



# **NURSI**PEDIA

PLS APPDEVELOPER

**PROYECTO INTEGRADO  
DESARROLLO DE APLICACIONES MULTIPLATAFORMA**

**JOSÉ LÓPEZ SALADO**

**16/06/2023**

## ÍNDICE

1.- Introducción	3
2.- Estudio de Viabilidad	4
2.1.- Descripción del Sistema Actual	
2.2.- Descripción del Sistema Nuevo	
2.3.- Identificación de Requisitos del Sistema	5
2.3.1.- Requisitos de información	
2.3.2.- Requisitos funcionales	
2.3.3.- Otros Requisitos	6
2.4.- Descripción de la solución	
2.5.- Planificación del proyecto	7
2.5.1.- Equipo de trabajo	
2.5.2.- Planificación temporal	
2.6.- Estudio del coste del proyecto	8
3.- Análisis del Sistema de Información	9
3.1.- Identificación del entorno tecnológico	
3.2.- Modelado de datos	12
3.2.1.- Modelo Entidad-Relación	
3.1.2.- Esquema de la base de datos	13
3.1.3.- Datos de prueba	
3.3.- Identificación de los usuarios participantes y finales	14
3.4.- Identificación de subsistemas de análisis	15
3.5.- Establecimiento de requisitos	
3.6.- Diagramas de Análisis	16
3.7.- Definición de interfaces de usuario	17
3.7.1.- Especificación de principios generales de interfaz	
3.7.2.- Especificación de formatos individuales de la interfaz de pantalla	19
3.7.3.- Identificación de perfiles de usuario	20
3.7.4.- Especificación de formatos de impresión	
3.7.5.- Especificación de la navegabilidad entre pantallas	21
4.- Construcción/Compilación del Sistema	22
5.- Breve manual de instalación y uso de la Aplicación	40
6.- Conclusiones	44
7.- Glosario de términos	
8.- Bibliografía	45

## INTRODUCCIÓN

NursiPedia es una aplicación móvil diseñada para ser usada por enfermeros, es una herramienta perfecta para hacer su trabajo diario mucho más sencillo y eficiente.

Está compuesta por muchas herramientas útiles. Por ejemplo, he incluido calculadoras para infusiones\*, disoluciones\*, cálculo del índice de masa corporal (IMC)\* y gasto cardíaco\* entre otras.

También les da acceso rápido a una gran lista de medicamentos, de los cuales podrán ver su información detallada, como advertencias, finalidad y propósitos terapéuticos.

Otra de las funciones que incluye es crear y guardar notas personalizadas. Podrán añadirle un título, escribir el contenido y ver la fecha de creación, e incluso añadir recordatorios a una hora específica que les avisará en forma de notificación.

En cuanto a las tecnologías que he utilizado, he creado NursiPedia con Android Studio y está programada en lenguaje de programación Kotlin.

Para gestionar la base de datos y los usuarios, he utilizado Firebase, una plataforma en la nube que nos ofrece Google y que nos permite múltiples funciones. Además, he empleado Retrofit y Postman para comunicarme con otros servicios externos y realizar llamadas a APIs.

A lo largo del documento, explicaré todos los aspectos del proyecto, desde el estudio de viabilidad hasta el análisis del sistema de información, la construcción del sistema y los manuales de instalación y uso. Además de explicar en detalle cada una de las herramientas utilizadas.



## ESTUDIO DE VIABILIDAD

### 2.1.- Descripción del Sistema Actual

Sin NursiPedia, los enfermeros se enfrentan a tareas y actividades de manera manual y que requieren un mayor tiempo y esfuerzo.

Las actividades como el cálculo de infusiones, disoluciones y el índice de masa corporal (IMC) se realizan utilizando fórmulas y cálculos manuales. Esto implicaba un mayor margen de error y una gran pérdida de tiempo.

Los enfermeros tienen que buscar y consultar libros de referencia o fuentes externas como internet para obtener información sobre medicamentos, sus propósitos terapéuticos y advertencias asociadas. Esto implicaba un proceso lento y no siempre se tiene acceso a la información más actualizada.

La notas y el registro de información se realiza de forma manual, utilizando papel y lápiz u otras aplicaciones de notas. Esto dificulta la organización y recuperación de datos importantes, al igual que no disponen de recordatorios.

### 2.2.- Descripción del Sistema Nuevo

Con NursiPedia, los enfermeros realizan estas tareas de forma rápida y sencilla, ya que la aplicación les proporciona soluciones y herramientas eficientes que mejoran la productividad y velocidad en su trabajo diario.

NursiPedia ofrece muchas calculadoras médicas, como las de infusiones, disoluciones, IMC y gasto cardíaco. Estas calculadoras automatizadas realizan los cálculos de forma rápida y precisa, ahorrando tiempo y reduciendo errores.

La aplicación cuenta con una extensa lista de medicamentos, donde los enfermeros pueden encontrar información detallada sobre cada uno, incluyendo su finalidad terapéutica, advertencias y nombres comerciales. Esto les permite acceder a la información de manera rápida y actualizada.

NursiPedia permite a los enfermeros crear y guardar notas personalizadas en forma de listas. Los enfermeros pueden acceder a estas notas desde cualquier dispositivo compatible con la aplicación y les proporciona la capacidad de establecer recordatorios en momentos específicos para mantenerse al tanto de las tareas importantes, además de poder crear y eliminar notas en todo momento desde cualquier lugar o dispositivo móvil.

## 2.3.- Identificación de Requisitos del Sistema

### 2.3.1.- Requisitos de información

Información del usuario, la aplicación debe ser capaz de saber en todo momento que usuario ha iniciado sesión para trabajar solo sobre las notas de dicho usuario.

Para ello sabemos en todo momento el email del usuario o enfermero que está usando la aplicación.

Información del medicamento, la aplicación a través de las llamadas a la API\* es capaz de saber en todo momento los datos de los medicamentos, conociendo su nombre científico, su nombre comercial, el propósito o finalidad del medicamento y las advertencias de uso.

Es importante mantener actualizada la información de los medicamentos para asegurar la precisión de los datos.

Información de las notas, la aplicación debe tener acceso en todo momento a la base de datos de Firebase Firestore que contiene la tabla notas con todas las notas que se hayan creado desde la aplicación, conociendo el título de cada una de ellas, el contenido, el id, la fecha y el email del usuario o enfermero que ha creado dicha nota.

Información de cálculos médicos, controlando en todo momento los números y operaciones usadas en cada una de las calculadoras que ofrece la aplicación, para llevar a cabo un buen control de los datos y de las operaciones realizadas.

### 2.3.2.- Requisitos funcionales

Registro de usuarios, permitiendo a los enfermeros crear una cuenta personal en NursiPedia para acceder a todas las funcionalidades de la aplicación, especialmente a las notas, guardando y mostrando únicamente las que corresponden a su usuario.

Cálculo automático de infusiones, la aplicación debe ser capaz de calcular automáticamente las horas, la velocidad y el volumen de la infusión según los datos ingresados.

Cálculo automático de disoluciones, la aplicación debe ser capaz de calcular automáticamente cantidad de medicamento, la concentración de la disolución y el volumen de la disolución según los datos ingresados.

Cálculo automático de pérdidas insensibles, la aplicación debe ser capaz de calcular automáticamente la dosis en mililitros de pérdidas insensibles según los datos ingresados.

Cálculo automático del índice de masa corporal, la aplicación debe ser capaz de calcular automáticamente el IMC según los datos de peso y altura ingresados.

Cálculo automático de reglas de tres, la aplicación debe ser capaz de calcular automáticamente el valor que falta según los datos ingresados.

Cálculo automático de gasto cardíaco, la aplicación debe ser capaz de calcular automáticamente el gasto cardíaco según la frecuencia cardíaca y el volumen sistólico del paciente.

### 2.3.3.- Otros Requisitos

Interfaz intuitiva y fácil de usar, la interfaz de NursiPedia debe ser intuitiva y amigable para facilitar su uso por parte de los enfermeros, para ello se utilizan colores que no molestan a la vista y las diferentes secciones se encuentran organizadas de forma bastante ergonómica.

Seguridad de los datos, la información de las notas de los enfermeros y otros datos sensibles como los detalles de los medicamentos están protegidos y seguros tanto por parte de Firebase como por parte de la API que contiene la información de los medicamentos.

Compatibilidad con diferentes dispositivos, la aplicación se puede usar tanto en dispositivos móviles Android de cualquier tamaño, como en tablets.

No está disponible sin conexión, ya que necesita de ella para acceder con tu cuenta a tus notas y para obtener los datos de los medicamentos, lo que si se permite sin conexión es utilizar todas las calculadoras o fórmulas que ofrece la aplicación.

## 2.4.- Descripción de la solución

La solución principal consiste en el desarrollo de una aplicación móvil utilizando Android Studio y el lenguaje de programación Kotlin. La aplicación estará diseñada específicamente para dispositivos Android, lo que permitirá a los enfermeros acceder a todas las funcionalidades y herramientas desde sus teléfonos móviles o tablets.

Para gestionar la base de datos y la autenticación de usuarios, he utilizado Firebase, una plataforma en la nube proporcionada por Google. Firebase ofrece servicios como Firestore, para almacenar y sincronizar datos en tiempo real, así como para autenticar y gestionar usuarios de forma segura, además de estadísticas sobre cuantas personas usan la aplicación y desde que localización lo hacen.

Se utilizará Retrofit, una biblioteca de Android, para realizar llamadas a APIs y comunicarse con servicios externos. Esto permitirá obtener información actualizada sobre medicamentos, gracias a OpenFDA API.

Interfaz de usuario intuitiva y amigable, para garantizar que los enfermeros puedan navegar fácilmente por la aplicación y acceder a las diferentes funcionalidades de manera rápida y eficiente.

## 2.5.- Planificación del proyecto

### 2.5.1.- Equipo de trabajo

En el desarrollo de NursiPedia, he sido el único responsable del diseño, desarrollo y puesta en marcha de la aplicación. He trabajado de manera individual en todas las etapas del proyecto, desde el análisis de requisitos hasta la implementación y pruebas.

He asumido todos los roles necesarios para llevar a cabo el proyecto exitosamente, tanto el diseño de interfaces, como la programación en Kotlin utilizando Android Studio y la gestión de la base de datos en Firebase.

Además, he realizado investigaciones y consultas adicionales cuando ha sido necesario para realizar tareas específicas, de las cuales no conocía su funcionamiento.

A pesar de trabajar de manera individual, he seguido buenas prácticas de desarrollo de software, como la documentación del código, la realización de pruebas y la optimización del rendimiento.

El equipo utilizado para llevar a cabo el desarrollo del proyecto, ha sido un ordenador de sobremesa con el Sistema Operativo de Windows 10, con la última versión de Android Studio instalada.

El equipo consta de 16 GB de RAM, 500 GB SSD de almacenamiento y un procesador Intel Core i5.

### 2.5.2.- Planificación temporal

Análisis de requisitos. *(10 de Marzo – 14 de Marzo)*

Definición de los requisitos funcionales y no funcionales de la aplicación.

Identificación de las necesidades y preferencias de los usuarios objetivo.

Diseño de interfaz y arquitectura. *(15 de Marzo – 20 de Marzo)*

Creación de prototipos de interfaz de usuario utilizando herramientas de diseño.

Definición de la arquitectura de la aplicación y la estructura de la base de datos.

Desarrollo de funcionalidades principales. *(21 de Marzo – 12 de Abril)*

Implementación de las funcionalidades principales de NursiPedia, como las calculadoras y la gestión de medicamentos.

Integración de Firebase para la autenticación de usuarios y el almacenamiento de datos.

Pruebas y correcciones. *(13 de Abril – 15 de Mayo)*

Realización de pruebas exhaustivas para identificar y corregir posibles errores o fallos de funcionamiento.

Ajustes y mejoras basados en los resultados de las pruebas. *(16 de Mayo – 5 de Junio)*

Optimización del rendimiento de la aplicación.  
Mejora de la usabilidad y experiencia del usuario.  
Documentación y entrega. (6 de Junio – 12 de Junio)

Elaboración de documentación técnica y manuales de usuario.  
Preparación para la entrega final del proyecto. (13 de Junio – 18 de Junio)

## 2.6.- Estudio del coste del proyecto

Para el coste total del proyecto no se incluye el sueldo del trabajador, en caso de que cobrase por las horas, contando que habrán sido un total de 200 horas, serían unos 1600-2000€, incluyendo el gasto energético producido por el equipo informático utilizado.



