
Docuementación de PySafet

Publicación

Cenditel

26 de May de 2015

1. Qué es PySafet?	3
2. Organización de trabajo	5
3. Motivación y objetivos	7
4. Ejemplos de algunos Diagrama de flujo de trabajo	9
5. Arquitectura	11
6. Sistema Automatizado de Firma Electrónica y Estampado Electrónico	13
7. Debian Wheezy 7.0 (64 bits/32bits)	15
7.1. Funciones	16
8. Ubuntu (32 bits/ 64 bits)	17
9. Archivos fuentes	19
9.1. Pasos para la compilación	20
9.2. Verificación de la instalación	20
9.3. Funciones	20
10. Paso para la creación del directorio .safet/	21
10.1. 1° PRIMER PASO	21
10.2. 2° SEGUNDO PASO	21
10.3. 3° TERCER PASO	21
11. Estructura del directorio .safet/	23
12. auth.conf (Archivo de autenticación y autorización)	25
13. safet.conf (Archivo de configuración inicial)	29
14. Clase MainWindow(Safet.MainWindow)	39
15. Clase SafetXmlObject (Safet.SafetXmlObject)	45

16. Clase SafetDocument	47
17. Clase SafetVariable	51
18. Clase SafetYAWL	53
19. Clase SafetWorkflow	55
20. Creación de las tablas en base de datos	57
21. Operaciones CRUD+(flujo) utilizando PySafet	65
21.1. 1° PRIMER PASO	65
22. Crear el flujo de trabajo utilizando un archivo XML	81
22.1. 1° PRIMER PASO	81
22.2. 2° SEGUNDO PASO	81
22.3. 3° TERCER PASO	82
22.4. 4° CUARTO PASO	82
22.5. 5° QUINTO PASO	86
23. Ver datos usando flujos de trabajo	113
23.1. 1° PRIMERA ACCION (Tarea “vRegistrado”)	113
23.2. Combinación de Json con HTML(javascript)	116
24. Ver fichas usando flujos de trabajo	125
24.1. Ficha (vRegistrado) (vPedido)	126
25. Ver gráficos generales usando flujos de trabajo	135
26. Instalación de Inflow	137
27. Visualización de la interfaz gráfica de inflow	141
Índice	151

1.- Introducción

Qué es PySafet?

Es una librería disponible para el lenguaje de programación python que permite construir aplicaciones informáticas rápidamente utilizando dos conceptos:

- **Flujo de trabajo(workflow):** Consiste en el estudio de aspectos operacionales de una actividad de trabajo, esto es, cómo se realizan y estructuran las tareas, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información y cómo se hace su seguimiento. **Más contenido** [AQUI](#).
- **Firma electrónica:** Es un concepto jurídico, equivalente electrónico al de la firma manuscrita, donde una persona acepta el contenido de un mensaje electrónico a través de cualquier medio electrónico válido.

toda la aplicación se desarrolla definiendo “que” como pysafet y no “como” Framework tradicional.

Los conocimientos para desarrollar esta aplicación son:

- **XML** (lenguaje de marcador). [Xml](#).
- **SQL** (lenguaje de consultas de base de datos). [Sql](#).
- **Python** (lenguaje dinámico). [Python](#).

Organización de trabajo

- Unas de las características de las organización de hoy en día es su funciones dinámicas, emergente y pro activo se plasman en sus procesos.
- Una organización es medible a través de sus procesos. Si los procesos funcionan bien la organización funciona bien.
- Es por ello que la gestión debe fundamentarse en los procesos, identificando de manera expedita posible fallos.
- Para medir los procesos se recurre a herramienta informática
- **¿ Qué pasa si estas herramientas no son flexibles (adaptarse en uno de los procesos)?**
 1. Muchos procesos o fases de procesos no se automatizan.
 2. Los dueños de los procesos no son los adecuados.
 3. Se produce aumentos de errores.
 4. No es buena la ejecución de los procesos (muchas veces es mala).
 5. Es frecuente la aparición de “cuellos de botella”; Al funcionamiento es pésimo cuando se aumenta la carga.
 6. No hay información oportuna y fiable.
 7. Las auditorias aumentan grandes esfuerzos y no se obtiene buenos resultados.

Ejemplo de una Organización

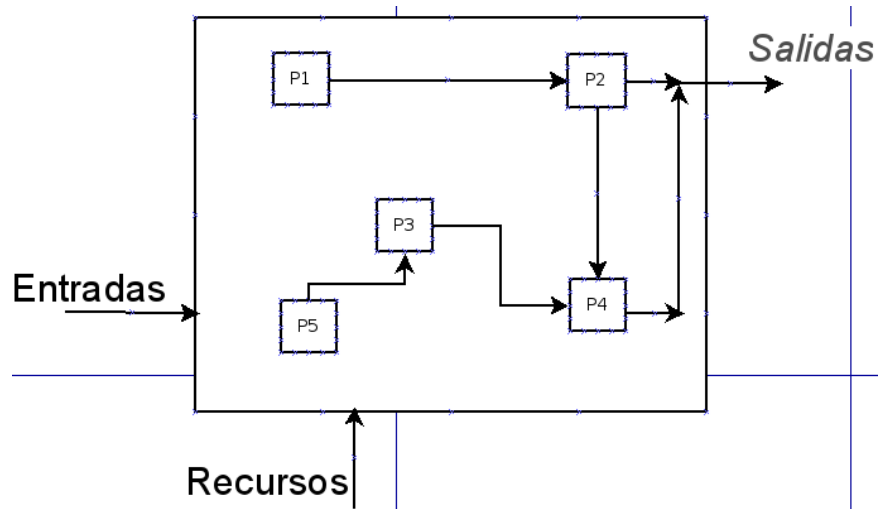


Figura1 Organización de trabajo.

Motivación y objetivos

Pysafet surge como idea de Fundacite y sucede para complementar la infraestructura de firma electrónica.

Se planteaba como un “**motor de flujos de trabajo**”.

La construcción de un motor de flujo de trabajo es compleja, hay dos modelos teóricos:

- **BPMN:** Extensión UML comunicar y como segundo objetivo implementar y calcular.
- **Redes de petri:** Calcular y segundo objetivos comunican.

Ejemplos de algunos Diagrama de flujo de trabajo

Todo proceso puede modelarse como un flujo de trabajo. Se hace necesario identificar “**Eventos**”, “**Estado**”, “**Fichas**”, “**Recursos**”, “**Dependencias**”, “**Roles**”, “**Patrones**”, “**Mensajes**”.

- **Eventos:** Posible hecho que cambia de estado es una ficha.
- **Estado:** Lugar donde reside una ficha.
- **Ficha:** Documento único dentro del proceso(puede cambiar de datos durante el cambio de estado, pero puede mantener un enlace o **id** único).
- **Recursos:** Lo que necesita el proceso para funcionar.
- **Roles:** Identidades (digitales) como posible o responsabilidades en los activos de información.
- **Patrones:** Compuestos de modelos generales o recurrentes en los procesos.
- **Mensajes:** Información sobre el a contecer de un evento dirigido a un rol externo (generalmente).

Ejemplos de flujo de trabajo

1.- Ejemplo: Solicitar un permiso (gráfico safet y petrii)

■ Gráfico usando PySafet

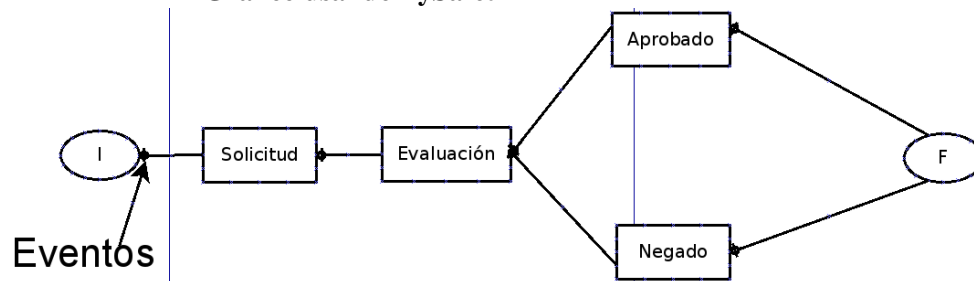


Figura 2 PySafet.

Ejercicio: Determinar Estado, eventos, fichas, dependencias, recursos y roles

■ Gráfico usando Petrii

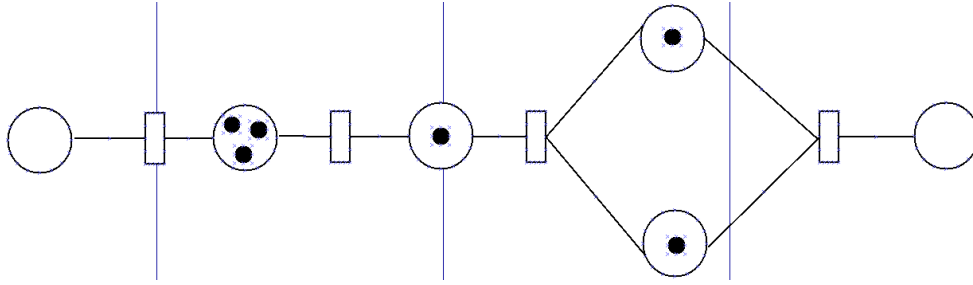


Figura 3 Petrii.

Ejercicio: Identificar eventos, estados, dependencias, roles, patrones, mensajes.

2.- Ejemplo: Caja de ahorro.

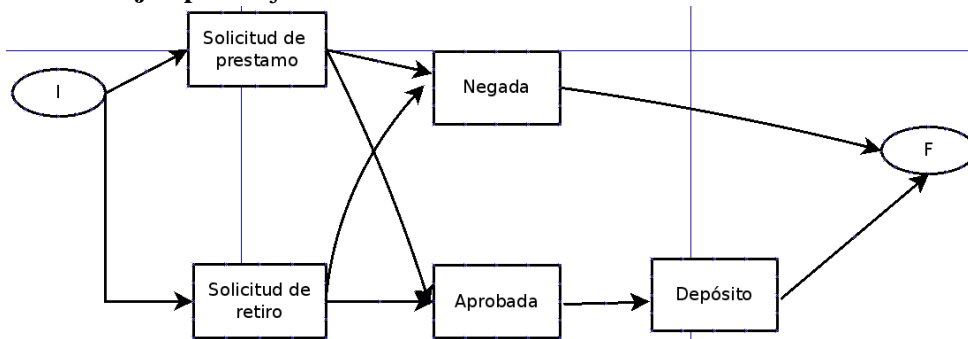


Figura 4 Caja de ahorro.

3.- Ejemplo: Hola mundo PySafet.

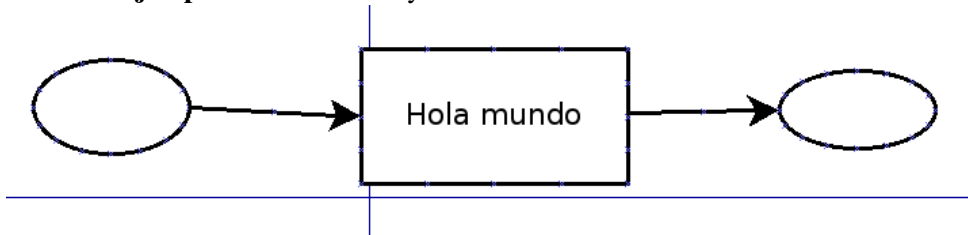


Figura5 Inicio de PySafet.

Arquitectura

Un motor de flujo de trabajo tiene como principal objetivo calcular (los) el siguiente estado(s) disponibles para la ficha.

■ **Se agregan:**

1. Visualización.
2. Reporte y tiempos.
3. Validación de estados.

Sistema Automatizado de Firma Electrónica y Estampado Electrónico

Es una herramienta que permite desarrollar nuevas aplicaciones de software con flujos de trabajo, es decir, automatización de forma expedita de procesos de una organización, agregando firma Electrónica y estampillado de tiempo.

La firma electrónica tiene un soporte legal en la Ley sobre Mensajes de Datos y Firma Electrónica (2001) de la República Bolivariana de Venezuela.

El sistema automatizado para la Firma Electrónica y Estampado de Tiempo (SAFET) surge como una herramienta que permite desarrollar nuevas aplicaciones de software que incluyan las funcionalidad de flujos de trabajo, firma Electrónica y estampillado de tiempo.

2.- Instalación

Debian Wheezy 7.0 (64 bits/32bits)

- Se Agrega el repositorio al archivo `/etc/apt/sources.list`

```
deb http://tibisay.cenditel.gob.ve/repositorio wheezy main
```

- Obtener la clave (GPG)

del repositorio Debían, utilizando la siguiente línea de comando

```
# wget http://tibisay.cenditel.gob.ve/repositorio/apt-seguridad.gpg.asc
# apt-key add apt-seguridad.gpg.asc
```

- Luego ejecutamos

```
# aptitude update
# aptitude install pysafet
```

```
root@debian:/home/cenditel# aptitude install python-setuptools
```

Se instalarán los siguiente paquetes NUEVOS:

python-setuptools

0 paquetes actualizados, 1 nuevos instalados, 0 para eliminar y 98 sin actualizar.

Necesito descargar 0 B/449 kB de ficheros. Después de desempaque tar se usarán 1.120 kB.

Seleccionando el paquete python-setuptools previamente no seleccionado.

(Leyendo la base de datos ... 308525 ficheros o directorios instalados actualmente.)

Desempaquetando python-setuptools (de ../python-setuptools_0.6.24-1_all.deb) ...

Configurando python-setuptools (0.6.24-1) ...

■

Figura 6 Instalación de pysafet.

7.1 Funciones

- **Importación** Dentro de la consola de python importamos safet para su uso

```
>>> import Safet
>>> dir(Safet)
['MainWindow', 'ParsedSqlToData', 'SafetDocument', 'SafetVariable', 'SafetWorkflow',
'SafetXmlObject', 'SafetYAWL', '__doc__', '__file__', '__name__', '__package__']
```

- **Otra forma de mostrar las funciones**

```
>>> for i in range(len(dir(Safet))):
...     dir(Safet)[i]
...
'MainWindow'
'ParsedSqlToData'
'SafetDocument'
'SafetVariable'
'SafetWorkflow'
'SafetXmlObject'
'SafetYAWL'
'__doc__'
'__file__'
'__name__'
'__package__'
```

Ubuntu (32 bits/ 64 bits)

Archivos fuentes

El código fuente de la librería Libsafet se encuentra alojada en una plataforma de desarrollo colaborativo de software (forja) llamada Github, esta plataforma utiliza como sistema de control de versión el software Git.

- Para la compilación se requiere que se instale las siguientes dependencias:

```
libgraphviz-dev
libtar-dev
g++
libglib2.0-dev
x11proto-xext-dev
libqt4-dev
libssl-dev
make
python-qt4-sql
python-sip-dev
python-qt4-dev
libqt4-sql-sqlite
```

- **Para instalar las dependencias desde la línea de comando o terminal (puede hacerlo con synaptics)**

```
# aptitude install libgraphviz-dev libtar-dev g++ libglib2.0-dev x11proto-xext-dev
libqt4-dev libssl-dev make python-qt4-sql python-sip-dev python-qt4-dev libq
```

- **Para clonar el repositorio de Libsafet debemos instalar el control de versiones git**

```
$ sudo aptitude install git
```

- **Luego ubicado en el directorio de trabajo (donde se va a descargar los fuentes) ejecuta el comando**

```
$ git clone https://github.com/victorrbravo/pysafet.git pysafet
```

Dentro del directorio de trabajo se crea el directorio pysafet donde se tiene el clone de la librería Libsafet

9.1 Pasos para la compilación

- **Compilación de la librería Libsafet**

```
$ cd pysafet/websafet
$ make clean
$ qmake-qt4
$ make
$ cd ../PySafet/
$ python configure.py
$ make
$ sudo make install
$ cd ../websafet
$ sudo make install
```

9.2 Verificación de la instalación

- **Se procede arrancar un interprete de Python, para la misma debe hacer los siguientes paso**

```
$ python
cenditel@CENDITEL:~/proyecto/pysafet/websafet$ python
Python 2.7.3 (default, Jan  2 2013, 13:56:14)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- **Dentro prompt de python (>>>), se importa la librería de Safet ejecutando el siguiente comando**

```
>>> import Safet
>>>
```

9.3 Funciones

- ```
>>> import Safet
>>> dir(Safet)
['MainWindow', 'ParsedSqlToData', 'SafetDocument', 'SafetVariable', 'SafetWorkflow',
'SafetXmlObject', 'SafetYAWL', '__doc__', '__file__', '__name__', '__package__']
```

### 3.- Configuración



---

## Paso para la creación del directorio .safet/

---

### 10.1 1° PRIMER PASO

Descargamos nuestro MensajesVersionInicial con estención .tar

**DOWNLOAD:**



ProductoVersionInicial.TAR

### 10.2 2° SEGUNDO PASO

- Creamos un directorio con cualquier nombre por ejemplo **Pysafet** en nuestro **<HOME>**

```
$ make $HOME/Pysafet
```

- Copiamos nuestro archivo (**.tar**) que descargamos a nuestra carpeta **<HOME>Pysafet/**. En este caso se descargo en la carpeta **“Descargas”**.

```
$ cp Descargas/ProductoVersionInicial.tar $HOME/Pysafet/
```

- Acedemos a la carpeta Pysafet

```
$ cd Pysafet/
```

### 10.3 3° TERCER PASO

- Crearemos un archivo (**.py**) por ejemplo **“MontarSafet.py”** dentro del directorio **<HOME>safe/**:

```
Pysafet$ touch MontarSafet.py
```

- Abrimos nuestro archivo (**MontarSafet.py**) y insertamos el siguiente Script de python:

```
import Safet
import os

myhome = os.getenv("HOME")
mymedia = myhome + "/tmp"
myurl = "http://localhost"

myinflow = Safet.MainWindow(myhome)

myinflow.setMediaPath(mymedia)
myinflow.setHostURL(myurl)

myinflow.doLoadConfiguration("ProductoVersionInicial.tar")
```

- Ejecutamos el archivo (**MontarSafet.py**) desde al consola comando como usuario normal:

```
Pysafe$ python MontarSafet.py
```

- Nos vamos al **<HOME>** a verificar si monto el (.safet/) con el comando **ls**:

```
Pysafet$ ls $HOME/.safet/

auth.conf digidoc.conf flowfiles images log reports xmlrepository
certs dtd graphs input mydb.db safet.conf
```

---

Estructura del directorio .safet/

---

- Con el comando (tree) visualizamos

```
Pysafet$ tree ../.safe/ -d
cenditel@debian:~$ tree .safet/ -d
.safet/ ----> Carpeta Principal .safet
├── certs ---->
├── dtd ---->
├── flowfiles ---->
├── graphs ----> Se encuentran los Graficas
├── images ----> Se encuentran las Imagenes
├── input ----> Se encuentran las Entradas
├── log ----> Se encuentran los Registros
├── reports ----> Se encuentran los Informes
└── xmlrepository ---->
 └── container1 ---->
```

10 directories

**Figura 7** Tree.

- Contenido de la carpeta input/

```
cenditel@debian:~$ tree .safet/input/
.safet/input/
├── defconfigure.xml
├── defconsole.xml
├── defmanagementsignfile.xml
├── deftrac.xml
└── defusers.xml
```

**Figura 8** Input.

#### 4.- Descripción de parámetros del directorio <HOME>.safet/

---

## auth.conf (Archivo de autenticación y autorización)

---

En este archivo (auth.conf) se especifican los valores de autorización, autenticación y conexión a las base de datos. los Tipo de gestor de base de datos que admitidos

**database.driver.1 = psql**

- Psql : [Postgresql](#).
- Sqlite : [sqlite3](#).

### 1.- [Database]:

En esta sección se configura la fuentes de datos. Una fuente de datos esta asociada a una conexión con una base de datos relacional. Es posible tener varias fuentes de datos, cada una representada por un indice.

[Database]

```
Información de configuración de fuentes de datos.
Una fuente de datos esta asociada a una conexión con una base
de datos relacional. Es posible tener varias fuentes de datos, cada
una representada por un indice.
```

```
Especifica el numero de fuentes de datos diferentes
database.datasources = 1
```

----->Ejem

**Note** Esté elemento (**database.datasources**) me indica el número de base de datos, de aquí en adelante todos los elementos que se requiera estará identificado por el indice asociado a una base de datos.

```
Para realizar la conexión de base de datos asociada a una fuente de datos
se requieren varios elementos:
```

```
Nombre de la fuente de datos 1
database.sourcename.1 = nombre_fuente_datos_1
```

```
Especifica el controlador asociado a la fuente de datos
Tipo de gestor de base de datos admitidos
```

```
database.driver.1 = controlador_fuente_datos_1
```

----->Ejemplo: data

```
Especifica si la fuente de datos esta activa o no
database.activated.1 = on

Especifica el nombre de host para la fuente de datos
database.host.1 = host_base_de_datos ----->Ejemplo: 127.

Especifica el nombre de la base de datos para la fuente de datos

database.db.1 = nombre_database .1

Especifica el puerto de conexion para la base de datos
database.port.1 = puerto_database.1

Especifica el nombre de usuario para la fuente de datos
database.user.1 = usuario_database.1

Especifica la contraseña de acceso para la fuente de datos
database.password.1 = contraseña_database.1
```

**2.- [Roles]:** En esta sección se define los roles que se van a utilizar en el sistema, la definición de un rol comprende dos **(2)** elementos:

- **roles.name.n** = nombre del rol n.
- **roles.description.n** = descripción del rol n.

[Roles]

```
roles.name.1 = Administrador
roles.description.1 = usuario(s) que administra el sistema

roles.name.2 = rol_1
roles.description.2 = descripción del rol -----> Ejemplo: Acciones aso

roles.name.3 = rol_2
roles.description.3 = descripción del rol -----> Ejemplo: Acciones aso
.
.
.

roles.name.n = rol_n
roles.description.3 = descripción del rol -----> Ejemplo: Acciones aso
```

**3.- [Auth]:** En esta sección se agrega los usuarios del sistemas, la definición de un usuario comprende cuatro **(4)** elementos:

- **auth.account.n** = nombre del usuario n.
- **auth.pass.n** = contraseña del usuario n cifrada en md5.
- **auth.realname** = nombre apellido y correo del usuario n separada por espacio.
- **auth..role.n** = rol del usuario n, este rol debe esta definido en el sección [Roles].

[Auth]

```
auth.account.1 = usuario 1 -----> Ejemplo:
auth.pass.1 = 0f885ebbc5a6c77dac0c319a7411f6039496f06f -----> Nota: co
auth.realname.1 = Nombre_apellido_usuario1 correo_usuario1 -----> Ejemplo:
auth.role.1 = rol del usuario -----> Ejemplo:
```

```
auth.account.2 = usuario1
auth.pass.2 = 22cd2dlc596f4091e248aff0e4aa0d47c84b2b36
auth.realname.2 = Nombre_apellido_usuario1 usuario1@mail.com
auth.role.2 = rol del usuario
```

```
.
.
.
```

```
auth.account.n = usuarion
auth.pass.n = ddc6fdd484d5a71b9619beb8b19a7ea06980d8ff
auth.realname.n = Nombre_apellido_usuarion usuarion@mail.com
auth.role.n = Desarrollador
```

#### 4.- [Permissions]:

En esta sección se van a definir los permisos de los usuarios en función de los roles y acciones que puedan ejecutar en el sistema, la definición de cada permisos comprende cuatro (4) elementos:

- **permises.operation.n** = nombre de la operación n a ejecutar (estas acción debe estar definida en el archivo deftrac.xml que mas adelante se explicara).
- **permises.accounts.n** = nombre del o los usuario(s) que ejecutara(n) la operación (esta usuario debe estar definido en la sección [Auth]). Si son varios usuarios deben estar separados por punto y como (;) **Ejemplo:** permises.accounts.n = usuario1;usuario2;usuarion.
- **permises.types.3** = tipo de acción a ejecutar, entre las acciones tenemos: read,execute y modify deben ir separadas por punto y como (;) **Ejemplo:** permises.types.3 = read;execute;modify.
- **permises.roles.4** = rol que puede ejecutar esta acción, de esta manera se puede indicar a un grupo de usuarios que tenga un rol en especifico ejecutar la acción sin la necesidad de especificarlo en el elemento **permises.accounts.n**, los roles debe estar definidos en la sección [Roles].

[Permissions]

```
permises.operation.1 = operacion_1
permises.accounts.1 = usuario1;admin
permises.types.1 = read;execute;modify
permises.roles.1 = Administrador;rol_1;rol_2
```

```
permises.operation.2 = operación_2
permises.accounts.2 = admin
permises.types.2 = read;execute;modify
permises.roles.2 = Administrador
```

```
.
.
permises.operation.n = operacion_n
permises.accounts.n = admin
permises.types.n = read;execute;modify
permises.roles.3 = Administrador;rol_1;rol_2

Operaciones de Consulta, Lista de datos y gráficos

permises.operation.24 = Listar_datos
permises.accounts.24 = admin
permises.types.24 = read;execute;modify
permises.roles.24 = Desarrollador;Administrador

permises.operation.25 = Listar_datos_con Autofiltro
permises.accounts.25 = admin
permises.types.25 = read;execute;modify
permises.roles.25 = Administrador

Viñetas

permises.operation.31 = Formulario
permises.accounts.31 = admin
permises.types.31 = read;execute;modify
permises.roles.31 = Administrador;Desarrollador

permises.operation.32 = Consulta
permises.accounts.32 = admin
permises.types.32 = read;execute;modify
permises.roles.32 = Administrador;Desarrollador
```



---

## safet.conf (Archivo de configuración inicial)

---

En este archivo se especifican los parámetros generales para el funcionamiento de la librería PySafet.

**1.- [Workflow]: (Sección de flujo de trabajo)** En esta sección se define el color del flujo y Sección para la interfaz Web.

```
[Workflow]

Establece el color por defecto
defaultcolor = white -----> # Admite white,black,blue y formato RGB #000000

Establece el color activo
activecolor = white

Establece el color pasivo
traccolor = white
```

**2.- [Operators]: (Operadores de flujo de trabajo)**

En esta sección se establecen las variables de las imágenes de los operadores que son definidos en un flujo de trabajo, entre las variables se encuentra:

- **split.xor** = Se activa un enlace saliente una vez terminada la tarea.
- **split.and** = Activa todos los vínculos salientes una vez terminada esta tarea
- **split.or** = Se utiliza para desencadenar algunos, pero no necesariamente todos los flujos salientes a otras tareas.
- **join.xor** = Esta tarea se activa cada vez que un nuevo enlace se ha activado.
- **join.and** = Esta tarea se activa cuando todos los vínculos se han activado.
- **join.or** = Esta tarea se activa cuando al menos un vínculo se ha activado.

```
[Operators]

Operador split.xor
split.xor = imgs/xor.png -----> # Archivo de imagen en el directorio <HOME>.safet/imgs

Operador split.and
```

```
split.and = imgs/and.png

Operador split.or
split.or = imgs/or.png

Operador join.xor
join.xor = imgs/join-xor.png

Operador join.and
join.and = imgs/join-and.png

Operador join.or
join.or = imgs/join-or.png

Ningun operador
none = imgs/none.png
```

### 3.- [Log]: (Registro de eventos)

En esta sección se configura las diferentes opciones de la información sobre el registro de los eventos. Entre las opciones tenemos:

- **log.filepath** = Establece la ruta absoluta donde reside el archivo de registro ejemplo: <HOME>.safet/.log
- **log.debug** = Habilita la opción de registro de mensajes de depuración (on/off)
- **log.action** = Habilita la opción de registro de de mensajes de acciones (on/off)
- **log.warning** = Habilita la opción de registro de de mensajes de advertencia (on/off)
- **log.error** = Habilita la opción de registro de de mensajes de error (on/off)
- **log.output** = Salida del registro de ejecución, las opciones son: default (file), file y screen para pantalla.

[Log]

```
Establece la ruta absoluta donde reside el archivo de registro
log.filepath = /home/pbuitrago/.safet/log

Habilita la opción de registro de mensajes de depuración
log.debug = on

Habilita la opción de registro de mensajes de acciones
log.action = on

Habilita la opción de registro de mensajes de advertencia
log.warning = on

Habilita la opción de registro de mensajes de error
log.error = on

Salida del registro de ejecución, las opciones son: default (file), file y screen
log.output = file
```

#### 4.- [XmlRepository]: (Repositorio xml)

Información sobre repositorio de documentos firmados electrónica mente bajo el formato “**bdoc**” (firma electrónica xml) asociados a flujos de trabajos.

