UNIVERSIDAD PRIVADA “FRANZ TAMAYO”

Ingeniería de Sistemas

PROYECTO HITO-5

CASO: Shitpost Game Memory

Autores:

Alexander Heytan Callisaya Valencia

Henry Javier Huarachi Quispe

Freddy Machaca Mamani

Kevin Oscar Mamani Laura

DOCENTE:

Lic. William Roddy Barra Paredes

El Alto – Bolivia 2022

Contenido

CAPITULO I 4

Introducción 4

1.2. Problema general 5

1.3. Objetivos 6

1.3.1 Objetivos generales 6

1.3.2 Objetivos específicos 6

CAPITULO II 7

MARCO TEORICO 7

2.2 Lenguaje Dart 7

2.2.1 ¿Qué hace a Dart especial? 7

2.3 Framework Flutter 8

2.4 Android Studio 8

2.4.1 Características de Android Studio 9

2.5 El shitposting 9

2.6 Hive 10

2.7 Provider 10

CAPITULO III 12

MARCO APLICATIVO 12

3.1 Describir de manera breve cuál es la aplicación a desarrollarse 12

3.2 Análisis y diseño del sistema utilizando estructura de datos 13

3.2.1 Nombre del proyecto (Sistema): Identificar un nombre adecuado para el sistema. 13

3.3 Código Android Studio, de todo el sistema 14

main.dart 14

constants.dart 15

game_settings.dart 15

theme.dart 15

game_controller.dart 16

game_controller.g.dart 20

game_opcao.dart 21

game_play.dart 22

game_page.dart 22

home_page.dart 23

nivel_page.dart 24

recordes_page.dart 25

recordes_repository.dart 27

recordes_repository.g.dart 28
card_game.dart 30
card_nivel.dart 32
game_score.dart 33
logo.dart 34
recordes.dart 35
start_button.dart 36
pubspec.yaml 37
3.4 Notas 38
3.5 Usabilidad 42
CAPITULO IV 49
4.1 Conclusiones 49

CAPITULO I

Introducción

El juego de memoria es un juego tradicional que consiste en recordar la ubicación de diferentes elementos en un tablero y volver a encontrarlos en pares. En este caso, hemos creado una versión de este juego utilizando el framework de desarrollo de aplicaciones móviles llamado Flutter.

Nuestro juego de memoria en Flutter “Game Memory” ofrece una interfaz de usuario amigable y atractiva, con diferentes temas y niveles de dificultad para que los jugadores puedan desafiarse a sí mismos y mejorar su habilidad de memoria. Además, incluye un sistema de puntaje.

Asimismo, los usuarios podrán disfrutar de una experiencia divertida y retadora mientras desarrollan su capacidad de memoria y atención.

1.2. Problema general

El problema general de las apps de juego de memoria es que, en general, ofrecen experiencias de juego similares y poco innovadoras, con contenido tradicional y poco atractivo. Esto puede generar aburrimiento y desinterés en los usuarios que buscan una experiencia de juego más desafiante y divertida.

Por esta razón, deberían elegir nuestra app "Shitpost game memory" ya que ofrece una experiencia de juego única y divertida, utilizando contenido satírico y cómico en lugar de imágenes tradicionales. Además, la app cuenta con una interfaz de usuario atractiva y fácil de usar, y permite a los jugadores personalizar su experiencia de juego seleccionando diferentes temas y niveles de dificultad. En resumen, nuestra app es una opción innovadora y divertida para los usuarios que buscan un juego de memoria diferente y desafiante.

1.3. Objetivos

1.3.1 Objetivos generales

El objetivo de este juego de memoria hecho en Flutter es proporcionar una forma divertida y desafiante de mejorar la capacidad de memoria y la habilidad en el razonamiento lógico de los jugadores, utilizando una interfaz intuitiva y fácil de usar.

1.3.2 Objetivos específicos

- Proporcionar una plataforma de juego fácil de usar y atractiva visualmente.
- Ofrecer un desafío para la memoria y el razonamiento lógico de los jugadores.
- Brindar una interfaz intuitiva y fácil de navegar.
- Permitir a los jugadores personalizar su experiencia de juego seleccionando los niveles de dificultad.
- Ofrecer un juego divertido y adictivo que mantenga a los jugadores comprometidos y desafiados.

CAPITULO II

MARCO TEORICO

2.2 Lenguaje Dart

Dart es un lenguaje open source desarrollado en Google con el objetivo de permitir a los desarrolladores utilizar un lenguaje orientado a objetos y con análisis estático de tipo. Desde la primera versión estable en 2011, Dart ha cambiado bastante, tanto en el lenguaje en sí como en sus objetivos principales.

2.2.1 ¿Qué hace a Dart especial?

A diferencia de muchos lenguajes, Dart se diseñó con el objetivo de hacer el proceso de desarrollo lo más cómodo y rápido posible para los desarrolladores. Por eso, viene con un conjunto bastante extenso de herramientas integrado, como su propio gestor de paquetes, varios compiladores/transpiladores, un analizador y formateador. Además, la máquina virtual de Dart y la compilación Just-in-Time hacen que los cambios realizados en el código se puedan ejecutar inmediatamente. Una vez en producción, el código se puede compilar en lenguaje nativo, por lo que no es necesario un entorno especial para ejecutar. En caso de que se haga desarrollo web, Dart se transpila a JavaScript.

En cuanto a la sintaxis, la de Dart es muy similar a lenguajes como JavaScript, Java y C ++, por lo que aprender Dart sabiendo uno de estos lenguajes es cuestión de horas.
Además, Dart consta de un gran apoyo para la asincronía, y trabajar con generadores e iterables es extremadamente sencillo.

Dart es un lenguaje de propósito general, y lo puedes utilizar casi para cualquier cosa:

- En aplicaciones web, utilizando la librería de arte: HTML y el transpilador para transformar el código en Dart en JavaScript, o utilizando frameworks como AngularDart.
- En servidores, utilizando las librerías de arte: http y arte: io. También hay varios frameworks que se pueden utilizar, como por ejemplo Aqueduct.
- En aplicaciones de consola.

