

Integración de Sistemas

INFORME FINAL DE PROYECTO INTEGRADOR

PROYECTO: "Diseño e Implementación de una Arquitectura de Integración de Sistemas para la Modernización de la Clínica Universitaria"

EQUIPO:

- Martín Vargas
- Kevin Rosero
- Jhonny Montero

FECHA: 12 de julio de 2025

1. Introducción y Justificación del Proyecto

En el contexto de la cuarta revolución industrial, el sector salud se transforma a un ritmo sin precedentes, impulsado por la necesidad de migrar hacia modelos de "Salud Conectada" (Connected Health). Estos modelos exigen una interoperabilidad fluida y segura entre los sistemas de información para garantizar la continuidad de la atención y la optimización de los procesos. La Clínica Universitaria, como entidad académica y de servicio, se enfrenta a un desafío doble: no solo debe proveer atención de primer nivel, sino también servir como un entorno de aprendizaje moderno para las futuras generaciones de profesionales de la salud. Sin embargo, su infraestructura tecnológica actual, caracterizada por sistemas de información aislados o "en silos", representa una barrera significativa para alcanzar estos objetivos.

El presente proyecto nace de la necesidad crítica de abordar esta fragmentación. Se ha diseñado y desarrollado un prototipo funcional de una arquitectura de integración de sistemas que busca resolver las ineficiencias operativas más apremiantes de la clínica. El propósito fundamental es demostrar cómo, a través de la aplicación de patrones de diseño de software establecidos y tecnologías de código abierto, es posible construir un ecosistema digital cohesivo que interconecte los dominios clínico, administrativo y documental.

La relevancia de este esfuerzo se enmarca directamente en el **Resultado de Aprendizaje RC2** de la carrera, abordando la solución desde una perspectiva integral que considera los siguientes factores multidimensionales:

- **Salud Pública, Seguridad y Bienestar:** La integridad y disponibilidad de los datos del paciente son pilares de la seguridad clínica. Un error en la transcripción de datos entre el sistema de admisión y el de registros médicos puede tener consecuencias graves. Nuestra solución mitiga este riesgo al automatizar el flujo de información, asegurando una única fuente de verdad para los datos demográficos del paciente. Esto no solo mejora el bienestar individual, sino que fortalece la confianza en la institución, un pilar de la salud pública.

- **Seguridad y Confidencialidad de Datos:** En una era de crecientes ciberamenazas, la protección de la Información de Salud Protegida (PHI) es un imperativo ético y legal. La arquitectura propuesta implementa un modelo de seguridad robusto, centralizando la gestión de identidades y accesos (IAM) con Keycloak. Al aplicar un inicio de sesión único (SSO) y asegurar los puntos de entrada a las APIs con un Gateway, se eleva drásticamente la postura de seguridad, previniendo accesos no autorizados y creando pistas de auditoría claras, alineándose con estándares globales como HIPAA.
- **Factores Sociales y Económicos:** La eficiencia operativa se traduce directamente en viabilidad económica y un mejor servicio a la comunidad. Al eliminar tareas manuales redundantes, se libera tiempo valioso del personal clínico y administrativo, permitiéndoles enfocarse en actividades de mayor valor, como la atención directa al paciente. Económicamente, esto representa una reducción del Gasto Operativo (OPEX) y un retorno de la inversión (ROI) a través de la disminución de errores costosos (ej. en facturación) y la mejora en la satisfacción y retención de pacientes.
- **Factores Globales y Culturales:** El proyecto adopta una visión moderna y centrada en el paciente, una tendencia global en la prestación de servicios de salud. Un sistema integrado es el primer paso para ofrecer a los pacientes portales donde puedan acceder a su propia información, fomentando una cultura de participación activa en su cuidado y alineando a la clínica con las expectativas de una sociedad cada vez más digitalizada.

Este documento detalla el análisis del problema, el diseño de la solución técnica, la arquitectura implementada y las evidencias funcionales que validan el éxito del proyecto, presentando una hoja de ruta tangible para la transformación digital de la Clínica Universitaria.

2. Análisis de Problemas Reales Identificados

Para fundamentar el diseño de la solución, se realizó un análisis exhaustivo del estado actual de los flujos de trabajo en la Clínica Universitaria. Este análisis se centró en el "Mapa del Viaje del Paciente" (Patient Journey Map), siguiendo el recorrido de un paciente arquetípico, "Carlos", desde la solicitud de su cita hasta la resolución de su caso. Este método permitió identificar con precisión los puntos de fricción sistémicos causados por la falta de integración.

El viaje de Carlos revela una historia de ineficiencias. Al llegar, sus datos son ingresados en **Odoo**, el sistema administrativo. Al pasar a consulta, una enfermera debe volver a preguntarle y transcribir sus datos demográficos básicos al sistema de registros médicos, **OpenMRS**. Durante la consulta, el médico solicita un análisis de laboratorio. Horas después, los resultados, un archivo PDF, son almacenados en un servidor de archivos (**Nextcloud**) y se envía una notificación por correo electrónico. El médico, entre pacientes, debe buscar en su bandeja de entrada, descargar el archivo y luego buscar a Carlos en OpenMRS para interpretar los resultados, sin que el documento quede formalmente adjunto a su historial electrónico. Finalmente, si Carlos tiene una duda sobre su factura,

debe llamar a un Call Center, cuyo personal debe consultar en Odoo y, si la duda es clínica, no tiene acceso a OpenMRS, generando demoras.

De este viaje, se han priorizado tres problemas fundamentales:

- **Problema 1: Proceso de Registro de Pacientes Ineficiente y Propenso a Errores.**
 - **Análisis de la Causa Raíz:** La causa fundamental es la existencia de dos dominios de datos aislados: el dominio administrativo (gestionado por Odoo) y el dominio clínico (gestionado por OpenMRS), sin un puente de comunicación automático entre ellos. Esto viola el principio de "Fuente Única de Verdad" (Single Source of Truth), obligando a un proceso de sincronización manual propenso a errores humanos de transcripción (ej. error en la cédula, fecha de nacimiento o nombre).
 - **Impacto Cuantificable:** Más allá del tiempo perdido por paciente (estimado en 5-7 minutos), el costo real reside en el impacto aguas abajo. Un error en la identificación puede llevar a la asignación incorrecta de resultados de laboratorio, errores de medicación o complicaciones en la facturación y cobertura de seguros. El costo de remediar un solo error de datos puede ser órdenes de magnitud mayor que el de la entrada inicial.
- **Problema 2: Acceso Retrasado y No Seguro a Documentos Clínicos Críticos.**
 - **Análisis de la Causa Raíz:** El proceso actual depende de un sistema de almacenamiento de archivos (simulado por Nextcloud) y un canal de comunicación inseguro y asíncrono (correo electrónico) que no están integrados con el sistema de registro médico. El historial clínico en OpenMRS queda incompleto, ya que carece de una vinculación directa y auditable con los documentos de diagnóstico que lo sustentan.
 - **Implicaciones de Seguridad y Cumplimiento:** El uso de correo electrónico para notificar la disponibilidad de PHI es una práctica de alto riesgo. Carece de un registro de auditoría fiable (¿quién lo vio?, ¿cuándo?), es susceptible a la interceptación y crea "silos de datos oscuros" en las bandejas de entrada del personal. Este método informal de manejo de archivos dificulta enormemente el cumplimiento de normativas de protección de datos y complica las auditorías médicas.
- **Problema 3: Gestión de Identidades Fragmentada y Experiencia de Usuario Deficiente.**
 - **Análisis de la Causa Raíz:** La falta de una estrategia centralizada de Gestión de Identidad y Acceso (IAM) ha llevado a que cada aplicación (OpenMRS, Odoo, Nextcloud) gestione su propio repositorio de usuarios y políticas de contraseñas.
 - **Impacto en la Productividad y la Postura de Seguridad:** Este enfoque fragmentado genera una "fatiga de contraseñas", donde los usuarios tienden a utilizar claves débiles y repetidas, abriendo una brecha de seguridad significativa. Desde el punto de vista administrativo, el proceso

