

Práctica 1: Montaje del Entorno de Desarrollo para Odoo 18 con Docker y Docker Compose

1. Introducción y objetivos

2. Proceso de configuración

3. Errores encontrados y soluciones aplicadas

4. Conclusiones finales

1. Introducción y objetivos

El objetivo principal de esta práctica es adquirir conocimientos básicos en la configuración y gestión de entornos de desarrollo utilizando tecnologías actuales como **Docker**, **Docker Compose** y **Visual Studio Code**.

Para ello deberemos alcanzar la siguiente meta:
Montar y levantar un entorno de desarrollo para Odoo 18 con los contenedores indispensables y demostrar el correcto funcionamiento del mismo ejecutándolo satisfactoriamente en un navegador.

Los servicios necesarios serán:

- Un Sistema de Planificación de Recursos Empresariales o ERP: **Odoo**
- Un Sistema de Gestión de Bases de Datos: **PostgreSQL**
- Opcionalmente un Visor Web de Bases de Datos para comprobar el correcto funcionamiento de los 2 servicios anteriores: **Adminer**

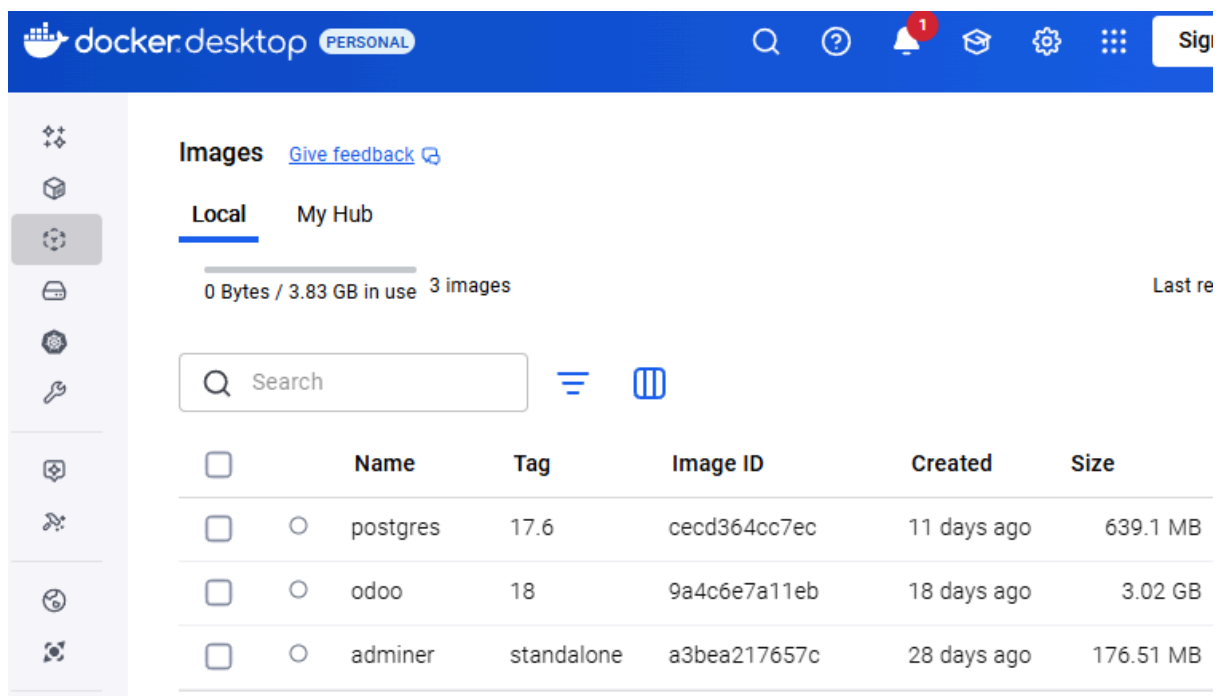
Tanto Docker como Docker Compose se instalarán con el instalador **Docker Desktop**, que aparte de configurar automáticamente el entorno también nos servirá como interfaz gráfica de usuario.

2. Proceso de configuración

Las imágenes de Docker utilizadas han sido descargadas de la web Docker Hub Container Image Library (<https://hub.docker.com/>) y son las siguientes:

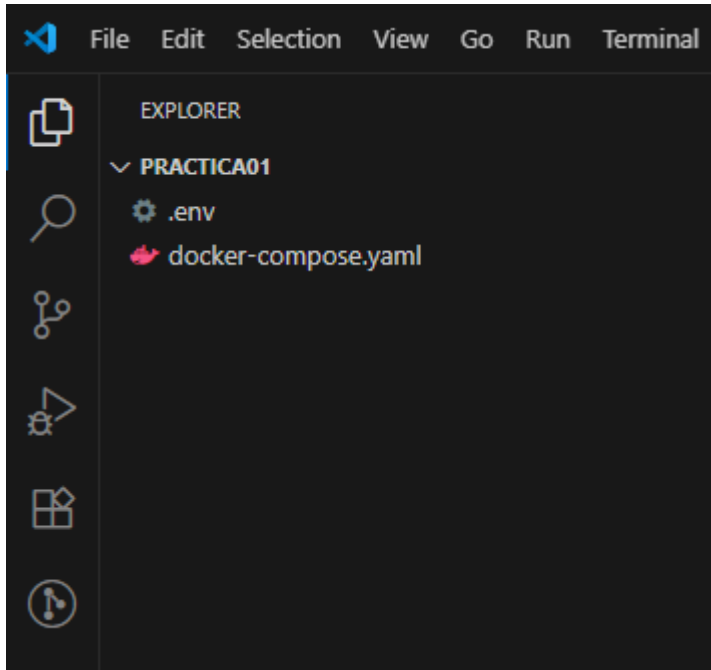
- Imagen oficial de Odoo, versión 18
- Imagen oficial de PostgreSQL, versión 17.6
- Imagen oficial de Adminer, versión 5.4 o standalone

