Pythonistas GDL

Marketing Presentation

Python en un ERP

Todo lo que encuentras en Odoo

¿Qué es odoo?

Presented By: **Eliezer Solano M.**

29 de Mayo de 2026

Pythonistas GDL

Marketing Presentation

Python en un ERP

Todo lo que encuentras en Odoo

¿Qué es odoo?

Presented By: **Eliezer Solano M.**

29 de Mayo de 2026

Hablar de Python y compartir con las personas

Que python sea eterno...

Primer punto e importante

Disclaimer
Esto no es pagado, ni tampoco es promoción pagada ni nada de eso.

Criterio individual
La idea es desmenuzar, entender e informar

Contexto general
Sepan que encontrar, con que se come y todos sus sabores

¿Qué es Odoo?

Un framework

Odoo no es solo un ERP; es un **framework completo** que se construye sobre Python y PostgreSQL, ofreciendo una base sólida para el desarrollo de aplicaciones empresariales.

Python en un ERP

¿Qué es un ERP?

Enterprise Resource Planning es el sistema que centraliza todos los procesos de una empresa: ventas, inventario, RRHH, contabilidad, marketing... en una sola plataforma

Odoo es uno de los ERPs más populares y está construido sobre Python.

¿Por qué un framework?

Arquitectura base reutilizable

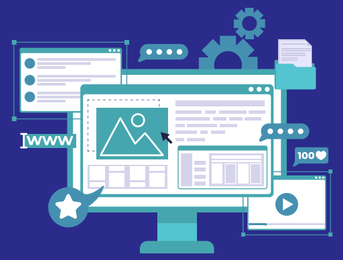
Define estructura, ciclo de vida y organización de aplicaciones

Propio (ORM) integrado.

Object-Relational Mapping (ORM) integrado sobre Python y PostgreSQL.

Motor de Plantillas

QWeb combina HTML y XML eficazmente



Inversión de Control (IoC)

El núcleo de Odoo (odoo-bin) arranca el motor y toma el control absoluto del flujo de ejecución, renderizado, persistencia y workflows.

Técnicamente

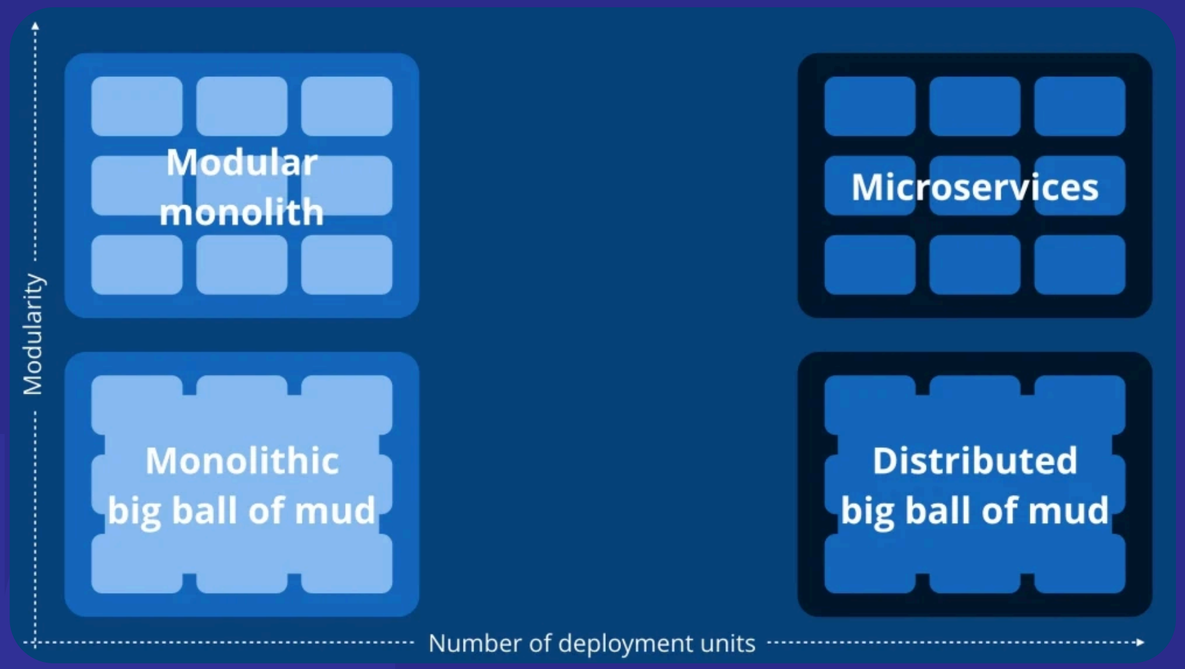
se describe como:

- **Monolito modular**

  Está dividido en miles de módulos independientes

- **Arquitectura de tres capas**

  Arquitectura modular de tres capas basada en el patrón MVC orientado a aplicaciones empresariales.



¿Por qué Odoo es popular?