2.3 Framework Flutter

Flutter es un framework de Dart para crear aplicaciones multiplataforma con un único código. A diferencia de otros frameworks multiplataforma como por ejemplo Ionic, el código de una aplicación de Flutter se compila a código nativo, por lo que el rendimiento alcanzado es superior a aplicaciones basadas en web-views. Además, a diferencia de React Native,

Flutter no utiliza componentes nativos, sino que viene con sus propios componentes, llamados widgets, por lo que la misma aplicación se verá igual en cualquier dispositivo, independientemente de su sistema operativo o la versión. Gracias a ello, el desarrollador no tiene que preocuparse por que el diseño de su aplicación se vea mal en dispositivos antiguos.

Además de aplicaciones móviles, con Flutter también se pueden hacer páginas web y aplicaciones de escritorio, aunque el soporte para páginas web está en beta, y por aplicaciones de escritorio en technical preview, por lo que quien los quiera usar habrá que esperar un tiempo más a que sea estable.
A continuación, tiene varios ejemplos de aplicaciones desarrolladas con Flutter:

- Google Ads
- AppTree
- Reflectly
- inLapp Coffee
- SSHButtons
-

2.4 Android Studio

Normalmente, toda aplicación, herramienta, página, o servidor digital que ofrece algún tipo de tarea en internet, posee lenguajes de programación o entornos de trabajo especializados. Por ejemplo, Python, que es un lenguaje muy utilizado en el desarrollo de Inteligencia Artificial.

Así, tal cual, pasa con el sistema operativo Android. Todas las aplicaciones y herramientas que se desarrollan para este SO en concreto, poseen su propia área o entorno de trabajo. Ese entorno es Android Studio, que permite una flexibilidad en

cuanto al **desarrollo de características y funciones** que puede tener una herramienta o app de dicho sistema.

Este entorno sirve para que las aplicaciones que se estén desarrollando sean mucho más eficiente y autosuficientes. Esto permite, incluso, tener compatibilidades con otros sistemas o plataformas.

2.4.1 Características de Android Studio

Android Studio permite la integración de características y funciones bastante positivas para las aplicaciones que, con el tiempo, se perfeccionan. De esta forma, tenemos lo siguiente:

- El sistema de compilación es flexible, además de ser compatible con Gradle, la cual permite la automatización de compilaciones de forma flexible y con gran rendimiento. Groovy y Kotlin DSL son los lenguajes utilizados para los scripts de compilación.
- La intención de este entorno es la de permitir al usuario trabajar de forma fluida y con una gran cantidad de funciones prácticas y útiles.
- Esta plataforma te permite desarrollar aplicaciones para cualquier dispositivo Android.
- Contiene plantillas de compilación que te ayudan a otorgar funciones comunes de otras apps de forma mucho más rápida, además de importar códigos de muestra.
- Mayor cantidad de herramientas de prueba con marcos de trabajo.
- Modificar fragmentos de código y recursos de una app sin necesidad de que esta se reinicie.
- Proporciona compatibilidad con servicios en la nube tal como Google Cloud Platform.
- Compatibilidad con lenguajes como NDK y C++.

2.5 El shitposting

El **shitposting** (del inglés shit, mierda; y posting, publicar) es una jerga de internet que se refiere al acto de publicar intencionalmente contenido de calidad pobre,

agresiva, inútil, irónica y actuando como troll, mashups, irrelevancia, incoherencia, spam no comercial, errores de ortografía o gramática. Todos son sellos distintivos del shitposting (hecho mayormente con propósitos cómicos) en un foro de Internet o red social, en algunos casos para desviar las discusiones, causar mayor reacción con el menor esfuerzo posible o para lograr que el sitio sea inútil para sus visitantes regulares.

Según el historiador italiano Steven Forti, el shitposting es un recurso muy utilizado por la extrema derecha para trollear y atacar a los adversarios políticos o sencillamente a los normies [a la gente normal] y llenar de contenido de baja calidad las redes sociales para desviar las discusiones y conseguir que lo publicado en un sitio sea inútil o, como mínimo, pierda su valor.

2.6 Hive

Hive es una base de datos nosql ideal para proyectos no complejos y con un volumen no muy grande de datos, está escrito en dart y almacena en cajas (box) los datos en par clave-valor. Estas cajas no tienen una estructura definida y pueden almacenar cualquier tipo de dato. Por defecto las cajas almacenan todos los tipos de datos primitivos con su respectiva clave pero también existe la flexibilidad de crear objetos llamados adaptadores (type adatapters). En este último se centra esta publicación para realizar el CRUD.

Un adaptador es un objeto creado por el usuario para almacenar ciertos tipos de datos, este es un archivo estático y generado a través del archivo adaptador base. En este archivo base se deben definir ciertas anotaciones, tanto para la propia clase como para los atributos de la clase,

2.7 Provider

Provider, es un paquete que ha sido lanzado recientemente (en Google I/O) el cual tiene como función, permitirnos manejar el estado de nuestra app. Como sabemos, en Flutter para poder construir una App, y alterar el aspecto visual de la misma, debemos trabajar con el “Estado de la App”. La forma más sencilla de hacer esto, es mediante el uso de “StateFul Widgets” y la función de “setState”.

Eso puede funcionar para alguna aplicación muy pequeña, sin embargo, cuando estamos construyendo una app que contiene varias pantallas y hay que acceder a esas propiedades en todas, el proceso se vuelve muy complejo.

Existen alternativas a Provider, como el patron BloC, o ScopedModel. Sin embargo, han dado justo en el punto con el paquete de Provider.

CAPITULO III

MARCO APLICATIVO

3.1 Describir de manera breve cuál es la aplicación a desarrollarse

"Game Memory" trata de un juego de memoria hecho en flutter que ofrece divertidas imágenes para encontrar pares. El usuario tendrá que recordar la ubicación de cada imagen para poder encontrar los pares. El objetivo es encontrar todos los pares y ganar el juego.

El juego también ofrece una tabla de clasificación para que los usuarios puedan ver su progreso y ver quién es el mejor jugador. El jugador tendrá la posibilidad de elegir entre varios niveles de dificultad, para adaptarse a sus habilidades y habilidades. El juego también ofrece diversos temas para que los usuarios se diviertan con la experiencia de juego.

3.2 Análisis y diseño del sistema utilizando estructura de datos

3.2.1 Nombre del proyecto (Sistema): Identificar un nombre adecuado para el sistema.

El nombre "Shitpost game memory" se refiere a un juego de memoria que utiliza contenido de "shitposting", una práctica en internet de publicar contenido de baja calidad o irrelevante.

