MANUAL DE TÉCNICO

Smart Story

Manual técnico orientado al desarrollo y funcionamiento adecuado de la aplicación desarrollada

Anthony Pulla

Tabla de contenido

1. Introducción

Smart story es una aplicación diseñada con el objetivo de estimular la imaginación y el interés por la lectura en los niños. Genera cuentos e historias personalizadas que se adaptan a los gustos individuales de cada niño, convirtiendo la lectura en una experiencia única y atractiva. Además de ser una fuente de entretenimiento, es un recurso educativo que ayuda a los niños a mejorar sus habilidades de lectura y comprensión.

1.1. *Objetivos*

Se ha creado este documento con el propósito de mostrar como fue el diseño del sistema, y al mismo tiempo dar referencias de como interactuar con el programa, con el fin de facilitar futuras actualizaciones.

1.1.1 Objetivos Específicos

- **Documentar el diseño del sistema:** Describir de manera exhaustiva la arquitectura, componentes, módulos y cualquier aspecto relevante del diseño del sistema.
- **Facilitar la comprensión:** Brindar información detallada que permita a los desarrolladores, usuarios y otros interesados comprender fácilmente la estructura y funcionamiento del sistema.
- **Proporcionar pautas de interacción:** Ofrecer instrucciones claras y detalladas sobre cómo interactuar con el programa, con el objetivo de facilitar su uso y comprensión.
- **Apoyar futuras actualizaciones:** Establecer una base sólida para futuras mejoras o modificaciones en el sistema al proporcionar información detallada sobre su diseño y funcionalidad actual.

1.2. *Alcance*

Este informe abordará de manera integral el rendimiento del proyecto Smart story desde su inicio hasta la fecha actual. Se centrará en la evaluación de los resultados obtenidos en términos de cumplimiento de objetivos, eficiencia operativa, calidad del trabajo realizado y cualquier desviación con respecto al plan inicial.

1.3. *Propósito*

Smart story tiene como propósito proporcionar una plataforma interactiva que inspire a los niños a amar la lectura y a explorar su creatividad. La aplicación está diseñada para mejorar las habilidades de lectura y comprensión de los niños al ofrecerles cuentos e historias que reflejan sus intereses personales.

2. Manual de ejecución

2.1. *Requerimientos Técnicos*

Para el desarrollo y mantenimiento de nuestra aplicación, se requiere una serie de competencias técnicas específicas. En primer lugar, utilizamos Flutter, un Kit de Desarrollo de Software (SDK), para la creación de la aplicación. Este SDK es altamente eficiente y flexible, permitiendo una experiencia de usuario óptima.

El lenguaje de programación que empleamos es Dart, conocido por su eficiencia y facilidad de uso. Además, implementamos GetX para la gestión del estado de la aplicación. Este marco de trabajo proporciona una estructura sólida y requiere un entendimiento profundo de su funcionamiento.

Además, nuestra aplicación se beneficia del uso de Firebase para el almacenamiento de datos. Esto implica que es necesario tener acceso a Firebase y comprender cómo interactuar con su API.

Finalmente, estamos utilizando LLM para la generación de cuentos. Esta tecnología avanzada nos permite crear narrativas atractivas y personalizadas, mejorando así la experiencia de nuestros usuarios.

