

Sistemas de Gestión Empresarial

# **UD 02. Instalación y configuración de un ERP**

---

## 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.

## ÍNDICE DE CONTENIDO

1.	Introducción	3
1.1	Contexto histórico de los ERP-CRM	3
1.2	Sistemas ERP-CRM en la propia empresa	3
1.3	Sistemas ERP-CRM en la nube	4
1.4	¿Qué elegir? ¿Sistema en la propia empresa o en la nube?	4
1.5	¿Y el software para nuestro sistema ERP?	5
2.	Tipos de instalación de un sistema ERP	6
3.	Licencias de software	7
4.	Preparación del servicio para instalar el sistema ERP	7
4.1.	Cálculo del hardware.	9
5.	Instalación de un sistema ERP Odoo 17	9
5.1.	Requisitos de Odoo 17	10
5.2.	Odoo 17 en Ubuntu Server - Parte 1: Instalación	10
5.3.	Odoo 17 en Ubuntu Server - Parte 2: Preparando Odoo para desarrollo	11
5.4.	Odoo 17 en Ubuntu Server - Parte 3: Arrancando Odoo	12
5.5.	Odoo 17 en Ubuntu Server - Parte 4: Errores típicos	14
5.6.	Odoo 17 - ¿Qué propuestas recomendáis para desarrollo?	15
5.7.	Odoo 17 en Docker: - Parte 1: Contenedor Odoo en producción	16
5.8.	Odoo 17 en Docker: - Parte 2: Contenedor Odoo para desarrollo	17
5.9.	Odoo 17 en Docker: - Parte 3: Docker Compose para Odoo	18
6.	Puesta en marcha de Odoo 17	20

## UD02. INSTALACIÓN Y CONFIGURACIÓN DE UN ERP

### 1. INTRODUCCIÓN

En esta unidad veremos qué factores generales debemos tener en cuenta para la instalación de un sistema ERP tras ello veremos distintas formas de como instalar “Odo”, el software que utilizaremos durante este curso.

#### 1.1 Contexto histórico de los ERP-CRM

Las posibilidades para la instalación de un sistema ERP-CRM se han multiplicado en los últimos años. Las tendencias, posibilidades técnicas y precios del hardware, software y la electricidad han ido modificando la manera en la que las empresas utilizan los ordenadores personales y los servidores.

Haciendo un breve e inexacto repaso histórico podemos reflexionar sobre los motivos técnicos y económicos de esta evolución:

- En los primeros años de la informática empresarial el Mainframe era casi la única opción: un único ordenador muy grande y caro con varias terminales. Este Mainframe era instalado en el edificio de la empresa. Esta solución solo estaba al alcance de grandes empresas.
- La llegada de los computadores personales y miniordenadores hizo que muchas empresas más pequeñas utilizaran esa solución.
  - En ocasiones era un único ordenador pequeño gestionado por la empresa.
  - Mientras tanto, los mainframes seguían ocupando el espacio de las grandes empresas.
  - En sus inicios, el PC y similares tenían carencias tanto a nivel de potencia y como de sistemas operativos orientados al mercado doméstico y no al mercado profesional. Aun así, su aterrizaje en la vida cotidiana influenció a la sociedad y a la industria, llevando que fabricaran servidores con la arquitectura de los PC y similares pero con una mayor fiabilidad requerida en el contexto industrial.
    - Esto es una gran mejora que permanece hasta hoy en día. Los servidores con esta arquitectura mejoraron considerablemente el hardware con procesadores y memorias más estables y sistemas operativos libres y propietarios más robustos (Unix, Linux, Windows NT/Server...).
- La llegada de Internet supuso otra alternativa. Ahora las empresas podían usar una red común para interconectar sus sedes, tanto con mainframes como con servidores PC, se podía tener un programa centralizado.
  - Al mismo tiempo, algunas empresas pudieron ofrecer servicios por Internet a otras empresas. Este fue el inicio de lo que llamamos “servicios en la nube”.

#### 1.2 Sistemas ERP-CRM en la propia empresa

La opción de tener un servidor en la propia empresa tiene una serie de ventajas e inconvenientes (relacionados entre sí) y cuyos principales problemas vienen dados tanto por los costes económicos como por la seguridad de los datos.

Tener un servidor en la propia empresa con un sistema ERP-CRM (o con cualquier otro tipo de servicio) supone algunos retos:

- Poner en marcha un servicio requiere una inversión inicial en hardware.
- El escalado de hardware para aumentar potencia/disminuir potencia es problemático:
  - Habitualmente se tiene hardware infrautilizado.
  - El aumento de potencia de un servicio requiere escalado vertical (aumento de prestaciones del servidor) o escalado horizontal (comprar más equipos).
    - El hardware limita las posibilidades de cómputo, así que es imposible escalar para aumentar potencia en momentos puntuales.
- El mantenimiento del sistema informático, el suministro de energía y la seguridad del sistema son gastos y responsabilidades asociados a la empresa.
  - Una ventaja de la gestión de datos interna es que no están en ordenadores fuera de la empresa, evitando riesgos tales como el espionaje industrial.

### 1.3 Sistemas ERP-CRM en la nube

El gasto tanto energético como de potencia de computación en una sola empresa suele ser desigual durante el día, suponiendo esto una ineficiencia energética. Por este motivo, algunas empresas, especialmente aquellas que ofrecen servicios por Internet, se plantearon vender esa potencia cuando no la necesitaban o compartirla entre muchos clientes. Esto propició el nacimiento de lo que hoy llamamos “los servicios en la nube”.

Con la llegada de los “servicios en la nube” (que realmente son servidores de otra empresa) las empresas pagan una cuota (fija, por tiempo de cómputo, por uso, etc.), pero a cambio:

- No realizan gastos relacionados con el hardware (instalación y escalado).
- No realizan gastos relacionados con el consumo eléctrico.
- Se reducen de forma drástica los gastos en mantenimiento y seguridad.
- Facilitan el acceso: para operar con estos sistemas la empresa solo necesitan un dispositivo (ordenador personal, smartphone, etc.) con conexión a Internet.
- Si se necesita más potencia, solo hay que contratarla (escalado vertical/horizontal).

