

# PROGRAMACIÓN III



MANUAL TECNICO

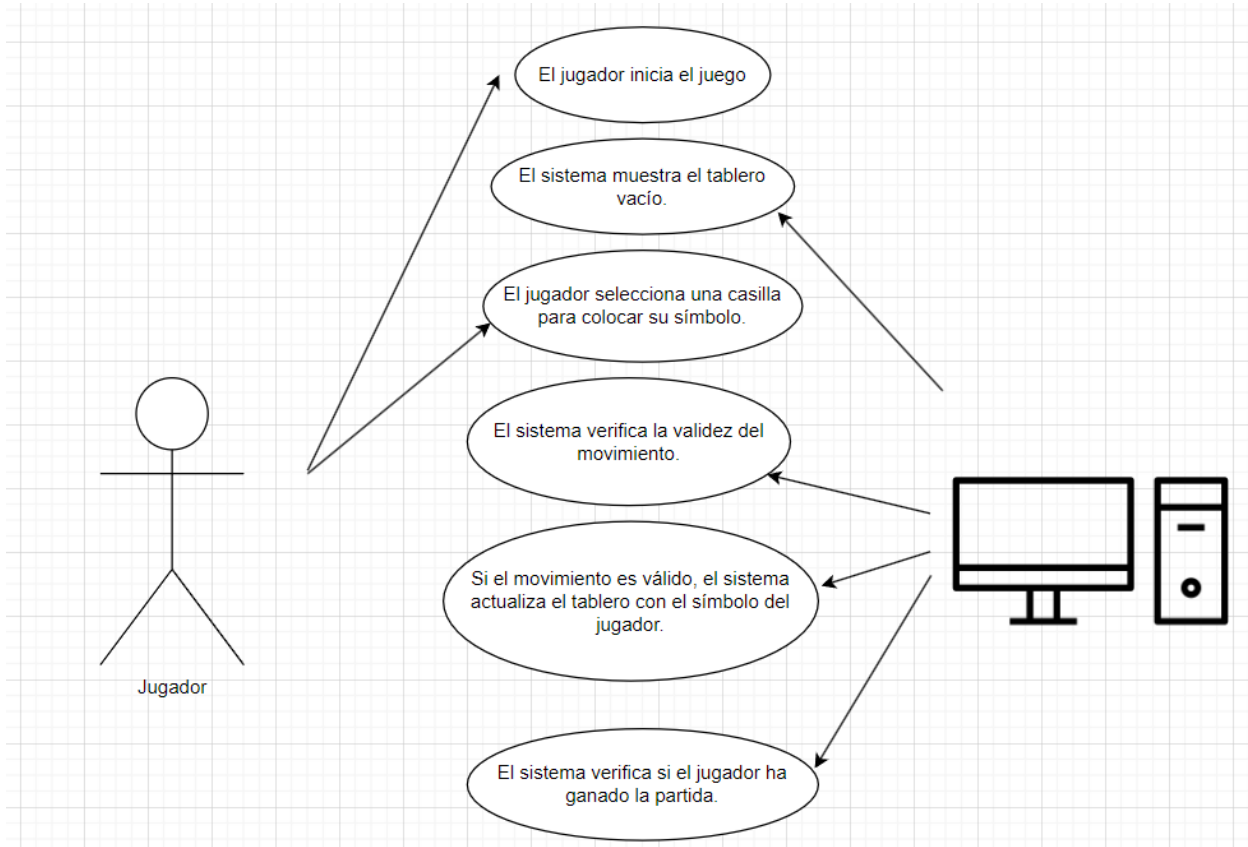


## Casos de Uso

### Jugar Partida

Actores: Jugador, computadora

Descripción: Este caso de uso describe la secuencia de acciones que realiza el jugador para jugar una partida del juego Tik Tak Toe.



## REPOSITORIO

Un repositorio en GitHub es un espacio digital de almacenamiento donde se pueden gestionar y almacenar archivos y proyectos de software utilizando el sistema de control de versiones Git. Esta plataforma en línea permite a los desarrolladores colaborar en proyectos de software de manera remota.

