Nueva Guatemala de la Asunción

Manual de Técnico

Proyecto 1: Pixel Art

Elaborado por: Estuardo Gabriel Son Mux

Carné: 202003894

Fecha: 11/09/2021

# Alcances y Objetivos del Programa

El programa realizado tiene como objetivo el análisis de los datos ingresados en un archivo \*.pxla el cual contiene la información para la creación de una o varias imágenes simulando pixel art, así mismo la creación de un archivo html que contiene la lista de tokens y errores que se presenten durante el análisis del archivo.

Esto se logra mediante el análisis léxico del contenido del archivo y la utilización de la librería cairosvg para la creación de las imágenes.

# Especificaciones Técnicas

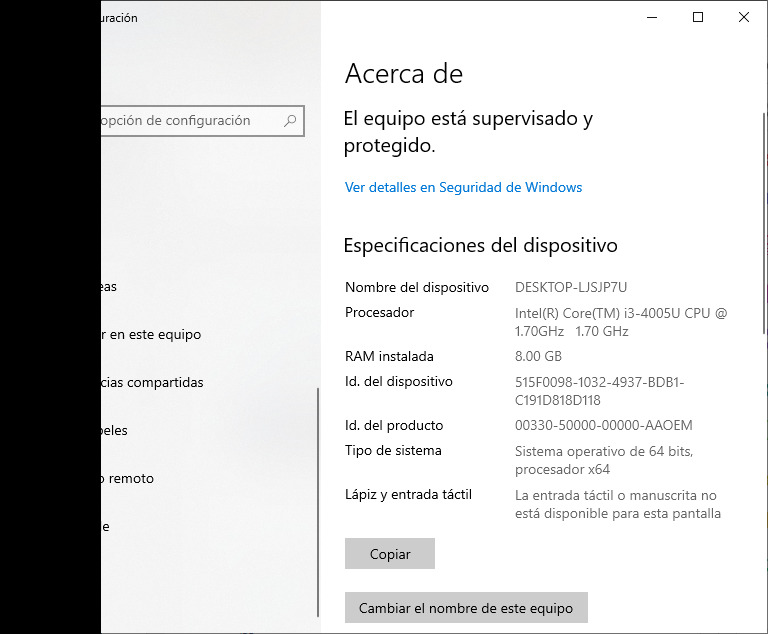
## Requisitos de Hardware

* Computadora con todos sus componentes para su correcto funcionamiento
* Laptop

## Requisitos de Software

### Sistema Operativo

El programa fue desarrollado en una laptop con sistema operativo Windows 10, 8GB de ram, Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz 1.70 GHz.



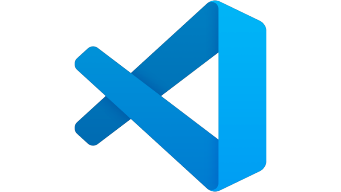
### Lenguaje de Programación

Python

Librería Cairosvg

### IDE o Editor de Código

Visual Estudio



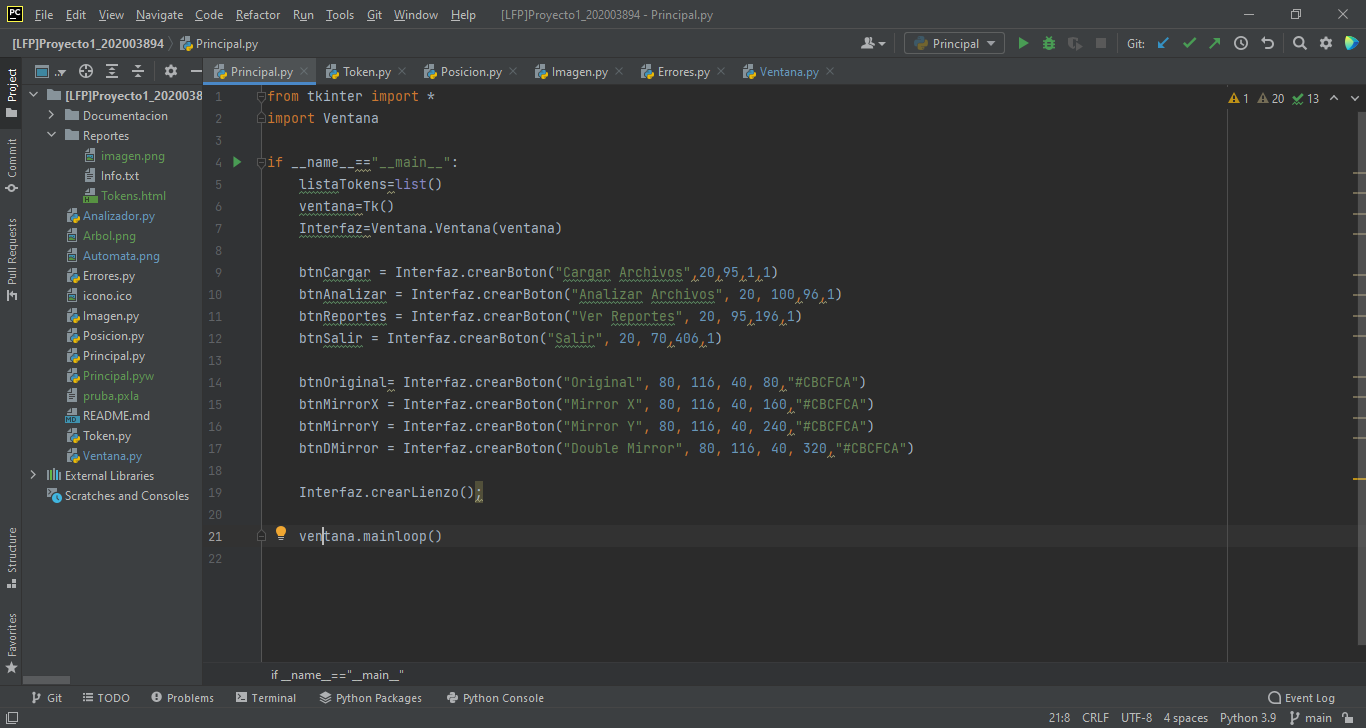
# Lógica del Programa

## Archivos Utilizados

El programa consta de siete archivos para el funcionamiento del mismo. Así mismo, cuenta con una imagen \*.ico el cual es el icono de la ventana y una carpeta llamada Reportes donde se almacena el reporte Html y la imagen generada.

