UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN

SOFTWARE CON MENCIÓN EN DESARROLLO WEB Y MÓVIL

RPC-SO-34-NO.778-2021

OPCIÓN DE TITULACIÓN:

PROYECTO DE TITULACIÓN CON COMPONENTES DE INVESTIGACIÓN APLICADA Y/O DE DESARROLLO

TEMA:

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA SAAS PARA LA GESTIÓN AUTOMATIZADA DE INSTANCIAS DE ODOO CE

AUTORES:

MANUEL SEBASTIÁN BEDOYA ORTEGA

PAUL ESTEBAN GUZHÑAY LEMA

DIRECTOR:

CRISTIAN FERNANDO TIMBI SISALIMA

CUENCA – ECUADOR

2025

Autores:

Manuel Sebastián Bedoya Ortega
Ingeniero en Ciencias de la Computación.
Candidato a Magíster en Software por la Universidad Politécnica Salesiana – Sede Cuenca.
mbedoyao@est.ups.edu.ec

Paul Esteban Guzhñay Lema
Ingeniero en Ciencias de la Computación.
Candidato a Magíster en Software por la Universidad Politécnica Salesiana – Sede Cuenca.
pguzhnay@est.ups.edu.ec

Dirigido por:

Cristian Fernando Timbi Sisalima
Ingeniero en Sistemas.
Máster Universitario en Ingeniería del Software para la Web.
ctimbi@ups.edu.ec

DEDICATORIA

Quiero dedicar esta Tesis a mis padres Manuel y Mónica gracias por todo su apoyo y su compañía, gracias por estar en los momentos en los que ya me queria rendir teniéndome paciencia y demostrándome que a pesar de tener muchas dificultades siempre llega algo mejor.

También quiero agradecer a mis hermanos Juan José y a la distancia a mi hermana María Elisa quienes con su ejemplo me guiaron para seguir mis sueños sin importar las distancias o las dificultades mientras tu creas en algo y tengas confianza nada es imposible, a mis abuelos maternos que desde el cielo me cuidan y me dan la sabiduría para seguir adelante y a mis abuelos paternos que todos los días oran por mi deseándome un futuro mejor a pesar de los tiempos difíciles siempre debemos seguir con una sonrisa.

Quiero dedicar también a mi novia y mejor amiga Vanessa Tinizaray la cual me enseña todos los días que puedo ser mejor y que no debo rendirme, quien me apoya en los momentos difíciles y siempre esta para mí, gracias por cuidarme y acompañarme en este camino y enseñándome que con esperanza y valentía todo puede ser posible.

DEDICATORIA

Primero quiero agradecer a Dios, por permitirme cumplir una nueva meta en mi vida.

Esta tesis está dedicada a mi padre y a mi madre, quienes han sido las personas mas importantes en mi vida. Gracias por impulsarme siempre a seguir mis sueños y por enseñarme que, aunque uno puede caer, siempre hay que levantarse y continuar. Les agradezco infinitamente por sus palabras de aliento, por estar siempre pendiente de mí y por esos abrazos, caricias llenas de amor. Los amo papi y mami.

También quiero dedicar este trabajo a mis dos abuelitas, Rosa y Gerardina. A mi abuelita Gerardina que está en el cielo y es un ángel en mi vida, sé que me está cuidando y protegiendo y que se siente feliz por este nuevo logro que he completado. Mi abuelita Rosa, la que es mi segunda madre, le agradezco siempre por ser esa abuelita cariñosa y por recibirme siempre con un abrazo lleno de amor, por estar para mis todas las veces que le he necesitado en mi vida y darme esas palabras de aliento.

A mi mejor amiga de este mundo, Erika, gracias por ser esa persona que me cuida como a un hermano menor, por tus consejos, tu apoyo y por recordarme que no debo rendirme. Nuestra amistad es de esas que solo ocurren una vez en la vida. A pesar de los miles de kilómetros que nos separan, siempre estamos pendiente de cada uno. Te quiero mucho Erika.

A mi mejor amiga de aventuras, Pris, por estar siempre lista para acompañarme en cualquier experiencia. Gracias por las risas que me regalas cada día y por convertirte en esa persona que uno no busca, pero que la vida pone en el camino porque realmente se necesita.

AGRADECIMIENTO

Agradezco a la vida por darme la oportunidad de cumplir este objetivo dándome valentía y confianza para cumplir las metas propuestas.

Agradezco a mis padres por depositar su confianza plena en mi y por estar pendientes de todo, los amo con todo mi corazón, a mi ñaño que siempre esta pendiente de lo que necesito y me apoya gracias por todo ñaño te amo, y a mi ñaña que a pesar de la distancia quiero que sepas que estoy muy orgulloso de todo lo que tus haces y que siempre voy a estar agradecido por ayudarme y aconsejarme para estar en el camino correcto te extraño mucho y te amo.

A mi tutor de Tesis Cristian quien nos instruyo y nos ayudo a mejorar este trabajo de titulación.

A mi compañero de Tesis Paul, gracias no tengo palabras para expresar todo el apoyo y compañía que me has dado en esta etapa, por estar en los momentos difíciles y por compartir también muy buenos momentos.

A mis amigos Pedro, Daniel, Daniel Gonzales, Yami, Danny, Sebastián y Tracy quienes nunca me dejaron solo y me escucharon cuando más lo necesite.

Y a mi Vane gracias por apoyarme todos los días, escucharme y estar pendiente de mí, te amo eres mi fuente de energía y esperanza.

AGRADECIMIENTO

Agradezco a Dios por darme la sabiduría y la fuerza para poder completar este logro importante en mi vida.

A mis amados padres por darme el impulso de perseguir y luchar por mis sueños, por enseñarme a levantarme y a salir adelante. No existen palabras suficientes para expresar lo agradecido que estoy con ustedes. Los amo profundamente, papi y mami.

A mi tutor, Cristian, por su guía académica, su paciencia y sus valiosos comentarios que ayudaron a mejorar este trabajo.

Agradezco también a mi compañero de tesis, Sebastián, gracias por tu apoyo constante, por permitirme compartir este camino lleno de desafíos y aprendizajes, y por tu amistad siempre verdadera y sincera. Te agradezco también por las risas y los buenos momentos que compartimos.

A mi familia que me impulsó a seguir luchando por mis sueños, gracias abuelito Julio, Pablo, Carlos, Mateo, Monse, Dome, Tia Ali, Tia Rocio, Maya, Daniela, Fernando, gracias por cada una de sus palabras de aliento.

A mis amigos Pedro, David, Danny, Daniel López, Daniel Gonzales y Tracy, gracias por su apoyo, consejos y amistad. Aunque la distancia nos separa, su compañía y motivación siempre estuvieron presentes.

A mi segunda familia, Juan José, Olguita, Pedro, Teffy y Pao, gracias por su amistad sincera, por las risas compartidas y por ayudarme tantas veces a comprender la vida desde otra perspectiva. A todos los que, de una u otra manera, formaron parte de este proceso, muchas gracias.

Finalmente, a todas las personas que, de una u otra forma, formaron parte de este proceso, les expreso mi más sincero agradecimiento.

Tabla de Contenido

Índice de Figuras

Índice de Contenidos

DISEÑO E IMPLEMENTACIÓN DE UNA PLATAFORMA SAAS PARA LA GESTIÓN AUTOMATIZADA DE INSTANCIAS DE ODOO CE

AUTOR(ES):

MANUEL SEBASTIÁN BEDOYA ORTEGA
PAUL ESTEBAN GUZHÑAY LEMA

Resumen

En el ámbito empresarial los sistemas ERP sirven para manejar distintos procesos del negocio, la facilidad que este sistema presta es el poder agrupar todos estos procesos dentro de un sistema único que ayude a automatizar tareas y la toma de decisiones con el fin de mejorar el rendimiento dichas empresas. Sin embargo, el mantenimiento y el despliegue pueden alcanzar un alto nivel de complejidad, lo que afecta a los tiempos de entrega y la capacidad de personalización. Estos factores resultan críticos debido a que las organizaciones necesitan eficacia al momento de implementar un nuevo sistema y un alto grado de adaptación de este a sus necesidades específicas como modificaciones de reportes o desarrollos de lógicas que impactan directamente al flujo de su negocio, requisitos que son determinantes a la hora de seleccionar una herramienta de gestión empresarial, lo que en muchos casos pueden convertirse en una limitación debido a la complejidad de sus desarrollos, ciertas actualizaciones y mantenimiento especialmente para empresas pequeñas y medianas que requieren soluciones más simples y accesibles . Debido a la complejidad que generan al utilizar estas herramientas, se plantea una plataforma SaaS que automatiza el ciclo de vida de las instancias de odoo Community Edition. Esto se logra mediante despliegue continuo, copias de seguridad y clonación de entornos de desarrollo, pruebas y produccion con el fin de mantener un ambiente estable para el cliente. De esta manera los posibles desarrollos y optimizaciones pueden ser evaluadas en un entorno independiente con datos reales reduciendo así complejidad operativa y extendiendo la utilización de sistemas a empresas con menos recursos económicos mediante la utilización del software Odoo Community Edition (CE). Se plantea una solución la cual utilizará varias tecnologías acopladas de forma que la comunicación sea eficientes y confiable en el momento que exista intercambio de información sensible para los usuarios optima. Como eje central, se implementó un Backend en Spring boot encargado de orquestar Servicios de Docker, que a su vez tiene comunicación con un frontend en Angular que proporciona interfaces para las gestiones del entorno

virtual. El desarrollo de este proyecto se lo realizo con un enfoque orientado en el metodo cascada el cual cuenta con diferentes fases las cuales son útiles para garantizar la utilidad de un sistema escalable el cual optimiza costos operativos y simplifica la administración de instancias de Odoo CE.

Palabras clave:

Docker, Spring boot, Angular, Odoo CE, Entidades, SaaS, Despliegue Continuo.

Abstract

In the business environment, ERP systems are used to manage various business processes. The advantage of these systems lies in their ability to group all these processes within a single system, helping to automate tasks and support decision-making with the goal of improving company performance. However, their deployment is often complex, slow, and costly, which limits their adoption by small and medium-sized enterprises.

To address this, a SaaS platform is proposed that automates the lifecycle of Odoo Community Edition instances through continuous deployment, backups, and cloning of development, testing, and production environments. This system reduces operational complexity and extends the use of ERP systems to companies with fewer economic resources by leveraging Odoo Community software. A solution is proposed that integrates several technologies in a way that ensures optimal communication. At the core is a backend developed in Spring Boot that orchestrates Docker services, along with a frontend built in Angular that offers intuitive interfaces for the different capabilities of our digital environment. The project was developed with an agile approach, including iterations and acceptance testing, to ensure the usefulness of a scalable system that optimizes operational costs and simplifies the management of Odoo CE instances.

Keywords:
Docker, Spring Boot, Angular, Odoo CE, Entities, SaaS, Continuous Deployment.

Glosario

API (Application Programming Interface): Conjunto de reglas que permiten la comunicación entre diferentes aplicaciones, facilitando el intercambio de datos e integración entre sistemas.

Angular: Framework de desarrollo sostenido por Google que permite gestionar crear y modificar interfaces web responsivas mediante el uso de componentes reutilizables los cuales estan programados en el lenguaje Typescript. (Google Developers, 2025)

Backend: Es la parte donde se desarrolla la lógica de un sistema encargado de la transmisión de los datos la seguridad y el flujo de la lógica del negocio para un correcto funcionamiento de un sistema.

Base de datos PostgreSQL: Sistema de gestión y manejo de bases de datos relacionales además de ser de libre uso para los usuarios, es conocido por ser escalable, robusto y con una capacidad muy buena en el manejo de transacciones y manejos de datos complejos.

Contenedor Docker: Entorno ligero que agrupa aplicaciones junto con sus dependencias, con el objetivo de ejecutar de una manera uniforme sistemas en diferentes sistemas operativos y entornos.

Continuous Deployment (Entrega continua): Practica de desarrollo de software que permite mantener el codigo con facilidad de despliegue en todo momento además de tener procesos de integración y pruebas automatizadas.

DevOps: Practicas que integran desarrollo de software y operaciones de TI (Tecnologías de la información) siendo su objetivo disminuir los ciclos de entrega y mejora la confiabilidad en los sistemas en los que se aplican estos ejercicios. (Forsgren, Humble, & Kim, 2018)

Dockerización: Procesos de empaquetar una aplicación y dependencias en contenedores para facilitar su portabilidad y despliegue en otros entornos. ((AWS), 2024)

ERP (Enterprise Resource Planing): Sistema de planificación de recursos empresariales el cual se encarga de integrar dentro de una plataforma procesos clave de una organización, tales como ventas, finanzas y recursos humanos. (Odoo , 2024)

Frontend: Es la parte visible de la aplicación se encarga de la presentación de datos y visualizaciones para el usuario a través de interfaces graficas que pueden llegar a ser muy intuitivas para el cliente.

Git: Sistema de control de versiones que sirve para gestionar cambios en el codigo de un proyecto de manera más eficiente, mejorando y optando por la facilidad y rapidez en el trabajo colaborativo.

GitHub: Plataforma en la nube basada en git para la facilitación de creación uso y mantenimiento de proyectos mediante repositorios e integración con herramientas de desarrollo.

Instancia de Odoo: entorno independiente de odoo que contiene modulos, configuraciones y datos para un proyecto u empresa la cual utiliza el sistema odoo.

Metodología Cascada: Modelo de desarrollo de software que se encamina completar todo el sistema mediante el uso de fases como análisis, diseño, desarrollo, etc. Se debe completar la fase anterior por completo para continuar en la siguiente.

Neutralización de datos (Stagging): Técnica para replicar un entorno de produccion en un entorno de pruebas, la cual utiliza datos reales, pero con componentes sensibles como el envio de correos o la comunicación con sistemas externos deshabilitado.

Odoo CE (Community Edition): Versión de Odoo de codigo abierto, un ERP modular que puede adaptarse a las necesidades de diferentes empresas.

Odoo.sh: Plataforma de gestión de instancias de Odoo que mezcla características de SaaS y PaaS, con la cual se puede integrar desarrollos y administrar entornos necesarios para el mantenimiento de Odoo Enterprise.

Orquestación de contenedores: Conjunto de procesos y herramientas que permiten mantener el despliegue de múltiples contenedores dockerizados de manera eficiente.

PYMES (Pequeñas y Medianas Empresas): Empresas que cuentan con ciertas características que los diferencian a grandes corporaciones las cueles pueden ser un motor fundamental en la economía. (Servicio Ecuatoriano de Normalización (INEN), 2021)

REST API: Interfaz de programación de aplicaciones que sigue la arquitectura REST, utilizando métodos de comunicación HTTP.

SaaS (Software as a Service): Modelo de distribución en el cual las aplicaciones se alojan en la nube y los usuarios acceden a sus servicios mediante la red. ((AWS), 2024)

Spring Boot: Framework basado en java el cual sirve para simplificar la utilización y la creación de aplicaciones y microservicios, al ofrecer configuración automática. (Spring, 2025)

Virtualización: Tecnología que sirve para crear entornos virtuales optimizando el hardware y reduciendo costos.

Volumen de Docker: Mecanismo el cual permite que datos, información y configuraciones se puedan almacenar fuera de contenedores, logrando así desacoplar el ciclo de vida de los datos del contenedor

1. Introducción

Dentro del manejo de empresas en la actualidad, los avances en la tecnología y la innovación digital se ha ido transformando en una necesidad que busca mantener la capacidad de competitividad, eficiencia y adaptación a las exigencias del mercado. En este escenario, los sistemas ERP cumplen acciones fundamentales dentro de la integración y automatización de procesos internos, así como la organización dentro de la empresa, gracias a esto es sencillo lograr tener una visión más amplia de áreas como finanzas, inventario o proyectos.

Tomando en cuenta todos sus beneficios, los ERP sufren de una gran problemática la cual es el mantenimiento y la implementación debido a que sigue siendo un proceso complejo y costoso, lo que es un obstáculo directo que limita la acogida de estos programas por parte de las pequeñas y medianas empresas.

Con la ayuda de Odoo CE, el cual es un software de codigo abierto, es aceptable como solucion para las PYMES debido a su flexibilidad, modularidad y bajo costo. Sin embargo, para su correcto manejo, mantenimiento y configuración se requiere de personal con el conocimiento adecuado a cerca de este sistema que no siempre están disponibles en las pequeñas y medianas empresas, restringiendo y evitando que se pueda aprovechar el uso de un sistema con tantas funcionalidades en su totalidad.

