Yellow Push

By: 4DEV

Manual de Usuario

Designer Studio
& object builder modules
Yellow Push

Programe gráficamente en menos tiempo y con menos código uniendo cajas que a su vez contienen código seguro y probado, garantizando confianza y un mejor desempeño con código abierto para desarrollar aplicaciones web nativas y aplicaciones móviles increibles.

IDENTIDAD TELECOM

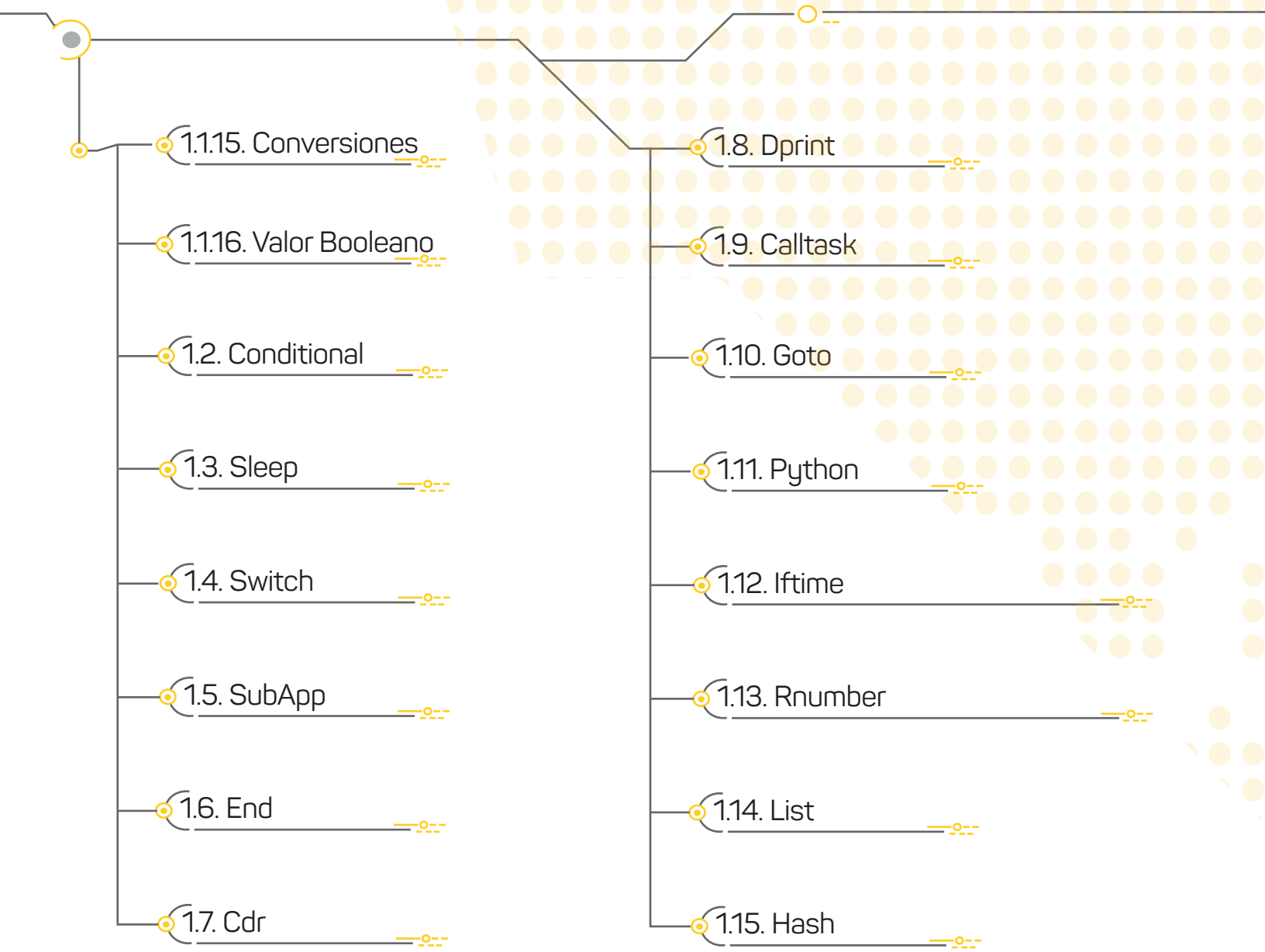
#YellowPushSolutions

Tel: 1 (786) 242-2224 Email: smsretail@identidadtelecom.net

www.identidadtelecom.net

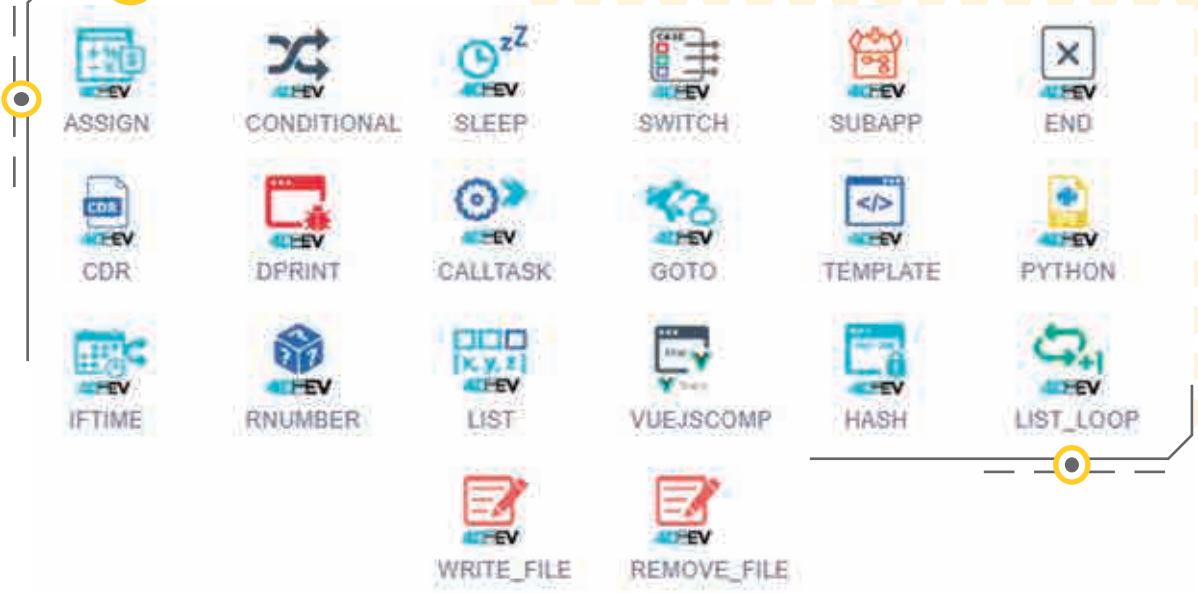
Identidad Telecom an **Identidad Technologies** company

Tabla de Contenido


1.1.15. Conversiones
1.1.16. Valor Booleano
1.2. Conditional
1.3. Sleep
1.4. Switch
1.5. SubApp
1.6. End
1.7. Cdr
1.8. Dprint
1.9. Calltask
1.10. Goto
1.11. Python
1.12. Iftime
1.13. Rnumber
1.14. List
1.15. Hash

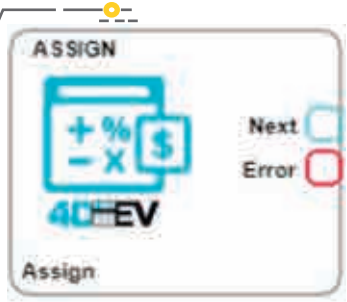
- 1.16. List_Loop
- 2. Modulo Base de Datos
  - 2.1. Conexión Mongo
  - 2.2. Ejecución Mongo
  - 2.3. Conexión MySql
  - 2.4. Ejecución MySql
  - 2.5. Conexión PostgreSQL
  - 2.6. Ejecución PostgreSQL

Comandos Básicos



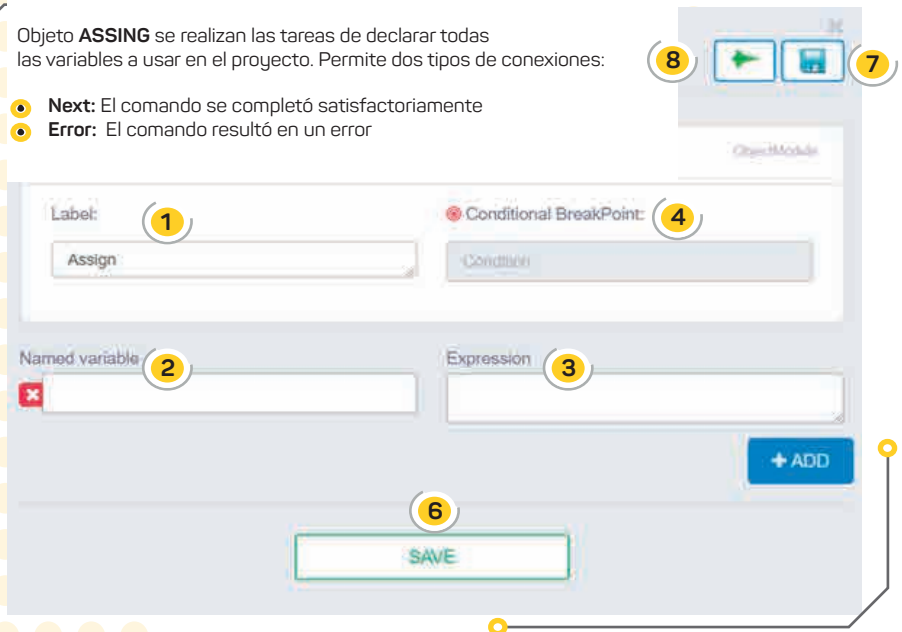
Esta sección describe la barra de herramientas y sus respectivas propiedades:

1.1 ASSING



Objeto **ASSING** se realizan las tareas de declarar todas las variables a usar en el proyecto. Permite dos tipos de conexiones:

- **Next:** El comando se completó satisfactoriamente
- **Error:** El comando resultó en un error



Numero	Campo	Definición
1	**Label**	Nombre del objeto **Assing.**
2	**Named Variable**	Nombre de la variable en la que se almacena la expresión. Las variables nombradas deben comenzar por $
3	**Expression**	El valor de la variable. El valor puede ser una variable, un string de variables, una expresión. Si es un valor string este debe ser encerrado en comillas (" "). Si el valor es un integer este no usa comillas.
4	**Conditional break point**	Campo para realizar interrupción (romper) por medio de un condicional en el modo debug
5	**+ADD**	Botón agregar más variables
6	**SAVE**	Botón Guardar cambios aplicados en este objeto.
7	**Guardar**	Botón Guardar cambios globales
8	**Deploy**	Botón aplicar cambios en el proyecto

1.1.1 Expresiones

A continuación, la sintaxis para todas las expresiones y una lista de todas las funciones que se pueden utilizar .

Estas son algunas reglas básicas:

- El nombre de las variables debe comenzar con el símbolo "$ ": ($xxx).
- El nombre de las variables pueden contener un número o puede contener una cadena.

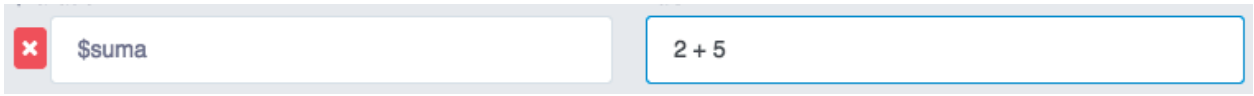
1.1.2 Operadores Aritméticos

A continuación, los operadores aritméticos con los que podemos trabajar y manipular las variables cuyo contenido es numérico, es decir, las del tipo **"int", "long" y "float".**

Se explica a continuación los ocho operadores aritméticos diferentes con los que podremos realizar operaciones básicas de forma sencilla.

1.1.3 Sumar

La forma más sencilla de realizar una suma es introducir los valores directamente para que el resultado de la misma sea el valor asignado a la variable como se muestra a continuación.




```
$ Suma = 2 + 5
```

1.1.3 Sumar

El resultado de la variable "suma" será 7. También es posible realizar sumas cuyo resultado sea variable en función de nuestras necesidades, por lo que también pueden realizarse del siguiente modo donde la suma se realizará en función del valor que las variables que puede ser modificado de forma sencilla.

$primerNumero	4
$segundoNumero	2
$suma	$primerNumero + $segundoNumero

$primerNumero = 4

$segundoNumero = 2

$suma = $primerNumero + $segundoNumero

1.1.4 Ley de signos

Se emplea cuando encontramos dos signos consecutivos ya sean positivos y/o negativos, se emplea la ley de los signos:

Tabla ley de signos		
+	+	+
+	-	-
-	-	+
-	+	-

Ejemplo:1

$numero	-4
$resultado	– $numero

```
$numero = -4

$resultado = – $numero

$resultado = 4
```

Se declara la variable $numero = -4

Se declara la segunda variable $resultado = --4

Esto es equivalente a $resultado = 4

Al realizar la operación de resultado se puede observar el valor 4 realizando la ley de los

signos (-)(-) = (+)

1.1.5 Restar

De la misma forma que se puede realizar la suma, para realizar la resta utilizamos el operador resta (-).

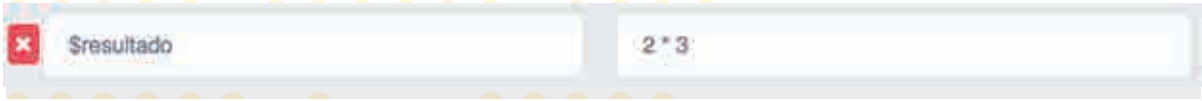
Un sencillo ejemplo podría ser calcular la diferencia entre los números 3 y 1:


```
$resta = 3 – 1
```

El resultado, y por lo tanto el valor de la variable "resta" será 2. Pero hay que tener en cuenta lo mismo que se explicó para el operador Suma, ya que realizar una resta o calcular con los operadores que se muestran a continuación resulta poco útil, lo que puede solucionarse con la creación de variables y operar con ellas como se explicó para el parámetro Suma.

1.1.6 Multiplicación

Para realizar una multiplicación utilizamos el operador asterisco (*) que puede ser utilizado como se muestra siendo el valor de la variable "resultado" 6.




```
$resultado = 2 * 3
```

1.1.7 División

Para la realización del cociente entre dos números en 4Dev hay que utilizar el operador (/) como se muestra a continuación, donde el resultado será un número real.


```
$division1 = 6 / 2

$division2 = 5.5 / 2

$division3 = 5 / 2

$division4 = 5.0 / 2
```

Por tanto, la variable llamada "$division1" será del tipo "int" con valor 3 y el de la variable "$division2" será 2.75 del tipo "float". Respecto a la variable "$division3" cabría esperar que sería del tipo "float" y con valor 2.5, pero en cambio, al calcularse partiendo de dos número de tipo "int", el resultado es del mismo tipo y su valor es 2. Para obtener un resultado del tipo "float" y obtener el resultado real tras el cálculo es necesario realizarlo de la misma forma que en "$division4" haciendo que al menos uno de los números sea del tipo "float".

1.1.8 División Entera

Existe una variedad de división donde el resultado no devolverá la parte decimal correspondiente al resultado, sino que solo devolverá la parte entera. Para realizar éste tipo de divisiones utilizamos el operador (//) doble barra como se ve a continuación.


```
$division = 5.0 // 2
```

De éste modo, solo obtenemos la parte entera tras la división, en éste caso 2.0. Pero el tipo de dato resultante tras ella es "float".

1.1.9 Exponentes

La forma de realizar exponentes con 4dev es sencilla gracias al operador doble çasterisco (**) que se emplea del siguiente modo.


```
$exponente = 2 ** 3
```

En este caso la variable "exponente" tendrá un valor de 8.

1.1.10. Módulos

La forma de calcular el módulo, el cual nos devuelve el resto de una división.
Para calcular el módulo utilizamos el operador (%) como se muestra a continuación.


```
$modulo = 5 % 2
```

$modulo = 5 /2

$modulo = 1

5 ÷ 2 = 2.5

```
 5 |2
 4 |2.5
 10
```

1.1.11 Funciones Aritméticas

Estas funciones realizan varias operaciones aritméticas como calcular el valor superior, inferior o absoluto de un número usando las funciones math.

$ceil_val = 2

La función math.floor devuelve el entero más pequeño

La función factorial Dice que se debe multiplicar una serie de números naturales descendentes. Ejemplos:

- 4 = 4 × 3 × 2 × 1 = 24
- 10 = 10 × 9 × 8 × 7 × 6 × 5 × 4× 3× 2× 1 = 3628800

$factorial_val = 3628800

1.1.12 Funciones Trigonométricas

Estas funciones relacionan los ángulos de un triángulo a sus lados. Con 4dev puedes calcular math.sin(x), math.cos(x), y math.tan(x) directamente.

math.sin (x) es igual al seno de x

$sin_val = 0.7071067811865476

math.cos (x) es igual al coseno de x

$cos_val = -1.0

math.tan (x) es igual al tangente de x

$tan_val = 0.5773502691896257

1.1.13 Listas

Las listas son conjuntos ordenados de elementos (números, cadenas, listas, etc).
Las listas se delimitan por corchetes ([ ]) y los elementos se separan por comas.
Las listas pueden contener elementos del mismo tipo:

```
$simple_list    [2, 3, 5, 7, 11, 13]
```

Pueden contener elementos de tipos distintos:

```
$mix_list    ["Lunes", 27, "Octubre", 1997]
```

Pueden contener listas Las listas pueden tener muchos niveles de anidamiento:

```
$list_list    [["Senderos", 1957], ["Hannah", 1986]]
```

Las operaciones más habituales que se realizan son las siguientes:

- lista[i]: Devuelve el elemento que está en la posición i de la lista.
- lista.pop(i): Devuelve el elemento en la posición i de una lista y luego lo borra.
- ista.append(elemento): Añade elemento al final de la lista.
- lista.insert(i, elemento): Inserta elemento en la posición i.
- lista.extend(lista2): Fusiona lista con lista2.
- lista.remove(elemento): Elimina la primera vez que aparece elemento.