En un repositorio de GitHub, los archivos y carpetas del proyecto se organizan de manera estructurada y se almacenan en la nube. Esto facilita el trabajo en equipo, ya que múltiples personas pueden acceder y contribuir al proyecto desde cualquier lugar con conexión a Internet.

Además de almacenar archivos, un repositorio de GitHub ofrece herramientas para gestionar el desarrollo del software, como seguimiento de problemas, solicitudes de extracción, revisiones de código y funciones de colaboración en equipo. Estas características hacen que GitHub sea una herramienta invaluable para el desarrollo de software colaborativo y la gestión de proyectos.

Nuestro repositorio de proyecto se encuentra ubicado en el siguiente enlace:

backend

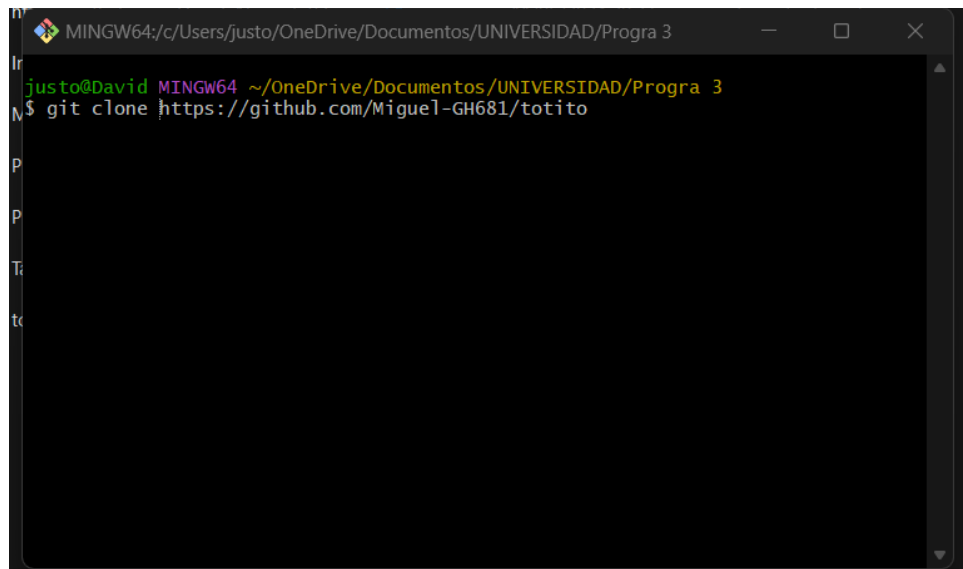
<https://github.com/Miguel-GH681/totito>

frontend

<https://github.com/Miguel-GH681/totito-web.git>

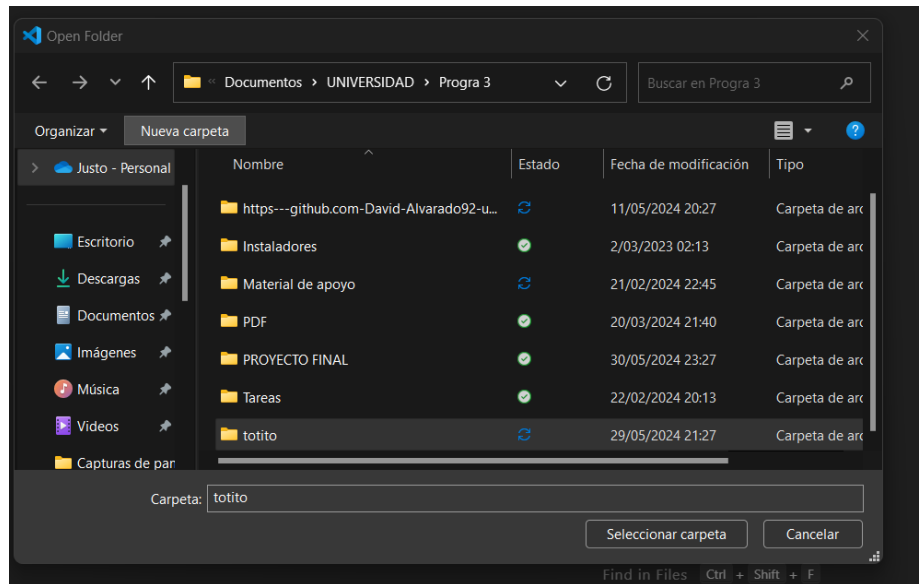
## Pasos para ejecutar el juego (Backend)