Tomando en cuenta todas estas situaciones, esta tesis plantea el diseño e implementación de una que implementara la tecnología de Software as a Service (SaaS) que este encargado de la gestión de instancias de Odoo CE. Esta solución permitirá realizar tareas como el despliegue automatizado, la configuración inicial, la gestión de usuarios, la creación de copias de seguridad, la clonación de entornos de prueba y otras funciones clave mediante una interfaz web intuitiva y accesible. Mas allá de simplificar el proceso de instalación, la propuesta se centra en reducir las dificultades relacionadas con el mantenimiento, la actualización y la evolución que pueden determinarse como los principales retos técnicos y económicos a lo largo de la vida del software. Dichas

actividades requieren tiempo, lo que suele dificultar su gestión en empresas pequeñas y en crecimiento ya que cuentan con poco personal Al ofrecer esta plataforma como servicio, se busca no solo disminuir la complejidad operativa dino también garantizar el mantenimiento actualización y escalabilidad en el uso de Odoo CE, eliminando las barreras técnicas y económicas que actualmente limitan el aprovechamiento de todo el potencial que este tipo de soluciones puede ofrecer en entornos con recursos limitados.

2. Determinación del Problema

Los sistemas ERP (Enterprise) ERP (Enterprise Resource Planning) se han consolidado como herramientas clave para la integración de las operaciones empresariales, permitiendo a las organizaciones gestionar de manera eficiente diversas funciones como finanzas, recursos humanos, inventarios y ventas. Los sistemas ERP se encargan de contener toda la información de una empresa en un solos sistema, lo que contribuye a la mejora operativa y reduce costos al eliminar procesos innecesarios y confirmado en investigaciones más recientes (García, 2021).

La acogida de un ERP ha tomado mucha relevancia en el contexto de la transformación digital de empresas, debido a que promueve el trabajo colaborativo entre departamentos de la empresas y mejora en la visibilidad de los procesos internos. En los últimos años, el modelo de despliegue de los ERPs ha ido evolucionando con el paso de soluciones tradicionales on-premise a modelos que estan integrados en la nube, conocidos como SaaS (Software como Servicio). Este modelo ofrece varias ventajas, entre las más conocidas son la accesibilidad remota, actualizaciones automáticas, la escalabilidad y reducción de costos asociados con infraestructura y mantenimiento (Usman, Ahmad, Zakaria, & Alkurdi, 2017). El uso de ERP como servicio a permitido que las pequeñas y medianas empresas puedan acceder a tecnologías avanzadas sin los costos elevados asociados con la implementación de soluciones locales. Sin embargo, las pequeñas y medianas empresas enfrentan limitaciones al intentar migrar su información a herramientas como Odoo Community Edition (Odoo CE). Al ser una plataforma Open Source y con funcionalidades altamente extensas y útiles, el

despliegue, configuración y mantenimiento dentro de un ambiente privado como servidores propios de cada empresa requieren conocimientos técnicos de alto nivel para el mantenimiento, consultas y manejo del sistema, debido a esto se genera una problemática significativa la cual impide que empresas puedan acceder a sus servicios, esto evidencia la falta de una plataforma que acceder a las PYMES, gestionar instancias de odoo sin necesidad de tener algun conocimiento avanzado ni hacer grandes inversiones en infraestructura, debido a esto se plantea una problemática en concreto y es: ¿Cómo se facilitaría la adopción de Odoo Community Edition en PYMES mediante una solucion que automatice su despliegue, configuración y mantenimiento bajo diferentes modelos y con la facilidad de poder ser escalable?

3. Objetivos generales y específicos

En esta sección, presentaremos la descripción de los objetivos que hemos establecidos para el desarrollo del proyecto. Los detalles específicos se van a detallar en los objetivos generales y específicos.

3.1 Objetivo General

Desarrollar una plataforma basada en un modelo SaaS que permita la gestión integral, automatizada y eficiente del despliegue, configuración y mantenimiento de instancias personalizadas de Odoo CE, facilitando su administración y escalabilidad para los usuarios finales

3.2 Objetivos Específicos

- Diseñar la arquitectura de la plataforma SaaS que permita gestionar de manera automatizada el despliegue, configuración y mantenimiento de instancias personalizadas de Odoo CE. Esto se logrará mediante el análisis de requisitos técnicos, la selección de herramientas adecuadas y la definición de procesos eficientes de automatización, con el propósito de establecer una base escalable y eficiente que soporte las necesidades de los usuarios.

- Desarrollar funcionalidades específicas para la gestión de usuarios y la automatización de copias de seguridad de las instancias

- Implementar un sistema automatizado de clonación de datos que facilite la réplica de entornos de producción en entornos de staging

- Desarrollar una interfaz web intuitiva que permita a los usuarios finales gestionar sus instancias de manera sencilla. Esto se realizará empleando tecnologías web modernas para diseñar una interfaz funcional y adaptable, con el propósito de simplificar la administración, y garantizar una experiencia de usuario amigable y eficiente.

4. Marco teórico referencial

La transformación digital ha impulsado la optimización de procesos con el propósito de mejorar los tiempos de respuesta y poder facilitar el uso de las herramientas de gestión. En este contexto, los sistemas ERP se posicionan como herramientas clave para centralizar operaciones, evitar brechas de datos y ofrecer una visión integral del negocio que facilita la toma de decisiones de manera efectiva y segura (Haro, y otros, 2023)

Dentro de las principales alternativas de ERP se encuentran SAP, Oracle Sage, Odoo, entre otras. De estas destaca Odoo por su edición Community de codigo abierto, cuyo carácter modular y adaptable permite ajustarse a las necesidades heterogéneas de las organizaciones (Bengochea, 2025) Esta característica constituye la base de la presente propuesta, al considerar Odoo CE como plataforma tecnológica para la construcción de la solución.

Si bien Odoo CE ofrece flexibilidad y capacidad de implementación para personalizaciones requeridas, estos desarrollos necesitan mantenimiento y soporte los cuales pueden implicar la necesidad de un equipo de profesionales para estas tareas. En este escenario puede ser una útil herramienta la utilización de Odoo.sh, una plataforma

SaaS/PaaS oficial que se encarga de tener un acceso más rápido y eficaz para gestionar entornos, además del mantenimiento de personalizaciones. Aunque Odoo.sh no forma parte de Odoo CE, representa una referente clave para una automatización en la nube que inspira la presente propuesta de investigación, manteniendo la versión Community como clave por su carácter de codigo abierto y acceso gratuito a las empresas.

Para guiar el desarrollo de la solución, se adopta la metodología en cascada, un modelo de ciclo de vida del software que se organiza en diferentes fases las cuales serán descritas a continuación. Este enfoque permite avanzar de manera estructurada y controlada, asegurando el avance del proyecto de manera estructurada y controlada en cada etapa. La aplicación continua en investigaciones recientes que demuestran que es un marco adecuado para proyectos con requisito claramente definidos (Barghoth, 2020).

Finalmente, se introduce la tecnología de contenedores Docker como soporte fundamental, para la estandarización del cuerpo de la configuración de entornos con sus dependencias para el correcto funcionamiento de las entidades de Odoo.

4.1. Transformación Digital y automatización de procesos

4.1.1. Alcance de la transformación digital

La transformación digital no se limita únicamente a procesos de negocio, sino que en el ámbito de los departamentos de TI y empresas de consultoría tecnológica representa un cambio estructural en la manera en que se gestionan los entornos de software. De acuerdo con (Páez-Gabriunas, 2021) la transformación digital debe entenderse como un proceso continuo que reconfigura la gestión tecnológica mediante la automatización de despliegues, provisión de entornos y mantenimiento de aplicaciones críticas. En este sentido, los diferentes equipos de TI logran trabajar con datos y sistemas en tiempo real, reduciendo la complejidad operativa y mejorando la fiabilidad de sus servicios (Marques & Ferreira, 2021)

4.1.2. Beneficios y retos para las organizaciones

En el contexto de las empresas de desarrollo y consultoría, la transformación digital nos permite:

- Optimizar el ciclo de vida del software, reduciendo tiempos en despliegues de entornos
- Automatizar tareas criticas como backups y restauraciones
- Escalar proyectos de forma más sencilla, al replicar entornos de producción, staging y desarrollo
- Mejorar la trazabilidad de cambios, facilitando auditorias y control de versiones

Sin embargo, entre los principales retos se encuentran la escasez de talento especializado en DevOps y contenedores, las limitaciones de infraestructura en PYMEs tecnológicas y la resistencia a la adopción de nuevas metodologías (Medallo, 2020). Estos factores demuestran la necesidad de la plataforma que simplifiquen la administración de instancias y reduzcan la complejidad técnica

4.1.3. Automatización de procesos de negocios

La automatización de los procesos en TI (BPA, Business Process Automation) consiste en usar tecnologías digitales para ejecutar tareas repetivas relacionadas con la gestión de software, como es la creación de contenedores, la configuración de redes internas o la restauración de entornos. Esto nos permite reducir errores y liberar recursos humanos para actividades de mayor valor. Según (Businessmap, 2023) esta práctica ya no está limitada a grandes corporaciones, sino que es viable también en PYMEs tecnológicas. Actualmente, la automatización está evolucionando hacia la automatización inteligente de procesos (IPA), que integra IA y analítica

avanzada, aplicada a la supervisión de despliegues y mantenimiento de infraestructuras (Appian, 2023)

4.1.4. Factores críticos para el éxito de Pymes

La adopción de la transformación digital en PYMEs dedicadas al desarrollo o consultoría de diversos factores críticos (SAP España, 2023) identifica como esenciales:

- Compromiso del liderazgo para guiar el cambio a medios digitales
- La capacitación a los empleados, garantizando el buen uso del sistema
- Selección de herramientas adecuadas, con soluciones escalables y compatibles con los datos existentes en una empresa
- Gestión del cambio organizacional, para reducir la resistencia interna y acelerar la transición

Estos factores son determinantes para que la transformación digital en entornos de TI se llegue a transformar en eficiencia operativa y sostenibilidad tecnológica en el tiempo

La transformación digital ha impulsado la optimización de procesos con el propósito de mejorar los tiempos de respuesta y facilitar el uso de herramientas de gestión. En este contexto, la automatización se presenta como un eje estratégico para las organizaciones, permitiendo integrar operaciones, disminuir errores humanos y garantizar mayor agilidad en la toma de decisiones (Haro, y otros, 2023). Para los departamentos de TI, este contexto se encarga de generar la necesidad de soluciones que integren todos los procesos empresariales además de ser escalables y con una capacidad muy buena de adaptación a todos los entornos. Esta búsqueda constante de eficiencia y personalización ayuda ah impulsar la implementación de software ERP, los cuales son capaces de centralizar funciones clave y convertirse en un soporte muy importante para las actividades de las empresas.

4.2. ERP como solución para la gestión de empresas

4.2.1. Integración y optimización de procesos empresariales mediante ERP

Los sistemas de gestión empresarial pueden integrar muchas áreas de la organización en una sola plataforma como contabilidad, nómina de empelados, produccion, inventario y ventas.

Para los departamentos de TI esto integración representa una forma de reducir la necesidad de dependencia de diferentes aplicaciones y la necesidad de conectarlas, además de facilitar la interoperabilidad entre procesos y mantener la consistencia de la información de la empresa. La unificación de registros no solo evita la duplicación de información, sino que también simplifica la administración técnica de la infraestructura, logrando así una disminución en los puntos de fallo y logrando la estandarización en la gestión. Los ERP han evolucionado de simples sistemas a plataformas digitales con mayor inteligencia gestionando datos en tiempo real, reduciendo costos operativos y transparencia en la recepción de información (Klaus, Rosemann, & Gable, 2023)

4.2.2. QERP como soporte para la transformación digital en PYMEs

Con el tiempo, los ERP dejaron de ser únicamente herramientas para convertirse en motores que impulsan la transformación digital, especialmente en pequeñas y medianas empresas. Para los equipos de TI, esta transformación significa contar con herramientas modulares y

escalables que les permitan implementar nuevas funcionalidades sin rehacer toda la infraestructura, responden más rápido a los cambios del mercado y reducir procesos manuales que suelen recaer en desarrollos internos. De acuerdo con investigaciones realizadas en el año 2021 la adopción de sistemas de gestión empresarial mejora la capacidad de innovación al mismo tiempo que reduce la complejidad en procesos manuales (Monk, 2021).

Los sistemas ERP (Enterprise Resource Planning) se han consolidado como herramientas clave para centralizar procesos y garantizar la coherencia de la información. Entre las alternativas más destacadas se encuentran SAP y Odoo, cada uno con distintos niveles de flexibilidad y costos asociados. Sin embargo, para departamentos de TI que buscan soluciones sostenibles y adaptables, Odoo CE se destaca por su flexibilidad, facilidad de adaptación y, ademas, por ser una alternativa gratuita. Esto lo convierte en una opción óptima para ser utilizada como base tecnológica, siempre y cuando el área de TI cuente con el conocimiento técnico para su despliegue, configuración y personalización.

4.3. Odoo Community

Odoo es un Sistema de planificación empresarial creado en Bélgica en el año 2005 el cual es flexible y facilita los procesos empresariales, su estructura es modular permite a las organizaciones que utilizan el software la capacidad para personalizar y ampliar las funcionalidades para cumplir sus necesidades operativas (Gharabti, El Kortbi, Nekhass, & Bendahmane, 2025).

Este sistema cuenta con dos distribuciones: Community y Enterprise. En el presente trabajo se toma como base la Community Edition de codigo abierto y con funcionalidades suficientes para cubrir las necesidades básicas de una empresa. Gracias a su comunidad activa, Odoo CE dispone de más de 40.000 aplicaciones que extienden sus capacidades, ademas de foros y redes de colaboración que impulsan su desarrollo y soporte. Actualmente, Odoo supera los 12 millones de usuarios a nivel mundial, quienes han migrado o adoptaron este programa para poder liberar todo su potencial de crecimiento (Odoo , 2024).

Inicialmente conocido como OpenERP, el sistema evolucionó hacia Odoo, incorporando mejoras significativas en sus módulos y alcanzado en la actualidad la versión 18.0, que incluye funcionalidades avanzadas en áreas como marketing digital y gestión de soporte en línea.

4.3.1. Implementación de odoo CE

Para la implementación de odoo Community en servidores locales tenemos que tomar en cuenta ciertos prerrequisitos. Entre los principales se encuentran una versión adecuada de Python con sus librerías correspondientes, la herramienta wkhtmltopdf para la generación de reportes en formato PDF y una base de datos de PostgreSQL, encargada de la persistencia de los datos (vrajatechnologies, 2024).

4.3.2. Limitaciones de odoo

Debido a su gran flexibilidad, Odoo puede llegar a traer consigo complicaciones debido a las constantes actualizaciones y gestión del sistema. Las personalizaciones realizadas pueden presentar complicaciones con estos cambios, lo que vuelve el mantenimiento más complejo y demandante en tiempo y costo. Además la estructura de odoo al igual que de otros ERPs puede llegar a presentar dificultades durante su configuración inicial a lo que es necesario la existencia de un experto en manejo de servidores y despliegue de aplicaciones (Velneo, 2024).

Odoo Community Edition (Odoo CE) es la versión de código abierto del ERP, caracterizada por su modularidad, capacidad de personalización y una amplia comunidad de desarrolladores que aportan miles de aplicaciones adicionales (Bengochea, 2025). Estas características la convierten en una plataforma accesible para pequeñas y medianas empresas, lo que justifica su selección como núcleo de la propuesta. No obstante, las limitaciones han dado paso a sistemas complementarios como Odoo.sh, que sirve de referencia para el desarrollo de soluciones más automatizadas y escalables.

4.4. Odoo.SH

Con el crecimiento de las implementaciones de Odoo a nivel mundial, surgió Odoo.sh, una plataforma oficial de alojamiento orientada principalmente a empresas que requieren entornos integrados y escalables.

Esta solución combina las ventas de PaaS (Plataforma como servicio) e IaaS (Infraestructura como servicio) consiguiendo que se pueda administrar gestionar y modificar las aplicaciones durante todo su ciclo de vida desde la etapa de desarrollo como la de pruebas (Solano María José [Vauxoo], 2025).

Entre sus principales características se encuentran:

- Manejo de ramas: desarrollo, staging y producción, facilitando la integración y pruebas antes del paso a producción (Cybrosys Technologies, 2025).
- Automatización de copias de seguridad: con la posibilidad de generar respaldos en cualquier momento
- Acceso por líneas de comandos: que brinda a los administradores controlo directo sobre los servidores creados.