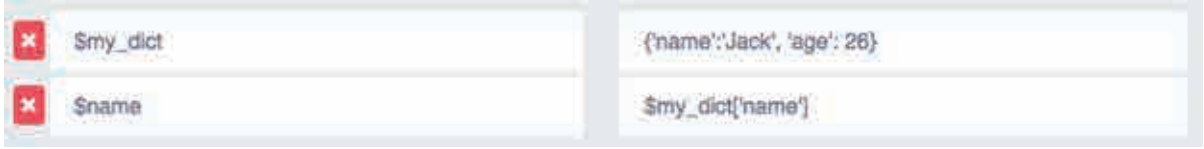
1.1.14 Diccionario

Los diccionarios son un tipo de estructuras de datos que permite guardar un conjunto no ordenado de pares clave-valor, siendo las claves únicas dentro de un mismo diccionario (es decir que no pueden existir dos elementos con una misma clave).

$dic_simple = {'a':'one', 'b':'two' }

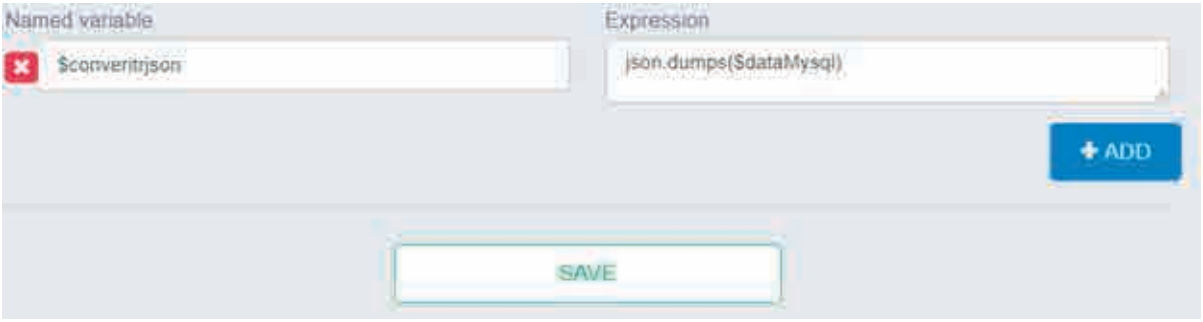
Pueden contener elementos de tipos distintos:

Como acceder a los elementos de un diccionario:



1.1.15 Conversiones

La conversión de Tipo de datos para especificar explícitamente el Tipo de Dato con el que deseamos trabajar



En la Variable $converitrjson guardamos la conversión de la variable json. dumps ($dataMySQL) en la cual la variable $dataMysql esta con el tipo de dato diccionario y se desea convertir la variable a json para operar.

- Se declara la variable $variablestr con el valor "5" definida como un string por las comillas dobles.
- Se declara la variable $Convertentero con la función de conversión int($variablestring)
- El valor final de la conversión es 5 definido como entero

- Se declara la variable $variableint con el valor 44 definida como un entero.
- Se declara la variable $ConvertString con la función de conversión str($variableint)
- El valor final de la conversión es 445 definido como String

- Se declara la variable $Dicionariocon con el valor {nombre:"pedro",edad:"45"} definida como un diccionario
- Se declara la variable $JsonConvert con la función de conversión json.loads($Dicionario)
- El valor final de la conversión es{nombre: “pedro”, edad:"45"} definido como Json

- Se declara la variable $json con el valor {nombre:"pedro",edad:"45"} definida como un json
- Se declara la variable $DicccionarioConvert con la función de conversión json.loads($json)
- El valor final de la conversión es{nombre: “pedro”, edad:"45"} definido como Diccionario

1.1.16 Valor Booleano

permite guardar un valor para representar los valores de verdadero (True) y falso (False) respectivamente.

Declaramos la variable $send_rsp asignamos el valor de false en el campo expresión con la primera letra en mayúscula.

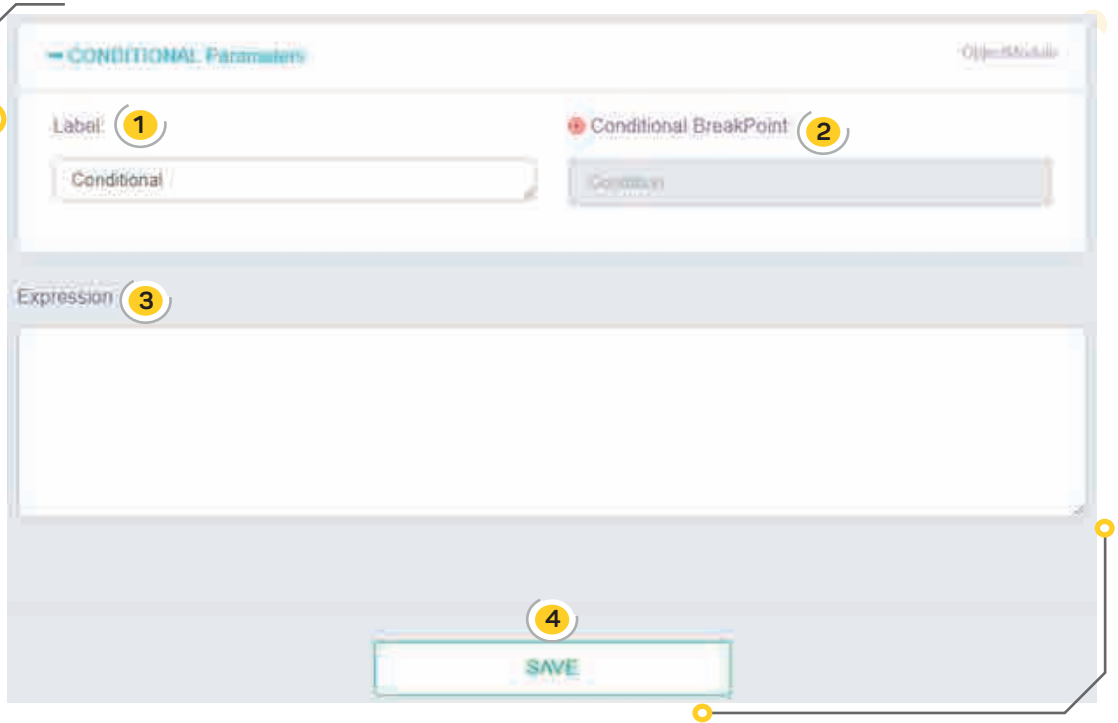
Declaramos la variable $send_rsp asignamos el valor de true en el campo expresión con la primera letra en mayúscula.

1.2 CONDITIONAL

Objeto Conditional Sentencia o grupo de sentencias que puede ejecutarse o no en función del valor de una condición. Permite tres tipos de conexiones.

- True: La expresión es verdadera
- False: La expresión es falsa
- Error: El comando resultó en un error.

Presionando doble click se despliega el siguiente menú



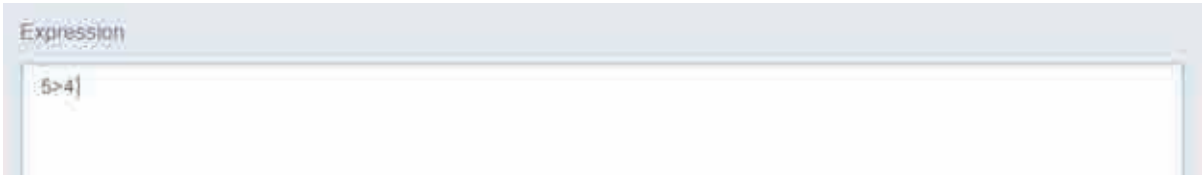
Propiedades Conditional

Numero	Campo	Definición
1	Label	Nombre del conditional
2	ConditionalBreakPoint	Campo para realizar interrupción (romper) por medio de un condicional en el modo **debug**
3	Expression	Expression lógica que es evaluada como verdadera o falsa Puede ser cualquier sentencia, incluyendo otras sentencias anidadas
4	Save	Botón guardar cambios

Expression

Algunas veces en nuestros flujos es necesario que tomemos algunas “decisiones”, esto en el sentido de que necesitamos decidir si ejecutar un flujo no.

La expresión evalúa básicamente una operación lógica, es decir una expresión que de como resultado verdadero o false.



El resultado de esta expresión es verdadera (true), en operaciones básicas 5 es mayor a 4 .

La expresión puede evaluar varias condiciones:

```
$data != 10 or $data != $value1
```

La expresión evalúa la primera condicional $data! = 10 , luego evalúa la segunda condición $data! = $value1

En dado caso alguna de las dos condicionales sea verdadera (true) el flujo continuara por la conexión true, de lo contrario continuara por false.

Al momento de fallar la condicional continua por error.

Comparadores

A continuación, los diferentes comparadores los cuales generan dos valores booleanos, 'True' o 'False':

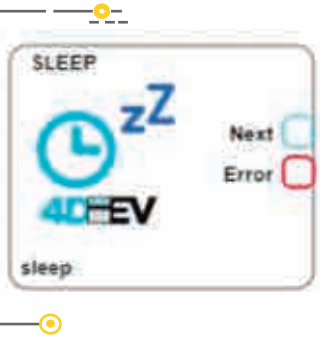
Símbolo	Operación	Ejemplo	Resultado
==	Igual que	5==5	True
!=	Diferente que	4!=5	True
>	Mayor que	98>100	False
<	Menor que	12>39	False
>=	Mayor o Igual	6>=6	True
<=	Menor que	3<=8	True

Operadores lógicos

Para continuar, podemos añadir más funcionalidad a nuestras comparaciones usando los operadores lógicos ‘and’, ‘or’ y ‘not’.

Ejemplo	Resultado		
and	Conjunción	12>2 and 5<10	True
or	Disyunción	9!=6 or 8<=5	True
not “ ”	Negación Agregar texto	not True “valor”	False

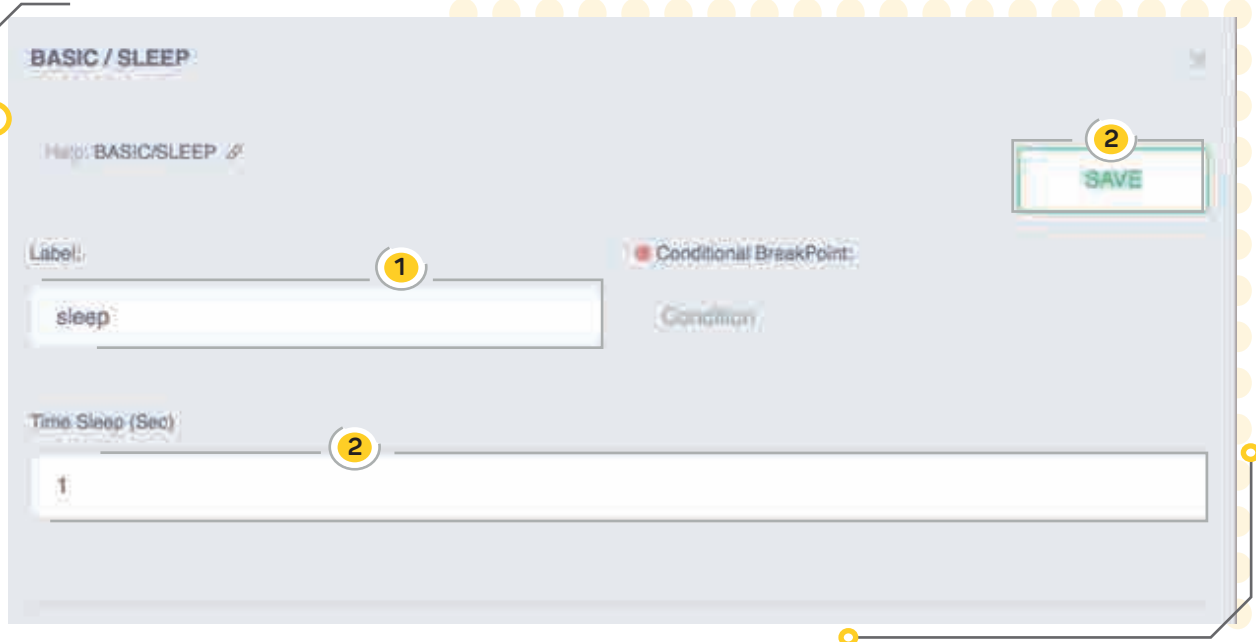
1.3 SLEEP



El comando Sleep Permite hacer una pausa por una determinada cantidad de segundos en la ejecución de un programa. Permite dos tipos de conexiones.

- Next: El comando se completó satisfactoriamente
- Error: El comando resultó en un error

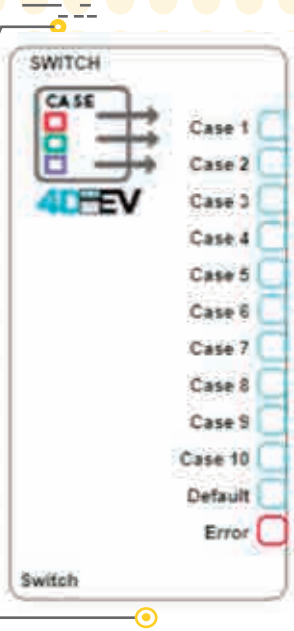
Presionando doble click se despliega el siguiente menú



Propiedades SLEEP

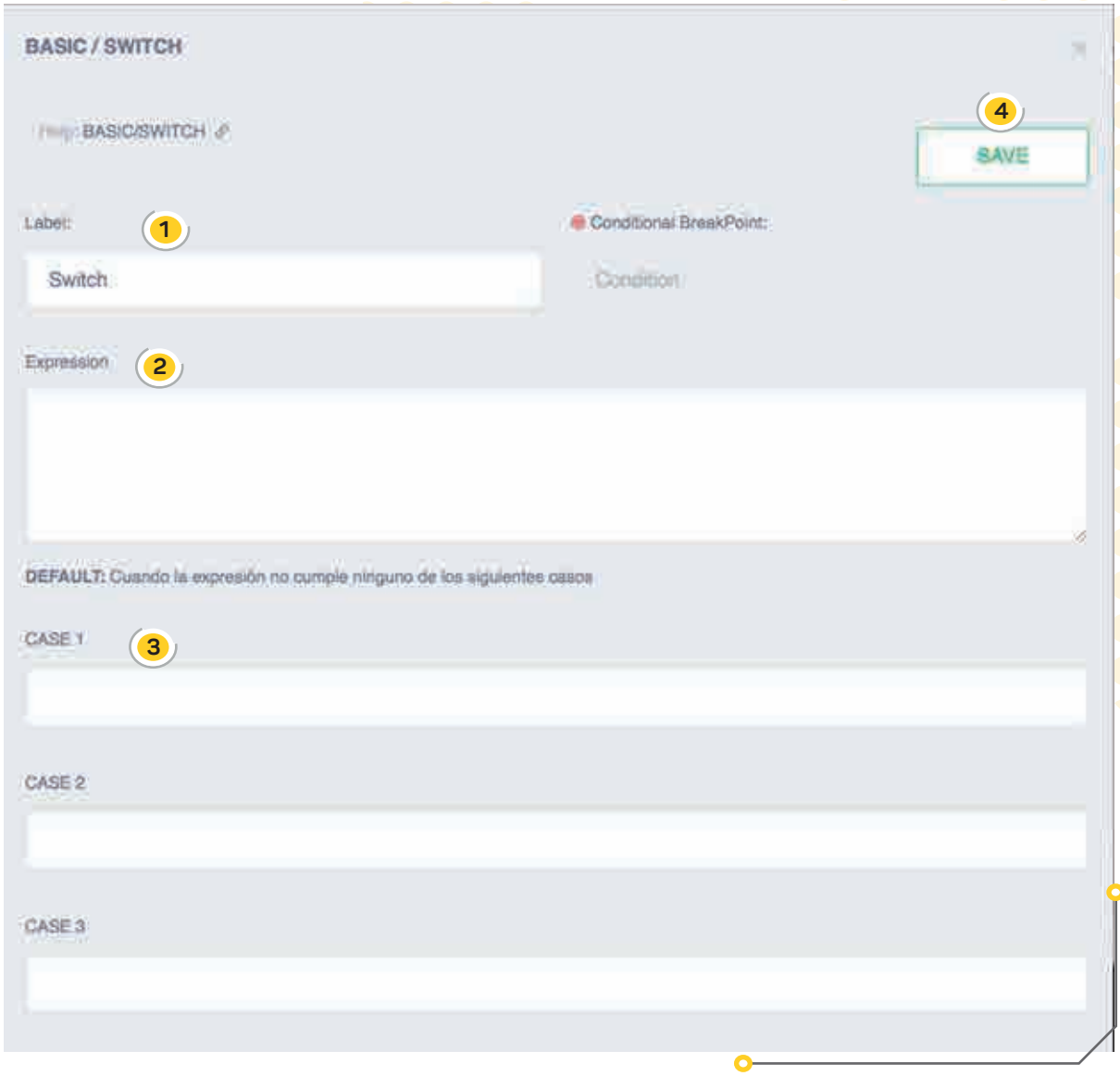
Número	Campo	Definición
1	Label	Nombre del objeto
2	Time Sleep	Tiempo en segundos que se pausa la ejecución del flujo.
3	SAVE	Botón guardar cambios

1.4 SWITCH



El comando Switch compara el valor de una variable con los valores especificados en las instrucciones case. Cuando se encuentra una sentencia case cuyo valor coincide con el de la variable, el código de esa declaración case se ejecuta. Permite los siguientes tipos de conexiones.

