Manual Tecnico

USAC POKEMON

Lenguajes Formales y de programación
Douglas Xavier Santiago Soto Mejia

IPC1

Douglas Xavier Santiago Soto Mejia

Tabla de contenido

Introducción **4**

Objetivo *4*

Contenido **4**

Requerimientos *4*

Carpetas **5**

Analyzer **5**

Tokens *5*

LexicalAnalyzer *6*

Analyzers **10**

Analyze.controller *10*

Analyze.route *11*

pages **12**

index.ts *12*

Package.json *13*

Menu.ejs *13*

Recomendaciones **21**

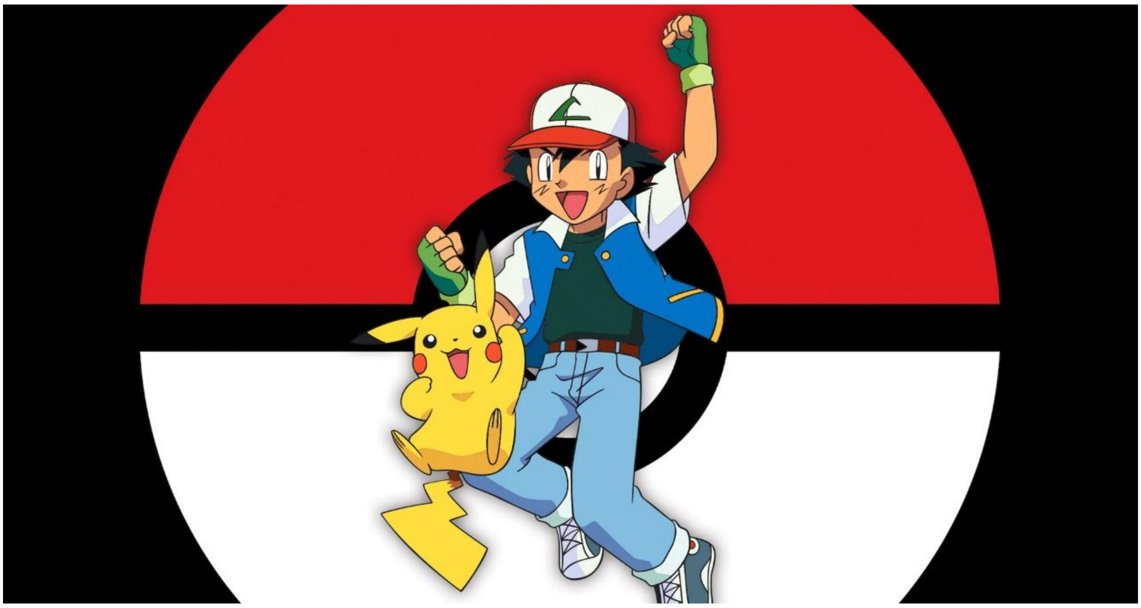
Manual Tecnico

Introducción

A través del lenguaje de programación TypeScript, se realiza un servidor web que simula la obtención de un jugador con sus Pokemones, los cuales pasaran por un análisis léxico previo.

Objetivo

El objetivo de la aplicación es analizar los pokemones de un listado que será dado en su momento, el cual, pasara por un análisis léxico, una vez hecho, y sin encontrar errores, el objetivo es tomar los 6 mejores pokemones de dicha lista.



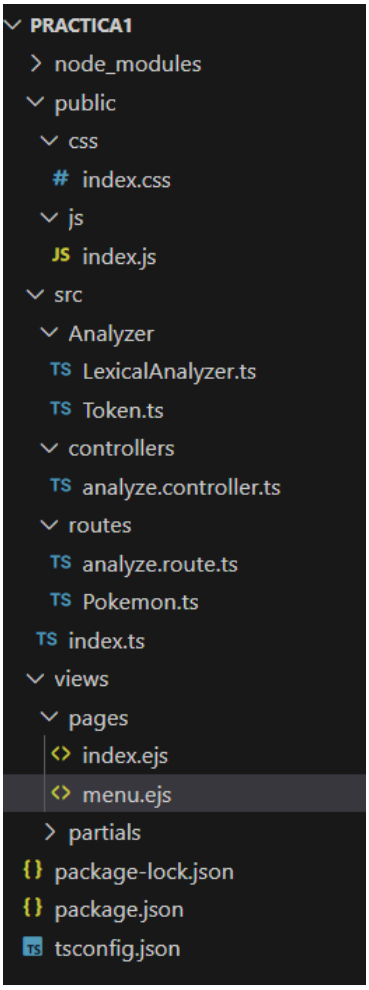
Contenido

Requerimientos

- Computadora Portatil.

- Windows 7 o superior.
- 4 de RAM minimo.
- Netbeans 8.2 o superior.
- Sistema operativo Windows

Carpetas

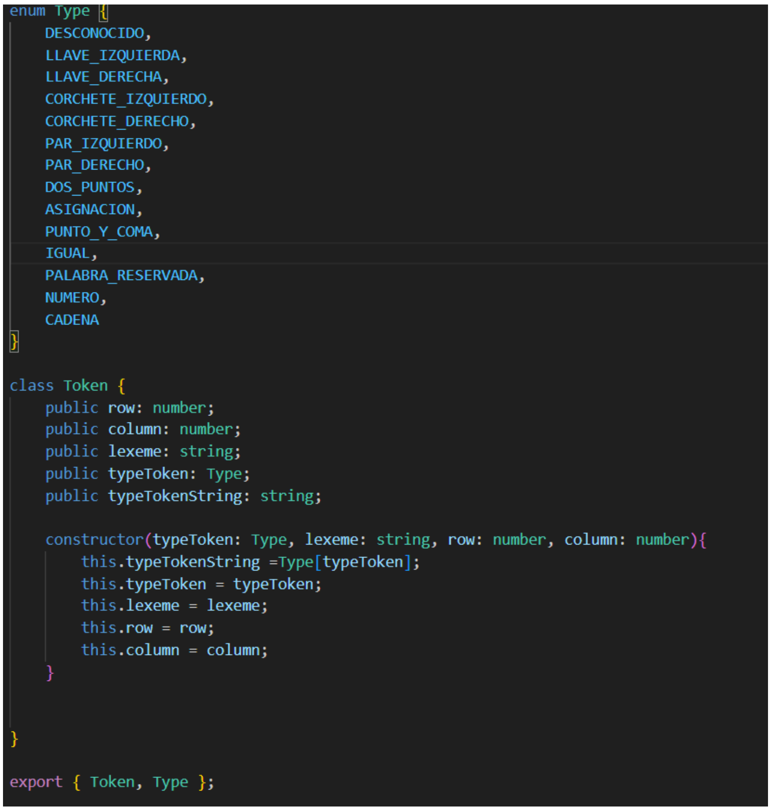


```
PRACTICA1
  node_modules
  public
    css
      index.css
    js
      index.js
  src
    Analyzer
      LexicalAnalyzer.ts
      Token.ts
    controllers
      analyze.controller.ts
    routes
      analyze.route.ts
      Pokemon.ts
    index.ts
  views
    pages
      index.ejs
      menu.ejs
    partials
  package-lock.json
  package.json
  tsconfig.json
```

Analyzer

Tokens

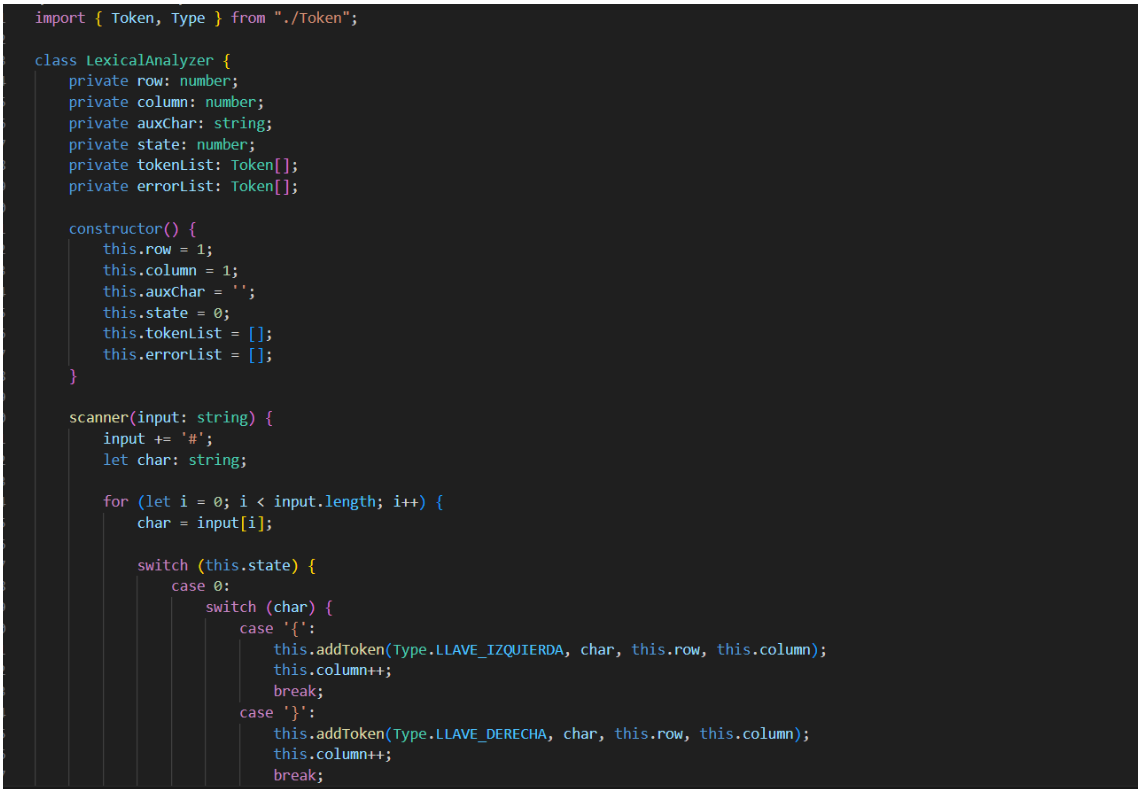
Codigo para el declaramiento de tokens



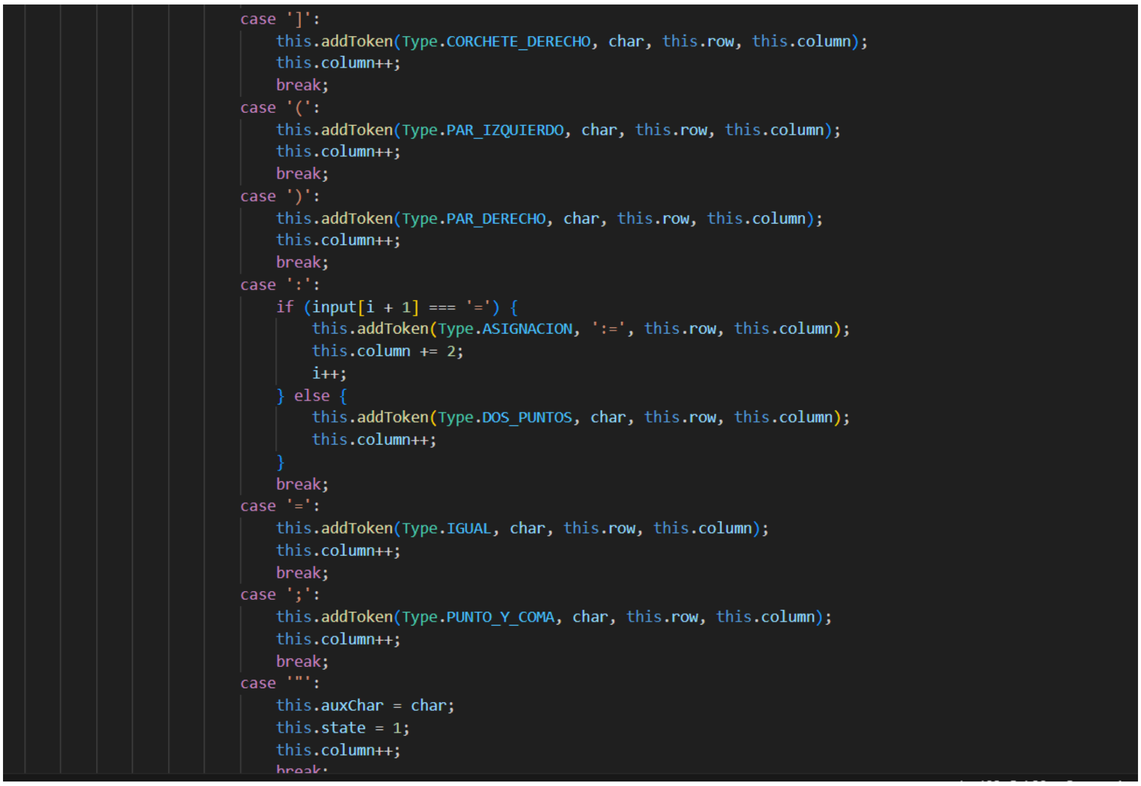
```
enum Type {
    DESCONOCIDO,
    LLAVE_IZQUIERDA,
    LLAVE_DERECHA,
    CORCHETE_IZQUIERDO,
    CORCHETE_DERECHO,
    PAR_IZQUIERDO,
    PAR_DERECHO,
    DOS_PUNTOS,
    ASIGNACION,
    PUNTO_Y_COMA,
    IGUAL,
    PALABRA_RESERVADA,
    NUMERO,
    CADENA
}

class Token {
    public row: number;
    public column: number;
    public lexeme: string;
    public typeToken: Type;
    public typeTokenString: string;

    constructor(typeToken: Type, lexeme: string, row: number, column: number){
        this.typeTokenString =Type[typeToken];
        this.typeToken = typeToken;
        this.lexeme = lexeme;
        this.row = row;
        this.column = column;
    }


}

export { Token, Type };
```

