Instituto Tecnológico de Costa Rica

Centro Académico de Limón

Lenguajes de programación

Profesor:

Allan Rodriguez Davila

Estudiante:

Elder León Pérez 2023166120

Owen Torres Porras 2023302034

Brasly villarebia Morales 2023105915

Semestre 2

Entrega Proyecto 4

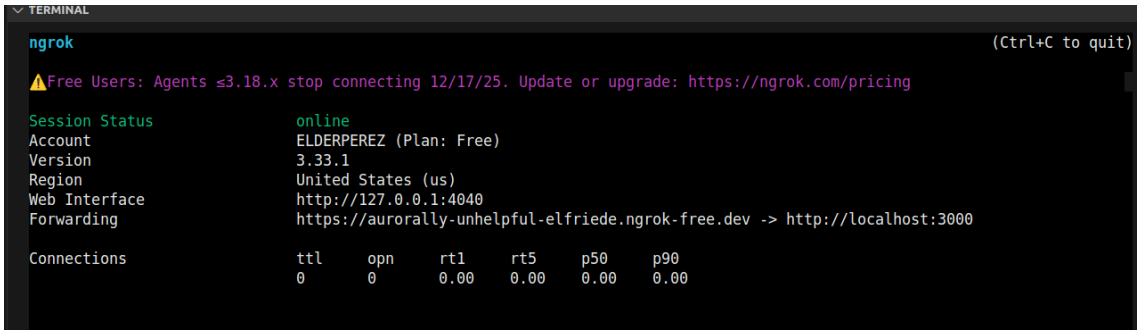
Manual de usuario

Asegúrese de haber clonado el repositorio correctamente antes de correr lo siguiente.

Instrucciones de compilación

Server.js

1. Ingrese a la siguiente dirección en una terminal (cd Programa/db)
2. Ejecute el siguiente comando (node server.js)
3. Ejecute esto solo si es el ditribuidor del api, en otra terminal (ngrok http 3000)
4. Copie en el archivo .env en la carpeta de frontend la direccion que le retorna el sistema **Forwarding**.



5. Con esto la conexión a la base de datos esta establecida.

ServidorSocket

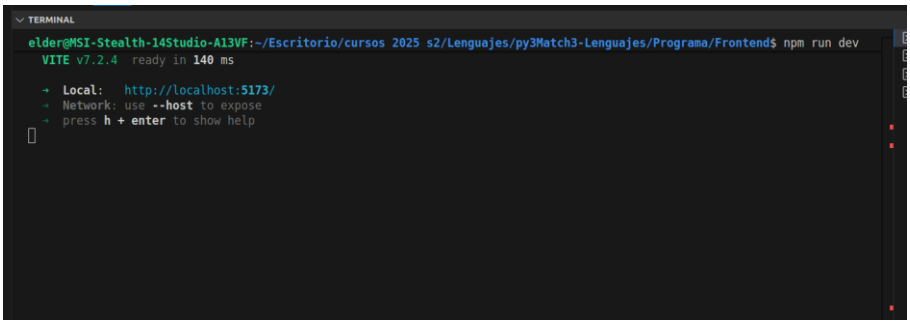
1. Ingrese a la siguiente dirección en una terminal:
   (cd Programa/Server)
2. Instale las dependencias necesarias (solo la primera vez):
   npm install socket.io
   npm install ngrok
3. Ejecute el servidor principal:

   node server.js

4. En otra terminal, levante el túnel de acceso remoto con ngrok:
   ngrok http 3000

Interfaz de juego

1. Ingrese a la siguiente ruta en otra terminal, (cd Programa/Frontend).
2. Ingrese en la misma terminal npm run dev
3. Presione ctrl + click derecho para abrir el componente de interfaz sobre el link que se muestra.

