# TEMA 5 - Entorno de desarrollo y primer módulo de Odoo

## ÍNDICE DE CONTENIDOS

1. Introducción
2. Creación de un directorio para módulos
3. Entorno de desarrollo
4. Activación del modo “desarrollador” en Odoo 18
5. Nuestro primer módulo: “Hola mundo”
6. Creación de módulos en Odoo
7. Ejemplo de módulo “Lista de tareas”
8. Depuración de Odoo dentro de un contenedor con Visual Studio Code

## 1. INTRODUCCIÓN

En unidades anteriores se ha visto cómo instalar Odoo, tanto en un entorno pensado para **producción** como en un entorno pensado para **desarrollo**. A partir de este punto se da un paso más: **no solo utilizar Odoo, sino comenzar a crear módulos propios**.

En esta unidad se pone el foco en la correcta preparación del **entorno de desarrollo**, que será la base para todo el trabajo posterior. Para ello se utilizará:

- Odoo ejecutándose mediante **Docker**.
- **Visual Studio Code** como editor de código, junto con sus extensiones.

Una vez configurado el entorno, se comenzará con el desarrollo de los **primeros módulos de Odoo**, siguiendo un enfoque progresivo y guiado.

### 1.1 ¿Cómo orientaremos esta unidad?

El primer objetivo de la unidad es comprobar que el **entorno de desarrollo funciona correctamente**. Con Odoo bien configurado y ejecutándose en Docker, es posible desarrollar módulos sin necesidad de una infraestructura compleja. Para comenzar, basta con disponer de:

- Una terminal.
- Un editor de texto.

Aun así, para trabajar de forma más cómoda y evitar errores habituales, se recomienda realizar una configuración previa tanto de Odoo como de las herramientas de desarrollo asociadas. En esta unidad se presta especial atención al uso de **Visual Studio Code en Windows 11**, por tratarse del entorno de trabajo principal utilizado.

Una vez comprobado que el entorno funciona correctamente, se introduce el desarrollo de módulos. El primer módulo que se crea es **muy sencillo**: no añade funcionalidad, pero puede instalarse sin errores. Este primer paso es importante porque permite confirmar que:

- Odoo detecta correctamente los módulos creados.
- El directorio de módulos está bien configurado.
- El entorno de desarrollo es correcto.

A partir de este módulo inicial, la funcionalidad se irá **ampliando de forma progresiva**. No se pretende crear un módulo completo de forma inmediata, sino comprender el proceso paso a paso: primero un módulo muy simple y, posteriormente, módulos cada vez más completos.

El desarrollo de módulos en Odoo puede llegar a ser complejo. Por este motivo, en esta unidad no se profundiza en todos los detalles técnicos, sino que se busca **sentar las bases** y comprender cómo funciona el desarrollo de módulos. En la siguiente unidad se abordará la programación de módulos con mayor profundidad.

# 2. CREACIÓN DE UN DIRECTORIO PARA MÓDULOS

## 2.1 ¿Qué es un directorio de módulos en Odoo?

Odoo es un sistema ERP **modular**. Esto significa que su funcionalidad no está concentrada en un único bloque de código, sino distribuida en **módulos** independientes que se pueden instalar, desinstalar o actualizar según las necesidades.

Un **módulo de Odoo** es una carpeta que contiene una estructura de ficheros concreta (Python, XML, CSV, etc.) y que amplía o modifica el comportamiento del sistema: nuevos modelos de datos, vistas, menús, reglas de acceso, informes, etc.

Para que Odoo pueda **localizar, leer e instalar estos módulos**, es necesario indicar uno o varios **directorios de módulos** (*addons paths*). Estos directorios actúan como "repositorios" donde Odoo busca módulos disponibles.

En resumen:

- El **directorio de módulos** es la carpeta donde se guardan los módulos de Odoo.
- Odoo escanea ese directorio al arrancar o al actualizar la lista de aplicaciones.
- Solo los módulos que estén dentro de un directorio configurado serán detectados por Odoo.

## 2.2 Directorios de módulos estándar y personalizados

Odoo dispone de dos tipos principales de directorios de módulos:

1. **Módulos estándar (oficiales)**

   Son los que vienen incluidos con Odoo (Ventas, Compras, Contabilidad, CRM, etc.).

   Forman parte del código base del sistema y **no deben modificarse** durante el desarrollo.

2. **Módulos personalizados (custom addons)**

   Son los módulos desarrollados por terceros o por el propio equipo de desarrollo.

   Estos módulos se almacenan en un **directorio separado**, distinto del código base de Odoo.

En entornos de desarrollo y formación profesional, **siempre se trabaja con módulos personalizados**, ya que:

- Evita modificar el núcleo del sistema.
- Facilita el mantenimiento y la actualización.
- Permite borrar o recrear el entorno sin perder el trabajo.

## 2.3 El directorio de módulos en Odoo con Docker

En esta unidad se trabaja con **Odoo desplegado mediante Docker y Docker Compose**, por lo que la configuración del directorio de módulos se realiza a través de **volúmenes**.

Un **volumen Docker** permite conectar una carpeta del sistema anfitrión (Windows 11) con una ruta interna del contenedor. De esta forma:

- Los ficheros se editan desde Windows.
- Odoo los utiliza desde el contenedor.
- No es necesario copiar archivos manualmente dentro del contenedor.

