UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN

SOFTWARE CON MENCIÓN EN DISEÑO DE ARQUITECTURA DE SISTEMAS

RPC-SO-34-NO.778-2021

OPCIÓN DE TITULACIÓN:

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

TEMA:

DESARROLLO DE UN PROTOTIPO DE SISTEMA APLICANDO ARQUITECTURA E INTEGRACIÓN CONTINUA Y DESPLIEGUE CONTINUO PARA EL REGISTRO Y CONTROL DE LOS ALMUERZOS PARA LOS EMPLEADOS DE LA EMPRESA LINKS.

AUTOR(ES)

JOSÉ LUIS SÁNCHEZ BAQUE

DIRECTOR:

MÓNICA DANIELA GÓMEZ RIOS

GUAYAQUIL – ECUADOR

2025

Autor(es):

José Luis Sánchez Baque
Ingeniero en Sistemas Computacionales
Candidato a Magíster en Software por la Universidad Politécnica Salesiana – Sede Guayaquil.
sanchezbaquejoseluis@gmail.com

Dirigido por:

Mónica Daniela Gómez Ríos
Ingeniero de Sistemas
Magister en Ciencias de la Computación Mención Networking
Magister en Ciencias de la Computación Mención Aplicaciones Distribuidas
mgomezr@ups.edu.ec

José Luis Sánchez Baque
Desarrollo de un prototipo de sistema aplicando arquitectura e integración continua y despliegue continuo para el registro y control de los almuerzos para los empleados de la empresa Links.

DEDICATORIA

Mi gratitud más sincera es para Dios, cuya presencia ha representado orientación y un acompañamiento firme durante cada etapa de este camino. En los momentos de mayor exigencia encontré en Él serenidad, fortaleza y claridad para seguir adelante, renovando mi ánimo y permitiéndome continuar con firmeza en la búsqueda del conocimiento.

A mi familia, y en particular a mis padres, quienes han sido un apoyo invaluable en mi vida, les dedico estas palabras con profundo afecto. Su acompañamiento constante, sus muestras de afecto y la calidez que siempre me brindaron se convirtieron en el impulso necesario para avanzar incluso en las etapas más retadoras. Agradezco su comprensión ante los sacrificios que demanda esta formación y por estar siempre presentes con paciencia y dedicación.

A mí mismo por la perseverancia en los días difíciles y por la dedicación constante en la búsqueda de conocimiento, lo ofrezco a mí mismo, como evidencia del camino recorrido y de la firmeza con la que enfrenté cada desafío.

AGRADECIMIENTO

Mi agradecimiento más profundo es para Dios, cuya compañía ha inspirado y fortalecido cada paso que di durante este recorrido formativo. En los instantes más complejos, sentirlo cerca me dio claridad y renovó mi ánimo para continuar con firmeza en la búsqueda del conocimiento.

A mis padres y hermanas, quienes han sido mi principal sostén en la vida, les dedico estas líneas con inmenso cariño. Su respaldo incondicional, sus gestos de ánimo y el afecto que siempre me ofrecieron fueron la fuerza que me impulsó a avanzar incluso en las etapas más complejas. Agradezco su comprensión ante los sacrificios que implica la formación profesional y su constante acompañamiento lleno de paciencia y dedicación.

A mi tutora, le expreso una especial gratitud por la dedicación con la que me orientó en cada fase de este trabajo. Su paciencia, claridad y compromiso académico fueron clave para fortalecer este proyecto y para mi crecimiento como profesional. Su guía ha sido un pilar fundamental en esta etapa.

A mis profesores, agradezco el conocimiento compartido y el compromiso demostrado durante toda la maestría. Sus enseñanzas ampliaron mi visión profesional y contribuyeron de manera significativa a mi desarrollo personal y académico.

Tabla de Contenido

Índice de Figuras 7
Índice de Tablas 8
Resumen 10
Abstract 11
1. Introducción 12
1.1 Antecedentes 13
1.2 Determinación del problema 14
1.2.1 Descripción del problema 14
1.2.2 Formulación del problema 15
1.2.3 Justificación del problema 15
1.2.4 Delimitación del problema 15
1.3 Justificación del problema 16
1.4 Objetivos 18
1.4.1 Objetivo general 18
1.4.2 Objetivos específicos 18
2. Marco teórico referencial 19
2.1 Estado del arte 19
2.2 Definiciones previas 21
2.4.1 Arquitectura de software 21
2.4.2 CI/CD 21
2.4.3 Patrones de diseño 21
2.4.4 Api rest 23
2.4.5 DevOps 24
2.4.6 Metodología Lean Startup 24
3. Materiales y metodología 26
3.1 Marco Metodológico 26
3.1.1 Tipo de investigación 26
3.1.2 Enfoque de investigación (mixto con predominio cualitativo) 26
3.1.3 Metodología aplicada: Lean Startup 27
3.1.4 Prácticas complementarias: DevOps e integración continua (CI/CD) 27
3.1.5 Trazabilidad metodológica entre objetivos e iteraciones 27

3.2 Proceso Metodológico 28
3.3 Herramientas y tecnologías 31
3.4 Unidad de análisis 33
3.5 Técnicas e instrumentos de recolección datos (enfoque por requerimientos) 36
3.6 Procedimiento de implementación 41
3.6.1 Preparación del entorno 41
3.6.2 Configuración de repositorios 41
3.6.3 Configuración de la base de datos 42
3.6.4 Implementación del pipeline CI/CD (Jenkins) 43
3.6.5 Despliegue controlado y verificación 44
3.7 Limitaciones y dificultades 45
4. Resultados y discusión 48
4.1 Resultados del desarrollo del sistema 48
4.2 Frontend (Angular 19) 52
4.3 Backend (Laravel 12) 54
4.4 Documentación Api del backend 56
4.5 Análisis de código e integración continua 58
4.6 Configuración del servidor y Nginx 62
4.7 Validación práctica del sistema 63
4.8 Interpretación de resultados 66
4.8.1 Indicador 1: Tiempo promedio de registro (antes vs. después) 66
4.8.2 Indicador 2: Errores detectados y corregidos por iteración 68
4.8.3 Indicador 3: Aceptación del prototipo por parte de los usuarios 69
4.8.4 Indicador 4: Estabilidad técnica del pipeline CI/CD 71
4.9 Discusión de resultados 73
5. Conclusiones 76
Referencias 77

Índice de Figuras

Ilustración 1 *Proceso metodológico del sistema de registro de almuerzos* 29
Ilustración 2 *Arquitectura general del sistema* 52
Ilustración 3 *Estructura del proyecto Frontend (Angular 19)* 53
Ilustración 4 *Arquitectura general del backend (Laravel 12)* 55
Ilustración 5 *Documentación del api* 57
Ilustración 6 *Flujo del pipeline CI/CD del sistema de registro de almuerzos* 58
Ilustración 7 *Repositorio GitHub* 59
Ilustración 8 *Jenkins - pipeline-overview* 60
Ilustración 9 *SonarQube – análisis de código* 60
Ilustración 10 *Configuración Nginx* 62
Ilustración 11 *Lista de empleado - agregar* 64
Ilustración 12 *Listar menús – ingresar nuevo menú* 64
Ilustración 13 *Listar menús – editar menú existente* 64
Ilustración 14 *Listar empleados – registrar almuerzo* 65
Ilustración 15 *Empleado – lista de almuerzos – agregar calificación* 65
Ilustración 16 *Proveedor – reporte* 66
Ilustración 17 *Captura de la medición técnica del tiempo de registro mediante la herramienta Network del navegador.* 67
Ilustración 18 *Pull Requests con bugs corregidos.* 69
Ilustración 19 *Formulario utilizado para evaluar la aceptación del prototipo.* 71
Ilustración 20 *Ejecución de la build en Jenkins* 73

Índice de Tablas

Tabla 1 *Patrones de diseño aplicados en el sistema*..... 22
Tabla 2 28
Tabla 3 *Iteraciones del proceso metodológico del sistema de registro de almuerzos* ... 30
Tabla 4 *Herramientas y tecnologías empleadas en el sistema de registro de almuerzos* 31
Tabla 5 *Roles participantes y alcance de validación del sistema* 34
Tabla 6 *Resumen de iteraciones (visión general)* 37
Tabla 7 *Iteración 01 — Acceso y control*..... 37
Tabla 8 *Iteración 02 — Maestros institucionales* 38
Tabla 9 Iteración 03 — Operación diaria (menús y registro) 39
Tabla 10 *Iteración 04 — Valor al usuario (calificaciones y reporte)*..... 39
Tabla 11 *Iteración 05 — Iteración V — Entorno, despliegue y calidad* 40
Tabla 12 *Reglas transversales (aplican a todas las iteraciones)* 40
Tabla 13 *Resumen del flujo CI/CD aplicado al sistema (backend, frontend y análisis de calidad)* 43
Tabla 14 *Limitaciones, dificultades y estrategias de mitigación* 45
Tabla 15 *Resultado alcanzado en el desarrollo del sistema de registro de almuerzos* .. 49
Tabla 16 *Aporte del frontend al OE1* 54
Tabla 17 *Aporte del backend al OE1*..... 56
Tabla 18 *Aporte de la documentación al OE1* 57
Tabla 19 *Aporte del proceso CI/CD al cumplimiento del OE2*..... 61
Tabla 20 *Resultados de la validación práctica del sistema* 63
Tabla 21 *Comparación vertical del tiempo de registro manual y automatizado* 67
Tabla 22 *Errores detectados y corregidos por iteración*..... 68
Tabla 23 *Porcentaje de aceptación del prototipo (n = 18)* 70
Tabla 24 *Métricas técnicas del pipeline CI/CD*..... 72
Tabla 25 *Comparación entre objetivos específicos, resultados alcanzados y nivel de cumplimiento*..... 74

Desarrollo de un prototipo de sistema aplicando arquitectura e integración continua y despliegue continuo para el registro y control de los almuerzos para los empleados de la empresa Links

Autor(es):

José Luis Sánchez baque

Resumen

El proyecto desarrolla un prototipo web para la gestión y control de almuerzos institucionales de la empresa Links, mediante una investigación aplicada con enfoque mixto y predominio cualitativo. La propuesta integra una arquitectura moderna junto con prácticas de integración continua y despliegue continuo (CI/CD), aplicadas bajo la metodología Lean Startup para validar funcionalidades mediante ciclos iterativos con usuarios reales.

El sistema fue implementado con Laravel 12 y Angular 19, comunicados mediante API REST y desplegados en un entorno Ubuntu Server 22.04 con Nginx y certificados SSL. La automatización del ciclo de desarrollo se gestionó con Jenkins y SonarQube, garantizando trazabilidad, calidad del código y estabilidad técnica del pipeline CI/CD.

Los resultados evidencian mejoras en tiempos de registro, reducción de errores y mayor claridad en la información. El aporte principal del trabajo consiste en modernizar un proceso crítico mediante una arquitectura escalable y un pipeline CI/CD confiable, contribuyendo a la transformación digital de la empresa Links.

Palabras clave:

Arquitectura de software; integración continua; despliegue continuo; Lean Startup; DevOps; Laravel; Angular; CI/CD.

Abstract

This project presents the development of a web-based prototype for managing and controlling institutional lunch services at the company Links, conducted through an applied research approach with a mixed methodology and a predominantly qualitative focus. The proposal integrates a modern software architecture along with Continuous Integration and Continuous Deployment (CI/CD) practices, implemented under the Lean Startup methodology to validate functionalities through iterative cycles with real users.

The system was built using Laravel 12 and Angular 19, communicating through REST APIs and deployed on an Ubuntu Server 22.04 environment with Nginx and SSL certificates. The development workflow was automated using Jenkins and SonarQube, ensuring code quality, traceability, and technical stability throughout the CI/CD pipeline.

The results show significant improvements in registration times, error reduction, and greater clarity in the information generated. The main contribution of this work is the modernization of a critical institutional process through a scalable architecture and a reliable CI/CD pipeline, supporting the digital transformation of the company Links.

Palabras clave:

Software architecture; continuous integration; continuous deployment; Lean Startup; DevOps; Laravel; Angular; CI/CD.

1. Introducción

En la actualidad, las organizaciones requieren procesos eficientes y confiables para administrar sus recursos operativos. La empresa Links, debido al aumento de su personal, enfrenta dificultades para mantener un control preciso sobre la provisión alimentaria que entrega diariamente. El procesamiento manual que se utiliza en la actualidad genera retrasos, inconsistencias y una limitada trazabilidad de la información. Esto tiene un impacto en la transparencia del proceso y también limita la habilidad de las instituciones para tomar decisiones fundamentadas en datos precisos.

Frente a este escenario, se propone el desarrollo de una solución tecnológica que haga posible la automatización del proceso, mejorar la precisión en los registros y ofrecer información centralizada y verificable. El proyecto adopta una investigación aplicada, orientada a resolver un problema real dentro de la empresa. Además, se utiliza un enfoque mixto con predominio cualitativo, que combina datos técnicos con la experiencia directa de los usuarios que utilizan el sistema durante su validación.

La propuesta se sustenta en una arquitectura moderna, basada en servicios web, acompañada de prácticas de Integración Continua y Despliegue Continuo (CI/CD) para asegurar estabilidad, trazabilidad y calidad en cada entrega. Estas prácticas se complementan con la metodología Lean Startup, aplicada mediante ciclos construir–medir–aprender, que permiten validar hipótesis, ajustar funcionalidades y evolucionar la solución según la retroalimentación de los usuarios internos. Este enfoque conjunto promueve innovación, rapidez de mejora y sostenibilidad del sistema.

El documento está organizado en cinco secciones. El primero presenta el problema, los objetivos y su justificación. El segundo desarrolla el marco teórico y los antecedentes pertinentes. El tercero detalla el proceso metodológico, las técnicas que se emplean y los materiales que se utilizan. El cuarto expone los resultados

conseguidos y su respectivo análisis. Finalmente, el quinto capítulo recoge las conclusiones generales del proyecto.

1.1 Antecedentes

La empresa Links ha registrado un incremento en su personal, lo cual ha intensificado las actividades asociadas a la gestión diaria de almuerzos. Este proceso, realizado actualmente de forma manual, ha generado demoras, registros imprecisos y dificultades para mantener un control eficiente. En este escenario, contar con un sistema tecnológico que permita registrar, organizar y supervisar los almuerzos se vuelve una necesidad para mejorar la eficiencia institucional y asegurar una adecuada administración de recursos.

Un antecedente relevante es el estudio realizado por (Delaney et al., 2022) sobre el sistema Flexischools en Australia, una plataforma utilizada en más de 800 instituciones educativas para gestionar pedidos de almuerzos de estudiantes. Los autores evidencian que, cuando el volumen de usuarios es alto, la digitalización del proceso mejora la logística, reduce errores y facilita la planificación operativa. Este caso demuestra la pertinencia de implementar soluciones tecnológicas en contextos donde se manejan grandes cantidades de registros alimentarios, como ocurre en Links.

La literatura destaca la importancia de incluir prácticas contemporáneas, entre ellas la integración y entrega continua (CI/CD). De acuerdo con (Bajpai & Lewis, 2022) explican que estos procesos permiten automatizar tareas críticas —compilación, pruebas y despliegue— reduciendo riesgos operativos y garantizando la trazabilidad del sistema. Este antecedente es especialmente relevante para el presente proyecto, que incluye prácticas DevOps para asegurar entregas sean frecuentes y fiables.

Finalmente, la búsqueda de metodologías que aceleren el desarrollo de productos tecnológicos también ha sido ampliamente estudiada.(Veretennikova & Vaskiv, 2018) sostienen que los ciclos cortos de construcción y validación, propios de Lean

Startup, favorecen la generación temprana de retroalimentación y la disminución de riesgos en proyectos innovadores. De manera similar, investigaciones aplicadas en instituciones educativas —como la realizada en el Politécnico Estatal de Jember— demuestran que trabajar con prototipos mínimos viables (MVP) permite ajustar funcionalidades de forma ágil y basada en evidencia. Este enfoque sustenta el uso de Lean Startup en el desarrollo del prototipo para Links.

1.2 Determinación del problema

Actualmente dentro de la empresa se recalca la importancia de un sistema que centralice y optimice todas las tareas involucradas. La implementación de este prototipo que ayude a la automatización, respaldado por la arquitectura moderna y prácticas de CI/CD, se presenta como una respuesta estratégica para superar este desafío y sentar las bases para lograr tener un manejo eficiente de los almuerzos en Links.

1.2.1 Descripción del problema

Actualmente se lleva una gestión manual de los almuerzos dentro de la empresa, lo que ha causado que exista un registro de datos imprecisos y es por esta razón que no se tiene una fiabilidad de esta información recopilada. Esto afecta la toma de decisiones y también genera mal entendidos y conflictos en la asignación.

Las ineficiencias operativas que actualmente se tienen dentro de la organización son porque no existe un sistema centralizado en el cual se pueda llevar un registro diario de todos los almuerzos ya que actualmente todo esto se hace de forma manual y no se puede tener una correcta organización, además que esto consume tiempo y recursos. Por tal motivo se requiere de este prototipo para los responsables puedan tomar decisiones con los datos recopilados, obstaculizando la planificación y el ajuste ágil de los recursos.

La gestión ineficiente de los almuerzos repercute de forma diaria y directamente sobre la experiencia que tienen los empleados ya que se han generado confusiones y retrasos al momento de la entrega.

