ADAM: Método Ágil para Adopción de estrategias de DevOps

JAIRO ALBERTO SOTO VELASQUEZ

Director
PhD. RAÚL MAZO PEÑA

REQUISITO PARA OPTAR AL TÍTULO DE MAGÍSTER EN INGENIERÍA

UNIVERSIDAD EAFIT
ESCUELA DE INGENIERÍA
MEDELLÍN - ANTIOQUIA - COLOMBIA
2023

Tabla de contenido

Resumen

Este trabajo de investigación se centra en el concepto de DevOps. DevOps es el acrónimo inglés de development (desarrollo) y operations (operaciones) y consiste en un conjunto de prácticas para mejorar la automatización y el monitoreo en todos los pasos de la construcción del software, desde la integración, las pruebas y el despliegue, hasta la configuración y la administración de la infraestructura necesaria para que el software funcione correctamente.

En muchos proyectos de desarrollo a nivel industrial se implementan las buenas prácticas de DevOps sin apalancarse en métodos de ingeniería, lo cual conlleva a que se implementen prácticas innecesarias para el proyecto, o a dejar de implementar otras que mejoran las diferentes etapas de este. En este trabajo se presenta el método de adopción de DevOps llamado ***Agile DevOps Adoption Methodology*** **(ADAM)**. El método consiste en una serie de actividades que se implementan de forma cíclica, comenzando con la identificación del estado actual de adopción de DevOps, siguiendo con la creación del backlog de buenas prácticas de DevOps, siguiendo con la definición de los objetivos de nivel de servicio, indicadores de nivel de servicio, y por último se utilizan prácticas de monitoreo y observabilidad para medir la efectividad del método propuesto. Este método de adopción lo evaluamos en el caso de una aplicación web para el diseño de líneas de productos llamada VariaMos. Los resultados obtenidos nos permitieron concluir sobre la velocidad y el nivel de automatización de la gestión del proceso de adopción de ADAM. En la evaluación, encontramos que usando ADAM los tiempos de despliegue y el *lead time* (plazo de entrega) disminuyeron en más de 500%, en comparación con un proceso de despliegue sin ningún método y ejecutado de manera manual. Además, el uso de ADAM no afectó la calidad del producto con el cual se evaluó el método; lo cual ha permitido una mayor evolución del producto y ha mejorado la confianza por parte de los usuarios finales. Además, los costos de mantenimiento se reducen en un 50%. En los resultados obtenidos también encontramos que el tiempo de adopción de DevOps es mucho mayor al inicio del proyecto, pero a medida que se adoptan las prácticas se va reduciendo el tiempo. Estos resultados constituyen una primera evidencia empírica del potencial que tiene el método ADAM propuesto en este trabajo en el mejoramiento de la adopción de DevOps en la construcción de aplicaciones Web.

Palabras clave

DevOps, aplicaciones Web, Infraestructura como código (IAC), Monitoreo continuo, Ingeniería de fiabilidad del sitio (SRE), DevSecOps,

1. Introducción

Motivación

Escuchamos hablar de DevOps, DevSecOps, MLOps, entre otros, y con el venir de los años y las nuevas tendencias tecnológicas probablemente vamos a encontrar nuevos términos relacionados con DevOps. De esta manera cuando vamos a adoptar DevOps en nuestros proyectos de software encontramos decenas de posibilidades y propuestas de cómo comenzar el proceso de adopción. En ocasiones encontramos que los procesos de adopción de prácticas de DevOps seleccionados están incompletos o no cumplen con las expectativas de los interesados, porque sólo abarcan una o algunas de las áreas del ciclo de vida de desarrollo de aplicaciones. En algunos proyectos se hace más énfasis en las mejores prácticas de desarrollo, en otros se hace más énfasis en la infraestructura y en otros, en los procesos de pruebas. Para que un proceso de adopción de DevOps sea lo más completo posible y podamos ver los resultados deseados es recomendable abarcar todas las áreas posibles como por ejemplo: Desarrollo, Aseguramiento de la Calidad, Infraestructura, y Seguridad. Y de esta manera determinar cual va a ser el grado de adopción de DevOps en cada una de estas áreas.

El proceso de ingeniería de software de hoy en día ya no termina con la entrega del o de los productos al cliente, como solía ser en el siglo pasado. El proceso de ingeniería de software ha sido extendido en los últimos 20 años con un conjunto de buenas prácticas y de herramientas que permiten disminuir los tiempos de desarrollo, mejorar la calidad del producto, disminuir los riesgos de seguridad entre otros, gracias a la automatización y a la cultura colaborativa, a este conjunto de buenas prácticas y de herramientas asociadas se les conoce como DevOps. El nombre DevOps viene de unir Dev que hace referencia a desarrollo y Ops que hace referencia a operaciones, y de esta manera se conjugan las diferentes técnicas y buenas prácticas de ambos procesos de una manera dependiente e interactiva. Por lo tanto podemos decir que “DevOps es un conjunto de prácticas que intenta cerrar la brecha entre el desarrollador y las operaciones en el núcleo de las cosas. y al mismo tiempo cubre todos los aspectos que ayudan en la entrega de software rápida, optimizada y de alta calidad” ***[1]***.

