Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza bajaForma

**Instituto Tecnológico de Costa Rica**

**Lenguajes de Programación**

**Proyecto 4: Juego LuiKi Kart**

**Profesor:**

**Allan Rodríguez Dávila**

**Estudiantes:**

**Jimena Méndez Morales - 2023113347**

**Ricardo Arce Aguilar - 2023215990**

**I Semestre, 2025**

Tabla de Contenidos

[**Tabla de Figuras** 2](#_Toc201593794)

[**Manual de usuario** 3](#_Toc201593795)

[**Pruebas de funcionalidad** 4](#_Toc201593796)

[**Descripción del problema** 9](#_Toc201593797)

[**Diseño del programa** 10](#_Toc201593798)

[**Librerías usadas** 11](#_Toc201593799)

[**Análisis de resultados** 13](#_Toc201593800)

[**Bitácora** 15](#_Toc201593801)

**Tabla de Figuras**

[*Ilustración 1. Ingreso de usuario* 5](#_Toc201829288)

[*Ilustración 2. Menú principal* 6](#_Toc201829289)

[*Ilustración 3. Creación*  *de partida* 6](#_Toc201829290)

[*Ilustración 4. Verificar estado del juego* 7](#_Toc201829291)

[*Ilustración 5. Lobby* 7](#_Toc201829292)

[*Ilustración 6. Error de jugadores preparados* 8](#_Toc201829293)

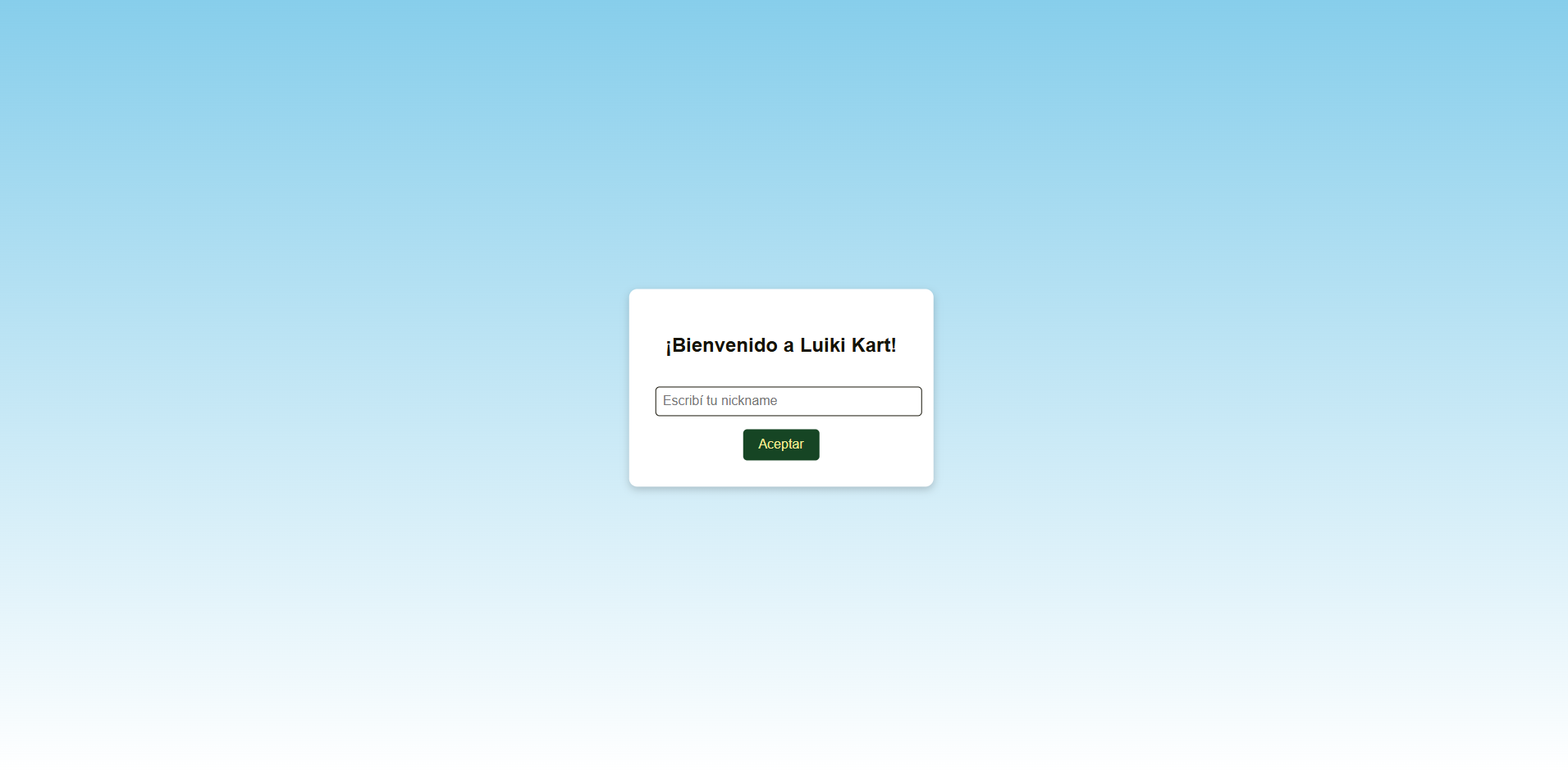
[*Ilustración 7. Todos los jugadores preparados* 8](#_Toc201829294)

