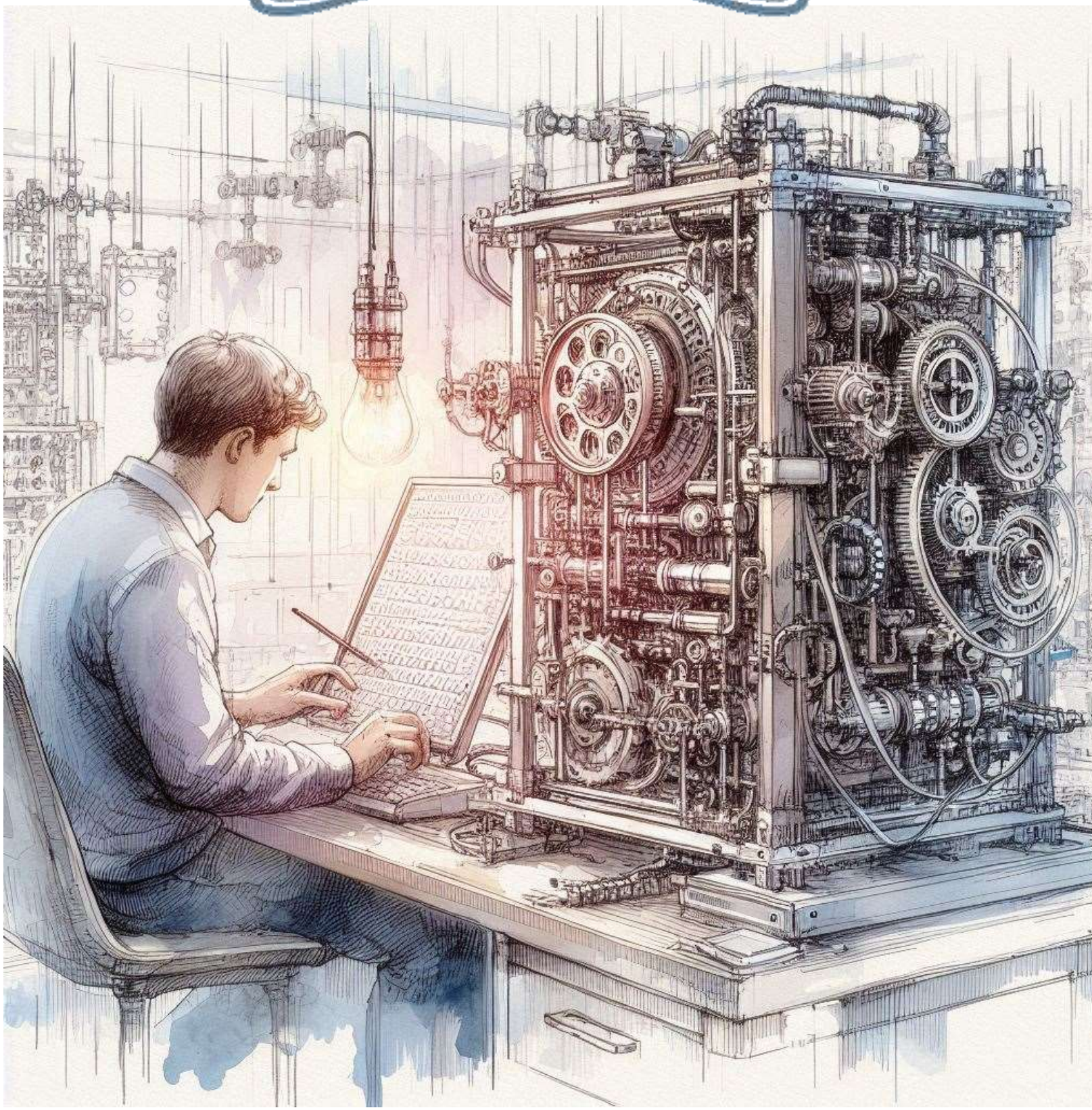
MANUAL DE USUARIO



Contenido

Analizador SQL

Introducción

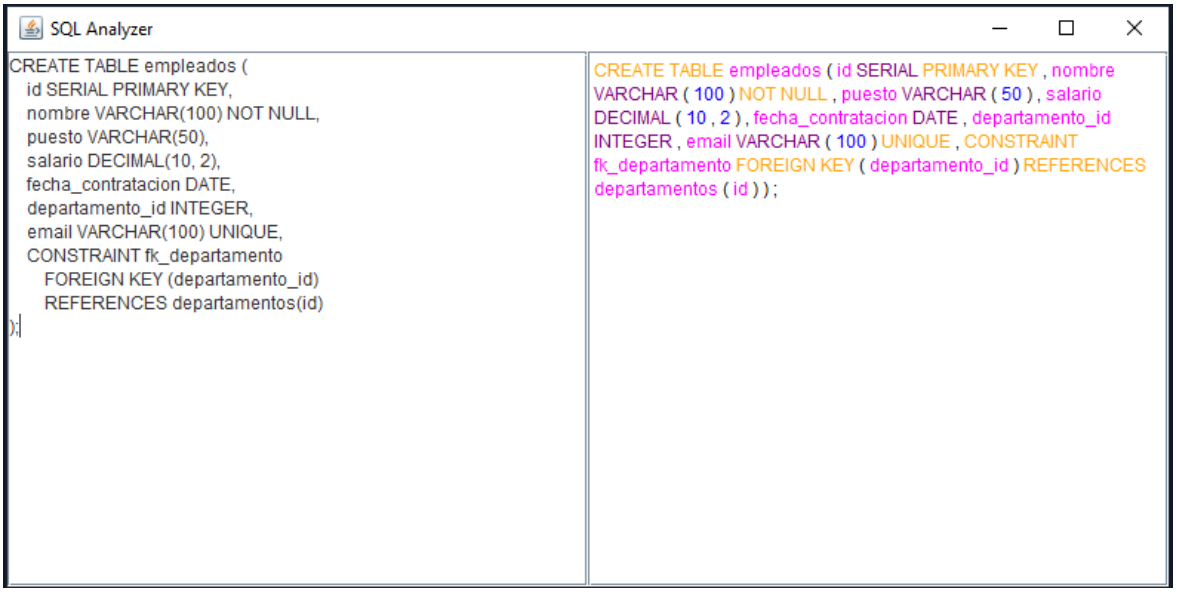
La siguiente aplicación en Java, titulada SQL Analyzer, es un entorno de análisis y visualización de código SQL desarrollado con Swing, una biblioteca para interfaces gráficas.

El programa permite a los usuarios ingresar código SQL en un área de texto, donde cada token (o unidad léxica) del código ingresado se identifica y se colorea