

# Modulo Farmacia

Privilage Care

**Carlos Martin Hernández de Jesús**  
**Cristian Eduardo Ojeda Gayosso**  
**Myriam Valderrábano Cortes**  
**Daniela Aguilar Torres**



# Tabla de Contenido

✦ 01 ✦

Diseño Figma

✦ 02 ✦

Frontend

✦ 03 ✦

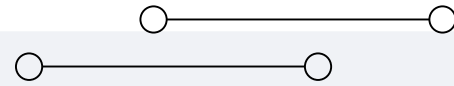
Backend

✦ 04 ✦

Trabajo Colaborativo

✦ 05 ✦

Conclusión





✦ 01 ✦

# Diseño Figma





# Figma

Figma es una de las herramientas más populares para diseño de interfaces (UI) y experiencia de usuario (UX).

- Colaboración en Tiempo Real
- Compatibilidad Multiplataforma
- Componentes Reutilizables
- Plan Gratuito

Es perfecto para conceptualizar, diseñar y crear ideas de interfaces interactivas.

<https://www.figma.com/design/OBYpnqY6tCk7NGEkiwAfN6/Farmacia?node-id=0-1&t=u2AOHZIf6Mhqdv7E-1>

Bootstrap Icons

Design System Organizer

Elegant Circles Icon Set by Iconduck

Iconify

Image Palette

Magical Graphics





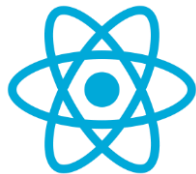
✦ 02 ✦

# Frontend





# Tecnologías Usadas



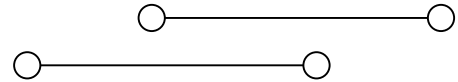
## React Native

Es una librería desarrollada por Facebook para crear aplicaciones móviles nativas y multiplataforma a partir de una misma base de código escrita en JavaScript y React.



## Expo

Es un entorno completo y extenso que permite crear proyectos cómodamente mediante la línea de comandos (Expo CLI), utilizar herramientas para ejecutar aplicaciones en su propio gestor (Expo Go).



# Estructura del Proyecto

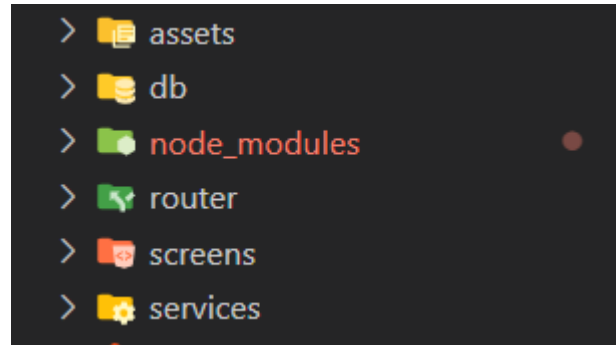
## Carpetas Base

Estas carpetas son base de la estructura de la aplicación, por lo tanto no deben ser alteradas.



## Carpetas App

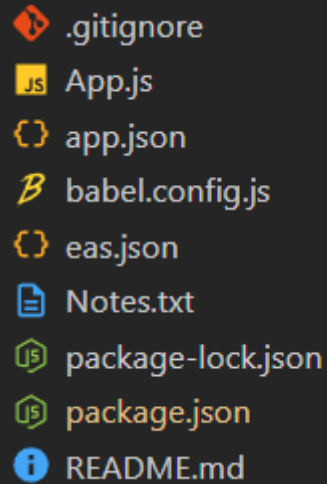
Estas carpetas son la construcción de la aplicación, contiene tanto paquetes, componentes, paginas y el ruteo de las paginas.












# Estructura del Proyecto

## Archivo de Configuración

Estos son archivos de configuración base del proyecto, algunos son para manejo de paquetes o configuración principal.