Una vez instaladas el resultado que deberíamos obtener al listar Imágenes en Docker Desktop debería ser similar a la siguiente captura:

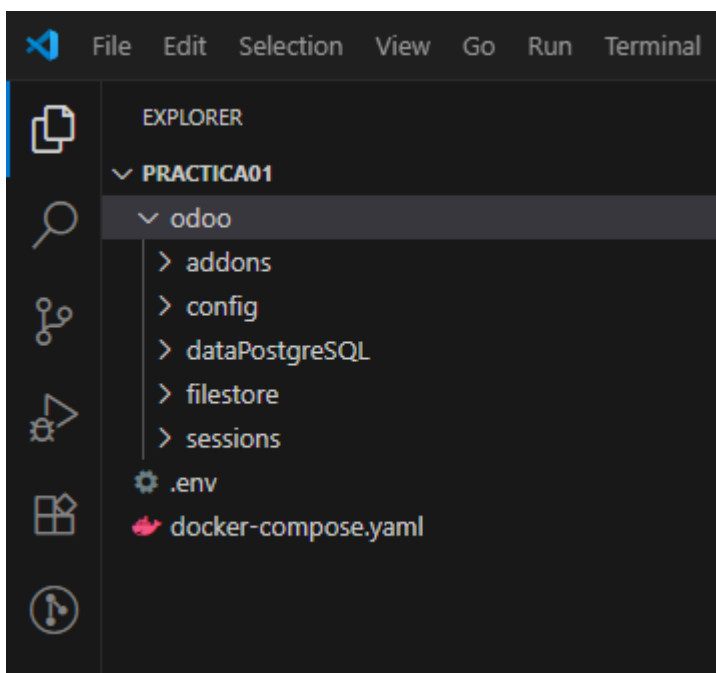


Una vez preparado el sistema Docker para generar contenedores en base a las imágenes instaladas deberemos configurar Visual Studio Code.

Para ello crearemos un nuevo proyecto con la siguiente estructura:



Una vez generados los contenedores quedará así:



Para conseguir la estructura anterior deberemos configurar el archivo **docker-compose.yaml** tal que así:

```
docker-compose.yaml
1  services:
2    web:
3      container_name: d-odoo
4      image: odoo:18
5      depends_on:
6        - db
7      ports:
8        - 8069:8069
9      volumes:
10       - ./odoo/addons:/mnt/extra-addons
11       - ./odoo/filestore:/var/lib/odoo/filestore
12       - ./odoo/sessions:/var/lib/odoo/sessions
13       - ./odoo/config/etc/odoo
14      environment:
15        - HOST=db
16        - USER=odoo
17        - PASSWORD=odoo
18      command: --dev=all
19
20      # Database management in a single PHP file.
21      # Comprobaremos si se crean las bases de datos
22      adminer:
23        container_name: d-adminer
24        image: adminer:standalone
25        restart: always
26        ports:
27          - 8080:8080
28
29      db:
30        container_name: d-postgres
31        image: postgres:17.6
32        environment:
33          - POSTGRES_PASSWORD=odoo
34          - POSTGRES_USER=odoo
35          - POSTGRES_DB=postgres
36        volumes:
37          - ./odoo/dataPostgreSQL:/var/lib/postgresql/data
38
```

En este ejemplo no se utilizan variables de entorno.

Una vez tengamos todo configurado sólo deberemos ejecutar los comandos '**docker compose up -d**' para levantar los contenedores de forma rápida y '**docker compose ps**' para comprobar que todo el proceso se ha realizado correctamente.

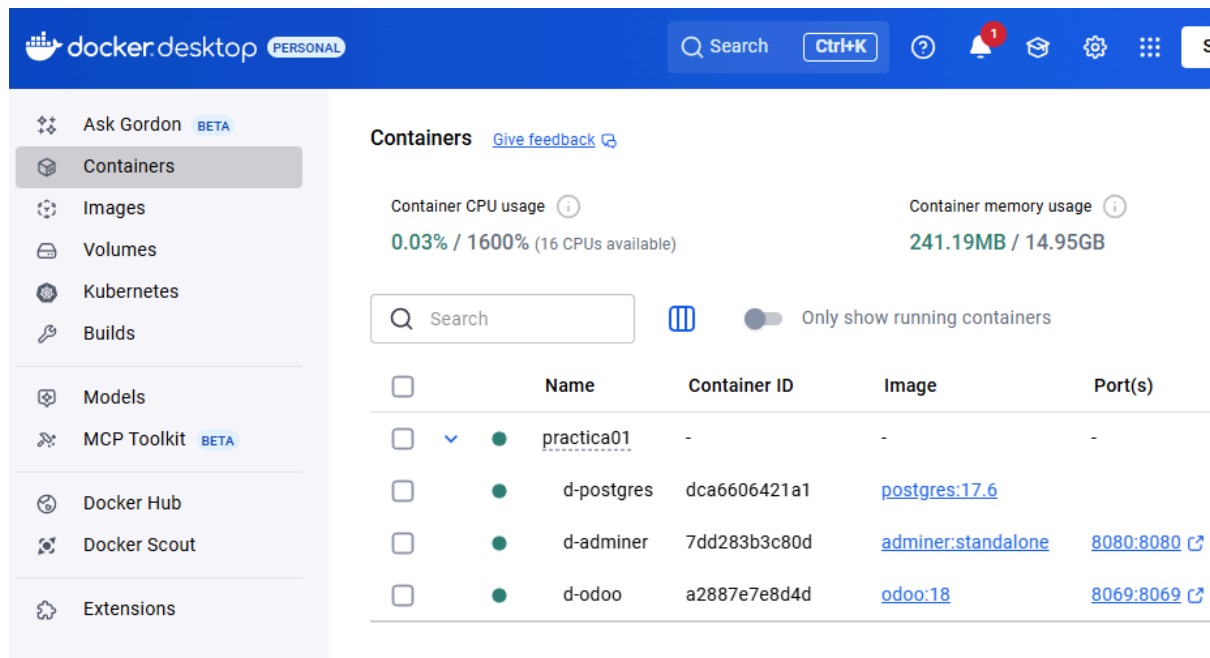
```
PS C:\Users\cristianf\Documents\proyectos\code\Practica01> docker compose up -d
[+] Running 3/3
 ✓ Container d-postgres Started
 ✓ Container d-adminer Started
 ✓ Container d-odoo Started
PS C:\Users\cristianf\Documents\proyectos\code\Practica01> docker compose ps
```

NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS
d-adminer	adminer:standalone	"entrypoint.sh docke..."	adminer	10 minutes ago	Up 10 seconds
d-odoo	odoo:18	"/entrypoint.sh --de..."	web	10 minutes ago	Up 9 seconds
d-postgres	postgres:17.6	"docker-entrypoint.s..."	db	10 minutes ago	Up 9 seconds

```
PS C:\Users\cristianf\Documents\proyectos\code\Practica01>
```

```
PORTS
0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp
0.0.0.0:8069->8069/tcp, [::]:8069->8069/tcp
5432/tcp
```

También podemos comprobar si todo ha ido bien en Docker Desktop.



Si quisiéramos parar y borrar los contenedores creados anteriormente deberíamos ejecutar el comando **'docker compose down'**.

El resultado en consola sería similar al siguiente:

```
PS C:\Users\cristianf\Documents\proyectos\code\Practica01> docker compose down
[+] Running 4/4
✓ Container d-adminer      Removed
✓ Container d-odoo         Removed
✓ Container d-postgres     Removed
✓ Network practica01_default Removed
PS C:\Users\cristianf\Documents\proyectos\code\Practica01> 
```

Abramos el servicio de odoo en el navegador (localhost:8069):

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

edby-ij7n-bkke

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

Master Password

Database Name → odoo-db

Email → odoo@odoo.com

Password →

Phone Number

Language English (US)