Aun así, el uso de “servicios en la nube” poseen varios inconvenientes:

- En algunos contextos, puede resultar más caro que poner en marcha tu la infraestructura.
- Los datos están almacenados físicamente en un servidor de otra empresa, con posibles problemas relacionados (por ejemplo, espionaje industrial).
- Para el uso del sistema dependen tanto de una buena conexión a Internet como del buen funcionamiento general de la empresa que presta servicios.

### 1.4 ¿Qué elegir? ¿Sistema en la propia empresa o en la nube?

No existe una respuesta “tajante” a esta pregunta, ya que depende del contexto y es algo a estudiar concienzudamente antes de implantar un sistema.

Hoy en día en los entornos empresariales conviven ambas opciones mencionadas, e incluso opciones híbridas (servidor en la empresa, pero apoyo puntual o algunos servicios en la nube).

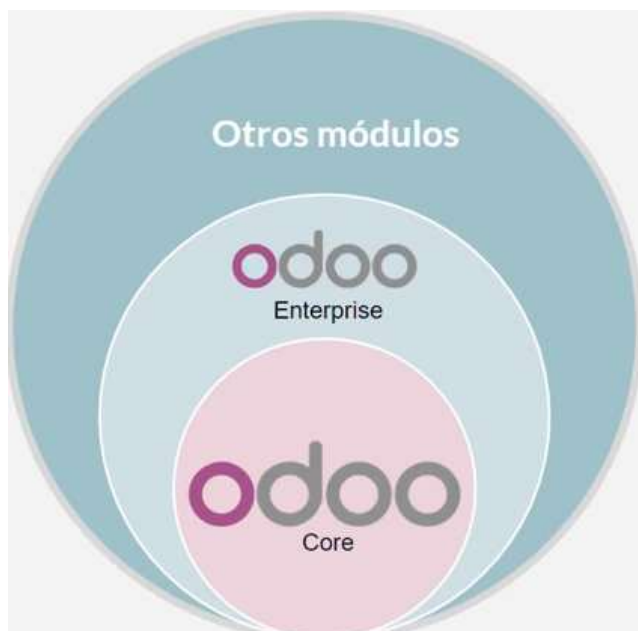
Los principales factores a la hora de tomar esta decisión son:

- El cumplimiento de las leyes de protección de datos. Este punto sobre todo influye en la decisión de contratar o no un “servicio en la nube”, ya que guardando determinados datos en determinados servicios en la nube podríamos estar incumpliendo la ley.
  - En Europa existe el reglamento RGPD y en España la legislación vigente en materia de protección de datos viene definida por la LOPDGDD.
- El precio de la electricidad y el consumo del hardware.
  - Existen dispositivos hardware orientados a tener un bajo consumo.
- El precio del hardware.
  - Existen dispositivos orientados a tareas de servidor muy baratos.

### 1.5 ¿Y el software para nuestro sistema ERP?

Durante la introducción hemos hablado de costes relacionados con el sistema que alojaría nuestro ERP. Pero... ¿Y el software? Existen multitud de opciones software ERP, tanto gratuitos, de pago, libres, mixtos (partes libres, partes de pago), etc.

Existe un software ERP llamado Odoo, que se presenta en dos versiones: “Community Edition” (software libre y gratuito) y “Enterprise Edition” (de pago).



En este curso utilizaremos **Odoo “Community edition”**, ya que es libre y gratuito. Por simplicidad, cuando nos refiramos a Odoo, estaremos refiriéndonos a esta versión.

Más información en [https://www.odoo.com/es\\_ES/](https://www.odoo.com/es_ES/) y <https://github.com/odoo/odoo>.

## 2. TIPOS DE INSTALACIÓN DE UN SISTEMA ERP

En este apartado vamos a tratar las distintas maneras en la que se puede instalar un sistema ERP y a mencionar los pros y contras de cada tipo de solución.

Comenzaremos con las soluciones no recomendadas (“cutres”), para pasar aquellas simples pero correctas e ir evolucionando hacia las más sofisticadas:

- **La carpeta compartida:** se trata de una práctica cada vez menos usada pero que se mantiene en muchas pequeñas empresas.
  - Hace años, algunos programas de gestión estaban pensados para un solo usuario. La única opción que daban para poder ser accedidos por varios ordenadores en una red local era compartir la carpeta de la base de datos y configurar varias instalaciones para acceder al mismo archivo. Además, esa “base de datos” era un fichero de “Access” o ficheros de texto plano.
  - **Esta solución no es correcta ni recomendable:** es una solución que propone muchos problemas, tanto con el acceso simultáneo, como problemas tanto de seguridad como de integridad.
- **Instalación On-Premise:** aquí entra en juego un servidor instalado en la propia empresa.
  - Los ordenadores cliente pueden acceder al sistema ERP mediante un software cliente del propio ERP o incluso mediante un navegador web.
    - Se debe configurar correctamente la red del sistema tanto para evitar conexiones externas no permitidas, como para proporcionar acceso seguro (si se requiere) desde fuera de la red.
    - Si se necesita acceder desde fuera de la red local, hay que configurar correctamente la red y la seguridad del acceso externo. Se suele pagar por el software entero o la instalación y el mantenimiento.
  - Si el servidor funciona de una forma segura y esta opción es viable económicamente para la empresa, es una solución correcta.
- **Instalación como servicio en la nube:** en esta solución prescinde de servidor en la empresa y se subcontrata la computación. Simplifica la instalación y el acceso externo. Además, se paga por lo que se necesita y es fácilmente escalable. Los “servicios en la nube” puede ser de muchos tipos, pero se suele distinguir entre:
  - **IAAS (Infraestructura como servicio):** nos proporcionan acceso a servidores, máquinas virtuales o contenedores. Hay que instalar y securizar el sistema operativo, el ERP y todo lo demás. Puede que la empresa de la nube ya nos lo ofrezca con un sistema operativo preinstalado, pero deberíamos tener control total de ese sistema operativo.
    - **Ejemplo:** los VPS entran dentro de este tipo de nube.
  - **PAAS (Plataforma como servicio):** En esta nube ya está el sistema operativo y algunos programas configurados. Sobre este sistema se puede desplegar el sistema ERP.
    - **Ejemplo:** lo que consideramos como “hosting tradicional” (PHP, MySQL, etc.).
  - **SAAS (Software como servicio):** en este caso ya está el ERP instalado y nos dan acceso a determinadas características según contratemos (cantidad de usuarios, acceso simultáneo, espacio de almacenamiento, copia de seguridad, etc.). No hay que preocuparse de nada más, pero estás limitado al programa que el proveedor ofrece.
    - **Ejemplo:** soluciones tipo Gmail o Google Drive, donde tenemos todo puesto en marcha y solo consumimos el servicio.