Combina:

- Modularidad Lego y Store
- Versión Community
- Rapidez de implementación

Sobre la implementación

Community

La versión open source y gratuita del framework ERP

Enterprise

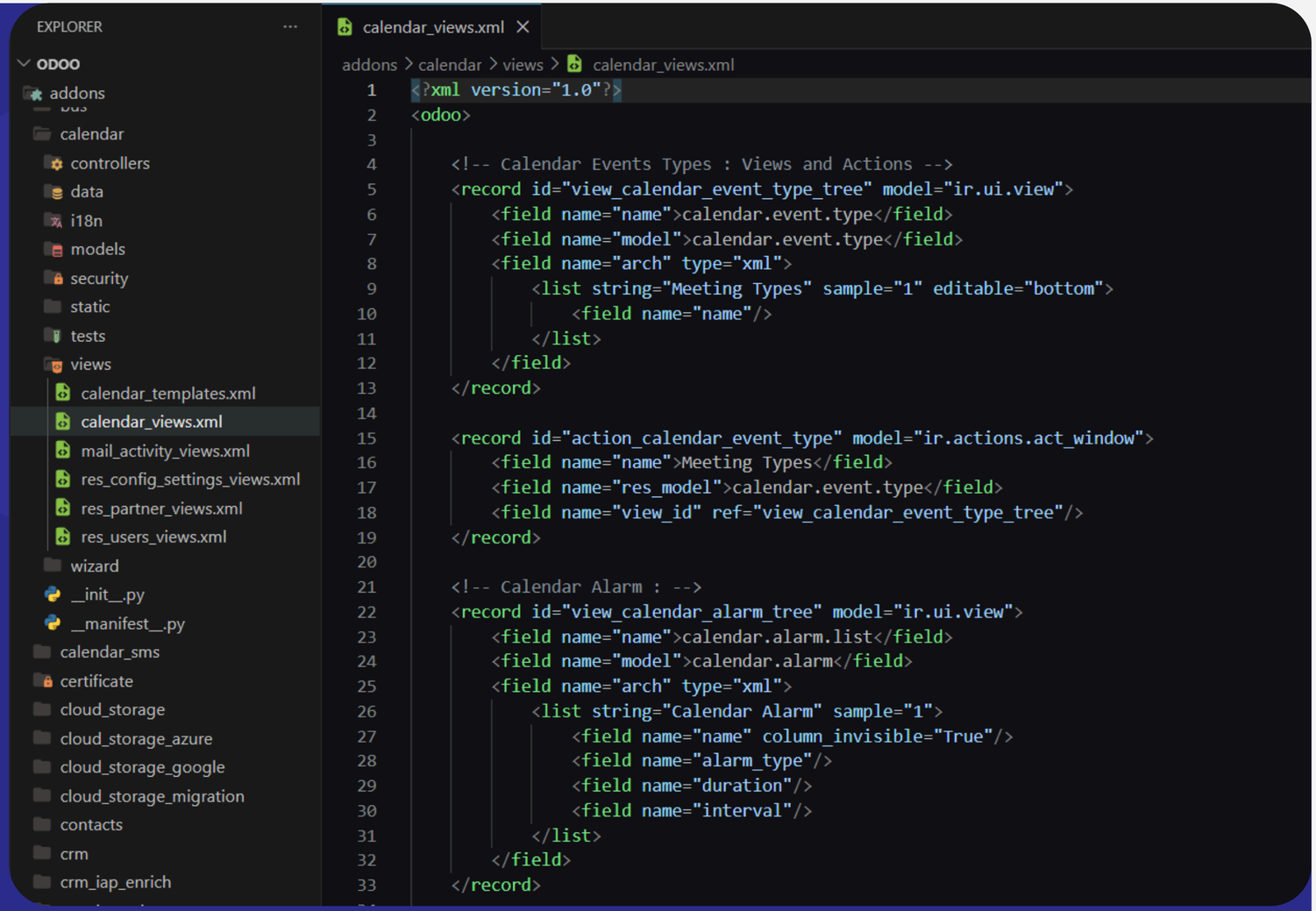
La versión de pago bajo licencia con módulos avanzados, soporte oficial, migraciones automáticas y una interfaz optimizada.