```
[XmlRepository]
En caso de utilizar un repositorio de documentos XML remoto se debe
especificar la informacion de acceso al mismo a traves de un servicio web
xmlrepository.remote.ip = 150.187.36.6
xmlrepository.remote.urlwebservice = http://150.187.36.6/cgi-bin/safet

Establece el tipo de repositorio de documentos XML. Valores posibles:
dir: para repositorio basado en un directorio local
dbxml: para repositorio DBXML
xmlrepository.type = dir

Establece la ruta absoluta al repositorio XML

xmlrepository.path = <HOME>.safet/xmlrepository

Establece el nombre del repositorio XML
xmlrepository.name = container1
```

### 5.- [Input]: (Entrada de datos)

## Opciones para la entrada/modificación de datos

```
[Input]
Establece la ruta absoluta para el directorio en la entrada/modificacion de datos
input.path = <HOME>.safet/input
#input.file = deflista.xml
input.file = deftrac.xml
```

## 6.- [System]: (Opciones generales para la aplicación Inflow)

Opciones generales referentes al funcionamiento del tipo de aplicación : Consola/Web/Gráfica.

```
[System]
system.evalexit = on -----> # Preguntar al salir de la aplicación "inflow"

Información referente a la base de datos o repositorio
para el acceso de la librería
```

## 7.- [Widgets]: (objetos para entrada de datos)

```
[Widgets]

Lector de archivo en el sistema
lista de archivos de acceso rápido
widgets.getfilewidget.* = <HOME>.safet/flowfiles/mensajes.xml

widgets.texteditwidget.wiki.leftbold = ''' ---> # Introducción para el wi
widgets.texteditwidget.wiki.rightbold = ''' " "
```

```
widgets.texteditwidget.wiki.leftitalic = '' "" ""
widgets.texteditwidget.wiki.rightitalic = '' "" ""
widgets.texteditwidget.wiki.leftunderline = __ "" ""
widgets.texteditwidget.wiki.rightunderline = __ "" ""
```

### 8.- [GeneralOptions]: (Opciones generales)

Esta sección incluye parámetro de uso generar.

```
[GeneralOptions]
```

```
generaloptions.consoleactions.* = Gráfico Mensajes;operacion:Generar_gráfico_coloreado
Cargar_archivo_flujo: <HOME>.safet/flowfiles/mensajes.xml
```

```
generaloptions.consoleactions.* = Grafico por clave;operacion:Generar_gráfico_para_cla
Cargar_archivo_flujo: <HOME>.safet/flowfiles/mensajes.xml Clave: 9 -----> # C
```

```
Titulo para el gráfico de flujo de trabajo
```

```
generaloptions.currentflowtitle = Tareas Proyecto SAFET
```

```
Opciones On (Incluir en el gráfico, No incluir), Off si no se quiere incluir
```

```
generaloptions.currentflowtitle.font = DejaVu Sans
```

```
Tamaño de la fuente para el texto de informacin en cada flujo de trabajo
```

```
generaloptions.currentflowtitle.size = 18
```

```
Separación de la fuente para el texto de informacin en cada flujo de trabajo
```

```
generaloptions.currentflowtitle.separation = 10
```

```
Directorio donde se almacenan los archivos de salida grafos(.png,svg)
```

```
generaloptions.dir.media = <HOME>PySafet
```

```
generaloptions.currentflowtitle.include = off
```

```
Ver http://en.wikipedia.org/wiki/List_of_tz_zones_by_name
```

```
Zona horaria
```

```
generaloptions.timezone = America/Caracas
```

```
mostrar información estadística en el grafo
```

```
generaloptions.extrainfo.showonly = off
```

```
generaloptions.extrainfo.showhumanize = on
```

```
Mostrar el dialogo para los parámetros para cada documento de flujo de trabajo
```

```
Se colocan las opciones generales del sistema SAFET (Lista de acciones guardadas, en
```

### 9.- [Stats]: ( Uso de las estadística para los grafos)

Valores para el calculo de estadísticas

```
[Stats]
```

```
#Activar la recolección de estadísticas
```

```
stats.activated = off
#Fecha de inicio de calculo de estadística, * significa que no hay fecha colocada
stats.fromdate = *
#Fecha de fin de calculo de estadística, * significa que no hay fecha colocada
stats.todate = *
```

## 10.- [Libdigidoc]: (Parámetro para la firma electrónica)

Establece variables de configuración para la librería Libdigidoc utilizada por Libsafet.

```
[Libdigidoc]
Ruta del archivo de configuración digidoc para usarlo con protocolo OSCP
libdigidoc.configfi:q
lepath = <HOME>.safet/digidoc.conf

directorio donde se almacenan los certificados de validación
libdigidoc.x509storepath = <HOME>.safet/certs

Tipo de validación de firma de Digidoc "ocsp" : via protocolo OSCP, "keystore": Repositorio
de claves del navegador Mozilla / Firefox
#libdigidoc.validationtype = keystore

Tipo de archivos "MIME" permitidos
libdigidoc.mimetypes.* = application/vnd.sun.xml.writer sxw
libdigidoc.mimetypes.* = application/vnd.sun.xml.writer.template stw
libdigidoc.mimetypes.* = application/vnd.sun.xml.writer.global sxd
libdigidoc.mimetypes.* = application/vnd.stardivision.writer sdw vor
libdigidoc.mimetypes.* = application/vnd.stardivision.writer-global sgl
libdigidoc.mimetypes.* = application/vnd.sun.xml.calc sxc
libdigidoc.mimetypes.* = application/vnd.sun.xml.calc.template stc
libdigidoc.mimetypes.* = application/vnd.stardivision.calc sdc
libdigidoc.mimetypes.* = application/vnd.sun.xml.impress sxi
libdigidoc.mimetypes.* = application/vnd.sun.xml.impress.template sti
libdigidoc.mimetypes.* = application/vnd.stardivision.impress sdd sdp
libdigidoc.mimetypes.* = application/vnd.sun.xml.draw sxd
libdigidoc.mimetypes.* = application/vnd.sun.xml.draw.template std
libdigidoc.mimetypes.* = application/vnd.stardivision.draw sda
libdigidoc.mimetypes.* = application/vnd.sun.xml.math sxm
libdigidoc.mimetypes.* = application/vnd.stardivision.math smf
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.text odt
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.text-template ott
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.text-web oth
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.text-master odm
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.graphics odg
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.graphics-template otg
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.presentation odp
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.presentation-template otp
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.spreadsheet ods
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.spreadsheet-template ots
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.chart odc
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.formula odf
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.database odb
libdigidoc.mimetypes.* = application/vnd.oasis.opendocument.image odi
```

```
libdigidoc.mimetypes.* = application/pdf pdf
libdigidoc.mimetypes.* = application/xml xml
libdigidoc.mimetypes.* = text/plain txt
libdigidoc.mimetypes.* = text/css css
libdigidoc.mimetypes.* = text/xml xml
libdigidoc.mimetypes.* = text/html html
libdigidoc.mimetypes.* = application/x-gzip tgz
libdigidoc.mimetypes.* = application/zip zip
```

### 11.- [Autofilter]: (Opciones para los Auto filtros)

Sirve para realizar clasificaciones automáticas.

```
[Autofilter]
autofilter.datetime.period = 672
```

### 12.- [DefaultValues]: (Valores por defecto)

Opciones que se toman por defecto.

```
[DefaultValues]

defaultvalues.report = yes
defaultvalues.digidoc.manifest = Investigador
defaultvalues.digidoc.city = Mérida
defaultvalues.digidoc.state = Mérida
defaultvalues.digidoc.country = VE
defaultvalues.digidoc.zip = 5101
defaultvalues.panel.info.secondstohide = 4000
```

### 13.- [Reports]: (Reportes)

Opcion que es para la generación de reportes.

```
[Reports]
Configuraciones para mostrar información en la aplicación Inflow

tipo de protocolo (file,http,ftp,entre otros)
reports.protocol = file

ruta para obtener la plantilla (HTML)
reports.path = <HOME>.safet/reports

nombre de la plantilla HTML + AJAX
reports.general.template = <HOME>.safet/reports/sf_plantillaLista01.html

nombre de la plantilla para documento a firmar
reports.documenttosign.template = <HOME>.safet/reports/sf_plantillaFirma01.html

transformar fecha numerica a formato de fecha
reports.options.datetransform = on
reports.options.dateformat = dd/MM/yyyy
```

### 14.- [Graphs]: (Grafos)

### Opciones Para los gráficos de flujo de trabajo

[Graphs]

```
opciones para el texto de información en cada flujo de trabajo
opciones on/off
graphs.infotext.include = on
```

```
Formato para el texto de informacin en cada flujo de trabajo (el %date indica Fecha
graphs.infotext.format = Generado en %datetime
```

```
Nombre de la fuente para el texto de información en cada flujo de trabajo
graphs.infotext.font = Book Antiqua
```

```
Tamaño de la fuente para el texto de información en cada flujo de trabajo
graphs.infotext.size = 14
```

```
Separación de la fuente para el texto de información en cada flujo de trabajo
graphs.infotext.separation = 30
```

### 15.- [Ftp]: (Transferencia FTP)

Opción para la trasferencia de archivos utilizando el protocolo **FTP**.

[Ftp]

```
ftp.default.server = victorbravo.info
ftp.default.account = victorrbravo
```

### 16.- [ExtraInfo]: (Información estadísticas de los grafos)

Parametros de la información extra de cada nodo

[ExtraInfo]

```
extrainfo.infotext.fields = owner,porcentage,completed
extrainfo.infotext.separator =
```

```
extrainfo.infotext.completed = [*]
```

```
Mostrar campos coloreados (lo ya pasados)
extrainfo.coloured = yes
```

```
Mostrar estadísticas al final del grafo (resumen)
extrainfo.summarize = yes
```

```
extrainfo.title = Porcentaje:
```

```
extrainfo.formula = sumall
```

```
extrainfo.pattern = #PAR0#CLA00-9#CLA1+#PAR1#SLA%
```

### 17.- [Plugins]: (Complementos o componentes)

Opciones para las librerías adicionales Plugins.

```
[Plugins]
Ruta donde se encuentra las librerías adicionales (plugins)
plugins.path = /usr/lib/libsafet
```

### 18.- [Plugins.Graphviz]: (Componentes Graphviz)

Opciones para la generación de grafos utilizando la biblioteca [Graphviz](#).

```
[Plugins.Graphviz]

Formato de archivo de salida de grafo, valores posibles: svg/png/gif
plugins.graphviz.graphtype = svg

Información a mostrar en cuadro de información extra Porc,Tokens,Total,InfoText,InfoDate
plugins.graphviz.extrainfo.show = Porc,Tokens,Total,InfoText,InfoDate

#Color activo de para ser utilizado en los gráficos
plugins.graphviz.task.fillcolor = #f1f1f1

#Color activo de la linea para ser utilizado en los gráficos
plugins.graphviz.task.color = black

Atributo utilizado en la estadística Opciones posibles (Color/Size/Line/Custom)
plugins.graphviz.stats.attribute = Color

Tamaño máximo para la estadística de (Tamaño Máximo)
plugins.graphviz.stats.sizemax = 2

Tamaño mínimo para la estadística de (Tamaño Mínimo)
plugins.graphviz.stats.sizemin = 1

Color para dibujar estadística
plugins.graphviz.stats.colorgradient = yellow

Color del texto para cuadro de estadística
plugins.graphviz.stats.fontcolor = black

Color de fondo para cuadro de estadística
plugins.graphviz.stats.fillcolor = antiquewhite

Estilo de la línea del cuadro de estadística
plugins.graphviz.stats.style = dashed

Mostrar cuadro de estadística
plugins.graphviz.showstats = yes

#Dirección del grafo TB (Arriba-Abajo) LR (Izquierda-Derecha)
plugins.graphviz.graph.rankdir = TB

Tamao del fuente para todos los nodos
plugins.graphviz.graph.fontsize = 12
```



```
Separador del rank
plugins.graphviz.graph.ranksep = 0.5 equally

Figura para la tarea (Task)
plugins.graphviz.task.shape = box

Estilo de la Figura para la tarea (Task) filled,bold,dotted,empty
plugins.graphviz.task.style = filled

#Color activo de para ser utilizado en los gráficos
plugins.graphviz.condition.fillcolor = #FFFFFF

#Color activo de la línea para ser utilizado en los gráficos
plugins.graphviz.condition.color = black

Figura para la (Condition) (box, ellipse, circle, etc.)
plugins.graphviz.condition.shape = ellipse

Estilo de la Figura para la tarea (Condition)
plugins.graphviz.condition.style = filled

Mostrar la información extra solo donde existan tokens (fichas)
plugins.graphviz.extrainfo.showwheretokens = on

#Mostrar la estadística en el nodo "FINAL", valores admitidos (on/off)
plugins.graphviz.extrainfo.showfinal = on
```

## **5.- API de Safet para el lenguaje de Python**



---

## Clase MainWindow(Safet.MainWindow)

---

**Clase MainWindow:**

**Constructor:**

**MainWindow (home-usurio) home-usuario:** Directorio del usuario donde se encuentra la carpeta **.safet/**

**Ejemplo**

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome) #---> MainWindow
```

**Listado de myinflow:**

1. **setHostURL (u) [Procedimiento]** Coloca el url del servidor web actual.

**Ejemplo:**

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
...
... myinflow.setHostURL(myurl) #---> setHostURL
```

2. **setMediaPath (m) [Procedimiento]** Coloca la ruta del directorio de los archivos gráficos.

**Ejemplo:**

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
...
... myinflow.setMediaPath(mymedia) #---> setMediaPath
```

3. **addInfoGraphDateText () [Retorna cadenas (String)]** Agregar la información de fechas al grafo.
4. **addNodeToXMLWorkflow (fname,beforenode = "",nodename = "newnode",isparallel = false,options"",qu**  
Agregar un nodo al grafo.
5. **autoComplete (path) [Retorna cadenas (String)]** Retorna la lista de autocompletación.
6. **changeConnXMLWorkflow (fname,currnode,nextnode,newnode,newoptions = "",newquery = "") [Retorna**  
Cambia una conexión entre nodo.
7. **checkUserRegister (fullname,account,email,passone,passtwo) [Retorna cadenas (String)]**  
Realiza el registro de un nuevo usuario.
8. **createBdoc (content) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]**  
Crea un documento "bdoc" vacío.
9. **currentDATA () [Retorna cadenas (String)]** Devuelve los datos (DATA) actual.
10. **currentError () [Retorna cadenas (String)]** Devuelve el error actual.

### Ejemplo:

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
... myinflow.setMediaPath(mymedia)
... myinflow.setHostURL(myurl)
...
... myconsult = u"operacion:Listar_datos Cargar_archivo_flujo:/home/cenditel/..."
...
... result = myinflow.login("usuario","usuario")
...
... if not result:
... print "Consult failed error: %s" % (myinflow.currentError()) ###---->
... exit()
```

11. **currentGraphTitle () [Retorna cadenas (String)]** Devuelve el título del último gráfico generado.

12. **currentJSON () [Retorna cadenas (String)]** Devuelve la información en formato JSON de la última acción generada

**Ejemplo:**

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
... myinflow.setMediaPath(mymedia)
... myinflow.setHostURL(myurl)
...
... myconsult = u"operacion:Listar_datos Cargar_archivo_flujo:/home/cenditel/."
...
... result = myinflow.login("usuario","usuario")
...
... if not result:
... print "Consult failed error: %s" % (myinflow.currentError())
... exit()
...
... print u"%s" % (myinflow.currentJSON()) ###---->>currentJSON()
```

13. **currentTable () [Retorna cadenas (String)]** Obtiene los datos en forma de tabla del último gráfico generado.

14. **delNodeToXMLWorkflow (fname,nodename) [Retorna cadenas (String)]** Elimina un nodo de un gráfico.

15. **doLoadConfiguration (fileName)[Procedimiento]** Carga un archivo de proyecto.

**Ejemplo**

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
...
... myinflow.doLoadConfiguration("Proyecto.tar") ###---->> doLoadConfiguration
```

16. **doSaveConfiguration (fileName)[Procedimiento]** Guarda un archivo de proyecto (en el directorio **.safet/** del usuario actual).

**Ejemplo**

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
```

```
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
...
... myinflow.doSaveConfiguration("usuario.tar") #####----->> doSaveConfigur
```

17. **doSaveGraph (mypars) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]**  
Guarda los parámetros de último gráfico generado.
18. **generateFormFooter (o) [Retorna cadenas (String)]** Imprime el HTML del pie de página general para **websafet**.
19. **generateFormHead (o) [Retorna cadenas (String)]** Imprime el HTML de la cabecera general para **websafet**.
20. **generateModifyHTML (operation,fieldname,key,secondkey,form) [Retorna cadenas (String)]**  
Genera una consulta “AJAX” para los campos de combos de selección.
21. **getFlowParameters (flowfilename) [Retorna cadenas (String)]** Obtiene una lista de parámetros dado un nombre de gráfico.
22. **getInfoOfUser (user) [Retorna cadenas (String)]** Obtiene la información de un usuario.
23. **hostMediaPath () [Retorna cadenas (String)]** Obtiene la ruta del directorio de archivos multimedia.
24. **hostURL () [Retorna cadenas (String)]** Obtiene el url del host (**websafet**).
25. **inputPath () [Retorna cadenas (String)]** Obtiene el directorio (ruta) de las aplicaciones actual (**.safet**).
26. **lastInfoGraph () [Retorna cadenas (String)]** Obtiene información sobre el último gráfico generado.
27. **loadReportTemplate (json,filename = “”,nameoperation = “Listar\_datos”) [Retorna cadenas (String)]**  
Carga una plantilla para generar una salida HTML (**websafet**).
28. **log (message)[Procedimiento]** Escribe en el “log” de safet **<HOME>.safet/log/safet.log**.
29. **login (name,pass) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]**  
Inicia una sesión **PySafet**.

### Ejemplo:

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
... myinflow.setMediaPath(mymedia)
```

```
... myinflow.setHostURL(myurl)
...
... myconsult = u"operacion:Listar_datos Cargar_archivo_flujo:/home/cenditel/."
...
... result = myinflow.login("usuario","usuario") ###---> login
```

30. **logout ()** [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]  
Sale una de las sesión **PySafet**.
31. **mediaPath ()** [Retorna cadenas (String)] Obtiene las rutas del directorio “media” utilizando para generar los gráficos (.svg/.png).
32. **menuCommands ()** [Retorna cadenas (String)] Obtiene la lista de acciones en formato HTML.
33. **menuForm (o)** [Retorna cadenas (String)] Obtiene un formulario escrito en HTML correspondiente a una acción en el archivo <HOME>.safet/input/deftrac.xml.
34. **postAction ()** [Retorna cadenas (String)] Obtiene la acción posterior a ejecutar en uan operación.
35. **registerLogin (user)** [Procedimiento] Escribe en el registro de **PySafet** el inicio de una sesión.
36. **registerLogout (user)** [Procedimiento] Escribe en el registro de **PySafet** el inicio de la salida de una sesión.
37. **sendCheckEmail (user,plink)** [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]  
Envía un correo de chequeo para el resgistro de un usuario.
38. **setConffileValues (values)** [Procedimiento] Coloca los valores de configuración.
39. **setHostMediaPath (m)** [Procedimiento] Coloca la ruta para los archivo de medias.
40. **setTemplatePath (t)** [Procedimiento] Coloca el directorio de los archivos de plantilla (templates).
41. **templatePath ()** [Retorna cadenas (String)] Obtiene el directorio para los archivos de plantilla (templates).
42. **toInputConsole (action, withpermises = true)** [Retorna verdadero si se ejecuto la acción, en caso contrario]  
Ejecuta una operación de consulta \*\* en el archivo **defconsole.xml**.

#### Ejemplo:

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
... myinflow.setMediaPath(mymedia)
... myinflow.setHostURL(myurl)
...
```

```
... myconsult = u"operacion:Listar_datos Cargar_archivo_flujo:/home/cenditel/.
...
... result = myinflow.toInputConsole(myconsult) ###----->>> toInputConsole
```

43. **toInputForm (action, withpermises = true) [Retorna cadenas (String)]** Ejecuta una operación de entrada de datos (Formulario) descrita en el archivo de **deftrac.xml**.

**Ejemplo:**

```
>>> import Safet
... import os
... myhome = os.getenv("HOME")
... mymedia = myhome + "/tmp"
... myurl = "http://localhost"
...
... myinflow = Safet.MainWindow(myhome)
... myinflow.setMediaPath(mymedia)
... myinflow.setHostURL(myurl)
...
... myconsult = u"operacion:agregar_mensaje Titulo: hola mundo"
...
... result = myinflow.toInputForm(myconsult) ###----->>> toInputForm
```

44. **toInputUsers (action) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]** Ejecuta una acción de gestión de usuarios descrita en el archivo **defusers.xml**.



---

## Clase SafetXmlObject (Safet.SafetXmlObject)

---

### Clase SafetXmlObject:

```
{ %TypeHeaderCode #include "../websafet/src/SafetXmlObject.h" %End
 public: SafetXmlObject(); bool syncAttributes(const QDomElement&);
};
```



---

## Clase SafetDocument

---

**Clase SafetDocument** [SafetXmlObject] Esta clase modela un documento de una variable en **PySafet** (el documento puede ser firmado electrónicamente).

### Listado de myinflow:

1. **getCertFileList (ext = ".pem")** [Retorna cadenas (String)] Obtiene la ruta del certificado usado para firmar el documento.
2. **numberOfDataFileOnOpenXAdESContainer ()** [Retorna un entero (int)] Número de archivos que contienen el documento firmado.
3. **numberOfSignaturesOnOpenXAdESContainer ()** [Retorna un entero (int)] Número de firmas que contienen el documento firmado.
4. **getDataFileName (index = 0)** [Retorna cadenas (String)] Nombre del archivo contenido de índice "index".
5. **getDataFileLength (index = 0)** [Retorna un (long)] Tamaño del archivo contenido de índice "index".
6. **getDataFileMimeType (index = 0)** [Retorna cadenas (String)] Tipo de archivo contenido de índice "indice".
7. **numberOfDataFileAttributes (index = 0)** [Retorna un entero (int)] Atributo del archivo contenido de índice "indice".
8. **getXmlQuery (query,dcount,info = "")** [Retorna cadenas (String)] Obtiene consulta XML.
9. **addSignatureToExistingDigidocFile (keyFile,passwd,certFile,inFile,outFile)** [Retorna un entero (int)] Agrega una firma al archivo.
10. **signWithPrivateKeyOnFile (keyFile,passwd,certFile,inFile,outFile,manifest,city,state,zip,country,notaryU)** Firma electrónicamente un archivo con un certificado PCKS#LZ.
11. **initPKCS11Lib (libName)** [Procedimiento] Inicia el manejador de la tarjeta inteligente.
12. **initDigidocConfigStore (configFile)** [Retorna un entero (int)] Inicializa la configuración.
13. **verifySignMadeWithSmartCard (fileName)** [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]

14. **verifySignMadeWithSmartCard** (fileName,signatureIndex) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]
15. **verifySignMadeWithSmartCard** (fileName,listOfSigners,isneg,op = SafetDocument::AND ) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]  
Verifica una firma electrónica que fue realizada utilizando tarjeta inteligente.
16. **testSmartCardDriver** (slot) [Procedimiento] Realiza una prueba a la conexión con la tarjeta inteligente.
17. **getCN** (signatureIndex) [Retorna cadenas (String)] Retorna el nombre común de la firma “signature Index” (commun name) .
18. **getSignerIndex** (cn) [Retorna un entero (int)] Retorna el índice de firma del correspondiente nombre común (ch).
19. **getCommonNameOfSigners** () [Retorna cadenas (String)] Obtiene una lista con el nombre común de los firmantes del archivo.
20. **getNumberOfPrivateKeys** (slot,passwd,libName) [Retorna un entero (int)] Obtiene el número de claves privadas utilizados para el archivo.
21. **writeFileToDisk** (string,name) [Retorna cadenas (String)] Escribe en otro archivo el documento.
22. **returnFileToString** (pathFileName) [Retorna cadenas (String)] Convierte el archivo a una cadena.
23. **decryptDocument** (inFile,outputFile,pin) [Retorna un entero (int)] Descifra (decencripta) el archivo.
24. **decryptString** (inFile,pin)[Retorna cadenas (String)] Descifra (decencripta) una cadena.
25. **getTempNameFiles** (n) [Retorna cadenas (String)] Crea “n” archivos temporales.
26. **getCommonNameFromCertificate** (certPath) [Retorna cadenas (String)] Obtiene el nombre común (CN) de un archivo de certificación.
27. **getCountryOfSignature** (index = 0) [Retorna cadenas (String)] Obtiene el valor de país “country” especificado para la firma electrónica de índice “Index”.
28. **getStateOfSignature** (index = 0) [Retorna cadenas (String)] Obtiene el valor del Estado (STATE) departamento o región especificando para la firma electrónica de índice “Indice”.
29. **getCityOfSignature** (index = 0) [Retorna cadenas (String)] Obtiene el valor de la Ciudad.
30. **getPostalCodeOfSignature** (index = 0) [Retorna cadenas (String)] Obtiene el valor de código Postal.
31. **getCountOfRoles** (index = 0) [Retorna un entero (int)] Obtiene el número de roles (nominación de cargo del firmante).
32. **getRole** (index = 0,roleIndex = 0) [Retorna cadenas (String)]
33. **getSingingTime** (index = 0) [Retorna cadenas (String)]

34. **getSingingTimeOnlyDate** (index = 0) [Retorna cadenas (String)]
  35. **getSingingTimeOnlyHour** (index = 0) [Retorna cadenas (String)]
  36. **getSignatureType** (index = 0) [Retorna cadenas (String)]
  37. **getSignatureFormat** (index = 0) [Retorna cadenas (String)]
  38. **getSignerCertificateIssuerName** (index = 0) [Retorna cadenas (String)]
  39. **getSignerCertificateSerial** (index = 0) [Retorna cadenas (String)]
  40. **getValidAt** (index = 0) [Retorna cadenas (String)]
  41. **getValidUntil (index = 0) [Retorna cadenas (String)]** Obtiene datos del certificado electrónico usado para la firma de índice “Índice”.
  42. **getPathOfSafetDocument () [Retorna cadenas (String)]** Obtiene la ruta donde se guarda los documentos firmados.
  43. **setPathOfSafetDocument** (path) [Procedimiento]
  44. **getSubjectDN** (index = 0) [Retorna cadenas (String)]
  45. **getIssuerDN** (index = 0) [Retorna cadenas (String)]
  46. **policies** (index = 0) [Retorna cadenas (String)]
  47. **getPublicKeyAlgorithm** (index = 0) [Retorna cadenas (String)]
  48. **getPublicKeyLength** (index = 0) [Retorna cadenas (String)]
  49. **writeX509ToFile** (path,PEM,index = 0) [Procedimiento]
- QByteArray versionNumber(int index = 0)
- QByteArray serialNumber(int index = 0);
- QByteArray toHex( const QByteArray &in, QChar separator );
- QByteArray authorityKeyIdentifier(int index = 0);
- QByteArray subjectKeyIdentifier(int index = 0);



---

## Clase SafetVariable

---

Clase **SafetVariable** : SafetXmlObject

**Listado de myinflow:**

1. **addChild** (SafetXmlObject) [Procedimiento]
2. **id** () [Retorna cadenas (String)]
3. **setId** () [Procedimiento]
4. **tokenlink** (s) [Retorna cadenas (String)]
5. **setTokenlink** (b) [Procedimiento]
6. **scope** (a) [Retorna cadenas (String)]
7. **setScope** (z) [Procedimiento]
8. **type** () [Retorna cadenas (String)]
9. **setType** (a) [Procedimiento]
10. **config** () [Retorna cadenas (String)]
11. **setConfig** (q) [Procedimiento]
12. **source** () [Retorna cadenas (String)]
13. **setSource** (f) [Procedimiento]
14. **documentsource** () [Retorna cadenas (String)]
15. **setDocumentsource** (w) [Procedimiento]
16. **description** () [Retorna cadenas (String)]
17. **setDescription** (f) [Procedimiento]
18. **foo** () [Retorna cadenas (String)]
19. **getDocumentCount** () [Retorna un entero (int)]

QList<SafetDocument\*>& getDocuments();

1. **getXMLDocument** (value,fieldno,documentid) [Retorna cadenas (String)]

2. **createXMLFileFromQuery** (query,outputFileName) [Retorna cadenas (String)]



---

## Clase SafetYAWL

---

Clase **SafetYAWL** : SafetXmlObject

**Listado de myinflow:**

1. **openXML** (a) [Procedimiento]
2. **convertXMLtoObjects** () [Procedimiento]
3. **openDataSources** () [Procedimiento]
4. **setCurrentPin** (p) [Procedimiento]
5. **currentPin** () [Retorna cadenas (String)]
6. **signDocumentsFromData** (c,nametowrite,list,doc,withsmartcard = true) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]
7. **verifyDocumentsFromData** (data,doc) [Retorna verdadero si se ejecuto la acción, en caso contrario retorna falso]
8. **loadAllConfFiles** (option = 3) [Procedimiento]
9. **loadPlugins** (plugname) [Procedimiento]



---

## Clase SafetWorkflow

---

### Clase SafetWorkflow: SafetXmlObject

#### Listado de myinflow:

enum OutputFormat { XML, JSON, SVG };

1. **SafetWorkflow** ()

2. **listTasks** (inc = false,c = "n") [Retorna cadenas (String)]

// SafetDocument getDocuments(const QString& idvariable, OutputFormat of = XML // const  
QString& info = "");

// SafetDocument getDocuments(const QString& idvariable, QList<QSqlField>& fields,int  
&howmanydocuments, // OutputFormat of = XML, // const QString& info = "");

### 6.- Tutorial sobre un ejemplo de sistema de inventario



---

## Creación de las tablas en base de datos

---

### Primer Paso:

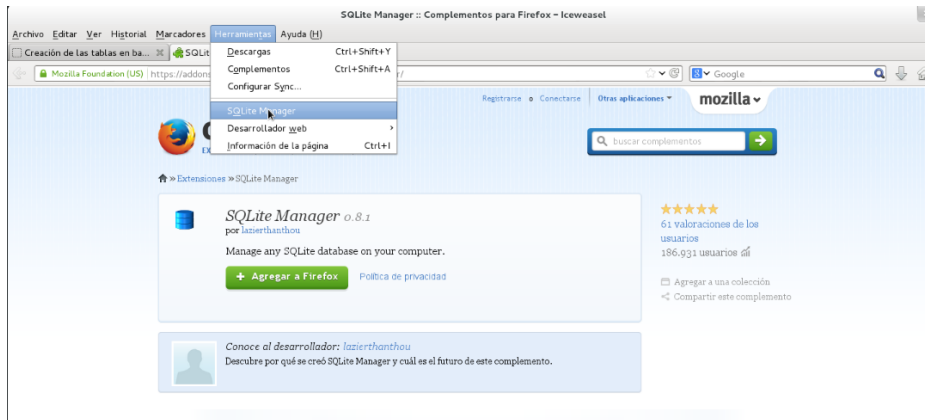
1. Se agrega la herramienta plugin del navegador web **Firefox** con el siguiente enlace. [sqlite-manager](#)
1. Aceptamos el botón Permitir.
2. Aceptamos el botón Instalar.



