

1^{er} Encuentro: Tryton (desarrollo rápido en Python)

Capacitación técnica

Una introducción al desarrollo en Tryton

Autores

Francisco M. Moyano Casco

Francisco Arata

Carlos Scotta

Fernando Sassetti

Ingrid Spessotti



Temario

1. Presentación

- 1.1 Características generales de Tryton
- 1.2 Herramientas de uso e instalación

2. Un módulo básico

- 2.1 Crear un modelo
- 2.2 Agregar campos a un modelo
- 2.3 Los campos del modelo: su representación gráfica y en base de datos
- 2.4 La presentación gráfica de los datos
- 2.5 Estructura de nuestro directorio
- 2.6 Arrancando el cliente
- 2.7 Instalando el modulo creado
- 2.8 Responder a las acciones del usuario: on_change



Temario

3. Características avanzadas

- 3.1 Workflows
- 3.2 Mejorando las vistas
- 3.3 Los campos function
- 3.4 Wizards (asistentes)
- 3.5 Cómo extender objetos preexistentes

4. Temas adicionales

- 4.1 Reportes
- 4.2 Permisos y reglas de acceso
- 4.3 proteus

5. Referencias y fuentes



1.1 Características Generales de Tryton

Tryton es una plataforma para aplicaciones de propósito general. Puede utilizarse para desarrollar sistemas con diferentes objetivos: ERP para empresas(así fue concebido), software para establecimientos de salud (como GNU Health) o para universidades, tiendas virtuales para ventas on-line, etc.

Características

- . Escrito en Python
- . Modular
- . Arquitectura de tres capas
 - Cliente de escritorio en GTK (tryton) y cliente web en javascript (sao, en desarrollo)
 - Servidor (trytond)
 - Gestor de base de datos (DBMS): PostgreSQL (también SQLite y MySQL)



1.2 Herramienta de uso e instalación

pip

Herramienta para instalar y gestionar paquetes de Python

virtualenv

Herramienta para crear entornos controlados de Python. Permite tener diferentes versiones de algunas librerías y controlar las dependencias instaladas para cada entorno

virtualenvwrapper

Conjunto de extensiones hechas sobre virtualenv para simplificar su uso.

Sistemas de Control de Versiones

Hay que mencionar además la gran utilidad que tienen sistemas como Mercurial o Git, que permiten llevar registro de las distintas modificaciones hechas al código.



1.2 Herramienta de uso e instalación

Instalación de pip y virtualenvwrapper

```
$ sudo apt-get install python-pip  
  
$ pip install --user virtualenvwrapper
```

Configuración de virtualenvwrapper

Hay que indicarle al sistema cómo trabajar con virtualenvwrapper. Podemos crear una carpeta llamada *.virtualenvs/*, en el home, y usarla para contener los entornos creados. Agregar en el archivo */home/usuario/.bashrc* las siguientes líneas:

```
export WORKON_HOME=$HOME/.virtualenvs  
source $HOME/.local/bin/virtualenvwrapper.sh
```

De esta manera, en el home tendremos dos carpetas:

- *\$HOME/.local/*:
todo lo que se instale con pip con la opción *--user*
- *\$HOME/.virtualenvs/*:
los entornos virtuales creados con *virtualenv[wrapper]*



1.2 Herramienta de uso e instalación

Creación del entorno e instalación de Tryton

```
$ mkvirtualenv mi_entorno  
(mi_entorno)$ pip install trytond'>3.8,<3.9'
```

Dependencias de Tryton

Es importante chequear cuáles son las dependencias para ejecutar Tryton en determinadas condiciones. Por ejemplo, para ejecutarlo en combinación con el gestor PostgreSQL es necesario instalar el paquete *psycpg2*

Para una referencia completa, dentro de la carpeta del servidor Tryton leer las indicaciones del archivo *INSTALL*

Para encontrar la carpeta donde esta instalado Tryton en nuestro entorno virtual:

```
(mi_entorno)$ cdsitepackages  
(mi_entorno)~/.virtualenvs/mi_entorno/lib/python2.7/site-packages$
```



1.2 Herramienta de uso e instalación

Instalación de módulos del repositorio oficial de Python

```
(mi_entorno)$ pip install trytond_party'>3.8,<3.9'
```

Esto instala el módulo party en su versión 3.8 mas actual, junto con sus dependencias: country, dentro de la carpeta modules del servidor trytond, en el entorno virtual. Es decir que en este caso instalamos dos módulos.