En la configuración utilizada en el curso, el directorio de módulos personalizados se define en el fichero `docker-compose.yml`. En el servicio correspondiente a Odoo aparece el siguiente volumen:

```
volumes:
  - ./extra-addons:/mnt/extra-addons
```

Esto indica que:

- La carpeta **`extra-addons`**, situada en el mismo directorio que el fichero `docker-compose.yml`, es el **directorio de módulos en el equipo anfitrión** (Windows 11).
- Esa misma carpeta se monta dentro del contenedor de Odoo en la ruta **`/mnt/extra-addons`**.

De esta forma, Docker conecta una carpeta del sistema operativo del anfitrión con una ruta interna del contenedor. Odoo tiene configurada esta ruta (`/mnt/extra-addons`) como parte de su **ruta de búsqueda de módulos**, por lo que todos los módulos almacenados en dicha carpeta serán detectados por el sistema.

## 2.4 ¿Para qué sirve este directorio en la práctica?

Durante esta unidad, el directorio de módulos servirá para:

- Crear el primer módulo **"Hola mundo"**.
- Generar módulos con **Odoo scaffold**.
- Editar ficheros Python y XML desde Visual Studio Code.
- Probar cambios sin reinstalar Odoo.
- Depurar módulos dentro del contenedor.

Todas las carpetas de módulos que se creen (por ejemplo `Ejemplo01-HolaMundo`, `lista_tareas`, etc.) **deben estar dentro de este directorio**.

Si un módulo se crea fuera de este directorio:

- Odoo no lo detectará.
- No aparecerá en la lista de aplicaciones.
- No se podrá instalar.

## 2.5 Recomendaciones

- Trabajar **exclusivamente** dentro del directorio de módulos configurado.
- No modificar ficheros del código base de Odoo.
- Usar Visual Studio Code como editor principal.
- Mantener una estructura clara de carpetas y nombres de módulos.
- Verificar siempre que el módulo aparece tras **Actualizar la lista de aplicaciones**.

Este enfoque reproduce una forma de trabajo muy similar a la que se utiliza en entornos profesionales, pero adaptada a un contexto educativo y controlado.

# 3. ENTORNO DE DESARROLLO

Para desarrollar módulos en Odoo es necesario disponer de un **entorno de desarrollo adecuado**, que facilite la edición del código, la organización de los ficheros y la detección de errores. Existen distintas herramientas que permiten realizar este trabajo, siendo las más utilizadas **Visual Studio Code** y **PyCharm**.

Además del editor o IDE, en el desarrollo de módulos resulta muy recomendable utilizar **herramientas adicionales**, como un sistema de control de versiones. En este sentido, el uso de **Git**, junto con plataformas como **GitHub** o **GitLab**, permite gestionar los cambios en el código, mantener un historial de versiones y trabajar de forma más ordenada.

En esta unidad se prioriza el uso de **Visual Studio Code**, por su versatilidad, su amplia adopción en entornos profesionales y su buena integración con Odoo y Docker.

## 3.1 Visual Studio Code

**Visual Studio Code** es un editor de código ligero y potente que permite trabajar con distintos lenguajes y tecnologías. Destaca por su interfaz clara y por su amplio ecosistema de extensiones, que permiten adaptar el editor a diferentes tipos de desarrollo.

### Extensiones recomendadas para trabajar con Odoo

Para facilitar el desarrollo de módulos en Odoo, se recomienda instalar las siguientes extensiones en Visual Studio Code:

- **Python**

  Proporciona soporte para el lenguaje Python, incluyendo resaltado de sintaxis, detección de errores y ayuda a la escritura de código.

  https://marketplace.visualstudio.com/items?itemName=ms-python.python

- **Odoo Snippets**

  Añade fragmentos de código reutilizables que agilizan la creación de ficheros habituales en módulos de Odoo.

https://marketplace.visualstudio.com/items?itemName=jeffery9.odoo-snippets

Estas extensiones permiten trabajar de forma más cómoda y estructurada, reduciendo errores comunes y mejorando la legibilidad del código durante el desarrollo de módulos.

## 4. ACTIVACIÓN DEL MODO "DESARROLLADOR" EN ODOO 18

Odoo dispone de un **modo desarrollador** que activa opciones avanzadas del sistema. Este modo es imprescindible cuando se trabaja en el desarrollo de módulos, ya que permite acceder a información técnica y a herramientas que no están visibles en el modo normal de uso.

Entre otras cosas, el modo desarrollador permite:

- Acceder a menús técnicos.
- Visualizar identificadores internos de vistas, campos y acciones.
- Analizar y depurar el comportamiento de los módulos durante su desarrollo.

En la documentación oficial de Odoo se describen las distintas formas de activar este modo:

https://www.odoo.com/documentation/17.0/es/applications/general/developer_mode.html

### Formas de activar el modo desarrollador

Existen varias opciones para activar el modo desarrollador en Odoo, todas ellas válidas:

- **Desde la propia interfaz de Odoo**, a través del menú de configuración.
- **Mediante un parámetro en la URL**, añadiendo una opción específica.
- **Utilizando una extensión del navegador**, que permite activar y desactivar el modo desarrollador de forma rápida.

