

Universidad Rafael Landívar
Facultad de Ingeniería
Ingeniería en Informática y Sistemas
Compiladores Sección 1
Ing. Pedro David Gabriel Wong

Documentación Primer Proyecto

Mario Aaron López Joaquín – 2477123
Walter José Herrera Salvatierra – 1109722
Jonathan Josué García González – 1045222
Giancarlo Ortega Bonilla – 1230222

Guatemala, 19 de marzo de 2025

Lógica del Analizador Léxico

Para este proyecto se puso en práctica todo lo aprendido en la clase teórica para crear un nuevo lenguaje de programación que va a ir orientado hacia programadores de Latinoamérica, donde se tomará en cuenta un lenguaje que no es sensible a mayúsculas y donde sus funciones por defecto y tipos de datos se definen en idioma español.

Por ello se implementó una lógica donde se considerarán todos los posibles patrones para crear los respectivos tokens que retornará nuestro analizador léxico. Se estableció que el código de entrada contendrá la siguiente distribución de tokens: identificadores que harán referencia a cada posible variable que se utilice en el código, números enteros, números decimales, palabras reservadas que son definidas ya por el lenguaje para declaración de variables y funciones predeterminadas, operadores aritméticos, operadores lógicos, comentarios que pueden ser de una línea o de múltiple línea, signos de agrupación y caracteres especiales que cumplen una función específica como el ; que identifica el fin de una instrucción específica.

El programa leerá todo el código fuente y retornará una tabla de símbolos que contendrá el tipo de token, su valor, la fila en que se encuentra dentro del código y su columna dentro de la misma fila. Todo lo que no se especifique dentro de los tokens será identificado como error y se mostrará en la salida del analizador léxico su valor, la fila en que se encuentra al igual que el número de columna.

Ya con todos los tokens establecidos correctamente nos aseguramos de que el código fuente contenga una entrada con todos los patrones correctos para poder trasladarnos a el analizador sintáctico que verificará que el orden de definición sea el correcto. Y los errores nos ayudarán a identificar las fallas del código fuente para poder corregirlas y evitar futuros inconvenientes al trasladar todos los tokens al analizador sintáctico.

Expresiones Regulares Utilizadas

Las expresiones regulares nos ayudan a que nuestro programa identifique los patrones de cada entrada que se identifica al momento de realizar la ejecución. Es importante definirlas correctamente y asociarlas con su token respectivo porque de lo contrario no considerará todos los posibles escenarios de entrada que el lenguaje puede contener.

- Identificadores: `[a-zA-Z_][a-zA-Z0-9_]*`
- Números Enteros: `-?\d+`
- Números Reales: `-?\d+(\.\d+)?`
- Caracteres: `([^\|\|\\]|\\|\\.)`
- Cadenas de Texto: `"\"[^\"]*"\\\"`
- Comentarios de una línea: `//.*?(?:\\n|$)`
- Comentarios de múltiples líneas: `\/\^[^]**+(?:\/\^[^]**+)*\/`
- Operadores y símbolos especiales: `==|>|=|<|=|\\|\\|&&|\\+|\\-|\\^|\\#|=|<|>|!|();|{|},|`

Listado de Palabras Reservadas

Las palabras reservadas ya vienen definidas dentro del propio lenguaje. Ayudan al programador a definir los tipos de datos existentes para toda variable que creen, llamar a todas las funciones por defecto dentro del lenguaje y realizar sentencias condicionales y cíclicas.

Las palabras definidas en el lenguaje son las siguientes y las agruparemos por su funcionalidad dentro del código:

- Definir tipos de datos: Entero, Real, Booleano, Caracter, Cadena.
- Funciones ya definidas dentro del lenguaje: EscribirLinea, Escribir, Longitud, aCadena.
- Sentencias de Control: If, Else, Else if
- Sentencias Cíclicas: For, Do, While

Librerías y Dependencias

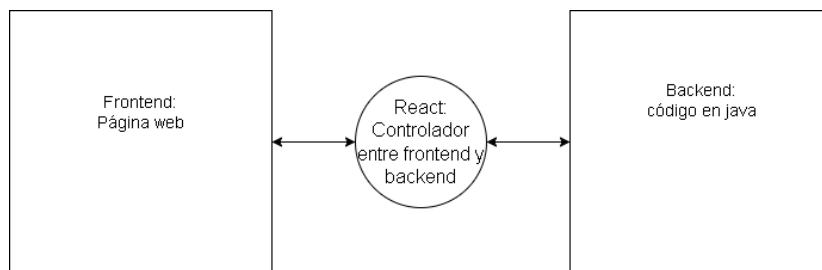
Para la definición de los tokens existentes el lenguaje se tuvo que implementar la librería regex que viene dentro del lenguaje java. A través de ella se definieron todas las expresiones regulares para cada tipo de token existente en el código fuente.