de acuerdo con su tipo. A su vez, el sistema es capaz de analizar léxicamente el código, generando representaciones visuales de los tokens en una segunda área de visualización, lo cual facilita la comprensión y depuración del SQL procesado.

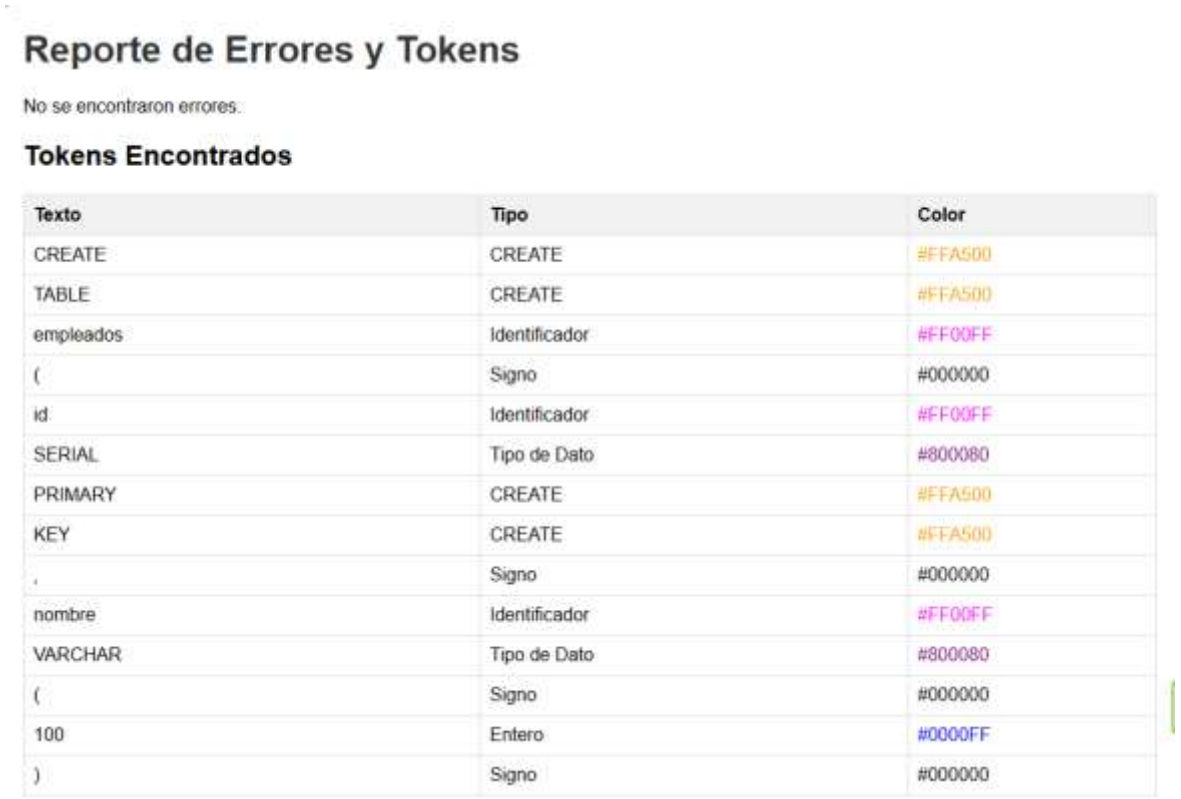
Al ejecutar la aplicación, el usuario encuentra una ventana principal con dos paneles de texto: uno para el ingreso de código SQL (JTextArea) y otro para la visualización de tokens (JTextPane).

