MANUAL TECNIO– PRACTICA 1

José Antonio García Roca

Carne 202401166

Sección: “P”

Indice

Introducción

La presente aplicación ha sido desarrollada como parte de la práctica de Lenguajes Formales y de Programación, con el objetivo de implementar un analizador léxico utilizando un Autómata Finito Determinista (AFD) en TypeScript. Este manual técnico describe la estructura interna del sistema, los principios teóricos aplicados, y las decisiones de diseño empleadas para construir el analizador que procesa archivos .pklfp, identifica tokens válidos, detecta errores léxicos y permite seleccionar a los seis mejores Pokémons según sus estadísticas. También se detalla el consumo de la PokeAPI para obtener información visual de los Pokémons seleccionados y la generación de los reportes correspondientes.

Objetivos

El objetivo de este manual técnico es documentar detalladamente la lógica, estructuras y componentes utilizados en la construcción de la aplicación, con énfasis en la implementación del analizador léxico basado en un Autómata Finito Determinista (AFD). Además, busca explicar el proceso de análisis de archivos .pklfp, la selección de Pokémons según sus estadísticas, la integración con la PokeAPI para obtener sus sprites, y la generación de reportes tanto de tokens como de errores léxicos, brindando así una guía clara para comprender el funcionamiento interno del sistema.

Requisitos Técnicos

Hardware

- Procesador Intel i5 o superior
- 4GB de RAM o mas

Software

- Typescript
- Express.js
- Node.js (Versiones más recientes)
- Navegador web moderno
- Acceso a internet

Clases Utilizadas

Pokemon: Esta clase almacena el objeto Pokemon, tiene los siguientes atributos:

- nombre: Almacena el nombre del pokemon, de tipo string
- tipo: Almacena el tipo del pokemon, de tipo string
- salud: Almacena la salud del pokemon, de tipo number
- ataque: Almacena el ataque del pokemon, de tipo number
- defensa: Almacena la defensa del pokemon, de tipo number
- ivs: Almacena los ivs del pokemon, de tipo number

Jugador: Esta clase almacena el objeto jugador, tiene los siguientes atributos:

- nombre: Nombre del jugador, de tipo string
- listaPokemon: Almacena los objetos Pokemon en una lista.
- pokemonTop: Almacena los seis mejores Pokemon del jugador en una lista.

Y las siguientes funciones:

- agregarPokemon: La cual agrega un Pokemon a la listaPokemon de nuestro jugador.
- ordenarPokemon: Ordena los pokemon de mayor a menor respecto a sus ivs.
- obtenerTop: obtiene los seis mejores pokemon de la listaPokemon.

Jugadores: Esta clase guarda una lista, la cual contiene todos los jugadores creados, además se encarga de guardar los jugadores con sus respectivos pokemon, contiene las siguientes funciones:

- agregarJugador: almacena un jugador en la lista
- construirJugador: se encarga de crear al jugador, crear los pokemon y añadir estos a su respectivo jugador

Token: es la encargada de generar los objetos tipo token: tiene los siguientes atributos:

- row: Es la fila donde se encuentra el token, tipo number
- columna: Es la columna donde se encuentra el inicio del token, tipo number
- lexeme: Es el lexema del token, tipo string
- typeToken: Guarda el tipo del Token, obtiene este tipo en base a una clase enumerada.
- typeTokenString: Guarda el tipo del Token en su forma de string.

LexicalAnalyzer: Es la encargada de generar los tokens y guardar su respectiva información

Codigo de las clases: https://drive.google.com/file/d/1HHDoxC3CK1XQgrN8V1pSA5RTyi9hjUBE/view?usp=sharing

Codigo Analizador Lexico(Lexical Analyzer): https://drive.google.com/file/d/1Saw38CthBG2w8chL9rRcOODjyRY873dl/view?usp=sharing

Módulos Principales

analyze.controller.ts

Es el encargado de realizar la comunicación entre el backend y el frontend:

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

let errorList: any[] = [];

let listaJugadores: Jugador[] = [];


export const analyze = (req: Request, res: Response) => {

    const input = req.body.txtArea;
    let lexicalAnalyzer: LexicalAnalyzer = new LexicalAnalyzer();

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

    let jugadores = new Jugadores();

    jugadores.construirJugador(tokenList);
    listaJugadores = jugadores.listaJugador;

    //console.log(JSON.stringify(jugadores.listaJugador,null,2));

    res.render('pages/index',{
        tokens: tokenList,
        errors: errorList,
        codigo: input,
        contador: 0,
        listaJugadores: listaJugadores
    });
}

export const inicio = (req: Request, res: Response) => {

    res.render('pages/index',{
        tokens: [],
        errors: [],
        codigo: '',
        contador: 0,
        listaJugadores: listaJugadores
    });


