UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 2
MSC. LUIS ESPINO
AUX. JUAN CARLOS MAEDA
GRUPO 18

TytusDB:Manual Usuario

ALEX RENÉ LÓPEZ ROSA 201602999
MIAMIN ELIEL BARRIOS ARRIVILLAGA 201603016
KEVIN GOLWER ENRIQUE RUIZ BARBALES 201603009
EDWARD DANILO GÓMEZ HERNÁNDEZ 201602909

Índice

Objetivos

- **General:** Ser una guía sencilla para cualquier usuario que desee hacer uso de la aplicación, indicando los pasos para usarla y explicando el resultado.

- **Específico:** Mostrar de manera amigable con imágenes ,cada una de las funcionalidades de la aplicación TytusDB ,así también dar algunos ejemplos básicos de código y pasos a seguir de algunas funciones.

Alcance

Se espera que TytusDB se pueda ejecutar en sistemas que posean un intérprete de Python 3.9.0 , Python 3.8.5 o superior.
Se pretende llegar a personas que tengan un conocimiento mínimo previo de SQL para que logre comprender algunas de las funciones que posee el programa.
El programa posiblemente tenga en un futuro múltiples actualizaciones y adiciones que mejorará la experiencia del usuario.
El proyecto tendrá como objetivo únicamente fines didácticos ,y como tal se podrá hacer uso del código fuente bajo la licencia (MIT)
El código fuente podrá servir a otros para construir , mejorar u orientarse para crear un intérprete reducido de sql postgres escrito en Python