Country

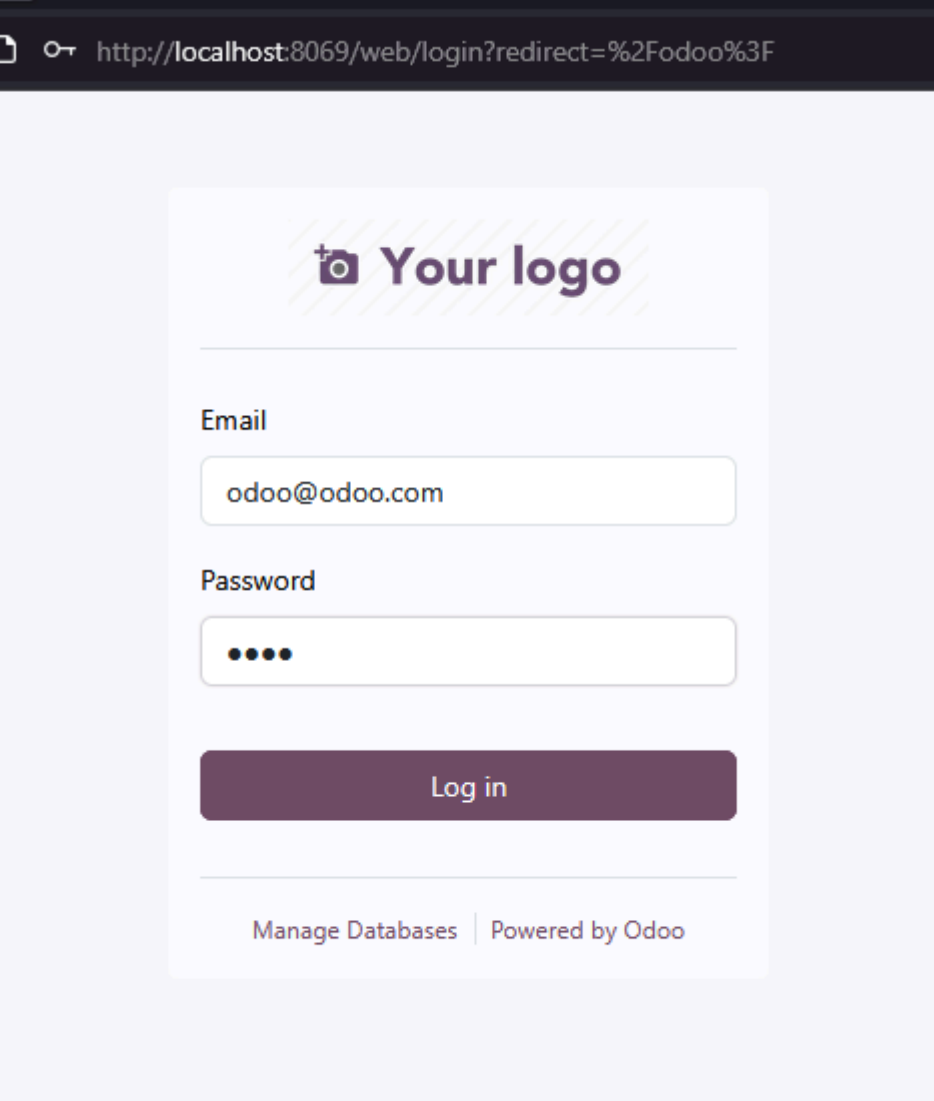
Demo Data ☐

Create database or restore a database

La primera vez siempre aparecerá esta vista de configuración inicial en la que deberemos cubrir los campos de Database Name, Email y Password.

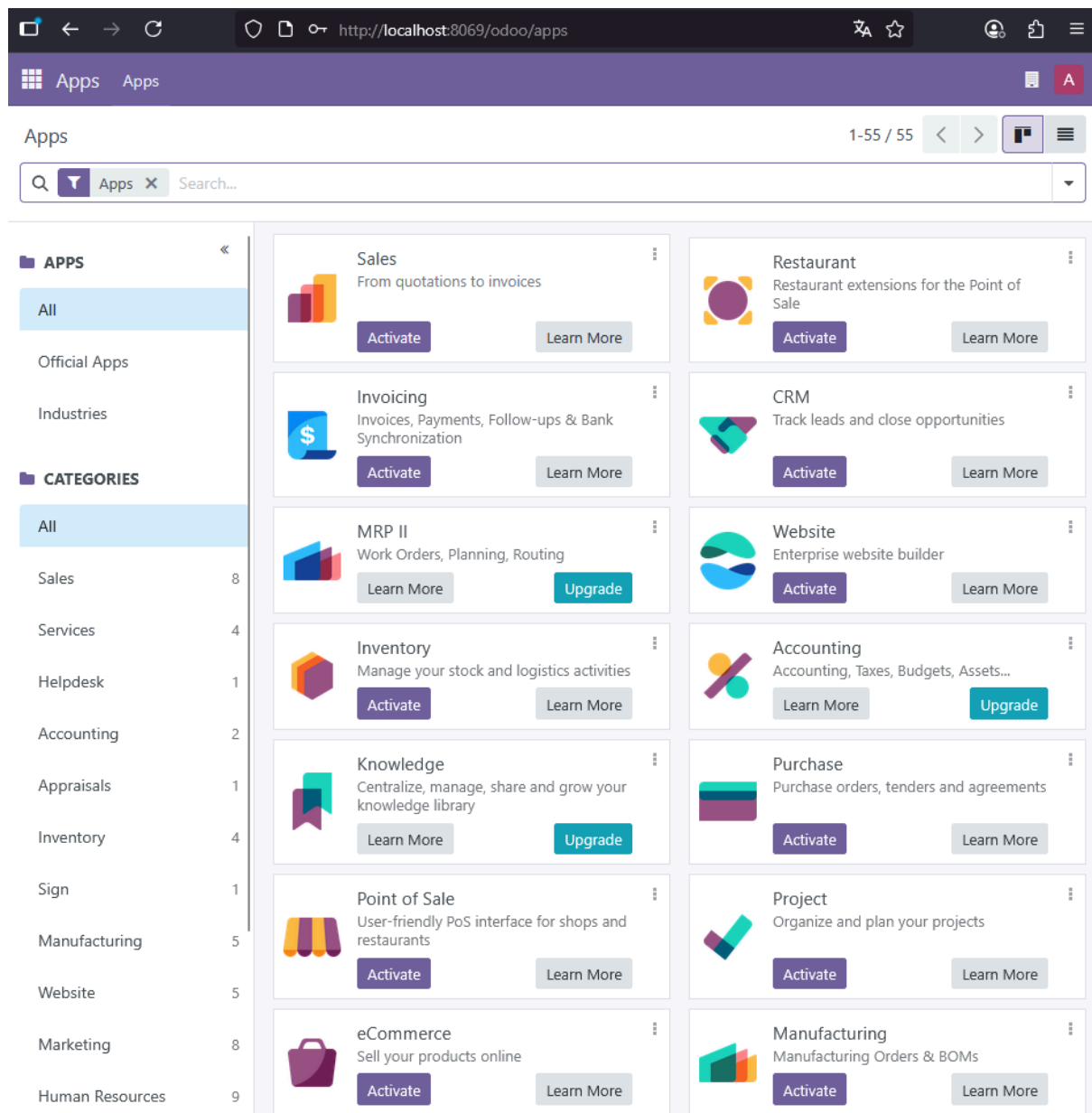
Una vez cubiertos pulsamos el botón *Create database*.

Nos redireccionará automáticamente a una vista de login.

A screenshot of a web browser showing the Odoo login page. The browser's address bar displays the URL: http://localhost:8069/web/login?redirect=%2Fodoo%3F. The login form is centered on a light purple background. It features a header with a camera icon and the text "Your logo". Below this is a horizontal line. The form contains two input fields: "Email" with the value "odoo@odoo.com" and "Password" with four dots indicating a masked password. A dark purple "Log in" button is positioned below the password field. At the bottom of the form, there is a horizontal line and two links: "Manage Databases" and "Powered by Odoo".

Introducimos el Email y el Password creados anteriormente y pulsamos el botón *Log in*.

Y entraremos al fin en el servicio de odoo.



