



## Proyecto Integrador

Matías Cedeño, Matheo Chávez, Anthony Cochea y Gabriel Erazo

Integración de Sistemas

Facultad de Ingeniería y Ciencias Aplicadas, Universidad De Las Américas

12 de julio 2025

### 1. Introducción y Justificación

La gestión eficiente de la información en el ámbito de la salud es un componente esencial para garantizar la calidad de atención, la seguridad del paciente y la sostenibilidad operativa de las instituciones. En la Clínica Universitaria, la existencia de sistemas de información fragmentados ha derivado en una serie de problemas críticos como la duplicidad de datos, demoras en los procesos administrativos y clínicos, y una experiencia de usuario deficiente tanto para pacientes como para el personal médico y administrativo. Esta situación no solo compromete la integridad y consistencia de la información médica, sino que también afecta la eficiencia institucional y la confianza de los usuarios.

Ante este escenario, el presente proyecto propone el diseño e implementación de una solución de integración basada en patrones modernos de arquitectura y en el uso de tecnologías open source. El objetivo es consolidar los distintos sistemas existentes en un ecosistema digital unificado, resiliente y seguro, que permita optimizar los flujos de trabajo, centralizar la información y mejorar significativamente la experiencia del usuario. Para ello, se emplean herramientas como Odoo ERP con un módulo personalizado para odontología, Zammad para la gestión de tickets, Nextcloud como repositorio documental, RabbitMQ como sistema de mensajería, Clerk como

proveedor de autenticación (OAuth2), y un Mailer Service desarrollado en Node.js, que utiliza el servicio de envío de correos Resend. Además, se integran herramientas de monitoreo y trazabilidad como Loki, Grafana y Prometheus, todas desplegadas en contenedores Docker.

Este proyecto se enmarca dentro del Resultado de Aprendizaje de la Carrera RC2, el cual establece la necesidad de aplicar principios de ingeniería para desarrollar soluciones que respondan a necesidades específicas del entorno. En este contexto, se han considerado múltiples factores que refuerzan la pertinencia de la propuesta:

- **Salud pública**, al mejorar la gestión de citas y prescripciones con menor margen de error;
- **Seguridad**, mediante autenticación centralizada y control de acceso basado en roles;
- **Bienestar**, optimizando la experiencia de usuario a través de interfaces unificadas;
- **Factores globales y económicos**, con una arquitectura escalable y basada en tecnologías libres;
- **Factores culturales y sociales**, al ofrecer interfaces accesibles y mejorar la interacción con los usuarios;
- **Factores ambientales**, al reducir el impacto asociado al uso de licencias comerciales.

De esta forma, la integración planteada no solo busca resolver los problemas técnicos identificados, sino también establecer una base tecnológica robusta que permita a la Clínica Universitaria evolucionar hacia un modelo de atención más ágil, seguro y centrado en el usuario.

## 2. Análisis de Problemas Reales Identificados

El diagnóstico de la situación actual en la Clínica Universitaria revela una serie de problemas estructurales e interconectados, derivados principalmente de la ausencia de una integración entre los distintos sistemas tecnológicos en uso. Esta fragmentación ha generado deficiencias operativas que afectan tanto la calidad del servicio como la seguridad y satisfacción de los usuarios.

- **Duplicidad de Datos:** La coexistencia de sistemas aislados provoca la generación redundante de información clave, como registros de pacientes, citas y prescripciones.

Esta repetición no solo representa un uso ineficiente de recursos, sino que también incrementa el riesgo de inconsistencias clínicas, errores diagnósticos y fallas en la trazabilidad del tratamiento.

- **Demoras en los Procesos:** La falta de sincronización entre plataformas conlleva a retrasos innecesarios en tareas fundamentales como la creación de citas médicas, el registro de prescripciones odontológicas o la resolución de incidentes administrativos. Este entorno obliga al personal a operar en múltiples sistemas de forma manual, generando ineficiencia y desgaste operativo.
- **Experiencia del Usuario Deficiente:** Pacientes, médicos y administradores interactúan con una infraestructura fragmentada que no ofrece un acceso centralizado ni autenticación unificada. Esta situación genera una experiencia frustrante: el paciente carece de un portal único para gestionar sus consultas y acceder a sus documentos, mientras que el personal debe navegar múltiples interfaces sin una visión completa del historial clínico o administrativo.
- **Falta de Notificaciones Automáticas:** Actualmente no existe un mecanismo eficiente para alertar al personal sobre eventos críticos en tiempo real, como la creación de tickets de soporte, agotamiento de inventario o cambios en la programación de citas, lo que retrasa la toma de decisiones y afecta la capacidad de respuesta institucional.
- **Deficiencias en la Seguridad de los Datos:** La ausencia de un sistema de autenticación centralizada y control de acceso estructurado expone los datos sensibles a riesgos de seguridad, comprometiendo la confidencialidad de la información clínica y administrativa.
- **Falta de Trazabilidad:** La inexistencia de monitoreo sistemático, logs unificados y métricas de rendimiento impide verificar la correcta ejecución de las integraciones y dificulta la identificación de fallos, afectando la resiliencia del sistema y la capacidad de mejora continua.

Estos problemas, al no ser tratados, no solo comprometen la calidad del servicio odontológico, sino que también afectan directamente la seguridad de los datos, la eficiencia operativa y la confianza de los usuarios. En este contexto, se plantea una arquitectura de integración como respuesta estratégica para abordar estos desafíos de manera efectiva.

### **3. Solución Técnica Propuesta**

Para solventar los problemas identificados en la Clínica Universitaria, se plantea una solución de integración basada en la orquestación de servicios y la centralización de la seguridad, aplicando patrones modernos de arquitectura y tecnologías open source. La propuesta integra múltiples sistemas mediante APIs, mensajería asíncrona y almacenamiento compartido, garantizando una comunicación eficiente, segura y trazable entre los componentes. Además, se asegura la resiliencia del ecosistema mediante herramientas de monitoreo y despliegue en contenedores Docker.

#### **Sistemas Integrados**

La arquitectura propuesta incorpora seis componentes clave, que forman el núcleo funcional del sistema:

- **Odoo ERP (con módulo de odontología personalizado):** Gestiona procesos clínico-administrativos como registro de pacientes, citas, prescripciones, generación de facturas y reporte de incidentes. Utiliza su API REST para integrarse con Zammad y publica mensajes en RabbitMQ para activar notificaciones. Además, transfiere archivos de prescripción a Nextcloud mediante WebDAV y utiliza Clerk para autenticación centralizada.
- **Zammad:** Actúa como sistema de mesa de ayuda para gestionar incidentes clínicos o administrativos reportados por pacientes o personal. Se comunica con Odoo vía API REST para sincronizar tickets y responde mediante el mismo canal. También transfiere documentos a Nextcloud y autentica usuarios a través de Clerk.
- **Nextcloud:** Sirve como repositorio seguro de documentos médicos y administrativos. Almacena prescripciones, reportes de incidentes y facturas en formatos como PDF o TXT, organizados por paciente. La comunicación se realiza mediante su API WebDAV desde Odoo y Zammad.
- **RabbitMQ:** Se implementa como sistema de mensajería para la publicación de eventos críticos (e.g., creación de tickets, actualizaciones de inventario), permitiendo el envío de notificaciones asíncronas a través del Mailer Service.

- **Clerk:** Gestiona la autenticación de usuarios mediante OAuth2, funcionando como proveedor de identidad (IdP). Genera tokens JWT con scopes específicos, los cuales son validados por Odoo y Zammad para autorizar operaciones según el rol (paciente, médico, administrador).
- **Mailer Service (Node.js):** Servicio que escucha los mensajes publicados en RabbitMQ y envía correos electrónicos automatizados utilizando el proveedor Resend. Notifica a pacientes o administradores sobre eventos relevantes como la creación de un ticket o una actualización en el sistema.

## Patrones de Integración Aplicados

La arquitectura implementa cuatro patrones de integración fundamentales:

- **API RESTful:** Facilita la comunicación síncrona entre Odoo y Zammad. Por ejemplo, cuando un paciente reporta un incidente desde el portal de Odoo, se genera una solicitud POST hacia Zammad para crear un ticket, el cual responde con un ID que queda sincronizado en ambos sistemas. Este patrón mejora la interoperabilidad y reduce duplicidades.
- **Transferencia de Archivos:** Utiliza la API WebDAV de Nextcloud para almacenar de manera estructurada los documentos generados por Odoo y Zammad (como prescripciones, reportes o facturas), organizados por carpetas de paciente. Esta integración asegura la trazabilidad documental y el acceso controlado a información sensible.
- **Mensajería (RabbitMQ):** Permite la emisión de eventos clave desde Odoo hacia RabbitMQ (como la creación de un ticket), los cuales son consumidos por el Mailer Service para enviar notificaciones por correo electrónico. Este enfoque mejora la capacidad de respuesta institucional mediante notificaciones en tiempo real.
- **Seguridad y Autenticación con SSO (Clerk):** Clerk proporciona un sistema de autenticación centralizado basado en OAuth2. Todos los usuarios se autentican a través de esta plataforma, la cual emite tokens JWT que son verificados por los sistemas integrados. Esto permite un control de acceso unificado y seguro, alineado con los principios de privacidad de la información clínica.