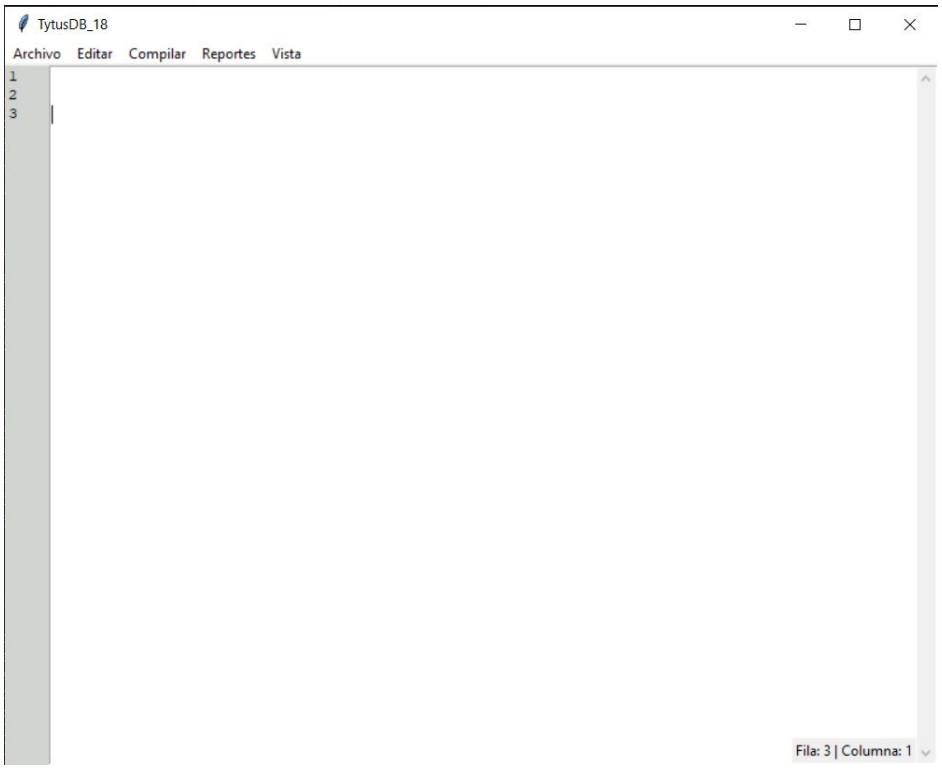
Introducción

Se presentará en el documento, una serie de explicaciones sencillas de como hacer uso de la aplicación TytusDB ,la aplicación se basa en un subconjunto de instrucciones del lenguaje SQL Postgres , este consta de múltiples comandos y funcionalidades que nos facilitan tareas como la visualización de consultas y errores, además de poder ver la tabla de símbolos generadas por el analizador, podremos acceder a reportes de errores al momento de ejecutar una consulta, este también consta de un apartado en donde se definen las optimizaciones hechas sobre el código de tres direcciones, la aplicación posee una interfaz gráfica amigable con el usuario para que este pueda familiarizarse rápidamente y así mismo facilitar el manejo y control del código introducido, se pretende que al finalizar la lectura del manual el usuario logre comprender de la manera más simple y rápida el funcionamiento de la aplicación

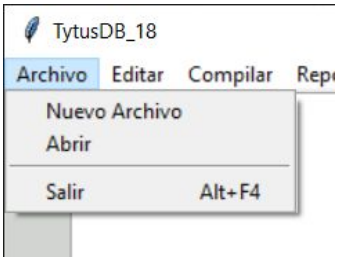
PARTES Y FLUJO DE LA APLICACIÓN

La aplicación se divide en los siguientes módulos:

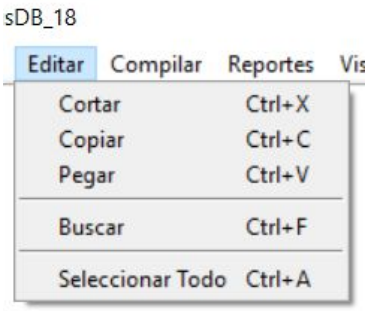
1. **Editor de texto:** El área de trabajo cuenta con un cuadro de texto en el cual se puede visualizar el número de fila y columna en la cual nos encontramos. En esta área se escriben todas las querys que se deseen



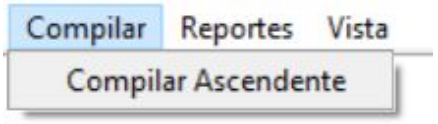
2. **Menù Archivo:** Este menú cuenta con las siguientes funcionalidades
    - **Nuevo Archivo:** Nos permite crear un archivo en blanco para poder escribir las querys que se necesiten
    - **Abrir:** Despliega un gestor de archivos para poder hacer una carga de un tipo de archivo .sql para poder ser leído su contenido y ser insertado en el área de trabajo
    - **Salir:** Nos permite salir de la aplicación



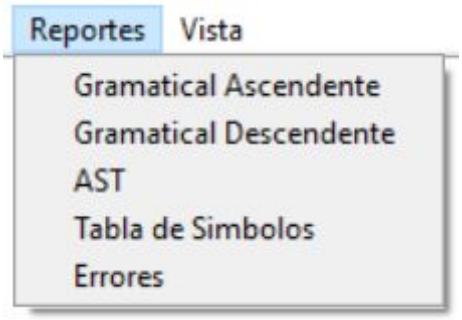
3. **Menú Editar:** Contiene funcionalidades básicas



4. **Menú compilar:** Esta opción nos permite analizar la entrada de las querys solicitadas en el área de trabajo.



5. **Menú Reportes:** Los reportes disponibles son los siguientes:



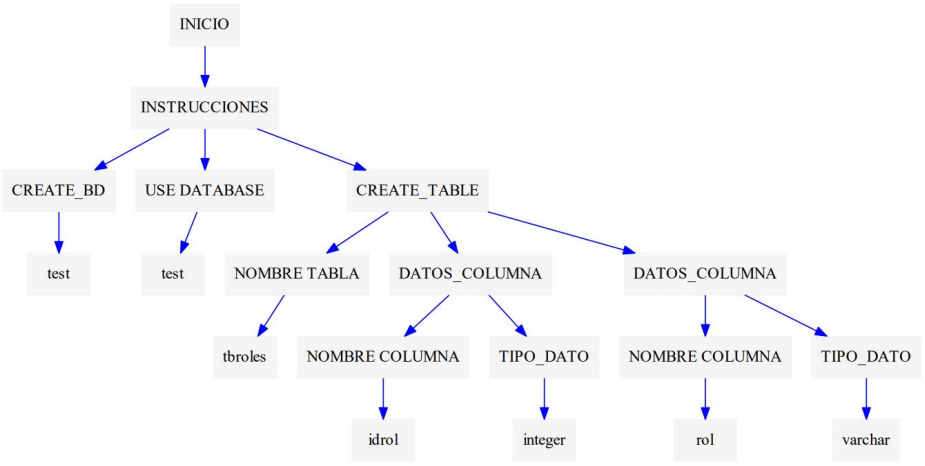
- **Reportes de errores léxico, sintácticos y semánticos.** Muestra el tipo, la descripción y el número de línea de los diferentes errores que se encontraron al momento de la ejecución

- **Reporte de tabla de símbolos.** Muestra las variables, funciones y procedimientos con mínimo los siguientes datos: identificador, tipo, dimensión, declarada en, y referencias.

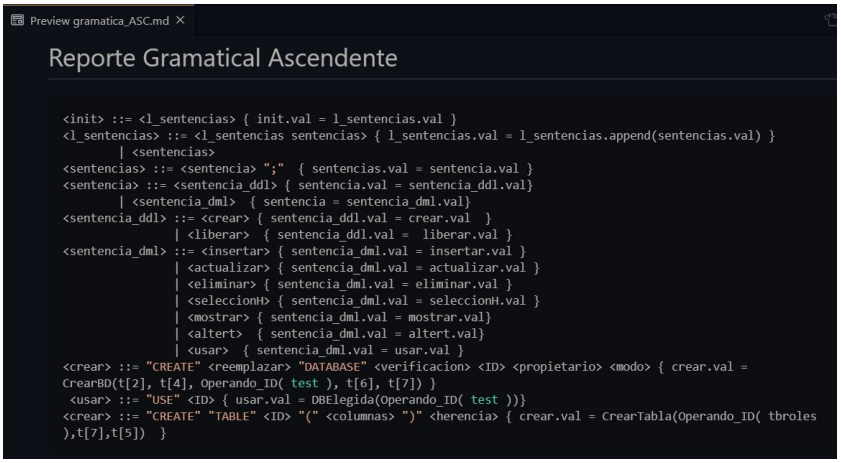
REPORTE TABLA DE SIMBOLOS

instruccion	identificador	tipo	referencia	dimension
INSERT				
	val1	integer	llaves1	1
INSERT				
	val1	integer	llaves1	1
INSERT				
	val2	varchar	llaves2	1
	val3	integer	llaves2	1
INSERT				
	val2	varchar	llaves2	1
	val3	integer	llaves2	1
INSERT				
	val3	integer	mitabla	1
	val2	varchar	mitabla	1
	val1	integer	mitabla	1
INSERT				
	val1	integer	mitabla	1
	val2	varchar	mitabla	1
	val3	integer	mitabla	1