1.2.2 Formulación del problema

Para este proyecto de tesis se plantea como desafío central la siguiente pregunta: "¿Cómo podemos desarrollar e implementar un prototipo de sistema innovador para gestionar los almuerzos de los empleados de la empresa Links, mejorando la eficiencia y transparencia, mediante el uso de arquitecturas modernas, integración continua, despliegue continuo y aplicando la metodología Lean Startup?

1.2.3 Justificación del problema

La necesidad de abordar la gestión manual de los almuerzos dentro de la empresa Links surge como una respuesta para mejorar la eficiencia operativa. Esta propuesta es interesante porque le permite a la organización adaptarse rápidamente a nuevos requerimientos a futuro. La introducción de un prototipo automatizado ayudará a manejar los recursos de manera más eficiente y a disponer de datos precisos que respalden las decisiones institucionales, al mismo tiempo que mejora la experiencia diaria de los empleados. Asimismo, incorporar una arquitectura tecnológica moderna junto con prácticas contemporáneas favorecerá que la empresa mantenga un ritmo de actualización sostenido y pueda desenvolverse adecuadamente en un escenario empresarial que cambia de forma permanente.

Se podrá obtener una justificación más detallada en el enunciado 1.3, donde también se habla de la problemática que se está planteando.

1.2.4 Delimitación del problema

Desarrollar un sistema que ayude a la empresa a registrar el menú de los almuerzos que se da por semanas, además de permitir registrar los empleados con cierta cantidad de datos personales requeridos, para poder llevar el control de los empleados que sí están almorzando y poder hacer la centralización de los almuerzos que fueron despachados para la empresa en el mes, así obtener un registro que ayude al personal encargado de esta gestión a la hora de contabilizar para hacer el pago al proveedor. Asimismo, se asegurará que el responsable registre con precisión el plato entregado a cada empleado según el menú programado para ese día. El sistema brindará al usuario la opción de evaluar diariamente o de semanal el menú recibido, además de registrar observaciones de mejora cuando lo considere

necesario. Obtener un archivo pdf que contenga información de los días que el empleado almorzó en la empresa.

El sistema no podrá hacer debido al alcance y el tiempo para desarrollar esta propuesta es: notificar por correo a cada usuario el menú de la semana, confirmar por correo a cada usuario el menú escogido en el día. No se podrá obtener la información del menú que más fue pedido y el que menos fue pedido.

1.3 Justificación del problema

La empresa Links atraviesa problemas relevantes debido a que el control de los almuerzos a sus empleados continúa realizándose de manera manual. Esta práctica provoca inconsistencias en los registros, ineficiencias en la administración de recursos y poca claridad en la información gestionada. Como consecuencia, se genera insatisfacción entre los empleados y se limita la capacidad institucional para fundamentar decisiones con datos precisos, lo que hace indispensable contar con una solución tecnológica que sistematice la gestión, seguimiento y verificación de los almuerzos suministrados.

En este marco, aplicar prácticas de (CI/CD) se convierte en un componente esencial para garantizar un desarrollo moderno, eficiente y confiable. Estas prácticas permiten automatizar tareas claves como la compilación, pruebas, análisis de calidad y despliegue, reduciendo errores y agilizando las entregas del software. Además, el desarrollo de un prototipo posibilita validar tempranamente las funcionalidades esenciales antes de invertir en una implementación completa, disminuyendo riesgos y asegurando que la solución final responda de manera precisa a los requerimientos auténticos de la institución.

La integración de CI/CD se relaciona estrechamente con enfoques DevOps, los cuales, según (Bijwe & Shankar, 2022), permiten que los sistemas en operación puedan actualizarse de forma frecuente y confiable. Desde esta perspectiva, DevOps se entiende como una cultura que integra equipos multifuncionales para

construir, probar y liberar software de manera rápida y automatizada (Macarthy & Bass, 2020), aumentando significativamente la eficiencia del desarrollo.

Este planteamiento refleja las transformaciones actuales en la ingeniería de software, donde la demanda creciente de nuevas funcionalidades ha impulsado la adopción de prácticas que automatizan el ciclo de construcción, validación y liberación del software, junto con la incorporación de metodologías ágiles y enfoques DevOps (Donca et al., 2022). En entornos donde se manejan múltiples fragmentos de código o despliegues continuos, estas herramientas permiten que los servicios (incluidos los basados en la nube) se adapten con rapidez a los cambios del mercado, manteniendo sistemas actualizados y reduciendo costos asociados a procesos manuales (Saboor et al., 2022).

La integración continua desempeña un papel esencial dentro del ciclo de desarrollo, ya que cada modificación realizada en el repositorio se somete automáticamente a compilación y pruebas, lo que permite detectar errores de forma temprana y mantener la estabilidad del proyecto. Como señalan (Gallaba & McIntosh, 2020), estas actividades requieren disciplina y mantenimiento constantes, aunque en proyectos de menor escala su impacto suele ser mínimo y ofrece beneficios significativos en términos de eficiencia y fiabilidad del software.

Por su parte, la metodología Lean Startup complementa estas prácticas al promover ciclos cortos de validación, aprendizaje continuo y mejora rápida. Este enfoque impulsa el desarrollo progresivo del producto mediante la retroalimentación constante de los usuarios, lo que permite prevenir riesgos, dividir tareas complejas en unidades manejables y validar hipótesis antes de invertir recursos en etapas avanzadas. Además, los productos mínimos viables (MVP), como maquetas o páginas de destino con funciones básicas, facilitan una validación temprana y efectiva, permitiendo obtener comentarios inmediatos de los primeros adoptantes y ajustar la solución de manera ágil (Gumilang et al., 2020; Matturro et al., 2021).

En síntesis, la unión entre una arquitectura tecnológica moderna, la aplicación de prácticas DevOps con procesos de integración y entrega continua, y el enfoque Lean Startup constituye una base firme para sustentar el desarrollo del prototipo propuesto para la empresa Links. Estos componentes fortalecen la exactitud en el control de los almuerzos y favorecen un funcionamiento más dinámico y adaptable, capaz de ajustarse eficazmente a las transformaciones y requerimientos funcionales del sistema empresarial.

1.4 Objetivos

1.4.1 Objetivo general

Implementar un prototipo de sistema innovador que utilice arquitectura moderna y prácticas de integración continua y despliegue continuo (CI/CD), con el fin de optimizar el proceso de registro y control de almuerzos para los empleados de la empresa Links, mediante la aplicación de la metodología Lean Startup

1.4.2 Objetivos específicos

- Definir una arquitectura moderna y escalable, seleccionando tecnologías innovadoras que se adapten a las necesidades específicas del sistema y garantizando seguridad y eficiencia mediante prácticas de desarrollo.
- Mostrar un flujo de Integración y Despliegue Continuo (CI/CD), configurando un pipeline que automatice las pruebas, la integración y despliegue, e implementando pruebas automatizadas para asegurar la calidad del software en cada iteración.
- Establecer la metodología Lean Startup para la mejora continua, definiendo métricas clave para evaluar la eficiencia en el registro y control de almuerzos, implementando ciclos de desarrollo cortos, y colaborando estrechamente con usuarios y el equipo operativo para adaptar ágilmente el sistema a las necesidades de la empresa Links.

2. Marco teórico referencial

2.1 Estado del arte

El estudio (Manzoor & Hasan, 2021) examina la usabilidad , los aspectos de interacción y presentación utilizados en las aplicaciones móviles del sector de alimentos. Los autores señalan que una interfaz intuitiva facilita los procesos de registros y entrega de alimentos. Este enfoque es relevante para el sistema propuesto en Links, donde la interacción del personal a cargo es vital para prevenir errores y hacer más fluida la operación diaria.

Por otro lado, (Belkhiria et al., 2022) exploran el uso de big data para conocer el entorno alimentario, mostrando que el análisis de datos puede revelar tendencias relevantes y contribuir a seleccionar mejores alternativas estratégicas. Esta necesidad se alinea con la necesidad institucional de Links de tener información exacta y consolidada para administrar los almuerzos eficientemente y con transparencia.

(Yang et al., 2022) proponen factores fundamentales para la administración de restaurantes saludables: la calidad del servicio, la seguridad alimentaria y llevar registros confiables. Estos resultados benefician al actual proyecto, ya que la empresa pretende mejorar la planificación de menús y recibir retroalimentación de sus trabajadores a través de calificaciones y comentarios.

En cuanto al diseño arquitectónico, (Itu, 2019) propone una arquitectura orientada a servicios que reduce el acoplamiento entre módulos y facilita la escalabilidad. Este tipo de diseño permite mantener sistemas flexibles, capaces de adaptarse a nuevos requerimientos sin afectar al resto de componentes. Esta visión técnica respalda la estructura modular implementada para el prototipo desarrollado en Links.

(Adam et al., 2019) analizan los beneficios de las API REST en sistemas empresariales, destacando su capacidad para intercambiar información mediante

operaciones HTTP estandarizadas. Este enfoque mejora la interoperabilidad entre servicios y simplifica la integración con otras plataformas. Dichos principios fundamentan el diseño del backend del sistema, que utiliza servicios REST para gestionar menús, registros y calificaciones.

Asimismo, (Yusof et al., 2022) muestran que la arquitectura REST facilita la integración de datos desde diversas fuentes, favoreciendo sistemas más mantenibles y escalables. Este aporte respalda la decisión de emplear esta arquitectura en el prototipo, asegurando que pueda ampliarse o conectarse a futuros módulos institucionales sin incurrir en reestructuraciones complejas.

Finalmente, (Martyn Coupland, 2021) destaca la relevancia de incorporar prácticas DevOps para fortalecer la calidad del software mediante la automatización de actividades como el armado del software, su verificación y su puesta en producción. Además, los procesos continuos de integración y entrega permiten identificar fallas con anticipación y mantener versiones estables del sistema. Este enfoque coincide con la estrategia adoptada en Links mediante el uso de Jenkins, GitHub y SonarQube para garantizar un desarrollo confiable y sostenible,

En síntesis, la literatura revisada muestra que la experiencia de usuario, la gestión adecuada de datos, las arquitecturas de servicios, las API REST y las prácticas DevOps constituyen pilares fundamentales para el diseño de sistemas modernos. Estos aportes sustentan conceptualmente el prototipo propuesto para Links, garantizando que se ajuste a las demandas vigentes y promueva la optimización constante del proceso institucional de almuerzos.

2.2 Definiciones previas

2.4.1Arquitectura de software

(Ángel Arias, 2016) define la arquitectura de software como el diseño estructural y organizativo que posee un determinado sistema de software. Proporciona un marco conceptual y técnico que guía el desarrollo del software, permitiendo que este sea eficiente, mantenible y que cumpla las necesidades planteadas por quienes utilizan el sistema. Este enfoque arquitectónico aborda cuestiones, entre ellas la división de tareas, asignación de responsabilidades, gestión de datos, comunicación entre los componentes y la escalabilidad del sistema.

2.4.2CI/CD

Según (Joel Lord, 2021) CI – Integración Continua (Continuous Integration) hace referencia a un proceso propio del desarrollo de software en el que los cambios realizados al código del sistema se incorporan automáticamente en un repositorio compartido varias veces al día. Ayudando a que se puede identificar y resolver los problemas de integración de manera temprana.

Despliegue continuo – CD (Continuos Deployment) establece que el software se entrega de una forma automatizada a un entorno de pruebas el cual está listo para ser desplegado después de realizar las pruebas.

Estas prácticas son de fundamental importancia en el desarrollo de software porque garantizan rapidez, calidad y eficiencia en cada entrega. Además, contribuyen a mantener versiones estables del sistema mediante procesos automáticos y confiables, lo que favorece un ciclo de desarrollo continuo y bien controlado.

2.4.3Patrones de diseño

Según (Oscar J. Iturralde, 2016), los patrones de diseño representan propuestas ya probadas que sirven como guía para enfrentar problemas frecuentes en el desarrollo de software, promoviendo código mantenible y flexible. Su aplicación

facilita la organización interna del sistema y reduce el acoplamiento entre componentes.

En el presente proyecto se utilizaron patrones creacionales, estructurales, de comportamiento y arquitectónicos, tanto en el backend desarrollado con Laravel 12 como en el frontend implementado con Angular 19. Estos patrones fueron seleccionados según la necesidad técnica de cada módulo y favorecieron la escalabilidad del prototipo.

Tabla 1
Patrones de diseño aplicados en el sistema

Categoría	Patrón	Validación realizada	Justificación
Creacional	Inyección de dependencias	Backend (Servicios, Repositorios) y Frontend (Servicios de Angular)	Permite crear objetos de forma controlada, reduciendo el acoplamiento y facilitando la reutilización y pruebas.
Estructural	Service–Repository	Backend (Laravel 12)	Separa la lógica de negocio del acceso a datos. Facilita mantenimiento, pruebas, reemplazo de repositorios y escalabilidad.
Estructural	Componentes reutilizables	Frontend (Angular 19)	Organización modular del UI: tablas, formularios, tarjetas. Mejora mantenibilidad y uniformidad visual.

Comportamiento	Observer (Observables RxJS)	Frontend (Angular)	Sincroniza datos entre componentes en tiempo real (menús, calificaciones, sesiones), habilitando un flujo reactivo.
Comportamiento	Strategy (Guards y reglas de acceso)	Frontend (Angular 19– CanActivate)	Implementa estrategias de acceso por rol; define comportamientos distintos según permisos del usuario.
Arquitectónico	MVC (Modelo–Vista–Controlador)	Backend (Laravel 12)	Organiza el sistema en modelos, controladores y vistas (para PDFs). Reduce acoplamiento y facilita la lógica.
Arquitectónico	CAPA Módulos y Servicios	Angular 19	Favorece la separación de responsabilidades entre features, core y shared, permitiendo escalabilidad.

Nota. Elaboración propia a partir de la arquitectura implementada en Laravel 12 y Angular 19.

La adopción de estos patrones permitió estructurar un sistema estable y escalable, reduciendo el acoplamiento entre módulos y facilitando la integración con el flujo CI/CD. Además, su uso fortaleció la claridad del código y la evolución futura del prototipo.

2.4.4API REST

Según (Mark Masse, 2011), una API REST es un medio de interacción que permite a distintos sistemas intercambiar información entre diferentes sistemas bajo los

lineamientos de la arquitectura REST. En este modelo, los elementos del sistema se identifican con direcciones únicas y se gestionan utilizando acciones estándar del protocolo HTTP. Este enfoque establece lineamientos que guían el diseño y funcionamiento de los servicios web, favoreciendo la simplicidad, la interoperabilidad entre plataformas y con el potencial de ampliar la transferencia de datos de manera eficiente.

2.4.5 DevOps

Como plantean (Kumar et al., 2024), DevOps se entiende como un enfoque orientado a integrar de manera continua las actividades de desarrollo con las labores operativas del software. Este enfoque se fundamenta en ciclos que abarcan planificación, pruebas, entrega y supervisión constante, lo que facilita identificar fallas con anticipación, disminuir los periodos de liberación y fortalecer la coordinación entre los equipos técnicos. Asimismo, la incorporación de herramientas de automatización, como Jenkins o GitHub, contribuye a mantener un ritmo de trabajo estable y bien estructurado durante todo el proceso de desarrollo.

En relación con el prototipo desarrollado en este trabajo, DevOps cumple un rol esencial para asegurar que el sistema permanezca alineado con las necesidades de la empresa Links. La integración y entrega continua —implementada mediante Jenkins y SonarQube— hace posible automatizar la construcción del software, verificar su calidad y desplegar nuevas versiones sin intervenciones manuales. Gracias a este modelo, la solución evoluciona con mayor rapidez, reduce riesgos operativos y queda preparada para responder a futuros requerimientos o incorporaciones dentro de la organización.

2.4.6 Metodología Lean Startup

Lean Startup plantea un modelo ágil para el desarrollo iterativo de productos, basado en comprobar hipótesis desde etapas tempranas y obtener retroalimentación permanente de quienes interactúan con la solución. Según (Kumar et al., 2024), la creación y evaluación progresiva de prototipos permite identificar oportunidades de mejora desde etapas iniciales, reduciendo riesgos y

acelerando la toma de decisiones basada en evidencia real. Este enfoque prioriza ciclos de experimentación rápida, lo que resulta especialmente útil en contextos donde los requisitos evolucionan constantemente.

Este proyecto aplicó Lean Startup para orientar la construcción iterativa del sistema de registro de almuerzos, guiando cada entrega mediante el ciclo construir–medir–aprender. La validación continua con responsables de cocina, empleados y administradores permitió ajustar funcionalidades clave, mejorar la usabilidad y asegurar que el prototipo responda a necesidades reales de la institución. Este proceso facilitó mejoras tempranas y sostenidas, fortaleciendo la calidad, escalabilidad y pertinencia de la solución final implementada.

3. Materiales y metodología

El presente capítulo detalla los métodos, técnicas y recursos utilizados durante la construcción del sistema de registro de almuerzos. Se explica cómo se incorporó un enfoque basado en Lean Startup para realizar iteraciones constantes del prototipo, complementado con prácticas DevOps que facilitaron los procesos de integración y despliegue del software. Asimismo, se describen las herramientas empleadas, las fases ejecutadas y los procedimientos aplicados, orientados a asegurar que la solución desarrollada mantuviera niveles adecuados de calidad, escalabilidad y sostenibilidad.

