OPTIMIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE EN LA EMPRESA XIRIUS A PARTIR DE LA IMPLEMENTACIÓN DE BUENAS PRÁCTICAS BASADAS EN DEVOPS

Informe final
Modalidad: Práctica Profesional

Oscar Yovany Chauza Narvaez
104617011552

Asesor de la empresa: Ing. Julián Solarte
Director: PhD(C). Manuel Alejandro Pastrana Pardo
Co director: PhD. Hugo Armando Ordoñez Erazo

Universidad del Cauca
Facultad de Ingeniería Electrónica y de Telecomunicaciones
Departamento de Sistemas
Ingeniería de Sistemas
Popayán, septiembre de 2023

Contenido

LISTA DE FIGURAS

LISTA DE TABLAS

Capítulo 1- Introducción

1.1. Justificación y planteamiento del problema

La industria del software evoluciona constantemente. Por tal motivo, las organizaciones buscan nuevas formas de mejorar sus procesos de desarrollo de software con el ánimo de incrementar su competitividad y satisfacer las necesidades del mercado, potencializando la calidad de sus productos [1]. En ese sentido, la velocidad de desarrollo de un producto software desde la perspectiva de un lanzamiento más rápido al mercado y con la calidad adecuada son la clave del éxito en la industria del software como sugiere [2]. Debido a esto, las empresas están adoptando DevOps, que se puede definir de acuerdo con [3] como una colaboración de los equipos de desarrollo y operación, con el objetivo de elaborar proyectos de calidad a tiempo y dentro del presupuesto planeado. Según [4]**,** DevOps puede proporcionar varios beneficios, tales como el desarrollo y lanzamiento más rápido de nuevas funcionalidades del software para los usuarios finales, una mejora en la calidad del software, una mejor colaboración y comunicación del equipo, entre otros.

Aunque lo anterior menciona la importancia y algunos de los beneficios que puede traer el uso de DevOps en las empresas, la implementación y adopción de prácticas es una labor compleja, en parte porque se desconoce ¿qué pueden implementar? y ¿cómo hacerlo exitosamente? ya que no existe una guía especifica que les permita identificar las practicas adecuadas según la cultura organizacional y las características de cada organización, sin incurrir en un proceso costoso de exploración como menciona [5]. Finalmente, este proceso se convierte en un ensayo y error cuyos costos son asumidos por las empresas. Dado el tamaño de estas, podría llevar a que sea un proceso lento, poco asumible, costoso y desmotivante, especialmente si se trata de una empresa muy pequeña o Very Small Entity (VSE) que tienen un máximo de 25 empleados y recursos limitados de acuerdo con [6]. Sin embargo, con la orientación y el apoyo adecuados, estas empresas pueden implementar prácticas de DevOps de manera efectiva y beneficiarse de sus ventajas.

Por lo anterior, Xirius Tech empresa caracterizada como una VSE busca optimizar su proceso de desarrollo de software acelerando la velocidad de puesta en producción de sus proyectos, obteniendo así una retroalimentación mayor por parte de los clientes sin sacrificar la calidad de los entregables. Para ello, la empresa busca adoptar un conjunto de prácticas basadas en lo que propone DevOps como complemento a su forma de trabajo actual, generando transformación de la cultura de organizacional, incrementando la calidad de sus entregables y reduciendo tanto los tiempos de despliegue, como los posibles reprocesos posteriores a la entrega.

Actualmente la empresa tiene implementadas como prácticas de calidad preventiva el versionamiento y la integración continua (IC), ambas bajo un modelo propio de funcionamiento, mediante la herramienta GIT. Sin embargo, desean revisar y

optimizar el modelo de versionamiento como oportunidad de mejora de su proceso. También desean revisar el script de IC implementado actualmente por el mismo motivo. Finalmente, la empresa quiere adoptar las prácticas de Análisis Estático de Código (AEC) y Despliegue continuo (DC) buscando la automatización del ciclo de calidad preventiva hasta este punto. Las prácticas de AEC y DC son, descritas brevemente a continuación para una mejor comprensión:

- Análisis estático de código: El análisis estático de código consiste en inspeccionar el código de un programa sin ejecutarlo, de forma rápida y eficiente de acuerdo con [7]. Además ayuda a verificar las convenciones internacionales de código o estándares, importantes según [6] por varias razones entre las que destacan: 1) todos los miembros del equipo van a escribir código de la misma manera, generando orden y mejor comprensión del código por parte de todos los integrantes del equipo de desarrollo como sugiere [8], 2) esta práctica aumenta la mantenibilidad, modificabilidad, escalabilidad y propiedad colectiva del código de acuerdo con [8], 3) las convenciones de código ayudan a disminuir las malas prácticas de codificación y aprender de ellas para reducir la deuda técnica de los proyectos como sugiere [8].

- Despliegue continuo: El despliegue continuo es una práctica que tiene como objetivo la puesta de la unidad desplegable de código (entregable) en el ambiente deseado (de prueba o producción) de acuerdo con [7], reduciendo las acciones que se hacen de forma manual por parte del equipo de operaciones para realizar los despliegues. Es decir que su objetivo es automatizar todo el flujo de trabajo para simplificar la liberación rápida del software y de esta forma entregar valor al cliente de forma ágil, fiable y repetida a un menor costo como sugiere [9].

Este proyecto propone una forma de minimizar los costos de exploración, a partir de la implementación de prácticas ya probadas e implementadas en otras compañías con características similares esperando generar un impacto positivo en la organización Xirius Tech. De acuerdo con lo anterior, el desarrollo de la práctica profesional estará dado por la propuesta de un proceso complementado por las sugerencias de DevOps que permita integrar de manera automatizada las prácticas de versionamiento, integración continua (IC), análisis estático de código (AEC) y el despliegue continuo (DC) utilizando como punto de partida el versionador ya implementado por la empresa y las solicitudes tipo pull request que se hacen al mismo, todo esto con la ayuda de un script de automatización.

La herramienta que se utilizarán para el versionado y la IC (actualmente implementado) será GitHub, para el DC se realizara a través de un pipeline en GitHub Actions y el servidor final donde se desplegaran los proyectos será Google Cloud Plataform. Para análisis estático de código la herramienta a utilizar será SonarCloud, debido a que esta permite a los miembros del equipo identificar vulnerabilidades en función de las diez principales preocupaciones de seguridad de aplicaciones web de Open Web Application Security Project (OWASP) [7]. También se identifican posibles errores, vulnerabilidades y malas prácticas de desarrollo utilizadas (olores de código) [7]. Con Sonar es posible identificar la duplicidad de

código y la cobertura de las pruebas, incluso detectando si las pruebas unitarias se han realizado o no [7]. Adicionalmente garantiza que se mantenga un estándar de código entre todos los desarrolladores, es decir convenciones de código [6].

1.2. Objetivos

1.2.1. Objetivo general

Proponer un proceso de DevOps para la correcta implementación de buenas prácticas de integración continua, análisis estático de código y despliegue continuo en la empresa XIRIUS a través de un proceso de automatización.

1.2.2. Objetivos específicos

- Caracterizar la empresa XIRIUS mediante proceso de entrevista y observación para determinar el estado actual de buenas prácticas implementadas respecto a integración continua (IC), análisis estático de código (AEC) y despliegue continuo (DC) vs el estado deseado y su impacto.

- Diseñar un proceso de automatización, que permita a partir del versionamiento dar soporte las actividades de integración continua (IC), análisis estático de código (AEC) y despliegue continuo (DC), por medio de la práctica de pull request para unificar el desarrollo y la operación en un mismo proceso, integrado y único.

- Evaluar el proceso propuesto, con ayuda de la técnica GQM (Goal-Question-Metric) por medio de las métricas número de despliegues exitosos y número de no conformes que deben ser medidos antes y después de la implementación de las prácticas para medir impacto generado en la empresa.

1.3. Metodología de desarrollo

Para el correcto desarrollo de la práctica profesional, se trabaja sobre una metodología ágil basada en scrum, donde cada sprint equivale a una semana.

Con el fin de cumplir exitosamente con los objetivos planteados anteriormente se definen las siguientes 6 etapas:

- **Análisis**

En esta etapa se realiza la caracterización de la empresa, se comprende el contexto general de trabajo de la empresa Xirius Tech SAS, apartir de esto se identifica el estado actual de la empresa respecto a las prácticas de DevOps implementadas y las prácticas que faltan por implementar u oportunidades de mejora

.

- **Planeación**

Se establece una compresión clara del proyecto, se definen los objetivos y requisitos planificación inicial del desarrollo, esto se realiza con el evento Sprint Planning Meeting y se realiza una vez al inicio de cada sprint a partir del primer sprint de desarrollo. De esta etapa se genera el sprint backlog y el tablero Scrum.

- **Capacitación**

En esta etapa se realiza la capacitación en las nuevas herramientas y procesos DevOps. Se enfoca en obtener el conocimiento y las habilidades necesarias para utilizar de forma correcta estas tecnologías. Esto incluye la formación en el uso de herramientas específicas como GitHub, SonarCloud y Google Cloud entre otras, así como la comprensión de los principios fundamentales de DevOps y sus beneficios para la organización.

- **Diseño**

Se realiza el diseño del proceso propuesto para las diferentes prácticas, esto mediante modelos de proceso para cada practica como versionamiento, integración continua, análisis estático de código y despliegue continuo. El diseño se centra en garantizar una integración armoniosa de estas prácticas en el flujo de trabajo existente de la empresa, con el fin de mejorar la calidad y velocidad en la entrega del software.

- **Implementación**

En esta etapa se realiza la implementación del proceso propuesto, se definen los roles y entregables resultantes, se implementa el script que permite automatizar en la herramienta de GitHub las prácticas de versionamiento, integración y despliegue continuos, se realiza la integración la práctica de análisis estático de código en el script creado, utilizando la herramienta SonarCloud.

- **Evaluación de la propuesta**

Esta etapa se lleva a cabo mediante se lleva a cabo mediante el uso de la técnica GQM, se realiza la definición y documentación del programa de medición (objetivos, preguntas, métricas e hipótesis), se recopilación datos sobre número de despliegues exitosos y número de no conformes antes y después de la implementación del proceso propuesto, se realiza la interpretación datos recopilados para obtener respuestas a las preguntas definidas, se evalúa el cumplimiento de los objetivos planteados, se genera de un reporte de los resultados obtenidos en la evaluación de la propuesta y se realiza una retroalimentación con el equipo de desarrollo.

1.4. Organización del documento

A continuación, se describe de manera general el contenido y organización del presente informe final:

Capítulo 1- Introducción: Hace referencia al presente capitulo donde se describe el planteamiento del problema, los objetivos a cumplir y metodología de desarrollo.

Capítulo 2- Marco teórico: En este capítulo se describen los conceptos teóricos más relevantes que se emplearon para la realización del trabajo de grado.

Capítulo 3- Caracterización de la empresa: En este capítulo se realiza la caracterización de la empresa identificando el contexto general de la empresa y oportunidades de mejora frente a prácticas DevOps.

Capítulo 4- Programación de actividades: En este capítulo se detalla las actividades realizadas para el desarrollo de la practica junto con su tiempo de elaboración.

Capítulo 5- Elaboración e implementación de la propuesta: En este capítulo se describe las actividades realizadas para llevar a cabo el diseño del proceso de automatización.

Capítulo 6- Evaluación del proceso propuesto: En este capítulo se describe el proceso llevado a cabo para la evaluación de la propuesta.

Capítulo 7- Lecciones aprendidas: En este capítulo se describe las lecciones aprendidas generadas a partir del desarrollo de la práctica profesional.

Capítulo 8- Conclusiones: En este capítulo se describe las lecciones aprendidas generadas a partir del desarrollo de la práctica profesional.

Capítulo 9- Bibliografía: En el último capítulo del presente documento se listan las referencias bibliográficas de los artículos, libros y demás recursos utilizados para el desarrollo del trabajo de grado.

Capítulo 2- Marco teórico

El uso de prácticas de desarrollo de software y su automatización a través de herramientas ya no es un lujo para los equipos de desarrollo de software hoy en día, sino parte de su forma de trabajo como respuesta a las necesidades de la industria que presenta un acelerado ritmo de entregas y requieren de mayores garantías de alta calidad que disminuyan el reproceso luego de su puesta en producción de acuerdo con [10].

A continuación, se presenta la descripción de algunos conceptos y herramientas relacionadas con prácticas de desarrollo de software y automatización, que se utilizarán en esta propuesta y son importantes de resaltar para facilitar la compresión de este documento.

2.1. DevOps

DevOps podría definirse como el conjunto de principios y prácticas que permiten afrontar de manera diferente el lanzamiento de productos de software, aumentando su calidad a través de la colaboración activa entre las áreas de desarrollo y las operaciones de la empresa según [10]. Esta relación se crea con el fin de mejorar la integración y comunicación de estas dos áreas, con el fin de obtener los beneficios de los enfoques modernos en el desarrollo de software. Un ejemplo de esto es un conjunto de principios que se presentan a continuación para una mejor apreciación del concepto y que son propuestos por [11]:

- **Acción enfocada al cliente:** El requisito del cliente puede cambiar ligeramente durante la construcción del servicio o producto. Si no hay una buena comunicación, pueden surgir problemas al momento de introducir una nueva función o característica, que requiera el cliente en una etapa posterior.

- **Crear con el objetivo en mente:** Deben dejarse de lado procesos de rol donde cada individuo tiene una función, en lugar de un pensamiento donde se visualice y comprenda todo el contexto y que compone la creación del producto o servicio, del trabajo que se está realizando.

- **Responsabilidad de extremo a extremo:** Las áreas de desarrollo y operación deben ser totalmente responsables de sus actividades y asegurar que se realicen de la mejor manera, con los mejores resultados durante todo el ciclo de vida del producto.

- **Equipos autónomos y multifuncionales:** Es muy recomendable tener miembros del equipo con un perfil que tenga un conocimiento profundo en un tema específico, pero con cierta comprensión en otros temas.

- **Mejora continua:** La cultura DevOps promueve la mejora continua. La forma de adquirir este pensamiento es a través de la experiencia, por lo que se

debe adoptar un pensamiento abierto para aprender lo más posible de los fracasos.

- **Automatiza todo lo que puedas:** No se trata solo de automatizar la liberación continua de procesos, consolidación y despliegue, sino observar que otras funciones o características se pueden automatizar para obtener una ventaja significativa.

DevOps no solo depende de las estrategias de comunicación entre las áreas de desarrollo y operación. También depende de qué tipo de herramientas se utilizan y cómo se comportan los integrantes, al momento de utilizarlas, así como de la cultura organizacional y los procesos que se ven impactados por su implementación [10]. Esto representa un desafío debido a la variedad de herramientas disponibles en constante evolución. Los objetivos de las herramientas son facilitar el trabajo de las diferentes áreas, permitir la liberación continua y mantener la confiabilidad del software [10].

2.2. Versionamiento

El versionamiento es un punto de partida para el proceso de buenas prácticas de software, para su implementación primero se recomienda comprender la estructura correcta de un modelo de versionado, seguido de los pasos para su uso y las prácticas que se deben implementar en el mismo, esto se realiza con ayuda de los versionadores, los cuales son herramientas que funcionan como repositorios que centralizan los cambios y aseguran la disponibilidad para todos los miembros del equipo de desarrollo (propiedad colectiva del código) de acuerdo con [6]. Adicionalmente, todos los cambios almacenados por la herramienta se guardan en un historial para su trazabilidad, y si se genera un error en la integración de los cambios, se vuelve a la versión estable anterior sin mayores demoras (fallo de recuperación). El propósito de esta herramienta es sincronizar los cambios locales con la versión alojada en la herramienta. Todos los cambios se alojan en una separación estructural que crea la herramienta llamada rama y, por lo general, la rama inicial que crea cada versionador recibe el término maestra como indica [6].

