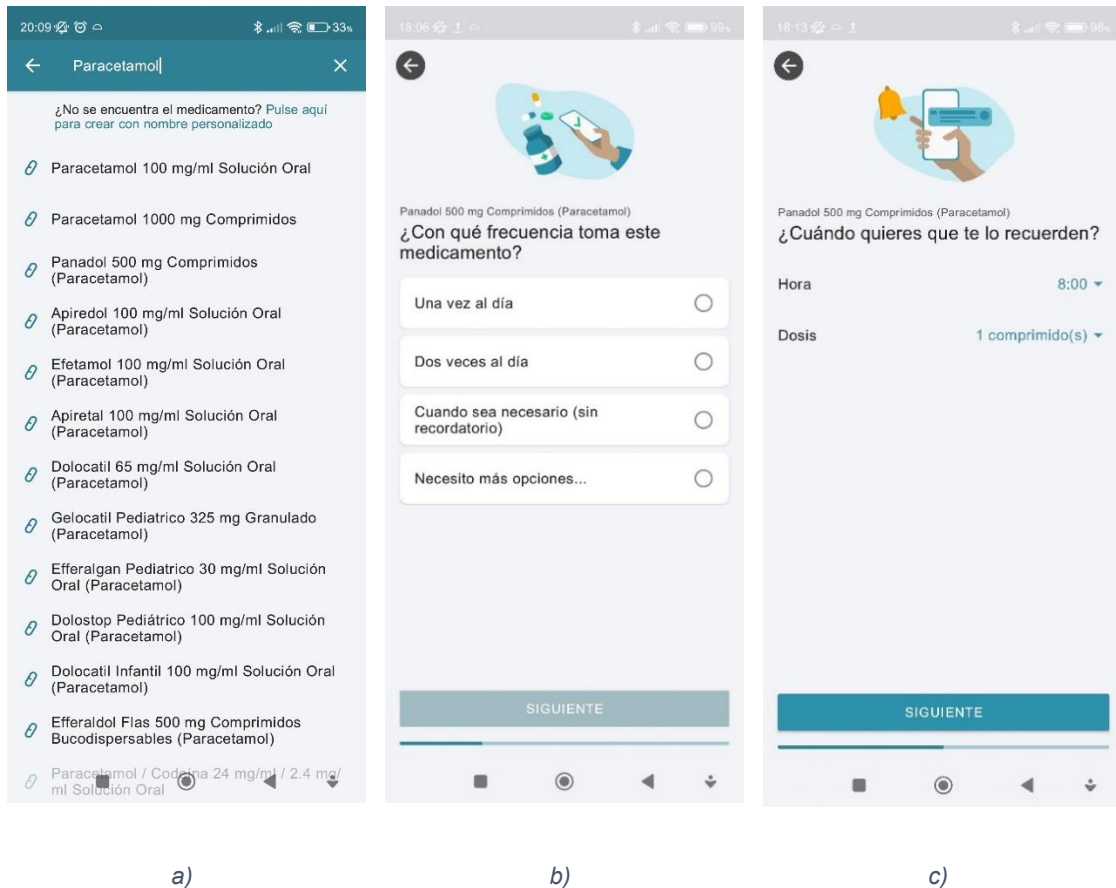


Figura 8.- Creación de alarma en MyTherapy con las siguientes interfaces: a) selección de medicamento, b) selección de frecuencia de administración, c) selección de dosis y hora de administración del medicamento

Posteriormente, la aplicación pregunta al usuario si quiere recibir recordatorios para rellenar las existencias del medicamento anteriormente seleccionado. Para ello, se indica el monto total de existencias actuales (en comprimidos, en polvo, en gotas, etc....) y se indica cuántas unidades deben quedar para que *MyTherapy* notifique al usuario de que se necesita conseguir más medicación. En la Figura 9 se muestra un ejemplo.

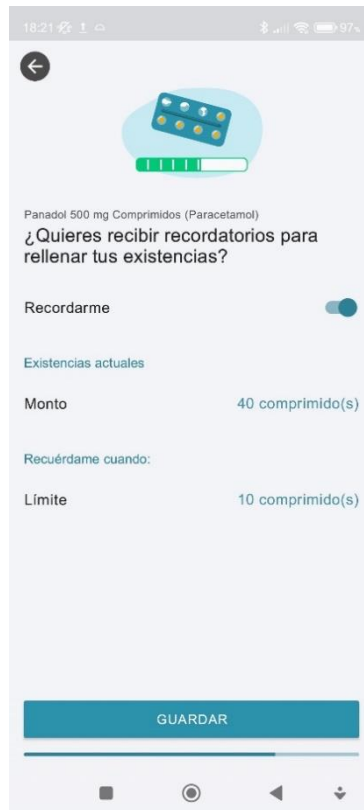


Figura 9.- Interfaz de recordatorio de relleno de existencias en MyTherapy

Finalmente, se le pregunta al usuario el motivo por el cual toma dicha medicación (Dolor, fiebre, etc....). Está permitido seleccionar varias opciones en el caso de que los motivos sean múltiples. Se muestra un ejemplo en la Figura 10.



Figura 10.- Interfaz de indicación del motivo en MyTherapy

Una vez el usuario ha dado de alta su primera alerta, accede al menú principal de la aplicación (Figura 11).



Figura 11.- Menú principal de MyTherapy

Además, cuando llega la hora de tomar la dosis, el usuario es notificado por el sistema tal como se muestra en la Figura 12.



Figura 12.- Ejemplo de notificación de dosis en MyTherapy

Además, *MyTherapy* incluye una serie de funciones interesantes tales como:

- Calendario/Lista en el que se muestran los días en los que se han administrado las dosis de las pautas existentes.
- Registro de síntomas para expresar cómo se siente el usuario, su estado de ánimo y sobre qué fecha y hora ha empezado a sentir dichos síntomas.

- Registro de tiempo en que se realizan diversos tipos de actividades físicas o lúdicas (Baloncesto, *aqua fitness*, bicicleta elíptica, escuchar música, baile, caminar, etc.).
- Medición de valores relacionados con la salud tales como la presión arterial, frecuencia cardíaca en reposo, peso, glucosa en sangre, colesterol, etc.
- Registro de farmacias mediante la aportación de la dirección, el código postal, el teléfono, el correo electrónico o la página web, ofreciendo así una forma fácil y rápida de contactar con dichos centros.
- Creación de citas con un médico y establecimiento de un recordatorio en caso de olvido.
- Gestión de médicos, para que, al igual que sucede con las farmacias, tener un acceso rápido y fácil a sus datos de contacto.

En resumen, la aplicación ofrece una gran variedad de funcionalidades relacionadas con las que se pretenden implementar a lo largo del transcurso del proyecto. A diferencia de *Medisafe*, *MyTherapy* ofrece control sobre valores como los citados anteriormente, permitiendo al usuario realizar un seguimiento más riguroso de su estado de salud. No obstante, a pesar de que se ofrece registrar el tiempo invertido en actividades deportivas, no se fomenta directamente la implantación de un estilo de vida saludable por parte del usuario. Por otra parte, no incluye alguna funcionalidad que sirva de ayuda para personas con algún tipo de discapacidad y así mejorar su experiencia de uso.

2.1.3. TOM



Figura 13.- Logotipo de TOM

TOM³ (cuyo logotipo se muestra en la Figura 13) es una aplicación que ofrece una serie de funcionalidades muy similares a las propuestas para este proyecto.

En primer lugar, cuando el usuario ejecuta el software por vez primera, se le advierte de que su armario de medicamentos está vacío y se le recomienda añadir su primer medicamento cuya administración se debe recordar. Además, se le ofrece al usuario la posibilidad de dar de alta el medicamento mediante el escaneo de su código web (código QR) para así agilizar el proceso. Este diálogo se muestra tal como aparece en la Figura 14.

³ <https://www.tommedications.com/es/>

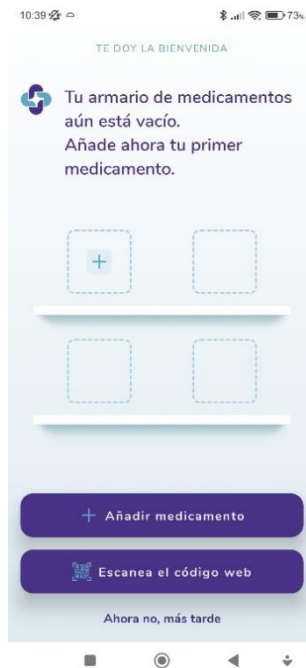
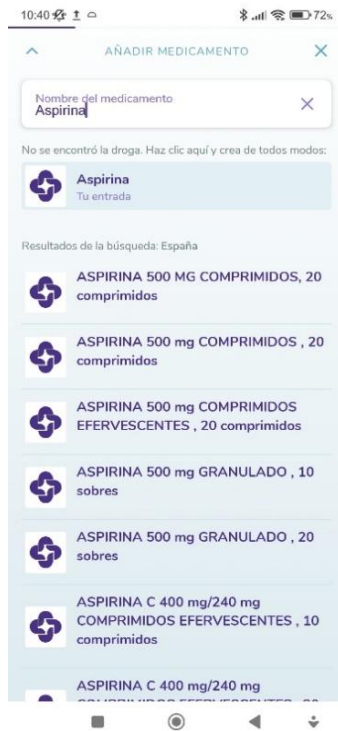
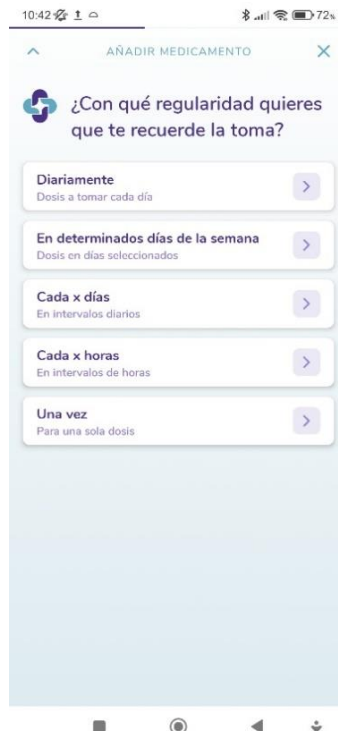


Figura 14.- Interfaz de adición de medicamentos en TOM

A continuación, el usuario busca en la base de datos el medicamento en cuestión aportando su nombre. Tras esto, el usuario indica la regularidad con la que se va a administrar el medicamento en cuestión. A continuación, a diferencia de las otras aplicaciones mostradas anteriormente, TOM ofrece al usuario la posibilidad de, o bien, indicar una fecha de finalización de pauta o si se va a seguir el tratamiento de forma continuada. Finalmente, al igual que en las aplicaciones anteriores, el usuario indica la frecuencia de administración del medicamento. Posteriormente, el usuario asigna una hora a la que se recuerda la toma del medicamento y la dosis adecuada. Todas las interfaces que se muestran a lo largo de este proceso aparecen en la Figura 15.



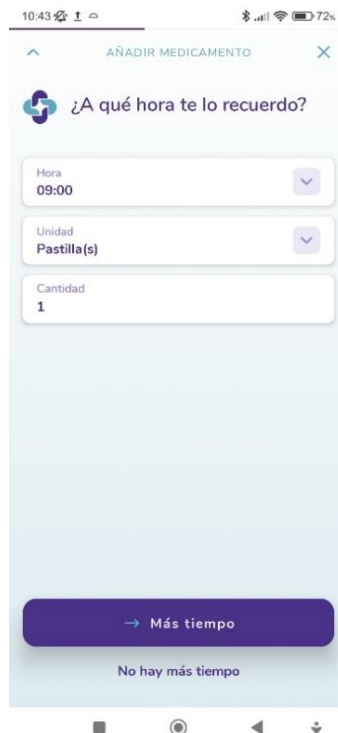
a)



b)



c)



d)

Figura 15.- Creación de alarma en TOM, donde: a) se selecciona un medicamento, b) se selecciona una frecuencia de administración, c) indicación de duración del tratamiento, d) selección de hora y dosis de notificación

Opcionalmente, el usuario tiene la posibilidad de añadir un recordatorio de recarga de existencias del medicamento en cuestión. Para ello, es necesario indicar la cantidad actual de existencias del medicamento, la unidad en la que se mide (pastillas, gotas, etc.) y cuántas unidades deben quedar para que se notifique al usuario que debe recargar las existencias. Luego, el usuario puede asignar una observación al recordatorio referente al momento de toma en cuanto a las comidas. Al finalizar, se muestra un resumen de los datos de la pauta. Todas estas funcionalidades están presentes en las interfaces mostradas en la Figura 16.

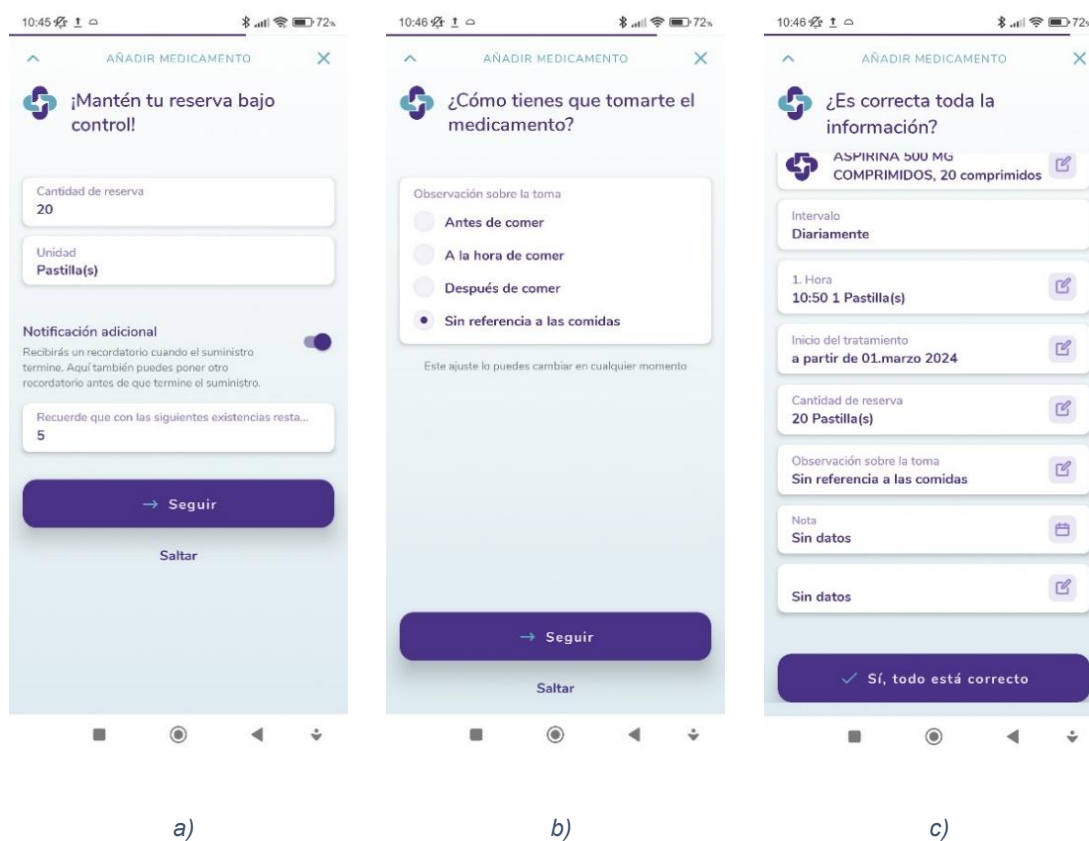


Figura 16.- Finalización de la creación de alarma en la que: a) se ofrece mantener un control de las unidades restantes del medicamento, b) asignar observación de la administración en cuanto a las comidas diarias, c) resumen de la información de la alar

El usuario opcionalmente puede añadir una foto del envase del medicamento con el objeto de facilitar su identificación (Figura 17).

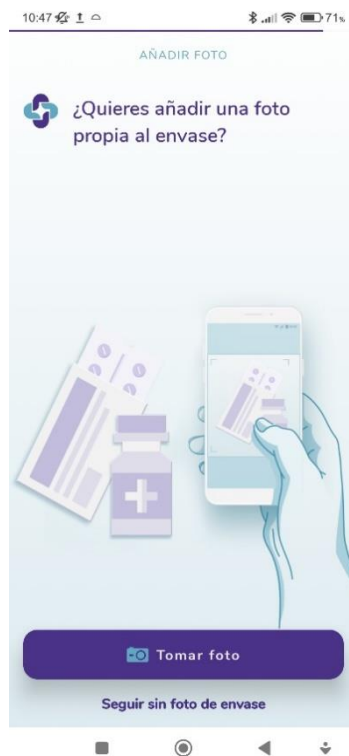


Figura 17.- Interfaz de adición de fotografía del envase del medicamento en TOM

Es aquí cuando la primera pauta del usuario ha sido dada de alta en el sistema, y accede al menú principal (Figura 18).

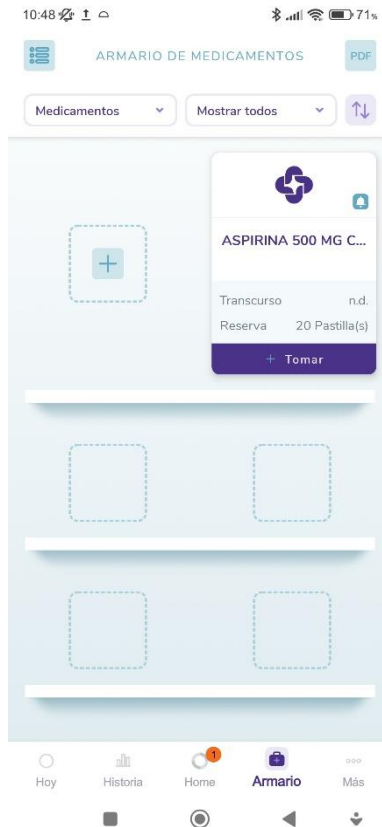


Figura 18.- Menú principal de TOM

Cuando llega la hora de la toma, el usuario recibe una notificación con información detallada del medicamento que debe tomar y la dosis. El usuario puede indicar que ha tomado la dosis, que la ha omitido o que se recuerde la misma toma quince minutos más tarde. Un ejemplo de este tipo de notificaciones se muestra en la Figura 19.

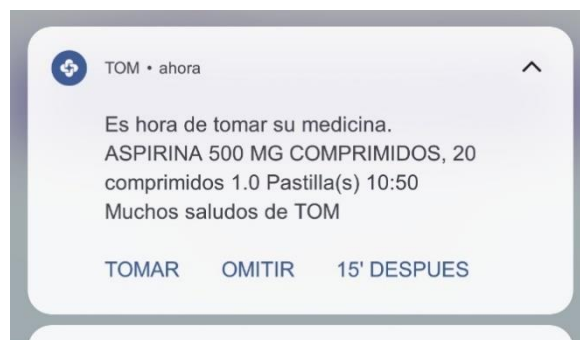


Figura 19.- Notificación de toma de medicamento en TOM

También puede indicar la toma en la misma aplicación (Figura 20).



Figura 20.- Interfaz de dosis pendientes para el día actual en TOM

Además, es posible ver el progreso y cumplimiento de las pautas establecidas, tal como se muestra en la Figura 21.



Figura 21.- Interfaz de históricos de pautas en TOM

Además de la funcionalidad de recordatorios de tomas, esta aplicación cuenta con un conjunto de funcionalidades que resultan interesantes para una aplicación como la propuesta para este proyecto. Estas funcionalidades son:

- Registro de toma diaria de agua.
- Recordatorios para establecer en momentos determinados del día cuál es el estado de ánimo del paciente.
- Gestión de alergias del usuario, ya sean alimentarias, medicamentosas u algún otro tipo.
- Gestión de mediciones de distintos valores relacionados con el estado de salud (calcio, bilirrubina, calorías ingeridas o quemadas, creatinina, densidad ósea, etc.).
- Registro de tiempo invertido en actividades que se consideran beneficiosas para la salud y el bienestar (pasear, leer, meditación, entrenamiento de fuerza, entrenamiento de algún deporte, etc.).
- Generación de informes de los medicamentos, mediciones y actividades del usuario en formato PDF.

En definitiva, esta aplicación es muy similar a *MyTherapy* en cuanto a las funcionalidades que ofrece. Sin embargo, esta ofrece la funcionalidad de registro diario de toma de agua, la cual no está presente en la aplicación analizada anteriormente. Por otro lado, a pesar de que el catálogo de actividades, de las cuales se puede registrar el tiempo invertido, es más amplio, no fomenta de forma directa hábitos de vida saludable, al igual que ocurría tanto con *MyTherapy* como con *Medisafe*. No obstante, resulta muy interesante la idea de generar informes en PDF de los datos que ha ido registrando el usuario en la aplicación para que el personal médico a cargo de dichos usuarios pueda analizar toda esta información. Para acabar, no se ha encontrado ningún tipo de funcionalidad en la que se incluya una distinción entre pacientes y médicos.

2.2. Análisis comparativo de las aplicaciones estudiadas

Una vez se han analizado todas y cada una de las aplicaciones seleccionadas, se procede a compararlas entre sí tomando como referencia los requisitos previamente establecidos, tal como se muestra en la Tabla 1.

Tabla 1.- Tabla comparativa de las alternativas analizadas

Características	Medisafe	MyTherapy	TOM
Recordatorio de pautas de medicamentos	Sí	Sí	Sí
Establecer pauta y dosis	Sí	Sí	Sí
Registro de nuevos medicamentos	Parcial	Parcial	Parcial
Roles paciente/médico	Parcial	Parcial	No
Efectos secundarios	No	No	No
Reservar citas médicas	Sí	Sí	No
Consejos para mantener una vida saludable	No	Parcial	Parcial

Personalización de la accesibilidad	Parcial	Parcial	Parcial
Indicador de cumplimiento de las pautas establecidas	No	Sí	Sí
Publicidad	No	No	No
Interfaz amigable	Sí	Parcial	Parcial
Mensajes de audio	No	No	No
Personalización de notificaciones	Sí	Sí	Sí
Localización de farmacias	No	No	No

En primer lugar, es evidente que las aplicaciones analizadas tienen un objetivo común que es el recordar una serie de pautas de medicamentos previamente establecidas por el usuario, ya que todas ellas cumplen con la funcionalidad de recordar dichas pautas, así como con la relacionada con establecer pautas y dosis.

A continuación, se ha observado en los programas analizados que se permite registrar medicamentos, pero no a nivel de registrar nuevos datos en una base de datos. Tal y como se plantea en dichas aplicaciones, el usuario solicita registrar un nuevo medicamento, lo busca en una base de datos, y una vez seleccionado, se procede a crear un recordatorio para dicho fármaco. Se ha decidido optar por un cumplimiento parcial ya que, aunque existe el concepto de registrar un nuevo medicamento, no es el mismo que se plantea para la aplicación a desarrollar.

Después, se ha observado que no se implementa ningún sistema de comunicación entre usuarios pacientes y médicos. Sí es cierto que, en alguna de estas soluciones, se le ofrece al usuario la posibilidad de registrar citas médicas, de registrar datos de contacto de un profesional médico, etc. Sin embargo, esta comunicación que ofrecen las aplicaciones no se corresponde con lo que se busca en la aplicación a desarrollar.

Por otro lado, en ninguna de estas aplicaciones se ha observado un control de los efectos secundarios que pueda provocar la administración de un medicamento determinado y que sean de gran importancia para un usuario determinado. Se permite seleccionar una gran variedad de medicamentos para generar nuevos recordatorios, pero no se puede saber nada acerca de sus efectos adversos.

Seguidamente, en algunas aplicaciones se permite realizar un rastreo de variables relacionadas con la salud del usuario tales como la tensión arterial, la cantidad diaria de agua ingerida o la frecuencia cardíaca. Aunque es beneficioso para el usuario, se echa en falta alguna funcionalidad de recomendación de directrices para mejorar el estado de salud del usuario.

Luego, no se reproduce ningún tipo de mensajes de audio que indiquen en cada momento lo que el usuario debe realizar y que puedan resultar útiles a la hora de utilizar alguna de estas aplicaciones.

A continuación, cabe destacar que todas las aplicaciones permiten personalizar aspectos relacionados con las notificaciones. Estos aspectos pueden ser la vibración del dispositivo cuando se recibe dicha notificación, el mensaje a mostrar, etc.

En ninguna de ellas se muestra publicidad de alguna entidad externa y en general las interfaces de usuario implementadas son de un grado alto de usabilidad. Aunque, personalmente, ha sido *Medisafe* la que más amigable ha resultado de utilizar.

Por último, se destaca que no se ha observado ninguna función para localizar las farmacias más cercanas según la localización del usuario en ninguna de las aplicaciones analizadas.

2.3. Conclusiones

Entre las aplicaciones analizadas, se han encontrado varias oportunidades de enriquecer una aplicación destinada a gestionar recordatorios de tomas de medicamentos. Entre las oportunidades encontradas, una de ellas es la de implementar un sistema de roles distinguidos entre paciente y médico, de tal forma que los pacientes puedan reservar citas con sus respectivos médicos, y a su vez los médicos puedan consultar las citas que tiene pendientes, y también que los médicos puedan tanto registrar nuevos medicamentos como recetar medicamentos a sus pacientes y estos puedan crear recordatorios de toma. Por otro lado, también se encuentra la oportunidad de ampliar el alcance de estos productos mediante la adición de consejos que ayuden a los pacientes a implantar un estilo de vida saludable. Por otra parte, se han encontrado deficiencias en el tratamiento de los efectos secundarios que vienen asociados a la administración de los medicamentos. No obstante, también se encuentran más oportunidades para mejorar la accesibilidad que ofrece una aplicación de este ámbito. De este modo, personas con algún tipo de discapacidad les resulte más sencillo trabajar con una aplicación de este tipo.

3. Análisis del problema

3.1. Especificación de requisitos

Una parte fundamental del proceso de ingeniería del software es la especificación de requisitos [8]. Se trata de una actividad que consiste en anotar todos los requisitos de forma clara, completa y coherente en forma de documento. Previamente, se debe de haber capturado potenciales requisitos de diversas fuentes de información (documentación existente, *stakeholders*, etc...). Mediante la realización de esta actividad, se obtiene un documento en el que se detallan de forma clara y precisa todos y cada uno de los requisitos capturados.

A la hora de especificar dichos requisitos, es importante destacar los dos tipos de requisitos que existen en el contexto de un producto software:

- **Requisitos funcionales:** son aquellos requisitos directamente relacionados con las funciones del sistema que se pretende desarrollar.
- **Requisitos no funcionales:** son un tipo de requisitos que explican las posibles restricciones y limitaciones que el sistema debe cumplir. No tienen ningún impacto sobre el conjunto de funcionalidades de la aplicación.

Una vez introducidos los tipos principales de requisitos que existen en un sistema software, se procede a enumerar los requisitos funcionales elicitados:

- Recordar pautas de medicamentos
- Establecer pautas de medicamentos
- Establecer dosis a pautas
- Registrar nuevos medicamentos
- Notificaciones de recogida de medicamentos según recetas
- Notificaciones de finalización de recetas
- Indicación de cumplimiento de pautas establecidas
- Registro de síntomas
- Integración con otros dispositivos
- Localización de farmacias cercanas
- Escaneo de código de barras para agilizar registro de medicamentos
- Ajustes para usuarios con discapacidades auditivas o motrices
- Perfiles familiares
- Registro de nuevos usuarios
- Inicio de sesión
- Roles paciente/médico
- Reservar citas médicas
- Selección de idioma
- Consejos de salud una vez se entra a la app
- Artículos estilo “*newsletter*” sobre salud
- Notificaciones personalizables
- Interacciones medicamentosas

A continuación, se enuncian los posibles requisitos no funcionales elicitados:

- La aplicación se comunicará con un servidor que se comunicará a su vez con la base de datos mediante comunicaciones HTTPS.
- El tiempo de respuesta de la aplicación cuando haga una petición al servidor deberá ser inferior a 2 segundos.

- Cada una de las interfaces que conforman la aplicación deben mostrarse al usuario en un tiempo inferior a 1 segundo.
- La aplicación deberá incorporar mensajes de voz para guiar a personas con discapacidades visuales durante su experiencia de uso en la aplicación.
- La aplicación deberá tener un porcentaje de tiempo en el que está operativa del 99,9%.

3.2. Tabla “MoSCoW”

El método “MoSCoW” es un método creado por Dai Clegg en 1994 para aumentar la agilidad en el desarrollo de aplicaciones y que sirve para determinar la priorización de requisitos dentro de un proyecto limitado de tiempo [9]. La idea es priorizar todos y cada uno de los requerimientos del proyecto por categorías de tal forma que se obtenga una lista ordenada de éstos según su prioridad de implementación. Es un método muy eficaz en entornos en los que se aplica una metodología ágil, como es el caso de este proyecto.

