UNIB.E

UNIVERSIDAD IBEROAMERICANA DEL ECUADOR

FACULTAD DE COMUNICACIÓN Y TECNOLOGÍAS

CARRERA: INGENIERÍA DE SOFTWARE

Automatización de procesos de integración de código, compilación, pruebas y despliegue para proyectos de software en el Ecuador.

Trabajo de Integración Curricular para la obtención del Título de Ingeniero de Software

Autores:
Alexander Patricio Mejía Arias
José Rodolfo Obando Maldonado

Tutor:
Harry Carpio, Msc.

Quito, Ecuador
Febrero, 2024

DECLARACIÓN DE AUTORÍA Y AUTORIZACIÓN PARA LA DIFUSIÓN DEL TRABAJO DE INTEGRACIÓN CURRICULAR

1. Nosotros, **José Rodolfo Obando Maldonado y Alexander Patricio Mejía Arias,** declaramos en forma libre y voluntaria, que los criterios emitidos en el presente Trabajo de Integración Curricular, titulado: **"Automatización de procesos de integración de código, compilación, pruebas y despliegue para proyectos de software en el Ecuador",** previo a la obtención del título profesional de **ingeniería de Software**, así como también los contenidos, ideas, análisis, conclusiones y propuestas son exclusiva responsabilidad de nosotros, como autores.

2. Declaramos, igualmente, tener pleno conocimiento de la obligación que tiene la Universidad Iberoamericana del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT, en formato digital una copia del referido Trabajo de Integración Curricular para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública, respetando los derechos de autor.

3. Autorizamos, finalmente, a la Universidad Iberoamericana del Ecuador a difundir a través del sitio web de la Biblioteca de la UNIB.E (Repositorio Digital Institucional), el referido Trabajo de Integración Curricular, respetando las políticas de propiedad intelectual de la Universidad Iberoamericana del Ecuador.

Quito, DM., a los 21 días del mes de febrero de 2024.

Firmado electrónicamente por:
JOSE RODOLFO OBANDO MALDONADO

José Obando

1718552365

Firmado electrónicamente por:
ALEXANDER PATRICIO MEJIA ARIAS

Alexander Mejía

1750050211

Campus Colón : 9 de Octubre N25-12 y Av. Colón, Quito 170522.
Campus Norte : José Queri y Av. Eloy Alfaro, Quito 170513
www.unibe.edu.ec

AUTORIZACIÓN DE PRESENTACIÓN FINAL DEL TRABAJO DE INTEGRACIÓN CURRICULAR POR PARTE DEL TUTOR

PhD. Alicia Elizundia

Decana de la Facultad de Comunicación y Tecnologías

Presente. -

Yo, **HARRY ALBERTO CARPIO SALVATIERRA, MSc,** Tutor del Trabajo de Integración Curricular realizado por los estudiantes **JOSÉ RODOLFO OBANDO MALDONADO y ALEXANDER PATRICIO MEJÍA ARIAS** de la carrera de **INGENIERÍA DE SOFTWARE** informo haber revisado el presente documento titulado **AUTOMATIZACIÓN DE PROCESOS DE INTEGRACIÓN DE CODIGO, COMPILACIÓN, PRUEBAS Y DESPLIEGUE PARA PROYECTOS DE SOFTWARE EN EL ECUADOR**, el mismo que se encuentra elaborado conforme a lo establecido en el Reglamento de Titulación y el Manual de Estilo de la Universidad Iberoamericana del Ecuador, UNIB.E de Quito, por lo tanto, autorizo la entrega del Trabajo de Integración Curricular a la Unidad de Titulación para la presentación final ante el tribunal evaluador.

Firmado electrónicamente por:
HARRY ALBERTO CARPIO SALVATIERRA

Atentamente,

Harry Carpio

Tutor

Campus Colón : 9 de Octubre N25-12 y Av. Colón, Quito 170522.
Campus Norte : José Queri y Av. Eloy Alfaro, Quito 170513
www.unibe.edu.ec

ACTA DE APROBACIÓN DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Facultad: Comunicación y Tecnologías

Carrera: Ingeniería de Software

Modalidad: Semipresencial

Nivel: 3er nivel de Grado

En el Distrito Metropolitano de Quito a los veinte días del mes de marzo del 2024 (20-03-2024) a las nueve horas con treinta minutos (09:30), ante el Tribunal de Presentación Oral, se presentó el señor: **MEJIA ARIAS ALEXANDER PATRICIO**, titular de la cédula de ciudadanía No. **1750050211** a rendir la evaluación oral del Trabajo de Integración Curricular: "**Automatización del proceso para proyectos de software mediante la implementación de integración continua y despliegue continuo CI/CD usando la metodología scrum y basado en la cultura DevOps.**", previo a la obtención del Título de Ingeniero de Software. Luego de la exposición, el referido estudiante obtiene las calificaciones que a continuación se detallan:

	Calificación
Lectura del Trabajo de Integración Curricular	9 /10
Evaluación Oral del Trabajo de Integración Curricular	9.6 /10
Calificación Final del Trabajo de Integración Curricular	9.4 /10

Para constancia de lo actuado, los miembros del Tribunal de Presentación Oral del Trabajo de Integración Curricular, firman el presente documento en unidad de acto, a los veinte días del mes de marzo del 2024 (20-03-2024).

PhD. Jesús Gómez
VICERRECTOR

UNIB.E DIRECCIÓN ACADÉMICA

PhD. Luisa Taborda
DIRECTOR ACADEMICO

Mgst. Harry Carpio
TUTOR

UNIB.E SECRETARIA DE ADMINISTRACIÓN Y REGISTRO

Mgst. Tonysé de la Rosa
LECTOR

ACTA DE APROBACIÓN DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Facultad: Comunicación y Tecnologías

Carrera: Ingeniería de Software

Modalidad: Semipresencial

Nivel: 3er nivel de Grado

En el Distrito Metropolitano de Quito a los veinte días del mes de marzo del 2024 (20-03-2024) a las nueve horas con treinta minutos (09:30), ante el Tribunal de Presentación Oral, se presentó el señor: **OBANDO MALDONADO JOSÉ RODOLFO**, titular de la cédula de ciudadanía No. **1718552365** a rendir la evaluación oral del Trabajo de Integración Curricular: **"Automatización del proceso para proyectos de software mediante la implementación de integración continua y despliegue continuo CI/CD usando la metodología scrum y basado en la cultura DevOps."**, previo a la obtención del Título de Ingeniero de Software. Luego de la exposición, el referido estudiante obtiene las calificaciones que a continuación se detallan:

	Calificación
Lectura del Trabajo de Integración Curricular	9 /10
Evaluación Oral del Trabajo de Integración Curricular	9.6 /10
Calificación Final del Trabajo de Integración Curricular	9.4 /10

Para constancia de lo actuado, los miembros del Tribunal de Presentación Oral del Trabajo de Integración Curricular, firman el presente documento en unidad de acto, a los veinte días del mes de marzo del 2024 (20-03-2024).

PhD. Jesús Gómez
VICERRECTOR

UNIB.E DIRECCIÓN ACADÉMICA

PhD. Luisa Taborda
DIRECTOR ACADEMICO

Mgst. Harry Carpio
TUTOR

UNIB.E SECRETARÍA DE ADMISIÓN Y REGISTRO

Mgst. Tonysé de la Rosa
LECTOR

INDICE GENERAL

INDICE DE TABLAS

INDICE DE FIGURAS

Alexander Mejía, José Obando. ***AUTOMATIZACIÓN DE PROCESOS DE INTEGRACIÓN DE CODIGO, COMPILACIÓN, PRUEBAS Y DESPLIEGUE PARA PROYECTOS DE SOFTWARE EN EL ECUADOR.*** Carrera de Ingeniería en Software. Quito Ecuador.2024. (92) pp.

RESUMEN

La automatización del proceso del ciclo de vida de proyectos de software a través de la implementación de CI/CD es un área de investigación y aplicación crítica en el desarrollo de software moderno.

El presente documento muestra como mediante la automatización de los procesos de código, integración, pruebas y despliegue de proyectos de software se logra mejorar la eficiencia y calidad del proceso de entrega de software.

En la primera parte, se examinarán los problemas, objetivos, justificaciones y alcances.

En la segunda parte, se resaltan los antecedentes, de la Integración Continua y el Despliegue Continuo, destacando su relevancia y beneficios en el desarrollo de software.

En el tercer capítulo, se delimitará el uso de la metodología empleada, se profundizará en la metodología Scrum y su aplicación en el desarrollo de software, con énfasis en cómo se puede combinar con CI/CD para impulsar la entrega continua de software y la mejora iterativa, mostrando también las técnicas utilizadas y el producto final.

En el cuarto capitulo, se mostrará el análisis de los resultados de la investigación y la eficacia del uso de la integración continua y despliegue continuo.

En el capítulo final, se mostrarán los hallazgos y reflexiones del presente trabajo.

Finalmente, se presenta un estudio de caso en el que se aplica la metodología Scrum y se adopta la cultura DevOps, enfocándose en la automatización del proceso a través de CI/CD en un equipo de desarrollo de software.

Se evalúan los resultados obtenidos, incluyendo mejoras en la productividad y la calidad del software, lo que se busca es mostrar cómo la automatización del proceso para proyectos de software implementando CI/CD, basada en la metodología Scrum y la cultura DevOps, puede llevar a un desarrollo más eficiente, una entrega continua y una mejora constante en el desarrollo de software actualmente.

Palabras claves: Devops, Jenkins, Integración Continua, Despliegue Continuo, GIT.

Alexander Mejía, José Obando. ***AUTOMATION OF CODE INTEGRATION, COMPILATION, TESTING AND DEPLOYMENT PROCESSES FOR SOFTWARE PROJECTS IN ECUADOR.*** Software Engineering Program. Quito Ecuador.2024. (92) pp.

ABSTRACT

The automation of the software project life cycle process through the implementation of Continuous Integration and Continuous Deployment (CI/CD) is a critical area of research and application in modern software development. This document demonstrates how automating the processes of code development, integration, testing, and software deployment improves the efficiency and quality of the software delivery process.

In the first section, we will examine the issues, objectives, justifications, and scopes of implementing CI/CD in software projects.

The second section highlights the background of Continuous Integration and Continuous Deployment, emphasizing their relevance and benefits in software development.

In the third chapter, the use of the employed methodology will be defined, delving into the Scrum methodology and its application in software development. Emphasis will be placed on how it can be combined with CI/CD to drive continuous software delivery and iterative improvement, showcasing the techniques used and the final product.

The fourth chapter presents the analysis of the research results and the effectiveness of continuous integration and continuous deployment.

In the final chapter, the findings and reflections of this work are presented. Finally, a case study is provided, implementing the Scrum methodology and DevOps culture, focusing on process automation through CI/CD in a software development team. The obtained results are evaluated, including improvements in productivity and software quality. The goal is to demonstrate how automating the software project process through CI/CD, based on the Scrum methodology and DevOps culture, can lead to more efficient development, continuous delivery, and constant improvement in software development today.

Keywords: Devops, Jenkins, Integración Continua, Despliegue Continuo, GIT.

INTRODUCCIÓN

En la era digital actual, la industria del desarrollo de software enfrenta desafíos constantes para garantizar la entrega de productos de alta calidad en plazos ajustados. La automatización del proceso para proyectos de software ha surgido como una solución eficaz para superar estas dificultades, al permitir una integración y despliegue más fluidos y consistentes. En este contexto, la implementación de Integración Continua y Despliegue Continuo (CI/CD) se ha convertido en una práctica esencial para mejorar la eficiencia, reducir errores y garantizar una entrega continua y rápida de software.

En esta tesis, se abordará la automatización de procesos de integración de código, compilación, pruebas y despliegue para proyectos de software mediante la implementación de CI/CD, centrándose en la metodología Scrum y basándose en la cultura DevOps. Estas dos últimas, Scrum y DevOps, son enfoques ampliamente reconocidos y utilizados en la industria para fomentar la colaboración, la agilidad y la mejora continua en los equipos de desarrollo.

La presente investigación muestra en cada capítulo el siguiente contenido:

Capítulo I: destaca la importancia de la Integración Continua (CI) y el Despliegue Continuo (CD) en el desarrollo de software. Se enfoca en la automatización de procesos clave, como la integración de código, compilación, pruebas y despliegue, a través de la implementación eficaz de prácticas de CI/CD. El objetivo es mejorar la eficiencia de los desarrolladores, permitiendo adaptarse a cambios, reducir riesgos y ofrecer valor continuo a los usuarios.

Capítulo II: resalta la relevancia de la Integración Continua (CI) y el Despliegue Continuo (CD) en el desarrollo de software para garantizar eficacia y satisfacción del usuario. Se centra en la automatización de procesos como la integración de código, compilación, pruebas y despliegue, mediante la implementación de prácticas sólidas de CI/CD. Este enfoque teórico busca proporcionar un contexto para la mejora de la eficiencia del desarrollador, facilitando la adaptación a cambios, la reducción de riesgos y la entrega continua de valor a los usuarios. La pregunta fundamental en este contexto teórico es cómo llevar a cabo la automatización de estos procesos en proyectos de software mediante CI/CD.

Capítulo III: en este capítulo de metodología, se presenta el diseño de investigación elegido y se justifica su elección para el proyecto. Se adopta un enfoque cuantitativo basado en el paradigma positivista, que busca la objetividad y se apoya en la

traducción de información a números. El diseño de investigación es no experimental y transversal, evaluando fenómenos en su entorno natural. La población se compone de documentos, como artículos académicos, libros, informes de la industria y documentos técnicos, con una antigüedad no superior a 5 años. La muestra se selecciona con criterios de inclusión relacionados con casos de estudio detallados sobre implementaciones exitosas de CI/CD, Scrum y DevOps en proyectos de software. Se revisan 20 documentos, seleccionando 7 que cumplen con los requisitos. La recolección de datos se realiza mediante investigación documental, utilizando la matriz de análisis como herramienta estructurada para organizar y categorizar el contenido.

Capítulo IV: se aborda el desarrollo de la propuesta en cuestión. En este segmento, se detallan los elementos fundamentales relacionados con la creación y evolución de la propuesta, ofreciendo información sobre la base teórica, la metodología empleada, los recursos utilizados y otros aspectos pertinentes para comprender y ejecutar la propuesta.

Capítulo V: se abordan las conclusiones y recomendaciones

CAPÍTULO I
EL PROBLEMA

En este capítulo se expone el problema de investigación relacionado con la integración continua y el despliegue continuo, adoptados en el desarrollo de software, con el objetivo de mejorar la eficiencia y calidad del proceso de entrega. En este contexto, las organizaciones en un entorno competitivo requieren una entrega rápida y fiable de aplicaciones de alta calidad. Para abordar esta necesidad, se han desarrollado prácticas y enfoques de Integración Continua (CI) y Despliegue Continuo (CD).

La integración y el despliegue continuos son procesos automatizados que buscan garantizar la estabilidad de los proyectos de software mediante la integración periódica del código fuente y la entrega rápida de las aplicaciones. Estos procesos posibilitan la detección temprana y la corrección de cualquier problema potencial, permitiendo una transición fluida del desarrollo.

Adoptar y aprovechar CI/CD en los proyectos de software puede resultar altamente beneficioso, ya que ofrece mejoras en la calidad del software, una entrega más rápida y una reducción del riesgo. Sin embargo, la implementación de estas prácticas no siempre es sencilla, y muchas organizaciones se enfrentan a desafíos significativos al intentarlo (Humble & Farley, 2010).

Implantar un marco eficaz de CI/CD puede ser un desafío sin un enfoque estructurado. Las organizaciones pueden encontrarse con dificultades para establecer el entorno de desarrollo adecuado, automatizar herramientas, definir estrategias de pruebas automatizadas y crear flujos de trabajo eficientes. Aspectos como la infraestructura de alojamiento, la gestión de la versión del código fuente, la gestión de la configuración y la supervisión del rendimiento de las aplicaciones desplegadas pueden complicar aún más la implementación. Además, la falta de conocimiento y experiencia en CI/CD puede constituir un obstáculo significativo para las organizaciones. La adopción exitosa de estas prácticas requiere un cambio cultural y una comprensión profunda de los conceptos, técnicas y herramientas asociadas con CI/CD. La falta de recursos especializados y la resistencia al cambio pueden dificultar aún más la adopción de CI/CD (Farcic, 2016).

Por lo tanto, resulta esencial crear una estructura fiable que proporcione reglas precisas, técnicas óptimas y métodos recomendados para ejecutar CI/CD en proyectos de software. Este marco contribuirá a que las empresas superen los problemas y desafíos cotidianos, simplificando la adopción efectiva de CI/CD y acelerando la entrega de aplicaciones de software de calidad.

Planteamiento del problema

Para seguir siendo competitivos y satisfacer las demandas de los usuarios, es esencial que los desarrolladores de software entreguen aplicaciones de alta calidad de forma rápida y fiable. Para conseguirlo, cada vez es más común la implantación de prácticas de Integración Continua (CI) y Despliegue Continuo (CD).

La Integración Continua garantiza que los cambios de código se integren de forma regular y automática (Fowler ,2006), permitiendo la detección temprana de cualquier problema y asegurando la correcta fusión del código.

Por otro lado, el Despliegue Continuo automatiza el proceso de despliegue del software en entornos de producción, proporcionando una entrega rápida y fiable de las aplicaciones a los usuarios finales.

El presente trabajo se centra en automatizar procesos de integración de código, compilación, pruebas y despliegue mediante la implantación eficaz de la Integración Continua y el Despliegue Continuo en proyectos de software.

Mediante el desarrollo de dicha práctica, los desarrolladores de software pueden garantizar que la CI/CD se implanta de forma eficaz, lo que conduce a ciclos más rápidos y a una mejora de la calidad del producto.

Este trabajo proporcionará orientación sobre la creación de un entorno de desarrollo adecuado, la configuración de herramientas de automatización, la definición de estrategias de pruebas automatizadas y el establecimiento de flujos de trabajo eficientes. Además, se tratarán temas relacionados con la infraestructura de alojamiento, el versionado del código fuente, la gestión de la configuración y la supervisión del rendimiento de las aplicaciones desplegadas.

El presente trabajo pretende que los desarrolladores sean más eficientes y rápidos en la entrega de software, promoviendo la adopción de prácticas de CI/CD. La adopción permitirá a las organizaciones seguir el ritmo de los cambios sin esfuerzo, reducir el riesgo y proporcionar valor continuo a sus usuarios.

Derivado de lo antes expuesto se plantea la siguiente interrogante:
¿Cómo automatizar el proceso de integración de código, compilación, pruebas y despliegue mediante CI/CD en proyectos de software?

Objetivos de la investigación

Objetivo general

- Automatizar procesos de integración de código, compilación, pruebas y despliegue para proyectos de software mediante la Integración Continua (CI) y el Despliegue Continuo (CD) usando la metodología scrum y basado en la cultura DevOps en el Ecuador.

Objetivos específicos

- Investigar las mejores prácticas, metodologías y herramientas disponibles en el ámbito de la Integración Continua y el Despliegue Continuo a través de una revisión documental.
- Desarrollar un caso de estudio que ilustre la aplicación efectiva de CI/CD en contexto de desarrollo de software.
- Aplicar los componentes fundamentales de CI/CD, como la configuración de herramientas, las pruebas automatizadas, los flujos de trabajo y la gestión de configuraciones.

Justificación e impacto de la investigación

La implantación de la Integración Continua y el Despliegue Continuo en los proyectos de software es esencial para que las organizaciones sigan siendo competitivas. Con estas prácticas, los desarrolladores garantizan que sus aplicaciones se prueben, desplieguen y actualicen regularmente de forma rápida y eficaz, lo que lleva a una mejora de la calidad del producto y a una mayor satisfacción del cliente.
Además, el CI/CD ayuda a reducir los riesgos asociados a los procesos manuales, automatizando tareas como las compilaciones, las pruebas, las revisiones de código y las implantaciones. Como resultado de este proceso de automatización, las organizaciones se benefician de unos ciclos de desarrollo más cortos, así como de una comercialización más rápida de las nuevas funciones y productos. En última instancia, la adopción de CI/CD ayuda a las organizaciones a ofrecer mejores experiencias a sus clientes, ahorrando tiempo y dinero a largo plazo.

Alcance de la investigación

El trabajo actual se enfoca en automatizar el proceso del ciclo de vida de proyectos de software mediante la implementación efectiva de Integración Continua y Despliegue Continuo.

Los desarrolladores de software pueden asegurarse de que la CI/CD se implante de manera efectiva mediante el desarrollo de esta práctica, lo que resulta en ciclos más rápidos y una mejora de la calidad del producto.

La creación de un entorno de desarrollo adecuado, la configuración de herramientas de automatización, la creación de estrategias de pruebas automatizadas y la creación de flujos de trabajo eficientes son todos temas que este trabajo proporcionará orientación. Además, se abordarán temas como la infraestructura de alojamiento, el versionado del código fuente, la gestión de la configuración y la supervisión del rendimiento de las aplicaciones desplegadas también serán abordados, todo esto mencionado de manera local en un sistema operativo Windows con las herramientas necesarias para su correcto funcionamiento, además de denotar que únicamente se automatizaran los procesos de integración de código, compilación, pruebas y despliegue para proyectos de software.

CAPÍTULO II
MARCO TEÓRICO

El siguiente capítulo presenta el marco teórico, en este sentido se analizarán fuentes bibliográficas, que permitan obtener conceptos relevantes que ayuden a profundizar y estudiar el tema propuesto en el proyecto donde se mostrara antecedentes y bases teóricas.

Antecedentes de la investigación

La automatización del proceso para proyectos de software mediante la implementación de integración continua y despliegue continuo (CI/CD) con la metodología Scrum y basada en la cultura DevOps crece en la industria del desarrollo de software. A continuación, se presentan algunos antecedentes relevantes sobre este tema:

De acuerdo con Morocho y León (2021), con su trabajo de investigación cuyo tema es: “Implementación de una arquitectura de integración y entrega continúa basada en contenedores y buenas prácticas DevOps en entornos de desarrollo web para la empresa Emagic S.A”, se centra en abordar dos factores críticos que afectan la calidad del software en el entorno actual: la complejidad de los requisitos y las prácticas de la industria del software. Para abordar esta problemática, se define un conjunto de procesos y un flujo de trabajo que se basan en la implementación de una arquitectura de contenedores y buenas prácticas DevOps. Además, se detalla cómo se incorporan diversas herramientas de integración y entrega continua para cumplir con todo el ciclo de desarrollo DevOps. El análisis del entorno de desarrollo del servidor de automatización permitió comprender la funcionalidad y el propósito de cada uno de los elementos involucrados en el proceso. Se identificó la capacidad de gestionar usuarios, mostrar tareas y procesos, y la posibilidad de ampliar las funcionalidades de Jenkins a través de complementos para una optimización más efectiva del sistema. Estos hallazgos proporcionan un contexto importante y constituyen un precedente valioso para comprender el entorno y las capacidades del servidor de automatización, lo que facilitará la toma de decisiones y la implementación de mejoras en el futuro.

Bases teóricas

DevOps

DevOps es un marco de trabajo y una filosofía en constante evolución que promueve un mejor desarrollo de aplicaciones en menos tiempo y la rápida publicación de, nuevas o revisadas funciones de software o productos para los clientes.
Se promueve una comunicación continua más fluida, la colaboración, la integración, la visibilidad y la transparencia entre equipos de desarrollo de aplicaciones (Dev) y sus homólogos en operaciones tecnológicas (Ops), según (NetApp, 2023).

Integración Continua

La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. La integración continua se refiere en su mayoría a la fase de creación o integración del proceso de publicación de software y conlleva un componente de automatización (p. ej., CI o servicio de versiones) y un componente cultural (p. ej., aprender a integrar con frecuencia).
Los objetivos clave de la integración continua consisten en encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones de software, según (Amazon Web Services, Inc., 2023)

Despliegue continuo

El despliegue continuo es una estrategia de desarrollo de software en la que los cambios de código de una aplicación se publican automáticamente en el entorno de producción. Esta automatización se basa en una serie de pruebas predefinidas. Una vez que las nuevas actualizaciones pasan esas pruebas, el sistema envía las actualizaciones directamente a los usuarios del software.
El despliegue continuo ofrece varias ventajas para las empresas que quieren escalar su porfolio de aplicaciones y TI.

En primer lugar, agiliza el tiempo de comercialización al eliminar el desfase entre la programación y el valor al cliente, que suele ser de días, semanas o incluso meses. (IBM, 2023)

Metodología Scrum

Scrum es un marco de trabajo ágil para la gestión de proyectos y desarrollo de productos. Se basa en la iteración y entrega incremental, dividiendo el trabajo en unidades llamadas sprints, que son periodos de tiempo predefinidos (generalmente de 1 a 4 semanas).

Roles

Scrum proporciona roles definidos los cuales son los siguientes:

- **Scrum Master**

  Es fundamental para la implementación exitosa de Scrum, su responsabilidad principal es eliminar obstáculos que puedan afectar el progreso del equipo y promover una cultura de mejora continua.

  Trabaja en estrecha colaboración con el Product Owner para asegurarse de que los objetivos del producto coincidan con la visión del equipo, así como con el Equipo de Desarrollo para ayudarlo a organizarse y resolver conflictos.

  La relación entre ambos roles es fundamental para mantener un entorno de trabajo productivo y enfocado en la entrega de valor.

- **Product Owner**

  Tiene la responsabilidad de definir y priorizar el backlog del producto y ocupa una posición central en Scrum.

  Tomando decisiones informadas sobre qué funcionalidades deben incluirse en cada incremento del producto, actúa como el vínculo clave entre el equipo de desarrollo y los interesados.

  Colabora estrechamente con el Scrum Master para garantizar que el proceso funcione bien y para resolver cualquier problema.

La interacción cercana con el equipo de desarrollo garantiza que la visión del producto se comprenda y se traduzca en entregas valiosas.

- **El Equipo de Desarrollo**

Constituye la fuente principal de incrementos de funcionalidad en cada Sprint.
El equipo trabaja en colaboración estrecha con el Product Owner para comprender y desarrollar las historias de usuario, que son autoorganizadas y multifuncionales.
Su participación activa en todas las ceremonias Scrum garantiza la transparencia y la colaboración durante el ciclo de vida del desarrollo.
La relación con el Scrum Master se centra en superar desafíos y mejorar continuamente los procesos y la eficiencia del equipo, contribuyendo así al éxito general del proyecto Scrum.

- **Ceremonias clave**

Scrum proporciona ceremonias, las cuales son las siguientes:

- **Reunión de Planificación o Planning**

El objetivo de la planificación del sprint es establecer el qué y el cómo se va a hacer, al inicio de cada Sprint, se realiza esta reunión para determinar cómo se enfocará el proyecto a partir de las etapas y plazos del Product Backlog.

- **Reunión Diaria o Daily**

El objetivo de la Daily es agilizar la comunicación, principalmente nos permite detectar los obstáculos que impiden el avance del desarrollo, ayudan a la toma de decisiones que favorezcan el cumplimiento de los tiempos y cumplir los objetivos.
Tres preguntas que se responden individualmente: ¿Qué hice ayer?, ¿Qué planeo hacer hoy?, ¿Qué tipo de ayuda necesito?, el Scrum Master debe esforzarse por resolver los problemas.

- **Revisión o Review**

El objetivo de la review del sprint, la cual se realiza al finalizar el mismo, es en la que se presenta el desarrollo finalizado y los asistentes proponen nuevas cosas potenciando así la colaboración entre todos.

- **Retrospectiva**

El equipo revisa los objetivos cumplidos del sprint terminado y anota los errores para evitar cometerlos nuevamente.

Esta etapa es necesaria para implementar mejoras desde la perspectiva del proceso de desarrollo.

El objetivo de la retrospectiva es identificar las mejoras potenciales del proceso y desarrollar un plan para implementarlas en la siguiente iteración.

