APLICACIÓN PARA FACILITAR EL USO DE UN TELÉFONO ANDROID EN PERSONAS MAYORES

APLICATION TO FACILITATE THE USE OF AN ANDROID PHONE IN THE ELDERLY

TRABAJO FIN DE GRADO
CURSO 2021-2022

AUTOR
JAVIER GÓMEZ MORALEDA

DIRECTOR
ANTONIO SARASA CABEZUELO

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DEDICATORIA

A todos nuestros mayores y en especial a mis abuelos por su amor incondicional.

AGRADECIMIENTOS

A mi familia por la confianza depositada en mi en cada cosa que hago. A mi pareja por su capacidad de alegrarme hasta los días más grises. Y a todos mis amigos por recordarme que lo más importante es rodearte de personas buenas.

RESUMEN

En este trabajo se describe la especificación, el diseño y la implementación de una aplicación para facilitar el uso de un móvil a las personas mayores. Aunque la aplicación está destinada a las personas mayores, puesto que suelen ser las que más complicaciones tienen con el uso de las nuevas tecnologías, puede servir también para aquellas personas con pocos conocimientos o dificultades debido a alguna discapacidad.

La aplicación proporciona al usuario una interfaz sencilla formada por diferentes módulos funcionales. El usuario tendrá la posibilidad de realizar llamadas telefónicas, tanto a través de un marcador, como mediante sus contactos favoritos, incluyendo una agenda de contactos que podrá ser modificada. También dispondrá de la posibilidad de crear una pantalla de inicio donde podrá agregar las aplicaciones que más utilice o visualizar una lista de estas. Por último, tendrá acceso a un recordatorio de medicamentos, donde visualizará los medicamentos que tiene que tomar cada día, así como una lista completa. Así mismo, se podrán añadir, editar y eliminar nuevos medicamentos de forma sencilla.

Palabras clave

Android, personas mayores, accesibilidad, brecha digital, medicamentos.

ABSTRACT

This work describes the specification, design and implementation of an application to facilitate the use of a mobile phone by the elderly. Although the application is aimed at the elderly, since they are usually the ones who have more complications with the use of new technologies, it can also be used by those with little knowledge or difficulties due to some disability.

The application provides the user with a simple interface consisting of different functional modules. The user will have the possibility to make phone calls, both through a dialer and through his or her favourite contacts, including a contact list that can be modified. They will also have the possibility of creating a home screen where they can add the applications they use most or display a list of these. Finally, the user will have access to a medication reminder, where he or she will see the medications to take each day, as well as a complete list. It is also easy to add, edit and delete new medicines.

Keywords

Android, elderly people, accessibility, digital divide, medicines.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

En este capítulo se explicará cual es la motivación que ha llevado a realizar este trabajo, los objetivos que se pretenden alcanzar y como está estructurada la memoria.

1.1 Motivación

Las personas mayores son las grandes olvidadas de nuestra generación. Mientras la tecnología avanza y prácticamente todo se realiza a través de una pantalla, muchas personas, en especial, nuestros mayores, son los que más sufren este avance, esto se conoce como brecha digital.

Generalmente, tanto nuestra generación, como la de nuestros padres, nos desenvolvemos bien haciendo uso de las nuevas tecnologías. Si miramos al pasado, vemos que la mayoría de las actividades como, por ejemplo, realizar una compra o un trámite bancario, antes requerían salir de casa. Sin embargo, todo eso ha cambiado, y es que gracias a los avances tecnológicos se puede hacer prácticamente todo a través de una pantalla, haciéndonos ahorrar tiempo y dinero. Es por eso por lo que cada día se ve como todos los negocios se van adaptando, ya sea un supermercado realizando envíos a domicilio o un banco cerrando oficinas en favor del uso de su aplicación.

Sin embargo, si a cualquier persona mayor con dificultades utilizando un teléfono inteligente que, posiblemente haya sido comprado por un familiar, intentamos enseñarle a utilizar estas aplicaciones complejas, van a terminar frustrados.

Para resolver esta brecha digital, en este trabajo se han propuesto el desarrollo de una aplicación accesible para cualquier tipo de usuario, en especial, para las personas mayores. Este trabajo no pretende solucionar este gran problema, sino aportar un granito de arena para que aquellas personas con miedo a utilizar un teléfono inteligente puedan hacer un uso básico de él.

Mediante la simplificación de aplicaciones que ya existen actualmente, será posible que, de forma autónoma, los usuarios puedan manejar aplicaciones tan básicas como una agenda de contactos, organizar una pantalla principal con sus aplicaciones

favoritas y, aprovechar el potencial de las aplicaciones mediante un recordatorio de medicamentos.

1.2 Objetivos

El objetivo principal del trabajo es la implementación de una aplicación con tres módulos funcionales, que permita a sus usuarios utilizar un teléfono inteligente de manera sencilla e intuitiva.

A continuación, se refina el objetivo principal en los siguientes objetivos específicos:

- Desarrollar un módulo que permita realizar llamadas mediante un marcador digital o haciendo uso de una agenda de contactos previamente cargada.
- Además, dicho módulo tendrá la posibilidad de modificar la agenda de contactos, mediante las operaciones básicas de añadir, editar y eliminar un contacto. También incluirá un registro de llamadas básico.
- Desarrollar un módulo con la funcionalidad de un laucher[1], donde el usuario pueda ver las aplicaciones instaladas y ejecutarlas. Además, también tendrá la posibilidad de añadir las que más utilice a un menú principal.
- Desarrollar un módulo que permita al usuario ver los medicamentos que tiene que tomar en el día, así como el total de estas. También podrá establecer recordatorios y marcar aquellas que ya se haya tomado.
- Además, dicho módulo tendrá la posibilidad de modificar el conjunto de los medicamentos, pudiendo añadir, editar y eliminar. Cada

[1] Un launcher es un programa informático que ayuda al usuario a localizar y arrancar otros programas en un dispositivo.

medicamento tendrá diferentes campos, como las horas del día en las que debe tomarlos o su intervalo de fechas.

1.3 Estructura de la memoria

A continuación, se describe de manera breve la estructura de la memoria.

- Capítulo 1: En este capítulo se describe la motivación del trabajo, los objetivos y la estructura de la memoria.
- Capítulo 2: En este capítulo se estudian herramientas similares a la que se ha realizado en el trabajo.
- Capítulo 3: En este capítulo se describe la tecnología utilizada para implementar el proyecto.
- Capítulo 4: En este capítulo se detalla la planificación del proyecto y sus fases de desarrollo.
- Capítulo 5: En este capítulo se definen los actores y casos de uso que se explicarán mediante tablas junto a sus requisitos.
- Capítulo 6: En este capítulo se define el modelo de datos de cada módulo funcional, explicando las diferentes tablas.
- Capítulo 7: En este capítulo se explicará los patrones arquitectónicos y de diseño utilizados.
- Capítulo 8: En este capítulo se realizará un estudio sobre el diseño e implementación de los casos de uso más relevantes.
- Capítulo 9: En este capítulo se explicarán las conclusiones y el trabajo futuro del proyecto.
- Anexo: Manual de usuario.

Chapter - Introduction

This chapter will explain the motivation behind this work, the objectives to be achieved and how the report is structured.

Motivation

Older people are the forgotten ones of our generation. While technology is advancing and practically everything is done through a screen, many people, especially our elders, are the ones who suffer the most from this advance, this is known as the digital divide.

Generally, both our generation, as well as our parents' generation, we do well with new technologies. If we look at the past, we see that most activities such as, for example, making a purchase or carrying out a bank transaction, used to require leaving home. However, all that has changed, and thanks to technological advances, practically everything can be done through a screen, saving time and money. That's why every day we see how all businesses are adapting, whether it's a supermarket making home deliveries, or a bank closing branches in favour of using their app.

However, if we try to teach any older person with difficulties using a smartphone, possibly bought by a family member, how to use these complex applications, they will end up frustrated.

To solve this digital divide, this work has proposed the development of an application accessible to any type of user, especially for the elderly. This work does not aim to solve this big problem, but to contribute a grain of sand so that those people who are afraid of using a smartphone can make a basic use of it.

By simplifying existing applications, it will be possible for users to independently manage basic applications such as a contact book, organize a home screen with their favourite applications and exploit the potential of applications by means of a medication reminder.

Objectives

The main objective of the work is the implementation of an application with three functional modules, which allows its users to use a smartphone in a simple and intuitive way.

The main objective is then refined into the following specific objectives:

- Develop a module that allows calls to be made via a digital dialer or by making use of a previously loaded contact book.
- In addition, this module will have the possibility to modify the contact book, by means of the basic operations of adding, editing, and deleting a contact. It will also include a basic call log.
- Develop a module with the functionality of a laucher[2], where the user can see the installed applications and run them. In addition, the user will also have the possibility to add the most used applications to a main menu.
- Develop a module that allows the user to see the medicines he/she has to take during the day, as well as the total of these. They will also be able to set reminders and mark those they have already taken.
- In addition, this module will have the possibility of modifying the set of medicines, being able to add, edit and delete. Each medicine will have different fields, such as the times of day at which it should be taken or its date range.

[2] A launcher is a computer program that helps the user to locate and start other programs on a device.

Memory structure

In the following, the structure of the report is briefly described.

- Chapter 1: This chapter describes the motivation for the work, the objectives, and the structure of the report.
- Chapter 2: This chapter discusses tools similar to the one used in the work.
- Chapter 3: This chapter describes the technology used to implement the project.
- Chapter 4: This chapter details the planning of the project and its development phases.
- Chapter 5: This chapter defines the actors and use cases that will be explained in tables together with their requirements.
- Chapter 6: This chapter defines the data model of each functional module, explaining the different tables.
- Chapter 7: This chapter explains the architectural and design patterns used.
- Chapter 8: This chapter will study the design and implementation of the most relevant use cases.
- Chapter 9: This chapter will explain the conclusions and future work of the project.
- Annex: User manual.

Capítulo 2 - Estado del arte

En este capítulo, se describirán las características principales de algunas aplicaciones similares a la desarrollada en este proyecto.

La aplicación desarrollada contiene tres módulos diferenciados, con propósitos distintos, por lo que el análisis se va a realizar haciendo dicha distinción.

Teléfono de Google [1]: Es la aplicación por defecto que viene instalada en la gran mayoría de dispositivos Android, a excepción de algunos fabricantes que incluyen la suya propia.

Sus principales características son:

- Creación de contactos con bastantes posibles campos a rellenar para distintos propósitos.
- Filtros antispam para llamadas
- Sincronización con la nube

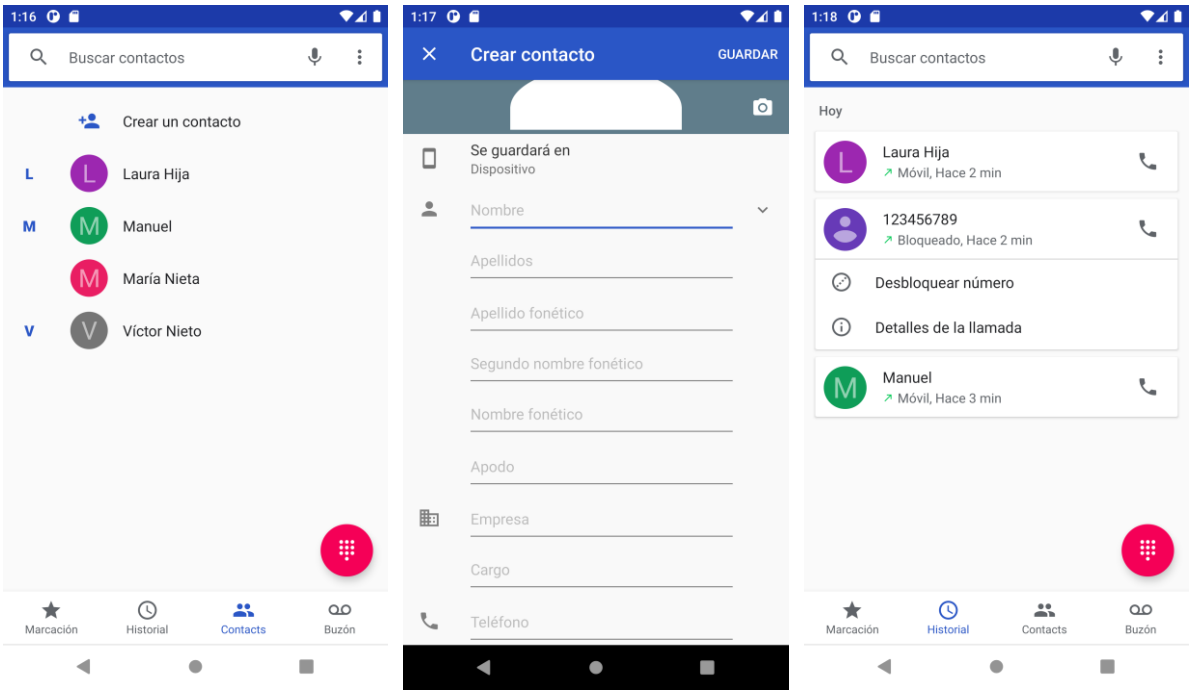

Figura 1 - Diferentes ventanas que ofrece Teléfono de Google

Pixel Launcher [2]: Es el launcher por defecto que viene instalado en los teléfonos de Google. En general, cada fabricante introduce una capa de personalización sobre el sistema operativo que consiste, entre otras cosas, en diseñar su propio launcher. A pesar de las diferencias, todos suelen tener las mismas bases. En primer lugar, tenemos varias pantallas de inicio donde podemos añadir aplicaciones o widgets. En segundo lugar, tendremos un cajón donde estarán todas las apps instaladas.

Sus principales características son:

- Es posible modificar los elementos de la pantalla manteniendo pulsado
- Incluye una pantalla de noticias
- Incluye un acceso al sistema de búsqueda Google Search

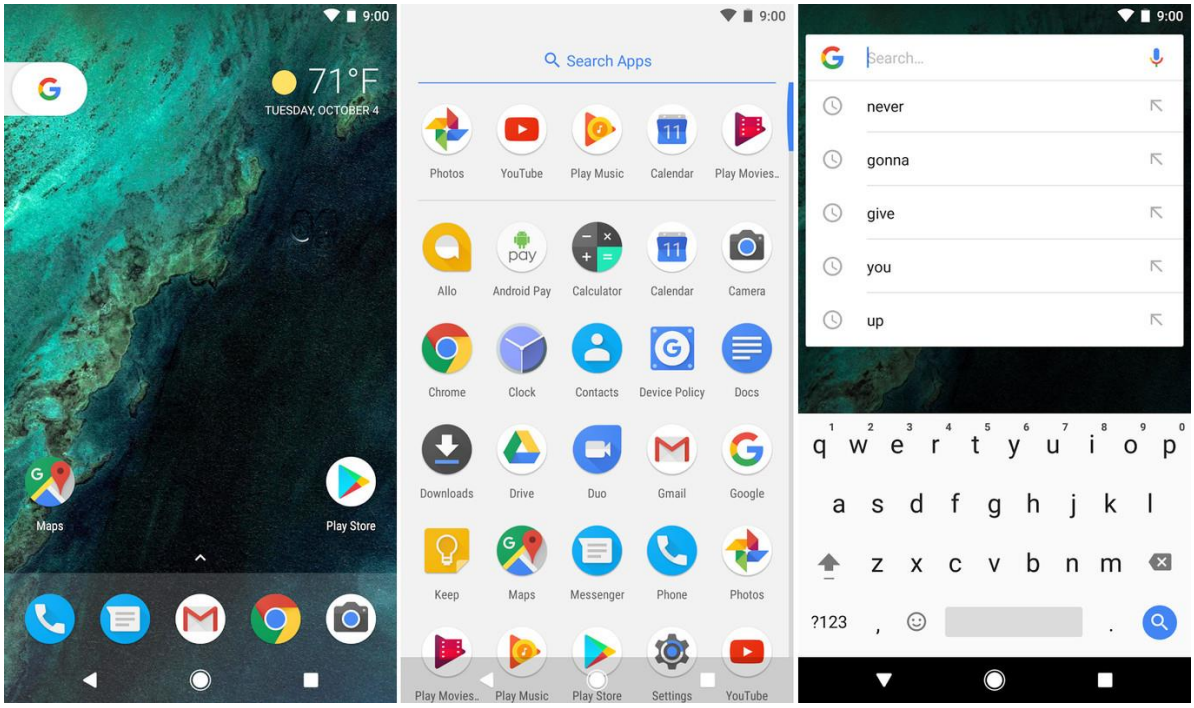

Figura 2 - Diferentes ventanas que ofrece Pixel Launcher

Recordatorio de Medicación MyTherapy [3]: Es un recordatorio gratuito de medicamentos, aunque tiene más funcionalidades. Es el más descargado de la tienda de Google y cuenta con una buena valoración. En general, permite llevar un control sobre los medicamentos que toma el usuario y realizar seguimientos a largo plazo.

Sus principales características son:

- Recordatorio de medicamentos con alarmas
- Permite llevar un control de medidas como el peso o la presión arterial.
- Incluye un diario de salud

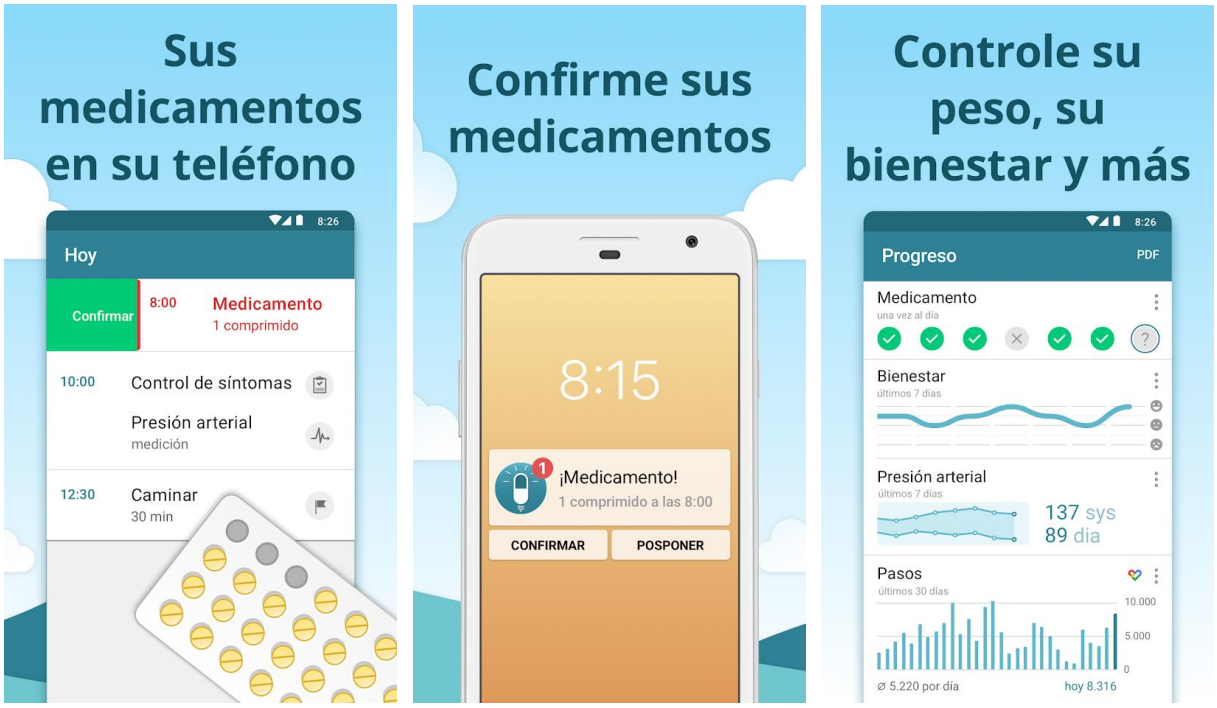

Figura 3 - Diferentes ventanas que ofrece MyTherapy

Capítulo 3 - Tecnología empleada

En este capítulo se van a presentar las tecnologías utilizadas en el proyecto.

3.1 Herramientas de la parte cliente

3.1.1 Android Studio

Android Studio [4] es el entorno de desarrollo integrado oficial para la plataforma Android. Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas GNU/Linux, macOS, Microsoft Windows y Chrome OS. Ha sido diseñado específicamente para el desarrollo de Android.

Entre sus características principales están el soporte para construcción basada en Gradle, un dispositivo virtual para ejecutar y probar aplicaciones o un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes a la interfaz de usuario.

3.2 Herramientas de la parte servidor

3.2.1 Proveedor de Contenidos

El Proveedor de Contenidos [5] es una interfaz utilizada para gestionar el acceso a los datos de una aplicación proporcionando una forma de compartirlos con otras aplicaciones. La plataforma de Android dispone de algunos proveedores propios, como pueden ser el proveedor de contactos [6], o el acceso al registro de llamadas.

La principal ventaja es que ofrece un control detallado sobre los permisos de acceso a los datos, de tal forma que, si creamos nuestro propio proveedor, podemos establecer permisos de lectura y/o escritura.

3.2.2 Proveedor de Contactos

El proveedor de Contactos [6] es una API de Android que administra el repositorio central de datos de personas en los dispositivos Android. Funciona como un proveedor

de contenidos. Principalmente, es la fuente de datos que se ve en la aplicación de contactos predeterminada del dispositivo y, además, se puede acceder a dichos datos desde tus propias aplicaciones.

La principal ventaja de utilizar este proveedor es que cuando se realizan modificaciones desde la aplicación, los cambios se ven reflejados en el sistema y todas las aplicaciones que accedan a los contactos podrán verlos.

3.2.3 SharedPreferences

SharedPreferences [7] es una API de Android que permite almacenar pares clave-valor de forma persistente en el sistema. Si se tiene una colección de datos relativamente pequeña y sencilla, la ventaja de utilizar esta interfaz es que no se necesita la creación de un fichero para almacenar, por ejemplo, la configuración de una app.

En general, se pueden almacenar tipos básicos (como cadenas de caracteres, enteros o booleanos), asociados a una clave única. Cuando se inserta un par con una clave que ya se existe, su valor se reemplaza y cuando se busca una que no existe, nos devuelve un valor por defecto. Si se manejan datos más complejos, hará falta hacer uso de otras herramientas.

3.2.4 Room

Room [8] es una biblioteca de Android que proporciona una capa de abstracción para SQLite que permite acceder a su base de datos sin problemas. Su principal ventaja es la abstracción, ya que Room mapea directamente los objetos a una base de datos, sin necesidad de crearla.

La librería funciona utilizando 3 componentes principales: la clase de la base de datos, las entidades de datos y los objetos de acceso a datos (DAOs), que serán explicados con detalle más adelante.

3.3 Otras herramientas

3.3.1 ViewModel

ViewModel [9] es un framework cuyo objetivo es almacenar y administrar los datos relacionados con la IU de manera optimizada para los ciclos de vida. Una de sus funcionalidades es la conservación de datos respecto a la vista tras cambios de configuración, como la rotaciones de pantallas.

La principal ventaja es dividir la propiedad de los datos de visualización de la lógica del controlador de IU. Es ideal para implementarlo en algunos patrones de diseño, como el MVVM [17].

3.3.2 LiveData

LiveData [10] es un componente de arquitectura Android para datos observables optimizado para ciclos de vida. De esta forma, respeta el ciclo de vida de otros componentes de las apps, garantizando que sólo actualice observadores de componentes que tienen un estado de ciclo de vida activo.

3.3.3 Gson

Gson [11] es una biblioteca de código abierto para Java, que permite la serialización y deserialización entre objetos Java y su representación en notación JSON. Fue desarrollada por Google como un proyecto interno, pero al final se publicó bajo una licencia Apache License 2.0.

3.3.4 ColorPicker

ColorPicker [12] es una biblioteca que permite generar un cuadro de diálogo simple con una cuadrícula de colores donde el usuario puede seleccionar uno. Su principal ventaja es la sencillez, unido a una gran personalización, ya que se pueden modificar los colores que se muestran, la forma, el mensaje, etc. Ha sido desarrollado bajo una licencia Apache 2.0 y se encuentra disponible en GitHub.