Primer Encuentro con Odoo

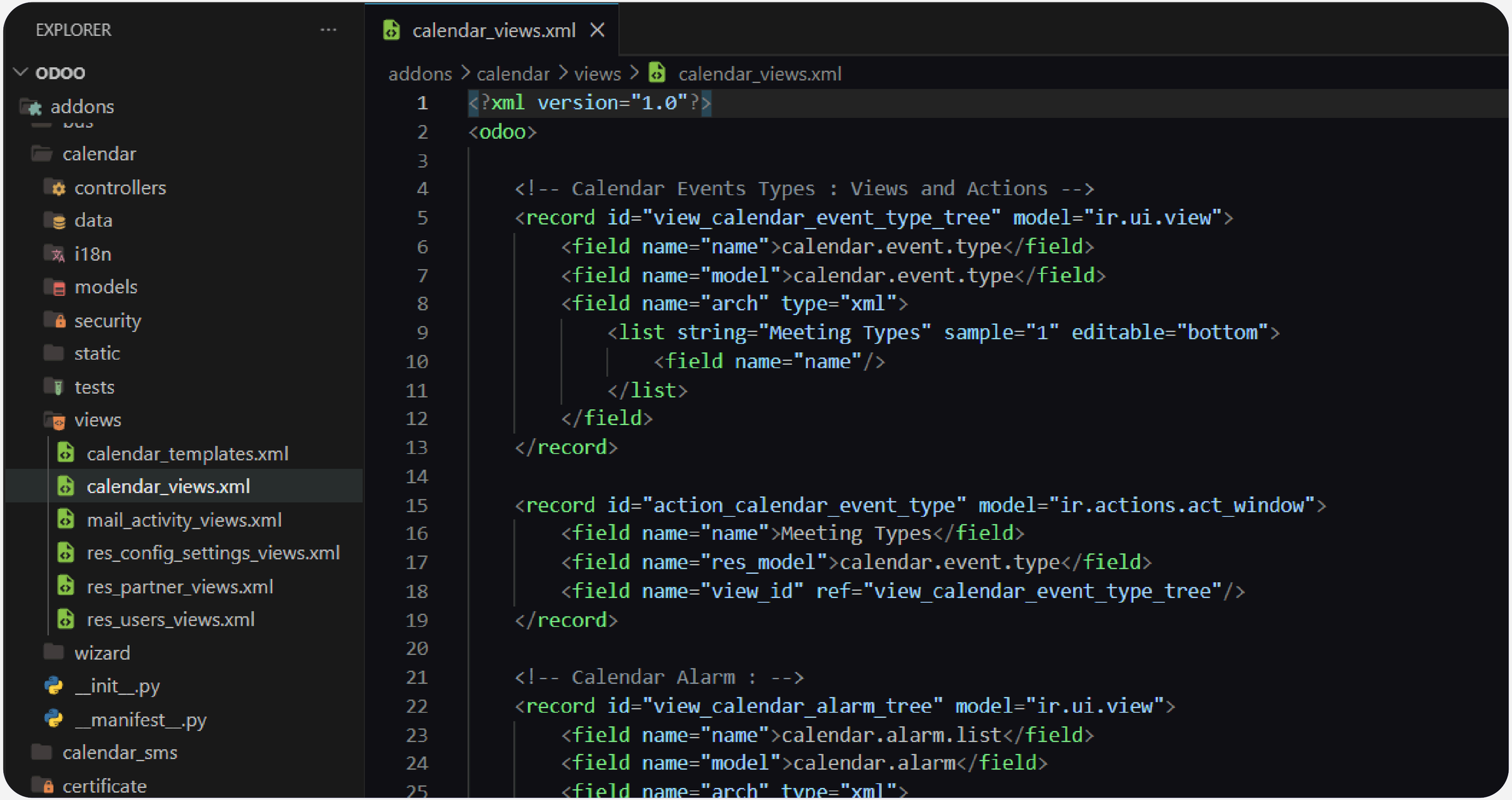
La magia de Odoo se revela al instalar módulos con facilidad, pero ese primer amor esconde complejidades y desafíos inesperados que se vuelven evidentes conforme se profundiza en el sistema.

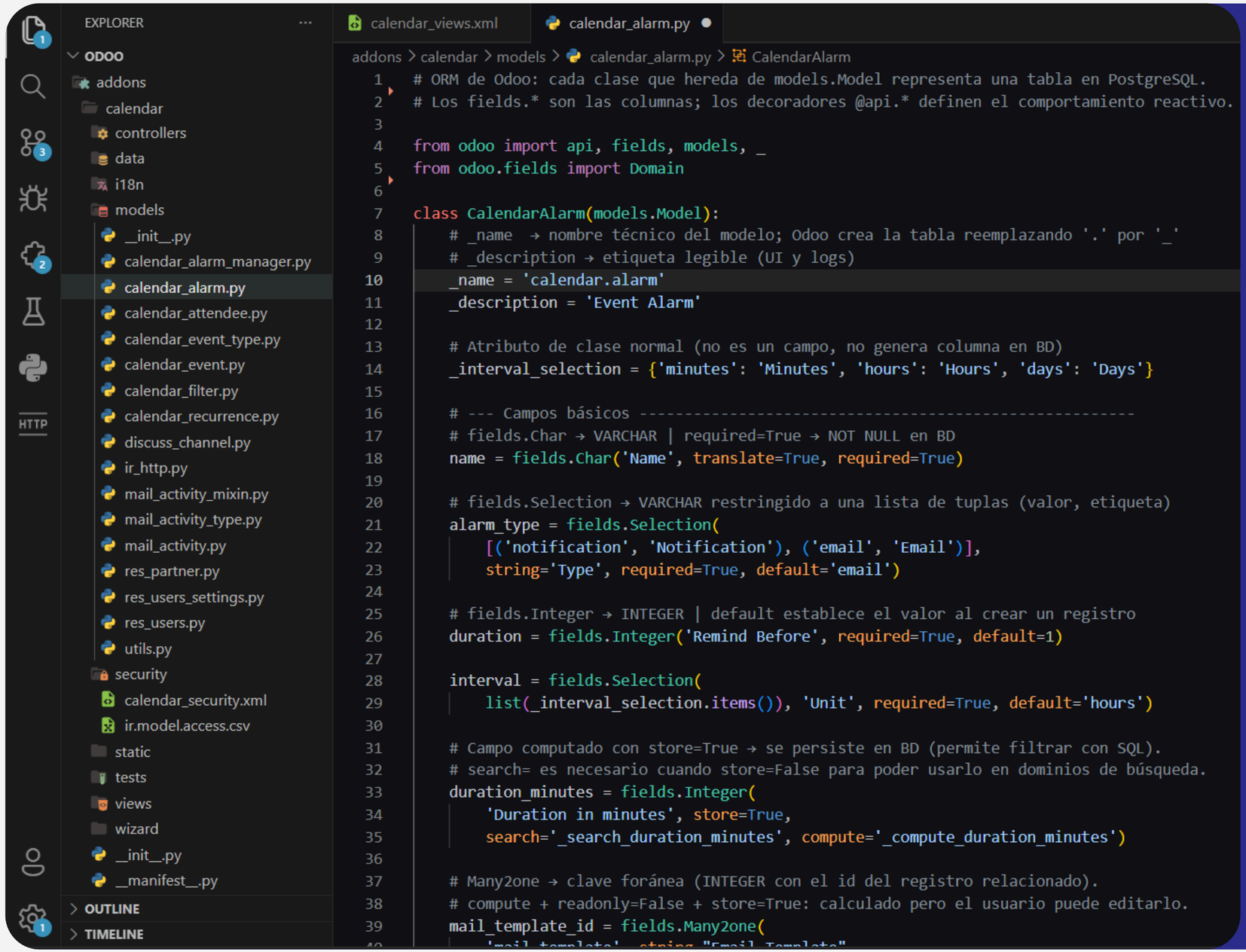
Un vistazo antes de . . .