LexicalAnalyzer

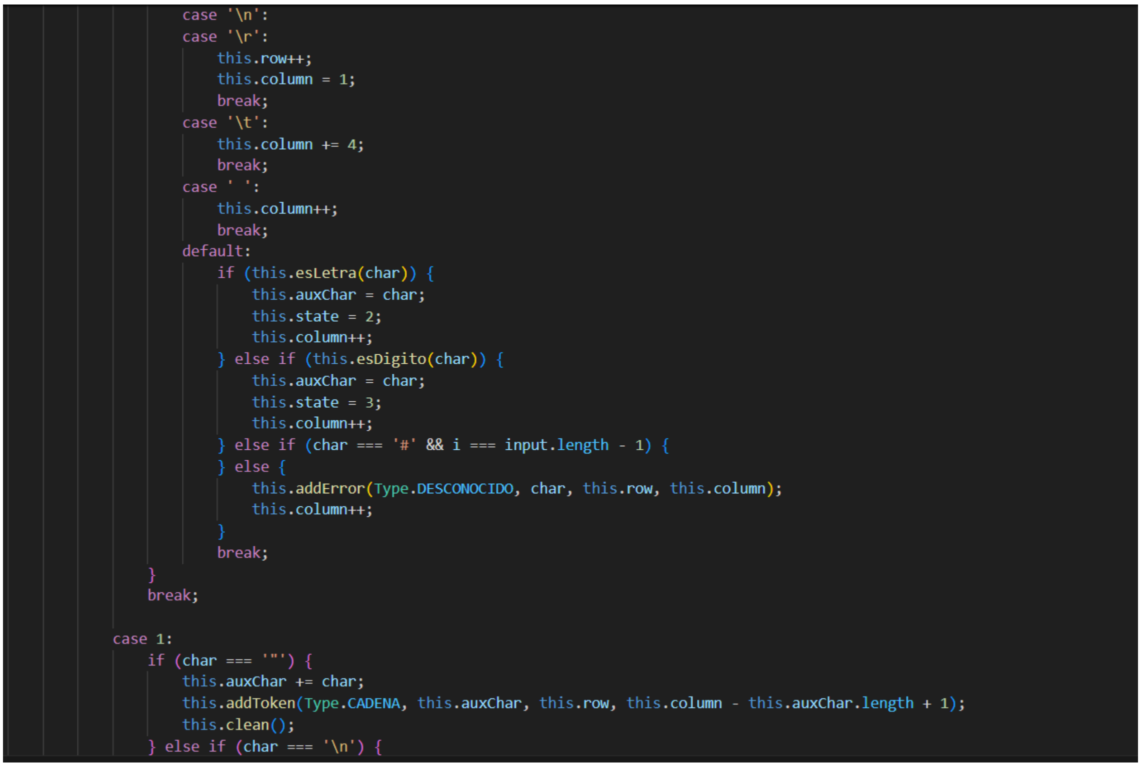
Codigo del analizador lexico.



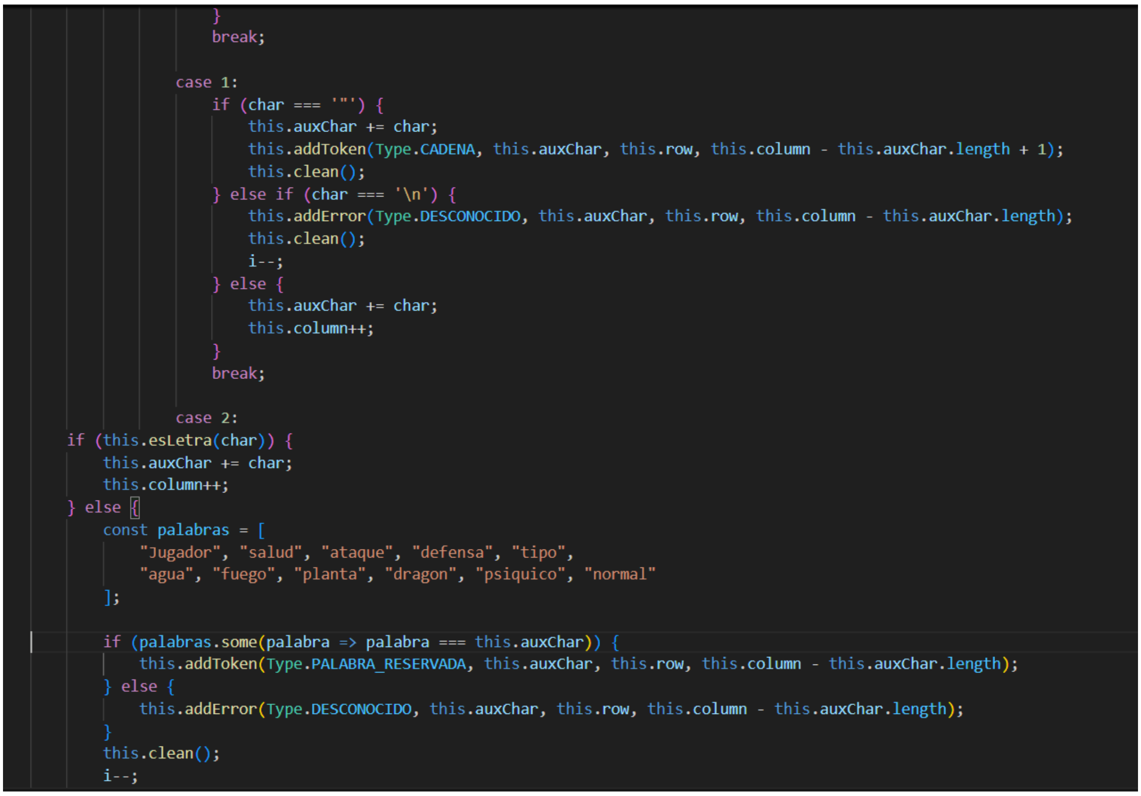
```
import { Token, Type } from "./Token";

class LexicalAnalyzer {
    private row: number;
    private column: number;
    private auxChar: string;
    private state: number;
    private tokenList: Token[];
    private errorList: Token[];

    constructor() {
        this.row = 1;
        this.column = 1;
        this.auxChar = '';
        this.state = 0;
        this.tokenList = [];
        this.errorList = [];
    }

    scanner(input: string) {
        input += '#';
        let char: string;

        for (let i = 0; i < input.length; i++) {
            char = input[i];

            switch (this.state) {
                case 0:
                    switch (char) {
                        case '{':
                            this.addToken(Type.LLAVE_IZQUIERDA, char, this.row, this.column);
                            this.column++;
                            break;
                        case '}':
                            this.addToken(Type.LLAVE_DERECHA, char, this.row, this.column);
                            this.column++;
                            break;
```



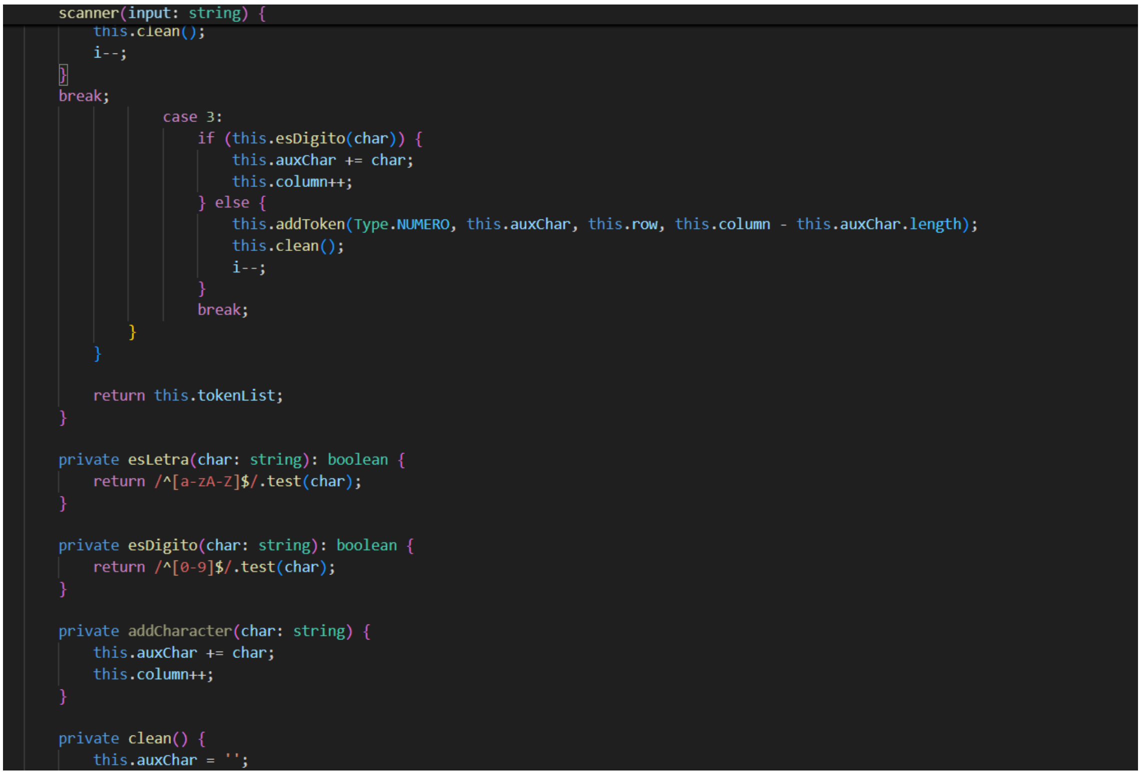
```
                        case ']':
                            this.addToken(Type.CORCHETE_DERECHO, char, this.row, this.column);
                            this.column++;
                            break;
                        case '(':
                            this.addToken(Type.PAR_IZQUIERDO, char, this.row, this.column);
                            this.column++;
                            break;
                        case ')':
                            this.addToken(Type.PAR_DERECHO, char, this.row, this.column);
                            this.column++;
                            break;
                        case ':':
                            if (input[i + 1] === '=') {
                                this.addToken(Type.ASIGNACION, ':=', this.row, this.column);
                                this.column += 2;
                                i++;
                            } else {
                                this.addToken(Type.DOS_PUNTOS, char, this.row, this.column);
                                this.column++;
                            }
                            break;
                        case '=':
                            this.addToken(Type.IGUAL, char, this.row, this.column);
                            this.column++;
                            break;
                        case ';':
                            this.addToken(Type.PUNTO_Y_COMA, char, this.row, this.column);
                            this.column++;
                            break;
                        case '"':
                            this.auxChar = char;
                            this.state = 1;
                            this.column++;
                            break;
```