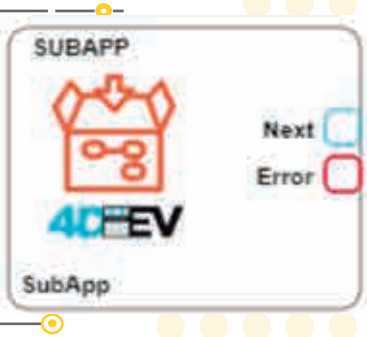
- **Case1-Case10:** Salida de la case que cumple con la expresión
- **Default:** La expresión no coincide con ningún case.
- **Error:** El comando resulto en un error no coincide con ningún de los cases,


BASIC / SWITCH
Help: BASIC/SWITCH
4
SAVE
Label:
1
Switch
Conditional BreakPoint:
Condition
Expression
2
DEFAULT: Cuando la expresión no cumple ninguno de los siguientes casos
CASE 1
3
CASE 2
CASE 3

Propiedades SWITCH

Número	Campo	Definición
1	Label	Nombre del objeto Switch
2	Expression	• Compara el valor de una variable o expresión con los valores especificados en las instrucciones case • Para las expresiones lógicas validas ver documentación del objeto conditional
3	Case1 hasta case 10	La expresión definida puede ser validada por cada uno de los 10 casos "case" posibles
4	SAVE	Botón guardar cambios

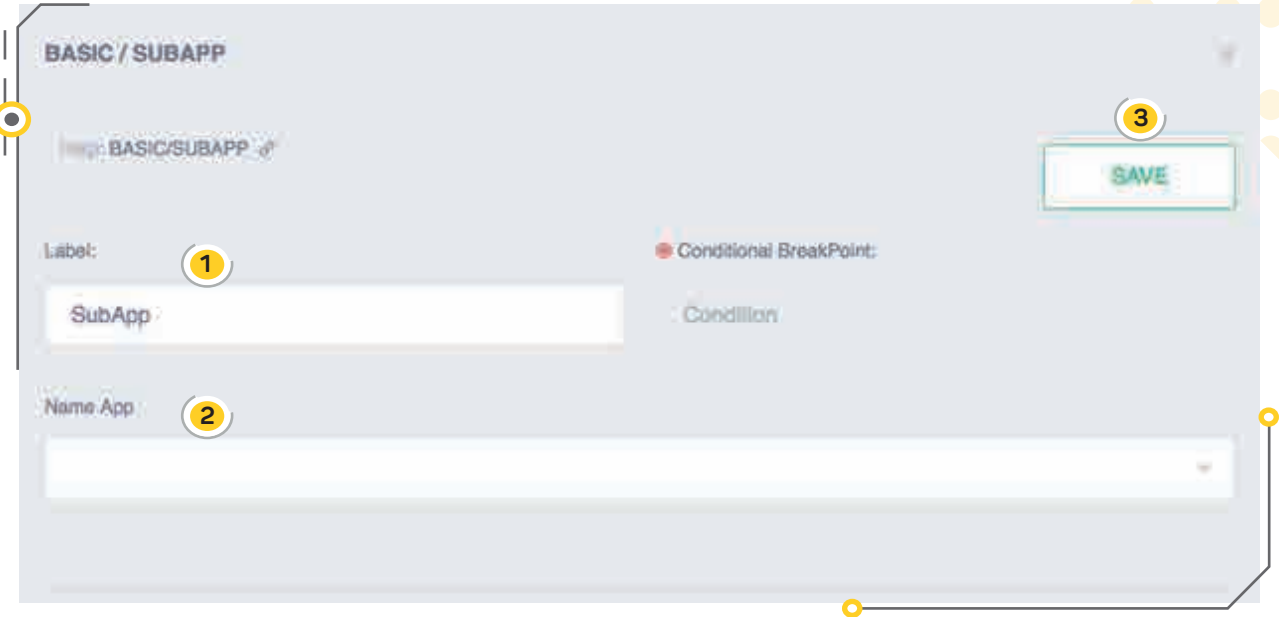
1.5 SUBAPP



El comando SubAPP ejecuta el flujo de una sub-aplicación. Una vez termina la ejecución de la sub-aplicación regresa a la aplicación que lo llamo(principal) para continuar con el siguiente objeto.

Permite las siguientes conexiones:

- **Next:** El comando se completó satisfactoriamente
- **Error:** El comando resulto en un error.



Propiedades SubApp

Número	Campo	Definición
1	Label	Nombre del objeto Switch
2	Nombre de la aplicación	Nombre de la sub-aplicación
3	SAve	Botón guardar cambios

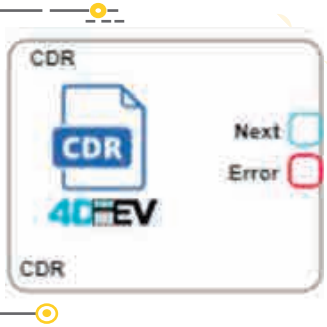
1.6 END

El comando End finaliza la aplicación actual.
Cuando se utiliza en la aplicación principal, simplemente termina. Cuando se utiliza en una sub-aplicación, termina esa aplicación y devuelve el control a la aplicación que lo llamo y un código de retorno a la la aplicación de nivel superior que se captura en la variable $end_return.

Propiedades End

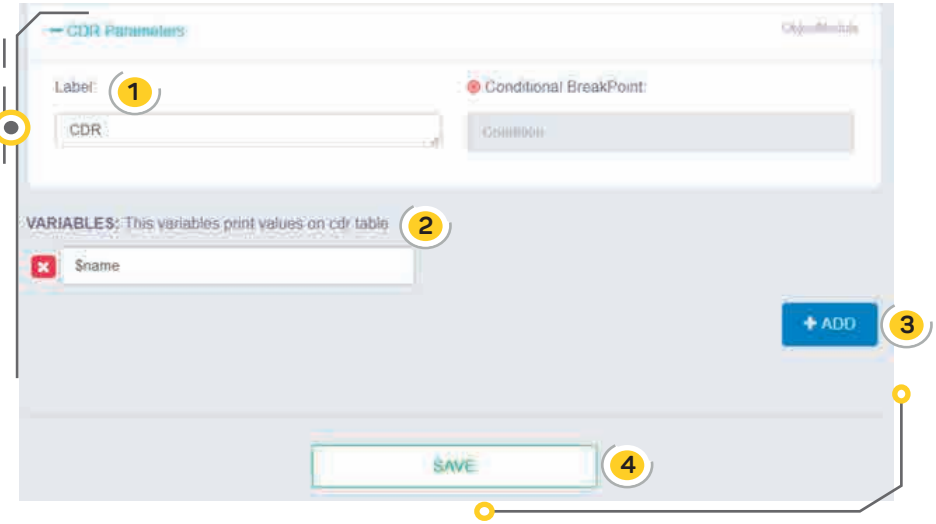
Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Return Value	Retornar variables
3	Save	Botón guardar cambios

1.7 CDR



El comando CDR le da al usuario la posibilidad de registrar y verificar las variables cargadas en el flujo del sistema incluidas en el CDR. Se puede incluir hasta 10 variables en un orden específico. Permite las siguientes conexiones:

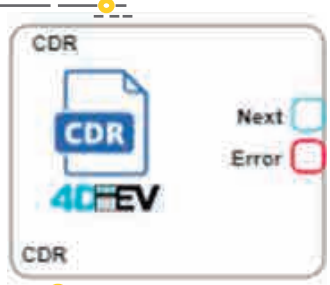
- **Next:** El comando se completó satisfactoriamente
- **Error:** El comando resulto en un error.



Propiedades CDR

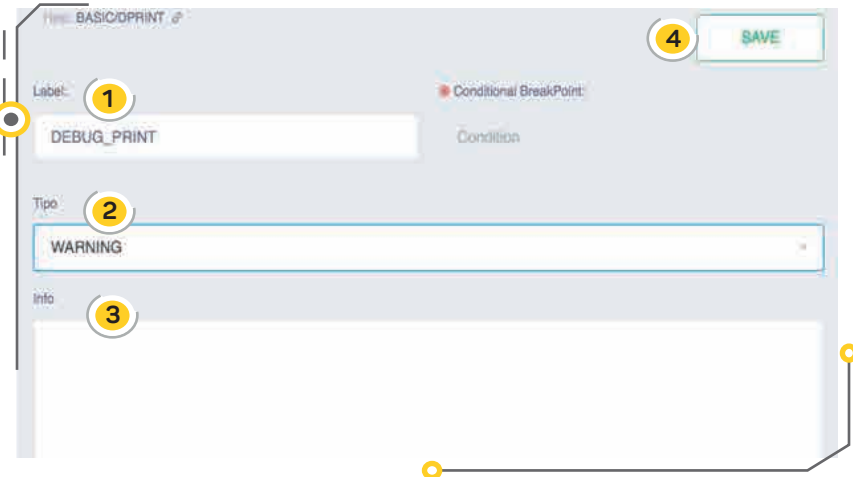
Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Variables	Campo declarar variables
3	ADD	Botón Agregar mas variables
4	Save	Botón guardar cambios

1.8 DPRINT



El comando DPRINT (Debug print) Permite al usuario mostrar en la consola debug un texto o un contenido de una variables. Solo es posible observar la salida si la aplicación se encuentra en modo debug. Permite las siguientes conexiones:

- **Next:** El comando se completó satisfactoriamente
- **Error:** El comando resulto en un error



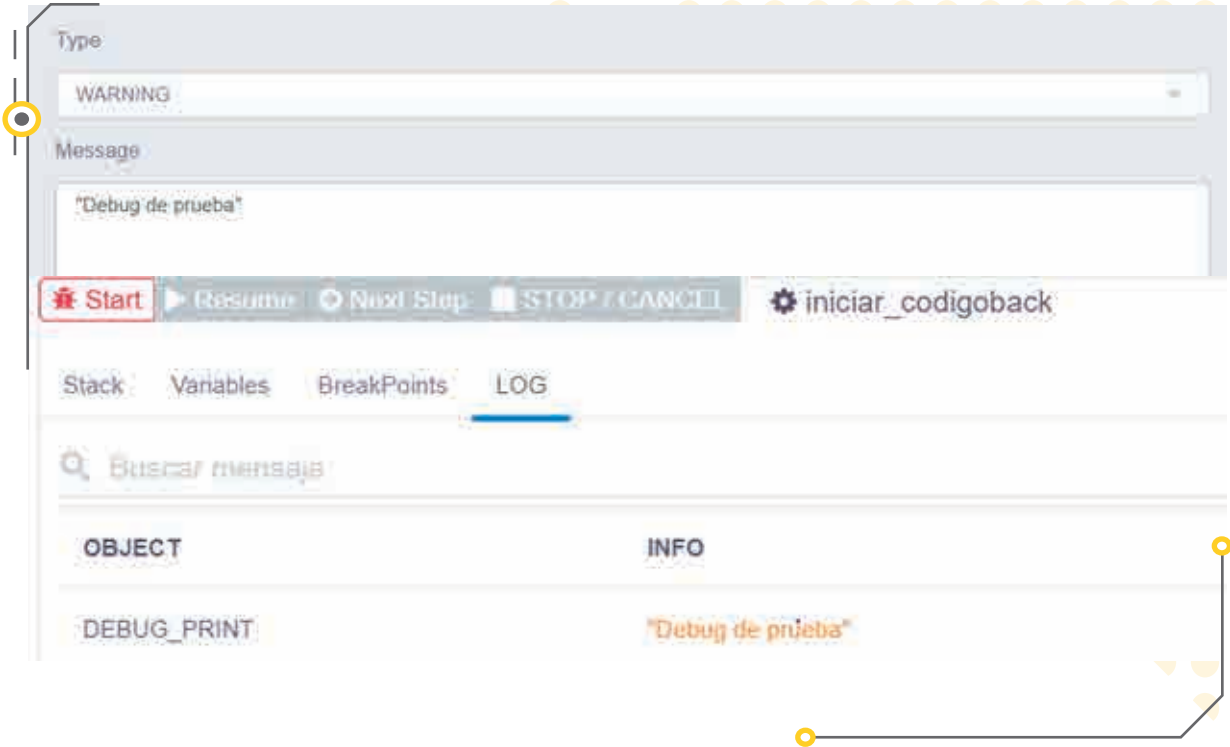
Propiedades Dprint

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Selección de Debug (Info/ Warning/Error)	Selecciones el tipo de evento desde el menú desplegable. **INFO:** Imprime en la consola el evento en color azul **WARNING:** Imprime en la consola el evento en color naranja **ERROR:** Imprime en la consola el evento en color Rojo
3	INFO	Texto que imprimirá en la consola. Es posible imprimir el valor de una variable con el siguiente formato {{ }}

Ejemplo

1.Vista de Dprint

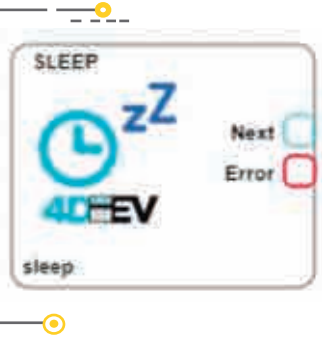
2.Vista del Debug en ejecución con la opción WARNING



2.Vista del Debug en ejecución con la opción INFO.

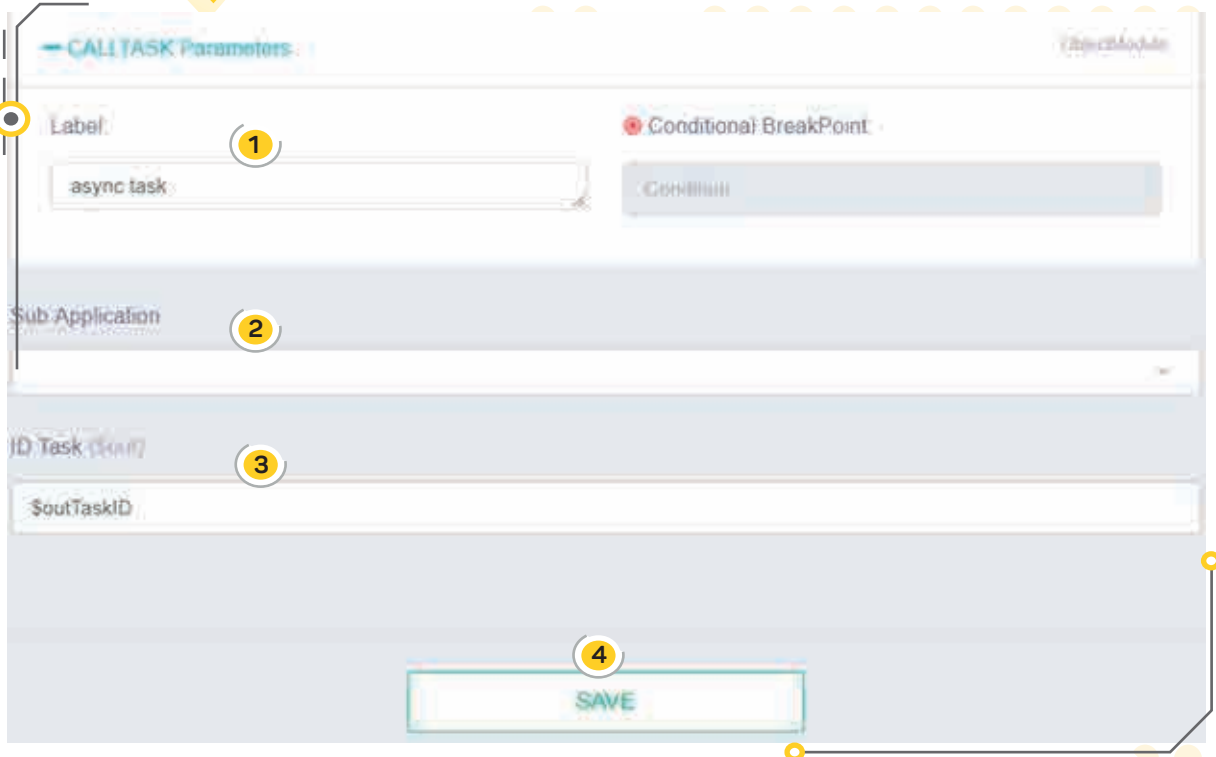
2.Vista del Debug en ejecución con la opción **ERROR.**

1.9 CALLTASK



El comando CALLTASK ejecuta una sub-aplicación de forma asíncrona con la aplicación principal. Permite las siguientes conexiones:

- Next: El comando se completó satisfactoriamente
- Error: El comando resulto en un error

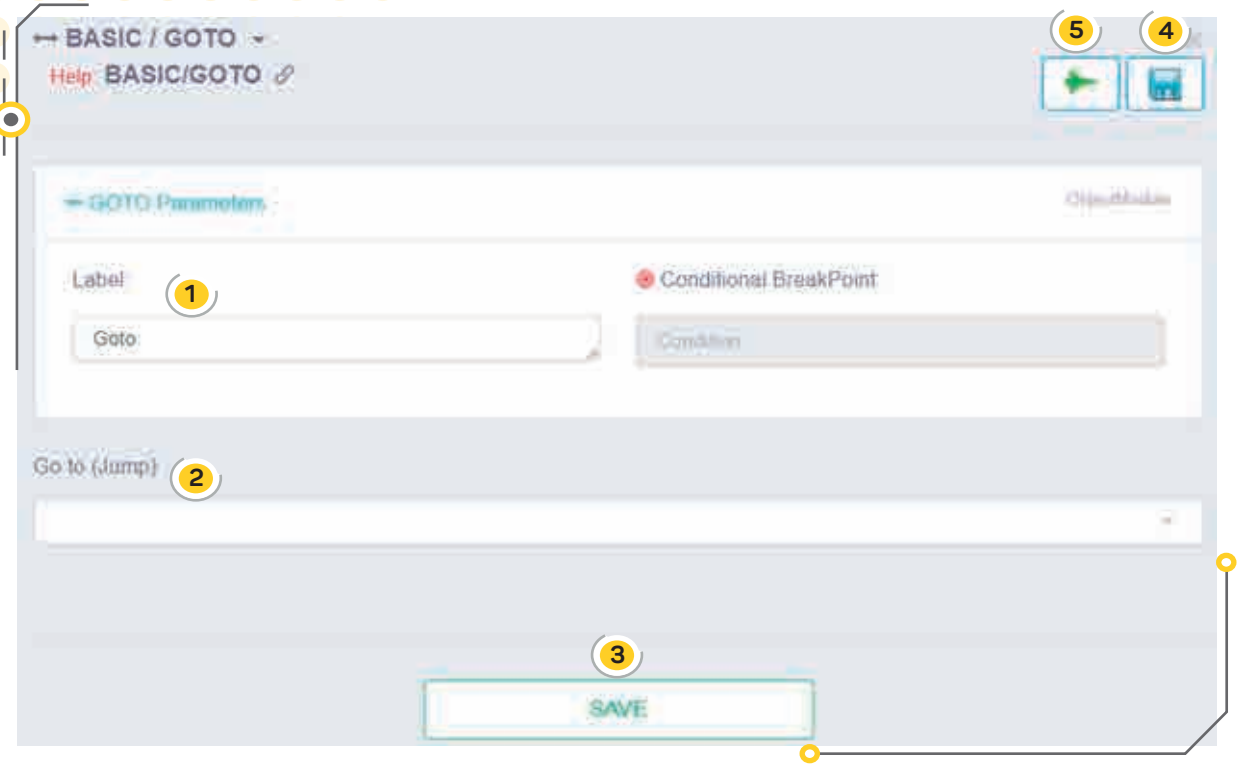


Propiedades Calltask

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Sub Application	Nombre de la Sub-aplicación a ejecutar.
3	ID Task	Variable de salida de la Sub-aplicación
4	save	Botón para guardar los cambios aplicados

