EBAVEL

Workprocess

La información de este documento está sujeta a cambios sin previo aviso. Los ejemplos de compañías, productos, personas, conceptos y cifras son ficticios. Ninguna asociación con los datos, las personas o los eventos de alguna compañía es intencional o inferida.

Fecha de última actualización 15 de junio de 2020

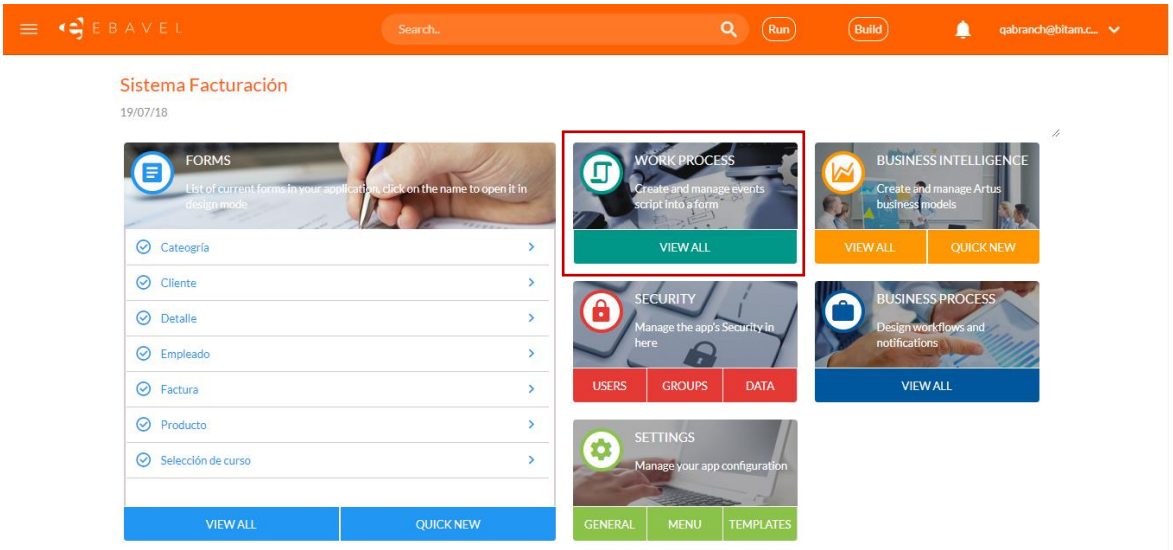
Contenido

Workprocess

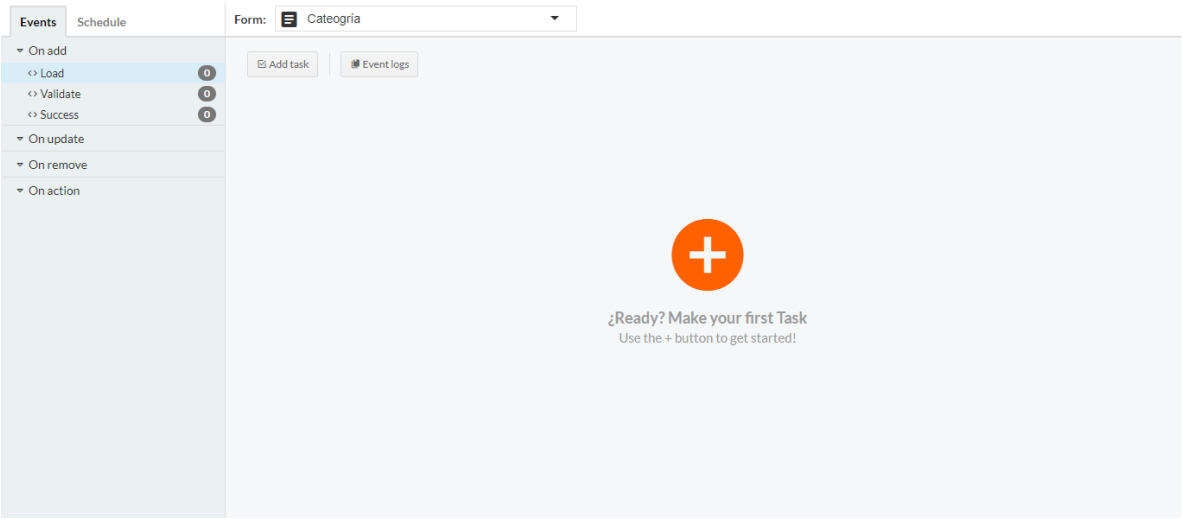
La funcionalidad de Workprocess permite crear tareas para ser ejecutados desde una forma dependiendo de la etapa (creación, edición o eliminación) de un registro. Se cuenta con 3 tipos de tarea:

- **Tarea sendEmail**. Envía un correo electrónico a uno o varios destinatarios.
- **Tarea Transfer Data.** Envía información de una forma a otra.
- **Tarea Script.** Crea un pseudocódigo para realizar una tarea más personalizada.

Para empezar a utilizarlo se colocará en el Diseñador de Aplicaciones; y este estará situado donde anteriormente estaba Batch Process.



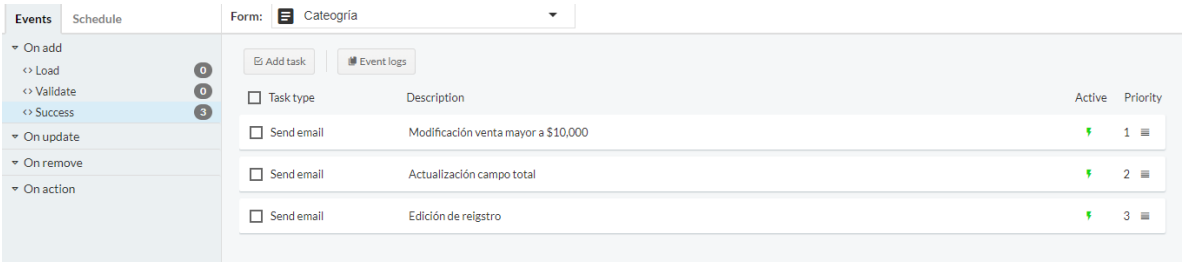
Al ingresar a la funcionalidad de Workprocess, desplegará la siguiente pantalla.



- **Form.** La forma a la que desea agregar la tarea.
- **Add task.** Para agregar una nueva tarea.
- **Event logs.** Acceso a la información básica del log.
- **Events.** Etapa en la que puede agregar una tarea.

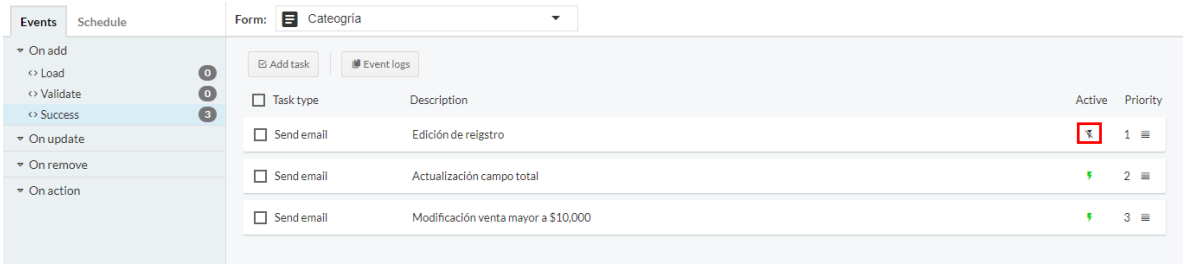
- **Schedule.** Periodo en el que se ejecutará una tarea.

Al tener varias tareas, se acomodarán por orden de prioridad, para cambiar el orden coloca el mouse en el icono, da clic y sin soltarlo lo arrastra para situarlo en el orden que desea se ejecuten sus tareas.

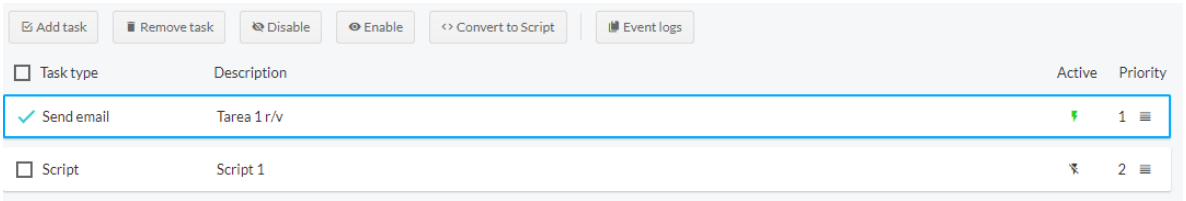


Como se puede observar, se cambiaron las prioridades de las tareas, de tal modo que "*Edición de registro*" paso a ser la primera que se ejecutará, luego "*Actualización campo total*" y por ultimo "*Modificación venta mayor a $10,000*".

Para hacer saber que una tarea esta activa el icono se mostrará en color verle, si desea desactivar una tarea, debe de dar clic al icono del rayo y este modificara su color a gris.

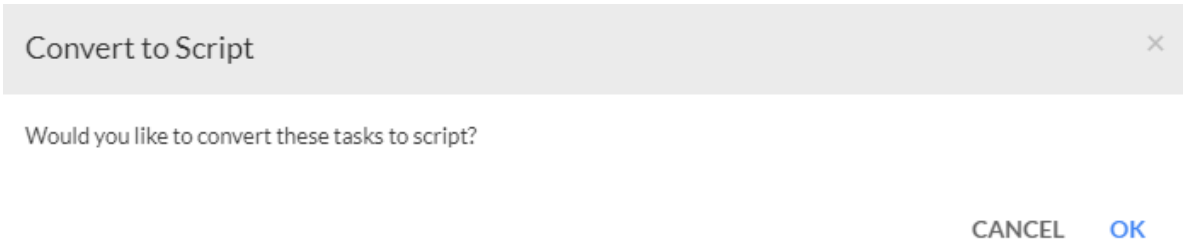


Al dar clic en el icono aparecerán cuatro iconos más.

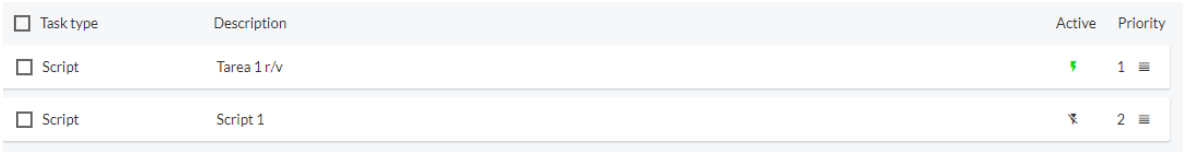


- **Remove task.** Eliminar la tarea seleccionada.
- **Disable.** Desactiva la tarea seleccionada.
- **Enable.** Activa la tarea seleccionada.
- **Convert to Script.** Convierte el sendEmail o Transfer Data a script

Para convertir una tarea a Script, selecciona la tarea y da clic en el botón **Convert to Script**, a continuación le mostrará un mensaje.



Al aceptar el mensaje, podra observar como se hace el cambio de tarea, ya que en Task type se cambiará en este caso de sendEmail a Script.



Al abrir la tarea podra ver que el sendEmail esta ahora escrito en código.

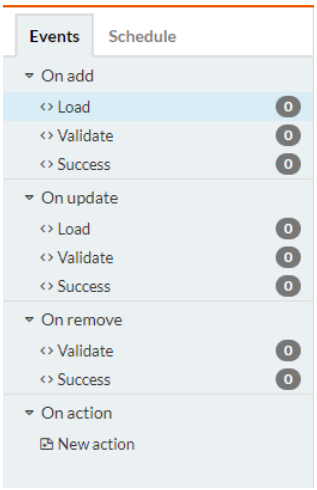
```
sendEmail([
    "to" => ["ralanis@bitam.com","rosajanet35@gmail.com"],
    "subject" => "Tarea 1",
    "message" => "<p><u></u>Se ha eliminado el registro correctamente.</p>",
    "documents" => [],
    "artus dashboards" => [],
    "artus reports" => [],
    "attach pdf record" => false
]);

alert("Se ha eliminado el registro correctamente");
```

Nota:

- Puede convertir 1 o más tareas en Script.

En la parte izquierda aparece el menú de Events, que consta de cuatro etapas:



- **On add.** Al agregar un registro a la forma.

    - **Load.** Al cargar la forma.
    - **Validate.** Antes de guardar la forma.
    - **Success.** Después de guardar la forma.

- **On update.** Al editar un registro de la forma.