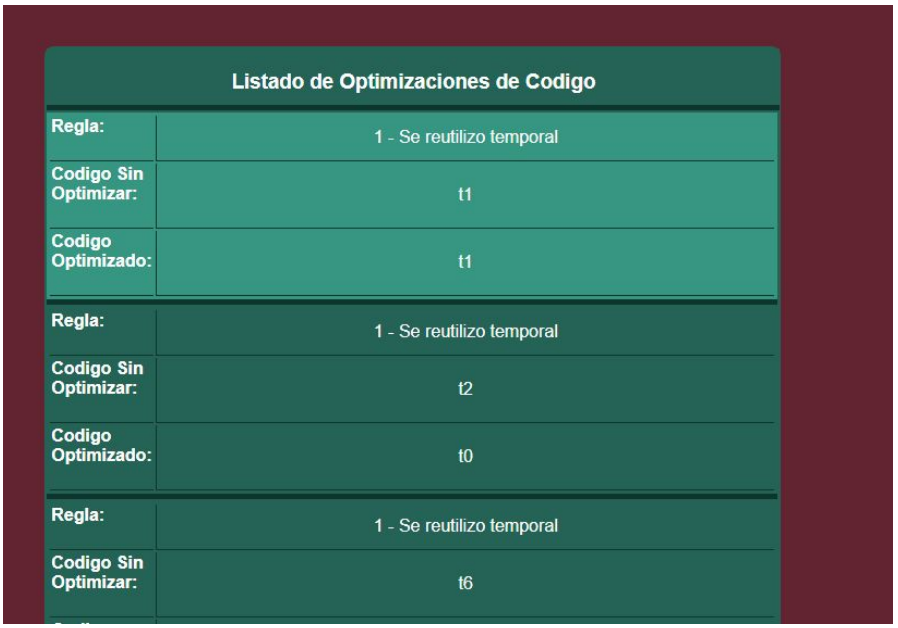
- **Reporte de AST.** Muestra el árbol de sintaxis abstracta utilizando Graphviz en una nueva ventana.



- **Reporte gramatical.** Abre archivo con Markdown que muestra las dos gramáticas con sintaxis BNF. En otro documento se muestra la definición dirigida por la sintaxis con la gramática ascendente, indicando que expresiones se utilizaron, precedencia, símbolos terminales y no terminales, y las reglas semánticas.



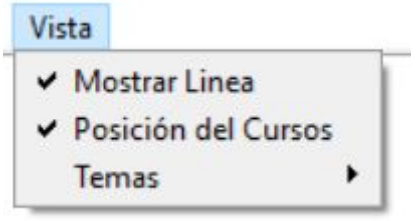
- **Reporte de Optimización.** Abre un archivo en formato html, este contiene la lista de optimizaciones hecha sobre el código de tres direcciones ,se subdivide cada elemento en Regla Aplicada, Codigo Sin optimizar y Código Optimizado



Listado de Optimizaciones de Codigo

Regla:	1 - Se reutilizo temporal
Codigo Sin Optimizar:	t1
Codigo Optimizado:	t1
Regla:	1 - Se reutilizo temporal
Codigo Sin Optimizar:	t2
Codigo Optimizado:	t0
Regla:	1 - Se reutilizo temporal
Codigo Sin Optimizar:	t6

6. **Menú Vista:** El menú vista nos proporciona las siguientes funcionalidades:
   - **Mostrar línea**
   - **Posición del cursor**
   - **Temas:** No permite escoger entre dos temas: Light o dark



Aplicación en funcionamiento

- **Create**

```
CREATE DATABASE IF NOT EXISTS test
    OWNER = 'root'
    MODE = 1;

USE test;

CREATE TABLE tbusuario (
    idusuario integer NOT NULL primary key,
        nombre varchar(50),
        apellido varchar(50),
        usuario varchar(15)  UNIQUE NOT NULL,
        password varchar(15) NOT NULL,
        fechacreacion date
);

CREATE TYPE tipodato AS ENUM('Entero','Cadena','Boolean');
```

Consola

```
> Creando base de datos: test
        Todo OK
> Seleccionando base de datos: test
        Base de datos seleccionada
> Creando Tabla:tbusuario
        Todo OK
> Creacion de Type: tipodato
        Type registrado con exito
        con valores: ['Entero', 'Cadena', 'Boolean']
```

