



MANUAL DE USUARIO

ORGANIZACIÓN DE LENGUAJES Y COMPILADORES 2



Carnet	Nombre
201503712	Oscar Rene Cuéllar Mancilla

29 DE SEPTIEMBRE DE 2019

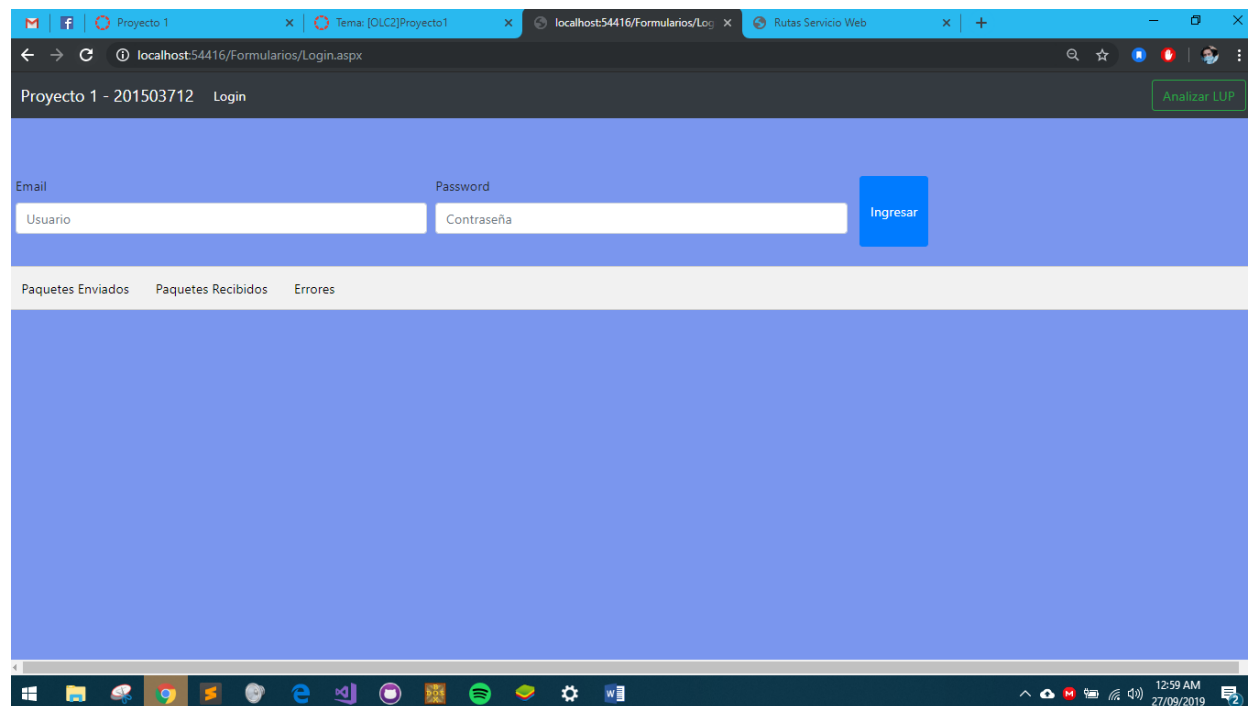
Contenido

1. Pasos para entrar a los modos de CQL-Client	2
1.1 LOGIN.....	2
2. Pasos para armar una entrada a partir de bloques.....	4
3. Instrucciones para Modo Principiante	6
3.2 Sentencias.....	6
3.2 Atributos	7
3.3 Operadores.....	7
3.4 Valores.....	7
4. Instrucciones para Modo Intermedio	8
4.1 Variables	8
4.2 Sentencias de Control	9
4.3 Ciclos	9
4.4 Operadores.....	10
4.5 Valores.....	10
4.6 Procedimientos	11
4. Instrucciones para modo Avanzado	12
4.2 Cargar un archivo CQL.....	12
4.2 Ejecutar Archivo CQL	12
4.3 Ejecutar Respuesta LUP	13
4.4 Visualización de Resultados	13
5. Flujo de la Aplicación	14
5.1 Cliente	14
5.2 Comunicación del Cliente al Servidor	15
5.3 Comunicación Servidor al Cliente.....	16
5.4 Cerrar Sesión.....	17

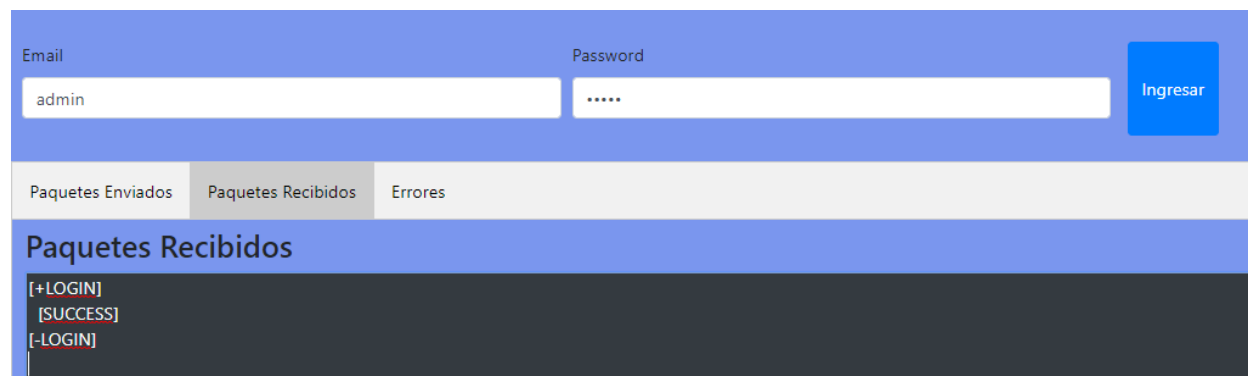
1. Pasos para entrar a los modos de CQL-Client