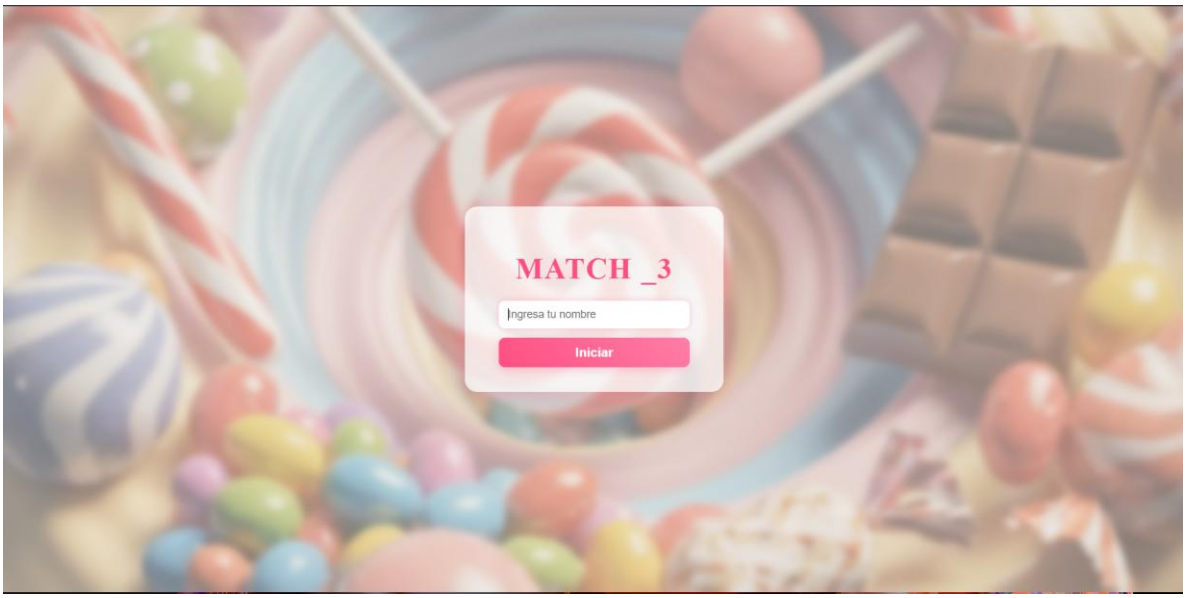


Con los pasos anteriores puede ir a las instrucciones de juego.

Pruebas de funcionalidad

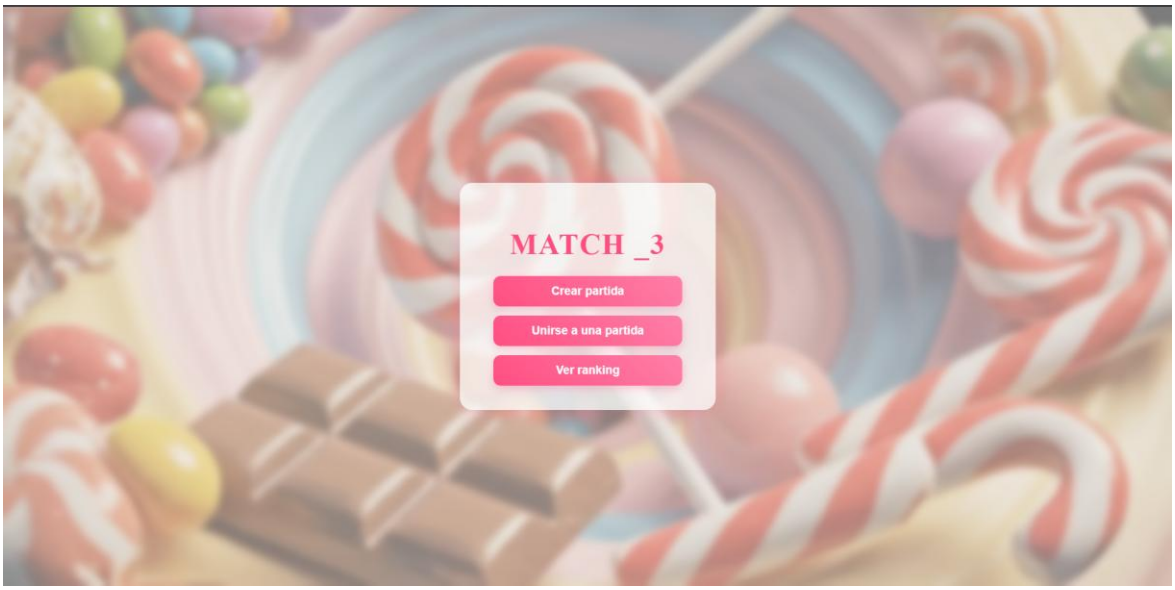
Login

Una ves que el programa se inicialice se le mostrar una especie del login donde solo tendrá que escribir el nombre que desea usar para jugar.



