202111718

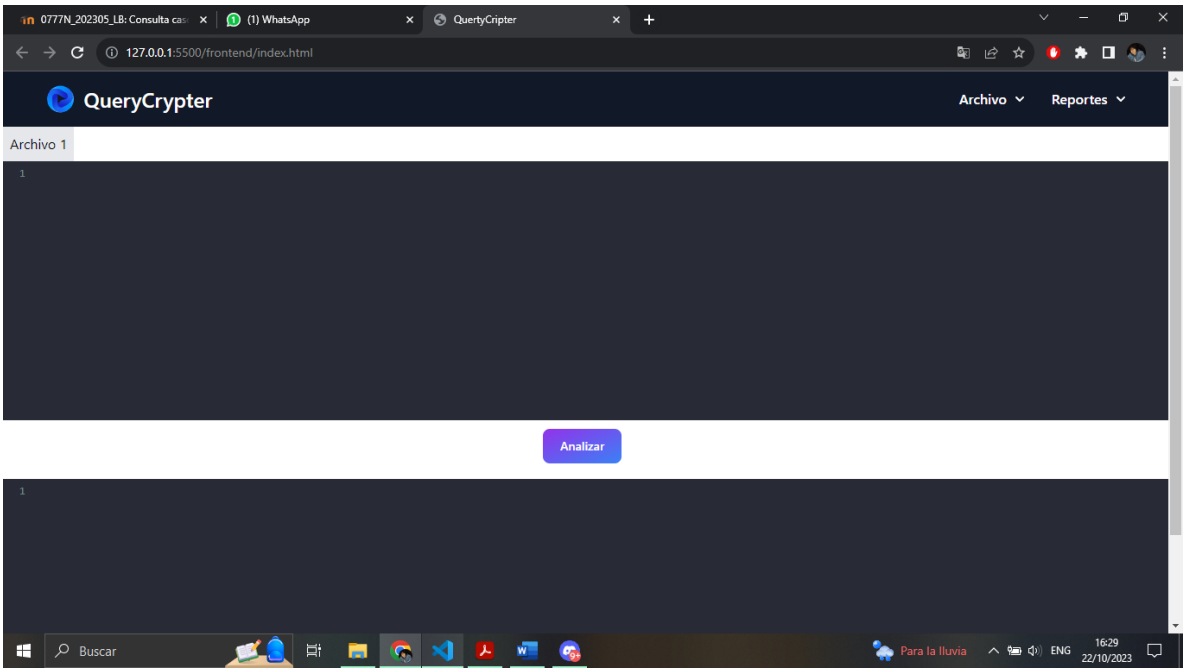
Manual de Usuario

Proyecto 2 Compi 1

Walter Javier Santizo Mazariegos

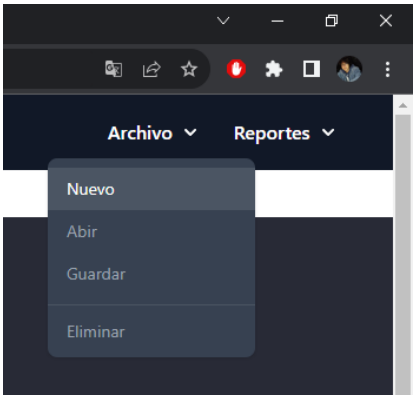
Entorno de Trabajo

Son todas las herramientas desde el editor de texto hasta la consola de salida, la pagina web se debe de ver de la siguiente forma:

