**Universidad de San Carlos de Guatemala**

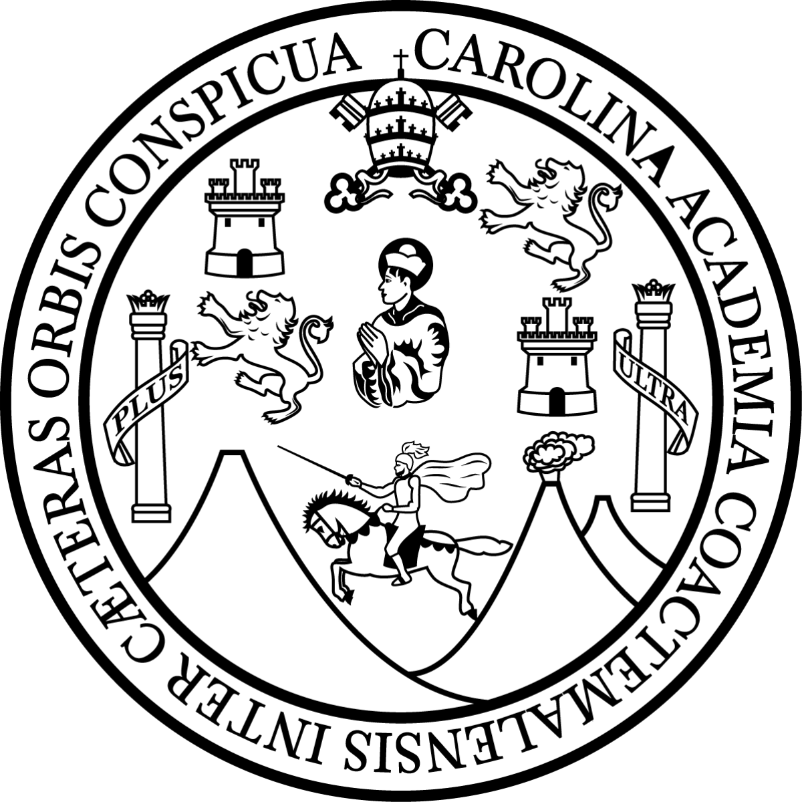
Facultad de Ingeniería

Curso

Sección:

Ing.

Tutor Académico:



Proyecto 1

TourneyJS: Analizador Léxico para

Gestión de Torneos Deportivos

Nombre: Diego Alberto Maldonado Gálvez

Carné: 202200092

Fecha:21/09/2025

# Introducción:

Este proyecto implementa un analizador lexico y sintactico en **JavaScript puro**, sin usar librerias de terceros.  
El sistema recibe como entrada un archivo de texto que describe un torneo con las secciones **TORNEO, EQUIPOS y ELIMINACION**.  
El programa se encarga de:

* Identificar tokens y clasificar su tipo.
* Detectar errores lexicos y de formato.
* Construir un modelo interno con la informacion de equipos, jugadores, partidos y resultados.
* Generar reportes en HTML y un archivo DOT de Graphviz.

Este manual explica la estructura del codigo, la organizacion en carpetas y los pasos para ejecutar el proyecto desde la linea de comandos.

# Plataforma de Ejecución:

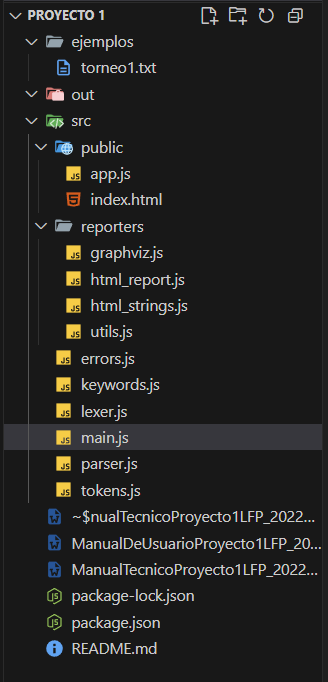
# El desarrollo del proyecto se realizó en Visual Studio Code (VS Code) como entorno de desarrollo principal, utilizando Node.js v22 para la ejecución del programa en consola. Se trabajó con el motor de ejecución de JavaScript que provee Node, lo que permitió usar módulos nativos como fs (para la lectura y escritura de archivos) y readline (para la interacción con el usuario en consola).