Demasiado lore, como se ve realmente



Ejemplo de View: XML


EXPLORER
ODOO
addons
calendar
controllers
data
i18n
models
security
static
tests
views
calendar_templates.xml
calendar_views.xml
mail_activity_views.xml
res_config_settings_views.xml
res_partner_views.xml
res_users_views.xml
wizard
__init__.py
__manifest__.py
calendar_sms
certificate
calendar_views.xml
addons > calendar > views > calendar_views.xml
<?xml version="1.0"?>
<odoo>
<!-- Calendar Events Types : Views and Actions -->
<record id="view_calendar_event_type_tree" model="ir.ui.view">
<field name="name">calendar.event.type</field>
<field name="model">calendar.event.type</field>
<field name="arch" type="xml">
<list string="Meeting Types" sample="1" editable="bottom">
<field name="name"/>
</list>
</field>
</record>
<record id="action_calendar_event_type" model="ir.actions.act_window">
<field name="name">Meeting Types</field>
<field name="res_model">calendar.event.type</field>
<field name="view_id" ref="view_calendar_event_type_tree"/>
</record>
<!-- Calendar Alarm : -->
<record id="view_calendar_alarm_tree" model="ir.ui.view">
<field name="name">calendar.alarm.list</field>
<field name="model">calendar.alarm</field>
<field name="arch" type="xml">
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25



Ejemplo de ORM:

Cada clase que hereda de models.Model representa una tabla en PostgreSQL.

Los fields.* son las columnas; los decoradores @api.* definen el comportamiento reactivo.

- fields.Char → VARCHAR
- fields.Integer → INTEGER
- fields.Selection → VARCHAR restringido a opciones
- fields.Many2one → clave foránea (INTEGER con FK)
- compute= → campo calculado (con store=True se persiste en BD)

calendar_views.xml | calendar_alarm.py | main.py

addons > calendar > controllers > main.py > CalendarController

```python
# Cada método decorado con @http.route es un endpoint accesible desde el navegador o API.

import odoo.http as http
from odoo.http import request
from odoo.tools.misc import get_lang

class CalendarController(http.Controller):

    # @http.route: registra la URL y su comportamiento.
    #   type='http'  → responde con HTML (formularios, redirecciones)
    #   type='jsonrpc' → responde con JSON (llamadas desde JavaScript/RPC)
    #   auth='user'    → requiere sesión iniciada
    #   auth='public'  → accesible sin login
    #   auth='calendar'→ autenticación por token (usada en emails de invitación)

    # YTI Note: Keep id and kwargs only for retrocompatibility purpose
    @http.route('/calendar/meeting/accept', type='http', auth="calendar")
    def accept_meeting(self, token, id, **kwargs):
        attendee = request.env['calendar.attendee'].sudo().search([
            ('access_token', '=', token),
            ('state', '!=', 'accepted')])
        attendee.do_accept()
        return self.view_meeting(token, id)

    @http.route('/calendar/recurrence/accept', type='http', auth="calendar")
    def accept_recurrence(self, token, id, **kwargs): ···

    @http.route('/calendar/meeting/decline', type='http', auth="calendar")
    def decline_meeting(self, token, id, **kwargs): ···

    @http.route('/calendar/recurrence/decline', type='http', auth="calendar")
    def decline_recurrence(self, token, id, **kwargs): ···

    @http.route('/calendar/meeting/view', type='http', auth="calendar")
    def view_meeting(self, token, id, **kwargs): ···

    @http.route('/calendar/meeting/join', type='http', auth="user", website=True)
    def calendar_join_meeting(self, token, **kwargs):
        event = request.env['calendar.event'].sudo().search([
```

Ejemplo de Controllers:

type=

- 'http' → responde con HTML (formularios, redirecciones)
- 'jsonrpc' → responde con JSON (llamadas desde JS/RPC)

