Sistema de estudios de Posgrado

Maestría en Tecnologías de la Información (MATI)

Gestión de servicios y Productos TIC

Diseño de una estrategia para la implementación de una cultura DevOps en el Instituto de Fomento y Asesoría Municipal, IFAM.

Ronald Solís Sanabria

Heredia, Costa Rica, 22 noviembre del 2023.

INDICE GENERAL

INDICE DE TABLAS

INDICE DE ILUSTRACIONES

DECLARACIÓN JURADA DE RESPETO AL DERECHO DE AUTOR

22 de noviembre del año 2023

Universidad Nacional
Facultad de Ciencias Exactas y Naturales
Escuela de Informática
Posgrado en Gestión de la Tecnología de Información y Comunicación (ProGesTIC)

FORMULARIO DE DEPÓSITO LEGAL, AUTORIZACIÓN DE USO DE DERECHOS PATRIMONIALES DE AUTOR E INCORPORACIÓN A REPOSITORIOS INSTITUCIONALES DE INFORMACIÓN DE ACCESO PÚBLICO.

La persona abajo firmante, en condición de estudiante de la Maestría en Tecnologías de la Información (MATI), Ronald Solís Sanabria y autor del trabajo final de graduación titulado “Diseño de una estrategia para la implementación de una cultura DevOps en el Instituto de Fomento y Asesoría Municipal, IFAM.” para optar al grado académico de Máster en: Administración de Tecnologías de Información con énfasis en Gestión de servicios y productos TIC.

De conformidad con lo establecido en el documento de “lineamientos generales para la realización del trabajo final de graduación” y demás normativa universitaria relacionada con estos trabajos de graduación, DECLARO BAJO FE DE JURAMENTO conociendo la responsabilidad civil, penal o administrativa en que podría incurrir al no decir la verdad, lo siguiente:

El documento, producto, obra audiovisual, software, resultado del trabajo final de graduación referido anteriormente es original, inédito y ha cumplido con todo el proceso de aprobación académico que confiere el grado académico postulado con esta obra.

El trabajo final de graduación referido anteriormente constituye una producción intelectual propia de la persona abajo firmante y a esta fecha no ha sido divulgado a terceros(as) de forma pública, por ningún medio de difusión impreso o digital.

Autorizo el depósito de un ejemplar en formato impreso y otro en formato digital (entregado en soporte de disco compacto), en la colección de trabajos finales de graduación del ProGesTIC de la Universidad Nacional, así como la realización de copias electrónicas adicionales para fines exclusivos de seguridad y conservación de la información.

1. En caso de que el trabajo final de graduación haya sido elaborado como obra en colaboración -bien se trate de obras en las que los autores(as) tienen el mismo grado de participación o aquellas en las que existe una persona autora principal y una o varias personas autoras secundarias-, todos(as) ellos(as) han contribuido intelectualmente en la elaboración del documento y en este acto, libero de responsabilidad a las autoridades del posgrado y a los funcionarios que custodian la colección del ProGesTIC, en relación con el reconocimiento que se realiza respecto de los niveles de participación asignados por el propio autor del proyecto.

2. En caso de que el trabajo final de graduación haya sido elaborado como obras en colaboración (conforme a lo dispuesto en el punto 4), el autor abajo firmante designa a _____como encargado(a) de recibir comunicaciones y representar con autoridad suficiente a los suscritos, en condición de agente autorizado(a) de los demás autores(as).

3. Reconozco que la colección de trabajos finales del ProGesTIC no emite criterios ni valoraciones académicas sobre lo planteado en el producto final del trabajo de

graduación y autorizo a esta dependencia para que proceda a poner a disposición del público la obra en mención, a través de los espacios físicos o virtuales que se posea, así como a través del Repositorio Institucional; a partir del cual los usuarios de dichas plataformas puedan acceder al documento y hacer uso de este en el marco de los fines académicos, no lucrativos y de respeto a la integridad del contenido del mismo así como la mención del autor o poseedor de sus derechos.

4. Manifiesto que todos los datos de citas dentro de texto y sus respectivas referencias bibliográficas, así como las tablas y figuras (ilustraciones, fotografías, dibujos, mapas, esquemas u otros) tienen la fuente y el crédito debidamente identificados y se han respetado los derechos de autor.

5. Autorizo la licencia gratuita no exclusiva de los derechos patrimoniales de autor para reproducir, traducir, distribuir y poner a disposición pública en formato electrónico, el documento depositado, para fines académicos, no lucrativos y por plazo indefinido en favor de la Universidad Nacional, que incluye además los siguientes actos:

   - La publicación y reproducción íntegra de la obra o parte de esta, tanto por medios impresos como electrónicos, incluyendo Internet y cualquier otra tecnología conocida o por conocer.

   - La traducción a cualquier idioma o dialecto de la obra o parte de esta.

   - La adaptación de la obra a formatos de lectura, sonido, voz y cualquier otra representación o mecanismo técnico disponible, que posibilite su acceso para personas no videntes parcial o totalmente, o con alguna otra forma de capacidades especiales que le impida su acceso a la lectura convencional del proyecto.

- La distribución y puesta a disposición de la obra al público, de tal forma que el público pueda tener acceso a ella desde el momento y lugar que cada uno elija, a través de los mecanismos físicos o electrónicos de que disponga.

- Cualquier otra forma de utilización, proceso o sistema conocido o por conocerse que se relacione con las actividades y fines académicos a los cuales se vincula la maestría, la colección de trabajos finales del ProGesTIC, la Escuela de Informática y la Universidad Nacional.

6. Reconozco que la colección de trabajos del ProGesTIC manifiesta actuar con diligencia para evitar la existencia en su sitio web de contenidos ilícitos y en caso de que tenga conocimiento efectivo de la existencia de infracciones a los derechos de propiedad intelectual, se reserva el derecho de proceder a bloquear el acceso durante el trámite del debido proceso para comprobar el incumplimiento y en caso de verificarse la falta, retirar definitivamente el acceso al proyecto depositado.

7. Acepto que la publicación y puesta a disposición del público del trabajo final de graduación, así como la presente autorización de uso de la obra, se regirá por la normativa institucional de la Universidad Nacional y la legislación de la República de Costa Rica. Adicionalmente, en caso de cualquier eventual diferencia de criterio o disputa futura, acepto que esta se dirimirá de acuerdo con los mecanismos de Resolución Alterna de Conflictos y la Jurisdicción Costarricense.

Autor: Ronald Solís Sanabria

RONALD ANDRES SOLIS SANABRIA (FIRMA)

Firmado digitalmente por RONALD ANDRES SOLIS SANABRIA (FIRMA)
Fecha: 2023.12.12 19:15:09 -06'00'

Fecha de entrega: 22 de noviembre del año 2023

Correo electrónico: rsolis@ifam.go.cr

AGRADECIMIENTOS

Expresar mi profundo agradecimiento a todas las personas que han contribuido de manera significativa a la realización de este trabajo final de graduación.

En primer lugar, quiero agradecer a mi familia, quienes han sido mi fuente de apoyo a lo largo de esta travesía académica. Su amor incondicional y constante han sido mi mayor motivación, impulsándome a alcanzar esta meta.

Asimismo, deseo expresar mi reconocimiento al Instituto de Fomento y Asesoría Municipal (IFAM), por brindarme la oportunidad de acceder a recursos y conocimientos fundamentales para el desarrollo de este proyecto de investigación. De igual manera agradezco al instituto el haber tomado en cuenta mi desarrollo profesional para brindar mejores aportes a la institución y gobiernos locales de todo el país.

Adicionalmente, agradecer la colaboración y guía por parte del profesor tutor, Tomás Rodríguez Arias. Su guía experta, paciencia y dedicación cada semana han sido fundamentales para la culminación exitosa de este trabajo de investigación gracias a sus comentarios constructivos y orientación profesional.

Finalmente, agradezco a todos aquellos que, de manera directa o indirecta, han contribuido a la realización de este trabajo final de graduación. Este logro no solo es mío, sino también de aquellos que han dejado su huella en mi camino académico y profesional.

CAPÍTULO I. EL PROBLEMA Y SU IMPORTANCIA

1.1 ANTECEDENTES

El Instituto de Fomento y Asesoría Municipal, IFAM, nace en la década de los años 70 como una forma de impulsar el desarrollo local. El Modelo de Servicio Futuro del IFAM se fundamenta en la Visión de Largo Plazo y Agenda Estratégica del Régimen Municipal de Costa Rica, con el fin de asegurar que las acciones del Instituto tengan un alineamiento directo y de impacto al desarrollo local del país.

Dentro de su plan estratégico, el IFAM se ha enfocado en crear varios productos de software de alto impacto nacional, con la expectativa de que sean bien aceptados por parte de los usuarios y gestionados por los encargados de administrar los productos de software de la Unidad de Tecnologías de Información. Sin embargo, al no contar con un proceso automatizado ni una cultura DevOps integrada con otros marcos o metodologías, la gestión y mantenimiento del servicio brindado a los gobiernos locales pueden verse afectados.

1.2 JUSTIFICACIÓN

Para realizar un proceso de desarrollo de software se utiliza el procedimiento institucional para la Gestión de Desarrollo de Sistemas de Información. Sin embargo, este proceso carece de una metodología de trabajo, automatización o marcos de trabajo para la gestión de servicios y el aseguramiento de la calidad.

Debido a la escasez de personal en la Unidad de Tecnologías de Información, algunos de sus miembros realizan varios roles simultáneamente dentro del proceso de entrega de software. Esto puede llevar a una aplicación incompleta o incorrecta

de las mejores prácticas de los marcos de trabajo o metodologías para la entrega de software.

El diseño de una estrategia para la implementación de una cultura DevOps en el Instituto de Fomento y Asesoría permitirá mejorar el proceso de entrega de software y realizar más adecuadamente los procesos de mejora continua. DevOps es un enfoque que permite a las empresas acelerar el ciclo de vida del software y aumentar la satisfacción del cliente, permitirá la eliminación de desperdicios y optimizar continuamente todos los procesos involucrados en la entrega de software, lo que se traduce en una mayor eficiencia y calidad en la entrega del servicio.

Se mencionan algunos beneficios de la estrategia:

1. Mejora en la eficiencia: DevOps permite optimizar continuamente los procesos de entrega de software y reduciendo el tiempo de espera en cada fase del ciclo de vida del software. De esta forma, se puede mejorar la eficiencia en la entrega de software y aumentar la velocidad en la que se ofrecen los servicios.

2. Mayor calidad: La cultura DevOps fomenta una colaboración efectiva entre los equipos de desarrollo y operaciones, lo que conduce a una mayor calidad en la entrega de software.

3. Reducción de costos: La implementación de una cultura DevOps puede ayudar a reducir los costos asociados con el desarrollo de software, al eliminar desperdicios y mejorar la eficiencia en la entrega de software. La

mejora en la calidad del software entregado reduce los costos asociados con la solución de errores y defectos.

4. Mayor satisfacción del cliente: La mejora en la calidad de los productos de software entregados y la mayor velocidad en la que se ofrecen los servicios se traducen en una mayor satisfacción del cliente. La implementación de una cultura DevOps puede ayudar a las organizaciones a responder rápidamente a las necesidades del cliente y ofrecer productos de software que satisfagan sus expectativas.

Con la implementación de una cultura DevOps, se espera que se mejore la eficiencia, calidad y velocidad en la entrega de software, lo que permitirá una gestión y mantenimiento más eficaz del servicio ofrecido a los gobiernos locales.

1.3 PROBLEMA

Actualmente, el IFAM no cuenta con una cultura DevOps para sus equipos de desarrollo y operaciones, lo que significa que la colaboración y comunicación entre ellos no se realiza de manera eficiente, asimismo no cuenta con el personal necesario para la ejecución de tareas. Algunos procesos todavía se realizan de manera manual, como la publicación de nuevos sistemas o versiones, se identifica como importante contar con el proceso automatizado en todos los sistemas de información, y de hacerlo de manera ordenada y utilizando las mejores prácticas, marcos de trabajo o metodologías que han beneficiado a grandes empresas a nivel mundial.

1.4 OBJETIVO GENERAL

Diseñar una estrategia para la implementación de una cultura DevOps en el Instituto de Fomento y Asesoría Municipal que permita mejorar la eficiencia, calidad y velocidad en la entrega de software.

1.5 OBJETIVOS ESPECÍFICOS

1. Realizar un análisis detallado de los procesos de desarrollo y operaciones actuales de la organización para identificar las áreas de mejora y los puntos críticos en la entrega de software
2. Identificar los roles, responsabilidades y herramientas que son críticas para la implementación de una cultura DevOps en la organización
3. Definir un instrumento que permita adoptar una cultura DevOps de manera efectiva y se fomente la colaboración y comunicación entre los equipos.

1.6 METAS POR ALCANZAR POR OBJETIVO

Tabla 1: Metas por alcanzar por objetivo

Objetivos Específicos	Metas
Realizar un análisis detallado de los procesos de desarrollo y operaciones actuales de la organización para identificar las áreas de mejora y los puntos críticos en la entrega de software	• Identificar y documentar los procesos actuales de desarrollo y operaciones de la organización, incluyendo los flujos de trabajo, herramientas utilizadas y responsabilidades de los equipos involucrados. • Identificar áreas de mejora en los procesos actuales que puedan afectar la entrega de software, tales como cuellos de botella en la comunicación, retrasos en la entrega y la falta de automatización.

	• Analizar y evaluar el rendimiento de los procesos actuales en términos de eficiencia, calidad y cumplimiento de plazos. • Documentar los hallazgos y recomendaciones de mejora en un informe detallado que sirva como base para la implementación de acciones correctivas.
Identificar los roles, responsabilidades y herramientas que son críticas para la implementación de una cultura DevOps en la organización	• Identificar los roles y responsabilidades clave en el contexto de DevOps, como desarrolladores, operadores, administradores de sistemas y especialistas en seguridad, y definir claramente sus funciones y responsabilidades en el ciclo de vida del software. • Identificar las herramientas de DevOps que son relevantes para la organización y evaluar su idoneidad y capacidad para integrarse en los procesos existentes de la organización.
Definir una estrategia que permita adoptar una cultura DevOps de manera efectiva y se fomente la colaboración y comunicación entre los equipos	• Fomentar la colaboración y comunicación entre los equipos. • Diseñar un instrumento de adopción de una cultura DevOps en la organización

CAPÍTULO II. MARCO TEÓRICO O REFERENCIAL

2.1 MARCO TEÓRICO

Definición de DevOps

DevOps es un conjunto de prácticas que combinan el desarrollo de software (Dev) y la operación de sistemas (Ops) para mejorar la eficiencia y la velocidad de entrega de software. Se trata de una cultura y filosofía que promueve la colaboración, la comunicación y la automatización entre los equipos de desarrollo y operaciones.

El objetivo principal de DevOps es lograr una entrega de software más rápida, frecuente y confiable, mediante la eliminación de silos y la integración de los equipos de desarrollo y operaciones en un flujo de trabajo continuo. Para lograr esto, los equipos de DevOps utilizan herramientas y prácticas como la integración continua, la entrega continua, la automatización de pruebas y despliegues, la monitorización continua y la gestión de la configuración.

DevOps se enfoca en el ciclo de vida completo de la entrega de software, desde la planificación y el desarrollo, hasta las pruebas, el despliegue y la operación en producción. Al adoptar DevOps, las organizaciones pueden acelerar la entrega de software, reducir errores y mejorar la calidad del software, lo que resulta en una mejor experiencia para los usuarios finales y una ventaja competitiva para la organización.

Según Héctor Hugo Vargas Villarreal en su libro "DevOps. Guía práctica para implementar la cultura DevOps en tu empresa" (2018), esta metodología busca eliminar las barreras tradicionales entre los equipos de desarrollo y operaciones, y en su lugar,

fomentar la colaboración y el trabajo en equipo a lo largo de todo el ciclo de vida del software.

En resumen, DevOps propone una serie de prácticas y herramientas que permiten integrar los procesos de desarrollo, pruebas y despliegue de software de forma automatizada, continua y sin interrupciones. Entre estas prácticas y herramientas se encuentran la integración continua (CI), la entrega continua (CD), la infraestructura como código (IaC), la monitorización continua, la gestión de configuraciones, entre otras.

Cultura DevOps

La cultura es el conjunto de valores, creencias, actitudes, costumbres, normas, tradiciones y comportamientos que comparte un grupo o una sociedad en particular. La cultura no es algo estático o inmutable, sino que evoluciona y se adapta a lo largo del tiempo en función de las circunstancias y las influencias externas.

En el contexto organizacional, la cultura se refiere a la forma en que las personas en una empresa interactúan entre sí, se comunican, toman decisiones, resuelven problemas y realizan su trabajo. La cultura de una organización puede influir en su eficacia, su productividad y su capacidad para alcanzar sus objetivos.

Una cultura fuerte y positiva puede ser un factor clave en el éxito de una empresa, ya que puede fomentar la innovación, la colaboración y el compromiso de los empleados. Por otro lado, una cultura negativa o tóxica puede afectar negativamente a la moral de los empleados, la eficacia de la empresa y su reputación en el mercado.

La cultura DevOps se refiere a los valores y prácticas que fomentan la colaboración, la transparencia, la responsabilidad y la innovación en el proceso de desarrollo de software.

Según Ernesto Amigo Pérez en su libro "DevOps para desarrolladores" (2017), la cultura DevOps se basa en la eliminación de las barreras tradicionales entre los equipos de desarrollo y operaciones, y en su lugar, fomentar una cultura de colaboración y trabajo en equipo a lo largo de todo el ciclo de vida del software. Para lograr esto, se deben fomentar los siguientes valores y principios:

- Comunicación: Es fundamental que exista una comunicación fluida y constante entre los equipos de desarrollo y operaciones, para garantizar que se tomen en cuenta las necesidades y requisitos de ambos equipos durante todo el proceso de desarrollo de software.
- Colaboración: La colaboración entre los equipos de desarrollo y operaciones es esencial para garantizar que el software se desarrolle, se pruebe y se implemente de manera rápida y eficiente.
- Automatización: La automatización de procesos es un pilar fundamental de DevOps, ya que permite reducir los errores humanos, acelerar el tiempo de entrega y mejorar la calidad del software.
- Feedback continuo: La retroalimentación constante entre los equipos de desarrollo y operaciones es clave para identificar y resolver los problemas de manera rápida y eficiente, y garantizar la mejora continua del proceso de desarrollo de software.

Filosofía DevOps

Una filosofía es un conjunto de ideas, valores, creencias y principios que una persona o un grupo sostiene y utiliza para comprender el mundo, tomar decisiones y guiar sus acciones. La filosofía puede ser una forma de vida, una teoría, una visión del mundo o una actitud general hacia la existencia y la realidad.

En el contexto organizacional, una filosofía se refiere a un conjunto de creencias, valores y principios que guían las acciones y decisiones de una empresa. La filosofía organizacional establece la dirección y el propósito de la empresa, y define cómo la empresa interactúa con sus clientes, empleados, socios y el entorno en general.

La filosofía organizacional puede ser explícita o implícita, y se refleja en la cultura de la empresa, en sus políticas, prácticas y en las decisiones que toma.

Una filosofía organizacional sólida y coherente puede ayudar a la empresa a alinear sus objetivos con los valores de sus empleados y clientes, mejorar la toma de decisiones, fomentar la colaboración y la innovación, y construir una cultura fuerte y sostenible.

Según Juan Manuel Cueva Lovelle en su libro "DevOps. El camino del éxito en la gestión de sistemas y servicios IT" (2019), la filosofía DevOps promueve la colaboración, la comunicación y la automatización de procesos, con el fin de lograr una entrega continua de software y servicios de TI que cumplan con los requisitos del negocio y de los usuarios.

Esta filosofía se basa en los siguientes principios:

1. Cultura de colaboración: DevOps promueve una cultura de colaboración entre los equipos de desarrollo y operaciones, fomentando la comunicación constante y la toma de decisiones en conjunto.
2. Automatización de procesos: La automatización de procesos es esencial para reducir los errores humanos, aumentar la velocidad y la eficiencia en el proceso de desarrollo y despliegue de software.
3. Enfoque en el usuario: La filosofía DevOps tiene como objetivo entregar valor al usuario final, garantizando que el software y los servicios de TI cumplan con sus necesidades y expectativas.
4. Mejora continua: DevOps se enfoca en la mejora continua del proceso de desarrollo y despliegue de software, a través del análisis de métricas y la retroalimentación constante.

La filosofía DevOps es un enfoque integral de desarrollo de software que se basa en la colaboración, la automatización y la mejora continua, con el objetivo de lograr una entrega rápida, segura y eficiente de software y servicios de TI que cumplan con los requisitos del negocio y de los usuarios.

Diferencia entre una cultura DevOps y una filosofía DevOps

La cultura DevOps se centra en crear un ambiente de trabajo colaborativo y ágil en el que los equipos de desarrollo y operaciones trabajen juntos para producir software de alta calidad de manera eficiente.

La filosofía DevOps se centra en acelerar el tiempo de comercialización del software, reducir el riesgo y mejorar la calidad, mediante la integración continua, la entrega continua, la automatización y la monitorización.

En el mercado actual, la filosofía DevOps a menudo se refiere a un conjunto de herramientas, tecnologías y prácticas que se utilizan para implementar la cultura DevOps. Estas herramientas pueden incluir herramientas de automatización de infraestructura, herramientas de integración y entrega continuas, y herramientas de monitorización y análisis de datos, entre otras.

Según Diego Quintana en su libro "DevOps, el camino a la agilidad en el desarrollo y la operación de software" (2017), la cultura DevOps se basa en valores como la colaboración, la comunicación, la automatización y la retroalimentación continua, mientras que la filosofía DevOps se enfoca en el proceso integral de desarrollo y despliegue de software, desde la planificación y el diseño, hasta la implementación y la gestión de la infraestructura.

La cultura DevOps y la filosofía DevOps están relacionadas y se apoyan mutuamente, pero no son lo mismo, ambas son complementarias y se requieren mutuamente para el éxito de la implementación de DevOps.

Ciclo de vida DevOps

El objetivo del ciclo de vida DevOps es proporcionar un enfoque eficiente y eficaz para el desarrollo, la entrega y el mantenimiento continuo del software.

Según Juan Manuel Cueva Lovelle en su libro "DevOps. El camino del éxito en la gestión de sistemas y servicios IT" (2019), el ciclo de vida DevOps se compone de las siguientes etapas:

1. Planificación: En esta etapa se define el alcance del proyecto, se establecen los objetivos y se determinan los recursos necesarios para su ejecución.
2. Diseño: En esta etapa se elaboran los diseños técnicos y funcionales del software, teniendo en cuenta los requisitos del negocio y de los usuarios.
3. Implementación: En esta etapa se desarrolla el software, se integran los componentes y se realizan pruebas unitarias para verificar su funcionalidad.
4. Testing: En esta etapa se llevan a cabo pruebas de integración y pruebas de rendimiento para verificar la calidad y el rendimiento del software.
5. Despliegue: En esta etapa se implementa el software en el entorno de producción, asegurando que todo el proceso de despliegue sea automatizado y repetible.
6. Monitorización: En esta etapa se monitorea el rendimiento y la disponibilidad del software en producción, identificando posibles problemas y realizando ajustes y mejoras.

Cada una de estas etapas está diseñada para integrar de manera continua los equipos de desarrollo y operaciones, con el objetivo de garantizar la entrega rápida, segura y eficiente de software y servicios de TI.

El ciclo de vida DevOps es un enfoque iterativo y continuo en el que las diferentes partes del equipo trabajan en estrecha colaboración para garantizar que el software se

desarrolle y se implemente de manera efectiva y eficiente, mientras se mantiene la calidad y la seguridad del software.

Desafíos en la implementación de una cultura DevOps

La implementación de una cultura DevOps puede resultar en una gran cantidad de beneficios para las organizaciones, como una mayor eficiencia en el proceso de desarrollo y despliegue de software, una reducción en los tiempos de lanzamiento al mercado, una mejor calidad del software y una mayor satisfacción del cliente. Sin embargo, también puede presentar algunos desafíos importantes que deben abordarse adecuadamente para lograr una implementación exitosa.

Según Abelardo Medrano y Pablo Valarezo en su artículo "Los desafíos de la implementación de DevOps" (2018), algunos de los principales desafíos en la implementación de una cultura DevOps incluyen:

1. Resistencia al cambio: La implementación de una cultura DevOps implica un cambio significativo en la forma en que las organizaciones desarrollan y despliegan software. Muchas veces, los miembros del equipo pueden resistirse a los cambios en su forma de trabajo, lo que puede dificultar la adopción de DevOps.
2. Falta de colaboración entre los equipos: La colaboración entre los equipos de desarrollo y operaciones es fundamental para el éxito de DevOps. Si los equipos trabajan de forma aislada, sin comunicarse ni compartir información, puede resultar en una falta de integración y colaboración, lo que puede obstaculizar la implementación de DevOps.

3. Dificultades en la automatización: La automatización es una parte fundamental de DevOps, ya que permite la entrega continua de software de alta calidad. Sin embargo, puede ser difícil automatizar ciertos procesos, lo que puede resultar en retrasos y errores en la implementación de DevOps.
4. Falta de estandarización: La falta de estandarización en los procesos de desarrollo y despliegue puede dificultar la implementación de DevOps, ya que puede resultar en procesos diferentes y complejos que requieren un gran esfuerzo para integrarse.
5. Problemas de seguridad: La implementación de una cultura DevOps puede presentar desafíos en términos de seguridad, ya que puede haber un mayor riesgo de vulnerabilidades y ataques en el proceso de entrega continua.

Estos desafíos pueden superarse a través de la comunicación efectiva, la capacitación, educación, automatización de procesos e implementación de prácticas de mejora continua.

Tecnologías necesarias para implementar una cultura DevOps

La implementación exitosa de una cultura DevOps requiere de la utilización de diversas tecnologías que permitan automatizar los procesos de desarrollo, pruebas y despliegue de software. Según el artículo "DevOps y la importancia de las herramientas tecnológicas" de Ramón Sánchez (2019), algunas de las tecnologías más importantes para implementar DevOps son las siguientes:

1. Herramientas de integración continua (CI, por sus siglas en inglés): Estas herramientas permiten la automatización de los procesos de compilación, pruebas

y empaquetado de código, lo que reduce el tiempo y la complejidad de la entrega de software.

2. Herramientas de entrega continua (CD, por sus siglas en inglés): Estas herramientas automatizan el proceso de liberación y despliegue de software en diferentes entornos, lo que permite una entrega rápida y segura de nuevas versiones de software.
3. Herramientas de gestión de configuración: Estas herramientas permiten la gestión y control de versiones del código fuente y la configuración de los diferentes entornos de desarrollo y producción.
4. Herramientas de automatización de pruebas: Estas herramientas permiten la automatización de pruebas de software, lo que mejora la calidad del software y reduce el tiempo necesario para realizar pruebas manuales.
5. Herramientas de monitoreo y análisis: Estas herramientas permiten la supervisión y análisis continuo de los sistemas y aplicaciones, lo que ayuda a detectar y resolver problemas de manera rápida y eficiente.
6. Herramientas de virtualización y contenedores: Estas herramientas permiten la creación y gestión de entornos de desarrollo y producción virtualizados y contenerizados, lo que reduce la complejidad de la gestión de infraestructura y mejora la escalabilidad y la disponibilidad del software.

La implementación de una cultura DevOps requiere de la utilización de diversas tecnologías, como las herramientas de integración continua, de entrega continua, de gestión de configuración, de automatización de pruebas, de monitoreo y análisis, y de virtualización y contenedores. Estas herramientas son esenciales para automatizar los

procesos de desarrollo, pruebas y despliegue de software, y garantizar una entrega rápida, segura y de alta calidad.

Roles y responsabilidades en una cultura DevOps

En una cultura DevOps, es importante definir y establecer claramente los roles y responsabilidades de los equipos de desarrollo y operaciones. Los equipos deben trabajar juntos para lograr los objetivos comunes y compartir la responsabilidad de la entrega de software.

Según el artículo "Roles y responsabilidades en un equipo DevOps" de Sandra López (2018), algunos de los roles y responsabilidades más comunes en una cultura DevOps son los siguientes:

1. Desarrolladores: Los desarrolladores son responsables de escribir el código y las pruebas unitarias para las nuevas funcionalidades y correcciones de errores. Además, deben asegurarse de que el código cumpla con los estándares de calidad y que sea compatible con la infraestructura de la organización.
2. Ingenieros de pruebas: Los ingenieros de pruebas son responsables de diseñar y ejecutar pruebas de software automatizadas y manuales, para garantizar que el software funcione correctamente y cumpla con los requisitos del usuario.
3. Ingenieros de operaciones: Los ingenieros de operaciones son responsables de la infraestructura de la organización y de garantizar que el software se pueda ejecutar de manera eficiente y segura en los diferentes entornos de producción. Además, deben monitorear y analizar los sistemas y aplicaciones para detectar y resolver problemas de manera proactiva.

4. Gerentes de proyecto: Los gerentes de proyecto son responsables de liderar el equipo DevOps y de garantizar que los proyectos se entreguen a tiempo y dentro del presupuesto. Además, deben asegurarse de que se cumplan los requisitos del usuario y de que se mantengan altos estándares de calidad.
5. Especialistas en seguridad: Los especialistas en seguridad son responsables de garantizar que el software cumpla con los estándares de seguridad y que se implementen medidas de seguridad adecuadas en la infraestructura de la organización.

Es importante destacar que estos roles y responsabilidades pueden variar según la organización y el proyecto específico. En una cultura DevOps, es común que los miembros del equipo trabajen juntos y se apoyen mutuamente, lo que puede requerir habilidades y conocimientos técnicos más amplios.

Beneficios de una cultura DevOps

La implementación de una cultura DevOps puede mejorar la satisfacción del cliente, ya que permite una entrega de software más rápida y de mayor calidad. Además, una cultura DevOps puede aumentar la agilidad y flexibilidad de una organización, lo que le permite adaptarse rápidamente a los cambios del mercado y a las demandas de los clientes.

Según el artículo "Los beneficios de la cultura DevOps" de Carlos Díaz (2019), algunos de los beneficios más destacados son los siguientes:

1. Entrega más rápida y frecuente de software: La cultura DevOps se enfoca en la automatización de procesos y la colaboración entre los diferentes equipos de trabajo, lo que permite una entrega de software más rápida y frecuente. Esto a su vez, permite a las organizaciones adaptarse mejor a las necesidades del mercado y del usuario, y mejorar su capacidad de innovación.
2. Mayor calidad del software: La automatización de pruebas y el enfoque en la colaboración entre los equipos de trabajo también pueden mejorar la calidad del software. Al trabajar juntos, los miembros del equipo pueden detectar y solucionar problemas de manera más rápida y eficiente, y garantizar que el software cumpla con los estándares de calidad requeridos.
3. Reducción de costos: La automatización de procesos y la mejora de la eficiencia en la entrega de software pueden reducir los costos de desarrollo y de operaciones.
4. Mayor colaboración y comunicación: La cultura DevOps promueve la colaboración y la comunicación entre los diferentes equipos de trabajo, lo que puede mejorar el ambiente laboral y la productividad de los miembros del equipo. Además, la mejora en la comunicación puede permitir una mejor comprensión de los requisitos del usuario y una entrega de software más acorde a sus necesidades.
5. Mayor seguridad: La cultura DevOps también puede mejorar la seguridad del software al enfocarse en la detección y resolución de problemas de manera proactiva. Además, la implementación de medidas de seguridad adecuadas puede garantizar que el software cumpla con los estándares de seguridad requeridos por las regulaciones y los clientes.

La mejora continua es un principio clave de una cultura DevOps, lo que significa que las organizaciones pueden seguir mejorando y evolucionando con el tiempo.

