Redin
Manual Técnico

- **Proyecto**: Redin
- **Tecnologías**: Flutter ,Dart
- **Minijuegos**: Blackjack, Roulette, Horse Racing
- **Proveedor de aleatoriedad**: Random.org
- **Autor**: Emilio Fernández Gallardo
- **Fecha**: 07/02/2025

Contenidos

Introducción

Este documento tiene como objetivo describir en detalle la arquitectura, diseño e implementación técnica del proyecto **Redin**, desarrollado con Flutter y Dart.

Este proyecto esta compuesto por tres minijuegos clásicos: **Blackjack**, **Roulette** y **Horse Racing**. Para garantizar la aleatoriedad en las jugadas, se ha implementado una integración con la API de Random.org, la cual provee números verdaderamente aleatorios. Se ha utilizado SQFlite con el propósito de almacenar localmente la cantidad de "coins" que posee el usuario.

Objetivos del Proyecto

- **Desarrollo Multijuego:** Implementar tres minijuegos distintos con reglas y lógicas propias.
- **Aleatoriedad Garantizada:** Utilizar Random.org para obtener resultados aleatorios y seguros en cada partida.
- **Plataforma Flutter:** Aprovechar las capacidades de Flutter para crear una interfaz “responsive” y atractiva.
- **Escalabilidad y Mantenimiento:** Diseñar una arquitectura modular que permita futuras ampliaciones y mejoras.

Arquitectura del Sistema

El sistema se estructura en tres capas principales:

1. **Capa de Presentación:**
   - Implementada en Flutter.
   - Gestiona la UI/UX de los minijuegos.
   - Se apoya en el patrón de diseño Provider para el manejo del estado.
2. **Lógica:**
   - Implementada en Dart.
   - Contiene la lógica específica de cada minijuego (reglas, flujo de juego, etc).
   - Se han creado widgets comunes y reutilizables entre los diferentes juegos.
3. **Integración Externa:**
   - Módulo de conexión a la API de Random.org.
   - Responsable de gestionar las peticiones HTTP y el tratamiento de respuestas para la generación de números aleatorios.

Requerimientos del Sistema

Software y Herramientas

- **Flutter:** Versión 3.24.4
- **Dart:** Versión 3.5.4
- **IDE:** Visual Studio Code, Android Studio o cualquier editor compatible.
- **Control de Versiones:** Git.
- **Gestor de Dependencias:** Pub (flutter pub).

Dependencias Principales

- **Provider:** Para el manejo del estado en Flutter.
- **Http:** Para las peticiones a la API de Random.org.
- **Google Fonts:** Para fuentes personalizadas
- **SQFlite:** Para guardar localmente la cantidad de coins de un usuario
- **AudioPlayers:** Para reproducir sonidos o música
- **Toast:** Para mostrar mensajes por pantalla rápidos
-

Requisitos del Dispositivo

- **Plataformas Soportadas:** Android, iOS
- **Requisitos Mínimos:** Dispositivos con versiones actualizadas de sistemas operativos compatibles con Flutter.

Instalación y Configuración

Pasos para la Instalación

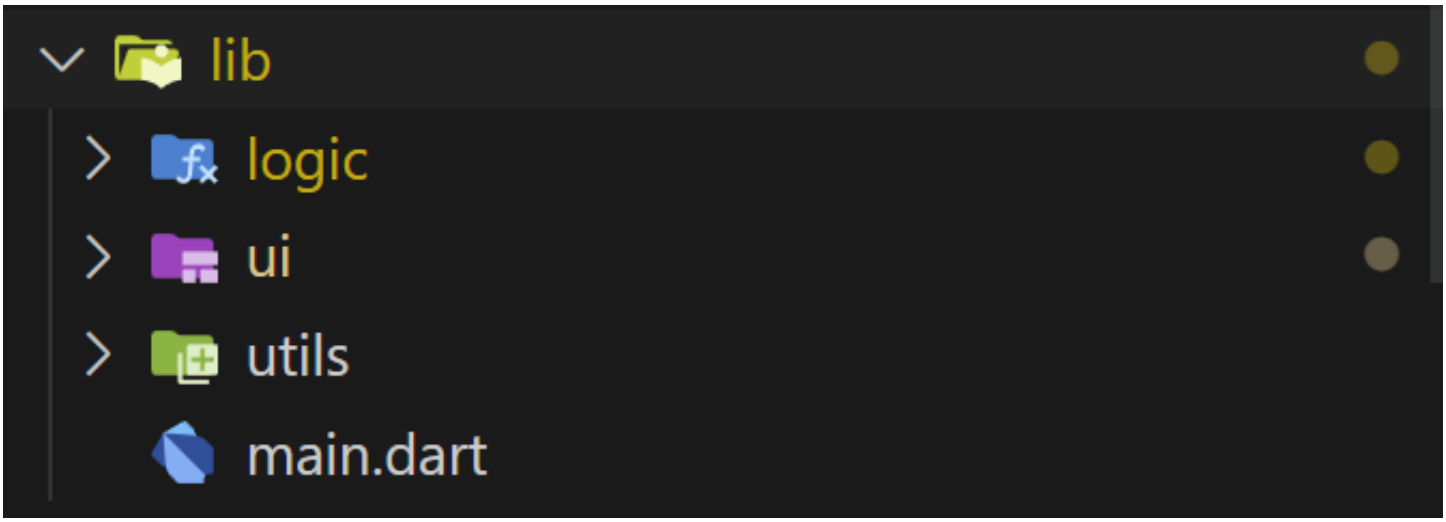
1. **Clonar el Repositorio:**
2. git clone https://github.com/EmilioFdez12/Redin-Proyecto-Flutter
3. Ejecutar flutter clean en la terminal de VSCode (por ejemplo)
4. Ejecutar flutter pub get
5. **Ejecutar la Aplicación:**
6. flutter run

Estructura del Proyecto

Vamos a centrarnos en la carpeta **lib** que es donde se encuentra el código.