1.1 LOGIN

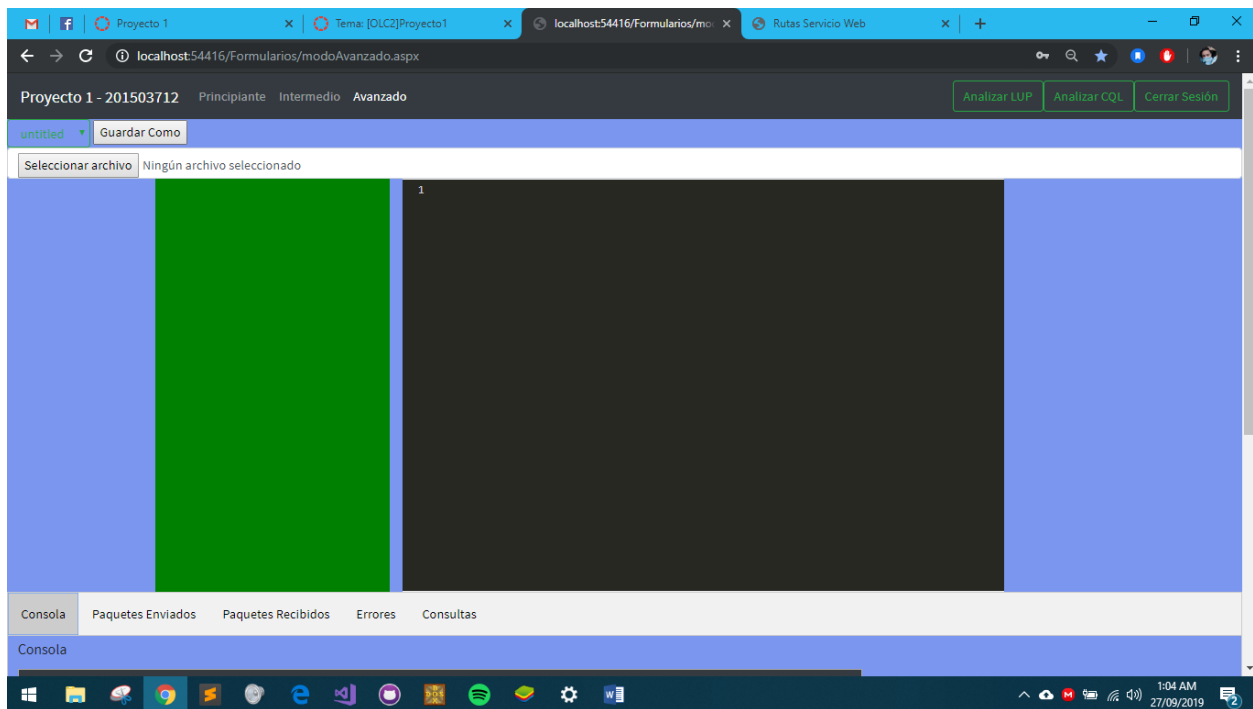
Una vez cargado el cliente podremos ver como página principal una interfaz de Login, en ella se solicita los campos de Usuario y Contraseña para poder entrar al sistema.



Estos datos son enviados al servidor y posteriormente el servidor nos responde con un mensaje de aceptación o rechazo como se muestra aquí:

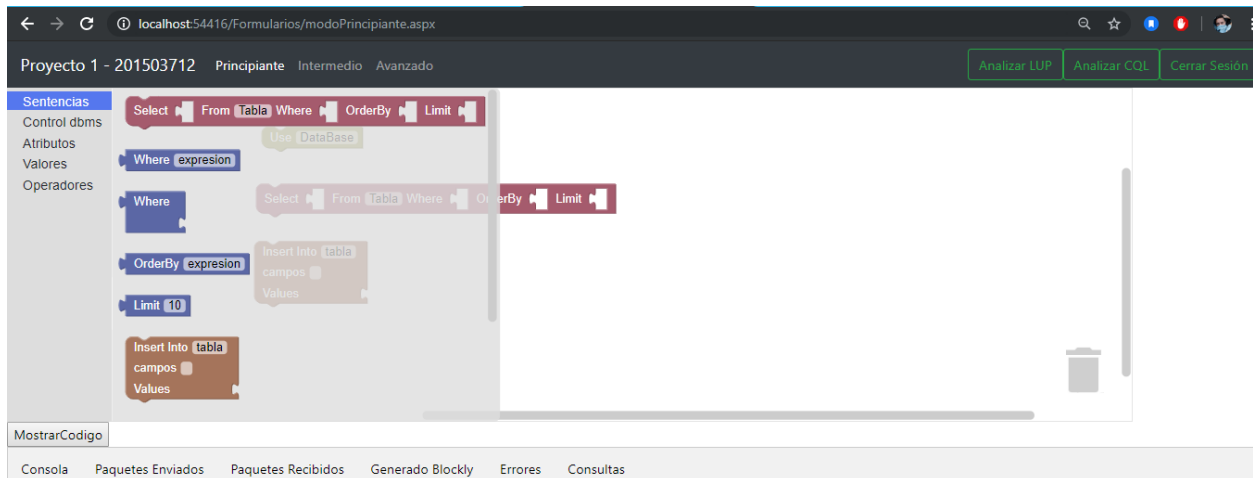


Como podemos observar al iniciar sesión con “Admin”-“Admin” que es el usuario por defecto de la base de datos, nos responde con un mensaje de SUCCESS, seguidamente deberemos presionar el botón, “ANALIZAR LUP” el cual analizará dicha cadena y decidirá si dejarnos o no entrar al sistema.



Una vez el sistema nos haya concedido el acceso, tendremos a nuestra disponibilidad todas las funcionalidades de CQ-Client, como lo son, los modos principiante, intermedio y avanzado.

2. Pasos para armar una entrada a partir de bloques



Para poder armar una entrada a partir de bloques deberemos seguir los siguientes pasos:

1. Entrar a alguno de los modos de edición de bloques (principiante o intermedio).
2. Tener en cuenta la consulta que queramos realizar, en este caso, un use, un select y un insert.
3. Buscar los bloques en sus respectivas categorías e irlos colocando en el orden que queramos evaluar dichas instrucciones.



Los bloques los podremos ir uniendo entre sí para tener un mejor control sobre el orden en que se evaluarán dichas instrucciones.

4. Cada uno de los bloques recibe un determinado tipo de bloques, como lo es el ejemplo del bloque select, en el cual su sentencia where, espera nada más un bloque de tipo where y la misma interfaz no nos dejará colocar un bloque de otro tipo que no sea where.

5. Luego de tener los bloques necesarios y en el orden necesario se procederá a presionar el botón de mostrar código, el cual nos mostrará el código generado por dichos bloques para los usuarios más experimentados y que puedan decidir si mandar a ejecutar dichas instrucciones o no.

The screenshot shows a Blockly workspace with a sidebar on the left containing categories: Sentencias, Control dbms, Atributos, Valores, and Operadores. The workspace contains the following blocks:

- Use DataBase** (green block)
- Select** block with sub-blocks: **Seleccion ***, **From Tabla**, **Where**, **OrderBy**, and **Limit**.
- Insert Into tabla** block with sub-blocks: **campos** and **Values**.
- crear lista con** block (purple) with three **String string** blocks (green) as inputs.

Below the workspace is a **MostrarCodigo** button. At the bottom, there is a tabbed interface with the following tabs: **Consola**, **Paquetes Enviados**, **Paquetes Recibidos**, **Generado Blockly** (active), **Errores**, and **Consultas**.