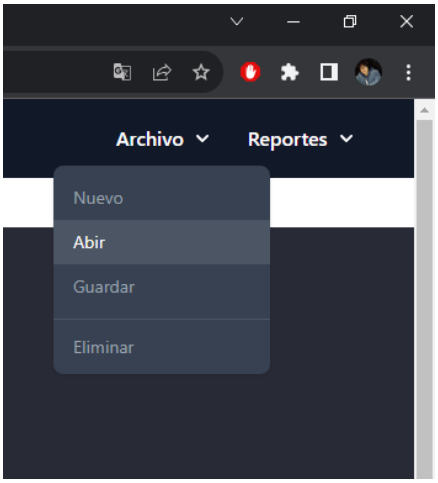


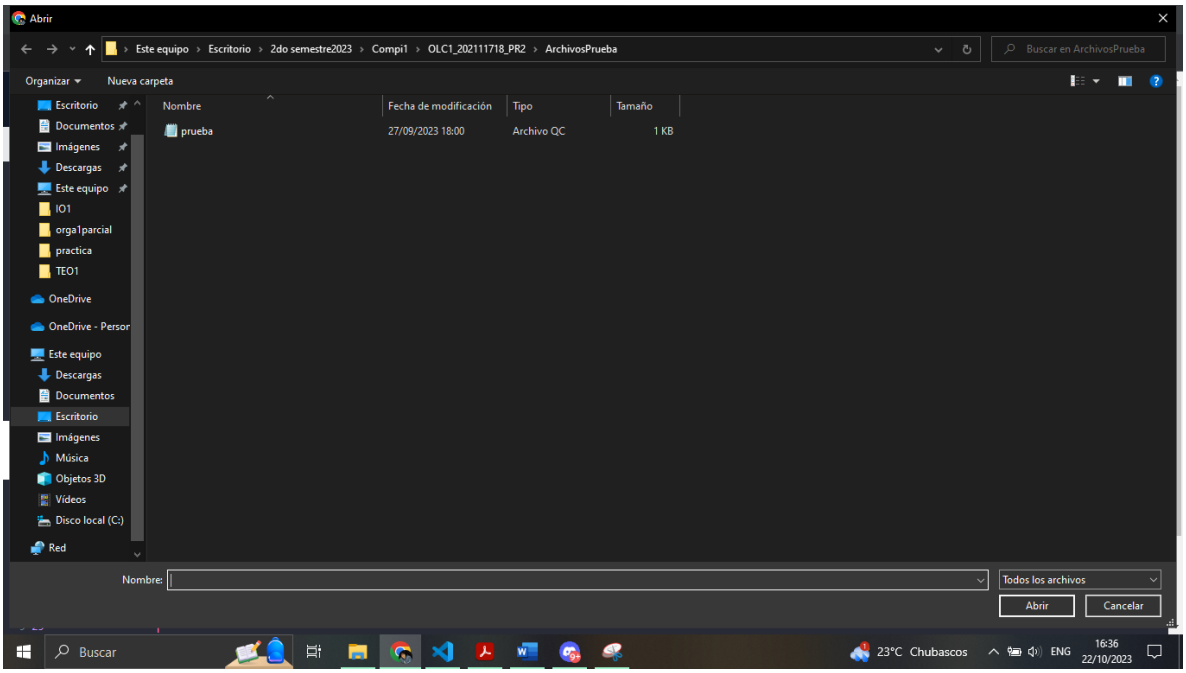
Funcionalidades

- Nuevo Archivo: El editor debe tiene la capacidad de crear archivos en blanco el cual podrá ser editado en una pestaña que tiene el nombre de la tab donde se abrió.

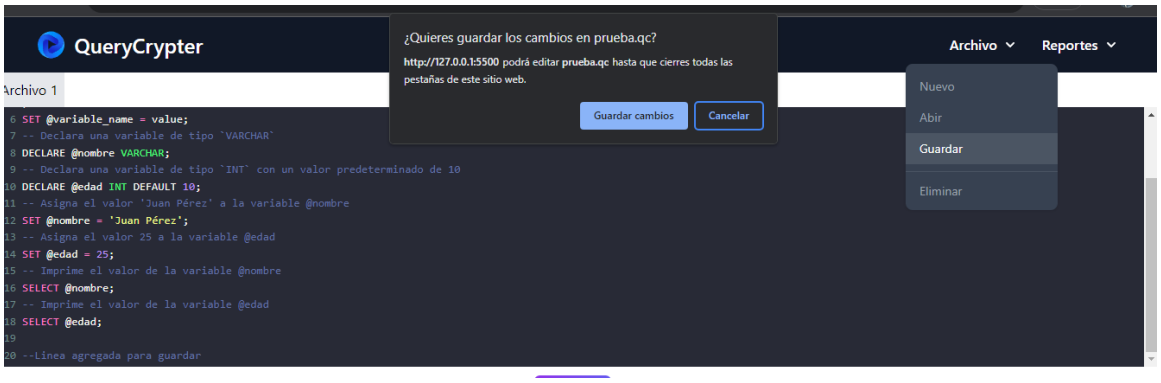


- Abrir Archivo: El editor tiene la capacidad de abrir archivos con las extensiones. qc cuyo contenido se mostrará en el área de entrada en una nueva pestaña con el nombre del archivo.