El nombre se eligió ya que el juego utiliza contenido satírico y cómico, en lugar de imágenes tradicionales utilizadas en juegos de memoria, lo que lo diferencia de otros juegos similares y lo convierte en una experiencia divertida y única. Además, el nombre refleja la naturaleza irreverente y provocativa del contenido utilizado en el juego.

3.3 Código Android Studio, de todo el sistema

Library

main.dart

```
import 'package:flutter/material.dart';
import
'package:flutter_memory_game/controllers/game_controller.dart';
import 'package:flutter_memory_game/pages/home_page.dart';
import
'package:flutter_memory_game/repositories/recordes_repository.dart';
import 'package:flutter_memory_game/theme.dart';
import 'package:hive_flutter/adapters.dart';
import 'package:provider/provider.dart';

void main() async {
  await Hive.initFlutter();

  runApp(MultiProvider(
    providers: [
      Provider<RecordesRepository>(create: (_) =>
RecordesRepository()),
      ProxyProvider<RecordesRepository, GameController>(
        update: (_, repo, __) => GameController(recordesRepository:
repo),
      ),
    ],
    child: const App(),
  ));
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Shitpost Memory',
      debugShowCheckedModeBanner: false,
      theme: ShitpostThemeColor.theme,
      home: const HomePage(),
    );
  }
}
```

constants.dart

```
enum Modo {
  normal,
  dificil,
}

enum Resultado {
  aprobado,
  eliminado,
}
```

game_settings.dart

```
class GameSettings {
  static const niveles = [6, 8, 10, 12, 16, 18, 20, 24, 28];

  static const cardOpcoes = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14];

  static gameBoardAxisCount(int nivel) {
    if (nivel < 10) {
      return 2;
    } else if (nivel == 10 || nivel == 12 || nivel == 18) {
      return 3;
    } else {
      return 4;
    }
  }
}
```

theme.dart

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

class ShitpostThemeColor {
  static const MaterialColor color = MaterialColor(
    _round6PrimaryValue,
    <int, Color>{
      50: Color(0xFFFCE4EC),
      100: Color(0xFFF8BBD0),
      200: Color(0xFFF48FB1),
      300: Color(0xFFF06292),
      400: Color(0xFFEC407A),
      500: Color(_round6PrimaryValue),
      600: Color(0xFFD81B60),
      700: Color(0xFFC2185B),
      800: Color(0xFFAD1457),
      900: Color(0xFF880E4F),
    },
  );
  static const int _round6PrimaryValue = 0XFFFF1D87;

  static const Color background = Color(0XFF121212);

  static ButtonStyle outlineButtonStyle({
```

```
    Color color = Colors.white,
    double padding = 24,
  }) {
    return OutlinedButton.styleFrom(
      primary: color,
      padding: EdgeInsets.symmetric(vertical: padding),
      side: BorderSide(color: color),
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(100)),
      ),
    );
  }

  static ThemeData theme = ThemeData(
    brightness: Brightness.dark,
    scaffoldBackgroundColor: background,
    primarySwatch: color,
    primaryColor: color,
    textTheme: GoogleFonts.wendyOneTextTheme(
      ThemeData.dark().textTheme,
    ),
    outlinedButtonTheme: OutlinedButtonThemeData(
      style: outlineButtonStyle(),
    ),
    appBarTheme: ThemeData.dark().appBarTheme.copyWith(
          elevation: 0,
          centerTitle: true,
          backgroundColor: Colors.transparent,
          titleTextStyle: GoogleFonts.wendyOne(fontSize: 25),
        ),
  );
}
```

Package Controllers

game_controller.dart

```
import 'package:flutter_memory_game/constants.dart';
import 'package:flutter_memory_game/game_settings.dart';
import 'package:flutter_memory_game/models/game_opcao.dart';
import 'package:flutter_memory_game/models/game_play.dart';
import
'package:flutter_memory_game/repositories/recordes_repository.dart';
import 'package:mobx/mobx.dart';

part 'game_controller.g.dart';

class GameController = GameControllerBase with _$GameController;
```

```
abstract class GameControllerBase with Store {
  @observable
  List<GameOpcion> gameCards = [];
  @observable
  int score = 0;
  @observable
  bool ganaste = false;
  @observable
  bool perdiste = false;

  late GamePlay _gamePlay;
  List<GameOpcion> _escojer = [];
  List<Function> _escojaCallback = [];
  int _aciertos = 0;
  int _numPares = 0;
  RecordesRepository recordesRepository;

  @computed
  bool get jugadaCompleta => (_escojer.length == 2);

  GameControllerBase({required this.recordesRepository}) {
    reaction((_) => ganaste == true, (bool gano) {
      if (gano) {
        recordesRepository.updateRecordes(gamePlay: _gamePlay,
score: score);
      }
    });
  }

  startGame({required GamePlay gamePlay}) {
    _gamePlay = gamePlay;
    _aciertos = 0;
    _numPares = (_gamePlay.nivel / 2).round();
    ganaste = false;
    perdiste = false;
    _resetScore();
    _generateCards();
  }

  _resetScore() {
    _gamePlay.modo == Modo.normal ? score = 0 : score =
_gamePlay.nivel;
  }

  _generateCards() {
    List<int> cardOpciones =
```

```
GameSettings.cardOpcoes.sublist(0)..shuffle();
    cardOpciones = cardOpciones.sublist(0, _numPares);
    gameCards = [...cardOpciones, ...cardOpciones]
        .map((opcion) =>
            GameOpcion(opcion: opcion, matched: false, selected:
false))
        .toList();
    gameCards.shuffle();
  }

  escojer(GameOpcion opcion, Function resetCard) async {
    opcion.selected = true;
    _escojer.add(opcion);
    _escojaCallback.add(resetCard);
    await _compararEscolhas();
  }

  _compararEscolhas() async {
    if (jugadaCompleta) {
      if (_escojer[0].opcion == _escojer[1].opcion) {
        _aciertos++;
        _escojer[0].matched = true;
        _escojer[1].matched = true;
      } else {
        await Future.delayed(const Duration(seconds: 1), () {
          for (var i in [0, 1]) {
            _escojer[i].selected = false;
            _escojaCallback[i]();
          }
        });
      }

      _resetJugada();
      _updateScore();
      _checkGameResult();
    }
  }

  _checkGameResult() async {
    bool allMatched = _aciertos == _numPares;
    if (_gamePlay.modo == Modo.normal) {
      await _checkResultModoNormal(allMatched);
    } else {
      await _checkResultModoRound6(allMatched);
    }
  }
```