Se muestra a continuación una imagen con todos los eventos:

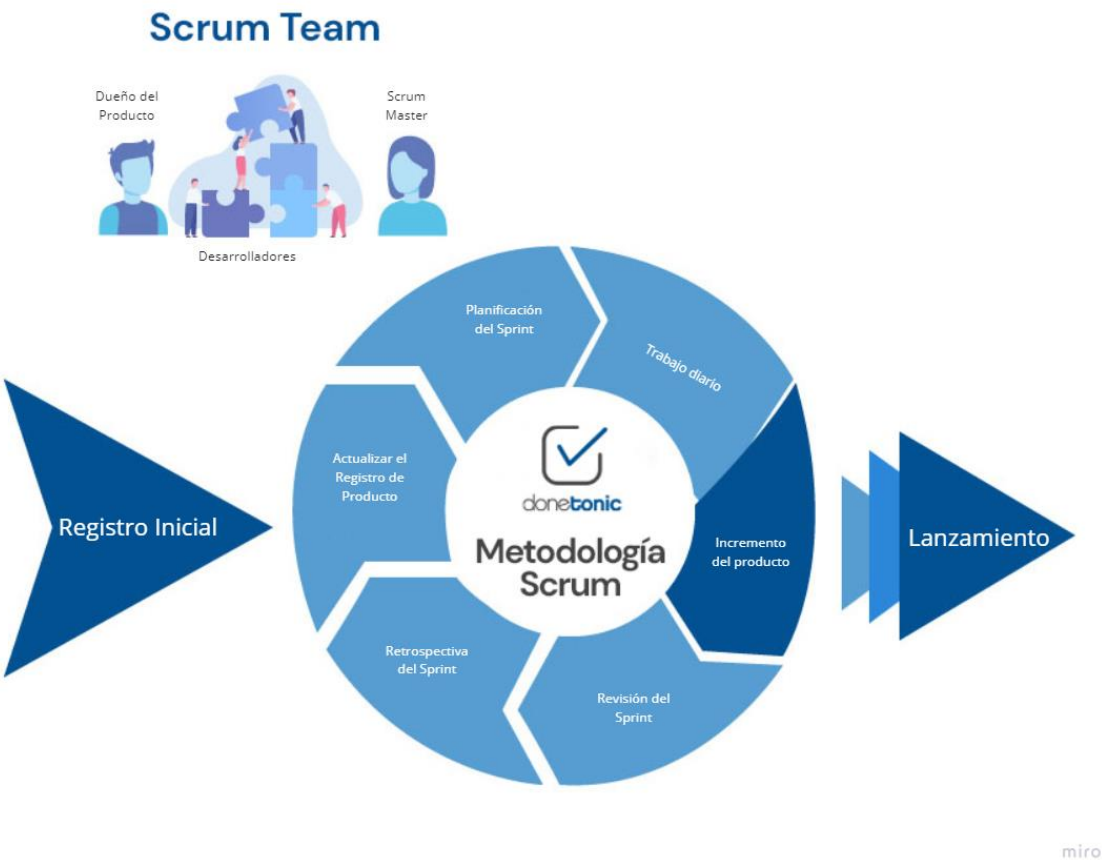

Figura 1. *Eventos Scrum (Donetonic, 2023)*

Se puede argumentar que Scrum se centra en la adaptabilidad, la colaboración y la entrega de valor continuo, la combinación de CI/CD, Scrum y DevOps

permite a los equipos lograr una entrega de software rápida, predecible y de alta calidad.

La Integración Continua y el Despliegue Continuo aseguran que los cambios en el código se prueben y desplieguen automáticamente, facilitando la entrega continua de nuevas características.

Scrum proporciona un marco ágil que se integra bien con la entrega continua, permitiendo la adaptabilidad a los cambios en los requisitos del cliente.

DevOps, al centrarse en la colaboración y la automatización, mejora la eficiencia del desarrollo y la operación conjunta.

En conjunto, estas prácticas y marcos de trabajo permiten a los equipos alcanzar una mayor velocidad de entrega, mejor calidad del software, y una capacidad para responder de manera ágil a las necesidades cambiantes del negocio.

CAPÍTULO III

MARCO METODOLOGICO

En este capítulo, se describirá el diseño de investigación seleccionado, justificando el por qué es la mejor opción para el proyecto.

También se analizará detalladamente las herramientas y estrategias que serán utilizadas para recolectar y analizar los datos relevantes.

Naturaleza de la investigación

El enfoque propuesto es cuantitativo y se fundamenta en el paradigma positivista, el cuál según Palella y Martins (2012), asume la objetividad como única vía para alcanzar el conocimiento, enfatiza que la información se puede traducir en números, busca explicar, predecir y controlar los fenómenos, así como verificar teorías y fundamenta el análisis en la estadística descriptiva e inferencial.

Diseño de investigación

El diseño propuesto es el no experimental, el cual contempla aquellos estudios que se realizan sin la manipulación de la variable independiente, debido a que ya ha sucedido y en los que solo se observan los fenómenos en su ambiente natural. (Solis, 2019).

Dentro de esta clasificación, se ha optado por el diseño transversal, el cual posibilita la evaluación de una situación, comunidad, evento, fenómeno o contexto en un punto específico del tiempo. Esta elección se alinea con los objetivos del caso de estudio, permitiendo la validación de las distintas fases del proyecto.

En cuanto al tipo de investigación aplicado, se ha seleccionado el enfoque documental. Este método posibilita el análisis de información existente, contribuyendo a la comprensión y evaluación de la situación estudiada.

Población y muestra

Población

Para obtener la población, se consideró los siguientes tipos de documentos, para asegurar que la información contenida en ellos tenga una fecha de publicación no superior a 5 años de antigüedad:

- **Artículos académicos:** Investigaciones publicadas en revistas científicas que abordan aspectos específicos de CI/CD, Scrum y DevOps en proyectos de software.
- **Libros:** Textos académicos y profesionales que proporcionen una visión integral de la implementación de CI/CD, Scrum y DevOps.
- **Informes de la Industria:** Documentos de organizaciones y empresas de tecnología que presentan análisis de tendencias, estadísticas y buenas prácticas en la automatización del proceso.
- **Guías y Documentos Técnicos:** Manuales, guías prácticas y documentos técnicos que ofrecen instrucciones detalladas sobre la aplicación de CI/CD, Scrum y DevOps.

Muestra

Criterios de inclusión

- Documentos que proporcionen casos de estudio detallados sobre implementaciones exitosas de CI/CD, Scrum y DevOps en proyectos de software.
- Mejores prácticas y estrategias para la automatización del proceso.
- Desafíos específicos enfrentados durante la implementación y soluciones propuestas.
- Documentos publicados desde 2018.

Criterios de exclusión

- Documentos que no estén directamente relacionados con la automatización del proceso en el desarrollo de software.
- Documentos que tenga fechas menores que el 2018.

Se han revisado 20 documentos en total y de estos, se han identificado y seleccionado 7 que cumplen con los criterios de inclusión establecidos.

Estos documentos han sido elegidos para proporcionar casos de estudio detallados sobre implementaciones exitosas de Continuous Integration/Continuous Deployment (CI/CD), Scrum y DevOps en proyectos de software. Además, abordan de manera significativa las mejores prácticas y estrategias para la automatización del proceso, así como los desafíos específicos enfrentados durante la implementación, ofreciendo soluciones propuestas.

Esta selección representa una muestra sólida que aborda de manera comprehensiva los aspectos clave de interés en el ámbito de desarrollo de software.

Técnicas e instrumentos de recolección de datos

El tipo de investigación es documental, la cual se basa en “el análisis, la búsqueda, crítica e interpretación de datos obtenidos y registrados en fuentes documentales impresas, audiovisuales o electrónicas” (Arias, 2016).

Una parte importante de nuestro proyecto de investigación es la revisión de documentos, que implica la búsqueda, recopilación y análisis crítico de documentos relacionados con nuestro campo de estudio. Los documentos provienen de una variedad de fuentes, incluidos libros, artículos, informes, tesis, patentes, registros oficiales, medios de comunicación y cualquier otro material impreso o digital que sea relevante para la investigación.

El instrumento a utilizar es la matriz de análisis, es una herramienta estructurada que nos ayuda a categorizar y organizar el contenido que se quiere examinar.

Metodología del producto

Planificación del Proyecto de Software

En el siguiente apartado se detalla la planificación, desarrollo y recursos utilizados para el presente trabajo de titulación, con sus respectivas metodologías y especificaciones.

Recursos

Tabla 1 *Recursos*

Recurso	Detalle
Hardware	• **Ordenador portátil** Intel Core i9 13th generación – 32GB RAM.
Software	• **Docker 24.0.7** Es una plataforma de contenerización que encapsula aplicaciones en contenedores y garantiza su compatibilidad en diferentes entornos, lo que simplifica el despliegue de aplicaciones. • **Kubernetes (Minikube)** Actúa como orquestador de contenedores, automatizando el despliegue y la escalabilidad de aplicaciones contenerizadas para garantizar su disponibilidad y gestión eficiente. • **Jenkins** Es una herramienta de automatización de CI/CD que mejora la eficiencia del desarrollo al facilitar la construcción, prueba y despliegue automatizado de aplicaciones. • **Visual Studio Code** Es un editor de código fuente que facilita la escritura y edición de código, ofreciendo extensiones para una variedad de tecnologías y aumentando la productividad del desarrollador. • **GIT** Es un sistema de control de versiones distribuido, y GitHub es una plataforma para alojar y colaborar con proyectos basados en GIT, lo que facilita la colaboración y el seguimiento de problemas. • **GitHub, GitHub Webhooks** Permite la integración con CI/CD al recibir notificaciones de eventos particulares en repositorios GitHub, lo que provoca flujos de trabajo automáticos.

Recurso	Detalle
	• **Docker Hub** Es un registro de imágenes de Docker que almacena y comparte imágenes de contenedores, lo que facilita su distribución. • **Slack** Es una plataforma que se integra con otras herramientas para mejorar la coordinación y la comunicación entre los miembros del equipo. • **Notion** Es una plataforma que se integra con otras herramientas para mejorar la coordinación y la comunicación entre los miembros del equipo. • **Grafana** Es una plataforma de análisis y visualización de datos, • **Prometheus** Es un sistema de monitorización y alerta que recopila métricas de sistemas y aplicaciones, proporciona paneles interactivos y genera alertas. • **NgRok** Ayuda a las pruebas y demostraciones de aplicaciones locales al exponer los servidores locales al mundo exterior. • **SonarQube** Es una plataforma de análisis de código que evalúa la calidad del código para encontrar problemas y vulnerabilidades potenciales. • **Java, Gradle, Node** Son tecnologías que se utilizan para crear aplicaciones empresariales y del lado del servidor. • **Angular CLI** Es una interfaz de línea de comandos que facilita el desarrollo, la construcción y la prueba de aplicaciones Angular.

Recurso	Detalle
	• **Kubectl** Es una herramienta de línea de comandos que facilita la gestión y operación de clústeres Kubernetes. • **Karma 6.4.1, Jasmine 3.6.0, Junit 5** Son frameworks utilizados para realizar pruebas unitarias y de integración, garantizando la calidad del código. • **Miro** Es una plataforma de colaboración en línea que permite la visualización y la colaboración de ideas y proyectos mediante pizarras virtuales. • **H2** Es un sistema de gestión de bases de datos relacionales en memoria que se utiliza para desarrollar y probar aplicaciones de bases de datos SQL.

Estimación del proyecto

El proyecto se planificó utilizando la técnica de estimación de Fibonacci, la misma que asigna valores de secuencia de Fibonacci a las tareas para representar su complejidad y esfuerzo asociados. Cada tarea fue evaluada y se le asignó un valor de la secuencia de Fibonacci que mejor reflejara su magnitud. La secuencia de Fibonacci consta de números como 1, 2, 3, 5, 8, 13 y así sucesivamente, permite una representación no lineal del esfuerzo, destacando las diferencias notables entre las tareas.

La estimación de las tareas mediante la secuencia de Fibonacci proporcionó una visión clara de la complejidad y el esfuerzo requerido a lo largo del proyecto. Además, facilitó la planificación efectiva y una ejecución eficiente, asegurando una implementación exitosa.

A continuación, se muestra los resultados obtenidos de la estimación para el presente proyecto.

Tabla 2. *Estimación del proyecto*

Sprint	Historia	Estimación (Fibonacci)
1	Configuración del entorno de desarrollo local	5
2	Instalación de Docker y demás herramientas necesarias	5
3	Instalación y configuración del clúster	8
4	Instalación y configuración de Jenkins	8
5	Instalación y configuración de Sonarqube	8
6	Desarrollo de proyecto frontend	8
7	Desarrollo de proyecto backend	8
8	Agregar Slack (Notificaciones)	3
9	Desarrollo de pipeline para integración continua (CI)	13
10	Desarrollo de pipeline para despliegue continuo (CD)	13
11	Agregar Prometeus y grafana (Métricas)	8
	Puntuación total	71

Ruta del proyecto

Para mostrar la ruta del proyecto, hemos desarrollado un diagrama de Gantt donde se muestra la planificación. El diagrama toma en cuenta todas las tareas que se deben cumplir en las fechas establecidas, como se muestra en el Anexo 1.

Herramientas de gestión del proyecto

Como herramienta de gestión de proyecto se utilizó Notion debido a su versatilidad al usar las metodologías agiles como SCRUM, además esta herramienta proporciona una interfaz intuitiva que facilita la creación de tareas y organización de documentos. Por último, Notion ofrece gran flexibilidad al momento de colaborar entre varios miembros de un equipo lo que se resulta en un ahorro de tiempo.

Herramientas de gestión de versiones

Para el apartado de herramientas de gestión de versiones se utilizó Git y GitHub. Estás dos son elementos son clave para garantizar el control de versiones.

Git

Es un sistema de control de versiones distribuido que permite el seguimiento de cambios en el código fuente durante el desarrollo de software y gestiona los repositorios localmente en la máquina.

GitHub

Es una plataforma basada en la web que utiliza Git para el control de versiones. GitHub proporciona un espacio en línea donde los desarrolladores pueden almacenar, colaborar y gestionar proyectos de software utilizando Git.

Análisis y Diseño

Visión y alcance

En esta sección se detalla la visión del proceso de automatización para proyectos de software, centrándose en la implementación de Integración Continua y Despliegue Continuo (CI/CD) usando la metodología Scrum y basándose en la cultura DevOps. Asimismo, se establecen los límites y el alcance de esta automatización para proporcionar una comprensión clara de sus objetivos y resultados esperados.

Nombre del producto

Automatización del proceso para proyectos de software mediante la implementación de integración continua y despliegue continuo CI/CD usando la metodología scrum y basado en la cultura DevOps

Cliente Objetivo

Desde programadores independientes hasta empresas de desarrollo de software.

Funcionalidades del sistema

El alcance del presente proyecto abarca los procesos de integración de código, compilación, pruebas y despliegue para proyectos de software mediante la

Integración Continua (CI) y el Despliegue Continuo (CD) usando la metodología scrum y basado en la cultura DevOps.

Se busca establecer y optimizar los procesos de Integración Continua (CI), abarcando la fusión regular de fragmentos de código de diversos desarrolladores en un repositorio compartido. La eficiencia y consistencia en la compilación del código fuente para convertirlo en ejecutables o artefactos deployables son aspectos esenciales, garantizando así la calidad y confiabilidad del software. Además, se incorpora la automatización de pruebas para verificar la funcionalidad y la integridad del software de manera continua, asegurando que los cambios introducidos no afecten adversamente a la aplicación existente.

El proyecto también aborda el Despliegue Continuo (CD), buscando automatizar el despliegue de nuevas versiones del software, con el objetivo de acelerar la entrega y reducir posibles errores humano.

La metodología Scrum se emplea como marco de trabajo, implementando sprints y reuniones diarias para facilitar la colaboración y la flexibilidad al momento del desarrollo de todo el proceso de CI y CD.

Por último, se promueve la cultura DevOps, fomentando la colaboración estrecha entre equipos de desarrollo y operaciones para mejorar la comunicación y la eficiencia a lo largo de todo el ciclo de vida del desarrollo, desde la planificación hasta la operación.

A continuación, se expone la descripción de todas las historias generadas para el presente proyecto.

Tabla 3. *Historia de usuario #1*

Historia de usuario	
Numero: 1	**Puntos de Historia:** 5
Nombre: Configuración del entorno de desarrollo local	
COMO miembro del equipo de desarrollo **QUIERO** configurar fácilmente mi entorno de desarrollo local utilizando un sistema de control de versiones **PARA** asegurar la coherencia en su implementación. CA1: Entorno **Como** desarrollador **Quiero** tener un conjunto de instrucciones claras y detalladas **Para** configurar mi entorno de desarrollo local utilizando el sistema de control de versiones. CA2: Configuración **Como** desarrollador **Quiero** que el proceso de configuración sea reproducible, **Para** que cualquier miembro del equipo pueda replicar fácilmente la configuración en sus propios entornos locales.	

Tabla 4. *Historia de usuario #2*

Historia de usuario	
Numero: 2	**Puntos de Historia:** 5
Nombre: Instalación de Docker y Herramientas Necesarias	
Como miembro del equipo de desarrollo **Quiero** una guía para la instalación de Docker y otras herramientas necesarias **Para** mi entorno de desarrollo local. CA1: Instalación **Como** desarrollador **Quiero** un conjunto de instrucciones detalladas **Para** instalar Docker en mi sistema operativo local. CA2: Configuración **Como** desarrollador **Quiero** que el proceso de instalación incluya la configuración inicial de Docker y la verificación **Para** su correcto funcionamiento**.** CA3: Instrucciones **Como** miembro del equipo, **Quiero** instrucciones para instalar cualquier otra herramienta necesaria **Para** el desarrollo local, como gestores de paquetes o herramientas de prueba.	

Tabla 5. *Historia de usuario #3*

Historia de usuario	
Numero: 3	**Puntos de Historia:** 8
Nombre: Instalación y configuración del clúster	
Como miembro del equipo de desarrollo **Quiero** detallar la instalación y configuración de un clúster Minikube en mi entorno de desarrollo local **Para** trabajar en entornos Kubernetes de manera eficiente y consistente. CA1: Instrucciones de instalación **Como** desarrollador **Quiero** un conjunto de instrucciones claras **Para** instalar Minikube en mi sistema operativo local. CA2: Configuración **Como** desarrollador **Quiero** que el proceso de instalación incluya la configuración inicial del clúster Minikube **Para** la verificación de su correcto funcionamiento. CA3: Instrucciones de configuración **Como** miembro del equipo **Quiero** instrucciones **Para** la configuración de características adicionales de Minikube según las necesidades del proyecto.	

Tabla 6. *Historia de usuario #4*

Historia de usuario	
Numero: 4	**Puntos de Historia:** 8
Nombre: Instalación y configuración del Jenkins	
Como miembro del equipo de desarrollo **Quiero** una guía **Para** la instalación y configuración de Jenkins en mi entorno de desarrollo local. CA1: Instrucciones de instalación **Como** desarrollador **Quiero** un conjunto de instrucciones claras **Para** instalar Jenkins en mi sistema operativo local. CA2: Proceso de instalación **Como** desarrollador **Quiero** que el proceso de instalación incluya la configuración inicial de Jenkins **Para** la verificación de su correcto funcionamiento CA3: Configuración **Como** desarrollador **Quiero** instrucciones **Para la** configuración de agentes Jenkins y la conexión con el sistema de control de versiones.	

Tabla 7. *Historia de usuario #5*

Historia de usuario	
Numero: 5	**Puntos de Historia:** 8
Nombre: Instalación y configuración de Sonarqube	
Como miembro del equipo de desarrollo **Quiero** el detalle de la instalación y configuración de SonarQube **Para** mi entorno de desarrollo local. CA1: Instrucciones de instalación **Como** desarrollador **Quiero** un conjunto de instrucciones claras **Para** instalar SonarQube en mi sistema operativo local. CA2: Configuración **Como** desarrollador **Quiero** que el proceso de instalación incluya la configuración inicial de SonarQube **Para** la verificación de su correcto funcionamiento.	

Tabla 8. *Historia de usuario #6*

Historia de usuario	
Numero: 6	**Puntos de Historia:** 8
Nombre: Desarrollo de proyecto frontend	
Como desarrollador frontend **Quiero** una historia de usuario que guíe el desarrollo del frontend de la plataforma de venta de cursos **Para** el caso de uso**.** CA1: Catalogo de cursos **Como** usuario **Quiero** explorar un catálogo organizado de cursos con imágenes, descripciones y detalles relevantes **Para** el caso de uso para el CI/CD. CA2: Carrito de compras **Como** usuario **Quiero** tener un carrito de compras donde pueda revisar y modificar los cursos seleccionados **Para** el caso de uso para el CI/CD**.**	

Tabla 9. *Historia de usuario #7*

Historia de usuario	
Numero: 7	**Puntos de Historia:** 8
Nombre: Desarrollo de proyecto backend	
Como desarrollador backend **Quiero** una historia de usuario que guíe el desarrollo del backend de la plataforma de venta de cursos **Para** el caso de uso**.** CA1: Catalogo de cursos **Como** administrador **Quiero** poder agregar, actualizar y eliminar cursos mediante API's **Para** el caso de uso. CA2: Comunicación **Como** desarrollador **Quiero** proporcionar una API RESTful que permita la comunicación eficiente entre el frontend y el backend **Para** el caso de uso**.**	

Tabla 10. *Historia de usuario #8*

Historia de usuario	
Numero: 8	**Puntos de Historia:** 3
Nombre: Agregar Slack (Notificaciones)	
Como miembro del equipo de desarrollo **Quiero** la integración de notificaciones en Slack con Jenkins **Para** recibir alertas y actualizaciones automáticas sobre los resultados de las compilaciones y despliegues realizados con Jenkins. CA1: Configuración **Como** administrador de Jenkins **Quiero** poder configurar la integración con Slack desde **Para** la interfaz de administración de Jenkins. CA2: Jenkins exitoso **Como** desarrollador **Quiero** recibir notificaciones en un canal de Slack **Para** cuando una construcción en Jenkins sea exitosa. CA3: Jenkins fallo **Como** desarrollador **Quiero** recibir notificaciones en un canal de Slack **Para** cuando una construcción en Jenkins falle.	

Tabla 11. *Historia de usuario #9*

Historia de usuario	
Numero: 9	**Puntos de Historia:** 13
Nombre: Desarrollo de pipeline para entrega continua (CI)	

Como desarrollador
Quiero que guíe el desarrollo del pipeline
Para integración continua (CI) con Jenkins.

CA1: Configuración
Como desarrollador
Quiero configurar el repositorio del proyecto
Para que Jenkins pueda acceder al código fuente y rastrear los cambios.

CA2: Build
Como desarrollador
Quiero definir un pipeline Jenkins que incluya etapas
Para la construcción y pruebas

CA3: Test
Como desarrollador
Quiero que el pipeline ejecute pruebas automatizadas
Para garantizar la calidad del código.

CA4: Notificación
Como desarrollador
Quiero recibir notificaciones en Slack
Cuando el pipeline sea exitoso o falle en alguna etapa.

CA5: Pull request, Branches
Como desarrollador,
Quiero que el pipeline se ejecute automáticamente en ramas específicas y pull requests
Para proporcionar una retroalimentación rápida.

Tabla 12. *Historia de usuario #10*

Historia de usuario	
Numero: 10	**Puntos de Historia:** 13
Nombre: Desarrollo de pipeline para despliegue continuo (CD)	
Como desarrollador **Quiero** definir un pipeline Jenkins que incluya etapas **Para** la construcción y pruebas CA1: Configuración **Como** desarrollador **Quiero** configurar los entornos **Para** que Jenkins pueda realizar despliegues automáticos. CA2: Deploy **Como** desarrollador **Quiero** que el pipeline se ejecute **Para** verificar la estabilidad del despliegue. CA3: Notificación **Como** desarrollador **Quiero** recibir notificaciones en Slack sobre el estado del pipeline de despliegue **Cuando** el pipeline sea exitoso o falle en alguna etapa.	

Tabla 13. *Historia de usuario #11*

Historia de usuario	
Numero: 11	**Puntos de Historia:** 8
Nombre: Agregar Prometeus y grafana (Métricas)	
Como desarrollador **Quiero** una guía para la integración de Prometheus y Grafana **Para** monitorear métricas clave de la aplicación. CA1: Configuración **Como** desarrollador **Quiero** configurar Prometheus para que recoja métricas del sistema, la aplicación y otros servicios relevantes **Para** realizar el seguimiento y la salud de los sistemas**.** CA2: Grafana **Como** desarrollador **Quiero** configurar Grafana **Para** que visualice y presente las métricas recopiladas por Prometheus. CA3: Paneles **Como** desarrollador **Quiero** crear paneles de métricas en Grafana **Para** mostrar información clave, como el uso de CPU, la memoria, el tráfico de red y la latencia. CA4: Notificación **Como** miembro del equipo de operaciones, **Quiero** integrar Prometheus y Grafana con Jenkins **Para** visualizar métricas relacionadas con los pipelines y despliegues.	

Glosario de términos

El glosario de palabras se creó considerando que no todos los lectores del presente trabajo de titulación tienen experiencia en la carrera de software. Al encontrarse con términos desconocidos durante la lectura, el glosario busca satisfacer su curiosidad al proporcionar definiciones y explicaciones.

Tabla 14. *Glosario de términos*

Término	Significado	Alias
DevOps	Cultura que promueve la colaboración entre desarrollo y operaciones para lograr entregas de software más rápidas y confiables	DevOps
Integración Continua	Práctica de fusionar continuamente el trabajo de todos los desarrolladores en un proyecto compartido de manera continua.	CI
Despliegue Continuo	Práctica que consiste en liberar cualquier cambio en el código de forma automática a un entorno de producción.	CD
Scrum	Es un marco de trabajo ágil para la gestión de proyectos y desarrollo de productos.	SCRUM

Modelo de componentes

Los diagramas de componentes son útiles para determinar cuáles son los elementos esenciales del proyecto y, al mismo tiempo, ilustran las conexiones existentes entre dichos elementos y como se incorporan uno o varios componentes.

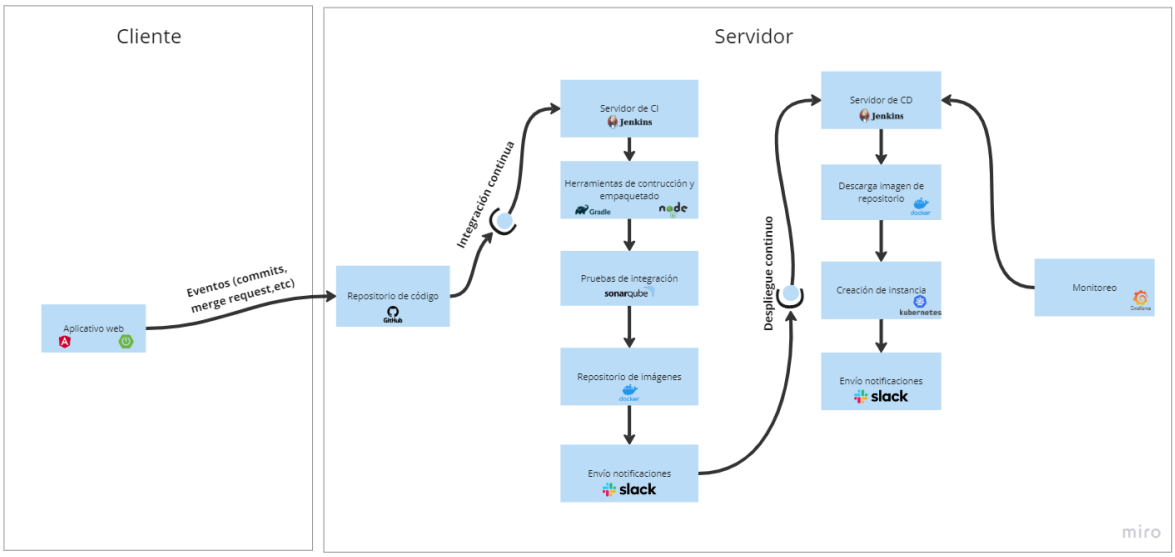

Figura 2. *Diagrama de componentes*

GitHub: Se integra con Jenkins para automatizar la compilación, las pruebas y la implementación de código.

Jenkins: automatiza la ejecución de las tareas.

Slack: permite notificar a los usuarios sobre el estado de la ejecución.

Docker: se encarga de construir las imágenes y con la integración de Jenkins subirlas al repositorio de Docker Hub.

