Universidad de San Carlos De Guatemala
Facultad de ingeniería
Escuela de Ciencias y Sistemas
Organización de Lenguajes y
Compiladores 1

Proyecto 2
Manual de Tecnico

Edson Saul Avila Ortiz
201902302

Tabla de contenido

Objetivos generales

Aplicar los conocimientos sobre la fase de análisis léxico y sintáctico de un compilador para la realización de un intérprete sencillo, con las funcionalidades principales para que sea funcional.

objetivos específicos

- Reforzar los conocimientos de análisis léxico y sintáctico para la creación de
- un lenguaje de programación.
- Aplicar los conceptos de compiladores para implementar el proceso de
- interpretación de código de alto nivel.
- Aplicar los conceptos de compiladores para analizar un lenguaje de
- programación y producir las salidas esperadas.
- Aplicar la teoría de compiladores para la creación de soluciones de software

requerimientos mínimos

Para el uso de la aplicación solamente se requiere tener acceso a la red en la cual se va a ejecutar.

para la ejecución de lo del servidor se necesitan las siguientes tecnologías.

tecnología utilizada

Angular

Se utilizó angular para la parte visual es decir el front end, debido a su gran versatilidad a la hora de trabajar, manejo de componentes y personalización al momento del desarrollo debido a que esta es una aplicación con fines educativos se procuró una interfaz amigable y de fácil entendimiento

Node

se utilizó node para la parte del backend la cual se encargaría de procesar ejecutar y registrar todos los procesos necesarios para el funcionamiento de la aplicación, se optó por usar esta herramienta debido a su fácil uso junto con typescript gracias a su fuerte tipado

Rest Api

se trabajaron dos servidores 1 para la parte del front end y otra para back end para un mejor orden y manejo de archivos dentro de la aplicación los puertos de los mismos son 4200 y 3000 respectivamente.

módulos back end

- Clases

  Es la parte encargada de la distribución de las distintas clases encargadas de funcionamiento del programa dentro de este módulo encontramos diversas clases como las B sentencias de control y sentencia cíclicas. cada una de las clases que se encuentren en este módulo representan una parte ejecutable dentro del programa

- Configuración

en este módulo se encuentra la configuración inicial del servidor debido a su naturalidad se optó por la elaboración de una configuración global separada de los diferentes módulos para una mejor personalización

- Controladores

dentro de esta clase se encuentra los controladores necesarios para la correcta comunicación entre servidores, Así mismo se encuentra los controladores para la comunicación entre el parser que fue generado gracias a la herramienta que hizo y el programa en sí

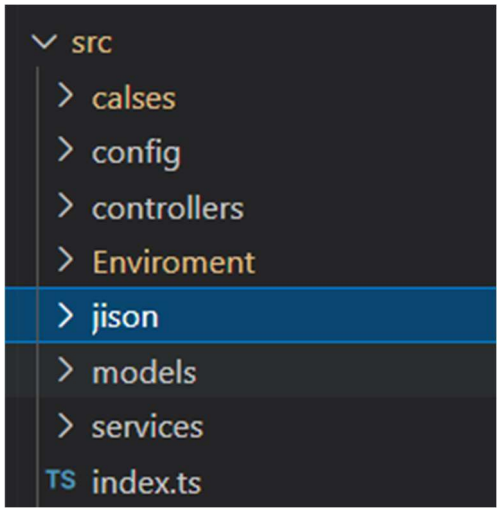
- Ambiente

dentro de este módulo se encuentran las 3 clases fundamentales para el funcionamiento del programa las cuales son:

1. Ambiente
2. Instrucción
3. símbolo

- Jison

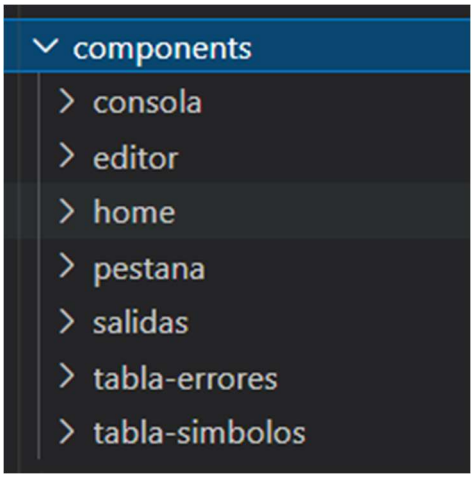
Es una es un generador de analizadores léxicos y sintácticos muy potente desarrollada para crear parsers en javascript. dentro del proyecto se optó por el uso de thaiscrypt debido a su tipado y dentro de este módulo se encuentra el archivo del cual se generó el parser.



módulos front end

- Componentes

Un componente en Angular es un bloque de código re-utilizable, que consta básicamente de 3 archivos: un CSS, un HTML (también conocido como plantilla o en inglés, template) y un TypeScript (en adelante, TS).



