

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

Python en un ERP

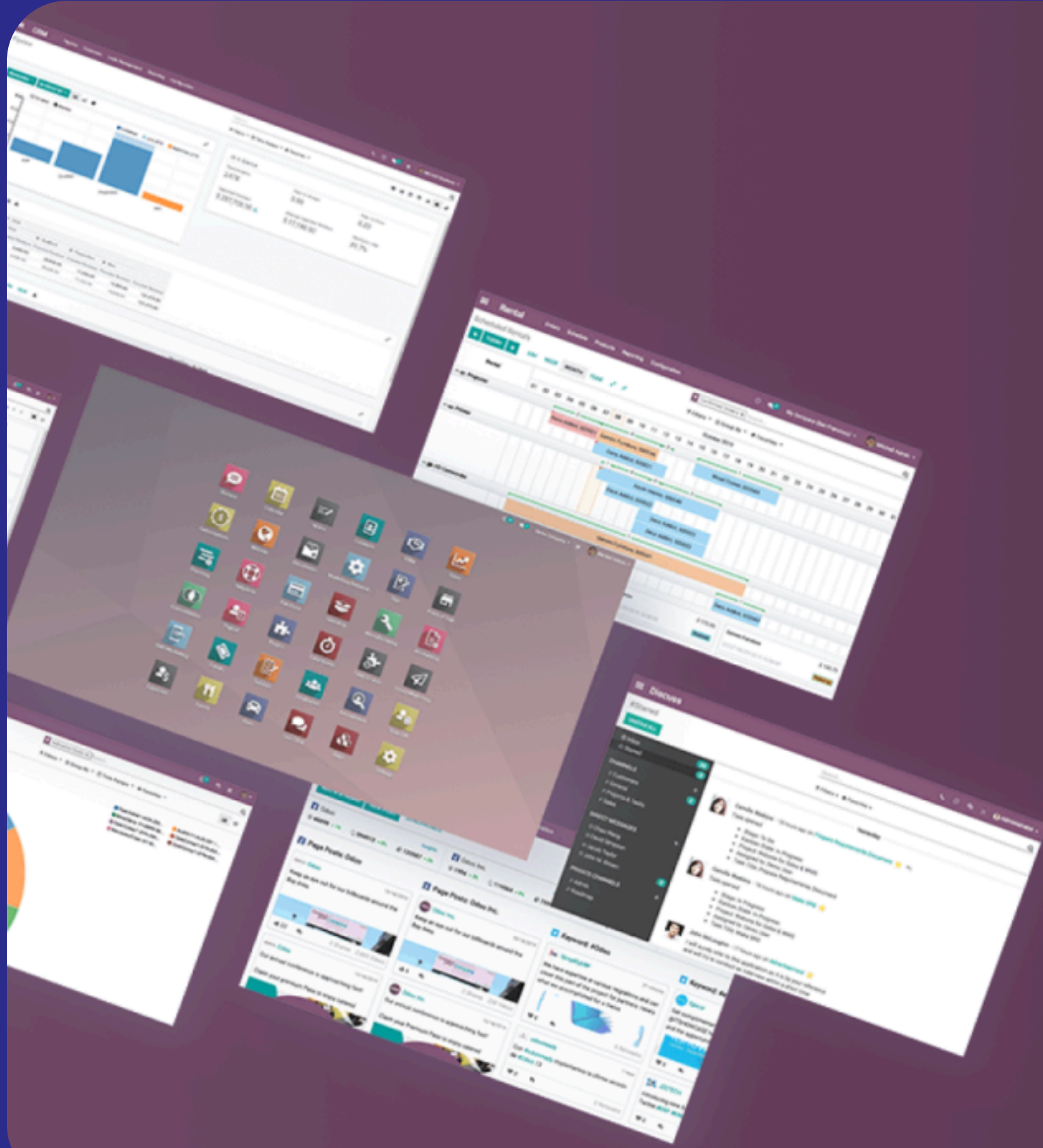


¿Qué es Odoo?



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.



¿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?