2.3. Integración continua (IC)

La práctica de la integración continua (IC) va de la mano con el control de versiones, según [6]. Si bien el versionado ayuda a centralizar cambios, mantener el orden y seguir la evolución del sistema en desarrollo, no permite verificar el impacto de los cambios generados en la unidad desplegable. Por lo anterior y lo siguiente [6], es recomendable adoptar la práctica de integración continua que pueda validar esto como parte de la cultura organizacional.

Para llevar a cabo la práctica de integración continua, es necesario contar con una herramienta de implementación que sea capaz de generar la unidad desplegable a partir de instrucciones que se den desde la consola, considerando el lenguaje de

programación y el sistema operativo correspondiente. Una vez completado el proceso, se debe notificar al equipo que la integración continua se ha llevado a cabo con éxito. En caso de que se presente algún error, se notificará al último miembro del equipo que haya realizado cambios para que realice las correcciones necesarias como indica [6].

2.4. GitHub

GitHub es una plataforma de alojamiento de código para el control de versiones y la colaboración. Permite a las personas trabajar juntos en proyectos desde cualquier lugar [12]. Además, cuenta con GitHub Actions, la cual permite realizar el proceso de integración e implementación continua, más conocido por sus siglas en ingles CI/CD, es el proceso de constante de validación e implementación de código en el entorno DevOps [13].
La gran parte del proceso de CI/CD se automatiza mediante el uso de algo llamado canalizaciones [13]. Se pueden crear múltiples canalizaciones con diferentes etapas y trabajos dentro de esas etapas para un entorno específico, por ejemplo, una canalización para el entorno de desarrollo, una para el entorno de prueba y otra para el entorno de producción. Dado que es posible que el desarrollador no necesite tener las mismas herramientas en el entorno de desarrollo y producción, es posible personalizar las canalizaciones para diferentes entornos [13].

2.5. Despliegue Continuo (DC)

Cuando el código fuente está versionado y el CI genera el archivo de la unidad desplegable, que debe colocarse en un servidor de aplicaciones para que el software esté operativo, el CD es posible, según [16]. La práctica de CD aprovecha que el CI ha generado la unidad desplegable, para no repetir este proceso, y traslada ese archivo al servidor (físico o en la nube) donde está alojado el servidor de aplicaciones que lo desplegará. Una vez que el archivo se ha llevado al servidor, se coloca en la ubicación requerida, según el servidor de la aplicación. A continuación, se ejecutan los comandos específicos del sistema operativo necesarios para realizar la implementación, lo que en la mayoría de los casos hace que el servidor de aplicaciones, pero no el servidor físico, requiera reiniciar su servicio.

2.6. Google Cloud Plataform (GCP)

Google Cloud consiste en un conjunto de recursos físicos, como computadoras y unidades de disco duro, y recursos virtuales, como máquinas virtuales (VM), que se encuentran en los centros de datos de Google en todo el mundo. Cada centro de datos está ubicado en una región, específicamente en Asia, Australia, Europa, América del Norte y América del Sur de acuerdo con [14].

Esta distribución de los recursos brinda varios beneficios, que incluyen redundancia en caso de fallas y menor latencia, ya que los recursos se encuentran más cerca de los clientes. La distribución también presenta algunas reglas sobre cómo se pueden usar los recursos en conjunto según [14].

2.7. Análisis Estático de código (AEC)

El análisis estático de código es una revisión automatizada que se realiza sobre el código sin necesidad de ejecutarlo. Es una práctica importante, ya que ayuda a identificar errores conocidos, vulnerabilidades y malas prácticas de programación mediante la aplicación de reglas predefinidas. Sin embargo, es posible que esta técnica genere falsos positivos, por lo que es necesario validar las revisiones realizadas. En este sentido, una herramienta de análisis estático de código es un complemento valioso para la revisión de código, ya que ayuda a minimizar el tiempo que los desarrolladores deben invertir en esta actividad de acuerdo con [15].

2.8. Open Web Application Security Project (OWASP)

El Open Web Application Security Project (OWASP) es una fundación sin fines de lucro que trabaja para mejorar la seguridad del software según [16]. A través de proyectos de software de código abierto liderados por la comunidad, cientos de capítulos locales en todo el mundo, decenas de miles de miembros y conferencias educativas y de capacitación líderes, la Fundación OWASP es la fuente para que los desarrolladores y tecnólogos protejan la web [16].

2.9. GQM

Es un modelo para la definición y documentación de programas de medición en ingeniería de software. Este modelo se basa en tres componentes principales: objetivos, preguntas y métricas.

Objetivos: Los objetivos son declaraciones de alto nivel que describen lo que se espera lograr con el programa de medición. Estos objetivos deben ser específicos, medibles, alcanzables, relevantes y oportunos. Por ejemplo, un objetivo podría ser mejorar la calidad del software entregado a los clientes.

Preguntas: Las preguntas son declaraciones que se utilizan para entender cómo se puede alcanzar un objetivo. Estas preguntas deben ser específicas, claras y responder a un objetivo específico. Por ejemplo, una pregunta podría ser: ¿Cuál es el porcentaje de errores en el software entregado a los clientes?

Métricas: Las métricas son los indicadores que se utilizan para responder a las preguntas. Estas métricas deben ser específicas, cuantificables y capaces de medir el rendimiento de un proceso o producto. Por ejemplo, una métrica podría ser el número de errores por cada 1000 líneas de código.

Capítulo 3- Caracterización de la empresa

En este capítulo se realizará una caracterización detallada de la empresa, incluyendo su contexto general de trabajo y los procesos utilizados en el desarrollo de software. Se describirá la configuración de inicio para un proyecto, incluyendo las condiciones, herramientas y metodología utilizadas. Además, se especificará el estándar de nomenclatura utilizado para la gestión de proyectos y versiones, y se detallará la configuración del entorno de desarrollo, incluyendo las condiciones necesarias y las reglas técnicas de desarrollo a seguir. También se abordará el proceso de gestión de errores, así como el estado actual de la empresa en cuanto a la implementación de buenas prácticas DevOps. Finalmente, se identificarán las oportunidades de mejora y las buenas prácticas que se proponen para intentar optimizar el flujo de trabajo y la calidad del software entregado.

4.1. Contexto general del flujo de trabajo de la empresa

Para obtener el contexto general de trabajo de la empresa Xirius Tech, en primer lugar, se realizó una reunión con el director de operaciones y el equipo de desarrollo, con el fin de conocer el paso a paso del proceso para el desarrollo de un proyecto software, a partir de esto se realizó un modelo de procesos de negocio (BPM) y el desglose de las actividades realizadas con su respectiva descripción con el fin de comprender y tener claridad de cada una de estas.

A continuación, se muestra el proceso para el desarrollo de un proyecto software en la empresa.

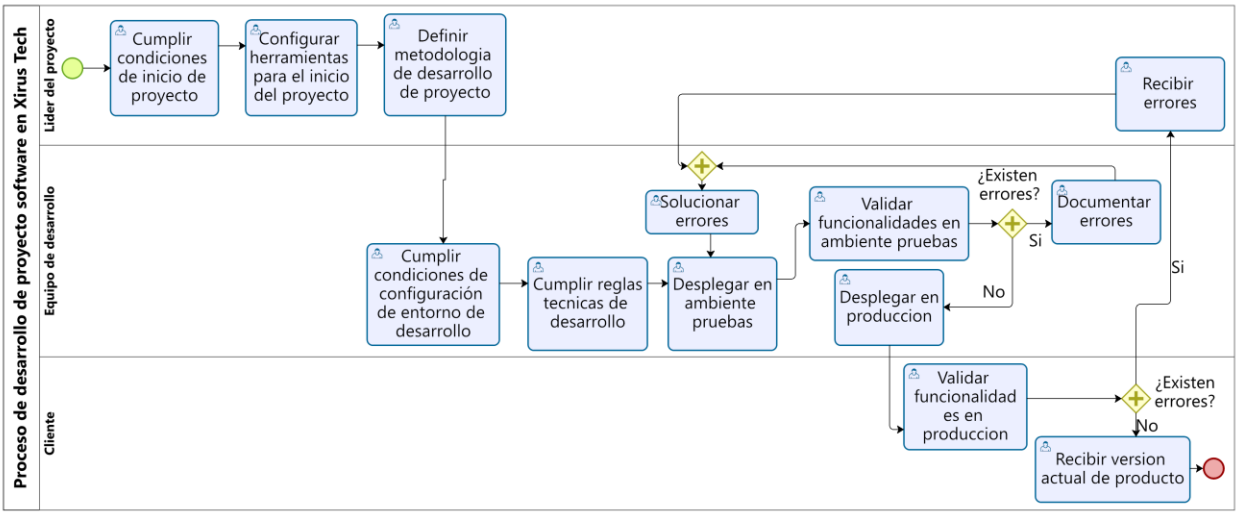

Figura 1. Desarrollo de un proyecto en Xirius Tech

4.2. Configuración y condiciones de inicio para un proyecto

A continuación, se muestra lo que se debe tener en cuenta para iniciar un proyecto en la empresa Xirius Tech.

- **Condiciones**

Para iniciar el proyecto en la fase de desarrollo, es necesario haber realizado las siguientes actividades:

- Haber especificado los requerimientos. Esta actividad se hace acorde con el formato de historias de usuario que utiliza la empresa.
- Haber analizado los requerimientos, que incluye:
  - Priorizar
  - Asignar
  - Aclarar
  - Verificar
  - Validar los requerimientos.
- Seleccionar el equipo de desarrollo responsable de recibir los requerimientos acordados con el cliente y llevar a cabo su implementación.

- **Herramientas**
  - **JIRA:** Para optimizar el inicio del proyecto se utiliza por parte de la empresa Jira Software, que es una herramienta en línea para la administración de tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos [17]. Con Jira se realizan las siguientes actividades:
    - **Planifica:** Crea historias de usuario e incidencias, planifica sprints y distribuye tareas entre el equipo de software.
    - **Supervisa:** Prioriza y analiza el trabajo del equipo en su contexto y con una completa visibilidad.
    - **Flujo de trabajo:** Cada equipo cuenta un proceso único para lanzar software. Utiliza un workflow predefinido o crea uno adaptado a la forma de trabajar del equipo.

  Además, Jira permite vincular los proyectos con los repositorios de GitHub de la empresa para facilitar el flujo de trabajo para el equipo de desarrollo. Adicionalmente, proporciona plugins que se adaptan a entornos de trabajo como visual code e Intellij IDEA que permite evidenciar las tareas que tienen asignadas cada miembro del equipo de desarrollo como sugiere [17].

  - **Google Drive:** Permite almacenar todos los artefactos y/o documentación que es necesaria para la ejecución del proyecto de acuerdo con [18].

- **Metodología de desarrollo en Xirius Tech**

Para el desarrollo de un proyecto software, la empresa usa una metodología basada en scrum, la cual cuenta con los siguientes roles y etapas:

1. **Roles**:
    - Scrum master
    - Product Owner
    - Development Team

2. **Etapas**:

    - **Análisis**: Durante esta etapa se establece una compresión clara de las necesidades del proyecto, definiendo los objetivos y requisitos mediante reuniones con los stakeholders, que son las personas interesadas en el desarrollo del proyecto. Además, durante las sesiones se identifica la problemática, se elicitan los requisitos, se definen objetivos y el alcance del proyecto, se definen los roles que tendrá el equipo Scrum, se crean las épicas y una lista priorizada de los requerimientos del proyecto (Backlog).
      De esta etapa se genera el Product Backlog, el cual se realiza utilizando el modelo de Cohn (Cohn, 2004) para la elaboración de las historias de usuario (HU) que lo componen.

      Además, en esta etapa se hace uso de la herramienta Jira la cual permite registrar el Backlog que se tiene actualmente, para esto las historias de usuario deben tener:
        - **Historias de usuario:** Las historias de usuario deben contener 6 aspectos:
            a. La descripción de la funcionalidad que se debe realizar
            b. Los criterios de aceptación de la historia de usuario
            c. La estrategia de implementación que usará la persona encargada de la historia de usuario
            d. La estrategia de pruebas que usará la persona encargada de la historia de usuario
            e. La estimación en horas de la tarea
            f. Las horas tomadas para realizar la tarea

    - **Planeación:** Se establece una compresión clara del proyecto, se definen los objetivos y requisitos; además de establecer el equipo y planificación inicial del desarrollo, esto se realiza con el evento Sprint Planning Meeting y se realiza una vez al inicio de cada sprint a partir del primer sprint de desarrollo. De esta etapa se genera el sprint backlog y el tablero Scrum.
    - **Diseño:** En esta etapa el equipo responde a las necesidades no funcionales del sistema, identificando los atributos de calidad a los que propende la solución y su debida respuesta a nivel de programación

mediante patrones de diseño. Lo anterior guía el proceso de toma de decisiones que permite garantizar una selección adecuada del lenguaje de programación y sus frameworks de desarrollo para obtener mejores resultados. Es un proceso continuo que se adapta y evoluciona a medida que el desarrollo del producto avanza.

- **Generación de código y despliegue:** Esta etapa consiste en traducir el diseño en una forma legible por la máquina, es decir el equipo de desarrollo escribe el código necesario para crear el producto software siguiendo todos los lineamientos definidos por la empresa y se va desplegando primero en ambiente de pruebas, posteriormente en ambiente productivo.
- **Pruebas:** Esta etapa consiste en hacer diferentes tipos de pruebas al producto software con el fin de garantizar su correcto funcionamiento, esto se hace continuamente cada vez sé que va agregando una nueva funcionalidad o el código ha sido actualizado.
- **Entrega producto:** Etapa donde se realiza la entrega del producto al cliente y se valida su correcto funcionamiento.

- **Estándar de nombre de las versiones y/o proyectos**

Una vez se tienen en cuenta las condiciones y la metodología de inicio, se crea el proyecto. El servicio de control de versiones para todos los proyectos es GitHub, todos los proyectos deben almacenarse en la cuenta de GitHub y cada proyecto debe ser nombrado en el siguiente orden:

- Nombre de la empresa para el cual se está desarrollando el producto. Ej: alpina.
- Nombre de la solución. Ej: conciliaciones
- Tipo de abstracción del software. Ej: (front. back, microservicio)

Por lo anterior un correcto uso sería: alpina-conciliaciones-back.

- **Configuración del entorno de desarrollo**

a) Condiciones

Para que cada desarrollador configure y gestione su entorno de trabajo, acorde a las directrices de la empresa, debe tener en cuenta los siguientes requisitos:

- Debe crear una cuenta en JIRA o proporcionar su correo electrónico para que el líder del proyecto lo asocie al proyecto en curso, para así poder hacer un seguimiento y una actualización de las tareas asignadas.

- Debe tener un usuario en GitHub para que el líder del proyecto asocie el repositorio a su usuario y así tenga acceso al código fuente del proyecto o función que le corresponde.
- Si trabaja en un entorno de trabajo como Intellij IDEA debe instalar el plugin de JIRA llamado “Jira Integration”, para que obtenga de manera directa las tareas que tiene asignadas y las diferentes características que puede actualizar de cada tarea desde el mismo entorno.

- Si trabaja en un entorno de trabajo como VisualCode debe instalar el plugin de JIRA llamado “Jira software and Bitbucket”, para que obtenga de manera directa las tareas que tiene asignadas y las diferentes características que puede actualizar de cada tarea desde el mismo entorno.

b) Reglas técnicas de desarrollo

En el proceso de desarrollo se deben seguir las siguientes reglas técnicas con el fin de garantizar la calidad y la eficiencia del proyecto.

- Seguir el estilo de nombrado definido por estándar para cada proyecto (ejemplo: camelCase para Java, snake_case para Python)
- No debe utilizarse ninguna librería o función obsoleta
- Tratar de utilizar las últimas funcionalidades de cada versión (ejemplo utilizar LocalDate en vez de Date en Java).
- Para cada historia de usuario asignadas al desarrollador debe de crearse una rama en el repositorio de GitHub del proyecto en curso, el nombre debe estar acorde a la historia de usuario que se asigna en Jira. Lo anterior con la finalidad de llevar un control de las funcionalidades del sistema para su integración.