3.3.5 Git

Git [13] es un software de control de versiones, pensado para el mantenimiento, compatibilidad y eficiencia de aplicaciones con un gran número de archivos de código fuente. Su propósito es llevar un registro de los cambios realizados en los archivos que se realizan en local, así como coordinar el trabajo sobre ficheros compartidos en un repositorio. Es un software libre distribuible bajo los términos de la versión 2 de la Licencia Pública General de GNU.

Entre sus características se encuentran el apoyo al desarrollo no lineal, gestión eficiente de proyectos grandes y el control de versiones, ya que es posible volver a versiones previas.

3.3.6 Java

Java [14] es un lenguaje de programación orientado a objetos y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems y que en 2010 fue adquirida por Oracle. Su sintaxis deriva en gran medida de C y C++, pero con menos utilidades de bajo nivel.

Su principal característica es que las aplicaciones desarrolladas con Java pueden ejecutarse en cualquier máquina virtual Java (JVM), sin importar la arquitectura de la computadora subyacente.

3.3.7 Modelio

Modelio [15] es un entorno de modelado desarrollado por Modeliosoft para el diseño de diagramas UML. Es un software de código abierto bajo la licencia GPLv3. Sus principales ventajas son la gran variedad de diagramas que contiene y la posibilidad de obtener el código correspondiente de los diagramas creados en el proyecto.

Capítulo 4 - Planificación

En este capítulo se van a detallar las tareas a realizar durante el proyecto, desde las primeras fases para la definición de objetivos y casos de uso, hasta la implementación.

4.1 Lista de tareas por fases

En primer lugar, se van a definir las distintas tareas a realizar, agrupadas por fases u objetivos. Se van a diferenciar las fases de planificación y diseño, de las fases concretas de desarrollo de cada módulo funcional de nuestra aplicación.

4.1.1 Fase de planificación y diseño

- Definición de objetivos
- Generación de la planificación
- Definición de casos de uso

4.1.2 Fase de investigación

- Investigación y descarga de herramientas
- Familiarización con el entorno
- Lectura de documentación básica

4.1.3 Fase de desarrollo del módulo teléfono

- Diseño del modelo de datos
- Agrupación de funcionalidades por vistas
- Diseño de un mockup a papel
- Desarrollo de la aplicación
- Testeo y resolución de bugs

4.1.4 Fase de desarrollo del módulo launcher

- Diseño del modelo de datos
- Agrupación de funcionalidades por vistas
- Diseño de un mockup a papel
- Desarrollo de la aplicación
- Testeo y resolución de bugs

4.1.5 Fase de desarrollo del módulo medicamentos

- Diseño del modelo de datos
- Agrupación de funcionalidades por vistas
- Diseño de un mockup a papel
- Desarrollo de la aplicación
- Testeo y resolución de bugs

4.2 Diagrama de Gantt

En segundo lugar, se va a definir la planificación utilizando un diagrama de Gantt. Cada tarea aparecerá con el color correspondiente a su fase de desarrollo.

Para cada tarea, se va a definir una fecha de inicio, una fecha de fin y el número total de días empleados. El resto de las columnas corresponden a las semanas, que va de 1 a 30, comenzando el 4 de octubre de 2021 y finalizando el 2 de mayo de 2022.

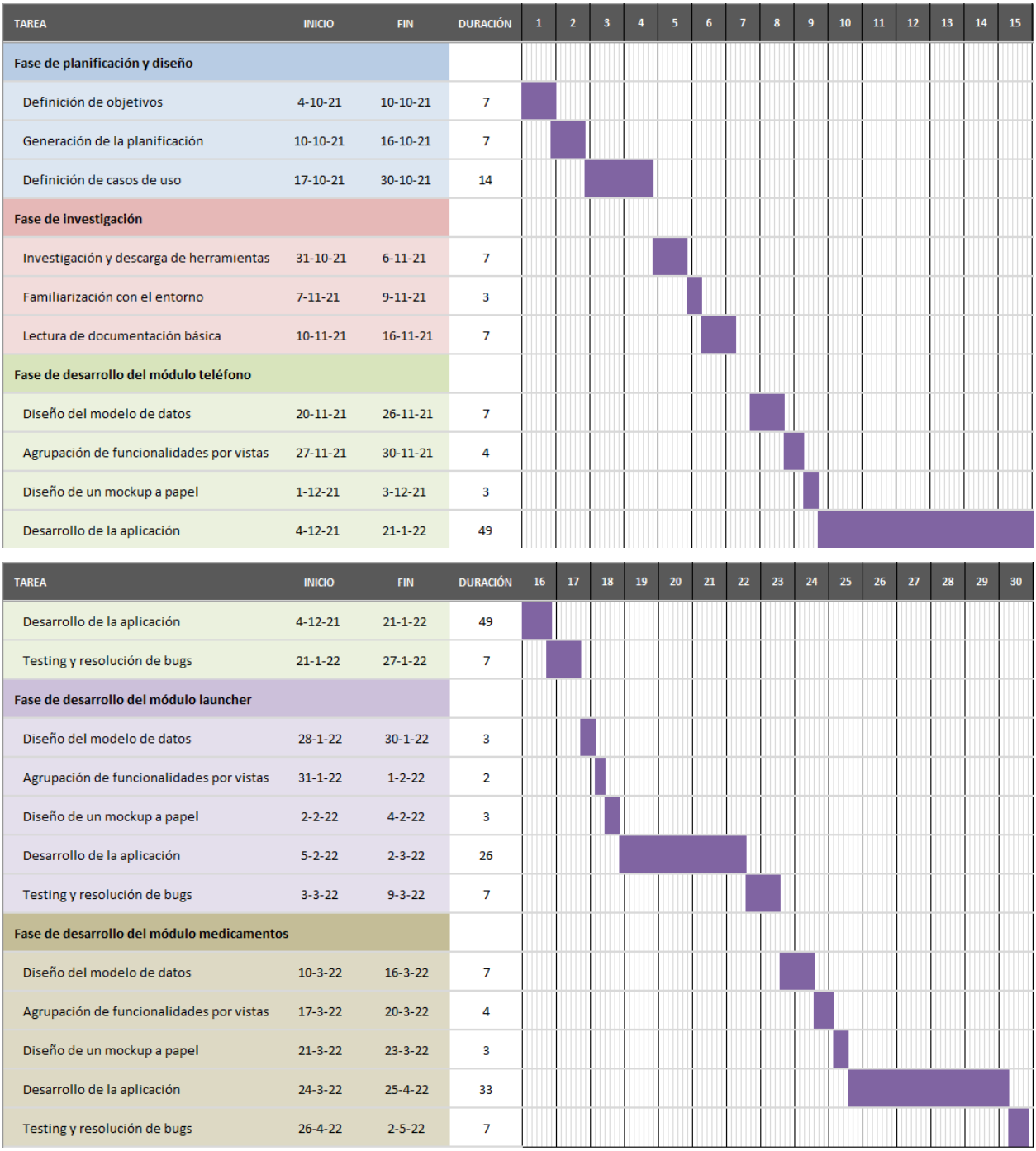

Figura 4 - Diagrama de Gantt

Capítulo 5 - Casos de uso

En este capítulo, se describen los actores y casos de uso definidos para la aplicación. Con el objetivo de facilitar la descripción de los casos de uso, estos se han agrupado en módulos funcionales.

5.1 Actores del sistema

A continuación, se describen el único actor del sistema. Puesto que la aplicación va a funcionar como una capa sobre el sistema operativo Android, no se va a necesitar ningún otro actor como, por ejemplo, un administrador.

- **Usuarios**

  Representan a los usuarios finales que van a utilizar el sistema. Ya que la aplicación está destinada a personas mayores, no será necesario hacer ningún tipo de registro. Podrán acceder a todas las funcionalidades sólo con abrir la aplicación.

5.2 Módulo de llamadas

En esta sección, se muestran los casos de uso del módulo de llamadas.

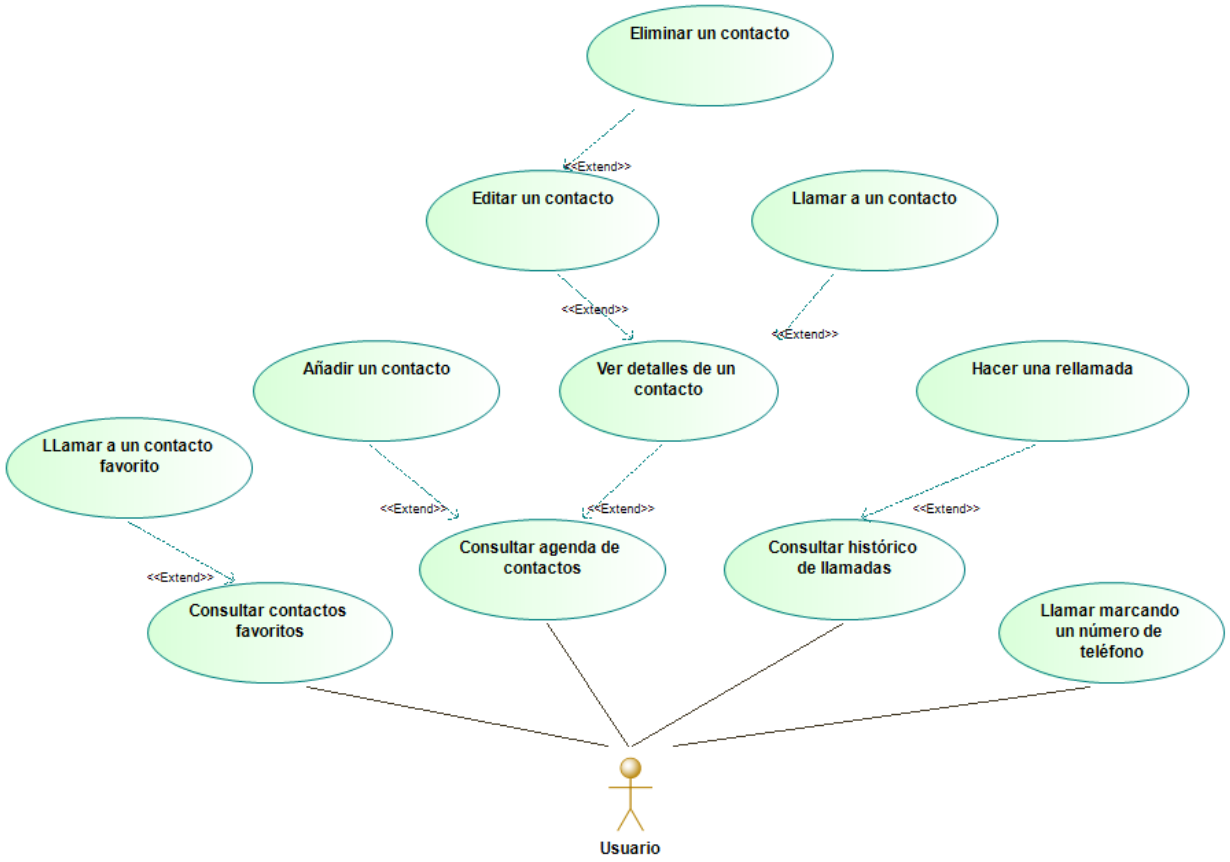

Figura 5 - Diagrama de los casos de uso del módulo llamadas

	Consultar contactos favoritos	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá ver los contactos favoritos.	
Entrada	NA	
Salida	Lista de contactos.	
Secuencia normal	Paso	Acción

	1	El usuario accede al apartado de llamadas, seleccionando el icono correspondiente en el box de aplicaciones.
	2	El sistema muestra la lista con los contactos favoritos, entre otras opciones.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	Si no hay ningún contacto favorito, se muestra un mensaje informativo.	

Tabla 1 - Consultar contactos favoritos

	Llamar a un contacto favorito	
Versión	1.0.0	
Prioridad	Alta	
Precondición	El contacto debe existir en la lista de favoritos.	
Descripción	El usuario podrá realizar una llamada a un contacto favorito, sin necesidad de introducir su número de teléfono.	
Entrada	Contacto telefónico.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	Desde la consulta de contactos favoritos, el usuario pulsa sobre el contacto que desea llamar
	2	El sistema realiza la llamada telefónica.
Postcondición	Se está realizando una llamada.	
Excepciones	Paso	Acción
	2	El número de teléfono no existe y el operador telefónico se lo indica al usuario.
Comentarios	La aplicación no puede comprobar si el número de teléfono existe realmente hasta que no se realiza la llamada.	

Tabla 2 - Llamar a un contacto favorito

	Consultar agenda de contactos	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá ver la agenda de contactos.	
Entrada	NA	
Salida	Listado de contactos almacenados en la agenda.	
Secuencia normal	Paso	Acción
	1	El usuario accede al apartado de llamadas, seleccionando el icono correspondiente en el box de aplicaciones.
	2	El usuario selecciona la opción de mostrar agenda de contactos.
	3	Se muestra una lista con un número pequeño de contactos. Se podrán consultar todos mediante una barra de "scroll" y de cada contacto, se muestra su nombre, número y fotografía.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	Si no hay ningún contacto almacenado, se muestra un mensaje informativo.	

Tabla 3 - Consultar agenda de contactos

	Añadir un contacto
Versión	1.0.0
Prioridad	Media
Precondición	NA
Descripción	El usuario podrá añadir un contacto.
Entrada	Nombre, número y foto (opcional) del contacto.
Salida	NA

Secuencia normal	Paso	Acción
	1	Desde la consulta de la agenda, el usuario pulsa el botón para añadir un contacto.
	2	Se muestra una ventana con campos vacíos que el usuario debe rellenar y pulsar el botón para guardar.
	3	El sistema añade el nuevo contacto.
Postcondición	Se añade un contacto.	
Excepciones	Paso	Acción
Comentarios	El campo número sólo permite dígitos y la foto puede seleccionarse de la galería o mediante la cámara.	

Tabla 4 - Añadir un contacto

	Ver detalles de un contacto	
Versión	1.0.0	
Prioridad	Alta	
Precondición	El contacto debe estar almacenado en la agenda de contactos.	
Descripción	El usuario podrá consultar la información de un contacto.	
Entrada	NA	
Salida	Información del contacto seleccionado.	
Secuencia normal	Paso	Acción
	1	Desde la consulta de la agenda, el usuario pulsa sobre el contacto deseado.
	4	Se mostrará la información ampliada del contacto seleccionado. Foto de perfil, nombre, número de teléfono, un botón para añadir a favoritos, otro para realizar una llamada y otro para editar.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	El contacto podría no contener fotografía de perfil.	

Tabla 5 - Ver detalles de un contacto

	Editar un contacto	
Versión	1.0.0	
Prioridad	Baja	
Precondición	El contacto debe estar almacenado en la agenda de contactos.	
Descripción	El usuario podrá editar un contacto.	
Entrada	Nombre, número y foto (opcional) del contacto.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	Desde la ventana para ver los detalles del contacto, el usuario pulsa el botón de edición.
	2	Se mostrará una ventana con campos que el usuario puede modificar y un botón para guardar los cambios.
	3	El sistema modificará el contacto.
Postcondición	Se ha modificado el contacto.	
Excepciones	Paso	Acción
Comentarios	El campo número sólo permite dígitos y la foto puede seleccionarse de la galería o mediante la cámara.	

Tabla 6 - Editar un contacto

	Llamar a un contacto	
Versión	1.0.0	
Prioridad	Alta	
Precondición	El contacto debe estar almacenado en la agenda de contactos.	
Descripción	El usuario podrá realizar una llamada a un contacto concreto, sin necesidad de introducir su número de teléfono.	
Entrada	Contacto telefónico.	
Salida	NA	
Secuencia normal	Paso	Acción

	1	Desde la ventana para ver los detalles del contacto, el usuario pulsa el botón de llamar.
	2	El sistema realiza la llamada telefónica.
Postcondición	Se está realizando una llamada.	
Excepciones	Paso	Acción
	2	El número de teléfono no existe y el operador telefónico se lo indica al usuario.
Comentarios	La aplicación no puede comprobar si el número de teléfono existe realmente hasta que no se realiza la llamada.	

Tabla 7 - Llamar a un contacto

	Eliminar un contacto	
Versión	1.0.0	
Prioridad	Baja	
Precondición	El contacto debe estar almacenado en la agenda de contactos.	
Descripción	El usuario podrá añadir un contacto.	
Entrada	Contacto telefónico.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	Desde la vista para editar un contacto, el usuario pulsa el botón de eliminar.
	2	Se muestra un mensaje de confirmación que el usuario debe aceptar.
	3	El sistema elimina el contacto.
Postcondición	Se elimina el contacto.	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 8 - Eliminar un contacto

	Consultar histórico de llamadas	
Versión	1.0.0	
Prioridad	Media	
Precondición	NA	
Descripción	El usuario podrá consultar el histórico de llamadas.	
Entrada	NA	
Salida	Listado con las llamadas recientes.	
Secuencia normal	Paso	Acción
	1	El usuario accede al apartado de llamadas, seleccionando el icono correspondiente en el box de aplicaciones.
	2	El usuario selecciona la opción de consultar el histórico de llamadas.
	3	Se muestra una lista con las llamadas recientes en orden descendente por fecha. En cada una se muestra el número o contacto (si tenemos el número en la agenda), la duración y si ha sido una llamada entrante o saliente.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	Si no hay ninguna llamada reciente, mostrará un mensaje informativo.	

Tabla 9 - Consultar histórico de llamadas

	Hacer una rellamada
Versión	1.0.0
Prioridad	Baja
Precondición	Debe existir al menos una llamada en el registro
Descripción	El usuario podrá realizar una rellamada a un número desde la vista del registro de llamadas.
Entrada	Número telefónico.
Salida	NA

Secuencia normal	Paso	Acción
	1	Desde la consulta del registro de llamadas, el usuario selecciona el número que desea rellamar.
	2	El sistema realiza la llamada telefónica.
Postcondición	Se está realizando una llamada.	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 10 - Hacer una rellamada

	Llamar marcando un número de teléfono	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá realizar una llamada telefónica.	
Entrada	Número de teléfono.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede al apartado de llamadas, seleccionando el icono correspondiente en el box de aplicaciones.
	2	El usuario selecciona la opción de introducir número manualmente.
	3	El usuario introduce el número y pulsa el botón de llamada.
	4	El sistema realiza la llamada telefónica.
Postcondición	El sistema está realizando una llamada.	
Excepciones	Paso	Acción
	4	El número de teléfono no existe y el operador telefónico se lo indica al usuario.
Comentarios	La aplicación no puede comprobar si el número de teléfono existe realmente hasta que no se realiza la llamada.	

Tabla 11 - Llamar marcando un número de teléfono

5.3 Módulo launcher

En esta sección se muestran los casos de uso del módulo launcher.

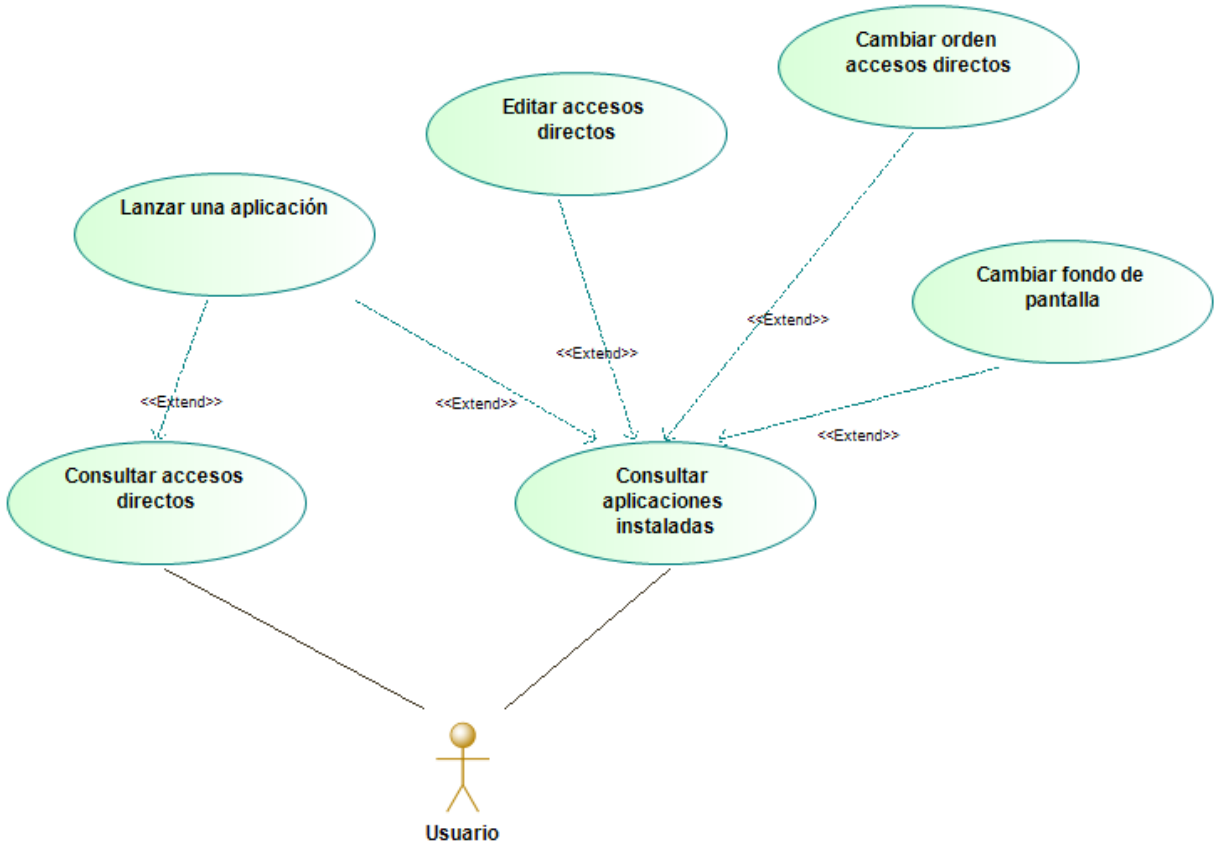

Figura 6 - Diagrama de los casos de uso del módulo launcher

	Consultar accesos directos	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá visualizar la lista de aplicaciones con acceso directo.	
Entrada	NA	
Salida	Lista de aplicaciones.	
Secuencia normal	Paso	Acción
	1	Al iniciar la aplicación, el usuario visualiza la lista de aplicaciones con acceso directo.

Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	Si no existen aplicaciones añadidas, se mostrará el escritorio vacío.	

Tabla 12 - Consultar accesos directos

	Consultar aplicaciones instaladas	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá visualizar la lista de aplicaciones instaladas.	
Entrada	NA	
Salida	Lista de aplicaciones.	
Secuencia normal	Paso	Acción
	1	Una vez iniciada la aplicación, el usuario debe pulsar sobre el icono para abrir el box de aplicaciones.
	2	El sistema mostrará todas las aplicaciones instaladas.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 13 - Consultar aplicaciones instaladas

	Lanzar una aplicación
Versión	1.0.0
Prioridad	Alta
Precondición	La aplicación debe estar instalada.
Descripción	El usuario podrá visualizar lanzar una aplicación instalada en el sistema.
Entrada	NA
Salida	NA

Secuencia normal	Paso	Acción
	1	Desde la lista de aplicaciones instaladas o los accesos directos, el usuario pulsa sobre la aplicación que desea abrir.
	2	El sistema abre la aplicación seleccionada.
Postcondición	Se ha lanzado una aplicación.	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 14 - Lanzar una aplicación

	Editar accesos directos	
Versión	1.0.0	
Prioridad	Media	
Precondición	NA	
Descripción	El usuario podrá editar los accesos directos de la pantalla principal.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	Desde la lista de aplicaciones instaladas, el usuario debe pulsar el botón para cambiar los accesos directos.
	2	El sistema muestra un checkbox en la fila de cada aplicación. El usuario marca o desmarca las casillas en función de su interés.
	3	El usuario pulsa el botón de guardar.
	4	El sistema actualiza los accesos directos.
Postcondición	Se ha actualizado la lista de accesos directos.	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 15 - Editar accesos directos

	Cambiar orden accesos directos	
Versión	1.0.0	
Prioridad	Media	
Precondición	NA	
Descripción	El usuario podrá editar el orden de los accesos directos de la pantalla principal.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	Desde la lista de aplicaciones instaladas, el usuario debe pulsar el botón para cambiar el orden los accesos directos.
	2	El sistema muestra únicamente las aplicaciones con acceso directo. Además, se muestran dos botones para subir y bajar la aplicación en función de su orden en la lista. El usuario interactúa con los botones en función de su interés.
	3	El usuario pulsa el botón de guardar.
	4	El sistema actualiza el orden de los accesos directos.
Postcondición	Se ha actualizado el orden de la lista de accesos directos.	
Excepciones	Paso	Acción
Comentarios	Si no hay ningún acceso directo, no aparecerá ningún elemento en la lista.	