La entrada de código está equipada con un DocumentListener que permite analizar el código en tiempo real, resaltando tokens a medida que se ingresan o editan.

La funcionalidad de análisis está impulsada por el SQLLexer, que se encarga de segmentar el código en tokens, asignando colores específicos a través de un StyledDocument, facilitando la distinción visual de palabras clave, identificadores, operadores, y demás componentes del lenguaje SQL.



Una vez que se identifican y clasifican los tokens, el programa los pasa al Analizador Sintáctico (AnalizadorSintactico), el cual evalúa la estructura y genera un reporte en un archivo HTML (Reportes.html). Este archivo documenta el proceso de tokenización y



Reporte de Errores y Tokens

No se encontraron errores.

Tokens Encontrados

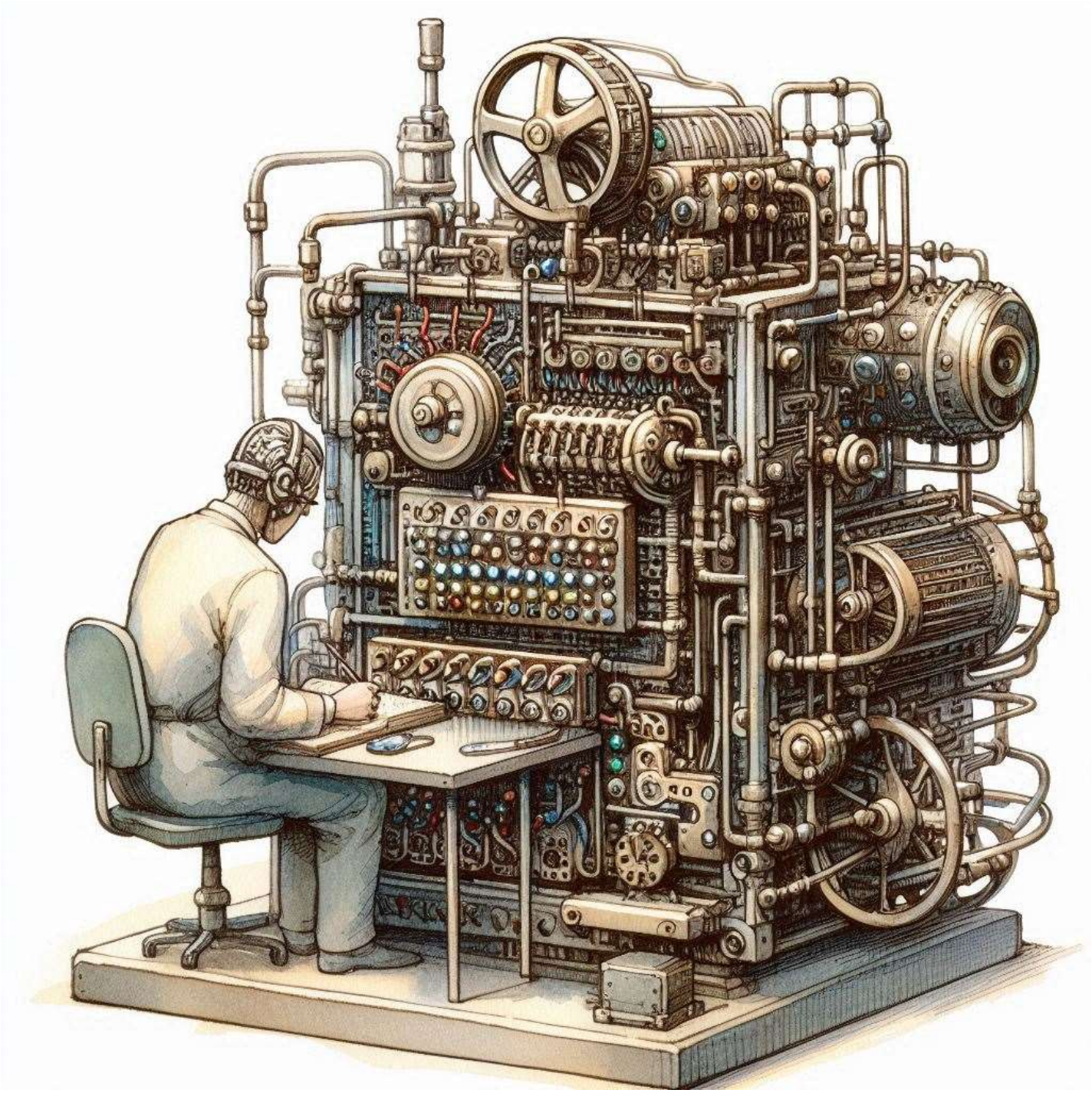
Texto	Tipo	Color
CREATE	CREATE	#FFA500
TABLE	CREATE	#FFA500
empleados	Identificador	#FF00FF
(	Signo	#000000
id	Identificador	#FF00FF
SERIAL	Tipo de Dato	#800080
PRIMARY	CREATE	#FFA500
KEY	CREATE	#FFA500
,	Signo	#000000
nombre	Identificador	#FF00FF
VARCHAR	Tipo de Dato	#800080
(	Signo	#000000
100	Entero	#0000FF
)	Signo	#000000

proporciona un registro detallado de los elementos analizados, útil para reportes y revisiones exhaustivas del código SQL.

