



UNIVERSIDAD TECNOLÓGICA DE EL SALVADOR.

FACULTAD DE INFORMÁTICA Y CIENCIAS
APLICADAS.

Asignatura: Técnicas de calidad de Software

Sección: 01

Tema: "MediPlus+" Aplicación móvil para la búsqueda de información sobre medicamentos y farmacias

Docente: José Orlando Girón Barrera

Integrantes:

Nombres	Apellidos	Correos
Carolina Lisseth	Montalvo Villamariona	2506462017@mail.utec.edu.sv
Diego Fernando	Bejarano Carías	2502112017@mail.utec.edu.sv
Gabriel Omar	Chávez Nieto	2518772017@mail.utec.edu.sv
Gabriela Alejandra	Ruiz Quinteros	2514892017@mail.utec.edu.sv
Luis Antonio Fredy	Romero Cruz	2517632017@mail.utec.edu.sv

Fecha de entrega : 06/12/2021

Contenido

INTRODUCCIÓN.....	3
OBJETIVOS	4
DEFINICIÓN DEL PROYECTO	5
JUSTIFICACIÓN DEL PROYECTO	6
PLANIFICACIÓN DEL PROYECTO.....	7
INTERFAZ GRÁFICA	9
Diseño de pantallas a utilizar	9
LOGIN	9
REGISTRO	10
MENÚ	11
MEDICAMENTOS.....	12
CATEGORIA – Ejemplo: Gripe, Tos y Asma.	13
MEDICAMENTO SELECCIONADO – Ejemplo: Abrilar jarabe frasco.	14
FARMACIAS.....	15
SUCURSALES – Ejemplo: Farmacia San Nicolas	16
SUCURSAL SELECCIONADA – Ejemplo: Antiguo Cuscatlán.....	17
INFORMACION DE LOS DESARROLLADORES	18
NAVEGABILIDAD.....	19
PANTALLAS PROGRAMADAS USANDO FLUTTER	20
DISEÑO DE LA BASE DE DATOS.....	31
MANUAL TÉCNICO	35
MANUAL DE USO DE LA APLICACIÓN.....	79
CAPTURAS DE PRESENTACIÓN FINAL	91
CONCLUSIÓN.....	94
ENLACE A CÓDIGO FUENTE DEL PROYECTO UBICADO EN REPOSITORIO.....	94
EVALUACIONES PARA CUARTO AVANCE	95
Porcentaje de avance del proyecto.....	95
Autoevaluación y otras evaluaciones por parte del grupo	95
Actividades realizadas por cada miembro del grupo	96

INTRODUCCIÓN

Dentro del presente documento se encuentra el contenido referente al proyecto de desarrollo grupal correspondiente a la asignatura electiva técnica 3, la cual es denominada “Técnicas de calidad de software”. El proyecto consiste en la realización de una aplicación móvil para la consulta de información sobre farmacias y medicamentos. A lo largo de este documento, el lector encontrará elementos como los siguientes:

- Definición y justificación teórica del proyecto, así como también la planificación de cada actividad llevada a cabo durante la realización del mismo, colocado a modo de cronograma y listado de actividades con sus integrantes designados.
- Presentación de planteamiento inicial de interfaz gráfica, mostrando lo que cada pantalla incluirá según una primera planificación, y concluyendo este apartado con la diagramación de la navegabilidad esperada entre pantallas.
- Un apartado con las capturas que muestran las pantallas programadas directamente utilizando las tecnologías necesarias.
- Diseño y explicación de la base de datos, presentando su composición y detallando lo necesario para que el lector observe cómo es que se han organizado los datos con los cuales la aplicación interactúa.
- Manuales escritos para explicar tanto el uso de la aplicación como el código fuente de esta.
- Imágenes que muestran las diapositivas oficiales que han de ser usadas el día que corresponda la defensa final del proyecto.
- La debida conclusión de todo el proyecto y un enlace directo al repositorio en donde el código fuente de la aplicación se encuentra guardado.

Esperando que el contenido dentro de este documento sea para el agrado y comprensión por parte de los lectores, y deseando que cada detalle explicado ayude a comprender los resultados de la programación del software que es parte principal del proyecto de desarrollo.

OBJETIVOS

Objetivo General.

- Desarrollar una aplicación donde el usuario tenga la facilidad de encontrar la información que necesite sobre medicamentos, brindando la confianza en que dicha información es verídica, así como de tener la opción de encontrar una farmacia donde sea más conveniente comprar sus medicamentos.

Objetivos específicos.

- Detallar la información de cada medicamento, sea este uno generalizado o por prescripción médica.
- Mostrar las farmacias donde el usuario pueda hacer la compra de sus medicamentos, midiendo factores como el económico y la cercanía de la farmacia.

DEFINICIÓN DEL PROYECTO

Todas las personas sabemos que la salud es importante, un factor que no se puede dejar olvidado. Por ello, los profesionales en el área de la medicina y salud en general invierten tiempo, conocimientos y mucho esfuerzo en la creación de distintos suplementos que brindan al consumidor las suficientes propiedades que su cuerpo necesita para procurar un estado de salud óptimo y adecuado a la persona, sea que esta persona se encuentre enferma o en otra condición anormal, o que simplemente necesite de ciertos complementos para llevar un estilo de vida lo más saludable y balanceado posible.

Este tipo de suplementos son los que comúnmente conocemos como medicinas o medicamentos, y los solemos ver en la forma de cápsulas, pastillas masticables, efervescentes, jarabes, cremas de uso tópico, inyecciones, preparados alimenticios, suplementos vitamínicos, entre otros.

Entonces, nuestro proyecto consistirá en la creación y desarrollo de una aplicación móvil que funcione como recopilatorio de toda la información relevante referente a los medicamentos, además de las farmacias donde éstos pueden ser conseguidos, todo ello mostrado al usuario a manera de listado.

La idea principal del proyecto de desarrollo de esta aplicación móvil será facilitar al usuario información más reciente y correcta de los medicamentos en una interfaz cómoda y más organizada. Lo que evitará inconvenientes para el usuario causados por haber buscado los detalles de sus medicinas en un sitio web de poca confianza y credibilidad, así como el detalle de tener que esperar mucho para poderse comunicar con su médico o algún otro especialista para recibir los datos que necesita.

JUSTIFICACIÓN DEL PROYECTO

Todas las personas día a día tienen que comprar medicamentos ya sea para uso propio o para algún familiar, pero siempre hay inconvenientes al momento de comprar y surgen muchas dudas, por ejemplo: ¿En cuál farmacia estará este medicamento?, ¿Qué precio aproximado tiene este medicamento?, ¿Cuál es la dosis recomendada de este medicamento? Todas estas preguntas son las que mayormente se hacen y la mayoría de las personas no tienen un médico privado al cual consultar para que le solvete estas preguntas y en algunos casos, tampoco se tiene una fuente confiable donde se puedan aclarar todas estas inquietudes.

Es por eso que se lleva a cabo el desarrollo de esta aplicación, con la finalidad de cubrir las necesidades del usuario, ya que funcionará como recopilatorio de la información referente a medicamentos que, a su vez, mostrará las farmacias donde éstos pueden ser conseguidos. Esto ayudará a que el usuario ahorre tiempo a la hora de buscar una farmacia, puesto que la aplicación mostrará las farmacias más cercanas, asimismo, el usuario podrá ver toda la información del medicamento que está buscando.

Todos estos problemas se resolverán con el desarrollo de esta aplicación móvil que tendrá una interfaz organizada y amigable para que todos los usuarios la puedan usar sin ningún problema o confusión. Este proyecto pretende llegar al mayor uso posible por parte de muchos usuarios, siempre pensando en los adultos mayores que se les hace complicado comprar sus medicamentos por los inconvenientes ya antes mencionados.

PLANIFICACIÓN DEL PROYECTO

El proyecto de desarrollo para la aplicación se planifica de la siguiente manera:

Actividades	Agosto				Septiembre					Octubre			Noviembre			Diciembre		
	S1	S2	S3	S4	S1	S2	S3	S4	S5	S1	S2	S3	S4	S1	S2	S3	S4	S1
Planteamiento inicial del proyecto	X																	
Definición del proyecto	X																	
Redacción de los objetivos		X																
Creación de diseños de pantallas		X																
Descripción de diseños de pantallas		X																
Planteamiento de la navegabilidad		X																
Revisión del documento para primera entrega		X																
Entrega del primer avance		X																
Analizar resultados del primer avance			X															
Redacción de justificación del proyecto			X	X														
Planificación del proyecto						X												
Programación de pantallas en flutter						X	X											
Descripción de programación de pantallas							X											
Revisión del documento para segunda entrega								X										
Entrega del segundo avance								X										
Revisión general de la documentación del proyecto									X									
Planificación de creación del manual técnico									X									
Planificación de creación del manual de uso									X									
Primera redacción de las conclusiones										X								
Primera creación de la presentación final										X								
Publicación de proyecto en versionadores y repositorios											X							
Revisión del documento para tercera entrega											X							
Entrega del tercer avance											X							
Revisión general del proyecto de desarrollo											X							
Finalización del documento oficial											X							
Finalización del manual técnico												X						
Finalización del manual de uso												X						
Publicación de proyecto en versionadores y repositorios												X						
Entrega de avances restantes													X					
Defensa del proyecto														X	X	X		

Adicional al cronograma, es necesario enlistar las diferentes actividades y los integrantes del grupo que serán responsables de la realización de éstas, lo que veremos en la siguiente tabla:

Actividades primarias	Integrantes				
	Diego	Fredy	Gabriel	Carolina	Gabriela
Planteamiento inicial del proyecto	X		X		
Definición del proyecto	X		X		
Redacción de los objetivos			X		
Creación de diseños de pantallas		X		X	X
Descripción de diseños de pantallas	X			X	
Planteamiento de la navegabilidad	X			X	
Revisión del documento para primera entrega	X	X	X	X	X
Entrega del primer avance	X	X	X	X	X
Analizar resultados del primer avance	X	X	X	X	X
Redacción de justificación del proyecto			X		
Planificación del proyecto	X				
Programación de pantallas en flutter	X	X	X	X	X
Descripción de programación de pantallas	X	X	X	X	X
Revisión del documento para segunda entrega	X	X	X	X	X
Entrega del segundo avance	X	X	X	X	X
Revisión general de la documentación del proyecto	X	X	X	X	X
Planificación de creación del manual técnico	X	X			
Planificación de creación del manual de uso	X		X		
Primera redacción de las conclusiones	X				X
Primera creación de la presentación final				X	X
Publicación de proyecto en versionadores y repositorios	X				
Revisión del documento para tercera entrega	X	X	X	X	X
Entrega del tercer avance	X	X	X	X	X
Revisión general del proyecto de desarrollo	X	X	X	X	X
Finalización del documento oficial					
Finalización del manual técnico	X	X			
Finalización del manual de uso	X		X		
Publicación de proyecto en versionadores y repositorios	X				
Entrega de avances restantes	X	X	X	X	X
Defensa del proyecto	X	X	X	X	X

INTERFAZ GRÁFICA

Diseño de pantallas a utilizar



LOGIN

Esta es la pantalla de inicio de sesión con un mensaje de bienvenida, servirá para ingresar a la aplicación. Incluye los siguientes controles:

1. Casilla de texto, en donde ingresara nombre de usuario.
2. Casilla de texto, en donde ingresara contraseña.
3. Botón para iniciar sesión.
4. Botón “Registrarse”, que redirige a la pantalla de registro en caso de que el usuario aún no posea una cuenta.



REGISTRO

Esta pantalla se compone de lo siguiente:

1. Casilla de texto, en donde ingresará su nombre el usuario.
2. Casilla de texto, en donde ingresará correo electrónico.
3. Casilla de texto, en donde ingresará nombre de usuario para iniciar sesión.
4. Casilla de texto, en donde ingresará contraseña.
5. Casilla de texto, en donde ingresará nuevamente la contraseña, para verificar que este correctamente escrita.
6. Botón “Registrarse”, para guardar su registro y tener su cuenta.
7. Botón “Cancelar”, por si desea regresar a la pantalla principal “Login”.



MENÚ

Pantalla que muestra las opciones del menú de la aplicación, se compone de las siguientes opciones:

1. Farmacias registradas: el usuario podrá visualizar las farmacias que estén afiliadas a nuestra aplicación.
2. Medicamentos: se visualizarán las categorías de los medicamentos.
3. Información: nos lleva a la pantalla que contiene los nombres de los desarrolladores de la aplicación.
4. Botón “Cerrar sesión”, por si el usuario desea cerrar sesión.
5. Botón “Cerrar app”, por si el usuario desea cerrar la aplicación sin la necesidad de desconectarse de su cuenta.

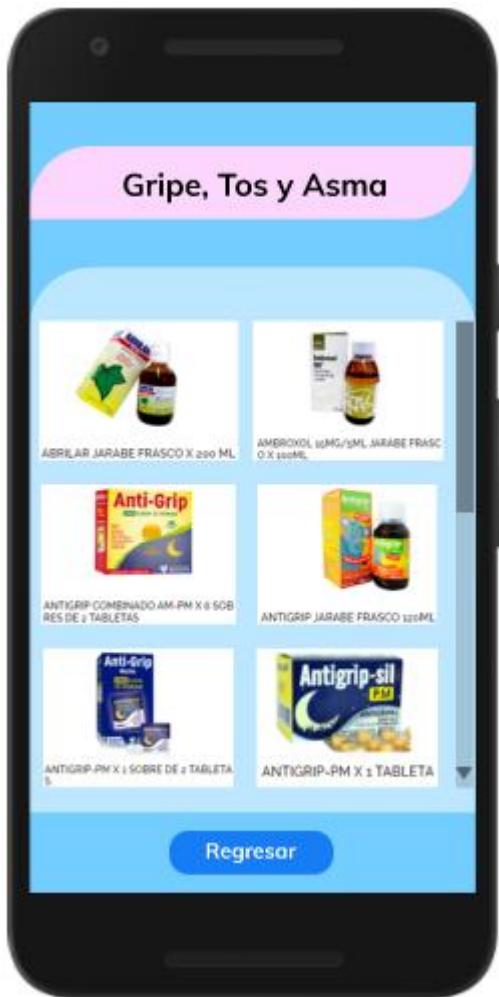


MEDICAMENTOS

Pantalla que muestra las categorías de los medicamentos, contiene lo siguiente:

1. Barra de búsqueda de medicamentos.
2. Lista de categorías de medicamentos.
3. Botón “Regresar”, por si el usuario desea volver al menú de la app.

CATEGORIA – Ejemplo: Gripe, Tos y Asma.



Aquí se muestran los medicamentos de la categoría que el usuario eligió a modo de listado con imágenes y el nombre del medicamento. En este ejemplo se seleccionó la categoría “Gripe, tos y asma”. La lista de resultados dependerá de la categoría seleccionada. Incluye un botón “Regresar”, por si el usuario desea volver a la pantalla anterior.



MEDICAMENTO SELECCIONADO – Ejemplo:
Abrilar jarabe frasco.

Aquí se muestra el medicamento seleccionado, con información detallada del producto. En este ejemplo, se seleccionó el medicamento conocido como “Abrilar Jarabe Frasco x 200 ml”. La información mostrada dependerá del medicamento seleccionado. Incluye un botón “Regresar”, por si el usuario desea volver a la pantalla anterior.



FARMACIAS

Similar a la pantalla de Categorías de medicamentos, sólo que en esta pantalla se muestran las opciones de las distintas farmacias afiliadas a la aplicación.

1. Barra de búsqueda de farmacias.
2. Lista de farmacias que están afiliadas a nuestra aplicación.
3. Botón “Regresar”, por si el usuario desea volver al menú de la app.

SUCURSALES – Ejemplo: Farmacia San Nicolas



Pantalla que muestra un listado de sucursales, en este ejemplo se seleccionó la opción de “Farmacia San Nicolas”. La lista de sucursales dependerá de la farmacia que el usuario haya seleccionado. Incluye:

1. Barra de búsqueda de sucursales de la farmacia seleccionada.
2. Listado de sucursales, si el usuario selecciona una de ellas se mostrará mayor información.
3. Botón “Regresar”, por si el usuario desea volver a la pantalla anterior.



SUCURSAL SELECCIONADA – Ejemplo: Antiguo Cuscatlán

Aquí se muestran la sucursal seleccionada, con información detallada de esta, en este ejemplo se seleccionó la sucursal “Antiguo” de la Farmacia San Nicolas, la información mostrada dependerá de la sucursal de la farmacia que se desea consultar. Incluye un botón "Regresar", por si el usuario desea volver a la pantalla anterior.