Kubernetes: se encarga de la contenerización y despliegue de las imágenes registradas en Docker Hub.

Grafana: se encarga de monitorizar el rendimiento de las aplicaciones en contenedores.

Diagrama de marco de trabajo

En el diagrama de marco de trabajo se describirán los pasos que forman parte de DevOps:

Build

Desde la primera parte de Build o Construcción en la cual, lo que realiza es el construir los componentes necesarios entre los que más se usa suelen ser Gradle y NodeJS.

Test

La etapa de Test o prueba, la cual permite encontrar cobertura del código, si cuenta con bugs o si está mal realizada alguna prueba unitaria, entre las más conocidas que se ocupan para medir este tipo de pruebas son Junit y SonarQube.

Hasta este momento se describe la parte de integración continua.

Release

Luego sigue con la etapa de reléase o lanzamiento, la cual nos permite integrar toda la parte que se realizó anteriormente, esto se lo realiza mediante Jenkins.

Deploy

Finalmente, para tener una construcción completa del despliegue continuo se realiza el deploy o también conocido como despliegue, el cual nos permite integrar todo lo realizado, para que sea finalmente puesto a prueba por el desarrollador.

Operate & Monitor

Adicionalmente contamos con la parte de Operate u operaciones y Monitor o conocido como monitoreo que también forman parte del despliegue continuo, para este apartado se necesita de Kubernetes que será nuestra principal fuente de operaciones, lo que permite es el proveer infraestructura para nuestros correctos despliegues y finalmente en el monitoreo nos encontraremos la parte de métricas para poder medir la salud de las aplicaciones, los que se utilizan suelen ser Prometheus y Grafana.

A continuación, se muestra el marco de trabajo completo:

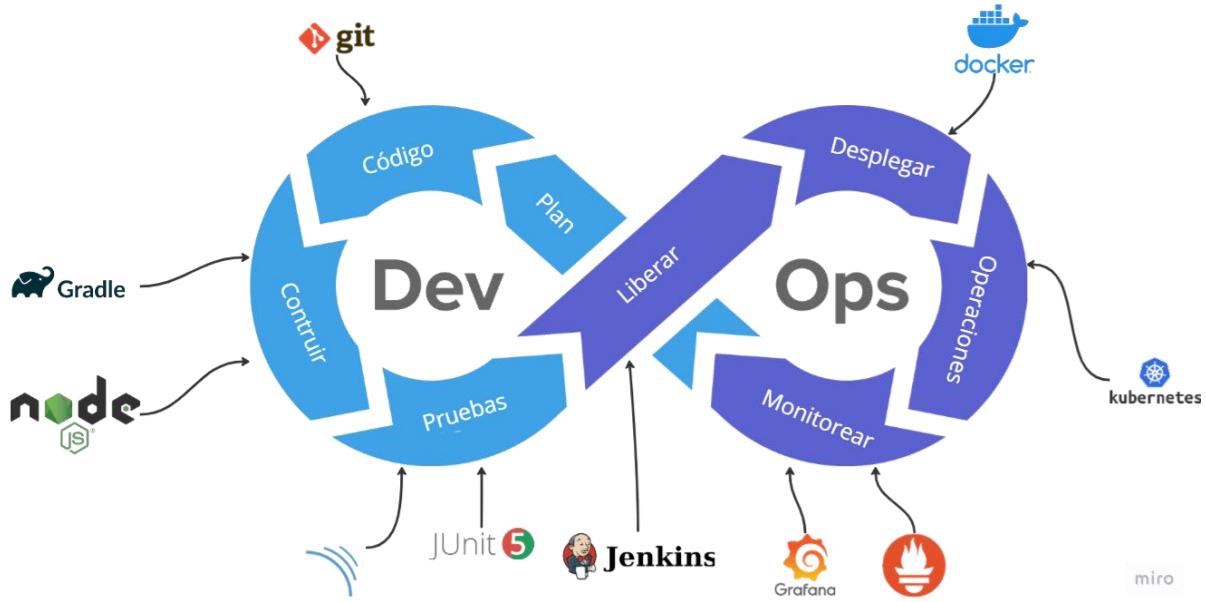

Figura 3. *Diagrama de Marco de Trabajo*

Diagrama de caso de uso

Se muestra a continuación los casos de uso correspondientes para la integración y despliegue continuo.

Integración continua

El proceso de integración continua inicia con la creación de un commit en el repositorio de GitHub del proyecto. Cada commit representa una unidad de cambio que se incorpora al código, introduciendo los ajustes necesarios.

Posteriormente, la construcción del código se lleva a cabo mediante la herramienta de Jenkins, donde se encuentran Gradle y Node.

Los pasos subsiguientes comprenden las pruebas, las cuales evalúan el correcto funcionamiento del código.

Estas pruebas verifican aspectos específicos del código, y la cobertura de código se mide para asegurar que un porcentaje suficiente del mismo ha sido evaluado.

En este contexto, utilizamos SonarQube como una herramienta de análisis de calidad del código.

La siguiente etapa consiste en la creación de una imagen de Docker, la cual encapsula el código y sus dependencias. Esta imagen se publica en un repositorio de imágenes, como Docker Hub, para facilitar su distribución y uso.

Finalmente, Slack notifica el éxito o fracaso de la integración continua, siendo considerada exitosa si cumple con todas las etapas y no exitosa si no lo hace.
Se muestra a continuación una figura representativa del flujo antes mencionado:

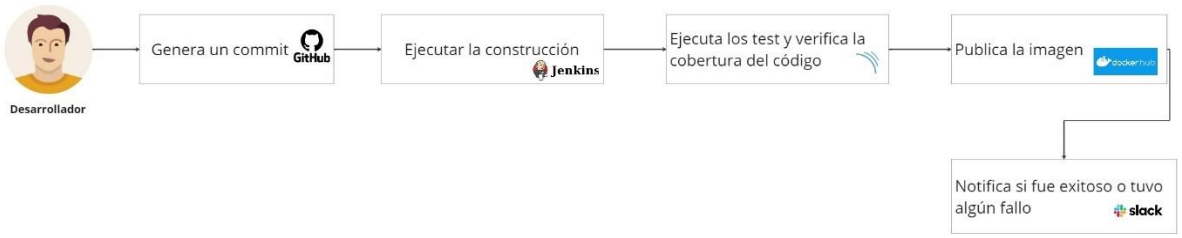

Figura 4. *Diagrama de Integración continua*

Despliegue continuo

El despliegue continuo es un proceso automatizado encargado de descargar la última imagen de una aplicación, en este caso, desde el repositorio de DockerHub.
En caso de existir un despliegue previo, se elimina y se ejecuta el despliegue con la nueva imagen.
Este flujo es esencial para entregar software de manera rápida y confiable.
Se muestra a continuación una figura representativa del flujo antes mencionado:

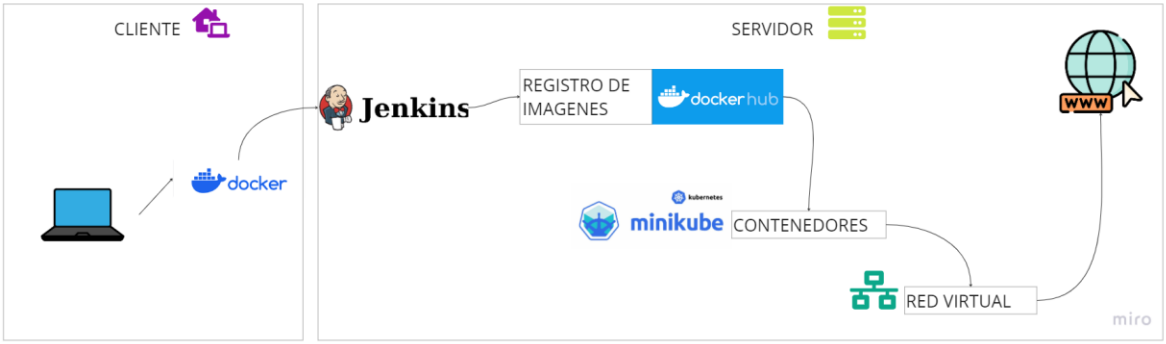

Figura 5. *Diagrama de despliegue continuo*

Diseños de interfaz de usuario

Como estudio de caso, se realizó un pequeño proyecto de venta de cursos, el cual cuenta con los diseños de interfaz de usuario para la parte visual de la prueba de CI/CD.
A continuación, se presentan los diseños:

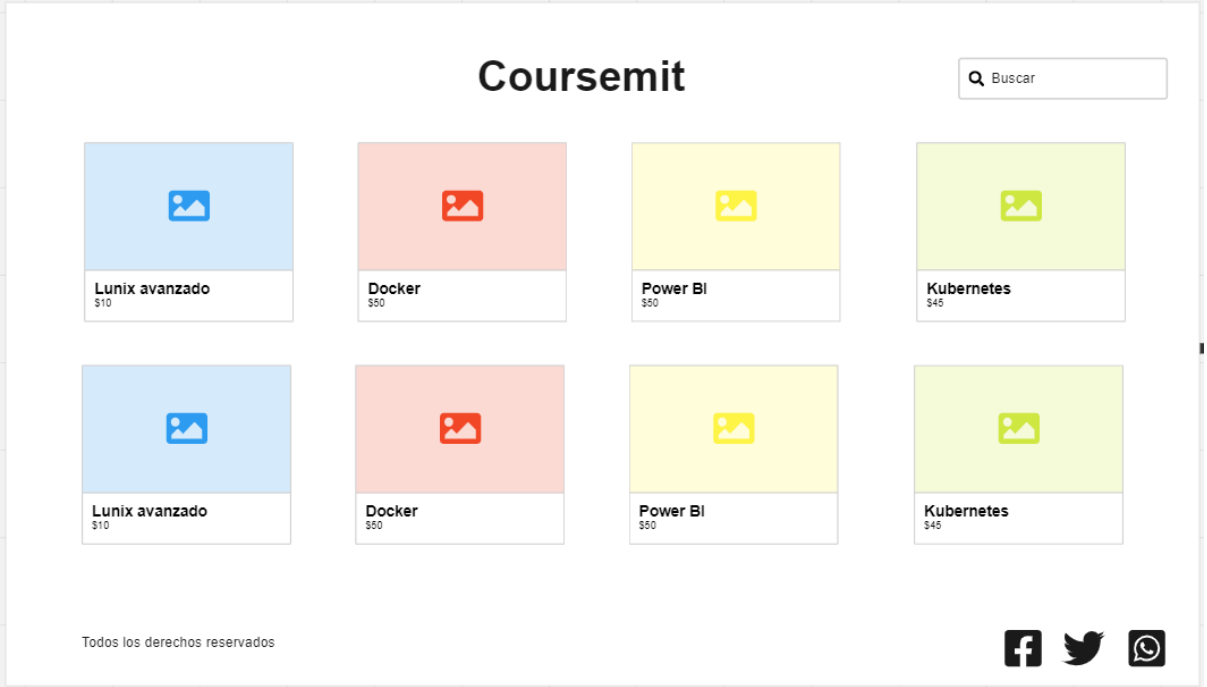

Figura 6. *Página principal*

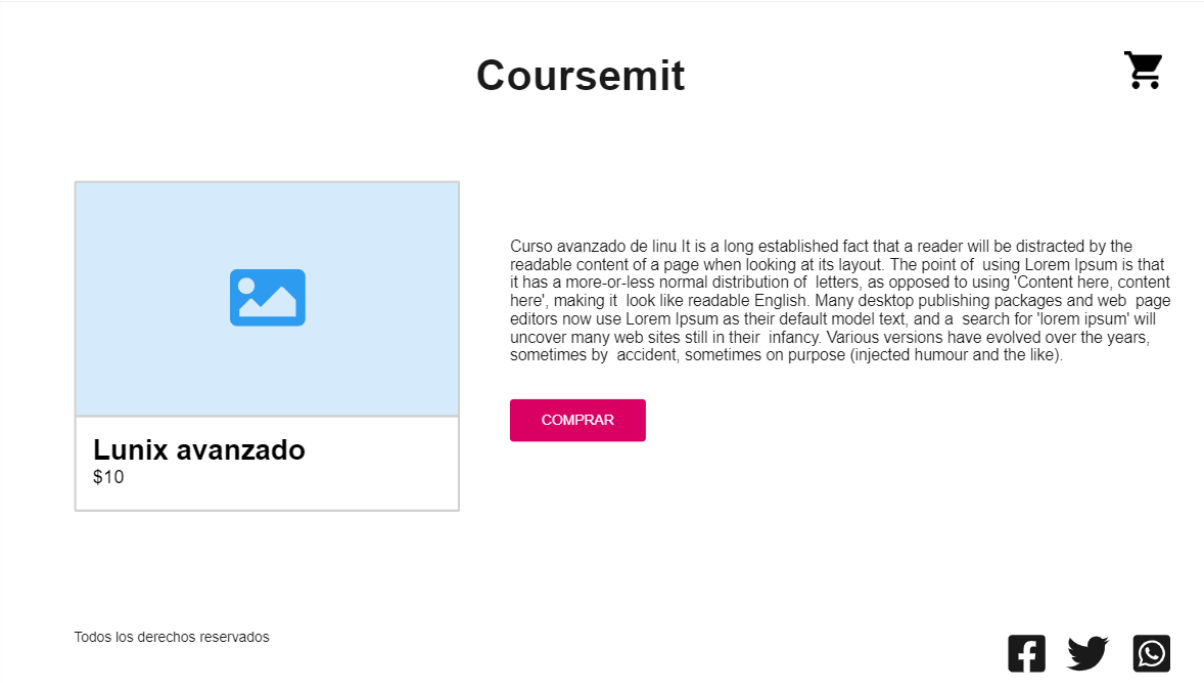

Figura 7. *Descripción del curso*

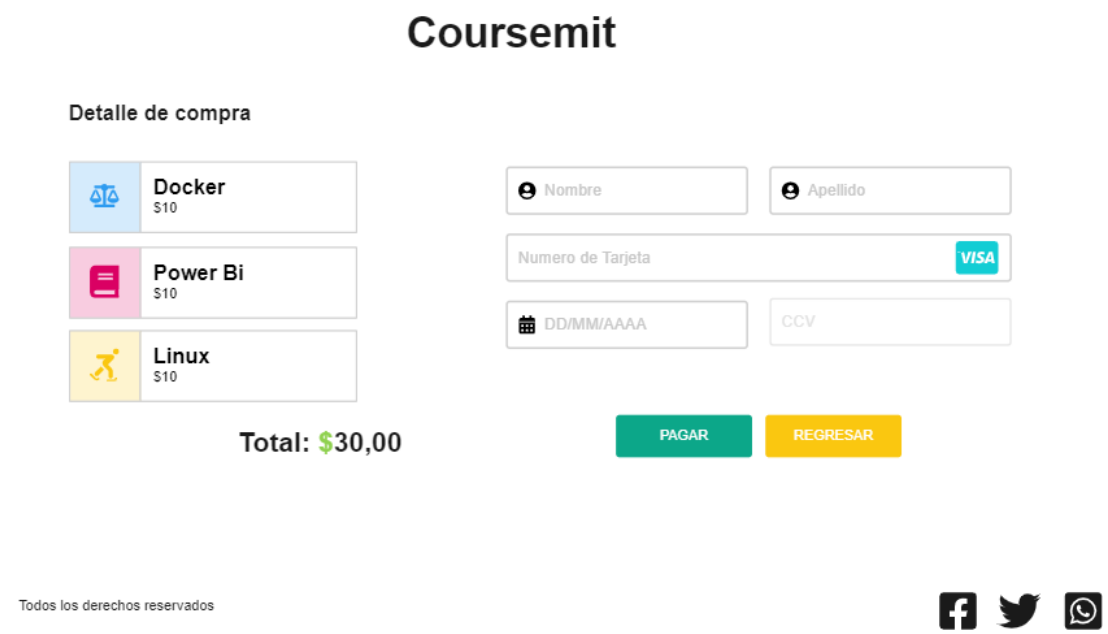

Figura 8. *Carrito de compras*

Figura 9. Creación de curso

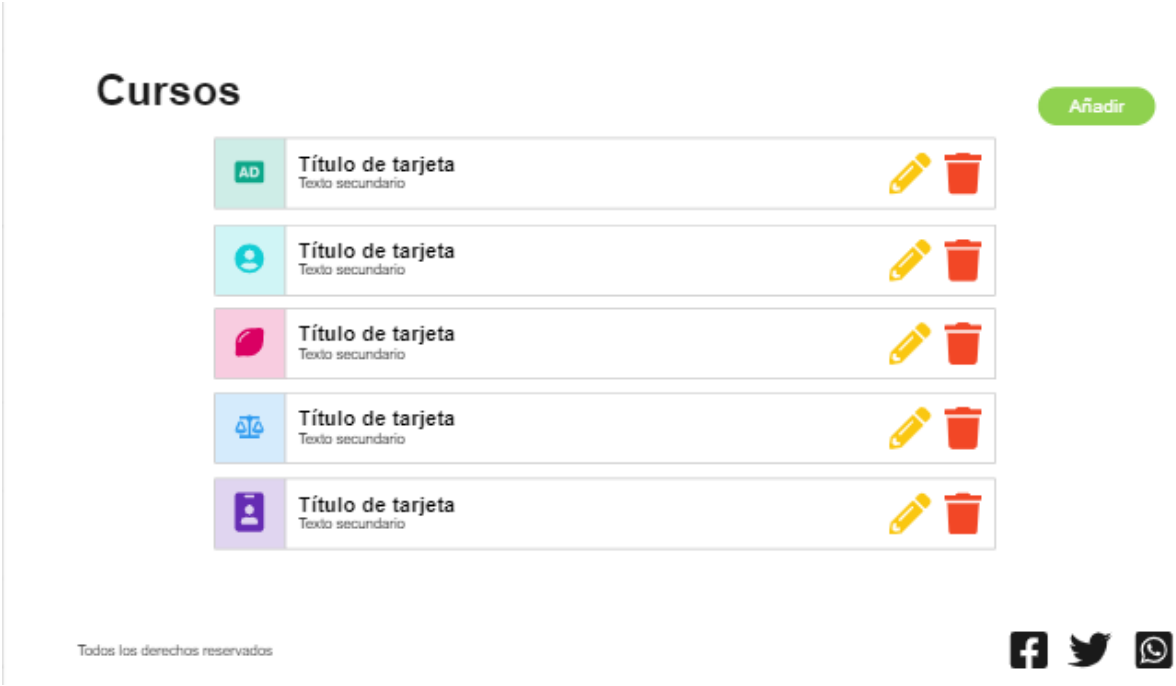

Figura 10. Lista de cursos

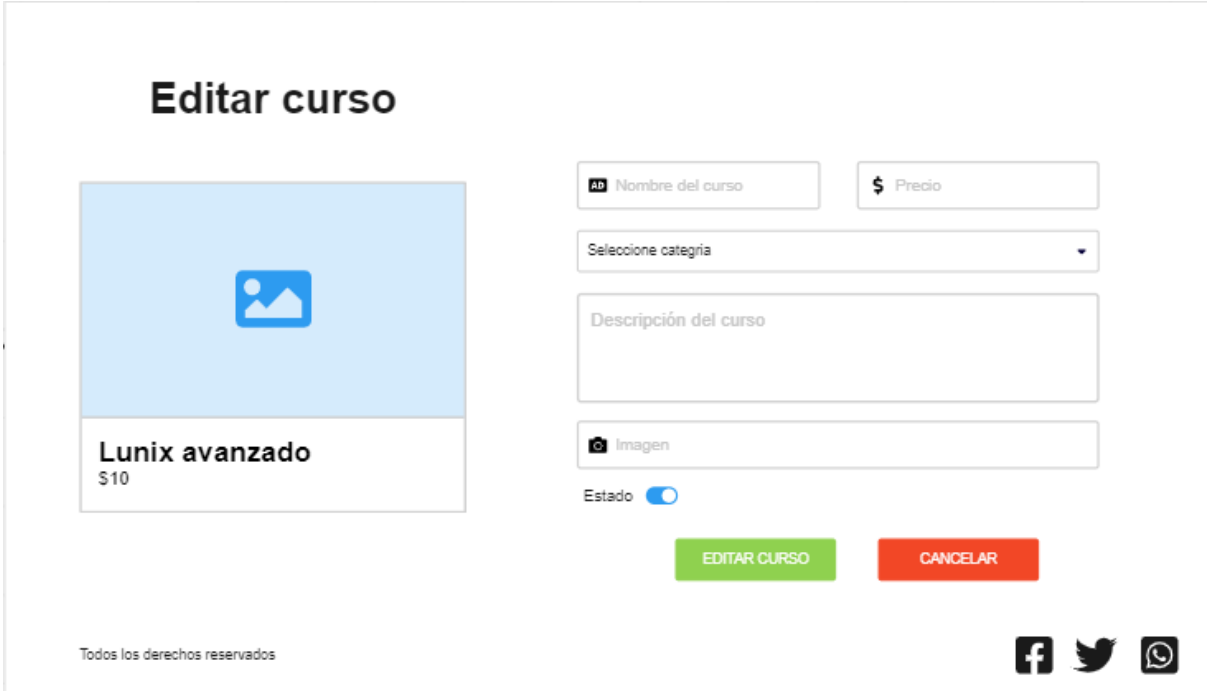

Figura 11. *Edición de cursos*

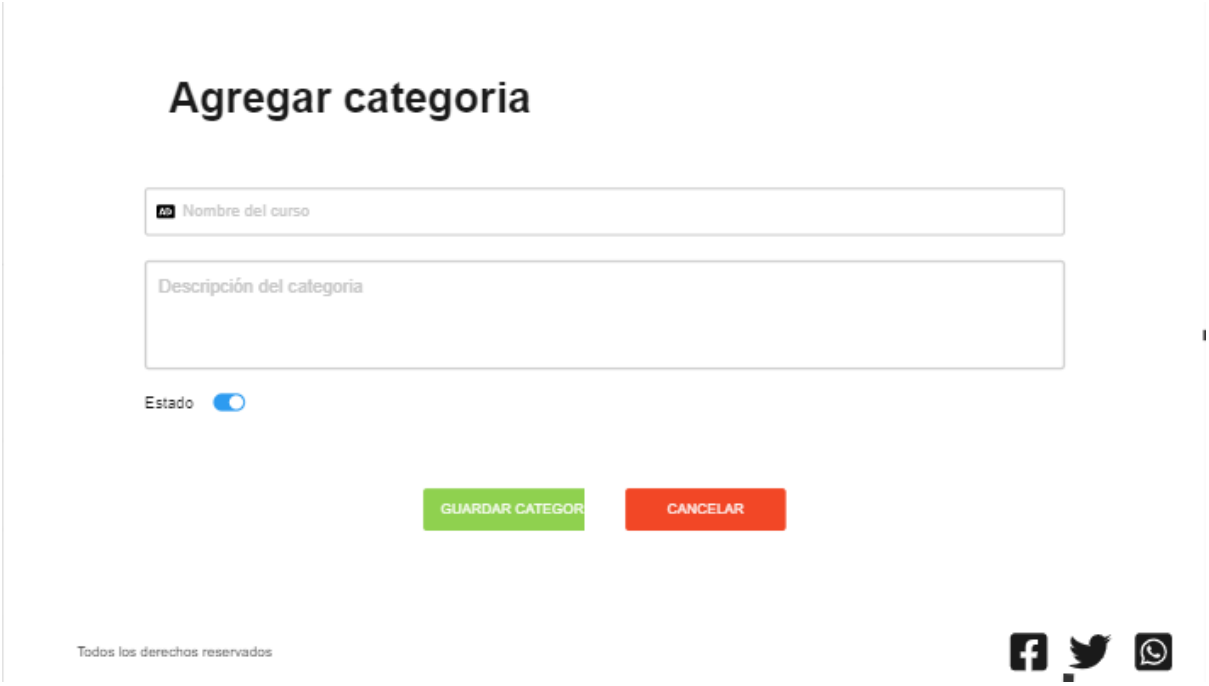

Figura 12. *Crear categoría*

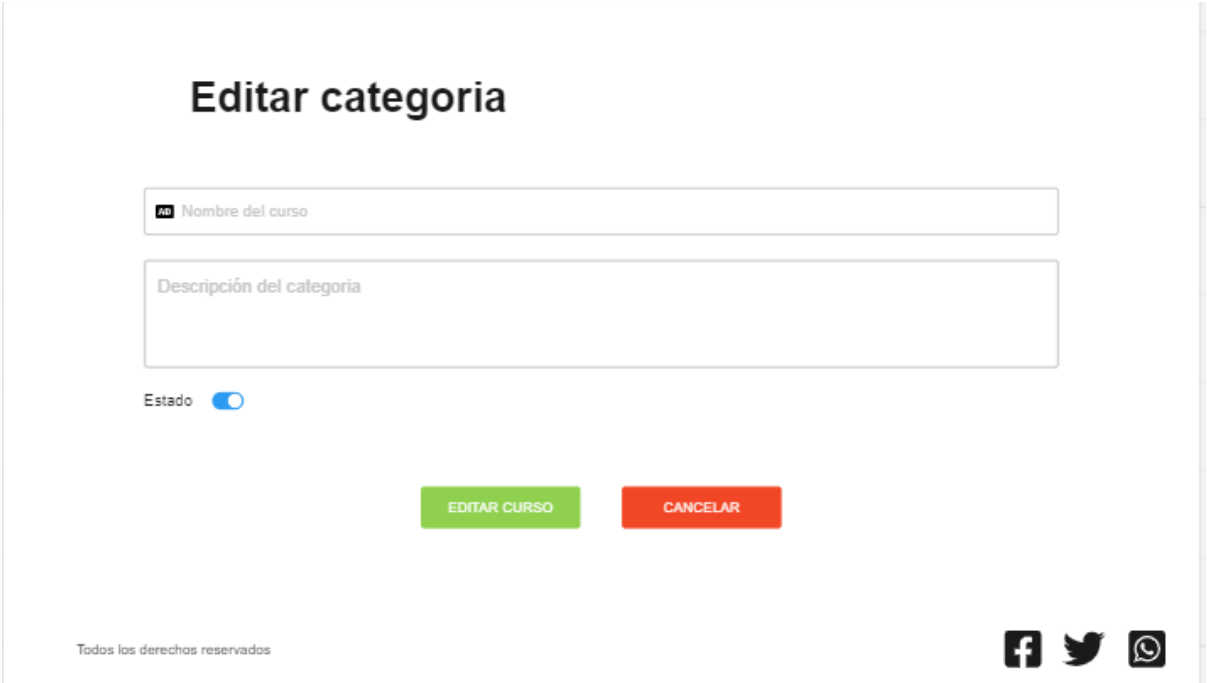

Figura 13. *Editar Categoría*

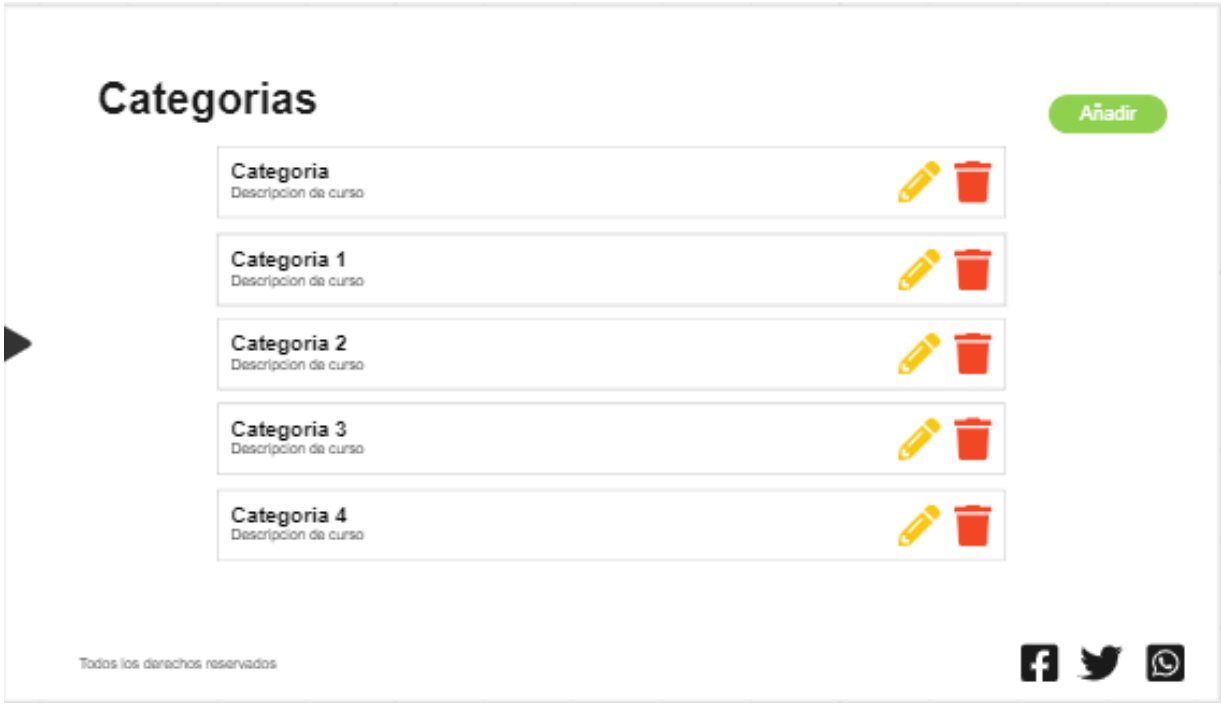

Figura 14. *Lista de categorías*

Desarrollo

Tecnologías utilizadas

En este apartado se detallan las herramientas y tecnologías que se utilizaron en el proceso de desarrollo.