- Los commits de acuerdo con [19], son una captura instantánea de los cambios que se tienen en ese momento en el proyecto software, estas instantáneas confirmadas pueden considerarse versiones seguras de un proyecto. Cuando se realicen estos commits sobre la funcionalidad descrita en la rama creada, deben de hacerse referenciando el id de la tarea, ejemplo:
  git commit -m “CDB-662 resto del commit”

- La rama máster en el repositorio estará bloqueada y solo se podrá hacer commits mediante pull requests, estos permiten revisar y debatir los cambios realizados por el desarrollador [20], cada pull request debe tener la aprobación de al menos un miembro del equipo y debe cumplir con los siguientes requisitos:

  i. Cumplir con los criterios de aceptación

  ii. Las clases, funciones y todo lo demás debe estar documentado

- La integración de todas las funcionalidades (merge), después de la aprobación del pull request solo debe realizarla o autorizarla el líder del proyecto.

Actualmente, en la empresa se realizan varias entregas a corto plazo para validar el funcionamiento del producto en desarrollo. Cada entrega sigue el estándar de nombrado de versiones tal cual como lo menciona GitHub.
Lo anterior con el fin de hacer un seguimiento de las nuevas funcionalidades, los commits y pull request que deben integrarse para el despliegue.

Adicionalmente, para llevar un mejor control del desarrollo de software que se está realizando, se verifica si cumple con los lineamientos de control de versiones semántico, con el fin de llevar el control adecuado de cada una de las entregas de acuerdo con la versión y a la dependencia con el producto en desarrollo.
Por último y para cumplir con los requerimientos de calidad del producto, los desarrolladores y el líder del proyecto realizan:

- Pruebas unitarias de la funcionalidad que implementaron
- Pruebas funcionales de todo el sistema en general

Finalmente se entrega la versión actualizada al cliente y/o usuarios involucrados, para terminar de evaluar los requerimientos funcionales del software y dar paso a la siguiente fase de desarrollo.

c) Reporte de bugs

Cuando se detecta un mal funcionamiento o fallo en el sistema (asociado comúnmente al termino bug) se creará un nuevo caso (ticket) en la herramienta designada como muestra la Figura 2, con el siguiente formato:

- Comportamiento esperado
- Comportamiento actual
- Soluciones posibles
- Pasos para reproducir el bug
    - Contexto
    - Ambiente
    - versión usada
    - Navegador/Celular nombre y versión.
    - Sistema operativo y su versión

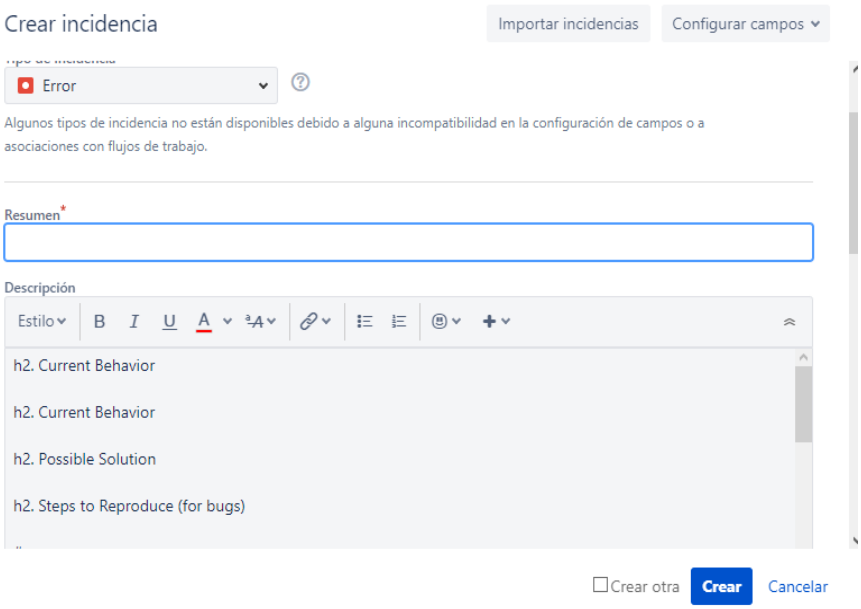

Figura 2. Plantilla para reportar un bug en Jira

4.3. Estado actual de la empresa respecto a prácticas DevOps

Para obtener el estado actual de la empresa Xirius Tech frente a algunas prácticas de DevOps como IC (Integración continua), AEC (análisis estático de código) y DC (despliegue continuo) se realiza un proceso de entrevista y observación, además se hace una encuesta a los desarrolladores, con el fin de identificar las prácticas que aplican en su día a día en los proyectos de desarrollo de software.

La elaboración de la preguntas se realiza teniendo en cuenta lo que menciona [21], sobre métricas para medir el estado de una empresa frente a prácticas DevOps, como frecuencia de despliegue. Así mismo se toma en consideración que Xirius es una VSE, por lo se tiene en cuenta la premisa que menciona [6] referente a que las entidades muy pequeñas (VSE) constituyen una gran parte de la industria y son las que más sufren, ya que sus procesos de desarrollo suelen ser empíricos y carecen de prácticas como control de versiones de código, integración continua (CI) y despliegue continuo (CD) para formular las preguntas respecto a estas prácticas.

A continuación, se muestran las preguntas realizadas y las respuestas obtenidas:

Tabla 1. Perfil profesional de los participantes

Id	**Ocupación**	**Experiencia** **Profesional**	**Estudios realizados**

PF1	Ingeniero de desarrollo	Dos (2) años	Pregrado
PF2	Ingeniero de desarrollo	Dos (2) años	Pregrado
PF3	Ingeniero de desarrollo	Menos de (1) año	Pregrado

Tabla 2. Preguntas y respuestas obtenidas

Pregunta	PF	Respuesta
¿Aproximadamente cuántos despliegues se realizan por semana en un proyecto en estado de desarrollo?	PF1	4
	PF2	5
	PF3	3
¿Aproximadamente cuántos casos de no conformidad hay por sprint en un proyecto en estado de desarrollo?	PF1	3
	PF2	2
	PF3	4
¿Conoce la diferencia entre despliegue continuo y entrega continua?	PF1	Si
	PF2	Si
	PF3	No
¿Conoce la diferencia entre versionar e integrar continuamente?	PF1	Si
	PF2	Si
	PF3	No
¿Se realiza push al finalizar la jornada de trabajo?	PF1	Si
	PF2	Si
	PF3	Algunas veces
¿En las inspecciones de código manual se tienen en cuenta el top 10 de recomendaciones de OWASP?	PF1	Algunas veces
	PF2	No
	PF3	Algunas veces
¿Se realiza revisión de uso adecuado de POO para evitar bloques duplicados?	PF1	Si
	PF2	Si
	PF3	Si
¿Se tiene definida una política de calidad de código para saber si está listo para entregar?	PF1	Si
	PF2	No
	PF3	No

Finalmente, se obtiene la siguiente tabla en la que se muestran las prácticas DevOps que aún no están implementadas u oportunidades de mejoras que se tiene en la empresa Xirius Tech. Esto se realiza teniendo en cuenta el modelo propuesto por [22],

donde los autores exponen como la norma ISO 19110 beneficia a la empresas VSE, se mencionan una serie de buenas prácticas que se asociadas a cada concepto de la norma, los cuales son, construcción y pruebas unitarias, verificación y validación, integración y pruebas, gestión del proyecto, arquitectura y diseño detallado, entrega del producto, análisis de requerimientos, control de versiones, y finalmente la autoevaluación

Tabla 3. Practicas DevOps que faltan por implementar

Nombre de la práctica	Descripción	Fuente
Versionamiento para base de datos	Esta práctica se refiere al uso de herramientas y metodologías que permiten la gestión de versiones de las bases de datos de una empresa. Esto ayuda a asegurar que los cambios realizados en la estructura de la base de datos sean registrados y gestionados de manera adecuada, permitiendo una fácil reversión en caso de errores o problemas.	[23]
Análisis de código estático	El análisis estático de código es una práctica que se utiliza para identificar errores y problemas en el código fuente de una aplicación. Esto se logra mediante el uso de herramientas y metodologías que permiten analizar el código sin necesidad de ejecutarlo, lo que ayuda a detectar errores antes de que se produzcan en tiempo de ejecución.	[24]
Despliegue continuo	El despliegue continuo es una práctica que consiste en automatizar el proceso de despliegue de las aplicaciones de una empresa. Esto ayuda a asegurar que las nuevas versiones de las aplicaciones sean desplegadas de manera rápida y eficiente, reduciendo el tiempo de inactividad y mejorando la experiencia del usuario.	[13]
Pruebas unitarias	Las pruebas unitarias son una práctica que consiste en escribir pruebas automatizadas para cada una de las unidades de código de una aplicación (funciones, métodos, etc.). Esto ayuda a garantizar que cada unidad de código funcione correctamente y de manera aislada, lo que a su vez reduce la posibilidad de errores en la aplicación en su conjunto.	Fuente propia
Estandarizado de prueba funcionales	Las pruebas funcionales estandarizadas en DevOps se refieren a la práctica de definir e implementar pruebas consistentes y repetibles para asegurar que el software	[25]

	funcione correctamente en diferentes entornos y configuraciones. Este tipo de pruebas suelen ser automatizadas e integradas en el pipeline de DevOps, lo que permite una retroalimentación continua y una identificación y resolución más rápida de problemas.	

4.4. Identificación de las prácticas que se implementaran u oportunidades de mejora

Teniendo en cuenta la información obtenida anteriormente, se analizaron las diferentes oportunidades de mejora o prácticas que se podrían implementar para este proyecto, se decide por alcance que se va a proponer de proceso DevOps que incluya las prácticas de integración continua (IC), análisis estático de código (AEC) y despliegue continuo (DC).

Con este proceso se espera mejorar la calidad del software y reducir los tiempos de entrega en el desarrollo de aplicaciones, debido a que la integración continua permitirá la detección temprana de errores y la resolución rápida de problemas, lo que a su vez mejorará la calidad del código y reducirá los costos asociados con la corrección de errores en fases posteriores del ciclo de vida del software.

Por otro lado, el análisis de código estático nos permitirá identificar posibles vulnerabilidades en el código antes de que se produzcan problemas de seguridad en la aplicación. Esto permitirá que los desarrolladores realicen correcciones tempranas en el código y reduzcan el riesgo de fallas en el sistema. Finalmente, el despliegue continuo nos permitirá reducir el tiempo necesario para lanzar nuevas versiones de la aplicación al mercado, lo que mejorará la satisfacción del cliente y la competitividad de la empresa.

Capítulo 4- Programación de actividades

En este capítulo se muestra la documentación referente a las actividades realizadas para identificar las necesidades de la empresa, y luego diseñar, implementar y evaluar el proceso propuesto. La activades se dividen en meses, cada actividad cuenta con una clasificación de prioridad teniendo en cuenta el orden o secuencia que se debe tener en cuenta para llevar a cabo este proyecto.

El desarrollo de las actividades de la práctica se hizo bajo la metodología actual de la empresa Xirius, la cual se basa en scrum y cada sprint equivale a una semana. Las actividades que no eran culminadas dentro del sprint se incluían en el siguiente sprint con una prioridad mayor, esto se realizó durante los 5 meses del desarrollo de la práctica con una intensidad horaria de 30 horas a la semana. A continuación, se detallan las actividades realizadas.

4.1. Primer mes de la práctica profesional

El primer mes de la práctica profesional tuvo como objetivo la caracterización de la empresa, identificación del contexto general, estado actual de la empresa y prácticas DevOps a implementar todo esto mediante un proceso de entrevista y con ayuda de una encuesta.

Tabla 4 - Primer mes de práctica: Actividades referentes a las reuniones en el mes.

Actividad	Prioridad	Duración prevista	Duración Real
Reunión de contexto general de empresa Xirius Tech	Alta	3	3
Reunión de identificación de buenas prácticas a implementar y planteamiento de la propuesta	Alta	3	3
Reunión de seguimiento semanal con el asesor	Alta	2	2
Reunión de seguimiento semanal con todo el equipo de Xirius	Alta	3	3

Durante todos los meses se programaron reuniones de seguimiento semanales tanto con el asesor de la empresa como con todo el equipo de desarrollo de Xirius, además se realizó una reunión de contexto general y otra de identificación de buenas prácticas a implementar y planteamiento de la propuesta esto con el equipo de desarrollo de Xirius y el director de trabajo de grado.

Tabla 5 - Primer mes de práctica: Actividades referentes a la caracterización de la empresa.

Actividad	Prioridad	Duración prevista	Duración Real
Documentación de contexto general del flujo de trabajo de Xirius Tech	Alta	30	30
Creación de encuesta para evaluar la empresa Xirius Tech frente a buenas prácticas DevOps.	Alta	15	15
Identificación de necesidades a partir del análisis de la entrevista, encuesta y documentación obtenida.	Alta	35	30
Análisis y priorización de las necesidades identificadas para la implementación de buenas prácticas DevOps en Xirius Tech.	Alta	20	15
Revisión y selección de herramientas y	Alta	20	20

tecnologías para la implementación de buenas prácticas DevOps en Xirius Tech.			
Planteamiento de propuesta de buenas prácticas a implementar en la empresa	Alta	16	16

Al finalizar el primer mes se tuvieron 6 actividades, con una duración total de 126 horas menos del tiempo previsto, ya que con la guía del director de trabajo de grado se logró analizar y priorizar las necesidades encontradas en la organización de forma más rápida.

4.2. Segundo mes de la práctica profesional

Tabla 6 - Segundo mes de práctica: Actividades referentes a las reuniones en el mes.

Actividad	Prioridad	Duración prevista	Duración Real
Reunión de explicación de SonarCloud	Alta	3	3
Reunión prueba de concepto	Alta	3	3
Reunión de seguimiento semanal con el asesor	Alta	2	2
Reunión de seguimiento semanal con todo el equipo de Xirius	Alta	3	3

Durante este mes se realizó algunas reuniones adicionales con el director de trabajo de grado para contextualización y revisión de la herramienta SonarCloud, así como también se realizó una prueba de concepto para comprender con mayor claridad la funcionalidad de la herramienta.

Tabla 7 - Segundo mes de práctica: Actividades referentes a capacitación y entendimiento de las tecnologías.

Actividad	Prioridad	Duración prevista	Duración Real
Revisión y análisis de herramienta Sonar Cloud	Alta	10	10
Taller práctico de herramienta Sonar Cloud	Alta	15	15
Taller práctico de contenerización con Docker	Alta	15	10
Taller de integración y despliegue continuo	Alta	30	30
Elección de reglas para Sonar y perfil de calidad	Alta	8	8
Revisión de pipeline para integración de análisis de código estático	Alta	25	20
Prueba con proyecto de ejemplo con integración continua, análisis de código estático y despliegue continuo.	Alta	30	35

Durante el segundo mes se realizaron 7 actividades, en las cuales se emplearon 128 horas acorde con la duración prevista, sin embargo, en algunas se utilizó un mayor

tiempo al previsto debido a la complejidad de la configuración de un despliegue en Google Cloud.

4.3. Tercer mes de la práctica profesional

Tabla 8 - Tercer mes de práctica: Actividades referentes a las reuniones en el mes.

Actividad	Prioridad	Duración prevista	Duración Real
Definición de roles y entregables de cada práctica	Alta	3	3
Reunión de seguimiento semanal con el asesor	Alta	2	2
Reunión de seguimiento semanal con todo el equipo de Xirius	Alta	3	3

Tabla 9 - Tercer mes de práctica: Diseño e implementación del script de automatización.