## Trazabilidad y Monitoreo

La solución contempla mecanismos de trazabilidad a través de herramientas de observabilidad desplegadas en Docker:

- **Loki:** Recolecta logs de Odoo, Zammad, Nextcloud, RabbitMQ, Mailer Service y Clerk, incluyendo eventos clave como autenticaciones, creación de tickets y errores del sistema.
- **Grafana:** Visualiza los logs y métricas a través de dashboards personalizables, permitiendo la supervisión en tiempo real del estado operativo del sistema.
- **Prometheus:** Recolecta métricas tanto a nivel de infraestructura (usando Node Exporter, cAdvisor) como de servicios específicos (PostgreSQL Exporter, RabbitMQ Exporter), facilitando el análisis de rendimiento y la detección temprana de fallos.

## Bases de Datos

La arquitectura utiliza dos instancias PostgreSQL:

- **clinica\_db:** Base de datos principal compartida por Odoo y Nextcloud, alojada en una instancia contenedora.
- **zammad\_production:** Base de datos dedicada para Zammad, desplegada en una instancia separada para aislar la carga operativa.

## Consideraciones de Diseño

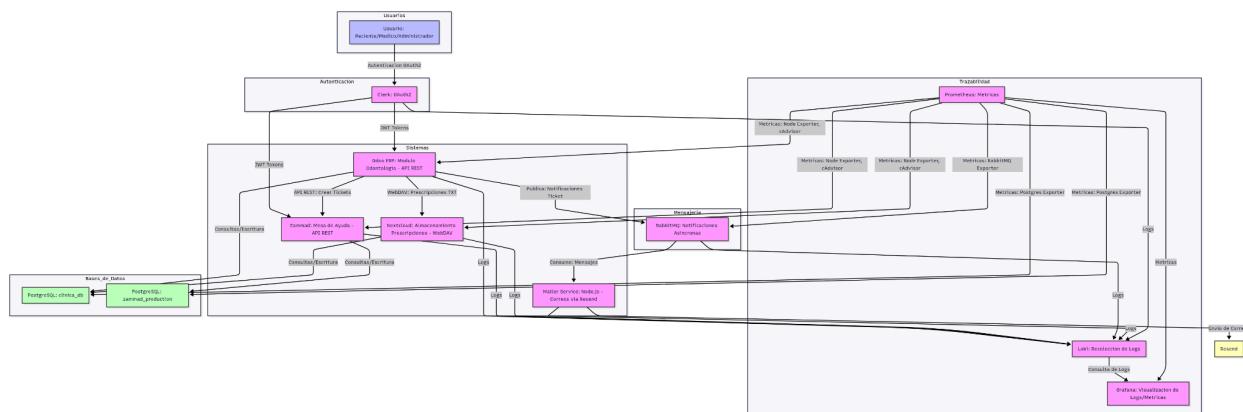
La solución propuesta responde de manera integral a los factores transversales definidos por el proyecto:

- **Salud Pública:** Optimiza la gestión odontológica, reduciendo errores en citas y prescripciones.
- **Seguridad:** Protege los datos sensibles mediante autenticación centralizada con Clerk.
- **Bienestar:** Mejora la experiencia del usuario con interfaces unificadas y notificaciones automáticas.
- **Factores Globales:** Utiliza una arquitectura escalable y portable basada en contenedores Docker.

- **Culturales:** Las interfaces son accesibles para usuarios con distintos niveles de conocimiento técnico.
- **Sociales:** Facilita la interacción entre pacientes y administración mediante sistemas de tickets.
- **Ambientales:** Se basa en herramientas open source, reduciendo el impacto ambiental del uso de software comercial.
- **Económicos:** Representa una solución rentable al evitar licencias propietarias, utilizando herramientas libres y servicios en fase de prueba.

## 4. Arquitectura General

### Diagrama de Arquitectura:



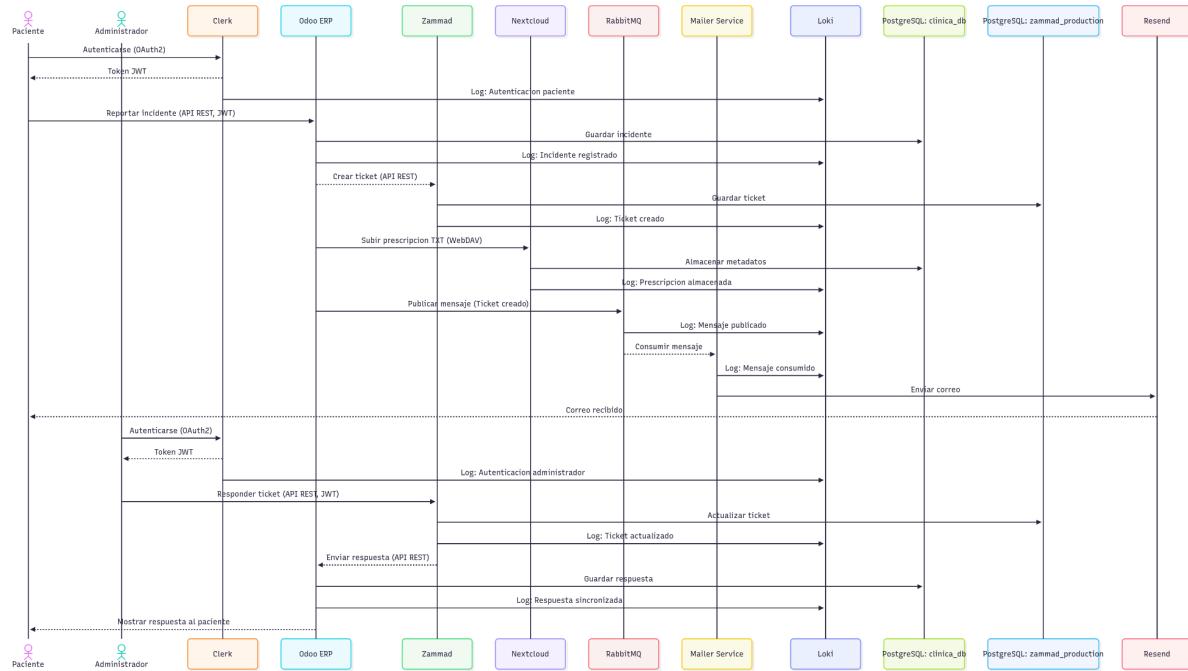
### Explicación del Diagrama:

- **Usuarios:** Pacientes, médicos y administradores acceden a Odoo y Zammad mediante autenticación OAuth2 proporcionada por Clerk.
- **Clerk:** Gestiona la autenticación y entrega tokens JWT para Odoo y Zammad.
- **Odoo ERP:** Ejecuta el módulo de odontología para gestionar pacientes, citas, prescripciones y reportes de incidentes. Envía tickets a Zammad (API REST), prescripciones a Nextcloud (WebDAV), y notificaciones a RabbitMQ.
- **Zammad:** Maneja tickets de incidentes, recibiendo datos desde Odoo y enviando respuestas que se sincronizan con Odoo.
- **Nextcloud:** Almacena prescripciones en formato TXT, organizadas por carpetas de pacientes, usando WebDAV desde Odoo.

- **RabbitMQ:** Publica mensajes sobre tickets creados, que son consumidos por el Mailer Service.
- **Mailer Service:** Servicio en Node.js que consume mensajes de RabbitMQ y envía correos electrónicos a través de Resend.
- **Bases de Datos:**
  - **clinica\_db:** Usada por Odoo y Nextcloud, alojada en la instancia postgres del primer docker-compose.
  - **zammad\_production:** Usada exclusivamente por Zammad, alojada en la instancia zammad-postgresql del segundo docker-compose.
- **Trazabilidad:**
  - **Loki:** Recolecta logs de Odoo, Zammad, Nextcloud, RabbitMQ, Mailer Service, y Clerk.
  - **Grafana:** Visualiza logs de Loki y métricas de Prometheus (recopiladas por Node Exporter, cAdvisor, Postgres Exporter, y RabbitMQ Exporter).
  - **Prometheus:** Recolecta métricas de infraestructura y servicios específicos.
- **Corrección del Error:** Las etiquetas de las bases de datos se simplificaron a clinica\_db y zammad\_production, eliminando los paréntesis que causaban el error de parseo. La asignación (Odoo/Nextcloud para clinica\_db y Zammad para zammad\_production) se aclara en esta explicación.