Cuando se trabaja de manera continuada en el desarrollo de módulos, la opción más cómoda suele ser el uso de una **extensión del navegador**, ya que evita repetir el proceso cada vez que se accede al sistema.

### Extensiones de navegador disponibles

Existen extensiones específicas para los navegadores más utilizados:

- **Firefox**:
  https://addons.mozilla.org/es/firefox/addon/odoo-debug/
- **Chrome**:
  https://chrome.google.com/webstore/detail/odoo-debug/hmdmhilocobgohohpdpolmibjklfgkbi?hl=es_PR

Estas extensiones permiten activar o desactivar el modo desarrollador con un solo clic.

**Atención**: para poder activar el modo desarrollador desde la interfaz de Odoo, es necesario que exista al menos **un módulo instalado** en el sistema (por ejemplo, el módulo de Ventas).

## 5. NUESTRO PRIMER MÓDULO: "HOLA MUNDO"

Un módulo de Odoo es el mecanismo mediante el cual se amplían o modifican las funcionalidades de este sistema ERP. Cada módulo puede añadir nuevos modelos de datos, vistas, menús o comportamientos al sistema.

Para comenzar a trabajar con el desarrollo de módulos, se creará el **módulo más sencillo posible**, conocido habitualmente como "Hola mundo". Este módulo no añade ninguna funcionalidad real al sistema, pero cumple un papel fundamental dentro del proceso de aprendizaje.

El objetivo de este primer módulo es **comprobar que el entorno de desarrollo está correctamente configurado** y que Odoo es capaz de detectar e instalar módulos personalizados.

## 5.1 Creación de la estructura del módulo

Para este primer ejemplo se creará una carpeta llamada `Ejemplo01-HolaMundo` dentro del directorio configurado para almacenar los módulos personalizados.

Esta carpeta representará el módulo y contendrá únicamente los ficheros mínimos necesarios para que Odoo lo reconozca como tal.

## 5.2 Ficheros mínimos de un módulo

Todo módulo de Odoo debe contener, al menos, dos ficheros:

### a) Fichero `__init__.py`

Este fichero indica que la carpeta debe ser tratada como un paquete de Python.

En este primer ejemplo, el fichero **debe estar vacío**, ya que el módulo no contiene todavía lógica de programación.

### b) Fichero `__manifest__.py`

Este fichero es el **manifiesto del módulo**. En él se incluye la información básica que Odoo necesita para identificar el módulo y mostrarlo en la lista de aplicaciones.

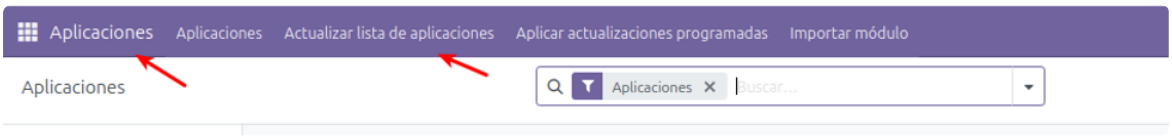
El contenido del fichero será el siguiente:

```
# -*- coding: utf-8 -*-
{'name': 'Ejemplo01-Hola mundo'}
```

En este caso, únicamente se indica el nombre del módulo. En módulos más avanzados, este fichero incluirá información adicional como dependencias, versión, vistas o permisos.

## 5.3 Comprobación del módulo en Odoo

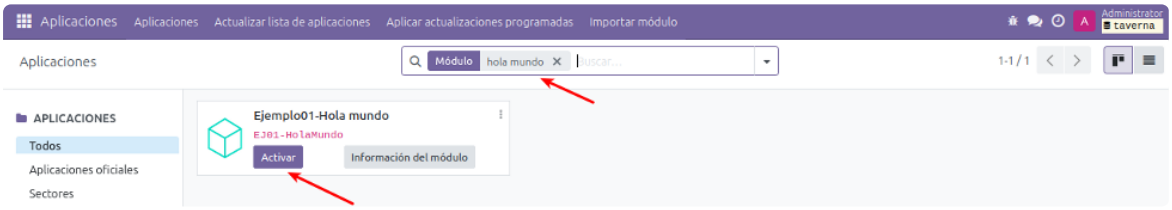
Una vez creada la estructura del módulo, y con el **modo desarrollador activado**, se puede acceder a la lista de módulos de Odoo y pulsar la opción **"Actualizar la lista de aplicaciones"**.




Tras actualizar la lista:

- Es necesario **eliminar los filtros de búsqueda por defecto**.
- En especial, se debe desactivar el filtro **"Aplicaciones"**.
- A continuación, se puede buscar el módulo por su nombre, en este caso **"Hola mundo"**.

Si todo está correctamente configurado, el módulo aparecerá en la lista de módulos disponibles.




## 5.4 Instalación del módulo

Una vez localizado el módulo, se puede proceder a su instalación. Aunque este módulo no realiza ninguna acción, su instalación confirma que:

- El directorio de módulos está bien configurado.
- Odoo detecta correctamente los módulos creados.
- El entorno de desarrollo funciona de forma correcta.