The **Generado Blockly** tab displays the following SQL code:

```
USE DataBase;
SELECT * FROM Tabla;
INSERT INTO tabla VALUES (("string"), ("string"), ("string"));
```

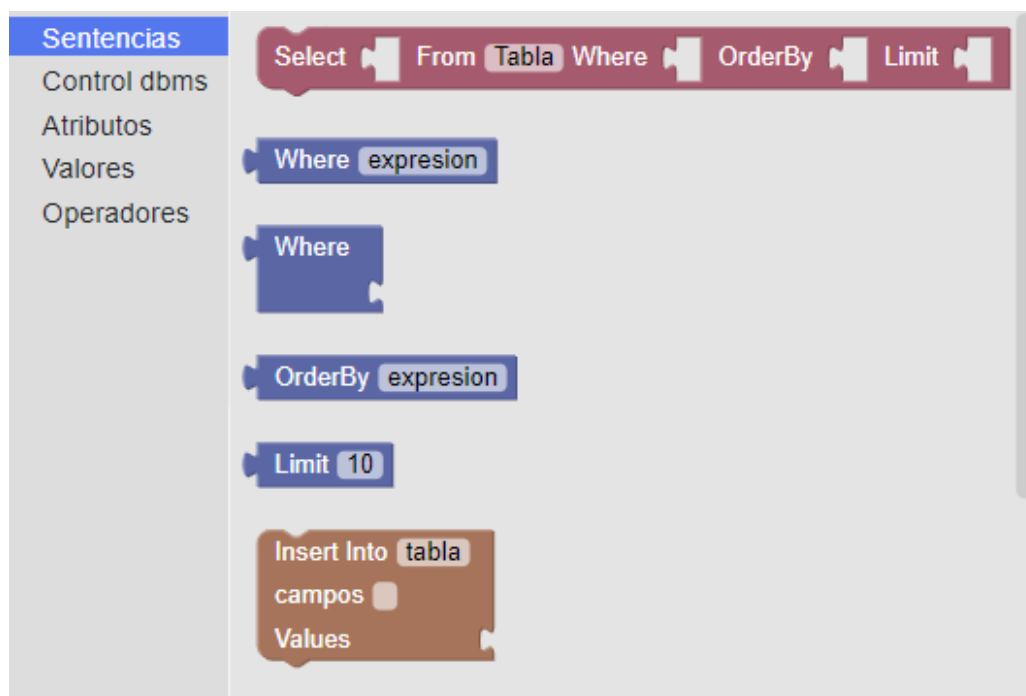
Podremos observar en el área de trabajo de “Generado Blockly” el código generado por los bloques que colocamos en la interfaz.

3. Instrucciones para Modo Principiante

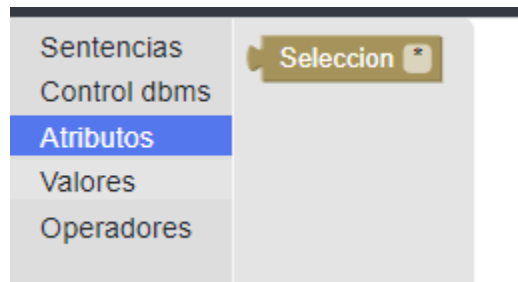
Este modo de ejecución de sentencias está limitado a ciertas instrucciones, entre las cuales tenemos:

- Sentencias. Contiene los bloques para las instrucciones Select, Insert, Update y Delete
- Atributos. Contiene bloques con listas para poder seleccionar nombre de Tablas y las columnas. Cuenta con el comodín "*" que permite seleccionar todos los campos para la instrucción Select
- Operadores. Contiene los bloques para poder agregar comparaciones booleanas, expresiones aritméticas y expresiones lógicas
- Valores. Contiene los bloques para poder agregar valores tales como enteros, decimales, valores booleanos, cadenas, fechas. También contiene el bloque para poder agregar lista de valores

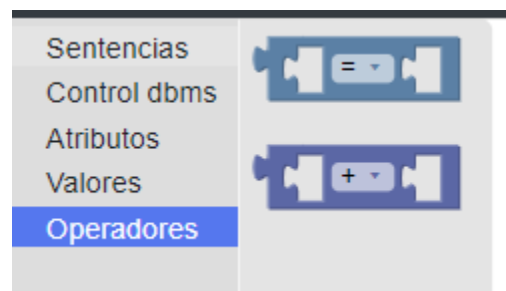
3.2 Sentencias



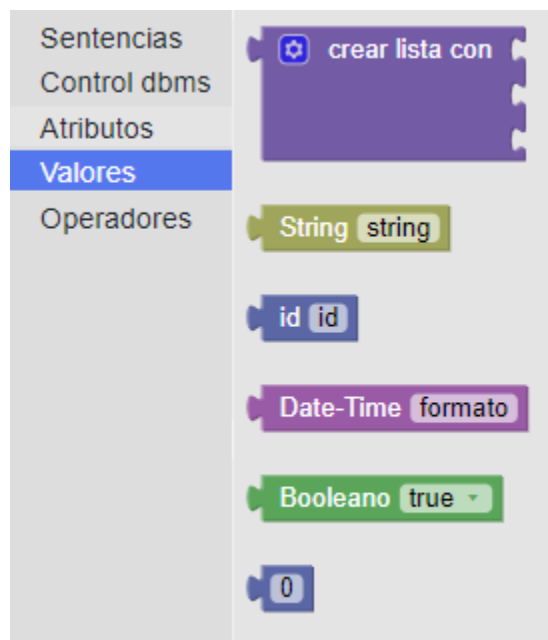
3.2 Atributos



3.3 Operadores



3.4 Valores

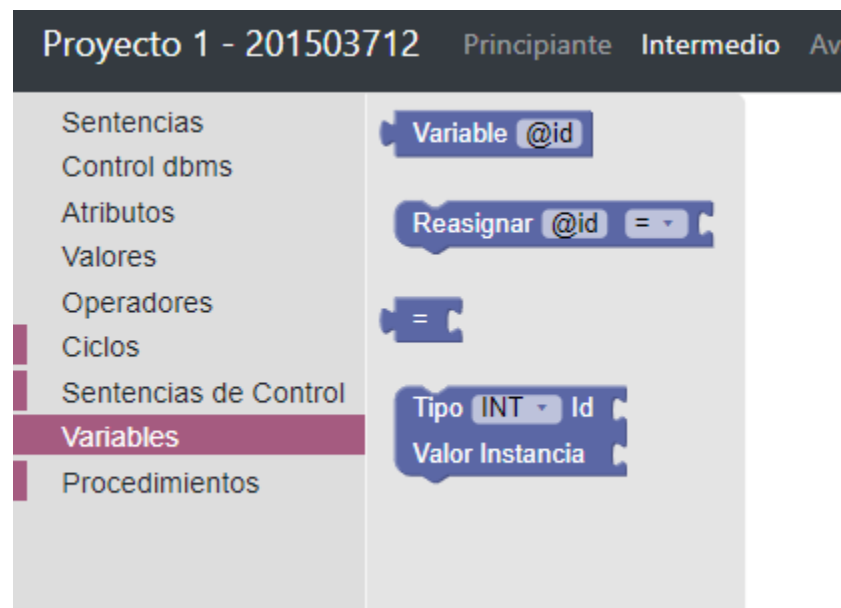


4. Instrucciones para Modo Intermedio

En este modo tenemos todas las instrucciones y sentencias del modo principiante y agregando como nuevas ciertas instrucciones que se describen a continuación:

- Variables. Permite la creación de variables con los tipos válidos de CQL (solo primitivos); también permite la asignación de variables con expresiones
- Sentencias de control. Contiene los bloques para poder agregar sentencias If / Else y Switch
- Ciclos. Contiene los bloques para poder agregar sentencias While, Do-While y For.
- Operadores. Contiene los bloques para poder agregar comparaciones booleanas, expresiones aritméticas, expresiones lógicas y llamadas a funciones.
- Valores. Contiene los bloques para poder agregar valores tales como enteros, decimales, booleanos, cadenas, fechas.
- Procedimientos. Contiene los bloques para poder llamar a procedimientos primitivos como LOG y las definidas por el usuario.

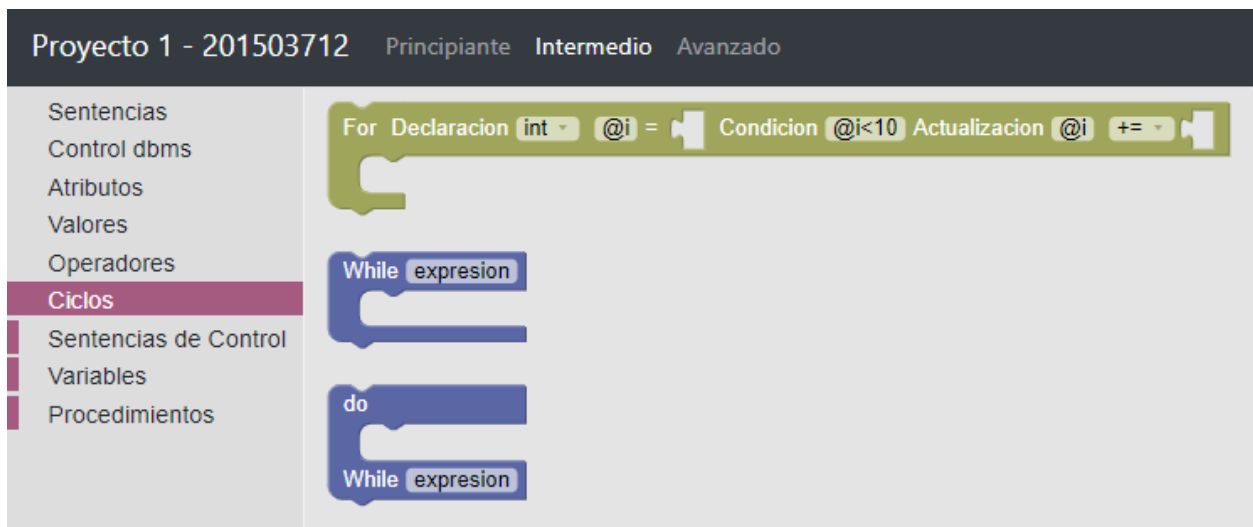
4.1 Variables



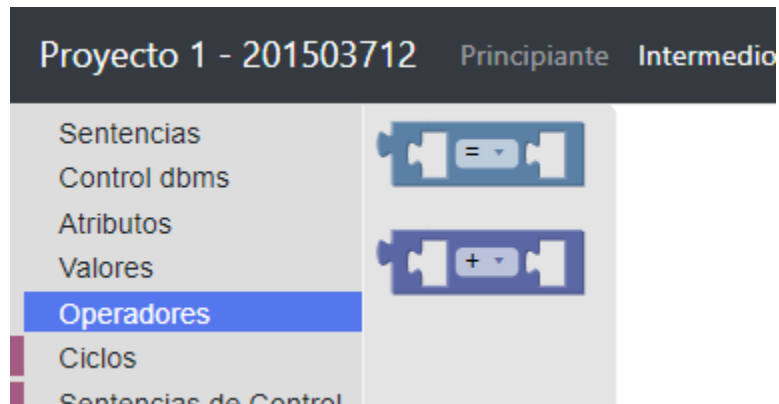
4.2 Sentencias de Control



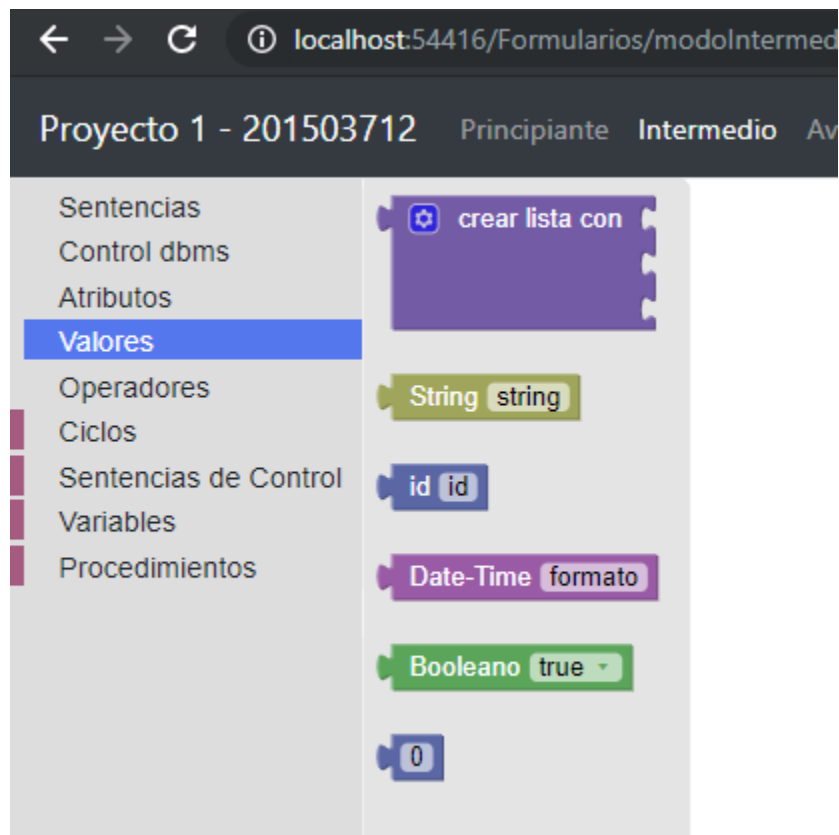
4.3 Ciclos



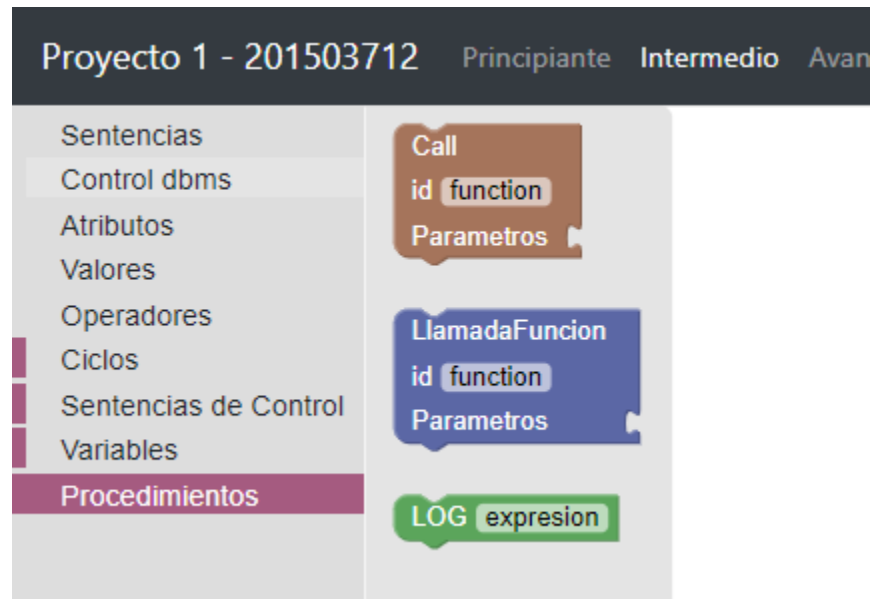
4.4 Operadores



4.5 Valores



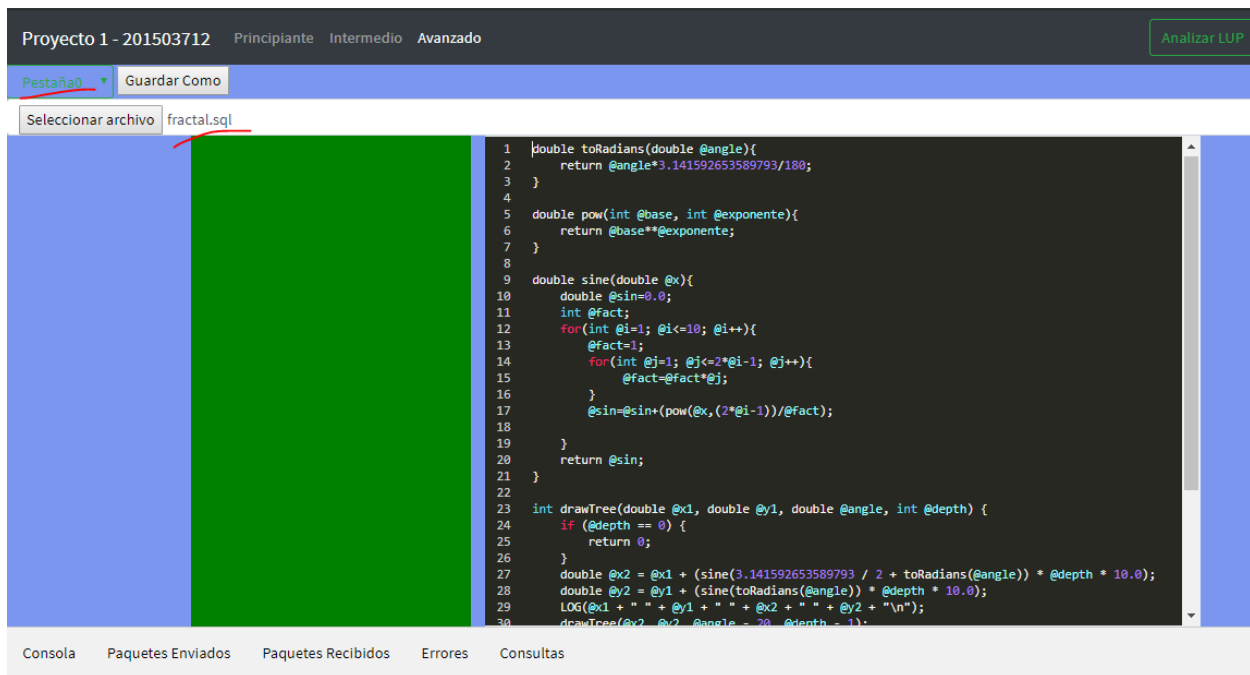
4.6 Procedimientos



4. Instrucciones para modo Avanzado

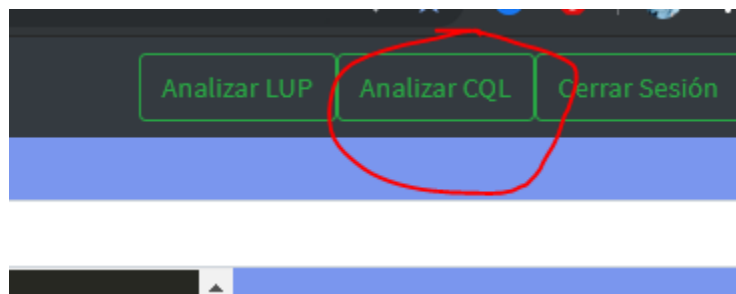
4.2 Cargar un archivo CQL

Seleccionamos el botón de “Seleccionar un archivo” el cual nos permitirá seleccionar un archivo alojado en nuestro ordenador, el cual podremos visualizar en el editor de texto de este modo de edición.



