Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas
Organización de Lenguajes y Compiladores 2
Vacaciones de diciembre 2018
Catedrático: Ing. Kevin Lajpop
Tutor académico: Javier Navarro

FISQL

Primer proyecto de laboratorio

Índice

1 Objetivos 3
1.1 Objetivo General 3
1.2 Objetivos Específicos 3
2 Descripción General 3
3 Descripción de servidores 5
3.1 Servidor Web 5
Características del IDE 5
3.2 Servidor de Base de Datos 9
Interfaz del servidor 10
4 Descripción de lenguajes 10
4.1 Lenguaje USQL 10
4.1.1 Componentes del lenguaje USQL 11
4.1.2 Sentencias USQL 15
4.2 Lenguaje de PlyCS 29
4.2.1 Descripción de paquetes 30
4.2.2 Inicio de sesión (login) 30
4.2.3 Paquete de finalización 30
4.2.4 Paquete de error 31
4.2.5 Paquete de instrucción USQL 31
4.2.6 Paquete de instrucción Reporte 32
4.3 Lenguaje XML 32
5 Manejo y almacenamiento de información 32
5.1 Sistema de Archivos 32
5.1.1 Archivo Maestro 33

5.1.2 Archivo DB 33
5.1.3 Archivo Tabla 34
5.1.4 Archivo Método 34
5.1.5 Archivo Objeto 35
5.1.6 Estructura general del sistema de archivos: 35
5.1.7 Intérprete XML 36
6 Reportes 37
7 Manejo de errores 39
7.1 Errores del lenguaje USQL 40
7.2 Errores del lenguaje XML 40
7.3 Errores PlyCS 40
8 Entregables y restricciones 41
8.1 Restricciones 41
8.2 Requisitos mínimos 41
8.3 Entregables 42
8.4 Fecha de Entrega: 42

1 Objetivos

1.1 Objetivo General

Aplicar los conocimientos sobre compiladores adquiridos en el curso para la creación de una aplicación.

1.2 Objetivos Específicos

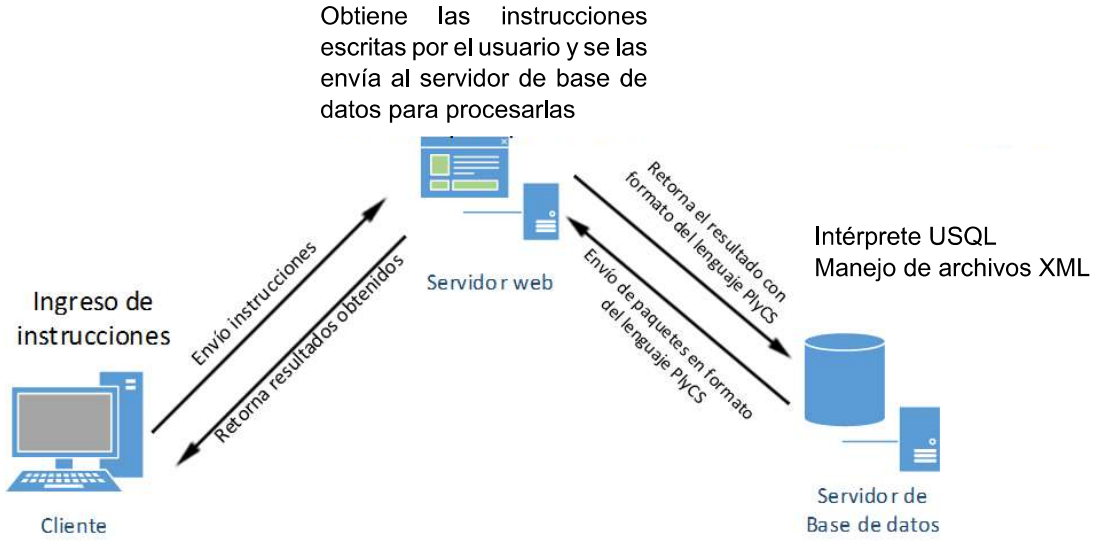
- Reafirmar los conceptos de compiladores de una forma práctica.
- Utilizar herramientas para la generación de analizadores léxicos y sintácticos.
- Implementar lenguajes intermedios para crear una interfaz de comunicación entre sistemas.
- Comunicar aplicaciones desarrolladas en distintos lenguajes de programación por medio de sockets.
- Aplicar conceptos de compiladores y utilizarlos dentro de una aplicación de software.

2 Descripción General

Se solicita que el estudiante desarrolle un sistema administrador de base de datos, capaz de manejar las instrucciones básicas de un DBMS relacional convencional junto con instrucciones adicionales de un paradigma orientado a objetos, dicho sistema recibirá el nombre de FISQL.

El sistema consistirá en dos servidores, uno web y uno con la lógica de la base de datos. Los usuarios podrán acceder al sistema desde sus máquinas personales, y se conectarán por red al servidor web que tendrá el frontend del sistema.

En el siguiente diagrama se muestra la estructura del sistema.



En el servidor web se encontrará el IDE con el que el usuario interactúa directamente, este tendrá un conjunto de herramientas que ayudarán al usuario a poder adaptarse fácilmente al sistema FISQL.

Dentro del IDE se deberá de brindar una ayuda gráfica, una forma de realizar acciones sobre la base de datos sin necesidad de escribir las sentencias en lenguaje USQL, esta ayuda grafica será llamada métodos gráficos.

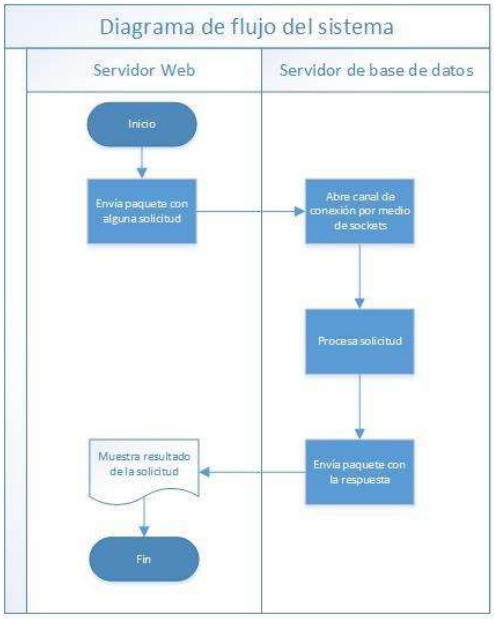
En la parte del servidor de base de datos se tendrá el intérprete del lenguaje con el que se realizarán las acciones en la base de datos, este lenguaje será conocido con el nombre de USQL. Este servidor será el encargado de almacenar y manipular la información según la entrada del código USQL.

El sistema incluye un módulo de reportes editables, lo que quiere decir que el usuario tendrá que diseñar el reporte, esto por medio del lenguaje html y permitirá incluir información de las bases de datos. Dichos reportes deben de ser visibles desde cualquier navegador. Este módulo puede ser utilizado desde el servidor web.

Para almacenar y manipular la información de las bases de datos se deberá manejar un sistema de archivos con formato XML, que contendrá la estructura de las bases de datos creadas, así como las tablas, objetos o procedimientos que contenga. Para la manipulación de la información el estudiante deberá de implementar un intérprete XML para leer la información guardada en los archivos.

Se requiere que, para la comunicación entre los servidores, se haga mediante un lenguaje interno llamado PlyCS con el fin que ambos sistemas se comuniquen mediante un lenguaje estandarizado.

El proceso general de comunicación entre los servidores y procesamiento de solicitudes se describe en el siguiente diagrama.



3 Descripción de servidores

En esta sección se detallan las tareas y funciones de cada uno de los elementos (servidores) que conforman la solución, se indica lo que deben de hacer y cómo lo deben de hacer. Además, se indican las herramientas a utilizar en la implementación de cada una.

3.1 Servidor Web

Este servidor proporcionará un IDE que permitirá el ingreso de instrucciones USQL por medio de un editor de texto.

Un IDE (Entorno de desarrollo integrado), el cual consiste de un editor de código fuente, métodos gráficos, auto-completado inteligente de código (IntelliSense).

El servidor web será el encargado de enviar las instrucciones al servidor de Base de datos y de recibir las respuestas que este retorne. El envío y recepción de información se hará por medio del lenguaje de comunicación PlyCS.

El servidor web se comunicará por medio del lenguaje de comunicación PlyCS con el servidor de base de datos, esto a través de paquetes, los cuales contendrán la información de la solicitud.

El servidor web será implementado en el lenguaje de programación Visual Basic.

Características del IDE

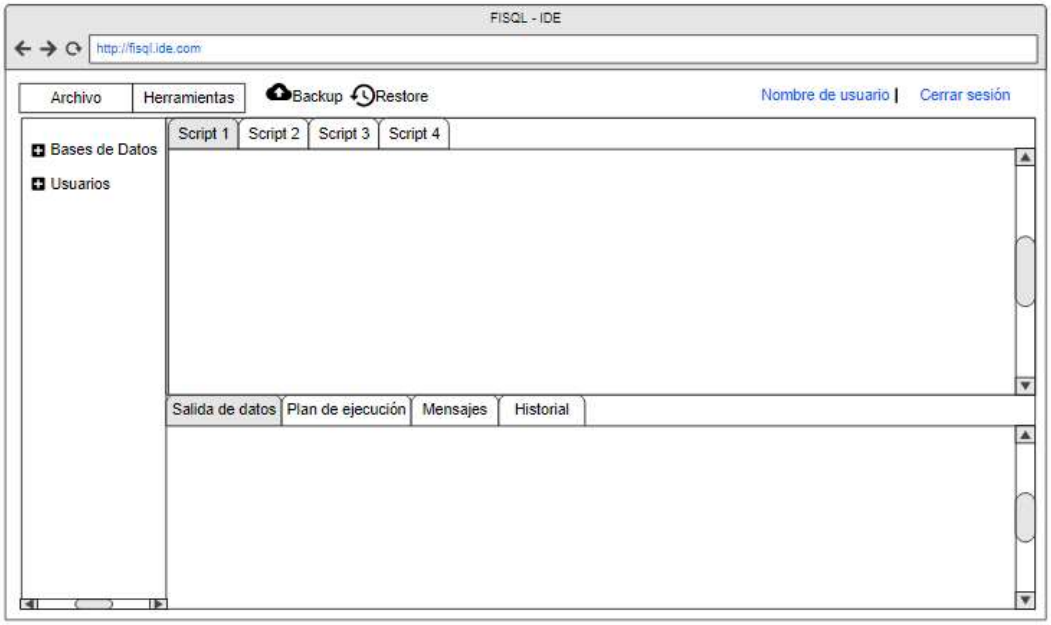
A continuación, se detallan todos los elementos con los que contará el IDE, que tendrá la función de servidor web y que servirá para interactuar con el servidor de base de datos.