Como curiosidad podemos comprobar si la base de datos que hemos creado en la primera conexión a odoo se ha generado correctamente.

Para ello usaremos Adminer, el contenedor opcional empleado en esta práctica.

Abrimos Adminer en el navegador (localhost:8080):

Idioma: Español

Adminer 5.4.0 5.4.1

Login

Motor de base de datos	PostgreSQL
Servidor	d-postgres
Usuario	odoo
Contraseña	••••
Base de datos	

☐ Guardar contraseña

- Seleccionamos PostgreSQL en Motor de base de datos.
- Introducimos en **nombre del contenedor de postgres** en el campo de Servidor.
- Introducimos el nombre de usuario asignado a la variable POSTGRES_USER del docker-compose.yaml
- Introducimos la contraseña asignada a la variable POSTGRES_PASSWORD del archivo .yaml

Y finalmente pulsamos el botón de Login.

Una vez dentro podemos comprobar que la base de datos se creó correctamente.

http://localhost:8080/?pgsql=d-postgres&username=odoo

PostgreSQL » d-postgres odoo

Seleccionar Base de datos

[Crear Base de datos](#) [Lista de procesos](#) [Variables](#)

Versión PostgreSQL: **17.6** a través de la extensión de PHP **PDO_PgSQL**

Logueado como: **odoo**

	Base de datos	Colación	Tablas	Size - Compute
<input type="checkbox"/>	odoo-db	C	?	?
<input type="checkbox"/>	postgres	en_US.utf8	?	?
<input type="checkbox"/>	template1	en_US.utf8	?	?

Selected (0)

[Eliminar](#)

Plugins cargados

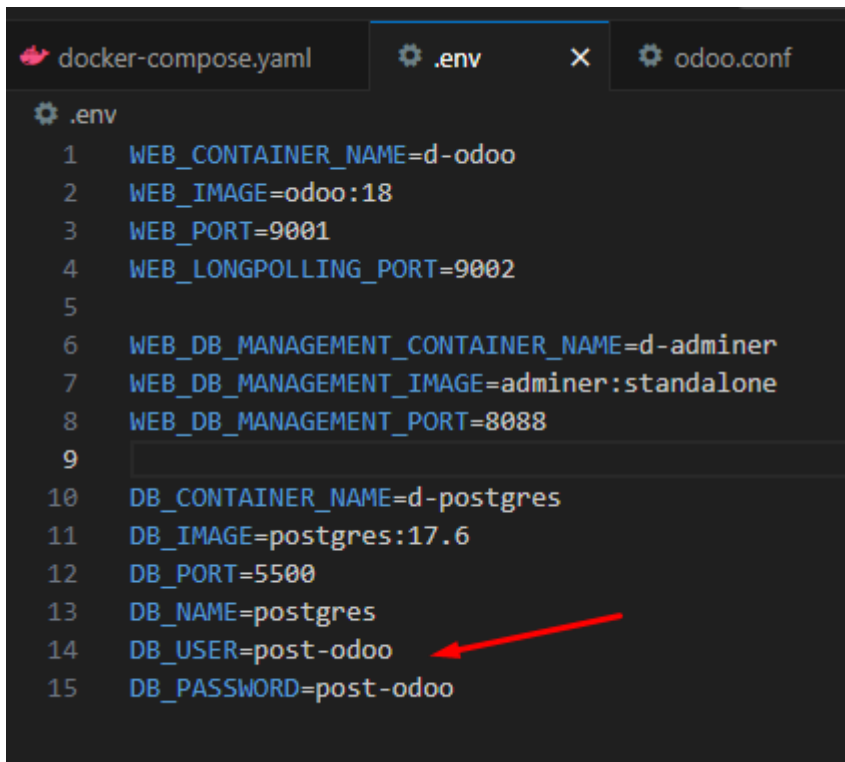
- **docker\DefaultServerPlugin:** Prefills the "Server" field with the ADMINER_DEFAULT_SERVER environment variable.