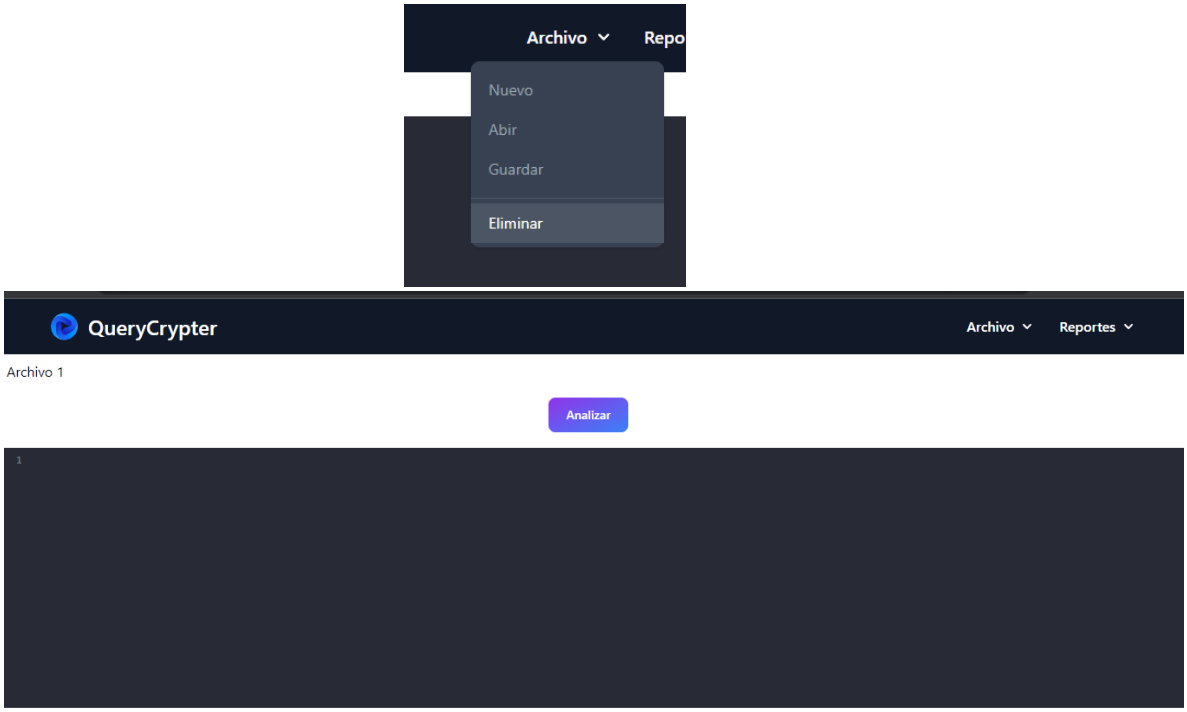




- Guardar: El editor debe tener la capacidad de guardar el estado del archivo en el que se estará trabajando.



- Eliminar Pestaña: Cada pestaña puede ser cerrada en cualquier momento. Si los cambios no se han guardado se descartan.



Ejecutar

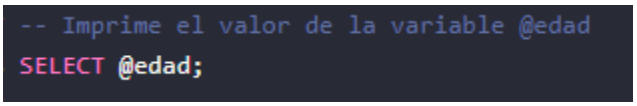
Se envía la entrada de la pestaña actualmente seleccionada al intérprete con la finalidad de realizar el análisis léxico, sintáctico y la ejecución de las instrucciones.