De acuerdo a la siguiente definición, DevOps es “una mezcla de patrones destinados a mejorar la colaboración entre el desarrollo y las operaciones. DevOps aborda objetivos e incentivos compartidos, así como procesos y herramientas compartidos” ***[2]***. Estos patrones se seleccionan de acuerdo al *stack* tecnológico que se esté utilizando para la construcción y ejecución de la aplicación. Las buenas prácticas a adoptar se definen con el equipo, teniendo como base el cumplimiento de los objetivos de nivel de servicio (SLOs) definidos para la aplicación. Se utiliza el proceso de gestión ágil para el seguimiento y ejecución del *backlog* del producto ***[3]*** y por último, se hace la medición periódica de los objetivos de nivel de servicio utilizando las técnicas de visualización y observabilidad definidas por la Ingeniería de Fiabilidad de Sitios (SRE del inglés Site Reliability Engineering).

Hipótesis de investigación

Se puede definir una metodología basándose en metodologías existentes como: metodologías ágiles para la gestión de los proyectos, metodologías de DevOps para la automatización del ciclo de vida de desarrollo y metodologías de fiabilidad de sitio o SRE, las cuales permiten identificar monitorear y valorar la adopción de estrategias de DevOps en proyectos de sistema de información.

Metodología de Investigación

La metodología utilizada en esta investigación se denomina Design Science ***[4]***. Esta metodología combina una serie de principios, prácticas y procedimientos, los cuales se utilizan en proyectos de investigación para sistemas de información. Los principios son:

- Creación y evaluación de artefactos de tecnologías de la información diseñados para resolver problemas organizacionales.
- Incluye procesos para el diseño de los artefactos, para resolver los problemas, para evaluar los diseños y para comunicar los resultados.
- Los artefactos generados deben incluir: desarrollos, modelos, métodos e instalaciones.

Las reglas prácticas que utiliza esta metodología son:

- Producir artefactos para resolver problemas.
- Se debe evaluar la utilidad, calidad y eficiencia del artefacto.
- Se debe comunicar apropiadamente la investigación a la audiencia apropiada.
- El desarrollo de los artefactos deben basarse sobre conocimientos y teorías existentes para obtener soluciones para el problema definido.

Esta metodología sigue seis etapas (Identificación del problema y motivación, Definición de los objetivos de la solución, Diseño y desarrollo, Demostración, Evaluación y Comunicación) relacionadas como se presenta en la ***figura 1***, resumidas en el resto de esta sección y desarrolladas a lo largo de esta tesina.