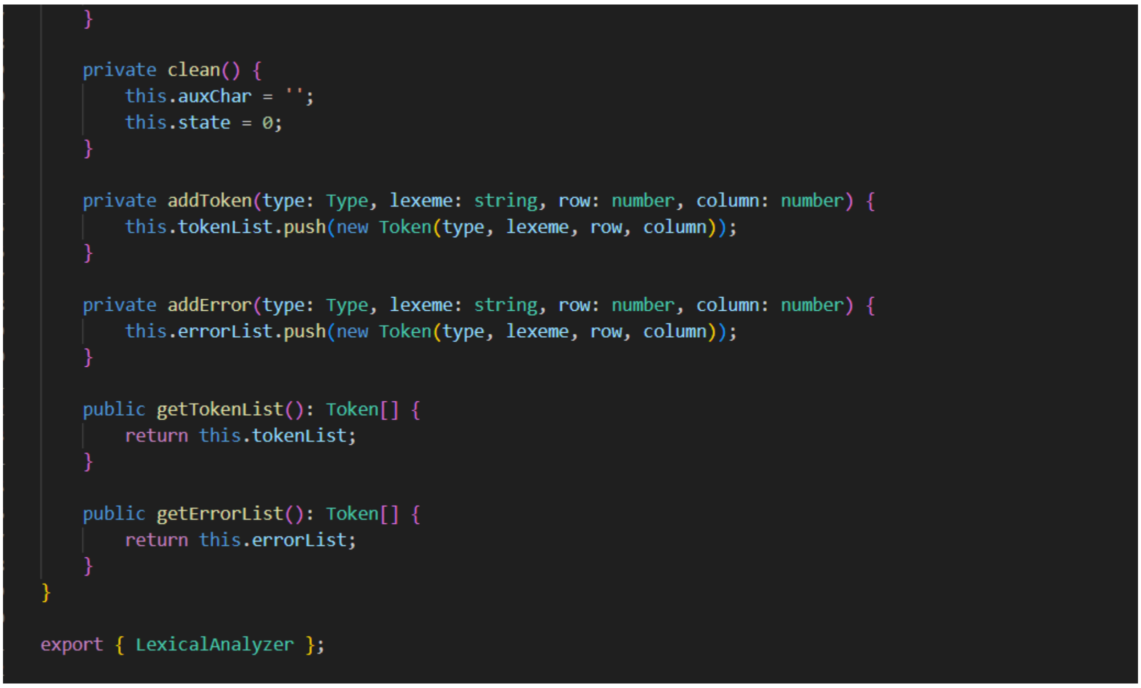
```
                    case '\n':
                    case '\r':
                        this.row++;
                        this.column = 1;
                        break;
                    case '\t':
                        this.column += 4;
                        break;
                    case ' ':
                        this.column++;
                        break;
                    default:
                        if (this.esLetra(char)) {
                            this.auxChar = char;
                            this.state = 2;
                            this.column++;
                        } else if (this.esDigito(char)) {
                            this.auxChar = char;
                            this.state = 3;
                            this.column++;
                        } else if (char === '#' && i === input.length - 1) {
                        } else {
                            this.addError(Type.DESCONOCIDO, char, this.row, this.column);
                            this.column++;
                        }
                        break;
                }
                break;

            case 1:
                if (char === '"') {
                    this.auxChar += char;
                    this.addToken(Type.CADENA, this.auxChar, this.row, this.column - this.auxChar.length + 1);
                    this.clean();
                } else if (char === '\n') {
```



```
                    }
                    break;

                case 1:
                    if (char === '"') {
                        this.auxChar += char;
                        this.addToken(Type.CADENA, this.auxChar, this.row, this.column - this.auxChar.length + 1);
                        this.clean();
                    } else if (char === '\n') {
                        this.addError(Type.DESCONOCIDO, this.auxChar, this.row, this.column - this.auxChar.length);
                        this.clean();
                        i--;
                    } else {
                        this.auxChar += char;
                        this.column++;
                    }
                    break;

                case 2:
    if (this.esLetra(char)) {
        this.auxChar += char;
        this.column++;
    } else {
        const palabras = [
            "Jugador", "salud", "ataque", "defensa", "tipo",
            "agua", "fuego", "planta", "dragon", "psiquico", "normal"
        ];

        if (palabras.some(palabra => palabra === this.auxChar)) {
            this.addToken(Type.PALABRA_RESERVADA, this.auxChar, this.row, this.column - this.auxChar.length);
        } else {
            this.addError(Type.DESCONOCIDO, this.auxChar, this.row, this.column - this.auxChar.length);
        }
        this.clean();
        i--;
```



```
    scanner(input: string) {
        this.clean();
        i--;
    }
    break;
                case 3:
                    if (this.esDigito(char)) {
                        this.auxChar += char;
                        this.column++;
                    } else {
                        this.addToken(Type.NUMERO, this.auxChar, this.row, this.column - this.auxChar.length);
                        this.clean();
                        i--;
                    }
                    break;
            }
        }

        return this.tokenList;
    }

    private esLetra(char: string): boolean {
        return /^[a-zA-Z]$/.test(char);
    }

    private esDigito(char: string): boolean {
        return /^[0-9]$/.test(char);
    }

    private addCharacter(char: string) {
        this.auxChar += char;
        this.column++;
    }

    private clean() {
        this.auxChar = '';
```

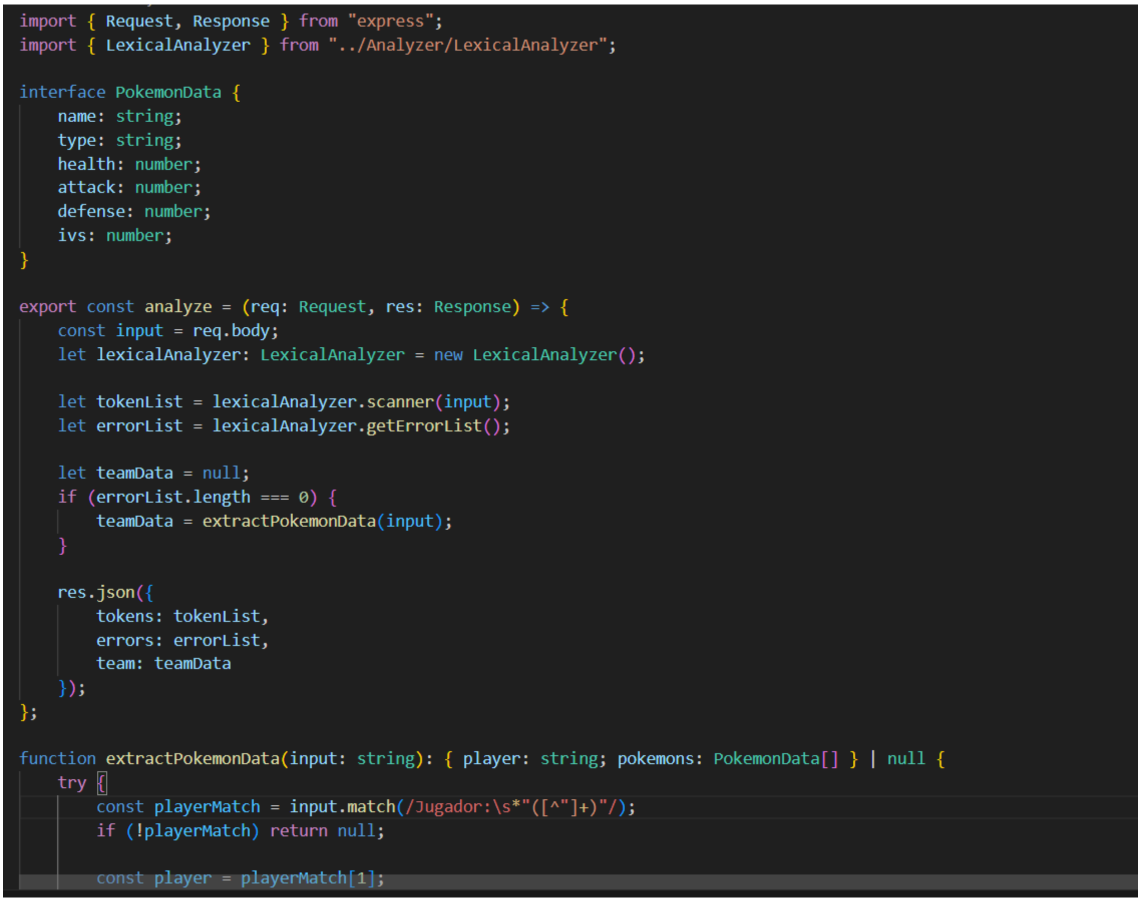


```
    }

    private clean() {
        this.auxChar = '';
        this.state = 0;
    }

    private addToken(type: Type, lexeme: string, row: number, column: number) {
        this.tokenList.push(new Token(type, lexeme, row, column));
    }

    private addError(type: Type, lexeme: string, row: number, column: number) {
        this.errorList.push(new Token(type, lexeme, row, column));
    }

    public getTokenList(): Token[] {
        return this.tokenList;
    }

    public getErrorList(): Token[] {
        return this.errorList;
    }
}

export { LexicalAnalyzer };
```

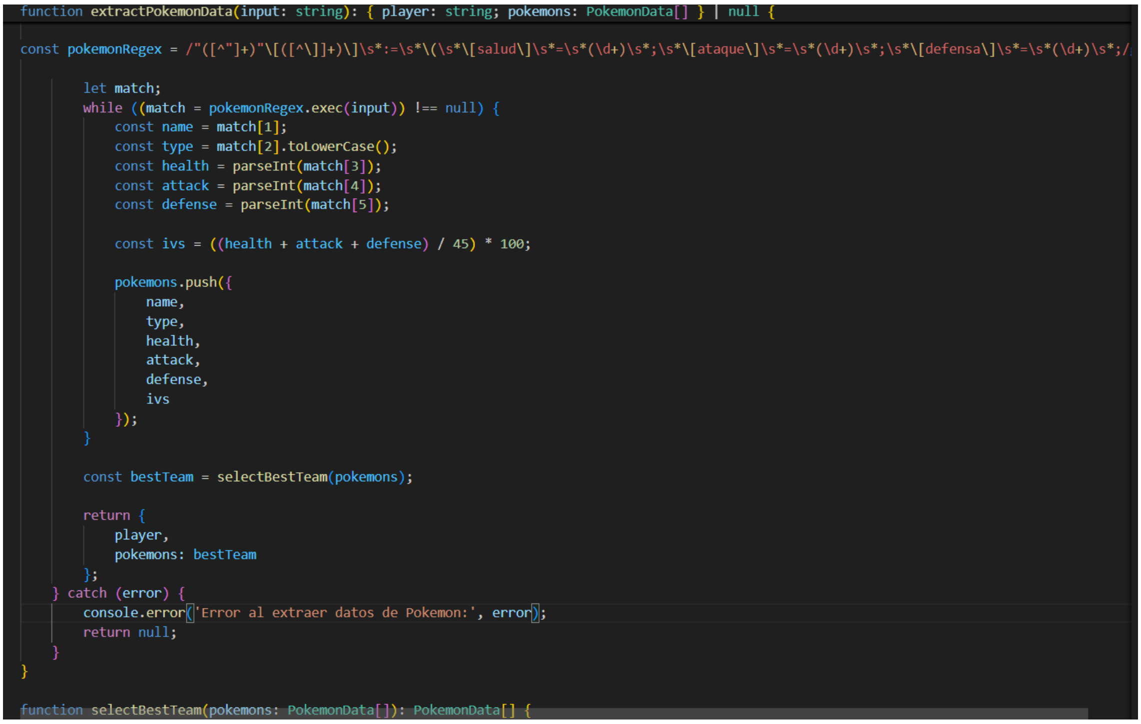
Analyzers

Analyze.controller

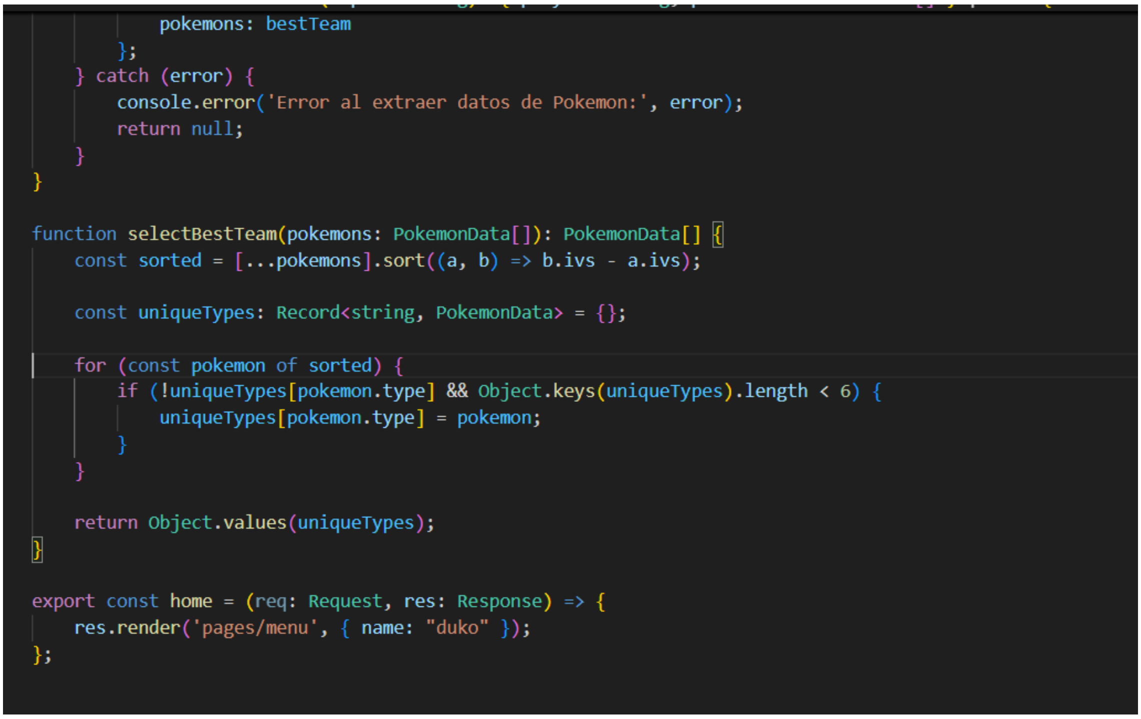
Codigo utilizado para controlador.



```
import { Request, Response } from "express";
import { LexicalAnalyzer } from "../Analyzer/LexicalAnalyzer";

interface PokemonData {
    name: string;
    type: string;
    health: number;
    attack: number;
    defense: number;
    ivs: number;
}

export const analyze = (req: Request, res: Response) => {
    const input = req.body;
    let lexicalAnalyzer: LexicalAnalyzer = new LexicalAnalyzer();

    let tokenList = lexicalAnalyzer.scanner(input);
    let errorList = lexicalAnalyzer.getErrorList();

    let teamData = null;
    if (errorList.length === 0) {
        teamData = extractPokemonData(input);
    }

    res.json({
        tokens: tokenList,
        errors: errorList,
        team: teamData
    });
};

function extractPokemonData(input: string): { player: string; pokemons: PokemonData[] } | null {
    try {
        const playerMatch = input.match(/Jugador:\s*"([^"]+)"/);
        if (!playerMatch) return null;

        const player = playerMatch[1];
```