Analizar

Instrucciones Aceptadas

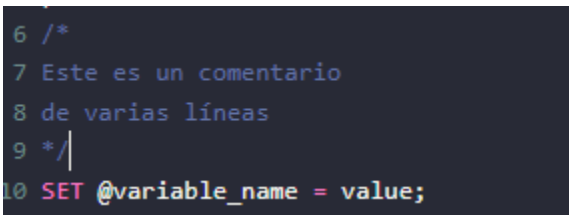
Comentarios de una línea

Los comentarios de una línea comienzan con el símbolo -- y terminan con el final de la línea. Por ejemplo, el siguiente código es un ejemplo de un comentario de una línea:



Comentarios de varias líneas

Los comentarios de varias líneas comienzan con los símbolos /* y se extienden hasta los símbolos */. Por ejemplo, el siguiente código es un ejemplo de un comentario de varias líneas



Tipos de datos

El lenguaje QueryCrypter admite una variedad de tipos de datos, que se pueden utilizar para almacenar diferentes tipos de datos

Tipo	Definición	Descripción	Ejemplo	observaciones
Enteros	int	Los tipos de datos enteros se utilizan para almacenar números enteros.	123, 568, 10, etc	Se maneja cualquier cantidad de dígitos.
Reales	double	Los tipos de datos reales se utilizan para almacenar números reales.	1.2, 50.23, 00.34, etc.	Se maneja cualquier cantidad de decimales
Date	date	Los tipos de datos de fecha se utilizan para almacenar valores de fecha.	2023-07-20	El formato de la fecha es YYYY-MM-DD
Cadena de caracteres	varchar	Los tipos de datos de cadena de caracteres se utilizan para almacenar cadenas de caracteres	"Hola mundo"	Cadena de caracteres de longitud variable,Se permitirá cualquier carácter entre las comillas dobles,incluyendo las secuencias de escape: \" comilla doble \\ barra invertida \n salto de linea \r retorno de carro \t tabulación
BOOLEAN	TRUE / FALSE.	El tipo de datos BOOLEAN se utiliza para almacenar valores	TRUE, FALSE	
		lógicos.		
NULL	NULL	El tipo de datos NULL se utiliza para representar un valor desconocido o no aplicable.	NULL	

Secuencias de Escape

El lenguaje QueryCrypter admite secuencias de escape para evitar que los caracteres especiales se interpreten incorrectamente. Las secuencias de escape se utilizan para definir ciertos caracteres especiales dentro de cadenas de texto.

Secuencia	Descripción	Ejemplo
\n	Salto de línea	“Hola\nMundo”
\\	Barra invertida	“C:\\miCarpeta\\Personal”
\“	Comilla doble	“\”Esto es una cadena\””
\t	Tabulación	“\tEsto es una tabulación”
\’	Comilla Simple	“\’Estas son comillas simples\’”

Operadores Aritméticos

Suma

Es la operación aritmética que consiste en realizar la suma entre dos o más valores. El símbolo a utilizar es el signo más +.

+	Int	Double	Date	Varchar	Boolean	Null
Int	int	Double	Date	Int		
Double	Double	Double	Date	Double		
Date	Date	Date		Date		
Varchar	Int	Double	Date	Varchar		
Boolean						
Null						

Resta

Es la operación aritmética que consiste en realizar la resta entre dos o más valores. El símbolo por utilizar es el signo menos –

-	Int	Double	Date	Varchar	Boolean	Null
Int	int	Double	Date	Int		
Double	Double	Double	Date	Double		
Date	Date	Date		Date		
Varchar	Int	Double	Date	Varchar		
Boolean						
Null						

Multiplicación

Operación aritmética que consiste en sumar un número (multiplicando) tantas veces como indica otro número (multiplicador). El sino para representar la operación es el asterisco *

*	Int	Double	Date	Varchar	Boolean	Null
Int	int	Double				
Double	Double	Double				
Date						
Varchar						
Boolean						
Null						

División

Operación aritmética que consiste en partir un todo en varias partes, al todo se le conoce como dividendo, al total de partes se le llama divisor y el resultado recibe el nombre de cociente. El operador de la división es la diagonal /.

/	Int	Double	Date	Varchar	Boolean	Null
Int	int	Double				
Double	Double	Double				
Date						
Varchar						
Boolean						
Null						

Módulo

Es una operación aritmética que obtiene el resto de la división de un número entre otro. El signo a utilizar es el porcentaje %.