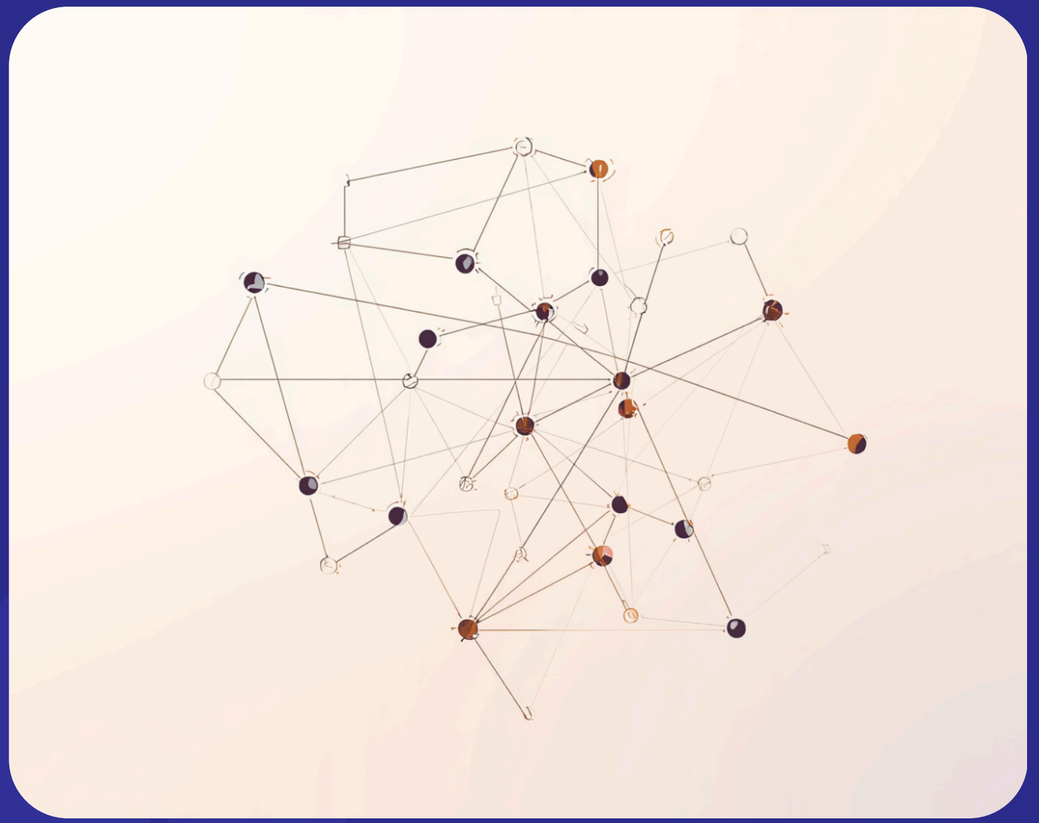
auth=

- 'user' → requiere sesión iniciada
- 'public' → accesible sin login
- 'calendar' → autenticación por token (emails de invitación)

En FastAPI eres el conductor. En Odoo eres el copiloto.

Dependencias Ocultas

El Espiral Invisible



El N1 silencioso

El problema silencioso del N+1

demo.py

```python
# Lo que parece inocente
for order in self.env['sale.order'].search([]):
    print(order.partner_id.name)  # N queries extra

# Lo que deberías hacer
orders = self.env['sale.order'].search([])
orders.mapped('partner_id.name')  # 1 query con JOIN
```

Problemas de lecturas

Leer un campo o llamar a write dentro de un loop puede generar decenas de queries sin que te des cuenta. Es crucial entender esta dinámica para evitar cuellos de botella.

Prefetch en contextos

El uso de prefetch puede desaparecer en contextos inesperados, y Odoo no te avisa. Esto puede resultar en un rendimiento deficiente si no se gestiona adecuadamente.

El Espiral del Crecimiento

01 Dependencias

A medida que el sistema se expande, las dependencias entre módulos se vuelven invisibles, complicando la gestión y el mantenimiento del ERP, lo que puede llevar a errores inesperados.

02 Mantenimiento

Sin una atención adecuada, el crecimiento del sistema puede desbordarse, creando un entorno que requiere **dedicación constante** y no se resuelve simplemente aumentando los recursos del servidor.

Transacciones y bloqueos

Entendiendo el costo de las transacciones

03 Costo oculto

Los onchanges encadenados pueden multiplicar el costo sin ser obvios en el profiler, creando cuellos de botella que pueden ser difíciles de diagnosticar y resolver en producción.

