

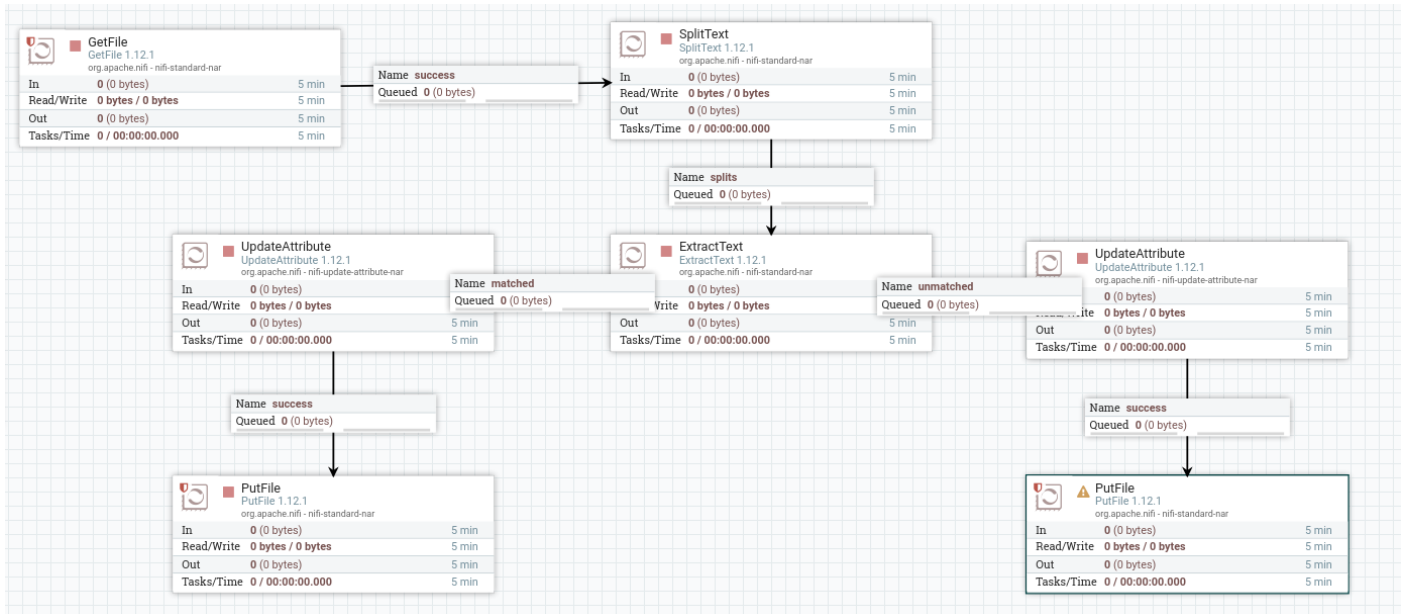
# PRÁCTICA: APACHE NIFI

Lucian Iacob

## Parte Obligatoria

### - FLUJO 1

Imagen general del flujo:



Detalle de los procesadores:

Para el procesador GetFile no es necesario dar mucho detalle pues ya lo vimos en clase, sólo añadir que partimos de un directorio llamado flujo1 y los datos los pasaremos a una carpeta llamada Datos

#### Required field

Property	Value
Input Directory	flujo1/Datos

A continuación utilizamos el procesador SplitText para separar cada fila en un documento aislado mediante los valores de “Line Split Count” = 1 y “Header Line Count” = 0 ya que no tenemos encabezado.

Running

SETTINGS SCHEDULING PROPERTIES COMMENTS

Required field

Property	Value
Line Split Count	1
Maximum Fragment Size	No value set
Header Line Count	0
Header Line Marker Characters	No value set
Remove Trailing Newlines	true

A continuación utilizamos un procesador ExtractText para crear un atributo a partir del contenido de cada línea. Para poder diferenciar cada línea, utilizaremos el número de tarjeta sanitaria que es algo individual que no se va a repetir para distintas personas, de ahí la expresión regular de 3 grupos de números. Destacar que también presenta ese formato numérico la tarjeta de la seguridad social. Pero como solo se queda con el primer match de la línea no lo tenemos en cuenta excepto en el caso de que el valor de la tarjeta sanitaria no esté, en ese caso se guardará la tarjeta de la seguridad social.

