

MANUAL TÉCNICO

ANALIZADOR SINTÁCTICO: PRACTICA 2 DE LENGUAJES FORMALES Y DE PROGRAMACIÓN

BREVE DESCRIPCIÓN

El presente proyecto, es una simple implementación de un analizador sintactico (parcial) que simula el comportamiento del lenguaje de programación Python.

Se ha dividido en *frontend* y *backend*, donde, en el backend se ha utilizado Java con el apoyo de librerías como JakartaEE para crear una API que pueda servir el funcionamiento de este analizador a clientes, en arquitectura cliente-servidor.

Para el frontend se ha hecho una simple interfaz con ayuda del framework de JavaScript *Vue.js*.

HERRAMIENTAS UTILIZADAS

● LENGUAJE DE PROGRAMACIÓN JAVA - VERSIÓN 17 (BACKEND)

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.

El lenguaje Java proporciona:

- El paradigma de la programación orientada a objetos.
- Ejecución de un mismo programa en múltiples sistemas operativos y plataformas.
- Es posible utilizarlo para múltiples propósitos, desde aplicaciones de escritorio hasta en servidores web.
- Tiene una curva de aprendizaje media pero también toma lo mejor de otros lenguajes orientados a objetos, como C++.

También se utilizaron librerías/bibliotecas externas como:

- **Jakarta EE** - Servicios web
- **Apache Tomcat** - Servidor web
- **Graphviz-java** - Para gráficos (en este caso no se provee ninguna funcionalidad de gráficos, solo está incluido dentro del código fuente)
- **Lombok** - Para utilidades puramente de desarrollo que faciliten el mismo

● **FRAMEWORK DE JAVASCRIPT - VUE.JS v3 (FRONTEND)**

Vuejs es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página. Fue creado por Evan You, y es mantenido por él y por el resto de los miembros activos del equipo central que provienen de diversas empresas como Netlify y Netguru.

Vue.js cuenta con una arquitectura de adaptación gradual que se centra en la representación declarativa y la composición de componentes. La biblioteca central se centra sólo en la capa de vista. Las características avanzadas necesarias para aplicaciones complejas como el enrutamiento, la gestión de estados y las herramientas de construcción se ofrecen a través de librerías y paquetes de apoyo mantenidos oficialmente, con Next.js como una de las soluciones más populares.

Vue.js permite extender el HTML con atributos HTML llamados directivas. Las directivas ofrecen funcionalidad a las aplicaciones HTML, y vienen como directivas incorporadas o definidas por el usuario.

● **IDE: INTELLIJ IDEA ULTIMATE**

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) escrito en Java para desarrollar software informático escrito en Java, Kotlin, Groovy y otros lenguajes basados en la JVM. Está desarrollado por JetBrains (anteriormente conocido como IntelliJ) y está disponible como una edición comunitaria con licencia de Apache 2, y en una edición comercial patentada. Ambos se pueden utilizar para el desarrollo comercial.

● **GITHUB**

Es un sistema de control de versiones de código y gestión de proyectos, a su vez también funciona como una plataforma de estilo red social diseñada para desarrolladores para poder compartir código entre más personas y colaborar en el mismo. Se utilizó GitFlow como metodología