La aplicación de este método ofrece al equipo de desarrollo una serie de ventajas. En primer lugar, es fácil de usar ya que se basa en principios simples y en el lenguaje humano de priorización (tiene que, debería, podría, no se hará). Por otro lado, es una solución muy buena para alcanzar un acuerdo con los “*stakeholders*” sobre qué requisitos se van a implementar en primer lugar y cuáles no se van a considerar para su implementación.

Sin embargo, en el caso de que la toma de decisiones dentro de una organización sea realizada por una sola persona, la priorización de requisitos resultará ser subjetiva y poco eficiente. Además, si las personas encargadas de realizar dicha priorización tienen poca experiencia, puede que el resultado no sea el óptimo.

Para priorizar los requisitos, se dividen en las siguientes cuatro categorías:

- **MUST** → Engloba los requisitos absolutamente críticos para el proyecto. Sin ellos, el proyecto se considerará fallido.
- **SHOULD** → Contiene aquellos requisitos que se consideran críticos, pero no imprescindibles.
- **COULD** → En este apartado, los requisitos presentes son los que estaría bien tenerlos, ya que añadirían valor al proyecto, pero que no son críticos para el resultado final.
- **WON'T** → Los requisitos de esta categoría no merecen la inversión y se podrían considerar más tarde.

A continuación, en la Tabla 2, se muestra una priorización de los requisitos funcionales previamente citados.

Tabla 2.- Tabla de priorización de requisitos mediante MoSCoW

MUST	<ul style="list-style-type: none"> • Recordar pautas de medicamentos • Establecer pautas de medicamentos • Establecer dosis a pautas
-------------	---

SHOULD	<ul style="list-style-type: none"> • Registro de nuevos usuarios • Inicio de sesión • Cierre de sesión • Roles paciente/médico
	<ul style="list-style-type: none"> • Registrar nuevos medicamentos • Indicación de cumplimiento de pautas establecidas • Localización de farmacias cercanas • Selección de idioma • Consejos de salud una vez se entra a la app • Interacciones medicamentosas • Consultar citas
COULD	<ul style="list-style-type: none"> • Notificaciones de recogida de medicamentos según recetas • Notificaciones de finalización según recetas • Escaneo de código de barras de medicamentos • Ajustes para usuarios con discapacidades auditivas, motrices y visuales • Reservar citas médicas • “<i>Newsletter</i>” sobre salud • Perfiles familiares
WON'T	<ul style="list-style-type: none"> • Registro de síntomas • Integración con <i>SmartWatch</i> • Notificaciones personalizables

3.3. Modelo de dominio

Un modelo de dominio es una representación visual estructurada de conceptos conectados entre sí u entidades del mundo real que involucra vocabulario, comportamientos y relaciones de todas estas [1].

El objetivo principal de este diagrama es mejorar la comprensión y comunicación entre los integrantes del equipo de desarrollo y los “*stakeholders*” del proyecto en la medida de lo posible.

Tal como se muestra en la Figura 22, el modelo de dominio está representado mediante un diagrama de clases formado por clases que representan un concepto del dominio en cuestión, una serie de relaciones que expresan qué tipo de relación existe entre los elementos del diagrama y por último la multiplicidad que restringe las relaciones existentes en el diagrama.

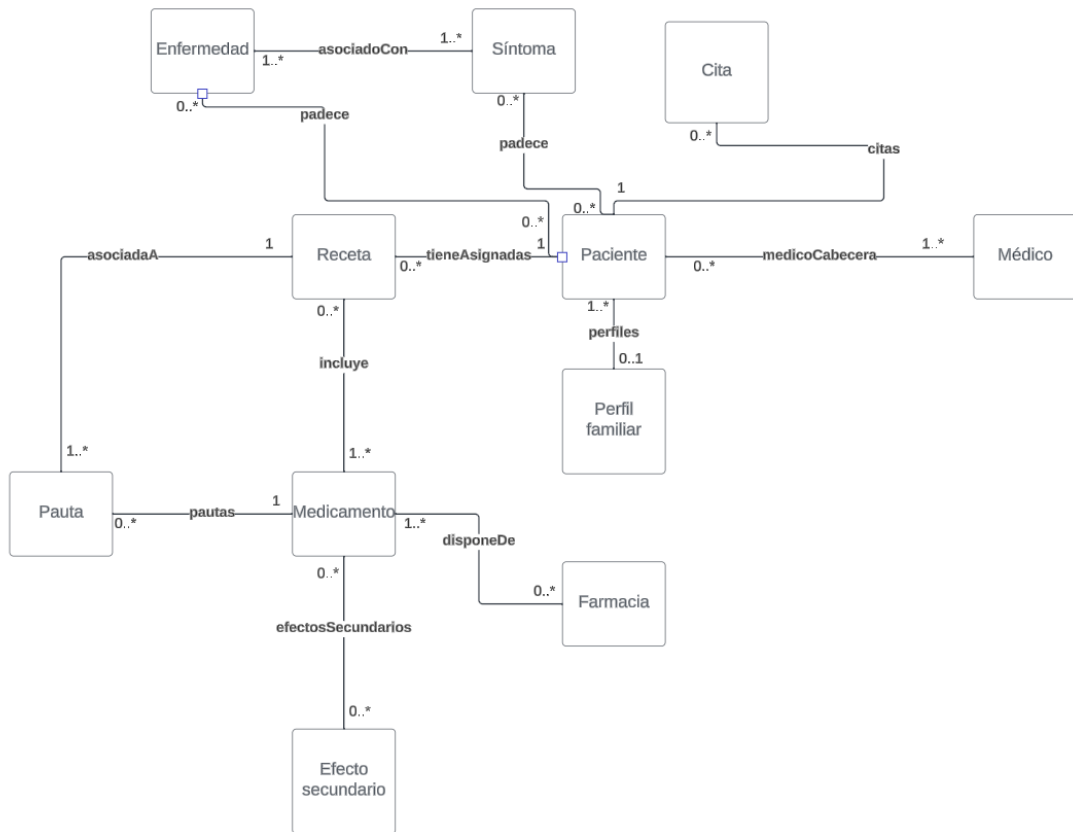


Figura 22.- Diagrama que representa el modelo de dominio del problema a resolver

Se han detectado las siguientes entidades:

- **Enfermedad** → Proceso por el que pasa un paciente cuando está enfermo. Una enfermedad puede tener asociados indefinidos síntomas y la misma enfermedad puede ser padecida por más de un paciente.
- **Síntoma** → Manifestación que revela una enfermedad. Un síntoma se puede presentar en indeterminados pacientes.
- **Paciente** → Usuario existente en la aplicación. Un paciente puede concertar indeterminadas citas con su médico, puede presentar indeterminados síntomas, puede padecer indeterminadas enfermedades, puede tener asignadas indeterminadas recetas, tiene asignado un médico de cabecera y existe la posibilidad de que dicho paciente forme parte de un perfil familiar.
- **Cita** → Reunión previamente acordada entre el paciente y su médico. Un paciente puede solicitar indefinidas citas.

- **Médico** → Personal sanitario que asiste a un número indeterminado de pacientes.
- **Receta** → Documento en el cual se indica los medicamentos a tomar por el paciente y las pautas a seguir. Una receta siempre es asignada a un solo paciente, incluye indeterminados medicamentos adscritos y tiene asociadas indeterminadas pautas.
- **Perfil familiar** → Grupo de pacientes que representa una misma unidad familiar atendida por el mismo médico.
- **Medicamento** → Sustancia que receta el médico al paciente para tratar una enfermedad determinada. Un medicamento puede estar incluido en indeterminadas recetas, estar disponible en indeterminadas farmacias, estar asociado con indeterminadas pautas y su administración puede conllevar indeterminados efectos secundarios.
- **Pauta** → Norma a seguir que establece el médico para la administración de un medicamento. Cada pauta siempre estará asociada a una sola receta y a un solo medicamento.
- **Farmacia** → Establecimiento en el que se comercia con medicamentos. Cada farmacia puede comerciar con uno o muchos medicamentos.
- **Efecto secundario** → Efecto adverso producido por la toma de un medicamento. Un solo efecto secundario puede ser producido por la toma de un número indeterminado de medicamentos.

3.4. Diagrama de casos de uso

El diagrama de casos de uso es un diagrama que forma parte de UML⁴. Es uno de los diagramas más utilizados y es utilizado para representar los actores o usuarios que podrían interactuar con el sistema a desarrollar y cuáles son las funcionalidades (casos de uso o requisitos funcionales) con las que se relacionan [10].

Este tipo de diagrama puede ser utilizado para gran variedad de fines, entre los cuales destacan los siguientes:

- Representar los actores o usuarios que pueden existir en el sistema.
- Representar las relaciones entre los distintos requisitos funcionales y los actores existentes.
- Guiar el desarrollo del sistema.
- Comunicarse de la forma más precisa posible entre el equipo de desarrollo y los “*stakeholders*”.

Las relaciones pueden conectar casos de uso con actores o conectar casos de uso entre sí. Cuando dos casos de uso se relacionan, la relación existente entre ambos puede ser de dos

⁴ <https://www.uml.org/>

tipos: <<include>> cuando la ejecución de un caso de uso conlleva la ejecución del caso de uso al que va conectado; y <<extend>> cuando la ejecución de un caso de uso es opcional en el caso de que el caso de uso al que se conecta sea ejecutado. Existe además otra relación conocida como generalización. En este caso, su uso es hacer que un actor herede los mismos comportamientos de otro actor.

Tal como se muestra en la Figura 23, existen tres tipos de actores en el sistema a desarrollar. Estos son el usuario no autenticado, el médico y el paciente. En primer lugar, el usuario no autenticado es aquel que accede al sistema y que no dispone todavía de credenciales de identificación para acceder al sistema o bien no se ha identificado todavía en el mismo. En segundo lugar, el médico es aquel usuario que dispone de los permisos necesarios para registrar nuevos medicamentos, crear recetas y asignar recetas a sus pacientes asignados. Por último, el paciente es aquel usuario que tiene asignadas recetas y que puede establecer alarmas de notificación de medicamentos en una hora determinada y reservar citas con su médico entre otras posibilidades.

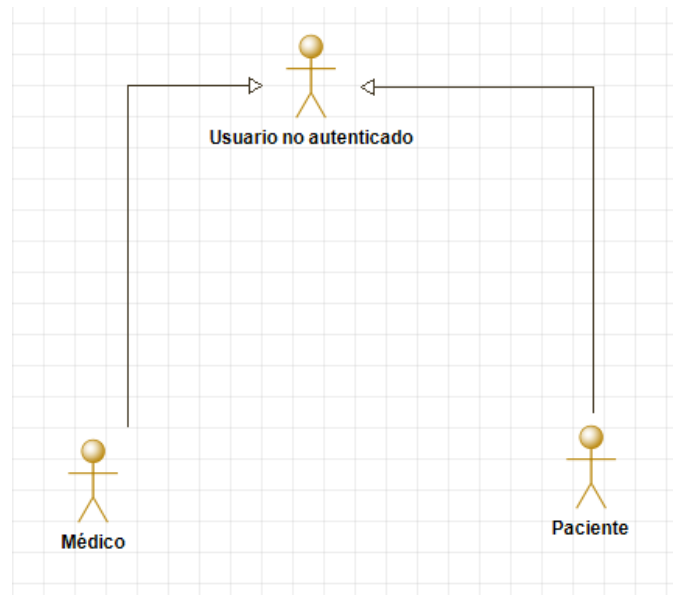


Figura 23.- Jerarquía de los actores presentes en la aplicación

En la Figura 24, se muestra un diagrama de casos de uso para el usuario no autenticado. En el contexto de este proyecto, un usuario no autenticado podrá registrarse en el sistema (Tabla 13), autenticarse mediante la aportación de sus credenciales (Tabla 10), seleccionar un médico de entre los existentes en el sistema (Tabla 24), en el caso de que no sea un médico, seleccionar el idioma de uso con el que quiere utilizar la aplicación (Tabla 23) y cerrar la sesión (Tabla 11). Se puede dar el caso de que, al no disponer de ningún tipo de credencial, se tenga que registrar en el sistema para poder acceder al mismo (Tabla 13).

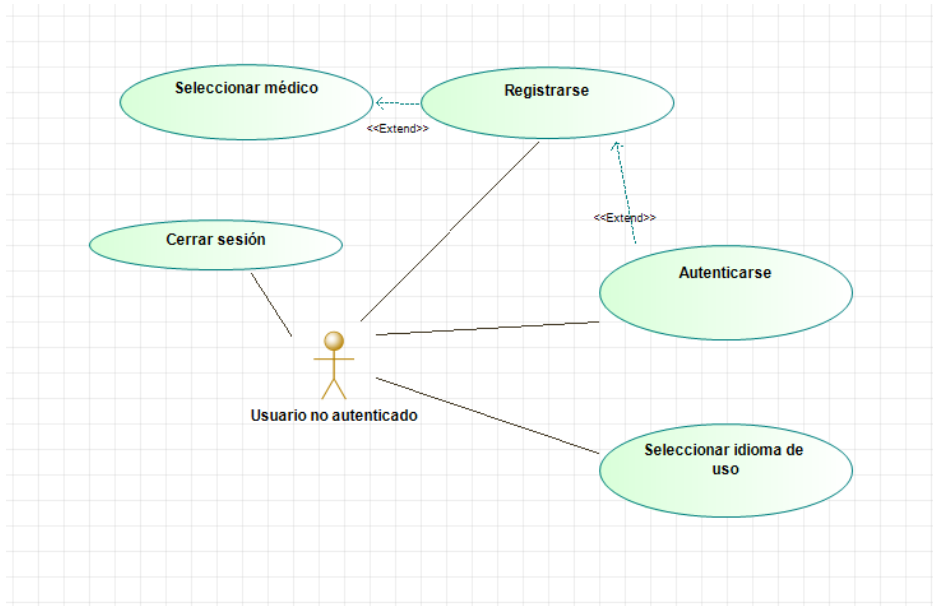


Figura 24.- Casos de uso del usuario no autenticado en la aplicación

Por otro lado, en la Figura 25 se muestra el diagrama de casos de uso de un paciente cualquiera identificado dentro del sistema. Como se puede observar, un paciente puede crear alertas de pautas (Tabla 16), reservar citas con su médico (Tabla 20), visualizar consejos para mejorar su salud (Tabla 14), borrar alarmas (Tabla 15), cancelar citas médicas (Tabla 30), consultarlas (Tabla 25), visualizar artículos sobre salud (Tabla 32), registrar síntomas sufridos (Tabla 31) y localizar farmacias cercanas (Tabla 21). Al crear una alerta de pauta determinada, se asignará un medicamento (Tabla 17), una dosis (Tabla 18) y un periodo de pauta a la misma (Tabla 19). Además, siempre que el paciente reserve una cita con su médico (Tabla 20), habrá comprobado la disponibilidad (Tabla 37) y seleccionado una fecha y una hora de cita (Tabla 26) previamente.

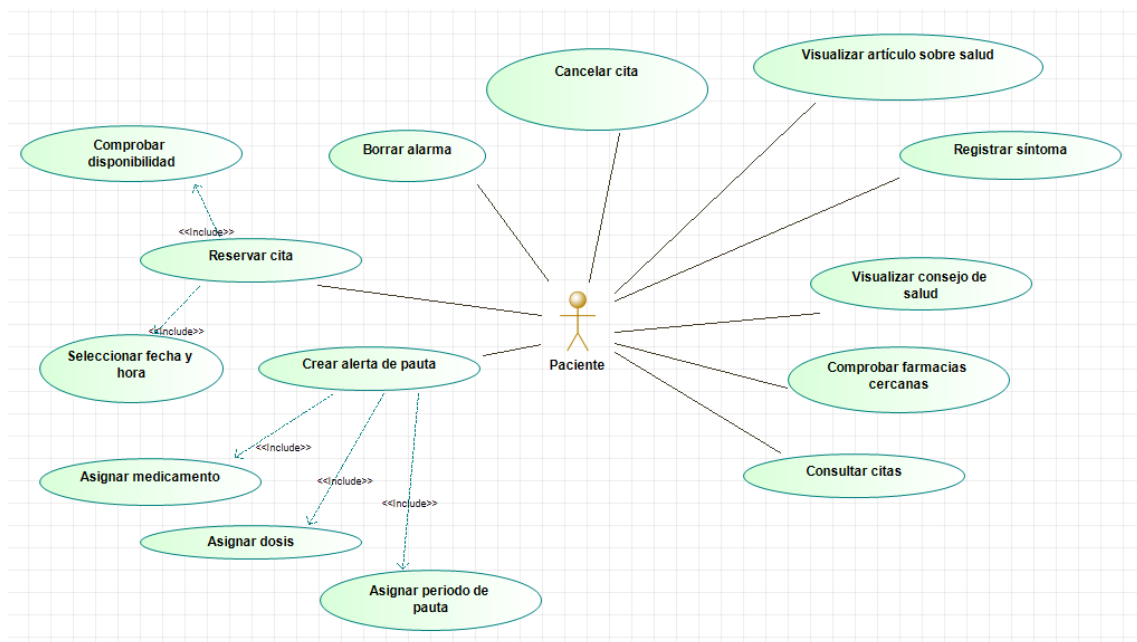


Figura 25.- Casos de uso del paciente en la aplicación

Por último, tal como se puede comprobar en la Figura 26, un usuario identificado como médico en la aplicación podrá crear recetas, localizar farmacias cercanas (Tabla 21), visualizar artículos sobre la salud (Tabla 32), visualizar consejos sobre bienestar (Tabla 14), consultar las citas pendientes (Tabla 25) y registrar nuevos medicamentos en la aplicación (Tabla 22). Cabe indicar que, al crear una receta para un paciente (Tabla 35), el médico debe asignar el medicamento idóneo para el mismo (Tabla 36) o bien registrarlo en el caso de que no exista (Tabla 22). También, al registrar un nuevo medicamento en el sistema, el médico deberá indicar si el medicamento necesita de prescripción médica (Tabla 33), indicar los efectos secundarios que puede conllevar su administración (Tabla 28) y establecer un nombre que identifique al medicamento en cuestión (Tabla 29). Además, puede existir la posibilidad de que se escanee el código de barras del medicamento para agilizar su registro (Tabla 34). Esto ayudaría a agilizar su asignación de nombre, ya que, al escanear dicho código, se cargaría en la aplicación.

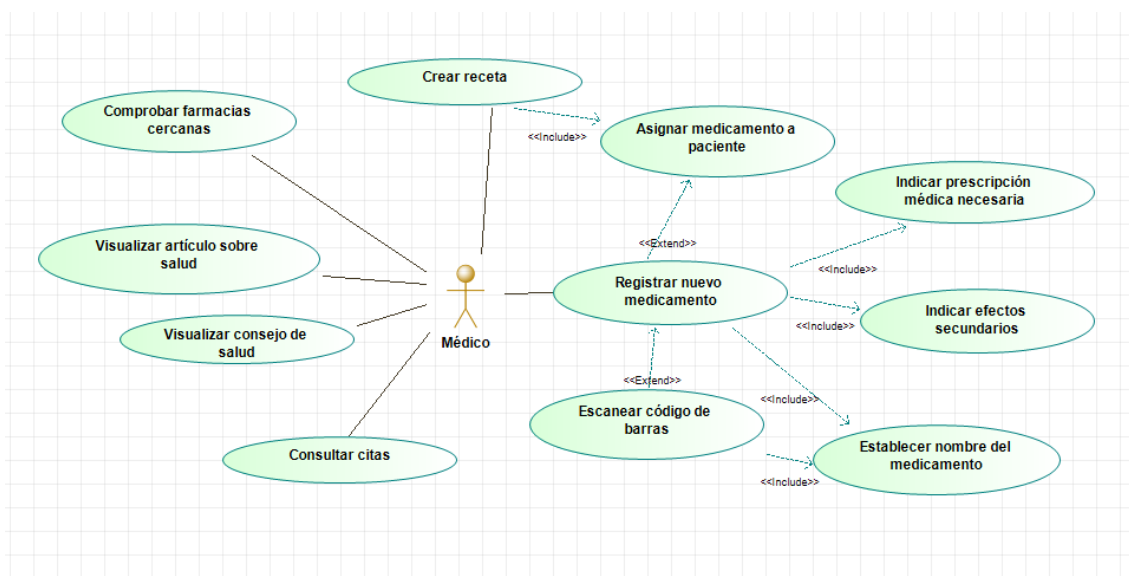


Figura 26.- Casos de uso del médico en la aplicación

3.5. Análisis de la seguridad

Una de las características principales con las que se pretende evaluar la calidad general del software es la seguridad, según la norma ISO/IEC 25010⁵. Cuando se habla de seguridad, se refiere al nivel de protección de la información y de los datos de los usuarios para que personas no autorizadas no puedan leerlos o modificarlos y que se permita el acceso a aquellos que sí que están autorizados para ello. A su vez, esta característica se divide en los atributos citados a continuación:

- **Confidencialidad:** Los datos deben ser accesibles solamente para aquellas personas autorizadas.
- **Integridad:** Los datos deben estar protegidos ante modificaciones o eliminaciones no autorizadas.
- **No repudio:** Se debe demostrar todas las acciones que han tenido lugar en el sistema, de tal modo que no sean repudiadas en el futuro.
- **Responsabilidad:** Se refiere a la capacidad de seguir las acciones de un usuario u entidad determinada.
- **Autenticidad:** Capacidad del sistema software para demostrar que la identidad de un usuario o de algún recurso es la que realmente se afirma.

⁵ <https://iso25000.com/index.php/normas-iso-25000/iso-25010>

- **Resistencia:** El producto software debe seguir ofreciendo el servicio incluso en condiciones de estar sufriendo algún tipo de ataque malicioso.

En este caso concreto, la seguridad es una característica crucial a la hora de evaluar la calidad del software. Se ha enfocado especial atención en construir este sistema software de tal modo que ofrezca confidencialidad, integridad, autenticidad y resistencia. En primer lugar, el producto software debe restringir el acceso a los datos para aquellas personas autorizadas para ello (el acceso a las citas médicas de un médico, por ejemplo). Por otro lado, datos como los referentes a las farmacias, deben estar protegidos ante la posibilidad de modificación e incluso borrado por parte de entidades externas. A continuación, se debe implementar un sistema de autenticación de usuarios (véase el capítulo 4.4.3.1) para verificar la identidad de todos y cada uno de ellos.

Por último, el sistema debe ofrecer sus servicios incluso en condiciones de ataques maliciosos. En concreto, se pretende establecer defensas ante los ataques de inyección de código y los accesos no autorizados a los datos. Para prevenir este tipo de problemas, se valorará alojar los datos de los usuarios en servidor ofrecido por algún proveedor de servicios de alta fiabilidad.

3.6. Análisis del marco legal y ético

Para cualquier profesional, es de relevante importancia cumplir con lo estipulado en el marco legal y ético en todas aquellas áreas en las que el resultado de este proyecto software repercute directa o indirectamente. A continuación, se analizan las distintas facetas legales y éticas del proyecto en cuestión.

3.6.1. Análisis de la protección de datos

Al tratarse de un producto software que va a trabajar con datos confidenciales como pueden ser los datos de citas médicas de los usuarios, se deben someter a un tratamiento específico. A continuación, se procede a analizar todo aquello para tener en cuenta.

Antes de analizar las acciones a realizar en este proyecto, se debe destacar que la principal normativa en la Unión Europea que regula todo aquello en lo que respecta al tratamiento de datos personales y a su circulación por los territorios constituyentes, es el Reglamento General de Protección de Datos o Reglamento 2016/679⁶. Es por ello por lo que el producto *software* a desarrollar debe respetar estrictamente esta normativa.

Tal y como se estipula en dicho reglamento, se deben tener en cuenta las siguientes consideraciones:

- Los usuarios, al darse de alta, deberán otorgar su consentimiento explícito para el tratamiento de los datos que hagan referencia a su estado de salud. Es decir, se deberá informar al usuario a través de, por ejemplo, una ventana emergente, en la que el usuario pueda aceptar o denegar su consentimiento para el tratamiento de sus datos sanitarios. En caso de denegación, no se registrará en el sistema.
- En el caso de que un usuario determinado desee la confirmación de parte del responsable del tratamiento de datos de si se está tratando o no con datos personales de su concierne. En caso afirmativo, tendrá derecho a comprobar todos sus datos personales.
- De acuerdo con el artículo 17.1 de la ley 41/2002⁷ de autonomía del paciente y derechos y obligaciones en materia de información y documentación clínica, los datos deberán

⁶ <https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:02016R0679-20160504>

⁷ <https://www.boe.es/eli/es/l/2002/11/14/41/con#a17>

conservarse al menos durante un periodo de cinco años desde su alta, para su eliminación en caso de que el usuario lo solicite explícitamente.

- En el caso de que los datos se alojen en alguna base de datos remota, se deberá asegurar de que los datos sean alojados en algún servidor localizado dentro del territorio europeo.

3.6.2. Propiedad intelectual

Uno de los aspectos a considerar dentro del marco del desarrollo de un sistema software es el tipo de licencia que se le va a aplicar. Una licencia software consta de un acuerdo en el que el usuario acepta una serie de términos y condiciones del desarrollador o entidad desarrolladora (en este caso, el autor del proyecto) para el uso del sistema software.

Para este proyecto, se ha empleado la licencia MIT⁸. La licencia MIT es un tipo de licencia creada por el Instituto Tecnológico de Massachussets (*Massachussets Institute of Technology*). Se trata de una licencia de software libre permisiva, lo cual permite al usuario realizar las siguientes acciones siempre y cuando se referencie al autor original del software:

- Usar el sistema software para cualquier tipo de propósito.
- Estudiar el código fuente y modificarlo con el objetivo de adaptar el software a las necesidades del usuario.
- Distribuir copias exactas del programa original o de alguna versión modificada.

Además, al tratarse de una aplicación que va a ejecutarse en dispositivos móviles con *Android*, y este estar sujeto a la licencia Apache 2.0⁹, se ofrece al desarrollador o entidad desarrolladora la libertad de emplear una licencia de libre elección para el trabajo realizado.

Por otro lado, las imágenes utilizadas dentro de la aplicación están sujetas a la licencia creada por el sitio web *Flaticon*¹⁰. De acuerdo con lo que se indica en dicha licencia, los usuarios disponen de total libertad para usar los contenidos bajo dicha licencia para cualquier propósito y a realizar modificaciones y trabajos derivados siempre que se respete la autoría del autor original, al igual que sucede con la licencia MIT descrita anteriormente. Para atribuir la autoría, se incluirá un apartado exclusivamente para ello en la documentación del repositorio, en el que se incluirá un enlace que dirigirá al contenido original en cuestión para cada imagen empleada.

3.7. Análisis de riesgos

En este caso, al tratarse de un proyecto de desarrollo de un *software* orientado a una gran variedad de usuarios, es importante analizar los riesgos existentes que pueden provocar que el producto desarrollado no sea aceptado o bien que no satisfaga al usuario. Es por ello por lo que se va a analizar los riesgos existentes en este proyecto en cuestión.

En primer lugar, existe el riesgo de que las interfaces diseñadas no sean fáciles para usar, provocando descontento entre el conjunto de usuarios y, por consiguiente, su rechazo. Se trata de un riesgo técnico al estar relacionado con un aspecto intrínseco del sistema *software*. Es por

⁸ <https://mit-license.org/>

⁹ <https://www.apache.org/licenses/LICENSE-2.0>

¹⁰ <https://www.flaticon.es/>

ello por lo que se decide aplicar pruebas de usabilidad (véase el capítulo 6.3) con un grupo de usuarios con el objetivo de valorar el grado de usabilidad de las interfaces de usuario. Además, también se emplearán las llamadas Heurísticas de Nielsen (véase el capítulo 5.3), cuyo objetivo es identificar aquellos aspectos relativos a las interfaces de usuario que podrían mejorarse para mejorar la usabilidad de la aplicación desarrollada.

A continuación, se ha detectado también un riesgo relacionado con los aspectos legales propios del dominio de la aplicación, ya que un incumplimiento de dicha legislación podría llevar a sanciones legales (véase el capítulo 3.6). Por lo tanto, se va a tratar de investigar la legislación vigente, tanto a nivel nacional como a nivel europeo, con el objeto de adaptar el sistema de acuerdo con dicha legislación.

Posteriormente, el hecho de que se produzcan fallos en la seguridad de la aplicación puede provocar que la privacidad de los datos de los usuarios se vea comprometida (véase el capítulo 3.6.1). Como medida para prevenir esto, se optará por alojar los datos en un servidor ofrecido por un proveedor de servicios de confianza (*Firebase*, por ejemplo), para así prevenir este tipo de problemas.