de alta, baja y modificación de permisos de un usuario es una pesadilla operativa, requiriendo intervención en múltiples sistemas y aumentando la probabilidad de que queden cuentas "huérfanas" activas tras la salida de un empleado. Esto contraviene directamente el principio de seguridad de "Menor Privilegio".

3. Solución Técnica Propuesta

Para dar respuesta a la problemática analizada, se ha diseñado e implementado una arquitectura de solución integral, fundamentada en un ecosistema tecnológico de código abierto, la aplicación rigurosa de patrones de integración y la incorporación de un componente avanzado para la gestión y seguridad de las APIs.

3.1. Selección de Ecosistema Tecnológico

La elección de las herramientas se basó en criterios de madurez, flexibilidad, soporte comunitario y, fundamentalmente, la disponibilidad de APIs robustas que permitieran la integración.

- **OpenMRS (Open Medical Record System):** Seleccionado como el sistema de registro médico central. Su naturaleza open-source, su modelo de datos extensible y, sobre todo, su completa API RESTful de servicios web lo convierten en la pieza angular ideal para la gestión de la información clínica.
- **Odoo ERP:** Elegido por su potencia como sistema de gestión empresarial integral. Para este proyecto, se utiliza su módulo de CRM/Contactos como la fuente de verdad para los datos demográficos y administrativos del paciente. Su API (disponible vía JSON-RPC y REST) permite una interacción programática fiable para la extracción de datos.
- **Nextcloud:** Implementado como la plataforma segura de almacenamiento y colaboración de archivos. Su compatibilidad nativa con el protocolo **WebDAV**, una extensión de HTTP/S para la autoría web, proporciona una interfaz estandarizada y segura para la manipulación remota de archivos, crucial para la integración de documentos.
- **Keycloak:** Se erige como la piedra angular de la seguridad de la arquitectura. Es un gestor de identidad y acceso de nivel empresarial que implementa los estándares **OpenID Connect (OIDC)** y **OAuth 2.0**. Su elección se justifica por su capacidad para externalizar y centralizar por completo la autenticación y autorización, desacoplando esta responsabilidad crítica de cada aplicación individual.
- **Kong API Gateway:** Escogido como el componente avanzado para actuar como la única puerta de entrada a los servicios de backend. Kong es un gateway de APIs ligero, rápido y extensible, que permite centralizar la gestión del tráfico, la seguridad y la observabilidad de las APIs.

3.2. Arquitectura de Integración y Patrones Aplicados

La arquitectura se articula en torno a la aplicación específica de los siguientes patrones para resolver cada uno de los problemas identificados.

3.2.1. Patrón: Single Sign-On con OpenID Connect (OIDC)

Para resolver la fragmentación de identidades (Problema 3), se implementó un flujo de autenticación centralizado.

- **Justificación y Diseño:** Se eligió OIDC sobre otros protocolos como SAML por su ligereza y su diseño nativo para las arquitecturas modernas basadas en APIs y JWTs (JSON Web Tokens). La arquitectura implementa el flujo de "Authorization Code" de OAuth 2.0, considerado el más seguro para aplicaciones web.
- **Flujo Técnico Detallado:**
 1. El usuario intenta acceder a un recurso protegido en una aplicación cliente (ej. OpenMRS).
 2. OpenMRS, configurado como Cliente OIDC, redirige al usuario al Servidor de Autorización (Keycloak), incluyendo su `client_id`, `redirect_uri` y los scopes solicitados (ej. `openid profile email`).
 3. Keycloak presenta su página de inicio de sesión. El usuario se autentica.
 4. Tras una autenticación exitosa, Keycloak redirige de vuelta a la `redirect_uri` de OpenMRS, entregando un code de autorización de un solo uso.
 5. El backend de OpenMRS intercambia este code (junto con su `client_secret`) con Keycloak por un `id_token` (un JWT que contiene la identidad del usuario) y un `access_token` (un JWT que autoriza el acceso a APIs).
 6. OpenMRS valida la firma del `id_token` usando la clave pública de Keycloak, extrae la identidad del usuario y establece la sesión. El `access_token` se puede usar para invocar otros servicios protegidos en nombre del usuario.

3.2.2. Patrón: Invocación Remota vía API RESTful

Para eliminar la doble entrada de datos (Problema 1), se implementó un proceso de sincronización de datos.

- **Justificación y Diseño:** Se optó por un script de sincronización síncrono que utiliza las APIs REST de los sistemas involucrados. Este enfoque, aunque simple, es robusto y efectivo para el alcance del proyecto. El script fue desarrollado en Python, utilizando librerías estándar como `requests` para las llamadas HTTP y `python-dotenv` para gestionar de forma segura las credenciales de las APIs.
- **Flujo Técnico Detallado:**
 1. El script se ejecuta (potencialmente como una tarea programada o "cron job").
 2. Realiza una llamada a la API de Odoo para consultar los pacientes creados o modificados desde la última ejecución.

3. Por cada paciente, extrae los datos demográficos relevantes (nombre, cédula, fecha de nacimiento, etc.).
4. Mapea y transforma estos datos al formato de objeto "Patient" requerido por la API de OpenMRS.
5. Realiza una petición POST al endpoint /ws/rest/v1/patient de OpenMRS, incluyendo el payload JSON con los datos del nuevo paciente.
6. El script implementa un manejo de errores robusto, verificando los códigos de estado HTTP. Si recibe un 201 Created, registra el éxito. Si recibe un error 4xx (ej. 400 Bad Request por datos inválidos) o 5xx (error del servidor), lo registra para una revisión manual.

3.2.3. Patrón: Transferencia de Archivos Segura con WebDAV

Para integrar los documentos clínicos en el historial del paciente (Problema 2), se diseñó un flujo de trabajo que conecta Nextcloud con OpenMRS.