Login

Para ingresar al servidor web, se contará con un sistema de autenticación en el que se ingresa el usuario y la contraseña. Al inicio el sistema contará con un usuario administrador por defecto, siendo sus credenciales *usuario: admin* y *contraseña: admin.* A partir de este usuario, se crearán más usuarios a los cuales se podrá asignar distintos permisos para diferentes elementos de una base de datos. Es importante mencionar que, si algún usuario no se ha autenticado, no podrá acceder a ninguna parte del IDE. Los usuarios que se manejarán en el IDE para el inicio de sesión son los mismos que estarán almacenados en la base de datos.

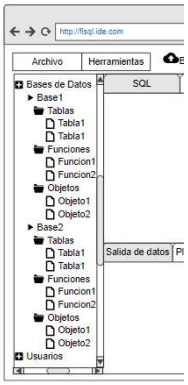
Vista general de IDE

El servidor web proporciona un IDE para facilitar al usuario el manejo del DBMS y sus componentes.

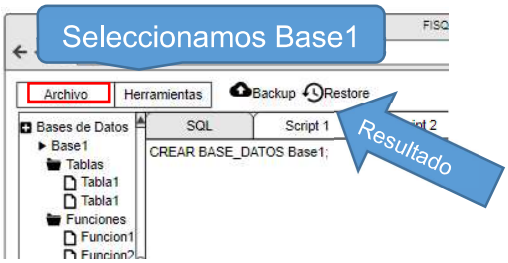


Se deberá tener un menú en forma de árbol en el que se debe de mostrar cada base de datos creada y los elementos que la conforman. A pesar de que a todos los usuarios se les mostrarán todas las bases de datos, cada usuario solo podrá acceder a las bases de datos permitidas.

Para cada base de datos se deben de tener los submenús que desplieguen las tablas, funciones, procedimientos, objetos y todo elemento que el estudiante considere necesario separar para un mejor orden en la presentación, con los que cuenta la base de datos.



Se deberá tener una pestaña por defecto “SQL” dentro del editor, en el que, al momento de seleccionar cualquier elemento del menú y submenú de una base de datos, se mostrarán las instrucciones en lenguaje USQL que se utilizaron para su creación. Esto para facilitar la creación de elementos similares en la base de datos, en el caso de que se necesite.

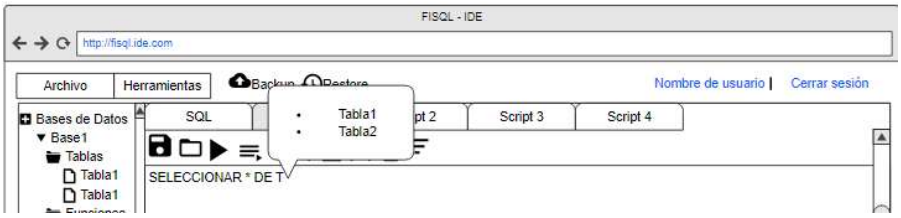


Editor de texto

El editor de código fuente, es una herramienta de construcción automática, que posee auto-completado inteligente de código USQL.

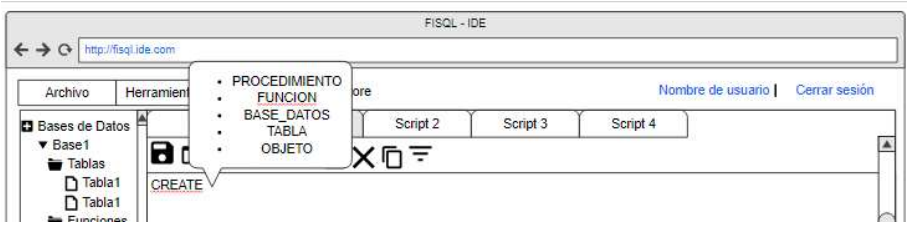
- En él se podrá ingresar todas las instrucciones soportadas por el lenguaje USQL.
- Se deberá permitir abrir o editar más de un archivo a la vez.
- Deberá implementarse de tal forma que ayude al usuario a predecir lo que quiere escribir.

Ejemplo 1 (lista de instrucciones): En la siguiente imagen se está escribiendo la instrucción que selecciona toda la información de una tabla de la base de datos "Base1", ya que en esta base de datos posee dos tablas "Tabla1" y "Tabla2", el editor muestra la lista de tablas que inician con la letra "T" para que el usuario pueda seleccionar la que necesite.

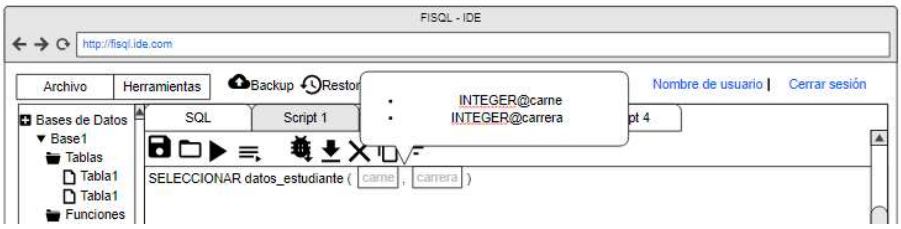


NOTA: Este ejemplo aplica para cualquier instrucción, es decir, se debe de sugerir al usuario siempre una lista de instrucciones USQL, funciones, objetos o tablas existentes, que coincidan con lo que se está escribiendo en ese momento.

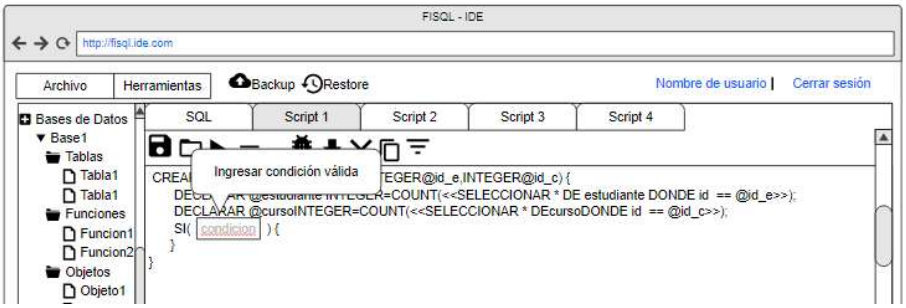
Ejemplo 2 (sugerencia de siguiente instrucción): Al finalizar la escritura de una instrucción, se debe sugerir cuál podría ser la siguiente instrucción a ingresar, de acuerdo a la última instrucción ingresada, tal y como lo muestra la siguiente imagen.



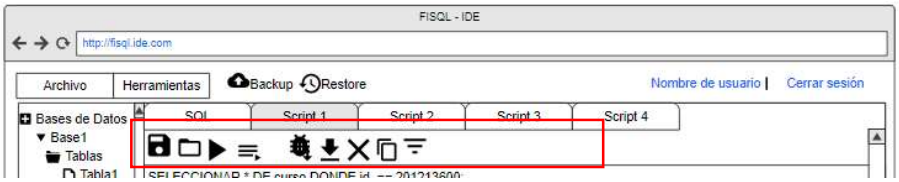
Ejemplo 3 (completar parámetros caso 1): En el caso de que se quiera llamar a un procedimiento, función definida por el usuario o una función propia del lenguaje. Se debe de autocompletar el número de parámetros requeridos por dicha función e indicando el tipo de dato de cada parámetro. Para este caso se combina la función del ejemplo 1 de esta sección para completar el nombre de la función o procedimiento y finalizar el autocompletado, se construye el "esqueleto de la función" y parámetros asociados. Tal como se muestra en la siguiente figura.



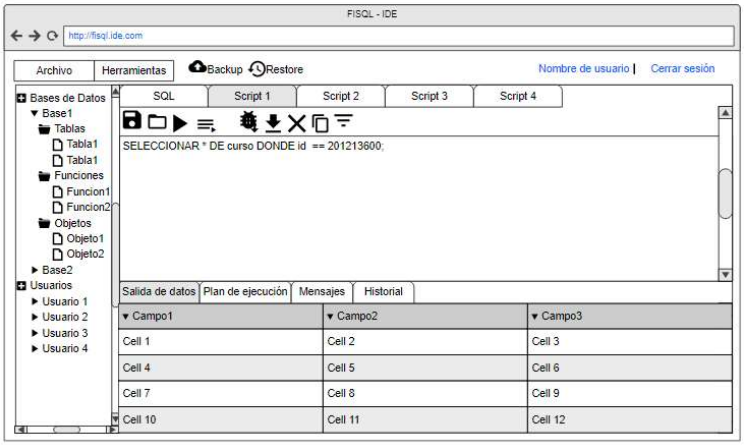
Ejemplo 4 (completar parámetros caso 2): En el caso de que se quiera escribir una sentencia de control, los parámetros para las expresiones se deben manejar tal como en el ejemplo anterior.



Cada pestaña del editor excepto “SQL”, deberá contener una serie de opciones extra que permitan ejecutar las instrucciones, las opciones mínimas son ejecución de script completo el cual ejecuta todo el contenido del archivo abierto, ejecución por selección, el cual ejecuta las instrucciones seleccionadas, guardar, abrir, copiar, exportar resultado, el cual exporta el resultado de una tabla seleccionada a un formato HTML, limpiar el área de texto.

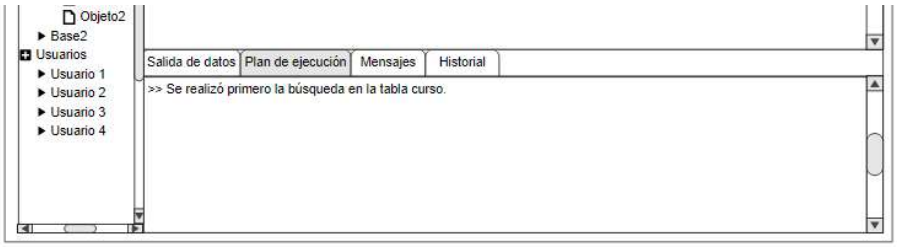


Salida de datos, plan de ejecución, mensajes e historial.

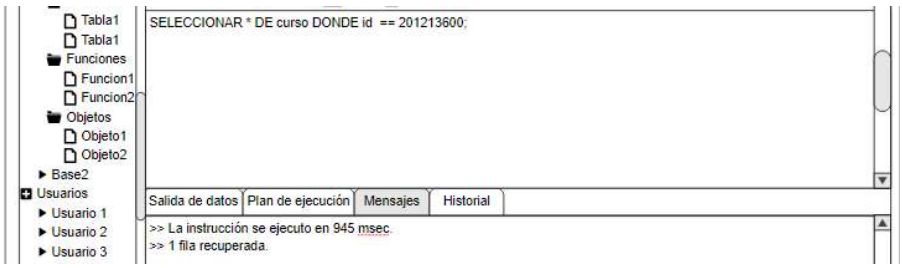


Salida de datos: Corresponde al área con forma de tabla, en donde se muestran los datos recuperados, en el caso de no recuperarse ningún dato solamente se mostrará el encabezado de la tabla con el nombre de los campos que se esperaban recuperar.