VIABILIDAD FINANCIERA



VIABILIDAD ECONÓMICA





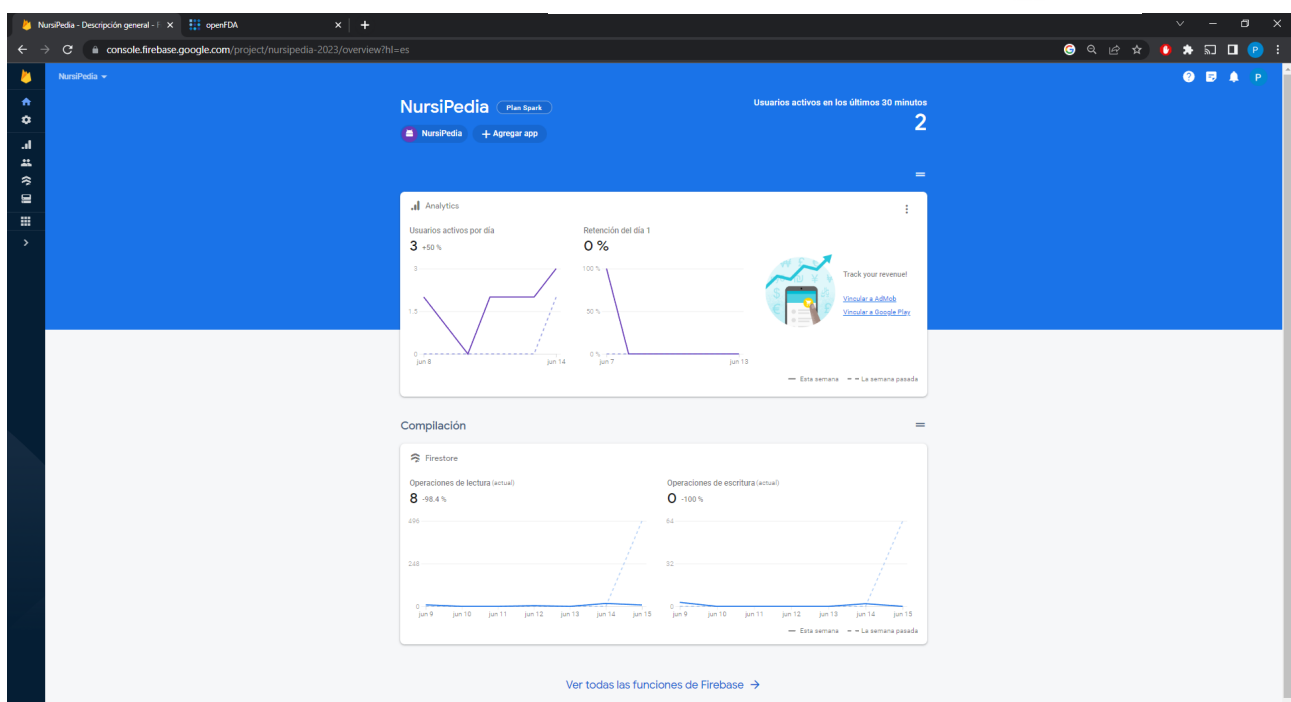
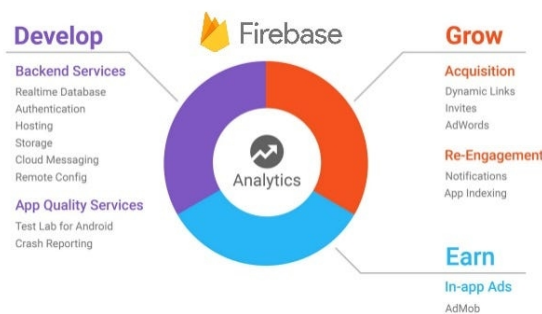
## ANÁLISIS DEL SISTEMA DE INFORMACIÓN

### 3.1.- Identificación del entorno tecnológico

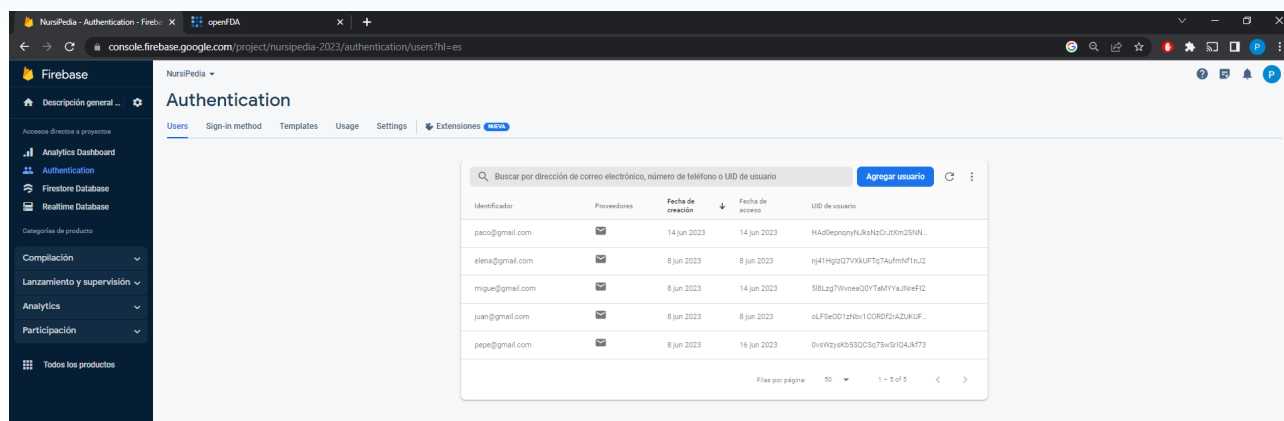
Como mencioné anteriormente, he usado Android Studio como el entorno de desarrollo integrado (IDE) principal para la creación de la aplicación, ya que proporciona un conjunto de herramientas y recursos específicos para el desarrollo de aplicaciones Android.

El lenguaje de programación que he usado ha sido Kotlin, ya que es un lenguaje moderno y claro, que se ha convertido en el lenguaje preferido para el desarrollo de aplicaciones Android, ya que ofrece mejoras en la productividad y la legibilidad del código.

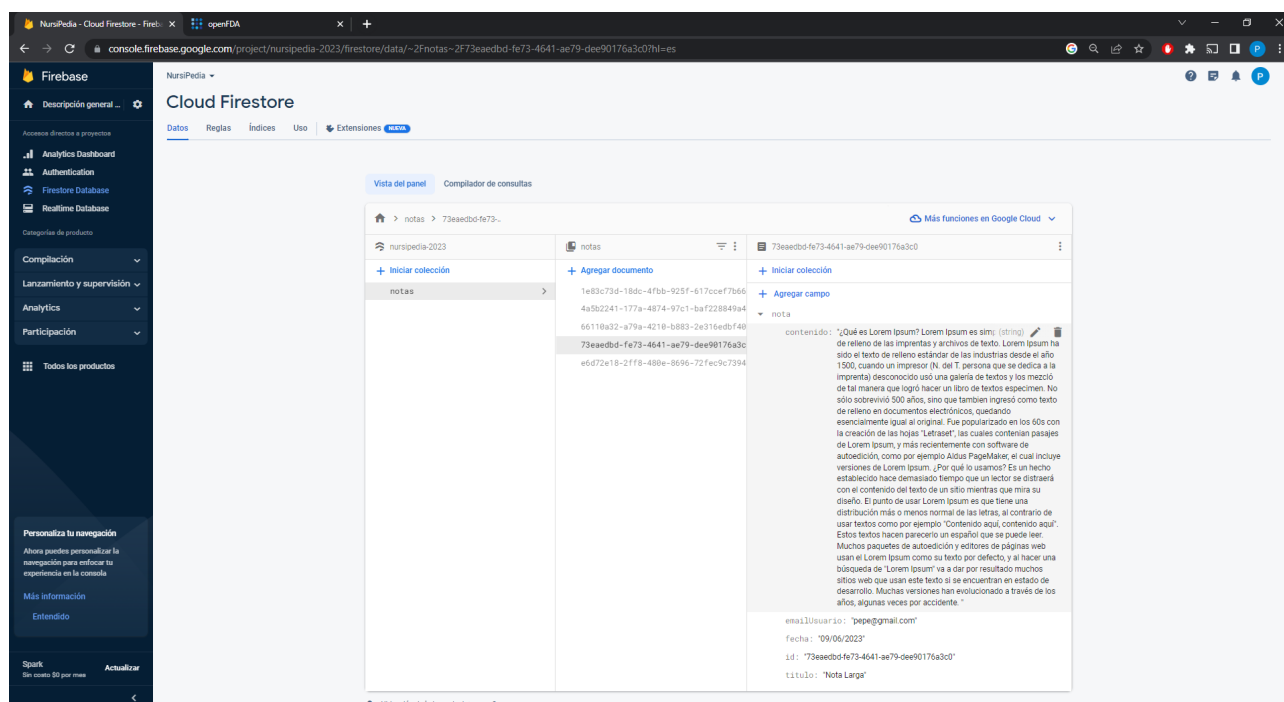
Para el control de bases de datos y autenticación de usuarios he usado Firebase, que es una plataforma de desarrollo móvil ofrecida por Google. Firebase proporciona una amplia gama de servicios, incluyendo almacenamiento en la nube, bases de datos en tiempo real, autenticación de usuarios, notificaciones push y más.



Como podemos ver en la siguiente captura, se muestran los usuarios creados hasta el momento en la aplicación junto a sus contraseñas cifradas, además nos permite eliminar y crear nuevos usuarios desde la propia interfaz de Firebase Auth.

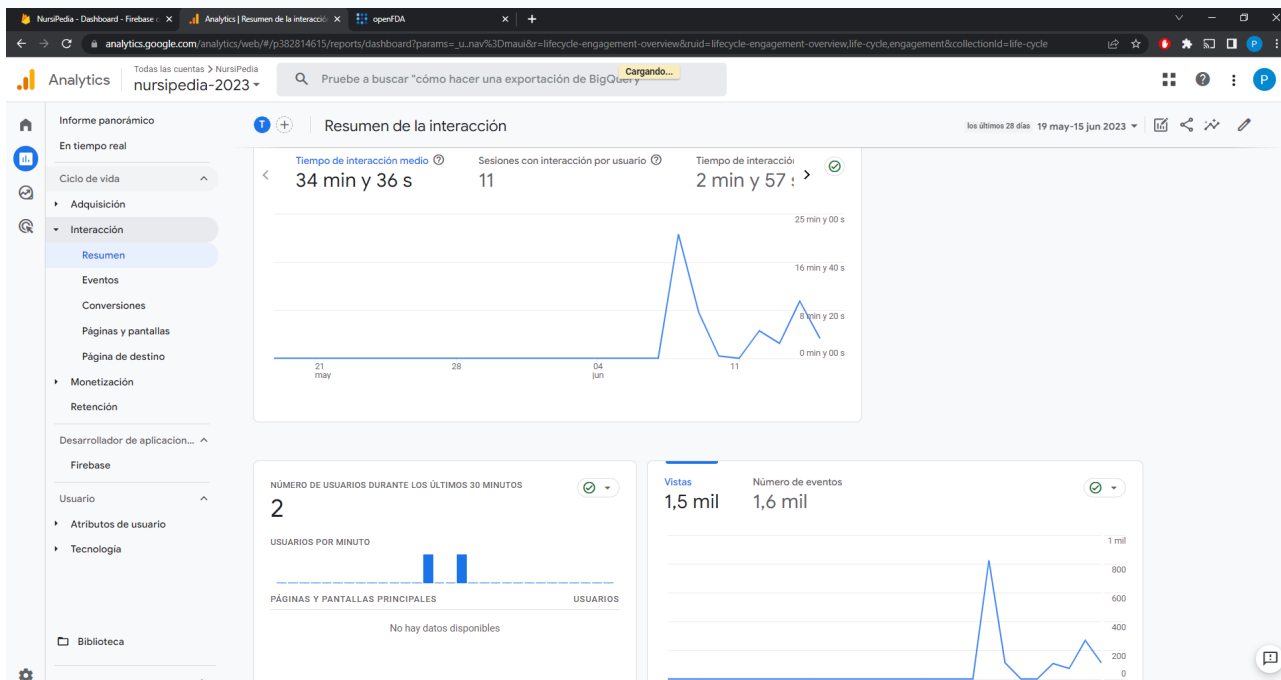


También nos ofrece la opción de guardar la información en forma de tablas, haciendo la función de base de datos en la nube, como en este caso, que nos permite crear una colección de notas en las que se guardarán todas las notas creadas por los diferentes usuarios que accedan a la aplicación, diferenciadas por su id.



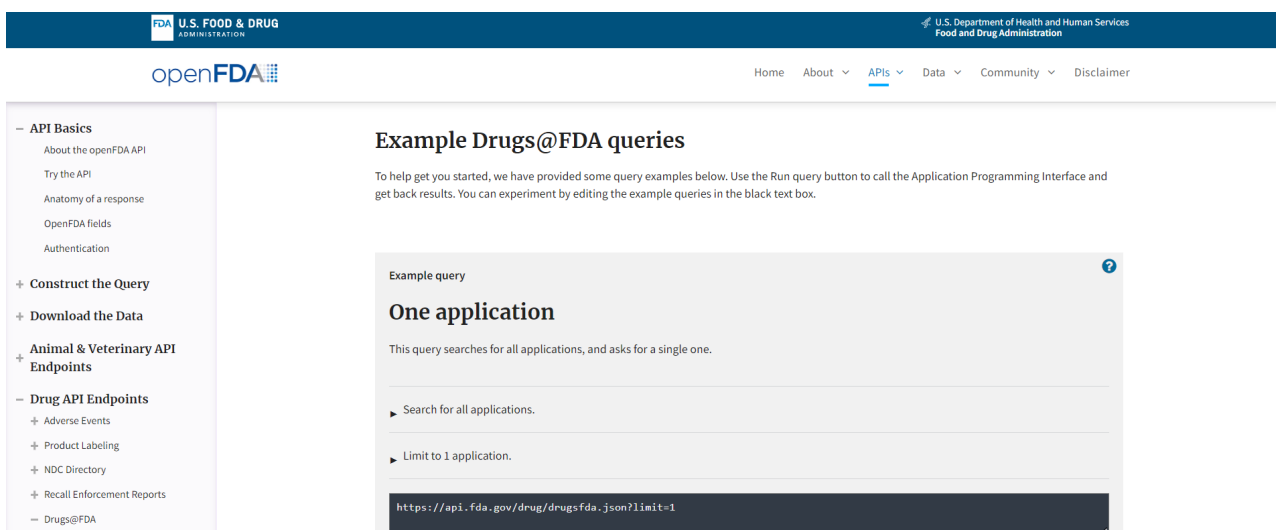
La parte del código donde realizamos todas estas subidas y bajadas de la base de datos la veremos más adelante en profundidad. Tanto en el caso de acceder o crear nuevos usuarios, como la de añadir o borrar nuevas notas.

También nos ofrece la posibilidad de ver las estadísticas de nuestra aplicación, el uso que le da cada usuario y el tiempo de interacción de cada uno de ellos, además de la localización desde la cual accede a la aplicación.