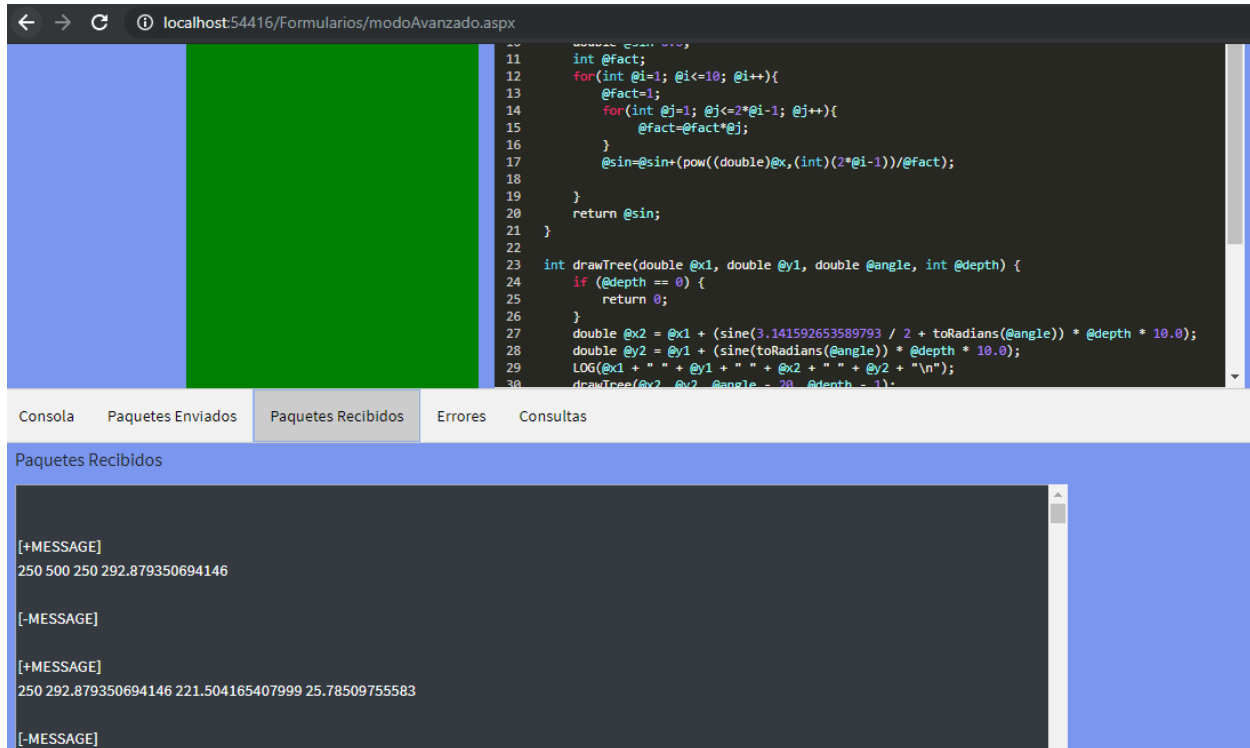
4.2 Ejecutar Archivo CQL

Posteriormente podremos analizar dicho archivo cql con el botón de Analizar CQL



4.3 Ejecutar Respuesta LUP

El servidor nos retornará una respuesta empaquetada en lenguaje LUP, esta deberemos analizarla con el botón “Analizar LUP”.



The screenshot shows a web application interface. At the top, there's a browser address bar showing 'localhost:54416/Formularios/modoAvanzado.aspx'. Below it is a code editor with a dark background and light-colored text. The code is in C# and defines a function 'drawTree' that takes parameters for coordinates, angle, and depth. The code is as follows:

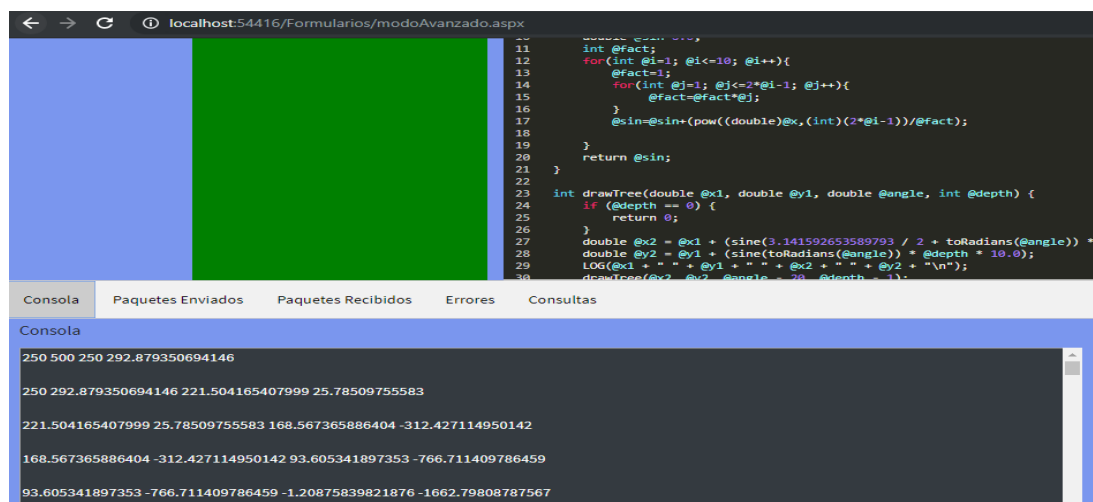
```
11 int @fact;  
12 for(int @i=1; @i<=10; @i++){  
13     @fact=1;  
14     for(int @j=1; @j<=2*@i-1; @j++){  
15         @fact=@fact*@j;  
16     }  
17     @sin=@sin+(pow((double)@x,(int)(2*@i-1))/@fact);  
18 }  
19 }  
20 return @sin;  
21 }  
22  
23 int drawTree(double @x1, double @y1, double @angle, int @depth) {  
24     if (@depth == 0) {  
25         return 0;  
26     }  
27     double @x2 = @x1 + (sine(3.141592653589793 / 2 + toRadians(@angle)) * @depth * 10.0);  
28     double @y2 = @y1 + (sine(toRadians(@angle)) * @depth * 10.0);  
29     LOG(@x1 + " " + @y1 + " " + @x2 + " " + @y2 + "\n");  
30     drawTree(@x2, @y2, @angle + 20, @depth - 1);  
31 }
```

Below the code editor is a tabbed interface with four tabs: 'Consola', 'Paquetes Enviados', 'Paquetes Recibidos', and 'Errores'. The 'Paquetes Recibidos' tab is selected, showing a list of received packages in a dark gray box. The packages are as follows:

```
[+MESSAGE]  
250 500 250 292.879350694146  
[-MESSAGE]  
[+MESSAGE]  
250 292.879350694146 221.504165407999 25.78509755583  
[-MESSAGE]
```

4.4 Visualización de Resultados

Luego de ser analizada podremos ver los resultados en la pestaña de consola.