Dentro de lib nos encontramos:

3 carpetas



Logic: dentro de esta carpeta encontramos otras **tres carpetas,** una para cada minijuego. En cada una de ellas se maneja la lógica de su respectivo minijuego.

Ui: dentro de esta carpeta encontramos dos mas, **Screens** y **Widgets**,en Screens se encuentran cada una de las pantallas que componen la aplicacióny en Widgets encontramos todos y cada uno de los widgets que son utilizados en cada pantalla algunos se comparten y otros están hechos para una pantalla en especifico.

Utils: por ultimo encontramos la carpeta **Utils**, dentro tenemos la carpeta **Music** y **Database**, en Music manejamos todo lo que tiene que ver con los sonidos o la música dentro la app , y en database tenemos un crud y un clase que se encarga de quitar o añadir los coins en función de lo que apueste el usuario.

Integración con Random.org API

La integración con la API de Random.org es fundamental para garantizar la aleatoriedad en los resultados de los minijuegos, en este caso el Blackjack (la api es limitada)

Funcionalidad

- **Objetivo:** Obtener una baraja de cartas totalmente barajada aleatoriamente

- **Método:** Peticiones HTTP POST al endpoint de Random.org.
- **Formato de Respuesta:** JSON.

Detalle de los Minijuegos

Blackjack

- **Descripción:** Juego de cartas en el que el objetivo es sumar 21 sin pasarse.
- **Lógica del Juego:**
  - Distribución inicial de cartas al jugador y al dealer.
  - Decisión del jugador: solicitar otra carta o plantarse.
  - Uso de Random.org para asignar valores aleatorios a las cartas.
- **Componentes Principales:**
  - **UI:** Pantalla de juego, botón de "HIT" (pedir carta), "Stand" (plantarse), etc.
  - **Provider:** Manejo del estado del juego y del saldo.
  - **Modelo:** Representación de cartas.

Roulette

- **Descripción:** Juego basado en una ruleta en la que el jugador apuesta a números o colores.
- **Lógica del Juego:**
  - Registro de las apuestas del usuario.
  - Giro de la ruleta: obtención de un número aleatorio que determina el resultado.
  - Validación y cálculo de ganancias.

Horse Racing

- **Descripción:** Simulación de una carrera de caballos en la que se determinan posiciones y ganadores.
- **Lógica del Juego:**
  - Inicialización de la carrera y asignación de probabilidades.
  - Actualización en tiempo real de la posición de cada caballo.
- **Componentes:**
  - **UI:** Animación de la carrera, marcador de posiciones.
  - **Provider y Modelos:** Gestión de datos en tiempo real y actualización de la carrera.

DAFO

Fortalezas (Diferenciadores y Ventajas)

Uso de Flutter: Desarrollo multiplataforma con una sola base de código.
Aleatoriedad Garantizada: Integración con la API de Random.org para obtener números verdaderamente aleatorios.
Arquitectura Modular: Uso de Provider y widgets reutilizables para facilitar el mantenimiento.
Interfaz Atractiva: Diseño responsive y uso de Google Fonts y AudioPlayers para mejorar la experiencia del usuario.

Debilidades (Limitaciones y Áreas de Mejora)

Dependencia de Random.org: La API tiene limitaciones de uso gratuito, lo que podría afectar la experiencia del usuario si se superan los límites.
Base de Datos Local: Actualmente, solo se almacena el saldo del usuario sin sincronización en la nube, lo que puede generar pérdida de datos si se desinstala la aplicación.

Oportunidades (Posibilidades de Crecimiento)

Expansión de Juegos: Se pueden agregar más minijuegos para aumentar la variedad y retención de usuarios.
Monetización: Implementación de compras dentro de la app o anuncios para generar ingresos.
Sincronización en la Nube: Integración con Firebase.

Amenazas (Factores Externos que Pueden Afectar el Proyecto)

Competencia con Apps Similares: Existen bastantes juegos de casino en el mercado, lo que puede dificultar destacar.
Costos de API y Servidores: Si la app crece, el uso intensivo de la API de Random.org o la implementación de una base de datos en la nube puede generar costos adicionales.

Problemas durante el desarrollo y mejoras posibles

Durante el desarrollo de la aplicación he tenido bastantes problemas pero a destacar son, sobre todo con la superposición de imágenes por ejemplo en el juego de blackjack para que una carta se pusiese encima de otra y se quedasen centradas.

Con el AudioManager también tuve problemas, a veces se duplicaba la música de fondo y otras no se paraba cuando salías de la aplicación.

Para mejorar de la aplicación tengo varias en mente, de interfaz poder elegir la apuesta con teclado, y poner también recompensas diarias por entrar a la aplicación esas son las que veo mas interesantes

Bibliografía

Inspiración para diseño: BLACK JACK UI • FLUTTER CODE FROM SCRATCH

Documentación en general: Flutter documentation

Para la api de números aleatorios: RANDOM.ORG - True Random Number Service