Para finalizar existe un manejo de servidores mediante la opción de entrada a la línea de comandos del servidor de la entidad creada por el sistema odoo sh (Solano María José [Vauxoo], 2025).

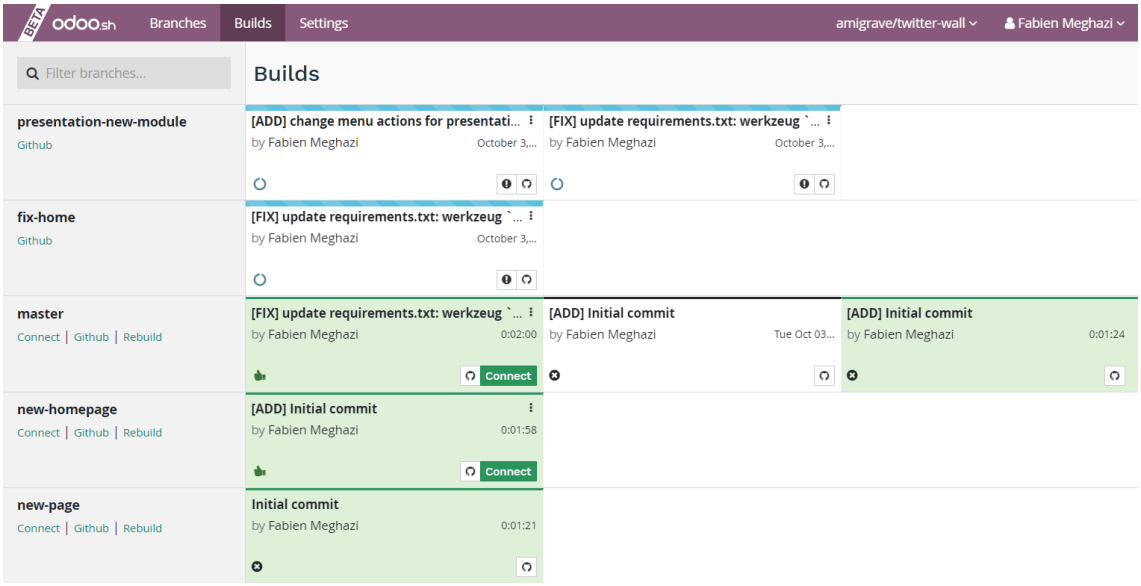


Figura 1. Odoo sh

Con el crecimiento global de las implementaciones de Odoo surgió Odoo.sh, una plataforma SaaS/PaaS oficial que integra despliegues, entornos de prueba, copias de seguridad y ramas de desarrollo de forma automatizada (Cybrosys Technologies, 2025).Si bien Odoo.sh solo funciona con la versión de Odoo Enterprise sus funcionalidades y la facilidad de mantenimiento y soporte para las empresas ha sido clave fundamental para la implementación de este sistema y para materializar de manera más eficiente y practica se propuso una metodología la cual organiza mejor las etapas del ciclo de vida del software y es la metodología cascada.

A partir del analisis de las funcionalidades que ofrece Odoo.sh, se identificó la necesidad de implementar una alternativa que permita replicar su comportamiento dentro de un entorno institucional o local, sin depender de licencias de propietarias ni de infraestructura externa. En este contexto, surge la adopción de Docker como tecnología base para la creación de los entornos aislados, escalables y reproducibles, capaces de gestionar instancias independientes de Odoo Community con características similares a las ofrecidas por Odoo.sh. Esta herramienta se convierte así en el pilar fundamental para la automatización del despliegue, mantenimiento y gestión del ciclo de vida de cada instancia dentro de la plataforma desarrollada

4.5. Docker

Es una plataforma que permite implementar aplicaciones rápidamente mediante el proceso de empaquetado de software para hacerlo en unidades estandarizadas los cuales se llaman contenedores que incluyen todas las especificaciones incluidas bases de datos, codigo y herramientas para que el sistema que este contenido se ejecute con normalidad utilizando aislamiento de recursos basados en el kernel de Linux como son namespaces para permitir que estas agrupaciones sean independientes dentro de una sola instancia.

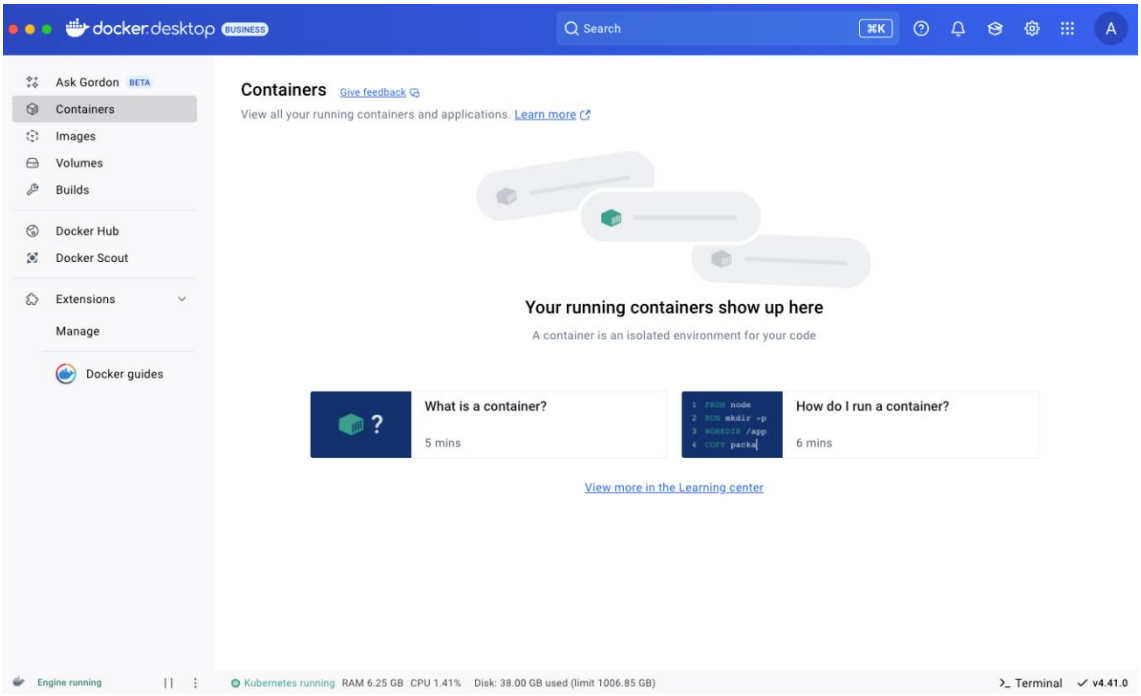


Figura 2. Menú de Docker

4.5.1 Virtualización

Este proceso sirve para la creación de instancias virtuales usando recursos reales como servidores físicos y redes permitiendo de esta manera que varias instancias puedan funcionar de manera aislada sin afectar a otras instancias y todas funcionando dentro del mismo entorno físico, tiene muchas ventajas, pero en comparación con el uso de Docker puede quedarse atrás debido a que Docker. Este simplifica el arranque de aplicación preconfiguradas. Y gracias al aumento de la comunidad que utiliza estos servicios empaquetados los cuales pueden ser arrancados sin necesidad de tener un conocimiento avanzado para poder

proporcionar aplicaciones empaquetadas a sus propios clientes (arteco-consulting, 2023).

4.5.2. IMÁGENES DOCKER

Una imagen de Docker es un fichero que es independiente el cual se utiliza para crear contenedores, este puede traer en su interior bibliotecas, dependencias y archivos necesarios para la correcta ejecución de un contenedor. Estas imágenes son fáciles para compartir y mover, debido a esa razón se puede utilizar la misma imagen para varios contenedores a la vez. Puede tener varias funcionalidades una de las más complejas es el seguimiento de sistemas complejos. En este proyecto se utilizan las imágenes de Odoo y de PostgreSQL para la creación de entidades de odoo CE mediante una pantalla en la que se elige la entidad y se crea el contenedor con todo lo necesario para su arranque (Amazon web service, 2025).

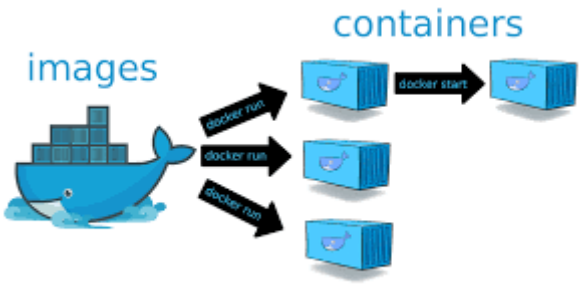


Figura 3. Imágenes de Docker

4.5.3. CONTENEDORES DOCKER

Un contenedor sirve para para guardar las imágenes y todas las dependencias para que la aplicación se ejecute de forma rápida, son una forma de virtualización que puede ejecutar desde un microservicio hasta un proceso de software de tamaño mayor, en este pueden introducir ejecutables, codigo binario además de archivos de configuración necesario y a diferencia de la utilización de máquinas virtuales; los contenedores no necesitan tener un SO(sistema operativo), esto los hace más ligeros y portables, teniendo así una carga relativamente menor,

trabajando en desarrollos muy amplias se pueden utilizar varios contenedores todos estos pueden ser gestionados mediante un orquestador el cual se encarga de la nivelación de carga de los contenedores y de su configuración (NetApp, 2025).

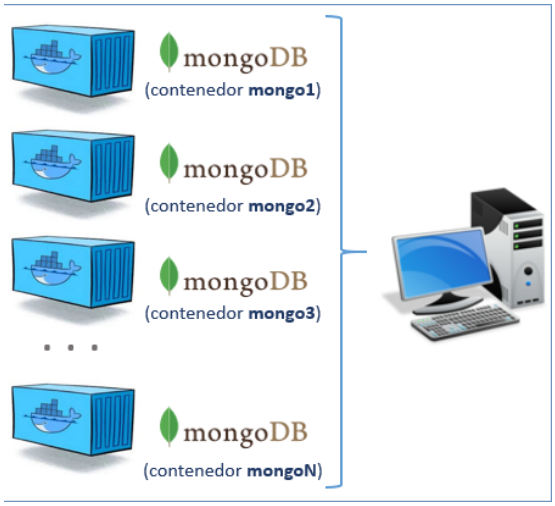


Figura 4.Contenedores de Docker

5. Materiales y metodología

El desarrollo del proyecto se realizó mediante un marco metodológico híbrido, el cual combina la estructura secuencial del modelo Cascada con prácticas de metodologías ágiles y principios DevOps, permitiendo así mantener un orden claro en las fases del proyecto, al mismo tiempo que se incorporó tolerancia, retroalimentación continua y automatización de procesos los cuales son de vital importancia en gestión de instancias de Odoo CE.

El modelo Cascada proporcionó la estructura principal, tomando como guía la secuencia de etapas del proyecto y asegurando que cada una genere entregables concretos antes de avanzar a la siguiente etapa manteniendo así la característica más importante de la metodología mencionada. Esto permitió garantizar el cumplimiento de los objetivos planteados.

De manera complementaria, se aplicaron otras prácticas ágiles, que facilitaron la adaptación progresiva de funcionalidades conforme surgieron nuevas necesidades durante el desarrollo. Entre las técnicas utilizadas se incluyen:

- Iteraciones cortas para desarrollar funcionalidades de manera incremental
- Reuniones periódicas con el tutor
- Entrega incremental de módulos

Asimismo, se incorporaron principios DevOps para mejorar la calidad operativa y la automatización del ciclo de vida de las instancias. Entre las prácticas aplicadas destacan:

- Automatización de despliegue utilizando Docker
- Integración continua manual mediante pruebas frecuentes en cada módulo backend y frontend
- Automatización operativa para la creación de instancias, ejecución de backups, restauraciones y la funcionalidad de merge, utilizando Docker y SpringBoot
- Monitoreo mediante logs

De esta manera, aunque la metodología es Cascada, el proyecto se beneficia de la flexibilidad del enfoque de las metodologías ágiles y la eficiencia operacional del

enfoque DevOps, dando lugar a un proceso de desarrollo más adaptable y solida con etapas de retroalimentación y en un proceso lineal.

5.1 Fases del desarrollo (Metodología Cascada)

En el desarrollo del proyecto se organizó con la base del proceso lineal y de etapas de la metodología en cascada, sin embargo, dentro de cada etapa se tomaron en cuenta ciertas características de los principios de DevOps para mejorar el proceso, manteniendo la coherencia del enfoque hibrido adoptado.

5.1.1 Planeación y Análisis de Requisitos

En esta etapa se establecieron las bases del proyecto mediante el levantamiento de requerimientos funcionales y no funcionales (Anexo). Se seleccionaron elementos clave como la creación de instancias por categoría, gestión de backups y restauraciones, visualización de logs e interacción general con contenedores de Odoo CE. También se identificaron limitaciones, tales como el uso de Docker en Windows, el consumo de recursos y la administración de contenedores.

Entregable de la etapa: Documento de requisitos funcionales y no funcionales (Anexo).

Lugar	Fecha	Duración	Temas tratados	Asistentes
Universidad Politécnica Salesiana	14/01/2025	45 minutos	-Planificación del desarrollo (metodología de implementación) -Especificación de requerimientos y tecnologías implementadas (Manejo de Docker, utilización de angular y Spring boot).	Ing. Cristian Timbi, Ing, Paul Guzhñay, Ing Sebastian Bedoya.

Universidad Politécnica Salesiana	24/03/2025	45 minutos	-Revisión ajustes en el desarrollo y manejo de la base de datos. -Para autenticación de usuarios con el uso de Github.	Ing. Cristian Timbi, Ing, Paul Guzhñay, Ing Sebastian Bedoya.
Universidad Politécnica Salesiana	07/07/2025	45 minutos	-Correcciones en desarrollos y manejo de configuraciones del sistema. (manejo de git) -Correcciones en métodos relacionados al entorno (Merge y Fork)	Ing. Cristian Timbi, Ing, Paul Guzhñay, Ing Sebastian Bedoya.
Virtual	21/08/2025	1 hora	-Correcciones, depuración y presentación final del proyecto técnico cumpliendo los requerimientos funcionales solicitados	Ing. Cristian Timbi, Ing, Paul Guzhñay, Ing Sebastian Bedoya.

Tabla 1. Reuniones realizadas

5.1.2 Diseño de la Solución

Se definió la arquitectura del sistema. En el Backend, se diseñó una API RESTful con Spring Boot, estableciendo la lógica de los servicios, controladores, repositorios y modelos de datos. En el Frontend, se desarrolló la aplicación utilizando Angular y Angular Material la cual nos sirvió para la personalización de interfaces más amigables con el usuario final.

Entregable de la etapa: Diagramas con la arquitectura del Backend, y frontend (Anexo 1 y 2).

5.1.3 Implementación del BackEnd

Se implementaron los servicios los cuales van a ser los encargados de manejar los contenedores Docker desde Java:

- Creación dinámica de contenedores Odoo y PostgreSQL.
- Asignación automática de puertos.
- Automatización de backups, restauraciones y eliminación de instancias.

Entregable: Código fuente del Backend con endpoints operativos.

5.1.4 Implementación del Frontend

Desarrollo de módulos y componentes Angular para consumir la API del Backend:

- Dashboard de proyectos.
- Sidebar dinámico.
- Pestañas en cada proyecto y rama de repositorio para logs, backups, restauraciones y comandos.
- Shell para consultas a la base de datos.

Entregable: versión integrada con funcionalidades avanzadas amigables al usuario

5.1.5 integración y Funcionalidades avanzadas

Implementación de funciones avanzadas como:

- Creación de instancias una rama de pruebas con un ambiente neutralizado el cual sirve para la aprobación de funcionalidades por parte del usuario final.

- Instancias de desarrollo con datos demo los cuales pueden funcionar como demostración para que el cliente decida si realiza personalizaciones.
- La unión entre entornos.
- Manejo de errores de restauración.
- Eliminación de volúmenes en uso.

Entregable: Sistema integrado con funciones avanzadas.

5.1.6 Pruebas Finales

Se realizaron un testeo de todas las funcionalidades (restauraciones, merges y eliminación de instancias). Ademas, se verifico la persistencia de los datos de las instancias, la correcta asociación entre proyectos e instancias y también el correcto funcionamiento de los backups.

En la fase final del desarrollo del proyecto se ha realizado un proceso de validación de los requerimientos que se han planteado con la ayuda de pruebas funcionales. Todas estas pruebas se hicieron con el objetivo de evaluar el desempeño de cada uno de los componentes y confirmar que las funcionalidades implementadas respondan de forma correcta a los objetivos planteados.