3. Errores encontrados y soluciones aplicadas

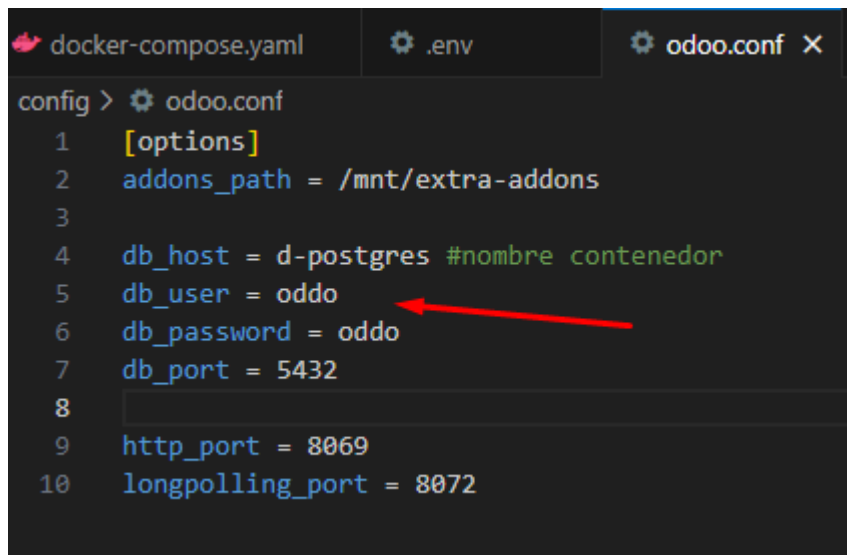
Algunos errores con los que me he topado siempre están relacionados con el correcto enlace de archivos de configuración, como podemos comprobar en el siguiente ejemplo:

```
d-postgres | Connection matched file "/var/lib/postgresql/data/pg_hba.conf" line 128: "host all all all scram-sha-256"
d-postgres | 2025-10-10 16:05:07.765 UTC [86] FATAL: password authentication failed for user "odoo"
d-postgres | 2025-10-10 16:05:07.765 UTC [86] DETAIL: Role "oddo" does not exist.
d-postgres | Connection matched file "/var/lib/postgresql/data/pg_hba.conf" line 128: "host all all all scram-sha-256"
d-postgres | 2025-10-10 16:05:08.774 UTC [87] FATAL: password authentication failed for user "oddo"
d-postgres | 2025-10-10 16:05:08.774 UTC [87] DETAIL: Role "oddo" does not exist.
d-postgres | Connection matched file "/var/lib/postgresql/data/pg_hba.conf" line 128: "host all all all scram-sha-256"
d-odoo | Database connection failure: connection to server at "d-postgres" (172.19.0.3), port 5432 failed: FATAL: p
or user "oddo"
d-odoo |
d-odoo exited with code 1
d-postgres | 2025-10-10 16:09:48.660 UTC [62] LOG: checkpoint starting: time
d-postgres | 2025-10-10 16:09:53.049 UTC [62] LOG: checkpoint complete: wrote 46 buffers (0.3%); 0 WAL file(s) added, 0
48 s, sync=0.010 s, total=4.390 s; sync files=11, longest=0.002 s, average=0.001 s; distance=269 kB, estimate=269 kB; lsn
```

Al intentar levantar los contenedores, odoo nunca llega a iniciarse. Comprobando los distintos archivos de configuración observamos lo siguiente:



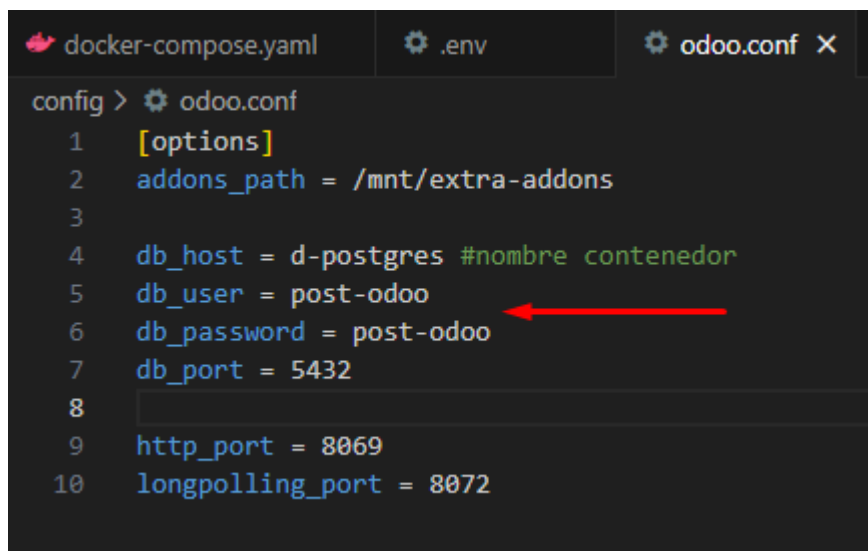
```
docker-compose.yaml .env x odoo.conf
.env
1 WEB_CONTAINER_NAME=d-odoo
2 WEB_IMAGE=odoo:18
3 WEB_PORT=9001
4 WEB_LONGPOLLING_PORT=9002
5
6 WEB_DB_MANAGEMENT_CONTAINER_NAME=d-adminer
7 WEB_DB_MANAGEMENT_IMAGE=adminer:standalone
8 WEB_DB_MANAGEMENT_PORT=8088
9
10 DB_CONTAINER_NAME=d-postgres
11 DB_IMAGE=postgres:17.6
12 DB_PORT=5500
13 DB_NAME=postgres
14 DB_USER=postgres
15 DB_PASSWORD=postgres
```



```
config > odoo.conf
1  [options]
2  addons_path = /mnt/extra-addons
3
4  db_host = d-postgres #nombre contenedor
5  db_user = ododo
6  db_password = ododo
7  db_port = 5432
8
9  http_port = 8069
10 longpolling_port = 8072
```

Existe una discrepancia entre el archivo de configuración de las variables de entorno **.env** y el archivo de configuración de odoo **odoo.conf**.

En este caso nos interesa que los campos del archivo de odoo coincidan con los env tal que así:



```
config > odoo.conf
1  [options]
2  addons_path = /mnt/extra-addons
3
4  db_host = d-postgres #nombre contenedor
5  db_user = post-odoo
6  db_password = post-odoo
7  db_port = 5432
8
9  http_port = 8069
10 longpolling_port = 8072
```