1. Como primer paso debemos de clonar nuestro repositorio de GITHUB, para ello será necesario ubicar en que carpeta alojaremos una copia del código fuente



```
mingw64 ~/OneDrive/Documentos/UNIVERSIDAD/Progra 3
justo@David MINGW64 ~/OneDrive/Documentos/UNIVERSIDAD/Progra 3
$ git clone https://github.com/Miguel-GH681/totito
```

2. Al tener clonado el repositorio debemos de abrir el repositorio en VISUAL STUDIO para ello demos de ir a archivos -> abrir carpeta



3. Una vez abierto el código se deberá de instalar los siguientes framework's:
  - a. **Flask:** Flask es un framework web ligero y flexible que se utiliza para construir aplicaciones web en Python. Permite crear rápidamente aplicaciones web simples o complejas mediante la definición de rutas, vistas y controladores.
  - b. **Cloudinary:** Cloudinary es un servicio en la nube que proporciona una plataforma integral para almacenar, gestionar y entregar imágenes y videos en aplicaciones web. El paquete cloudinary de Python proporciona una API cliente para interactuar con los servicios de Cloudinary desde una aplicación Python.
  - c. **Flask-CORS:** CORS (Cross-Origin Resource Sharing) es un mecanismo que permite a los servidores indicar a los navegadores si se permite que una solicitud se haga desde un origen diferente al del sitio web actual. Flask-CORS es una extensión de Flask que simplifica la configuración de CORS en aplicaciones Flask, lo que permite el intercambio de recursos entre diferentes dominios de forma segura.
  - d. **python-dotenv:** python-dotenv es una biblioteca que permite cargar variables de entorno desde un archivo .env en proyectos Python. Esto es útil para configurar la aplicación con valores sensibles o personalizados, como claves de API o credenciales de bases de datos, sin tener que exponerlos directamente en el código fuente.
4. Luego de tener instalado los framework's procedemos a ejecutar el programa seleccionando el archivo api.py

```

9  from flask import Flask, request, jsonify
10 from flask_cors import CORS, cross_origin
11 from tree_controller import tree_controller
12
13 app = Flask(__name__)
14 cors = CORS(app)
15 app.config['CORS_HEADERS'] = 'Content-Type'
16
17 tree_controller = tree_controller()
18
19 class TotitoApi:
20
21     @app.route('/add_movement', methods=['POST'])
22     @cross_origin()
23     def add_movement():
24         tree_controller.init_game()
25         tree_controller.getGraph("tree.dot")
26
27         data = request.json
28         cell = int(data.get('cell'))
29         cpu_first_player = data.get('cpu_first_player')
30
31         results = tree_controller.get_cpu_movement(cell, cpu_first_player)
32         tree_controller.getGraph("tree.dot")
33         # url = tree_controller.upload_images("tree.dot.png")
34
35         return jsonify([
36             'iswinner': results[0],
37             'cpu_movement': results[1],
38             'tree_url': ""
39         ])
40
41     @app.route('/reset_game', methods=['GET'])
42     @cross_origin()
43     def reset_game():
44         tree_controller.clear_data()
45         tree_controller.getGraph("tree.dot")

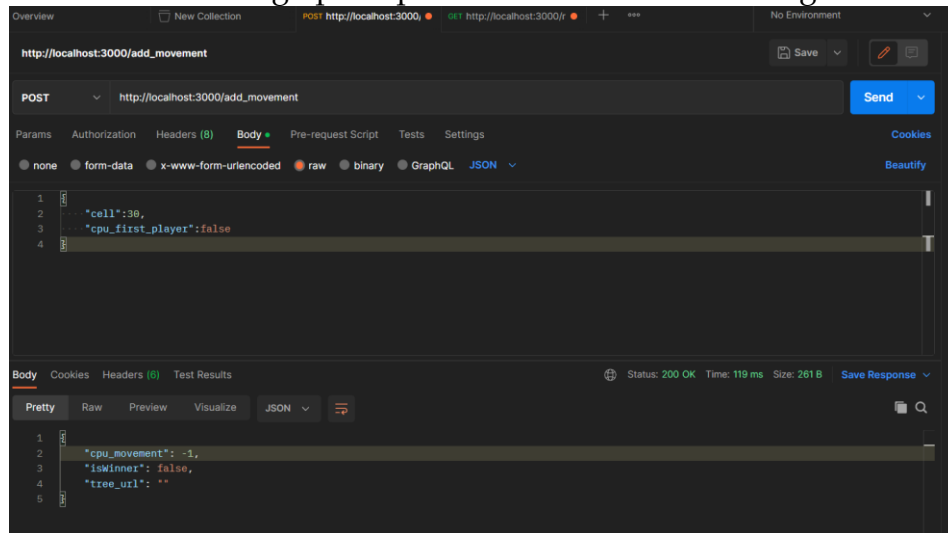
```