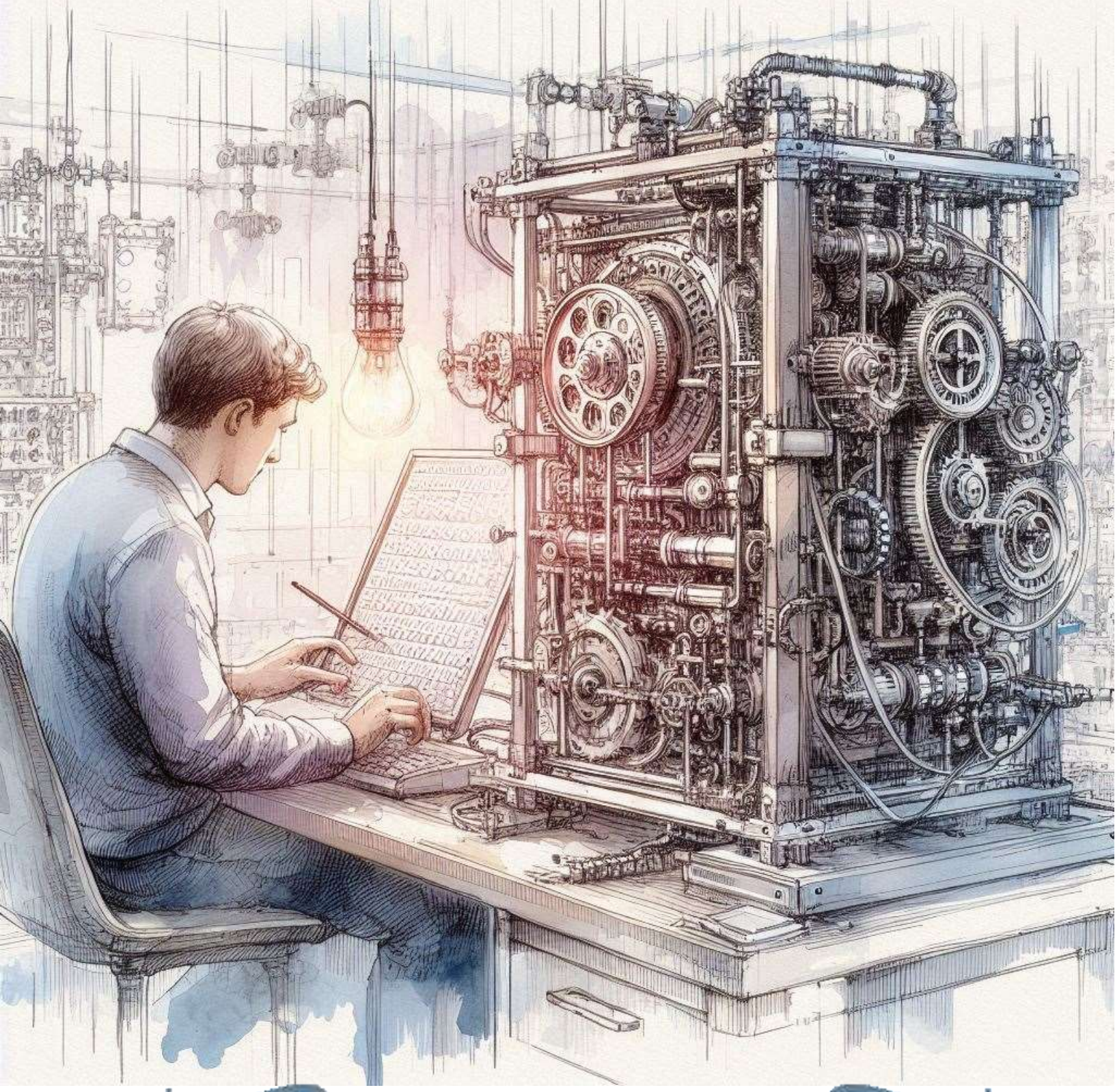
El programa implementa una función para convertir colores de tipo String a objetos Color, lo que permite adaptar los tokens visualizados a una amplia gama de colores, ya sea en formato hexadecimal o en nombres predefinidos.

Este detalle asegura una personalización visual precisa, esencial para una mejor experiencia de usuario en el análisis de código SQL. En resumen, SQL Analyzer es una herramienta robusta y adaptable, diseñada para realizar análisis léxico y sintáctico de código SQL, visualizando los resultados en tiempo real y generando reportes detallados para el usuario.

AUTOR



NOMBRE: VICTOR VEGA
CARNÉ: 202230772



MANUAL TÉCNICO

Contenido

LENGUAJE DE PROGRAMACIÓN JAVA-VERSIÓN 17

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++.