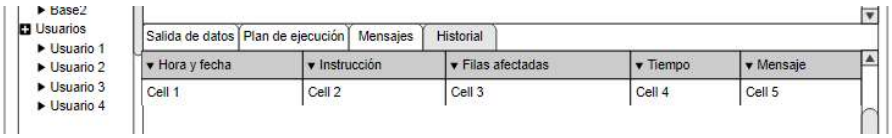
Plan de ejecución: Explica el orden del proceso que se realizó según la operación ejecutada, por ejemplo, en el caso de la operación en la figura anterior, el resultado sería el que se muestra en la siguiente imagen, si en el proceso intervino más de un elemento de la BD, se debe explicar en esta área.



Mensajes: En esta área se debe de reportar el resultado de la operación, es decir, si fue exitosa se debe de indicar lo que se realizó y si existió algún error, se debe de indicar de igual manera.



Historial: Es una tabla que guarda cada instrucción ejecutada durante la sesión del usuario, debe de mostrar la hora y fecha de ejecución, la instrucción ejecutada, filas afectadas (si fuera el caso), el tiempo de ejecución y un mensaje (de éxito o fracaso).



Errores.

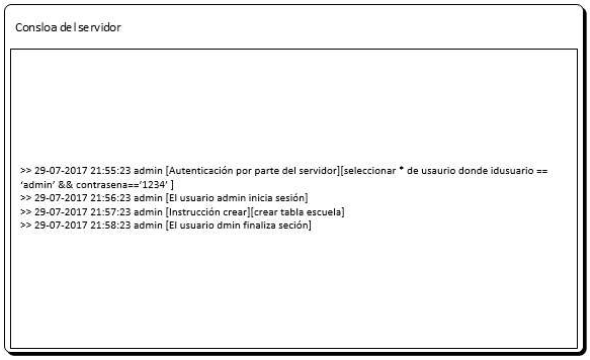
En el caso de que ocurra algún tipo de error (léxico, sintáctico, semántico), se deberá reportar la posición exacta, subrayando dicha instrucción o palabra que sea la causante.

3.2 Servidor de Base de Datos

Este servidor contendrá toda la lógica del manejo de instrucciones USQL, y el almacenamiento de la información para el uso que Servidor Web requiera. De la misma forma para el envío y recepción de información se hará por medio del lenguaje de comunicación PlyCS. El servidor de Bases de datos incluirá dos intérpretes para el manejo de la información, el intérprete USQL para el manejo de las bases de datos y un intérprete de XML que se encarga de manejar el almacenamiento en disco como tal.Este servidor estará siempre habilitado para que el servidor web se conecte a través de un socket.

Interfaz del servidor

El servidor de base de datos estará escrito en el lenguaje de programación C# y debe de tener disponible una consola de salida tipo log que notifique las transacciones que se vayan realizando en tiempo real.



Se mostrará como mínimo la fecha y hora, usuario, instrucción realizada, descripción breve de la acción que realizado en el caso de que fuera otro tipo de instrucción. Queda a criterio del estudiante agregar más información que crea necesaria.

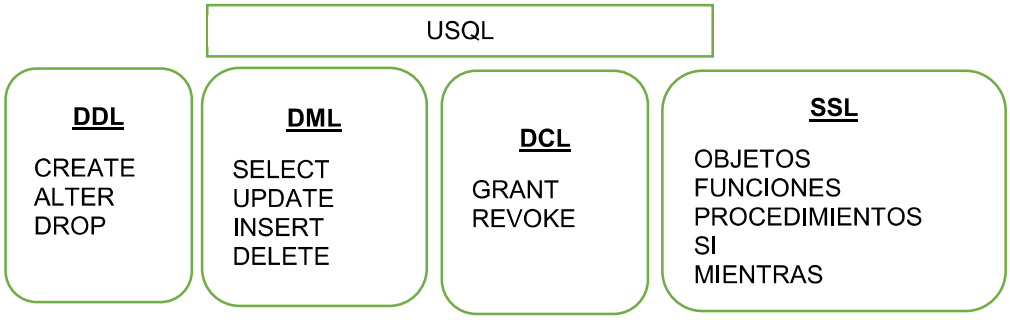
4 Descripción de lenguajes

Se contará con tres lenguajes para la realización de las distintas tareas en los servidores. El primer lenguaje será para las instrucciones USQL, el cual será procesado por el servidor de base de datos. Se manejará el lenguaje estándar XML para el almacenamiento de la información en el servidor de base de datos. Para la comunicación entre el servidor de base de datos y el servidor web, se contará con el lenguaje PlyCS. A continuación, se detalla cada uno de ellos.

4.1 Lenguaje USQL

El lenguaje USQL es un lenguaje de bases de datos relacionales que adopta varias funcionalidades de la programación orientada a objetos. Este lenguaje se divide en 4 partes: DML, DDL, DCL y SSL.

La estructura del lenguaje se ve representado en el siguiente diagrama.



En donde:

DDL (Data Definition Language por sus siglas en inglés): conjunto de instrucciones de definición de datos. Este set de instrucciones se encarga de definir la estructura de las bases de datos, tablas, procedimientos y objetos.

DML (Data Manipulation Language por sus siglas en inglés): conjunto de instrucciones que manipulan los datos. Este set de instrucciones se encarga de manipular la información siguiendo la estructura definida por las instrucciones DDL.

DCL (Data Control Language por sus siglas en inglés): conjunto de instrucciones que administran los permisos de los usuarios a las bases de datos.

SSL (System Statement Language por sus siglas en inglés): conjunto de sentencias propias del sistema USQL, permite la escritura de sentencias de control y funciones propias utilizadas en el manejo de la información.

4.1.1 Componentes del lenguaje USQL

En esta sección se presentan todos los elementos e instrucciones que soportará este lenguaje y que pueden ser ingresados para la administración y manipulación de la información.

Elementos USQL

Usuarios

Un usuario del sistema FISQL es aquel que puede acceder al sistema y hacer uso de él. Por defecto el sistema viene configurado con un usuario admin y contraseña admin con los permisos para manipular todo tipo de información del sistema. Todas las bases de datos creadas deben de tener permisos para este usuario.

Bases de datos

El objeto USQL bases de datos es conjunto de tablas, procedimientos y objetos diseñada para un acceso fácil y rápido.

Tablas

El objeto USQL tabla es un conjunto de elementos definidos, los objetos USQL Table deben de pertenecer a una base de datos.

Procedimientos almacenados

El objeto USQL procedimientos almacenados es un conjunto de instrucciones que se guardara en una base de datos para ser utilizada posteriormente.

Objetos almacenados

Los usuarios serán capaces de crear sus propios objetos, y estos a su vez pueden ser utilizados dentro de las tablas y procedimientos.

Llaves primarias

Es un tipo de campo que puede ser usado en las tablas USQL y servirá para identificar los registros dentro de la tabla, un campo que es llave primaria tiene las propiedades

de No Nulo (no puede ser vacío), y Único (no se puede repetir con otro campo de la misma tabla) de forma irrevocable.

Llave foránea

Es un tipo de campo que puede ser usado en las tablas USQL y servirá para denotar que existe relación entre este campo y la llave primaria de otra tabla que debe de ser especificada. Un campo que es llave primaria tiene las propiedades de No Nulo (no puede ser vacío) de forma irrevocable.

Tipos de datos

Para las variables y funciones se permiten los tipos de datos listados en la siguiente tabla.

Tipo de dato	Descripción	Ejemplos
Text	Tipo de dato que maneja cadenas de caracteres, el inicio y el fin de la cadena debe de ser comillas dobles "cadena". En el tipo de dato Text se permite el uso de caracteres de escape “\n” y “\t”.	“Cadena de ejemplo” “Hola mundo\n”
Integer	Tipo de dato que maneja valores numéricos enteros.	23 -55 92 42
Double	Tipo de dato que maneja valores numéricos decimales o enteros.	11.5 -30.33 0.05
Bool	Este tipo de dato consiste únicamente en dos tipos de resultados los cuales son valores “verdaderos” y “falsos”, la representación para los mismos es 1 para verdadero y 0 para falso;	1 0
Date	Tipo de datos para el manejo de fechas. El formato definido es dd-MM-yyyy, en donde "dd" denota día del mes, "MM" denota mes del año y "yyyy" denota el año.	15-07-2017
DateTime	Tipo de datos para el manejo de fecha y hora. El formato definido es dd-MM-yyyy hh:mm:ss, en donde "dd" denota día del mes, "MM" denota mes del año, "yyyy" denota el año, "hh" denota la hora en formato de 24hrs, "mm" denota los minutos y "ss" denota los segundos.	15-07-2017 18:00:00

Expresiones Aritméticas

El lenguaje USQL tendrá las operaciones aritméticas básicas que son: suma, resta, multiplicación, división y potencias.

	BOOL	INTEGER/DOUBLE	DATE/DATETIME	TEXT
	SUMA (+)			
BOOL	OR	Suma	Error	Concatenación
INTEGER/DOUBLE	Suma	Suma	Error	Concatenación
TEXT	Concatenación	Concatenación	Error	Concatenación

DATE/DATETIME	Error	Error	Error	Concatenación
	RESTA (-)			
Booleano	Error	Resta	Error	Error
Número	Resta	Resta	Error	Error
Cadena	Error	Error	Error	Error
DATE/DATETIME	Error	Error	Error	Error
	MULTIPLICACION (*)			
Booleano	AND	Multiplicación	Error	Error
Número	Multiplicación	Multiplicación	Error	Error
Cadena	Error	Error	Error	Error
DATE/DATETIME	Error	Error	Error	Concatenación
	DIVISION (/)			
Booleano	Error	División	Error	Error
Número	División	División	Error	Error
Cadena	Error	Error	Error	Error
DATE/DATETIME	Error	Error	Error	Concatenación
	POTENCIA (*)			
Booleano	Error	Potencia	Error	Error
Número	Potencia	Potencia	Error	Error
Cadena	Error	Error	Error	Error
DATE/DATETIME	Error	Error	Error	Concatenación

NOTA: Está permitido las operaciones con las funciones. Ej. 5 – Suma(9, 10);

Precedencia de operadores y símbolos de agrupación

En la tabla que se presenta a continuación se define la precedencia y asociatividad de cada uno de los operadores definidos anteriormente, ordenados de menor a mayor precedencia. Los paréntesis son los únicos símbolos de agrupación a utilizar.

Símbolo precedencia asociatividad

Símbolo	Precedencia	Asociatividad
+ -	1	Izquierda
* /	2	Izquierda
^	3	Derecha
- (unario)	4	No aplica
== != < > <= >=	5	No aplica
||	6	Izquierda
&&	7	Izquierda
!	8	Izquierda
( )	9	No aplica

Nombre de variables

Los nombres de las variables están definidos como una sucesión de caracteres que inicia con una arroba (@), seguido de una letra que puede estar sucedida por cero o más caracteres como letras, dígitos o guiones bajos.