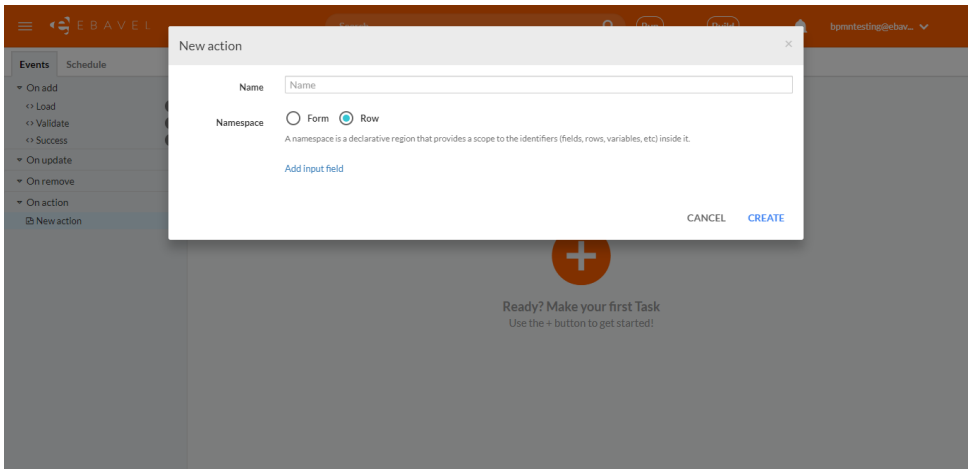
    - **Load.** Al cargar la forma.
    - **Validate.** Antes de guardar la forma.
    - **Success.** Después de guardar la forma.

- **On remove.** Al eliminar un registro de la forma.

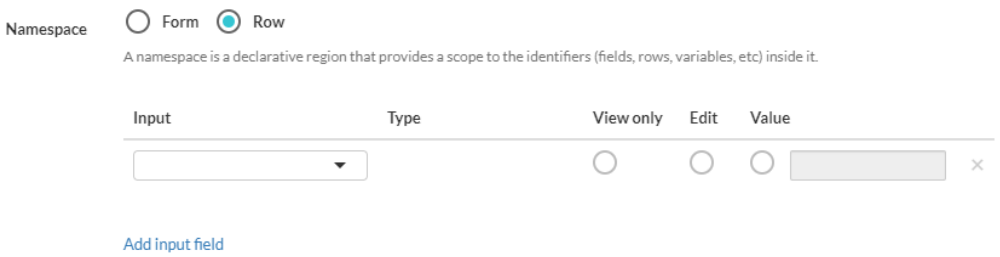
    - **Validate.** Antes de borrar el registro.
    - **Success**. Después de borrar el registro.

- **On action.** Permite crear acciones para la forma. Estas acciones funcionan justo como las acciones convencionales con la particularidad que permiten la ejecución de tareas de Work Process al resolverse.

    - **New action**. Abre el editor para crear una acción nueva.



- **Name:** Nombre de la acción.
- **Namespace:** Se refiere si la acción afecta a la forma o a la fila.

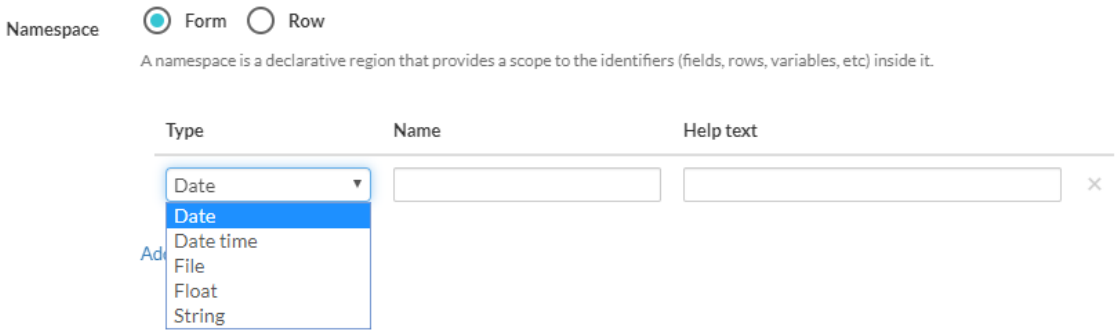


Si es Row, significa más bien que afecta a un registro en específico. Se agrega un campo de la forma el cual se vería afectado por la ejecución de acción. Las opciones disponibles son de solo lectura, edición libre o asignarle un valor en específico con el área de escritura.

Ya creada la acción se le puede agregar tareas tanto tipo Script como Send Email y estas pueden ejecutarse en cualquier momento tras activar las acciones.

En las acciones de Workprocess que son de namespace = form, se agregó parámetros, estos parámetros estarán disponibles como si fuera el 'registro' (porque en este contexto, estas acciones No tienen un registro) y serán accesibles por la variable input.

Los tipos de parámetros soportados son: ***Date, Date time, File, Float, String.***



Para soportar el parámetro *File* se agregó una función llamada File que le puedes mandar como argumento el parámetro File o un campo de tipo File (en caso de los campos File que puedes subir varios archivos, solo tomara el primero de ellos).

Las funciones disponibles del objeto File son las siguientes:

- **getContent():** Obtiene el contenido del archivo, ya sea si es binario regresara ese código binario o si es un archivo de texto, pues regresara ese contenido.
- **getExtension():** Obtiene la extensión del archivo.
- **count():** Si el archivo es un ZIP, obtendrá el número de archivos, si es cualquier otro siempre regresara el valor de 1.}

Notas:

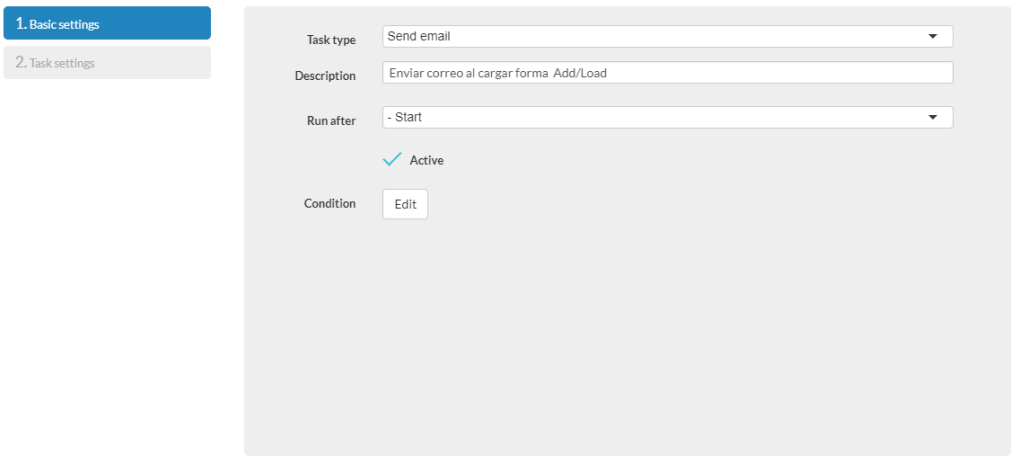
- Si el archivo es un Zip podrá ser usado en un foreach para interactuar con cada uno de los registros dentro del Zip.
- Los archivos que se suban con los parámetros no se guardan permanentemente, después de ser usados automáticamente serán eliminados.
- Tampoco se pueden insertar esos archivos en un registro de eBavel.

Tarea Send Email

1. Selecciona la forma en la que desea agregar la tarea.

2. Selecciona en el menú de Events la etapa en la que desea agregar su tarea. Da clic en el botón **Add task**.

3. A continuación, aparecerá la ventana Basic settings la cual incluye la información básica de la tarea.

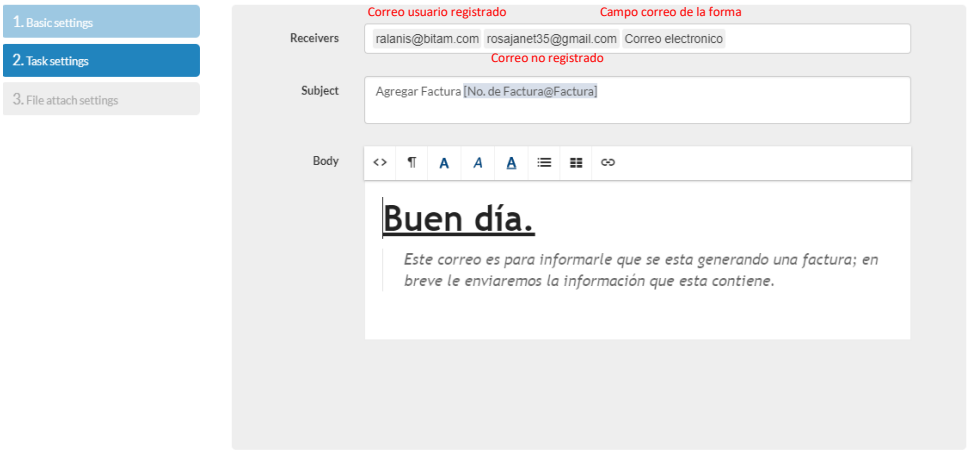


- **Task type.** Tipo de tarea que deseamos agregar (Send Email, Script o Transfer Data).
- **Description.** Nombre o descripción de la tarea (máximo 512 caracteres).
- **Run after.** Seleccionar en qué orden se ejecutarán las tareas.
- **Active.** Activar o desactivar la tarea.
- **Condition.** Condición en la que se ejecutará la tarea (opcional).

4. Después se mostrará la ventana Task settings, en la cual se definen los datos para enviar el correo.

Nota:

- Si se desea ingresar información de los campos de la forma, se debe de abrir corchetes [ ] y desplegará una lista con los nombres de los campos de la forma.



- **Receivers.** Correo(s) de la(s) persona(s) a las que se les enviará el correo. Se puede enviar un correo a:
  - Un usuario ya registrado.
  - Agregar un correo que no esté registrado escribiendo el correo de la persona y al terminar dar clic en la opción Add que se despliega.

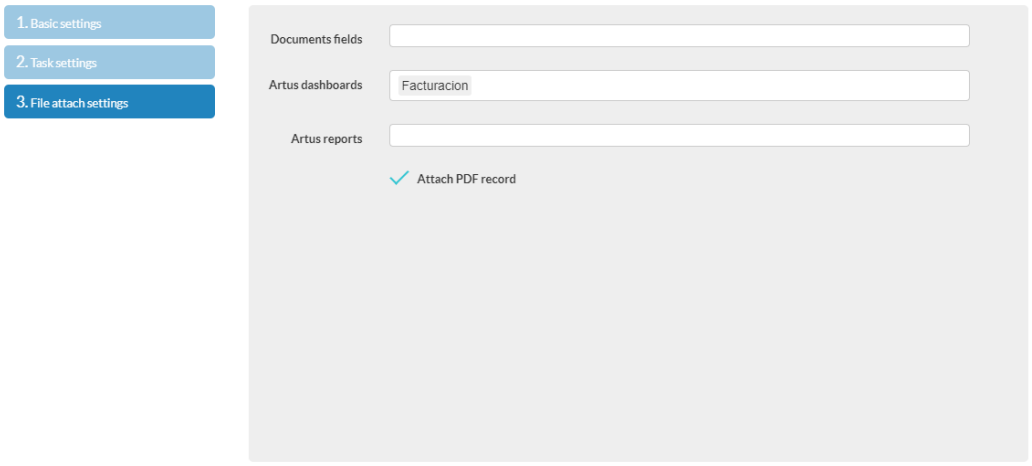
- Al correo que se esté registrando en la forma (con el capo EMAIL).

- **Subjet.** Asunto del correo.

- **Body.** Cuerpo del correo, dentro del cual se encuentran las siguientes opciones.

  - **HTML.** Formato de texto en HTML.
  - **Format.** Formato del texto.
  - **Bold.** Texto en negritas.
  - **Italic.** Texto en cursiva.
  - **Underline.** Texto subrayado.
  - **Lists.** Lista de viñetas y enumeradas.
  - **Table.** Agregar una tabla.
  - **Link.** Agregar un Link.

5. Al dar clic en next, posteriormente se presenta la pantalla File Attach settings, en la cual se agregan documentos que se anexarán al correo. (opcional).



- **Documents fields.** Agrega documentos que se tienen en la forma con el campo Document.
- **Artus dashboards.** Agrega un pdf de un dashboard de Artus.
- **Artus reports.** Agrega un pdf de Artus reports.
- **Attach PDF record.** Agrega un PDF de la forma.

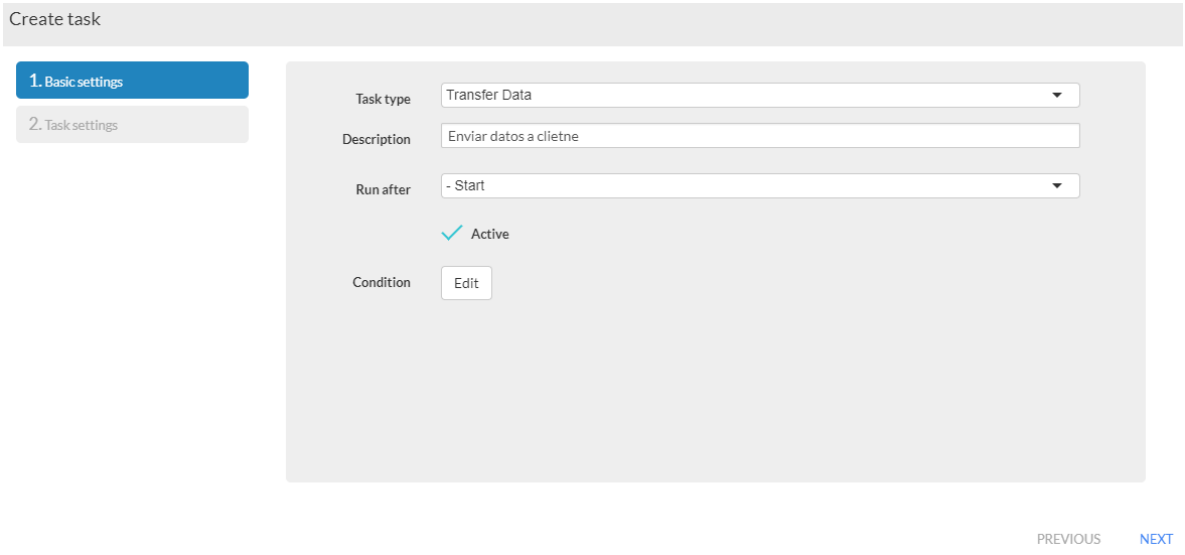
Notas:

- Se pueden agregar 1 o más campos de tipo documento en Documents fields.
- En On Add / Load no se mostrarán los campos de la forma ni podrá enviar documentos, ya que como apenas se agregará el registro, mostraría todos los campos vacíos.
- Para agregar un documento en Documents fields debe de dar clic en el campo que está en blanco y le aparecerá el nombre del campo y lo selecciona.
- Para agregar un Artus dashboard debe de dar clic en el campo en blanco y se desplegará una lista de los dashboard, puede seleccionar 1 o más.