Además, al ser este un proyecto que se realizó una página web como frontend se utilizó el framework react para poder conectar todo lo ingresado desde la página web con la definición de los tokens que está bajo un archivo en lenguaje java.

Arquitectura de la Solución

Para la solución del proyecto se utilizó una arquitectura que implementó una conexión Cliente-Servidor.

- Cliente: se implementó una página web con toda la interfaz gráfica donde el usuario puede escribir el código del lenguaje de programación o subir un archivo de texto con el código ya definido.
- Servidor: se implementó la lógica para definir todos los tokens que se retornarán en la tabla de símbolos como salida del analizador léxico. Dentro se definieron todos los patrones a través de expresiones regulares, funciones y procedimientos donde se manejó el uso de tokens y errores para que sean devueltos como salida del programa dentro de la página web.



Tecnología utilizada para Backend y Frontend

Debido a que la arquitectura implementada dentro de la solución de nuestro proyecto requiere una vista realizada a través de frontend que conecte todo lo ingresado por parte del usuario con todas las funcionalidades que se establecieron bajo un backend que maneja todo en base a los datos de entrada.

Para el frontend se implementó una página web que se programó en lenguaje html que se conectó al framework de react que manipula javascript para que se pueda asociar todo a la información contenida en el backend.

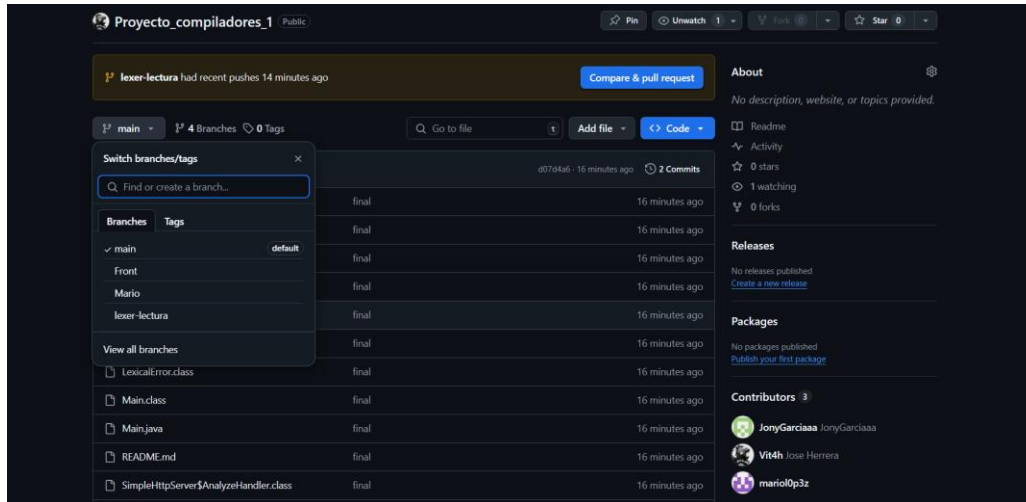
Para el backend se realizaron archivos en lenguaje java, ya que a través de la librería regex perteneciente a java.util se pudo definir toda expresión regular necesaria para generar los tokens que serán la salida de nuestro analizador léxico.

La funcionalidad general del programa se basa en que el usuario podrá ingresar su código a través de un archivo de entrada de texto o bien escribir en un editor que se visualiza dentro del frontend. Una vez se ejecuta, el Modelo Vista-Controlador asociará la entrada para poder conectar toda esta información con las funciones definidas en el backend. Este proceso identificará cada lexema dentro del código de entrada para asociarlo con un tipo de token que ya fue previamente definido. Si existe una coincidencia se guardará en la tabla de símbolos, caso contrario, se guardará como un error léxico que se mostrará en la página web. Una vez finalizada la ejecución se observará en la página web una tabla con todos los tokens definido; mostrando el tipo, valor, fila y columna de cada token.