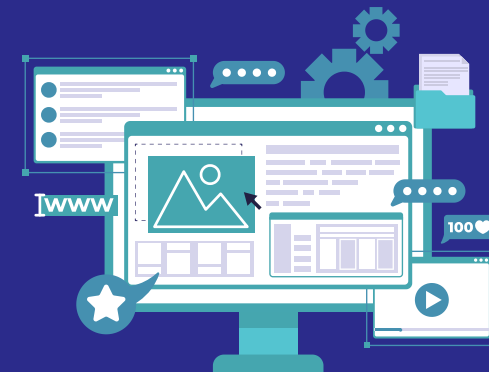
Arquitectura base reutilizable

Define estructura, ciclo de vida y organización de aplicaciones



Motor de Plantillas

QWeb combina HTML y XML eficazmente



Propio (ORM) integrado.

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



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:



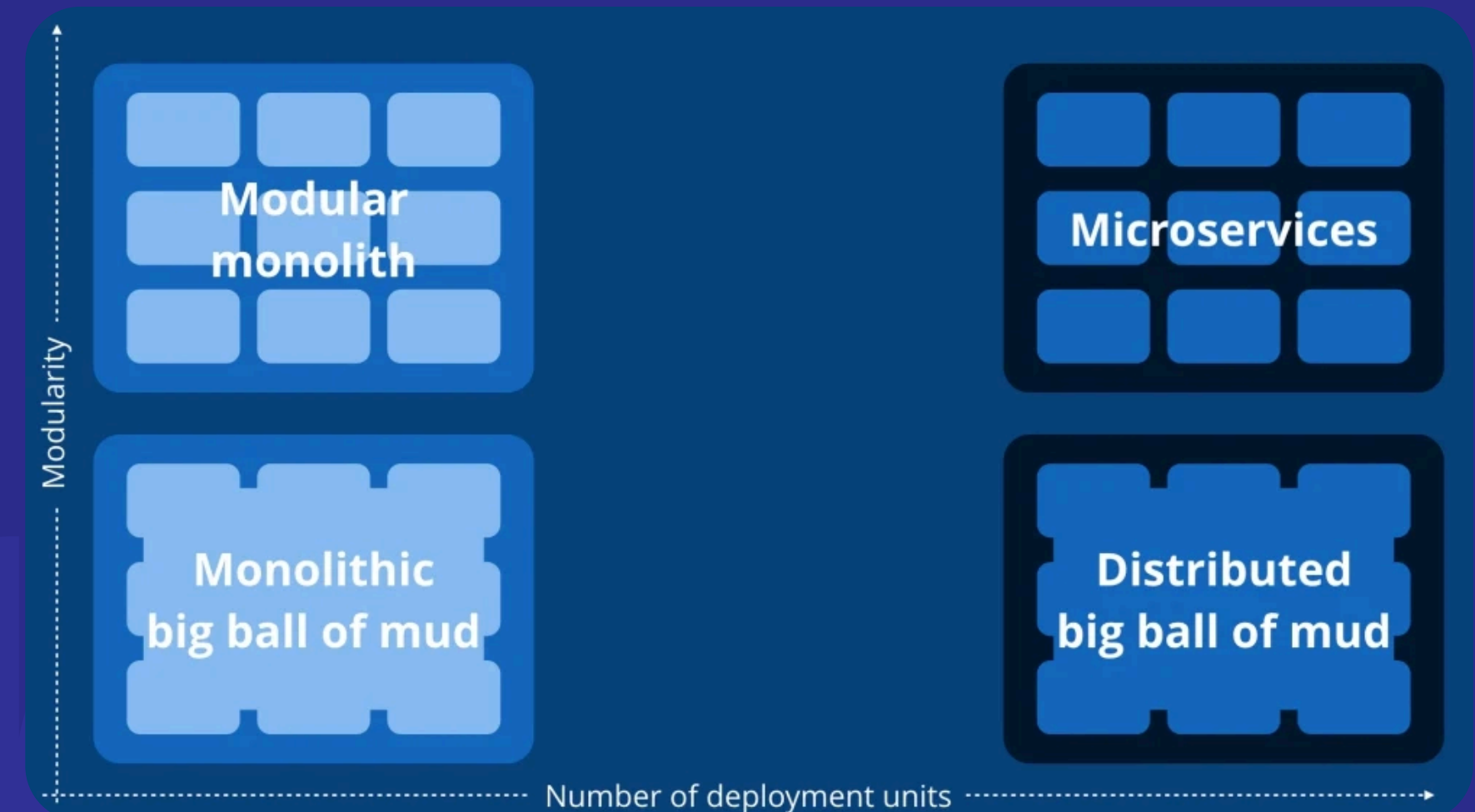
Monolito modular

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

Python en un ERP



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



```
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
      └─ cloud_storage
         ├── cloud_storage_azure
         ├── cloud_storage_google
         └── cloud_storage_migration
      └─ contacts
      └─ crm
      └─ crm_iap_enrich
```

```
calendar_views.xml
1 <?xml version="1.0"?>
2 <odoo>
3
4 <!-- Calendar Events Types : Views and Actions -->
5 <record id="view_calendar_event_type_tree" model="ir.ui.view">
6   <field name="name">calendar.event.type</field>
7   <field name="model">calendar.event.type</field>
8   <field name="arch" type="xml">
9     <list string="Meeting Types" sample="1" editable="bottom">
10      <field name="name"/>
11    </list>
12  </field>
13 </record>
14
15 <record id="action_calendar_event_type" model="ir.actions.act_window">
16   <field name="name">Meeting Types</field>
17   <field name="res_model">calendar.event.type</field>
18   <field name="view_id" ref="view_calendar_event_type_tree"/>
19 </record>
20
21 <!-- Calendar Alarm : -->
22 <record id="view_calendar_alarm_tree" model="ir.ui.view">
23   <field name="name">calendar.alarm.list</field>
24   <field name="model">calendar.alarm</field>
25   <field name="arch" type="xml">
26     <list string="Calendar Alarm" sample="1">
27       <field name="name" column_invisible="True"/>
28       <field name="alarm_type"/>
29       <field name="duration"/>
30       <field name="interval"/>
31     </list>
32   </field>
33 </record>
```

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

```
1 <?xml version="1.0"?>
2 <odoo>
3
4 <!-- Calendar Events Types : Views and Actions -->
5 <record id="view_calendar_event_type_tree" model="ir.ui.view">
6     <field name="name">calendar.event.type</field>
7     <field name="model">calendar.event.type</field>
8     <field name="arch" type="xml">
9         <list string="Meeting Types" sample="1" editable="bottom">
10             <field name="name"/>
11         </list>
12     </field>
13 </record>
14
15 <record id="action_calendar_event_type" model="ir.actions.act_window">
16     <field name="name">Meeting Types</field>
17     <field name="res_model">calendar.event.type</field>
18     <field name="view_id" ref="view_calendar_event_type_tree"/>
19 </record>
20
21 <!-- Calendar Alarm : -->
22 <record id="view_calendar_alarm_tree" model="ir.ui.view">