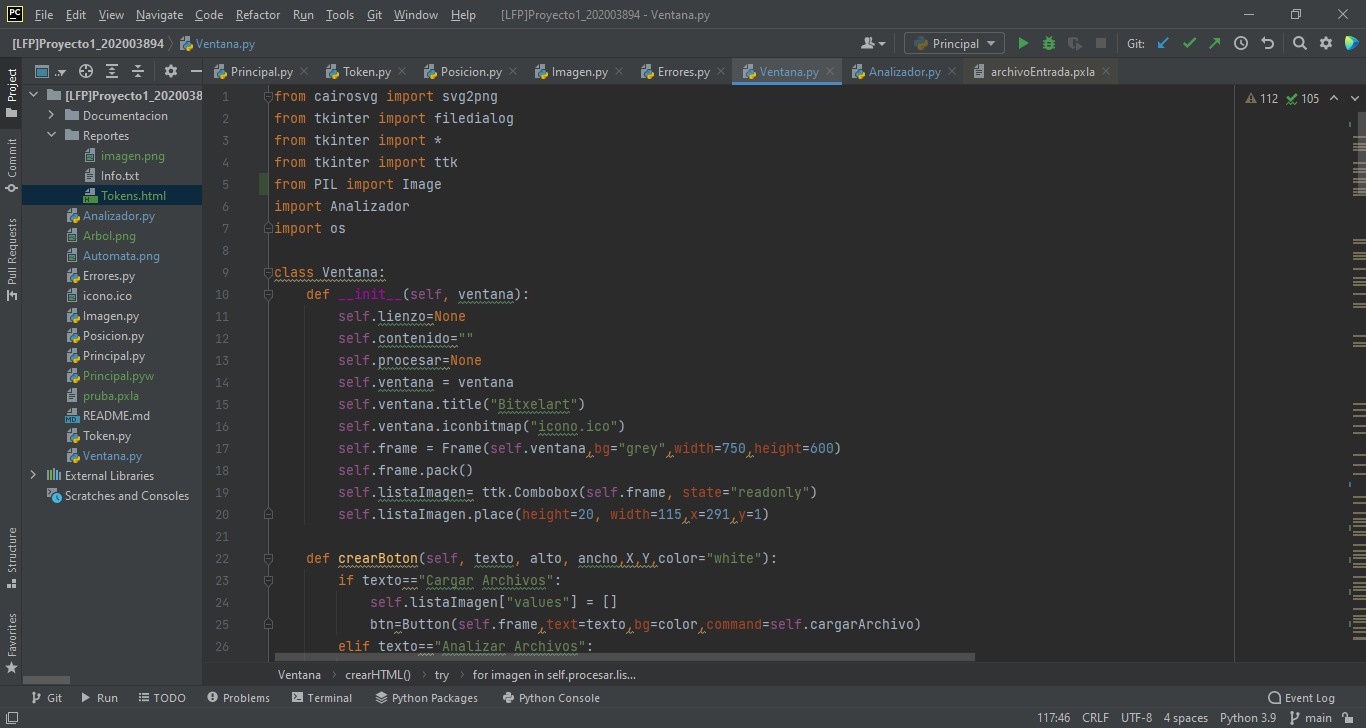
### Principal.py

En este archivo se encuentra inicializada la clase de ventana para que sea posible su visualización, así como la implementación de procesos de la clase Ventana.

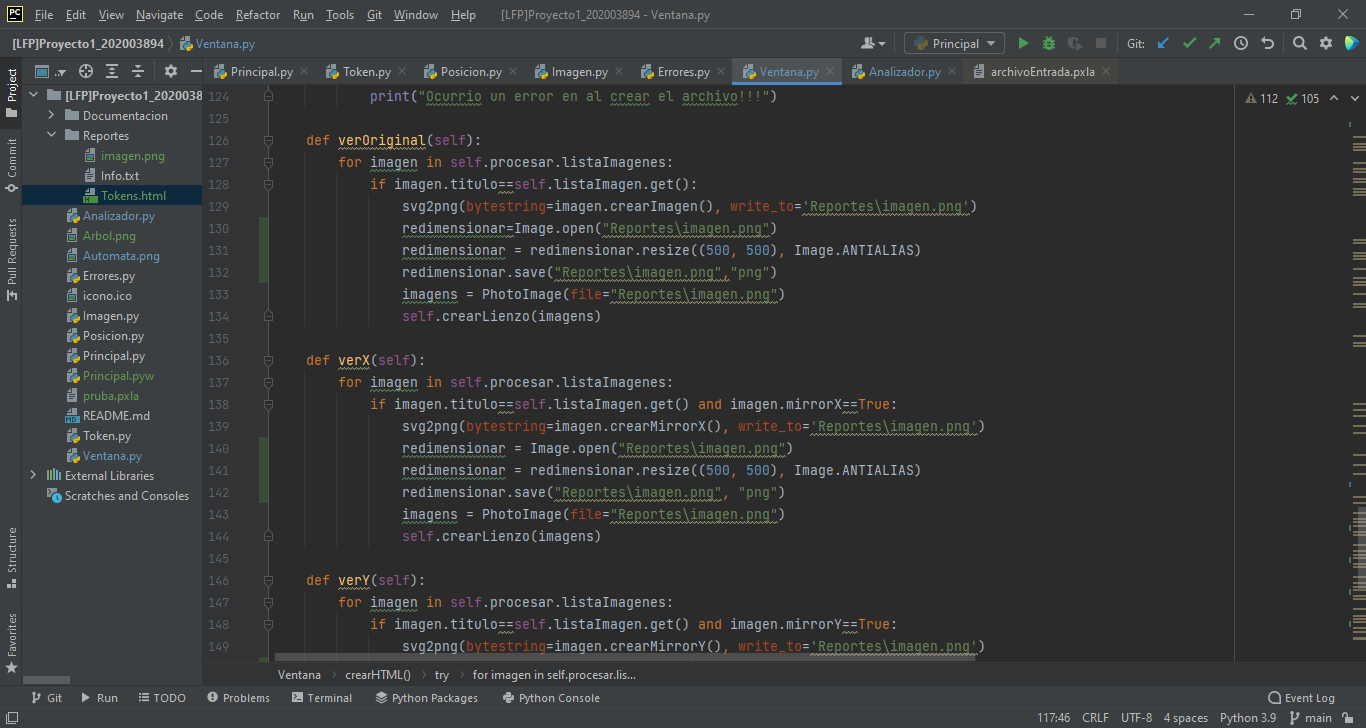


### Ventana.py

En este archivo se encuentran todas las configuraciones de la ventana principal del programa.