3.1 Marco Metodológico

3.1.1 Tipo de investigación

El presente proyecto se enmarca en un estudio de carácter aplicado, orientado al desarrollo y validación de un prototipo funcional que responda a un problema real dentro de la empresa Links. Este enfoque permitió analizar el proceso institucional de registro de almuerzos, identificar sus limitaciones operativas y proponer una solución tecnológica verificable en un entorno real. De acuerdo con (Peralta & Pyka, 2025) , la ingeniería de software requiere procesos metodológicos ordenados que garanticen la construcción de sistemas confiables y sostenibles en el tiempo, lineamiento que guía el carácter aplicado del presente trabajo.

3.1.2 Enfoque de investigación (mixto con predominio cualitativo)

Se adoptó un enfoque mixto con predominio cualitativo, combinando la interpretación de percepciones, experiencias y observaciones de los usuarios con métricas derivadas del funcionamiento técnico del prototipo. El componente cualitativo permitió comprender cómo administradores, responsables de cocina y empleados interactúan con el sistema, aportando información esencial para ajustar la usabilidad y la lógica de negocio. De manera complementaria, se analizaron datos

cuantitativos básicos —como tiempos de registro, reducción de errores y métricas del pipeline CI/CD— que fortalecieron la validación técnica del sistema. Esta integración permitió evaluar la solución desde una perspectiva tanto funcional como experiencial.

3.1.3 Metodología aplicada: Lean Startup

La construcción del prototipo se guio mediante la metodología Lean Startup, basada en el ciclo iterativo construir–medir–aprender**.** Cada iteración permitió desarrollar funcionalidades incrementales (MVP parciales), validarlas con usuarios reales y aplicar mejoras tempranas según la retroalimentación obtenida. Este proceso favoreció un avance continuo, redujo riesgos de diseño y aseguró que las características del sistema se ajustaran a necesidades verificadas en campo. Como señalan (Peralta & Pyka, 2025), los enfoques iterativos permiten diseñar soluciones sostenibles y alineadas con la evolución del problema y los usuarios.

3.1.4 Prácticas complementarias: DevOps e integración continua (CI/CD)

De forma complementaria, se integraron prácticas DevOps orientadas a automatizar etapas clave, entre ellas la construcción, pruebas, análisis de código y despliegue continuo. Para ello se configuró un entorno basado en Jenkins, GitHub y SonarQube, lo que permitió garantizar trazabilidad de versiones, verificación constante del código y despliegues controlados en el servidor VPS. Estas prácticas fortalecieron la confiabilidad del prototipo y aseguraron una evolución ordenada del sistema durante su desarrollo, en concordancia con los lineamientos modernos de ingeniería de software.

3.1.5 Trazabilidad metodológica entre objetivos e iteraciones

Para asegurar la coherencia entre los objetivos específicos (qué se busca lograr) y las iteraciones Lean Startup (cómo se lograron), se estableció una trazabilidad explícita presentada al final de la sección 3.2. Esta relación muestra cómo cada objetivo se materializó mediante incrementos funcionales, validaciones con

usuarios internos, evidencia técnica generada en el pipeline CI/CD y ajustes derivados del ciclo construir–medir–aprender. Esta trazabilidad garantiza que el proceso metodológico, el enfoque mixto y los resultados obtenidos mantienen una alineación completa a lo largo del proyecto.

3.2 Proceso Metodológico

La metodología Lean Startup fue utilizada para estructurar el desarrollo del prototipo, usando el ciclo iterativo de construir–medir–aprender para validar progresivamente las funcionalidades del sistema con los usuarios internos. Cada iteración permitió desarrollar un MVP parcial, medir su desempeño técnico y operativo, y aplicar ajustes derivados de la retroalimentación institucional. Este proceso garantizó una evolución continua del sistema y la coherencia con las necesidades reales de la empresa.

El enfoque mixto definido en la sección 3.1 se aplicó combinando datos cualitativos, retroalimentación de administradores, responsables de cocina y empleados, con datos cuantitativos obtenidos del funcionamiento del prototipo. Estos incluyeron tiempos de registro, frecuencia de uso por módulo, métricas del pipeline CI/CD y registros emitidos en la base de datos. La integración de ambos tipos de datos permitió evaluar cada iteración desde perspectivas técnicas y experienciales, fortaleciendo la validez del proceso.

Tabla 2

Trazabilidad entre objetivos específicos e iteraciones Lean Startup

Objetivo específico (OE)	Iteración donde se cumplió	Implementación realizada	Validación aplicada
OE1. Definir una arquitectura moderna y escalable.	I01 – I03	Desarrollo de módulos base, seguridad JWT, estructura multicapa y mantenimientos institucionales.	Validación del administrador y pruebas funcionales.

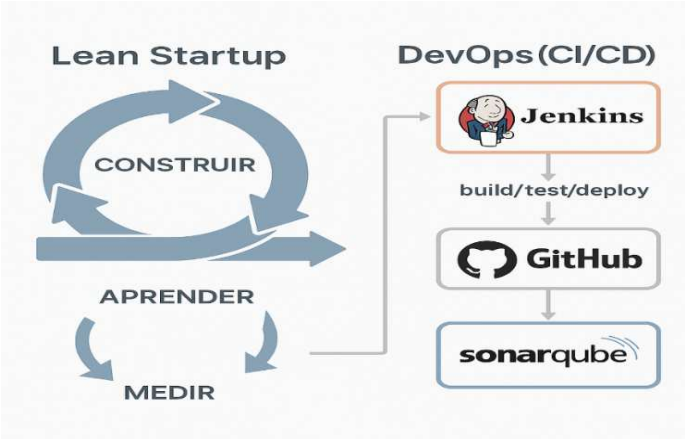
OE2. Implementar un flujo CI/CD con prácticas DevOps.	I05	Pipeline automatizado con Jenkins, análisis SonarQube y despliegue controlado en VPS.	Evidencia de quality gates, ejecución del pipeline y despliegue exitoso.
OE3. Aplicar Lean Startup mediante ciclos cortos.	I01 – I05	Construcción incremental de MVPs, ajustes continuos y retroalimentación directa.	Validación con empleados, responsable de cocina y administrador.

Nota. La tabla presenta la trazabilidad entre los objetivos específicos y las iteraciones Lean Startup, demostrando cómo cada objetivo se alcanzó mediante el ciclo construir–medir–aprender.

La siguiente ilustración complementa esta trazabilidad metodológica mostrando cómo se articulan las iteraciones con las prácticas DevOps dentro del ciclo construir–medir–aprender.

Ilustración 1

Proceso metodológico del sistema de registro de almuerzos



Nota. El diagrama muestra la integración del ciclo Lean Startup con la prácticas DevOps de integración y despliegue continuo (CI/CD)

La automatización de los procedimientos se realizó a través de un flujo CI/CD que fue implementado con Jenkins, conectado al repositorio GitHub y combinado con SonarQube para llevar a cabo el análisis estático del código. Esto permitió garantizar

trazabilidad, calidad y estabilidad en cada despliegue. Según (Chen et al., 2025), la automatización y el aprendizaje continuo permiten optimizar el código y acelerar la entrega de valor en entornos productivos, reforzando la efectividad de este enfoque.

El desarrollo completo se organizó en cinco iteraciones y cada una incluyó construcción, validación y ajustes derivados de la retroalimentación. Este procedimiento favoreció un avance progresivo y la reducción de errores operativos, garantizando estabilidad durante la evolución del sistema. En este sentido, aplicar procesos metodológicos estructurados facilitó mantener la calidad del software y un control claro de su trazabilidad a lo largo del proyecto.

Tabla 3

Iteraciones del proceso metodológico del sistema de registro de almuerzos

Iteración	Objetivo principal	Validación realizada
01	Registro y mantenimiento de empleados, área, departamentos, usuarios, roles y permisos. Implementación del sistema de autenticación y control de acceso.	Aprobación del administrador institucional.
02	Creación y gestión de menús semanales y registro diario de almuerzos.	Validación del responsable de cocina.
03	Calificaciones de almuerzo por los empleados y visualizaciones de observaciones	Retroalimentación directa de los usuarios.
04	Generación de reporte PDF mediante la integración de DomPdf.	Validación funcional en entorno pruebas y producción.
05	Automatización de despliegue con Jenkins y control de calidad con SonarQube	Validación técnica en entorno controlado.

Nota. Cada iteración fue seguida de sesiones de validación y ajustes del sistema.

En conjunto, el proceso metodológico integró los principios de la ingeniería de software aplicada, la filosofía Lean Startup y las prácticas DevOps para la automatización continua. Esta combinación metodológica permitió obtener un prototipo estable, validado de forma incremental y alineado con los objetivos específicos de la investigación. Tal como afirman (Pastor et al., 2025), los enfoques basados en modelos conceptuales favorecen la coherencia, modularidad y evolución progresiva de los sistemas en entornos iterativos.

3.3 Herramientas y tecnologías

Para garantizar la calidad, trazabilidad y sostenibilidad del software a lo largo de su ciclo de vida, la creación del prototipo del sistema para registrar almuerzos se basó en un conjunto de herramientas, tecnológicas y prácticas modernas que garantizaron la calidad, trazabilidad y sostenibilidad del software durante su ciclo de vida.

La siguiente tabla sintetiza las principales herramientas, su objetivo y las pruebas de uso que apoyan el proceso de creación e implementación del sistema.

Tabla 4

Herramientas y tecnologías empleadas en el sistema de registro de almuerzos

Categoría	Herramienta / Tecnología	Propósito principal	Evidencia / Uso
Metodologías y prácticas	Lean Startup	Validar iterativamente el prototipo bajo el ciclo construir-medir-aprender.	Planificación de iteraciones y MVPs con validación directa de usuarios.
	DevOps / CI-CD	Automatizar los procedimientos de despliegue continuo e integración con quality gates	Pipeline en Jenkins para el despliegue.

Gestión y control de versiones	GitHub	Control de versiones, ramas (Gitflow), issues, pull request y tags de releases.	Repositorio del proyecto con historial de commits y release (front y back)
Integración y calidad	Jenkins 2.528.1	Automatización del flujo del checkout -> build ->análisis estático -> deploy.	Pipeline configurado en servidor Vps.
	SonarQube Community Edition v10.6	Análisis de código: cobertura, vulnerabilidades, duplicaciones y code smells.	Quality gate aprobado previo al despliegue.
Frontend	Angular 19 (SPA)	Interfaz dinámica con PrimeNG, formularios reactivos y guards por roles	Componentes funcionales en el sistema
	PrimeNG 19	Biblioteca de componentes visuales reutilizables	Tablas, modales y formularios adaptados a cada módulo.
Backend	Laravel 12 (PHP 8.3)	Lógica de negocio, API REST, validaciones y manejo de servicios/repositorio.	Endpoint documentados y controladores implementados.
	DomPDF 3.1	Generación de reportes PDF (reporte consolidados de almuerzos).	Evidencia: exporta PDF desde la opción de reportes.

Base de datos	Microsoft SQL Server2022	Almacenamiento estructurado y normalizado de datos.	Esquemas y tablas en producción.
Infraestructura y despliegue	Ubuntu Server 22.04(VPS en la nube)	Entorno de ejecución del backend y front en un servidor remoto.	VPS alojado en OVH Cloud con dominio.
	Nginx + PHP-FPM	Servidor web y monitor de ejecución de PHP.	Configuración de hosts virtuales y rutas /var/www/….
	Certbot (SSL)	Seguridad y cifrado HTTPS de los subdominios	Certificados activos en cada servicio.

Nota. Elaboración a partir de la configuración del entorno de desarrollo y producción.

Para el proceso CI/CD se seleccionó Jenkins debido a su facilidad de integración con servidores VPS, su compatibilidad con proyectos Laravel y Angular, y su capacidad para ejecutar pipelines completamente personalizados sin depender de servicios externos. A diferencia de GitHub Actions o GitLab CI, Jenkins permite un control total del entorno, mayor flexibilidad en el uso de plugins y una autonomía completa dentro de la infraestructura propia de la empresa.

La combinación de Laravel, Angular, y una infraestructura basada en Ubuntu VPS con herramientas DevOps como Jenkins y SonarQube, garantizó una entrega continua, escalable y alineada con las buenas prácticas de ingeniería de software. Asegurando la integridad de los datos y la estabilidad de las versiones liberadas.

3.4 Unidad de análisis

Los trabajadores y responsables internos de la empresa Links, quienes intervienen directamente en el proceso de registro, entrega y calificación de almuerzos. La población total asciende a 44 empleados, y para la validación del prototipo se seleccionó una muestra intencional de 18 participantes, considerando su relación

directa con el proceso alimentario y su disponibilidad para realizar pruebas en el entorno piloto.

Se incluyeron colaboradores vinculados de forma directa al proceso de almuerzos, ya sea en actividades de registro, control, preparación o consumo. También participaron quienes utilizan los módulos del sistema según su rol, como administradores, responsables de cocina, empleados o supervisores. Asimismo, solo se consideró a quienes contaban con disponibilidad para asistir a las sesiones de evaluación dentro del periodo definido, asegurando la participación de usuarios relacionados con el flujo operativo.

Se excluyó al personal que no interactúa con el proceso alimentario o cuya carga laboral impedía participar en la validación. Esto aplicó principalmente a áreas como contabilidad, gerencias y unidades estratégicas con alta demanda operativa. También se excluyó a colaboradores cuya agenda institucional no permitía asistir a las pruebas, garantizando que la validación se realizara únicamente con usuarios directamente involucrados en el sistema.

La muestra se distribuyó entre los perfiles operativos clave, lo que permitió obtener retroalimentación desde distintos niveles de responsabilidad y asegurar una evaluación integral del funcionamiento técnico del prototipo. Esta composición garantizó una representación equilibrada entre administración, operación de cocina, empleados y supervisión externa.

Tabla 5
Roles participantes y alcance de validación del sistema

Rol / Actor	Descripción de funciones en el sistema	Módulos utilizados	Participación en validación	N° de usuarios implicados
Administrador	Supervisa la gestión institucional, controla	Áreas, Departamentos, Usuarios, Roles, Reportes.	Aprobación de estructura funcional.	2

	catálogos, usuarios y permisos.			
Responsable de cocina	Registra los almuerzos diarios, crear menos semanales, revisa calificaciones.	Platos, Menús del día, Registro de almuerzos, Calificaciones	Validación funcional y flujo de registro diario	3
Empleado	Usuario final que consulta menús y califica los almuerzos recibidos.	Mis almuerzos, Calificar almuerzo	Prueba de usabilidad y retroalimentación sobre experiencia de usuario.	10
Proveedor / Supervisor	Accede al reporte consolidado por rango de fecha para su descarga pdf.	Reporte Proveedor	Validación de formato y consistencia de reportes exportados.	1
Equipo técnico	Desarrolla, despliega y monitoreo el sistema (Devops, CI/CD, SonarQube)	Jenkins, GitHub, SonarQube, Vps Server	Validación técnica y despliegue controlado.	2

Nota. Elaborado a partir de las sesiones de validación y pruebas funcionales del sistema en el entorno piloto.

Durante las sesiones de validación, los participantes ejecutaron tareas como registrar áreas y empleados, crear menús semanales, asignar almuerzos diarios, calificar el servicio recibido y verificar los reportes PDF generados. La validación

funcional, las pruebas de aceptación del usuario, la observación del flujo operativo y el análisis de los resultados generados por el sistema durante su ejecución en el ambiente de prueba se utilizaron para evaluar la conformidad.

En términos éticos, no se manejaron datos sensibles ni información que comprometa la privacidad de los participantes; únicamente se utilizaron registros operativos e institucionales. Previo a la validación, se informó a los usuarios sobre el objetivo académico del proyecto y se obtuvo su consentimiento verbal, garantizando confidencialidad, respeto y el cumplimiento de buenas prácticas de investigación aplicada.

3.5 Técnicas e instrumentos de recolección datos (enfoque por requerimientos)

Los datos se recogieron a través de la observación directa y pruebas de usabilidad, donde los usuarios ejecutaron tareas reales para validar los requerimientos funcionales del sistema. Se aplicó la metodología Lean Startup para obtener información progresiva en cada ciclo construir–medir–aprender. Este proceso permitió identificar fallos, registrar incidencias y ajustar la lógica según el comportamiento observado en cada módulo.

Los instrumentos utilizados incluyeron listas de verificación de requerimientos, registros de pruebas, métricas de uso y evidencias técnicas generadas por Jenkins y SonarQube. Un requerimiento se consideró cumplido cuando el usuario completaba la acción sin errores o cuando la respuesta coincidía con lo especificado. La validación por roles —administrador, responsable de cocina, empleado y supervisor— aseguró la pertinencia operativa de cada incremento funcional.

La verificación del cumplimiento también se apoyó en métricas del pipeline CI/CD, como compilaciones exitosas, aprobación de filtros de calidad y ausencia de vulnerabilidades críticas. La retroalimentación obtenida en las sesiones de prueba

permitió ajustar la experiencia de uso y mejorar la coherencia del flujo. Esta combinación garantizó una evaluación adecuada desde la perspectiva técnica y operativa.