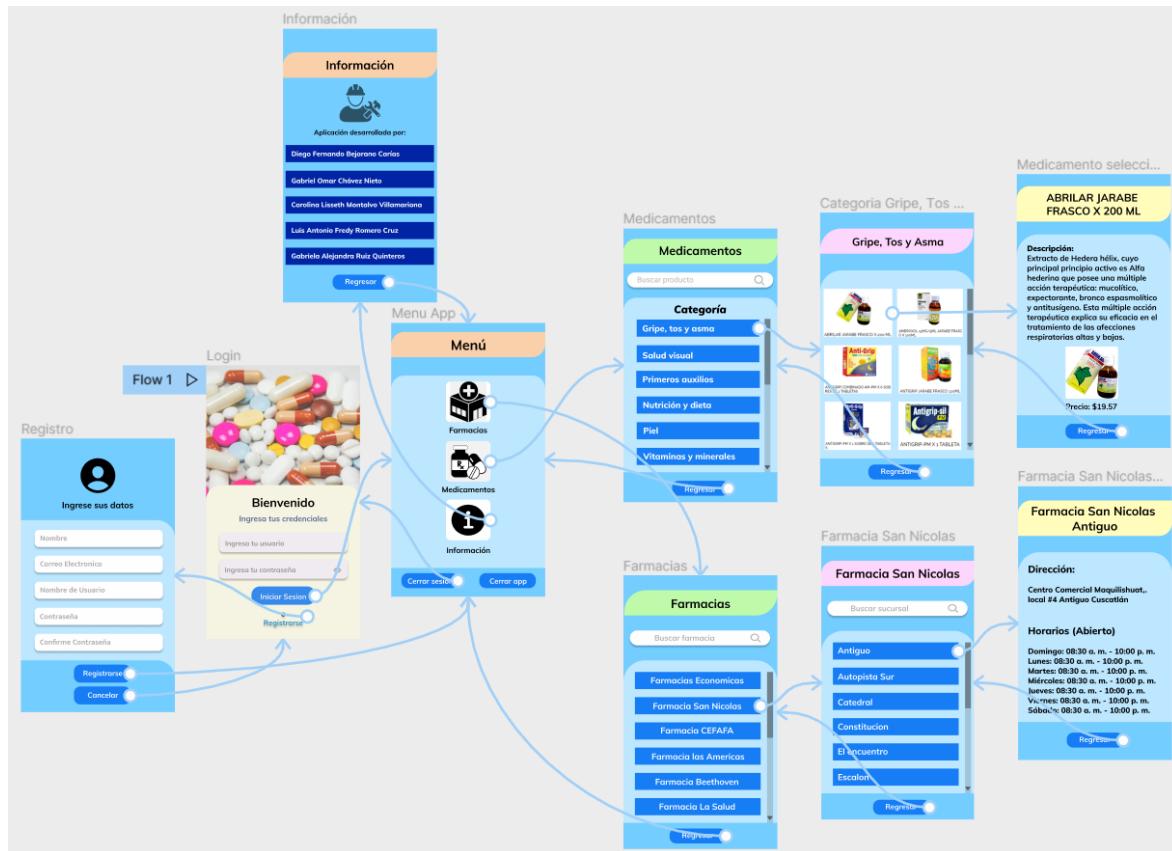
INFORMACION DE LOS DESARROLLADORES



Pantalla en la que se muestra información acerca de los desarrolladores de la aplicación. El botón “Regresar” llevará al menú de la aplicación.

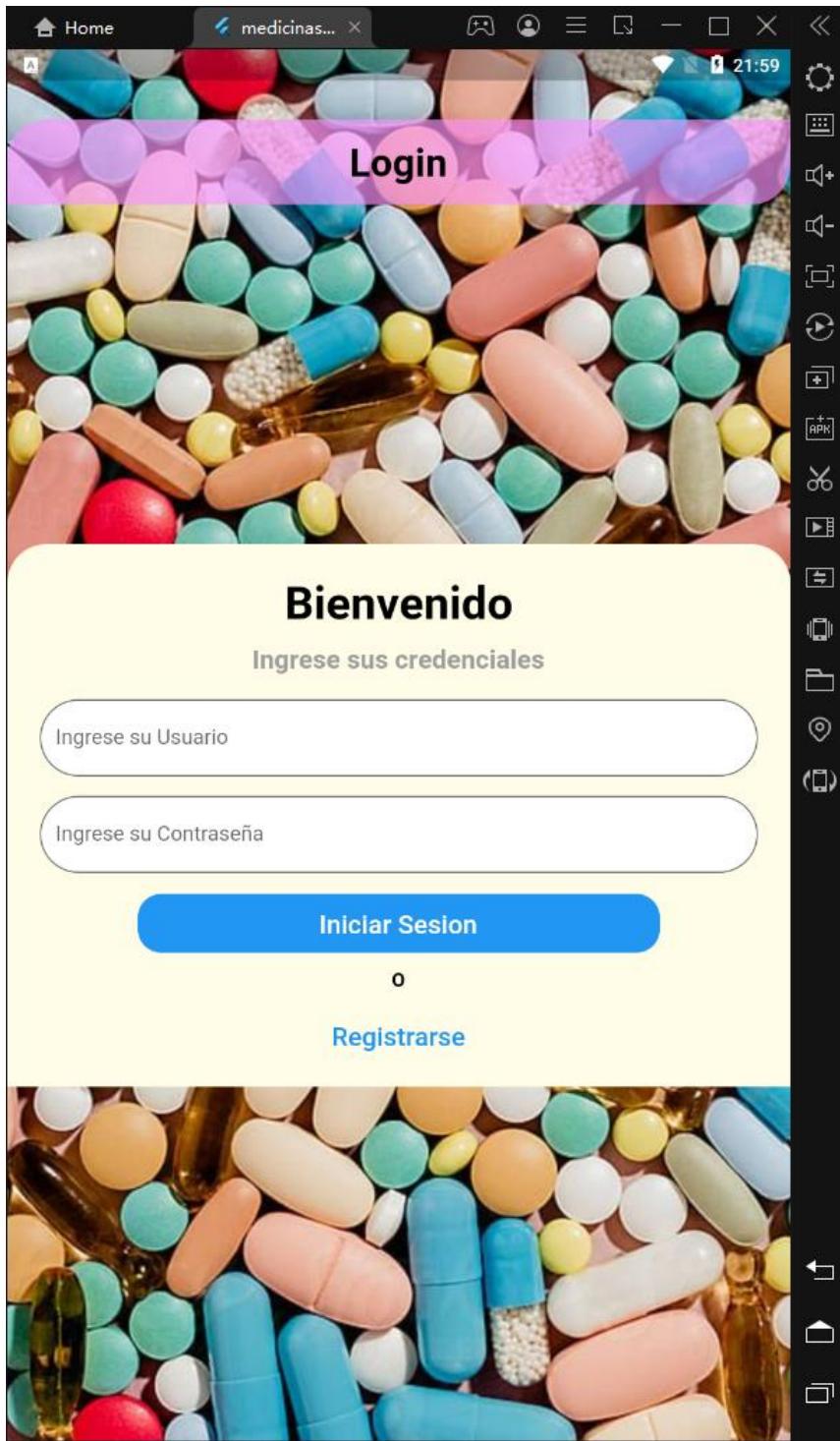
NAVEGABILIDAD

Cada una de las pantallas en el proyecto han sido ubicadas de la manera más ordenada posible para mostrar el flujo que la aplicación tendrá haciendo uso de todas las pantallas explicadas en la sección anterior.