2.1.1. Hardware

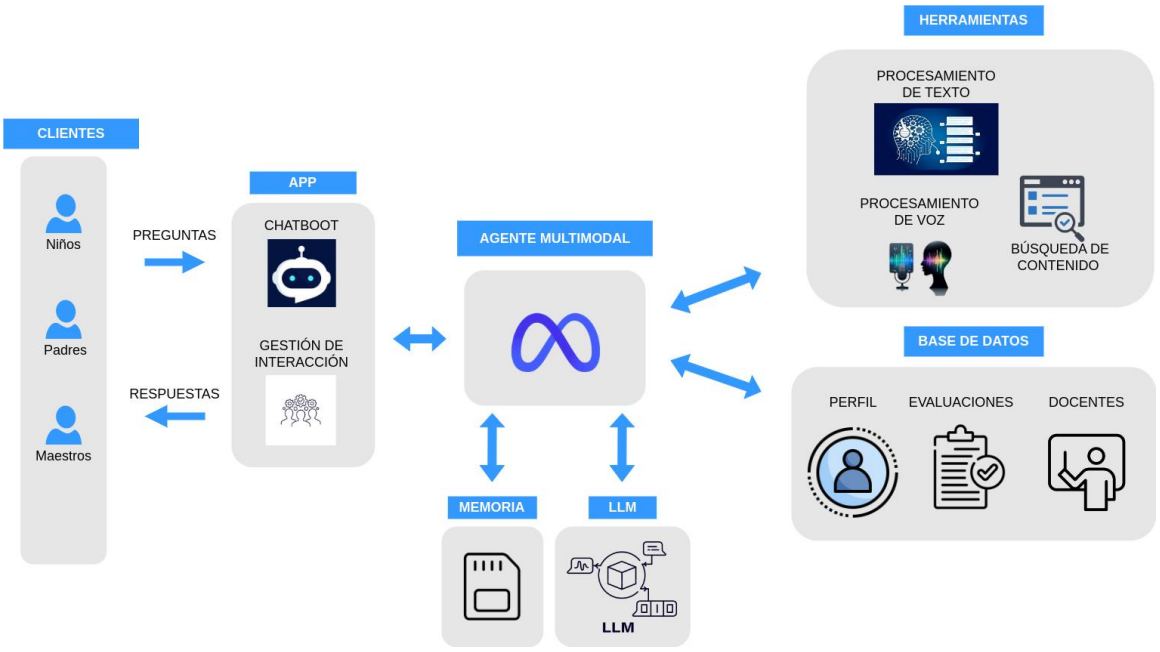
- Sistema Operativo: Especificar el sistema operativo requerido para ejecutar la aplicación, como Windows, Linux, Android.
- Memoria RAM: Al menos 2 GB de RAM. Sin embargo, se recomienda tener 4 GB o más para un rendimiento óptimo, especialmente si la generación de cuentos implica operaciones intensivas de procesamiento de texto.
- Almacenamiento: Espacio de almacenamiento suficiente para la aplicación y cualquier contenido adicional, como preguntas del cuestionario y posiblemente historias generadas. Una cantidad mínima de 16 GB de almacenamiento debería ser suficiente.
- Conectividad: Se recomienda tener conectividad a Internet para descargar recursos adicionales, como preguntas del cuestionario o actualizaciones de la aplicación.

2.1.2. Software

- *Flutter*: Necesitarás tener instalado el SDK de Flutter en tu máquina de desarrollo para trabajar con el código de la aplicación.
- *Dart*: Como el lenguaje de programación de la aplicación, necesitarás un entorno de desarrollo que soporte Dart.
- *GetX*: Como la aplicación utiliza GetX para la gestión del estado, necesitarás tener conocimientos sobre cómo funciona y cómo se implementa en Flutter.
- *Firebase*: Como la aplicación utiliza Firebase para la autenticación de usuarios y el almacenamiento de datos, necesitarás acceso a Firebase y saber cómo interactuar con su API.
- *Visual Studio Code:* Este es el editor de código recomendado para trabajar con Flutter y Dart. Proporciona una variedad de extensiones que pueden facilitar el desarrollo con estos lenguajes.

2.2. *Diseño / Construcción*

En esta sección presentamos una visión general de cómo se estructura nuestra aplicación. Como se puede ver en el diagrama a continuación, se nos muestra como esta distribuida la aplicación.



El diagrama se divide en tres secciones principales: Clientes, Servicios y Servicios de Base de Datos. Los clientes interactúan con nuestra aplicación y se comunican con los servicios a través de Firebase. Los servicios incluyen operaciones CRUD de usuario y datos, que son manejadas por Firebase.

