UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
SEGUNDO SEMESTRE 202

C

MANUAL TÉCNICO
PROYECTO 2

Nombre: Angel Francisco Sique Santos
Carné: 202012039

INTRODUCCIÓN

El curso de Organización de Lenguajes y Compiladores 1, ha puesto en marcha un nuevo proyecto, requerido por la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería, que consiste en crear un lenguaje de programación para que los estudiantes, del curso de Introducción a la Programación y Computación 1, aprendan a programar y tener conocimiento de todas las generalidades de un lenguaje de programación. Cabe destacar, que este lenguaje será utilizado para generar sus primeras prácticas de laboratorio del curso antes mencionado.

Por lo tanto, a usted, que es estudiante del curso de Compiladores 1, se le encomienda realizar el proyecto llamado MFMScript, dado sus altos conocimientos en temas de análisis léxico, sintáctico y semántico.

REQUISITOS MÍNIMOS

500mb de disco duro

2gb de RAM

Windows 7

Equipo Intel Pentium o superior

Herramientas de análisis léxico y sintáctico: JFlex/CUP

Java 14

Graphviz

OPCIONES DEL PROGRAMA

De la parte del servidor lo primero es crear nuestras rutas que serán consumidas desde el frontend a través de peticiones get y post.

```
class Server {
    public app: express.Application;
    public port: number = 3003;

    constructor() {
        this.app = express();
        this.config();
        this.routes();
    }

    config(): void {
        this.app.set('port', process.env.PORT || this.port);
        this.app.use(cors());
        this.app.use(express.json());
        this.app.use(express.urlencoded({ extended: false }));
    }

    routes(): void {
      this.app.use("/enviar",apiRoutes);
    }      You, hace 6 días • Ordenar carpetas

    start(): void {
        this.app.listen(this.app.get('port'), () => {
            console.log('Server on port', this.app.get('port'));
        });
    }
```

En este caso tenemos 4, /errores es un get que devuelve la tabla de errores generado en el

```
You, hace 4 horas | 1 author (You)
import { Router } from "express";

import { apiController } from "../contollers/apiController";

You, hace 4 horas | 1 author (You)
class ApiRoutes {
  public router: Router = Router();

  constructor() {
    this.config();
  }

  config(): void {
    this.router.get("/errores", apiController.getErrores);
    this.router.post("/", apiController.funcion2);
    this.router.get("/ast", apiController.getAST);
    this.router.get("/simbolos", apiController.getSimbolos);      You, hace 4 ho
  }
}

const apiRoutes = new ApiRoutes();
export default apiRoutes.router;
```

análisis.

/ast nos devuelve el svg del ast generado en el análisis. / es un post donde se envía el código a ser analizado. /simbolos devuelve la tabla de símbolos generada.

```
public async funcion2(req: Request, res: Response) {

  const s = Singleton.getInstance();
  try{
    s.clear_error();
    s.clear_symbol();
    //Eliminar ../AST.svg
    const fs = require("fs");
    const path = require("path");
    const filePath = path.join(__dirname, "../AST.svg");
    fs.unlinkSync(filePath);
  }catch{

  }


  try {
    const parser = require("../Grammar/grammar.js");
    const ast = parser.parse(req.body.texto.toString());
    const env = new Env(null);
    let result = "";
    for (const instruccion of ast) {
      try {
```

El análisis comienza limpiando los posibles datos de un previo análisis generado. Luego se llama a la gramática generada con jison y se le envía el texto de la entrada oara que sea analizado.

```
*/
public add_error(data: any) {
    this.error +=
        '<tr class="bg-gray-800 border-gray-700 hover:bg-gray-600">' +
        '<th scope="row" class="py-4 px-6 font-medium whitespace-nowrap text-white">' + dat
        '<td class="py-4 px-6">' + data.descripcion + "</td>" +
        '<td class="py-4 px-6">' + data.linea + "</td>" +
        '<td class="py-4 px-6">' + data.columna + "</td>" +
        '<td class="py-4 px-6">' + data.error + "</td>" +
        "</tr>\n"
}

/**
```