Condiciones

Las condiciones son operaciones que sirven para determinar la verdad o falsedad de la misma. Para ello se hace uso de operaciones aritméticas, relacionales y lógicas.

Expresiones relacionales

Las operaciones relacionales sirven para comparar dos expresiones, el sistema debe de manejar las operaciones de igualdad, diferente, menor, mayor, menor o igual y mayor o igual.

Nombre	Símbolo	Descripción
Igual	==	Esta operación comprueba si dos expresiones tienen el mismo valor, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Diferente	!=	Esta operación comprueba si dos expresiones tienen distinto valor, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Menor que	<	Esta operación comprueba si la primera expresión es menor a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Mayor que	>	Esta operación comprueba si la primera expresión es mayor a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Menor o igual	<=	Esta operación comprueba si la primera expresión es menor o igual a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).
Mayor o igual	>=	Esta operación comprueba si la primera expresión es mayor o igual a la segunda expresión, de ser así retorna 1 (verdadero) de lo contrario retorna 0(falso).

Expresiones lógicas

También se pueden utilizar operaciones lógicas. Sus operandos tienen que ser de tipo booleano, de lo contrario se debe reportar error. El resultado de la operación tiene que ser de tipo booleano.

Nombre	Símbolo	Descripción
AND	&&	Este operador comprueba que tanto la primera expresión como la segunda expresión tengan valor verdadero. Si esto se cumple retorna 1 (verdadero) de lo contrario retorna 0(falso).
OR	||	Este operador comprueba que la primera expresión o la segunda expresión tenga valor verdadero. Si esto se cumple retorna 1 (verdadero) de lo contrario retorna 0(falso).
NOT	!	Este operador cambia el valor de la expresión de la que viene acompañada.

4.1.2 Sentencias USQL

En esta sección se detallan todas las sentencias que pueden ser escritas y manipuladas por el DBMS.

Sentencias DDL

Crear

La sentencia "crear" nos permite, como su nombre lo indica, crear un objeto USQL anteriormente mencionados.

La sintaxis varía según sea el tipo de objeto USQL que se desea crear.

Crear base_datos

SINTAXIS

```
CREAR BASE_DATOS Nombre_Base_Datos;
```

Ej.

```
CREAR BASE_DATOS Proyecto1db;
```

Crear tabla

SINTAXIS

```
CREAR TABLA Nombre_Tabla (Tipo_Dato Nombre_campo
complemento_Campo, Tipo_Dato Nombre_campo complemento_Campo);
```

Complemento_Campo:

La tabla tiene un complemento opcional que incluyen los siguientes:

- Nulo o No Nulo: Este complemento se utiliza para especificar si se aceptara el valor nulo en este campo. Para este se utilizarán las palabras reservadas "Nulo" o "No Nulo";
- Autoincrementable: Este complemento se utiliza para especificar si el campo se incrementará de forma automática, para esto se ingresará la palabra reservada "Autoincrementable". Se deberá de validar que el campo sea de tipo integer.
- Llave_Primaria: Este complemento se utilizará para especificar si el campo será llave principal de la tabla. Para este se utilizará la palabra reservada "Llave_Primaria".
- Llave_Foranea: Este complemento se utilizará para especificar si el campo será llave foránea. Para este se utilizará la palabra reservada "Llave_Foranea" seguido del nombre de la tabla a la que se hace referencia. Se deberá validar que exista la tabla y el atributo que se está referenciando, así como que el tipo de dato sea el correcto. Al momento de insertar un valor en la tabla se deberá validar que el valor de la llave foránea exista en la tabla referenciada.
- Único: Este complemento se utilizará para especificar que este campo no se repetirá. Para este se utilizará la palabra reservada "Unico".

Ej.

```
CREAR TABLA Estudiante (INTEGER id Llave_Primaria
Autoincrementable, TEXT Nombre No Nulo, DATE fh_nac No Nulo, BOOL
trabaja);
```

Crear objeto

SINTAXIS

```
CREAR OBJETO Nombre_Objeto (Tipo_Dato Nombre_atributo, Tipo_Dato
Nombre_atributo);
```

Ej.

```
CREAR OBJETO Empresa (INTEGER codigo, TEXT nombre, TEXT telefono);
```

Crear Procedimiento

SINTAXIS

```
CREAR PROCEDIMIENTO Nombre_Procedimiento (Tipo_Dato
Nombre_parametro, Tipo_Dato Nombre_parametro){
      Sentencias;
}
```

Ej.

```
CREAR PROCEDIMIENTO set_Salario (DOUBLE @salario, DOUBLE
@comision, INTEGER @codigo){
              ACTUALIZAR TABLA trabajador (salario) VALORES
(@salario+@comision) DONDE codigo == @codigo;
}
```

Crear función

SINTAXIS

```
CREAR FUNCION Nombre_Funcion (Tipo_Dato Nombre_parametro,
Tipo_Dato Nombre_parametro) Tipo_Dato_Retorno{
      Sentencias;
}
```

NOTA: Entre las sentencias deberá de existir el retorno del tipo de dato correspondiente.

Ej.

```
CREAR FUNCION suma (INTEGER @var1, INTEGER @var2) INTEGER{
      RETORNO @var1 * @var2;
}
```

Retorno

Esta sentencia finalizara la función y retornara el valor correspondiente.

SINTAXIS

```
RETORNO EXPRESION;
```

Llamada de métodos y funciones

Cuando se desee hacer el llamado a un método o función almacenado se deberá de hacer de la siguiente forma:

SINTAXIS

```
Nombre_Metodo|Funcion ( Lista_valores);
```

Ej.

```
Set_Salario(8500.00,964.50);

@var_ejemplo = suma(10,15);
```

Crear usuario

Esta sentencia USQL permitirá agregar nuevos usuarios al sistema, solamente el usuario Admin podrá agregar nuevos usuarios.

SINTAXIS

```
CREAR USUARIO Nombre_Usuario COLOCAR password = Cadena;
```

Ej.

```
CREAR USUARIO usuario1 COLOCAR password = “holaMundo”;
```

Usar base_datos

Esta sentencia USQL permitirá seleccionar una base de datos previamente creada para trabajar sobre esta. Si no se ejecuta este comando previo a comandos que modifiquen elementos en una base de datos se deberá reportar el error.

SINTAXIS

```
USAR Nombre_base_datos;
```

Ej.

```
USAR Proyecto1;
```

Alterar

La sentencia alterar nos permitirá hacer modificaciones (agregar o quitar) a las estructuras de los siguientes objetos-USQL:

- Tablas
- Objetos

Alterar tablas

Permitirá agregar o quitar campos de dicha tabla;

SINTAXIS

```
ALTERAR TABLA Nombre_tabla AGREGAR (Tipo_Dato Nombre_campo
complemento_Campo, Tipo_Dato Nombre_campo complemento_Campo);

ALTERAR TABLA Nombre_tabla QUITAR Lista_Nombres_campo;
```

Ej.

```
ALTERAR TABLA Estudiante AGREGAR (INTEGER edad, TEXT direccion);

ALTERAR TABLA Estudiante QUITAR trabaja;
```

Alterar objeto

Permitirá agregar o quitar atributos de dicho objeto.

SINTAXIS

```
ALTERAR OBJETO Nombre_Objeto AGREGAR (Tipo_Dato Nombre_Atributo,
Tipo_Dato Nombre_Atributo);

ALTERAR OBJETO Nombre_Objeto QUITAR Lista_Nombres_Atributos;
```

Ej.

```
ALTERAR OBJETO empresa AGREGAR (INTEGER  nit);

ALTERAR TABLA empresa QUITAR direccion, nit;
```

Alterar usuario

Permitirá modificar la contraseña de dicho usuario. Esta sentencia la utilizara el usuario Admin para modificar la contraseña de cualquier usuario incluyendo su propia contraseña. Por otro lado, el resto de usuarios solo pueden modificar su propia contraseña.

SINTAXIS

```
ALTERO USUARIO Nombre_Usuario CAMBIAR password = Cadena;
```

Ej.

```
ALTERO USUARIO Admin CAMBIAR password = “1234”;
```

Eliminar

La sentencia eliminar nos permitirá eliminar objetos-USQL. Los objetos-USQL solamente podrán ser eliminados por el usuario Admin o el usuario que los ha creado.

SINTAXIS

```
ELIMINAR Objeto-USQL nombre_Objeto-USQL.
```

Ej.

```
ELIMINAR TABLA estudiante;

ELIMINAR BASE_DATOS Proyecto1;

ELIMINAR OBJETO empresa;

ELIMINAR USER usuario1;
```

Sentencias DDL

Insertar

La sentencia insertar servirá para agregar registros a las tablas USQL.

Insertar en tabla

De esta forma nos permitirá ingresar un registro a la tabla, deberán de ingresarse todos los campos correspondientes al registro.

SINTAXIS

```
INSERTAR EN TABLA Nombre_Tabla (Lista_valores);
```

Ej.

```
INSERTAR EN TABLA estudiante ("Jose Luis Figueroa", 10-10-1994,1);
```

Insertar en tabla(especial)

Esta opción nos dará a elegir que campos son los que se van a ingresar en la tabla correspondiente.

SINTAXIS

```
INSERTAR EN TABLA Nombre_Tabla (Lista_campos) VALORES
(Lista_valores);
```

Ej.

```
INSERTAR EN TABLA estudiante (Nombre, fh_nac) VALORES ("Jose Luis
Figueroa", 10-10-1994);
```

Actualizar

La sentencia actualizar servirá para modificar registros de las tablas USQL.

SINTAXIS

```
ACTUALIZAR TABLA Nombre_Tabla  (Lista_campos) VALORES
(Lista_valores) DONDE condiciones;
```

Ej.

```
ACTUALIZAR TABLA estudiante (nombre, trabaja) VALORES ("Jose Luis
Martinez Hernandez",1) DONDE nombre = "Jose Luis Martinez" &&
trabaja = 0;

ACTUALIZAR TABLA estudiante (nombre, trabaja) VALORES ("Carlos
Enriquez",0);
```

NOTA: si no se incluye el campo en la cláusula DONDE con sus condiciones se actualizarán todos los registros de la tabla.

Borrar

La sentencia borrar servirá para eliminar registros de las tablas USQL.

SINTAXIS

```
BORRAR EN TABLA Nombre_Tabla DONDE condiciones;
```

Ej.

```
BORRAR EN TABLA estudiante DONDE nombre = "José Luis Martínez"
&&trabaja = 0;

BORRAR EN TABLA estudiante;
```

NOTA: si no se incluye el campo en la cláusula DONDE con sus condiciones se eliminarán todos los registros de la tabla.