El comando pip list también permite ver los paquetes instalados

```
(mi_entorno)$ pip list  
Genshi (0.7)  
lxml (3.4.4)  
pip (6.1.1)  
polib (1.0.6)  
psycpg2 (2.6)  
python-dateutil (2.4.2)  
python-sql (0.6)  
pytz (2015.2)  
relatorio (0.6.1)  
[...]
```



1.2 Herramienta de uso e instalación

Como activar un entorno, desactivarlo y removerlo

```
$ workon mi_entorno  
(mi_entorno)$ ....  
(mi_entorno)$ deactivate  
$rmvirtualenv mi_entorno
```

Otros comandos que podrían ser útiles

```
$lsvirtualenv // lista los entornos virtuales creados  
$cpvirtualenv mi_entorno mi_entorno2 // para copiar los entornos virtuales  
$mvvirtualenv // para mover o renombrar un entorno virtual
```



1.2 Herramienta de uso e instalación

El archivo trytond.conf

```
[jsonrpc]
listen=*:8000
data=/var/www/localhost/tryton

[database]
uri=postgresql://usuario:password@localhost:5432
path=/var/lib/tryton/data

[session]
timeout=3600
super_pwd=V6imlhDMI0fiY
```

¡Que es qué!

jsonrpc: define la interfaz de red.

- listen define si se establecerá una conexión local o abierta (si se conectara con otros dispositivos)
- data se las debo

database: adonde van a parar los datos persistentes

session: parametros de sesión



1.2 Herramienta de uso e instalación

La contraseña de super usuario

En el archivo de configuración la contraseña de super usuario tryton debe estar encriptada. Para encriptarla contraseña podemos ejecutar el siguiente comando en una terminal. El resultado que se obtiene es el que hay que agregar en el archivo de configuración, como valor de super_pwd.

```
$ python -c 'import getpass,crypt,random,string;
print crypt.crypt(getpass.getpass(),
""'.join(random.sample(string.ascii_letters + string.digits, 8)))'
```



1.2 Herramienta de uso e instalación

¡¿NADIE PREGUNTO COMO SE CREA EL USUARIO DE POSTGRES?!

En el archivo de configuración, necesitamos para tener acceso a la capa de la base de datos con un usuario de la misma con su correspondiente contraseña.

Recordando la línea de dicho archivo:

```
[...]  
[database]  
uri=postgresql://usuario:password@localhost:5432  
path=/var/lib/tryton/data  
[...]
```

Vemos que nuestro usuario *postgres* se llamará *usuario* y su correspondiente password será *password* (valga la redundancia). Lo que sigue de *@* corresponde a la localización de nuestra capa persistente, que en este caso es *localhost*, la misma máquina donde se este ejecutando el servidor *trytond*, que tiene el puerto *5432* escuchando, que es el que esta configurado por defecto en *postgres*.

Para crear el susodicho usuario, bastara entonces abrir un terminal y hacer

```
$sudo su - postgres -c "createuser --createdb --no-createrole --no-superuser -W usuario1"
```

Y a continuación se nos pedirá definir una contraseña.

Para saber que significa cada parámetro ingresado en la línea anterior, hacemos

```
$sudo su - postgres -c "createuser --help"
```



1.2 Herramienta de uso e instalación

Arrancamos el servidor!

```
$ workon mi_entorno
(mi_entorno)$ trytond -c RUTA_AL_TRYTOND_CONF/trytond.conf -v
[2017-05-12 12:16:16,718] INFO trytond.server using RUTA_AL_TRYTOND_CONF/trytond.conf as configuration file
[2017-05-12 12:16:16,718] INFO trytond.server initialising distributed objects services
[2017-05-12 12:16:16,745] INFO trytond.server starting JSON-RPC protocol on *:8100
```

¡DEBE ESTAR BIEN HECHO EL ARCHIVO DE CONFIGURACION, PORQUE NO MUESTRA ERROR!



2. Un módulo básico

Un módulo mínimo de Tryton consiste de dos archivos:

`__init__.py`

Archivo estándar de todos los módulos de Python

`tryton.cfg`

Permite establecer cuáles son las dependencias del módulo y los archivos XML adicionales que deben ser utilizados

Ejemplo de `tryton.cfg`

```
[tryton]
version=3.8.0
depends:
    party

xml:
    opportunity.xml
```



2.1. Crear un modelo

Para empezar a trabajar, nos ubicaremos en la carpeta *modules* dentro de *trytond*, en nuestro entorno virtual, que es donde se encontraran los módulos de nuestro servidor, ya sean propios o de algún repositorio.

En Tryton un modelo es una clase normal de Python, que hereda de la clase *Model*. Pero para que el modelo persista en la base de datos hay que heredar de *ModelSQL*.

¿Que significa esto? → que nuestro modelo estará representado por una tabla en la capa persistente (la base de datos). Luego veremos como agregarle atributos.

