

## UD06. DESARROLLO DE MÓDULOS. CONCEPTOS GENERALES

**Sistemas de Gestión Empresarial**

**2 Curso // CFGS DAM // Informática y Comunicaciones**

**Alfredo Oltra**

**Cicles  
Formatius**

## ÍNDIX

|                                   |          |
|-----------------------------------|----------|
| <b>1 INTRODUCCIÓN</b>             | <b>4</b> |
| <b>2 LA BASE DE DATOS</b>         | <b>7</b> |
| <b>3 COMPOSICIÓN DE UN MÓDULO</b> | <b>8</b> |
| 3.1 Composición del módulo        | 8        |
| <b>4 BIBLIOGRAFIA</b>             | <b>9</b> |
| <b>5 AUTORES</b>                  | <b>9</b> |

Versión: 231204.1442


## Licencia




**Reconocimiento – NoComercial – CompartirIgual (by-nc-sa).** No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

## Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 **Atención.** Importante prestar atención a esta información.

 **Interesante.** Ofrece información sobre algún detalle a tener en cuenta.

## 1 INTRODUCCIÓN

*Odoo* tiene una arquitectura cliente-servidor de 3 capas:

- La base de datos en un servidor *PostgreSQL*.
- El servidor *Odoo*, que engloba la lógica de negocio y el servidor web.
- La capa de cliente que es una *SPA (Single Page Application)*. La capa cliente está subdividida en al menos 3 interfaces muy diferenciadas:
  - El *Backend* donde se administra la base de datos por parte de los administradores y empleados de la empresa.
  - El *Frontend* o página web, donde pueden acceder los clientes y empleados. Puede incluir una tienda y otras aplicaciones.
  - El *TPV* para los terminales punto de venta que pueden ser táctiles.



Además de usar su propio cliente, *Odoo* admite que otras aplicaciones interactúen con su servidor usando XML-RPC. También se pueden desarrollar *web controllers* para crear una API para aplicaciones web o móviles, por ejemplo..

Aparte de la arquitectura de 3 capas, *Odoo* es un sistema modular. Esto quiere decir que se puede ampliar con módulos de terceros (oficiales o no) y módulos desarrollados por nosotros mismos.

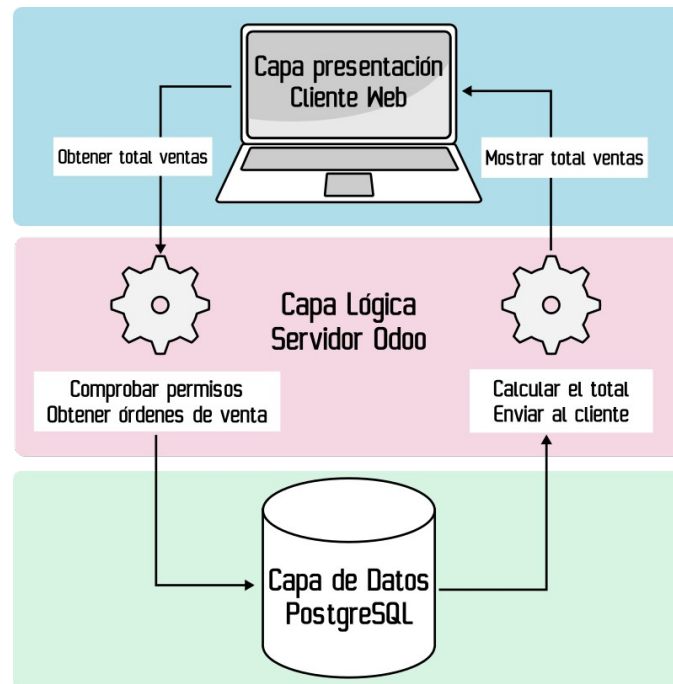
De hecho, en *Odoo* hay un módulo (llamado módulo *base*) que contiene el funcionamiento básico del servidor y a partir de ahí se cargan todos los demás módulos.

Cuando instalamos *Odoo*, antes de instalar ningún módulo, tenemos acceso al *backend* donde gestionar poco más que las opciones y usuarios. Es necesario instalar los módulos necesarios para un funcionamiento mínimo. Por ejemplo, lo más típico es instalar al menos los módulos de ventas, compras, CRM y contabilidad.