# Diagrama de clases

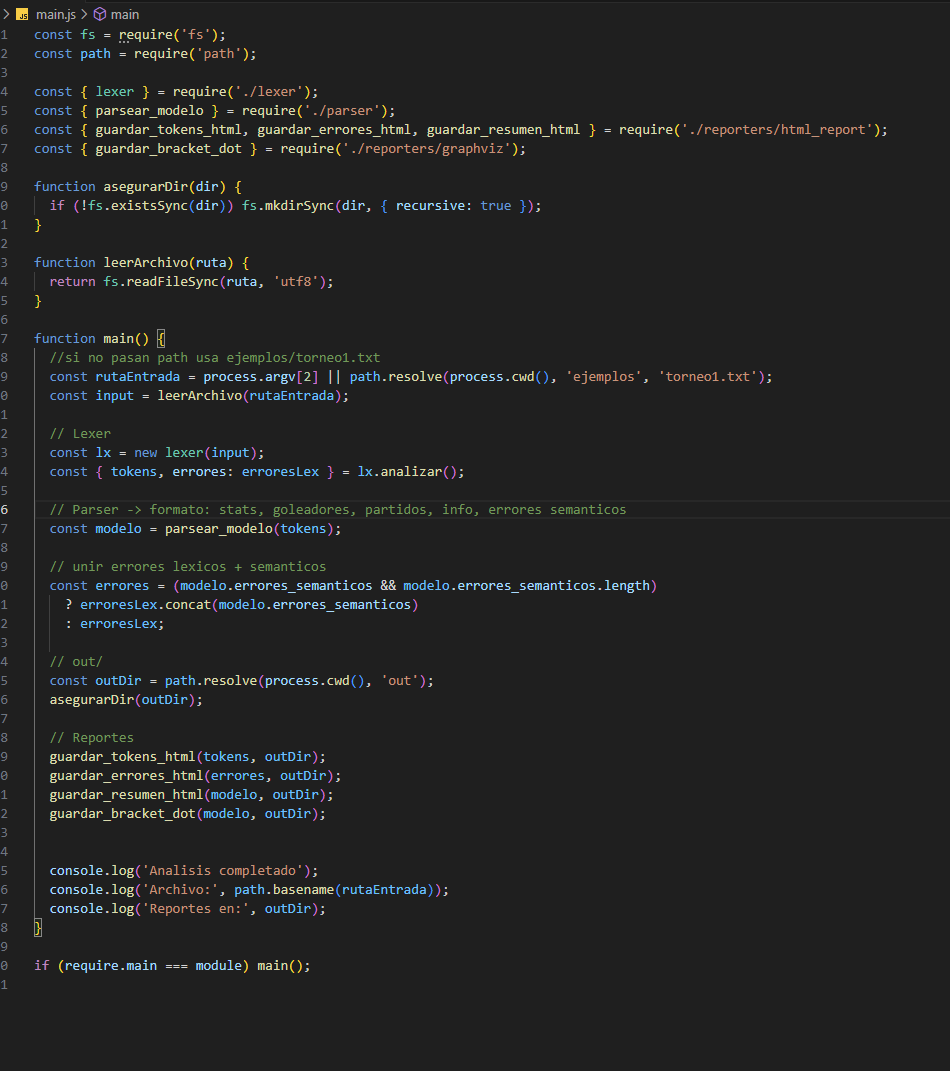
# 

# Como funciona el programa:

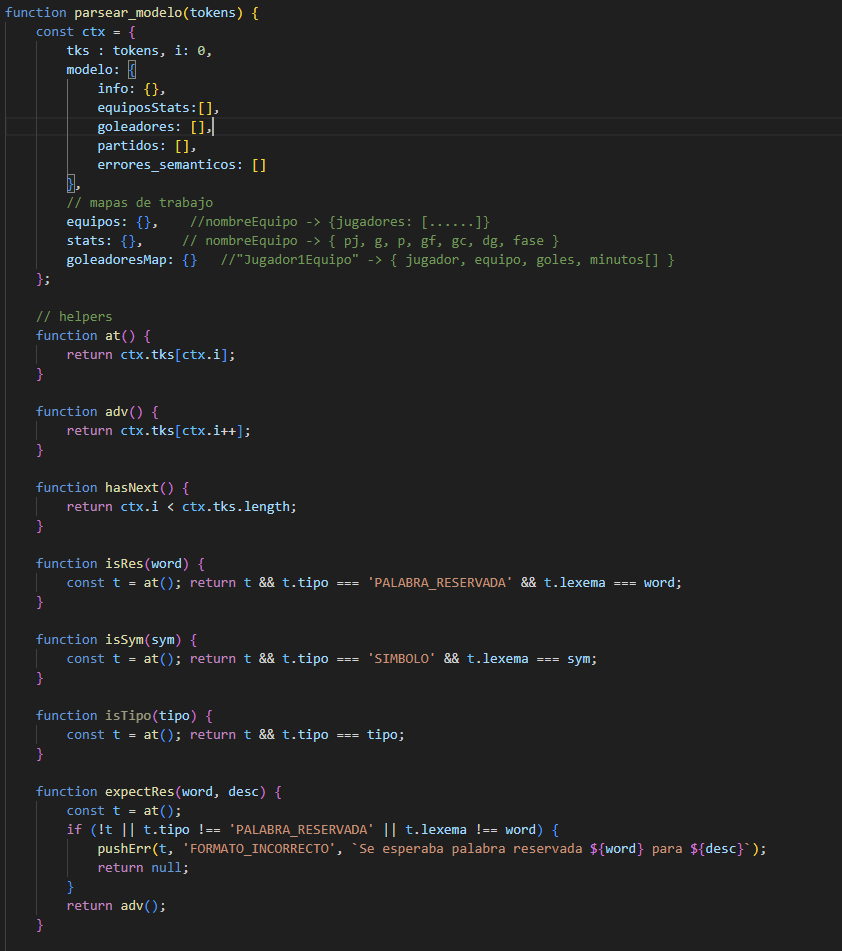
1. Estructura del proyecto:

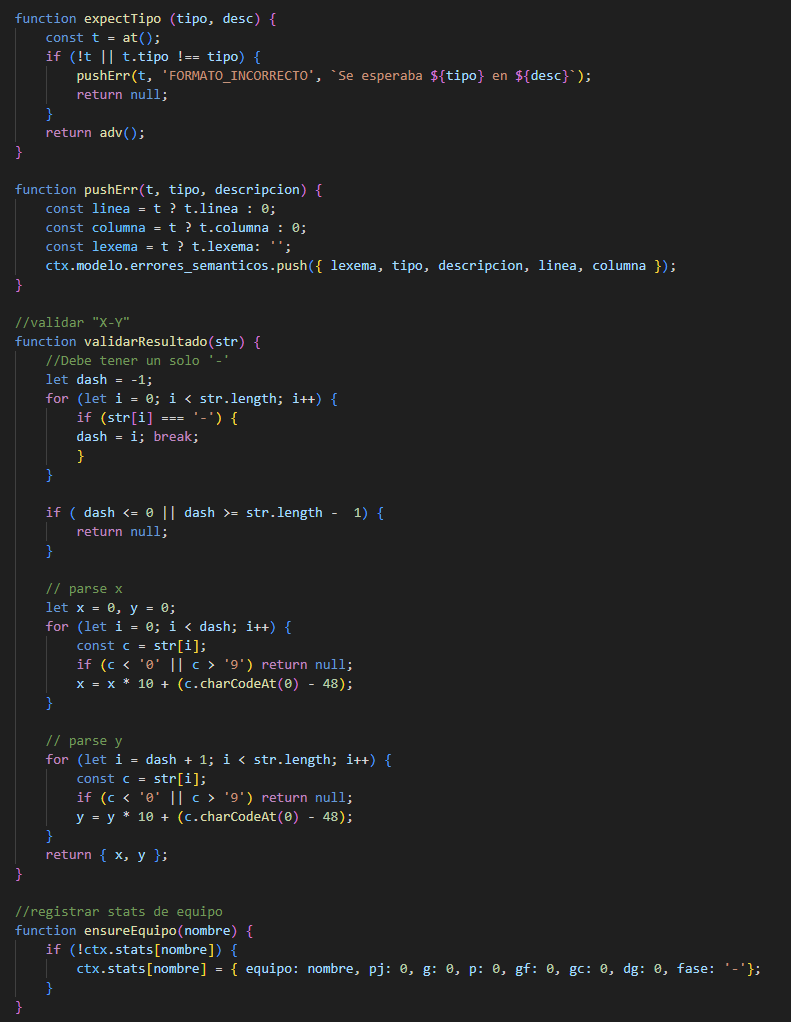


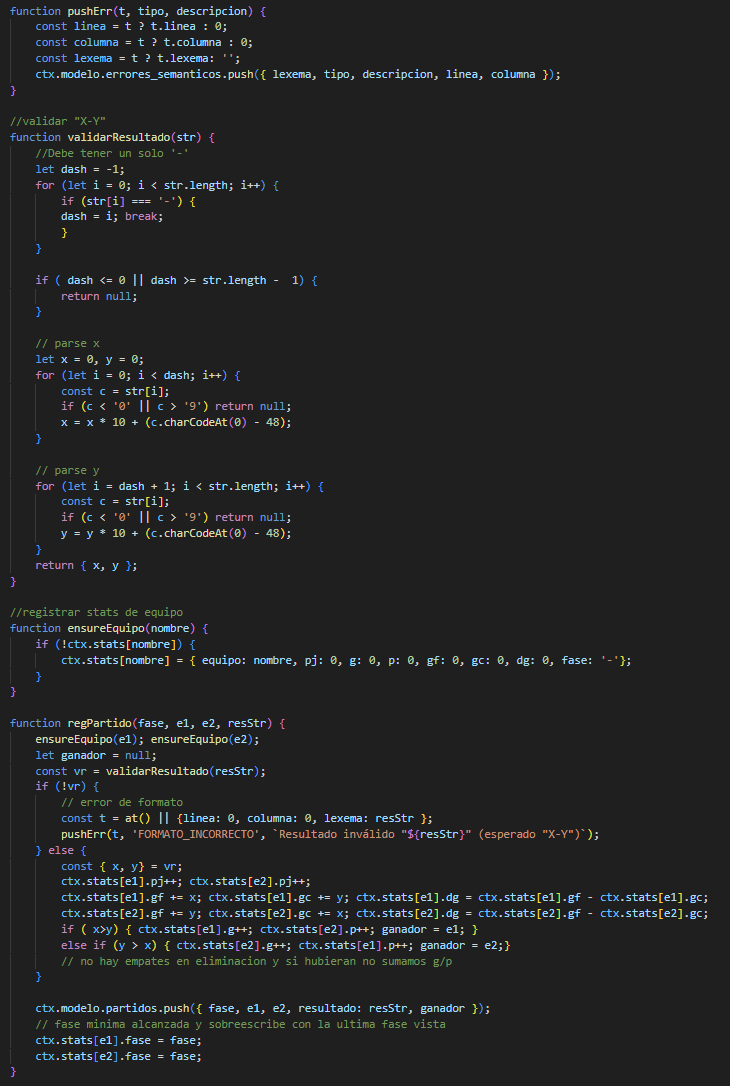
1. Main: Aca mandamos a llamar nuestro parser y leer para ejecutar las funciones principales

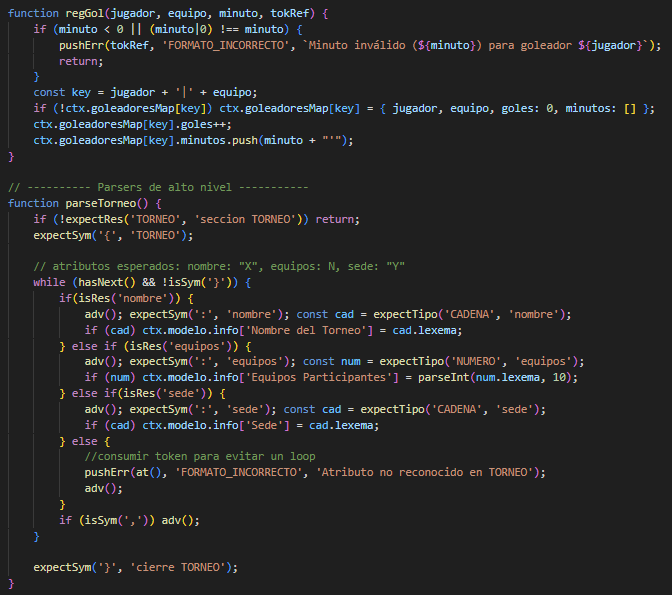


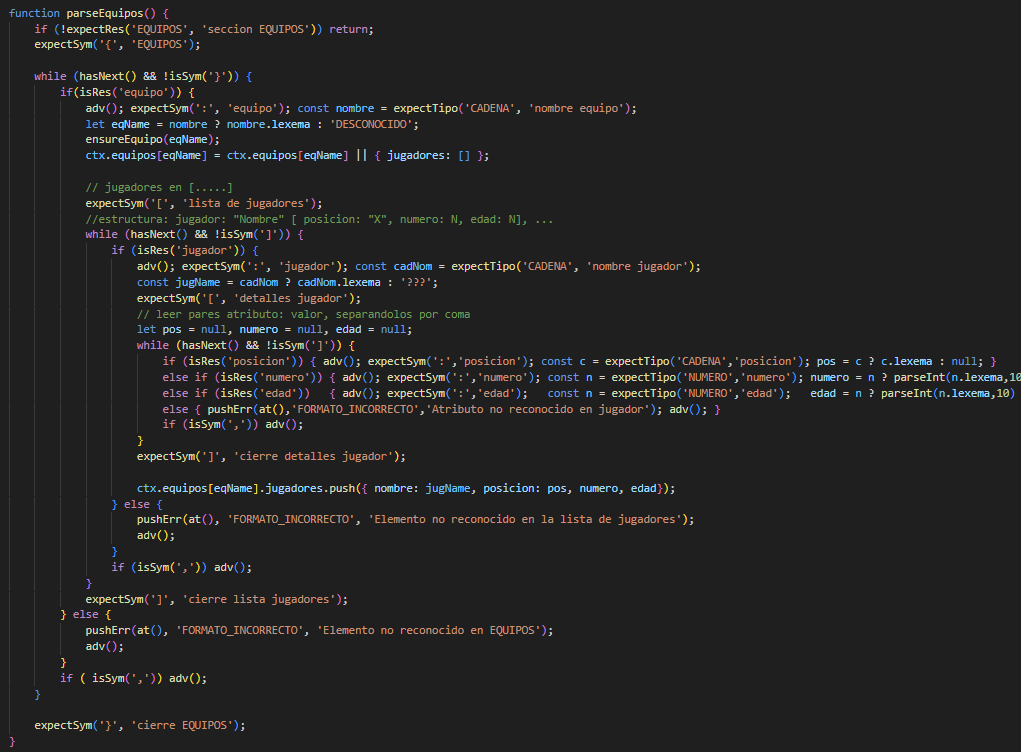
1. Parser: Recibe la lista de tokens ya leidos por mi analizador lexico, recorre los tokens una sola vez sección por sección (TORNEO, EQUIPOS, ELIMINACION) validando el formato y acumulando datos para compactarlo después en modelo