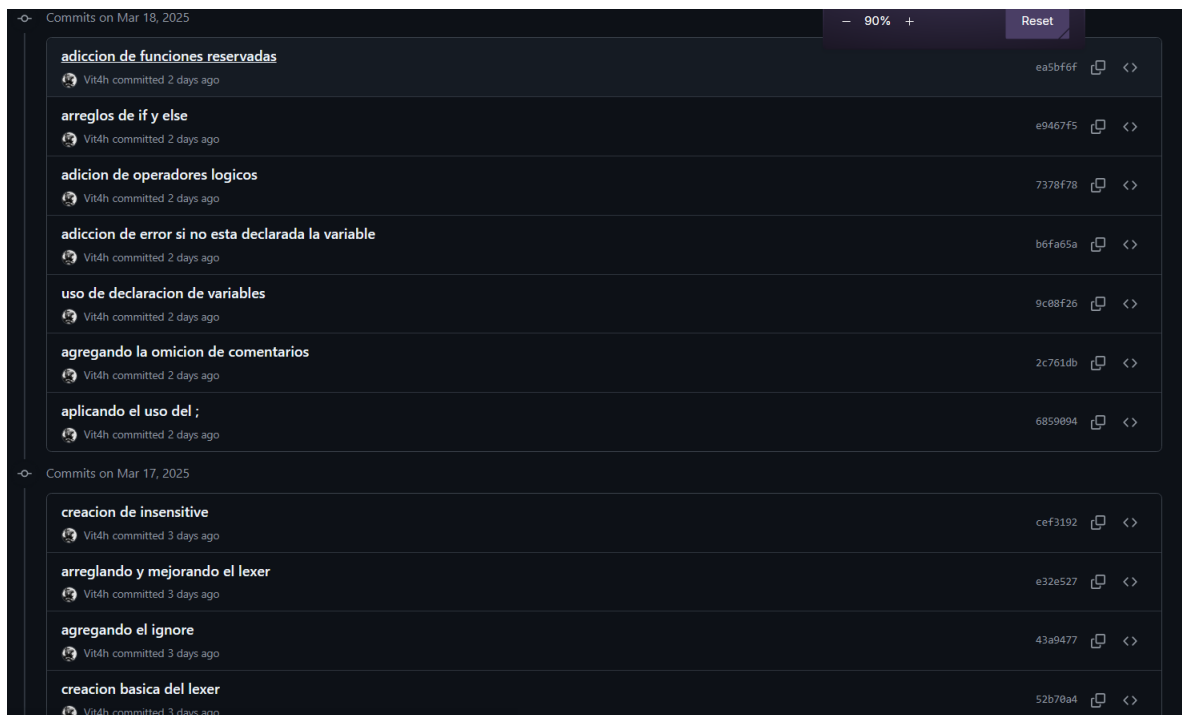
Link de GitHub







https://github.com/Vit4h/Proyecto_compiladores_1/tree/main





El proyecto se trabajó en diferentes ramas para dejar la master o main como la rama de producción haciendo de ello una forma mas organizada de manejar el proyecto.













Acá Podemos observar comitts del lexer donde se trabajo hasta parchear unos bugs y mejorando su productividad.



Commits on Mar 20, 2025		
manejo de errores finalizado	Vi4h committed 16 minutes ago	a6f348e  <>
mejora de manejo de errores	Vi4h committed 20 minutes ago	7f487e0  <>
final	Vi4h committed 11 hours ago	23ea941  <>
arreglo	Vi4h committed 11 hours ago	218c826  <>
Commits on Mar 19, 2025		
proyecto 1.0 finalizado	Vi4h committed 17 hours ago	8f6fc9e  <>
tabla de símbolos y ubicacion de linea y columna	Vi4h committed 17 hours ago	cec8ae5  <>

Commits on Mar 17, 2025		
Ajustes para el analizador lexico con case insensitive	mariol0p3z committed 3 days ago	2db6c04  <>
Ajustar todo a la branch Mario	mariol0p3z committed 3 days ago	1b92e58  <>
Creacion de analizador lexico	mariol0p3z authored 3 days ago	Verified a94d2a1  <>
Commits on Mar 10, 2025		
Initial commit	Vi4h authored last week	Verified 5ad754a  <>

Commits on Mar 20, 2025		
front y back unidos ya sin errores	JonyGarciaaa committed 38 minutes ago	e6b9ab0  <>
ya esta conectado el front y back y ahora solo falta terminar detalles	JonyGarciaaa committed 2 hours ago	4872310  <>
Commits on Mar 19, 2025		
front ya unido con back falta correcion de errores	JonyGarciaaa committed 12 hours ago	db0704b  <>
La correaaa comitt 2 Gianca	Gianca882 committed yesterday	3735387  <>
Commits on Mar 18, 2025		
Arreglar commit de gianca	Gianca882 committed 2 days ago	989472b  <>
Commit gianca arreglado	Gianca882 committed 2 days ago	ecf2d4e  <>
Front, primer commit Giancarlo	Gianca882 committed 2 days ago	7760a9c  <>
Commits on Mar 17, 2025		
primera parte front	JonyGarciaaa committed 3 days ago	4de9f89  <>

Commits on Mar 20, 2025		
final	JonyGarciaaa committed 20 minutes ago	d07d4a6  <>
Commits on Mar 10, 2025		
Initial commit	Vi4h authored last week	Verified 5ad754a  <>