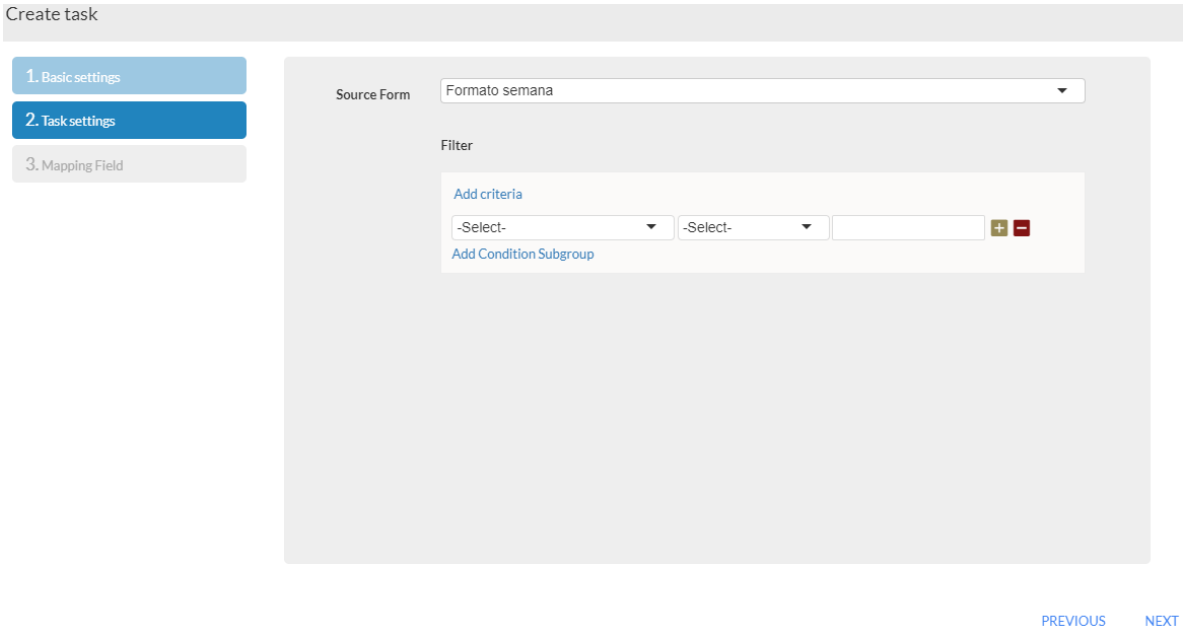
Tarea Transfer Data

Su función es que al ocurrir un evento específico en una forma en la que este se haya definido, se muevan los datos de una forma a otra forma en específico. Esto en base a un filtro y al mapeo que se le defina al transfer data.

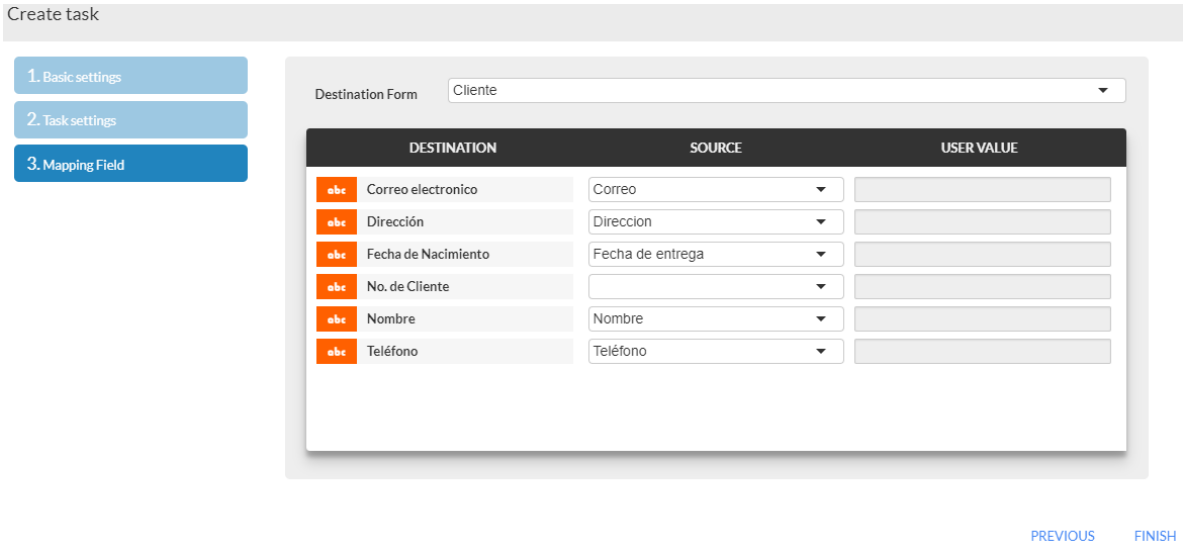
Selecciona una forma y la etapa sobre la que se ejecuta el evento, en task type selecciona Transfer Data.



A continuación, en **Source Form** debe seleccionar la forma de origen, la cual tendrá los datos que desea enviar a otra forma, puede ser la misma forma en la que se ejecutará el evento u otra diferente; opcional puede agregar una condición.



Posteriormente en **Destination Form** selecciona la forma de destino a la cual se enviaran los datos; en la tabla que aparece debajo es donde mapeara los campos de la forma destino con los de la forma de origen. En **Destination** le aparecerán los campos de la forma destino y en **Source** los campos de la forma origen y de la forma en que se ejecutará el evento.

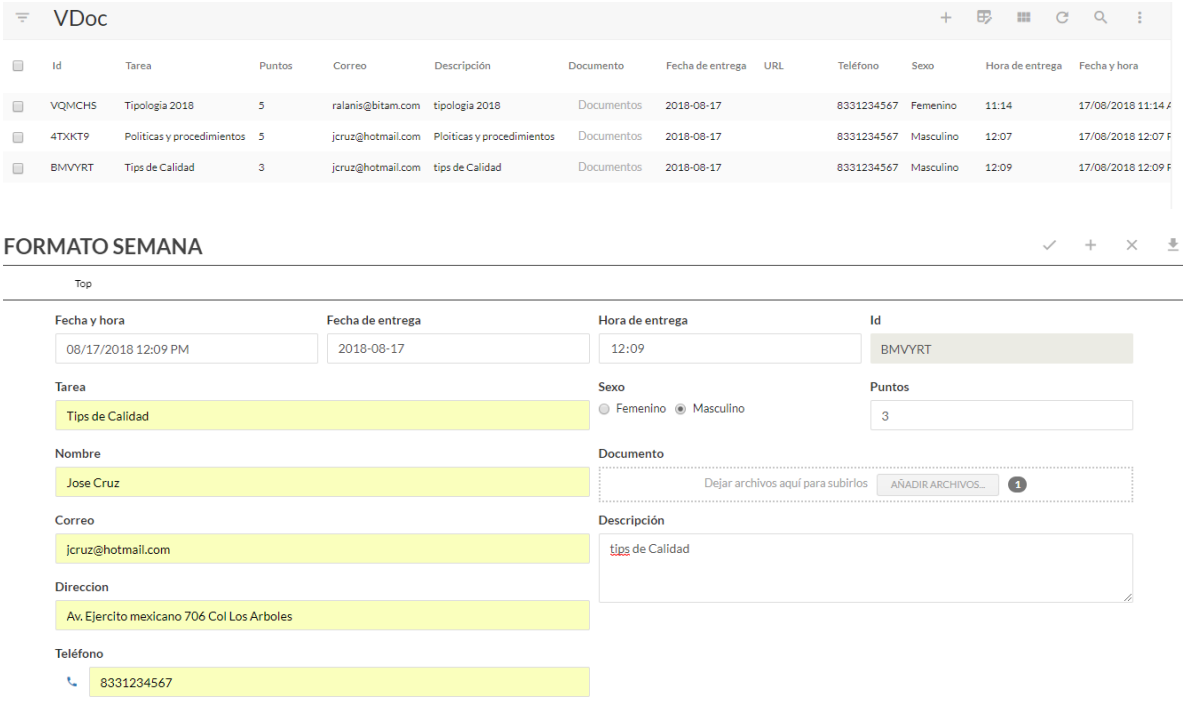


Cabe mencionar en esta parte, que en la lista oculta de campos de la forma fuente seleccionada, también aparecerá una lista bajo el encabezado *Current Section*, estos campos hacen referencia al registro que lanzó el evento de transfer y por tanto se convertirán en valores constantes cuando se ejecute la tarea. Si su forma fuente es la misma que la forma que lanzó el evento, entonces observará los mismo campos, pero sí hay diferencia usar unos u otros.

Si usted utiliza los campos de *Current Section* para mapearlos, recuerde que será solamente valores que tengan en el registro que lanzó el evento y ejecutó la tarea. Si utiliza los nombres de campo que aparecen bajo el nombre de la forma fuente, entonces se utilizarán los valores de los registros almacenados en esa forma que cumplan con los criterios definidos en la sección de **Source Form**.

Al ejecutar la aplicación debe de ir a la vista donde esta la forma de origen en este caso la vista es

“VDoc” y el transfer data se ejecutará al modificar un registro y guardarlo.



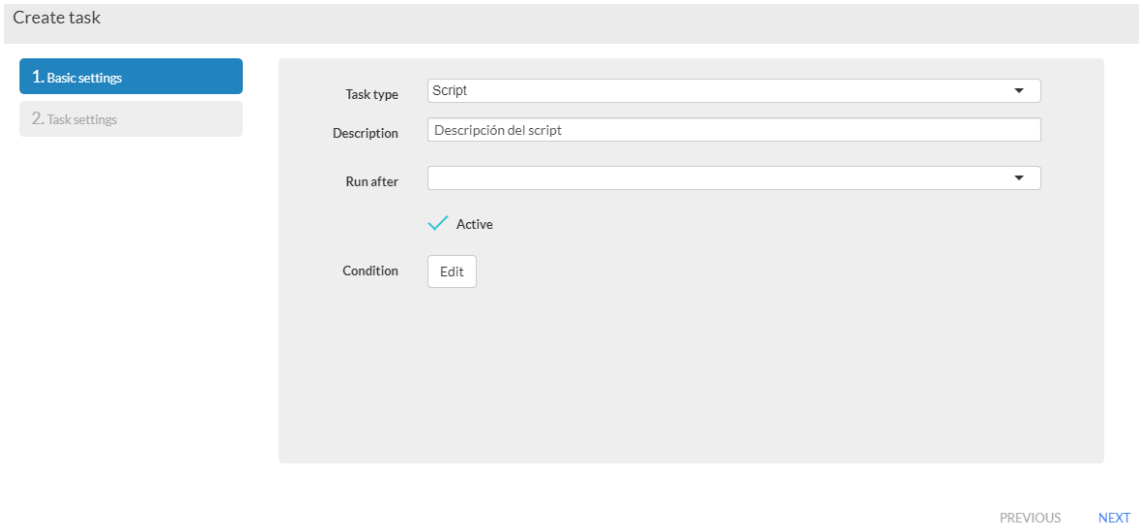
Al guardar los cambios, debe de ir a la vista de destino en la cual se visualizará el registro que se modificó anteriormente.