Las tablas de esta sección sintetizan los incrementos validados, mostrando requerimientos, entregables y evidencia recopilada en cada iteración. Su estructura permite visualizar la trazabilidad del sistema desde la autenticación hasta el despliegue continuo. Con este enfoque modular, las tablas funcionan como instrumentos que documentan la evolución del prototipo y la satisfacción de los requerimientos definidos.

Tabla 6
Resumen de iteraciones (visión general)

Iteración	Objetivo del incremento	Entregables esperados
01	Acceso y control de usuarios.	Autenticación JWT, gestión de roles, permisos y menú dinámico Angular.
02	Maestros institucionales.	Mantenimientos de empleados, áreas y departamentos. Con validaciones.
03	Operación diaria.	Gestión de menús, registro diario de almuerzos y control de integridad.
04	Valor al usuario.	Calificaciones, observaciones y reporte PDF.
05	Entorno y calidad.	Flujo CI/CD, análisis SonarQube, despliegue Nginx.

Nota. Elaboración a partir de la estructura metodológica Lean Startup. Cada iteración representa un incremento funcional del sistema.

Tabla 7
Iteración 01 — Acceso y control

ID	Requerimiento	Entregable esperado
RF-1.1	Login con credenciales seguras.	Token JWT emitido desde backend (Laravel 12) y expiración controlada.

RF-1.2	Logout de sesión.	Revocación del token con endpoints dedicados.
RF-1.3	Roles y permisos.	Tablas roles, permissions, role_has_permissions con asignación por usuario.
RF-1.4	Guardas y rutas protegidas	CanActivate por rol; redirección automática según privilegio.
RF-1.5	Menú lateral dinámico	Carga condicional desde claims del usuario; visibilidad según permisos.
RF-1.6	Registro de auditoría	Tabla de inicio/cierre de sesión con IP, fecha y usuario.

Nota. Iteración enfocada en establecer la base de seguridad y control de acceso al sistema, asegurando la autenticación y autorización de los diferentes tipos de usuarios.

Tabla 8
Iteración 02 — Maestros institucionales

ID	Requerimiento	Entregable esperado
RF-2.1	Módulo de empleados.	CRUD completo con validaciones, búsqueda y paginación.
RF-2.2	Módulo de áreas.	Jerarquía de áreas con relación a departamentos.
RF-2.3	Módulo de departamentos.	Asociación con áreas y validación referencial.
RF-2.4	Validación de datos.	Reglas de negocio en FormRequest y validaciones en formularios Angular.
RF-2.5	Toast UI.	Notificaciones con GlobalToastService y control de estados.

RF-2.6	Seeders base.	Población inicial de catálogos, roles y permisos predeterminados.

Nota. Esta iteración establece los módulos institucionales que conforman las tablas maestras del sistema, asegurando la correcta organización y coherencia de los datos base.

Tabla 9
Iteración 03 — Operación diaria (menús y registro)

ID	Requerimiento	Entregable esperado
RF-3.1	Catálogo de tipos de platos.	Seeder en base tabla catálogos.
RF-3.2	Menú del día/semana.	Creación de menús semanales vinculando platos con fecha.
RF-3.3	Crear menús por cada día.	Permitir crear varios menús por cada día.
RF-3.4	Registro de almuerzos.	Asignación diaria de almuerzos por empleado.
RF-3.5	Consulta operativa menús.	Tabla dinámica filtrable rango fechas, y búsqueda general.
RF-3.6	Integridad de transacciones.	Manejo de operaciones en bloque y control de errores. Evitar duplicados.

Nota. Iteración orientada a la automatización de los procesos operativos diarios, estableciendo la lógica de negocio principal del sistema de almuerzos.

Tabla 10
Iteración 04 — Valor al usuario (calificaciones y reporte)

ID	Requerimiento	Entregable esperado
RF-4.1	Calificación de almuerzos.	Registro único de puntuación diaria (escala 1–5).
RF-4.2	Observaciones opcionales.	Comentarios de mejora almacenados por empleado.
RF-4.3	Reporte por proveedor	Consolidado de almuerzos por rango de fechas.

RF-4.4	Permisos de acceso a reporte	Restricción de descargas según rol del usuario.

Nota. Esta etapa incorpora la interacción y la retroalimentación de los usuarios, lo que produce datos útiles para el análisis y la administración de la calidad del servicio.

Tabla 11
Iteración 05 — Iteración V — Entorno, despliegue y calidad

ID	Requerimiento	Entregable esperado
RF-5.1	Integración continua (CI).	Pipeline automático que ejecuta compilación y pruebas.
RF-5.2	Despliegue continuo (CD).	Copia y actualización de archivos en /var/www (VPS).
RF-5.3	Control de calidad.	Integración con SonarQube y quality gates.
RF-5.4	Certificación SSL.	Emisión de certificados con Certbot y redirección HTTPS.
RF-5.5	Variables de entorno.	Configuración de .env segura sin exponer credenciales.

Nota. Iteración centrada en la implementación de prácticas DevOps, asegurando la calidad, estabilidad y sostenibilidad del sistema en el entorno de producción.

Tabla 12
Reglas transversales (aplican a todas las iteraciones)

ID	Requerimiento	Entregable esperado
RT-1	Consistencia de interfaz.	Estilo uniforme, íconos, tipografía y componentes PrimeNG.
RT-2	Rendimiento básico.	Paginación y consultas indexadas en SQL Server.
RT-3	Seguridad general.	Token, Sanitización de datos, CORS.
RT-4	Trazabilidad de versiones.	Ramas feature, tags de release y changelog mínimo.

Nota. Los requerimientos transversales complementan todas las iteraciones, asegurando consistencia visual, seguridad y trazabilidad del producto completo.

3.6 Procedimiento de implementación

El procedimiento de implementación comprendió la preparación de la infraestructura, la configuración de los repositorios y la automatización del despliegue mediante un pipeline CI/CD, garantizando trazabilidad, reproducibilidad y estabilidad del entorno. Bajo las prácticas DevOps, el proceso permitió una entrega continua y controlada del sistema.

3.6.1 Preparación del entorno

Se utilizó un servidor VPS con Ubuntu Server 22.04 como entorno principal de despliegue.

Se configuraron los siguientes componentes:

- **Servidor web**: instalación de Nginx y PHP-FPM (versión 8.3) para gestionar las solicitudes HTTP.
- **Gestión de dominios**: configuración de los subdominios principales:
  - https://front.link-lunch.online (interfaz Angular).
  - https://api.link-lunch.online (API Laravel).
- **Certificados SSL:** generación y renovación automática mediante Certbot para habilitar el cifrado HTTPS.
- **Firewall y permisos**: habilitación de puertos 80/443, asignación de usuarios de servicio (jenkins, www-data) y permisos sobre /var/www/.

Nota. Esta fase estableció la base del entorno de ejecución y seguridad, asegurando la comunicación cifrada entre los componentes del sistema.

3.6.2 Configuración de repositorios

Los repositorios se alojaron en GitHub, separando el código fuente del backend (Laravel) y frontend (Angular).

Se aplicó la estrategia GitFlow para un control ordenado de versiones:

- Rama main: versión estable desplegada en producción.
- Rama develop: código en validación y pruebas.
- Ramas feature/: desarrollo de funcionalidades específicas (ej. feature-calificaciones, feature-reportes).

Se configuraron GitHub Secrets y variables de entorno (DB_HOST, APP_ENV, JWT_SECRET, CI_TOKEN, etc.) para integrarse con Jenkins y evitar exposición de credenciales.

Nota. El control de versiones permitió mantener la trazabilidad entre los commits, releases y despliegues automatizados.

3.6.3 Configuración de la base de datos

Para el almacenamiento y gestión de la información, el sistema utiliza Microsoft SQL Server como motor de base de datos principal. En este apartado se detallan las configuraciones implementadas para su correcto funcionamiento, las cuales incluyeron los siguientes ajustes:

- Creación del usuario sa con credenciales seguras y asignación de roles con privilegios mínimos.
- Ejecución de migraciones para generar las tablas base (empleados, almuerzos, menus, roles, users, etc.).
- Carga inicial mediante seeders, garantizando datos base (roles, áreas, tipos de platos, permisos).
- Programación de respaldos automáticos diarios con un script en /opt/backup_sql/ usando cron.

Nota. Esta etapa aseguró la persistencia y disponibilidad de la información, estableciendo políticas de respaldo y recuperación.

3.6.4 Implementación del pipeline CI/CD (Jenkins)

Se configuró un pipeline en Jenkins con el propósito de automatizar, de manera continua, el proceso de construcción y actualización del sistema. Esta automatización permitió coordinar cada una de las fases involucradas en la integración del código y su posterior publicación en el entorno de despliegue. El flujo fue definido dentro de un archivo Jenkinsfile, en el cual se estructuraron las etapas que conforman la ejecución programada. A continuación, se detallan las actividades contempladas dentro de dicho pipeline.

Tabla 13

Resumen del flujo CI/CD aplicado al sistema (backend, frontend y análisis de calidad)

Etapa del proceso	Acción realizada	Descripción / Propósito técnico
Backend (Laravel)	Checkout del código	Obtención del repositorio actualizado desde GitHub.
	Instalación de dependencias	composer install --no-dev para optimizar el entorno de producción.
	Seeders	php artisan db:seed para cargar datos base necesarios.
	Despliegue del backend	Copia del proyecto hacia /var/www/back-sistema-registro-almuerzo.
	Reinicio de servicios	Reinicio de PHP-FPM y revisión de logs para garantizar el funcionamiento.
Frontend (Angular)	Checkout del código	Obtención del código actualizado desde GitHub.
	Instalación de dependencias	npm ci para instalar paquetes de manera limpia.
	Compilación	npm run build para generar los archivos de producción.

	Despliegue del frontend	Copia de /dist/ a /var/www/front-sistema-registro-almuerzo.
	Validación visual	Pruebas rápidas del funcionamiento y conexión con la API.
Análisis de calidad (SonarQube)	Ejecución del análisis estático	Revisión automática de calidad del código antes del despliegue final.
	Métricas evaluadas	code smells, duplicaciones, vulnerabilidades y cobertura.

Nota. El flujo CI/CD permitió automatizar la construcción, validación, análisis y despliegue tanto del backend como del frontend, garantizando consistencia, seguridad y trazabilidad en cada entrega realizada al entorno VPS.

3.6.5 Despliegue controlado y verificación

El despliegue se ejecutó de forma controlada y secuencial en el siguiente orden:

Base de datos: migraciones, seeders y respaldo.

1. Backend: despliegue en producción y verificación de endpoints (/api/empleados, /api/almuerzos, /api/calificaciones).
2. Frontend: carga de la interfaz y revisión de componentes (menús, formularios, reportes).
3. Health checks: verificación de respuesta HTTP 200, pruebas de conexión con la API y validación del SSL activo.
4. Control de permisos: comprobación de roles (admin, responsable, empleado, proveedor).

Nota. El despliegue se realizó en ventanas de mantenimiento controladas, minimizando riesgos de caída de servicio y asegurando el correcto funcionamiento de las rutas críticas.

3.7 Limitaciones y dificultades

Durante el desarrollo e implementación del prototipo de sistema de registro de almuerzos, se presentaron diversos desafíos técnicos, metodológicos y operativos. Con el objetivo de asegurar la estabilidad, la continuidad y la calidad del producto final, estas restricciones se abordaron a través de tácticas de mitigación que se implementaron en cada etapa del proyecto.

Tabla 14
Limitaciones, dificultades y estrategias de mitigación

Aspecto	Descripción de la limitación o dificultad	Estrategia de mitigación aplicada	Nivel de efectividad
Acceso a usuarios y tiempos de validación	La participación de los usuarios internos en las pruebas de aceptación del usuario se vio reducida por la carga de sus actividades operativas.	Se implementaron pruebas durante el desarrollo es decir se extendió mucho más el tiempo para poder cubrir todos los escenarios.	Parcialmente efectiva (se logró la validación, aunque con tiempos extendidos).
Infraestructura del VPS (CPU/RAM/IO)	Debido a la capacidad limitada del servidor en la nube, el rendimiento se vio afectado de forma temporal al realizar los despliegues.	Las ejecuciones se realizaron únicamente tras la revisión y aprobación de los desarrolladores responsables antes de fusionar nuevas modificaciones,	Efectiva (se estabilizó el rendimiento del servidor).

		evitando despliegues directos y optimizando el uso de recursos.	
Datos reales para pruebas funcionales	Restricciones en el uso de información real de empleados por motivos de privacidad.	Se protegió la información mediante enmascaramiento de datos y se utilizaron conjuntos de prueba anonimizados en un entorno diseñado con registros simulados.	Efectiva (se protegieron datos y se mantuvo la funcionalidad).

Nota. Elaborado a partir de los registros técnicos del pipeline, bitácoras de Jenkins y reportes de integración del entorno de despliegue.

Dado que se adoptaron a tiempo medidas correctivas y preventivas, las restricciones detectadas no perjudicaron la realización de los propósitos del proyecto.

La adopción de prácticas DevOps, junto con la metodología Lean Startup, permitió mantener un flujo de trabajo iterativo y controlado, asegurando la continuidad del desarrollo y la entrega progresiva del sistema.

Estas experiencias aportaron un aprendizaje significativo para la futura evolución del sistema, fortaleciendo la resiliencia de su infraestructura y su modelo de despliegue continuo.

En conjunto, las limitaciones identificadas no afectaron los resultados finales del proyecto ni impidieron el cumplimiento de los objetivos planteados. Si bien algunas dificultades requirieron extender las validaciones o ajustar ciertos procesos técnicos, todas las funcionalidades previstas fueron desarrolladas, probadas y

verificadas correctamente. Las estrategias de mitigación aplicadas resultaron suficientes para mantener la continuidad del trabajo y garantizar la calidad del prototipo, cumpliendo con los requerimientos funcionales y técnicos establecidos

4. Resultados y discusión

Los resultados logrados en el desarrollo del sistema de registro de almuerzos para la compañía Links se exponen en este capítulo. Se detallan los módulos creados, la arquitectura implementada y la evidencia recopilada durante las pruebas realizadas con los usuarios internos. Además, se resumen los avances alcanzados mediante el uso de Lean Startup y las prácticas DevOps aplicadas en el proceso de integración y despliegue continuo.

Además, se analiza cómo estos resultados permiten verificar el logro de los objetivos establecidos en el estudio. Se muestra de qué manera la arquitectura propuesta (OE1), el pipeline CI/CD configurado (OE2) y las iteraciones del ciclo construir–medir–aprender (OE3) contribuyeron al desarrollo y validación del prototipo. Con ello se evalúa su impacto técnico y operativo dentro del proceso institucional.

4.1 Resultados del desarrollo del sistema

La metodología Lean Startup fue utilizada para desarrollar el sistema, lo que posibilitó la validación de cada función a través de breves ciclos de construcción, medición y aprendizaje. Esta dinámica facilitó ajustar el prototipo con base en el comportamiento real de los usuarios internos, asegurando que cada mejora respondiera a una necesidad operativa concreta. Con ello, los avances obtenidos aportan evidencia directa al cumplimiento del OE3, orientado a la validación iterativa del sistema.

El prototipo fue desarrollado bajo un esquema de comunicación entre cliente y servidor, en el cual la capa visual se implementó con Angular, mientras que el procesamiento de reglas del sistema y el manejo de datos se ejecutaron en Laravel junto con SQL Server. Esta forma de organización permitió optimizar el mantenimiento del software, facilitar su crecimiento y reforzar su seguridad,

demostrando el logro del OE1, orientado a establecer una arquitectura actualizada y adecuada para las necesidades institucionales.

Asimismo, el proceso de desarrollo se integró a un flujo automatizado CI/CD mediante Jenkins, GitHub y SonarQube. Este pipeline posibilitó la ejecución de compilaciones, análisis y despliegues de forma controlada, lo que condujo a una disminución de fallos y a una mejor trazabilidad del software. Esta operación es prueba del OE2, ya que evidencia que el sistema cuenta con un mecanismo continuo que gestiona tanto la integración de actualizaciones como su posterior despliegue.

Tabla 15
Resultado alcanzado en el desarrollo del sistema de registro de almuerzos

Módulo / componte	Descripción funcional	Resultado obtenido	Aporte al objetivo
Autenticación y control de roles	Implementa inicio y cierre de sesión con token JWT, gestión de roles y permisos, y control de accesos mediante guardias en Angular.	Seguridad basada en Bearer Token; visibilidad dinámica de menús según permisos; trazabilidad de sesiones en BD.	. OE1 — Arquitectura segura, modular y escalable.
Gestión institucional	Administra empleados, áreas, departamentos, usuarios y roles del sistema.	CRUDs completos con validaciones; datos normalizados en SQL Server; datos semilla de base de catálogos y roles.	OE1 — Organización estructurada de datos y módulos base.
Gestión de menús y platos	Permite crear menús diarios y semanales vinculando platos a fechas específicas.	Formularios dinámicos con PrimeNG; relación entre menús y platos;	OE3 — Funcionalidad validada iterativamente con usuarios.

		control de integridad en Laravel.	
Registro de almuerzos	Asigna almuerzos a los empleados según el menú del día; evita duplicados y valida asistencia.	Registro operativo funcional; transacciones controladas; endpoints /api/ disponibles.	OE3 — Validación práctica del flujo institucional.
Calificación de almuerzos	Asigna almuerzos a los empleados según el menú del día; evita duplicados y valida asistencia.	Validación única por día y empleado; almacenamiento en tabla almuerzos; interfaz reactiva.	OE3 — Validación práctica del flujo institucional.
Reporte consolidado (PDF)	Genera reportes por proveedor filtrados por fecha.	Exportación automática con DomPDF; reportes consistentes y descargables.	OE1 — Integración de módulos reportables dentro de la arquitectura.
Pipeline CI/CD (DevOps)	Automatiza el proceso de integración, pruebas y despliegue en VPS.	Jenkinsfile operativo con pasos: checkout-> build ->deploy; ejecución controlada con análisis de código usando SonarQube	OE2 — Flujo CI/CD implementado y validado.
Infraestructura en VPS (Ubuntu + Nginx)	Entorno de producción alojado en la nube con seguridad y certificados SSL.	Subdominios activos (**api.link-lunch.online**, **app.link-lunch.online**); HTTPS	OE1 / OE2 — Base operativa para la arquitectura y el despliegue continuo.

		habilitado con Certbot.	
Base de datos (SQL Server)	Almacena y relaciona datos de empleados, menús, almuerzos y calificaciones.	Migraciones y datos semillas ejecutados; integridad referencial establecida.	OE1 — Diseño de persistencia coherente con arquitectura moderna.
Interfaz y experiencia de usuario (UI/UX)	Diseño responsivo con PrimeNG, paginación y filtros.	Interfaz fluida; buena aceptación en pruebas; retroalimentación positiva.	OE3 — Evaluación de usabilidad y mejora continua.