1.10 GOTO

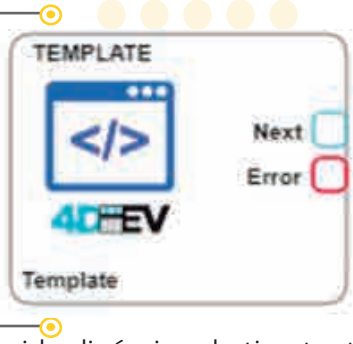
El comando GOTO se utiliza para saltar al siguiente comando. Es una manera fácil de vincular un bloque de flujo de aplicación con otro sin tener que dibujar un conector entre ellos. Los flujos de aplicación pueden aparecer mucho más ordenados cuando se emplea Goto



Propiedades Goto

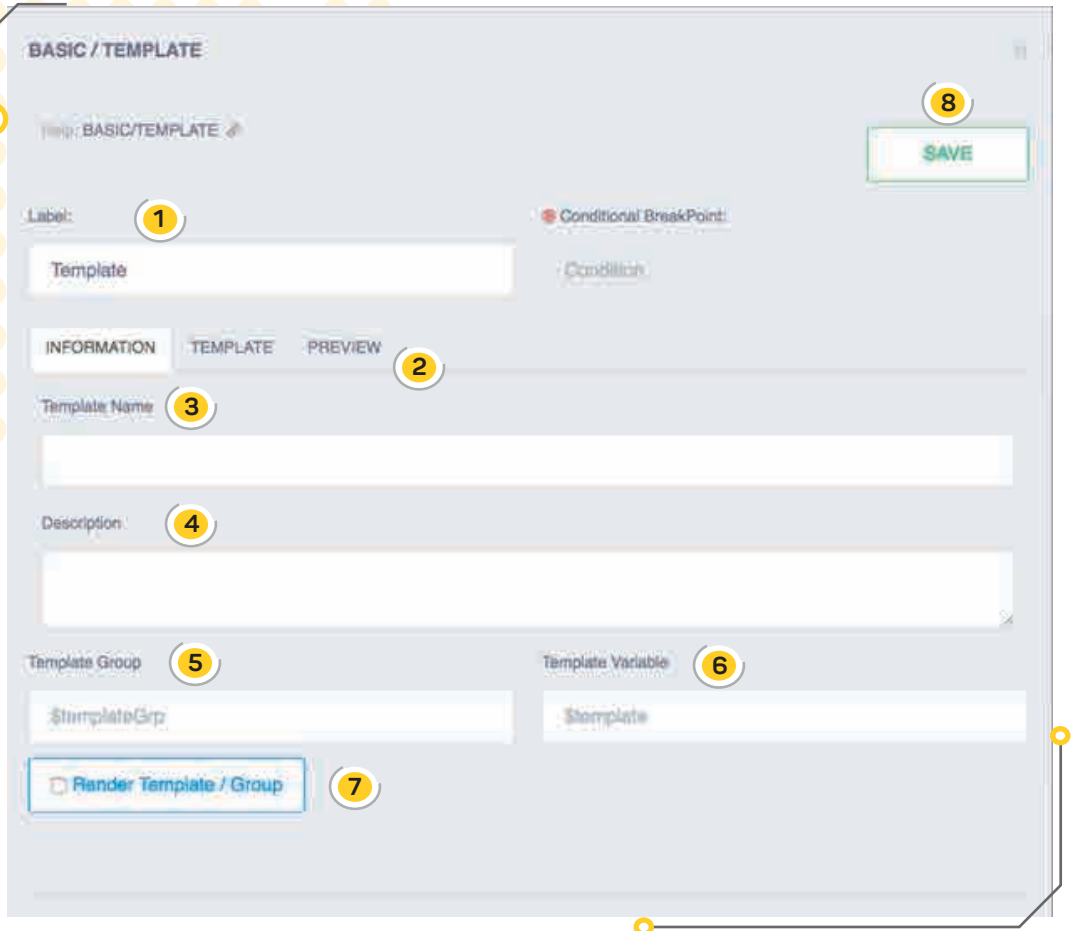
Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Go to (jump)	Selección de objeto a donde se desea redirigir el flujo sin necesidad de conexión.
3	save	Botón para guardar los cambios aplicados
4	Guardar	Botón Guardar cambios globales
5	Deploy	Botón aplicar cambios en el proyecto

1.11 TEMPLATE



El comando TEMPLATE Permite generar contenido dinámico de tipo text, html, xml, json , etc
Este objeto está basado en jinja2
Permite las siguientes conexiones:

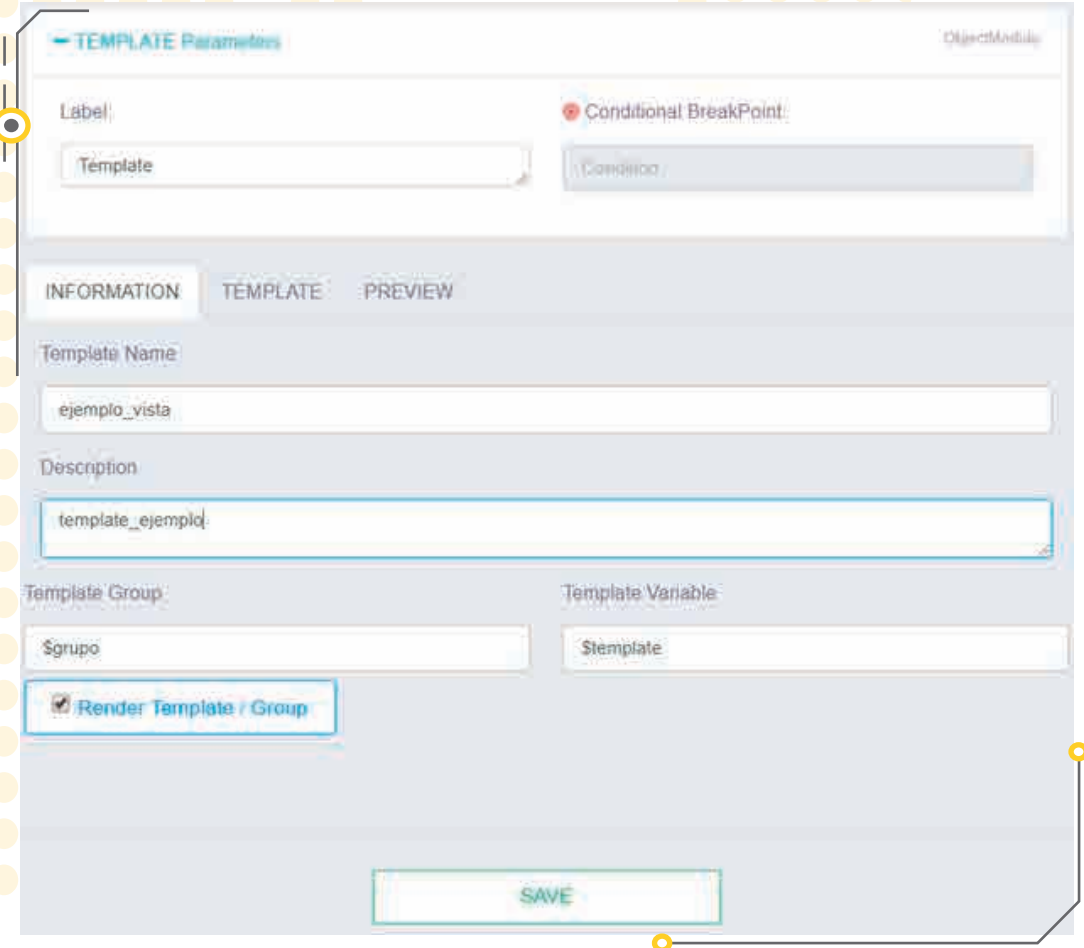
- **Next:** El comando se completó satisfactoriamente
- **Error:** El comando resulto en un error



Propiedades Template

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Menú (INFORMATION/TEMPLATE/ PREVIEW)	TEMPLATE: Código de diseño para la vista del cliente PREVIEW: Previsualización de cómo queda la página.
3	Template Name	Nombre del template
4	Description	Descripción corta del template
5	Template Group	Nombre de la variable que contiene el grupo
8	SAVE	Botón para guardar los cambios aplicados

Ejemplo



Variables declaradas

Template Group

- Nombre del grupo: $grupo
  Seleccionado el render como se observa en la siguiente imagen el template ejecuta todos los templates agrupados con el mismo nombre de grupo, además ejecuta todo el código empaquetado en un template

- Template

```
<!DOCTYPE html>
<html>
  <head>
    <title>Flask Template Example</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="http://netdna.bootstrapcdn.com/bootstrap/3.0.0/css/
    bootstrap.min.css" rel="stylesheet" media="screen">
    <style type="text/css">
      .container {
        max-width: 500px;
        padding-top: 100px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <p>Name:{{my_string}} {{dataMysql['username']}}  </p>
      <p>fecha: {{Fecha}}</p>
      <p>Su codigo es :  {{dataMysql['password']}}</p>
      <p>Value from the list: {{my_list[3]}}</p>
       <p>Loop through the list:</p>
      <ul>
        {% for n in my_list %}
        <li>{{n}}</li>
        {% endfor %}
      </ul>
    </div>
    <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
    <script src="http://netdna.bootstrapcdn.com/bootstrap/3.0.0/js/
    bootstrap.min.js"></script>
  </body>
</html>
```

Ejecución de codigo

Name:hola crisitan

fehcga: 28/ 05/ 18

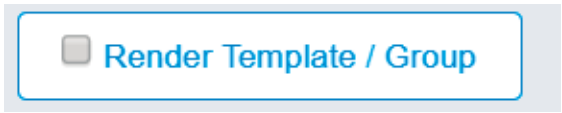
Su codigo es : 123456

Value from the list: 4

Loop through the list

- 1
- 2
- 3
- 4
- 5

Render no seleccionado



Name: {{my_string}} {{dataMysql']}}['username']}}

fecha: {{ Fecha }}

Su codigo es : {{dataMysql['password']}}

Value from the list: {{ my_list[3]}}

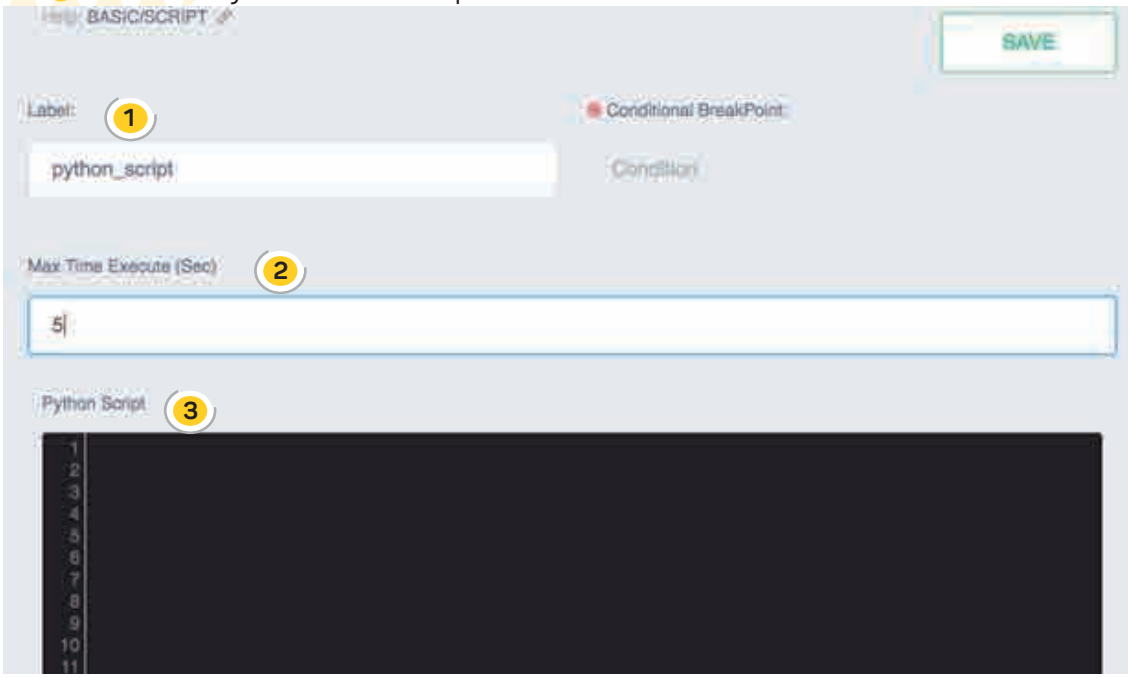
Loop through the list:

- {%for n in my_list%}
  {{n}}
  {%endfor%}

1.12 Script Python

Objeto SCRIPT PYTHON permite ejecutar código Python dentó de una aplicación
Permite las siguientes conexiones:

- Default: No tiene retorno o Esta fuera de los valores 0-9
- Timeout: La ejecución del script supero el tiempo de espera.
- Error: La ejecución del script resultó en un error.



Propiedades Script Python

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Max Time Excute(sec)	Tiempo máximo que puede tardar en ejecutarse
3	Python Script	El script que será ejecutado
4	save	Botón guardar cambios

Pasando datos desde el script

La siguiente función es usada para pasar valores a variables de la aplicación desde el script de python:
set_4dev_variable()
Ejemplo:

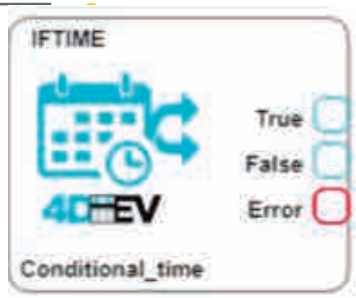
```
a= 2
b = 3
suma = a  + b
set_4dev_variable($suma, suma)
```

Generar una salida por conector:

```
set_4dev_variable($_pyreturn, 0)
```

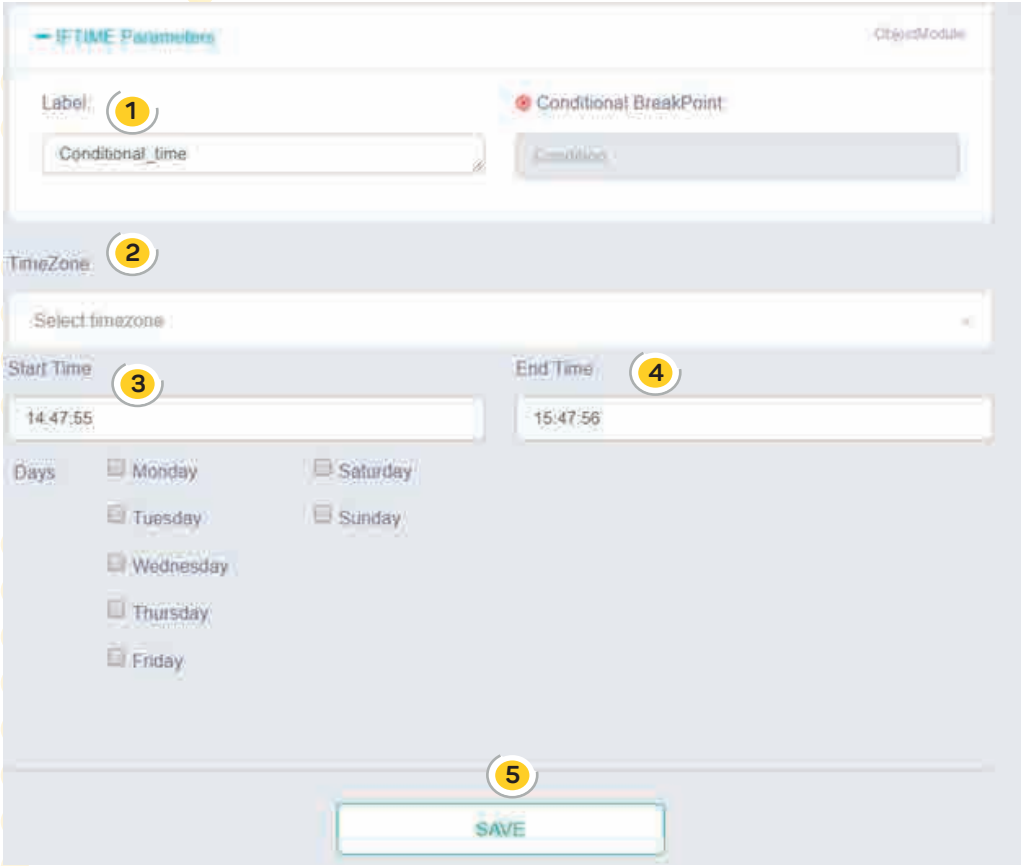
Para mas información y preguntas frecuentes acerca de los script de python por favor ver el siguiente sitio

