Documentación Notebook

El presente documento muestra la documentación de la aplicación “Notebook”, contiene el diagrama de clases y la estructura de como se realizaron las gramáticas para el analizador sintáctico.

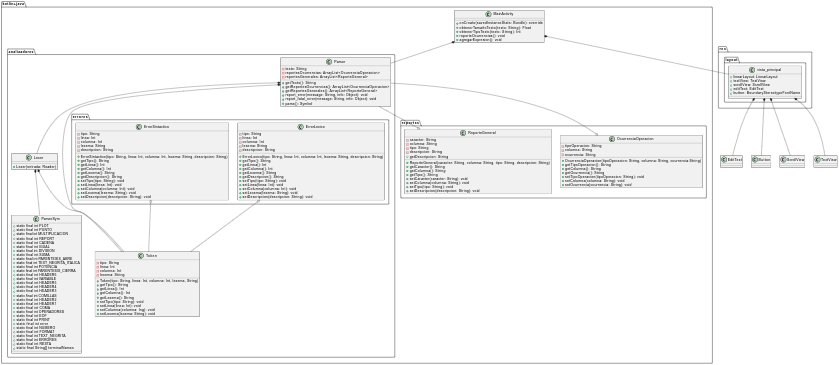
Tecnologías utilizadas

Para la realización de la aplicación móvil se utilizó lo siguiente:

* Kotlin como lenguaje principal.
* Java versión 22 para utilizar jflex y cup.
* Jflex para la creación del analizador léxico, jar utilizado: jflex-full-1.9.1.jar.
* Cup para la creación del analizador sintáctico por medio de gramáticas, jar utilizado: java-cup-11b.jar.
* Android Studio como IDE para la creación de la aplicación.
* PlantUML para crear el diagrama de clases.

Diagrama de clases

El siguiente diagrama de clases contiene la estructura de cómo se planteó la solución para crear la aplicación móvil.



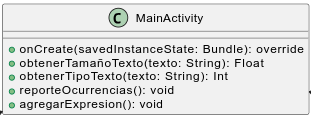
Cada paquete y contenido se describe a continuación:

# Paquete kotlin+java

Este paquete contiene principalmente toda la parte del backend, ya que en este paquete existen más paquetes los cuales se encargan de los analizadores léxico y sintáctico, además de clases tanto en java como kotlin que funcionan como recursos para otras clases.

## MainActivity

Esta clase es la clase principal de la aplicación, es donde comienza la ejecución del programa, hace uso de la vista que se creó en el archivo xml, además de que utiliza el texto recibido del frontend y lo analiza con el lexer y parser.



Método onCreate

Este método se encarga de dar inicio a la actividad de la aplicación, es decir, funciona como el método main en java.

Método obtenerTamañoTexto

Después de que el parser devuelve un texto válido entra en esta función que permite modificar el tamaño del texto para posteriormente mostrar en el frontend el tamaño el texto según se ingresó.

Método obtenerTipoTexto

Este método funciona similar al anterior, la diferencia es de que este se encarga de modificar el tipo de texto para mostrar texto en italica, negrita o negrita cursiva.

Método agregarExpresion

Este método recibe el texto que el método onCreate recibió del frontend y envía a este método, en este método existen las instancias del lexer y parser y hace uso de los métodos anteriores si es el caso.

Método reporteOcurrencias

Este método funciona igual que los demás, ya que si recibe un parámetro especial del parser éste se ejecutará para mostrar una tabla con las ocurrencias de operaciones aritméticas que se ejecutaron.

## Paquete analizadores

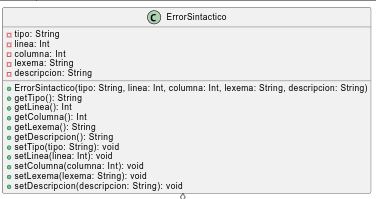
En este paquete se encuentran las clases java y kotlin necesarias para el funcionamiento del lexer y parser, además contiene un paquete el cuál.

### Paquete errores

Este paquete contiene únicamente 2 clases kotlin, se encargan de manejar objetos de tipo errores ya sea léxico o sintáctico.