Menú

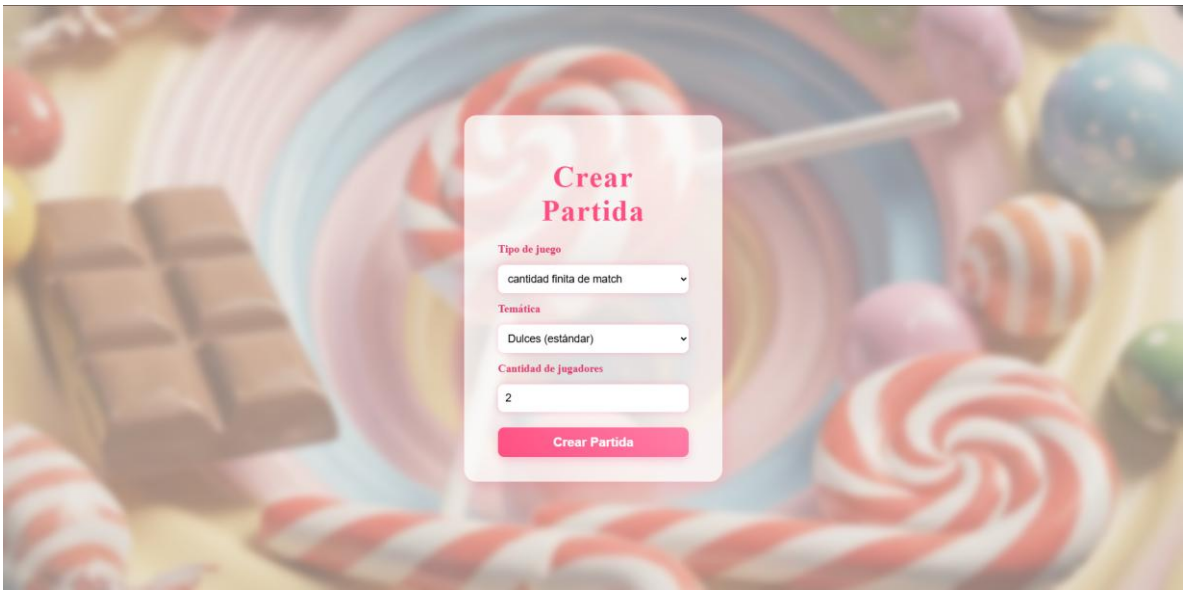
Al escribir el nombre se mostrará un menú con las opciones que solicita el juego.



Crear Partida.

menú de creación de partida

Es un formulario que solicita el tipo de juego que desea realizar con las configuraciones.