```
  _checkResultModoNormal(bool allMatched) async {
    await Future.delayed(
        const Duration(seconds: 1), () => ganaste = allMatched);
  }

  _checkResultModoRound6(bool allMatched) async {
    if (_chancesAcabaram()) {
      await Future.delayed(
          const Duration(milliseconds: 400), () => perdiste = true);
    }
    if (allMatched && score >= 0) {
      await Future.delayed(
          const Duration(seconds: 1), () => ganaste = allMatched);
    }
  }

  _chancesAcabaram() {
    return score < _numPares - _aciertos;
  }

  _resetJugada() {
    _escojer = [];
    _escojaCallback = [];
  }

  _updateScore() {
    _gamePlay.modo == Modo.normal ? score++ : score--;
  }

  restartGame() {
    startGame(gamePlay: _gamePlay);
  }

  nextLevel() {
    int nivelIndex = 0;

    if (_gamePlay.nivel != GameSettings.niveles.last) {
      nivelIndex = GameSettings.niveles.indexOf(_gamePlay.nivel) +
1;
    }

    _gamePlay.nivel = GameSettings.niveles[nivelIndex];
    startGame(gamePlay: _gamePlay);
  }
}
```

```dart
game_controller.g.dart
part of 'game_controller.dart';

mixin _$GameController on GameControllerBase, Store {
  Computed<bool>? _$jogadaCompletaComputed;

  @override
  bool get jugadaCompleta =>
      (_$jogadaCompletaComputed ??= Computed<bool>(() =>
super.jugadaCompleta,
              name: 'GameControllerBase.jogadaCompleta'))
          .value;

  final _$gameCardsAtom = Atom(name:
'GameControllerBase.gameCards');

  @override
  List<GameOpcion> get gameCards {
    _$gameCardsAtom.reportRead();
    return super.gameCards;
  }

  @override
  set gameCards(List<GameOpcion> value) {
    _$gameCardsAtom.reportWrite(value, super.gameCards, () {
      super.gameCards = value;
    });
  }

  final _$scoreAtom = Atom(name: 'GameControllerBase.score');

  @override
  int get score {
    _$scoreAtom.reportRead();
    return super.score;
  }

  @override
  set score(int value) {
    _$scoreAtom.reportWrite(value, super.score, () {
      super.score = value;
    });
  }
```

```
  final _$venceuAtom = Atom(name: 'GameControllerBase.venceu');

  @override
  bool get ganaste {
    _$venceuAtom.reportRead();
    return super.ganaste;
  }

  @override
  set ganaste(bool value) {
    _$venceuAtom.reportWrite(value, super.ganaste, () {
      super.ganaste = value;
    });
  }

  final _$perdeuAtom = Atom(name: 'GameControllerBase.perdeu');

  @override
  bool get perdiste {
    _$perdeuAtom.reportRead();
    return super.perdiste;
  }

  @override
  set perdiste(bool value) {
    _$perdeuAtom.reportWrite(value, super.perdiste, () {
      super.perdiste = value;
    });
  }

  @override
  String toString() {
    return '''
gameCards: ${gameCards},
score: ${score},
venceu: ${ganaste},
perdeu: ${perdiste},
jogadaCompleta: ${jugadaCompleta}
    ''';
  }
}
```

PACKAGE MODELS

game_opcao.dart

```
class GameOpcion {
  int opcion;
```

```
  bool matched;
  bool selected;

  GameOpcion(
      {required this.opcion, required this.matched, required
this.selected});
}
```

game_play.dart

```
import 'package:flutter_memory_game/constants.dart';

class GamePlay {
  Modo modo;
  int nivel;

  GamePlay({required this.modo, required this.nivel});
}
```

PACKAGE PAGES

game_page.dart

```
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import
'package:flutter_memory_game/controllers/game_controller.dart';
import 'package:flutter_memory_game/game_settings.dart';
import 'package:flutter_memory_game/models/game_opcao.dart';
import 'package:flutter_memory_game/models/game_play.dart';
import 'package:flutter_memory_game/theme.dart';
import 'package:flutter_memory_game/widgets/card_game.dart';
import 'package:flutter_memory_game/widgets/feedback_game.dart';
import 'package:flutter_memory_game/widgets/game_score.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:provider/provider.dart';

class GamePage extends StatelessWidget {
  final GamePlay gamePlay;

  const GamePage({Key? key, required this.gamePlay}) : super(key:
key);

  @override
  Widget build(BuildContext context) {
    final controller = Provider.of<GameController>(context);
    return Scaffold(
      appBar: AppBar(
```

```
        automaticallyImplyLeading: false,
        title: GameScore(modo: gamePlay.modo),
      ),
      // body: const FeedbackGame(resultado: Resultado.eliminado),

      body: Observer(
        builder: (_) {
          if (controller.ganaste) {
            return const FeedbackGame(resultado:
Resultado.aprobado);
          } else if (controller.perdiste) {
            return const FeedbackGame(resultado:
Resultado.eliminado);
          } else {
            return Center(
              child: GridView.count(
                shrinkWrap: true,
                crossAxisCount:
GameSettings.gameBoardAxisCount(gamePlay.nivel),
                mainAxisSpacing: 15,
                crossAxisSpacing: 15,
                padding: const EdgeInsets.all(24),
                children: controller.gameCards
                    .map(
                      (GameOpcion go) =>
                          CardGame(modo: gamePlay.modo, gameOpcion:
go),
                    )
                    .toList(),
              ),
            );
          }
        },
      ),
    );
  }
}
```

home_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import 'package:flutter_memory_game/pages/nivel_page.dart';
import 'package:flutter_memory_game/theme.dart';
import 'package:flutter_memory_game/widgets/logo.dart';
import 'package:flutter_memory_game/widgets/recordes.dart';
import 'package:flutter_memory_game/widgets/start_button.dart';
```