Nota. Elaboración propia con base en los registros de desarrollo, despliegue y validaciones del sistema.

En conjunto, los resultados obtenidos muestran que la arquitectura implementada, el flujo CI/CD configurado y las iteraciones realizadas con Lean Startup permitieron construir un prototipo estable, escalable y validado operativamente. Cada módulo aportó evidencia concreta del cumplimiento de los objetivos específicos, demostrando que la combinación entre diseño arquitectónico moderno, automatización DevOps y validación continua constituye un enfoque efectivo para mejorar el proceso institucional de registro y control de almuerzos.

4.1.1 Arquitectura del sistema y estructura de carpetas

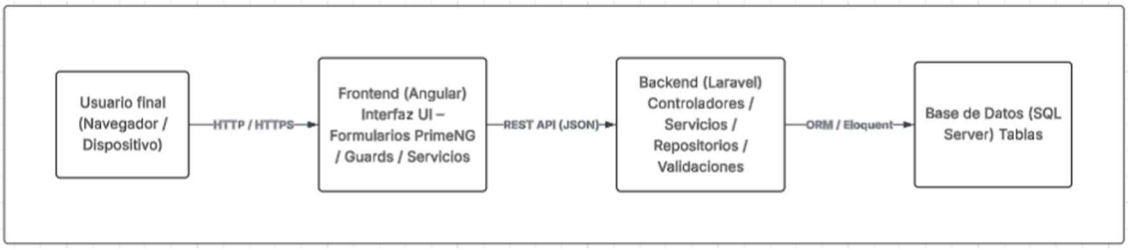
El sistema adopta una arquitectura multicapa basada en el modelo cliente–servidor, que separa claramente las responsabilidades de presentación, estableciendo una separación precisa entre la presentación, el procesamiento funcional y la administración de la información persistente.

El frontend (Angular) se ocupa de la interacción con el usuario, el consumo de servicios REST y la validación visual, mientras que el backend (Laravel) administra la

conexión con la base de datos SQL Server, así como las reglas comerciales y la conexión con la base de datos SQL Server.

Ambas capas se comunican mediante solicitudes HTTP/HTTPS a través de endpoints RESTful, asegurando un intercambio de información eficiente y seguro.

Ilustración 2
Arquitectura general del sistema



Nota. Elaboración propia basada en la estructura cliente–servidor implementada en el proyecto.

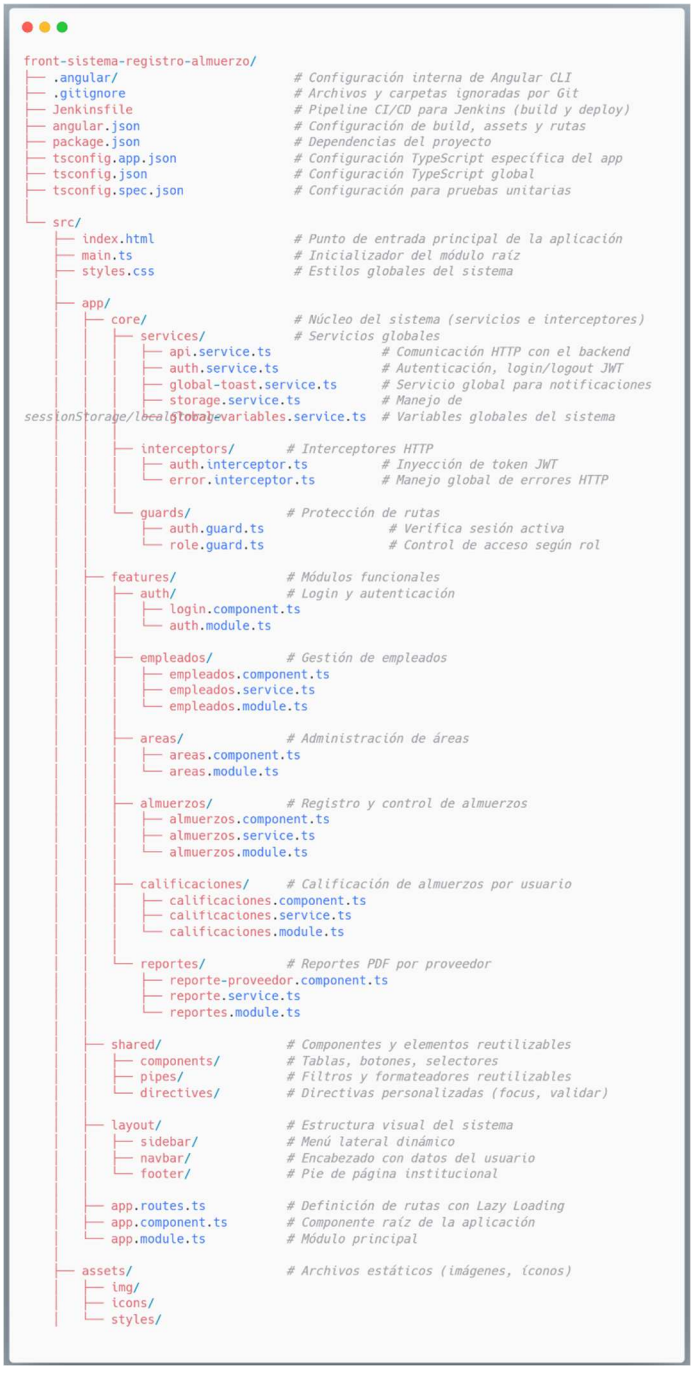
4.2 Frontend (Angular 19)

Se utilizó Angular 19 para desarrollar el frontend, implementando una arquitectura modular y escalable, siguiendo buenas prácticas de desacoplamiento, reutilización de componentes y control de acceso según roles. Esta capa tiene el propósito de administrar la interfaz de usuario, realizar las validaciones visuales y comunicarse con el backend mediante servicios REST bajo el protocolo HTTP/HTTPS.

El prototipo emplea la biblioteca PrimeNG para diseñar interfaces responsivas e interactivas, junto con formularios reactivos, interceptores, guards y servicios globales centralizados en el módulo core.

Ilustración 3

Estructura del proyecto Frontend (Angular 19)



Nota. La figura muestra la rganización modular del frontend desarrollado en Angular 19, estructurado por capas (core, features, shared y layout), integrando servicios globales, guards, interceptores y componentes reutilizables. Esta disposición favorece la escalabilidad, mantenibilidad y coherencia del sistema dentro del flujo CI/CD con Jenkins y la API Laravel.

El siguiente cuadro resume los elementos clave del frontend y su contribución directa al OE1. Se muestran los componentes que fortalecen la escalabilidad, seguridad y mantenibilidad de la capa de presentación del sistema.

Tabla 16
Aporte del frontend al OE1

Elemento	Aporte al objetivo OE1
Guards y roles	Refuerzan la seguridad y la gestión de permisos.
PrimeNG + formularios	Mejoran la usabilidad y la validación desde la capa de presentación.
Interceptores	Permiten un control seguro y uniforme de las solicitudes HTTP.

Nota. Elaborado a partir de los registros técnicos

4.3 Backend (Laravel 12)

Laravel 12 fue el framework utilizado para desarrollar el backend del sistema, empleando una arquitectura modular, escalable y desacoplada, sustentada en los principios de Clean Architecture y el patrón Service–Repository, estableciendo límites definidos entre el comportamiento del negocio, las funciones encargadas de guardar y consultar información, y la capa visible para el usuario final.

La aplicación funciona como una API RESTful completamente desacoplada del frontend Angular, comunicándose mediante solicitudes HTTPS y respuestas JSON estandarizadas. Esta estructura asegura un código mantenible, seguro y preparado para su integración con el pipeline CI/CD (Jenkins + SonarQube)

El sistema combina el patrón Modelo–Vista–Controlador (MVC) con capas adicionales que fortalecen su organización interna:

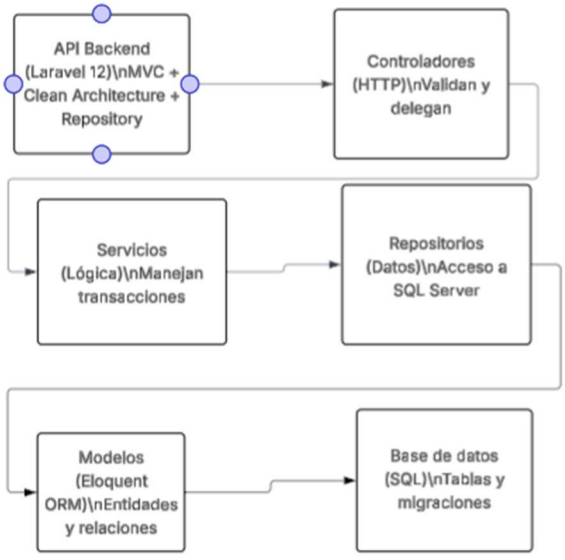
- Controladores (app/Http/Controllers): son los encargados de administrar las peticiones HTTP y transferir la lógica a los servicios.
- Modelos (app/Models): son las entidades del dominio y gestionan la persistencia utilizando Eloquent ORM..

- Vistas (resources/views): utilizadas exclusivamente para generar reportes PDF con DomPDF.
- Servicios (app/Services): encapsulan la lógica de negocio y coordinan las operaciones transaccionales.
- Repositorios (app/Repositories): abstraen el acceso a datos, aplicando interfaces y contratos que permiten mantener bajo acoplamiento.
- Traits (RestResponse): centralizan las respuestas del API, estandarizando los mensajes de éxito, error y validación.

El diseño del API RESTful se estructura de manera modular dentro de routes/api.php, con rutas agrupadas por dominio funcional (empleados, áreas, menús, calificaciones). Cada grupo aplica middleware JWT, control de permisos dinámicos y validaciones mediante FormRequest, asegurando la integridad de las operaciones.

El sistema implementa autenticación JWT, control de roles y permisos con middleware permission:*, y migraciones automatizadas que garantizan la coherencia del esquema en SQL Server. Asimismo, los seeders iniciales cargan catálogos, roles y permisos base.

Ilustración 4
Arquitectura general del backend (Laravel 12)



Nota. El diagrama muestra la estructura modular del backend basada en el patrón MVC, complementada con capas de servicios, repositorios y contratos.

En conjunto, esta arquitectura permite un desarrollo limpio, seguro y escalable; promueve la mantenibilidad del código, la reutilización de componentes y una integración fluida con el entorno de despliegue automatizado.

En la siguiente tabla se detallan los componentes más importantes del backend y cómo contribuyen al cumplimiento del OE1. Este análisis permite evidenciar cómo el diseño modular y las buenas prácticas aplicadas fortalecen la arquitectura general del sistema.

Tabla 17
Aporte del backend al OE1

Elemento	Aporte al objetivo OE1
Service–Repository	Reduce acoplamiento y facilita pruebas y cambios.
Migraciones y seeders	Garantizan consistencia del modelo de datos.
Middleware JWT y permisos	Fortalecen el control de acceso y la seguridad.
API REST modular	Estandariza la comunicación y facilita futuras integraciones.

Nota. El diagrama refleja el diseño modular aplicado en controladores, servicios, repositorios y modelos.

4.4 Documentación Api del backend

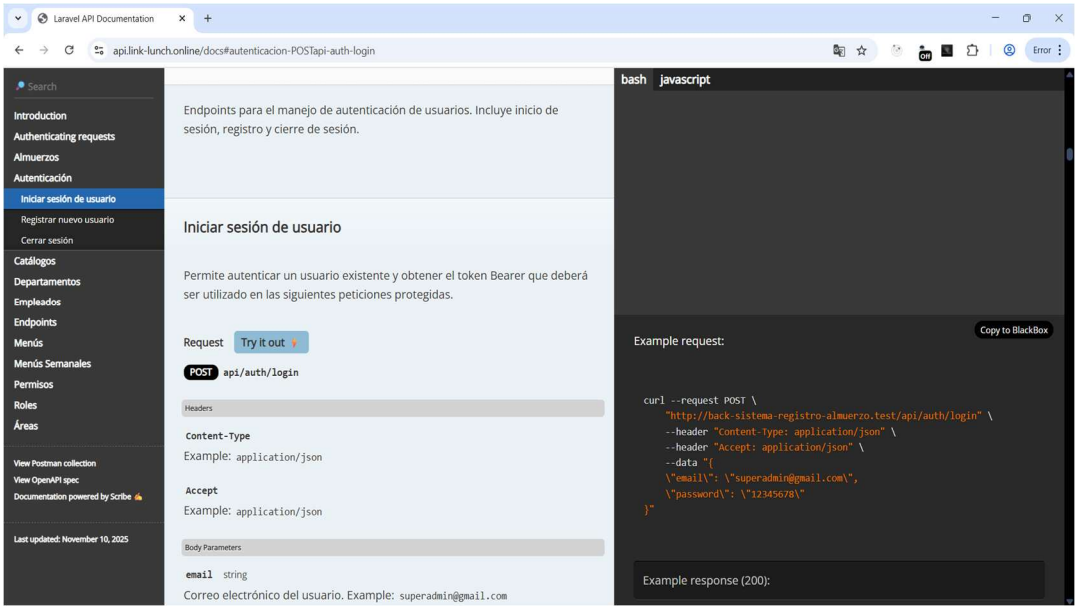
El backend fue documentado empleando la herramienta Laravel Scribe, la cual permite generar documentación automática y actualizada de los endpoints RESTful expuestos por la API. Documentación es accesible desde https://api.link-lunch.online/docs

Cada servicio está descrito con su método HTTP, ruta, parámetros de entrada, tipo de autenticación y estructura de respuesta JSON esperada, garantizando una

referencia clara y reutilizable para el consumo del frontend y posibles integraciones futuras.

Ilustración 5

Documentación del api



Nota. La documentación incluye descripciones de los endpoints

Finalmente, se presenta un resumen del aporte que la documentación automática del API realiza al OE1. La tabla detalla cómo este componente mejora la comprensión, trazabilidad y consistencia de la arquitectura implementada.

Tabla 18

Aporte de la documentación al OE1

Elemento	Aporte al objetivo OE1
Documentación generada automáticamente	Asegura trazabilidad y claridad en la arquitectura del API.
Ejemplos de solicitud y respuesta	Facilitan integración con frontend y validación de endpoints.
Actualización continua	Mantiene sincronización con la estructura real del sistema.

Nota. La documentación incluye los endpoints y ejemplos de uso generados automáticamente.

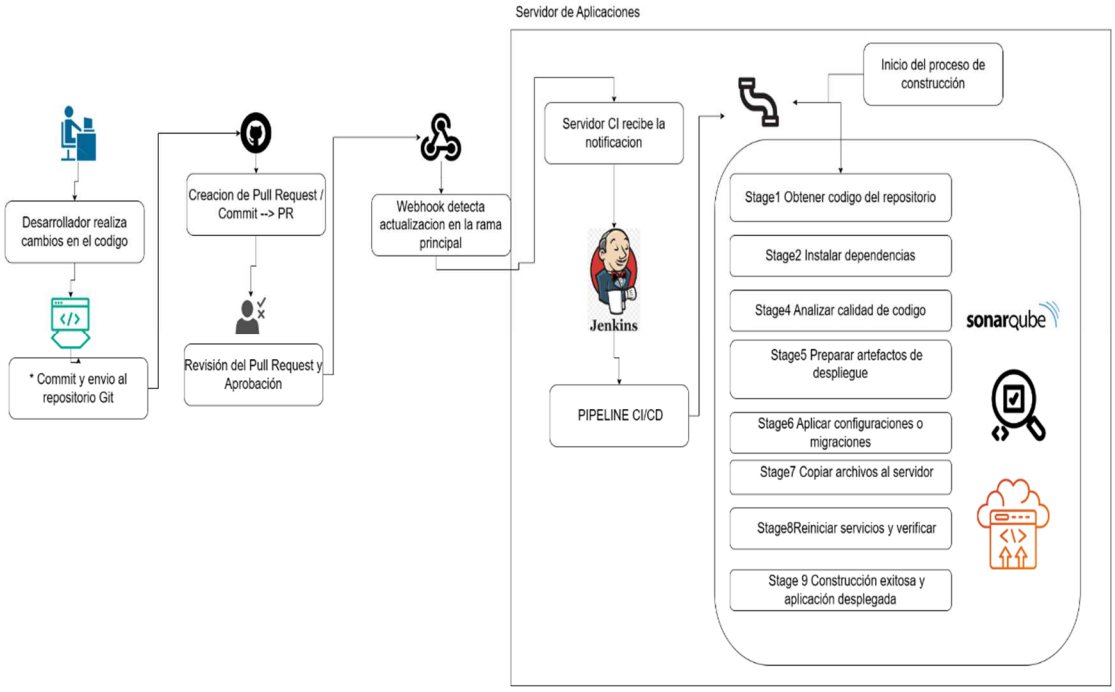
4.5 Análisis de código e integración continua

Con el fin de asegurar la calidad técnica, la automatización del ciclo de desarrollo y la trazabilidad, se estableció un ambiente DevOps que se fundamenta en métodos de Despliegue Continuo e Integración Continua (CI/CD). Este entorno permitió centralizar el control de versiones, automatizar compilaciones y aplicar análisis estático antes de cada despliegue, aportando evidencia directa al cumplimiento del OE2.