Actividad	Prioridad	Duración prevista	Duración Real
Análisis de requisitos para la automatización	Alta	12	12
Definición de modelo de versionamiento	Alta	10	10
Definición de modelo de SCA (Análisis de código estático)	Alta	10	10
Definición de modelo general	Alta	10	10
Configuración de ambientes para despliegue aplicación	Alta	30	30
Documentación de roles y entregables resultantes	Media	15	15
Implementación del script de integración continua en el entorno de certificación y producción	Alta	25	25
Revisión de código y optimización	Alta	15	15

Durante el tercer mes, se realizó un enfoque integral en el diseño y la implementación del script de automatización. El éxito de este periodo radicó en una buena planificación, lo que permitió cumplir con las horas asignadas. En total, se lograron acumular 127 horas de trabajo, evidenciando una gestión efectiva del tiempo y de los recursos disponibles para este proyecto.

4.4. Cuarto mes de la práctica profesional

Tabla 10 - Cuarto mes de práctica: Actividades referentes a las reuniones en el mes.

Actividad	Prioridad	Duración prevista	Duración Real
Retroalimentación y discusión con el equipo de desarrollo y asesor	Alta	5	5
Reunión de seguimiento semanal con	Alta	2	2

el asesor			
Reunión de seguimiento semanal con todo el equipo de Xirius	Alta	3	3

Tabla 11 - Cuarto mes de práctica: Seguimiento y monitoreo de proceso propuesto.

Actividad	Prioridad	Duración prevista	Duración Real
Revisión de código y optimización	Alta	20	22
Implementación de mejoras y optimizaciones	Alta	25	24
Evaluación de rendimiento y ajustes	Alta	15	18
Identificación y resolución de problemas potenciales	Alta	20	22
Monitoreo de integración continua con SonarCloud y Git Actions	Alta	15	16
Monitoreo y optimización de despliegue continuo en Google Cloud	Alta	15	14
Actualización y mejora del pipeline YAML	Alta	15	16

Durante este mes se hizo seguimiento y monitoreo al proceso propuesto, con esto se fueron haciendo ajustes para obtener un correcto funcionamiento y posteriormente hacer la evaluación de este proceso. En este mes se realizaron 7 actividades con un total de 132 horas y se tenía previsto 125 horas, esto debe a que no se tenía gran experiencia con las tecnologías sin embargo con la capacitación y ayuda del asesor se logró hacer los ajustes necesarios para obtener una buena implementación.

4.5. Quinto mes de la práctica profesional

Tabla 12 - Quinto mes de práctica: actividades referentes a las reuniones en el mes.

Actividad	Prioridad	Duración prevista	Duración Real
Reunión para contextualización de encuestas	Alta	3	3
Reunión de informe con resultados obtenidos	Alta	3	3
Reunión de seguimiento semanal con el asesor	Alta	2	2
Reunión de seguimiento semanal con todo el equipo de Xirius	Alta	3	3
Retroalimentación con el equipo de desarrollo y los stakeholders	Alta	3	3

Tabla 13 - Quinto mes de práctica: actividades referentes a evaluación del proceso propuesto.

Actividad	Prioridad	Duración prevista	Duración Real
Definición de objetivos para GQM	Alta	15	15
Definición de preguntas	Alta	20	20
Selección de métricas	Alta	15	15
Recopilación de datos	Alta	20	20
Interpretación de datos	Alta	25	22
Análisis y comparación de resultados con los objetivos definidos	Alta	15	15
Documentación de resultados finales y lecciones aprendidas	Alta	15	20

El último mes se enfocó en la evaluación del trabajo propuesto, se tenían previstas 125 horas y se utilizaron 127 horas, esto debido a que la generación del reporte con lleva un análisis y una retrospección para las lecciones aprendidas, lo que con llevo mayor tiempo del planeado.

Capítulo 5- Elaboración e implementación de la propuesta

5.1. Capacitación y entendimiento de tecnologías

5.1.1. SonarCloud

La herramienta Sonar ha sido diseñada con el objetivo principal de proporcionar una interfaz clara y fácil de usar para los desarrolladores. Con esta interfaz, los desarrolladores pueden acceder fácilmente a información detallada sobre posibles problemas en su base de código, desde la fuente raíz hasta la ubicación específica donde se produce el problema. Esta característica es particularmente útil para los desarrolladores que trabajan en proyectos grandes y complejos, ya que les permite identificar rápidamente y resolver problemas en el código de manera eficiente como sugiere [26]. SonarCloud tiene una guía clara para la resolución de problemas, está basada en 3 niveles:

- **Visualización de problemas:** Se muestra al desarrollador el problema resaltado en el contexto del código.
- **Navegación de problemas:** Permite navegar a través de las diferentes ubicaciones o fragmentos de código que contribuyen al problema.
- **Descripción de la regla:** Ayuda a comprender el problema y proporciona ejemplos de código que muestran la solución con el fin de que se haga la corrección fácilmente.

Al tener una comprensión clara del problema y una descripción completa de la regla que se está incumpliendo, la solución de problemas en el código se vuelve más fácil, esto también ayuda a que el desarrollador tenga en cuenta estos problemas y mejore sus habilidades en programación.

SonarCloud mejora el código, debido a que ayuda a mantener el código limpio y cumplir con altos estándares de codificación. Por otro lado, aporta más valor al usuario evitando problemas en el producto software debido a que se minimizan los errores que tanto odian los usuarios [26].

A continuación, se muestran los tipos de métricas que analiza SonarCloud:

- **Bug:** Son errores en el código que pueden impedir que el programa funcione según lo previsto. Afectan la confiabilidad del código.

- **Vulnerabilidades:** Son problemas en el código que podrían ser aprovechados por un mal actor para comprometer la seguridad de la aplicación.

- **Código oloroso:** Son características del código que, aunque no impiden el correcto funcionamiento del programa puede causar deuda técnica, es decir pueden afectar negativamente la mantenibilidad del código.

- **Security Hot pots:** Los puntos críticos de seguridad son áreas del código que pueden causar problemas de seguridad y, por lo tanto, deben revisarse. Por diseño, SonarCloud es más permisivo al identificar puntos críticos de seguridad que al identificar vulnerabilidades y otros problemas. Un problema casi siempre es un problema real, mientras que un punto de acceso de seguridad a menudo puede ser una falsa alarma (pero vale la pena verificarlo). Al separar los puntos de acceso de los problemas, SonarCloud mantiene la precisión de su detección de problemas al mismo tiempo que proporciona a los desarrolladores advertencias útiles según los criterios menos estrictos del punto de acceso [27].

- **Duplicación de código:**
  - **Bloques duplicados:** Número de bloques duplicados.
  - **Archivos duplicados:** Número de archivos involucrados en duplicaciones.
  - **Líneas duplicadas:** Número de líneas involucradas en duplicaciones.
  - **Porcentajes de líneas duplicadas:** Indica la proporción de líneas repetidas respecto al total de líneas, expresada como un porcentaje.
- **Cobertura para pruebas:** Verificar la cobertura de prueba de su código con herramientas de informes de cobertura es una parte esencial del proceso de desarrollo según [27].

SonarCloud permite configurar la importación automática de informes de cobertura de prueba producidos por sus herramientas específicas del lenguaje de programación e integrarlos a los resultados del análisis de SonarCloud. Al configurar esta integración, aporta una métrica adicional importante a tener en cuenta al establecer los estándares de calidad de un proyecto de acuerdo con [27].

5.2. Definición de roles y entregables resultantes

5.2.1. Roles involucrados en el proceso

Para la definición de los siguientes roles, teniendo en cuenta que algunos se encuentran presentes en el proceso de desarrollo de software tradicional, pero con alguna modificación para DevOps, de igual manera hay roles de Scrum que han sido parte de soluciones que buscan lograr exitosamente la adopción de DevOps en VSE [28].
Para asumir los siguientes roles en el contexto de DevOps es necesario que las personas involucradas tengan conocimientos acerca de DevOps [28].

Tabla 14. Roles involucrados en el proceso propuesto

Rol	Definición
Arquitecto software	De acuerdo con [29], se encarga de definir la arquitectura de los sistemas tomando las decisiones de diseño de alto nivel y estableciendo los estándares técnicos, incluyendo plataformas, herramientas y estándares de programación, teniendo en cuenta los requisitos funcionales, no funcionales y las necesidades del negocio.
Coordinador de liberaciones	Según [28], autoriza y realiza los despliegues al entorno de producción.
Desarrollador de software	De acuerdo con [30], es el encargado de convertir en componentes o subconjuntos de software (clases, módulos, pantallas, rutinas, subsistemas, programas en general) las especificaciones (funcionales y técnicas) para ser integrados en aplicaciones.
Facilitador	Según [28], es la persona encargada de liderar al equipo de trabajo en la gestión ágil del proyecto para que alcance sus objetivos hasta llegar a la etapa final de la iteración. Es el moderador de las reuniones y establece soluciones en conjunto para eliminar los obstáculos que dificultan el desarrollo ágil del equipo.
Ingeniero de pruebas	De acuerdo con [31], es el encargado de ejecutar pruebas en diversos componentes y funciones para detectar problemas técnicos y permite garantizar que los clientes obtengan productos funcionales de alta calidad.

Ingeniero DevOps	De acuerdo con [28], guía a la empresa en la adopción de DevOps mediante prácticas y herramientas tecnológicas que soportan la integración, entrega, despliegue y monitoreo continuo. Permite cerrar la brecha y mejora la coordinación y colaboración entre los roles que antes estaban aislados (desarrollo y operaciones de TI).
Líder de desarrollo	Según [28], es el encargado de liderar los procesos del equipo de desarrollo de software. En algunas empresas de desarrollo de software puede denominarse Líder Técnico, haciendo referencia al desarrollador de software más experimentado del equipo.
Usuario final	De acuerdo con [32], es el software o la persona que utiliza de manera directa la aplicación software del cliente.

5.2.2. Entregables resultantes

Tabla 15. Entregables resultantes

Entregable	Definición	Referencia
Archivo de configuración de la imagen del contenedor	Archivo o documento que contiene los comandos para ensamblar la imagen de un contenedor. Una imagen es una plantilla que se utiliza para construir un contenedor y ejecutarlo. Esta imagen incluye los binarios y librerías necesarias para que la aplicación se ejecute satisfactoriamente en un entorno.	[33]
Archivo de configuración de la infraestructura	Archivo o documento que describe en lenguaje a un alto nivel la instalación y configuración de la infraestructura requerida para la tubería CI/CD y los entornos de la organización (desarrollo, pruebas, preproducción, producción). Indica proveedores de infraestructura (Amazon, Google, etc.), conexión hacia ellos y recursos solicitados (servidores, red, almacenamiento, etc.) Este archivo se almacena en un repositorio. Además, si se requiere instalar una o más herramientas de la tubería CI/CD en este archivo se deberá establecer, por ejemplo: cuál herramienta se	[28]

	requiere instalar, qué versión, si requiere o no alguna librería para ejecutarla correctamente, entre otros.	
Archivo que empaqueta la aplicación software	Archivo que permite recopilar en un solo fichero varios ficheros diferentes, almacenándolos en un formato comprimido para que ocupen menos espacio. Es posible ejecutar este archivo sin necesidad de descomprimirlo. Este archivo contiene todos los ficheros que forman la aplicación software compilada.	[28]
Documento de los requisitos de infraestructura	Documento que especifica los requisitos de infraestructura que soportarán a la tubería CI/CD y además los despliegues de las funcionalidades, por ejemplo: el número de servidores, capacidad de CPU, memoria, sistema operativo, etc.	[28]
Imagen construida del contenedor	Es la imagen construida que será utilizada para crear un contenedor. Este artefacto se almacena en un repositorio de imágenes construidas, por ejemplo: Nexus, Docker Hub, entre otros.	Fuente propia
Lista de las herramientas CI/CD que se deben instalar y configurar	Documento que especifica la lista de herramientas tecnológicas a instalar y configurar para el despliegue de la tubería CI/CD	[28]
Lista de los proveedores de la infraestructura	Documento que especifica la lista de los proveedores de servicios de computación que proveerán los recursos para la implementación y el despliegue de los requisitos y la instalación, configuración y ejecución de la tubería CI/CD.	[28]
Lista de métricas para medir la calidad del código con SonarCloud	Documento que contiene los datos cuantitativos y cualitativos de cada métrica de calidad que se va a evaluar durante el análisis del código.	Fuente propia

5.3. Diseño del Script de automatización

5.3.1. Modelo de versionamiento

El siguiente es el modelo de versionamiento que se tiene actualmente en la empresa Xirius Tech para la inserción de cambios o nuevas funcionalidades

al proyecto software, se cuenta con un control de versión donde se controla la versión del código fuente utilizando Git. Git es usado para el almacenamiento de código fuente y el seguimiento de cambios en el mismo [34].
.

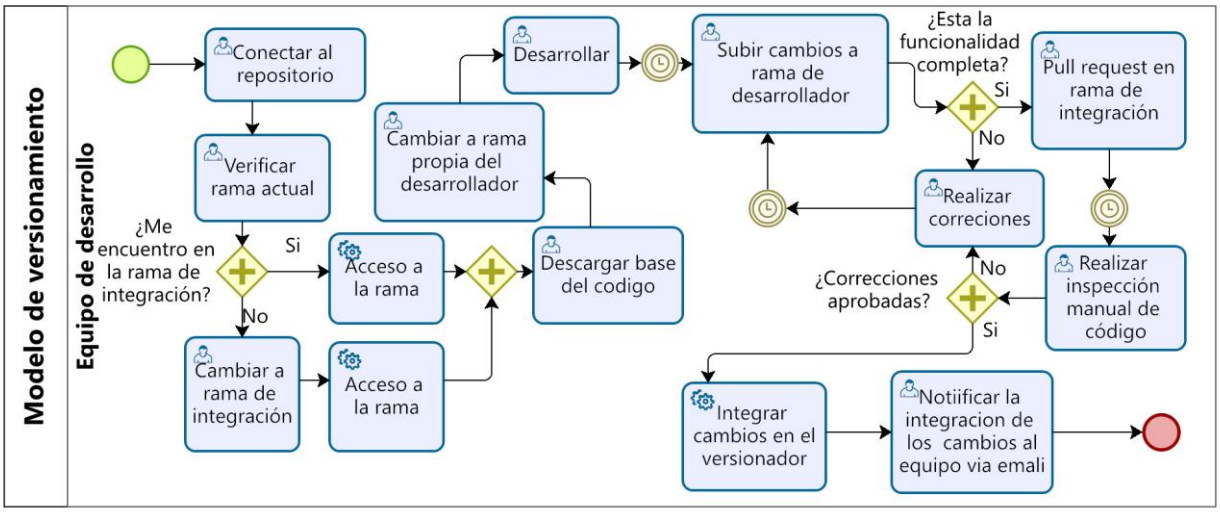

Figura 3. Modelo de versionamiento tomado de [35]

5.3.2. Modelo de proceso de análisis de código estático

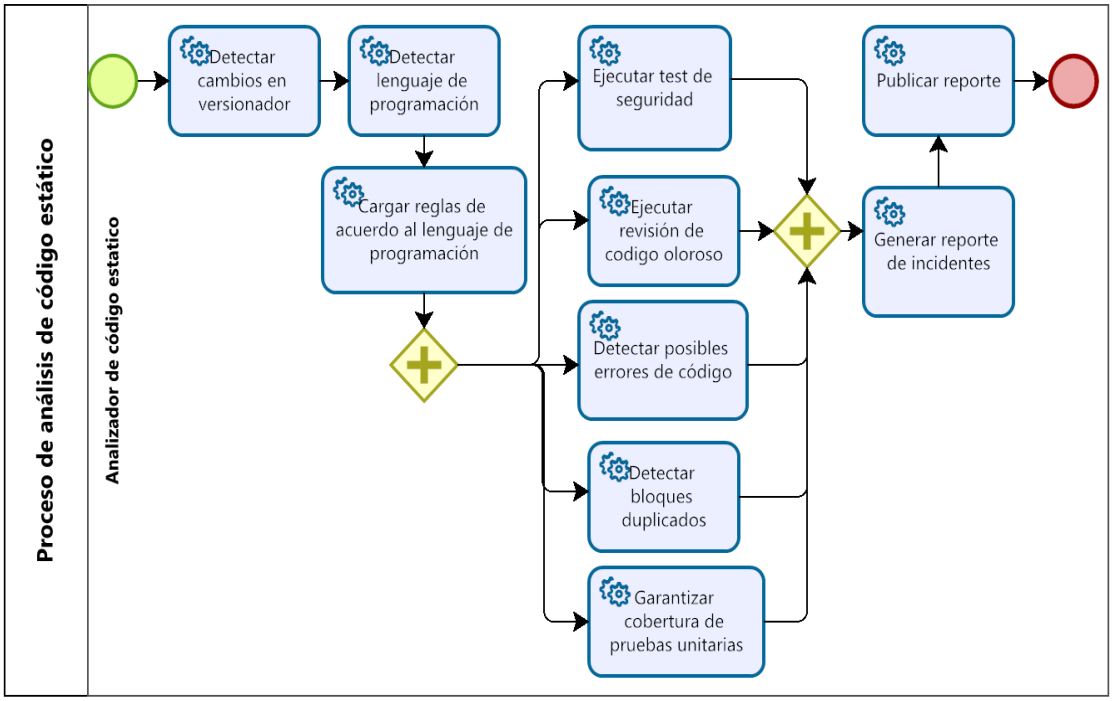

Figura 4. Modelo para proceso de análisis de código estático tomado de [35]

En la imagen anterior se puede evidenciar el proceso de análisis de código estático en este caso de Sonar Cloud, el proceso inicia a partir de la detección de modificaciones o nuevos cambios en el versionador, es decir GitHub y finaliza con la publicación de un reporte con los hallazgos encontrados, con esto aplicar las respectivas correcciones a nivel de código.