Running STOP & CONFIGURE

SETTINGS SCHEDULING PROPERTIES COMMENTS

Required field

Property	Value
Enable Canonical Equivalence	false
Enable Case-insensitive Matching	false
Permit Whitespace and Comments in Pattern	false
Enable DOTALL Mode	false
Enable Literal Parsing of the Pattern	false
Enable Multiline Mode	false
Enable Unicode-aware Case Folding	false
Enable Unicode Predefined Character Classes	false
Enable Unix Lines Mode	false
Include Capture Group 0	true
Enable repeating capture group	false
tarjeta_sanitaria	[0-9]+-[0-9]+-[0-9]+

OK

(Corregido: la expresión sería más precisa si fuera "[0-9]{3}-[0-9]{2}-[0-9]{4}")

A continuación pasamos a un procesador llamado UpdateAttribute para cambiar el nombre a nuestro fichero en función del atributo tarjeta\_sanitaria creado previamente para poder diferenciar cada registro (Accedemos a su valor mediante tarjeta\_sanitaria.0). Utilizamos 2 procesadores de este tipo, uno para cuando ExtractText haga match con el valor de la tarjeta sanitaria/seguridad social y otro para cuando no haga ningún match porque falten ambos valores en la línea. Para este caso final utilizaremos el identificador único que se genera para cada archivo uuid() para nombrar el fichero.

Aquí nos surge otro problema y es que hay líneas que están repetidas. Esto nos llevaría a tener dos ficheros llamados de la misma manera por la tarjeta sanitaria/social. Como no se especifica si se quieren eliminar los repetidos o no, vamos a añadir al número de la tarjeta\_sanitaria el número de línea que

representa el fichero respecto al documento original mediante el valor fragment.index y así guardamos todas las líneas de manera reconocible y manteniendo repetidos.

Running

STOP & CONF

SETTINGS

SCHEDULING

PROPERTIES

COMMENTS

Required field

Property	Value
Delete Attributes Expression	No value set
Store State	Do not store state
Stateful Variables Initial Value	No value set
Cache Value Lookup Cache Size	100
filename	\$(tarjeta_sanitaria.0):\${fragment.index}.csv

Por último usaremos dos PutFile que ya vimos en clase dejando los archivos en el directorio Salida dentro de flujo1 a partir del documento original. Dentro del directorio Salida dividiremos entre los casos en los que haya match o los que no:

Required field

Property	Value
Directory	flujo1/Salida/Matched
Conflict Resolution Strategy	replace
Create Missing Directories	true

Required field

Property	Value
Directory	flujo1/Salida/Unmatched
Conflict Resolution Strategy	fail
Create Missing Directories	true