Para nuestro ejemplo vamos a crear un archivo llamado *opportunity.py*, en el que ingresaremos el siguiente código

```
from trytond.model import ModelSQL, ModelView, fields

__all__ = ['Opportunity']

class Opportunity(ModelSQL, ModelView):
    'Opportunity'
    __name__ = 'training.opportunity'
    _rec_name = 'description'
```



2.1. Crear un modelo

Para que Tryton tome nota de la existencia de este modelo es necesario registrarlo en el Pool de modelos. En el pool se encontraran todos los modelos definidos en los distintos modulos que vayamos agregando a nuestro sistema.

En `__init__.py`:

```
from trytond.pool import Pool
from .opportunity import *

def register():
    Pool.register(
        Opportunity,
        module='training', type_='model')
```



2.2. Agregar campos a un modelo

Volvemos a **opportunity.py**, donde empezamos entonces a agregar los atributos a nuestro modelo:

```
from trytond.model import ModelSQL, ModelView, fields

__all__ = ['Opportunity']

class Opportunity(ModelSQL, ModelView):
    'Opportunity'|
    __name__ = 'training.opportunity'
    _rec_name = 'description'

    description = fields.Char('Description', required=True)
    start_date = fields.Date('Start Date', required=True)
    end_date = fields.Date('End Date')
    party = fields.Many2One('party.party', 'Party', required=True)
    comment = fields.Text('Comment')
```



2.2. Agregar campos a un modelo

Para definir los atributos (campos) de la clase se utiliza *fields*. Existen distintos tipos de campos, por ejemplo Char, Integer, Boolean, Date, entre otros. Cada uno de ellos tiene atributos que establecen su comportamiento:

string	Nombre o etiqueta para el campo
required	Si se le asigna True, el campo es requerido
readonly	Si se le asigna True, no se permite editar el campo desde la interfaz de usuario
domain	Una lista que define un dominio o subconjunto de registros válidos
states	Es un diccionario de Python con al menos alguna de las siguientes claves: required, readonly e invisible. Los valores posibles son expresiones python que se evalúan con valores de cada registro

2.3. Los campos del modelo: su representación gráfica y en base de datos

Al desarrollar una aplicación utilizando la plataforma Tryton tenemos la ventaja de que fácilmente podemos ver representados nuestros campos en una base de datos (capa de persistencia de datos) y también de manera gráfica (capa de presentación).

En cierta medida estamos hablando del mapeo objeto-relacional (conocido por su nombre en inglés, Object-Relational mapping, o sus siglas ORM).

Veamos algunos ejemplos:

fields.Char

En un modulo Tryton:

```
description = fields.Char('Description', required=True)
```

En la interfaz gráfica

Description:

En la base de datos (SQL)

```
description VARCHAR NOT NULL
```



2.3. Los campos del modelo: su representación gráfica y en base de datos

fields.Date

En un modulo Tryton:

```
end_date = fields.Date('End Date')
```

En la interfaz gráfica

End Date:

En la base de datos (SQL)

```
end_date DATE
```

fields.Many2One

En un modulo Tryton:

```
party = fields.Many2One('party.party', 'Party', required=True)
```

En la interfaz gráfica

Party:

En la base de datos (SQL):

```
party INTEGER NOT NULL
```

```
FOREIGN KEY (party) REFERENCE party_party(id)
```



2.4. La presentación gráfica de los datos

Para poder usar la capa de presentación, el modelo debe heredar ModelView

```
class Opportunity(ModelSQL, ModelView):
```

La representación gráfica se determina por medio de archivos XML.

Hay que indicarle a Tryton de dónde tomar la definición XML. Y para ello hay que crear un archivo llamado opportunity.xml

En el archivo tryton.cfg:

```
xml:  
    opportunity.xml
```

Una vista (View) es un objeto más de Tryton; se almacena en base de datos y guarda información que toma el servidor para construir la representación gráfica.

Hay dos tipos principales de vista (hay mas):

- **Tree:** es una lista de registros
- **Form:** es una vista de formulario, que permite crear y editar un registro



2.4. La presentación gráfica de los datos

Nuestro archivo *opportunity.xml* debe contener las siguientes etiquetas, dentro de las cuales iremos definiendo la representación gráfica:

```
<?xml version="1.0"?>
<tryton>
  <data>

  </data>
</tryton>
```



2.4. La presentación gráfica de los datos

Vista Tree

En la definición se hace referencia al objeto View (ir.ui.view) que va a contener la vista.