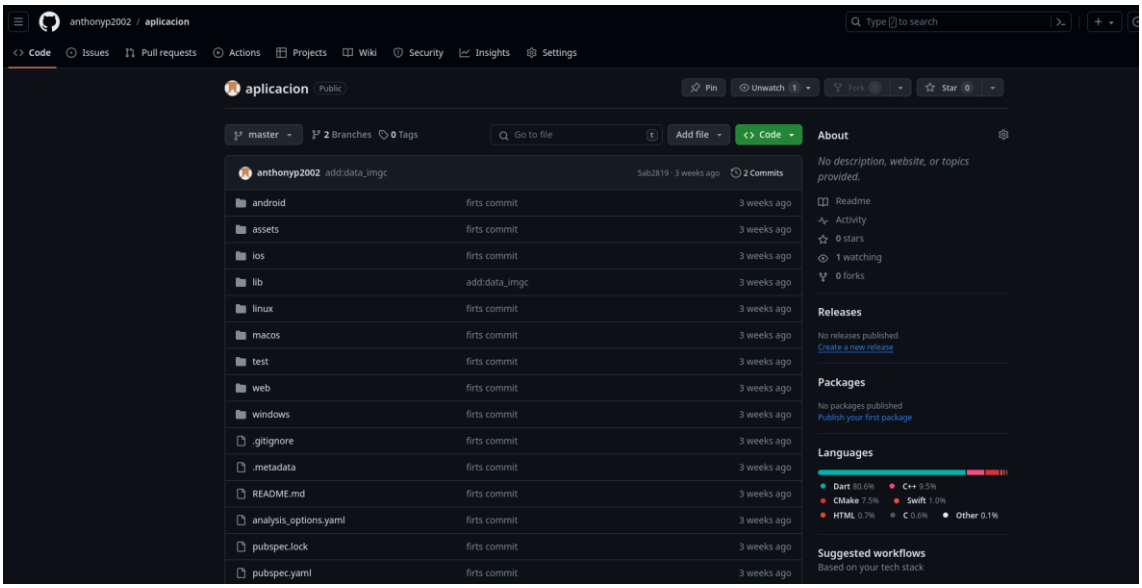
Los servicios de base de datos son donde almacenamos y recuperamos nuestros datos. El diagrama también muestra una estructura detallada de los servicios de base de datos, que incluyen secciones para 'Maestro' y 'Alumno', cada una con sus propios atributos.

Es importante destacar que este diagrama es para fines didácticos y proporciona una visión simplificada de nuestra estructura de aplicación.

2.2.1. Configuraciones e instalaciones del código fuente.

Para una mejor visualización nuestro código se encuentra publicado en un repositorio de Github para que puedan descargar el código y correrlo en un editor de textos para realizar los cambios respectivos.

Repositorio de Github: https://github.com/anthonyp2002/tales-app



2.2.2. Instalación

Para la ejecución del código tenemos que tener en cuenta que hay que instalar el los siguientes programas a continuación:

2.2.2.1 Descarga e instalación de Flutter

Para desarrollar y ejecutar el programa, es necesario tener instalado Flutter. Puedes descargarlo desde la página oficial de Flutter. Sigue las instrucciones proporcionadas para tu sistema operativo.

2.2.2.2 Configuración del entorno de desarrollo

Una vez instalado Flutter, es necesario configurar el entorno de desarrollo. Esto puede incluir la instalación de un editor de código como VS Code o Android Studio, y la configuración de un emulador o dispositivo para probar la aplicación.

2.2.2.3 Descarga e instalación de Dart

Dart es el lenguaje de programación utilizado en Flutter. Puedes descargarlo e instalarlo siguiendo las instrucciones en la página oficial de Dart.