%	Int	Double	Date	Varchar	Boolean	Null
Int	int	Double				
Double	Double	Double				
Date						
Varchar						
Boolean						
Null						

Negación Unaria

Es una operación que niega el valor de un número, es decir que devuelve el contrario del valor original.

-num	Int
Int	int
Double	Double
Date	
Varchar	
Boolean	
Null	

Operadores Relacionales

Los operadores relacionales en SQL son símbolos o palabras clave utilizados para comparar valores en una consulta o condición, y así determinar si se cumple o no una relación o comparación entre esos valores

Secuencia	Descripción	Ejemplo
=	Igualación: Compara ambos valores y verifica si son iguales.	SELECT * FROM productos WHERE precio = 100;
!=	Diferenciación: Compara ambos lados y verifica si son distintos.	SELECT * FROM productos WHERE precio != 100;
<	Menor que: Compara ambos lados y verifica si el derecho es mayor que el izquierdo.	SELECT * FROM productos WHERE precio < 100;
<=	Menor o igual que: Compara ambos lados y verifica si el derecho es mayor o igual que el izquierdo.	SELECT * FROM productos WHERE precio <= 100;

>	Mayor que: Compara ambos lados y verifica si el izquierdo es mayor que el derecho.	SELECT * FROM productos WHERE precio > 100;
>=	Mayor o igual que: Compara ambos lados y verifica si el	SELECT * FROM productos WHERE precio >= 100;
	izquierdo es mayor o igual que el derecho.	

Operadores Lógicos

Los operadores lógicos sirven para combinar dos valores booleanos y dar un retorno del resultado, sea verdadero o falso.

Secuencia	Descripción	Ejemplo
AND	Es el “y” lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.	SELECT * FROM productos WHERE precio > 100 AND nombre = 'Producto 1';
OR	Es el “o” lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.	SELECT * FROM productos WHERE precio > 100 OR nombre = 'Producto 1';
NOT	Es el “o” lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta..	SELECT * FROM productos WHERE NOT nombre = 'Producto 1';

Declaración de Variable

Las variables nos permiten almacenar un valor y recuperarlo más adelante para emplear en otras sentencias. Pueden utilizarse para almacenar valores temporales, para realizar cálculos complejos o para simplificar la escritura de consultas.

```
1 -- Declaración de una variable
2 declare @nombre varchar;
3 -- Declaración de varias variables
4 declare @nombre varchar, @edad int;
```

Una vez declarada una variable, se puede asignar un valor a la variable utilizando la sentencia SET. El valor que se asigna a una variable debe ser del mismo tipo de datos que la variable

```
1 -- Asigna el valor 'Juan Pérez' a la variable @nombre
2 SET @nombre = 'Juan Pérez';
3 -- Asigna el valor 25 a la variable @edad
4 SET @edad = 25;
```

4.9.1 Create Table

Dado que QueryCrypter es un lenguaje basado en SQL el proyecto deberá de ser capaz de crear estructuras de datos simulando tablas de n dimensiones para almacenar datos ejemplo de la estructura para crear una tabla de datos

```
1 CREATE TABLE Clientes
2 (
3 ID_Cliente INT,
4 Nombre VARCHAR,
5 CorreoElectronico VARCHAR
6 );
```

Alter Table

Con Alter table QueyCrypter debe ser capaz de modificar la estructura de una tabla existente, como agregar, eliminar o modificar el nombre de las

```
ALTER TABLE Clientes
ADD Salario double;

ALTER TABLE Clientes
DROP COLUMN Salario;

ALTER TABLE Clientes
RENAME TO NuevoClientes;

ALTER TABLE nombre_de_la_tabla
RENAME COLUMN CorreoElectronico TO
CorreoElectronico1;
```

Drop Table

Con drop table QueyCrypter debe ser capaz de eliminar una tabla y todos sus datos de la estructura de datos

```
DROP TABLE Clientes;
```