En los campos de este objeto se define el modelo que está asociado a la vista (training.opportunity), el tipo (tree) y el nombre del archivo que contiene la definición: opportunity_list

```
<record model="ir.ui.view" id="opportunity_view_list">
  <field name="model">training.opportunity</field>
  <field name="type">tree</field>
  <field name="name">opportunity_list</field>
</record>
```

El nombre de archivo al que se alude está alojado por defecto en una subcarpeta del módulo:
view/opportunity_list.xml

```
<?xml version="1.0"?>
<tree string="Opportunities">
  <field name="party"/>
  <field name="description"/>
  <field name="start_date"/>
  <field name="end_date"/>
</tree>
```



2.4. La presentación gráfica de los datos

Vista Form

También en este caso tenemos la referencia al objeto View (ir.ui.view).

La diferencia es que el tipo es form, y el nombre del archivo que contiene la definición es opportunity_form

```
<record model="ir.ui.view" id="opportunity_view_form">
  <field name="model">training.opportunity</field>
  <field name="type">form</field>
  <field name="name">opportunity_form</field>
</record>
```

Otra vez, el nombre de archivo al que se alude está alojado por defecto en una subcarpeta del módulo:
view/opportunity_list.xml

```
<?xml version="1.0"?>
<form string="Opportunity">
  <label name="party"/>
  <field name="party"/>
  <label name="description"/>
  <field name="description"/>
  <label name="start_date"/>
  <field name="start_date"/>
  <label name="end_date"/>
  <field name="end_date"/>
  <separator name="comment" colspan="4"/>
  <field name="comment" colspan="4"/>
</form>
```



2.4. La presentación gráfica de los datos

Armando las piezas

Definidas las vistas, sólo falta armarlas o pegarlas, para que actúen de manera coordinada.

Para ello se utiliza el objeto `ActionActWindow`, que define un tipo particular de acción, en este caso, abrir una ventana.

En `opportunity.xml`:

```
<record model="ir.action.act_window" id="act_opportunity_open">
  <field name="name">Opportunities</field>
  <field name="res_model">training.opportunity</field>
</record>
```

Y vinculadas a esta nueva acción, dos acciones/vistas (pueden haber mas)

```
<record model="ir.action.act_window.view" id="act_opportunity_open_view1">
  <field name="sequence" eval="10"/>
  <field name="view" ref="opportunity_view_list"/>
  <field name="act_window" ref="act_opportunity_open"/>
</record>
```

```
<record model="ir.action.act_window.view" id="act_opportunity_open_view2">
  <field name="sequence" eval="20"/>
  <field name="view" ref="opportunity_view_form"/>
  <field name="act_window" ref="act_opportunity_open"/>
</record>
```



2.4. La presentación gráfica de los datos

Agregar entradas al menú

Por último, esta acción es disparada por el usuario por medio de una entrada del menú de Tryton, que se puede crear de la siguiente manera:

```
<menuitem name="training" id="menu_training"/>  
  
<menuitem parent="menu_training" action="act_opportunity_open"  
          id="menu_opportunity_form"/>
```