5. Al tener nuestro entorno levantado podemos ejecutar los valores del totito
  - a. Para ingresar valores en la tabla debemos de ejecutar el siguiente API:

[http://localhost:3000/add\\_movement](http://localhost:3000/add_movement)

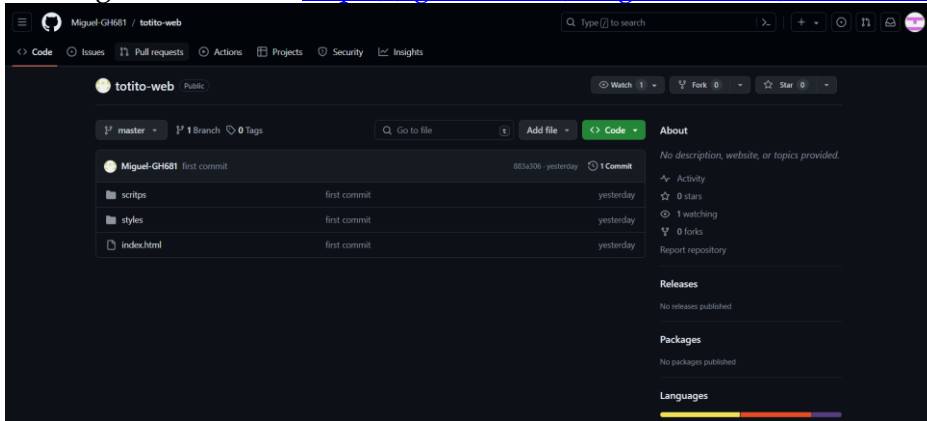
En la sección de Body debemos de ingresar el valor del tablero que estaremos