Si volvemos a levantar los contenedores observamos lo siguiente:

```
d-postgres | 2025-10-10 17:04:27.090 UTC [50] LOG: checkpoint complete: wrote 3 buffers (0.0%); 0 WAL file(s)
total=0.043 s; sync files=2, longest=0.003 s, average=0.003 s; distance=0 kB, estimate=0 kB; lsn=0/14ED7B8, redo
d-postgres | 2025-10-10 17:04:27.104 UTC [49] LOG: database system is shut down
d-postgres | done
d-postgres | server stopped
d-postgres |
d-postgres | PostgreSQL init process complete; ready for start up.
d-postgres | 2025-10-10 17:04:27.193 UTC [1] LOG: starting PostgreSQL 17.6 (Debian 17.6-2.pgdg13+1) on x86_64-
-bit
d-postgres | 2025-10-10 17:04:27.194 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
d-postgres | 2025-10-10 17:04:27.194 UTC [1] LOG: listening on IPv6 address ":::", port 5432
d-postgres | 2025-10-10 17:04:27.199 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
d-postgres | 2025-10-10 17:04:27.227 UTC [63] LOG: database system was shut down at 2025-10-10 17:04:27 UTC
d-postgres | 2025-10-10 17:04:27.264 UTC [1] LOG: database system is ready to accept connections
d-odoo | 2025-10-10 17:04:28.594 1 INFO ? odoo: Odoo version 18.0-20250918
d-odoo | 2025-10-10 17:04:28.595 1 INFO ? odoo: Using configuration file at /etc/odoo/odoo.conf
d-odoo | 2025-10-10 17:04:28.596 1 INFO ? odoo: addons paths: ['/usr/lib/python3/dist-packages/odoo/addons'
a-addons']
d-odoo | 2025-10-10 17:04:28.596 1 INFO ? odoo: database: post-odoo@d-postgres:5432
d-odoo | Warn: Can't find .pfb for face 'Courier'
d-odoo | 2025-10-10 17:04:28.778 1 INFO ? odoo.addons.base.models.ir_actions_report: Will use the Wkhtmltop
d-odoo | 2025-10-10 17:04:28.817 1 INFO ? odoo.addons.base.models.ir_actions_report: Will use the Wkhtmltop
d-odoo | 2025-10-10 17:04:29.037 1 INFO ? odoo.service.server: Watching addons folder /usr/lib/python3/dist
d-odoo | 2025-10-10 17:04:29.037 1 INFO ? odoo.service.server: Watching addons folder /var/lib/odoo/.local/
d-odoo | 2025-10-10 17:04:29.037 1 INFO ? odoo.service.server: Watching addons folder /mnt/extra-addons
d-odoo | 2025-10-10 17:04:31.157 1 INFO ? odoo.service.server: AutoReload watcher running with watchdog
d-odoo | 2025-10-10 17:04:31.157 1 INFO ? odoo.service.server: HTTP service (werkzeug) running on f4f9529f3
```

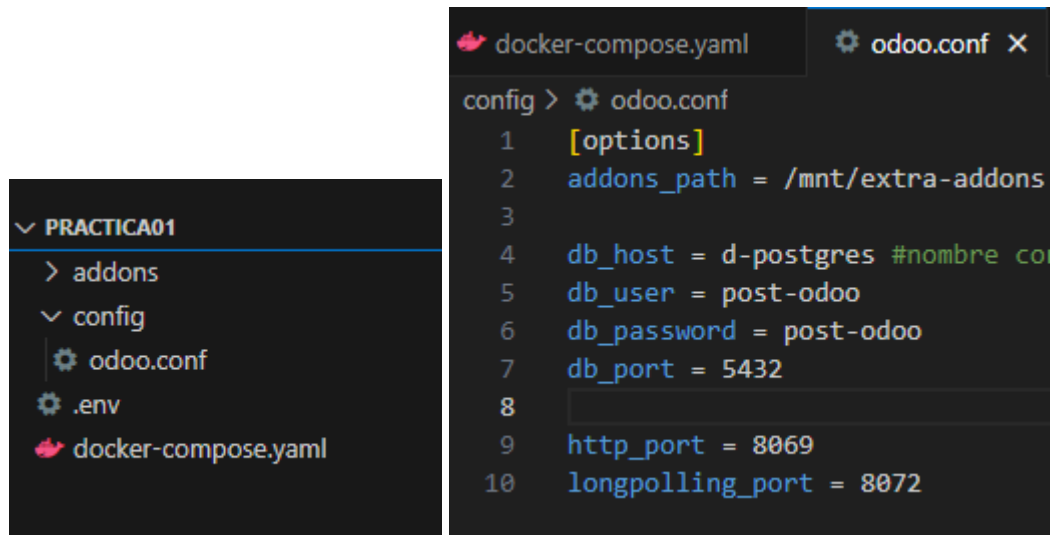
```
PS C:\Users\cristianf\Documents\proyectos\code\Practica01> docker compose ps
NAME                IMAGE              COMMAND                  SERVICE      CREATED      STATUS
d-adminer           adminer:standalone "entrypoint.sh docke..." web-db-manage 2 minutes ago Up 2 minutes
d-odoo              odoo:18            "/entrypoint.sh --de..." web           2 minutes ago Up 2 minutes
d-postgres          postgres:17.6      "docker-entrypoint.s..." db            2 minutes ago Up 2 minutes
```

```
PORTS
0.0.0.0:8088->8088/tcp, [::]:8088->8088/tcp
0.0.0.0:9001->8069/tcp, [::]:9001->8069/tcp, 0.0.0.0:9002->8072/tcp, [::]:9002->8072/tcp
0.0.0.0:5500->5432/tcp, [::]:5500->5432/tcp
```

Los contenedores se han levantado adecuadamente, ya tenemos todo el sistema preparado y listo para funcionar.