Cliente

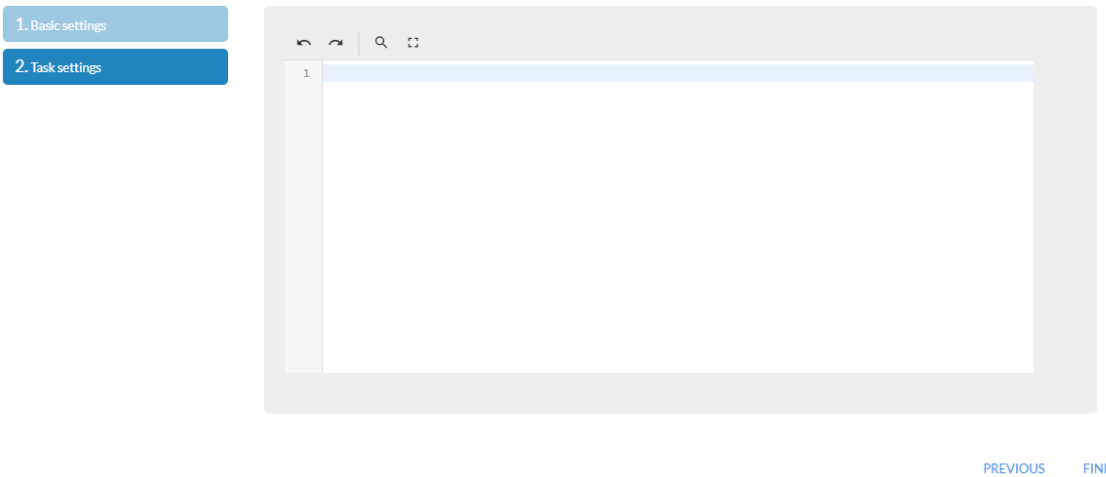
No. de Cliente	Nombre	Dirección	Teléfono	Correo electronico	Fecha de Nacimiento
BES3R4	Jose luis	Av. Ejercito mexicano 706	8331234567	jcruz@hotmail.com	17/08/2018
CM4125	Jose luis	Calle Benito Juarez 2143	8331234567	jcruz@hotmail.com	17/08/2018
MNSKF6	Camila Garza	Carlos Medina 986	1234567	cgarza@gmail.com	17/08/2018
CYKDHT	Jose luis	Calle primera 129	8331234567	jcruz@hotmail.com	17/08/2018
SCIA7M	Mariana Arellano	Av Ejercito Mexicano 987	1234567	marellano@bitam.com	17/08/2018
HBY1UM	Jose luis	Av. Ejercito mexicano 706	8331234567	jcruz@hotmail.com	17/08/2018
NXFD7Y	Julia Peñaloza	Av. Ejercito mexicano 706	1234567	jpenaloza@gmail.com	17/08/2018
36GMSF	Jose luis	Av. Ejercito mexicano 706	8331234567	jcruz@hotmail.com	17/08/2018
JFY275	Rosa Janet	Av. Ejercito mexicano 706	1234567	ralanis@bitam.com	17/08/2018
HSSPGY	Jose luis	Av. Ejercito mexicano 706 Col Los Arboles	8331234567	jcruz@hotmail.com	17/08/2018
6SPP4U	Jose Cruz	Av. Ejercito mexicano 706 Col Los Arboles	8331234567	jcruz@hotmail.com	17/08/2018

Tarea Script

Para agregar un script dar clic en el botón **Add task**, en Task type seleccionar Script, agregar una descripción.



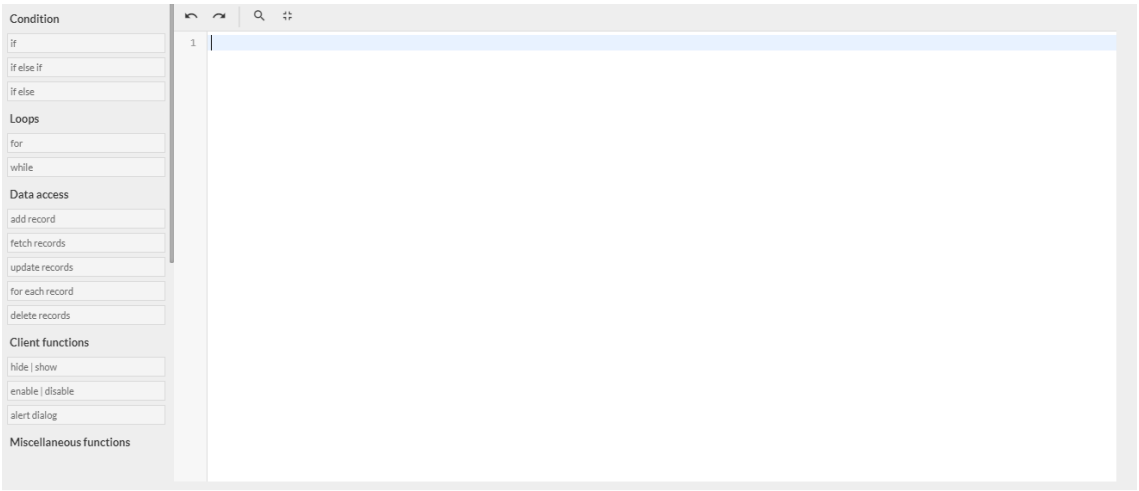
Al dar clic en next, aparecerá la siguiente ventana.



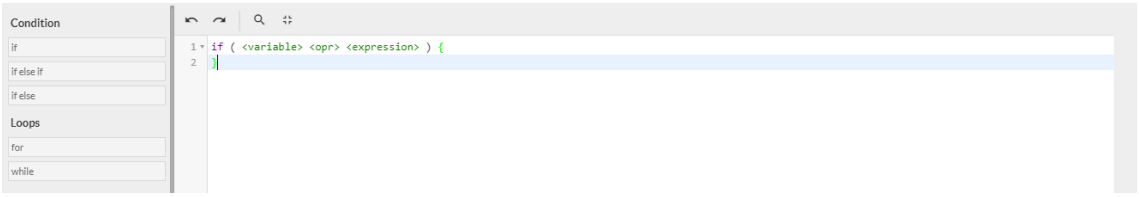
- **Undo.** Se deshará la última acción que se haya realizado.
- **Redo.** Se cancelará el último "Deshacer" que se haya realizado y se sustituirá la acción que se haya deshecho.

- **Search.** Buscar una palabra en el script.
- **Full screen.** Pantalla completa.

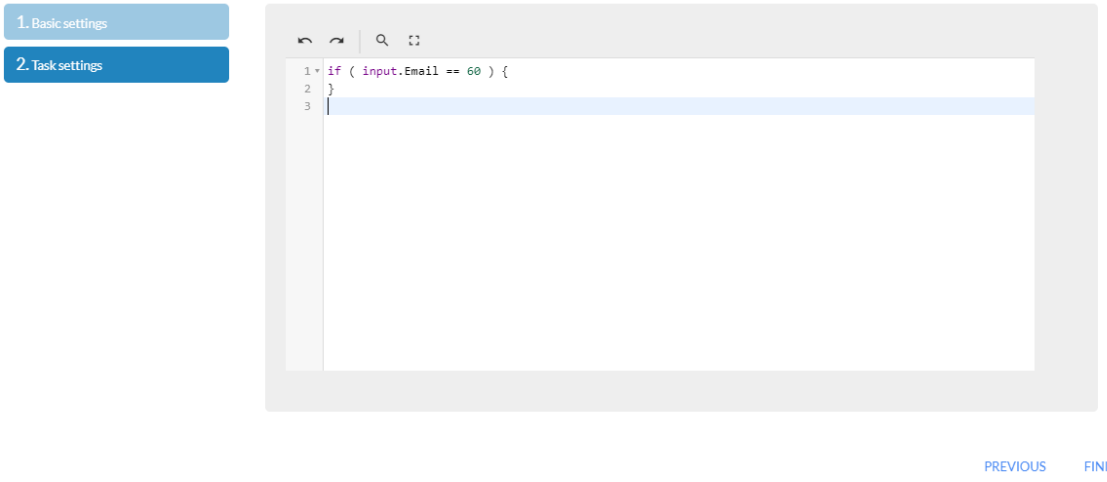
Al ejecutar la pantalla completa estará disponible en el lateral izquierdo un panel con las funciones disponibles actualmente.



Para utilizar las funciones solo debe arrastrarlas al editor.



Como puede notar, las variables vienen en color verde, solo se debe de dar clic en ellas para que muestren los datos de la forma que desea utilizar, en caso de que usted desee agregar un valor fijo alfanumérico solo debe de agregarlo entre comillas simple o dobles, y en caso de un numero agregarlo sin comillas.



Al terminar su script solo debe de dar clic en el botón Finish.

Para agregar variables debe de agregar en nombre de la variable y lo que desea asignarle a la variable, puede ser un valor alfanumérico, numérico o un campo de la forma.

```
myvariable= 5;
myvariable2="hola";
myvariable3= input.Email;
```

Nota:

- Para agregar datos de campos múltiples Choice se deben de agregar entre corchetes [ ] y separados por comas (ejem. Variable= [opcion1, opción2, opciónN]).

Obtener un conjunto de registros de una forma

Para obtener un conjunto de registros de una forma podrá usar la sintaxis:

<*variable*> = **from** <*form*> **select** <*modifier*> <*field*> **where** <*field*> <*opr*> <*expression*>

Podrá utilizar la opción Fetch record para obtener esta sintaxis arrastrándola a la zona del script.

Lo que deja en la variable es un apuntador al registro para que pueda utilizarlo para obtener los valores de los campos.

Ejemplo:

vParcialidad = **FROM** Parcialidad **SELECT** Parcialidad_ID, Saldo;

Regresará un objeto con las propiedades de *Parcialidad_ID, Saldo*.
Si utiliza la sección Where para obtener un conjunto de registros más específico, por ejemplo:

vParcialidad = **FROM** Parcialidades **SELECT** Parcialidad_ID, Saldo
WHERE Saldo > 0 AND Cliente = “Bitam de México”;

Regresará un objeto con la información de Parcialidad_ID, Saldo, Cliente.
Si desea que solamente le traiga un solo registro, entonces deberá añadir la cláusula ***one***, por ejemplo:

myvar = **from** Alumno **select one** Nombre, Email, Estado, Fecha **where** Promedio > 80;
alert(myvar);

Si desea que le traiga todos los registros que cumplan con la condición entonces puede usar **all**.

myvar = **from** Alumno **select all** Nombre, Promedio **where** Promedio == 80;

Nota:

- Si se utiliza la función Alert y el variable objeto como parámetro, se mostrará una concatenación de los valores de los campos separados por coma, para que pueda visualizar la información al menos del primer registro que regrese el conjunto.

Funciones condicionales

If

Ejecuta el código si la condición es verdadera.

If(<variable> <opr ><expression>)
{Instrucciones}

```
if ( input.Promedio > 60 ) {
  insert into Capacidad
  (
    Salon = "S001"
    Nombre = input.Nombre
    Promedio = input.Promedio
  );
}
```

If else

Ejecuta un código si una condición es verdadera y otro código si la condición es falsa.

if(<variable> <opr> <expression>)
{Instrucciones}
Else
{Instrucciones}

```
if ( input.Promedio < 60) {
   alert("Promedio menor a 60");
} else {
  alert("Promedio mayor o igual a 60");
}
```

If else if

Ejecuta diferentes códigos para más de dos condiciones

if(<variable> <opr> <expression>)
{Instrucciones}
else if(<variable> <opr> <expression>)
{Instrucciones}

```
if ( input.Promedio < 60 ) {
  alert("Promedio menor que 60");
} else if ( input.Promedio > 60 ) {
  alert("Promedio mayor que 60");
}
```

Nota:

- No se puede utilizar If anidados.

Funciones loop

For

Se usa cuando sabes de antemano cuántas veces debe ejecutarse el script.

for(<myvariable> = <expression>; <myvariable> <opr> <expression>;<expression>) {Instrucciones}

```
for ( var =0; var < 5; var++ ) {
  alert(var);
}
```

While

Ejecuta un bloque de código mientras que la condición especificada es verdadera.

while(<variable> <opr> <expression>) {Instrucciones}

```
var=input.Promedio;
while ( var < 60 ) {
  alert(var);
  var=var+10;
}
```

Break

Puede utilizar la instrucción break para interrumpir un ciclo.

Ejemplo:

```
vPago = 10;
While i < 10
{
vPagoParcial = vSaldo - vPago;
if (vPagoParcial == 0)
{
break;
}
vPago = vPago + 10;
i = i + 1;
}
```

Funciones de acceso de datos

Add Record

Se utiliza para insertar nuevos registros en un formulario.

insert into <form>(
<field> = <expression>
<field> = <expression>
<field> = <expression>);

- **Form:** Forma a la que se va a agregar el registro.
- **Field:** Campo de la forma a la que se va a agregar el registro.
- **Expression:** Campo de la forma donde se está creando la tarea, alfanumérico o numérico.

```
insert into Capacidad
(
    Salon = "S001"
    Nombre = input.Nombre
    Promedio = input.Promedio
);
```

Nota:

- La inserción de nuevos registros mediante eventos, diagramas de Workflow, importación de Excel o incluso tareas de Workprocess ***No causara que se disparen tareas de Workprocess***. La excepción a esta regla ocurre cuando se utiliza una forma de eBavel como destino de datos en eBavel Forms.

Fetch Record

Se utiliza para seleccionar datos de una aplicación

<variable> = **from** <form> **select** <modifier> <field> **where** <field> <opr> <expression> **order by** <field> desc/asc

- **Variable:** Variable en la que se guardará la información.
- **Form:** Forma en la cual se va a buscar el o los registros
- **Modifier:** Para traer la información de la forma. Limit one row (para un solo registro), All rows(todos los registros)
- **Field:** Campo de la forma.
- Opr: igual(==), diferente (!=), mayor que (>), menor que (<), mayor o igual a (=>), menor o igual a (<=).
- **Expression:** Campo de la forma, número o alfanumérico.
- **order by:** utilice esta para indicar que el resultado deberá estar ordenado por uno o mas campos en forma ascendente o descendente.

```
myvar = from Alumno select all Nombre where Promedio > 90;
alert myvar.Nombre;
```

Se utiliza **myvar.campo** para obtener la información del campo que se desea mostrar, ya que en caso de solo tener **myvar** es un record set.