Integración de DevOps con otros marcos o metodologías

La integración de DevOps con otras metodologías y marcos puede ser una estrategia valiosa para mejorar la eficiencia en el desarrollo de software.

Según el artículo "La integración de DevOps con otros marcos de trabajo" de Ana Belén Gómez (2019), algunas de las metodologías y marcos de trabajo con los que se puede integrar DevOps son los siguientes:

1. Agile: DevOps y Agile comparten una filosofía similar en cuanto a la entrega continua de software y la colaboración entre los diferentes equipos de trabajo. La integración de ambas metodologías puede permitir una entrega de software más rápida y eficiente, así como una mejora en la calidad del software.
2. ITIL: ITIL es un marco de trabajo para la gestión de servicios de tecnología de la información. La integración de DevOps con ITIL puede permitir una mejora en la gestión de los servicios de tecnología de la información, así como una mejor comunicación y colaboración entre los equipos de trabajo.
3. Lean: Lean es una metodología para la mejora continua de procesos. La integración de DevOps con Lean puede permitir una mejora en la eficiencia de los procesos de desarrollo y una reducción de los tiempos de entrega de software.
4. Six Sigma: Six Sigma es una metodología para la mejora de la calidad de los procesos. La integración de DevOps con Six Sigma puede permitir una mejora en la calidad del software y una reducción en los errores y problemas.

Según el artículo "La integración de Kanban en DevOps" de Carlos Sánchez (2019), Kanban es un marco de trabajo ágil que se enfoca en la gestión visual y la mejora continua de los procesos.

La integración de Kanban en DevOps puede permitir una mejora en la gestión del flujo de trabajo y en la coordinación entre los diferentes equipos de trabajo. Algunas de las prácticas de Kanban que se pueden integrar en DevOps son las siguientes:

1. Tableros Kanban: Los tableros Kanban son una herramienta visual que permite gestionar el flujo de trabajo y el estado de las tareas. La integración de tableros Kanban en DevOps puede permitir una mejor visualización del estado de las tareas y una mejor coordinación entre los diferentes equipos de trabajo.
2. Limitación del trabajo en progreso: Kanban enfatiza la importancia de limitar el trabajo en progreso para evitar la congestión en el flujo de trabajo. La integración de esta práctica en DevOps puede permitir una mejor gestión del flujo de trabajo y una reducción en los tiempos de entrega.
3. Mejora continua: Kanban se enfoca en la mejora continua de los procesos. La integración de esta práctica en DevOps puede permitir una mejora constante en la eficiencia y calidad del software entregado.

La integración de DevOps con otros marcos o metodologías puede permitir una mejora en la entrega de software y la eficiencia en el desarrollo

Consideraciones para la integración de DevOps con otros marcos o metodologías

La integración de DevOps con otras metodologías y marcos puede presentar algunas consideraciones, según el artículo "Integrando DevOps con otras metodologías ágiles" de Juan Pérez (2021), algunas de las consideraciones a tener en cuenta son las siguientes:

1. Objetivos claros: Antes de integrar DevOps con otra metodología o marco de trabajo, es importante definir los objetivos que se buscan lograr. De esta forma, se podrá evaluar la compatibilidad entre los objetivos de cada una de ellas y definir una estrategia clara para la integración.
2. Compatibilidad cultural: La integración de DevOps con otra metodología o marco de trabajo también puede tener un impacto en la cultura organizacional. Es importante tener en cuenta la compatibilidad cultural entre ambas metodologías para asegurar una integración efectiva.
3. Proceso de integración gradual: La integración de DevOps con otra metodología o marco de trabajo puede ser un proceso gradual, donde se va incorporando de forma progresiva cada uno de los elementos de la nueva metodología en el proceso de DevOps. Esto permitirá una adaptación más efectiva y una menor resistencia al cambio.
4. Identificación de solapamientos: Es importante identificar los solapamientos entre las metodologías para evitar duplicidades y asegurar una integración efectiva. De esta forma, se podrán aprovechar las fortalezas de cada una de ellas y minimizar los impactos negativos en el proceso de integración.
5. Formación y capacitación: La integración de DevOps con otra metodología o marco de trabajo puede requerir una formación y capacitación específica. Es

importante asegurar que los equipos de trabajo estén preparados para la integración y tengan las habilidades necesarias para trabajar con la nueva metodología.

Es importante tener en cuenta que cada organización es única y que la integración de DevOps con otras metodologías puede requerir un enfoque personalizado. La evaluación de las consideraciones mencionadas anteriormente puede ayudar a asegurar una integración efectiva y minimizar los impactos negativos en el proceso

Gestión de Cambios

La gestión de cambios es el proceso sistemático y planificado para administrar y controlar los cambios en un sistema o proceso, asegurando que los cambios se realicen de manera efectiva y eficiente, y que minimice el impacto negativo en el negocio o en el sistema.

Según Pressman en su libro **“**Ingeniería del software: un enfoque práctico” (2010), la gestión de cambios es una actividad crucial en el desarrollo de software, ya que los cambios en los requisitos o el diseño del software son comunes y pueden afectar significativamente el cronograma y el presupuesto del proyecto si no se manejan adecuadamente. La gestión de cambios implica la identificación, evaluación, aprobación, implementación y seguimiento de los cambios realizados en el software.

La gestión de cambios permite garantizar que los cambios en el software se implementen de manera eficiente y segura, los procesos de la gestión de cambios implican:

1. Planificación de cambios: La planificación de cambios implica la identificación y documentación de los cambios necesarios en el software y los sistemas. La planificación de cambios también debe incluir la definición de los procedimientos de gestión de cambios, la evaluación de riesgos y la identificación de los recursos necesarios para implementar los cambios.
2. Evaluación de cambios: La evaluación de cambios implica la revisión de los cambios propuestos para determinar si son necesarios y si cumplen con los estándares y procedimientos establecidos. La evaluación de cambios también debe incluir la identificación de posibles impactos en el software y los sistemas.
3. Autorización de cambios: La autorización de cambios implica la aprobación de los cambios propuestos antes de que se implementen. La autorización de cambios también debe incluir la definición de los roles y responsabilidades de los equipos de DevOps y los interesados en el proceso de gestión de cambios.
4. Implementación de cambios: La implementación de cambios implica la ejecución de los cambios propuestos y la realización de pruebas para verificar que el software y los sistemas funcionan correctamente después de la implementación del cambio.
5. Monitoreo de cambios: El monitoreo de cambios implica el seguimiento de los cambios implementados para garantizar que funcionen correctamente y que no hayan causado impactos negativos en el software y los sistemas.

Mejora Continua

La mejora continua es un proceso gradual y constante de mejora de la eficiencia, la calidad y la productividad. Se trata de un enfoque sistemático que busca optimizar los procesos y productos existentes, y desarrollar nuevos enfoques y soluciones para mejorar el rendimiento.

La mejora continua implica la identificación de áreas que necesitan mejoras, implementación de cambios, evaluación de los resultados y la realización de ajustes adicionales. Se trata de un proceso iterativo en el que se buscan mejoras continuas a lo largo del tiempo, en lugar de buscar cambios drásticos y repentinos.

La mejora continua es un aspecto clave de una cultura DevOps, la mejora continua implica el proceso de identificar áreas para mejorar y desarrollar estrategias para implementar mejoras. Algunas de las prácticas de mejora continua:

1. Establecimiento de objetivos de mejora continua: Los equipos de DevOps deben establecer objetivos de mejora continua que reflejen las necesidades del negocio y los desafíos que enfrentan en la entrega de software. Los objetivos de mejora continua también deben incluir la medición de los resultados y el establecimiento de indicadores clave de rendimiento (KPI) para monitorear el progreso.
2. Recopilación de datos y análisis: Los equipos de DevOps deben recopilar datos sobre la entrega de software y analizarlos para identificar áreas de mejora. Los datos pueden incluir métricas de rendimiento, como el tiempo de entrega, la calidad del software y la satisfacción del usuario.
3. Identificación de oportunidades de mejora: Los equipos de DevOps deben identificar oportunidades de mejora a través de la recopilación y análisis de datos

y la colaboración con otros equipos en la organización. Las oportunidades de mejora pueden incluir la automatización de procesos, la mejora de la eficiencia de la entrega de software y la mejora de la calidad del software.

4. Implementación de mejoras: Los equipos de DevOps deben implementar mejoras identificadas a través de la colaboración y el análisis de datos. La implementación de mejoras también debe incluir la capacitación de los miembros del equipo y la documentación de los procesos para garantizar la consistencia y la eficiencia.
5. Monitoreo y evaluación: Los equipos de DevOps deben monitorear y evaluar los resultados de las mejoras implementadas para garantizar que se logren los objetivos de mejora continua. Los KPI deben ser monitoreados y revisados regularmente para garantizar que los objetivos de mejora se cumplan.

Según la norma ISO 9000:2015, la mejora continua se define como el "proceso recurrente de mejorar el sistema de gestión de la calidad para alcanzar los objetivos de la organización"

Juego de Lego y Chocolate

El juego de Lego y Chocolate es una simulación diseñada para aclarar el "Por qué" y el "Qué" de DevOps y facilitar el desarrollo de una sólida comprensión fundamental de DevOps Culture. Creado con el método Training from the Back of the Room, utiliza principios de neurociencia cognitiva para maximizar el compromiso y la retención del aprendizaje.

En esta poderosa simulación basada en roles, los participantes experimentan cómo funciona una gran organización antes, durante y después de adoptar la cultura DevOps. Ven los beneficios de mejorar las habilidades, aprenden a eliminar los silos, implementan el "cambio a la izquierda" en seguridad, adoptan el pensamiento sistémico y practican la optimización del flujo de valor desde el negocio hasta el desarrollo y las operaciones de TI.

Según Dana Pylayeva en el libro Introduction to DevOps with chocolate, lego and scrum game (2017), el juego de chocolate y lego está diseñado para involucrar a los cinco sentidos y aprovechar el lado emocional del cerebro. Al trabajar con lego y chocolate, los participantes experimentan las desventajas de la optimización local y aprenden a expandir su vista para incluir todo el sistema. Mediante el uso de avatares, personajes y tarjetas de roles, los participantes obtienen una comprensión de los roles de desarrollo y operaciones, así como sus interdependencias.

En conclusión, el presente marco teórico ha proporcionado una base sólida para comprender los fundamentos y conceptos clave que sustentan la investigación. No obstante, es importante destacar que la amplitud y complejidad del tema abordado requieren de una consideración más detallada. Con el fin de facilitar información adicional, se ha incluido información en el Anexo 1: Terminología Complementaria.

CAPÍTULO III. MARCO METODOLÓGICO

3.1 METODOLOGÍA DE LA INVESTIGACIÓN

El marco metodológico para diseñar una estrategia de una cultura DevOps en una organización implica la definición de los procedimientos, estrategias y operativa necesarios para alcanzar los objetivos específicos y generales establecidos en este proyecto.

3.2 DISEÑO METODOLÓGICO

Se utilizará una metodología basada en dos fases: análisis y diseño. Cada fase incluirá un conjunto de actividades específicas que se detallarán:

3.2.1 ANÁLISIS

En esta fase se llevarán a cabo las siguientes actividades:

- Estudio del contexto actual: Se analizarán los procesos y prácticas actuales de entrega de software, con el fin de identificar los problemas y oportunidades de mejora. También se identificarán los principales actores involucrados en el proceso de entrega de software y se definirán sus roles y responsabilidades.
- Identificación de necesidades y requisitos: Se identificarán los objetivos y requisitos para la creación de una estrategia para la implementación de una cultura DevOps en la organización.

3.2.2 DISEÑO

En esta fase se llevarán a cabo las siguientes actividades:

- Definición de la cultura DevOps: Se definirán los principios y prácticas que conforman la cultura DevOps y se identificarán los procesos que son críticos para la entrega de software.
- Diseño de la estrategia de integración: Se diseñará una estrategia que permita integrar la cultura DevOps con otras metodologías o marcos en la organización, considerando los objetivos y requisitos previamente definidos.
- Definición de roles y responsabilidades: Se definirán claramente los roles y responsabilidades de los actores involucrados en el proceso de entrega de software, con el fin de promover una colaboración efectiva entre los equipos de desarrollo y operaciones.
- Selección de herramientas: Se seleccionarán las herramientas necesarias para gestionar automatizar el proceso de entrega de software y se definirán las integraciones necesarias para que estas herramientas funcionen de manera efectiva.

3.3 TIPO DE DISEÑO DE INVESTIGACIÓN

Se empleará un enfoque cualitativo en la investigación. La elección de este diseño se basa en la necesidad de comprender a profundidad los procesos actuales de entrega de software en la organización, así como en la identificación de sus procesos críticos. También se busca definir y establecer claramente los roles, responsabilidades y herramientas necesarias para implementar con éxito una cultura DevOps en la organización.

3.3.1 ANÁLISIS

En esta fase se llevarán a cabo las siguientes actividades:

- Revisión de la literatura existente: Se realizará una revisión de la literatura relacionada con la implementación de una cultura DevOps en organizaciones similares. Esto ayudará a comprender los conceptos teóricos, los enfoques utilizados y los resultados obtenidos en investigaciones previas.
- Análisis de datos existentes: Si hay datos disponibles relacionados con los procesos y prácticas actuales de entrega de software en la organización, se podrían analizar para identificar patrones, tendencias o áreas de mejora. Esto podría incluir datos de tiempos de entrega, errores frecuentes, satisfacción del cliente, entre otros.

3.3.2 DISEÑO

En esta fase, se llevará a cabo un diseño cualitativo con el fin de comprender a profundidad las percepciones, experiencias y efectos de la implementación de una cultura DevOps en la organización. Además, se busca obtener información detallada sobre los procesos actuales de entrega de software con el propósito de definir y establecer claramente los roles, responsabilidades y herramientas necesarias para lograr de manera exitosa una implementación de la cultura DevOps en la organización.

3.4 TÉCNICA DE INVESTIGACIÓN

En el presente estudio, se utilizarán diversas técnicas de investigación para recolectar y analizar información relacionada con diseñar una estrategia de una cultura DevOps en la organización:

1. Entrevistas individuales: Se llevarán a cabo entrevistas individuales en base al ANEXO 2: GUÍA DE ENTREVISTA INDIVIDUAL con diferentes miembros de la organización, incluyendo a los desarrolladores de software, gerentes de proyectos y otros miembros del equipo. Estas entrevistas permitirán obtener una comprensión profunda de los procesos actuales de entrega de software en la organización, identificar los procesos críticos y las posibles oportunidades de mejora, y también conocer las diferentes perspectivas y opiniones de los miembros del equipo sobre la implementación de una cultura DevOps.

   Posibles Entrevistados:

Tabla 2: Posibles Entrevistados

Nombre	Rol
Colaborador 1	Profesional Especialista UTI
Colaborador 2	Encargado UTI
Colaborador 3	Profesional UTI
Colaborador 4	Profesional UTI

2. Grupos focales: Se llevarán a cabo grupos focales en base al ANEXO 3: GUIA DE GRUPOS FOCALES con los equipos de desarrollo de software y otros miembros del equipo de la organización. Estos grupos focales permitirán obtener una visión más amplia de las diferentes perspectivas y opiniones de los miembros del equipo sobre la implementación de una cultura DevOps, identificar los principales desafíos y obstáculos que podrían surgir durante la implementación a nivel de equipo, y también discutir y evaluar posibles soluciones y estrategias para superar estos desafíos.

Posibles Grupos Focales:

Grupo 1:

Tabla 3: Posibles Grupos Focales grupo 1

Nombre	Rol
Colaborador 1	Profesional Especialista UTI
Colaborador 4	Profesional UTI

Grupo 2:

Tabla 4: Posibles Grupos Focales grupo 2

Nombre	Rol
Colaborador 2	Encargado UTI
Colaborador 3	Profesional UTI

3. Observación participante: Se llevará a cabo observación participante en base al ANEXO 4: FORMATO DE OBSERVACIÓN PARTICIPANTE esto permitirá comprender de manera más profunda los procesos y prácticas de entrega de software en la organización, identificar los problemas y obstáculos que surgen durante la implementación de una cultura DevOps, y también evaluar la efectividad de las soluciones y estrategias propuestas para superar estos problemas.

4. Análisis FODA: Se llevará a cabo un análisis FODA que permitirá evaluar la situación actual y las perspectivas de implementación de una cultura DevOps en la organización. Este análisis proporcionará una visión integral de las fortalezas,

oportunidades, debilidades y amenazas que se presentan, lo cual es fundamental para comprender los factores internos y externos que pueden influir en el éxito de la implementación de una cultura DevOps en la organización. El análisis FODA (Fortalezas, Oportunidades, Debilidades y Amenazas) es una herramienta estratégica que permite evaluar y comprender la situación actual de una organización en relación con un objetivo o proyecto específico. El análisis FODA se vuelve crucial para identificar las necesidades y requisitos fundamentales que se deben abordar para garantizar el éxito de la implementación de una cultura DevOps en la organización.

3.5 PROCESAMIENTO DE DATOS E INFORMACIÓN

En cuanto a la etapa de procesamiento de datos e información, se seguirán los siguientes procedimientos:

1. Se realizará el procesamiento de datos de las entrevistas, grupos focales y observación participante para asegurar una comprensión completa y detallada de los datos obtenidos en base a los Anexos 2, 3 y 4.
2. Se realizará un análisis FODA para comprender las fortalezas, oportunidades, debilidades y amenazas de la organización.
3. Se realizará el proceso de revisión de procesos existentes de desarrollo y entrega de software de la organización.

CAPÍTULO IV. DIAGNOSTICO Y ANALISIS DE RESULTADOS

4.1 DIAGNOSTICO DE LA SITUACION ACTUAL DE DESARROLLO Y ENTREGA DE SOFTWARE DEL IFAM

4.1.1 REVISIÓN DE LOS PROCESOS EXISTENTES

Según los documentos oficiales del Instituto de Fomento y Asesoría Municipal (IFAM), específicamente datados en diciembre de 2020 y elaborados por el equipo de tecnologías de información, se establecieron políticas y procedimientos con el objetivo de asegurar la calidad y la eficiencia en la entrega software. A continuación, se detallan las políticas y procedimientos utilizados:

Tabla 5: Políticas y Procedimientos

Id	Nombre	Política	Procedimiento
1	Política de desarrollo de sistemas de información	x	
2	Política de calidad de software	x	
3	Política de control de versiones	x	
4	Procedimiento para la Gestión de Desarrollo Sistemas de Información IFAM-DE-UTI-GDSI		x

1. Política de desarrollo de sistemas de información:

1.1. Objetivo: Buscar un mejor rendimiento en el área de desarrollo de software de la Unidad de Tecnologías de Información, se plantea el proyecto de estandarización de la arquitectura tecnológica de desarrollo de la unidad, buscando la creación de aplicaciones en un mismo lenguaje de programación, componentes, transacciones, etc.

1.2. Alcance: Lograr con ASP.net en NET CORE utilizar el modelo, vista, controlador el cual puede incorporar tecnologías como HTML, CCS, JS y JQUERY, luego estos sitios web totalmente responsivos se puedan complementar en aplicaciones móviles gracias a Xamarin como contenedor web. La unidad, dada la gran cantidad de desarrollos que tiene año a año, tanto a nivel municipal como institucional, debe estandarizar sus desarrollos, para capacitar a sus funcionarios en una misma línea y generar productos de primera calidad potenciando más conocimiento de la herramienta.

1.3. Estándar tecnológico: Los sistemas deberán ser desarrollados en lenguaje C#, utilizando .NET core más reciente. Las tecnologías para utilizar dentro de cada nuevo sistema están basadas en herramientas que pueden asociarse dentro de lenguajes de programación .NET, de las cuales se mencionan las siguientes:

   a. ASP.NET Core
   b. Javascript
   c. JQuery
   d. HTML5
   e. CSS3
   f. Bootstrap Templates Responsive

g. Transact-SQL en base de datos SQL Server
h. Visual Studio
i. Team Foundation Services

1.4. Arquitectura general: Se utilizan 3 servidores debidamente preparados donde los desarrolladores y el equipo del proyecto podrán realizar lo requerido, cada proyecto finalizado deberá pasar ciertas etapas para ponerse en marcha, de las cuales se plantea:

a. Desarrollo: Los desarrolladores realizarán el sistema basado en los requerimientos previamente suministrados por el Project Manager del proyecto. El Arquitecto de Soluciones revisará el producto o etapa verificando que cumpla con el estándar tecnológico de la unidad.
b. Certificación: El Project Manager y el arquitecto de soluciones revisarán con los encargados del requerimiento que cumpla con lo requerido, en esta etapa se certifica el desarrollo.
c. Producción: El nuevo sistema o funcionalidad podrá ser utilizado por el usuario final.

1.5. Diseño general:

a. Client Browser: Es el navegador web mediante el cual el usuario final podrá acceder al sistema, la aplicación publicada por el área de infraestructura tecnológica asignará toda la configuración necesaria para un acceso seguro a la aplicación.
b. FrontEnd: Es el componente visual y de interacción del usuario final, su principal función es el despliegue de la información y como interfaz para

la interacción del usuario con la aplicación. Dicho componente es el encargado de interactuar con el BackEnd, enviando solicitudes de manipulación de DATA y recibiendo respuestas de este para mostrarlas al usuario en el navegador.

c. BackEnd: Es el componente encargado de gestionar y hacer cumplir las reglas de negocio sobre la manipulación de los datos. Además, contiene lo necesario para el acceso a la Base de Datos. Otra de sus responsabilidades es consumir los WebServices externos. Dicho componente interactúa con el FrontEnd recibiendo solicitudes que el Backend, valida, procesa y genera una respuesta correspondiente.
d. Base de datos: Es el componente encargado de almacenar los datos necesarios para el correcto funcionamiento del sistema.
e. WebAPI: Su función principal es la de ser interfaz de programación de la aplicación. Contiene el conjunto de rutinas que proveen acceso a las funciones de la aplicación.

1.6. Análisis o Toma de Requerimientos: Es el paso inicial de cualquier desarrollo de software, para una detallada toma de requerimientos se propone el siguiente estándar:

a. Inicio del Proyecto
b. Recopilación y Aprobación de Requerimientos
c. Control de Cambios
d. Identificación de Riesgos
e. Seguimiento

f. Entrega

g. Diseño

h. Desarrollo

2. Política de calidad de software:

2.1. Objetivo: Proporcionar a los usuarios de los diferentes sistemas de información sistemas totalmente depurados y cumpliendo altos estándares de calidad.

2.2. Alcance: El propósito de esta política es brindar una metodología de procesos unificada, que establezca los principios básicos para la coordinación en la revisión o testing de las aplicaciones. Las disposiciones técnicas contenidas en este instrumento son de acatamiento obligatorio por parte de la unidad de tecnologías de información. Para que la normativa y estandarización sean efectivas, resulta necesario el cumplimiento de los estándares, la metodología definida y él envío de la información específica que el órgano competente solicite.

2.3. Control de calidad:

a. Inspección de la calidad: Se revisará que el producto cumpla con la arquitectura de desarrollo de la UTI, cumpliendo los principios de lenguaje de programación, lógica transaccional, etc.

b. Testing: El arquitecto de soluciones deberá revisar el funcionamiento del producto, revisando que cumpla con la toma de requerimientos solicitada por el usuario, usabilidad del producto, consumo de servicios web e interacción con la base de datos correspondiente. El arquitecto podrá realizar alguno de los 3 tipos de prueba:

1. Según si ejecuta o no el código
2. Según el uso de herramientas
3. Según lo que verifican

c. Aseguramiento de la calidad: El arquitecto de soluciones realiza el control de calidad con el usuario responsable del requerimiento, esto con el fin de que el mismo cumpla con lo solicitado por el usuario.
d. Aprobación: Se aprueba el desarrollo por parte de la UTI y usuario encargado del requerimiento.

3. Política de control de versiones:

3.1. Objetivo: Proporcionar a los usuarios de los diferentes sistemas de información, la seguridad de la no afectación cuando se realiza la implementación de una nueva versión del sistema.

3.2. Alcance: El propósito de esta política es brindar una metodología de procesos unificada, que establezca los principios básicos para la coordinación de las nuevas versiones de las aplicaciones. Las disposiciones técnicas contenidas en este instrumento son de acatamiento obligatorio por parte de la unidad de tecnologías de información. Para que la normativa y estandarización sean efectivas, resulta necesario el cumplimiento de los estándares, la metodología definida y él envío de la información específica que el órgano competente solicite.

3.3. Control de versiones: Para el manejo de versiones, cuando un desarrollador o equipo de desarrollo requiera realizar un cambio, se copiará al servidor de

desarrollo correspondiente la última versión basado en la fecha del servidor de control de versiones:

a. Donde deberá cumplir la siguiente subestructura:

    1. • Fuentes -> FECHAVERSION
    2. • Ejecutables -> FECHAVERSION
    3. • Scripts de base de datos -> FECHAVERSION
    4. • Manuales

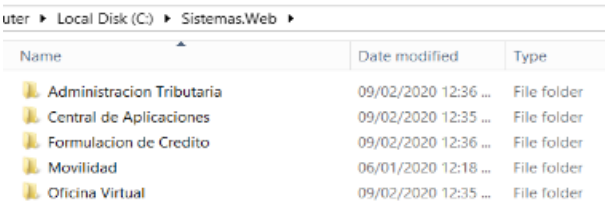
b. 

Ilustración 1: Política de control de versiones parte 1

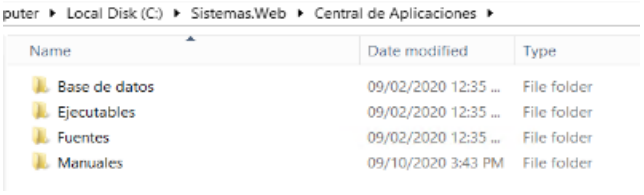
c. 

Ilustración 2: Política de control de versiones parte 2

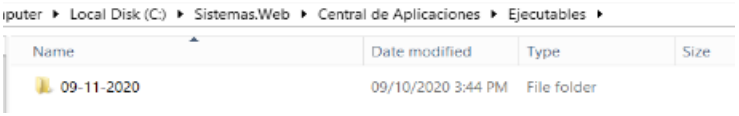
d. 

Ilustración 3: Política de control de versiones parte 3

3.4. Planificación: El arquitecto de soluciones realiza la planificación correspondiente, categorizando la nueva versión como:

a. Mayor: Donde la nueva versión tiene un alto impacto al sistema o es una nueva funcionalidad del sistema.

b. Menor: Donde la nueva versión tiene un bajo impacto al sistema, contiene corrección de errores anteriores o mejoras mínimas del sistema.

c. Emergencia: Donde la nueva versión tiene una funcionalidad que es de suma importancia para la toma de decisiones institucional o contiene una corrección de un error significativo que pueda poner en riesgo la funcional del sistema y seguridad de la UTI.

3.5. Verificación del cumplimiento de estándares: El arquitecto de soluciones revisará que el producto cumpla con la arquitectura de desarrollo de la última versión suministrada.

3.6. Verificación de cumplimiento de pruebas: El arquitecto de soluciones realiza la verificación del proceso de Control de calidad del servicio y el mismo se encuentre aprobado de manera satisfactoria.

3.7. Respaldo versión actual: El arquitecto de soluciones deberá realizar un respaldo de la versión actual en el servidor de producción, la cual tendrá a disposición en caso de presentarse alguna situación y deba realizar un ROLLBACK de la aplicación. En caso de no presentarse afectación podrá eliminar la versión respaldada, ya que la misma se encontrará en el servidor de control de versiones con la fecha de implementación.

3.8. Implementación: El arquitecto de soluciones deberá realizar la implementación de la nueva versión del sistema en el servidor de producción.

4. Procedimiento para la Gestión de Desarrollo Sistemas de Información IFAM-DE-UTI-GDSI:

4.1. Objetivo: Realizar una coordinación sobre el desarrollo de sistemas a nivel de la unidad de tecnologías de información.

4.2. Alcance: Desarrollo o mejora del sistema de información

4.3. Alineamiento con el plan estratégico institucional: El estándar de desarrollo de sistemas de información, permitirá al equipo de trabajo llevar un orden de la asignación de requerimientos, generando de una misma manera sistemas de información requeridos para el cumplimiento de metas institucionales.

4.4. Diagramas de Flujo (Anexo 5: Procedimientos Institucionales).

4.1.2 RESULTADOS DE ENTREVISTAS, GRUPOS FOCALES Y OBSERVACIÓN PARTICIPANTE

Se llevaron a cabo entrevistas individuales (Anexo 6: Resultados guía de entrevista individual), grupos focales (Anexo 7: Resultados guía de grupos focales), y observación participante (Anexo 8: Resultados formato de observación participante), con un total de cinco participantes según anexo correspondiente. Estas actividades se llevaron a cabo con los miembros del equipo de desarrollo de software y operaciones del IFAM. Las sesiones se realizaron a lo largo de una semana, realizando las preguntas diseñadas para cada tipo de entrevista, tal como se establece en el punto 1.3 de la sección de técnicas de investigación del capítulo IV del marco metodológico de este documento.

4.1.3 ANÁLISIS DE RESULTADOS DE ENTREVISTAS, GRUPOS FOCALES Y OBSERVACIÓN PARTICIPANTE

A partir de los resultados obtenidos en un departamento conformado por 9 personas, se llevó a cabo una muestra con un tamaño de 4 personas claves en el proceso de entrega de software, lo cual permitió identificar lo siguiente:

El análisis de resultados de las entrevistas individuales (Anexo 9: Análisis de resultados guía de entrevista individual), revela una serie de aspectos importantes, existen desafíos en la colaboración entre equipos y falta de información de metodologías como DevOps. Se identifican oportunidades de mejora en la comunicación, control de versiones, automatización de pruebas y asignaciones.

Los colaboradores reconocen los beneficios potenciales de una cultura DevOps, para mejorar la colaboración y la calidad. Se sugieren temas como la capacitación y el establecimiento de equipos para poder llevar a cabo la implementación de una cultura DevOps.

Dentro de los desafíos se incluyen la resistencia al cambio y la necesidad de adquirir nuevas habilidades técnicas.

Estos hallazgos proporcionan información valiosa para la implementación de una cultura DevOps en la organización, destacando las áreas de mejora y los obstáculos potenciales a considerar.

El análisis de resultados de los grupos focales (Anexo 10: Análisis de resultados guía de grupos focales), revela desafíos comunes en la implementación de una cultura DevOps, como la resistencia al cambio, la falta de comunicación y colaboración, la falta de claridad en roles y responsabilidades, y la gestión del tiempo y los recursos. Las soluciones propuestas se centran en la capacitación, la comunicación abierta, la

participación de los miembros del equipo en la toma de decisiones y la asignación adecuada de recursos.

El análisis de resultados de la observación participante (Anexo 11: Análisis de resultados formato de observación participante), revela los obstáculos y problemas específicos que afectan la implementación de una cultura DevOps en la organización. Estos hallazgos brindan una comprensión clara de las áreas que requieren atención y mejoras para lograr una implementación exitosa de una cultura DevOps.

4.1.4 ANÁLISIS DE FORTALEZAS, OPORTUNIDADES, DEBILIDADES Y AMENAZAS

El análisis FODA en relación con la implementación de una cultura DevOps en la organización se realizó en base a la recopilación de información y análisis de resultados provenientes de entrevistas individuales, grupos focales, observación participante y PEI Institucional del IFAM:

Fortalezas:

- Algunos sistemas de información cuentan con un control de versiones implementado en Azure DevOps.
- Disposición del IFAM para invertir en nuevas tecnologías y capacitación al personal.
- IFAM dentro de su Plan estratégico institucional 2021-2030 cuenta con el objetivo estratégico: “Desarrollar un ecosistema de soluciones e infraestructura tecnológica que facilite la transformación digital del Régimen Municipal”.