Este primer módulo servirá como base para los siguientes ejemplos, en los que se irá incorporando funcionalidad de manera progresiva.

# 6. CREACIÓN DE MÓDULOS EN ODOO

Los módulos son la forma en la que Odoo amplía y organiza sus funcionalidades. Cada módulo puede considerarse una pequeña **aplicación independiente**, que se integra dentro del sistema Odoo y añade nuevas características al conjunto.

Un módulo puede cumplir funciones muy diversas: crear nuevos modelos de datos, añadir vistas, incorporar menús, modificar comportamientos existentes o ampliar funcionalidades ya disponibles.

En esta unidad se comenzará con un tipo de módulo sencillo y fácil de entender: un módulo que crea **nuevos modelos de datos** y permite visualizarlos a través de **nuevos menús**. Este enfoque facilita la comprensión de la estructura básica de un módulo antes de abordar desarrollos más complejos.

> **Nota**: además de crear módulos nuevos, Odoo permite modificar modelos y vistas existentes para ampliar funcionalidades. Este tipo de desarrollo se estudiará en unidades posteriores mediante el uso de herencia.

## 6.1 Creación de módulos con "Odoo Scaffold"

Odoo proporciona una herramienta denominada **"Odoo Scaffold"**, cuyo objetivo es **generar automáticamente la estructura base de un módulo**. Esta herramienta crea las carpetas y ficheros necesarios para comenzar a trabajar, evitando errores habituales al crear la estructura manualmente.

Antes de utilizar esta herramienta, es necesario comprobar que el entorno de trabajo está correctamente configurado y que Odoo se está ejecutando sin problemas.

### Ejecución de Odoo Scaffold en entornos Docker

Cuando Odoo se ha desplegado mediante Docker, el comando **"Odoo scaffold"** debe ejecutarse **dentro del contenedor de Odoo**, ya que es ahí donde se encuentra el entorno de ejecución.

Para acceder al contenedor se puede utilizar un comando similar al siguiente:

```
docker exec -it IDCONTENEDOR /bin/bash
```

Si se ha utilizado Docker Compose, el comando equivalente sería:

```
docker-compose exec web /bin/bash
```

### Creación del módulo

Una vez dentro del contenedor, se puede crear el módulo ejecutando el comando **"Odoo scaffold"**, indicando el nombre del módulo y el directorio donde se almacenará:

```
odoo scaffold lista_tareas /mnt/extra-addons/
```

Este comando genera automáticamente una carpeta llamada `lista_tareas` dentro del directorio de módulos personalizados.

> **Atención**: el nombre del módulo no debe comenzar por un número ni contener el carácter "-", ya que Odoo no acepta esos formatos.

### Ajuste de permisos en entorno educativo

Cuando se trabaja con Odoo dentro de un contenedor Docker, puede ser necesario ajustar los permisos de la carpeta del módulo para facilitar la edición de los ficheros desde el sistema anfitrión.

Este ajuste se realiza desde el interior del contenedor con el siguiente comando:

```
chmod 777 -R /mnt/extra-addons/lista_tareas
```

Esta medida se utiliza **únicamente en entorno educativo**, con el fin de evitar problemas de permisos durante el desarrollo.

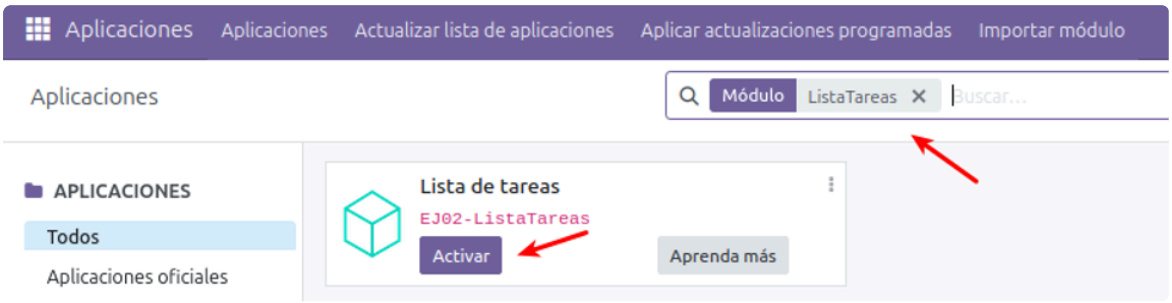
### Comprobación del módulo en Odoo

Una vez creado el módulo:

- Se accede a la lista de módulos de Odoo.
- Se actualiza la lista de aplicaciones.

- Se eliminan los filtros de búsqueda.
- Se busca el módulo por su nombre, en este caso **"lista_tareas"**.

Si todo está correctamente configurado, el módulo aparecerá disponible para su instalación.




### Alternativa al uso de Odoo Scaffold

Si no se desea utilizar el comando **"Odoo scaffold"**, es posible descargar manualmente la estructura base de un módulo desde la documentación oficial de Odoo:

https://www.odoo.com/documentation/17.0/_downloads/b7f3a4243ae7f3166cd5c4d23a256739/my_module.zip

La estructura descargada contiene código de ejemplo que, por defecto, aparece comentado. Para utilizar dicho código, será necesario descomentar el contenido de los ficheros correspondientes.

