ANALIZADOR LÉXICO

MANUAL TÉCNICO ANALIZADOR LÉXICO

Introducción

Este manual técnico proporciona una guía detallada para comprender la lógica detrás del programa “Analizador léxico”, utilizando la interfaz gráfica de java swing y el lenguaje de programación Java, el controlador de versiones Git, github.
Este manual técnico proporciona una descripción exhaustiva de las características del programa, su funcionamiento interno y los requisitos de instalación y configuración. Además, se incluyen detalles sobre la responsabilidad del desarrollador y la compañía desarrolladora, en cuanto a posibles riesgos asociados con el uso durante sesiones prolongadas o usos inadecuados, cambios en el código o el implemento de mejoras o cambio de sus funciones originales (mantenimiento).
Al momento de modificar un carácter o algo diferente a lo solicitado el programa podría dejar de funcionar debidamente, por favor tómelo en cuenta al momento de realizar un cambio o implementar una mejora, por ello se adjunta toda la documentación de análisis y diseño necesaria para comprender el programa.

Presentación

Bienvenidos desarrolladores al manual técnico de un Analizador Léxico, en este manual podrán encontrar lo necesario para poder entender la lógica y el proceso para poder realizar ajustes, cambios o posibles mejoras al programa.

¿Qué incluye este manual?

En este manual, encontrará una descripción detallada de las características del desarrollo del Analizador Léxico, su funcionamiento interno y los requisitos necesarios para implementar o cambiar algo del Analizador Léxico desde tu sistema.
- Aspectos destacados:
  - Herramientas utilizadas para el desarrollo del programa
  - Desarrollo del programa
  - Documentación Interna
  - Documentación externa

¿Por qué es importante este manual?

Este manual técnico es esencial para comprender completamente el funcionamiento del Analizador Léxico, así como para garantizar una experiencia de desarrollo sencilla, clara y concisa.

Resumen

Este manual técnico facilitará la implementación de una mejora o el cambio de una funcionalidad para cualquier desarrollador, es decir el mantenimiento del programa.

Objetivo

El objetivo de este manual es que como desarrolladores nos apoyemos facilitandonos el entendimiento y la comprensión de este programa, este programa nació con la finalidad de ser algo entretenido para nosotros, por ello se implementaron buenas prácticas de programación, por ello mismo se espera que este manual sea fácil de entender e interpretar por medio de las herramientas utilizadas.

Aspectos Técnicos

- Herramientas Utilizadas Para el Desarrollo del Programa:

· Java

Analizador Léxico es un juego desarrollado y creado exclusivamente con java. La versión usada de Java para desarrollar el juego Analizador Léxico es la versión “21.0.5” y la versión de JSE es la misma “21.0.5” y esta misma corresponde a la versión del JDK descargadas desde los sitios oficiales de java, con las variables de entorno configuradas.

· *Command Prompt*

Para observar los cambios realizados y las pruebas realizadas fue necesario ejecutar el archivo ejecutable .jar desde la terminal de windows 10.

· NetBeans

NetBeans fue el IDE en el cual se realizó el programa, donde se extrajo el archivo .jar,
en el proceso de la creación del proyecto se utilizó “Java Maven” y “Java Application”, la versión de NetBeans es la versión “Apache NetBeans IDE 22” y todas las dependencias utilizadas por este yacen descritas en el pom.xml

· Sistema Operativo

El programa Java y Netbeans fue utilizado en la versión 10 de Windows, con una arquitectura de 64 bits de un procesador intel.

- Requisitos Mínimos del Sistema

El programa está diseñado para que cualquier equipo que cuente con una consola (terminal “cmd”) pueda utilizar el programa, debido a que no es un programa que exige demasiados recursos, así que no hay requisitos mínimos (del hardware) propios del sistema para poder ejecutar el programa como tal, pero el programa podría llegarse a ver afectado por: La terminal del usuario (código cerrado o abierto con restricciones), una mala instalación de la JVM o una exageración (datos muy grandes) en los datos ingresados por el usuario en las entradas que requiere el programa para funcionar.

Documentación

Como motivo de facilitar el trabajo en equipo y realizar programación abierta para todo el público se implementaron distintos métodos de documentación, estas las clasificamos en:

- Documentación Interna

Definiendo la documentación interna como la herramienta en programación que está destinada a ser utilizada por los desarrolladores y miembros del equipo de desarrollo de software durante el proceso de creación de un programa o sistema. Esto porque esta documentación no está dirigida al usuario final, sino que está diseñada para facilitar el entendimiento y la colaboración entre los miembros del equipo de desarrollo.