Por otro lado, se prevé diseñar el sistema ante un posible aumento tanto del número de usuarios como de la carga de procesamiento del servidor de datos, de tal modo que el sistema sea lo más escalable posible. Tal y como se ha comentado anteriormente, es de crucial importancia la selección del servidor de datos para abordar toda esta problemática.

Cabe destacar que emplear un proveedor de servicios externo para agilizar el desarrollo del producto *software* también puede suponer un riesgo a tener en cuenta. Cualquier problema que pueda surgir y que esté relacionado con el proveedor de servicios, puede afectar sustancialmente tanto a la planificación y el alcance del proyecto, como al desarrollo en sí, ya que todo esto supondría un retraso y una consecuente postergación de hitos.

3.8. Identificación y análisis de soluciones posibles

Habitualmente, a la hora de abordar un proyecto de cualquier ámbito, suele existir más de una opción posible para abordar un problema determinado. En este capítulo, se detallan cuáles han sido las distintas alternativas que se han valorado según los problemas a resolver a lo largo del desarrollo de este proyecto.

3.8.1. *AlarmManager* vs. *Cloud Messaging*

En primer lugar, la decisión principal que se tuvo que tomar es qué tecnología utilizar para implementar el sistema de disparo de alarmas de administración de dosis de algún medicamento. Las dos principales alternativas que se valoraron fueron la clase *AlarmManager*¹¹ nativa de *Android* y la solución *Cloud Messaging*¹² ofertada por *Firebase*. En este capítulo se analizarán las características de cada una de estas alternativas y se argumentará el porqué de la decisión final.

A continuación, se procede a analizar la clase *AlarmManager* de *Android*. Cabe destacar que las alarmas establecidas mediante esta opción son a nivel local, es decir, no se almacenan en ningún agente externo. Una de las características más llamativas de esta alternativa es que permite activar operaciones en horas o intervalos de tiempo determinados, de tal modo que se comience

¹¹ <https://developer.android.com/reference/android/app/AlarmManager>

¹² <https://firebase.google.com/docs/cloud-messaging?hl=es>

a ejecutar una serie de acciones una vez se dispare la alarma, incluso fuera del ciclo de vida de la aplicación a desarrollar. Además, es muy útil a la hora de minimizar los recursos que la aplicación requiere para su funcionamiento. Sin embargo, dichas alarmas establecidas se pierden cuando se apaga o reinicia el dispositivo en cuestión. Por lo tanto, habría que diseñar la aplicación de modo que las alarmas previamente establecidas sean reiniciadas una vez el dispositivo vuelva a estar activo.

Por otro lado, la solución *Cloud Messaging* de *Firebase* es una solución utilizada para el envío tanto de notificaciones como de mensajes a dispositivos móviles, así como a aplicaciones de forma segura y eficiente. Esta alternativa permite una comunicación en tiempo real con envío instantáneo de mensajes a los usuarios. No obstante, al usar *Cloud Messaging*, se depende de disponer de una conexión de red. Por lo tanto, en aquel caso en el que el dispositivo en cuestión esté fuera de línea en una red poco fiable, no se recibirán correctamente las notificaciones. Además, no todos los mensajes podrían enviarse correctamente, ya que se ofrece una tasa de envíos exitosos del 95%.

Finalmente, se ha optado por emplear la clase *AlarmManager* por los siguientes motivos. El primero de todos ha sido la independencia de conexión a alguna red, ya que las alarmas se establecen a nivel local, tal y como se ha comentado anteriormente. Asimismo, con esta opción se asegura que la alarma se dispara una vez se alcance la hora o el intervalo de tiempo especificado previamente, es decir, muestra mayor fiabilidad que la alternativa perdedora. Por último, aunque emplear esta clase suponga diseñar el sistema de una forma determinada, se llevará a cabo una implementación para cubrir aquellos casos en los que el dispositivo sea apagado o reiniciado.

3.8.2. Cloud Firestore vs. Realtime Database

Una de las cuestiones que se han planteado antes de empezar con el desarrollo del proyecto ha sido el escoger cuál de las herramientas de *Firebase* enfocadas en el almacenamiento de datos se debería utilizar en el proyecto. Las alternativas son dos: *Cloud Firestore* y *Realtime Database*.

Tabla 3.- Resumen de características de *Cloud Firestore* y *Realtime Database*

	Cloud Firestore	Realtime Database
Modelo de datos	Colecciones de documentos	JSON
Soporte en tiempo real y sin conexión	Para <i>Apple</i> , <i>Android</i> , y aplicaciones web	Para <i>Apple</i> y <i>Android</i>
Presencia	No compatible	Presencia apoyada
Consultas	Consultas indexadas mediante clasificación y filtrado compuestos	Consultas profundas con funciones limitadas de clasificación y filtrado
Escrituras y transacciones	Operaciones avanzadas de escritura y transacción	Operaciones básicas de escritura y transacción
Fiabilidad y rendimiento	Alojamiento de datos en distintas regiones y tiempos de respuesta no superiores a 30 milisegundos	Bases de datos limitadas a la disponibilidad zonal dentro de la región determinada y tiempos de respuesta superiores a 10 milisegundos
Disponibilidad	99,999 %	99,95 %
Escalabilidad	Escalado automático	Fragmentación requerida
Seguridad	Reglas sin cascada que combinan autorización y validación	Lenguaje de reglas en cascada que separa autorización y validación

Antes de analizar cuál de las dos alternativas es más interesante para el desarrollo de este proyecto, se debe indicar cuáles son las características que se pretende conseguir en el MVP (Mínimo Producto Viable) de este proyecto. En primer lugar, uno de los requisitos que se pretende cumplir en este proyecto es alcanzar una disponibilidad del 99,9% del tiempo. Por otro lado, se pretende construir un sistema que ofrezca cierto grado de escalabilidad en el futuro ante el posible aumento del número de usuarios. Por último, se requiere que la base de datos proporcione los datos solicitados en un tiempo inferior a dos segundos. A continuación, se analizan ambas alternativas.

En cuanto al grado de disponibilidad que se requiere, ambas alternativas cumplen con lo esperado. Sin embargo, tal como se muestra en la Tabla 3, el porcentaje de tiempo disponible que ofrece *Cloud Firestore* es ligeramente superior a *Realtime Database*. Si se analiza cuál de las dos alternativas es más escalable, se puede observar que *Cloud Firestore* es preferible ya que se ofrece hasta un millón de conexiones simultáneas y diez mil escrituras por segundo frente a las doscientas mil conexiones y mil escrituras que se ofrecen en el caso de emplear *Realtime Database*. Además, se implementa un escalado automático, lo cual no es posible en la alternativa contraria. Por último, al igual que sucedía con la disponibilidad, ambas soluciones ofrecen tiempos de respuesta inferiores a 30 y 10 milisegundos respectivamente, lo cual hace que ambas soluciones cumplan con lo especificado anteriormente. Cabe destacar que ambas tecnologías emplean un modelo de datos no relacional (bases de datos NoSQL). Sin embargo, mediante el uso de documentos que se ofrece en *Cloud Firestore*, permite una organización más sencilla de datos complejos y jerárquicos. Una característica de la que no se beneficiaría en caso de que se optase por la otra opción.

Una vez analizadas ambas alternativas, se ha optado por utilizar *Cloud Firestore* porque ofrece el mayor grado de disponibilidad de las alternativas estudiadas y además dota a los sistemas software desarrollados de una mayor capacidad para escalar a cantidades de datos y de usuarios mucho mayores. No obstante, se obtendrían mejores tiempos de respuesta si se utilizase *Realtime Database*.

3.9. Solución propuesta

Tal y como se ha comentado en capítulos anteriores, la solución a desarrollar va a ser la siguiente.

En primer lugar, se va a emplear la arquitectura *software* definida en capítulos posteriores (véase el capítulo 4.1), la cuál es también la más recomendada según las guías oficiales de *Android*. Esta arquitectura constará de tres capas: una capa de persistencia, otra de dominio y otra de presentación. En cuanto a la capa de persistencia, es la que se encargará de recuperar cualquier tipo de dato solicitado de la base de datos, insertar nuevos datos, eliminar datos existentes y modificarlos. Por otro lado, la capa de dominio será utilizada por la capa de presentación (en concreto por los contenedores de estado) para poder mostrar distintos tipos de información al usuario. Cabe destacar que cualquier dato, antes de ser enviado a la capa de persistencia, será transformado al *Data Transfer Object* correspondiente. Por último, la capa de presentación será la encargada de mostrar los elementos gráficos, así como la información correspondiente al usuario.

En segundo lugar, se va a utilizar la solución *Cloud Firestore* de *Firebase* para almacenar los datos con los que se trabaje durante el ciclo de vida de la aplicación. Paralelamente, se utilizará *Firebase Authentication* para la gestión de los usuarios de la aplicación (véase el capítulo 4.4.3).

El conjunto de pruebas a realizar constará de pruebas unitarias, de accesibilidad, de integración y de regresión. Además, se realizará una prueba de usabilidad con posibles usuarios para evaluar el grado de usabilidad de la aplicación (véase el capítulo 6). Se empleará para ello *frameworks* orientados a las pruebas de software. Estos son *JUnit* y *Espresso* para ser exactos.

Para la implementación del sistema de alarmas que se integrará en la aplicación, se optará por diseñar una solución empleando la clase *AlarmManager* nativa de *Android* (véase el capítulo 3.8.1).

Por otro lado, se va a emplear el uso de un modelo de calidad para determinar si el producto *software* desarrollado cumple con los atributos de calidad básicos para una aplicación móvil como la desarrollada en este caso.

Para abordar el desarrollo de este proyecto, se va a emplear una metodología ágil dirigida por *sprints* (véase el capítulo 1.4) en los que se tratará de abordar todas las tareas planificadas previamente para cada uno de estos periodos de tiempo. En el caso de aquellas tareas relacionadas con la implementación de alguna funcionalidad, se realizarán las pruebas pertinentes para verificar su correcto funcionamiento.

3.10. Plan de trabajo

Tal como se muestra en la Tabla 4, el proyecto está compuesto de un total de 6 *sprints*. Cada *sprint* cuenta con una duración de 20 días, en los cuáles se trata de realizar todo el trabajo planificado previamente.

Cabe destacar que el Sprint 0 es distinto del resto, y es que dicho *sprint* está enfocado a realizar tareas previas al abordaje del proyecto tales como preparar el entorno de desarrollo y de las herramientas necesarias, realizar la estimación inicial de los elementos del *Product Backlog*, definir la arquitectura del producto *software* y planificar el trabajo a realizar en el *sprint* 1 entre otras.

Tabla 4.- Planificación de los sprints que conforman el proyecto

SPRINTS	FECHAS (ambas inclusive)
Sprint 0	14 de febrero – 5 de marzo
Sprint 1	6 de marzo – 26 de marzo
Sprint 2	27 de marzo – 16 de abril
Sprint 3	17 de abril – 7 de mayo
Sprint 4	8 de mayo – 28 de mayo
Sprint 5	29 de mayo – 18 de junio

Para estimar el alcance del proyecto, se ha tenido en cuenta las siguientes consideraciones:

- Se estima que el trabajo a desempeñar a lo largo del trabajo debe ser de trescientas horas (12 créditos x 25 horas/crédito = 300 horas).

- Por lo tanto, para cada *sprint* se le asigna un total de cincuenta horas.
- Se debe dejar cierto margen de tiempo en cada *sprint*, dando por hecho que, en alguna de las Unidades de Trabajo planificadas, la estimación inicial no será suficiente y se requerirá de más tiempo para finalizar alguna de ellas. Por consiguiente, se ha optado por dejar un 10% de la estimación teórica máxima en todos y cada uno de los *sprints*. Siguiendo estos cálculos, el tiempo dedicado a cada *sprint* sería de cuarenta y cinco horas.
- Esto equivaldría a tres horas diarias de dedicación durante cinco días a la semana.

Para cada unidad de trabajo existente en el *Product Backlog*, se realiza una estimación de las horas que trabajo que se necesitarán para su realización en el caso de que se vaya a realizar en el *sprint* próximo. Este número de horas debe comprender tanto la implementación de la funcionalidad en cuestión, así como la implementación de las pruebas pertinentes. Una vez estimadas las unidades de trabajo que conformarán un *sprint* determinado, se comprueba si la estimación total se ajusta al alcance previamente definido. En el caso de que se exceda, se trataría de reestimar algunas de las unidades de trabajo en cuestión de tal modo que la estimación total se ajuste al alcance.

3.10.1. Sprint 0

Para el *Sprint 0* de este proyecto, se han realizado las siguientes tareas:

- Creación del modelo de dominio de la aplicación.
- Preparación del entorno de desarrollo a utilizar durante el proyecto.
- Inicialización del repositorio de trabajo que almacenará todos los archivos que compondrán el proyecto.
- Definición de la arquitectura que seguirá el producto software a desarrollar.
- Análisis de aplicaciones similares a la propuesta.
- Selección de la base de datos de *Firebase* a utilizar.
- Formación en desarrollo en *Android*.
- Formación en *Firebase*.

Al tratarse de un *Sprint 0* del proyecto, no surgió ningún tipo de imprevistos ya que se estimó al alza la duración de muchas de estas tareas.

3.10.2. Sprint 1

El *Sprint 1* fue el primero de este proyecto en el que se empezó a implementar las funcionalidades especificadas en apartados anteriores. En concreto se implementaron las siguientes:

- Inicio de sesión.
- Cierre de sesión.
- Registro de nuevos usuarios.
- Establecer pautas de medicamentos.
- Recordar las pautas de medicamentos previamente establecidas.
- Mostrar consejos de salud al usuario cada vez que se inicia la aplicación.

Antes de empezar este *Sprint*, se planificó también la implementación de las siguientes funcionalidades:

- Reservar citas médicas.

- Mostrar la localización de farmacias cercanas.
- Indicar al usuario el grado de cumplimiento de las pautas establecidas.
- Registro de nuevos medicamentos.

Sin embargo, al haberse ajustado tanto a la capacidad de trabajo máxima y haber surgido imprevistos durante el desarrollo de algunas de las funcionalidades anteriormente citadas, siendo necesario reestimar las horas de trabajo requeridas, se tuvo que reducir el alcance de este *Sprint*.

3.10.3. Sprint 2

A lo largo del *Sprint 2*, se llegaron a implementar las siguientes funcionalidades:

- Carga de la interfaz de usuario correspondiente, según el rol del usuario autenticado (paciente o médico).
- Registro de nuevos medicamentos.
- Agregación de funcionalidad para seleccionar el tipo de dosis del medicamento en el proceso de creación de alarmas.
- Cancelación de alarmas y borrado de la base de datos.
- Selección de idioma (castellano, catalán o inglés).

No obstante, se planificó también para este *Sprint* las implementaciones siguientes:

- Localización de farmacias cercanas.
- Indicación de cumplimiento de pauta.
- Reserva de citas médicas.

Pero debido a la corta estimación de algunas unidades de trabajo y a los imprevistos que surgieron, se tuvieron que desplazar su respectiva realización al *Sprint* siguiente.

3.10.4. Sprint 3

A la finalización del *Sprint 3*, se implementaron las siguientes funcionalidades:

- Localización de farmacias cercanas dentro de un radio de 60 kilómetros.
- Reproducción de mensajes de audio cuyo objetivo es guiar al usuario durante la creación de alarmas.
- Modificación de las interfaces (véase el capítulo 6.4) con el objeto de mejorar la accesibilidad de la aplicación.
- Indicación del cumplimiento de una pauta en cuestión (nº de dosis notificadas contra el número de dosis administradas).

A diferencia de los *Sprints* anteriores, se empezó a implementar la funcionalidad relacionada con la reserva de citas médicas por parte de los usuarios pacientes, pero la implementación en cuestión no se concluyó en este *Sprint*, sino que se tuvo que terminar en el *Sprint* siguiente.

3.10.5. Sprint 4

A la finalización del Sprint 4, se han implementado las siguientes funcionalidades y, además, se han realizado las siguientes tareas:

- Se ha terminado de implementar la funcionalidad de reserva de citas médicas por parte de los usuarios. Esta funcionalidad también abarca la consulta de citas médicas por parte de los usuarios con rol de médico.
- Restablecimiento de alarmas al reiniciar el dispositivo.
- Realización de cuestionarios de usabilidad por parte de una cantidad razonable de usuarios.
- Aplicación del modelo de calidad seleccionado para este proyecto.
- Interacciones medicamentosas del usuario.

A partir de este Sprint, se ha decidido no implementar más funcionalidades y centrar todos los esfuerzos del último *Sprint* en mejorar la usabilidad de las interfaces de usuario, así como la calidad del software en general (buena mantenibilidad del código, modularidad alta, etc....).

3.10.6. Sprint 5

Tal y como se ha comentado en el apartado anterior, para este *sprint*, se decidió dedicar todo el esfuerzo a mejorar la calidad del producto software lo máximo posible. Con lo cual, no se ha llevado a cabo ninguna implementación de alguna funcionalidad nueva. Para ser exactos, las tareas llevadas a cabo durante este periodo han sido las siguientes:

- Aplicación de pruebas de regresión.
- Documentación del código fuente del producto software.
- Refactorización del código.
- Aplicación de las conocidas como Heurísticas de Nielsen para evaluar la calidad de las interfaces de usuario implementadas.
- Mejora de las interfaces de usuario.

Es aquí cuando finaliza el desarrollo de este proyecto.

4. Diseño de la solución

4.1. Arquitectura del sistema

A la hora de definir una arquitectura software, se debe tener en cuenta ciertos principios que se deben respetar con el objeto de asegurar solidez y facilidad de prueba a medida que el software va creciendo (implementando nuevas funcionalidades o bien corrigiendo defectos encontrados).

A continuación, se detallan todos y cada uno de estos principios, conocidos como principios SOLID [11]:

- **Principio de responsabilidad única** → Indica que cualquier clase existente en el programa debería ser responsable de solamente una cosa y, por lo tanto, debe tener una sola razón para cambiar.
- **Principio abierto/cerrado** → Exige que las clases del sistema deben ser aptas para ser extendidas y así poder implementar nuevos requerimientos, y a su vez, debe ser cerrada a cualquier modificación, es decir, poder alterar el comportamiento de una clase sin modificar su código fuente.
- **Principio de sustitución de Liskov** → Establece que cualquier clase existente en el sistema debe poder ser sustituida por cualquiera de sus subclases o clases hija sin alterar la corrección del software.
- **Principio de segregación de la interfaz** → Establece que es preferible disponer de una gran variedad de interfaces de ámbito específico que un escaso número de ámbito general. Así, cualquier clase del sistema no necesita conocer cómo realizar determinadas acciones que son irrelevantes para su desempeño.
- **Principio de inversión de la dependencia** → Indica que las clases de un sistema software deben depender exclusivamente de interfaces o bien, clases abstractas.

Es por ello por lo que se ha optado por una arquitectura multicapa ya que se separa las distintas responsabilidades existentes en el sistema en diversas capas que se describen seguidamente. Para las aplicaciones para *Android*, se debe disponer al menos de una capa de presentación que será la encargada de mostrar los datos de la aplicación al usuario y una capa de persistencia en la que está contenida la lógica de negocio de la aplicación y en la que se exponen los datos con los que se trabaja. Además, según las guías oficiales de *Android Developers*, es la más recomendada. No obstante, el desarrollador puede agregar una capa adicional a la arquitectura denominada como capa de dominio, en la que se simplifican las interacciones entre la capa de presentación y de persistencia descritas anteriormente. Un diagrama ejemplo de este tipo de arquitectura se puede observar en la Figura 27.

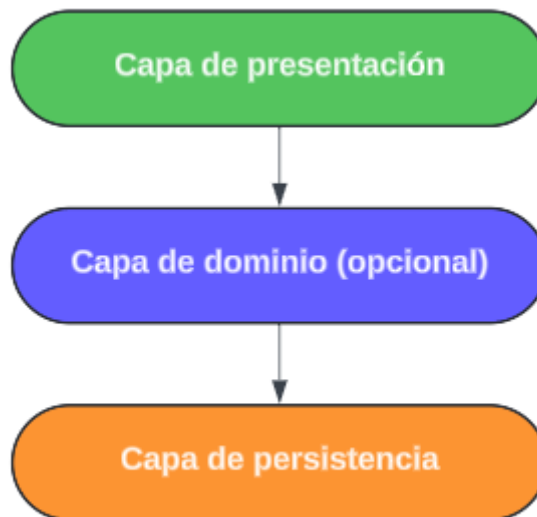


Figura 27.- Diagrama de la arquitectura de la aplicación. Imagen adaptada de “Guía de arquitectura de apps – Arquitectura de app recomendada” de la página web *Android Developers*

Para profundizar, se va a detallar cuáles son los elementos contenidos dentro de la capa de presentación de una arquitectura software perteneciente a una aplicación *Android* [12]. En primer lugar, se encuentran los elementos de interfaz tales como actividades, fragmentos, etc. Son los que se encargan de definir el diseño de cada una de las interfaces de usuario que conforman la aplicación en cuestión. Por otro lado, los elementos de interfaz previamente descritos están asociados a lo que se conoce como un contenedor de estado. Este elemento se encarga de retener los datos que necesita la interfaz a la que está asociado y controlan la lógica de negocio de la aplicación. El patrón que se sigue en la comunicación entre la capa de datos y los elementos de la IU y sus respectivos contenedores de estado es el siguiente:

- Cada contenedor de estado solicita a la capa de datos toda la información que se va a exponer posteriormente en su IU asociada.
- Durante el ciclo de vida de dicha IU se producen eventos de usuario que son notificados a sus respectivos contenedores de usuario. Estos eventos pueden ser muy variados (el usuario pulsa un botón, desliza hacia abajo para actualizar la IU, etc....).
- Los contenedores de estado, al ser notificados de estos eventos, actualizan el estado de la IU.
- Dicho estado es enviado a la IU para su renderización.

Tal como se muestra en la Figura 28 y en la Figura 29 las vistas se comunican con sus respectivos contenedores de estados ante cualquier evento producido en la interfaz de usuario.

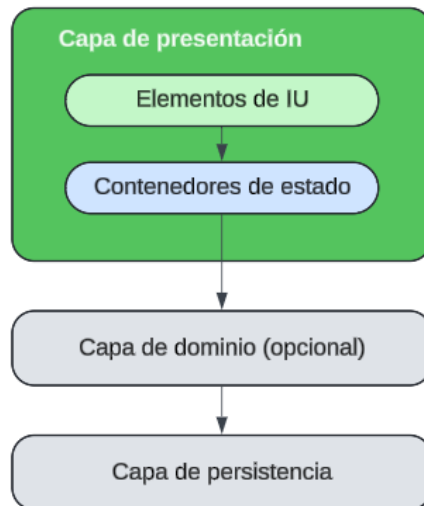


Figura 28.- Elementos contenidos dentro de la capa de presentación de la arquitectura. Imagen adaptada de "Arquitectura de app recomendada – Capa de la IU" de la página web *Android Developers*

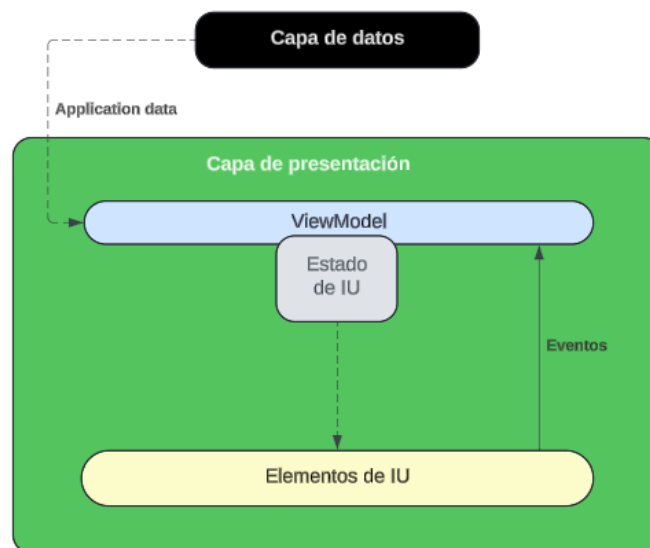


Figura 29.- Diagrama que muestra el funcionamiento del flujo unidireccional de datos en la arquitectura de la aplicación. Imagen adaptada del sitio "Capa de la IU - Contenedores de estado" de la web *Android Developers*

Tal como se ha comentado en este mismo apartado, se puede dar el caso en el que sea necesaria la adición de una capa de lógica de negocio. Esta capa es la responsable de encapsular la lógica de negocio de la aplicación en cuestión [13]. Esta capa es totalmente opcional, ya que, dependiendo del tipo de aplicación, puede no ser requerida. Uno de sus posibles usos sería encapsular la lógica de negocio compleja o lógica simple que se reutiliza en múltiples contenedores de estado. En la Figura 30, se muestra con qué capas interactúa.

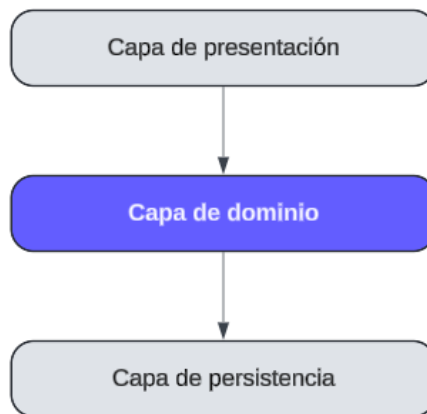


Figura 30.- Capa de dominio en la arquitectura de la aplicación. . Imagen adaptada de “Arquitectura de app recomendada – Capa de dominio” de la página web *Android Developers*

Por último, se procede a describir todos aquellos elementos que están contenidos dentro de la capa de datos de una arquitectura software de una aplicación *Android* [14]. En primer lugar, se encuentran las clases conocidas como repositorios. Estas clases son las encargadas de contener de cero a indefinidas fuentes de datos. Son las responsables de centralizar las modificaciones de los datos, abstraer fuentes de datos al resto de la aplicación y de exponer los datos al resto del sistema entre otras. Se debe crear un repositorio por cada tipo de datos distinto con el que se trabaje en el sistema. Por otro lado, las clases fuentes de datos son las encargadas de trabajar con una sola fuente de datos de entre las existentes en la aplicación. Estas fuentes de datos podrían tratarse de un archivo, una base de datos local, etc. Son el puente entre el sistema y la aplicación para cualquier operación de modificación, borrado u adición de datos. En la Figura 31, se muestra cómo estos dos tipos de clases se comunican entre sí, es decir, muestra como una clase repositorio actúa sobre una clase de fuente de datos.

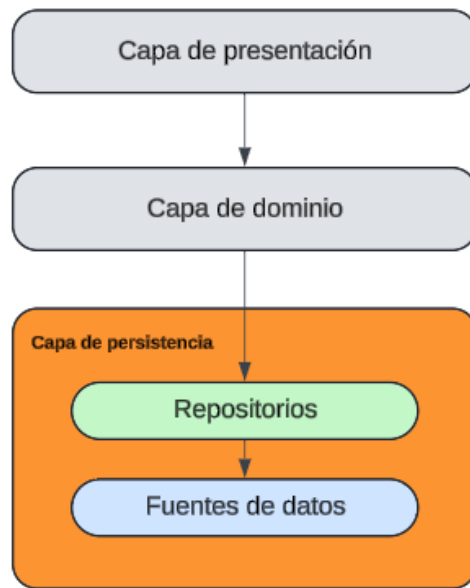


Figura 31.- Repositorios y fuentes de datos en la capa de persistencia de la arquitectura. Imagen adaptada de “Arquitectura de app recomendada – Capa de datos” de la página web *Android Developers*

En definitiva, este tipo de arquitectura es interesante para los equipos de desarrollo dado que mejora la mantenibilidad y calidad del software en general, mejora la escalabilidad de la misma, es decir, permite que cada vez más usuarios y más dispositivos hagan uso de la aplicación con conflictos de código mínimos, es mucho más fácil probar una aplicación que siga este tipo de arquitectura y además es mucho más fácil hallar posibles defectos en el código al separar responsabilidades en el sistema. Es por ello por lo que se va a emplear esta arquitectura a lo largo del desarrollo del proyecto. Más adelante se detallará cómo está estructurado.

4.2. Capa de presentación

En esta sección se tratará de mostrar los componentes que definen la capa de presentación del producto *software*. Para ello, se van a mostrar una serie de bocetos realizados previamente a la implementación de cada componente. Por consiguiente, no reflejan el arte final de la interfaz gráfica, ya que su función principal es guiar a lo largo del desarrollo de la interfaz.