## 6.2 Estructura de un módulo Odoo

Al igual que ocurre en otros *frameworks*, Odoo utiliza una **estructura de directorios y ficheros bien definida** para organizar los módulos. Esta estructura no es arbitraria: Odoo la utiliza para localizar el código, interpretar los datos y cargar correctamente cada parte del módulo.

Comprender esta estructura es fundamental, ya que permite identificar rápidamente **dónde se define cada elemento** de un módulo y facilita tanto el desarrollo como el mantenimiento del código.

### Ficheros básicos de un módulo

Todo módulo de Odoo comienza con dos ficheros fundamentales:

- `__manifest__.py`

  Es el fichero más importante del módulo. Contiene la información necesaria para que Odoo reconozca el módulo, conozca su nombre, sus dependencias y los ficheros que deben cargarse.

  La información se define utilizando un **diccionario de Python**.

- `__init__.py`

  Indica que el directorio del módulo debe tratarse como un paquete de Python. En este fichero se importan los submódulos o directorios que contienen la lógica del módulo.

Estos dos ficheros son imprescindibles para que Odoo pueda detectar e instalar el módulo.

### Subdirectorios del módulo

Cuando se utiliza la herramienta **"Odoo scaffold"**, se crea una estructura base que incluye varios subdirectorios, cada uno con una función concreta. Los directorios que contienen código Python incluyen, a su vez, su propio fichero `__init__.py`.

La estructura generada es la siguiente:

- `controllers/`

  Contiene los controladores del módulo, que permiten definir rutas web y gestionar peticiones HTTP.

  - `__init__.py`
  - `controllers.py`

- `demo/`

  Incluye datos de demostración que pueden cargarse al instalar el módulo, pensados para pruebas o ejemplos.

  - `demo.xml`

- `models/`

  Contiene la definición de los modelos de datos del módulo, es decir, las tablas y campos que se crean en la base de datos.

  - `__init__.py`

    - `models.py`
- `security/`

  Define los permisos de acceso a los modelos del módulo.
    - `ir.model.access.csv`
- `views/`

  Contiene las vistas XML que definen cómo se muestran los datos en la interfaz de Odoo.
    - `templates.xml`
    - `views.xml`
- `__init__.py`
- `__manifest__.py`

### Función de los ficheros principales

A continuación se describe brevemente la función de los ficheros más relevantes del módulo:

- **`models/models.py`**

  Define los modelos de datos del módulo y los campos que los componen.
- **`views/views.xml`**

  Describe las vistas asociadas al módulo, como formularios, vistas de lista (*tree*), menús y acciones.
- **`demo/demo.xml`**

  Incluye datos de ejemplo que permiten probar el módulo una vez instalado.
- **`controllers/controllers.py`**

  Contiene ejemplos de controladores que permiten definir rutas web y gestionar peticiones externas.
- **`views/templates.xml`**

  Incluye ejemplos de vistas *qweb*, normalmente utilizadas junto con los controladores.
- **`__manifest__.py`**

Es el manifiesto del módulo. Además de información general, indica qué ficheros deben cargarse al instalar o actualizar el módulo.

En los módulos creados con scaffold, es necesario descomentar la línea que hace referencia al fichero de permisos `security/ir.model.access.csv` para que el módulo funcione correctamente.

Cada parte del módulo tiene una ubicación concreta dentro de la estructura.

Saber dónde colocar cada fichero es tan importante como el contenido del propio código.

## 6.3 Módulos en producción

En un **entorno de producción**, los módulos de Odoo siguen un comportamiento muy concreto que es importante comprender para evitar confusiones durante el desarrollo.

Cuando se **instala** o **actualiza** un módulo, Odoo utiliza la información indicada en el fichero `__manifest__.py` para saber qué ficheros debe cargar y cómo debe actualizar la base de datos. Este proceso es el único momento en el que Odoo lee y aplica los cambios definidos en los ficheros de datos del módulo.

Odoo funciona siguiendo un modelo denominado *data-driven* (dirigido por datos). Esto significa que, al instalar un módulo, la información contenida en los ficheros XML (vistas, menús, acciones, datos, etc.) **se almacena en la base de datos**.

Por este motivo:

- Los cambios realizados en ficheros **XML**
  - **no se reflejan automáticamente** en la aplicación.
  - Solo se aplican cuando el módulo se **instala o se actualiza**.
  - Reiniciar el servicio Odoo **no es suficiente** para ver estos cambios.

En cambio, los ficheros **Python** del módulo se cargan de nuevo cada vez que se inicia el servicio Odoo. Por ello:

- Los cambios en código Python se pueden aplicar:
  - reiniciando el servicio Odoo,
  - o actualizando el módulo si el cambio lo requiere.

### Reinicio del servicio en entornos Docker

Cuando Odoo se ejecuta dentro de un contenedor Docker, reiniciar el servicio equivale a **reiniciar el contenedor**.