ingresando como valor del luego para que automáticamente el código nos

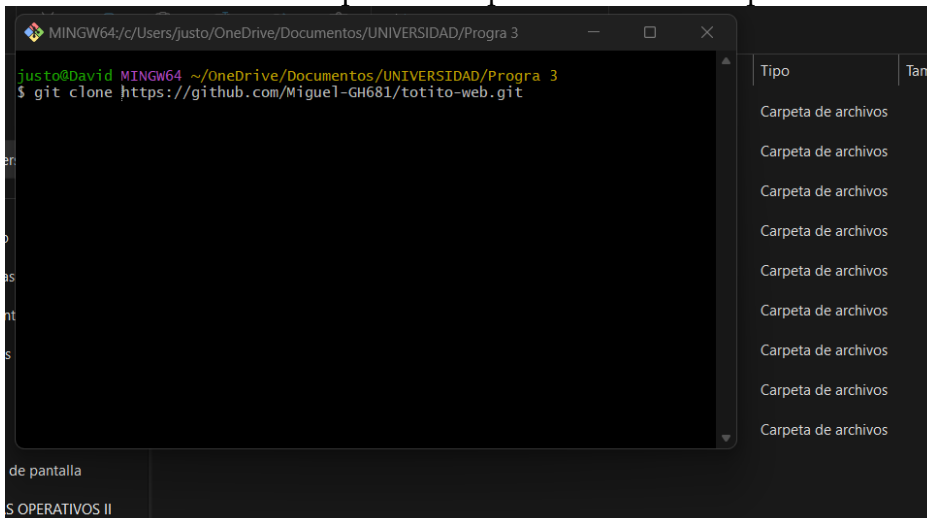


## Pasos para levantar FRONT END

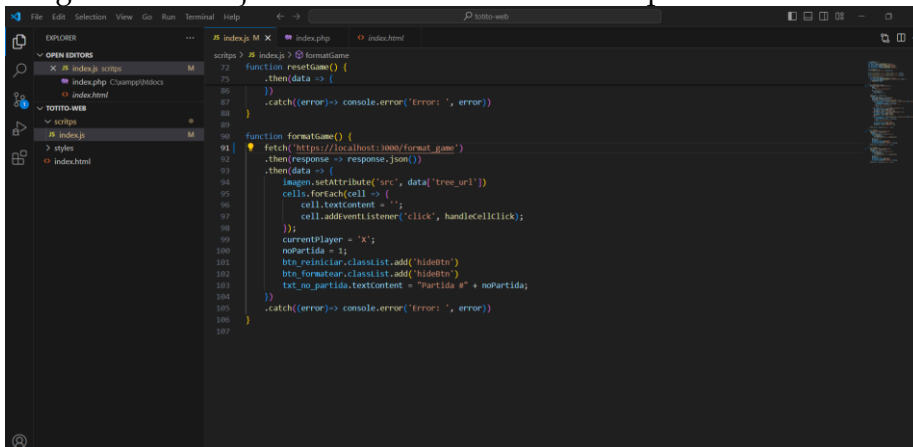
1. Como primera parte debemos de clonar el repositorio de front end este lo podremos realizar en el siguiente enlace: <https://github.com/Miguel-GH681/totito-web.git>