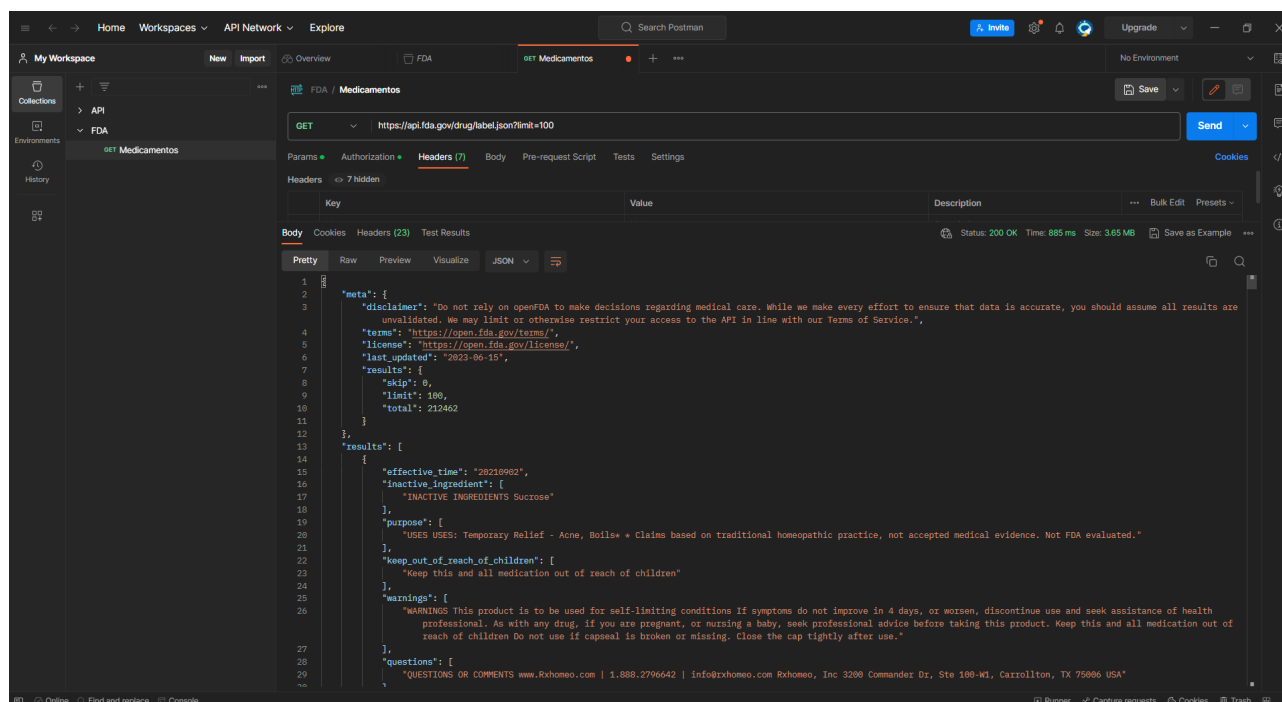
También he usado Retrofit, que es una biblioteca de cliente HTTP que se utiliza para realizar llamadas a API y comunicarse con servicios externos en NursiPedia. Retrofit simplifica la integración de API al proporcionar una interfaz sencilla para definir y realizar solicitudes HTTP.

La API utilizada es OpenFDA\* (<https://open.fda.gov/apis/>), API que contiene información sobre medicamentos entre otras cosas, con información detallada de características como su nombre comercial, las advertencias de uso o el propósito de dicho medicamento.



Otra de las herramientas utilizadas a destacar es Postman, que es una herramienta de desarrollo de API que he utilizado para probar y validar las llamadas a API en NursiPedia. Permite enviar solicitudes HTTP personalizadas, visualizar las respuestas y realizar pruebas de integración de API de manera eficiente.

Aquí un ejemplo de la llamada que he usado para recibir el listado de todos los medicamentos.



## 3.2.- Modelado de datos

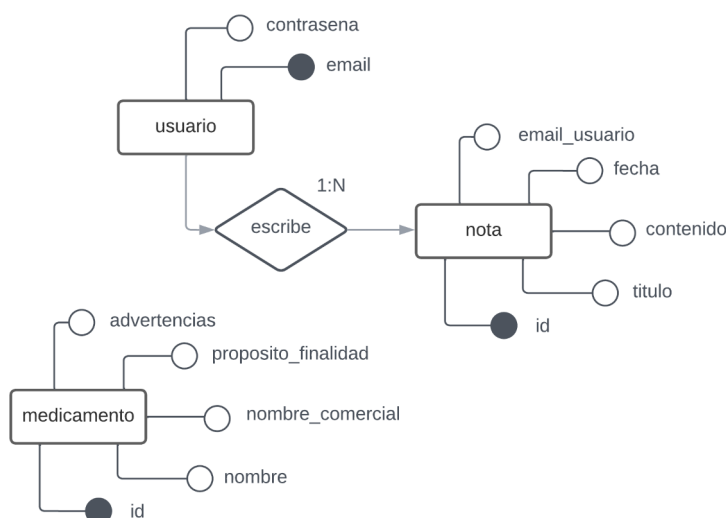
### 3.2.1.- Modelo Entidad-Relación

La base de datos de NursiPedia no es muy compleja, ya que la mayor parte de la información es obtenida de una API o cargada en Firebase Firestore, que se encarga de tratar los datos automáticamente.

Se pueden identificar las siguientes entidades y sus atributos:

Usuario:

- Email (Clave primaria)
- Contraseña



Medicamento:

- ID de medicamento (clave primaria)
- Nombre del medicamento
- Nombre comercial
- Propósito o finalidad terapéutica
- Advertencias

Notas:

- ID de nota (clave primaria)
- Título
- Contenido
- Fecha
- Email del usuario

### 3.1.2.- Esquema de la base de datos

Tabla "Usuario":

- email (clave primaria)
- contraseña

Tabla "Medicamento":

- id\_medimento (clave primaria)
- nombre
- nombre\_comercial
- proposito\_finalidad
- advertencias

Tabla "Notas":

- id (clave primaria)
- titulo
- contenido
- fecha
- email\_usuario (clave foránea)

<div>  &gt; notas &gt; 1e83c73d-18dc-         </div> <div>Más funciones en Google Cloud</div>		
nursipedia-2023	notas	1e83c73d-18dc-4fbb-925f-617ccef7b661
+ Iniciar colección	+ Agregar documento	+ Iniciar colección
notas >	1e83c73d-18dc-4fbb-925f-617ccef7b661 4a5b2241-177a-4874-97c1-baf228849a4 66110a32-a79a-4210-b883-2e316edbf40 73eaedbd-fe73-4641-ae79-dee90176a3c e6d72e18-2ff8-480e-8696-72fec9c7394	+ Agregar campo nota contenido: "Prueba" emailUsuario: "migue@gmail.com" fecha: "08/06/2023" id: "1e83c73d-18dc-4fbb-925f-617ccef7b661" titulo: "Nota de Migue"

### 3.1.3.- Datos de prueba

Tabla "Usuario":

- email: migue@gmail.com
- contraseña: Migue1234+

Tabla "Medicamento":

- id\_medimento: 1
- nombre\_medimento: Paracetamol
- nombre\_comercial: Tylenol
- proposito\_terapeutico: Alivio del dolor y fiebre
- advertencias: No exceder la dosis recomendada

Tabla "Notas":

- id\_nota: 1
- título: Reunión con el equipo médico
- contenido: Discutir el plan de tratamiento de un paciente
- fecha: 12-10-2016
- email\_usuario: [migue@gmail.com](mailto:migue@gmail.com)

Document ID	nota
1e83c73d-18dc-4fbb-925f-617ccef7b661	{emailUsuario: "migue@gmail.com", fecha: "08/06/2023", contenido: "Prueba", id: "1e83c73d-18dc-4fbb-925f-617ccef7b661", título: "Nota de Migue"}
4a5b2241-177a-4874-97c1-baf228849a4d	{título: "Nota de Elena", emailUsuario: "elena@gmail.com", id: "4a5b2241-177a-4874-97c1-baf228849a4d", fecha: "08/06/2023", contenido: "Prueba"}
66110a32-a79a-4210-b883-2e316edbf409	{fecha: "08/06/2023", emailUsuario: "juan@gmail.com", id: "66110a32-a79a-4210-b883-2e316edbf409", contenido: "Prueba", título: "Nota de Juan"}
73eae9bd-fe73-4641-ae79-dee90176a3c0	{emailUsuario: "pepe@gmail.com", id: "73eae9bd-fe73-4641-ae79-dee90176a3c0", título: "Nota Larga", contenido: "¿Qué es Lorem Ipsum? Lorem Ipsum es simplemente el texto de relleno de las imprentas y archivos de texto. Lorem Ipsum ha sido el texto de relleno estándar de las industrias desde el año 1500, cuando un impresor (N. del T. persona que se dedica a la imprenta) desconocido usó una galería de textos y los mezcló de tal manera que logró hacer un libro de textos especimen. No sólo sobrevivió 500 años, sino que también ingresó como texto de relleno en documentos electrónicos, quedando esencialmente igual al original. Fue popularizado en los 60s con la creación de las hojas "Letraset", las cuales contenían pasajes de Lorem Ipsum, y más recientemente con software de autoedición, como por ejemplo Aldus PageMaker, el cual incluye versiones de Lorem Ipsum. ¿Por qué lo usamos? Es un hecho establecido hace demasiado tiempo que un lector se distraerá con el contenido del texto de un sitio mientras que mira su diseño. El punto de usar Lorem Ipsum es que tiene una distribución más o menos normal de las letras, al contrario de usar textos como por ejemplo "Contenido aquí, contenido aquí". Estos textos hacen parecerlo un español que se puede leer. Muchos paquetes de autoedición y editores de páginas web usan el Lorem Ipsum como su texto por defecto, y al hacer una búsqueda de "Lorem Ipsum" va a dar por resultado muchos sitios web que usan este texto si se encuentran en estado de desarrollo. Muchas versiones han evolucionado a través de los años, algunas veces por accidente.", fecha: "09/06/2023"}
e6d72e18-2ff8-480e-8696-72fec9c7394d	{título: "Nota de Pepe", emailUsuario: "pepe@gmail.com", contenido: "Prueba", fecha: "08/06/2023", id: "e6d72e18-2ff8-480e-8696-72fec9c7394d"}

Items per page: 100 1 - 5 de 5

### 3.3.- Identificación de los usuarios participantes y finales

Los enfermeros son los usuarios principales de NursiPedia. Utilizan la aplicación para acceder a herramientas útiles, consultar información sobre medicamentos, realizar cálculos y guardar notas personalizadas. También pueden registrarse y autenticarse en la aplicación.

Los administradores del sistema, que son los usuarios encargados de administrar y mantener el sistema NursiPedia. Tienen permisos adicionales para gestionar usuarios, actualizar la base de datos de medicamentos y realizar tareas de mantenimiento.

Los desarrolladores, o como es el caso, el desarrollador, también ya que son responsables de construir y mantener el sistema, implementar nuevas funcionalidades y solucionar posibles problemas o errores.

También será necesario que alguno de los administradores tenga acceso a Firebase y Firebase Firestore para controlar el flujo de la aplicación y eliminar o crear nuevas cuentas en caso de que sea necesario.

### 3.4.- Identificación de subsistemas de análisis

Subsistema de Autenticación y Registro es el subsistema que se encarga de gestionar el proceso de registro y autenticación de usuarios. Permite a los enfermeros crear una cuenta en NursiPedia y acceder al sistema de forma segura.

Subsistema de Herramientas de Cálculo que es el subsistema integrado en la aplicación que proporciona las herramientas de cálculo necesarias para los enfermeros, como calculadoras de infusiones, cálculo del índice de masa corporal (IMC) y cálculo de gasto cardíaco. Permite a los usuarios realizar cálculos precisos de forma rápida y sencilla.

Subsistema de Gestión de Medicamentos que se encarga de almacenar y gestionar la información relacionada con los medicamentos. Permite a los enfermeros buscar medicamentos, acceder a su información detallada, como propósito terapéutico y advertencias, y mantener la base de datos actualizada.

Subsistema de Notas Personalizadas que permite a los enfermeros crear y gestionar notas personalizadas. Permite agregar títulos, escribir contenido, guardar fechas y establecer recordatorios para las notas.

### 3.5.- Establecimiento de requisitos

#### *Requisitos Funcionales:*

Permite a los enfermeros registrar y autenticarse en la aplicación.

Los usuarios pueden acceder a herramientas de cálculo, como calculadoras de infusiones y cálculo de índice de masa corporal (IMC).

Los enfermeros pueden buscar medicamentos y acceder a su información detallada.

El sistema permite la creación y gestión de notas personalizadas.

Los usuarios pueden recibir notificaciones de recordatorios para las notas establecidas.

#### *Requisitos de Usabilidad:*

La interfaz de usuario es intuitiva y fácil de usar para los enfermeros.

El sistema es compatible con dispositivos móviles Android.

#### *Requisitos de Seguridad:*

El sistema garantiza la confidencialidad de la información personal de los usuarios.

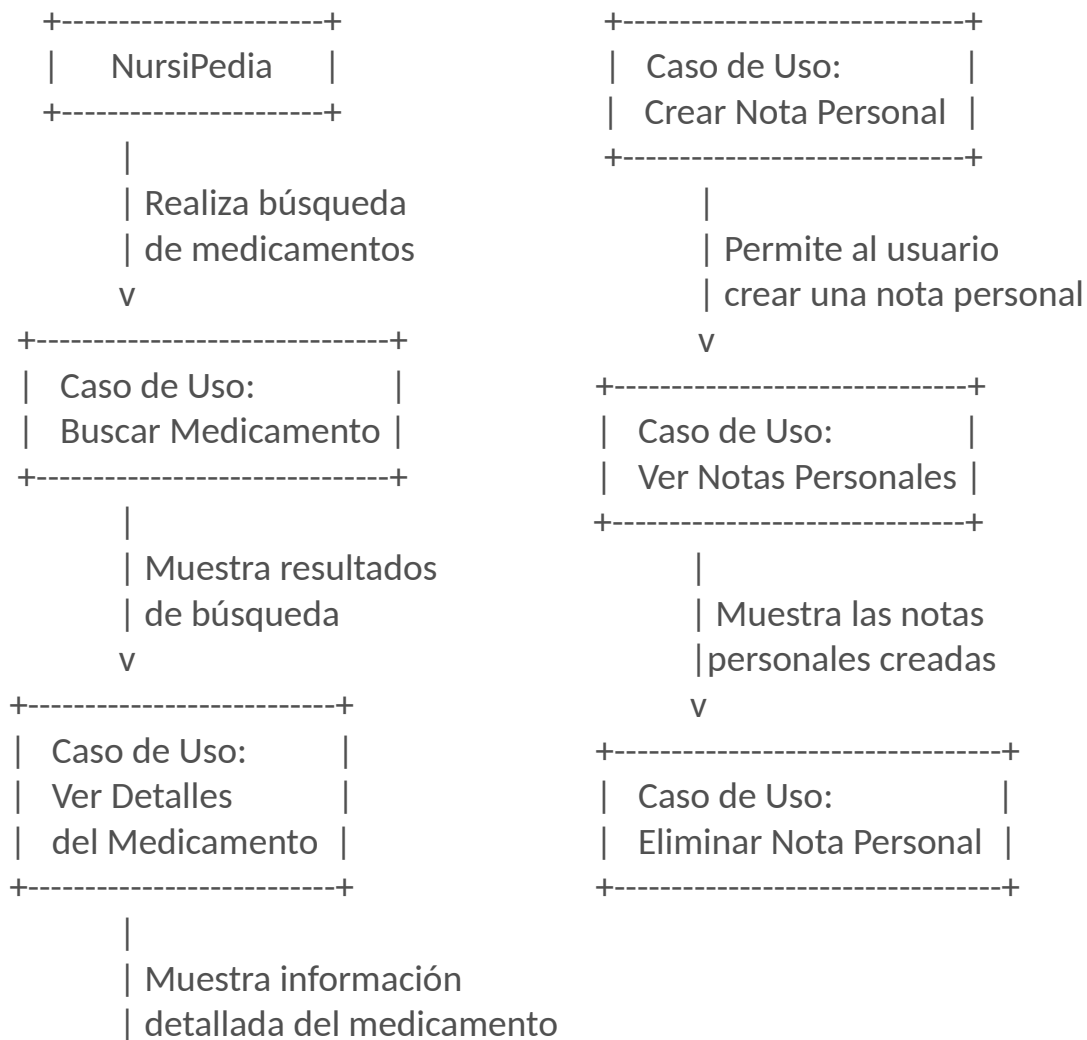
Se implementan medidas de seguridad para proteger los datos de los usuarios, como contraseñas encriptadas y comunicaciones seguras.