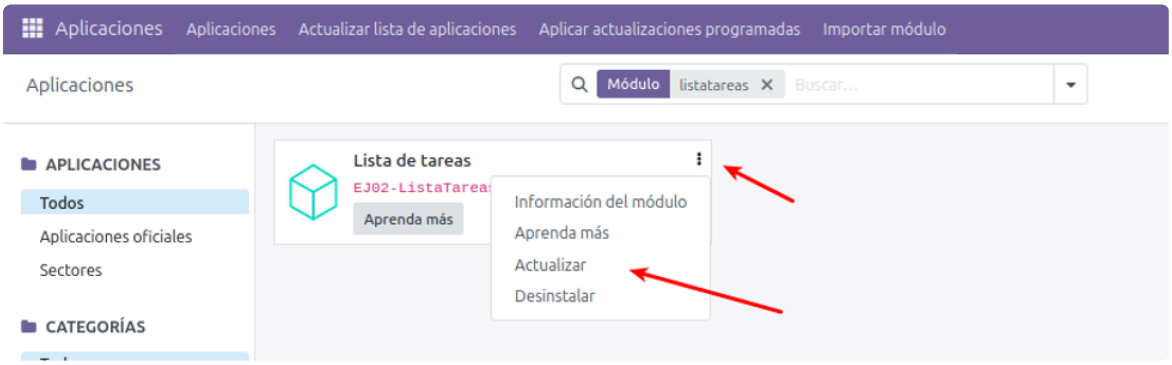
Esto puede hacerse con el siguiente comando:

```
docker restart IDCONTENEDOR
```

Si se está utilizando Docker Compose, el comando equivalente sería:

```
docker-compose restart web
```

Este reinicio recarga el código Python, pero **no actualiza automáticamente las vistas ni los datos definidos en XML**.




> En producción, los cambios en XML requieren **actualizar el módulo**.
>
> Reiniciar Odoo solo recarga el código Python.

## 6.4 Módulos en desarrollo

Durante el desarrollo de módulos, Odoo ofrece un **modo de desarrollo** que facilita el trabajo y permite ver los cambios de forma inmediata.

Cuando Odoo se arranca con la opción `--dev=all`, el servidor:

- Lee las vistas, los datos y el código Python **directamente desde los ficheros**.
- No utiliza únicamente la información almacenada en la base de datos.
- Permite ver cambios en tiempo real sin necesidad de actualizar el módulo ni reiniciar el servicio.

Gracias a este modo, es posible modificar:

- vistas,
- datos,
- o código Python,

y observar los cambios de forma inmediata.

### Uso del modo desarrollo

El modo desarrollo resulta muy cómodo para trabajar durante la fase de creación y prueba de módulos. Sin embargo, **no debe utilizarse en entornos de producción**, ya que:

- reduce el rendimiento,
- y puede suponer riesgos de seguridad.

Por este motivo, este modo se reserva exclusivamente para entornos de desarrollo.

> El modo desarrollo facilita la creación de módulos,
>
> pero solo debe usarse durante la fase de desarrollo.

## 7. EJEMPLO DE MÓDULO “LISTA DE TAREAS”

Hasta ahora hemos creado un primer módulo muy sencillo (“Hola mundo”) cuyo único objetivo es comprobar que:

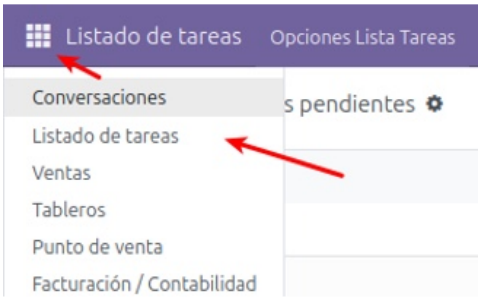
- El directorio de módulos está correctamente configurado.
- Odoo detecta módulos personalizados.
- Se puede instalar un módulo sin errores.

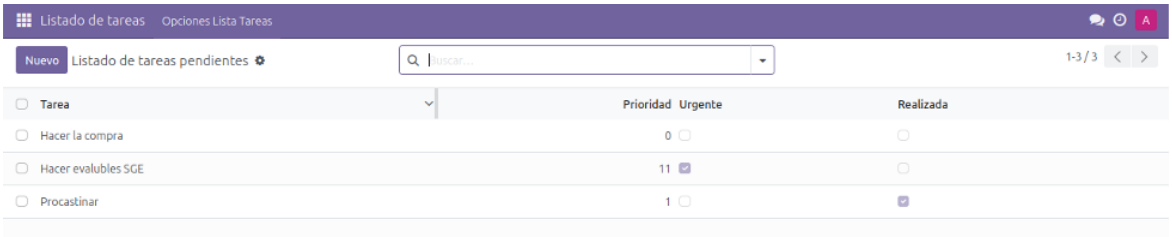
Este primer módulo **no añade funcionalidad**, pero es un paso imprescindible para validar el entorno de desarrollo.

## 1. Objetivo del módulo

El módulo **"Lista de tareas"** permitirá:

- Crear tareas.
- Asignar una prioridad numérica a cada tarea.
- Marcar una tarea como realizada o no.
- Calcular automáticamente si una tarea es urgente.




El módulo añadirá:

- Un nuevo modelo de datos.
- Una vista de lista.
- Un menú propio dentro de Odoo.

## 2. Creación del módulo con Odoo Scaffold

Como estamos trabajando con Odoo en Docker, la creación del módulo se realiza **dentro del contenedor**.