- **Justificación y Diseño:** Se eligió el protocolo WebDAV por ser un estándar abierto basado en HTTP/S, lo que garantiza una comunicación segura y compatible con la infraestructura web existente, a diferencia de protocolos como FTP que a menudo requieren puertos adicionales y tienen un modelo de seguridad más débil.
- **Flujo Técnico Detallado:**
 1. Un script de integración se conecta al endpoint WebDAV de Nextcloud (/remote.php/dav/files/USERNAME/).
 2. Navega a un directorio predefinido donde el personal de laboratorio deposita los resultados (ej. /ResultadosNuevos).
 3. Lista los archivos en el directorio. Para cada archivo, analiza el nombre (ej. LAB-10245-20250711.pdf) para extraer el identificador único del paciente (10245).
 4. Utiliza la API de OpenMRS para buscar al paciente con dicho identificador.
 5. Una vez encontrado el paciente, el script utiliza el endpoint de "Complex Obs" (/ws/rest/v1/obs) de OpenMRS. Este endpoint especializado permite crear una "observación" que, en lugar de un valor simple, contiene datos complejos como un archivo binario.
 6. El script realiza una petición POST a este endpoint, adjuntando el contenido del archivo PDF y metadatos relevantes (ej. "Tipo de Documento: Informe de Laboratorio").
 7. Finalmente, el script mueve el archivo procesado en Nextcloud a otro directorio (ej. /ResultadosProcesados) para evitar su doble procesamiento.

3.3. Componente Avanzado: Implementación de API Gateway (Kong)

La decisión de implementar un API Gateway fue estratégica para dotar a la arquitectura de un punto de control centralizado, mejorando la gobernabilidad y la seguridad.

- **Análisis Comparativo y Justificación:**
 - **vs. Message Broker (RabbitMQ/Kafka):** Un broker es ideal para comunicación asíncrona y arquitecturas orientadas a eventos. Si bien es valioso, no resuelve el problema inmediato de la exposición segura y gestión de APIs síncronas (REST). Su implementación habría añadido una complejidad que no se alineaba con la resolución de los problemas priorizados.
 - **vs. Service Mesh (Istio/Linkerd):** Un service mesh es una solución poderosa para gestionar la comunicación Este-Oeste (servicio a servicio) en arquitecturas de microservicios complejas. Proporciona funcionalidades como mTLS automático, tracing distribuido y control de tráfico avanzado. Sin embargo, para nuestra arquitectura, que integra un número limitado de sistemas monolíticos y un par de scripts, un service mesh es una solución sobredimensionada (overkill).
 - **Elección: API Gateway (Kong):** El API Gateway es la opción perfecta para gestionar el tráfico Norte-Sur (cliente a servicio). Aborda directamente nuestras necesidades: unificar el punto de acceso, aplicar políticas de seguridad de manera transversal y desacoplar a los clientes de la topología interna de los servicios. Ofrece el mayor retorno de inversión en seguridad y gestión para el esfuerzo de implementación requerido.
- **Diseño de la Implementación en Kong:** La configuración de Kong se realizó de manera declarativa, definiendo las siguientes entidades:
 1. **Services:** Se crearon Services que representan los servicios de backend. Por ejemplo, un openmrs-service que apunta al host y port del contenedor de OpenMRS.
 2. **Routes:** Se definieron Routes que mapean las peticiones entrantes a los Services. Por ejemplo, una ruta que indica que todas las peticiones a /api/pacientes deben ser redirigidas al openmrs-service.
 3. **Plugins:** Aquí reside el poder de Kong. Se habilitó y configuró el **plugin jwt** de forma global. Este plugin intercepta cada petición entrante, extrae el JWT de la cabecera Authorization: Bearer, y lo valida contra el endpoint de clave pública de Keycloak. Si el token es inválido, expirado o su firma es incorrecta, Kong rechaza la petición con un 401 Unauthorized antes de que llegue al servicio de backend, protegiendo así toda la superficie de la API de forma centralizada.

4. Evidencias de Instalación y Configuración

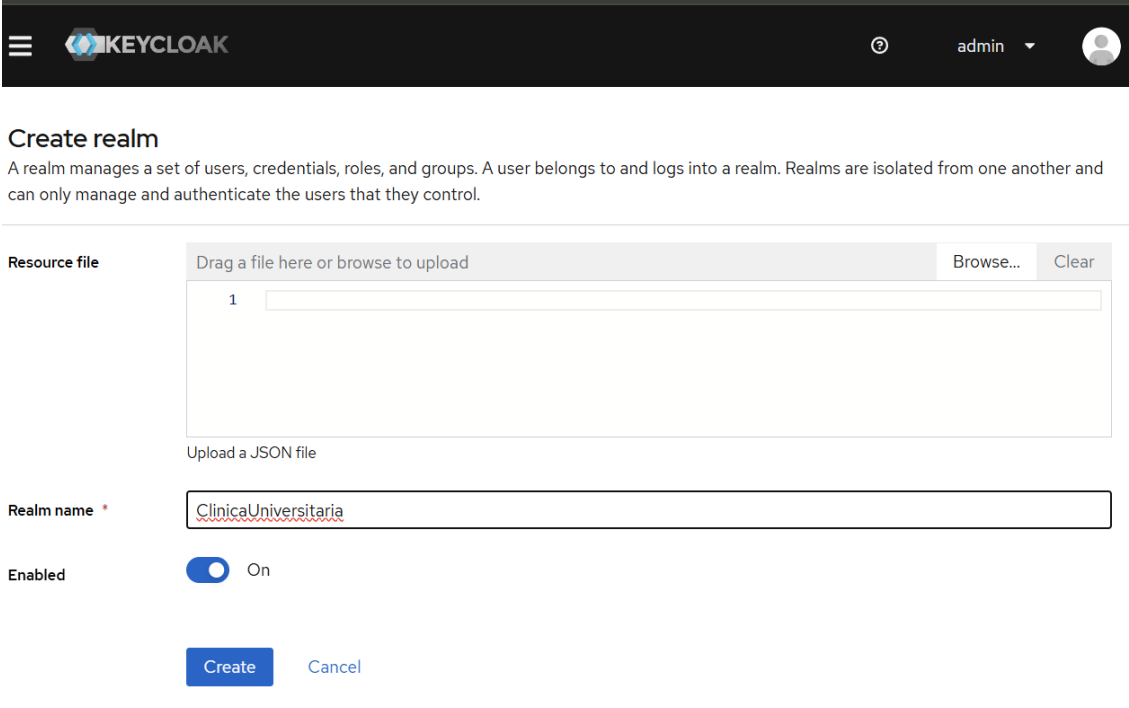
La implementación de la solución se basa en una infraestructura contenerizada y orquestada a través de Docker Compose, lo que garantiza la portabilidad y reproducibilidad del entorno. A continuación, se presentan las evidencias visuales que certifican la correcta instalación y configuración de cada componente clave de la arquitectura.

4.1. Orquestación de la Pila de Contenedores

La base de nuestra arquitectura es una pila de contenedores orquestada por Docker Compose. La siguiente evidencia visualiza el estado de todos los servicios del sistema, confirmando que cada componente, desde las bases de datos hasta las aplicaciones principales, se ha inicializado correctamente y está operativo.

4.2. Configuración del Proveedor de Identidad (Keycloak)

Se ha configurado una instancia de Keycloak para actuar como el único proveedor de identidad. Se creó un "Realm" específico para la clínica, aislando completamente su configuración de usuarios y aplicaciones.