Select

Con select deberá recuperar datos de alguna tabla, también será utilizado para obtener valores de alguna variable a su vez podrá filtrar datos basándose en la cláusula while queda a criterio y creatividad de cada uno como presentar el resultado en consola, a continuación se presenta las diferentes estructuras del select

```
select * from Empleados;
```

id	Nombre	Edad	Departamento	Salario
1	Nuevo Nombre	25	Ventas	666
2	Javier	23	Enfermeria	2500
3	Pablo	20	Bodega	2600
4	Nuevo Nombre	21	Ventas	666

```
SELECT Nombre, Edad,id
FROM Empleados
WHERE not id=2;
```

```
Nombre              |Edad               |id                 |
--------------------+-------------------+-------------------+
Nuevo Nombre        |25                 |1                  |
--------------------+-------------------+-------------------+
Pablo               |20                 |3                  |
--------------------+-------------------+-------------------+
Nuevo Nombre        |21                 |4                  |
--------------------+-------------------+-------------------+
```

update

Se utiliza para modificar datos existentes en una tabla. Se podrá actualizar los valores de una o varias columnas en función de una condición específica. Ejemplo de estructura

```
update Empleados
set Nombre="Nuevo Nombre", Salario=666
where Departamento='Ventas';
```

truncate

se utiliza para eliminar todos los registros de una tabla, pero no elimina la estructura de la tabla en sí. Ejemplo de estructura

```
22
23 TRUNCATE TABLE Empleados;
24
```

delete

Se utiliza para eliminar registros de una tabla que cumplan con ciertas condiciones. Ejemplo de la estructura

```
14 DELETE FROM Empleados
15 WHERE Departamento = 'Ventas';
```

If

La sentencia if permite ejecutar una acción si se cumple una condición y otra acción si no se cumple.

```
DECLARE @nota INT;
SET @nota = 70;
IF @nota >= 61 THEN
PRINT 'Ganó el laboratorio';
ELSE
PRINT 'Perdió el laboratorio
END IF;
```

```
Ganó el laboratorio
```

CASE Simple

El CASE simple se utiliza cuando quieres comparar una expresión con varios valores y retornar un resultado específico cuando se cumple una de las condiciones.

Archivo 1

```
DECLARE @nota INT;
SET @nota = 70;
CASE @nota
WHEN 100 THEN 'Sobresaliente'
WHEN 99 THEN 'Muy bueno'
WHEN 98 THEN 'Bueno'
ELSE 'Aprobado'
END;
```

CASE Buscado

El CASE buscado se utiliza cuando deseas realizar evaluaciones condicionales basadas en múltiples condiciones independientes.

```
DECLARE @nota INT;
SET @nota = 10;
CASE
WHEN @nota > 85 THEN 'Excelente'
WHEN @nota >= 61 AND @nota <= 85 THEN 'Aprobado'
ELSE 'No Aprobado'
END;
```

```
No Aprobado
```

While

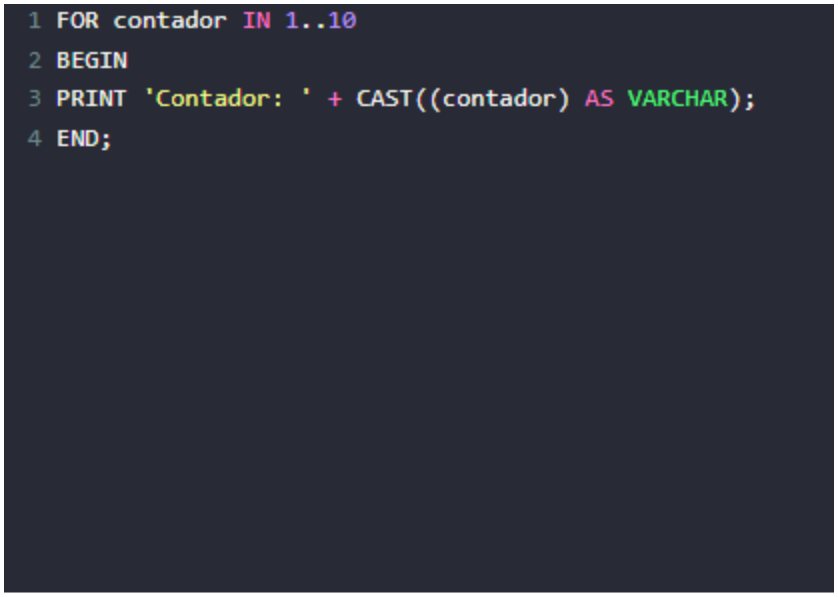
El bucle While se utiliza para ejecutar un bloque de código mientras una condición sea verdadera. La condición se verifica antes de cada ejecución del bloque, y si es verdadera, el bloque se ejecuta; de lo contrario, el bucle se detiene

```
DECLARE @contador INT default 0;
WHILE @contador < 10
BEGIN
PRINT 'Contador: ' + CAST(@contador AS VARCHAR);
SET @contador = @contador + 1;
END;
```