**Manual de usuario**

Enlace del video:

**Pruebas de funcionalidad**

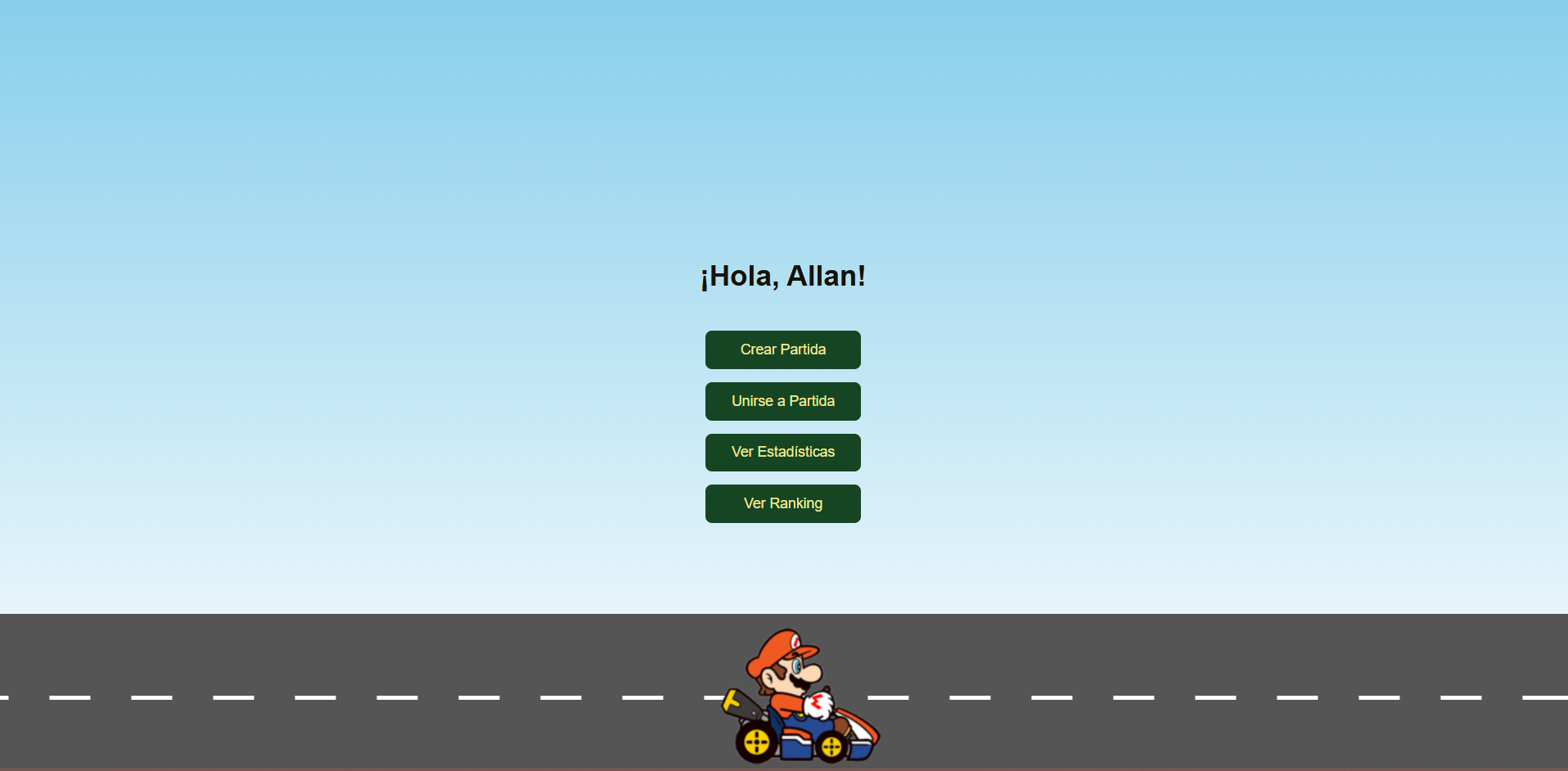
1. Ingreso del nickname: El usuario ingresa el nombre que desea para el juego.