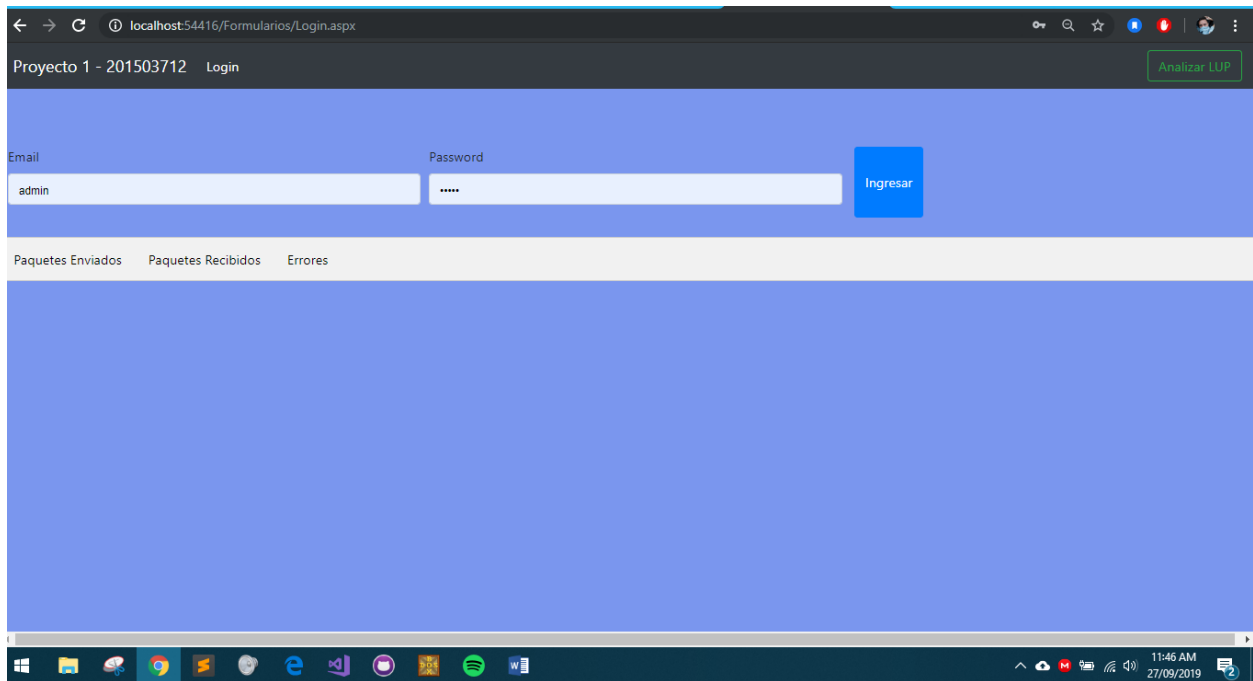
The screenshot shows the same web application interface, but with the 'Consola' tab selected. The console output is displayed in a dark gray box. The output is as follows:

```
250 500 250 292.879350694146  
250 292.879350694146 221.504165407999 25.78509755583  
221.504165407999 25.78509755583 168.567365886404 -312.427114950142  
168.567365886404 -312.427114950142 93.605341897353 -766.711409786459  
93.605341897353 -766.711409786459 -1.20875839821876 -1662.79808787567
```

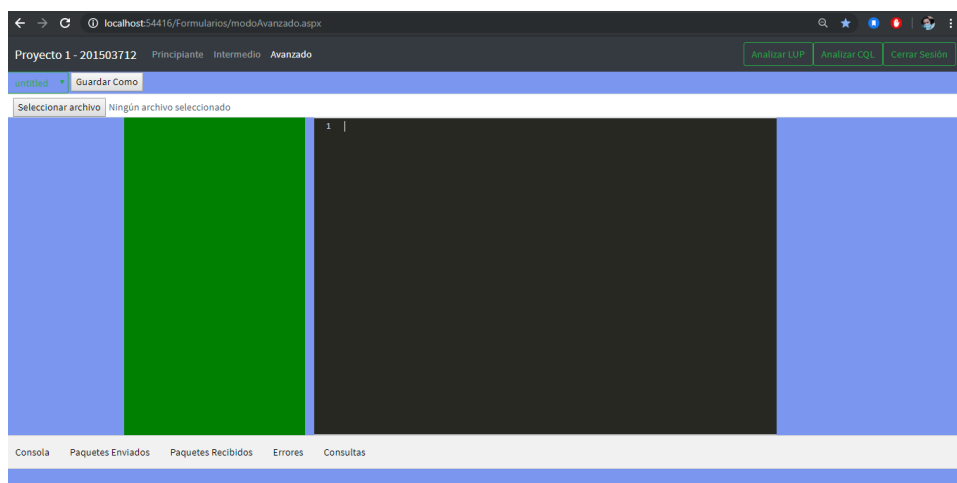
5. Flujo de la Aplicación

5.1 Cliente

La solución comienza con la vista del cliente al usuario



En él podemos hacer cualquier operación de base de datos CQL, utilizando cualquiera de los modos, principiante, intermedio y avanzado.



5.2 Comunicación del Cliente al Servidor

Una vez tengamos establecidas las instrucciones a ejecutar de CQL, podremos proceder a enviarlas al servidor de base de datos, para lo cual se empaquetan dichas instrucciones en un lenguaje de paquetes llamado LUP. Como se muestra a continuación:

The screenshot shows a web browser window at `localhost:54416/Formularios/modoAvanzado.aspx`. The interface has a top bar with a search icon, a star, and a user profile icon. Below the browser window, there's a tabbed interface with four tabs: "Consola", "Paquetes Enviados", "Paquetes Recibidos", and "Errores". The "Paquetes Enviados" tab is selected, showing a list of sent packages. The package content is displayed in a dark-themed text area. The package starts with a header `[+QUERY] [+USER] admin[-USER][+DATA]/` followed by a series of SQL queries and LUP commands. The SQL queries include an `UPDATE` statement to set `order_status` to `@ORDER_STATUS_DELIVERED`, a `WHERE` clause, a `SELECT` statement, and a `commit` statement. The LUP commands include a `LOG` statement, a `CREATE DATABASE` statement, and a `commit` statement. The package ends with a `CREATE DATABASE IF NOT EXISTS` statement.

```
487 UPDATE orders;
488 SET order_status = @ORDER_STATUS_DELIVERED;
489 WHERE order_status != @ORDER_STATUS_DELIVERED;
490
491
492
493
494 SELECT * FROM orders;
495
496
497 commit;
```