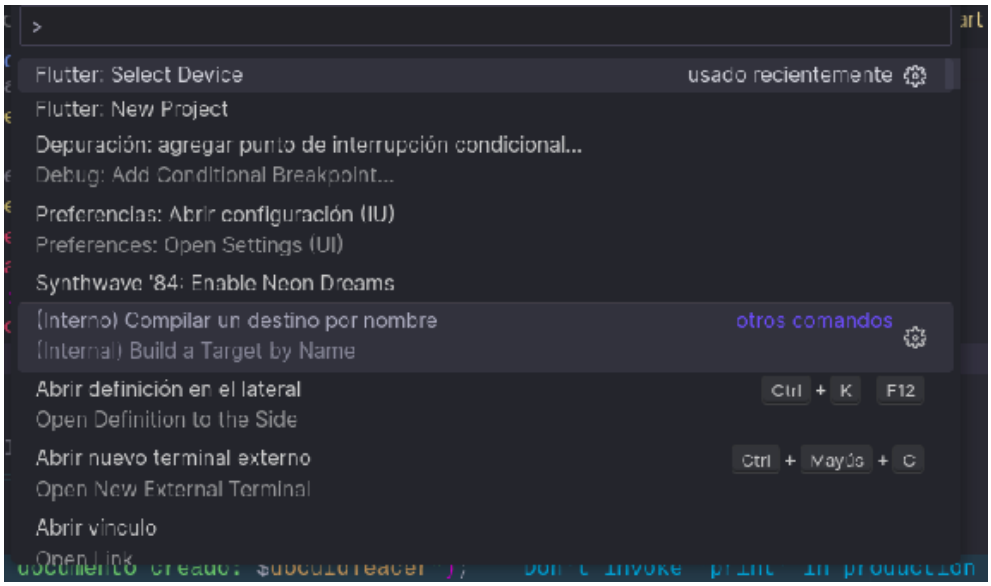
2.2.2.5 Descarga e instalación de GetX y Firebase

Además, necesitarás instalar GetX para la gestión del estado y Firebase para la autenticación de usuarios y el almacenamiento de datos. Puedes hacerlo añadiendo las dependencias correspondientes en el archivo **pubspec.yaml** de tu proyecto Flutter y luego ejecutando el comando **flutter pub get.**

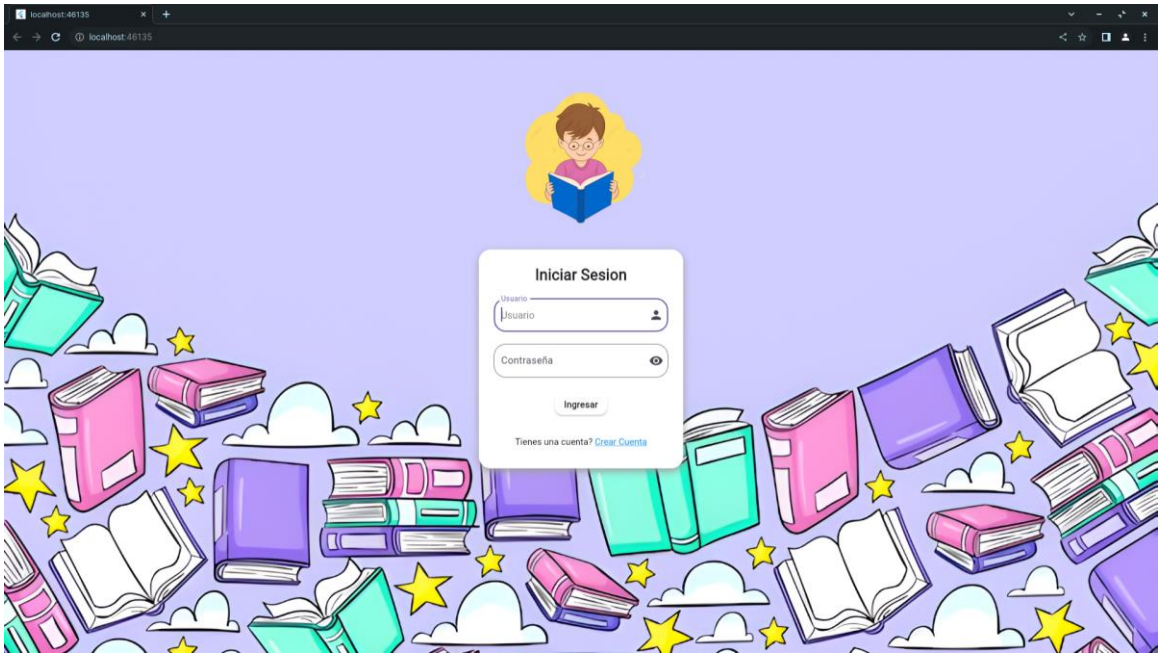
En resumen, la instalación requiere la descarga e implementación de varias herramientas, incluyendo Flutter, Dart, GetX y Firebase. Estos pasos aseguran que el entorno de desarrollo esté correctamente configurado para el desarrollo y mantenimiento de la aplicación. La guía proporciona instrucciones claras y precisas para facilitar este proceso.