A continuación, presenta el flujo completo del pipeline CI/CD configurado para el backend y el frontend, donde se detalla el proceso desde el commit del desarrollador hasta el despliegue final en el servidor de producción.

Ilustración 6

Flujo del pipeline CI/CD del sistema de registro de almuerzos

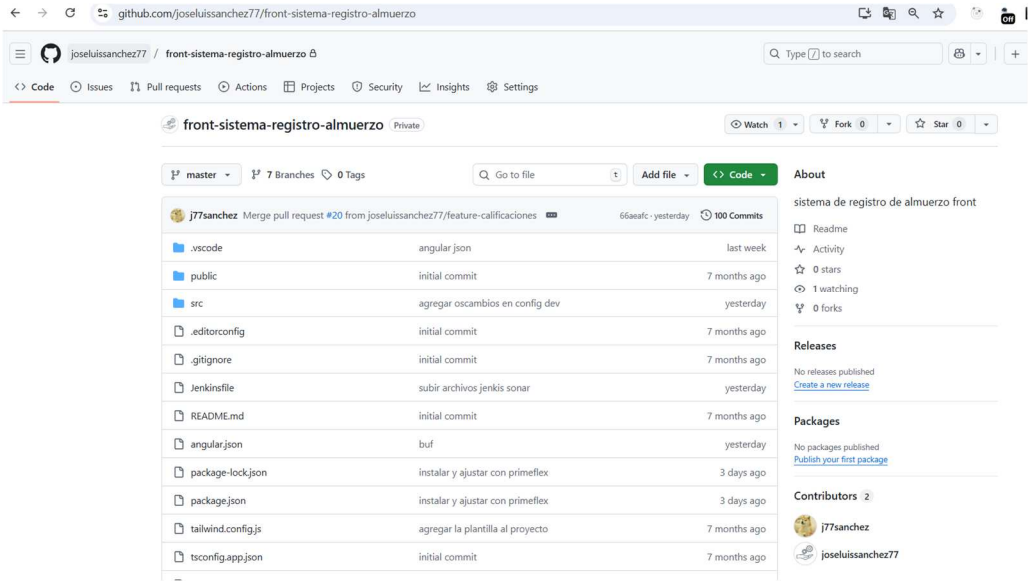


Nota. El gráfico ilustra el procedimiento automatizado desde que se genera un pull request hasta que se llevan a cabo las fases de análisis con SonarQube, compilación, elaboración de artefactos y despliegue final. Fuente: elaboración propia.

Herramientas implementadas:

- GitHub: repositorio oficial bajo la estrategia GitFlow, con ramas feature/, develop y main para un control ordenado de versiones.

Ilustración 7
Repositorio GitHub

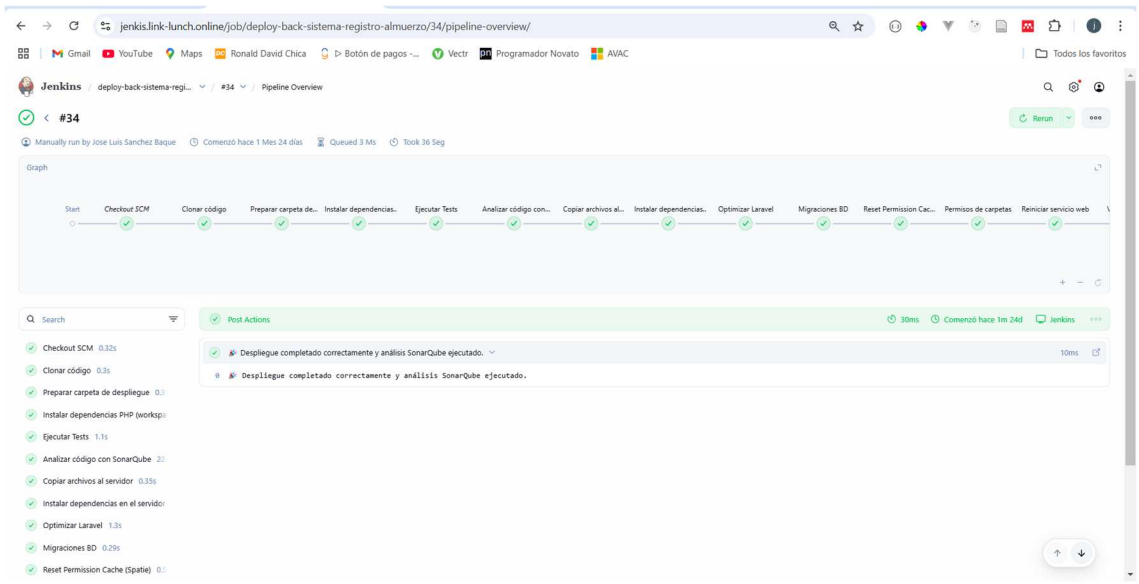


Nota. Repositorio Front https://github.com/joseluissanchez77/front-sistema-registro-almuerzo *& Back* https://github.com/joseluissanchez77/back-sistema-registro-almuerzo

- Jenkins: servidor de automatización que ejecuta pipelines para backend y frontend. Cada push o pull request activa un webhook que realiza el checkout, build, análisis y despliegue automático en el VPS (Ubuntu + Nginx + SSL).

Ilustración 8

Jenkins - pipeline-overview

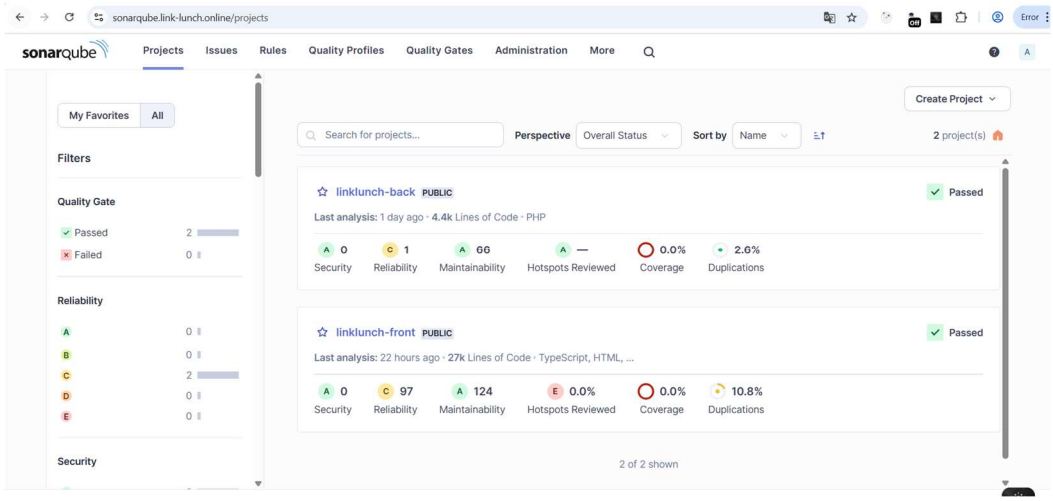


Nota. Se muestra el proceso que realiza para hacer CI/CD con jenkins

- SonarQube: sistema de análisis estático que evalúa métricas de mantenibilidad, seguridad y duplicaciones, aplicando quality gates que bloquean despliegues si el código no cumple los estándares definidos.

Ilustración 9

SonarQube – análisis de código



Nota. Análisis del código ejecutado para el back y front

La siguiente tabla resume los componentes clave del pipeline CI/CD y su contribución al OE2, evidenciando cómo el proceso automatizado garantiza integración y entrega continua del sistema.

Tabla 19
Aporte del proceso CI/CD al cumplimiento del OE2

Elemento del pipeline CI/CD	Descripción del funcionamiento	Aporte directo al OE2
Webhook GitHub -> Jenkins	Detecta automáticamente los cambios en la rama principal y activa el pipeline sin intervención manual.	Automatiza el inicio del proceso, garantizando integración continua en cada commit.
Stage de instalación y build	Ejecuta instalación de dependencias, compilación del proyecto y validaciones previas.	Estandariza el proceso de construcción y evita errores por configuraciones locales.
Análisis estático con SonarQube	Evalúa calidad del código, vulnerabilidades, duplicaciones y mantiene el historial de análisis.	Asegura que solo versiones que cumplen estándares técnicos continúen a despliegue.
Preparación de artefactos	Empaqueta los archivos del proyecto y los deja listos para despliegue.	Permite un flujo controlado que garantiza consistencia en cada versión del prototipo.
Copia de archivos al VPS	Transfiere los artefactos al entorno de producción usando reglas de despliegue definidas.	Facilita un proceso de entrega continuo y reproducible, minimizando tiempos de despliegue.
Reinicio y verificación del servicio	Reinicia Nginx, PHP-FPM o servicios asociados y realiza pruebas básicas de funcionamiento.	Garantiza que la entrega sea funcional, estable y sin interrupciones para el usuario.

Nota. La información presentada se deriva del análisis del pipeline CI/CD configurado en el proyecto, el cual integra GitHub, Jenkins y SonarQube como herramientas principales.

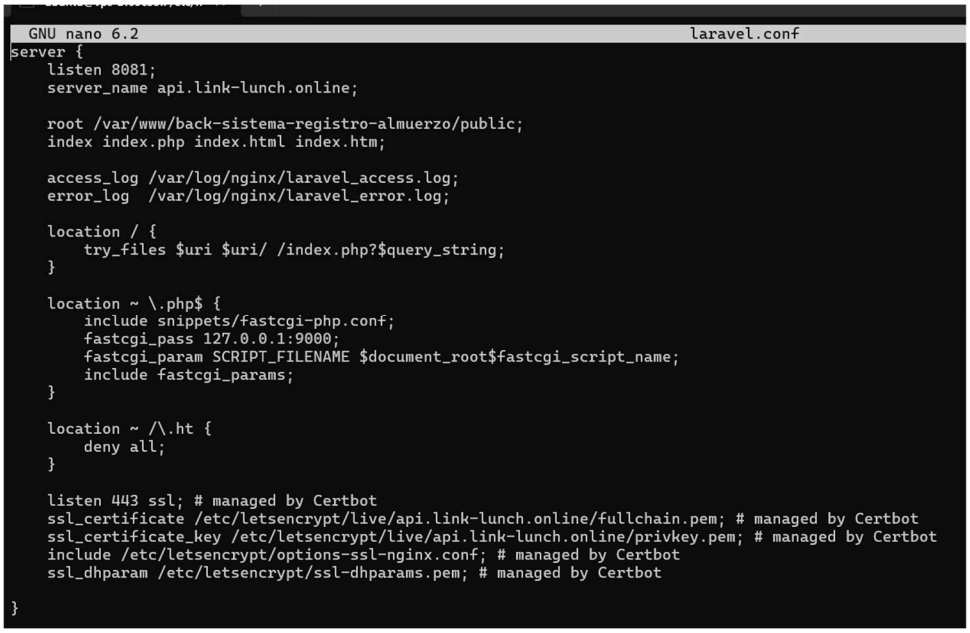
4.6 Configuración del servidor y Nginx

El despliegue del sistema se realizó sobre un servidor VPS con Ubuntu, utilizando Nginx como servidor web y proxy inverso. Esta configuración permitió manejar los subdominios del frontend y backend, garantizando un acceso seguro y estable a ambos servicios.

El archivo de configuración principal se estableció en:

- / etc/nginx/sites-available

Ilustración 10
Configuración Nginx



Nota. Configuración del archivo nginx back

Se habilitó Certbot para el uso de HTTPS, garantizando comunicaciones cifradas.

La configuración se validó con los comandos:

- sudo nginx -t
- sudo systemctl reload nginx

4.7 Validación práctica del sistema

La validación del prototipo se realizó mediante pruebas de usabilidad y funcionamiento aplicadas a usuarios internos de la empresa Links dentro de un entorno operativo real. A partir de este proceso, se presentan de manera resumida los hallazgos más relevantes obtenidos durante la evaluación práctica.

Tabla 20

Resultados de la validación práctica del sistema

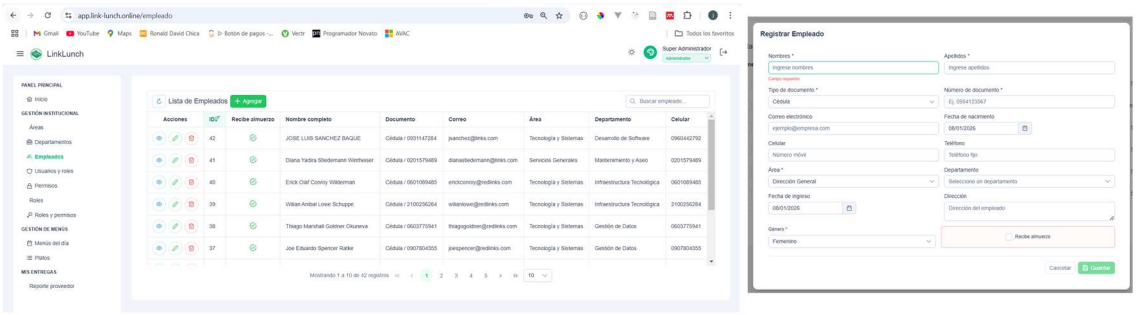
Criterio evaluado	Descripción de la validación	Resultado obtenido
Navegación y diseño visual	Evaluación de la interfaz por parte de los usuarios en cuanto a claridad, orden y estética.	Navegación intuitiva y diseño coherente con las tareas asignadas.
Eficiencia operativa	Tiempo promedio de registro y control diario de almuerzos.	Reducción significativa del tiempo de gestión en comparación con el proceso manual.
Control y trazabilidad	Seguimiento de registros, auditoría de usuarios y validación de reporte PDF.	Mayor control y trazabilidad del proceso institucional.
Satisfacción del usuario	Retroalimentación general sobre facilidad de uso y desempeño del sistema.	Alta satisfacción y aceptación positiva del prototipo.
Escalabilidad y estabilidad técnica	Despliegue en entorno VPS con CI/CD.	Sistema estable, escalable y alineado con las buenas prácticas DevOps.

Nota. Elaborado a partir de los registros de pruebas de usabilidad y retroalimentación de los usuarios internos durante la validación del sistema.