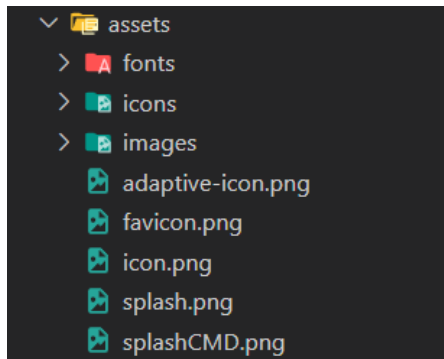
-  .gitignore
-  App.js
-  app.json
-  babel.config.js
-  eas.json
-  Notes.txt
-  package-lock.json
-  package.json
-  README.md



# Desarrollo del Proyecto

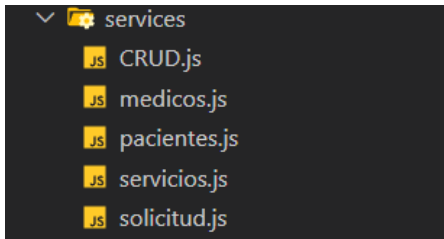
## Assets

Dentro de esta carpeta podremos encontrar elementos como imágenes, icono y fuentes para poder implementar en el proyecto.



## Services

Dentro de esta carpeta podremos encontrar los endpoints para la conexión al API.



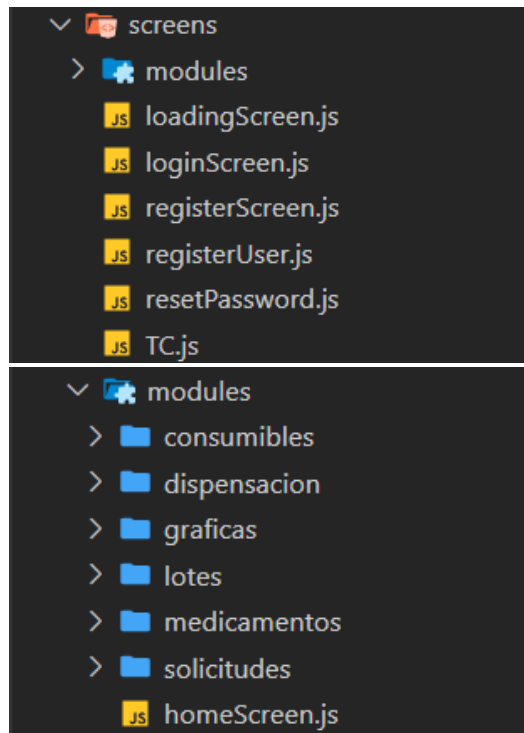
# Desarrollo del Proyecto

## Screens

Esta carpeta contiene las pantallas principales que se muestran al momento de iniciar el proyecto, como el login, Register, etc.

## Modules

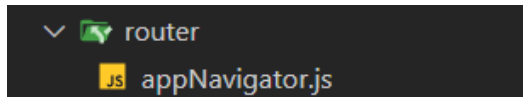
Esta carpeta contiene los componentes principales del modulo de farmacia, es decir, las pantallas que conforman esta.



# Desarrollo del Proyecto

## Routes

Esta carpeta contiene únicamente un archivo, que es el que se encarga de la navegación entre pantallas



## App.js

Es un archivo que únicamente contiene la carga de la pantalla principal.

```
import React from 'react';
import AppNavigator from './router/appNavigator';

const App = () => {
  return <AppNavigator />;
};

export default App;
```



✦ 03 ✦

# Backend





# Tecnologías Usadas



## Python

Es un lenguaje de programación, orientado a objetos, con una semántica dinámica integrada, principalmente para el desarrollo web y de aplicaciones informáticas.



## FastAPI

Es un framework web de alto rendimiento para construir APIs con Python. Ayuda a los desarrolladores a crear aplicaciones de forma rápida y eficaz.



## Uvicorn

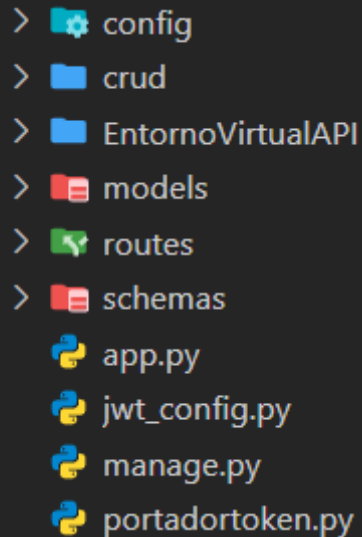
Es un servidor web ASGI (significa Interfaz de puerta de enlace de servidor asíncrono) que se utiliza para ejecutar aplicaciones FastAPI.