Otros parámetros para <menuitem ... />

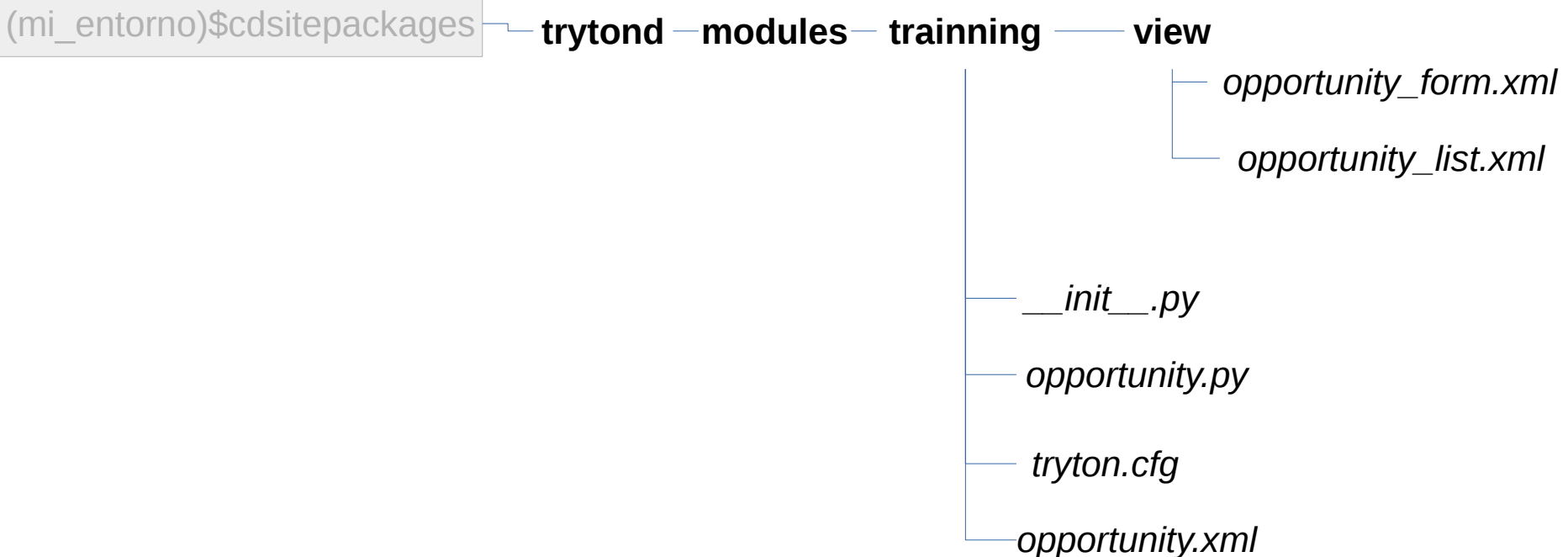
- *sequence*
- *icon*



2.5 Estructura de nuestro directorio

La disposición de nuestro módulo

Ya tenemos los archivos dispuestos y ubicados adecuadamente en sus respectivas rutas
Una propuesta de como nos debería quedar podría ser:



Sí es la primera vez que corremos nuestro servidor, y aún no hemos creado ninguna base de datos para la misma, se reconocerá automáticamente los módulos bajo la carpeta *modules*, siempre y cuando estén bien definidas.

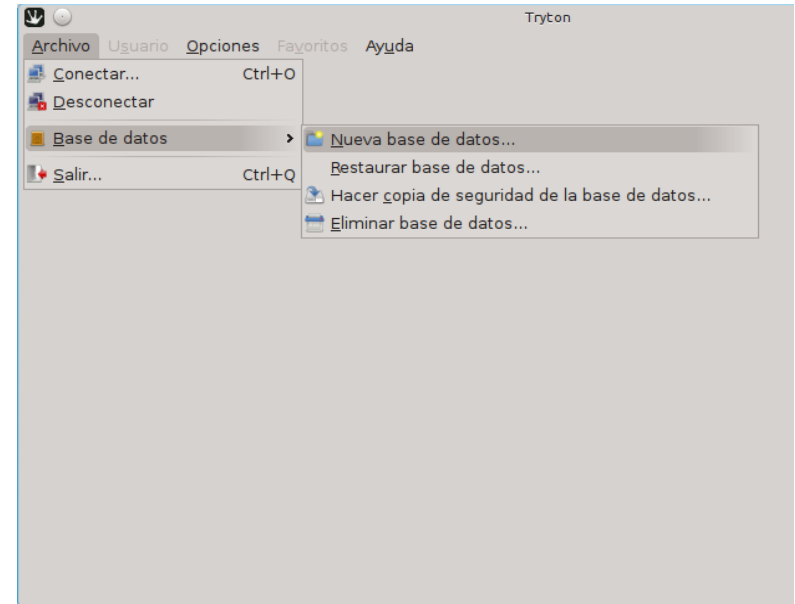
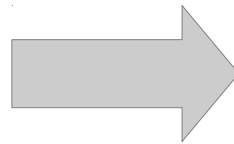
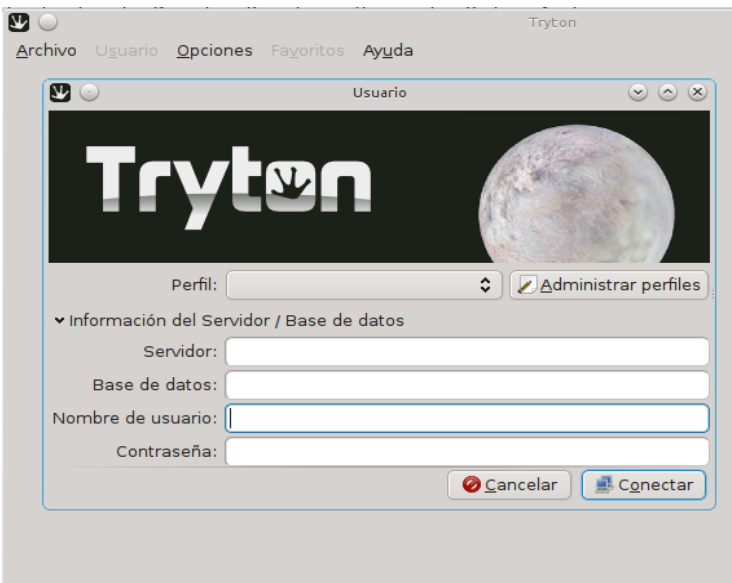


2.6 Arrancando el cliente

Creando una nueva base de datos.