A continuación, se muestran los bocetos realizados inicialmente de estas interfaces:

- Inicio de sesión (Figura 32 – a))
 - Se trata de un formulario que consta de dos campos de texto (uno para la dirección de correo electrónico y otro para la contraseña), cuya finalidad es que el usuario inicie sesión.
- Cierre de sesión (Figura 32 – b))
 - Permite al usuario cerrar la sesión y volver a la interfaz de inicio de sesión.
- Registro de nuevos usuarios (Figura 32 – c))
 - Permite a los usuarios no autenticados crear una nueva cuenta de usuario en el sistema a través de la cumplimentación de un formulario que recoge los datos del nuevo usuario.



a)



b)



c)

Figura 32.- Bocetos de las interfaces de usuario: a) Inicio de sesión b) Cierre de sesión y c) Registro de nuevos usuarios

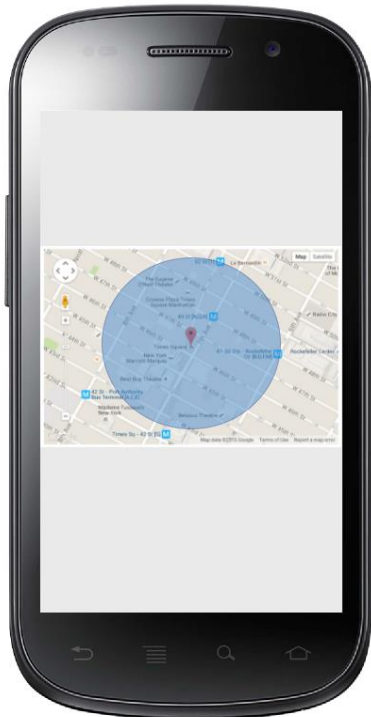
- Consejos de salud (Figura 33 – a))
 - Diálogo que se muestra al ejecutar la aplicación y que muestra un consejo enfocado a fomentar un estilo de vida saludable.
- Recordar pautas de medicamentos (Figura 33 – b))
 - Consiste en mostrar una notificación al usuario una vez se dispara una alarma previamente creada.
- Localización de farmacias cercanas (Figura 33 – c) y d))
 - Está formada por un mapa que contiene una serie de marcadores, en los que cada uno representa una de las farmacias más cercanas a la posición del usuario. Al pulsar en un marcador determinado, se muestra un bocadillo que contiene datos de la farmacia en cuestión (nombre, dirección, teléfono de contacto, etc....).



a)



b)



c)



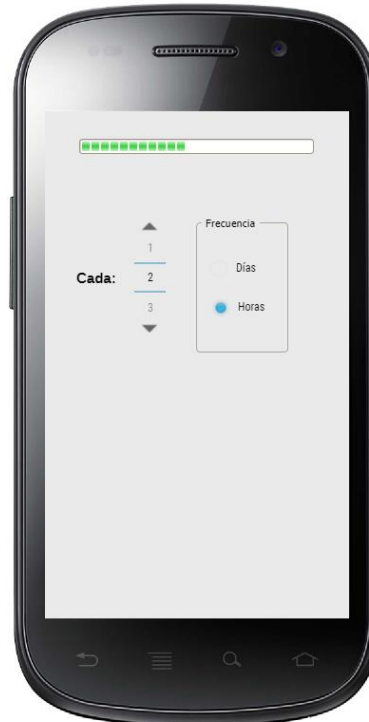
d)

Figura 33.- Bocetos de las interfaces de usuario: a) Consejos de salud b) Recordar pautas de medicamentos c) Localización de farmacias cercanas - Mapa y d) Localización de farmacias cercanas - Detalle de ubicación

- Establecer pautas de medicamentos (Figura 34)
 - Proceso en el cual se trata de crear una nueva pauta de un medicamento determinado. Para ello, se selecciona un medicamento y una dosis. Seguidamente, se selecciona la frecuencia de repetición de la alarma. Por último, se indica la hora de inicio a la que se debe empezar el ciclo de alarmas repetitivas. Para pasar de una interfaz a otra, se pretende agregar botones para avanzar o retroceder a fases anteriores.



a)



b)

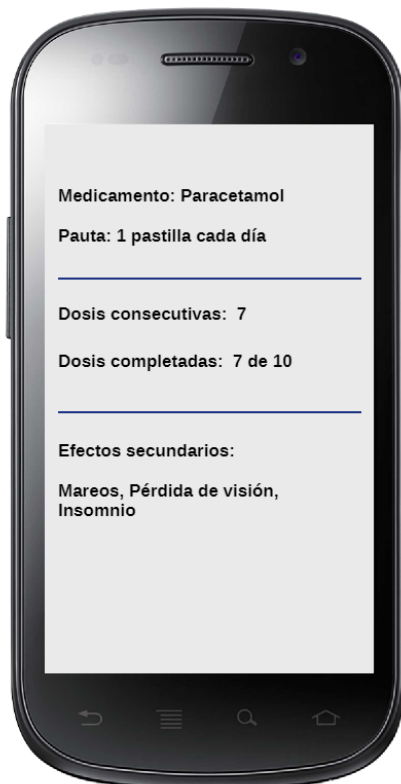


c)

Figura 34.- Bocetos de las interfaces de usuario: a) Selección de dosis b) Selección de frecuencia y c) Selección de hora de inicio

- Indicación de cumplimiento de pauta (Figura 35 – a))

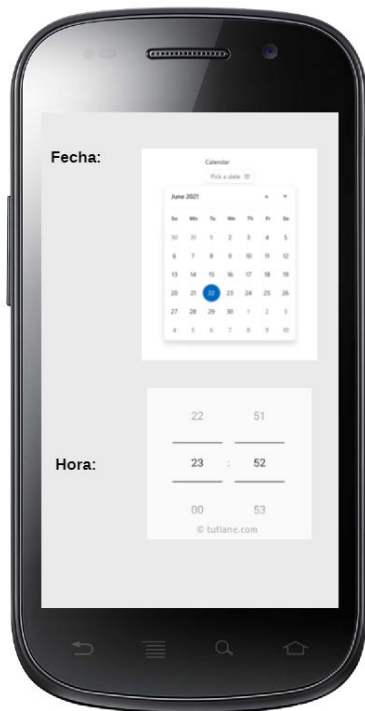
- Interfaz a la que se accede al pulsar sobre una alarma en cuestión. Se trata de una interfaz en la que se muestra un resumen de los datos de una alarma, entre ellos las dosis administradas consecutivas y el número total de dosis notificadas.
- Registro de medicamentos (Figura 35 – b))
 - Formulario para médicos en el que se trata de dar de alta un nuevo medicamento en el sistema.
- Reservar citas médicas (Figura 35 – c))
 - Se compone de un selector de fecha y otro para la hora, con el objeto de reservar una cita médica.
- Selección de tipo de dosis (Figura 35 – d))
 - Se muestra una lista de presentaciones de medicamento (pastillas, sobres, mililitros, etc....) y que el usuario seleccione una de ellas. Es una interfaz que puede ser complementaria al proceso de creación de una alarma.



a)



b)



c)



d)

Figura 35.- Bocetos de las interfaces de usuario: a) Indicación de cumplimiento de pauta b) Registro de medicamentos c) Reservar citas médicas d) Selección de tipo de dosis

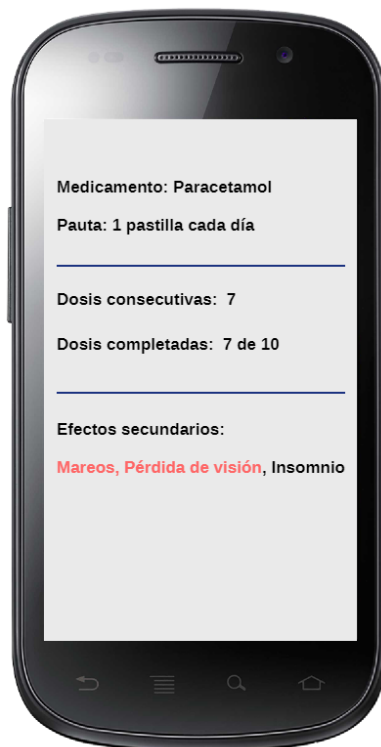
- Roles Paciente/Médico (Figura 36 – a) y b))
 - En función de si un usuario es un paciente o un médico, se carga el menú principal destinado a unos u a otros. Lo que cambiará en cada caso será la barra de navegación inferior, donde el usuario selecciona las distintas secciones de la aplicación a las cuáles se le otorga acceso.
- Interacciones medicamentosas (Figura 36 – c))
 - Cuando un usuario visualiza los detalles de una alarma determinada, se le muestra en color rojo aquellos efectos secundarios que son críticos para el paciente.



a)



b)



c)

Figura 36.- Bocetos de las interfaces de usuario: a) Roles paciente/médico - Interfaz de paciente b) Roles paciente/médico - Interfaz de médico y c) Interacciones medicamentosas

Por otro lado, también se procede a definir la navegabilidad entre interfaces a través de la Figura 37.

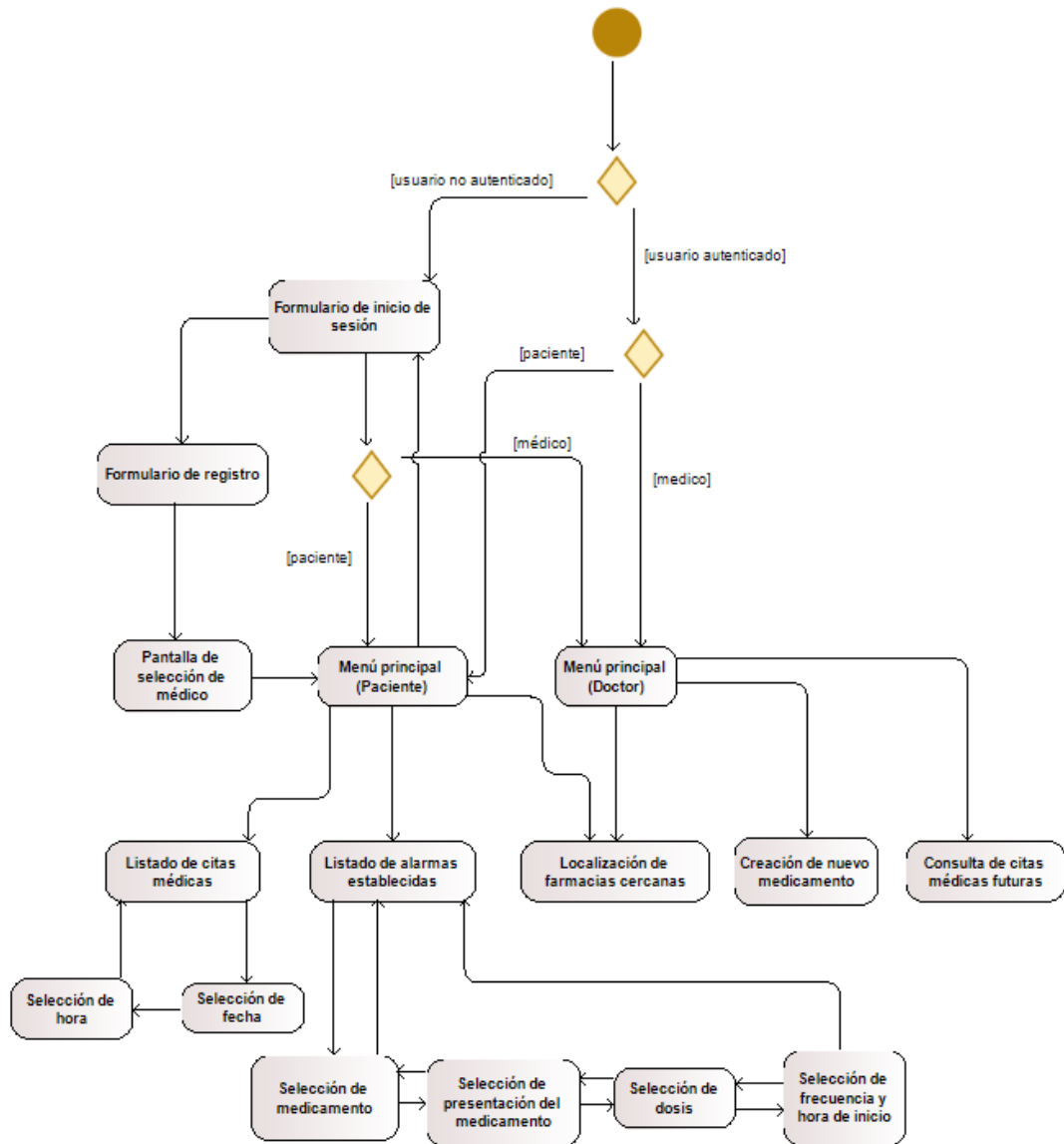


Figura 37.- Mapa de navegación entre las interfaces de usuario

Además de los bocetos mostrados previamente, también se ha creado un logo para la aplicación, cuyo aspecto se muestra en la Figura 38.

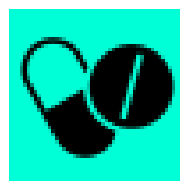

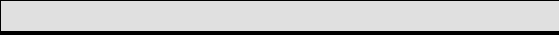










Figura 38.- Logotipo de MedsScheduler

Por otro lado, se han definido temas que se aplicarán en función del tema usado en el dispositivo del usuario (claro u oscuro). A continuación, en la Tabla 5, se muestran los colores que se han empleado.

Tabla 5.- Colores usados en los temas claro y oscuro de la aplicación

Tema claro	
Color principal de la marca	
Variante del color principal	
Color del texto sobre el color primario	
Color secundario de la marca	
Variante del color secundario	
Color del texto sobre el color secundario	
Color de la barra de estado	
Tema oscuro	
Color principal de la marca	
Variante del color principal	
Color del texto sobre el color primario	
Color secundario de la marca	
Variante del color secundario	
Color del texto sobre el color secundario	
Color de la barra de estado	

Por último, cabe destacar que se ha empleado un tamaño de entre veinte y veinticinco *sp* para el tamaño de letra de los textos mostrados en las interfaces que componen la aplicación.

4.3. Diseño detallado

En este capítulo se va a detallar cómo es la estructura interna del proyecto software desarrollado.

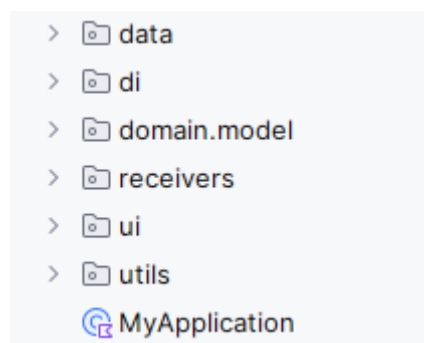


Figura 39.- Estructura de directorios en los que se organiza el código fuente de este proyecto software

Existen tres módulos principales en la aplicación. Estos son el módulo principal (cuya estructura se muestra en la Figura 39), el módulo de interfaces de usuario y el módulo de pruebas.

En primer lugar, el módulo principal de la aplicación se compone de los siguientes paquetes en función de la utilidad que tienen: *data*, *di*, *domain.model*, *receivers*, *ui*, y *utils*, además de la clase *MyApplication*.

A continuación, se procede a detallar el funcionamiento del paquete *data*. Es en este módulo en el que están contenidas todas aquellas clases e interfaces de las que se compone la capa de persistencia de la aplicación. Para cada tipo de dato existente en la aplicación, se define una serie de operaciones que se pueden realizar sobre dicho tipo de dato, reflejadas en la interfaz a implementar por cada posible fuente de datos. Además, se definirá un repositorio también en el que se solicitará a una o varias fuentes de datos los datos obtenidos para su posterior conversión al modelo de la aplicación. Un ejemplo de cómo está estructurado este paquete se muestra en la Figura 40.

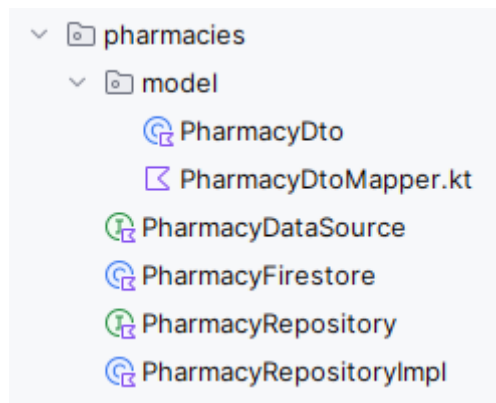


Figura 40.- Ejemplo de implementación en el paquete *data*

Todas las clases que gestionan la inyección de dependencias (véase el capítulo 4.6) están contenidas en el paquete *di*. Por cada clase u objeto que se quiera inyectar en la aplicación, se creará una nueva clase en este paquete y se implementará dicha inyección. A continuación, en la Figura 41, se muestra el contenido de este paquete.

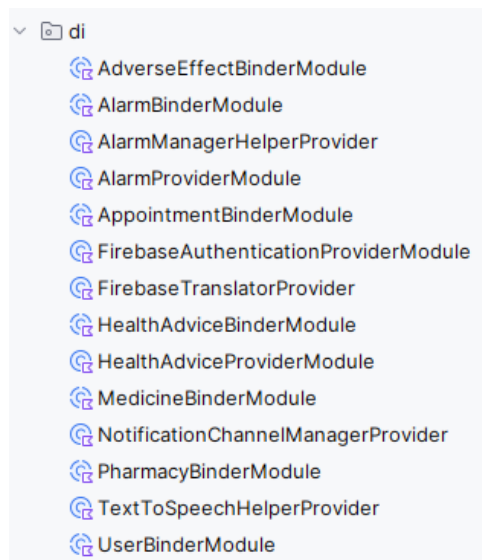


Figura 41.- Contenido del paquete *di*

Acto seguido, se describe el paquete *domain.model*. Es aquí donde se concentran todas las clases pertenecientes a la capa de dominio de la aplicación. Se tratan de clases de datos en las

que en cada una se especifica un nombre con el que identificar el objeto en cuestión y los atributos que lo definen. A continuación, en la Figura 42 se muestra su contenido.

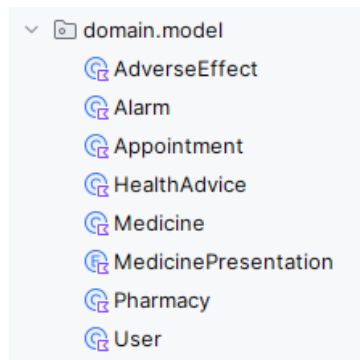


Figura 42.- Contenido del paquete domain.model

Consecutivamente, el paquete *receivers* es el que contiene todas las clases que implementan la interfaz *BroadcastReceiver* nativa de Android. La función de cada clase es realizar las acciones especificadas en su interior una vez se recibe una alerta determinada del sistema. De tal modo que actúan como un disparador. En la Figura 43 se muestra su contenido.

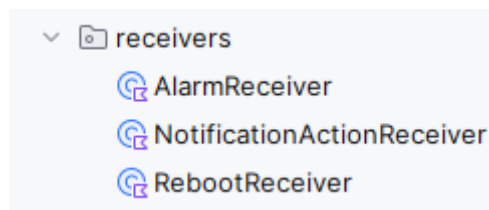


Figura 43.- Contenido del paquete receivers

A continuación, el paquete *ui* es aquel que contiene todas las actividades y *fragments* de la aplicación. Son las clases que gestionan el funcionamiento de las interfaces de usuario que se utilizan en el software en cuestión. En la Figura 44 se muestra su estructura.

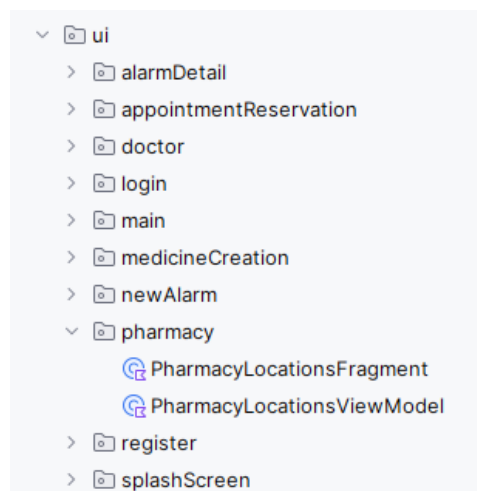


Figura 44.- Paquetes contenidos dentro del paquete ui, donde se muestra un ejemplo de Fragment y ViewModel

Después, en el paquete *utils* (Figura 45), es donde están contenidas todas aquellas clases que se consideran de uso general en toda la aplicación. Es aquí donde están contenidos los componentes en los que se gestionan aspectos como la creación de alarmas, la reproducción de mensajes de audio, la creación del canal de notificaciones, etc....

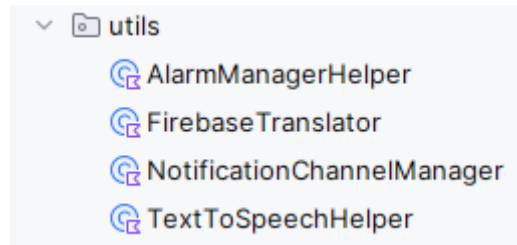


Figura 45.- Contenido del paquete *utils*

Por último, es en la clase *MyApplication* donde se centraliza la inicialización de componentes de la aplicación. Es la primera clase que se inicializa cada vez que se lanza el software a ejecución.

Después de haber explicado con detalle la organización del código fuente, cabe destacar la existencia de dos módulos también de interés en la estructura del código fuente del proyecto *software*. Estos módulos son el módulo que contiene los archivos en los que se define el diseño de las interfaces de usuario del producto software; y el módulo de *testing* (véase el capítulo 6).

4.4. Tecnología utilizada

Para desarrollar la aplicación en cuestión, se han utilizado diversas tecnologías con el objeto de implementar las funcionalidades descritas previamente. A continuación, se indican y se describen cada una de ellas.

4.4.1. Android Studio



Figura 46.- Logotipo de *Android Studio*

El entorno integrado de desarrollo (IDE) utilizado a lo largo del transcurso de este proyecto ha sido *Android Studio* (cuyo logotipo se puede observar en la Figura 46).

Android Studio es el IDE oficial utilizado para el desarrollo de software para *Android*. Además, está basado en el entorno, también muy popular, conocido como *IntelliJ IDEA* [15]. De entre las funciones más destacables que ofrece esta herramienta, se encuentran:

- Integración con Git (véase el capítulo 4.9).
- Entorno unificado para poder desarrollar aplicaciones que funcionen en una gran variedad de dispositivos *Android*.
- Creación de emuladores *Android* sobre los que depurar y ejecutar las aplicaciones en desarrollo.

4.4.2. Kotlin



Figura 47.- Logotipo de Kotlin

Para el desarrollo de esta aplicación, se ha empleado el lenguaje de programación conocido como *Kotlin*¹³ (cuyo logotipo se muestra en la Figura 47). Se trata de lenguaje de programación de tipado estático (se comprueba durante el tiempo de compilación) que es capaz de funcionar totalmente junto a código escrito en *Java*¹⁴ [16]. Desde el año 2019, Google se declaró *Kotlin First*, es decir, los nuevos proyectos se empezarían a escribir usando *Kotlin* y una vez finalizados se implementarían usando *Java*. Este hecho provocó que el sector del desarrollo móvil se centrara en este lenguaje, llegando así al 80% de las aplicaciones más populares en *Android* que funcionan con *Kotlin*.

Entre las características más destacables de este lenguaje de programación, se encuentran las siguientes:

- Posibilidad de trabajar con valores nulos.
- Curva de aprendizaje sencilla gracias a la simplicidad de su sintaxis.
- Integración del paradigma orientado a objetos (POO) y de la programación funcional en un único lenguaje de programación (funciones lambda).
- Uso de corrutinas para optimizar la programación asíncrona.
- Interoperabilidad total con *Java*.

¹³ <https://kotlinlang.org/>

¹⁴ <https://www.java.com/es/>

Por ello, *Kotlin* es considerado por Google como el lenguaje de programación principal para desarrollar aplicaciones para *Android* [17].

4.4.3. Firebase



Figura 48.- Logotipo de Firebase

Firebase¹⁵ (cuyo logotipo se muestra en la Figura 48) es una plataforma orientada al desarrollo web y al desarrollo de aplicaciones móviles que fue lanzada en el año 2011 y adquirida posteriormente por *Google* en el año 2014.

Su función principal es facilitar el desarrollo en la medida de lo posible a los programadores sin renunciar a la calidad que se suele requerir en los proyectos software. Dispone de una serie de utilidades de gran relevancia para los desarrolladores, entre otras:

- **Realtime Database:** Sirve para almacenar datos en una base de datos NoSQL en la nube.
- **Firebase Authentication** (véase el capítulo 4.4.3.1): Este producto ofrece servicios para permitir autenticación mediante contraseña o mediante número de teléfono entre otros.
- **Cloud Firestore** (véase el capítulo 4.4.3.2): Al igual que *Realtime Database*, este producto ofrece la posibilidad de almacenar datos en la nube. Destaca por ofrecer una base de datos flexible y escalable para aplicaciones móvil, web o desarrollo de servidores.
- **Crashlytics:** Es un producto para generar informes completos sobre fallos producidos en la aplicación.
- **Analytics Dashboard:** Permite a los desarrolladores consultar información muy detallada sobre la interacción de los usuarios con el producto software en cuestión.
- **Storage:** Herramienta de gran utilidad para desarrolladores que deseen almacenar contenido multimedia, ya sea fotografías o vídeos.
- **Cloud Messaging:** Solución que permite enviar mensajes a los clientes de una forma confiable y gratuita.

En el caso concreto de este proyecto, se han utilizado las herramientas de *Cloud Firestore* y *Firebase Authentication*.

4.4.3.1. Firebase Authentication

En una gran variedad de aplicaciones, es fundamental implementar un sistema de autenticación de usuarios para así garantizar la privacidad de los datos almacenados en la aplicación de cada

¹⁵ <https://firebase.google.com/>

uno de ellos. Por ello, para este proyecto se ha empleado el servicio de autenticación ofrecido por *Firebase* conocido como *Firebase Authentication*.

Firebase Authentication proporciona una serie de herramientas listas para usar y que sirven para autenticar a los usuarios en la aplicación a desarrollar [18]. Además, acepta autenticaciones mediante la aportación de una contraseña, un número de teléfono o bien mediante proveedores federados tales como *Google*, *Facebook*, *Twitter*, etc....

En concreto, en este proyecto se ha usado para implementar un sistema de autenticación mediante la aportación de una dirección de correo electrónico y una contraseña. Una vez autenticado el usuario, su dirección de email se usa para recolectar determinados datos de *Cloud Firestore*.

4.4.3.2. Cloud Firestore

La base de datos principal utilizada en este proyecto es *Cloud Firestore*. Se trata de una base de datos escalable y flexible para el desarrollo de aplicaciones para *smartphones* [19]. En esta base de datos se almacenará todo aquello que no esté relacionado con las alarmas que sean establecidas por los usuarios en la aplicación.

Además, cabe recalcar que es una base de datos NoSQL, es decir, que sigue un enfoque de diseño distinto al que se puede encontrar en las bases de datos relacionales [20]. En el caso concreto de *Cloud Firestore*, los datos están organizados en colecciones, las cuales contienen documentos. Un documento en el caso de las bases de datos de *Cloud Firestore*, es equivalente a un registro de una tabla en el contexto de las bases de datos relacionales.

Sus funciones más importantes son:

- **Flexibilidad:** Admite estructuras de datos jerárquicas, permitiendo así que un documento pueda contener otros documentos o subcolecciones.
- **Consultas expresivas:** Se pueden ejecutar consultas con el objeto de recuperar documentos de una colección determinada cuyos datos coincidan con los parámetros de la consulta realizada.
- **Actualizaciones en tiempo real:** La base de datos utiliza la sincronización de datos para actualizar los datos de cualquier dispositivo conectado a esta.
- **Asistencia sin conexión:** Esto significa que *Cloud Firestore* dispone de una caché que almacena datos que la aplicación usa de forma activa. Esto permite a la aplicación modificar y consultar datos, aunque el dispositivo móvil no disponga de conexión a Internet.
- **Gran escalabilidad:** Operaciones atómicas por lotes, replicación de datos por región, etc....

La selección de esta base de datos para el desarrollo del proyecto ha sido influenciada por las características anteriormente descritas, ya que son de mucho interés para asegurar al usuario una experiencia de uso cómoda y sencilla.