5.3.3. Modelo completo del proceso propuesto

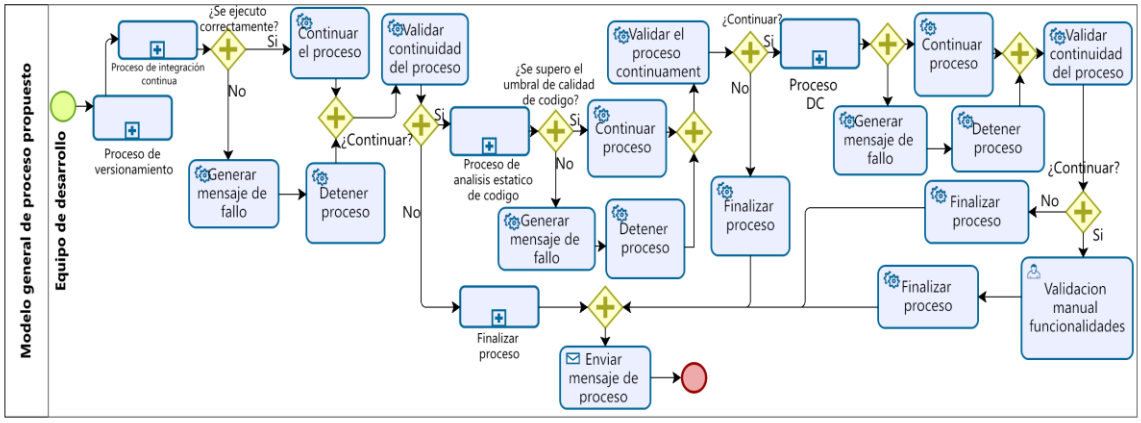

Figura 5. Modelo completo de proceso propuesto tomado y adaptado de [35]

En la anterior imagen se puede evidenciar, el proceso completo que se va a implementar, desde el proceso de versionamiento que ya se tenía de manera implementado correctamente, incluyendo el proceso de integración continua (IC),

proceso de análisis de código estático (AEC) y finalmente despliegue continuo (DC).

5.3.4. Configuración de ambientes para despliegue aplicación

Creación de ambientes

Para este proyecto se realizó el despliegue mediante Cloud Run, no obstante, también se investigó y documento el despliegue con Compute Engine. A continuación, se muestra una guía específica para los dos tipos de despliegue.

Creación del entorno de ejecución mediante Cloud Run

Cloud run permite ejecutar contenedores y la misma plataforma se encarga de administrar la cantidad de máquinas, servidores de balanceo entre otros recursos de acuerdo con [36].

Para desplegar una aplicación mediante Cloud Run se debe:

- Crear la imagen del contenedor.
- Subir la imagen al repositorio de Google Cloud.
- Crear el servicio y seleccionar la imagen previamente subida

Para desplegar un proyecto Maven en Cloud Run se debe seguir los siguientes pasos:

- Primero se debe construir el proyecto ingresando el siguiente comando en una terminal con la misma ubicación del proyecto (los IDE ya tienen la opción para generar una terminal): **mvn -B package -DskipTests -P [profile] --file pom.xml, [profile**] es el perfil que contiene los valores de las variables del proyecto (prod, test, dev).

- Segundo, construir la imagen de Docker mediante el comando: **docker build -t [IMAGE-NAME]**

- Tercero, agregarle una etiqueta a la imagen: **docker tag [IMAGE-NAME] gcr.io/[PROJECT-ID]/[IMAGE-NAME], [IMAGE-NAME]** es el nombre de la imagen previamente creada y **[PROJECT-ID]** es el id del proyecto que aparece en el apartado de información del proyecto en la interfaz principal de Google Cloud.

- Cuarto, subir la imagen al repositorio de imágenes de Google Cloud. **docker push gcr.io/[PROJECT-ID]/[IMAGE-NAME]**

Si se presentan errores de permisos al subir la imagen, se debe asegurar de tener instalado Google SDK y autenticarse con la misma cuenta de Google Cloud mediante su interfaz de línea de comandos, luego de ello se

desplegarán los proyectos asociados a la cuenta con la cual se hizo la autenticación, lo siguiente es seleccionar el proyecto sobre el cual se está trabajando e ingresar el comando gcloud auth configure-docker, abrir otra terminal en el proyecto y volver a ingresar el comando del cuarto paso.

Para hacer uso de la plataforma Cloud Run se debe habilitar el servicio Cloud Run y crear un servicio, esto despliega un formulario para personalizar el servicio.

- La URL de la imagen del contenedor debe aparecer al momento de dar clic en seleccionar.
- La región y zona se elige teniendo en cuenta de donde son los usuarios ej., si los usuarios son de Colombia se elige una región donde el ping sea bajo desde Colombia, para Bayport us-east4.
- Asignar CPU al momento de que le llegue una solicitud al servicio.
- Para que la API pueda ser consumida se debe permitir las invocaciones sin autenticar.
- Lo siguiente es la configuración de variables de entorno (4) y conexión a la base de datos, que si ya está creada aparecerá en lista de (5) y finalmente crear.

Creación del entorno de ejecución mediante Compute Engine

Al usar Compute Engine se crea una máquina virtual (VM) la cual se debe configurar manualmente para desplegar la aplicación, por ende, es un proceso más extenso que al usar Cloud Run, sin embargo, se tiene más control sobre las instalaciones.
En líneas generales para desplegar una aplicación de esta forma se debe:

- Crear la VM
- Instalar los requisitos como JDK, Maven, etc.
- Crear un servicio para arrancar y dejar en ejecución la aplicación.
- Construir el proyecto (.jar)
- Subir a la VM el proyecto construido (.jar)
- Iniciar el servicio previamente creado

Para crear la VM se debe habilitar el servicio de Compute Engine, posteriormente crear una instancia, luego llenar el formulario con la configuración de la VM deseada, como se muestra a continuación.

- La región y zona se selecciona teniendo en cuenta de donde son los usuarios, si los usuarios son de Colombia se elige una región donde el ping sea bajo desde Colombia.

- Dependiendo de los requerimientos de la aplicación software se escoge la máquina virtual, si se necesita bastante CPU, RAM o GPU, existen opciones optimizadas para este tipo de recursos, por otro lado,

para proyectos que no exigen alta demanda de recursos con una máquina de uso general mediana es suficiente.

- En el tipo de arranque se selecciona el sistema operativo y la versión del sistema, por lo general una distribución de Linux.

- El firewall se configura para permitir el tráfico hacia la VM.

Después de crear la VM se accede mediante el protocolo SSH (consola) y en el caso de Linux haciendo uso de la herramienta de paquetes se instalan los requisitos como el JDK, por ejemplo, en Debian se podría hacer mediante el comando:

- **sudo apt install default-jre**

Se instalará el JDK por defecto para Debian que desde la versión 10 en adelante es el JDK 11. Para construir el proyecto se hace uso el comando.

- **mvn -B package -DskipTests -P [profile] --file pom.xml** [profile] es el perfil o ambiente donde se desplegará la aplicación, contiene los valores de las variables del proyecto (prod, test, dev) esto genera un .jar en la carpeta target del proyecto.

Para crear el servicio se debe acceder a la ruta **/etc/systemd/system** mediante la consola de comandos y crear un archivo con el comando **touch service** cuyo contenido debe ser el siguiente:

[Unit]
Description=finesa

After=syslog.target

[Service]

User=root

ExecStart=/home/oscarchauza/tecnologia/source/finesa-test.jar

SuccessExitStatus=143

Environment="BD_USER=root"

Environment="BD_PASS=admin"

Environment="BD_URL=jdbc:
mysql://34.151.210.81/finesa?useSSL=false&createDatabaseIfNotExist=true"

Environment="GOOGLE_APPLICATION_CREDENTIALS=/home/oscar_chauza/tecnologia/finesa-credentials.json"

After: Destino donde se van a guardar los logs
User: Usuario que va a ejecutar el servicio
ExecStart: Es la ruta donde se encuentra el ejecutable
Enviroment: Guarda las variables de entorno, las mismas del perfil.xml además de las credenciales de las cuentas de servicio como se ve en la última línea.

Mediante el comando **sudo [nombre del servicio] start** se arranca el servicio y en el directorio **/var/log** se encuentran los archivos logs para verificar que todo haya sido exitoso, en el archivo daemon.log se puede acceder a ellos mediante el comando tail.

Creación de la Base de Datos

Para la creación de la base de datos lo primero que se debe hacer es habilitar el servicio Cloud SQL, una vez habilitado el servicio se debe crear una instancia y seleccionar el motor de base de datos (MySQL, PostgreSQL, SQL Server), luego de ello se mostrará un formulario con la información necesaria para la creación de la instancia como se muestra a continuación.

- Nombre de la instancia y la contraseña para el usuario root.
- Versión de la base de datos
- La región de la instancia de base de datos debe ser la misma región donde está ubicado el backend, esto con el fin de disminuir la latencia.
- Si la base de datos no va a ser usada con una alta exigencia de inserciones y consultas, una maquina con núcleo compartido o ligera es más que suficiente.
- Hay que recordar que la capacidad solo puede aumentarse mas no disminuirse una vez creada la instancia.
- Cuando la base de datos se va a usar para producción se debe de habilitar la IP privada.
- Por último, ya se puede crear la instancia.

Creación de Buckets

Para la creación de buckets se debe activar el servicio de Cloud Storage y se deben usar las siguientes opciones:

- Tipo de Ubicación: Multi Región
- Ubicación: us (varias regiones de estados unidos)
- Clase: estándar
- Control de acceso: uniforme

Creación de Cloud Functions

Para la creación de Cloud Functions es importante seleccionar la región donde está ubicado el backend para disminuir la latencia y como entorno la primera generación.

Dependiendo de la necesidad, una función se puede activar con base a un evento como por ejemplo la subida de un archivo a un bucket o a través de HTTP o PUB/SUB. Si las funciones se van a activar mediante HTTP como en el caso de bayport hay que permitir las invocaciones sin autenticar.

La memoria y el tiempo de espera dependerán de los procesos que realizará la función por ejemplo para ejecutar Puppeteer con node.js la memoria varia desde 512 MB hasta 1GB y un tiempo máximo de 8 minutos.

5.3.5. Integración del Análisis Estático de Código (AEC)

- **Creación de proyecto en Sonar Cloud y conexión con GitHub**

Anteriormente se creó la organización y repositorio en GitHub con los cuales se va a realizar la conexión para el AEC con Sonar Cloud. A continuación, se muestran los pasos para configurar dicha conexión.

1. Debe iniciar sesión con su cuenta de GitHub con en el siguiente enlace Iniciar Sesión Sonar Cloud
2. Luego, se debe dar clic en “Mis proyectos”, después en “analizar un nuevo proyecto”

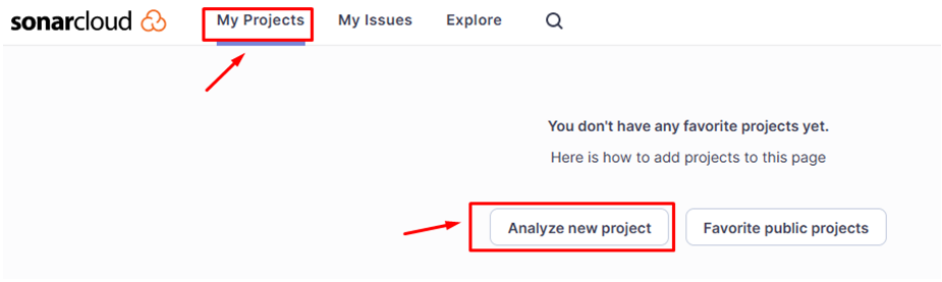

Figura 6. Sonar Cloud analizar nuevo proyecto

3. El siguiente paso es dar clic “importar una organización desde GitHub”, luego se debe seleccionar la organización y finalmente otorgar los permisos para acceder al repositorio. Se puede elegir dar acceso de lectura a todos los repositorios o solo a uno en particular, para este caso se dará acceso a todos los repositorios, ya que se necesita analizar el código de todos, por último, se debe dar clic en “instalar”.

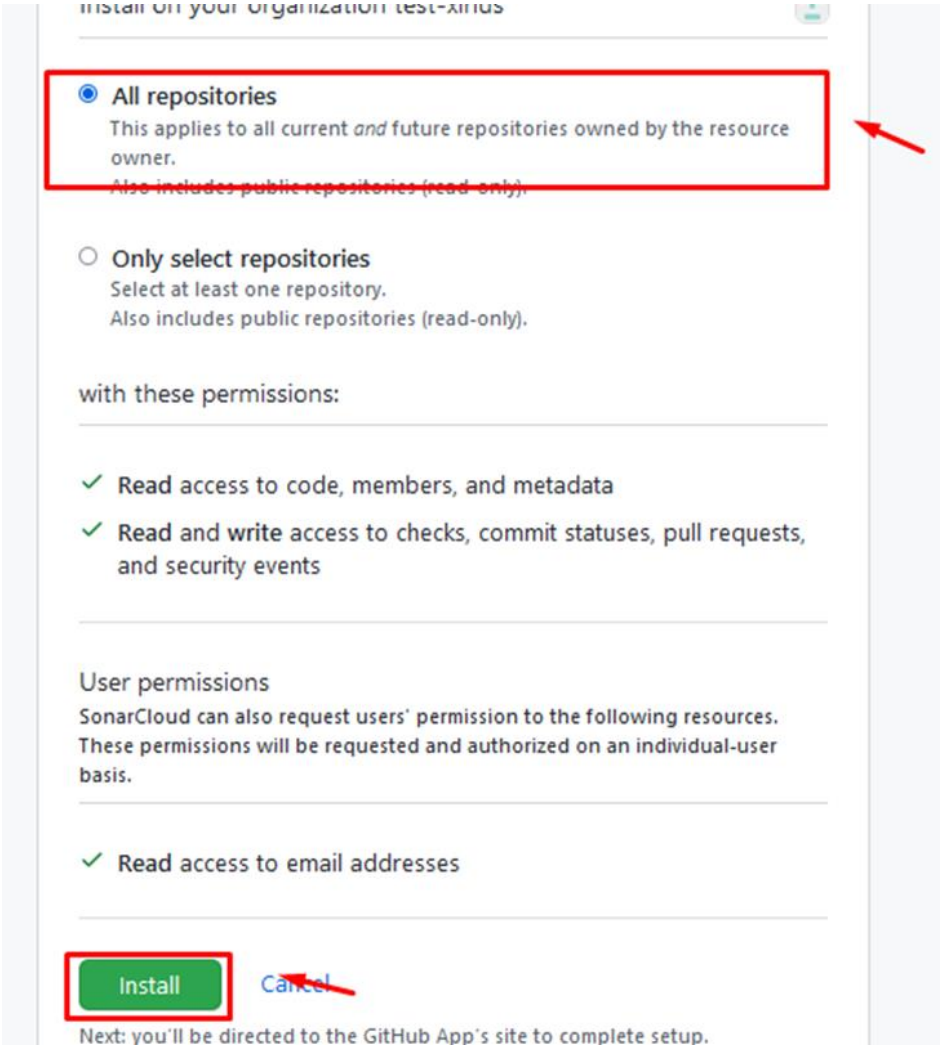

Figura 7. Permisos repositorios

4. Ahora se debe configurar el nombre de la organización para Sonar Cloud, en este caso se va a dejar el mismo que coloca la plataforma por defecto, el cual es el nombre que se obtiene de GitHub, luego damos clic en “continuar”.