Tabla 16 - Cambiar orden accesos directos

	Cambiar fondo de pantalla
Versión	1.0.0
Prioridad	Media
Precondición	La foto que el usuario quiere poner debe existir en la galería.
Descripción	El usuario podrá cambiar el fondo de pantalla
Entrada	Fotografía.

Salida	NA	
Secuencia normal	Paso	Acción
	1	Desde la lista de aplicaciones instaladas, el usuario debe pulsar el botón para cambiar el fondo de pantalla
	2	El sistema abre la galería y el usuario elige una foto que tenga almacenada.
	3	El sistema actualiza el fondo de pantalla.
Postcondición	Se ha actualizado el fondo de pantalla.	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 17 - Cambiar fondo de pantalla

5.4 Módulo medicamentos

En esta sección se muestran los casos de uso del módulo medicamentos.

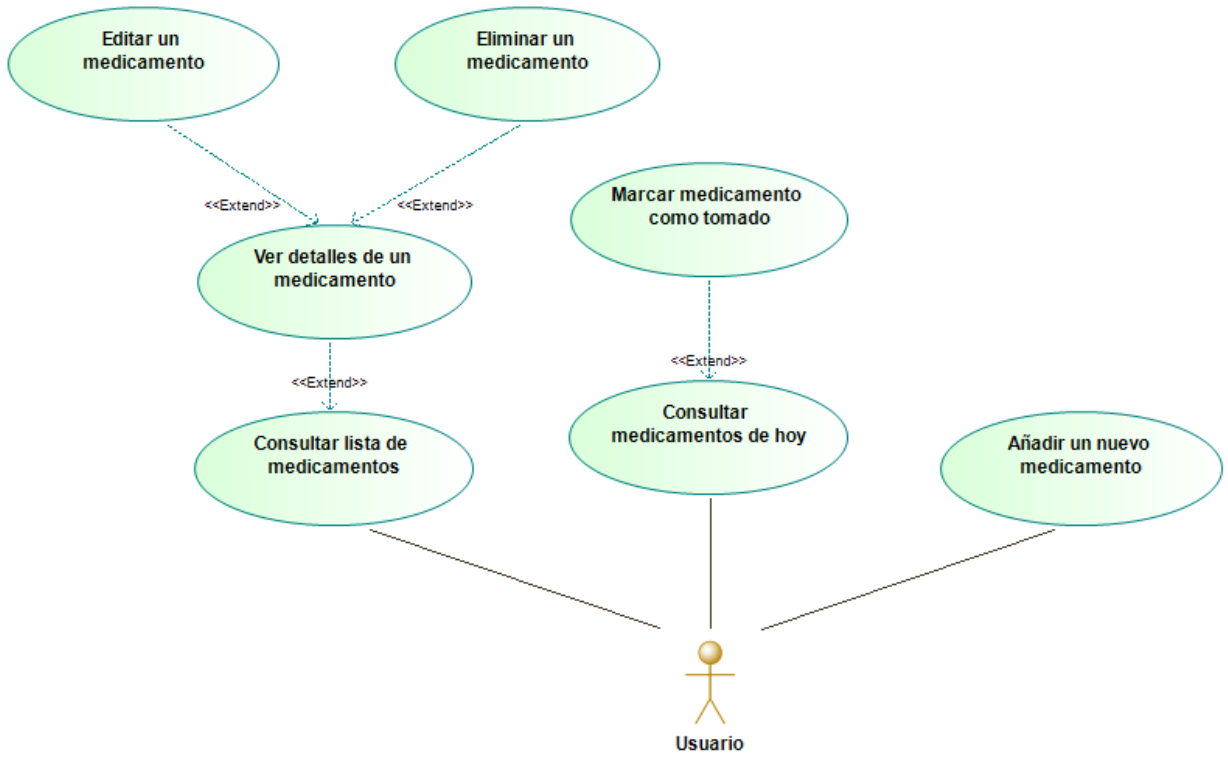

Figura 7 - Diagrama de los casos de uso del módulo launcher

	Consultar lista de medicamentos	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá ver los medicamentos guardados.	
Entrada	NA	
Salida	Lista de medicamentos.	
Secuencia normal	Paso	Acción
	1	El usuario accede al apartado de medicamentos, seleccionando el icono correspondiente en el box de aplicaciones.
	2	Se abrirá una nueva ventana y el usuario pulsa el botón para mostrar la pestaña correspondiente al listado de medicamentos.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	Si no hay ningún medicamento, se muestra un mensaje informativo.	

Tabla 18 - Consultar lista de medicamentos

	Ver detalles de un medicamento	
Versión	1.0.0	
Prioridad	Alta	
Precondición	El medicamento debe haber sido creado.	
Descripción	El usuario podrá ver los detalles de un medicamento.	
Entrada	NA	
Salida	Medicamento.	
Secuencia normal	Paso	Acción
	1	Desde la pantalla de la lista de medicamentos, el usuario pulsa sobre el medicamento que desea consultar.

	2	El sistema muestra los detalles de un medicamento, como la el nombre, la foto, cuando debe tomarlo, etc.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 19 - Ver detalles de un medicamento

	Editar un medicamento	
Versión	1.0.0	
Prioridad	Media	
Precondición	El medicamento debe haber sido creado.	
Descripción	El usuario podrá editar un medicamento	
Entrada	Nuevos datos del medicamento.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	Desde la pantalla de visualizar detalles de un medicamento, el usuario selecciona la opción para editar.
	2	Se mostrarán múltiples ventanas para rellenar los datos correspondientes al mismo. Al final, el usuario pulsa el botón de aceptar.
	3	El sistema actualiza el medicamento.
Postcondición	El medicamento ha sido actualizado.	
Excepciones	Paso	Acción
Comentarios	Al ser tanta información, los campos se irán mostrando en diferentes pantallas y no es necesario que modifique todos los datos del medicamento.	

Tabla 20 - Editar un medicamento

	Eliminar un medicamento	
Versión	1.0.0	
Prioridad	Media	
Precondición	El medicamento debe haber sido creado.	
Descripción	El usuario podrá eliminar un medicamento.	
Entrada	NA	
Salida	Medicamento.	
Secuencia normal	Paso	Acción
	1	Desde la pantalla de visualizar detalles de un medicamento, el usuario selecciona la opción para editar.
	2	Se mostrará un mensaje de confirmación que el usuario debe aceptar.
	3	El sistema elimina el medicamento.
Postcondición	El medicamento ha sido eliminado.	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 21 - Eliminar un medicamento

	Consultar medicamentos de hoy	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá ver los medicamentos que debe tomar hoy.	
Entrada	NA	
Salida	Lista de medicamentos.	
Secuencia normal	Paso	Acción
	1	El usuario accede al apartado de medicamentos, seleccionando el icono correspondiente en el box de aplicaciones.

	2	Se abre una nueva ventana y el sistema mostrará la lista de los medicamentos que el usuario debe tomar hoy.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	Si no hay ningún medicamento que deba tomar, se muestra un mensaje informativo.	

Tabla 22 - Consultar medicamentos de hoy

	Marcar medicamento como tomado	
Versión	1.0.0	
Prioridad	Alta	
Precondición	El medicamento debe existir.	
Descripción	El usuario podrá confirmar que ha tomado el medicamento.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede al apartado de medicamentos, seleccionando el icono correspondiente en el box de aplicaciones.
	2	Se abrirá una nueva ventana con los medicamentos que debe tomar el usuario hoy y el usuario debe pulsar el botón de completado del medicamento correspondiente.
	3	El sistema marcará el medicamento como tomado.
Postcondición	Se ha marcado el medicamento como completado.	
Excepciones	Paso	Acción
Comentarios	NA	

Tabla 23 - Marcar medicamento como tomado

	Añadir un nuevo medicamento	
Versión	1.0.0	
Prioridad	Alta	
Precondición	NA	
Descripción	El usuario podrá añadir un nuevo medicamento.	
Entrada	Datos del medicamento.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede al apartado de medicamentos, seleccionando el icono correspondiente en el box de aplicaciones.
	2	Se abrirá una nueva ventana y el usuario pulsa el botón para añadir un nuevo medicamento.
	3	Se mostrarán múltiples ventanas para rellenar los datos correspondientes al mismo. Al final, el usuario pulsa el botón de aceptar.
	4	El sistema añade el nuevo medicamento.
Postcondición	Se ha añadido un nuevo medicamento.	
Excepciones	Paso	Acción
Comentarios	Al ser tanta información, los campos se irán mostrando en diferentes pantallas.	

Tabla 24 - Añadir un nuevo medicamento

Capítulo 6 - Modelo de datos

En este capítulo se describe el modelo de datos utilizado para la persistencia de la información en el sistema desarrollado.

6.1 Módulo teléfono

En esta sección se describen las tablas que almacenan la información de los contactos y del registro de llamadas en el sistema operativo. Esta información se gestiona a través del componente proveedor de contactos y el proveedor del registro de llamadas respectivamente.

Si bien es cierto que se podría haber desarrollado una base de datos propia para este propósito, no tendría demasiado sentido. En primer lugar, ya existe una base de datos desarrollada para ello y se tendrían datos duplicados. En segundo lugar, si el usuario se instala una aplicación para almacenar contactos, todas las modificaciones que realicen en esta aplicación, se verán reflejadas en la descargada.

El proveedor de contactos administra el repositorio central de personas en dispositivos Android y mantiene tres tipos de datos acerca de cada una, que se corresponde con una tabla diferente, como se ilustra en la Figura 8.

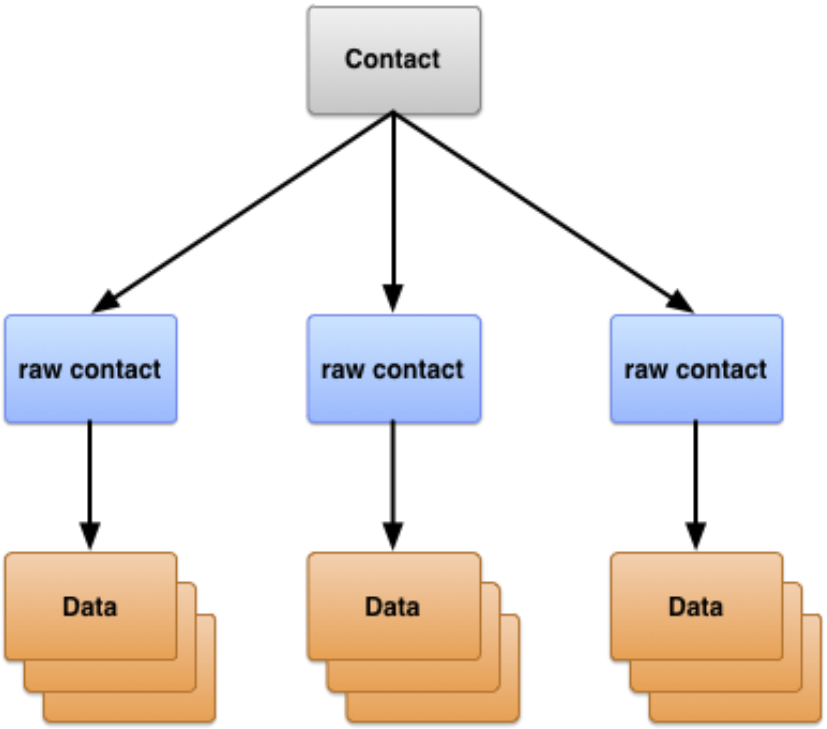

Figura 8 - Estructura de las tablas del proveedor de contactos

Las tres tablas se denominan con los nombres de sus clases de contrato, y nos definen constantes para los URI[3] de contenido, es decir, como se debe utilizar la base de datos. Para cada una, se va a detallar su propósito general y las columnas más importantes que han sido útiles para el desarrollo del módulo.

Por otro lado, el proveedor del registro de llamadas tiene una estructura más sencilla, que será explicada más adelante en esta sección.

[3] URI (de sus siglas en inglés, Uniform Resource Identifier), es una secuencia de caracteres compacta que identifica un recurso abstracto o físico. Definido en RFC 2396 [15].

6.1.1 Tabla de contactos sin procesar (RawContacts)

Un contacto sin procesar representa la información de una persona que proviene desde un único tipo de cuenta y nombre de cuenta. Esto se debe a que se admiten datos de diferentes servicios para almacenar contactos

La mayoría de los datos se almacenan en una o más filas de la tabla de datos (Data), donde cada fila contiene una columna con el identificador de la fila principal en esta tabla.

Las columnas más destacables son:

- **_ID:** Identificador único.
- **ACCOUNT_NAME:** Nombre de cuenta del tipo de cuenta que es el origen del contacto.
- **ACCOUNT_TYPE:** El tipo de cuenta que es el origen del contacto.
- **DELETED:** Marca de eliminado que se mantiene hasta que se elimina definitivamente del repositorio.

Para entenderlo mejor, se va a explicar mediante un ejemplo. Si el usuario inicia sesión en Gmail con el correo "prueba@gmail.com" y agrega a "Pedro Picapiedra" y, después sigue en Twitter a "@PPicapiedra" (ID de Pedro Picapiedra en Twitter), se habrán generado dos contactos sin procesar:

1. Un primer contacto sin procesar para "Pedro Picapiedra" asociado a "prueba@gmail.com" y cuyo tipo de cuenta es Google.
2. Un segundo contacto sin procesar para "Pedro Picapiedra" asociado con "@prueba" (nuestro ID de Twitter) y cuyo tipo de cuenta es Twitter.

6.1.2 Tabla de datos (Data)

Como se ha mencionado previamente, en esta tabla se guardan los contactos vinculados a un identificador del contacto sin procesar. Esto permite que un contacto tenga múltiples instancias del mismo tipo de dato, por ejemplo, un teléfono fijo y otro móvil, que se guardan como filas independientes pero vinculadas al mismo contacto sin procesar.

Las columnas que componen esta tabla se van a dividir en las que tienen un nombre genérico y las que tienen un nombre descriptivo. Esto se debe a que, en función del tipo de dato, las genéricas tienen significados diferentes.

Las columnas más destacables con nombre descriptivo son:

- **_ID:** Identificador único.
- **MIMETYPE:** El tipo de datos que se guarda en esta fila como un tipo de MIME [4] personalizado.
- **RAW_CONTACT_ID:** El valor de la columna _ID del contacto sin procesar.
- **IS_PRIMARY:** Si el tipo de datos se repite para un contacto, indica si esta fila es el valor predeterminado.

Las columnas genéricas van desde **DATA1** a **DATA15** y generalmente se suele utilizar la primera para los datos más frecuentes. Además, existen otras clases adicionales que definen constantes del nombre de columna específico para una fila de esta tabla, como se muestra en el ejemplo de la Figura 9.

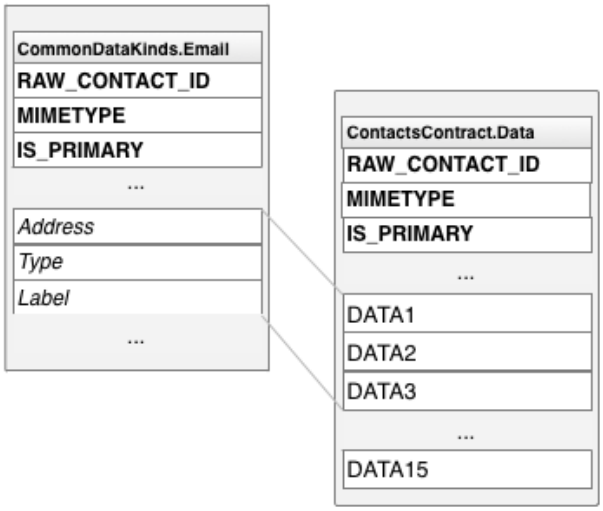

Figura 9 - Ejemplo de columnas descriptivas

[4] Un MIME es una etiqueta utilizada para identificar un tipo de dato para saber cómo manejarlo.

6.1.3 Tabla de contactos (Contacts)

En esta tabla se encuentran la combinación de las filas de contactos sin procesar de todos los tipos y nombres de cuentas. Esto facilita la gestión interna de los contactos, ya sea para visualizar un contacto o modificarlo, ya que es posible la creación de nuevas filas de contactos sin procesar en un contacto ya existente.

Para llevar a cabo este propósito, el proveedor de contactos vincula el identificador de esta tabla con la columna CONTACT_ID de los contactos sin procesar.

Las columnas más destacables son:

- **_ID:** Identificador único.
- **LOOKUP_KEY:** Enlace permanente con la fila del contacto ante posibles modificaciones.
- **NAME_RAW_CONTACT_ID:** El ID del contacto sin procesar que contiene el nombre que se va a mostrar del contacto.
- **DISPLAY_NAME_PRIMARY:** El nombre del contacto proporcionado por la columna anterior.
- **PHOTO_ID:** Referencia a la fila que contiene la foto en la tabla Data.
- **PHOTO_URI:** Utilizado para recuperar la foto a tamaño completo del contacto.
- **PHOTO_THUMBNAIL_URI:** Utilizado para recuperar la miniatura de la foto del contacto.
- **HAS_PHONE_NUMBER:** Indica si el usuario contiene al menos un número de teléfono.
- **STARRED:** Si el contacto pertenece a los contactos favoritos.

Por último, en la figura 10 se puede observar un ejemplo de la relación entre estas tres tablas.

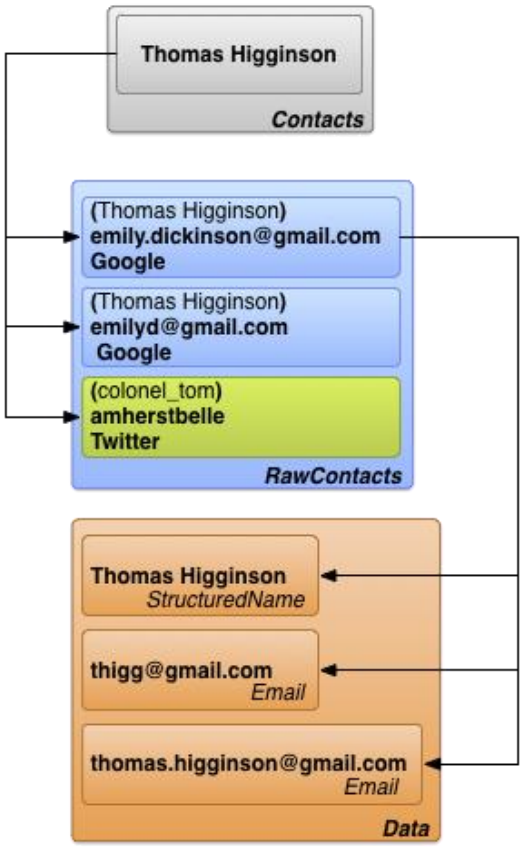

Figura 10 - Relaciones entre las tres tablas

6.1.4 Tabla del registro de llamadas (CallLog)

Para acceder al contenido del registro de llamadas se accede utilizando el Proveedor de Contenidos [5]. En esta tabla se encuentran todas las llamadas, tanto entrantes y salientes, como perdidas, realizadas a través del dispositivo.

Los campos más destacables son:

- **NUMBER:** El número de teléfono asociado a la llamada.
- **DATE:** La fecha de la llamada en milisegundos desde el Epoch[5].
- **DURATION:** Duración de la llamada en segundos.

---

[5] Epoch hace referencia a una fecha concreta tomada como punto de partida desde la cual los sistemas informáticos llevan la cuenta del tiempo que ha pasado.

- **TYPE:** El tipo de llamada. Puede ser entrante, saliente o perdida, entre otros.
- **CACHED_LOOKUP_URI:** El contacto cacheado asociado al número de teléfono (si existe en la agenda) en formato URI.
- **CACHED_NAME:** El nombre cacheado asociado al número (si existe).
- **CACHED_PHOTO_URI:** El ID de la foto cacheado asociado al número (si existe).

6.2 Módulo launcher

En esta sección se describe la estructura utilizada para el módulo launcher. El propósito de este módulo es mostrar la lista completa de aplicaciones por un lado y, por otro, tener una lista de aplicaciones favoritas a modo de "acceso directo". Es por ello por lo que no es necesario una implementación de una base de datos como tal.

En este caso se va a utilizar un mapa, donde se almacenarán pares clave-valor, definidos a continuación:

- **KEY:** Identificador único de la aplicación.
- **VALUE:** Posición que ocupa la aplicación en la lista de accesos directos.

La clave se va a obtener mediante la concatenación el paquete de la aplicación con su nombre.

El valor tiene una lógica más compleja, puesto que, o bien almacena la posición que ocupa en la lista, o bien contiene el valor -1. Esto se hace para poder mantener un orden a la vez que se simplifica la estructura de datos. De esta forma, cuando el usuario añade una aplicación nueva a los accesos directos, sólo es necesario consultar el número de aplicaciones con acceso directo, para después almacenarla con ese valor, e incrementarlo en uno.

Para asegurar la persistencia de nuestro modelo de datos, haremos uso de la API SharedPreferences [7], que utilizaremos a través de una interfaz. Esta herramienta proporciona garantías de consistencia altas, por lo que no existirán pérdidas de datos.

6.3 Módulo medicamentos

En esta sección se describen las tablas utilizadas para el módulo medicamentos. Por un lado, existirá la tabla relacionada con la información de un medicamento y por otro, los recordatorios que se van a generar en base a los medicamentos.

6.3.1 Tabla medicamentos

Contiene la información de los medicamentos y está formada por los siguientes campos:

- **id:** Identificador único.
- **name:** Nombre del medicamento.
- **description:** Breve descripción del medicamento.
- **weekDays:** Mapa con la clave día de la semana y valor booleano que indica si dicho día lo toma.
- **hours:** Lista de horas a las que toma el medicamento en el día.
- **unitsPerDosis:** Número de unidades que tiene que tomar en cada dosis.
- **startDate:** Fecha de comienzo del tratamiento.
- **endDate:** Fecha de fin del tratamiento.
- **photo:** String con el identificador de la fotografía del medicamento.
- **notifications:** Valor booleano que indica si tiene o no activados los recordatorios para este medicamento.
- **color:** Número entero que representa el identificador del color.

6.3.2 Tabla notificaciones

Contiene la información de las notificaciones y está formada por los siguientes campos:

- **id:** Identificador único.
- **hour:** Hora establecida para la notificación.
- **medicationId:** Id del medicamento asociado a esta notificación.
- **weekDay:** Enumerado con el día de la semana correspondiente a la medicación.

- **startDate:** Fecha de comienzo del tratamiento.
- **endDate:** Fecha de fin del tratamiento.
- **medicationName:** Nombre del medicamento.
- **medicationColor:** Número entero que representa el identificador del color.
- **completed:** Booleano que indica si el usuario ha tomado el medicamento asociado a esta notificación.

Capítulo 7 - Arquitectura

En este capítulo se describe la arquitectura del sistema, explicando de forma detallada los diferentes patrones de diseño y arquitectónicos aplicados para el desarrollo de los componentes software.