## Diagrama de Flujo de integración (Mermaid)

El siguiente diagrama de secuencia ilustra un flujo funcional típico: un paciente reporta un incidente en Odoo, se crea un ticket en Zammad, se envía una prescripción a Nextcloud, y se notifica al paciente vía correo usando RabbitMQ y el Mailer Service. Se actualiza para reflejar las bases de datos separadas (clinica\_db para Odoo/Nextcloud y zammad\_production para Zammad).



## Explicación del Flujo:

1. El paciente se autentica en Clerk, recibe un token JWT, y Clerk registra el evento en Loki.
  2. El paciente reporta un incidente en Odoo, que se guarda en clinica\_db y genera un log en Loki.
  3. Odoo envía una solicitud POST a Zammad para crear un ticket, que se guarda en zammad\_production y genera un log.
  4. Odoo sube una prescripción en formato TXT a Nextcloud vía WebDAV, que almacena metadatos en clinica\_db y registra un log.
  5. Odoo publica un mensaje en RabbitMQ sobre el ticket creado, que genera un log.
  6. El Mailer Service consume el mensaje, envía un correo vía Resend, y registra el evento en Loki.
  7. El administrador se autentica en Clerk, responde al ticket en Zammad (guardado en zammad\_production), y la respuesta se sincroniza con Odoo (guardada en clinica\_db).
  8. El paciente ve la respuesta en Odoo.

## Contrato de Api (Swagger)

The screenshot shows the Swagger Editor interface with the following details:

- Left Sidebar (Code Snippets):**
  - Line 785: Unauthorized response with schema and example.
  - Line 793: Unauthorized response with message and example.
  - Line 800: Unauthorized response with message and example.
  - Line 801: Forbidden response with message and example.
  - Line 809: Forbidden response with message and example.
- System Section:**
  - GET /api/v1/info: Información de la API
- Incidents Section:**
  - GET /api/v1/incidents: Listar reportes de incidentes
  - POST /api/v1/incidents: Crear nuevo reporte de incidente
  - GET /api/v1/incidents/{incident\_id}: Obtener detalles de incidente
  - PUT /api/v1/incidents/{incident\_id}: Actualizar incidente
- Patients Section:**
  - GET /api/v1/patients: Listar pacientes
  - GET /api/v1/patients/{patient\_id}: Obtener detalles de paciente
- Appointments Section:**
  - GET /api/v1/appointments: Listar citas

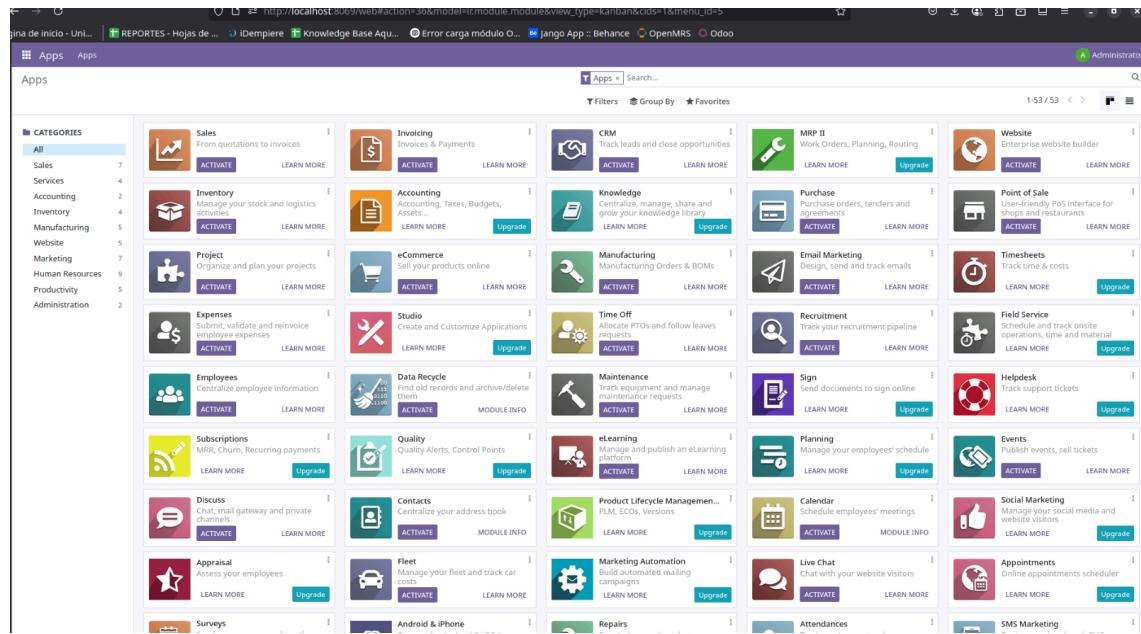
## 5. Capturas de Configuración y funcionamiento

### Configuración de Grafana

The screenshot shows the Grafana configuration interface for a Loki data source. The 'Settings' tab is active, displaying the following configuration:

- Name:** loki
- URL:** http://loki:3100
- Authentication:** No Authentication
- TLS settings:**
  - Add self-signed certificate (radio button selected)
  - SSL Client Authentication (radio button)
  - Skip TLS certificate validation (radio button)

### Configuración de Odoo



## Configure Clerk para SSO, Oddo y Zammand

The screenshot shows the Clerk configuration interface under the 'Integrations' section. The left sidebar includes links for Overview, Users, Organizations, Subscriptions (Beta), and Configure. The 'Configure' link is currently active. The main area is titled 'OAuth applications' and contains the following sections:

- User & Authentication** (with sub-links: Email, phone, username; SSO connections; Web3; Multi-factor; Restrictions; Attack protection)
- OAuth applications** (with sub-links: Dynamic client registration, which is enabled)
  - A search bar and a 'Sort by: Created' dropdown.
  - A 'Add OAuth application' button.
  - A table listing registered OAuth clients:
 

Name	Scopes	Created
z zammand	Public, email, openid, profile	July 8, 2025
o odoo	Public, email, openid, profile	July 8, 2025
- Session management** (with sub-links: Sessions, JWT templates)
- Compliance** (with sub-links: Legal)
- Feature management** (with sub-links: Features)
- Organization management**

zammad

**Scopes**

email  openid  profile

**Public**  
If enabled, the Authorization Code with PKCE (Proof Key for Code Exchange) flow can be used, particularly beneficial for applications that cannot securely store Client Secrets, such as native and mobile apps.

**Consent screen** Recommended  
The consent screen shows users a confirmation dialog before they authorize OAuth applications. This allows users to review the permissions they're granting.  
The consent screen is an important security feature that helps users understand what permissions they're granting to OAuth applications.

Documentation [Learn how to configure](#) [Edit](#)

[Delete application](#)

Created on July 8, 2025  
App updated 3d ago

**Redirect URIs**  
For OAuth requests, the provided URI must exactly match one of the listed URIs. Specify at least one URI for authentication to work.

Enter URI [Add URI](#)

http://localhost:8080/auth/oidc/callback  
http://localhost:8080/auth/openid\_connect/callback

**Application credentials**  
Manage your OAuth 2.0 credentials.

**Client ID**  
TVLyeuSmbjv9yRAe

**Client Secret**  
\*\*\*\*\* [Regenerate](#)

← OAuth applications

**odoo**  
Public

**Basic info**  
Update this application's details

**Name**  
odoo

**Scopes**

email  openid  profile

**Public**  
If enabled, the Authorization Code with PKCE (Proof Key for Code Exchange) flow can be used, particularly beneficial for applications that cannot securely store Client Secrets, such as native and mobile apps.

**Consent screen** Recommended  
The consent screen shows users a confirmation dialog before they authorize OAuth applications. This allows users to review the permissions they're granting.  
The consent screen is an important security feature that helps users understand what permissions they're granting to OAuth applications.

Documentation [Learn how to configure](#) [Edit](#)

[Delete application](#)

Created on July 8, 2025  
App updated 3d ago

**Redirect URIs**  
For OAuth requests, the provided URI must exactly match one of the listed URIs. Specify at least one URI for authentication to work.