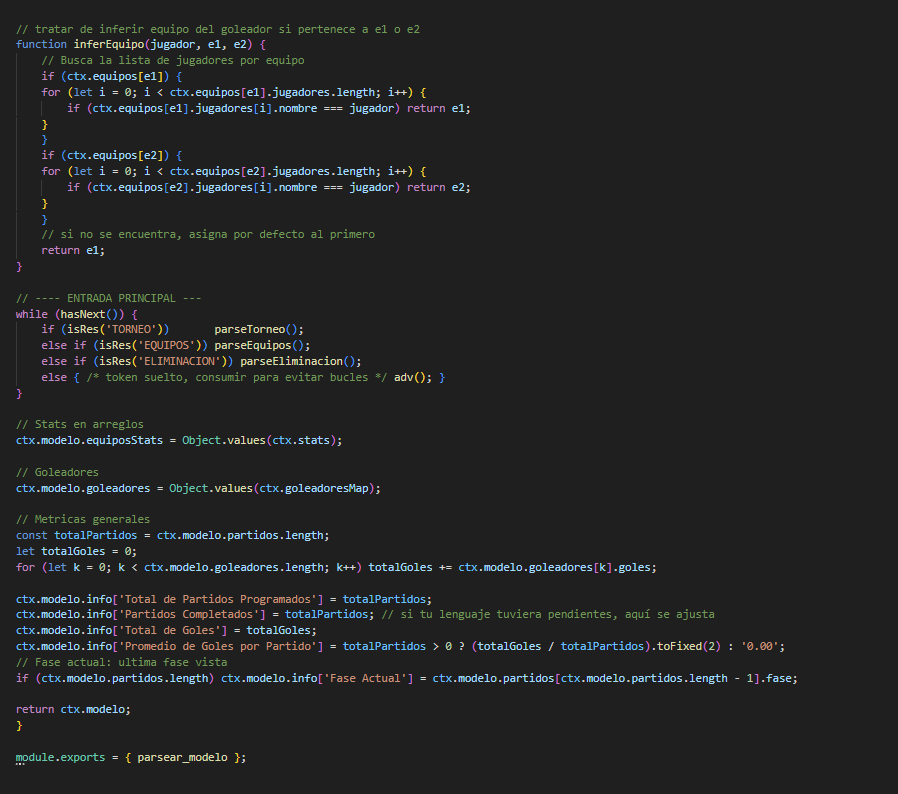




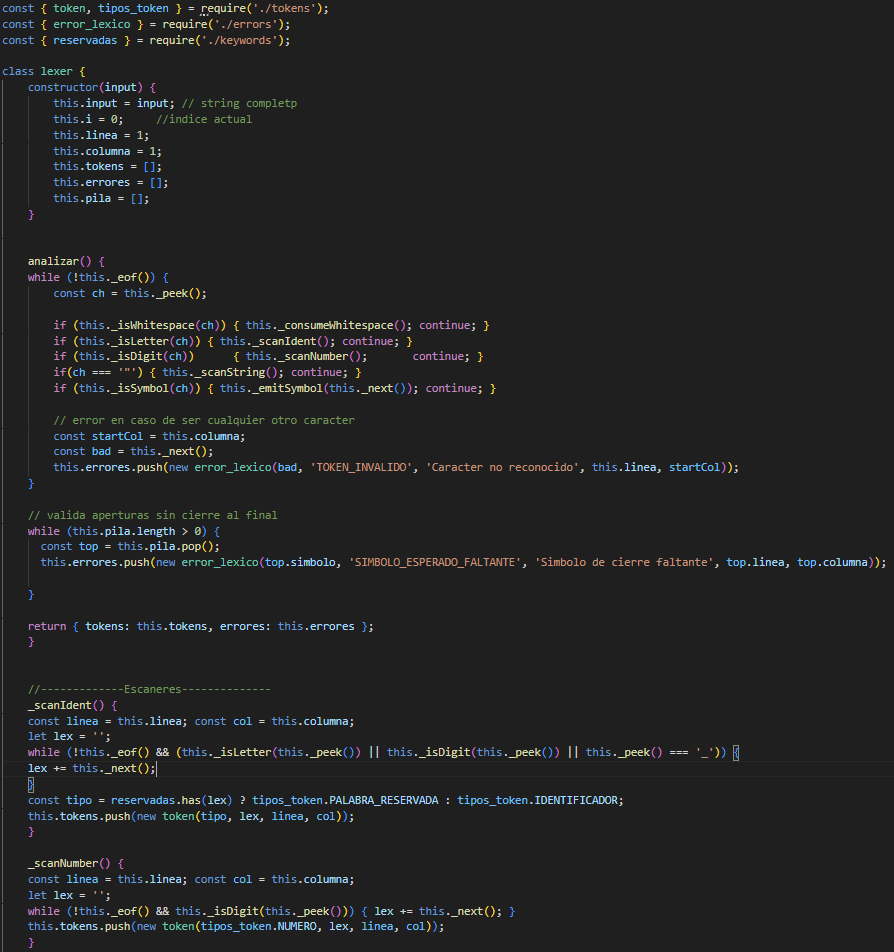




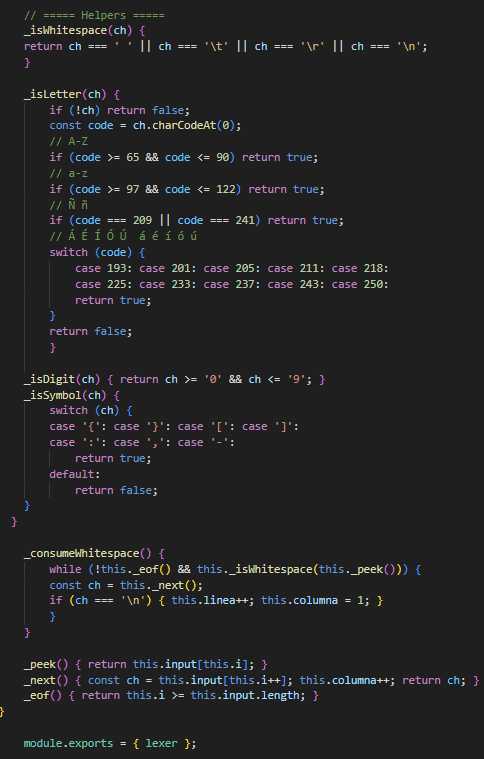




1. Lexer: Recibe como entrada el texto completo del archivo y los clasifica en espacios en blanco, letra, digito, comilla doble, símbolo permitido y cualquier otro como error. Este devuelve tokens y errores



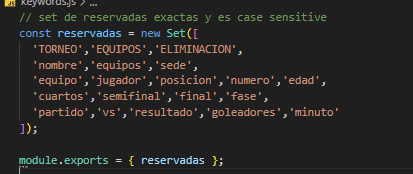




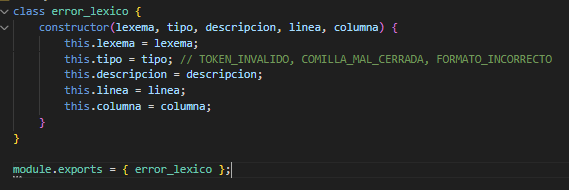
1. Tokens: Es un contenedor de datos con 4 campos