}

export const reporte = (req: Request, res: Response) => {

    res.render('pages/reporte',{
        errors: errorList,
        contador: 0
    });
}

export const pokemones = (req: Request, res: Response) => {
    const nombreJugador = req.params.nombre.toLowerCase();
    const pagjugador = listaJugadores.find(j => j.nombre.replace(/"/g, '').toLowerCase() === nombreJugador);

    if(pagjugador){
        return res.render('pages/jugador',{
            pagjugador: pagjugador.nombre,
            listaPokemon: pagjugador.pokemonTop
        });
    }

    res.status(404).send("Jugador no encontrado");
}
```

analyze.route.ts

Maneja las rutas de nuestro servidor para poder usarlas

```ts
import { Router, Request, Response } from "express";
import { analyze, inicio, reporte, pokemones } from "../controllers/analyze.controller";

const analyzeRouter = Router();

analyzeRouter.get('/', inicio)
analyzeRouter.post('/analyze', analyze);
analyzeRouter.get('/report', reporte);
analyzeRouter.get('/jugadores/:nombre', pokemones);

export default analyzeRouter;
```

index.ts

Se encarga de la configuración del servidor

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

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

app.set('view engine', 'ejs');


app.use(express.urlencoded({extended: true}));
app.use(express.json());
app.use(express.static('public'));
app.use(express.text());
app.use('/', analyzeRouter);

app.listen(PORT, () => {
    console.log(`The server is running on https://localhost:${PORT}`)
});
```

Vistas

index.ejs

Codigo html, de la página inicio

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"  href="CSS/index.css">

    <title>Inicio</title>
</head>

<body>
    <div id="container">
        <div id="nav">
            <nav>
                <ul id="navPrin">
                    <li class="liNav">Pokemon USAC</li>
                <a href="/">
                    <li class="liNav">Home</li>
                </a>
                <a href="/report" target="blank">
                    <li class="liNav">Reporte Errores</li>
                </a>

                <li class="liNav" id="archivo">
                    <a href="archivo">Archivo</a>
                    <ul id="navArchivo">
                        <li id="limpiar">Limpiar Editor</li>
                        <li>
                            <label for="cargar" style="cursor:pointer;">Cargar Archivo</label>
                            <input type="file" id="cargar" accept=".pklfp" style="display: none;">
                        </li>
                        <li id="guardar">Guardar archivo</li>
                    </ul>

                </li>

                </a>
                <a href="">
                    <li class="liNav">Manual tecnico</li>
                </a>
                <a href="">
                    <li class="liNav">Manual de usuario</li>
                </a>
                </ul>
            </nav>
        </div>
        <div class="body">
            <div class="txtEdit">
                <h3>Editor de Texto</h3>
                <form action="/analyze" id="form-Analyze" method="post">
                    <textarea type="text" name="txtArea" id="txtArea" rows="30" cols="65"
                        placeholder="Escribe tu codigo aqui"><%= typeof codigo !== 'undefined' ? codigo: '' %></textarea>
                    <button type="submit" value="Enviar" onclick="abrirPestanas()">Analizar</button>
                    <script>
                        function abrirPestanas() {
                            const jugadores = <%- JSON.stringify(listaJugadores.map(j => j.nombre)) %>;
                            console.log(jugadores);

                            jugadores.forEach(nombreCrudo => {
                                const nombre = nombreCrudo.replace(/"/g, ''); // Elimina comillas dobles
                                const url = `/jugadores/${encodeURIComponent(nombre)}`;
                                window.open(url, '_blank');
                            });
                        }
                    </script>
                </form>
            </div>
            <div class="tabla">
                <table>
                    <tr id="headerTabla">
                        <th colspan="5">Tabla de tokens</th>
                    </tr>
                    <tr>
                        <th>No.</th>
                        <th>Fila</th>
                        <th>Columna</th>
                        <th>Lexema</th>
                        <th>Token</th>
                    </tr>
                    <% if (tokens && tokens.length > 0) { %>
                    <% tokens.forEach(token => { %>
                    <tr>
                        <td><%= contador = contador + 1 %></td>
                        <td><%= (token.row + 1)/2 %></td>
                        <td><%= token.column %></td>
                        <td><%= token.lexeme %></td>
                        <td><%= token.typeTokenString %></td>
                    </tr>
                    <% }) %>
                </table>
            </div>
                    <% } %>
        </div>