**Figura 12** sqlite-manager.

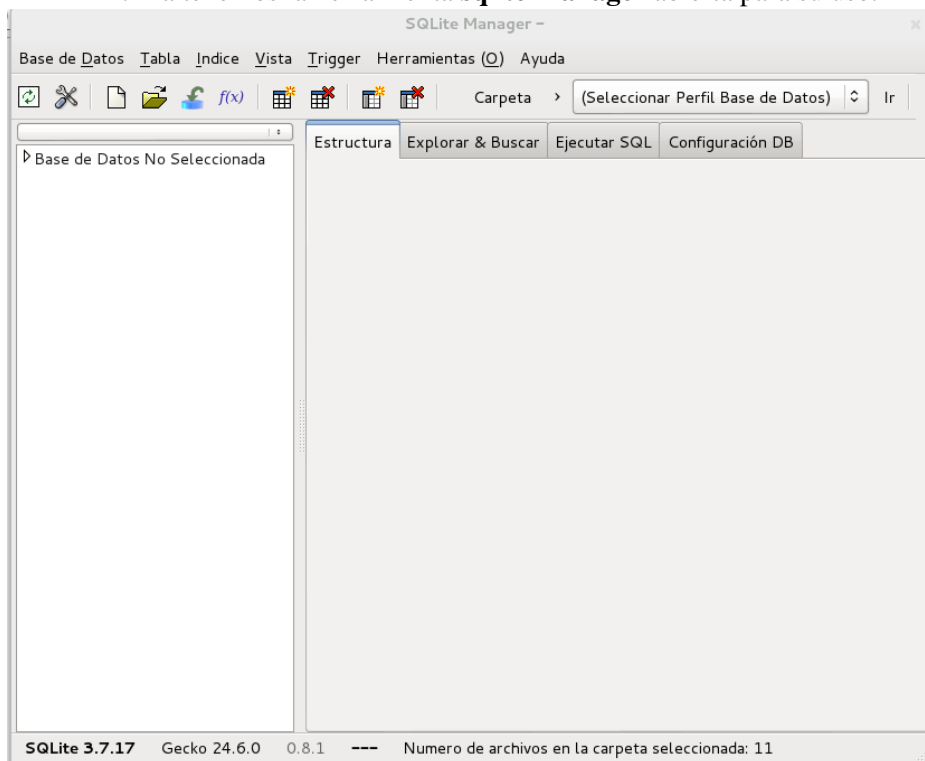
### Segundo Paso:

1. Abrimos **sqlite-manager** en el navegador web **Firefox**.
2. Buscamos en la barra de herramientas a **sqlite-manager**.
3. Le damos **click** donde dice **sqlite-manager**.



**Figura 13** Navegador sqlite-manager.

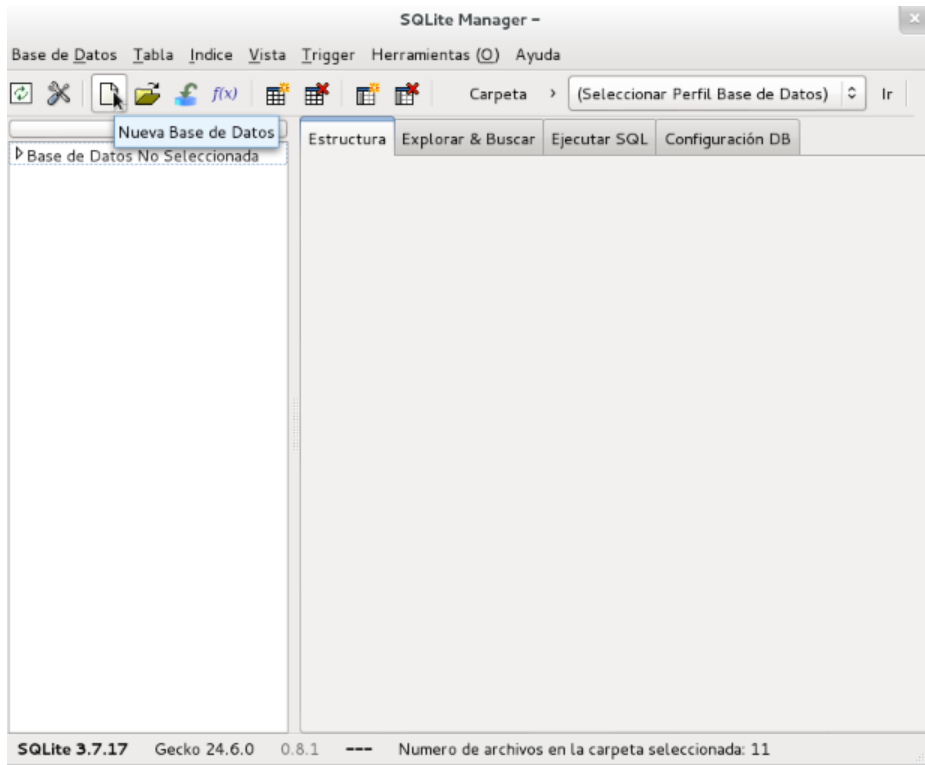
4. Ya tenemos la herramienta **sqlite-manager** abierta para su uso.



**Figura 14** sqlite-manager abierto.

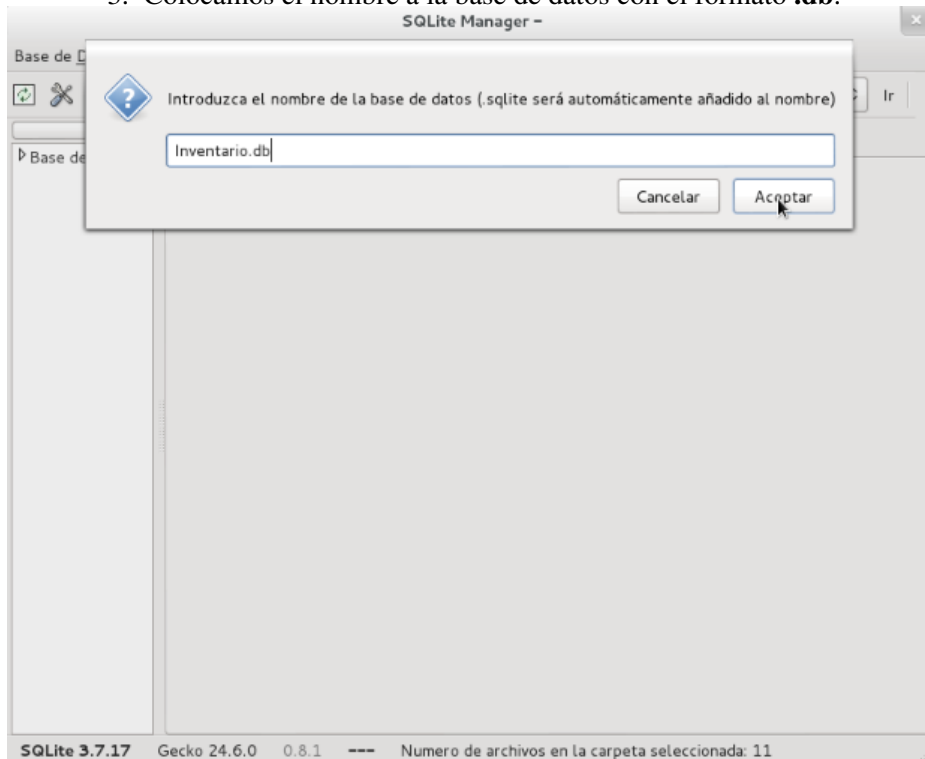
### Tercer Paso:

1. Creamos la base de datos.
2. Pulsamos donde esta la pagina en blanco como aparece en la imagen.



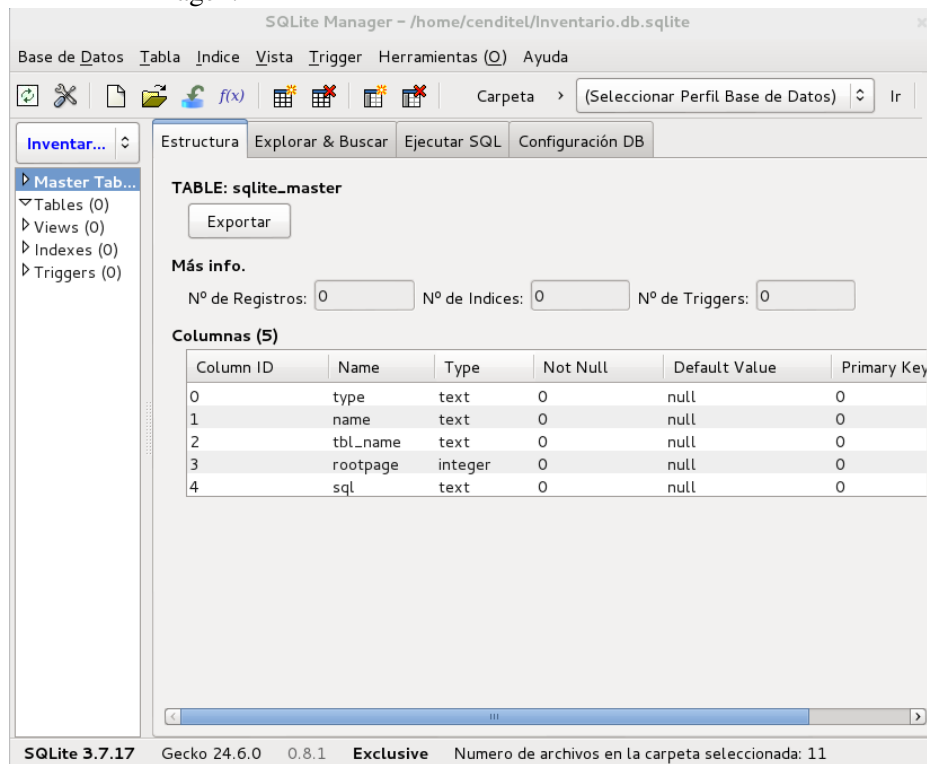
**Figura 15** Opción.

3. Colocamos el nombre a la base de datos con el formato **.db**.



**Figura 16** Nombre de la base de datos.

4. La guardamos en mi carpeta personal <HOME>.safet/.
5. Se abrirá automáticamente la base de datos con sus campos, como aparece en la siguiente imagen.



**Figura 17** Base de datos.

### Cuarto Paso:

Creación de la tablas “productos”, “productos\_registro”.

1. Buscamos en la barra de herramientas donde dice **Table**.
2. Pulsamos en la opción de **Crear Tabla**.



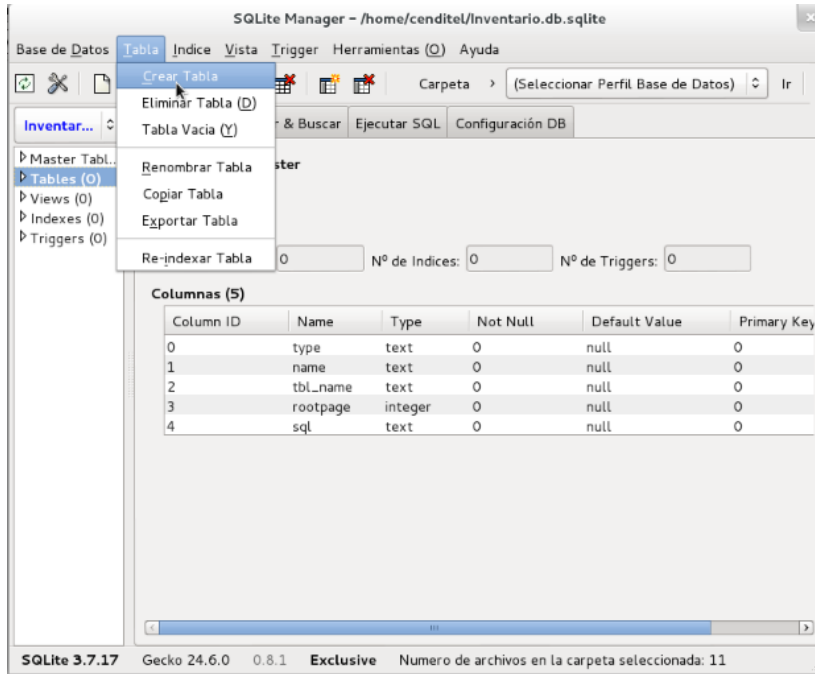


Figura 18 Opción de tables.

3. Nos aparecerá un formulario vacío sin nombre ni atributos, como en la siguiente imagen.

The screenshot shows the 'Definir Columnas' form. At the top, there are fields for 'Base de Datos' (main) and 'Nombre Tabla' (empty). Below these are checkboxes for 'Tabla Temporal' and 'Si No Existe'. The main section is a table with the following columns: 'Nombre Columna', 'Tipo Datos', 'Clave Primaria?', 'Autoinc?', 'Permitir Null?', 'Única?', and 'Valor Por Defecto'. The table is empty, with 10 rows available for input. At the bottom, there are 'Cancelar' and 'OK' buttons.

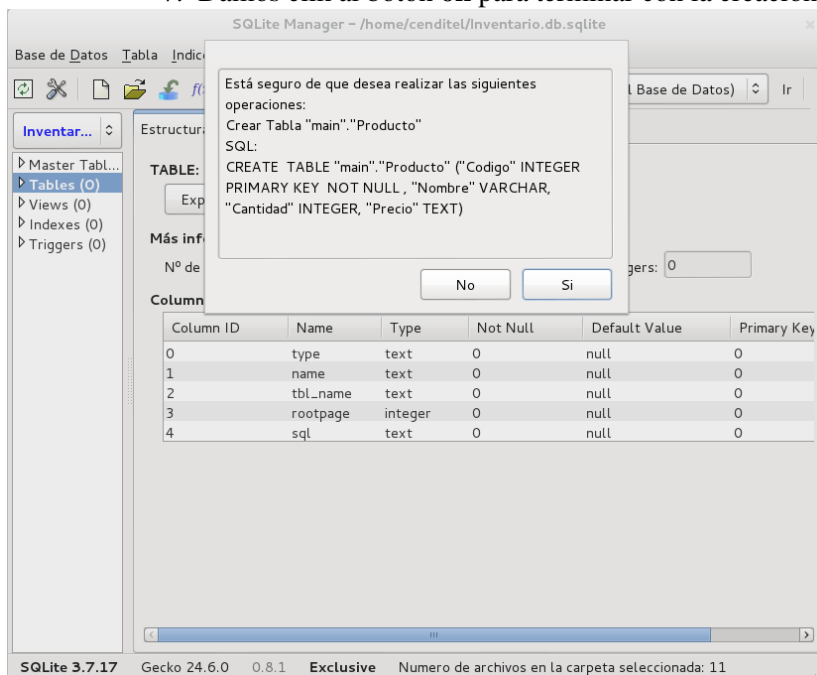
Figura 19 Formulario de tablas.

4. Llenaremos el formulario para la tabla **productos**, como aparece en la imagen.
5. Luego le pulsamos al botón **ok**.

**Figura 20** Llenado del formulario de tablas.

6. Nos muestra un mensaje como se muestra en la imagen.

7. Damos clic al botón **ok** para terminar con la creación.



**Figura 21** Mensaje.

**Nota:**

Seguimos los pasos anteriores para la creación de las demás tablas a utilizar.

8. Llenamos el formulario para la tabla **productos\_registro**.

Base de Datos:  Nombre Tabla:

☐ Tabla Temporal ☐ Si No Existe

**Definir Columnas**

| Nombre Columna | Tipo Datos | Clave Primaria?                        | Autoinc?                               | Permitir Null?                         | Única?                      | Valor Por Defecto |
|----------------|------------|----------------------------------------|----------------------------------------|----------------------------------------|-----------------------------|-------------------|
| id             | INTEGER    | <input checked="" type="checkbox"/> Sí | <input checked="" type="checkbox"/> Sí | <input type="checkbox"/> Si            | <input type="checkbox"/> Si |                   |
| productoid     | INTEGER    | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Si | <input type="checkbox"/> Si |                   |
| fecha          | INTEGER    | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Si | <input type="checkbox"/> Si |                   |
| rol            | TEXT       | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Si | <input type="checkbox"/> Si |                   |
| regstatus      | TEXT       | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Si | <input type="checkbox"/> Si |                   |
|                |            | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Si | <input type="checkbox"/> Si |                   |
|                |            | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Si | <input type="checkbox"/> Si |                   |
|                |            | <input type="checkbox"/> Sí            | <input type="checkbox"/> Sí            | <input checked="" type="checkbox"/> Si | <input type="checkbox"/> Si |                   |

Figura 22 Llenado del formulario de tablas.

## Quinto Paso:

### 1. Visualizamos la tabla **productos** creada.

SQLite Manager - /home/cenditel/Inventario.db.sqlite

Base de Datos:  Tabla:  Índice:  Vista:  Herramientas:  Ayuda:

Carpetas: (Seleccionar Perfil Base de Datos) Ir

Estructura Explorar & Buscar Ejecutar SQL Configuración DB

Master Table (1)  
Tables (3)  
  ↳ producto  
  ↳ productos\_registro  
  ↳ sqlite\_sequence  
Views (0)  
Indexes (0)  
Triggers (0)

**TABLE: producto**

**Crear sentencia**

CREATE TABLE "producto" ("id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL , "mensaje" TEXT, "status" TEXT, "cantidad" INTEGER, "nombre" TEXT, "pedir" INTEGER)

**Más info.**

Nº de Registros:  Nº de Índices:  Nº de Triggers:

**Columnas (6)**

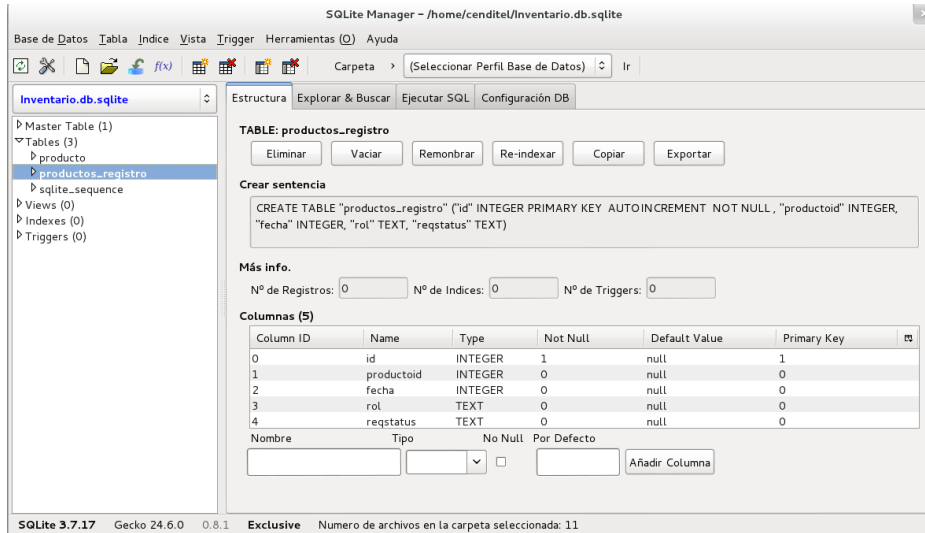
| Column ID | Name     | Type    | Not Null | Default Value | Primary Key |
|-----------|----------|---------|----------|---------------|-------------|
| 0         | id       | INTEGER | 1        | null          | 1           |
| 1         | mensaje  | TEXT    | 0        | null          | 0           |
| 2         | status   | TEXT    | 0        | null          | 0           |
| 3         | cantidad | INTEGER | 0        | null          | 0           |
| 4         | nombre   | TEXT    | 0        | null          | 0           |
| 5         | pedir    | INTEGER | 0        | null          | 0           |

Nombre:  Tipo:  No Null: ☐ Por Defecto:

SQLite 3.7.17 Gecko 24.6.0 0.8.1 Exclusive Numero de archivos en la carpeta seleccionada: 11

Figura 23 Tabla productos.

### 2. Visualizamos la tabla **productos\_registro** creada.



**Figura 24** Tabla productos\_registro.

### ¿Para qué se crean las tablas “productos” y “productos\_registro”?

- La tabla “productos” respresenta la lista de fichas (**token**) en los flujos de trabajo (Wonk-flow).
- La tabla “productos\_registros” representan la lasta de eventos de cambio de estado (status) en los flujos de trabajo.

NOTA

Si ha seguido el tutorial correctamente obtendra los archivos que puede bajar en el enlace que se encuentran abajo

### DOWNLOAD:



mydb.db

---

## Operaciones CRUD+(flujo) utilizando PySafet

---

Las operaciones **CRUD**, son acciones generales que representan un modelo básico para la creación de sistema de información. Puede ser una descripción más detallada en el siguiente enlace [SOBRE CRUD](#).

El simbolo “+” indica las adición de operaciones asociada a flujo de trabajo.

A continuación comenzaremos a ver los pasos de las operaciones CRUB con un archivo que se llama deftrac.xml la cual va a estar en el directorio **\*\*<HOME>.safet/input/\***

### 21.1 1° PRIMER PASO

#### 21.1.1 Escribimos el encabezado (información de Autor).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--
Documento : deftrac.xml
Creado : Fulano de tal
Autor : Fulano de tal
Descripcion: Archivo de Entrada para SAFET - Inflow
-->
```

- Escribimos la ruta al validador XML(formato DTD)

```
<!DOCTYPE operations SYSTEM "file:///home/cenditel/.safet/dtd/yawlinput.dtd">
```

- Ahora podemos empezar a escribir las operaciones siguientes:

- C(Insertar)
- R(Listar\*)
- U(Actualizar)
- D(Borrar o eliminar)
- +(Flujo de trabajo)

```
<operations suffix=":" commandname="operacion">
```

```
<operation name="Productos" desc="Agregar, Modificar, Eliminar, Listar" icon="firma
```

## 2.- Operation Insertar “Productos” C(Insertar)

Las operaciones en PySafet consiste en una lista de comandos que se ejecutan secuencialmente.

Los comandos se componen de campos “**Fields**” que corresponden a los diferentes tipos de datos básicos y complejos. Por ejemplo analicemos el código **XML** del archivo **deftrac.xml** (operación agregar producto). Se define los siguiente:

- 1.- El tipo de comando es “**agregar**”.(**type**), los tipos posibles son “**agregar**”, “**actualizar**” y “**eliminar**”.
- 2.- La tabla de la base de datos donde se realizará el comando “**command**” es “**productos**”.
- 3.- Luego se especificar la lista de campos “**fields**”. Para este campo se especificaran dos campos, el nombre del producto “**Nombre**”, y el estado del producto que tomará el valor literal “**literal**”, “**Registrado**”.
- 4.- El segundo comando de la operación “**Agregar\_producto**” también es del tipo “**Agregar**” y ojo los campos que son necesarios para registrar el eventos “**Agregar\_producto**” en la tabla “**productos\_registro**”.
- 5.- La palabra de PySafet “**\_USERNAME**” se utiliza para obtener el usuario actual

A continuación se mostrara el archivo **deftrac.xml**:

```
<operation name="agregar_producto" desc="" icon="plus.png">
 <command id ="1" type="agregar" table="productos" >
 <fields>
 <field type="string" icon="resumen.png" mandatory="yes" validation=""
 title="Nombre" desc="">
 nombre
 </field>
 <field type="string" literal="Registrado" mandatory="yes" >
 status
 </field>
 </fields>
 </command>
 <command id ="1" type="agregar" table="productos_registro" >
 <fields>
 <field type="datetime" mandatory="yes"
 function="seq from sqlite_sequence where name='productos' " input="no">
 productoid
 </field>
```

```
<field type="datetime" mandatory="yes" function="datetime('now') "
 format="time_t" input="no">
fecha
</field>

<field type="string" literal="_USERNAME" mandatory="yes" >
rol
</field>

<field type="string" literal="Registrado" mandatory="yes" >
regstatus
</field>

</fields>

</command>

</operation>
```

#### ■ Ejecutamos el Script para Insertar un producto

Seguidamente vamos a utilizar esta operación “**agregar\_producto**” en un Script de python como se muestra a continuación:

##### ■ Operación: agregar\_producto

##### ■ Nombre: El nombre del Producto agregar

```
-*- coding: utf-8 -*-

D(Borrar o eliminar)
myconsult = u"operacion:borrar_producto id:5"

U(Actualizar)
#myconsult = u"operacion:modificar_producto id:3 Nombre:Amoxicilina"

+(Flujo de trabajo)
myconsult = u"operacion:Actualizar_producto id:3 Estado_producto:Pedido"

Importación de la librería Safet y os
import Safet
import os

Aqui obtengo mi home,media y url
myhome = os.getenv("HOME")
mymedia = myhome + "/tmp"
myurl = "http://localhost"

Constructor principal
myinflow = Safet.MainWindow(myhome)

myinflow.setMediaPath(mymedia)
```

```
myinflow.setHostURL(myurl)

Si no es un usuario registrado el metodo "login" retorna False
result = myinflow.login("admin", "admin")

C(Insertar)
Agregamos el producto a ingresar por ejemplo "Champu Olorin",
"ComplejoB", "Aspirina", "Acetaminofén", "Ibuprofeno".

myconsult = u"operacion:agregar_producto Nombre: ComplejoB"

if result:
 result = myinflow.toInputForm(myconsult)
else:
 print "\n ---Usuario autenticado---\n"
 exit()

if result:
 print "\n --Se Actualizo correctamente el producto---\n"
else:
 print "\n No se Actualizo el producto....!!!\n"

if not result:
 print "\nConsulta failed error: %s\n" % (myinflow.currentError())
 exit()

print u" %s\n" % (myinflow.currentJSON())
```

Crearemos un archivo ".py" con cualquier nombre y copiamos el Script y lo ejecutamos de la siguiente manera

```
$ python Script_Insertar_producto.py
```

- Observen la siguiente imagen:

```
cenditel@debian:~$ python Script_Insertar_producto.py
QFSFileEngine::open: No file name specified
```