En este archivo también se encuentran las configuraciones de los distintos botones que se encuentran en la ventana, como ejemplo la creación del archivo Html y la visualización de las imágenes dentro de la aplicación.



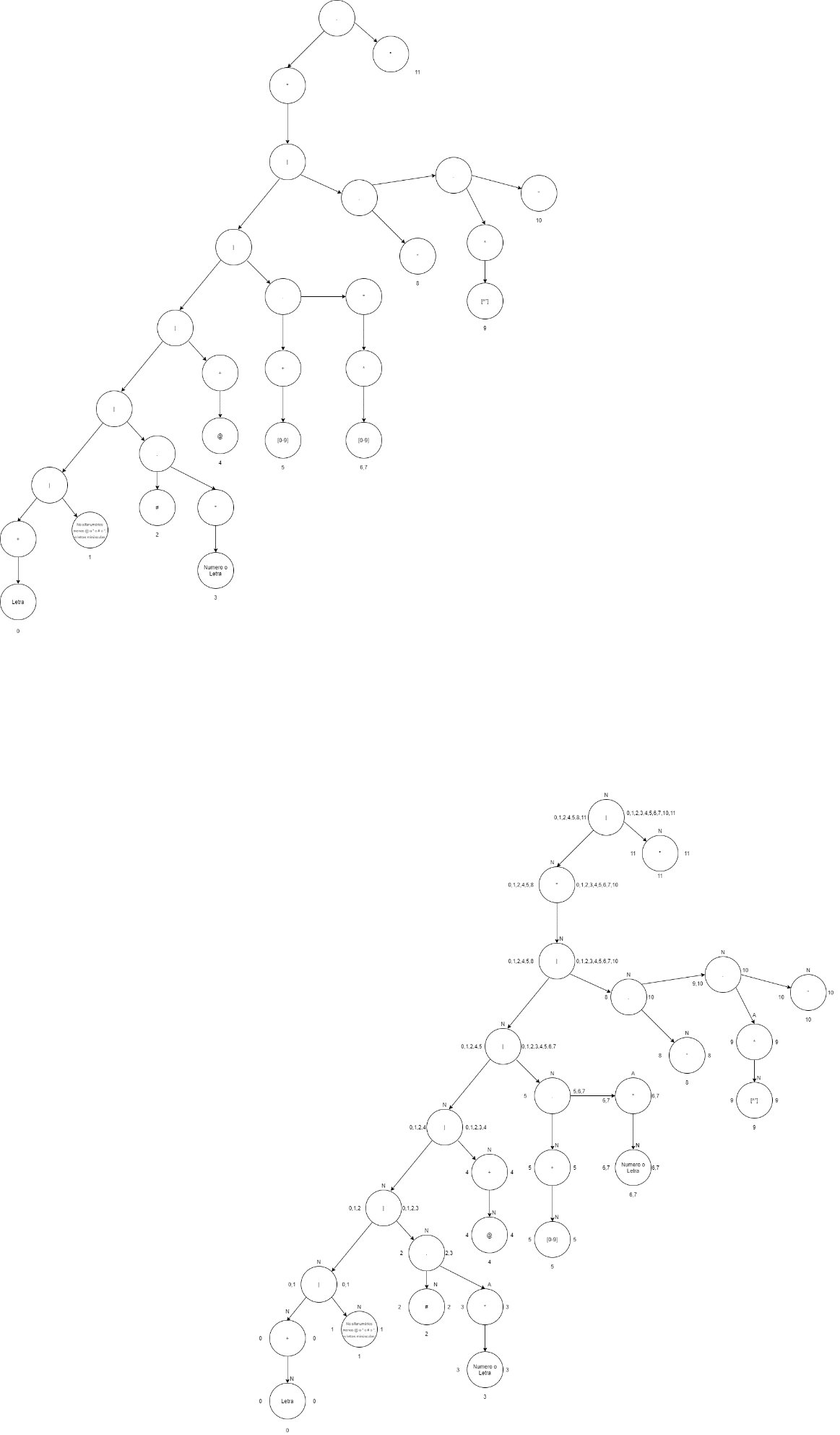
### Analizador.py

En este archivo se encuentran los procesos encargados del análisis léxico del archivo que sea ingresado, reconociendo los tokens que formen parte del lenguaje o caracteres que no pertenezcan al lenguaje. Así mismo se encarga de crear los objetos de tipo imagen y almacenarlos en una lista.