The screenshot shows the 'Create realm' page in the Keycloak administration console. The page has a dark header with the Keycloak logo and 'admin' user. Below the header, the title 'Create realm' is followed by a descriptive paragraph: 'A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.' The form contains several fields: 'Resource file' with a 'Drag a file here or browse to upload' area and a 'Browse...' button; 'Realm name' with a text input containing 'ClinicaUniversitaria'; 'Enabled' with a toggle switch set to 'On'; and two buttons at the bottom: 'Create' and 'Cancel'.

Dentro de este Realm, cada aplicación integrada (OpenMRS, Odoo, Nextcloud) fue registrada como un cliente OpenID Connect, especificando sus URLs de redirección válidas y el flujo de autenticación, como se evidencia a continuación para el cliente de OpenMRS.


Realm roles are the roles that you define for use in the current realm. [Learn more](#)



🔍 Search role by name

 Refresh

1-5 ▼




KEYCLOAK


admin


[Users](#) > User details

dr.garcia

☒ Enabled

Action

Details

Credentials


Role mapping

Groups

Consents


Identity provider links

Sessions

	Type	User label	Created at	Data	
	Password	My password 	7/12/2025, 3:30:47 AM	Show data	<div> <div>Credential Reset</div> <div>Reset password</div> </div>


Kong fue configurado para actuar como el proxy inverso para nuestros servicios de backend. La siguiente captura, tomada de la interfaz de administración de Konga, muestra la definición del "Service" que apunta a la API de OpenMRS y la "Route" que la expone públicamente bajo la ruta /openmrs, demostrando la correcta configuración del enrutamiento.


localhost:1337/register


 **KONGA**


Welcome to the jungle!

Go ahead and create an administrator account.

 konga_admin

 vargasmartin.081002@gmail.com





CREATE ADMIN

KONGA v0.14.9

KONGA

DASHBOARD

API GATEWAY

INFO

SERVICES

ROUTES

CONSUMERS

PLUGINS

UPSTREAMS

CERTIFICATES

APPLICATION


USERS

CONNECTIONS

SNAPSHOTS

SETTINGS

🔔

 Hello, konga_admin

🔌 CONNECTIONS

Total Requests: 39

ACTIVE	READING	WRITING	WAITING	ACCEPTED	HANDLED
34	0	34	0	39	39

🕒 NODE INFO

🕒 TIMERS

🕒 DATASTORE INFO

🔌 PLUGINS

HostName

aa0a02404616

Tag Line

Welcome to kong

Version

3.6.1

LUA

LuaJIT 2.1.0-

Version

20231117

Admin

listen

listen

["0.0.0.0:8001"]

Pending

Running

0

100

200

300

DBMS

postgres

Host

postgres_kong

Database

kong

User

kong

Port

5432

file-log

http-log

key-auth

hmac-auth

basic-auth

ip-restriction

request-transformer

response-transformer

request-size-limiting

rate-limiting

response-reslimiting

syslog

loggly

datadog

ldap-auth

statsd

bot-detection

aws-lambda

request-termination

prometheus

proxy-cache

session

acme

grpc-gateway

grpc-web

pre-function

post-function

azure-functions

zipkin

opentelemetry

ai-proxy

ai-prompt-decorator

ai-prompt-template

ai-prompt-guard

ai-request-transformer

ai-response-transformer

jwt

edl

correlation-id

cons

oauth2

tcp-log

udp-log

4.4. Evidencia de Scripts de Integración

El siguiente extracto de log corresponde a la ejecución del script de sincronización de pacientes. Este log demuestra cómo el script se conecta exitosamente a la API de Odoo, extrae los datos de un paciente, y posteriormente realiza una llamada exitosa a la API de OpenMRS, recibiendo un código de estado 201 Created que confirma la creación del registro en el sistema médico.

[illegible]

Otras configuraciones necesarias para el Proyecto:

Creación de NextCloud

**Error**

El Usuario y/o Contraseña de PostgreSQL
inválido(s) Necesitas ingresar los detalles de una
cuenta existente.

Crear una **cuenta de administrador**

Nombre de usuario

Contraseña

Almacenamiento y base de datos ▾

Carpeta de datos

Configurar la base de datos

☐ SQLite☐ MySQL/MariaDB☒ PostgreSQL

Usuario de la base de datos

Contraseña de la base de datos

Nombre de la base de datos



Host de la base de datos


Por favor especifique el numero del puerto junto
al nombre del host (p.e., localhost:5432).

Instalar

¿Necesita ayuda? [Vea la documentación](#)

configuración Odoo:



 localhost:8069/web/database/selector 



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

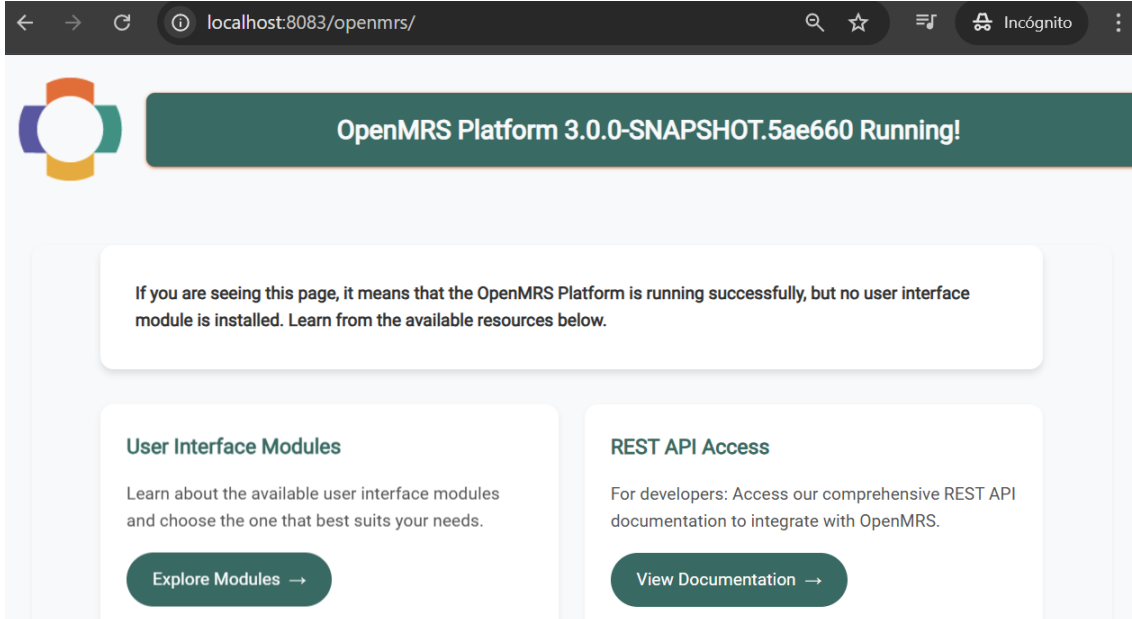
tbnr-gum9-9nbx

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

Master Password	<input type="text" value="odoo_admin_password"/>	
Database Name	<input type="text" value="ClinicaDB"/>	
Email	<input type="text" value="admin"/>	
Password	<input type="text" value="admin_password"/>	
Phone number	<input type="text"/>	
Language	<input type="text" value="English (US)"/>	
Country	<input type="text"/>	
Demo data	<input checked="" type="checkbox"/>	