Durante esta etapa se revisó cada requisito que se planteó previamente, ejecutando pruebas diseñadas para simular escenarios e interacciones en las que entrar en un proceso crítico del sistema para verificar su correcta reacción.

Este proceso permitió verificar que el sistema conste de estabilidad, precisión y coherencia frente a diferentes condiciones de uso.

Finalmente, se muestran los resultados obtenidos tras la ejecución de las pruebas funcionales las cuales se realizaron de forma metódica para asegurar el cumplimiento de los requerimientos y comprobar la operatividad general de la solución propuesta.

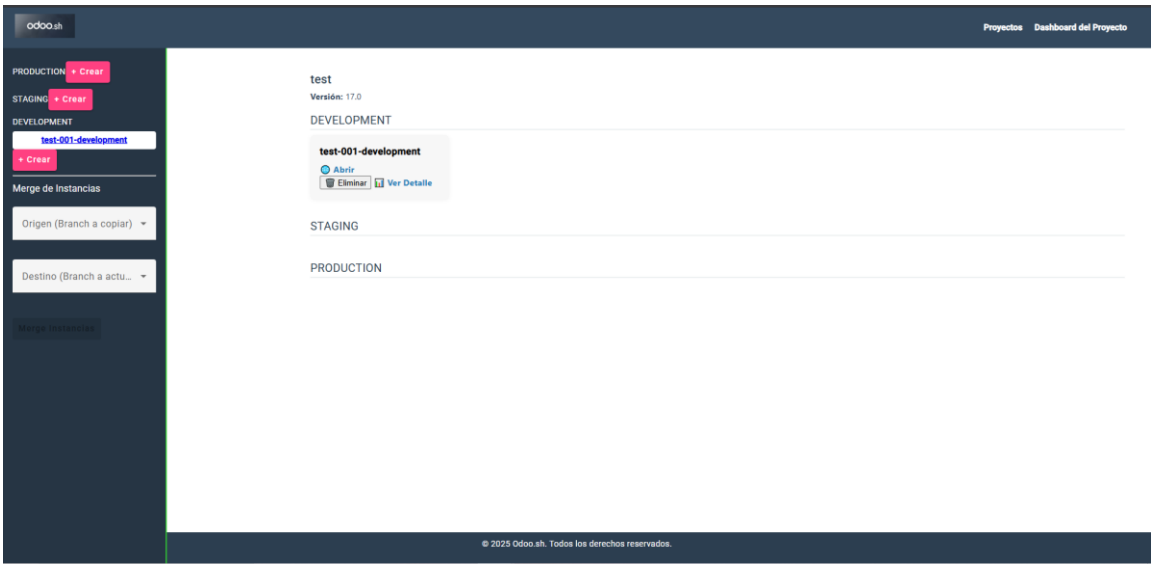
<table>
<tr><td rowspan="3">Prueba de funcionalidad 1</td><td>Código</td><td>PFSW-1</td><td>Resultado</td></tr>
<tr><td>Responsable</td><td>Paul Guzhñay</td><td rowspan="2">Aprobado</td></tr>
<tr><td>Fecha</td><td></td></tr>
<tr><td colspan="4">Requerimiento:
Permitir la creación de entornos dinámicos de Odoo para cada cliente registrado diferenciando entre clientes los proyectos creados</td></tr>
<tr><td colspan="4">Resultado Esperado:
El sistema por medio de una interfaz gráfica permite crear nuevas instancias de Odoo al ingresar en ese formulario los parámetros necesitados nuestro un mensaje de éxito y redirige a la nueva instancia creada</td></tr>
<tr><td colspan="4">Resultado Obtenido:
El resultado obtenido nos demuestra que se creó de una manera exitosa la instancia.
 
</td></tr>
</table>

Tabla 2. Prueba Funcional 1: Crear una Instancia nueva

Tabla 3. Prueba Funcional 2: Consulta de datos en la Base de Datos

Prueba de funcionalidad 2	Código	PFSW-2	Resultado
	Responsable	Paul Guzhñay	Aprobado
	Fecha		

Requerimiento:

Brindar soporte a los datos o modificaciones de registro en caso de errores mediante un acceso seguro a los datos del sistema

Resultado Esperado:

El sistema con el uso de una interfaz permitirá realizar consultas sobre los datos de una instancia especifica, utilizando consultas o instrucciones SQL preexistentes en la interfaz .

Resultado Obtenido:

El resultado obtenido nos muestra una pantalla con el resultado de la consultada en relación con la misma entidad y rama de donde se realiza la consulta

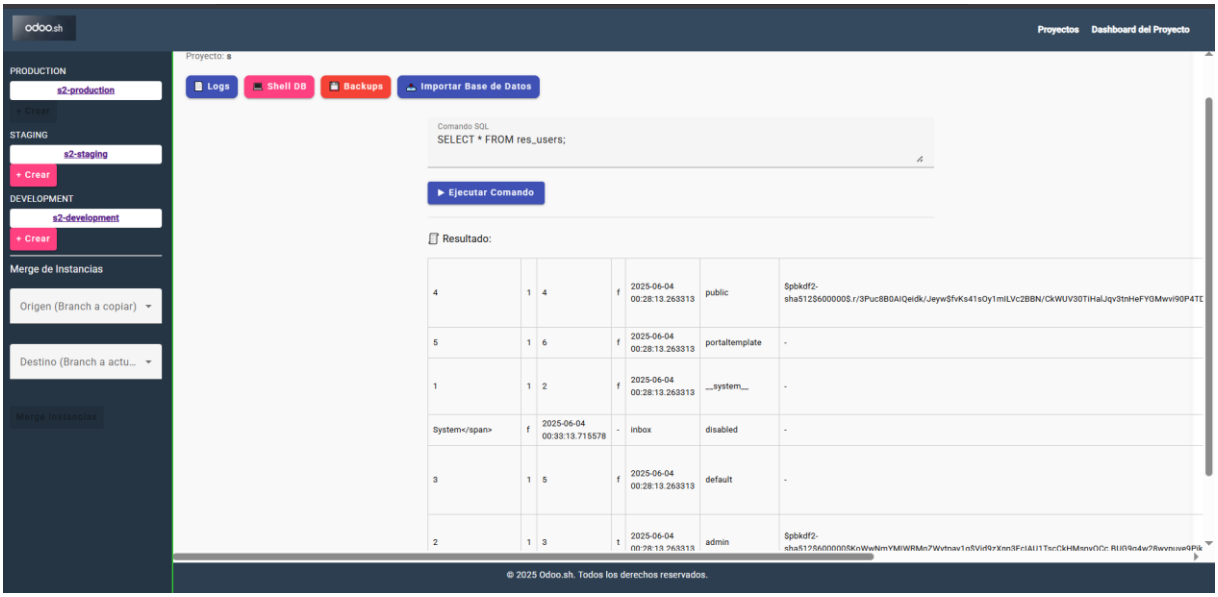
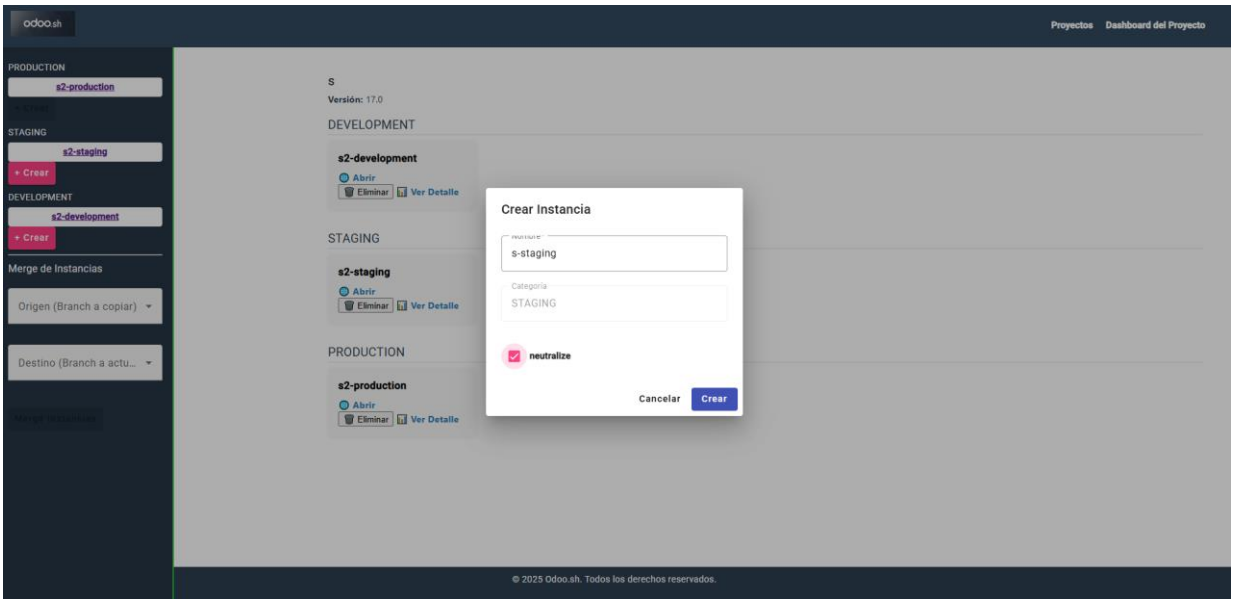
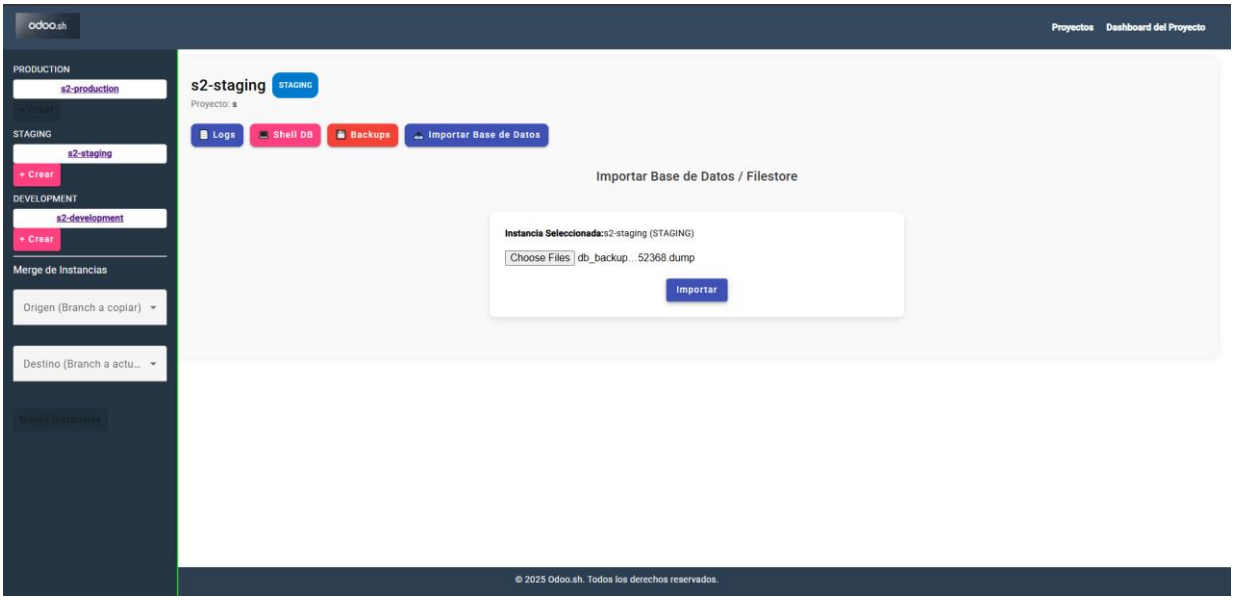


Tabla 4. Prueba Funcional 3: Creación de instancias con neutralize de datos

<table>
<tr><td rowspan="3">Prueba de funcionalidad 3</td><td>Código</td><td>PFSW-3</td><td>Resultado</td></tr>
<tr><td>Responsable</td><td>Paul Guzhñay</td><td rowspan="2">Aprobado</td></tr>
<tr><td>Fecha</td><td></td></tr>
<tr><td colspan="4">Requerimiento:
Garantizar el uso de un ambiente de pruebas con datos reales sin la necesidad de estar en un ambiente de producción</td></tr>
<tr><td colspan="4">Resultado Esperado:
El sistema proveerá de un ambiente de pruebas con los datos de producción, sin embargo, estos datos serán neutralizados al igual que ciertas funcionalidades como él envió de correos notándose con un mensaje de (base de datos neutralizada en el ambiente)</td></tr>
<tr><td colspan="4">Resultado Obtenido:
El resultado obtenido nos permite visualizar una pantalla con para poder realizar la creación de la entidad de Odoo con la configuración activada de “neutralize”
 
</td></tr>
</table>

<table>
<tr><td rowspan="3">Prueba de funcionalidad 4</td><td>Código</td><td>PFSW-4</td><td>Resultado</td></tr>
<tr><td>Responsable</td><td>Paul Guzhñay</td><td rowspan="2">Aprobado</td></tr>
<tr><td>Fecha</td><td></td></tr>
<tr><td colspan="4">Requerimiento:
Realizar Importaciones de bases de datos facilitando la migración de sistemas o la carga de una base en una entidad nueva</td></tr>
<tr><td colspan="4">Resultado Esperado:
El sistema deberá permitir la carga de documentos de tipo dump o un archivo comprimido de tipo ZIP el cual deberá contener el archivo de la base de datos y el archivo filestore</td></tr>
<tr><td colspan="4">Resultado Obtenido:
El resultado obtenido es poder importar una nueva base de datos como se observa en la primera imagen, luego recibimos un mensaje de confirmación como se observa en la segunda imagen
 
</td></tr>
</table>

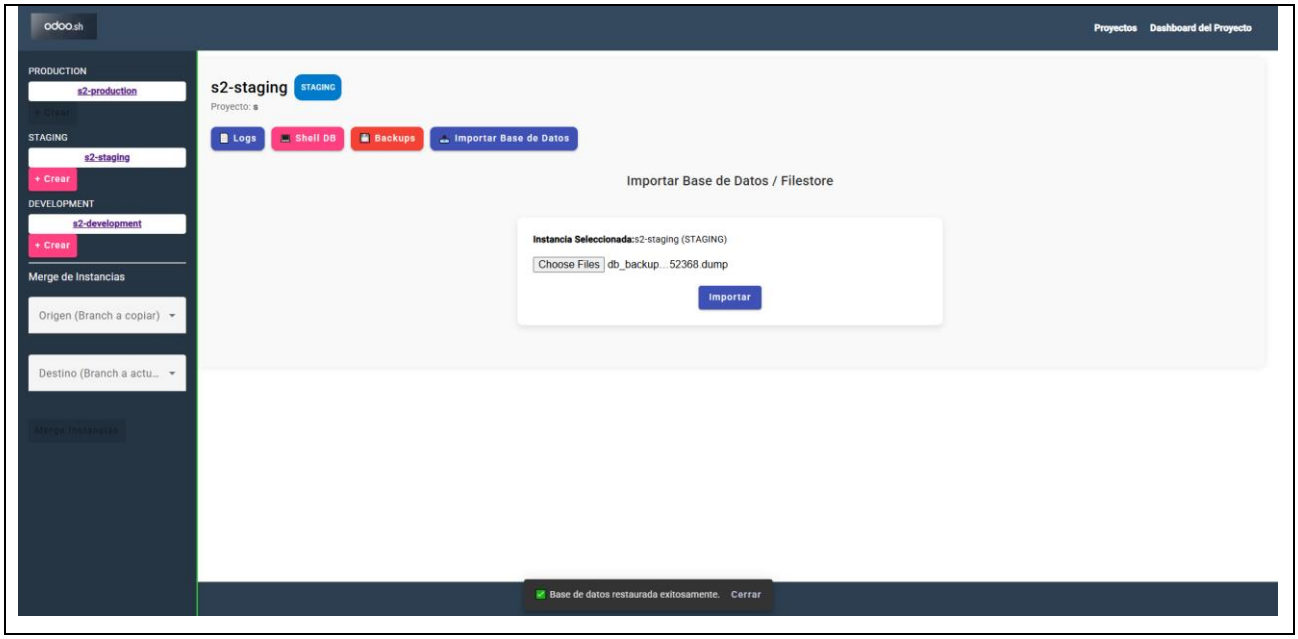


Tabla 5. Prueba Funcional 4: Importar base de datos

<table>
<tr><td rowspan="3">Prueba de funcionalidad 5</td><td>Código</td><td>PFSW-5</td><td>Resultado</td></tr>
<tr><td>Responsable</td><td>Paul Guzhñay</td><td rowspan="2">Aprobado</td></tr>
<tr><td>Fecha</td><td></td></tr>
<tr><td colspan="4">Requerimiento:
Permitir cambios entre entornos tres tipos de ramas las cuales son producción, pruebas y desarrollo.</td></tr>
<tr><td colspan="4">Resultado Esperado:
Con la ayuda de una interfaz gráfica, el software permitirá la fusión de ramas y reflejar un mensaje de éxito</td></tr>
<tr><td colspan="4">Resultado Obtenido:
Como resultado de la prueba funcional, se realizó el merge de dos instancias (s2-development; s2-staging) como se observa en la primera imagen. En la segunda imagen obtenemos la notificación en donde se visualiza que se realizó de forma correcta el merge.
 
