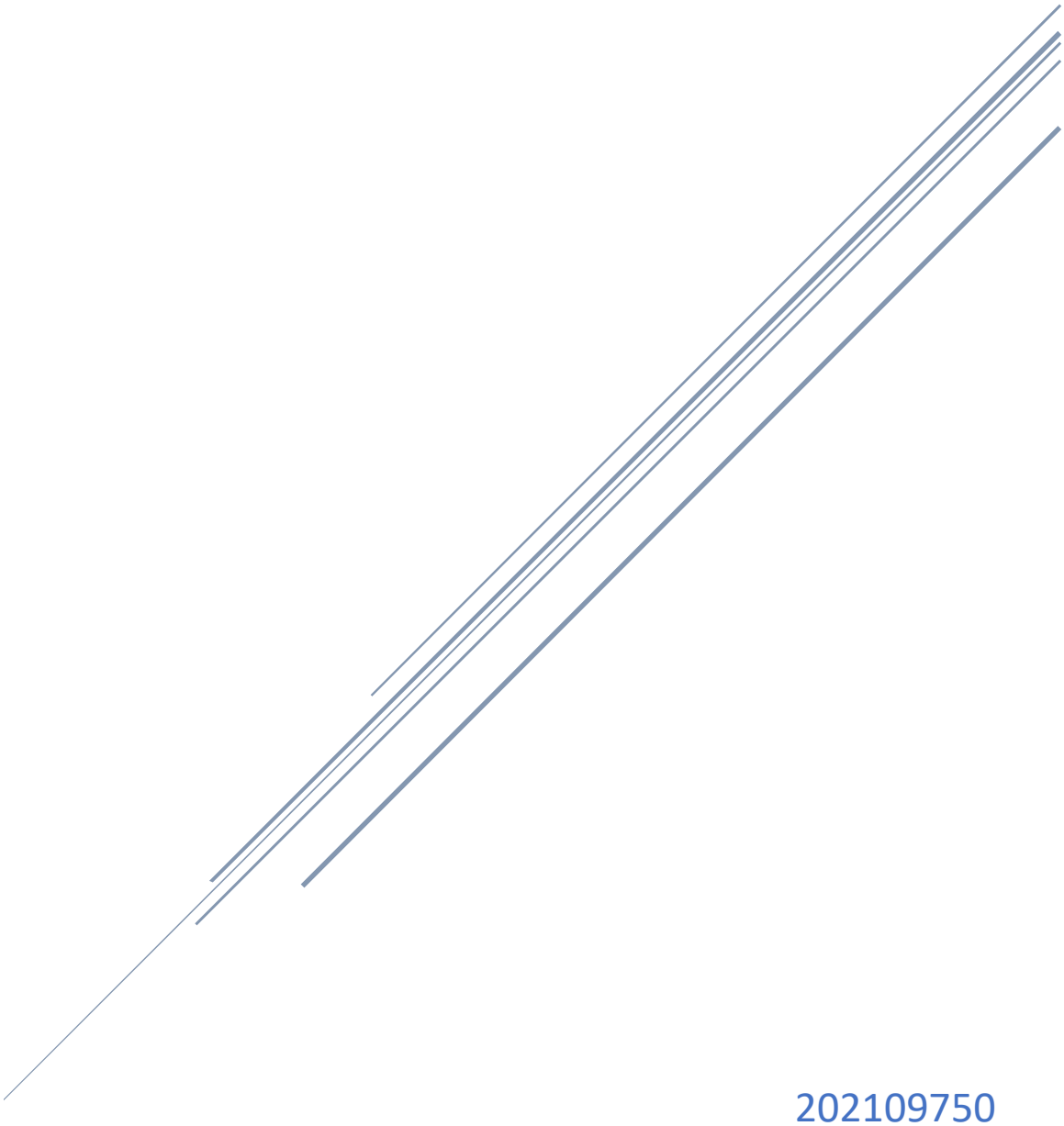
# MANUAL TÉCNICO

Luis Antonio Castro Padilla



202109750

24/04/2024

Index.ts

El archivo index.ts contiene información vital para el flujo del programa, pues aquí se detallan los puertos que usará el programa para su ejecución, así como brindar las urls que usará el programa, además hará el llamado para poder interpretar el código recibido en el área de texto del editor.

```
import { Request, Response } from "express"

const parser = require("../Analizadores/Gramatica")

function interprete(contenido:string){
    try {
        const ast = parser.parse(contenido)
        ast.Ejecutar()
        console.log("Análisis finalizado 2")
        return ast.getConsola()
    } catch (error) {
       console.error(error)
    }

}

const express = require('express')
const cors = require('cors')
const app = express()
const port = 3000

app.use(cors())
app.use(express.json())

app.post('/interpretar', (req:Request, res:Response) => {
  const contenido = req.body.contenido
  const interpretado = interprete(contenido)
  res.json({resultado:interpretado})
})

app.get('/', (req:Request, res:Response) => {
    res.send("Hola mundo")
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

La Clase AST creará el árbol abstracto sintáctico, que será recorrido y de él se obtendrán todos los nodos hojas, hijos, padres y nietos para crear el árbol sintáctico, necesario para el análisis semántico.