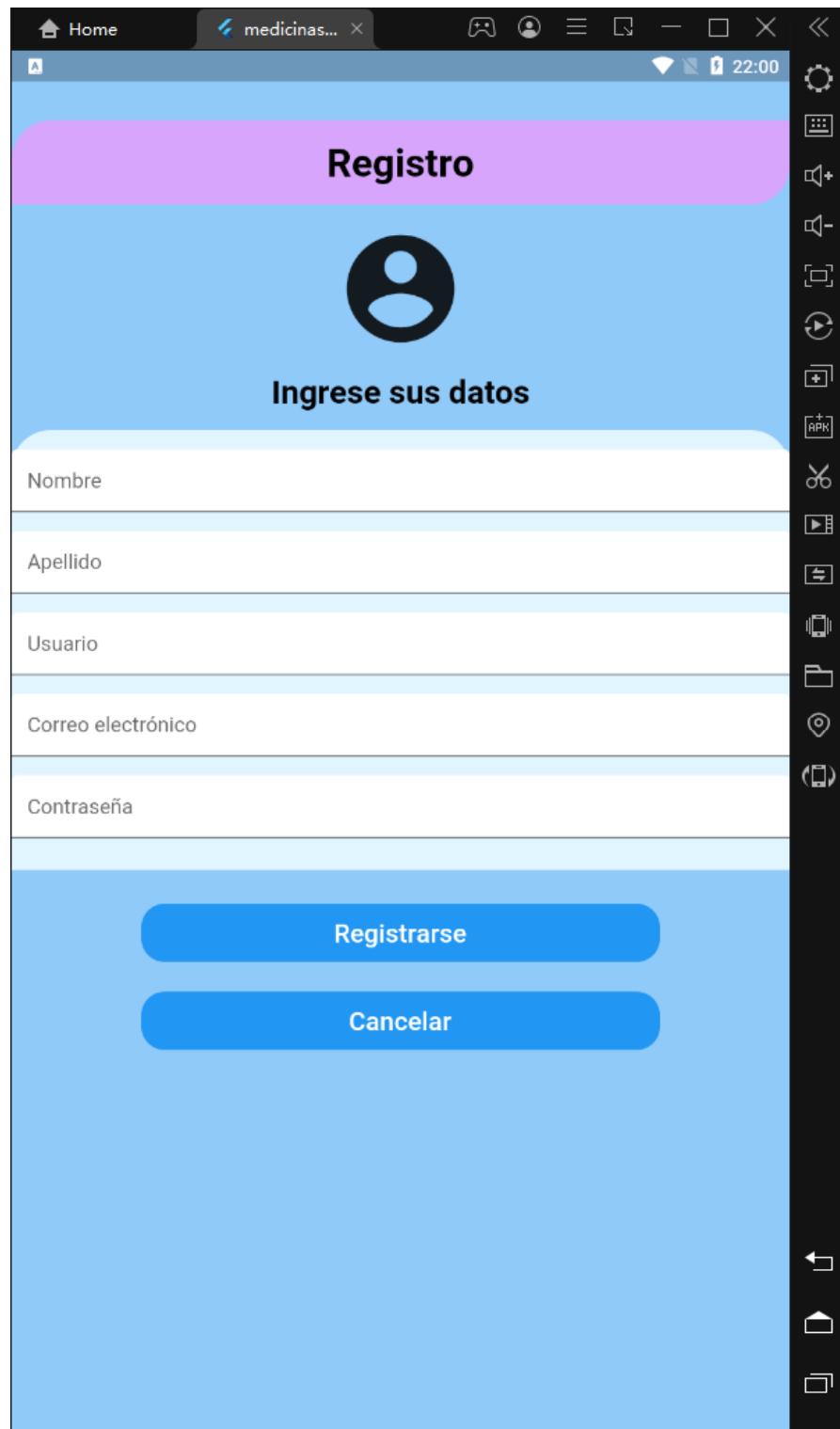


PANTALLAS PROGRAMADAS USANDO FLUTTER

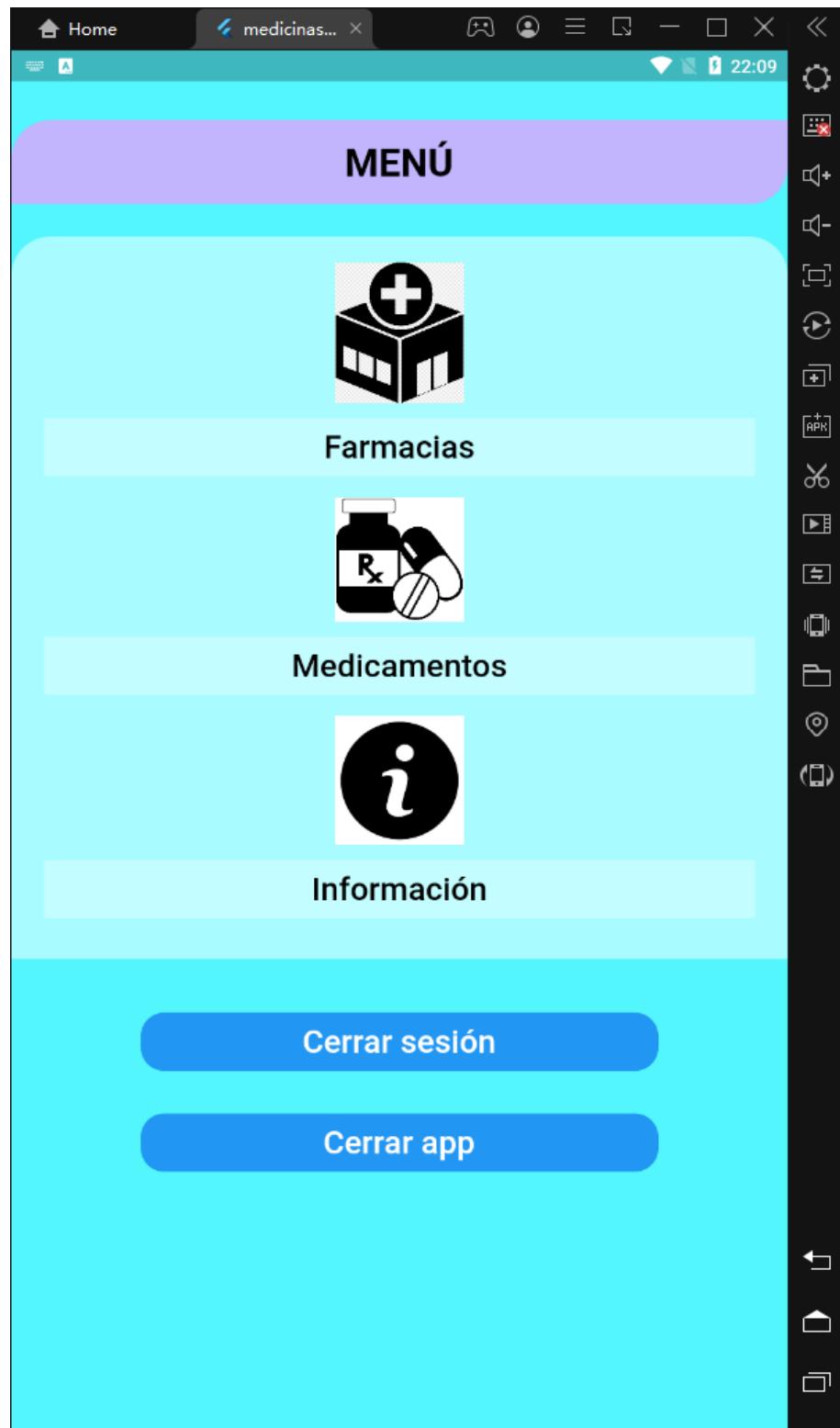
LOGIN



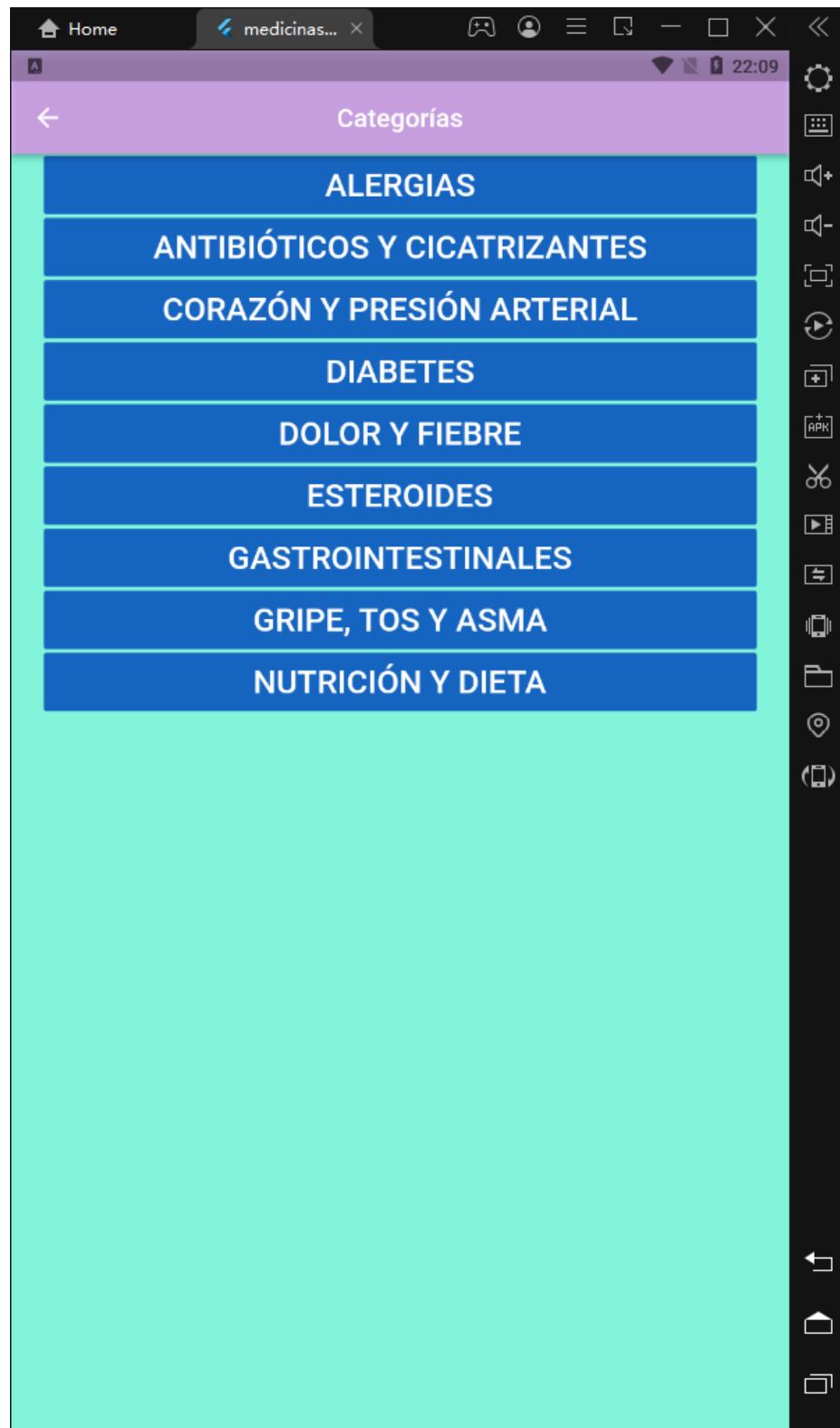
REGISTRO



MENÚ



MEDICAMENTOS



CATEGORÍA - Ejemplo



MEDICAMENTO SELECCIONADO – Ejemplo

AFRIN 0.5MG SOLUCION NASAL
ADULTOS 15ML

ADULTO

Afrin®

Clorhidrato de Oximetazolina
Descongestionante

0.5 mg/ml
Solución
Nasal
Adultos

Efecto rápido
y prolongado
durante 12 horas

Frasco 15 ml

ADULTO

Afrin®

Clorhidrato de Oximetazolina
Descongestionante

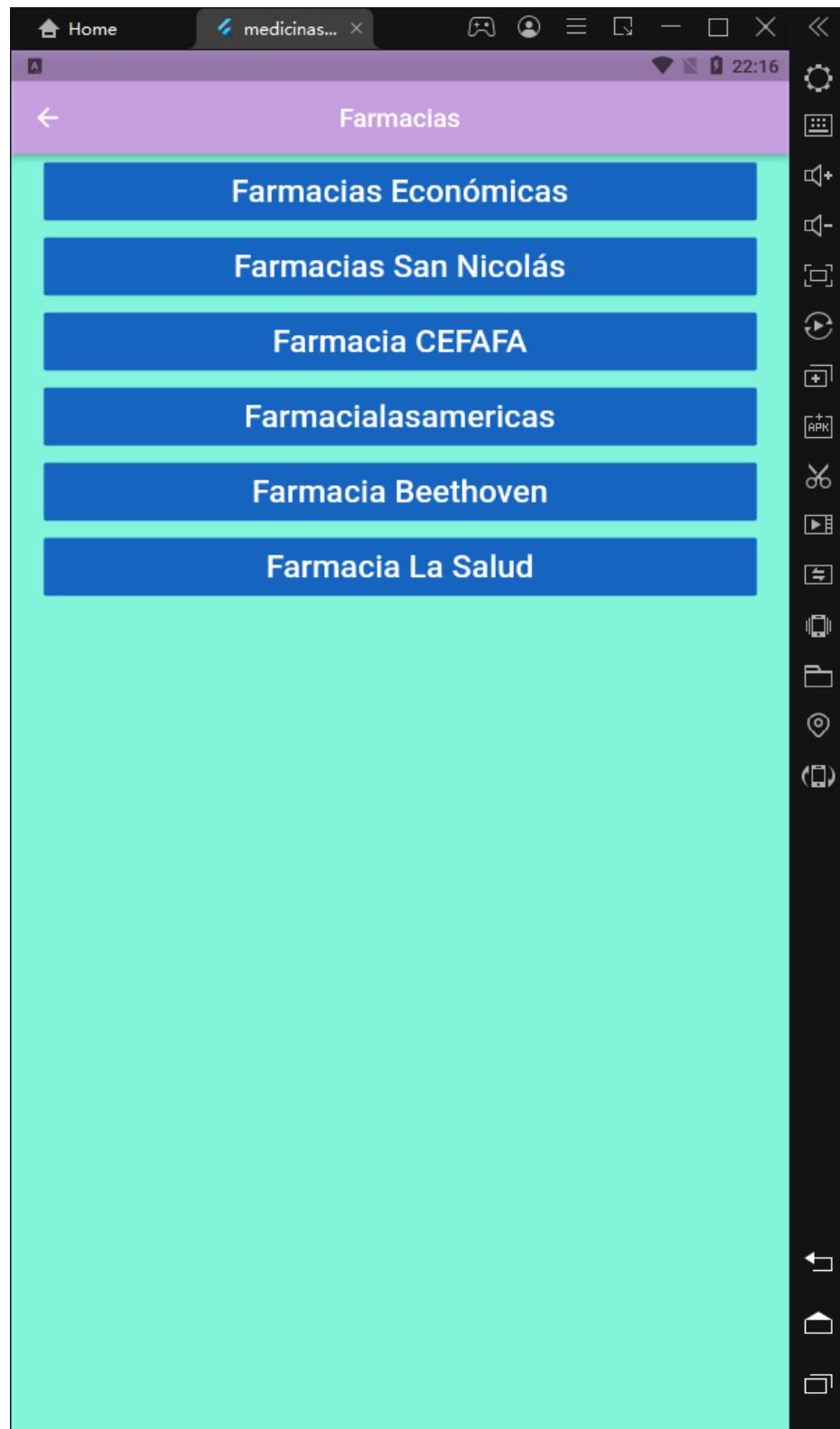
0.5 mg/ml
Solución
Nasal
Adultos

Efecto rápido
y prolongado
durante 12 horas

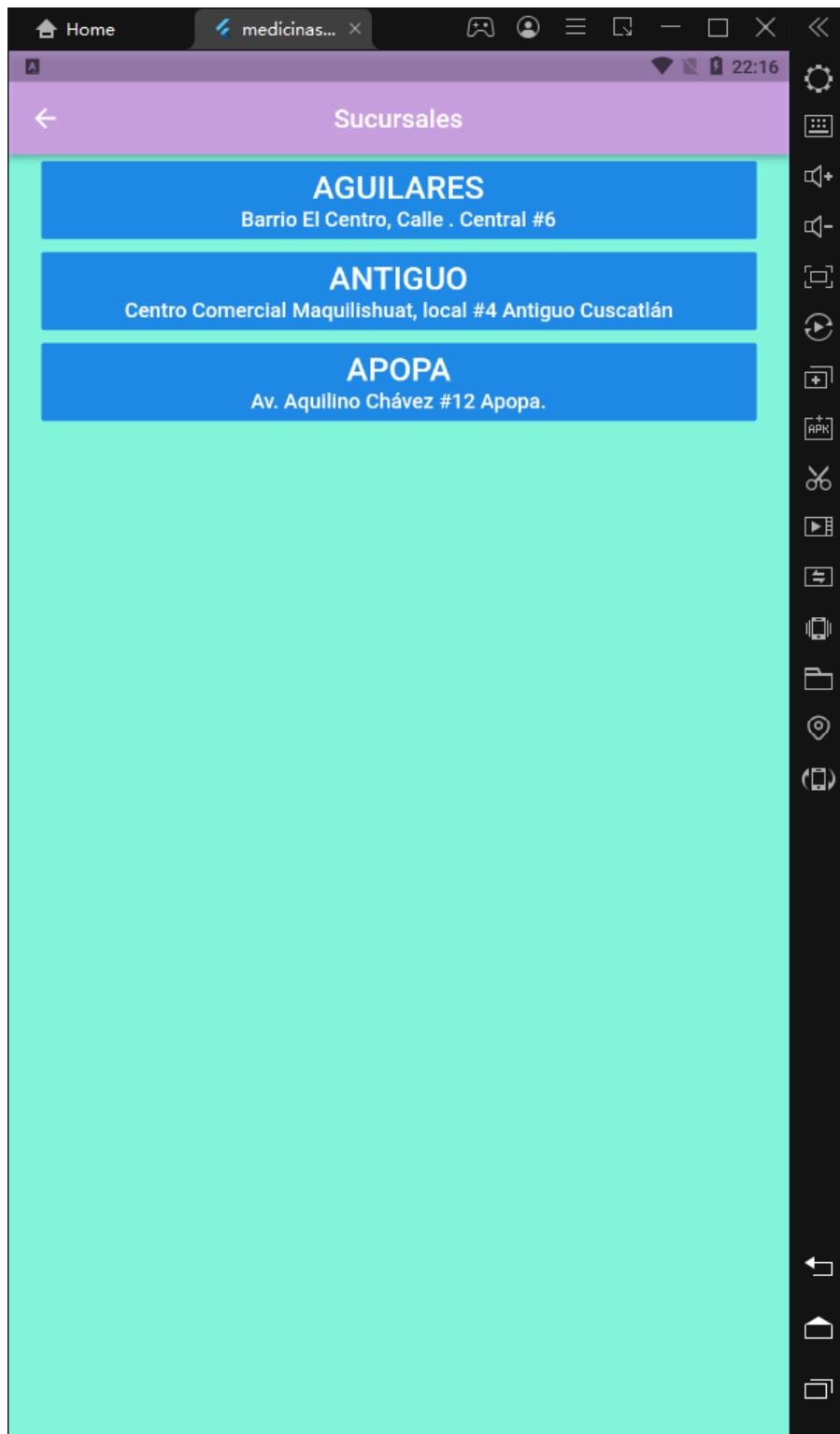
Frasco 15 ml

Es un descongestivo que alivia la congestión nasal por estrechamiento de los vasos sanguíneos de la nariz. Actúa en pocos minutos y su efecto se mantiene durante horas.