Seleccionar

La sentencia seleccionar es una instrucción para realizar la consulta de información, esta es la encargada de realizar búsquedas en las tablas de las bases de datos. Tiene la opción de ordenar el resultado ya sea de manera ascendente o descendente mediante la palabra reservada "ORDENAR_POR" seguido del nombre del campo con el que se hará el ordenamiento y luego las palabras reservadas ASC para ascendente o DESC para descendente.

SINTAXIS

```
SELECCIONAR lista_campos DE Lista_Tablas DONDE condiciones
ORDENAR_POR Nombre_Campo_Ordenar [ASC|DESC];
```

La cláusula DONDE con sus condiciones y la cláusula ORDENAR POR serán opcionales.

Ej.

```
SELECCIONAR nombre, fh_nac DE estudiante DONDE id > 10 ORDENAR POR
fh_nac;

SELECCIONAR * DE estudiante, asignacion DONDE estudiante.id ==
asignación.id_estudiante;

SELECCIONAR nombre, fh_nac DE estudiante DONDE id < 10;

SELECCIONAR * DE estudiante;
```

NOTA: si se usa el asterisco (*) en lugar de la lista de campos se realizará la consulta de todos los campos en el orden que aparecen almacenados en el sistema.

Sentencias DCL

Otorgar

Esta sentencia de USQL permitirá asignar permisos a los usuarios de acceder a los objetos-USQL de la base de datos. Solamente el usuario Admin podrá dar permisos a otros usuarios. Al momento de crear usuarios, este ya tendrá permisos por defecto para crear cualquier objeto-USQL. Si no se tienen permisos sobre un objeto-USQL este no podrá ver su información, editarla, borrarla ni utilizarla de ninguna forma.

SINTAXIS

```
OTORGAR PERMISOS Nombre_Usuario, Nombre_Base_datos.Nombre_objeto-
USQL;
```

Ej.

```
OTORGAR PERMISOS usuario1, Proyecto1.estudiante;
OTORGAR PERMISOS usuario1, Proyecto1.*;
```

NOTA: Al momento de crear un Objeto-USQL, automáticamente se darán los permisos sobre estos.

NOTA: Si se desea otorgaran permisos a todos los objetos-USQL pertenecientes a la base de datos se podrá usar el símbolo asterisco (*).

Denegar

Esta sentencia de USQL permitirá denegar permisos a los usuarios para usar los objetos-USQL de las bases de datos. Solamente el usuario Admin puede denegar permisos a otros usuarios.

SINTAXIS

```
DENEGAR PERMISOS Nombre_Usuario, Nombre_Base_datos.Nombre_objeto-
USQL;
```

Ej.

```
DENEGAR PERMISOS usuario1, Proyecto1.estudiante;
DENEGAR PERMISOS usuario1, Proyecto1.*;
```

NOTA: No se podrá quitar permisos al usuario admin.

NOTA: Si se desea denegar permisos a todos los objetos-USQL pertenecientes a la base de datos se puede usar el símbolo asterisco (*).

Sentencias SSL

DECLARACIÓN DE VARIABLES

Para la declaración de variables, la sintaxis será la siguiente: Primero se escribirá la palabra reservada DECLARAR, seguido por una lista de identificadores separados por coma y luego el tipo de dato de la variable, para finalizar siempre con un punto y coma (;). De manera opcional, se podrá realizar una asignación inicial anteponiendo al punto y coma un signo igual (=) seguido de una expresión.

SINTAXIS

```
DECLARAR Lista_variables Tipo_Dato [ = Expresión ] ;
```

Ej.

```
DECLARAR @variable1, @variable2 INTEGER = 23;
DECLARAR @ejemplo TEXT = “Hola mundo”;
```

INSTANCIA DE OBJETOS

El lenguaje permitirá crear instancias de los objetos almacenados previamente. Para esto se utilizará la palabra reservada DECLARAR seguida del nombre de la instancia, posteriormente se coloca el nombre del objeto almacenado;

SINTAXIS

```
 DECLARAR nombre_instancia nombre_Objeto_Almacenado;
```

Ej.

```
DECLARAR @empresa1 EMPRESA;
```

ASIGNACIÓN DE VARIABLES

Para la asignación de variables, la sintaxis será la siguiente: Primero se escribirá el identificador, seguido del signo igual (=), luego la expresión que representa el nuevo valor para la variable seguido de punto y coma (;) .

SINTAXIS

```
Nombre_variable = Expresión;
```

Ej.

```
@a = 3 * 23;
@ejemplo = saludo("Hola");
```

ASIGNACIÓN A LOS ATRIBUTOS DE UN OBJETO

Para asignarle valor a los atributos de las instancias de los objetos, se realizará por medio del nombre de la instancia seguida de un punto (.) y el nombre del atributo.

SINTAXIS

```
Nombre_instancia.Nombre_Atributo = Expresión ;
```

Ej.

```
@empresa1.nit = 87654321-0;
```

SI

Esta instrucción iniciara con la palabra reservada "SI", seguida de una condición encerrada entre paréntesis. Contará con un cuerpo que consiste en un conjunto de sentencias encerradas entre llaves ({}), que se ejecutan si la condición tiene un valor verdadero. De manera opcional se puede incluir un conjunto de sentencias que se ejecuten en caso de que la condición sea falsa, este cuerpo de instrucciones viene seguido de la palabra reservada "SINO", este cuerpo de instrucciones también deberá ir encerrado entre llaves ({}).

SINTAXIS

```
SI(  <CONDICION> ){
      < INSTRUCCIONES >
}
[ SINO {
      < INSTRUCCIONES >
}]
```

Ej.

```
SI( @a < 10 ){
      @b = 23;
}SINO{
      @b = 46;
}
```

SELECCIONA

Esta sentencia de control evaluará una expresión y compara su resultado con cada uno de los valores definidos en los casos del cuerpo de la sentencia. Cada caso cuenta con un valor y un cuerpo de instrucciones.

La sentencia de control iniciara con la palabra reservada “SELECCIONA”, seguido de una expresión encerrada entre paréntesis. Contará con una lista de casos, cada caso iniciara con la palabra reservada “CASO” seguida del valor puntual del caso, dos puntos (:), y un cuerpo de sentencias. Opcionalmente, se podrá agregar un caso que se ejecute por defecto si ninguno de los casos cumple con la expresión inicial con la palabra reservada “DEFECTO” seguida por dos puntos (:) y un cuerpo de sentencias.

Los tipos de datos que permitirá la sentencia de control son *Text, Integer y Double.* Si se evalúa una expresión de tipo *Text*, los valores para los casos deben ser valores de tipo *Text*, de igual forma para el tipo *Integer y Double.*

El comportamiento de esta sentencia de control se preguntará cuál es el caso que cumple con la expresión inicial, y de ahí en adelante ejecutara todas las sentencias de los casos, sin incluir las instrucciones asociadas al bloque “DEFECTO”, hasta encontrar la palabra reservada “DETENER”. Si no se cumple con ninguna de las condiciones se deberá ejecutar el bloque de instrucciones asociadas al bloque “DEFECTO”.

SINTAXIS

```
SELECCIONA (  < EXPRESION > ){
      CASO < VALOR1 > :
            < INSTRUCCIONES >
      CASO < VALOR2 > :
            < INSTRUCCIONES >
      [ CASO < VALORN >:
            < INSTRUCCIONES >]
      [ DEFECTO :
            < INSTRUCCIONES >]
}
```

Ej.

```
SELECCIONA (  @a * 15 ){
      CASO 10 :
            < INSTRUCCIONES >
      CASO 15 :
            < INSTRUCCIONES >
      DEFECTO :
            < INSTRUCCIONES >
}
```

CICLO PARA

Este ciclo comienza con la palabra reservada “PARA”, consta de una declaración inicial de una variable de tipo *Integer*, seguida de punto y coma; una condición para mantener el ciclo en ejecución seguida de punto y coma; un incremento (++) o decremento (--), que al final de cada iteración afectará directamente el valor de la variable declarada al inicio de este ciclo; y un cuerpo de instrucciones a ejecutar siempre y cuando la condición se cumpla. Se evaluará la condición, si es verdadera ejecutara el cuerpo y cambia el valor de la variable inicial, si es falsa sale del ciclo.

SINTAXIS

```
PARA(  DECLARAR < ID > INTEGER =  < EXPRESION > ; < CONDICION >; <
OPERADOR > ){
      < INSTRUCCIONES >
}
```

Ej.

```
PARA(  DECLARAR @a = 10; @a > 0; --){
      @b = 23;
}
```

CICLO MIENTRAS

Este ciclo iniciara con la palabra reservada “MIENTRAS”, seguida de una condición encerrada entre paréntesis. Contará con un cuerpo que consiste en un conjunto de sentencias encerradas entre llaves ({}) que se ejecutara toda vez la condición sea verdadera.

SINTAXIS

```
MIENTRAS(  <CONDICION> ){
      < INSTRUCCIONES >
}
```

Ej.

```
MIENTRAS(  @a < 10 ){
      @a = @a + 1;
}
```

DETENER

Esta sentencia se ejecutará siempre y cuando sea encuentre dentro del cuerpo de un ciclo *PARA o MIENTRAS,* o dentro del cuerpo de un caso de una sentencia SELECCIONA. Al ser encontrada la ejecución deberá ser detenida y regresar a un nivel superior de la sentencia o ciclo que será detenido.

SINTAXIS

```
DETENER;
```

Ej.

```
MIENTRAS(  @a < 10 ){
      @a = @a + 1;
      DETENER;
}
```

IMPRIMIR

Esta instrucción permitirá mostrar en la consola de salida una cadena de texto. Esta instrucción solamente recibirá como parámetro una expresión que dé como resultado una cadena de tipo TEXT.

SINTAXIS

```
IMPRIMIR(Expresión);
```

Ej.

```
IMPRIMIR("Hola mundo!");
```

FECHA

La función FECHA retornara la fecha del sistema en formato DATE;

SINTAXIS

```
FECHA();
```

Ej.

```
@fecha = FECHA();
```

FECHA_HORA

La función FECHA_HORA retornara la fecha y hora del sistema en formato DATETIME;

SINTAXIS

```
FECHA_HORA();
```

Ej.

```
@fecha_h = FECHA_HORA();
```

CONTAR

Esta instrucción recibirá como parámetro un SELECCIONAR encerrado en los símbolos "<<" ">>" y retorna el número de registros que tiene dicho SELECCIONAR.

SINTAXIS

```
CONTAR(<< SELECCIONAR>>);
```

Ej.

```
CONTAR(<<SELECCIONAR * DE estudiante>>);
```

Backups

Los backups o copias de seguridad servirán para sacar un respaldo de la información, por seguridad se recomienda sacar backups periódicamente para que, en caso de pérdida o daño de la información de la base de datos, se tenga una copia para poder recuperarla. Se manejarán dos tipos de backups, usqldump y backups completos.