2.2.3. Ejecución del Código

Para ejecutar el código en Visual Studio Code, primero necesitaremos seleccionar tu dispositivo Android, necesitarás abrir la paleta de comandos en Visual Studio Code. Puedes hacer esto presionando Ctrl+Shift+P en Windows o Command+Shift+P en Mac. Luego, escribe “Flutter: Select Device” y selecciona tu dispositivo Android de la lista.



Después abrimos una nueva terminal y escribe **flutter run**. Esto iniciará la aplicación en el entorno seleccionado como en la imagen seleccionada.



2.2.4. Configuración y despliegue con Docker

1) **Dockerfile:** El archivo Dockerfile se encarga de definir la configuración del entorno de construcción y ejecución de la aplicación Flutter. A continuación, se describen las principales secciones de este archivo.

```
Dockerfile
RUN mkdir /app/
COPY /app/ /app/
WORKDIR /app/
### This command provides detailed information about the Flutter installation, environment, and any missing dependencies.
### And it helps ensure that the necessary tools and configurations are in place for building Flutter applications.
#
RUN flutter doctor -v
### This command enables web support in the Flutter SDK.
### It configures the Flutter environment to include web-specific libraries and dependencies required for building Flutter web app
#
RUN flutter config --enable-web
### This command builds the Flutter web application in release mode.
### It generates the necessary files and assets for running the application on the web.
### The --web-renderer=auto flag allows Flutter to automatically choose the appropriate web renderer (HTML or Canvas) based on the
#
RUN flutter build web --release --no-tree-shake-icons --web-renderer=html

FROM nginx:latest
# copy the info of the built web app to nginx
COPY --from=build-env /app/build/web /usr/share/nginx/html
# Expose and run nginx
EXPOSE 80
```

2) **Docker Compose:** El archivo docker-compose.yml orquesta la configuración de los servicios y la interconexión de los contenedores. Aquí está la sección relevante.

```
docker-compose.yml
services:
  flutter-app:
    build:
      context: flutter
      dockerfile: Dockerfile
    image: flutter-app:latest
    container_name: flutter-app
    ports:
      - "80:80"
    restart: always
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
```

3) **Nginx:** Este nginx.conf define la configuración del servidor Nginx que sirve la aplicación web Flutter. Aquí está la configuración básica.

```
nginx.conf
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 1024;
}

http {
        server {
            listen 80;
            listen [::]:80;
            root /usr/share/nginx/html;
            server_name bot.ups.edu.ec;

            location / {
                alias /usr/share/nginx/html/;
            }
```

4) **GitHub Actions Workflow:** La configuración de GitHub Actions está dividida en varias etapas: check-changes, build, y deploy. Estas etapas automatizan la verificación de cambios, la construcción de la aplicación y el despliegue. Asegúrate de personalizar las variables de entorno según tus necesidades.

main-deploy.yml

```yaml
name: Tales App Deployment

on:
  push:
    branches:
      - main
  workflow_dispatch:
jobs:
  check-changes:
    name: Check Changes
    runs-on: ubuntu-latest
    outputs:
      flutter: ${{ steps.changes.outputs.flutter }}
    steps:
      - uses: actions/checkout@v4
      - uses: dorny/paths-filter@v3
        id: changes
        with:
          filters: |
            flutter:
              - 'flutter/**'
  build:
    name: Build
    runs-on: self-hosted
    needs: [check-changes]
    if: always() &&
      needs.check-changes.result == 'success'
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Build Docker Flutter App Website
        if: needs.check-changes.outputs.flutter == 'true'
        run: docker compose build --no-cache flutter-app
      - name: Clean
        run: docker image prune -f
```