Oportunidades:

- Permite mejorar la eficiencia y calidad de la entrega de software.
- Permite establecer canales de comunicación efectivos y promover una cultura de responsabilidad compartida.
- Permite mejorar la calidad del software.
- La capacitación en mejores prácticas y herramientas relacionadas con DevOps puede mejorar el rendimiento del equipo.
- Permite mejorar los procesos de entrega de software en la organización.
- Fomenta la colaboración entre equipos de desarrollo y operaciones.

Debilidades:

- Falta de estandarización en los procedimientos de gestión de versiones y control de código fuente.
- Limitada implementación de la integración continua y el despliegue continuo.
- Escasa automatización de pruebas y dependencia de pruebas manuales.
- Falta de claridad en los roles y responsabilidades dentro de los equipos.
- La resistencia al cambio por parte de algunos miembros del equipo.
- La falta de conciencia sobre los beneficios y el valor de una cultura DevOps.

Amenazas:

- Falta de asignación de recursos presupuestarios para la adquisición de herramientas y capacitaciones para implementación de una cultura DevOps.

- Nuevas políticas de gobierno por parte de la Contraloría General de la República o el Ministerio de Ciencia, Innovación, Tecnología y Telecomunicaciones que soliciten modificar el modelo de entrega de software de las organizaciones gubernamentales.

El objetivo principal de este análisis FODA es proporcionar una visión clara y completa de los factores internos y externos que pueden influir en la implementación de una cultura DevOps. Lo cual permitirá identificar áreas clave que requieren atención y enfoque, así como también generar estrategias y acciones específicas para aprovechar las fortalezas y oportunidades, y superar las debilidades y amenazas identificadas.

4.1.5 ANÁLISIS DE LAS NECESIDADES Y REQUISITOS DEL IFAM PARA LA IMPLEMENTACIÓN DE UNA CULTURA DEVOPS

Las necesidades y requisitos identificados se realizan en base al análisis realizado a las entrevistas, grupos focales, observación participante y FODA:

Tabla 6: Necesidades y Requisitos

Necesidad / Requisito	Relación con el análisis de resultados
Estandarización de procesos	En base a los puntos 1.1 Revisión de los procesos existentes, 1.3 Análisis de resultados de entrevistas, grupos focales y observación participante y 1.4 Análisis de fortalezas, oportunidades, debilidades y amenazas, se identifica la estandarización de procesos como un requisito

	para garantizar la eficiencia y calidad en la entrega de software.
Automatización de procesos	En base a los puntos 1.1 Revisión de los procesos existentes, 1.3 Análisis de resultados de entrevistas, grupos focales y observación participante y 1.4 Análisis de fortalezas, oportunidades, debilidades y amenazas, se identifica la automatización de procesos como un requisito para agilizar y optimizar la entrega de software, ya que al automatizar tareas repetitivas se puede mejorar la eficiencia, calidad y tiempos de entrega.
Enfoque en la calidad del software	En base a los puntos 1.1 Revisión de los procesos existentes, 1.3 Análisis de resultados de entrevistas, grupos focales y observación participante y 1.4 Análisis de fortalezas, oportunidades, debilidades y amenazas, se identifica el enfoque en la calidad de software como una necesidad para garantizar que los productos sean confiables y así satisfacer las necesidades de los usuarios.
Colaboración efectiva entre equipos	En base a los puntos 1.1 Revisión de los procesos existentes, 1.3 Análisis de resultados de entrevistas, grupos focales y observación participante y 1.4 Análisis de fortalezas, oportunidades, debilidades y amenazas, se identifica la colaboración efectiva entre equipos como un requisito para lograr la entrega de software de manera

	eficiente ya que, al fomentar la comunicación, el trabajo en equipo y responsabilidades ayudará a facilitar la implementación de una cultura DevOps.
Capacitación	En base a los puntos 1.1 Revisión de los procesos existentes, 1.3 Análisis de resultados de entrevistas, grupos focales y observación participante y 1.4 Análisis de fortalezas, oportunidades, debilidades y amenazas, se identifica la capacitación como una necesidad para mejorar las habilidades y conocimientos del equipo en relación con la entrega de software y así facilitar la adopción exitosa de una cultura DevOps.

Es importante detallar que los requisitos son elementos esenciales y obligatorios para lograr un objetivo o necesidad, mientras que las necesidades son aspectos importantes y deseables para alcanzar un objetivo o mejorar una situación existente.

Los requisitos y necesidades identificados están justificados por la importancia de mejorar la eficiencia operativa, asegurar la calidad del software y fomentar la colaboración entre equipos, Ambos desempeñan un papel fundamental en la implementación exitosa de una cultura DevOps:

Tabla 7: Necesidades y Requisitos vs Análisis

Necesidad / Requisito	**Análisis**

Estandarización de procesos	1. Es fundamental establecer procedimientos estandarizados para la gestión de versiones (control de código fuente) en todos los sistemas de información de la organización. 2. La falta de estandarización en estos procesos puede generar confusión, retrasos y dificultades en la colaboración entre equipos.
Automatización de procesos	1. Se requiere una mayor implementación de la integración continua y el despliegue continuo (CI/CD) mediante la adopción de herramientas y tecnologías que permitan la automatización de los procesos. 2. La automatización de pipelines de CI/CD agilizará la entrega de software y reducirá la dependencia de procesos manuales, mejorando la eficiencia y la calidad del desarrollo y despliegue de aplicaciones.
Enfoque en la calidad del software	1. Existe una necesidad de priorizar la automatización de pruebas y mejorar la calidad del software entregado. 2. La implementación de estrategias de pruebas automatizadas, como pruebas unitarias, de integración y de regresión, garantizará una

	cobertura adecuada y reducirá la dependencia de pruebas manuales. 3. La incorporación de herramientas de análisis estático de código y pruebas de rendimiento contribuirá a mejorar la calidad del software en términos de rendimiento, seguridad y mantenibilidad.
Colaboración efectiva entre equipos	1. Es esencial fomentar una colaboración efectiva entre los equipos de desarrollo y operaciones. 2. Se deben establecer canales de comunicación claros y promover una cultura de responsabilidad compartida en la entrega de software. 3. La falta de comunicación y colaboración entre los equipos puede generar retrasos, conflictos y dificultades en la implementación de una cultura DevOps.
Capacitación	1. Es necesario brindar capacitación y apoyo continuo a los miembros del equipo en las mejores prácticas y herramientas relacionadas con DevOps. 2. La capacitación permitirá adquirir las habilidades necesarias para adoptar las prácticas y tecnologías de DevOps de manera efectiva.

3. El apoyo continuo garantizará la comprensión y el compromiso del equipo en la implementación exitosa de una cultura DevOps.

El análisis de resultados de las necesidades y requisitos del IFAM revela que para implementar una cultura DevOps de manera exitosa, se requiere de manera obligatoria la estandarización de procesos, la automatización de procesos y la colaboración efectiva entre equipos, y de manera importante el enfoque en la calidad del software y la capacitación. Donde estos requisitos y necesidades deben abordarse de manera integral en la organización.

En conclusión, la información recopilada durante este capítulo proporciona una base sólida del proceso de investigación. Sin embargo, en el Anexo 12: Literatura Complementaria, se encuentra una recopilación de la literatura adicional que respalda y contextualiza los hallazgos obtenidos.

CAPÍTULO V. SOLUCIÓN DEL PROBLEMA

5.1 PROPUESTA DE SOLUCIÓN

5.1.1 CULTURA ORGANIZACIONAL

La cultura organizacional actual de la Unidad de Tecnologías de Información del IFAM revela varios obstáculos que pueden surgir durante la implementación de una cultura DevOps. Este proceso se realizó en base al análisis de resultados de entrevistas, grupos focales y observación participante del capítulo IV diagnóstico y análisis de resultados, lo cual permitió identificar los obstáculos presentados.

Estos obstáculos son categorizados como comunes en organizaciones que buscan adoptar DevOps y requieren atención para garantizar una adopción exitosa:

Tabla 8: Obstáculos y Estrategia de mitigación

Obstáculos Identificados	Estrategia de Mitigación
Resistencia al cambio: La resistencia al cambio es un desafío común cuando se introduce una nueva cultura organizacional. Los empleados pueden sentirse inseguros o incómodos con los cambios propuestos.	**Capacitación y formación:** Proporcionar capacitación y programas de formación sobre una cultura DevOps para todo el personal puede ayudar a reducir la resistencia al cambio. Los empleados deben comprender los beneficios de DevOps y cómo afectará positivamente sus roles.
Falta de comunicación y colaboración: La falta de	**Comunicación:** Fomentar una comunicación abierta y transparente como

comunicación y colaboración entre equipos puede obstaculizar la implementación de una cultura DevOps.	parte esencial del proceso. Esto incluye la creación de canales de comunicación efectivos y reuniones regulares entre los equipos. **Colaboración:** Impulsar una colaboración efectiva es esencial para lograr la implementación de una cultura DevOps. Establecer objetivos compartidos motivará al equipo a colaborar para lograr completar metas comunes.
Falta de claridad en roles y responsabilidades: La falta de definición clara de los roles y responsabilidades puede generar confusión y conflictos en la organización.	**Participación de los miembros del equipo:** Involucrar a los miembros del equipo en la toma de decisiones puede ayudar a definir roles y responsabilidades de manera más clara.

Para poder alcanzar los objetivos específicos de la implementación de una cultura DevOps en el Instituto de Fomento y Asesoría Municipal, y en base a la estrategia de mitigación identificada, se detalla cómo se lograrán alcanzar estos objetivos:

Tabla 9: Estrategia de mitigación e Instrumentos

Objetivo Especifico	Estrategia de Mitigación			
	Capacitación y formación	*Comunicación*	*Colaboración*	*Participación de los miembros del equipo*
Definir un instrumento que permita adoptar una cultura DevOps de manera efectiva y fomentar la colaboración y comunicación entre los equipos	Esto permitirá mejorar el ambiente laboral del equipo sobre temas donde el equipo comparte el conocimiento adquirido. Se aplicará el instrumento denominado *Capacitación*.	Esto permite promover la transparencia y la visibilidad de las actividades y proyectos. Se aplicará el instrumento denominado *Tablero Kanban*.	Esto permite promover la colaboración de manera proactiva y efectiva entre los equipos dentro de la organización. Se aplicará el instrumento denominado *Juego de Lego y Chocolate*.	Esto fomenta la interacción entre los equipos. Se aplicará el instrumento denominado *Retroalimentación*.

La implementación de una cultura DevOps en la Unidad de Tecnologías de Información del IFAM es un desafío factible, pero requiere una estrategia clara para abordar los obstáculos identificados.

Es importante mencionar que los colaboradores reconocen los beneficios potenciales de una cultura DevOps, lo que demuestra un alto porcentaje de éxito. La estrategia permitirá superar los obstáculos y realizar una transición más ligera hacia una cultura DevOps, mejorando la colaboración, la calidad y la eficiencia en la Unidad de Tecnologías de Información del IFAM.

5.2 DESARROLLO DE LA SOLUCIÓN

5.2.1 DISEÑO DE UNA ESTRATEGIA PARA LA IMPLEMENTACIÓN DE UNA CULTURA DEVOPS

El diseño de instrumentos desempeñará un papel fundamental en el cumplimiento de las estrategias de mitigación destinadas a lograr el cumplimiento del objetivo específico de establecer una herramienta que facilite adoptar una cultura DevOps de manera efectiva y fomentar la colaboración y comunicación entre los equipos.

5.2.1.1 INSTRUMENTO CAPACITACIÓN

Diseñar un instrumento de capacitación proporcionara una guía integral para capacitar a los profesionales de tecnologías de información de IFAM, en los principios y prácticas fundamentales de DevOps.

Objetivo:

Proporcionar una estructura para llevar a cabo capacitaciones efectivas de una cultura DevOps, con el fin de promover la colaboración y comunicación entre equipos.

Materiales necesarios:

- Material didáctico actualizado
- Instructor
- Modalidad (Presencia / Virtual)

Pasos del instrumento:

1. Preparación:
    - Seleccionar o contratar al instructor con experiencia en cultura DevOps.
    - Definir la duración y la frecuencia de la capacitación.
    - El instructor debe crear el material de capacitación, el cual debe incluir presentaciones y documentación para los participantes.
    - El instructor debe preparar un entorno físico o virtual si la capacitación se llevará a cabo en línea.

2. Invitación y registro:
    - Invitar a los participantes del grupo de Tecnologías de Información del IFAM.
    - El instructor debe contar con un sistema de registro para obtener información sobre los participantes y su asistencia.

3. Contenido:
    - Introducción a la Cultura DevOps:
        - ✓ ¿Qué es DevOps?
        - ✓ Principios de DevOps
    - Prácticas y Herramientas de DevOps.
    - Cultura DevOps:
        - ✓ Cultura DevOps
        - ✓ Casos de Éxito
    - Kanban:
        - ✓ Introducción a Kanban
        - ✓ Principios de Kanban
        - ✓ Tablero Kanban

4. Evaluación:

El instructor debe realizar una evaluación final tipo quiz para evaluar la comprensión y retención de los conceptos.

Participantes:

Miembros del grupo de Tecnologías de Información del IFAM.

Resultados esperados:

1. Comprendan los principios y valores de DevOps.

2. Mentalidad para colaborar y comunicarse de manera efectiva entre equipos.
3. Conocimiento de las prácticas y herramientas clave de DevOps.
4. Motivación para implementar una cultura DevOps en su lugar de trabajo.

Este instrumento proporciona una base sólida para una capacitación introductoria que permita la adopción de una cultura DevOps en el grupo de Tecnologías de Información del IFAM.

5.2.1.2 INSTRUMENTO TABLERO KANBAN

Diseñar un instrumento tipo tablero Kanban proporcionara una herramienta esencial para visualizar y gestionar el flujo de trabajo, optimizando la comunicación y permitiendo una toma de decisiones más informada a los profesionales de tecnologías de información de IFAM.

Objetivo:

Mejorar la comunicación en un entorno DevOps al visualizar y gestionar el flujo de trabajo en ejecución.

Materiales necesarios:

- Pizarra o tablero en blanco (físico o digital).
- Tarjetas adhesivas y Marcadores (si es físico).
- Acceso a herramientas de gestión de proyectos como Trello, Jira, Azure DevOps, entre otros (si es digital).

Pasos del instrumento:

1. Definir las columnas del tablero: Diseñar las columnas que representarán las etapas. Propuesta: "Backlog", "En desarrollo", "Pruebas", "Aprobado", "Despliegue" y "En producción".

2. Selección del proyecto: Seleccionar un proyecto en el cual se simulará el proceso.

3. Crear tarjetas: Cada tarea o elemento de trabajo debe representarse mediante una tarjeta. Las tarjetas deben contener información clave, como la descripción de la tarea, el responsable, la prioridad y los plazos.

4. Asignar tareas a participantes: Identificar quién es responsable de cada tarea. Esto fomenta la responsabilidad individual.

5. Configurar el tablero Kanban: Crear y configurar el tablero Kanban utilizando software o herramientas adecuadas.

6. Mover las tarjetas a través del tablero: A medida que las tareas avancen en el proceso DevOps, deben moverse de una columna a otra. Esto debe hacerse de manera colaborativa.

7. Revisar y actualizar: Realizar reuniones periódicas de seguimiento para revisar el tablero Kanban y actualizar el estado de las tareas, esto fomenta la comunicación entre los miembros del equipo de trabajo.

Participantes:

Miembros del grupo de Tecnologías de Información del IFAM.

Resultados esperados:

1. Mejora de la visibilidad: Todos los equipos tienen una visión clara del progreso y los problemas en el proceso DevOps.
2. Mayor colaboración: Los equipos trabajan juntos para resolver problemas y eliminar cuellos de botella de manera más eficiente.
3. Mayor comunicación: Las actualizaciones en tiempo real y las reuniones de seguimiento fomentan la comunicación efectiva entre los equipos.
4. Reducción de tiempos de entrega: Al identificar y abordar rápidamente los problemas, se aceleran los ciclos de desarrollo y entrega.
5. Mejora de la calidad: La gestión visual y la revisión continua conducen a una mayor calidad del producto y procesos más eficientes.

Este instrumento ayudará a identificar rápidamente cuellos de botella, resolver problemas y acelerar los tiempos de entrega, un tablero Kanban bien implementado se convierte en un motor para el éxito en la búsqueda de la comunicación, eficiencia y adopción de una cultura DevOps para el grupo de Tecnologías de Información del IFAM.

5.2.1.3 INSTRUMENTO JUEGO DE LEGO Y CHOCOLATE

Diseñar un instrumento con el juego de LEGO y el chocolate contribuirá a fomentar la colaboración y el trabajo en equipo dentro del grupo de tecnologías de información del IFAM, promoviendo una mentalidad ágil, la mejora continua y aprendizaje compartido esenciales de una cultura DevOps. Esto puede ser una forma efectiva y divertida de promover los valores esenciales de la cultura DevOps:

Objetivo:

Fomentar una mentalidad de comunicación, colaboración, mejora continua y el aprendizaje compartido dentro del grupo de Tecnologías de Información del IFAM.

Materiales necesarios:

- Sets de LEGO.
- Chocolate como recompensa.
- Espacio adecuado para realizar actividades.

Pasos del instrumento:

1. Sesión de repaso inicial: Breve sesión de repaso sobre los principios fundamentales de una cultura DevOps y la importancia de la colaboración, la comunicación y el trabajo en equipo.

2. Actividad de construcción de LEGO: Dividir a los participantes en equipos pequeños (mixtos de diferentes áreas). Proporcionar a cada equipo un set de LEGO y asignar una tarea específica de construcción (La tarea debe ser un

desafío que requiera colaboración y comunicación efectiva para completarla con éxito).

3. Facilitación y observación: Mientras los equipos trabajan en sus proyectos de LEGO, se deberá actuar como facilitador y observador. Observar cómo se comunican y colaboran los miembros del equipo.

4. Discusión y reflexión: Después de que todos los equipos hayan completado sus construcciones de LEGO, se debe organizar una sesión de discusión. Preguntar a los participantes sobre sus experiencias. Animar a compartir la experiencia de cómo se sintieron trabajando en equipo y cuáles fueron los desafíos que enfrentaron. Destacar la importancia de aprender de los errores y la necesidad de comunicación efectiva.

5. Recompensas: Como recompensa por la colaboración y esfuerzo en la actividad, proporcionar chocolate a todos los participantes. Esto no solo como una recompensa, sino que también enfatiza la importancia de celebrar los logros en equipo.

6. Plan de acción: Invitar a los participantes a discutir cómo pueden aplicar las lecciones aprendidas en la actividad de LEGO a su trabajo diario en Tecnologías de Información.

Participantes:

Miembros del grupo de Tecnologías de Información del IFAM.

Resultados esperados:

1. Mejora en la comunicación: Los participantes deberían experimentar una mejora en la comunicación entre los miembros de sus equipos. Deben sentirse más cómodos compartiendo ideas y expresando sus opiniones.
2. Colaboración efectiva: Se espera que los equipos muestren una mayor capacidad para colaborar de manera efectiva. Deben aprender a aprovechar las fortalezas individuales y trabajar juntos para superar desafíos.
3. Conciencia de la importancia de la cultura DevOps: Los participantes deben comprender la importancia de una cultura DevOps y cómo esta contribuye a la eficiencia y calidad de los productos que brindan.
4. Aprendizaje de la retroalimentación: Los participantes deben estar dispuestos a aprender de los errores y desafíos que enfrentaron durante la actividad de LEGO. Deben comprender que los errores son oportunidades de mejora.
5. Compromiso con la mejora continua: Los participantes deben estar motivados para aplicar lo aprendido en su trabajo diario. Deben ser conscientes de la necesidad de buscar constantemente formas de mejorar los procesos y la colaboración en su equipo.
6. Reconocimiento de logros en equipo: Se espera que los participantes valoren la importancia de celebrar los logros en equipo y reconozcan que el éxito no solo depende de individuos, sino de la colaboración de todo el grupo.

7. Plan de acción concreto: El encargado de la unidad de tecnologías de información en conjunto con sus colaboradores podrá generar un plan de acción que incluya medidas específicas para implementar una cultura DevOps en su entorno de trabajo. Esto podría incluir cambios en los procesos, la implementación de herramientas DevOps y roles específicos dentro de su grupo de colaboradores.

La combinación de la actividad práctica, la comunicación efectiva y la participación de los miembros del equipo crea un entorno propicio para la adopción de una cultura DevOps en el grupo de Tecnologías de Información del IFAM.

5.2.1.4 INSTRUMENTO RETROALIMENTACIÓN

Diseñar un instrumento de retroalimentación es crucial para mejorar continuamente las prácticas DevOps en una organización, ya que la retroalimentación constante y la mejora continua son fundamentales para el éxito de DevOps.

Objetivo:

Evaluar y mejorar la cultura DevOps en una organización mediante la recopilación de datos y la retroalimentación de los participantes.

Materiales necesarios:

- Cuestionario o encuesta de retroalimentación.
- Disponibilidad para reuniones individuales o grupales

Pasos del instrumento:

1. Preparación:

- Diseñar un cuestionario de retroalimentación que aborde aspectos clave de la cultura DevOps.
- Seleccionar a los participantes que proporcionarán retroalimentación.

2. Recopilación de datos:

- Distribuir el cuestionario a los participantes seleccionados.
- Proporcionar instrucciones claras sobre cómo completar el cuestionario y establecer una fecha límite para la entrega de respuestas.

3. Análisis de datos:

- Recopilar y analizar las respuestas del cuestionario para identificar patrones y áreas de mejora en la cultura DevOps.

4. Reunión de retroalimentación:

- Organizar reuniones individuales o grupales con los participantes para discutir los resultados y obtener una comprensión de los mismos.
- Fomentar la discusión abierta entre los equipos.

5. Plan de acción:

- Identificar las áreas específicas que requieran mejora.

- Establecer objetivos SMART (Específicos (Specific), measurable (Medibles), alcanzables (Achievable), realistas (Realistic) y de duración limitada (Time-bound)) para abordar las áreas específicas.
- Comunicar a los participantes los objetivos para el cumplimiento de las mejoras requeridas.

Participantes:

Miembros del grupo de Tecnologías de Información del IFAM.

Resultados esperados:

1. Una evaluación clara de la cultura DevOps actual en la organización.
2. Identificación de áreas de mejora específicas.
3. Un plan de acción con objetivos SMART para mejorar la cultura DevOps.
4. Compromiso y colaboración de los equipos para implementar mejoras.
5. Mejora continua de la cultura DevOps en la organización.

La mejora continua es fundamental en DevOps, la retroalimentación constante y la adaptación a medida que evoluciona la organización son claves en una cultura DevOps en el grupo de Tecnologías de Información del IFAM.

Los cuatro (4) instrumentos combinan aprendizaje experiencial, colaboración práctica y recompensas para fomentar una mentalidad ágil y una sencilla adopción de prácticas de una cultura DevOps en el grupo de Tecnologías de Información de IFAM.

Lo que permite crear un ambiente agradable y participativo que puede motivar a los participantes a adoptar el cambio y trabajar de manera más efectiva.

5.3 DESARROLLO DE LA ESTRATEGIA PARA LA IMPLEMENTACIÓN DE UNA CULTURA DEVOPS EN EL INSTITUTO DE FOMENTO Y ASESORIA MUNICIPAL, IFAM.

5.3.1 PLAN DE TRABAJO

Se desarrollará un plan detallado que describa el paso a paso de cómo se implementará la estrategia de una cultura DevOps en el IFAM en base a los instrumentos anteriormente generados:

INSTRUMENTO	MES 1 M1 - PILOTO	MES 2	MES 3 M2/M6 - PUESTA EN MARCHA	MES 4	MES 5	MES 6
Capacitación	✓	✓	✓	✓	✓	✓
Kanban	✓	✓	✓	✓	✓	✓
Lego y Chocolate	✓	✓	✓	✓	✓	✓
Retroalimentación	✓	✓	✓	✓	✓	✓

✓= Punto de revisión

Ilustración 4: Plan de trabajo

Un componente importante de la planificación y gestión efectiva de cualquier proyecto de investigación es la representación visual de las tareas y su programación a

lo largo del tiempo. Por eso razón, se muestra un resumen de tipo diagrama Gantt para el plan de trabajo de este proyecto:

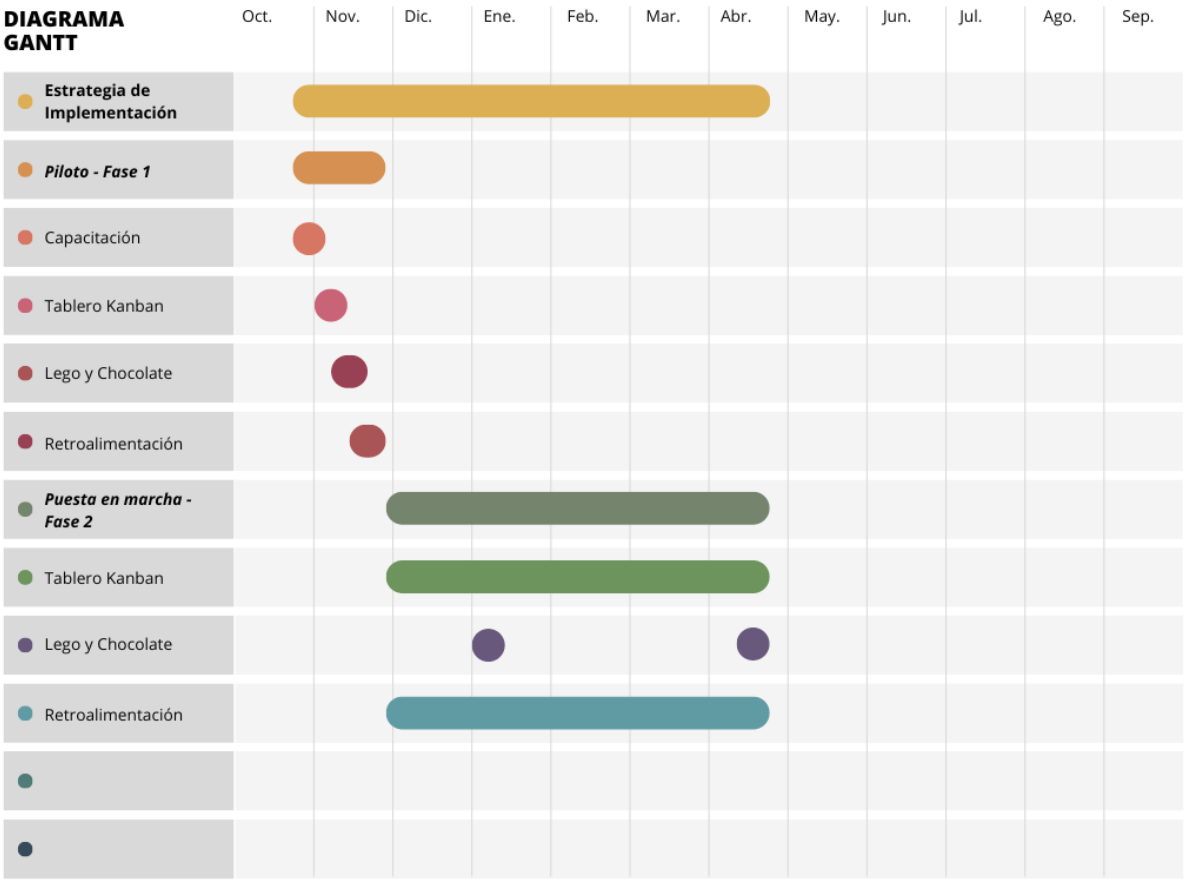


Ilustración 5: Diagrama GANTT

Cronograma completo anexo a este documento

Esta estrategia de implementación busca transformar la cultura de trabajo en el equipo de tecnologías de información del IFAM hacia una mentalidad o cultura DevOps, utilizando herramientas efectivas para promover la colaboración y el aprendizaje compartido en el equipo de trabajo.

5.3.2 RESULTADOS ESPERADOS

Los resultados esperados ayudarán a medir el éxito de la estrategia y a garantizar que se logren los resultados esperados en la implementación de una cultura DevOps.

1. Mejora en la Comunicación:

   - Medir la calidad de la comunicación dentro de los equipos.
   - Objetivo: Aumento del Índice de Comunicación.

2. Colaboración Efectiva:

   - Evaluar la efectividad de la colaboración entre los equipos.
   - Objetivo: Incremento del Índice de Colaboración.

3. Conciencia de la Importancia de la Cultura DevOps:

   - Medir el nivel de conocimiento y comprensión de la cultura DevOps a través de la ejecución del plan de trabajo.
   - Objetivo: Aumento del nivel de conciencia de DevOps.

4. Aprendizaje de la Retroalimentación:

   - Evaluar la capacidad de los equipos para aplicar retroalimentación.

- Objetivo: Aumento del uso efectivo de retroalimentación.

5. Compromiso con la Mejora Continua:

- Registrar el número de propuestas de mejora presentadas por los equipos.
- Objetivo: Aumento en los procesos de mejora continua.

5.4 PROCEDIMIENTO DE IMPLEMENTACIÓN

5.4.1 EJECUCIÓN DE LA ESTRATEGIA DE IMPLEMENTACIÓN DE UNA CULTURA DEVOPS

5.4.1.1 IMPLEMENTACIÓN DEL PILOTO

Se realizó un piloto para evaluar la efectividad de la estrategia de una cultura DevOps en el IFAM en base a los instrumentos y el plan de trabajo realizado.

El piloto servirá para identificar posibles desafíos y permitirá realizar mejoras antes de su adopción generalizada de así requerirse, plan de trabajo del piloto ejecutado anexo a este documento.

En conclusión, el plan piloto implementado ha demostrado ser una herramienta valiosa para recopilar información detallada sobre las actividades llevadas a cabo. La documentación detallada de las actividades realizadas se encuentra disponible en el Anexo 13: Actividades del Piloto, proporcionando una visión completa y transparente de los procesos implementados.

5.4.1.2 RESULTADOS DEL PILOTO

Los instrumentos diseñados y utilizados en el piloto demostraron ser apropiados y efectivos. Sin embargo, durante el proceso de prueba, se identificaron algunas recomendaciones que deben ser consideradas para mejorar aún más la calidad y la eficacia de los instrumentos. Estas sugerencias surgieron a partir de las observaciones y retroalimentación proporcionada por los participantes, se detallan los nuevos instrumentos con cambios en sus pasos:

Instrumento Tablero Kanban:

Objetivo:

Mejorar la comunicación en un entorno DevOps al visualizar y gestionar el flujo de trabajo en ejecución.

Materiales necesarios:

- Pizarra o tablero en blanco (físico o digital).
- Tarjetas adhesivas y Marcadores (si es físico).
- Acceso a herramientas de gestión de proyectos como Trello, Jira, Azure DevOps, entre otros (si es digital).

Pasos del instrumento:

1. Definir las columnas del tablero: Diseñar las columnas que representarán las etapas. Propuesta: "Backlog", "En desarrollo", "Pruebas", "Aprobado", "Despliegue" y "En producción".

2. Selección de proyectos: Seleccionar los proyectos en los cuales se simulará el proceso.

3. Configurar tableros Kanban: Crear y configurar los tableros Kanban utilizando software o herramientas adecuadas.

4. Crear tarjetas: Cada tarea o elemento de trabajo debe representarse mediante una tarjeta. Las tarjetas deben contener información clave, como la descripción de la tarea, el responsable, la prioridad y los plazos.

5. Asignar tareas a participantes: Identificar quién es responsable de cada tarea. Esto fomenta la responsabilidad individual.

6. Piloto del uso de Kanban: A medida que las tareas avancen en el proceso DevOps, deben moverse de una columna a otra. Esto debe hacerse de manera colaborativa.

