Universidad De San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Laboratorio organización de lenguajes y compiladores 1
Sección “C”

“MANUAL TÉCNICO”

Diego André Mazariegos Barrientos

Carné: 202003975

Objetivos

General:

Brindar al lector una guía que contenga la información del manejo de clases, atributos, métodos y del desarrollo de la interfaz gráfica para facilitar futuras actualizaciones y futuras modificaciones realizadas por terceros.

Específicos:

- Mostrar al lector una descripción lo más completa y detallada posible del SO, IDE entre otros utilizados para el desarrollo de la aplicación.

- Proporcionar al lector una concepción y explicación técnica - formal de los procesos y relaciones entre métodos y atributos que conforman la parte operativa de la aplicación.

Introducción

Este manual técnico tiene como finalidad dar a conocer al lector que pueda requerir hacer modificaciones futuras al software el desarrollo de la aplicación denominada “Compscript”. Para cumplir con el objetivo propuesto se incluye la descripción de las pantallas que el usuario manejara para el ingreso de datos, manejo de la simulación y de resultados, todo esto a través de gráficos para su mayor compresión. Desarrollada durante el transcurso de las semanas de mayo y abril del año 2022, indicando el IDE utilizado para su creación, su versión, requerimientos del sistema, etc…

La aplicación tiene el objetivo de cumplir con los requerimientos solicitados por el curso de Organización de Lenguajes y Compiladores 1, perteneciente a la Facultada de Ingeniería de la Universidad de San Carlos de Guatemala. Dichos requerimientos incluyen el análisis de los datos de entrada con un formato previamente establecido, la generación de reportes entre otras funcionalidades.

Descripción de la Solución

Para poder desarrollar este proyecto se analizó lo que el cliente solicitaba y lo que el cliente realmente necesitaba, sus restricciones tanto humanas, de equipo y financieras del proyecto y empresa; y el ambiente y forma de trabajo de los futuros operadores de la aplicación.

Entre las consideraciones encontramos con mayor prioridad están:

- Realizar la lectura del archivo de entrada con el formato correcto.

- Análisis completo del archivo de entrada una vez verificada que la entrada sea correcta lexicalmente, sintácticamente y semánticamente, procediendo a elaborar las respectivas funcionalidades dependiendo de las acciones que tome el usuario.

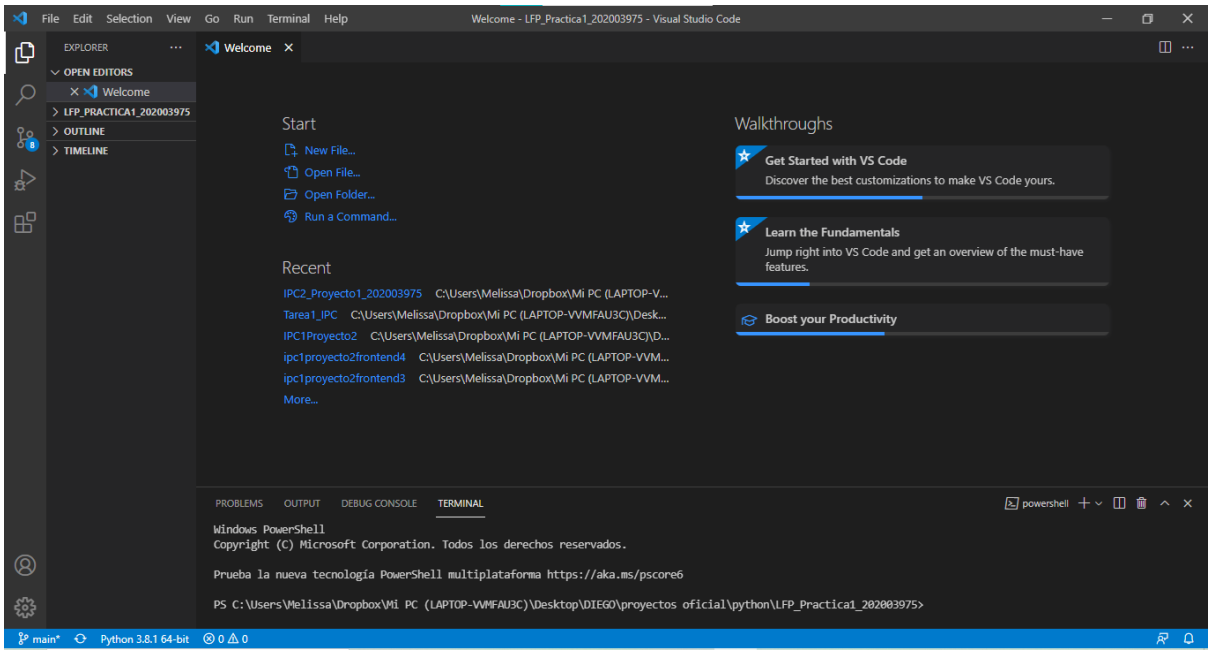
- Generación de reportes solicitados.

- Presentación de la interfaz de forma agradable y fácil de usar.

IDE

El IDE con el que se desarrolló el proyecto “Compscript” fue Visual Studio Code, debido a su apoyo al desarrollador gracias a su asistente que detecta errores semánticos, sintácticos del código por lo cual ayudan y hacen que la duración de la fase de programación sea más corta, además posee una interfaz muy agradable y fácil de entender en el modo debugging.

Requerimientos de IDE:



Hardware

Visual Studio Code es una pequeña descarga (<200 MB) y ocupa un espacio en disco de <500 MB. VS Code es liviano.

Se recomienda:

Procesador de 1,6 GHz o más rápido.