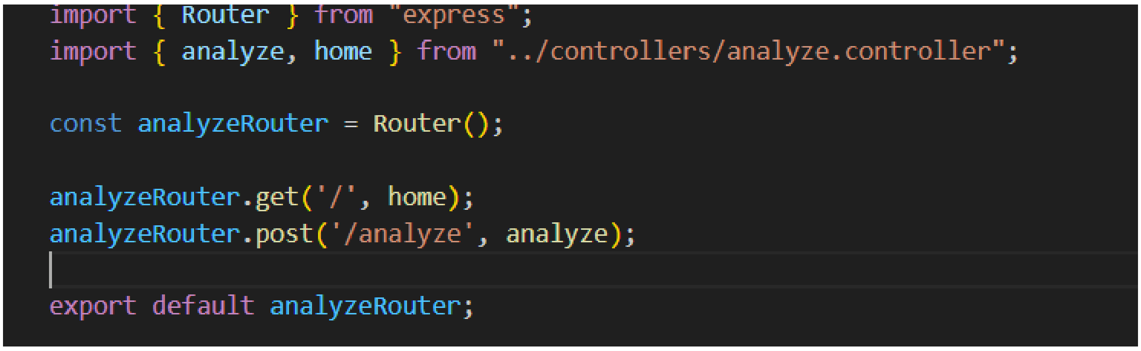
```
function extractPokemonData(input: string): { player: string; pokemons: PokemonData[] } | null {

const pokemonRegex = /"([^"]+)"\[([^\]]+)\]\s*:=\s*\(\s*\[salud\]\s*=\s*(\d+)\s*;\s*\[ataque\]\s*=\s*(\d+)\s*;\s*\[defensa\]\s*=\s*(\d+)\s*;/

        let match;
        while ((match = pokemonRegex.exec(input)) !== null) {
            const name = match[1];
            const type = match[2].toLowerCase();
            const health = parseInt(match[3]);
            const attack = parseInt(match[4]);
            const defense = parseInt(match[5]);

            const ivs = ((health + attack + defense) / 45) * 100;

            pokemons.push({
                name,
                type,
                health,
                attack,
                defense,
                ivs
            });
        }

        const bestTeam = selectBestTeam(pokemons);

        return {
            player,
            pokemons: bestTeam
        };
    } catch (error) {
        console.error('Error al extraer datos de Pokemon:', error);
        return null;
    }
}

function selectBestTeam(pokemons: PokemonData[]): PokemonData[] {
```



```
            pokemons: bestTeam
        };
    } catch (error) {
        console.error('Error al extraer datos de Pokemon:', error);
        return null;
    }
}

function selectBestTeam(pokemons: PokemonData[]): PokemonData[] {
    const sorted = [...pokemons].sort((a, b) => b.ivs - a.ivs);

    const uniqueTypes: Record<string, PokemonData> = {};

    for (const pokemon of sorted) {
        if (!uniqueTypes[pokemon.type] && Object.keys(uniqueTypes).length < 6) {
            uniqueTypes[pokemon.type] = pokemon;
        }
    }

    return Object.values(uniqueTypes);
}

export const home = (req: Request, res: Response) => {
    res.render('pages/menu', { name: "duko" });
};
```

Analyze.route

Codigo utilizado para la ruta a llamar.

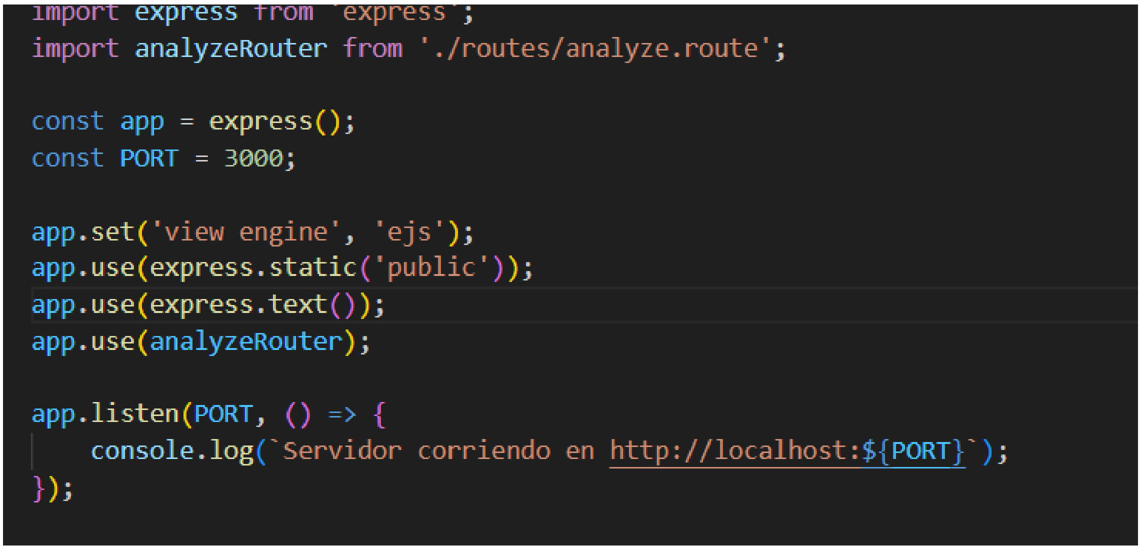


```
import { Router } from "express";
import { analyze, home } from "../controllers/analyze.controller";

const analyzeRouter = Router();

analyzeRouter.get('/', home);
analyzeRouter.post('/analyze', analyze);

export default analyzeRouter;
```

pages

index.ts

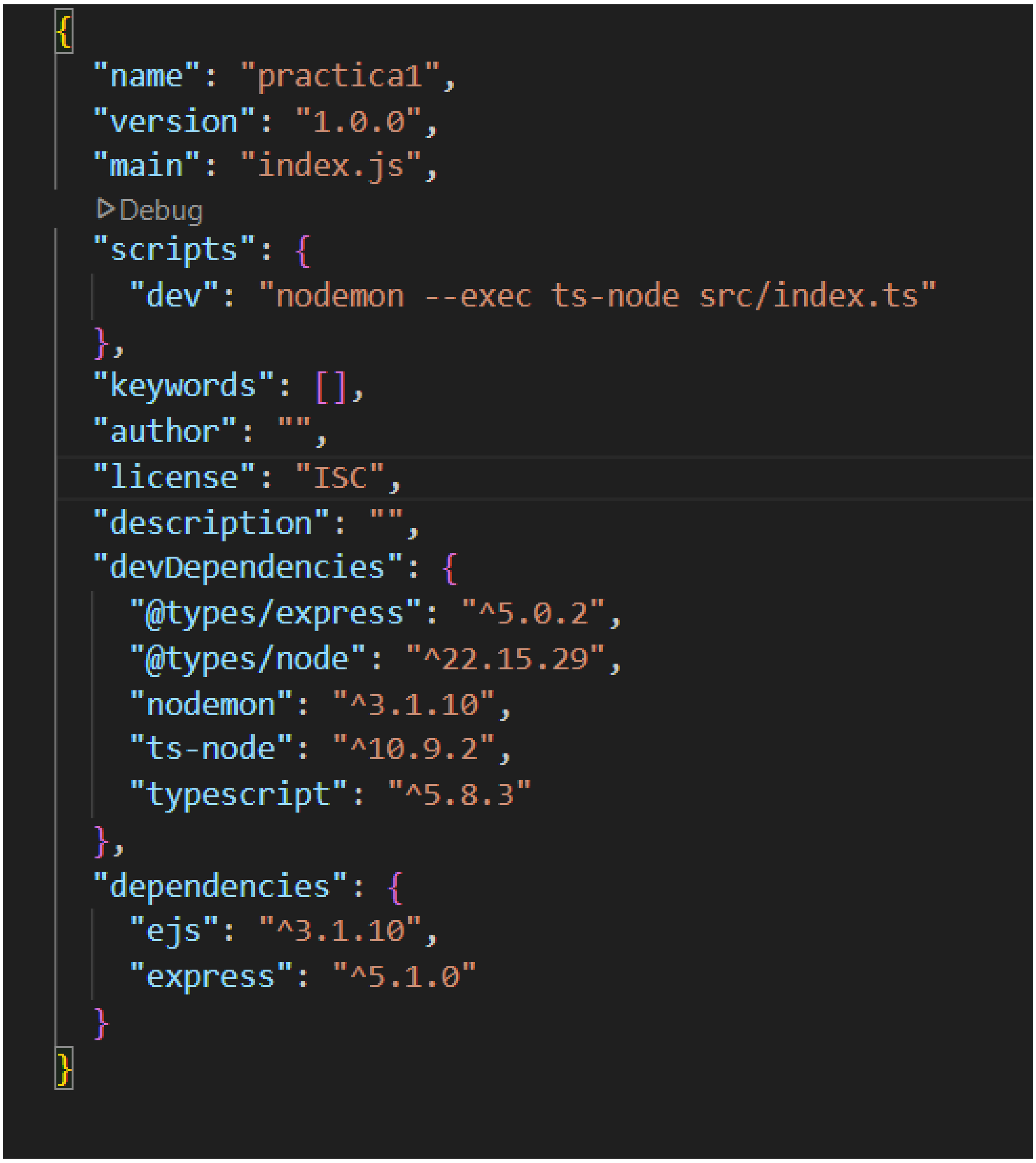
codigo principal para definir la ruta del servidor.



```
import express from 'express';
import analyzeRouter from './routes/analyze.route';

const app = express();
const PORT = 3000;

app.set('view engine', 'ejs');
app.use(express.static('public'));
app.use(express.text());
app.use(analyzeRouter);

app.listen(PORT, () => {
    console.log(`Servidor corriendo en http://localhost:${PORT}`);
});
```