### 2.1 Acceso al contenedor

Desde una terminal:

```
docker exec -it odoo18 /bin/bash
```

### 2.2 Creación de la estructura base

Una vez dentro del contenedor, ejecutamos:

```
odoo scaffold lista_tareas /mnt/extra-addons
```

Si el comando se ejecuta correctamente, se creará una carpeta llamada `lista_tareas` dentro del directorio de módulos personalizados.

### 2.3 Permisos (entorno educativo)

Para poder editar los archivos cómodamente desde Windows con Visual Studio Code, se asignan permisos completos al módulo:

```
chmod 777 -R /mnt/extra-addons/lista_tareas
```

## 3. Estructura del módulo

El módulo creado contiene, entre otros, los siguientes elementos:

- `__manifest__.py` → información del módulo.
- `models/` → definición del modelo de datos.
- `views/` → vistas XML.
- `security/` → permisos de acceso.

A partir de ahora trabajaremos principalmente con:

- `__manifest__.py`
- `models.py`
- `views.xml`

## 4. Configuración del manifiesto (`__manifest__.py`)

El fichero `__manifest__.py` describe el módulo y le indica a Odoo qué debe cargar.

```
# -*- coding: utf-8 -*-
{
    'name': "Lista de tareas",

    'summary': """
    Sencilla Lista de tareas""",

    'description': """
    Sencilla lista de tareas utilizadas para crear un nuevo módu
    modelo de datos
    """,

    'author': "Sergi García",
    'website': "https://apuntesfpinformatica.es",

    # Indicamos que es una aplicación
    'application': True,

    # En la siguiente URL se indica qué categorías pueden usarse
    # https://github.com/odoo/odoo/blob/17.0/odoo/addons/base/da

    # Vamos a utilizar la categoría Productivity
    'category': 'Productivity',
    'version': '0.1',

    # Indicamos lista de módulos necesarios para que este funcio
    # En este ejemplo solo depende del módulo "base"
    'depends': ['base'],

    # Esto siempre se carga
    'data': [
        # Este primero indica la política de acceso del módulo
        'security/ir.model.access.csv',
        # Cargamos las vistas y las plantillas
        'views/views.xml',
    ]
}
```

- `name`: nombre del módulo.
- `application`: indica que aparece como aplicación.
- `depends`: módulos necesarios para funcionar.

- `data`: ficheros XML y CSV que se cargarán al instalar el módulo.

### 5. Definición del modelo de datos (`models.py`)

En este módulo se define un **nuevo modelo de datos** que representará las tareas.

```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api

# Definimos el modelo de datos
class lista_tareas(models.Model):
    # Nombre y descripción del modelo de datos
    _name = 'lista_tareas.lista_tareas'
    _description = 'lista_tareas.lista_tareas'

    # Elementos de cada fila del modelo de datos
    # Los tipos de datos a usar en el ORM son:
    # https://www.odoo.com/documentation/17.0/developer/referenc

    tarea = fields.Char()
    prioridad = fields.Integer()
    urgente = fields.Boolean(compute="_value_urgente", store=Tru
    realizada = fields.Boolean()

    # Este cómputo depende de la variable prioridad
    @api.depends('prioridad')
    # Función para calcular el valor de urgente
    def _value_urgente(self):
        # Para cada registro
        for record in self:
            # Si la prioridad es mayor que 10, se considera urge
            if record.prioridad > 10:
                record.urgente = True
            else:
                record.urgente = False
```

- `tarea`: descripción de la tarea.
- `prioridad`: número entero.
- `realizada`: indica si la tarea está hecha.

- `urgente`: campo calculado:
  - se marca como `True` si la prioridad es mayor que 10.
  - no se edita manualmente.

El método `_value_urgente` se ejecuta automáticamente cuando cambia el campo `prioridad`.

### 6. Creación de las vistas (`views.xml`)

Las vistas definen cómo se muestran los datos en la interfaz web de Odoo.

```
<odoo>
    <data>

        <!-- explicit list view definition -->
        <!-- Definimos cómo es la vista explícita de la lista
        <record model="ir.ui.view" id="lista_tareas.list">
            <field name="name">lista_tareas list</field>
            <field name="model">lista_tareas.lista_tareas</fie
            <field name="arch" type="xml">
                <tree>
                    <field name="tarea"/>
                    <field name="prioridad"/>
                    <field name="urgente"/>
                    <field name="realizada"/>
                </tree>
            </field>
        </record>

        <!-- actions opening views on models -->
        <!-- Acciones al abrir las vistas en los modelos -->
        <!-- https://www.odoo.com/documentation/17.0/developer
        <record model="ir.actions.act_window" id="lista_tareas
            <field name="name">Listado de tareas pendientes</f
            <field name="res_model">lista_tareas.lista_tareas<
            <field name="view_mode">tree,form</field>
        </record>

        <!-- Top menu item -->
        <menuitem name="Listado de tareas" id="lista_tareas.me

        <!-- menu categories -->
        <menuitem name="Opciones Lista Tareas"
                  id="lista_tareas.menu_1"
                  parent="lista_tareas.menu_root"/>

        <!-- actions -->
        <menuitem name="Mostrar lista"
                  id="lista_tareas.menu_1_list"
                  parent="lista_tareas.menu_1"
                  action="lista_tareas.action_window"/>

    </data>

```

</odoo>

La vista de lista muestra:

- todas las tareas,
- con sus campos principales,
- en formato de lista.