1.13 Conditional Time



Conditional time es usado para validar y ejecutar procesos por fecha, día, hora o zona horaria.
Permite las siguientes conexiones

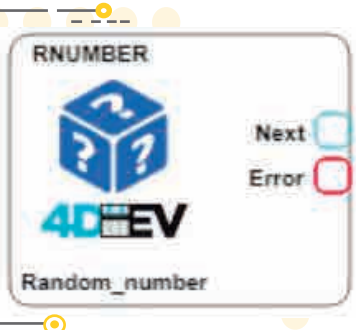
- True: La expresión es verdadera
- False: La expresión es falsa
- Error: El comando resultó en un error



Propiedades Conditional Time

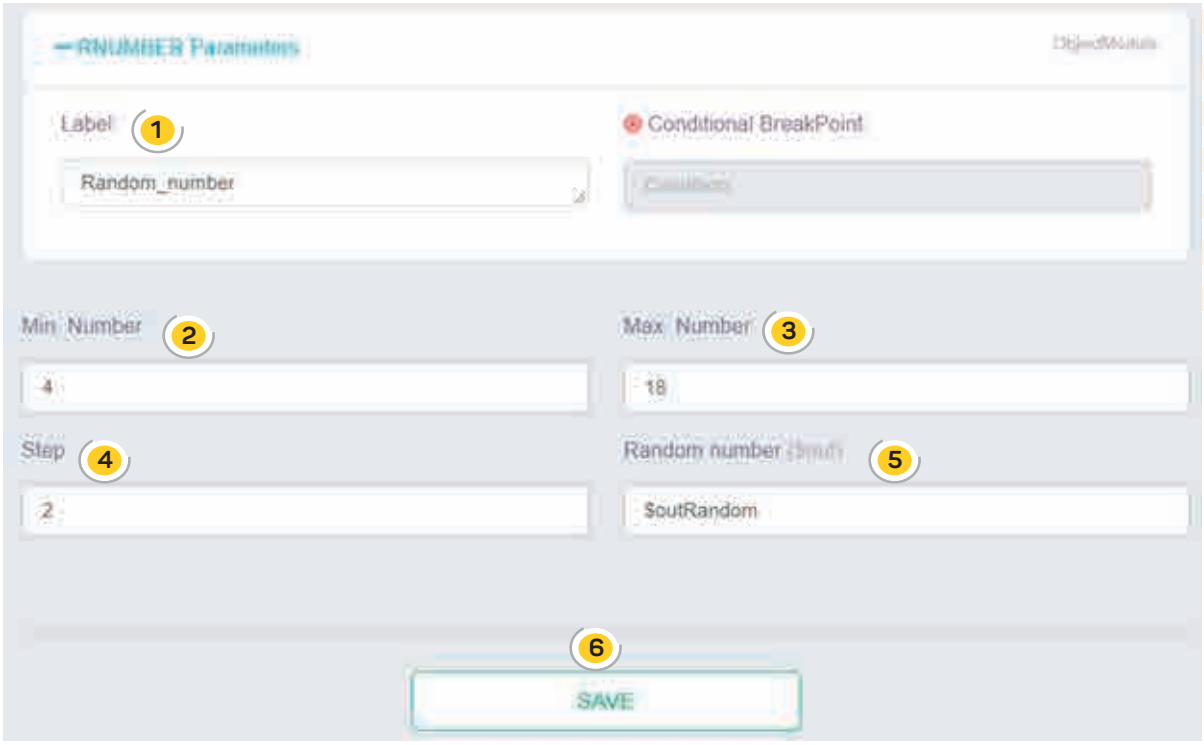
Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Time zone	Zona horaria para seleccionar
3	Start Time	Hora de inicio. Días para seleccionar
4	End Time	Hora final
5	save	Botón guardar cambios

1.14 Random Number



El comando Random Number es usado para generar números aleatorios.
Permite las siguientes conexiones

- **Next:** El comando se completó satisfactoriamente
- **Error:** El comando resultó en un error.



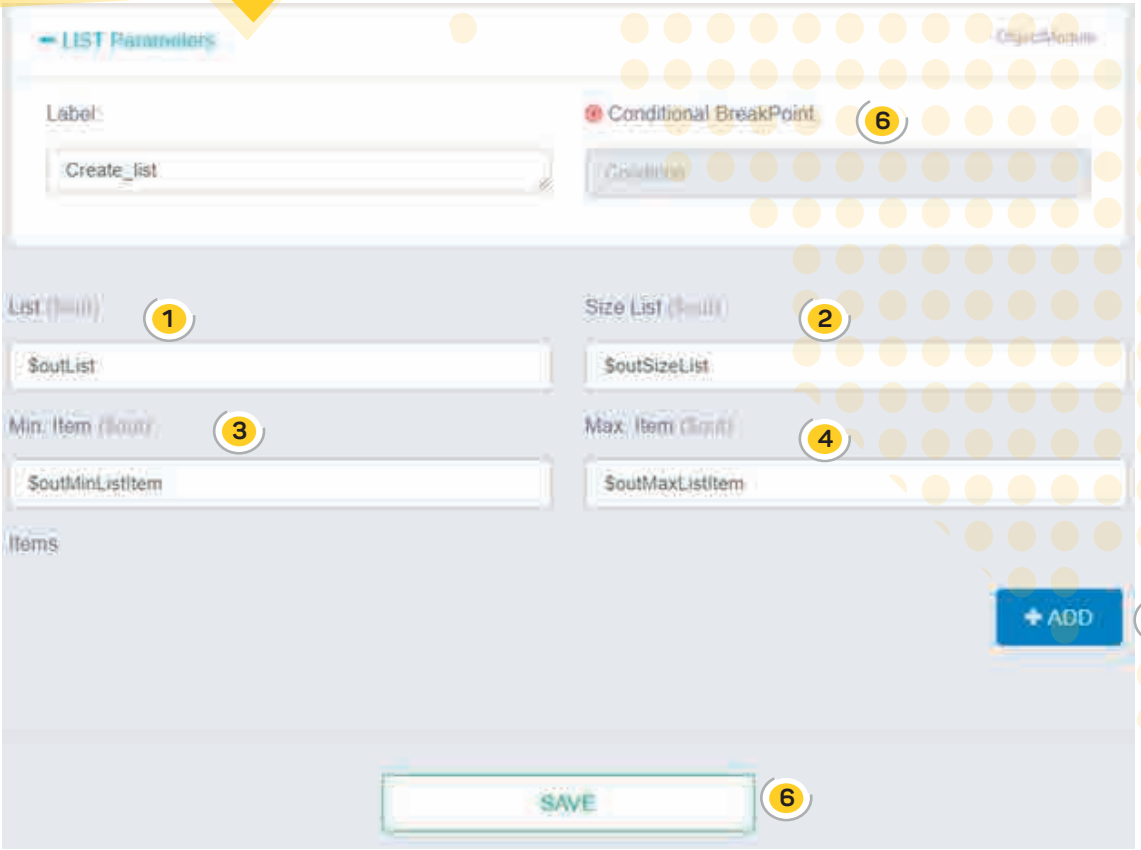
Propiedades Random Number

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Min.Number	Numero mínimo para empezar
3	Max.Number	Número máximo para generar
4	Step	Numero de incremento
5	SAVE	Botón guardar cambios

1.15 List

Objeto list nos permite crear listas dinámicas con una colección ordenada de artículos que pueden ser de diferentes tipos.
Permite las siguientes conexiones

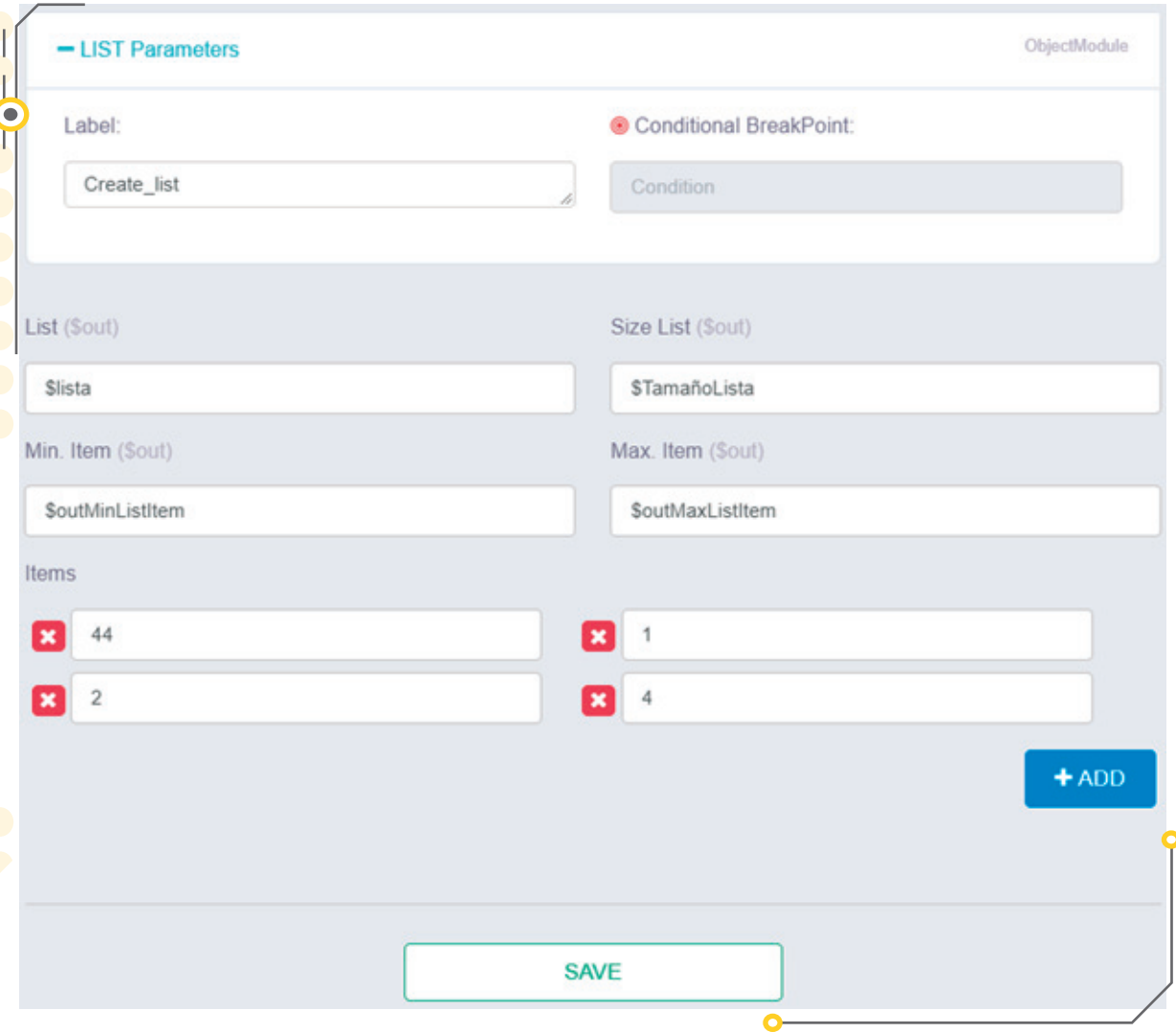
- Next: El comando se completó satisfactoriamente
- Error: El comando resultó en un error.



Propiedades List

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Size List	Tamaño de la lista
3	Min.Item	Valor mínimo de la lista
4	Min.Item	Valor máximo de la lista
5	ADD	Botón agregar valores de la lista

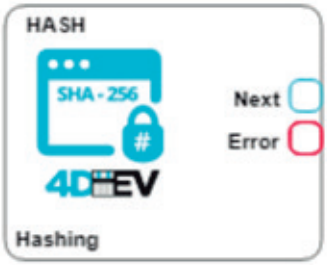
Ejemplo



En el ejemplo se declara la lista $lista los valores de la lista son 44, 1.2,4

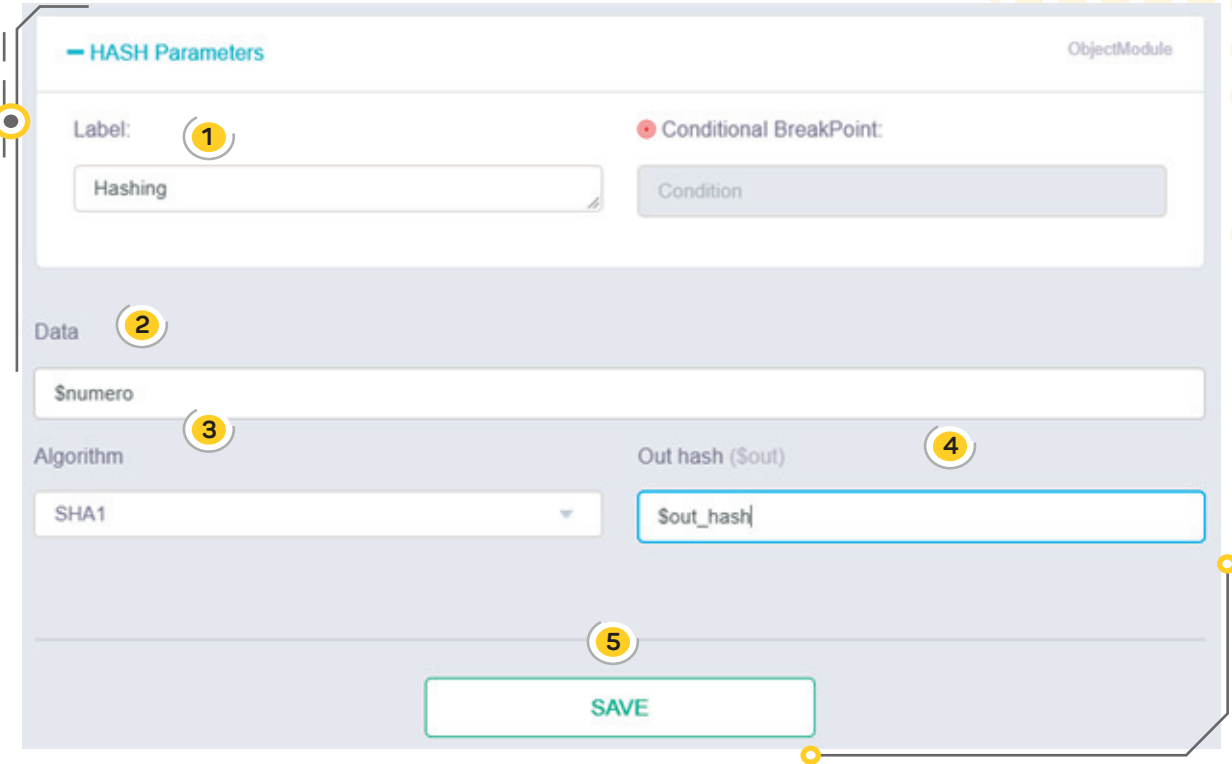
$lista = [44,1.2,4]

1.16 Hash



Este módulo implementa una interfaz de modo seguro a muchos algoritmos .
Se incluyen los algoritmos de hash de seguridad FIPS SHA1, así como el algoritmo MD5.
Permite las siguientes conexiones

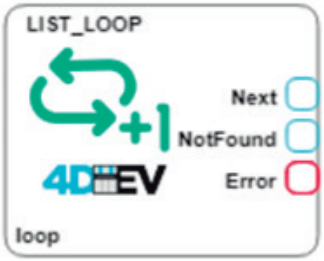
- **Next:** El comando se completó satisfactoriamente
- **Error:** El comando resultó en un error



Propiedades Hass

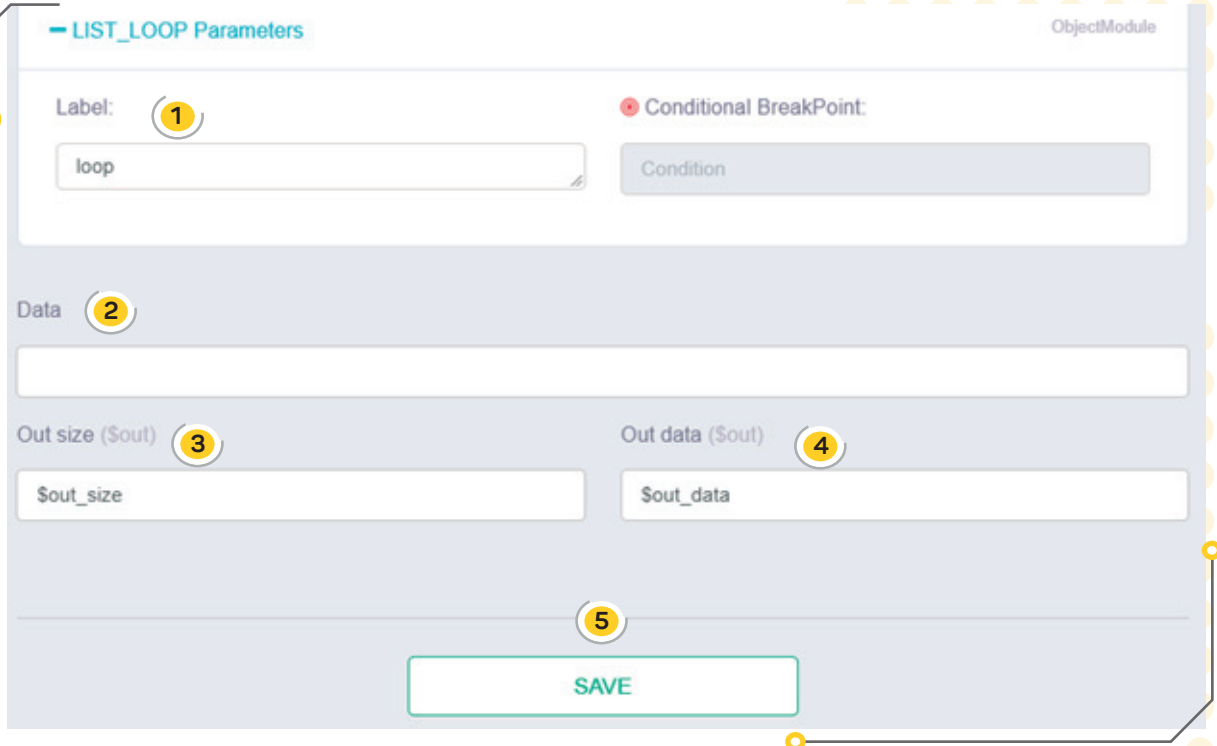
Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Data	Variable a hashear
3	Algorithm	Algoritmo seleccionar
4	Out hash	Hora final
5	SAVE	Botón guardar cambios

1.17 List_Loop



Este módulo nos permite que un bloque de código se repita un cierto número de veces dependiendo del dato a repetir.
Permite las siguientes conexiones:

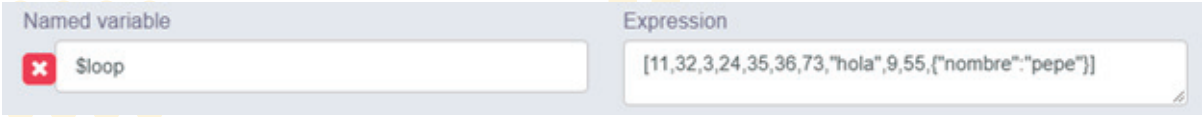
- **Next:** El comando se completó satisfactoriamente,
- **Error:** El comando resultó en un error,
- **NotFound:** El ciclo termina.