Enter URI [Add URI](#)

http://localhost:8069/auth\_oauth/signin

## Configuración del provider en Odoo

The screenshot shows the Odoo interface for managing providers. The top navigation bar includes links like 'de inicio - Uni...', 'REPORTES - Hojas de ...', 'iDempiere', 'California Gold Nutriti...', 'Jango App :: Behance', 'Odoo', 'RabbitMQ Managem...', and a 'T' icon. Below the navigation is a purple header bar with tabs: 'Settings', 'General Settings', and 'Users & Companies'. The main content area has a sub-header 'Settings / Providers' with a 'clerk' entry. A 'New' button is visible. The configuration details are as follows:

Provider name	clerk
Auth Flow	OpenID Connect (authorization code flow)
Token Map ?	e.g from:to upn:email sub:user_id
Client ID	b3Dn5lm7SExf
Client Secret ?	
Allowed	<input checked="" type="checkbox"/>
Login button label ?	clerk
Authorization URL	https://humble-yeti-77.clerk.accounts.dev/oauth/authorize
Scope	openid profile email
Userinfo URL	https://humble-yeti-77.clerk.accounts.dev/oauth/userinfo
Token URL ?	https://humble-yeti-77.clerk.accounts.dev/oauth/token
JWKS URL ?	https://humble-yeti-77.clerk.accounts.dev/.well-known/jwks.json
Data Endpoint	

## Configuración de Roles

The screenshot shows the Odoo interface for role configuration. It is organized into several sections:

- ACCOUNTING**:
  - Invoicing ?
  - Bank
- INVENTORY**:
  - Inventory ?
  - Purchase ?
- WEBSITE**:
  - Website
- HUMAN RESOURCE**:
  - Employees ?
  - Expenses ?
- PRODUCTIVITY**:
  - Dashboard ?
- ADMINISTRATION**:
  - Administration
- OTHER**:
  - Dental Hospital: User
  - Dental Hospital: Manager
  - Dental Hospital: Administrator** (highlighted in a gray box)
  - Dental Hospital ?
  - Dental Hospital: User

## Configuración RabbitMQ

The screenshot shows the RabbitMQ Management Interface's Overview page. At the top, it displays the version: RabbitMQ 3.13.7 Erlang 26.2.5.13 and a refresh timestamp: Refreshed 2025-07-1'. Below the header, there are tabs for Overview, Connections, Channels, Exchanges, Queues and Streams, and Admin. The Overview tab is selected. The main content area is titled "Overview" and contains sections for "Totals" and "Nodes". The "Totals" section includes metrics like Queued messages (last minute), Currently idle, Message rates (last minute), and Global counts. The "Nodes" section lists a single node: rabbit@9753dfc6785b with details such as File descriptors (46), Socket descriptors (0), Erlang processes (445), Memory (167 MiB), Disk space (823 GiB), Uptime (4h 47m), Cores (8), Info (basic 2 rss), and a button to Reset stats. Below the nodes table are sections for Churn statistics, Ports and contexts, Export definitions, and Import definitions. At the bottom of the page, there are links for HTTP API, Documentation, Tutorials, New releases, Commercial edition, Commercial support, Discussions, Discord, Plugins, and GitHub.

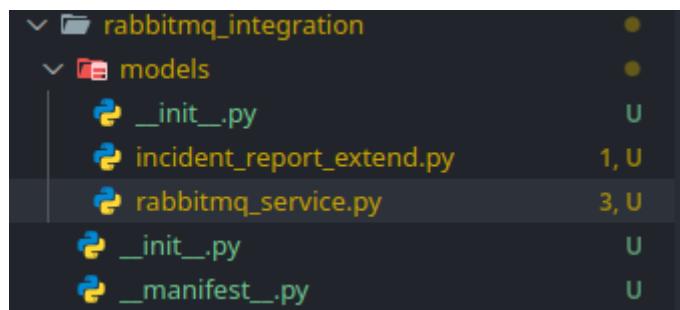
## Añadir cola zammad\_alerts

The screenshot shows the RabbitMQ Management Interface's Queues and Streams page. At the top, it displays the version: RabbitMQ 3.13.7 Erlang 26.2.5.13 and a refresh timestamp: Refreshed 2025-07-1'. Below the header, there are tabs for Overview, Connections, Channels, Exchanges, Queues and Streams, and Admin. The Queues and Streams tab is selected. The main content area is titled "Queues" and shows a table for "All queues (1)". The table has columns for Overview, Messages, and Message rates. It lists one queue: zammad\_alerts under the virtual host /. The queue is of type classic, state running, and has 0 ready, 0 unacked, and 0 total messages. Below the table, there is a form to "Add a new queue". The form fields include: Virtual host: /, Type: Default for virtual host, Name: (empty), Durability: Durable, and Arguments: (empty). There are also buttons for Add, Auto expire, Message TTL, Overflow behaviour, Single active consumer, Dead letter exchange, Dead letter routing key, Max length, Max length bytes, and Leader locator. At the bottom of the page, there are links for HTTP API, Documentation, Tutorials, New releases, Commercial edition, Commercial support, Discussions, Discord, Plugins, and GitHub.

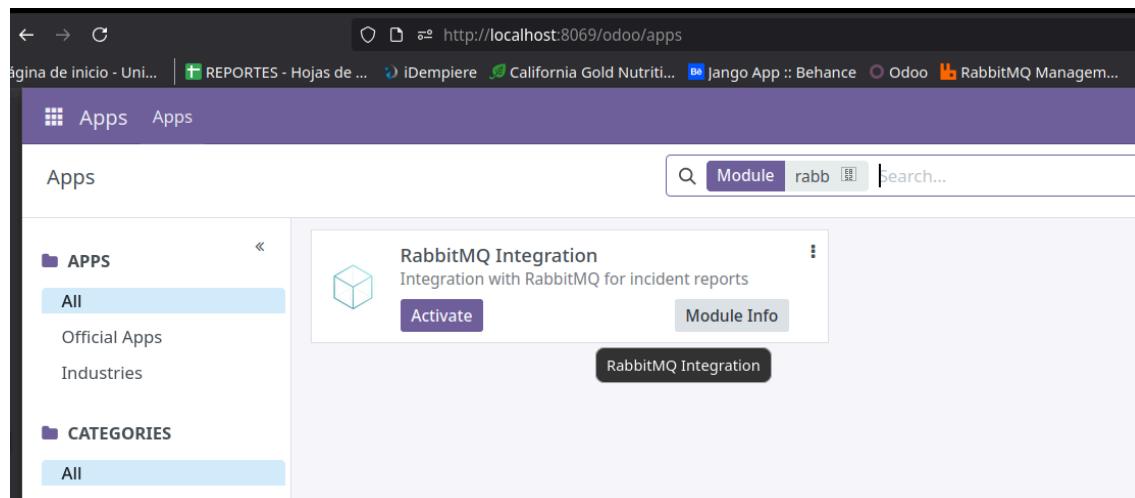
## Instalar Pika

```
[a]
Defaulting to user installation because normal site-packages is not writeable
WARNING: Skipping /usr/lib/python3.12/dist-packages/charset_normalizer-3.3.2.dist-info due to invalid metadata entry 'name'
collecting pika
  Downloading pika-1.3.2-py3-none-any.whl.metadata (13 kB)
  Downloading pika-1.3.2-py3-none-any.whl (155 kB)
    ━━━━━━━━━━━━━━━━━━━━ 155.4/155.4 kB 2.6 MB/s eta 0:00:00
WARNING: Skipping /usr/lib/python3.12/dist-packages/charset_normalizer-3.3.2.dist-info due to invalid metadata entry 'name'
```