Notion

Notion facilitó la colaboración efectiva, al funcionar como el centro de operaciones. En lugar de simplemente organizar información, Notion desempeñó un papel fundamental en la implementación de un tablero Scrum que permitió dar seguimiento al progreso y conocer las responsabilidades individuales.

La consolidación de toda la información crucial ayudó a que cada miembro del equipo encontrara lo que necesitaba con facilidad. Gracias a Notion, las comunicaciones se volvieron más transparentes, y la colaboración alcanzó un nivel de eficiencia notable.

Angular

Angular se utilizó como un framework para el desarrollo Frontend, permitiendo la creación de una aplicación web dinámica. Su arquitectura modular y escalabilidad brindaron la capacidad de desarrollar interfaces de usuario interactivas y eficientes.

Spring Boot

Spring Boot simplificó considerablemente el desarrollo de la aplicación backend al proporcionar configuración automática y predefinida.

Este enfoque permitió concentrarse directamente en la lógica esencial de la aplicación, eliminando las complicaciones relacionadas con las configuraciones detalladas y personalizadas.

De esta manera, Spring Boot se convirtió en una herramienta de gran ayuda, acelerando el ciclo de desarrollo y promoviendo la creación de aplicaciones Java eficientes y funcionales.

GitHub

GitHub fue esencial como repositorio central para el código fuente de las aplicaciones, siendo una pieza fundamental en la infraestructura. Facilitó crucialmente la colaboración entre los miembros del equipo al posibilitar un control de versiones efectivo. La integración exitosa de GitHub con otras herramientas, como Jenkins, generó un flujo de trabajo más eficiente y la automatización de procesos esenciales en el ciclo de desarrollo.

Docker

Docker fue fundamental al permitir empaquetar las aplicaciones y sus dependencias en contenedores.

Esto garantizó la consistencia y la portabilidad entre diferentes entornos de desarrollo y despliegue. La utilización de Docker facilitó la creación de entornos aislados y reproducibles, mejorando la eficiencia en el desarrollo y la implementación de software.

Minikube

Minikube resultó ser muy útil para facilitar la creación y gestión de clústeres de Kubernetes locales, además de permitir probar y desarrollar aplicaciones en un entorno Kubernetes a pequeña escala, replicando localmente un clúster completo.

SonarQube

SonarQube contribuyó significativamente al mejoramiento continuo de la calidad del código. Esta herramienta se convirtió en un control efectivo al identificar posibles problemas, vulnerabilidades y patrones de código malicioso al proporcionar un análisis exhaustivo del código fuente. La capacidad de abordar estas áreas mejoró la calidad del código y estableció un estándar sólido para el desarrollo, contribuyendo a la sostenibilidad y la construcción de software resiliente.

Jenkins

Jenkins actuó como pieza fundamental en la implementación de integración continua y entrega continua (CI/CD), automatizando tareas cruciales en el ciclo de desarrollo. Desde la compilación hasta las pruebas y el despliegue, Jenkins desempeñó un papel clave en la detección temprana de errores y en la entrega rápida y eficiente de software.

Grafana

Grafana fue esencial para el análisis y la visualización de datos. Al permitir la creación de paneles interactivos y dashboards, Grafana facilito la monitorización en tiempo real, mejorando la capacidad para comprender y actuar de manera efectiva sobre los datos generados por las aplicaciones.

Prometheus

Prometheus proporcionó un sistema robusto de monitoreo y alerta, fue esencial para recopilar métricas de los sistemas, además su capacidad de almacenamiento eficiente y consultas flexibles permitió obtener información detallada sobre el rendimiento y el estado de las aplicaciones.

Slack

Slack, por su parte, sirvió como una plataforma esencial de mensajería. Con características como canales de conversación, integraciones y la capacidad de compartir archivos, Slack mejoró la eficiencia y coordinación.

Además, desempeñó un papel clave al proporcionarnos notificaciones instantáneas para nuestros procesos de Integración Continua y Despliegue Continuo,

manteniéndonos siempre informados sobre el estado y los cambios en nuestros proyectos.

NgRok

Nos permitió el realizar como puente de conexión local hacia afuera, permitiendo poner un DNS, para poder conectar con el Github Webhook y posteriormente poder obtener y enviar todas las notificaciones de cualquier cambio y que este a la escucha en Jenkins.

Producto de software desarrollado

En este apartado se describe cada uno de los pasos a seguir y demostrando el uso del producto de software desarrollado.

Instalación de Docker

Antes de proceder con la instalación de Docker en Windows, se recomienda verificar que el sistema cumpla con los requisitos necesarios. Después se accede al sitio oficial de Docker en https://www.docker.com/products/docker-desktop para descargar el instalador correspondiente.

El usuario hace clic en "Download for Windows" y esperar a que se complete la descarga, como se muestra en la figura:

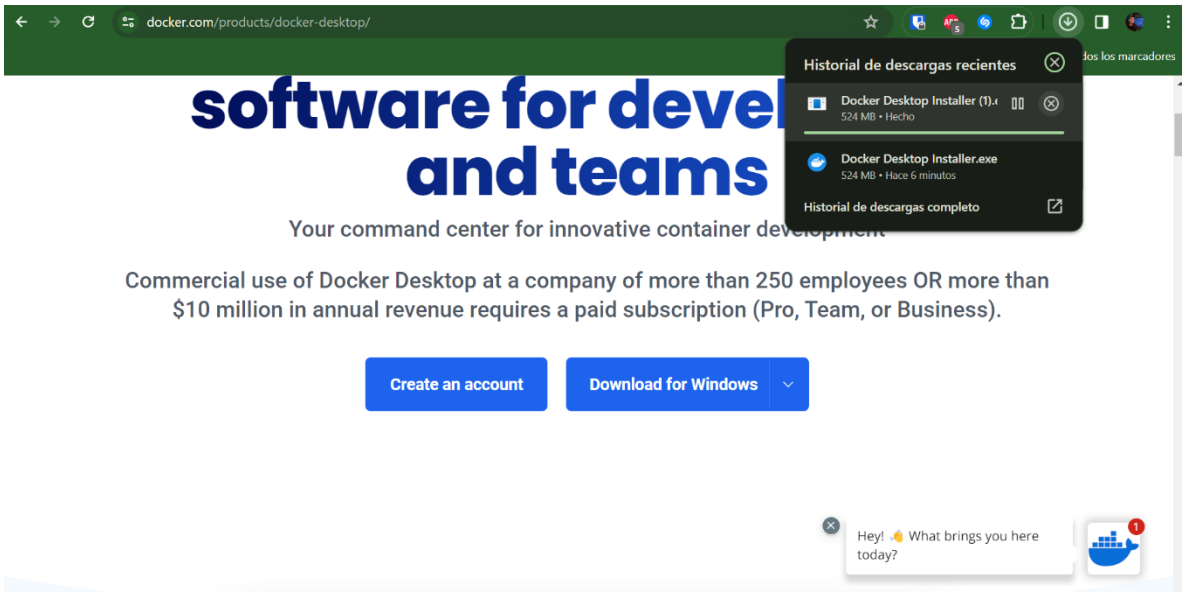

Figura 15. *Proceso de descarga*

Después de completar la descarga, iniciar el instalador de Docker Desktop. A lo largo del proceso de instalación, se le pedirá que acepte los términos del servicio luego de aceptarlos dejamos que la instalación se complete como se muestra en la figura:

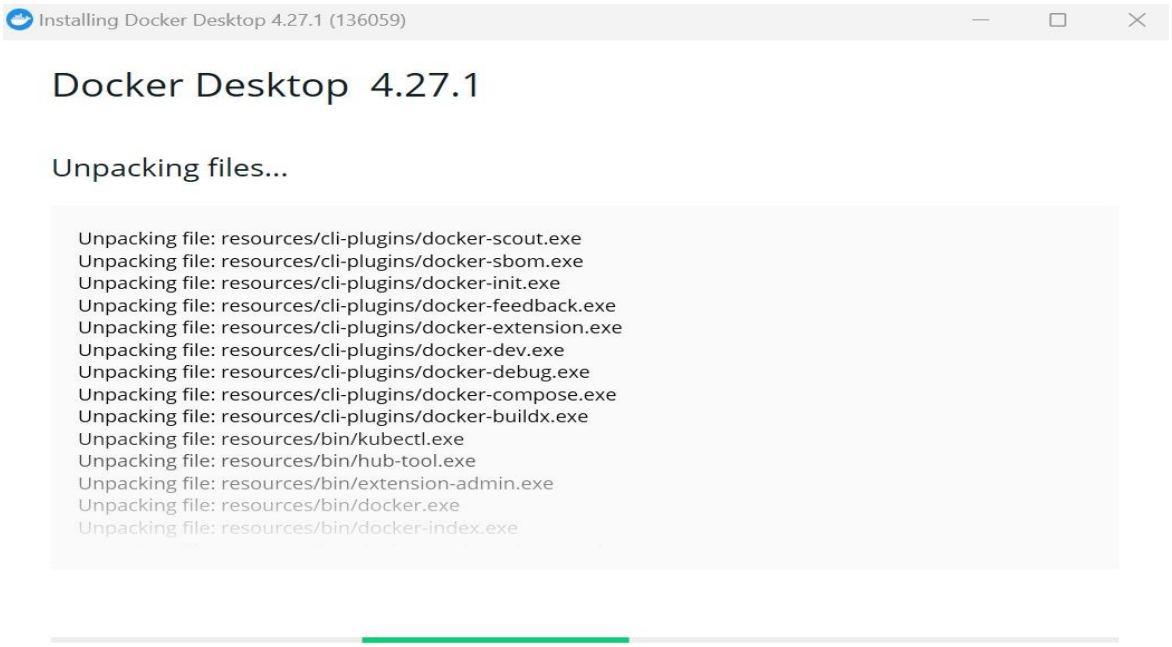

Figura 16. *Proceso de instalación*

Para finalizar una vez completado reiniciar el equipo.

Instalación de Minikube

Acceder al sitio oficial https://minikube.sigs.k8s.io/docs/start/ y después seleccionar las instrucciones específicas para el sistema operativo en uso (Windows, MacOS o Linux).

En este caso seleccionamos la opción de Windows y se procede a descargarlo como se muestra en la figura:

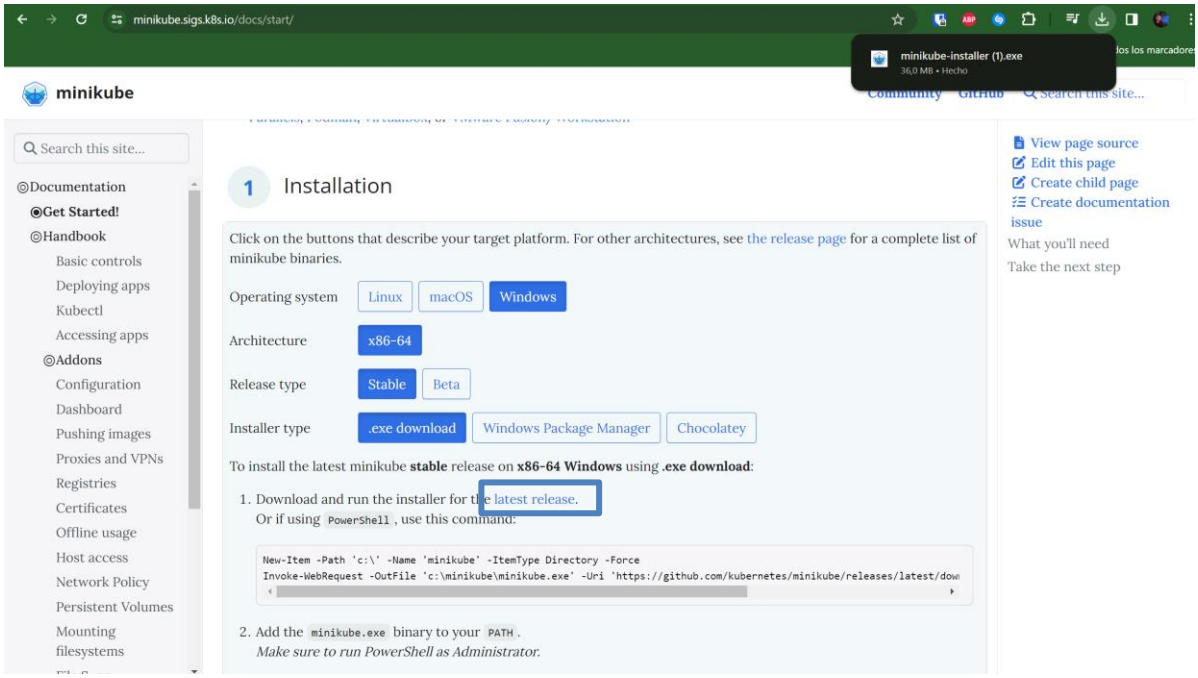

Figura 17. *Selección y descarga opción Windows*

Una vez completada la descarga, ejecutar el archivo con permisos de administrador una vez ejecutado seleccionar el idioma, luego se le pedirá que acepte los términos y condiciones y por último seleccionar el lugar donde se instalara Minikube.

Tras estos pasos, se muestra una ventana con un mensaje que muestra que la instalación se ha completado de forma satisfactoria como se muestra en las figuras:

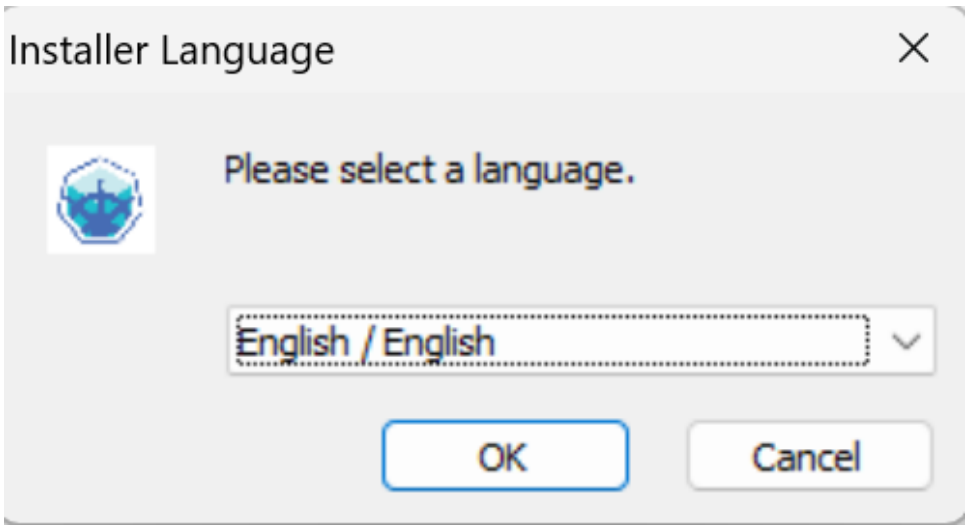

Figura 18. *Selección de idioma*

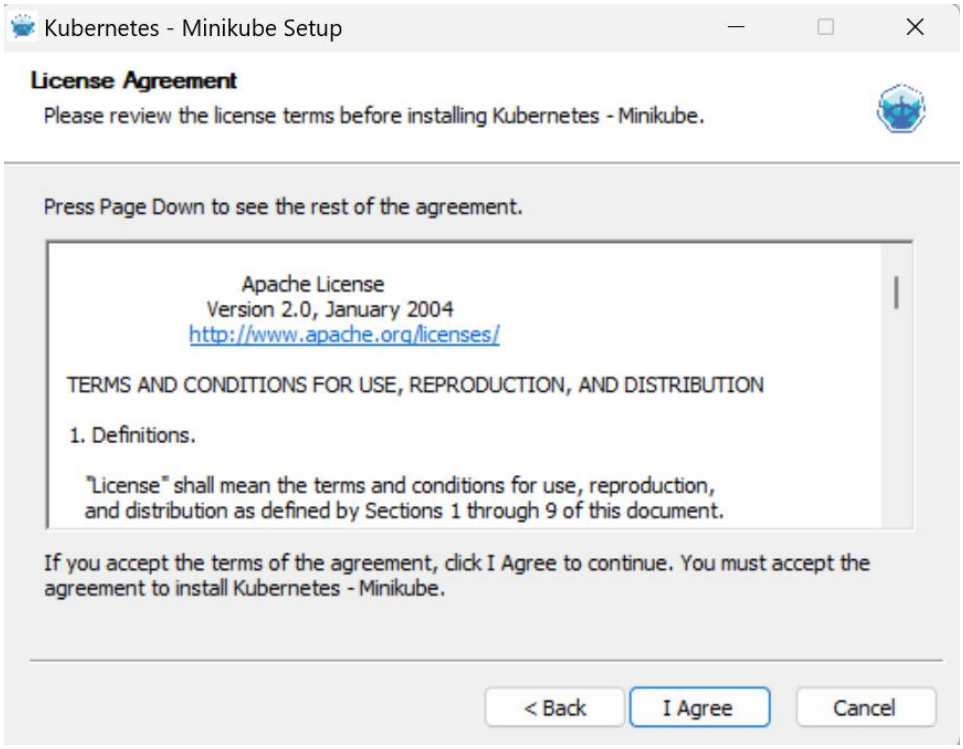

Figura 19. *Aceptación de términos y condiciones*

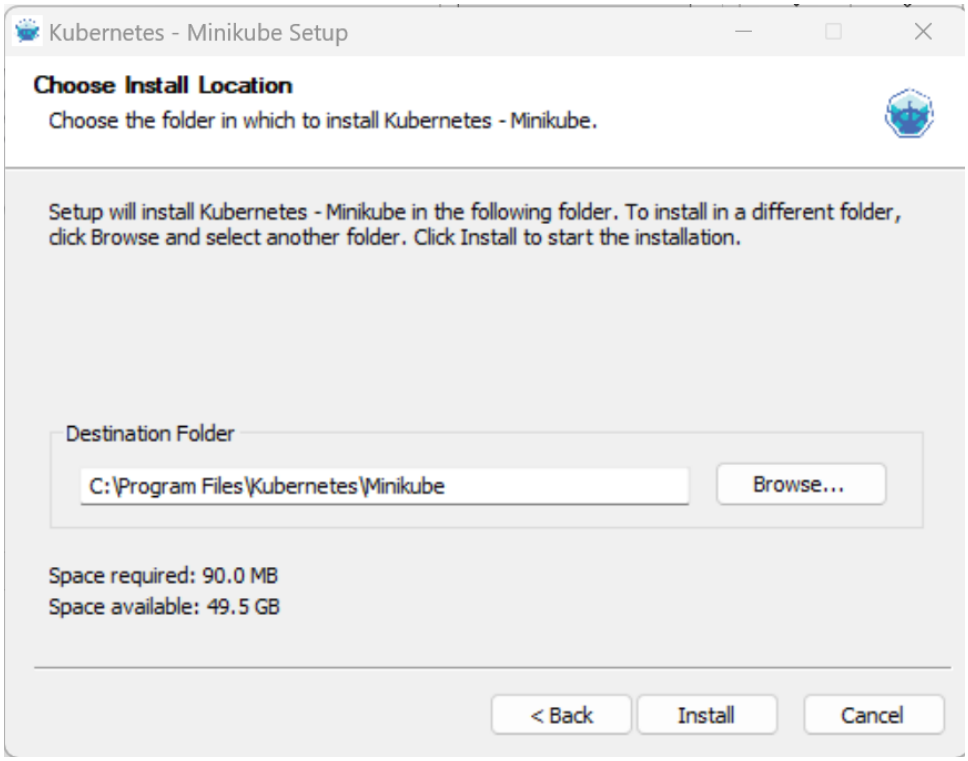

Figura 20. *Selección del lugar de instalación*

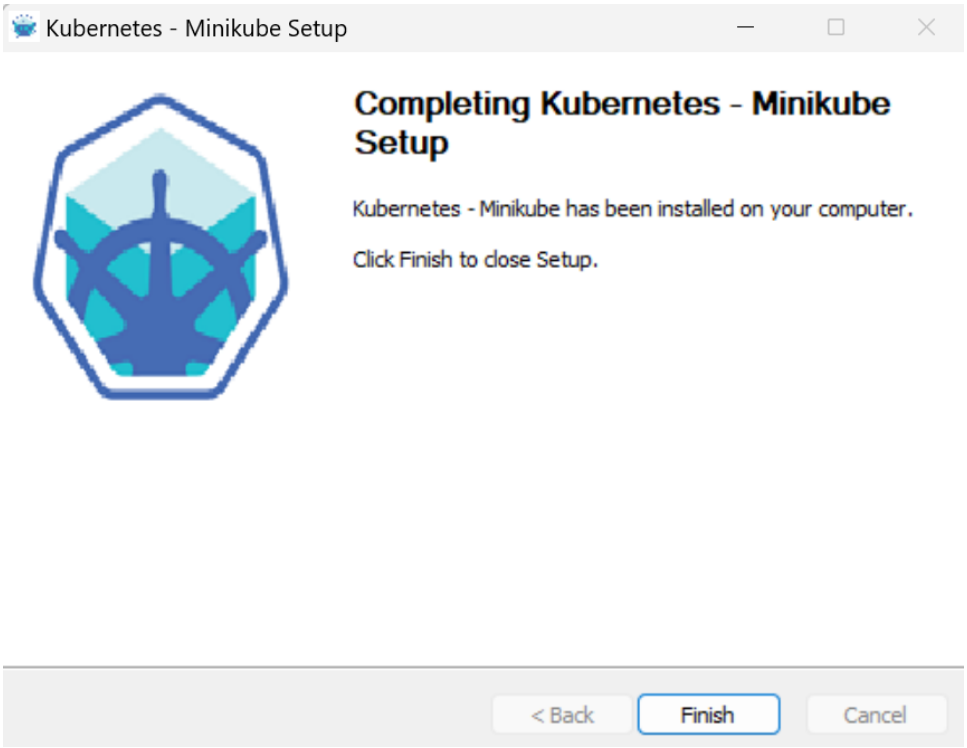

Figura 21. *Mensaje de instalación correcta*

Una vez finalizada la instalación, para levantar minikube ejecutamos el siguiente comando:

```
minikube start --driver=docker
```

en una terminal de Windows.

Instalación de Jenkins

En primer lugar, abrir una terminal y ejecute el siguiente comando:

```
docker pull alexmejiaarias/jenkins-ci-cd-3
```

para descargar la imagen oficial de Jenkins desde Docker Hub:

A continuación, se muestra la ejecución de los comandos:

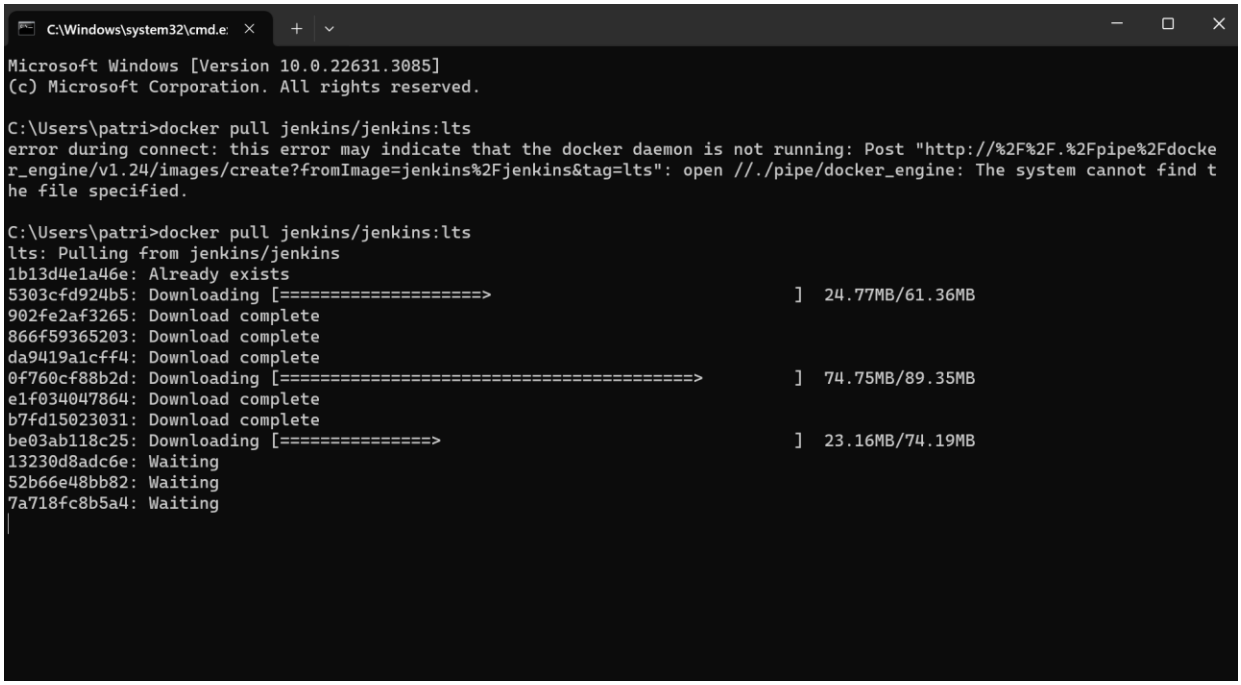

Figura 22. *Descarga de la imagen oficial de Jenkins*

Una vez descargada la imagen, ejecutar el siguiente comando:

```
docker run -p 8098:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:lts-jdk17
```

tener en cuenta la disponibilidad de los puertos para evitar algún problema.

Este comando creara y ejecutara un contenedor con Jenkins.