A continuación mostramos los ficheros que se han generado en la carpeta Salida para comprobar que todo ha transcurrido sin problema.

```
[umaster@ibmuamdocker flujo1]$ ls Salida/Matched/
023-89-7059:3130.csv  174-82-0854:5191.csv  296-10-5168:1816.csv  443-47-7815:3576.csv  596-50-9196:9404.csv  733-75-6356:2595.csv  867-39-3765:6025.csv
061-17-8130:8112.csv  174-82-0854:5192.csv  296-15-5529:1123.csv  443-47-7815:3577.csv  596-50-9196:9405.csv  733-92-9226:5315.csv  867-40-9343:2854.csv
100-10-0647:7243.csv  174-83-4199:5195.csv  296-20-9235:6121.csv  443-47-7815:3578.csv  596-50-9196:9406.csv  734-06-9474:463.csv  867-40-9343:2855.csv
100-17-1167:3034.csv  174-83-6630:9569.csv  296-20-9235:6122.csv  443-47-8367:5736.csv  596-51-3855:843.csv  734-17-0271:4087.csv  867-40-9343:2856.csv
100-17-1167:3035.csv  174-83-6630:9570.csv  296-27-1790:7937.csv  443-62-4206:6278.csv  596-60-4315:9148.csv  734-59-0683:8999.csv  867-75-2446:4944.csv
100-22-0524:2075.csv  174-83-6630:9571.csv  296-42-3388:4646.csv  443-63-7760:6974.csv  596-72-1347:4708.csv  734-75-1198:3637.csv  868-01-5630:1743.csv
100-56-6337:6910.csv  174-83-6630:9572.csv  296-50-6970:152.csv  443-66-7356:8008.csv  597-02-3512:2343.csv  734-83-7039:69.csv  868-12-6746:6516.csv
100-61-1163:6852.csv  174-83-6630:9573.csv  297-24-7789:1130.csv  443-78-7413:4360.csv  597-04-7154:7740.csv  735-06-2492:925.csv  868-12-6746:6517.csv
100-73-8912:9221.csv  174-87-2041:7814.csv  297-24-7789:1131.csv  443-84-6727:7847.csv  597-53-7601:2166.csv  735-06-2492:926.csv  868-12-6746:6518.csv
100-77-9822:4525.csv  174-87-2041:7815.csv  297-24-7789:1132.csv  443-91-0883:5065.csv  597-66-7875:6184.csv  735-14-8740:2204.csv  868-12-6746:6520.csv
100-90-8053:8549.csv  174-87-2041:7816.csv  297-24-7789:1133.csv  443-91-1758:369.csv  597-67-0227:9778.csv  735-14-8740:2205.csv  868-12-6764:6519.csv
100-90-8053:8550.csv  174-87-2041:7817.csv  297-28-3622:2954.csv  443-93-3345:9679.csv  597-80-7851:7749.csv  735-14-8740:2206.csv  868-23-9236:3203.csv
100-90-8053:8551.csv  174-87-2041:7818.csv  297-28-3622:2955.csv  444-23-2006:2155.csv  597-80-7851:7750.csv  735-58-5842:588.csv  868-25-1078:1211.csv
100-91-0262:6790.csv  174-96-0899:2634.csv  297-73-8526:7968.csv  444-33-7958:3983.csv  597-80-7851:7751.csv  735-65-9401:2464.csv  868-47-6265:1744.csv
100-92-3180:8448.csv  174-96-0899:2635.csv  297-82-9595:9106.csv  444-45-7605:443.csv  597-89-8598:3444.csv  735-65-9401:2465.csv  868-47-6265:1745.csv
100-92-3637:694.csv  174-96-0899:2636.csv  297-83-6791:3907.csv  444-56-5802:5646.csv  597-89-8598:3445.csv  735-65-9401:2466.csv  868-47-6265:1746.csv
100-92-7795:1347.csv  175-06-5591:1999.csv  297-84-2237:4388.csv  444-62-2377:8066.csv  597-89-8598:3446.csv  735-89-9130:3457.csv  868-47-6265:1747.csv
101-01-4347:2571.csv  175-06-5591:2000.csv  297-89-9589:8977.csv  444-62-3277:8064.csv  597-89-8598:3447.csv  736-07-6706:4782.csv  868-47-6265:1748.csv
101-13-0108:2467.csv  175-06-6686:3202.csv  297-95-0540:9333.csv  444-62-3277:8065.csv  597-89-8598:3448.csv  736-14-7798:9839.csv  868-59-0391:2763.csv
101-27-4621:21.csv  175-16-8857:7648.csv  297-96-8492:845.csv  444-62-3277:8067.csv  597-92-3369:1388.csv  736-14-7798:9840.csv  868-60-5562:8668.csv
101-27-4621:4921.csv  175-20-0138:514.csv  297-96-8492:846.csv  444-95-9451:5100.csv  598-12-2270:8233.csv  736-14-7798:9841.csv  868-63-9124:8332.csv
101-27-4621:4922.csv  175-26-0763:6475.csv  298-43-9439:5974.csv  445-01-4244:7584.csv  598-34-6760:1944.csv  736-23-6094:4372.csv  868-66-3510:8235.csv
101-27-4621:4923.csv  175-32-7971:4315.csv  298-45-7144:8318.csv  445-01-4244:7585.csv  598-34-6760:1945.csv  736-37-3466:4688.csv  868-70-1600:5901.csv
101-27-4621:4924.csv  175-47-8748:7783.csv  298-63-3357:2458.csv  445-01-4244:7586.csv  598-34-6760:1946.csv  736-43-4412:6674.csv  868-85-7717:4208.csv
101-32-8601:835.csv  175-49-5714:2829.csv  298-79-0748:5701.csv  445-09-7613:2138.csv  598-34-6760:1947.csv  736-54-3410:6683.csv  869-09-5209:7114.csv
```