7. Revisar y actualizar: Realizar reuniones periódicas de seguimiento para revisar el tablero Kanban y actualizar el estado de las tareas, esto fomenta la comunicación entre los miembros del equipo de trabajo.

Participantes:

Miembros del grupo de Tecnologías de Información del IFAM.

Resultados esperados:

1. Mejora de la visibilidad: Todos los equipos tienen una visión clara del progreso y los problemas en el proceso DevOps.

2. Mayor colaboración: Los equipos trabajan juntos para resolver problemas y eliminar cuellos de botella de manera más eficiente.
3. Mayor comunicación: Las actualizaciones en tiempo real y las reuniones de seguimiento fomentan la comunicación efectiva entre los equipos.
4. Reducción de tiempos de entrega: Al identificar y abordar rápidamente los problemas, se aceleran los ciclos de desarrollo y entrega.
5. Mejora de la calidad: La gestión visual y la revisión continua conducen a una mayor calidad del producto y procesos más eficientes.

CAPÍTULO VI. ANALISIS FINANCIERO

6.1 ANALISIS FINANCIERO

El análisis financiero en el contexto de la implementación de una cultura DevOps desempeña un papel fundamental al evaluar el impacto económico y la rentabilidad de las iniciativas relacionadas. Esto se debe a que la mayoría de los beneficios están vinculados a mejoras en la comunicación, colaboración, eficiencia, calidad y velocidad. Es crucial tener en cuenta que estos aspectos no pueden traducirse directamente en ingresos directos.

Los costos asociados se basan en el plan de trabajo realizado, que engloba los gastos operativos relacionados con la ejecución de proyectos, tareas o actividades específicas llevadas a cabo por los profesionales de la unidad de tecnologías de información de IFAM y sus respectivos puestos. Para el período de 6 meses señalado en el plan de trabajo, estos costos ascienden a ¢23.944.500 (Veintitrés millones novecientos cuarenta y cuatro mil quinientos).

6.1.1 INGRESOS

Dentro del marco de una cultura DevOps, la medición de los ingresos puede presentar ciertos desafíos, principalmente debido a la naturaleza de esta metodología. A diferencia de otras áreas empresariales, los ingresos directos no son siempre de fácil medición en un proceso de cambio organizacional, ya que sus resultados no se traducen de manera inmediata en ventas o ingresos financieros concretos. Por lo tanto, es esencial enfocarse en la evaluación del cumplimiento de los resultados esperados, tal como se mencionan en este documento, los cuales reflejan el valor generado por una cultura DevOps.

En este sentido, se puede argumentar que el proyecto genera ingresos en la medida en que su ejecución conlleva eficiencias en los procesos de trabajo, como la implementación de Kanban, una comunicación más efectiva y una mayor colaboración entre equipos. Y en cual se puede considerar un 5% de mejora en la eficiencia como punto de partida para este análisis financiero donde el coste mensual del equipo de tecnologías de información corresponde a ¢14.240.000 aprox.

Se debe destacar que la medición del retorno de inversión (ROI) en una cultura DevOps puede ser más indirecta que en otros sectores. En este contexto, cuantificar el ingreso de un proyecto de preparación exitoso no se basa en las eficiencias tradicionalmente asociadas a la implementación de herramientas DevOps, ya que estas aún no se han materializado. En cambio, se justifica la inversión en nuevos principios en función de los resultados esperados, que apuntan a establecer las bases para futuras mejoras en la comunicación, colaboración, retroalimentación, agilidad, eficiencia y rentabilidad del negocio.

Cronograma Mensual de Ingresos (¢):

Año 1

Tabla 10: Análisis financiero ingresos año 1

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Mejora a la eficiencia 5%	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0

Año 2

Tabla 11: Análisis financiero ingresos año 2

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Mejora a la eficiencia 5%	¢0	¢0	¢0	¢0	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Total	¢0	¢0	¢0	¢0	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

Año 3

Tabla 12: Análisis financiero ingresos año 3

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Mejora a la eficiencia 5%	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Total	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

Año 4

Tabla 13: Análisis financiero ingresos año 4

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Mejora a la eficiencia 5%	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Total	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

Año 5

Tabla 14: Análisis financiero ingresos año 5

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Mejora a la eficiencia 5%	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Total	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

6.1.2 EGRESOS

Los gastos asociados a una cultura DevOps en el IFAM se basan en el plan de trabajo estructurado previamente:

Cronograma Mensual de Egresos (¢):

Año 1

Tabla 15: Análisis financiero egresos año 1

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Piloto	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢5.680.500	¢0
Puesta en Marcha	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢3.652.800

Total	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢5.680.500	¢3.652.800

Año 2

Tabla 16: Análisis financiero egresos año 2

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Piloto	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Puesta en Marcha	¢3.652.800	¢3.652.800	¢3.652.800	¢3.652.800	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢3.652.800	¢3.652.800	¢3.652.800	¢3.652.800	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0

Año 3

Tabla 17: Análisis financiero egresos año 3

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Piloto	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Puesta en Marcha	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0

Año 4

Tabla 18: Análisis financiero egresos año 4

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Piloto	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0

Puesta en Marcha	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0

Año 5

Tabla 19: Análisis financiero egresos año 5

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Piloto	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Puesta en Marcha	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0

6.1.3 FLUJO DE CAJA

Flujo de Caja Mensual (¢):

Año 1

Tabla 20: Análisis financiero flujo de caja año 1

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Ingresos	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Egresos	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢5.680.500	¢3.652.800
Total	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	-¢5.680.500	-¢3.652.800

Año 2

Tabla 21: Análisis financiero flujo de caja año 2

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Ingresos	¢0	¢0	¢0	¢0	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Egresos	¢3.652.800	¢3.652.800	¢3.652.800	¢3.652.800	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	-¢3.652.800	-¢3.652.800	-¢3.652.800	-¢3.652.800	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

Año 3

Tabla 22: Análisis financiero flujo de caja año 3

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Ingresos	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Egresos	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

Año 4

Tabla 23: Análisis financiero flujo de caja año 4

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Ingresos	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Egresos	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

Año 5

Tabla 24: Análisis financiero flujo de caja año 5

	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
Ingresos	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000
Egresos	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0	¢0
Total	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000	¢712.000

Flujo de Caja Acumulado (¢):

Tabla 25: Análisis financiero flujo de caja acumulado

AÑO	FLUJO	ACUMULADO
0	¢23.944.500	¢0
1	-¢9.333.300	-¢9.333.300
2	-¢8.915.200	-¢18.248.500
3	¢8.544.000	-¢9.704.500
4	¢8.544.000	-¢1.160.500
5	¢8.544.000	¢7.383.500

6.1.4 VAN/TIR

VAN y TIR proyectado a cinco años

Tabla 26: Análisis financiero VAN y TIR

		AÑO 1	AÑO 2	AÑO 3	AÑO 4	AÑO 5
Inversión Inicial	-¢23.944.500					
Ingresos	¢31.328.000	¢0	¢5.696.000	¢8.544.000	¢8.544.000	¢8.544.000
Egresos	¢23.944.500	¢9.333.300	¢14.611.200	¢0	¢0	¢0
Total (Ingresos – Gastos)	¢7.383.500	-¢9.333.300	-¢8.915.200	¢8.544.000	¢8.544.000	¢8.544.000
TIR	11%					

VAN	¢7.383.500
Tasa de descuento	0%

En base a los 5 años de este ejercicio financiero, se espera que la inversión en la cultura DevOps genere un ROI positivo. Esto se debe a la mejora en la eficiencia de los procesos, lo que reduce los costos operativos y acelera la entrega de productos y servicios al mercado. La implementación de principios DevOps se traduce en una mayor eficiencia operativa que por ende se muestra en ahorro de costos y mejoras en la calidad de los productos o servicios. Además de los beneficios financieros demostrados en el periodo de 5 años, la cultura DevOps aporta beneficios intangibles significativos, como un mayor compromiso y satisfacción de los empleados, una cultura organizacional más ágil y una mayor capacidad para la innovación y la resolución de problemas por parte del equipo de tecnologías de información del IFAM.

CAPÍTULO VII. CONCLUSIONES Y RECOMENDACIONES

7.1 CONCLUSIONES

1. La implementación de una estrategia para adoptar una cultura DevOps en el IFAM permitirá mejorar la eficiencia, calidad y velocidad en la gestión y mantenimiento de servicios, reflejando un impacto positivo en la experiencia del usuario y la reducción de costos.
2. La obtención de un cambio cultural se revela como un factor crítico y determinante para el éxito de las organizaciones en su proceso de transformación digital.
3. La observación participante revela desafíos significativos en la implementación de una cultura DevOps, como falta de estandarización y colaboración entre equipos, afectando la eficiencia y calidad.
4. El análisis FODA destaca la necesidad de aprovechar fortalezas, abordar debilidades, aprovechar oportunidades y mitigar amenazas para impulsar una implementación exitosa de una cultura DevOps en el IFAM.
5. El análisis financiero refuerza la cultura DevOps al evidenciar un retorno potencialmente significativo sobre la inversión. Esto se traduce en ahorros y beneficios que mejoran la eficiencia operativa, así como la comunicación y colaboración en el IFAM.
6. La realización de un piloto con herramientas de capacitación, tablero Kanban y actividades de compartimiento demuestra ser una estrategia efectiva para consolidar una cultura DevOps en la organización.
7. La cultura DevOps, al fomentar la comunicación y colaboración entre equipos, contribuye a un aumento en la satisfacción del cliente y a la capacidad de

adaptación rápida de la organización a las cambiantes necesidades del mercado.

7.2 RECOMENDACIONES

La implementación exitosa de DevOps va más allá de simplemente adoptar herramientas y automatizaciones. Antes de enrumbarse en este nuevo viaje de transformación digital, es esencial comprender que la cultura DevOps desempeña un papel fundamental en su éxito. Ignorar la importancia de la cultura DevOps y centrarse únicamente en las prácticas técnicas y las herramientas puede llevar a resultados negativos e incluso al fracaso de la iniciativa.

Algunas recomendaciones a la alta gerencia del IFAM:

1. Concientización sobre la Cultura DevOps: Antes de cualquier implementación técnica, concientizar a todos los miembros del equipo sobre la importancia de la cultura DevOps en la mejora continua y la colaboración.
2. Formación y Capacitación Continua: Proporcionar programas de formación constante permite asegurar que todos estén al tanto de las prácticas y principios de DevOps.
3. Fomentar la Colaboración: Establecer canales de comunicación fluidos incentivando la colaboración activa y el intercambio de conocimientos para superar las barreras tradicionales.

4. Liderazgo Transformacional: Contar con líderes que no solo respalden la implementación técnica, sino que también fomenten un cambio cultural, promoviendo la responsabilidad compartida y la innovación.
5. Adaptación de la filosofía DevOps: Continuar con el seguimiento y evaluación constante de las prácticas implementadas para la adaptación a la filosofía DevOps, el Anexo 14: Adaptación de una filosofía DevOps muestra este proceso de adaptación en base a las tecnologías utilizadas por la institución

CAPÍTULO VIII. ANÁLISIS RETROSPECTIVO

8.1 RETROSPECTIVA

La implementación de una cultura DevOps en IFAM ha marcado un cambio significativo, y este análisis retrospectivo destaca elementos clave que han definido su éxito. Más allá de la adopción de herramientas y prácticas técnicas, la evolución cultural se erige como el pilar fundamental, evidenciando que la transformación digital va de la mano con una profunda reconfiguración en la mentalidad organizacional. La observación participante revela desafíos significativos, como la falta de estandarización y colaboración entre equipos, subrayando la necesidad crítica de abordar cuestiones culturales para lograr eficiencia y calidad. El análisis FODA actúa como un mapa estratégico, delineando no solo fortalezas y debilidades, sino también oportunidades y amenazas cruciales. El impacto financiero, al prometer un retorno significativo sobre la inversión, valida no solo la eficiencia operativa mejorada, sino también la solidez de la estrategia implementada. La estrategia de piloto con herramientas de capacitación y actividades de compartimiento no solo se revela como efectiva, sino como un testimonio tangible de cómo acciones concretas consolidan una cultura DevOps. Además, la cultura DevOps no solo ha contribuido a la satisfacción del cliente, sino que ha catalizado la capacidad de adaptación rápida a las dinámicas necesidades del mercado.

Este análisis retrospectivo resalta la complejidad y la riqueza del proceso de implementación de una cultura DevOps en el IFAM. A medida que la organización avanza, la comprensión profunda de su cultura y el compromiso continuo con la mejora serán los pilares que aseguren el éxito de este proceso.

REFERENCIAS

Ambrosio, J. (25 de 01 de 2022). *about.gitlab.com*. Obtenido de about.gitlab.com: https://about.gitlab.com/blog/2022/01/25/how-to-build-out-your-devops-team/

Amigo Pérez, E. (2017). *DevOps para desarrolladores.* Anaya Multimedia.

Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business.* Blue Hole Press.

Arana, I. &. (2017). An exploratory study on the factors influencing the adoption of agile methodologies in software-intensive organizations. Information and Software Technology. *https://doi.org/10.1016/j.infsof.2016.09.003*, 131-143.

AWS. (2023). *aws.amazon.com.* Obtenido de aws.amazon.com: https://aws.amazon.com/es/devops/what-is-devops/

AXELOS. (2021). *ITIL® 4 Foundation: ITIL 4 Edition.*

Berg, C. (2020). *DevOps for Beginners.*

CertiProf. (2023). *CertiProf.* Obtenido de CertiProf: https://certiprof.com/

COIS, C. A. (30 de 04 de 2015). *insights.sei.cmu.edu.* Obtenido de https://insights.sei.cmu.edu/blog/devops-case-study-netflix-and-the-chaos-monkey/

COIS, C. A. (05 de 02 de 2015). *insights.sei.cmu.edu.* Obtenido de https://insights.sei.cmu.edu/blog/devops-case-study-amazon-aws/

Connoly, B. (02 de 07 de 2018). *ciospain.es*. Obtenido de ciospain.es: https://www.ciospain.es/gobierno-ti/gartner-avisa-de-los-riesgos-de-implantar-devops

Cubillo, M. (25 de 08 de 2020). *encora.com.* Obtenido de encora.com: https://www.encora.com/es/blog/devops-mejores-practicas

Cueva Lovelle, J. M. (2019). *DevOps. El camino del éxito en la gestión de sistemas y servicios IT.* Ediciones ENI.

Cueva Lovelle, J. M. (2019). *DevOps. El camino del éxito en la gestión de sistemas y servicios IT.* Ediciones ENI.

devops-certification.org. (2023). *devops-certification.org*. Obtenido de devops-certification.org: https://www.devops-certification.org/What_Are_The_Roles_In_Your_DevOps_Organization.php

Díaz, C. (2019). Los beneficios de la cultura DevOps. *Revista de Innovación y Desarrollo Tecnológico*, 45-56.

Gómez, A. B. (2019). La integración de DevOps con otros marcos de trabajo. *Revista de Tecnologías de la Información y Comunicación*, 78-92.

Google. (2023). *cloud.google.com*. Obtenido de cloud.google.com: https://cloud.google.com/architecture/devops/devops-culture-transform?hl=es-419

Hurtado, H. (13 de 01 de 2020). *medium.com*. Obtenido de medium.com: https://medium.com/@hernanhurtado/11-errores-que-se-deben-evitar-al-querer-implementar-devops-4c98637c56f6

ISO. (2015). *Patente nº ISO 9000:2015.*

Kniberg, H. &. (2010). *Kanban y Scrum, haciendo que los procesos de software funcionen de manera más efectiva.* C4Media Inc.

learn.microsoft.com. (24 de 01 de 2023). *learn.microsoft.com*. Obtenido de learn.microsoft.com: https://learn.microsoft.com/en-us/devops/what-is-devops

Londoño Zapata, J. M. (2019). *Lean Software Development: Implementación para el desarrollo de software ágil.* Ediciones de la U.

López, S. (2018). Roles y responsabilidades en un equipo DevOps. *Revista de Innovación y Desarrollo Tecnológico*, 12-25.

M. Ali Babar, M. A. (2018). Integrating ITIL and DevOps: A Systematic Literature Review. *Journal of Systems and Software*.

Medrano, A. &. (2018). Los desafíos de la implementación de DevOps. *Revista Científica de Tecnología Informática y Comunicación*, 33-44.

Microsoft. (2023). *azure.microsoft.com.* Obtenido de https://azure.microsoft.com/en-us/solutions/devops/devops-at-microsoft

Microsoft. (2023). *azure.microsoft.com.* Obtenido de azure.microsoft.com: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops#culture

Nielsen, N. K. (2020). Bridging the Gap between ITIL and DevOps: A Systematic Literature Review. *Journal of Systems and Software*, 11-22.

Payne, J. (2017). *The Agile Testing Collection.* Wiley.

Pereira, F. B. (2021). A systematic literature review on the application of Kanban in software development. *Journal of Software Engineering Research and Development*, 1-29.

Pérez, J. (2021). Integrando DevOps con otras metodologías ágiles. *Revista de Tecnología, Informática y Comunicación*, 20-35.

Poppendieck, M. &. (2003). *Lean Software Development: An Agile Toolkit.* Addison-Wesley Professional.

Pressman, R. (2014). *Ingeniería del software: un enfoque práctico.* McGraw Hill.

Pressman, R. S. (2010). *Ingeniería del software: Un enfoque práctico.* McGraw-Hill.

Pylayeva, D. (2017). *Introduction to DevOps with Chocolate, LEGO and Scrum Game.* Apress.

Quintana, D. (2017). *DevOps, el camino a la agilidad en el desarrollo y la operación de software.* Ediciones ENI.

Reed, J. P. (2016). *DevOps Handbook: Cómo crear el flujo de trabajo de entrega continua que su empresa necesita.* IT Revolution Press.

Sánchez, C. (2019). La integración de Kanban en DevOps. *Revista de Tecnologías de la Información y Comunicación*, 44-56.

Sánchez, R. (2019). DevOps y la importancia de las herramientas tecnológicas. *Revista de Investigación Académica*, 53-67.

Sharma, V. &. (2021). *DevSecOps: Bridging the Gap between Development, Security, and Operations.* Journal of Physics: Conference Series.

Thorpe, E. (s.f.). *DEVOPS: GUÍA COMPLETA PARA PRINCIPIANTES APRENDE DEVOPS PASO A PASO.* 2019.

Villarreal, V. (2018). *DevOps. Guía práctica para implementar la cultura DevOps en tu empresa.* Alfaomega.

ANEXOS

En este apartado, se presentan los anexos que respaldan y complementan la información expuesta en este trabajo. Los anexos constituyen una extensión visual y documental que enriquece la comprensión del marco conceptual sobre el cual se sustenta la presente investigación. Cada anexo proporciona detalles adicionales, datos relevantes y material complementario que contribuyen a la robustez y claridad de las bases teóricas abordadas en los diferentes capítulos. A través de los anexos, se busca ofrecer una visión más completa y detallada de las fuentes consultadas, metodologías aplicadas, y otros elementos que respaldan la fundamentación teórica. Estos anexos no solo fortalecen la validez de la investigación, sino que también permiten al lector explorar con mayor profundidad los aspectos clave que dan forma al enfoque teórico de este estudio.

A continuación, contenido utilizado para cada capítulo:

ANEXO 1: TERMINOLOGÍA COMPLEMENTARIA

Definición de ITIL

ITIL (Information Technology Infrastructure Library) es un conjunto de prácticas y enfoques recomendados para la gestión de servicios de tecnología de la información (TI). ITIL proporciona un marco de referencia que describe los procesos, procedimientos, tareas y roles necesarios para administrar de manera efectiva los servicios de TI en una organización.

ITIL se basa en una serie de libros publicados por AXELOS, una organización dedicada a la gestión de las mejores prácticas en TI. Estos libros definen un conjunto de conceptos clave y procesos que abarcan todo el ciclo de vida del servicio de TI, desde la estrategia y el diseño hasta la transición, la operación y la mejora continua.

El objetivo principal de ITIL es alinear los servicios de TI con las necesidades del negocio, mejorar la calidad de los servicios, aumentar la eficiencia operativa y proporcionar un enfoque estructurado para la gestión de incidentes, problemas, cambios y otros aspectos relacionados con los servicios de TI.

ITIL se divide en varias áreas de enfoque, conocidas como "prácticas", según el libro “ITIL® 4 Foundation: ITIL 4 Edition” de Axelos (2021)

Desafíos de ITIL en una cultura DevOps

Integrar ITIL (Information Technology Infrastructure Library) en una cultura DevOps puede presentar varios desafíos:

1. Diferentes enfoques de gestión del cambio: ITIL y DevOps tienen perspectivas diferentes sobre la gestión del cambio. Mientras que ITIL tiende a enfocarse en el cambio controlado y documentado, DevOps promueve cambios rápidos y frecuentes. Conciliar estas dos perspectivas puede ser un desafío y requerir un enfoque equilibrado.
2. Velocidad vs. estabilidad: DevOps se centra en la entrega rápida y continua de software, mientras que ITIL busca garantizar la estabilidad y confiabilidad de los servicios de TI. Estos objetivos a veces pueden entrar en conflicto, ya que DevOps

puede favorecer la velocidad sobre la estabilidad, lo que puede generar tensiones con los procesos de gestión de cambios y la estabilidad operativa de ITIL.

3. Cultura y resistencia al cambio: Implementar una cultura DevOps implica cambios tanto en los procesos como en la mentalidad de las personas. ITIL, por otro lado, tiene una estructura y procesos más establecidos. La resistencia al cambio puede surgir cuando se intenta introducir prácticas DevOps y ITIL.
4. Medición del rendimiento: ITIL tiene una amplia gama de indicadores clave de rendimiento (KPI) establecidos para medir el rendimiento de los servicios de TI. Sin embargo, en un entorno DevOps, es posible que se requieran métricas diferentes para evaluar la eficacia de las prácticas ágiles y la entrega continua. Alinear y medir adecuadamente el rendimiento en ambos enfoques puede ser un desafío.
5. Colaboración entre equipos: ITIL a menudo sigue una estructura organizativa jerárquica y funcional, mientras que DevOps se basa en equipos multidisciplinarios y colaborativos. La colaboración y la comunicación efectiva entre estos equipos pueden ser desafiantes, especialmente si existen barreras culturales o de procesos entre ellos.
6. Automatización: La automatización es un pilar fundamental de DevOps, permitiendo la entrega rápida y continua de software. Sin embargo, ITIL puede requerir procesos manuales y aprobaciones rigurosas. Encontrar el equilibrio adecuado entre la automatización en DevOps y los controles y procesos establecidos por ITIL puede requerir un enfoque cuidadoso.

Según el artículo “Bridging the Gap between ITIL and DevOps: A Systematic Literature Review” de Nielsen, N. K. en la revista Journal of Systems and Software (2020), estos desafíos incluyen diferencias en los enfoques de gestión del cambio, el equilibrio entre velocidad y estabilidad, resistencia al cambio en la cultura organizativa, medición del rendimiento, colaboración entre equipos y automatización.

Beneficios de ITIL en una cultura DevOps

La implementación de ITIL (Information Technology Infrastructure Library) en una cultura DevOps puede brindar varios beneficios significativos para la entrega de software, aunque ITIL y DevOps son enfoques diferentes, se pueden combinar para mejorar la gestión de servicios y la entrega de software de manera eficiente:

1. Estándares y mejores prácticas: ITIL proporciona un conjunto sólido de estándares y mejores prácticas para la gestión de servicios de TI. Al integrar estos estándares con los principios de DevOps, se pueden establecer pautas claras y repetibles para la entrega de software, lo que ayuda a garantizar la calidad y la consistencia.
2. Gestión de cambios controlada: ITIL ofrece un enfoque estructurado para la gestión de cambios, lo cual es fundamental para garantizar que los cambios en el software se realicen de manera controlada y se minimicen los riesgos asociados. Al incorporar las prácticas de gestión de cambios de ITIL en una cultura DevOps, se puede lograr una mayor visibilidad y control sobre los cambios en el software, lo que permite una entrega más segura y confiable.

3. Mejora continua: ITIL promueve la mejora continua de los servicios de TI. Al combinar esta mentalidad de mejora continua con los principios ágiles de DevOps, se fomenta una cultura de aprendizaje y adaptación constante. Esto impulsa la evolución y optimización de los procesos de entrega de software, lo que a su vez conduce a una mayor eficiencia y calidad.
4. Gestión de incidentes y problemas: ITIL ofrece un enfoque estructurado para la gestión de incidentes y problemas en el entorno de TI. Al integrar estos procesos con las prácticas de DevOps, se puede establecer una respuesta rápida y eficiente ante incidencias y problemas relacionados con el software. Esto permite identificar y resolver problemas de manera proactiva, minimizando así el impacto en los usuarios finales y mejorando la satisfacción del cliente.
5. Métricas y medición: ITIL proporciona un marco para la definición y medición de métricas clave de rendimiento en la entrega de servicios de TI. Al aplicar estas métricas en un entorno DevOps, se puede obtener una mayor visibilidad sobre el rendimiento y la eficacia de los procesos de entrega de software. Esto facilita la identificación de áreas de mejora, la toma de decisiones basada en datos y la optimización continua.

La combinación de ITIL y DevOps en la entrega de software ofrece beneficios significativos según el artículo "Integrating ITIL and DevOps: A Systematic Literature Review" de M. Ali Babar, Muhammad Aufeef Chauhan, Muhammad Ovais Ahmad (2018) destaca que la adopción de ITIL en una cultura DevOps puede mejorar la gestión de servicios de TI y la entrega de software al proporcionar estándares, mejores prácticas y procesos estructurados.

Definición de Lean Software Development

Lean Software Development es una metodología ágil de desarrollo de software que se basa en los principios y prácticas del pensamiento Lean para eliminar los desperdicios, reducir el tiempo de entrega y aumentar el valor para el cliente. Esta metodología se enfoca en la entrega continua de software de alta calidad, utilizando un enfoque iterativo e incremental. Fue desarrollada por Mary Poppendieck y Tom Poppendieck en su libro "Lean Software Development: An Agile Toolkit" (2003).

La metodología Lean Software Development se basa en siete principios fundamentales que guían el proceso de desarrollo de software:

1. Eliminar los desperdicios: Se enfoca en eliminar todo lo que no agrega valor al cliente. Esto incluye la eliminación de procesos innecesarios, la reducción del trabajo en proceso y la optimización del flujo de trabajo.
2. Ampliar el aprendizaje: Se enfoca en aprender de los errores y mejorar continuamente. Esto implica la realización de experimentos y pruebas de forma constante para aprender y mejorar el proceso de desarrollo de software.
3. Decidir lo más tarde posible: Se enfoca en tomar decisiones basadas en información actualizada y relevante. Esto implica posponer las decisiones hasta que sea necesario, para que se puedan tomar con información actualizada.
4. Entregar rápidamente: Se enfoca en entregar software de alta calidad de forma rápida y constante. Esto implica utilizar un enfoque iterativo e incremental para la entrega de software.

5. Empoderar al equipo: Se enfoca en dar a los equipos de desarrollo la libertad y la responsabilidad para tomar decisiones y resolver problemas de forma autónoma.
6. Crear la integridad: Se enfoca en la creación de un sistema coherente y estable. Esto implica la realización de pruebas y verificaciones de forma constante para asegurar que el sistema esté funcionando correctamente.
7. Ver el todo: Se enfoca en la comprensión del sistema completo y no solo de las partes individuales. Esto implica la colaboración entre equipos y departamentos para asegurar que todos estén trabajando juntos hacia el mismo objetivo.

Empresas líderes que utilizan Lean Software Development

La metodología Lean Software Development según Mary Poppendieck y Tom Poppendieck en su libro "Lean Software Development: An Agile Toolkit" (2003), es utilizada por empresas de todo el mundo, en diversos sectores de la industria del software. Algunas de las empresas más conocidas que han adoptado esta metodología incluyen:

1. Spotify: El popular servicio de streaming de música utiliza la metodología para mejorar continuamente su plataforma y ofrecer una experiencia de usuario de alta calidad.
2. Toyota: El fabricante de automóviles es uno de los pioneros en la aplicación de los principios Lean en la producción y ha aplicado estos mismos principios al desarrollo de software.
3. Intel: La empresa líder en tecnología utiliza la metodología para mejorar la calidad y velocidad del desarrollo de software.

4. IBM: La empresa de tecnología IBM también ha adoptado los principios de LSD en su desarrollo de software. IBM ha utilizado la metodología para mejorar la calidad y la eficiencia en su proceso de desarrollo.
5. Ericsson: La empresa de telecomunicaciones Ericsson ha utilizado LSD para mejorar la eficiencia y la calidad en su proceso de desarrollo de software. La metodología ha ayudado a Ericsson a mejorar la colaboración y la comunicación entre los equipos de desarrollo.

Lean Software Development vs otras metodologías o marcos

Tabla 27: Lean Software Development vs otras metodologías o marcos

Metodología o Marco	Comparación
Agile	Agile es una metodología de desarrollo de software que se enfoca en la flexibilidad y la adaptación al cambio a través de la entrega iterativa e incremental de software funcional. Al igual que Lean Software Development, Agile busca entregar valor al cliente de forma eficiente, pero no hace hincapié en la eliminación de desperdicios. Además, Agile se enfoca en la entrega de software funcional, mientras que Lean Software Development se enfoca en la eliminación de desperdicios en todo el proceso de desarrollo. En el libro "Lean Software Development: An Agile Toolkit" de Mary and Tom Poppendieck (2003), se establecen algunas diferencias clave entre Lean Software Development y Agile: 1. Enfoque en el flujo de trabajo: LSD se centra en optimizar el

	flujo de trabajo, reduciendo los tiempos de espera y eliminando los cuellos de botella en el proceso de desarrollo de software. Agile se centra en la entrega continua de software funcional. 2. Planificación: LSD se basa en la planificación Just-in-Time (JIT), lo que significa que se planea justo antes de que se necesite. Agile se basa en la planificación iterativa y en sprints regulares. 3. Manejo del cambio: LSD se centra en la flexibilidad y la adaptación al cambio, pero trata de minimizar los cambios a medida que se avanza en el proceso de desarrollo. Agile se centra en el cambio continuo y en la entrega de valor al cliente de manera iterativa. 4. Liderazgo y colaboración: LSD se enfoca en la creación de equipos de trabajo altamente colaborativos, con liderazgo centrado en el desarrollo de personas. Agile se centra en la colaboración, pero también enfatiza la importancia del liderazgo y la responsabilidad individual. Scrum es un marco de trabajo ágil que se enfoca en la colaboración, la adaptación al cambio y la entrega de valor de forma iterativa e incremental.
Scrum	Scrum se centra en el trabajo en equipo y la gestión del proyecto, mientras que Lean Software Development se enfoca en la eliminación de desperdicios en todo el proceso de desarrollo. Según el libro "Lean Software Development: An Agile Toolkit" de

	Mary Poppendieck y Tom Poppendieck (2003), la principal diferencia entre Lean Software Development y Scrum es la filosofía subyacente detrás de cada enfoque. Mientras que Lean se enfoca en la eliminación de desperdicios y la optimización del flujo de trabajo, Scrum se enfoca en la entrega iterativa y la mejora continua. En otras palabras, Lean se enfoca en la eficiencia, mientras que Scrum se enfoca en la eficacia.
Kanban	Kanban es un marco de trabajo ágil que se enfoca en la gestión visual de los procesos y la entrega de valor al cliente de forma continua. Kanban se centra en la eliminación de desperdicios y la mejora continua del proceso de trabajo, al igual que Lean Software Development. Sin embargo, Kanban se enfoca en la gestión visual de los procesos, mientras que Lean Software Development se enfoca en la eliminación de desperdicios en todo el proceso de desarrollo.
Extreme Programming (XP)	Según el libro "Kanban y Scrum, haciendo que los procesos de software funcionen de manera más efectiva" de Henrik Kniberg y Mattias Skarin (2010), mientras que Lean Software Development se centra en la eliminación del desperdicio en todo el proceso de desarrollo de software, Kanban se enfoca en la visualización y limitación del trabajo en curso para mejorar la eficiencia y el flujo de trabajo. XP es una metodología ágil de desarrollo de software que se enfoca en la entrega de software de alta calidad a través de prácticas como la programación en pareja, la integración continua y las pruebas automatizadas.