Puede ser creada, una vez iniciado el servidor, desde la aplicación cliente.

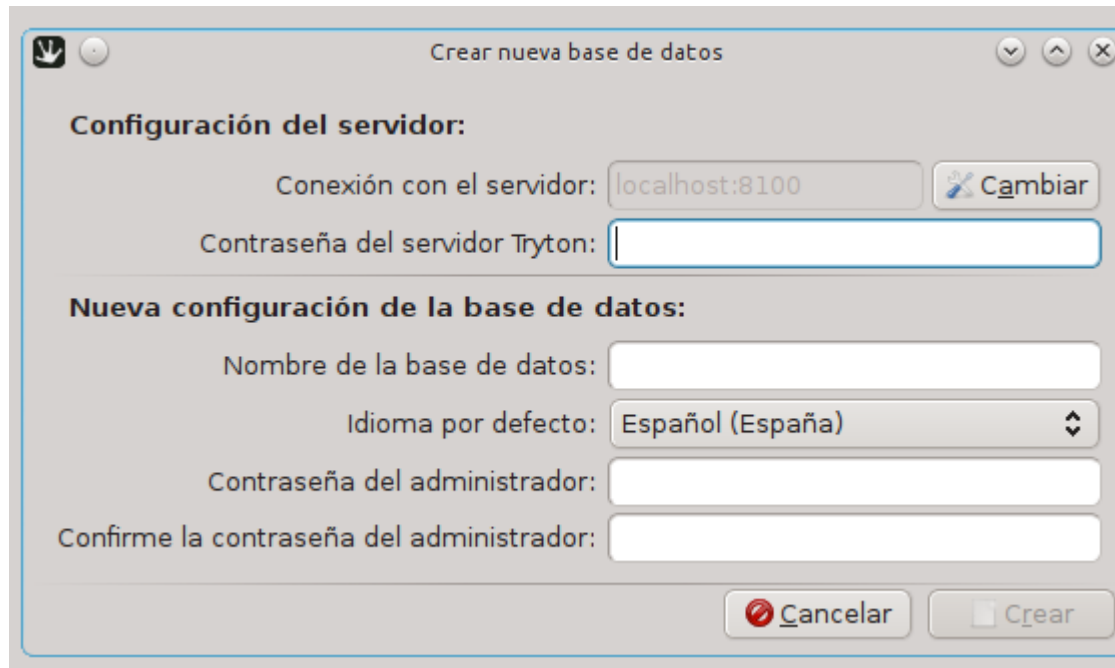
Cuando iniciemos el cliente, nos encontraremos con la siguiente pantalla, donde deberemos ir al menú *Archivo*, cerrando la ventana de login, y seleccionando *Base de datos*→ *Nueva base de datos*...



2.6 Arrancando el cliente

Creando una nueva base de datos.

Tendremos entonces una ventana donde deberemos poner los parametro adecuados, que tendrán que coincidir con los del archivo trytond.conf



The screenshot shows a window titled "Crear nueva base de datos" (Create new database). It contains two main sections: "Configuración del servidor:" (Server configuration) and "Nueva configuración de la base de datos:" (New database configuration). In the server configuration section, there is a text field for "Conexión con el servidor:" (Connection to the server) containing "localhost:8100" and a "Cambiar" (Change) button. Below it is a text field for "Contraseña del servidor Tryton:" (Tryton server password). In the database configuration section, there is a text field for "Nombre de la base de datos:" (Database name), a dropdown menu for "Idioma por defecto:" (Default language) set to "Español (España)", and two text fields for "Contraseña del administrador:" (Administrator password) and "Confirme la contraseña del administrador:" (Confirm administrator password). At the bottom right, there are "Cancelar" (Cancel) and "Crear" (Create) buttons.

Una vez hecho esto, habrá que aguardar un momento a que se cree la misma, lo que llevará algún tiempo. Mientras se puede ir viendo los mensajes en el *terminal* donde esta corriendo el servidor, para poder leer en detalle el proceso de la creación de la base de datos.