2. Debemos de ubicar la carpeta en la que se contrará el repositorio clonado:

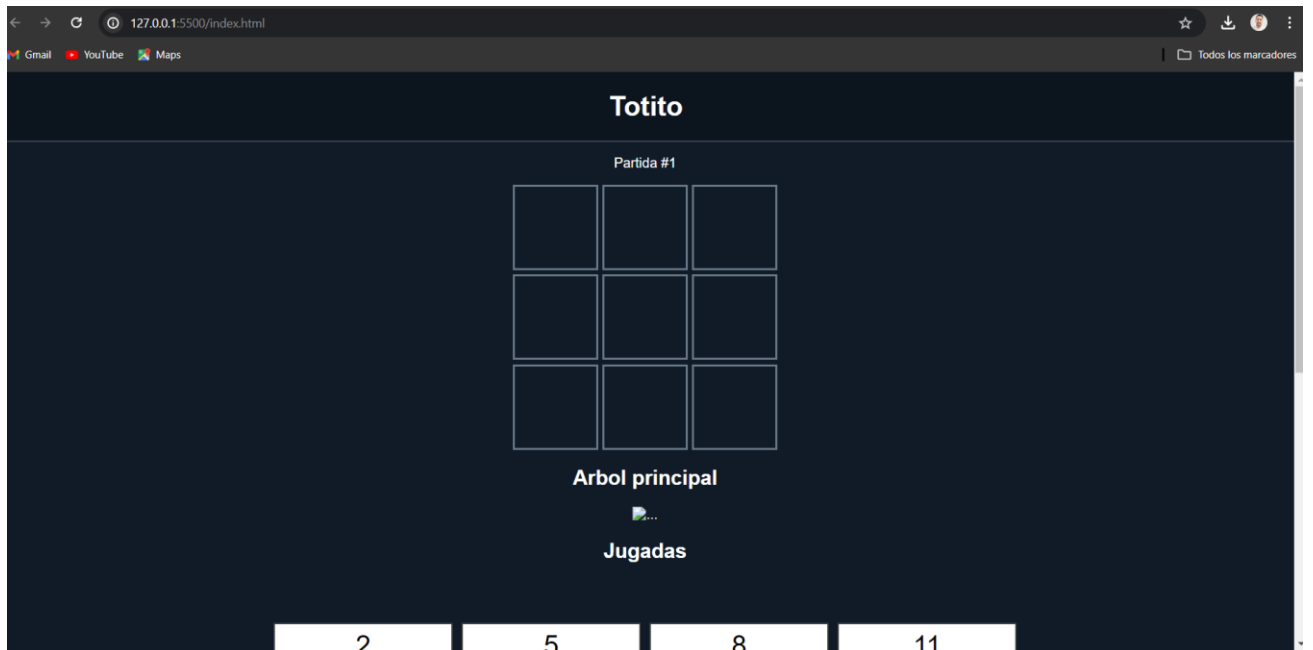
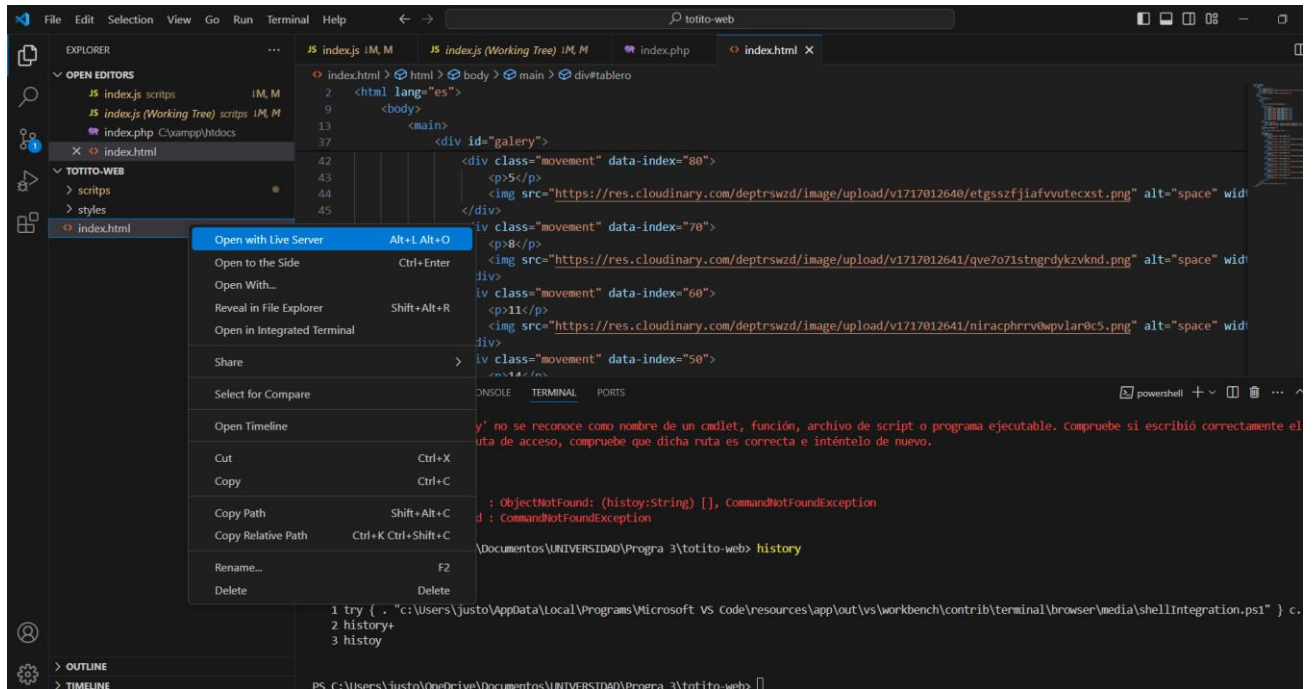