IDE: SUBLIME TEXT

Sublime Text es un editor de texto avanzado que se utiliza principalmente para la programación y la codificación. Fue desarrollado por Jon Skinner en 2008 y se ha convertido en una herramienta popular entre los desarrolladores debido a su facilidad de uso y su gran variedad de funciones y características.

- Resaltado de sintaxis: Sublime Text es capaz de reconocer diferentes lenguajes de programación y resaltar la sintaxis de manera que el código sea más fácil de leer.
- Autocompletado: Sublime Text puede autocompletar el código en función de lo que el usuario está escribiendo, lo que acelera el proceso de escritura y reduce los errores. Búsqueda y reemplazo: Sublime Text tiene una herramienta de búsqueda y reemplazo avanzada que permite al usuario buscar y reemplazar cadenas de texto en todo el documento.

IDE: APACHE NETBEANS 17

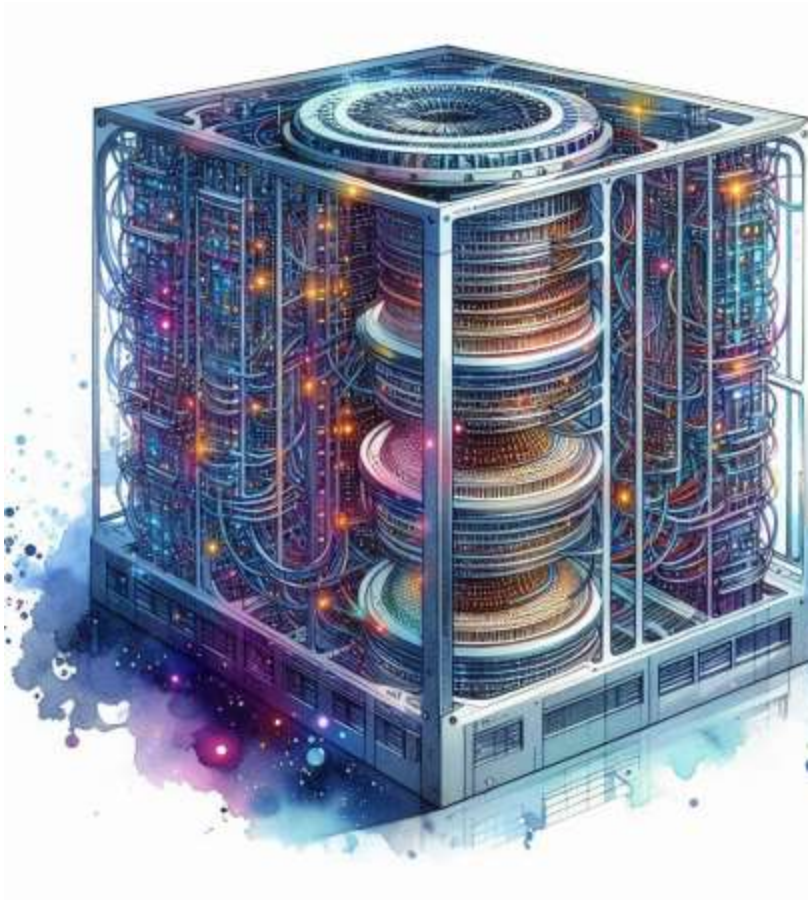
NetBeans es un entorno de desarrollo integrado, de código abierto y sigue la filosofía del software libre, y está hecho principalmente para el uso del lenguaje de programación Java.

- También existen muchísimos módulos extras para extender su funcionamiento.

  NetBeans es un producto libre y gratuito sin restricciones de uso y es un proyecto de código abierto de gran éxito con una gran base de usuarios y una comunidad en constante crecimiento.

SISTEMA OPERATIVO: WINDOWS 10

Windows 10 es un sistema operativo desarrollado por Microsoft y es la última versión de la familia de sistemas operativos Windows. Fue lanzado en julio de 2015 y ha sido una actualización importante en comparación con las versiones anteriores de Windows.



- `AnalizadorLexico`

La clase `SQLLexer` se encarga de realizar el análisis léxico para un lenguaje SQL, categorizando y tokenizando el código fuente en tipos específicos, como palabras clave, identificadores, números, cadenas, operadores y espacios en blanco. Se trata de una clase Java que utiliza estructuras y métodos internos para procesar, identificar y categorizar cada tipo de símbolo en SQL, lo cual es fundamental en el análisis léxico.