· Comentarios en Código

Para explicar el código y tener buenas prácticas de programación, en cada clase se comentaron y enunciaron los métodos utilizados. Esto por si en algún momento se desee realizar un cambio en especifico. El formato de la explicación de cada método es utilizando javadoc, donde se explica lo que realiza el método, qué necesita y que retorna en dado caso lo haga.

* Comentarios de clases:

Se describe de forma breve la razón de existir de la clase, así como su autor, la versión de java que se utilizó para hacerla o las clases que pueden estar relacionadas a está y la fecha en la cual se creó y se terminó.

```
/**
 * Clase LexemaController es la clase encargada de ser el verificador de la
 * información del frontend y así porder hacer la lógica para el backend.
 *
 * @author YmCris
 * @since Aug 25, 2025
 */
public class LexemaController {
```

* Comentarios de métodos:

Se describe lo más detalladamente posible la acción que tiene el método, que representan los parámetros y que retorna (si aplica).

```java
// ********* MÉTODOS PARA VER SI UN LEXEMA PERTENECE AL LENGUAJE ***********
/**
 * Método encargado de analizar si un lexema pertentece a un lenguaje.
 *
 * @param palabra - palabra a analizar
 * @return true si pertenece al lenguaje
 */
private boolean lexemaPertenezcaAlLenguaje(Token palabra) {
    if (palabra.getLexema().isEmpty() || palabra.getLexema().isBlank()) {
        return false;
    }
    char[] caracteresDePalabra = palabra.getLexema().toCharArray();
    for (int i = 0; i < caracteresDePalabra.length; i++) {
        if (caracterPerteneceAlLenguaje(caracteresDePalabra[i]) == false) {
            palabra.setEsToken(esToken: true);
            palabra.setColor(color: "rojo");
            System.out.println("El lexema " + palabra.getLexema() + " no pertenece al lenguaje");
            return false;
        }
    }
    System.out.println("El lexema " + palabra.getLexema() + " pertenece al lenguaje");
    palabra.setEsToken(esToken: true);
    return true;
}
```

* Comentarios de variables:

En el comentario de variables no se utiliza el formato de javadoc porque representa un crecimiento del tamaño de una clase de manera exponencial, por lo tanto solo se hace referencia el tipo de variables que son, ya sean de referencia o primitivas y en algunas muy importantes se hace un comentario simple.

```java
private String[] palabrasReservadas;
private ArrayList<Token[]> palabrasDeLasLineas;//Guarda las palabras de cada línea
```

```java
// VARIABLES DE REFERENCIA -----------------------------------------------------
private ArrayList<String> lineas;
private JFAnalizador jfAnalizador;
private String[] palabrasReservadas;
private ArrayList<Token[]> palabrasDeLasLineas;//Guarda las palabras de cada línea

// VARIABLES PRIMITIVAS --------------------------------------------------------
private char[] signosDePuntuacion;
private char[] signosDeAgrupacion;
private char[] operadoresAritmeticos;
```

* Estructura de una clase de backend:

En la creación de una clase de frontend se divide el contenido de la clase en base a la siguiente “Plantilla”, separando así todos los elementos más importantes de la clase, para que sea más fácil de entender a todos los programadores, aunque claro que la estructura de esta planilla se ve modificada en muchas clases con métodos sobreescritos, funciones, etc.

```java
/**
 * Clase HolaMundo
 *
 * @author YmCris
 * @since Mar 31, 2025
 */
public class HolaMundo {

    // VARIABLES DE REFERENCIA ---------------------------------------------------


    // VARIABLES PRIMITIVAS ------------------------------------------------------


    // INSTANCIAS ----------------------------------------------------------------


    // MÉTODO CONSTRUCTOR --------------------------------------------------------


    // MÉTODOS -------------------------------------------------------------------


    //  GETTERS & SETTERS ------------------------------------------------------


}
```

* Configuraciones en el pom.mxl:

Para utilizar el recurso de archivos JSON se utilizó la dependencia de Gson de google, por lo cual es importante declarar esa dependencia en el archivo pom, de lo contrario el ejecutable no

podrá acceder a ese recurso por lo cual el programa no funcionará.

```
    <groupId>ymcris.languages.practice.one</groupId>
    <artifactId>AnalizadorLexico</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <exec.mainClass>ymcris.languages.practice.lexicalanalyzer.AnalizadorLexico</exec.mainClass>
    </properties>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.13.1</version>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <configuration>
                    <archive>
                        <manifest>
                            <addClasspath>true</addClasspath>
                            <mainClass>ymcris.languages.practice.lexicalanalyzer.AnalizadorLexico</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.5.0</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <transformers>
                                <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                    <mainClass>ymcris.languages.practice.lexicalanalyzer.AnalizadorLexico</mainClass>
                                </transformer>
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
            </plugin>

        </plugins>
    </build>
</project>
```

- Documentación Externa

Definiendo la documentación externa como la herramienta en programación que transmite información importante destinada a ser utilizada por usuarios finales, clientes, administradores de sistemas u otras partes interesadas que no forman parte del equipo de desarrollo de software. Esta documentación está diseñada para proporcionar orientación, instrucciones y recursos para el uso adecuado y efectivo del software desarrollado.

· Descripción de clases:

Definiendo una **clase** como una planilla la cual sirve como molde para la creación de instancias o para realizar acciones (Basado en la SRP “Single Responsibility Principle”), comunicándose entre otras clases a través de acciones (Métodos), sin embargo ahora se implementó el concepto de Modelo, Vista y control, esto para tener mejor estructurado todo el código y facilitar el desarrollo y su mantenimiento, por lo que se describe de esa manera.

Frontend

- JFMenuPrincipal: Ventana principal encargada de iniciar el menú del sistema.
- JDCargarArchivo: Diálogo para seleccionar un archivo de texto.
- JDEditarPalabra: Diálogo para editar una palabra del archivo JSON.
- JDGuardarArchivo: Diálogo para guardar un archivo de texto en una carpeta elegida por el usuario.
- JDReportes: Diálogo para mostrar reportes, con opciones de generación y exportación.
- JFConfigJSON: Ventana que permite al usuario configurar el archivo JSON a su gusto.
- JFAnalizador: Ventana que muestra el analizador y centraliza la mayor parte de sus funcionalidades.

Backend

- Archivo: Encargada de analizar el archivo de entrada, leerlo y devolver un ArrayList con su contenido.
- ArchivoJSON: Crea y actualiza el archivo JSON, además de usar métodos de la clase Token para modificarlo.
- Comentario: Contiene un bloque de comentario (una o varias líneas).
- LexicoDelLenguaje: Representa el concepto de token en el programa, con atributos necesarios para crear y modificar el archivo JSON.
- ClasificadorDeToken: Recibe los tokens definidos por el usuario y los transforma según su estructura.
- Token: Representa una palabra en el análisis léxico, con atributos que permiten identificar qué tipo de lexema es.