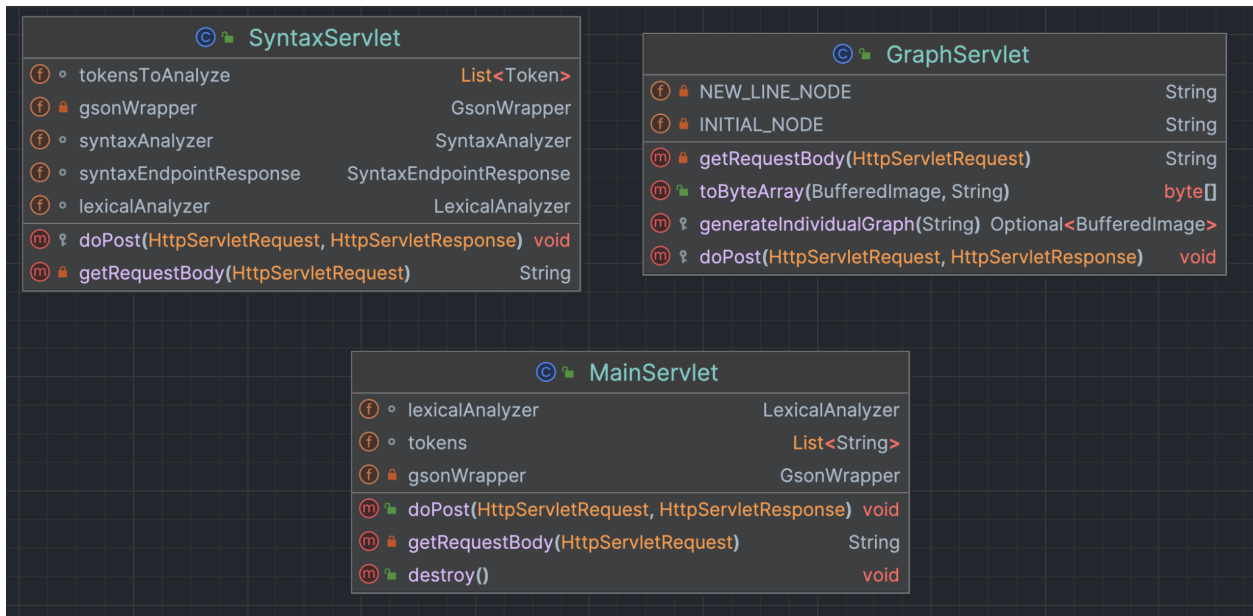
● **MACOS v13 - SISTEMA OPERATIVO**

macOS (previamente Mac OS X, luego OS X) es una serie de sistemas operativos gráficos desarrollados y comercializados por Apple desde 2001. Es el sistema operativo principal para la familia de computadoras Mac de Apple. Dentro del mercado de computadoras de escritorio, portátiles, hogareñas y mediante el uso de la web.

macOS se basa en tecnologías desarrolladas entre 1985 y 1997 en NeXT. Se logró la certificación UNIX 03 para la versión Intel de Mac OS X 10.5 Leopard y todos los lanzamientos de Mac OS X 10.6 Snow Leopard hasta la versión actual también tienen la certificación UNIX 03. macOS comparte su núcleo basado en Unix, llamado Darwin, y muchos de sus frameworks con iOS 16, tvOS y watchOS.

DIAGRAMA DE CLASES

Paquete: Controllers






































Paquete: Models

```

    @ (f) foundTokens List<Token>
    @ (f) currentColumn int
    @ (f) currentLine int
    @ (f) symbolsTable Map<String, TokenType>

    @ (m) getFoundTokens() List<Token>
    @ (m) createToken(String) void
    @ (m) findCodeTokens(String) void

```

©  SyntaxAnalyzer		
 	symbolTable	SyntaxSymbolTable
 	methodCallsTable	MethodCallsTable
 	errorsTable	ErrorsTable
 	tokens	List<Token>
 	currentIndex	int
<hr/>		
 	getSymbolTable()	SyntaxSymbolTable
 	getErrorsTable()	ErrorsTable
 	getMethodCallsTable()	MethodCallsTable
 	assignment()	boolean
 	arguments()	boolean
 	analyzeTokens(List<Token>)	void
 	methodCall()	boolean
 	tokenIsNotCommentOrNewLine(int)	boolean
 	parse()	void
 	expression()	boolean
 	match(TokenType)	boolean
 	term()	boolean

Token		
f	lexeme	String
f	column	int
f	type	TokenType
f	line	int
f	pattern	String
m	getPattern()	String
m	getLexeme()	String
m	getType()	TokenType
m	getLine()	int
m	getColumn()	int
m	setLexeme(String)	void
m	setPattern(String)	void
m	setType(TokenType)	void
m	setLine(int)	void
m	setColumn(int)	void