</td></tr>
</table>

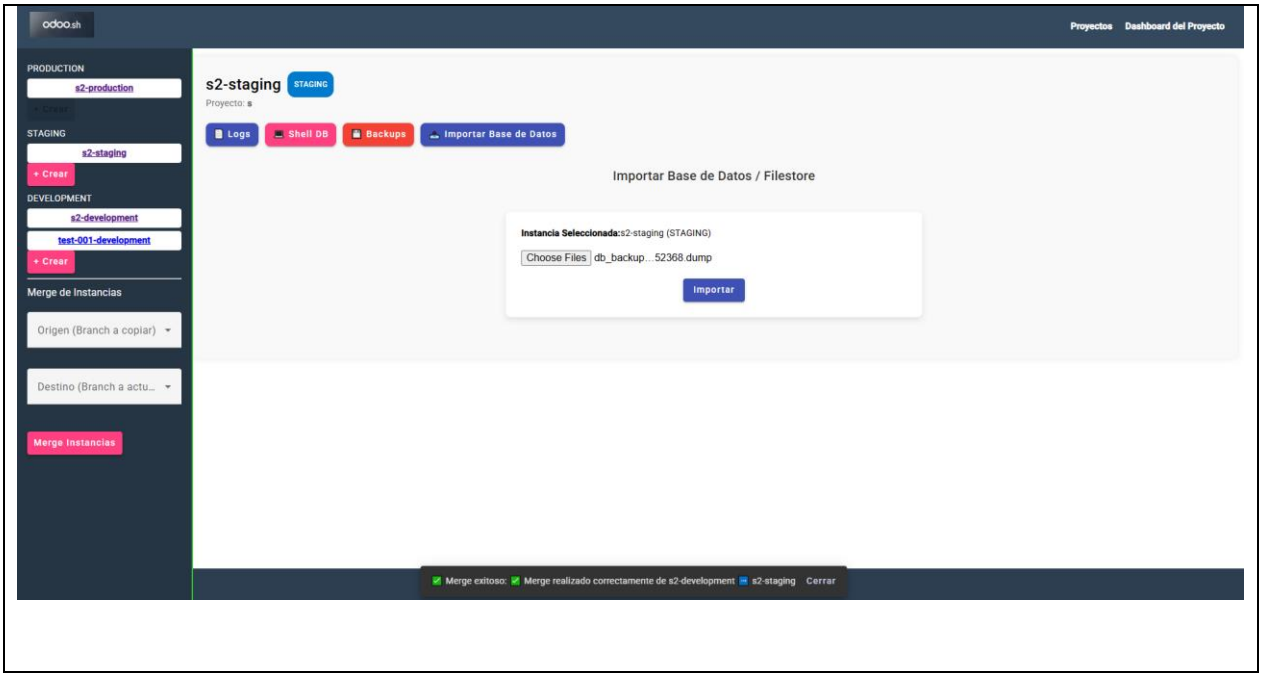


Tabla 6. Prueba Funcional 5: Merge entre 2 instancias

6. Resultados y discusión

En el desarrollo de la plataforma se cumplieron satisfactoriamente los objetivos planteados, alcanzando como resultado una solución funcional, modular y eficiente para la automatización del ciclo de vida de las instancias de Odoo Community. El sistema combina mediante el uso de contenedores Docker con una interfaz web intuitiva, logrando materializar el objetivo general de construir una plataforma SaaS que permita gestionar de forma integral y automatizada el despliegue, configuración y mantenimiento de instancias personalizadas de Odoo CE, facilitando su administración y escalabilidad para los usuarios finales.

En cuanto con el diseño de la arquitectura, se estableció una estructura basada en contenedores Docker que garantizo el aislamiento de cada instancia, junto con volúmenes personalizados y la asignación dinámica de puertos y nombres únicos. Este diseño responde al objetivo de crear una base escalable y repetible que soporte

la ejecución simultanea de múltiples entornos sin generar conflicto, optimizando el despliegue y mantenimiento de las instancias.

Respecto a las funcionalidades para la gestión de usuarios y copias de seguridad, se implementó un sistema de respaldo y restauración capaz de exportar de manera independiente la base de datos y los archivos filestore de Odoo.

Los respaldos se organizan automáticamente por fecha de creacion, lo que permite mantener un control ordenado y facilita tanto la restauración como la descarga de archivos. Con esta implementación, se cumple el objetivo de dotar a la plataforma de mecanismos confiables de seguridad de datos y continuidad operativa.

Entre los resultados más destacados se encuentra la funcionalidad de merge, que permite migrar datos y archivos entre instancias. Gracias a esta funcionalidad se completó el objetivo de automatizar la clonación de entornos, permitiendo que una instancia de producción pueda replicarse en un entorno de staging para realizar pruebas, manteniendo la integridad del sistema y garantizando la sincronización de entornos.

En lo que respecta a la interfaz del usuario, se creó un dashboard que muestra los proyectos creados y cada una de las funcionalidades que hay dentro de cada uno de ellos. Las instancias de cada proyecto se muestran en una clasificación de categorías, las cuales son: Production, Staging y Development. Otra funcionalidad es que cada componente despliega de forma automática funcionalidades extras para cada instancia, como lo son: visualización de logs, ejecución de comandos, realizar respaldos e importar bases de datos. Esto nos ayudo a cumplir el objetivo de brindar una interfaz web intuitiva que simplifica la gestión de instancias.

Durante el desarrollo del proyecto se presentaron varios obstáculos, los cuales se fueron superando de poco a poco. Alguno de los mayores obstáculos o desafíos que se presento fueron los siguientes: la gestión de los permisos del filestore en los volúmenes de docker, la sincronización de los datos en la restauración y la ejecución de los comandos en los entornos. Cada uno de los obstáculos, se llegaron a superar mediante prueba y error, ajustar configuraciones en los contenedores y

el analisis de los errores que se llegaron a mostrar en la consola. Cada obstáculo que superamos nos ayudo a desarrollar una plataforma más óptima y eficiente.

En cuanto a los resultados que obtuvimos, estos permiten concluir que la plataforma desarrollada en el proyecto se presenta como una solución alterna para los departamentos de TI. Ademas, se logró automatizar tareas que son de alta exigencia, siendo más específicos, todo lo que respecta a Odoo.

Algo adicional que presenta la plataforma es el uso de Docker, el cual nos ayuda a tener entornos aislados, lo cual nos sirve para poder simplificar ciertas tareas, como lo es la gestión de diversos entornos. Esto también nos ayuda ya que permite replicar o migrar entornos de una forma más rápida y efectiva, sin importar cual sea el objetivo, estos pueden llegar a ser: mantenimientos, desarrollos o producción.

Asimismo, las distintas funcionalidades que presenta la plataforma como lo son: backup, restauración, merge y la importación de base de datos, brinda una mayor robustez y también permite tener una continuidad en las operaciones. Gracias a las automatizaciones se logra reducir lo que son errores humanos.

Por otra parte, la interfaz desarrollada en Angular, resulta a ser amigable con el usuario, debido a que se puede adaptar a cualquier nivel de usuario. Para los distintos miembros de los departamentos de TI, es una gran ventaja ya que se puede acceder y ejecutar comandos en el servidor.

Por otra parte, el proyecto llega a demostrar que si es posible hacer una plataforma de gestión automatizada de instancias de ERP. Para los departamentos de TI, brinda una gran ayuda ya que se puede mantener el control de la infraestructura, optimizar el de los recursos, evitar el uso de servidores externos.

Ademas, de los resultados alcanzados, de la misma forma también se llego a identificar ciertas limitaciones que pueden mejorarse en el futuro. La principal limitación fue la compatibilidad entre versiones de Docker, Odoo y PostgreSQL, lo que hace que se tengan que ajustar ciertas configuraciones en el futuro.

De la misma forma, al comparar esta plataforma con herramientas en el mercado, se llega a observar que se ofrece funcionalidades similares a soluciones comerciales orientadas al despliegue automatizado de Odoo, pero con la gran ventaja de basarnos en tecnologías completas de codigo abierto. Esto nos permite reducir costos y permite que la infraestructura sea administrada por equipos de TI, lo cual evita dependencias externas.

6.1. Diseño

En esta sección, se presenta una explicación detallada de las principales funcionalidades realizadas en el proyecto, las cuales fueron diseñadas y adaptadas para operar de manera óptima en entornos Docker. Cada una de estas funcionalidades desempeña un rol fundamental en el cumplimiento de los objetos establecidos, contribuyendo a la automatización y eficiencia del sistema.
A continuación, se describen las características más relevantes que fueron implementadas en la plataforma:

- Crear instancias
- Merge entre las instancias
- Generar backups
- Realizar restauración de los backups
- Arquitectura General

6.1.1. Arquitectura General

La arquitectura propuesta para el proyecto se enfoca en un sistema modular y desacoplado, orientado en asegurar la mantenibilidad y automatización.
Para ello el proyecto se compone de tres capas principales: Frontend, Backend y Docker este constando con imágenes de Odoo y PostgreSQL. Estas capas se comunican de manera coordinada, lo que permite gestionar el ciclo de vida de

las instancias de Odoo desde una interfaz web amigable y fácil de usar, tal como lo podemos observar en la figura 11.

Con esta arquitectura usamos el principio de separación de responsabilidades:

- Frontend se encarga de la interacción con el usuario
- Backend administra la lógica del negocio y las operaciones sobre contenedores
- Docker encapsula los entornos aislados de Odoo y Base de Datos

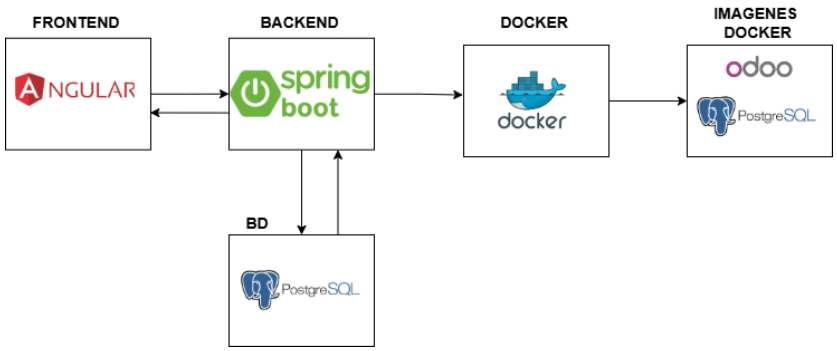


Figura 5. Arquitectura del proyecto

A continuación, se va a describir el funcionamiento de cada una de las capas establecidas en el proyecto:

- **Frontend:** Esta es la capa de presentación fue desarrollada con la tecnología de Angular, el mismo que nos permite construir interfaces web reactivas. Aquí el usuario puede realizar lo siguiente:
  - Crear instancias Odoo
  - Ver, descargar o restaurar backups
  - Eliminar instancias
  - Realizar operaciones de merge entre instancias
  - Visualizar logs

Angular se comunica de forma directa con el Backend mediante los servicios REST, los cuales envía los parámetros necesarios y se muestra los resultados de cada método en tiempo real.

- **Backend:** El Backend está construido con la tecnología de Spring Boot y actúa como el núcleo de la lógica del sistema. Las funciones desarrolladas son las siguientes:
    - Orquestar contenedores de Docker para poder crear, eliminar, respaldar y restaurar las instancias.
    - Gestionar la conexión con la base de datos
    - Generar endpoints seguros y eficientes para el frontend
    - Ejecutar comandos del sistema a través de la terminal para poder manejar de forma dinámica las operaciones de Docker
    - Administrar condiciones específicas de cada una de las instancias
- **Docker:** Esta tecnología cumple el rol de ser el virtualizador ligero, el cual nos permite ejecutar múltiples contenedores de Odoo y PostgreSQL. Esto nos ayuda a simplificar la gestión del entorno y garantizar que cada una de las instancias de Odoo funcione de manera correcta

6.2. Codificación

6.2.1 Dashboard de proyectos

Al iniciar la creación de un proyecto desde el frontend, se valida primero la sesion de GitHUb, ya que la plataforma crea de forma automática un repositorio asociado como se observa en la Figura 11.

```
openCreateDialog(): void {
  const dialogRef = this.dialog.open(ProjectCreateDialogComponent, {
    width: '400px'
  });

  dialogRef.afterClosed().subscribe((result: Project | undefined) => {
    if (result) {
      this.projectService.create(result).subscribe({
        next: (created) => {
          this.snackBar.open('Proyecto creado', 'Cerrar', {
            duration: 3000,
            panelClass: 'snackbar-success'
          });
          this.projects.push(created);
        },
        error: (err) => {
          if (err.status === 401) {
            this.snackBar.open('Token expirado. Redirigiendo a GitHub...', 'Cerrar', {
              duration: 3000,
              panelClass: 'snackbar-error'
            });
            this.projectService.handleUnauthorized();
          } else if (err.status === 409) {
            this.snackBar.open(' El repositorio ya existe en GitHub', 'Cerrar', {
              duration: 4000,
              panelClass: 'snackbar-error'
            });
          } else {
            this.snackBar.open('Error al crear proyecto', 'Cerrar', {
              duration: 3000,
              panelClass: 'snackbar-error'
            });
          }
        }
      });
    }
```

Figura 6. Codigo Frontend para crear los proyectos

```java
public ResponseEntity<?> create(@RequestBody Project project, Authentication authentication) {
    String username = authentication.getName();
    Optional<User> userOpt = userRepo.findByUsername(username);

    if (userOpt.isEmpty()) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Usuario no válido");
    }

    User user = userOpt.get();
    project.setUser(user);

    try {
        githubService.createRepository(user.getGithubToken(), project.getName());
    } catch (AuthenticationCredentialsNotFoundException ex) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body("Token de GitHub expirado o inválido. Vuelva a
    } catch (Exception e) {
        System.err.println(" Error general al crear repositorio: " + e.getMessage());
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Error creando repositorio en GitHub:
    }

    Project savedProject = projectRepo.save(project);
    return ResponseEntity.ok(savedProject);
}
```

Figura 7. Codigo Backend para crear proyectos

Si el token ha expirado o no existe, el BackEnd responde con 401 como se observa en la Figura 12 y el cliente es redirigido al flujo de autentificación de GitHub. En caso de éxito el BackEnd manda como respuesta un estado 200, y en el Frontend se traduce como un mensaje de éxito al usuario.

6.2.2 Crear instancias

En esta sección se va a mostrar el proceso para crear una instancia en las distintas categorías que se dispone.

```typescript
submit(): void {

  if (this.isProcessing) return;

  if (this.form.invalid) return;

  this.isProcessing = true;

  const payload = {
    ...this.form.value,
    projectId: this.data.projectId
  };

  this.odoo.createOdooInstance(payload).subscribe({
    next: (inst) => {
      this.instanceCreated.emit();
      this.dialogRef.close(inst);
    },
    error: (err) => {
      this.isProcessing = false;
      this.form.setErrors({ server: err?.error?.message || 'Error del servidor' });
    }
  });
```

Figura 8. Codigo Frontend para crear instancia

```java
public Map<String, String> createInstance(@RequestBody InstanceRequest request) {
    Map<String, String> response = new HashMap<>();

    try {
        String url = dockerService.createOdooInstance(
                request.getName(),
                request.getCategory(),
                request.getProjectId(),
                request.isNeutralize(),
                request.getCodeSourceCategory(),
                request.isCopyDataFromProduction()
        );
        if (url != null && url.startsWith("http")) {
            response.put("message", "Instancia de Odoo creada con éxito en " + request.getCategory());
            response.put("url", url);

            String branchName = request.getCategory().toLowerCase() + "-" + request.getName().replaceAll( regex: "
            response.put("branch", branchName);
        } else {
            response.put("message", "❌ Error: No se pudo crear la instancia.");
        }
    } catch (Exception e) {
        e.printStackTrace();
        response.put("message", "❌ Error inesperado: " + e.getMessage());
    }
    return response;
}
```

Figura 9. Codigo Backend para crear instancia

En la creación de las instancias se inicia desde un dialogo en el frontend, que valida el formulario y construye un payload con los siguientes parámetros:

- Nombre
- Categoría
- ProjectId
- Neutralize
- CodeSourceCategory
- CopyDataFromProduction

Donde se invoca el endpoint como se puede observar en la Figura 13.