*Ilustración 1. Ingreso de usuario*

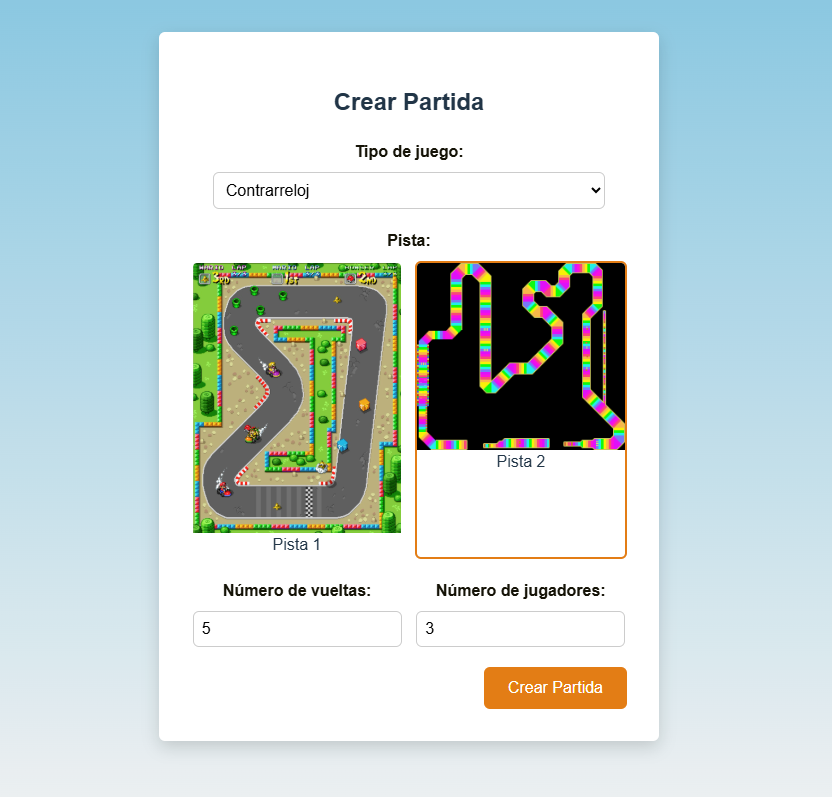
1. Menú principal: El usuario puede acceder a distintas secciones:

* Crear una partida
* Unirse a una partida
* Ver estadísticas de la partida que jugó
* Ver ranking



*Ilustración 2. Menú principal*

1. Crear partida: Al crear una partida, se ingresan los campos como qué tipo de juego desea jugar, la pista, la cantidad de rondas o vueltas y la cantidad de jugadores.