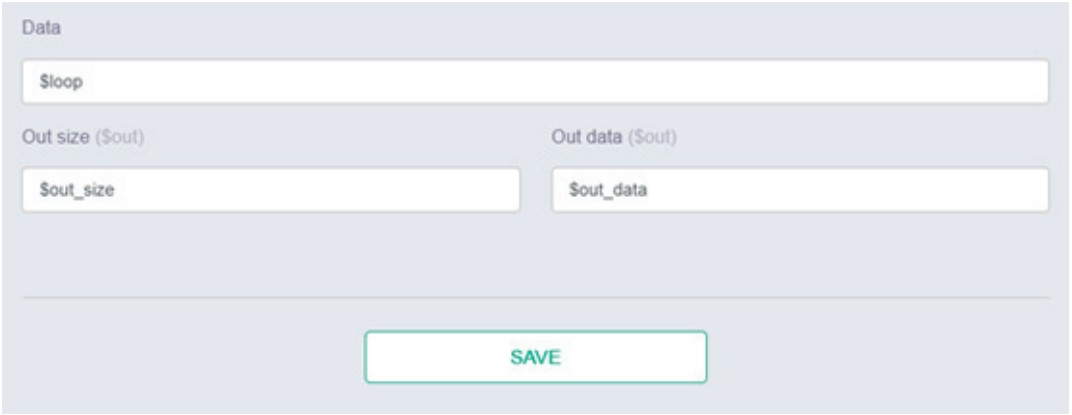
Propiedades List_Loop

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Data	Variable o expresión para el ciclo
3	Out size	Variable de salida
4	Out data	Variable con la conexión de Mysql
5	SAVE	Botón guardar cambios

Ejemplo:
Declaramos la variable con el valor [11,32,3,24,35,36,73,"hola",9,55,{"nombre":" pepe"}] definida como arreglo.

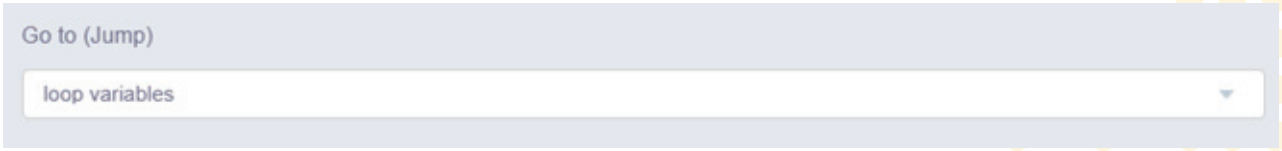


Escribimos la variable loop ya creada anteriormente

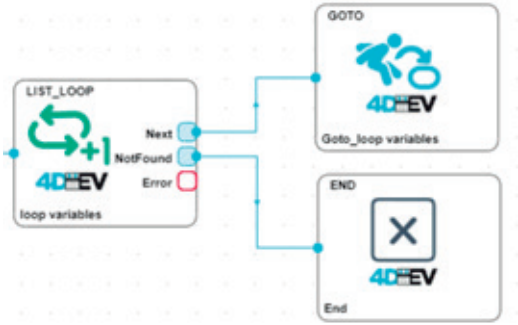


Creamos un Goto para formar un bucle que repite el bloque de instrucciones un número predeterminado de veces.

Redireccionamos el Goto hacia el loop.



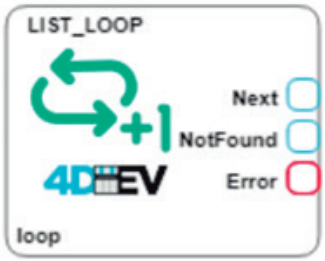
La Conexión NotFound es la salida en la cual termina el ciclo después de recorrer los objetos de la variable contenida en el campo Data



Modulo Base de datos

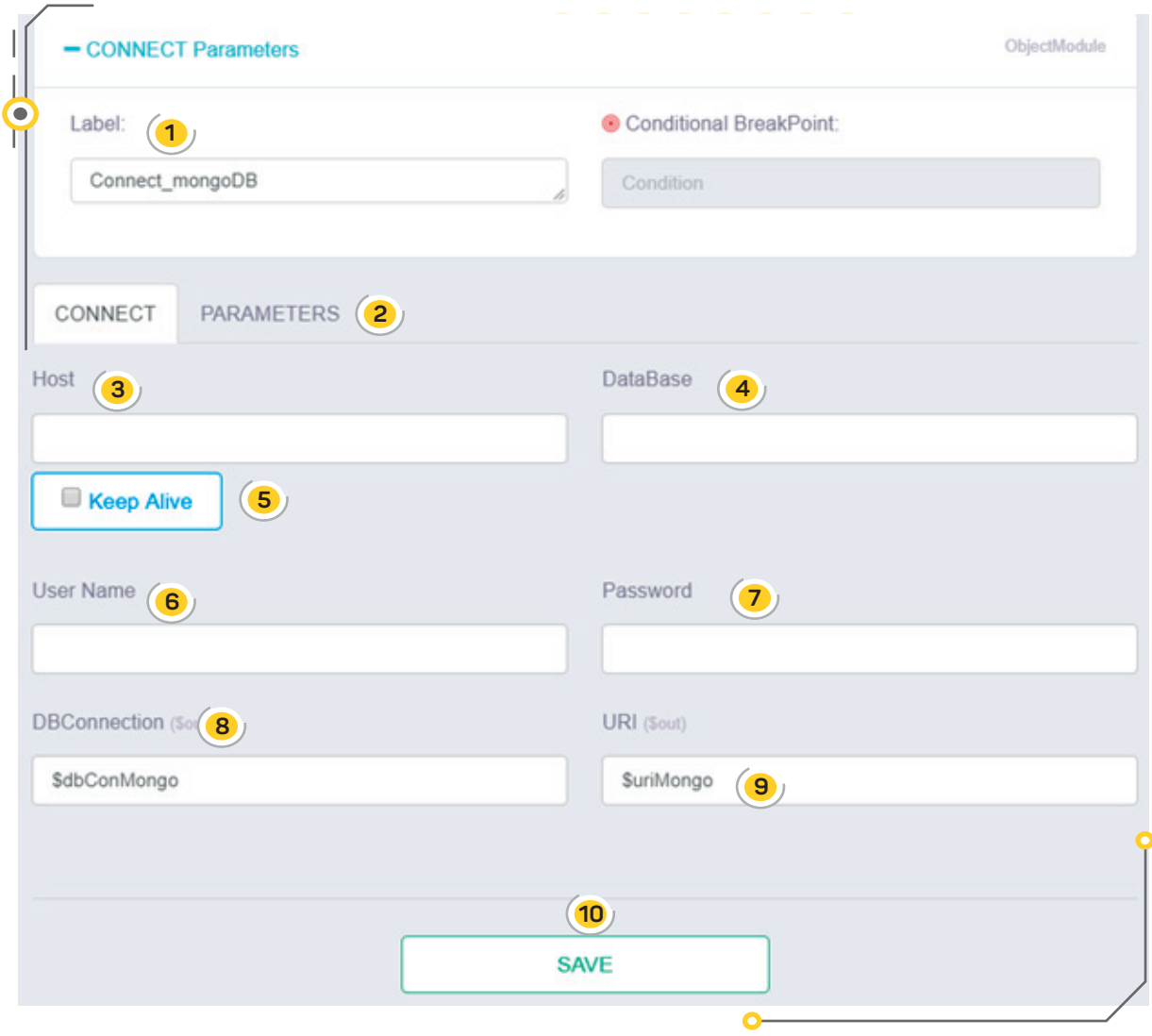
Barra de herramientas para conexión a base de datos

2.1Connect_mongoDB



Conexión de base de datos a Mongo
Permite las siguientes conexiones:

- **Success** la Conexión se completó satisfactoriamente,
- **Error:** El comando resultó en un error,

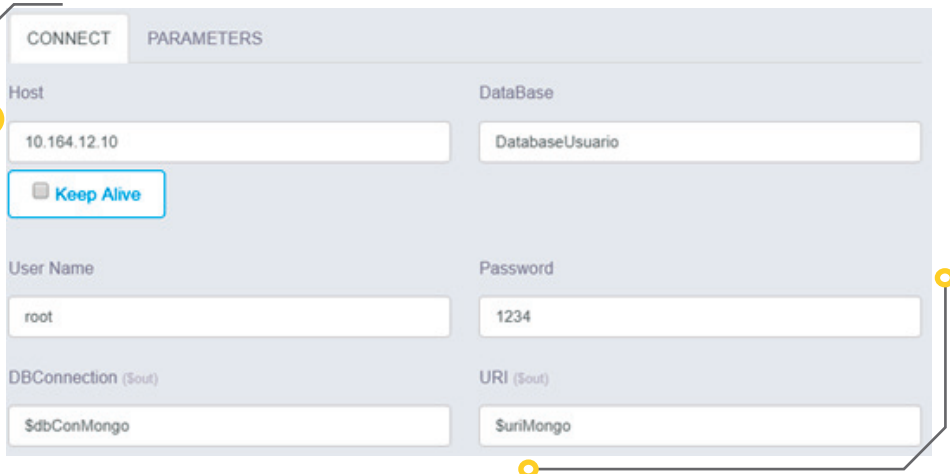

CONNECT Parameters
ObjectModule
Label:
1
Connect_mongoDB
Conditional BreakPoint:
Condition
CONNECT
PARAMETERS
2
Host
3
DataBase
4
Keep Alive
5
User Name
6
Password
7
DBConnection
8
URI ($out)
$dbConMongo
$uriMongo
9
10
SAVE

Propiedades Connect_mongoDB

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Menú (CONNECT/PARAMETERS)	CONNECT: PARAMETERS:
3	Host	Dirección de la base de datos
4	DataBase	Nombre de la base de datos
5	Keep Alive	
6	User Name	Usuario base de datos
7	Password	Contraseña base de datos
8	DBconnection	
9	URI	
10	SAVE	Botón guardar cambios

Ejemplo

Realizamos la conexión a base de datos colocamos el Host, el nombre de la base de datos, usuario de base de datos , la contraseña.

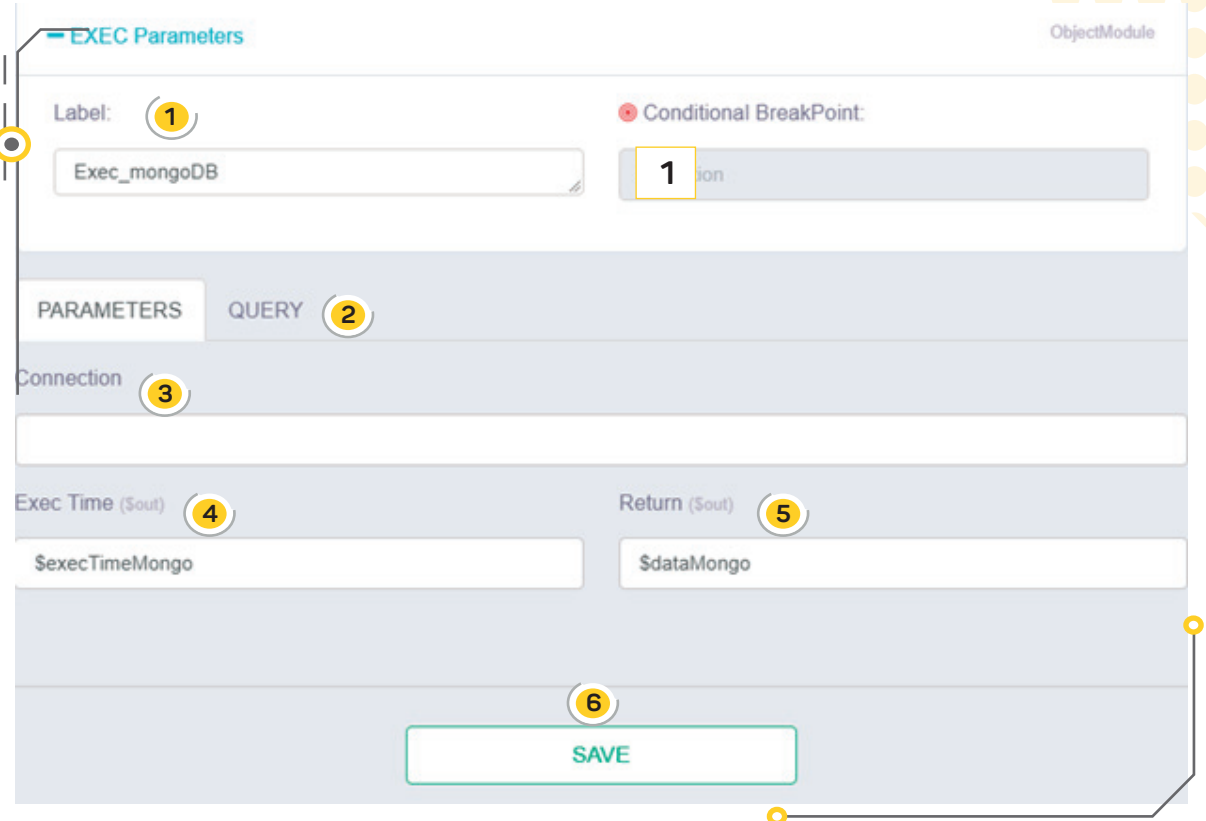


2.1Connect_mongoDB

Módulo de Ejecución de sentencias Query en Mongo

Permite las siguientes conexiones:

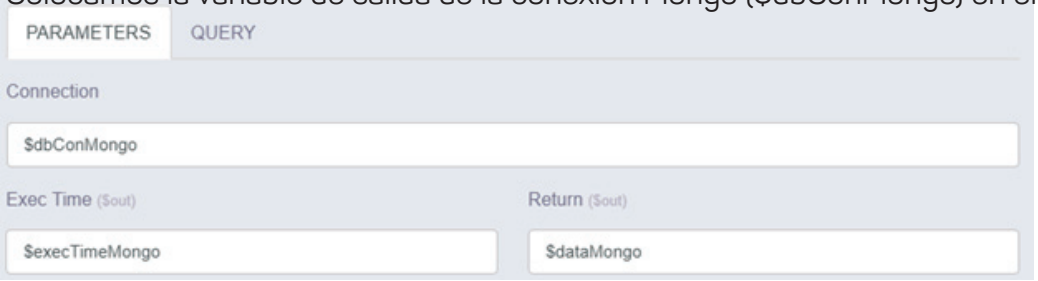
- Success la Conexión se completó satisfactoriamente,
- Error: El comando resultó en un error,
- NoFund : Consulta sin datos retornados



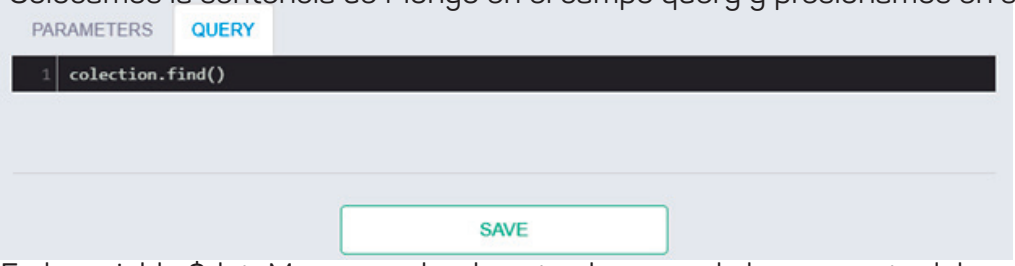
Propiedades Exec_mongoDB

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Menú (CONNECT/PARAMETERS)	PARAMETERS: QUERY:
3	Connection	Campo conexión BD
4	Exec Time	Tiempo de ejecución consulta
5	Return	Variable de salida
6	SAVE	Botón guardar cambios

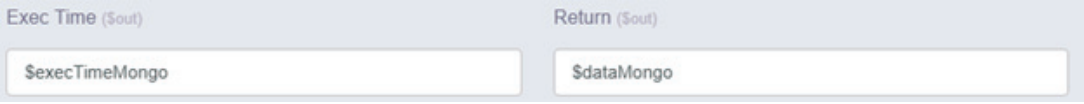
Ejemplo:
Colocamos la variable de salida de la conexión Mongo ($dbConMongo) en el campo Connection



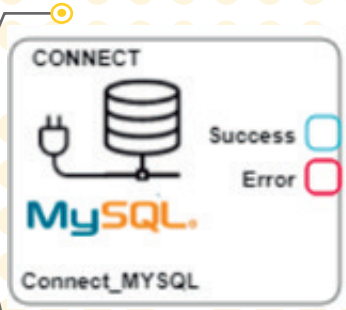
Colocamos la sentencia de Mongo en el campo query y presionamos en save