7.1 Patrones arquitectónicos

Los patrones arquitectónicos describen de forma abstracta la solución a un problema común en el diseño de componentes software. A diferencia de los patrones de diseño, los arquitectónicos intentan dar solución a nivel de sistema y/o componentes de este.

7.1.1 MVVM (Model-View-ViewModel)

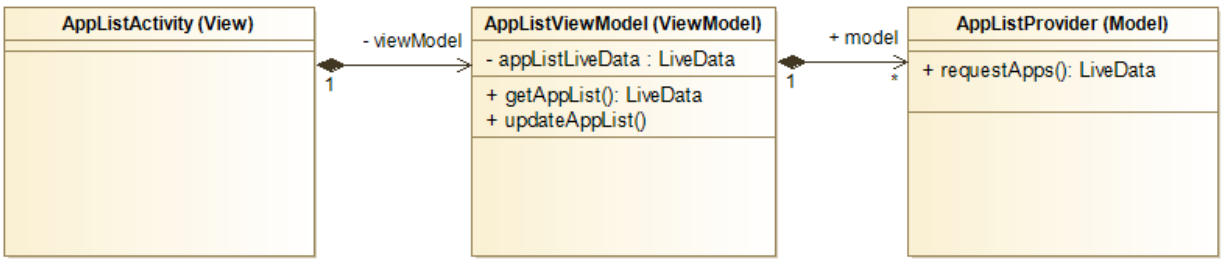

Figura 11 - Aplicación MVVM en la lista de apps

El patrón Model-View-ViewModel [17] (traducido al castellano como *Modelo-Vista-Modelo de vista*), sirve para desacoplar la interfaz del usuario, es decir, la vista (Activity o Fragment en el SDK de Android), de la lógica de la aplicación.

Las capas que lo componen son las siguientes:

- **Capa del modelo:** Es la responsable de la abstracción de las diferentes fuentes de datos. Trabaja junto con el modelo de la vista para guardar y/o recuperar datos.
- **Capa de la vista:** Se encarga de informar al modelo de la vista de las acciones del usuario. No contiene ninguna lógica y simplemente espera los cambios mediante un observador.

- **Capa del modelo de la vista:** Su principal función es hacer de enlace entre la capa del modelo y de la vista. Si el usuario interactúa con la interfaz, se encarga de transmitirlo al modelo y, cuando el modelo se actualiza, se lo notifica a la vista.

Generalmente se utiliza haciendo uso de la librería DataBinding [20] o ViewBinding [19]. En el desarrollo de este trabajo se ha elegido la segunda opción.

Este patrón ha sido aplicado en el desarrollo del módulo launcher y medicamentos. En la Figura 11 se pueden observar las tres clases que lo componen para mostrar la lista de aplicaciones instaladas.

7.1.2 Router interface

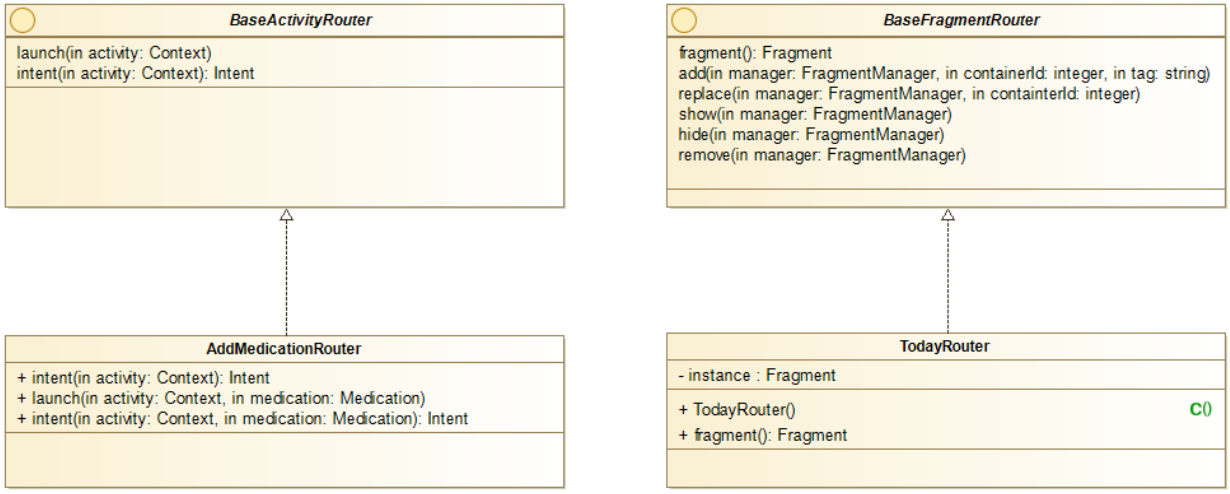

Figura 12 - Aplicación Router en añadir medicación y ver medicamentos de hoy

Router es un patrón utilizado para gestionar las vistas del proyecto. Haciendo uso de interfaces, se va a poder realizar el lanzamiento de nuevas vistas o la interacción con ellas. Para este propósito, se van a definir dos interfaces distintas:

- **BaseFragmentRouter:** Con esta interfaz se va a controlar que solo haya una única instancia del fragmento. Además, se podrá añadir el fragment a un contenedor, esconderlo o eliminarlo.
- **BaseActivityRouter:** Con esta interfaz se va a realizar el lanzamiento de una nueva activity, pudiendo añadir parámetros necesarios.

Este patrón se utiliza junto al anterior y ha sido utilizado en el módulo medicamentos. Ha sido desarrollado en base a un proyecto de una aplicación llamada Twitimer [18], disponible en GitHub. En la Figura 12 se puede observar su aplicación en el activity para añadir una nueva medicación y en el fragment para visualizar las medicinas de hoy.

7.2 Patrones de diseño

Los patrones de diseño son soluciones reutilizables y aplicables a diferentes problemas comunes del sistema, que sirven para evitar errores y ahorrar tiempo de desarrollo.

7.2.1 ViewBinding

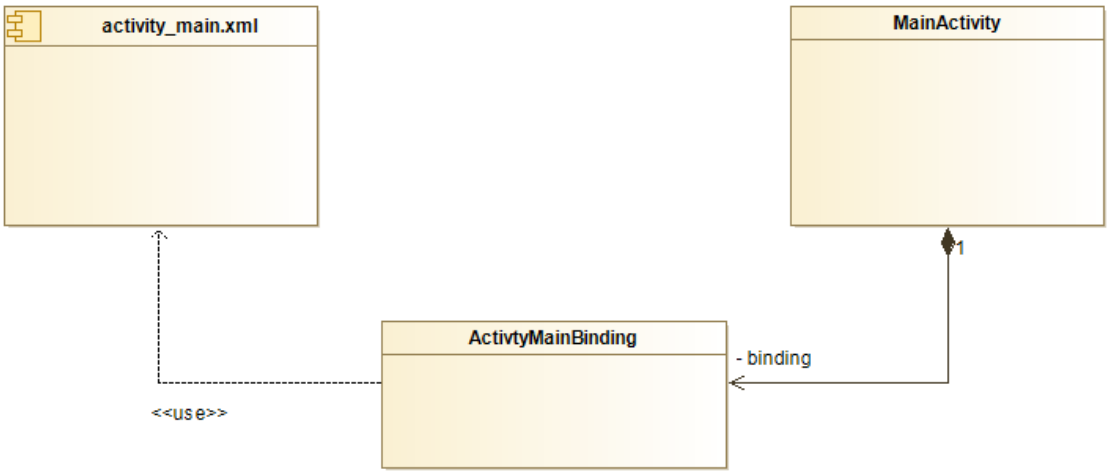

Figura 13 - ViewBinding en la vista principal del launcher

ViewBinding [19] (o vinculación de vista en castellano), es una función que permite vincular un fichero de vista con el activity o fragment que va a hacer uso de esta. Es una implementación más simple y con un mejor rendimiento que la librería DataBinding [20], pero con funciones más limitadas que son suficientes para el uso que se va a hacer de ellas.

Para ello, Android genera automáticamente una clase Binding a partir del fichero de vista XML. Después, sólo se instancia esa clase desde un activity o fragment para acceder a los componentes.

La principal ventaja es que reemplaza al método encargado de vincular la vista utilizando un identificador, lo que elimina la posibilidad de no encontrar el componente y retorne nulo. Además, aumenta la simplicidad del código, puesto que los identificadores que se le asignan a un componente tienen que ser únicos en ese fichero de vista, no en todo el proyecto. Con esto, se evita la utilización de nombres complejos y se reducen las posibilidades de errores.

La vinculación de vista ha sido aplicada en el módulo launcher y medicamentos. En la Figura 13 se puede observar su aplicación en la vista principal del launcher.

7.2.2 Repository

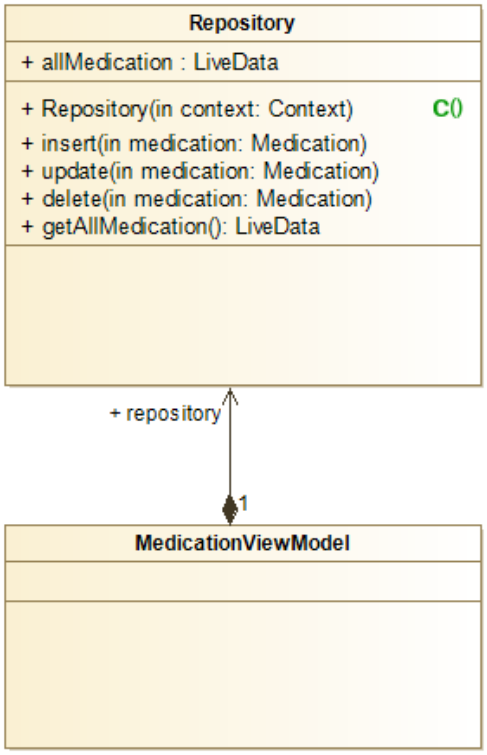

Figura 14 - Repository en el modelo de la vista de los medicamentos

El patrón Repository [21] es una abstracción utilizada para ocultar todas las posibles fuentes de datos de la aplicación. Ha sido utilizada en el módulo medicamentos para acceder únicamente a la base de datos local, mediante la librería Room [8].

La clase ViewModel [9], será la que contenga una instancia del Repositorio, que contendrá los métodos relativos a consultas de base de datos, como insertar o eliminar.

Estos métodos, llaman clases estáticas anidadas que realizan dichas operaciones en segundo plano, para evitar bloquear la vista.

Se ha utilizado en el módulo de los medicamentos. En la Figura 14 se puede observar su aplicación sobre el modelo de vista de la lista de medicamentos.

7.2.3 Singleton

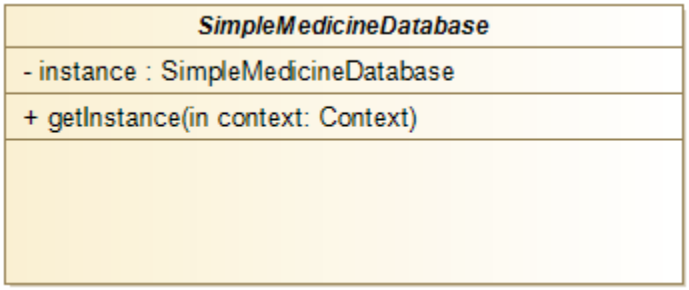

Figura 15 - Singleton en el acceso a la base de datos de los medicamentos

Singleton [22] se encarga restringir la creación de objetos pertenecientes a una clase para garantizar que sólo se crea una única instancia de esta. Esto permite que haya un único punto de acceso global.

Este patrón ha sido utilizado en el módulo medicamentos, para tener una única instancia de la base de datos, aunque también se ha aplicado en la creación de fragments con el patrón Router. En la Figura 15 se puede observar su aplicación para el acceso a la base de datos.

7.2.4 DAO (Data Access Object)

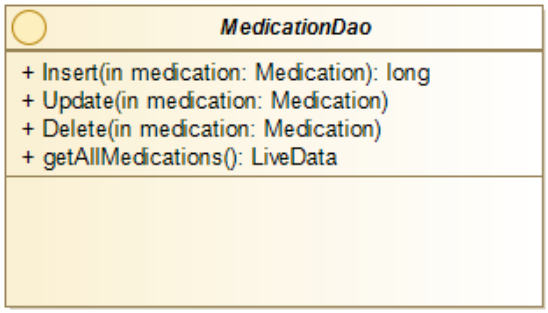

Figura 16 - Aplicación DAO tabla medicamentos

Un DAO [23][24] o Data Access Object (Objecto de Acceso a Datos en castellano), es un componente que proporciona una interfaz común entre la aplicación y los dispositivos de almacenamiento de datos.

Este patrón ha sido utilizado en el módulo medicamentos junto a la librería Room [8][24], de tal forma que en los métodos del DAO se añaden anotaciones con la consulta a la base de datos. En la Figura 16 se puede observar la aplicación sobre la tabla de medicamentos.

Capítulo 8 - Diseño

En este capítulo se van a describir los principales aspectos acerca de la funcionalidad y el diseño realizado para el desarrollo de cada módulo funcional del proyecto.

8.1 Módulo teléfono

En este primer módulo, se van a comentar los colores elegidos, la gestión de los permisos, las diferentes vistas de la aplicación y como se obtienen los datos del Proveedor de Contenidos [5].

8.1.1 Colores

Para el desarrollo de la paleta de colores de este módulo se ha utilizado la web de Material.io [25], donde se ha decidido escoger un color oscuro como color principal y un color claro más vistoso como secundario. En este caso, se ha optado por una tonalidad azul oscuro y otra naranja, con sus respectivas variantes. En la Figura 17 se puede observar el archivo de configuración con los colores utilizados.

```
<color name="primaryColor">#311b92</color>
<color name="primaryLightColor">#6746c3</color>
<color name="primaryDarkColor">#000063</color>
<color name="secondaryColor">#ffa000</color>
<color name="secondaryLightColor">#ffd149</color>
<color name="secondaryDarkColor">#c67100</color>
<color name="primaryTextColor">#ffffff</color>
<color name="secondaryTextColor">#000000</color>
```

Figura 17 - Archivo de configuración donde se guardan los colores del módulo

8.1.2 Gestión de permisos

Para el desarrollo de este módulo, se van a necesitar solicitar ciertos permisos al usuario. Los permisos de Android sirven para proteger la privacidad del usuario. A través de ellos se puede acceder a datos o acciones restringidas.

Los permisos se van a utilizar, deben ser declarados en el archivo *Android Manifest* como se observa en la Figura 18, que describe la información esencial de la aplicación.

```
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Figura 18 - Declaración de permisos en Android Manifest

Una vez declarados los permisos, hay que asegurarse de que el usuario da el visto bueno para que el sistema pueda acceder a la funcionalidad que se necesita. Sin embargo, una vez iniciada la aplicación por primera vez, no sería una buena práctica solicitarlos de golpe, por lo que es recomendable hacerlo cuando se vayan a necesitar.

En primer lugar, los permisos esenciales que se van a solicitar son la realización de llamadas, el acceso de lectura y escritura de contactos, acceso al registro de llamadas y estado del teléfono. Sin estos permisos, no se podrían realizar llamadas o mostrar los contactos almacenados, por lo que, si el usuario los rechaza, la aplicación no tendría funcionalidad.

El resto de los permisos se van a solicitar cuando se quiera añadir o actualizar la foto de un contacto, ya que se necesitará acceso a la cámara o al almacenamiento interno. En la Figura 19 se observa la solicitud de permisos nada más iniciar la aplicación.

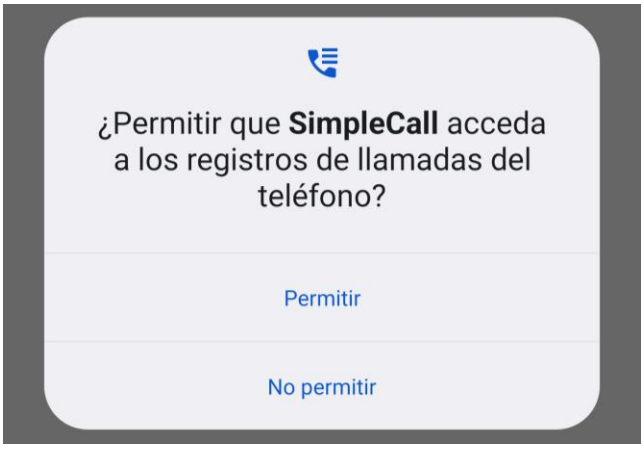

Figura 19 - Solicitud de permisos al iniciar la aplicación

8.1.3 Vista principal

La vista principal de la aplicación se compone de un activity en la que se encuentran distintos elementos. En primer lugar, se hace uso de una clase *TabLayout*, un componente que permite añadir pestañas a la vista. Cada pestaña cuenta con un título y un emoticono representativo, como se observa en la Figura 20. Las pestañas disponibles son: favoritos, agenda y registro de llamadas.

Figura 20 - Pestañas disponibles en el módulo teléfono

Una vez definidas las pestañas, es necesario hacer uso de *ViewPager2*, que permite la utilización de distintos fragments en formato deslizable. Además, se necesitará una clase *Adapter* que le proporcione el fragment que debe mostrar en cada momento. En la Figura 21 se observa el método que devuelve el fragment asociado a una posición.

```
@Override
public Fragment createFragment(int position) {

    switch (position) {
        case 0:
            this.favContactListFragment = new FavContactListFragment(favContactList, selectedFavContact);
            return favContactListFragment;
        case 1:
            this.contactListFragment = new ContactListFragment(contactModelList, selectedContact, mainActivity);
            return contactListFragment;
        case 2:
            this.callLogListFragment = new CallLogListFragment(callModelList, phoneContactMap, selectedCall);
            return callLogListFragment;
        default:
            return null;
    }

}
```

Figura 21 - Método del Adapter que devuelve el fragment según una posición

Por último, un elemento que se va a encontrar en todas las pestañas va a ser el botón flotante del marcador telefónico, ubicado en la esquina derecha de la parte inferior de la pantalla. El diseño del botón se puede observar en la Figura 22.

Figura 22 - Botón para acceder al marcador telefónico

8.1.4 Pestaña de agenda

En esta pestaña se van a mostrar todos los contactos almacenados en la base de datos de Android proporcionada por el Proveedor de Contactos [6]. La lista se muestra con una orientación vertical, con un único elemento por cada fila como se observa en la Figura 23. De cada contacto, se va a mostrar una foto y un nombre, priorizando un diseño minimalista para evitar que el usuario se pierda. Para esta implementación se hace uso de un *RecyclerView*, un componente que permite mostrar una lista de elementos del mismo tipo y que requerirá de un *Adapter* para insertar cada elemento en la interfaz.

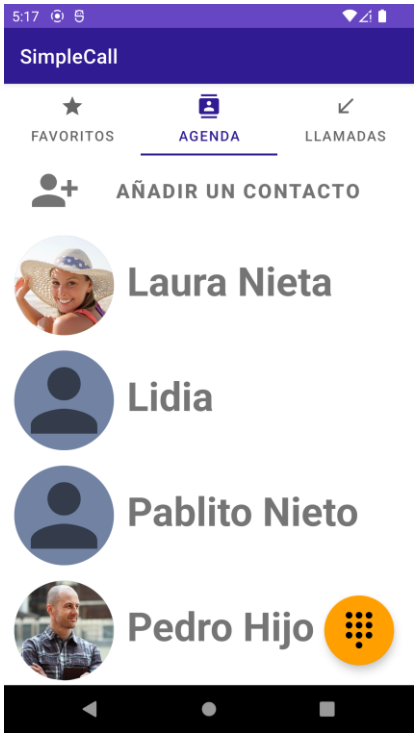

Figura 23 - Vista de la agenda de contactos

Para acceder a la información detallada del contacto o realizar una modificación de este, únicamente se debe pulsar sobre el mismo. Además, en la parte

superior de la misma hay un botón para añadir nuevos contactos que proporciona una interfaz similar. La vista relacionada con estos apartados será explicada más adelante.

8.1.5 Pestaña de favoritos

En esta pestaña se van a mostrar todos los contactos favoritos almacenados en la base de datos de Android proporcionada por el Proveedor de Contactos [6]. A diferencia de la lista anterior, esta tendrá formato de cuadrícula, con dos elementos por fila como se observa en la Figura 24 También se va a dar prioridad a la fotografía del contacto, por lo que aparecerá en un mayor tamaño y el nombre aparecerá más pequeño debajo.

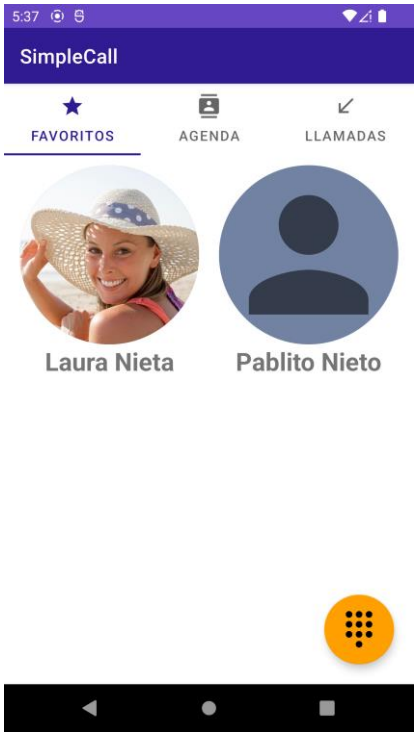

Figura 24 - Vista de los contactos favoritos

Al igual que en la pestaña anterior, la interacción se hará pulsando sobre el contacto deseado, pero en esta ocasión, la aplicación realizará una llamada telefónica directamente.

8.1.6 Pestaña de registro de llamadas

En esta pestaña se va a mostrar el registro de llamadas utilizando el Proveedor de Contenidos [5]. La lista tendrá una orientación vertical, con una llamada por fila como se observa en la Figura 25. Para cada llamada, se mostrará la foto del contacto, si dicho contacto existe en la agenda y tiene una fotografía, el nombre, un icono que indica si la llamada es entrante, saliente o perdida, y hace cuanto ha ocurrido.

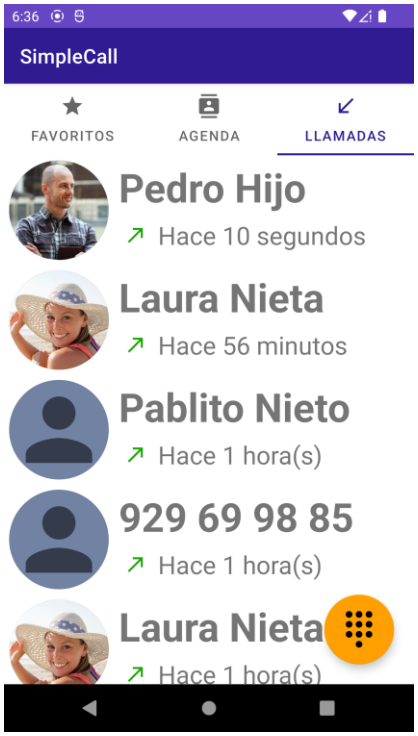

Figura 25 - Vista del registro de llamadas

Al igual que en las anteriores pestañas, si se realiza una pulsación sobre una llamada, se realizará una rellamada.

8.1.7 Marcador telefónico

Al marcador telefónico se accede mediante el botón anaranjado mencionado previamente. La pantalla que se muestra es un nuevo *activity* donde se encuentra un teclado numérico sencillo con las teclas del mismo color de la aplicación. Para simplificar, se ha optado por diseñar un teclado en vez de usar el teclado numérico del sistema. También se incluye un botón para eliminar los números introducidos y un botón para realizar la llamada. Si se desea volver atrás, sólo hay que pulsar la flecha de la barra

superior o pulsar el botón "atrás" del propio sistema, como se puede observar en la Figura 26.

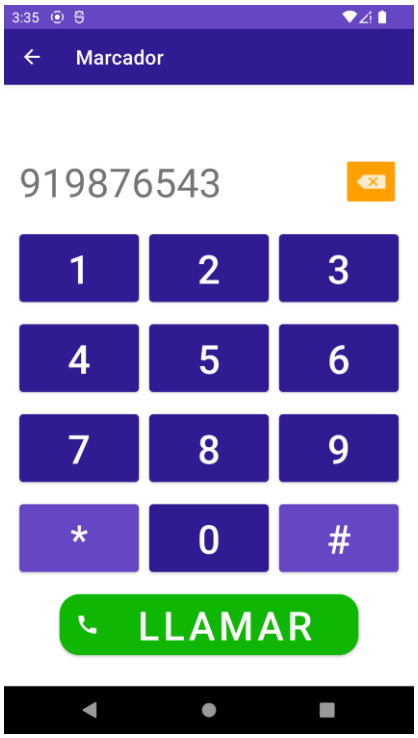

Figura 26 - Vista del marcador telefónico

Para realizar la llamada, se realizará un *Intent*, que es un mecanismo de Android para lanzar nuevos *activities*. En este caso, no se necesita un *activity* propio, puesto que se llamará a uno de Android pasando el número introducido como argumento. Este método se aplica tanto en el marcador como en la llamada a un contacto y se puede observar su implementación en la Figura 27.

Por último, no se sabrá si el número introducido es válido o no hasta que el proveedor de telefonía dé una respuesta.