1 GB de RAM.

Software

OS X El Capitan (10.11+).

Windows 7 (con .NET Framework 4.5.2), 8.0, 8.1 y 10 (32 y 64 bits).

Linux (Debian): Ubuntu Desktop 16.04, Debian 9.

Linux (Red Hat): Red Hat Enterprise Linux 7, CentOS 8, Fedora 24.

Requisitos adicionales de Windows

Se requiere Microsoft .NET Framework 4.5.2 para VS Code. Si está utilizando Windows 7, asegúrese de que .NET Framework 4.5.2 esté instalado.

Requisitos adicionales de Linux

GLIBCXX versión 3.4.21 o posterior.

GLIBC versión 2.15 o posterior.

Requisitos del programa

Sistema operativo	Memoria RAM mínima	Memoria RAM recomendada	Espacio en disco mínimo	Espacio en disco recomendado
El programa puede ser instalado en cualquier sistema operativo.	512 MB	8 GB	4.21 MB	1 GB

Máquina en la cual fue desarrollado el programa

Especificaciones del dispositivo

HP Laptop [redacted]

Nombre del dispositivo	[redacted]
Procesador	Intel(R) Core(TM) i3-1005G1 CPU @ 1.20GHz 1.19 GHz
RAM instalada	8.00 GB (7.70 GB utilizable)
Id. del dispositivo	[redacted]
Id. del producto	[redacted]
Tipo de sistema	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil	Compatibilidad con entrada manuscrita

Copiar

Librerías Utilizadas

Las librerías utilizadas para el desarrollo de este proyecto fueron:

"cors": "versión 2.8.5",
"cross-env": "versión 7.0.3",
"express": "versión 4.17.3",
"morgan": "versión 1.10.0".

Estas librerías fueron utilizadas para realizar que el servidor funcionara y corriera en el puerto que nosotros le indicáramos básicamente son la base del backend desarrollado en el proyecto.

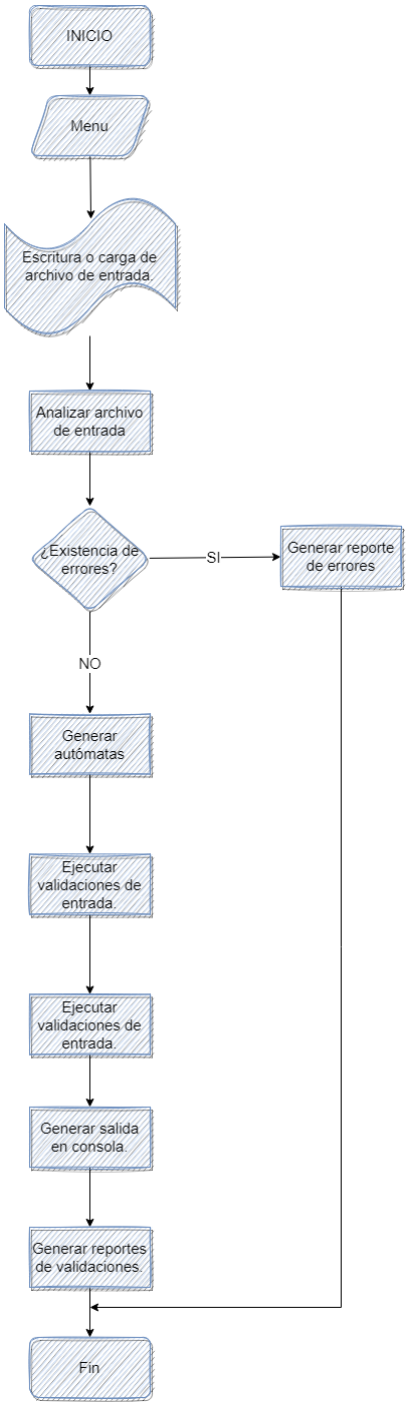
"jison": "versión 0.4.18".

Librería que se utilizó para convertir la gramática LALR en código Javascript.

"nodemon": "versión 2.0.15",
"ts-node": "versión 10.7.0",
"TypeScript": "versión 4.6.3".

Estas librerías son las que se utilizaron para poder trabajar con programación orientada a objetos ya que el proyecto era muy largo y era más fácil trabajar con el paradigma. Nodemon lo que hace es estar pendientes de cambios en el código TypeScript y pasarlo a Javascript, así como simplemente simular que se ejecuta, ts-node se encarga de convertir el código en sí y TypeScript pues es el lenguaje que no trae compilador directo en sí, pero ayuda para la detección de errores en tiempo de desarrollo.

Diagrama flujo



Clases del programa

En el programa se utilizó una gran cantidad de clases, pero entre ellas las principales son:

- **Instrucción.ts**
  Es una interfaz que es la base principal de la abstracción en el proyecto, se fundamenta en que las instrucción tienen características en común y es que poseen línea y columna en donde aparecen y además todas se ejecutan pero atención que esa ejecución no posee valor de retorno.

- **Expresión.ts**
  Es una interfaz igual de importante que la interfaz Instrucción.ts y bastante similar con la única diferencia y es que las expresiones si retornan un valor al ejecutarse.

- **Retorno.ts**
  Es una clase de tipo type que pues básicamente se utiliza para controlar los retornos así y poder verificar más fácil el valor y el tipo de los valores que retorna una expresión.

- **Environment.ts**
  Clase encargada de almacenar todas las variables que se encuentre, esta las almacena en un mapa para que sea más fácil controlar sus accesos.

- **Simbolo.ts**
  Un símbolo es todo aquella instrucción que posea un identificador único y estos se almacenan en los Environment.ts.