Package.json

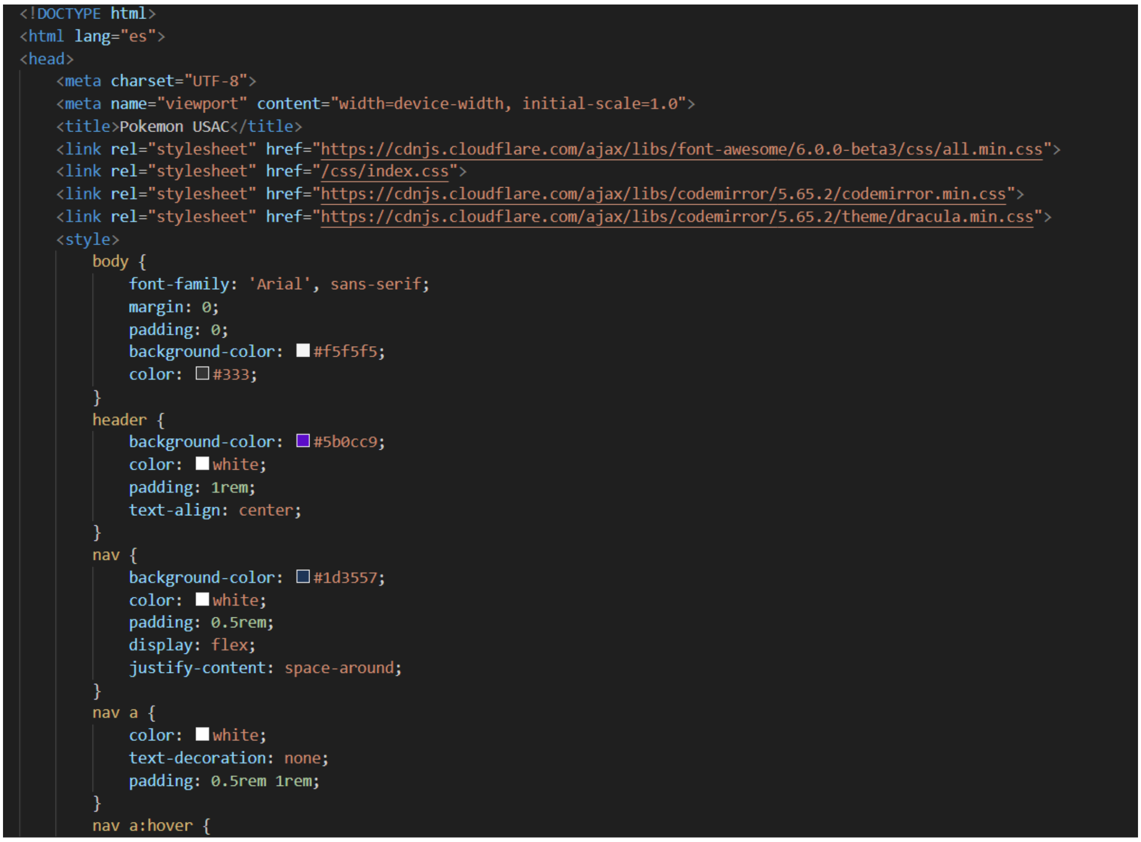
codigo que contiene la ruta



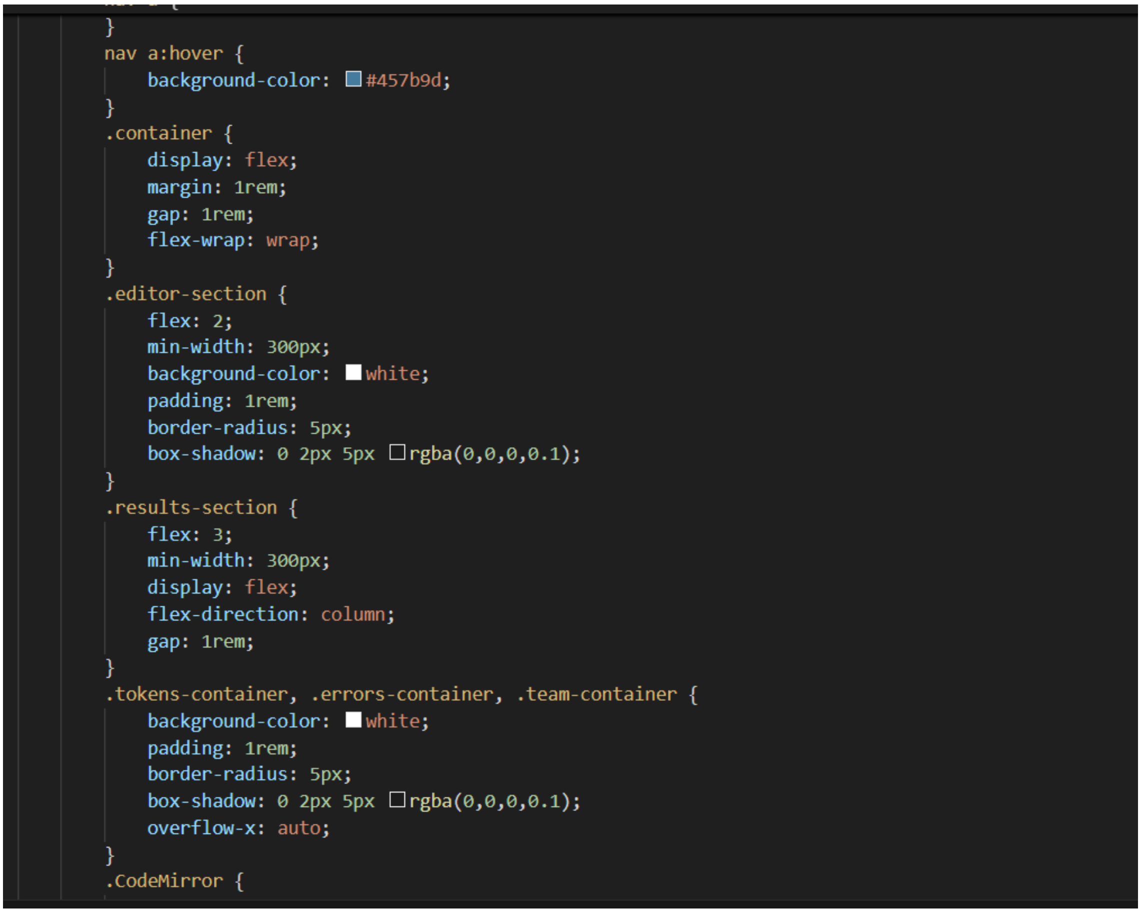
```
{
  "name": "practica1",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon --exec ts-node src/index.ts"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "devDependencies": {
    "@types/express": "^5.0.2",
    "@types/node": "^22.15.29",
    "nodemon": "^3.1.10",
    "ts-node": "^10.9.2",
    "typescript": "^5.8.3"
  },
  "dependencies": {
    "ejs": "^3.1.10",
    "express": "^5.1.0"
  }
}
```

Menu.ejs

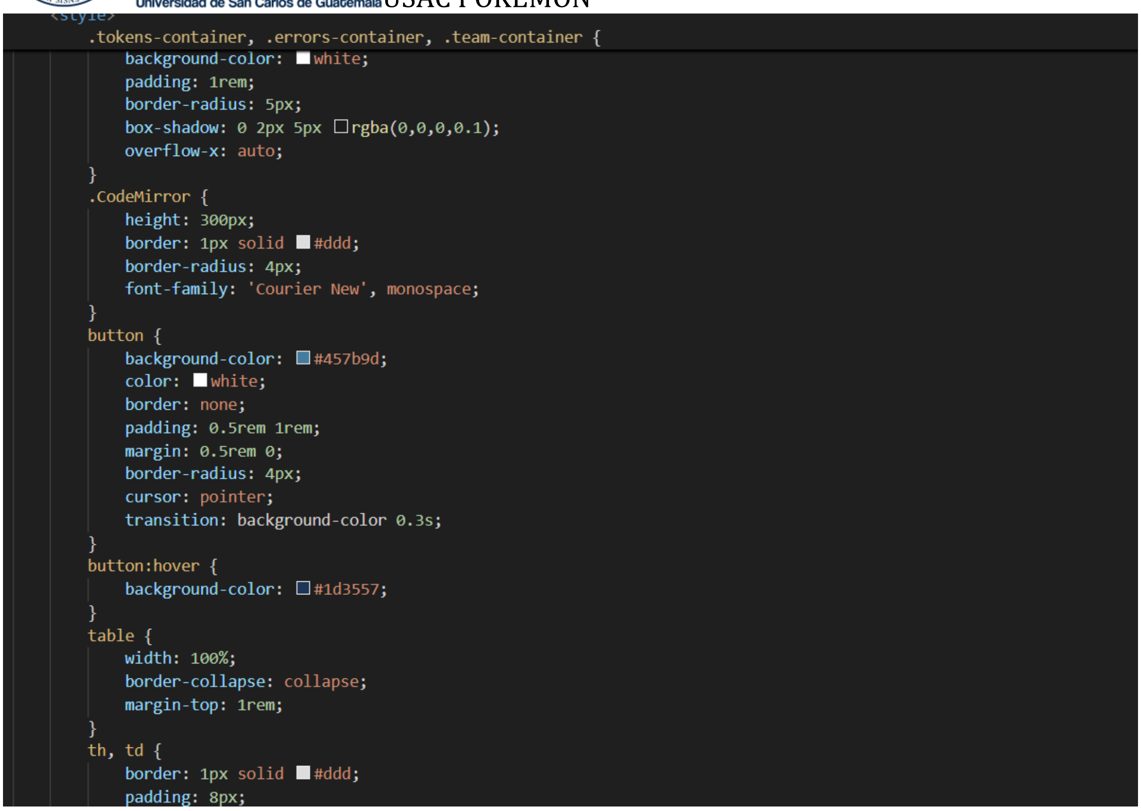
codigo que contiene el html de la pagina principal.



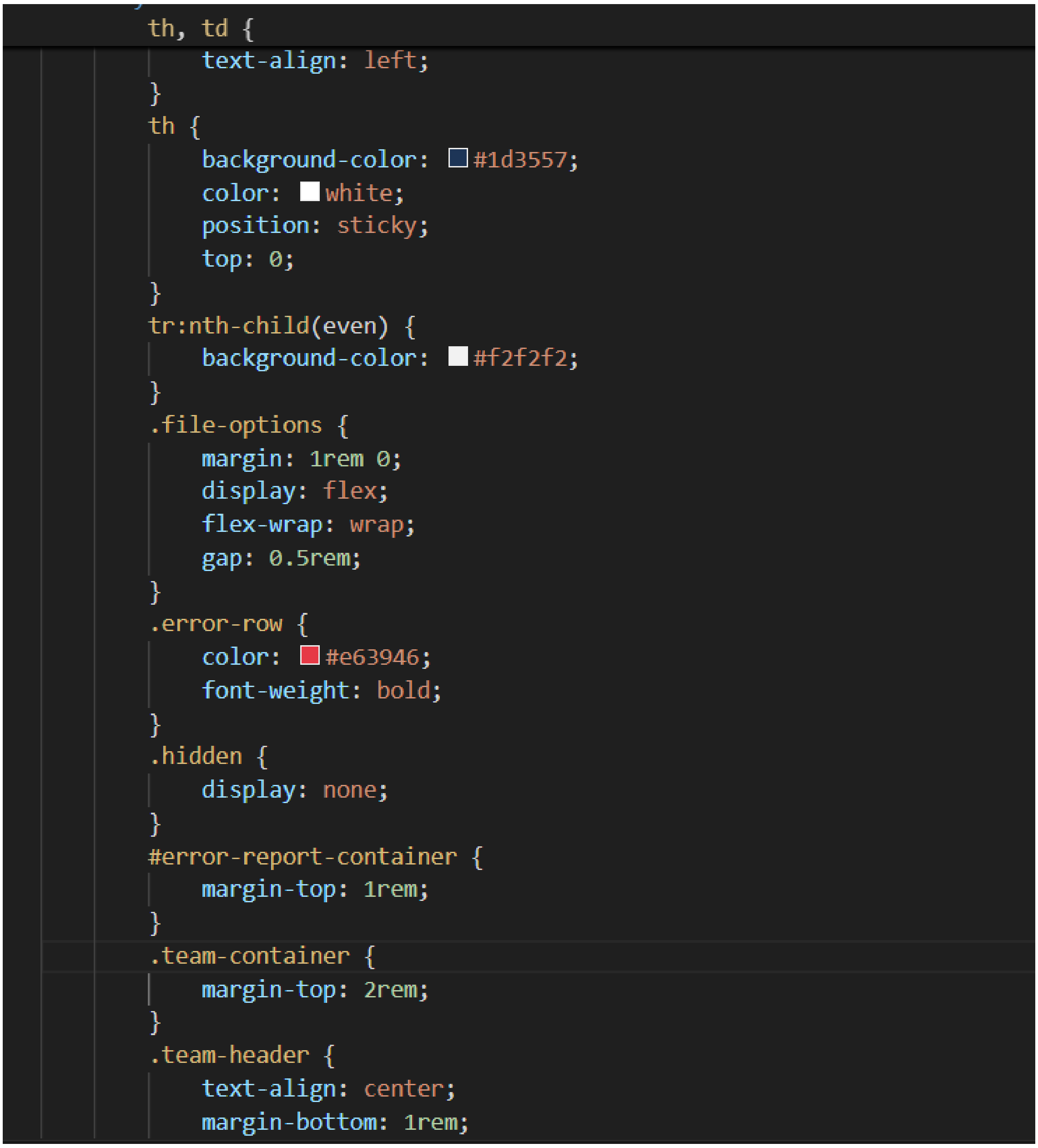
```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pokemon USAC</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
    <link rel="stylesheet" href="/css/index.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/codemirror.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/theme/dracula.min.css">
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f5f5f5;
            color: #333;
        }
        header {
            background-color: #5b0cc9;
            color: white;
            padding: 1rem;
            text-align: center;
        }
        nav {
            background-color: #1d3557;
            color: white;
            padding: 0.5rem;
            display: flex;
            justify-content: space-around;
        }
        nav a {
            color: white;
            text-decoration: none;
            padding: 0.5rem 1rem;
        }
        nav a:hover {
```