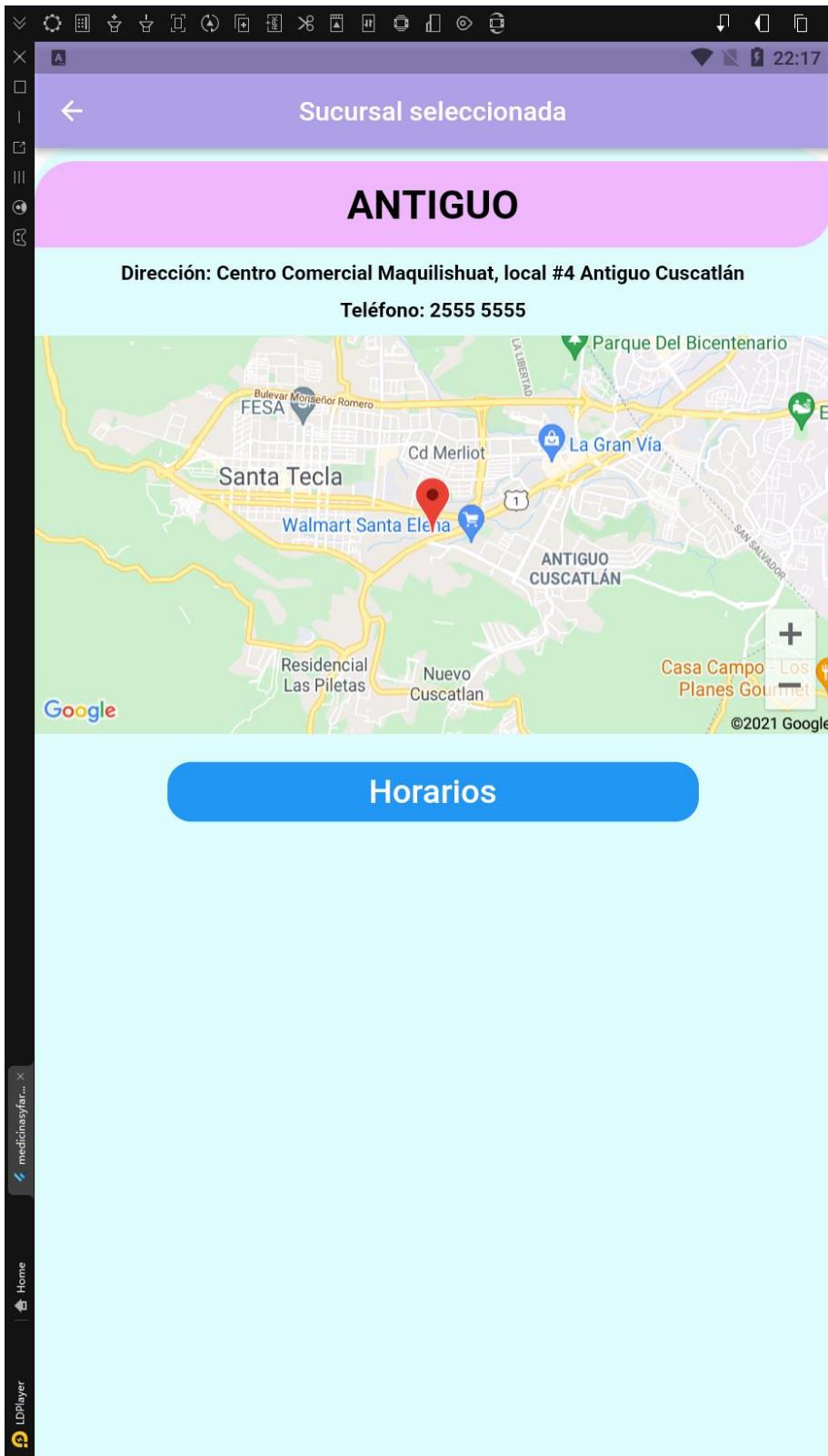
FARMACIAS



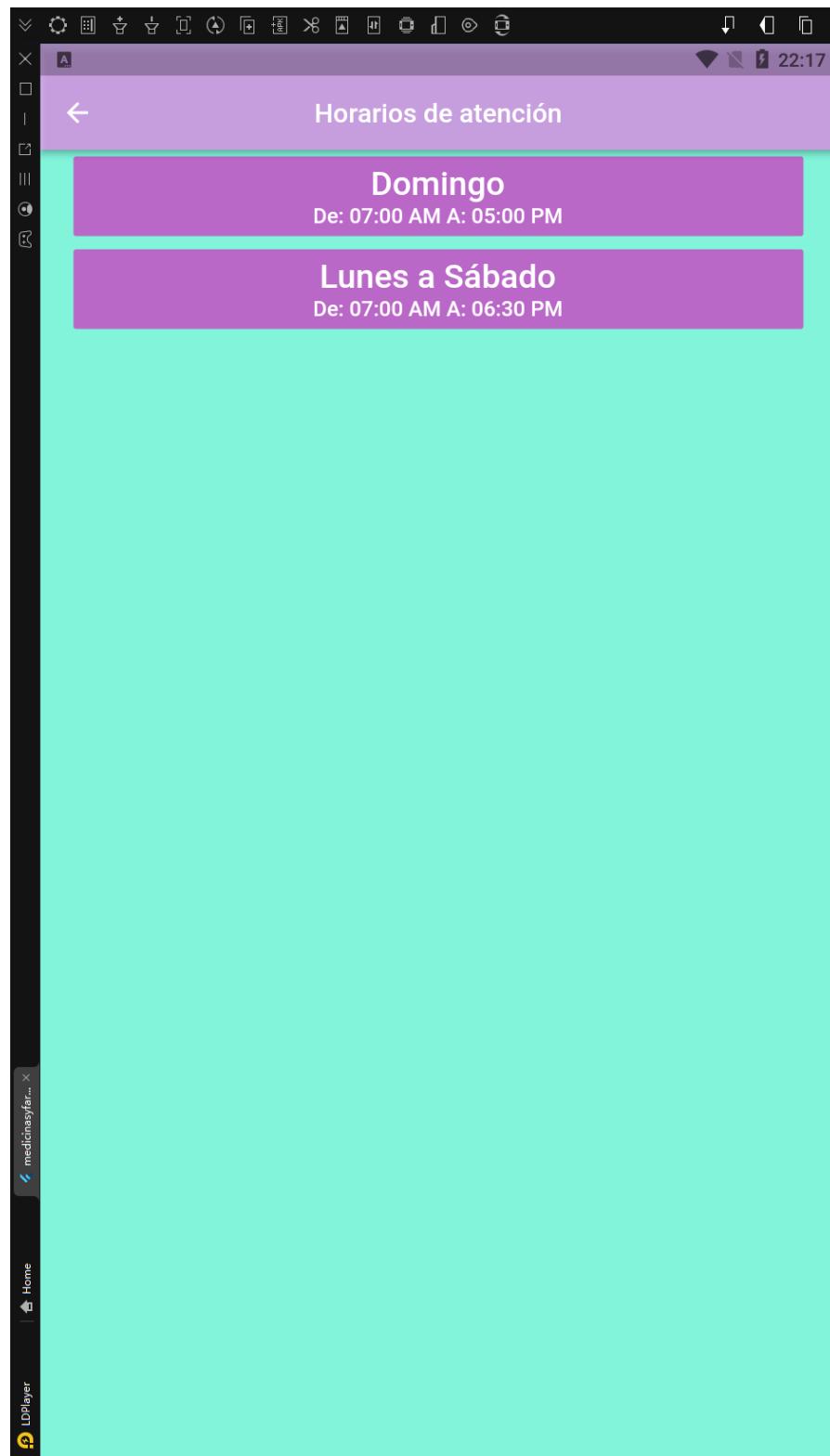
SUCURSALES – Ejemplo



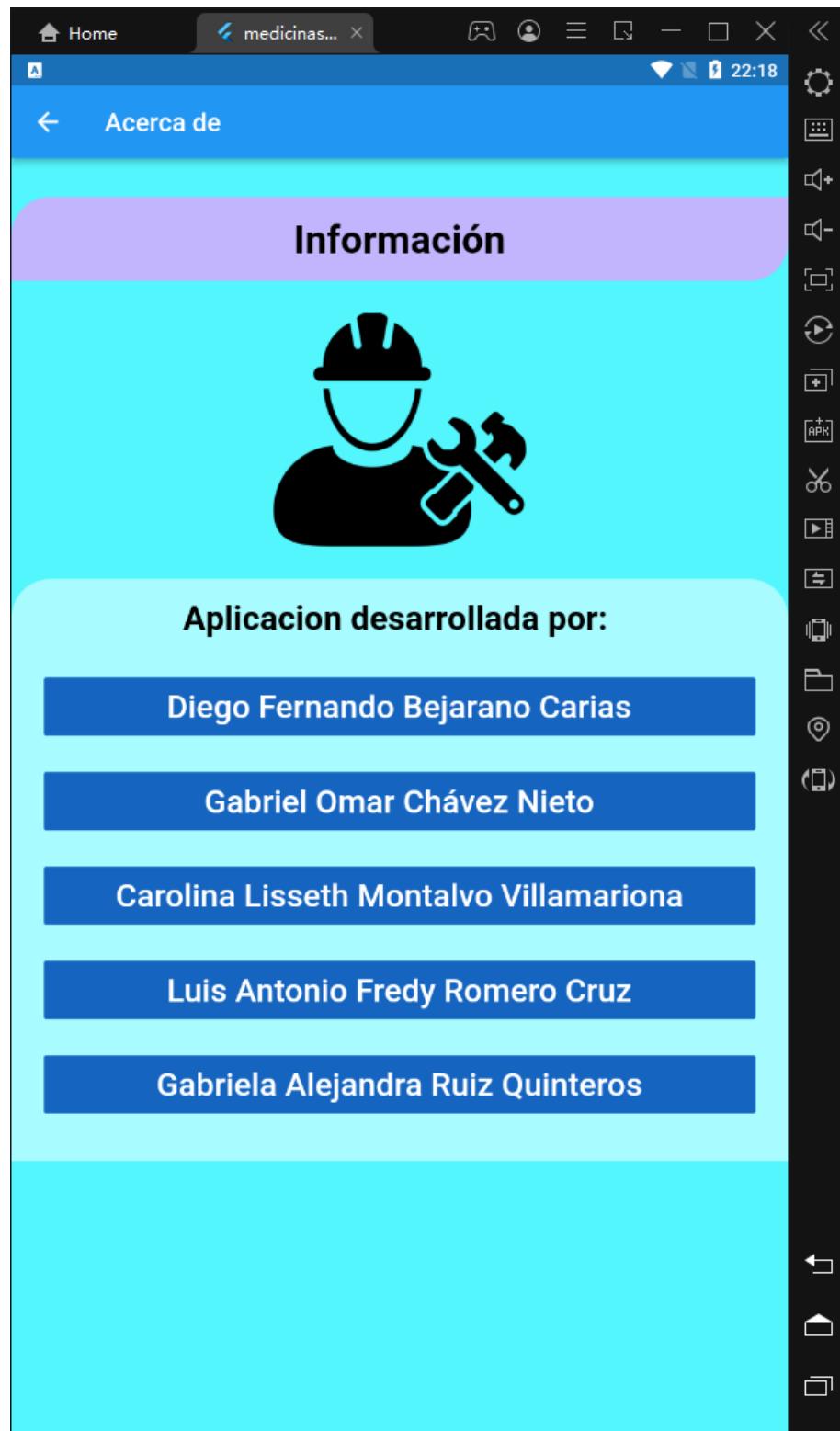
SUCURSAL SELECCIONADA – Ejemplo



HORARIOS DE SUCURSAL SELECCIONADA - Ejemplo

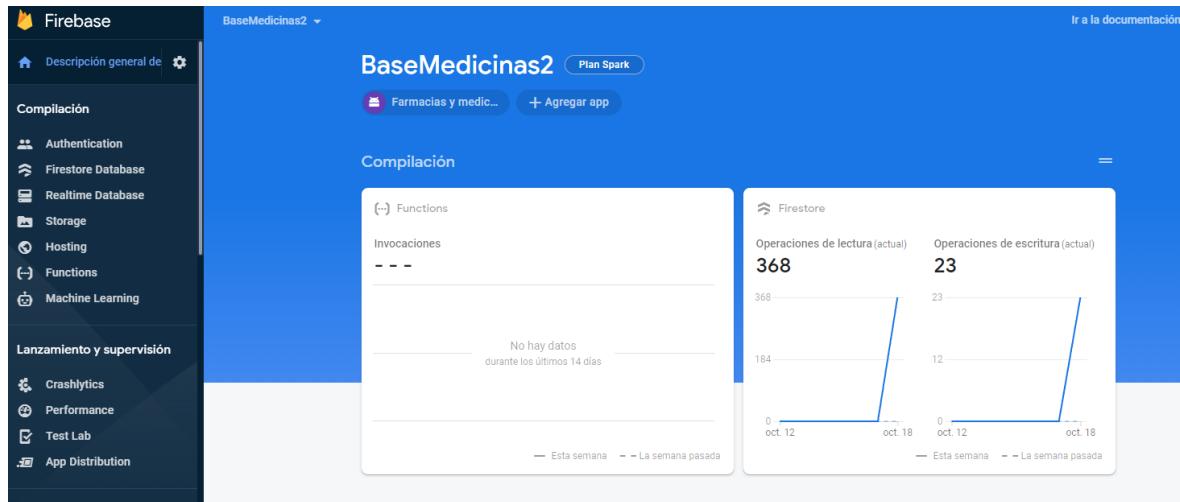


INFORMACIÓN DE LOS DESARROLLADORES

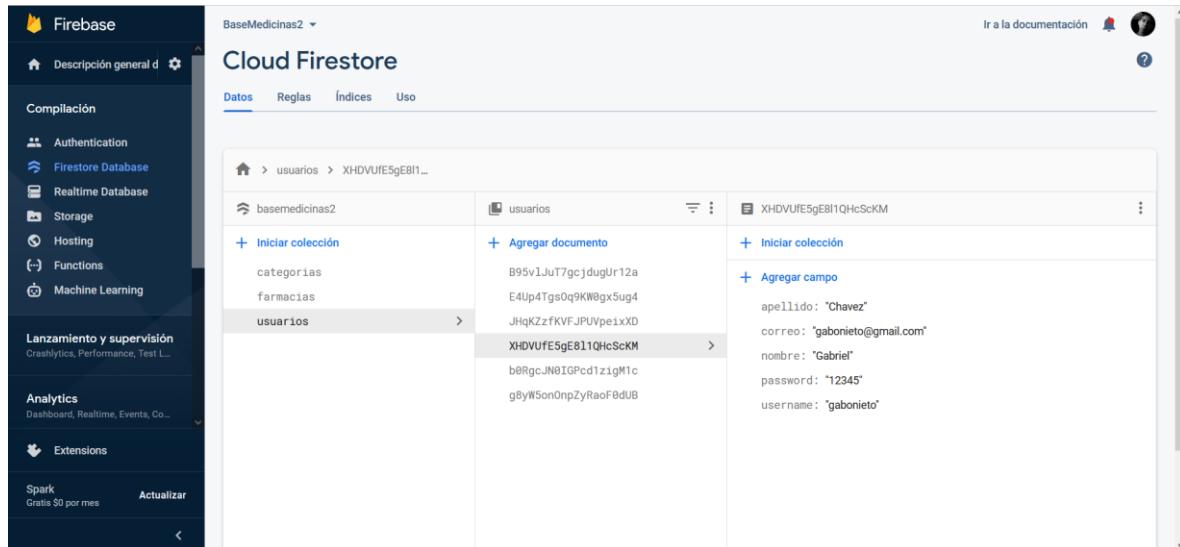


DISEÑO DE LA BASE DE DATOS

Hemos creado la Base de Datos llamada **BaseMedicinas** la cual nos servirá para almacenar todos los datos para nuestra aplicación de Flutter. Ya se ha vinculado la aplicación a la Base de Datos.



En la Base de Datos tenemos la colección llamada **Usuarios** donde se estarán guardando los usuarios que se registren en la aplicación, se guardan por medio de un id y dentro de cada id se encuentran los campos del usuario que serían, Nombre, Apellido, Username, Password y Correo.



En la colección de **Farmacias**, se encuentran registradas las farmacias por medio de un Id y su respectivo nombre.

The screenshot shows the Cloud Firestore interface. On the left is a sidebar with various icons. The main area shows a hierarchical view of collections: 'farmacias' is a child of 'basemedicinas2'. A new document is being added under 'farmacias', with the name 'Farmacias Económicas' entered in the 'Agregar campo' (Add field) section. The 'farmacias' collection also contains a 'categorias' field.

Siguiendo en la colección de **Farmacias**, tenemos el nombre de la farmacia y seguido tenemos la colección de **sucursales**, donde se almacenan las sucursales de la farmacia que se desea ver, mostrando la dirección y el nombre y el teléfono, así como también los campos para la latitud y longitud que conforman las coordenadas de las ubicaciones exactas de cada sucursal.

The screenshot shows the Cloud Firestore interface. It is navigating through the 'farmacias' collection to the 'sucursales' subcollection of document '1'. A new document is being created with the following fields:

- nombre: "Farmacias Económicas"
- direccion: "Calle Morazan, Local # 3 Ciudad Delgado"
- latitud: "13.72178145627624"
- longitud: "-89.16932841763028"
- nombre: "CIUDAD DELGADO"
- telefono: "2530 2518"

Siguiendo en la colección de **Sucursales**, tenemos la colección de **horarios**, donde se muestran los horarios de la sucursal que se desea ver, en esta colección se muestran campos referentes a los días que la sucursal está abierta, la hora de inicio y fin en las que se atiende.

The screenshot shows the Cloud Firestore interface with the 'horarios' collection selected. The collection contains one document with the following fields:

- dia: "Todos los días"
- horafin: "06:00 PM"
- horainicio: "08:00 AM"

En la colección de **Categorías**, se encuentran registradas las categorías por medio de un Id y su respectivo nombre.

The screenshot shows the Cloud Firestore interface with the 'categorias' collection selected. The collection contains one document with the following fields:

- nombre: "Categoria1"

Siguiendo en la colección de **Categorías**, tenemos la colección de **medicinas**, donde se almacenan la descripción, la imagen y el nombre de la medicina.

The screenshot shows the Google Cloud Firestore interface. On the left, there's a sidebar with various icons. The main area has a header 'Cloud Firestore' with tabs for 'Datos', 'Reglas', 'Índices', and 'Uso'. Below the header, the path is shown as: Home > categorias > 1 > medicinas > 1. There are three sections: 1. A left panel with '+ Iniciar colección' and a dropdown menu set to 'medicinas'. 2. A middle panel with '+ Agregar documento' and a list showing document 1 and document 2. 3. A right panel with '+ Iniciar colección' and '+ Agregar campo'. The '+ Agregar campo' section is expanded, showing fields: 'descrip': 'It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using', 'imagen': 'medicina1.jpg', and 'nombre': 'Medicina1'.

Cabe destacar que cada una de las colecciones internas (Sucursales, horarios y medicinas) corresponden a datos independientes para cada uno de los documentos o registros de las colecciones principales (usuarios, farmacias y categorías), dicho esto, se entiende que cada farmacia tendrá sus propias sucursales, y cada sucursal poseerá su registro propio de horarios de atención. Además, cada categoría de medicamentos contendrá su listado con las medicinas que califican para dicha categoría, teniendo en cuenta que estas medicinas sólo pertenecerán a una categoría. En el caso de la colección “usuarios”, esta no contiene ningún nivel de profundidad, por lo cual sólo se cuentan con los documentos o registros únicos para cada usuario.

MANUAL TÉCNICO

Definición del código principal de la Aplicación

1. Pantalla Login

Importando librerías:

```
import 'package:flutter/material.dart';
import 'package:medicinasyfarmacias/database/dbusuarios.dart';
import 'package:medicinasyfarmacias/pantallas/menu.dart';
import 'package:medicinasyfarmacias/pantallas/registro.dart';
import '../funciones/funciones.dart';
```

Clase Principal:

```
class Login extends StatefulWidget {
    const Login({Key? key}) : super(key: key);

    @override
    _LoginState createState() => _LoginState();
}

late DatabaseUsuarios db;
List docs = [];

final _fromKey = GlobalKey<FormState>();
final userController = TextEditingController();
final passController = TextEditingController();
String loggedUser = "";

class _LoginState extends State<Login> {
    initialise() {
        db = DatabaseUsuarios();
        db.initialise();
        db.read().then((value) => {
            setState(() {
                docs = value;
            })
        });
    }
}
```

Iniciando el estado de la conexión:

```
@override  
void initState() {  
    super.initState();  
    initialise();  
}  
  
@override  
Widget build(BuildContext context) {  
    return Scaffold(  
        body: cuerpoP(),  
    );  
}  
}
```

Validando el acceso para el inicio de sesión:

```
bool validarLogin(String user, String pass) {  
    bool result = false;  
    int i = 0;  
    while (i < docs.length) {  
        if (docs[i]['username'] == user && docs[i]['password'] == pass) {  
            result = true;  
            loggedUser = docs[i]['nombre'];  
            i = docs.length;  
        }  
        i++;  
    }  
    return result;  
}
```

Método del widget del cuerpo principal de la pantalla:

```
Widget cuerpoP() {  
    return Container(  
        decoration: const BoxDecoration(  
            image: DecorationImage(  
                image: NetworkImage(  
                    "https://i.pinimg.com/736x/7d/38/96/7d3896c0fa5bb329fba53caa200224a2.jpg"),  
                fit: BoxFit.cover)),  
        child: Center(  
            child: ListView(  
                children: <Widget>[
```

```
const SizedBox(  
    height: 30,  
) ,
```

Título de la pantalla y sus características:

```
tituloPantalla("Login"),  
const SizedBox(  
    height: 15,  
) ,  
space(),  
const SizedBox(  
    height: 15,  
) ,  
const contenedorDat(),  
const SizedBox(  
    height: 25,  
) ,  
],  
) ,  
) ,  
);  
}
```

Método de la clase Dialogo:

```
class Dialogo extends StatelessWidget {  
// ignore: prefer_typing_uninitialized_variables  
final title;  
// ignore: use_key_in_widget_constructors  
const Dialogo(this.title);
```

Método del Widget de aviso y especificando sus características principales y su funcionalidad:

```
@override
Widget build(BuildContext context) {
    return AlertDialog(
        title: const Text("Aviso"),
        content: Text(title),
        actions: [
            FlatButton(
                onPressed: () {
                    Navigator.of(context).pop();
                },
                child: const Text("OK"))
        ],
    );
}

Widget space() {
    // ignore: prefer_const_literals_to_create_immutables
    return Column(children: <Widget>[
        const SizedBox(height: 230),
    ]);
}
```

Método del contenedor principal donde se agrega textos de indicaciones y campos para la validación de los datos del login:

```
class contenedorDat extends StatelessWidget {
    const contenedorDat({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return Container(
            width: double.infinity,
            decoration: const BoxDecoration(
                color: Color(0xFFFFFDE7),
                borderRadius: BorderRadius.only(
                    topLeft: Radius.circular(30),
                    topRight: Radius.circular(30),
                ),
            ),
        ),
    },
}
```

```

alignment: Alignment.center,
child: Form(
  key: _fromKey,
  child: Column(
    children: <Widget>[
      const SizedBox(height: 25),
      const Text(
        "Bienvenido",
        style: TextStyle(
          color: Colors.black,
          fontWeight: FontWeight.bold,
          fontSize: 35,
        ),
      ),
      const SizedBox(height: 10),
      const Text(
        "Ingrese sus credenciales",
        style: TextStyle(
          color: Colors.grey,
          fontWeight: FontWeight.bold,
          fontSize: 20,
        ),
      ),
      const SizedBox(height: 20),
      Container(
        padding: const EdgeInsets.symmetric(horizontal: 25),

```

Método de la indicación, ingreso y validación de la credencial usuario:

```

        child: TextFormField(
          controller: userController,
          validator: (value) {
            if (value!.isEmpty) {
              return "Debe colocar un usuario";
            }
            return null;
          },
          decoration: const InputDecoration(
            border: OutlineInputBorder(
              borderRadius: BorderRadius.all(
                Radius.circular(30.0),
              ),
            ),
          ),

```

```

        hintText: "Ingrese su Usuario",
        fillColor: Colors.white,
        filled: true,
      ),
    ),
  ),
  const SizedBox(height: 15),
  Container(
    padding: const EdgeInsets.symmetric(horizontal: 25),