4.5. ML Kit Translator



Figura 49.- Logotipo de ML Kit

ML Kit (cuyo logotipo se muestra en la Figura 49) es un SDK utilizado en dispositivos móviles que trae el uso del aprendizaje automático para aplicaciones tanto para *Android* como para *iOS* [21]. Se trata de una herramienta muy útil para abordar un desarrollo de *software* de este tipo.

De entre las APIs que se ofrecen, se ha usado en concreto la API de traducción durante el desarrollo de este proyecto. Esta API ofrece la posibilidad de traducir textos entre más de cincuenta idiomas distintos, además de que no se requiere el envío de ningún texto a algún servidor remoto para ello. Los modelos de traducción utilizados han sido probados con la misma tecnología utilizada para el modo sin conexión de la aplicación *Google Traductor*.

Para acabar, se ha utilizado esta herramienta para la traducción de los textos contenidos en un consejo de salud determinado. Estos textos se traducen al idioma establecido en el dispositivo del usuario.

4.6. Hilt

Un problema muy común al que se enfrentan muchos desarrolladores de software es tratar de asegurar un bajo acoplamiento entre los distintos componentes con los que se trabaja en el proyecto software en cuestión (véase el capítulo 4.1). Para ello existe un principio que describe una serie de técnicas para reducir dicho acoplamiento. Este principio es conocido como inyección de dependencias [22]. Las técnicas que comprende este principio en cuestión son las siguientes:

- **Uso de abstracciones** en lugar de referencias directas a clases, facilitando el reemplazo de componentes.
- **Permitir a las clases recibir referencias de los componentes** que requiere, en lugar de instanciarlos directamente o a través de algún tipo de factoría.

Para reducir lo máximo posible el trabajo de inyectar dependencias de forma manual en el proyecto realizado, se ha usado *Hilt*¹⁶.

Hilt es una biblioteca destinada para desarrollos en *Android* cuyo objetivo es automatizar la inserción de dependencias [23]. De este modo, ya no resulta necesario crear manualmente cada clase y sus respectivas dependencias manualmente.

Esta biblioteca ofrece un método estándar que consiste en proporcionar contenedores para cada clase del proyecto en desarrollo y administrar automáticamente sus ciclos de vida.

¹⁶ <https://dagger.dev/hilt/>

Por otro lado, cabe destacar que *Hilt* se basa en otra popular biblioteca conocida como *Dagger*¹⁷, también orientada a automatizar la inserción de dependencias de proyectos software. No obstante, *Hilt* tiene los siguientes objetivos con respecto a *Dagger*:

- Ofrecer una infraestructura más simple para las aplicaciones de *Android*.
- Facilitar la configuración, legibilidad y la compartición de código entre distintas aplicaciones.
- Ofrecer al desarrollador una forma sencilla de aprovisionar diferentes vinculaciones para distintas compilaciones, como pruebas, lanzamiento o depuración.

En definitiva, *Hilt* es una herramienta muy útil para desarrolladores de *Android*, ya que permite reducir la cantidad de código resultante si se usara *Dagger* en su lugar y además facilita mucho la creación de objetos que dependen de otras clases para su respectiva creación.

4.7. Espresso



Figura 50.- Logotipo del framework Espresso

Uno de los tipos de pruebas que se considera que se deben realizar en un software que funcionará sobre *Android* es la prueba de interfaz de usuario, ya que verifica las posibles interacciones que el usuario puede realizar con cada interfaz presente en la aplicación a desarrollar (pulsar un botón, rellenar un campo de texto, etc...) [24]. Además, es interesante también realizar pruebas de flujo de usuarios que permitan abarcar las rutas de acceso más comunes por parte de los usuarios. Para abordar este tipo de pruebas, se ha optado por utilizar la herramienta conocida como *Espresso*.

Espresso, cuyo logotipo se muestra en la Figura 50, es un *framework* creado por *Google* cuyo objetivo es permitir a los desarrolladores escribir pruebas para la interfaz de usuario [25].

Este *framework* sincroniza automáticamente acciones de prueba con la interfaz de usuario de la aplicación en cuestión. Además, permite ejecutar pruebas tanto en dispositivos físicos como en dispositivos virtuales (emuladores).

¹⁷ <https://dagger.dev/>

En definitiva, *Espresso* está orientado especialmente a aquellos desarrolladores que consideran que la automatización de pruebas es una parte fundamental del ciclo de vida del desarrollo en *Android*.

4.8. JUnit

Todo proyecto software debe ser sometido a una serie de pruebas para así garantizar que el software que se entrega al cliente es de calidad, ya que mediante este proceso se puede identificar una serie de defectos introducidos durante el proceso de desarrollo y prevenir fallos [26].

Para abordar las pruebas a realizar en este proyecto software, se ha utilizado *JUnit*. Se trata de un *framework* dedicado a la automatización de pruebas, ya sean unitarias o de integración [27]. Principalmente, es utilizada en proyectos software escritos en el lenguaje de programación *Java*, aunque también existen versiones compatibles con otros lenguajes como *Kotlin*, *C++*, *Python*, etc....

4.9. Mantenimiento y Gestión de Versiones

Para el desarrollo del proyecto, es importante poder disponer de alguna herramienta que permita al equipo de desarrollo gestionar las distintas versiones que se generan de un programa determinado dentro de un proyecto software, ya que permitiría recuperar versiones específicas del programa en cualquier momento del transcurso del proyecto. Además, este tipo de herramientas permite controlar qué usuarios han modificado un archivo concreto y regresar a versiones anteriores del archivo entre otras.

Por ello, se ha optado por utilizar el software de control de versiones conocido como *Git*¹⁸ (cuyo logotipo se muestra en la Figura 51). *Git* nació como resultado de un desarrollo que inició la comunidad desarrolladora del *kernel* Linux a raíz de la decisión de la compañía desarrolladora de *BitKeeper* (el software utilizado por la comunidad anteriormente) de dejar de ofrecer su producto de forma gratuita [28]. Algunos de los objetivos que se propusieron para esta herramienta son:

- Velocidad.
- Sencillez.
- Soporte para desarrollo no lineal mediante la gestión de ramas paralelas.
- Completamente distribuido.
- Capacidad para albergar proyectos de gran envergadura eficientemente.

Las razones por las que se ha decidido utilizar *Git* han sido las siguientes. En primer lugar, casi todas las operaciones se realizan localmente, es decir, no es necesaria ningún tipo de información de otra máquina de la red para realizar la gran mayoría de operaciones. Puede darse el caso de que se realicen cambios en el proyecto en algún momento en el que el desarrollador no dispone de conexión a Internet, y que estos cambios se envíen al repositorio de código una vez se disponga de conexión. En segundo lugar, *Git* ofrece integridad ya que cualquier cambio es verificado mediante un *checksum* antes de ser almacenado, y a partir de ese momento, dicho cambio es identificado con dicho valor.

¹⁸ <https://git-scm.com/>



Figura 51.- Logotipo de Git

5. Desarrollo de la solución propuesta

En este capítulo, se trata de describir el desarrollo de la solución propuesta, es decir, de qué forma se ha pasado de la propuesta planteada a la solución final. Además, se van a describir también qué clases de problemas surgidos durante el desarrollo del proyecto, las dificultades encontradas, particularidades, etc.

5.1. Implementación de la creación de alarmas

Uno de los principales desafíos a afrontar a lo largo del desarrollo de software era la implementación del sistema de creación de alarmas para recordar las distintas pautas de medicamentos que el usuario pudiese establecer. A continuación, se describe cómo se ha implementado dicho sistema.

Antes de explicar el proceso de creación de alarmas, cabe destacar que el proceso de establecer una alarma se realiza en un método llamado *addAlarm*, el cuál recibe un objeto de tipo *Alarm* con los datos de la alarma a establecer. Dicho método se encuentra en una clase llamada *AlarmManagerHelper*, la cual se encarga de realizar operaciones relacionadas con las alarmas, como por ejemplo crearlas o cancelarlas.

En primer lugar, se crea un componente de tipo *Intent* el cual es el encargado de permitir una comunicación con el componente llamado *AlarmReceiver*. Este componente es el que se encarga de definir las acciones a realizar una vez se reciba una alarma determinada (crear la notificación a mostrar en este caso). Al objeto *Intent* mencionado anteriormente, se le adjuntan datos para ser enviados al componente receptor de las alarmas. Dichos datos son el identificador de la alarma y el nombre del medicamento en cuestión.

A continuación, se asigna un texto determinado a mostrar en la notificación que va a variar en función de la presentación de medicamento seleccionada previamente (pastillas, sobres, mililitros, etc.). Luego, se asigna al *intent* los datos relativos a la dosis y a la presentación del medicamento, ambos seleccionados durante el proceso de creación.

Acto seguido, se procede a transformar el identificador de la alarma en un entero de 32 bits, para así poder utilizarlo como identificador en otro tipo de *intent* que se describe más adelante. Dicho identificador se carga en una nueva variable la cual es de tipo *PendingIntent*. La función de variables de este tipo es muy similar al *intent* encargado de enviar los datos correspondientes al receptor de alarmas, solo que, en este caso, se otorga el permiso para poder ejecutar otro *intent* en el futuro, incluso en aquellos casos en los que la aplicación ya no está en ejecución. Para este caso concreto, la variable de tipo *PendingIntent* empleada, se encarga de iniciar el receptor de alarmas cuando se reciba una alarma determinada.

A continuación, en la Figura 52, se muestra el código que realiza todo lo explicado sobre el proceso de creación de alarmas hasta el momento.


```

// Create an Intent to broadcast when the alarm goes off
val intent = Intent(context, AlarmReceiver::class.java)

intent.putExtra(name: "idAlarm", alarm.id)
intent.putExtra(name: "medicineName", alarm.medicineName)

// Determine the presentation of the medicine
val medicinePresentation = when (alarm.medicinePresentation) {
    context.getString(R.string.pillAbrev) -> context.getString(
        R.string.showPill)
    context.getString(R.string.packetAbrev) -> context.getString(
        R.string.showPacket)
    context.getString(R.string.MLABrev) -> context.getString(
        R.string.showML)
    else -> ""
}

intent.putExtra(name: "medicinePresentation", medicinePresentation)
intent.putExtra(name: "dosage", alarm.quantity)

// Create a PendingIntent to be triggered when the alarm goes off
val alarmId = alarm.id.toInt()
val pendingIntent = PendingIntent.getBroadcast(
    context,
    alarmId,
    intent,
    PendingIntent.FLAG_MUTABLE
)

```

Figura 52.- Primera fase del proceso de creación de una alarma

Seguidamente, se obtiene una instancia del objeto *Calendar* a la que se le establece la hora y minuto de inicio, en la que la alarma deberá ser disparada según el intervalo establecido previamente una vez se alcance dicha hora y minuto.

Después, se habilita el componente *RebootReceiver* para cerciorarse de que las alarmas existentes en el sistema sean establecidas de nuevo siempre que el dispositivo sea reiniciado.

Por último, mediante la única instancia existente en la aplicación de la clase *AlarmManager*, se establece un intervalo de repetición de alarmas inexacto proporcionando los datos del calendario comentados previamente, la frecuencia de repetición previamente seleccionada y la variable de tipo *PendingIntent* creada para iniciar el receptor de alarmas.

En la Figura 53, se muestra la última parte de código encargado de la creación de alarmas.

```

// Set the calendar time for the alarm
val calendar = Calendar.getInstance().apply { this: Calendar
    set(Calendar.HOUR_OF_DAY, alarm.hourStart)
    set(Calendar.MINUTE, alarm.minuteStart)
}

// Enable the RebootReceiver to ensure alarms are reset on reboot
val receiver = ComponentName(context, RebootReceiver::class.java)
context.packageManager.setComponentEnabledSetting(
    receiver,
    PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
    PackageManager.DONT_KILL_APP
)

// Set the alarm using AlarmManager
alarmManager.setInexactRepeating(
    AlarmManager.RTC_WAKEUP,
    calendar.timeInMillis,
    alarm.frequency,
    pendingIntent
)

```

Figura 53.- Segunda fase del proceso de creación de alarmas

5.2. Modelo de calidad empleado

En el penúltimo *sprint* del proyecto, con el objetivo de evaluar la calidad del producto software desarrollado, se optó por utilizar un modelo de calidad.

Un modelo de calidad es un documento que recoge una serie de características relacionadas con el producto software que se deben cumplir para garantizar que el sistema desarrollado es de calidad. Para este proyecto se ha utilizado el modelo de calidad que se muestra en la documentación oficial de Android orientada para desarrolladores [29].

En este modelo se evalúa la calidad en función de las siguientes características propias de una aplicación para *Android*:

- **Experiencia visual:** La aplicación desarrollada debe proporcionar los patrones de interacción y de diseño visual propios de *Android* para garantizar una experiencia de usuario lo más intuitiva posible.
- **Funcionalidad:** La aplicación debe ser funcionalmente correcta, tal como estaba previsto.
- **Rendimiento y estabilidad:** El sistema debe ofrecer el rendimiento, compatibilidad, estabilidad y la capacidad de respuesta esperada por los usuarios.
- **Privacidad y seguridad:** La aplicación desarrollada debe gestionar los datos personales del usuario con el nivel de permisos adecuado.
- **Google Play:** Si procede, se debe asegurar de que la aplicación puede publicarse en la plataforma *Google Play*.

5.3. Heurísticas de Nielsen

Una vez se han diseñado e implementado todas las interfaces de usuario que conforman la aplicación, se ha procedido a evaluar su usabilidad, es decir, cómo de fácil es para el usuario interactuar con ellas. Para ello, se han empleado las conocidas como “Heurísticas de Nielsen”.

Las heurísticas de Nielsen son una serie de directrices de usabilidad orientadas a ayudar a los diseñadores a enfocarse en los elementos más esenciales que forman parte de una interfaz gráfica [30]. Las directrices que conforman este conjunto son las siguientes:

- **Visibilidad del estado del sistema:** La aplicación debe informar a los usuarios de su estado actual, dando una *feedback* apropiado en un intervalo de tiempo razonable.
- **Similitud entre el sistema y el mundo real:** El sistema debe de hablar un lenguaje entendible usando términos que resulten familiares para el conjunto de usuarios, en lugar de emplear términos más orientados a la naturaleza del sistema.
- **Control por parte del usuario y libertad:** Es común encontrar casos en los que los usuarios eligen funciones del sistema por equivocación y necesitan una “salida de emergencia” marcada para abandonar el estado no deseado sin tener que ir a través de un extenso diálogo.
- **Consistencia y cumplimiento de estándares:** No debe haber ningún tipo de ambigüedad entre las distintas acciones en el sistema. No se debe generar ninguna confusión al usuario y tampoco se debería preguntar si dos acciones provocan el mismo efecto.
- **Prevención de errores:** Hay que tratar de evitar todos aquellos errores a los que el usuario pueda llegar y que el programador prevé que el usuario va a cometer. Este tipo de errores deben resolverse de antemano.
- **Minimizar la carga de la memoria del usuario:** Se debe mostrar toda la información por la que el usuario ha pasado para que no tenga que volver atrás para recordar su selección. Las instrucciones que detallan cómo se usa el sistema deben ser fácilmente accesibles siempre que se desee consultarlas.
- **Flexibilidad y eficiencia de uso:** El sistema debe ofrecer cierta flexibilidad de uso tanto para usuarios novatos como para usuarios más experimentados.
- **Diálogos estéticos y diseño minimalista:** Los diálogos deben contener solamente información totalmente relevante. Cada unidad extra de información en un diálogo compete con unidades de información y disminuye su visibilidad relativa.
- **Ayuda para reconocer, diagnosticar y recuperarse de errores:** Los mensajes de error deben estar escritos en un lenguaje entendible para el usuario, evitando tecnicismos lo máximo posible, y deben siempre sugerir una solución o un camino de salida.
- **Ayuda y documentación:** A pesar de que es preferible usar el sistema sin disponer de ningún tipo de documentación, puede llegar a ser necesario dar ayuda a los usuarios. Cualquier tipo de información debe poder encontrarse fácilmente, enfocada en las acciones que debe realizar el usuario, listar los pasos exactos a seguir y no ser muy extensa.

Tras haber aplicado las directrices anteriormente descritas, se han encontrado los siguientes defectos en la aplicación:

- A la hora de establecer un recordatorio, se permite al usuario avanzar a través de las distintas etapas de creación de alarma, pero no se le muestra qué datos ha seleccionado previamente. Se debe añadir algún tipo de elemento que muestre toda esta información.
- Durante el proceso de creación de alarmas, no se muestra al usuario un identificador de la fase en la que se encuentra. Se debe añadir alguna etiqueta de texto u otro elemento gráfico que cumpla la misma función.
- A la hora de eliminar una alarma, no se indica explícitamente al usuario cómo se debe hacer. Para ello se debe añadir algún elemento gráfico (una flecha indicando la dirección para deslizar, por ejemplo) que ayude al usuario a intuir la acción a realizar.

5.4. Refactoring y análisis estático del código fuente

Una vez se ha terminado de realizar los trabajos de implementación de funcionalidades de la aplicación, se ha procedido a mejorar la estructura y la calidad del código fuente escrito a lo largo de los *sprints* realizados. Para lograrlo, se ha realizado un proceso de *refactoring* en toda la aplicación.

Cuando se habla de *refactoring*, se refiere a la técnica disciplinada usada para reestructurar una o diversas partes que conforman el código fuente de un sistema software, modificando su estructura interna sin verse alterado su comportamiento externo, es decir, el comportamiento visible para los usuarios [31].

Mediante esta técnica, se logra reducir el coste que supone agregar nuevas funcionalidades a la aplicación o resolver cualquier tipo de defecto que se pueda encontrar en el código fuente, repercutiendo positivamente sobre el grado de mantenibilidad del sistema desarrollado.

Adicionalmente, con el objetivo de complementar este proceso de mejora de la calidad del código fuente, se ha empleado también una herramienta de análisis estático. El análisis estático está compuesto de diversas comprobaciones realizadas sobre el código fuente sin necesidad de ejecutarlo [32]. A través de estas comprobaciones, se han podido encontrar defectos como por importaciones innecesarias de librerías externas, fragmentos de código inalcanzables o construcciones del lenguaje que podrían ser simplificadas.

Gracias a la combinación de ambas técnicas, se consigue un código fuente simplificado, de mayor calidad, más fácil de entender y, sobre todo, más fácil de mantener.

6. Pruebas

Con el objeto de ofrecer un producto software de calidad, este debe haberse sometido a una serie de pruebas para asegurar su correcto funcionamiento. A continuación, se describen todos los tipos de pruebas que se han aplicado al software desarrollado.

6.1. Pruebas unitarias

Uno de los tipos de pruebas más importantes que conforman las pruebas realizadas en este proyecto son las pruebas unitarias.

El objetivo principal de este tipo de pruebas es verificar el correcto funcionamiento de un bloque de código pequeño y aislado del resto de la aplicación (un método, una clase, etc....) [33]. En el caso concreto de una aplicación para *Android*, es interesante probar el correcto funcionamiento de los siguientes componentes:

- **Contenedores de estado:** ya que son los que almacenan el estado de una interfaz de usuario determinada. Es importante comprobar que la gestión de valores en su interior se realiza correctamente ante las diversas interacciones que se pueden producir en dicha interfaz.
- **Componentes de la capa de datos:** Es importante asegurar el correcto funcionamiento de aquellos componentes encargados de recuperar datos locales y remotos (especialmente los repositorios) y ofrecerlos a la capa de la lógica de negocio para su respectivo tratamiento.
- **Clases de la capa de dominio:** En este proyecto, se han realizado pruebas para comprobar la correcta conversión entre un objeto de tipo *Data Transfer Object*, el tipo de dato que se recibe de una fuente de datos determinada, y el objeto que representa el dominio de la aplicación.
- **Clases de utilidad:** Aquellas encargadas de realizar operaciones determinadas tales como manipulación de cadenas de texto, matemáticas, etc....

No obstante, a lo largo de este proyecto, se ha tratado de evitar probar funcionalidades cuyo código está muy relacionado con el uso de una biblioteca o *framework*. Este tipo de pruebas se deben evitar no solo en proyectos *software* para *Android*, sino para cualquier tipo de desarrollo, ya que estas pruebas lo único que verifican es el correcto funcionamiento de una biblioteca que ya ha sido probada, no del código del programa. En definitiva, las pruebas de un *framework* son una pérdida de tiempo.

6.2. Pruebas de integración

Las pruebas de integración son una clase de pruebas en las que se evalúa la forma en la que interactúan y operan varios módulos dentro de un sistema software [34]. Por otro lado, un módulo es un conjunto de componentes o clases que, en conjunto, ofrecen una serie de funcionalidades como si se tratase de una sola entidad.

En este caso, se han realizado este tipo de pruebas para evaluar la cooperación entre los contenedores de estado (véase el capítulo 4.1) y la capa de persistencia de la aplicación. Y es que cuando un contenedor de estado solicita realizar una operación a esta capa, se espera recibir los resultados en el formato esperado, que las acciones solicitadas funcionen correctamente y sumado a todo esto, que funcione sin ningún tipo de fallo.

En resumen, este tipo de pruebas son complementarias con las pruebas unitarias (véase el capítulo 6.1), ya que pueden existir casos en los que no solo sea interesante evaluar el funcionamiento de un componente en solitario, sino también evaluar el funcionamiento de una colección de componentes que trabajan entre sí para ofrecer la visión de un componente único al desarrollador.

6.3. Pruebas de usabilidad

Tal y como se comentaba en capítulos anteriores, la usabilidad es una característica clave en este producto software. Uno de los objetivos es ofrecer un software que sea fácil de usar para el mayor número posible de usuarios. Es por ello por lo que se han realizado una serie de pruebas conocidas como pruebas de usabilidad.

Las pruebas de usabilidad consisten en evaluar la medida con la que un producto se puede usar por un grupo determinado de usuarios con el objeto de conseguir satisfacción en el usuario al utilizar la aplicación [35]. En este caso, se ha definido un guion que recorre la mayor parte de funcionalidades que ofrece la aplicación y que deben seguir los usuarios durante la prueba. Posteriormente, a dichos usuarios se les ha propuesto realizar un cuestionario evaluando el grado de usabilidad que asignarían a cada una de las funcionalidades mostradas. En total, se ha realizado la prueba con un total de diez usuarios. A continuación, en la Tabla 6 se muestra el número de usuarios que han seleccionado cada opción en cada una de las funcionalidades a probar.

Tabla 6.- Resultados del cuestionario de usabilidad de la aplicación

	Totalmente de acuerdo	De acuerdo	Ni de acuerdo ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
Registro como paciente	9	1			
Iniciar sesión	10				
Consejos de salud	8	2			
Crear alarmas	7	3			
Mostrar información detallada de una alarma	5	4	1		
Borrar alarmas	6	1	1	1	1
Localización de farmacias	2	4	2	2	

Reservar cita como paciente	6	3	1		
Cerrar sesión	6	3	1		
Iniciar sesión como médico	8	2			
Consultar citas	10				
Añadir medicamentos	7	2	1		

A continuación, se muestra una tabla con el promedio de los resultados obtenidos, teniendo la opción “Totalmente en desacuerdo” un valor de uno, y la opción “Totalmente de acuerdo” de cinco.

Tabla 7.- Promedio de los resultados obtenidos por funcionalidad probada

Registro como paciente	4.9	Localización de farmacias	3.6
Iniciar sesión	5.0	Reservar cita como paciente	4.5
Consejos de salud	4.8	Cerrar sesión	4.5
Crear alarmas	4.7	Iniciar sesión como médico	4.8
Mostrar información detallada de una alarma	4.4	Consultar citas	5.0
Borrar alarmas	4.0	Añadir medicamentos	4.6

Tal y como se ha podido observar en la tabla anterior (Tabla 7), las funcionalidades que peor valoración han recibido han sido las siguientes: localización de farmacias y borrar alarmas. Se deduce que las dificultades encontradas a la hora de borrar una alarma se basan en que no se ha implementado ningún tipo de señal que ayude al usuario a deducir qué interacción debe realizar para borrar la alarma. Para mejorar la usabilidad para esta función, se mejorará la interfaz de usuario relativa al listado de alarmas para hacerlo más intuitivo. Por otro lado, en cuanto a la localización de farmacias cercanas, se debe a un fallo de implementación encontrado que estaba relacionado con la otorgación de permisos de ubicación por parte del usuario. Por lo tanto, se ha de mejorar dicha funcionalidad para que funcione de la forma más fiable posible.

Cabe destacar que las funcionalidades que se consideran más ligadas a los objetivos de este proyecto, es decir, los recordatorios de pautas, los consejos de salud y la reserva de citas médicas han obtenido una valoración casi perfecta (Tabla 7).

En definitiva, se considera que los resultados obtenidos han sido muy satisfactorios, a pesar de que hay ciertos aspectos a mejorar. Por lo tanto, se llega a la conclusión de que la aplicación desarrollada es sencilla de utilizar.

6.4. Pruebas de accesibilidad

Uno de los objetivos fijados anteriormente para este proyecto era asegurar la facilidad de uso del sistema mediante opciones de accesibilidad (véase el capítulo 1.2). Para asegurar el correcto cumplimiento de este objetivo, se ha optado por realizar una serie de pruebas de accesibilidad.

Concretamente, todos los esfuerzos se han centrado en adecuar el tamaño tanto del texto como de los elementos gráficos con los que el usuario interactúa durante el uso de la aplicación desarrollada y seleccionar determinadas combinaciones de colores que tengan una alta relación de contraste, ya que en el caso contrario es posible que las interfaces gráficas pierdan nitidez para ciertos usuarios. También se habrá incluido la etiqueta *"contentDescription"* para todas las imágenes, lo que permite que se lea dicho texto a las personas con problemas visuales que tengan activas las opciones de accesibilidad de su dispositivo.

En primer lugar, cabe destacar una aplicación móvil que ha sido empleada para detectar elementos cuyo diseño se podría mejorar para así mejorar también la accesibilidad de la aplicación desarrollada. Esta aplicación es conocida como *"Test de Accesibilidad"* (cuyo logotipo se muestra en la Figura 54).



Figura 54.- Icono de la aplicación "Test de Accesibilidad"

El funcionamiento de esta aplicación consiste en la realización de capturas de imagen de las distintas interfaces de usuario presentes en la aplicación a comprobar. A continuación, la aplicación analiza dichas capturas y finalmente se ofrece al usuario una serie de sugerencias de mejora de accesibilidad, resaltando el elemento a mejorar en cuestión. Se muestra un ejemplo de uso de la aplicación en la Figura 55.

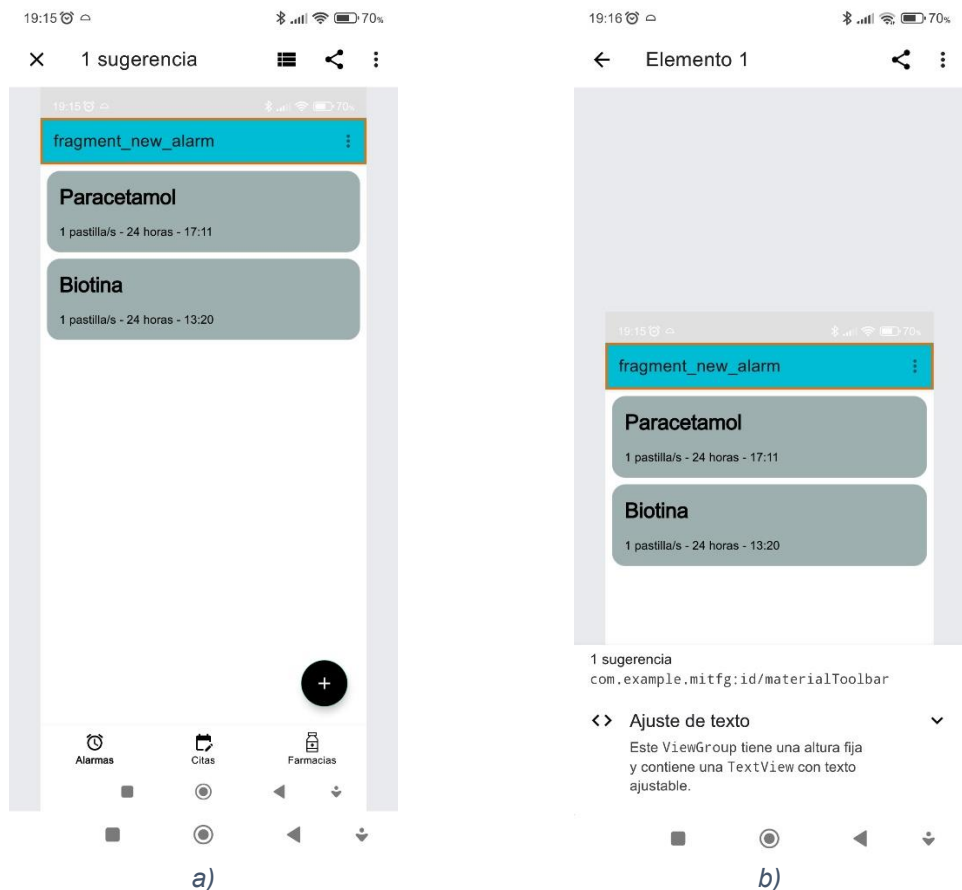


Figura 55.- Funcionamiento de la aplicación "Test de Accesibilidad" en la que: a) Se resalta el elemento cuya accesibilidad podría mejorarse, b) Se muestra el detalle de la sugerencia en cuestión

Además del análisis y de las correcciones realizadas gracias a la aplicación anteriormente descrita, también se han realizado pruebas con el *framework* conocido como *Espresso*, que además de permitir la interacción con los distintos elementos de la interfaz gráfica, permite también ejecutar verificaciones de accesibilidad en la aplicación.