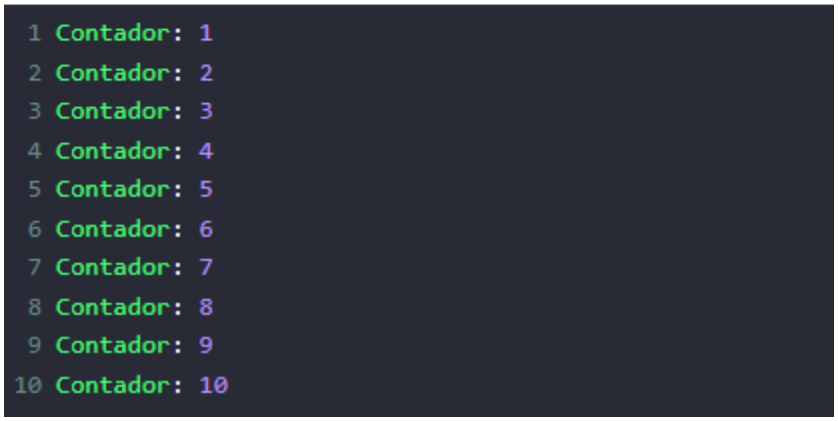
```
Contador: 0
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
Contador: 6
Contador: 7
Contador: 8
Contador: 9
```

For

El bucle FOR se utiliza para iterar a través de un conjunto predefinido de valores o elementos, como una secuencia de números. A diferencia de WHILE, FOR tiene un número fijo de iteraciones basadas en el conjunto de valores especificados.



```
FOR contador IN 1..10
BEGIN
PRINT 'Contador: ' + CAST((contador) AS VARCHAR);
END;
```



```
Contador: 1
Contador: 2
Contador: 3
Contador: 4
Contador: 5
Contador: 6
Contador: 7
Contador: 8
Contador: 9
Contador: 10
```

Funciones Nativas

Print

Se utiliza para imprimir expresiones.

Lower

Convierte una cadena de texto a minúsculas.

Upper

Convierte una cadena de texto a mayúsculas.

Round

Redondea un número decimal al número entero más cercano o a una cantidad específica de decimales.

Length

Devuelve la longitud (número de caracteres) de una cadena de texto.

Truncate

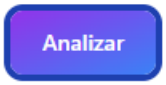
Trunca un número decimal para eliminar los decimales.

TypeOf

Devuelve el tipo de datos de una expresión o valor.