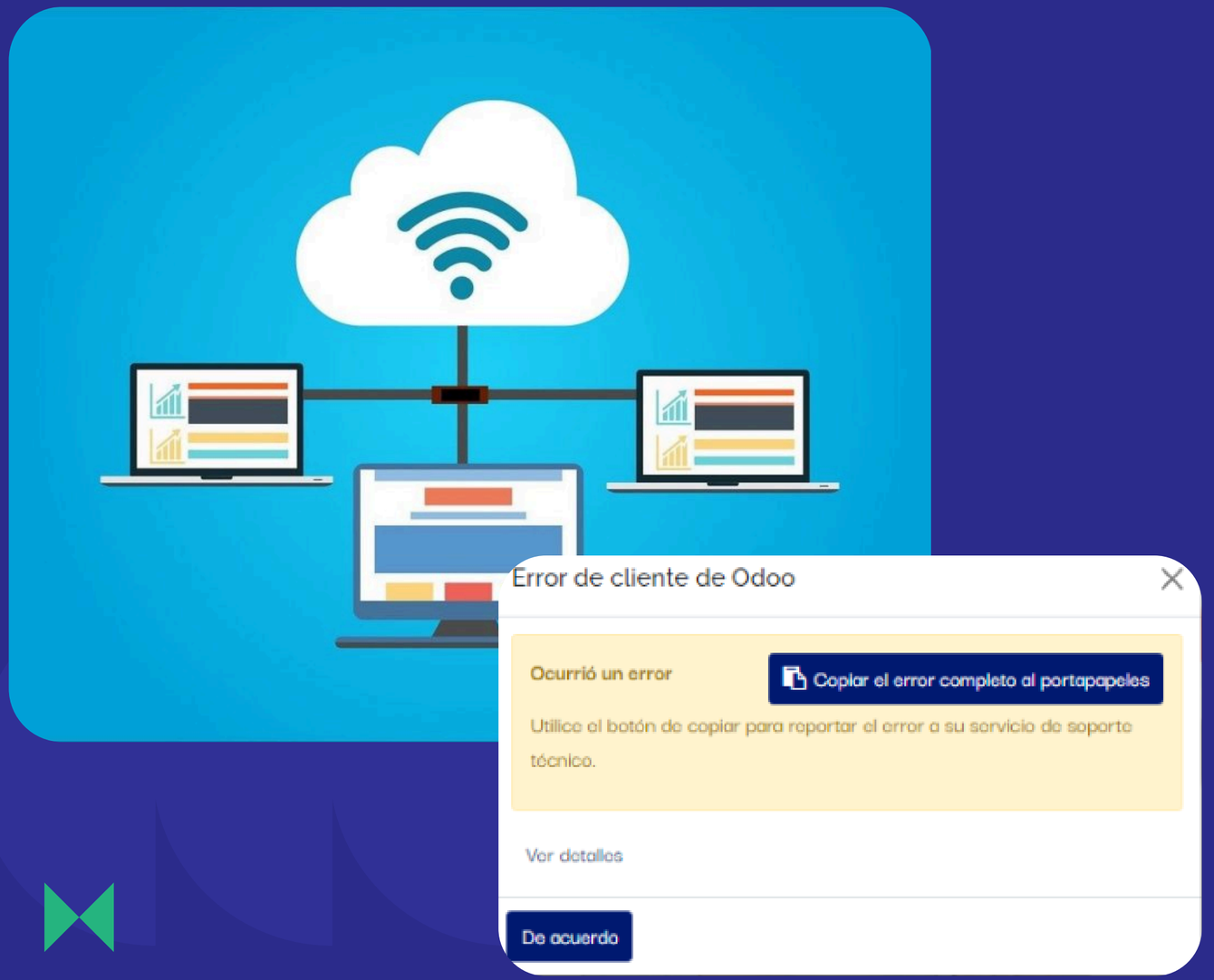
04 Bloqueos reales

Las transacciones largas en PostgreSQL pueden causar bloqueos reales que afectan el rendimiento del sistema. Esto sucede cuando múltiples procesos intentan acceder a los mismos recursos simultáneamente.

Un módulo rompe todo

En pequeño todo bien pero escalar...

Sin una atención adecuada, el crecimiento del sistema puede desbordarse, creando un entorno que requiere dedicación constante y no se resuelve simplemente aumentando los recursos del servidor.



Ahora toca migrar

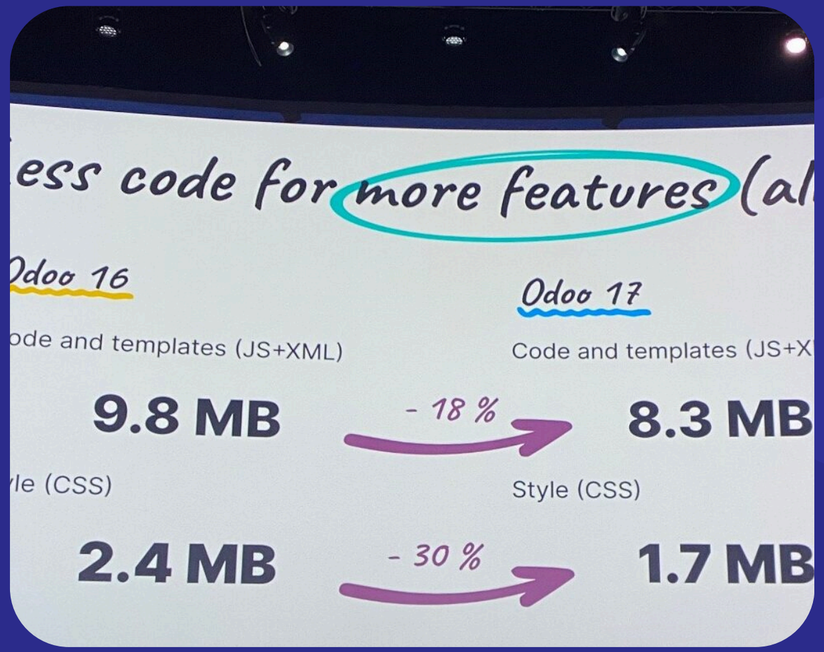
16 → 17 → 18 → 19

Bajo la premisa de

Aprovechar mejoras de rendimiento, tipado y concurrencia

Parte de la prueba de fuego

Los modulos que ocupen funcionen especificas pueden romperse, impacto de relaciones de ORM