BACKUP USQLDUMP

Función propia de USQL que permitirá sacar un archivo que contenga todas las instrucciones ingresadas, para ello cada base de datos deberá de guardar un registro tipo bitácora y al momento de accionar un bakcup de tipo USQLDUMP se enviara una copia al usuario. El archivo final debe de tener una extensión udmp.

SINTAXIS

```
BACKUP USQLDUMP Nombre_Base_dato nombre_archivo_backup;
```

Ej.

```
BACKUP USQLDUMP Proyecto1 Proyecto1_backup;
```

NOTA: El sistema colocara la extensión .udmp al devolver el archivo, no será necesario que el usuario lo ingrese.

BACKUP COMPLETO

Función propia de USQL que permitirá sacar una copia de todos los archivos que maneja la base de datos internamente en forma comprimida (zip) y se las manda al usuario.

SINTAXIS

```
BACKUP COMPLETO Nombre_Base_datos Nombre_archivo_backup;
```

Ej.

```
BACKUP COMPLETO Proyecto1 Proyecto1_backup2;
```

NOTA: El sistema colocara la extensión .zip al devolver el archivo, no será necesario que el usuario lo ingrese.

Restaurar backup

El restaurar son instrucciones que permiten cargar la información de un backup.

RESTAURAR BACKUP USQLDUMP

SINTAXIS

```
RESTAURAR USQLDUMP “ruta_archivo”;
```

Ej.

```
RESTAURAR USQLDUMP “/home/usuario/backups/backup_prueba.udmp”
```

RESTAURAR BACKUP COMPLETO

SINTAXIS

```
RESTAURAR COMPLETO “ruta_archivo”;
```

Ej.

```
RESTAURAR COMPLETO “/home/usuario/backups/backup_prueba.zip”
```

Comentarios

Dentro del lenguaje se permitirá escribir comentarios de una sola línea y comentarios de múltiples líneas. Los comentarios de una sola línea empezarán con el símbolo numeral (#) y terminaran con un salto de línea. Los comentarios de múltiples líneas comenzaran con un símbolo numeral seguido de un asterisco (*) y terminaran con un asterisco (*) seguido de un símbolo numeral (#).

SINTAXIS

```
# Comentario de una línea
#* Comentario de varias líneas *#
```

Ejemplos lenguaje USQL

Ejemplo de creación de base de datos y sus elementos.

```
CREAR BASE_DATOS Ejemplo1;

USAR Ejemplo1;

CREAR OBJETO Direccion(INTEGER avenida, INTEGER calle, TEXT
nombre, TEXT descripcion);

CREAR TABLA Estudiante (
INTEGER id Llave_Primaria Autoincrementable,
TEXT Nombre No Nulo,
DATE fh_nac No Nulo,
BOOL trabaja,
Direccion direccion);

CREAR TABLA Curso(
INTEGER id Llave_Primaria Autoincrementable,
TEXT Nombre No Nulo,
INTEGER creditos No Nulo);

CREAR TABLA Asignacion(
INTEGER id Llave_Primaria Autoincrementable,
DATETIME fh_Asignacion No Nulo,
INTEGER id_estudiante Llave_Foranea Estudiante,
INTEGER id_curso Llave_Foranea Curso);

PREAR ROCEDIMIENTO Asignar (INTEGER @id_e, INTEGER @id_c){
    DECLARAR @estudiante INTEGER= CONTAR(<<SELECCIONAR * DE
estudiante DONDE id  ==  @id_e>>);
    DECLARAR @curso INTEGER= CONTAR(<<SELECCIONAR * DE curso DONDE
id  ==  @id_c>>);
    SI (@curso > 0){
        SI(@estudiante > 0){
             INSERTAR EN TABLA asignacion(id_estudiante, id_curso)
VALORES (@id_e, @id_c);
        }SINO{
            IMPRIMIR("El usuario ingresado no existe");
        }
    }SINO{
        IMPRIMIR("El curso ingresado no existe");
    }
}

CREARFUNCION crearDireccion (INTEGER @calle, INTEGER @avenida,
TEXT @nombre, TEXT @descripcion) Direccion{
    DECLARAR @direc DIRECCION;
    @direc.calle = @calle;
```

```
    @direc.avenida = @avenida;
    @direc.nombre = @nombre;
    @direc.descripcion = @descripcion;
    RETORNO @direc;
}
```

Ejemplo de creación de objetos USQL, carga de información y consultas.

```
CREAR BASE_DATOS Ejemplo2;
USAR Ejemplo2;

CREAR TABLA Proveedor(
  INTEGER id Llave_Primaria Autoincrementable,
  TEXT nombre No Nulo,
  TEXT direccion No Nulo
);

CREAR TABLA Producto(
  INTEGER id Llave_Primaria Autoincrementable,
  TEXT nombre No Nulo,
  DOUBLE precio No Nulo,
  INTEGER id_proveedor Llave_Foranea Proveedor
);

CREAR PROCEDIMIENTO llenarTablas(){
  DECLARAR @producto INTEGER = 0;
  #inserta 10 registros en la tabla proveedor
  PARA(DECLARAR @i INTEGER = 0; @i < 10; ++){
    INSERTAR EN TABLA Proveedor(id, nombre, direccion)
      VALORES (@i, "Proveedor" + @i, "Dirección" + @i);

      #inserta 20 productos por proveedor
      PARA(DECLARAR @j INTEGER = 0; @j < 20; ++){
        DECLARAR @precio DOUBLE = generarPrecio(@producto);
        INSERTAR EN TABLA Producto
          (@producto, "Producto" + @producto, @precio, @i);
        @producto = @producto + 1;
      }
  }
}

CREAR FUNCION generarPrecio(INTEGER @a){
  DECLARAR @retorno DOUBLE = 1.5;
  RETORNO @retorno * @a;
}
```

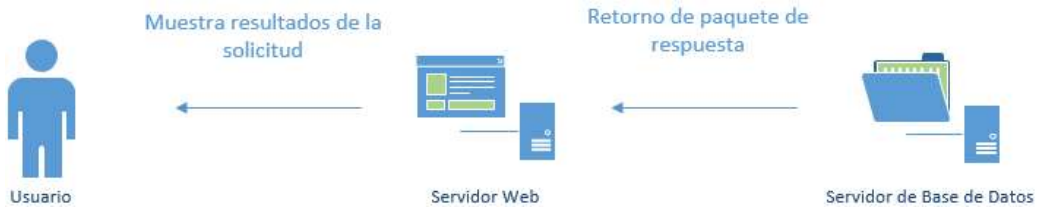
```
#llenar las tablas
llenarTablas();

#imprimir cantidad de registros de la tabla Producto
IMPRIMIR(COUNT(<<SELECCIONAR * DE Producto>>));
#el resultado de la instrucción debería ser:
#200

#borrar todos los productos del proveedor con id 1
BORRAR EN TABLA Producto DONDE id_proveedor Llave_Foranea = 1;

#imprimir cantidad de registros de la tabla Producto
IMPRIMIR(COUNT(<<SELECCIONAR * DE Producto>>));
#el resultado de la instrucción debería ser:
#180
```

4.2 Lenguaje de PlyCS

El sistema deberá ser capaz de establecer comunicación entre el servidor de base de datos y el servidor web, esta comunicación deberá realizarse por medio de sockets utilizando un lenguaje de comunicación intermedio entre los servidores, que se describen a continuación:

I. El usuario siempre iniciará la comunicación con el servidor web donde se alojará el IDE y éste le comunicará al servidor de Base de datos que existe una nueva conexión.

II. El servidor de base de datos deberá responder la solicitud del servidor web y comenzará a transmitir la información mediante el socket.



III. El servidor web enviará un mensaje de fin de sesión al servidor de base de datos para indicar que la sesión ha finalizado.

IV. Esta comunicación podrá realizarse varias veces según sea necesario durante la sesión.

El usuario por medio del servidor web podrá realizar cualquier solicitud del siguiente listado:

1. Inicio de sesión.
2. Fin de sesión
3. Instrucción USQL
4. Esquema de base de datos.
5. Reportes

4.2.1 Descripción de paquetes

Los paquetes que se utilizarán para la comunicación entre el servidor web y el servidor de base de datos están basados en el lenguaje de intercambio de datos PlyCS, similar al lenguaje JSON, cada fase de la comunicación contará con un paquete específico que el servidor de base de datos o el servidor web deberá reconocer y realizar la operación de analizar y reconocer los valores que estos paquetes contienen. Todas las solicitudes que realice el servidor web deberán llevar un número entero aleatorio que será devuelto en la respuesta del servidor para verificar que la respuesta corresponde a la solicitud inicial del servidor web.

4.2.2 Inicio de sesión (login)

El paquete de login contendrá información que se utilizará para autenticar al usuario.

Paquete de login (envía servidor web)	Paquete de login (envía servidor de base de datos)
[“validar”: 1500, “login”: [“comando” => “seleccionar * de usuarios donde usuario = ‘juan’ && password => ‘compi2’”]]	[“validar”: 1500, “login”: [“usuario”: “juan”, “nombre”:”juan pedro”, “login”: true]]

4.2.3 Paquete de finalización

Este paquete indicará al servidor de base de datos que el servidor web finalizó la sesión y que se debe reiniciar el proceso de comunicación para hacer una nueva solicitud.

Paquete de finalización (envía servidor web)
[“paquete”: “fin”]

4.2.4 Paquete de error

Este paquete notificará al servidor web que ha ocurrido un error, en alguno de los procesos.

Paquete de error	Tipos de error
[“paquete”: “error”, “tipo”: “lenguaje”, “msg”: “ha ocurrido un error léxico en el lenguaje”, “datos”: [“Error”: “Léxico”, “Archivo”: “seleccionar * from usuarios”, “col”: 14, “fila”: 1]]	->Lenguaje: debe presentar qué error de lenguaje ha ocurrido, en que archivo o instrucción y la posición. ->Otro: únicamente debe presentar un mensaje que indique cual fue la causa del error.

4.2.5 Paquete de instrucción USQL

Este paquete trasladará desde el servidor web todas las instrucciones USQL que el usuario ingrese, al servidor de base de datos para su procesamiento y retorno de la respuesta.

Paquete de instrucción (envía servidor web)	Paquete de instrucción (envía servidor de base de datos)
[“paquete”: “usql”, “instrucción”: “seleccionar * de usuarios;”,]	[“paquete”: “usql”, “datos”: [[“usuario”=>1 “nombre”=>”paco” “telefono”=>12345678] [“usuario”=>2 “nombre”=>”pedro” “telefono”=>12345678]]]

4.2.6 Paquete de instrucción Reporte

Este paquete trasladará desde el servidor web todas las instrucciones “seleccionar” que el usuario ingrese en el módulo de reportes, al servidor de base de datos para su procesamiento y retorno de la respuesta.