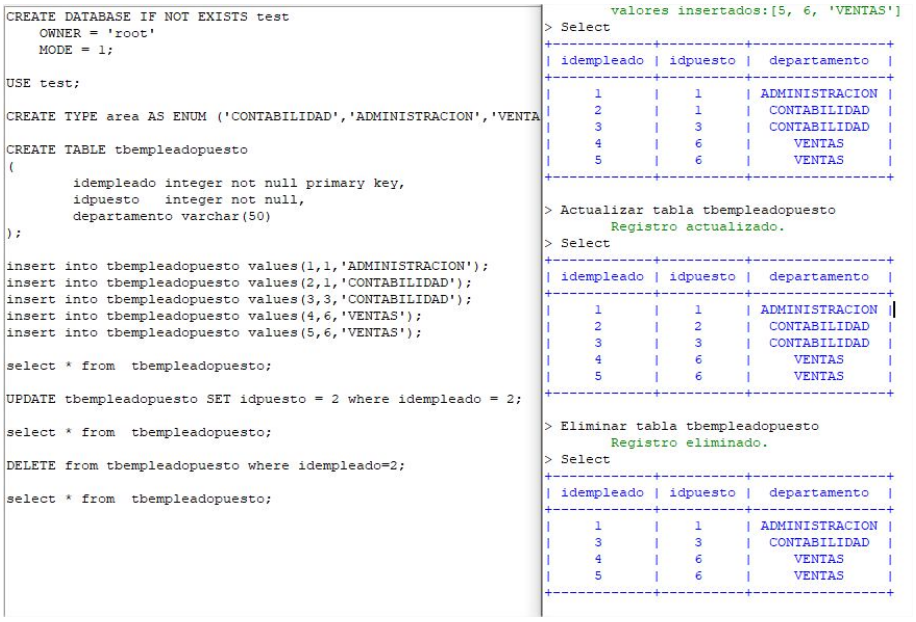
- **Insert**

```
    idusuario integer NOT NULL primary key,
        nombre varchar(50),
        apellido varchar(50),
        usuario varchar(15)  UNIQUE NOT NULL,
        password varchar(15) NOT NULL,
        fechacreacion date
);

CREATE TABLE tbroles (
    idrol integer NOT NULL primary key,
        rol varchar(15)
);
DROP TABLE tbroles;

CREATE TABLE tbrol (
    idrol integer NOT NULL primary key,
        rol varchar(15)
);



CREATE TABLE tbrolxusuario (
    idrol integer NOT NULL ,
        idusuario integer NOT NULL
);

insert into tbrol values (1,'Administrador
insert into tbrol values (2,'Admin');
insert into tbrol values (3,'Ventas');
```

Consola

```
> Creando base de datos: test
        Todo OK
> Seleccionando base de datos: test
        Base de datos seleccionada
> Creando Tabla:tbusuario
        Todo OK
> Creando Tabla:tbroles
        Todo OK
> Eliminar Tabla:tbroles
        Tabla eliminada
> Creando Tabla:tbrol
        Todo OK
> Creando Tabla:tbrolxusuario
        Todo OK
> Insertado en Tabla:tbrol
        valores insertados:[1, 'Administrador']
> Insertado en Tabla:tbrol
        valores insertados:[2, 'Admin']
> Insertado en Tabla:tbrol
        valores insertados:[3, 'Ventas']
```