23     <field name="name">calendar.alarm.list</field>
24     <field name="model">calendar.alarm</field>
25     <field name="arch" type="xml">
```

```
EXPLORER
...
ODOO
addons
  calendar
    controllers
    data
    i18n
    models
      __init__.py
      calendar_alarm_manager.py
      calendar_alarm.py
      calendar_attendee.py
      calendar_event_type.py
      calendar_event.py
      calendar_filter.py
      calendar_recurrence.py
      discuss_channel.py
      ir_http.py
      mail_activity_mixin.py
      mail_activity_type.py
      mail_activity.py
      res_partner.py
      res_users_settings.py
      res_users.py
      utils.py
    security
      calendar_security.xml
      ir.model.access.csv
    static
    tests
    views
    wizard
      __init__.py
      __manifest__.py
  > OUTLINE
  > TIMELINE

calendar_views.xml
calendar_alarm.py
CalendarAlarm
1 # ORM de Odoo: cada clase que hereda de models.Model representa una tabla en PostgreSQL.
2 # Los fields.* son las columnas; los decoradores @api.* definen el comportamiento reactivo.
3
4 from odoo import api, fields, models, _
5 from odoo.fields import Domain
6
7 class CalendarAlarm(models.Model):
8     # _name → nombre técnico del modelo; Odoo crea la tabla reemplazando '.' por '_'
9     # _description → etiqueta legible (UI y logs)
10    _name = 'calendar.alarm'
11    _description = 'Event Alarm'
12
13    # Atributo de clase normal (no es un campo, no genera columna en BD)
14    _interval_selection = {'minutes': 'Minutes', 'hours': 'Hours', 'days': 'Days'}
15
16    # --- Campos básicos -----
17    # fields.Char → VARCHAR | required=True → NOT NULL en BD
18    name = fields.Char('Name', translate=True, required=True)
19
20    # fields.Selection → VARCHAR restringido a una lista de tuplas (valor, etiqueta)
21    alarm_type = fields.Selection(
22        [('notification', 'Notification'), ('email', 'Email')],
23        string='Type', required=True, default='email')
24
25    # fields.Integer → INTEGER | default establece el valor al crear un registro
26    duration = fields.Integer('Remind Before', required=True, default=1)
27
28    interval = fields.Selection(
29        list(_interval_selection.items()), 'Unit', required=True, default='hours')
30
31    # Campo computado con store=True → se persiste en BD (permite filtrar con SQL).
32    # search= es necesario cuando store=False para poder usarlo en dominios de búsqueda.
33    duration_minutes = fields.Integer(
34        'Duration in minutes', store=True,
35        search='_search_duration_minutes', compute='_compute_duration_minutes')
36
37    # Many2one → clave foránea (INTEGER con el id del registro relacionado).
38    # compute + readonly=False + store=True: calculado pero el usuario puede editarlo.
39    mail_template_id = fields.Many2one(
40        'mail_template', string="Email Template"
```