Paquete de Reporte (envía servidor web)	Paquete con información para el reporte (envía servidor de base de datos)
[“paquete”: “reporte”, “instrucción”: “seleccionar * de usuarios;”,]	[“paquete”: “reporte”, “datos”: [<tabla> <tr> <th>usuario</th> <th>nombre</th> <th>telefono</th> </tr> <tr> <td>1</td> <td>Paco</td> <td>12345678</td> </tr> <tr> <td>2</td> <td>Pedro</td> <td>12345678</td> </tr> </tabla>]]

4.3 Lenguaje XML

Este lenguaje será el utilizado para el almacenamiento de la información. Será XML estándar y la estructura de los archivos se detallan en la sección 5.1. Para los lenguajes PlyCS y XML, dado a que estos serán implementados por el estudiante de forma interna en los distintos servidores (web y de base de datos), se debe de verificar la estructura correcta de cada uno.

5 Manejo y almacenamiento de información

En esta sección se presenta la forma en la que se manejará y almacenará la información en el servidor de base de datos luego de realizar una instrucción.

5.1 Sistema de Archivos

Para el almacenaje de información se usará la estructura de archivos xml, con una estructura ya definida. Existen 5 tipos de archivos para el almacenaje de la información, estos son los siguientes:

5.1.1 Archivo Maestro

Solamente podrá existir un archivo maestro con la información general de las bases de datos, usuario y demás información que detalle el funcionamiento general del DBMS. Este archivo contiene la lista de todas las bases de datos existentes, incluyendo el nombre y la dirección del archivo de la base de datos. La estructura será la que se muestra en el siguiente cuadro.

Registros MAESTRO

```
<DB>
        <nombre>DB1</nombre>
        <path>C:/Documentos/BD/DB1.usac</path>
</DB>
<DB>
        <nombre>DB2</nombre>
        <path>C:/Documentos/BD/DB2.usac</path>
</DB>
```

5.1.2 Archivo DB

Existirá un archivo DB por cada base de datos creada, este almacenará la información de las tablas, procedimientos y objetos que le pertenezcan. La estructura será la que se muestra en el siguiente cuadro.

Registros DB

```
<Procedure>
        <path>C:/Documentos/BD/proc.usac</path>
</Procedure>
<Object>
        <path>C:/Documentos/BD/obj.usac</path>
</ Object >
<Tabla>
        <nombre>nombre_tabla1</nombre>
        <path>C:/Documentos/BD/nombre_tabla1.usac</path>
        <rows>
                <int>Nombre_campo1</int>
                <text>Nombre_campo2</text>
                <bool>Nombre_campo3</bool>
        </rows>
</Tabla>
```

5.1.3 Archivo Tabla

Existirá un archivo Table por cada tabla creada, este archivo almacenará todas las tuplas de una tabla con sus campos correspondientes de acuerdo con lo definido al momento de crearla. La estructura será la que se muestra en el siguiente cuadro.

```
Registros TABLA
<Row>
        <Nombre_campo1> 1 </Nombre_campo1>
        <Nombre_campo2> “Texto” </Nombre_campo2>
        <Nombre_campo3> True </Nombre_campo3>
</Row>
<Row>
        <Nombre_campo1> 2 </Nombre_campo1>
        <Nombre_campo2> “Texto” </Nombre_campo2>
        <Nombre_campo3> False </Nombre_campo3>
</Row>
```

5.1.4 Archivo Método

Existirá un archivo Método por cada base de datos creada y almacenará todos los procedimientos y funciones almacenadas. La estructura de este archivo permitirá almacenar el nombre del procedimiento, los parámetros de entrada y salida, así como las instrucciones definidas en el cuerpo de la misma. La estructura será la que se muestra en el siguiente cuadro.

```
Registros PROCEDURE
<Proc>
< nombre >nombre_proc1</ nombre >
        <params>
                <int>Nombre_param1</int>
                <text>Nombre_param2</text>
                <bool>Nombre_param3</bool>
        </params>
        <src> Instrucciones </src>
</ Proc >
<Proc>
        <nombre>nombre_proc2</ nombre >
        <params>
                <int>Nombre_param1</int>
                <text>Nombre_param2</text>
                <bool>Nombre_param3</bool>
        </params>
        <src> Instrucciones </src>
</ Proc >
```

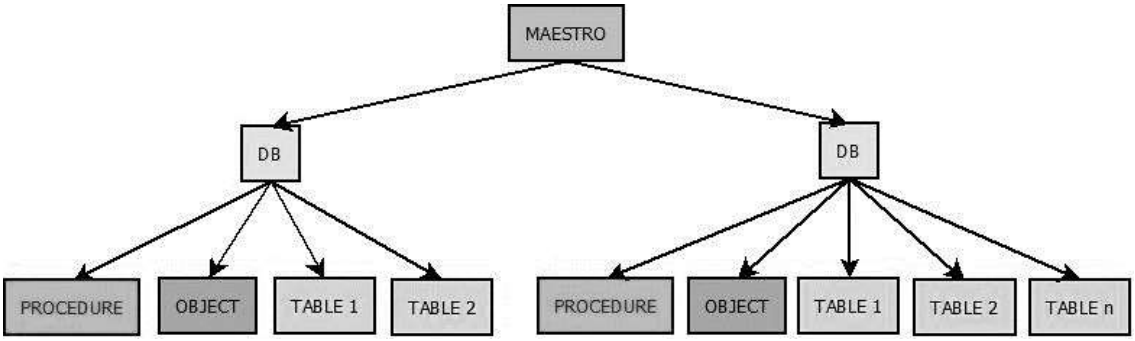
5.1.5 Archivo Objeto

Existirá un archivo objeto por cada base de datos creada y almacenará la información de todos los objetos declarados de la base de datos. En este archivo se guardarán los nombres de cada objeto y sus atributos, así como el tipo de cada uno de ellos. La estructura será la que se muestra en el siguiente cuadro.

```
                    Registros OBJECT
<Obj>
        < nombre >nombre_obj1</ nombre >
        <attr>
                <int>Nombre_param1</int>
                <text>Nombre_param2</text>
                <bool>Nombre_param3</bool>
        </attr>
</ Obj >
<Obj >
        <nombre>nombre_obj2</ nombre >
        <attr>
                <int>Nombre_param1</int>
                <text>Nombre_param2</text>
                <bool>Nombre_param3</bool>
        </attr>
</ Obj >
```

5.1.6 Estructura general del sistema de archivos:

La estructura general del sistema de archivos del DBMS será la de un árbol, tal como se muestra en la siguiente imagen.

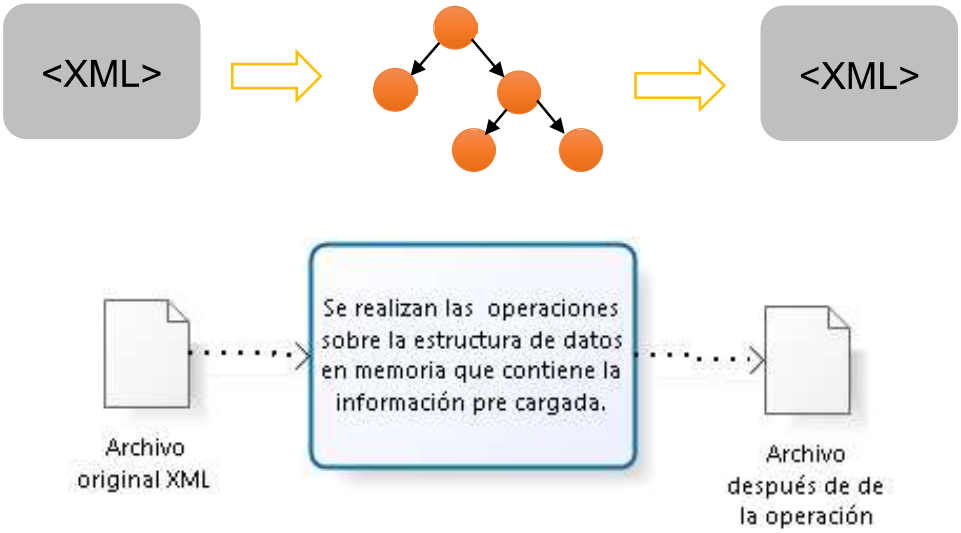


5.1.7 Intérprete XML

El intérprete XML será el encargado de manejar la estructura y almacenamiento de la información en el servidor, por medio de archivos escritos en lenguaje XML. El intérprete, antes de realizar alguna operación deberá verificar la integridad de estos archivos.

El servidor de base de datos, para poder almacenar la información, manejará los cinco tipos de archivos XML.

El estudiante debe de desarrollar un intérprete para el manejo de estos archivos, este será el encargado de realizar la carga de estos archivos a memoria en forma de una estructura de datos, para facilitar las búsquedas y manipulación de la información, tal como se muestra en el diagrama siguiente.



De acuerdo con el tipo de instrucción que se le enviará al servidor de base de datos, el sistema debe de manipular los distintos archivos, para poder realizar la operación solicitada. Por ejemplo, si se crea una nueva base de datos, se debe de agregar un nuevo registro al archivo maestro, se debe de crear el archivo BD correspondiente, así como los archivos tabla, método y objeto. De la misma forma si es una instrucción seleccionar, se deberá de verificar la existencia de los archivos correspondientes para poder realizar las manipulaciones necesarias sobre los archivos, en este caso la unión de uno o varios archivos Table. Y esto para casos como seleccionar, actualizar y todas las instrucciones soportadas por el sistema (USQL).

NOTA: Los esquemas mostrados para los archivos xml muestran un caso general, a la hora de la implementación el estudiante puede agregar las etiquetas que considere necesarios para el correcto funcionamiento de su sistema.

6 Reportes

Este módulo del sistema está diseñado para que se puedan crear reportes personalizados por medio del lenguaje html en conjunto con una serie de instrucciones adicionales.

El módulo contará con su propio editor de texto que permitirá el ingreso del código html junto con consultas (seleccionar) USQL. Al finalizar el reporte el usuario puede ejecutarlo y de no tener errores se podrá ver el reporte desde el navegador. De existir un error debe de notificarlo en la consola de salida.