```

Método de la indicación, ingreso y validación de la credencial contraseña:

```

      child: TextFormField(
        controller: passController,
        validator: (value) {
          if (value!.isEmpty) {
            return "Debe colocar una contraseña";
          }
          return null;
        },
        decoration: const InputDecoration(
          border: OutlineInputBorder(
            borderRadius: BorderRadius.all(
              Radius.circular(30.0),
            ),
          ),
          hintText: "Ingrese su Contraseña",
          fillColor: Colors.white,
          filled: true,
        ),
        obscureText: true,
      ),
    ),
  ),
  const SizedBox(height: 15),

```

Método del botón iniciar sesión se muestra las especificaciones del botón tales como, medida horizontal y vertical, tamaño del botón, el formato de los bordes del botón, la característica que especifica el funcionamiento del botón, luego se agrega el texto que lleva el botón y las características de este mismo como: color, tamaño, posición y estilo del texto:

```
Container(  
    margin: const EdgeInsets.symmetric(horizontal: 100),  
    child: ButtonTheme(  
        minWidth: double.infinity,  
        height: 45.0,  
        child: FlatButton(  
            color: Colors.blue,  
            shape: RoundedRectangleBorder(  
                borderRadius: BorderRadius.circular(18.0),  
            ),  
            //padding: EdgeInsets.symmetric(horizontal: 100, vertical: 15),  
            onPressed: () {
```

Validación de la credencial ingresada:

```
if (_fromKey.currentState!.validate()) {  
    if (validarLogin(  
        userController.text, passController.text)) {  
        Navigator.push(  
            context,  
            MaterialPageRoute(  
                builder: (context) => const MenuApp()));
```

Resultado de ser una credencial valida y texto de respuesta:

```
    showDialog(  
        context: context,  
        builder: (BuildContext context) {  
            return Dialogo("Bienvenid@ " + loggedUser);  
        });  
}
```

Resultado de ser una credencial invalida y texto de respuesta:

```
else {  
    showDialog(  
        context: context,  
        builder: (BuildContext context) {  
            return const Dialogo(  
                title: "Error",  
                message: "Credencial Invalida");  
        });  
}
```

```
        "CREDENCIALES INCORRECTAS");
    });
}
userController.text = "";
passController.text = "";
}
},
child: const Text(
    "Iniciar Sesion",
    style: TextStyle(
        fontSize: 20,
        color: Colors.white,
        fontFamily: "Tahoma",
    ),
),
),
),
),
);
```

En el método se muestra las especificaciones del botón “Registrarse” tales como, medida horizontal y vertical, tamaño del botón, el formato de los bordes del botón, la característica que especifica el funcionamiento del botón, luego se agrega el texto que lleva el botón y las características de este mismo como: color, tamaño, posición y estilo del texto:

```
const SizedBox(height: 10),  
const Text(  
    "0",  
    style: TextStyle(  
        color: Colors.black,  
        fontWeight: FontWeight.bold,  
        fontSize: 15,  
    ),  
,  
const SizedBox(height: 10),  
FlatButton(  
    onPressed: () {  
        Navigator.push(context,  
            MaterialPageRoute(builder: (context) => const Registro()));  
    },  
    child: const Text(  
        "Registrarse",  
        style: TextStyle(  
            fontSize: 20,  
            color: Colors.blue,
```

```

        fontFamily: "Tahoma",
    ),
),
),
),
const SizedBox(height: 15),
],
),
),
),
);
}
}
}

```

2. Pantalla Registro

Importando Librerías:

```

import 'package:flutter/material.dart';
import 'package:medicinasyfarmacias/database/dbusuarios.dart';
import '../funciones/funciones.dart';

```

Clase principal de la pantalla:

```

class Registro extends StatelessWidget {
const Registro({Key? key}) : super(key: key);

@Override
Widget build(BuildContext context) {
    return const Scaffold(
        body: cuerpoP(),
    );
}
}

late DatabaseUsuarios db;
class cuerpoP extends StatefulWidget {
const cuerpoP({Key? key}) : super(key: key);

@Override
State<cuerpoP> createState() => _cuerpoPState();
}
class _cuerpoPState extends State<cuerpoP> {

```

```

List docs = [];
initialise() {
    db = DatabaseUsuarios();
    db.initialise();
}
@Override
void initState() {
    super.initState();
    initialise();
}

```

Widget de Contenedor principal de la pantalla, agregando las características generales:

```

@Override
Widget build(BuildContext context) {
    return Container(
        decoration: const BoxDecoration(
            color: Color(0xFF90CAF9),
        ),
        child: Center(
            child: ListView(
                children: <Widget>[
                    const SizedBox(
                        height: 30,
                    ),

```

Métodos que serán usados en la pantalla como el ícono, campo de datos, y los botones:

```

                tituloPantalla("Registro"),
                const SizedBox(
                    height: 15,
                ),
                campoIcono(),
                const SizedBox(
                    height: 15,
                ),
                campoDatos(),
                const SizedBox(
                    height: 15,
                ),
                const contenedorDat(),
                const SizedBox(

```

```

        height: 25,
    ),
    const botonRegistrarse(),
    const SizedBox(
        height: 20,
    ),
    const botonCancelar(),
    const SizedBox(
        height: 30,
    ),
),
],
),
),
);
}
}

```

Metodo del Icono:

```

Widget campoIcono() {
    return const Icon(
        Icons.account_circle,
        size: 100,
    );
}

```

Método del campo de texto donde se ingresaran los datos requeridos en los campos indicados y se plantean las características visuales de los contenedores y botones:

```

Widget campoDatos() {
    return const Text(
        "Ingrese sus datos",
        textAlign: TextAlign.center,
        style: TextStyle(
            color: Colors.black, fontSize: 25.0, fontWeight: FontWeight.bold),
    );
}

final _fromKey = GlobalKey<FormState>();
final nombreController = TextEditingController();
final apellidoController = TextEditingController();
final usuarioController = TextEditingController();
final correoController = TextEditingController();
final contraseniaController = TextEditingController();

```

```

class contenedorDat extends StatelessWidget {
  const contenedorDat({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      width: double.infinity,
      decoration: const BoxDecoration(
        color: Color(0xFFE1F5FE),
        borderRadius: BorderRadius.only(
          topLeft: Radius.circular(30),
          topRight: Radius.circular(30),
        ),
      ),
      alignment: Alignment.center,
      child: Form(
        key: _fromKey,
        child: Column(
          children: <Widget>[
            const SizedBox(height: 15),

```

Validando el registro del campo Nombre y agregando las características del contenedor del dato:

```

        TextFormField(
          controller: nombreController,
          validator: (value) {
            if (value!.isEmpty) {
              return "Debe colocar un nombre";
            }
            return null;
          },
          decoration: const InputDecoration(
            fillColor: Colors.white,
            filled: true,
            hintText: "Nombre",
          ),
        ),
        const SizedBox(height: 15),

```

Validando el registro del campo Apellido y agregando las características del contenedor del dato:

```
TextField(  
    controller: apellidoController,  
    validator: (value) {  
        if (value!.isEmpty) {  
            return "Debe colocar un apellido";  
        }  
        return null;  
    },  
    decoration: const InputDecoration(  
        fillColor: Colors.white,  
        filled: true,  
        hintText: "Apellido",  
    ),  
,  
    const SizedBox(height: 15),
```

Validando el registro del campo Usuario y agregando las características del contenedor del dato:

```
TextField(  
    controller: usuarioController,  
    validator: (value) {  
        if (value!.isEmpty) {  
            return "Debe colocar un usuario";  
        }  
        return null;  
    },  
    decoration: const InputDecoration(  
        fillColor: Colors.white,  
        filled: true,  
        hintText: "Usuario",  
    ),  
,  
    const SizedBox(height: 15),
```

Validando el registro del campo Correo y agregando las características del contenedor del dato:

```
TextField(  
    controller: correoController,  
    validator: (value) {  
        if (value!.isEmpty) {  
            return "Debe colocar un correo";  
        }  
        return null;  
    },  
    const SizedBox(height: 15),
```

```

        },
        return null;
    },
    decoration: const InputDecoration(
        fillColor: Colors.white,
        filled: true,
        hintText: "Correo electrónico",
    ),
),
const SizedBox(height: 15),

```

Validando el registro del campo Contraseña y agregando las características del contenedor del dato:

```

TextField(
    obscureText: true,
    controller: contraseniaController,
    validator: (value) {
        if (value!.isEmpty) {
            return "Debe colocar una contraseña";
        }
        return null;
    },
    decoration: const InputDecoration(
        fillColor: Colors.white,
        filled: true,
        hintText: "Contraseña",
    ),
),
const SizedBox(height: 25),
],
),
),
),
);
}
}

```

Método para programar el botón de registrarse y especificar las características de este el color del botón, el tamaño, el estilo del borde del botón, se agrega la función característica del botón y agregar el texto del botón luego se especifican las características del texto, el tamaño de la letra, color y estilo que llevara el

texto, así mismo se valida la funcionalidad del botón guardando todos los datos ingresados:

```
class botonRegistrarse extends StatelessWidget {  
    const botonRegistrarse({Key? key}) : super(key: key);  
  
    @override  
    Widget build(BuildContext context) {  
        return Container(  
            margin: const EdgeInsets.symmetric(horizontal: 100),  
            child: ButtonTheme(  
                minWidth: double.infinity,  
                height: 45.0,  
                child: FlatButton(  
                    color: Colors.blue,  
                    shape: RoundedRectangleBorder(  
                        borderRadius: BorderRadius.circular(18.0),  
                    ),  
                ),  
        );  
    }  
}
```

Validando el registro de los datos ingresando correctamente:

```
//padding: EdgeInsets.symmetric(horizontal: 100, vertical: 15),  
onPressed: () {  
    if (_fromKey.currentState!.validate()) {  
        db.create(  
            nombreController.text,  
            apellidoController.text,  
            usuarioController.text,  
            correoController.text,  
            contraseniaController.text,  
        );  
        Navigator.pop(context, true);  
        showDialog(  
            context: context,  
            builder: (BuildContext context) {  
                return const Dialogo("Usuario registrado exitosamente");  
            });  
    }  
},  
child: const Text(  
    "Registrarse",  
    style: TextStyle(  
        fontSize: 20,  
        color: Colors.white,
```

```

        fontFamily: "Tahoma",
    ),
}),
),
),
),
);
}
}

class Dialogo extends StatelessWidget {
// ignore: prefer_typing_uninitialized_variables
final title;
// ignore: use_key_in_widget_constructors
const Dialogo(this.title);

@Override
Widget build(BuildContext context) {
    return AlertDialog(
        title: const Text("Aviso"),
        content: Text(title),
        actions: [
            FlatButton(
                onPressed: () {
                    Navigator.of(context).pop();
                },
                child: const Text("OK"))
        ],
    );
}
}

```

Método para programar el botón Cancelar y especificar las características de este el color del botón, el tamaño, el estilo del borde del botón, se agrega la función característica del botón y agregar el texto del botón luego se especifican las características del texto, el tamaño de la letra, color y estilo que llevara el texto, así mismo se valida la funcionabilidad del botón al cancelar el registro de los datos que se habían ingresado:

```

class botonCancelar extends StatelessWidget {
    const botonCancelar({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {

```

```

        return Container(
            margin: const EdgeInsets.symmetric(horizontal: 100),
            child: ButtonTheme(
                minWidth: double.infinity,
                height: 45.0,
                child: FlatButton(
                    color: Colors.blue,
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(18.0),
                    ),
                    //padding: EdgeInsets.symmetric(horizontal: 100, vertical: 15),
                    onPressed: () {
                        Navigator.pop(context);
                    },
                    child: const Text(
                        "Cancelar",
                        style: TextStyle(
                            fontSize: 20,
                            color: Colors.white,
                            fontFamily: "Tahoma",
                        ),
                    )));
        ),
    );
}
}

```

3. Pantalla Menú

Importando Librerias:

```

import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import '../funciones/funciones.dart';
import '../pantallas/farmacias.dart';
import '../pantallas/categorias.dart';
import '../pantallas/informacion.dart';

```

Método de la Clase Principal:

```

class MenuApp extends StatefulWidget {
    const MenuApp({Key? key}) : super(key: key);

```

```

@Override
_MenuAppState createState() => _MenuAppState();
}

class _MenuAppState extends State<MenuApp> {
@Override
Widget build(BuildContext context) {
return Scaffold(
body: cuerpoMenu(),
);
}
}

Widget cuerpoMenu() {
return Container(
decoration: const BoxDecoration(
color: Color(0xAA00F4FF),
),
child: Center(
child: ListView(
children: <Widget>[
const SizedBox(
height: 30,
),

```

Título de la Pantalla:

```

tituloPantalla("MENÚ"),
const SizedBox(
height: 25,
),
```

Método de Opciones del Menú:

```

opcionesMenu(),
const SizedBox(
height: 40,
),
```

Método del Botón Cerrar Sesión:

```

const botonCerrarSesion(),
const SizedBox(
height: 30,
),
```

Método del Botón Cerrar App:

```

const botonCerrarApp(),
const SizedBox(
```

```

        height: 30,
    ),
],
),
),
);
}
}

```

Método del Widget donde se desarrollan las opciones del Menú, y se detallan las características visuales del contenedor en que se muestra:

```

Widget opcionesMenu() {
    return Container(
        width: double.infinity,
        decoration: const BoxDecoration(
            color: Color(0xAAD2FEFF),
            borderRadius: BorderRadius.only(
                topLeft: Radius.circular(30),
                topRight: Radius.circular(30),
            ),
        ),
        alignment: Alignment.center,
        child: Column(
            children: <Widget>[
                const SizedBox(height: 20),
                Column(
                    children: <Widget>[
                        Image.asset('assets/img/farmacias.png', width: 100)
                    ],
                ),
            ],
        ),
    );
}

```

Opción Farmacias:

```

const SizedBox(height: 10),
const botonOpciones(nombreBoton: 'Farmacias'),
const SizedBox(height: 15),
Column(
    children: <Widget>[
        Image.asset('assets/img/medicamentos.png', width: 100)
    ],
),

```

Opción Medicamentos:

```

const SizedBox(height: 10),
const botonOpciones(nombreBoton: 'Medicamentos'),
const SizedBox(height: 15),

```

```

Column(
  children: <Widget>[
    Image.asset('assets/img/informacion.jpg', width: 100)
  ],
),

```

Opción Información:

```

const SizedBox(height: 10),
const botonOpciones(nombreBoton: 'Información'),
const SizedBox(height: 30),
],
),
);
}
class botonOpciones extends StatelessWidget {
const botonOpciones({Key? key, required this.nombreBoton}) : super(key: key);
final String nombreBoton;
@Override
Widget build(BuildContext context) {
return Container(
margin: const EdgeInsets.symmetric(horizontal: 25),
child: ButtonTheme(
minWidth: double.infinity,
height: 45.0,

```

Mostrando las nuevas Opciones: Categorías e Información al validarse que la primera opción seleccionada fue la de medicamentos:

```

child: FlatButton(
color: const Color(0xAAD2FEFF),
onPressed: () {
if (nombreBoton == "Farmacias") {
Navigator.push(context,
MaterialPageRoute(builder: (context) => const Farmacias()));
} else if (nombreBoton == "Medicamentos") {
Navigator.push(
context,
MaterialPageRoute(
builder: (context) => const Categorias()));
} else {
Navigator.push(
context,
MaterialPageRoute(

```

```

        builder: (context) => const Informacion())));
    }
},
child: Text(
  nombreBoton,
  style: const TextStyle(
    fontSize: 25,
    color: Colors.black,
    fontFamily: "Tahoma",
  ),
),
)),
),
);
}
}
}

```

Metodo donde se desarrolla el botón Cerrar Sesión:

```

class botonCerrarSesion extends StatelessWidget {
  const botonCerrarSesion({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: const EdgeInsets.symmetric(horizontal: 100),
      child: ButtonTheme(
        minWidth: double.infinity,
        height: 45.0,

```

Validando el resultado de la opción del Funcionamiento al presionar el botón de Cerrar Sesión:

```

        child: FlatButton(
          color: Colors.blue,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(18.0),
          ),
          onPressed: () {
            Navigator.of(context).pop();
            //Navigator.push(context,
            //  MaterialPageRoute(builder: (context) => const Login()));
          },
          child: const Text(
            "Cerrar sesión",
            style: TextStyle(

```

```

        fontSize: 25,
        color: Colors.white,
        fontFamily: "Tahoma",
    ),
),
)),
),
);
}
}

```

Metodo donde se desarrolla el botón Cerrar App:

```

class botonCerrarApp extends StatelessWidget {
    const botonCerrarApp({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return Container(
            margin: const EdgeInsets.symmetric(horizontal: 100),
            child: ButtonTheme(
                minWidth: double.infinity,
                height: 45.0,

```

Validando el resultado de la opción del Funcionamiento al presionar el botón de Cerrar App:

```

                child: FlatButton(
                    color: Colors.blue,
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(18.0),
                    ),
                    onPressed: () {
                        SystemNavigator.pop();
                    },
                    child: const Text(
                        "Cerrar app",
                        style: TextStyle(
                            fontSize: 25,
                            color: Colors.white,
                            fontFamily: "Tahoma",
                        ),
                    )));
},
);
}
}