Las acciones al abrir las vistas en los modelos son:

- La **acción** abre la vista del modelo.
- El **menú principal** aparece en Odoo.
- El **submenú** permite acceder al listado de tareas.

### 7. Instalación del módulo

Una vez guardados todos los archivos:

1. Entrar en Odoo.
2. Activar **modo desarrollador**.
3. Ir a **Aplicaciones**.
4. Pulsar **Actualizar lista de aplicaciones**.
5. Buscar **"Lista de tareas"**.
6. Instalar el módulo.

Si todo está correcto:

- aparecerá el menú,
- se podrán crear tareas,
- y el campo "urgente" se calculará automáticamente.

Un módulo de Odoo se construye combinando:

- un modelo (Python),
- una vista (XML),
- y un menú.

Cada parte cumple una función distinta, pero todas son necesarias.

# 8. DEPURACIÓN DE ODOO DENTRO DE UN CONTENEDOR CON VISUAL STUDIO CODE

En este apartado se introduce la **depuración de módulos de Odoo** utilizando **Visual Studio Code** y **Docker**. La depuración permite ejecutar el código paso a paso y observar qué ocurre internamente cuando Odoo procesa un módulo.

Esta técnica es **muy útil en proyectos grandes**, pero en una primera unidad **no es imprescindible dominarla**. Por ese motivo, este apartado se considera **contenido avanzado** y su uso es **opcional**.

## 8.1 ¿Qué es depurar un módulo?

Depurar un módulo significa:

- detener la ejecución del programa en un punto concreto,
- observar el valor de las variables,
- y entender cómo se ejecuta el código línea a línea.

En Odoo, la depuración se realiza sobre el **código Python de los módulos**, y resulta especialmente útil cuando:

- un módulo no se comporta como se espera,
- aparecen errores difíciles de localizar,
- o se quiere comprender mejor cómo funciona internamente.

## 8.2 Herramientas necesarias

Para poder depurar Odoo dentro de un contenedor Docker se aprovechan las funcionalidades de **desarrollo remoto** de Visual Studio Code.

Es necesario tener instaladas las siguientes extensiones:

- **Extensión de Odoo para Visual Studio Code**
  https://marketplace.visualstudio.com/items?itemName=Odoo.odoo
- **Extensión Dev Containers** (desarrollo con contenedores Docker)

https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-containers

Además, será necesario disponer de la extensión **Python**, que Visual Studio Code suele sugerir automáticamente.

## 8.3 Acceso al contenedor desde Visual Studio Code

El acceso al contenedor de Odoo se realiza mediante el icono **"Remote Development"** de Visual Studio Code.

Al pulsar este icono:

- se mostrará un panel lateral,
- en el que aparecerán los contenedores Docker disponibles,
- y los volúmenes asociados.

Desde este panel se puede **adjuntar (attach)** Visual Studio Code al contenedor de Odoo.

Al hacerlo:

- se abrirá una **nueva ventana de Visual Studio Code**,
- que estará conectada directamente al contenedor,
- y desde la que se trabajará durante la depuración.

## 8.4 Pasos generales para depurar Odoo

De forma resumida, el proceso de depuración consiste en:

1. **Adjuntar Visual Studio Code al contenedor de Odoo** mediante Remote Development.
2. Instalar la extensión de **Python** si el entorno lo solicita.
3. Abrir una carpeta desde el explorador (**Open Folder**).
4. Seleccionar la carpeta `/mnt/extra-addons`, donde se encuentran los módulos personalizados.
5. Acceder al apartado **"Run & Debug"**.

6. Ejecutar la configuración de Python denominada **"Odoo attach debug"**.

A partir de ese momento, es posible:

- colocar puntos de ruptura (*breakpoints*),
- ejecutar el código paso a paso,
- y analizar el comportamiento del módulo.

## 8.5 Importante para esta unidad

- **No es obligatorio** configurar ni utilizar la depuración en esta unidad.
- **No se evaluará** el uso de la depuración.
- El objetivo principal sigue siendo:
  - crear módulos,
  - definir modelos,
  - crear vistas y menús,
  - e instalar correctamente un módulo en Odoo.

Si la depuración no funciona en tu equipo o resulta complicada, **no es un problema**. Se trata de una herramienta avanzada que se retomará más adelante, cuando se tenga mayor experiencia con Odoo.

## 8.6 Para saber más

Quien quiera profundizar en este tema puede consultar la documentación oficial de Visual Studio Code sobre desarrollo con contenedores:

https://code.visualstudio.com/docs/devcontainers/containers

# 9. BIBLIOGRAFÍA

- **Sistemas de Gestión Empresarial IOC**:

https://ioc.xtec.cat/materials/FP/Materials/2252_DAM/DAM_2252_M10/web/html/index.html

- **Wikipedia**:
  https://es.wikipedia.org/wiki/Sistema_de_planificación_de_recursos_empresariales
- **Documentación de Odoo**:
  https://www.odoo.com/documentation/master/reference/http.html