Update Record

Se utiliza para modificar los registros existentes en una forma.

update <form>(
<field> = <expression>
<field> = <expression>
<field> = <expression>)
where <field> <opr> <expression>;

- **Form:** Forma en la que se va a modificar el registro.
- **Field:** Campo de la forma en la que se va a modificar el registro.
- **Expression:** Campo de la forma donde se está creando la tarea, alfanumérico o numérico.

```
update Alumno
(
    Estado = "Baja"
) where Promedio < 60;
```

Actualizar datos utilizando los valores guardados

Puede ser que sea necesario cambiar el valor de un campo a través de una expresión aritmética que incluya el valor guardado del mismo campo.

Para hacer eso es necesario añadirle como prefijo el carácter “@” al nombre del campo, de esa forma eBavel lo diferenciará de una variable de su script.

La sintaxis sería como sigue:

```
update <Form>
        (
                Field1 =@Field1 <operand> <Value or Variable>
        )
where Field2 == <Value or Variable>;
```

Por ejemplo, suponga que quiere actualizar el saldo de una cuenta contable con el valor de un ajuste, en este caso será necesario utilizar el valor guardado del saldo y restarle o sumarle el ajuste. A continuación se muestra el código de este ejemplo:

```
update Saldos
    (
        Saldo = @Saldo + Ajuste
    )
where cuenta == vCuenta;
```

Nota:

- No se debe dejar espacio entre un campo y otro incluido en el estatuto o después o antes de los paréntesis.

For Each

Repite un grupo de instrucciones para cada elemento de una colección.

foreach(<variable> **in from** <form> **select** <modifier> <field>
where <field> <opr> <expression>) {Instructiones}

- **Variable:** Nombre de la variable donde se guardara la información.
- **Form:** Forma en la cual se va a hacer la búsqueda.
- **Modifier:** Para traer la información de la forma. Limit one row (para un solo registro), All rows(todos los registros)
- **Field1:** Campo de la forma en el cual se va a hacer la búsqueda.
- **Field2:** Campo de la forma.
- **Expression:** Campo de la forma, número o alfanumérico.
- **Opr:** igual(==), diferente (!=), mayor que (>), menor que (<), mayor o igual a (=>), menor o igual a (<=).

```
foreach( myvar in from Alumno select one Nombre where Promedio > 95 ) {
  alert(myvar.Nombre);
}
```

Se utiliza **myvar.campo** para obtener la información del campo que se desea mostrar, ya que en caso de solo tener **myvar** mostraría toda la información del registro.

Delete Record

Se utiliza para eliminar registros de una forma.

delete <form> **where** <field> <opr> <expression>;

- **Form:** Forma en la cual se va a hacer la búsqueda.
- **Field:** Campo de la forma.
- **Opr:** igual(==), diferente (!=), mayor que (>), menor que (<), mayor o igual a (=>), menor o igual a (<=).
- **Expression:** Campo de la forma, número o alfanumérico.

```
delete Alumno where Promedio < 60;
```

Funciones de cliente

Estas funciones son visibles al estar en la etapa Load.

Hide | Show

Ocultar o mostrar un campo de la forma.

<hide_show> <field>;

- **hide_show:** hide for field (mostrar) / show for field (ocultar).
- **Field:** Campo de la forma.

```
hide Email;
show Fecha;
```

Enable | Disable

Habilitar o deshabilitar un campo de la forma.

<enable_disable> <field>;

- **Enable_disable:** enable for field (habilitar) / disable for field (deshabilitar).
- **Field:** Campo de la forma.

```
enable Id;
disable Email;
```

Alert Dialog

Muestra una alerta con un mensaje específico.

alert <expression>;

- **Expression**: mensaje que se va a mostrar.

```
alert "Registro guardado";
```

Alert “Texto”;	Mostrar texto.
Alert input.nombre_campo;	Mostrar un campo de la forma.
anombre = input.nombre_campo; alert anombre;	Mostrar una variable.

Funciones varias

Last Insert Unique Id

Devuelve el primer valor de id que fue establecido por la instrucción de inserción más reciente. Para utilizarla se debe de agregar primero un insert.

```
insert into Alumno
(
    Nombre = "Prueba"
    Promedio = 60
    Sexo = "Femenino"
  Email="ralanis@bitam.com"
  Estado="Alta"
);
myvar=lastUniqueID();
alert myvar;
```

Nota:

- El campo Id no debe de tener mascara ya que no lo soporta.

Send Email

Esta función permite enviar un correo electrónico incluyendo información de la forma, documentos adjuntos o elementos de Artus. La misma recibe un arreglo como argumento; éste puede contener las siguientes definiciones:

- **To:** Dirección(es) de correo(s) electrónico(s). El separador, de necesitarlo, es una coma.
- **Subject:** Asunto del correo.
- **Message:** Cuerpo del correo. Puede incluir código HTML.
- **Documents:** Campo(s) tipo Documento de la forma que se deseen adjuntar al correo. El separador, de necesitarlo, es una coma.
- **Artus dashboards:** Identificador(es) de Artus Dashboards que se deseen adjuntar al correo. El separador, de necesitarlo, es una coma.
- **Artus reports:** Identificador(es) de Artus Reports que se deseen adjuntar al correo. El separador, de necesitarlo, es una coma.
- **Attach PDF records:** Permite adjuntar un PDF con el registro. Los valores permitidos son: TRUE o FALSE.

Ejemplo:

```
sendEmail([
    "to" => ["correo.uno@bitam.com", "correo.dos@bitam.com"],
    "subject" => "Notificación de prueba",
    "message" => "<p>Este es un mensaje de prueba.</p>",
    "documents" => [input.Orden_de_compra],
    "artus dashboards" => ["29003","58"],
    "artus reports" => ["9","3"],
    "attach pdf record" => true
]);
```

Send SMS

Esta función permite para enviar SMS a teléfonos móvil a través del Workprocess.

Cada envío de SMS tiene un costo extra, es por eso que se activara por repositorio con un número determinado de 'créditos' (SMS) disponibles. Una vez que se termine los créditos la función comenzara a regresar false.

Ejemplo usando número telefónico fijo:

```
mensaje = input.Mensaje;

numeros = ['8194549176','7382816395'];
enviar = sendSMS(["numbers" => numeros, "message" => mensaje]);

alert(enviar);
```

Ejemplo usando el campo del número telefónico:

```
numero = input.Telefono;
mensaje = input.Mensaje;

enviar = sendSMS(["numbers" => numero, "message" => mensaje]);

alert(enviar);
```

Notas:

- No se valida que el teléfono sea correcto o que exista.
- No valida la longitud del mensaje, el límite de mensaje es de 160 caracteres. Si el mensaje es más largo que esa cantidad de caracteres se cortara.
- Siempre que logre enviar uno o más SMS al número o números proporcionados la función regresara *true*. En caso de fallo por que le hace falta créditos o cualquier otro tipo de error regresara *false*.
- En modo desarrollo no enviara mensajes SMS y la función siempre regresara *True*. Solo en producción es cuando se enviaran los mensajes.

Simple XML

Esta función permite procesar un archivo XML, ya sea que provenga de un campo Documento, File, cadena de texto o de una petición a un WebService.

Ejemplo usando un campo de la forma:

```
contenidoXML = simpleXML( input.CampoDeLaForma);
if ( contenidoXML ) {
    valorAtributo = contenidoXML.xpath('/NombreDelNodo').getAttribute('NombreDelAtributo');
    alert(valorAtributo);
}
```

Ejemplo usando una petición a un WebService:

```
archivoXML  = HTTPRequest('http://servidor/documento.xml', ['method' => 'GET']);
contenidoXML = simpleXML(archivoXML);
if (contenidoXML){
    valorAtributo = contenidoXML.xpath('/NombreDelNodo').getAttribute('NombreDelAtributo');
    alert(valorAtributo);
}
```

Generate PDF

Esta función permite generar un archivo PDF mediante un script de Workprocess.

Los parámetros de entrada de la función son los siguientes:

- **BodyPDF:** String que contiene el contenido del documento PDF a generar.
- **HeaderPDF:** String que contiene la cabecera o header del PDF. Este parámetro es opcional.
- **FooterPDF:** String que contiene el pie de página o footer del PDF. Este parámetro es opcional.

```
pdf= generatePDF( input.rich_text, input.texto, input.alfanumerico);
```

Notas:

- Al usar campos de la forma para el parámetro BodyPDF, es muy recomendable utilizar el campo Rich Text debido a su compatibilidad con texto HTML.

El retorno de esta función será un objeto tipo File que puede ser manejado por otras funciones de Workprocess, este documento se generará en el folder de archivos temporales de eBavel.

Las manipulaciones que son posibles realizar al objeto File resultante hasta este punto son:

- Guardarlo en un campo Document de un registro mediante asignación directa del campo (input.Document = pdf) o mediante las funciones de manipulación de datos (Update/Insert).

```
pdf= generatePDF( input.rich_text, input.texto, input.alfanumerico);
input.documento= pdf;
```

- Enviarlo como adjunto vía e-mail con la función sendEmail de Workprocess.

```
sendEmail([
    "to" => [input.correo],
    "subject" => "TITULO",
    "message" => "<p>Mensaje importante.</p>",
    "documents" => [input.documento],
    "artus dashboards" => [],
    "artus reports" => [],
    "attach pdf record" => false
]);
```

HttpRequest

Esta función permite realizar peticiones a la red a servicios REST API o sitios web. La misma recibe dos argumentos: la URL del servicio y un arreglo de parámetros de la petición. Éste arreglo de parámetros puede contener las siguientes definiciones:

- **Data:** Array de datos que se enviarán al servidor. Estos datos se pueden convertir en String en una petición GET o enviarse por POST como un formulario. Si el contentType es de tipo JSON se enviarán en formato JSON.
- **DataType:** De forma predeterminada detectará el tipo de respuesta enviada por el servidor y preparará los datos, pero en caso de que no reciba los HEADERS correctos, aquí se podrá forzar el formato de la respuesta. Dos valores disponibles son: TEXT o JSON.
- **Method:** El método HTTP para usar: GET, POST, PUT, DELETE o PATCH.
- **ContentType:** El tipo de contenido a enviar al servidor. De manera predeterminada será como si lo enviara un formulario. Para enviar JSON debe establecerse el valor de application/json.
- **Username:** El nombre de usuario si el servicio requiere autentificación básica.
- **Password:** La contraseña del usuario.

Ejemplo:

```
clavegenerada = httpRequest("https://jsonplaceholder.typicode.com/posts/1", [
    "data" => ["Folio" => input.id_unico],
    "contentType" => 'application/json',
    "dataType" => 'text',
    "method" => "PUT",
    "username" => "wbadmin",
    "password" => "wbadmin"
]);
```

Cancel Submit

Esta función es visible al estar en la etapa Validate.

Cancela él envió de la forma.

```
if ( input.Promedio < 60 ) {
  cancel submit;
}
```

Funciones de fecha

Current Date

Devuelve la fecha actual con el formato 'YYYY-MM-DD'

```
mydate=currentDate();
alert mydate;
```

Current Date Time

Devuelve la fecha y hora actual con el formato 'YYYY-MM-DD' hh:mm:ss

```
mydate=currentDateTime();
alert mydate;
```

Date

Devuelve una fecha con un formato especificado.

date(string formato, fecha opcional)

- **String Formato:** Cualquier formato que se muestra más adelante en una tabla.
- **Fecha Opcional:** Campo date o datetime, función currentDate o currentDateTime, o un string con la fecha en formato estándar ('YYYY-MM-DD HH:mm:ss') o un timestamp (numérico).

```
mydate=date("d",currentDate());
```

Formato	Descripción	Ejemplo
	Día	
d	Día del mes, 2 dígitos con ceros iniciales.	*01* a *31*
D	Nombre del día con abreviatura de 3 letras.	*Mon* hasta *Sun*
j	Día del mes sin ceros iniciales.	*1* a *31*
***l* ('L' minúscula)**	Nombre del día de la semana.	*Sunday* hasta *Saturday*
N	Representación numérica del día de la semana. (ISO-8601)	*1* (para lunes) hasta *7* (para domingo)
S	Sufijo ordinal inglés para el día del mes, 2 caracteres.	*st*, *nd*, *rd* o *th*. (se puede utilizar en conjunto con el formato j)
w	Representación numérica del día de la semana.	*0* (para domingo) hasta *6* (para sábado)
z	El día del año (comenzando por 0).	*0* hasta *365*
	Semana	
W	Número de la semana, las semanas comienzan en lunes. (ISO-8601)	*42* (la 42ª semana del año)
	Mes	
F	Nombre del mes en inglés.	*January* hasta *December*
m	Número de mes con ceros iniciales.	*01* hasta *12*