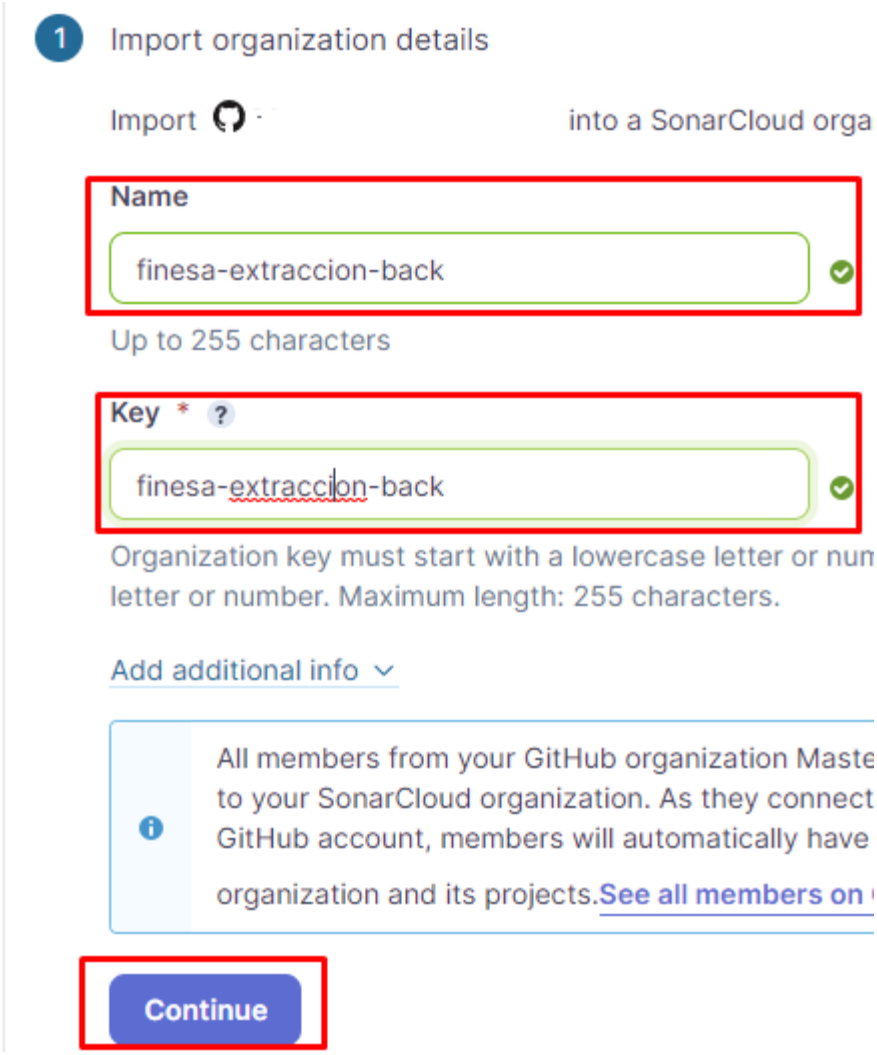

Figura 8. Nombre Organización en Sonar Cloud

5. El siguiente paso es configurar el pago y plan que se desea escoger, los planes vienen por cantidad de líneas de código que se analizarán mensualmente. Una vez configurado el pago y plan se debe dar clic en “Crear Organización”. Una vez creada la organización, seleccionamos los repositorios que se van a analizar. Para este es caso “bayport-verification” y se da clic en “configurar”.

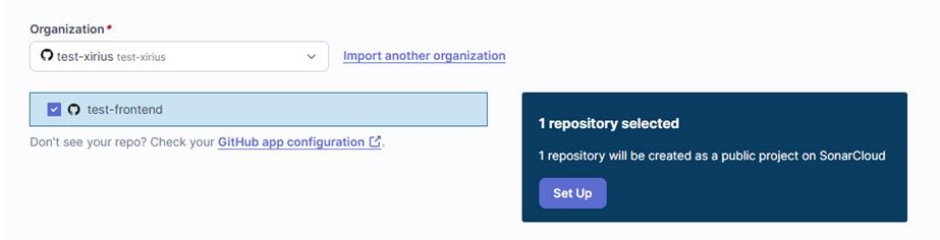

Figura 9. Selección de proyecto para analizar

6. Ahora, se debe configurar en el repositorio de GitHub los siguientes parámetros. Primero se debe entrar a “administración”, luego a “método de análisis”, ahí se debe elegir “analizar proyecto con GitHub Action”.

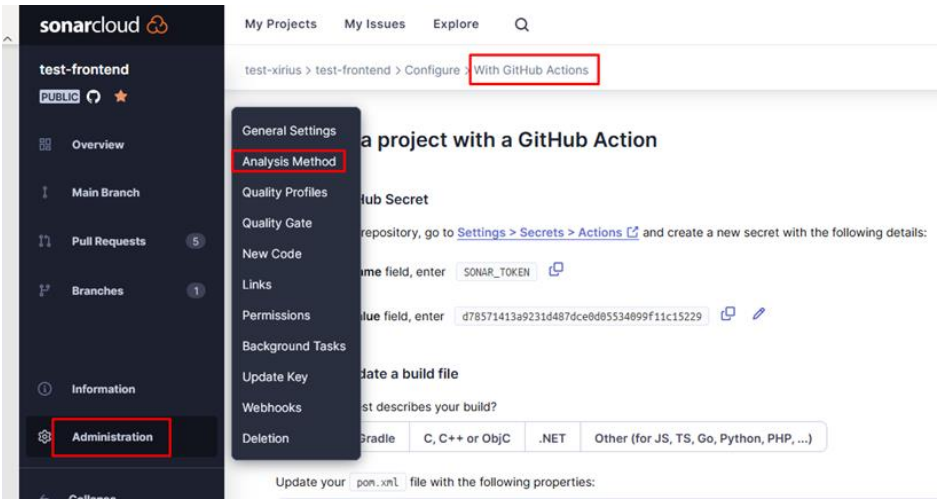

Figura 10. Selección de método de análisis

7. Luego, se debe configurar la acción secreta en el repositorio con los datos que muestra la plataforma, luego se debe añadir al proyecto una propiedad para este caso se está utilizando Maven por lo que se actualizara con la nueva propiedad proporcionada por la plataforma Sonar Cloud el pom.xml.

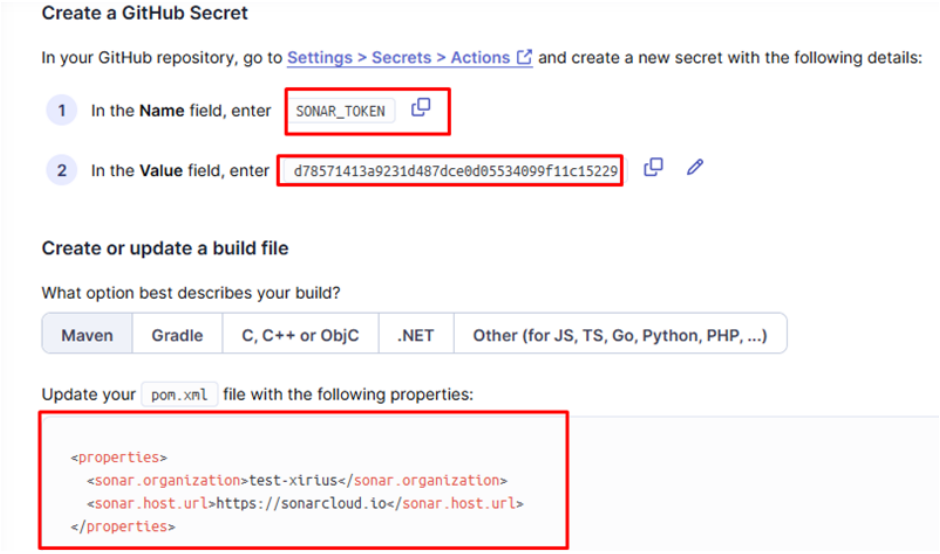

Figura 11. Configuración de llave secreta y propiedad en pom.xml

8. Ahora, en el repositorio se debe crear o actualizar el siguiente archivo “.github/workflows/gcp.yml” y se debe copiar el script que la plataforma provee, como se muestra en la siguiente imagen, por defecto el script solo analiza la rama main, si se quiere añadir otra rama, en la parte de branch se puede realizar dicha modificación, añadiendo el nombre de la nueva rama.

.

```yaml
name: SonarCloud
on:
  push:
    branches:
      - master
  pull_request:
    types: [opened, synchronize, reopened]
jobs:
  build:
    name: Build and analyze
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
        with:
          fetch-depth: 0  # Shallow clones should be disabled for a better relevancy of analysis
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: 11
          distribution: 'zulu' # Alternative distribution options are available.
      - name: Cache SonarCloud packages
        uses: actions/cache@v3
        with:
          path: ~/.sonar/cache
          key: ${{ runner.os }}-sonar
          restore-keys: ${{ runner.os }}-sonar
      - name: Cache Maven packages
        uses: actions/cache@v3
        with:
          path: ~/.m2
          key: ${{ runner.os }}-m2-${{ hashFiles('**/pom.xml') }}
          restore-keys: ${{ runner.os }}-m2
      - name: Build and analyze
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}  # Needed to get PR information, if any
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
        run: mvn -B verify org.sonarsource.scanner.maven:sonar-maven-plugin:sonar -Dsonar.projectKey=backend-java_jlkjj
```

Figura 12. Flujo de trabajo en Git Actions

9. Una vez finalizada la configuración, el proyecto ya está listo para ser analizado, y cada vez que se hace una nueva inserción de código en GitHub por medio de pull request, será analizado por la herramienta Sonar Cloud con el fin de garantizar una mejor calidad de código.

5.3.6. Implementación de script diseñado anteriormente

```yaml
name: GitHub CI/CD Cloudrun
# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events in develop branch
  push:
    branches: [ develop]
  pull_request:
    types: [opened, synchronize, reopened]
env:
  REGION: us-east4 # Cloud Run zone
  PROJECT_ID: xirius-test # GCP project
  BASE_IMAGE: gcr.io/xirius-test/master-computing #Container registry entry for the api
  SERVICE_NAME: master-computing #Cloud run service name

# A workflow run is made up of one or more jobs that can run sequentially or in parallel
jobs:
  # This workflow contains a single job called "test-build-deploy"
  test-build-deploy:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest
    # Enviroment for deployment
    environment:
          name: 'Testing'
```

Figura 13. Activación y configuración inicial del flujo de trabajo

En la anterior imagen se evidencia el código para GitHub Actions donde se tiene la activación del flujo de trabajo mediante un disparador de tipo push o pull request sobre la rama develop, esto con el fin de hacer la integración y despliegue continuos de cada una de las nuevas funcionalidades, luego se configura la región donde se realizara el despliegue posteriormente variables como identificador del proyecto, base de imagen registrada en Container registry y el nombre de servicio para hacer la conexión con Cloud Run, finalmente se define el entorno de despliegue, para este caso es Testing.

A continuación, en la Figura 14 se muestra el script con los pasos previos al despliegue:

```
48          - name: Set up Cloud SDK
49            uses: google-github-actions/setup-gcloud@v0
50            with:
51              project_id: ${{ secrets.GCP_PROJECT_ID_TEST }}
52              service_account_key: ${{ secrets.GCP_SA_KEY_TEST }}
53              export_default_credentials: true
54
55          - name: Login to GCR # Login to GCP
56            uses: docker/login-action@v1
57            with:
58              registry: gcr.io
59              username: _json_key
60              password: ${{ secrets.GCP_SA_KEY_TEST }}
61
62          - name: LOGIN GCP ACTIONS
63            uses: google-github-actions/auth@v0
64            with:
65              credentials_json: ${{ secrets.GCP_SA_KEY_TEST }}
66
67          - name: Build & Publish Image # Use the dockerfile to publish image
68            uses: docker/build-push-action@v2
69            id: build
70            with:
71              context: .
72              push: true
73              tags: ${{ env.BASE_IMAGE }}:${{ github.sha }}
```

Figura 14. Pasos previos para desplegar

Pasos para desplegar usando Cloud Run

```
75          - name: Deploy to Cloud Run
76            id: deploy
77            uses: google-github-actions/deploy-cloudrun@main
78            with:
79              region: ${{ env.REGION }}
80              service: ${{ env.SERVICE_NAME }}
81              image: ${{ env.BASE_IMAGE }}:${{ github.sha }}
```

Figura 15. Despliegue en Cloud Run

En la anterior imagen se muestra el despliegue en cloud run haciendo git actions, también se utilizan las variables definidas inicialmente en el flujo de trabajo para indicar la zona, el nombre del servicio para despliegue y la imagen con su respectiva llave sha para garantizar la integridad y que se esté utilizando la imagen correcta.

Capítulo 6- Evaluación del proceso propuesto

7.1. Definición de los objetivos específicos que se espera alcanzar con el programa de medición

- Evaluar el impacto del proceso de DevOps propuesto en la empresa XIRIUS, a través de la implementación de buenas prácticas de integración continua (IC), análisis estático de código (AEC) y despliegue continuo (DC), utilizando un proceso de automatización.

7.2. Definición de las preguntas que se deben responder para lograr los objetivos.

- ¿Cuál es el impacto del proceso de DevOps propuesto en la calidad del software?
- ¿Cuál es el impacto del proceso de DevOps propuesto en los tiempos de entrega?
- ¿Qué beneficios se obtienen de la implementación de prácticas de IC, AEC y DC?

7.3. Selección de las métricas que se utilizarán para responder a las preguntas.

Las métricas seleccionadas son:

- Número de despliegues exitosos antes de la implementación de las prácticas por semana (NDEA)
- Número de no conformes que deben ser medidos antes de la implementación de las prácticas por semana (NNA)
- Número de despliegues exitosos después de la implementación de las prácticas por semana (NDED).
- Número de no conformes que deben ser medidos después de la implementación de las prácticas por semana (NND).