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
1 # Cada método decorado con @http.route es un endpoint accesible desde el navegador o API.
2
3 import odoo.http as http
4 from odoo.http import request
5 from odoo.tools.misc import get_lang
6
7 class CalendarController(http.Controller):
8
9     # @http.route: registra la URL y su comportamiento.
10    # type='http' → responde con HTML (formularios, redirecciones)
11    # type='jsonrpc' → responde con JSON (llamadas desde JavaScript/RPC)
12    # auth='user' → requiere sesión iniciada
13    # auth='public' → accesible sin login
14    # auth='calendar' → autenticación por token (usada en emails de invitación)
15
16    # YTI Note: Keep id and kwargs only for retrocompatibility purpose
17    @http.route('/calendar/meeting/accept', type='http', auth="calendar")
18    def accept_meeting(self, token, id, **kwargs):
19        attendee = request.env['calendar.attendee'].sudo().search([
20            ('access_token', '=', token),
21            ('state', '!=', 'accepted')])
22        attendee.do_accept()
23        return self.view_meeting(token, id)
24
25    @http.route('/calendar/recurrence/accept', type='http', auth="calendar")
26    def accept_recurrence(self, token, id, **kwargs): ...
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41    @http.route('/calendar/meeting/decline', type='http', auth="calendar")
42    def decline_meeting(self, token, id, **kwargs): ...
43
44
45
46
47
48
49    @http.route('/calendar/recurrence/decline', type='http', auth="calendar")
50    def decline_recurrence(self, token, id, **kwargs): ...
51
52
53
54
55
56
57
58
59
60
61
62
63    @http.route('/calendar/meeting/view', type='http', auth="calendar")
64    def view_meeting(self, token, id, **kwargs): ...
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94    @http.route('/calendar/meeting/join', type='http', auth="user", website=True)
95    def calendar_join_meeting(self, token, **kwargs):
96        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
1 # Lo que parece inocente
2 for order in self.env['sale.order'].search([]):
3     print(order.partner_id.name) # N queries extra
4
5 # Lo que deberías hacer
6 orders = self.env['sale.order'].search([])
7 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.



Error de cliente de Odoo

Ocurrió un error [Copiar el error completo al portapapeles](#)

Utilice el botón de copiar para reportar el error a su servicio de soporte técnico.

[Ver detalles](#)

[De acuerdo](#)