- Aparte de estos tipos de “servicios en la nube”, podemos tener algunos servicios de forma individual en la nube, como por ejemplo servicios de bases de datos (como Firebase) o servicios de APIs REST o GraphQL, etc. En cualquier caso, deben garantizar una alta disponibilidad, seguridad y escalabilidad.

### 3. LICENCIAS DE SOFTWARE

A la hora de elegir el software de un sistema ERP, uno de los factores es la licencia y su precio. Vamos a distinguir entre licencias libres y licencias propietarias.

Se considera licencia libre aquella que permita la modificación y redistribución del software. Cualquier licencia que no permita la modificación y redistribución del software se considera licencia propietaria.

⚠ **Atención:** Una licencia libre no implica que el software sea gratuito. Una licencia propietaria no implica que el software sea de pago.

El modelo de negocio del software propietario es fácil de intuir a primera vista (creo un producto y lo vendo, cobro servicios asociados, etc.). El modelo de negocio del software libre es muy diverso, desde el pago por el mantenimiento, cursos, instalación o personalización del sistema, donaciones o venta de productos adicionales.

Existe una gran cantidad de licencias libres. En el siguiente enlace existe una comparativa de las principales [https://en.wikipedia.org/wiki/Comparison\\_of\\_free\\_and\\_open-source\\_software\\_licences](https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licences)

Una de las primeras licencias libres y la más garantista es GPL y sus distintas versiones. Otras de las licencias libres más conocidas son la MIT, BSD o Apache.

### 4. PREPARACIÓN DEL SERVICIO PARA INSTALAR EL SISTEMA ERP

Cada sistema ERP es distinto y posee necesidades de potencia distintas. Estas necesidades suelen depender de factores como:

- Cantidad de datos que alberga el sistema.
- Número de usuarios simultáneos.
- Tecnología utilizada por el software ERP.
- Sistema operativo sobre el que corre el software ERP.

Para poner en marcha un sistema ERP es recomendable utilizar hardware diseñado para servidores. La calidad de los componentes, la refrigeración o tecnologías como el ECC en la memoria RAM hacen más estables estos ordenadores frente a PC domésticos u otras alternativas para el

usuario final (Raspberry Pi, Arduino, etc.).

Los usuarios que interactúan con el sistema ERP, sí pueden usar PCs, tablets, smartphones u otros sistemas domésticos como cliente del servicio.

Otro aspecto a tener en cuenta es la seguridad, teniendo en cuenta tanto medidas de seguridad pasiva como activa. Deben tenerse en cuenta aspectos como:

- Protección contra personas (ataques físicos y remotos).
- Protección contra los elementos: calor, humedad, sobretensiones o incendios.
- Disponibilidad del sistema y copia de seguridad.
- Cumplimiento de la legislación vigente.

No es tarea de este módulo explorar todas las protecciones disponibles, pero deben tenerse en cuenta y exigirlas en una instalación real. Algunas de las medidas de protección básicas son:

- El servidor debe tener un SAI que lo proteja y lo mantenga encendido en caso de fallo de la red eléctrica.
- Es recomendable que el servidor en sí disponga de elementos (procesador, RAM, discos, etc.) por encima de la potencia mínima necesaria para que funcione el sistema ERP.
  - No es necesario pasarse demasiado, ya que si lo hacemos incrementaremos tanto el gastos de hardware y el consumo eléctrico sin obtener un gran beneficio. Normalmente estos programas tienen una documentación en la que describen los requerimientos mínimos.
- Respecto a la seguridad de los datos, se recomienda redundancia en los discos, ya sea con RAID o con sistemas de archivos redundantes como ZFS o BTRFS.
  - Esta redundancia no excluye la necesidad de establecer una política de copias de seguridad externas al sistema.

Una vez puesto en marcha el servidor hay que elegir el sistema operativo base y un posible sistema de virtualización. En el caso del sistema ERP Odoo, que tratamos en estos apuntes, para una puesta en marcha en un sistema real se recomienda Ubuntu Server.

Este sistema operativo puede ser el sistema instalado en la máquina o estar virtualizado.

La virtualización podemos realizarla con:

- Máquinas virtuales con hipervisor (tipo Virtualbox). Estas máquinas realizan una simulación completa del hardware, influyendo esto en una disminución de rendimiento.
- Una alternativa a la virtualización con hipervisor, pero con mejor rendimiento (cercano al rendimiento nativo) son los contenedores, ya sea completos como LXD o contenedores de aplicaciones como Docker.

Sistemas operativos base como Proxmox simplifican la gestión tanto de máquinas virtuales como de contenedores, copias de seguridad de los mismos y almacenamiento en red.