- Encabezado

Es el componente encargado de la navegación es decir en éste se encuentra la barra de navegación que permitirá al usuario moverse entre diferentes pestañas.

- Consola

este componente almacena un pequeño editor de texto el cual no es modificable para el usuario el objetivo de este editor de texto es poder demostrar al usuario la salida en consolas simulando como si fuera una consola de cualquier lenguaje de programación

- Editor

el editor es un componente encargado de almacenar todo el código qué será procesado por la aplicación, también se utilizó una librería para poder facilitar la visualización del mismo.

- Home

```
export class HomeComponent implements OnInit {


  constructor() {
  }

  ngOnInit(): void {

  }

}
```

- Pestañas

```
export class PestanaComponent implements OnInit {

  private regresoCompilador: any = [];
  private temporal:codeModel;
  regresoConsosla:string=''
  regresoErrores:Array<errorModel>=[];
  regresoSimbolos:Array<tablaSimbolosModel>=[];
  constructor(public dataService:VentanaService,private compi:CompilarService) {
    this.temporal={
      numeroVista: 0,
    code: '',
    console:'',
    listaE: [],
    listaSimbolos:[]
    }
  }
```

- Salidas

```
export class SalidasComponent implements OnInit {

  @Input() listaErrores:Array<errorModel>=[];
  @Input() listaSimbolos:Array<tablaSimbolosModel>=[];

  constructor(public dataService:VentanaService) {
    //this.listaErrores = this.dataService.listaVentanas[this.dataService.ventanaActual].listaE;
   }

  ngOnInit(): void {
    this.listaErrores = this.dataService.listaVentanas[this.dataService.ventanaActual].listaE;
  }

  actualizar(){
    this.listaErrores = this.dataService.listaVentanas[this.dataService.ventanaActual].listaE;
    this.listaSimbolos = this.dataService.listaVentanas[this.dataService.ventanaActual].listaSim
  }
}
```

- tabla errores

```
export class TablaErroresComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

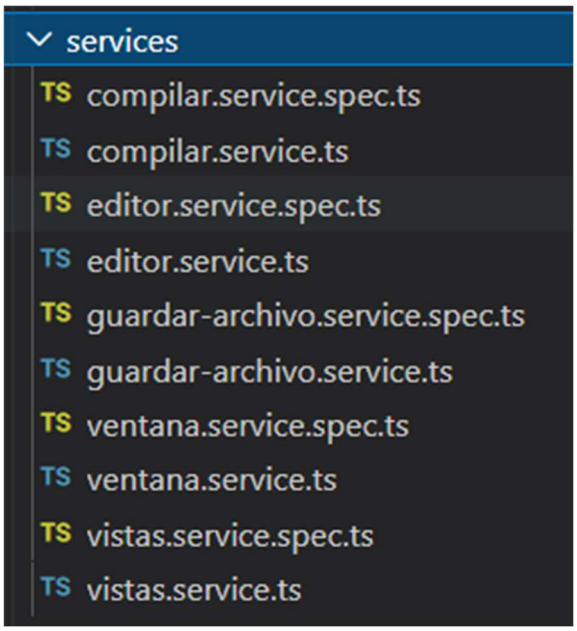
- Modelos

```
import { errorModel } from "./error-model";
import {tablaSimbolosModel} from "./simbolos-model"
export interface codeModel{
    numeroVista: number,
    code: string,
    console:string,
    listaE:Array<errorModel>,
    listaSimbolos:Array<tablaSimbolosModel>
}
```

```
export interface errorModel{
    tipoError: string,
    mensaje: string,
    fila:string,
    columna:string
}
```

```
export interface tablaSimbolosModel{
    identificador:string,
    tipo:string,
    tipo_dato:string,
    entorno:string,
    linea:string,
    columna:string,
    valor:string
}
```

- Servicios


services
compilar.service.spec.ts
compilar.service.ts
editor.service.spec.ts
editor.service.ts
guardar-archivo.service.spec.ts
guardar-archivo.service.ts
ventana.service.spec.ts
ventana.service.ts
vistas.service.spec.ts
vistas.service.ts