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

  selecionarNivel(Modo modo, BuildContext context) async {
    await Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => NivelPage(modo: modo),
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Padding(
        padding: const EdgeInsets.all(24.0),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            const Logo(),
            StartButton(
              title: 'Modo Normal',
              color: Colors.white,
              action: () => selecionarNivel(Modo.normal, context),
            ),
            StartButton(
              title: 'Modo Insano',
              color: ShitpostThemeColor.color,
              action: () => selecionarNivel(Modo.dificil, context),
            ),
            const SizedBox(height: 60),
            const Recordes(),
          ],
        ),
      ),
    );
  }
}
```

nivel_page.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import 'package:flutter_memory_game/game_settings.dart';
```

```dart
import 'package:flutter_memory_game/models/game_play.dart';
import 'package:flutter_memory_game/widgets/card_nivel.dart';

class NivelPage extends StatelessWidget {
  final Modo modo;

  const NivelPage({Key? key, required this.modo}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final niveis = GameSettings.niveles
        .map((n) => CardNivel(gamePlay: GamePlay(modo: modo, nivel:
n)))
        .toList();
    return Scaffold(
      appBar: AppBar(
        title: const Text('Nível del Juego'),
      ),
      body: Padding(
        padding: const EdgeInsets.only(top: 48),
        child: GridView.count(
          crossAxisCount: 3,
          mainAxisSpacing: 20,
          crossAxisSpacing: 20,
          padding: const EdgeInsets.all(24),
          children: niveis,
        ),
      ),
    );
  }
}
```

recordes_page.dart

```dart
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import
'package:flutter_memory_game/repositories/recordes_repository.dart';
import 'package:flutter_memory_game/theme.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:provider/provider.dart';

class RecordesPage extends StatelessWidget {
  final Modo modo;

  const RecordesPage({Key? key, required this.modo}) : super(key:
key);
```

```
getModo() {
  return modo == Modo.normal ? 'Normal' : 'Dificil';
}

List<Widget> getRecordesList(Map recordes) {
  final List<Widget> widgets = [];

  recordes.forEach((nivel, score) {
    widgets.add(ListTile(
      title: Text('Nível $nivel'),
      trailing: Text(score.toString()),
      tileColor: Colors.grey[900],
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(15)),
      ),
    ));

    widgets.add(const Divider(color: Colors.transparent));
  });

  if (widgets.isEmpty) {
    widgets.add(
      const Center(
        child: Text('AUN NO AHI RECORDS!'),
      ),
    );
  }

  return widgets;
}

@override
Widget build(BuildContext context) {
  final repository = Provider.of<RecordesRepository>(context);

  return Scaffold(
    appBar: AppBar(
      title: const Text('Recordes'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(12.0),
      child: Observer(
        builder: (context) => Column(
          mainAxisAlignment: MainAxisAlignment.start,
          children: [
```

```
              Padding(
                padding: const EdgeInsets.only(top: 36, bottom: 24),
                child: Center(
                  child: Text(
                    'Modo ${getModo()}',
                    style: const TextStyle(
                        fontSize: 28, color:
ShitpostThemeColor.color),
                    ),
                ),
              ),
              ...getRecordesList(modo == Modo.normal
                  ? repository.recordesNormal
                  : repository.recordesRound6),
            ],
          ),
        ),
      ),
    );
  }
}
```

PACKAGE REPOSITORIES

recordes_repository.dart

```
import 'package:flutter_memory_game/constants.dart';
import 'package:flutter_memory_game/models/game_play.dart';
import 'package:hive/hive.dart';
import 'package:mobx/mobx.dart';

part 'recordes_repository.g.dart';

class RecordesRepository = RecordesRepositoryBase with
_$RecordesRepository;

abstract class RecordesRepositoryBase with Store {
  late final Box _recordes;
  late final GamePlay gamePlay;
  @observable
  Map recordesRound6 = {};
  @observable
  Map recordesNormal = {};

  RecordesRepositoryBase() {
    _initRepository();
  }
```

```
  _initRepository() async {
    await _initDatabase();
    await loadRecordes();
  }

  _initDatabase() async {
    _recordes = await Hive.openBox('gameRecordes3');
  }

  @action
  loadRecordes() {
    recordesNormal = _recordes.get(Modo.normal.toString()) ?? {};
    recordesRound6 = _recordes.get(Modo.dificil.toString()) ?? {};
  }

  updateRecordes({required GamePlay gamePlay, required int score}) {
    final key = gamePlay.modo.toString();

    if (gamePlay.modo == Modo.normal &&
        (recordesNormal[gamePlay.nivel] == null ||
            score < recordesNormal[gamePlay.nivel])) {
      recordesNormal[gamePlay.nivel] = score;
      _recordes.put(key, recordesNormal);
    } else if (gamePlay.modo == Modo.dificil &&
        (recordesRound6[gamePlay.nivel] == null ||
            score > recordesRound6[gamePlay.nivel])) {
      recordesRound6[gamePlay.nivel] = score;
      _recordes.put(key, recordesRound6);
    }
  }
}
```

recordes_repository.g.dart

```
part of 'recordes_repository.dart';

mixin _$RecordesRepository on RecordesRepositoryBase, Store {
  final _$recordesRound6Atom =
      Atom(name: 'RecordesRepositoryBase.recordesRound6');

  @override
  Map<dynamic, dynamic> get recordesRound6 {
    _$recordesRound6Atom.reportRead();
    return super.recordesRound6;
```

```
  }

  @override
  set recordesRound6(Map<dynamic, dynamic> value) {
    _$recordesRound6Atom.reportWrite(value, super.recordesRound6, ()
{
      super.recordesRound6 = value;
    });
  }

  final _$recordesNormalAtom =
      Atom(name: 'RecordesRepositoryBase.recordesNormal');

  @override
  Map<dynamic, dynamic> get recordesNormal {
    _$recordesNormalAtom.reportRead();
    return super.recordesNormal;
  }

  @override
  set recordesNormal(Map<dynamic, dynamic> value) {
    _$recordesNormalAtom.reportWrite(value, super.recordesNormal, ()
{
      super.recordesNormal = value;
    });
  }

  final _$RecordesRepositoryBaseActionController =
      ActionController(name: 'RecordesRepositoryBase');

  @override
  dynamic loadRecordes() {
    final _$actionInfo =
_$RecordesRepositoryBaseActionController.startAction(
        name: 'RecordesRepositoryBase.loadRecordes');
    try {
      return super.loadRecordes();
    } finally {

_$RecordesRepositoryBaseActionController.endAction(_$actionInfo);
    }
  }

  @override
  String toString() {
    return '''
```

```
recordesRound6: ${recordesRound6},
recordesNormal: ${recordesNormal}
    ''';
  }
}
```