En caso de optar por la nube, si contratamos un IAAS también debemos tener en cuenta la potencia contratada. De hecho, es más importante afinar correctamente, ya que podemos incrementar los costes sin tener un beneficio en cuanto a rendimiento.



#### 4.1. Cálculo del hardware.

Para el cálculo de los parámetros el número de usuarios y el uso que hacen de el.

Para ello se calculan los worker. Web workers te permiten ejecutar cálculos intensivos de CPU en un subproceso en segundo plano, liberando el hilo principal para actualizar la interfaz de usuario. Si encuentras que la aplicación realiza una gran cantidad de cálculos, como generar dibujos CAD o realizar cálculos geométricos pesados, el uso de web workers puede ayudar a aumentar el rendimiento de la aplicación.

El cálculo del workers se puede calcular de varias maneras, en función del hardware o en función de los usuarios. Los valores máximos serían:

- Regla general:  $(\#CPU * 2) + 1$  (para el cron de odoo).
- 1 worker  $\approx$  6 usuarios concurrentes.

Para el cálculo del tamaño de la memoria consideramos:

- Que el 20% de las solicitudes son serias, mientras que el 80% son más simples.
- Se estima que un worker pesado consume alrededor de 1 GB de RAM una vez que todos los campos calculados y las peticiones SQL están bien diseñadas, etc.
- Se estima que un worker ligero en el mismo escenario consume alrededor de 150 MB de RAM.

La fórmula sería:

$$\text{RAM necesaria} = \#worker * ( (\text{light\_worker\_ratio} * \text{light\_worker\_ram\_estimation}) + (\text{heavy\_worker\_ratio} * \text{heavy\_worker\_ram\_estimation}) )$$

Un ejemplo sería:

- Servidor con 4 CPU, 8 hilos.
- 60 usuarios concurrentes.

Número de worker:

Según los usuarios:

$60 \text{ usuarios} / 6 = 10$  <- Número teórico de workers necesarios.

Según el hardware:

$(4 * 2) + 1 = 9$  <- Número teórico máximo de workers.

Usaremos 8 workers + 1 para el cron. También usaremos un sistema de monitoreo para medir la carga del CPU y verificar que la carga de trabajo se encuentre entre 7 y 7.5.

El cálculo de la RAM sería:

$$\text{RAM} = 9 * ((0.8 * 150) + (0.2 * 1024)) \approx 3 \text{ GB RAM para Odoo.}$$

## 5. INSTALACIÓN DE UN SISTEMA ERP ODOO 17

En este apartado en primer lugar trataremos los requisitos mínimos para instalar el sistema ERP

Odoo en su versión 17. Tras ello, explicaremos cómo realizar una instalación manual en un sistema Ubuntu. Finalmente, explicaremos cómo poner en marcha Odoo 17 mediante contenedores Docker usando Docker y Docker Compose.

### 5.1. Requisitos de Odoo 17

Los requisitos oficiales para instalar Odoo 17 están disponibles en el sitio web de Odoo:

[https://www.odoo.com/documentation/17.0/es/administration/on\\_premise/packages.html](https://www.odoo.com/documentation/17.0/es/administration/on_premise/packages.html)

A efectos prácticos, Odoo 17 no necesita mucha potencia para funcionar. Puede funcionar sin problemas en cualquier ordenador con varios núcleos y al menos 512MB de RAM, aunque con esa configuración, si recibe muchos accesos simultáneos la máquina se puede quedar corta.

Como en todas las aplicaciones que consultan bases de datos, el acceso al disco puede suponer un cuello de botella. Por eso es recomendable utilizar unidades SSD, RAIDs o sistemas de archivos como ZFS o BTRFS con varios discos.

Odoo 17 funciona perfectamente en máquinas virtuales y contenedores. Algunas opciones de configuración pueden ser:

- Sistema operativo: Ubuntu Server e instalación directa de Odoo.
- Sistema operativo: Ubuntu Server, Virtualización con KVM o similar. Las máquinas virtuales tendrán instalado Ubuntu Server.
- Sistema operativo: Ubuntu Server, contenedores con LXD de Ubuntu.
- Sistema operativo: Ubuntu Server, contenedores Docker.
- Sistema operativo: Proxmox, máquinas virtuales o contenedores LXC gestionados por Proxmox.

Como se ve, en todas las ocasiones se eligen máquinas reales y virtuales Ubuntu. Esto es porque Odoo está desarrollado en ese sistema y esto nos ayuda a garantizar que la implantación del sistema funcione correctamente.

### 5.2. Odoo 17 en Ubuntu Server - Parte 1: Instalación

En este manual vamos explicar cómo realizar una instalación de Odoo 17 mediante un paquete “.deb” en un sistema Ubuntu Server.


En caso de instalarlo en España, hay que hacer algunas comprobaciones previas. Algunos contenedores o VPS tienen configurado el idioma en Inglés. Si instalamos, la base de datos PostgreSQL se configurará automáticamente en ASCII extendido y no aceptará ciertas letras españolas. Por eso hay que configurar el idioma del sistema:

```
dpkg-reconfigure locales
```

Ubuntu tiene en sus repositorios oficiales Odoo, pero suele ser una versión antigua. Por ello vamos a configurar el sistema para instalar la versión 17. Para ello hay que ejecutar estos comandos como root o con un usuario “sudoers” usando sudo:

```
apt-get update
apt-get install ca-certificates wget gnupg
wget -O - https://nightly.odoo.com/odoo.key | apt-key add -
echo "deb http://nightly.odoo.com/17.0/nightly/deb/ ." >
/etc/apt/sources.list.d/odoo17.list
apt-get update && apt-get install odoo
```

Mediante los comandos anteriores lo que hemos hecho es instalar el certificado de Odoo para que la máquina confíe en esos repositorios, crear un fichero con la información de los repositorios de Odoo en ***“/etc/apt/sources.list.d/odoo17.list”*** y finalmente instalar Odoo 17.