## Módulo de Odoo para Pika

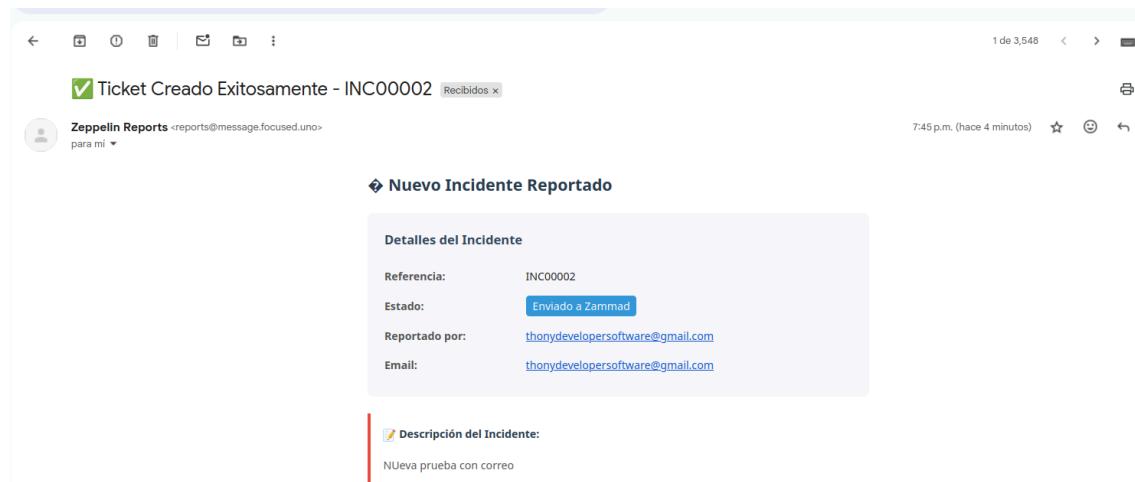


## Activar el módulo

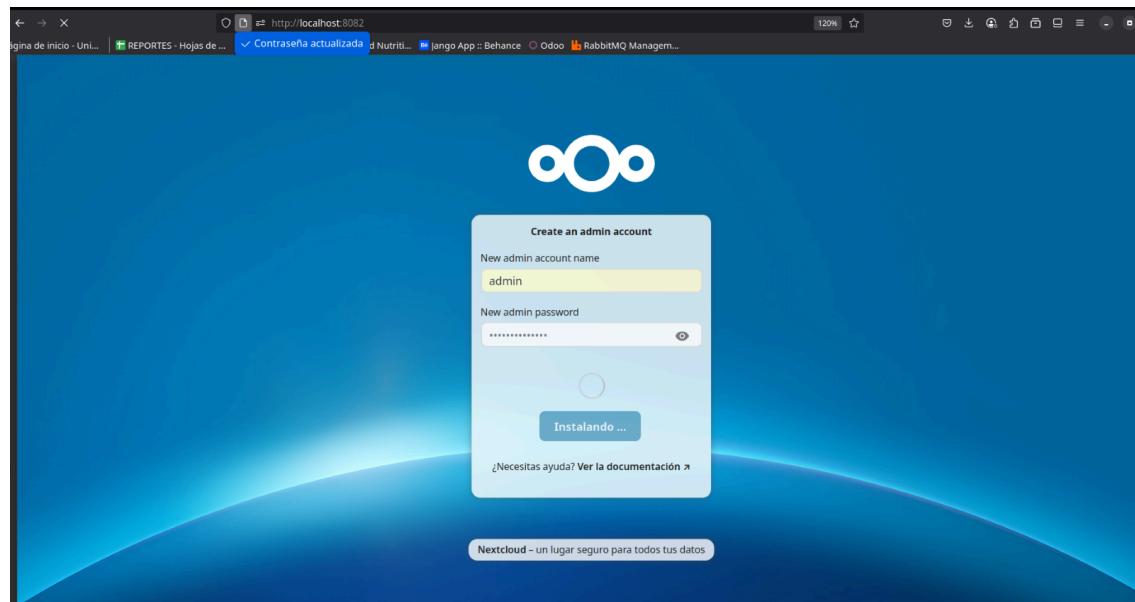


## Pruebas validando el correcto funcionamiento

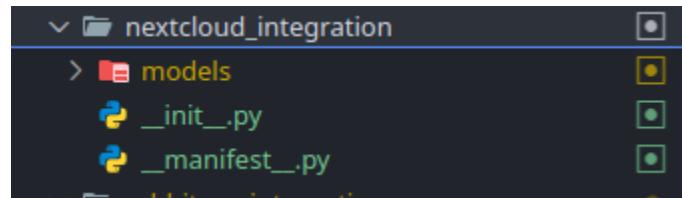
```
→ clinica-integracion git:(main) ✘ docker-compose logs mailer
mailer-1 | ● Escuchando mensajes en la cola "zammad_alerts"...
mailer-1 | ✅ Mensaje de incidente recibido: {
mailer-1 |   name: 'INC00002',
mailer-1 |   description: 'NUeva prueba con correo',
mailer-1 |   user_name: 'thonydevelopersoftware@gmail.com',
mailer-1 |   user_email: 'thonydevelopersoftware@gmail.com',
mailer-1 |   state: 'sent',
mailer-1 |   create_date: '2025-07-12T00:45:26.836846'
mailer-1 |
mailer-1 | } ✓ Correo de confirmación enviado correctamente.
```



## Configuración next cloud



## Nuevo módulo



## Añadir trust ip

```
{"error": "Trusted domain error.", "code": 15}  
↳ clinica-integracion git:(main) ✘ docker exec clinica-integracion-nextcloud-1 php /var/www/html/oc  
c config:system:set trusted_domains 0 --value=nextcloud  
System config value trusted_domains => 0 set to string nextcloud  
↳ clinica-integracion git:(main) ✘ docker exec clinica-integracion-nextcloud-1 php /var/www/html/oc  
c config:system:set trusted_domains 1 --value=localhost:8082  
System config value trusted_domains => 1 set to string localhost:8082  
↳ clinica-integracion git:(main) ✘ docker exec clinica-integracion-nextcloud-1 php /var/www/html/oc  
c config:system:set trusted_domains 2 --value=172.19.0.3  
System config value trusted_domains => 2 set to string 172.19.0.3  
↳ clinica-integracion git:(main) ✘ docker exec clinica-integracion-odoo-1 curl -v http://nextcloud/
```

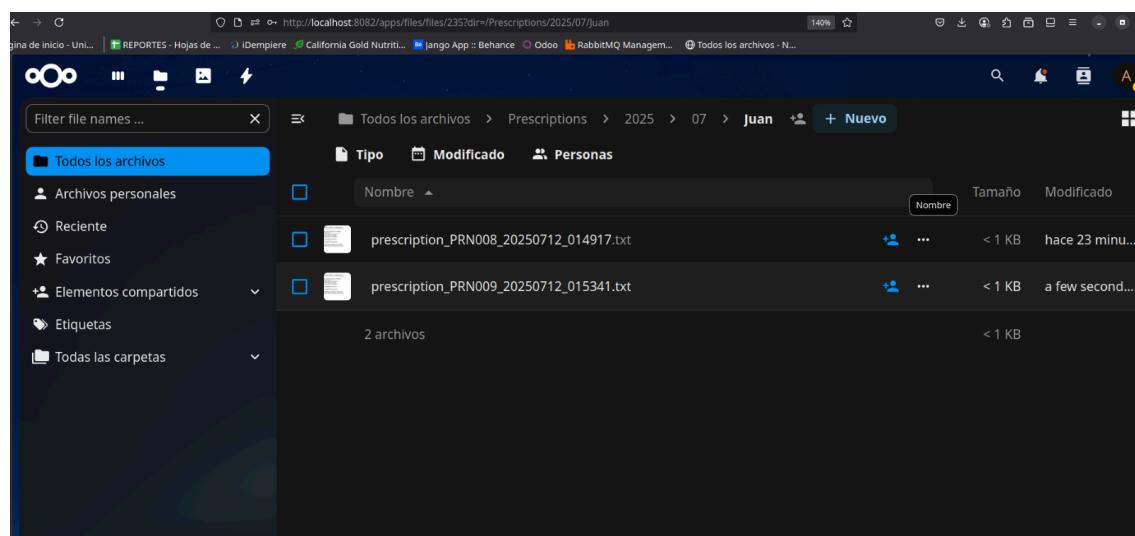
## Envío de prescripciones médicas por paciente

Screenshot of the Odoo Dental Prescription module showing a successful upload to NextCloud.

**Success**  
Prescription uploaded to NextCloud successfully

**PRN009**

Patient ?	PT00002 - Juan	Treatment ?	Teeth Whitening							
Appointment ?	APT/00001	Treatment Cost ?	\$ 50.00							
Selected Teeth ?	left	Prescribed Doctor ?	Dr. Sarah Johnson							
Referred Dentist ?	Doctor Penianieto	Prescription Date ?	07/12/2025							
Next Appointment Date ?										
Medicament	Generic Name	Dosage ...	Medicament Fo...	Quantity	Price	Medicine Take	Morning	After Noon	Night	Days
Add a line										



## Funcionamiento de Odoo

Gestión de pacientes, doctores

Screenshot of the Odoo application interface showing a patient record for "Anthony".

**Patient Details:**

- Contact:** Street..., Street 2..., City, State, ZIP.
- Personal Information:** Is Patient (checked), Date of Birth, Age (0), Patient No. (PT00001), gender (Male).
- Identifiers:** Tax ID (if not applicable).
- Communication:** Mobile, Email.

**Navigation:** Contacts & Addresses (selected), Sales & Purchase, Internal Notes.

Screenshot of the Odoo application interface showing a list of doctors.

All	Employee Name	Designation	Specialised In	Work Email	Work Mobile	Is a Dentist
<input type="checkbox"/>	Doctor Penianieto	asdad	Orthodontist	crre01.1.1@gmail.com		
<input type="checkbox"/>	Dr. Michael Brown			michael.brown@clinic.com		
<input type="checkbox"/>	Dr. Sarah Johnson			sarah.johnson@clinic.com		

## Gestión de citas

Screenshot of the Odoo Dental Clinic module showing the appointment creation interface.