PACKAGE WIDGETS

card_game.dart

```
import 'dart:async';
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import
'package:flutter_memory_game/controllers/game_controller.dart';
import 'package:flutter_memory_game/models/game_opcao.dart';
import 'package:flutter_memory_game/theme.dart';
import 'package:provider/provider.dart';

class CardGame extends StatefulWidget {
  final Modo modo;
  final GameOpcion gameOpcion;

  const CardGame({
    Key? key,
    required this.modo,
    required this.gameOpcion,
  }) : super(key: key);

  @override
  State<CardGame> createState() => _CardGameState();
}

class _CardGameState extends State<CardGame>
    with SingleTickerProviderStateMixin {
  late final AnimationController animation;

  @override
  void initState() {
    super.initState();
    animation = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 400),
    );
  }

  @override
```

```
  void dispose() {
    animation.dispose();
    super.dispose();
  }

  AssetImage getImage(double angulo) {
    if (angulo > 0.5 * pi) {
      return
AssetImage('images/${widget.gameOpcion.opcion.toString()}.png');
    } else {
      return widget.modo == Modo.normal
          ? const AssetImage('images/card_normal.png')
          : const AssetImage('images/card_round.png');
    }
  }

  flipCard() {
    final game = context.read<GameController>();

    if (!animation.isAnimating &&
        !widget.gameOpcion.matched &&
        !widget.gameOpcion.selected &&
        !game.jugadaCompleta) {
      animation.forward();
      game.escojer(widget.gameOpcion, resetCard);
    }
  }

  resetCard() {
    animation.reverse();
  }

  @override
  Widget build(BuildContext context) {
    return AnimatedBuilder(
      animation: animation,
      builder: (BuildContext context, _) {
        final angulo = animation.value * pi;
        final transform = Matrix4.identity()
          ..setEntry(3, 2, 0.002)
          ..rotateY(angulo);

        return GestureDetector(
          onTap: () => flipCard(),
          child: Transform(
            transform: transform,
```

```
            alignment: Alignment.center,
            child: Container(
              decoration: BoxDecoration(
                border: Border.all(
                  color: widget.modo == Modo.normal
                      ? Colors.white
                      : ShitpostThemeColor.color,
                  width: 2,
                ),
                borderRadius: const
BorderRadius.all(Radius.circular(5)),
                image: DecorationImage(
                  fit: BoxFit.cover,
                  image: getImage(angulo),
                ),
              ),
            ),
          ),
        );
      },
    );
  }
}
```

card_nivel.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import 'package:flutter_memory_game/controllers/game_controller.dart';
import 'package:flutter_memory_game/models/game_play.dart';
import 'package:flutter_memory_game/pages/game_page.dart';
import 'package:flutter_memory_game/theme.dart';
import 'package:provider/provider.dart';

class CardNivel extends StatelessWidget {
  final GamePlay gamePlay;

  const CardNivel({Key? key, required this.gamePlay}) : super(key: key);

  startGame(BuildContext context) {
    context.read<GameController>().startGame(gamePlay: gamePlay);

    Navigator.push(
      context,
      MaterialPageRoute(
        fullscreenDialog: true,
        builder: (BuildContext context) => GamePage(gamePlay: gamePlay),
```

```
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: () => startGame(context),
      borderRadius: const BorderRadius.all(Radius.circular(10)),
      child: Container(
        width: 90,
        height: 90,
        padding: const EdgeInsets.all(8),
        decoration: BoxDecoration(
          border: Border.all(
            color: gamePlay.modo == Modo.normal
                ? Colors.white
                : ShitpostThemeColor.color,
          ),
          borderRadius: const BorderRadius.all(Radius.circular(10)),
          color: gamePlay.modo == Modo.normal
              ? Colors.transparent
              : ShitpostThemeColor.color.withOpacity(.6),
        ),
        child: Center(
          child: Text(gamePlay.nivel.toString(),
              style: const TextStyle(fontSize: 30)),
        ),
      ),
    );
  }
}
```

game_score.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import 'package:flutter_memory_game/controllers/game_controller.dart';
import 'package:flutter_mobx/flutter_mobx.dart';
import 'package:provider/provider.dart';

class GameScore extends StatelessWidget {
  final Modo modo;
  const GameScore({Key? key, required this.modo}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final controller = Provider.of<GameController>(context);

    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
```

```
      children: [
        Row(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Icon(modo == Modo.dificil
                ? Icons.my_location
                : Icons.touch_app_rounded),
            const SizedBox(width: 10),
            Observer(
                builder: (_) => Text(controller.score.toString(),
                    style: const TextStyle(fontSize: 25))),
          ],
        ),
        Image.asset('images/host.png', width: 38, height: 60),
        TextButton(
          child: const Text('Salir', style: TextStyle(fontSize: 18)),
          onPressed: () => Navigator.pop(context),
        ),
      ],
    );
  }
}
```

logo.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/theme.dart';

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

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Padding(
          padding: const EdgeInsets.only(bottom: 20),
          child: Image.asset('images/host.png', width: 200, height: 125),
        ),
        Padding(
          padding: const EdgeInsets.only(bottom: 40),
          child: RichText(
            text: TextSpan(
              text: 'SHITPOST ',
              style: DefaultTextStyle.of(context).style.copyWith(fontSize:
30),
              children: const [
                TextSpan(
                  text: 'Memory',
                  style: TextStyle(color: ShitpostThemeColor.color),
                )
```

```
              ],
            ),
          ),
        ),
      ],
    );
  }
}
```

recordes.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/constants.dart';
import 'package:flutter_memory_game/pages/recordes_page.dart';
import 'package:flutter_memory_game/theme.dart';

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

  @override
  State<Recordes> createState() => _RecordesState();
}

class _RecordesState extends State<Recordes> {
  showRecordes(Modo modo) async {
    await Navigator.push(
      context,
      MaterialPageRoute(
        builder: (BuildContext context) => RecordesPage(modo: modo),
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Card(
      color: Colors.grey[900],
      child: Padding(
        padding: const EdgeInsets.all(12.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const Padding(
              padding: EdgeInsets.all(12),
              child: Text(
                'Records',
```

```
                style: TextStyle(
                  color: ShitpostThemeColor.color,
                  fontSize: 22,
                ),
              ),
            ),
            ListTile(
              title: const Text('Modo Normal'),
              trailing: const Icon(Icons.chevron_right),
              onTap: () => showRecordes(Modo.normal),
            ),
            ListTile(
              title: const Text('Modo Insano'),
              trailing: const Icon(Icons.chevron_right),
              onTap: () => showRecordes(Modo.dificil),
            )
          ],
        ),
      ),
    );
  }
}
```

start_button.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_memory_game/theme.dart';

class StartButton extends StatelessWidget {
  final String title;
  final Color color;
  final Function action;

  const StartButton({
    Key? key,
    required this.title,
    required this.color,
    required this.action,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(top: 24),
      child: OutlinedButton(
        style: ShitpostThemeColor.outlineButtonStyle(color: color),
        onPressed: () => action(),
```