Todos los errores son guardados en una variable en forma de elemento de una tabla de html.

```
public add_symbol(id: any, tipo: any, tipo2: any, entorno: any, linea: any, columna: any) {
    this.symbol +=
        '<tr class="bg-gray-800 border-gray-700 hover:bg-gray-600">' +
        '<th scope="row" class="py-4 px-6 font-medium whitespace-nowrap text-white">' + id +
        '<td class="py-4 px-6">' + tipo + "</td>" +
        '<td class="py-4 px-6">' + tipo2 + "</td>" +
        '<td class="py-4 px-6">' + entorno + "</td>" +
        '<td class="py-4 px-6">' + linea + "</td>" +
        '<td class="py-4 px-6">' + columna + "</td>" +
        "</tr>\n"
        console.log("agregado");
}
```

De la misma forma los símbolos son guardados para luego ser enviados al frontend.

El frontend es creado a base de ReactJS. En este caso se creo un html donde se movieron todos los componentes.

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=
  <meta name="theme-color" content="#000000">
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.
  <script src="https://cdn.tailwindcss.com"></script>      You, hace 7 horas • Mostr
  <title>Proyecto 2</title>
</head>

<body>
  <noscript>
    You need to enable JavaScript to run this app.
  </noscript>
  <!--  Navbar -->
  <div id="navbar"></div>
  <div id="root" class="animate__animated animate__bounceInLeft"></div>
  <!--  Router -->
  <div id="router"></div>
</body>

</html>
```

Como librerías de css se usaron tailwindcdd y animate.css. En este html se crearon 3 div uno para la barra de navegación, otro para los elementos principales del programa y otro para las rutas.

```
const bar = document.getElementById("navbar");
ReactDOM.render(
  <React.StrictMode>
    <Navbar />
  </React.StrictMode>,
  bar
);

//Router navigation
const root = document.getElementById("router");
ReactDOM.render(
  <React.StrictMode>

    <Router>
      <Routes>
        <Route path="/" element={  <App />} />
        <Route path="/about" element={<About />} />
        <Route path="/reportes" element={<Reportes />} />
      </Routes>
    </Router>
    </React.StrictMode>,
  root
);
```

Este está inicializado con dos componentes, Router que es el que nos ayuda a movernos y generar los componentes que necesitemos, Navbar que es la barra de navegación que ayuda a que se más fácil la movilización entre páginas.

```
      <input id="file" type="file" />
      <i class="fa fa-cloud-upload"></i> Cargar Archivo
  </label>
  <button  class="Abrir" onClick={openFile}>Abrir Archivo Cargado</button>
  <button  class="Agregar" onClick={handleClean}>Nuevo archivo</button>
  <button  class="Guardar" onClick={saveFile}>Guardar</button>
  </div>
  <br />
    <div class="editor">
      <Editor
        height="50vh"
        theme="vs-dark"
        width="120vh"
        defaultLanguage="javascript"
        defaultValue="// some comment"
        onMount={handleEditorDidMount}
        /* Tabs */


      />
    </div>

    <br></br>
    <div class="ejecutar">
```

En el path / tenemos el renderizado del componente App este componente genera la ventana principal con el editor y los botones de guardado y creación de archivos.

```
import React from "react";

const About = () => {
  return (
    <div class="contenido">
      <h2>Descripción General</h2>
      <div class="div-1">
      <p>
        El curso de Organización de Lenguajes y Compiladores 1, ha puesto en
        marcha un nuevo proyecto, requerido por la Escuela de Ciencias y
        Sistemas de la Facultad de Ingeniería, que consiste en crear un lenguaje
        de programación para que los estudiantes, del curso de Introducción a la
        Programación y Computación 1, aprendan a programar y tener conocimiento
        de todas las generalidades de un lenguaje de programación. Cabe
        destacar, que este lenguaje será utilizado para generar sus primeras
        prácticas de laboratorio del curso antes mencionado.
      </p>
      <p>
        Por lo tanto, a usted, que es estudiante del curso de Compiladores 1, se
        le encomienda realizar el proyecto llamado MFMScript, dado sus altos
        conocimientos en temas de análisis léxico, sintáctico y semántico.
      </p>
      </div>
      <h2>Arquitectura General del proyecto</h2>
      <div class="div-2">
```

El path /About crea el componente de About.jsx que es una página con información introductoria sobre el proyecto realizado.