3. Luego se deberá ejecutar visual studio con la carpeta clonada del front end



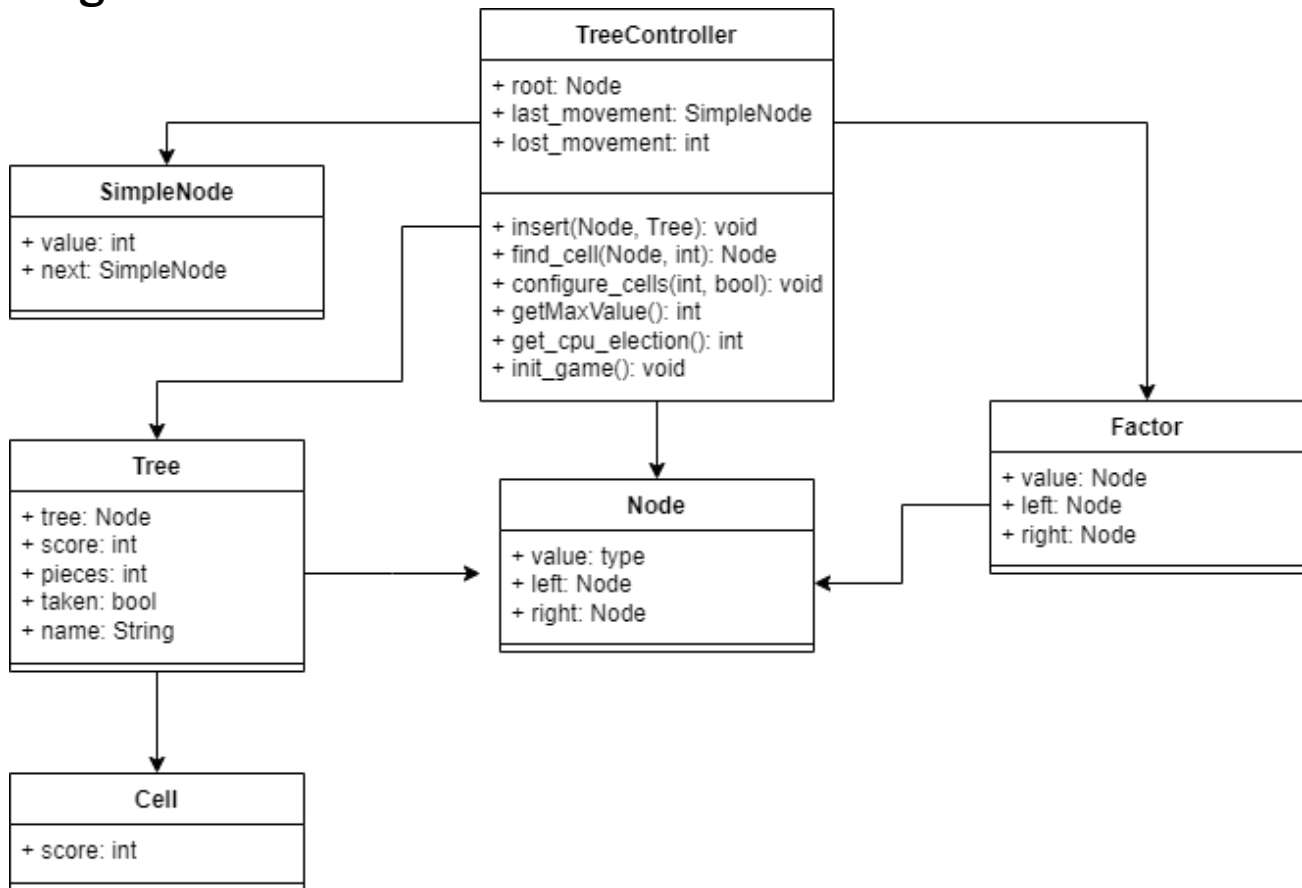
4. Será necesario levantar el FRONT END de la siguiente manera:







## Diagrama de clases



## Documentación en Gitbook

<https://app.gitbook.com/invite/VVq29OvMXPZ9RUICWcHD/MZmoeEaHe4EpYE8tMdNk>