```
--Se inserto correctamente el producto---
```

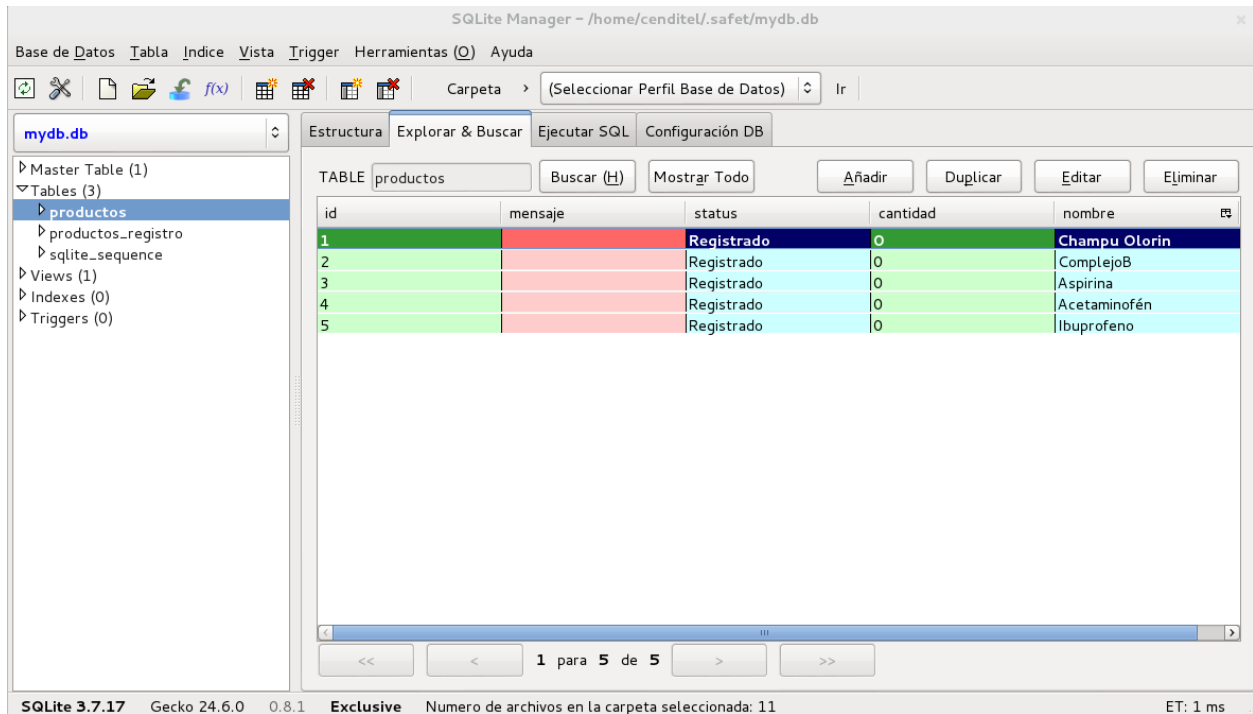
```
{ "ticket": "n/t", "result": "false" }
```

```
cenditel@debian:~$ █
```

**Figura 25** Insertar producto.

- Observen la base de datos en la siguiente imagen:





SQLite Manager - /home/cenditel/safet/mydb.db

Base de Datos Tabla Índice Vista Trigger Herramientas (O) Ayuda

mydb.db

Master Table (1)  
Tables (3)  
  ↳ productos  
  ↳ productos\_registro  
  ↳ sqlite\_sequence  
Views (1)  
Indexes (0)  
Triggers (0)

ESTRUCTURA Explorar & Buscar Ejecutar SQL Configuración DB

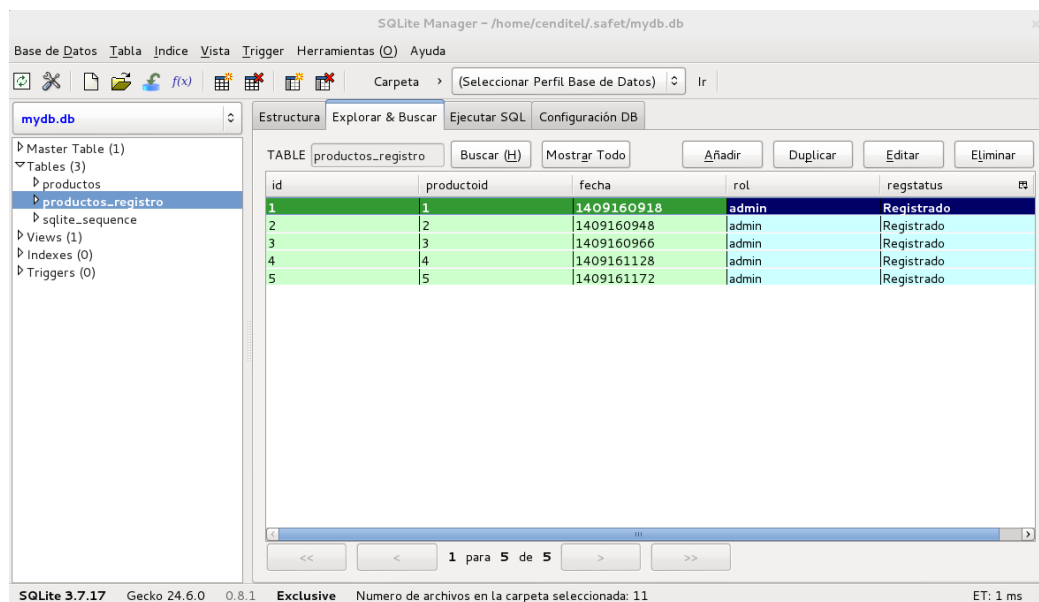
TABLE productos Buscar (H) Mostrar Todo Añadir Duplicar Editar Eliminar

id	mensaje	status	cantidad	nombre
1		Registrado	0	Champu Olorin
2		Registrado	0	ComplejoB
3		Registrado	0	Aspirina
4		Registrado	0	Acetaminofén
5		Registrado	0	Ibuprofeno

1 para 5 de 5

SQLite 3.7.17 Gecko 24.6.0 0.8.1 Exclusive Numero de archivos en la carpeta seleccionada: 11 ET: 1 ms

Figura 21.1: Figura 26: Tabla productos



SQLite Manager - /home/cenditel/safet/mydb.db

Base de Datos Tabla Índice Vista Trigger Herramientas (O) Ayuda

mydb.db

Master Table (1)  
Tables (3)  
  ↳ productos  
  ↳ productos\_registro  
  ↳ sqlite\_sequence  
Views (1)  
Indexes (0)  
Triggers (0)

ESTRUCTURA Explorar & Buscar Ejecutar SQL Configuración DB

TABLE productos\_registro Buscar (H) Mostrar Todo Añadir Duplicar Editar Eliminar

id	productoid	fecha	rol	regstatus
1	1	1409160918	admin	Registrado
2	2	1409160948	admin	Registrado
3	3	1409160966	admin	Registrado
4	4	1409161128	admin	Registrado
5	5	1409161172	admin	Registrado

1 para 5 de 5

SQLite 3.7.17 Gecko 24.6.0 0.8.1 Exclusive Numero de archivos en la carpeta seleccionada: 11 ET: 1 ms

Figura 27 Tabla productos\_registro.

### 3.- Operation Eliminar “Productos” D(Borrar o eliminar)

```
<operation name="borrar_producto" desc="Elimina un ticket por id" icon="clear.png">
```

```
<command id ="1" type="eliminar" table="productos">
```

```
<fields>
```

```

 <field type="combolisttable" options="id:productos::id ||
 mandatory="yes" pr
 id
 </field>

</fields>

</command>

<command id ="1" type="eliminar" table="productos_registro">

 <fields>

 <field type="string" mandatory="yes" title="id" >
 productoid
 </field>

 </fields>

</command>

</operation>

```

### ■ Ejecutamos el Script para elimiar el producto

Seguidamente vamos a utilizar esta operación “**borrar\_producto**” en un Script de python como se muestra a continuación:

#### ■ Operación: borrar\_producto

- **id:** Aqui colocaremos el valor de **id** del producto por ejemplo borraremos el producto **Ibuprofeno** y su **id** es **5**. Observe la *Figura 26: Tabla productos*

```

-*- coding: utf-8 -*-

C(Insertar)
Agregamos el poducto a ingresar por ejemplo "Champu Olorin",
"ComplejoB", "Aspirina", "Acetaminofén", "Ibuprofeno".
#myconsult = u"operacion:agregar_producto Nombre: ComplejoB"

U(Actualizar)
#myconsult = u"operacion:modificar_producto id:3 Nombre:Amoxicilina"

+(Flujo de trabajo)
myconsult = u"operacion:Actualizar_producto id:3 Estado_producto:Pedido"

Importación de la librería Safet y os
import Safet
import os

Aqui obtengo mi home,media y url
myhome = os.getenv("HOME")

```

```

mymedia = myhome + "/tmp"
myurl = "http://localhost"

Constructor principal
myinflow = Safet.MainWindow(myhome)

myinflow.setMediaPath(mymedia)
myinflow.setHostURL(myurl)

Si no es un usuario registrado el metodo "login" retorna False
result = myinflow.login("admin", "admin")

D(Borrar o eliminar)
Eliminamos el porducto numero 5
myconsult = u"operacion:borrar_producto id:5"

if result:
 result = myinflow.toInputForm(myconsult)
else:
 print "\n ---Usuario autenticado---\n"
 exit()

if result:
 print "\n --Se Actualizo correctamente el producto---\n"
else:
 print "\n No se Actualizo el producto....!!!\n"

if not result:
 print "\nConsulta failed error: %s\n" % (myinflow.currentError())
 exit()

print u" %s\n" % (myinflow.currentJSON())

$ python Eliminar_producto.py

```

Observen la siguiente imagen:

```

cenditel@debian:~/cursoSafet$ python Eliminar_producto.py
QFSFileEngine::open: No file name specified

--Se borro correctamente el producto--

{ "ticket": "n/t", "result": "false" }

cenditel@debian:~/cursoSafet$ █

```

**Figura 27** Script Eliminar producto.

- **Observen en la db en la siguiente imagen:** Observe la *Figura 26: Tabla productos*

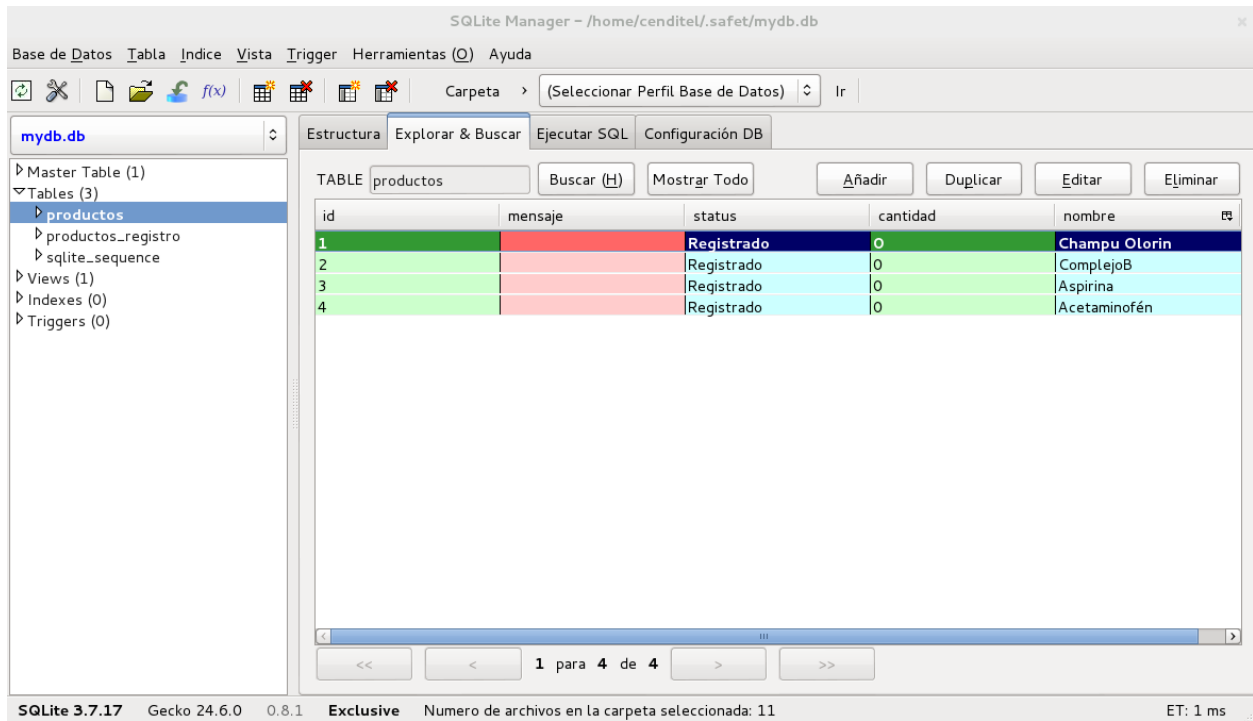


Figura 21.2: Figura 26: Tabla productos

- **Observen en la db en la siguiente imagen:** Observe la *Figura 27: Tabla productos\_registro*

#### 4.- Operation Actualizar “Nombre del Producto” U(Actualizar)

```
<operation name="modificar_producto" desc="" icon="plus.png">
```

```
<command id="1" type="actualizar" table="productos" >
```

```
<fields>
```

```
<field type="combolisttable" options="id:productos:id ||
mandatory="yes" primarykey="yes" t
```

```
id
</field>
```

```
<field type="string" icon="resumen.png" mandatory="yes" va
title="Nombre" des
```

```
nombre
</field>
```

```
</fields>
```

```
</command>
```

```
</operation>
```

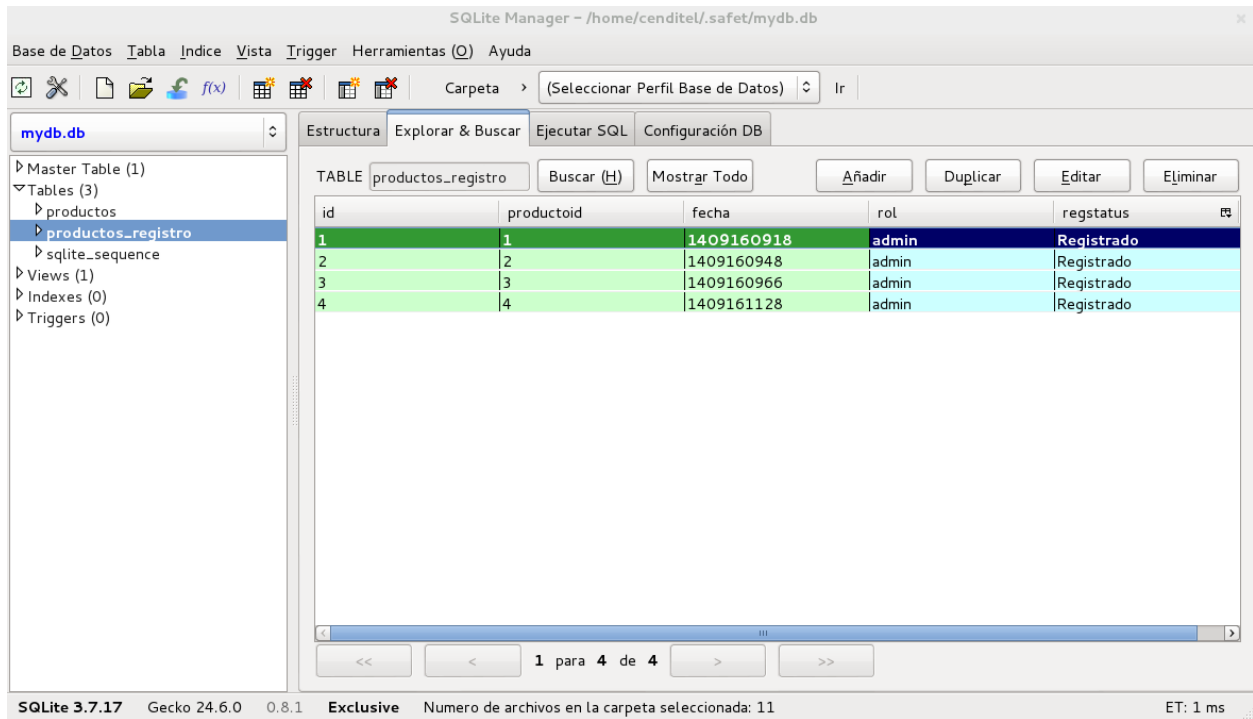


Figura 21.3: Figura 27: Tabla productos\_registro

#### ■ Ejecutamos el Script para Actualizar el nombre producto

Seguidamente vamos a utilizar esta operación “**Actualizar\_producto**” en un Script de python como se muestra a continuación:

- **Operación:** modificar\_producto
- **id:** Aqui colocaremos el valor de **id** del producto por ejemplo vamos a actualizar el producto **Aspirina** y su **id** es **3**. Observe la *Figura 26: Tabla productos*
- **Nombre:** Aqui se coloca el nombre al que le vamos a modificar por ejemplo **Amoxicilina** por **Aspirina**.

```
-*- coding: utf-8 -*-

C(Insertar)
Agregamos el producto a ingresar por ejemplo "Champu Olorin",
"ComplejoB", "Aspirina", "Acetaminofén", "Ibuprofeno".
#myconsult = u"operacion:agregar_producto Nombre: ComplejoB"

D(Borrar o eliminar)
Eliminamos el producto numero 5
#myconsult = u"operacion:borrar_producto id:5"

+(Flujo de trabajo)
myconsult = u"operacion:Actualizar_producto id:3 Estado_producto:Pedido"
```

```
Importación de la librería Safet y os
import Safet
import os

Aqui obtengo mi home, media y url
myhome = os.getenv("HOME")
mymedia = myhome + "/tmp"
myurl = "http://localhost"

Constructor principal
myinflow = Safet.MainWindow(myhome)

myinflow.setMediaPath(mymedia)
myinflow.setHostURL(myurl)

Si no es un usuario registrado el metodo "login" retorna False
result = myinflow.login("admin", "admin")

U(Actualizar)
Se actualizara el nombre del producto número 3
myconsult = u"operacion:modificar_producto id:3 Nombre:Amoxicilina"

if result:
 result = myinflow.toInputForm(myconsult)
else:
 print "\n ---Usuario autenticado---\n"
 exit()

if result:
 print "\n --Se Actualizo correctamente el producto---\n"
else:
 print "\n No se Actualizo el producto....!!!\n"

if not result:
 print "\nConsulta failed error: %s\n" % (myinflow.currentError())
 exit()

print u" %s\n" % (myinflow.currentJSON())

$ python Modificar_producto.py
```

Observen la siguiente imagen:

```
cenditel@debian:~$ python Script.py
QFSFileEngine::open: No file name specified

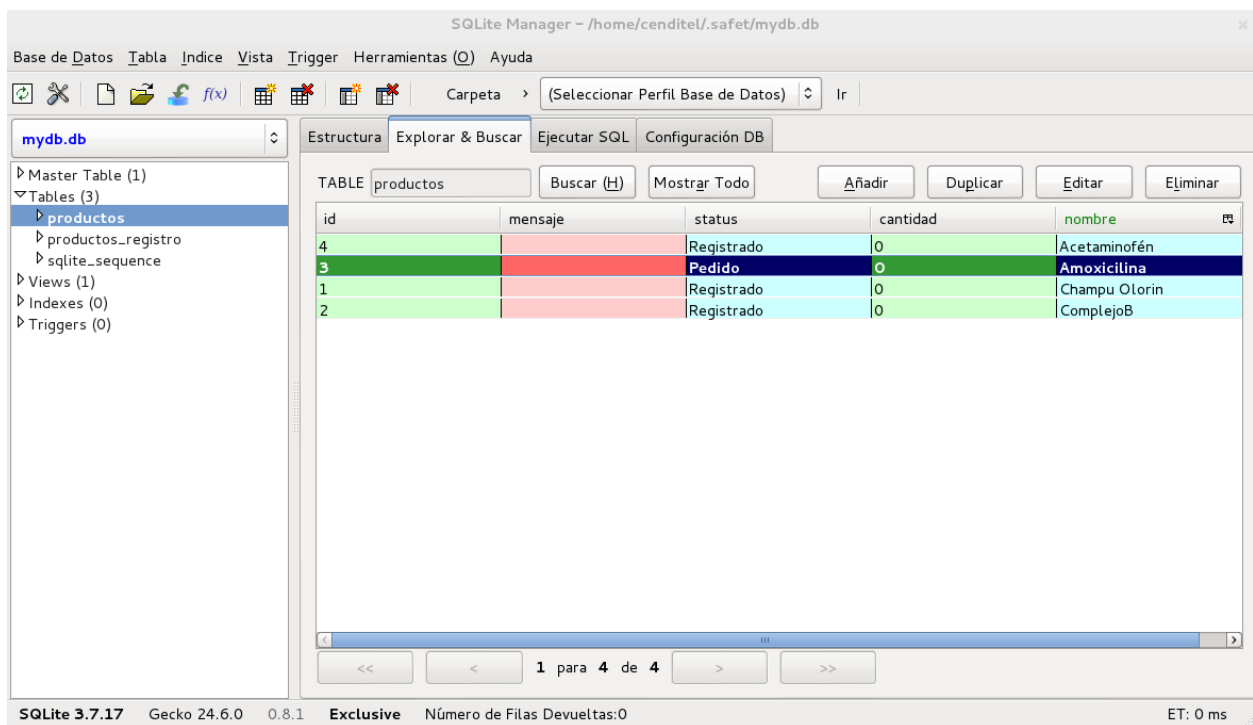
--Se Actualizo correctamente el producto--

{ "ticket": "n/t", "result": "false" }

cenditel@debian:~$
```

**Figura 29** Script Actualizar producto.

- **Observen en la db en la siguiente imagen:** Observe la *Figura 26: Tabla productos*



**Figura 21.4: Figura 26: Tabla productos**

## 5.- Operation Actualizar “Productos” +(Flujo de trabajo)

```
<operation name="Actualizar_producto" desc="Pasa de estado un determinado ticket">

 <command id ="1" type="actualizar" table="productos">

 <fields>

 <field type="combolisttable" options="id:productos::'(' ||
 mandatory="yes" primaryKey
 id
```

```

 </field>

 <field type="comboflow" mandatory="yes" options="next "
 path="/home/cenditel/.safet/flowfiles/productos.xml"
 status=
 </field>

 </fields>
</command>

<command id ="1" type="agregar" table="productos_registro" >

 <fields>

 <field type="datetime" mandatory="yes" function="seq from
 productoid
 </field>

 <field type="datetime" mandatory="yes" function="datetime
 format="time_t" in
 fecha
 </field>

 <field type="string" literal="_USERNAME" mandatory="yes"
 rol
 </field>

 <field type="string" title="Status" mandatory="yes" >
 regstatus
 </field>

 </fields>

</command>

</operation>

```

#### ■ Ejecutamos el Script para Actualizar el Status del producto

Seguidamente vamos a utilizar esta operación “**Actualizar\_producto**” en un Script de python como se muestra a continuación:

- **Operación:** Actualizar\_producto
- **id:** Aqui colocaremos el valor de **id** del producto por ejemplo vamos a actualizar el producto **Aspirina** y su **id** es **3**. Observe la *Figura 26: Tabla productos*
- **Estado\_producto:** Aqui se coloca el estado del producto es decir si es por llegar, Agotarse, Pedido, En Espera. En este caso esta **Registrado** vamos a modificarlo a **pedido**.

```
-*- coding: utf-8 -*-
```



```
C(Insertar)
Agregamos el podocto a ingresar por ejemplo "Champu Olorin",
"ComplejoB", "Aspirina", "Acetaminofén", "Ibuprofeno".
#myconsult = u"operacion:agregar_producto Nombre: ComplejoB"

D(Borrar o eliminar)
Eliminamos el porducto numero 5
#myconsult = u"operacion:borrar_producto id:5"

U(Actualizar)
Se actualizara el nombre del producto número 3
#myconsult = u"operacion:modificar_producto id:3 Nombre:Amoxicilina"

Importación de la librería Safet y os
import Safet
import os

Aqui obtengo mi home,media y url
myhome = os.getenv("HOME")
mymedia = myhome + "/tmp"
myurl = "http://localhost"

Constructor principal
myinflow = Safet.MainWindow(myhome)

myinflow.setMediaPath(mymedia)
myinflow.setHostURL(myurl)

Si no es un usuario registrado el metodo "login" retorna False
result = myinflow.login("admin", "admin")

+(Flujo de trabajo)
Se actualizara su estado
myconsult = u"operacion:Actualizar_producto id:3 Estado_producto:Pedido"

if result:
 result = myinflow.toInputForm(myconsult)
else:
 print "\n ---Usuario autenticado---\n"
 exit()

if result:
 print "\n --Se Actualizo correctamente el producto---\n"
else:
 print "\n No se Actualizo el producto....!!!\n"

if not result:
```

```
print "\nConsulta failed error: %s\n" % (myinflow.currentError())
exit()

print u" %s\n" % (myinflow.currentJSON())

$ python Actualizar_Status_producto.py
```

Observen la siguiente imagen:

```
cenditel@debian:~/cursoSafet$ python Modificar_producto.py
QFSFileEngine::open: No file name specified