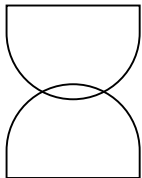
# Estructura del Proyecto

Contiene toda la estructura para la construcción del API, como por ejemplo:

- **Config:** contiene el archivo de configuración y conexión a la base de datos.
- **Crud:** contiene los archivos de cada tabla, para realizar las operación de crear, leer, actualizar y eliminar.
- **ENV:** es la carpeta que contiene la creación del entorno virtual de Python.
- **Models:** Contiene la estructura de las construcciones de las tablas.
- **Routes:** contiene la construcción de los endpoints. para realizar las solicitudes.
- **Schemas:** Contiene las estructura y configuraciones de los modelos.



```
> config
> crud
> EntornoVirtualAPI
> models
> routes
> schemas
app.py
jwt_config.py
manage.py
portadortoken.py
```





✦ 04 ✦



# Trabajo Colaborativo



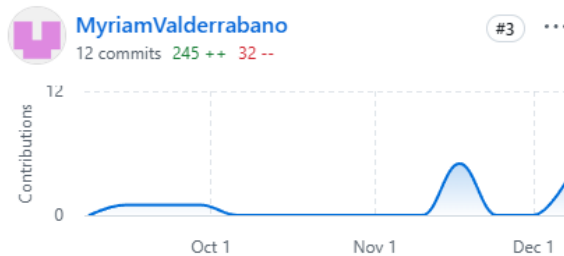


# Commits

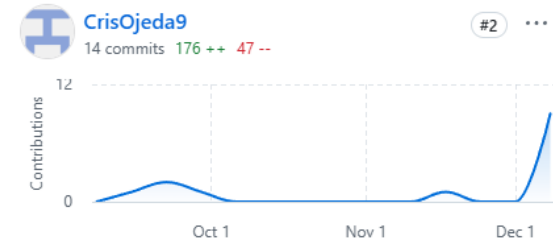
## Carlos Martin



## Myriam



## Cristian Eduardo



## Daniela







# Grafica

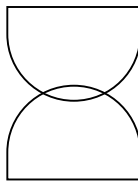




05

# Conclusión

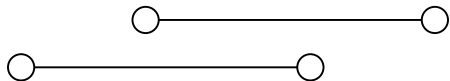




# Conclusión

El desarrollo de este proyecto requiere un enfoque integral para implementar una aplicación funcional que gestiona todos los módulos. A lo largo del proceso, se logró estructurar y diseñar componentes organizados en React Native, estableciendo una arquitectura que facilita la escalabilidad y el mantenimiento futuro del código.

No obstante, uno de los desafíos más complejos fue la conexión con la API. A pesar de los esfuerzos, no fue factible establecer la conexión entre la aplicación móvil y el backend debido a dificultades que incluyeron configuraciones, rutas o dependencias. Aunque la API no pudo ser integrada, el proyecto cumplió con los objetivos de diseño y funcionalidad en el entorno local, dejando una base sólida para que la conexión con el backend pueda abordarse en el futuro. El aprendizaje adquirido durante este proceso, especialmente en la gestión de datos y el uso de herramientas como React Native, será invaluable para proyectos futuros.





# Evidencia Conexión

# Intento de Conexión Terceros



```
1 import * as Google from 'expo-auth-session/providers/google';
2 import * as AuthSession from 'expo-auth-session';
3 import { LoginManager, AccessToken } from 'react-native-fbsdk-next';
4 import * as WebBrowser from 'expo-web-browser';
5
6 WebBrowser.maybeCompleteAuthSession();
7
8 const GOOGLE_CLIENT_ID = '652709953322-fi0qj0i0vqt5ruh42lm3pq44kmuulbrg.apps.googleusercontent.com';
9 const DISCORD_CLIENT_ID = '1316967528915927092';
10 const DISCORD_REDIRECT_URI = AuthSession.makeRedirectUri();
```



```
1 ERROR ReferenceError: Property 'googleRequest' doesn't exist
2
3 This error is located at:
4   in LoginScreen (created by SceneView)
5   in StaticContainer
6   in EnsureSingleNavigator (created by SceneView)
7   in SceneView (created by StackNavigator)
```