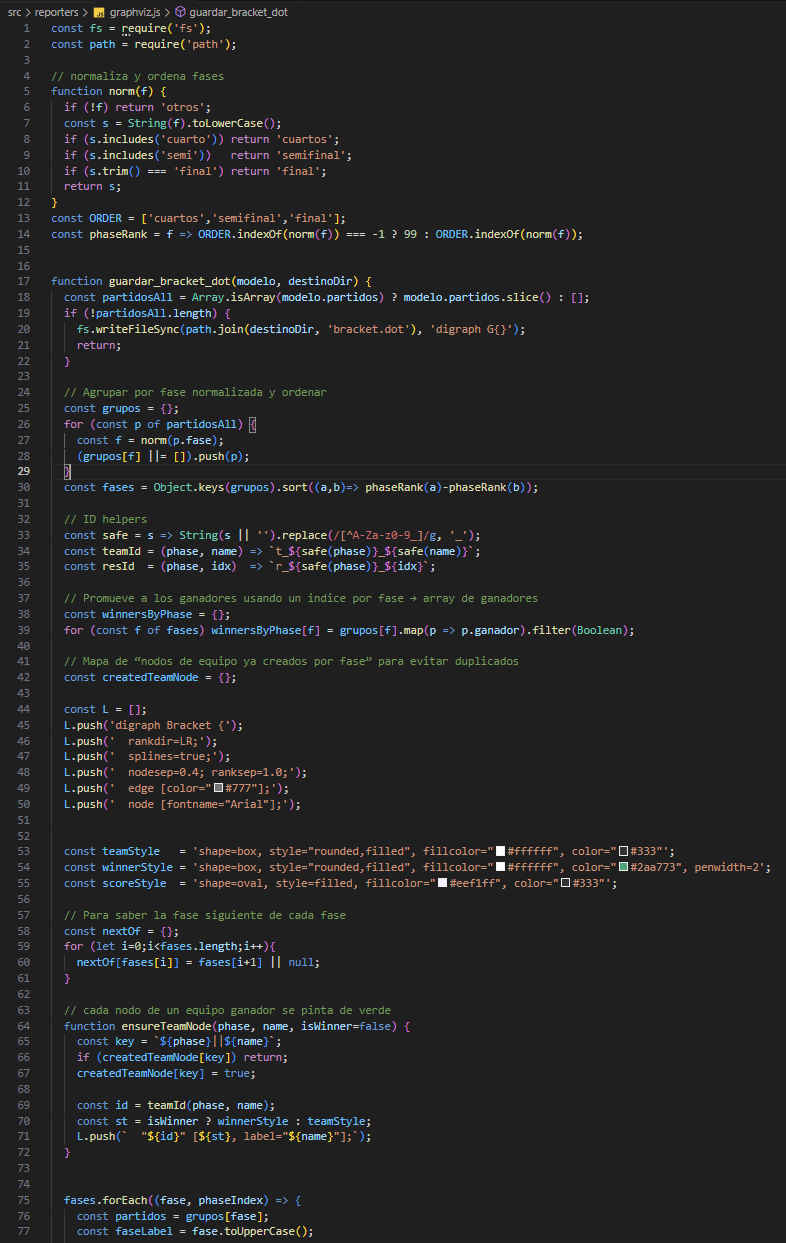
1. Keywords: Es un set que se utiliza en el lexer juntando letras/dígitos para formar lex, si reservadas.has(lex) => token es una PALABRA\_RESERVADA, de lo contrario es un IDENTIFICADOR, deben coincidir exactamente en mayúsculas y minúsculas o se tomara como IDENTIFICADOR

