

DOCUMENTO DE ERRORES Y SOLUCIONES

Alumno: Andoni Villanueva Urrestarazu

Alumno: xgarciaber@educacion.navarra.es

Alumno: itorranchu1@educacion.navarra.es

Alumno: dmoyavilch1@educacion.navarra.es

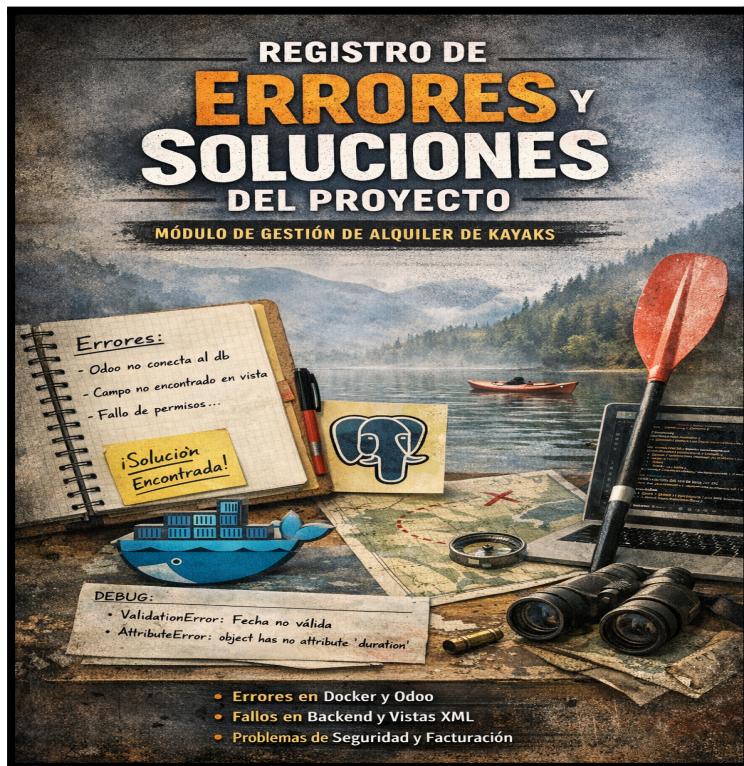
Docente: furtasuofi@educacion.navarra.es Fernando Urtasun

Docente: bgonzalpiz@educacion.navarra.es Beatriz Gonzalez Pizarro

Unidad Temaria : UT6

Grado : Desarrollo De Aplicaciones Multiplataforma (2.DAM.B)

Centro : Maria Ana Sanz 25/26



ÍNDICE

Introducción.....	3
1. Problemas de entorno (Docker y base de datos).....	3
Incidencia 1.1 – Odoo no conecta con PostgreSQL.....	3
Incidencia 1.2 – Nuestro módulo no aparecía en la lista de apps.....	4
2. Problemas en el backend (modelos y ORM).....	5
Incidencia 2.1 – Error de validación de fechas al crear reservas.....	5
Incidencia 2.2 – Atributo “duration” inexistente.....	5
3. Problemas en vistas XML.....	6
Incidencia 3.1 – Campo inexistente en una vista.....	6
Incidencia 3.2 – El botón “Confirmar reserva” no aparecía.....	7
4. Problemas de permisos.....	8
Incidencia 4.1 – Usuarios sin acceso a los servicios de kayak.....	8
5. Problemas de lógica de negocio.....	9
Incidencia 5.1 – No se generaba factura al confirmar una reserva.....	9
Conclusión.....	10

Introducción

Este compendio técnico ha sido diseñado para proporcionar al equipo de desarrollo y soporte una referencia rápida y fiable ante los desafíos técnicos más recurrentes en el ecosistema Odoo-Docker de **KayakYa**. El módulo, orientado a la gestión de alquileres de kayaks, integra flujos complejos de facturación, mensajería y validación temporal que pueden presentar estados de error específicos. Este documento no solo busca listar problemas, sino estandarizar las metodologías de diagnóstico y resolución para asegurar la continuidad del servicio.

1. Problemas de entorno (Docker y base de datos)

Incidencia 1.1 – Odoo no conecta con PostgreSQL

Qué pasó

Al arrancar los contenedores, Odoo no terminaba de iniciar y en los logs aparecía:

Python

```
psycopg2.OperationalError: could not connect to server: Connection refused
```

Qué estaba ocurriendo realmente

El contenedor de Odoo intentaba conectarse a la base de datos antes de que PostgreSQL estuviese listo, o bien estaba intentando conectarse a `localhost` en lugar al servicio de Docker.

Cómo lo solucionamos

- Comprobamos que el servicio de base de datos (`db`) estuviera levantado.

En el archivo de configuración de Odoo cambiamos:

Python

```
db_host = localhost
```

por

Python

```
db_host = db
```

- (el nombre del servicio en **docker-compose.yml**).
- Añadimos **depends_on** en Docker Compose para forzar el orden de arranque.