#### *Requisitos de Rendimiento:*

El sistema tiene tiempos de respuesta rápidos al buscar medicamentos o realizar cálculos.

La aplicación funciona sin problemas y sin bloqueos o errores frecuentes.

### 3.6.- Diagramas de Análisis



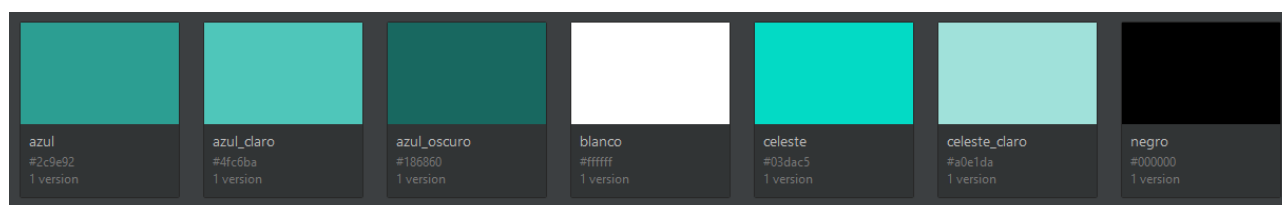


### 3.7.- Definición de interfaces de usuario

#### 3.7.1.- Especificación de principios generales de interfaz

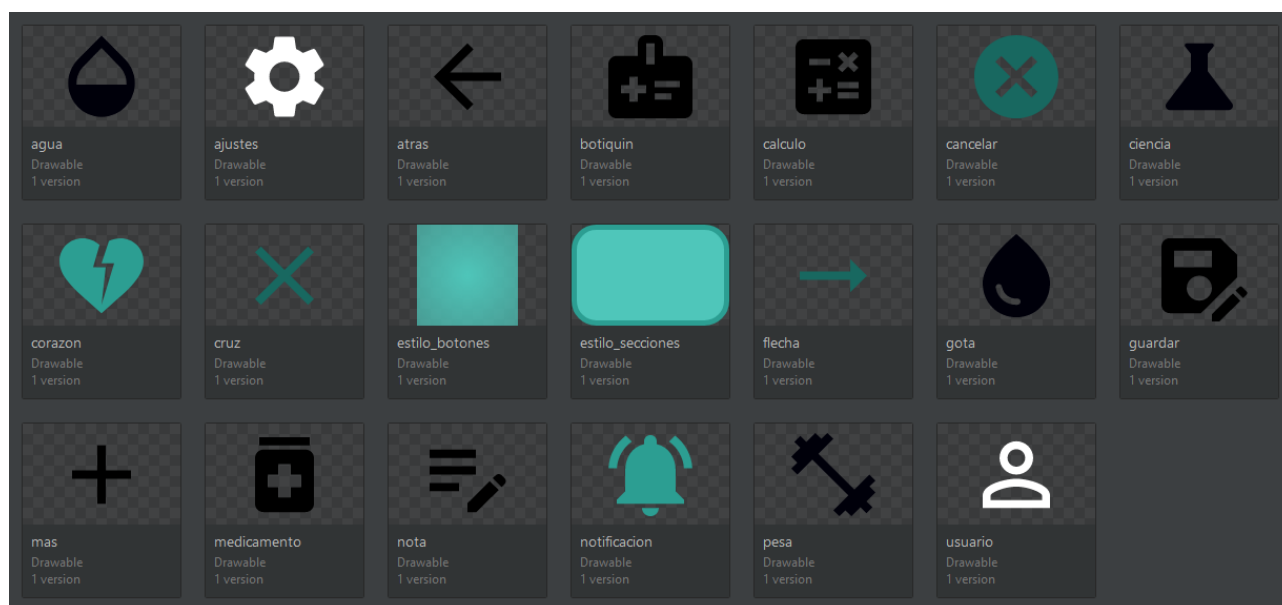
La interfaz es simple y fácil de entender, evitando elementos innecesarios y proporcionando una navegación clara.

Se mantiene una apariencia y comportamiento coherentes en todas las pantallas de la aplicación, utilizando los mismos elementos de diseño, colores y estilos.



La interfaz estará diseñada pensando en la comodidad y facilidad de uso del usuario, considerando aspectos como el tamaño de los botones, la legibilidad de los textos y la accesibilidad.

Se utilizan iconos y símbolos reconocibles y comprensibles para representar acciones y funciones, evitando confusiones y facilitando el uso de la aplicación.



Se proporciona retroalimentación visual al usuario cuando realice acciones, como cambios de estado de los botones o confirmaciones de operaciones.

Los elementos de la interfaz se organizan de manera lógica, facilitando la comprensión y navegación por la aplicación.

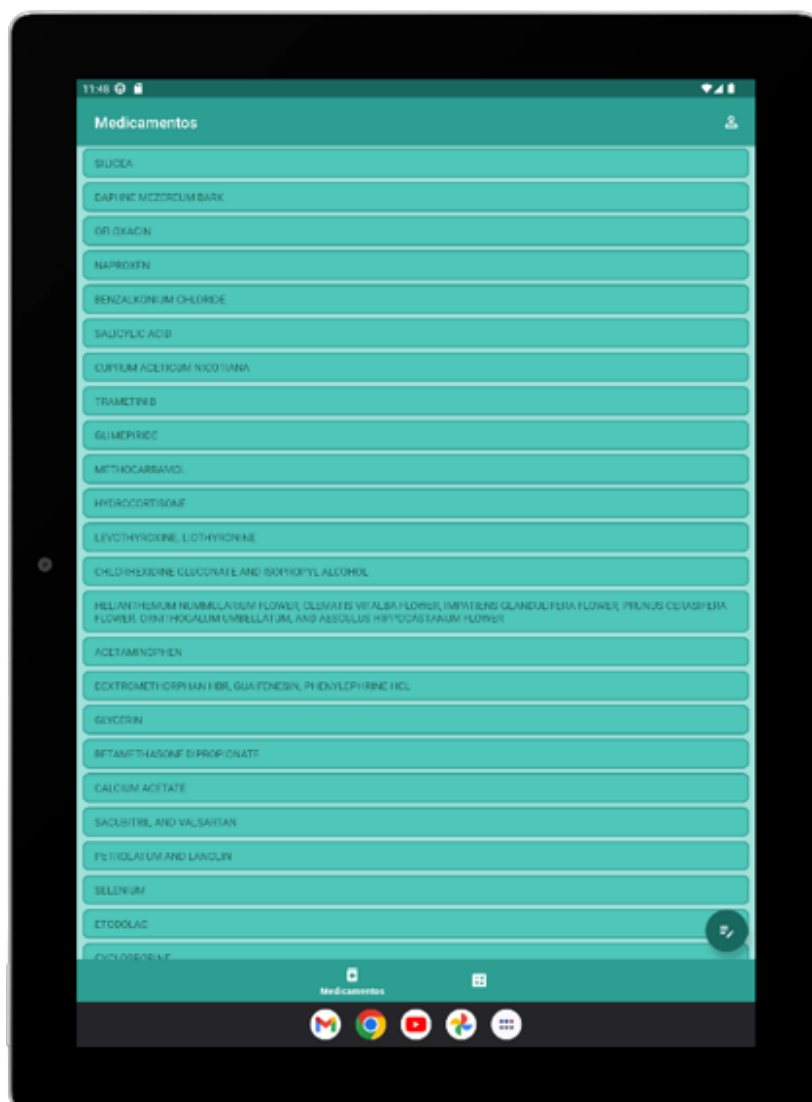
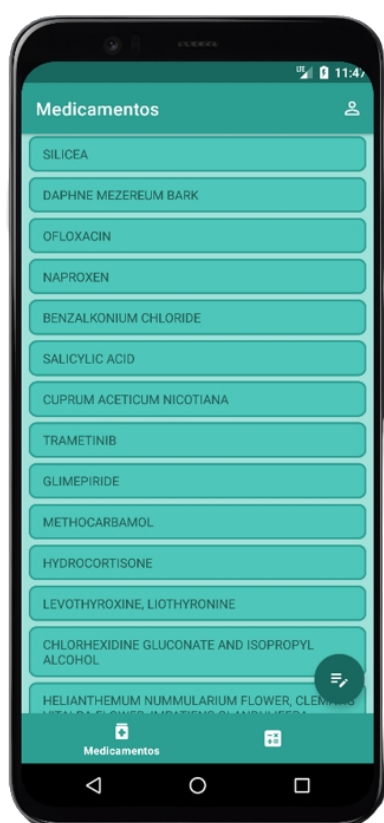
Menú Superior →



Menú Inferior →



La interfaz se adapta a diferentes tamaños de pantalla y dispositivos, asegurando una experiencia óptima tanto en teléfonos móviles como en tabletas.



### 3.7.2.- Especificación de formatos individuales de la interfaz de pantalla

*Pantalla de inicio de sesión:*

Campos de entrada para correo electrónico y contraseña, botón de inicio de sesión.  
Botón de registro.

*Pantalla de registro:*

Campos de entrada para correo electrónico, repetir email, contraseña y confirmación de contraseña, botón de registro y botón volver.

*Pantalla principal:*

Íconos de navegación para acceder a diferentes secciones de la aplicación, como "Medicamentos", "Calculadoras", "Notas" y "Perfil".

*Pantalla de información de cuenta:*

Email del usuario y botón de cerrar sesión.

*Pantalla de fórmulas:*

Botones con acceso a cada una de las fórmulas.

*Pantalla de medicamentos:*

Lista de resultados de medicamentos con nombres y nombres comerciales.  
Botón de "Notas".

*Pantalla de detalle del medicamento:*

Nombre del medicamento, nombre comercial, propósito terapéutico, advertencias.

*Pantalla de notas:*

Lista de notas personales con títulos, breve descripción de los contenidos y opción eliminar cada nota.

*Pantalla de detalle de la nota:*

Título, contenido, fecha de creación y botón de recordatorio.

*Pantalla de infusiones:*

Volumen a infundir, velocidad de infusión, horas, fórmula y botón calcular.

*Pantalla de disoluciones:*

Cantidad de medicamento, volumen de la disolución, concentración de la disolución, fórmula y botón calcular.

*Pantalla de pérdidas insensibles:*

Desplegable con tres opciones según el factor de pérdidas insensibles, peso, pérdidas insensibles, fórmula y botón calcular.

*Pantalla de IMC:*

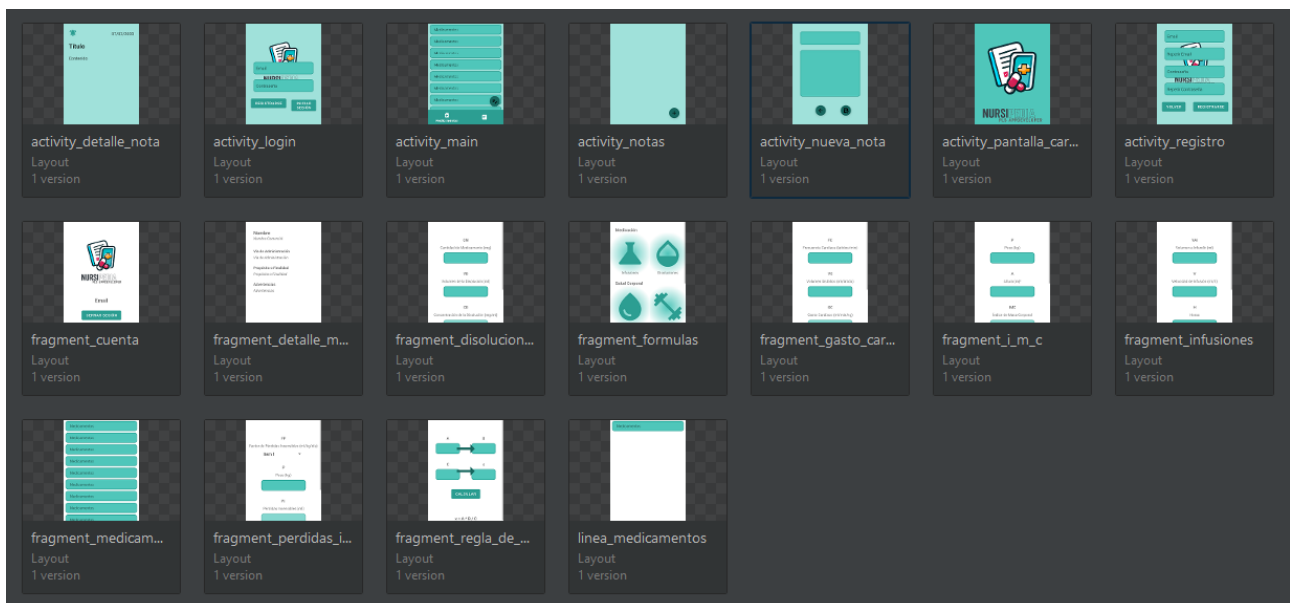
Peso, altura, índice de masa corporal, fórmula y botón calcular.