 **Importante:** la versión instalada se trata de la versión nightly. Es decir, estás instalando una versión que se actualiza cada noche con los últimos cambios de Odoo.


### 5.3. Odoo 17 en Ubuntu Server - Parte 2: Preparando Odoo para desarrollo


El objetivo de este módulo no es tan solo desplegar un sistema ERP, sino aprender a desarrollar para estos sistemas. Por eso, una vez instalado Odoo, vamos a configurarlo para poder desarrollar con él.

En Odoo es diferente el entorno necesario para producción al entorno para desarrollo. Es altamente recomendable que ambos sean sistemas separados.

**¿Por qué separar producción y desarrollo?** El desarrollo de módulos implica probar cosas que pueden corromper datos, evitar que el servidor tenga que pararse y arrancarse, etc.

 **Atención 1:** el desarrollo de módulos implica probar cosas que pueden corromper datos, evitar que el servidor tenga que pararse y arrancarse, etc.

 **Atención 2:** algunas configuraciones utilizadas en entornos de desarrollo, no son adecuadas para sistemas de producción por motivos de rendimiento y/o seguridad.

 **Atención 3:** algunos pasos del proceso natural de desarrollo pueden dejar “residuos” en la base de datos que pueden ser problemáticos. En ocasiones podemos causar daños al sistema ERP de forma que sea más sencillo reinstalar el sistema de cero que intentar reparar el daño.

Algunos aspectos a tener en cuenta según el tipo de servidor:

- **Servidor en producción:** para un servidor en producción tendremos que configurar correctamente el servicio en ***“/etc/odoo/odoo.conf”*** con una buena contraseña maestra y la ruta de los módulos adicionales en caso de haberla.

- Además, se debe configurar correctamente las reglas del firewall y proporcionar acceso por HTTPS mediante un servidor que actúe como proxy (ejemplo Nginx).
- **Servidor de desarrollo:** no hace falta configurar una gran seguridad, pero es interesante tener un directorio fácil de encontrar para los módulos nuevos y una configuración específica para cuando arrancamos el servicio manualmente al actualizar un módulo.

Cuando se instala Odoo mediante un repositorio (como hemos hecho en este documento), se crea automáticamente un usuario del sistema llamado “odoo” que sirve para que por motivos de seguridad y aislamiento, el servicio arranque con ese usuario y no como root.

Es muy importante que los servicios no los arranque root, ya que evitamos problemas de seguridad.

En el caso de Odoo, si arrancamos el servicio como el usuario “odoo”, tendremos los permisos adecuados sobre los ficheros para que todo funcione sin problemas.

En un servidor de desarrollo, es una buena idea que el usuario con el que se trabaje en el servidor de Odoo para el desarrollo sea el usuario “odoo”.

Por eso podemos darle una contraseña y hacer que su shell sea /bin/bash usando los comandos:

```
sudo passwd odoo
sudo usermod -s /bin/bash odoo
```

Esta última parte no se aplica a un servidor de producción, donde no es buena idea que los usuarios que controlan servicios tengan acceso al shell.

Una vez el usuario tiene shell y accedemos a él, es muy probable que su directorio personal sea “/var/lib/odoo”. No es necesario cambiar esto y ahí podemos crear nuestros módulos personales.

Finalmente, hemos de preparar PostgreSQL. En primer lugar, hemos de asegurarnos que el servicio está funcionando (es necesario para que funcione Odoo). Podemos hacerlo con:

```
service postgresql start
```

Una vez tenemos el servicio arrancado, nos pasamos al usuario “postgres” con el comando:

```
su postgres
```

una vez en ese usuario, ejecutamos (suponiendo que el usuario que lanzara el servicio sea “odoo”):

```
createuser odoo
```

Si fuera otro usuario (“root”, “www”, etc.) cambiaríamos “odoo” por el usuario que pondrá en marcha el servicio. Después de esto ya podemos lanzar el servicio de Odoo sin problemas.

#### 5.4. Odoo 17 en Ubuntu Server - Parte 3: Arrancando Odoo

Odoo puede arrancarse de dos formas:

- Manualmente, invocando al comando “odoo”.

- Automáticamente, como servicio del sistema.

### Arranque manual:

Al arrancar Odoo de manera manual, simplemente podemos lanzarlo con el comando:

```
odoo
```

Este comando arrancará el sistema siguiendo alguna configuración específica (fichero *“.odoorc”* dentro del home del usuario que lo ha lanzado) o si no la hay, utilizando la configuración base de Odoo presente en el fichero *“/etc/odoo/odoo.conf”*.

Para preparar Odoo para un sistema de desarrollo, debemos indicarle que los módulos que utilizará estarán tanto en el directorio oficial como en nuestro *“home”*. Podemos hacerlo con un comando similar a:

```
odoo --addons-path="/var/lib/odoo/modules,/usr/lib/python3/dist-packages/odoo/addons"  
--save
```

Este comando tiene la opción para especificar las rutas donde hay módulos y para que guarde estas rutas. Este comando almacenará estas rutas junto a otras configuraciones en el fichero *“.odoorc”* en el directorio personal del usuario *“odoo”* o del usuario que ejecute el comando anterior.

### Arranque automático:

Si no ejecutamos manualmente Odoo, este se ejecutará siempre que se reinicie el sistema de forma automática. Cuando se ejecuta así es un proceso que actúa como demonio y guarda su historial (log) en *“/var/log/odoo”*.

Esta configuración es útil para servicios en producción.

A veces, este sistema es incómodo para depurar, así que en entornos de desarrollo se recomienda parar el servicio oficial y posteriormente arrancarlo con el comando *“odoo”*.