Controller

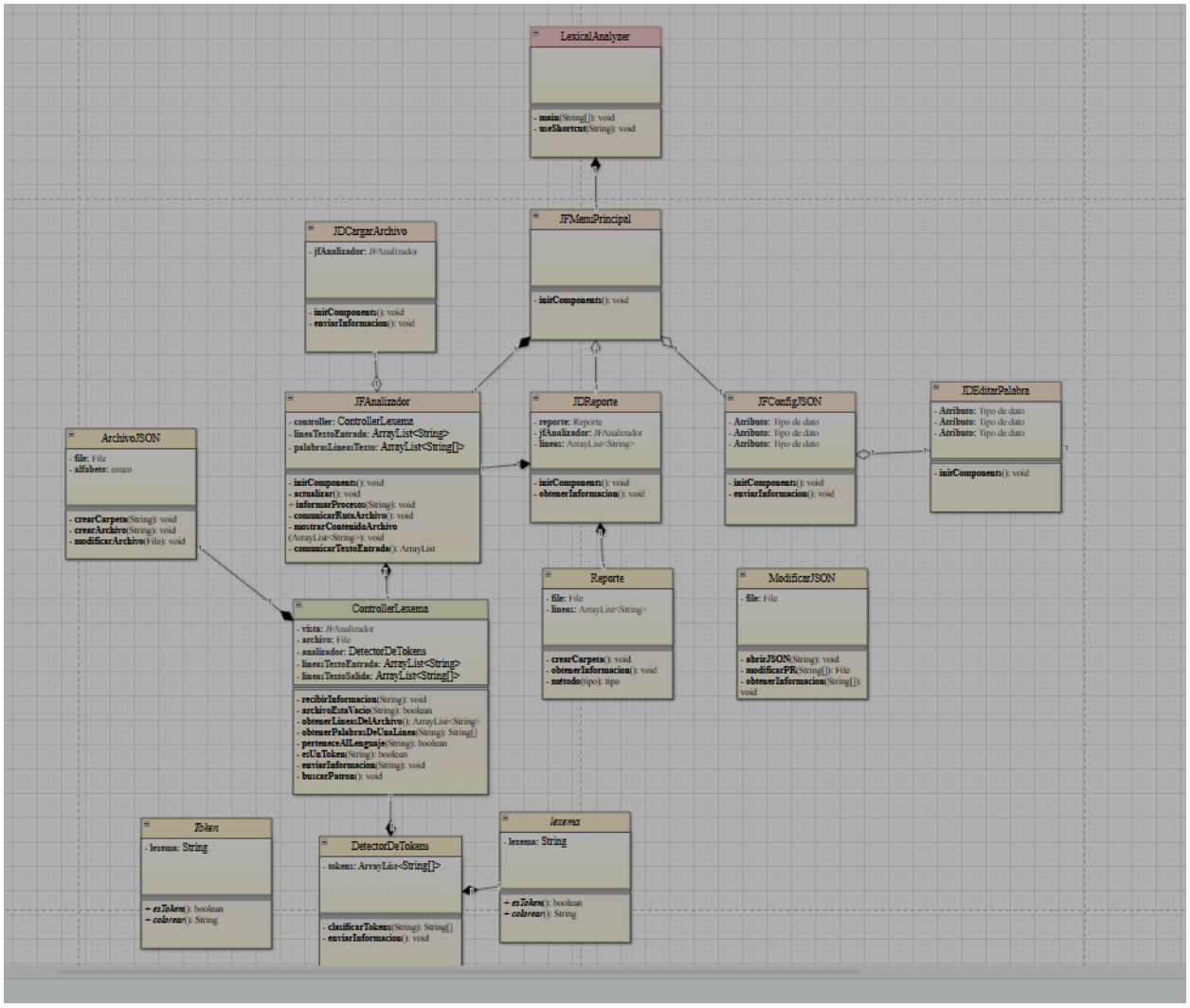
- LexemaController: Verificador de la información proveniente del frontend, que conecta con la lógica del backend.

· Fase de análisis:

Para representar gráficamente el programa se emplea el diagrama de UML diseñado en flowChart.

Fase de Análisis:

1. Diagrama de clases

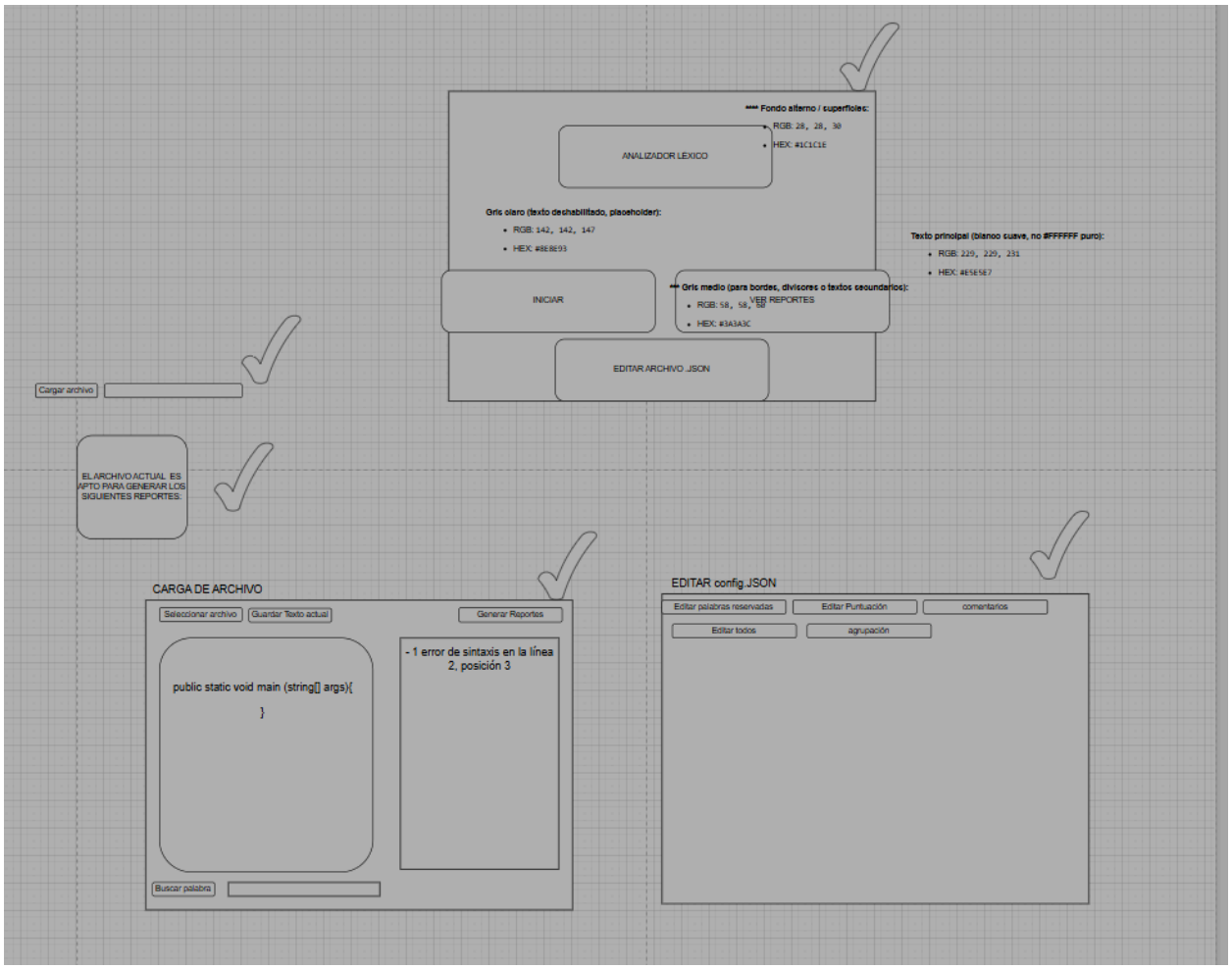


· Fase de diseño:

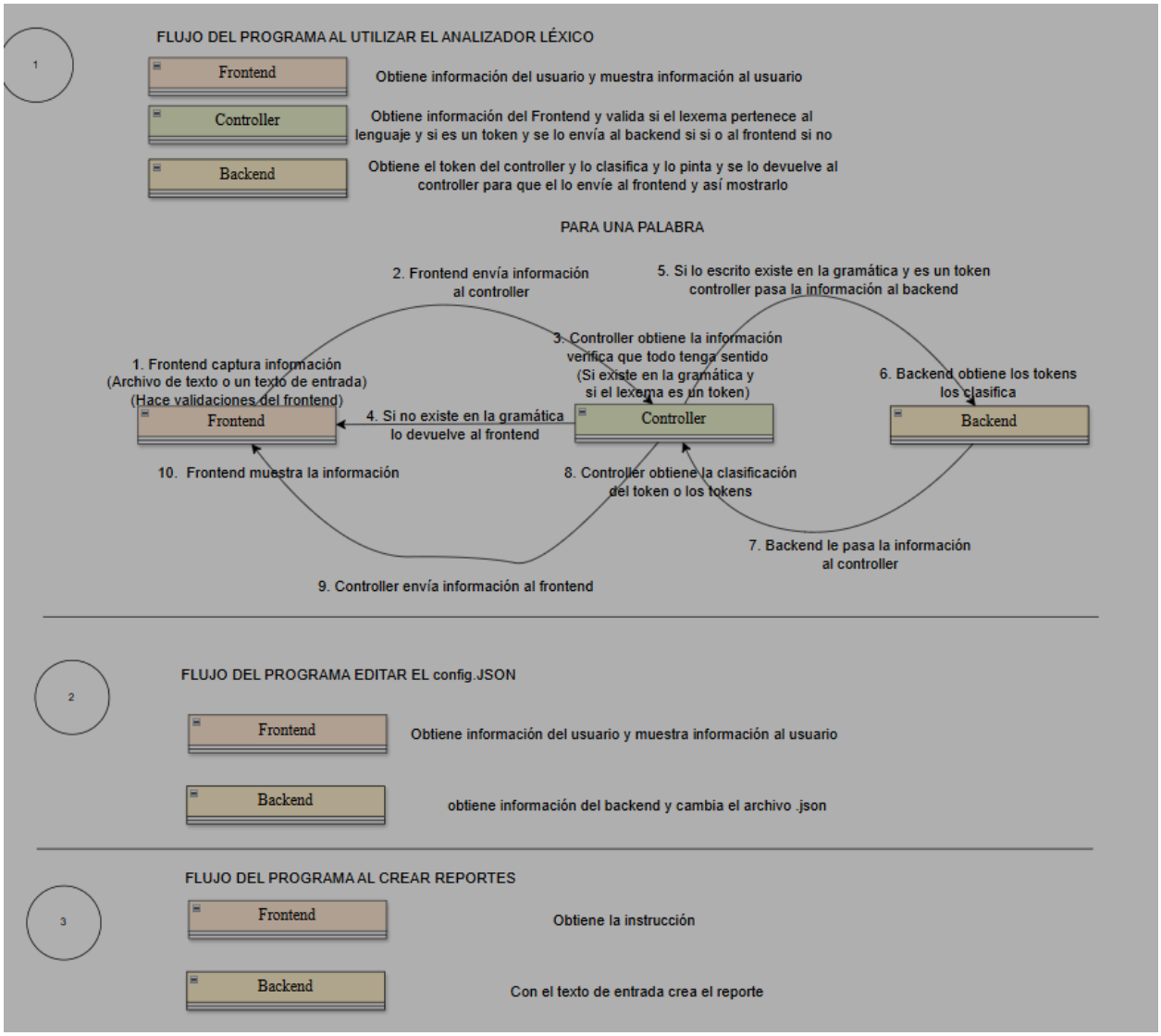
Para representar gráficamente el programa se plantea un diseño de distintas partes del programa.