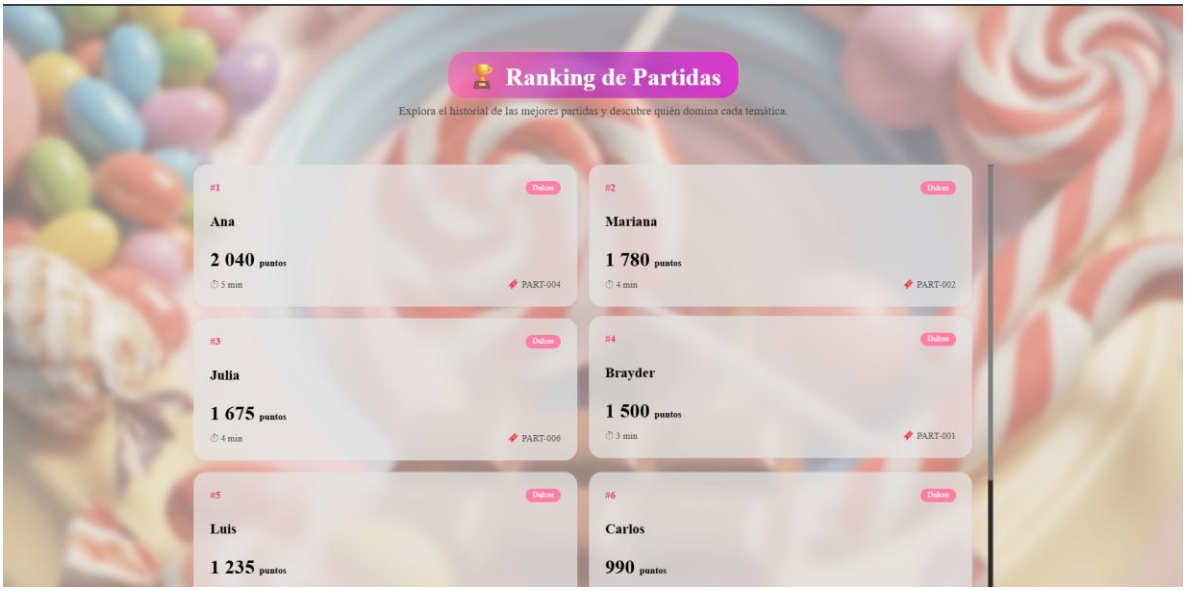


Juego

Conectar a partida ya existente.

Ver Rankin.

Se mostrará un interfaz con todos lo historiales de partidas que se han jugado, mostrando el mejor jugador y la cantidad de puntos además mostrara la configuración de la partida.



Descripción del problema.

El problema de este proyecto es crear una plataforma que permita jugar un Match-3 (Parecido a Candy Crush) multijugador, con una interfaz moderna y que además guarde los datos de las partidas para generar un ranking. En este proyecto se solicita modelar una estructura que aplique la programación orientada objetos para la organización lógica del juego (tablero, celdas, jugadores y partidas).

Debido a todo esto se necesita la integración de una forma de sincronizar el juego entre múltiples jugadores y buscar la forma de almacena los datos de las partidas jugadas y mantener un código mantenible.

Diseño del programa

Enlace para visualizar mejor los diagramas: https://lucid.app/lucidchart/258ef8fe-ac7d-48b2-8727-2353157ba5ba/edit?invitationId=inv_8bc81423-5335-4cfa-8641-2b9b5672c594

