¡21!

Manual técnico

Índice

1. Introducción

¡21! es una aplicación basada en un juego de cartas de azar, donde puedes luchar contra la máquina en quién saca mejor puntuación sin sobrepasar 21 puntos.

2. Análisis del problema

a. Problemática

¡21! es un juego que otorga entretenimiento al jugador, de partidas cortas, donde puedes mejorar tu puntuación basada en las victorias obtenidas.

b. Clientes potenciales

Cualquier usuario móvil que desee estar entretenido durante un corto periodo de tiempo y le gusten los juegos basados en el azar.

c. Análisis DAFO

d. Monetización y beneficios

La aplicación no está pensada para ser monetizada, una aplicación tan básica podría atraer a muchas personas por su uso, así que hacerla gratuita haría que tuviera mucho público. En el futuro, se podría poner mejoras para monetizar esas actualizaciones, o simplemente ponerle un precio de compra ridículo a la aplicación.

3. Diseño de la solución

a. Tecnologías elegidas

Dart y Flutter.
Trabajado en Virtual Studio Code.

b. Consideraciones técnicas

Versión de Dart: 3.1.3
Versión de Flutter: 3.13.6
Testeado con un Pixel 3a API 30

c. Clases y sus funciones

1. **MyApp**
   - 1.1. **Widget build(BuildContext context):** Construye la interfaz de la aplicación.

2. **HomeScreen**
   - 2.1. **_HomeScreenState createState() => _HomeScreenState():** Crea el estado de _HomeScreenState.

3. **_HomeScreenState**
   - 3.1. **void initState():** Inicia el widget y configura y configura el valor de la puntuación.
   - 3.2. **_actualizarPuntuacion():** Actualiza la puntuación desde la instancia.
   - 3.3. **void _mostrarDialogoConfirmacion(BuildContext context):** Muestra una ventana emergente para resetear la puntuación.
   - 3.4. **Widget build(BuildContext context):** Construye la interfaz.

4. **Juego**
   - 4.1. **_JuegoState createState() => _JuegoState():** Crea el estado de _JuegoState.

5. _JuegoState

5.1. **void initState():** Inicia la mano del jugador, del adversario y la baraja. Ambas manos roban una carta.

5.2. **int _sumarCartas(List<int> cartas):** Suma las cartas de una mano y devuelve el total.

5.3. **String _ mostrarPrimeraCarta(List<int> cartas):** Muestra el número de la primera carta, y el resto como “?”.

5.4. **void _realizarAccionRobar(Mano mano):** Roba una carta de la baraja y alterna el estado de los botones.

5.5. **void _realizarAccionPasar(Mano mano):** Pasa el turno y alterna el estado de los botones.

5.6. **void _turnoContrario():** Gestiona el turno del adversario, robando o pasando turno, y comprueba y finaliza el juego si ambos jugadores pasaron.

5.7. **void _comprobarYFinalizar():** Comprueba los valores de las manos y actualiza la puntuación mostrándolo en una ventana emergente.

5.8. **String _estadoJugador(Mano mano):** Devuelve un string de la última acción de la mano.

5.9. **Widget _buildCartasMias(List<int> cartas):** Construye la visualización de las cartas del jugador.

5.10. **Widget _buildCartasAdversario(List<int> cartas):** Construye la visualización de las cartas del adversario.

5.11. **Widget build(BuildContext context):** Construye la interfaz del juego.

6. Instrucciones

6.1. **Widget build(BuildContext context):** Construye la interfaz de las instrucciones.

7. Creditos

7.1. **Widget build(BuildContext context):** Construye la interfaz de los créditos.

8. Baraja

8.1. **Baraja() : cartas = <int>[]:** Constructor, genera una lista de números del 1 al 11 y aleatoriza el orden.

9. Mano

9.1. **void robarCarta(Baraja baraja):** Roba una carta que va desde la mano de la baraja a la mano del jugador. Establece pasar como false.

9.2. **void pasarTurno():** Pasa turno. Establece pasar como true.

9.3. **bool suEstado():** Obtiene el booleano de pasar.

10. Puntuacion

10.1. **int get puntuacion => _puntuacion:** devuelve el valor de la puntuación.
10.2. **set puntuacion(int valor):** establece y guarda la puntuación en persistencia.
10.3. **Future<void> _guardarPuntuacion(int puntuacion) async:** Guarda la puntuación en persistencia.
10.4. **Future<void> cargarPuntuacion() async:** Carga la puntuación. Establece la puntuación como 0 si no existe.
10.5. **void incrementar(int valor):** Suma la puntuación con el valor proporcionado. Actualiza la nueva puntuación.
10.6. **Future<void> resetearPuntuacion() async:** Reinicia la puntuación a 0 y lo guarda en persistencia.

d. Posibles mejoras

- El Drawer debe estar un poco mejor estructurado.
- Las instrucciones deben ser más y un poco más claras.
- El botón “Pasar” debería estar desactivado cuando el jugador haya conseguido 21 puntos o más.
- El final de la partida.
  - Las cartas del adversario deben mostrarse cuando termina la partida.
  - El modo en que el juego no muestra las puntuaciones no es del todo llamativo.
  - Puede no volver al menú si al pulsar la ventana emergente no se pulsa “OK”.

4. Documentación de la solución

https://github.com/Meowzaur/ProyectoFlutter

5. Enlaces de interés

- Documentación general de Dart: https://dart.dev/guides
- Documentación general de Flutter: https://docs.flutter.dev/
- Documentación de cómo utilizar SharedPreferences: https://www.cybrosys.com/blog/how-to-use-shared-preferences-in-flutter
- Documentación de ejemplos de ShowDialog: https://flutterdesk.com/showdialog-flutter/