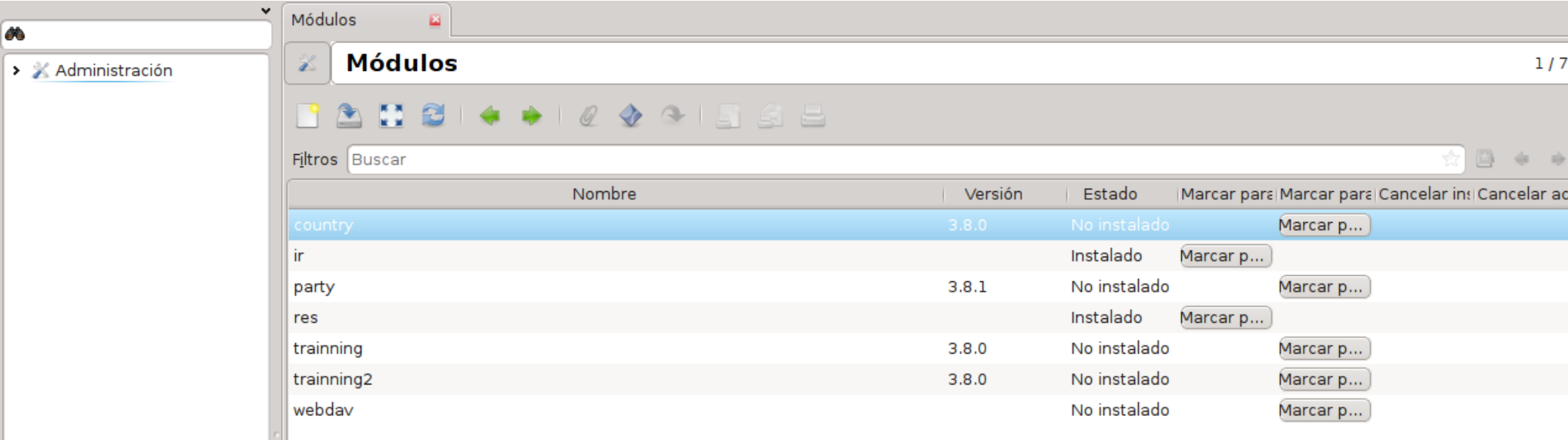
Luego de creada la base de datos, si todo salio bien, no habrá mensaje de errores. Si los hubiera, hay que copiar la traza del error y estudiarla, para poder determinar en donde se encuentra.

2.7 Instalando el modulo creado

Login

En la pantalla de *login*, aparecerán los detalles de la conexión y base de datos creada, para poder loguearnos con el superusuario (admin). A continuación podremos definir mas usuarios para nuestro sistema, pero por ahora podemos obviarla haciendo clic en *cancelar*.

Una vez hecho esto, habrá abierta una pestaña con los módulos disponibles para instalar.



The screenshot shows a web application interface with a sidebar on the left containing a link to 'Administración'. The main area is titled 'Módulos' and contains a table of modules. The table has columns for 'Nombre' (Name), 'Versión' (Version), 'Estado' (Status), and 'Acciones' (Actions). The 'country' module is highlighted in blue. The 'training' module is marked for installation. The 'webdav' module is also listed.

Nombre	Versión	Estado	Acciones
country	3.8.0	No instalado	Marcar p...
ir		Instalado	Marcar p...
party	3.8.1	No instalado	Marcar p...
res		Instalado	Marcar p...
training	3.8.0	No instalado	Marcar p...
training2	3.8.0	No instalado	Marcar p...
webdav		No instalado	Marcar p...

Haciendo clic en el botón marcar para instalar de training (o como sea que hayamos nombrado a nuestro modulo), se marcaran también las dependencias que serán necesarias instalar.

Para concluir con la instalación, se hace clic en el botón de *Ejecutar acción* (el rombito jaspeado entre los botones *Añadir un adjunto al registro* y *Abrir registros relacionados*), donde se dará inicio a la instalación de los módulos marcados

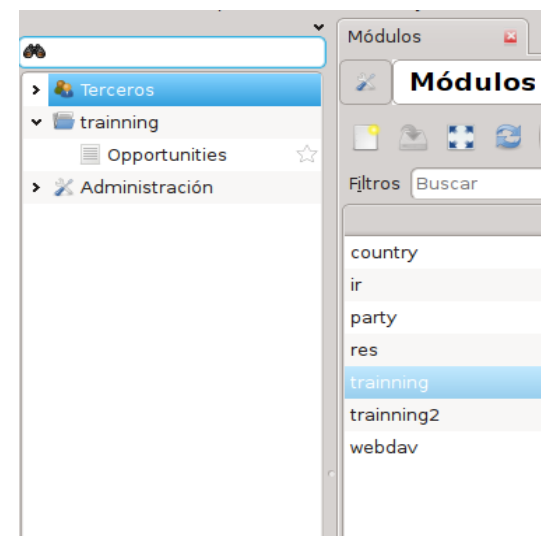
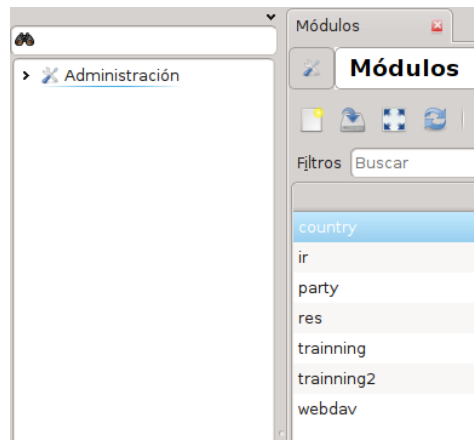
2.7 Instalando el modulo creado