Diagrama UML

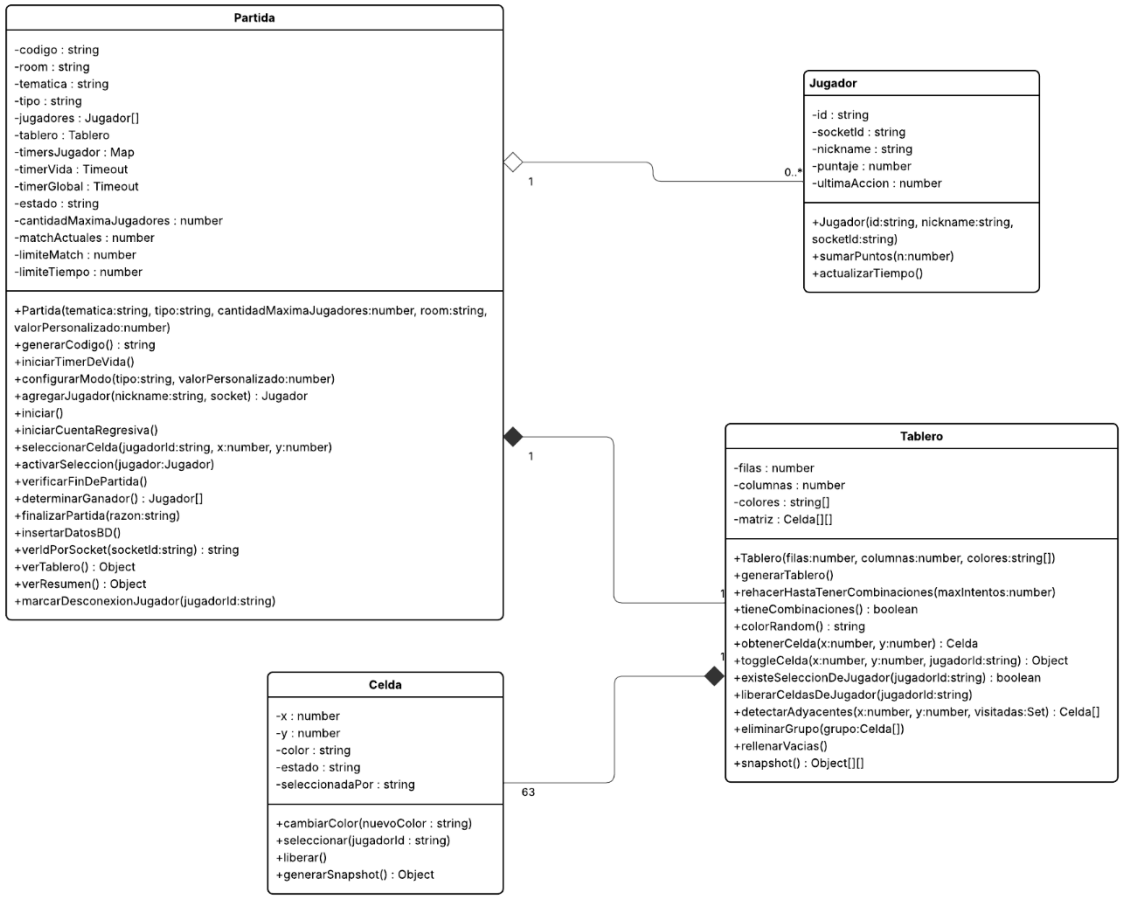


Diagrama de distribución

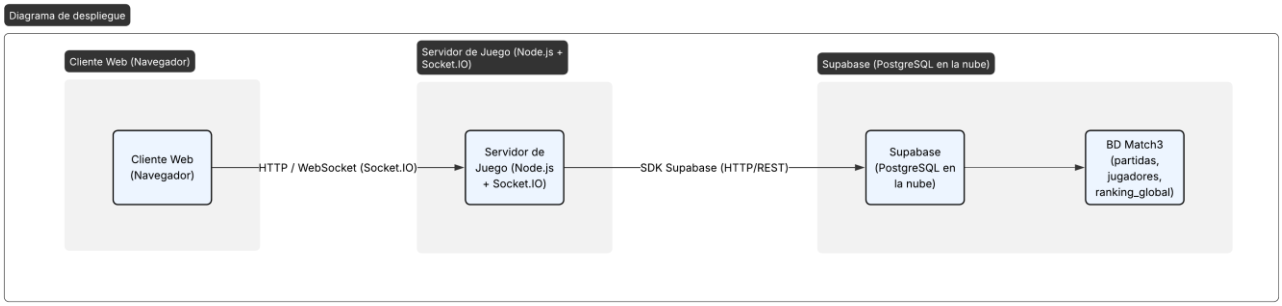
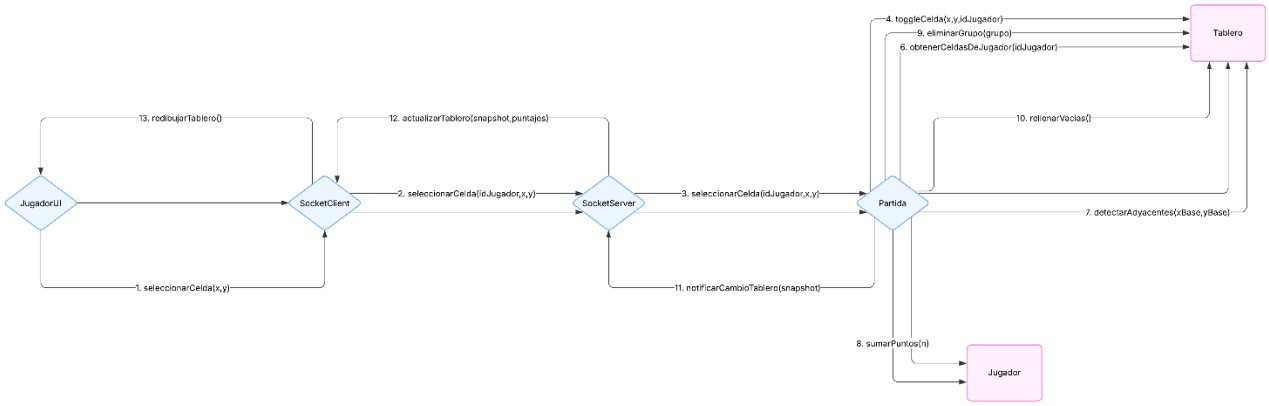