# Intento de Conexión Terceros



```
1 const [googleRequest, googleResponse, googlePromptAsync] = Google.useAuthRequest({
2   clientId: GOOGLE_CLIENT_ID,
3 });
4
5 useEffect(() => {
6   if (googleResponse?.type === 'success') {
7     const { authentication } = googleResponse;
8     Alert.alert('Inicio de sesión con Google', `Token: ${authentication?.accessToken}`);
9   }
10 }, [googleResponse]);
```

```
1 const handleDiscordLogin = async () => {
2   const authUrl = `https://discord.com/api/oauth2/authorize?client_id=${DISCORD_CLIENT_ID}&redirect_uri=${encodeURIComponent(
3     DISCORD_REDIRECT_URI
4   )}&response_type=token&scope=identify email`;
5
6   const result = await AuthSession.startAsync({ authUrl });
7   if (result.type === 'success') {
8     Alert.alert('Inicio de sesión con Discord', `Token: ${result.params.access_token}`);
9   } else {
10    Alert.alert('Inicio de sesión cancelado');
11  }
12 };
```

```
1 const handleFacebookLogin = async () => {
2   try {
3     const result = await LoginManager.logInWithPermissions(['public_profile', 'email']);
4     if (result.isCancelled) {
5       Alert.alert('Inicio de sesión cancelado');
6       return;
7     }
8     const data = await AccessToken.getCurrentAccessToken();
9     Alert.alert('Inicio de sesión con Facebook', `Token: ${data?.accessToken}`);
10  } catch (error) {
11    Alert.alert('Error', `Error al iniciar sesión con Facebook: ${error.message}`);
12  }
13 };
```

# Intento de Conexión Terceros



```
1  <View style={styles.container}>
2  { /* Google Login Button */}
3  <TouchableOpacity
4    style={[styles.button, { backgroundColor: '#DB4437' }]}
5    disabled={!googleRequest}
6    onPress={() => googlePromptAsync()}
7  >
8  </TouchableOpacity>
9
10 { /* Facebook Login Button */}
11 <TouchableOpacity
12   style={[styles.button, { backgroundColor: '#3b5998' }]}
13   onPress={handleFacebookLogin}
14 >
15 </TouchableOpacity>
16
17 { /* Discord Login Button */}
18 <TouchableOpacity
19   style={[styles.button, { backgroundColor: '#7289da' }]}
20   onPress={handleDiscordLogin}
21 >
22 </TouchableOpacity>
23 </View>
```

# Intento de Conexión API



```
1 import axios from 'axios';
2
3 async function obtenerPersonas() {
4   try {
5     const response = await axios.get("http://127.0.0.1:8000/personal_medico/", {
6       headers: {
7         'Content-Type': 'application/json',
8       }
9     });
10
11     if (response.status !== 200) {
12       throw new Error("Error al obtener las personas");
13     }
14
15     return response.data;
16   } catch (error) {
17     console.error("Error al enviar la petición de personas al servidor: " + error);
18     throw error;
19   }
20 }
21
22 export { obtenerPersonas };
23
```

```
1 import axios from 'axios';
2
3 async function obtenerServiciosMedicos() {
4   try {
5     const response = await axios.get("http://127.0.0.1:8000/servicios_medicos/", {
6       headers: {
7         'Content-Type': 'application/json',
8       }
9     });
10
11     if (response.status !== 200) {
12       throw new Error("Error al obtener los servicios médicos");
13     }
14
15     return response.data;
16   } catch (error) {
17     console.error("Error al enviar la petición de servicios médicos al servidor: " + error);
18     throw error;
19   }
20 }
21
22 export { obtenerServiciosMedicos };

```



# Intento de Conexión API



```
1 import axios from 'axios';
2
3 async function registrarSolicitud(data) {
4   try {
5     const response = await axios.post('http://127.0.0.1:8000/solicitudes/', data, {
6       headers: {
7         'Content-Type': 'application/json',
8       }
9     });
10    if (response.status !== 200) {
11      throw new Error('Error al registrar la solicitud: ${response.data.detail}');
12    }
13    return response.data;
14  } catch (error) {
15    console.error(error.message);
16    throw new Error('Error al registrar la solicitud');
17  }
18 }
```

```
1 async function obtenerSolicitudes() {
2   try {
3     const response = await axios.get('http://127.0.0.1:8000/solicitudes/', {
4       headers: {
5         'Content-Type': 'application/json',
6       }
7     });
8     if (response.status !== 200) {
9       throw new Error('Error al obtener las solicitudes: ${response.data.detail}');
10    }
11    return response.data;
12  } catch (error) {
13    console.error(error.message);
14    throw new Error('Error al obtener las solicitudes');
15  }
16 }
```



# Gracias por su Atención

[https://github.com/Daniela-AG15/Modulo\\_Farmacia.git](https://github.com/Daniela-AG15/Modulo_Farmacia.git)