	XP se enfoca en la calidad del software y la entrega de valor al cliente, al igual que Lean Software Development. Sin embargo, XP se enfoca en prácticas específicas de desarrollo de software, mientras que Lean Software Development se enfoca en la eliminación de desperdicios en todo el proceso de desarrollo. Según la investigación de Arana y Vallecillo (2017), LSD y XP son dos metodologías ágiles que comparten algunos principios, como la entrega continua de software y la mejora continua del proceso, pero también tienen diferencias significativas en términos de enfoque en los requisitos del software, el proceso de desarrollo y la entrega de software. SAFe (Scaled Agile Framework) es un marco de trabajo para el desarrollo ágil a gran escala que se utiliza para coordinar y escalar el trabajo de varios equipos en una organización. SAFe se centra en la colaboración, la entrega continua de valor al cliente y la mejora continua del proceso de desarrollo de software.
SAFe	SAFe es un enfoque diseñado para escalar el desarrollo de software ágil en grandes organizaciones, mientras que Lean Software Development se enfoca en la mejora continua y la eliminación de desperdicios en el proceso de desarrollo de software a nivel de equipo. Ambos enfoques comparten algunos principios y prácticas de desarrollo ágil, pero difieren en su enfoque en la escala, roles, responsabilidades, planificación y mejora continua.

	De acuerdo con Jeffrey Payne en su libro "The Agile Testing Collection" (2017), LSD y SAFe comparten la misma filosofía de mejora continua y entregas de valor tempranas. Sin embargo, LSD se enfoca más en la mejora de procesos y reducción de desperdicios, mientras que SAFe se enfoca en la gestión centralizada y estandarización de procesos. En última instancia, la elección entre LSD y SAFe dependerá de las necesidades y objetivos específicos del proyecto y la organización en cuestión.

Desafíos de Lean Software Development en una cultura DevOps

La implementación de Lean Software Development en una cultura DevOps también presenta algunos desafíos:

1. Falta de comprensión de los principios y prácticas de Lean Software Development: Si los equipos de DevOps no están familiarizados con las prácticas de Lean, puede ser difícil para ellos aplicarlas correctamente y obtener los beneficios esperados.
2. Resistencia al cambio: Las organizaciones que han estado utilizando otras metodologías o marcos de trabajo durante mucho tiempo pueden tener dificultades para cambiar a Lean Software Development. Además, algunos miembros del equipo pueden ser reacios a cambiar su forma de trabajo o pueden tener dificultades para adaptarse a las nuevas prácticas.
3. Falta de herramientas y tecnologías adecuadas: La implementación de Lean Software Development en una cultura DevOps las herramientas y tecnologías

utilizadas en DevOps pueden ser diferentes de las utilizadas en Lean Software Development, lo que puede dificultar la integración de las dos metodologías.

4. Enfoque en la mejora continua: Lean Software Development puede chocar con la necesidad de una implementación rápida y frecuente en una cultura DevOps. La prioridad en DevOps es la entrega rápida y frecuente, mientras que Lean Software Development se centra en la mejora continua y la eliminación de desperdicios. Por lo tanto, encontrar un equilibrio entre estos dos objetivos puede ser un desafío.

Según el autor Jorge Mario Londoño Zapata en su libro "Lean Software Development: Implementación para el desarrollo de software ágil" (2019), la implementación exitosa de Lean Software Development en una cultura DevOps requiere del compromiso y liderazgo de la alta dirección, así como de la capacitación y empoderamiento de los equipos de desarrollo. También se requiere de una planificación cuidadosa y la medición continua de los resultados para identificar oportunidades de mejora.

Beneficios de Lean Software Development en una cultura DevOps

La implementación de Lean Software Development en una cultura DevOps también presenta algunos beneficios:

1. Mayor calidad del software: Lean Software Development se enfoca en la eliminación de desperdicio y la mejora continua, lo que puede conducir a la identificación y resolución temprana de problemas de calidad en el software. Además, el enfoque en la colaboración y la comunicación dentro de un equipo de desarrollo ágil puede aumentar la calidad del trabajo entregado.

2. Entregas más rápidas y frecuentes: La eliminación de desperdicios y la mejora continua también pueden ayudar a reducir el tiempo que lleva entregar una nueva funcionalidad. La priorización clara del valor del cliente, el enfoque en la entrega temprana y frecuente y la eliminación de cuellos de botella y procesos innecesarios pueden reducir significativamente el tiempo entre la concepción de una idea y su entrega al cliente.
3. Mayor satisfacción del cliente: El enfoque en el valor del cliente y la entrega temprana y frecuente puede ayudar a garantizar que el software entregado cumpla con las expectativas del cliente. Esto puede conducir a una mayor satisfacción del cliente.
4. Aumento de la eficiencia y productividad: Lean Software Development se enfoca en la eliminación de desperdicios y la mejora continua, lo que puede ayudar a eliminar procesos innecesarios y reducir el tiempo de espera. Esto puede aumentar la eficiencia y la productividad del equipo de desarrollo.
5. Mejor colaboración y comunicación: Lean Software Development fomenta la colaboración y la comunicación dentro del equipo de desarrollo y con los stakeholders. Esto puede mejorar la calidad del trabajo entregado y garantizar que todos los involucrados estén alineados en cuanto a los objetivos y prioridades del proyecto.

Según Poppendieck y Poppendieck en su libro “Lean Software Development: An Agile Toolkit” (2011), la implementación de Lean Software Development en una cultura DevOps puede mejorar significativamente la calidad del software y reducir los tiempos de entrega. Esto se debe a que Lean Software Development se enfoca en la eliminación

de desperdicios y la mejora continua del proceso, lo que lleva a una mayor eficiencia y a la entrega de productos de mayor valor para el cliente.

Definición de Kanban

Kanban es un método de gestión de procesos y trabajo que se utiliza para visualizar, controlar y mejorar el flujo de trabajo en un sistema. Kanban se originó en la industria manufacturera y fue introducido por primera vez por Toyota en los años 40, como una herramienta para mejorar la eficiencia en la producción. Desde entonces, ha evolucionado y se ha adaptado a una amplia variedad de sectores y disciplinas, incluyendo la gestión de proyectos y el desarrollo de software.

Según Pereira, Branco y Carneiro en el artículo “A systematic literature review on the application of Kanban in software development” (2021), el método Kanban se basa en el uso de tableros visuales para visualizar el flujo de trabajo y controlar el progreso de las tareas. Cada tarea se representa en una tarjeta, que se mueve a través de diferentes columnas del tablero para indicar su estado de avance. El objetivo es maximizar el flujo de trabajo, minimizar los tiempos de espera y reducir los cuellos de botella, lo que a su vez permite una entrega más rápida y eficiente de los productos o servicios.

Para implementar el método Kanban, se requiere de la creación de un tablero visual que represente el flujo de trabajo, el establecimiento de límites de trabajo en progreso (WIP) para evitar la sobrecarga del equipo, y la medición y análisis del rendimiento para identificar oportunidades de mejora y optimización.

Integración de Lean Software Development y Kanban

La integración de Lean Software Development y Kanban se basa en la idea de que la gestión del flujo de trabajo es fundamental para la mejora continua y la eliminación de desperdicios en el desarrollo de software. Según David J. Anderson en su libro "Kanban: Successful Evolutionary Change for Your Technology Business" (2010), Kanban se enfoca en mejorar la eficiencia y efectividad del flujo de trabajo, mientras que Lean Software Development se enfoca en mejorar la calidad del producto y la satisfacción del cliente.

Lean Software Development busca maximizar el valor entregado al cliente al mismo tiempo que se minimizan los desperdicios en el proceso de desarrollo. Por otro lado, Kanban es un marco de trabajo que se enfoca en la visualización y optimización del flujo de trabajo. La integración de Lean Software Development y Kanban puede proporcionar un enfoque más ágil, eficiente y efectivo para el desarrollo de software.

La integración de Lean Software Development y Kanban se puede lograr de varias maneras. Algunas formas en que estas prácticas se pueden integrar:

1. Visualización del flujo de trabajo: Kanban se basa en la visualización del flujo de trabajo para mejorar la eficiencia y la transparencia. En este sentido, Lean Software Development puede aprovechar esta práctica para visualizar el flujo de trabajo del desarrollo de software. Esto permitirá a los equipos ver claramente cómo se está desarrollando el software, identificar cuellos de botella y encontrar formas de mejorar el proceso.
2. Optimización del proceso: Kanban se centra en la optimización del proceso a través de la eliminación de desperdicios y la mejora continua. De manera similar,

Lean Software Development busca optimizar el proceso a través de la eliminación de desperdicios y la mejora continua. La integración de ambos enfoques puede proporcionar un marco sólido para la eliminación de desperdicios y la mejora continua del proceso de desarrollo de software.

3. Establecimiento de límites WIP: Kanban establece límites en el trabajo en proceso (WIP) para evitar la sobrecarga de trabajo. Lean Software Development también hace hincapié en la importancia de limitar el WIP para evitar la sobrecarga de trabajo. La integración de ambas prácticas puede proporcionar una forma efectiva de gestionar el WIP y garantizar que el equipo esté trabajando en la cantidad adecuada de tareas para maximizar la eficiencia.
4. Enfoque en la entrega de valor: Lean Software Development y Kanban se centran en la entrega de valor al cliente. La integración de ambos enfoques puede proporcionar una forma efectiva de asegurarse de que el equipo esté trabajando en las tareas más importantes y valiosas para el cliente.

Automatización

La automatización se refiere al uso de tecnología para realizar tareas o procesos de forma automatizada, sin intervención humana.

La automatización permite una entrega continua y rápida de software. En DevOps, se utilizan diferentes herramientas de automatización para los procesos manuales y repetitivos en la entrega de software.

Algunas de las herramientas de automatización comunes utilizadas en DevOps incluyen:

1. Herramientas de gestión de configuración: Estas herramientas se utilizan para automatizar la configuración del software y el entorno en el que se ejecuta.
2. Herramientas de integración continua: Estas herramientas se utilizan para integrar y construir el código de forma continua.
3. Herramientas de entrega continua: Estas herramientas se utilizan para automatizar la entrega del software.

Según J. Paul Reed en su libro "DevOps Handbook: Cómo crear el flujo de trabajo de entrega continua que su empresa necesita" (2016), la automatización es una de las cinco características clave de una cultura DevOps exitosa. Reed enfatiza que la automatización no solo se trata de la eliminación de trabajo manual, sino que también permite la estandarización y la mejora continua de los procesos. La automatización se puede lograr mediante el uso de herramientas de automatización de infraestructura, como Chef y Puppet, herramientas de automatización de prueba, como Selenium, y herramientas de automatización de despliegue, como Jenkins y Ansible.

La automatización mejora la eficiencia y la calidad de la entrega de software, al automatizar los procesos manuales y repetitivos lo que reduce la probabilidad de errores humanos y mejora la consistencia en la entrega de software.

Pruebas de Software

Las pruebas de software garantizan la calidad del software entregado. En DevOps, se utilizan diferentes estrategias de pruebas de software para garantizar que el software entregado cumpla con los requisitos del cliente, sea seguro y estable.

Algunas de las estrategias de pruebas de software comunes utilizadas en DevOps incluyen:

1. Pruebas unitarias: Estas pruebas se utilizan para probar componentes individuales del software. Las pruebas unitarias se realizan utilizando herramientas de prueba automatizadas y se integran en el proceso de integración continua.
2. Pruebas de integración: Estas pruebas se utilizan para probar cómo los diferentes componentes del software se integran entre sí. Las pruebas de integración se realizan utilizando herramientas de prueba automatizadas y se integran en el proceso
3. Pruebas de aceptación: Estas pruebas se utilizan para probar si el software cumple con los requisitos del cliente. Las pruebas de aceptación se realizan en colaboración con el cliente y se utilizan para validar que el software entregado cumple con los requisitos definidos.
4. Pruebas de rendimiento: Estas pruebas se utilizan para probar el rendimiento del software en diferentes escenarios de carga. Las pruebas de rendimiento se realizan utilizando herramientas de prueba automatizadas y se utilizan para garantizar que el software entregado sea escalable y estable bajo diferentes niveles de carga.

Según Pressman en su libro “Ingeniería del software: un enfoque práctico (7ª ed.)” (2014), las pruebas de software son un proceso de ejecución controlada de un sistema con el propósito de descubrir errores, y de verificar que el software cumpla con las

especificaciones y requisitos definidos. Además, se busca asegurar que el software se desempeñe de manera eficiente y efectiva, y que cumpla con los estándares de calidad requeridos. Las pruebas de software pueden ser realizadas por desarrolladores o por personal especializado en pruebas de software, y pueden ser manuales o automatizadas

Las pruebas de software permiten detectar errores tempranos en el proceso de entrega, permiten mejorar la calidad del software entregado y garantizar que cumpla con los requisitos del cliente.

Ciclo de vida de desarrollo de software

El ciclo de vida del software es el proceso completo que involucra la planificación, diseño, desarrollo, prueba, implementación y mantenimiento del software. En una cultura DevOps, el ciclo de vida del software es una parte importante del proceso de entrega de software, ya que se enfoca en la calidad y la eficiencia de la entrega de software.

Según Pressman en su libro "Ingeniería del software: un enfoque práctico" (2010), el ciclo de vida del software puede ser representado por diferentes modelos, tales como el modelo en cascada, el modelo en espiral, el modelo iterativo e incremental, y el modelo de desarrollo ágil y comprende seis fases:

1. Planificación: La planificación es la etapa inicial del ciclo de vida del software en DevOps. Durante esta etapa, se identifican las necesidades del negocio y se definen los requisitos del software. También se establecen los objetivos de la entrega de software, se establece la prioridad de los requisitos y se planifican los recursos necesarios.

2. Diseño: La etapa de diseño implica la creación de la arquitectura del software y la definición de los componentes y módulos necesarios para cumplir con los requisitos establecidos en la etapa de planificación.
3. Desarrollo: La etapa de desarrollo implica la creación del código fuente del software utilizando prácticas de codificación de alta calidad y la implementación de pruebas para garantizar la calidad del código.
4. Prueba: La etapa de prueba implica la realización de pruebas en el software para detectar errores y garantizar la calidad del software antes de su implementación. Las pruebas se realizan de manera continua y automatizada para detectar problemas temprano en el ciclo de vida del software.
5. Implementación: La etapa de implementación implica la entrega del software al entorno de producción y la configuración de los servidores para soportar la nueva aplicación o actualización. La implementación se realiza de manera continua y automatizada, lo que permite una entrega de software más rápida y eficiente.
6. Mantenimiento: La etapa de mantenimiento implica la corrección de errores y la aplicación de actualizaciones en el software para garantizar su funcionamiento continuo.

Métricas de rendimiento

Las métricas de rendimiento son medidas cuantitativas que se utilizan para evaluar el desempeño o éxito de un sistema, proceso o persona en función de objetivos específicos, pueden ser utilizadas en una amplia variedad de contextos, incluyendo negocios, deportes, educación, salud, tecnología, entre otros.

Las métricas de rendimiento en una cultura DevOps permiten medir el éxito de la entrega de software y garantizar la calidad y eficiencia del proceso. Las métricas de rendimiento también pueden ayudar a identificar áreas de mejora y a tomar decisiones informadas sobre cómo mejorar la entrega de software.

Algunas de las métricas de rendimiento importantes en una cultura DevOps incluyen:

1. Tiempo de ciclo: El tiempo de ciclo es el tiempo que tarda desde que se inicia una tarea hasta que se completa. El tiempo de ciclo se refiere al tiempo que tarda en entregarse el software desde la planificación hasta la implementación. Una métrica de tiempo de ciclo reducido puede indicar una entrega de software más rápida y eficiente.
2. Frecuencia de entrega: La frecuencia de entrega se refiere a la cantidad de veces que se entrega software en un período de tiempo determinado. Una métrica de frecuencia de entrega más alta puede indicar una entrega de software más rápida y eficiente.
3. Tiempo de recuperación: El tiempo de recuperación se refiere al tiempo que tarda en restaurarse el servicio en caso de una interrupción del servicio. Una métrica de tiempo de recuperación reducido puede indicar una capacidad de respuesta más rápida a los problemas y una menor cantidad de tiempo de inactividad del servicio.
4. Tiempo de espera en cola: El tiempo de espera en cola se refiere al tiempo que un trabajo pasa en una cola de espera antes de que se inicie su procesamiento.

Una métrica de tiempo de espera en cola reducido puede indicar una capacidad de procesamiento más rápida y una entrega de software más eficiente.

5. Tasa de éxito de la implementación: La tasa de éxito de la implementación se refiere a la cantidad de implementaciones exitosas en relación con el total de implementaciones. Una alta tasa de éxito de implementación puede indicar una entrega de software de mayor calidad y una menor cantidad de problemas en producción.

Según Pressman en su libro “Ingeniería del software: un enfoque práctico (7ª ed.)” (2014), las métricas de rendimiento se utilizan para evaluar y mejorar la calidad del proceso de desarrollo de software y pueden ser recopiladas durante todo el ciclo de vida del software. Las métricas de rendimiento pueden ser utilizadas para identificar cuellos de botella y áreas de mejora, lo que puede ayudar a los equipos de desarrollo a tomar medidas proactivas para mejorar la eficiencia y la calidad del software.

DevSecOps

DevSecOps es una metodología de desarrollo de software que integra la seguridad en todo el ciclo de vida de desarrollo de software (SDLC) y tiene como objetivo principal crear software de alta calidad y seguro en un ciclo de tiempo más corto.

DevSecOps es una práctica emergente en el mundo de la tecnología que busca integrar la seguridad de forma temprana y continua en los procesos de desarrollo, implementación y operación de software. Esta práctica se basa en la idea de que la seguridad no puede ser una consideración después de que el software ha sido desarrollado, sino que debe ser un aspecto integral de todo el ciclo de vida del software.

En DevSecOps, el equipo de seguridad se une al equipo de desarrollo y operaciones para colaborar en la creación y gestión de un ambiente seguro. Los desarrolladores y operadores de sistemas son capacitados para comprender las mejores prácticas de seguridad, mientras que los especialistas en seguridad pueden contribuir a la automatización y pruebas de seguridad continuas.

Algunos de los elementos clave de DevSecOps incluyen el uso de herramientas de seguridad integradas, el monitoreo y análisis de los registros de seguridad, la incorporación de la seguridad en la cultura de la organización y el mantenimiento de una buena comunicación entre los equipos de desarrollo, operaciones y seguridad.

Según Sharma y Goyal en el artículo “DevSecOps: Bridging the Gap between Development, Security, and Operations” (2021), DevSecOps es una práctica de integración de seguridad en el ciclo de vida de desarrollo de software, que proporciona seguridad integrada en todas las etapas de la entrega de software. DevSecOps es un enfoque más proactivo y holístico de seguridad en comparación con los enfoques tradicionales de seguridad que se enfocan en la seguridad después de que se ha desarrollado el software. La metodología DevSecOps busca aumentar la eficiencia, calidad y seguridad en todo el proceso de desarrollo de software, en lugar de agregar seguridad después de la implementación.

La integración de DevSecOps en una cultura DevOps puede ayudar a mejorar la calidad y seguridad del software entregado al mercado, al tiempo que se reduce el tiempo de entrega al enfatizar la seguridad desde el principio, los equipos de desarrollo pueden

detectar y solucionar problemas de seguridad temprano en el ciclo de vida del software, lo que ahorra tiempo y reduce costos.

ANEXO 2: GUÍA DE ENTREVISTA INDIVIDUAL

Título: Evaluación de los procesos actuales de entrega de software y la implementación de una cultura DevOps en la organización.

Objetivo: Obtener una comprensión profunda de los procesos actuales de entrega de software en la organización, identificar los procesos críticos y las posibles oportunidades de mejora, y conocer las diferentes perspectivas y opiniones de los miembros del equipo sobre la implementación de una cultura DevOps.

Temas por tratar:

- Experiencia en la organización y el rol actual del entrevistado.
- Procesos actuales de entrega de software.
- Identificación de los procesos críticos en la entrega de software.
- Oportunidades de mejora en los procesos actuales.
- Perspectivas y opiniones sobre la implementación de una cultura DevOps.

Preguntas:

1. ¿Cuál es su experiencia en la organización y cuál es su rol actual?
2. ¿Cuáles son los procesos actuales de entrega de software en la organización?
3. ¿Cuáles son los principales desafíos que enfrenta en los procesos actuales de entrega de software?
4. ¿Cuáles son los procesos críticos en la entrega de software y por qué son importantes?
5. ¿Ha identificado oportunidades de mejora en los procesos actuales de entrega de software? ¿Cuáles son?
6. ¿Qué herramientas y tecnologías utiliza actualmente para la entrega de software?
7. ¿Ha trabajado anteriormente con una cultura DevOps? ¿Cuál fue su experiencia?
8. ¿Cómo cree que la implementación de una cultura DevOps podría mejorar los procesos actuales de entrega de software?
9. ¿Qué cambios o ajustes deberían hacerse para implementar una cultura DevOps en la organización?
10. ¿Qué desafíos o barreras podrían surgir durante la implementación de una cultura DevOps en la organización?

Fecha de la entrevista:

ANEXO 3: GUIA DE GRUPOS FOCALES

Introducción:

El propósito de este grupo focal es obtener una visión más amplia de las diferentes perspectivas y opiniones de los miembros del equipo sobre la implementación de una cultura DevOps en la organización. Este grupo focal permitirá identificar los principales desafíos y obstáculos que podrían surgir durante la implementación, y también discutir y evaluar posibles soluciones y estrategias para superar estos desafíos.

Participantes: El grupo focal estará compuesto por los equipos de desarrollo, operaciones y jefatura. Se invitará a un máximo de 2 participantes para cada grupo focal.

Procedimiento:

1. Introducción: el moderador presentará la agenda del grupo focal y se presentará a los participantes. También se les informará sobre los objetivos y la importancia del grupo focal.
2. Establecimiento de normas: el moderador establecerá las normas para el grupo focal y asegurará que todos los participantes las comprendan.
3. Discusión temática: el moderador presentará temas específicos relacionados con la implementación de una cultura DevOps, y los participantes discutirán y compartirán sus perspectivas y opiniones sobre cada tema.

4. Identificación de desafíos y obstáculos: el grupo focal discutirá los principales desafíos y obstáculos que podrían surgir durante la implementación de una cultura DevOps en la organización.

5. Discusión de soluciones y estrategias: el grupo focal discutirá y evaluará posibles soluciones y estrategias para superar los desafíos y obstáculos identificados anteriormente.

6. Cierre: el moderador agradecerá a los participantes por su tiempo y su contribución, y les informará sobre los próximos pasos del proyecto.

Confidencialidad: Se garantizará la confidencialidad de los participantes del grupo focal. Los datos y opiniones compartidos en el grupo focal serán utilizados únicamente para los propósitos del proyecto y no se divulgarán a terceros sin el consentimiento explícito de los participantes.

ANEXO 4: FORMATO DE OBSERVACIÓN PARTICIPANTE

Objetivos de la observación:

- Comprender los procesos y prácticas de entrega de software en la organización.
- Identificar problemas y obstáculos en la implementación de una cultura DevOps.
- Evaluar la efectividad de las soluciones y estrategias propuestas para superar estos problemas.

Fecha y hora de la observación:

Lugar de la observación:

Duración de la observación:

Miembros del equipo observados:

Tabla 28: Observación partiipante

Nombre	Rol
Colaborador 1	Profesional Especialista UTI
Colaborador 3	Profesional UTI

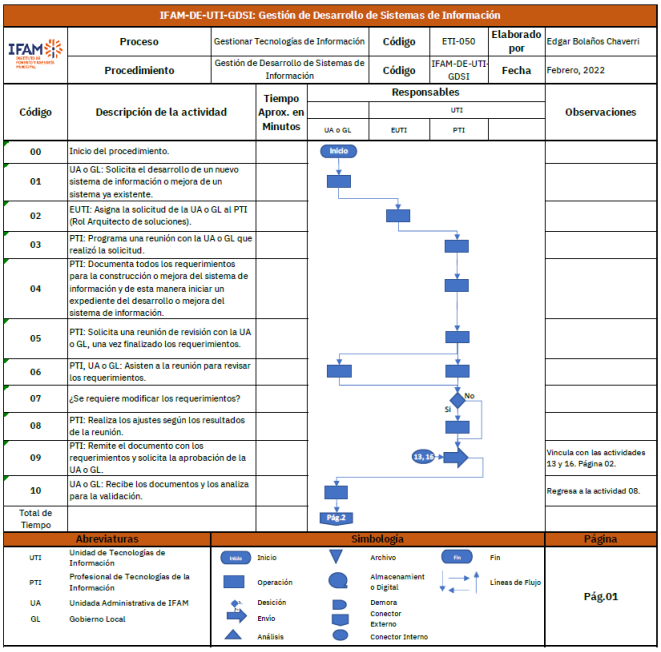
Procesos y prácticas observadas:

Observaciones y notas:

Problemas y obstáculos identificados durante la observación:

ANEXO 5: PROCEDIMIENTOS INSTITUCIONALES

a. ANEXO Definición de requerimientos:



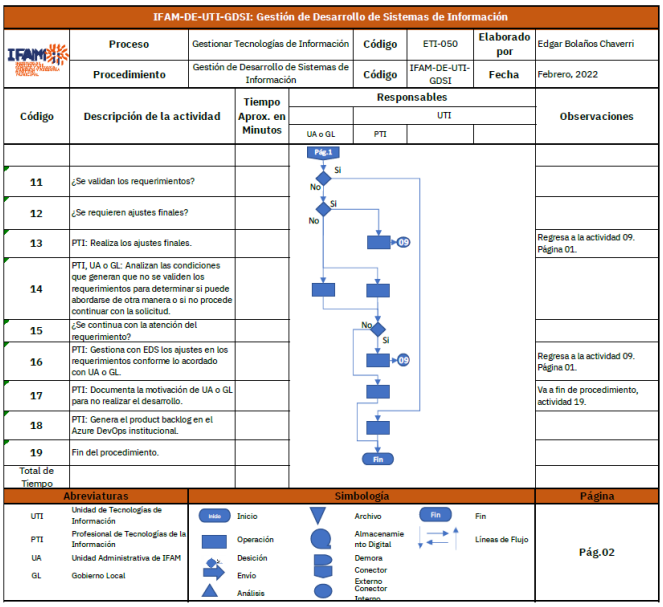
IFAM-DE-UTI-GDSI: Gestión de Desarrollo de Sistemas de Información

Proceso	Gestionar Tecnologías de Información	Código	ETI-050	Elaborado por	Edgar Bolaños Chaverri
Procedimiento	Gestión de Desarrollo de Sistemas de Información	Código	IFAM-DE-UTI-GDSI	Fecha	Febrero, 2022

Código	Descripción de la actividad	Tiempo Aprox. en Minutos	Responsables			Observaciones
				UTI		
			UA o GL	EUTI	PTI	
00	Inicio del procedimiento.					
01	UA o GL: Solicita el desarrollo de un nuevo sistema de información o mejora de un sistema ya existente.					
02	EUTI: Asigna la solicitud de la UA o GL al PTI (Rol Arquitecto de soluciones).					
03	PTI: Programa una reunión con la UA o GL que realizó la solicitud.					
04	PTI: Documenta todos los requerimientos para la construcción o mejora del sistema de información y de esta manera iniciar un expediente del desarrollo o mejora del sistema de información.					
05	PTI: Solicita una reunión de revisión con la UA o GL, una vez finalizado los requerimientos.					
06	PTI, UA o GL: Asisten a la reunión para revisar los requerimientos.					
07	¿Se requiere modificar los requerimientos?					
08	PTI: Realiza los ajustes según los resultados de la reunión.					
09	PTI: Remite el documento con los requerimientos y solicita la aprobación de la UA o GL.					Vincula con las actividades 13 y 16. Página 02.
10	UA o GL: Recibe los documentos y los analiza para la validación.					Regresa a la actividad 08.
Total de Tiempo						

Abreviaturas		Simbología				Página
UTI	Unidad de Tecnologías de Información	Inicio	Archivo	Fin	Fin	Pág.01
PTI	Profesional de Tecnologías de la Información	Operación	Almacenamiento Digital		Líneas de Flujo	
UA	Unidada Administrativa de IFAM	Desición	Demora			
GL	Gobierno Local	Envio	Conector Externo			
		Análisis	Conector Interno			

Ilustración 6: Definición de requerimientos parte 1



IFAM-DE-UTI-GDSI: Gestión de Desarrollo de Sistemas de Información

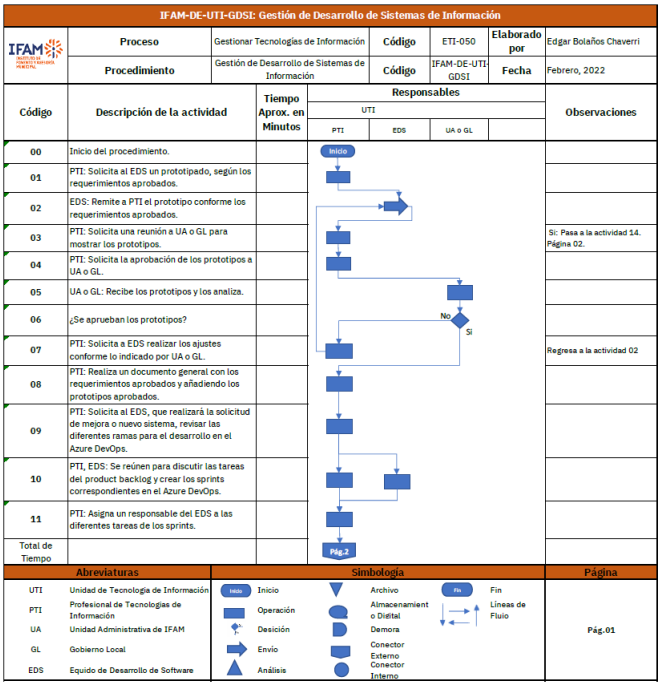
Proceso	Gestionar Tecnologías de Información		Código	ETI-050	Elaborado por	Edgar Bolaños Chaverri
Procedimiento	Gestión de Desarrollo de Sistemas de Información		Código	IFAM-DE-UTI-GDSI	Fecha	Febrero, 2022

Código	Descripción de la actividad	Tiempo Aprox. en Minutos	Responsables				Observaciones
				UTI			
			UA o GL	PTI			
11	¿Se validan los requerimientos?						
12	¿Se requieren ajustes finales?						
13	PTI: Realiza los ajustes finales.						Regresa a la actividad 09. Página 01.
14	PTI, UA o GL: Analizan las condiciones que generan que no se validen los requerimientos para determinar si puede abordarse de otra manera o si no procede continuar con la solicitud.						
15	¿Se continua con la atención del requerimiento?						
16	PTI: Gestiona con EDS los ajustes en los requerimientos conforme lo acordado con UA o GL.						Regresa a la actividad 09. Página 01.
17	PTI: Documenta la motivación de UA o GL para no realizar el desarrollo.						Va a fin de procedimiento, actividad 19.
18	PTI: Genera el product backlog en el Azure DevOps institucional.						
19	Fin del procedimiento.						
Total de Tiempo							

Abreviaturas		Simbología				Página
UTI	Unidad de Tecnologías de Información	Inicio	Inicio	Archivo	Fin	Fin
PTI	Profesional de Tecnologías de la Información	Operación	Operación	Almacenamiento Digital	Líneas de Flujo	Líneas de Flujo
UA	Unidad Administrativa de IFAM	Desición	Desición	Demora		Pág.02
GL	Gobierno Local	Envío	Envío	Conector Externo		
		Análisis	Análisis	Conector Interno		

Ilustración 7: Definición de requerimientos parte 2

b. Desarrollo del sistema de información.