Diagrama de comunicación



Librerías usadas

Librerías usadas

Frontend

- React y React-DOM: para construer y rendierizar componentes en interfaz.
- React-Router-DOM: para manejo de rutas en la interfaz.
- TypeScript: tipado estático para mayor seguridad.

Servidor en tiempo real (Socket.IO + Express)

- Socket.IO: comunicación bidireccional para movimientos, selección de celdas y sincronización del tablero.
- Express: creación del servidor base para WebSockets.

REST API + Base de Datos (Express + Supabase)

- Express: endpoints REST.
- Dotenv: carga de claves y configuración segura.
- @supabase/supabase-js: conexión y consultas a PostgreSQL.

Comunicación del sistema

- **Frontend** conectado **Socket.Server:** WebSockets para actualizaciones instantáneas (match, selección de celda, jugadores entrando/saliendo).
- **Socket.Server** conectado **Backend lógico:** instancia y controla objetos Partida, Tablero, Jugador y Celda.
- **Backend** conectado **Supabase:** inserta partidas, jugadores y ranking.

- **Frontend** conectado **API REST:** obtiene ranking, partidas, jugadores y tablero.

Análisis de resultados

Objetivo	Estado	Descripción
Creación del tablero de juego en interfaz	Logrado	
creación de interfaz rankin.	Logrado	
Implementación de programación orientada a objetos.	Logrado	
Desarrollo de Backend.	Logrado	
Desarrollo de Frontend.	Logrado	
Aplicación de websockets para multi usuarios.	Logrado	
configuración de Tablero.	No Logrado	
Incorporar autenticación por nickname.	Logrado	
Persistir datos en base de datos (SupaBase).	Logrado	
Publicar servidor y cliente con Ngrok	Logrado	
Verificar movimientos validos	Logrado	

Descripción manual de los principales algoritmos.

Detección de adyacentes

Este algoritmo identifica todas las celdas que están conectadas entre sí y que tienen el **mismo color**.
Es fundamental para saber si el jugador logró formar un **grupo válido de 3 o más celdas**, lo que produce un “match”

Toma la celda y revisa los vecinos para confirmar si la posición cuenta con el match.

Detección de combinaciones

Antes de comenzar a jugar, el tablero debe asegurar que exista **al menos una combinación posible**.

- Recorre **todas** las celdas del tablero.

- Para cada celda, ejecuta el algoritmo anterior (detección de adyacentes).
- Si algún grupo encontrado tiene tamaño **igual o mayor a 3**, entonces el tablero es válido.
- Si no, se considera un tablero sin combinaciones.

Regeneración de tablero

Si el tablero generado aleatoriamente **no tiene combinaciones**, se necesita crear uno nuevo que sí sea jugable.

- Genera un tablero aleatorio.
- Usa el algoritmo de verificación de combinaciones.
- Si no encuentra ninguna:
    - Genera un nuevo tablero desde cero.
    - Repite esto hasta que algún tablero tenga combinaciones o se alcance un límite de intentos (para evitar bucles infinitos).

Activación de selección

Este algoritmo decide si la selección hecha por el jugador **forma un grupo válido**, y en ese caso:

- Elimina las celdas del tablero.
- Suma puntos al jugador.
- Rellena los espacios vacíos.
- Actualiza el estado de la partida.

Bitácora

https://github.com/Elder19/py3Match3-Lenguajes