M	Nombre del mes con abreviatura de 3 letras.	*Jan* hasta *Dec*
n	Número de mes sin ceros iniciales.	*1* hasta *12*
t	Número de días que tiene el mes dado.	*28* hasta *31*
	Año	
L	Si es un año bisiesto	*1* si es bisiesto, *0* si no.
o	Año según el número de la semana (ISO-8601).	*1999* o *2003*
Y	Representación de 4 dígitos de un año	*1999* o *2003*
y	Representación de dos dígitos de un año	*99* o *03*
	Hora	
a	Ante meridiem y Post meridiem en minúsculas.	*am* o *pm*
A	Ante meridiem y Post meridiem en mayúsculas.	*AM* o *PM*
B	Hora Internet.	*000* hasta *999*
g	Formato de 12 horas de una hora sin ceros iniciales.	*1* hasta *12*
G	Formato de 24 horas de una hora sin ceros iniciales.	*0* hasta *23*
h	Formato de 12 horas de una hora con ceros iniciales.	*01* hasta *12*
H	Formato de 24 horas de una hora con ceros iniciales.	*00* hasta *23*
i	Minutos, con ceros iniciales.	*00* hasta *59*
s	Segundos, con ceros iniciales.	*00* hasta *59*
u	Microsegundos.	*654321*
	Zona Horaria	
e	Identificador de zona horaria.	*UTC*, *GMT*, *Atlantic/Azores*
***I* (i mayúscula)**	Si la fecha está en horario de verano o no.	*1* si está en horario de verano, *0* si no.
O	Diferencia de la hora de Greenwich (GMT) en horas.	*+0200*
P	Diferencia con la hora de Greenwich (GMT) con dos puntos entre horas y minutos.	*+02:00*
T	Abreviatura de la zona horaria	*EST*, *MDT* ...
Z	Índice de la zona horaria en segundos. El índice para zonas horarias al oeste de UTC siempre es negativo, y para aquellas al este de UTC es siempre positivo.	*-43200* hasta *50400*
	Fecha/Hora Completa	
c	Fecha ISO 8601	2004-02-12T15:19:21+00:00
r	Fecha con formato (RFC 2822)	*Thu, 21 Dec 2000 16:01:07 +0200*
U	Segundos desde la Época Unix (1 de Enero del 1970 00:00:00 GMT)	1535136333

Nota:

- Todos los nombres de los campos que tengan acento, en el script no se deben usar con acento porque no está soportado.
- Si los nombres de los campos tienen espacios, en el script se reemplazará el espacio por el guion bajo.

Date add

Devuelve una fecha después de que se haya agregado un cierto intervalo de tiempo / fecha.

dateAdd(<value>, <interval>, <quantity>)

- **Value:** Campo date,datetime o función CurrentDate.
- **Interval:** día,mes,año
- **Quantity:** Cantidad que se va a agregar a la fecha.´

```
dateAdd( input.Fecha_de_ingreso,"second", 3 )
```

Date subtract

Devuelve una fecha después de que se ha restado un determinado intervalo de tiempo / fecha.

dateSub(<value>, <interval>, <quantity> **)**

- **Value:** Campo date,datetime o función CurrentDate.
- **Interval:** día,mes,año,hora,minuto,segundo
- **Quantity**: Cantidad que se va a restar a la fecha.

```
dateSub( input.Fecha, "year", 5 )
```

Date difference

Devuelve la diferencia entre dos valores de fecha, en función del intervalo especificado.

dateDiff(<value1>, <value2>, <interval> **)**

- **value1:** Campo date, datetime o función CurrentDate
- **value2:** Campo date, datetime o función CurrentDate
- **interval:** Intervalo de día, mes, año, hora, Segundo, minuto.

```
dateDiff(input.Fecha_de_ingreso, input.Fecha,"day")
```

Date format

Da formato a una fecha según lo especificado por una máscara de formato.

dateFormat(<value>, <date_format> **)**

- **value:** Campo date,datetime o función CurrentDate
- **date_format:** Tipo de format para la fecha

  - Y-m-d H:i:s (2013-03-10 17:16:18)
  - F j, Y, g:i:a (March 10,2013 5:16 pm)

```
dateFormat( input.Fecha_de_ingreso,"F j, Y, g:i a")
```

Funciones de cadena

Concat

Concatena dos o más expresiones juntas.

concat(<params> **)**

- **Params:** Expresiones que se van a concatenar, se separan por comas.

```
concat( "hola",input.Nombre );
```

Nota:

- No está soportado concatenar campos numéricos.

Ends with

Comprueba si una cadena termina con la otra cadena. Si es correcto regresa true.

endsWith(<string>, <string>)

Ejemplo:

```
endsWith( 'Produccion', 'ccion' );
```

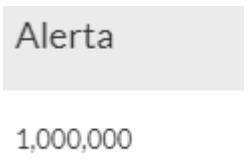
Number format

Formatea un número, es decir lo separa por comas.

number_format(<value>)

Ejemplo:

```
numero = 1000000;
alert(number_format(numero));
```



Replace

Reemplaza 1 o más caracteres de una cadena por un valor especificado.

replace(<string>, <search>, <replace> **)**

- **String:** Cadena en el que se aplicará el replace.
- **Search:** Valor o carácter buscado a cambiar.
- **Replace:** Valor o carácter nuevo.

```
replace( input.Nombre, "a", "o" )
```

Starts with

Comprueba si una cadena empieza con la otra cadena. Si es correcto regresa true.

startsWith(<string>, <string>)

Ejemplo:

```
startsWith( 'Produccion', 'Pro' );
```

Strlen

Devuelve la longitud de una cadena.

strlen(<string> **)**

- **String:** campo de tipo alfanumérico.

```
strlen( input.Nombre );
```

strpos

Encuentra la posición del carácter en una cadena de texto.

strpos(string,search**);**

- **String:** Cadena de texto donde se hará la búsqueda.
- **Search:** carácter que se buscará.

```
texto=input.Nombre;
buscar="r";
funcbus=strpos(texto,buscar);
```

Notas:

- Puede buscarse un carácter o una palabra, en casi de ser una palabra mostrará la posición del primer carácter.
- La búsqueda empieza desde el numero 0 hasta N, contando también los espacios.
- Si el carácter buscado se encuentra en la posición 0 (la primera letra) devolverá vacio.

Strtolower

Convierte una cadena de texto en minúsculas.

strtolower(string **)**;

```
var=strtolower(input.Nombre);
```

Strtoupper

Convierte una cadena de texto en mayúsculas.

strtoupper(String **);**

- **String**: Puede ser un campo de la forma o un texto (entre comillas dobles “”).

```
var=strtoupper(input.Nombre);
var=strtoupper("ingrese el nombre correctamente.");
```

substr

Devuelve una parte del string definida por los parámetros start y length.

substr(string, int_start [,int_length]**)**

- **string:** Puede ser una cadena de caracteres o un campo alfanumérico y debe tener al menos un carácter.
- **int_start:** Posición donde comenzará la cadena devuelta.
  - Si int_start es positivo, la cadena que devolvera empezará en la posición 0 del string. (Ejemplo en la palabra “bitam”, el carácter en la posición 0 es “b”)
  - Si int_start es negativo, la cadena que regresa empezará haciendo el conteo del string de la parte final hacia adelante.
- **int_lenght:** Número de caracteres que devolverá.
  - Si int_length es positivo, devolverá la cadena empezando desde la posición definida por int_star con la longitud definida en int_length (dependiendo de la longitud del string).
  - Si int_length es negativo, se omitirán el número de caracteres al final del string. (Ejemplo en la palabra “bitam”, si int_star= 0 e int_lenght =-2, devolverá la cadena “bit”)

Ejemplo	Resultado
`substr("bienvenido",0,-1);`	bienvenid
`substr("bienvenido",1,3);`	ien
`substr("bienvenido",-3,-3);`	Vacío
`substr("bienvenido",-6,5);`	venid
`substr("bienvenido",-5,-2);`	eni

Notas:
- Si la longitud del String es menor que int_start la función devolverá False.
- Si se omite int_length la subcadena empezara por int_start hasta el final de la cadena donde será devuelta.
- Si se especifica int_length y es 0, False o Null devolverá una cadena vacia.

Trim

Elimina los espacios iniciales y finales de una cadena.

trim(<string>)

- **String:** campo de tipo alfanumérico.

```
text=trim(input.Nombre);
update Alumno
(
    Nombre = text
)
where Id == input.Id;
```

Funciones Matemáticas

Abs

Devuelve el valor absoluto de un número.

abs(<value>)

- **Número:** Valor decimal a convertir en valor absoluto.

Ejemplo:

```
abs( -4 );
```

Ceil

Devuelve el valor redondeado hacia arriba.

ceil(<value>)

- **Número:** Valor numérico a redondear.

Ejemplo:

```
ceil( 5.62998 );
```

Floor

Devuelve el valor redondeado hacia abajo.

floor(<value>)

- **Número:** Valor numérico a redondear.

Ejemplo:

```
floor( 4.5698 );
```

Mod

Devuelve el resto después de que un número se divide por divisor.

mod(<value1>, <value2>)

- **Número:** Número dividendo.
- **Divisor:** Número divisor.

Ejemplo:

```
mod( 5.7 , 1.3 );
```

Pow

Devuelve el resultado de un número elevado a una potencia.

pow(<value1>, <value2>)

- **Número:** Número a elevar a una potencia.
- **Potencia:** Exponente.

Ejemplo:

```
pow( 2, 8 );
```

Round

Redondea un número a un número específico de dígitos.

round(<value>)

- **Número:** Valor numérico a redondear.

Ejemplo:

```
round( 3.7 );
```

Sqrt

Devuelve la raíz cuadrada de un número.

sqrt(<value>)

- **Número:** Valor numérico al que se le calculara la raíz cuadrada.

Ejemplo:

```
sqrt( 10 );
```

Truncate

Trunca un número a un entero eliminando la parte decimal del número.

truncate(<value>)

- **Número:** Valor numérico al que se le eliminara la parte decimal.

Ejemplo:

```
truncate( 187.256 );
```

Funciones Criptográficas

Estas funciones permiten que el usuario pueda configurar un script de firmado electrónico de documentos.

Notas:

- Si se utiliza un archivo que ya viene codificado con PEM, favor de utilizar la función fileGetContents para obtener la llave/certificado correspondiente.

Get file contents

Obtiene la representación del contenido del archivo para manipulación de este.

fileGetContents(<campo_documento>)

Regresa el contenido del archivo indicado en el Campo.

DER to PEM key converter

Convierte una llave con codificación DER a codificación PEM. Regresa la llave con codificación PEM.

derToPemKey(<campo_documento>)

```
pemKey = derToPemKey(input.Llave_privada);
privateIdentifier = getPrivateKey(pemKey, input.Password);
cifrado = base64Encode(signData(input.Mensaje, privateIdentifier));
input.Sello = cifrado;
```

Regresa la llave con codificación PEM

DER to PEM certificate converter

Convierte un certificado con codificación DER a codificación PEM.

Recibe como parámetro un campo tipo file o una cadena de texto con codificación DER.

derToPemCertificate(<campo_document>)

```
certificado = derToPemCertificate(input.Certificado);
certificado = getCertificate(certificado);
llave = getPublicKey(certificado);
esValido = verifySignature(input.Mensaje, base64Decode(input.Sello), llave);
if (esValido == 1) {
    alert "Valido!";
} else if (esValido == 0) {
    alert "No valido!";
} else {
    alert "Error!";
}
releaseKey(llave);
```

Regresa el certificado con codificación PEM.

Get private key

Analiza una llave dada y la prepara para ser usada por otras funciones criptográficasRecibe como parámetros una cadena de llave privada con codificación PEM, y la contraseña del archivo de llave.

getPrivateKey(<contenido_de_llave>, <clave_secreta_de_llave >)

Regresa valor booleano falso (false) en caso de error, identificador de llave en caso de éxito.

Generate signature

Calcula una firma para el contenido especificado generando un certificado criptográfico digital utilizando el identificador asociado que se le proporciona.

Recibe como parámetros un texto a codificar y un identificador de recurso de llave privada, generado por la función getPrivateKey.

signData(<Contenido_a_firmar>, <identificador_de_llave>)

Regresa el sello binario correspondiente al contenido dado.

Release key resource

Libera un recurso de llave, asociado a un identificador.

Recibe como parámetro un identificador de recurso generado por la función getPublicKey o getPrivateKey.

releaseKey (<identificador_del_recurso>)

Regresa N/A, función vacía.

Base64 encode

Codifica el contenido proporcionado con codificación tipo Base64.

base64Encode(<contenido_a_codificar>)

Regresa el contenido codificado con algoritmo Base64.

```
pemKey = fileGetContents(input.Llave_privada);
privateIdentifier = getPrivateKey(pemKey, input.Password);
cifrado = base64Encode(signData(input.Mensaje, privateIdentifier));
input.Sello = cifrado;
```

Base64 decode

Codifica el contenido proporcionado con codificación tipo Base64.

base64Decode(<contenido_a_codificar>)

Regresa el contenido plano, decodificado con algoritmo Base64.

Get certificate data

Obtiene el contenido de un certificado estándar x.509 dado (codificación PEM).

Recibe como parámetro un campo file con un certificado o una cadena de texto con codificación PEM.

getCertificateData(<campo file>)

Regresa la matriz de valores del contenido del certificado proporcionado.

Get certificate identifier

Analiza un certificado (estándar x.509) y genera un identificador para el recurso.

Recibe como parámetro una cadena de texto con codificación PEM o un campo file con un certificado en la misma codificación.

getCertificate(<certificado>)

Regresa el identificador de recurso asociado al certificado.