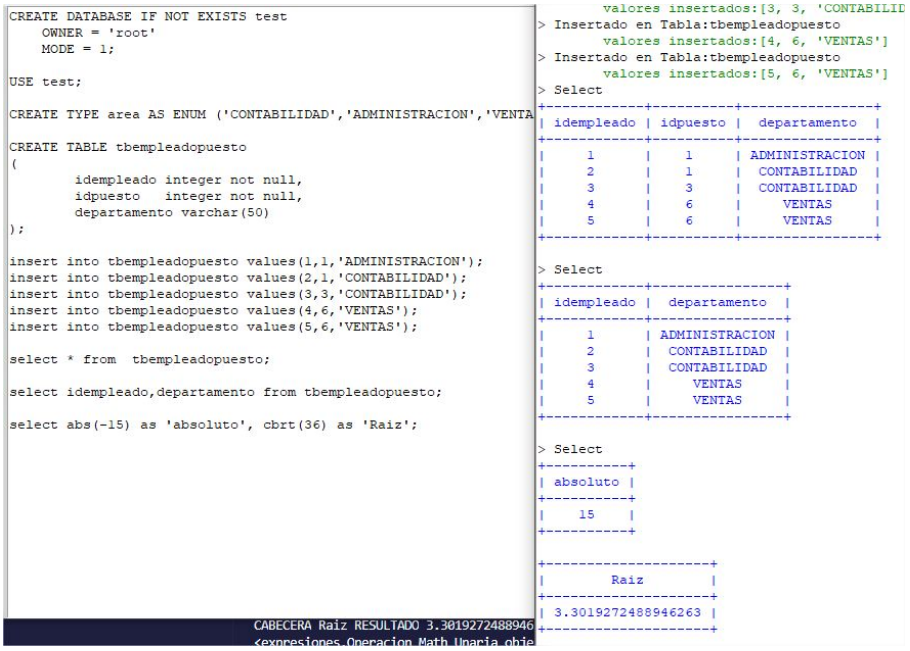
- **Update y Delete**



```
CREATE DATABASE IF NOT EXISTS test
    OWNER = 'root'
    MODE = 1;

USE test;

CREATE TYPE area AS ENUM ('CONTABILIDAD','ADMINISTRACION','VENTA

CREATE TABLE tbempleadopuesto
(
        idempleado integer not null primary key,
        idpuesto   integer not null,
        departamento varchar(50)
);

insert into tbempleadopuesto values(1,1,'ADMINISTRACION');
insert into tbempleadopuesto values(2,1,'CONTABILIDAD');
insert into tbempleadopuesto values(3,3,'CONTABILIDAD');
insert into tbempleadopuesto values(4,6,'VENTAS');
insert into tbempleadopuesto values(5,6,'VENTAS');

select * from  tbempleadopuesto;

UPDATE tbempleadopuesto SET idpuesto = 2 where idempleado = 2;

select * from  tbempleadopuesto;

DELETE from tbempleadopuesto where idempleado=2;

select * from  tbempleadopuesto;
```

```
        valores insertados:[5, 6, 'VENTAS']
> Select
+------------+----------+----------------+
| idempleado | idpuesto |  departamento  |
+------------+----------+----------------+
|     1      |    1     | ADMINISTRACION |
|     2      |    1     |  CONTABILIDAD  |
|     3      |    3     |  CONTABILIDAD  |
|     4      |    6     |     VENTAS     |
|     5      |    6     |     VENTAS     |
+------------+----------+----------------+

> Actualizar tabla tbempleadopuesto
        Registro actualizado.
> Select
+------------+----------+----------------+
| idempleado | idpuesto |  departamento  |
+------------+----------+----------------+
|     1      |    1     | ADMINISTRACION |
|     2      |    2     |  CONTABILIDAD  |
|     3      |    3     |  CONTABILIDAD  |
|     4      |    6     |     VENTAS     |
|     5      |    6     |     VENTAS     |
+------------+----------+----------------+

> Eliminar tabla tbempleadopuesto
        Registro eliminado.
> Select
+------------+----------+----------------+
| idempleado | idpuesto |  departamento  |
+------------+----------+----------------+
|     1      |    1     | ADMINISTRACION |
|     3      |    3     |  CONTABILIDAD  |
|     4      |    6     |     VENTAS     |
|     5      |    6     |     VENTAS     |
+------------+----------+----------------+
```