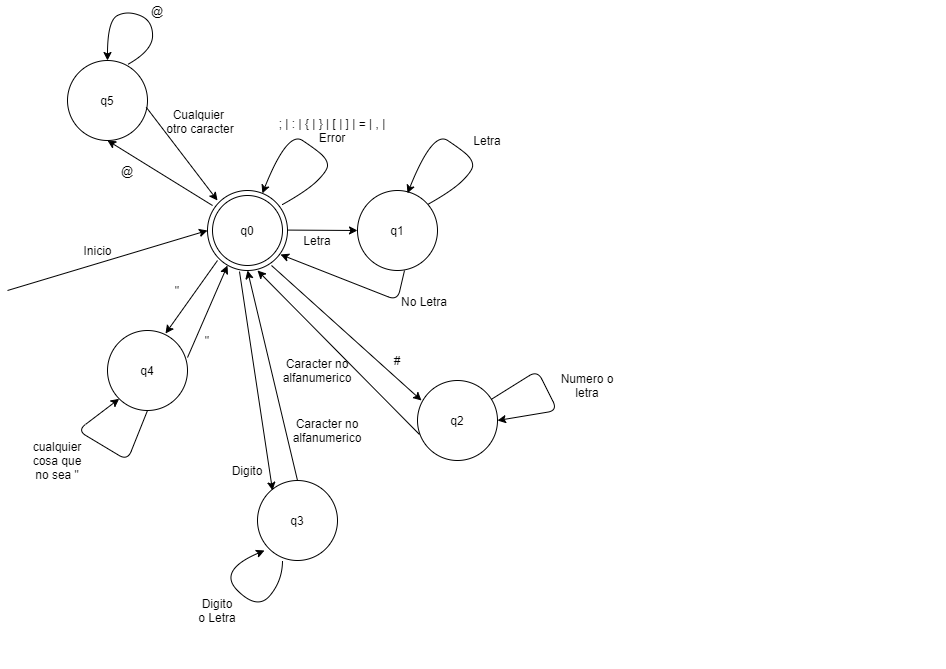
Los tokens que reconoce el analizador se presentan en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
| Token | Lexema | Patrón |
| TITULO | TITULO | TITULO |
| ALTO | ALTO | ALTO |
| ANCHO | ANCHO | ANCHO |
| FILAS | FILAS | FILAS |
| COLUMNAS | COLUMNAS | COLUMNAS |
| CELDAS | CELDAS | CELDAS |
| FILTROS | FILTROS | FILTROS |
| Filtro | DOUBLEMIRROR | MIRRORX | MIRRORY | DOUBLEMIRROR |
| Llave Agrupación | { | { | } |
| Corchetes | [ | [ | ] |
| Punto y Coma | ; | ; |
| Coma | , | , |
| Entero | 30 | [0-9]+ |
| Separador de Imagen | @@@@ | @{4} |
| String | “Pokeball” | "[^"]\*" |
| Igual | = | = |
| Booleano | TRUE | TRUE | FALSE |
| Codigo Color Hex | #92D050 | #[a-fA-F0-9]{6} |

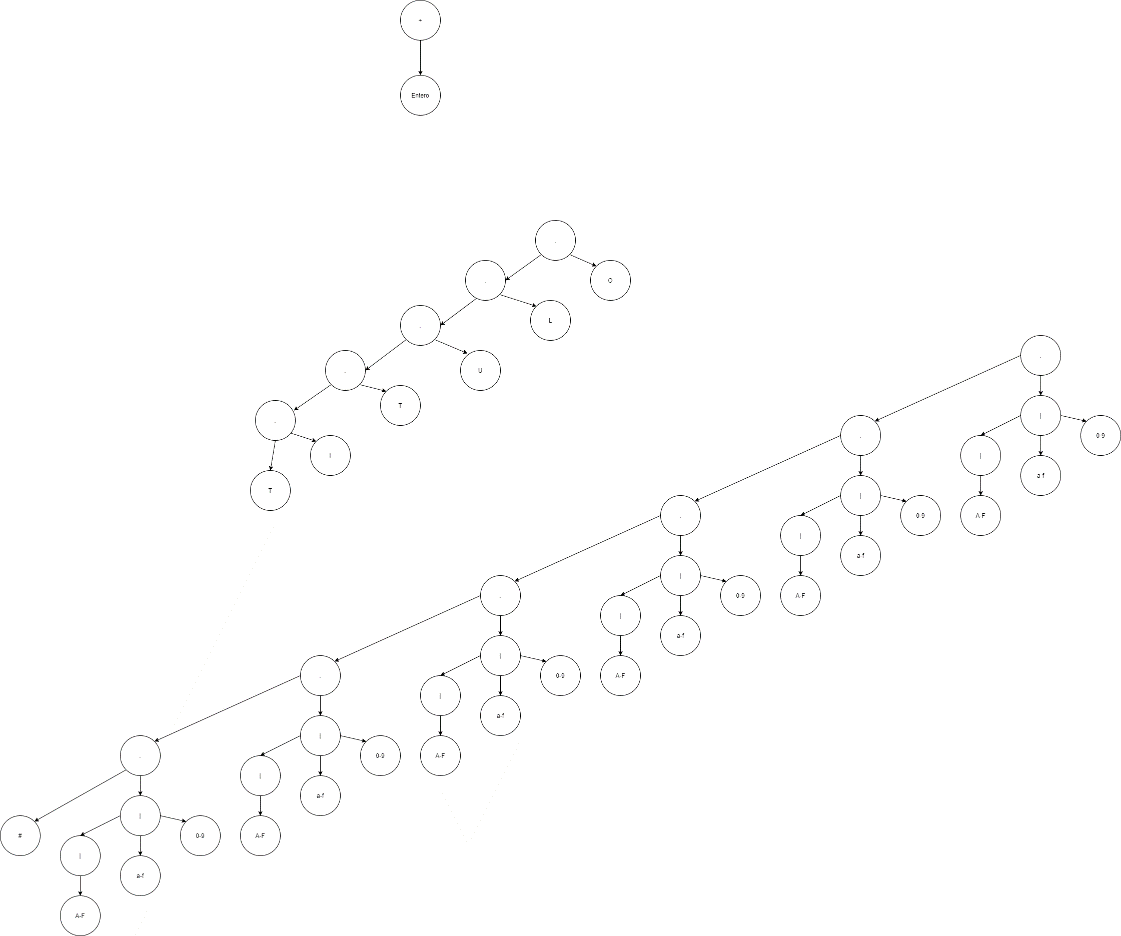
Para la elaboración del autómata se utilizó el siguiente diagrama de árbol:



El árbol anteriormente mostrado dio como resultado el siguiente autómata:



Sin embargo, este autómata aún no cuenta con lo necesario para cada reconocer los tokens del lenguaje por lo que mediante un análisis con comparaciones con condicionales if y que cumplan con la forma del token:



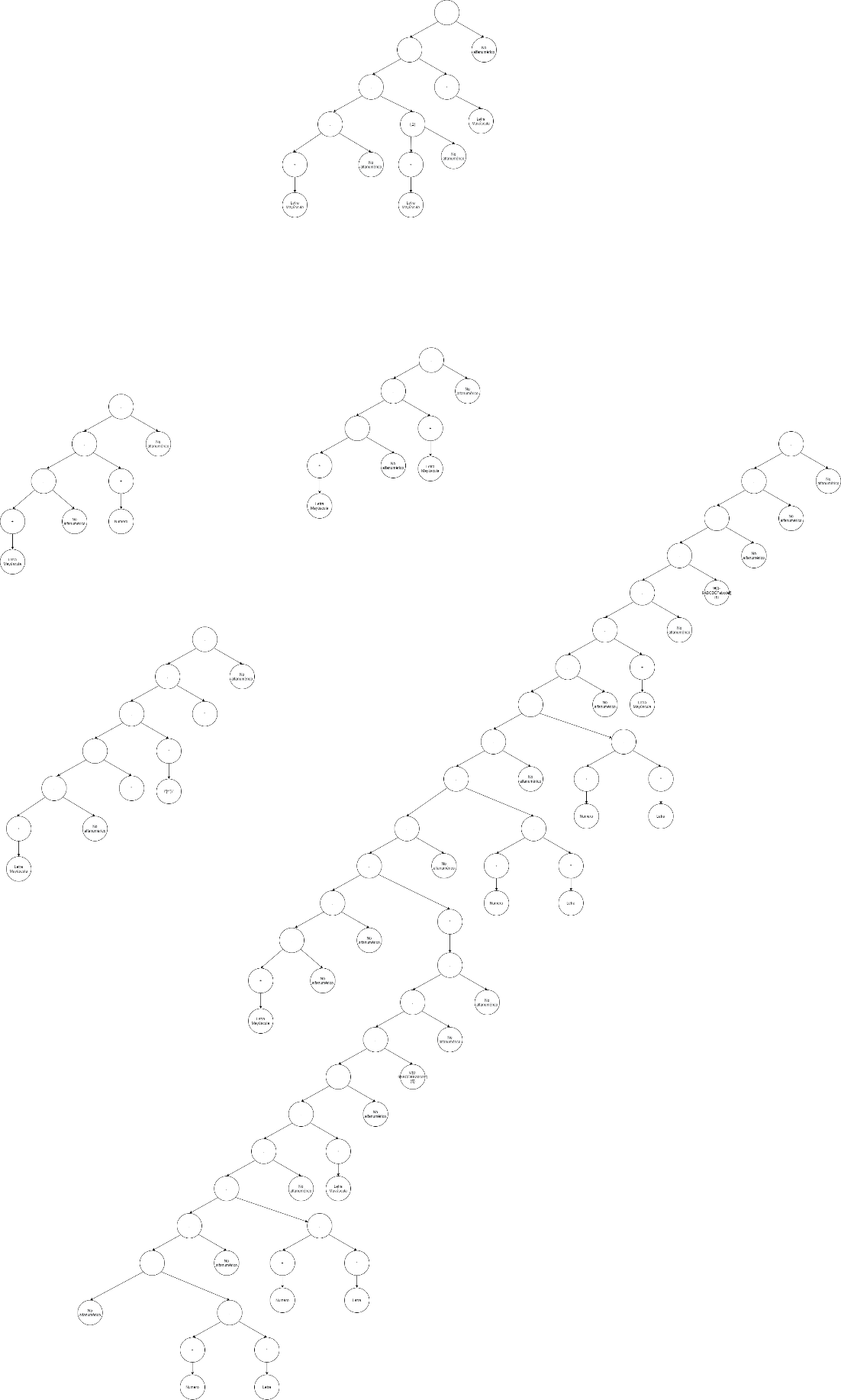
Codigo color Hex

Titulo

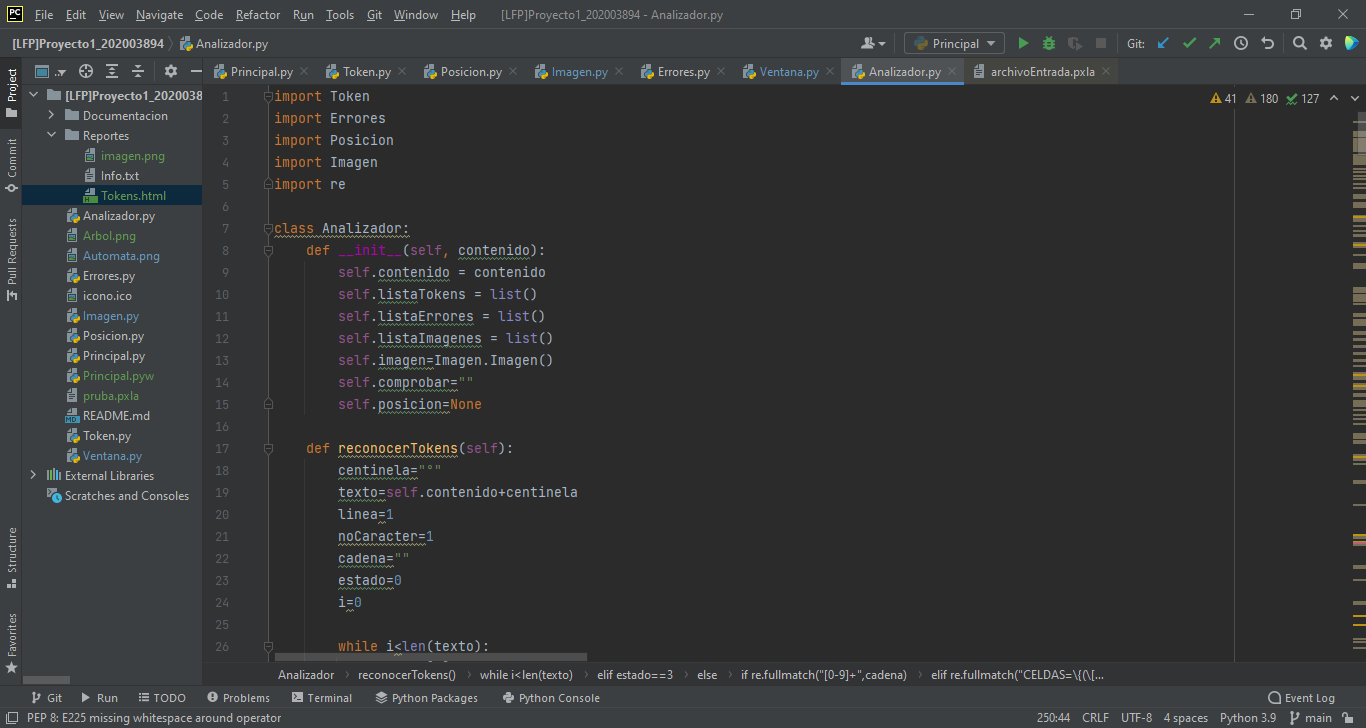
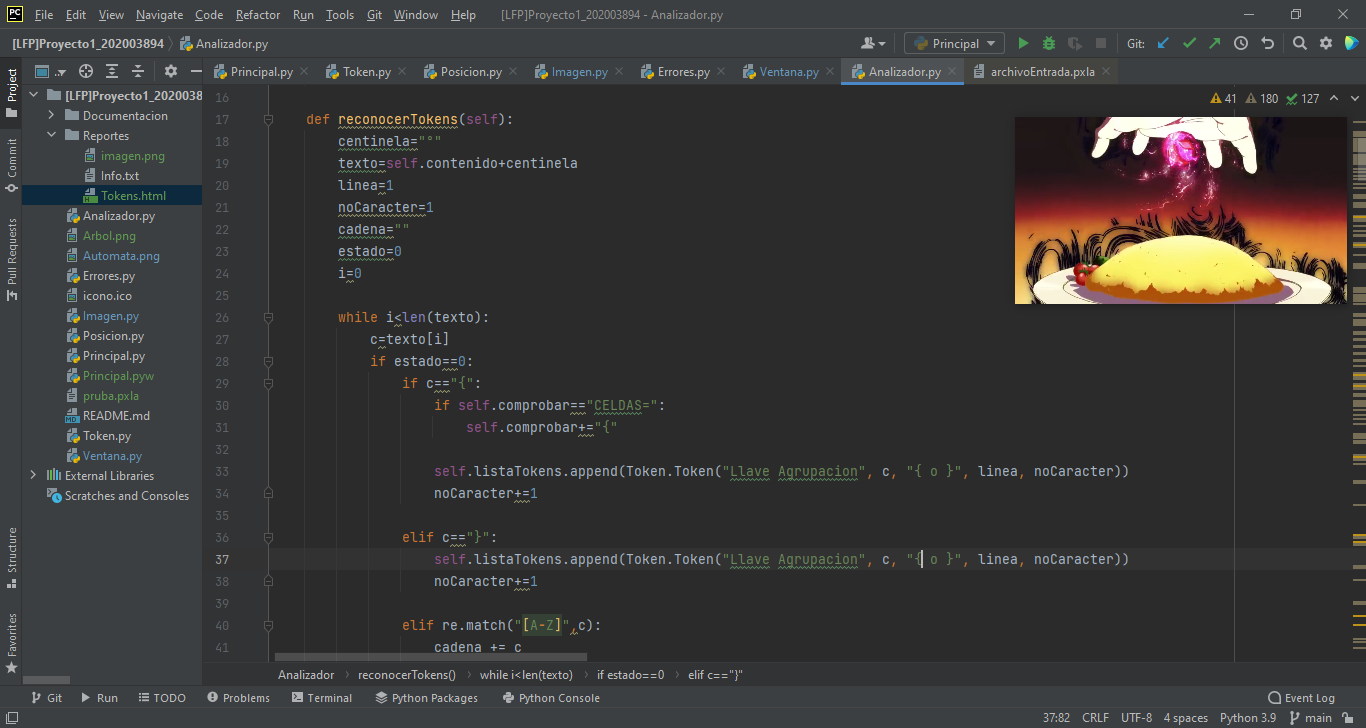
Entero

*Ejemplo de Estructura de árbol de los tokens*

Ya para reconocer los datos que serán utilizados para la creación de los