Archivo 1

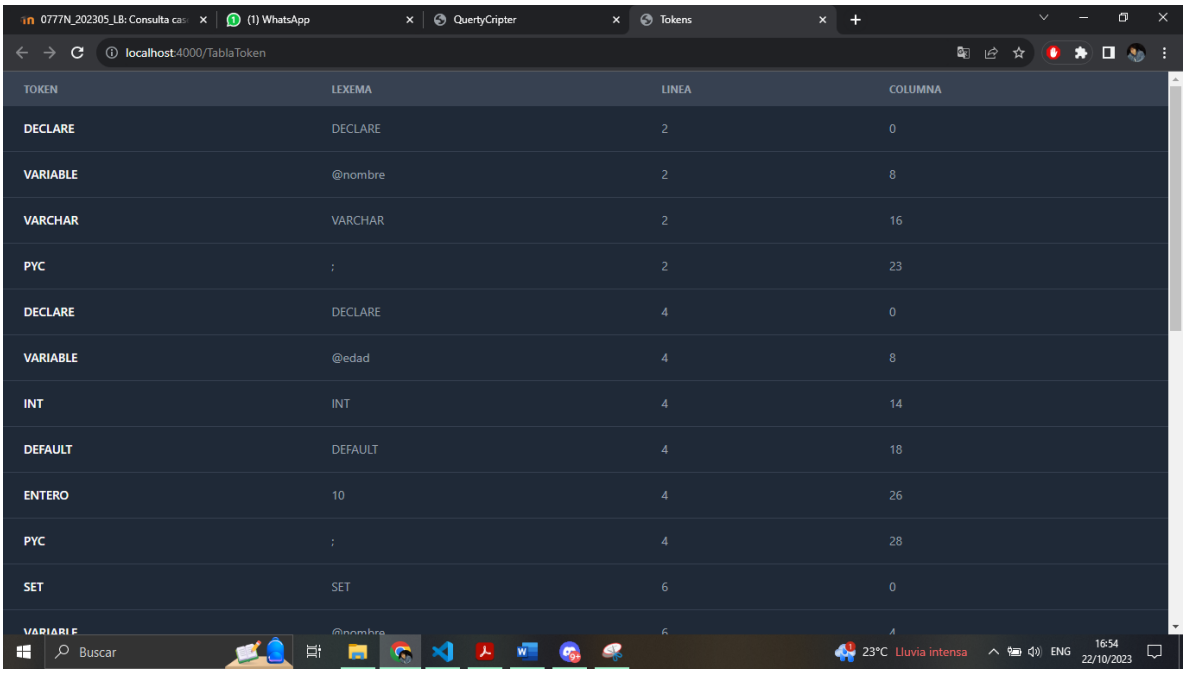
```
PRINT LOWER('HOLA MUNDO');
PRINT UPPER('hola mundo');
print ROUND(5.678, 2);
PRINT LEN('Hola mundo');
PRINT TRUNCATE(8.945, 1);
PRINT TYPEOF(123);
```




```
hola mundo
HOLA MUNDO
5.68
10
8.9
INT
```

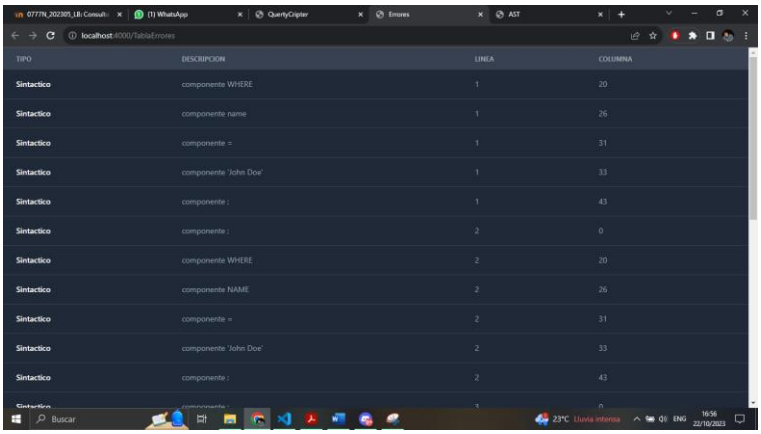
Reportes

- Reporte de Tokens: Se mostrarán todos los tokens reconocidos por el analizador léxico



TOKEN	LEXEMA	LINEA	COLUMNA
DECLARE	DECLARE	2	0
VARIABLE	@nombre	2	8
VARCHAR	VARCHAR	2	16
PYC	;	2	23
DECLARE	DECLARE	4	0
VARIABLE	@edad	4	8
INT	INT	4	14
DEFAULT	DEFAULT	4	18
ENTERO	10	4	26
PYC	;	4	28
SET	SET	6	0

- Reporte de Errores: Se mostrarán todos los errores léxicos y sintácticos encontrados.



- Reporte de Tabla de Símbolos: Se mostrarán todas las variables almacenadas en la tabla de símbolos.

ID	TIPO	TIPO	ENTORNO	LINEA	COLUMNA
@nombre	Juan Pérez	VARCHAR	Global	6	14
@edad	25	INT	Global	8	12

- Generar Árbol de Análisis Sintáctico: Se deberá generar una imagen del árbol de análisis sintáctico resultante del análisis.