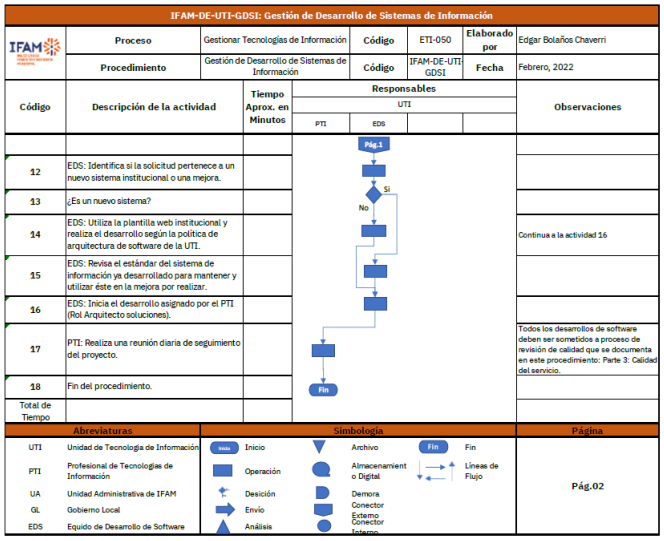
IFAM-DE-UTI-GDSI: Gestión de Desarrollo de Sistemas de Información

Proceso	Gestionar Tecnologías de Información		Código	ETI-050	Elaborado por	Edgar Bolaños Chaverri
Procedimiento	Gestión de Desarrollo de Sistemas de Información		Código	IFAM-DE-UTI-GDSI	Fecha	Febrero, 2022

Código	Descripción de la actividad	Tiempo Aprox. en Minutos	Responsables				Observaciones
			UTI				
			PTI	EDS	UA o GL		
00	Inicio del procedimiento.						
01	PTI: Solicita al EDS un prototipado, según los requerimientos aprobados.						
02	EDS: Remite a PTI el prototipo conforme los requerimientos aprobados.						
03	PTI: Solicita una reunión a UA o GL para mostrar los prototipos.						Si: Pasa a la actividad 14. Página 02.
04	PTI: Solicita la aprobación de los prototipos a UA o GL.						
05	UA o GL: Recibe los prototipos y los analiza.						
06	¿Se aprueban los prototipos?						
07	PTI: Solicita a EDS realizar los ajustes conforme lo indicado por UA o GL.						Regresa a la actividad 02
08	PTI: Realiza un documento general con los requerimientos aprobados y añadiendo los prototipos aprobados.						
09	PTI: Solicita al EDS, que realizará la solicitud de mejora o nuevo sistema, revisar las diferentes ramas para el desarrollo en el Azure DevOps.						
10	PTI, EDS: Se reúnen para discutir las tareas del product backlog y crear los sprints correspondientes en el Azure DevOps.						
11	PTI: Asigna un responsable del EDS a las diferentes tareas de los sprints.						
Total de Tiempo							

Abreviaturas		Simbología				Página
UTI	Unidad de Tecnología de Información	Inicio	Inicio	Archivo	Fin	Fin
PTI	Profesional de Tecnologías de Información	Operación	Operación	Almacenamiento Digital	Líneas de Fluio	Líneas de Fluio
UA	Unidad Administrativa de IFAM	Desición	Desición	Demora		Pág.01
GL	Gobierno Local	Envío	Envío	Conector Externo		
EDS	Equido de Desarrollo de Software	Análisis	Análisis	Conector Interno		

Ilustración 8: Desarrollo del sistema de información parte 1



IFAM-DE-UTI-GDSI: Gestión de Desarrollo de Sistemas de Información

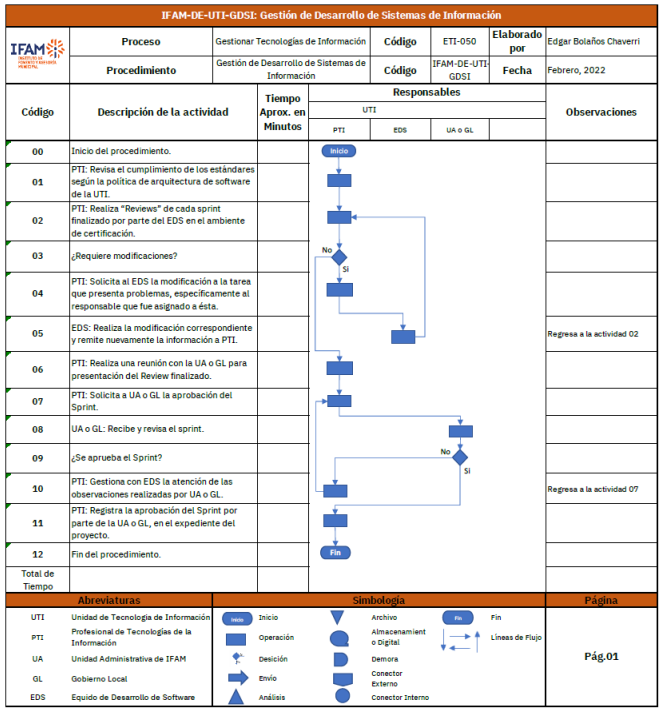
Proceso	Gestionar Tecnologías de Información	Código	ETI-050	Elaborado por	Edgar Bolaños Chaverri
Procedimiento	Gestión de Desarrollo de Sistemas de Información	Código	IFAM-DE-UTI-GDSI	Fecha	Febrero, 2022

Código	Descripción de la actividad	Tiempo Aprox. en Minutos	Responsables UTI: PTI	EDS			Observaciones
12	EDS: Identifica si la solicitud pertenece a un nuevo sistema institucional o una mejora.						
13	¿Es un nuevo sistema?						
14	EDS: Utiliza la plantilla web institucional y realiza el desarrollo según la política de arquitectura de software de la UTI.						Continua a la actividad 16
15	EDS: Revisa el estándar del sistema de información ya desarrollado para mantener y utilizar éste en la mejora por realizar.						
16	EDS: Inicia el desarrollo asignado por el PTI (Rol Arquitecto soluciones).						
17	PTI: Realiza una reunión diaria de seguimiento del proyecto.						Todos los desarrollos de software deben ser sometidos a proceso de revisión de calidad que se documenta en este procedimiento: Parte 3: Calidad del servicio.
18	Fin del procedimiento.						
Total de Tiempo							

Abreviaturas		Simbología				Página
UTI	Unidad de Tecnología de Información	Inicio	Archivo	Fin	Fin	Pág.02
PTI	Profesional de Tecnologías de Información	Operación	Almacenamiento Digital		Líneas de Flujo	
UA	Unidad Administrativa de IFAM	Desición	Demora			
GL	Gobierno Local	Envío	Conector Externo			
EDS	Equido de Desarrollo de Software	Análisis	Conector Interno			

Ilustración 9: Desarrollo del sistema de información parte 2

c. Calidad del servicio



IFAM-DE-UTI-GDSI: Gestión de Desarrollo de Sistemas de Información

Proceso	Gestionar Tecnologías de Información	Código	ETI-050	Elaborado por	Edgar Bolaños Chaverri
Procedimiento	Gestión de Desarrollo de Sistemas de Información	Código	IFAM-DE-UTI-GDSI	Fecha	Febrero, 2022

Código	Descripción de la actividad	Tiempo Aprox. en Minutos	Responsables UTI: PTI	EDS	UA o GL		Observaciones
00	Inicio del procedimiento.						
01	PTI: Revisa el cumplimiento de los estándares según la política de arquitectura de software de la UTI.						
02	PTI: Realiza "Reviews" de cada sprint finalizado por parte del EDS en el ambiente de certificación.						
03	¿Requiere modificaciones?						
04	PTI: Solicita al EDS la modificación a la tarea que presenta problemas, específicamente al responsable que fue asignado a ésta.						
05	EDS: Realiza la modificación correspondiente y remite nuevamente la información a PTI.						Regresa a la actividad 02
06	PTI: Realiza una reunión con la UA o GL para presentación del Review finalizado.						
07	PTI: Solicita a UA o GL la aprobación del Sprint.						
08	UA o GL: Recibe y revisa el sprint.						
09	¿Se aprueba el Sprint?						
10	PTI: Gestiona con EDS la atención de las observaciones realizadas por UA o GL.						Regresa a la actividad 07
11	PTI: Registra la aprobación del Sprint por parte de la UA o GL, en el expediente del proyecto.						
12	Fin del procedimiento.						
Total de Tiempo							

Abreviaturas		Simbología				Página
UTI	Unidad de Tecnología de Información	Inicio	Archivo	Fin	Fin	Pág.01
PTI	Profesional de Tecnologías de la Información	Operación	Almacenamiento Digital		Líneas de Flujo	
UA	Unidad Administrativa de IFAM	Desición	Demora			
GL	Gobierno Local	Envío	Conector Externo			
EDS	Equido de Desarrollo de Software	Análisis	Conector Interno			

Ilustración 10: Calidad del servicio

d. Control de versiones

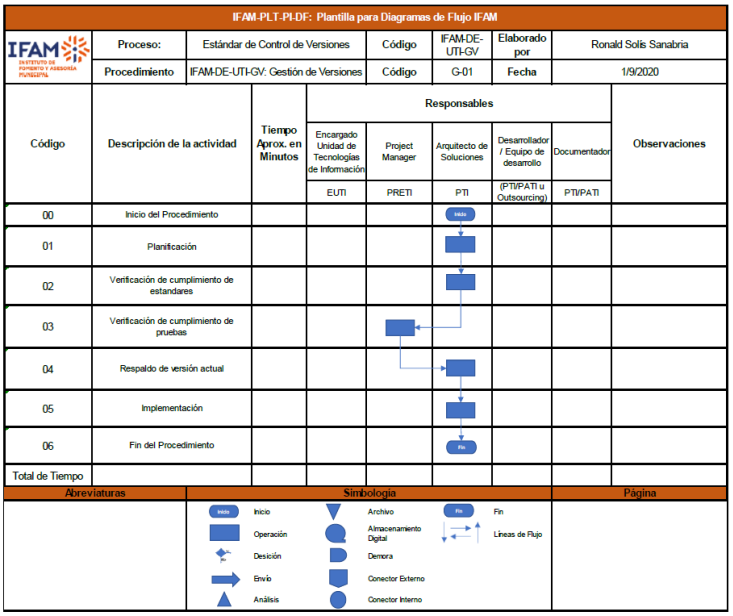


Ilustración 11: Control de versiones

e. Publicación

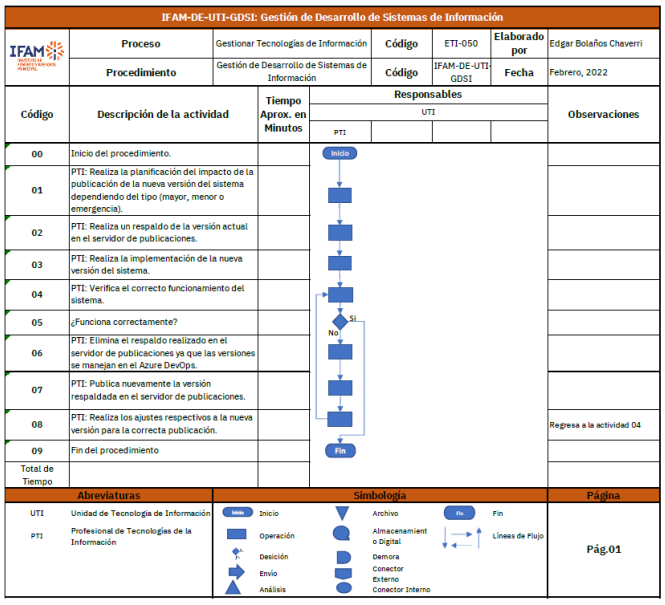


Ilustración 12: Publicación

ANEXO 6: RESULTADOS GUÍA DE ENTREVISTA INDIVIDUAL

Tabla 29: Resultados ANEXO 1

Pregunta	Colaborador 1	Colaborador 3	Colaborador 2	Colaborador 4
1. ¿Cuál es su experiencia en la organización y cuál es su rol actual?	Tengo 20 años de experiencia en esta organización.	He estado trabajando aquí por un tiempo y mi posición actual es en el área de Infraestructura.	Tengo más de 10 años de experiencia en esta organización y soy el Encargado de la Unidad de Tecnologías de Información.	Tengo más de 10 años en IFAM y soy profesional de Tecnologías de Información.
2. ¿Cuáles son los procesos actuales de entrega de software en la organización?	Utilizamos el modelo de desarrollo ágil Scrum para la entrega de software.	Creo que seguimos algún tipo de metodología ágil, pero no estoy seguro	Utilizamos metodologías ágiles como Scrum, para la entrega de software.	No estoy seguro de los procesos exactos o de alguna metodología.

		de los detalles específicos.		
3. ¿Cuáles son los principales desafíos que enfrenta en los procesos actuales de entrega de software?	Uno de los principales desafíos es la falta de colaboración entre los equipos.	Los plazos ajustados suelen ser un desafío en la entrega de software.	Uno de los principales desafíos es la falta de colaboración entre los equipos de desarrollo y operaciones.	La división entre equipos de trabajo.
4. ¿Cuáles son los procesos críticos en la entrega de software y por qué son importantes?	La planificación adecuada y la gestión de la calidad son procesos críticos desde mi punto de vista.	Supongo que la codificación y las pruebas son importantes, así como el pase a producción.	La planificación, el control de calidad y la gestión de configuración son procesos críticos para garantizar la entrega de software exitosa.	El control de calidad y la gestión de versiones.

5. ¿Ha identificado oportunidades de mejora en los procesos actuales de entrega de software?	Sí, hemos identificado la necesidad de mejorar la comunicación entre los equipos y el control de versiones.	No estoy seguro, no he estado involucrado en las discusiones sobre mejoras en los procesos de desarrollo al ser del área de infraestructura.	Sí, hemos identificado la necesidad de mejorar la automatización de pruebas y la integración continua.	Si, mejorar la comunicación y asignaciones.
6. ¿Qué herramientas y tecnologías utiliza actualmente para la entrega de software?	No utilizamos, se realiza un proceso de publicación manual para el proyecto de ingresos, conozco que otro software	Trabajamos con algunas herramientas de Microsoft, pero no estoy al tanto de todas ellas.	Utilizamos herramientas como el Azure DevOps.	No utilizamos, pero conozco que existen herramientas con Jira para este tipo de trabajo.

	institucional utiliza el Azure DevOps.			
7. ¿Ha trabajado anteriormente con una cultura DevOps? ¿Cuál fue su experiencia?	No.	No, pero he oído hablar de ella.	No, pero he oído hablar de ella.	No.
8. ¿Cómo cree que la implementación de una cultura DevOps podría mejorar los procesos actuales de entrega de software?	Mejorando la colaboración y comunicación entre equipos.	Mejorando la comunicación entre equipos.	La implementación de una cultura DevOps promovería una mayor colaboración entre los equipos de desarrollo y operaciones, lo que	Mejorando la colaboración entre equipos.

			agilizaría la entrega de software y mejoraría la calidad.	
9. ¿Qué cambios o ajustes deberían hacerse para implementar una cultura DevOps en la organización?	Procesos de capacitación.	Capacitación.	Sería necesario establecer equipos, fomentar la comunicación y adoptar herramientas.	Capacitación.
10. ¿Qué desafíos o barreras podrían surgir durante la implementación de una cultura DevOps en la organización?	Algunos desafíos podrían ser la resistencia al cambio y la necesidad de adquirir nuevas habilidades técnicas.	La resistencia al cambio.	Algunos desafíos pueden incluir la resistencia al cambio por parte de los empleados, la necesidad de adquirir nuevas habilidades	La resistencia al cambio.

		técnicas y la integración de herramientas y procesos existentes.

ANEXO 7: RESULTADOS GUIA DE GRUPOS FOCALES

1. Grupo 1

Tabla 30: Resultados ANEXO 2 grupo 1

Pregunta	Colaborador 1	Colaborador 4
1. ¿Cuáles son los principales desafíos que enfrentan en la implementación de una cultura DevOps en la organización?	Uno de los principales desafíos que enfrentamos es la resistencia al cambio por parte de algunos miembros del equipo de desarrollo. Algunos están acostumbrados a trabajar de manera tradicional y les resulta difícil adaptarse a los nuevos procesos y herramientas de DevOps. También	Otro desafío que enfrentamos es la falta de herramientas adecuadas para la implementación de DevOps. Necesitamos invertir en nuevas tecnologías y capacitar a nuestro personal para que puedan utilizarlas de manera efectiva.

	enfrentamos dificultades en la integración de las tareas de desarrollo y operaciones, ya que hay una falta de comunicación y colaboración efectiva entre los dos equipos.	
2. ¿Qué posibles soluciones y estrategias propondrían para superar estos desafíos?	Para superar la resistencia al cambio, sería importante realizar sesiones de capacitación y concientización para el equipo de desarrollo. También podríamos asignar mentores o líderes de proyecto que puedan guiar a los miembros del equipo en la adopción de prácticas DevOps. En cuanto a la integración entre desarrollo y operaciones, podríamos establecer reuniones regulares y fomentar una cultura de colaboración y comunicación abierta.	Podríamos explorar opciones de colaboración con proveedores externos que ofrezcan soluciones DevOps completas. Además, es fundamental que se comprenda los beneficios y el valor que una cultura DevOps puede aportar a la organización.

2. Grupo 2

Tabla 31: Resultados ANEXO 2 grupo 2

Pregunta	Colaborador 2	Colaborador 3
1. ¿Cuáles son los principales desafíos que enfrentan en la implementación de una cultura DevOps en la organización?	Uno de los principales desafíos que enfrentamos es la falta de claridad en los roles y responsabilidades dentro del equipo. A veces, no queda claro quién es responsable de qué tarea, lo que genera confusión y retrasos en los proyectos. También enfrentamos dificultades en la estandarización de los procesos, ya que cada equipo tiene sus propias prácticas y herramientas preferidas.	Otro desafío que enfrentamos es la resistencia al cambio por parte de algunos miembros del equipo. Algunos podrían considerar que la implementación de DevOps pueda afectar su forma de trabajar o incluso poner en peligro sus puestos de trabajo. Además, enfrentamos desafíos en la gestión del tiempo y los recursos, ya que la implementación de una cultura DevOps requiere tiempo adicional para la capacitación y la adopción de nuevas prácticas.
2. ¿Qué posibles soluciones y estrategias propondrían para	Para abordar la falta de claridad en los roles y responsabilidades, sería beneficioso establecer un marco claro de roles y responsabilidades dentro del equipo de	En relación con la resistencia al cambio, es importante brindar una comunicación clara y transparente sobre los beneficios y objetivos de la implementación de DevOps. Debemos

superar desafíos?	estos	trabajo. Esto podría incluir la designación de líderes de proyecto y la definición de las responsabilidades de cada miembro del equipo. Además, deberíamos fomentar la comunicación abierta y transparente entre los miembros del equipo para asegurarnos de que todos estén al tanto de las tareas y los plazos.	involucrar a los miembros del equipo en el proceso de toma de decisiones y proporcionarles oportunidades para expresar sus preocupaciones y sugerencias. Además, la capacitación y el apoyo continuo son fundamentales para asegurar que todos los miembros del equipo tengan las habilidades necesarias para adoptar las prácticas DevOps. En cuanto a la gestión del tiempo y los recursos, debemos asignar recursos adecuados y establecer un plan de implementación realista que tome en cuenta las tareas adicionales requeridas para la adopción de DevOps.

ANEXO 8: RESULTADOS FORMATO DE OBSERVACIÓN PARTICIPANTE

Fecha y hora de la observación: 09/06/2023 16:00

Lugar de la observación: IFAM sede central

Duración de la observación: 1 hora 15 minutos

Miembros del equipo observados:

Tabla 32: Resultados ANEXO 3

Nombre	Rol
Colaborador 1	Profesional Especialista UTI
Colaborador 3	Profesional UTI
Investigador	Profesional UTI

Procesos y prácticas observadas:

- Gestión de versiones y control de código fuente.
- Integración continua y despliegue continuo.
- Automatización de pruebas y calidad de software.
- Colaboración entre equipos de desarrollo y operaciones.

Observaciones y notas:

- En cuanto a la gestión de versiones y control de código fuente, se observó que existe una falta de estandarización en los procedimientos. No se implementa un sistema de control de versiones de manera sistemática solo algunos sistemas de información cuentan con su control en el Azure DevOps.

- En relación con la integración continua y el despliegue continuo, se notó que los procesos manuales predominan sobre la automatización. La implementación de pipelines de CI/CD es limitada, solo un sistema de información tiene el proceso automatizado siendo este el sistema institucional.
- En cuanto a la automatización de pruebas y calidad de software, se observó que las pruebas automatizadas son escasas y no se realizan de manera exhaustiva. Existe una falta de estrategia para abordar la calidad del software y se depende en gran medida de pruebas manuales.
- En relación con la colaboración entre equipos de desarrollo y operaciones, se percibe una falta de comunicación efectiva y una escasa colaboración entre ambos equipos. No se promueve el trabajo conjunto y la responsabilidad compartida en la entrega de software.

Problemas y obstáculos identificados durante la observación:

- Falta de estandarización en los procesos de gestión de versiones y control de código fuente.
- Limitada implementación de la integración continua y el despliegue continuo.
- Escasa automatización de pruebas y calidad de software.
- Deficiente colaboración entre equipos de desarrollo y operaciones.

ANEXO 9: ANÁLISIS DE RESULTADOS GUÍA DE ENTREVISTA INDIVIDUAL

Tabla 33: Análisis de resultados ANEXO 1

Pregunta	Análisis
1. ¿Cuál es su experiencia en la organización y cuál es su rol actual?	Se puede observar que hay una combinación de experiencia y roles tanto con la gestión de proyectos como con las tecnologías de información e infraestructura. Esto indica que el equipo cuenta con conocimientos y experiencia relevantes para la implementación de mejoras en los procesos de entrega de software.
2. ¿Cuáles son los procesos actuales de entrega de software en la organización?	Se puede destacar que los colaboradores 1 y 3 mencionan el uso de metodologías ágiles, en particular Scrum, lo que indica una orientación hacia enfoques iterativos e incrementales en la entrega de software. Por otra parte, la falta de certeza por parte de los colaboradores 2 y 4 sugiere una posible falta de alineación en cuanto a los procesos utilizados.
3. ¿Cuáles son los principales desafíos que enfrenta en los procesos actuales de entrega de software?	Se puede identificar que los desafíos mencionados por los colaboradores se centran en la colaboración, los plazos, la alineación entre equipos y la división del trabajo. Estos desafíos pueden tener un impacto negativo en la eficiencia y calidad de los procesos de entrega de software.

4. ¿Cuáles son los procesos críticos en la entrega de software y por qué son importantes?	Se identifica que los procesos críticos mencionados incluyen la planificación, el control de calidad, la gestión de configuración, la codificación, las pruebas y la gestión de versiones. Estos procesos son considerados importantes por los colaboradores, ya que influyen en la calidad y éxito de la entrega de software
5. ¿Ha identificado oportunidades de mejora en los procesos actuales de entrega de software?	Los colaboradores 1, 3 y 4 han identificado oportunidades de mejora relacionadas con la comunicación entre equipos, el control de versiones, la automatización de pruebas, la integración continua y las asignaciones. Estas áreas de mejora indican la necesidad de una mayor eficiencia y colaboración en los procesos de entrega de software.
6. ¿Qué herramientas y tecnologías utiliza actualmente para la entrega de software?	Se identifica que hay una falta de claridad en cuanto a las herramientas y tecnologías utilizadas en la entrega de software. Solo el colaborador 3 menciona específicamente el uso de Azure DevOps. Esto sugiere una posible falta de estandarización en las herramientas utilizadas y la necesidad de una evaluación más precisa de las herramientas que podrían mejorar los procesos de entrega de software.
7. ¿Ha trabajado anteriormente con una	Ninguno de los colaboradores ha tenido experiencia previa trabajando con una cultura DevOps. Aunque,

cultura DevOps? ¿Cuál fue su experiencia?	algunos han oído hablar de ella, lo que indica un nivel de desconocimiento con los conceptos asociados a DevOps.
8. ¿Cómo cree que la implementación de una cultura DevOps podría mejorar los procesos actuales de entrega de software?	Los colaboradores coinciden en que la implementación de una cultura DevOps podría mejorar la colaboración y comunicación entre los equipos. También se menciona que una cultura DevOps agilizaría la entrega de software, mejoraría la calidad y fomentaría una mayor colaboración entre los equipos de desarrollo y operaciones.
9. ¿Qué cambios o ajustes deberían hacerse para implementar una cultura DevOps en la organización?	Se indican procesos de capacitación para adquirir nuevas habilidades y conocimientos, así como el establecimiento de equipos multidisciplinarios, mejora en la comunicación y adopción de herramientas. Estos cambios reflejan una comprensión de que la implementación de una cultura DevOps requeriría un enfoque holístico y la adaptación de aspectos tanto técnicos como organizacionales.
10. ¿Qué desafíos o barreras podrían surgir durante la implementación de una cultura DevOps en la organización?	Se identifica la resistencia al cambio como un desafío común que podría surgir durante la implementación de una cultura DevOps. Adicionalmente, se menciona la necesidad de adquirir nuevas habilidades técnicas y la integración de herramientas a los procesos existentes como posibles barreras.

ANEXO 10: ANÁLISIS DE RESULTADOS GUIA DE GRUPOS FOCALES

1. Grupo 1

Tabla 34: Análisis de resultados ANEXO 2 grupo 1

Pregunta	Análisis
1. ¿Cuáles son los principales desafíos que enfrentan en la implementación de una cultura DevOps en la organización? 2. ¿Qué posibles soluciones y estrategias propondrían para superar estos desafíos?	El principal desafío mencionado es la resistencia al cambio por parte de algunos miembros del equipo de desarrollo. También se destaca la falta de comunicación y colaboración entre los equipos de desarrollo y operaciones, así como la falta de herramientas adecuadas para la implementación de DevOps. Para abordar la resistencia al cambio, se sugiere realizar sesiones de capacitación y concientización, asignar mentores o líderes de proyecto y fomentar una cultura de colaboración y comunicación abierta. Respecto a la falta de herramientas, se propone explorar opciones de colaboración con proveedores externos y enfatizar la comprensión de los beneficios de una cultura DevOps.

2. Grupo 2

Tabla 35: Análisis de resultados ANEXO 2 grupo 2

Pregunta	Análisis

1. ¿Cuáles son los principales desafíos que enfrentan en la implementación de una cultura DevOps en la organización? 2. ¿Qué posibles soluciones y estrategias propondrían para superar estos desafíos?	Se mencionan la falta de claridad en los roles y responsabilidades dentro del equipo, la dificultad en la estandarización de procesos y la resistencia al cambio por parte de algunos miembros del equipo. También se señalan los desafíos en la gestión del tiempo y los recursos debido a las tareas adicionales requeridas para la implementación de DevOps. Para abordar la falta de claridad en los roles y responsabilidades, se sugiere establecer un marco claro, designar líderes de proyecto y promover la comunicación abierta y transparente. Para superar la resistencia al cambio, se recomienda una comunicación clara sobre los beneficios y objetivos de DevOps, la participación de los miembros del equipo en la toma de decisiones, la capacitación y el apoyo continuo

ANEXO 11: ANÁLISIS DE RESULTADOS FORMATO DE OBSERVACIÓN PARTICIPANTE

Tabla 36: Análisis de resultados ANEXO 3

Actividad	Análisis
Procesos y prácticas observadas	1. Gestión de versiones y control de código fuente: Existe una falta de estandarización en los

procedimientos, lo que indica que no se sigue un enfoque consistente en la gestión de versiones y control de código fuente. La implementación del sistema de control de versiones se limita a algunos sistemas de información en el Azure DevOps.

2. Integración continua y despliegue continuo: Predominan los procesos manuales en lugar de la automatización en la integración continua y el despliegue continuo. Solo un sistema de información cuenta con un proceso automatizado mediante pipelines de CI/CD, mientras que en otros sistemas prevalecen los enfoques manuales.

3. Automatización de pruebas y calidad de software: Se observa una escasez de pruebas automatizadas y una dependencia significativa de las pruebas manuales. La falta de una estrategia sólida para abordar la calidad del software indica la necesidad de mejorar la automatización de pruebas y la implementación de herramientas de análisis estático de código y pruebas de rendimiento.

	4. Colaboración entre equipos de desarrollo y operaciones: Se percibe una falta de comunicación efectiva y colaboración entre los equipos de desarrollo y operaciones. No se promueve el trabajo conjunto y la responsabilidad compartida en la entrega de software.
Problemas y obstáculos identificados durante la observación	Se identificaron durante la observación la falta de estandarización en los procesos de gestión de versiones, la limitada implementación de la integración continua y el despliegue continuo, la escasa automatización de pruebas y calidad de software, y la deficiente colaboración entre equipos de desarrollo y operaciones.

ANEXO 12: LITERATURA COMPLEMENTARIA

Análisis para la adopción exitosa de una cultura DevOps

La adopción de una cultura DevOps no se puede realizar de forma inmediata. Requiere de un proceso iterativo de cambio en el cual tanto la tecnología como el personal de la organización deben ser preparados. Es fundamental aumentar la colaboración para lograr una aproximación a una cultura DevOps, una forma de lograr esto es trasladar el enfoque desde la integración continua (CI) hacia el desarrollo continuo (CD).

Revisión de literatura DevOps

La revisión de la literatura sobre DevOps ofrece una visión clara y profunda de este enfoque que se ha convertido en un paradigma clave para mejorar la colaboración y la eficiencia en los equipos de desarrollo de software. Esta sección explorará diversos estudios y publicaciones relevantes que abordan diferentes temas asociados con la implementación exitosa de una cultura DevOps.

Casos de éxito

La adopción de DevOps ha ganado popularidad en los últimos años como un enfoque para mejorar la eficiencia y la calidad en el desarrollo y despliegue de software. Los casos de éxito en la implementación de DevOps proporcionan ejemplos de cómo las organizaciones han logrado mejoras significativas en su rendimiento y resultados.

1. Netflix:

Netflix, Inc. es una empresa de entretenimiento y una plataforma de streaming estadounidense.