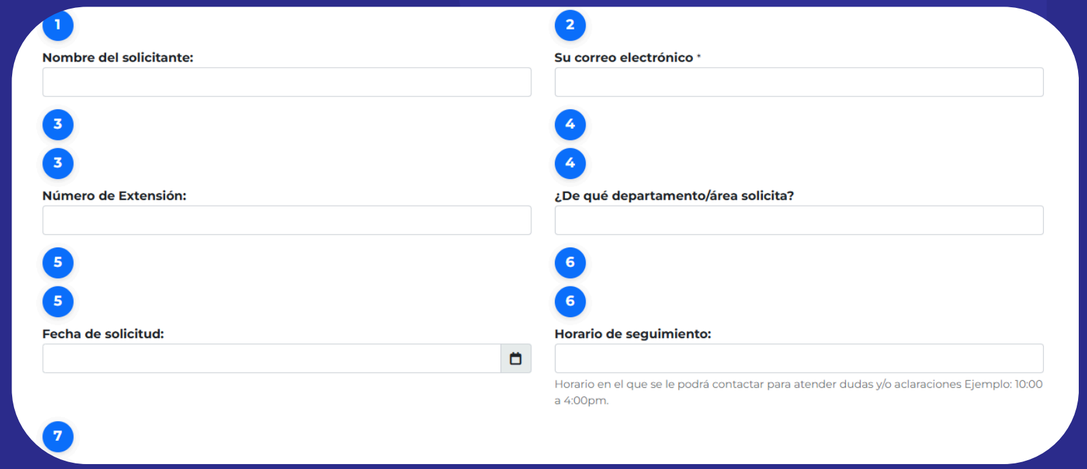


Odoo	Python	PostgreSQL
17	3.10	15
18	3.11	16
19	3.12	17

Ejecución sin control adecuado

Server Actions
Acciones automatizadas que pueden entrar en conflicto y loops infinitos

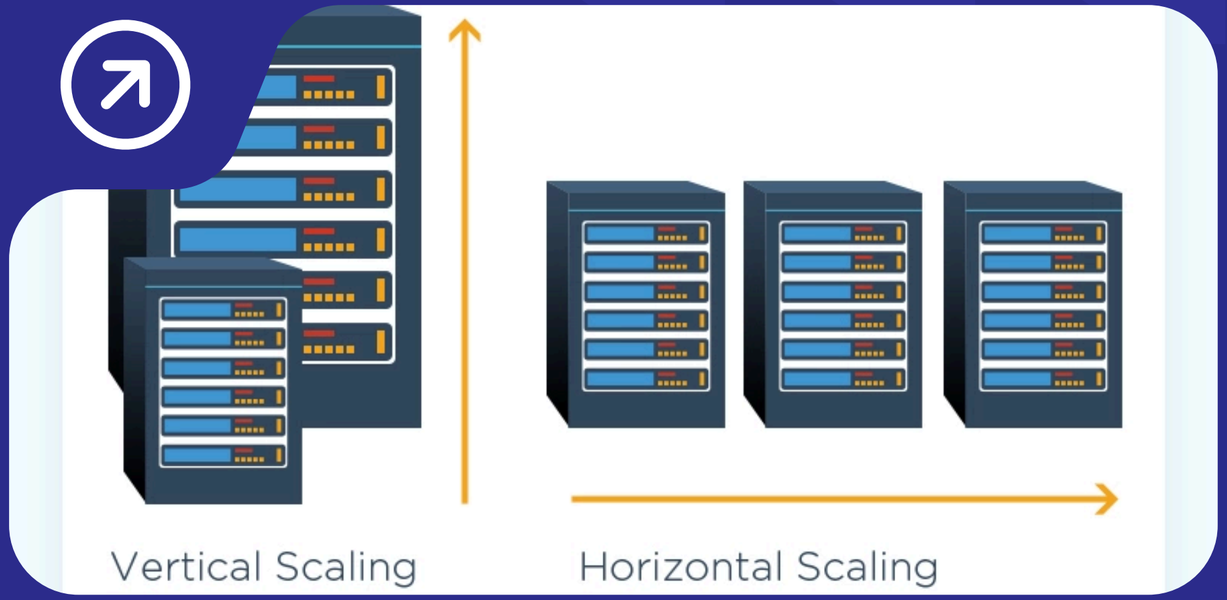
Bugs de QWEB
Inyección de bloques por cada edición sobre los



Hay problemas que ni a billetazos...

Barras Barras

La carga de trabajo en escenarios grandes puede solucionarse con escalamiento vertical, pero llega un punto donde se vuelve inviable...



¿Qué hacer cuando Odoo rechina?

El verdadero mas vale prevenir, consideraciones para escalar

Recomendaciones Generales

Multi-worker

Una configuración inadecuada de multi-worker puede generar conflictos y afectar el rendimiento del sistema, dificultando la gestión de usuarios concurrentes y llevando a tiempos de respuesta lentos.

Longpolling

Longpolling, si no se configura correctamente, puede causar una carga excesiva en el servidor, lo que resulta en un mal rendimiento y una mala experiencia para el usuario final.