Para que resulte más claro también mostramos el conteo de ficheros dentro del directorio Salida:

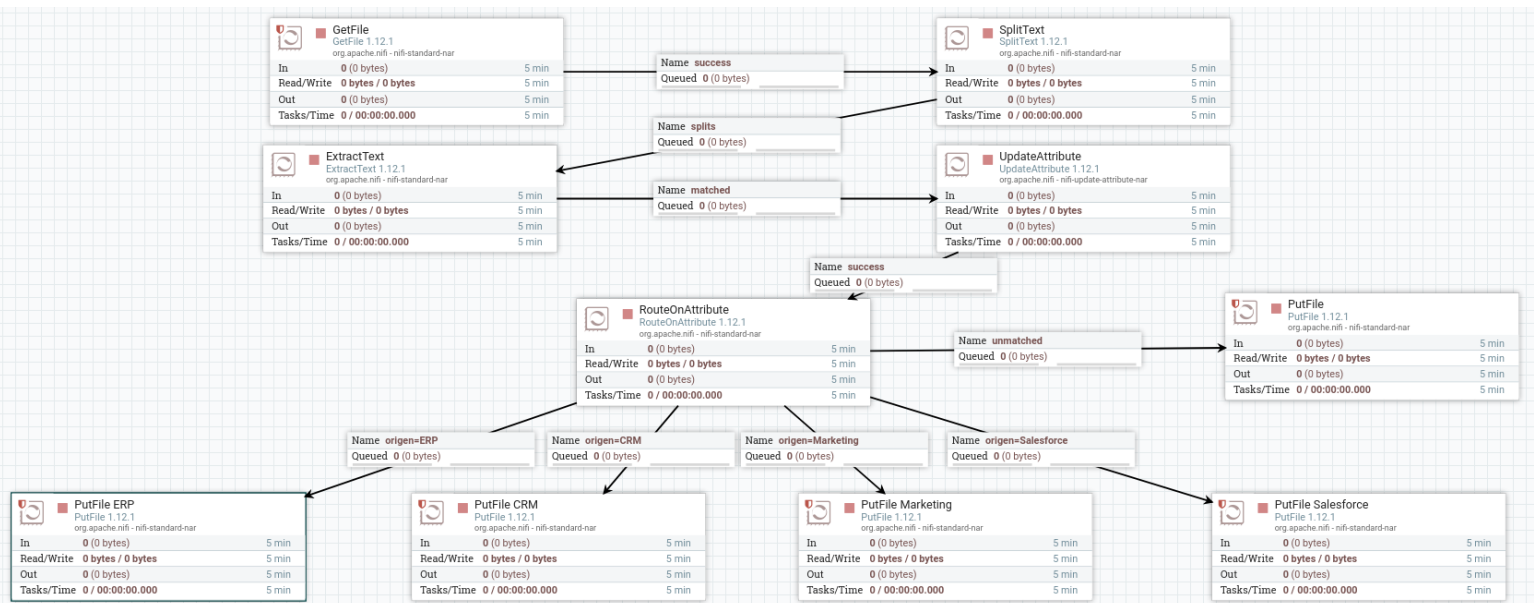
```
[umaster@ibmuamdocke flujo1]$ ls Salida/Matched/ | wc -l
9999
[umaster@ibmuamdocke flujo1]$
```

El archivo original tenía 9999 líneas así que todo parece estar en orden dentro de la carpeta Matched.