7.4. Recopilación de datos

Tabla 16. Recopilación de datos antes y después de implementar proceso propuesto

Pregunta	PF	Antes de implementar proceso DevOps	Después de implementar proceso DevOps
¿Aproximadamente cuántos despliegues se realizan por semana en un proyecto en estado de	PF1	4	5
	PF2	5	7
	PF3	3	6

desarrollo?			
¿Aproximadamente cuánto tiempo toma la revisión manual de un pull request realizado por otro Ingeniero de desarrollo?	PF1	Normalmente 5 minutos y pull complejos 10 minutos.	Normalmente 4 minutos y pull complejos 10 minutos
	PF2	Depende del tamaño y complejidad. La mayoría toman 4 minutos, y pull request complejos 10 minutos.	Depende del tamaño y complejidad. La mayoría toman 3 minutos, y pull request complejos 10 minutos
	PF3	Normalmente 6 minutos y pull complejos 15 minutos	Normalmente 4 minutos y pull complejos 10 minutos
¿Aproximadamente cuántos casos de no conformidad hay por sprint en un proyecto en estado de desarrollo?	PF1	3	2
	PF2	2	2
	PF3	4	3

7.5. Cálculo de promedios antes de implementar proceso propuesto

Promedio de despliegues por semana aproximados = (4 + 5 + 3) / 3 = 4
Promedio tomado para revisión de pull request complejos = (10 + 10 + 15) / 3 = 12 minutos
Promedio tomado para revisión de pull request con baja complejidad = (5 + 4 + 6) / 3 = 5 minutos
Promedio de casos de no conformidad por sprint = (3 + 2 + 4) / 3 = 3

7.6. Cálculo de promedios después de implementar proceso propuesto

Promedio de despliegues por semana = (5 + 7 + 6) / 3 = 6
Promedio tomado para revisión de pull request complejos = (10 + 10 + 10) / 3 = 10 minutos
Promedio tomado para revisión de pull request con baja complejidad (3 + 4 + 5) = 4 minutos
Promedio de casos de no conformidad por sprint = (2 + 2 + 3) / 3 = 2

7.7. Interpretación de datos

Teniendo en cuenta la recopilación de datos realizada anteriormente, podemos decir que la implementación del proceso propuesto ha aportado positivamente en el desarrollo de software llevado a cabo en la empresa y respondiendo a las preguntas planteadas inicialmente, se obtiene las siguientes respuestas.

- ¿Cuál es el impacto del proceso de DevOps propuesto en la calidad del software?

El impacto del proceso DevOps fue positivo en la calidad del software debido a que con ayuda de SonarCloud los desarrolladores fueron mejorando sus prácticas de programación y se minimizo el número casos de no conformidades. Adicionalmente, los desarrolladores van aprendiendo sobre como desarrollan mejorando su forma de trabajo proyecto a proyecto.

- ¿Cuál es el impacto del proceso de DevOps propuesto en la velocidad de entrega?

  El impacto fue positivo debido a que se realizó un número mayor de despliegues en ambiente de desarrollo, logrando hacer que se pueda probar los servicios expuestos por la aplicación software mediante pruebas de aceptación y pasar a producción de forma más rápida, una vez se hayan hecho las distintas pruebas al software.

- ¿Qué beneficios se obtienen de la implementación de prácticas de IC, AEC y DC?

  Uno de los beneficios ha sido la minimización del tiempo promedio de revisión de pull request, con una reducción de 12 a 10 minutos para pull request de alta complejidad y de 5 a 4 minutos para pull request de baja complejidad. Esto indica que la automatización de análisis de código estático ha mejorado la velocidad y precisión de las revisiones de código.

Con base a la información proporcionada, se puede concluir que la implementación de un pipeline de integración continua, análisis de código y despliegue continuo ha resultado positivo en el proceso de desarrollo de software debido a las siguientes razones:

- En primer lugar, el número promedio de despliegues exitosos por semana ha aumentado de 4 a 6, lo que indica que la automatización de procesos y el uso de herramientas especializadas como SonarCloud y Google Cloud han ayudado a mejorar el proceso de desarrollo.

- En segundo lugar, el tiempo promedio de revisión de solicitudes de extracción ha disminuido, con una reducción de 12 a 10 minutos para solicitudes de alta complejidad y de 5 a 4 minutos para solicitudes de baja complejidad. Esto sugiere que la implementación de un pipeline de Build y análisis de código automatizado ha mejorado la velocidad y precisión de las revisiones de código.

- En tercer lugar, el número promedio de no conformidades por sprint ha disminuido de 3 a 2, lo que indica que la implementación de pruebas de calidad automatizadas y el despliegue continuo han mejorado la calidad de los cambios de código.

7.8. Generación de reporte de resultados

Se genero un reporte de resultados con las herramientas Sonar Cloud en cual se obtienen las siguientes graficas.

En la siguiente grafica se muestra el perfil de calidad configurado, en el cual se pueden observar la cantidad de reglas activas para detectar bugs o errores, vulnerabilidades, código oloroso y seguridad en el código, para este caso se está utilizando lenguaje de programación Java. Las reglas inactivas se deben a que ya son obsoletos por lo que no es conveniente que se aplique dicha revisión.

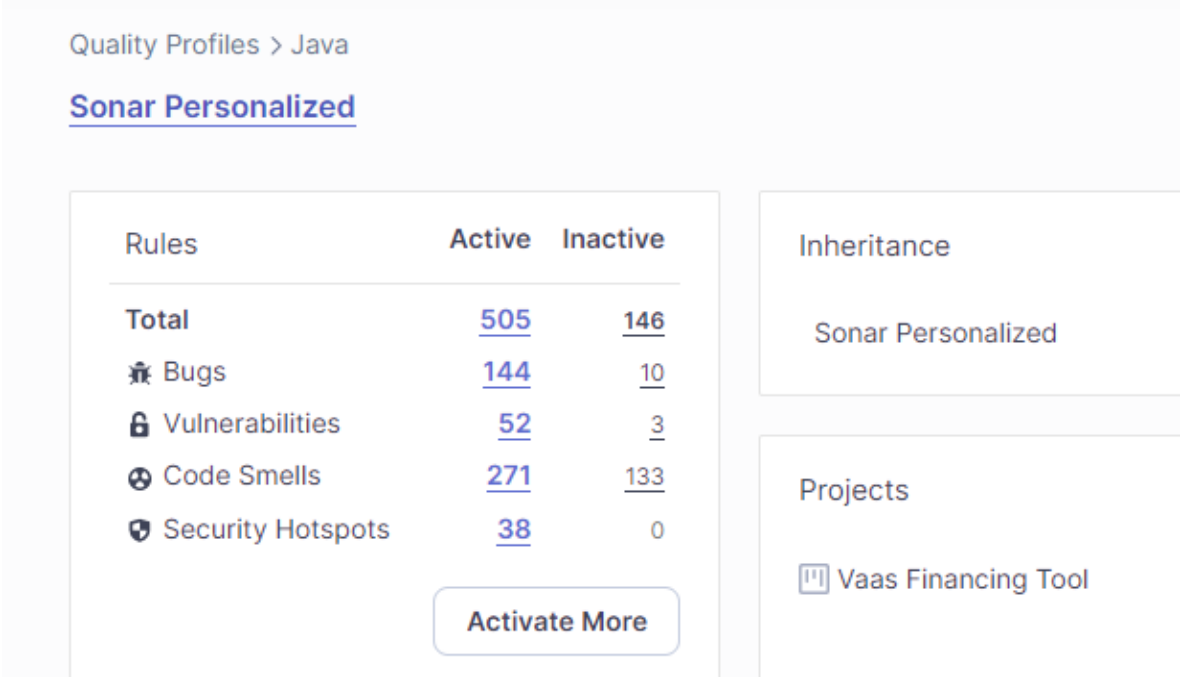

Figura 16 Perfil de calidad configurado

Sonar Cloud realiza un análisis teniendo en cuenta ciertos niveles de gravedad, en la siguiente imagen se puede observar que se presenta código oloroso de tipo informativo, es decir no tiene mucha gravedad y repercusión en el aplicativo, sin embargo, es importante que en etapas posteriores se completen dichas tareas.

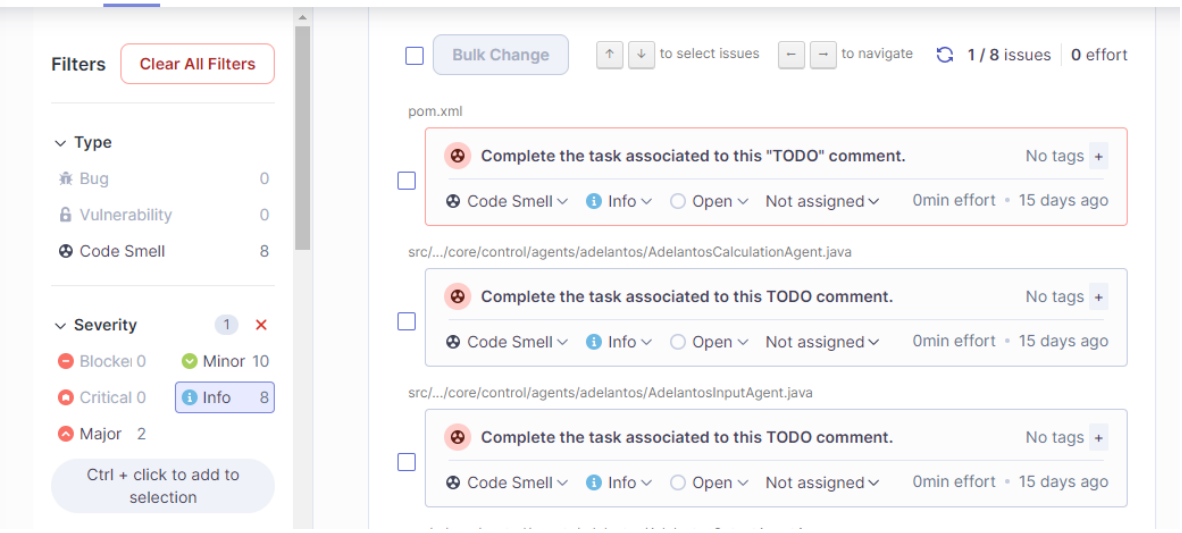

Figura 17 Código oloroso de tipo informativo

En la siguiente imagen se muestra el código oloroso con gravedad menor encontrado, el primero hace referencia a que cualquier método debe iniciar con minúscula, el segundo indica que para retornar un resultado no es necesario asignar a una variable el valor u objeto obtenido y luego retornar, sino que se puede retornar inmediatamente el valor u objeto, el tercero y último hace referencia a que todas las variables deben seguir el nombrado CamelCase. Esto ayuda a que los desarrolladores mejoren sus prácticas de programación y se mejore la calidad de código.

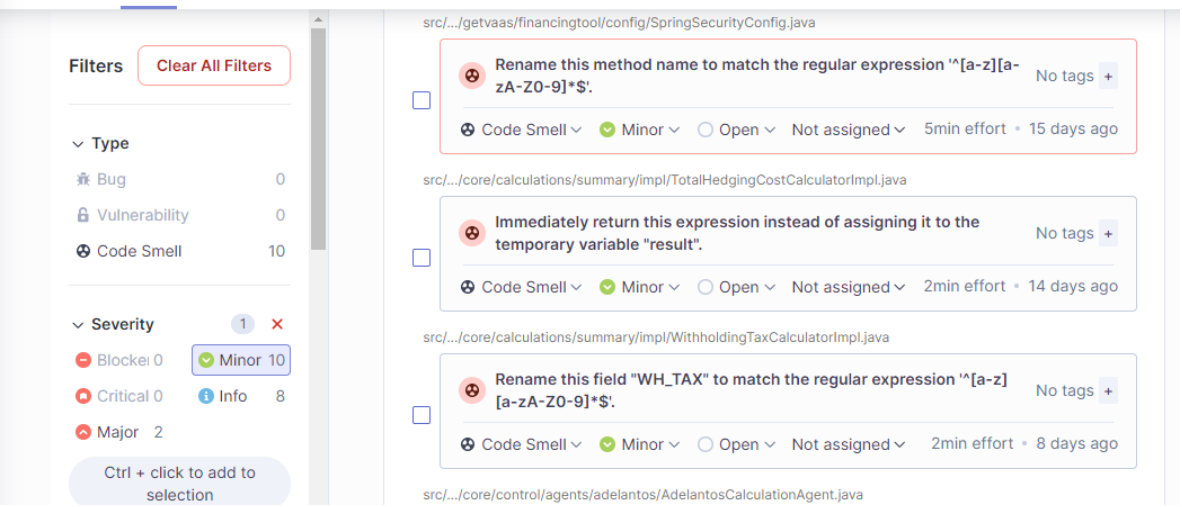

Figura 18. Código oloroso con gravedad menor

A continuación, se muestran los dos errores con gravedad mayor que se obtuvieron en el análisis de código estático, estos tienen un impacto negativo en el rendimiento y correcto funcionamiento de la aplicación, el primero hace referencia a que cuando se tiene una clase solo con métodos estáticos, se debe declarar un constructor explicito privado, ya que de lo contrario se crea un constructor público por defecto, lo cual sería una mala práctica, ya que para utilizar los métodos estáticos no es necesario tener una instancia de la clase. Con respecto al segundo error, indica los valores en las pruebas con JUnit están en el orden incorrecto, es decir el valor esperado y el valor llegado están intercambiados, por lo que a la hora de surgir un error en la prueba puede ser confuso de entender, y se recomienda se coloque en el orden adecuado para facilitar la compresión, cuando no logre pasar una prueba unitaria.

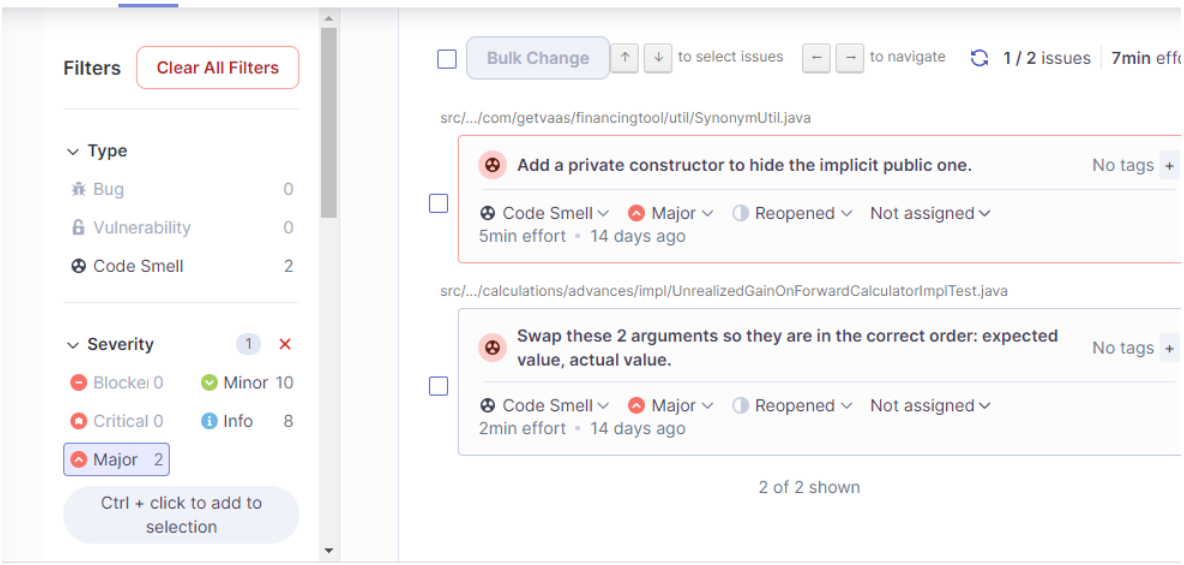

Figura 19 Código oloroso con gravedad mayor

A continuación, se muestra el resumen de análisis estático de código realizado con la herramienta Sonar Cloud para el proyecto, en el cual se puede observar cómo se fue minimizando la cantidad de código oloroso presentado desde el primer pull request hasta el último, esto se debe a que los desarrolladores una vez reconocían la mala práctica que estaban cometiendo y en cada reunión de sprint se realizó una revisión con los errores más comunes con el fin dar una retroalimentación, con el fin de evitar que se repitan las malas prácticas ya cometidas y el equipo mejore continuamente.