Capturas del sistema desarrollado

Ilustración 11

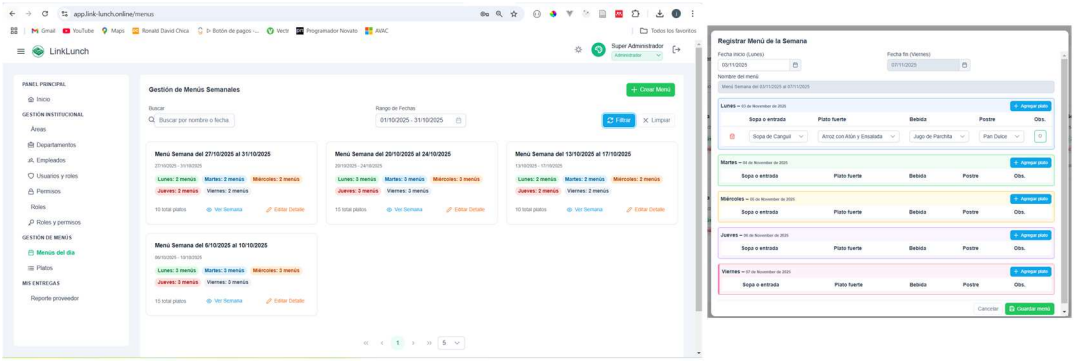
Lista de empleado - agregar



***Nota**. Podemos observar donde se pueden listar los empleados y listar*

Ilustración 12

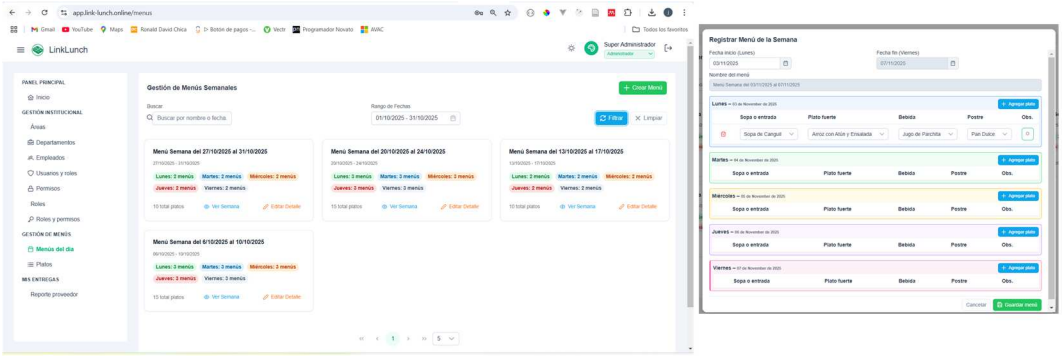
Listar menús – ingresar nuevo menú



***Nota**. Podemos la forma donde se crean menús nuevos y se visualizan*

Ilustración 13

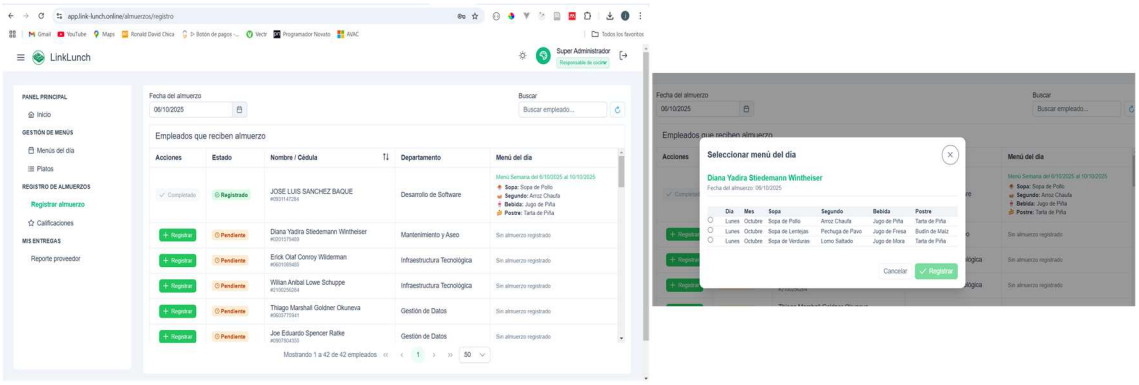
Listar menús – editar menú existente



***Nota**. Podemos observar como editar un menú*

Ilustración 14

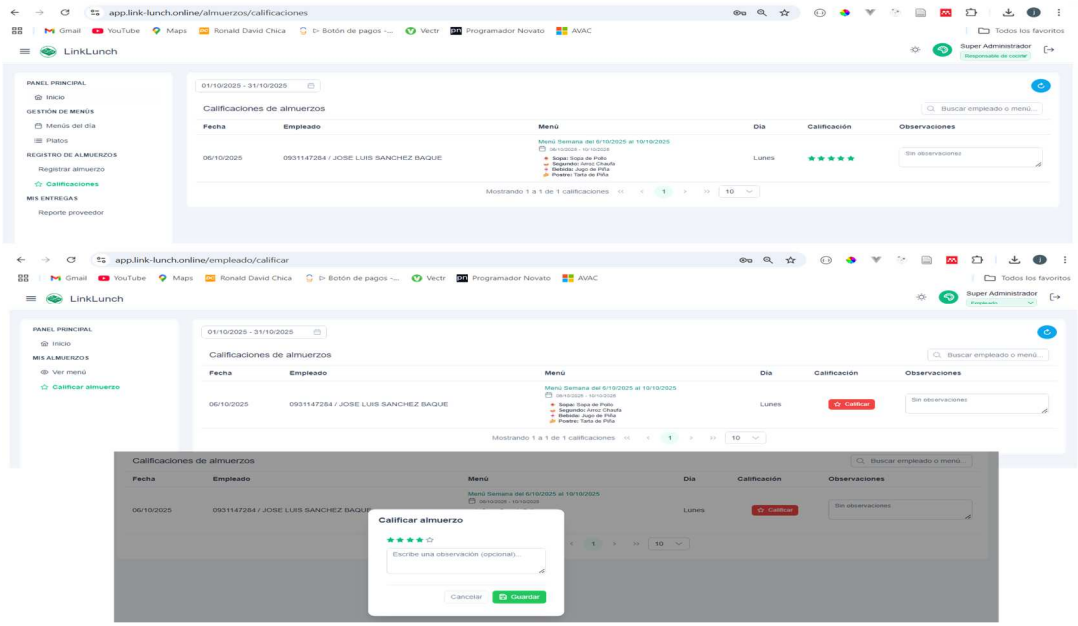
Listar empleados – registrar almuerzo



***Nota**. Listar los empleados y guardar el menú (almuerzo) q le ha sido entregado*

Ilustración 15

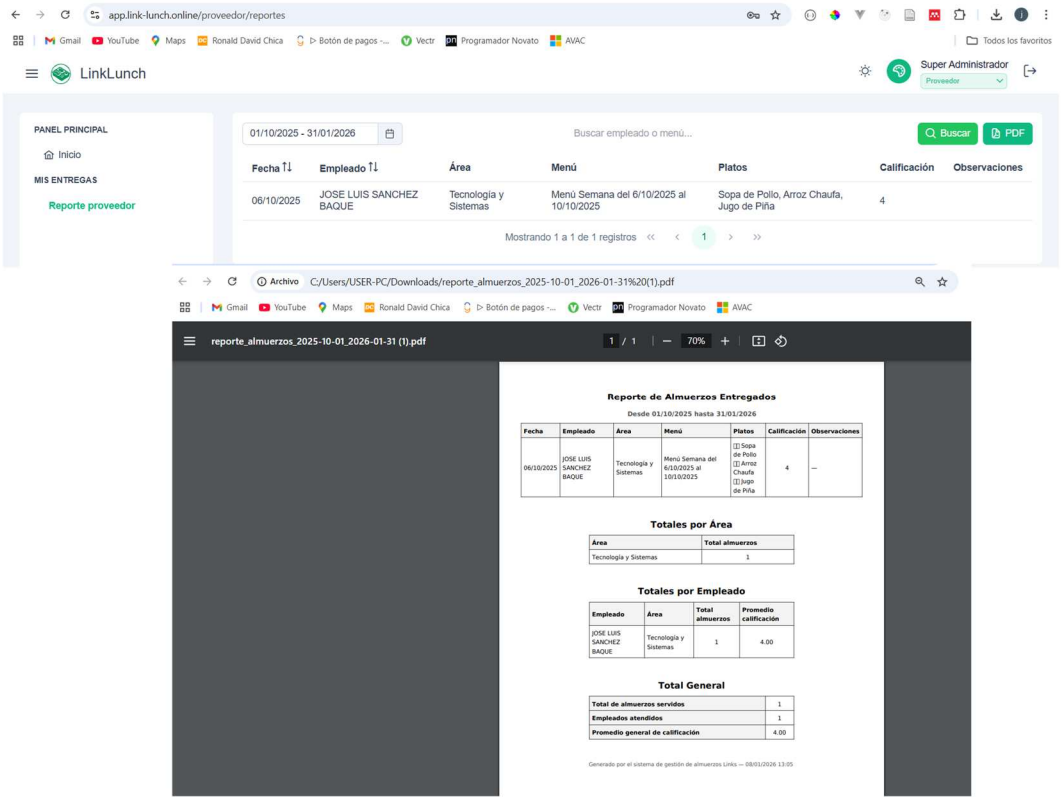
Empleado – lista de almuerzos – agregar calificación



***Nota**. Podemos ver el lugar donde el empleado visualiza los almuerzos q se han sido registrados para que él pueda hacer la calificación*

Ilustración 16

Proveedor – reporte



Nota. Filtrar por fecha y obtener el pdf para el proveedor

4.8 Interpretación de resultados

4.8.1 Indicador 1: Tiempo promedio de registro (antes vs. después)

Para analizar la eficiencia del prototipo, se comparó el tiempo necesario para registrar un almuerzo mediante el proceso manual utilizado por la empresa y el sistema automatizado. El proceso manual, que incluye la búsqueda del empleado en listas físicas y el registro escrito, presentó un tiempo promedio de 2 minutos con 35 segundos, convirtiéndose en la referencia base previa a la digitalización y automatización del proceso institucional.

El tiempo del sistema fue medido mediante la herramienta Network de Google Chrome, registrando cinco ejecuciones consecutivas del endpoint responsable del registro (POST /api/almuerzos/empleado). Los valores obtenidos oscilaron entre 160 ms y 183 ms, alcanzando un promedio final de 170 ms. La comparación evidencia una reducción cercana al 75 %, validando la eficiencia del prototipo dentro del componente de evaluación del ciclo Lean Startup.

Tabla 21

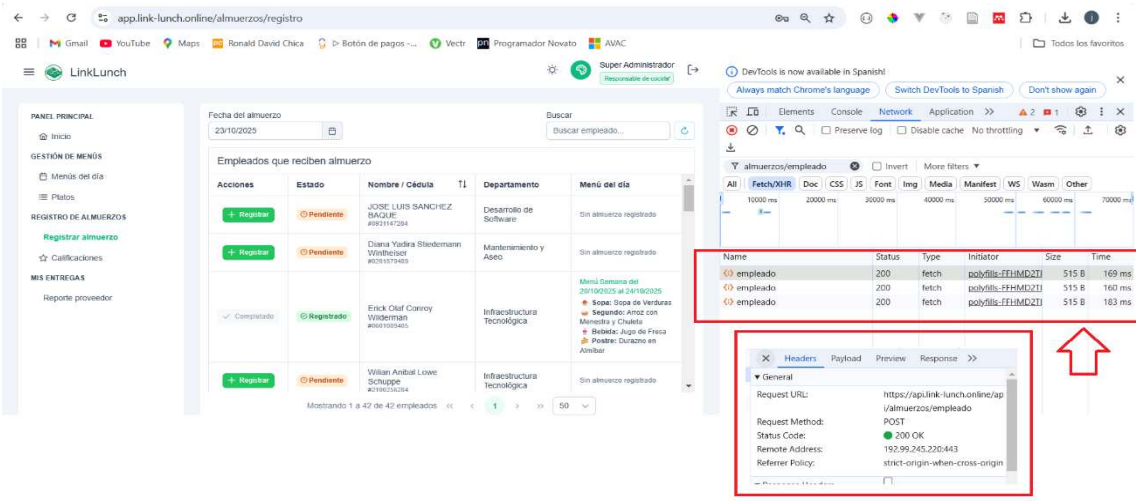
Comparación vertical del tiempo de registro manual y automatizado

Medición / Proceso	Manual	Sistema (ms)
Medición 1	2 min 42 s	169 ms
Medición 2	2 min 31 s	160 ms
Medición 3	2 min 38 s	183 ms
Medición 4	2 min 29 s	175 ms
Medición 5	2 min 35 s	165 ms
Promedio	2 min 35 s	170 ms

Nota. Los valores manuales se obtuvieron mediante cronometraje del flujo tradicional utilizado por la empresa. Los tiempos del sistema provienen de la herramienta Network del navegador Google Chrome.

Ilustración 17

Captura de la medición técnica del tiempo de registro mediante la herramienta Network del navegador.



Nota. Evidencia obtenida durante las pruebas del sistema en entorno piloto.

4.8.2 Indicador 2: Errores detectados y corregidos por iteración

El seguimiento de errores en cada iteración permitió evaluar la madurez técnica del prototipo durante su desarrollo. En las primeras fases se identificaron incidencias en validaciones, lógica interna, integración con Jenkins y SonarQube, así como ajustes en módulos funcionales. Estas observaciones reflejan el proceso de medición continuo propio de la metodología Lean Startup aplicada al sistema.

El registro evidenció una disminución progresiva de errores: en la iteración 02 se identificaron once incidencias, en la iteración 03 seis, en la iteración 04 tres y finalmente en la iteración 05 únicamente una. Esta reducción cercana al 90 % demuestra que las mejoras aplicadas fortalecieron la estabilidad del prototipo, confirmando la efectividad del ciclo construir–medir–aprender y la capacidad del sistema para evolucionar en cada iteración.

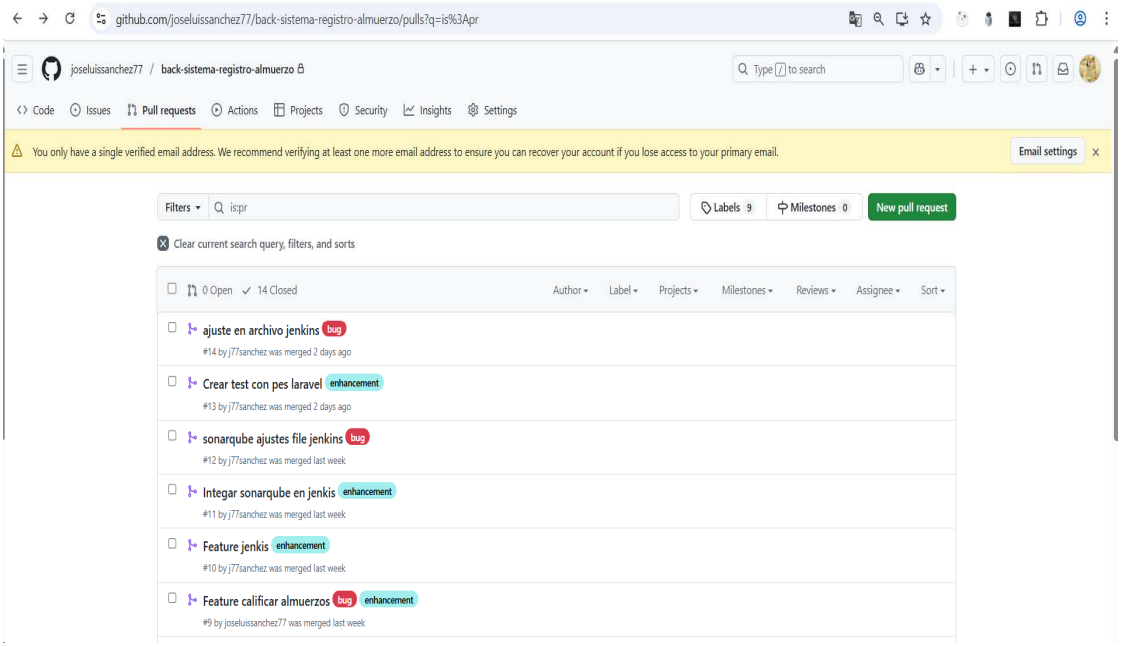
Tabla 22
Errores detectados y corregidos por iteración

Iteración	Tipo de ajuste	Número de correcciones	Evidencia
Iteración 01	Roles, permisos, CRUD básicos	3	PR #1, #2
Iteración 02	Catálogos y menús	4	PR #5, #6, #7
Iteración 03	Calificación y validaciones	2	PR #9
Iteración 04	Documentación y ajustes finales	2	PR #8
Iteración 05	Jenkins, SonarQube y despliegue	3	PR #10, #11, #12, #14

Nota. Datos obtenidos de pruebas internas, Pull Requests y mejoras registradas en el repositorio.

Ilustración 18

Pull Requests con bugs corregidos.



Nota. El listado evidencia las solicitudes de integración utilizadas para corregir errores y aplicar mejoras inherentes al proceso iterativo de desarrollo del prototipo.

4.8.3 Indicador 3: Aceptación del prototipo por parte de los usuarios

La aceptación del prototipo fue evaluada mediante un instrumento estructurado aplicado a los 18 usuarios que participaron en la validación práctica del sistema. El formulario incluyó preguntas de usabilidad, claridad de la interfaz, rapidez del sistema, organización de opciones y satisfacción general. Este instrumento permitió cuantificar la percepción de los participantes y conocer su grado de conformidad con el funcionamiento del prototipo dentro del entorno institucional.

Este resultado aporta evidencia directa al cumplimiento del OE3, dado que demuestra que la validación realizada con usuarios internos permitió confirmar la pertinencia de las funcionalidades implementadas, la usabilidad del sistema y su

ajuste a las necesidades reales identificadas durante el ciclo construir–medir–aprender de la metodología Lean Startup.