Una vez ejecutado el comando accedemos mediante nuestro navegador a la url con el puerto que asignamos en este caso http://localhost:8098/login?from=%2F como se muestra en la figura:

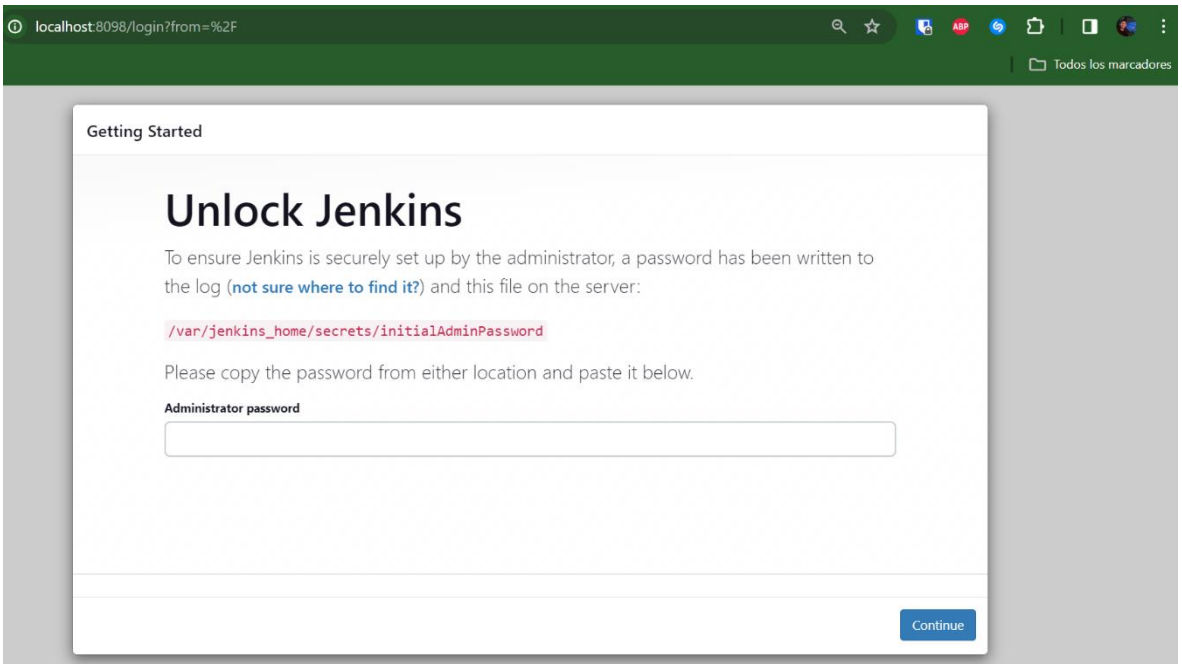

Figura 23. *Primera ventana de Jenkins*

En esta ventana se nos pedirá una contraseña la misma que encontramos en los logs del contenedor como se muestra en la figura:

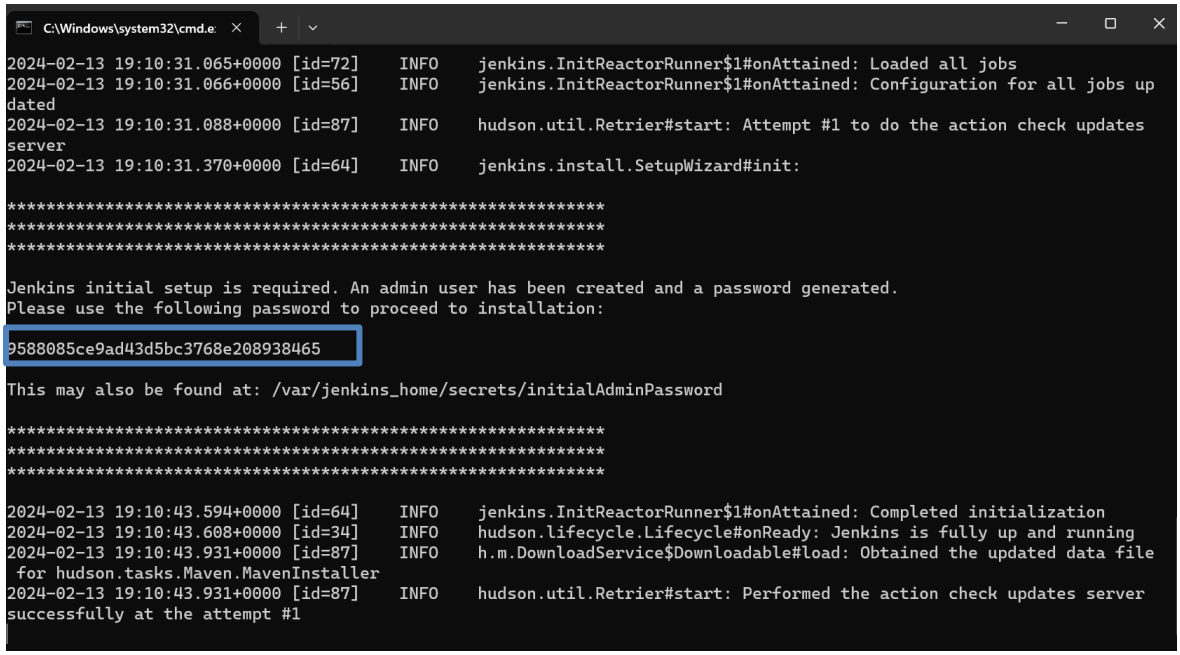

Figura 24. *Contraseña generada en los logs*

Copiar la contraseña, pegar y presionar continuar, luego se mostrará una pantalla con dos opciones es de instalación de Jenkins, una para instalar Jenkins con plugins predefinidos y otra para realizar una instalación personalizada como se muestra en la figura:

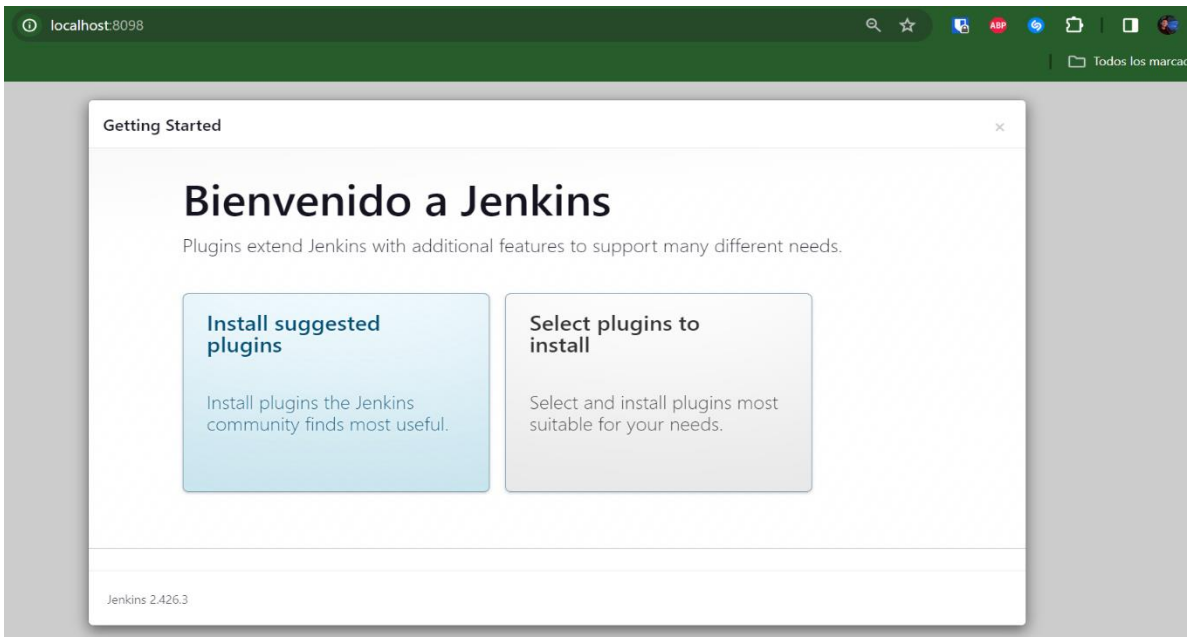

Figura 25. *Modos de instalación de Jenkins*

Seleccionamos la opción de instalación con plugins y comenzara la instalación de Jenkins como se muestra en la figura:

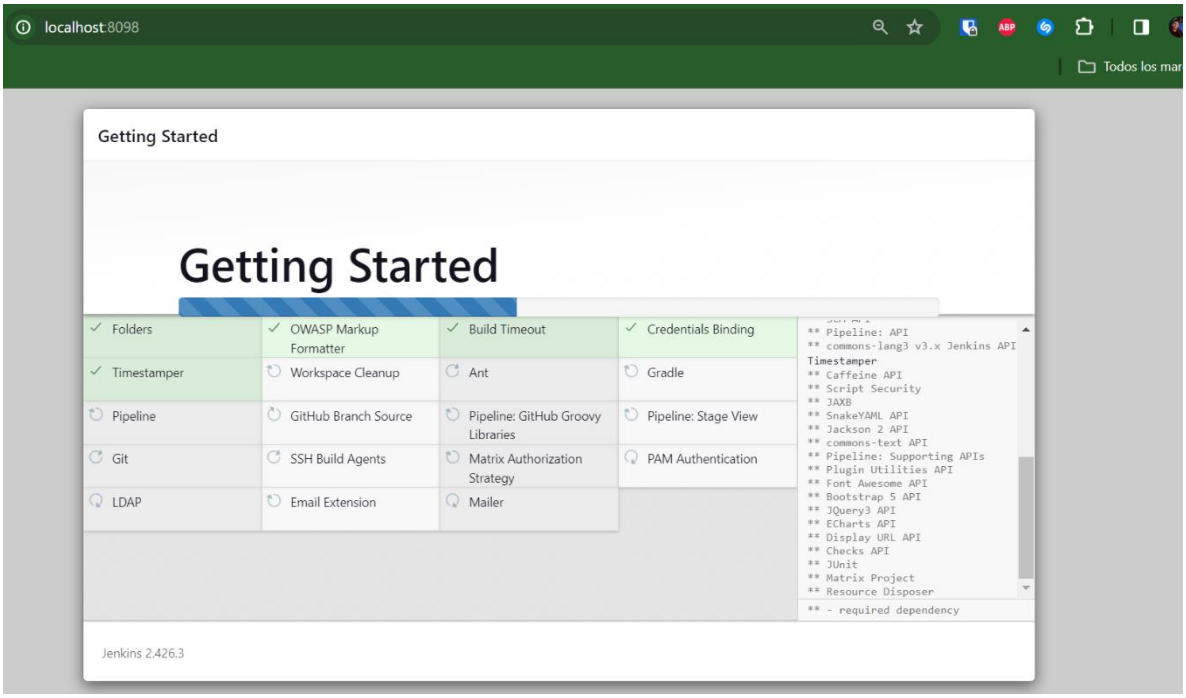

Figura 26. *Proceso de instalación de Jenkins con plugins*

Una vez finalizada la instalación de Jenkins se nos muestra una pantalla donde estableceremos un usuario Admin, aquí ingresar un usuario, contraseña, nombres completos y correo, luego presionar continuar como se muestra en la figura:

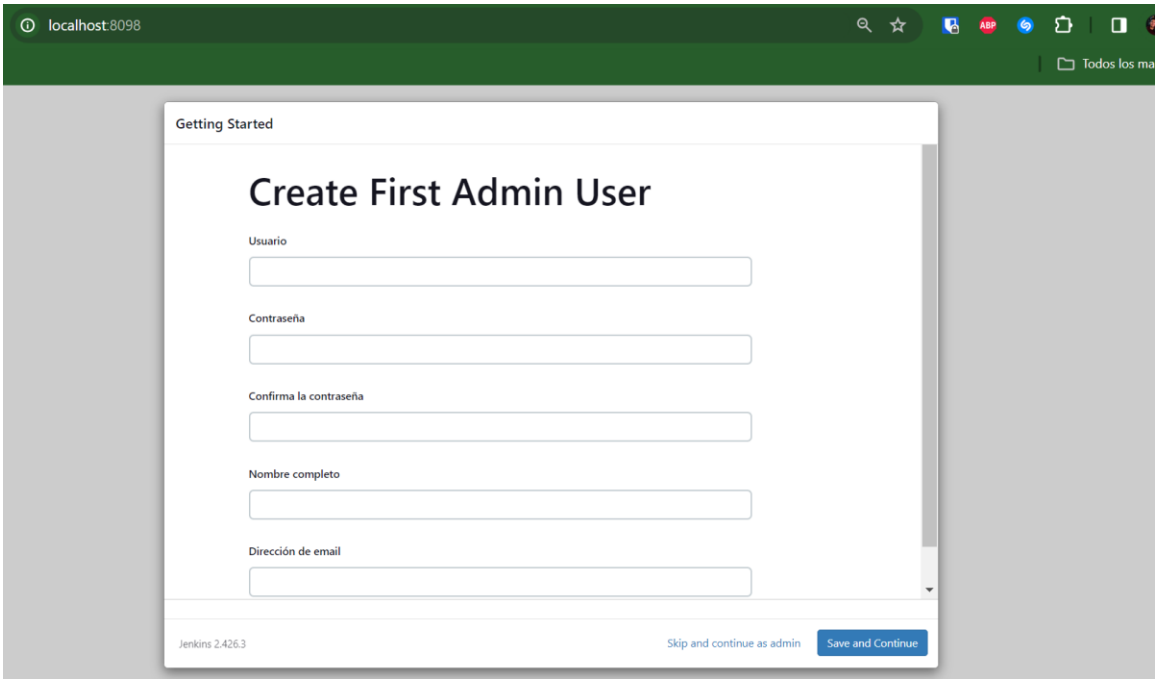

Figura 27. *Pantalla de creación de usuario admin*

Por último, se muestra una pantalla donde se especifica la url de Jenkins, por defecto toma nuestra url. Presionar guardar y finalizar y con esto se terminaría la instalación de Jenkins como se muestra en la figura:

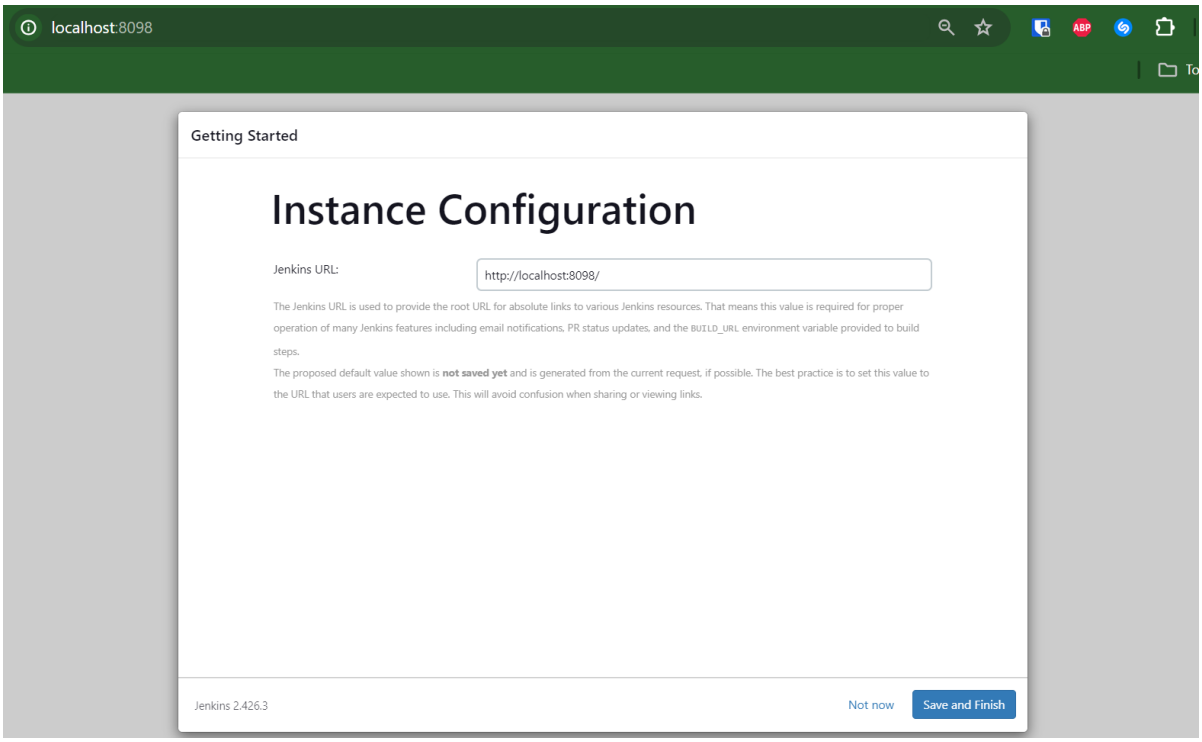

Figura 28. *Pantalla donde se establece la url de Jenkins*

Instalación de SonarQube

Para agregar Sonar lo primero que haremos es descargar el docker de Sonar de la siguiente url: https://hub.docker.com/_/sonarqube

Una vez descargado, ejecuta el siguiente comando:

```
docker run -d --name sonarqube -p 9000:9000 -p 9092:9092 sonarqube
```

Luego crearemos una red virtual con lo siguiente:

```
docker network create jenkins_sonarqube
```

Con el siguiente comando docker network ls, podremos verificar si la red se creó correctamente como se muestra en la figura:

```
C:\Users\jroba>docker network ls
NETWORK ID     NAME                DRIVER    SCOPE
3c35f8606801   bridge              bridge    local
9f6180fcfa53   host                host      local
99d3cb410c97   jenkins_sonarqube   bridge    local
89a23e29a67e   minikube            bridge    local
112e310849de   none                null      local
```

Figura 29. *Networks en Docker*

Luego conectaremos los contenedores a la red virtual con el siguiente comando:

```
docker network connect jenkins_sonarqube sonarqube
```

Para ver los detalles del contenedor el siguiente comando:

```
docker container inspect sonarqube
```

Nos mostrara lo siguiente.

Figura 30. *Inspección de SonarQube*

Instalación de NgRok

Antes de iniciar la instalación, es necesario contar con una cuenta previamente creada en ngrok, ya que en este proceso se proporcionará un token que será necesario más adelante.

Ir la página oficial de ngrok en https://ngrok.com/download. Una vez allí, se ofrecen dos opciones para la instalación: a través de Chocolatey o descargando el archivo ZIP.

Una vez completada la instalación, se debe ejecutar el siguiente comando en una terminal de Windows:

```
ngrok config add-authtoken <token>
```

ingresando el token proporcionado en nuestra cuenta ngrok.

Con estos pasos, ahora estamos listos para ejecutar el siguiente comando, especificando el puerto en el que Jenkins está siendo ejecutado:

```
ngrok http 8098
```

Esto nos proporciona una URL publica que expone nuestro servicio local a internet, como se muestra en las figuras:

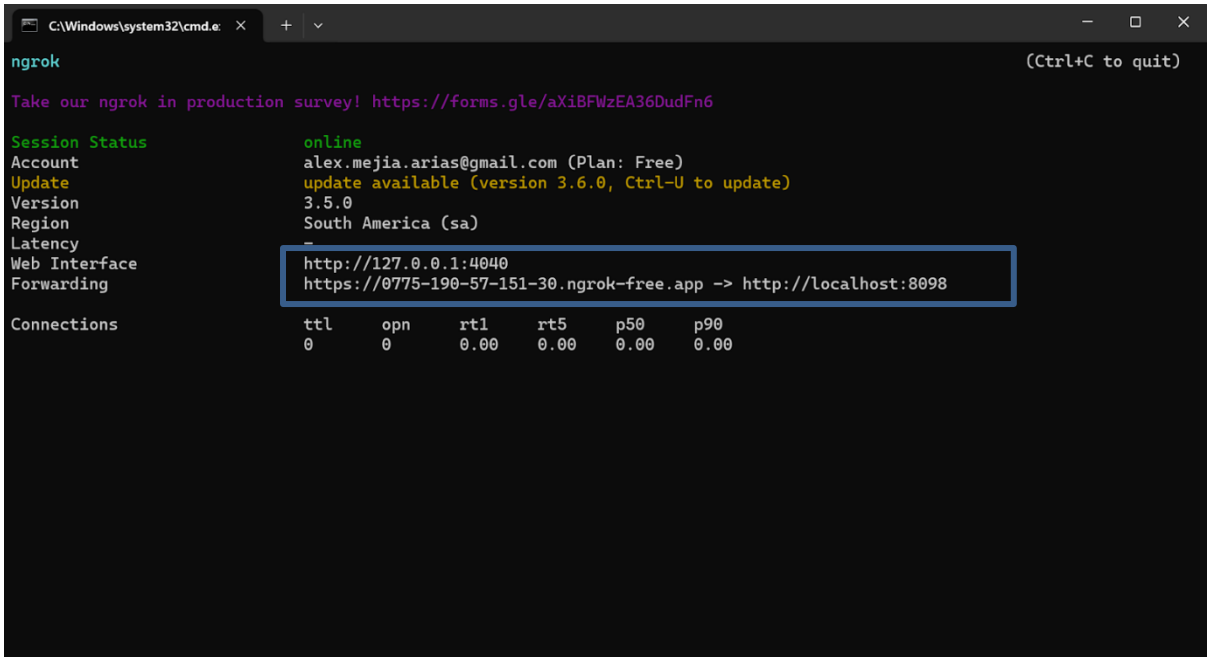

Figura 31. *NgRok habilitado*

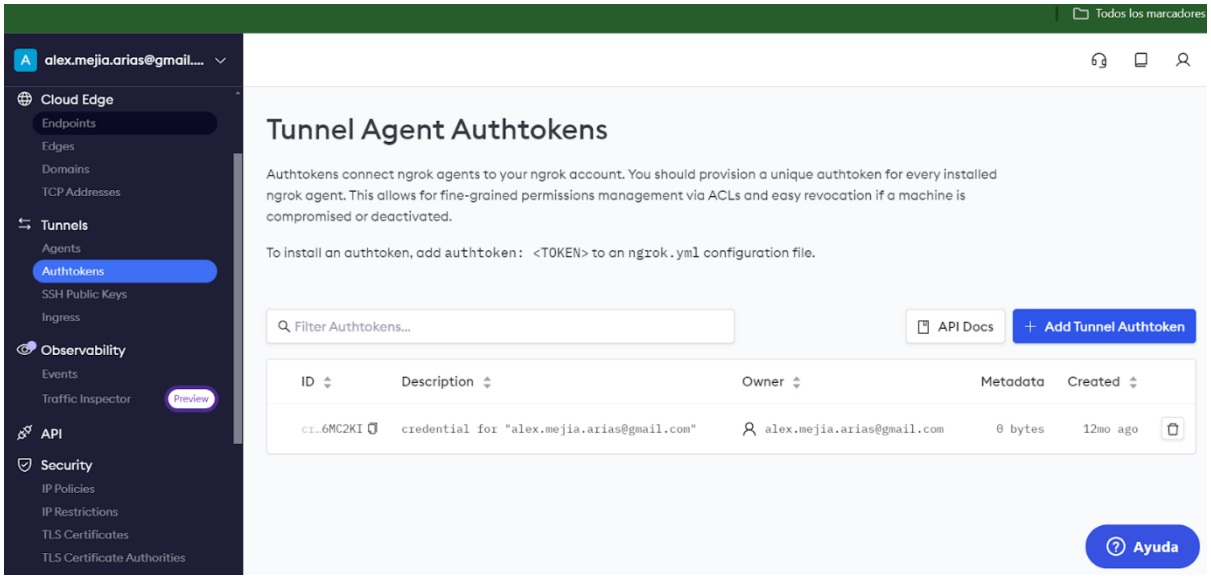

Figura 32. *Token para NgRok*

Configuración de GitHub Webhooks

En las configuraciones del proyecto, acceder al apartado de "Webhooks" y agregar un nuevo webhook.

En el formulario que se despliega, ingresar la URL generada por ngrok.

Seleccionar "application/json" como tipo de contenido.

¿En la sección “Which events would you like to trigger this webhook?", elegir "Send me everything" y finalmente, presione "Agregar Webhook", como se muestra en las siguientes figuras:

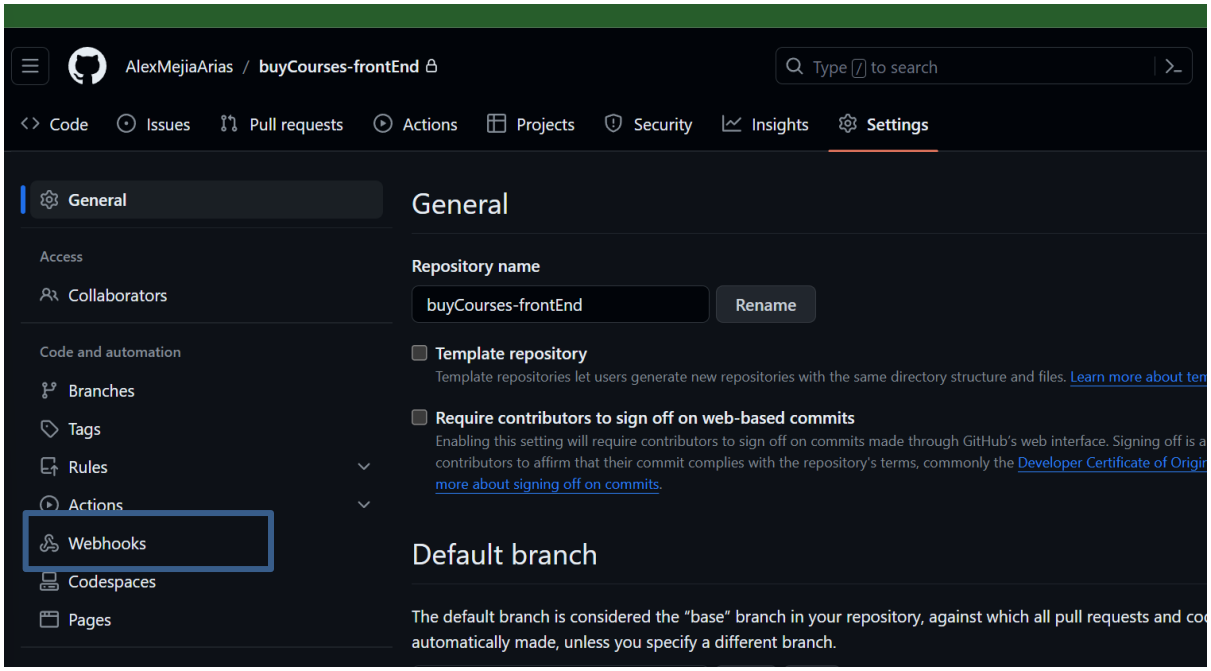

Figura 33. *Webhook del proyecto*

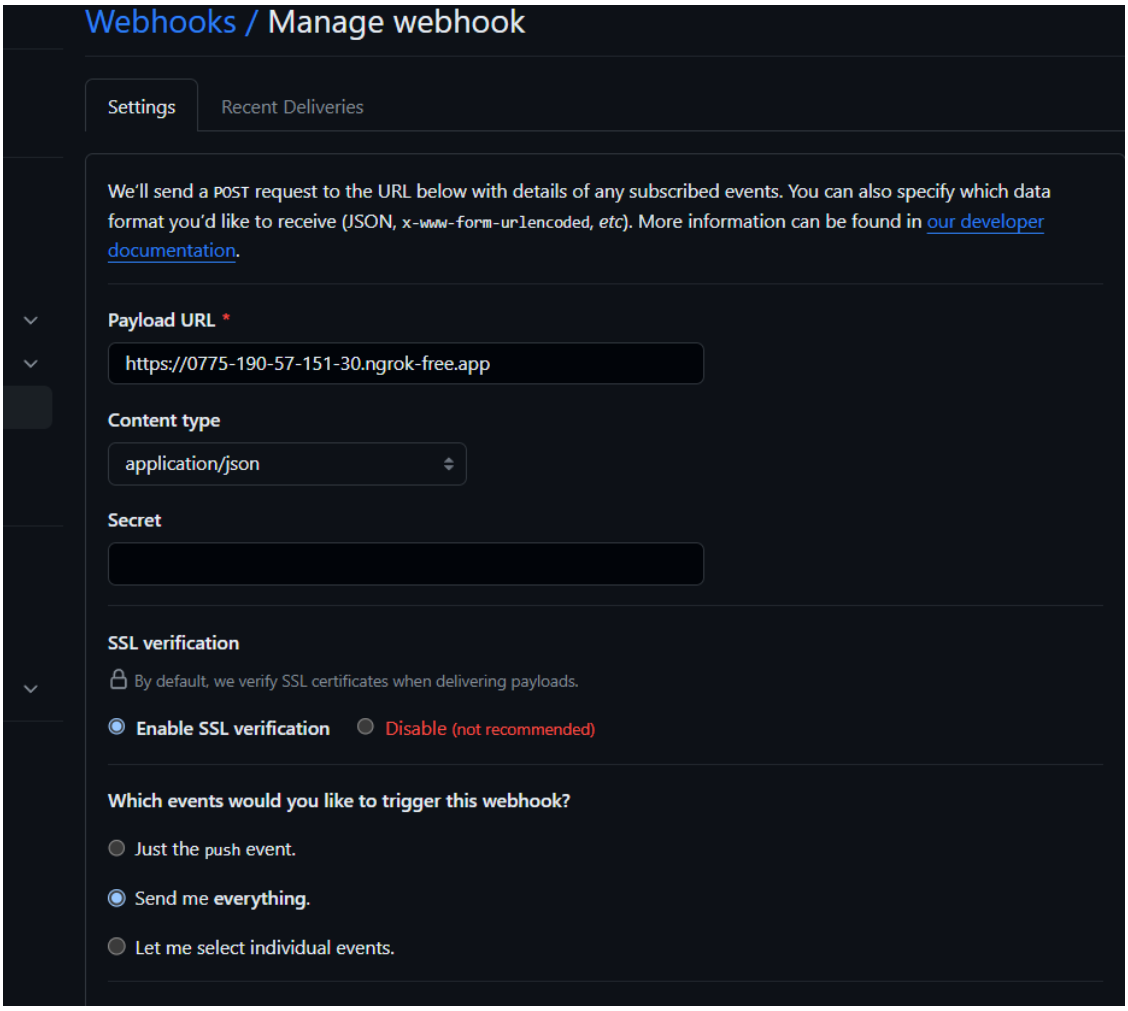

Figura 34. *Configuración de Webhook*

Configuración de credenciales GitHub en Jenkins

Ingresa a la cuenta de GitHub donde se alojan los proyectos, aquí ingresamos en la configuración y en este apartado seleccionar la opción configuración de desarrollador como se muestra en la figura:

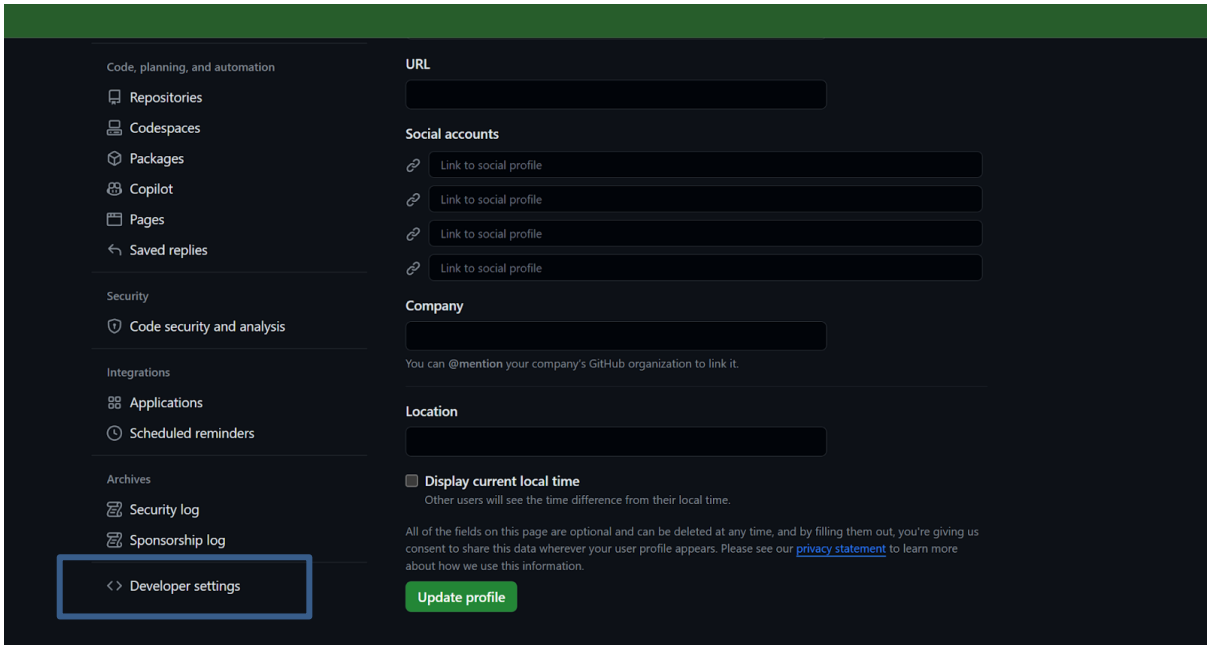

Figura 35. *Pantalla que muestra las opciones de configuración del usuario*

En este apartado crear un nuevo token personal, ingresar un nombre, seleccionar un tiempo de expiración y otorgar todos los permisos y finalmente guardar como se muestra en la figura:

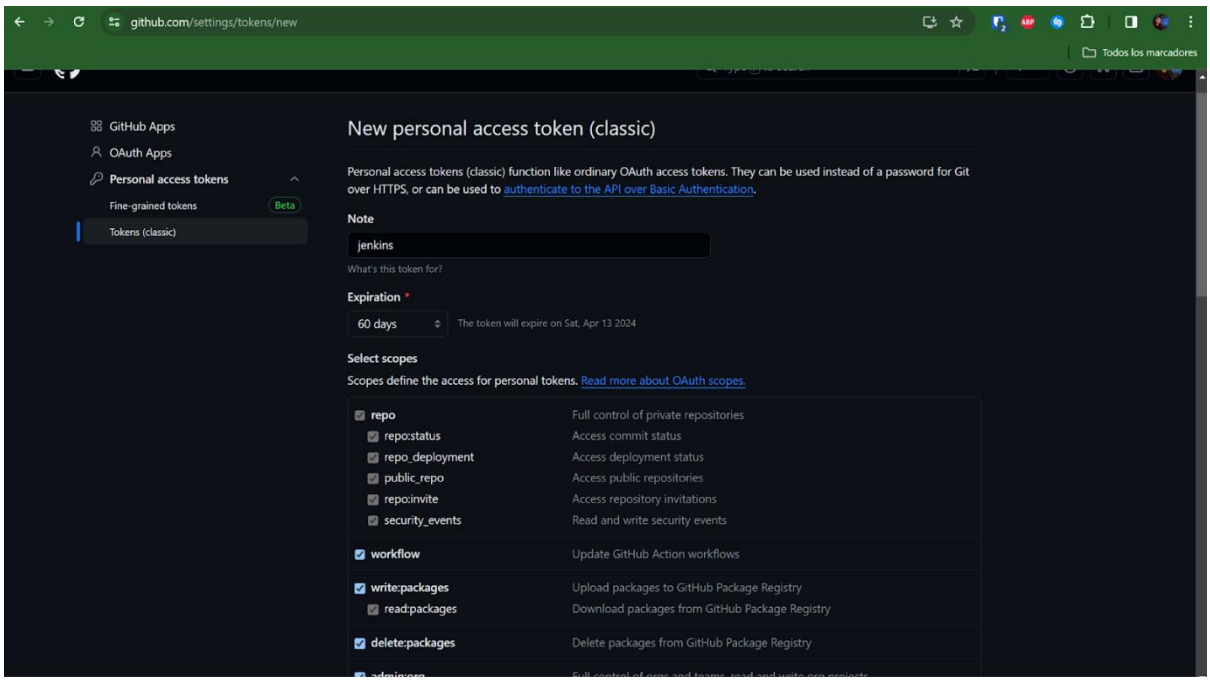

Figura 36. *Pantalla de creación de token*

Después de generar el token, es recomendable copiarlo y almacenarlo, ya que una vez que la pagina se recargue, no será posible visualizarlo nuevamente como se muestra en la figura:

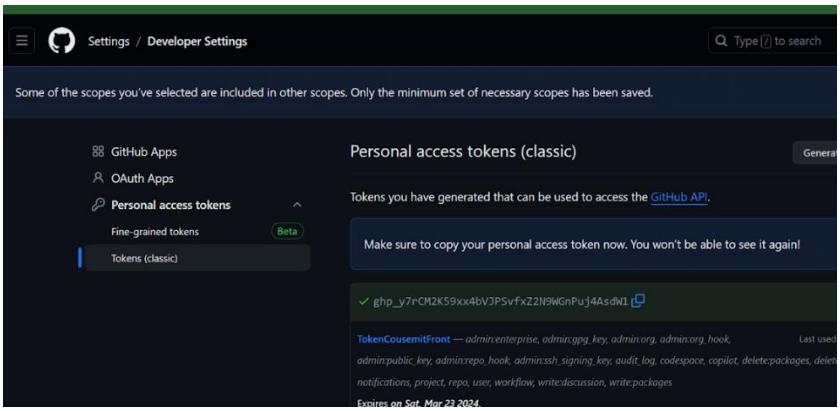

Figura 37. *Pantalla que muestra token generado*

Una vez generado el token, regresar a Jenkins y acceder a la gestión de credenciales de Jenkins para agregar la nueva credencial.

Presionar en "Agregar credencial". Aparecerá un formulario en el que se selecciona el tipo de credencial como "Nombre de usuario y contraseña", con un alcance global. Se ingresa el nombre de usuario de GitHub y se utiliza el token como contraseña. En el campo de ID, se puede proporcionar una descripción, luego se hace clic en "Crear" como se muestra en las siguientes figuras:

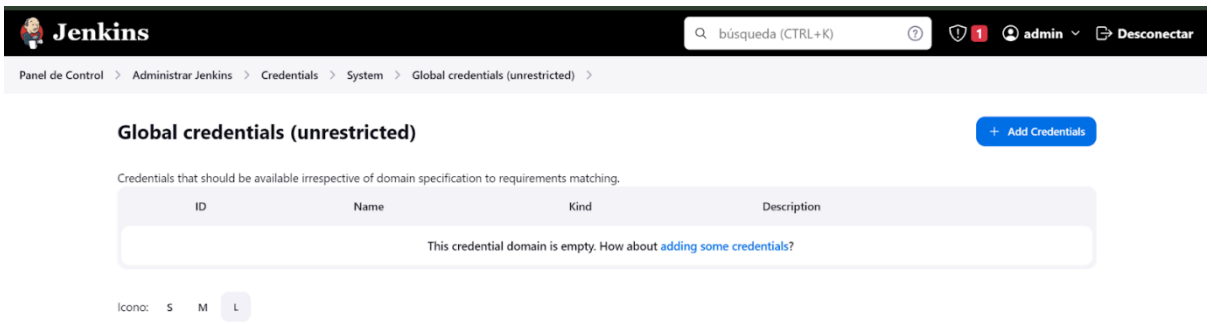

Figura 38. *Pantalla de listado de credenciales*

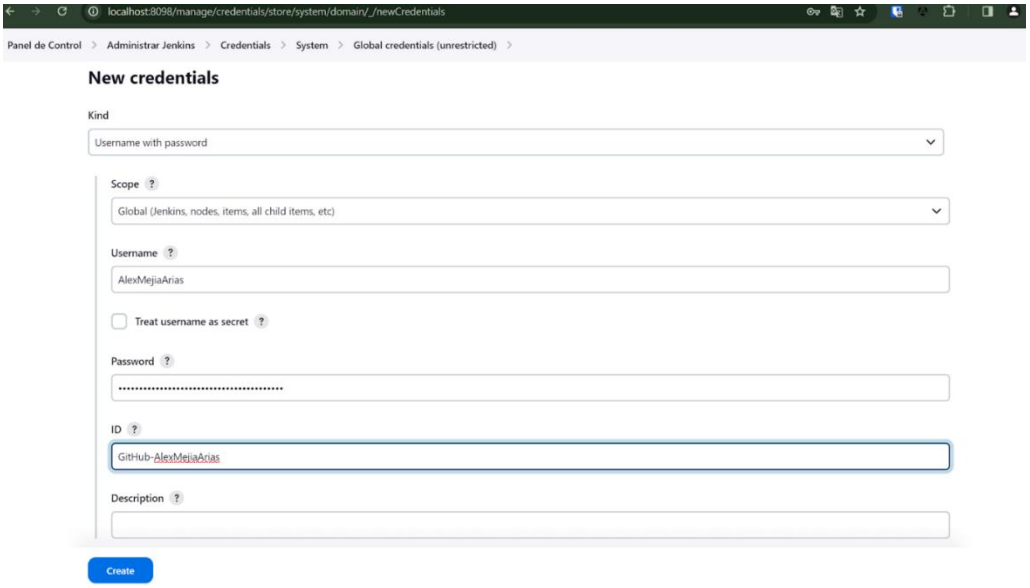

Figura 39. *Formulario de creación de nueva credencial*

Creación de pipeline

Procedemos a crear una nueva tarea, seleccionamos crear un proyecto de estilo libre e ingresamos un nombre como se muestra en la figura:

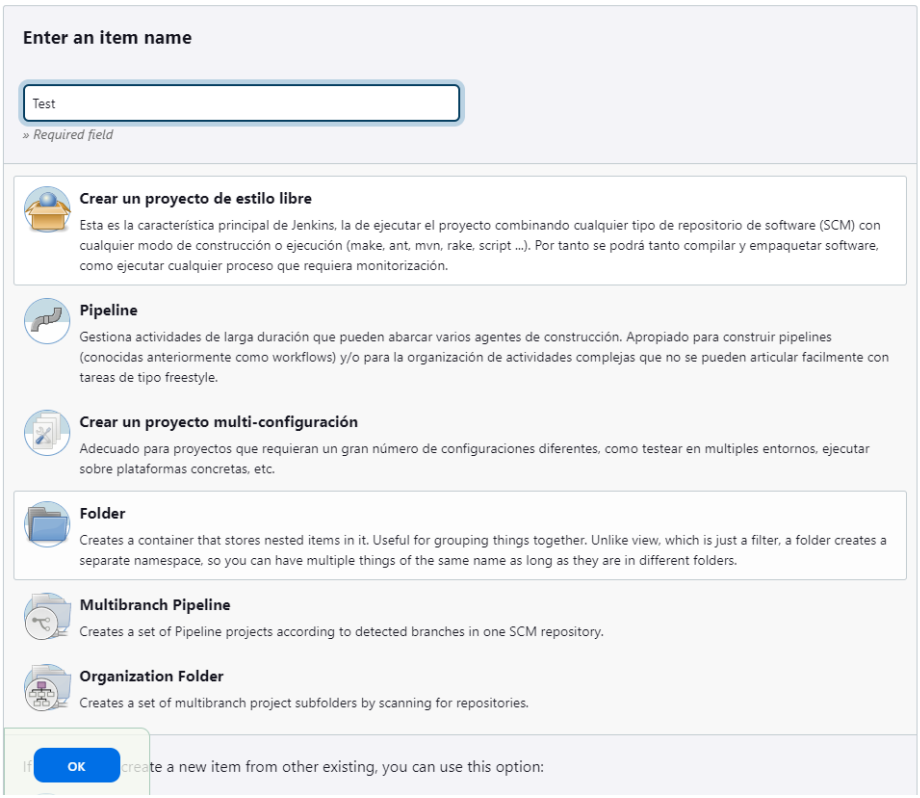

Figura 40. *Creación de Pipeline*

En la sección general, seleccionamos Github project y ponemos la url destinada, igual en el system ingresar lo siguiente:

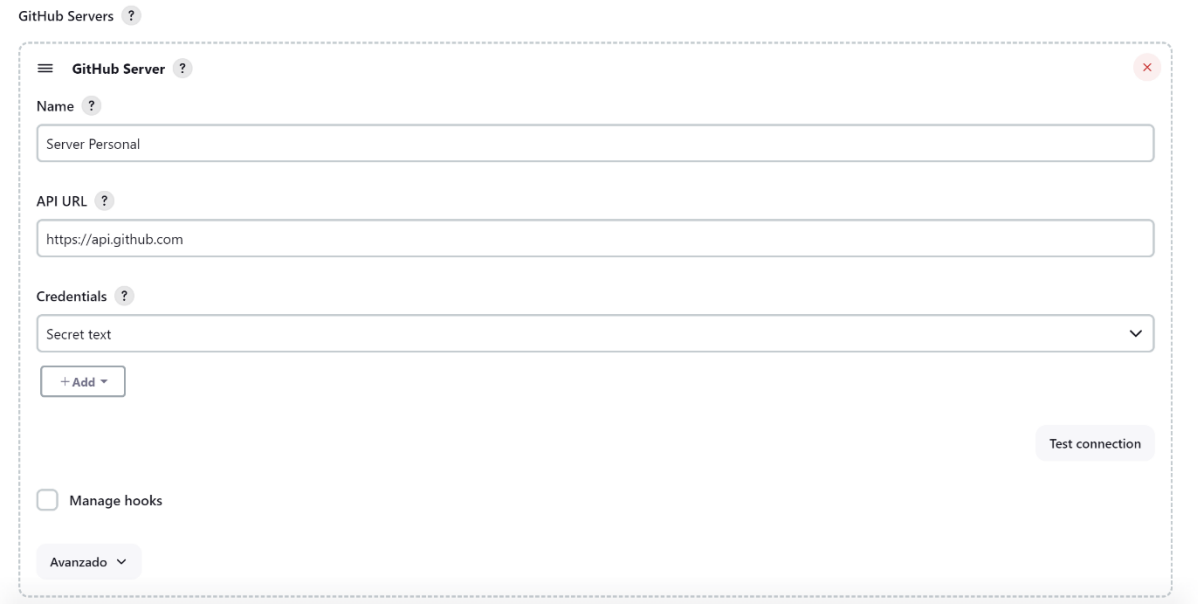

Figura 41. *Configuración de Pipeline*

En la configuración del origen del código fuente, seleccionamos la opción Git y se añade el enlace del clone HTTPS. Seleccionamos las credenciales de GitHub configuradas previamente y, por último, se especifica la rama correspondiente como se muestra en la figura:

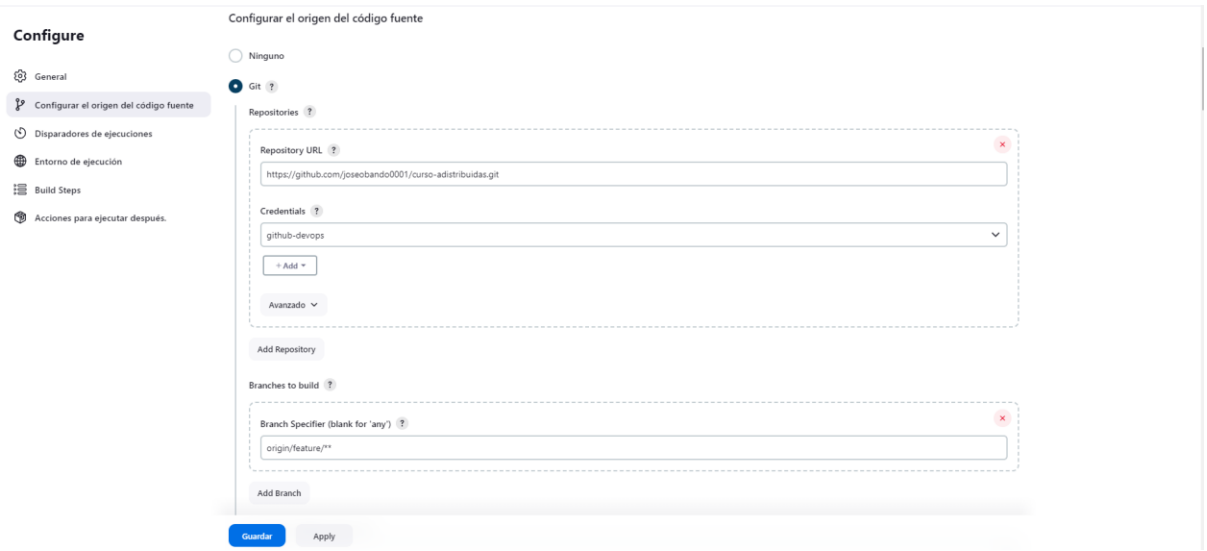

Figura 42. *Configuración de origen del codigo fuente*

Adicionalmente en aditional behaviours ponemos lo siguiente, esto con la finalidad de ponerle un nombre como se muestra en la figura:

Figura 43. *Usuario de Jenkins*

Adicionalmente, se cuenta con la manera de que se ejecute cada que se haga un commit y para eso habilitamos en Disparadores de ejecuciones, la opción GitHub hook trigger for GITScm polling como se muestra en la figura:

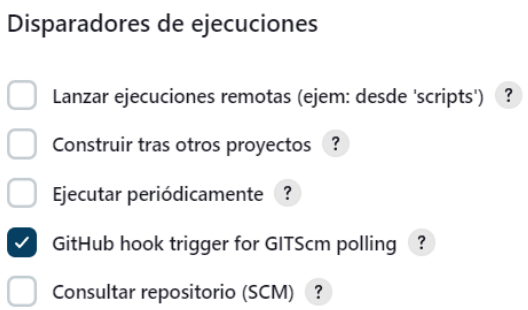

Figura 44. *Habilitación de Hook*

Spring Boot

En build steps empezaremos con la opcion de Gradle, check en el de utilizar Use Gradle Wrapper y en los task pondremos:

```
clean build
```

como se muestra en la imagen.

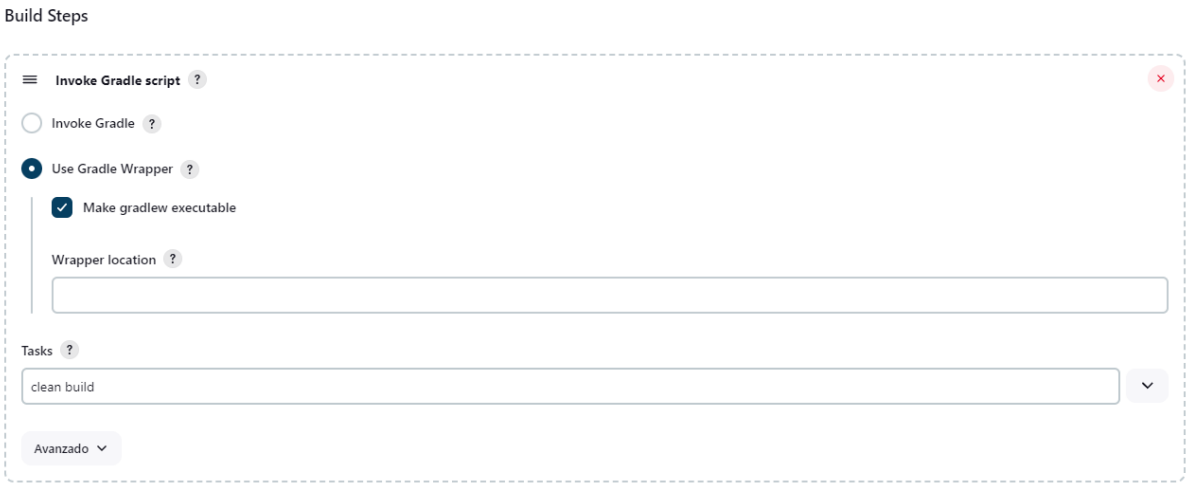

Figura 45. *Configuración de Gradle*

Angular

En build steps empezaremos con la opción de shell para instalar dependencias, realizar los test y construir el proyecto como se muestra en la imagen.

Figura 46. *Configuración de Angular*

Configuración de SonarQube

En el pipeline pondremos en Administrar Jenkins e instalaremos el plugin SonarQube Scanner for Jenkins como se muestra en la figura:

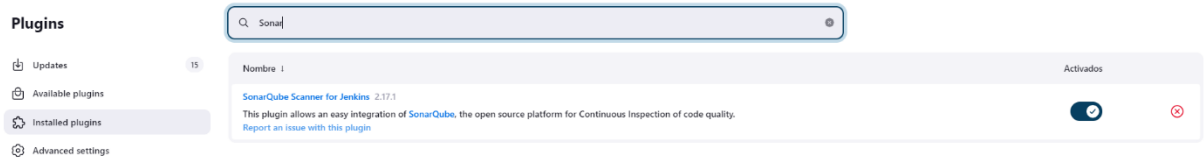

Figura 47. *Configuración de SonarQube*

También en el apartado de System en Admin Jenkins tendremos que ir al apartado de SonarQube servers e ingresar lo siguiente como se muestra en la imagen:

Figura 48. *Instalación en Jenkins de SonarQube*

Recordar poner la credencial correspondiente, donde se genera un token en nuestro SonarQube, en el apartado de Security como se muestra en la imagen:

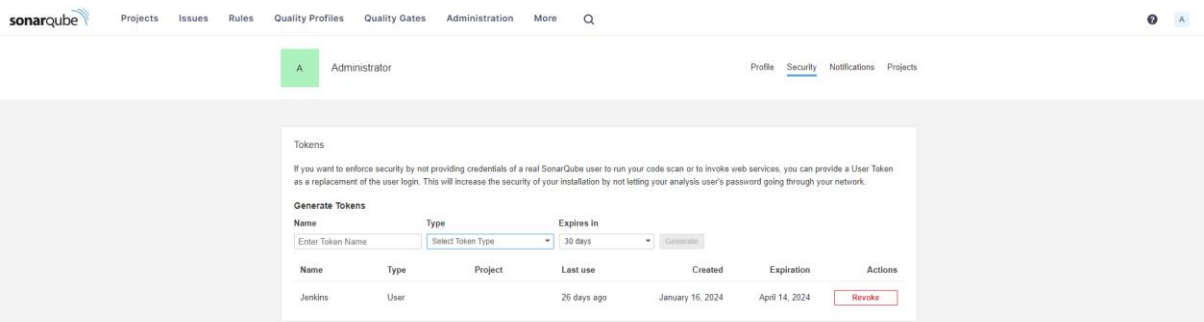

Figura 49. *Token de SonarQube*

En Tools igual adicionar lo siguiente:

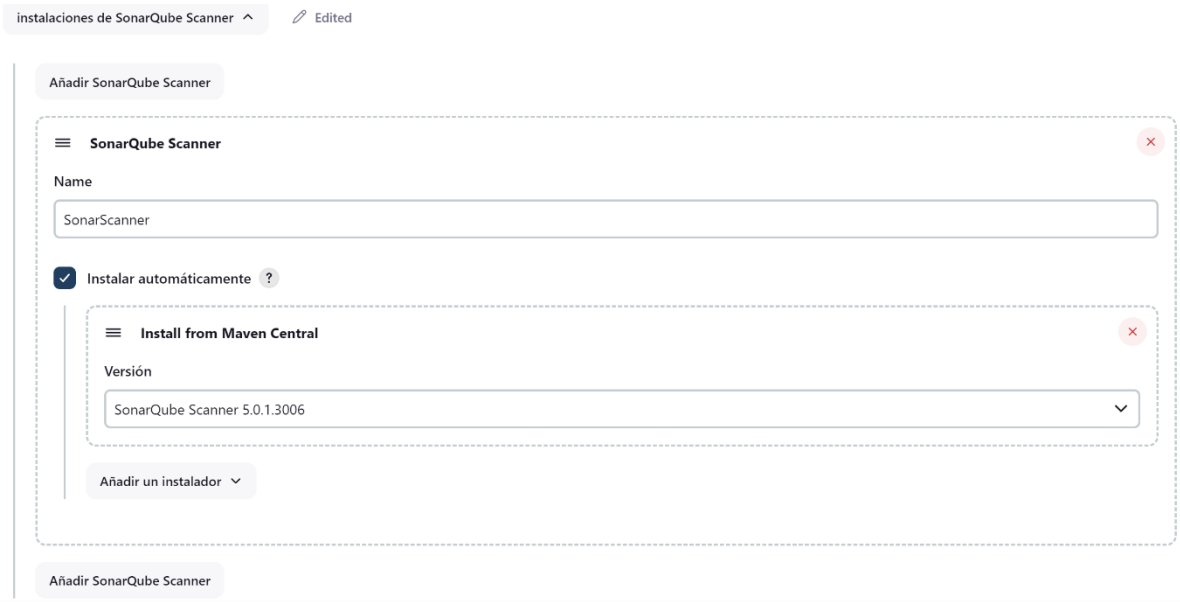

Figura 50. *Adición del escaneo por Sonar*

Gradle

Luego en la configuración del pipeline pondremos en build steps la opción execute SonarQube Scanner, las siguientes propiedades:

```
1. sonar.projectKey=sonarqube
2. sonar.sources=src/main/java
3. sonar.java.binaries=build/classes
```

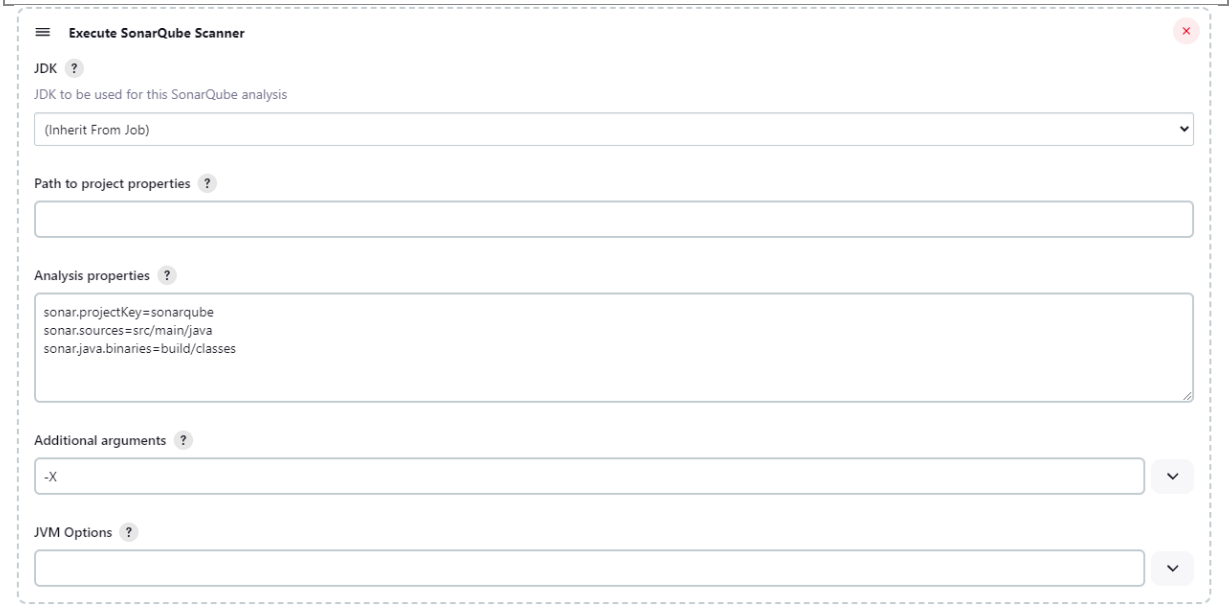

Figura 51. *Configuración de Sonar para Gradle*

Angular

Luego en la configuración del pipeline pondremos en build steps la opción execute SonarQube Scanner, las siguientes propiedades:

```
1. sonar.projectKey=coursemit-front
2. sonar.projectVersion=1.0
3. sonar.sources=src
```

Agregar Docker

Primeramente, en Tools pondremos en Instalaciones de Docker y seleccionaremos lo siguiente como se muestra en la figura.