```
const Home = () => {
    return (
        <>
        <div>
        <h1>Reportes</h1>
        <br />
        <br />
        </div>
        <div class="botones-div">
        <button  class="Abrir" onClick={getReporteErrores} style={{backgroundColor: "#8D9EFF"
        <button  class="Agregar" onClick={getReporteAST} style={{backgroundColor: "#BCE29E"}}
        <button  class="Guardar" onClick={getReporteSimbolos} style={{backgroundColor: "#F0EB
        </div>
        <center>
        <div id='reporte' class="reporte">
        </div>
        <br />
        <br />
        </center>


        </>
    );
    };
```

El path /Reportes crea el componente Home.jsx que contiene los botones para mostrar los reportes de errores, símbolos y ast. Cuando se da click a alguno de los botones se hace un get hacía el lado del servidor para que le envíe el respectivo reportes.

```
async function getReporteSimbolos(){
    try {
        const response = await fetch("http://localhost:3003/enviar/simbolos", {
          method: 'GET',
          headers: {}
        });
        var reporte = document.getElementById('reporte');
        if (response.ok) {
          const result = await response.json();
          console.log(result.html);

            reporte.innerHTML = result.html;
            RErrores = result.html;
        }else{

            reporte.innerHTML = RErrores;
        }
      } catch (err) {
        console.error(err);
      }
}
```

```
async function getReporteErrores(){
    try {
        const response = await fetch("http://localhost:3003/enviar/errores", {
          method: 'GET',
          headers: {}
        });
        var reporte = document.getElementById('reporte');
        if (response.ok) {
          const result = await response.json();
          console.log(result.html);

            reporte.innerHTML = result.html;
            RErrores = result.html;
        }else{

            reporte.innerHTML = RErrores;
        }
      } catch (err) {
        console.error(err);
      }
}
```

```
async function getReporteAST(){
    //Get a svg file from the server and show it in div reporte
    try {
        const response = await fetch("http://localhost:3003/enviar/ast", {
            method: 'GET',
            headers: {}
            });
            if (response.ok) {
                const result = await response.json();
                console.log(result.html);

                //save the svg in a iframe
                var iframe = document.createElement('iframe');
                iframe.setAttribute('src', 'data:image/svg+xml;utf8,' + encodeURICompor
                iframe.setAttribute('width', '100%');
                iframe.setAttribute('height', '500px');
                iframe.setAttribute('frameborder', '0');
                iframe.setAttribute('style', 'border: none;');
                iframe.setAttribute('id', 'iframe-svg');

                //show the svg in the div
                var reporte = document.getElementById('reporte');
                reporte.innerHTML = '';
                reporte.appendChild(iframe);
```

```
You, anteayer | 1 author (You)
import { Env } from "../Symbols/env";      You, hace 6 dí

You, anteayer | 1 author (You)
export abstract class Instruccion{
    //atributos
    constructor(public line: number, public column: numbe
        this.line = line;
        this.column = column;
    }

    public abstract ejecutar(Env: Env):any;
    public abstract getNodo():string;
}
```

Todas las instrucciones son derivadas de la clase abstracta Instrucción.

```
    |'expreID' '(' ')' {$$=$1;}
;

TIPOS:
    'pr_int' {$$=$1;}
    |'pr_char' {$$=$1;}
    |'pr_string' {$$=$1;}
    |'pr_bool' {$$=$1;}
    |'pr_double' {$$=$1;}
;

LISTA_EXPREID: LISTA_EXPREID ',' 'expreID' {$$=$1; $1.push(new Variable($3,@3.f
    | 'expreID' {$$= [new Variable($1,@1.first_line,@1.first_column)];}
;

VALORES:
    'expreBOOL' {$$=$1;}
    |EXPRESION {$$=$1;}
    |CASTEO {$$=$1;}
    |CONDICION {$$=$1;}
    |TERNARIO {$$=$1;}
```

Las instrucciones se crean en la gramática generada del archivo jison.