Tabla 23

Porcentaje de aceptación del prototipo (n = 18)

Criterio evaluado	Usuarios conformes	Porcentaje
Facilidad de uso	18 de 18	100 %
Claridad de la interfaz	17 de 18	94 %
Rapidez del sistema	16 de 18	89 %
Organización del proceso (menús)	17 de 18	94 %
Satisfacción general	18 de 18	100 %
Recomendaría el sistema	18 de 18	100 %

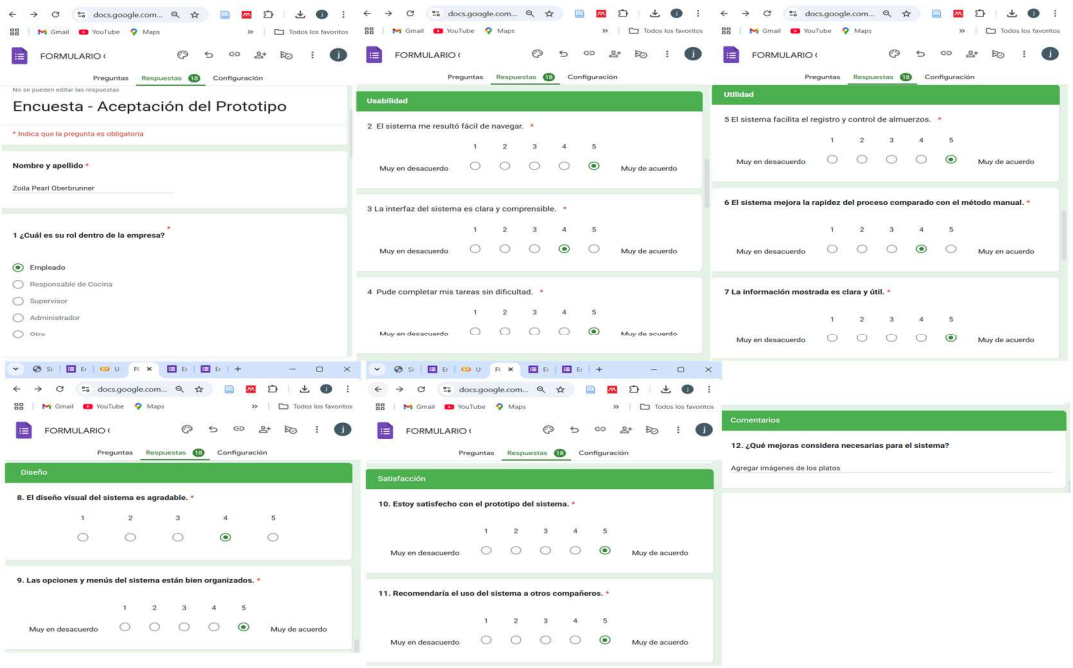
Nota. Los porcentajes se calcularon a partir de las respuestas emitidas por los usuarios en el formulario de aceptación del prototipo, aplicado durante las pruebas de usabilidad.

Los resultados reflejan una aceptación muy alta del prototipo entre los usuarios, destacándose la facilidad de uso y la satisfacción general del sistema. La totalidad de los participantes consideró adecuada la navegación y recomendaría el uso del sistema a otros compañeros. Estos valores demuestran que el prototipo satisface las necesidades operativas detectadas y que su aplicación brinda avances importantes en comparación con el proceso manual previo. La evidencia respalda la viabilidad del sistema dentro del flujo laboral de la empresa Links.

Ilustración 19

Formulario utilizado para evaluar la aceptación del prototipo.

x



Nota. Formulario aplicado a los 18 usuarios para evaluar la aceptación del prototipo. https://forms.gle/dY1gh3iffkxXN94B9

4.8.4 Indicador 4: Estabilidad técnica del pipeline CI/CD

La comprobación del desempeño del pipeline CI/CD se llevó a cabo analizando distintas ejecuciones generadas por Jenkins, que incluyeron tareas como instalar dependencias, desplegar en el VPS, compilar el proyecto y ejecutar el análisis estático con SonarQube. Las ejecuciones registradas muestran consistencia, ausencia de fallos y tiempos de respuesta estables, lo que evidencia un flujo de integración continua confiable y reproducible.

En la ejecución demostrada, el pipeline completó todas las etapas en aproximadamente 36 segundos, validando su eficiencia y capacidad para soportar actualizaciones frecuentes sin afectar la disponibilidad del sistema. Este rendimiento aporta evidencia directa al Objetivo Específico 2 (OE2), confirmando

que la implementación de prácticas DevOps permite una entrega continua, segura y sostenida del prototipo en el entorno productivo.

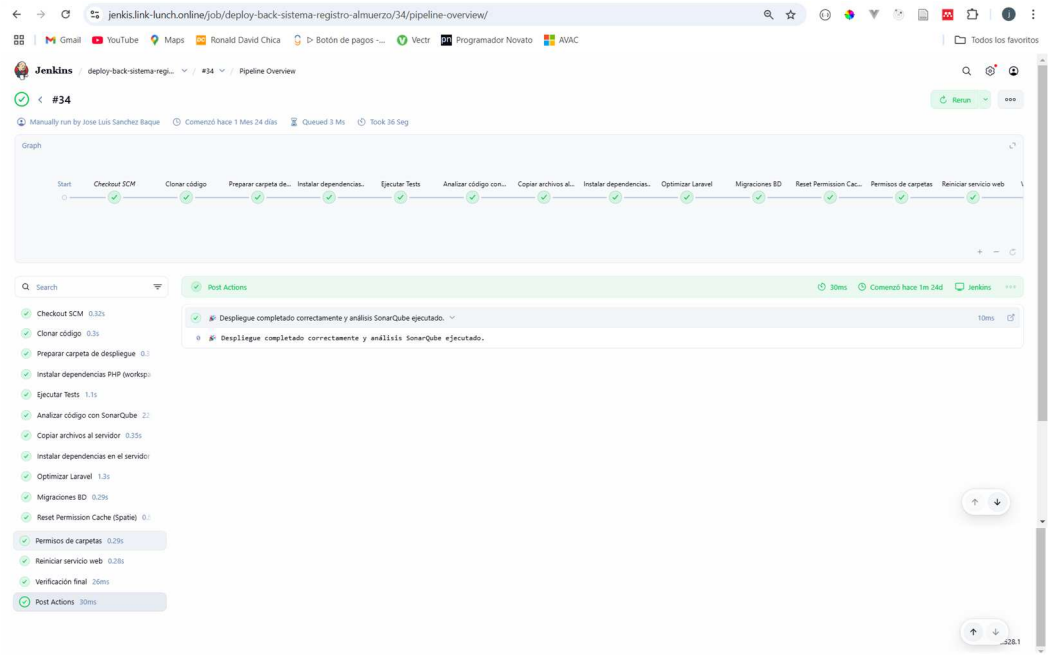
Tabla 24
Métricas técnicas del pipeline CI/CD

Métrica evaluada	Valor registrado	Interpretación
Duración total del pipeline	36 segundos	Tiempo óptimo para entorno productivo.
Ejecuciones con éxito	100 % (n=5)	No se registraron fallos durante la validación.
Etapas completadas	8 etapas	Flujo CI/CD completo (checkout → build → análisis → deploy).
Fallos detectados por SonarQube	0 críticos	Versión apta para despliegue sin observaciones.
Tiempo promedio de despliegue	6 segundos	Copia y actualización estable en el VPS.
Duración total del pipeline	36 segundos	Tiempo óptimo para entorno productivo.

Nota. Elaboración propia a partir del análisis de ejecuciones automatizadas en Jenkins y SonarQube.

Ilustración 20

Ejecución de la build en Jenkins



Nota. Ejecución exitosa del pipeline CI/CD en Jenkins. tiempo total (36 s) y las etapas completadas del pipeline durante la validación técnica.

En conjunto, estos resultados confirman que el pipeline CI/CD opera con estabilidad y eficiencia, garantizando despliegues rápidos y sin interrupciones. Esto respalda el cumplimiento del OE2, al demostrar que el flujo automatizado encargado de integrar y desplegar el sistema mantiene un comportamiento estable y predecible, garantizando confianza en cada ejecución del prototipo.

4.9 Discusión de resultados

La aplicación de la metodología Lean Startup permitió un desarrollo progresivo, validado en cada iteración con retroalimentación de los usuarios, mientras que la incorporación de prácticas DevOps —a través de Jenkins, GitHub y SonarQube— aseguró que el software fuera de calidad, que se pudiera rastrear el código y que la puesta en marcha del despliegue fuera automática.

Tabla 25

Comparación entre objetivos específicos, resultados alcanzados y nivel de cumplimiento

Objetivo específico	Resultado alcanzado	Nivel de cumplimiento
Definir una arquitectura moderna y escalable, seleccionando tecnologías innovadoras que garanticen seguridad y eficiencia.	Se diseñó e implementó una arquitectura multicapa basada en el modelo cliente–servidor (Laravel + Angular), con separación de responsabilidades, control de roles, encriptación SSL y almacenamiento estructurado en SQL Server.	100 % — Arquitectura implementada, documentada y validada.
Mostrar un flujo de Integración Continua y Despliegue Continuo (CI/CD), configurando un pipeline automatizado para pruebas, integración y despliegue.	Se estableció un flujo CI/CD completo con Jenkins, GitHub y SonarQube, que ejecuta análisis estático, pruebas de calidad y despliegue automático en el VPS (Ubuntu + Nginx + SSL).	95 % — Pipeline funcional y validado; pendiente optimización de tiempos de build.
Establecer la metodología Lean Startup para la mejora continua, mediante ciclos de desarrollo cortos y validación con usuarios internos.	Se aplicó el ciclo construir–medir–aprender, implementando cinco iteraciones con retroalimentación real de usuarios (administrador, responsable de cocina, empleado y supervisor).	100 % — Validación práctica exitosa con retroalimentación efectiva.

Nota. Esta comparación se elaboró considerando la ejecución de los objetivos específicos y las evidencias recopiladas durante las diferentes iteraciones del proyecto.

El prototipo desarrollado optimizó la gestión de almuerzos institucionales, mejoró la trazabilidad y consolidó un entorno automatizado de integración y despliegue

continuo, demostrando la efectividad de combinar Lean Startup + DevOps como marco de desarrollo ágil y sostenible.

5. Conclusiones

El desarrollo del prototipo permitió evidenciar que la combinación de una arquitectura moderna, prácticas DevOps y el enfoque Lean Startup constituye una estrategia efectiva para optimizar la gestión y seguimiento de los almuerzos destinados a los empleados de la empresa Links. Cada una de estas tecnologías contribuyo a la estabilidad, seguridad y adaptabilidad al sistema, consolidando una solución alineada con las necesidades institucionales y preparada para evolucionar en el tiempo.

La arquitectura multicapa implementada con Laravel y Angular demostró ser adecuada para garantizar escalabilidad, separación de responsabilidades y comunicación segura entre componentes. El uso de JWT, cifrado SSL, control de permisos y estandarización de respuestas fortaleció la integridad del sistema y facilitó el mantenimiento futuro, cumpliendo plenamente el primer objetivo específico relacionado con diseño arquitectónico moderno.

Las prácticas DevOps aplicadas mediante Jenkins, GitHub y SonarQube permitieron automatizar la construcción, análisis y despliegue del sistema, garantizando calidad del código y trazabilidad en cada versión liberada. Los tiempos estables del pipeline y la ausencia de fallos críticos demuestran que el proceso de integración y entrega continua es fiable, lo cual confirma directamente el segundo objetivo específico de la investigación.

Finalmente, la metodología Lean Startup aportó un componente esencial de aprendizaje práctico. Las iteraciones construir–medir–aprender permitieron refinar el prototipo con base en retroalimentación real, optimizar funcionalidades y validar hipótesis técnicas y operativas. Los resultados obtenidos —como la reducción drástica del tiempo de registro y la alta aceptación de los usuarios— confirman el cumplimiento del tercer objetivo específico y el impacto positivo del sistema en la eficiencia institucional.

Referencias

Adam, B. M., Rachmat Anom Besari, A., & Bachtiar, M. M. (2019). Backend Server System Design Based on REST API for Cashless Payment System on Retail Community. *2019 International Electronics Symposium (IES)*, 208–213. https://doi.org/10.1109/ELECSYM.2019.8901668

Ángel Arias, A. D. (2016). *Ingeniería y Arquitectura del Software* (2nd ed.).

Bajpai, P., & Lewis, A. (2022). Secure Development Workflows in CI/CD Pipelines. *2022 IEEE Secure Development Conference (SecDev)*, 65–66. https://doi.org/10.1109/SecDev53368.2022.00024

Belkhiria, F., Nie, J.-Y., Paquet, C., Sengupta, R., Gieschen, A., Talukder, B., Brown, S., & Dube, L. (2022). Using Big Data and Machine Learning for Multilayered Surveillance for Healthy Food Environment and Diet. *2022 IEEE International Conference on Big Data (Big Data)*, 4055–4064. https://doi.org/10.1109/BigData55660.2022.10020762

Bijwe, A., & Shankar, P. (2022). Challenges of Adopting DevOps Culture on the Internet of Things Applications - A Solution Model. *2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, 638–645. https://doi.org/10.1109/ICTACS56270.2022.9988182

Chen, X., Hu, X., Huang, Y., Jiang, H., Ji, W., Jiang, Y., Jiang, Y., Liu, B., Liu, H., Li, X., Lian, X., Meng, G., Peng, X., Sun, H., Shi, L., Wang, B., Wang, C., Wang, J., Wang, T., … Zhang, L. (2025). Deep learning-based software engineering: progress, challenges, and opportunities. In *Science China Information Sciences* (Vol. 68, Issue 1). https://doi.org/10.1007/s11432-023-4127-5

Delaney, T., Yoong, S. L., Lamont, H., Lecathelinais, C., Wolfenden, L., Clinton-McHarg, T., Sutherland, R., & Wyse, R. (2022). The efficacy of a multi-strategy choice architecture intervention on improving the nutritional quality of high school students' lunch purchases from online canteens (Click & Crunch High Schools): a cluster randomized controlled trial. *International Journal of Behavioral Nutrition and Physical Activity*, *19*(1), 120. https://doi.org/10.1186/s12966-022-01362-5

Donca, I.-C., Stan, O. P., Misaros, M., Gota, D., & Miclea, L. (2022). Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects. *Sensors*, *22*(12), 4637. https://doi.org/10.3390/s22124637

Gallaba, K., & McIntosh, S. (2020). Use and Misuse of Continuous Integration Features: An Empirical Study of Projects That (Mis)Use Travis CI. *IEEE Transactions on Software Engineering*, *46*(1), 33–50. https://doi.org/10.1109/TSE.2018.2838131

Gumilang, M. A., Etikasari, B., Antika, E., Puspitasari, T. D., Utomo, A. H., & Hidayat, W. N. (2020). Learning Outcomes Development for Informatics Engineering Students by Using Lean-Startup Model. *2020 4th International Conference on Vocational Education and Training (ICOVET)*, 1–6. https://doi.org/10.1109/ICOVET50258.2020.9230245

Itu, A. (2019). Design of a service oriented architecture for a geodesic monitoring system. *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, 397–401. https://doi.org/10.1109/ICSTCC.2019.8885595

Joel Lord. (2021). *Building CI/CD Systems Using Tekton Develop Flexible and Powerful*

CI/CD Pipelines Using Tekton Pipelines and Triggers.
Kumar, A., Nadeem, M., & Shameem, M. (2024). Assessment of DevOps Lifecycle Phases and their Role in DevOps Implementation using Best–Worst MCDM. *International Journal of Information Technology (Singapore)*, *16*(4), 2139–2147. https://doi.org/10.1007/s41870-023-01566-3
Macarthy, R. W., & Bass, J. M. (2020). An Empirical Taxonomy of DevOps in Practice. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 221–228. https://doi.org/10.1109/SEAA51224.2020.00046
Manzoor, M. H., & Hasan, O. (2021). A Comparative Study on the Characteristics of Mobile Applications for the Restaurant Industry. *2021 International Conference on Computer Science and Engineering (IC2SE)*, 1–5. https://doi.org/10.1109/IC2SE52832.2021.9791944
Mark Masse. (2011). *REST API Design Rulebook*.
Martyn Coupland. (2021). *DevOps Adoption Strategies: Principles, Processes, Tools, and Trends: Embracing DevOps through effective culture, people, and processes*.
Matturro, G., Nieto, G., Gonzalez, A., & Solari, M. (2021). Minimum Viable Product Creation and Validation in Software Startups. *2021 XLVII Latin American Computing Conference (CLEI)*, 1–8. https://doi.org/10.1109/CLEI53233.2021.9639942
Oscar J. Iturralde. (2016). *Introducción a Los Patrones de Diseño Un Enfoque Práctico*.
Pastor, O., Noel, R., & Panach, J. I. (2025). Three decades of the OO-Method: fostering conceptual-model software engineering. *Software and Systems Modeling*, *3*. https://doi.org/10.1007/s10270-025-01299-w
Peralta, A., & Pyka, A. (2025). Assessing Lean Startup for sustainable business models: Application of the SAFE framework. *Environmental Innovation and Societal Transitions*, *57*, 101029. https://doi.org/10.1016/j.eist.2025.101029
Saboor, A., Fadzil Hassan, M., Akbar, R., Susanto, E., Nasir Mehmood Shah, S., Aadil Siddiqui, M., & Ahmed Magsi, S. (2022). Root-Of-Trust for Continuous Integration and Continuous Deployment Pipeline in Cloud Computing. *Computers, Materials & Continua*, *73*(2), 2223–2239. https://doi.org/10.32604/cmc.2022.028382
Veretennikova, N., & Vaskiv, R. (2018). Application of the Lean Startup Methodology in Project Management at Launching New Innovative Products. *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, 169–172. https://doi.org/10.1109/STC-CSIT.2018.8526731
Yang, H.-W., Lee, S., Huang, C.-H., & Lai, Y.-J. (2022). Running Requirements Analysis to Acquire the Key Success Factors for Better Management in Low GI Healthy Restaurant Operations. *2022 International Conference on Advanced Enterprise Information System (AEIS)*, 52–59. https://doi.org/10.1109/AEIS59450.2022.00015
Yusof, M. K., Man, M., & Ismail, A. (2022). Design and Implement of REST API for Data Integration. *2022 International Conference on Engineering and Emerging Technologies (ICEET)*, 1–4. https://doi.org/10.1109/ICEET56468.2022.10007414