Según C. Aarón Cois (COIS, insights.sei.cmu.edu, 2015), Netflix es un caso de estudio fantástico para DevOps porque su proceso de ingeniería de software muestra una comprensión fundamental del pensamiento de DevOps y un enfoque en los atributos de calidad a través del proceso asistido por automatización. Los profesionales de DevOps adoptan un enfoque impulsado en los atributos de calidad para satisfacer las necesidades comerciales, aprovechando los procesos automatizados para lograr consistencia y eficiencia. El servicio de transmisión de Netflix es un gran sistema

distribuido alojado en Amazon Web Services (AWS). Dado que hay tantos componentes que tienen que trabajar juntos para proporcionar transmisiones de video confiables a los clientes en una amplia gama de dispositivos, los ingenieros de Netflix necesitaban concentrarse en los atributos de calidad de confiabilidad y solidez para los componentes del lado del servidor y del lado del cliente. En resumen, concluyeron que la única forma de sentirse cómodo manejando el fracaso es practicar constantemente el fracaso. Para lograr el nivel deseado de confianza y calidad, al más puro estilo DevOps, los ingenieros de Netflix se propusieron automatizar las fallas. Para lograr este resultado, Netflix modificó drásticamente su proceso de ingeniería mediante la introducción de una herramienta llamada Chaos Monkey, la primera de una serie de herramientas conocidas colectivamente como Netflix Simian Army. Chaos Monkey es básicamente un script que se ejecuta continuamente en todos los entornos de Netflix, causando caos al cerrar instancias de servidor al azar. Por lo tanto, mientras escriben código, los desarrolladores de Netflix operan constantemente en un entorno de servicios poco confiables e interrupciones inesperadas. Este caos no solo brinda a los desarrolladores una oportunidad única para probar su software en condiciones de fallas inesperadas, sino que también los incentiva a crear sistemas tolerantes a fallas para que su trabajo diario como desarrolladores sea menos frustrante. Esto es DevOps en su máxima expresión: alterar el proceso de desarrollo y usar la automatización para configurar un sistema donde la economía del comportamiento favorece la producción de un nivel deseable de calidad de software. En respuesta a la creación de software en este tipo de entorno, los desarrolladores de Netflix diseñarán sus sistemas para que sean modulares y comprobables.

2. Microsoft:

Microsoft Corporation es una empresa tecnológica multinacional estadounidense que produce software de computadora, productos electrónicos de consumo, computadoras personales y servicios relacionados, con sede en el campus de Microsoft ubicado en Redmond, Washington, Estados Unidos.

Según Microsoft (Microsoft, azure.microsoft.com, 2023), hace DevOps mediante:

- Colaboración entre equipos: Desglosar los silos entre los equipos es esencial para una adopción exitosa de DevOps. La comunicación, la visibilidad y la alineación con los objetivos es la forma en que los equipos optimizan la colaboración de DevOps y construyen mejores productos juntos.
- Adoptar una mentalidad de crecimiento: DevOps en Microsoft se trata de aprendizaje continuo. Los equipos deben cambiar la forma en que trabajan, adoptar nuevos procesos y ver el fracaso como una oportunidad para aprender. El viaje nunca termina.
- Permitiendo el cambio a través de la tecnología: Los equipos de Microsoft confían en las mejores herramientas disponibles. Azure nos permite operar de manera confiable la infraestructura a escala y automatizar los procesos. El uso y la contribución al código abierto acelera nuestra capacidad de innovar.

Los casos de éxito en la implementación de DevOps demuestran que esta metodología puede generar resultados positivos en términos de eficiencia, calidad y

satisfacción del cliente. Empresas líderes como Netflix y Microsoft han obtenido ventajas competitivas al adoptar DevOps.

Cultura DevOps

Según AWS (AWS, 2023), DevOps consiste en eliminar las barreras entre dos equipos que anteriormente estaban aislados, el de desarrollo y el de operaciones. En algunas organizaciones, es posible que no existan equipos de desarrollo y operaciones distintos y que los ingenieros se encarguen de todo. Con DevOps, los dos equipos colaboran para optimizar la productividad de los desarrolladores y la confiabilidad de las operaciones. Se esfuerzan por comunicarse con frecuencia, incrementar la eficacia y mejorar la calidad de los servicios que proporcionan a los clientes. Se hacen totalmente responsables de sus servicios, a menudo más allá de lo que tradicionalmente abarcarían sus funciones o puestos de trabajo, al pensar en las necesidades de los clientes y cómo pueden ayudar a satisfacerlas. Los equipos de control de calidad y seguridad también podrían integrarse estrechamente en estos equipos. Independientemente de su estructura organizativa, las organizaciones que utilizan un modelo de DevOps disponen de equipos que visualizan todo el ciclo de vida de desarrollo y de la infraestructura como parte de sus responsabilidades.

Según Microsoft (Microsoft, azure.microsoft.com, 2023), Aunque la adopción de prácticas de DevOps automatiza y optimiza los procesos con tecnología, todo comienza con la cultura interna de la organización y con las personas que participan en ella. El

desafío de cultivar una cultura de DevOps requiere cambios profundos en la forma en la que las personas trabajan y colaboran. Pero cuando las organizaciones se comprometen a implementar una cultura de DevOps, pueden crear un entorno que facilite el desarrollo de equipos de alto rendimiento mediante:

- Colaboración, visibilidad y alineación: Una buena cultura de DevOps se distingue, entre otras cosas, por la colaboración entre los equipos, que comienza con la visibilidad. Diferentes equipos, como el de desarrollo o el de operaciones de TI, deben compartir entre sí sus procesos de DevOps, sus prioridades y sus preocupaciones. Estos equipos también deben planificar juntos el trabajo y alinear sus objetivos y los indicadores del éxito en relación con el negocio.
- Cambios en el ámbito y en la responsabilidad: A medida que se alinean, los equipos asumen y participan en más fases del ciclo de vida, no solo las que son principales para su rol. Por ejemplo, los desarrolladores asumen responsabilidad no solo por la innovación y la calidad establecidas en la fase de desarrollo, sino también por el rendimiento y la estabilidad que sus cambios producen en la fase de uso. Al mismo tiempo, los operadores de TI se aseguran de incluir la gobernanza, la seguridad y el cumplimiento normativo en las fases de planificación y desarrollo.
- Ciclos de lanzamiento de versiones más cortos: Los equipos de DevOps mantienen la agilidad porque lanzan versiones de software en ciclos cortos. Los ciclos de lanzamiento de versiones más cortos facilitan la planificación y la administración de los riesgos, porque el progreso es incremental, lo que reduce el

impacto en la estabilidad del sistema. El acortamiento de los ciclos de lanzamiento de versiones permite también a las organizaciones adaptarse y reaccionar a las necesidades cambiantes de los clientes y a la presión competitiva.

- Aprendizaje continuo: Los equipos de DevOps de alto rendimiento establecen una mentalidad de crecimiento. Aceptan el fracaso y responden rápido a los errores, e incorporan lo que aprenden a los procesos, de modo que mejoran continuamente, aumentan la satisfacción del cliente, y agilizan la innovación y la capacidad de adaptación al mercado de forma constante. DevOps es un recorrido, por lo que siempre hay espacio para crecer.

Lo mencionado por AWS y Microsoft demuestran que para implementar una cultura DevOps de manera efectiva, es necesario adoptar una cultura interna que promueva la colaboración y comunicación. Con esto los equipos DevOps asumen responsabilidades más amplias en todo el ciclo de vida del desarrollo del software, y trabajan en ciclos de lanzamiento de versiones más cortos y así responder rápidamente a las necesidades del cliente y la competencia.

Mejores prácticas

Según Manuel Cubillo en blog de Encora.com (Cubillo, 2020), una serie de prácticas clave para la buena implementación de una cultura DevOps son:

- Pruebas Automatizadas Continuas: Uno de los aspectos clave de DevOps es garantizar la calidad desde el inicio. Correr pruebas lo antes posible y tan menudo

como sea posible es la única forma de garantizar un código de calidad. Esto no se puede lograr sin la automatización de pruebas: la práctica de aplicar herramientas y frameworks para crear un script que automatiza las pruebas comunes (aquellas que verifican la funcionalidad del software). Al automatizarlas, éstas se pueden correr en cualquier momento del proceso. Ejecutar pruebas automatizadas constantemente permite que cualquier problema se solucione apenas sea detectado, lo cual significa que la calidad del código será mucho mejor.

- Integración Continua: Como es usual en DevOps, todas las prácticas se complementan y se refuerzan mutuamente. La integración continua (CI por sus siglas en inglés) combina un conjunto de herramientas, premisas y prácticas que aceleran la entrega de software de alta calidad. Se eliminan los embotellamientos (como, por ejemplo, las pruebas manuales) y se minimiza el riesgo de que surjan problemas técnicos en las últimas fases del ciclo de lanzamiento. Con CI, los equipos construyen el software y corren pruebas unitarias cada vez que un desarrollador añade código nuevo al repositorio. Así, acortar los ciclos de retroalimentación se hace más fácil, agilizando las respuestas a las demandas del mercado y erradicando errores rápidamente. CI les permite a los desarrolladores crear soluciones de primer nivel en una serie de pasos cortos y rutinarios. Al brindar retroalimentación instantánea, el código se revisa y se modifica regularmente. Una vez más, la calidad queda garantizada desde el inicio.
- Despliegue Continuo: Con un despliegue continuo, cualquier cambio que se le haga al código pasa por todo el proceso y se promueve sistemáticamente al

siguiente ambiente de prueba, donde la integración sucede automáticamente. Por lo general, esto continúa a través de todo el proceso hasta que en algún momento se requiere de una aprobación manual (generalmente durante la transición entre desarrollo y operaciones). Es este paso manual que el despliegue continuo revoluciona: lo automatiza. Esto significa que cualquier versión actualizada del software que funciona pasa a producción sin ningún tipo de interacción humana. Gracias al despliegue continuo, desarrollo puede disminuir el tiempo que transcurre entre cambiar/añadir una funcionalidad al código y su despliegue a producción. En otras palabras, la capacidad de respuesta del negocio mejora.

- Monitoreo Continuo: Cuando un software pasa a la etapa de producción, es el equipo de operaciones el que se hace cargo. Se aseguran de que el software funcione correctamente. Para lograrlo, deben de vigilar el ambiente en todo momento, así garantizando su estabilidad. El monitoreo continuo es un proceso que constantemente modifica y ajusta las herramientas de monitorización (o su configuración). Aquí es donde, una vez más, la automatización es crucial: es el pilar de esta práctica. Les permite a operaciones mantener actualizada la configuración en medio de los crecientes cambios que se pueden dar dentro de una infraestructura.

Según AWS (AWS, 2023), las siguientes son prácticas recomendadas de DevOps:

- Integración continua: La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código

en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. Los objetivos clave de la integración continua consisten en encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones de software.

- Entrega continua: La entrega continua es una práctica de desarrollo de software mediante la cual se compilan, prueban y preparan automáticamente los cambios en el código y se entregan a la fase de producción. Amplía la integración continua al implementar todos los cambios en el código en un entorno de pruebas o de producción después de la fase de creación. Cuando se la entrega continua se implementa de manera adecuada, los desarrolladores dispondrán siempre de un artefacto listo para su implementación que se ha sometido a un proceso de pruebas estandarizado.
- Microservicios: La arquitectura de microservicios es un enfoque de diseño que sirve para crear una sola aplicación como conjunto de servicios pequeños. Cada servicio se ejecuta en su propio proceso y se comunica con otros servicios mediante una interfaz bien definida utilizando un mecanismo ligero, normalmente una interfaz de programación de aplicaciones basada en HTTP (API). Los microservicios se crean en torno a las capacidades empresariales. Cada servicio abarca un único propósito. Puede utilizar distintos marcos o lenguajes de programación para escribir microservicios e implementarlos independientemente, como servicio único, o como grupo de servicios.

- Infraestructura como código: La infraestructura como código es una práctica mediante la que se aprovisiona y administra infraestructura con técnicas de desarrollo de código y de software, como el control de versiones y la integración continua. El modelo orientado a la API de la nube permite a los desarrolladores y administradores de sistemas interactuar con la infraestructura mediante programación y a escala, en lugar de configurar y ajustar recursos manualmente. Así, los ingenieros pueden interactuar con la infraestructura con herramientas basadas en código y tratar la infraestructura de un modo parecido a como tratan el código de la aplicación. Como están definidos por el código, la infraestructura y los servidores se pueden implementar con rapidez con patrones estandarizados, actualizar con las últimas revisiones y versiones o duplicar de forma repetible.
- Monitoreo y registro: Las organizaciones monitorean métricas y registros para ver cómo el desempeño de las aplicaciones y la infraestructura afecta a la experiencia que el usuario final tiene de su producto. Cuando recopilan, categorizan y analizan los datos y registros generados por las aplicaciones y la infraestructura, las organizaciones pueden entender cómo los cambios y actualizaciones afectan a los usuarios, esto les aporta información sobre la causa raíz de los problemas o cambios inesperados. El monitoreo activa se vuelve cada vez más importante, ya que los servicios deben estar disponibles las 24 horas del día, los 7 días de la semana, a medida que la frecuencia de actualizaciones de las aplicaciones e infraestructura incrementa. La creación de alertas y el análisis en tiempo real de los datos también ayuda a las organizaciones a monitorear sus servicios de forma proactiva.

- Comunicación y colaboración: El incremento en la comunicación y la colaboración en una organización es uno de los aspectos culturales clave de DevOps. El uso de las herramientas de DevOps y la automatización del proceso de entrega de software establece la colaboración al reunir físicamente los flujos de trabajo y las responsabilidades de los equipos de desarrollo y operaciones. Además, estos equipos establecen normas culturales sólidas que giran en torno a compartir información y facilitar la comunicación mediante el uso de aplicaciones de chat, sistemas de seguimiento de proyectos o problemas y wikis. De este modo, se acelera la comunicación entre los equipos de desarrollo y operaciones e incluso con otros equipos, como marketing y ventas, lo que permite que todos los departamentos de la organización se alineen mejor con los objetivos y proyectos.

Según Microsoft (Microsoft, azure.microsoft.com, 2023), Más allá del establecimiento de una cultura de DevOps, los equipos ponen en práctica el método DevOps implementando determinadas prácticas a lo largo del ciclo de vida de las aplicaciones. Algunas de estas prácticas ayudan a agilizar, automatizar y mejorar una fase específica. Otras abarcan varias fases y ayudan a los equipos a crear procesos homogéneos que favorezcan la productividad:

- Integración y entrega continuas (CI/CD): La administración de la configuración hace referencia a la administración del estado de los recursos de un sistema, incluidos los servidores, las máquinas virtuales y las bases de datos. El uso de herramientas de administración de la configuración permite a los equipos distribuir

cambios de un modo controlado y sistemático, lo que reduce el riesgo de modificar la configuración del sistema. Los equipos utilizan herramientas de administración de la configuración para hacer un seguimiento del estado del sistema y evitar alteraciones en la configuración, que es como se desvía la configuración de un recurso del sistema a lo largo del tiempo del estado definido para él. Al usarla junto con la infraestructura como código, resulta fácil elaborar plantillas y automatizar la definición y la configuración de sistemas, lo que permite a los equipos usar entornos complejos a escala.

- Control de versiones: El control de versiones es la práctica de administrar el código por versiones, haciendo un seguimiento de las revisiones y del historial de cambios para facilitar la revisión y la recuperación del código. Esta práctica suele implementarse con sistemas de control de versiones, como Git, que permite que varios desarrolladores colaboren para crear código. Estos sistemas proporcionan un proceso claro para fusionar mediante combinación los cambios en el código que tienen lugar en los mismos archivos, controlar los conflictos y revertir los cambios a estados anteriores. El uso del control de versiones es una práctica de DevOps fundamental que ayuda a los equipos de desarrollo a trabajar juntos, dividir las tareas de programación entre los miembros del equipo y almacenar todo el código para poder recuperarlo fácilmente si fuese necesario. El control de versiones es también un elemento necesario en otras prácticas, como la integración continua y la infraestructura como código.
- Desarrollo ágil de software: El método ágil es un enfoque de desarrollo de software que hace hincapié en la colaboración en equipo, en los comentarios de los clientes

y usuarios, y en una gran capacidad de adaptación a los cambios mediante ciclos cortos de lanzamiento de versiones. Los equipos que practican la metodología ágil proporcionan mejoras y cambios continuos a los clientes, recopilan sus comentarios y, después, aprenden y ajustan el software en función de lo que el cliente quiere y necesita. El método ágil es muy diferente a otros marcos más tradicionales, como el modelo en cascada, que incluye ciclos de lanzamiento de versiones largos definidos por fases secuenciales. Kanban y Scrum son dos marcos populares asociados al método ágil.

- Infraestructura como código: La infraestructura cómo código define las topologías y los recursos del sistema de un modo descriptivo que permite a los equipos administrar esos recursos igual que lo harían con el código. Las diferentes versiones de esas definiciones se pueden almacenar en sistemas de control de versiones, donde se pueden revisar y revertir, de nuevo, igual que el código. La práctica de la infraestructura como código permite a los equipos implementar recursos del sistema de un modo confiable, repetible y controlado. Además, la infraestructura como código ayuda a automatizar la implementación y reduce el riesgo de errores humanos, especialmente en entornos complejos de gran tamaño. Esta solución repetible y confiable para la implementación de entornos permite a los equipos mantener entornos de desarrollo y pruebas que sean idénticos al entorno de producción. De igual modo, la duplicación de entornos en otros centros de datos y en plataformas en la nube es más sencilla y más eficiente.
- Administración de configuración: La administración de la configuración hace referencia a la administración del estado de los recursos de un sistema, incluidos

los servidores, las máquinas virtuales y las bases de datos. El uso de herramientas de administración de la configuración permite a los equipos distribuir cambios de un modo controlado y sistemático, lo que reduce el riesgo de modificar la configuración del sistema. Los equipos utilizan herramientas de administración de la configuración para hacer un seguimiento del estado del sistema y evitar alteraciones en la configuración, que es como se desvía la configuración de un recurso del sistema a lo largo del tiempo del estado definido para él. Al usarla junto con la infraestructura como código, resulta fácil elaborar plantillas y automatizar la definición y la configuración de sistemas, lo que permite a los equipos usar entornos complejos a escala.

- Supervisión continua: La supervisión continua significa tener visibilidad total y en tiempo real del rendimiento y el estado de toda la pila de aplicaciones, desde la infraestructura subyacente donde se ejecutan las aplicaciones hasta los componentes de software de niveles superiores. La visibilidad consiste en la recopilación de datos de telemetría y metadatos, así como en el establecimiento de alertas para condiciones predefinidas que garanticen la atención de un operador. La telemetría incluye registros y datos de eventos recopilados de varias partes del sistema que se almacenan dónde pueden analizarse y consultarse. Los equipos de DevOps de alto rendimiento se aseguran de establecer alertas útiles que les permitan tomar medidas y recopilan datos de telemetría muy completos para obtener conclusiones a partir de enormes cantidades de datos. Estas conclusiones ayudan a los equipos a mitigar los problemas en tiempo real y a ver cómo mejorar la aplicación en futuros ciclos de desarrollo.

En base a lo indicado por expertos y proveedores de servicios tecnológicos, se pueden mencionar las siguientes prácticas como esenciales en el proceso de implementación de una cultura DevOps:

1. Integración Continua (CI): La integración continua implica combinar los cambios en el código de manera frecuente y ejecutar pruebas unitarias. Esto acelera la entrega de software de alta calidad, elimina cuellos de botella y minimiza los problemas técnicos en las etapas finales del ciclo de lanzamiento.

2. Despliegue Continuo (CD): El despliegue continuo automatiza el proceso de promover los cambios en el código al siguiente entorno de prueba y, eventualmente, a producción. Permite una entrega rápida y sin intervención humana, mejorando la capacidad de respuesta del negocio.

3. Monitoreo: El monitoreo garantiza la estabilidad del software en producción. A través de la automatización, se ajustan y modifican constantemente las herramientas de monitoreo, lo que permite detectar y solucionar problemas en tiempo real.

Herramientas

Según CertiProf (CertiProf, 2023), en su material de estudio para la certificación DevOps Essentials Profesional Certificate. El uso de herramientas no es hacer DevOps,

hacer DevOps sin el uso de herramientas es complejo si entendemos que DevOps tiene componentes de automatización, DevOps es y será un tema cultural y no será tan solo el conjunto de herramientas, la herramienta por más buena que sea no logrará hacer que la cultura cambie, sin herramientas adecuadas, la fusión de roles y deberes de DevOps puede crear caos y resultados no deseados que incluyen problemas relacionados con escalabilidad, confiabilidad y administración de carga. Las siguientes herramientas constituyen un apoyo para la cultura DevOps:

1. GIT: Fue producido siguiendo los requerimientos de la comunidad Linux para SCM, es decir, el software Source-Control-Management que podría soportar sistemas distribuidos. Esta es la herramienta más comúnmente utilizada para el manejo de fuentes que está disponible hoy en día.

2. Cloud: La infraestructura de la nube juega junto con las herramientas de DevOps para la automatización y orquestación de despliegues de aplicaciones, además coordina el apoyo a los procesos deseados.

3. Docker: Docker aporta portabilidad a las aplicaciones a través de la tecnología de contenedores, donde las aplicaciones pueden ejecutarse en unidades autónomas que se mueven a través de las plataformas.

4. JS: Las plataformas JS tienen bibliotecas que permiten a las aplicaciones servir como servidores web sin necesidad de software como el Apache-HTTP-Server o

el Microsoft-IIS, sin tener que hacer que los servidores web acorten las listas de tareas pendientes de DevOps y optimicen el flujo de código desde el desarrollo hasta el despliegue. Además, el JavaScript se puede utilizar tanto en cliente como en servidor.

5. Chef: Las herramientas Chef DevOps proporcionan marcos para automatizar y administrar la infraestructura a través de DSL simple. Chef pone una confianza en sus definiciones reutilizables llamados libros de cocina y recetas escritas en Ruby.

6. Jenkins: Este es un motor para la integración extensible y continua que se utiliza como una de las principales herramientas de los ingenieros de DevOps que desean monitorear repetidas ejecuciones de trabajo.

7. Puppet: Puppet permite la gestión de la configuración y del software durante la realización de cambios rápidos y repetibles. Puppet impone automáticamente la coherencia en entornos y puede trabajar en máquinas físicas y virtuales. Tiene cadenas de herramientas comunes y respalda las mejores prácticas clave en DevOps que incluyen la entrega continua.

Según AWS (AWS, 2023), Estas herramientas automatizan tareas manuales, ayudan a los equipos a administrar entornos complejos a escala y mantienen a los ingenieros en control de la gran velocidad que permite alcanzar DevOps:

1. Integración y entrega continua: Permiten almacenar y versionar el código fuente de su aplicación de forma segura y crear, probar e implementar su aplicación automáticamente en AWS o su entorno en las instalaciones.
    a. AWS CodePipeline
    b. AWS CodeBuild
    c. AWS CodeDeploy
    d. AWS CodeStar

2. Microservicios: Permite implementar una arquitectura de microservicios con contenedores o informática sin servidor.
    a. Amazon Elastic Container Service
    b. AWS Lambda

3. Infraestructura como código: Permite aprovisionar, configurar y administrar sus recursos de la infraestructura de AWS con código y plantillas. Monitorice y garantice la conformidad de la infraestructura:
    a. AWS CloudFormation
    b. AWS OpsWorks
    c. AWS Systems Manager
    d. AWS Config

4. Monitoreo y registro: Permite registrar logs y monitorear el desempeño de la aplicación y la infraestructura casi en tiempo real:

a. Amazon CloudWatch
b. AWS X-Ray
c. AWS CloudTrail
d. Amazon DevOps Guru

5. Plataforma como servicio: Permite implementar aplicaciones web sin la necesidad de aprovisionar y administrar la pila de infraestructura y aplicaciones.
    a. AWS Elastic Beanstalk

6. Control de versiones: Permite alojar repositorios Git seguros y altamente escalables en la nube:
    a. AWS CodeCommit

Según Microsoft (Microsoft, azure.microsoft.com, 2023), Azure DevOps dispone de una integración sólida con Azure y un completo conjunto de tecnologías que ayudan a entregar software de forma segura y rápida:

1. Integración y entrega continua: Desde el código hasta la nube, automatiza cada parte del proceso de DevOps con integración continua y entrega continua:
    a. Azure Pipelines
    b. Github Actions
    c. Jenkins
    d. Spinnaker

2. Control de Versiones:
    a. Git
    b. GitHub
    c. Azure Repos

3. Infraestructura como código: Permite definir los recursos en la nube de un modo descriptivo para que los entornos se versionen y confirmen en conjunto con el código de las aplicaciones en los repositorios. Tratar la infraestructura como código te permite implementar recursos en la nube de un modo replicable y fiable, lo cual garantiza la gobernanza total de los entornos en la nube:
    a. Azure Bluepints
    b. Terraform
    c. Ansible

4. Administración de la configuración: Permite administrar la configuración de los recursos en todo el sistema para aplicar los estados deseados, distribuir actualizaciones de configuración y automatizar la resolución de cambios y problemas imprevistos.
    a. Azure Automation
    b. Ansible
    c. Chef
    d. Puppet

5. Monitoreo o Supervisión: Supervisa el estado de la infraestructura e integra la información en los paneles existente:
    a. Grafana
    b. Kibana
    c. Azure Monitor
    d. Azure Boards

Según el libro DevOps for beginner de Craig Berg (Berg, 2020), el flujo de trabajo de DevOps utiliza muchas herramientas. Algunas herramientas populares utilizadas en DevOps:

1. Puppet: Ofrece funciones como la entrega y publicación de los cambios de forma rápida y frecuente.

2. Ansible: Permite la implementación de código automatizado, la orquestación dentro del servicio, el aprovisionamiento en la nube y que tiene otros servicios y herramientas de tecnología de la información esenciales.

3. Docker: Es una popular herramienta DevOps que utiliza contenedores como base de operaciones. Es una herramienta de alto nivel que permite a los desarrolladores crear, mover y ejecutar aplicaciones distribuidas en todos los sistemas.

4. Nagios: Es una de las herramientas más útiles en las operaciones de DevOps, su utilidad es especialmente en funciones como la determinación de errores y corrección de errores utilizando herramientas como la red, infraestructura, monitores de registro y sistemas de servidor.

5. Chef: Es una herramienta de código abierto y basada en la nube que utiliza la codificación Ruby para desarrollar bloques de construcción esenciales. Su principal uso es en la automatización de infraestructuras, donde ayuda a reducir las tareas manuales y repetitivas.

6. Jenkins: Es una herramienta utilizada para monitorear la ejecución de tareas repetitivas dentro de un entorno; permite la integración continua dentro de DevOps.

7. Git: La herramienta más popular dentro de la comunidad de desarrollo es un sistema de control de versiones distribuido de código abierto.

8. Splunk: Hace que los datos de la máquina sean accesibles y utilizables por todos los usuarios autorizados, brinda inteligencia de las operaciones del entorno para usar Devops.

9. Selenium: Es un marco DevOps portátil gratuito y de código abierto que se utiliza para probar marcos web, proporciona una interfaz interactiva fácil de usar para desarrollar pruebas automáticas.

Según el libro DevOps guía completa para principiantes aprende DevOps paso a paso de Ethan Thorpe (Thorpe), la implementación de DevOps correctamente significa que se debe crear un determinado entorno de TI en DevOps antes de poder usar las diferentes herramientas de DevOps:

1. Contenerización y Orquestación: Permite envolver software en un sistema de archivos completo que contendrá todo para implementaciones de software rápidas y completamente automatizadas dentro de un entorno más estable.
    a. Docker
    b. Rkt
    c. Linux Conteiners LXC
    d. AWS Elastic Container Service
    e. Docker Swarm

2. Pruebas: Permite facilitar la entrega continua y la integración continua a medida que automatiza los procesos repetibles.
    a. Solano Labs
    b. Gradle
    c. Jenkins

3. Gestión de configuración: Permite que los equipos que trabajna en la placa declaren fácilmente y automaticen la configuración general del sistema con el fin de disfrutar de una mejor:
    a. Chef
    b. AWS CloudFormation
    c. Puppet
    d. Ansible
    e. Terraform

4. Implementación continua: Permite automatizar el proceso general de creación de nuevas aplicaciones y software:
    a. Capistrano

5. Monitoreo: Permite que sus equipos digitales midan fácilmente el impacto general de esos ciclos de lanzamiento de software frecuentes a través de emultilples entornos.
    a. PagerDuty
    b. Sensu
    c. Blue Medora
    d. VictorOps
    e. New Relic
    f. XMatters

6. Supervisión y seguridad; Facilitan la gestión de todos los servicios, asi como mantener software a salvo de diferentes amenazas:
    a. Monic
    b. Upstar
    c. Blue Pill
    d. Snort
    e. Sqreen
    f. Evident.ip

7. Colaboración y planificación: Permite a los equipos disfrutar de una colaboración y comunicación transparestes:
    a. Slack
    b. Service Now
    c. Jira

En base a lo indicado por expertos y proveedores de servicios tecnológicos, el uso de herramientas es importante en la implementación de una cultura DevOps, pero se debe destacar que las herramientas por sí solas no constituyen DevOps, sino que son un apoyo para facilitar la automatización, la colaboración y la entrega continua.

DevOps es principalmente un enfoque cultural que busca la integración entre los equipos de desarrollo y operaciones para lograr una entrega de software más rápida y

confiable, donde las herramientas deben combinarse con la cultura para obtener los mejores resultados.

Finalmente, se puede identificar que las herramientas más utilizadas incluyen herramientas de versionamiento, herramientas de integración y entrega continua, herramientas de gestión de configuración y herramientas de monitoreo.

Roles y responsabilidades

Según Microsoft (learn.microsoft.com, 2023), DevOps permite la coordinación y la colaboración entre roles anteriormente aislados como desarrollo, operaciones de TI, ingeniería de calidad y seguridad. Los equipos adoptan la cultura, las prácticas y las herramientas de DevOps para aumentar la confianza en las aplicaciones que crean, responder mejor a las necesidades de los clientes y alcanzar los objetivos comerciales más rápido. DevOps ayuda a los equipos a proporcionar valor a los clientes continuamente al producir productos mejores y más confiables.

Según Johanna Ambrosio en el sitio web de GitLab.com (Ambrosio, 2022), Contratar los roles de DevOps correctos puede requerir una combinación de arte, ciencia y suerte, pero es factible:

1. Desarrolladores: DevOps es un deporte de equipo hoy en día. Los desarrolladores prueban el código, actúan como campeones de seguridad, aprovisionan infraestructura y escriben scripts de automatización.

2. Ingeniero de operaciones/administrador de sistemas: en los tiempos antiguos, esta es la persona que se aseguraba de que el software pudiera ejecutarse sin problemas en producción y enviaba alarmas si no lo hacía. Pero en un equipo de DevOps, los operadores administrarán la nube, ayudarán a crear monitoreo y análisis integrados en el código, administrarán las herramientas, se ocuparán de las herramientas y, por supuesto, ayudarán a resolver problemas.

3. Evangelista: Alguien debe asegurarse de que el resto de la empresa sepa lo que está haciendo su equipo, cantar sus alabanzas y comunicar cuáles son las necesidades más apremiantes de la empresa. Idealmente, esta es una persona de alto nivel que se sienta en el Comité Ejecutivo o la junta directiva de la empresa. Más que un simple animador, un evangelista en un equipo de DevOps debe hacer que todos en la empresa se involucren en DevOps, se comprometan con su éxito y estén felices de gastar el presupuesto en el esfuerzo.

4. Gerente de proyecto/gerente de lanzamiento: este rol de DevOps realiza un seguimiento del progreso del equipo en relación con los objetivos comerciales, establece metas y cronogramas, e intenta que todo funcione a tiempo.

5. Probador de control de calidad/ingeniero de automatización: un profesional de pruebas desempeña un papel fundamental en un equipo de DevOps, incluso con la llegada de los "desarrolladores que prueban" y la automatización de pruebas.

6. Ingeniero de seguridad: es fundamental incorporar la seguridad y el cumplimiento desde el principio, en lugar de intentar agregarlo al final, cuando solucionar los problemas se vuelve más costoso. Un ingeniero de seguridad en un equipo de DevOps debe ser estratégico y práctico.

7. Profesional de la experiencia del usuario (UX): este rol de DevOps es el defensor del usuario final, la persona que está totalmente enfocada en cómo se ve y funciona el software desde la perspectiva del cliente.

Según CertiProf (CertiProf, 2023), en su material de estudio para la certificación DevOps Essentials Profesional Certificate, se debe contar con un equipo DevOps conformado por:

- Master de Procesos
- Master de Servicios
- Ingeniero de DevOps
- Coordinador de Lanzamiento
- Ingeniero de Confiabilidad
- Equipo de Desarrollo
- Equipo de Operación.