```
// CALL BUTTON
bCall.setOnClickListener(v -> {
    callDone = true;
    startActivity(new Intent(Intent.ACTION_CALL, Uri.parse("tel:" + tpPhone.getText())));
    tpPhone.setText("");
});
```

Figura 27 - Realización de una llamada desde el marcador

8.1.8 Visualizar detalles y editar un contacto

Para visualizar la información de un contacto, será necesario acceder a la agenda de contactos y pulsar sobre el contacto que se quiere visualizar. Se abrirá un nuevo *activity* con la información de este. En este caso se mostrará la foto de perfil, nombre, número de teléfono y una estrella amarilla si está guardado como contacto favorito o una estrella transparente con un borde gris en otro caso.

Como se observa en la Figura 28, aparte de los campos mencionados, aparece un botón flotante en la esquina inferior derecha que permite realizar una edición del contacto. Si se pulsa sobre este botón, se accede a una vista similar donde se puede modificar todos los campos, desde la foto, el nombre o el número, hasta si es favorito. Además, para cambiar la foto existen diferentes opciones: eliminar la foto si dispone de una, elegir una foto de la galería o realizar una foto desde la cámara.

Una vez que se haya terminado la edición, para guardar los cambios simplemente se debe pulsar el botón flotante verde y para descartar los cambios se debe pulsar el botón de volver atrás. Por último, si se quiere eliminar el contacto de la base de datos, se debe pulsar el botón rojo en la misma vista de edición.

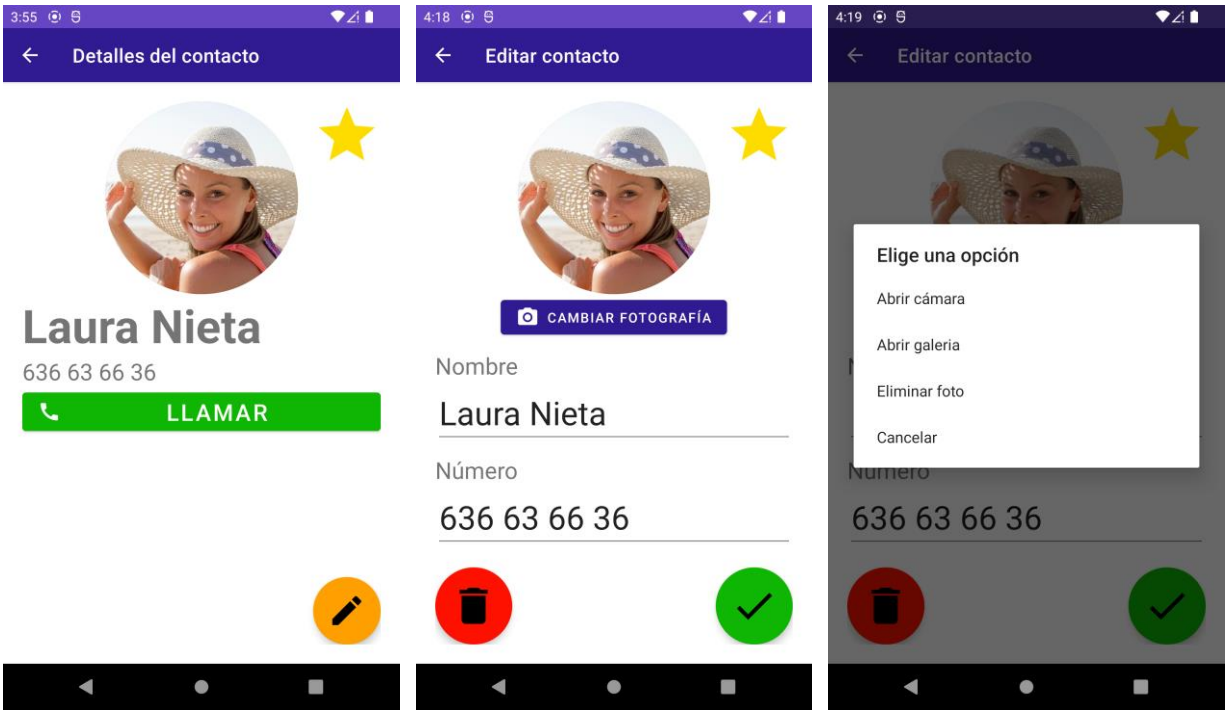

Figura 28 - Vista de los detalles y la edición de un contacto

8.1.9 Añadir un contacto

La vista para añadir un contacto guarda muchas similitudes con la de editar. La principal diferencia es que los campos aparecen vacíos, el botón de eliminar no aparece y el botón para añadir a favoritos tampoco, como se observa en la Figura 29.

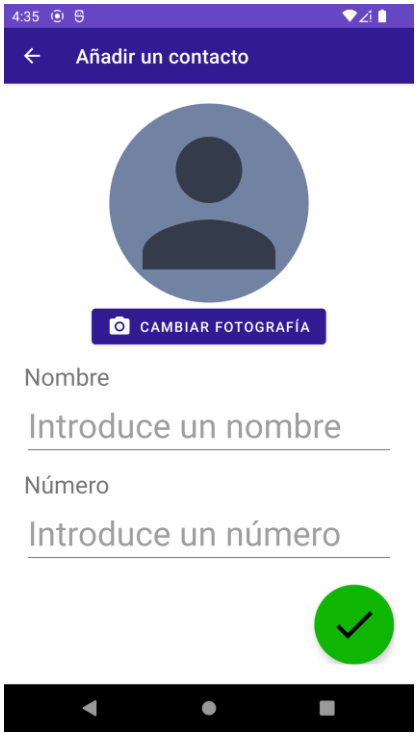

Figura 29 - Vista para añadir un contacto

8.1.10 Acceso y modificación de datos

Para realizar el acceso y la modificación de los datos, se van a utilizar los Proveedores de Contenido [5]. En concreto, se utilizará el proveedor de contactos, contenido en el propio sistema operativo. El acceso se realizará haciendo uso de un objeto *Context*, una interfaz que permite el acceso a determinados recursos y clases de la aplicación, como es el caso.

El patrón para acceder a un *ContentProvider* desde una IU se realiza mediante un *CursorLoader*, que realiza una consulta similar a SQL en segundo plano. Mediante el *ContentResolver*, se accederá al *ContentProvider* o proveedor de contenido deseado. En la Figura 30 se muestra una representación de esta comunicación.

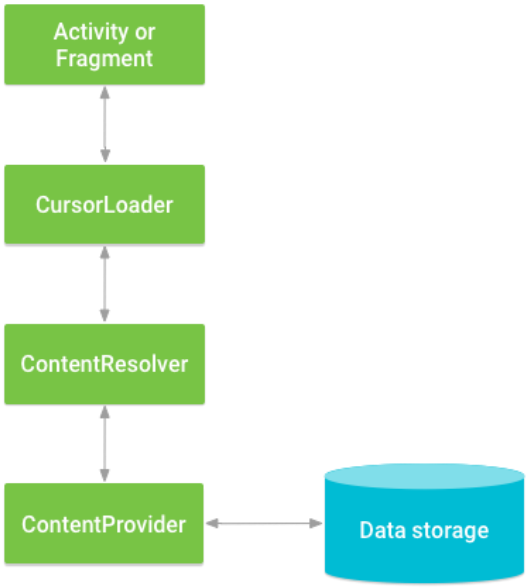

Figura 30 - Acceso a Proveedor de Contenido

Este patrón se aplica tanto en la recuperación de contactos, como del registro de llamadas. En la Figura 31 se muestra el método que carga todas las llamadas registradas en el sistema y las añade en una lista. Todos los métodos de recuperación y actualización de datos se encuentran en una clase preparada para ello.

```
public void loadCallLogs(List<CallModel> callModelList){
    // Order by date
    String sortOrder = CallLog.Calls.DATE + " DESC";
    // Query
    Cursor cursor = contentResolver.query(CallLog.Calls.CONTENT_URI,  projection: null,  selection: null,  selectionArgs: null, sortOrder);

    if (cursor.getCount() > 0) {
        while (cursor.moveToNext()) {
            callModelList.add(recoverCallModel(cursor));
        }
        cursor.close();
    }
}
```

Figura 31 - Método que carga el registro de llamadas

La recuperación de los contactos es similar, pero algo más compleja, puesto que hay que recuperar más datos. Además, para distinguir entre contactos favoritos se utilizan dos listas, que se van rellenando en la función de carga de forma apropiada.

8.2 Módulo launcher

En este módulo, se van a comentar las vistas que componen el launcher, su funcionamiento y la gestión de permisos.

8.2.1 Gestión de permisos

En primer lugar, no se necesita pedir ningún permiso al usuario al iniciar el módulo, ya que no impide el correcto funcionamiento. Sin embargo, una de las opciones disponibles será cambiar el fondo de pantalla. Siguiendo las recomendaciones de Android, se solicitará dicho permiso cuando el usuario pulse esa opción.

```
<uses-permission android:name="android.permission.SET_WALLPAPER" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

Figura 32 - Permisos relativos al fondo de pantalla

En segundo lugar, como un launcher es una aplicación que se encarga de lanzar otras aplicaciones, será necesario que el dispositivo lo reconozca como tal. Para ello se debe declarar dentro del *Manifest* que la actividad principal sea un launcher como se indica en la Figura 33, así cuando se realice una pulsación el botón *home* del dispositivo, se devolverá al usuario a esa vista.

```
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.HOME" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Figura 33 - Filtros para establecer el launcher

8.2.2 Vista principal

La vista principal del módulo se compone de un *activity*, donde se muestran las *apps* que el usuario ha añadido. En primer lugar, se encuentra un reloj digital donde el usuario puede consultar la hora y, abajo del todo, un botón para acceder al box de aplicaciones, donde están todas las aplicaciones instaladas.

Para la lista de aplicaciones que tienen acceso directo, se utiliza un *RecyclerView* con su respectivo *adapter*, que mostrará las apps en una lista vertical, con cuatro elementos por fila, como se puede observar en la Figura 34.

Para interactuar con las aplicaciones, el usuario deberá pulsar sobre ellas y se abrirán automáticamente. Además, el fondo de pantalla es el que viene por defecto en el dispositivo, que puede ser cambiado de forma fácil.

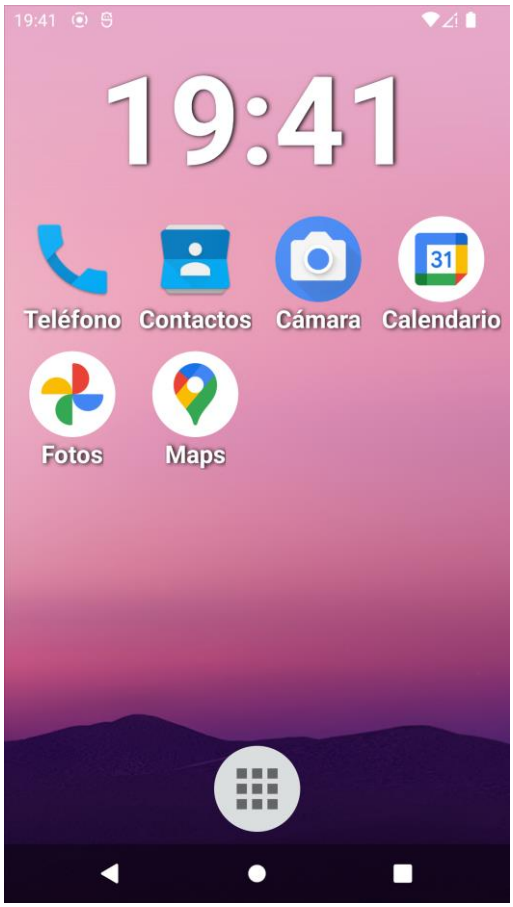

Figura 34 - Vista principal de la aplicación

8.2.3 Vista de aplicaciones

La vista con todas las aplicaciones se compone, nuevamente, de un *RecyclerView*. En este caso, aparecerá una única aplicación por fila, de la cual se mostrará su icono y su nombre, como se observa en la Figura 35. El formato elegido hace un intento por simplificar al máximo lo que se representa en la pantalla, con textos grandes para que el usuario no se confunda.

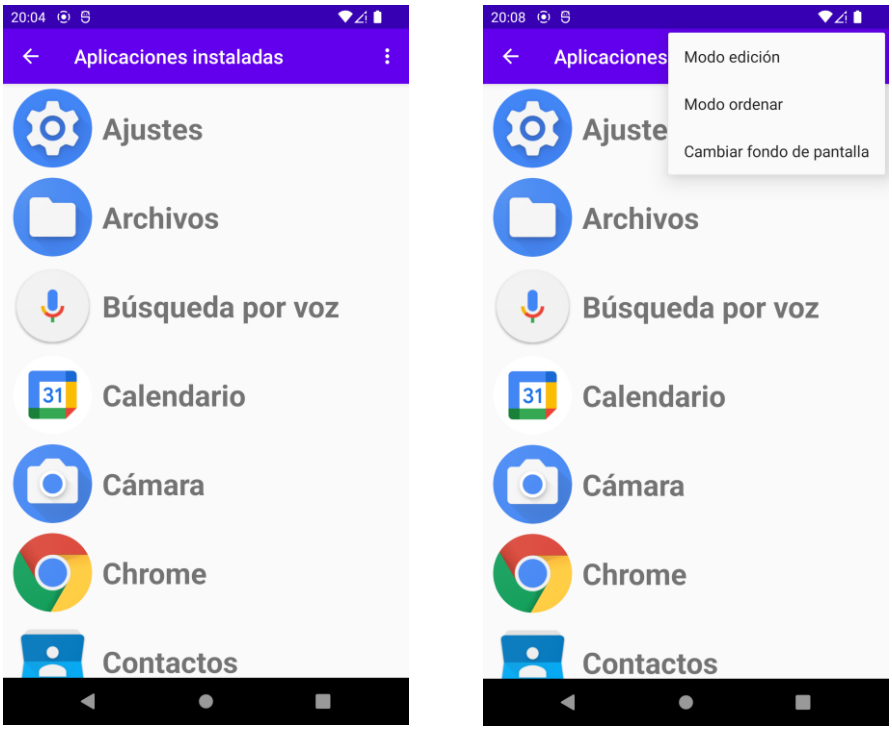

Figura 35 - Vista de aplicaciones instaladas

Desde esta vista, se van a poder modificar los accesos directos de las aplicaciones, así como su orden. Esto se realiza mediante el botón con los tres puntos ubicado en la parte superior de la pantalla y permite que diferentes tipos de usuario puedan hacer uso de la aplicación. Por una parte, los usuarios con menos conocimientos, que normalmente requieren de un familiar o un tercero, podrán utilizar el módulo sin miedo a desconfigurar nada, ya que está lo suficientemente escondido. Por otra parte, los usuarios que no tengan esa dependencia podrán configurar las aplicaciones que más usen a su antojo.

Al pulsar sobre dicho botón, se desplegará un menú con tres opciones:

- La primera opción hará aparecer un *checkbox* en cada aplicación. Si aparece marcado significa que la aplicación se encuentra como acceso directo. Pulsando sobre estos cuadrados, el usuario podrá añadir y quitar aplicaciones.
- La segunda opción permitirá ordenar las aplicaciones que están como acceso directo mediante dos botones, uno de subir y otro de bajar. Si el

usuario pulsa el de subir, se intercambiará con la aplicación que se encuentre por encima suya, y si se pulsa el de bajar, con la de abajo.

- La tercera y última opción abrirá la galería del usuario donde podrá seleccionar el fondo de pantalla que desee.

En la Figura 36 pueden observarse estas opciones.

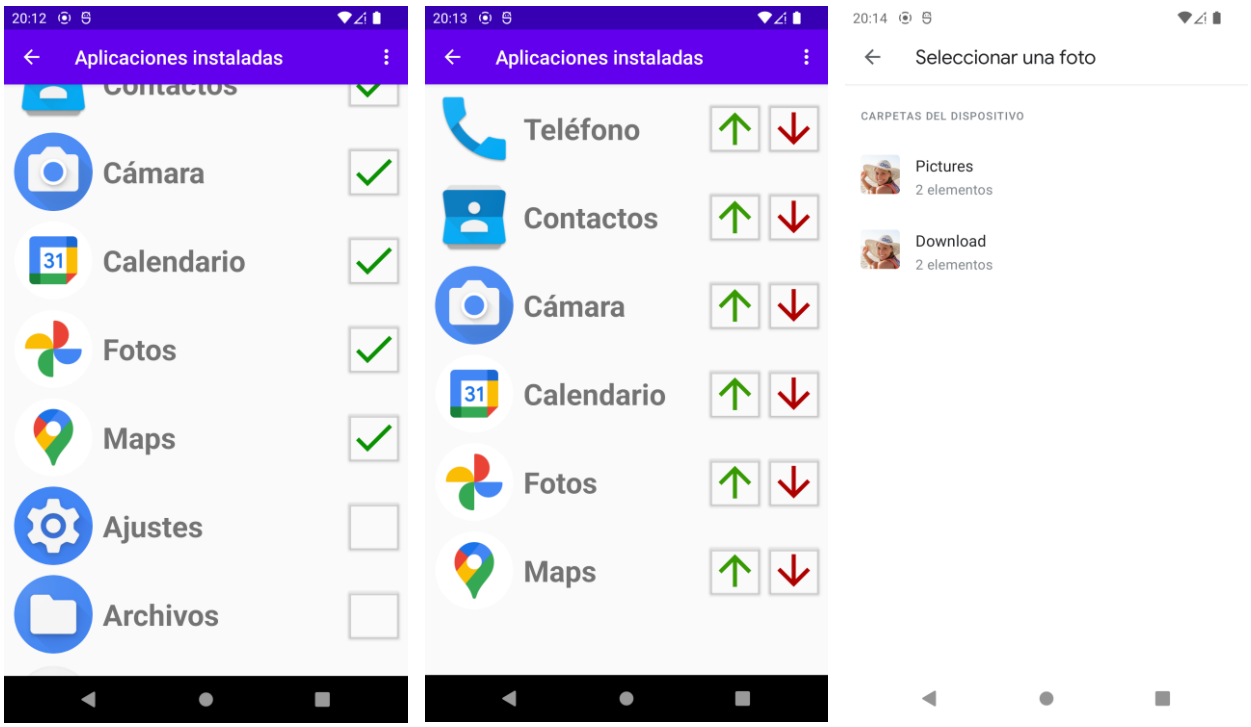

Figura 36 - Diferentes opciones en la vista de aplicaciones

8.2.4 Acceso a las aplicaciones instaladas

Para recuperar las aplicaciones instaladas en nuestro sistema, se utilizará la clase *PackageManager*, que proporciona información relacionada a los paquetes instalados en el dispositivo.

El método encargado de recuperar las aplicaciones creará una nueva lista con Objetos de tipo *AppModel*, una clase propia que contiene el nombre de la app, el nombre del paquete, el icono y su posición en el menú principal (-1 si no tiene acceso directo). Después, creará un objeto *PackageManager* y se realizará una consulta que devolverá las aplicaciones instaladas.

Por último, solo quedará ir recorriendo la lista y comprobando, mediante la API de SharedPreferences [7], las aplicaciones que tienen acceso directo. Al final sólo se ordenará la lista en función del tipo de orden (alfabético o según si tiene acceso directo o no). En la Figura 37 se observa la implementación.

```
public static List<AppModel> fetchApps(OrderTypeAppModel orderType){
    List<AppModel> appList = new ArrayList<>();
    PackageManager packageManager = context.getPackageManager();
    Intent intent = new Intent(Intent.ACTION_MAIN,  uri: null);
    intent.addCategory(Intent.CATEGORY_LAUNCHER);
    List<ResolveInfo> allApps = packageManager.queryIntentActivities(intent,  flags: 0);
    SharedPreferences myPreferences = PreferenceManager.getDefaultSharedPreferences(context);

    for (ResolveInfo ri : allApps) {
        String label = ri.loadLabel(packageManager).toString();
        String packageName = ri.activityInfo.packageName;
        Drawable icon = ri.activityInfo.loadIcon(packageManager);

        AppModel app = new AppModel(label, packageName, icon,  position: -1);
        app.setPosition(myPreferences.getInt(app.getID(),  defValue: -1));

        appList.add(app);
    }

    if(orderType == OrderTypeAppModel.ORDER_BY_SHORTCUT_AND_NAME)
        Collections.sort(appList, new AppComparatorByShortcutAndName());
    else if(orderType == OrderTypeAppModel.ORDER_BY_NAME)
        Collections.sort(appList, new AppComparatorByName());

    return appList;
}
```

Figura 37 - Método que devuelve las aplicaciones instaladas

8.3 Módulo medicamentos

En este último módulo, se van a comentar los colores elegidos, las diferentes vistas de la aplicación y como se crean las notificaciones *push*.

8.3.1 Colores

Para el desarrollo de la paleta de colores de este módulo se ha utilizado la web de Material.io [25], eligiendo de color principal un tono de verde, que suele tener un efecto positivo sobre la salud, y de secundario uno naranja. El fichero de configuración relacionado puede observarse en la Figura 38.

```xml
<color name="primary">#4CAF50</color>
<color name="primary_dark">#388E3C</color>
<color name="primary_light">#C8E6C9</color>
<color name="secondary">#FF5722</color>
<color name="secondary_dark">#AC4B2D</color>
<color name="secondary_light">#FDC7B5</color>
<color name="primary_text">#212121</color>
<color name="secondary_text">#757575</color>
```

Figura 38 - Archivo de configuración donde se guardan los colores del módulo

8.3.2 Vista principal

La vista principal de la aplicación se compone de un *activity* en la que se encuentran distintos elementos. En esta ocasión, se ha optado por utilizar un *BottomNavigationView* para separar las distintas pestañas de la vista. Este componente aparece al final de la pantalla y permite al navegación entre *fragments*. Cada botón cuenta con un icono y nombre representativo, como se observa en la Figura 39.

Figura 39 - Menú de botones en el módulo medicamentos

Cuando se realiza la carga del *activity*, uno de los métodos que se va a ejecutar es la carga de las pestañas. Gracias al patrón Router, se va a controlar cuando se crean, muestran y esconden los *fragments* de una forma eficiente, puesto que sólo existirá una única instancia de cada uno. Esto es importante, ya que, si se crease un *fragment* cada vez que se pulsa un botón, afectaría directamente al rendimiento. En la Figura 40 se observa la implementación de la carga de pestañas.