Recomendaciones Generales

Tuning de PostgreSQL

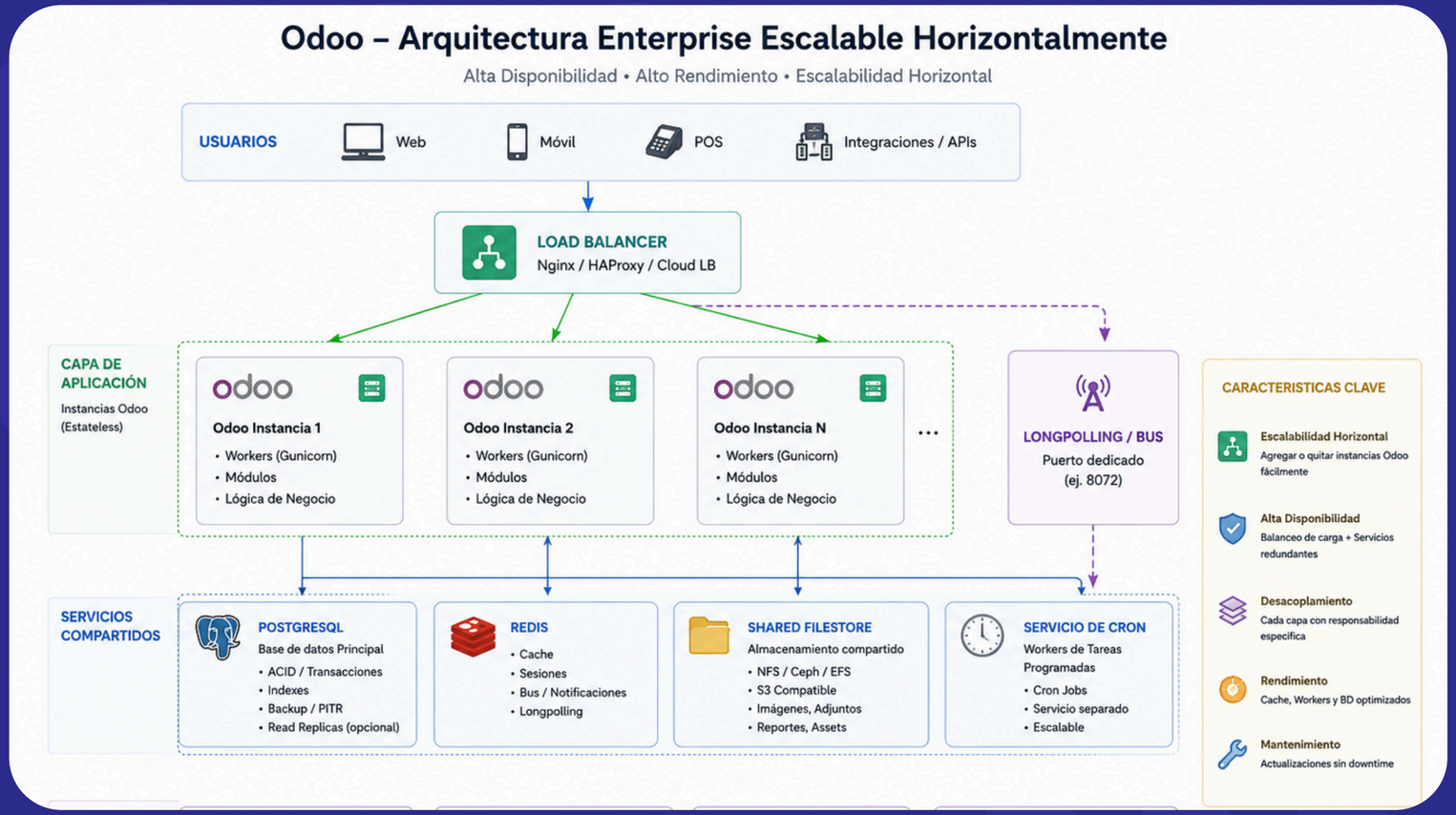
La externalización de procesos pesados, permitiendo reducir carga, mejorar tiempos de respuesta y aumentar la concurrencia del sistema.

Descentralización

Mover tareas intensivas a servicios externos, evitando saturación de workers y mejorando la estabilidad antes de escalar horizontalmente.

Estructura desacoplada

Es posible pasar monolito modular a una arquitectura distribuida donde múltiples nodos de aplicación trabajan sobre servicios centralizados, facilitando alta concurrencia, tolerancia a fallos y crecimiento escalable.



Conclusiones

Enfoque

Tratar Odoo como un **framework serio** en lugar de un simple ERP te ayudará a desarrollar mejores soluciones, optimizar procesos y mantener el control sobre tus proyectos empresariales.

Ventaja

Conocer bien Odoo te permitirá **sacar el máximo provecho** de sus funcionalidades y evitar errores comunes que podrían costar tiempo y recursos.

Conclusiones

Realidades objetivas

Aspecto	Dato
Curva de aprendizaje	Alta — ORM propio, IoC, XML + Python mezclados
Versión gratuita	Community (open source, LGPL-3)
Stack core	Python 3.10–3.12 + PostgreSQL 15–17
Módulos disponibles	Miles, desde contabilidad hasta e-commerce

Conclusiones

Realidades objetivas

Ideal cuando	Considera otra opción si...
Necesitas un sistema integrado rápido	Tu único problema es una API pequeña
Quieres Python en backend empresarial	Necesitas escalar masivamente desde el día 1
Tienes equipo técnico para customizar	No tienes quien mantenga el stack
Quieres open source sin vendor lock-in	El presupuesto no cubre un Odoo developer

Odoo no es magia — es un framework serio que exige entenderlo a fondo. Pero si lo dominas, tienes una ventaja real

Preguntas

eliezermartinez

eliezermartinez.netlify.app

¡Gracias por su atención!