**INFORMACIÓN DEL PACIENTE**

- Patient No. (Placeholder: Escriba o seleccione No. de Paciente)
- Age: 0
- Patient
- Mobile
- Gender
- Responsible: Administrator

**APPOINTMENT DETAIL**

- Dentist
- Date: 07/10/2025
- Available Times?
- Treatments
- Booking Time?
- Treatment Type
- Patient Category: Old (radio button selected)

## Gestión de prescripciones

Screenshot of the Odoo Dental Clinic module showing the prescription creation interface.

**PRN002**

Patient?	PAT002 - Jane Doe	Treatment?	Gum Contouring							
Appointment?	APT004	Treatment Cost?	\$ 60.00							
Selected Teeth?		Prescribed Doctor?	Dr. Michael Brown							
Referred Dentist?	Dr. Michael Brown	Prescription Date?	07/11/2025							
Next Appointment Date?										
Medicament	Generic Name	Dosage ...	Medicament Form	Quantity	Price	Medicine Take	Morning	After Noon	Night	Days
Medicina		0	Tablets	2	1.00	After Food	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2.00 <input type="checkbox"/>
Medicina		0	Liquid	7	1.00	After Food	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	7.00 <input type="checkbox"/>

Add a line

## Inventario

### Gestión de productos

The screenshot shows the Odoo Inventory module. At the top, there are tabs for Overview, Operations, Products, Reporting, and Configuration. The Products tab is selected. Below the tabs, there are buttons for New, Products, New, On Hand (0.00), Forecasted (0.00), Documents (0), Purchased (0.00), and More. There are also buttons for Update Quantity, Replenish, and Print Labels.

The main area is titled "Product" and has a search bar with placeholder text "e.g. Cheese Burger". Below the search bar are checkboxes for Sales, Purchase, Expenses, Is Medicine, and a camera icon for image upload.

The "General Information" tab is selected. It contains the following fields:

- Generic Name:** Cheese Burger
- Dosage Strength:** 0
- Product Type:** Goods (radio button selected)
- Invoicing Policy:** Ordered quantities
- Track Inventory:** Checked
- Sales Price:** \$ 1.00
- Sales Taxes:** 15% (incl. \$ 0.15)
- Cost:** \$ 0.00
- Purchase Taxes:** 15%
- Category:** All
- Reference:** (empty)
- Barcode:** (empty)

At the bottom left, there is a section for "INTERNAL NOTES".

## Órdenes de compra de inventario

The screenshot shows the Odoo Purchase module. At the top, there are tabs for Purchase, Orders, Products, Reporting, and Configuration. The Purchase tab is selected. Below the tabs, there are buttons for New, Requests for Quotation, and a search bar.

The main area displays statistics for RFQs:

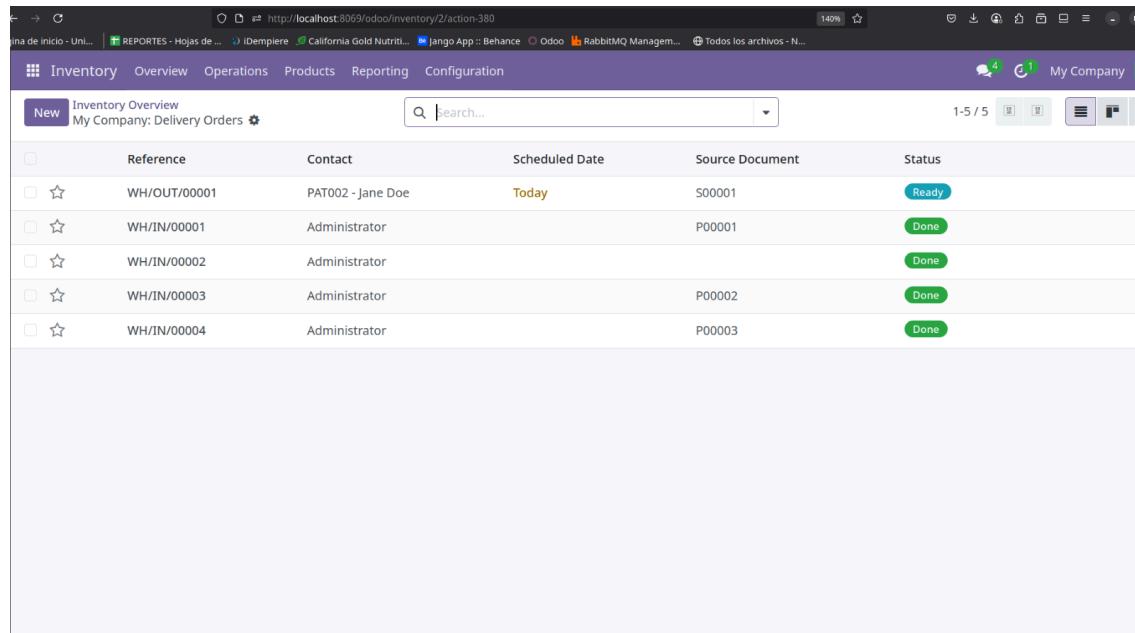
All RFQs	0 To Send	0 Waiting	0 Late	Avg Order Value	\$ 414.00	Purchased Last 7 Days	\$ 1,242.00
My RFQs	0	0	0	Lead Time to Purchase	0 Days	RFQs Sent Last 7 Days	0

Below the statistics, there is a table listing RFQ details:

	Reference	Vendor	Buyer	Order Deadline	Activities	Source Document	Total	Status
<input type="checkbox"/>	P00003	Administrator	Administrator	(empty)	(empty)	(empty)	\$ 977.50	<a href="#">Purchase Order</a>
<input type="checkbox"/>	P00002	Administrator	Administrator	(empty)	(empty)	(empty)	\$ 258.75	<a href="#">Purchase Order</a>
<input type="checkbox"/>	P00001	Administrator	Administrator	(empty)	(empty)	(empty)	\$ 5.75	<a href="#">Purchase Order</a>

The total value for all RFQs is \$ 1,242.00.

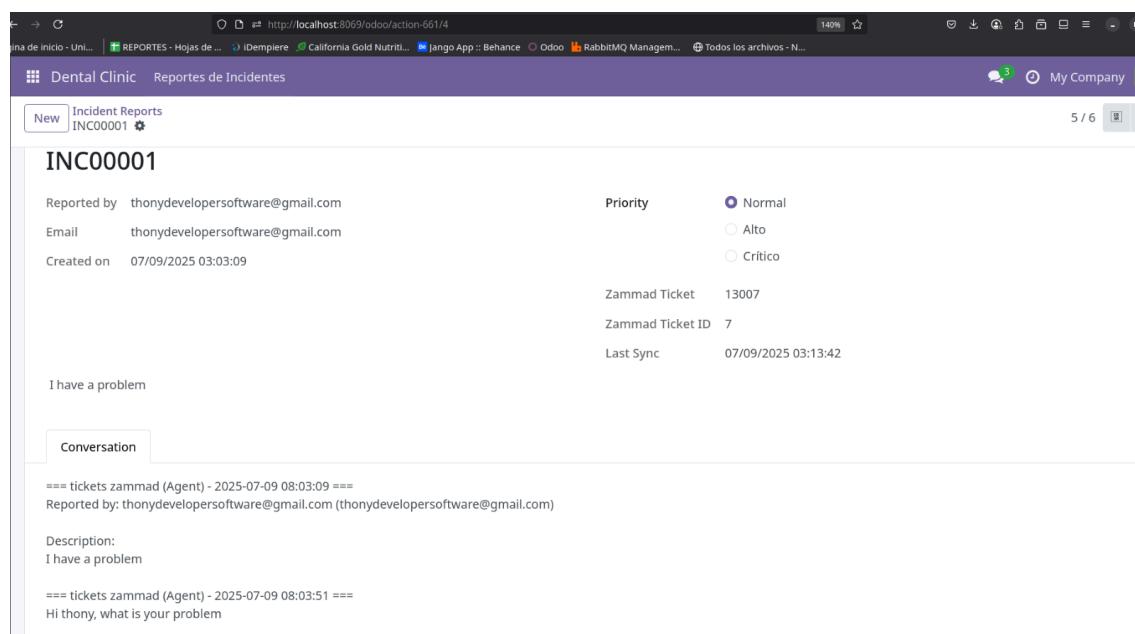
## Órdenes confirmadas de inventario



The screenshot shows the Odoo Inventory Overview page. At the top, there's a search bar with placeholder text 'Search...'. Below it is a table with columns: Reference, Contact, Scheduled Date, Source Document, and Status. The table contains five rows of data:

	Reference	Contact	Scheduled Date	Source Document	Status
<input type="checkbox"/>	WH/OUT/00001	PAT002 - Jane Doe	Today	S00001	Ready
<input type="checkbox"/>	WH/IN/00001	Administrator		P00001	Done
<input type="checkbox"/>	WH/IN/00002	Administrator			Done
<input type="checkbox"/>	WH/IN/00003	Administrator		P00002	Done
<input type="checkbox"/>	WH/IN/00004	Administrator		P00003	Done

## Funcionamiento de tickets



The screenshot shows the Odoo Incident Report details for ticket INC00001. The ticket information includes:

- Reported by: thonydevelopersoftware@gmail.com
- Email: thonydevelopersoftware@gmail.com
- Created on: 07/09/2025 03:03:09
- Priority: Normal (radio button selected)
- Zammad Ticket: 13007
- Zammad Ticket ID: 7
- Last Sync: 07/09/2025 03:13:42

The ticket description is: "I have a problem".