```
private void loadTabs() {
    if(medicationFragment != null)
        medicationFragment.remove(getSupportFragmentManager());
    if(todayFragment != null)
        todayFragment.remove(getSupportFragmentManager());

    medicationFragment = new MedicationRouter();
    todayFragment = new TodayRouter();

    binding.homeBottomNavigationView.setOnItemSelectedListener(item -> {
        if (selectedItem != item.getItemId()) {
            selectedItem = item.getItemId();

            if (item.getItemId() == R.id.home_menu_medication) {
                todayFragment.hide(getSupportFragmentManager());
                if (getSupportFragmentManager().findFragmentByTag(String.valueOf(R.id.home_menu_medication)) == null) {
                    medicationFragment.add(getSupportFragmentManager(), R.id.homeContainer, String.valueOf(R.id.home_menu_medication));
                }
                medicationFragment.show(getSupportFragmentManager());
            } else if (item.getItemId() == R.id.home_menu_today) {
                medicationFragment.hide(getSupportFragmentManager());
                if (getSupportFragmentManager().findFragmentByTag(String.valueOf(R.id.home_menu_today)) == null) {
                    todayFragment.add(getSupportFragmentManager(), R.id.homeContainer, String.valueOf(R.id.home_menu_today));
                }
                todayFragment.show(getSupportFragmentManager());
            }
            return true;
        }
        return false;
    });
}
```

Figura 40 - Método que carga los fragmentos

Por último, un elemento que se va a encontrar en todas las pestañas va a ser el botón flotante para añadir un medicamento, ubicado en la esquina derecha de la parte inferior de la pantalla. El diseño del botón se puede observar en la Figura 41.

Figura 41 - Botón para añadir un nuevo medicamento

8.3.3 Pestaña de la lista de medicamentos

En esta pestaña se van a mostrar los medicamentos que el usuario ha ido añadiendo a la aplicación. Para ello, se hace uso del componente *RecyclerView*, que mostrará un medicamento por cada fila. De cada medicamento, tendremos en primer lugar una barra de color asociado al mismo, que permite distinguirlos entre los demás. También se mostrará la foto asociada al mismo (una por defecto si no tiene), su nombre, el número de pastillas y de tomas al día y hasta que fecha tiene que tomarlo, como se

observa en la Figura 42. Además, si el usuario pulsa sobre cualquier elemento de la lista, podremos acceder a los detalles del medicamento y realizar una edición.

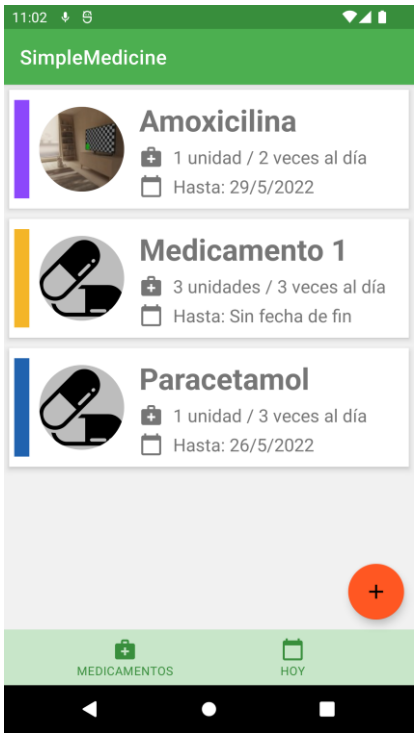

Figura 42 - Vista de la lista de medicamentos

Haciendo uso de un *Adapter*, se cargarán desde la base de datos los medicamentos y se irán rellenando los campos directamente. La implementación se puede observar en la Figura 43.

```
public void bindView(Medication medication) {
    if(medication.getPhoto() != null) {
        itemMedicationListBinding.image.setImageURI(Uri.parse(medication.getPhoto()));
    }
    else {
        itemMedicationListBinding.image.setImageResource(R.drawable.default_pill);
    }
    itemMedicationListBinding.name.setText(medication.getName());
    itemMedicationListBinding.unidades.setText(medication.getUnitsPerDosisString());
    itemMedicationListBinding.fin.setText(medication.getEndDateString());
    itemMedicationListBinding.constraintLayout.setOnClickListener(view -> new SeeMedicationRouter().launch(view.getContext(), medication));
    itemMedicationListBinding.color.setBackgroundColor(medication.getColor());
}
```

Figura 43 - Método que rellena los datos de una fila del medicamento

8.3.4 Pestaña de los medicamentos de hoy

En esta pestaña, el usuario puede consultar los medicamentos que debe tomar en el día de hoy. Para ello, se hace uso de nuevo de un componente *RecyclerView*,

pero en este caso, cada elemento de la lista consistirá en agrupar los medicamentos que tengan la misma hora de toma. En cada fila se mostrará la hora en la que debe tomar el medicamento, una lista con los nombres de los medicamentos resaltados con su color, un mensaje para saber si se lo ha tomado o no y un botón para confirmar que se lo ha tomado, como se observa en la Figura 44.

Figura 44 - Vista de los medicamentos de hoy

En este caso, la lógica del *Adapter* correspondiente al *RecyclerView* es un poco más compleja. En primer lugar, vamos a tener dos *adapters*, uno para la lista de las notificaciones y otro para la lista de los medicamentos dentro de cada notificación. Como las notificaciones que se recuperan de la base de datos contienen una hora y un medicamento, entre otras cosas, haremos uso de un mapa en el que pondremos como clave una hora y valor una notificación. El método *statusCheckButton* se encarga de dibujar los botones y el texto y el método *updateDatabase* de guardar los cambios en la base de datos cuando se pulsa el botón de confirmación. La implementación se encuentra en la Figura 45.

```
public void bindView(HourModel hour) {
    itemTodayListBinding.hour.setText(hour.toString());
    TodayChildRecyclerViewAdapter recyclerViewAdapter = new TodayChildRecyclerViewAdapter();
    itemTodayListBinding.recyclerView.setLayoutManager(new LinearLayoutManager(itemView.getContext()));
    itemTodayListBinding.recyclerView.setAdapter(recyclerViewAdapter);
    recyclerViewAdapter.updateNotificationList(notificationMap.get(hour));

    List<NotificationModel> list = notificationMap.get(hour);
    if(list != null) {
        for(NotificationModel notificationModel: list)
            statusCheckButton(notificationModel.isCompleted(), hour);
    }

    itemTodayListBinding.checkButton.setOnClickListener(v -> {
        statusCheckButton( completed: true, hour);
        updateDatabase(hour);
    });
}
```

Figura 45 - Método que rellena los datos de una fila de una notificación

8.3.5 Añadir un medicamento

Como se ha mencionado previamente, para añadir un medicamento el usuario deberá pulsar el botón ubicado en la esquina inferior derecha. Una vez realizado ese paso, se abrirá un nuevo *activity* donde podrá introducir los datos del medicamento para guardarlos. Considerando que se guardan demasiados datos, ponerlos todos en una única vista puede confundir al usuario, por lo que se ha optado por dividirlo en cuatro pasos distintos. De esta forma, el usuario puede introducir los datos que se le solicitan sin sentirse agobiado por un número alto de campos.

Para esto, sobre el *acitvity* principal se irán cambiando los *fragments* que se visualizan. Esto se consigue con un *Adapter*, que contendrá una lista de *fragments,* aunque en este caso serán Routers por la utilización de dicho patrón. El *adapter* contendrá además un método para instanciar los fragmentos y otro para saber el número de páginas que mostrar en el *ViewPager*, el componente empleado para cambiar entre *fragments*. Los métodos del *adapter* se pueden observar en la Figura 46.

```java
@NonNull
@Override
public Fragment createFragment(int position) {
    return pages.get(position).fragment();
}

@Override
public int getItemCount() {
    return pages.size();
}
```

Figura 46 - Métodos del adapter

Además, de un *ViewPager*, el *activity* contará con dos botones inferiores para ir navegando entre las diferentes páginas, como se observa en la Figura 47.

Figura 47 - Botones inferiores de navegación

```java
// Viewpager

pageAdapter = new AddMedicationPageAdapter( context: this, viewModel.getFragments());
binding.viewPager.setAdapter(pageAdapter);
binding.viewPager.setUserInputEnabled(false);
binding.viewPager.registerOnPageChangeCallback(onPageSelected(position) -> {
        super.onPageSelected(position);
        selection = position;
        binding.buttonPrev.setText((position == 0)? "Cancelar": "Anterior");
        binding.buttonNext.setText((position == viewModel.getPages() - 1)? "Guardar": "Siguiente");
});
```

Figura 48 - Inicialización del ViewPager en el activity

Ahora se va a realizar un análisis del proceso de rellenado de un medicamento de manera individual.

En la primera pantalla, se solicitará al usuario que introduzca el nombre del medicamento de manera obligatoria. También tendrá como campos opcionales una descripción, la posibilidad de realizar una fotografía con la cámara (con la solicitud de permisos pertinente, como en módulos anteriores) y la elección de un color asociado a ese medicamento, como se observa en la Figura 49.

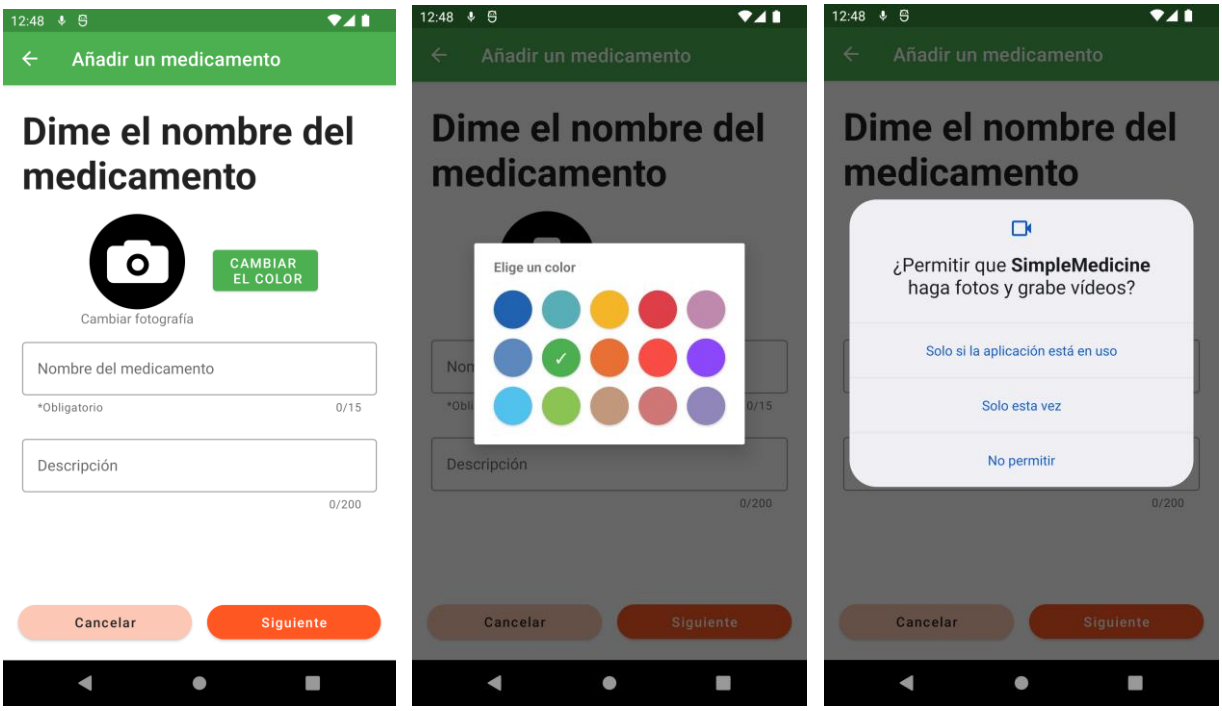

Figura 49 - Añadir un medicamento página 1

En la segunda pantalla, se le mostrará una lista con *checkBoxes* con todos los días de la semana donde el usuario podrá seleccionar individualmente cada día, como se observa en la Figura 50.

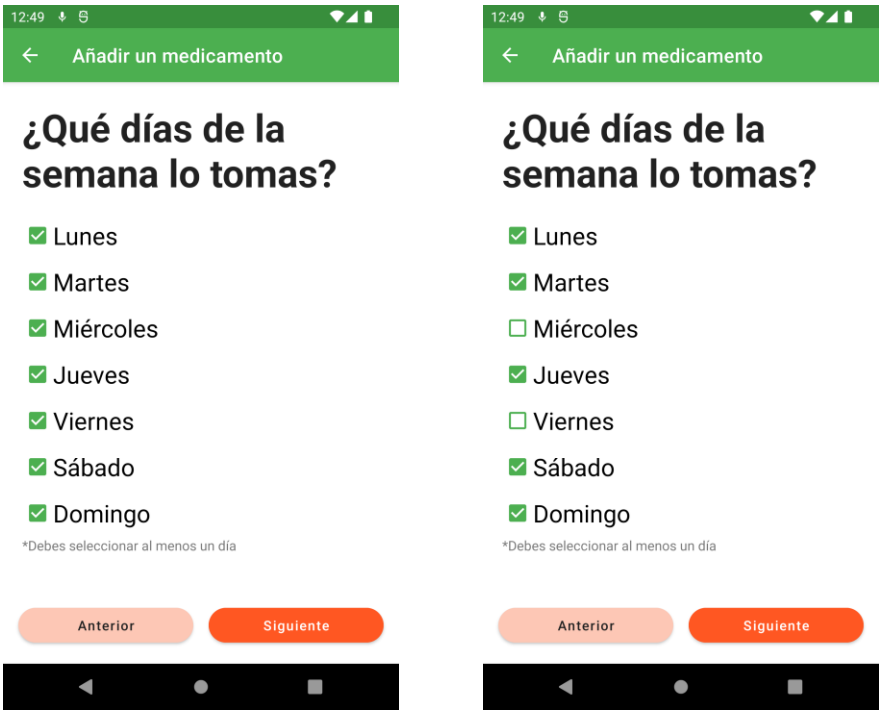

Figura 50 - Añadir un medicamento página 2

En la tercera pantalla, el usuario deberá elegir la dosis de cada toma, es decir, cuantas pastillas toma en cada momento, junto con las horas del día que las toma. Para añadir las horas, se le abrirá un cuadro de diálogo con un reloj para seleccionar la hora. En la Figura 51 se puede observar esta pantalla.

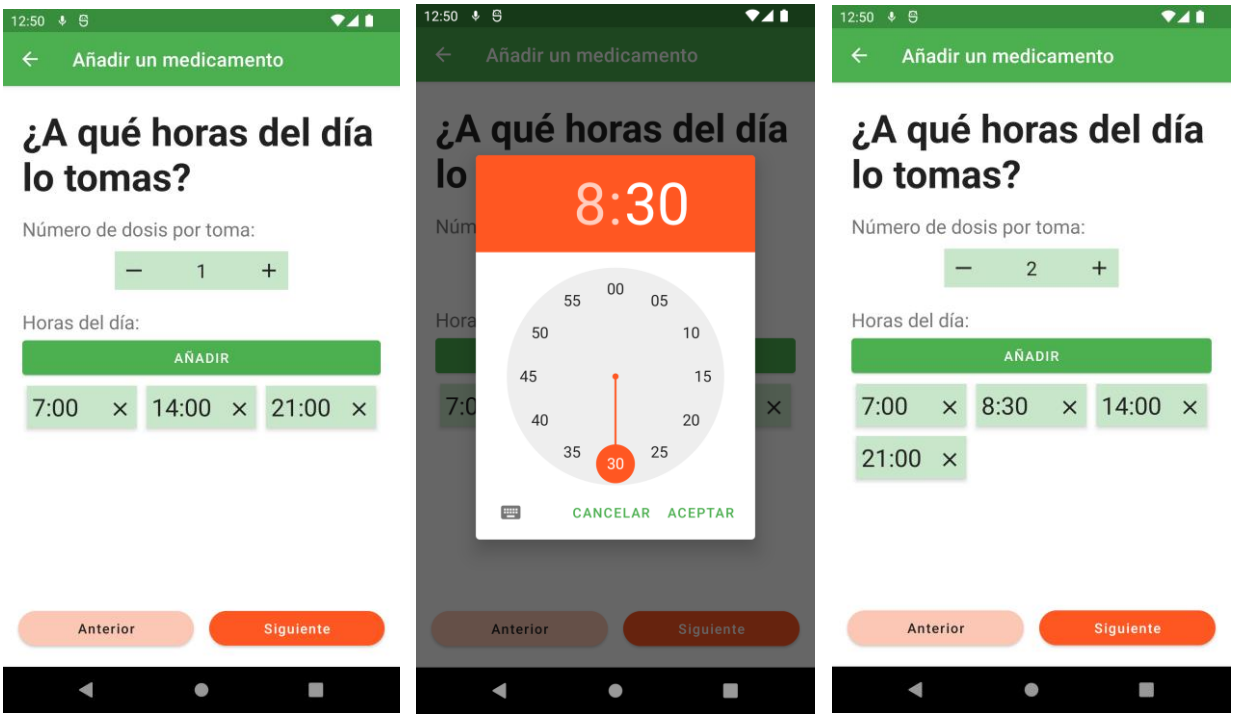

Figura 51 - Añadir un medicamento página 3

Por último, el usuario deberá elegir el intervalo de fechas en el que va a tomar el medicamento. Si tiene que tomarlo de forma indefinida, puede seleccionar el *checkbox* con la opción "Sin fecha de fin". Además, podrá elegir si quiere o no recibir notificaciones, como se ve reflejado en la Figura 52.

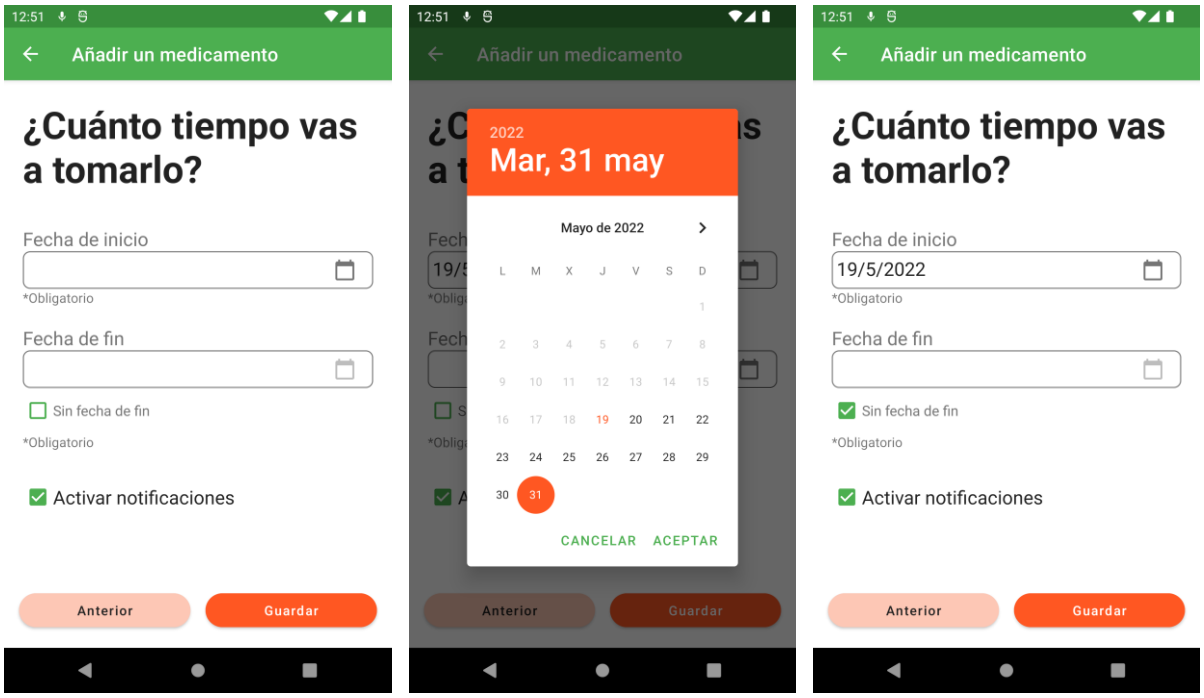

Figura 52 - Añadir un medicamento página 4

Una vez haya pulsado el botón de guardar, se habrá generado un nuevo elemento en la lista de medicamentos y la notificación correspondiente en la vista de notificaciones, en el caso en que la tuviese que tomar.

8.3.6 Visualizar detalles de un medicamento

Para visualizar los detalles completos de un medicamento, el usuario deberá pulsar sobre uno de ellos en la lista de medicamentos. Al realizar esta acción, se le abrirá un nuevo *activity* con dos partes diferenciadas, una cabecera y un cuerpo.

La cabecera aparece con el color asociado al medicamento y en ella se puede ver la foto, el nombre y la descripción. En el cuerpo, se encuentran diferentes tarjetas con el número de unidades por toma, los días de la semana que se toman, las horas del día, el periodo y si tiene activas las notificaciones.

Por último, aparecerá un icono con tres puntos en la parte superior que, al pulsarlo, se desplegará un menú con las opciones para editar el medicamento o eliminarlo. La opción de editar el medicamento es exactamente la misma que para añadir, con la diferencia de que los campos aparecerán rellenados, como se observa en la Figura 53.

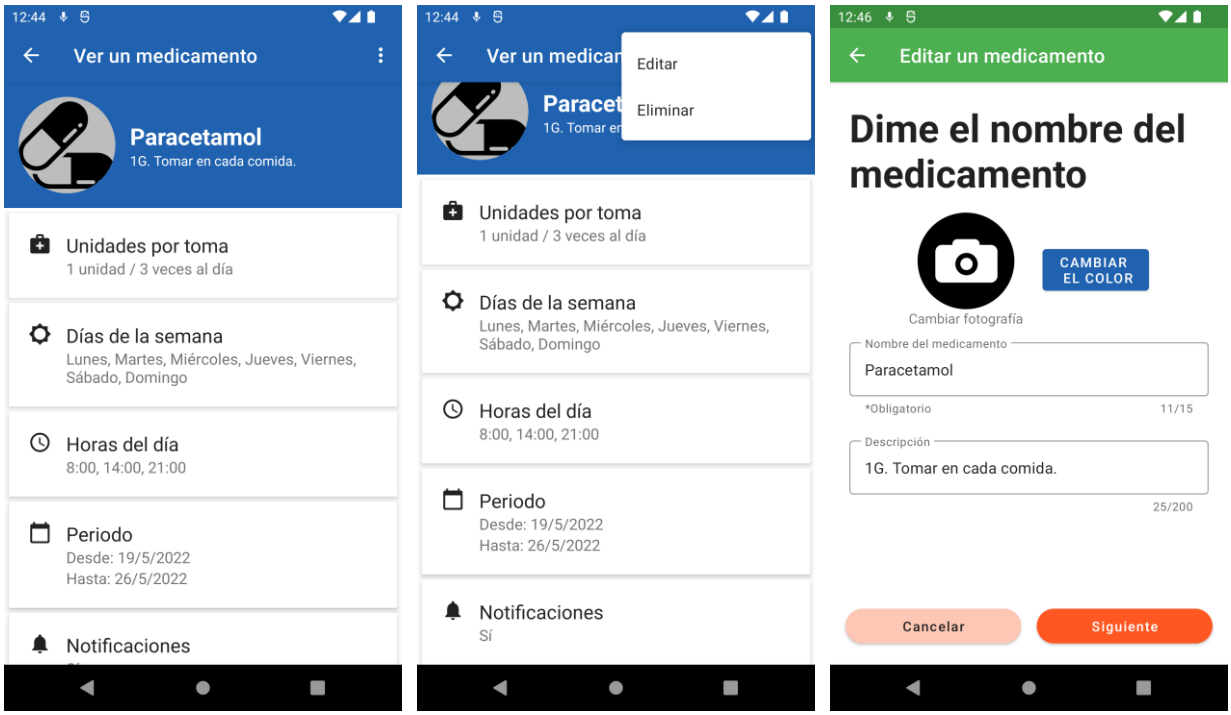

Figura 53 - Visualizar y editar un medicamento

8.3.7 Generación de notificaciones

Para la generación de notificaciones se hace uso de la clase *AlarmManager*, que registrará un *Intent* que el propio sistema recogerá y despertará a la aplicación correspondiente.

Para adaptarlo a la aplicación, se generará una alarma para cada notificación almacenada en el sistema, siempre que el usuario la haya activado para la medicación corresponde. La clase *Calendar* permite establecer una fecha y obtener el tiempo en milisegundos. Por defecto, la instancia tiene establecido el día del sistema, por lo que será necesario ajustarlo a la notificación, estableciendo el día de la semana, la hora y los minutos, como se observa en la Figura 54.

Además, será necesario establecer un identificador único a cada alarma, que se utilizará para eliminarlas en el caso de que el usuario borre la notificación o la marque como completada.