Fase de Diseño:

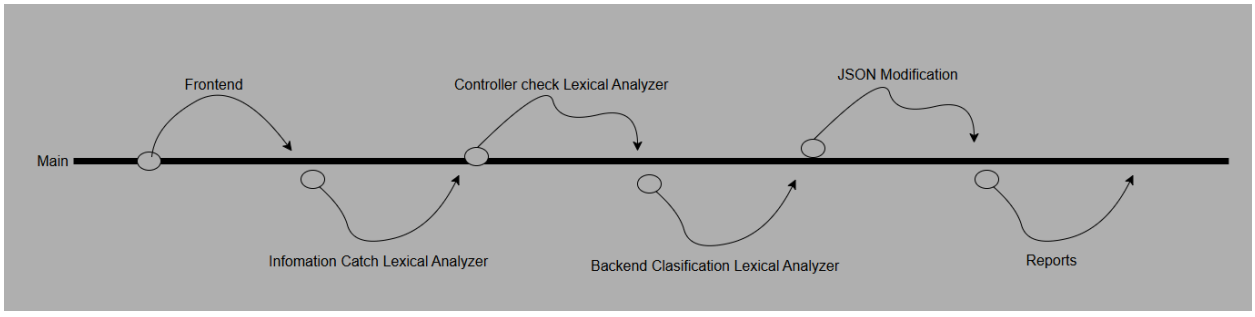
1. Bosquejo de la apariencia del programa