- **Select**



```
CREATE DATABASE IF NOT EXISTS test
    OWNER = 'root'
    MODE = 1;

USE test;

CREATE TYPE area AS ENUM ('CONTABILIDAD','ADMINISTRACION','VENTA

CREATE TABLE tbempleadopuesto
(
        idempleado integer not null,
        idpuesto   integer not null,
        departamento varchar(50)
);

insert into tbempleadopuesto values(1,1,'ADMINISTRACION');
insert into tbempleadopuesto values(2,1,'CONTABILIDAD');
insert into tbempleadopuesto values(3,3,'CONTABILIDAD');
insert into tbempleadopuesto values(4,6,'VENTAS');
insert into tbempleadopuesto values(5,6,'VENTAS');

select * from  tbempleadopuesto;

select idempleado,departamento from tbempleadopuesto;

select abs(-15) as 'absoluto', cbrt(36) as 'Raiz';
```

```
CABECERA Raiz RESULTADO 3.3019272488946
<expresiones.Operacion.Math.Unaria.obje
```

```
        valores insertados:[3, 3, 'CONTABILID
> Insertado en Tabla:tbempleadopuesto
        valores insertados:[4, 6, 'VENTAS']
> Insertado en Tabla:tbempleadopuesto
        valores insertados:[5, 6, 'VENTAS']
> Select
+------------+----------+----------------+
| idempleado | idpuesto |  departamento  |
+------------+----------+----------------+
|     1      |    1     | ADMINISTRACION |
|     2      |    1     |  CONTABILIDAD  |
|     3      |    3     |  CONTABILIDAD  |
|     4      |    6     |     VENTAS     |
|     5      |    6     |     VENTAS     |
+------------+----------+----------------+

> Select
+------------+----------------+
| idempleado |  departamento  |
+------------+----------------+
|     1      | ADMINISTRACION |
|     2      |  CONTABILIDAD  |
|     3      |  CONTABILIDAD  |
|     4      |     VENTAS     |
|     5      |     VENTAS     |
+------------+----------------+

> Select
+----------+
| absoluto |
+----------+
|    15    |
+----------+

+--------------------+
|        Raiz        |
+--------------------+
| 3.3019272488946263 |
+--------------------+
```

- **Index**

Archivo Editar Compilar Reportes Vista

```sql
CREATE FUNCTION myFuncion(texto text) RETURNS text AS $$
BEGIN
        RETURN texto;
END;
$$ LANGUAGE plpgsql;

select @myFuncion('INICIO CALIFICACION FASE 2');

CREATE TABLE tbProducto (idproducto integer not null primary key,
                                                  producto varchar
                                                  fechacreacion da
                                                  estado integer);

CREATE UNIQUE INDEX idx_producto ON tbProducto (idproducto);

CREATE TABLE tbCalificacion (idcalifica integer not null primary
                                                          item var
                                                          punteo i

CREATE UNIQUE INDEX idx_califica ON tbCalificacion (idcalifica);

INSERT INTO tbProducto values(1,'Laptop Lenovo',now(),1);
INSERT INTO tbProducto values(2,'Bateria para Laptop Lenovo T420'
INSERT INTO tbProducto values(3,'Teclado Inalambrico',now(),1);
INSERT INTO tbProducto values(4,'Mouse Inalambrico',now(),1);
INSERT INTO tbProducto values(5,'WIFI USB',now(),1);

CREATE FUNCTION ValidaRegistros(tabla varchar(50),cantidad intege
DECLARE resultado INTEGER;
                retorna   INTEGER;
BEGIN
        if tabla = 'tbProducto' then
            resultado := (SELECT COUNT(*) FROM tbProducto);
        if cantidad = resultado then
                        retorna := 1;
```