```

4. Pantalla Medicamentos

Importando Librerías:

```
import 'package:flutter/material.dart';
import 'package:medicinasyfarmacias/database/dbcategorias.dart';
import 'package:medicinasyfarmacias/pantallas/cat_seleccionada.dart';
```

Metodo De la Clase Principal:

```
class Categorias extends StatefulWidget {
  const Categorias({Key? key}) : super(key: key);

  @override
  _CategoriasState createState() => _CategoriasState();
}
```

Método donde se desarrollan las categorías de los medicamentos:

```
class _CategoriasState extends State<Categorias> {
  late DatabaseCategorias db;
  List docs = [];
  initialise() {
    db = DatabaseCategorias();
    db.initialise();
    db.read().then((value) => {
      setState(() {
        docs = value;
      })
    });
  }
}

@Override
void initState() {
  super.initState();
  initialise();
}
```

Widget don se están Agregando las características visuales de la pantalla y los elementos de esta misma :

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color(0xFF81F4DA),
    appBar: AppBar(
      centerTitle: true,
      //color de fondo del appbar
      backgroundColor: const Color(0xAAFA94FD),
      //Texto "Farmacias"
      title: const Text("Categorías"),
    ),
    body: ListView.builder(
      itemCount: docs.length,
      itemBuilder: (BuildContext context, int index) {
```

Contenedor donde se desarrolla el boton que dirigira hacia la ctegoria seleccionada:

```
  return Container(
    margin: const EdgeInsets.symmetric(horizontal: 25),
    child: ButtonTheme(
      minWidth: double.infinity,
      height: 45.0,
      child: FlatButton(
        color: Colors.blue[800],
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => CategoriaSeleccionada(
                categoria: docs[index], db: db)));
        }
      )
    )
  );
}
```

```
 },
```

Se desarrollan las características visuales del texto del nombre de la categoría que se seleccione:

```
        child: Text(
            docs[index]['nombre'],
            style: const TextStyle(
                fontSize: 25,
                color: Colors.white,
                fontFamily: "Tahoma",
            ),
        )),  
    ),  
);  
,  
);  
}  
}
```

5. Pantalla Categoría Seleccionada

Importando Librerias:

```
import 'package:flutter/material.dart';
import 'package:medicinasyfarmacias/database/dbcategorias.dart';
import 'package:medicinasyfarmacias/pantallas/medicamento_select.dart';
```

Método de La clase Principal:

```
class CategoriaSeleccionada extends StatefulWidget {
    CategoriaSeleccionada({Key? key, required this.categoría, required this.db})
        : super(key: key);
    Map categoría;
    DatabaseCategorías db;
    @override
    _CategoriaSeleccionadaState createState() => _CategoriaSeleccionadaState();
}
```

Método donde se desarrolla el funcionamiento y muestra de elementos de la categoría seleccionada:

```
class _CategoriaSeleccionadaState extends State<CategoriaSeleccionada> {
    late DatabaseCategorias db;
    List docs = [];
    initialise() {
        db = DatabaseCategorias();
        db.initiliase();
        db.readMedicinas(widget.categoría['id']).then((value) => {
            setState(() {
                docs = value;
            })
        });
    }
}

@Override
void initState() {
    super.initState();
    initialise();
}
```

Método del Widget donde agregan las características visuales de los elementos correspondientes y se muestra el listado de medicamentos pertenecientes a esta categoría seleccionada:

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor: const Color(0xFF81F4DA),
        appBar: AppBar(
            centerTitle: true,
            backgroundColor: const Color(0xAAFA94FD),
            title: const Text("Medicamentos"),
        ),
        body: ListView.builder(
            itemCount: docs.length,
            itemBuilder: (BuildContext context, int index) {
                return Container(
                    alignment: Alignment.center,
                    margin: const EdgeInsets.symmetric(horizontal: 25),
                    padding: const EdgeInsets.symmetric(vertical: 5),
                    child: ButtonTheme(
```

```
        minWidth: double.infinity,  
        height: 45.0,
```

Método del botón que permite la navegación y selección de los medicamentos visibles en esta categoría:

```
child: FlatButton(  
            color: Colors.blue[600],  
            onPressed: () {  
                Navigator.push(  
                    context,  
                    MaterialPageRoute(  
                        builder: (context) =>  
                            MedSeleccionado(medicina: docs[index], db: db)));  
            },  
            child: Text(  
                docs[index]['nombre'],  
                style: const TextStyle(  
                    fontSize: 25,  
                    color: Colors.white,  
                    fontFamily: "Tahoma",  
                ),  
            ),  
        ),  
    ),  
),  
);  
},  
);  
}  
}
```

6. Pantalla Medicamento Seleccionado

Importando Librerias:

```
import 'package:firebase_storage/firebase_storage.dart';  
import 'package:flutter/material.dart';  
import 'package:medicinasyfarmacias/database/dbcategorias.dart';  
import 'package:medicinasyfarmacias/funciones/funciones.dart';
```

Metodo de la Clase Principal:

```
class MedSeleccionado extends StatefulWidget {  
    MedSeleccionado({Key? key, required this.medicina, required this.db})  
        : super(key: key);  
    Map medicina;  
    DatabaseCategorias db;  
    @override  
    _MedSeleccionadoState createState() => _MedSeleccionadoState();  
}  
}
```

Declaracion de Variables:

```
String nombreM = "";  
String descripcionM = "";  
String imagenM = "";
```

Método de la de la clase donde se desarrolla el contenido y al información del medicamento seleccionado:

```
class _MedSeleccionadoState extends State<MedSeleccionado> {  
    @override  
    void initState() {  
        super.initState();  
        nombreM = widget.medicina['nombre'];  
        descripcionM = widget.medicina['descrip'];  
        imagenM = widget.medicina['imagen'];  
    }  
  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            appBar: AppBar(  
                centerTitle: true,  
                backgroundColor: const Color(0xAAAA94FF),  
                title: const Text("Medicamento seleccionado"),  
            ),  
            body: Container(  
                padding: const EdgeInsets.all(20.0),  
                width: double.infinity,
```

Se centra el título que esta contenido dentro del appbar y se agregan otras características de diseño:

```
                ),  
            ),  
            padding: const EdgeInsets.all(20.0),  
            width: double.infinity,
```

```

decoration: const BoxDecoration(
  color: Color(0xAAD2FEFF),
  borderRadius: BorderRadius.only(
    topLeft: Radius.circular(30),
    topRight: Radius.circular(30),
  ),
),
),

```

Se agrega un listview donde se visualizara los detalles del medicamento, y se modifican las características de diseño de este mismo:

```

alignment: Alignment.topCenter,
child: ListView(
  children: <Widget>[
    const SizedBox(height: 15),
    tituloPantalla(nombreM),
    const SizedBox(
      height: 20,
    ),
),

```

Acá se está agregando la imagen de referencia relacionada con el medicamento seleccionado y se modifican las características de la imagen:

```

FutureBuilder(
  future: _getImage(context, imagenM),
  builder: (BuildContext context, AsyncSnapshot<Widget> snapshot) {
    if (snapshot.connectionState == ConnectionState.done) {
      return SizedBox(
        width: MediaQuery.of(context).size.width / 1.2,
        height: MediaQuery.of(context).size.width / 1.2,
        child: snapshot.data,
      );
    }
    if (snapshot.connectionState == ConnectionState.waiting) {
      return SizedBox(
        width: MediaQuery.of(context).size.width / 1.2,
        height: MediaQuery.of(context).size.width / 1.2,
        child: const CircularProgressIndicator(),
      );
    }
    return Container();
  },
),
const SizedBox(

```

```
        height: 15,  
    ),
```

Método donde se agrega la descripción del medicamento seleccionado:

```
    Text(  
        descripcionM,  
        style: const TextStyle(  
            color: Colors.black,  
            fontWeight: FontWeight.normal,  
            fontSize: 20,  
        ),  
        ),  
    ],  
,  
,  
),  
resizeToAvoidBottomInset: false,  
);  
}  
}
```

Método donde se validan los datos relacionados a los registros de medicamentos en Firebase:

```
Future<Widget> _getImage(BuildContext context, String imageName) async {  
    late Image image;  
    await FireStoreService.loadImage(context, imageName).then((value) {  
        image = Image.network(  
            value.toString(),  
            fit: BoxFit.scaleDown,  
        );  
    });  
    return image;  
}  
class FireStoreService extends ChangeNotifier {  
    FireStoreService();  
    static Future<dynamic> loadImage(BuildContext context, String Image) async {  
        return await FirebaseStorage.instance.ref().child(Image).getDownloadURL();  
    }  
}
```

7. Pantalla Farmacias

Importando las librerías:

```
import 'package:flutter/material.dart';
import 'package:medicinasyfarmacias/database/dbfarmacias.dart';
import 'package:medicinasyfarmacias/pantallas/sucursales.dart';
```

El StatefulWidget que conocemos, aquí se crea toda la metodología donde se interactúa con la base de datos, se necesita un contexto dentro del método donde se realizar la consulta a la base de datos, por eso fue más :

```
class Farmacias extends StatefulWidget {
  const Farmacias({Key? key}) : super(key: key);

  @override
  _FarmaciasState createState() => _FarmaciasState();
}
```

```
class _FarmaciasState extends State<Farmacias> {
  Elementos para lograr la conexión con la base de datos, usando el archivo
  dart "dbFarmacias"
  el cual tiene una clase llamada "DatabaseFarmacias", la cual contiene las
  funciones para realizar el CRUD:
```

Objeto de la clase DatabaseFarmacias:

```
late DatabaseFarmacias db;
```

Listado donde se irán guardando los registros de la colección consultada de la base:

```
List docs = [];
```

Inicialización del objeto:

```
initialise() {
```

```
  db = DatabaseFarmacias();
```

```
  db.initiliase();
```

Se hace uso de la función para leer todos los datos, se configura el estado internamente para asignar cada valor (registro de la colección) como elementos para la lista "docs", guarda todos los registros de la colección Farmacias en el listado que luego se usará para mostrarlos en pantalla:

```
db.read().then((value) => {
  setState(() {
    docs = value;
  })
});
}
```

Se inicia el estado de la conexión:

```
@override
void initState() {
  super.initState();
  initialise();
}
```

El widget principal, este contiene una barra de aplicación "AppBar" que a su vez, contendrá un botón para regresar, el título y una opción de búsqueda con un ícono este Widget contiene también el listado de las farmacias:

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color(0xFF81F4DA),
    appBar: AppBar(
      centerTitle: true,
      backgroundColor: const Color(0xAAFA94FD),
      title: const Text("Farmacias"),
    ),
)
```

AQUÍ SE MUESTRAN LOS REGISTROS DE LA COLECCIÓN FARMACIAS DIRECTO DE LA BASE DE DATOS:

Se utiliza un constructor de una vista de listado, dentro de la cual, por medio del context, se crean elementos del mismo formato uno por cada elemento de la colección de la base de datos:

```
body: ListView.builder(
  La cantidad de farmacias:
  itemCount: docs.length,
```

El elemento que se creará para cada farmacia, en este caso, un botón azul con text blanco:

```
itemBuilder: (BuildContext context, int index) {  
    return Container(  
        padding: const EdgeInsets.symmetric(vertical: 5),  
        margin: const EdgeInsets.symmetric(horizontal: 25),  
        child: ButtonTheme(  
            minWidth: double.infinity,  
            height: 45.0,  
            child: FlatButton(  
                color: Colors.blue[800],
```

Esta función permite al botón realizar una acción :

```
        onPressed: () {  
            Navigator.push(  
                context,  
                MaterialPageRoute(  
                    builder: (context) =>  
                    Sucursales(farmacia: docs[index], db: db)));  
        },  
        // ignore: avoid_unnecessary_containers  
        child: Text(  
            docs[index]['nombre'],  
            style: const TextStyle(  
                fontSize: 25,  
                color: Colors.white,  
                fontFamily: "Tahoma",  
            ),  
        )),  
    ),  
);  
},  
),  
);  
}  
}
```

8. Pantalla Sucursales

Importando Librerías:

```
// ignore_for_file: deprecated_member_use, camel_case_types, avoid_unnecessary_containers
//import 'package:firebase_core/firebase_core.dart';
//import 'package:cloud_firestore/cloud_firestore.dart';
//import 'package:firebase_auth/firebase_auth.dart';
//import 'package:medicinasyfarmacias/pantallas/sucursales.dart';
//import '../funciones/funciones.dart';
import 'package:flutter/material.dart';
import 'package:medicinasyfarmacias/database/dbfarmacias.dart';
import 'package:medicinasyfarmacias/pantallas/sucursal_seleccionada.dart';
```

El StatefulWidget que conocemos, aquí se crea toda la metodología donde se interactúa con la base de datos.

Se necesita un contexto dentro del método donde se realizar la consulta a la base de datos:

```
class Sucursales extends StatefulWidget {
  Sucursales({Key? key, required this.farmacia, required this.db})
      : super(key: key);
  Map farmacia;
  DatabaseFarmacias db;
  @override
  _SucursalesState createState() => _SucursalesState();
}
```

```
String idFarmacia = "";
```

```
class _SucursalesState extends State<Sucursales> {
```

Elementos para lograr la conexión con la base de datos, usando el archivo dart "dbFarmacias" el cual tiene una clase llamada "DatabaseFarmacias", la cual contiene las funciones para realizar el CRUD.

Objeto de la clase DatabaseFarmacias:

```
late DatabaseFarmacias db;
```

Listado donde se irán guardando los registros de la colección consultada de la base:

```
List docs = [];
```

Inicialización del objeto:

```
initialise() {
```

```
  db = DatabaseFarmacias();
```

```
  db.initiliase();
```

Se hace uso de la función para leer todos los datos, se configura el estado internamente para asignar cada valor (registro de la colección) como elementos para la lista "docs":

```
db.readSucursales(widget.farmacia['id']).then((value) => {
    setState(() {
        docs = value;
    })
});
}
}
```

Se inicia el estado de la conexión:

```
@override
void initState() {
    idFarmacia = widget.farmacia['id'];
    super.initState();
    initialise();
}
```

El widget principal, este contiene una barra de aplicación "AppBar" que a su vez, contendrá un botón para regresar, el título y una opción de búsqueda con un ícono

Este Widget contiene también el listado de las farmacias:

```
@override
Widget build(BuildContext context) {
    return Scaffold(
        Color de fondo:
        backgroundColor: const Color(0xFF81F4DA),
        Barra superior:
        appBar: AppBar(
            Centra el título que está contenido dentro del appbar:
            centerTitle: true,
            Color de fondo del appbar:
            backgroundColor: const Color(0xAAFA94FD),
            Texto "Farmacias":
            title: const Text("Sucursales"),
        ),
}
```

AQUÍ SE MUESTRAN LOS REGISTROS DE LA COLECCIÓN FARMACIAS DIRECTO DE LA BASE DE DATOS:

Se utiliza un constructor de una vista de listado, dentro de la cual, por medio del context, se crean elementos del mismo formato uno por cada elemento de la colección de la base de datos:

```
body: ListView.builder(  
    La cantidad de farmacias:  
    itemCount: docs.length,
```

El elemento que se creará para cada farmacia, en este caso, un botón azul con text blanco:

```
itemBuilder: (BuildContext context, int index) {  
    return Container(  
        padding: const EdgeInsets.symmetric(vertical: 5),  
        margin: const EdgeInsets.symmetric(horizontal: 25),  
        child: ButtonTheme(  
            minWidth: double.infinity,  
            height: 60.0,  
            child: FlatButton(  
                color: Colors.blue[600],  
  
                onPressed: () {  
                    Navigator.push(  
                        context,  
                        MaterialPageRoute(  
                            builder: (context) => SucursalSeleccioanda(  
                                sucursal: docs[index],  
                                db: db,  
                                idFarmacia: idFarmacia)));  
                },  
                child: Container(  
                    child: Column(  
                        children: [  
                            Text(  
                                docs[index]['nombre'],  
                                style: const TextStyle(  
                                    fontSize: 25,  
                                    color: Colors.white,  
                                    fontFamily: "Tahoma",  
                                )),  
                            ),  
                            Text(  
                                docs[index]['telefono'],  
                                style: const TextStyle(  
                                    color: Colors.white,  
                                    fontFamily: "Tahoma",  
                                )),  
                            ),  
                        ),  
                    ),  
                ),  
            ),
```

```

        docs[index]['direccion'],
        style: const TextStyle(
            fontSize: 16,
            color: Colors.white,
            fontFamily: "Arial",
        ),
    ),
),
],
),
),
),
),
),
),
),
),
);
},
),
);
}
}
}

```

9. Pantalla Sucursal Seleccionada

Importando Librerias:

```

import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:medicinasyfarmacias/database/dbfarmacias.dart';
import 'package:medicinasyfarmacias/pantallas/horarios_sucursal.dart';
import 'package:medicinasyfarmacias/funciones/funciones.dart';

```

El StatefulWidget que conocemos, aquí se crea toda la metodología donde se interactúa con la base de datos.