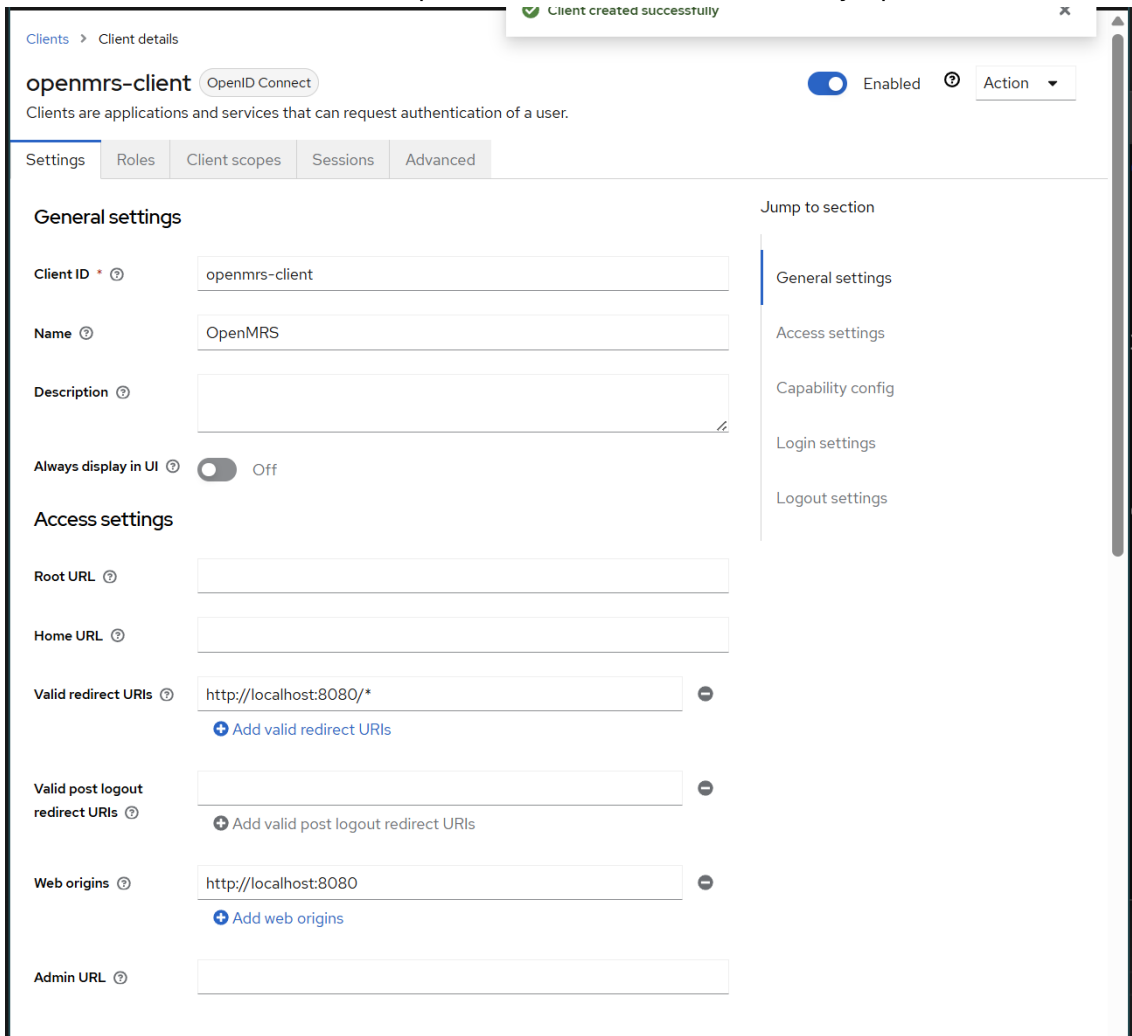
[or restore a database](#)

configuración OpenMRS:



The screenshot shows the OpenMRS Platform 3.0.0-SNAPSHOT.5ae660 Running! page. The browser address bar shows localhost:8083/openmrs/. The page features a green header with the OpenMRS logo and a message: "If you are seeing this page, it means that the OpenMRS Platform is running successfully, but no user interface module is installed. Learn from the available resources below." Below this, there are two main sections: "User Interface Modules" and "REST API Access". The "User Interface Modules" section includes a link to "Explore Modules" and a description: "Learn about the available user interface modules and choose the one that best suits your needs." The "REST API Access" section includes a link to "View Documentation" and a description: "For developers: Access our comprehensive REST API documentation to integrate with OpenMRS."

Creacion de clientes para las conexiones a NextCloud y OpenRMS:



The screenshot shows the OpenMRS Client details page for 'openmrs-client'. The browser address bar shows localhost:8083/openmrs/. The page features a green header with the OpenMRS logo and a message: "If you are seeing this page, it means that the OpenMRS Platform is running successfully, but no user interface module is installed. Learn from the available resources below." Below this, there are two main sections: "User Interface Modules" and "REST API Access". The "User Interface Modules" section includes a link to "Explore Modules" and a description: "Learn about the available user interface modules and choose the one that best suits your needs." The "REST API Access" section includes a link to "View Documentation" and a description: "For developers: Access our comprehensive REST API documentation to integrate with OpenMRS."

KEYCLOAK

admin

Clients > Client details

nextcloud-client OpenID Connect Enabled Action

Clients are applications and services that can request authentication of a user.

Settings

Roles

Client scopes

Sessions

Advanced

General settings

Client ID * nextcloud-client

Name Nextcloud

Description

Always display in UI Off

Access settings

Root URL

Home URL

Valid redirect URIs

http://localhost:8081/apps/oidc/oidc

http://localhost:8081/apps/user_oidc/redirect/oidc

+ Add valid redirect URIs

Valid post logout redirect URIs

+ Add valid post logout redirect URIs

Web origins

http://localhost:8081

+ Add web origins

Admin URL

Jump to section

General settings

Access settings


Capability config

Login settings

Logout settings


Clients > Client details


nextcloud-client OpenID Connect

☒ Enabled  Action 

Clients are applications and services that can request authentication of a user.