*Pantalla de regla de tres:*

Peso, altura, índice de masa corporal, fórmula y botón calcular.

*Pantalla de gasto cardíaco:*

Frecuencia cardíaca, volumen sistólico, gasto cardíaco, fórmula y botón calcular.



### 3.7.3.- Identificación de perfiles de usuario

Anteriormente se mencionó los perfiles de usuarios que pueden acceder a NursiPedia, todos estos perfiles tienen el mismo acceso a la aplicación, pero el desarrollador tiene acceso al código de la misma y el administrador tiene acceso a la base de datos a través de Firebase.

### 3.7.4.- Especificación de formatos de impresión

La aplicación no genera ningún documento externo, ya que todo se genera en la nube, por ello es Firebase el encargado de controlar éstos datos o ficheros.

Sin embargo, en el caso de las notas, trabajamos con Gson y Json, al igual que las llamadas recibidas de la API.

### 3.7.5.- Especificación de la navegabilidad entre pantallas

#### *Pantalla de inicio de sesión:*

Desde esta pantalla, los usuarios pueden acceder a la pantalla de registro si no tienen una cuenta. Después de iniciar sesión correctamente, se dirigen a la pantalla principal de la aplicación.

#### *Pantalla de registro:*

Los usuarios pueden registrarse en la aplicación ingresando su correo electrónico y contraseña. También tienen la opción de volver a la pantalla de inicio de sesión si ya tienen una cuenta.

#### *Pantalla principal:*

En esta pantalla, los usuarios tienen acceso a los diferentes módulos de la aplicación, como "Medicamentos", "Calculadoras", "Notas" y "Perfil".

Al hacer clic en cada módulo, se abrirá la pantalla correspondiente.

Pantallas de módulos específicos:

Cada módulo tiene su propia pantalla con funcionalidades específicas.

Los usuarios pueden navegar entre estas pantallas utilizando los botones o enlaces proporcionados en cada pantalla.

Desde la pantalla de "Medicamentos", los usuarios pueden hacer clic en un medicamento específico para ver más detalles en la pantalla de "Detalle del Medicamento".

#### *Pantalla de información de cuenta:*

Los usuarios pueden acceder a esta pantalla desde la pantalla principal para ver información sobre su cuenta, como el correo electrónico asociado.

También pueden cerrar sesión desde esta pantalla.

#### *Pantalla de notas:*

Desde la pantalla principal, con la sección de medicamentos seleccionada, los usuarios pueden acceder a la sección de "Notas".

En esta pantalla, se muestra una lista de las notas personales creadas por el usuario.

El usuario también podrá pulsar el botón para eliminar cualquiera de las notas.

#### *Pantalla de crear una nueva nota:*

Los usuarios pueden crear una nueva nota haciendo clic en el botón correspondiente en la pantalla de notas.

Al hacerlo, se abrirá una nueva pantalla donde podrán ingresar un título, contenido de la nota.

#### *Detalle de la nota:*

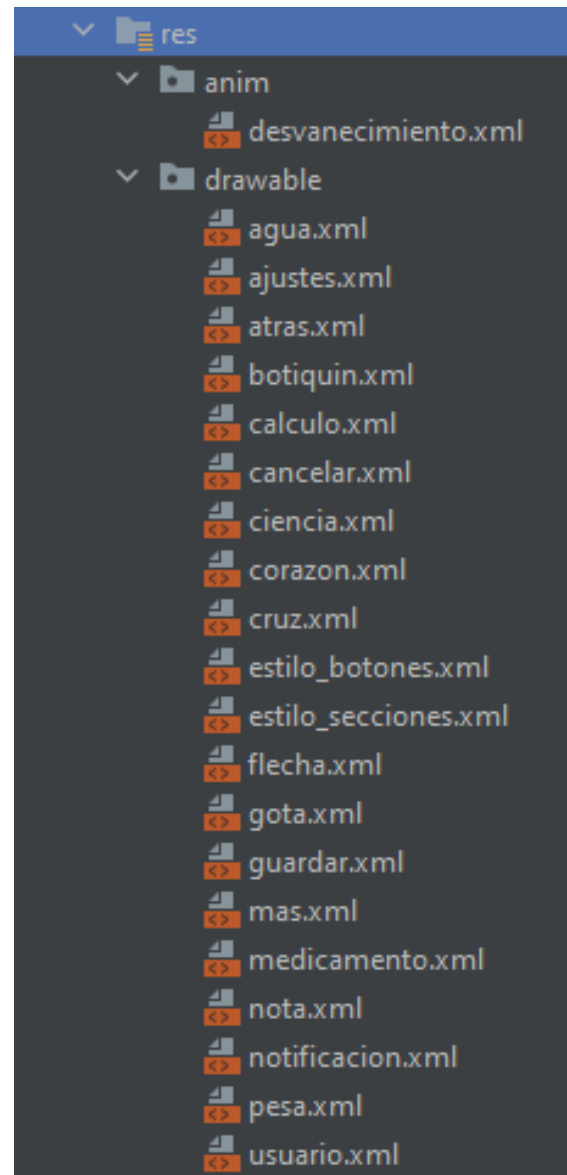
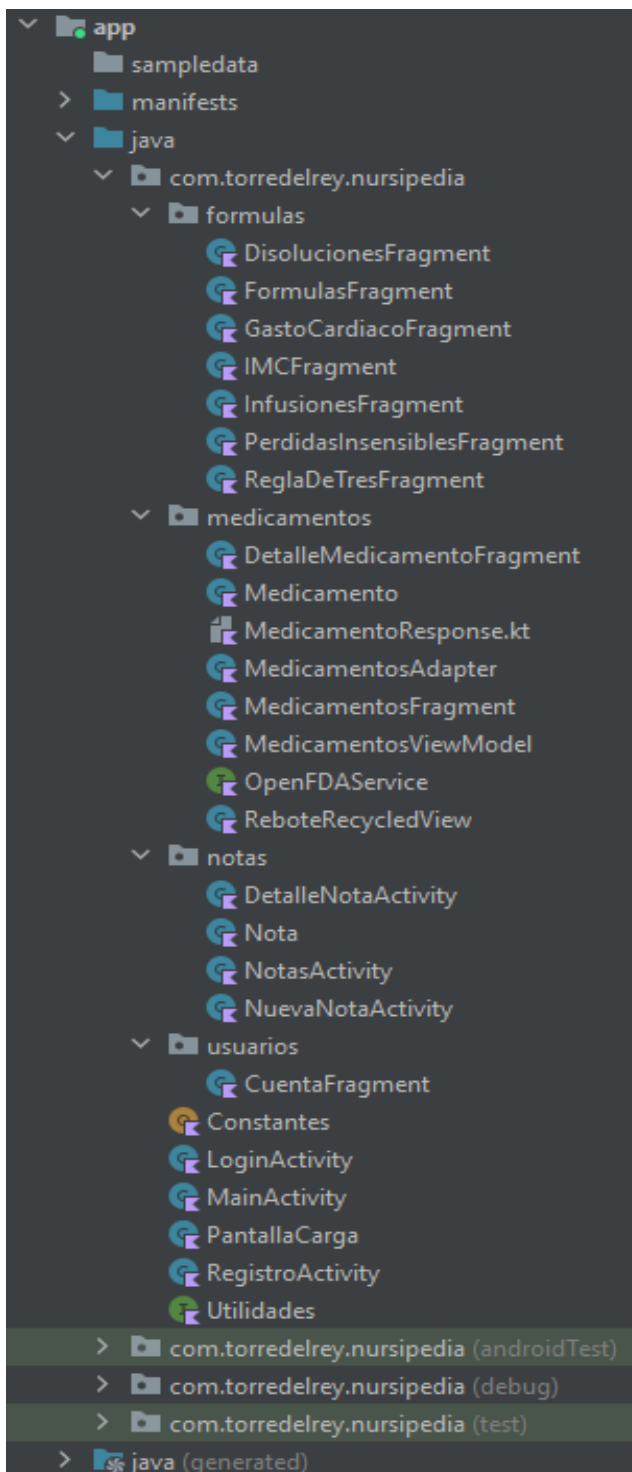
Al hacer clic en una nota de la lista, se abrirá una pantalla de detalle de la nota.

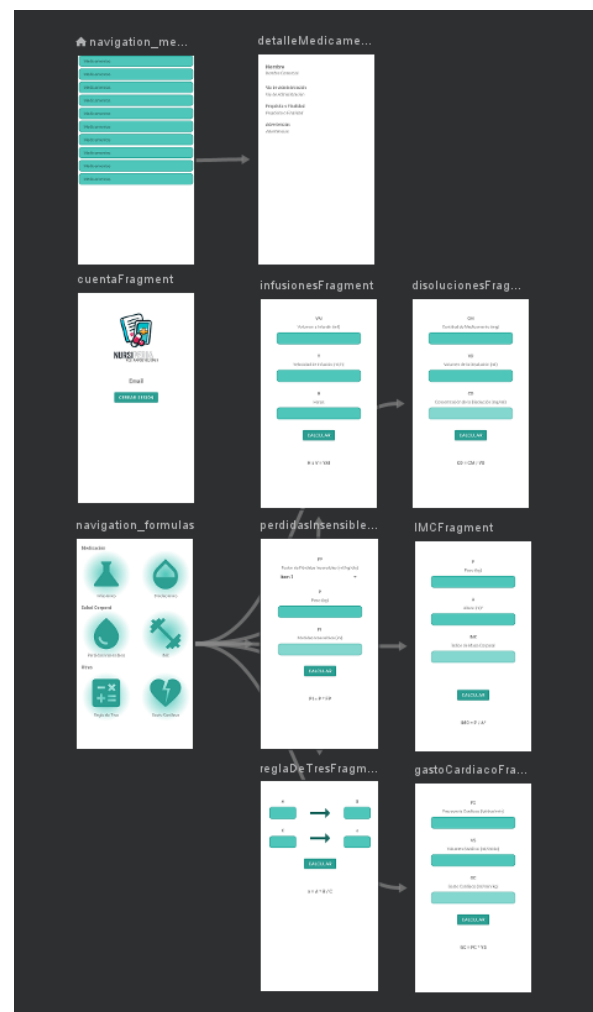
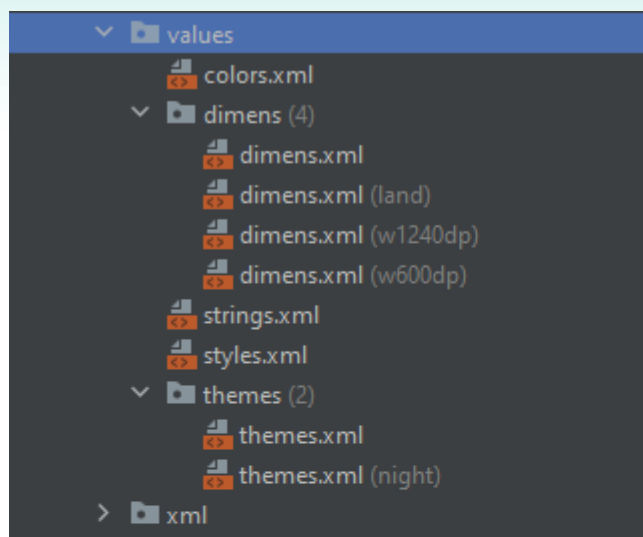
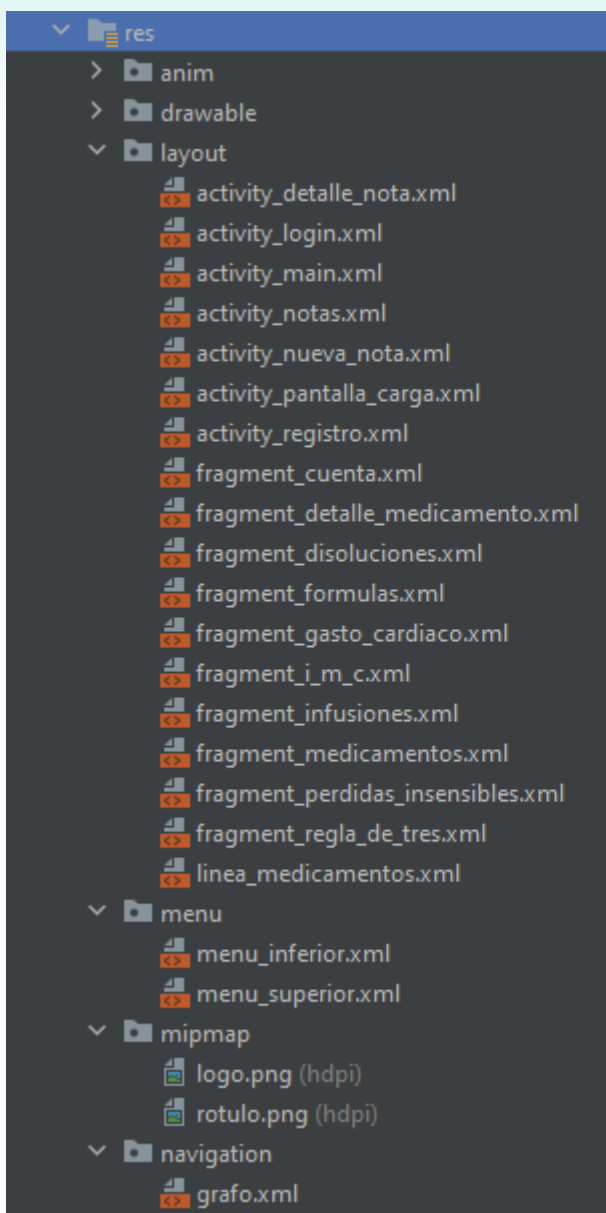
En esta pantalla, se muestra el título, contenido y fecha de creación de la nota.

Los usuarios también tienen la opción de configurar un recordatorio para la nota.

## CONSTRUCCIÓN/COMPILACIÓN DEL SISTEMA

Primero voy a mostrar la estructura de carpetas dentro de Android Studio, además de los archivos creados.