Estaría bien recalcar que la cantidad de líneas repetidas detectadas es muy grande. En caso de desear eliminarlas podríamos usar el procesador DetectDuplicate después de ExtractText indicándole que busque información sobre el atributo tarjeta\_sanitaria.0 y que los no repetidos vayan al UpdateAttribute mientras que los repetidos se vayan fuera.

## - FLUJO 2

Imagen general del flujo:



Partimos de un GetFile y un SplitText exactamente iguales que los del flujo1. Al llegar al ExtractText ahora debemos sacar un atributo más que es el origen de los datos. Este atributo corresponde con la primera palabra de la línea antes de la coma mediante la siguiente expresión regular:

origen		<code>^(.+?),</code>	
tarjeta_sanitaria		<code>[0-9]+-[0-9]+-[0-9]+</code>	

Mantenemos la tarjeta sanitaria para nombrar los ficheros y ya no nos preocupamos por el caso de que no haga match ya que vimos en el caso anterior que todas las líneas hacen match.

Tras esto, usamos UpdateAttribute para renombrar el fichero tal y como hicimos en el caso anterior.

Ahora pasamos a un procesador nuevo, un RouteOnAttribute que nos permite mover los ficheros según el valor de determinado atributo elegido (en nuestro caso el origen de los datos). Para ello definimos varias condiciones para enrutar:

Property		Value
<b>Routing Strategy</b>	?	<b>Route to Property name</b>
origen=CRM	?	\${origen:equals('CRM')}
origen=ERP	?	\${origen:equals('ERP')}
origen=Marketing	?	\${origen:equals('Marketing')}
origen=Salesforce	?	\${origen:equals('Salesforce')}

Según el valor del atributo origen definimos varios caminos que tomen los ficheros hacia sus respectivas carpetas en las que serán depositados mediante los 5 procesadores PutFile que vemos en el flujo general (las 4 fuentes de datos y el caso de que no haya match al buscar la primera palabra entre esas cuatro porque falte el valor en el registro).

Por mostrar alguno de ellos:

Property		Value
<b>Directory</b>	?	<b>flujo2/Salida/Matched/ERP</b>
<b>Conflict Resolution Strategy</b>	?	<b>fail</b>
<b>Create Missing Directories</b>	?	<b>true</b>

Siguiendo el esquema de directorios anterior tendremos una carpeta Matched donde tendremos las otras 4 correspondientes al origen de los datos (ERP en el ejemplo) y una carpeta Unmatched para el caso de que no esté el valor.

Property		Value
<b>Directory</b>	?	<b>flujo2/Salida/Unmatched</b>
<b>Conflict Resolution Strategy</b>	?	<b>fail</b>

Para comprobar que el enrutamientos de los datos se ha hecho correctamente primero lanzamos estos comandos que nos cuenten las líneas que hay para cada fuente de datos:

```
[umaster@ibmuamdocke flujo2]$ cat UAM_Clientes_Completo.csv | grep "ERP," | wc -l
1450
[umaster@ibmuamdocke flujo2]$ cat UAM_Clientes_Completo.csv | grep "CRM," | wc -l
5166
[umaster@ibmuamdocke flujo2]$ cat UAM_Clientes_Completo.csv | grep "Salesforce," | wc -l
1333
[umaster@ibmuamdocke flujo2]$ cat UAM_Clientes_Completo.csv | grep "Marketing," | wc -l
2051
[umaster@ibmuamdocke flujo2]$
```

Nos suman las 10000 líneas que tiene el archivo original o sea que parece no haber valores ausentes para este campo.