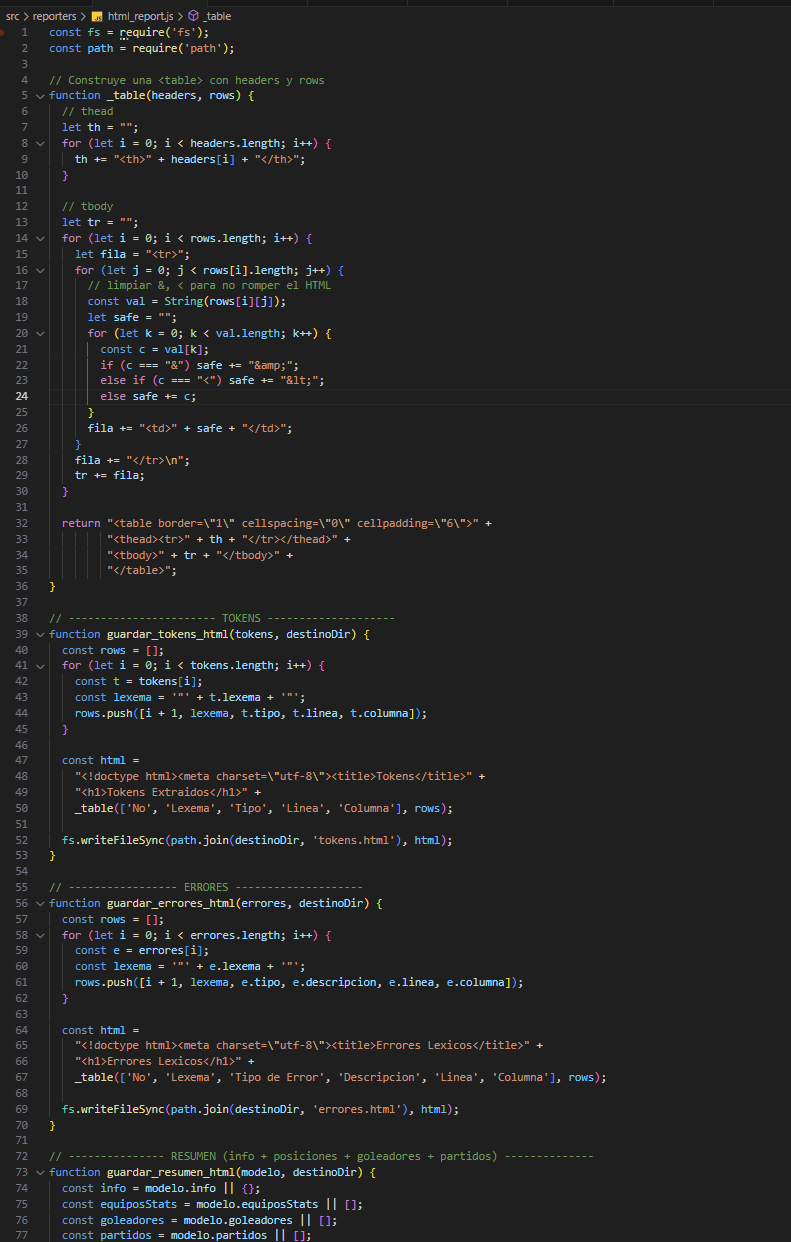


1. Errors: Lo utiliza el leer para registrar cada problema detectado con suficiente información para poder mostrarlo en la tabla HTML

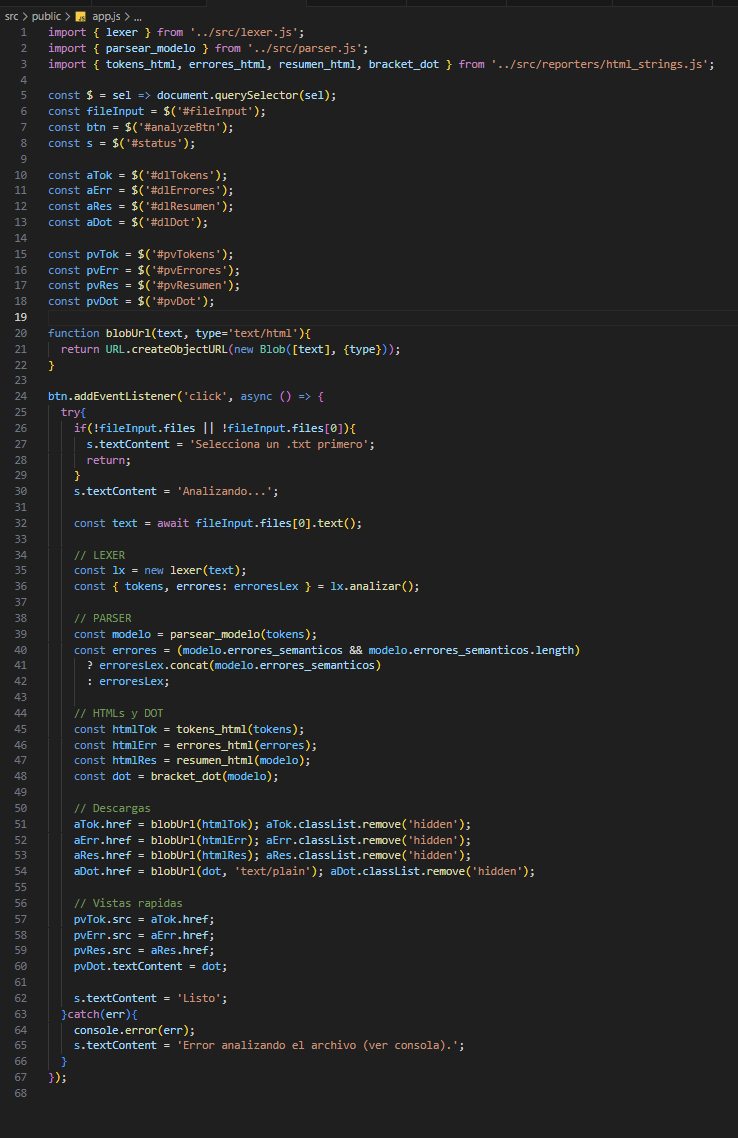


1. Reporters: Generan los html y el bracket con graphviz





1. App



1. Index: Es el HTML que muestra todos mis archivos guardados en la carpeta out