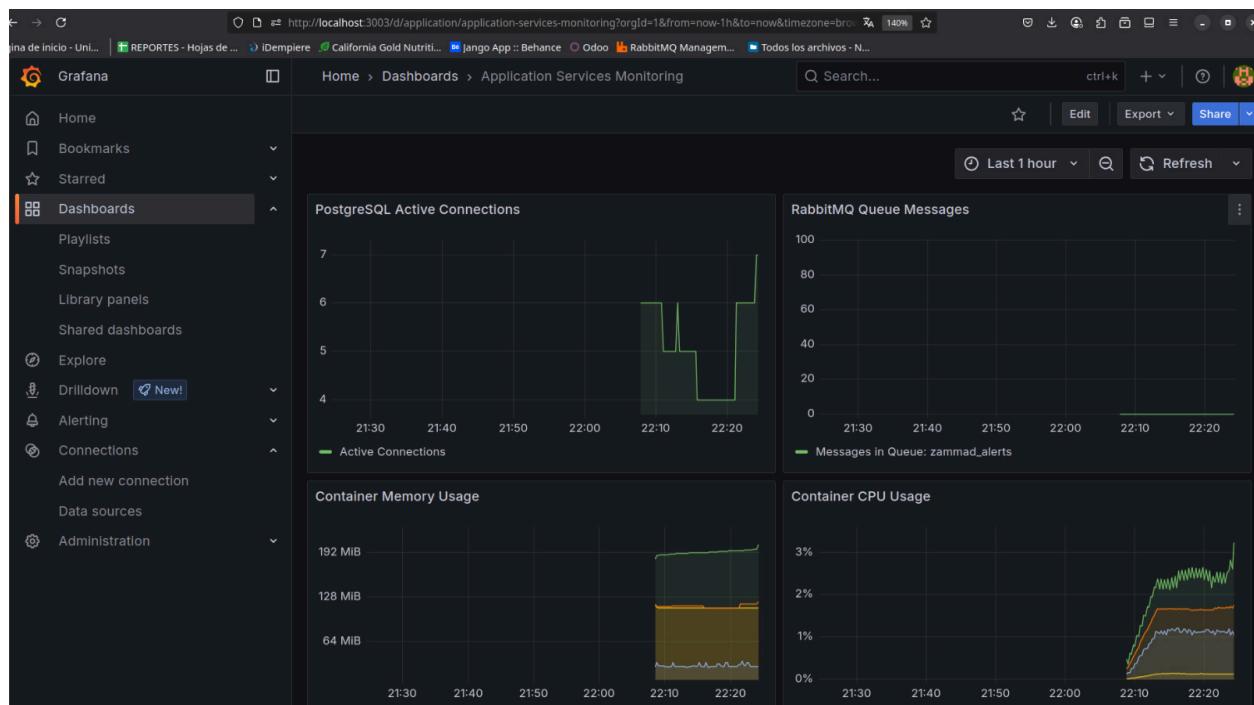
The conversation history is as follows:

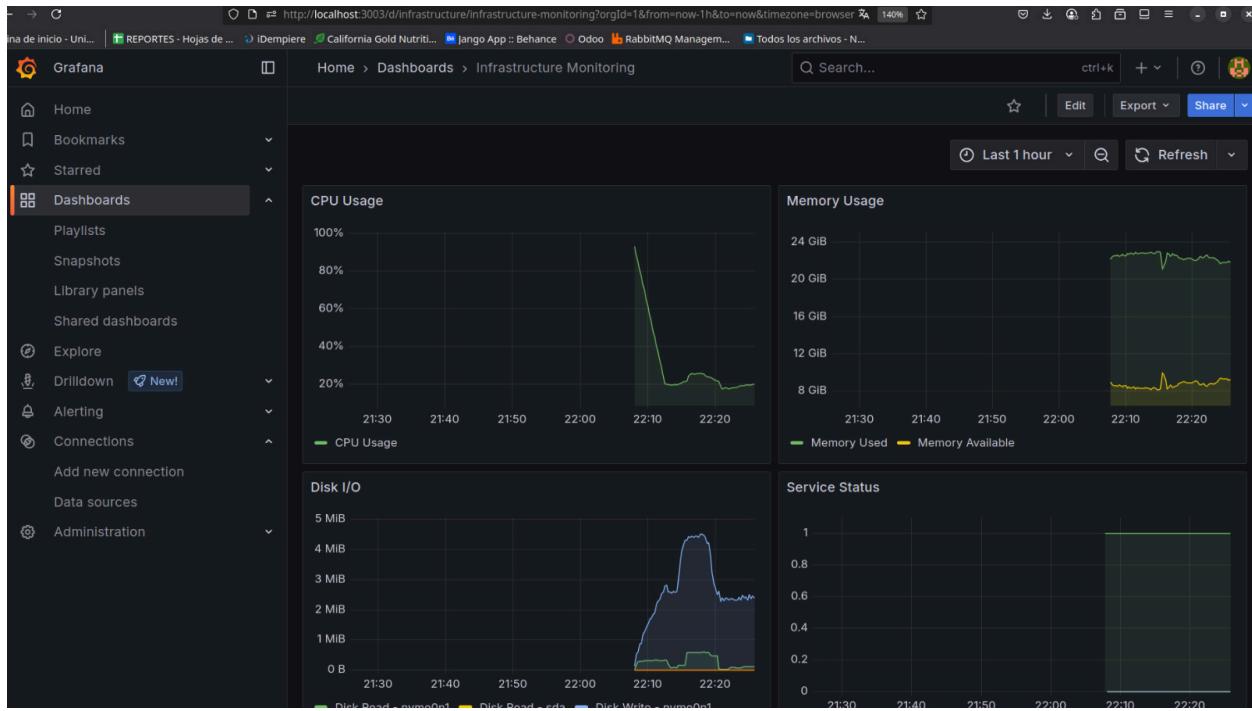
- Agent (Agent) - 2025-07-09 08:03:09: === tickets zammad (Agent) - 2025-07-09 08:03:09 ===  
Reported by: thonydevelopersoftware@gmail.com (thonydevelopersoftware@gmail.com)
- Description:  
I have a problem
- Agent (Agent) - 2025-07-09 08:03:51: === tickets zammad (Agent) - 2025-07-09 08:03:51 ===  
Hi thony, what is your problem

The screenshot shows a ticket management interface. On the left, there's a sidebar with options like 'Tablero', 'Vista general', 'Inbound Call', and 'Test Ticket' (which is selected). The main area displays a ticket titled 'Test Ticket' (Ticket#13004) created 2 days 19 hours ago. It contains two messages: 'Test body' and 'gracias por responder'. A note at the bottom says 'Ingrese nota o seleccionar archivo adjunto...'. On the right, there's a sidebar for ticket details: Propietario (empty), Estado (nuevo), Prioridad (2 normal), Etiquetas (Aregar Tag), Enlaces (Añadir Enlace), and Notificaciones (Suscribir). At the bottom, there are buttons for 'Permanecer en la pestaña' and 'Actualizar'.