--Se Actualizo correctamente el producto--

{ "ticket": "n/t", "result": "false" }

cenditel@debian:~/cursoSafet$
```

Figura 29 Script Actualizar producto.

- Observen en la db en la siguiente imagen: Observe la *Figura 26: Tabla producto*

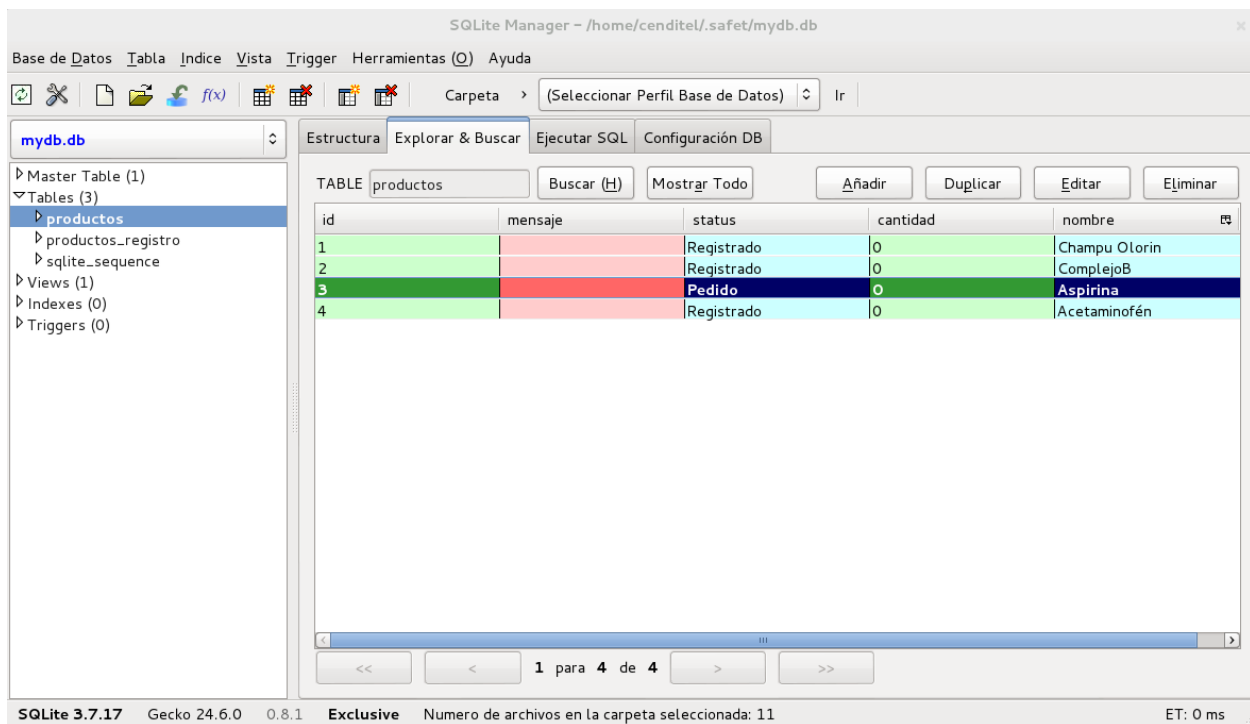
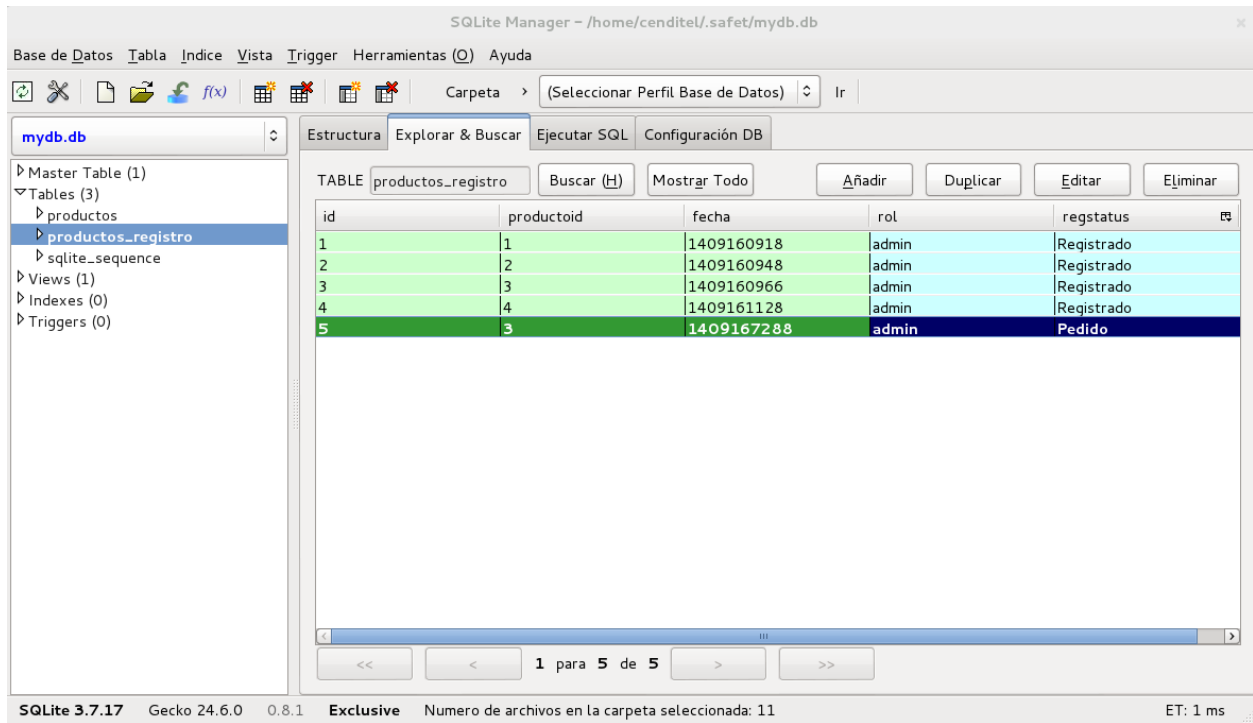


Figura 21.5: Figura 26: Tabla producto

- Observen en la db en la siguiente imagen: Observe la *Figura 27: Tabla producto\_registro*



SQLite Manager - /home/cenditel/safet/mydb.db

Base de Datos Tabla Índice Vista Trigger Herramientas (O) Ayuda

mydb.db

Master Table (1)  
Tables (3)  
  ▶ productos  
  ▶ productos\_registro  
  ▶ sqlite\_sequence  
Views (1)  
Indexes (0)  
Triggers (0)

ESTRUCTURA Explorar & Buscar Ejecutar SQL Configuración DB

TABLE productos\_registro Buscar (H) Mostrar Todo Añadir Duplicar Editar Eliminar

id	productoid	fecha	rol	regstatus
1	1	1409160918	admin	Registrado
2	2	1409160948	admin	Registrado
3	3	1409160966	admin	Registrado
4	4	1409161128	admin	Registrado
5	3	1409167288	admin	Pedido

1 para 5 de 5

SQLite 3.7.17 Gecko 24.6.0 0.8.1 Exclusive Numero de archivos en la carpeta seleccionada: 11 ET: 1 ms

Figura 21.6: Figura 27: Tabla producto\_registro

**Nota:**

Si tienen error aquí están los archivos

- Archivo deftrac:

**DOWNLOAD:**



deftrac.xml

- Archivo Script:

**DOWNLOAD:**



Script.py



---

## Crear el flujo de trabajo utilizando un archivo XML

---

El flujo de trabajo (workflow) es el estudio de los aspectos operacionales de una actividad de trabajo: cómo se estructuran las tareas, cómo se realizan, cuál es su orden correlativo, cómo se sincronizan, cómo fluye la información que soporta las tareas y cómo se le hace seguimiento al cumplimiento de las tareas. Más información [AQUI](#).

A continuación seguimos los siguientes paso:

### 22.1 1° PRIMER PASO

- Crearemos un archivo llamado **productos.xml**.
- Lo guardamos en el siguiente directorio **<HOME>.safet/flowfiles/**.

### 22.2 2° SEGUNDO PASO

#### 22.2.1 Encabezado

Abrimos el archivo **productos.xml** y Escribimos el siguiente encabezado (XML) que aparece en el siguiente block:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE yawl SYSTEM 'file:///home/cenditel/.safet/dtd/yawlworkflow.dtd'>

<!--
Documento : productos.xml
Creado : 16/10/08 09:27 AM
Autor : nombre_autor
Descripcion: Archivo generado por plantilla de la Libreria SAFET
-->
```

---

**Nota:** El comentario es opcional.

---

## 22.3 3° TERCER PASO

### 22.3.1 Etiquetas Principales

En el archivo **productos.xml** se utilizaran las siguientes **etiquetas principales (XML)**:

- **<yawl>**: Etiqueta principal yawl.
- **<workflow>**: Nombre del flujo.
- **<token>**: Ficha: Nombre de la tabla y Nombre del campo.

Abrimos el archivo **productos.xml** y debajo del *Encabezado* copiamos el siguiente código (XML) que aparece en el siguiente block:

```
<yawl version="0.01">
 <workflow id="productos">
 <token keysource="productos" key="id"/>

 <!-- **CUERPO**
 Aqui vamos a colocar todos los estados o tareas.
 #####
 ## - INICIAL ##
 ## - REGISTRADO. ##
 ## - PEDIDO ##
 ## - DISPONIBLE ##
 ## - POR_LLEGAR ##
 ## - POR_AGOTARSE ##
 ## - AGOSTADO ##
 ## - FINAL ##
 #####
 -->

 </workflow>
</yawl>
```

---

**Nota:**

- EL cuadro se me muestra comentado es opcional solo para visualizar como van hacer las tareas.
  - Todas las tareas o estados van donde esta el cuadro opcional.
- 

## 22.4 4° CUARTO PASO

### 22.4.1 Condiciones Principales (INICIAL Y FINAL)

En las **condiciones principal (inicial y final)** se utilizaran las siguientes **etiquetas (XML)**:

- **<condition>inicial:** Condición inicial
- **<condition>final:** Condición final
- **<port> inicial:** Dentro de esta etiqueta van las conexiones continene una conexión.
- **<port> final:** No tiene a quien conectarse.
- **<connection> inicial:** Como no hay nada registrado inicio apunta a final
- **<connection> final:** final no apunta a nada

Abrimos el archivo **productos.xml** y dentro de las *Etiquetas Principales* insertamos el siguiente código (XML) que aparece en el siguiente block:

```
<!--
#####
Condición inicial
#####
-->
<condition type="start" id="inicial">
 <port side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>
</condition>

<!--
#####
Condición final
#####
-->
<condition id="final">
 <port side="forward" type="split">
 <connection source=""/>
 </port>
</condition>
```

Ahora ejecutaremos los siguiente pasos:

- Crearemos una carpeta llamada **tmp** en directorio **\$HOME/tmp**, desde la consola de comando.

```
$ mkdir $HOME/tmp
```

- Crearemos un archivo con extensión **.py** en directorio **\$HOME**, desde la consola de comando.

```
$ touch $HOME/Script_graficos.py
```

- Abrimos el archivo **.py** que creamos en el paso anterior la cual lo llamamos **Script\_graficos.py** y copiamos el siguiente Script\*\*(python)\*\*:

```
-*- coding: utf-8 -*-

import Safet
import os
```

```
myhome = os.getenv("HOME")
mymedia = myhome + "/tmp"
myurl = "http://localhost"

myinflow = Safet.MainWindow(myhome)

myinflow.setMediaPath(mymedia)
myinflow.setHostURL(myurl)

result = myinflow.login("admin", "admin")

myconsult = u"operacion:Generar_gráfico_coloreado \
Cargar_archivo_flujo: %s/.safet/flowfiles/productos.xml" % (myhome)

if not result:
 print "Authentication failed"
 exit()

result = myinflow.toInputConsole(myconsult)

if not result:
 print "Consult failed error: %s" % (myinflow.currentError())
 exit()

print u"%s" % (myinflow.currentJSON())
```

- Ejecutamos el archivo **Script\_graficos.py**, desde la consola de comando como usuario normal.

```
$ python $HOME/Script_graficos.py
```

---

**Nota:** Al ejecutar el archivo **.py (Script\_graficos.py)** nos mostrara un mensaje donde salen reflejadas las 2 *Condiciones Principales (INICIAL Y FINAL)*:

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db'
is still in use, all queries will cease to work.
.....wheretokens: on
.....newnode: |inicial| # Nueva condición inicial
.....newnode: |final| # Nueva condición final
qt_temp.XM6827.svg # Obtenemos la nueva imagen con su nombre la cual co
```

**El nombre de la imagen (.svg) es temporal, es decir su nombre varia constante mente.**

---

- Nos vamos al directorio **\$HOME/tmp**, desde la consola de comando.

```
$ cd $HOME/tmp
```

---

**Nota:** En el directorio **\$HOME/tmp** se escriben los archivos de grafos **(.svg o .png)**,



usted puede definir el directorio de escritura y temporal utilizando los siguientes parámetros en el archivo **safet.conf** que se encuentra en el directorio **\$HOME/.safet/**.

```
plugins.graphviz.infile = /home/fulano/tmp # directorio para archivos temporales
plugins.graphviz.outfile = /home/fulano/tmp # directorio de salida para archivos
```

- Escribimos el comando **ls** para ver las imágenes **.svg**, desde la consola de comando.

```
tmp$ ls
qt_temp.XM7929.svg
```

- Con el comando **eog** vemos la primera imagen **.svg(qt\_temp.XM6827.svg )** , desde la consola de comando.

```
tmp$ eog qt_temp.XM6827.svg
```

**Nota:** Si realizó los pasos correctos se mostrara el grafo, en el cual se define lo siguiente:

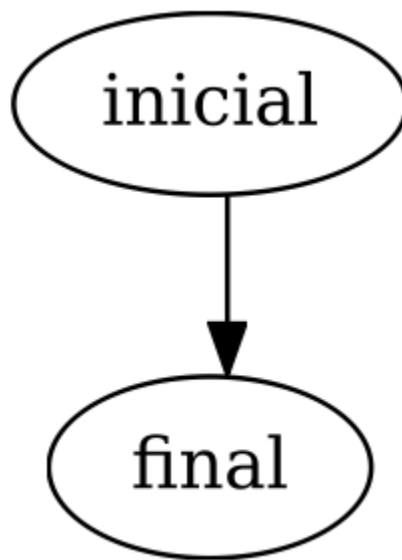


Figura 22.1: **Figura 28: Condiciones (inicial y final).**

Descripción de la *Figura 28: Condiciones (inicial y final).*:

- El círculo **inicial** apunta al círculo **final**.
- En este inventario esta vacío.
- En esta imagen vemos como se conectan las condiciones **inicial y final**.

**Nota:** En esta imagen no se muestran ninguna tarea ya que no hemos realizado

ninguna, para ello pasamos al *5° QUINTO PASO* donde comenzaremos a realizar las tareas.

---

## 22.5 5° QUINTO PASO

### 22.5.1 5.1 - PRIMERA TAREA (Registrado)

En la primera tarea (**Registrado**) se utilizaran las siguientes **etiquetas (XML)**:

- **<connection>**: **Registrado** apunta a **final**, esto varia.
- **<task>**: Nombre de la tarea (**Registrado**) y su mensaje.
- **<port>**: Registrado apunta a una opción.
- **<variable>**: Variable **vRegistro** donde me aparecerá un mensaje la hora en el cual se hizo ese registro.

Abrimos el archivo **productos.xml** y insertamos el código (**xml**) de la **tarea Registrado** que aparece en el siguiente block:

---

**Nota:** La condición **inicial** se modifiko en la etiqueta **<connection>** la cual apuntará la tarea **Registrado**.

---

```
<!--

| Condición inicial |

-->
<condition type="start" id="inicial">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Registrado" source="Regi
 </port>
</condition>

<!--

| Registrado |

-->
<task title="en inventario" id="Registrado">
 <port side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>
<variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vRegistro" rolfield="(select rol from productos_registro
 where productoid=productos.id and regstatus='Registrado') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Registrado') as fecha"/>
```

</task>

```
<!--

| Condición final |

-->
<condition id="final">
 <port side="forward" type="split">
 <connection source=""/>
 </port>
</condition>
```

Ahora ejecutaremos los siguiente pasos:

---

**Nota:** En el 4° CUARTO PASO creamos la carpeta llamada **tmp** y el archivo llamado **.py** (**Script\_graficos.py**) ,la cual se utilizaran en este paso a seguir.

---

- Ejecutamos el mismo archivo **.py** (**Script\_graficos.py**) con el mismo contenido, desde la consola de comando como usuario normal.

```
$ python $HOME/Script_graficos.py
```

---

**Nota:** Al ejecutar el archivo **.py** (**Script\_graficos.py**) nos mostrara un mensaje donde salen reflejadas las 2 **Condiciones principales** y la primera tarea (**Registrado**):

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db'
is still in use, all queries will cease to work.
.....wheretokens: on
.....newnode: |Registrado| # Nueva tarea Registrado
.....newnode: |inicial| # Condición inicial
.....newnode: |final| # Condición final
qt_temp.XM6827.svg # Obtenemos la nueva imagen con su nombre la cual
```

**El nombre de la imagen (.svg) es temporal, es decir su nombre varia constante mente.**

---

**Nota:** El nombre de la imagen (.svg) es temporal, es decir su nombre varia constante mente.

---

- Nos vamos al directorio **\$HOME/tmp**,desde la consola de comando.

```
$ cd $HOME/tmp
```

- Escribimos el comando **ls** para ver las 2 imágenes **.svg** que hemos obtenido ,desde la consola de comando.

```
tmp$ ls
qt_temp.XM6827.svg , qt_temp.XM6970.svg
```

- Con el comando **eog** vemos la la segunda imagen **.svg(qt\_temp.XM6827.svg )** ,desde la consola de comando.

```
tmp$ eog qt_temp.XM6970.svg
```

---

**Nota:** Si realizó los pasos correctos, se mostrara el grafo como en la imagen siguiente *Figura 29: Registrado*:

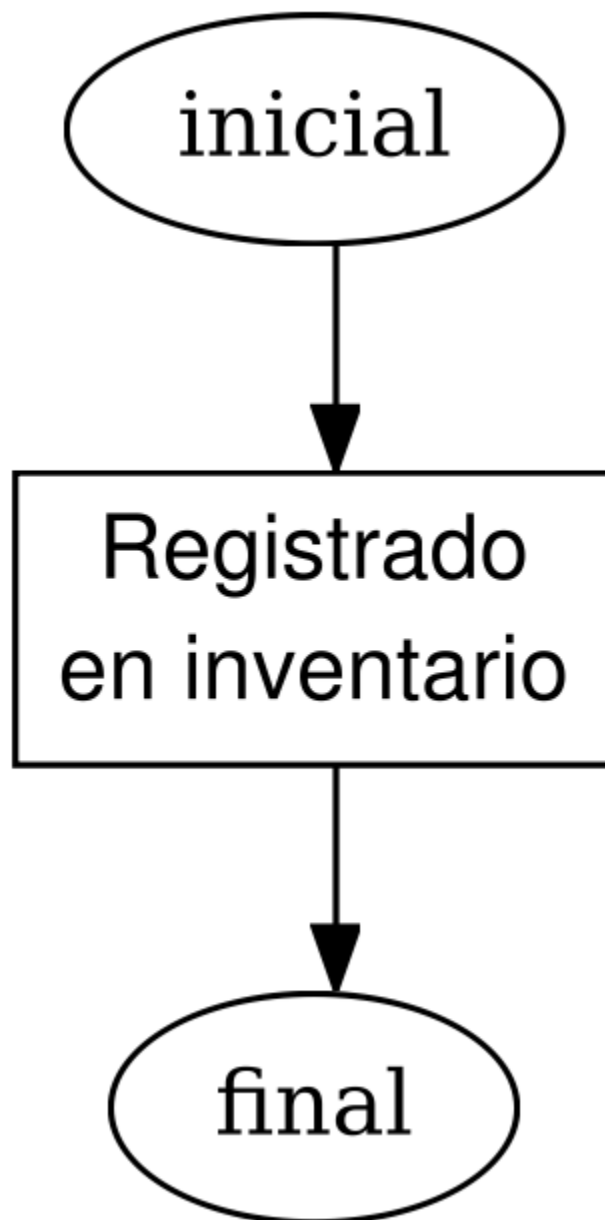


Figura 22.2: **Figura 29: Registrado**

Descripción de la *Figura 29: Registrado*:

- La condición del circulo **inicial** apunta a la primera tarea de cuadro **Registrado**.
- La primera tarea **Registrado** apunta a la condición de circulo **final**
- En esta imagen vemos como se conectan la condición **inicial** con la primera tarea **Registrado** y la tarea con la condición **final**.

## 22.5.2 5.2 - SEGUNDA TAREA (Pedido)

En la segunda tarea (**Pedido**) se utilizaran las siguientes **etiquetas (XML)**:

- **<connection>**: **Pedido** apunta a **final**, esto varia.
- **<task>**: Nombre de la tarea (**Pedido**)
- **<port>**: **Pedido** apunta a dos opciones con el operador **OR**.
- **<variable>**: Variable **vPedido** donde me aparecerá un mensaje la hora en el cual se hizo ese pedido.

Abrimos el archivo **productos.xml** y insertamos el código (**xml**) de la **tarea Pedido** que aparece en el siguiente block:

**Nota:** La tarea **Registrado** se modifiko en la etiqueta **<connection>** la cual apuntará a la tarea **Pedido**.

```
<!--

| Condición inicial |

-->
<condition type="start" id="inicial">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Registrado" source="Regi
 </port>
</condition>

<!--

| Registrado |

-->
<task title="en inventario" id="Registrado">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Pedido" source="Pedido"/
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vRegistrado" rolfield="(select rol from productos_registro
```

```
where productoid=productos.id and regstatus='Registrado') as rol" scope="task"
timestampfield="(select fecha from productos_registro
where productoid=productos.id and regstatus='Registrado') as fecha"/>
</task>

<!--

| Pedido |

-->
<task title="" id="Pedido" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPedido" rolfield="(select rol from
productos_registro
 where productoid=productos.id and regstatus='Pedido') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Pedido') as fecha"/>
</task>

<!--

| Condición final |

-->
<condition id="final">
 <port side="forward" type="split">
 <connection source=""/>
 </port>
</condition>
```

Ahora ejecutaremos los siguiente pasos:

---

**Nota:** En el 4º *CUARTO PASO* creamos la carpeta llamada **tmp** y el archivo llamado **.py** (**Script\_graficos.py**) ,la cual se utilizaran en este paso a seguir.

---

- Ejecutamos el mismo archivo **.py** (**Script\_graficos.py**) con el mismo contenido, desde la consola de comando como usuario normal.

```
$ python $HOME/Script_graficos.py
```

---

**Nota:** Al ejecutar el archivo **.py** (**Script\_graficos.py**) nos mostrara un mensaje donde salen reflejadas las 2 **Condiciones principales** y la dos tarea (**Registra-do**),(**Pedido**):

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection ' /home/cenditel/.safet/mydb.db'
is still in use, all queries will cease to work.
.....wheretokens: on
```

```

.....newnode: |Pedido| # Nueva tarea Pedido
.....newnode: |Registrado| # Tarea Registrado
.....newnode: |inicial| # Condición inicial
.....newnode: |final| # Condición final
qt_temp.XM5792.svg # Obtenemos la nueva imagen con su nombre la cual

```

**El nombre de la imagen (.svg) es temporal, es decir su nombre varia constantemente.**

---

**Nota:** El nombre de la imagen (.svg) es temporal, es decir su nombre varia constantemente.

---

- Nos vamos al directorio **\$HOME/tmp**, desde la consola de comando.

```
$ cd $HOME/tmp
```

- Escribimos el comando **ls** para ver las 3 imágenes **.svg** que hemos obtenido ,desde la consola de comando.

```
tmp$ ls
qt_temp.XM6827.svg, qt_temp.XM6970.svg, qt_temp.XM5792.svg
```

- Con el comando **eog** vemos la tercera imagen **.svg(qt\_temp.XM5792.svg)** ,desde la consola de comando.

```
tmp$ eog qt_temp.XM5792.svg
```

---

**Nota:** Si realizó los pasos correctos, se mostrara el grafo como en la imagen siguiente *Figura 30: Pedido:*

Descripción de la *Figura 30: Pedido:*

- La condición del circulo **inicial** apunta a la primera tarea de cuadro **Registrado**.
  - La primera tarea **Registrado** apunta a la segunda tarea **Pedido**.
  - La segunda tarea apunta a la condición de circulo **final**
  - En esta imagen vemos como se conectan la condición **inicial** con la primera tarea **Registrado**, la tarea primera tarea con la segunda tarea **Pedido** y la segunda tarea con la condición **final**.
- 

### 22.5.3 5.3 - TERCERA TAREA (Disponible)

En la tercera tarea (**Disponible**) se utilizaran las siguientes **etiquetas (XML)**:

- **<task>**: Nombre de la tarea (**Diponible**).
- **<port>**: Disponible apunta a una opción.

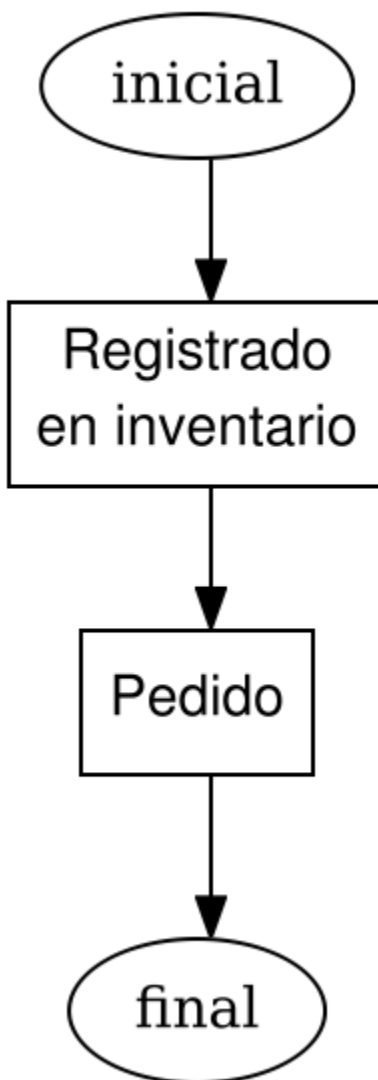


Figura 22.3: **Figura 30: Pedido**



- **<connection>**: **Disponible** apunta a **final**, esto varia.
- **<variable>**: Variable **vDisponible** donde nos aparecerá (**nombre,id,status,fecha y hora**) de esa acción.

Abrimos el archivo **productos.xml** y insertamos el código (xml) de la **tarea Disponible** que aparece en el siguiente block:

**Nota:** La tarea **Pedido** se modifiko las etiqueta **<port>** donde indicará que habran 2 opciones a ocurrir y la etiqueta **<connection>** la cual apuntará a la tarea **Disponible**.

```
<!--

| Condición inicial |

-->
<condition type="start" id="inicial">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Registrado" source="Regi
 </port>
</condition>

<!--

| Registrado |

-->
<task title="en inventario" id="Registrado">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Pedido" source="Pedido"/
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vRegistrado" rolfield="(select rol from productos_registro
 where productoid=productos.id and regstatus='Registrado') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Registrado') as fecha"/>
</task>

<!--

| Pedido |

-->
<task title="" id="Pedido" textualinfo="">
 <port pattern="or" side="forward" type="split">
 <connection query="select status from productos" options="Disponible" source="Disp
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPedido" rolfield="(select rol from
productos_registro
 where productoid=productos.id and regstatus='Pedido') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
```

```
 where productoid=productos.id and regstatus='Pedido') as fecha"/>
</task>

<!--

| Disponible |

-->
<task title="" id="Disponible" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vDisponible" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Disponible') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Disponible') as fecha"/>
</task>

<!--

| Condición final |

-->
<condition id="final">
 <port side="forward" type="split">
 <connection source=""/>
 </port>
</condition>
```

Ahora ejecutaremos los siguiente pasos:

---

**Nota:** En el 4º *CUARTO PASO* creamos la carpeta llamada **tmp** y el archivo llamado **.py** (**Script\_graficos.py**) ,la cual se utilizaran en este paso a seguir.

---

- Ejecutamos el mismo archivo **.py** (**Script\_graficos.py**) con el mismo contenido, desde la consola de comando como usuario normal.

```
$ python $HOME/Script_graficos.py
```

---

**Nota:** Al ejecutar el archivo **.py** (**Script\_graficos.py**) nos mostrara un mensaje donde salen reflejadas las 2 **Condiciones principales** y la 3 tarea (**Registro**),(**Pedido**),(**Disponible**):

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db'
is still in use, all queries will cease to work.
.....wheretokens: on
.....newnode: |Disponible| # Nueva tarea (Primera opción) Disponible
```

```

.....newnode: |Pedido| # Tarea Pedido
.....newnode: |Registrado| # Tarea Registrado
.....newnode: |inicial| # Condición inicial
.....newnode: |final| # Condición final
qt_temp.XM6088.svg # Obtenemos la nueva imagen con su nombre la cual

```

**El nombre de la imagen (.svg) es temporal, es decir su nombre varia constantemente.**

---

**Nota:** El nombre de la imagen (.svg) es temporal, es decir su nombre varia constantemente.

---

- Nos vamos al directorio **\$HOME/tmp**, desde la consola de comando.

```
$ cd $HOME/tmp
```

- Escribimos el comando **ls** para ver las 4 imágenes **.svg** que hemos obtenido ,desde la consola de comando.

```
tmp$ ls
qt_temp.XM6827.svg, qt_temp.XM6970.svg, qt_temp.XM5792.svg, qt_temp.XM6088.svg
```

- Con el comando **eog** vemos la tercera imagen **.svg(qt\_temp.XM6088.svg)** ,desde la consola de comando.

```
tmp$ eog qt_temp.XM6088.svg
```

---

**Nota:** Si realizó los pasos correctos, se mostrara el grafo como en la imagen siguiente *Figura 31: Disponible*

Descripción de la *Figura 31: Disponible*:

- La condición del circulo **inicial** apunta a la primera tarea de cuadro **Registrado**.
  - La primera tarea **Registrado** apunta a la segunda tarea **Pedido**.
  - La segunda tarea apunta a 2 tareas la primera opción de tarea es la del cuadro **Disponible**
  - La primera opción de tarea apunta a la condición de circulo **final**
  - En esta imagen vemos como se conectan la condición **inicial** con la primera tarea **Registrado**, la tarea primera tarea con la segunda tarea **Pedido**, la segunda tarea apunta a dos opciones de tareas la cual la primera opción es la tarea **Disponible** y la primera opción de tarea apunta a la condición **final**.
- 

## 22.5.4 5.4 - CUARTA TAREA (Por\_llegar)

En la cuarta tarea (**Por\_llegar**) se utilizaran las siguientes **etiquetas (XML)**:

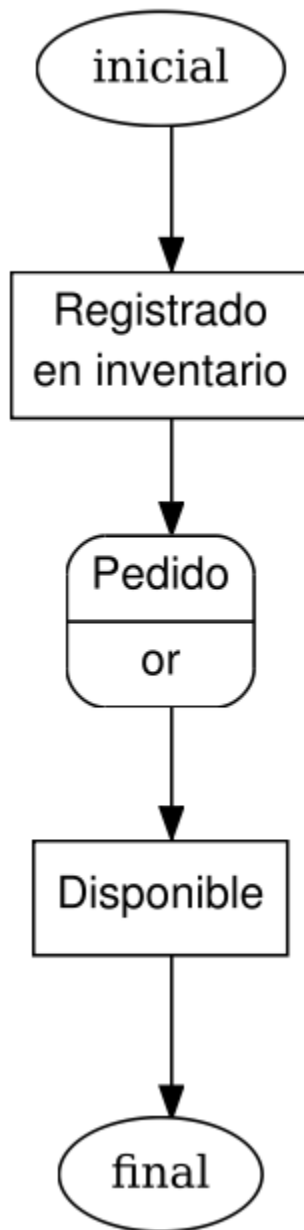


Figura 22.4: **Figura 31: Disponible**

- **<connection>**: **Por\_llegar** apunta a **final**, esto varia.
- **<task>**: Nombre de la tarea (**Por\_llegar**).
- **<port>**: **Por\_llegar** apunta a una opción.
- **<variable>**: Variable **vPor\_llegar** donde nos aparecerá (**nombre,id,status,fecha y hora**) de esa acción.