Get public key

Extrae la llave publica de un certificado y lo prepara para su uso. El parámetro de entrada de esta función puede ser alguno de los siguientes:

Recurso como representación de un certificado (identificador).

Ruta del certificado (valor de un input file/document que contenga al certificado, con codificación PEM).

Una llave publica codificada con formato PEM

getPublicKey(<certificado>)

Regresa el valor booleano falso (false) en caso de error, identificador de recurso en caso de éxito.

Verify signature

Verifica que una firma corresponda al contenido proporcionado.

verifySignature(<contenido>, <firma_digital>, <identificador_llave_publica>)

Regresa el valor numérico, 1 (uno) en caso de que la firma y el contenido correspondan, 0 (cero) en caso de que no correspondan y -1 (1 negativo) en caso de error.

Release certificate

Libera el recurso de certificado asociado al identificador.

releaseCertificate(<identificador_de_certificado>)

Regresa N/A, función vacía.

```
llave = fileGetContents(input.Llave_publica);
llave = getPublicKey(llave);
esValido = verifySignature(input.Mensaje, base64Decode(input.Sello), llave);
if (esValido == 1) {
    alert "Valido!";
} else if (esValido == 0) {
    alert "No valido!";
} else {
    alert "Error!";
}
releaseKey(llave);
```

Base de datos

Sum

Obtener la suma total de los valores de una columna de tipo numérico.

FROM Forma SELECT **SUM**(nombre_campo) WHERE condición;

- Para acceder al valor del campo debe de hacerlo de la siguiente manera:

  Variable.nombre_campo

Ejemplos:

```
fsum= FROM Capacidad SELECT SUM (Promedio);
```

```
fsum= FROM Capacidad SELECT SUM (Promedio) WHERE Nombre="Rosa";
alert fsum.Promedio;
```

Count

Devuelve el número de filas de un campo determinado.

FROM Forma SELECT **COUNT**(nombre_campo) WHERE condición;

Ejemplos:

```
fcount=FROM Capacidad SELECT COUNT(Promedio);
```

```
fcount=FROM Capacidad SELECT COUNT(Nombre) WHERE Nombre="Rosa";
alert fcount.Nombre;
```

Avg

Retorna el promedio de una columna de tipo numérico.

FROM Forma SELECT **AVG**(nombre_campo) WHERE condición;

Ejemplos:

```
favg=FROM Capacidad SELECT AVG(Promedio);
```

```
favg=FROM Capacidad SELECT AVG(Promedio) WHERE Nombre="Rosa";
alert favg.Promedio;
```

Min
Devuelve el valor mínimo.

FROM Form SELECT **MIN**(nombre_campo) WHERE condición;

Ejemplos:

```
fmin= FROM Capacidad SELECT MIN(Promedio);
```

```
fmin= FROM Capacidad SELECT MIN(Promedio) WHERE Nombre="Rosa";
alert fmin.Promedio;
```

Max
Devuelve el valor máximo.

FROM Form SELECT **MAX**(nombre_campo) WHERE condición;

Ejemplos:

```
fmax= FROM Capacidad SELECT MAX(Promedio);
```

```
fmax= FROM Capacidad SELECT MAX(Promedio) WHERE Nombre="Rosa";
alert fmax.Promedio;
```

Nota:

- Estas funciones pueden utilizarse solas o combinadas, para crear consultas más complejas.

Función RunWorkflow

Esta función permite disparar el inicio de un flujo de Workflow. Esta función no se encuentra en el listado funciones en el panel de script de Workprocess.

Esta función únicamente recibe un parámetro de entrada, el cual es una variable que tiene asignado el resultado de una función Fetch.

```
Tiendas = from Tienda select all ID where Bandera == 'test';
RunWorkflow(Tiendas);
```

La función requiere que una de las columnas que se eligen en el fetch sea el ID consecutivo de la fila, ya que esta columna es necesaria en la función "RunWorkflow" para disparar los flujos definidos para la forma.

Función executeQuery

La función executeQuery permite la ejecución de consultas SQL dentro de la funcionalidad de Workprocess.

Esta función no aparecerá en el cuadro de funciones, por la prioridad del uso de scripts al contrario de utilizar SQL de forma directa.

executeQuery("statutoSQL");

```
executeQuery("UPDATE forma_ejemplo SET nombre = 'X' WHERE opciones = 20");
```

La función executeQuery puede soportar parámetros adicionales para el estatuto sql y entonces, sería la siguiente sintaxis:

executeQuery("statutoSQL", [variable1, variable2, …]);

```
executeQuery("UPDATE forma_ejemplo SET nombre = ? WHERE opciones = ?", [input.nombre,
input.opcion]);
```

Por otro lado, si se necesita parámetros, se deberá añadir el caracter "?" por cada parámetro, como un comodín dentro del query. Este carácter debe usarse sin comillas ya que al momento de realizar la consulta el motor interno de SQL lo reemplaza por los parámetros en el mismo orden que se encuentren definidos.

Observe el siguiente ejemplo:

```
Vquery= executeQuery("SELECT * FROM forma_ejemplo WHERE opciones = ?", [input.opcion]);

alert (Vquery);
```

El query o consulta debe definirse bajo los estándares de sintaxis soportados por SQL, para facilidad de uso, el query puede almacenarse en una variable por separado. Independientemente de si es colocado dentro de una variable, la sentencia del query siempre va dentro de comillas dobles.

Observe el siguiente ejemplo:

```
vQuery= "INSERT INTO tabla_ejemplo (nombre, cantidad, costo, opciones, fecha) VALUES (?, ?, ?, ?,
?)";

vQueryOK= executeQuery(vQuery, [input.nombre, input.cantidad, input.numero, input.opcion,
input.fecha]);

alert (vQueryOK);
```

Los parámetros pueden ser un valor fijo alfanumérico, numérico o un campo de la forma (valor variable). Los parámetros estarán localizados dentro de corchetes y separados por comas. Los valores alfanuméricos deberán colocarse dentro de comillas dobles. Como comentado anteriormente, estos parámetros reemplazarían los comodines (?) colocados dentro del query.

La función *executeQuery* puede ejecutarse directamente sin necesidad de colocarla en una variable o llamarla, no obstante, si se recomienda guardarla dentro de una variable y después llamarla dentro de una función alert con el fin de realizar una verificación e informar al usuario sobre su ejecución.

En caso de que el query ejecutado sea válido, marcara un valor verdadero, regresará el valor 1. Por otro lado, si el query es inválido será tomado como falso y tendrá un valor de vacio. Consultas que traten principalmente sobre el desplegado de registros como SELECT **no son soportadas** debido a la manera de operar de la función. Solamente son válidas los estatutos que actualicen datos, por ejemplo:

```
executeQuery("UPDATE forma_ejemplo SET nombre = ? WHERE opciones = ?", ["usuario", 20]);

executeQuery("DELETE FROM forma_ejemplo WHERE nombre = ? AND opciones = ?", [input.nombre,
input.opcion]);
```

Consultas SQL válidas, como INSERT, UPDATE y DELETE realizan cambios que pueden ser apreciados directamente desde la vista de la forma.

```
vQuery= "CALL stTableSelect";

vQueryOk= executeQuery(vQuery);

alert (vQueryOk);

vQuery= "CALL stTableSelect (?, ?, ?, ?, ?)";

vQueryOk= executeQuery(vQuery, [input.nombre, input.cantidad, input.numero, input.opcion,
input.fecha]);

alert (vQueryOk);
```

Las consultas que invoquen un procedimiento almacenado (Stored Procedure), también son válidas para procedimientos ya existentes. Si los procedimientos reciben parámetros, entonces deberá de especificarse de la misma forma descrita arriba.

Alcance y Riesgos

La función executeQuery permite la ejecución de una gran cantidad de queries tal como si fuera SQL y eso trae la desventaja de permitir mucha libertad en el manejo de los registros, queda bajo responsabilidad del usuario los cambios que cause en la tabla, así también quedan restringidos los cambios que este pueda hacer según los permisos que tenga otorgados.

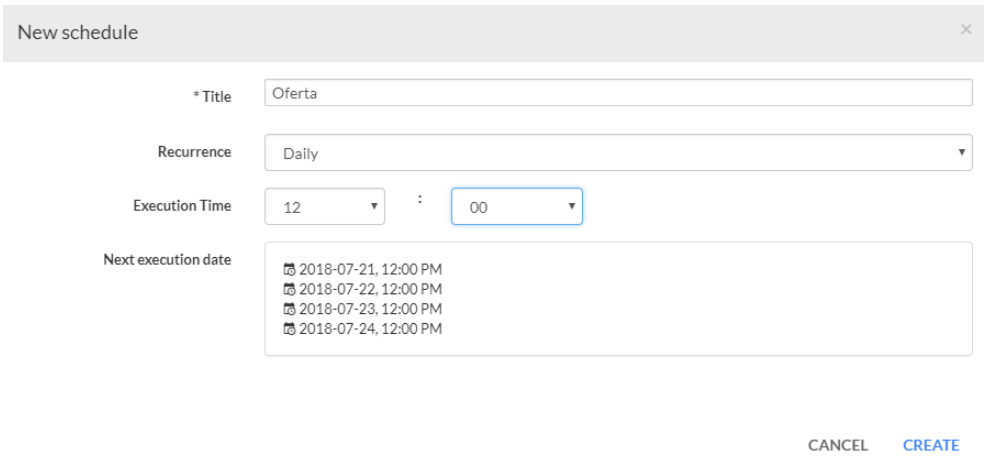
Por motivo al proceso de ejecución síncrono de la función, eBavel esperara hasta que termine de realizarse para continuar debido al tiempo de ejecución de las sentencias SQL, lo que implicaría un largo periodo de espera en el caso de consultas complicadas como los Stored Procedures de diseño complejo.

Evite utilizar estatutos SQL que creen objetos o los eliminen en la base de datos, puesto que estarán sujetos a espacio y permisos.

Programación de tareas en Workprocess

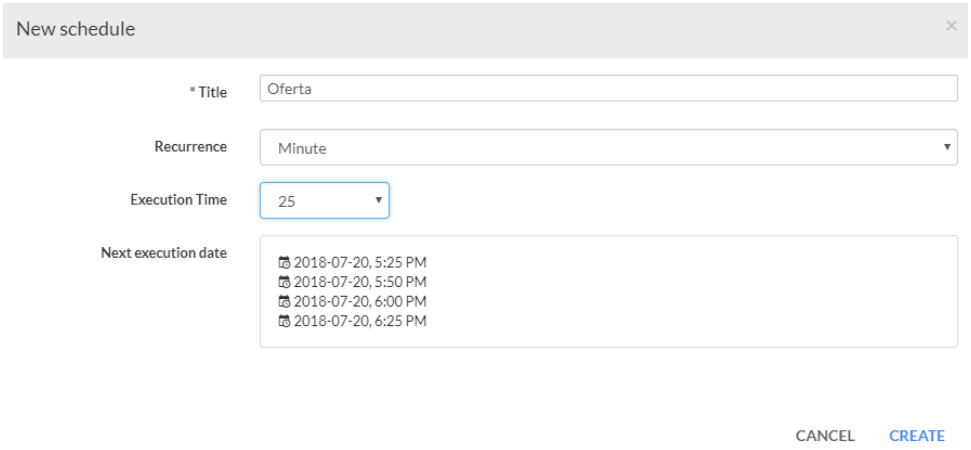
Frecuencia con la que se ejecutará una o varias tareas. Esta opción no funciona cuando la instalación de Ebavel es en modo local.

En la frecuencia Daily muestra la opción de Execution time, donde se debe seleccionar la hora en la que se ejecutará la tarea diariamente.

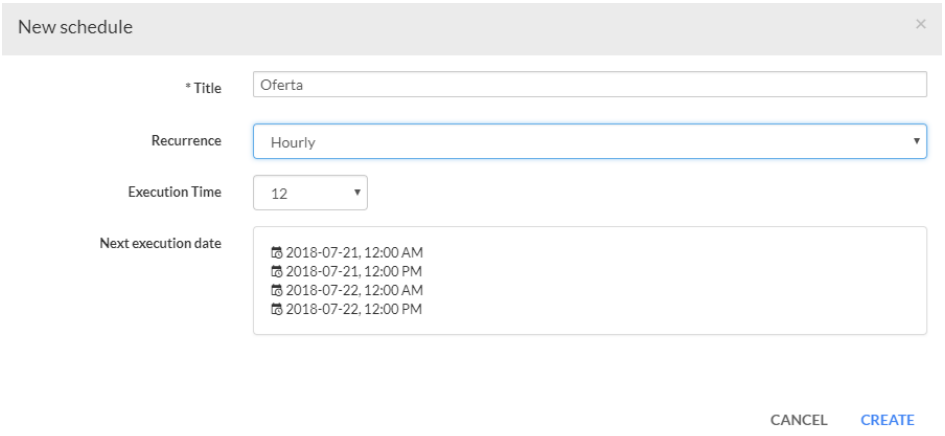


- **Title.** Nombre del Schedule.
- **Recurrence.** Frecuencia en la que se ejecutará la tarea.
- **Execution time.** Tiempo en que se ejecutará la tarea.
- **Next execution date.** Horario en el que se ejecutará la tarea.