Paquete: Models/Syntaxis

MethodCallTableItem

methodNameString

timesCalledint

lastCalledAtLineint

getMethodName()String

getTimesCalled()int

getLastCalledAtLine()int

setMethodName(String)void

setTimesCalled(int)void

setLastCalledAtLine(int)void

equals(Object)boolean

canEqual(Object)boolean

hashCode()int

toString()String

incrementTimesCalled(int)void

SyntaxEndpointResponse

tokensFoundList<Token>

syntaxSymbolTableSyntaxSymbolTable

errorsTableErrorsTable

methodCallsTableMethodCallsTable

setMethodCallsTable(MethodCallsTable)void

getTokensFound()List<Token>

getSyntaxSymbolTable()SyntaxSymbolTable

getErrorsTable()ErrorsTable

getMethodCallsTable()MethodCallsTable

setTokensFound(List<Token>)void

setSyntaxSymbolTable(SyntaxSymbolTable)void

setErrorsTable(ErrorsTable)void

equals(Object)boolean

toString()String

canEqual(Object)boolean

hashCode()int

SymbolTableItem

lineint

typeSymbolType

symbolString

columnint

valueString

getValue()String

getSymbol()String

canEqual(Object)boolean

getType()SymbolType

getLine()int

getColumn()int

setSymbol(String)void

setType(SymbolType)void

setValue(String)void

hashCode()int

setLine(int)void

setColumn(int)void

equals(Object)boolean

toString()String

SyntaxSymbolTable

symbolTableItemsList<SymbolTableItem>

getSymbolTableItems()List<SymbolTableItem>

setSymbolTableItems(List<SymbolTableItem>)void

equals(Object)boolean

canEqual(Object)boolean

hashCode()int

toString()String

addSymbolTableItem(SymbolTableItem)void

SymbolType

METHOD_CALL

VARIABLE

METHOD

valueOf(String)SymbolType

values()SymbolType[]

MethodCallsTable

itemsList<MethodCallTableItem>

totalMethodsCalledint

registerItem(String, int)void

addMethodCallTableItem(MethodCallTableItem)void

incrementMethodCall(String, int)void

ErrorsTable

errorsTableItemsList<ErrorTableItem>

addErrorTableItem(ErrorTableItem)void

removeErrorTableItem(int, int)void

ErrorTableItem

errorString

lineint

columnint

getError()String

getLine()int

getColumn()int

setError(String)void

setLine(int)void

setColumn(int)void






equals(Object)boolean

canEqual(Object)boolean










































hashCode()int










toString()String














Paquete: Filters






©  CORSFilter		
(m)  	init(FilterConfig)	void
(m)  	doFilter(ServletRequest, ServletResponse, FilterChain)	void

Paquete: Utils

©  TokenType		
(f)  	KEYWORD_WHILE	
(f)  	LOGIC_TRUE	
(f)  	LITERAL	
(f)  	LOGIC_FALSE	
(f)  	KEYWORD_FINALLY	
(f)  	KEYWORD_RAISE	
(f)  	KEYWORD_DEL	
(f)  	PUNCTUATION_PARENTHESIS_OPEN	
(f)  	KEYWORD_EXCEPT	
(f)  	NEWLINE	
(f)  	KEYWORD_NONE	
(f)  	NEW_LINE	
(f)  	STRING	
(f)  	KEYWORD_DEF	
(f)  	KEYWORD_GLOBAL	
(f)  	OPERATOR_ASSIGNMENT	
(f)  	NUMERIC_CONSTANT_DECIMAL	
(f)  	KEYWORD_CONTINUE	
(f)  	KEYWORD_ASSERT	
(f)  	SPACE	

©  DefinitionsUtils		
(f)  	alphabet	String
(f)  	patternsMap	Map<TokenType, String>
(f)  	keywordsPatternsMap	Map<String, String>
(m)  	fillSymbolsTableAsStringType(Map<String, TokenType>)	void

©  TokenUtils		
(m)  	isParenthesesBalanced(String)	boolean
(m)  	isValidIdentifier(String, Map<String, TokenType>)	boolean
(m)  	containsLettersOrUnderscores(String)	boolean
(m)  	isInAlphabet(char)	boolean
(m)  	isLogicOperator(String)	boolean
(m)  	isValidLexeme(String)	boolean

©  GsonWrapper		
(f)  	gson	Gson
(m)  	sendAsJson(HttpServletResponse, Object)	void

CLASES Y ALGORITMOS

Modelos:

Las clases dentro del paquete de modelos, representan a las entidades que interactúan en el proceso del análisis léxico de este programa.