Finalización

La instalación de los módulos dentro del sistema puede llevar algún tiempo, sobretodo cuando se trabaja con importación de datos predefinidos, como es el caso del modulo *country*. Es recomendable ir leyendo el *terminal* de manera de ir sabiendo que es lo que va ocurriendo con la instalación e irnos familiarizando los mensajes de la misma. Desde el cliente solo se nos informara si se ha tenido éxito o no con la instalación.

Si algún modulo instalado requiriese de alguna configuración mas, aparecerá a continuación un asistente para la misma. Sino, aparecerá un mensaje de que se realizo la configuración con éxito.

Otra vez, si todo sale bien, no habrá mensajes de errores. Si los hubiera, copiar la traza y estudiarla para poder determinar que es lo que salio mal.



2.7 Instalando el modulo creado

Nota final

Si por algún raro motivo, no nos apareciese nuestro modulo *trainning* en el registro de módulos a instalar, o bien una vez instalado no se pueda acceder al mismo, deberemos parar el servidor y ejecutar:

```
$ trytond -c RUTA_AL_TRYTOND_CONF/trytond.conf -d NUESTRA_BD -u NUESTRO_MODULO -v
```

Con esto estamos pidiendo a *trytond*,



- c que utilizando la configuración definda en trytond.conf
- d en la base de datos NUESTRA_BD (o cualquiera sea el nombre definido para la misma)
- u actualice el modulo NUESTRO_MODULO (*trainning* para nuestro caso)
- v en modo *verbose*, (mostrándonos en pantalla el proceso)





¿ Para dónde vamos ?





Repetir el encuentro para seguir avanzando en/con/para el aprendizaje/enseñanza



¿Qué proponemos?





Que practiquen lo aprendido para el próximo
encuentro

Que vengan (de venir, no de vengarse)

Que traigan sus dudas (serán muchas!)

Que no se queden con las ganas!!!

Que vean los canales y la documentación





Fuentes de consulta!!!!!!!

www.tryton.org Sitio oficial de tryton, donde puede descargarse los códigos fuentes y principales enlaces a otras fuentes

doc.tryton.org/3.8 documentación oficial de la versión 3.8

www.gnusolidario.org Sitio oficial del proyecto gnu solidario

<https://groups.google.com/forum/#!forum/tryton> Grupo oficial de tryton. También esta el grupo español y el argentino.

moyanocasco.franciscom@gmail.com el mas lerdo para responder.





Mas Fuentes de consulta!!!!!!!

<http://www.python.org.ar/> Comunidad Python argentina. Mucha documentación sobre el lenguaje, lista de correo, noticias de eventos, foro de consultas.

www.linux-malaga.org/index.php?s=file_download&id=9 *Python en 14* páginas. Guía de referencia rápida de python.

<https://www.python.org/dev/peps/pep-0020/> *PEP 20*. El Zen de python. **!!!Lectura obligatoria!!!**

<https://www.python.org/dev/peps/pep-0008/> *PEP 8* ("Pepocho"). Guía de estilo para codificar en Python.

Luego de leer PEP 20.





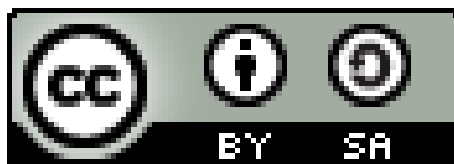
Muchas gracias por venir!

Cátedra de Salud Pública – Facultad de Ingeniería (UNER)

E-mail: saludpublica@bioingenieria.edu.ar

Facebook: www.facebook.com/catedraSaludPublica





Basado en el trabajo *Tryton: Capacitación técnica. Una introducción al desarrollo en Tryton* de los autores: Mario Puntin, Adrián Bernardi <contacto@silix.com.ar>; Versión:0.2; Fecha:Mayo de 2015

Todo el contenido esta licenciado bajo Creative Commons Attribution-ShareAlike 4.0 International License.

Las capturas de pantalla a la interfaz de Tryton fueron tomadas desde mi computadora personal