En la variable $dataMongo es donde esta almacenada la respuesta del query

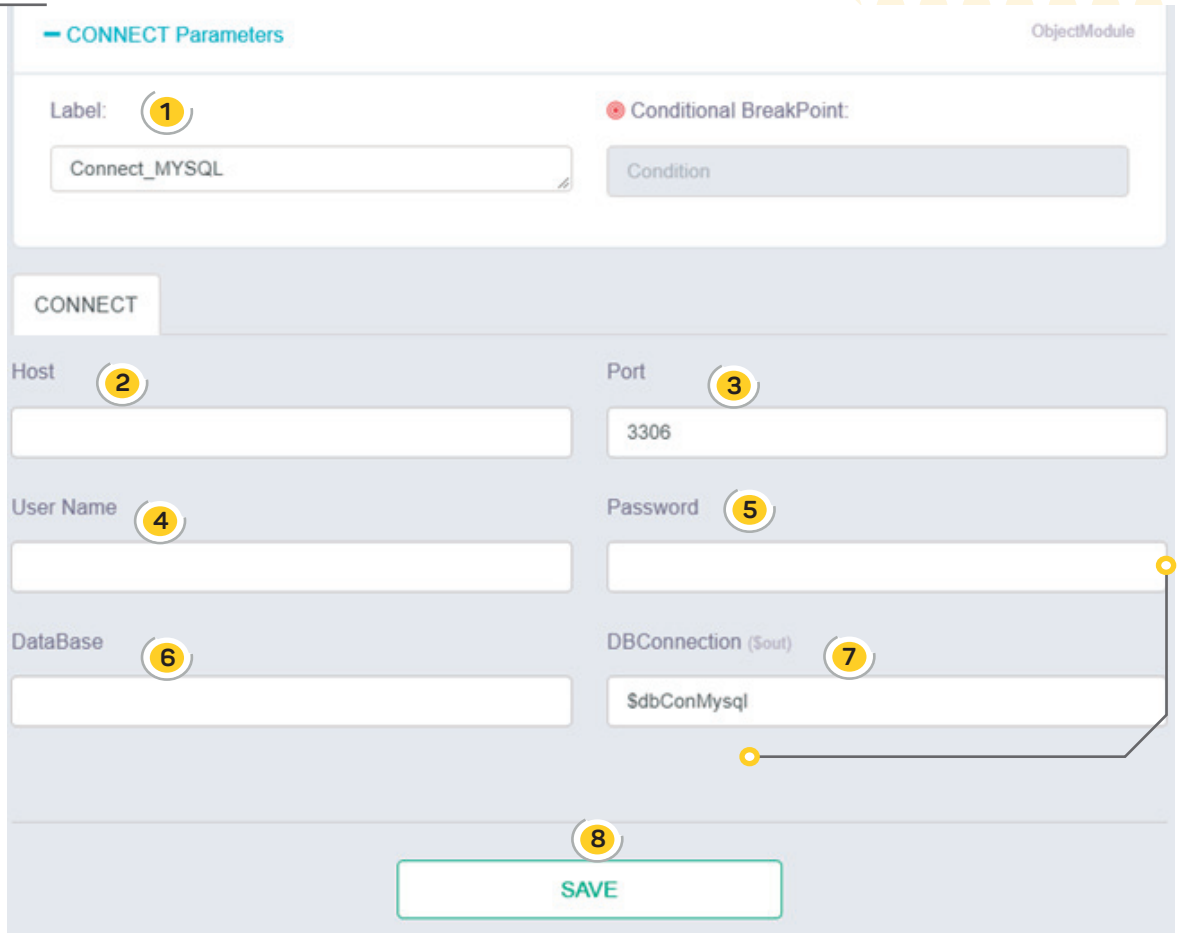


2.1Connect_MYSQL



Conexión de base de datos MYSQL
Permite las siguientes conexiones:

- **Success** la Conexión se completó satisfactoriamente,
- **Error:** El comando resultó en un error,

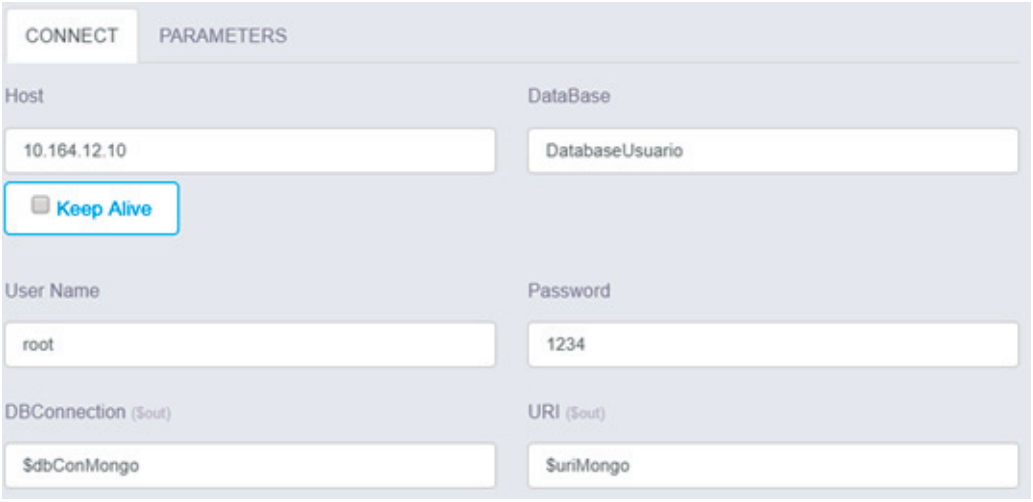


Propiedades Exec_mongoDB

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Host	Dirección de base de datos a conectar
	Port	Puerto para conexión
3	DataBase	Nombre de usuario
4	Keep Alive	Contraseña
5	DataBase	Nombre de la base de datos
6	DBConnection	Variable de salida para la conexión
7	SAVE	Botón guardar cambios

Ejemplo

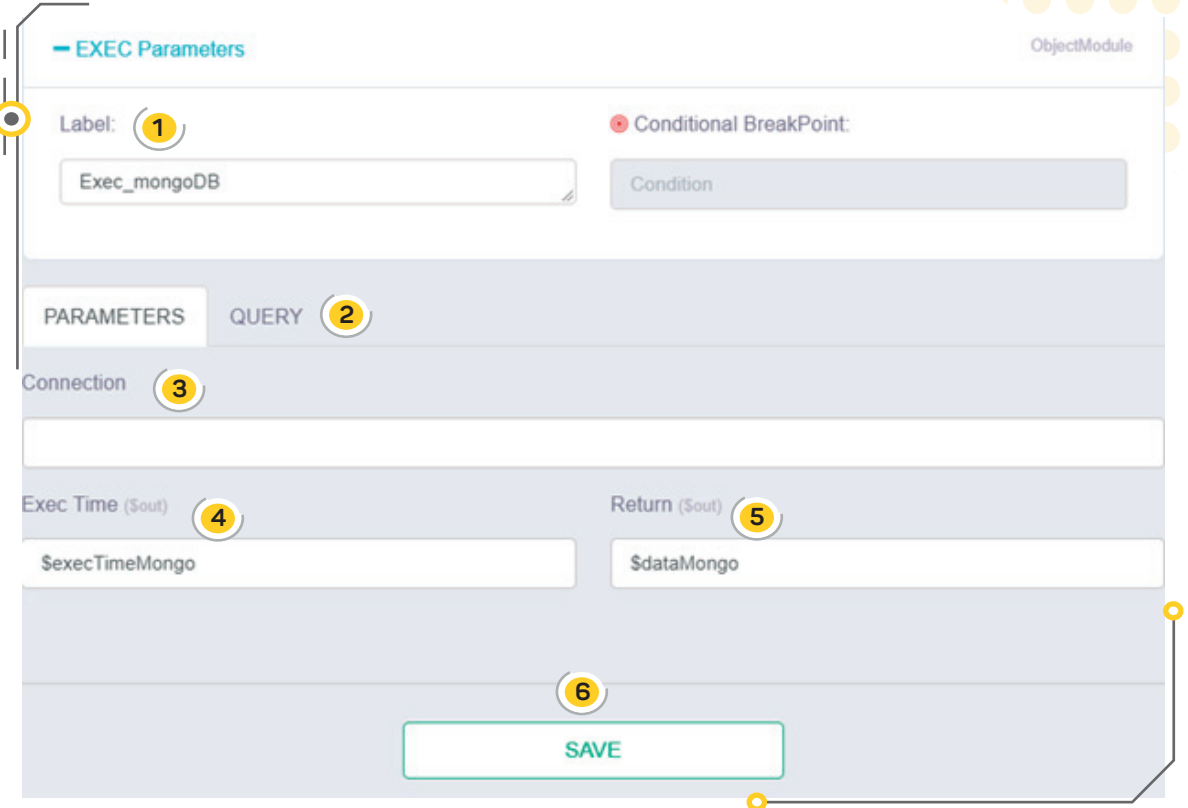
Realizamos la conexión a base de datos colocamos el Host , el nombre de la base de datos, usuario de base de datos , la contraseña.



2.2 Exec_mongoDB

Este módulo nos permite que un bloque de código se repita un cierto número de veces dependiendo del dato a repetir.
Permite las siguientes conexiones:

- **Next:** El comando se completó satisfactoriamente,
- **Error:** El comando resultó en un error,
- **NotFound:** El ciclo termina.

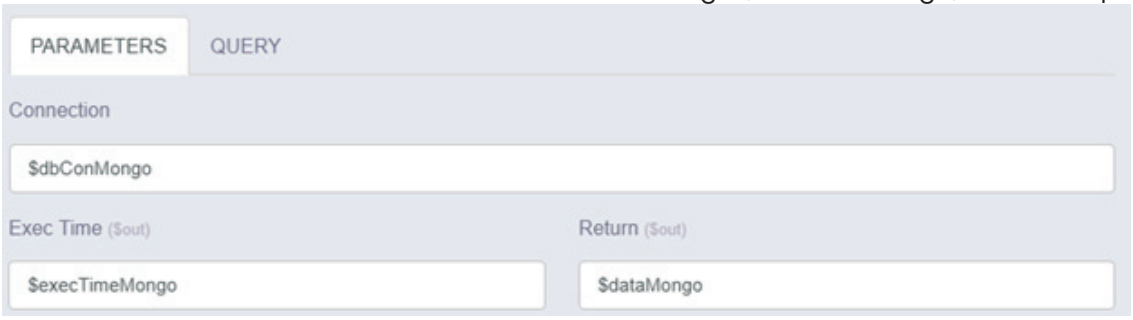


Propiedades Exec_mongoDB

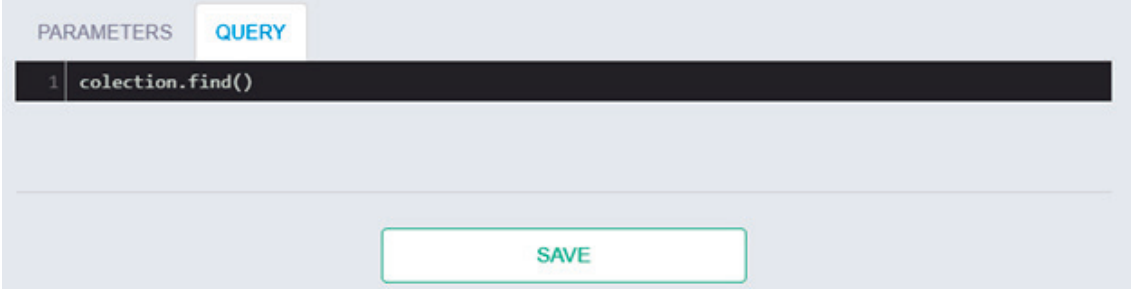
Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Menú (PARAMETERS/QUERY)	PARAMETERS: QUERY:
3	Connection	Campo conexión BD
4	Exec Time	Tiempo de ejecución consulta
5	Return	Variable de salida
6	SAVE	Botón guardar cambios

Ejemplo:

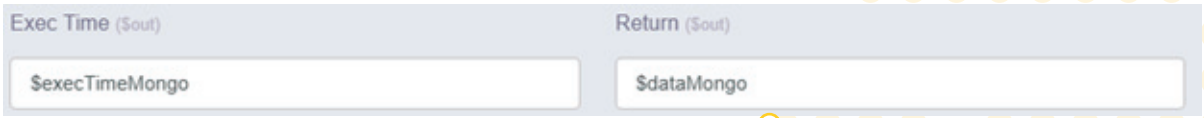
Colocamos la variable de salida de la conexión Mongo ($dbConMongo) en el campo Connection



Colocamos la sentencia de Mongo en el campo query y presionamos en save



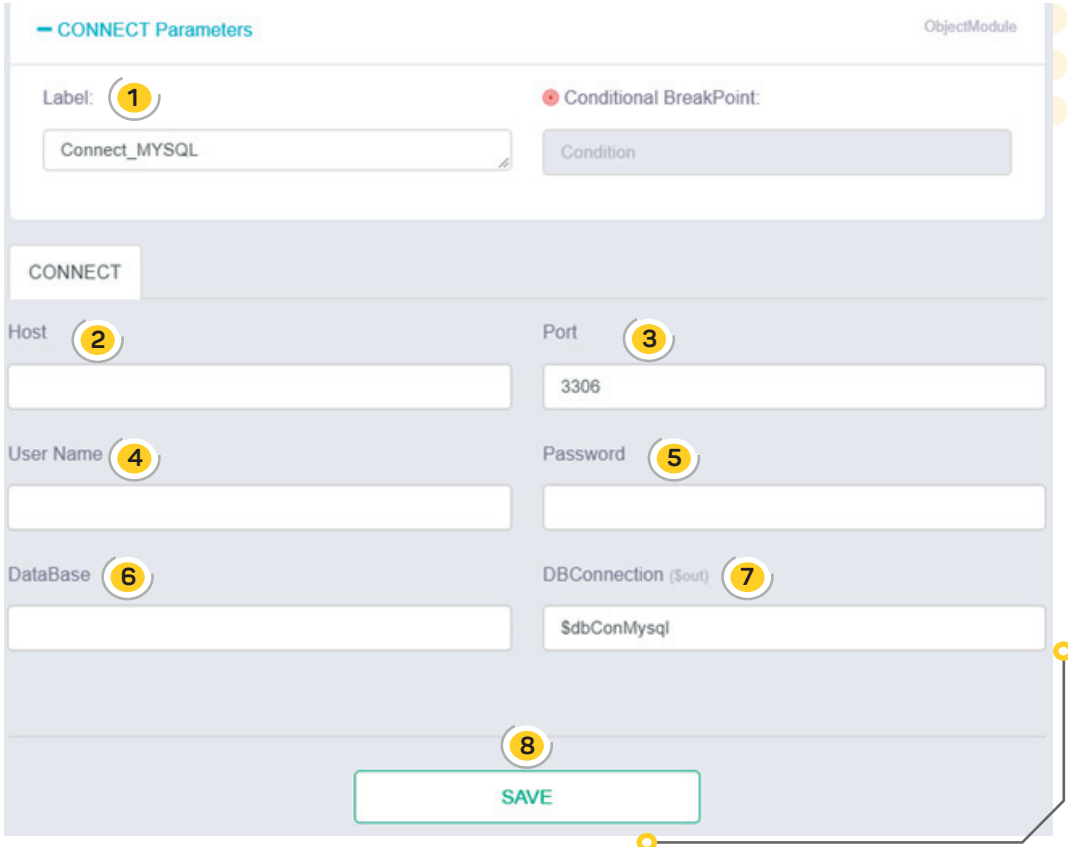
En la variable $dataMongo es donde esta almacenada la respuesta del query



2.3 Connect_MYSQL

Conexión de base de datos MYSQL
Permite las siguientes conexiones:

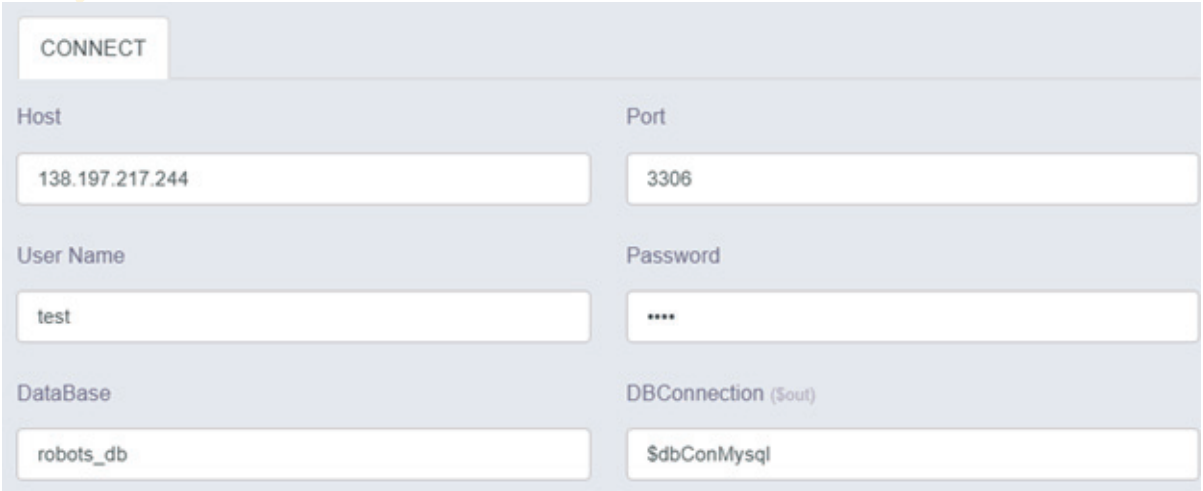
- **Success** la Conexión se completó satisfactoriamente,
- **Error:** El comando resultó en un error,