Éste es el grafo utilizado para la navegación entre pantallas.

Una vez mostrado la organización de los archivos, nos detendremos a explicar detenidamente algunos fragmentos de código importantes.

Empezaré con los diferentes formatos utilizados en la aplicación, en especial estos dos, que nos permiten controlar que el email y la contraseña cumplan los requisitos típicos que exigen para su creación en cualquier página web.

```
// Formatos
const val FORMATO_EMAIL = "^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*\\.\\w{2,}+$"
const val FORMATO_CONTRASENA = "^(?=.*[A-Z])(?=.*[a-z])(?=.*\\d)(?=.*[\\W_]).{8,}$"
```

'^': Indica que la coincidencia debe comenzar al inicio de la cadena.

'\\w+': Coincide con uno o más caracteres alfanuméricos (letras, dígitos o guiones bajos).

'[\\.-]?': Coincide opcionalmente con un punto o un guion medio.

'\\w+': Coincide con uno o más caracteres alfanuméricos.

'([\\.-]?\\w+)\*': Permite que se repita la combinación de un punto o guion medio seguido de uno o más caracteres alfanuméricos.

'@': Coincide con el carácter "@".

'\\w+': Coincide con uno o más caracteres alfanuméricos.

'([\\.-]?\\w+)\*': Permite que se repita la combinación de un punto o guion medio seguido de uno o más caracteres alfanuméricos.

'(\\.\\w{2,})+': Coincide con uno o más grupos de un punto seguido de al menos dos caracteres alfanuméricos. Esto se utiliza para representar el dominio, como ".com", ".net", etc.

'\$': Indica que la coincidencia debe llegar al final de la cadena.

'^': Indica que la coincidencia debe comenzar al inicio de la cadena.

'(?=.\*[A-Z])': Utiliza un lookahead para verificar que haya al menos una letra mayúscula en la cadena.

'(?=.\*[a-z])': Utiliza un lookahead para verificar que haya al menos una letra minúscula en la cadena.

'(?=.\*\\d)': Utiliza un lookahead para verificar que haya al menos un dígito en la cadena.

'(?=.\*[\\W\_])': Utiliza un lookahead para verificar que haya al menos un carácter especial o guion bajo en la cadena.

'.{8,}': Coincide con cualquier carácter y asegura que la longitud de la cadena sea de al menos 8 caracteres.

'\$': Indica que la coincidencia debe llegar al final de la cadena.



El siguiente fragmento de código que voy a mostrar es el proceso que se ejecuta al iniciar sesión.

```
private fun login(email: String, contrasena: String) {  
    // Función signIn de la clase FirebaseAuth  
    auth.signInWithEmailAndPassword(email, contrasena).addOnCompleteListener(this) { task ->  
        if (task.isSuccessful) {  
            Toast.makeText(this, Constantes.MENSAJE_INICIO_SESION_EXITOSO, Toast.LENGTH_SHORT).show()  
  
            // SharedPreferences  
            val sharedPreferences: SharedPreferences = getSharedPreferences(Constantes.CLAVE_PREFERENCIAS,  
Context.MODE_PRIVATE)  
            val editor: SharedPreferences.Editor = sharedPreferences.edit()  
            editor.putString(Constantes.CLAVE_EMAIL, email)  
            editor.putBoolean(Constantes.CLAVE_SESION_INICIADA, true)  
            editor.apply()  
  
            // Navegación a la pantalla de Inicio  
            val intent = Intent(this, MainActivity::class.java)  
            startActivity(intent)  
            finish()  
        } else {  
            Toast.makeText(this, Constantes.MENSAJE_ERROR_INICIO_SESION, Toast.LENGTH_SHORT).show()  
        }  
    }  
}
```

Este código utiliza el método `signInWithEmailAndPassword` de Firebase Authentication para intentar iniciar sesión con la dirección de correo electrónico y contraseña proporcionados. Esta función devuelve un objeto `Task` que representa el resultado de la operación de inicio de sesión.

El método `addOnCompleteListener` se utiliza para agregar un listener al `Task` que se creó en el paso anterior. El bloque de código dentro de las llaves se ejecutará cuando la tarea se complete, independientemente de si fue exitosa o no.

Luego verificamos si la tarea de inicio de sesión fue exitosa o no. Si `task.isSuccessful` es `true`, significa que el inicio de sesión fue exitoso y se realiza lo siguiente.

Mostramos un mensaje.

Se utiliza `SharedPreferences` para almacenar información del usuario, como el correo electrónico y el estado de sesión iniciada.

Luego navegamos a la siguiente pantalla.

Usamos `finish()` para cerrar la actividad actual y evitar que el usuario pueda volver atrás.

Si la tarea de inicio de sesión no fue exitosa (`task.isSuccessful` es `false`), se muestra un mensaje de error de inicio de sesión.

Continuamos con el registro de usuario.

```
private fun registrarUsuario(email: String, password: String) {  
    // Función createUser de la clase FirebaseAuth  
    auth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(this) { task ->  
        if (task.isSuccessful) {  
            Toast.makeText(this, Constantes.MENSAJE_REGISTRO_EXITOSO, Toast.LENGTH_SHORT).show()  
  
            // Navegación a la pantalla de Login  
            val intent = Intent(this, LoginActivity::class.java)  
            startActivity(intent)  
            finish()  
        } else {  
            Toast.makeText(this, Constantes.MENSAJE_ERROR_REGISTRO, Toast.LENGTH_SHORT).show()  
        }  
    }  
}
```

Se utiliza el método createUserWithEmailAndPassword de FirebaseAuth para crear un nuevo usuario en Firebase Authentication utilizando la dirección de correo electrónico y la contraseña proporcionadas. Esta función devuelve un objeto Task que representa el resultado de la operación de registro.

Luego se agrega un listener al Task que se creó en el paso anterior utilizando addOnCompleteListener. El bloque de código dentro de las llaves se ejecutará cuando la tarea se complete, independientemente de si fue exitosa o no.

Luego, al igual que en el ejemplo anterior, se verifica si la tarea de registro fue exitosa o no. Si task.isSuccessful es true, significa que el registro fue exitoso y se realiza lo siguiente:

Se muestra un mensaje de éxito.

Navegamos a la pantalla de inicio de sesión.

Se llama a finish() para cerrar la actividad actual y evitar que el usuario pueda volver atrás.

Si la tarea de registro no fue exitosa (task.isSuccessful es false), se muestra un mensaje de error.

Finalizamos con el código de autenticación con la función de cerrar sesión.

```
override fun cerrarSesion() {  
    // SharedPreferences  
    val sharedPreferences: SharedPreferences = getSharedPreferences(Constants.CLAVE_PREFERENCIAS,  
Context.MODE_PRIVATE)  
    val editor: SharedPreferences.Editor = sharedPreferences.edit()  
  
    // Borrar SharedPreferences  
    editor.clear()  
    editor.apply()  
  
    // Cerrar sesión del usuario actual  
    val auth = FirebaseAuth.getInstance()  
  
    // Función signOut de la clase FirebaseAuth  
    auth.signOut()  
    Toast.makeText(applicationContext, Constantes.MENSAJE_SESION_CERRADA, Toast.LENGTH_SHORT).show()  
  
    // Navegación a pantalla Login  
    val intent = Intent(this, LoginActivity::class.java)  
    startActivity(intent)  
    finish()  
}
```

Obtenemos una instancia de SharedPreferences para acceder a las preferencias compartidas de la aplicación y un editor a través del método edit() para realizar cambios en las preferencias.

Borramos todos los datos almacenados en SharedPreferences mediante el método clear() del editor.

Aplicamos los cambios realizados en SharedPreferences mediante el método apply() del editor.

Obtenemos una instancia de FirebaseAuth para acceder al servicio de autenticación de Firebase.

Utilizamos el método signOut() de la clase FirebaseAuth para cerrar la sesión del usuario actual.

Mostramos un mensaje.

Volvemos a navegar a la pantalla de inicio de sesión.

Se finaliza la actividad actual llamando al método finish(), lo que garantiza que la actividad actual se cierre después de iniciar la nueva actividad.

El siguiente código que vamos a ver es el de la pantalla de carga.

```
Handler(Looper.getMainLooper()).postDelayed( {  
    if (sesionIniciada) {  
        // Muestra la pantalla Principal  
        val intent = Intent(this, MainActivity::class.java)  
        startActivity(intent)  
    } else {  
        // Muestra la pantalla de Login  
        val intent = Intent(this, LoginActivity::class.java)  
        startActivity(intent)  
    }  
    // Animación de desvanecimiento creada y guardada en la carpeta anim  
    overridePendingTransition(R.anim.desvanecimiento, R.anim.desvanecimiento)  
    finish()  
}, 2000)
```

Creamos una instancia de Handler pasando `Looper.getMainLooper()` como argumento. Esto asegura que el Handler se ejecute en el hilo principal de la interfaz de usuario (UI). Luego, llamamos al método `postDelayed` para ejecutar un bloque de código después de un retraso de 2000 milisegundos (2 segundos).

Verificamos el valor de la variable `sesionIniciada`. Si es `true`, significa que la sesión está iniciada e iniciamos la aplicación con el usuario ya iniciado, entrando directamente a la pantalla principal.

Si `sesionIniciada` es `false`, significa que la sesión no está iniciada y accedemos a la pantalla de inicio de sesión.

Luego aplicamos una transición personalizada entre actividades. `R.anim.desvanecimiento` hace referencia a una animación definida en los recursos del proyecto.

Por ultimo se llama al método `finish()` para finalizar la actividad actual y evitar que el usuario pueda volver atrás.

Lo siguiente que vamos a ver es el tratamiento de las notas en Firebase Firestore.

Lo primero que veremos es la creación de una nota.

```
db.collection(Constants.CLAVE_NOTAS).document(notaNueva.id).set(hashMapOf(Constants.CLAVE_NOTA to notaNueva))
```

Primero accedemos a la colección de notas en Firebase Firestore utilizando la clave “notas”, que representa el nombre de la colección.

Luego se especifica el documento en la colección al que se desea acceder.  
notaNueva.id es el ID de la nueva nota que se quiere guardar. Esto crea una referencia al documento específico en la colección.

Establecemos los datos de la nueva nota en el documento. Utilizamos hashMapOf para crear un objeto HashMap que contiene un par clave-valor.

Constants.CLAVE\_NOTA (“nota”) es la clave que representa el campo en el documento que almacenará los datos de la nota.

notaNueva es la instancia de la clase Nota que contiene los valores de la nueva nota.  
Se guarda como el valor asociado a la clave “nota” en el HashMap.

Lo siguiente que vamos a ver es la carga de las notas al entrar en la pantalla notas, obteniendo estas notas de la base de datos.

```
private fun cargarNotasDeFirebase(email: String) {  
    // Obtenemos una instancia de Firebase y realizamos una consulta de la tabla nota filtrando por email  
    val db = FirebaseFirestore.getInstance()  
    db.collection(Constants.CLAVE_NOTAS)  
        .whereEqualTo(Constants.CLAVE_NOTA_EMAIL_USUARIO, email)  
        .get()  
        .addOnSuccessListener { querySnapshot ->  
            for (document in querySnapshot.documents) {  
                val notaData = document.get(Constants.CLAVE_NOTA) as HashMap<*, *>?  
                val nota = notaData?.let { Nota().apply { fromHashMap(it) } }  
                if (nota != null) {  
                    crearCelda(nota)  
                }  
            }  
        }  
        .addOnFailureListener {  
            // En caso de no encontrar notas nos aparece un mensaje  
            Toast.makeText(applicationContext, Constants.MENSAJE_ERROR_CARGAR_NOTAS,  
                Toast.LENGTH_SHORT).show()  
        }  
}
```

Obtenemos una instancia de FirebaseFirestore utilizando FirebaseFirestore.getInstance().

Accedemos a la colección de notas en Firestore utilizando db.collection(Constants.CLAVE\_NOTAS), donde Constants.CLAVE\_NOTAS es la clave que representa el nombre de la colección de notas.

Realizamos una consulta en la colección filtrando las notas por el campo Constants.CLAVE\_NOTA\_EMAIL\_USUARIO, que debe ser igual al correo electrónico especificado.

Llamamos al método get() para obtener los resultados de la consulta.

Agregamos un OnSuccessListener para manejar el caso en que la consulta sea exitosa. Dentro de este bloque de código, se itera sobre los documentos devueltos en querySnapshot.documents.

Si notaData no es nulo, se crea una nueva instancia de Nota y se invoca la función fromHashMap para asignar los valores de notaData a los atributos de la instancia de Nota. Esto se realiza mediante la función de extensión apply.

Si se crea correctamente la instancia de Nota, se llama a la función crearCelda(nota) para crear una representación visual de la nota en la interfaz de usuario.

En caso de que ocurra un error durante la consulta, se agrega un OnFailureListener para mostrar un mensaje de error.

Lo siguiente será eliminar una nota al pulsar el botón.

```
private fun eliminarNotaDeFirebase(nota: Nota) {  
    // Filtramos por el id de la nota en la tabla notas de FirebaseFirestore y la borramos  
    val db = FirebaseFirestore.getInstance()  
    val notasRef = db.collection(Constants.CLAVE_NOTAS)  
    notasRef.whereEqualTo(Constants.CLAVE_NOTA_ID, nota.id)  
        .get()  
        .addOnSuccessListener { querySnapshot ->  
            for (document in querySnapshot.documents) {  
                document.reference.delete()  
                .addOnSuccessListener {  
                    // Si funciona correctamente se elimina la nota de la tabla notas en FirebaseFirestore  
                    Toast.makeText(applicationContext, Constantes.MENSAJE_NOTA_ELIMINADA,  
                        Toast.LENGTH_SHORT).show()  
                }  
                .addOnFailureListener {  
                    // En caso de no poder eliminar la nota nos muestra un mensaje y no ocurre nada en  
                    // FirebaseFirestore  
                    Toast.makeText(applicationContext, Constantes.MENSAJE_ERROR_ELIMINAR_NOTA,  
                        Toast.LENGTH_SHORT).show()  
                }  
            }  
        }  
        .addOnFailureListener {  
            // En caso de no encontrar la nota nos muestra un mensaje y no ocurre nada en FirebaseFirestore  
            Toast.makeText(applicationContext, Constantes.MENSAJE_BUSCAR_NOTA,  
                Toast.LENGTH_SHORT).show()  
        }  
}
```

Accedemos a la colección de notas en Firestore utilizando `db.collection(Constants.CLAVE_NOTAS)`, donde `Constants.CLAVE_NOTAS` es la clave que representa el nombre de la colección de notas.