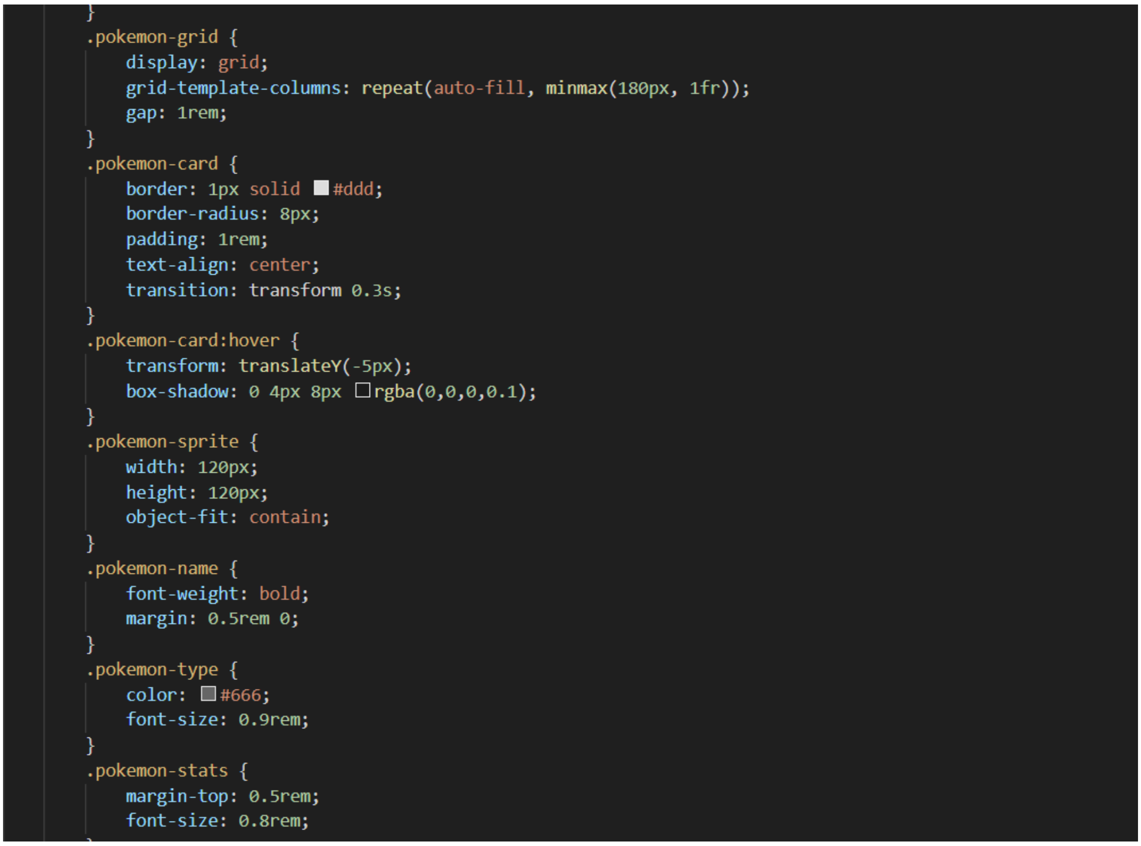
```
        nav a {
        }
        nav a:hover {
            background-color: #457b9d;
        }
        .container {
            display: flex;
            margin: 1rem;
            gap: 1rem;
            flex-wrap: wrap;
        }
        .editor-section {
            flex: 2;
            min-width: 300px;
            background-color: white;
            padding: 1rem;
            border-radius: 5px;
            box-shadow: 0 2px 5px rgba(0,0,0,0.1);
        }
        .results-section {
            flex: 3;
            min-width: 300px;
            display: flex;
            flex-direction: column;
            gap: 1rem;
        }
        .tokens-container, .errors-container, .team-container {
            background-color: white;
            padding: 1rem;
            border-radius: 5px;
            box-shadow: 0 2px 5px rgba(0,0,0,0.1);
            overflow-x: auto;
        }
        .CodeMirror {
```



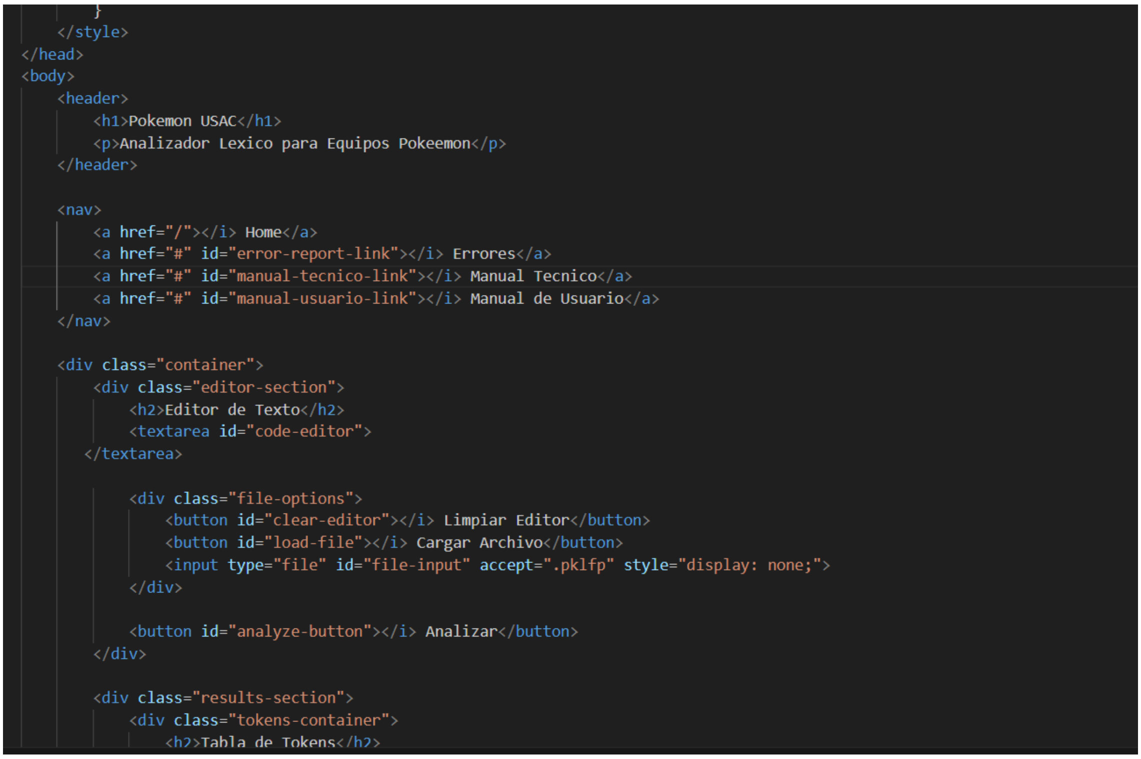
```
<style>
    .tokens-container, .errors-container, .team-container {
        background-color: white;
        padding: 1rem;
        border-radius: 5px;
        box-shadow: 0 2px 5px rgba(0,0,0,0.1);
        overflow-x: auto;
    }
    .CodeMirror {
        height: 300px;
        border: 1px solid #ddd;
        border-radius: 4px;
        font-family: 'Courier New', monospace;
    }
    button {
        background-color: #457b9d;
        color: white;
        border: none;
        padding: 0.5rem 1rem;
        margin: 0.5rem 0;
        border-radius: 4px;
        cursor: pointer;
        transition: background-color 0.3s;
    }
    button:hover {
        background-color: #1d3557;
    }
    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 1rem;
    }
    th, td {
        border: 1px solid #ddd;
        padding: 8px;
```



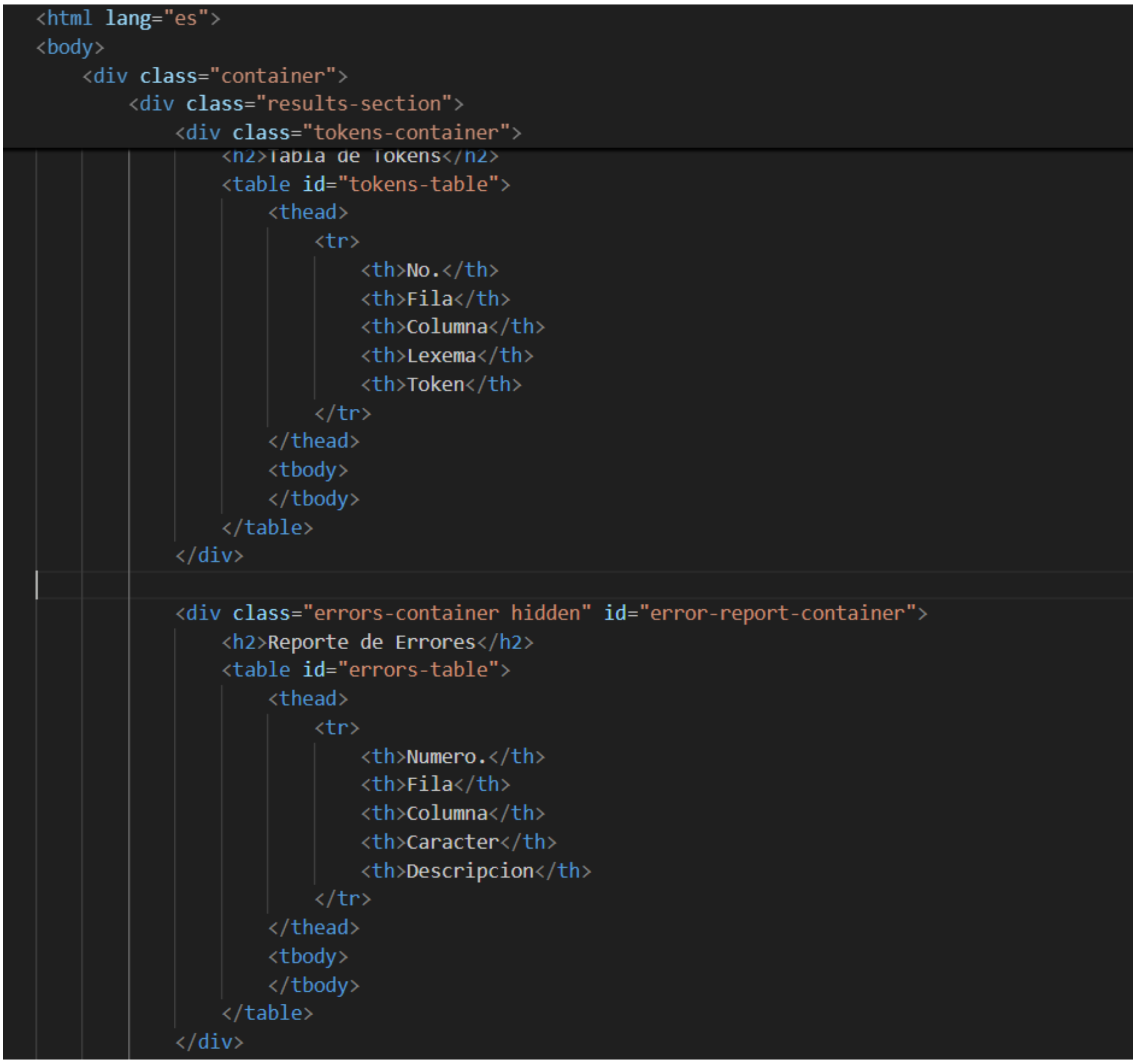
```
        th, td {
            text-align: left;
        }
        th {
            background-color: #1d3557;
            color: white;
            position: sticky;
            top: 0;
        }
        tr:nth-child(even) {
            background-color: #f2f2f2;
        }
        .file-options {
            margin: 1rem 0;
            display: flex;
            flex-wrap: wrap;
            gap: 0.5rem;
        }
        .error-row {
            color: #e63946;
            font-weight: bold;
        }
        .hidden {
            display: none;
        }
        #error-report-container {
            margin-top: 1rem;
        }
        .team-container {
            margin-top: 2rem;
        }
        .team-header {
            text-align: center;
            margin-bottom: 1rem;
```



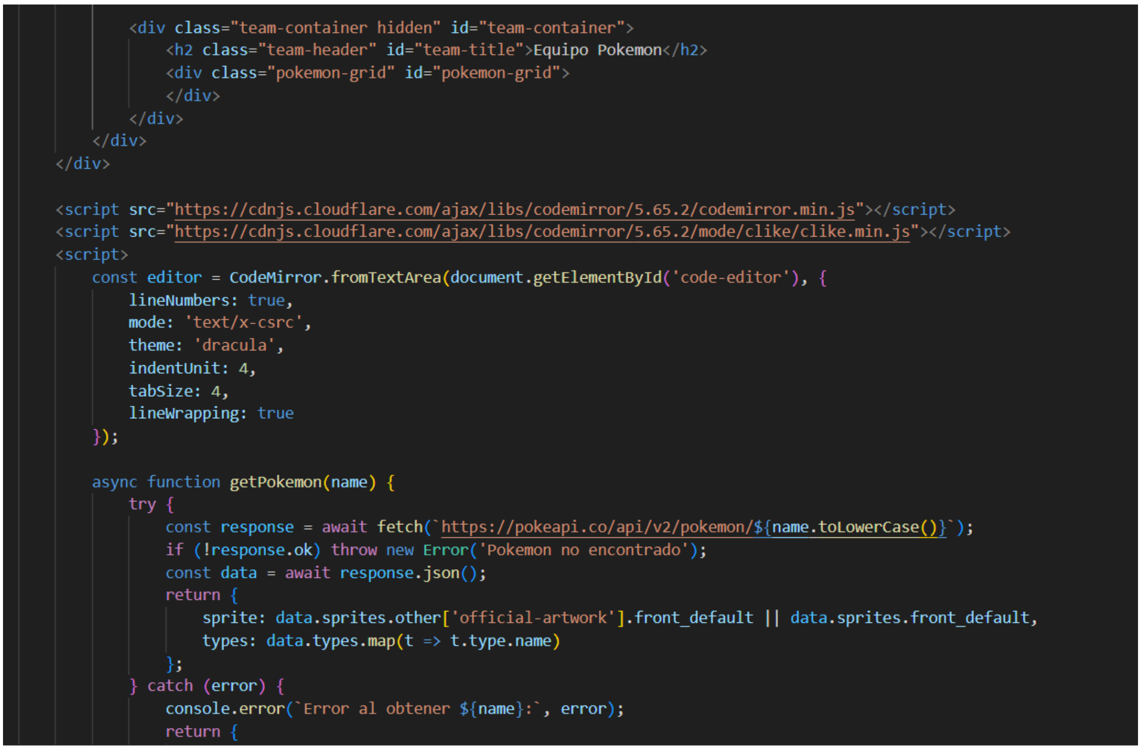
```
            }
        .pokemon-grid {
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(180px, 1fr));
            gap: 1rem;
        }
        .pokemon-card {
            border: 1px solid ■#ddd;
            border-radius: 8px;
            padding: 1rem;
            text-align: center;
            transition: transform 0.3s;
        }
        .pokemon-card:hover {
            transform: translateY(-5px);
            box-shadow: 0 4px 8px □rgba(0,0,0,0.1);
        }
        .pokemon-sprite {
            width: 120px;
            height: 120px;
            object-fit: contain;
        }
        .pokemon-name {
            font-weight: bold;
            margin: 0.5rem 0;
        }
        .pokemon-type {
            color: ■#666;
            font-size: 0.9rem;
        }
        .pokemon-stats {
            margin-top: 0.5rem;
            font-size: 0.8rem;
```