Por último, se ha validado el diseño de la aplicación mediante la participación de una serie de personas a las que se les ha propuesto probar la aplicación y así obtener retroalimentación para implementar mejoras.

En definitiva, la accesibilidad es un punto clave en la aplicación propuesta y por ello se debe comprobar que se cumple un alto grado para ofrecer la mejor experiencia de uso posible al conjunto total de los usuarios finales.

6.5. Pruebas de regresión

Siempre que se realiza cualquier tipo de cambio en el proyecto software, ya sea la implementación de una nueva funcionalidad o la corrección de algún defecto encontrado en el código fuente, se debe verificar que las funcionalidades implementadas anteriormente en el proyecto en desarrollo siguen funcionando correctamente [36]. Previamente a la entrega del producto software desarrollado, se han realizado una serie de pruebas conocidas como pruebas de regresión.

Las pruebas de regresión son pruebas que están completamente enfocadas en verificar el correcto funcionamiento del sistema después de haber completado el trabajo de desarrollo. Pueden realizarse tanto de forma manual (ejecutando el sistema y comprobando su funcionamiento) como de forma automatizada (mediante ejecución de pruebas automatizadas) y son esenciales dentro del ciclo de vida del software para mantener la calidad de este.

La realización de estas pruebas se ha planificado para el último Sprint (véase el capítulo 3.10.6), con el objeto de comprobar el correcto desempeño del sistema. Además, se han combinado pruebas manuales y pruebas automatizadas realizadas con *Espresso* y *JUnit* (véase los capítulos 4.7 y 4.8).

7. Resultados obtenidos

Tras el desarrollo del proyecto, en este capítulo, se va a tratar de los resultados obtenidos relacionados con distintos procesos aplicados al proyecto.

7.1. Resultados de las pruebas del modelo de calidad

Para evaluar la calidad del sistema software desarrollado, se han realizado las pruebas indicadas de acuerdo con el modelo de calidad disponible en la documentación oficial de desarrollo en *Android* (véase el capítulo 5.2).

Sin embargo, no se han aplicado ciertas pruebas al considerarse que no proceden en este producto software en cuestión. A continuación, se presentan las pruebas no realizadas y el motivo por el cual se considera que no proceden en este caso:

Las pruebas relacionadas con el funcionamiento de la aplicación cuando está almacenada en algún almacenamiento externo no proceden, ya que la aplicación se instalará en la memoria principal del dispositivo, y no en ninguna tarjeta SD.

Por otro lado, tampoco se realizan pruebas relacionadas con proveedores de contenido registrados en la aplicación, ya que tampoco se emplea ninguno.

No se van a realizar pruebas relacionadas con elementos de tipo *WebView*, ya que, en las interfaces de usuario implementadas, no se usa ningún elemento de este tipo.

Por último, no proceden ninguna de las pruebas relacionadas con la publicación de la aplicación desarrollada en *Google Play*, ya que, al tratarse de un MVP, no se pretende distribuir a través de este canal. Es por este motivo por el que no se consideran este tipo de pruebas. En versiones futuras, conforme se vaya mejorando el producto *software*, sea a su vez más estable y se pueda poner en producción, entonces se procederá a abordar este tipo de cuestiones.

A continuación, en la Tabla 8, se muestran las pruebas realizadas, así como los resultados obtenidos inicialmente.

Tabla 8.- Resumen de las pruebas realizadas para evaluar la calidad del software y los resultados obtenidos

CR-3	Volver hacia atrás desde cada una de las pantallas.	Sí
CR-1	Ir a la página principal del dispositivo y reiniciar la aplicación.	Sí
CR-5	Rotar el dispositivo entre las orientaciones vertical y horizontal un mínimo de tres veces.	No
CR-9	Observar las notificaciones, expandirlas y presionar en todas las acciones disponibles	Sí
CR-0	Navegación entre todos los componentes de la aplicación (interfases, diálogos, etc....)	Sí
CR-2	Cambiar a otra aplicación en ejecución y volver a la aplicación, en todas y cada una de las interfaces del software.	Sí
CR-6	Comprobación de servicios en ejecución de la aplicación.	Sí

CR-7	Suspender y reactivar la pantalla del dispositivo.	Sí
CR-8	Bloquear y desbloquear el dispositivo.	Sí
CR-10	Compatibilidad con funciones de Descanso y <i>App Standby</i> .	Sí
SD-1	Repetición de pruebas en tarjeta SD del dispositivo.	No procede
SP-1	Compilación con última versión disponible del SDK.	Sí
SP-2	Comprobación de existencia de dependencias obsoletas.	No
SP-3	Detección de uso de interfaces no pertenecientes al SDK.	Sí
PM-1	Repetición de pruebas con la creación de perfiles de <i>StrictMode</i> habilitada.	Sí
BA-1	Repetición de pruebas en los ciclos de Descanso y <i>App Standby</i> .	Sí
SC-1	Revisión de todos los datos guardados en el almacenamiento externo.	Sí
SC-2	Revisión de cómo se controlan y procesan los datos que se cargan desde el almacenamiento externo.	No procede
SC-3	Revisión de proveedores de contenido en el archivo de manifiesto de <i>Android</i> .	No procede
SC-4	Revisión de los permisos de la aplicación y del tiempo de ejecución.	Sí
SC-5	Revisión del estado de exportación de todos los componentes definidos en el archivo de manifiesto de <i>Android</i> .	Sí
SC-6	Revisión de la configuración de seguridad de red de la aplicación.	Sí
SC-7	Para cada <i>WebView</i> , navegación a una página que requiera JavaScript.	No procede
SC-8	En cada <i>WebView</i> , navegación a sitios y contenido que la aplicación no cargue de forma directa.	No procede
SC-9	Declaración de una configuración de seguridad de red que inhabilite el tráfico de texto simple.	Sí
SC-10	Observación del registro del dispositivo mientras se prueban todas las funcionalidades. No se debe filtrar información privada del usuario.	Sí
GP-1	Comprobación de los datos de la aplicación en <i>Google Play Console</i> .	No procede
GP-2	Descarga del gráfico de funciones y las capturas de pantalla, y reducción de su tamaño de modo que coincidan con el de la pantalla de los dispositivos y factores de forma objetivo.	No procede
GP-3	Revisión de materiales incluidos en la descarga del <i>software</i> .	No procede

Tal y como se ha podido comprobar en la Tabla 8, no se han superado las pruebas con código CR-5 y SP-2. El motivo por el que no se han superado son los siguientes: no se había contemplado la posibilidad de usar la aplicación en modo apaisado y, por otro lado, algunas de las dependencias utilizadas en el proyecto no se utilizan en su versión más reciente. Para superar estas pruebas, se van a realizar las siguientes acciones. Por un lado, se procederá a crear, para cada interfaz de la aplicación, su versión a mostrar cuando el dispositivo esté orientado horizontalmente. Y por el otro lado, se actualizarán todas las dependencias existentes en el proyecto a su versión más reciente.

7.2. Funcionalidades principales de la aplicación

En esta sección se pretende presentar todas las funcionalidades implementadas a lo largo del desarrollo de este proyecto. Para ello, se van a presentar ejemplos prácticos para entender lo mejor posible el uso de cada funcionalidad. Las funcionalidades que se van a mostrar son las siguientes: inicio de sesión, registro, cierre de sesión, visualización de consejos de salud, creación de alarmas, visualización de los detalles de una alarma, reserva de citas médicas, localización de farmacias cercanas y creación de nuevos medicamentos. Todas las interfaces que componen la aplicación se pueden encontrar en los anexos (véase el capítulo 10.4).

En primer lugar, se procede a explicar las funcionalidades de inicio de sesión, registro y cierre de sesión. Por ejemplo, un usuario llamado Bob pretende crearse una cuenta de usuario dentro del sistema. Para ello, el usuario rellena el formulario de registro con sus datos (Figura 56 – a)). Seguidamente, en caso de no tratarse de un médico, Bob procede a seleccionar un médico (Figura 56 – b)). Si se tratase de un médico accedería directamente al menú principal de la aplicación. Una vez ha accedido, tiene la posibilidad de cerrar sesión (Figura 56 – c)), para volver a la página de inicio de sesión e introducir sus credenciales (Figura 56 – d)).

17:04

Registrarse

Nombre:
Bob

Apellidos:
Gutiérrez

Email:
bobgutierrez@gmail.com

Contraseña:
.....

¿Es usted médico?

REGISTRARSE

a)

17:07

Selecciona un médico

Rodrigo Pineda

Marcos Gálvez

Adolfo Martínez

Claudia Valiente

b)

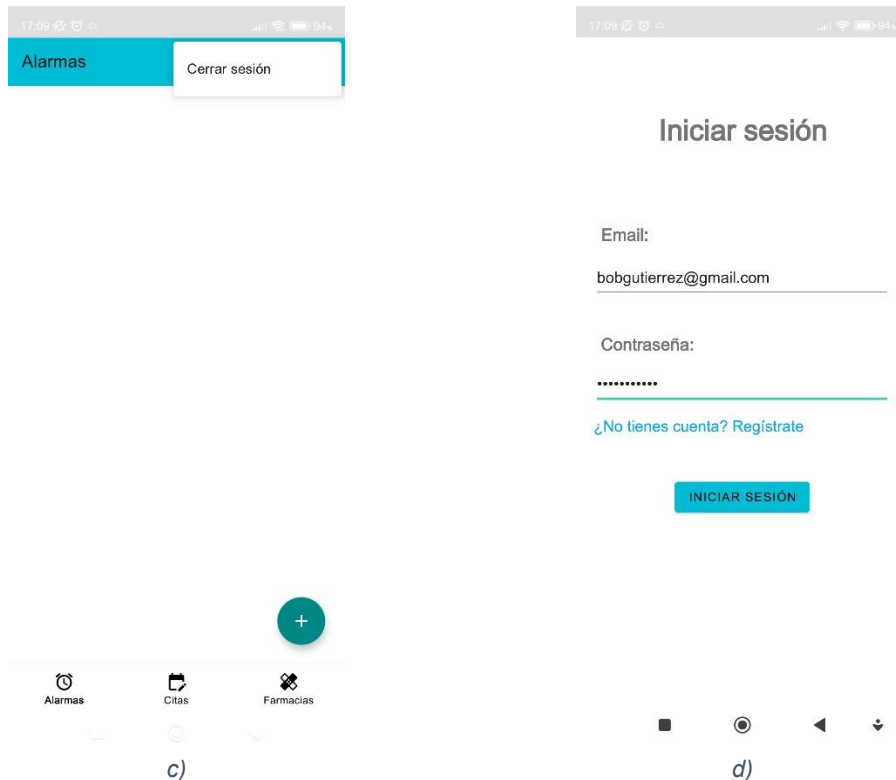
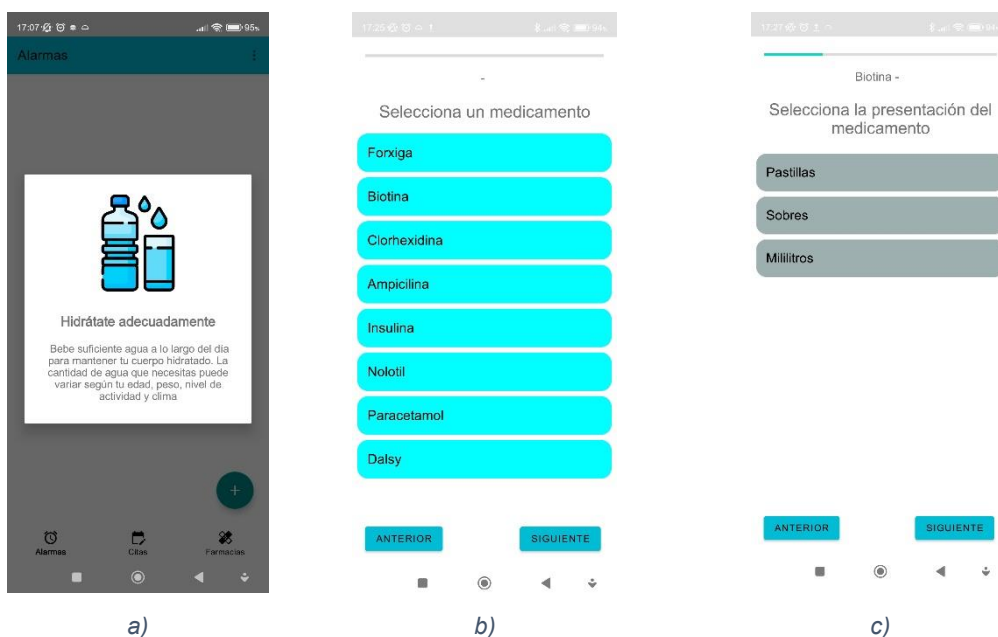


Figura 56.- Funcionalidades de la aplicación: a) Registro de nuevos usuarios, b) Selección de médico, c) Cierre de sesión, d) Inicio de sesión

Una vez Bob ha accedido al menú principal, se le muestra un consejo de salud aleatorio de entre los existentes en el sistema (Figura 57 – a)). Ahora, Bob quiere registrar una alarma de tal modo que le recuerde que se debe de tomar una pastilla de biotina cada 8 horas. Para ello, pulsa en el botón +, presente en la sección de alarmas del menú principal. A continuación, selecciona la biotina como medicamento de la alarma (Figura 57 – b)). Seguidamente, selecciona “Pastillas” como la presentación del medicamento a administrar (Figura 57 – c)). Después, selecciona la dosis indicada (Figura 57- d)). Por último, selecciona la frecuencia de repetición de la alarma y la hora de inicio (Figura 57 – e)). Cuando la alarma se dispara se muestra la notificación correspondiente (Figura 57 – f)).



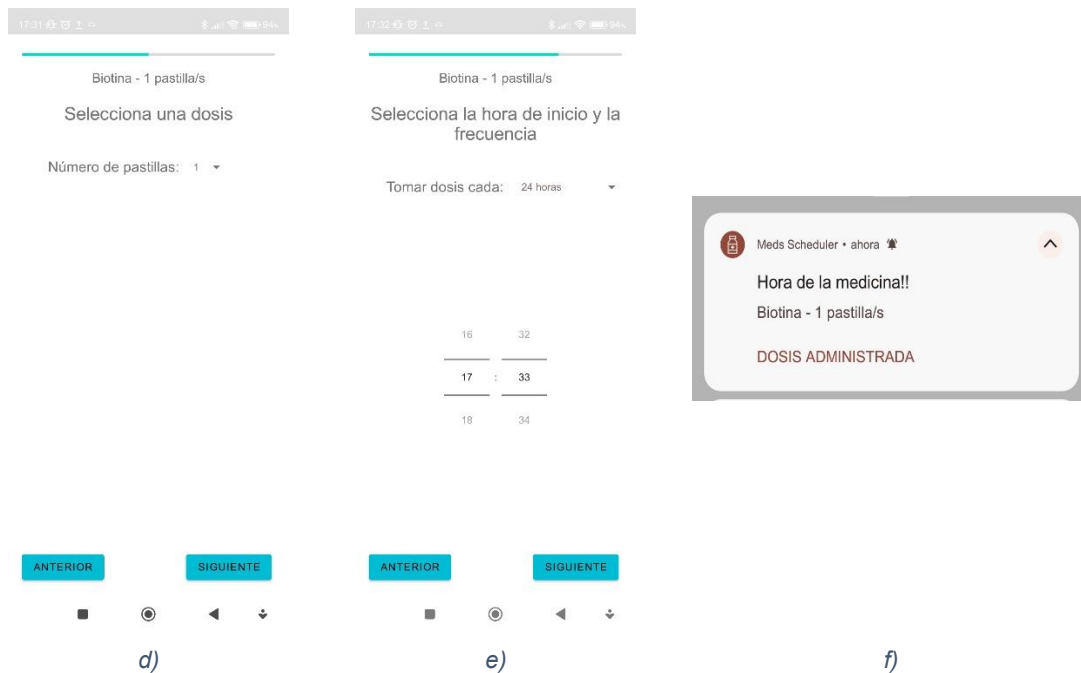
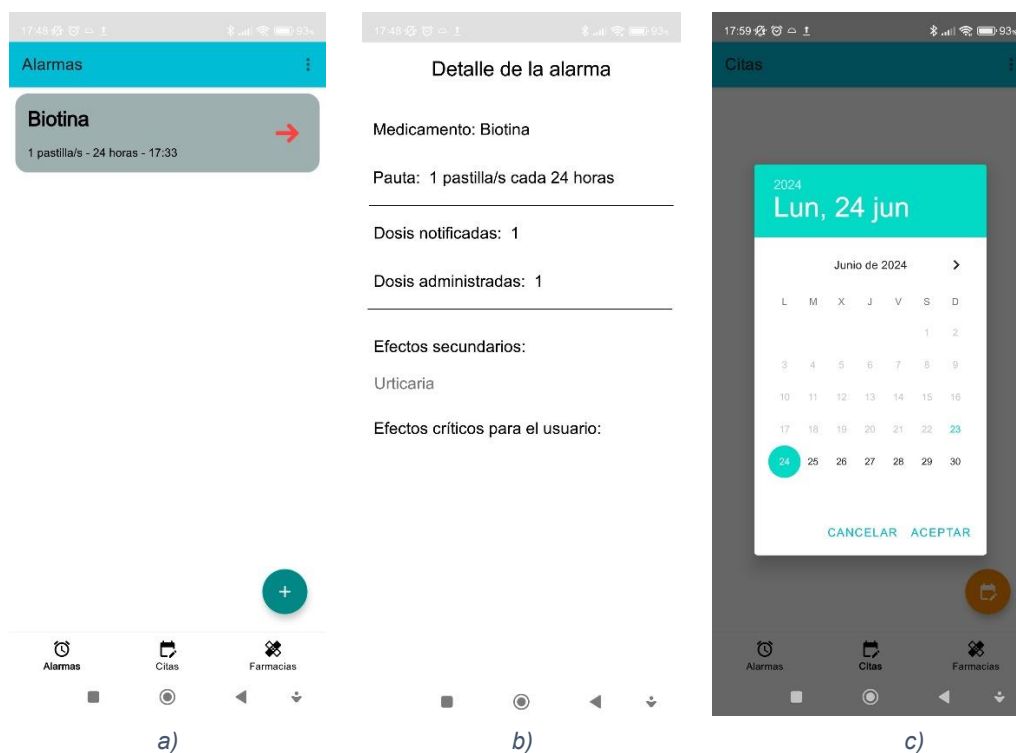
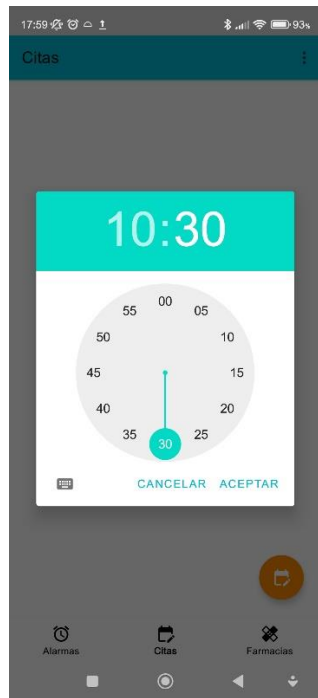


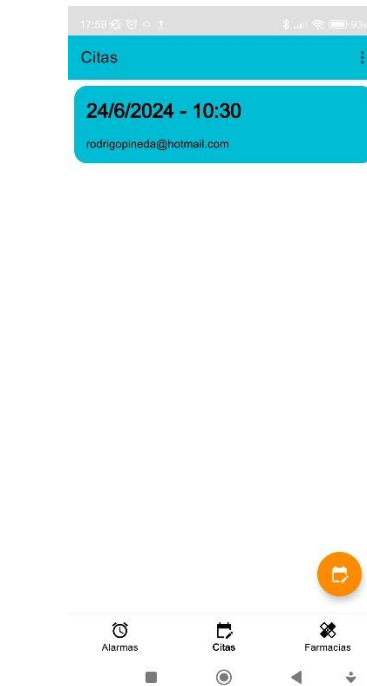
Figura 57.- Funcionalidades de la aplicación: a) Consejo de salud, b) Selección de medicamento, c) Selección de presentación del medicamento, d) Selección de dosis, e) Selección de hora y fecha de inicio y f) Notificación de la alarma

Si Bob indica en la notificación que se ha administrado la dosis, se añadirá al número de dosis administradas de la alarma. Esta información la puede comprobar pulsando en la alarma en cuestión (Figura 58 – a) y b)). A continuación, Bob desea reservar una cita con su médico. Para ello, se dirige al apartado de “Citas” y pulsa sobre el botón del calendario. Finalmente, selecciona una fecha y una hora de cita (Figura 58 – c), d) y e)).





d)



e)

Figura 58.- Funcionalidades de la aplicación: a) Listado de alarmas, b) Detalle de alarma, c) Selección de fecha, d) Selección de hora, e) Listado de citas médicas

Ahora Bob necesita consultar las farmacias más cercanas a su posición, ya que necesita ir a adquirir algunas unidades de biotina. Para ello, se dirige a la sección de farmacias, otorga los permisos necesarios a la aplicación (Figura 59 – a)) y se le mostrarán una serie de marcadores en los que cada uno representa una farmacia distinta. Para saber más información de una farmacia, basta con clicar sobre uno de estos marcadores (Figura 59 – b)).



a)



b)

Figura 59.- Funcionalidades de la aplicación: a) Diálogo de otorgación de permisos de localización y b) Muestra de datos de una farmacia

A partir de este punto, se va a tratar con otro tipo de ejemplo. Alice es una doctora que quiere empezar a trabajar con el sistema desarrollado. Para ello, se registra en el sistema con un perfil de médica. En primer lugar, Alice quiere añadir la loratadina al conjunto de medicamentos registrados en la base de datos. Es por ello por lo que accede a la sección de medicamentos, clicando en el botón de añadir medicamentos y rellena el formulario con los datos necesarios (Figura 60).

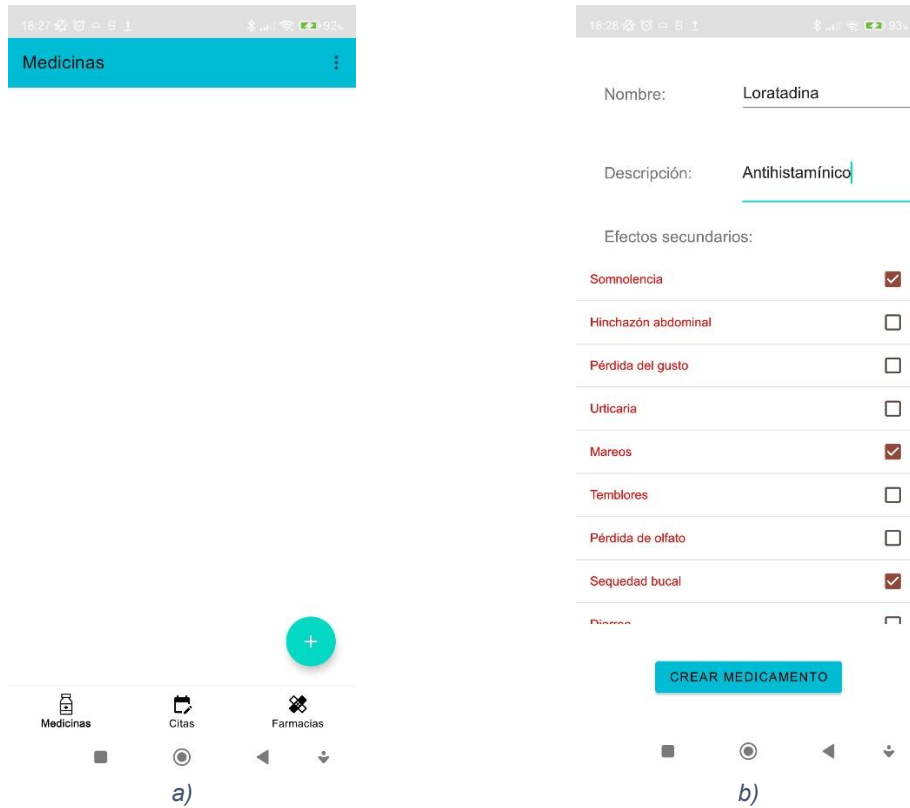


Figura 60.- Funcionalidades de la aplicación: a) Sección de medicinas y b) Formulario de registro de nuevas medicinas

En definitiva, se ha mostrado un recorrido por las funcionalidades principales de la aplicación a través de ejemplos sencillos donde Alice y Bob han sido los protagonistas.

8. Conclusiones

A lo largo del desarrollo de este proyecto, se ha tratado de implementar las funcionalidades propuestas según el orden de prioridad asignado anteriormente (véase el capítulo 3.2). De entre las funcionalidades que se ha propuesto implementar, se ha logrado implementar todos los de máxima prioridad, los que se consideraba que deberían estar presentes e incluso alguno de los que se consideraba que podrían incluirse. Sin embargo, se han dejado por implementar algunos de los requisitos propuestos. Estos son los siguientes: escaneo del código de barras del medicamento, notificaciones de recogida de medicamentos según receta, notificaciones de finalización de recetas, *newsletter* sobre salud y perfiles familiares.

En cuanto a los objetivos previamente fijados del proyecto (véase el capítulo 1.2), se considera que se ha cumplido con todos los objetivos propuestos en mayor o menor medida. A través del sistema de citas médicas implementado, se ha ofrecido un sistema que actúa como punto en común entre pacientes y médicos. Por otro lado, se considera que se ha cumplido totalmente con los objetivos de proporcionar un sistema de recordatorios de administración de medicamentos y diseñar la aplicación de tal modo que sea lo más accesible posible. Por último, se considera que, a través de la funcionalidad de consejos de salud, se ayuda al conjunto de usuarios a tomar acciones con el objetivo de implantar un estilo de vida más saludable. Es por ello por lo que se considera que esta funcionalidad es la que cumple el objetivo relacionado con el fomento de un estilo de vida saludable.

A continuación, se procede a comentar las dificultades encontradas a lo largo del transcurso del proyecto. En primer lugar, las principales dificultades encontradas se hallan en las estimaciones realizadas para cada *sprint* llevado a cabo. Fue complicado finalizar todas las tareas planificadas para cada *sprint*, provocando el traslado de tareas a *sprints* posteriores, y a su vez, un atraso en el desarrollo de la aplicación y obligando a abordar ciertas tareas de una forma más conservadora con el objetivo de cumplir con las expectativas esperadas para el MVP. Para solventar esta dificultad, se optó por reducir la capacidad de trabajo máxima teórica para así poder completar una mayor proporción de las tareas planificadas de una forma más fácil.

Por otro lado, a la hora de obtener datos de la base de datos de forma asíncrona, surgió un problema que consistía en que, al cargar ciertas interfaces de usuario, se mostraban dichas interfaces en pantalla, pero no se mostraban los datos que se esperaba obtener. Para solucionarlo, se diseñó la carga de los datos en las interfaces de usuario de tal modo que se comprobase constantemente si los datos se habían recibido. Si ese era el caso, se actualizaba el contenido de la interfaz con los datos obtenidos.

A continuación, cabe destacar que también se encontraron dificultades a la hora de implementar la funcionalidad relacionada con la localización de las farmacias más cercanas. En un principio, se pretendía realizar una integración de datos a través de distintas fuentes de datos que recogiesen los datos de las farmacias de una región determinada (Comunidad Valenciana, Madrid, etc.). Al no encontrarse fuentes de datos fiables y teniendo en cuenta que se tendría que desarrollar algún *script* de carga de datos, se optó por establecer una serie de datos de prueba, al menos, para el MVP. Tal y como se comenta en los posibles desarrollos futuros (véase el capítulo 8.2), una de las posibilidades es abordar esta problemática de forma más precisa, estableciendo desde un primer momento las fuentes de datos a emplear y diseñando el proceso de carga de datos, cuál si de un proyecto software distinto se tratase.