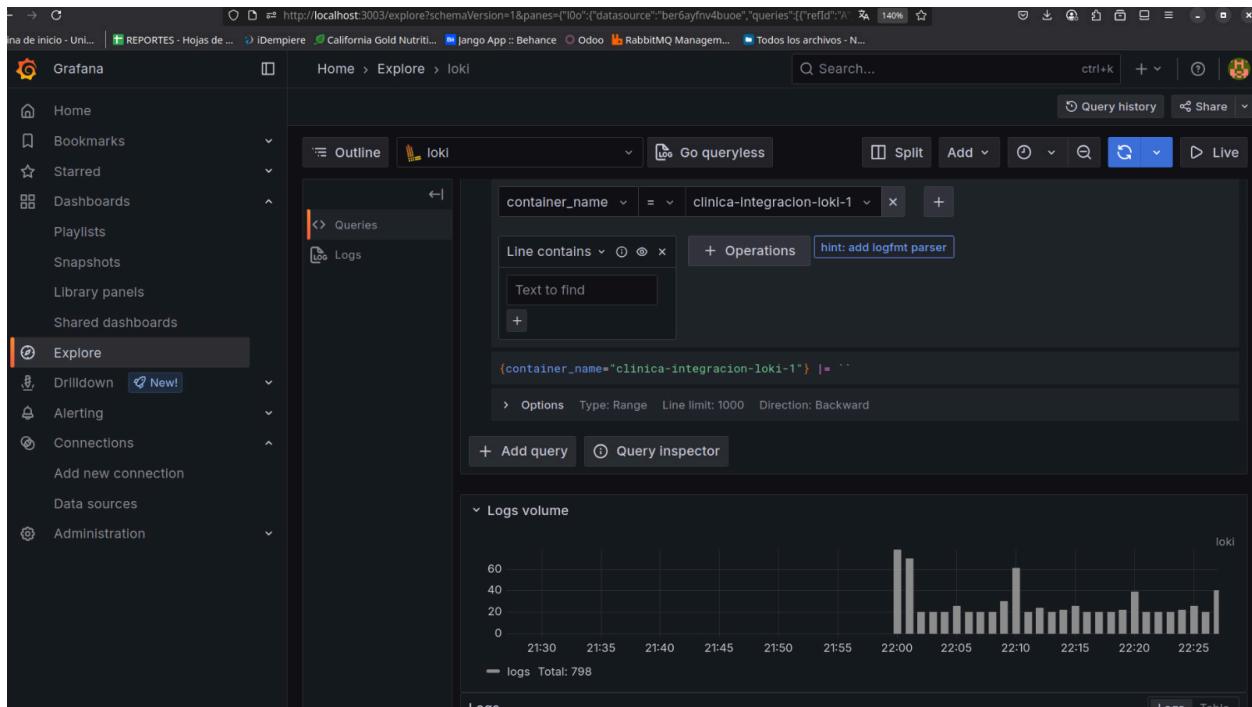
Para monitorear las aplicaciones usamos Grafana con Prometheus y con Loki

## Grafana con Prometheus





## Grafana con Loki



## **6. Conclusiones y Reflexión Crítica**

El desarrollo de esta solución de integración para la Clínica Universitaria permitió evidenciar cómo una arquitectura basada en patrones de integración modernos y tecnologías open source puede solventar de forma efectiva los problemas derivados de sistemas fragmentados. A través de la implementación de componentes como Odoo ERP, Zammad, Nextcloud, RabbitMQ, Clerk y un servicio de mensajería en Node.js, se logró construir un ecosistema digital cohesionado, seguro y escalable.

Los cuatro patrones de integración aplicados, API RESTful, transferencia de archivos, mensajería asíncrona y autenticación centralizada, permitieron abordar con éxito las principales deficiencias identificadas: duplicidad de datos, demoras operativas, falta de trazabilidad y experiencia de usuario deficiente. Además, el uso de herramientas de monitoreo como Loki, Grafana y Prometheus fortaleció la observabilidad del sistema, facilitando la detección de fallos y la mejora continua.

La solución implementada no solo responde a requerimientos funcionales, sino que también considera factores de salud pública, seguridad, bienestar, escalabilidad y sostenibilidad económica. En definitiva, se estableció una base tecnológica robusta que puede escalarse y adaptarse a otras áreas clínicas o institucionales con requerimientos similares.

### **Conclusión y Recomendación – Anthony Cochea**

#### **Conclusión:**

Participar en este proyecto me permitió comprender en profundidad los desafíos técnicos que conlleva integrar sistemas complejos y heterogéneos. A lo largo del desarrollo, enfrenté múltiples obstáculos relacionados con autenticación, flujos de datos y documentación deficiente en algunas plataformas, especialmente al implementar Clerk como proveedor OAuth en un entorno ERP. Esta experiencia reforzó la importancia de la adaptabilidad, el diseño modular y la observabilidad como pilares clave en soluciones escalables.

#### **Recomendación:**

Mi principal recomendación es realizar una investigación exhaustiva previa sobre las tecnologías a utilizar, ya que, en mi caso, Clerk funcionaba correctamente con Zammad pero no

existía un módulo funcional para Odoo. Esto obligó a replantear parte de la arquitectura en etapas avanzadas. Es esencial identificar estas limitaciones técnicas desde el inicio, validar compatibilidades y prever planes de contingencia para evitar reprocesos.

---

## **Conclusión y Recomendación – Gabriel Erazo**

### **Conclusión:**

Durante el proyecto, logré consolidar mis conocimientos en integración de sistemas distribuidos, especialmente en el uso de herramientas como Grafana, Loki y Prometheus para monitoreo y trazabilidad. El despliegue en contenedores Docker facilitó la estandarización del entorno y redujo fricciones durante las pruebas, confirmando el valor de la infraestructura como código y la automatización para proyectos complejos.

### **Recomendación:**

Recomiendo priorizar desde etapas tempranas la implementación de herramientas de observabilidad. Tener visibilidad sobre el estado de cada servicio permite una respuesta más rápida ante errores, facilita el debugging, y mejora significativamente la experiencia de mantenimiento del sistema. Además, fomenta una cultura de responsabilidad operativa y mejora continua.

---

## **Conclusión y Recomendación – Matías Cedeño**

### **Conclusión:**

Mi participación me permitió fortalecer competencias relacionadas con la gestión de usuarios y roles, así como en la creación de flujos funcionales seguros y coherentes para los distintos actores del sistema. Comprendí la importancia de la experiencia del usuario como eje transversal, incluso en soluciones técnicas, y cómo una autenticación centralizada puede simplificar la interacción y aumentar la seguridad.

### **Recomendación:**

Sugiero que en futuros proyectos se priorice el diseño centrado en el usuario desde el inicio. No

basta con que el sistema funcione técnicamente: debe ser intuitivo, accesible y coherente para todos los perfiles involucrados. Invertir tiempo en mapear adecuadamente los casos de uso reales facilita una integración más efectiva y evita fricciones innecesarias en la adopción.

---

## **Conclusión y Recomendación – Matheo Chávez**

### **Conclusión:**

Este proyecto fue una oportunidad para explorar en profundidad las capacidades de Odoo como ERP modular, así como los retos al desarrollar un módulo específico para el área odontológica. A pesar de su flexibilidad, la curva de aprendizaje y la complejidad del framework fueron notables. El trabajo en equipo y la documentación colaborativa fueron clave para superar estos desafíos.

### **Recomendación:**

Recomiendo mantener una documentación técnica clara y colaborativa a lo largo del desarrollo. En proyectos con múltiples servicios y dependencias, una buena documentación permite no solo mantener la trazabilidad, sino también facilitar la escalabilidad del equipo y el traspaso de conocimiento. Esta práctica reduce tiempos de incorporación y mejora la calidad técnica del proyecto.

## **Reflexión Crítica**

El proceso de diseño e implementación de esta solución evidenció tanto logros importantes como desafíos relevantes. Uno de los principales aciertos fue la correcta selección de tecnologías open source, que permitieron construir una solución funcional sin incurrir en altos costos de licenciamiento. Asimismo, la integración modular de servicios mediante contenedores Docker facilitó el despliegue y la escalabilidad del sistema, fortaleciendo su adaptabilidad a distintos entornos.

No obstante, también se identificaron retos que deben considerarse en futuras iteraciones. En particular, Odoo presentó una curva de aprendizaje considerable, debido a la complejidad de su

estructura modular y a las limitaciones de su documentación para integraciones personalizadas, especialmente al implementar el módulo de odontología y gestionar flujos API con otros servicios.

Otro aspecto crítico es la sostenibilidad operativa a largo plazo: si bien la solución funciona en un entorno de prueba controlado, su implementación en un entorno real requeriría políticas claras de respaldo de datos, gestión de usuarios, auditorías de acceso y soporte técnico especializado.

Finalmente, este proyecto permitió comprender el verdadero valor de la integración de sistemas no solo como una solución técnica, sino como una estrategia para mejorar procesos, reducir errores clínicos y colocar al usuario, paciente o profesional, en el centro del sistema de salud. El aprendizaje obtenido trasciende lo técnico y aporta una visión más integral y estratégica sobre el rol de la ingeniería de software en contextos reales y de alto impacto social.