En el Backend, el controlador recibe la solicitud y delega en el método CreateOdooInstance, en donde se orquesta la provisión de contenedores, la configuración por categoría y cuando corresponde, la neutralización y copia desde producción, al finalizar retorna el mensaje, la URL de acceso como se observa en la Figura 14

6.2.3 Merge entre distintas instancias

Esta es una de las funcionalidades más complejas que se llegó a desarrollar y por eso lo importante que llega a ser para el sistema. El merge de las instancias nos permite transferir los cambios realizados en una instancia fuente hacia una instancia destino.

```
mergeBranches(): void {
  if (!this.selectedSource || !this.selectedTarget) {
    this.snackBar.open('Selecciona instancia origen y destino para hacer el merge.', 'Cerrar', { duration: 4000 });
    return;
  }
  if (this.selectedSource.name === this.selectedTarget.name) {
    this.snackBar.open('No puedes hacer merge de la misma instancia sobre sí misma.', 'Cerrar', { duration: 4000 });
    return;
  }
  const confirmacion = confirm(`¿Seguro que quieres hacer merge de ${this.selectedSource.name} ➡ ${this.selectedTarget.name}?`);
  if (!confirmacion) return;
  this.isProcessing = true;
  console.log(" Enviando payload de merge:", {
    source: this.selectedSource.name,
    target: this.selectedTarget.name,
    projectId: this.projectId
  });
  console.log(" Categorías:", this.selectedSource.category, this.selectedTarget.category);
  this.odooService.mergeInstances(
    this.selectedSource.name,
    this.selectedTarget.name,
    this.projectId,
    this.selectedSource.category,
    this.selectedTarget.category,
  ).subscribe({
    next: (response) => {
      console.log('Respuesta del merge:', response);
      if (response.includes('✅')) {
        this.snackBar.open(` Merge exitoso: ${response}`, 'Cerrar', { duration: 6000, panelClass: ['success-snackbar'] });
      } else if (response.includes('⚠')) {
        this.snackBar.open(` Merge con advertencias: ${response}`, 'Cerrar', { duration: 6000, panelClass: ['warning-snackbar'] });
      } else if (response.includes('❌')) {
        this.snackBar.open(` Error en el merge: ${response}`, 'Cerrar', { duration: 6000, panelClass: ['error-snackbar'] });
      } else {
        this.snackBar.open(` Resultado del merge: ${response}`, 'Cerrar', { duration: 6000 });
      }
      this.loadInstances();
    },
    error: (error) => {
      console.error('Error al hacer merge:', error);
      if (error.status === 401) {
        this.snackBar.open(' Token expirado. Redirigiendo a GitHub...', 'Cerrar', {
          duration: 6000,
          panelClass: ['error-snackbar']
        });
        this.odooService.handleUnauthorized();
      } else {
        this.snackBar.open('Error grave al hacer merge. Revisa los logs del backend.', 'Cerrar', {
          duration: 6000,
          panelClass: ['error-snackbar']
        });
      }
    },
    complete: () => {
```

Figura 10. Codigo desde el Frontend para el merge

```java
@PostMapping("/merge") no usages  Paulguzhnay *
public ResponseEntity<String> mergeInstances(@RequestBody MergeRequest request) {
    try {
        String result = dockerService.mergeOdooInstances(
                request.getSource(),
                request.getTarget(),
                request.getProjectId()
        );

        System.out.println(" Resultado del merge: " + result);

        return result.startsWith("✅")
                ? ResponseEntity.ok(result)
                : ResponseEntity.status(500).body(result);
    } catch (Exception e) {
        e.printStackTrace();
        return ResponseEntity.status(500).body(" Error inesperado: " + e.getMessage());
    }
}
```

Figura 11. Codigo desde el Backend para el merge

El proceso de merge se inicia desde el frontend seleccionado la instancia origen y destino, se valida que ambas existan y que no sean las mismas y se procede a solicitar la confirmación del usuario.

Luego el cliente invoca el servicio de merge enviando el nombre de instancia, categorías y ProjectId, gestionando estados de carga y la visualización del resultado como se observa en el codigo de la Figura 15

En el Backend, se delega la operación al método mergeOdooInstance, el cual es responsable de orquestar la integración. El controlador unifica la respuesta y devuelve 200 cuando el método se ha realizado con éxito y codigo 500 si existió algun problema.

6.2.4 Dashboard de una instancia

En este apartado, vamos a tener opciones que son de forma general, es decir, vamos a poder visualizar los logs, ejecutar comandos de la base de datos, generar y restaurar backups y poder importar base de datos.

```
<div class="instance-detail">
  <div class="header">
    <h2>{{ instanceName }} <span class="badge">{{ category }}</span></h2>
    <p class="project-name">Proyecto: <strong>{{ projectName }}</strong></p>
  </div>

  <div class="actions">
    <a mat-raised-button [ngClass]="{ active: currentTab === 'logs' }" color="primary"
      [routerLink]="['/projects', projectName, 'instance', instanceName, 'logs']" [queryParams]="{ category }">
      Logs
    </a>
    <a mat-raised-button [ngClass]="{ active: currentTab === 'shellexec' }" color="accent"
      [routerLink]="['/projects', projectName, 'instance', instanceName, 'shellexec']" [queryParams]="{ category }">
      Shell de Comandos
    </a>

    <a mat-raised-button [ngClass]="{ active: currentTab === 'shell-db' }" color="accent"
      [routerLink]="['/projects', projectName, 'instance', instanceName, 'shell-db']" [queryParams]="{ category }">
      Shell DB
    </a>

    <a mat-raised-button [ngClass]="{ active: currentTab === 'backups' }" color="warn"
      [routerLink]="['/projects', projectName, 'instance', instanceName, 'backups']" [queryParams]="{ category }">
      Backups
    </a>

    <a mat-raised-button [ngClass]="{ active: currentTab === 'import-db' }" color="primary"
      [routerLink]="['/projects', projectName, 'instance', instanceName, 'import-db']" [queryParams]="{ category }">
      Importar Base de Datos
    </a>
  </div>


  <router-outlet></router-outlet>
</div>
```

Figura 12. Codigo para el detalle de las instancias

Para poder visualizar los distintos detalles de las instancias se expone una navegación por secciones (Logs, Shell de Comandos, Shell DB, Backups, Importar Base de datos) mediante enlaces enrutados de Angular como se observa en la Figura 17.

6.2.5 Generar backups

En esta opción del meno vamos a poder generar backups de la instancia que hemos ingresado.

```
createBackupForInstance(instance: OdooInstance): void {
  this.isCreatingBackup = true;

  const payload = {
    name: instance.name,
    category: instance.category,
    projectId: this.projectId
  };

  console.log(" Payload enviado para backup:", payload);

  this.backupService.createBackupForInstance(payload).subscribe({
    next: (response) => {
      console.log('Backup creado:', response);
      this.showNotification(response.message, 'success');
      this.loadBackups();
    },
    error: (error) => {
      console.log('Error al crear backup:', error);
      const msg = error.error?.message || 'Error al crear backup';
      this.showNotification(msg, 'error');
    },
    complete: () => {
      this.isCreatingBackup = false;
    }
  });
}
```

Figura 14. Codigo Frontend para generar backup

```
public ResponseEntity<Map<String, String>> backupInstance(@RequestBody BackupRequest request, Authentication authe
    Map<String, String> responseMap = new HashMap<>();
    try {
        String username = authentication.getName();
        String result = dockerService.createBackup(username, request.getProjectId(), request.getName(), request.ge

        responseMap.put("message", result);
        return result.startsWith("✅")
                ? ResponseEntity.ok(responseMap)
                : ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(responseMap);

    } catch (Exception e) {
        e.printStackTrace();
        responseMap.put("message", "Error inesperado: " + e.getMessage());
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(responseMap);
    }
}
```

Figura 15. Codigo Backend para generar backup

La creación de backups se gestiona desde el frontend mediante un método que construye un payload con el nombre de la instancia, categoría y projectId, lo envía al servicio de backups. Durante esta operación, se controla el estado de ejecución y se muestran notificaciones en base al resultado todo esto se refleja en la Figura 19

Mientras en el Backend, el endpoint recibe la solicitud, obtiene el usuario autentificado y delega al método createBackup la creación del respaldo. Si se tuvo éxito se retorna un 200, caso contrario 500. Las excepciones se capturan y devuelven como 400 con un mensaje estándar como se observa en la Figura 20

6.2.6 Realizar la restauración de backups

Este es un proceso que complementa a la copia de seguridad o backup. Esta funcionalidad nos permite recuperar el estado completo de una instancia a partir de una copia generada anteriormente.

```
restoreBackupGroup(group: any): void {
  this.isRestoring = true;
  this.currentStep = 'Iniciando...';
  this.restoringGroupId = group.timestamp;
  const dbFile = group.backups.find((b: any) => b.name.includes('db_backup'))?.name;
  const odooFile = group.backups.find((b: any) => b.name.includes('odoo_data'))?.name;
  if (!dbFile || !odooFile) {
    this.showNotification('No se encontraron archivos válidos.', 'error');
    this.isRestoring = false;
    return;
  }
  const match = dbFile.match(/^db_backup_([^_]+)_([A-Z]+)_([^_]+)_/);
  if (!match || match.length < 4) {
    this.showNotification('❌ No se pudo determinar el proyecto, categoría o instancia.', 'error');
    this.isRestoring = false;
    return;
  }
  const [, projectFromFile, categoryFromFile, instanceFromFile] = match;
  const payload = {
    name: instanceFromFile,
    category: categoryFromFile,
    dbBackupFileName: dbFile,
    odooBackupFileName: odooFile,
    projectId: this.projectId
  };
  console.log('Payload para restauración:', payload);
  this.backupService.restoreSpecific(payload).subscribe({
    next: (response: string) => {
      this.showNotification(response, 'success');
      this.isRestoring = false;
      this.currentStep = '';
    },
    error: (error) => {
      console.error('Error al restaurar:', error);
      this.showNotification('Error durante la restauración', 'error');
      this.isRestoring = false;
      this.currentStep = '';
    }
  });
}
```

Figura 16.Codigo Frontend para la restauración

```java
public ResponseEntity<String> restoreSpecific(@RequestBody Map<String, String> payload, Authentication authentication) {
    try {
        Long projectId = Long.parseLong(payload.get("projectId"));
        String instance = payload.get("name");
        String category = payload.get("category");
        String dbFile = payload.get("dbBackupFileName");
        String odooFile = payload.get("odooBackupFileName");

        System.out.println("projectId "+projectId);
        System.out.println(" instance "+instance);
        System.out.println("category "+category);
        System.out.println("dbFile "+dbFile);
        System.out.println(" odooFile "+odooFile);

        String username = authentication.getName(); //  Extraer nombre del usuario autenticado

        String result = dockerService.restoreBackup(username, projectId, instance, category, dbFile, odooFile);

        return result.startsWith("✅")
                ? ResponseEntity.ok(result)
                : ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(result);

    } catch (Exception e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Error en los datos enviados.");
    }
}
```

Figura 17. Codigo del Backend para la restauración

La restauración se inicia desde el frontend seleccionando el grupo que se va a restaurar (archivos de odoo y base de datos). El componente valida la presencia de los archivos requeridos y extrae proyecto, categoría e instancias a partir del nombre del archivo. Con estos datos se crea un payload, gestionando estados de procesos y envía los datos mediante el servicio de backups como se observa en el codigo de la Figura 21.

En el Backend, el endpoint mapea el payload, donde se obtiene el usuario autentificado y llama al método restoreBackup y se procede a la restauración. Si fue exitoso se va a devolver codigo 200 y si falla codigo 500 o 400, como se observa en la Figura 22.

6.2.7 Visualizar logs de las instancias

Esta funcionalidad es una herramienta indispensable para poder diagnosticar y poder depurar.

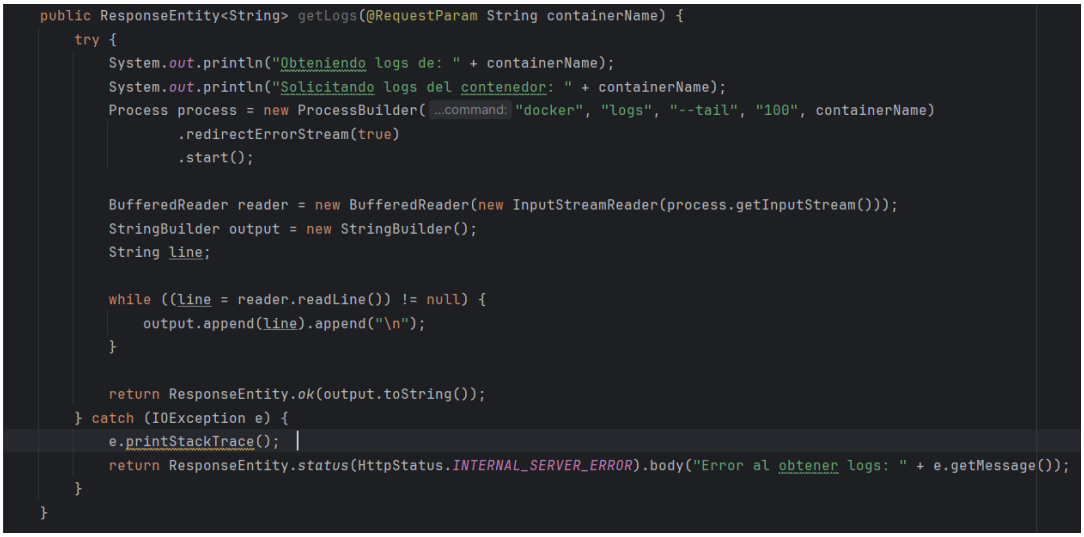
```
verLogs(containerName: string): void {
  this.selectedContainer = containerName;
  this.terminal.clear();
  this.terminal.writeln(`Viendo logs de: ${containerName}...`);

  if (this.intervalId) {
    clearInterval(this.intervalId);
  }

  this.fetchLogs(containerName);

  this.intervalId = setInterval(() => {
    this.fetchLogs(containerName);
  }, 10000);
}
```

Figura 18. Codigo de Frontend para los logs



```
public ResponseEntity<String> getLogs(@RequestParam String containerName) {
    try {
        System.out.println("Obteniendo logs de: " + containerName);
        System.out.println("Solicitando logs del contenedor: " + containerName);
        Process process = new ProcessBuilder( ...command: "docker", "logs", "--tail", "100", containerName)
                .redirectErrorStream(true)
                .start();

        BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
        StringBuilder output = new StringBuilder();
        String line;

        while ((line = reader.readLine()) != null) {
            output.append(line).append("\n");
        }

        return ResponseEntity.ok(output.toString());
    } catch (IOException e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Error al obtener logs: " + e.getMessage());
    }
}
```

Figura 19. Codigo de Backend para ver logs

La visualización de logs se gestiona mediante el frontend con el método verLogs, que selecciona el contenedor, limpia la terminal y muestra un encabezado inicial. Invoca periódicamente el servicio de obtención de logs cada 10 segundos, lo que permite actualizar la salida en tiempo real como se muestra en la figura 23.

En el Backend, el método getLogs, recibe el nombre del contenedor como parámetro y ejecuta el comando: docker logs --tail 100 <containerName> mediante un ProcessBuilder. El resultado se captura línea por línea y se devuelve como respuesta en texto plano. Si existiera error, retorna codigo 500 con el mensaje descriptivo. Todo esto se puede observar en la Figura 24.