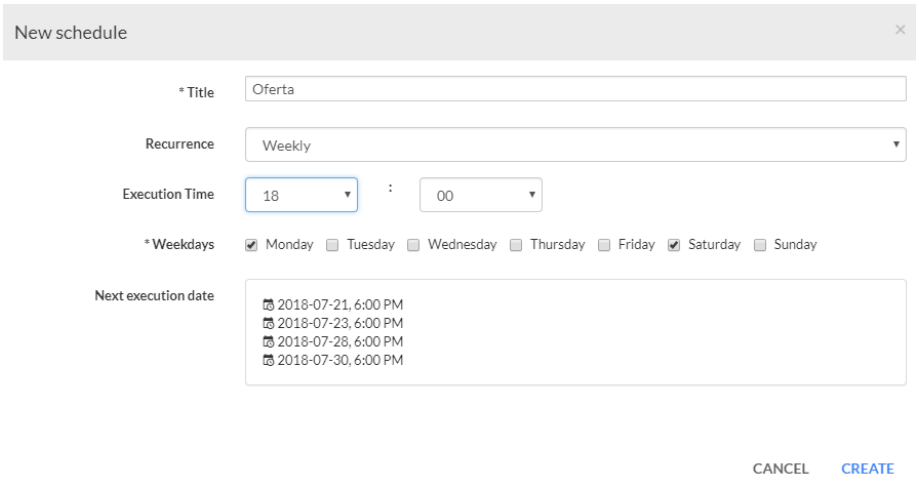
En el tiempo de ejecución de la frecuencia Minute, se desplegará una lista para seleccionar cada cuantos minutos se ejecutará la tarea. (Ejemplo: cada 25 minutos).



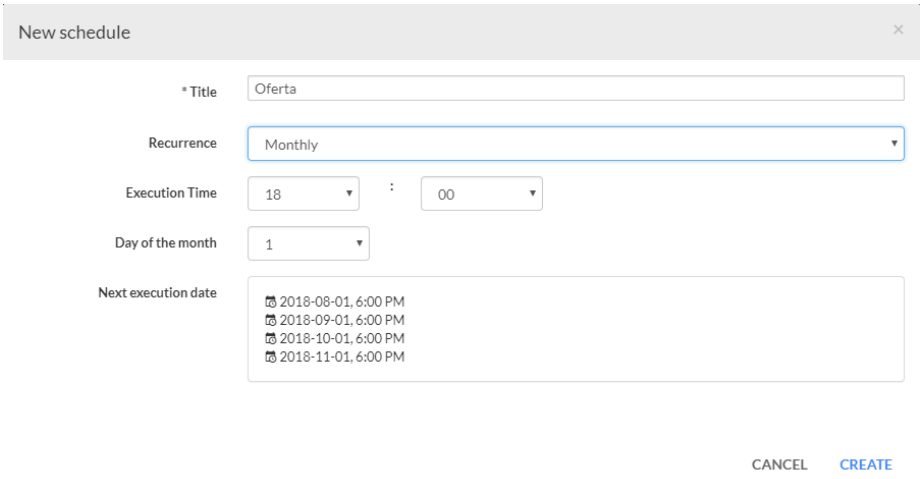
En el tiempo de ejecución de la frecuencia Hourly, se desplegará una lista para seleccionar cada cuantas horas se ejecutará la tarea. (Ejemplo cada 12 horas).



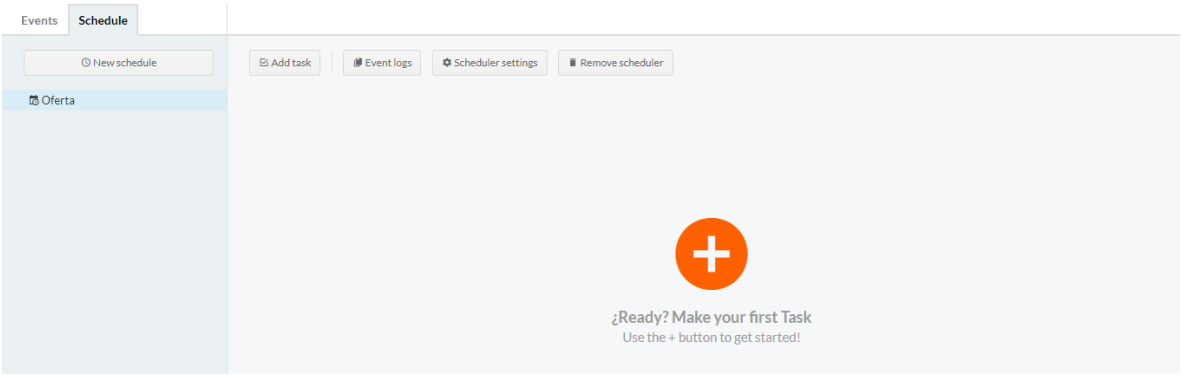
En el tiempo de ejecución de la frecuencia Weekly, se debe seleccionar la hora en la que desea que se ejecute la tarea, además de que aparecerá la opción Weekdays para seleccionar el día o los días. (Ejemplo todos los lunes y sabados a las 18:00 hrs).



En el tiempo de ejecución de la frecuencia Monthly, se debe seleccionar la hora en la que desea que se ejecute la tarea, además de que aparecerá la opción Day of the month para seleccionar el día del mes. (Ejemplo El día primero de cada mes a las 18:00 hrs).



Una vez seleccionada la recurrencia y el tiempo de ejecución, da clic en **Create** y aparecerá la siguiente ventana.



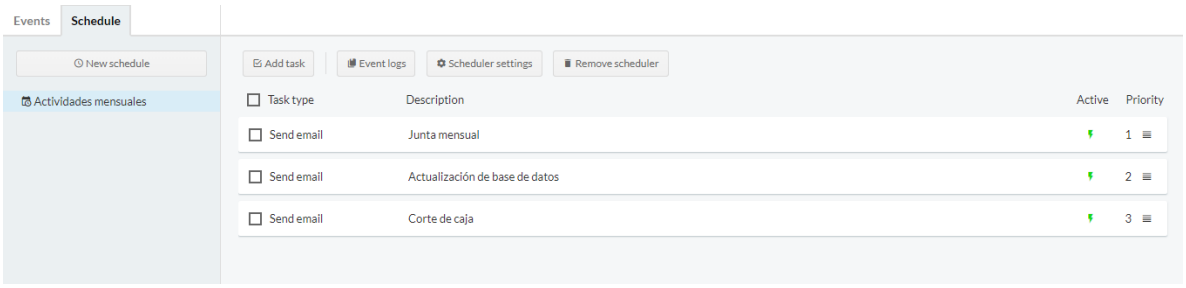
- **New Schedule.** Agrega un nuevo schedule.
- **Add Task.** Agrega una tarea.

- **Event log.** Acceso a la información básica del log.
- **Scheduler settings.** Modifica la información del schedule.
- **Remove Scheduler.** Eliminar el schedule.

Para agregar una tarea da clic en el botón Add task, y se desplegará la pantalla Create task, la cual es la misma que la de eventos.

Observe el siguiente ejemplo:

Tenemos un Schedule llamado “Actividades mensuales” que se ejecutará el día primero de cada mes a las 9:00 a.m., la cual contiene 3 tareas de tipo Send Email.



Todas las actividades estan activas, y tienen prioridades, es decir la tarea “Junta mensual” sera la primera que se ejecutará, por consecuente después de que se termine de ejecutar esa tarea, se ejecutará “Actualización de base de datos” y por último “Corte de caja”.

Funciones para recuperar datos de un archivo de timbrado XML

Se puede utilizar una tarea tipo script para recuperar datos de un archivo de un comprobante timbrado que tenga el formato XML según la normatividad del SAT v3.3

Para ello es necesario añadir a la forma un campo tipo Documento en el que se cargará el documento XML.

La ejecución del workprocess no se ejecutará con una importación del documento en forma masiva sino solo a través de la ejecución bajo demanda o por calendario.

Dentro del script, lo primero es instanciar el objeto que representará al XML con la función simpleXML.

XML = simpleXML(mivariable); donde mivariable es el campo tipo documento, como el campo puede tener varios archivos se recomienda usar un Foreach para procesar cada uno de ellos. Si solamente va a usar un archivo, entonces no es necesario usar el Foreach. Observe el siguiente ejemplo:

```
foreach(each in input.XML) {
    xml = simpleXML(each);
    if(xml) {
```

En este ejemplo el campo XML es un campo tipo documento y está en la forma donde se está lanzando el evento (de ahí la referencia “input.”).

Una vez instanciado, se tiene disponible dos métodos en el objeto:

.xpath(): Busca el nodo que cumpla la ruta definida.

```
foreach(each in input.XML) {
    xml = simpleXML(each);
    if(xml) {
        foreach(concepto in xml.xpath('/cfdi:Comprobante')) {
```

.getAttribute(): Obtiene el valor del atributo del nodo actual

```
foreach(each in input.XML) {
    xml = simpleXML(each);
    if(xml) {
        foreach(concepto in xml.xpath('/cfdi:Comprobante')) {
            varSerie=concepto.getAttribute('Serie');
            varFolio=concepto.getAttribute('Folio');
            varFecha=concepto.getAttribute('Fecha');
            varSello=concepto.getAttribute('Sello');
            varNCertificado=concepto.getAttribute('NoCertificado');
            varCertificado=concepto.getAttribute('Certificado');
            varMoneda=concepto.getAttribute('Moneda');
            varTotal=concepto.getAttribute('Total');
```

Una vez recuperado los datos, los puede insertar en otra forma a través de un “Insert into”. En el Ejemplo se Insertaron los datos en una forma llamada “Datos_CFDI”.

Dentro del insert, los datos en color negro (Serie, Folio, Fecha, etc.) hacen referencia a los campos que tiene la forma, y los datos en color azul se refiere a las variables antes declaradas que tienen en memoria los datos del XML.

```
foreach(each in input.XML) {
    xml = simpleXML(each);
    if(xml) {
        foreach(concepto in xml.xpath('/cfdi:Comprobante')) {
            varSerie=concepto.getAttribute('Serie');
            varFolio=concepto.getAttribute('Folio');
            varFecha=concepto.getAttribute('Fecha');
            varSello=concepto.getAttribute('Sello');
            varNCertificado=concepto.getAttribute('NoCertificado');
            varCertificado=concepto.getAttribute('Certificado');
            varMoneda=concepto.getAttribute('Moneda');
            varTotal=concepto.getAttribute('Total');
            insert into Datos_CFDI
            (
              Serie = varSerie
              Folio = varFolio
              Fecha = varFecha
              Sello = varSello
              NumeroCertificado = varNCertificado
              Moneda = varMoneda
              Total = varTotal
              Certificado = varCertificado
            );
        }
```

A continuación, se muestra un ejemplo completo de cómo se puede hacer la carga de un XML a distintas formas.

```
foreach(each in input.XML) {
    xml = simpleXML(each);
    if(xml) {
        foreach(concepto in xml.xpath('/cfdi:Comprobante')) {
            varSerie=concepto.getAttribute('Serie');
            varFolio=concepto.getAttribute('Folio');
            varFecha=concepto.getAttribute('Fecha');
            varSello=concepto.getAttribute('Sello');
            varNCertificado=concepto.getAttribute('NoCertificado');
            varCertificado=concepto.getAttribute('Certificado');
            varMoneda=concepto.getAttribute('Moneda');
            varTotal=concepto.getAttribute('Total');
            insert into Datos_CFDI
            (
              Serie = varSerie
              Folio = varFolio
              Fecha = varFecha
              Sello = varSello
              NumeroCertificado = varNCertificado
              Moneda = varMoneda
              Total = varTotal
              Certificado = varCertificado
            );

        }
        foreach(concepto in xml.xpath('/cfdi:Comprobante/cfdi:Emisor')) {
            varNombre=concepto.getAttribute('Nombre');
            varRFC=concepto.getAttribute('Rfc');
            varRegimenFiscal=concepto.getAttribute('RegimenFiscal');
            insert into Emisores
             (
               Nombre = varNombre
               RFC = varRFC
               RegimenFiscal = varRegimenFiscal
             );
        }
        foreach(concepto in xml.xpath('/cfdi:Comprobante/cfdi:Receptor')) {
            varNombreReceptor=concepto.getAttribute('Nombre');
            varRFCReceptor=concepto.getAttribute('Rfc');
            varUsoCFDI=concepto.getAttribute('UsoCFDI');
            insert into Receptores
             (
```

```
                Nombre = varNombreReceptor
                RFC = varRFCReceptor
                UsoCFDI = varUsoCFDI
            );
        }
        foreach(concepto in xml.xpath('/cfdi:Comprobante/cfdi:Conceptos/cfdi:Concepto')) {
                varNumeroIdentificacion = concepto.getAttribute('NoIdentificacion');
                varDescripcion = concepto.getAttribute('Descripcion');
                varCantidad = concepto.getAttribute('Cantidad');
                varUnidad = concepto.getAttribute('Unidad');
                insert into Conceptos
                (
                    NumeroIdentificacion = varNumeroIdentificacion
                    Descripcion =  varDescripcion
                    Cantidad = varCantidad
                    Unidad = varUnidad
                );
        }
    }
}
```