    </div>
    </div>
<script src="js/index.js"> </script>
</body>

</html>
```

jugador.ejs

Codigo html, de la pagina que muestra al jugador

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"  href="/CSS/index.css">
    <link rel="stylesheet"  href="/CSS/pokemon.css">

    <title>Jugador <%= pagjugador %></title>
</head>

<body>
    <div id="container">
        <div id="nav">
            <nav>
                <ul id="navPrin">
                        <li class="liNav">Pokemon USAC</li>
                    <a href="/">
                        <li class="liNav">Home</li>
                    </a>
                    <a href="">
                        <li class="liNav">Manual tecnico</li>
                    </a>
                    <a href="">
                        <li class="liNav">Manual de usuario</li>
                    </a>
                </ul>
            </nav>
        </div>
        <div>
            <h1>Equipo Pokemon de <%= pagjugador%></h1>
            <div class="container">
                <% listaPokemon.forEach((pokemon) => { %>
                    <div class="item">
                        <p> <%= pokemon.nombre %> </p>
                        <img id=<%= pokemon.nombre %> alt=<%= pokemon.nombre%>>
                        <p> <%= pokemon.tipo %> </p>
                    </div>
                <% }); %>
            </div>
        </div>
    </div>
    <script src="/js/pokemon.js"> </script>
</body>

</html>
```

reporte.ejs

Codigo html de la pagina donde se muestra el reporte de errores.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"  href="CSS/index.css">

    <title>Reporte Errores</title>
</head>

<body>
    <div id="container">
        <div id="nav">
            <nav>
                <ul id="navPrin">
                        <li class="liNav">Pokemon USAC</li>
                    <a href="/">
                        <li class="liNav">Home</li>
                    </a>
                    <a href="">
                        <li class="liNav">Manual tecnico</li>
                    </a>
                    <a href="">
                        <li class="liNav">Manual de usuario</li>
                    </a>
                </ul>
            </nav>
        </div>
        <div class="body">
            <div class="tabla">
                <table>
                    <tr id="headerTabla">
                        <th colspan="5">Tabla de tokens</th>
                    </tr>
                    <tr>
                        <th>No.</th>
                        <th>Fila</th>
                        <th>Columna</th>
                        <th>Lexema</th>
                        <th>Token</th>
                    </tr>
                    <% if (errors && errors.length > 0) { %>
                    <% errors.forEach(error => { %>
                    <tr>
                        <td><%= contador = contador + 1 %></td>
                        <td><%= (error.row + 1)/2 %></td>
                        <td><%= error.column %></td>
                        <td><%= error.lexeme %></td>
                        <td><%= error.typeTokenString %></td>
                    </tr>
                    <% }) %>
                </table>
            </div>
                    <% } %>
        </div>

    </div>
    </div>
</body>

</html>
```

Archivos JavaScript

index.js

Se encarga de las funcionalidades de la página index.ejs

```
document.addEventListener("DOMContentLoaded", () => {
    const btnLimpiar = document.getElementById("limpiar");
    const btnCargar = document.getElementById("cargar");
    const btnGuardar = document.getElementById("guardar");
    const txtArea = document.getElementById("txtArea");

    if(btnLimpiar && txtArea){
        btnLimpiar.addEventListener("click", () => {
            txtArea.value = "";
            txtArea.focus();
        });
    }

    if(btnCargar && txtArea){
        btnCargar.addEventListener("change", (event) =>{
        const file = event.target.files[0];

        if(file){
            const ruta = file.name.split('.').pop().toLowerCase();

            if(ruta !== "pklfp"){
                alert("Archivo Incorrecto, solo se permiten archivos .pkfp");
                btnCargar.value = "";
                return;
            }

            const reader = new FileReader();
            reader.onload = function (e) {
                txtArea.value = e.target.result;
                alert("Archivo Cargado Correctamente");
            };
            reader.readAsText(file);
        };
        });
    }

    if(btnGuardar && txtArea){
        btnGuardar.addEventListener("click", () => {
            const informacion = txtArea.value;

            const datos = new Blob([informacion], {type: "text/plain;charset=utf-8"});
            const url = URL.createObjectURL(datos);
            const a = document.createElement("a");


            a.href = url;
            a.download = "datos_jugadores.pklfp";
            document.body.appendChild(a);
            a.click();
            document.body.removeChild(a);
            URL.revokeObjectURL(url);
        });
    }
});
```

Función limpiar: Limpia el textarea donde se introduce nuestra información.

Funcion cargar: Abre un cuadro de dialogo para que podamos cargar nuestro archivo .pklfp y mostrarlo en nuestro textarea

Funcion guardar: Nos permite guardar la información en nuestro textarea en un archivo .pklfp

pokemon.js

Es donde nos comunicamos con nuestra API, para poder obtener los sprites de los pokemon.

```
document.addEventListener('DOMContentLoaded', () => {

    const getPokemon = async (name, img) => {
        let response = await fetch(`https://pokeapi.co/api/v2/pokemon/${name.replace(/"/g, '').toLowerCase()}`);
        let result = await response.json();

        let sprite = result.sprites.other['official-artwork'].front_default;

        img.setAttribute('src', sprite);
    }

    const imagenes = document.getElementsByTagName('img');

    for(let i = 0; i < imagenes.length; i++){

        let pokemon = imagenes[i].getAttribute('id');

        getPokemon(pokemon, imagenes[i]);

    }

});
```