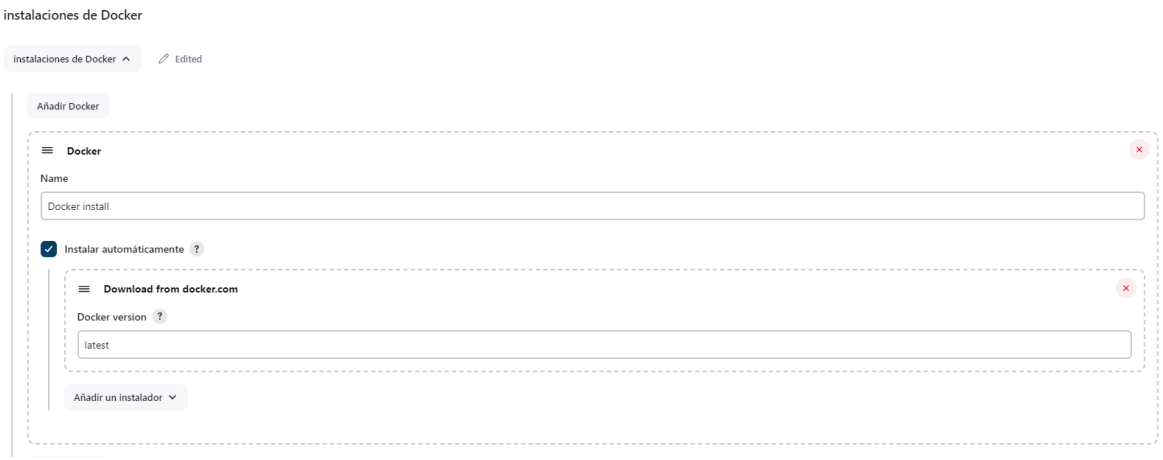

Figura 52. *Adición de Docker en Jenkins*

También instalaremos CloudBees Docker Build and Publish, que se mostrara de la siguiente manera:

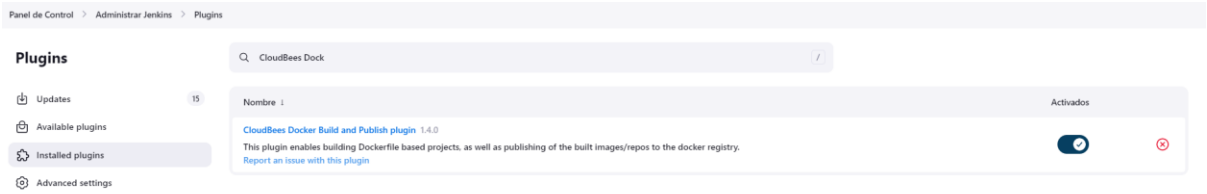

Figura 53. *Adición de Plugin de Docker*

En el pipeline nos mostrara una opción Docker Build and Publish y pondremos las siguientes configuraciones, igual adicionalmente poner las credenciales de DockerHub como se muestra en la imagen:

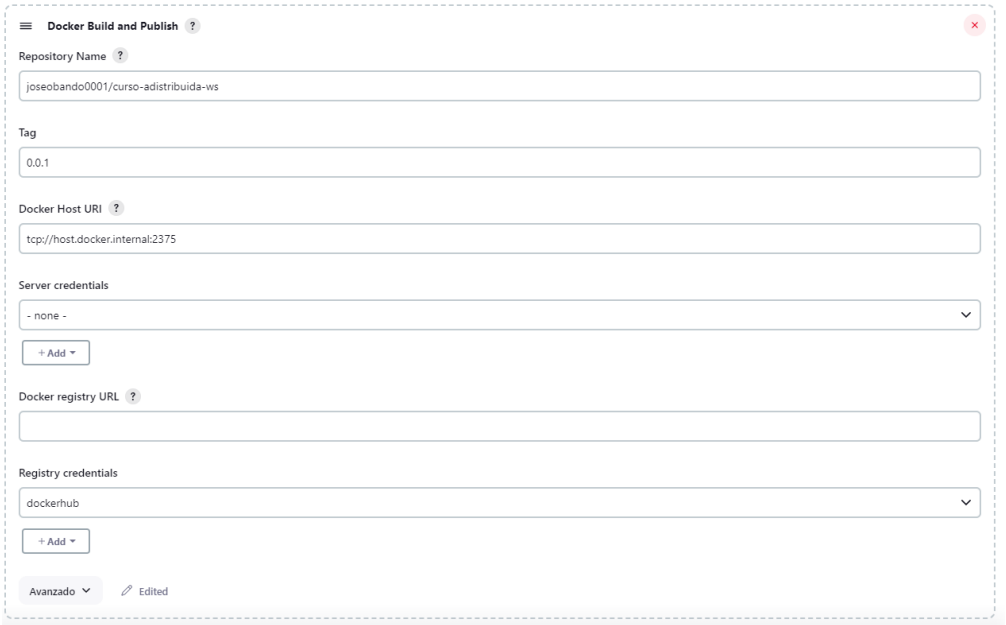

Figura 54. *Configuración de publicación en DockerHub*

Recordar activar la siguiente opción llamada Expose Daemon on tcp: para poder publicar y utilizar las funciones de Docker como se muestra en la imagen:

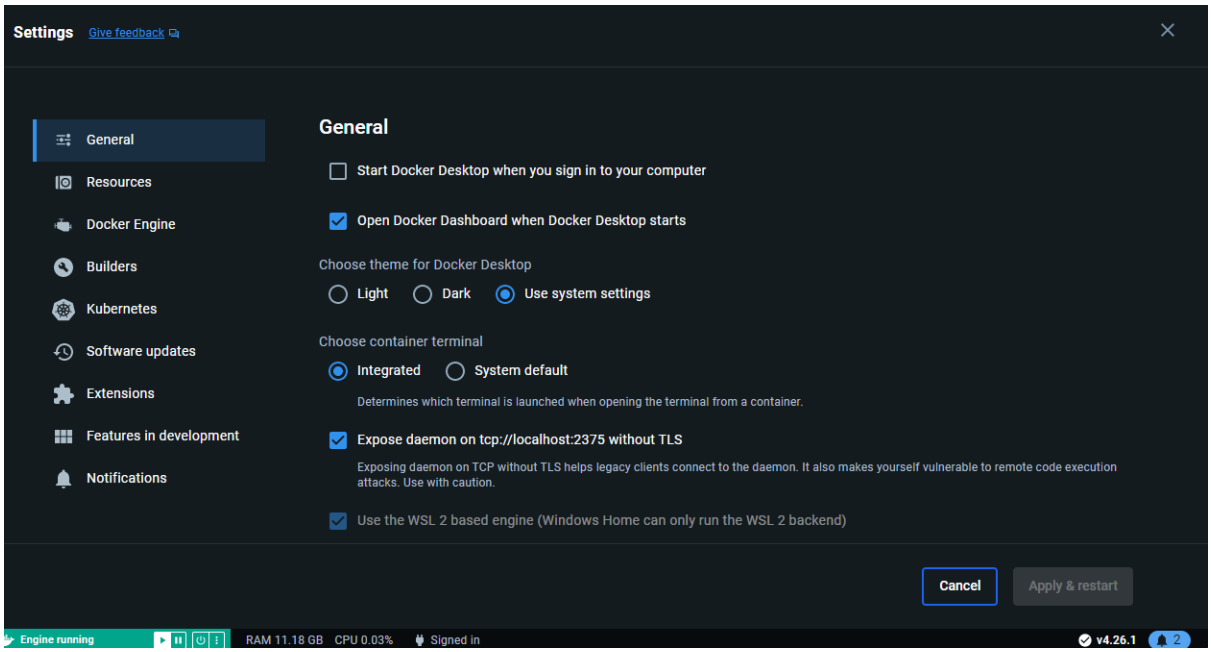

Figura 55. *Habilitación de Exposición de Docker*

Finalmente agregamos la opción de Git Publisher y debería estar de la siguiente manera:

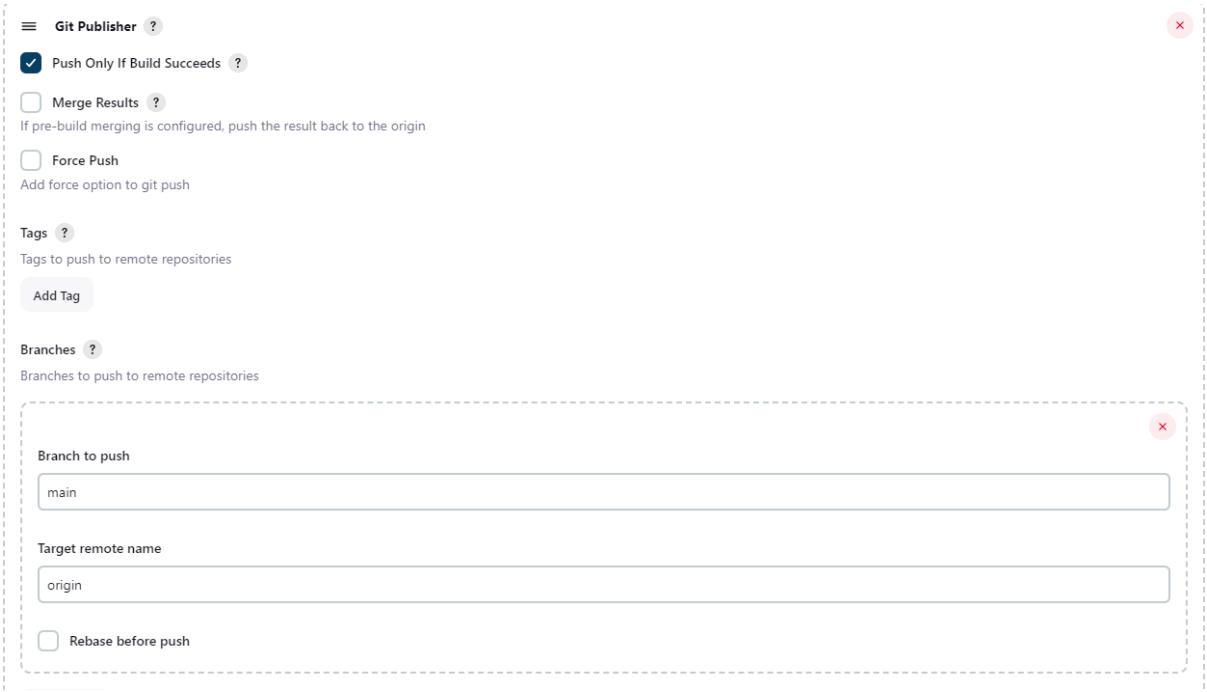

Figura 56. *Publicación*

Agregar Slack

Agregamos el plugin Slack Notification Plugin en Plugins.

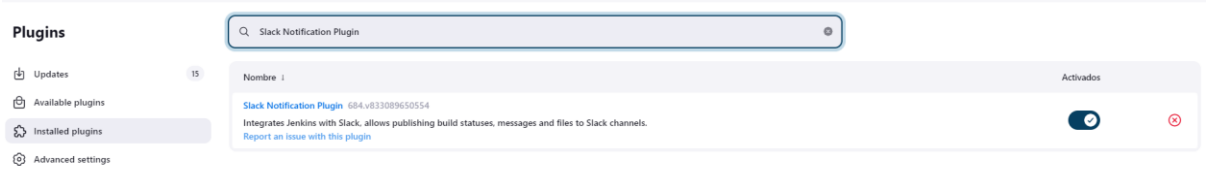

Figura 57. *Plugin de Slack*

En la aplicación de Slack se pondrá en el apartado de aplicaciones y seleccionaremos Jenkins CI como se muestra en la figura:

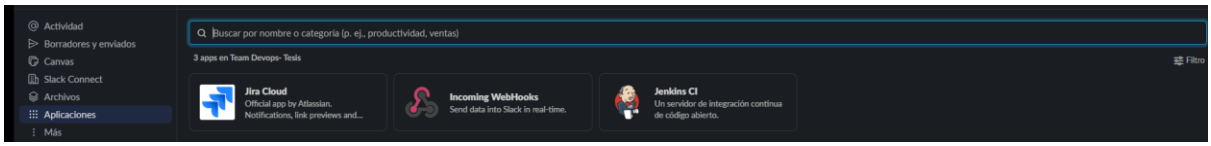

Figura 58. *Configuración de Slack*

Agregamos nuestro canal de preferencia y en el pipeline agregaremos lo siguiente:

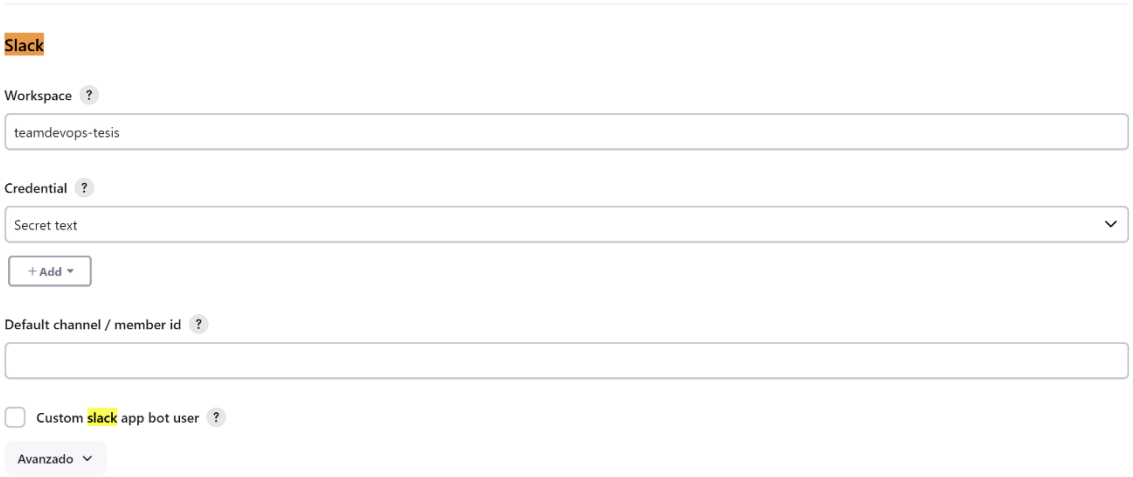

Figura 59. *Configuración en Pipeline de Slack*

Pipeline de CI (Integración Continua)

Una vez configurado, se procede a ejecutar el pipeline de CI, donde muestra con éxito el tiempo que le toma completar como se muestra en la siguiente imagen:

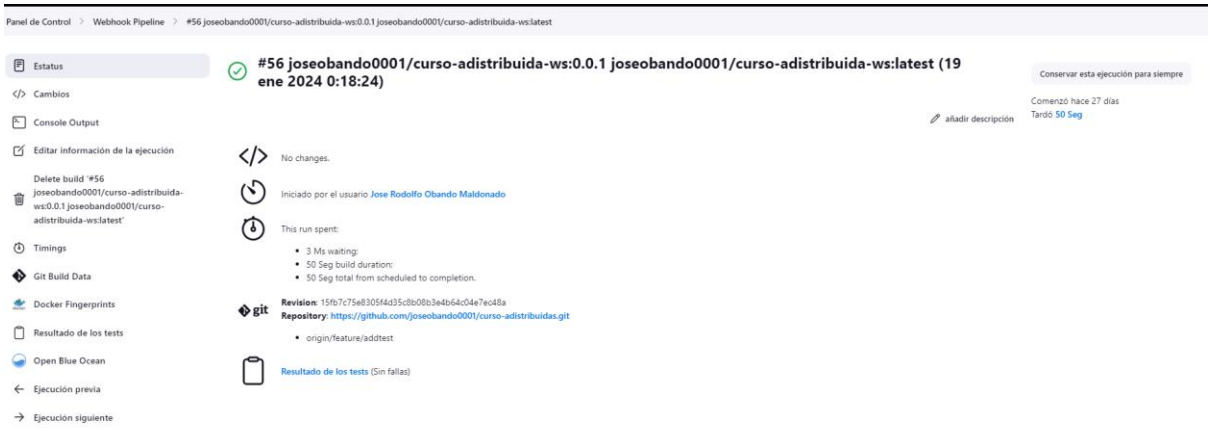

Figura 60. *Pipeline de CI*

Además, se puede observar los logs de cómo fue todo el proceso de CI como se muestra en la siguiente figura:

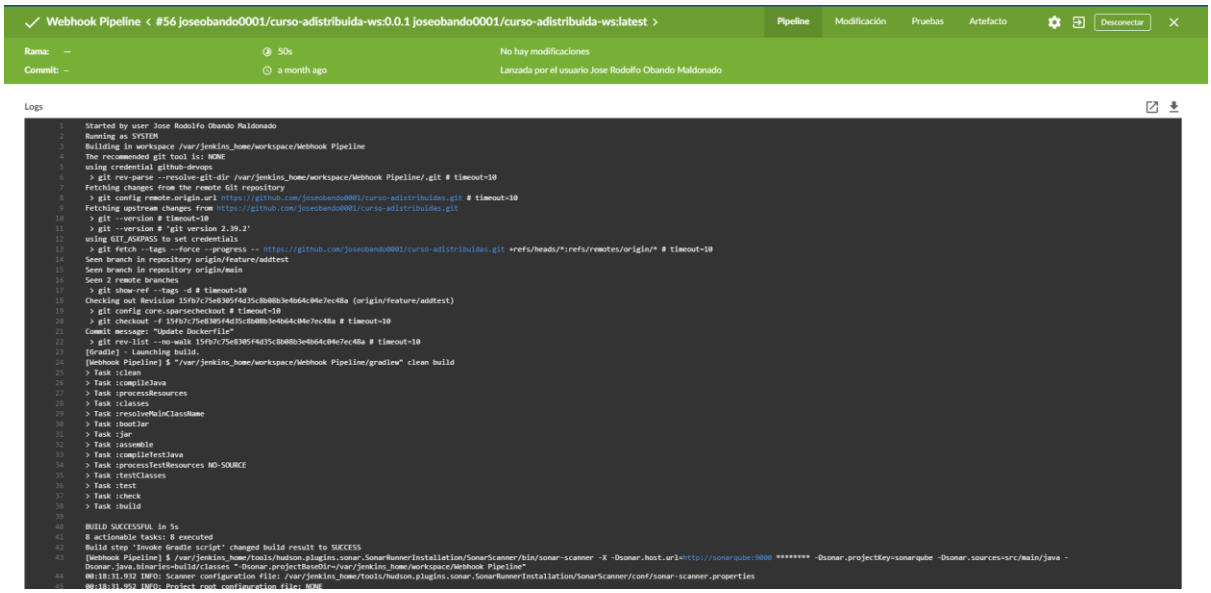

Figura 61. *Logs de CI*

Pipeline de CD (Despliegue Continuo)

En este apartado se explicará la configuración realizada, para la ejecución del despliegue continuo.

Primero se debe poner la url de Github de la siguiente manera:

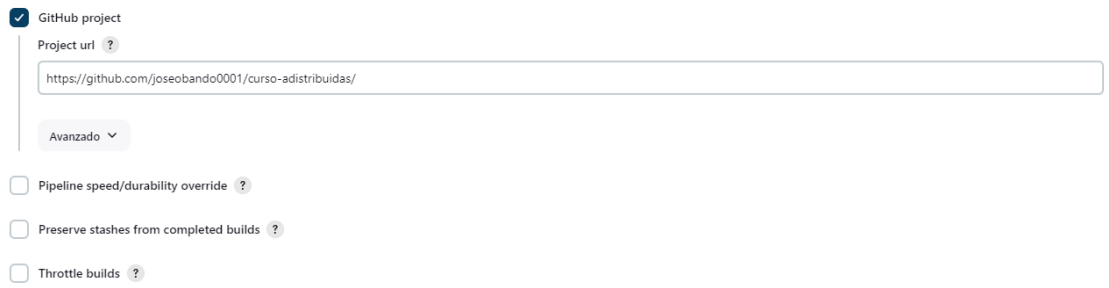

Figura 62. *Configuración de CD*

Adicionalmente, se agrega las siguientes configuraciones como se muestra en las siguientes figuras:

Figura 63. *Configuración de Pipeline CD*

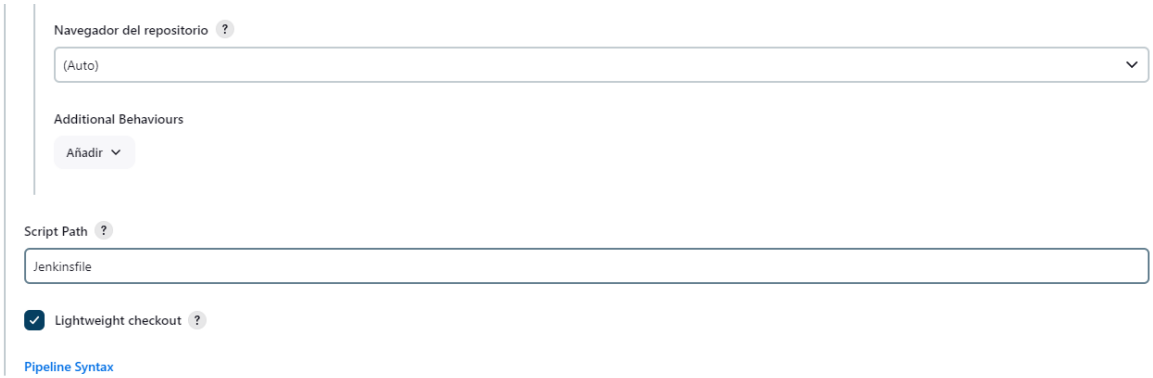

Figura 64. *Configuración de Script para CD*

Finalmente, una vez configurado, se procede a mostrar la ejecución exitosa, la cual se muestra de la siguiente manera:

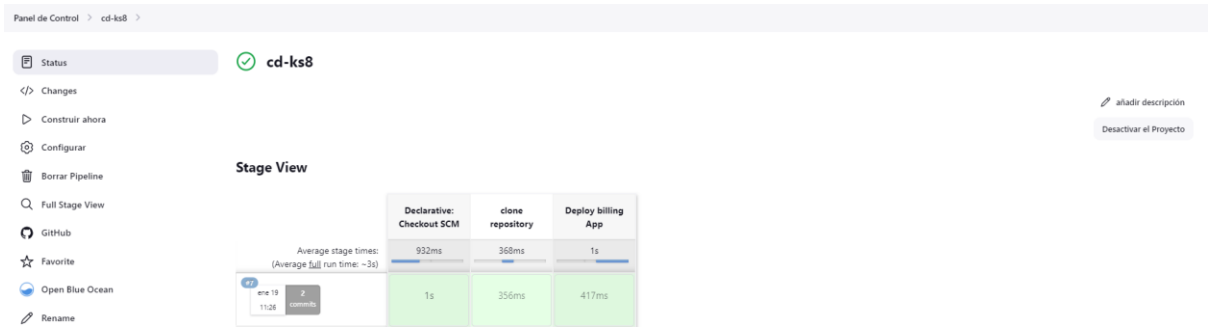

Figura 65. *Ejecución de CD*

Adicionalmente, se puede observar logs y de otra forma más amigable como se muestra en la siguiente figura:

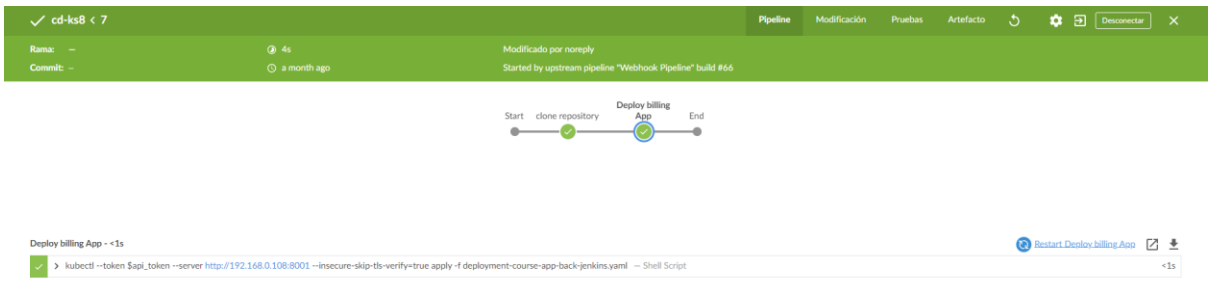

Figura 66. *Ejecución de CD en Blue Ocean*

Pruebas

En la siguiente sección se explican las técnicas de prueba desarrolladas y la aplicación de estas en el caso de estudio.

Técnicas de pruebas

La técnica de pruebas, que se aplicaron en la presente investigación son de caja blanca mediante la herramienta SonarQube, la cual nos permite realizar un análisis, para evaluar código fuente y obtener métricas de confiabilidad, mantenibilidad y seguridad, que nos permitirá mejorar la calidad del código y también la de probar la integración de código, compilación, pruebas y despliegue para proyectos de software.