Ahora toca migrar

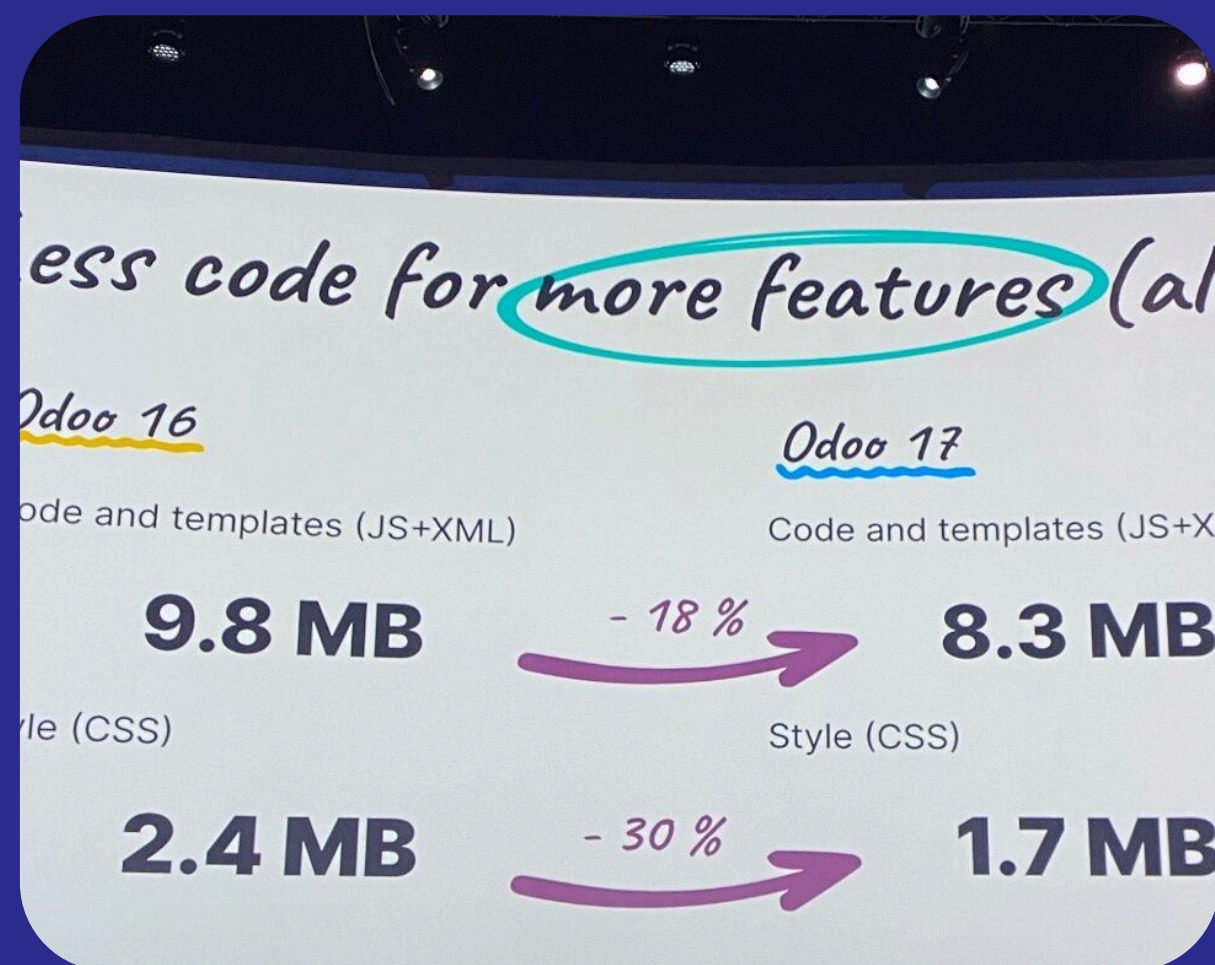
16 → 17 → 18 → 19

☀ Bajo la premisa de

Aprovechar mejoras de rendimiento, tipado y concurrencia

☀ Parte de la prueba de fuego

Los modulos que ocupen funciones 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



```
Al actualizar
Campos de activación?
Antes de actualizar el dominio? Emparejar todos los registros
476572 REGISTRO(S) EDITAR DOMINIO
Aplicar en? Emparejar todos los registros
476572 REGISTRO(S) EDITAR DOMINIO

Código Python
Ayuda

1
2
3 def normalize(text):
4     result = ''
5     for ch in text.lower():
6         result += ACCENT_MAP.get(ch, ch)
7     return result
8
9 def get_nombre(description):
10    if not description:
11        return 'Colaborador(a)'
12    for line in description.splitlines():
13        line = line.strip()
14        if not line:
15            continue
16        if '___' in line or 'Otra información' in line or 'Otra informacion' in line:
17            break
18        if ':' in line:
19            continue
20        return line
21    for line in description.splitlines():
22        if 'Nombre del solicitante' in line and ':' in line:
23            valor = line.split(':', 1)[-1].strip().rstrip(' ').strip()
24            if valor:
```

1 Nombre del solicitante:

2 Su correo electrónico *

3

3

4 Número de Extensión:

4 ¿De qué departamento/área solicita?