Propiedades Connect_MYSQL

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Host	Dirección de base de datos a conectar
3	Port	Puerto para conexión
4	DataBase	Nombre de usuario
5	Keep Alive	Contraseña
6	DataBase	Nombre de la base de datos
7	DBConnection	Variable de salida para la conexión
8	SAVE	Botón guardar cambios

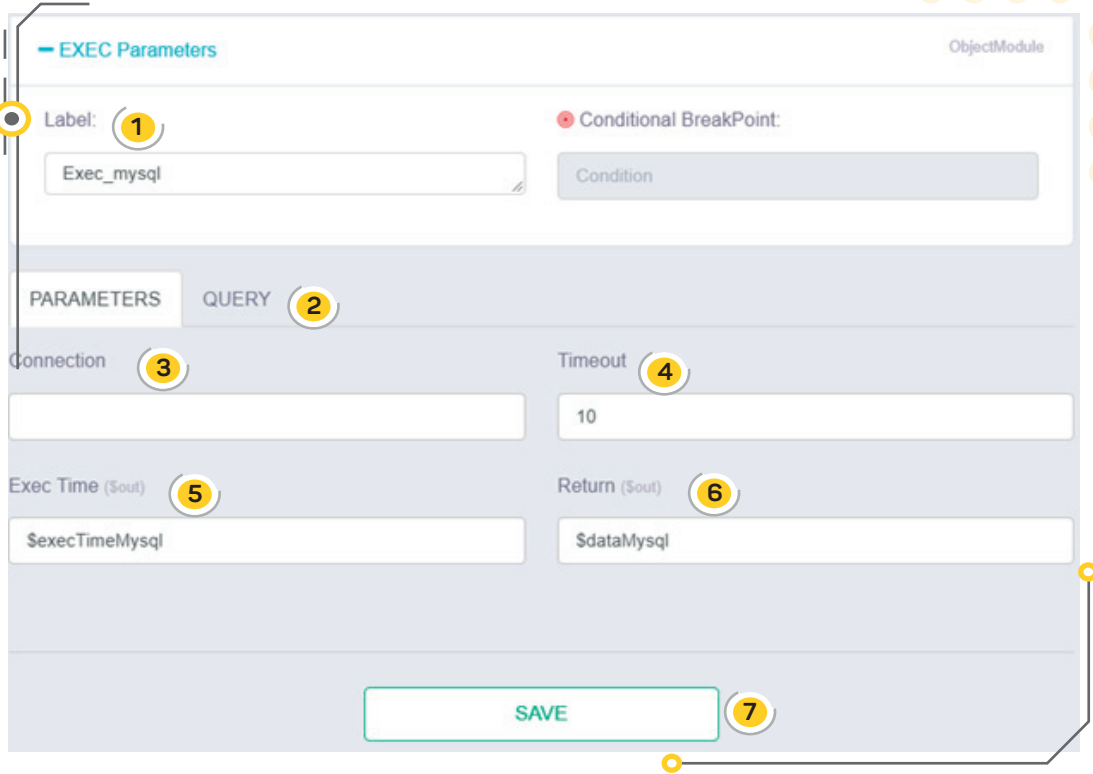
Ejemplo:
Realizamos la conexión a base de datos colocamos el Host, el Puerto, el nombre de la base de datos, usuario de base de datos, la contraseña. Variable de salida con todos los parámetros de conexión $dbConMysql



2.4 Exec_MYSQL

Módulo de Ejecución de sentencias Query en MySQL
Permite las siguientes conexiones:

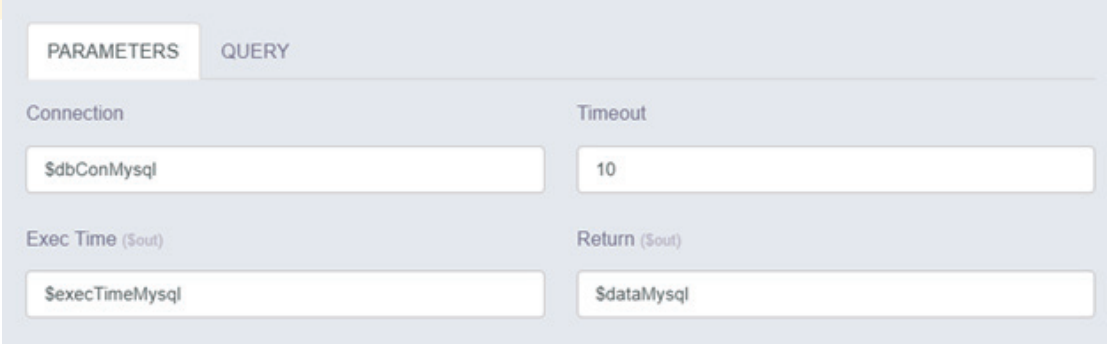
- **Success** la Conexión se completó satisfactoriamente,
- **Error:** El comando resultó en un error,
- **NoFound:** Consulta sin datos retornados



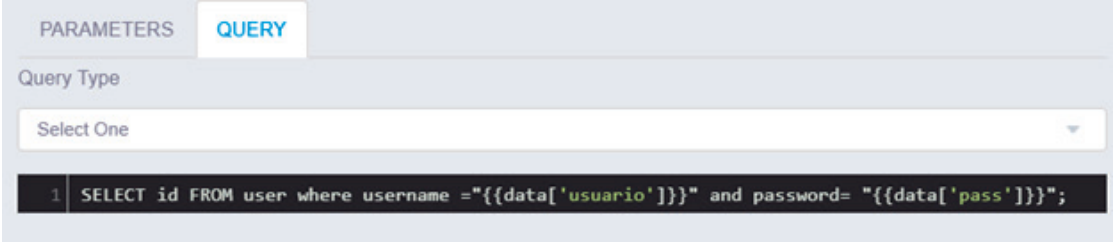
Propiedades Exec_MYSQL

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Menú (PARAMETERS/QUERY)	PARAMETERS QUERY
3	Connection	Puerto para conexión
4	Timeout	Tiempo de ejecución
5	Exec.tiem	Tiempo de ejecución consulta
6	Return	Variable de salida
7	Save	Botón guardar cambios

Ejemplo
Colocamos la variable de salida de la conexión Mysql ($dbConMysql) en el campo Connection

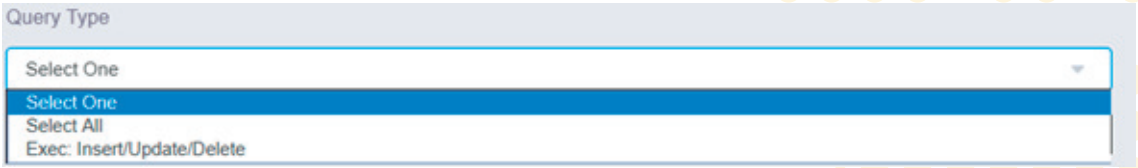


Colocamos la sentencia de MySQL en el campo query.

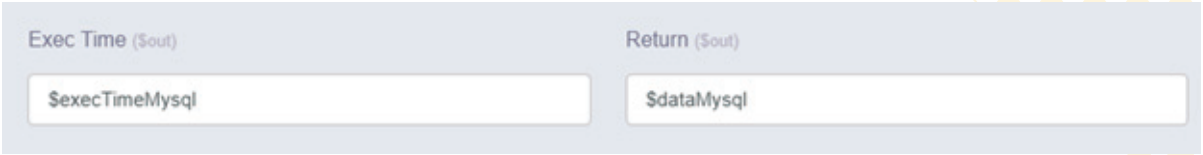


El campo Query Type Despliega un menú:

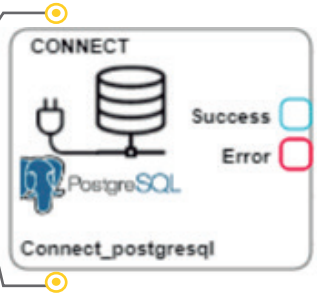
- Select one: seleccionar un campo
- Select All: Selccionar todos los campos
- Exec.Insert/Update/Delete :Ejecutar Insertar/Actualizar/Borrar



En la variable $dataMysql es donde esta almacenada la respuesta del query

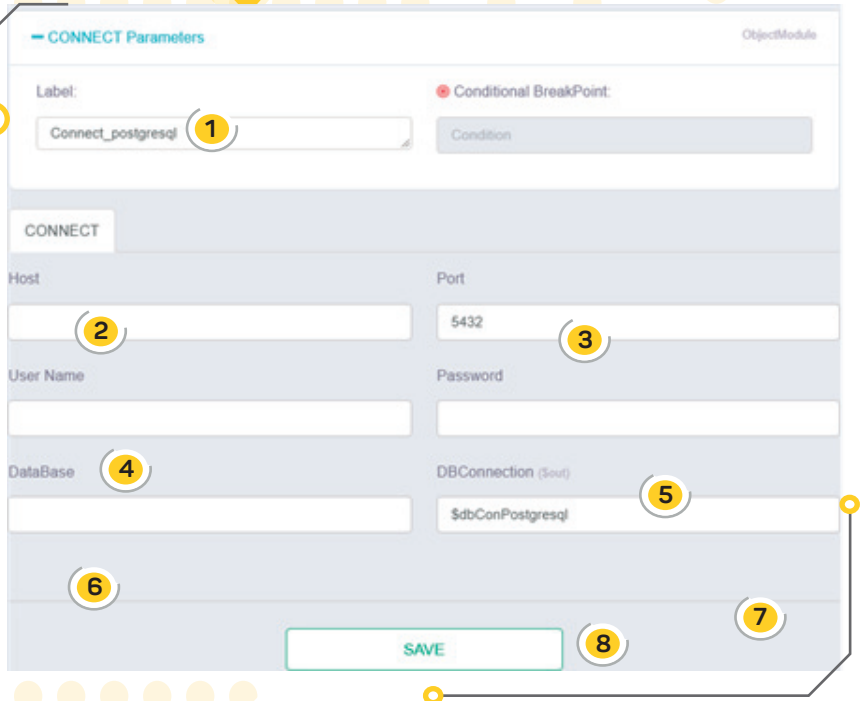


2.5 Connect_PostgreSQL



Conexión de base de datos PostgreSQL
Permite las siguientes conexiones:

- **Success** la Conexión se completó satisfactoriamente,
- **Error:** El comando resultó en un error,

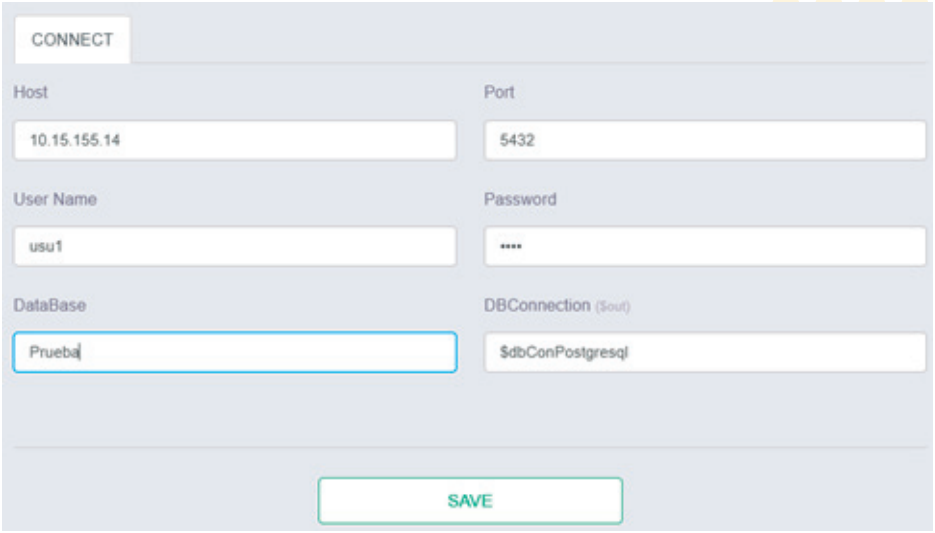


Propiedades Connect_mongoDB

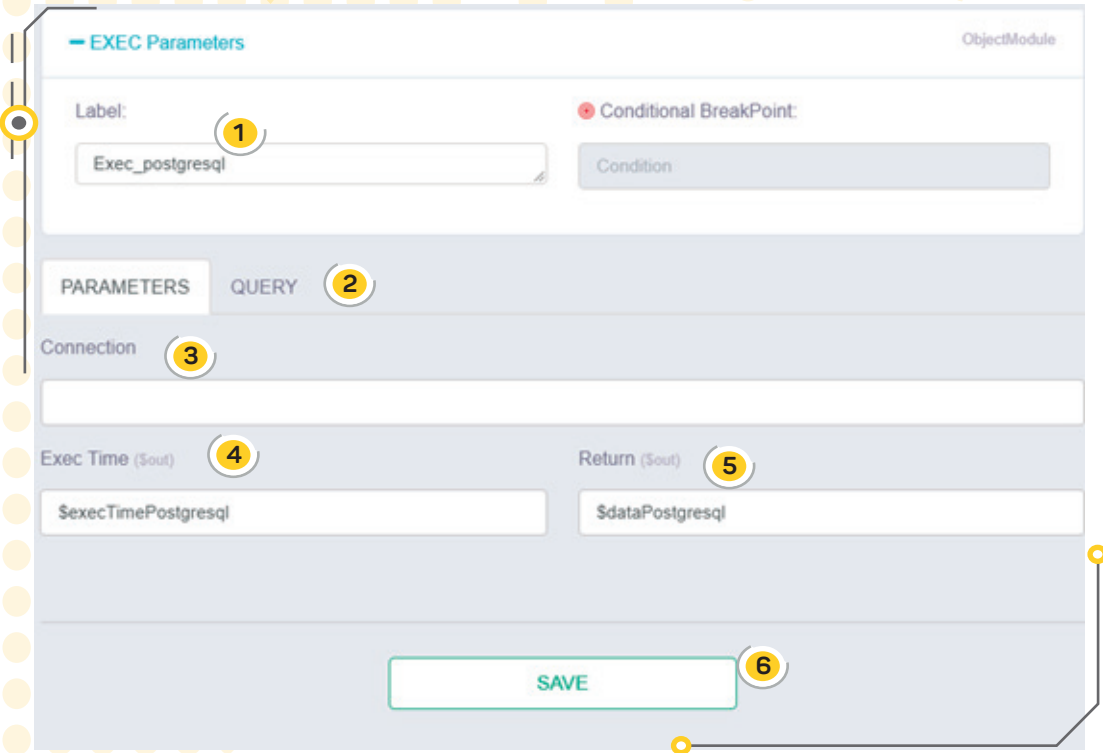
Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Host	Dirección de base de datos a conectar
3	Port	Puerto de conexión
4	User name	Usuario de conexión para base de datos
5	Password	Contraseña del usuario de base de datos
6	Database	Nombre de base de datos a conectar
7	DBConnection ($out)	Variable de salida
8	Save	Botón guardar cambios

Ejemplo

Realizamos la conexión a base de datos colocamos el Host, el Puerto, el nombre de la base de datos, usuario de base de datos, la contraseña. Variable de salida con todos los parámetros de conexión $dbConPostgresql.



Módulo de Ejecución de sentencias Query en PostgreSQL

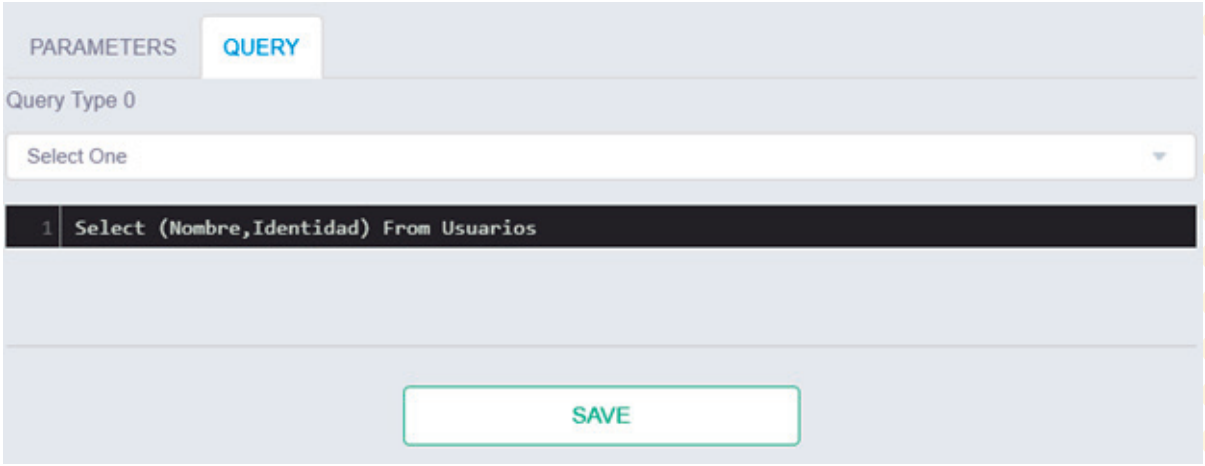


Propiedades Exec_PostgreSQL

Número	Campo	Definición
1	Label	Nombre asignado al objeto
2	Menú (PARAMETERS/QUERY)	PARAMETERS: Vista principal QUERY: consulta de BD
3	Connection	Variable conexión BD
4	Exec.Time	Tiempo de ejecución consulta
5	Return	Variable de salida
6	SAVE	Botón guardar cambios

Ejemplo

Colocamos la sentencia de PostgreSQL en el campo query y procedemos guardamos con el botón save



Manual de Usuario

Yellow Push

By: 4DEV

Designer Studio
& object builder modules
Yellow Push

Programe gráficamente en menos tiempo y con menos código uniendo cajas que a su vez contienen código seguro y probado, garantizando confianza y un mejor desempeño con código abierto para desarrollar aplicaciones web nativas y aplicaciones móviles increibles.

#YellowPushSolutions

Tel: 1 (786) 242-2224 Email: smsretail@identidadtelecom.net

www.identidadtelecom.net

IDENTIDAD TELECOM

Identidad Telecom an Identidad Technologies company