Abrimos el archivo **productos.xml** y insertamos el código (xml) de la **tarea Por\_llegar** que aparece en el siguiente block:

---

**Nota:** En la tarea **Pedido** se agrega otra etiqueta **<connection>** que sería segunda opción de tarea.

---

```
<!--

| Condición inicial |

-->
<condition type="start" id="inicial">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Registrado" source="Regi
 </port>
</condition>

<!--

| Registrado |

-->
<task title="en inventario" id="Registrado">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Pedido" source="Pedido"/
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vRegistrado" rolfield="(select rol from productos_registro
 where productoid=productos.id and regstatus='Registrado') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Registrado') as fecha"/>
</task>

<!--

| Pedido |

-->
<task title="" id="Pedido" textualinfo="">
 <port pattern="or" side="forward" type="split">
 <connection query="select status from productos" options="Disponible" source="Disp
 <connection query="select status from productos" options="Por_llegar" source="Por_
 </port>
```

```
<variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPedido" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Pedido') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Pedido') as fecha"/>
</task>

<!--

| Disponible |

-->
<task title="" id="Disponible" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vDisponible" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Disponible') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Disponible') as fecha"/>
 </task>

<!--

| Por_llegar |

-->
<task title="" id="Por_llegar" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPor_llegar" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Por_llegar') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Por_llegar') as fecha"/>
 </task>

<!--

| Condición final |

-->
<condition id="final">
 <port side="forward" type="split">
 <connection source=""/>
 </port>
</condition>
```

Ahora ejecutaremos los siguiente pasos:

---

**Nota:** En el 4° CUARTO PASO creamos la carpeta llamada **tmp** y el archivo llamado **.py (Script\_graficos.py)** ,la cual se utilizaran en este paso a seguir.

---

- Ejecutamos el mismo archivo **.py (Script\_graficos.py)** con el mismo contenido, desde la consola de comando como usuario normal.

```
$ python $HOME/Script_graficos.py
```

---

**Nota:** Al ejecutar el archivo **.py (Script\_graficos.py)** nos mostrara un mensaje donde salen reflejadas las 2 **Condiciones principales** y la 4 tarea (**Registro**),(**Pedido**),(**Disponible [OR]** **Por\_llegar**):

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db'
is still in use, all queries will cease to work.
.....wheretokens: on
.....newnode: |Disponible| # Tarea (Primera opción) Disponible
.....newnode: |Pedido| # Tarea Pedido
.....newnode: |Por_llegar| # Nueva tarea (Segunda opción) Por_llegar
.....newnode: |Registrado| # Tarea Registrado
.....newnode: |inicial| # Condición inicial
.....newnode: |final| # Condición final
qt_temp.XM6368.svg # Obtenemos la nueva imagen con su nombre la cual
```

**El nombre de la imagen (.svg) es temporal, es decir su nombre varia constantemente.**

---

**Nota:** El nombre de la imagen (.svg) es temporal, es decir su nombre varia constantemente.

---

- Nos vamos al directorio **\$HOME/tmp**,desde la consola de comando.

```
$ cd $HOME/tmp
```

- Escribimos el comando **ls** para ver las 5 imágenes **.svg** que hemos obtenido ,desde la consola de comando.

```
tmp$ ls
qt_temp.XM6827.svg, qt_temp.XM6970.svg, qt_temp.XM5792.svg, qt_temp.XM6088.svg, qt
```

- Con el comando **eog** vemos la tercera imagen **.svg(qt\_temp.XM6368.svg)** ,desde la consola de comando.

```
tmp$ eog qt_temp.XM6368.svg
```

---

**Nota:** Si realizó los pasos correctos, se mostrara el grafo como en la imagen siguiente *Figura 32: Por\_llegar*

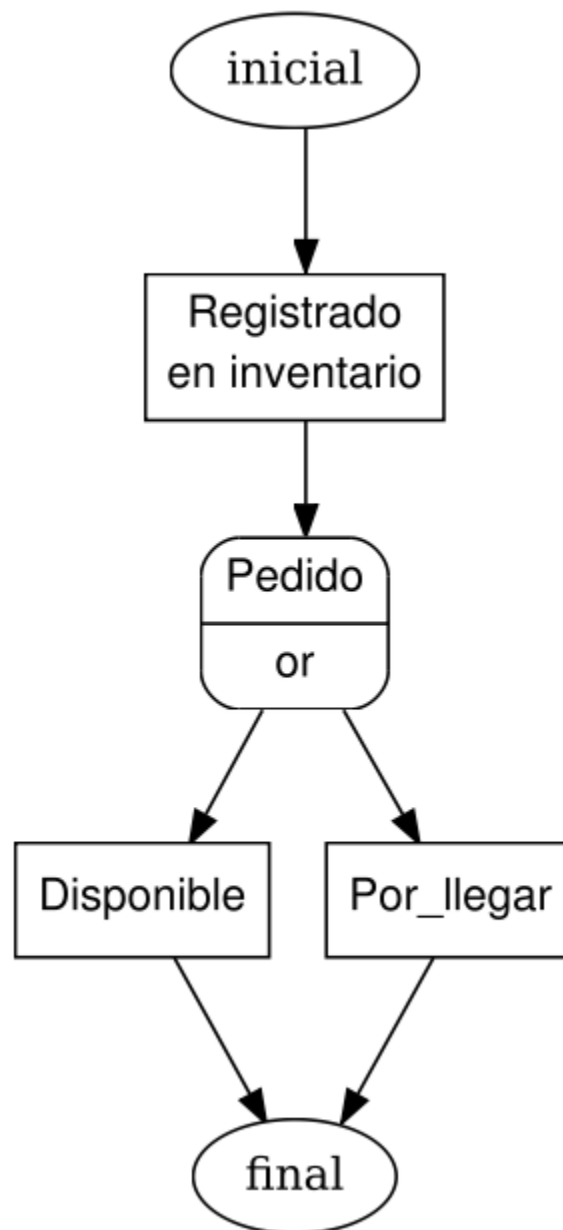


Figura 22.5: **Figura 32: Por\_llegar**



Descripción de la *Figura 32: Por\_llegar*:

- La condición del círculo **inicial** apunta a la primera tarea de cuadro **Registrado**.
- La primera tarea **Registrado** apunta a la segunda tarea **Pedido**.
- La segunda tarea apunta a 2 tareas, la primera opción de tarea es la del cuadro **Disponible** y la segunda opción de tarea es la del cuadro **Por\_llegar**.
- Las dos opción de tareas apunta a la condición de círculo **final**
- En esta imagen vemos como se conectan la condición **inicial** con la primera tarea **Registrado**, la tarea primera tarea con la segunda tarea **Pedido**, la segunda tarea apunta a dos opciones de tareas la cual la primera opción es la tarea **Disponible** y la segunda opción de tarea **Por\_llegar**, las dos opciones de tarea apuntan a la condición **final**.

### 22.5.5 5.5 - QUINTA TAREA (Por\_agotarse)

En la quinta tarea (**Por\_agotarse**) se utilizaran las siguientes **etiquetas (XML)**:

- **<connection>**: **Por\_agotarse** apunta a **final**, esto varia.
- **<task>**: Nombre de la tarea (**Por\_agotarse**).
- **<port>**: **Por\_agotarse** apunta a una opción.
- **<variable>**: Variable **vPor\_agotarse** donde nos aparecerá (**nombre,id,status,fecha y hora**) de esa acción.

Abrimos el archivo **productos.xml** y insertamos el código (**xml**) de la **tarea Por\_agotarse** que aparece en el siguiente block:

**Nota:** En las opciones de tareas **Disponible** y **Por\_llegar** se modificará la etiqueta **<connection>** que apuntaran a la siguiente tarea **Por\_agotarse**.

```
<!--

| Condición inicial |

-->
<condition type="start" id="inicial">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Registrado" source="Regi
 </port>
</condition>

<!--

| Registrado |

-->
```

```

-->
<task title="en inventario" id="Registrado">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Pedido" source="Pedido"/>
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="text"
 id="vRegistrado" rolfield="(select rol from productos_registro
 where productoid=productos.id and regstatus='Registrado') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Registrado') as fecha"/>
</task>

<!--

| Pedido |

-->
<task title="" id="Pedido" textualinfo="">
 <port pattern="or" side="forward" type="split">
 <connection query="select status from productos" options="Disponible" source="Disponible">
 <connection query="select status from productos" options="Por_llegar" source="Por_llegar">
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="text"
 id="vPedido" rolfield="(select rol from productos_registro
 where productoid=productos.id and regstatus='Pedido') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Pedido') as fecha"/>
</task>

<!--

| Disponible |

-->
<task title="" id="Disponible" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="select status from productos" options="Por_agotarse" source="Por_agotarse">
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="text"
 id="vDisponible" rolfield="(select rol from productos_registro
 where productoid=productos.id and regstatus='Disponible') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Disponible') as fecha"/>
</task>

<!--

| Por_llegar |

-->
```

```

<task title="" id="Por_llegar" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="select status from productos" options="Por_agotarse" source="Po
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPor_llegar" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Por_llegar') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Por_llegar') as fecha"/>
</task>

<!--

| Por_agotarse |

-->
<task title="" id="Por_agotarse" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPor_agotarse" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Por_agotarse') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Por_agotarse') as fecha"/>
</task>

<!--

| Condición final |

-->
<condition id="final">
 <port side="forward" type="split">
 <connection source=""/>
 </port>
</condition>

```

Ahora ejecutaremos los siguiente pasos:

**Nota:** En el 4° CUARTO PASO creamos la carpeta llamada **tmp** y el archivo llamado **.py** (**Script\_graficos.py**) ,la cual se utilizaran en este paso a seguir.

- Ejecutamos el mismo archivo **.py** (**Script\_graficos.py**) con el mismo contenido, desde la consola de comando como usuario normal.

```
$ python $HOME/Script_graficos.py
```

**Nota:** Al ejecutar el archivo **.py** (**Script\_graficos.py**) nos mostrara un mensa-  
je donde salen reflejadas las 2 **Condiciones principales** y la 4 tarea (**Registra-**

do),(Pedido),(Disponible [OR] Por\_llegar):

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db'
is still in use, all queries will cease to work.
.....wheretokens: on
.....newnode: |Disponible| # Tarea (Primera opción) Disponible
.....newnode: |Pedido| # Tarea Pedido
.....newnode: |Por_agotarse| # Nueva tarea Por_agotarse
.....newnode: |Por_llegar| # Tarea (Segunda opción) Por_llegar
.....newnode: |Registrado| # Tarea Registrado
.....newnode: |inicial| # Condición inicial
.....newnode: |final| # Condición final
qt_temp.XM6667.svg # Obtenemos la nueva imagen con su nombre la cu
```

**El nombre de la imagen (.svg) es temporal, es decir su nombre varia constante mente.**

---

**Nota:** El nombre de la imagen (.svg) es temporal, es decir su nombre varia constante mente.

---

- Nos vamos al directorio **\$HOME/tmp**, desde la consola de comando.

```
$ cd $HOME/tmp
```

- Escribimos el comando **ls** para ver las 5 imágenes **.svg** que hemos obtenido ,desde la consola de comando.

```
tmp$ ls
qt_temp.XM6827.svg, qt_temp.XM6970.svg, qt_temp.XM5792.svg, qt_temp.XM6088.svg,
qt_temp.XM6368.svg, qt_temp.XM6667.svg
```

- Con el comando **eog** vemos la tercera imagen **.svg(qt\_temp.XM6667.svg)** ,desde la consola de comando.

```
tmp$ eog qt_temp.XM6667.svg
```

**Nota:** Si realizó los pasos correctos, se mostrara el grafo como en la imagen siguiente *Figura 33: Por\_agotarse*

Descripción de la *Figura 33: Por\_agotarse*:

- La condición del circulo **inicial** apunta a la primera tarea de cuadro **Registrado**.
- La primera tarea **Registrado** apunta a la segunda tarea **Pedido**.
- La segunda tarea apunta a 2 tareas, la primera opción de tarea es la del cuadro **Disponible** y la segunda opción de tarea es la del cuadro **Por\_llegar**.
- Las dos opción de tareas apunta a la quinta tarea del cuadro **Por\_agotarse**.

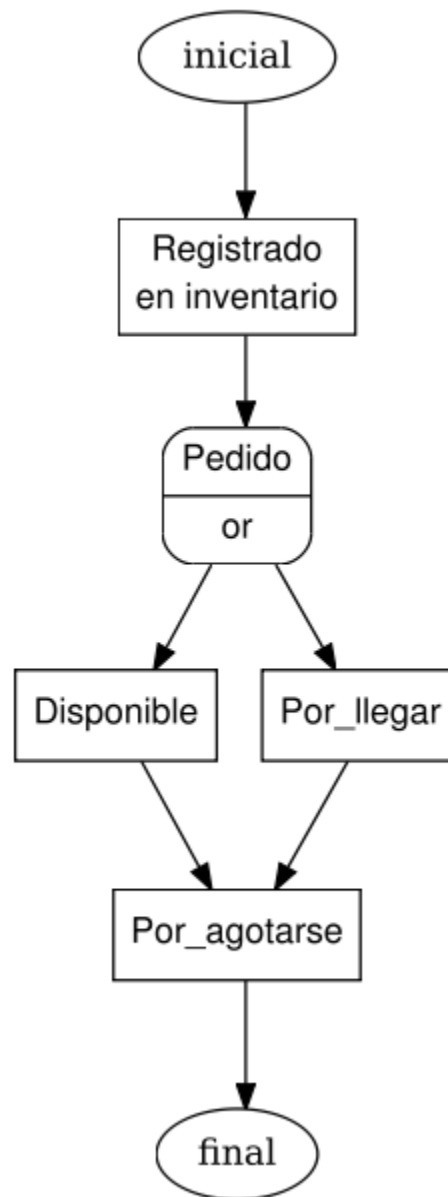


Figura 22.6: **Figura 33: Por\_agotarse**

- La quinta tarea apunta a la condición de círculo **final**
  - En esta imagen vemos como se conectan la condición **inicial** con la primera tarea **Registrado**, la tarea primera tarea con la segunda tarea **Pedido**, la segunda tarea apunta a dos opciones de tareas la cual la primera opción es la tarea **Disponible** y la segunda opción de tarea **Por\_llegar**, las dos opciones de tarea apuntan a la quinta tarea **Por\_agotarse** y apunta a la condición **final**.
- 

### 22.5.6 5.6 - SEXTA TAREA (Agotado)

En la sexta tarea (**Agotado**) se utilizaran las siguientes **etiquetas (XML)**:

- **<connection>**: **Agotado** apunta a (**final**, **Pedido**),esto varia.
- **<task>**: Nombre de la tarea (**Agotado**).
- **<port>**: **Agotado** apunta a dos opción.
- **<variable>**: Variable **vAgotado** donde nos aparecerá (**nombre,id,status,fecha y hora**) de esa acción.

Abrimos el archivo **productos.xml** y insertamos el código (**xml**) de la tarea **Agotado** que aparece en el siguiente block:

---

**Nota:** En la tarea **Por\_agotarse** se modificará la etiqueta **<connection>** que apuntaran a la siguiente tarea **Agotado**.

---

```
<!--

| Condición inicial |

-->
<condition type="start" id="inicial">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Registrado" source="Regi
 </port>
</condition>

<!--

| Registrado |

-->
<task title="en inventario" id="Registrado">
 <port side="forward" type="split">
 <connection query="select status from productos" options="Pedido" source="Pedido"/
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vRegistrado" rolfield="(select rol from productos_registro
 where productoid=productos.id and regstatus='Registrado') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
```

```

 where productoid=productos.id and regstatus='Registrado') as fecha"/>
</task>

<!--

| Pedido |

-->
<task title="" id="Pedido" textualinfo="">
 <port pattern="or" side="forward" type="split">
 <connection query="select status from productos" options="Disponible" source="Disp
 <connection query="select status from productos" options="Por_llegar" source="Por_
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPedido" rolfield="(select rol from
productos_registro
 where productoid=productos.id and regstatus='Pedido') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Pedido') as fecha"/>
</task>

<!--

| Disponible |

-->
<task title="" id="Disponible" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="select status from productos" options="Por_agotarse" source="Po
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vDisponible" rolfield="(select rol from
productos_registro
 where productoid=productos.id and regstatus='Disponible') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Disponible') as fecha"/>
</task>

<!--

| Por_llegar |

-->
<task title="" id="Por_llegar" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="select status from productos" options="Por_agotarse" source="Po
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="
 id="vPor_llegar" rolfield="(select rol from
productos_registro
 where productoid=productos.id and regstatus='Por_llegar') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Por_llegar') as fecha"/>
</task>

```

```
<!--

| Por_agotarse |

-->
<task title="" id="Por_agotarse" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="select status from productos" options="Agotado" source="Agotado" />
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="text"
 id="vPor_agotarse" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Por_agotarse') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Por_agotarse') as fecha"/>
</task>
```

```
<!--

| Agotado |

-->
<task title="" id="Agotado" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 <connection query="select status from productos" options="Pedido" back="yes" source="Agotado" />
 </port>
 <variable config="1" documentsource="select id,nombre,status from productos" type="text"
 id="vAgotado" rolfield="(select rol from#xa;productos_registro
 where productoid=productos.id and regstatus='Agotado') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Agotado') as fecha"/>
</task>
```

```
<!--

| Condición final |

-->
<condition id="final">
 <port side="forward" type="split">
 <connection source=""/>
 </port>
</condition>
```

Ahora ejecutaremos los siguiente pasos:

---

**Nota:** En el 4° CUARTO PASO creamos la carpeta llamada **tmp** y el archivo llamado **.py** (Script\_graficos.py) ,la cual se utilizaran en este paso a seguir.



- Ejecutamos el mismo archivo **.py (Script\_graficos.py)** con el mismo contenido, desde la consola de comando como usuario normal.

```
$ python $HOME/Script_graficos.py
```

**Nota:** Al ejecutar el archivo **.py (Script\_graficos.py)** nos mostrara un mensaje donde salen reflejadas las 2 **Condiciones principales** y la 4 tarea (**Registrado**),(**Pedido**),(**Disponible [OR] Por\_llegar**):

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db'
is still in use, all queries will cease to work.
.....wheretokens: on
.....newnode: |Agotado| # Nueva tarea Agotado
.....newnode: |Disponible| # Tarea (Primera opción) Disponible
.....newnode: |Pedido| # Tarea Pedido
.....newnode: |Por_agotarse| # Tarea Por_agotarse
.....newnode: |Por_llegar| # Tarea (Segunda opción) Por_llegar
.....newnode: |Registrado| # Tarea Registrado
.....newnode: |inicial| # Condición inicial
.....newnode: |final| # Condición final
qt_temp.XM6954.svg # Obtenemos la nueva imagen con su nombre la cu
```

**El nombre de la imagen (.svg) es temporal, es decir su nombre varia constantemente.**

- Nos vamos al directorio **\$HOME/tmp**, desde la consola de comando.

```
$ cd $HOME/tmp
```

- Escribimos el comando **ls** para ver las 6 imágenes **.svg** que hemos obtenido ,desde la consola de comando.

```
tmp$ ls
qt_temp.XM6827.svg, qt_temp.XM6970.svg, qt_temp.XM5792.svg, qt_temp.XM6088.svg,
qt_temp.XM6368.svg, qt_temp.XM6667.svg, qt_temp.XM6954.svg
```

- Con el comando **eog** vemos la tercera imagen **.svg(qt\_temp.XM6954.svg)** ,desde la consola de comando.

```
tmp$ eog qt_temp.XM6954.svg
```

**Nota:** Si realizó los pasos correctos, se mostrara el grafo como en la imagen siguiente *Figura 34: Agotado*

Descripción de la *Figura 34: Agotado*:

- La condición del circulo **inicial** apunta a la primera tarea de cuadro **Registrado**.
- La primera tarea **Registrado** apunta a la segunda tarea **Pedido**.

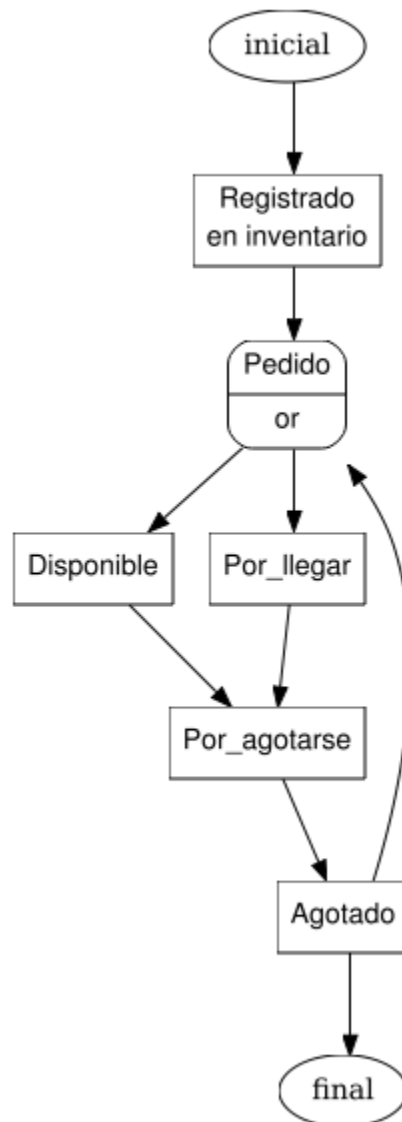


Figura 22.7: **Figura 34: Agotado**

- La segunda tarea apunta a 2 tareas, la primera opción de tarea es la del cuadro **Disponible** y la segunda opción de tarea es la del cuadro **Por\_llegar**.
- Las dos opción de tareas apunta a la quinta tarea del cuadro **Por\_agotarse**.
- La quinta tarea apunta a la sexta tarea del cuadro **Agotado**.
- La sexta tarea apunta a la condición de circulo **final**.
- En esta imagen vemos como se conectan la condición **inicial** con la primera tarea **Registrado**, la tarea primera tarea con la segunda tarea **Pedido**, la segunda tarea apunta a dos opciones de tareas la cual la primera opción es la tarea **Disponible** y la segunda opción de tarea **Por\_llegar**, las dos opciones de tarea apuntan a la quinta tarea **Por\_agotarse**, la quinta tarea apunta a la sexta tarea **Agotado** y la sexta tarea apunta a la condición **final**.

---

**Nota:** Si por algún motivo les causo errores en los pasos aquí están el archivo para que los **descarguen** como se muestra en la siguiente imagen.

---

**DOWNLOAD:**



PRODUCTOS.XML



---

## Ver datos usando flujos de trabajo

---

En esta sección de explicar las acciones necesarias para obtener reportes asociados a la configuración de datos del directorio **.safet/ (PySafet)**. Tomando como ejemplo el caso inventario explicaremos las acciones a realizar para obtener un listado de todos los datos.

A continuaciones mostraremos las siguientes acciones:

### 23.1 1° PRIMERA ACCION (Tarea “vRegistrado”)

Listaremos los productos que se encuentran registrados (**en estado registrado**), con la tarea **vRegistrado** la cual conocimos en los pasos anteriores:

**Ver también:**

```
<!--

| Registrado |

-->
<task title="en inventario" id="Registrado">
<port side="forward" type="split">
<connection query="true" options="" source="final"/>
</port>
 <!-- nombre de las columnas(id,nombre,status), variable(id="vRegistrado")-->
<variable config="1" documentsource="select id,nombre,status from productos"
type="sql" tokenlink="" id="vRegistrado" rolfield="(select rol from productos_registro
where productoid=productos.id and regstatus='Registrado') as rol" scope="task"
timestampfield="(select fecha from productos_registro
where productoid=productos.id and regstatus='Registrado') as fecha"/>
</task>
```

#### 23.1.1 1° PRIMER PASO

- Crearemos un directorio en nuestro **<HOME>/JsonInterfaz**

```
$ mkdir $HOME/JsonInterfaz
```

- Crearemos un directorio que creamos **JsonInterfaz/**

```
$ cd $HOME/JsonInterfaz
```

### 23.1.2 2° SEGUNDO PASO

- Crearemos un archivo **.py** por ejemplo **ListarProducto.py** en el directorio **JsonInterfaz/**:

```
JsonInterfaz$ touch ListarProducto.py
```

- Abrimos el archivo **ListarProducto.py**, insertamos el siguiente código:

```
-*- coding: utf-8 -*-

import Safet
import os

myhome = os.getenv("HOME")
mymedia = myhome + "/tmp"
myurl = "http://localhost"

myinflow = Safet.MainWindow(myhome)

myinflow.setMediaPath(mymedia)
myinflow.setHostURL(myurl)

myconsult = u"operacion:Listar_datos_general \
Cargar_archivo_flujo: /home/cenditel/.safet/flowfiles/productos.xml \
Variable: vRegistrado"

result = myinflow.login("admin", "admin")

if not result:
 print "Authentication failed"
 exit()

result = myinflow.toInputConsole(myconsult)

if not result:
 print "Consult failed error: %s" % (myinflow.currentError())
 exit()

print u"%s" % (myinflow.currentJSON())
```

---

**Nota:** La operación seleccionada es “**Listar\_datos**”, el archivo donde se realiza la operación es “**productos.xml**” y el directorio donde se encuentra es **<HOMR>.safet/flowfiles/**.

---

La variable seleccionada “**vRegistrado**” se define los siguiente:

- La variable debe estar definida en el archivo “**productos.xml**”.

- La operación **Listar\_datos** esta definida en el archivo **defconsole.xml** que se encuentra en el directorio **<HOME>.safet/input/**

### 23.1.3 3° TERCER PASO

- Ejecutamos el archivo **ListarProducto.py** desde la consola de comando como usuario normal:

```
JsonInterfaz$ python ListarProducto.py
```

**Nota:** Al llamar el método **toInputConsole** con las opciones dicha anteriormente se generaría el siguiente mensaje **Json**.

```
QFSFileEngine::open: No file name specified

Aquí nos indica que se esta conectando a la base de datos
que esta en el directorio <HOME>.safet/
QSqlDatabasePrivate::removeDatabase:
connection '/home/cenditel/.safet/mydb.db' is still in use,
all queries will cease to work.
QSqlDatabasePrivate::removeDatabase:
connection '/home/cenditel/.safet/mydb.db' is still in use,
all queries will cease to work.
QSqlDatabasePrivate::removeDatabase:
connection '/home/cenditel/.safet/mydb.db' is still in use,
all queries will cease to work.