Según DEVOPS-CERTIFICATION.ORG (devops-certification.org, 2023):

1. Generalista DevOps: La función principal de un generalista de DevOps es garantizar el establecimiento sin problemas, el progreso eficiente y saludable y la mejora continua de las prácticas de DevOps en una organización de DevOps y en sus equipos de DevOps. Por lo tanto, la competencia y la perspectiva de cada empleado de una organización DevOps para poder actuar con equipos DevOps es un factor fundamental que determina el nivel de éxito y la vida útil de las organizaciones DevOps.

2. Ejecutivo DevOps: Los ejecutivos de DevOps son responsables de la utilización y aplicación exitosas de los conocimientos de DevOps dentro de sus organizaciones. Son jugadores clave para que los equipos de DevOps permitan el cambio cultural de hacer y pensar al estilo DevOps. Tienen una capacidad comprobada para influir constructivamente en los equipos y la administración para hacer las cosas. Los ejecutivos de DevOps respaldan la alineación de los equipos de DevOps con las estrategias comerciales y son responsables de la identificación, selección, determinación del alcance, priorización y, en última instancia, gestión de la penetración de DevOps en sus organizaciones.

3. Gerente de Proyectos DevOps: DevOps Project Manager es la persona responsable de lograr los objetivos establecidos del proyecto. Las responsabilidades clave del administrador de proyectos de DevOps incluyen la creación de objetivos de proyecto claros y alcanzables, la creación de los

requisitos del proyecto y la gestión de las restricciones del triángulo de gestión de proyectos, que son el costo, el cronograma, el alcance y la calidad.

4. Propietario del producto DevOps: El rol de Propietario de producto de DevOps es un rol único y amplio en DevOps que combina todos los aspectos desafiantes de los roles tradicionales de Gerente de proyecto y Gerente de producto. Además, el propietario del producto DevOps representa el punto de vista del cliente en un equipo DevOps al garantizar que se realice el trabajo correcto en el momento adecuado. Debe estar estrechamente integrado con los equipos y procesos generales de desarrollo y entrega de software DevOps para garantizar el máximo valor agregado para todas y cada una de las versiones del producto.

5. Arquitecto DevOps: Un Devops Architect posee la arquitectura, el diseño y el desarrollo de herramientas y procesos de implementación de productos. En este rol, se espera que DevOps Architect diseñe y desarrolle soluciones innovadoras para construir y mantener la arquitectura del producto, sus herramientas y procesos relacionados para la integración continua y la canalización de implementación continua.

6. Desarrollador DevOps: Como desarrollador de DevOps, transforma los objetivos comerciales de sus clientes en soluciones y sistemas de software. La principal diferencia entre un desarrollador ordinario y un desarrollador de DevOps es que: Como desarrollador de DevOps, durante todo el curso de su trabajo, es consciente

de los objetivos comerciales y las demandas comerciales de sus clientes. Es plenamente consciente de por qué su empleador lo ha contratado y de lo que sus clientes necesitan y esperan de usted.

7. Ingeniero de Operaciones DevOps: Los ingenieros de operaciones de DevOps son responsables de monitorear, mantener e implementar el software y la infraestructura de última generación detrás de la tecnología de sus productos. Implementan y mantienen Infraestructuras de Red y Servidores en Centros de Datos. También participan en los equipos de implementación y entrega de DevOps en las instalaciones y desarrollan planes de contingencia de implementación y entrega de productos.

8. Ingeniero de control de calidad DevOps: Los ingenieros de garantía de calidad de DevOps desempeñan un papel proactivo en el procesamiento de puntos de venta únicos de su producto, requisitos, casos de uso, arquitectura de software y otros materiales de diseño de software para descubrir los tipos de prueba deseados para validar la calidad del producto bajo prueba. Trabajan con los desarrolladores de DevOps y los gerentes de proyectos de DevOps para determinar las metodologías y herramientas de implementación de pruebas para ejecutar y operar el trabajo de prueba.

9. Ingeniero de seguridad de la información DevOps: En las configuraciones tradicionales de las organizaciones de TI, la seguridad de la información es en

gran medida una idea de último momento. Es otro requisito no funcional que a menudo se atiende cuando es más difícil, costoso y agitado identificar y solucionar los problemas.

10. Administrador de versiones de DevOps: Los gerentes de DevOps Release trabajan para abordar la gestión y coordinación del producto desde el desarrollo hasta la producción. Por lo general, trabajan en más detalles técnicos y obstáculos en los que un gerente de proyecto tradicional no puede participar. Los gerentes de lanzamiento de DevOps supervisan la coordinación, la integración y el flujo de desarrollo, prueba e implementación para respaldar la entrega continua. Están enfocados no solo en crear, sino también en mantener la cadena de herramientas de entrega de aplicaciones de extremo a extremo.

11. Entrenador DevOps: Al igual que todas las demás tendencias de hipercrecimiento en nuestra industria de TI, la adopción de la Metodología DevOps tampoco es inmune a posibles malentendidos y conceptos erróneos. Un número significativo de equipos y empresas de DevOps cometen errores organizativos, de comportamiento y operativos que afectan negativamente el rendimiento de los equipos de DevOps y su ajuste a las organizaciones en general, y por lo general aún no muy ágiles. Desafortunadamente, estas inconsistencias a veces terminan con la abolición de las prácticas de DevOps de las organizaciones que perjudican a nuestra industria y a los partidarios de DevOps.

Según el libro DevOps for beginner de Craig Berg (Berg, 2020), los ingenieros Devops son trabajadores de tiempo completo que generalmente tienen varias responsabilidades relacionadas con la producción y el mantenimiento del ciclo de desarrollo de software:

1. Gestión eficiente de proyectos a través de plataformas abiertas basadas en estándares.

2. Asegurar la resolución oportuna de los problemas del sistema utilizando los servicios de seguridad en la nube.

3. Desarrollar, analizar y evaluar scripts de automatización en los sistemas.

4. Mejorar la calidad.

En base a lo indicado por las diferentes fuentes, se identifican una serie de roles utilizados en los equipos de DevOps:

1. Desarrolladores: Los desarrolladores desempeñan un papel fundamental en un equipo DevOps ya que son los responsables de registrar el código en un sistema de control de versiones para su posterior liberación en ambiente de producción.

2. Ingeniero de Operaciones: Esta persona se encarga de garantizar que el software funcione sin problemas en producción. En un equipo de DevOps, gestionan la infraestructura, crean monitoreo integrado en el código, administran herramientas y resuelven problemas.

3. Project Manager (Evangelista): Este rol se encarga de supervisar el progreso del equipo en relación con los objetivos comerciales, establecer metas, cronogramas y garantizar la entrega oportuna de los productos o servicios.

4. Ingeniero de QA: Los profesionales de pruebas desempeñan un papel crucial en un equipo de DevOps al garantizar la calidad del software. Además de las pruebas manuales, también se enfocan en la automatización de pruebas.

5. Ingeniero de seguridad: Este rol se encarga de incorporar la seguridad y el cumplimiento desde el inicio del ciclo de desarrollo, en lugar de tratar de agregarlo posteriormente. Un ingeniero de seguridad en un equipo de DevOps adopta un enfoque estratégico y práctico para garantizar la seguridad del software.

Propuestas de valor

Según Microsoft (learn.microsoft.com, 2023), cuando un equipo adopta la cultura, las prácticas y las herramientas de DevOps, puede lograr cosas increíbles como las siguientes:

1. Acelera el tiempo de comercialización: A través de mayores eficiencias, colaboración mejorada en equipo, herramientas de automatización e implementación continua, los equipos pueden reducir rápidamente el tiempo desde el inicio del producto hasta el lanzamiento al mercado.

2. Adaptarse al mercado y la competencia: Una cultura DevOps exige que los equipos se centren primero en el cliente. Al combinar la agilidad, la colaboración en equipo y el enfoque en la experiencia del cliente, los equipos pueden brindar valor a sus clientes de manera continua y aumentar su competitividad en el mercado.

3. Mantener la estabilidad y confiabilidad del sistema: Al adoptar prácticas de mejora continua, los equipos pueden incorporar una mayor estabilidad y confiabilidad de los productos y servicios que implementan. Estas prácticas ayudan a reducir las fallas y el riesgo.

4. Mejorar el tiempo medio de recuperación: La métrica de tiempo medio de recuperación indica cuánto tiempo se tarda en recuperarse de un error o una infracción. Para administrar fallas de software, brechas de seguridad y planes de mejora continua, los equipos deben medir y trabajar para mejorar esta métrica.

Según AWS (AWS, 2023), algunas propuestas de valor corresponden:

1. Velocidad: El modelo de DevOps permite a los equipos de desarrollo y operaciones lograr estos resultados. Por ejemplo, los microservicios y la entrega continua permiten que los equipos se hagan responsables de los servicios y los actualicen con mayor rapidez.

2. Entrega rápida: Incrementa la frecuencia y el ritmo de las versiones, a fin de innovar y mejorar el producto con mayor rapidez. Cuanto más rápido publique nuevas características y arregle errores, más rápido podrá responder a las necesidades de los clientes y desarrollar una ventaja competitiva. La integración continua y la entrega continua son prácticas que automatizan el proceso de publicación de software, desde la creación hasta la implementación.

3. Confiabilidad: Garantiza la calidad de las actualizaciones de la aplicación y los cambios en la infraestructura a fin de poder realizar entregas más rápidas de forma confiable mientras ofrece una experiencia positiva a los usuarios finales.

4. Escalado: La automatización y coherencia ayudan a administrar sistemas complejos o cambiantes de forma eficaz con menos riesgo.

5. Colaboración mejorada: Desarrolla equipos más eficaces con un modelo cultural de DevOps, que enfatiza valores como la propiedad y la responsabilidad. Los desarrolladores y equipos de operaciones colaboran estrechamente, comparten

muchas responsabilidades y combinan sus flujos de trabajo. Así se reducen las ineficacias y se ahorra tiempo.

6. Seguridad: Adoptar un modelo de DevOps sin sacrificar la seguridad si utiliza políticas de conformidad automatizadas, controles minuciosos y técnicas de administración de la configuración.

En base a lo indicado por proveedores lideres de servicios tecnológicos, se identifica que una cultura DevOps reduce los plazos de entrega, las implementaciones se hacen con más frecuencia y se crea software de mayor calidad.

Riesgos

Según Byron Connoly en el sitio web ciospain.es (Connoly, 2018) un informe realizado por Gartner sobre DevOps en el cual los analistas aseguran que las organizaciones:

1. Necesitan poner los pies en la tierra DevOps y dar el valor real para el cliente, identificar quiénes son los clientes y lo que ven como valor en lugar de conectarse con el valor que el esfuerzo aportará a la organización y sus equipos DevOps.

2. La incapacidad de gestionar el cambio organizacional, algunos clientes continúan pasando por alto la importancia del cambio organizacional y enfocan los esfuerzos de las herramientas DevOps, un enfoque que no rendirá un valor significativo a

largo plazo. Las herramientas no son la solución a un problema cultural, según el informe.

3. La falta de colaboración es que muchos esfuerzos de DevOps están limitados a un solo dominio, como TI, en lugar de involucrar a otros grupos y partes interesadas, aseguran los investigadores, "DevOps requiere que las personas trabajen para encontrar soluciones, derribar barreras y funcionar como un equipo de unidades pequeñas, como en los deportes militares y profesionales".

4. Fallo al adoptar un enfoque interactivo con la idea de lanzar DevOps en un solo paso, pero la historia muestra que este enfoque tiene un alto error. Los investigadores dijeron que DevOps implica demasiadas variables para que este tipo de enfoque tenga éxito en una gran organización de TI. Los grupos de TI con más de 100 empleados encontrarán este enfoque desafiante, dijeron.

Según Google (Google, 2023), las organizaciones de alto rendimiento nunca están satisfechas y siempre intentan mejorar lo que hacen. El trabajo de mejora es continuo y es parte del trabajo diario de los equipos. Las personas en estas organizaciones entienden que no realizar cambios es tan riesgoso como los cambios mismos y no usan frases como “esta es la forma en la que siempre lo hicimos” para justificar su resistencia al cambio. Sin embargo, esto no significa que hay que adoptar un método indisciplinado para abordar los cambios. La administración de cambios debe realizarse de manera

científica en pos de un equipo medible o un objetivo organizacional, algunos errores frecuentes en la transformación de la cultura:

1. Tratan la transformación como un proyecto que se hace una sola vez. En las organizaciones de alto rendimiento, mejorar es un esfuerzo continuo y forma parte del trabajo diario de todos.

2. Tratan la transformación como un esfuerzo vertical. En este modelo, se cambian las vías jerárquicas organizacionales, se desplazan y reestructuran equipos y se implementan procesos nuevos.

3. No llegan a un acuerdo y no dan a conocer el resultado deseado. En ocasiones, las transformaciones se ejecutan con objetivos mal definidos o con objetivos cualitativos (en lugar de cuantitativos), como “tiempos de salida al mercado más rápidos” o “costos más bajos”.

En base a lo indicado por expertos y proveedores lideres de servicios tecnológicos, se identifican los siguientes riesgos como los principales a mitigar para la implementación de una cultura DevOps:

1. Resistencia al cambio organizacional: Ignorar la importancia del cambio organizacional y enfocarse únicamente en las herramientas de DevOps puede ser

perjudicial. Es fundamental abordar los aspectos culturales y fomentar la colaboración entre diferentes grupos y partes interesadas.

2. Falta de colaboración entre equipos: Se requiere un enfoque multidisciplinario que involucre a diferentes grupos y fomente la colaboración para encontrar soluciones y derribar barreras.

3. Tratar la transformación como un proyecto único: Las mejoras y la transformación cultural deben ser un esfuerzo continuo y formar parte del trabajo diario de todos en la organización. Tratarlo como un proyecto único limita el potencial de mejora continua.

ANEXO 13: ACTIVIDADES DEL PILOTO

Actividades del piloto:

Semana 1: Capacitación (Del 30 de octubre al 03 de noviembre del 2023)

Tabla 37: Piloto semana 1

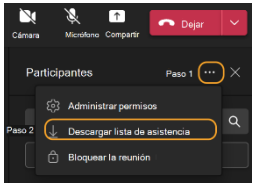
Actividad del plan de trabajo	Participantes	Resultados
Capacitación -> Preparación -> *Seleccionar o contratar al instructor con experiencia en cultura DevOps.*	• Encargado de tecnologías de información	El encargado de la unidad de tecnologías de información asigna a un profesional de su unidad como el instructor de esta actividad. Esto debido a que la administración pública no puede realizar procesos de contratación de manera tan rápida o sencilla en base a la ejecución de este piloto.

Capacitación -> Preparación -> *Definir la duración y la frecuencia de la capacitación.*	1. Encargado de tecnologías de información 2. Facilitador (Profesional de tecnologías de información)	Se define la semana del 30 de octubre al 03 de noviembre 2023 como la semana para realizar la capacitación. Se toman como base los documentos de estudio para contar con las certificaciones respectivas emitidas por Certiprof:
Capacitación -> Preparación -> *Crear el material de capacitación, incluyendo presentaciones y documentación para los participantes.*	1. Facilitador (Profesional de tecnologías de información)	• DevOps Essentials Profesional Certificate. • Kanban Essentials Profesional Certificate. Estos documentos en el piloto serán brindados a los participantes para

		su revisión y entendimiento, de igual manera como insumo para los que quieran optar por las certificaciones posteriormente.
Capacitación -> Preparación -> *Preparar un entorno físico o virtual si la capacitación se llevará a cabo en línea.*	1. Encargado de tecnologías de información 2. Facilitador (Profesional de tecnologías de información)	Se asigna un entorno virtual vía MS TEAMS para realizar el proceso de capacitación.
Capacitación -> Invitación y registro-> *Invitar a los participantes del grupo de Tecnologías de Información del IFAM.*	1. Encargado de tecnologías de información	En reunión presencial el día 30 de octubre, el encargado indica sobre la actividad a realizar, donde la misma es voluntaria para los participantes y donde se cuenta con la disponibilidad para la ejecución de las tareas del piloto en horario laboral.

Se cuenta con anuencia de participación de los siguientes colaboradores:

- Dos (2) profesionales del área de infraestructura
- Un (1) profesional del área de desarrollo
- Un (1) profesional asistente del área de desarrollo
- Un (1) gestor técnico del área de soporte
- Un (1) encargado de tecnologías de información

Capacitación -> Invitación y registro-> *Preparar el sistema de registro para obtener información sobre los participantes y su asistencia.*	1. Facilitador (Profesional de tecnologías de información)	Se utilizó el método de asistencia brindado por la herramienta MS TEAMS:  *Ilustración 13: Asistencia MS TEAMS*
Capacitación -> Capacitación -> *Contenido.*	1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura 3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo	En base a la documentación brindada se conversó sobre los diferentes temas según: • ¿Qué es DevOps?: Pagina 5 DevOps Essentials Profesional Certificate • Principios de DevOps: Se conversa sobre los principios

	5. Un (1) gestor técnico del área de soporte 6. Un (1) encargado de tecnologías de información	indicados en el marco teórico de este documento en la sección de cultura DevOps: Comunicación, Colaboración, Automatización, Feedback continuo o retroalimentación. • Cultura DevOps: Pagina 54 DevOps Essentials Profesional Certificate y se conversa sobre la cultura DevOps indicada en el marco teórico de este documento en la sección de cultura DevOps. • Casos de Éxito: Se conversa sobre los casos de éxito indicados en el punto 3.1 Revisión de

literatura sección 3.1.1 Casos de éxito de este documento.

- Historia y origen de Kanban: Pagina 15 Kanban Essentials Profesional Certificate.
- Beneficios de implementar Kanban en equipos y organizaciones: Paginas 17, 18, 23-24 Kanban Essentials Profesional Certificate.
- Limitación del trabajo en progreso (WIP): Pagina 54 Kanban Essentials Profesional Certificate.
- Visualización del flujo de trabajo: Paginas 22-24 Kanban Essentials Profesional Certificate.

		• Gestión de Flujo: Paginas 30-31 Kanban Essentials Profesional Certificate. • Tablero Kanban: Paginas 32-35 Kanban Essentials Profesional Certificate.
Capacitación -> Capacitación -> *Evaluación.*	1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura 3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo 5. Un (1) gestor técnico del área de soporte	Se realizó una mesa redonda para conversar sobre el contenido de la capacitación y conceptos más relevantes del proceso realizado. Esto fue consensuado por el grupo en base a las actividades semanales ya realizadas y las cargas de trabajo del grupo.

	6. Un (1) encargado de tecnologías de información	

Semana 2: Implementación de Tablero Kanban (Del 06 de noviembre al 10 de noviembre del 2023)

Tabla 38: Piloto semana 2

Actividad del plan de trabajo	Participantes	Resultados
Implementación de Tablero Kanban-> Definir las columnas del tablero-> *Diseñar las columnas que representarán las etapas del flujo de trabajo (Backlog, En desarrollo, Pruebas, Aprobado, Despliegue, En producción).*	1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura 3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo 5. Un (1) gestor técnico del área de soporte	Se definen las columnas del tablero en base a las etapas mencionadas (Backlog, En desarrollo, Pruebas, Aprobado, Despliegue, En producción).

	6. Un (1) encargado de tecnologías de información	
Implementación de Tablero Kanban-> Selección del proyecto-> *Seleccionar un proyecto en el cual se simulará el proceso.*	1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura 3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo	Se tuvo que realizar la división del grupo en base a los proyectos existentes, ya que algunos participantes del piloto no pertenecían al mismo proyecto. Se toma la retroalimentación para el instrumento, debido a que no debería ser solo la selección de un proyecto sino de todos los proyectos actuales.
Implementación de Tablero Kanban-> Crear tarjetas -> *Crear tarjetas para cada tarea o elemento de trabajo. Incluir*	1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura	Se crean las tareas respectivas.

descripción de la tarea, responsable, prioridad y plazos.	3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo	
Implementación de Tablero Kanban-> Asignar tareas a participantes-> *Identificar quién es responsable de cada tarea y asignarlas a los participantes correspondientes.*	1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura 3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo	Se asignan las tareas iniciales a los participantes.
Implementación de Tablero Kanban-> Configurar el tablero Kanban->	1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura	Se crea el tablero Kanban utilizando el Azure DevOps. Se menciona por parte de los participantes que consideran que esta tarea debe estar

Crear y configurar el tablero Kanban utilizando software o herramientas adecuadas. Implementación de Tablero Kanban-> Piloto del uso de Kanban->*Iniciar el flujo de trabajo moviendo las tarjetas de una columna a otra a medida que las tareas avanzan en el proceso DevOps.* Implementación de Tablero Kanban-> Revisar y actualizar->*Realizar una reunión de seguimiento para revisar el tablero*	3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo 1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura 3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo 1. Facilitador (Profesional de tecnologías de información) 2. Dos (2) profesionales del área de infraestructura	previo a las 2 anteriores, ya que significó la creación nuevamente de las tarjetas y asignaciones. Se realiza el ejercicio de uso del tablero Kanban. Al ser una herramienta gráfica y de seguimiento el proceso fue sencillo y práctico para los participantes. Se externa por parte de los participantes la facilidad del uso del tablero y que lograron percibir mejoras en

Kanban y actualizar el estado de las tareas.	3. Un (1) profesional del área de desarrollo 4. Un (1) profesional asistente del área de desarrollo	la comunicación y ejecución de las tareas. Las tarjetas utilizadas como ejercicio se finalizan en el tablero Kanban.

Semana 3: Sesión de formación en Cultura DevOps con actividad de LEGO y Chocolate (Del 13 de noviembre al 15 de noviembre del 2023)

Tabla 39: Piloto semana 3 parte 1

Actividad del plan de trabajo	Participantes	Resultados
Sesión de formación en Cultura DevOps con actividad de LEGO y Chocolate-> Sesión de repaso inicial-> *Breve sesión de repaso sobre los principios*	• Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo	Se realiza una mesa redonda para refrescar conocimientos con los participantes.

fundamentales de una cultura DevOps.		
Sesión de formación en Cultura DevOps con actividad de LEGO y Chocolate-> Actividad de construcción de LEGO -> *Dividir a los participantes en equipos pequeños (mixtos de diferentes áreas).*	• Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte	Se dividen los participantes en 3 grupos de 2. El facilitador ayuda en la participación de los 3 grupos.

Sesión de formación en Cultura DevOps con actividad de LEGO y Chocolate-> Actividad de construcción de LEGO -> *Proporcionar a cada equipo un set de LEGO y asignar una tarea específica de construcción.*	• Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información)	Se asignan las siguientes actividades a los grupos: Grupo 1: Creación de puente, el cual permita soportar el peso de otras figuras de lego. Grupo 2: Creación de edificio, simulando el Burj Khalifa. Grupo 3: Creación de laberinto, el cual permita que una pelota pueda realizar el recorrido de principio a fin.
Sesión de formación en Cultura DevOps con actividad de		Se realiza el ejercicio de construcción.

LEGO y Chocolate-> Actividad de construcción de LEGO -> *Tarea de construcción en equipos.*	• Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información	
Sesión de formación en Cultura DevOps con actividad de LEGO y Chocolate-> Facilitación y observación -> *Actuar como facilitador mientras los equipos trabajan en sus proyectos de LEGO.*	• Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo	El facilitador apoya a los diferentes grupos en el proceso de construcción, así como ayudar en la comunicación y colaboración de los grupos.

	• Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte	
Sesión de formación en Cultura DevOps con actividad de LEGO y Chocolate-> Facilitación y observación -> *Observar cómo se comunican y colaboran los miembros del equipo.*		Se identifica que los grupos 1 y 2, mantienen una comunicación un poco más fluida, lo cual ha hecho que la tarea se más complicada para el grupo 3.

	• Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información)	
Sesión de formación en Cultura DevOps con actividad de LEGO y Chocolate-> Discusión y reflexión -> *Organizar una sesión de discusión después de que todos los equipos hayan completado sus construcciones de LEGO.*		Se realiza una mesa redonda para conversar de las actividades y los retos al realizar las mismas.
Sesión de formación en Cultura DevOps con actividad de		Se brindan las recompensas de chocolate.

LEGO y Chocolate-> Recompensas -> *Proporcionar chocolate a todos los participantes como recompensa por la colaboración y el esfuerzo en la actividad.*	• Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información	

Semana 3: Evaluación y Mejora de la Cultura DevOps (Del 16 de noviembre al 17 de noviembre del 2023)

Tabla 40: Piloto semana 3 parte 2

Actividad del plan de trabajo	Participantes	Resultados
Evaluación y Mejora de la Cultura DevOps -> Preparación ->	• Facilitador (Profesional de tecnologías de información)	Se crea el siguiente cuestionario:

Diseñar un cuestionario de retroalimentación que aborde aspectos clave de la cultura DevOps.	• Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información	2. ¿Cómo describiría la comprensión general sobre una cultura DevOps? Excelente Bueno Aceptable Malo 3. ¿Cómo calificaría la comunicación y colaboración entre el equipo de tecnologías de información? Excelente Bueno Aceptable Malo 4. ¿Cómo describiría la claridad y visibilidad del tablero Kanban? Excelente

Bueno

Aceptable

Malo

5. ¿Cómo es la comunicación entre los miembros del equipo respecto al estado de las tareas en el tablero Kanban?

   Excelente

   Bueno

   Aceptable

   Malo

6. ¿Considera que los comentarios o retroalimentación realizada ayuda a la mejora al proceso?

   Si

   No

Evaluación y Mejora de la Cultura DevOps -> Preparación -> *Seleccionar a los participantes que proporcionarán retroalimentación.*	• Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información	Participantes indicados.
Evaluación y Mejora de la Cultura DevOps -> Recopilación de datos -> *Proporcionar instrucciones claras sobre cómo completar el*	• Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura	Se brindan a los participantes 10 minutos para completar el cuestionario.

cuestionario y establecer una fecha límite para la entrega de respuestas.	• Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información)	
Evaluación y Mejora de la Cultura DevOps -> Análisis de datos -> *Recopilar y analizar las respuestas del cuestionario para identificar patrones y áreas de mejora en la cultura DevOps.*	• Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo	El grupo de participantes brindo las siguientes respuestas al cuestionario: Pregunta 1: Excelente (5) Bueno (1) Aceptable (0) Malo (0) Pregunta 2:

	• Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información	Excelente (4) Bueno (2) Aceptable (0) Malo (0) Pregunta 3: Excelente (6) Bueno (0) Aceptable (0) Malo (0) Pregunta 4: Excelente (5) Bueno (1) Aceptable (0) Malo (0) Pregunta 5: Si (6)

		No (0)
Evaluación y Mejora de la Cultura DevOps -> Reunión de retroalimentación -> *Reunión individual o grupal con los participantes para discutir los resultados y obtener una comprensión de los mismos.*	• Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo	Los resultados del cuestionario permiten identificar que la actividad formativa ayuda al mejoramiento de la comunicación y colaboración entre los equipos. Se realiza una mesa redonda con los participantes para conversar sobre los resultados.

Evaluación y Mejora de la Cultura DevOps -> Plan de acción -> *Identificar las áreas específicas que requieran mejora.*	• Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura • Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información	En base a los resultados del cuestionario, no se cuentan con mejoras de peso que aplicar.

	• Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura	
Evaluación y Mejora de la Cultura DevOps -> Plan de acción -> *Establecer objetivos SMART para abordar las áreas específicas.*	• Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información • Facilitador (Profesional de tecnologías de información) • Dos (2) profesionales del área de infraestructura	Al ser un plan piloto y dado los resultados positivos del cuestionario no se crean objetivos SMART para abordar algún área específica.
Evaluación y Mejora de la Cultura DevOps -> Plan de acción -> *Comunicar a los participantes los*		No se realiza la actividad al no contar con objetivos que mencionar.

objetivos para el cumplimiento de las mejoras requeridas.	• Un (1) profesional del área de desarrollo • Un (1) profesional asistente del área de desarrollo • Un (1) gestor técnico del área de soporte • Un (1) encargado de tecnologías de información

ANEXO 14: ADAPTACIÓN DE UNA FILOSOFÍA DEVOPS

Adaptación de una filosofía DevOps al contexto actual del IFAM

Para una organización como el IFAM, una organización mayormente dependiente de productos de Microsoft, se proponen recomendaciones clave para la implementación exitosa de la filosofía DevOps, con el objetivo de gestionar la transición de manera estratégica y eficiente, transformando aún más la cultura organizativa. Primeramente, se destaca la necesidad de cultivar una sólida cultura de colaboración entre los equipos involucrados, mientras que se recomienda la utilización de herramientas específicas de Microsoft, como Azure DevOps, para automatizar procesos, gestionar el ciclo de vida de aplicaciones, y supervisar tareas, estableciendo así una base sólida para la adopción efectiva de una filosofía DevOps:

Tabla 41: Adaptación de la filosofía DevOps al contexto del IFAM

Herramienta	Detalle
Azure DevOps	Anteriormente conocida como Visual Studio Team Services, es una plataforma integral que permite gestionar el ciclo de vida de las aplicaciones, incluyendo control de versiones, seguimiento de problemas, automatización de compilaciones y despliegues, y colaboración entre equipos de desarrollo y operaciones.

Azure Repos (Control de versiones):	Azure Repos es una parte esencial de Azure DevOps que proporciona un sistema de control de versiones basado en Git. Permite a los equipos de desarrollo gestionar y rastrear cambios en el código fuente de manera efectiva, facilitando la colaboración en tiempo real. Con Azure Repos, los desarrolladores pueden realizar ramificaciones, fusiones y revisar el historial de cambios de manera sencilla.
Azure Pipelines (Automatización de compilación e implementación):	Azure Pipelines es una potente herramienta de automatización que permite a los equipos de desarrollo configurar flujos de trabajo automatizados para compilar, probar y desplegar sus aplicaciones en diferentes entornos. Puedes definir tuberías (pipelines) para aplicaciones web, aplicaciones móviles, contenedores, entre otros. Azure Pipelines facilita la integración continua y la entrega continua (CI/CD), lo que acelera la entrega de software de alta calidad.
Azure Boards (Gestión de tareas y seguimiento):	Azure Boards proporciona una plataforma de seguimiento y gestión de

	proyectos ágil. Los equipos pueden crear historias de usuario, tareas, épicas y más para planificar y realizar un seguimiento de las actividades del proyecto. Permite asignar tareas, definir prioridades y medir el progreso de manera efectiva, lo que ayuda a mantener a todos los miembros del equipo sincronizados.
Azure Test Plans (Pruebas continuas):	Azure Test Plans es una herramienta que permite a los equipos de calidad planificar y realizar un seguimiento de pruebas de software de manera sistemática. Permite crear casos de prueba, asignarlos a miembros del equipo y rastrear los resultados de las pruebas. La integración con Azure Pipelines facilita la automatización de pruebas para lograr pruebas continuas y la recopilación de resultados detallados.
Azure Artifacts (Gestión de paquetes y dependencias):	Azure Artifacts permite a los equipos gestionar paquetes de software y dependencias de manera eficiente. Permite crear y alojar repositorios de paquetes de

	NuGet, npm y Maven, lo que simplifica la gestión de dependencias en tus proyectos.
Azure DevTest Labs (Automatización de infraestructura):	Azure DevTest Labs facilita la creación y gestión de entornos de desarrollo y pruebas. Permite automatizar la implementación de máquinas virtuales, configurar políticas de uso y ofrecer a los equipos entornos aislados y personalizados para el desarrollo y las pruebas.

Al implementar Azure DevOps y fomentar una filosofía DevOps en la organización, permitirá lograr una mayor agilidad, innovación y satisfacción del cliente. Azure DevOps es la herramienta perfecta para habilitar esta transformación y maximizar el valor que la organización puede ofrecer en sus primeros pasos a través de la filosofía DevOps.