```
Intent intent = new Intent(context, AlarmReceiver.class);
AlarmManager alarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
PendingIntent pendingIntent;
if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.M)
    pendingIntent = PendingIntent.getBroadcast(context, numAlarms, intent, PendingIntent.FLAG_IMMUTABLE);
else
    pendingIntent = PendingIntent.getBroadcast(context, numAlarms, intent,  flags: 0);
Calendar calendar = Calendar.getInstance();
calendar.set(Calendar.SECOND, 0);
calendar.set(Calendar.MINUTE, notification.getHour().getMinutes());
calendar.set(Calendar.HOUR_OF_DAY, notification.getHour().getHours());
calendar.add(Calendar.DAY_OF_WEEK,  amount: (WeekDayEnum.getWeekDayInt(notification.getWeekDay()) - 1) % 7);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), pendingIntent);
numAlarms++;
```

Figura 54 - Generación de una alarma

Cuando el sistema despierte a la aplicación, se deberá tener un método que genere la notificación y la lance. Esto se hace creando una clase que extienda de *BroadcastReceiver*, que generará una notificación por un canal concreto, como se observa en la Figura 55.

```
public class AlarmReceiver extends BroadcastReceiver {

    public static final String CHANNEL_ID = "_main_channel";

    @Override
    public void onReceive(Context context, Intent intent) {
        NotificationCompat.Builder builder = new NotificationCompat.Builder(context, CHANNEL_ID)
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle("¡Hora de tomar la medicación!")
                .setAutoCancel(true)
                .setDefaults(NotificationCompat.DEFAULT_ALL)
                .setPriority(NotificationCompat.PRIORITY_HIGH);

        NotificationManagerCompat notificationManagerCompat = NotificationManagerCompat.from(context);
        notificationManagerCompat.notify( id: 123, builder.build());
    }
}
```

Figura 55 - Clase AlarmReceiver

Por último, a partir de la versión de Android Oreo (API 26), es necesario crear un canal de notificaciones. Esto se hace para dar más control al usuario sobre las notificaciones que recibe, aunque en este caso sólo existirá un tipo. En la Figura 56 se puede observar el método que crea el canal de notificaciones nada más iniciar la aplicación. Si ya está creado, no hace nada.

```java
private void createNotificationChannel() {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is new and not in the support library
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID,  name: "Notificaciones push", importance);
        channel.setDescription("main channel");
        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Figura 56 - Creación de un canal de notificaciones

Capítulo 9 - Conclusiones y trabajo futuro

9.1 Conclusiones

Los usuarios, mediante una interfaz sencilla e intuitiva, podrán realizar gestiones básicas sobre una agenda de contactos, realizar llamadas telefónicas, lanzar aplicaciones de terceros y gestionar sus medicamentos sin depender de otra persona.

La aplicación está pensada para personas con pocos conocimientos en el uso de aplicaciones móviles, pero intentando mantener una cierta libertad para aquellas con algo de más experiencia.

Se considera que la aplicación aquí desarrollada cumple los objetivos generales que se plantearon y en general, aporta un granito de arena para todas aquellas personas que se ven afectadas por la brecha tecnológica.

9.2 Trabajo futuro

La funcionalidad del proyecto se podría mejorar de la siguiente manera:

- **Evaluación con usuarios:** Lo más interesante de cara al futuro es realizar una evaluación con usuarios, ya que al final son los que van a hacer uso de la aplicación y nos pueden dar una idea de cómo se sienten ellos y con qué acciones se deben tomar para mejorar la aplicación.
- **Configuración de la aplicación:** Para una mayor adaptabilidad a los usuarios, la aplicación podría integrar un servicio de ajustes en el que poder configurar desde el tamaño del texto hasta limitar las acciones que el usuario puede hacer de la misma. Esto sería útil para adaptarse a los usuarios finales, en función de sus capacidades físicas.
- **Accesibilidad por voz:** Actualmente, la aplicación se controla haciendo uso de la pantalla, por lo que usuarios con una visión reducida podrían tener dificultades para utilizarla. Una solución interesante sería habilitar un botón a través del cual, mediante comandos previamente aprendidos, se pueda controlar la aplicación de manera casi completa. Por ejemplo, que a la hora

de añadir un contacto se pueda realizar un dictado al teléfono y rellene los campos automáticamente.

- **Mejora de notificaciones:** El módulo de medicamentos informa al usuario de que debe tomar su medicación, pero por ahora no proporciona información concreta.
- **Videollamadas:** Debido a la pandemia, muchas personas mayores quedaron aisladas durante el confinamiento y, aunque sea algo que esperemos que no vuelva a suceder, sería interesante ampliar la funcionalidad del módulo de llamadas para que mientras hablan por teléfono también se pudieran ver las caras.
- **Citas del médico:** Sería interesante añadir una pestaña más en el módulo de medicamentos que permita a los usuarios añadir citas con el médico, de esta forma, aparecerían en la pestaña de hoy al igual que los medicamentos.
- **Histórico de medicamentos:** Otra funcionalidad interesante para el módulo de medicamentos sería la posibilidad de llevar un histórico de medicamentos haciendo uso del calendario del sistema.
- **Reporte de errores:** Por último, se podría desarrollar un apartado para reportar errores o realizar sugerencias por parte de los usuarios para mejorar la aplicación.

Chapter - Conclusions and future work

Conclusions

Users, through a simple and intuitive interface, will be able to perform basic management of a contact list, make phone calls, launch third-party applications, and manage their medication without depending on another person.

The application is designed for people with little knowledge in the use of mobile applications but trying to maintain a certain freedom for those with more experience.

It is considered that the application developed here fulfils the general objectives that were set out and, in general, contributes a grain of sand for all those people who are affected by the technological gap.

Future work

The functionality of the project could be improved in the following way:

- **Evaluation with users**: The most interesting thing for the future is to carry out an evaluation with users, since in the end they are the ones who are going to make use of the application and they can give us an idea of how they feel and with what actions should be taken to improve the application.
- **Configuration of the application**: For greater adaptability to users, the application could integrate a settings service in which they could configure everything from the size of the text to limiting the actions that the user can perform on the application. This would be useful for adapting to end users, depending on their physical capabilities.
- **Voice accessibility**: Currently, the application is controlled using the screen, so users with reduced vision may have difficulty using it. An interesting solution would be to enable a button through which, by means of previously learned commands, the application can be controlled almost completely. For example, when adding a contact, it could be dictated to the phone and fill in the fields automatically.

- **Improved notifications**: The medication module informs the user that he/she should take his/her medication, but so far does not provide concrete information.
- **Video calls**: Due to the pandemic, many elderly people were isolated during the confinement and, although it is something that hopefully will not happen again, it would be interesting to extend the functionality of the call module so that while talking on the phone they can also see each other's faces.
- **Doctor's appointments**: It would be interesting to add an extra tab in the medication module that allows users to add doctor's appointments, so they would appear in today's tab as well as the medications.
- **Medication history**: Another interesting functionality for the medication module would be the possibility of keeping a medication history using the system's calendar.
- **Error reporting**: Finally, a section could be developed to report errors or make suggestions from users to improve the application.

BIBLIOGRAFÍA

[1] Teléfono de Google. Disponible: https://play.google.com/store/apps/details?id=com.google.android.dialer. [Último acceso: 18 de mayo de 2022]

[2] Pixel Launcher. Disponible: https://play.google.com/store/apps/details?id=com.google.android.apps.nexuslauncher. [Último acceso: 18 de mayo de 2022]

[3] MyTherapy. Disponible: https://www.mytherapyapp.com/es. [Último acceso: 18 de mayo de 2022]

[4] Android Studio. Disponible: https://es.wikipedia.org/wiki/Android_Studio. [Último acceso: 4 de mayo de 2022]

[5] Proveedor de Contenidos. Disponible: https://developer.android.google.cn/guide/topics/providers/content-providers?hl=es-419. [Último acceso: 18 de mayo de 2022]

[6] Proveedor de contactos. Disponible: https://developer.android.com/guide/topics/providers/contacts-provider. [Último acceso: 7 de mayo de 2022]

[7] SharedPreferences. Disponible: https://developer.android.com/reference/android/content/SharedPreferences. [Último acceso: 7 de mayo de 2022]

[8] Room. Disponible: https://developer.android.com/training/data-storage/room. [Último acceso: 6 de mayo de 2022]

[9] ViewModel. Disponible: https://developer.android.com/topic/libraries/architecture/viewmodel. [Último acceso: 7 de mayo de 2022]

[10] LiveData. Disponible: https://developer.android.com/topic/libraries/architecture/livedata?hl=es-419#livedata-in-architecture. [Último acceso: 7 de mayo de 2022]

[11] Gson. Disponible: https://es.wikipedia.org/wiki/Gson. [Último acceso: 7 de mayo de 2022]

[12] ColorPicker. Disponible: https://github.com/kristiyanP/colorpicker. [Último acceso: 7 de mayo de 2022]

[13] Git. Disponible: https://es.wikipedia.org/wiki/Git. [Último acceso: 4 de mayo de 2022]

[14] Java. Disponible: https://www.oracle.com/es/java/. [Último acceso: 4 de mayo de 2022]

[15] Modelio. Disponible: https://www.modelio.org/about-modelio/features.html. [Último acceso: 6 de mayo de 2022]

[16] RFC 2396. Disponible: https://www.rfc-editor.org/rfc/rfc2396.html. [Último acceso: 10 de mayo de 2022]

[17] MVVM. Disponible: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/. [Último acceso: 13 de mayo de 2022]

[18] Twitimer. Disponible: https://github.com/mouredev/Twitimer-Android. [Último acceso: 13 de mayo de 2022]

[19] ViewBinding. Disponible: https://developer.android.com/topic/libraries/view-binding. [Último acceso: 14 de mayo de 2022]

[20] DataBinding. Disponible: https://developer.android.com/topic/libraries/data-binding. [Último acceso: 14 de mayo de 2022]

[21] Repository. Disponible: https://dev.to/rodrassilva/android-repository-pattern-using-room-retrofit-and-coroutines-58kb. [Último acceso: 15 de mayo de 2022]

[22] Singleton. Disponible: https://es.wikipedia.org/wiki/Singleton. [Último acceso: 15 de mayo de 2022]

[23] DAO. Disponible: https://es.wikipedia.org/wiki/Objeto_de_acceso_a_datos. [Último acceso: 15 de mayo de 2022]

[24] DAO + ROOM. Disponible: https://developer.android.com/reference/androidx/room/Dao. [Último acceso: 15 de mayo de 2022]

[25] Material.io. Disponible: https://material.io/resources/color/. [Último acceso: 18 de mayo de 2022]

ANEXO: GUÍA DEL USUARIO

Para comenzar a utilizar la aplicación, el primer paso es ir al menú principal del dispositivo y buscar la aplicación *SimpleLauncher* que tiene un icono de un cohete. Una vez localizada, se tiene que pulsar sobre ella para abrirla.

Una vez abierta, como la aplicación no requiere ningún inicio de sesión, ya estará funcionando. El siguiente paso que debe hacer el usuario es pulsar el botón *home* del teléfono, de esta forma el móvil reconocerá el launcher y le preguntará al usuario si desea establecer la aplicación como aplicación de inicio. El usuario deberá elegir la opción de "Siempre". En la figura 57 se puede observar el primer inicio de sesión.

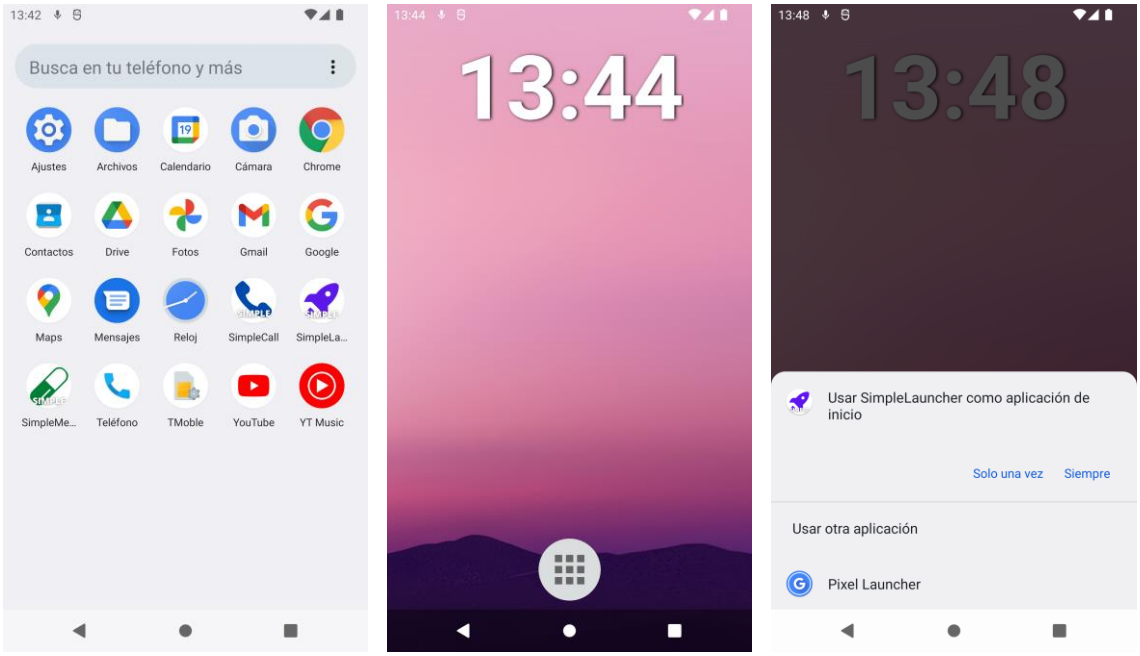

Figura 57 - Primer inicio de la aplicación

La primera vez que iniciemos la aplicación, se mostrará un escritorio vacío, únicamente con la hora y un botón que nos va a llevar al *box* de aplicaciones. Pulsando sobre el mismo, se abrirá una lista con las aplicaciones instaladas, y mediante el botón de los tres puntos ubicado en la parte superior, se podrán modificar los accesos directos, el orden de estos y el fondo de pantalla. En la Figura 58 se puede observar la vista de aplicaciones instaladas.

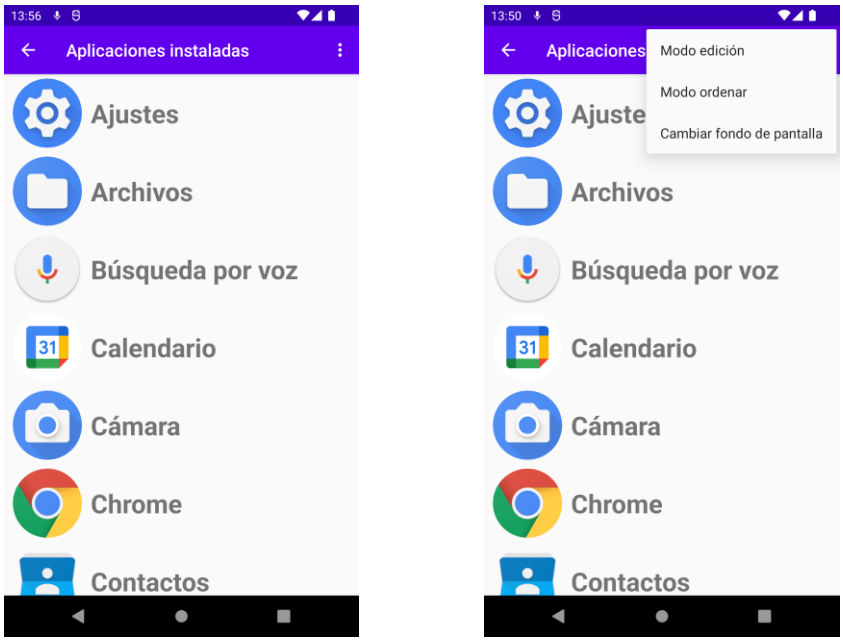

Figura 58 - Vista de las aplicaciones

Para cambiar los accesos directos, el usuario debe pulsar la opción de "Modo edición". Una vez hecho esto, en cada aplicación aparecerá un *checkbox*, que el usuario debe pulsar en el caso de que quiera añadir una aplicación. Una vez seleccionadas las deseadas, para salir se puede pulsar de nuevo el botón de los tres puntos. Cuando haya realizado esta acción aparecerá un mensaje informativo en la pantalla, como se observa en la Figura 59.

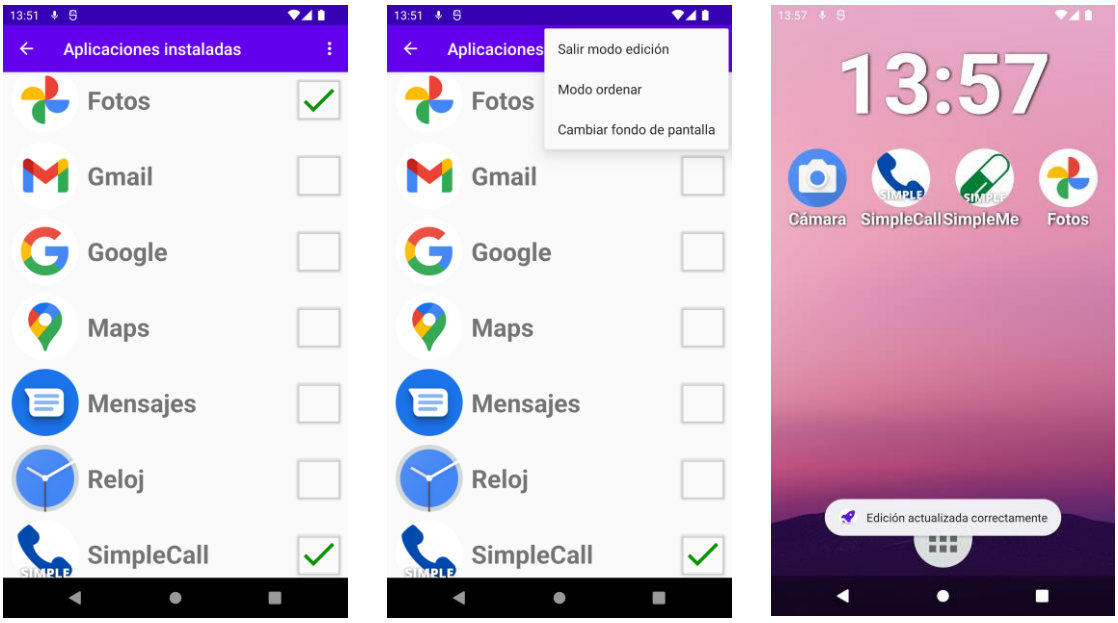

Figura 59 - Pantalla para cambiar los accesos directos

Para cambiar el orden en el que las aplicaciones aparecen en el menú principal, el usuario deberá pulsar la opción de "Modo ordenar". Ahora aparecerán sólo aquellas aplicaciones con acceso directo y el usuario deberá interactuar con las flechas para cambiar el orden. Al igual que en la anterior opción, cuando finalice aparecerá un mensaje en la pantalla, como se observa en la Figura 60.

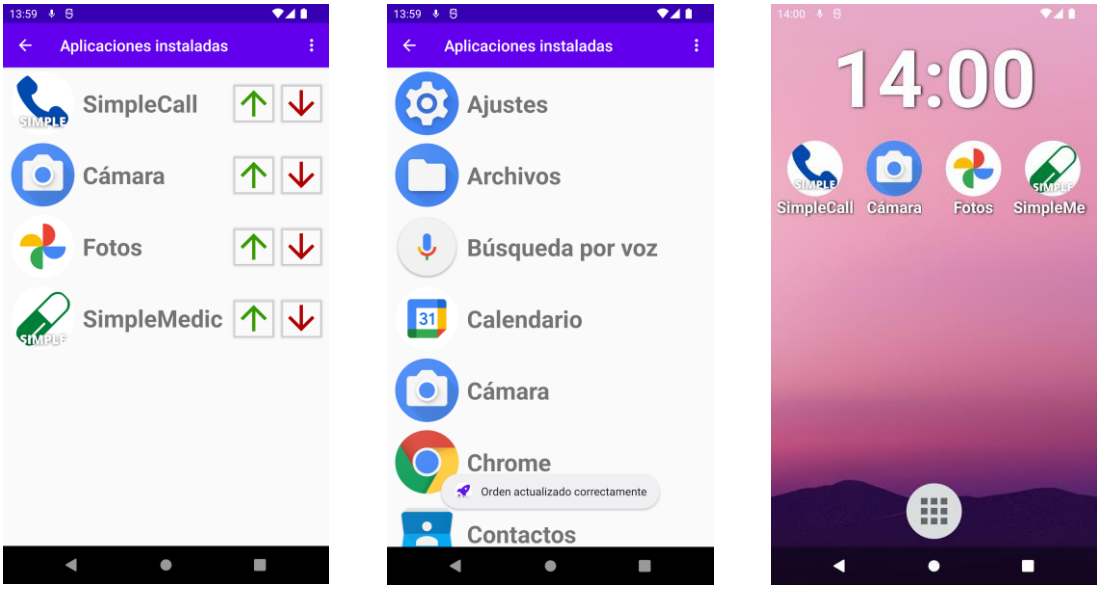

Figura 60 - Pantalla para ordenar las aplicaciones

Para cambiar el fondo de pantalla, se seguirán los mismos pasos, seleccionando la opción "Cambiar fondo de pantalla", como se observa en la Figura 61.

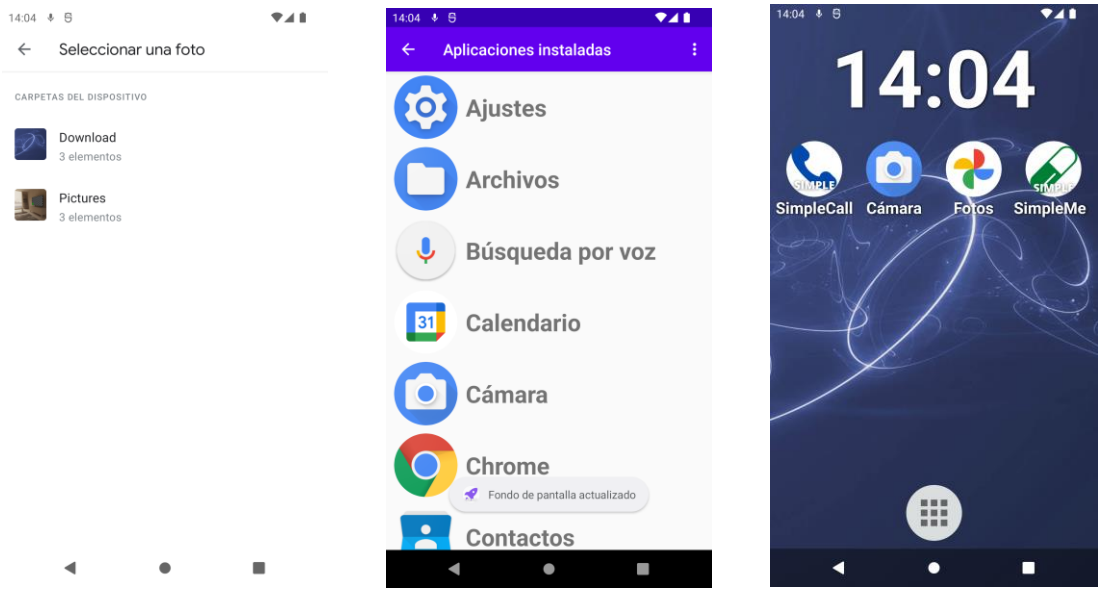

Figura 61 - Pantalla para cambiar el fondo

Ahora, para acceder al módulo de llamadas, el usuario deberá pulsar el icono de un teléfono con el nombre "SimpleCall". Nada más abrirlo, se le solicitarán algunos permisos que deberá aceptar. Si tiene algún contacto almacenado en el dispositivo, le aparecerán directamente. Lo mismo ocurre con el registro de llamadas, como se observa en la Figura 62. Para navegar por el módulo el usuario deberá pulsar los botones de la parte superior.