```

SucursalSeleccionada(
    {Key? key,
    required this.sucursal,
    required this.db,
    required this.idFarmacia})
    : super(key: key);

Map sucursal;
DatabaseFarmacias db;
String idFarmacia;

@Override

```

```

    _SucursalSeleccioandaState createState() => _SucursalSeleccioandaState();
}

```

Declarando Variables:

```

late Map sucursal;
late DatabaseFarmacias db;
String idFarm = "";
String nombreS = "";
String direccionS = "";
String telefonoS = "";
double latitudS = 0;
double longitudS = 0;

class _SucursalSeleccioandaState extends State<SucursalSeleccioanda> {
    // ignore: prefer_final_fields
    Set<Marker> _markers = {};

    void _onMapCreated(GoogleMapController controller) {
        setState(() {
            _markers.add(Marker(
                markerId: MarkerId(nombreS),
                position: LatLng(latitudS, longitudS),
                infoWindow: InfoWindow(
                    title: "Sucursal",
                    snippet: nombreS,
                ),
            ));
        });
    }
}

```

Se inicia la conexión para poder mostrar los datos ya registrados:

```

@Override
void initState() {
    super.initState();
    db = widget.db;
    sucursal = widget.sucursal;
    nombreS = widget.sucursal['nombre'];
    direccionS = widget.sucursal['direccion'];
    telefonoS = widget.sucursal['telefono'];
    latitudS = double.parse(widget.sucursal['latitud']);
}

```

```

longitudS = double.parse(widget.sucursal['longitud']);
idFarm = widget.idFarmacia;
}

```

Widget Principal:

```

@Override
Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
            Centra el título que está contenido dentro del appbar:
            centerTitle: true,
            color de fondo del appbar:
            backgroundColor: const Color(0xAAAA94FF),

```

Texto "Farmacias":

```

        title: const Text("Sucursal seleccionada"),
    ),

```

Contenedor detallando sus características y los elementos de la información de la sucursal:

```

    body: Container(
        width: double.infinity,
        decoration: const BoxDecoration(
            color: Color(0xAAD2FEFF),
            borderRadius: BorderRadius.only(
                topLeft: Radius.circular(30),
                topRight: Radius.circular(30),
            ),
        ),
        alignment: Alignment.topCenter,
        child: Column(
            children: <Widget>[
                const SizedBox(height: 10),
                tituloPantalla(nombreS),
                const SizedBox(
                    height: 10,
                ),
                Text("Dirección: " + direccionS,
                    style: const TextStyle(
                        color: Colors.black,
                        fontWeight: FontWeight.bold,
                        fontSize: 15)),

```

```

const SizedBox(
    height: 10,
),
Text("Teléfono: " + telefonoS,
    style: const TextStyle(
        color: Colors.black,
        fontWeight: FontWeight.bold,
        fontSize: 15)),
const SizedBox(
    height: 10,
),
const SizedBox(
    height: 300,
    child: GoogleMap(
        onMapCreated: _onMapCreated,
        markers: _markers,
        initialCameraPosition: CameraPosition(
            target: LatLng(latitudS, longitudS), zoom: 13),
    ),
),
const SizedBox(
    height: 20,
),
Container(
    margin: const EdgeInsets.symmetric(horizontal: 100),
    child: ButtonTheme(
        minWidth: double.infinity,
        height: 45.0,
        child: FlatButton(
            color: Colors.blue,
            shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(18.0),
            ),
            //padding: EdgeInsets.symmetric(horizontal: 100, vertical: 15),
            onPressed: () {
                Navigator.push(
                    context,
                    MaterialPageRoute(
                        builder: (context) => HorarioSeleccionado(
                            sucursal: sucursal,
                            db: db,
                            idFarmacia: idFarm)));
            }
        )
    )
);

```

```
},
```

Se muestran los horarios de la sucursal:

```
        child: const Text(
            "Horarios",
            style: TextStyle(
                fontSize: 25,
                color: Colors.white,
                fontFamily: "Tahoma",
            ),
        )),  
    ),  
    ),  
],  
,  
),  
resizeToAvoidBottomInset: false,  
);  
}  
}
```

10. Pantalla Información de los desarrolladores

Importando Librerías:

```
import 'package:flutter/material.dart';
import '../funciones/funciones.dart';
```

Inicializando la Clase Principal:

```
class Informacion extends StatefulWidget {
    const Informacion({Key? key}) : super(key: key);

    @override
    _InformacionState createState() => _InformacionState();
}

class _InformacionState extends State<Informacion> {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
```

```

        appBar: AppBar(
            title: const Text("Acerca de"),
        ),
        body: informacionApp(),
    );
}
}

Widget informacionApp() {
    return Container(
        decoration: const BoxDecoration(
            color: Color(0xAA00F4FF),
        ),
        child: Center(
            child: ListView(
                children: <Widget>[
                    const SizedBox(
                        height: 30,
                    ),

```

Título de la Pantalla especificando las características de este :

```

        tituloPantalla("Información"), //Titulo de la
pantalla
        const SizedBox(
            height: 15,
        ),
        imgDesarrollador(),
        const SizedBox(
            height: 15,
        ),
        acercaDe(),
        const SizedBox(
            height: 10,
        ),
    ],
),
);
}
}
```

Método para colocar la imagen de referencia a la pantalla:

```
Widget imgDesarrollador() {  
    return Column(  
        children:  
<Widget>[Image.asset('assets/img/hombresitoxD.png', width:  
200)],  
    );  
}
```

Contenedor que muestra la información de los integrantes del equipo y se especifican las características del contenedor:

```
Widget acercaDe() {  
    return Container(  
        width: double.infinity,  
        decoration: const BoxDecoration(  
            color: Color(0xAAD2FEFF),  
            borderRadius: BorderRadius.only(  
                topLeft: Radius.circular(30),  
                topRight: Radius.circular(30),  
            ),  
        ),  
        alignment: Alignment.center,  
        child: Column(  
            children: <Widget>[  
                const SizedBox(height: 15),  
                const Text(  
                    "Aplicacion desarrollada por: ",  
                    style: TextStyle(  
                        color: Colors.black,  
                        fontWeight: FontWeight.bold,  
                        fontSize: 26,  
                    ),  
            ],  
        ),  
    ),
```

Agregando la información detallada de los integrantes:

```
const SizedBox(height: 30),  
botonIntegrante("Diego Fernando Bejarano Carias"),  
const SizedBox(height: 25),  
botonIntegrante("Gabriel Omar Chávez Nieto"),  
const SizedBox(height: 25),
```

```

        botonIntegrante("Carolina Lisseth Montalvo
Villamariona"),
        const SizedBox(height: 25),
        botonIntegrante("Luis Antonio Fredy Romero Cruz"),
        const SizedBox(height: 25),
        botonIntegrante("Gabriela Alejandra Ruiz
Quinteros"),
        const SizedBox(height: 35),
    ],
),
);
}

```

Método donde se desarrolla el funcionamiento del botón del integrante seleccionado y las características del diseño general de este botón:

```

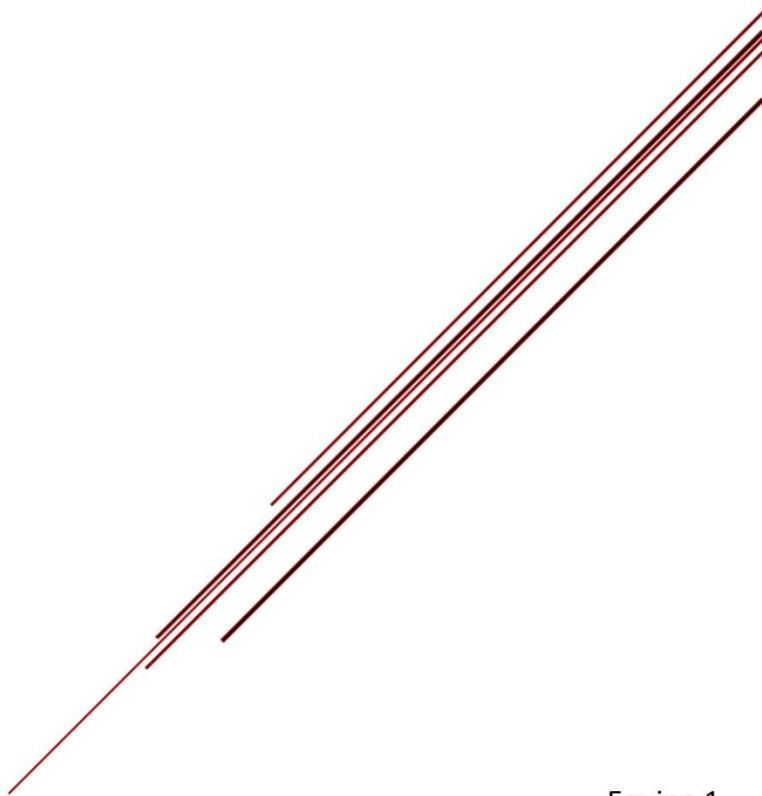
Widget botonIntegrante(String nombreBoton) {
    return Container(
        margin: const EdgeInsets.symmetric(horizontal: 25),
        child: ButtonTheme(
            minWidth: double.infinity,
            height: 45.0,
            child: FlatButton(
                color: Colors.blue[800],
                //padding: EdgeInsets.symmetric(horizontal: 100,
                vertical: 15),
                onPressed: () {},
                child: Text(
                    nombreBoton,
                    style: const TextStyle(
                        fontSize: 25,
                        color: Colors.white,
                        fontFamily: "Tahoma",
                    ),
                )));
}

```

MANUAL DE USO DE LA APLICACIÓN

MANUAL DE USO

MediPlus+ App medicamentos y farmacias



Equipo 1
Flutter y firebase

INICIANDO SESION

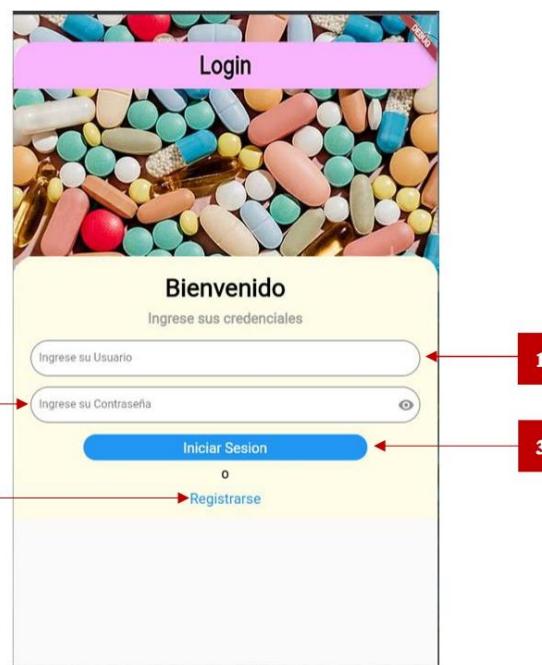
Paso 1. Ingrese a la aplicación que tiene descargada en su celular.

Paso 2. La primera pantalla que mostrará la app será la de login.

Paso 3. Ingrese sus datos en las casillas correspondientes.

Paso 4. De clic en el botón: "Iniciar sesión".

Paso 5. En caso de que aún no posea una cuenta, de clic en el botón: "Registrarse".



Pantalla de inicio, nos permite ingresar a la app y muestra un mensaje de Bienvenida. La información de cada usuario esta registrada en una base de datos creada en Firebase.

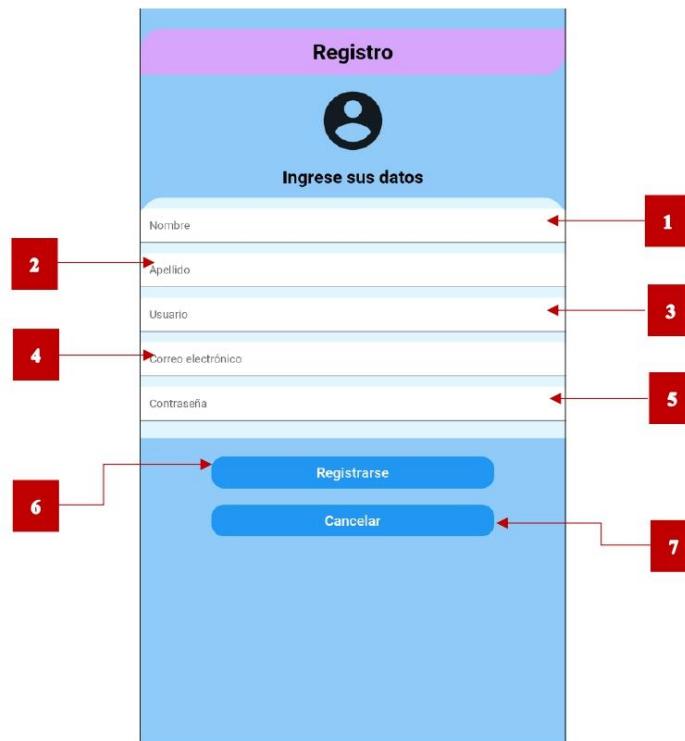
1. Casilla de texto, en donde ingresara nombre de usuario.
2. Casilla de texto, en donde ingresara contraseña.
3. Botón para iniciar sesión, redirige a la pantalla del menú de la app.
4. Botón Registrarse, que redirige a la pantalla de registro en caso de que aún no posea una cuenta.

REGISTRANDOSE POR PRIMERA VEZ

Pasos 1. Ingrese sus datos, sin dejar ningún campo vacío.

Paso 2. De clic al botón: "Registrarse"

Paso 3. En caso de que, si tiene una cuenta, de clic al botón: "Cancelar" para volver a la pantalla de login y poder iniciar sesión.



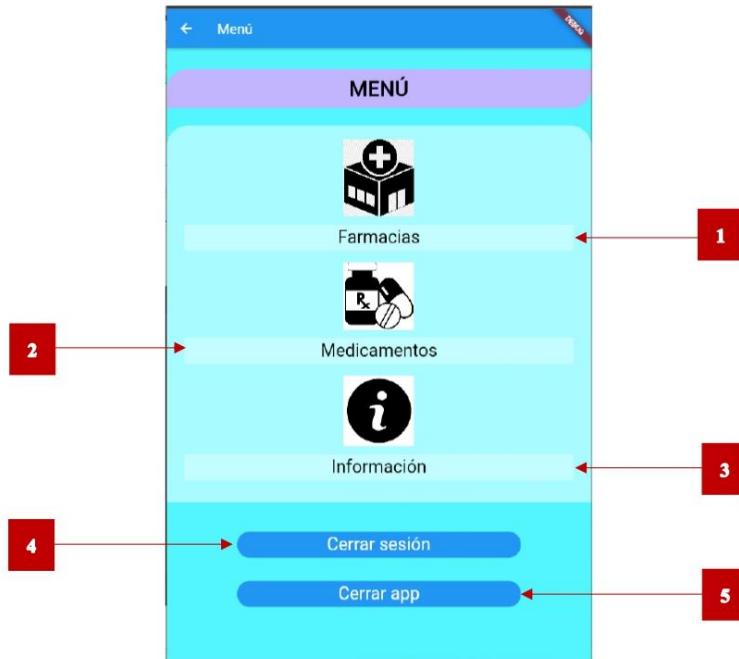
Pantalla de registro, la información que proporcione el usuario se almacenara en la base de datos de Firebase.

1 a 5. Casillas de texto para ingresar respectivamente: El nombre, apellido, nombre de usuario, correo electrónico y contraseña.

6. Botón Registrarse, para guardar su registro y tener una cuenta.

7. Botón Cancelar, por si desea regresar a la pantalla principal "Login".

MENÚ



Pantalla de menú, muestra las opciones del menú de la app.

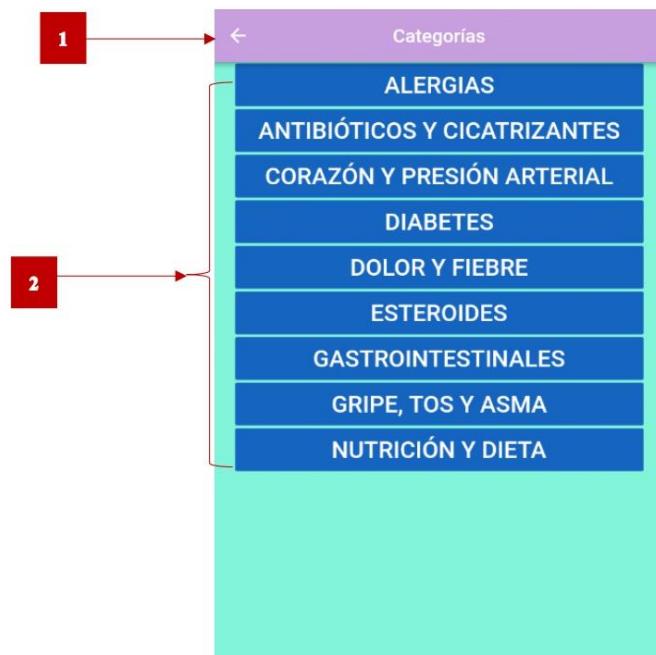
1. Farmacias registradas, si selecciona esta opción podrá visualizar las farmacias que estén afiliados a nuestra aplicación.
2. Medicamentos, si selecciona esta segunda opción podrá visualizar las categorías de los medicamentos.
3. Información, nos llevara a la pantalla que contiene los nombres de los desarrolladores de la app.
4. Botón Cerrar sesión, por si desea cerrar la sesión de usuario.
5. Botón Cerrar app, por si desea cerrar la aplicación.

MEDICAMENTOS

Paso 1. Iniciar sesión en la app.

Paso 2. Seleccionar la opción "Medicamentos" en el menú de la app.

Paso 3. Mostrará el listado de categorías de Medicamentos



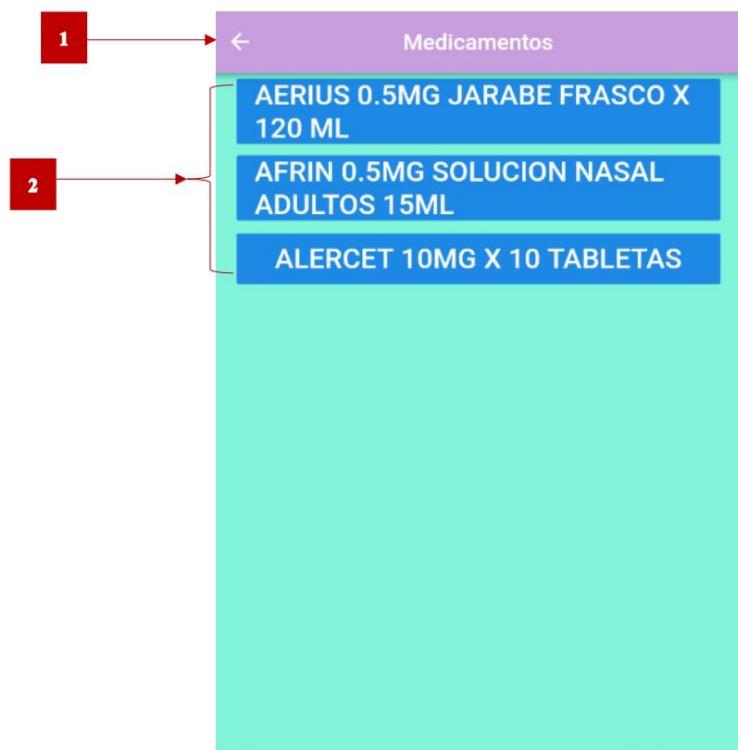
1. En la barra de Categorías podrá regresar al menú de la app.
2. Se muestra el listado de categorías de medicinas en el cual puede seleccionar el que desee ver.

CATEGORIA SELECCIONADA - MEDICAMENTOS.

Paso 1. Una vez ha seleccionado una categoría (Como se muestra en la pantalla anterior). La aplicación le mostrara la pantalla de "Categoría Seleccionada"

Paso 2. Se mostrará el listado de medicamentos de la categoría que haya seleccionado anteriormente.

Paso 3. Desde la barra podrá regresar al listado de categorías de medicamentos.



1. En la barra de Categoría Seleccionada podrá regresar al listado de categorías.
2. Se muestra el listado de medicamentos que pertenecen a la categoría seleccionada,

MEDICAMENTO SELECCIONADO.



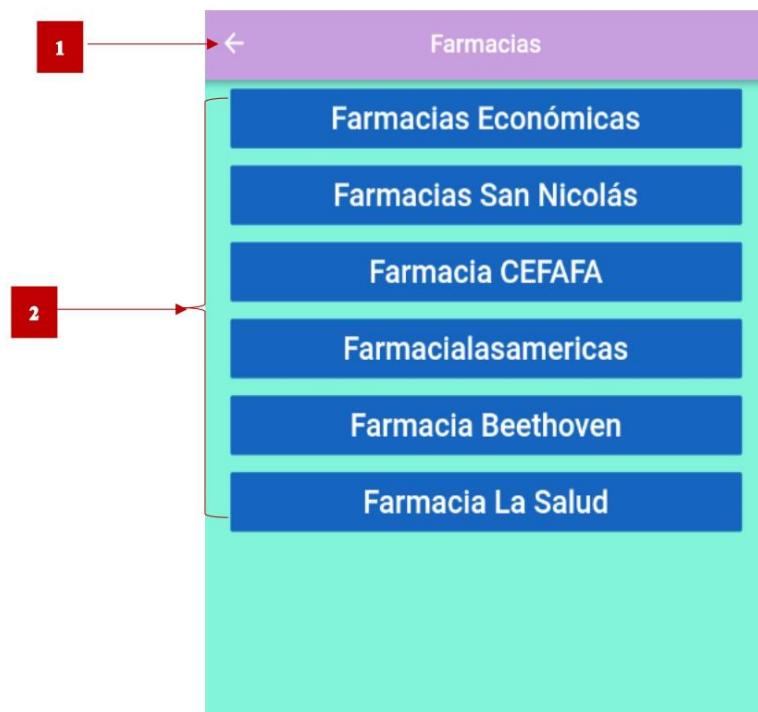
1. Desde la barra de medicamento seleccionado podrá regresar al listado de categoría seleccionada.
2. Se muestra el nombre, la imagen y la descripción del medicamento seleccionado.

FARMACIAS.

Paso 1. Iniciar sesión en la app.

Paso 2. Seleccionar la opción “Farmacias” en el menú de la app.

Paso 3. Mostrara el listado de las farmacias.



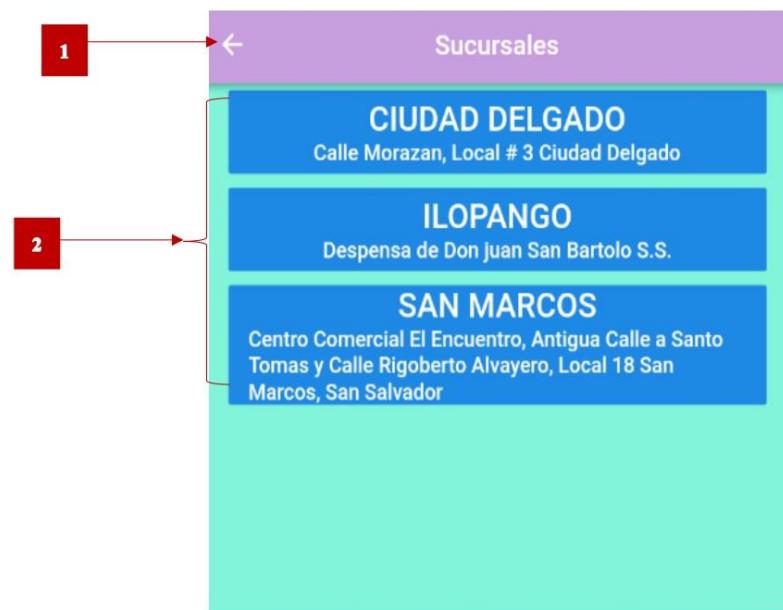
1. Desde la barra de Farmacias puede regresar al menú de la app.
2. Se muestra el listado de farmacias en el cual puede seleccionar la que desee ver.

FARMACIA SELECCIONADA - SUCURSALES

Paso 1. Una vez ha seleccionado una farmacia (Como se muestra en la pantalla anterior). La aplicación le mostrara la pantalla de "Sucursales"

Paso 2. Se mostrará el listado de sucursales de la farmacia que haya seleccionado anteriormente.

Paso 3. Desde la barra podrá regresar al listado de farmacias.



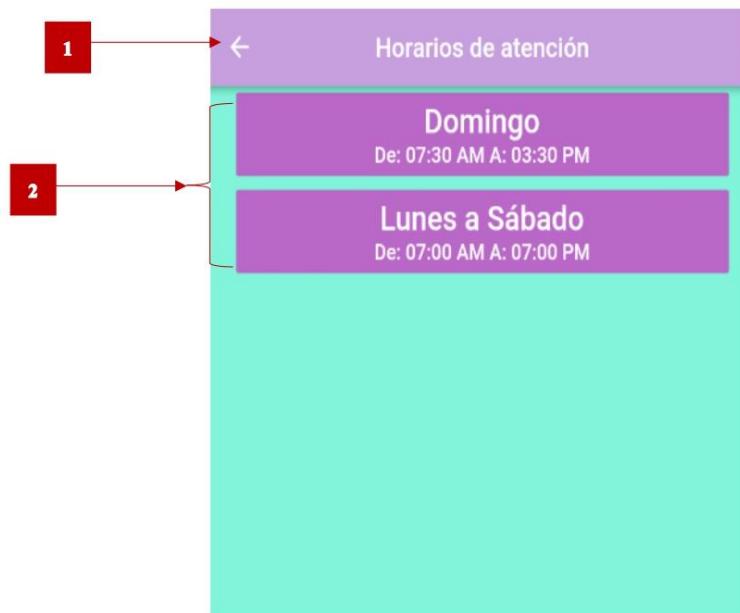
1. En la barra de Sucursales podrá regresar al listado de farmacias.
2. Se muestra el listado de sucursales que pertenecen a la farmacia seleccionada.

SUCURSAL SELECCIONADA



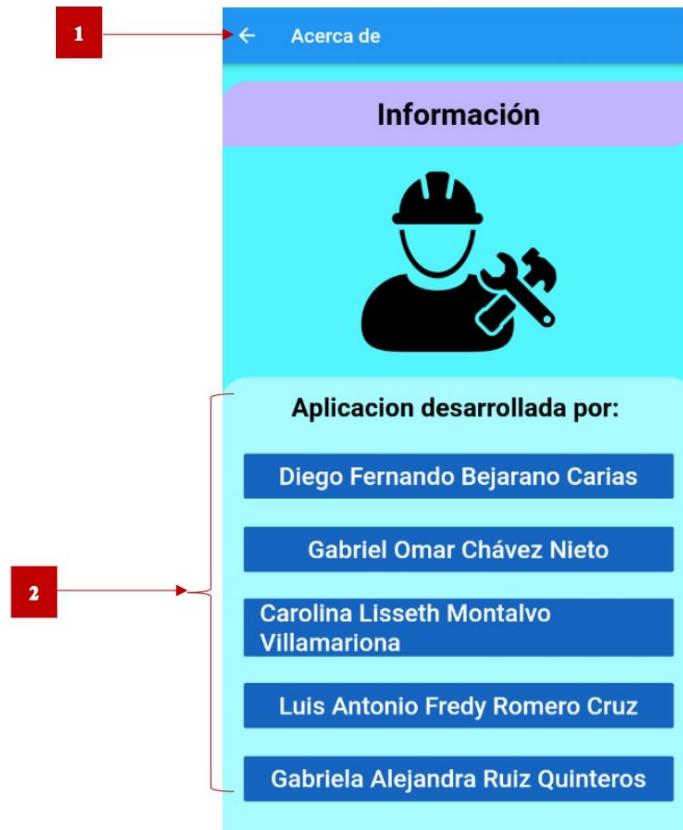
1. En la barra de Sucursal Seleccionada podrá regresar al listado de sucursales
2. Se muestra la dirección y el teléfono de la sucursal seleccionada y se encuentra la ubicación por medio de Google Maps para poder llegar.
3. Un botón para poder ver los horarios de atención de la sucursal.

HORARIOS – SUCURSAL.



1. En la barra de Horarios de atención podrá regresar a la sucursal seleccionada.
2. Se muestran los horarios de atención de Lunes a Domingo.

INFORMACION DE LOS DESARROLLADORES.



1. En la barra Acerca de podrá regresar al menú de la app.
2. Se muestra la información de los desarrolladores de la aplicación.

CAPTURAS DE PRESENTACIÓN FINAL

**Universidad Tecnológica
de El Salvador**



FACULTAD DE INFORMÁTICA Y CIENCIAS APLICADAS.

Técnicas de calidad de software

MediPlus

"Aplicación móvil para la búsqueda de información sobre medicamentos y farmacias"

Objetivo General.

·Desarrollar una aplicación donde el usuario tenga la facilidad de encontrar la información que necesite sobre medicamentos, brindando la confianza en que dicha información es verídica, así como de tener la opción de encontrar una farmacia donde sea más conveniente comprar sus medicamentos.

Objetivos específicos.

·Detallar la información de cada medicamento, sea este uno generalizado o por prescripción médica.

·Mostrar las farmacias donde el usuario pueda hacer la compra de sus medicamentos, midiendo factores como el económico y la cercanía de la farmacia.

DEFINICIÓN DEL PROYECTO

Todas las personas sabemos que la salud es importante, un factor que no se puede dejar olvidado. Por lo tanto nuestro proyecto consistirá en la creación y desarrollo de una aplicación móvil que funcione como recopilatorio de toda la información relevante referente a los medicamentos, además de las farmacias donde éstos pueden ser conseguidos.



02

JUSTIFICACIÓN DEL PROYECTO

Todas las personas día a día tienen que comprar medicamentos ya sea para uso propio o para algún familiar, pero siempre hay inconvenientes al momento de comprar y surgen muchas dudas, por ejemplo: ¿En cuál farmacia estará este medicamento?, ¿Qué precio aproximado tiene este medicamento?, ¿Cuál es la dosis recomendada de este medicamento?.

Es por eso que se lleva a cabo el desarrollo de esta aplicación, con la finalidad de cubrir las necesidades del usuario, ya que funcionará como recopilatorio de la información referente a medicamentos que, a su vez, mostrará las farmacias donde éstos pueden ser conseguidos.

Tecnologías Utilizadas



Built with
Firebase

Es una plataforma en la nube para el desarrollo de aplicaciones web y móvil.



Visual Studio Code

Es un editor de código fuente que permite trabajar con diversos lenguajes de programación



Flutter

Es un framework de código abierto desarrollado por Google para crear aplicaciones nativas de forma fácil, rápida y sencilla.

Funcionamiento de la Aplicación



CONCLUSIÓN

Mediante la programación de esta aplicación móvil, hemos determinado que solventa la necesidad de que los usuarios puedan informarse acerca de los horarios de las farmacias, las sucursales donde puede adquirir su medicamento, así como también tener un acceso más rápido y eficaz a los medicamentos que se le recetan. Con el desarrollo de esta aplicación móvil se logrará que los usuarios eviten presentarse y preguntarle a un profesional que probablemente no tenga el tiempo de ayudarle ya que lo puede consultar desde su móvil donde desee, así como también a la hora que desee.

ENLACE A CÓDIGO FUENTE DEL PROYECTO UBICADO EN REPOSITORIO

El código fuente de la aplicación móvil ha sido almacenado y actualizado constantemente en el siguiente repositorio de Github:

<https://github.com/Fernex48/Electiva3ProyectosGrupales>

EVALUACIONES PARA CUARTO AVANCE

Porcentaje de avance del proyecto

En este cuarto y último avance, se ha finalizado con las últimas versiones de la descripción de la base de datos utilizada, así como también del manual técnico y el manual de uso. Además de ello, contamos con las conclusiones oficiales del proyecto, como es también el caso de la presentación final para el día de la defensa.

Al final, nosotros como grupo hemos terminado el 100% del proyecto de desarrollo.

Autoevaluación y otras evaluaciones por parte del grupo

Se ha colocado una tabla con las evaluaciones correspondientes, colocadas en escala del 0 al 10, siendo 10 la calificación más alta. Cabe destacar lo siguiente, cada una de las calificaciones han sido puestas con honestidad, por lo que cada miembro del grupo ha revisado y está de acuerdo con lo que cada uno ha aportado como calificación. Por ejemplo, en las notas para el líder, cada miembro del equipo brindó su propia nota, y esta no fue manipulada en ningún sentido por el líder. Así como las notas del líder hacia el grupo no fueron modificadas por ningún otro miembro, y cada nota de la columna de Autoevaluación fue colocada individualmente por cada miembro.

Evaluaciones para el primer avance (En escala del 1 al 10)			
Integrantes del grupo	Auto evaluación	Evaluación del líder hacia el grupo	Evaluación de los integrantes hacia el líder
Diego Bejarano (Líder)	10	10	10
Carolina Montalvo	10	10	10
Gabriel Chávez	10	10	10
Gabriela Ruiz	10	10	10
Luis Romero	10	10	10

Autoevaluación	Cada uno se colocará una nota del 1 al 10 según sienta que ha sido su desempeño.
Evaluación del líder hacia el grupo	El líder del grupo va a ponerles una nota a cada uno personalmente.
Evaluación de los integrantes hacia el líder	Todos deberán colocarle una nota que crean que el líder merezca según su desempeño.

Actividades realizadas por cada miembro del grupo

Diego Bejarano (Líder):

- Programación oficial de la aplicación móvil.
- Recopilación del contenido para la cuarta entrega.

Carolina Montalvo:

- Realización de pruebas generales de la aplicación móvil.
- Seguimiento general del documento.

Gabriel Chávez:

- Redacción de la segunda versión del manual de uso.
- Seguimiento general del documento.

Gabriela Ruiz:

- Creación de la presentación final.
- Primer avance de conclusiones.

Luis Romero:

- Redacción de la segunda versión del manual técnico.
- Seguimiento general del documento.