```
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              title,
              style: const TextStyle(fontSize: 20),
            )
          ],
        ),
      ),
    );
  }
}
```

pubspec.yaml

```
name: flutter_memory_game
description: A new Flutter project.

environment:
  sdk: ">=2.12.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  google_fonts: ^2.1.0
  provider: ^6.0.1
  flutter_mobx: ^2.0.2
  hive: ^2.0.4
  hive_flutter: ^1.1.0


  cupertino_icons: ^1.0.2

dev_dependencies:
  flutter_test:
    sdk: flutter
  build_runner: ^2.1.5
  mobx_codegen: ^2.0.4


  assets:
    - images/
```

3.4 Notas

Clase Main

```
void main() async {
  await Hive.initFlutter();

  runApp(MultiProvider(
    providers: [
      Provider<RecordesRepository>(create: (_) =>
RecordesRepository()),
      ProxyProvider<RecordesRepository, GameController>(
        update: (_, repo, __) => GameController(recordesRepository:
repo),
      ),
    ],
    child: const App(),
  ));
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Shitpost Memory',
      debugShowCheckedModeBanner: false,
      theme: ShitpostThemeColor.theme,
      home: const HomePage(),
    );
  }
}
```

Clase con el tema de Color

```
class ShitpostThemeColor {
  static const MaterialColor color = MaterialColor(
    _round6PrimaryValue,
    <int, Color>{
      50: Color(0xFFFCE4EC),
      100: Color(0xFFF8BBD0),
      200: Color(0xFFF48FB1),
      300: Color(0xFFF06292),
      400: Color(0xFFEC407A),
      500: Color(_round6PrimaryValue),
      600: Color(0xFFD81B60),
```

```
      700: Color(0xFFC2185B),
      800: Color(0xFFAD1457),
      900: Color(0xFF880E4F),
    },
  );
  static const int _round6PrimaryValue = 0XFFFF1D87;

  static const Color background = Color(0XFF121212);

  static ButtonStyle outlineButtonStyle({
    Color color = Colors.white,
    double padding = 24,
  }) {
    return OutlinedButton.styleFrom(
      primary: color,
      padding: EdgeInsets.symmetric(vertical: padding),
      side: BorderSide(color: color),
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.all(Radius.circular(100)),
      ),
    );
  }

  static ThemeData theme = ThemeData(
    brightness: Brightness.dark,
    scaffoldBackgroundColor: background,
    primarySwatch: color,
    primaryColor: color,
    textTheme: GoogleFonts.wendyOneTextTheme(
      ThemeData.dark().textTheme,
    ),
    outlinedButtonTheme: OutlinedButtonThemeData(
      style: outlineButtonStyle(),
    ),
    appBarTheme: ThemeData.dark().appBarTheme.copyWith(
          elevation: 0,
          centerTitle: true,
          backgroundColor: Colors.transparent,
          titleTextStyle: GoogleFonts.wendyOne(fontSize: 25),
        ),
  );
}
```

Clase con las configuraciones del juego

```
class GameSettings {
  static const niveles = [6, 8, 10, 12, 16, 18, 20, 24, 28];

  static const cardOpcoes = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
13, 14];

  static gameBoardAxisCount(int nivel) {
    if (nivel < 10) {
      return 2;
    } else if (nivel == 10 || nivel == 12 || nivel == 18) {
      return 3;
    } else {
      return 4;
    }
  }
}
```

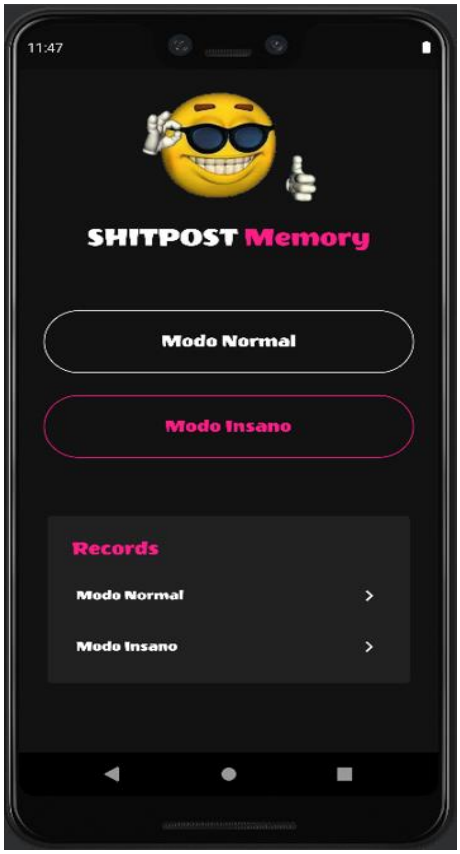
Clase del título del menú principal

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

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Padding(
          padding: const EdgeInsets.only(bottom: 20),
          child: Image.asset('images/host.png', width: 200, height:
125),
        ),
        Padding(
          padding: const EdgeInsets.only(bottom: 40),
          child: RichText(
            text: TextSpan(
              text: 'SHITPOST ',
              style:
DefaultTextStyle.of(context).style.copyWith(fontSize: 30),
              children: const [
                TextSpan(
                  text: 'Memory',
                  style: TextStyle(color: ShitpostThemeColor.color),
                )
              ],
            ),
          ),
        ),
      ],
    );
  }
```

3.5 Usabilidad

a. Adjuntar imágenes del sistema.