La herramienta Sonar Cloud aporto de manera positiva ya que ayudo a mejorar la calidad de código, mejorar las prácticas de programación llevadas a cabo por los desarrolladores e impulso la cultura DevOps en la empresa la cual busca ir mejorando continuamente. En la siguiente grafica se puede apreciar dicho efecto.

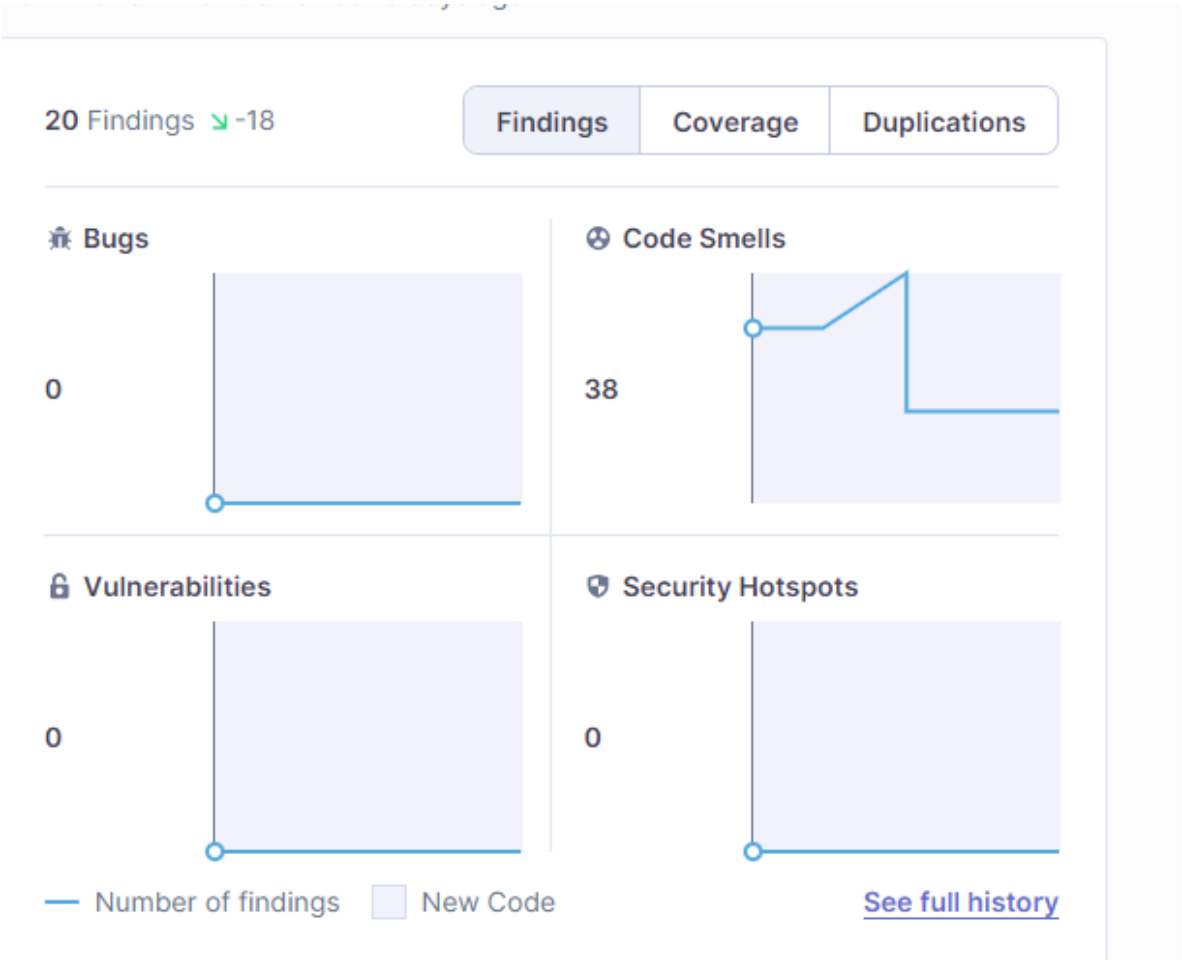

Figura 20. Resumen general de hallazgos en Sonar Cloud

Capítulo 7- Lecciones aprendidas

Una de las lecciones aprendidas con este proyecto es la importancia de la cultura DevOps en el campo del desarrollo de software. Si cada desarrollador la adopta y la practica puede aprender y mejorar de forma continua a partir de sus errores o falencias. El objetivo final es desarrollar un software cada vez mejor y cumplir los requerimientos del cliente.

Con la ayuda de este proyecto se logra comprender que los pequeños detalles como desde la forma de escribir una línea de código, dar el nombre a una variable, hasta hacer una funcionalidad compleja, se debe tener en cuenta estándares de programación ya que al final, esto va tener un repercusión sobre todo el proyecto y puede generar que el código sea mantenible y escalable o de lo contrario que no se

pueda modificar, es decir sea difícil de entender y extender debido a malas prácticas de programadas usadas, a lo que muchas veces se debe generar un nuevo código

En ese mismo, el uso de SonarLint en IDE IntelliJ como primera barrera para evitar las malas prácticas de programación fue de mucha ayuda, debido a que gracias a esto cada desarrollador primero verifica su código localmente, luego realiza un pull request a la rama develop, donde se hace el análisis estático con Sonar Cloud, con esto corroborar que se cumple con la calidad necesaria y que otro desarrollador valide la lógica implementada.

Capítulo 8- Conclusiones

La adopción de prácticas DevOps en las empresas es un proceso difícil debido a que no se tiene una guía clara de cómo implementar, lo que con lleva a mucha incertidumbre, ya que depende de cada tipo de empresa, tener en cuenta el contexto según esto observar que prácticas son viables de aplicar, con el proyecto realizado se puede concluir que la implementación de análisis estático de código ha aportado de manera positiva a la empresa, debido a que se minimiza el tiempo dedicado a revisión de código, así como también se encuentran defectos y malas prácticas de código de forma rápida, esto ayuda a que se minimice el número de no conformidades presentados por sprint. En este mismo sentido, SonarCloud recomienda utilizar como una primera barrera contra las malas prácticas, el uso de la extensión SonarLint que es gratuita, con el fin de mejorar con respecto buenas prácticas de desarrollo y estándares de programación, esto ha ayuda que el proceso de desarrollo sea más efectivo, seguro y rápido.

Teniendo en cuenta los resultados obtenidos se puede inferir que la implementación de prácticas DevOps, ha generado mejoras significativas en la calidad del software y en la eficiencia del proceso de desarrollo. Estos resultados respaldan la importancia de adoptar una cultura DevOps en la cual los desarrolladores y el equipo de trabajo desempeñen un rol fundamental para mejorar continuamente el proceso de desarrollo de software en una empresa.

En conclusión, la implementación de prácticas DevOps ha demostrado su relevancia y beneficios en el ámbito empresarial. Desde la automatización de pruebas unitarias hasta la creación de arquitecturas, DevOps ofrece un amplio campo de aplicación. En este proyecto específico, la adopción de tres prácticas esenciales ha generado resultados favorables en términos de calidad del software y velocidad de entrega al cliente. Además, se ha destacado que DevOps no se limita únicamente a prácticas técnicas, sino que representa una cultura en la cual los desarrolladores y el equipo de trabajo desempeñan un papel esencial para mejorar el proceso de desarrollo de software en una empresa.

Capítulo 9- Bibliografía

[1] "[PDF] Towards the adoption of DevOps in software product organizations: A maturity model approach | Semantic Scholar." https://www.semanticscholar.org/paper/Towards-the-adoption-of-DevOps-in-software-product-Feijter-Vliet/b91f23a8d8cc3932a28889e69d334a662900e5a0 (accessed Apr. 11, 2022).

[2] A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," *Comput. Sci. Rev.*, vol. 38, p. 100308, Nov. 2020, doi: 10.1016/J.COSREV.2020.100308.

[3] S. Rafi, M. A. Akbar, W. Yu, A. Alsanad, A. Gumaei, and M. U. Sarwar, "Exploration of DevOps testing process capabilities: An ISM and fuzzy TOPSIS analysis," *Appl. Soft Comput.*, vol. 116, p. 108377, Feb. 2022, doi: 10.1016/J.ASOC.2021.108377.

[4] J. Díaz, D. López-Fernández, J. Pérez, and Á. González-Prieto, "Why are many businesses instilling a DevOps culture into their organization?," *Empir. Softw. Eng.*, vol. 26, no. 2, pp. 1–50, Mar. 2021, doi: 10.1007/S10664-020-09919-3/TABLES/17.

[5] L. E. Lwakatare *et al.*, "DevOps in practice: A multiple case study of five companies," *Inf. Softw. Technol.*, vol. 114, pp. 217–230, Oct. 2019, doi: 10.1016/J.INFSOF.2019.06.010.

[6] M.-A. Pastrana-Pardo, H.-A. Ordóñez-Erazo, C.-A. Cobos-Lozada, M.-A. Pastrana-Pardo, H.-A. Ordóñez-Erazo, and C.-A. Cobos-Lozada, "Approach to the Best Practices in Software Development Based on DevOps and SCRUM Used in Very Small Entities," *Rev. Fac. Ing.*, vol. 31, no. 61, p. e14828, Sep. 2022, doi: 10.19053/01211129.V31.N61.2022.14828.

[7] B. Chess and G. Mcgraw, "Static analysis for security," *IEEE Secur. Priv.*, vol. 2, no. 6, pp. 76–79, Nov. 2004, doi: 10.1109/MSP.2004.111.

[8] M. Pastrana, H. Ordóñez, A. Rojas, and A. Ordoñez, "Ensuring Compliance with Sprint Requirements in SCRUM: Preventive Quality Assurance in SCRUM," *Adv. Intell. Syst. Comput.*, vol. 924, pp. 33–45, 2019, doi: 10.1007/978-981-13-6861-5_3.

[9] D. Yang *et al.*, "DevOps in Practice for Education Management Information System at ECNU," *Procedia Comput. Sci.*, vol. 176, pp. 1382–1391, Jan. 2020, doi: 10.1016/J.PROCS.2020.09.148.

[10] M. Alejandro, P. Pardo, H. Armando Ordoñez Erazo, and C. A. Cobos Lozada, "Documenting and implementing DevOps good practices with test automation and continuous deployment tools through software refinement," *Period. Eng. Nat. Sci.*, vol. 9, no. 4, pp. 854–863, Nov. 2021, doi: 10.21533/PEN.V9I4.2239.G1002.

[11] "6 Principles of DevOps – DevOps Agile Skills Association (DASA)." https://www.devopsagileskills.org/dasa-devops-principles/ (accessed Apr. 11, 2022).

[12] "Hello World - GitHub Docs." https://docs.github.com/en/get-started/quickstart/hello-world (accessed Apr. 11, 2022).

[13] K. Priyadarsini, E. F. I. Raj, A. Y. Begum, and V. Shanmugasundaram, "Comparing DevOps procedures from the context of a systems engineer," *Mater.*

Today Proc., Oct. 2020, doi: 10.1016/J.MATPR.2020.09.624.
[14] "Descripción general de Google Cloud | Overview." https://cloud.google.com/docs/overview (accessed Apr. 22, 2022).
[15] "¿Qué es el análisis de código estático? | Guía de CI/CD de TeamCity | JetBrains." https://www.jetbrains.com/es-es/teamcity/ci-cd-guide/concepts/static-code-analysis/ (accessed May 16, 2023).
[16] "OWASP Foundation | Open Source Foundation for Application Security." https://owasp.org/ (accessed Apr. 11, 2022).
[17] "Resumen de Jira | Productos, proyectos y alojamiento." https://www.atlassian.com/es/software/jira/guides/getting-started/introduction (accessed Mar. 09, 2023).
[18] "Plataforma de almacenamiento personal en la nube y uso compartido de archivos - Google." https://www.google.com/intl/es/drive/ (accessed Mar. 10, 2023).
[19] "git commit | Atlassian Git Tutorial." https://www.atlassian.com/es/git/tutorials/saving-changes/git-commit (accessed Jul. 02, 2023).
[20] "Acerca de las solicitudes de incorporación de cambios - Documentación de GitHub." https://docs.github.com/es/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests (accessed Jul. 02, 2023).
[21] Google and DORA, "Accelerate State of DevOps 2022," 2023.
[22] M. Pastrana, H. Ordóñez, C. Cobos, M. Pastrana, H. Ordóñez, and C. Cobos, "ISO 29110 en Colombia: de la teoría a la práctica," *Rev. Guillermo Ockham*, vol. 17, no. 2, pp. 71–80, Dec. 2019, doi: 10.21500/22563202.4299.
[23] "Tecnología de DevOps: Administración de cambios en la base de datos | Capacidades de DevOps | Google Cloud." https://cloud.google.com/architecture/devops/devops-tech-database-change-management?hl=es-419 (accessed Mar. 10, 2023).
[24] A. Kaur and R. Nayyar, "A Comparative Study of Static Code Analysis tools for Vulnerability Detection in C/C++ and JAVA Source Code," *Procedia Comput. Sci.*, vol. 171, pp. 2023–2029, Jan. 2020, doi: 10.1016/J.PROCS.2020.04.217.
[25] W. A. Leiton Muñoz and B. A. Moya Loaiza, "Documentación e implementación de buenas prácticas DevOps con herramientas de automatización de pruebas y de despliegue a través del refinamiento de software de Smart Campus," 2020, doi: 10.1/JQUERY.MIN.JS.
[26] "Inicio | Documentos de SonarCloud." https://docs.sonarcloud.io/ (accessed May 09, 2022).
[27] "Enriching Your Analysis | SonarCloud Docs." https://docs.sonarcloud.io/enriching/overview/ (accessed Mar. 11, 2023).
[28] S. Camilo Certuche Díaz Karen Andrea Zúñiga Galíndez Director, C. Jesús Pardo Calvache Codirector, and J. Guerrero Astaiza, "Proceso para soportar DevOps en la integración, entrega y despliegue continuo en pequeñas y medianas empresas de desarrollo de software," 2021, Accessed: Mar. 13, 2023. [Online]. Available: http://repositorio.unicauca.edu.co:8080/xmlui/handle/123456789/5834.
[29] "Cámara de la Industria Argentina del Software - CESSI." https://cessi.org.ar/perfil_it/arquitecto-de-software/ (accessed Mar. 13, 2023).
[30] "Cámara de la Industria Argentina del Software - CESSI." https://cessi.org.ar/perfil_it/desarrollador-de-software/ (accessed Mar. 13, 2023).

[31] “Descripción del puesto: Ingeniero de pruebas (m/h/x) exemple - Workable.” https://resources.workable.com/es/ingeniero-de-pruebas-descripcion-del-puesto (accessed Mar. 13, 2023).
[32] T. Wood-Harper, “Research methods in information systems: using action research,” *Res. methods Inf. Syst.*, vol. 169, no. Jackson 1977, p. 191, 1985.
[33] “Dockerfile y contenedores de Windows | Microsoft Learn.” https://learn.microsoft.com/es-es/virtualization/windowscontainers/manage-docker/manage-windows-dockerfile (accessed Mar. 30, 2023).
[34] “Git.” https://git-scm.com/ (accessed Apr. 09, 2023).
[35] J. Homepage, M.-A. Pastrana-Pardo, H.-A. Ordóñez-Erazo, and C.-A. Cobos-Lozada, “Process Model Represented in BPMN for Guiding the Implementation of Software Development Practices in Very Small Companies Harmonizing DEVOPS and SCRUM,” *Rev. Fac. Ing.*, vol. 31, no. 62, pp. e15207–e15207, Dec. 2022, doi: 10.19053/01211129.V31.N62.2022.15207.
[36] “Implementa en Cloud Run | Documentación de Cloud Run | Google Cloud.” https://cloud.google.com/run/docs/deploying?hl=es-419#console (accessed Apr. 09, 2023).