Para ampliar las funcionalidades o adaptar *Odoo* a las necesidades de una empresa, no hay que modificar el código fuente de *Odoo*. Tan solo necesitamos crear un módulo.

Los módulos de *Odoo* pueden modificar el comportamiento del programa, la estructura de la base de datos y/o la interfaz de usuario. En principio, un módulo se puede instalar y desinstalar y los cambios que implicaba el módulo se revierten completamente.

*Odoo* facilita el desarrollo de módulos porque, además de un ERP, es un framework de programación. *Odoo* tiene su propio framework tipo RAD (*Rapid Application Development*). Esto significa que con poco esfuerzo se pueden conseguir aplicaciones con altas prestaciones y seguras.



El poco esfuerzo es relativo. Para desarrollar correctamente en *Odoo* son necesarios amplios conocimientos de *Python*, *XML*, *HTML*, *Javascript* y otras tecnologías asociadas como *QWeb*, *JQuery*, *XML-RPC*, etc. La curva de aprendizaje es alta y la documentación es escasa. Además, los errores son más difíciles de interpretar al no saber todo lo que está pasando por debajo. La frustración inicial se verá compensada con una mayor agilidad y menos errores.

Este framework se basa en algunos de los principios generales de los RAD modernos:

- La capa **ORM (Object Relational Mapping)** entre los objetos y la base de datos.
  - La combinación *Clase Python* ↔ *ORM* ↔ *Tabla PostgreSQL* se conoce como *Modelo*.
  - El programador no efectúa el diseño de la base de datos, solo de las clases y sus relaciones.
  - Tampoco es necesario hacer consultas SQL, casi todo se puede hacer con los métodos de ORM de Odoo.
- La arquitectura **MVC** (Modelo-Vista-Controlador).
  - El **modelo** se programa declarando *clases* de *Python* que heredan de *models.Model*. Esta herencia provoca que actúe el *ORM* y se *mapean* en la *base de datos*.

- Las **vistas** se **definen** normalmente en **archivos XML** y son listas, formularios, calendarios, gráficos, menús, etc. Este **XML** será enviado al cliente web donde el **framework Javascript** de **Odoo** lo **transforma** en **HTML**.
- El **controlador** también se define en ficheros **Python**, normalmente **junto al modelo**. El controlador son los **métodos** que proporcionan la **lógica** de **negocio** (no confundir con los *web controllers* que hemos comentado anteriormente)
- **Odoo** tiene una arquitectura de **Tenencia Múltiple**, de manera que un único servidor puede proporcionar servicio a muchos clientes de bases de datos diferentes.
- Las versiones de pago proporciona un diseñador de informes y de interfaces.
- El framework facilita la traducción de la aplicación a muchos idiomas.



El puesto que *Odoo* facilita la traducción, es una buena práctica programar todo en inglés, tanto el nombre de las variables como los textos que mostramos a los usuarios. Posteriormente, podemos añadir las traducciones necesarias.

## 2 LA BASE DE DATOS

Gracias al ORM, no hay un diseño definido de la base de datos. La base de datos de una empresa puede tener algunas tablas muy diferentes a otras en función del mapeado que el *ORM* haya hecho con las clases activas en esa empresa. Por tanto, es difícil encontrar un diseño entidad-relación o algo similar en la documentación de *Odoo*.



Eso no significa que no exista, simplemente que es muy dependiente de los módulos instalados

*Odoo* tiene algunos modelos ya creados y bien documentados como:

- *res.partner* (clientes, proveedores, etc.).
- *sale.order* (orden de venta).

Estos modelos existen de base debido a que están en casi todas las empresas y versiones de *Odoo*. Pero ni siquiera estos tienen en la base de datos las mismas columnas o relaciones que en otras empresas. Muchas veces necesitamos saber el nombre del modelo, del campo o de la tabla en la base de datos. Para ello, *Odoo* proporciona en su *backend* el modo desarrollador para saber el modelo y campo poniendo el ratón encima de un campo de los formularios.