Mostramos el diagrama de árbol de nuestro flujo:

```
[umaster@ibmuamdocker flujo2]$ tree
.
├── Datos
├── Salida
│   ├── Matched
│   │   ├── CRM
│   │   ├── ERP
│   │   ├── Marketing
│   │   └── Salesforce
│   └── Unmatched
└── UAM_Clientes_Completo.csv
```

Tras pasar los datos a la carpeta Datos veamos qué ha ocurrido:

```
[umaster@ibmuamdocker flujo2]$ ls Salida/Matched/ERP/
100-92-3637:694.csv  181-54-4145:1433.csv  299-88-0360:1016.csv  446-13-1432:776.csv  602-80-6292:856.csv  746-23-3391:261.csv  879-25-6831:527.csv
100-92-7795:1347.csv  181-54-4145:1434.csv  299-88-0360:1017.csv  446-13-1432:777.csv  603-61-8115:1346.csv  746-23-3391:262.csv  879-80-2770:1414.csv
101-27-4621:21.csv   182-40-8507:677.csv   299-88-0360:1019.csv  446-67-0339:240.csv  604-23-8139:225.csv  746-23-3391:263.csv  882-11-9493:724.csv
101-32-8601:835.csv  182-63-5622:972.csv   299-88-3060:1018.csv  446-67-0339:241.csv  604-54-0433:103.csv  746-30-6892:500.csv  882-53-8999:1060.csv
101-32-8601:836.csv  183-63-6124:754.csv   300-40-2809:1323.csv  447-31-6959:1278.csv  604-73-4818:210.csv  746-30-6892:501.csv  882-53-8999:1061.csv
101-59-0738:1358.csv  183-76-9314:852.csv   301-16-7211:11.csv    449-03-1640:248.csv  604-73-4818:211.csv  746-30-6892:502.csv  882-53-8999:1062.csv
102-31-4977:24.csv   183-81-3737:413.csv   301-82-1113:427.csv   450-67-9628:560.csv  604-73-4818:212.csv  746-30-6892:503.csv  882-53-8999:1063.csv
102-34-1977:25.csv   183-83-6341:395.csv   302-19-0756:1348.csv  450-67-9628:561.csv  604-73-4818:214.csv  746-30-6892:504.csv  882-53-8999:1064.csv
102-34-8441:1010.csv  183-83-6341:396.csv   306-01-6040:819.csv    450-67-9628:562.csv  607-43-4818:213.csv  746-44-6271:6.csv    882-57-2338:415.csv
102-34-8441:1011.csv  183-83-6341:397.csv   306-42-0592:936.csv    452-10-4749:482.csv  607-52-6249:580.csv  748-34-6906:1020.csv  882-57-2338:417.csv
```

Parecen haberse creado ficheros en las carpetas, contemos cuántos hay para comprobar:

```
[umaster@ibmuamdocker Matched]$ ls -l ERP | wc -l
1451
[umaster@ibmuamdocker Matched]$ ls -l CRM | wc -l
5166
[umaster@ibmuamdocker Matched]$ ls -l Salesforce | wc -l
1334
[umaster@ibmuamdocker Matched]$ ls -l Marketing/ | wc -l
2052
```

```
[umaster@ibmuamdocker Salida]$ ls -l Unmatched/
total 0
[umaster@ibmuamdocker Salida]$
```

Se verifica que no hay ni un registro en Unmatched y las demás carpetas contienen registros acordes a los valores de su origen de datos.

## Parte Opcional

### - Caso 1

Supongamos que queremos colocar nuestros archivos en una tabla SQL.

