UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 2

SECCIÓN “B”

SEGUNDO SEMESTRE, 2023

# MANUAL TÉCNICO

Diego Andrés Huite Alvarez

202003585

08/09/2023

# Índice

# INTRODUCCIÓN

Bienvenido al manual técnico de "T-swift", un software diseñado analizar un toy-language con sintaxis similar a la de swift

## REQUERIMIENTOS

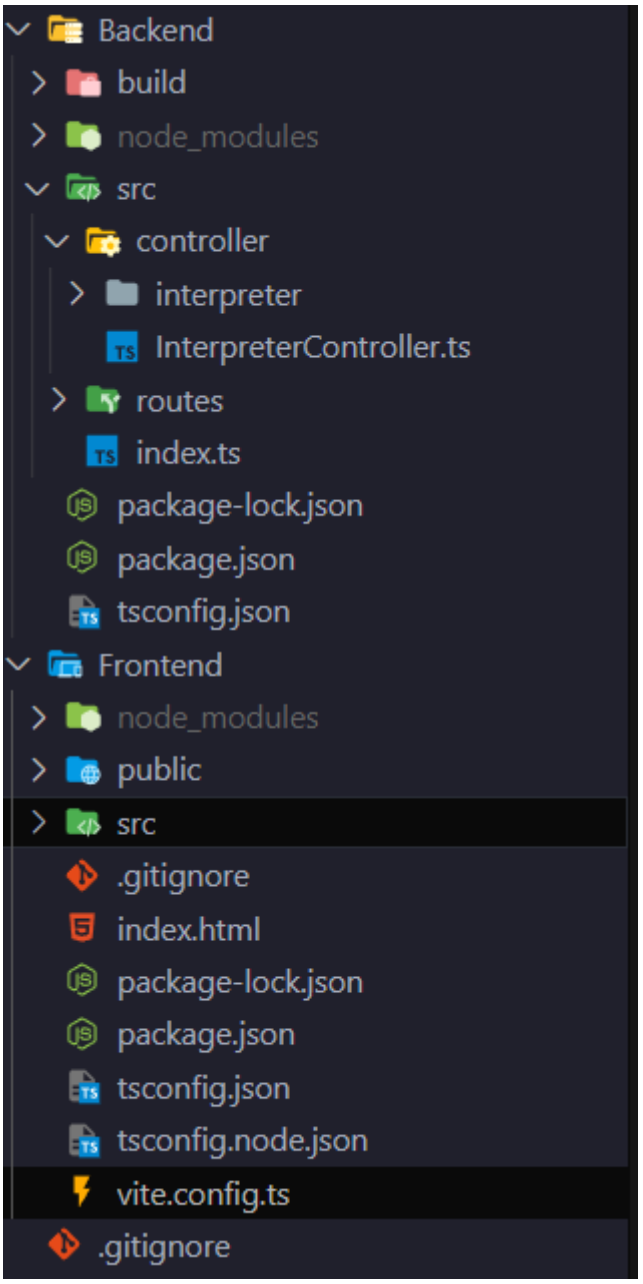
Software:

- Sistema operativo windows, linux o macOS
- Node js y cualquier instalador de paquetes
- Navegador web moderno que soporte react
- ANTLR4
- Go

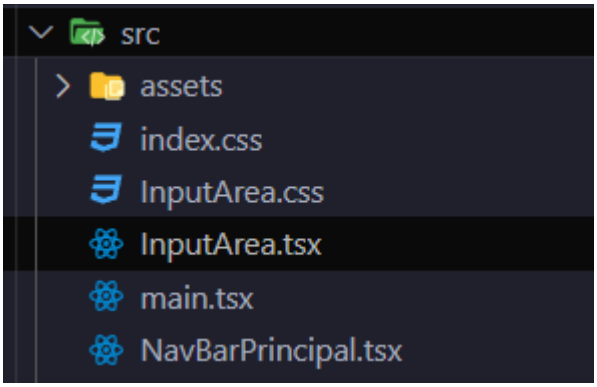
Hardware:

- Mouse
- Teclado
- Monitor
- 4gb de ram

**ESTRUCTURA DEL FRONTEND**

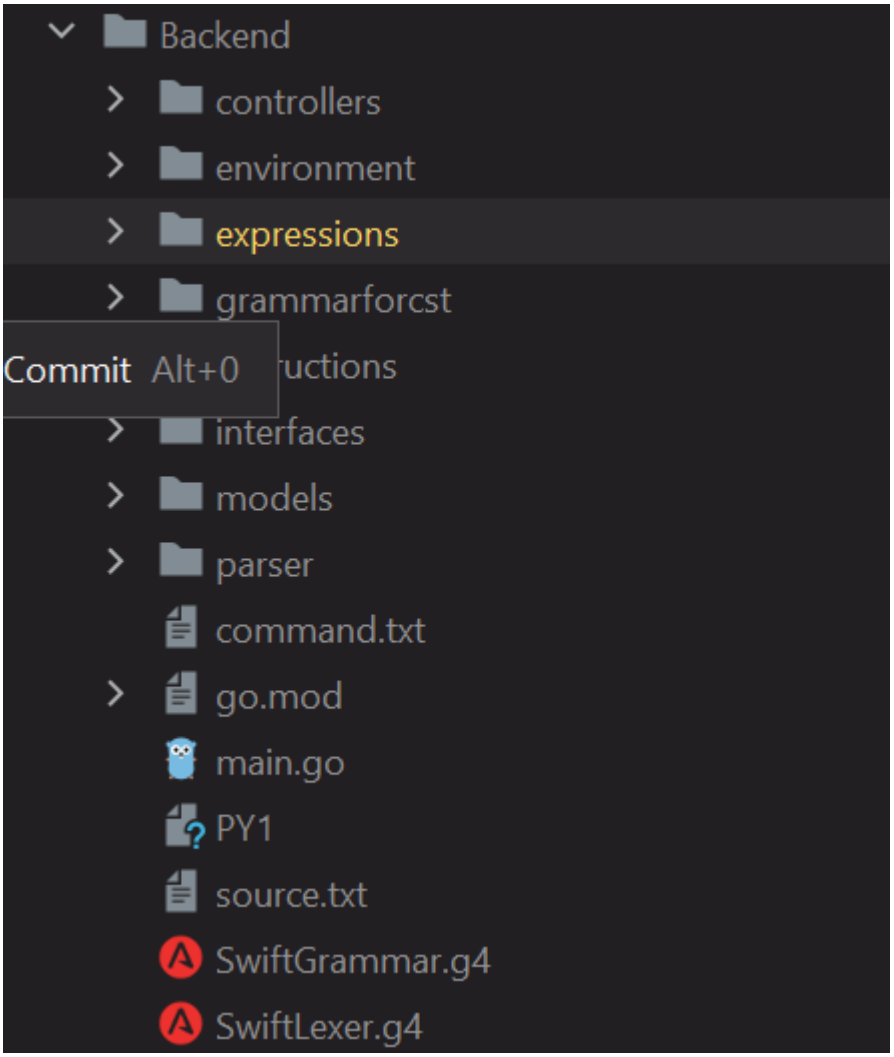



El software está conformado por dos carpetas, siendo una de estas para Frontend y la otra para el Backend. Para el Frontend se hizo uso de react + vite




Y cuenta con los componentes inputArea y NavBarPrincipal. El InputArea es el componente que se encarga de mostrar la consola en el frontend, y el NavBarPrincipal es una navbar hasta arriba de la página.

**ESTRUCTURA DEL BACKEND**




Primeramente, el AntlrController.go es el archivo encargado de recibir las peticiones al api y realizar acciones en base a ellas, también está la carpeta de rutas, que se encarga de enrutar hacia el AntlrController las rutas, esto principalmente hecho para tener un enrutamiento limpio.

**Enviroment:** Se encarga de representar un entorno en el lenguaje en desarrollo, así como sus símbolos

**expressions: Se** encarga de representar cada una de las expresiones del lenguaje, contiene expresiones para realizar operaciones aritméticas, booleanas, relacionales y demás

**Grammarforcst:** guarda una gramática de copia para que antlr4 tools pueda generar el cst correctamente.

**instructions: Se** encarga de representar cada una de las instrucciones del lenguaje, contiene instrucciones como print, ifs, fors, etc

## ANALISIS LEXICO Y SINTACTICO

Para realizar estos dos tipos de análisis en el lenguaje se hizo uso de la herramienta “ANTLR4”, una herramienta para generar un scanner y un parser en base a las expresiones regulares que establezcamos y en baste también al conjunto de producciones que establezcamos.

Vista previa del analizador léxico:

```
lexer grammar SwiftLexer;

// --------------- Tokens
// types
RINT: 'Int';
RFLOAT: 'Float';
RBOOL: 'Bool';
RSTRING: 'String';
RCHARACTER: 'Character';

// reserved words
RTRUE: 'true';
RFALSE: 'false';
RPRINT: 'print';
RIF: 'if';
RELSE: 'else';
```

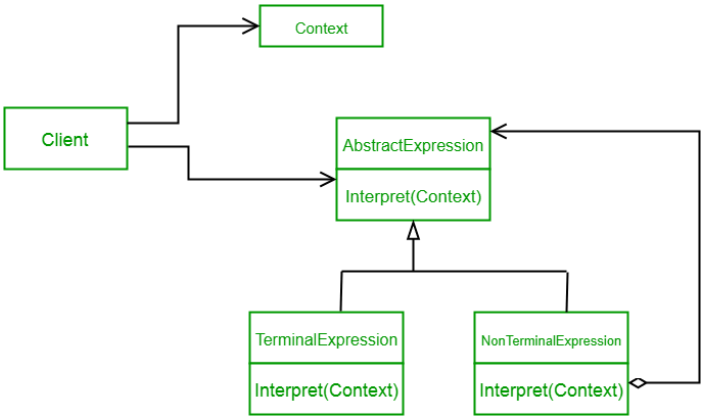
Vista previa analizador sintáctico:

```
| structmodification PTOCOMA? {$inst = $structmodification.newstructmod}
| guardstatement  {$inst = $guardstatement.newguard}
;


// INSTRUCTIONS

structfuncall returns [interfaces.Instruction newstructfunccall]
: id1=ID PTO ID  PARIZQ PARDER  {$newstructfunccall = instructions.NewStructFuncCall(
;
```

**PATRÓN INTERPRETE**




Se hizo uso del patrón interprete para poder llevar el control y flujo de cada uno de las sentencias y expresiones del lenguaje

En este caso tenemos dos structs simulando clases abstractas, una para instrucción y otra para las expresiones.En la carpeta Expresiones tenemos cada una de las expresiones