El nombre de las clases de *Python* se realiza utilizando nomenclatura CamelCase. Sin embargo, el nombre de un modelo será siempre: *modulo.modelo*. Si el modelo tiene un nombre compuesto, se separa por `_`. En la base de datos, al nombrar el modelo el punto se sustituye por una barra baja. Por ejemplo, *sale.order* hacer referencia al modelo order del módulo sale y en la base de datos estará relacionado con la tabla *sale\_order*



Aconsejamos dedicar unos minutos a conocer la base de datos usando el modo desarrollador y el cliente de terminal de *PostgreSQL*. Para ello, podemos repasar las consultas SQL sacando, por ejemplo, el nombre de los clientes que no han hecho ningún pedido.

### 3 COMPOSICIÓN DE UN MÓDULO

*Odoo* es un programa modular. Tanto el servidor como el cliente se componen de módulos que extienden al módulo base. Cualquier cosa que se quiera modificar en *Odoo* se ha de hacer creando un módulo.



Puesto que *Odoo* es de código abierto y todo el código está en Python, que no es un lenguaje compilado, podemos alterar los ficheros *Python* o *XML* de los módulos oficiales, cambiando lo que nos interese. Esto puede funcionar, pero **es una mala práctica**, ya que:

- Cualquier actualización de los módulos oficiales borraría nuestros cambios.
- Si no actualizamos, perderemos acceso a nueva funcionalidades y estaremos expuestos a problemas de seguridad.
- Revertir cambios es más difícil y la solución suele pasar por volver a la versión oficial.

Podemos crear módulos para modificar, eliminar o ampliar partes de otros módulos. También podemos crear módulos para añadir funcionalidades completamente nuevas a *Odoo* sin interferir con el resto del programa. En cualquier caso, el sistema modular está diseñado para que se puedan instalar y desinstalar módulos sin afectar al resto del programa.

Por ejemplo: puede que una empresa no necesite todos los datos que pide *Odoo* al registrar un producto. Como solución, podemos hacer un módulo que elimine de la vista los campos innecesarios. Si luego se comprueba que esos campos eran necesarios, solo hay que desinstalar el módulo y vuelven a aparecer.

Este sistema modular funciona porque, cada vez que se reinicia el servidor o se actualiza un módulo, se interpretan los ficheros *Python* que definen los modelos y el ORM mapea las novedades en la base de datos. Además, se cargan los datos de los ficheros *XML* en la base de datos y se actualizan los datos que han cambiado.

#### 3.1 Composición del módulo

Los módulos modifican partes de Modelo-Vista-Controlador. De esta manera, un módulo se compone de ficheros Python, XML, CSS o Javascript entre otros. Todos estos archivos deben estar en una carpeta con el nombre del módulo.

Hay una estructura de subcarpetas y de nombres de archivos que casi todos los módulos respetan. Pero todo depende de lo que ponga en el fichero `__manifest__.py`. Este fichero contiene un diccionario de *Python* con información del módulo y la ubicación de los demás ficheros. Además, el archivo `__init__.py` indica qué ficheros *Python* se han de importar.



Dentro de un módulo podemos encontrar:

- Ficheros *Python* que definen los modelos y los controladores.
- Ficheros XML que definen datos que deben ir a la base de datos. Dentro de estos datos, podemos encontrar:
  - Definición de las vistas y las acciones.
  - Datos de demo.
  - Datos estáticos que necesita el módulo.
- Ficheros estáticos como imágenes, CSS, Javascript, etc. que deben ser cargados por la interfaz web.
- Controladores web para gestionar las peticiones web.

Los módulos se guardan en un directorio indicado en la opción `--addons-path` al lanzar el servidor o en el fichero de configuración `odoo.conf`. Los módulos pueden estar en más de un directorio y dependen del tipo de instalación o la distribución o versión que se instale.

Para crear un módulo se puede hacer manualmente creando la carpeta, el *manifest*, los directorios y ficheros o utilizando una herramienta de línea de comandos llamada *odoo scaffold*.

```
odoo scaffold nombremodulo /rutadondecolocarlo
```

Una vez ejecutado este comando, tenemos en la ruta indicada, la estructura básica de directorios y ficheros con un poco de código de ejemplo.

## 4 BIBLIOGRAFIA

1. [Documentación de Odoo](#)

## 5 AUTORES

A continuación ofrecemos en orden alfabético (por apellido) el listado de autores que han hecho aportaciones a este documento.

- Jose Castillo Aliaga
- Sergi García Barea
- Alfredo Oltra