Necesitamos darle un esquema a nuestros datos, para ello usaríamos un Controller Service (configuración a nivel de grupo no de procesador) llamado CSVReader:

NiFi Flow Configuration

GENERAL CONTROLLER SERVICES

Name	Type	Bundle	State	Scope
CSVReader	CSVReader 1.12.1	org.apache.nifi-nifi-record-serialization-services-nar	Disabled	NiFi Flow

Las propiedades que presenta este controlador son:

Property		Value	
<b>Schema Access Strategy</b>	?	<b>Use String Fields From Header</b>	
Schema Registry	?	No value set	
Schema Name	?	\${schema.name}	
Schema Version	?	No value set	
Schema Branch	?	No value set	
Schema Text	?	\${avro.schema}	
<b>CSV Parser</b>	?	<b>Apache Commons CSV</b>	
Date Format	?	No value set	
Time Format	?	No value set	
Timestamp Format	?	No value set	
<b>CSV Format</b>	?	<b>Custom Format</b>	
<b>Value Separator</b>	?	,	
<b>Record Separator</b>	?	\n	
<b>Treat First Line as Header</b>	?	false	
<b>Treat First Line as Header</b>	?	false	
Ignore CSV Header Column Names	?	false	
<b>Quote Character</b>	?	"	
<b>Escape Character</b>	?	\	
Comment Marker	?	No value set	
Null String	?	No value set	
<b>Trim Fields</b>	?	true	
<b>Character Set</b>	?	UTF-8	

Lo primero que necesitamos es un esquema de los datos, la opción que menos trabajo supondría es añadir el header con los nombres de las columnas (lo tenemos) al principio del archivo con los datos y así en la celda “Schema Access Strategy” poder hacer que utilice el header como esquema. Es importante remarcar que en el controlador SplitText en el que dividíamos cada línea en fichero teníamos deshabilitado el header, tras este cambio habría que habilitarlo para que cada fichero lo mantuviera.

Hay otros valores que nos podrían interesar como el formato de la fecha, el separador entre valores, el encoding o el trim de los valores para eliminar espacios en blanco.

Dicho esto podemos pasar al uso de dos procesadores necesarios para acceder a la tabla SQL:

Hay un procesador llamado ConvertJSONtoSQL que nos servirá para pasar nuestros datos en formato JSON a una tabla relacional y un procesador llamado ConvertRecord para pasar nuestro texto plano a formato JSON previamente.

El procesador ConvertRecord presenta las siguientes propiedades:



### Required field

Property		Value
Record Reader	?	No value set
Record Writer	?	No value set
Include Zero Record FlowFiles	?	true

Necesitamos indicar un RecordReader que en nuestro caso será el CSVReader que hemos descrito antes y un RecordWriter que será un nuevo controlador global: JSONRecordSetWriter:

### Required field

Property		Value
Schema Write Strategy	?	Do Not Write Schema
Schema Cache	?	No value set
Schema Protocol Version	?	1
Schema Access Strategy	?	Inherit Record Schema
Schema Registry	?	No value set
Schema Name	?	\${schema.name}
Schema Version	?	No value set
Schema Branch	?	No value set
Schema Text	?	\${avro.schema}
Date Format	?	No value set
Time Format	?	No value set
Timestamp Format	?	No value set
Pretty Print JSON	?	false
Suppress Null Values	?	Never Suppress

Podemos dejarlo como viene por defecto gracias a los cambios que ya hemos hecho previamente.

Tras esto definido podríamos terminar de configurar nuestro ConvertRecord (Importante habilitar los controladores antes de nada):

### Required field

Property		Value
Record Reader	?	CSVReader
Record Writer	?	JsonRecordSetWriter
Include Zero Record FlowFiles	?	true

Conectamos este procesador al siguiente ConvertJSONtoSQL:



SETTINGS	SCHEDULING	PROPERTIES	COMMENTS
Required field			+
Property	Value		
JDBC Connection Pool	?	No value set	
Statement Type	?	No value set	
Table Name	?	No value set	
Catalog Name	?	No value set	
Schema Name	?	No value set	
Translate Field Names	?	true	
Unmatched Field Behavior	?	Ignore Unmatched Fields	
Unmatched Column Behavior	?	Fail on Unmatched Columns	
Update Keys	?	No value set	
Quote Column Identifiers	?	false	
Quote Table Identifiers	?	false	
SQL Parameter Attribute Prefix	?	sql	