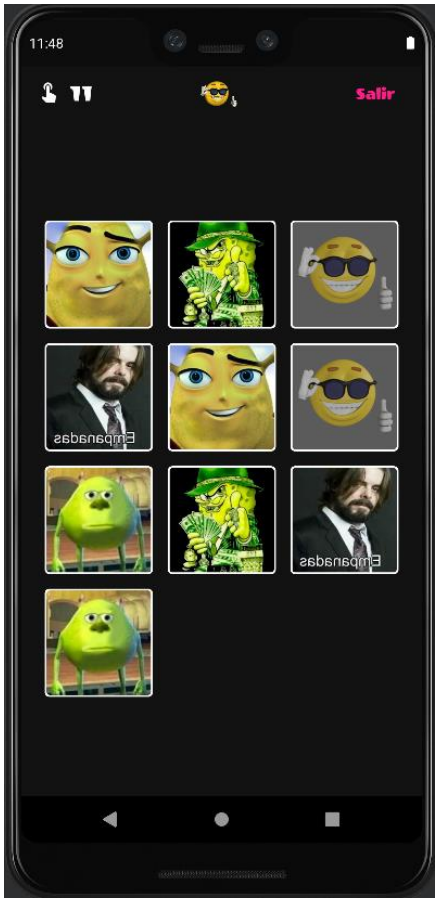


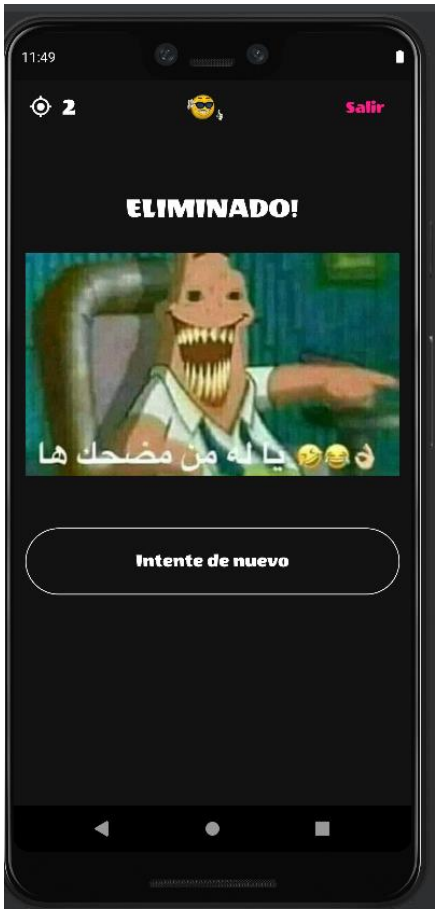

11:47
0
Salir


11:47
4
Salir
Empanadas


5
Salir
APROVADO!
FELICIDADES
Próximo Nível

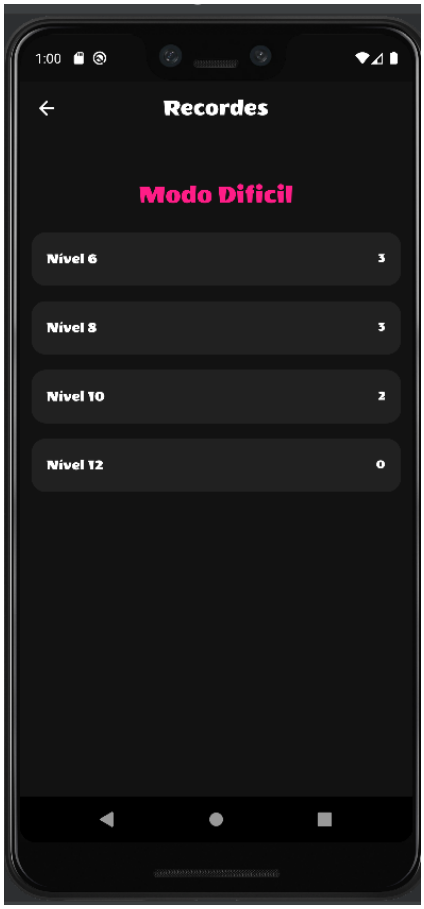

11:48
0
Salir

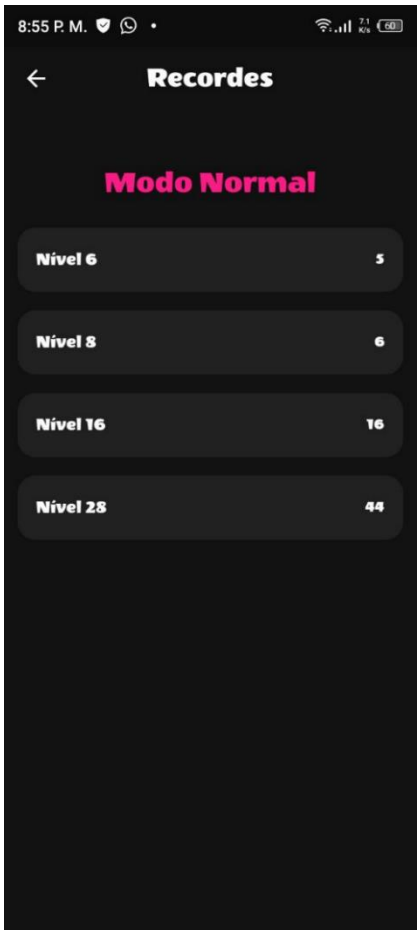

11:48
Salir


11:49
2
Salir
ELIMINADO!
يا له من مضحك ها
Intente de nuevo


12:58
10
Salir


11:51
4
Salir


1:00
Recordes
Modo Dificil
Nivel 6
3
Nivel 8
3
Nivel 10
2
Nivel 12
0


8:55 P. M.
Recordes
Modo Normal
Nivel 6
5
Nivel 8
6
Nivel 16
16
Nivel 28
44

b. Adjuntar un video de la funcionalidad del sistema.

https://www.youtube.com/watch?v=QXyVN8uVU1E

CAPITULO IV

4.1 Conclusiones

Después del proceso de programación de la app "Shitpost game memory", llegamos a las siguientes conclusiones:

Utilizar el framework Flutter nos permitió desarrollar la app de forma rápida y sencilla, con una interfaz de usuario atractiva y fácil de usar.

La amplia gama de componentes y herramientas disponibles en Flutter nos ayudó a crear un juego divertido y adictivo, con diferentes niveles de dificultad.

La flexibilidad y facilidad de uso de Flutter nos permitió implementar un sistema de puntuación que motiva a los jugadores a mejorar su desempeño.

En general, el proceso de programación utilizando Flutter fue eficiente y nos permitió crear una app de alta calidad que cumple con las expectativas de los jugadores.