2. Modelo del flujo del programa



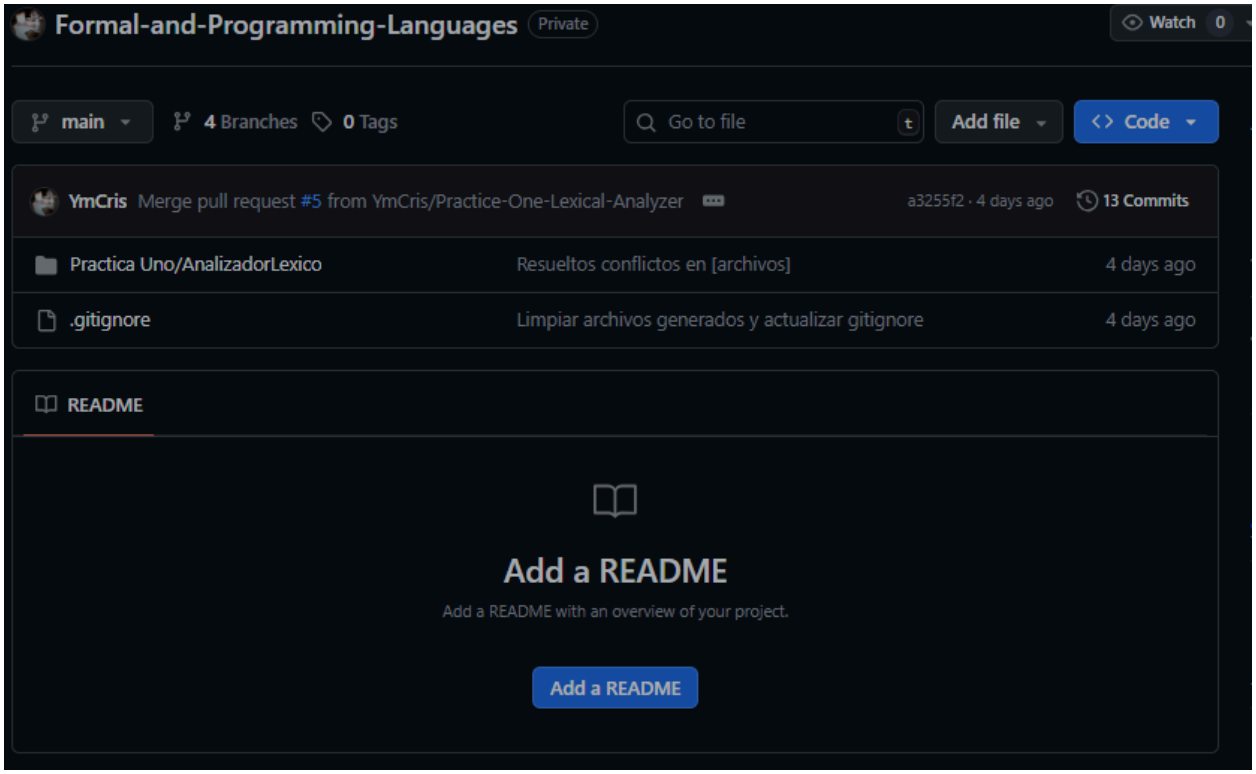
3. Uso de las ramas en git



· Repositorio en GitHub:

Para una mejor descripción de los avances en el transcurso de la creación del programa se describen detalladamente los cambios en los commits del repositorio.

Repositorio en GitHub:



· Documentación en JavaDoc:

Para representar los comentarios en código se incluye dentro de la carpeta target el index del Overview.

Overview (AnalizadorLéxico 1.0-SNAPSHOT API)

AnalizadorLexico 1.0-SNAPSHOT API

Packages

Package	Description
ymcris.languages.practice.lexicalanalyzer	
ymcris.languages.practice.lexicalanalyzer.backend.file	
ymcris.languages.practice.lexicalanalyzer.backend.json	
ymcris.languages.practice.lexicalanalyzer.backend.lenguaje	
ymcris.languages.practice.lexicalanalyzer.backend.manejodetokens	
ymcris.languages.practice.lexicalanalyzer.backend.tokens	
ymcris.languages.practice.lexicalanalyzer.controller.validation	
ymcris.languages.practice.lexicalanalyzer.frontend	
ymcris.languages.practice.lexicalanalyzer.frontend.dialogs	
ymcris.languages.practice.lexicalanalyzer.frontend.json	
ymcris.languages.practice.lexicalanalyzer.frontend.uploadfile	

Problemas del Programa

En la compañía, entendemos que la satisfacción del cliente es nuestra prioridad número uno. Por eso, nos comprometemos a ofrecer un servicio de calidad que garantice una experiencia sin problemas al utilizar nuestros productos. En caso de que encuentres algún inconveniente durante la descarga, instalación o ejecución de nuestro programa, estamos aquí para ayudarte. Nos comprometemos a brindar asistencia técnica y, si es necesario, realizar devoluciones dentro de un plazo de 12 horas después de tu compra.

Es importante destacar que la mayoría de los problemas que puedas encontrar suelen ser el resultado de una ejecución incorrecta, pero no te preocupes, estamos aquí para guiarte en cada paso del proceso. Simplemente reiniciando el programa o volviendo a instalarlo, muchos de estos problemas se pueden resolver fácilmente.

Sin embargo, debemos señalar que cualquier modificación no autorizada al programa o descarga incorrecta podría limitar nuestra capacidad para abordar el problema. En tales casos, nos reservamos el derecho de no proporcionar asistencia.

Nuestro equipo de soporte técnico está siempre disponible para ayudarte a resolver cualquier inconveniente que puedas encontrar. Queremos que tengas la mejor experiencia posible con nuestros productos y haremos todo lo posible para asegurarnos de que así sea.