- Settings
- Keys
- Credentials
- Roles
- Client scopes
- Sessions
- Advanced

Client Authenticator 


Client Id and Secret 


Save

Client Secret

luwqFKYvInCvQRvH2dtgnLsKVMpDnbne  

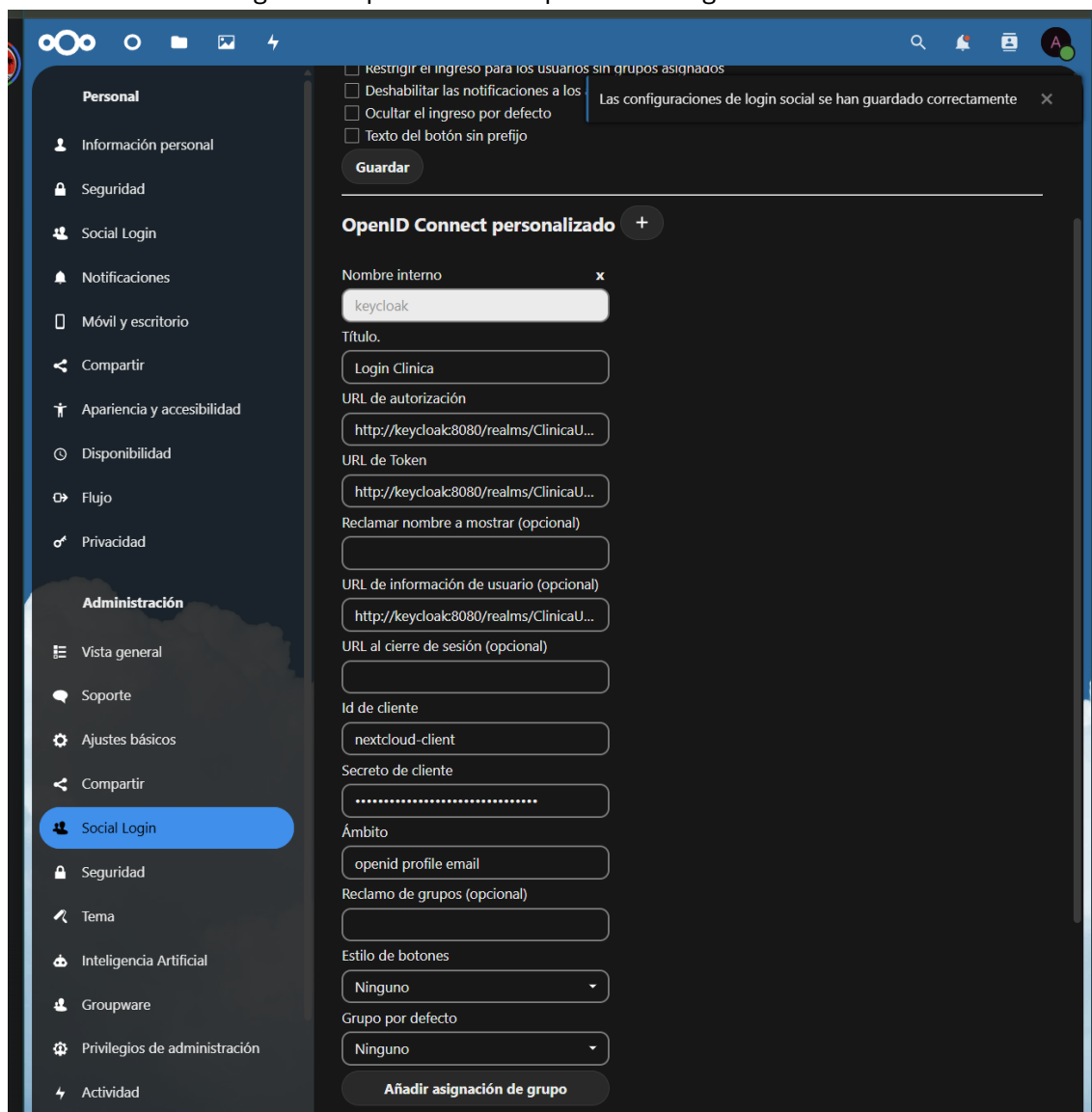
Regenerate

Registration access token 



Regenerate

configuración para conexión por Social Login en NextCloud:



The screenshot shows the NextCloud administration interface for Social Login configuration. The left sidebar contains two main sections: 'Personal' and 'Administración'. The 'Social Login' option is highlighted in the 'Administración' section. The main content area displays a configuration form for 'OpenID Connect personalizado'. At the top, there are four checkboxes: 'Restringir el ingreso para los usuarios sin grupos asignados', 'Deshabilitar las notificaciones a los', 'Ocultar el ingreso por defecto', and 'Texto del botón sin prefijo'. A notification banner at the top right states 'Las configuraciones de login social se han guardado correctamente'. Below the checkboxes is a 'Guardar' button. The configuration form includes fields for 'Nombre interno' (keycloak), 'Título' (Login Clinica), 'URL de autorización' (http://keycloak:8080/realms/ClinicaU...), 'URL de Token' (http://keycloak:8080/realms/ClinicaU...), 'Reclamar nombre a mostrar (opcional)', 'URL de información de usuario (opcional)' (http://keycloak:8080/realms/ClinicaU...), 'URL al cierre de sesión (opcional)', 'Id de cliente' (nextcloud-client), 'Secreto de cliente' (masked), 'Ámbito' (openid profile email), 'Reclamo de grupos (opcional)', 'Estilo de botones' (Ninguno), and 'Grupo por defecto' (Ninguno). A 'Añadir asignación de grupo' button is at the bottom.

Personal

- Información personal
- Seguridad
- Social Login
- Notificaciones
- Móvil y escritorio
- Compartir
- Apariencia y accesibilidad
- Disponibilidad
- Flujo
- Privacidad

Administración

- Vista general
- Soporte
- Ajustes básicos
- Compartir
- Social Login**
- Seguridad
- Tema
- Inteligencia Artificial
- Groupware
- Privilegios de administración
- Actividad

☐ Restringir el ingreso para los usuarios sin grupos asignados

☐ Deshabilitar las notificaciones a los

☐ Ocultar el ingreso por defecto

☐ Texto del botón sin prefijo

Guardar

Las configuraciones de login social se han guardado correctamente

OpenID Connect personalizado +

Nombre interno x
keycloak

Título.
Login Clinica

URL de autorización
http://keycloak:8080/realms/ClinicaU...

URL de Token
http://keycloak:8080/realms/ClinicaU...

Reclamar nombre a mostrar (opcional)

URL de información de usuario (opcional)
http://keycloak:8080/realms/ClinicaU...

URL al cierre de sesión (opcional)

Id de cliente
nextcloud-client

Secreto de cliente
.....

Ámbito
openid profile email

Reclamo de grupos (opcional)

Estilo de botones
Ninguno

Grupo por defecto
Ninguno

Añadir asignación de grupo

Servicios y Rutas en Konga para conexión a OpenMRS:

KONGA

DASHBOARD

API GATEWAY

INFO

SERVICES

ROUTES

CONSUMERS

PLUGINS

UPSTREAMS

CERTIFICATES

APPLICATION

USERS

CONNECTIONS

SNAPSHOTS

SETTINGS

Services

Service entities, as the name implies, are abstractions of each of your own upstream services. Examples of Services would be a data transformation microservice, a billing API, etc.