6.2.8 Consultas de a la base de datos de las instancias

Se implemento una funcionalidad para poder realizar consultas SQL de forma segura, esto permite al usuario ejecutar comandos directamente a la base de datos de la instancia en específico.

```
executeCommand(): void {
  if (!this.command.trim()) {
    this.result = '! El comando no puede estar vacío';
    return;
  }

  this.isLoading = true;
  this.result = ' Ejecutando...';
  this.csvRows = [];

  const payload = {
    command: this.command,
    name: this.instanceName,
    category: this.category
  };

  console.log('Enviando comando con:', payload);

  this.shellService.executeDbCommand(payload).subscribe({
    next: (output: string) => {
      this.result = output;
      this.isLoading = false;

      // Intentar parsear como CSV
      this.csvRows = output
        .trim()
        .split('\n')
        .map(row => row.split(',').map(cell => cell.replace(/^"|"$/g, '').trim()));
    },
    error: (err) => {
      this.result = '❌ Error ejecutando comando: ' + (err?.error || 'Error desconocido');
      this.isLoading = false;
      this.csvRows = [];
    }
  });
}
```

Figura 20. Codigo desde el Frontend para la consulta

```java
public ResponseEntity<String> executeDbCommand(@RequestBody Map<String, String> payload) {
    String command = payload.get("command");
    String instanceName = payload.get("name");
    String category = payload.get("category");

    if (command == null || instanceName == null || category == null) {
        return ResponseEntity.badRequest().body("❌ Faltan datos requeridos.");
    }

    String output = dockerService.executeSqlCommandInInstance(command, instanceName, category);
    return ResponseEntity.ok(output);
}
```

Figura 21. Codigo de Backend para la consulta

La ejecución de comandos SQL se realiza desde el front mediante el método executeCommand. Primero se valida que el comando no este vacio, luego se construye un payload con el texto del comando, el nombre de la instancia y la categoría. Se envía al servicio, muestra un estado de carga y procesa la salida. Si el resultado tiene formato de texto tabular, se convierte en filas para poder visualizar, como lo muestra en la Figura 25

En el Backend el método executeDbCommand recibe el payload, valida que se incluya los campos requeridos y delega la ejecución al método executeSqlCommandInInstance y retorna un texto plano como respuesta como se observa en la Figura 26

6.2.9 Importar Base de Datos

Esta funcionalidad llega a ser complementaria e indispensable en el ciclo de vida de una instancia. Esto fue desarrollado con el fin de poder permitir a los usuarios que puedan reutilizar configuraciones previas, restaurar versiones antiguas o llegar a migrar sistemas desde otros entornos hacia la plataforma

```
importDatabase(fileInput: HTMLInputElement, form: any): void {
  if (!this.selectedDbFile || !this.selectedInstance) return;
  this.isImporting = true;
  const formData = new FormData();
  formData.append('dbFile', this.selectedDbFile);
  formData.append('name', this.selectedInstance.name);
  formData.append('category', this.selectedInstance.category);
  if (this.selectedOdooFile) {
    formData.append('odooFile', this.selectedOdooFile);
  }
  this.http.post('http://localhost:8080/docker/import-db-advanced', formData, {
    reportProgress: true,
    observe: 'events',
    responseType: 'text'
  }).subscribe({
    next: (event) => {
      if (event.type === HttpEventType.Response) {
        const responseText = event.body || 'Importación finalizada.';
        this.snackBar.open(responseText, 'Cerrar', { duration: 6000, panelClass: ['success-snackbar'] });
        this.selectedDbFile = null;
        this.selectedOdooFile = null;
        fileInput.value = '';
        form.resetForm();
        this.isImporting = false;
      }
    },
    error: (err) => {
      console.error(' Error al importar:', err);
      this.snackBar.open(' Error al importar.', 'Cerrar', { duration: 6000, panelClass: ['error-snackbar'] });
      this.isImporting = false;
    }
  });
}
```

Figura 22. Codigo del Frontend para la importación

```java
public ResponseEntity<String> importDbAndOdoo(
        @RequestParam("dbFile") MultipartFile dbFile,
        @RequestParam(value = "odooFile", required = false) MultipartFile odooFile,
        @RequestParam("name") String name,
        @RequestParam("category") String category) {

    try {
        String result = dockerService.importDatabaseAndOdooFiles(dbFile, odooFile, name, category);
        return ResponseEntity.ok(result);
    } catch (Exception e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body(" Error durante importación: " + e.getMessage());
    }
}

}
```

Figura 23. Codigo del Backend para la importación

La importación se gestiona desde el frontend mediante un formulario que permite seleccionar el archivo de base de datos o el archivo de Odoo. Con estos parámetros se construye un FormData que incluye también el nombre de la instancia y la categoría. La solicitud se envia al endpoint de importación con seguimiento de progreso, al completarse se notifica los resultados y se limpia los campos del formulario como se observa en la Figura 27.

En el Backend, el método importDbAndOdoo es el que recibe los archivos de entrada. La lógica del procesamiento fue delegada al método importDatabaseAndOdooFiles que se encarga de restaurar la base de datos y cargar los archivos en Odoo. Si se ejecuta con éxito, se retorna un mensaje de confirmación caso contrario envía un codigo de error 500 tal como lo podemos observar en la Figura 28

7. Cronograma

En esta sección se presenta una descripción de las actividades ejecutadas durante las fases de diseño y construcción del proyecto. Se exponen de forma cronológica las etapas que integraron el proceso, desde la planificación inicial hasta la implementación final del sistema. Cada una de estas fases fue ejecutada con un seguimiento constante, garantizando la coherencia metodológica y asegurando la efectividad en el cumplimiento de los objetivos planteados.

Nombre de la tarea	**Responsable**	**Duración (días)**	**Comienzo**	**Fin**
Proyecto		150	02-01-25	31-05-25
Fase 01: Requerimientos y planificación	SB – GP	25	02-01-25	26-01-25
OE.1		25		
ACT.1 Estudio de Odoo y requisitos técnicos	SB – GP	10	02-01-25	11-01-25
ACT.2 Diseño preliminar y arquitectura general	SB – GP	10	12-01-25	21-01-25
ACT.3 Revisión/validación inicial	SB – GP – CT	5	22-01-25	26-01-25
Fase 02: Desarrollo Backend	SB – GP	40	27-01-25	07-03-25
OE.2		40		
ACT.1 Implementación lógica de creación/gestión	SB – GP	15	27-01-25	10-02-25
ACT.2 Módulo de backups y restauración	SB – GP	10	11-02-25	20-02-25
ACT.3 Merge de instancias y eliminación	SB – GP	10	21-02-25	03-03-25

ACT.4 Seguridad básica (tokens/JWT)	SB – GP – CT	5	04-03-25	07-03-25
Fase 03: Desarrollo Frontend (Angular)	SB – GP	30	08-03-25	07-04-25
OE.3		30		
ACT.1 Dashboard, proyectos y sidebar	SB – GP	10	08-03-25	17-03-25
ACT.2 Backups, merge, shell y logs	SB – GP	10	18-03-25	27-03-25
ACT.3 Importación de base de datos	SB – GP	5	28-03-25	02-04-25
ACT.4 Pruebas de integración y validación frontend	SB – GP – CT	5	03-04-25	07-04-25
Fase 04: Pruebas Finales y Documentación	SB – GP – CT	30	08-04-25	10-05-25
OE.4		30		
ACT.1 Pruebas de funcionalidad y despliegue	SB – GP	15	08-04-25	22-04-25
ACT.2 Redacción de informe final	SB – GP	10	23-04-25	03-05-25
ACT.3 Correcciones y presentación	SB – GP – CT	5	04-05-25	10-05-25

Tabla 7. Cronograma de ejecución

8. Conclusiones

El proyecto presenta una plataforma SaaS diseñada para centralizar y automatizar el ciclo de vida de instancias personalizadas de Odoo CE, su objetivo es tener un plano de control único para desplegar, configurar, mantener y escalar múltiples entornos.

La arquitectura del sistema se apoya en contenedores y servicios desarrollados con Spring Boot y Angular, lo que permite establecer aislamientos independientes por

instancia y se estandarizan los procesos operativos, lo que reduce los tiempos de provisión y soporte. Asimismo, se incorporó una consola operativa en tiempo real, con sesiones persistentes a través de un WebSocket, que permite interactuar de forma remota y segura, disminuyendo la necesidad de accesos directos al servidor y mejorando la eficiencia del diagnóstico.

También, se integraron las funcionalidades para realizar copias de seguridad y la clonación de producción a staging, junto con una interfaz intuitiva y fácil de usar para el usuario.

Durante el desarrollo se consolidaron buenas prácticas de robustez y seguridad, incluyendo temas como validación de parámetros, aislamiento por instancia y el manejo confiable de flujos de datos, estos fortalecen la estabilidad del sistema y sienta las bases para su escalabilidad.

En conjunto, la solución entrega mejoras tangibles en eficiencia operativa, uniformidad entre entornos y reducción de riesgos. Todo ello deja al sistema preparado para evolucionar con capas de observabilidad, gobierno de accesos y orquestación avanzada, potenciando su crecimiento y mantenibilidad a largo plazo.

9. Trabajo a Futuro

- **Gestión de costos:** habilitar políticas de apagado programado en *staging*, etiquetado por proyecto y tableros de costos y de uso para decisiones informadas.
- **Cumplimiento normativo y privacidad:** alinear políticas con marcos de referencia (por ejemplo, la protección de datos personales), incluyendo el derecho al olvido y la minimización de datos.
- **Optimización del desempeño:** incorporar *pooling* de conexiones, *workers* para procesos en segundo plano y cachés de resultados para cargas reiterativas.
- **API de provisión:** exponer interfaces seguras para crear, clonar, pausar y escalar instancias, además de *webhooks* para integraciones con terceros.

- **Capas de observación enriquecidas:** ampliar la telemetría y los *runbooks* de respuesta a incidentes

10. Recomendaciones

En base a los resultados obtenidos durante el desarrollo de esta plataforma se plantearon las siguientes recomendaciones con el objetivo de garantizar la correcta adopción del sistema dentro de entornos pymes:

- Mantenimiento y monitoreo: realizar monitoreos periódicos de contenedores y volúmenes con el objetivo de evitar la existencia de fallos y garantizar la continuidad del servicio.
- Gestión de copias de seguridad: reforzar los procesos de generación y restauración de las copias de seguridad en el sistema mediante verificaciones automáticas de consistencia y notificaciones asegurando que el usuario puede tener confiabilidad en la recuperación de la información en caso de que existan fallos o errores.
- Documentación y manuales de usuario: Elaborar estas herramientas además de guías técnicas para que las personas que van a estar encargadas del sistema dentro de las empresas clientes puedan manejar la plataforma de manera sencilla sin requerir conocimientos o contratación de expertos en desarrollo o manejo de ERPs lo que reduce costos.
- Mejora de experiencia de usuario: optimizar la interfaz con métricas y visualizaciones que ayuden a facilitar la gestión de instancias que ayuden a una claridad sobre el estado del sistema

11. Referencias

(AWS), A. W. (2024). *Computación en la nube con AWS*. Obtenido de Amazon Web Services (AWS): https://aws.amazon.com/es/what-is-aws/

Al-Mashari, M. (2003). Enterprise resource planning (ERP) systems: a research agenda. *Industrial Management & Data Systems*, 22-27.

Amazon web service. (2025). *What's the Difference Between Docker Images and Containers?* Obtenido de AWS: https://aws.amazon.com/es/compare/the-difference-between-docker-images-and-containers/

Appian. (2 de marzo de 2023). *¿Qué es la automatización inteligente de procesos (IPA)?* Obtenido de Appian Blog: https://appian.com/es/blog/acp/process-automation/what-is-intelligent-process-automation

Arbañil, C. A. (2023). TRANSFORMACIÓN DIGITAL COMO ESTRATEGIA DE MEJORA EN LAS ORGANIZACIONES. *ResearchGate*, 1577-169.

arteco-consulting. (2023). *Tutorial de Docker*. Obtenido de Arteco Consulting: https://www.arteco-consulting.com/articulos/tutorial-docker/

Atlassian. (2025). *¿Qué es la computación en la nube? Visión general de la nube*. Obtenido de Atlassian: https://www.atlassian.com/es/microservices/cloud-computing

Barghoth, M. E. (2020). A comprehensive software project management framework. Journal of Computer and Communications. *Journal of Computer and Communications*, 51-69.

Bengochea, D. (24 de Enero de 2025). *Los 12 mejores ERP del mercado para tu negocio*. Obtenido de Outvio: https://outvio.com/es/blog/mejores-erp/

Businessmap. (14 de JUNIO de 2023). *Automatización de procesos de negocio (BPA): guía completa*. Obtenido de Businessmap: https://businessmap.io/es/gestion-de-procesos-bpm/automatizacion-de-procesos-de-negocio

Calles-García, J., & González-Pérez, P. (2011). *La Biblia del Footprinting.*

Comisión Económica para América Latina y el Caribe (CEPAL). (6 de Noviembre de 2024). *CEPAL*. Obtenido de CEPAL: https://www.cepal.org/es/comunicados/transformacion-digital-real-efectiva-puede-ayudar-america-latina-caribe-superar-trampas

Cybrosys Technologies. (2025). *Odoo.SH: Manage Branches*. Obtenido de Cybrosys Technologies: https://www.cybrosys.com/odoo/odoo-books/odoo-16-development/odoo-sh/manage-branches/

DEV Community. (15 de Septiembre de 2024). *Inversion of Control in Spring Framework.* Obtenido de DEV.to: https://dev.to/be11amer/inversion-of-control-in-spring-framework-4mc0

Forsgren, N., Humble, J., & Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps – Building and Scaling High Performing Technology Organizations.* Portland, OR: IT Revolution Press.

Freeman, A. (2022). *Pro Angular 13.* Reino Unido: Apress.

García, J. &. (2021). ERP systems and business performance in SMEs: Evidence from Latin America. *Journal of Small Business Management*, 712-730.

geeksforgeeks. (7 de marzo de 2025). *Understanding Inversion of Control with Example.* Obtenido de GeeksforGeeks: https://www.geeksforgeeks.org/spring-understanding-inversion-of-control-with-example/

Gharabti, A., El Kortbi, I., Nekhass, H., & Bendahmane, A. (2025). Enhancing HR communication and retention in Odoo ERP through AI integration. *Brazilian Journal of Education, Technology and Society (BRAJETS)*, 1-13.

Google Developers. (2025). *Build maintainable and scalable web apps with Angular.* Obtenido de Google Developers: https://developers.google.com/web/tools/angular

Haro, A., Martínez, J., Chango, D., Zambrano, E., Recalde, R., & Rojas, C. (2023). Enterprise Resource Planning (ERP) procesos para una implementación óptima y eficiente. *Prometeo Conocimiento Científico*, 123–138.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Boston: Addison-Wesley.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation.* Upper Saddle River, NJ: Addison-Wesley Professional.

IBM. (2025). *¿Qué son IaaS, PaaS y SaaS?* Obtenido de IBM: https://www.ibm.com/mx-es/think/topics/iaas-paas-saas

IBM. (2025). *Java Spring Boot*. Obtenido de IBM: https://www.ibm.com/mx-es/topics/java-spring-boot

Jackson, Gita. (13 de Diciembre de 2024). *What Is the Software Development Life Cycle (SDLC)?* Obtenido de IBM: https://www.ibm.com/think/topics/sdlc

Klaus, H., Rosemann, M., & Gable, G. (2023). What is ERP? 20 years on. *Journal of Information Technology*, 25-41.

Marques, R., & Ferreira, J. (2021). Digital transformation in SMEs: A systematic literature review. *Journal of Small Business and Enterprise Development*, 855-876.

Medallo, D. (2020). DevOps: hacia una cultura de integración y despliegue continuo. *Revista de Ingeniería de Software*, 45–58.

Monk, E. (2021). *Concepts in enterprise resource planning.* Boston: Cengage Learning.

NetApp. (2025). *¿Qué son los contenedores?* Obtenido de NetApp: https://www.netapp.com/es/devops/what-are-containers

Odoo . (2024). *Odoo: Open Source ERP and CRM*. Obtenido de Odoo: https://www.odoo.com/es_ES

Páez-Gabriunas, I. (2021). *Transformación digital en las organizaciones.* Bogota: Universidad del Rosario.