REPORTE TABLA DE SIMBOLOS

instruccion	identificador	tipo	unique	referencia	dimension
INDEX					
	idx_producto	Normal	UNIQUE	tbproducto	1
INDEX					
	idx_califica	Normal	UNIQUE	tbcalificacion	1
INSERT					
	idproducto	integer		tbproducto	1
	producto	varchar		tbproducto	1
	fechacreacion	date		tbproducto	1
	estado	integer		tbproducto	1
INSERT					
	idproducto	integer		tbproducto	1
	producto	varchar		tbproducto	1
	fechacreacion	date		tbproducto	1
	estado	integer		tbproducto	1
INSERT					
	idproducto	integer		tbproducto	1
	producto	varchar		tbproducto	1
	fechacreacion	date		tbproducto	1
	estado	integer		tbproducto	1
INSERT					
	idproducto	integer		tbproducto	1
	producto	varchar		tbproducto	1
	fechacreacion	date		tbproducto	1
	estado	integer		tbproducto	1
INSERT					
	idproducto	integer		tbproducto	1
	producto	varchar		tbproducto	1
	fechacreacion	date		tbproducto	1
	estado	integer		tbproducto	1
INSERT					
	idcalifica	integer		tbcalificacion	1

- **Function**

```sql
INSERT INTO tbProducto values(1,'Laptop Lenovo',now(),1);
INSERT INTO tbProducto values(2,'Bateria para Laptop Lenovo T420',now(),1);
INSERT INTO tbProducto values(3,'Teclado Inalambrico',now(),1);
INSERT INTO tbProducto values(4,'Mouse Inalambrico',now(),1);
INSERT INTO tbProducto values(5,'WIFI USB',now(),1);

CREATE FUNCTION ValidaRegistros(tabla varchar(50),cantidad integer) RETURNS integer AS $$
DECLARE resultado INTEGER;
                retorna   INTEGER;
BEGIN
        if tabla = 'tbProducto' then
            resultado := (SELECT COUNT(*) FROM tbProducto);
        if cantidad = resultado then
                        retorna := 1;
                else
                        retorna := 0;
                end if;
        end if;
RETURN retorna;
END;
$$ LANGUAGE plpgsql;
```

- **Procedure**

```sql
CREATE INDEX abc ON tbbodega ((lower(bodega)));

create procedure sp_validainsert()
language plpgsql
as $$
begin
        insert into tbbodega values(1,'BODEGA CENTRAL',1);
        insert into tbbodega (idbodega,bodega) values(2,'BODEGA ZONA 12');
        insert into tbbodega (idbodega,bodega,estado) values(3,'BODEGA ZONA 11',1);
        insert into tbbodega (idbodega,bodega,estado) values(4,'BODEGA ZONA 1',1);
        insert into tbbodega (idbodega,bodega,estado) values(5,'BODEGA ZONA 10',1);
end; $$;

EXECUTE sp_validainsert();

insert into tbCalificacion values(4,'Valida Store Procedure',@ValidaRegistros('tbbodega',5));
```

- **Código de tres direcciones**

```
#importar modulos
import CD3  as CD3  #modulo codigo 3 direcciones
from goto import with_goto  #modulo goto

@with_goto  # Decorador necesario
def main():
#Codigo Resultante
    #Create DataBase
    t0='dbfase2'
    t1=CD3.EReplace()
    if(t1):
        goto .dropDB11
        label.dropDB11
        CD3.EDropDatabase()
    CD3.ECreateDatabase()

    #Use Database
    t0='dbfase2'
    CD3.EUseDatabase()

    #Crear Funcion
    t0='myfuncion'
    t1=['texto']
    t2=False #Reemplazar funcion
    t3=CD3.ECreateFuncion()
    if(t3==False):
```

Conclusión

Con la explicación descrita en este documento ,cualquier persona con un conocimiento básico en SQL lograra comprender de una manera sencilla y rápida, el manejo de la aplicación TytusDB , el programa brindará al usuario una forma fácil e intuitiva de usarlo ,además de que si se deseara el ampliar el conocimiento del lenguaje postgres , se podrá usar la documentación de la misma para mejorar la experiencia con el programa ,este tambien podra ser vinculado para el manejo de la información , con una sección en donde la información se almacenará en una estructura de datos específica y también ser vinculado por parte de un gestor en donde se podrán hacer llamadas a esta y se retorna la información que el comando requiera, todo esto para simular un sistema de base de datos