```
export class AST {
    public instrucciones: Instruccion[]
    public consola:string[]
    public contextoGlobal:Contexto
    public contadorExec: number
    constructor(instrucciones: Instruccion[]){
        this.instrucciones = instrucciones
        this.consola = []
        this.contextoGlobal = new Contexto(null)
        this.contadorExec = 0;
    }

    public Ejecutar(){
        // Primera pasada
        this.instrucciones.forEach(instruccion => {
            if (instruccion instanceof Funcion
                || instruccion instanceof Declaracion
            ){
                instruccion.interpretar(this.contextoGlobal,this.consola)
            }
       });
     // Segunda pasada
     this.instrucciones.forEach(instruccion => {
        if(instruccion instanceof Execute){
            if(this.contadorExec>0) {
            const print = new Print(new Primitivo("Ya existe una función
Exec",TipoDato.STRING,0,0),true,0,0)
            print.interpretar(this.contextoGlobal,this.consola)
            return
            }
            instruccion.interpretar(this.contextoGlobal,this.consola)
            this.contadorExec++
        }else if (!(instruccion instanceof Funcion)
            && instruccion instanceof Declaracion){
            const print = new Print(new Primitivo("Esta instruccion debe
estar en una función",TipoDato.STRING,0,0),true,0,0)
            print.interpretar(this.contextoGlobal,this.consola)
        }
   });
```

```
    }
    public getConsola(){
        console.log(this.consola)
        let salid = ""
        for (let index = 0; index < this.consola.length; index++) {
            salid += this.consola[index].toString();
        }
        console.log(salid)
        return salid

    }
}
```

Expresión.ts: Esta será una clase abstracta que tendrá como base devolver un resultado, es una parte cruda de código, únicamente tomará en cuenta el contexto.

```
export abstract class  Expresion{
    public line: number;
    public column: number;
   constructor(line: number, colum:number){
        this.line = line;
        this.column = colum;
    }
   public abstract interpretar(contexto:Contexto):Resultado

 }
```

Instrucción.ts: Esta también será una clase abstracta, como distinción recibe como parámetro consola, que será un arreglo de tipo string que ayudará a recorrer los datos almacenados en la consola para así poder usar la consola de salida.

```
export abstract class Instruccion{
    public line: number;
    public column: number;
    // Esta clase siempre pedirá línea y columna
    constructor(line: number, colum:number){
        this.line = line;
        this.column = colum;
    }
    // Método que siempre debe ejecutarse en todos los objetos que hereda
    public abstract interpretar(contexto:Contexto,consola:string[]):null |
string

 }
```

JavaScript: Es un lenguaje de programación de alto nivel, interpretado por los navegadores web para brindar funcionalidades interactivas a las páginas web. JavaScript es multiplataforma y se utiliza tanto en el desarrollo del lado del cliente como del servidor.

TypeScript: Es un superconjunto de JavaScript desarrollado por Microsoft. Añade tipado estático opcional y otros características a JavaScript, lo que facilita la detección temprana de errores y el mantenimiento de código a gran escala. TypeScript se compila a JavaScript estándar para ejecutarse en cualquier entorno que admita JavaScript.

Node.js: Es un entorno de ejecución de JavaScript del lado del servidor, construido sobre el motor V8 de Google Chrome. Node.js permite ejecutar código JavaScript fuera del navegador, lo que lo hace adecuado para el desarrollo de aplicaciones web del lado del servidor y herramientas de línea de comandos.

Jison: Es un generador de analizadores sintácticos para JavaScript inspirado en Bison. Permite definir gramáticas formales para analizar texto y generar analizadores léxicos y sintácticos en JavaScript. Jison se usa comúnmente en el desarrollo de compiladores, intérpretes y otras herramientas relacionadas con el procesamiento de lenguajes de programación.

React: Es una biblioteca de JavaScript de código abierto desarrollada por Facebook para construir interfaces de usuario interactivas y de una sola página (SPA). React utiliza un enfoque basado en componentes, lo que significa que las interfaces de usuario se dividen en componentes reutilizables y autónomos que gestionan su propio estado y renderizado.