En resumen, considero que se ha aprendido muchísimo sobre las posibilidades que se ofrecen para desarrollar una aplicación en *Android* (facturación, publicación en plataformas de *Google*, interactuar con los distintos componentes integrados en un dispositivo móvil, etc.), ya que la realización de este proyecto me ha obligado a analizar todas y cada una de las distintas posibilidades que se han presentado para llevar a cabo una tarea concreta. Además, se considera que se han cumplido con los objetivos previamente establecidos (véase el capítulo 1.2). En cuanto a aprender las bases necesarias para desarrollar una aplicación móvil, cabe

destacar que no ha sido un problema grave, ya que muchos de los conocimientos aplicados fueron aprendidos durante el curso de la asignatura “Desarrollo de Aplicaciones para Dispositivos Móviles” (véase el capítulo 8.1). Aun así, se ha profundizado mucho más en este ámbito por cuenta propia.

8.1. Relación del trabajo desarrollado con los estudios cursados

Durante el desarrollo de esta aplicación, se han utilizado conocimientos adquiridos a lo largo de la realización de los estudios realizados. Dichos conocimientos se relacionan directamente con las siguientes asignaturas:

- **Proceso de Software (PSW):** El uso de una metodología ágil para el desarrollo de este trabajo vino motivado por el aprendizaje y la aplicación de la filosofía de dicha metodología en esta asignatura. La aplicación de dicha metodología en este proyecto me ha permitido contar con una versión entregable del producto al final de cada iteración realizada y a su vez, obtener constantemente retroalimentación para mejorar tanto el producto final como la aplicación de la metodología.
- **Desarrollo de Aplicaciones para Dispositivos Móviles (DADM):** Todos los conocimientos sobre desarrollo de software en Android puestos en práctica, así como descritos en apartados anteriores, fueron adquiridos durante el periodo en el que cursé dicha asignatura. Todos estos conocimientos engloban aquellos referentes a *Kotlin*, arquitectura de software para aplicaciones Android, uso de base de datos *Room* para almacenar datos localmente, creación de notificaciones, etc....
- **Calidad de Software (CSO):** Para realizar los análisis sobre la calidad del software desarrollado a lo largo de este proyecto, se han aplicado modelos de calidad de referencia para verificar el grado de cumplimiento de los requisitos previamente especificados. Dicha actividad es vital para asegurar la entrega de un producto software fiable, de calidad y que cumpla con su funcionalidad. Además, las heurísticas de Nielsen utilizadas para valorar la usabilidad de las interfaces de usuario fueron utilizadas durante el curso de esta asignatura.
- **Ingeniería del Software (ISW):** Técnicas utilizadas en el transcurso de este proyecto como la definición y especificación de casos de uso y la implantación de un proceso de software con fases enfocadas a construir software de calidad como especificación de requisitos, análisis, diseño o prueba han sido conocidas a través de dicha asignatura. Concretamente, se han utilizado más aquellos conocimientos relacionados con el análisis y abstracción del dominio del problema a resolver.
- **Análisis y Especificación de Requisitos (AER):** Al comienzo de este proyecto, se realizó una serie de tareas enfocadas a obtener requisitos potenciales para el tipo de aplicación que se pretendía desarrollar. Todas estas técnicas fueron conocidas y puestas en práctica a lo largo de esta asignatura. Estas técnicas, entre otras, fueron la realización de un modelo de dominio, la realización de diagramas de casos de uso y la priorización de los requisitos obtenidos.
- **Mantenimiento y Evolución de Software (MES):** A la hora de construir un software de calidad, no solamente se tiene en cuenta que todas las funcionalidades sean implementadas o que su desempeño en un ámbito de producción sea bueno. Es inconcebible crear software sin asumir que deberá pasar por un proceso de mantenimiento una vez entregado al usuario final, ya sea implementar nuevas funcionalidades, corregir defectos encontrados o mejorar atributos internos del programa. Es por ello por lo que se ha tratado de ofrecer cierto grado de mantenibilidad en este proyecto. Para ello, se han aplicado guías de estilo de código para que este sea más legible y también se ha documentado gran parte del código para que, en el caso de que otros desarrolladores lleguen a trabajar con el código de la aplicación, puedan entender con facilidad cómo funciona. Por otro

lado, la arquitectura *Clean* empleada también facilita la evolución y mantenimiento del sistema software, ya que permite la sustitución de componentes, su comprobación, la aplicación de pruebas, etc....

- **Diseño de Software (DDS):** Para que un software sea de calidad, es importante establecer una arquitectura en la que en cada componente estén bien definidas sus funciones y una serie de patrones de diseño para así facilitar la modificación de este en etapas futuras. Estos conocimientos también están muy relacionados con aquellos descritos en MES.
- **Deontología y Profesionalismo (DYP):** Todo aquello relacionado con la propiedad intelectual (véase el capítulo 3.6.2) como pueden ser las licencias de software o los derechos de autor, además de la protección de datos personales (véase el capítulo 3.6.1), han sido conceptos estudiados durante el transcurso de esta asignatura.
- **Integración e Interoperabilidad (IEI):** En la capa de persistencia, se realiza un proceso de *mapping* para transformar el objeto de datos obtenido en un objeto acorde al modelo de dominio de la aplicación. Todos esto se trata de una integración de datos, una operación cuyo objetivo fue aprendido a lo largo de la realización de esta asignatura.

8.2. Trabajos futuros

A pesar de haber cumplido los objetivos propuestos al inicio de este proyecto, no se han implementado todas las funcionalidades que se hubiesen deseado o de la forma más idónea. Todo esto fue debido a imprevistos durante el desarrollo de los distintos *Sprints* realizados que provocaron la demora en la implementación de ciertas funcionalidades y, por consiguiente, la postergación de diversas tareas. Es por ello por lo que se ha decidido plasmar los posibles futuros trabajos a realizar en este documento.

A continuación, se describen brevemente todos y cada uno de los posibles trabajos a implementar en la aplicación en versiones posteriores:

- Establecimiento de alarmas que se disparen una única vez.
- Establecimiento de alarmas con intervalos personalizados por el usuario.
- Escaneo del código de barras o código QR de un medicamento con el objetivo de agilizar el registro de algún medicamento en el sistema.
- Creación y gestión de perfiles familiares, pudiendo gestionar incluso las pautas de medicamentos de menores a cargo.
- Integración con otros dispositivos como *smartwatches* o dispositivos similares para llevar un registro de parámetros como por ejemplo la frecuencia cardíaca.
- Control y seguimiento de la tensión arterial por parte del usuario.
- Integración de datos de todas las farmacias a nivel nacional o a nivel regional a través de alguna fuente de datos publicada.
- Creación y gestión de recetas médicas, de tal modo que los médicos puedan recetar medicamentos a sus pacientes y estos puedan crear alarmas solamente para los medicamentos recetados.
- Gestión de stock disponible de un medicamento determinado, de tal modo que se reciba alguna notificación de recarga cuando se dispongan de pocas unidades.
- Funcionalidad de artículos científicos o periodísticos relacionados con la salud.
- Posibilidad de personalizar las notificaciones de alarma de pauta (modificación del mensaje de la notificación, cambiar colores, etc....).

Tal y como se ha podido comprobar, son numerosas y diversas las posibilidades que se abren para más adelante. En el caso de que se continuase con el proyecto o bien se realizase algún mantenimiento perfectivo, se tendrán en cuenta las implementaciones anteriores.

9. Referencias

- [1] Sociedad Española de Geriátría y Gerontología, «Farmacología y Envejecimiento. Los medicamentos en las personas mayores,» IMC, Madrid, 2016.
- [2] Instituto de Mayores y Servicios Sociales, «Los Mayores a un clic,» 17 Abril 2023. [En línea]. Available: <https://imserso.es/espacio-mayores/estadisticas/mayores-un-clic#:~:text=A%201%20de%20enero%20de,Fuente%3A%20Estad%3%ADstica%20del%20Padr%C3%B3n%20Continuo..> [Último acceso: 26 Mayo 2024].
- [3] MSDManuals, «Aging and Medications,» Agosto 2023. [En línea]. Available: <https://www.msdmanuals.com/home/older-people%E2%80%99s-health-issues/aging-and-medications/aging-and-medications>. [Último acceso: 2024 Mayo 26].
- [4] MedlinePlus, «Síndrome de Alport,» 19 Mayo 2023. [En línea]. Available: <https://medlineplus.gov/spanish/ency/article/000504.htm#:~:text=El%20s%C3%ADndrome%20de%20Alport%20es,la%20audi%C3%B3n%20y%20problemas%20oculares..> [Último acceso: 26 Mayo 2024].
- [5] Red Hat, «¿Qué es la metodología ágil?,» 19 Julio 2022. [En línea]. Available: <https://www.redhat.com/es/topics/devops/what-is-agile-methodology>. [Último acceso: 20 Febrero 2024].
- [6] C. Lasa Gómez, A. Álvarez García y R. de las Heras del Dedo, Métodos Ágiles. Scrum, Kanban, Lean, Madrid: Ediciones Anaya Multimedia, 2018.
- [7] K. Schwaber y J. Sutherland, «La guía de Scrum,» Noviembre 2020. [En línea]. Available: <http://www.scrum.org>. [Último acceso: 20 Febrero 2024].
- [8] IEEE Xplore, «830-1998 - IEEE Recommended Practice for Software Requirements Specifications,» 9 Diciembre 2009. [En línea]. Available: <https://ieeexplore.ieee.org/document/720574>. [Último acceso: 28 Mayo 2024].
- [9] Mobiliza Academy, «¿Qué es el método MoSCoW?,» 11 Diciembre 2022. [En línea]. Available: <https://www.linkedin.com/pulse/qu%C3%A9-es-el-m%C3%A9todo-moscow-mobiliza-academy/?originalSubdomain=es>. [Último acceso: 19 Febrero 2024].
- [10] DiagramasUML, «Diagrama de casos de uso,» [En línea]. Available: <https://diagramasuml.com/casos-de-uso/>. [Último acceso: 19 Febrero 2024].
- [11] C. R. Martin, Agile Principles, Patterns, and Practices in C#, Prentice Hall, 2006.
- [12] Android Developers, «Capa de la IU,» 22 Febrero 2024. [En línea]. Available: <https://developer.android.com/topic/architecture/ui-layer?hl=es-419>. [Último acceso: 26 Febrero 2024].
- [13] Android Developers, «Capa de dominio,» 22 Febrero 2024. [En línea]. Available: <https://developer.android.com/topic/architecture/domain-layer?hl=es-419>. [Último acceso: 26 Febrero 2024].
- [14] Android Developers, 22 Febrero 2024. [En línea]. Available: <https://developer.android.com/topic/architecture/data-layer?hl=es-419>. [Último acceso: 26 Febrero 2024].

- [15] Android Developers, «Introducción a Android Studio,» 30 Abril 2024. [En línea]. Available: <https://developer.android.com/studio/intro?hl=es-419>. [Último acceso: 13 Mayo 2024].
- [16] A. Guimerá Orozco, «Qué es Kotlin y características,» 5 Abril 2021. [En línea]. Available: <https://openwebinars.net/blog/que-es-kotlin/>. [Último acceso: 23 Marzo 2024].
- [17] Android Developers, «Enfoque de prioridad de Kotlin en Android,» 22 Febrero 2024. [En línea]. Available: <https://developer.android.com/kotlin/first?hl=es-419>. [Último acceso: 23 Marzo 2024].
- [18] Firebase, «Firebase Authentication - Introduction,» 20 Marzo 2024. [En línea]. Available: <https://firebase.google.com/docs/auth?hl=es>. [Último acceso: 15 Mayo 2024].
- [19] Firebase, «Cloud Firestore - Introducción,» 19 Enero 2024. [En línea]. Available: <https://firebase.google.com/docs/firestore?hl=es-419>. [Último acceso: 23 Marzo 2024].
- [20] IBM, «¿Qué son las bases de datos NoSQL?,» [En línea]. Available: <https://www.ibm.com/es-es/topics/nosql-databases>. [Último acceso: 23 Marzo 2024].
- [21] ML Kit, «ML Kit,» 8 Marzo 2024. [En línea]. Available: <https://developers.google.com/ml-kit/guides?hl=es-419>. [Último acceso: 7 Junio 2024].
- [22] J. M. Aguilar, «Qué es la inyección de dependencias y cómo funciona,» 25 Agosto 2020. [En línea]. Available: <https://www.campusmvp.es/recursos/post/que-es-la-inyeccion-de-dependencias-y-como-funciona.aspx>. [Último acceso: 1 Mayo 2024].
- [23] Android Developers, «Inserción de dependencias con Hilt,» 30 Abril 2024. [En línea]. Available: <https://developer.android.com/training/dependency-injection/hilt-android?hl=es-419>. [Último acceso: 1 Mayo 2024].
- [24] Android Developers, «Qué probar en Android,» 2 Diciembre 2023. [En línea]. Available: <https://developer.android.com/training/testing/fundamentals/what-to-test?hl=es-419>. [Último acceso: 4 Mayo 2024].
- [25] G. Prabhakar, «What is Espresso Testing? How does it works?,» 20 Octubre 2022. [En línea]. Available: <https://www.browserstack.com/guide/what-is-espresso-testing-how-does-it-work#:~:text=Espresso%20Testing%20Framework%20is%20an,complexity%20of%20managing%20different%20threads..> [Último acceso: 4 Mayo 2024].
- [26] Softtesting, «Por qué son necesarias las pruebas de Software?,» 26 Noviembre 2019. [En línea]. Available: https://softtesting.com/es_es/por-que-son-necesarias-las-pruebas-de-software/#:~:text=Las%20pruebas%20son%20necesarias%20porque,previenen%20los%20fallos%20%5B2%5D.. [Último acceso: 6 Mayo 2024].
- [27] Junta de Andalucía, «<https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/248#:~:text=JUnit%20se%20trata%20de%20un,asegurar%20su%20consistencia%20y%20funcionalidad..>» [En línea]. Available: <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/248#:~:text=JUnit%20se%20trata%20de%20un,asegurar%20su%20consistencia%20y%20funcionalidad..> [Último acceso: 6 Mayo 2024].
- [28] S. Chacon y B. Straub, Pro Git, Apress, 2021.
- [29] Android Developers, «Calidad básica de las apps,» 29 Febrero 2024. [En línea]. Available: <https://developer.android.com/docs/quality-guidelines/core-app-quality?hl=es-419>. [Último acceso: 14 Junio 2024].

- [30] UX247, «Principios de usabilidad: Las 10 heurísticas de usabilidad de Jakob Nielsen para el diseño de interfaces de usuario,» 20 Marzo 2024. [En línea]. Available: <https://ux247.com/es/usability-principles/>. [Último acceso: 2 Junio 2024].
- [31] Refactoring, «Refactoring,» [En línea]. Available: <https://refactoring.com/>. [Último acceso: 22 Junio 2024].
- [32] JetBrains, «¿Qué es el análisis de código estático?,» [En línea]. Available: <https://www.jetbrains.com/es-es/teamcity/ci-cd-guide/concepts/static-code-analysis/>. [Último acceso: 22 Junio 2024].
- [33] Amazon Web Services, «¿Qué son las pruebas unitarias?,» [En línea]. Available: <https://aws.amazon.com/es/what-is/unit-testing/#:~:text=Una%20prueba%20unitaria%20es%20un,la%20l%C3%B3gica%20te%C3%B3rica%20del%20desarrollador..> [Último acceso: 7 Mayo 2024].
- [34] Qalified, «Pruebas de Integración: qué son, tipos y ejemplos,» 29 Agosto 2023. [En línea]. Available: <https://qalified.com/es/blog/pruebas-de-integracion-que-son/>. [Último acceso: 30 Mayo 2024].
- [35] TestingIt, «Beneficios de las pruebas de usabilidad en apps móviles,» 23 Marzo 2022. [En línea]. Available: <https://www.testingit.com.mx/blog/pruebas-de-usabilidad>. [Último acceso: 30 Mayo 2024].
- [36] Abstracta Team, «¿Qué son las pruebas de regresión en Agile?,» 2 Enero 2024. [En línea]. Available: <https://es.abstracta.us/blog/pruebas-regresion-entorno-agile/>. [Último acceso: 16 Mayo 2024].
- [37] U. Nations, «Objetivos de Desarrollo Sostenible,» Naciones Unidas, [En línea]. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>. [Último acceso: 16 Febrero 2024].
- [38] N. Rodríguez, «Domain Modeling: Lo que necesitas saber antes de codificar,» Septiembre 2021. [En línea]. Available: <https://www.thoughtworks.com/es-es/insights/blog/agile-project-management/domain-modeling-what-you-need-to-know-before-coding>. [Último acceso: 16 Febrero 2024].
- [39] Android Developers, «Guía de arquitectura de apps,» 22 2 2024. [En línea]. Available: <https://developer.android.com/topic/architecture?hl=es-419>. [Último acceso: 26 Febrero 2024].
- [40] Visure Solutions, «Qué es la especificación de requisitos: definición, mejores herramientas y técnicas,» [En línea]. Available: <https://visuresolutions.com/es/blog/requirements-specification/#:~:text=Una%20especificaci%C3%B3n%20de%20requisitos%20es,trabajo%20futuro%20en%20el%20proyecto..> [Último acceso: 1 Mayo 2024].

10. Anexos

10.1. Reflexión sobre la relación del TFG con los Objetivos de Desarrollo Sostenible (ODS)



Figura 61.- Logotipo de la Agenda 2030

El 25 de septiembre de 2015, la Organización de Naciones Unidas (ONU) adoptó la Agenda 2030 para el Desarrollo Sostenible (cuyo logotipo se muestra en la Figura 61), un documento en el que se encuentran un total de 17 objetivos para todo el planeta con el fin de garantizar una vida sostenible, pacífica, próspera y justa en la tierra para todos, ahora y en el futuro [37]. Estos objetivos son conocidos como “Objetivos de Desarrollo Sostenible” (ODS). Cada objetivo tiene metas específicas que deben alcanzarse antes de 2030.

1. Fin de la pobreza
2. Hambre Cero
3. Salud y Bienestar
4. Educación de Calidad
5. Igualdad de género
6. Agua limpia y saneamiento
7. Energía asequible y no contaminante
8. Trabajo decente y crecimiento económico
9. Industria innovación e infraestructura
10. Reducción de las desigualdades
11. Ciudades y comunidades sostenibles
12. Producción y consumos responsables
13. Acción por el clima
14. Vida submarina
15. Vida de ecosistemas terrestres
16. Paz, justicia e instituciones
17. Alianzas para lograr objetivos.

En la Tabla 9 se indica para cada Objetivo de Desarrollo Sostenible el grado de relación con el proyecto.

Tabla 9.- Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS)

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.	X			

ODS 4.	Educación de calidad.				X
ODS 5.	Igualdad de género.				X
ODS 6.	Agua limpia y saneamiento.				X
ODS 7.	Energía asequible y no contaminante.				X
ODS 8.	Trabajo decente y crecimiento económico.				X
ODS 9.	Industria, innovación e infraestructuras.				X
ODS 10.	Reducción de las desigualdades.				X
ODS 11.	Ciudades y comunidades sostenibles.				X
ODS 12.	Producción y consumo responsables.				X
ODS 13.	Acción por el clima.				X
ODS 14.	Vida submarina.				X
ODS 15.	Vida de ecosistemas terrestres.				X
ODS 16.	Paz, justicia e instituciones sólidas.				X
ODS 17.	Alianzas para lograr objetivos.				X

De los anteriores objetivos de desarrollo sostenible mencionados, el proyecto está relacionado con:

- **Salud y bienestar**, es evidente que la ayuda que ofrece la aplicación para recordar a los usuarios la administración de un medicamento determinado, el momento exacto y la dosis adecuada conlleva una mejora por parte de los usuarios en el cumplimiento de las pautas indicadas por los profesionales sanitarios. Además, tal y como se ha explicado en la motivación (véase el capítulo 1.1), existe un gran porcentaje de personas de edad avanzada en España y muchas de ellas consumen algún tipo de medicamento. Por lo tanto, existe esa necesidad de ofrecer una herramienta que ayude a la hora de gestionar las prescripciones de los medicamentos correspondientes.

10.2.Glosario

- **API:** Conjunto de definiciones y protocolos usado con el objeto de diseñar e integrar cualquier producto software en otro distinto.
- **Checksum:** Valor numérico derivado de datos que se utiliza para verificar la integridad de esos datos. Es una forma de garantizar que los datos no hayan sido alterados durante su transmisión, almacenamiento o procesamiento.
- **Framework:** Herramienta de asistencia definida que puede servir de base para la organización y desarrollo de *software*.
- **Heurística:** Técnica de la indagación y del descubrimiento.
- **Kernel:** Programa el cual es parte de un sistema operativo, y que se ejecuta en modo privilegiado.
- **Script:** Lenguaje de programación empleado para manipular, personalizar y automatizar la instalación de un sistema existente.
- **SDK:** Conjunto de herramientas de desarrollo específicas de plataformas para programadores.

- **Stakeholder:** Cualquier individuo que, de alguna manera, es afectado por las acciones de una determinada actividad.
- **MVP:** Siglas de *Minimum Viable Product* (Mínimo Producto Viable en castellano). Se refiere al producto software que cumple con las características básicas para satisfacer a los usuarios potenciales.
- **Mapping:** Proceso mediante el cuál se trata de hacer coincidir los campos de una fuente de datos con los de otra.

10.3.Descripción detallada de los casos de uso

Tabla 10.- CU_01 - Inicio de sesión con usuario y contraseña

Identificador de caso de uso:	CU_01	
Nombre:	Iniciar sesión con usuario y contraseña	
Descripción:	El usuario no autenticado introduce en la vista de inicio de sesión sus credenciales (nombre de usuario y contraseña). Si son válidas, accede al menú principal, y si no es el caso, se le negará la entrada al sistema.	
Actores:	Usuario no autenticado	
Flujo Normal:		
Actor	Sistema	
1. El usuario introduce su nombre de usuario en la casilla correspondiente.		
2. El usuario introduce su contraseña en la casilla correspondiente.		
3. El usuario pulsa en el botón "Iniciar sesión".		
	4. El sistema comprueba que dichas credenciales existen en la base de datos.	
5. El usuario accede al menú principal de la aplicación.		
Excepciones		
Excepciones		Sistema
4. Las credenciales no existen.	a.- Muestra un mensaje de error indicando que las credenciales son incorrectas. b.- Regresar al paso 1.	
3. No se introduce nombre de usuario.	a.- Muestra un mensaje de error indicando que hay campos sin cumplimentar. b.- Regresar al paso 1.	
3. No se introduce contraseña.	a.- Muestra un mensaje de error indicando que hay campos sin cumplimentar. b.- Regresar al paso 1.	
Postcondiciones:	El usuario accede al menú principal de la aplicación.	

Tabla 11.- CU_02 - Cierre de sesión

Identificador de caso de uso:	CU_02	
Nombre:	Cierre de sesión	
Descripción:	El usuario, ya sea paciente o médico, puede terminar la sesión en cualquier momento, volviendo así a la interfaz de inicio de sesión.	
Actores:	Paciente Médico	
Flujo Normal:		
Actor	Sistema	
1. El usuario selecciona "Cerrar sesión".		
	2. El sistema olvida las credenciales del usuario activo.	
3. El usuario vuelve a la interfaz de inicio de sesión.		
Excepciones	Sistema	
Postcondiciones:	El usuario vuelve a la interfaz de inicio de sesión y el sistema olvida las credenciales del usuario anterior.	

Tabla 12.- CU_03 - Listar alarmas establecidas

Identificador de caso de uso:	CU_03	
Nombre:	Listar alarmas establecidas	
Descripción:	Se muestra en pantalla todas las alarmas establecidas hasta el momento por parte del paciente.	
Actores:	Paciente	
Precondiciones:	El usuario debe haber iniciado sesión previamente y estar autenticado dentro del sistema.	
Flujo Normal:		
Actor	Sistema	
	1. El sistema muestra las alarmas establecidas por el usuario.	
Excepciones	Sistema	
Postcondiciones:	El usuario debe tener la opción de eliminar alguna de las alarmas establecidas.	

Tabla 13.- CU_04 - Registrarse

Identificador de caso de uso:	CU_04	
Nombre:	Registrarse	
Descripción:	Si el usuario no dispone de una cuenta con la que autenticarse dentro de la aplicación, deberá crearse una aportando los datos requeridos y sus credenciales.	
Actores:	Usuario no autenticado	
Precondiciones:	<ul style="list-style-type: none"> • El usuario no debe estar autenticado en el sistema. 	
Flujo Normal:		
Actor	Sistema	

1. El usuario pulsa en el botón “¿No tienes cuenta? Regístrate”.	
2. El usuario cumplimenta los campos con su nombre, apellidos, dirección de email y contraseña.	
3. El usuario indica si se trata de un médico o no.	
4. El usuario pulsa en el botón “Registrarse”.	
	5. El sistema comprueba que los datos aportados son válidos.
6. En el caso de que no sea médico, se le enviará a la interfaz de selección de médico, en la que se muestran los distintos médicos existentes en el sistema y en la que el usuario seleccionará un médico.	
	7. El sistema almacena el email del médico en los datos del usuario.
	8. El sistema registra las credenciales tanto en <i>Firestore</i> como en <i>Firebase Authentication</i> .
9. En caso contrario, el usuario obtiene acceso al menú principal de la aplicación.	
Excepciones	Sistema
5. No se han completado todos los campos.	a.- El sistema muestra un mensaje de error indicando que hay campos sin cumplimentar. b.- Se vuelve al paso 2.
5. Formato de dirección de email inválido.	a.- El sistema muestra un mensaje de error informando del formato incorrecto de la dirección de email. b.- Se vuelve al paso 2.
5. Contraseña de longitud inferior a 8 caracteres.	a.- El sistema muestra un mensaje de error, indicando que debe ser igual o superior a 8 caracteres. b.- Se vuelve al paso 2.
5. Usuario con dirección de email ya existente en el sistema.	a.- El sistema muestra un mensaje informando sobre la existencia de dicho usuario. b.- Se vuelve al paso 2.
Postcondiciones:	Las credenciales y todos los datos aportados por el usuario deben quedar registrados en <i>Firestore</i> , y el usuario debe haber podido acceder al menú principal de la aplicación.

Tabla 14.- CU_05 - Visualizar consejo de salud

Identificador de caso de uso:	CU_05
Nombre:	Visualizar consejo de salud
Descripción:	Cada vez que el usuario acceda al menú principal de la aplicación, se le mostrará mediante un <i>popup</i> un consejo para implantar un estilo de vida más saludable.

Actores:	Paciente, Médico
Precondiciones:	El usuario debe estar autenticado en el sistema
Flujo Normal:	
Actor	Sistema
	1. El sistema solicita a la base de datos la información necesaria de un consejo de salud aleatorio.
	2. El sistema carga dichos datos en los elementos que conforman el <i>popup</i> .
	3. El sistema muestra el consejo de salud al usuario.
Excepciones	Sistema
Postcondiciones:	Se debe haber mostrado el consejo de salud al usuario

Tabla 15.- CU_06 - Borrar alarma

Identificador de caso de uso:	CU_06
Nombre:	Borrar alarma
Descripción:	Si el usuario no desea recibir más notificaciones de alguna de las alarmas establecidas previamente, deberá poder cancelarlas y borrarlas de la base de datos.
Actores:	Paciente, Médico
Precondiciones:	<ul style="list-style-type: none"> • El usuario debe estar previamente autenticado en el sistema. • Deben haberse creado alarmas previamente.
Flujo Normal:	
Actor	Sistema
1. El usuario desliza hacia la derecha la alarma que desea eliminar.	
	2. El sistema elimina dicha alarma de la base de datos.
	3. El sistema cancela la alarma de modo que ya no se envíen notificaciones relacionadas con dicha alarma.
Excepciones	Sistema
Postcondiciones:	La alarma dejará de aparecer en la lista de alarmas existentes y se dejará de recibir notificaciones referentes a dicha alarma.