Realizamos una consulta en la colección filtrando las notas por el campo `Constants.CLAVE_NOTA_ID`, que debe ser igual al ID de la nota que se desea eliminar.

Llamamos al método `get()` para obtener los resultados de la consulta.

Para cada documento, se llama al método `delete()` en `document.reference` para eliminar el documento de Firestore.

Agregamos un `OnSuccessListener` para manejar el caso en que la eliminación sea exitosa. Dentro de este bloque de código, se muestra un mensaje de éxito y se informa al usuario que la nota ha sido eliminada.

En caso de que ocurra un error durante la eliminación, se agrega un `OnFailureListener` para mostrar un mensaje de error y se informa al usuario que ha ocurrido un error al intentar eliminar la nota.

La función recordatorio hace lo siguiente.

```
private fun mostrarRecordatorio(notaNueva: Nota) {
    val notificationId = 1
    val channelId = Constantes.CANAL_RECORDATORIO

    // Hora Actual
    val calendar = Calendar.getInstance()
    val horaActual = calendar.get(Calendar.HOUR_OF_DAY)
    val minutoActual = calendar.get(Calendar.MINUTE)

    // TimePickerDialog
    val timePickerDialog = TimePickerDialog(
        this,
        { _: TimePicker, hora: Int, minuto: Int ->
            // Calcula el tiempo hasta que se tenga que mostrar el recordatorio
            val tiempoActual = System.currentTimeMillis()
            val calendarRecordatorio = Calendar.getInstance().apply {
                set(Calendar.HOUR_OF_DAY, hora)
                set(Calendar.MINUTE, minuto)
                set(Calendar.SECOND, 0)
                set(Calendar.MILLISECOND, 0)
            }

            val tiempoRecordatorio = calendarRecordatorio.timeInMillis

            // Mostrar la notificación después del tiempo especificado
            Handler().postDelayed({
                // Crear notificación
                val builder = NotificationCompat.Builder(applicationContext, channelId)
                    .setSmallIcon(R.drawable.notificacion)
                    .setContentTitle(notaNueva.titulo)
                    .setContentText(notaNueva.contenido)
                    .setPriority(NotificationCompat.PRIORITY_HIGH)
                    .setAutoCancel(true)

                // Mostrar Notificación
                with(NotificationManagerCompat.from(applicationContext)) {
                    if (ActivityCompat.checkSelfPermission(applicationContext,
Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {
                        return@postDelayed
                    }
                    mediaPlayer.start()
                    notify(notificationId, builder.build())
                }
            }, tiempoRecordatorio - tiempoActual)
        },
        horaActual,
        minutoActual,
        true
    )
    timePickerDialog.show()
}
```



Definimos el ID de la notificación como 1 y el ID del canal de notificación como "recordatorio\_channel". El canal de notificación es necesario para versiones de Android posteriores a la 8.0 (API nivel 26).

Obtenemos la hora actual del sistema utilizando `Calendar.getInstance()`. Se extrae la hora del día y los minutos actuales.

Creamos un `TimePickerDialog` para que el usuario seleccione la hora y los minutos en los que desea recibir el recordatorio. Este diálogo se muestra en la actividad actual (`this`).

Cuando el usuario selecciona la hora y los minutos deseados en el `TimePickerDialog`, se ejecuta el bloque de código en la lambda del `TimePickerDialog`. En este bloque, se realizan las siguientes acciones:

Se calcula el tiempo hasta que se deba mostrar el recordatorio. Esto se hace comparando la hora y los minutos seleccionados con el tiempo actual en milisegundos (`System.currentTimeMillis()`). Se crea un objeto `Calendar` llamado `calendarRecordatorio` para representar la fecha y hora del recordatorio.

Se crea un `Handler().postDelayed()` para mostrar la notificación después del tiempo especificado. Dentro de este bloque de código diferido, se crea una notificación utilizando `NotificationCompat.Builder`. Se establecen los detalles de la notificación, como el ícono, el título y el contenido, así como la prioridad alta y la cancelación automática.

Finalmente, activamos el sonido de notificación con `mediaPlayer.start()`, sonido que añadimos anteriormente a la carpeta `raw` y se muestra la notificación utilizando `NotificationManagerCompat.notify()` pasando el ID de la notificación y el objeto `Notification` construido.

El `TimePickerDialog` se muestra utilizando `timePickerDialog.show()`, lo que permite al usuario seleccionar la hora y los minutos para programar el recordatorio.

Lo siguiente que veremos es el código necesario para realizar las llamadas a las API's.

Preparamos la llamada.

```
interface OpenFDAService {  
    @GET(Constants.GET_MEDICAMENTOS)  
    suspend fun obtenerMedicamentos(): MedicamentoResponse  
}
```

Es una anotación utilizada por Retrofit para indicar el tipo de solicitud HTTP que se realizará. En este caso, se utiliza la anotación @GET para realizar una solicitud GET a la URL especificada "/drug/label.json?limit=100". Esto indica que se desea obtener información sobre los medicamentos desde la API de OpenFDA y se limita la respuesta a 100 registros.

Preparamos la respuesta con los datos que vamos a recibir de la API en la llamada anterior.

```
data class MedicamentoResponse(  
    val results: List<Result>  
)  
  
data class Result (  
    val purpose: List<String>? = null,  
    val warnings: List<String>? = null,  
    val openfda: Openfda?,  
)  
  
data class Openfda (  
    val brand_name: List<String>? = null,  
    val generic_name: List<String>? = null,  
    val route: List<String>? = null,  
)
```

MedicamentoResponse representa la respuesta del servicio OpenFDA para obtener una lista de medicamentos. Tiene una propiedad llamada results que es una lista de objetos de tipo Result.

Result, representa un resultado individual de la respuesta del servicio OpenFDA. Contiene propiedades como purpose, warnings y openfda. Las propiedades purpose y warnings son listas de cadenas de texto opcionales (List<String>?), mientras que la propiedad openfda es un objeto de tipo Openfda que contiene información adicional relacionada con el medicamento.

Openfda, representa información específica de la marca y el nombre genérico del medicamento, así como su vía de administración. Tiene propiedades como brand\_name, generic\_name y route, que son listas de cadenas de texto opcionales (List<String>?).

Preparamos el objeto que vamos a utilizar y crear con los datos obtenidos de la API.

```
data class Medicamento(
    val nombre: String? = null,
    val nombre_comercial: String? = null,
    val via: String? = null,
    val proposito: String? = null,
    val advertencia: String? = null
): Parcelable {

    constructor(parcel: Parcel) : this(
        parcel.readString(),
        parcel.readString(),
        parcel.readString(),
        parcel.readString(),
        parcel.readString()
    )

    override fun writeToParcel(parcel: Parcel, flags: Int) {
        parcel.writeString(nombre)
        parcel.writeString(nombre_comercial)
        parcel.writeString(via)
        parcel.writeString(proposito)
        parcel.writeString(advertencia)
    }

    override fun describeContents(): Int {
        return 0
    }

    companion object CREATOR : Parcelable.Creator<Medicamento> {
        override fun createFromParcel(parcel: Parcel): Medicamento {
            return Medicamento(parcel)
        }

        override fun newArray(size: Int): Array<Medicamento?> {
            return arrayOfNulls(size)
        }
    }
}
```

La clase Medicamento es una clase de datos que implementa la interfaz Parcelable. Esto permite que un objeto Medicamento se pueda pasar entre componentes de la aplicación, como fragmentos o actividades, a través de argumentos o intenciones.

Constructor secundario (constructor(parcel: Parcel)), que se utiliza para crear un objeto Medicamento a partir de un objeto Parcel, que es una forma de empaquetar y desempaquetar datos para su transferencia. Se utiliza el método readString del Parcel para leer los valores de cada propiedad y asignarlos a las propiedades correspondientes del Medicamento.

Los métodos writeToParcel y describeContents son necesarios para implementar la interfaz Parcelable. El método writeToParcel se utiliza para escribir los valores de las propiedades en un

Parcel, mientras que el método describeContents devuelve un entero que describe el contenido del Parcel (normalmente se devuelve 0).

Objeto compañero (companion object CREATOR): Este objeto compañero implementa la interfaz Parcelable.Creator<Medicamento> y proporciona dos métodos estáticos: createFromParcel y newArray. Estos métodos se utilizan para crear un Medicamento a partir de un Parcel y crear un arreglo de Medicamento con el tamaño especificado, respectivamente.

Por último, lo unimos todo y realizamos la llamada a la API para obtener sus datos.

```
CoroutineScope(Dispatchers.IO).launch {
    try {
        val response = service.obtenerMedicamentos()
        if (response.results.isNotEmpty()) {
            // Creamos un objeto Medicamento con los datos que nos interesan de la API
            val medicamentos = response.results.map { result ->
                Medicamento(
                    nombre = result.openfda?.generic_name?.joinToString(", "),
                    nombre_comercial = result.openfda?.brand_name?.joinToString(", "),
                    via = result.openfda?.route?.joinToString(", "),
                    proposito = result.purpose?.joinToString(", "),
                    advertencia = result.warnings?.joinToString(", ")
                )
            }

            withContext(Dispatchers.Main) {
                vm.actualizarLista(medicamentos)
            }
        } else {
            Log.e(Constants.MENSAJE_ERROR_API, Constants.MENSAJE_LISTA_MEDICAMENTOS_VACIA)
        }
    } catch (exception: Exception) {
        Log.e(Constants.MENSAJE_ERROR_API, exception.message ?:
            Constants.MENSAJE_ERROR_DESCONOCIDO)
    }
}
```

Creamos un nuevo CoroutineScope utilizando el Dispatchers.IO como contexto. Esto indica que las operaciones dentro del bloque del coroutine se ejecutarán en un subproceso de E/S dedicado para realizar tareas de red y otras operaciones bloqueantes sin bloquear el hilo principal de la interfaz de usuario.

Utilizamos launch para iniciar un nuevo coroutine. Esto permite que el bloque de código dentro del coroutine se ejecute de forma asíncrona.

Dentro del coroutine, realizamos una llamada a service.obtenerMedicamentos() para obtener los datos de medicamentos desde la API.

Verificamos si la respuesta (response) contiene algún resultado (results.isNotEmpty()). Si la lista de resultados no está vacía, se procesan los datos.

Para cada elemento en `response.results`, creamos un objeto `Medicamento` utilizando los datos relevantes de la API. Se utilizan operaciones de mapeo y concatenación (`joinToString`) para obtener valores más legibles a partir de las listas de la respuesta.

Después de procesar los datos, utilizamos `withContext(Dispatchers.Main)` para cambiar al contexto del hilo principal (interfaz de usuario).

Dentro del contexto del hilo principal, llamamos al método `actualizarLista()` del objeto `vm` (`ViewModel`) para actualizar la lista de medicamentos con los datos obtenidos.

Si la respuesta está vacía, se registra un mensaje de error utilizando `Log.e()`.

Si ocurre una excepción durante la llamada a la API, se registra un mensaje de error utilizando `Log.e()`.

Para crear la animación visual de rebote en la lista de medicamentos usamos la siguiente función.

```
override fun onScrolled(recyclerView: RecyclerView, dx: Int, dy: Int) {
    super.onScrolled(recyclerView, dx, dy)

    val layoutManager = recyclerView.layoutManager as LinearLayoutManager?
    val primeraPosicion = layoutManager!!.findFirstCompletelyVisibleItemPosition()
    val ultimaPosicion = layoutManager.findLastCompletelyVisibleItemPosition()
    val contadorMedicamentos = layoutManager.itemCount

    if (primeraPosicion == 0 || ultimaPosicion == contadorMedicamentos - 1) {
        // Comprobamos si hay movimiento
        val posicionActual = recyclerView.computeVerticalScrollOffset()

        if (posicionAnterior != -1 && posicionActual != posicionAnterior) {
            // Animación de rebote
            val animator = ObjectAnimator.ofFloat(recyclerView, Constantes.ANIMACION_EJE_Y, 0f, 50f, 0f)
            animator.interpolator = DecelerateInterpolator()
            animator.duration = 500
            animator.start()
        }
        posicionAnterior = posicionActual
    }
}
```

Guardamos en todo momento la posición actual y la anterior, para de esta forma compararlas y crear la animación correctamente.

Para ello tenemos que obtener el número total de medicamentos y dar valores al rebote para que surja efecto.

La animación de los botones de la pantalla fórmulas hace lo siguiente.

```
fun animacionBotones(boton: Button, siguientePantalla: NavDirections): ObjectAnimator {  
    // Animación que hace que el botón se haga más grande y vuelva a su estado original  
    val animacion = ObjectAnimator.ofPropertyValuesHolder(  
        boton,  
        PropertyValuesHolder.ofFloat(View.SCALE_X, 1f, 1.3f, 1f),  
        PropertyValuesHolder.ofFloat(View.SCALE_Y, 1f, 1.3f, 1f),  
    )  
    animacion.addListener(object: Animator.AnimatorListener {  
        override fun onAnimationStart(animation: Animator) { }  
        override fun onAnimationEnd(animation: Animator) {  
            // Al finalizar la animación navega a la siguiente pantalla  
            findNavController().navigate(siguientePantalla)  
            padre.hacerInvisibleMenuInferior()  
        }  
        override fun onAnimationCancel(animation: Animator) { }  
        override fun onAnimationRepeat(animation: Animator) { }  
    })  
    return animacion  
}
```

Pasamos a la función dos elementos, un botón y una dirección o pantalla siguiente.

Creamos una instancia de ObjectAnimator para realizar la animación del botón. La animación cambia la escala del botón para hacerlo más grande y luego volver a su estado original.

Utilizamos el método "ofPropertyValuesHolder" de ObjectAnimator para definir los valores de propiedad que cambiarán durante la animación. En este caso, cambiamos la escala en el eje X y en el eje Y del botón.

Añadimos un listener al objeto de animación para escuchar los eventos de inicio y fin de la animación.