Un ejemplo de parada de servicio y posterior arranque manual:


```
sudo service odoo stop  
odoo
```

Así podremos observar el historial (log) en tiempo real por la terminal y es nos será más sencillo detectar algunos problemas. Además, esta forma de arrancar permite cosas como actualizar un módulo en una empresa al reiniciar, con el comando:

```
odoo -u modulo -d empresa
```

Si además, queremos lanzarlo como si estuviéramos en un entorno de consola Python, podemos usar el comando:

```
odoo shell -u modulo -d empresa
```

 **Interesante:** si quieres saber más sobre parámetros para lanzar “odoo”, puedes consultar en <https://www.odoo.com/documentation/17.0/es/developer/misc/other/cmdline.html>

### 5.5. Odoo 17 en Ubuntu Server - Parte 4: Errores típicos

Al realizar la instalación manual en Ubuntu Server, hay algunos errores que se suelen repetir si nos saltamos algún paso. Estas son las soluciones a los errores más frecuentes:

#### Error 1:

Aparece un error como este:

```
OperationalError: FATAL: no existe el rol «odoo»
```

```
OperationalError: FATAL: role "odoo" does not exist
```

Eso es porque no ha configurado correctamente PostgreSQL. Este error suele ocurrir cuando ya estaba instalado el SGBD con configuraciones no compatibles con el instalador de Odoo.

Para solucionarlo, hay que crear el usuario:

```
su - postgres -c "createuser -s odoo"
```

#### Error 2:

A veces no funciona el servidor cuando instalamos una base de datos en español. Puede que no tengamos bien configurado el UTF-8 en la base de datos.

Una posible solución es cambiar al usuario “postgres”:

```
su postgres
```

acceder a lanzar código “psql”

```
psql
```

Y lanzar este código:

```
postgres=# update pg_database set datallowconn = TRUE where datname = 'template0';
postgres=# \c template0
template0=# update pg_database set datistemplate = FALSE where datname = 'template1';
template0=# drop database template1;
template0=# create database template1 with template = template0 encoding = 'UTF8';
template0=# update pg_database set datistemplate = TRUE where datname = 'template1';
template0=# \c template1
template1=# update pg_database set datallowconn = FALSE where datname = 'template0';
```

#### Error 3:


Puede que nos olvidemos de la contraseña del usuario de una base de datos.

Podemos solucionar entrando en PostgreSQL (como en el anterior error con “su postgres” y “psql” y ejecutando este comando adaptado a nuestro contexto:

```
update res_users set password='test' where login='admin';
```

### 5.6. Odoo 17 - ¿Qué propuestas recomendáis para desarrollo?

Antes de nada, independientemente de la propuesta de desarrollo que uséis, indicar que es **recomendable utilizar “git”** para ir haciendo una copia de seguridad/versionado del proyecto en un servidor Git, tal como GitHub o GitLab

 **Interesante:** si quieres usar correctamente y fácilmente “git”, esta guía puede ayudarte <https://rogerdudler.github.io/git-guide/index.es.html>


En el punto anterior hemos comentado tanto la instalación manual de Odoo en Ubuntu Server como su propuesta de configuración para usarla como entorno de desarrollo.


 **Importante:** durante el curso podréis utilizar cualquiera de estas propuestas de desarrollo.

En este apartado vamos a realizar una segunda propuesta: instalar Odoo en un contenedor de forma que se aísle el servicio, pero se puede utilizar un IDE externo.

En los siguientes puntos explicaremos cómo aplicar esta propuesta usando contenedores “Docker” junto con la herramienta “Docker Compose”. Este apartado no se explica.

### 5.7. Odoo 17 en Docker: - Parte 1: Contenedor Odoo en producción

 **Interesante:** si no conoces como usar “Docker”, puedes serte muy útil revisar este curso con ejemplos prácticos <https://sergarb1.github.io/CursoIntroduccionADocker/>

 **Interesante:** si ya conoces como usar “Docker”, puedes serte muy útil esta “CheatSheet” <https://raw.githubusercontent.com/sergarb1/CursoIntroduccionADocker/main/FuentesCurso/Docker%20CheatSheet%20COMPLETA.pdf>

Para poner en marcha Odoo 17 en modo producción crearemos dos contenedores:


- El primer contenedor contendrá la base de datos PostgreSQL en su versión 10.
- El segundo contenedor contendrá el servidor Odoo.

Creamos el contenedor de PostgreSQL con:

```
docker run -d -v /home/usuario/OdooDesarrollo/dataPG:/var/lib/postgresql/data -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db postgres:10
```

Donde los parámetros indican lo siguiente:

- “-d”: ejecuta el contenedor en segundo plano.
- “-v /home/usuario/OdooDesarrollo/dataPG:/var/lib/postgresql/data”: monta el directorio del contenedor “/var/lib/postgresql/data” (donde se encuentra la información almacenada por PostgreSQL) en el directorio del anfitrión “/home/usuario/OdooDesarrollo/dataPG”.
  - El fin de esto es almacenar la información de la base de datos en la máquina anfitrión.
- “-e POSTGRES\_USER=odoo -e POSTGRES\_PASSWORD=odoo -e POSTGRES\_DB=postgres”: establece dentro del contenedor esas variables de entorno. A efectos prácticos, esas variables le indican que creen en la base de datos un usuario “odoo”, con contraseña “odoo” y que la base de datos a usar se llama “postgres”.
- “--name db”: nombre que le daremos a nuestro contenedor Docker.
- “postgres:10”: indicamos que usaremos la imagen de Docker Hub llamada “postgres” y de entre ellas usaremos la versión 10.
  - Mas información de esta imagen en [https://hub.docker.com/\\_/postgres](https://hub.docker.com/_/postgres)

 **Importante:** si en lugar del parámetro “-d”, utilizamos el parámetro “-t”, ejecutaremos el contenedor en primer plano y veremos en la terminal información del inicio de PostgreSQL u Odoo. Esto es interesante para detectar problemas.