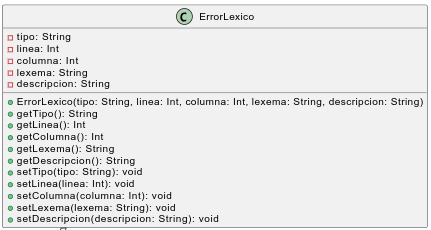
#### ErrorLexico

Esta clase se encarga de ser el modelo para crear objetos de tipo errores léxicos, estos almacenen datos además de los getter y setters para cada atributo.



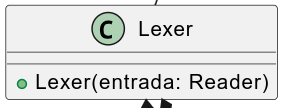
#### ErrorSintactico

Tiene el funcionamiento similar al de error léxico, contiene únicamente sus atributos, cosntructor, getters y setters.



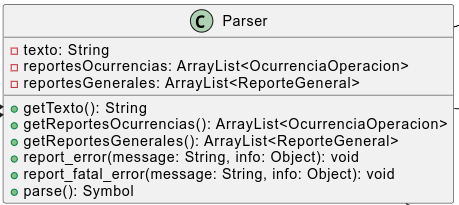
### Lexer

Esta clase contiene los métodos necesarios para obtener los tokens, esta clase es generada en base a un archivo Lexer.flex que contiene las reglas para los tokens.



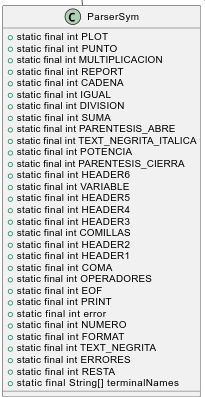
### Parser

Esta clase es generada en base a un archivo llamado Parser.cup, en ese archivo se crean las gramáticas que serán utilizadas para que el programa acepte todo correctamente.



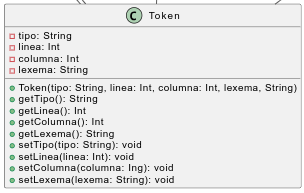
### ParserSym

Esta clase también es generada por archivo Parser.cup, est únicamente conteiene enums con los tokens que se encontraron durante el análisis léxico.



### Token

Esta clase sirve de modelo para guardar los tokens encontrados en el lexer y los guarda en un arraylist, únicamente contiene atributos, constructor, getters y setters.

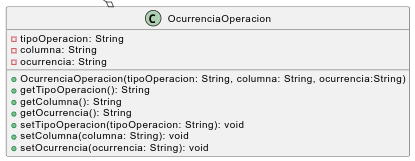


## Paquete reportes

Este paquete contiene dos clases kotlin, se encargan de crear los modelos para ser utilizados en un arraylist el cual almacenan diferentes tipos de datos.

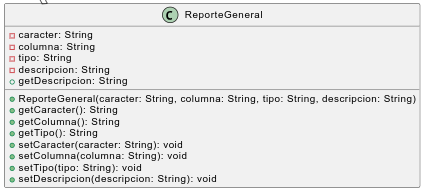
### OcurrenciaOperacion

Esta clase se encarga de ser un modelo el cual se almacena en un arraylist donde se guarda cada vez que existe una operación aritmética.



### ReporteGeneral

Esta clase es un modelo donde se utiliza en un arraylist que almacena cada vez que se encuentra un error léxico o sintáctico.



# Paquete res

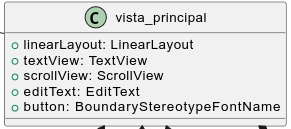
Este paquete contiene los archivos necesarios para el frontend de la aplicación.

## Paquete layout

Este paquete se encarga de tener las vistas que tendrá la aplicación móvil.

### vista\_principal

Este es un archivo, en el diagrama de clases aparece como clase, pero realmente es un archivo xml que contiene etiquetas que funcionan para crear los componentes que tendrá la vista que verá el usuario, en la imagen se muestra atributos con fines visuales para saber los componentes que se utilizaron.



# Gramáticas

Las gramáticas utilizadas en el parser.cup son las siguientes:

## Terminales:

Los terminales son los tokens donde las gramáticas terminarán, es decir, al ingresar algún texto estos se comparan con los terminales para definir que el texto ingresado es correcto, las terminales definidas para este analizador son las siguientes:

cadena, numero, suma, resta, multiplicación, división, parentesis\_abre, parentesis\_cierra, potencia, igual, header1, header2, header3, header4, header5, header6, text\_negrita, text\_negrita\_italica, comillas, coma,