Pivotal / Spring Team. (2018). *Embedded Web Servers.* Obtenido de Spring Boot Reference Guide: https://docs.spring.io/spring-boot/docs/2.0.6.RELEASE/reference/html/howto-embedded-web-servers.html

Red.es. (8 de noviembre de 2022). *Red.es.* Obtenido de Red.es web site: https://www.red.es/es/actualidad/noticias/la-transformacion-digital-de-los-procesos-de-negocio-toma-fuerza-en-el-tejido

SAP España. (12 de Julio de 2023). *La automatización en las empresas: beneficios y retos*. Obtenido de SAP News Center: https://news.sap.com/spain/2023/07/automatizacion-en-las-empresas

Servicio Ecuatoriano de Normalización (INEN). (15 de Octubre de 2021). *MiPymes y organizaciones de economía popular y solidaria son una pieza clave para la economía del país*. Obtenido de Normalización Ecuatoriana: https://www.normalizacion.gob.ec/mipymes-y-organizaciones-de-economia-popular-y-solidaria-son-una-pieza-clave-para-la-economia-del-pais/

Solano María José [Vauxoo]. (22 de enero de 2025). *Todo lo que necesitas saber sobre Odoo.sh: funciones y beneficios*. Obtenido de Vauxoo: https://www.vauxoo.com/en_US/blog/functional-hacks-4/todo-lo-que-necesitas-saber-sobre-odoo-sh-funciones-y-beneficios-260

Spring. (2025). *Building REST services with Spring.* Obtenido de Spring Guides: https://spring.io/guides/tutorials/rest/

Teco Tutorials. (26 de Octubre de 2024). *Implement Role-based Access Control in Spring Boot 3.* Obtenido de Teco Tutorials: https://blog.tericcabrel.com/role-base-access-control-spring-boot

Usman, U. Z., Ahmad, M., Zakaria, N. H., & Alkurdi, A. H. (2017). A review of key factors of cloud enterprise resource planning (ERP) adoption by SMEs. *Journal of Theoretical and Applied Information Technology*, 3884–3901.

Velneo. (6 de Marzo de 2024). *¿Qué es Odoo? Ventajas, inconvenientes y alternativa*. Obtenido de Velneo: https://velneo.com/blog/alternativa-odoo-ventajas-inconvenientes/

VMware / Spring Team. (2023). *Spring Boot Reference Documentation (v 3.0.14).* Obtenido de docs.spring.vmware.com: https://docs.spring.vmware.com/spring-boot/docs/3.0.14/reference/htmlsingle/

vrajatechnologies. (28 de junio de 2024). *The Ultimate Guide to ODOO Server Hosting in 2024*. Obtenido de Vraja Technologies: https://www.vrajatechnologies.com/blog/1/the-ultimate-guide-to-odoo-server-hosting-in-2024-14

ANEXO *1*

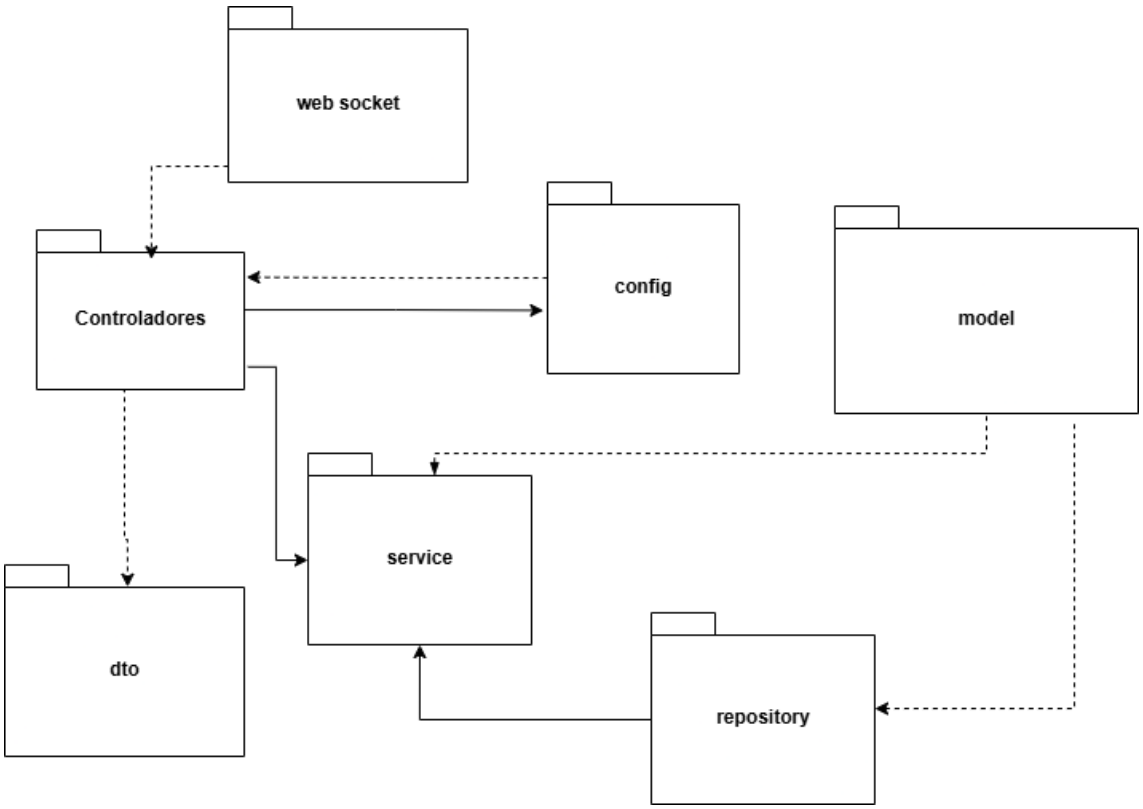


Figura 24. Diagrama por capas del backend

ANEXO 2

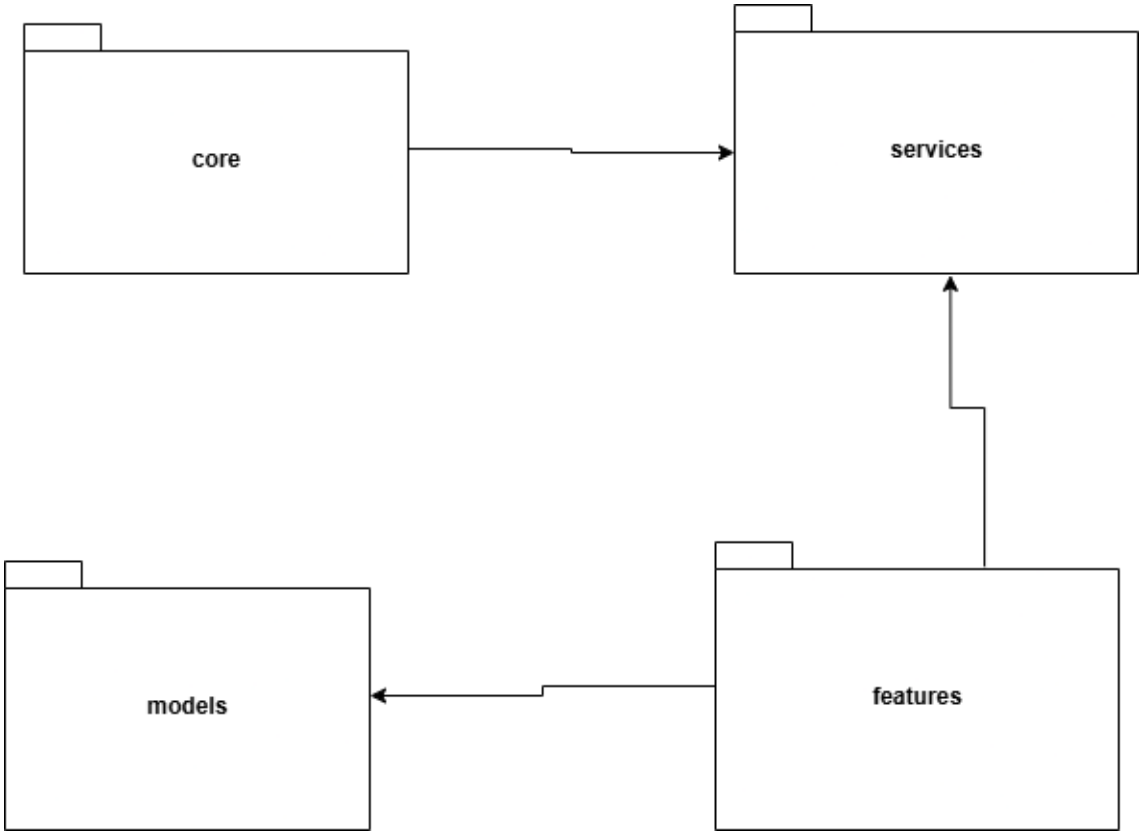


Figura 25. Diagrama del Frontend

ANEXO 3

Los requerimientos que se implementaron en el proyecto creado. A continuación, se detallarán cada uno de los requerimientos funcionales y no funcionales:

<table>
<tr><td>Identificador:
RF01</td><td>Nombre:
Creación de instancias</td><td>Funcional</td></tr>
<tr><td colspan="2">Descripción:
Implementar una funcionalidad para generar instancias de odoo desde la interfaz, facilitando la gestión de un entorno individual</td><td>Categoría (Visible/ No Visible):
Visible</td></tr>
<tr><td colspan="3">Objetivo:
Permitir la creación de entornos dinámicos de Odoo para cada cliente registrado diferenciando entre clientes los proyectos creados</td></tr>
<tr><td colspan="2">Datos de entrada:
Nombre del proyecto
Versión
Puerto
Puerto base de datos</td><td>Datos de salida:
• Instancia desplegada
• Redirección a la instancia creada</td></tr>
<tr><td colspan="3">Criterios de aceptación:
El sistema va a permitir la creación de instancias del ERP sin errores mostrando mensajes de éxito, dicha entidad contiene los módulos base de Odoo</td></tr>
<tr><td colspan="3">Prioridad:
Alta</td></tr>
</table>

Tabla 8. Requerimiento Funcional RF01, Crear instancias

<table>
<tr><td>Identificador:
RF02</td><td>Nombre:
Consulta de la base de datos de la entidad</td><td>Funcional</td></tr>
<tr><td colspan="2">Descripción:
Permitir la visualización, modificación y consultas en la base de datos de la entidad a través de una interfaz para generar querys controlados</td><td>Categoría (Visible/ No Visible):
Visible</td></tr>
<tr><td colspan="3">Objetivo:
Brindar soporte a los datos o modificaciones de registro en caso de errores mediante un acceso seguro a los datos del sistema</td></tr>
<tr><td colspan="2">Datos de entrada:
Query de búsqueda
Filtros</td><td>Datos de salida:
• Datos resultantes del Query</td></tr>
<tr><td colspan="3">Criterios de aceptación:
El sistema podrá realizar búsquedas, modificaciones y eliminaciones sin afectar el rendimiento del sistema</td></tr>
<tr><td colspan="3">Prioridad:
Media</td></tr>
</table>

Tabla 9. Requerimiento Funcional RF02, Consultar a la base de datos

<table>
<tr><td>Identificador:
RF03</td><td>Nombre:
Neutralización de datos (Rama Stagging)</td><td>Funcional</td></tr>
<tr><td colspan="2">Descripción:
Implementar una funcionalidad que pueda crear una base de dato para pruebas la cual utilizara los datos de producción, pero desactivando ciertos componentes como servidores de correo, pagos y envíos</td><td>Categoría (Visible/ No Visible):
Visible</td></tr>
<tr><td colspan="3">Objetivo:
Garantizar el uso de un ambiente de pruebas con datos reales sin la necesidad de estar en un ambiente de producción</td></tr>
<tr><td colspan="2">Datos de entrada:
Entorno de producción
Base de datos</td><td>Datos de salida:
• Copia neutralizada de base de datos con registros utilizados y creados desde un ambiente de producción</td></tr>
<tr><td colspan="3">Criterios de aceptación:
Crear un ambiente que sirva para realizar pruebas dentro de un entorno copia de producción que, no realice algunas funcionalidades que puedan modificar datos y conectarse con otros sistemas</td></tr>
<tr><td colspan="3">Prioridad:
Alta</td></tr>
</table>

Tabla 10.Requerimiento Funcional RF03, Neutralización de datos en la Rama Stagging

<table>
<tr><td>Identificador:
RF04</td><td>Nombre:
Importación de base de datos</td><td>Funcional</td></tr>
<tr><td colspan="2">Descripción:
Posibilitar al usuario un método para la capacidad de cargar bases de datos externar, restaurar o migrar información en formato de odoo</td><td>Categoría (Visible/ No Visible):
Visible</td></tr>
<tr><td colspan="3">Objetivo:
Agilizar la migración de datos de un entorno a otro</td></tr>
<tr><td colspan="2">Datos de entrada:
Archivo. Dump o un archivo comprimido que contiene el Filestore y el archivo. dump</td><td>Datos de salida:
• Instancia con Datos y módulos cargados importados</td></tr>
<tr><td colspan="3">Criterios de aceptación:
El aplicativo va a poder cargar archivos válidos y notificar e imposibilitar la carga en caso de ser necesario</td></tr>
<tr><td colspan="3">Prioridad:
Alta</td></tr>
</table>

Tabla 11. Requerimiento Funcional RF04, Importar Base de Datos

<table>
<tr><td>Identificador:

RF05</td><td>Nombre:

Fusión de ramas (Merge)</td><td>Funcional</td></tr>
<tr><td colspan="2">Descripción:

Habilitar la opción de fusionar entidades entre ramas del repositorio desde una interfaz</td><td>Categoría (Visible/ No Visible):

Visible</td></tr>
<tr><td colspan="3">Objetivo:

Permitir cambios entre entornos tres tipos de ramas las cuales son producción, pruebas y desarrollo.</td></tr>
<tr><td colspan="2">Datos de entrada:

Rama origen

Rama destino

Mensaje de confirmación</td><td>Datos de salida:

• Mensaje de fusión exitosa</td></tr>
<tr><td colspan="3">Criterios de aceptación:

Se debe realizar la fusión notificando al usuario y controlando versiones</td></tr>
<tr><td colspan="3">Prioridad:

Alta</td></tr>
</table>

Tabla 12.Requerimiento Funcional RF05, Fusionar Ramas

Identificador: RNF01	Nombre: Gestión de Backups	No Funcional
Descripción: Desarrollar un mecanismo para la realización de copias de seguridad periódicas de la entidad de Odoo		**Categoría (Visible/ No Visible):** No visible
Objetivo: Proteger los datos en el caso de que exista una falla o algún problema que requiera volver a una versión anterior de la base		
Datos de entrada: Ruta de almacenamiento Fecha de la instancia		**Datos de salida:** • Respaldo generado • Notificación de generación de respaldo
Criterios de aceptación: El software podrá generar copias de seguridad automáticas según la configuración y según la necesidad del cliente		
Prioridad: Alta		

Tabla 13.Requerimiento No Funcional RNF01, Gestionar Backups

<table>
<tr><td>Identificador:
RNF02</td><td>Nombre:
Visualización de actividades en el sistema</td><td>No Funcional</td></tr>
<tr><td colspan="2">Descripción:
Mantener un registro accesible del sistema permitiendo visualizar acciones, errores y modificaciones en datos desde interfaz</td><td>Categoría (Visible/ No Visible):
Visible</td></tr>
<tr><td colspan="3">Objetivo:
Proveer una solución que facilite la auditoría, análisis de errores además del comportamiento del sistema durante su ejecución.</td></tr>
<tr><td colspan="2">Datos de entrada:
Tipo de log</td><td>Datos de salida:
• Registro de los logs o actividades en un formato entendible</td></tr>
<tr><td colspan="3">Criterios de aceptación:
El aplicativo debe mostrar logs organizados según su tiempo de ejecución el cual será filtrado según la necesidad del usuario</td></tr>
<tr><td colspan="3">Prioridad:
Alta</td></tr>
</table>

Tabla 14. Requerimiento No Funcional RNF02, Visualizar actividades en el sistema

<table>
<tr><td>Identificador:

RNF03</td><td>Nombre:

Datos demo en la rama de desarrollo</td><td>No Funcional</td></tr>
<tr><td colspan="2">Descripción:

Establecer datos precargados demo para configuración y desarrollo</td><td>Categoría (Visible/ No Visible):

No visible</td></tr>
<tr><td colspan="3">Objetivo:

Mejorar el desarrollo mediante un entorno preconfigurado con registros que se crean al momento de generar la instancia de Odoo</td></tr>
<tr><td colspan="2">Datos de entrada:

Instancia Creada

Datos de creación de instancia</td><td>Datos de salida:

• Entorno de desarrollo con datos demo</td></tr>
<tr><td colspan="3">Criterios de aceptación:

La instancia en la rama de develop debe incluir automáticamente productos, contactos y configuraciones sin necesidad de estas ser creadas por el usuario</td></tr>
<tr><td colspan="3">Prioridad:

Media</td></tr>
</table>

Tabla 15. Requerimiento No Funcional RNF03, Data demo en la rama Development

ANEXO 4

El archivo README.md forma parte del repositorio principal del proyecto y tiene como objetivo describir de manera general el propósito, la estructura y las instrucciones de instalación y uso de la plataforma desarrollada.

Enlace Frontend: https://github.com/Paulguzhnay/TesisMaestria/tree/FrontEndF

Enlace Backend: https://github.com/Paulguzhnay/TesisMaestria/tree/backend