La variable que esta utilizando
{ "safetvariable" : "vRegistrado",

"safetkey" : "",

"safetttitle" : "vRegistrado",
"safetreport" : "operacion:Listar_datos_general \
Cargar_archivo_flujo:/home/cenditel/.safet/flowfiles/productos.xml \
Variable: vRegistrado",

Aquí nos dice que hay dos registro en la db
"safetlistcount" : "2",

Nos mostrara los datos en una lista
"safetlist" :
[
Aqui nos aparece el nombre de las columnas(id,nombre,status) y
sus datos
{"id":"1", "id":"1", "nombre":"Jabón Protex", "status":"Registrado"},
{"id":"2", "id":"2", "nombre":"Pañales Pampers", "status":"Registrado"}
],

#Nos mostrara la columnas es decir el keys
"safetcolumns" :
[
```

```
{ "key": "id", "label": "id", "width": "10",
 "resizeable": "true", "sortable": "true"},

{ "key": "id", "label": "id", "width": "10",
 "resizeable": "true", "sortable": "true"},

{ "key": "nombre", "label": "nombre", "width": "90",
 "resizeable": "true", "sortable": "true"},

{ "key": "status", "label": "status", "width": "90",
 "resizeable": "true", "sortable": "true"}
]
}
```

---

## 23.2 Combinación de Json con HTML(javascript)

Listaremos los datos con interfaz gráfica de nuestro **Json**, con los siguientes pasos:

### 23.2.1 1° PRIMER PASO

- Crearemos un archivo **.py** por ejemplo (**ScriptJson.py**), en el directorio **JsonInterfaz/**:

```
JsonInterfaz$ touch ScriptJson.py
```

### 23.2.2 2° SEGUNDO PASO

- Abrimos el archivo que creamos **.py**(**ScriptJson.py**) insertamos el siguiente código:

```
#!/user/bin python
-*- coding: utf-8 -*-

importamos la librerías a utilizar
import Safet
import os
import json

#función para convertir el json en datos con listas
def jsonToJquery(myoldarray):

 # Variables que me van a almacenar listar o arreglos
 mynewarray = []
 currcolumns = []
 currkeys = []

 # Una variable para almacenar
 stringcolumn = ""

 # Estructura de repetición para obtener
```



```
Las keys es decir los dato de
las columnas (id,nombre,status)
for reg in myoldarray:
 # reg.keys() me obtiene una en una la columnas
 currkeys = reg.keys()
 myvalue = ""
 regarray = []

 # Estructura de repetición anidada
 # Se obtienen los datos de la columnas
 # es decir del (id,nombre,status)
 for currkey in currkeys:
 # optienes el datos
 myvalue = reg[currkey]
 # Insertamos el dato dentro de la lista regarray[]
 regarray.insert(0,myvalue)

 # Se añade la lista regarray[] es decir datos de las tablas
 mynewarray.append(regarray)

#Estructura de repetición para obtener las columnas para javascript
for currkey in currkeys:
 # Optenemos las columnas que se almacenan en un diccionario
 currcolumn = { "sTitle" : currkey }
 # Se inserta en la lista currcolumn[] el diccionario
 currcolumns.insert(0,currcolumn)

Retornamos la función con 2 listas
mynewarray[] que son los datos de la tablas o columnas
currcolumns[] que es un diccionario que contiene las columnas de
la tablas de los datos
return (mynewarray,currcolumns)

función para escribir los datos obtenido en un archivo .js(javascript)
def writeJsonData(data,columns,Variable,filename):

 # u"%s" para almacenan los datos y las columnas
 stringnewarray = u"%s" % data
 stringcolumns = u"%s" % columns

 # se abre el archivo de tipo escritura
 file = open(filename, "w")

 # Escribimos el nombre de lista para datos y para las columnas
 datatowrite = u"dataInventory \
 = %s\n\ncolumnInventory = %s" % (stringnewarray, stringcolumns)

 # Se obtiene el valor de la variable
 variablewrite = "Variable = %s" % (Variable)

 # datatowrite.replace para que en el archivo
```

```
.js se me eliminar "u'"por "'"
datatowrite = datatowrite.replace("u'", "'")
variablewrite = variablewrite.replace("u'", "'")

salto de linea
datatowrite = datatowrite + "\n"

Escribimos en archivo las dos listas
file.write(datatowrite)
file.write(variablewrite)

Cerramos el archivo
file.close()

Mi función principal
def Principal():

 # Se localiza en el directorio <HOME>
 myhome = os.getenv("HOME")
 mymedia = myhome + "/tmp"
 myurl = "http://localhost"

 # Mi constructor MainWindow
 myinflow = Safet.MainWindow(myhome)

 myinflow.setMediaPath(mymedia)
 myinflow.setHostURL(myurl)

 # Mi consulta de vRegistrado y su operación listar datos
 #que se encuentra en el directorio <HOME>.safe/flowfiles/
 myconsult = u"operacion:Listar_datos \
 Cargar_archivo_flujo: %s/.safet/flowfiles/productos.xml \
 Variable: vRegistrado " % (myhome)

 # Nombre de usuario y si password
 result = myinflow.login("admin", "admin")

 if not result:
 print "Authentication failed"
 exit()

 # Obtiene los el json de archivo toInputConsole.xml
 result = myinflow.toInputConsole(myconsult)

 if not result:
 print "Consult failed error: %s" % (myinflow.currentError())
 exit()

 # Se obtiene mis datos json
 mystringdata = u"%s" % myinflow.currentJSON();

 # Me carga los datos json
 mydata = json.loads(mystringdata)
```

```
Llámanos a la función con el nombre jsonToJquery()
para modificar el json
Se le pasa 1 parámetro
primero mydata con el nombre de la lista

Se almacenan los datos en 2 variable
La cual la función esta retornando 2 resultados
el array y las columnas del json
(mynewarray, currcolumns) = jsonToJquery(mydata["safetlist"])

#Llámanos a la función que la llámanos writeJsonData()
para insertar los datos en un archivo
#Se le pasan 3 parámetro:
#primero mynewarray
#Segundo currcolumns
#Tercero el nombre del archivo .js
writeJsonData(mynewarray, currcolumns, [mydata["safetvariable"]], "dataInventory.js")

if __name__ == "__main__":

 # Mi función principal
 Principal()
```

---

**Nota:** Este Script se utiliza para poder ver los datos con la libreria jquery de manara más dinamica.

---

### 23.2.3 3° TERCER PASO

- Ejecutamos el archivo .py(**Script.Json.py**) desde la consola de comando como usuario normal:

```
JsonInterfaz$ python ScriptJson.py
```

---

**Nota:** Nos mostrara el siguiente mensaje, significando que funciono correctamente el Script

```
QFSFileEngine::open: No file name specified
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db' is still in use, all queries will cease to work.
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db' is still in use, all queries will cease to work.
QSqlDatabasePrivate::removeDatabase: connection '/home/cenditel/.safet/mydb.db' is still in use, all queries will cease to work.
```

---

**Nota:** AL ejecutar Script (**Script.Json.py**) se genera un archivo **javascript** llamado **dataInventory.js** que contiene las columnas de la tabla y sus datos

---

### 23.2.4 4° CUARTO PASO

- Crearemos un archivo **.html** por ejemplo (**InventarioJson.html**), en el directorio **JsonInterfaz/**:

```
JsonInterfaz$ touch InventarioJson.html
```

- Abrimos el archivo que creamos **.html(InventarioJson.html)**, insertamos el siguiente código:

```
<html>
<head>

 <link rel="stylesheet" type="text/css"
href="http://ajax.aspnetcdn.com/ajax/
 jquery.dataTables/1.9.4/css/jquery.dataTables.css">
</head>

<body>

<!-- Aqui se crea la tabla dentro de la etiqueta <body>-->
<table cellpadding="0" cellspacing="0" border="0"
 class="display" name="safettable0" id="safettable0">
</table>

<script src="dataInventory.js"> </script>

<script type="text/javascript" charset="utf8"
 src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.2.min.js">
</script>

<script type="text/javascript" charset="utf8"
 src="http://ajax.aspnetcdn.com/ajax/
 jquery.dataTables/1.9.4/jquery.dataTables.min.js">
</script>

<script>

 $(function() {
 $("#safettable0").dataTable({
 "aaData": dataInventory,
 "aoColumns": columnInventory
 });
 })
</script>

</body>
</html>
```

---

**Nota:** En este código HTML se utilizó la librería (jquery) y la librería (jquery.dataTables) de manera interna. Para ver los datos mas dinámicos utilizando (javascript y json).

---

### 23.2.5 5° QUINTO PASO

- Abrimos con cualquier navegador web el archivo Json.html y no mostrara de la siguiente forma:  
*Figura 28: Json*

**Nota:**

---

Show  entries

Search:

id	nombre	status
1	Jabón Protex	Registrado
2	Pañales Pamper	Registrado
3	Vitamina C	Registrado
4	Vitamina D	Registrado
5	Vitamina B12	Registrado
6	Vitamina K	Registrado
7	Vitamina B1	Registrado
8	Vitamina B8	Registrado

Showing 1 to 8 of 8 entries

◀ Previous Next ▶

Figura 23.1: Figura 28: Json

La siguiente *Figura 28: Json* indica lo siguiente:

- Nos mostrara el nombre de las columnas.
  - Nos mostrara los datos de cada columna.
  - Nos mostrara un buscador **Search**.
  - Nos mostrara **Show** que nos indicara cuantos datos quiere que se muestren en la tabla.
- 
- Aquí esta el ejemplo, damos **Clik** a JSON.HTML

### 23.2.6 6° SEXTO PASO

En el script de **Combinación de Json con HTML(javascript)** del *2° SEGUNDO PASO* en la función principal tenemos la consulta, la cual nos sirve para consultar los datos de las más variables, es decir, de las variables (“vPedido”, “vDisponible”, “vPor\_llegar”, “vPor\_agotarse”, “vAgotado”), con solo cambiarle el nombre de la variable.

Por ejemplo habíamos colocado anteriormente la variable “vRegistrado”, la cambiamos por “vDisponible” o la que sea.

Ya cambiado la variable solo se ejecuta el **Script** como en el *Etiquetas Principales*, la cual al ejecutarlo nos generara el archivo **.js(ScriptJson.py)** con los datos y las columnas, como lo hacia con la variable “vRegistrado”, igualmente vemos en el navegador con el mismo código **HTML** del archivo (**InventarioJson.html**), que nos muestra el contenido con el mismo formato de la librerías (**jquery y jquery.dataTables**).

Ver también:

---

**Nota:** Si vamos a utilizar la variable “vPedido” tómese en cuenta lo siguiente:

---

1. La variable “vPedido” necesita saber la **cantidad** de pedidos, para poder mostrar los datos.
2. Abrimos el archivo **.xml(productos.xml)** de directorio **<HOME>.safet/flowfiles/**:

---

**Nota:** Vemos el código y tenemos los atributos (**id,nombre,status**):

```
<!--

| Pedido |

-->
<task title="" id="Pedido" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>

 <!-- atributos (id,nombre,status) -->
 <variable config="1" documentsource="select id,nombre,status from productos"
 type="sql" tokenlink=""
 id="vPedido" rolfield="(select rol from
productos_registro
 where productoid=productos.id and regstatus='Pedido') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Pedido') as fecha"/>
 </task>
```

---

3. Se le decimos que nos muestre el atributo **cantidad** de la siguiente manera:

**Ver también:**

```
<!--

| Pedido |

-->
<task title="" id="Pedido" textualinfo="">
 <port pattern="none" side="forward" type="split">
 <connection query="true" options="" source="final"/>
 </port>

 <!-- atributos (id,nombre,cantidad,status) -->
 <variable config="1" documentsource="select id,nombre,cantidad,status from pr
 type="sql" tokenlink=""
 id="vPedido" rolfield="(select rol from
productos_registro
 where productoid=productos.id and regstatus='Pedido') as rol" scope="task"
 timestampfield="(select fecha from productos_registro
 where productoid=productos.id and regstatus='Pedido') as fecha"/>
 </task>
```

4. Listo ya podemos utilizar la variable “vPedido” para consultar, por ejemplo cuantos productos se han pedido. Claro colocanod la variable **vPedido** y luego ejecutando el Script del 2° *SEGUNDO PASO*,

como en la *Figura 29: Json*

**Nota:**

Show  entries Search:

cantidad	id	nombre	status
25	8	Vitamina B8	Pedido
30	6	Vitamina K	Pedido
30	4	Vitamina D	Pedido
220	2	Pañales Pampers	Pedido
300	5	Vitamina B12	Pedido
600	7	Vitamina B1	Pedido
1000	1	Jabón Protex	Pedido
1200	3	Vitamina C	Pedido

Showing 1 to 8 of 8 entries ◀ Previous Next ▶

Figura 23.2: **Figura 29: Json**

La siguiente *Figura 29: Json* indica lo siguiente:

- Nos mostrara el nombre de las columnas, más el atributo **cantidad**.
- Nos mostrara los datos de cada columna.
- Nos mostrara un buscador **Search**.
- Nos mostrara **Show** que nos indicara cuantos datos quiere que se muestren en la tabla.

- Aquí esta el ejemplo, damos **Click** a JSON.(HTML)





## Ver fichas usando flujos de trabajo

En los pasos anteriores obtenemos los datos de dos **acciones** **vRegistrado** y **vPedido**, la cual utilizamos como ejemplo como se muestra en la siguiente imágenes:

### Nota: vRegistrado

Show  entries

Search:

id	nombre	status
1	Jabón Protex	Registrado
2	Pañales Pampers	Registrado
3	Vitamina C	Registrado
4	Vitamina D	Registrado
5	Vitamina B12	Registrado
6	Vitamina K	Registrado
7	Vitamina B1	Registrado
8	Vitamina B8	Registrado

Showing 1 to 8 of 8 entries

◀ Previous Next ▶

Figura 24.1: **Figura 28: vRegistrado**

La siguiente *Figura 28: vRegistrado* indica lo siguiente:

- Nos mostrara el nombre de las columnas.
- Nos mostrara los datos de cada columna.
- Nos mostrara un buscador **Search**.
- Nos mostrara **Show** que nos indicara cuantos datos quiere que se muestren en la tabla.

### Nota: vPedido

La siguiente *Figura 29: vPedido* indica lo siguiente:

- Nos mostrara el nombre de las columnas, más el atributo **cantidad**.

Show  entries

Search:

cantidad	id	nombre	status
25	8	Vitamina B8	Pedido
30	6	Vitamina K	Pedido
30	4	Vitamina D	Pedido
220	2	Pañales Pampers	Pedido
300	5	Vitamina B12	Pedido
600	7	Vitamina B1	Pedido
1000	1	Jabón Protex	Pedido
1200	3	Vitamina C	Pedido

Showing 1 to 8 of 8 entries

◀ Previous Next ▶

Figura 24.2: Figura 29: vPedido

- Nos mostrara los datos de cada columna.
- Nos mostrara un buscador **Search**.
- Nos mostrara **Show** que nos indicara cuantos datos quiere que se muestren en la tabla.

A continuación vamos a demostrar, como obtener fichas de esa tabla de datos:

## 24.1 Ficha (vRegistrado) (vPedido)

Los paso para obtener una ficha que vamos a seguir mas adelante, nos sirven para las demás acciones menos una, es decir, no solo para la acción **vRegistrado** sino tambien para las acciones (**vDisponible**, **vPor\_llegar**, **vPor\_agotarse**, **vAgotado**). Para la acción **vPedido** los pason son diferentes ya que ah esta acción se agrega el el atributo **cantidad**, porque para mostrar los datos depende del valor cantidad.

Las demas acciones tienen, el mismo esquemas de mostrar los datos en fichas como se muestra en la siguiente imagen *Figura 30: Ficha vRegistrado*:

**Nota:** La imagen nos muestra el nombre de la ficha la cual sirve para las acciones (**vDisponible**, **vPor\_llegar**, **vPor\_agotarse**, **vAgotado**), la cual tienen los mismos atributos que las demás acciones.

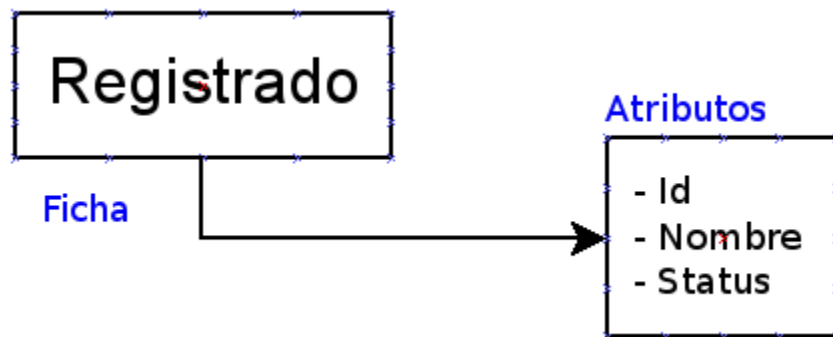


Figura 24.3: Figura 30: Ficha vRegistrado

La imagen muestra el nombre de la ficha y sus atributos: *Figura 32: Ficha vPedido*

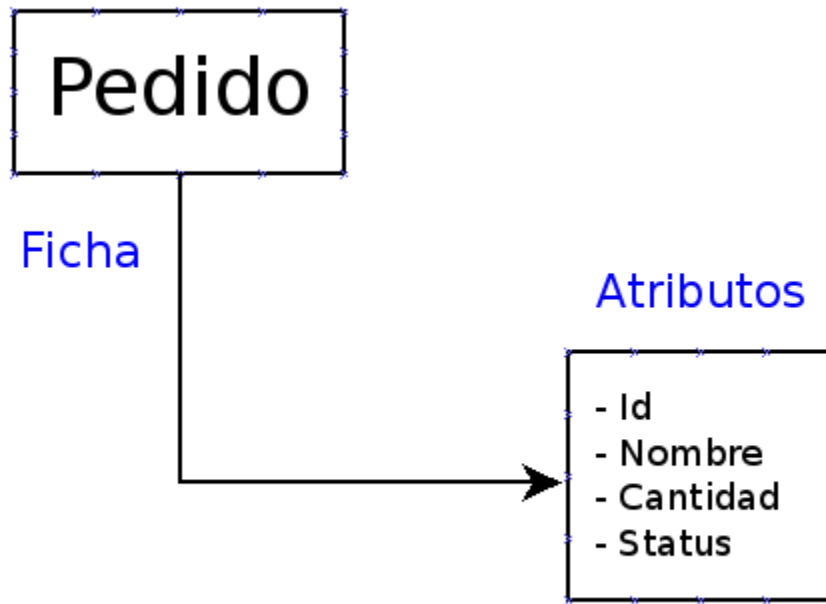


Figura 24.4: Figura 32: Ficha vPedido

### 24.1.1 1° PRIMER PASO

- Nos vamos directorio que hemos creado anteriormente que se llama **JsonInterfaz/**

```
$ cd $HOME/JsonInterfaz
```

### 24.1.2 2° SEGUNDO PASO

- Ejecutamos el Script que insertamos en el archivo que creamos anteriormente **.py(ScriptJson.py)**, con la variable (**vRegistrado**) o con la la variable **vPedido** o la que guste.

```
JsonInterfaz$ python ScriptJson.py
```

### 24.1.3 3° TERCER PASO

- Crearemos un archivo **.css** por ejemplo (**style.css**), en el directorio **JsonInterfaz/**:

```
JsonInterfaz$ touch style.css
```

- Abrimos el archivo que creamos **.css(style.css)**, insertamos el siguiente código:

```

body {
 background-repeat:no-repeat;
 background-position:top center;
 background-color:#657077;
 margin:40px;
}

#tabbed_box_1 {
 margin: 0px auto 0px auto;
 width:300px;
}

.tabbed_box h4 {
 font-family:Arial, Helvetica, sans-serif;
 font-size:23px;
 color:#ffffff;
 letter-spacing:-1px;
 margin-bottom:10px;
}

.tabbed_box h4 small {
 color:#e3e9ec;
 font-weight:normal;
 font-size:9px;
 font-family:Verdana, Arial, Helvetica, sans-serif;
 text-transform:uppercase;
 position:relative;
 top:-4px;
 left:6px;
 letter-spacing:0px;
}

.tabbed_area {
 border:1px solid #494e52;
 background-color:#636d76;
 padding:8px;
}

ul.tabs {
 margin:0px; padding:0px;
 margin-top:5px;
 margin-bottom:6px;
}

ul.tabs li {
 list-style:none;
 display:inline;
}

ul.tabs li a {
 background-color:#464c54;
 color:#ffe5b5;
 padding:8px 14px 8px 14px;
 text-decoration:none;
 font-size:9px;
 font-family:Verdana, Arial, Helvetica, sans-serif;
 font-weight:bold;
 text-transform:uppercase;
}

```

```

 border:1px solid #464c54;
 background-repeat:repeat-x;
 background-position:bottom;
 }
 ul.tabs li a:hover {
 background-color:#2f343a;
 border-color:#2f343a;
 }
 ul.tabs li a.active {
 background-color:#ffffff;
 color:#282e32;
 border:1px solid #464c54;
 border-bottom: 1px solid #ffffff;
 background-repeat:repeat-x;
 background-position:top;
 }
 .content {
 background-color:#ffffff;
 padding:10px;
 border:1px solid #464c54;
 font-family:Arial, Helvetica, sans-serif;
 background-repeat:repeat-x;
 background-position:bottom;
 }
 #content_2, #content_3 { display:none; }

 .content ul {
 margin:0px;
 padding:0px 20px 0px 20px;
 }
 .content ul li {
 list-style:none;
 border-bottom:1px solid #d6dde0;
 padding-top:15px;
 padding-bottom:15px;
 font-size:13px;
 }
 .content ul li:last-child {
 border-bottom:none;
 }
 .content ul li a {
 text-decoration:none;
 color:#3e4346;
 }
 .content ul li a small {
 color:#8b959c;
 font-size:9px;
 text-transform:uppercase;
 font-family:Verdana, Arial, Helvetica, sans-serif;
 position:relative;
 left:4px;
 top:0px;
 }

```

```
.content ul li a:hover {
 color:#a59c83;
}
.content ul li a:hover small {
 color:#baae8e;
}
```

### 24.1.4 4° CUARTO PASO

- Crearemos un archivo **.html** por ejemplo (**Fichas.html**), en el directorio **JsonInterfaz/**:

```
JsonInterfaz$ touch FichavRegistrado.html
```

- Abrimos el archivo que creamos **.html(Fichas.html)**, insertamos el siguiente código:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
 xml:lang="en" lang="en">
```

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html;
 charset=ISO-8859-1">
```

```
<title>Fichas</title>
```

```
<link rel="stylesheet" type="text/css" media="screen">
```

```
<script src="fichas/dataInventory.js"> </script>
```

```
<script>
```

```
function Fichas() {
```

```
var valor = columnInventory.length;
```

```
var valor1 = dataInventory.length-1;
```

```
var j = 0, flecha = false;
```

```
if (columnInventory[j].sTitle == "cantidad"){
 flecha = true;
```

```
for (var i = 0; i < valor; i++){
```

```
if(flecha == true && i == 2){
```

```
document.write(" "
 + columnInventory[j].sTitle + ": "
 + dataInventory[valor1][j]+ "");
```

```
document.write(" "
 + columnInventory[i+1].sTitle + ": "
```

```

 + dataInventory[valor1][i+1]+ "");
 }
 else{
 document.write(" "
 + columnInventory[i+1].sTitle + ": "
 + dataInventory[valor1][i+1]+ "");
 }
}

}
else{
 for (var i = 0; i < valor; i++){
 document.write(" "
 + columnInventory[i].sTitle + ": "
 + dataInventory[j][i]+ "");
 }
}

}
</script>
</head>
<body>

<div id="tabbed_box_1" class="tabbed_box">
<h4>Ficha de la tabla
<script> document.write (Variable[0]);
</script></h4>

<div class="tabbed_area">
<ul class="tabs">
Ficha

<div id="content_1" class="content">

<script> Fichas ();</script>

</div>
</div>
</div>

</body>
</html>

```

---

**Nota:** Este código sirve para todas la variables, se utilizo estilo css y se muestran los datos del archivo dataInventory.js utilizando código javascript.

---

### 24.1.5 5° QUINTO PASO

- Abrimos el archivo **FichavRegistrado.html** con cualquier navegador web y se nos mostrara de la siguiente forma: *Figura 31: Ficha de un registro*

---

**Nota:**

- Esta imagen nos muestra los datos de una fila de la tabla **vRegistrado**, por ejemplo los datos de la primera fila en forma de una ficha:

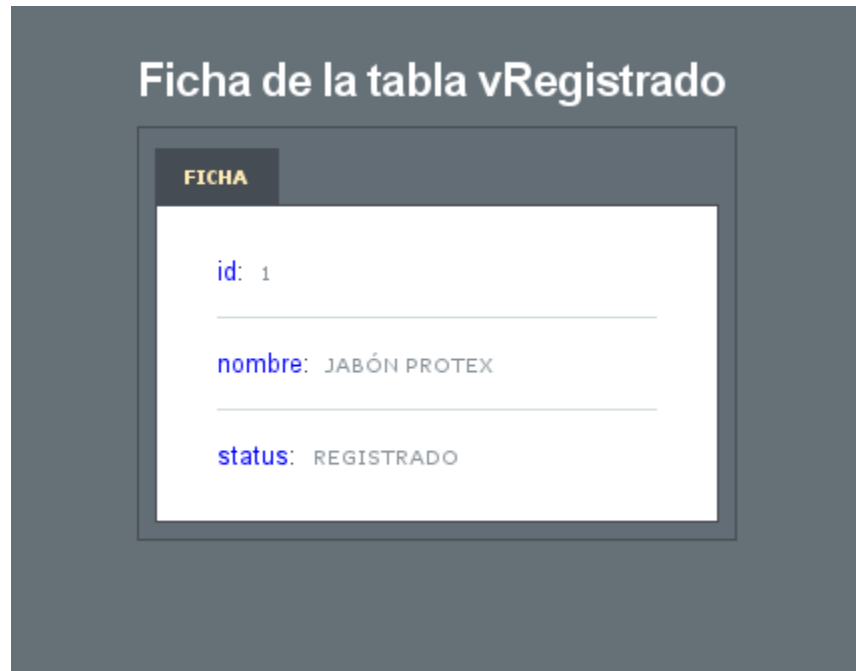


Figura 24.5: **Figura 31: Ficha de un registro**

**La siguiente** *Figura 31: Ficha de un registro* \*\* indica lo siguiente:\*\*

- Nos mostrara el **id** y su valor
  - Nos mostrara el **Nombre** y su valor.
  - Nos mostrara el **Status** y su valor.
- Esta imagen nos muestra los datos de una fila de la tabla **vPedido**, por ejemplo los datos de la primera fila en forma de una ficha:

**La siguiente** *Figura 31: Ficha de un pedido* \*\* indica lo siguiente:\*\*

- Nos mostrara el **id** y su valor
- Nos mostrara el **Nombre** y su valor.
- Nos mostrara el **Cantidad** y su valor.



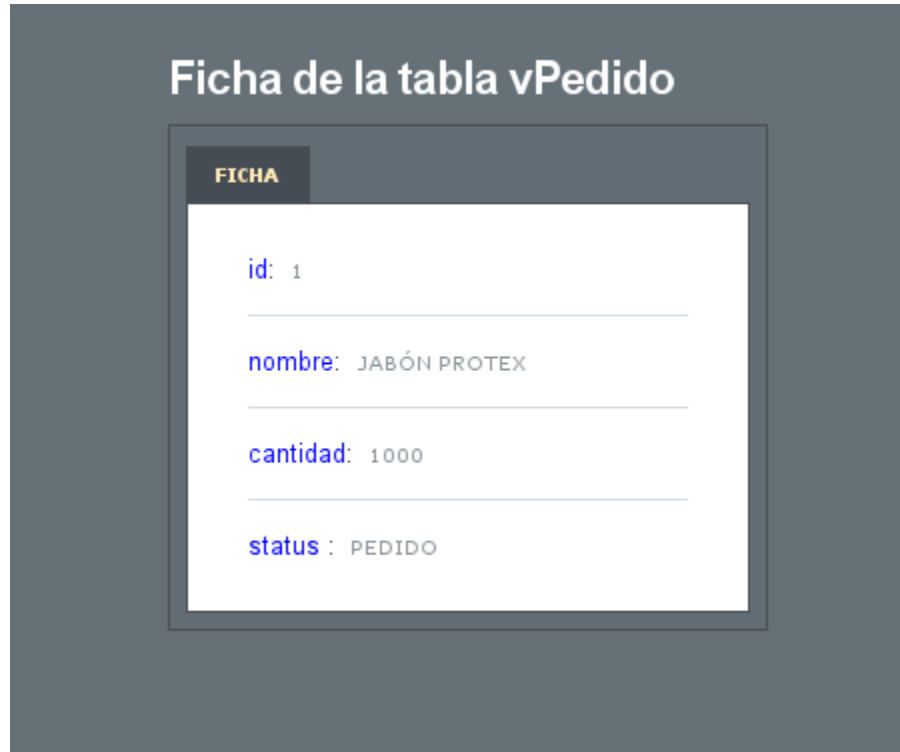


Figura 24.6: **Figura 31: Ficha de un pedido**

- Nos mostrara el **Status** y su valor.

- 
- Aquí esta los 2 ejemplos de las fichas le damos **clik** algunas de ellas:
  - FichavRegistrado.html
  - FichavPedido.html



---

## Ver gráfos generales usando flujos de trabajo

---

NULL

### 7.- Tutorial Inflow



---

## Instalación de Inflow

---

**Pre-requisitos:**

Conocimiento básico de instalación de paquetes, y actividades mínimas de administración.

**Paquetes necesarios:**

1. Librería Trolltech Qt >= 4.4 (**libqt4-dev**).
2. Librería de Conexión a Datos Qt PSQL (**libqt4-sql-psql**).
3. Librería de Conexión a Datos Qt SQLITE (**libqt4-sql-sqlite**).
4. Librería XML2 (**libxml2-dev**).
5. Librería de archivo tarados (**libtar-dev**).
6. Librería Graphviz (**libgraphviz-dev**, para plugins de visualización de gráfico).
7. Librería **libtar-dev** (Utilidad “tar” para respaldo de archivos).
8. Compilador **g++** y **make**.

Desde la shell de Unix, puede instalar todo el conjunto de paquetes necesarios [descritos desde el paso "a" hasta el paso "h"], a través del siguiente comando:

```
aptitude install g++ libqt4-dev libqt4-sql-psql libxml2-dev libqt4-sql-sqlite
```

**Observen la siguiente imagen:**

```
root@debian:/home/cenditel# aptitude install g++ libqt4-dev libqt4-sql-psql libxml2-dev libqt4-sql-sqlite libtar-dev libgraphviz-dev g++ make
Se instalarán los siguiente paquetes NUEVOS:
 libxml2-dev
Se actualizarán los siguientes paquetes:
 libxml2
1 paquetes actualizados, 1 nuevos instalados, 0 para eliminar y 140 sin actualizar.
Necesito descargar 1.804 kB de ficheros. Después de desempaquetar se usarán 2.757 kB.
¿Quiere continuar? [Y/n/?] y
Des: 1 http://security.debian.org/ wheezy/updates/main libxml2 amd64 2.8.0+dfsg1-7+wheezy1 [904 kB]
25% [1 libxml2 444 kB/904 kB 49%] 14,7 kB/s 1min. 32seg.
```

**Figura 25** Paquetes de inflow.

### Primer paso:

Crear un directorio, en el cual almacenará los archivos fuentes.