datos de la imagen durante el análisis léxico al reconocer un token si se cumple con la estructura que se encuentra en las expresiones regulares de las condicionales IF:



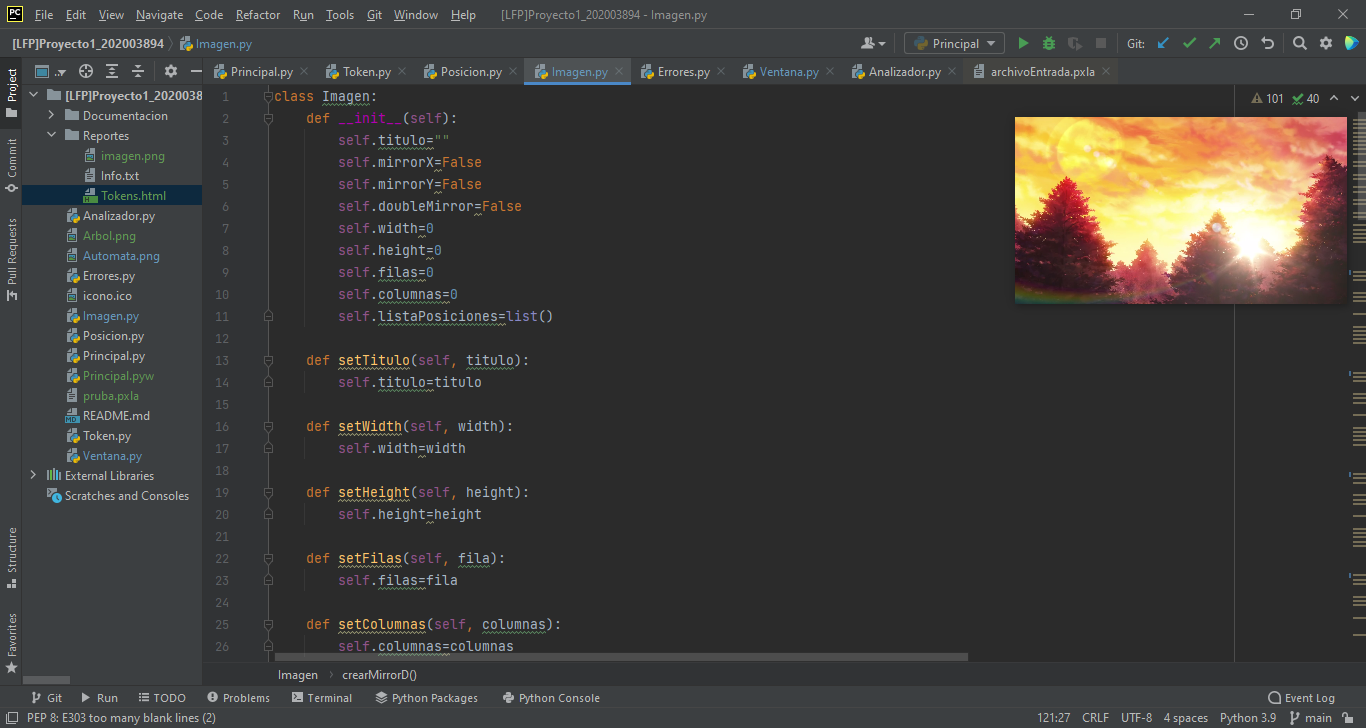
*Estructura expresiones regulares para recolección de datos*