- Componentes principales

- **Tipos de Token Definidos:**
  - La clase declara constantes (`KEYWORD`, `IDENTIFIER`, `NUMBER`, `STRING`, `OPERATOR` y `WHITESPACE`) que representan los tipos de tokens que el lexer puede identificar.
- **Estado Inicial y Estados de Léxico:**
  - `YYINITIAL` se define como el estado inicial del lexer, y `ZZ_LEXSTATE` se utiliza para definir los posibles estados en los que el lexer puede operar.
- **Tamaño de Buffer y Caracteres Map:**
  - `ZZ_BUFFERSIZE` establece el tamaño de buffer (16384) para manejar el análisis de grandes bloques de código.
  - `ZZ_CMAP_TOP` y `ZZ_CMAP_BLOCKS` son arreglos que contienen mapas de caracteres para facilitar el reconocimiento de patrones dentro del código SQL. Estos se crean y llenan a partir de datos empaquetados, lo cual reduce el uso de memoria al definir los caracteres reconocidos en SQL.
- **Acciones y Transiciones:**
  - `ZZ_ACTION` y `ZZ_TRANS` son arreglos que contienen información de las acciones y transiciones del lexer según los caracteres y los patrones encontrados. Se procesan en métodos auxiliares como `zzUnpackAction` y `zzUnpacktrans`.
- **Mapas de Fila:**
  - `ZZ_ROWMAP` es un mapa de filas que gestiona las posiciones en las transiciones, optimizando la asignación de patrones en el proceso de análisis.
- **Anotación `@SuppressWarnings`:**
  - La anotación `@SuppressWarnings("fallthrough")` indica que la clase permite que las instrucciones se "caigan" o avancen en un `switch-case` sin

advertencias, útil para optimizar y reducir código repetitivo en el análisis de tokens.

- Funcionamiento General

- Al leer el código fuente SQL, `SQLLexer` utiliza su conjunto de estados y transiciones para identificar patrones, caracterizando el código en tokens específicos según el tipo y el contexto en el que aparezcan. Con esto, se generan los tokens que posteriormente pueden ser usados en un analizador sintáctico para validar la estructura SQL.
- La clase `SQLLexer` permite a un analizador léxico SQL clasificar eficientemente elementos del lenguaje SQL y preparar el código para su posterior procesamiento, mejorando así la precisión y el rendimiento del análisis léxico.

AnalizadorSintactico

- *Atributos*

- **`private Iterator<Token> tokens;`**
  - Este atributo almacena un iterador sobre la lista de tokens generados por el analizador léxico. Permite recorrer los tokens de manera secuencial durante el proceso de análisis.
- **`private Token tokenActual;`**
  - Representa el token que se está procesando actualmente. Se actualiza en cada avance a través del método `avanzar()`.
- **`private Set<String> mensajesDeError;`**
  - Un conjunto que almacena mensajes de error únicos. Utilizar un conjunto garantiza que los mensajes de error no se repitan, lo que optimiza la presentación de los errores encontrados durante el análisis.
- *Constructor*

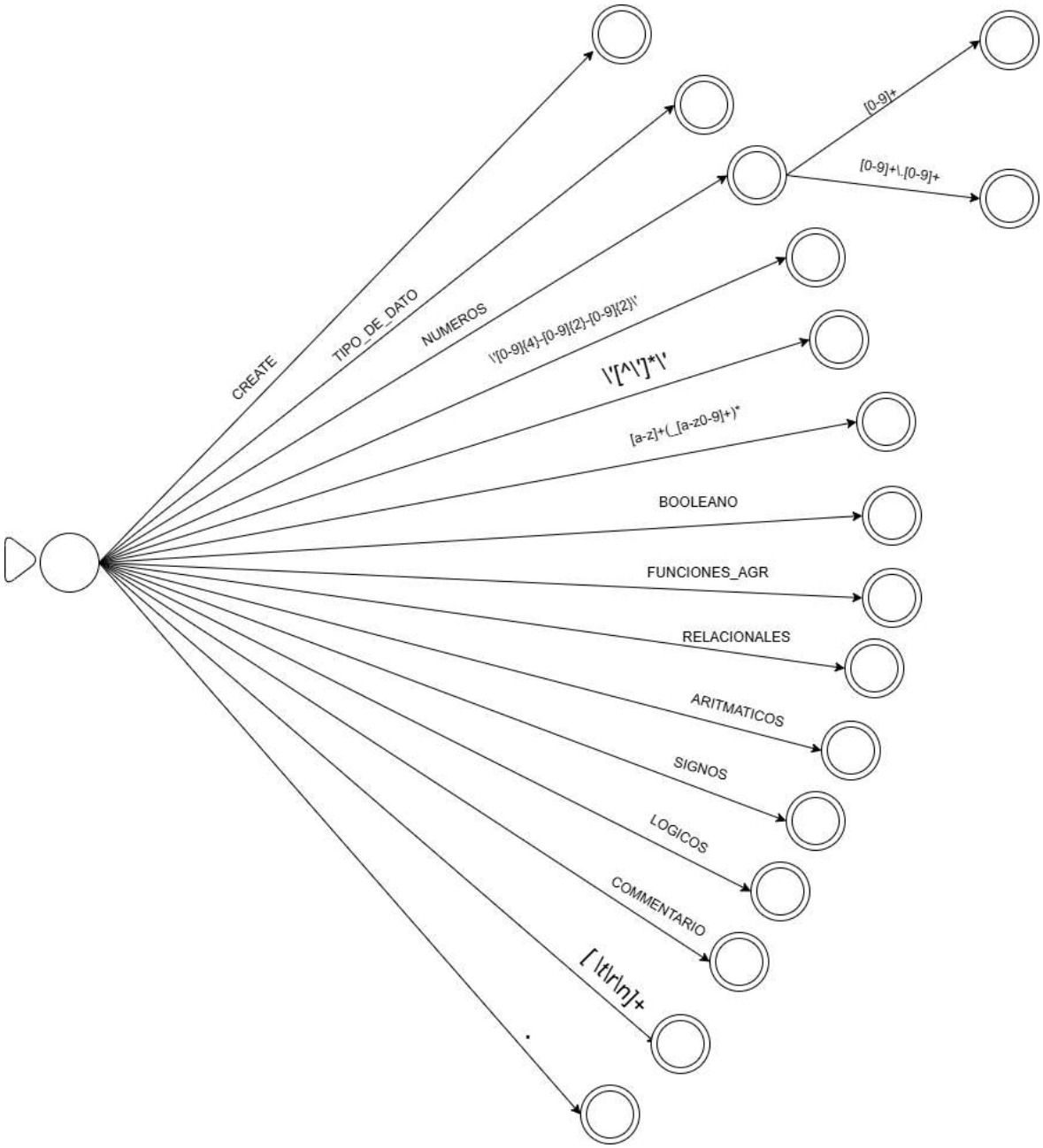
- **`public AnalizadorSintactico(List<Token> tokens)`**
  - Este constructor inicializa el iterador de tokens y avanza al primer token disponible. También inicializa el conjunto de mensajes de error.
- *Métodos*