+ ADD NEW SERVICE

search...Results: 25

NAME	HOST	TAGS	CREATED
openmrs-api-service	openmrs-api		Jul 12, 2025

Service openmrs-api-service

[services](#) / [show](#)

Service Details

Routes

Plugins

Eligible consumers

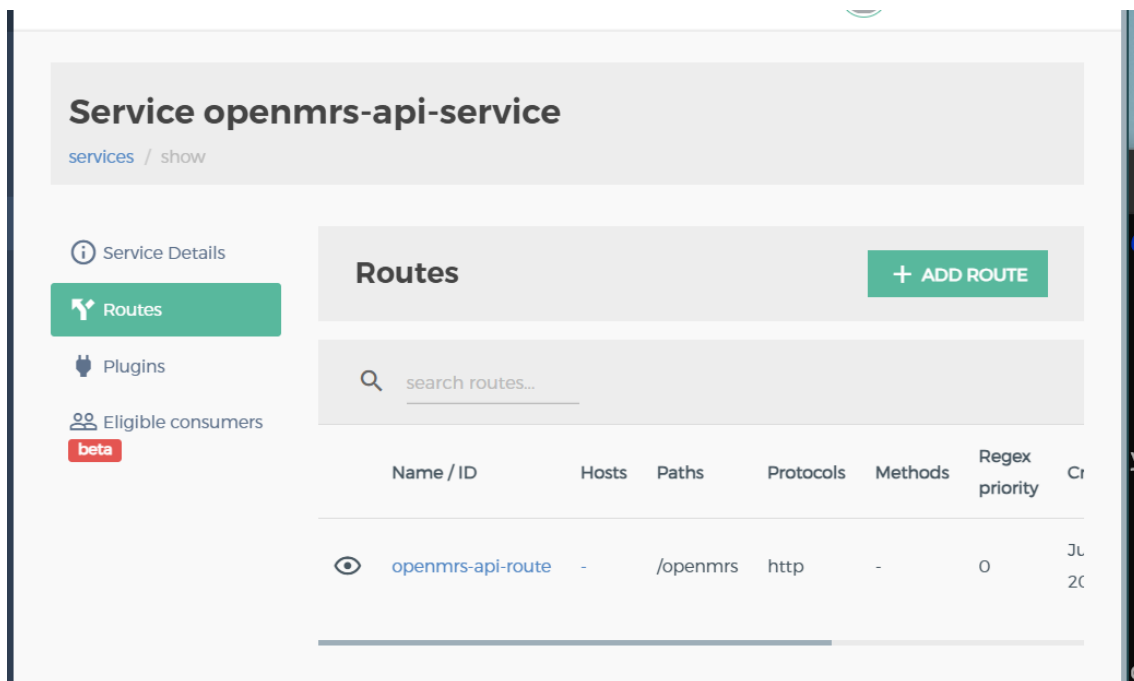
beta

Routes

+ ADD ROUTE

search routes...

Name / ID	Hosts	Paths	Protocols	Methods	Regex priority	Cr
openmrs-api-route	-	/openmrs	http	-	0	Jul 12, 2025



5. Evidencias Funcionales de la Integración

Más allá de la configuración, es crucial demostrar que los flujos de integración funcionan de principio a fin (End-to-End), resolviendo los problemas identificados. Las siguientes secuencias de capturas ilustran los flujos de trabajo clave en acción.

5.1. Flujo de Prueba E2E-01: Autenticación Unificada de Personal Clínico (SSO)

Este flujo demuestra la resolución del Problema 3. El objetivo es verificar que un usuario puede autenticarse una sola vez y acceder a múltiples sistemas.

- El proceso inicia cuando un médico intenta acceder al portal de OpenMRS. El sistema, en lugar de mostrar su propio login, redirige al usuario a la página de autenticación centralizada de Keycloak.
- Tras introducir sus credenciales únicas, el usuario es autenticado exitosamente por Keycloak y redirigido de vuelta a OpenMRS, donde se le concede acceso inmediato a su panel de control.
- Para confirmar la sesión de SSO, el médico abre el portal de Nextcloud en una nueva pestaña. El sistema lo reconoce y le da acceso directo a sus archivos sin solicitar nuevamente las credenciales, validando la experiencia de inicio de sesión único.



Iniciar sesión en Nextcloud

Iniciar sesión con nombre de usuario o cor...

Iniciar sesión con nombre de usuario o c

Contraseña

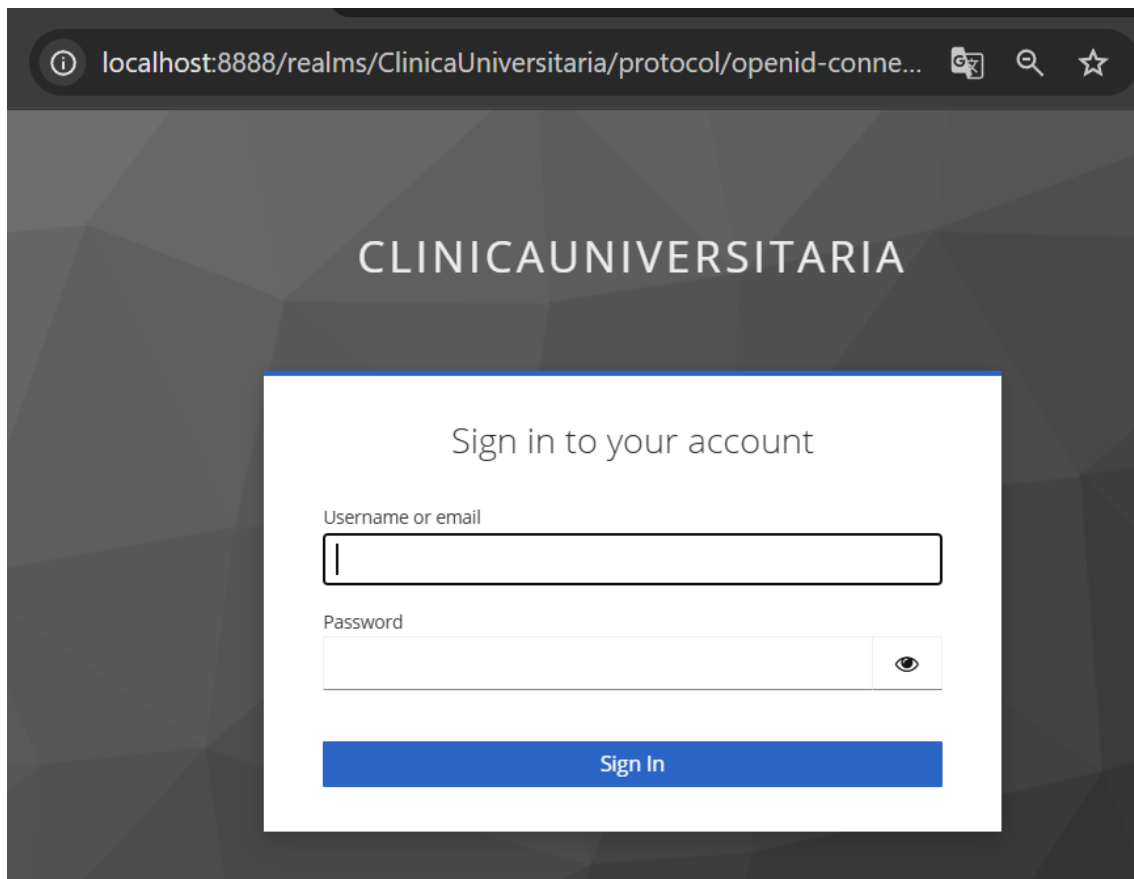


→ Iniciar sesión

¿Contraseña olvidada?