Por un lado está la entidad Token, la cual se encarga de albergar todas las propiedades que un token necesita para ser identificado y guardado dentro del registro del programa. Y por otro lado, está el modelo del analizador léxico en sí y el que contiene toda la lógica necesaria para el procesamiento del texto de entrada para identificar los tokens dentro del código.

Por el lado del analizador sintáctico, se tiene definido primero, el modelo del analizador como tal, que contiene toda la lógica necesaria para el procesamiento de los tokens generados previamente, y luego están los modelos para representar la información que se devuelve al hacer el análisis sintáctico, como la tabla de símbolos, llamadas de métodos y el formato de la respuesta HTTP que devuelve toda esta información en conjunto.

Controladores:

Los controladores (en este caso Servlets, de la librería JakartaEE) se encargan de la interacción entre backend y frontend, ya que estos crean una API con tres endpoints definidos (`/analyze`, `/graph`, `/syntax`) a los que se pueden realizar peticiones con los parámetros necesarios de cada caso. En el caso del frontend comunicándose con el backend, los controladores se encargan de procesar la solicitud y sus parámetros para poder identificar la información necesaria para realizar el análisis del código que se proporcione, y en el caso contrario, backend a frontend, éstos se encargan de procesar los datos provenientes de la lógica del analizador y proporcionarlos hacia el frontend en un formato determinado.

Utilidades:

En este paquete, están algunas utilidades definidas, que son comúnmente usadas entre varias clases y se han extraído en diferentes clases para evitar la repetición del código.

Componentes de interfaz gráfica:

Debido a que la interfaz gráfica de este proyecto ha sido realizada como una parte separada del programa encargado del análisis como tal, su estructura depende de las tecnologías que sean utilizadas para su construcción.

En el caso del framework Vue.js, las interfaces gráficas deben irse construyendo en forma de componentes declarativos reutilizables, definidos en un formato específico de Vue.js el cual son los Componentes de Un solo archivo (SFC, single file components desde el inglés) los cuales encapsulan en un solo formato/archivo, la estructura HTML de un componente, sus estilos con CSS y su lógica e interactividad con JavaScript.

INSTRUCCIONES PARA EJECUCIÓN

A continuación, se presentan los pasos necesarios para poner en funcionamiento el proyecto, el cual está dividido en los componentes de Backend y Frontend:

Backend:

- Abrir el proyecto Backend utilizando la IDE de preferencia.
- Descargar localmente el servidor web Apache Tomcat (su ejecutable)
- Verificar que la IDE sea compatible con la tecnología Maven para Java.
- Localizar el archivo de configuración 'pom.xml' en el interior del proyecto.
- Iniciar el proyecto empleando las herramientas proporcionadas por la IDE o a través de los comandos correspondientes de Maven.
- El componente Backend quedará activo y listo para procesar las solicitudes que se reciban.

Frontend:

- Asegurarse de tener Node.js instalado en el sistema. Si aún no está instalado, se puede descargar e instalar desde la página oficial nodejs.org.
- Abrir una terminal en la ubicación del proyecto Frontend.
- Ejecutar el comando `npm install` para instalar las dependencias requeridas para el proyecto.
- Una vez que finalice la instalación, utilizar el comando `npm run serve` para iniciar el servidor de desarrollo de Vue.js.
- Abrir el navegador web y acceder a la dirección proporcionada por el servidor de desarrollo.
- En este punto, el componente Frontend estará en funcionamiento, permitiendo la interacción con el proyecto a través del navegador.
- Siguiendo estos pasos, se podrá poner en marcha tanto el componente Backend como el Frontend del proyecto de manera efectiva. Estas instrucciones simplificadas serán de utilidad para llevar a cabo la implementación con éxito.