main-deploy.yml

```yaml
jobs:
  build:
    steps:
      - name: Build Docker Flutter App Website
      - name: Clean
        run: docker image prune -f
  deploy:
    name: Deploy
    runs-on: self-hosted
    needs: build
    if: always() && needs.build.result == 'success'
    env:
      URL_WEBSITE: ${{ vars.URL_WEBSITE }}
    environment:
      name: Website
      url: https://${{ env.URL_WEBSITE }}
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Deploy
        run: docker compose up -d
      - name: Restart Docker Compose Services
        run: docker compose restart flutter-app
```

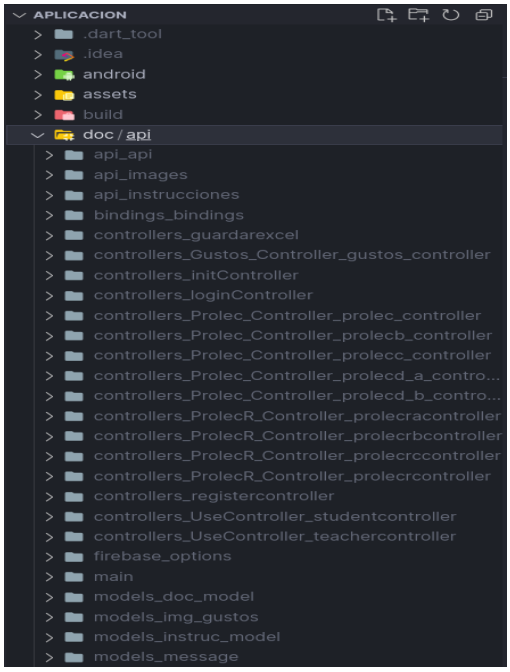
Con todo esto podemos ejecutar en nuestra consola los siguientes comandos para poder subir a nuestro GitHub y nos genere una url para poder acceder al la app.

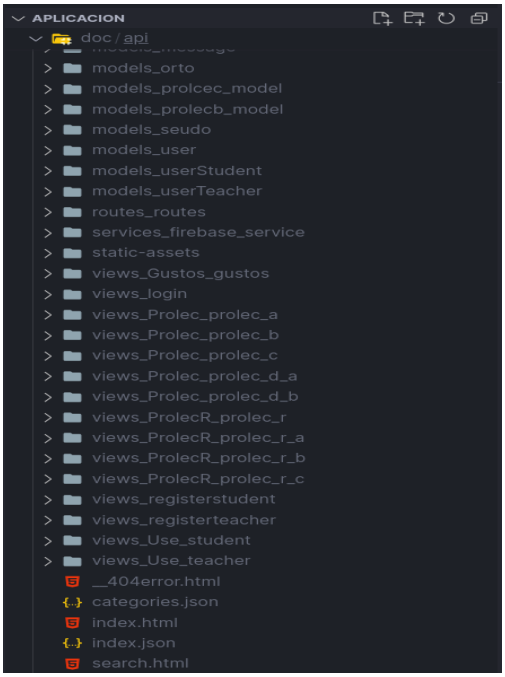
```
apulla@apulla:~/Escritorio/App/FLutter/app-aplicacion$ git status
apulla@apulla:~/Escritorio/App/FLutter/app-aplicacion$ git add .
apulla@apulla:~/Escritorio/App/FLutter/app-aplicacion$ git commit -m "Update Deploy"
apulla@apulla:~/Escritorio/App/FLutter/app-aplicacion$ git push
```

En si con todo esto se ha establecido un eficiente flujo de trabajo con GitHub Actions para automatizar la construcción y despliegue de una aplicación Flutter. La verificación de cambios se enfoca en directorios específicos de Flutter, asegurando que la construcción solo ocurra cuando sea necesario. La aplicación se compila en un entorno Docker, y el despliegue se realiza utilizando Docker Compose, reiniciando los servicios para reflejar los cambios. Este enfoque automatizado mejora la eficiencia del desarrollo y mantiene la coherencia entre el código fuente y la aplicación desplegada.