- **`private void avanzar()`**
  - Este método se encarga de avanzar al siguiente token del iterador. Si no hay más tokens, establece `tokenActual` en `null`, indicando el final de la secuencia.
- **`public void analizar()`**
  - Este es el método principal del analizador sintáctico. Procesa secuencialmente los tokens y determina qué tipo de instrucción SQL se está analizando. Maneja las siguientes operaciones:
    - **CREATE DATABASE**
    - **CREATE TABLE**
    - **INSERT INTO**
    - **SELECT**
    - **UPDATE**
    - **DELETE**

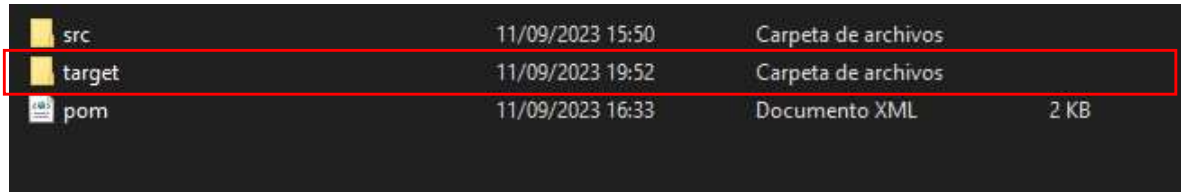
- Registra errores cuando se detectan inconsistencias o estructuras sintácticas incorrectas.

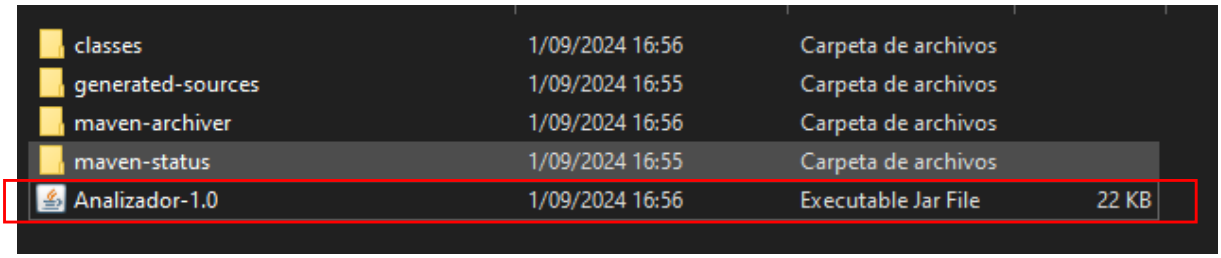
- **`private void analizarCrearBaseDeDatos()`**
  - Este método analiza la estructura de la instrucción `CREATE DATABASE`. Verifica que el siguiente token sea un identificador y que la instrucción finalice correctamente con un punto y coma (`;`).
- **`private void analizarCrearTabla()`**
  - Analiza la instrucción `CREATE TABLE`, asegurándose de que siga el formato correcto. Esto incluye la verificación del identificador de la tabla, la apertura y cierre de paréntesis, y la existencia de un punto y coma al final.
- **`private void analizarEstructuraTabla()`**
  - Este método se utiliza para analizar la definición de las columnas y restricciones dentro de la declaración de una tabla. Permite procesar declaraciones de columnas y claves.
- **`private void analizarEstructuraDeclaracion()`**
  - Analiza una declaración de columna individual, validando el identificador del nombre de la columna, el tipo de dato y restricciones como `PRIMARY KEY`, `NOT NULL` y `UNIQUE`.
- **`private void analizarTipoDeDato()`**
  - Este método se encarga de validar y analizar los tipos de datos permitidos en una declaración de columna, como `SERIAL`, `INTEGER`, `VARCHAR`, entre otros.
- **`private void analizarEstructuraDeLlaves()`**
  - Analiza las declaraciones de llaves foráneas, asegurándose de que la sintaxis sea correcta y que se especifiquen correctamente las referencias a otras tablas.
- **`private void registrarError(String mensaje)`**
  - Este método se utiliza para registrar mensajes de error. Solo añade el mensaje al conjunto si no está presente, garantizando que se muestre cada error una única vez.
- **`public void generarArchivoHTML(String nombreArchivo, List<Token> tokens)`**
  - Genera un archivo HTML que contiene un reporte de errores y los tokens procesados. Se asegura de que los errores y los tokens se presenten de manera clara y estructurada.
- **`private String escapeHtml(String texto)`**
  - Escapa caracteres especiales en una cadena de texto para asegurar que se presenten correctamente en el HTML.