Las capturas definitivas del proyecto serían las siguientes:

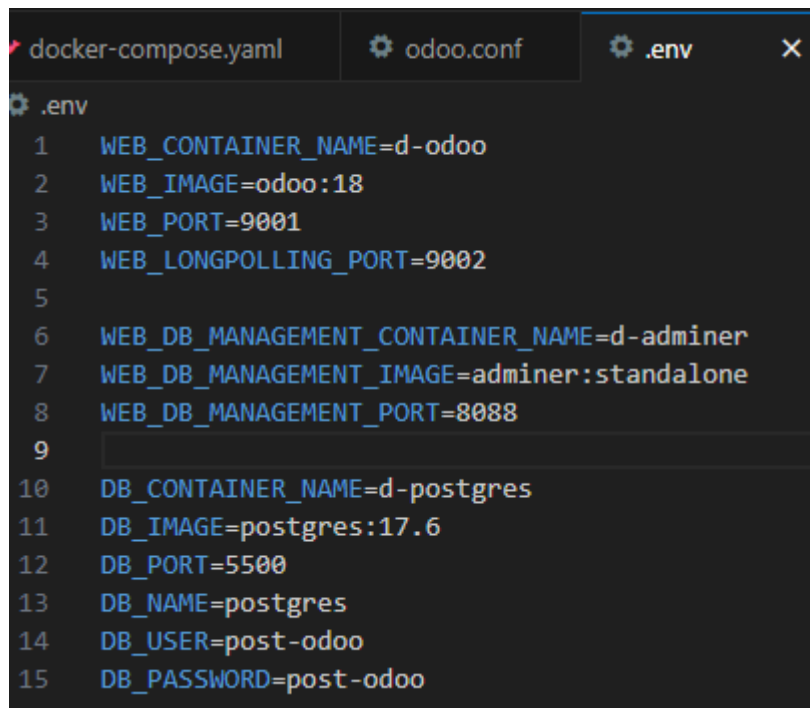
Estructura del proyecto y odoo.conf



The screenshot shows a code editor with two panels. The left panel displays the project structure under the name 'PRACTICA01'. It includes a folder 'addons', a folder 'config' containing 'odoo.conf', a file '.env', and a file 'docker-compose.yaml'. The right panel shows the content of 'odoo.conf' with the following configuration:

```
config > odoo.conf
1  [options]
2  addons_path = /mnt/extra-addons
3
4  db_host = d-postgres #nombre co
5  db_user = post-odoo
6  db_password = post-odoo
7  db_port = 5432
8
9  http_port = 8069
10 longpolling_port = 8072
```

.env



The screenshot shows a code editor with the '.env' file open. The file contains the following environment variables:

```
.env
1  WEB_CONTAINER_NAME=d-odoo
2  WEB_IMAGE=odoo:18
3  WEB_PORT=9001
4  WEB_LONGPOLLING_PORT=9002
5
6  WEB_DB_MANAGEMENT_CONTAINER_NAME=d-adminer
7  WEB_DB_MANAGEMENT_IMAGE=adminer:standalone
8  WEB_DB_MANAGEMENT_PORT=8088
9
10 DB_CONTAINER_NAME=d-postgres
11 DB_IMAGE=postgres:17.6
12 DB_PORT=5500
13 DB_NAME=postgres
14 DB_USER=post-odoo
15 DB_PASSWORD=post-odoo
```


docker-compose.yml

```
docker-compose.yml x  odoo.conf  .env

docker-compose.yml
1  services:
2
3      web:
4          container_name: ${WEB_CONTAINER_NAME}
5          image: ${WEB_IMAGE}
6          depends_on:
7              - db
8          ports:
9              # Redireccionamiento de puertos
10             - ${WEB_PORT}:8069
11             - ${WEB_LONGPOLLING_PORT}:8072
12          volumes:
13             - ./addons:/mnt/extra-addons
14             - ./config:/etc/odoo
15             - ./odoo/filestore:/var/lib/odoo/filestore
16             - ./odoo/sessions:/var/lib/odoo/sessions
17          environment:
18             - HOST=db
19             - USER=odoo
20             - PASSWORD=odoo
21          command: --dev=all
22
23          # Database management in a single PHP file.
24          # Comprobaremos si se crean las bases de datos
25          web-db-management:
26              container_name: ${WEB_DB_MANAGEMENT_CONTAINER_NAME}
27              image: ${WEB_DB_MANAGEMENT_IMAGE}
28              restart: always
29              ports:
30                 - ${WEB_DB_MANAGEMENT_PORT}:8080
31
32          db:
33              container_name: ${DB_CONTAINER_NAME}
34              image: ${DB_IMAGE}
35              environment:
36                 - POSTGRES_DB=${DB_NAME}
37                 - POSTGRES_USER=${DB_USER}
38                 - POSTGRES_PASSWORD=${DB_PASSWORD}
39              ports:
40                 - ${DB_PORT}:5432
41              volumes:
42                 - ./odoo/dataPostgreSQL:/var/lib/postgresql/data
43
```

4. Conclusiones finales

Docker es una herramienta poderosísima.

Tiene portabilidad, ya que los contenedores pueden ejecutarse en cualquier entorno compatible.

Tiene aislamiento, que evita conflictos de dependencias entre aplicaciones.

También ofrece eficiencia de recursos y escalabilidad (se pueden crear o eliminar múltiples instancias rápidamente), y agiliza el desarrollo y despliegue de aplicaciones.

Docker compose es una herramienta muy útil. Es una extensión de Docker que permite trabajar con varios contenedores de forma simultánea haciendo que estos se conecten y relacionen entre sí de la forma en la que el desarrollador crea conveniente.

La sensación después de terminar esta práctica es que todo el ecosistema Docker tiene un potencial para el desarrollo muy elevado.