Con el contenedor PostgreSQL ya en marcha, creamos el contenedor con Odoo con:

```
docker run -d -v -p 8069:8069 --name odooprod --user="root" --link db:db odoo:17
```



Donde los parámetros indican lo siguiente:

- **"-d"**: ejecuta el contenedor en segundo plano.
- **"-p 8069:8069"**: mapeamos el puerto 8069 del contenedor (donde accedemos a Odoo) al puerto 8069 de la máquina anfitrión, para poder acceder a Odoo con <http://localhost:8069>
- **"--name odooprod"**: damos el nombre "odooprod" a nuestro contenedor.
- **"--user=root"**: fuerza a que el contenedor se ejecute internamente como "root" y no como el usuario "odoo" que va por defecto en la imagen.
- **"--link db:db"**: enlazamos con una red virtual este contenedor con el contenedor "db".

## 5.8. Odoo 17 en Docker: - Parte 2: Contenedor Odoo para desarrollo

Para lanzar Odoo en un contenedor preparado para desarrollo, creamos también dos contenedores. Creamos el contenedor de PostgreSQL de forma similar a como hicimos en el apartado anterior con:

```
docker run -d -v /home/usuario/OdooDesarrollo/dataPG:/var/lib/postgresql/data -e POSTGRES_USER=odoo -e POSTGRES_PASSWORD=odoo -e POSTGRES_DB=postgres --name db postgres:10
```

Un servidor de producción está pensado para ponerlo en marcha, realizar pocas paradas y mantener el contenido del contenedor. Se suele hacer copias completas del contenido para únicamente restaurar el contenedor en caso de urgencia.

Sin embargo, en entornos de desarrollo, es más habitual "romper cosas". A efectos prácticos, es habitual reiniciar contenedores o incluso borrarlos y construirlos de nuevo.

Para ello, vamos a modificar la forma de crear los contenedores guardando algunas cosas en volúmenes para realizar "persistencia" del servicio Odoo.

Creamos el contenedor de Odoo, con algunas diferencias respecto al anterior:

```
docker run -d -v /home/usuario/OdooDesarrollo/volumesOdoo/addons:/mnt/extra-addons -v /home/usuario/OdooDesarrollo/volumesOdoo/firestore:/var/lib/odoo/filestore -v /home/usuario/OdooDesarrollo/volumesOdoo/sessions:/var/lib/odoo/sessions -p 8069:8069 --name odoodev --user="root" --link db:db -t odoo:17 --dev=all
```

A continuación, comentamos las diferencias:

- **"-v /home/usuario/OdooDesarrollo/addons:/mnt/extra-addons"**: la imagen de Odoo 17 por defecto carga los módulos del directorio del contenedor **"/mnt/extra-addons"**, por lo cual mapeamos ese directorio a nuestro directorio de la máquina anfitrión **"/home/usuario/OdooDesarrollo/addons"**, donde desarrollaremos usando un IDE externo.
- **"-v /home/usuario/OdooDesarrollo/volumesOdoo/firestore:/var/lib/odoo/filestore"**
- **" y "-v /home/usuario/OdooDesarrollo/volumesOdoo/sessions:/var/lib/odoo/sessions"**: como en desarrollo es posible que paremos y montemos muchas veces los contenedores Docker, montamos estos volúmenes para tener persistencia de los directorios de Odoo **"firestore"** y la **"sessions"**. Para ello, mapeamos esos directorios del contenedor a nuestra máquina anfitrión dentro del directorio **"/home/usuario/OdooDesarrollo/volumesOdoo/"**.

- “**--dev=all**”: le pasa ese parámetro a Odoo para facilitar tareas de desarrollo.
  - El detalle de que realiza esta opción se puede ver en el siguiente enlace <https://www.odoo.com/documentation/17.0/es/developer/misc/other/cmdline.html#cmdoption-odoo-bin-dev>

⚠ **Atención:** para poder desarrollar sin problemas, es recomendable darle todos los permisos al directorio “/home/usuario/OdooDesarrollo/volumesOdoo/addons”, con un comando similar a “**sudo chmod -R 777 /home/usuario/volumesOdoo/addons**”.

Con esto, tendremos listo nuestro entorno de desarrollo “Dockerizado”. Hemos conseguido que los contenedores corran de manera aislada los servicios de base de datos y Odoo, mientras que nosotros podremos desarrollar usando un IDE instalado en el anfitrión trabajando dentro del directorio “/home/usuario/OdooDesarrollo/addons”.

#### 5.9. Odoo 17 en Docker: - Parte 3: Docker Compose para Odoo

💬 **Interesante:** si conoces como usar “Docker Compose”, puedes serte muy útil revisar este curso con ejemplos prácticos <https://sergarb1.github.io/CursoIntroduccionADocker/>

💬 **Interesante:** si ya conoces como usar “Docker Compose”, puedes usar esta “CheatSheet” <https://raw.githubusercontent.com/sergarb1/CursoIntroduccionADocker/main/FuentesCurso/Docker%20CheatSheet%20COMPLETA.pdf>

Docker Compose es una herramienta que nos facilita el despliegue de varios contenedores usando una configuración definida en un fichero. Este fichero por defecto debe llamarse “**docker-compose.yml**”.

Si nos situamos en el directorio donde esté nuestro fichero “docker-compose.yml”, podemos iniciar el servicio completo simplemente escribiendo:

```
docker-compose up -d
```

Al lanzar esta orden, se crearán en el directorio actual:

- **Carpeta “addons”:** ahí desarrollaremos nuestros módulos mediante un IDE externo.
- **Carpeta “pgData”:** ahí se almacenará la persistencia de datos de nuestra base de datos.

⚠ **Atención:** para poder desarrollar sin problemas, es recomendable darle todos los permisos al directorio “addons”, con un comando similar a “**sudo chmod -R 777 ./addons**”.

Podemos parar el servicio completo simplemente escribiendo:

```
docker-compose down
```

Adjuntamos a esta unidad un zip con el fichero “docker-compose.yml” para entorno de producción y con el fichero “docker-compose.yml” para entorno de desarrollo. A continuación, además mostramos el contenido del fichero “docker-compose.yml” para entorno de desarrollo.