*Ilustración 3. Creación de partida*

1. Unirse a una partida: Una vez creada esa partida, se pueden crear otros usuarios para unirse a ella como el ejemplo.



*Ilustración 4. Verificar estado del juego*

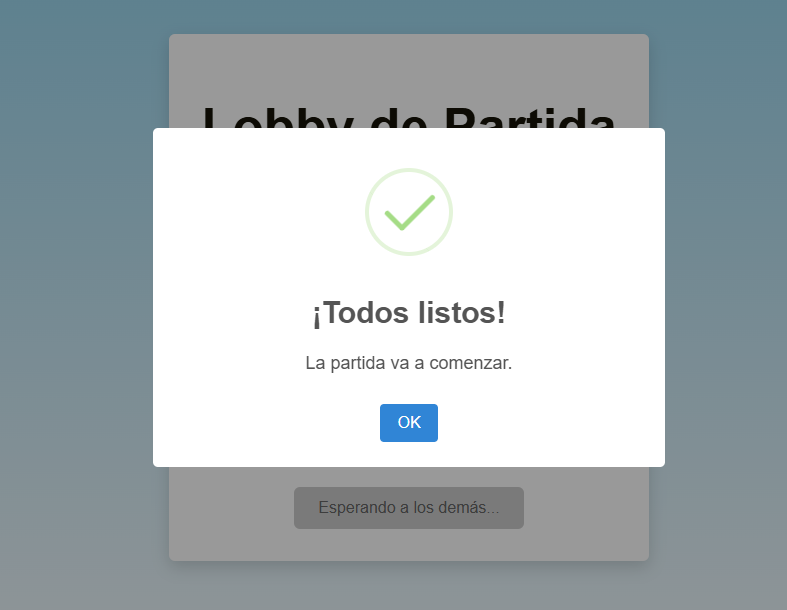
1. Lobby: Al momento que se une un usuario o el usuario crea la partida, entonces se puede visualizar el lobby donde todos están a la espera, también es importante identificar que si no está la cantidad completa de número de jugadores, entonces no deja que los demás estén listos como en la imagen 7 y si todos están listos, se notifica igualmente como la imagen 8.