Desde ese momento Odoo pudo conectarse correctamente a la base de datos.

Incidencia 1.2 – Nuestro módulo no aparecía en la lista de apps

Qué pasó

Después de actualizar la lista de aplicaciones, el módulo no aparecía para instalar.

Causa detectada

El archivo **__manifest__.py** tenía un error de sintaxis y/o faltaba el campo:

Python

```
'installable': True
```

Solución aplicada

- Corregimos la sintaxis del diccionario del manifest.
- Reiniciamos Odoo actualizando directamente el módulo:

```
Python  
odoo -u kayak-ya
```

Tras eso, el módulo ya aparecía y se podía instalar.

2. Problemas en el backend (modelos y ORM)

Incidencia 2.1 – Error de validación de fechas al crear reservas

Síntoma

Al crear una reserva aparecía un **ValidationError** indicando que la fecha no era válida.

Motivo

Habíamos definido una restricción que impedía usar fechas anteriores al momento actual (`_check_dates`).

En pruebas estábamos usando fechas pasadas.

Qué hicimos

- Ajustamos las fechas de prueba para que fueran futuras.
- En entornos de test, desactivamos temporalmente la restricción para poder insertar datos antiguos.

Incidencia 2.2 – Atributo “duration” inexistente

Error mostrado

```
Python
```

```
AttributeError: 'kayak.booking' object has no attribute 'duration'
```

Qué lo provocaba

Estábamos intentando acceder a:

```
Python
```

```
self.duration
```

cuando realmente la duración pertenece al servicio relacionado.

Corrección

En el método de cálculo de la fecha fin cambiamos a:

```
Python
```

```
record.service_id.duration
```

Así Odoo pudo calcular correctamente **end_date**.

3. Problemas en vistas XML

Incidencia 3.1 – Campo inexistente en una vista

Error

Python

```
Field 'X' does not exist in model 'kayak.booking'
```

Por qué ocurrió

Añadimos un campo en el XML de la vista pero:

- o no estaba definido en el modelo Python
- o no habíamos actualizado el módulo tras añadirlo

Pasos para arreglarlo

1. Revisamos que el nombre del campo fuera exactamente el mismo en `.py` y `.xml`.

Reiniciamos y actualizamos el módulo dentro del contenedor:

None

```
docker exec -u 0 odoo odoo -u kayak-ya -d <bbdd> --stop-after-init
```

Incidencia 3.2 – El botón “Confirmar reserva” no aparecía

Qué veíamos

El botón desaparecía aunque la vista cargaba bien.

Causa

La condición de visibilidad era:

```
Python  
invisible="state != 'draft'"
```

Como el registro no estaba en estado **draft**, Odoo lo ocultaba.

Solución

Ajustamos la lógica de estados o modificamos la condición **invisible** para que el botón fuese visible en los estados necesarios.

4. Problemas de permisos

Incidencia 4.1 – Usuarios sin acceso a los servicios de kayak

Mensaje de error

```
Python  
Access Error
```

al intentar ver registros de **kayak.service**.

Origen del problema

Faltaban permisos en el archivo:

security/ir.model.access.csv

Arreglo

Añadimos las entradas correspondientes dando al menos permiso de lectura:

Python

```
perm_read,perm_write,perm_create,perm_unlink
```

para los modelos **kayak.booking** y **kayak.service** a los grupos de usuario necesarios.

5. Problemas de lógica de negocio

Incidencia 5.1 – No se generaba factura al confirmar una reserva

Comportamiento observado

La reserva cambiaba a confirmada pero el campo **invoice_id** quedaba vacío.

Qué estaba fallando

- El método **action_confirm** no llegaba a ejecutar **_create_invoice**, o
- el cliente no tenía datos suficientes para facturar.

Cómo lo resolvimos

- Revisamos los logs para detectar en qué punto fallaba el flujo.
- Completamos los datos del cliente (dirección y datos fiscales).
- Verificamos que el modelo **account.move** permitía crear facturas de tipo cliente (**out_invoice**).

Conclusión

Conclusión y Buenas Prácticas

La robustez del módulo KayakYa depende de un equilibrio entre la configuración del entorno (Docker), la integridad del modelo de datos y la correcta gestión de dependencias externas como los módulos de `mail` y `account`. Se recomienda:

- 1. Logs constantes:** Ante cualquier error, el comando `docker logs -f odoo` es la primera herramienta de diagnóstico.
- 2. Actualización del Manifiesto:** No olvides declarar cada nuevo archivo XML y cada nueva dependencia en el `__manifest__.py`.
- 3. Mantenimiento del Entorno:** Mantener las imágenes de PostgreSQL y Odoo actualizadas para evitar errores de compatibilidad de drivers (psycopg2).