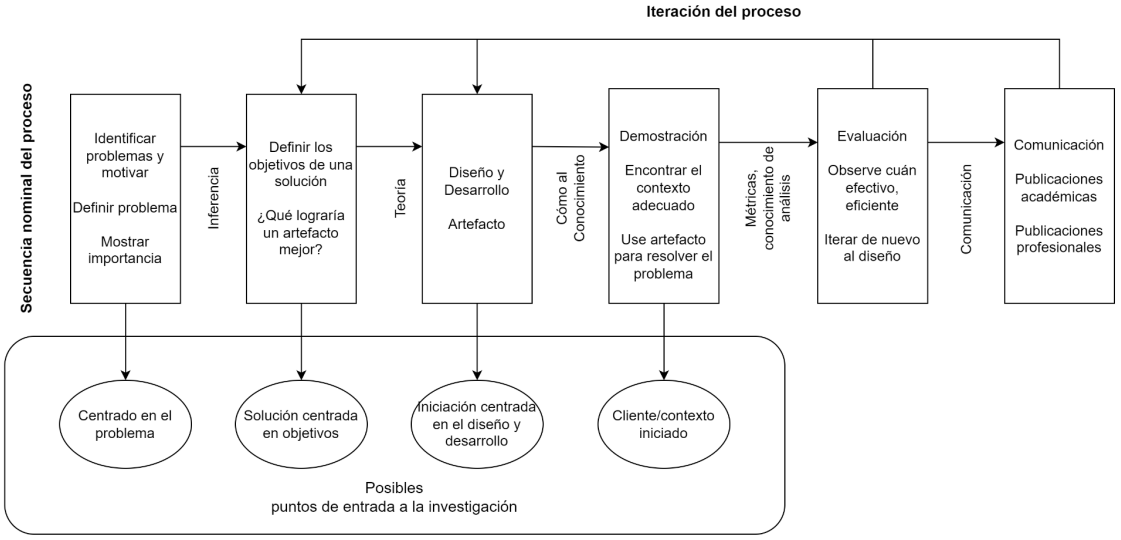

Figura 1: DSRM Process Model ***[4]***

Identificación del problema y motivación

Las metodologías y buenas prácticas de DevOps se implementan en los procesos de desarrollo de sistemas de información, obteniendo resultados cuantificables en dichos procesos, los cuales dan como resultado, tiempos de entrega más rápidos, disminución de errores, automatización de actividades técnicas, entre otros. Implementar estas buenas prácticas requiere recursos de capital y de personal, los cuales se integran directamente en el proceso de desarrollo. Evidenciamos la necesidad de proponer un proceso que nos permita medir la efectividad de la adopción de las buenas prácticas de DevOps, debido a que comúnmente se adoptan prácticas de DevOps en los proyectos de desarrollo sin identificar su verdadero impacto en el proyecto a nivel de costos, conocimiento requerido y complejidad que se adiciona al proyecto de desarrollo con estas nuevas actividades. Adicionalmente, no se cuenta con una manera objetiva para justificar el recurso adicional requerido para la implementación de las prácticas de DevOps.

De esta manera, en este proyecto de maestría hemos definido tres preguntas de investigación (PI):

PI1: ¿Cómo elegir la estrategia de DevOps para adoptar en un proyecto de sistemas de información?

PI2: ¿Cómo medir el progreso de adopción de prácticas de DevOps?

PI3: ¿Cómo medir el valor agregado que aporta el estado de adopción de DevOps al proyecto?

Definición de los objetivos

Los objetivos propuestos para este trabajo son los siguientes:

Objetivo 1: Definir un catálogo de buenas prácticas de DevOps, basado en investigaciones y en frameworks adoptados por la industria.

Objetivo 2: Crear el backlog de actividades de DevOps en una herramienta de gestión de proyectos tipo AGIL. En este backlog se adicionan las buenas prácticas, representadas en historias de usuario.

Objetivo 3: Definir los objetivos organizacionales (SLO), los objetivos de nivel de servicio (SLI), y los acuerdos de niveles de servicio, SLA, así como los demás componentes de la metodología SRE.

Objetivo 4: Evaluar el método de la adopción de las buenas prácticas de DevOps y verificar su coordinación con los objetivos de la organización.

Propuesta experimental para responder a las preguntas de investigaciòn

La pregunta de investigación 1 en la cual nos formulamos ¿Cómo elegir la estrategia de DevOps para adoptar en un proyecto de sistemas de información? nos lleva a investigar las estrategias existentes en diferentes bases de datos de información y nos dirige en dirección de la utilización del SRE (*Site Reliability Engineering*). DevOps es un conjunto de buenas prácticas que se utilizan para automatizar los procesos de desarrollo pero no explica cuáles prácticas debe implementar, ni cual es el orden en que las debo implementar. Esta tarea corresponde al SRE, un proceso que nos guía en cuanto a la implementación de las buenas prácticas de DevOps ***[5]***.

La pregunta de investigación 2, en cual nos formulamos: ¿Cómo medir el progreso de adopción de prácticas de DevOps? se resuelve investigando las diferentes metodologías para gestión de proyectos de desarrollo lo cual nos encaminó a la utilización de metodologías ágiles. Estas metodologías ágiles utilizan backlogs de producto compuestos por actividades, como Historias de Usuario, Épicas y otras ***[6]***, en las cuales se definen criterios de medición como son: Criterios de aceptación, porcentaje de implementación, porcentaje de priorización de las historias y de las actividades, y esfuerzo requerido. Esto nos llevó a definir el objetivo 2 de la investigación, en el cual es necesario crear un backlog de producto con las Historias de Usuario que representan las prácticas de DevOps a implementar.

La tercera pregunta de investigación (¿Cómo medir el valor agregado que aporta el estado de adopción de DevOps al proyecto?) se apoya de las metodologías para gestión de proyectos ágiles, para medir avance de implementación de las buenas prácticas, apoyadas de las prácticas de SRE en las cuales se definen los SLOs (*Service Level Objectives*), y los SLIs (*Service Level Indicators*) y todos estos indicadores que se generan se combinan los datos recolectados en las herramientas de monitoreo y observabilidad y se presentan los resultados en herramientas de inteligencia de negocio. Para lograr esto se definieron los objetivos 3 y 4, en los cuales se definen los SLI y los SLOs que seràn evaluados periódicamente para determinar el grado de aporte de la adopción de prácticas de DevOps en la consecución de los objetivos del proyecto.