ARCHIVOS RESTANTES



ARCHIVOS JAR



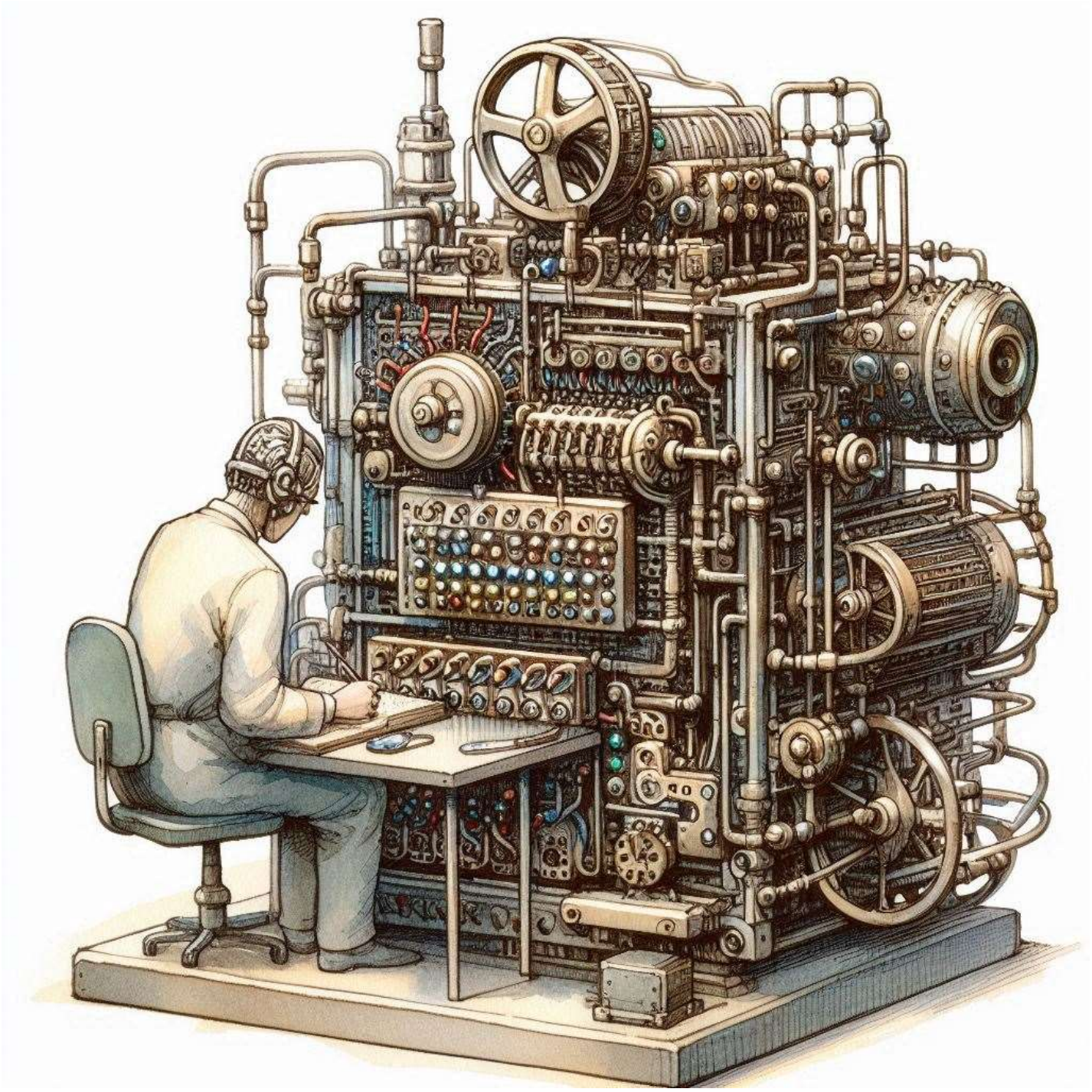


Repositorio

Link del repositorio:

https://github.com/Vega1402/AnalizadorSQL

AUTOR



NOMBRE: VICTOR VEGA
CARNÉ: 202230772