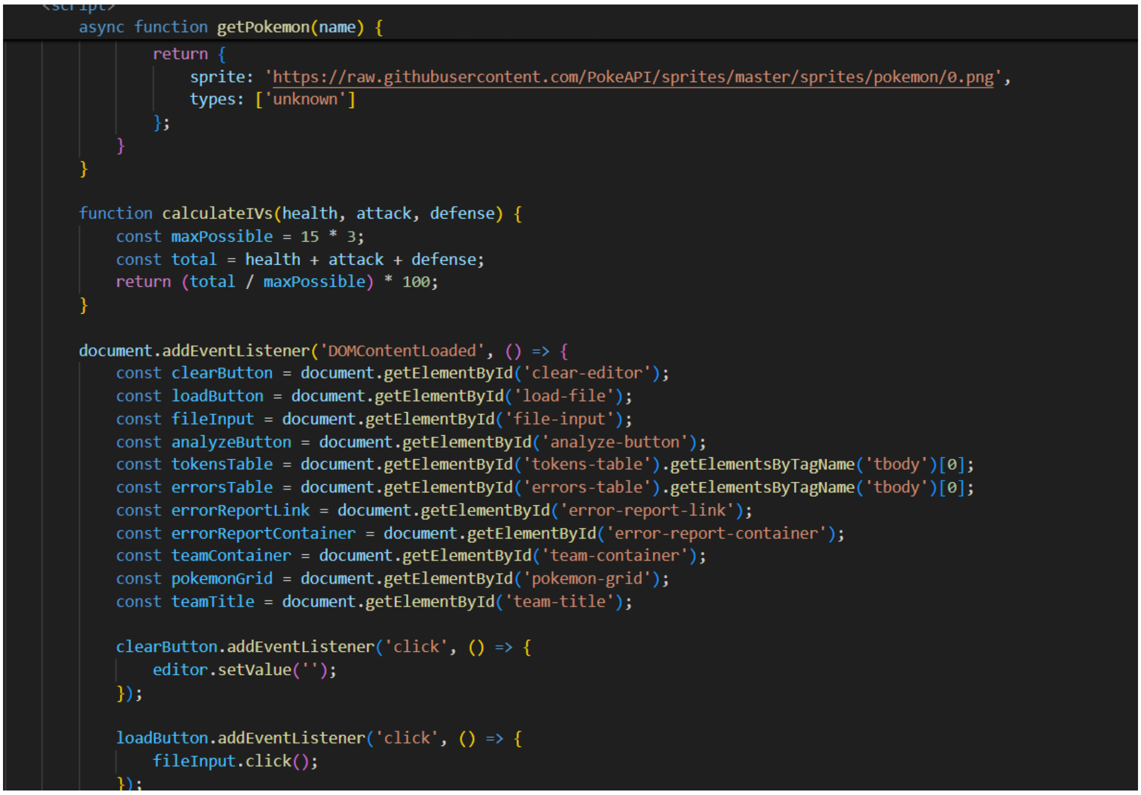
```
            }
        </style>
    </head>
    <body>
        <header>
            <h1>Pokemon USAC</h1>
            <p>Analizador Lexico para Equipos Pokeemon</p>
        </header>

        <nav>
            <a href="/"></i> Home</a>
            <a href="#" id="error-report-link"></i> Errores</a>
            <a href="#" id="manual-tecnico-link"></i> Manual Tecnico</a>
            <a href="#" id="manual-usuario-link"></i> Manual de Usuario</a>
        </nav>

        <div class="container">
            <div class="editor-section">
                <h2>Editor de Texto</h2>
                <textarea id="code-editor">
           </textarea>

                <div class="file-options">
                    <button id="clear-editor"></i> Limpiar Editor</button>
                    <button id="load-file"></i> Cargar Archivo</button>
                    <input type="file" id="file-input" accept=".pklfp" style="display: none;">
                </div>

                <button id="analyze-button"></i> Analizar</button>
            </div>

            <div class="results-section">
                <div class="tokens-container">
                    <h2>Tabla de Tokens</h2>
```



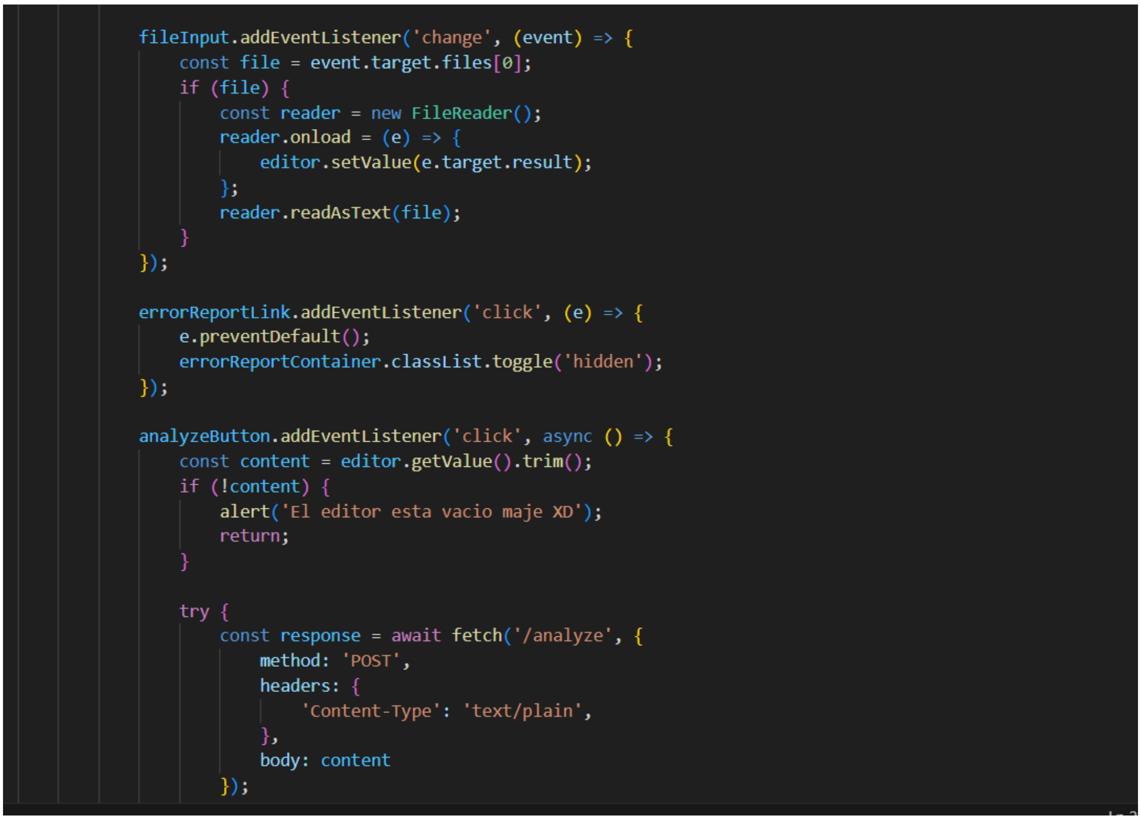
```
<html lang="es">
<body>
    <div class="container">
        <div class="results-section">
            <div class="tokens-container">
                <h2>Tabla de Tokens</h2>
                <table id="tokens-table">
                    <thead>
                        <tr>
                            <th>No.</th>
                            <th>Fila</th>
                            <th>Columna</th>
                            <th>Lexema</th>
                            <th>Token</th>
                        </tr>
                    </thead>
                    <tbody>
                    </tbody>
                </table>
            </div>

            <div class="errors-container hidden" id="error-report-container">
                <h2>Reporte de Errores</h2>
                <table id="errors-table">
                    <thead>
                        <tr>
                            <th>Numero.</th>
                            <th>Fila</th>
                            <th>Columna</th>
                            <th>Caracter</th>
                            <th>Descripcion</th>
                        </tr>
                    </thead>
                    <tbody>
                    </tbody>
                </table>
            </div>
```



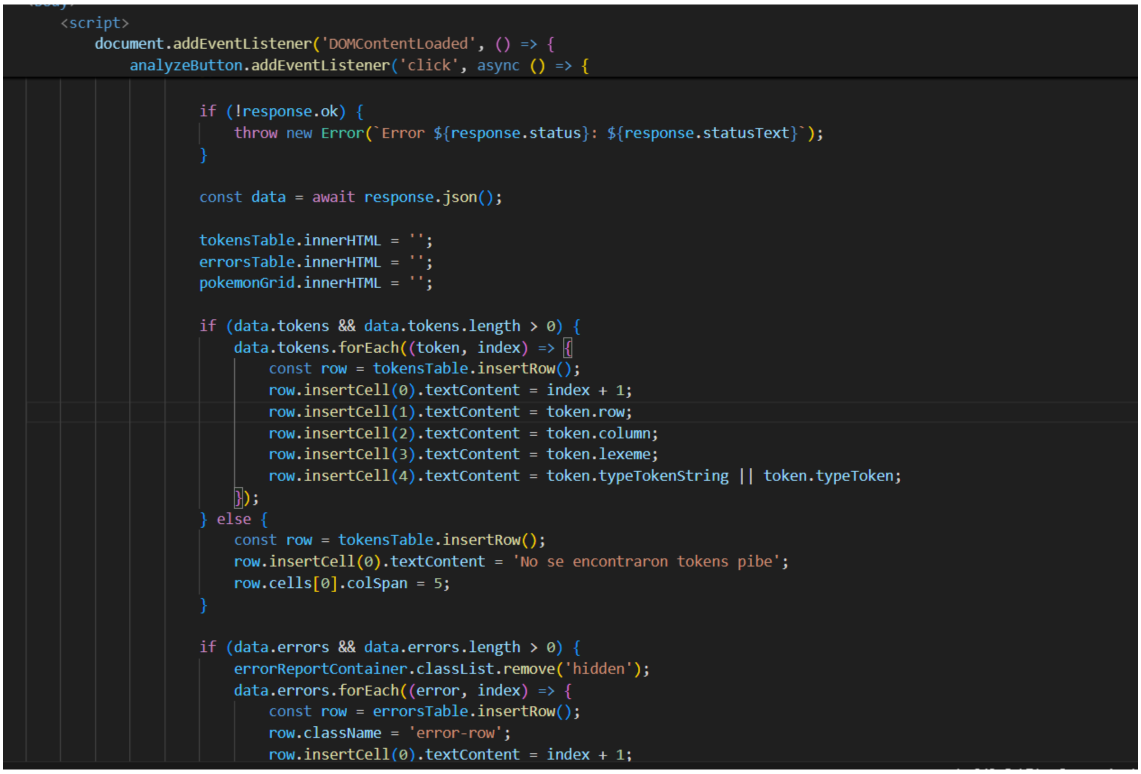
```
        <div class="team-container hidden" id="team-container">
            <h2 class="team-header" id="team-title">Equipo Pokemon</h2>
            <div class="pokemon-grid" id="pokemon-grid">
            </div>
        </div>
    </div>
</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/codemirror.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/mode/clike/clike.min.js"></script>
<script>
    const editor = CodeMirror.fromTextArea(document.getElementById('code-editor'), {
        lineNumbers: true,
        mode: 'text/x-csrc',
        theme: 'dracula',
        indentUnit: 4,
        tabSize: 4,
        lineWrapping: true
    });

    async function getPokemon(name) {
        try {
            const response = await fetch(`https://pokeapi.co/api/v2/pokemon/${name.toLowerCase()}`);
            if (!response.ok) throw new Error('Pokemon no encontrado');
            const data = await response.json();
            return {
                sprite: data.sprites.other['official-artwork'].front_default || data.sprites.front_default,
                types: data.types.map(t => t.type.name)
            };
        } catch (error) {
            console.error(`Error al obtener ${name}:`, error);
            return {
```