*Ilustración 5. Lobby*



*Ilustración 6. Error de jugadores preparados*



*Ilustración 7. Todos los jugadores preparados*

**Descripción del problema**

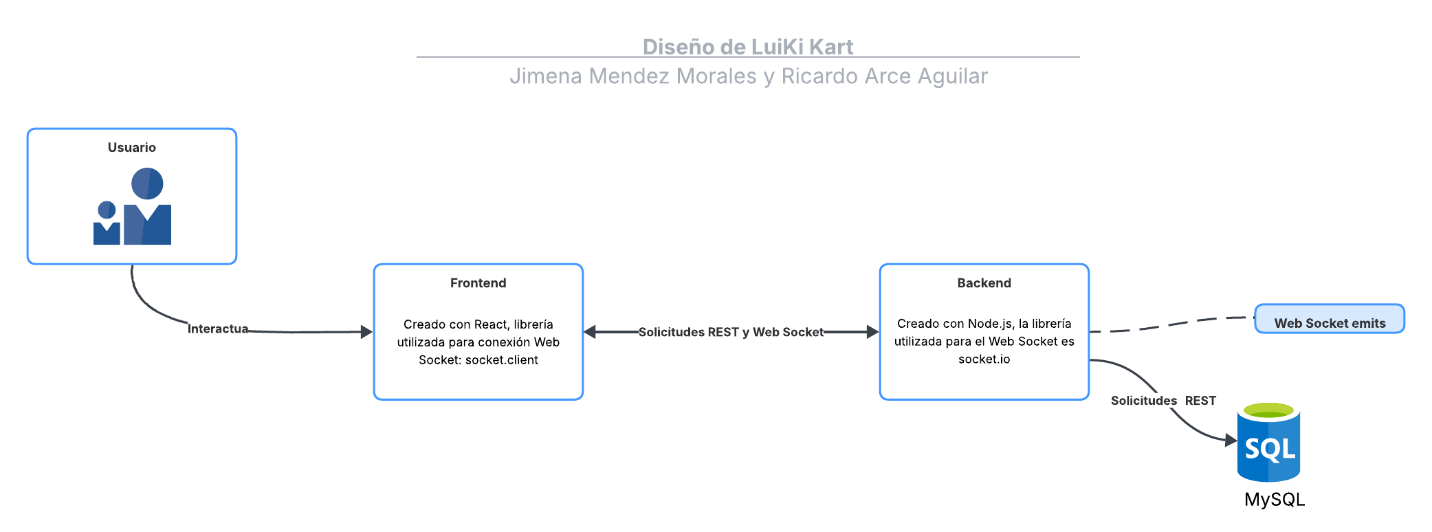
El Sudoku es un juego de lógica numérica originado en Japón, conocido por desafiar la capacidad de razonamiento de sus jugadores. Consiste en una cuadrícula de 9×9 dividida en subcuadrículas de 3×3 que debe ser completada con los dígitos del 1 al 9, sin repetirlos en ninguna fila, columna o región. Si bien su estructura es simple, resolver un tablero correctamente puede representar un reto significativo, especialmente cuando el número de pistas iniciales es reducido.

Este proyecto busca simular dicho juego mediante la implementación de un sistema informático que no solo reproduzca las reglas del Sudoku, sino que también brinde una experiencia interactiva para el usuario. Para lograrlo, se requiere un enfoque híbrido entre el paradigma lógico (usando SWI-Prolog) y la programación orientada a objetos (con Java y su biblioteca Swing).

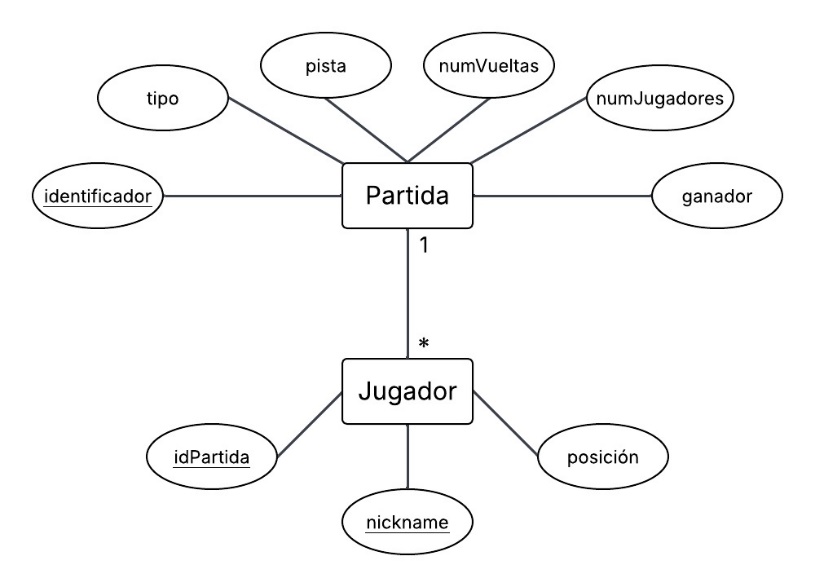
La generación de tableros aleatorios, la validación de jugadas, la autosolución del juego, y la sugerencia de movimientos se delegan al backend en Prolog. Por otro lado, la interfaz gráfica de usuario se implementa en Java, permitiendo al jugador interactuar con el sistema de forma intuitiva, ingresando valores, solicitando sugerencias, verificando su progreso y observando estadísticas detalladas.

