Manual Técnico V1.0.0

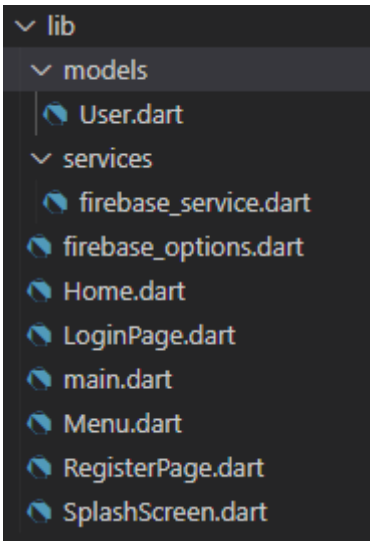
Herramientas digitales

Cómo IDE, se ha utilizado Visual Studio Code para la manipulación de código.
Para persistencia de datos se ha utilizado Firebase - Firestore Database
Como lenguaje de programación se ha utilizado Flutter.

La base de datos se utiliza para la persistencia de datos de Usuarios. Sus puntos, diamantes, usernames, passwords, etc.

Clases

Dentro de la carpeta lib encontramos todas las clases que conforman nuestra aplicación. Allí podremos encontrar subcarpetas. Empezando por la de models. En esta carpeta se encuentra la clase User, la cual actúa como una clase Objeto, representando a un usuario el cual tiene los mismos atributos que en la base de datos.




```dart
class User {
  String username;
  String password;
  int diamonds;
  int points;

  User({
    required this.username,
    required this.password,
    this.diamonds = 0,
    this.points = 0,
  });
}
```

Luego en la carpeta services, encontramos la clase firebase_service.dart. Dicha clase contiene todos los métodos necesarios para interactuar con la base de datos. Hacer consultas, registros, etc.

```dart
import 'package:cloud_firestore/cloud_firestore.dart';

FirebaseFirestore db = FirebaseFirestore.instance;

Future<bool> checkCredentials(String username, String password) async {
  QuerySnapshot snapshot = await db
      .collection('users')
      .where('username', isEqualTo: username)
      .where('password', isEqualTo: password)
      .get();
  return snapshot.docs.isNotEmpty;
}

Future<bool> checkIfUsernameExists(String username) async {
  QuerySnapshot snapshot = await db
      .collection('users')
      .where('username', isEqualTo: username)
      .get();
  return snapshot.docs.isNotEmpty;
}

Future<void> addNewUser(String username, String password) async {
  await db.collection('users').add({
    'username': username,
    'password': password,
  });
}
```

Luego encontraremos un fichero llamado firebase_options.dart el cual se crea por defecto al realizar la conexión con la base de datos. Ahí tendremos algunos datos que necesita la base de datos para realizar la conexión correspondiente.

Por último veremos el resto de clases, que exceptuando por el main.dart, todas son clases que representan cada página independiente de la aplicación.
Se muestra a continuación una breve explicación del funcionamiento o lógica de cada clase.

SplashScreen > Pantalla de carga inicial, dura 5 segundos y desaparece redirigiendo al Login.

LoginPage > Pantalla del Login para acceder a la aplicación, allí también podremos acceder a la pantalla de registro.

RegisterPage > Página de registro para crear un nuevo usuario.

Menu > Primera página que visualizamos al acceder desde el Login, aquí tendremos múltiples opciones como un Drawer desplegable con información de nuestra cuenta, ajustes, el top ranking y la opción de cerrar sesión. Además en la pantalla principal (fuera del drawer) encontramos tres botones, cada uno hará referencia a una dificultad diferente de modos de juego. Por cuestiones de tiempo la única opción disponible es la de modo fácil.

Home > Página que contiene el juego del Ahorcado. Aquí es donde podremos interactuar con la aplicación en su mayoría y disfrutar jugando del clásico juego.

Assets

En la carpeta de Assets encontraremos todas las imágenes que se han utilizado para la creación de nuestra interfaz gráfica. Desde fondos de pantalla, iconos, etc.

Complicaciones

La mayor dificultad a la hora de crear el proyecto fue la conexión a la base de datos. En principio se utilizó un JSON, pero al ser muy enredada la manera de modificar los datos del archivo, lo mejor era pasarse a una base de datos. Firebase fue la opción que se utilizó, pero al principio dio muchos problemas ya que el proyecto no estaba creado desde cero al realizar la conexión. Por lo que se perdió mucho tiempo en este proceso.

Bibliografía

Para ahorrar tiempo y realizar de manera visual el hombrecito del ahorcado, se recurrió a un repositorio público de github. Se ha cogido código de allí, pero se ha Refactorizado (ya que había mucha repetición de código innecesaria y se podía llegar al mismo resultado en menos líneas), Personalizado y Adaptado a la necesidad inicial que pretendía tener el proyecto propio.
A su vez el proyecto sirvió como inspiración para la creación de la UI del juego.

LINK: https://github.com/Shahadan2001/HaNgMaN

Código que pinta al hombrecito:

```dart
class LinePainter extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {
    var paint = Paint()
      ..color = Color.fromARGB(255, 255, 255, 255)
      ..strokeWidth = 10
      ..strokeCap = StrokeCap.round;

    var paint1 = Paint()
      ..color = Color.fromARGB(255, 255, 255, 255)
      ..strokeWidth = 10
      ..strokeCap = StrokeCap.round
      ..style = PaintingStyle.stroke;

    canvas.drawLine(
      Offset(0, size.height),
      Offset(size.width / 4, size.height),
      paint,
    );
    canvas.drawLine(
      const Offset(0, 0),
      Offset(0, size.height),
      paint,
    );
    canvas.drawLine(
      const Offset(0, 0),
      Offset(size.height / 1.5, 0),
      paint,
    );
    canvas.drawLine(
      Offset(size.height / 1.5, 0),
      Offset(size.height / 1.5, size.height / 7),
      paint,
    );
    if (chance >= 1) {
      canvas.drawCircle(
        Offset(size.height / 1.5, size.height / 3.7),
        size.width * 1 / 9,
        paint1,
      );
    }
    if (chance >= 2) {
      canvas.drawLine(
        Offset(size.height / 1.5, size.height / 1.4),
        Offset(size.height / 1.5, size.height / 2.6),
        paint,
      );
    }
```