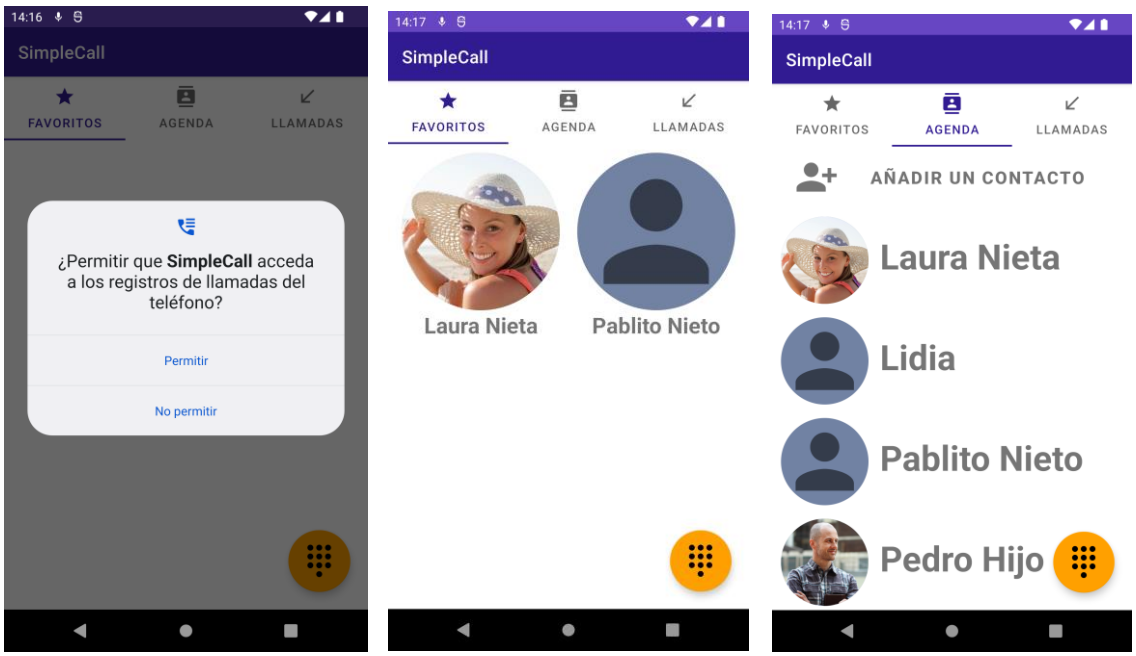

Figura 62 - Inicio del módulo teléfono

Para ver los detalles de un contacto, el usuario deberá dirigirse al apartado "Agenda" y seleccionar un usuario. Para llamar al contacto, el usuario deberá pulsar el botón de llamada. Para realizar su edición, deberá pulsar el icono con el lápiz ubicado en la esquina inferior derecha. Una vez realizada la edición, deberá pulsar el icono de guardado. También podrá eliminar el contacto pulsando el botón de la papelera, como se observa en la Figura 63.

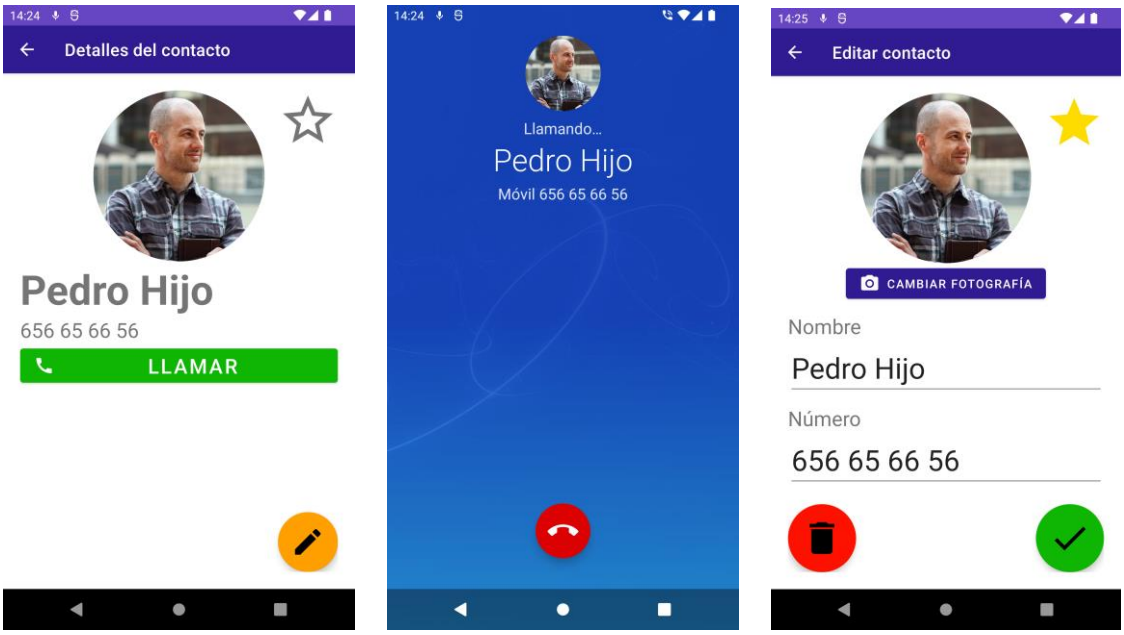

Figura 63 - Visualización y edición de un contacto

También es posible realizar una llamada desde la pestaña de favoritos o el registro de llamadas. En este caso solo deberá pulsar sobre el contacto deseado.

Si el usuario lo desea, se podrá realizar una marcación telefónica manual a través del icono que se encuentra en la parte inferior de la pantalla principal. El usuario deberá introducir el número y pulsar el botón de llamar, como se observa en la Figura 64.

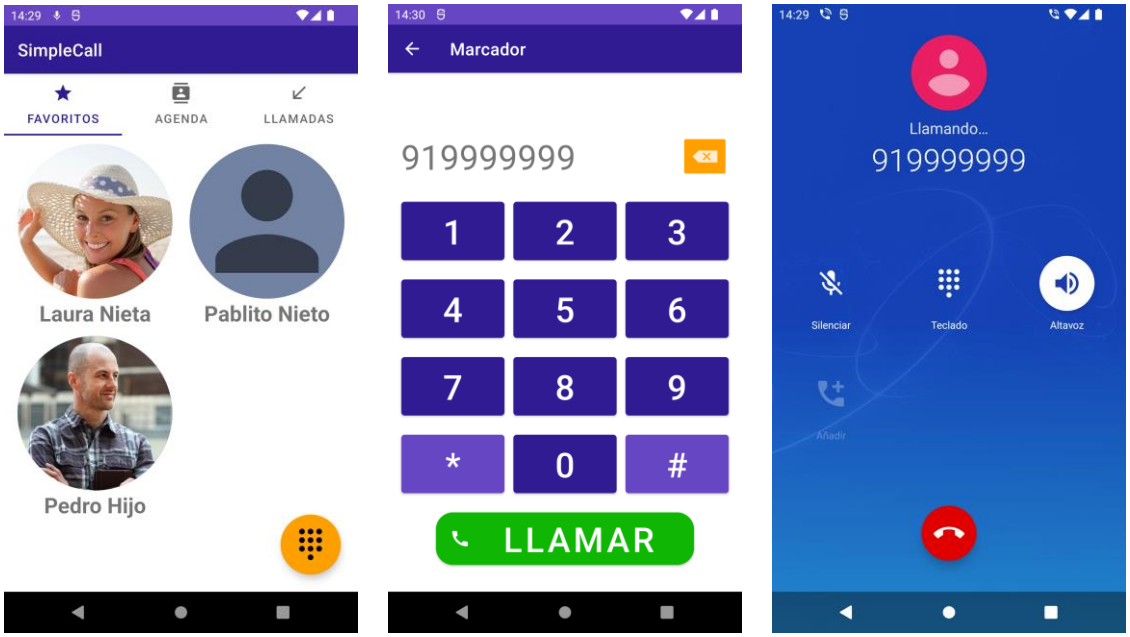

Figura 64 - Llamada telefónica mediante el marcador

Por último, el usuario podrá agregar nuevos contactos desde la vista de agenda. Sólo deberá pulsar el botón de "Añadir un contacto", ubicado en la parte superior. De esta forma, se abrirá una nueva ventana para rellenar los datos del nuevo contacto. Para guardarlo deberá pulsar el botón verde, como se observa en la Figura 65.

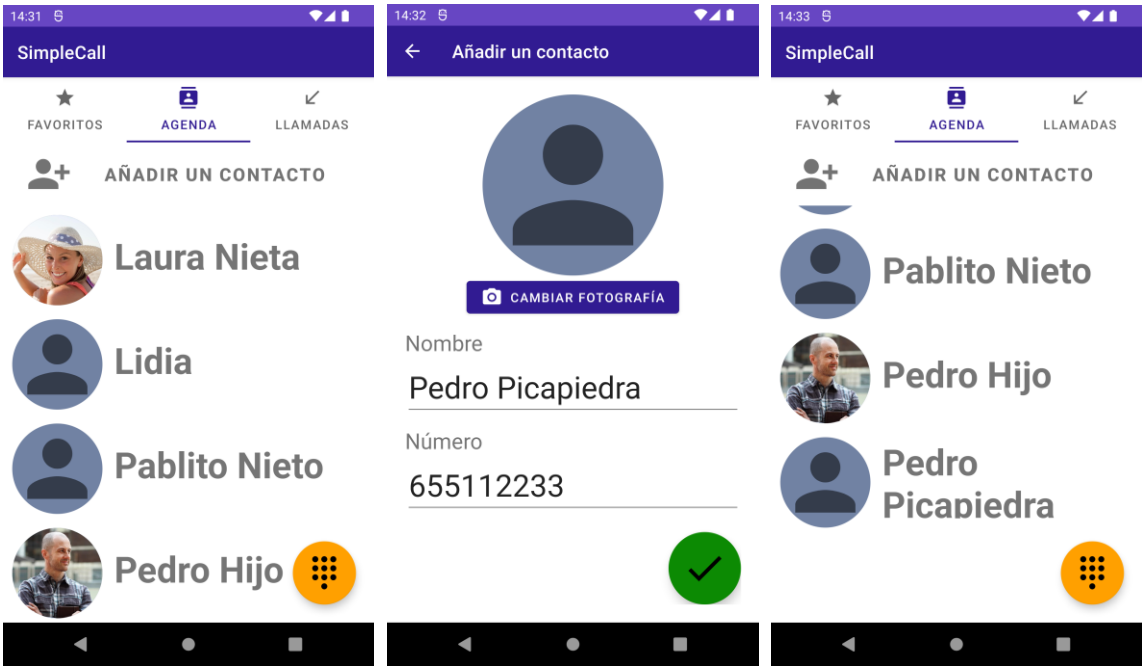

Figura 65 - Añadir un nuevo contacto

Para acceder al módulo de los medicamentos, el usuario debe pulsar el botón de inicio de su dispositivo y buscar la aplicación con el icono de una pastilla, llamada "SimpleMedicine". Nada más iniciarse la aplicación, no habrá ningún medicamento, por lo que el usuario deberá pulsar el botón naranja de la esquina inferior derecha para añadir uno nuevo, como se muestra en la Figura 66.

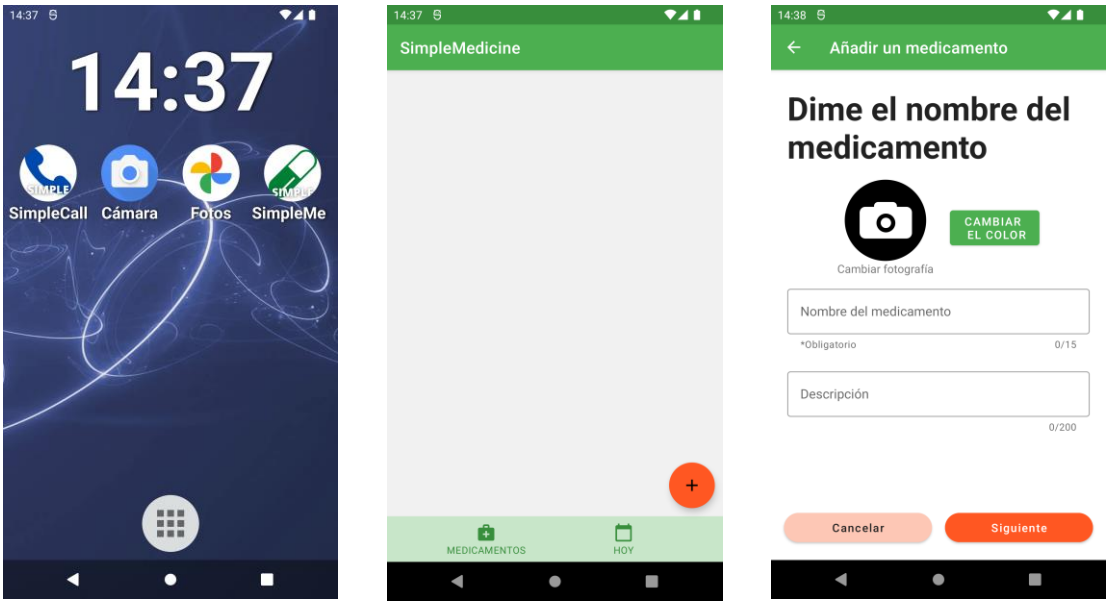

Figura 66 - Inicio del módulo de los medicamentos

Ahora el usuario deberá ir rellenando las distintas pantallas, avanzando mediante la pulsación del botón de "Siguiente", como se muestra en la Figura 67.

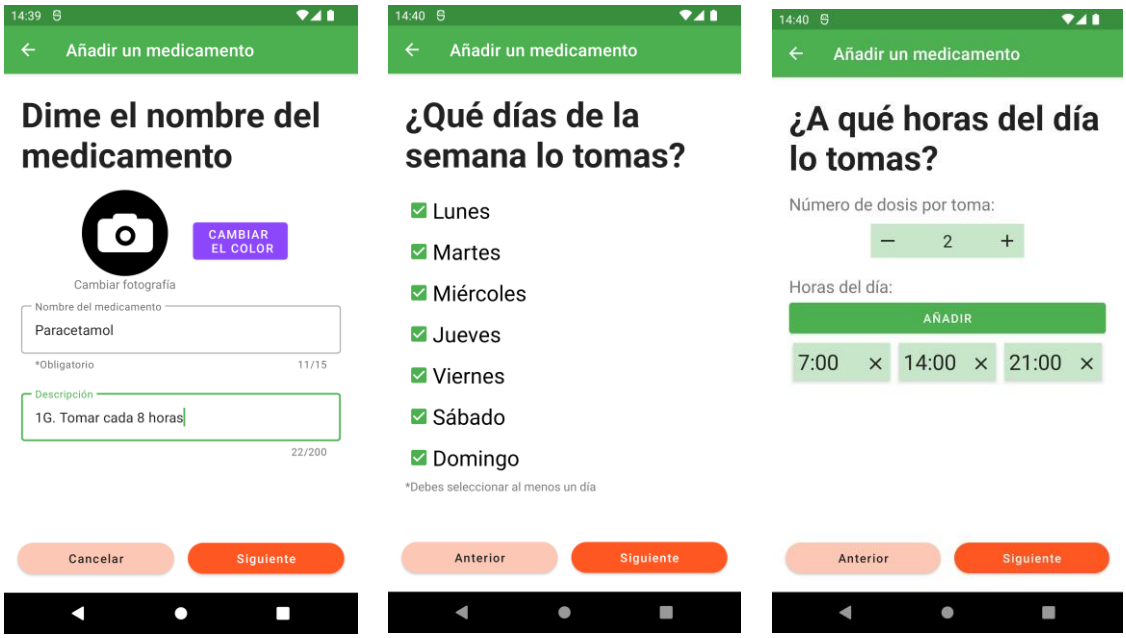

Figura 67 - Añadir un medicamento

Cuando el usuario llegue a la última pantalla, deberá pulsar el botón de "Guardar". Ahora se puede observar como el medicamento aparece en la lista de medicamentos y, si así lo ha rellenado, también aparecerá la notificación correspondiente, como se observa en la Figura 68.

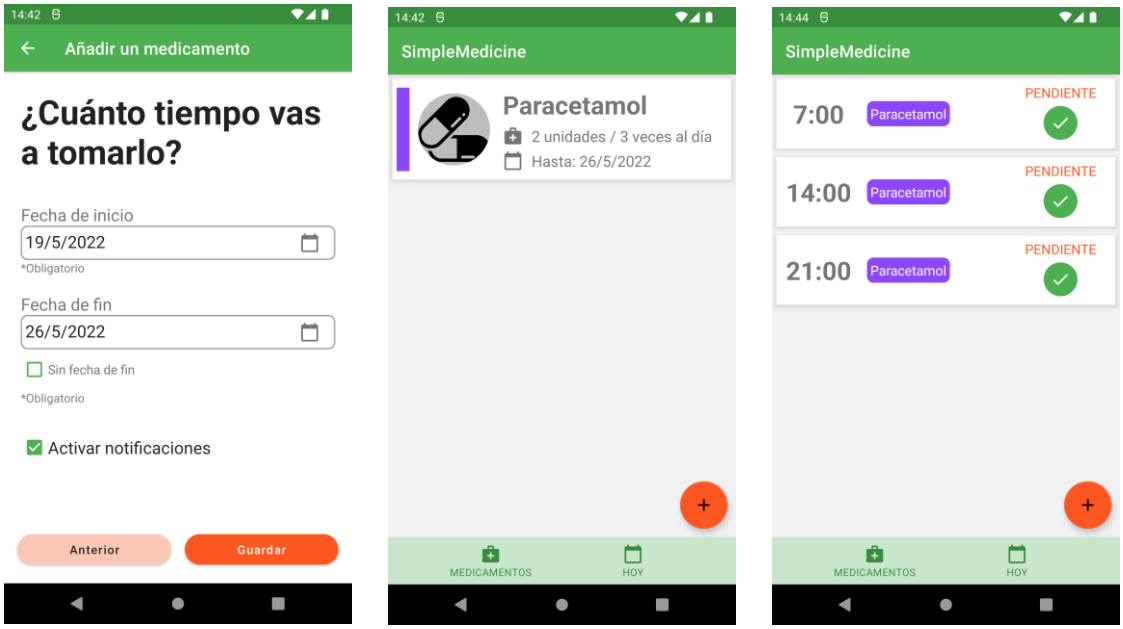

Figura 68 - Visualizar medicamento añadido

Ahora, el usuario podrá visualizar los detalles de un medicamento tocando sobre él. Además, también podrá editarlo o borrarlo mediante el icono de los tres puntos que aparece en la parte superior. El proceso de edición es idéntico al de añadir un medicamento nuevo con la diferencia de que los campos aparecen rellenos, como se observa en la Figura 69.

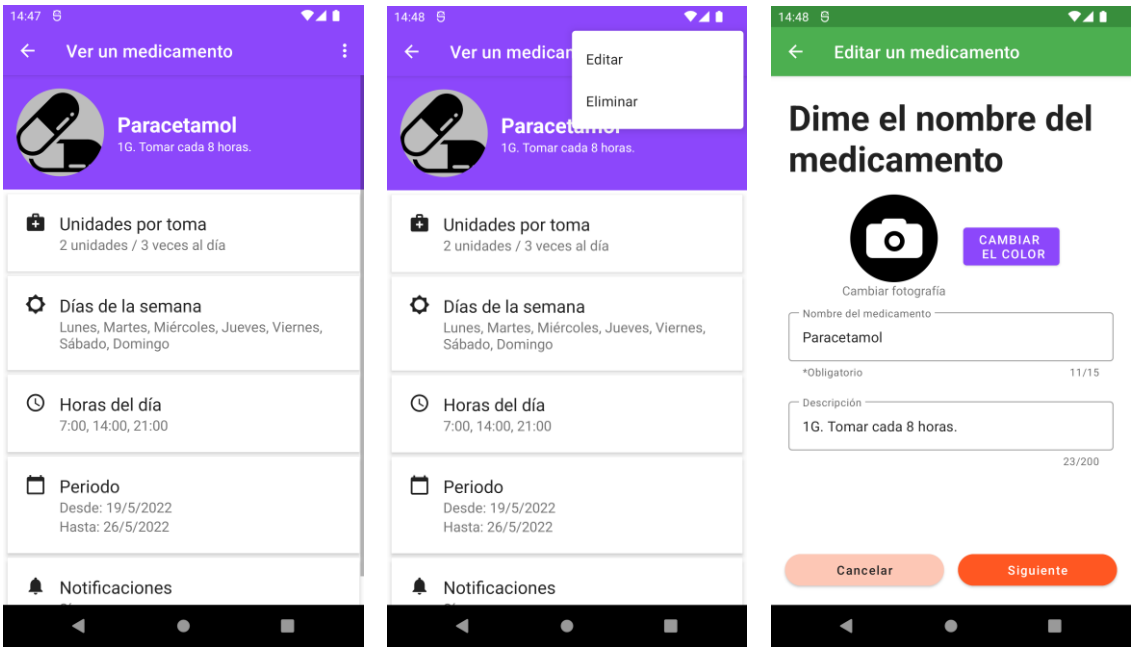

Figura 69 - Detalles y edición de un medicamento

El usuario podrá marcar como completados aquellas notificaciones cuyo medicamento ya se ha tomado. Simplemente deberá pulsar el botón con el check verde, que pasará a estar deshabilitado. Si tiene varios medicamentos con la misma hora, aparecerán agrupados en una misma notificación, como se observa en la figura 70.

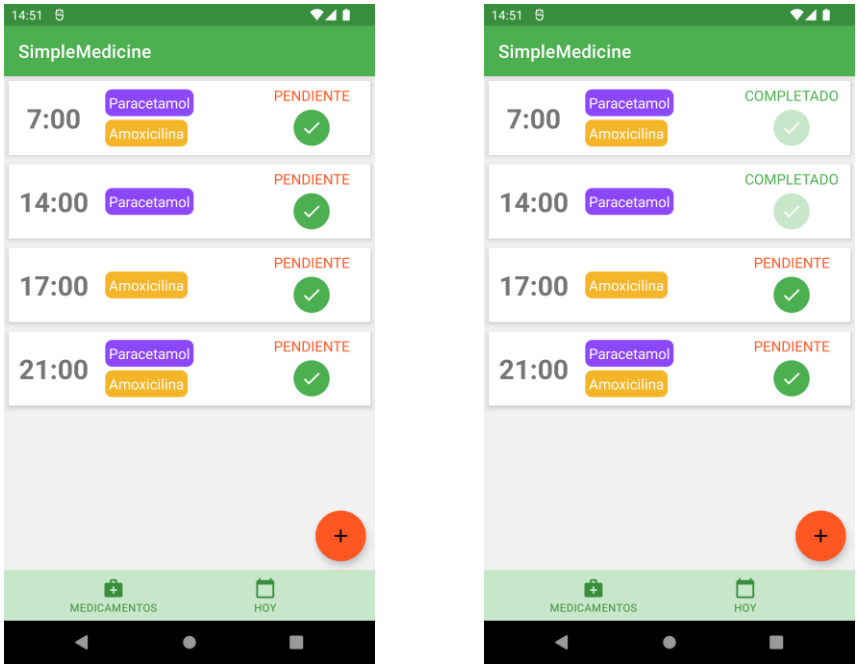

Figura 70 - Confirmación de los medicamentos

Por último, la aplicación enviará una notificación al usuario para avisarle de que debe tomarse la medicación, como se observa en la Figura 71.

Figura 71 - Notificación