Vemos que necesitamos una conexión JDBC, esta se habilitará mediante otro controlador que podremos elegir entre los que nos vienen:

Requires Controller Service  
 DBCPService 1.12.1 from org.apache.nifi - nifi-standard-services-api-nar

Compatible Controller Services

DBCPConnectionPool 1.12.1 ✓

Controller Service Name  
 DBCPConnectionPool


Bundle  
 org.apache.nifi - nifi-dbcp-service-nar


Tags  
 database, pooling, dbcp, jdbc, connection, store


Este controlador necesita configurarse también de acuerdo con ciertos valores de nuestra base de datos:

SETTINGS	PROPERTIES	COMMENTS
Required field		+
Property	Value	
Database Connection URL	?	No value set
Database Driver Class Name	?	No value set
Database Driver Location(s)	?	No value set
Kerberos Credentials Service	?	No value set
Kerberos Principal	?	No value set
Kerberos Password	?	No value set
Database User	?	No value set
Password	?	No value set
Max Wait Time	?	500 millis
Max Total Connections	?	8
Validation query	?	No value set
Minimum Idle Connections	?	0
Max Idle Connections	?	8
Max Connection Lifetime	?	-1

Tras rellenar estos valores y los del procesador podremos terminar el flow con el procesador PutSQL:

	<b>ConvertRecord</b> ConvertRecord 1.12.1 org.apache.nifi - nifi-standard-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

	<b>ConvertJSONToSQL</b> ConvertJSONToSQL 1.12.1 org.apache.nifi - nifi-standard-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

	<b>PutSQL</b> PutSQL 1.12.1 org.apache.nifi - nifi-standard-nar	
In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Habría que prestar atención para introducir los valores necesarios en lo que se refiere a nuestra tabla de datos de manera correcta y también recordar que seguimos teniendo muchas entradas repetidas que no hemos eliminado.

## - Caso 2

Este flujo sería muy similar al flujo2 de la parte obligatoria, ahora habría que capturar la parte correspondiente a la fecha de nacimiento mediante la expresión regular adecuada. Tras revisar gran parte de los registros, se ha visto que hay varios formatos:

MM/DD/YY aproximadamente la mitad

MM/DD/YYYY aproximadamente la mitad

YYYY-MM-DD unos pocos y datos sin sentido (e.g. 1960-07-00)

Con la expresión regular “[0-9]+\V[0-9]+\V[0-9]+” capturaríamos los dos primeros formatos y por tanto la mayoría de los datos, pero si queremos hilar más fino y capturar los 3 casos la expresión regular debería ser: “,{8,10}},.?,\*\$” (En <https://regex101.com/> se puede comprobar que funcionan). Con esto podríamos crear el atributo fecha de nacimiento.

Dado que fechas puede haber todas las que se nos ocurra, sería conveniente quedarse con cierta parte de la fecha para discriminar y guardar en un directorio. Podríamos por ejemplo coger el mes y crear 12 carpetas, una para cada uno y ya dentro guardar los archivos con el nombre siendo la fecha completa y el fragment.index para evitar la repetición de nombres como hicimos antes.

Siguiente problema, sacar el mes. Para ello utilizaremos las expresiones de lenguaje de Nifi para establecer condicionales.

```
>> ${fecha_de_nacimiento:
```

```
>> contains('-')
```

```
>> :ifElse(
```

```
>>   ${filename:substring(5, 6)},
```

```
>>   ${filename:substring(0, 1)}))
```

Si fecha\_de\_nacimiento contiene guiones nos quedamos con los caracteres 5-6 y si no, con los caracteres 0-1.

Con esto sacaríamos el mes y podríamos utilizar el procesador RouteOnAttribute utilizando las 12 condiciones que suponen los meses del año de manera completamente análoga a cómo lo hicimos con el enrutamiento por origen de los datos “\${mes>equals('01')}, \${mes>equals('02')}, etc.”

Por lo demás sería todo igual al fujo2 como ya hemos dicho.