En el método "onAnimationEnd", que se ejecuta al finalizar la animación, se realiza la navegación a la siguiente pantalla utilizando el NavController y el método "navigate" con la dirección "siguientePantalla". Además, se llama al método "hacerInvisibleMenuInferior" en el objeto "padre".

De la siguiente forma obtenemos el factor de pérdidas del desplegable de la pantalla de pérdidas insensibles.

```
// Función para obtener el factor correspondiente a la opción seleccionada en el Spinner
private fun obtenerFactor(opcion: String): Double? {
    return when (opcion) {
        Constantes.MENSAJE_FACTOR_PERDIDAS_BEBES -> 65.0
        Constantes.MENSAJE_FACTOR_PERDIDAS_NINOS -> 50.0
        Constantes.MENSAJE_FACTOR_PERDIDAS_ADULTOS -> 35.0
        else -> null
    }
}
```

Devuelvo el valor dependiendo de la opción que se haya elegido.

Al igual pasa con el estado de IMC en la pantalla de índice de masa corporal.

```
// Función para obtener el estado del IMC según su valor
private fun obtenerEstadoIMC(imc: Double): String {
    return when {
        imc < 18.5 -> Constantes.MENSAJE_IMC_BAJO_PESO
        imc < 24.9 -> Constantes.MENSAJE_IMC_PESO_SALUDABLE
        imc < 29.9 -> Constantes.MENSAJE_IMC_SOBREPESO
        imc < 34.9 -> Constantes.MENSAJE_IMC_OBESIDAD_1
        imc < 39.9 -> Constantes.MENSAJE_IMC_OBESIDAD_2
        else -> Constantes.MENSAJE_IMC_OBESIDAD_3
    }
}
```

Dependiendo del valor obtenido, muestra un mensaje u otro, indicando el estado o forma física del paciente.

## MANUAL DE INSTALACIÓN Y USO DE LA APLICACIÓN

### *Requisitos del sistema*

Dispositivo móvil con sistema operativo Android 6.0 o superior.

Conexión a Internet para descargar la aplicación y acceder a los recursos en línea.

### *Instalación de la aplicación*

Descarga el .apk aportado junto a éste documento y ejecútalo, automáticamente el dispositivo te abrirá el menú de instalación, pulsas en instalar y listo, ya tienes acceso a la aplicación.

### *Registro de cuenta*

Abre la aplicación NursiPedia en tu dispositivo.

En la pantalla de inicio de sesión, haz clic en el enlace "Registrarse".

Completa el formulario de registro con tu dirección de correo electrónico y una contraseña segura.

Haz clic en el botón "Registrarse" para crear tu cuenta de NursiPedia.

### *Inicio de sesión*

Abre la aplicación NursiPedia en tu dispositivo.

En la pantalla de inicio de sesión, ingresa tu dirección de correo electrónico y contraseña.

Haz clic en el botón "Iniciar sesión" para acceder a tu cuenta de NursiPedia.

### *Uso de la aplicación*

Pantalla principal: Una vez que hayas iniciado sesión, serás redirigido a la pantalla principal de NursiPedia. Aquí encontrarás íconos de navegación para acceder a diferentes secciones de la aplicación, como "Medicamentos", "Calculadoras" y "Perfil".

Búsqueda de medicamentos: En la sección "Medicamentos", podrás realizar búsquedas de medicamentos por su nombre o nombre comercial. Ingresa el término de búsqueda en el campo correspondiente y haz clic en el botón de búsqueda para obtener los resultados.

Además encontrarás un botón que te permitirá acceder a tus notas.

Visualización de detalles del medicamento: Una vez que hayas realizado una búsqueda de medicamentos y obtengas los resultados, podrás hacer clic en un medicamento específico para ver más detalles, como el propósito terapéutico y advertencias asociadas.

Creación y gestión de notas: En la sección "Notas", podrás crear y gestionar tus propias notas. Haz clic en el botón "+" para agregar una nueva nota y completa los campos requeridos. También podrás ver y eliminar notas existentes.

Utilización de calculadoras médicas: En la sección "Calculadoras", encontrarás diversas calculadoras médicas para realizar cálculos relacionados con infusiones, disoluciones, pérdidas insensibles, índice de masa corporal (IMC), regla de tres y gasto cardíaco.



*Uso de la Calculadora de Infusiones*

En la pantalla principal de NursiPedia, selecciona la sección "Calculadoras".

Haz clic en el ícono de "Infusiones" para acceder a la calculadora.

En la pantalla de la calculadora de infusiones, ingresa el volumen a infundir, la velocidad de infusión y las horas.

Haz clic en el botón "Calcular" para obtener el resultado.

El resultado mostrará la fórmula utilizada y el resultado calculado.

*Uso de la Calculadora de Disoluciones*

En la pantalla principal de NursiPedia, selecciona la sección "Calculadoras".

Haz clic en el ícono de "Disoluciones" para acceder a la calculadora.

En la pantalla de la calculadora de disoluciones, ingresa la cantidad de medicamento, el volumen de la disolución y la concentración de la disolución.

Haz clic en el botón "Calcular" para obtener el resultado.

El resultado mostrará la fórmula utilizada y el resultado calculado.

*Uso de la Calculadora de Pérdidas Insensibles*

En la pantalla principal de NursiPedia, selecciona la sección "Calculadoras".

Haz clic en el ícono de "Pérdidas Insensibles" para acceder a la calculadora.

En la pantalla de la calculadora de pérdidas insensibles, selecciona el factor de pérdidas insensibles, ingresa el peso y haz clic en el botón "Calcular".

El resultado mostrará la fórmula utilizada y el resultado calculado.

*Uso de la Calculadora de Índice de Masa Corporal (IMC)*

En la pantalla principal de NursiPedia, selecciona la sección "Calculadoras".

Haz clic en el ícono de "IMC" para acceder a la calculadora.

En la pantalla de la calculadora de IMC, ingresa el peso y la altura.

Haz clic en el botón "Calcular" para obtener el resultado.

El resultado mostrará la fórmula utilizada y el resultado calculado.

*Uso de la Calculadora de Regla de Tres*

En la pantalla principal de NursiPedia, selecciona la sección "Calculadoras".

Haz clic en el ícono de "Regla de Tres" para acceder a la calculadora.

En la pantalla de la calculadora de regla de tres, ingresa los valores requeridos en cada campo.

Haz clic en el botón "Calcular" para obtener el resultado.

El resultado mostrará la fórmula utilizada y el resultado calculado.

*Uso de la Calculadora de Gasto Cardíaco*

En la pantalla principal de NursiPedia, selecciona la sección "Calculadoras".

Haz clic en el ícono de "Gasto Cardíaco" para acceder a la calculadora.

En la pantalla de la calculadora de gasto cardíaco, ingresa la frecuencia cardíaca y el volumen sistólico.

Haz clic en el botón "Calcular" para obtener el resultado.

El resultado mostrará la fórmula utilizada y el resultado calculado.

Información de cuenta: En la sección "Perfil", podrás ver información de tu perfil, como tu dirección de correo electrónico, y cerrar sesión.

### Cierre de sesión

Para cerrar sesión en NursiPedia, ve a la sección "Perfil".

Haz clic en el botón "Cerrar sesión" para finalizar tu sesión actual.

Serás redirigido a la pantalla de inicio de sesión.

The screenshots show the following screens of the NursiPedia app:

- Inicio de Sesión (Login):** Features the NursiPedia logo and fields for Email and Contraseña (Password) with buttons for REGISTRARSE (Register) and INICIAR SESIÓN (Login).
- Perfil (Profile):** Similar to the login screen, but includes a Volver (Return) button and a REGISTRARSE button.
- Fórmulas (Formulas):** A menu with icons for Infusiones (Infusions), Disoluciones (Solutions), Salud Corporal (Body Health), Pérdidas Insensibles (Sensory Losses), and Otras (Others). It also includes a Regla de Tres (Rule of Three) and Gasto Cardíaco (Cardiac Output) section.
- Medicamentos (Medications):** A list of various medications including SILICEA, DAPHNE MEZEREUM BARK, CEFLOXACIN, NAPROXEN, BENZALKONIUM CHLORIDE, SALICYLIC ACID, CUPRUM ACETICUM NICOTIANA, TRAMETINIB, GLIMEPIRIDE, METHOCARBAMOL, HYDROCORTISONE, LEVOTHYROXINE, LIOTHYRONINE, and CHLORHEXIDINE GLUCONATE AND ISOPROPRAL.
- Detalle Medicamento (Medication Detail):** Shows details for GLYCERIN (OASIS Tears Lubricant Eye), including administration route (OPHTHALMIC), purpose (lubricant eye drops), and warnings.
- Infusiones (Infusions):** A calculation tool with input fields for VAI (Volume to Infuse in ml), V (Infusion Rate in ml/h), and H (Hours). It includes a CALCULAR button and the formula  $H \times V = VAI$ .
- Disoluciones (Solutions):** A calculation tool with input fields for CM (Quantity of Medication in mg), VD (Volume of the Solution in ml), and CD (Concentration of the Solution in mg/ml). It includes a CALCULAR button and the formula  $CD = CM / VD$ .
- Pérdidas Insensibles (Sensory Losses):** A calculation tool with input fields for FP (Factor of Sensory Losses in ml/kg/day), P (Weight in kg), and PI (Sensory Losses in ml). It includes a CALCULAR button and the formula  $PI = P \times FP$ .

← Índice de Masa Corporal

P

Peso (kg)

A

Altura (m)<sup>2</sup>

IMC

Índice de Masa Corporal

CALCULAR

IMC = P / A<sup>2</sup>

← Regla de Tres

A

→ B

C

→ x

CALCULAR

$x = A * B / C$

← Gasto Cardíaco

FC

Frecuencia Cardíaca (latidos/min)

VS

Volumen Sistólico (ml/latido)

GC

Gasto Cardíaco (ml/min/kg)

CALCULAR

GC = FC \* VS

Notas

Nota Larga

¿Qué es Lorem Ipsum? ...

Nota de Pepe

Prueba

+

Nueva Nota

←

Nota

09/06/2023

Nota Larga

¿Qué es Lorem Ipsum?

Lorem Ipsum es simplemente el texto de relleno de las imprentas y archivos de texto. Lorem Ipsum ha sido el texto de relleno estándar de las industrias desde el año 1500, cuando un impresor (N. del T. persona que se dedica a la imprenta) desconocido usó una galería de textos y los mezcló de tal manera que logró hacer un libro de textos especímen. No sólo sobrevivió 500 años, sino que también ingresó como texto de relleno en documentos electrónicos, quedando esencialmente igual al original. Fue popularizado en los 60s con la creación de las hojas "Letraset", las cuales contenían pasajes de Lorem Ipsum, y más recientemente con software de autoedición, como por ejemplo Aldus PageMaker, el cual incluye versiones de Lorem Ipsum.

¿Por qué lo usamos?

Es un hecho establecido hace demasiado tiempo que un lector se distraerá con el contenido del texto de un sitio mientras que mira su diseño. El punto de usar Lorem Ipsum es que tiene una distribución más o

Nota

09/06/2023

Nota Larga

17:56

CANCELAR ACEPTAR

¿Por qué lo usamos?

Es un hecho establecido hace demasiado tiempo que un lector se distraerá con el contenido del texto de un sitio mientras que mira su diseño. El punto de usar Lorem Ipsum es que tiene una distribución más o

← Cuenta

pepe@gmail.com

CERRAR SESIÓN

## CONCLUSIONES

En conclusión, la aplicación aporta a un enfermero múltiples herramientas para mejorar su rendimiento en el trabajo diario. A pesar de no abarcarlo todo, les puede ayudar en muchas tareas.

La aplicación todavía puede mejorarse incluyendo otras funciones, pero en rasgos generales creo que cumple con las necesidades básicas.

Con este proyecto he aprendido mucho utilizando herramientas no vistas en clase como pueden ser Firebase o Retrofit, además de las animaciones, estilos, inclusión de música y otras muchas cosas.

## GLOSARIO DE TÉRMINOS

**Infusiones:** Proceso de administración de líquidos y medicamentos a través de una vía intravenosa.

**Disoluciones:** Son mezclas de solutos y solventes utilizadas en la preparación de medicamentos y soluciones médicas.

**IMC (Índice de Masa Corporal):** Es una medida que se utiliza para evaluar la relación entre el peso y la altura de una persona y determinar su categoría de peso. Es una herramienta comúnmente utilizada en la evaluación del estado nutricional y el riesgo de enfermedades relacionadas con el peso.

**Gasto cardíaco:** Cantidad de sangre que el corazón bombea por minuto, utilizado como indicador de la eficiencia del corazón y la circulación sanguínea.

**API:** Siglas en inglés de Application Programming Interface (Interfaz de Programación de Aplicaciones). Se refiere a un conjunto de reglas y protocolos que permiten la comunicación y la interacción entre diferentes componentes de software.

**OpenFDA:** Es una plataforma desarrollada por la Administración de Alimentos y Medicamentos de los Estados Unidos (FDA, por sus siglas en inglés) que proporciona acceso público a datos sobre medicamentos, dispositivos médicos y productos alimentarios. OpenFDA ofrece una API (Interfaz de Programación de Aplicaciones) que permite a los desarrolladores acceder y utilizar estos datos de manera programática.

La API de OpenFDA proporciona una forma estructurada y fácil de acceder a la información recopilada por la FDA. Los datos disponibles incluyen información sobre medicamentos aprobados, retiros de productos, eventos adversos reportados, etiquetas de alimentos, entre otros.

## BIBLIOGRAFÍA

Documentación oficial de Android:

URL: <https://developer.android.com/docs>

Kotlin Coroutines en Android:

URL: <https://developer.android.com/kotlin/coroutines>

Android Developers:

URL: <https://developer.android.com/>

Kotlin Programming Language:

URL: <https://kotlinlang.org/>

GitHub:

URL: <https://github.com/>

Stack Overflow:

URL: <https://stackoverflow.com/questions/tagged/android>

YouTube MoureDev (tutoriales de Android y Firebase):

URL: <https://www.raywenderlich.com/android>

Documentación oficial de Firebase:

URL: <https://firebase.google.com/docs>

Documentación oficial de Retrofit:

URL: <https://square.github.io/retrofit>

MedlinePlus:

URL: <https://medlineplus.gov/>

OpenFDA:

URL: <https://open.fda.gov/>