Iniciar sesión con dispositivo

Log in with Login Clinica



5.2. Flujo de Prueba E2E-02: Sincronización de Nuevos Pacientes (API RESTful)

Este flujo valida la solución al Problema 1, demostrando la eliminación de la doble entrada de datos.

- Primero, el personal de admisión registra a una nueva paciente, "Isabel Vega", en el sistema administrativo Odoo.

[ESPACIO PARA CAPTURA DE PANTALLA: Formulario de creación de contacto/paciente en Odoo]

- Tras la ejecución del script de sincronización, se realiza una búsqueda de la paciente "Isabel Vega" en el sistema médico OpenMRS. La evidencia muestra que el registro fue creado automáticamente con los datos correctos, confirmando el éxito de la integración.

Contacts / New

Individual Company

Ana Torres

Company Name...

Contact Street... Street 2... City State ZIP Country Tax ID ? e.g. BE0477472701

Job Position e.g. Sales Director

Phone

Mobile

Email

Website e.g. https://www.odoo.com

Title e.g. Mister

Tags e.g. "B2B", "VIP", "Consulting", ...

Contacts & Addresses Sales & Purchase Internal Notes

ADD

Send message Log note Activities

Mitchell Admin Creating a new record...

5.3. Flujo de Prueba E2E-03: Asociación de Documentos Clínicos (Transferencia de Archivos)

Este flujo demuestra la solución al Problema 2, integrando documentos externos al historial clínico.

- El personal de laboratorio sube un informe en PDF correspondiente a la paciente "Isabel Vega" en el directorio designado de Nextcloud.
- Después de que el script de integración procesa el archivo, el médico puede visualizar en la ficha de la paciente dentro de OpenMRS una nueva entrada en la sección de "Adjuntos" u "Observaciones", que contiene el informe de laboratorio.

5.4. Flujo de Prueba E2E-04: Validación del API Gateway como Punto de Acceso Seguro

Este flujo final valida la correcta implementación del componente avanzado, Kong.

- Se realiza una petición utilizando una herramienta como Postman. La petición no se dirige directamente a OpenMRS, sino al endpoint expuesto por Kong (<http://localhost:8000/openmrs/...>). La cabecera Authorization contiene el Bearer Token (JWT) obtenido de Keycloak. La captura muestra la petición y la respuesta 200 OK con los datos del paciente, probando que Kong enruta correctamente la petición y la valida exitosamente.

```
(venv) C:\Users\marti\OneDrive\Documentos\proyectos\Clinica_Uni_ProyectoInt_IntegracionSistemas\scripts>python odoo_to_o
penmrs_sync.py
2025-07-12 10:50:14,129 - INFO - =====
2025-07-12 10:50:14,129 - INFO - = Iniciando script de sincronización: Odoo -> OpenMRS =
2025-07-12 10:50:14,129 - INFO - =====
2025-07-12 10:50:14,707 - INFO - Conexión a Odoo exitosa.
2025-07-12 10:50:14,900 - INFO - Se encontraron 27 pacientes en Odoo para procesar.
```

6. Conclusiones y Reflexión Crítica

6.1. Conclusiones

Este proyecto ha demostrado de manera concluyente la viabilidad y el valor de una arquitectura de integración bien diseñada para resolver problemas operativos del mundo real en un entorno de salud. La implementación exitosa de un prototipo funcional ha permitido validar que es posible transformar una colección de sistemas aislados en un ecosistema tecnológico cohesionado, seguro y eficiente.

Se han alcanzado todos los objetivos propuestos, mitigando los tres puntos de fricción críticos identificados en el análisis inicial:

1. La **ineficiencia y el riesgo de error en el registro de pacientes** fueron resueltos mediante la automatización del flujo de datos desde Odoos a OpenMRS, aplicando el patrón de **invocación remota de APIs RESTful**.
2. El **acceso inseguro y desarticulado a documentos clínicos** fue solucionado a través de un proceso automatizado de **transferencia de archivos** entre Nextcloud y OpenMRS, centralizando la información en el historial clínico electrónico.
3. La **experiencia de usuario fragmentada y las brechas de seguridad** debidas a la gestión de identidades dispersa fueron eliminadas con la implementación de un robusto **Single Sign-On (SSO)** orquestado por Keycloak.

La incorporación de **Kong como API Gateway** no fue un mero requisito académico, sino una decisión arquitectónica fundamental que ha dotado a la solución de una capa de gobernanza y seguridad perimetral, actuando como un punto de control unificado y protegiendo los servicios de backend.

6.2. Reflexión Crítica y Trabajo Futuro

A pesar del éxito del proyecto, la naturaleza de la ingeniería de software es iterativa. Una reflexión crítica sobre la solución implementada revela varias vías para su evolución y madurez futura:

- **Evolución hacia una Arquitectura Orientada a Eventos (EDA):** Los scripts de sincronización actuales funcionan bajo un modelo de sondeo (polling), que puede no ser el más eficiente en tiempo real. La próxima iteración lógica sería refactorizar estas integraciones hacia una arquitectura orientada a eventos. Utilizando un **Message Broker como RabbitMQ**, Odoos podría publicar un evento **PacienteCreado** a un topic exchange. Un servicio consumidor suscrito a ese tema reaccionaría instantáneamente para crear el registro en OpenMRS. Este enfoque asíncrono mejora el desacoplamiento, la escalabilidad y la resiliencia del sistema.
- **Madurez de la Gobernanza de APIs:** Se implementó la validación de JWT en Kong. El siguiente paso sería explotar su ecosistema de plugins para una gobernanza más sofisticada. Se podría implementar el rate-limiting-plugin para prevenir abusos de la API, el prometheus-plugin para exportar métricas detalladas de tráfico, y gestionar la configuración del gateway como código mediante herramientas de "Infraestructura como Código" (IaC) como **deck**, integrándolo en un pipeline de CI/CD para despliegues automatizados y seguros.

- **Observabilidad Unificada:** Actualmente, cada componente genera sus propios logs. Un paso crucial hacia una operación de nivel de producción sería establecer una pila de observabilidad centralizada. Esto implicaría configurar los servicios para exportar logs en un formato estructurado (ej. JSON) a un agregador como **Loki**, métricas a **Prometheus**, y trazas distribuidas (si se avanza a microservicios) a **Jaeger**. Todo esto sería visualizado en dashboards unificados en **Grafana**, proporcionando una visión holística de la salud y el rendimiento del sistema integrado.
- **Consideración de un Service Mesh:** Si la arquitectura de la clínica evoluciona y los monolitos como OpenMRS o Odoo se descomponen en microservicios más pequeños, la gestión de la comunicación interna (Este-Oeste) se volverá compleja. En ese escenario, la introducción de un **Service Mesh como Istio** se volvería indispensable para gestionar de forma transparente el cifrado de tráfico interno (mTLS), el descubrimiento de servicios, los reintentos automáticos y los despliegues canary, complementando al API Gateway que seguiría gestionando el tráfico Norte-Sur.