```
$ mkdir inflow
```

### Segundo paso:

Descargar los fuentes (git ...)

```
$ git clone https://bitbucket.org/victorrbravo/inflow.git
```

### Observen la siguiente imagen:

```
cenditel@debian:~/Inflow$ git clone https://bitbucket.org/victorrbravo/inflow.git
Cloning into 'inflow'...
remote: Counting objects: 1426, done.
remote: Compressing objects: 100% (1172/1172), done.
Receiving objects: 9% (129/1426), 1.41 MiB | 40 KiB/s
```

**Figura 26** inflow.

### Tercer paso:

Ir al directorio descomprimido `cd inflow/`

```
$ cd inflow
inflow$
```

### Cuarto paso:

Ejecutar "qmake" o "qmake-qt4" para generar el archivo de compilación (Makefile)

```
$ qmake-qt4
```

### Observen la siguiente imagen:

```
cenditel@debian:~/Inflow/inflow$ qmake-qt4
Project MESSAGE: Configuración y compilación de SAFET
Project MESSAGE: ... Verificación de librería Libxml2: OK
Project MESSAGE: ... Verificación de encabezados de Libgraphviz: OK
Project MESSAGE: ... Verificación de librería Libgraphviz: OK
Project MESSAGE: ... Verificación de encabezados de libtar-dev: OK
Project MESSAGE: ... Verificación de compilador g++ : OK
```

**Figura 27** qmake-qt4.

### Quinto paso:

Si en la pantalla no se muestran errores, realizar la compilación con "make".

```
$ make
```

### Observen la siguiente imagen:

```
cenditel@debian:~/Inflow/inflow$ make
cd src/ && make -f Makefile
make[1]: se ingresa al directorio `/home/cenditel/Inflow/inflow/src'
/usr/bin/qmake-qt4 -o Makefile src.pro
Project MESSAGE: Configuring for xml2...
Project MESSAGE: Configuring for gsoap...
Project MESSAGE: Configuring for xml2...
Project MESSAGE: Configuring for gsoap...
Project MESSAGE: Configuring for xml2...
Project MESSAGE: Configuring for gsoap...
make[1]: se sale del directorio `/home/cenditel/Inflow/inflow/src'
make[1]: se ingresa al directorio `/home/cenditel/Inflow/inflow/src'
make -f Makefile.Release
make[2]: se ingresa al directorio `/home/cenditel/Inflow/inflow/src'
g++ -c -m64 -pipe -O2 -fPIC -w -D REENTRANT -DQT_WEBKIT -DQT_NO_DEBUG -DSAFET_N
O_DBXML -DSAFET_XML2 -DSAFET_TAR -DSAFET_GSOAP -DQT_NO_DEBUG -DQT_WEBKIT_LIB -D
QT_SVG_LIB -DQT_SQL_LIB -DQT_XML_LIB -DQT_GUI_LIB -DQT_NETWORK_LIB -DQT_CORE_LI
B -DQT_SHARED -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -I/usr/include/qt4/QtCo
re -I/usr/include/qt4/QtNetwork -I/usr/include/qt4/QtGui -I/usr/include/qt4/QtX
ml -I/usr/include/qt4/QtSql -I/usr/include/qt4/QtSvt -I/usr/include/qt4/QtWebKi
```

**Figura 28** make.

#### Sexto paso:

Realizar la instalación en los directorios del sistema (es necesario tener perm

```
$ su
```

Escriba la contraseña de administrador y asegúrese que se está en el directori

#### Séptimo paso:

Instalación como administrador

```
make install
```

;Si no se presenta ningún mensaje de error SAFET está instalado correctamente

#### Observen la siguiente imagen:

```
root@debian:/home/cenditel/Inflow/inflow# make install
cd src/ && make -f Makefile install
make[1]: se ingresa al directorio `/home/cenditel/Inflow/inflow/src'
make -f Makefile.Release install
make[2]: se ingresa al directorio `/home/cenditel/Inflow/inflow/src'
install -m 644 -p /home/cenditel/Inflow/inflow/src/libdotar.h /usr/include/libs
afet/
install -m 644 -p /home/cenditel/Inflow/inflow/src/SafetAutofilter.h /usr/inclu
de/libsafet/
install -m 644 -p /home/cenditel/Inflow/inflow/src/SafetBinaryRepo.h /usr/inclu
de/libsafet/
install -m 644 -p /home/cenditel/Inflow/inflow/src/SafetCipherFile.h /usr/inclu
de/libsafet/
install -m 644 -p /home/cenditel/Inflow/inflow/src/SafetCondition.h /usr/includ
e/libsafet/
install -m 644 -p /home/cenditel/Inflow/inflow/src/SafetConfFile.h /usr/include
/libsafet/
install -m 644 -p /home/cenditel/Inflow/inflow/src/SafetConfiguration.h /usr/in
clude/libsafet/
install -m 644 -p /home/cenditel/Inflow/inflow/src/SafetConnection.h /usr/inclu
```

**Figura 29** make install.





## Visualización de la interfaz gráfica de inflow

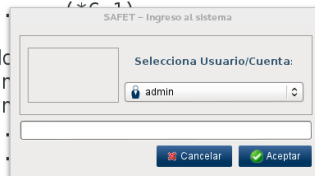
Desde la línea de comando como usuario normal podemos observar ejemplo con inflow

### Primer paso:

Interfaz gráfica de infow  
\$ inflow

### Observen la siguiente imagen:

```
cenditel@debian:~$ inflow
...window...
...MainWindow::doSaveConfiguration...connect
...plugname:
...contains...empty
MainWindow::MainWindow.....(5)...
MainWindow::MainWindow.....(*6)...
MainWindow::MainWindow.....(*6,1)...
DomModel::commandslist count:
(1)...opcion...DockSbMenu::add
QMainWindow::addDockWidget: in
QMainWindow::addDockWidget: in
MainWindow::MainWindow.....
MainWindow::MainWindow.....
MainWindow::MainWindow.....(*6,4)...
MainWindow::MainWindow.....(*6,5)...
MainWindow::MainWindow.....(7)...
..asignado archivo log: /home/cenditel/.safet/log/safet.log
MainWindow::MainWindow.....(8)...
MainWindow::restoreWindowState...docksinfo: 71
... IconWidget(QWidget *parent) ... 1
... IconWidget(QWidget *parent) ... 2
```



**Figura 30** inflow.

### Segundo paso:

Introducimos al usuario/password por defecto admin/admin

### Observen la siguiente imagen:

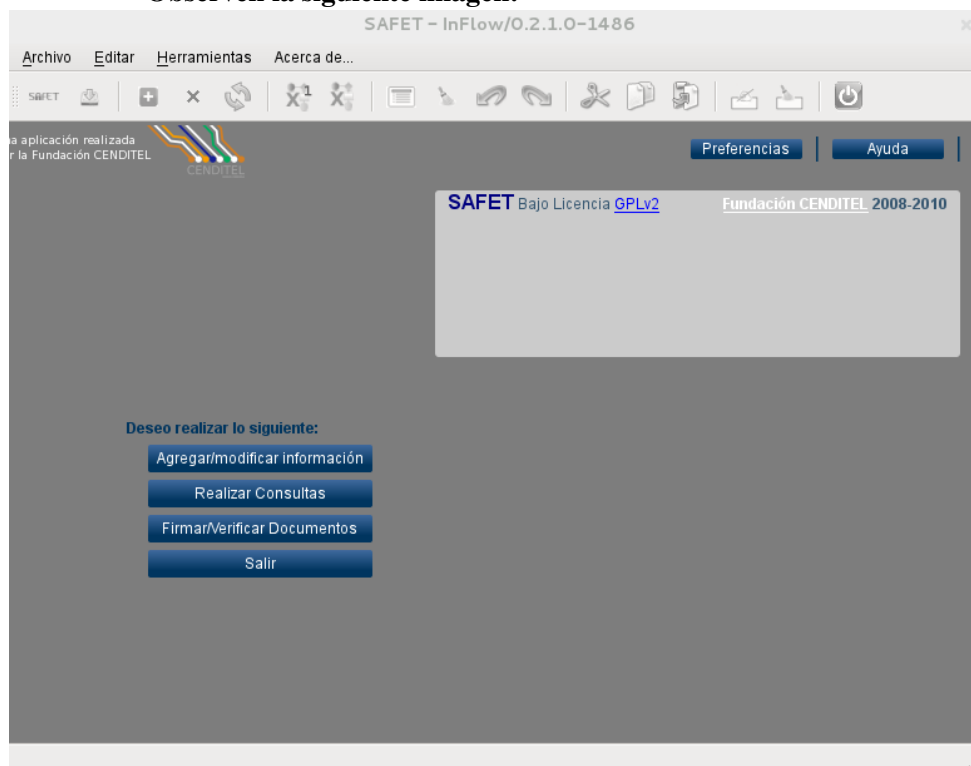


**Figura 31** usuario/password.

### Tercer paso:

Observamos la entrada de inflow.

### Observen la siguiente imagen:

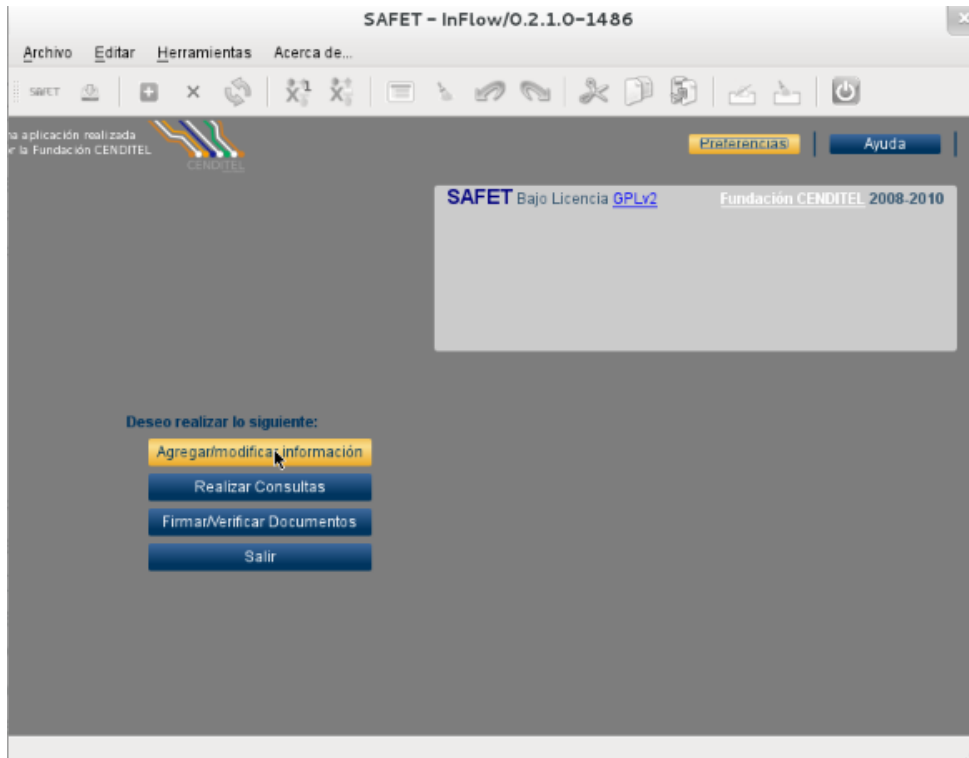


**Figura 32** Entrada inflow.

### Cuarto paso:

Pulsamos algunas de las acciones a realizar, en este caso pulsaremos la opción Ag

### Observen la siguiente imagen:

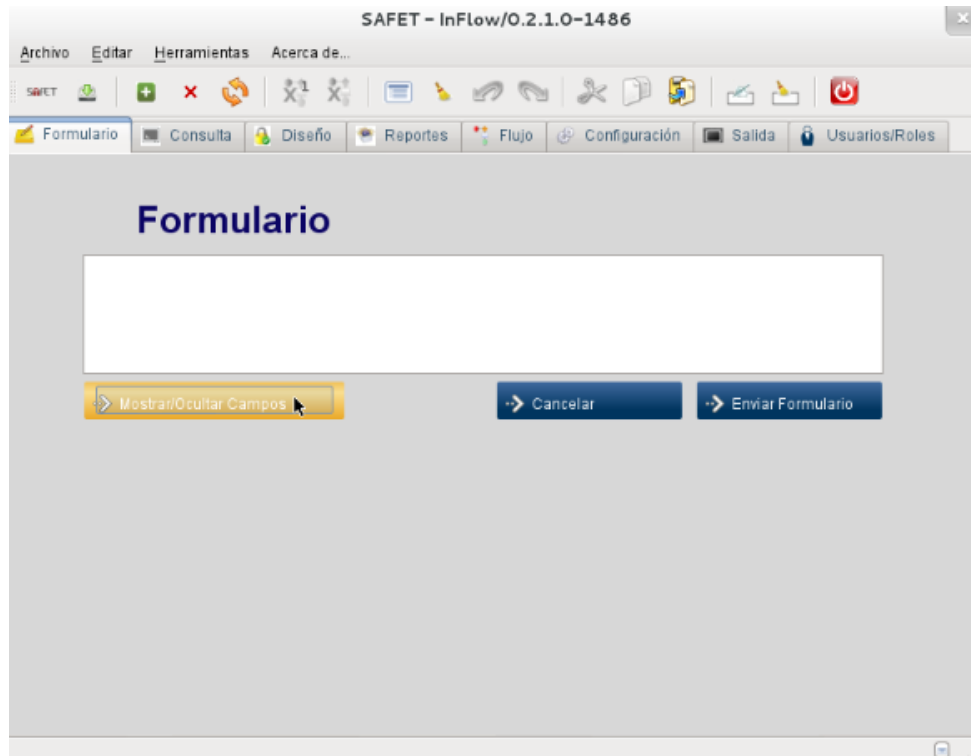


**Figura 33** Acciones inflow.

**Quinto paso:**

Pulsamos el botón acción Mostrar/Ocultar Campos

**Observen la siguiente imagen:**

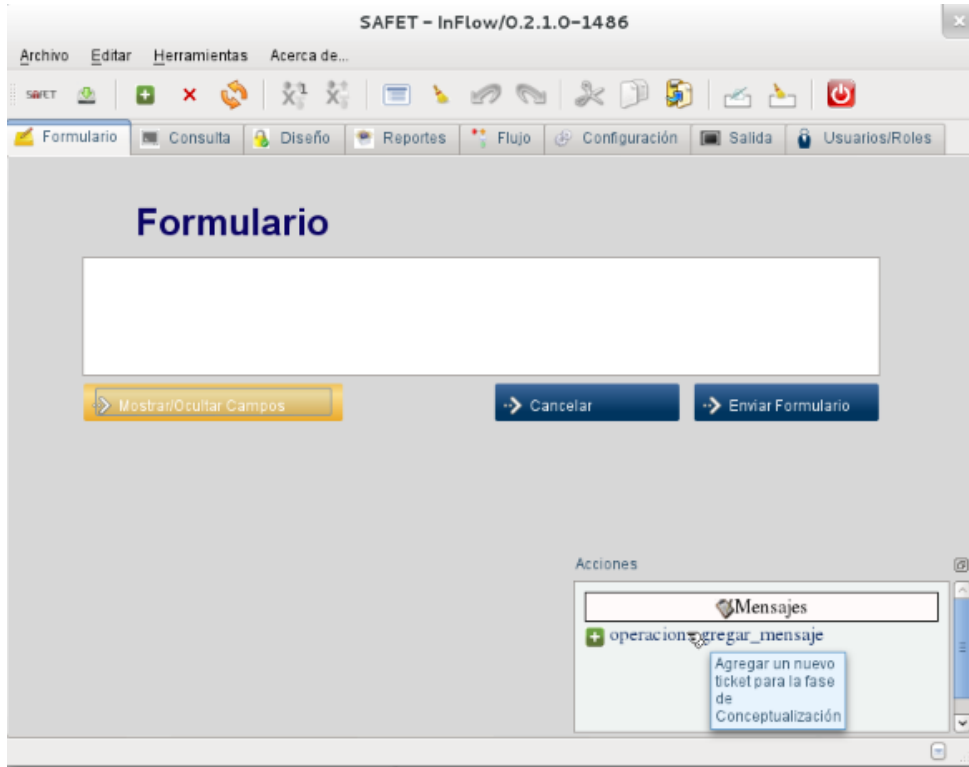


**Figura 34** botón Mostrar/Ocultar.

### Sexto paso:

Nos mostrara las acciones a realizar, pulsamos alguna operación que deseamos real

**Observen la siguiente imagen:**

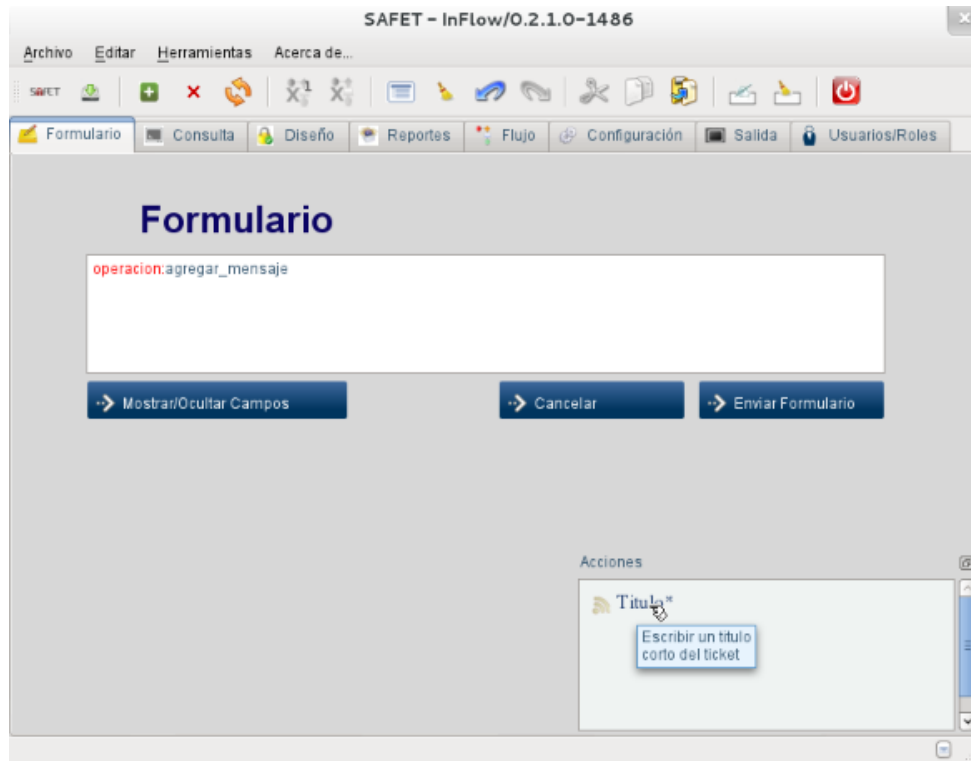


**Figura 35** Operaciones.

**Séptimo paso:**

- Nos mostrara la operación que va a realizar que seria "agregar\_mensaje", Nos pide
- Pulsamos en la acción el título\* para agregarlo

**Observen la siguiente imagen:**

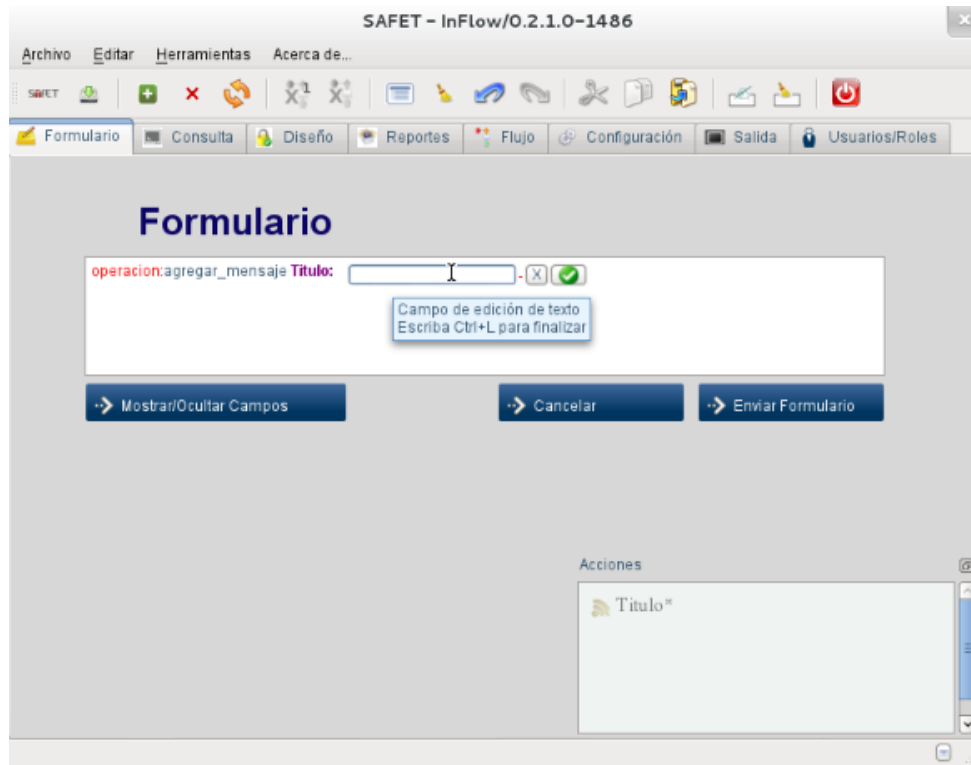


**Figura 36** titulo\*.

### **Octavo paso:**

Nos mostrar en formulario en banco para escribir el titulo.

**Observen la siguiente imagen:**

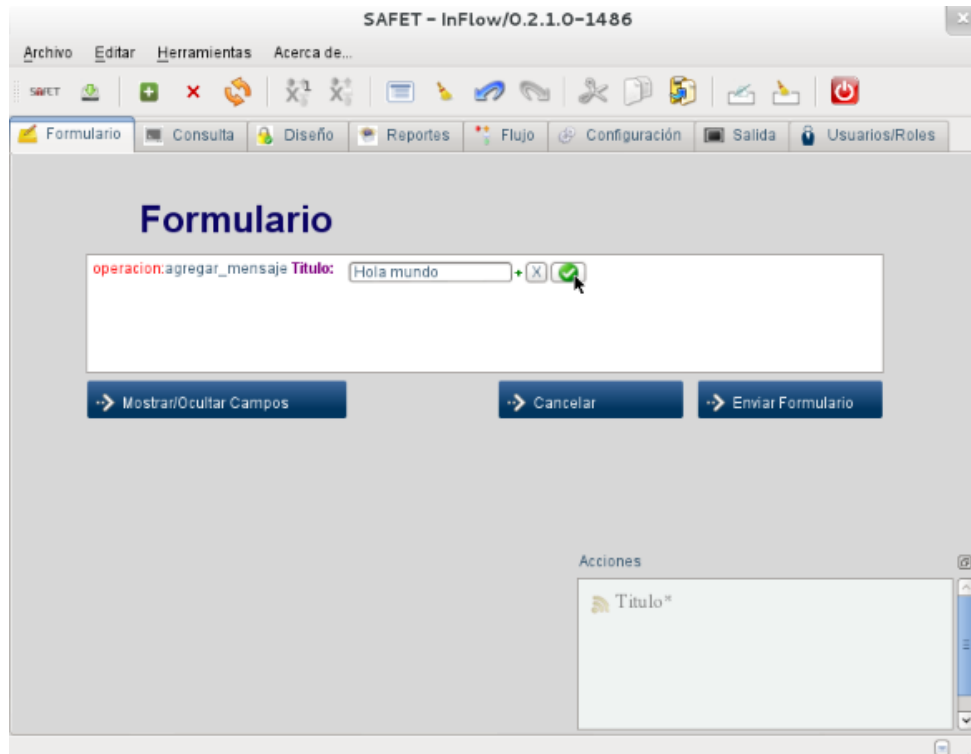


**Figura 37** Formulario.

**Noveno paso:**

Se llena el formulario y pulsamos el botón con la flecha verde para aceptar.

**Observen la siguiente imagen:**



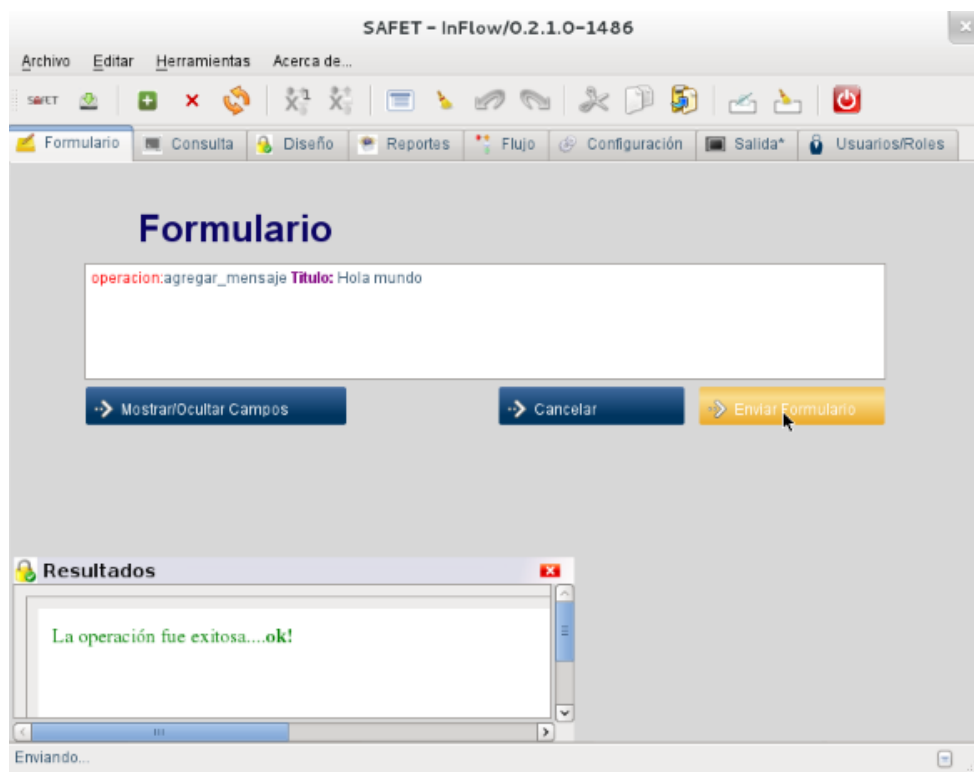
**Figura 38** botón.

### Décimo paso:

- Pulsamos el botón que dice "Enviar Formulario" para finalizar la operación.
- Mostrara un mensaje de "Resultados" diciendo "La operación fue exitosa....ok".
- Listo ya fue agregado el mensaje

**Observen la siguiente imagen:**





**Figura 39** Operación exitosa.

LICENCIA



Figura 27.1: Documentación PySafet by Cenditel Mérida - Venezuela is licensed under a [Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional License](http://creativecommons.org/licenses/by-nc-nd/4.0/). Creado a partir de la obra en <http://seguridad.cenditel.gob.ve/safet/doc>. Puede hallar permisos más allá de los concedidos con esta licencia en <http://seguridad.cenditel.gob.ve/safet/doc>.



## A

Archivos fuentes, [19](#)

Arquitectura, [11](#)

## C

Clase MainWindow(Safet.MainWindow), [39](#)

Clase SafetDocument, [47](#)

Clase SafetVariable, [51](#)

Clase SafetWorkflow, [55](#)

Clase SafetXmlObject (Safet.SafetXmlObject), [45](#)

Clase SafetYAWL, [53](#)

Creación de las tablas en base de datos, [57](#)

Crear el flujo de trabajo utilizando un archivo XML,  
[81](#)

## D

Debian Wheezy 7.0 (64 bits/32bits), [15](#)

## E

Estructura del directorio .safet/, [23](#)

## F

Funciones, [16](#), [20](#)

## I

Instalación de Inflow, [137](#)

## M

Motivación y objetivos, [7](#)

## O

Operaciones CRUD+(flujo) utilizando PySafet, [65](#)

## P

Paso para la creación del directorio .safet/, [21](#)

Pasos para la compilación, [20](#)

## S

Sistema Automatizado de Firma Electrónica y Estampado Electrónico, [13](#)

## U

Ubuntu (32 bits/ 64 bits), [17](#)

## V

Ver datos usando flujos de trabajo, [113](#)

Ver gráfos generales usando flujos de trabajo, [135](#)

Verificación de la instalación, [20](#)

Visualización de la interfaz gráfica de inflow, [141](#)