*Parte del código de la Clase Analizador*

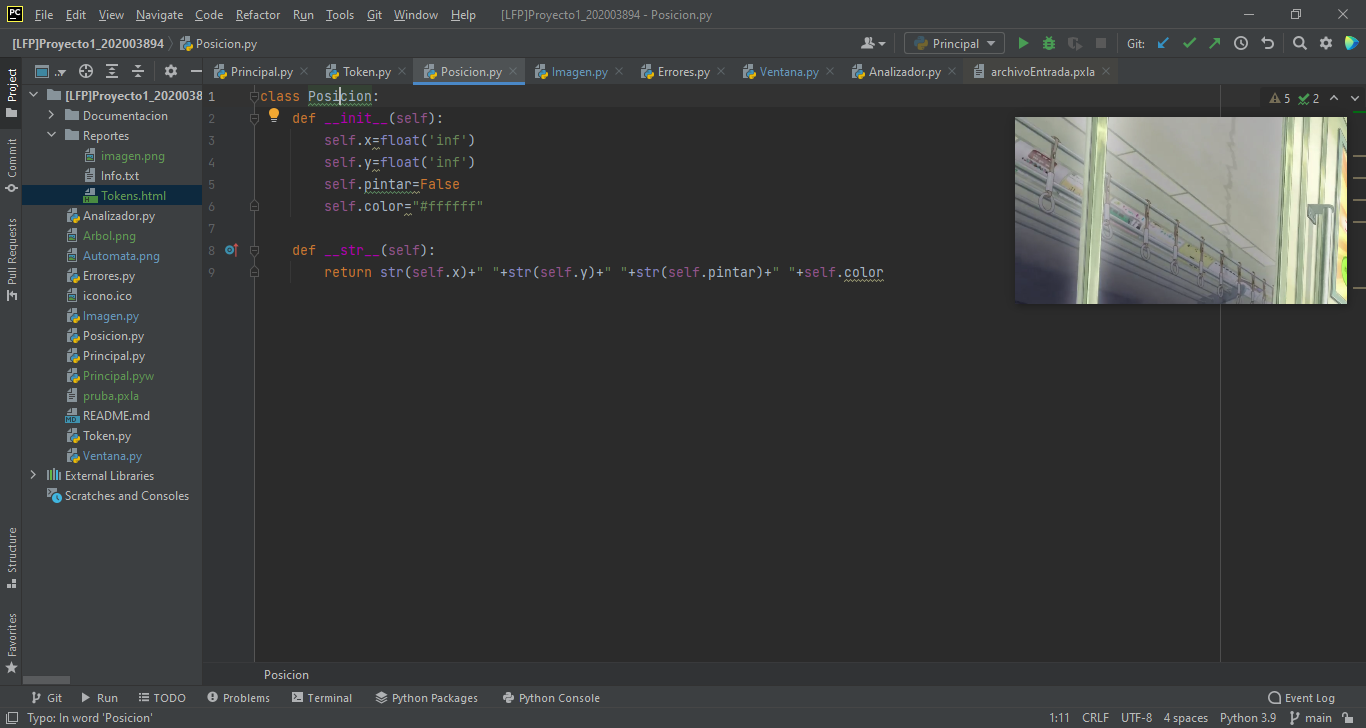
### Imagen.py

Esta clase es de utilidad para la creación de objetos de tipo imagen almacenando sus datos de nombre, celdas, alto, ancho, filas y columnas, también se incluyen los filtros aplicables.