Pruebas realizadas

Al ejecutar el pipeline que contiene las fases de CI (Integración continua), podemos mostrar toda la serie de pasos que realiza para poder ejecutar las pruebas y enviarlas hacia SonarQube como se muestra en la siguiente imagen:

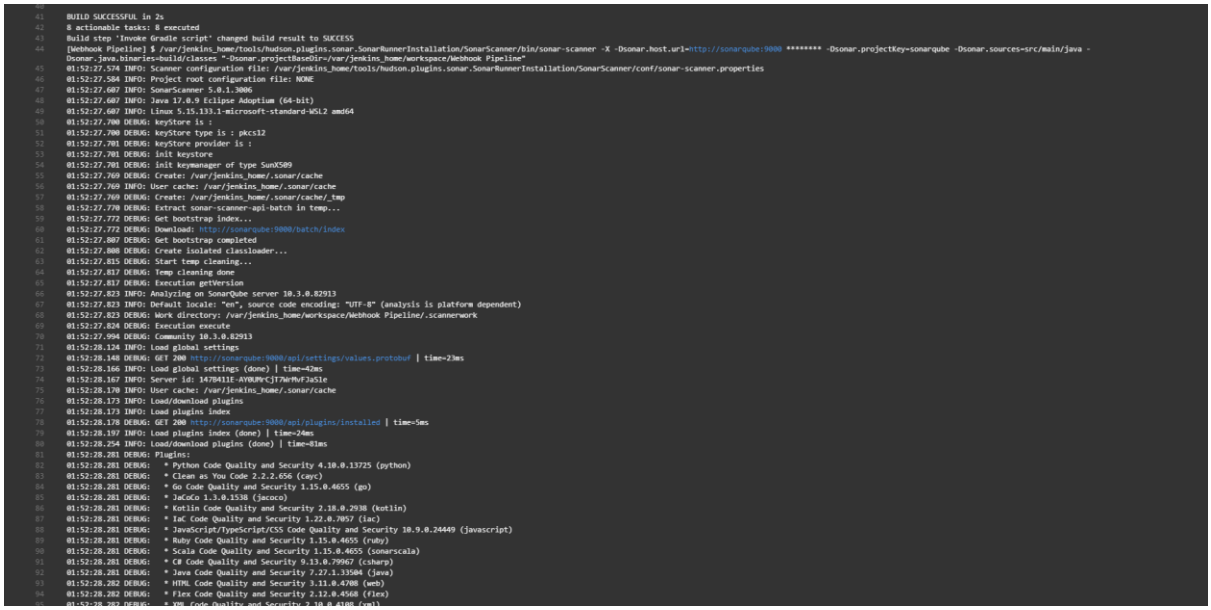

Figura 67. *Pruebas de SonarQube*

Aquí la parte donde finaliza.

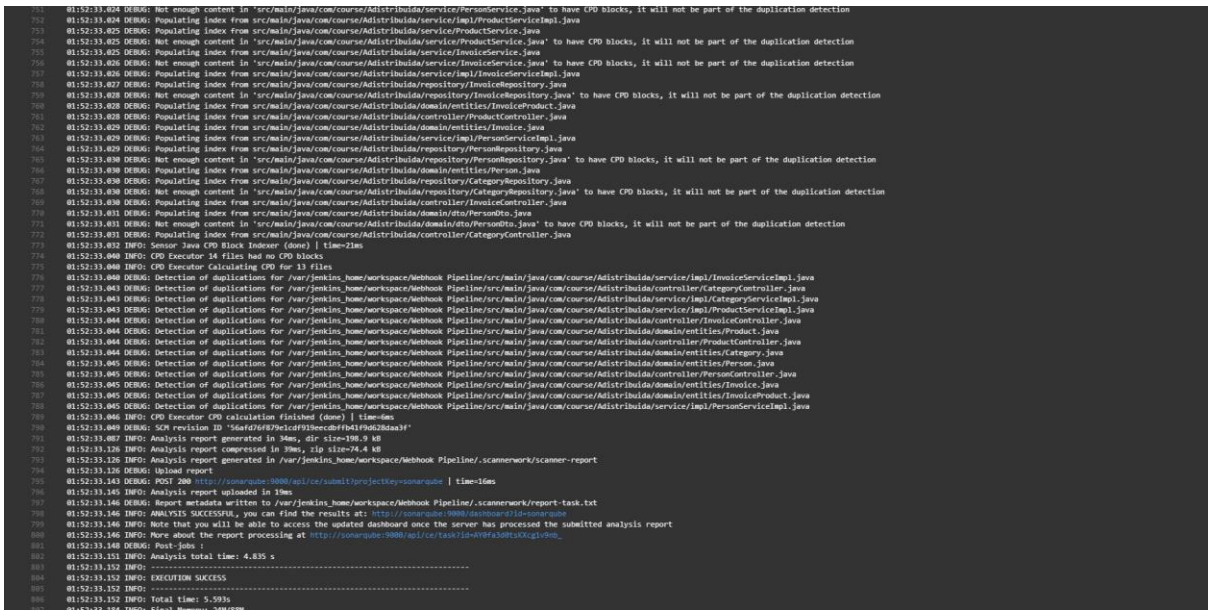

Figura 68. *Finalización de Prueba en logs*

También podemos mostrar las pruebas realizadas hacia la limpieza y construcción del código localmente y cuanto toma realizarlo, como se muestra en la siguiente imagen:

Figura 69. *Ejecución de clean para Spring Boot*

Figura 70. *Ejecución de build para Spring Boot*

Al igual que el despliegue localmente

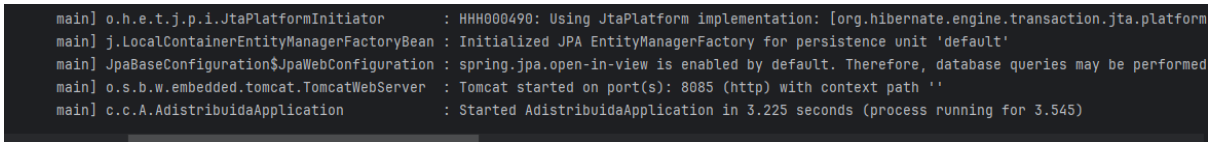

Figura 71. *Despliegue local de Spring Boot*

Mientras que en el pipeline con Integración Continua podemos demostrar que la construcción y limpieza toma el siguiente tiempo:

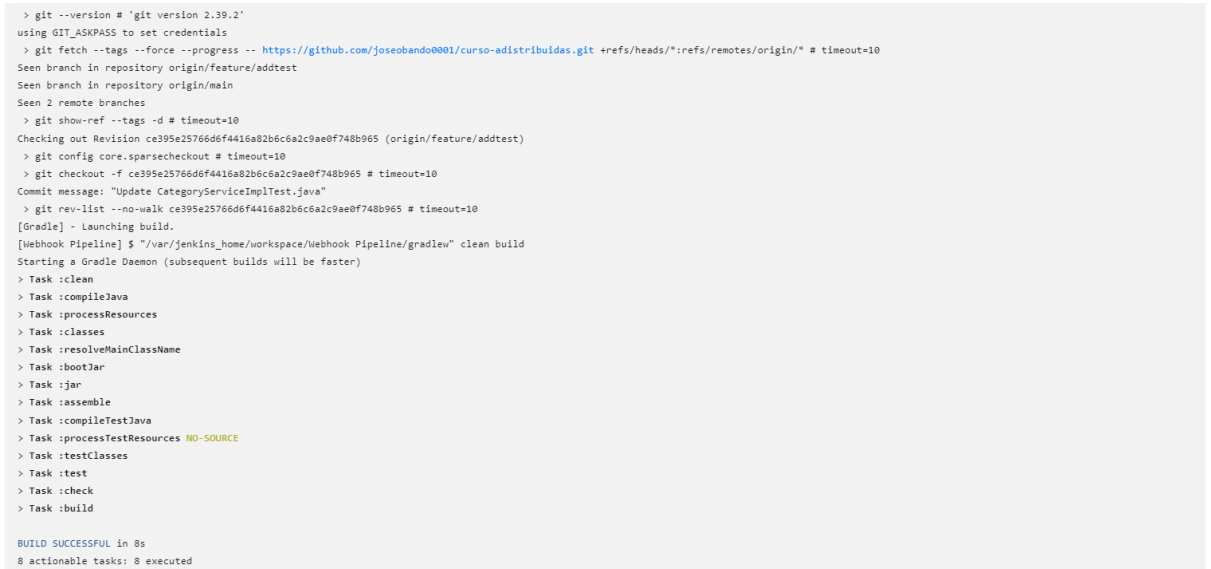

Figura 72. *Clean y Build de CI*

Y desplegado, se encuentra el siguiente resultado:

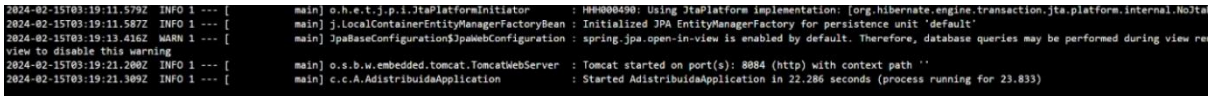

Figura 73. *Proyecto desplegado con CD*

Resultados obtenidos

Finalmente podemos observar por un lado en el SonarQube, el proyecto con todo pasado.

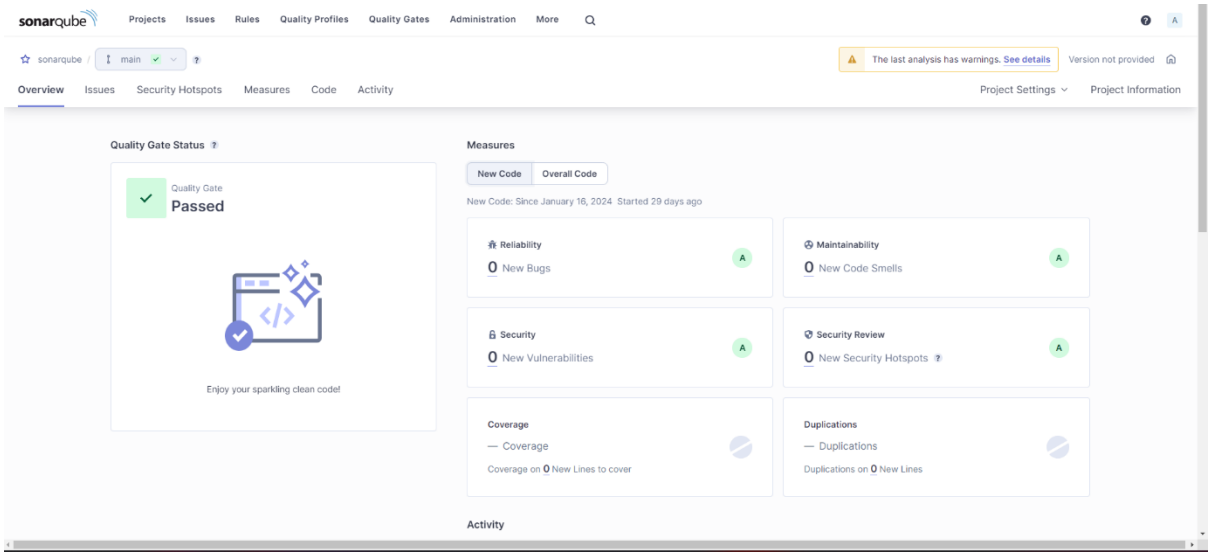

Figura 74. *Resultados de SonarQube*

Además de que también nos refleja los resultados de las pruebas en el pipeline.

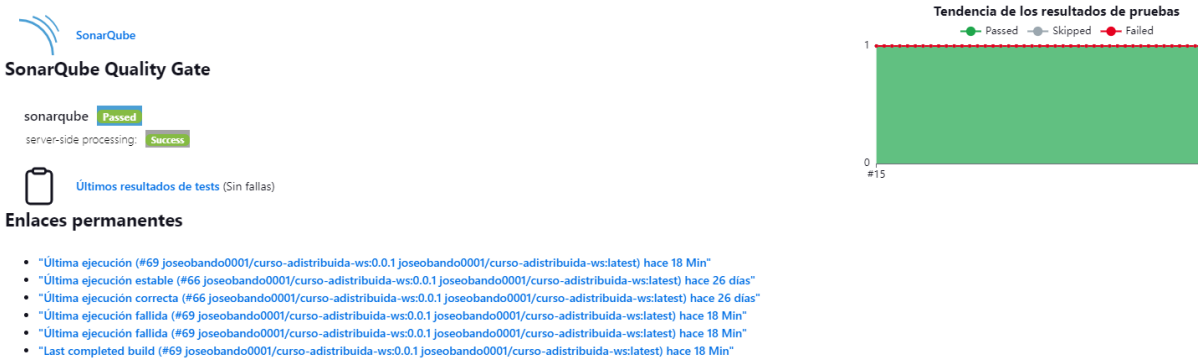

Figura 75. *Pruebas de Sonar en Pipeline CI*

Para la construcción, pruebas y despliegue se muestra la siguiente tabla en relación con la integración continua y despliegue continuo en tiempo, realizando una comparación con la construcción y despliegue localmente.

Tabla 15. *Resultados de pruebas*

# de Pruebas	**Local (Construcción y Despliegue)**	**Integración Continua (Construcción)**	**Despliegue Continuo**
1	42s	12s	17s
2	29s	5s	15s
3	25s	2s	10s
Promedio	32s	6.33s	14s

Para concluir se determina que mediante SonarQube se puede garantizar métricas de calidad y además mediante las pruebas de construcción, pruebas y despliegue también se demuestra una mejoría significativa al utilizar integración continua y despliegue continuo.

CAPITULO IV

ANÁLISIS E INTERPRETACIÓN DE LOS RESULTADOS

En este capítulo se muestra el resultado de la investigación en la que se consideraron los criterios de inclusión y exclusión detallados anteriormente, utilizando la técnica de la matriz de análisis que se encuentra a continuación:

Tabla 16. *Matriz de análisis*

Autor	Titulo	Fecha	Aporte	Análisis o interpretación
Tom Hall	¿En qué consiste la cultura de DevOps?	14/11/2023	Comprensión Profunda de DevOps.	El artículo explica de manera clara y detallada lo que implica la cultura de DevOps, no sólo en términos de herramientas y prácticas, sino también en la importancia de la mentalidad, los hábitos y la cultura organizacional.
Diego Cortez	Introducción a CI/CD	27/07/2023	Proceso y etapas de CI/CD	El artículo permite conocer las etapas del flujo de trabajo entre lo que se encontró de CI/CD
Hernández Salazar Edwin, Beltrán Carlos Alberto	SCRUM, Un enfoque práctico de metodología ágil para la ingeniería de software	2020	Metodología SCRUM	El artículo permite reconocer a SCRUM como un marco de trabajo para crear un ambiente colaborativo en proyectos de software, enfocando su criterio en crear equipos altamente eficientes y productivos.

Autor	Titulo	Fecha	Aporte	Análisis o interpretación
JetBrains	¿Qué es CI/CD en DevOps?	27/07/2023	CI/CD en DevOps	El artículo aporta una comprensión completa de cómo CI/CD en el marco de DevOps aborda los desafíos del desarrollo de software al proporcionar eficiencia, calidad y entrega continua de valor a los usuarios.
Jonathan Guerrero, Karen Zúñiga, Camilo Certuche y César Pardo	A systematic mapping study about DevOps	2020	Adopción de DevOps en empresas de desarrollo de software.	El artículo contribuye a la comprensión de los desafíos y oportunidades asociados con la adopción de DevOps en empresas de desarrollo de software. Además, destaca la falta de información detallada sobre actividades, tareas, roles y otros elementos importantes del proceso para la adopción de DevOps, así como la falta de una terminología unificada.
Juan Carlos Mercedes Brito	Integración y entrega continua (CI/CD) con Jenkins	2022	Uso de Jenkins	Este artículo describe las ventajas de implementar estas técnicas, especifica las diferentes herramientas utilizadas en este ámbito y profundiza en el uso de Jenkins, una de las herramientas de integración continua más ampliamente utilizada
Ken Schwaber y Jeff Sutherland	The Scrum Guide	2020	Guía de Scrum	Este artículo ofrece una guía clara y flexible para la implementación de Scrum, resaltando la importancia de la transparencia y la mejora continua. Con roles bien

Autor	Titulo	Fecha	Aporte	Análisis o interpretación
				definidos, valores clave y una licencia Creative Commons, la guía promueve la colaboración efectiva en el desarrollo ágil, enfatizando la adaptabilidad y el aprendizaje continuo.

Igualmente se muestra un gráfico representativo de los documentos incluidos y excluidos en la investigación.

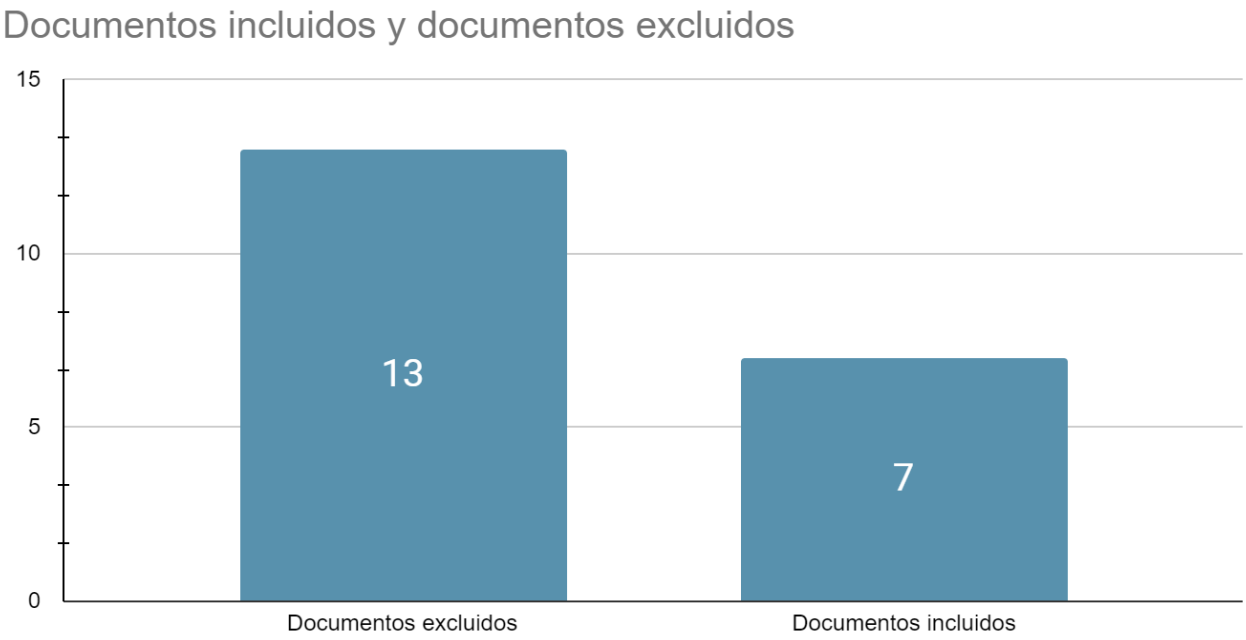

Figura 76. *Gráfico de barras*

Se revisaron un total de 20 documentos, de estos se excluyeron 13 ya que no tenía información sobre el tema tratado o porque no cumplía con los criterios de inclusión. En cambio, se seleccionaron 7 documentos que presentaron una visión completa en distintos ámbitos.

Se abordo la cultura de DevOps, destacando tanto herramientas como prácticas y se enfatiza los desafíos de desarrollo con eficiencia y entrega continua de valor.

Además, la introducción y el proceso de CI y CD son detallados y desglosado en etapas, SCRUM se presenta como un marco de trabajo para equipos eficientes en desarrollo de software y ofrece claridad en roles promoviendo transparencia y mejora continua en el desarrollo ágil.

Resultados del producto

En este apartado se detallan los resultados obtenidos a través de las pruebas realizadas en el estudio de caso, en este modo se tomó en cuenta en segundos, tal como se muestra en la Tabla 15.

Para medir se tomó 3 ejecuciones entre las cuales con el cálculo de la suma y la división del número total se obtuvo el resultado de cada ámbito, tanto como fue el de la construcción local y despliegue del código, al igual que la integración y despliegue continuo, donde se logró sacar el promedio mediante la siguiente formula:

Donde

P: Promedio

n: Intentos

$$P = \frac{n1 + n2 + n3}{ntotal}$$

Finalmente resulto tener más tiempo el uso local de los casos de uso, mientras que con el uso de la integración continua y despliegue continuo se logró alcanzar a reducir bastante tiempo, adicionalmente que en el apartado del despliegue continuo no tiene riesgo de cerrarse por error, ya que el mismo queda habilitado para su uso sin ningún problema, como lo tendría en el despliegue local.

CAPITULO V
CONCLUSIONES Y RECOMENDACIONES

En el siguiente capítulo se detallan los hallazgos y reflexiones del presente trabajo de titulación.

Conclusiones

- En el presente trabajo de titulación se logró investigar las mejores prácticas mediante el marco de trabajo DevOps, el ámbito de la integración continua y despliegue continuo y logrando efectuar un gran hallazgo con toda la revisión documental.
- Se lograron definir todos los principales componentes, configuraciones de las herramientas necesarias, al igual que las pruebas y la gestión de todos los flujos de trabajo, mediante el uso de Jenkins, Docker y Minikube como principales herramientas.
- Se logro desarrollar un caso de estudio con CI/CD mediante el cual se logró levantar un proyecto backend y frontend, para posteriormente medir con éxito la aplicación efectiva del trabajo de titulación.

Recomendaciones

- Se recomienda utilizar un servicio de nube como Amazon Web Services, Google Cloud Platform, para el uso de las herramientas y no tener inconvenientes con puertos locales.
- Se recomienda mantener actualizadas las herramientas para tener un sobresaliente funcionamiento para los proyectos de software.
- El utilizar un flujo de trabajo como GitFlow o Git Trunk-Based, para mejorar la integración continua y despliegue continuo en cualquier proyecto de software a utilizar.

REFERENCIAS

Red Hat. (2022, 11 mayo). *La integración y la distribución continuas (CI/CD)*. *Recuperado 15 de noviembre de 2023, de,* https://www.redhat.com/es/topics/devops/what-is-ci-cd

Atlassian. (2023, 15 noviembre). *Entrega continua: Primeros pasos con CI/CD* | Atlassian. *Recuperado 15 de noviembre de 2023, de* https://www.atlassian.com/es/continuous-delivery

Farcic, V. (2016). *The DevOps 2.0 Toolkit: Automating the Continuous Deployment Pipeline with Containerized Microservices*. *Recuperado 15 de noviembre de 2023, de* https://www.amazon.com/DevOps-2-0-Toolkit-Containerized-Microservices/dp/152391744X

IBM. be79d0beafad

Fowler, M. F. (2006). *Continuous Integration* (1.ª ed.). Addison-Wesley.

Atlassian. (2023, noviembre 15). *¿Qué es DevOps?. Recuperado 15 de noviembre de 2023, de* https://www.atlassian.com/es/devops

Donetonic. (2023, 16 noviembre). *Qué son los eventos en Scrum*. DoneTonic. *Recuperado 16 de noviembre de 2023, de* https://donetonic.com/es/que-son-los-sprints-en-scrum/

NimbleWork. (2023, 13 julio). *¿Qué es la metodología Scrum? y gestión de proyectos ScRum*. Nimblework. *Recuperado 16 de noviembre de 2023, de* https://www.nimblework.com/es/agile/que-es-scrum/

Solís, L. D. M. (2019, 28 julio). *Diseños de investigaciones con enfoque cuantitativo de tipo no experimental*. Investigalia. *Recuperado 16 de noviembre de 2023, de* https://investigaliacr.com/investigacion/disenos-de-investigaciones-con-enfoque-cuantitativo-de-tipo-no-experimental/

Meneses, N. (2023, junio). *Git vs GitHub (Diferencias y características). Recuperado 23 de noviembre de 2023, de* https://www.codingdojo.la/2023/06/09/git-vs-github-diferencias-y-caracteristicas/

Cristancho, F., Cristancho, F., & Cristancho, F. (2022, 1 agosto). *¿Qué es Angular y para qué sirve?* - Talently. Talently Blog. *Recuperado 23 de noviembre de 2023, de* https://talently.tech/blog/que-es-angular/

Miro. (2024, 1 febrero). *Diagrama de componentes UML: qué es y cómo hacerlo | Miro.* https://miro.com/. Recuperado 1 de febrero de 2024, de https://miro.com/es/diagrama/que-es-diagrama-componentes-uml/

Arias, F. (2016). *Proyecto de Investigación. Introducción a la metodología de la científica.* Caracas *:* Episteme*.*

Gonçalves, M. J. (2024, enero 17). ¿Qué es Angular y para qué sirve? Blog de Hiberus. Recuperado 12 de febrero de 2024, de https://www.hiberus.com/crecemos-contigo/que-es-angular-y-para-que-sirve/

Red Hat. (2023, 23 enero). ¿Qué es Docker y cómo funciona? Ventajas de los contenedores Docker. Recuperado 12 de febrero de 2024, de https://www.redhat.com/es/topics/containers/what-is-docker

Iglesias, J. (2022, 7 octubre). ¿Qué es y para qué sirve Notion? ¿QUÉ ES NOTION Y PARA QUÉ SIRVE? Recuperado 12 de febrero de 2024, de https://javieriglesias.marketing/blog/que-es-y-para-que-sirve-notion

Equipo editorial de IONOS. (2020, 20 octubre). Minikube: lo mejor de Kubernetes con el mínimo esfuerzo. IONOS Digital Guide. Recuperado 12 de febrero de 2024, de https://www.ionos.es/digitalguide/servidores/herramientas/kubernetes-minikube/

Terreros, D. (2023, 20 enero). Qué es Slack, para qué sirve y cómo utilizarlo. Qué Es Slack, Para Qué Sirve y Cómo Utilizarlo. Recuperado 12 de febrero de 2024, de https://blog.hubspot.es/marketing/guia-slack

Eduardo. (2021, 10 diciembre). Como utilizar la base de datos H2 en Spring boot - eduesqui - Medium. Medium. Recuperado 20 de febrero de 2024, de https://medium.com/eduesqui/como-utilizar-la-base-de-datos-h2-en-spring-boot-db1241d1f7f3

Team, K. (2022, 30 marzo). ¿Qué es Kubectl? | KeepCoding Bootcamps. KeepCoding Bootcamps. Recuperado 20 de febrero de 2024, de https://keepcoding.io/cloud-computing/que-es-kubectl/

NgRock. (2024, 20 febrero). What is Ngrok? | Ngrok Documentation. Recuperado 20 de febrero de 2024, de https://ngrok.com/docs/what-is-ngrok/

Fernandez, O. (2023, 29 octubre). Prometheus: Introducción a la Monitorización de Métricas. Aprender BIG DATA. Recuperado 20 de febrero de 2024, de https://aprenderbigdata.com/prometheus/

Martínez, J. (2023, 27 abril). Qué es Grafana y primeros pasos. OpenWebinars.net. Recuperado 20 de febrero de 2024, de https://openwebinars.net/blog/que-es-grafana-y-primeros-pasos/

Team, K. (2023, 9 marzo). ¿Qué es Docker Hub? | KeepCoding Bootcamps. KeepCoding Bootcamps. Recuperado 20 de febrero de 2024, de https://keepcoding.io/blog/que-es-docker-hub/

Coppola, M. (2023, 13 febrero). Qué es Java, para qué sirve, características e historia. Hubspot. Recuperado 20 de febrero de 2024, de https://blog.hubspot.es/website/que-es-java

Schwaber, K. S., & Sutherland, J. S. (2020, noviembre). *La Guía de Scrum.* Scrum Guides. Recuperado 23 de noviembre de 2023, de https://scrumguides.org/scrum-guide.html

BIBLIOGRAFIA

Hall, T. H. (2023, 15 noviembre). *Cultura de DevOps | Atlassian. Atlassian. Recuperado 15 de noviembre de 2023, de* https://www.atlassian.com/es/devops/what-is-devops/devops-culture

Diego. Cortez. (2023, 27 julio). *Introducción a CI/CD - Medium. Medium. Recuperado 23 de noviembre de 2023, de* https://medium.com/@diego.coder/introducci%C3%B3n-a-ci-cd-be79d0beafad

Salazar, E. Y. H. (2020). *SCRUM, un enfoque práctico de metodología ágil para la ingeniería de software.* https://revistas.udistrital.edu.co/index.php/tia/article/view/15702

JetBrains. (2023, 15 noviembre). *¿Qué es CI/CD en DevOps? . Recuperado 15 de noviembre de 2023, de* https://www.jetbrains.com/es-es/teamcity/ci-cd-guide/devops-ci-cd/

J. Guerrero, K. Zuniga, C. Certuche and C. Pardo, “A systematic mapping study about Devops”, Jou. Cie. Ing., vol. 12, no. 1, pp. 48-62, 2020. doi:10.46571/JCI.2020.1.5

Miguel, M. M. (2022, 23 enero). *Integración y entrega continua (CI/CD) con Jenkins. Recuperado 16 de noviembre de 2023, de* https://openaccess.uoc.edu/handle/10609/137990

ANEXOS

Anexo 1. Diagrama de Gantt