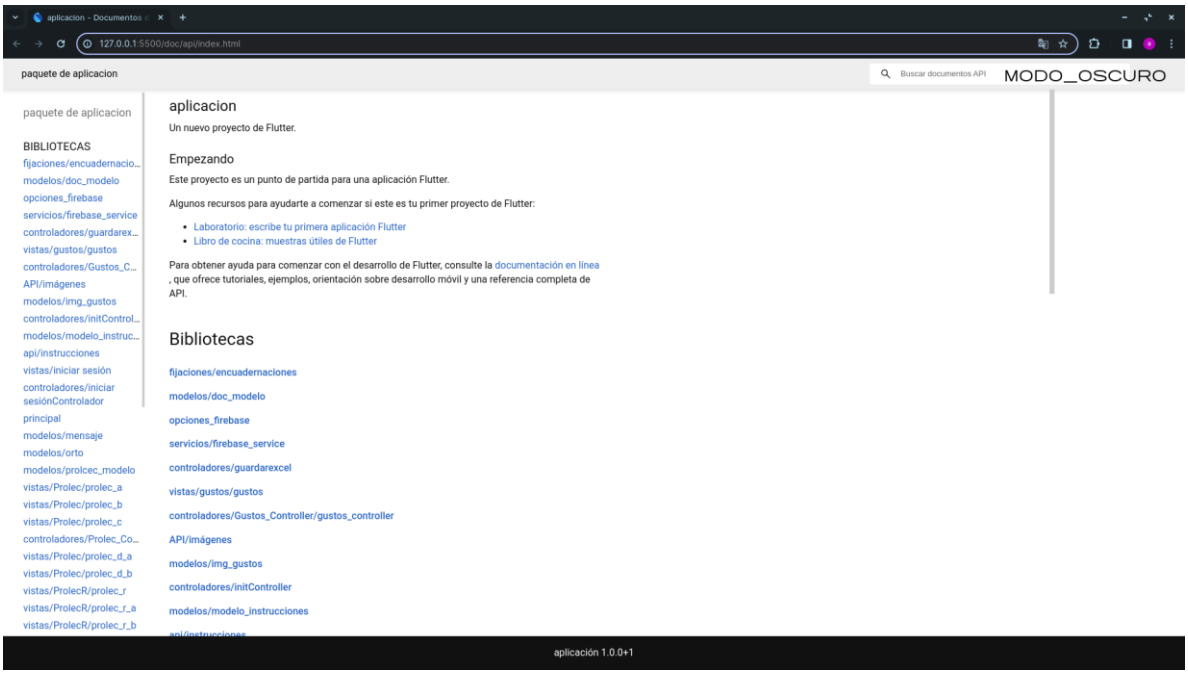
2.3. *Documentación*

Para poder ingresar a la documentación del proyecto este tiene una carpeta que se llama doc donde contiene toda la documentación de nuestro proyecto y dentro de esta tenemos un html el cual lo podemos abrir normalmente.





En este apartado se ve que esta un index.html el cual se puede ejecutar y se nos mostrara toda la documentación de la aplicación.



2.4. *Consideraciones / Recomendaciones*

A continuación, presentamos algunas consideraciones y recomendaciones importantes para tener en cuenta al instalar y utilizar nuestra aplicación:

1. Versión de Flutter: Asegúrate de tener la última versión de Flutter instalada en tu sistema. Esto garantizará la compatibilidad con todas las características de nuestra aplicación.
2. Dispositivo Android: Nuestra aplicación ha sido optimizada para su uso en dispositivos Android. Asegúrate de tener un dispositivo Android compatible para probar la aplicación.
3. Conexión a Internet: Algunas características de nuestra aplicación requieren una conexión a Internet estable. Asegúrate de tener una buena conexión a Internet al utilizar nuestra aplicación.
4. Uso de Firebase: Nuestra aplicación utiliza Firebase para su backend. Asegúrate de seguir las instrucciones de conexión a Firebase detalladas en la sección 2.2.1.