#### Fichero “docker-compose.yml” (desarrollo):

```
version: '3.3'

services:
  #Definimos el servicio Web, en este caso Odoo
  web:
    #Indicamos que imagen de Docker Hub utilizaremos
    image: odoo:17
    #Indicamos que depende de "db", por lo cual debe ser procesada primero "db"
    depends_on:
      - db

    # Port Mapping: indicamos que el puerto 8069 del contenedor se mapeara con el
    mismo puerto en el anfitrión
    # Permitiendo acceder a Odoo mediante http://localhost:8069
    ports:
      - 8069:8069

    # Mapeamos el directorio de los contenedores (como por ejemplo "/mnt/extra-
    addons" )
    # en un directorio local (como por ejemplo en un directorio
    "./volumesOdoo/addons")
    # situado en el lugar donde ejecutemos "Docker compose"
    volumes:
      - ./volumesOdoo/addons:/mnt/extra-addons
      - ./volumesOdoo/odoo/filestore:/var/lib/odoo/filestore
      - ./volumesOdoo/odoo/sessions:/var/lib/odoo/sessions


    # Definimos variables de entorno de Odoo
    environment:
      - HOST=db
      - USER=odoo
      - PASSWORD=odoo
    # Indica que pasa ese parametro al arrancar el servicio Odoo
    command: --dev=all
  #Definimos el servicio de la base de datos
  db:
    image: postgres:10

    # Definimos variables de entorno de PostgreSQL
```

```
environment:
  - POSTGRES_PASSWORD=odoo
  - POSTGRES_USER=odoo
  - POSTGRES_DB=postgres
  # Mapeamos el directorio del contenedor "var/lib/postgresql/data" en un
  # directorio "./volumesOdoo/dataPostgreSQL"
  # situado en el lugar donde ejecutemos "Docker compose"
volumes:
  - ./volumesOdoo/dataPostgreSQL:/var/lib/postgresql/data
```

## 6. PUESTA EN MARCHA DE ODOO 17

Una vez realizada la instalación con cualquiera de las alternativas propuestas anteriormente, accederemos mediante nuestro navegador a Odoo mediante <http://localhost:8069> y deberemos realizar una configuración inicial. Aquí un ejemplo de dicha configuración:



Warning, your Odoo database manager is not protected. To secure it, we have generated the following master password for it:

**53cu-diye-asm4**

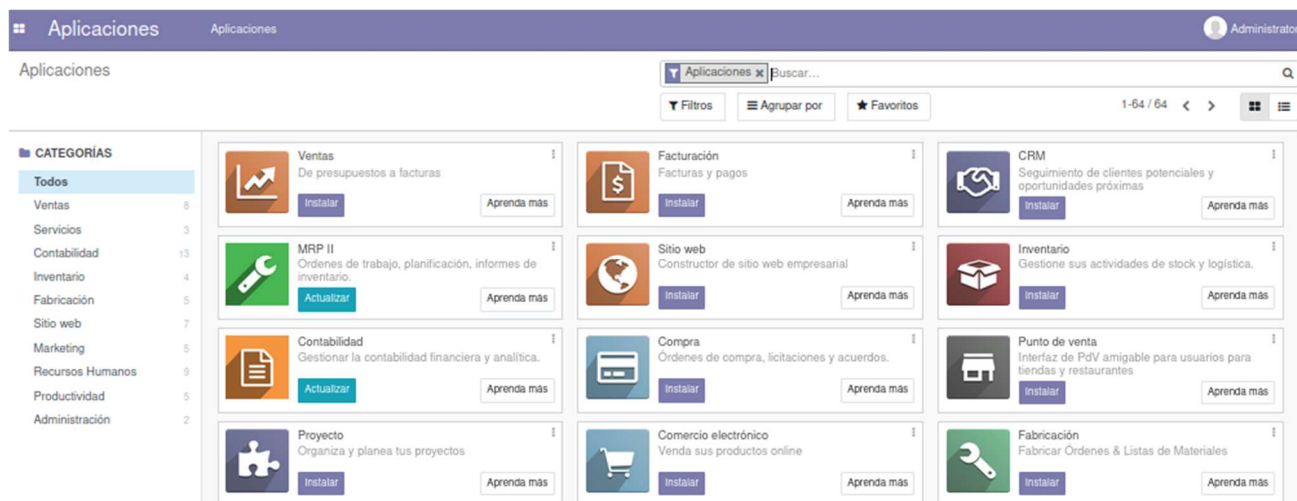
You can change it below but be sure to remember it, it will be asked for future operations on databases.

Master Password	<input type="password" value="....."/>
Database Name	<input type="text" value="postgre"/>
Email	<input type="text" value="correo.cuenta.odoo@gmail.com"/>
Password	<input type="password" value="...."/>
Phone number	<input type="text"/>
Language	<input type="text" value="Spanish / Español"/>
Country	<input type="text" value="Spain"/>
Demo data	<input type="checkbox"/>
<input type="button" value="Create database"/> or restore a database	

A primera vista, se nos mostrará un “Password maestro” que podemos cambiar si queremos. Deberemos almacenar en un lugar seguro ese “Password maestro” para poder recuperar nuestro sistema ante problemas de usuarios.

Además, se nos pedirá configurar Odoo según los parámetros de nuestra instalación. En esta configuración crearemos un usuario administrador, e indicaremos nuestro país (esto realizará algunas adaptaciones para empresas locales) e idioma de Odoo. Incluso nos permite cargar la instalación con datos de demostración (útil para realizar pruebas, conocer cómo funciona Odoo, etc.).

Una vez esté todo listo, al pulsar “Create database” se creará e inicializará Odoo. Tened paciencia (tarda unos minutos). Si todo ha ido bien, llegaréis a una pantalla similar a la siguiente:



Esto indica que Odoo 17 se ha instalado correctamente y ya podemos trabajar con él.