Etiquetas html permitidas:

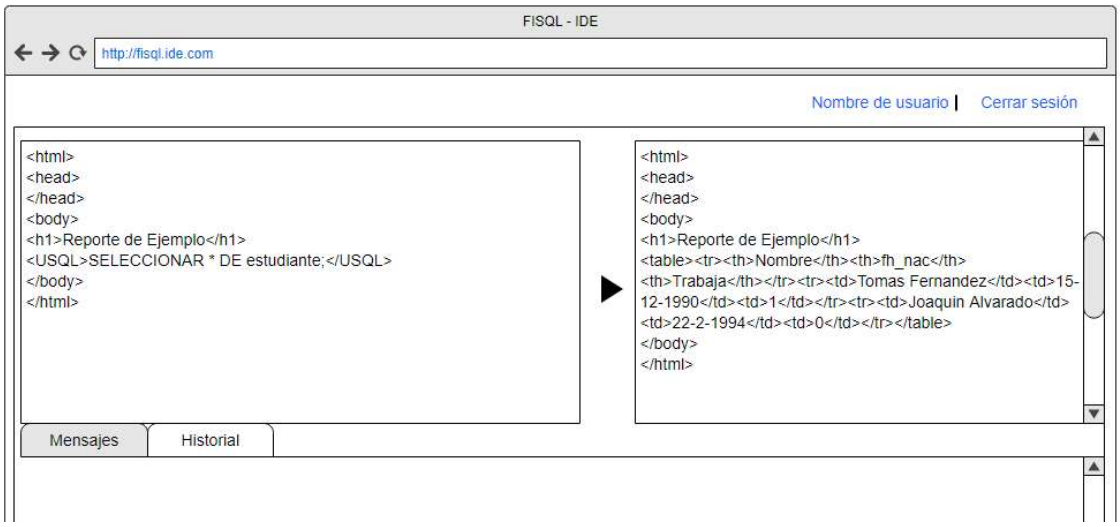
Etiqueta	Parámetros opcionales	Descripción
<HTML>		Etiqueta inicial del documento html.
<HEAD>		Etiqueta que denota el área declarativa del documento html.
<BODY>	Bgcolor = "color"	Etiqueta que denota el área del cuerpo de html.
<HEADER>		Etiqueta que denota un encabezado dentro del body.
<TABLE>		Etiqueta para el manejo de tablas en el documento html.
<TR>		Etiqueta para el manejo de filas dentro de las tablas en html.
<TD>		Etiqueta para el manejo de columnas dentro de las filas en html.
<TH>		Etiqueta para el manejo de columnas de tipo encabezado dentro de las filas en html.
<b>		Etiqueta que denota que el texto interno debe de mostrarse en negrita.
	Align = "[left|center|rigth]"	Etiqueta para el manejo de subsecciones dentro del body.

Dentro del lenguaje html se deberá permitir el ingreso de la etiqueta <USQL><USQL> Que permitirá realizar consultas a la base de datos y retornará la tabla de resultados en formato html.

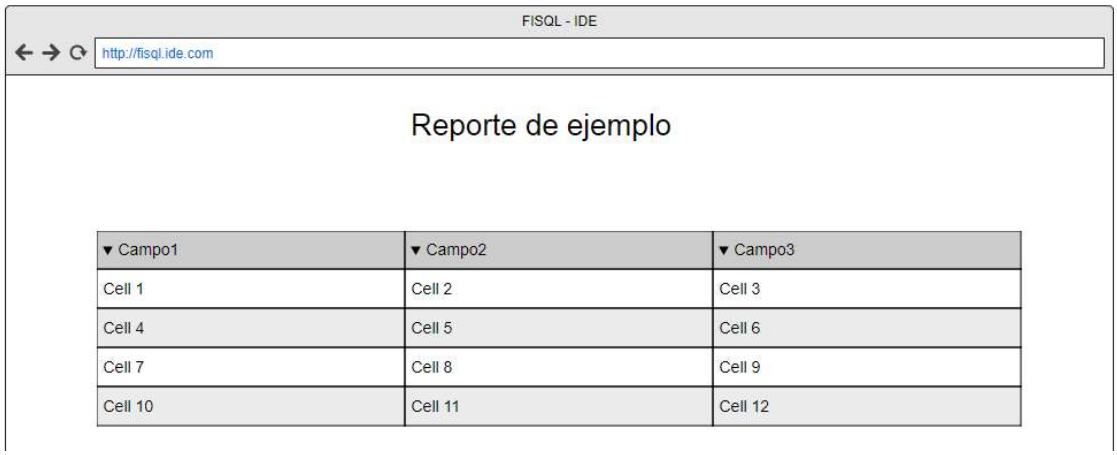
Ejemplo de código de entrada y salida:

```
Entrada:
<html>
  <head>
  </head>
  <body>
    <h1>Reporte de Ejemplo</h1>
    <USQL>
      SELECCIONAR * DE estudiante;
    </USQL>
  </body>
</html>

Salida:
<html>
  <head>
  </head>
  <body>
    <h1>Reporte de Ejemplo</h1>
    <table>
      <tr>
        <th>Nombre</th>
        <th>fh_nac</th>
        <th>Trabaja</th>
      </tr>
      <tr>
        <td>Tomas Fernandez</td>
        <td>15-12-1990</td>
        <td>1</td>
      </tr>
      <tr>
        <td>Joaquin Alvarado</td>
        <td>22-2-1994</td>
        <td>0</td>
      </tr>
    </table>
  </body>
</html>
```

Este ejemplo se verá visualmente dese el IDE de la siguiente forma:



Como podemos observar la imagen anterior después de ejecutar el reporte se nos mostrara el código html resultado. Este código resultado al ser visto desde el navegador se visualizaría de la siguiente manera.



7 Manejo de errores

En los lenguajes se debe ser capaz de manejar errores y recuperarse de los mismos en cada fase de análisis de los lenguajes, para cada error se debe de indicar la línea, columna, descripción y tipo de error, los tipos de errores son los siguientes:

- Errores léxicos.
- Errores sintácticos.
- Errores semánticos.

7.1 Errores del lenguaje USQL

Para la recuperación de errores en el lenguaje USQL, el sistema debe de ser capaz de ejecutar todas las instrucciones correctas en la entrada, para las instrucciones que tengan un error de cualquier tipo, el sistema deberá de notificar el error en la consola de salida, especificando línea, columna, tipo, descripción. A continuación, se muestra un ejemplo del manejo de errores USQL.

```
1  CREAR BASE_DATOS Ejemplo1;
2
3  USAR Ejemplo1;
4
5  CREAR OBJETO Direccion(INTEGER avenida, INTEGER calle, TEXT nombre, TEXT descripcion);
6
7  CREAR TABLA Estudiante (
8  INTEGER id Llave_Primaria Autoincrementable,
9  TEXT Nombre No Nulo,
10 DATE fh_nac No Null
11 BOOL trabaja
12 Direccion direccion);
13
```

En la línea 10, se encuentra un error de tipo sintáctico (Null), así como en la línea 11 (falta coma). En este caso se deben de ejecutar con éxito las tres instrucciones anteriores a la que contiene el error y reportar todos aquellos que contiene errores. En el caso de que exista una instrucción sin errores luego de una con errores, ésta se debe de ejecutar con éxito.

Para el lenguaje USQL, se estable que la detección de errores se realizara por medio del método de recuperación en modo pánico. El cual consiste en que al momento de encontrar un error desechar símbolos de entrada hasta encontrar un símbolo de sincronización como podría ser el “;”. Si el analizador encuentra un error este de debe de entrar en modo pánico, continuar con el análisis del mismo hasta el máximo posible de la entrada.

7.2 Errores del lenguaje XML

Para el lenguaje XML deberá de existir un manejo de error de recuperación a nivel de fase. Este consiste en que, al momento de encontrar un error, el analizador sintáctico deberá de realizar una corrección local del resto de la entrada, esto se logra sustituyendo la cadena restante por otra cadena que permite que el análisis continúe. Por ejemplo, la sustitución de un carácter erróneo al final de la instrucción en lugar del punto y coma (;) podrá reemplazar dicho carácter o ignorar símbolos que no correspondan a una instrucción. A pesar de haberse recuperado de cualquier error es necesario que dicho error se notifique en la consola de salida del servidor de base de datos.

En caso de que el intérprete XML no pueda recuperarse, se deberá de notificar que el archivo ha sido corrupto, y mostrar la línea y columna en que se dejó de analizar el archivo. Para este caso es necesario mostrar el error en la consola de salida del servidor de base de datos y mandar un mensaje de error al usuario. El formato del paquete de error se encuentra en el numeral 4.4.2.

7.3 Errores PlyCS

Para el lenguaje PlyCS se debe de implementar una recuperación de errores en modo pánico para asegurarse que el mensaje se haya recibido correctamente. De encontrar un error en el mensaje debe de notificarse la línea, columna, tipo y descripción del error en la consola de salida del IDE.

8 Entregables y restricciones

8.1 Restricciones

- El IDE en el servidor web deberá de estar desarrollada con el lenguaje Visual Basic.
- Se deberá utilizar Irony para la generación de los analizadores del servidor de base de datos.
- Se deberá utilizar GoldParser para la generación de los analizadores en el servidor web.
- Como IDE sugerido para el desarrollo del servidor se propone Visual Studio 2013 o posterior.
- La comunicación entre el servidor web y el servidor de base de datos será por medio de sockets.
- Copias de proyectos tendrán de manera automática una nota de 0 puntos y serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas los involucrados.
- Todos los lenguajes implementados no hacen distinción entre mayúsculas y minúsculas, es decir no son case sensitive.

8.2 Requisitos mínimos

Los requerimientos mínimos del proyecto serán funcionalidades del sistema que permitirán un ciclo de ejecución básica, para tener derecho a calificación se deben cumplir con lo siguiente:

- USQL
  - DECLARAR
  - ASIGNACION
  - EXPRESIONES
  - INSTANCIA OBJETOS
  - CREAR
    - CREAR BASE_DATOS
    - CREAR TABLA
    - CREAR PROCEDIMIENTOS
    - CREAR FUNCIONES
    - CREAR USUARIO
  - INSERTAR
  - ACTUALIZAR
  - SELECCIONAR
  - CONDICIONES
  - SI
  - SELECCIONA
  - MIENTRAS
  - IMPRIMIR
  - BACKUP USQLDUMP
  - RESTAURAR USQLDUMP
- REPORTES

- Comunicación servidor web – servidor de base de datos:
  - Conexión directa y comprobable entre servidores mediante el uso de sockets.
  - Envío y recepción de los paquetes entre servidores.
- IDE
  - Autenticación de usuarios (login)
  - Visualización de resultados (Salida de datos, Mensajes, Historial)
  - Editor de texto
- Interfaz del servidor de bases de datos.
- Tener acumulado el 61% en promedio en las evaluaciones presenciales.

8.3 Entregables

- Aplicaciones funcionales: servidor web y servidor de base de datos
- Código fuente
- Gramáticas (Irony y GoldParser)
- Manual Técnico
- Manual de Usuario

Deberán entregarse todos los archivos necesarios para la ejecución de la aplicación, así como el código fuente, la gramática y la documentación. Tomar en cuenta que la calificación del proyecto se realizará con los archivos entregados en la fecha establecida. El estudiante es completa y únicamente responsable de verificar el contenido de los entregables. La forma de entrega será virtual, los detalles de la entrega se informarán días antes de la fecha de entrega. La calificación se realizará sobre archivos ejecutables. Se proporcionarán archivos de entrada al momento de calificación.

8.4 Fecha de Entrega:

19 de diciembre antes de las 23:59

Crédito y agradecimiento a los siguientes tutores académicos que redactaron el enunciado para el segundo semestre de 2017, ya que en base al mismo fue redactado este enunciado:

Juan Ruiz, Elmer Calel y Henry Taracena