**Diseño del programa**

* Arquitectura planeada del proyecto:



* Diagrama Entidad-Relación de la base de datos:



**Librerías usadas**

**Backend (Node.js + Express)**

* **mysql2/promise**: Esta librería permite conectarse a bases de datos MySQL utilizando Promesas. Facilita el uso de async/await, lo que hace que las consultas a la base de datos sean más limpias y fáciles de manejar.
* **dotenv**: Se encarga de cargar variables de entorno desde un archivo .env. Esto es útil para proteger información sensible como contraseñas, claves API o configuraciones del entorno.
* **express**: Framework web ligero y rápido para construir APIs REST. Simplifica la gestión de rutas, solicitudes HTTP y middlewares.
* **http**: Módulo nativo de Node.js que permite crear un servidor HTTP. Se usa para integrarlo con socket.io y permitir la comunicación en tiempo real.
* **socket.io**: Biblioteca que permite implementar WebSockets para comunicación bidireccional entre el cliente y el servidor. Se utiliza para sincronizar eventos del juego como el estado de los jugadores.
* **cors**: Middleware que permite manejar políticas de acceso entre diferentes dominios (CORS). Es esencial cuando el frontend y el backend están en servidores diferentes.
* **router (./routes.js)**: Archivo que contiene las rutas REST definidas para interactuar con las distintas funcionalidades del sistema (crear partidas, asignar jugadores, etc.).

**Frontend (React.js)**

* **react**: Biblioteca principal para construir interfaces de usuario en componentes reutilizables. Se encarga del renderizado y la gestión del estado visual de la aplicación.
* **react-router-dom**: Permite implementar navegación entre diferentes vistas dentro de una aplicación de una sola página (SPA). Se utiliza para manejar rutas como /lobby, /home, /game, etc.
* **sweetalert2**: Biblioteca para mostrar alertas modernas, visualmente atractivas y personalizables, como notificaciones de error o confirmaciones.
* **createContext**: Parte del sistema de contexto de React. Se usa para compartir información global (como el socket) entre componentes sin pasar props manualmente.
* **BrowserRouter**:
* Componente que envuelve la aplicación y permite el manejo del historial del navegador y las rutas amigables.

**Análisis de resultados**

|  |  |  |
| --- | --- | --- |
| **Objetivos** | **Alcanzado** | **No alcanzado** |
| El jugador debe poder elegir un nickname. |  |  |
| El jugador debe poder crear una partida. |  |  |
| El jugador debe poder seleccionar el tipo de juego de una partida, ya sea en modo versus o modo contrarreloj. |  |  |
| El jugador debe poder seleccionar el número de vueltas a recorrer en una partida. |  |  |
| El jugador debe poder seleccionar el número de jugadores en una partida. |  |  |
| El jugador debe poder unirse a una partida existente. |  |  |
| El jugador debe poder marcarse como “preparado”. |  |  |
| El jugador debe poder cambiar la dirección de su vehículo usando los botones de flechas. |  |  |
| El vehículo del jugador debe detenerse si choca con una pared o el vehículo de otro jugador. | No se logró por temas de tiempo |  |
| El jugador debe poder ver el nombre de los demás jugadores durante la partida. | No se logró por temas de tiempo |  |
| El jugador debe poder ver una advertencia cuando se esté desplazando en sentido antihorario. | No se logró por temas de tiempo |  |
| El jugador debe poder ver el ganador de la partida. | No se logró por temas de tiempo |  |
| El jugador debe poder ver una sección de estadísticas en la que se detalle, por cada juego, el nombre de los jugadores, su tiempo o posición, la pista y el identificador de partida. |  |  |
| El jugador debe poder ver una sección de ranking en la que se detalle, por cada partida, el ganador, su tiempo, la pista, el número de vueltas y el identificador de partida. |  |  |

**Bitácora**

Enlace del repositorio: <https://github.com/jimendezm/proyecto-4-lp.git>