Consola Paquetes Enviados Paquetes Recibidos Errores Consultas

Paquetes Enviados

```
[+QUERY] [+USER] admin[-USER][+DATA]/
.....
+ OLC2 Proyecto 1
+ Validación de mínimos
.....
*/
int @carnet = 201503712;
String @nombre = "Oscar Cuellar";

LOG("t** EJECUTADO SCRIPT \"MINIMO\" **\n\t** CARNET: " + @carnet + " **\n\t** NOMBRE: " + @nombre + " **");

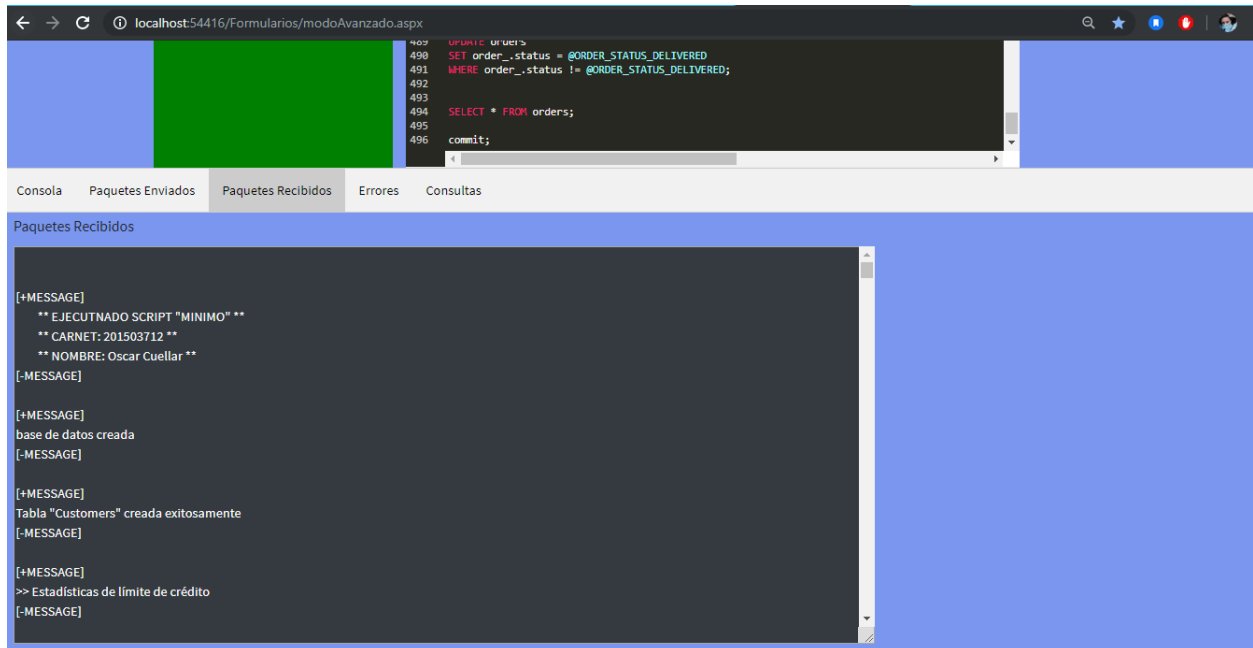
CREATE DATABASE _01_database; // base de datos para mínimos

LOG("base de datos creada");

commit; // debe crear la base de datos en los archivos *.chison

CREATE DATABASE IF NOT EXISTS _01_dAtAbAsE; // no debe dar error
```

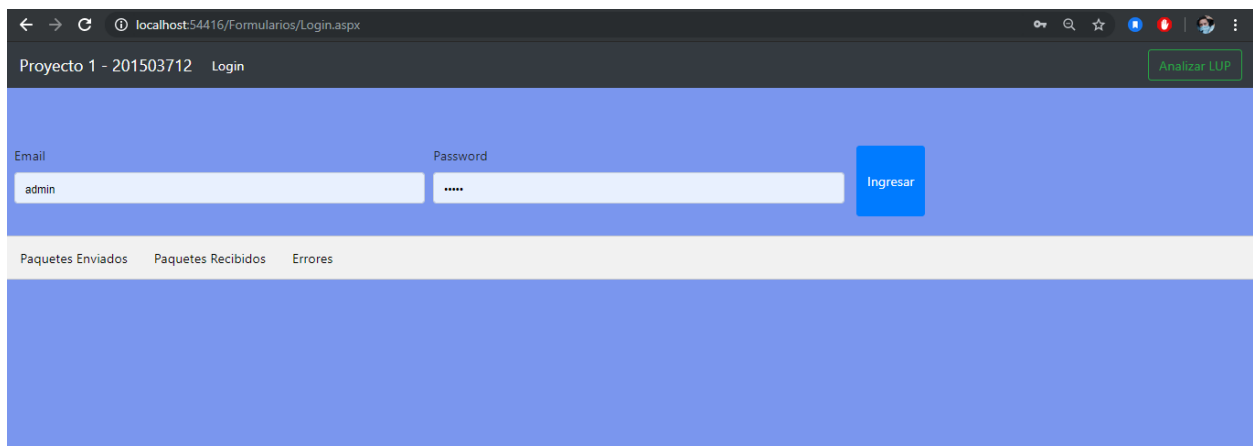
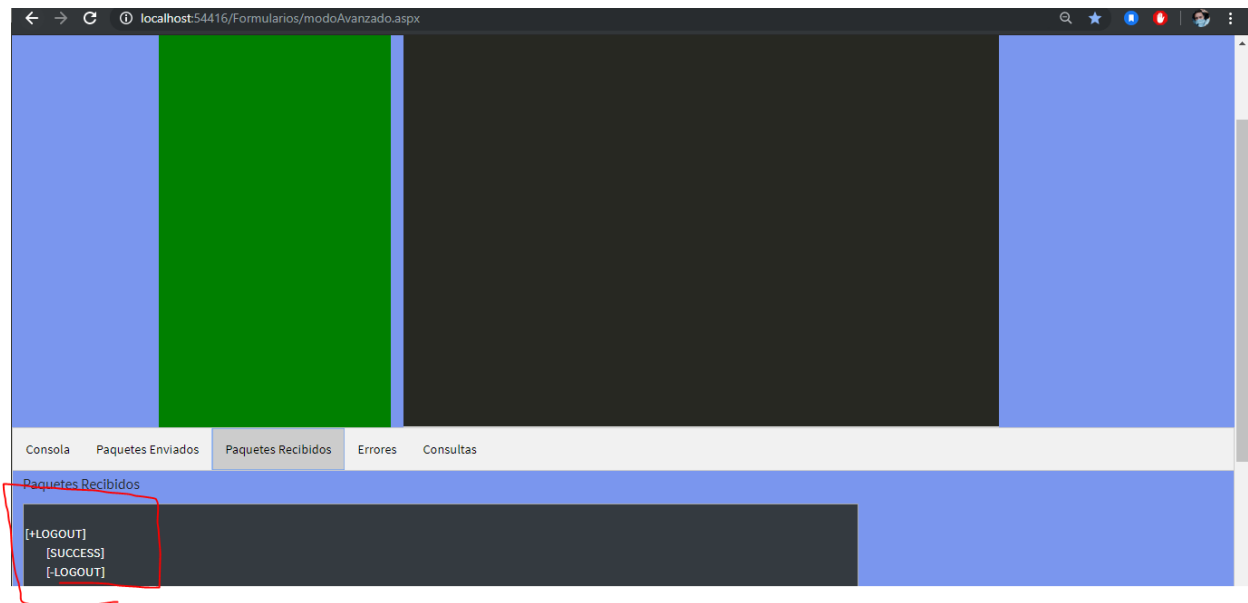

5.3 Comunicación Servidor al Cliente



Se puede observar en la pestaña “Paquetes Recibidos”, el empaquetado recibido en lenguaje LUP desde el servidor. Una vez analizado este paquete podremos observar los resultados en los diferentes componentes HTML, como el TreeView que muestra la información en la DBMS y las consolas que muestran los mensajes impresos, la tabla de errores y tablas de consultas.

5.4 Cerrar Sesión

Para cerrar sesión, deberemos hacer click en el botón Cerrar Sesión, el cual enviará una solicitud de Logout al Servidor el cual nos dirá si es posible realizarla o no.



Nuevamente nos regresará a la ventana de inicio.