5

5

6 Fecha de solicitud:

6 Horario de seguimiento:

Horario en el que se le podrá contactar para atender dudas y/o aclaraciones Ejemplo: 10:00 a 4:00pm.



Hay problemas que ni a билетazos...

Barras Barras

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



Vertical Scaling

Horizontal Scaling



¿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.



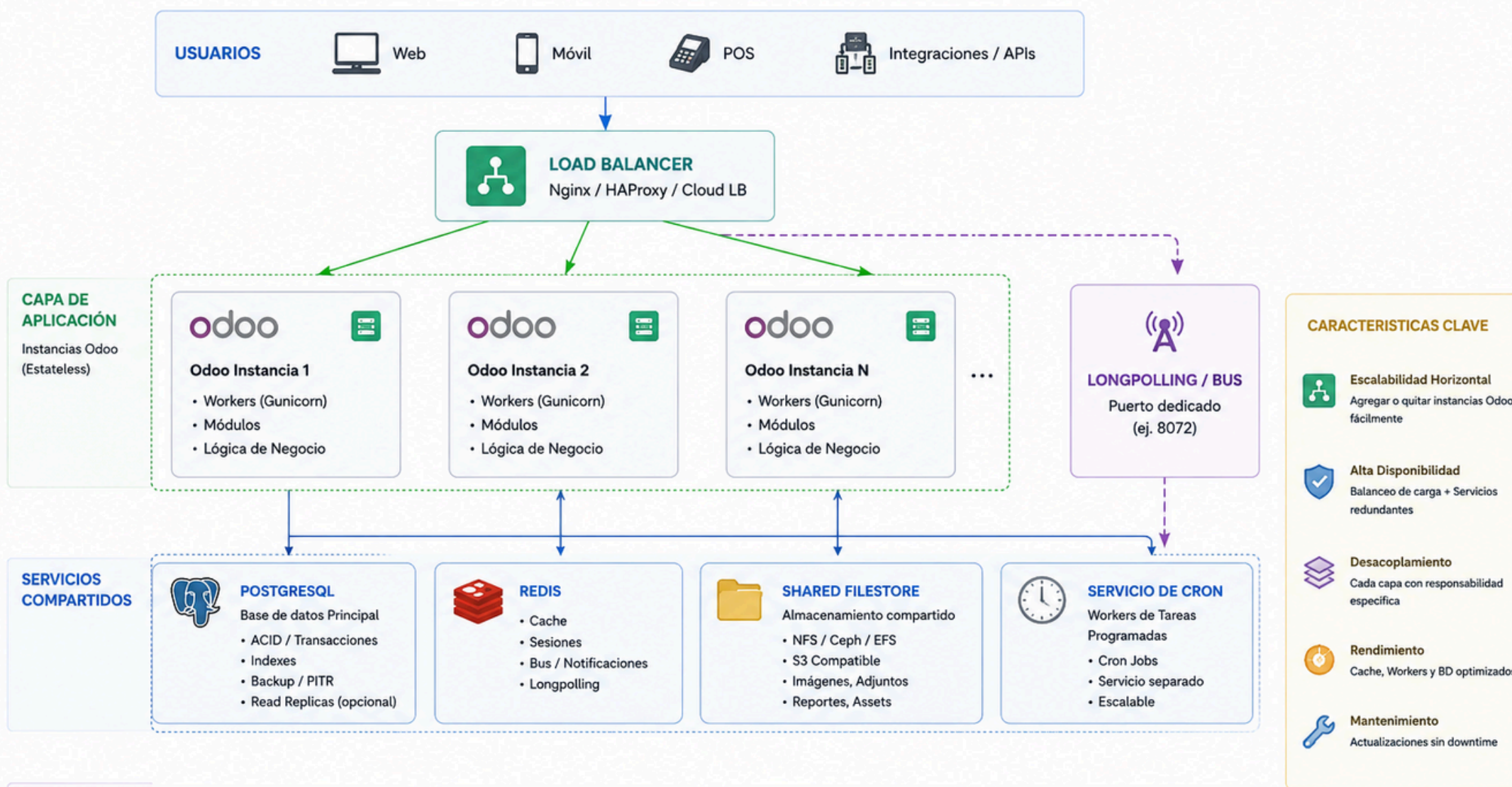
Estructura desacoplada

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.