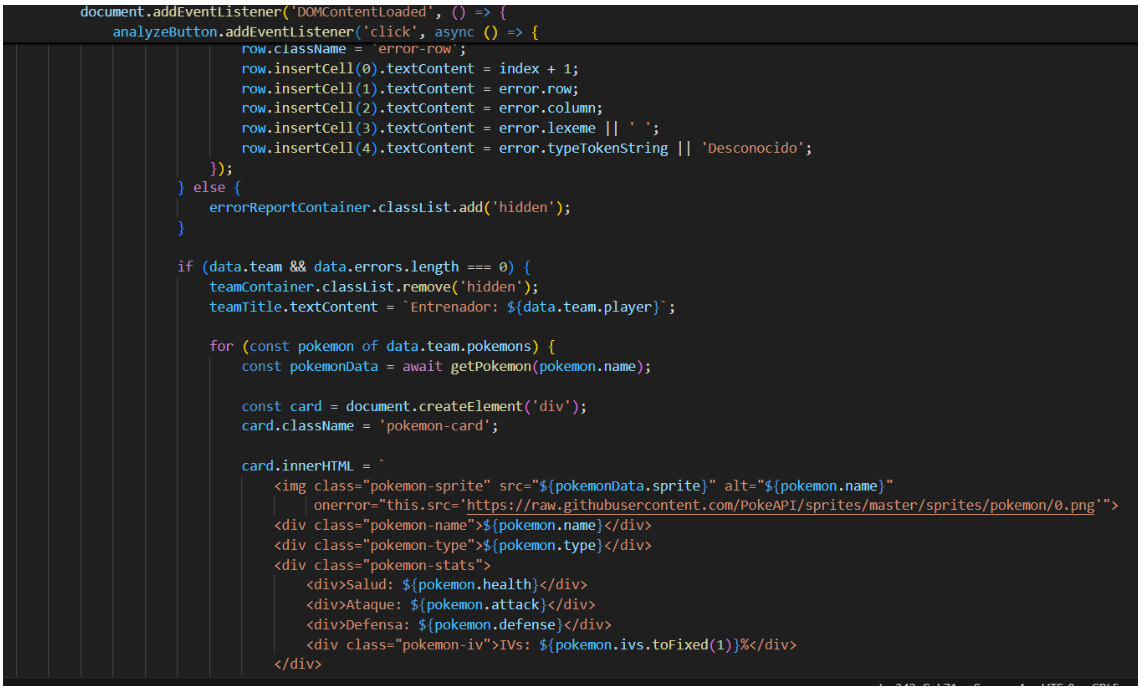
```
<script>
    async function getPokemon(name) {
            return {
                sprite: 'https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/0.png',
                types: ['unknown']
            };
        }
    }

    function calculateIVs(health, attack, defense) {
        const maxPossible = 15 * 3;
        const total = health + attack + defense;
        return (total / maxPossible) * 100;
    }

    document.addEventListener('DOMContentLoaded', () => {
        const clearButton = document.getElementById('clear-editor');
        const loadButton = document.getElementById('load-file');
        const fileInput = document.getElementById('file-input');
        const analyzeButton = document.getElementById('analyze-button');
        const tokensTable = document.getElementById('tokens-table').getElementsByTagName('tbody')[0];
        const errorsTable = document.getElementById('errors-table').getElementsByTagName('tbody')[0];
        const errorReportLink = document.getElementById('error-report-link');
        const errorReportContainer = document.getElementById('error-report-container');
        const teamContainer = document.getElementById('team-container');
        const pokemonGrid = document.getElementById('pokemon-grid');
        const teamTitle = document.getElementById('team-title');

        clearButton.addEventListener('click', () => {
            editor.setValue('');
        });

        loadButton.addEventListener('click', () => {
            fileInput.click();
        });
```



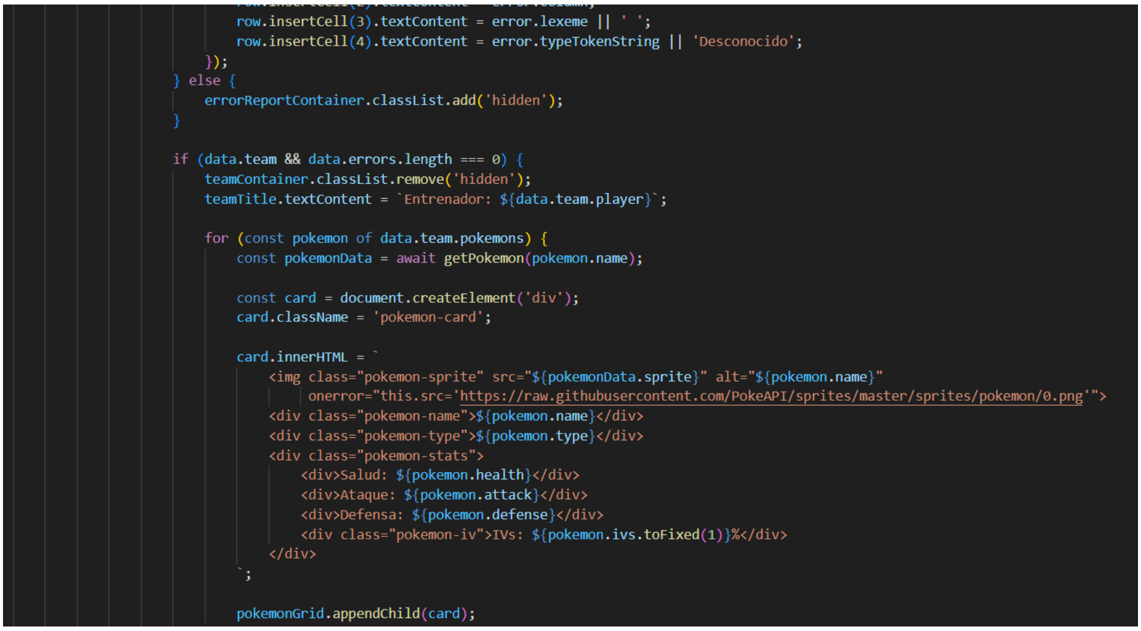
```
        fileInput.addEventListener('change', (event) => {
            const file = event.target.files[0];
            if (file) {
                const reader = new FileReader();
                reader.onload = (e) => {
                    editor.setValue(e.target.result);
                };
                reader.readAsText(file);
            }
        });

        errorReportLink.addEventListener('click', (e) => {
            e.preventDefault();
            errorReportContainer.classList.toggle('hidden');
        });

        analyzeButton.addEventListener('click', async () => {
            const content = editor.getValue().trim();
            if (!content) {
                alert('El editor esta vacio maje XD');
                return;
            }

            try {
                const response = await fetch('/analyze', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'text/plain',
                    },
                    body: content
                });
```



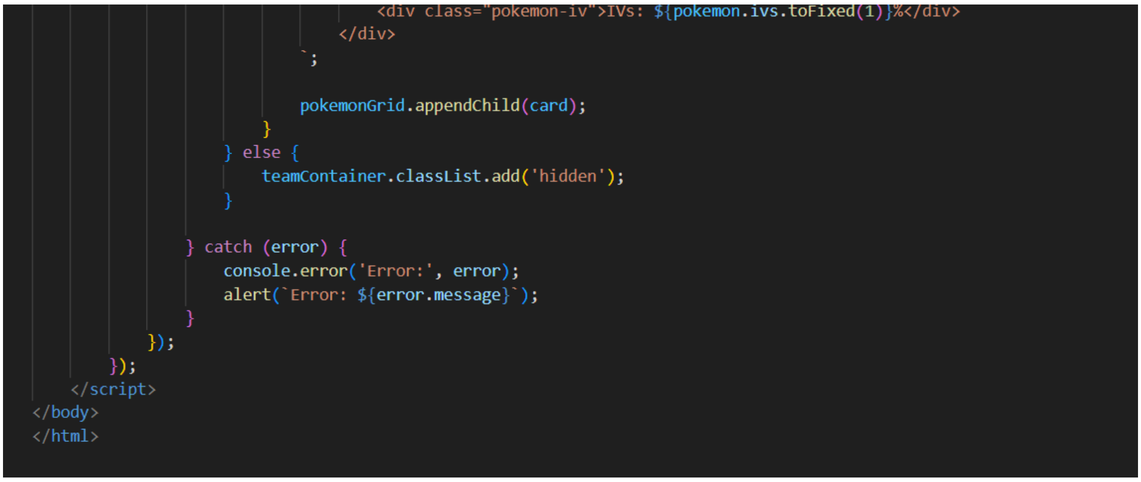
```
    <script>
        document.addEventListener('DOMContentLoaded', () => {
            analyzeButton.addEventListener('click', async () => {

                    if (!response.ok) {
                        throw new Error(`Error ${response.status}: ${response.statusText}`);
                    }

                    const data = await response.json();

                    tokensTable.innerHTML = '';
                    errorsTable.innerHTML = '';
                    pokemonGrid.innerHTML = '';

                    if (data.tokens && data.tokens.length > 0) {
                        data.tokens.forEach((token, index) => {
                            const row = tokensTable.insertRow();
                            row.insertCell(0).textContent = index + 1;
                            row.insertCell(1).textContent = token.row;
                            row.insertCell(2).textContent = token.column;
                            row.insertCell(3).textContent = token.lexeme;
                            row.insertCell(4).textContent = token.typeTokenString || token.typeToken;
                        });
                    } else {
                        const row = tokensTable.insertRow();
                        row.insertCell(0).textContent = 'No se encontraron tokens pibe';
                        row.cells[0].colSpan = 5;
                    }

                    if (data.errors && data.errors.length > 0) {
                        errorReportContainer.classList.remove('hidden');
                        data.errors.forEach((error, index) => {
                            const row = errorsTable.insertRow();
                            row.className = 'error-row';
                            row.insertCell(0).textContent = index + 1;
```



```
        document.addEventListener('DOMContentLoaded', () => {
            analyzeButton.addEventListener('click', async () => {
                            row.className = 'error-row';
                            row.insertCell(0).textContent = index + 1;
                            row.insertCell(1).textContent = error.row;
                            row.insertCell(2).textContent = error.column;
                            row.insertCell(3).textContent = error.lexeme || ' ';
                            row.insertCell(4).textContent = error.typeTokenString || 'Desconocido';
                        });
                    } else {
                        errorReportContainer.classList.add('hidden');
                    }

                    if (data.team && data.errors.length === 0) {
                        teamContainer.classList.remove('hidden');
                        teamTitle.textContent = `Entrenador: ${data.team.player}`;

                        for (const pokemon of data.team.pokemons) {
                            const pokemonData = await getPokemon(pokemon.name);

                            const card = document.createElement('div');
                            card.className = 'pokemon-card';

                            card.innerHTML = `
                                <img class="pokemon-sprite" src="${pokemonData.sprite}" alt="${pokemon.name}"
                                     onerror="this.src='https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/0.png'">
                                <div class="pokemon-name">${pokemon.name}</div>
                                <div class="pokemon-type">${pokemon.type}</div>
                                <div class="pokemon-stats">
                                    <div>Salud: ${pokemon.health}</div>
                                    <div>Ataque: ${pokemon.attack}</div>
                                    <div>Defensa: ${pokemon.defense}</div>
                                    <div class="pokemon-iv">IVs: ${pokemon.ivs.toFixed(1)}%</div>
                                </div>
```



```
                            row.insertCell(2).textContent = error.column;
                            row.insertCell(3).textContent = error.lexeme || ' ';
                            row.insertCell(4).textContent = error.typeTokenString || 'Desconocido';
                        });
                    } else {
                        errorReportContainer.classList.add('hidden');
                    }

                    if (data.team && data.errors.length === 0) {
                        teamContainer.classList.remove('hidden');
                        teamTitle.textContent = `Entrenador: ${data.team.player}`;

                        for (const pokemon of data.team.pokemons) {
                            const pokemonData = await getPokemon(pokemon.name);

                            const card = document.createElement('div');
                            card.className = 'pokemon-card';

                            card.innerHTML = `
                                <img class="pokemon-sprite" src="${pokemonData.sprite}" alt="${pokemon.name}"
                                     onerror="this.src='https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/0.png'">
                                <div class="pokemon-name">${pokemon.name}</div>
                                <div class="pokemon-type">${pokemon.type}</div>
                                <div class="pokemon-stats">
                                    <div>Salud: ${pokemon.health}</div>
                                    <div>Ataque: ${pokemon.attack}</div>
                                    <div>Defensa: ${pokemon.defense}</div>
                                    <div class="pokemon-iv">IVs: ${pokemon.ivs.toFixed(1)}%</div>
                                </div>
                            `;

                            pokemonGrid.appendChild(card);
```



```
                                    <div class="pokemon-iv">IVs: ${pokemon.ivs.toFixed(1)}%</div>
                                </div>
                            `;

                            pokemonGrid.appendChild(card);
                        }
                    } else {
                        teamContainer.classList.add('hidden');
                    }

                } catch (error) {
                    console.error('Error:', error);
                    alert(`Error: ${error.message}`);
                }
            });
        });
    </script>
</body>
</html>
```

Recomendaciones

Ahora conociendo las funcionalidades del programa, se recomienda al usuario seguir las indicaciones dadas anteriormente, con el fin de evitar un mal funcionamiento del programa que provoque el cierre inmediato del mismo.