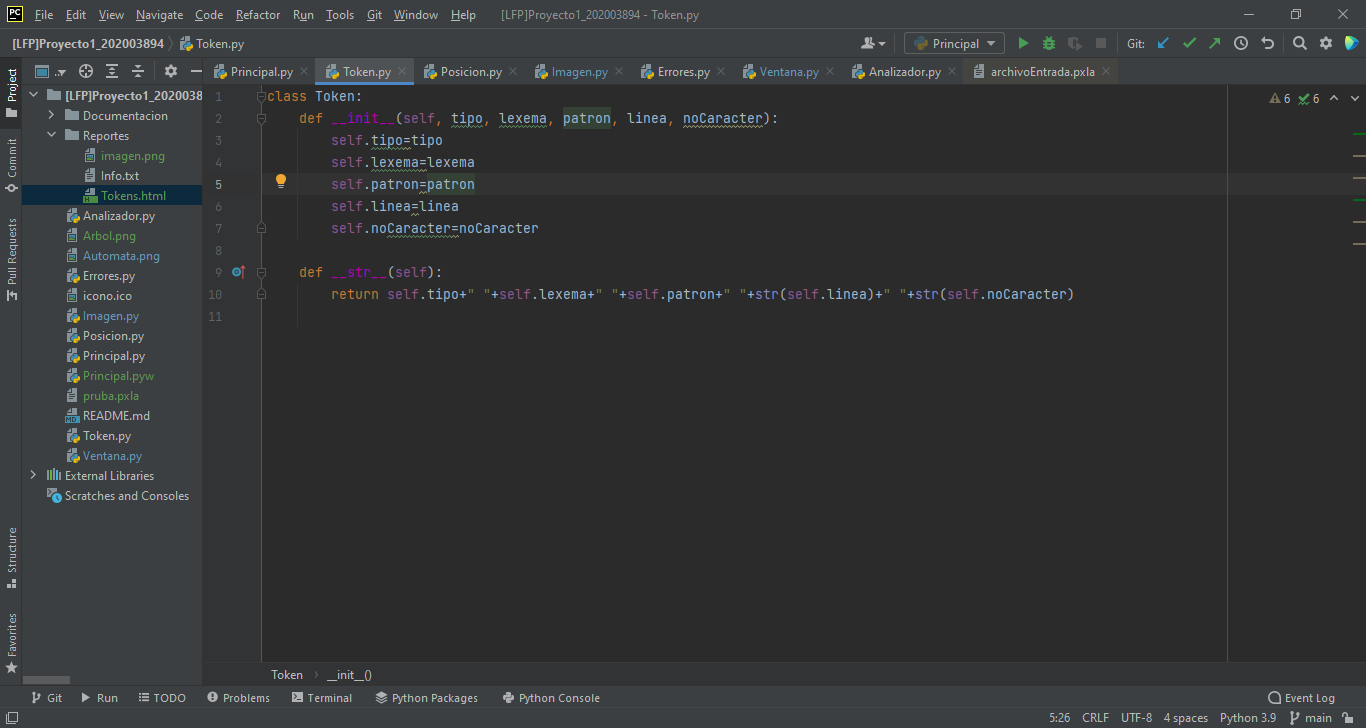
### Posicion.py

Esta clase almacena las posiciones de cada celda, los colores y si la verificación de si debe ser o no pintada.



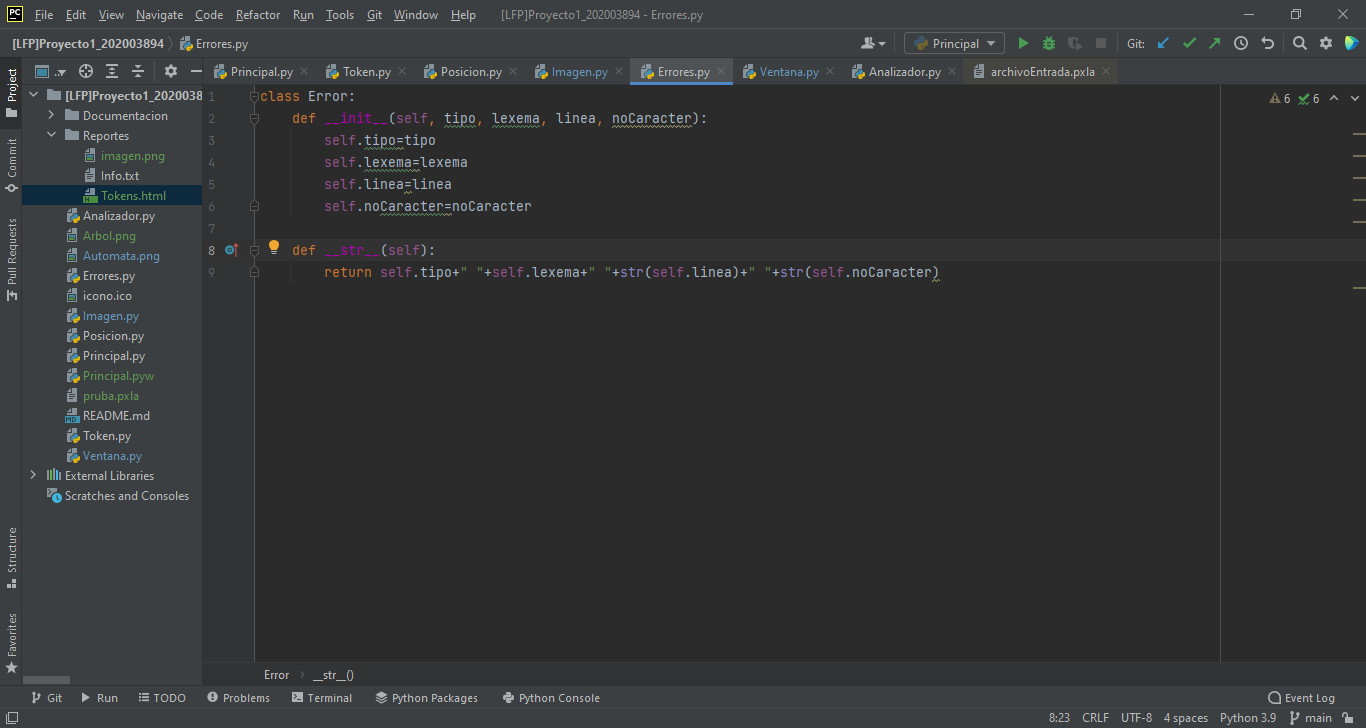
### Token.py

Después de reconocer un token el analizador léxico de la clase analizador crea un objeto de la clase token que contiene el token reconocido, el patrón al que pertenece, la fila y la posición en la que se encuentra dentro de la fila.



### Errores.py

En el caso de que el analizador léxico detecte una cadena o carácter que no pertenezca al lenguaje, se creara un objeto de la clase errores que contiene la posición y fila en la que se encuentra.



# Flujo del Programa

Se inicia el programa con una ventana principal la cual es invocada en el archivo Principal.py y se forma con la clase Ventana, es atreves de esta que se llega a las funciones las diferentes funciones del programa.

La función de carga de archivo accede la clase Analizador.

En la clase Analizador se crean objetos a partir de las clases Imagen, Posiciones, Token y Errores los cuales almacenan los datos de las imágenes especificadas en el archivo de entrada.

La clase Imagen contiene en sus atributos una lista de posiciones que será la encargada de almacenar los objetos Posiciones creados por la clase Analizador.