Odoo – Arquitectura Enterprise Escalable Horizontalmente

Alta Disponibilidad • Alto Rendimiento • Escalabilidad Horizontal



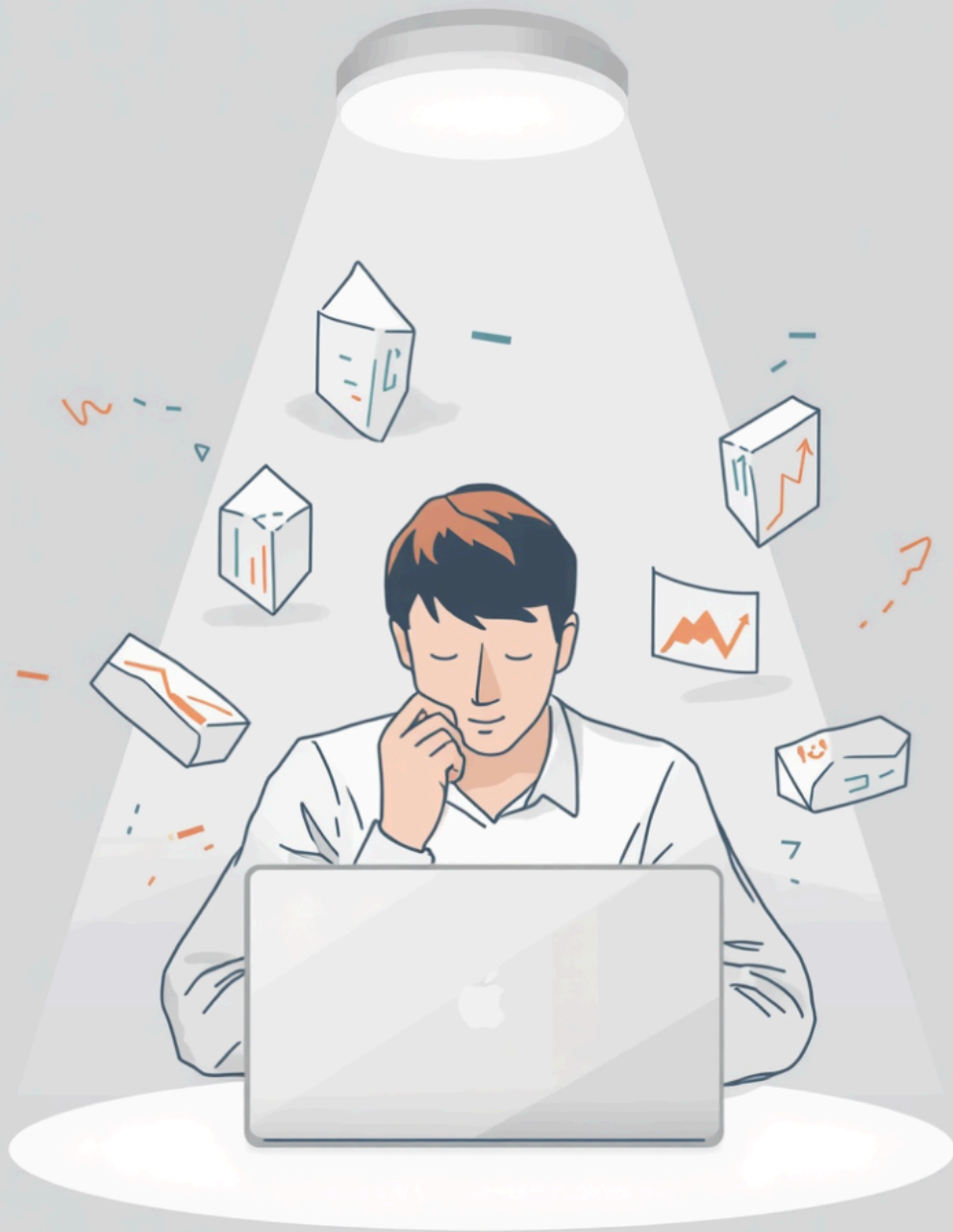
Conclusiones

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



eliezmartinez



eliezmartinez.netlify.app



**¡Gracias por su
atención!**