Tabla 16.- CU_07 - Crear alerta de pauta

Identificador de caso de uso:	CU_07
Nombre:	Crear alerta de pauta
Descripción:	Proceso que los usuarios han de seguir para establecer una alarma relacionada con la pauta de algún medicamento. Durante este proceso, se selecciona el medicamento, la presentación de este, la dosis por cada alarma y la frecuencia y hora de inicio de la alarma.
Actores:	Paciente
Precondiciones:	<ul style="list-style-type: none"> • El usuario debe estar previamente autenticado en el sistema.
Flujo Normal:	
Actor	Sistema

1. El usuario selecciona en el botón “+” para crear una alarma.	
	2. El sistema muestra una lista con los medicamentos existentes en la base de datos.
3. El usuario selecciona alguno de los medicamentos listados.	
	4. El sistema registra el medicamento en la alarma a crear.
	5. El sistema muestra una lista con las posibles presentaciones de medicamentos almacenadas en la base de datos.
6. El usuario selecciona una presentación de medicamento.	
	7. El sistema muestra las cantidades posibles de la dosis según la presentación seleccionada previamente.
8. El usuario selecciona una cantidad de entre las mostradas.	
	9. El sistema registra la cantidad seleccionada en la alarma.
	10. El sistema muestra los selectores de la hora de comienzo de la alarma y de la frecuencia de repetición.
11. El usuario selecciona una hora de comienzo de la alarma y una frecuencia de repetición	
	12. El sistema registra la hora de comienzo y la frecuencia de la alarma
	13. El sistema crea la alarma para su ejecución dentro del rango establecido
Excepciones	Sistema
Postcondiciones:	La alarma creada se debe disparar de acuerdo con el rango de tiempo establecido.

Tabla 17.- CU_08 - Asignar medicamento

Identificador de caso de uso:	CU_08
Nombre:	Asignar medicamento
Descripción:	Proceso para asignar un medicamento a una pauta determinada.
Actores:	Paciente
Precondiciones:	<ul style="list-style-type: none"> El paciente debe estar autenticado dentro de la aplicación.
Flujo Normal:	
Actor	Sistema
1. El usuario selecciona un medicamento determinado.	
	2. El sistema lo asigna a la alarma a crear.
Excepciones	Sistema
Postcondiciones:	El nombre del medicamento debe quedar registrado dentro de los datos que componen la alarma a crear.

Tabla 18.- CU_09 - Asignar dosis

Identificador de caso de uso:	CU_09
Nombre:	Asignar dosis
Descripción:	Proceso para asignar una dosis a una pauta determinada.
Actores:	Paciente
Precondiciones:	<ul style="list-style-type: none"> El paciente debe estar autenticado dentro de la aplicación.
Flujo Normal:	
Actor	Sistema
1. El usuario selecciona una dosis determinada para la alarma en proceso de creación.	
	2. El sistema registra la cantidad total de la dosis en los datos de la alarma.
Excepciones	Sistema
Postcondiciones:	La cantidad que se debe tomar en cada dosis debe quedar registrada en la alarma a crear.

Tabla 19.- CU_10 - Asignar periodo de pauta

Identificador de caso de uso:	CU_10
Nombre:	Asignar periodo de pauta
Descripción:	Proceso para asignar el periodo en el que se deben disparar las alarmas en el dispositivo móvil del usuario.
Actores:	Paciente
Precondiciones:	<ul style="list-style-type: none"> El paciente debe estar autenticado dentro de la aplicación.
Flujo Normal:	
Actor	Sistema
1. El usuario selecciona un periodo para la alarma.	
	2. El sistema registra el periodo en la alarma en los datos de esta.
Excepciones	Sistema
Postcondiciones:	El periodo de repetición de la alarma queda establecido en sus datos.

Tabla 20.- CU_11 - Reservar cita

Identificador de caso de uso:	CU_11
Nombre:	Reservar cita
Descripción:	Proceso por el cual el usuario reserva una cita con su médico.
Actores:	Paciente
Precondiciones:	<ul style="list-style-type: none"> El paciente debe estar previamente autenticado en la aplicación.
Flujo Normal:	
Actor	Sistema
1. El paciente selecciona una fecha para la cita.	

	2. El sistema registra la fecha en los datos de la cita.
3. El paciente selecciona una hora para la cita.	
	4. El sistema registra la hora en los datos de la cita.
	5. El sistema añade la cita a la base de datos.
Excepciones	Sistema
Postcondiciones:	La cita médica queda registrada en la base de datos de la aplicación y se le muestra al usuario los datos de esta junto con algún dato de contacto de su médico.

Tabla 21.- CU_12 - Comprobar farmacias cercanas

Identificador de caso de uso:	CU_12
Nombre:	Comprobar farmacias cercanas
Descripción:	Funcionalidad en la que se muestran todas las farmacias más cercanas según la localización del usuario.
Actores:	Paciente, Médico
Precondiciones:	<ul style="list-style-type: none"> El usuario debe estar previamente autenticado dentro del sistema.
Flujo Normal:	
Actor	Sistema
1. El usuario otorga permisos para acceder a su localización.	
	2. El sistema recolecta de la base de datos todas aquellas farmacias situadas dentro de un radio de 60 km. Desde la posición del usuario.
	3. El sistema crea en el mapa un marcador por cada farmacia recolectada.
Excepciones	Sistema
Postcondiciones:	El sistema muestra las farmacias que se encuentran dentro del radio establecido según la posición del usuario. Cada marcador corresponde con una farmacia y al pulsar en uno, se muestra la dirección de la farmacia y su nombre.

Tabla 22.- CU_13 - Registrar nuevo medicamento

Identificador de caso de uso:	CU_13	
Nombre:	Registrar nuevo medicamento	
Descripción:	Proceso por el cual un médico registra un nuevo medicamento en el sistema.	
Actores:	Médico	
Precondiciones:	<ul style="list-style-type: none"> El usuario debe estar previamente autenticado en el sistema. 	
Flujo Normal:		
Actor	Sistema	
1. El usuario asigna un nombre para identificar al medicamento.		
2. El usuario asigna una descripción al medicamento.		
3. El usuario selecciona una serie de efectos secundarios que podrían ser producidos por la administración del medicamento en proceso de creación.		
4. El usuario pulsa en el botón "Crear medicamento".		
	5. El sistema comprueba que todos los datos introducidos son válidos.	
	6. El sistema registra el nuevo medicamento en la base de datos del sistema.	
Excepciones		
Actor	Sistema	
5. Campos sin rellenar	a.- El sistema muestra un mensaje de error. b.- Se vuelve al paso 1.	
Postcondiciones:	El medicamento ha sido añadido a la base de datos del sistema software satisfactoriamente.	

Tabla 23.- CU_14 - Seleccionar idioma de uso

Identificador de caso de uso:	CU_14	
Nombre:	Seleccionar idioma de uso	
Descripción:	En función del idioma establecido en el dispositivo del usuario, la aplicación se mostrará en este idioma. Los idiomas soportados serán castellano, catalán e inglés.	
Actores:	Usuario no autenticado, Paciente y Médico	
Precondiciones:		
Flujo Normal:		
Actor	Sistema	
1. El usuario establece un idioma determinado en su dispositivo.		
	2. El sistema adapta los textos para mostrarlos en el idioma establecido.	

Excepciones		Sistema
1. El idioma establecido no es ninguno de los soportados		a.- Por defecto, se mostrará la aplicación en castellano.
Postcondiciones:	La aplicación deberá mostrarse en el idioma establecido por el usuario.	

Tabla 24.- CU_15 - Seleccionar médico

Identificador de caso de uso:	CU_15	
Nombre:	Seleccionar médico	
Descripción:	En caso de que el usuario a registrar no sea médico, este seleccionará un médico de entre los existentes en el sistema.	
Actores:	Usuario no autenticado	
Precondiciones:	<ul style="list-style-type: none"> El usuario debe haber indicado previamente en el registro que no es un médico 	
Flujo Normal:		
Actor		Sistema
		1. El sistema lista todos los médicos existentes en el sistema.
2. El usuario selecciona uno de los médicos listados.		
		3. El sistema registra el médico seleccionado en los datos del usuario.
Excepciones		Sistema
Postcondiciones:	<ul style="list-style-type: none"> Los datos del usuario deben haber sido actualizados con el identificador del médico seleccionado. 	

Tabla 25.- CU_16 - Consultar citas

Identificador de caso de uso:	CU_16	
Nombre:	Consultar citas	
Descripción:	Los médicos deben poder consultar las próximas citas de sus pacientes.	
Actores:	Médico, Paciente	
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe tener rol de médico. Deben existir citas de usuarios con el médico autenticado. 	
Flujo Normal:		
Actor		Sistema
1. El médico accede a la sección de consulta de citas.		
		2. El sistema muestra una lista todas las futuras citas de los pacientes del médico en cuestión.
Excepciones		Sistema

Postcondiciones:	<ul style="list-style-type: none"> Una lista de citas médicas pendientes del médico autenticado
-------------------------	--

Tabla 26.- CU_17 - Seleccionar fecha y hora

Identificador de caso de uso:	CU_17	
Nombre:	Seleccionar fecha y hora	
Descripción:	A la hora de reservar una cita médica, el usuario debe seleccionar la fecha y la hora.	
Actores:	Paciente	
Precondiciones:	<ul style="list-style-type: none"> El usuario debe tener rol de paciente 	
Flujo Normal:		
Actor	Sistema	
1. El usuario selecciona una fecha concreta para la cita médica, de entre las posibles.		
	2. El sistema registra dicha fecha en los datos de la cita.	
3. El usuario selecciona una hora concreta para la cita médica.		
	4. El sistema registra la hora seleccionada en los datos de la cita	
	5. El sistema registra la cita médica nueva en la base de datos.	
Excepciones	Sistema	
5. Cita médica con mismo día y misma fecha ya existente	a.- Se notifica al usuario de que la hora seleccionada está ocupada, y se le solicita seleccionar otra fecha u hora.	
Postcondiciones:	<ul style="list-style-type: none"> La cita médica debe haber sido registrada en la base de datos del sistema. 	

Tabla 27.- CU_18 - Cancelar cita

Identificador de caso de uso:	CU_18	
Nombre:	Cancelar cita	
Descripción:	Los pacientes deben tener la opción en todo momento de cancelar una cita médica previamente creada.	
Actores:	Paciente	
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser un paciente. Debe existir alguna cita médica creada por el usuario. 	
Flujo Normal:		
Actor	Sistema	
1. El usuario desliza hacia la derecha de la pantalla la cita a eliminar.		
	2. El sistema borra dicha cita de la base de datos.	
Excepciones	Sistema	
Postcondiciones:	<ul style="list-style-type: none"> La cita médica ha sido eliminada de la base de datos. 	

Tabla 28.- CU_19 - Indicar efectos secundarios

Identificador de caso de uso:	CU_19	
Nombre:	Indicar efectos secundarios	
Descripción:	A la hora de registrar un nuevo medicamento, se deben indicar, si es el caso, sus posibles efectos adversos.	
Actores:	Médico	
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser médico. 	
Flujo Normal:		
	Actor	Sistema
	1. El usuario rellena todos los campos de registro del medicamento.	
	2. El usuario selecciona, si es el caso, los efectos adversos provocados por la administración de este.	
		3. El sistema registra tanto los datos del nuevo medicamento como los datos de los efectos adversos en la base de datos.
	Excepciones	Sistema
Postcondiciones:	<ul style="list-style-type: none"> Los efectos adversos se han añadido satisfactoriamente al medicamento 	

Tabla 29.- CU_20 - Establecer nombre del medicamento

Identificador de caso de uso:	CU_20	
Nombre:	Establecer nombre del medicamento	
Descripción:	Cuando se está registrando un nuevo medicamento en el sistema, se debe asignar un nombre a dicho medicamento.	
Actores:	Médico	
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser médico 	
Flujo Normal:		
	Actor	Sistema
	1. El usuario introduce el nombre del medicamento.	
	2. El usuario introduce el resto de los datos para la creación del medicamento.	
		3. El sistema añade el medicamento nuevo a la base de datos.
	Excepciones	Sistema
	3. Medicamento con nombre proporcionado ya existente en el sistema.	a.- Se muestra un mensaje indicando que ya existe un medicamento con dicho nombre.
Postcondiciones:	<ul style="list-style-type: none"> El nuevo medicamento ha sido registrado en la base de datos. 	

Tabla 30.- CU_21 - Cancelar cita

Identificador de caso de uso:	CU_21	
Nombre:	Cancelar cita	
Descripción:	Si un paciente desea cancelar una cita, debe poder hacerlo de forma muy sencilla	
Actores:	Paciente	
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser un paciente 	
Flujo Normal:		
Actor	Sistema	
1. El paciente desliza hacia la derecha de la pantalla la cita que desea cancelar.		
	2. El sistema borra dicha cita de la base de datos.	
Excepciones		
Sistema		
Postcondiciones:	<ul style="list-style-type: none"> La cita médica eliminada de la base de datos satisfactoriamente. 	

Tabla 31.- CU_22 - Registrar síntoma

Identificador de caso de uso:	CU_22	
Nombre:	Registrar síntoma	
Descripción:	El usuario debe tener la posibilidad de registrar los síntomas que siente en caso de que se esté medicando, y así su médico asociado pueda hacer un seguimiento.	
Actores:	Paciente	
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser un paciente. 	
Flujo Normal:		
Actor	Sistema	
1. El usuario accede al apartado de registro de síntomas.		
2. El usuario describe a través de un campo de texto los síntomas que está padeciendo.		
	3. El sistema comprueba que todos los datos introducidos son válidos.	
	4. Los síntomas son almacenados en la base de datos, para estar a disposición del médico asignado.	
Excepciones		
Sistema		
3. Algún campo está vacío	a.- Se muestra un mensaje de error indicando que se deben rellenar los datos requeridos.	
Postcondiciones:	<ul style="list-style-type: none"> El síntoma ha sido almacenado en la base de datos satisfactoriamente. 	

Tabla 32.-CU_23 - Visualizar artículo sobre salud

Identificador de caso de uso:	CU_23	
Nombre:	Visualizar artículo sobre salud	
Descripción:	El usuario tendrá la posibilidad de consultar noticias o artículos que guarden relación con el sector de la salud.	
Actores:	Paciente, Médico	
Precondiciones:	<ul style="list-style-type: none"> El usuario debe estar autenticado previamente. 	
Flujo Normal:		
Actor	Sistema	
1. El usuario accede al apartado de artículos.		
	2. El sistema carga todos los artículos existentes.	
Excepciones		
Sistema		
Postcondiciones:	<ul style="list-style-type: none"> El usuario tiene todos los artículos existentes a su disposición. 	

Tabla 33.- CU_24 - Indicar prescripción médica necesaria

Identificador de caso de uso:	CU_24	
Nombre:	Indicar prescripción médica necesaria	
Descripción:	A la hora de crear un nuevo medicamento, el médico debe poder indicar si el medicamento necesita de prescripción médica.	
Actores:	Médico	
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser médico 	
Flujo Normal:		
Actor	Sistema	
1. El médico rellenará los datos que se requieren para el nuevo medicamento.		
2. El usuario indicará o no en el mismo formulario si el medicamento nuevo está sujeto a prescripción médica.		
	3. El sistema comprueba que todos los datos introducidos son válidos.	
	4. El sistema almacena los datos del nuevo medicamento en la base de datos.	
Excepciones		
Sistema		
3. Algún dato no se ha introducido.	a.- Se le muestra al usuario un mensaje de error indicando que se deben rellenar todos los campos requeridos.	
Postcondiciones:	<ul style="list-style-type: none"> El medicamento se ha registrado satisfactoriamente, indicando si necesita prescripción médica o no. 	

Tabla 34.- CU_25 - Escanear código de barras

Identificador de caso de uso:	CU_25	
Nombre:	Escanear código de barras	
Descripción:	Con el objeto de agilizar el registro de un nuevo medicamento en el sistema, el médico en cuestión podrá escanear con su dispositivo móvil el código de barras o código QR de un medicamento para así obtener todos sus datos rápidamente.	
Actores:	Médico	
Precondiciones:	<ul style="list-style-type: none"> • El usuario autenticado debe ser médico • El medicamento en cuestión no debe existir previamente en el sistema. 	
Flujo Normal:		
Actor	Sistema	
1. El usuario otorga permisos a la aplicación para hacer uso de la cámara.		
2. El usuario enfoca el código a escanear.		
	3. El sistema obtiene todos los datos asociados al medicamento.	
	4. Los datos de interés se cargan en el formulario de creación de medicamentos.	
Excepciones	Sistema	
1. El usuario no otorga permisos a la aplicación para usar la cámara.	a.- No se permitirá hacer uso de la funcionalidad al usuario.	
Postcondiciones:	<ul style="list-style-type: none"> • Los datos del medicamento se han cargado automáticamente en el formulario gracias al escaneo de su código de barras o QR. 	

Tabla 35.- CU_26 - Crear receta

Identificador de caso de uso:	CU_26	
Nombre:	Crear receta	
Descripción:	Cualquier usuario con rol de médico podrá asignar recetas médicas a cualquier paciente a su cargo.	
Actores:	Médico	
Precondiciones:	<ul style="list-style-type: none"> • El usuario autenticado debe ser un médico. • El médico autenticado debe tener pacientes a su cargo. 	
Flujo Normal:		
Actor	Sistema	
1. El usuario selecciona el usuario al que asignar la receta.		
2. El usuario selecciona el medicamento a asignar al usuario seleccionado.		
3. El usuario indica la dosis que el paciente debe tomar.		
	4. El sistema comprueba que todos los datos introducidos son válidos.	
	5. El sistema registra la nueva receta en la base de datos.	
Excepciones	Sistema	

Postcondiciones:	<ul style="list-style-type: none"> La receta ha sido añadida a la base de datos satisfactoriamente.
-------------------------	--

Tabla 36.- CU_27 - Asignar medicamento a paciente

Identificador de caso de uso:	CU_27
Nombre:	Asignar medicamento a paciente
Descripción:	Cuando se está creando una receta médica, el médico debe poder asignar un medicamento a esta.
Actores:	Médico
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser un médico. El médico autenticado debe tener pacientes a su cargo.
Flujo Normal:	
Actor	Sistema
1. El médico selecciona un paciente al que asignar el medicamento.	
2. El médico selecciona un medicamento a asignar.	
3. El médico rellena el resto de los datos necesarios.	
	4. El sistema comprueba que todos los datos introducidos son válidos.
	5. El sistema añade los datos a la base de datos.
Excepciones	
Sistema	
Postcondiciones:	<ul style="list-style-type: none"> La receta ha sido añadida a la base de datos satisfactoriamente, con el paciente destino y el medicamento asignado.

Tabla 37.- CU_28 - Comprobar disponibilidad

Identificador de caso de uso:	CU_28
Nombre:	Comprobar disponibilidad
Descripción:	Cuando un paciente se disponga a reservar una cita con su médico, debe poder comprobar las fechas y horas disponibles.
Actores:	Paciente
Precondiciones:	<ul style="list-style-type: none"> El usuario autenticado debe ser un paciente.
Flujo Normal:	
Actor	Sistema
1. El usuario pulsa en el botón para reservar citas médicas.	
	2. El sistema comprobará las citas existentes en el sistema.
	3. El sistema ofrecerá las fechas y horas ya ocupadas por otros usuarios.
	4. En función de estas fechas, el sistema las mostrará como no seleccionables.
	5. Para las horas, se le mostrará al usuario un listado con las horas disponibles.
Excepciones	
Sistema	

Postcondiciones:	<ul style="list-style-type: none"> El usuario ha reservado una cita médica habiendo podido comprobar qué fechas y horas estaban disponibles en ese momento.

10.4. Interfaces de usuario de la aplicación

A continuación, en este capítulo, se muestra el resultado final de las interfaces de usuario que componen la aplicación desarrollada. Son las siguientes:

- Listado de alarmas establecidas (Figura 62 – a))
- Listado de citas pendientes (Figura 62 – b))
 - Selección de fecha (Figura 64 – c))
 - Selección de hora (Figura 64 – d))
- Localización de farmacias cercanas (Figura 62 – c))
- Creación de alarmas
 - Selección de medicamento (Figura 63 – a))
 - Selección de presentación de medicamento (Figura 63 – b))
 - Selección de dosis (Figura 63 – c))
 - Selección de frecuencia y hora de inicio (Figura 63 – d))
- Consejo de salud (Figura 64 – a))
- Detalle de alarma (Figura 64 – b))
- Inicio de sesión (Figura 65 – a))
- Formulario de registro (Figura 65 – b))
- Selección de médico (Figura 65 – c))
- Formulario de registro de nuevos medicamentos (Figura 65 – d))



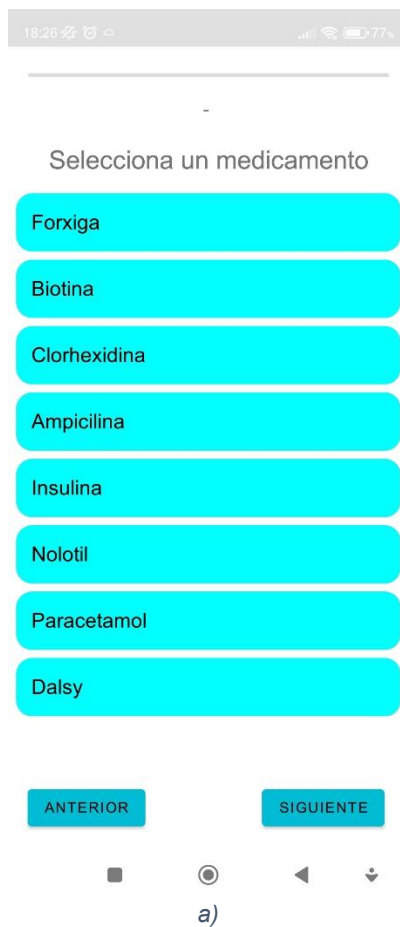
a)



b)



Figura 62.- Interfaces de usuario de la aplicación: a) listado de alarmas establecidas, b) Listado de citas médicas, c) Localización de farmacias cercanas



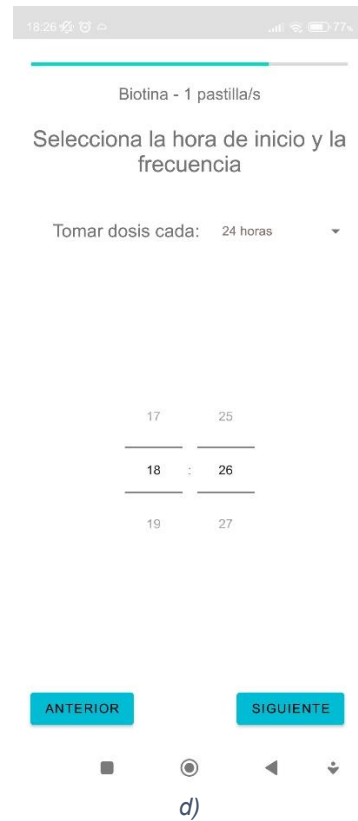
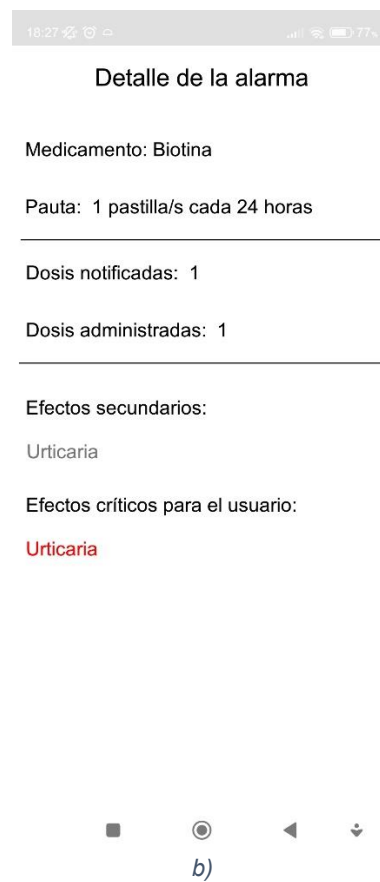
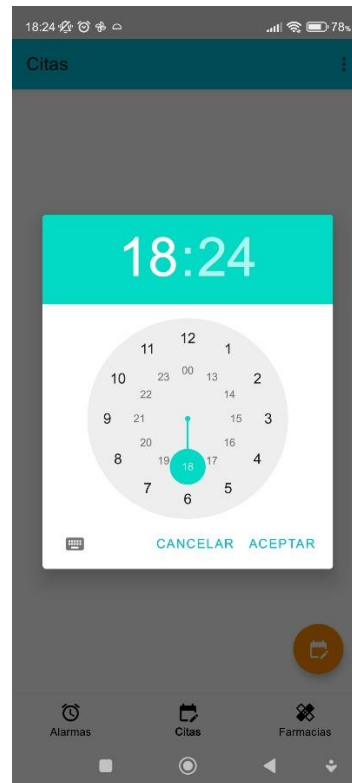


Figura 63.- Interfaces de usuario de la aplicación: a) Selección de medicamento, b) Selección de la presentación del medicamento, c) Selección de la dosis, d) Selección de la hora de inicio y la frecuencia



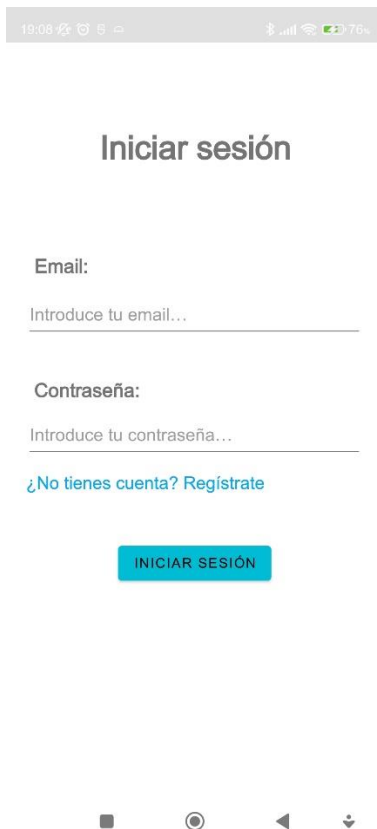


c)

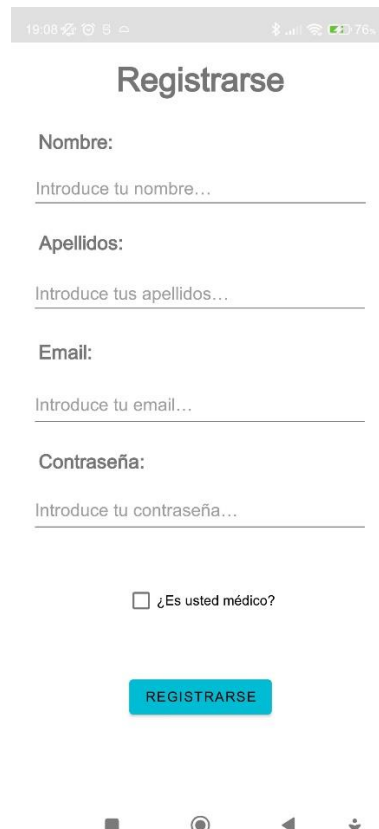


d)

Figura 64.- Interfaces de usuario de la aplicación a) consejo de salud, b) detalle de una alarma, c) selección de fecha de cita médica, d) selección de hora de cita médica



a)



b)

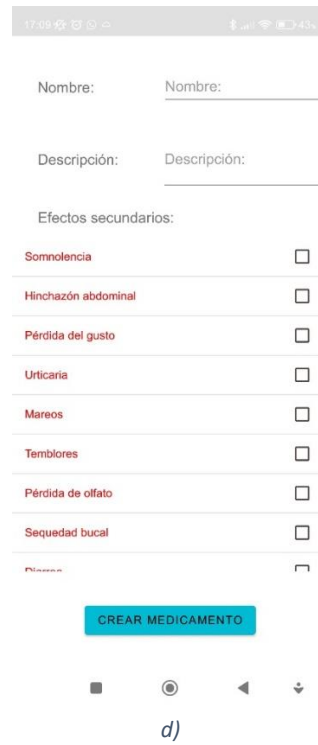
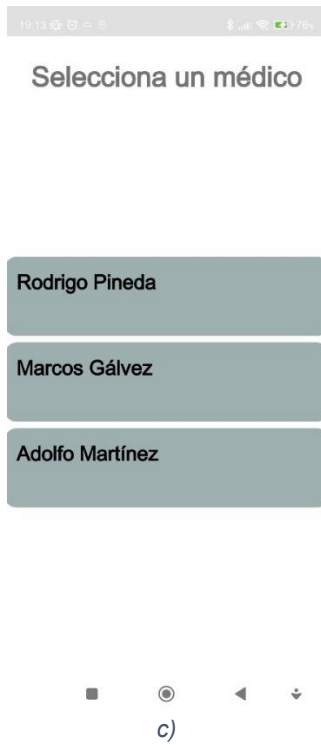


Figura 65.- Interfaces de usuario de la aplicación: a) inicio de sesión, b) formulario de registro, c) selección de médico, d) formulario de registro de nuevos medicamentos