Demostración

Para la demostración de la metodología ADAM, aplicamos los pasos propuestos en el proceso de desarrollo, despliegue y mantenimiento de un software para el diseño de líneas de productos. Con las diferentes herramientas y técnicas propuestas demostramos cómo se logra adoptar y monitorear las diferentes prácticas de DevOps integrándose a los planes de desarrollo y despliegue de la aplicación web VariaMos (www.variamos.com), la cual es usada para probar los conceptos de la propuesta presentada en esta tesina. Adicionalmente, con la adopción de procedimientos de SRE, como el establecimiento de los SLOs y SLIs respectivamente, medimos la efectividad de la adopción de este proceso para la aplicación VariaMos.

Evaluación

La evaluación del proyecto se realizó sobre la implementación de la metodología ADAM propuesta en el proyecto Variamos. Esta evaluación la detallamos en la sección 6 de esta tesina. En esta sección mostraremos los diferentes pasos implementados. Empezando por la selección de las buenas prácticas de DevOps a implementar, las cuales ayudan a alcanzar los objetivos de nivel de servicio propuestos por el equipo de desarrollo de la aplicación, los cuales se extraen de las prácticas de Ingeniería de Fiabilidad de Sitios. Adicionalmente veremos cómo se implementan prácticas de monitoreo, las cuales soportan la validez de los indicadores y de los objetivos y son extraídos de las prácticas de DevOps y de Ingeniería de Fiabilidad de Sitios. Por último se evalúa la adopción de gestión de proyectos ágiles, y vemos como con la creación de un backlog, se logra dar visibilidad al equipo de las prácticas de DevOps a implementar.

Comunicación

El resultado de esta investigación será documentado en esta tesina. Adicionalmente, está disponible el repositorio de código de manera pública en GitHub ***[7]***. Se publicó también una Wiki pública con documentación completa de la investigación y de los procesos de adopción de DevOps y su respectiva evaluación ***[8]***. Se publicó una versión resumida de este documento en la red LinkedIn ***[9]***. Se publicó una serie de videos que explican la implementación del proceso en el canal de YouTube: DevOps Central ***[10].***

Contribuciones

En este trabajo de investigación abordamos el proceso de la automatización del ciclo de desarrollo de software, denominado DevOps. **En este trabajo de maestría se diseñó el método denominado ADAM**, el cual fue evaluado sobre una aplicación Web (llamada Variamos) para el desarrollo de líneas de productos y de sistemas auto-adaptables. Al inicio de esta investigación, la aplicaciòn Web del proyecto Variamos estaba implementada sobre una arquitectura monolítica del tipo cliente servidor, y al finalizar el proyecto quedó implementada sobre una infraestructura en la nube utilizando servicios en la nube del tipo servicios de aplicaciones, servicios de contenedores, servicios de monitoreo, servicios de despliegue, entre otros.

Con el objetivo de que todo el equipo vea el progreso y el estado de cada una de las implementaciones de VariaMos, este proyecto de maestrìa diseñò e implementò un **pànel de tableros** de gestión de proyectos àgiles [3], de monitoreo y observabilidad [11], y de inteligencia de negocios [12] para el proyecto Variamos. [3]. Con la creación de esos tableros se logró definir de manera objetiva, y teniendo en cuenta los acuerdos con todos los interesados, que las historias de usuario de automatización, los niveles de servicio, las métricas, alertas y errores del proyecto, ademàs del estado de las prácticas de DevOps implementadas fueron visibilizadas como elementos integrales del proyecto.

Además, siguiendo el método **ADAM,** en este proyecto de maestría se crearon las **prácticas de DevOps más relevantes para el caso de aplicación Variamos**: Infraestructura como código, Estrategias de Control de versiones, monitoreo continuo entre otras [13]. Durante el desarrollo de este hito, enfrentamos retos como la definición de las plataformas y los entornos de ejecución, los cuales afectan el desarrollo y la implementación del proyecto. Creando, e implementado un DSL basado en un framework de industria, el cual permite al equipo de desarrollo y definir los proveedores y los frameworks disponibles, de acuerdo a la arquitectura definida para el proyecto. Este DSL está diseñado para definir los servicios y frameworks de tres de los más reconocidos proveedores de servicio de la nube.

Organización del documento

En el capítulo 2 presentamos los conceptos claves de las metodologías utilizadas para implementar el método ADAM. En el capítulo 3 presentamos las metodologías y buenas prácticas de DevOps utilizadas en proyectos de desarrollo. En el capítulo 4 presentamos el resumen del método ADAM y los diferentes pasos que se definieron para implementar este proceso en un proyecto de desarrollo. En el capítulo 5 se detallan los diferentes pasos del proceso ADAM y el detalle de cómo se deben de implementar. En el capítulo 6 explicamos cómo evaluamos la propuesta y en el capítulo 7 nos valemos de los backlogs de productos, los SLO y SLI para evaluar los resultados de la adopción de DevOps en nuestro caso de aplicación. En el capítulo 8 se explica cómo se presentan los resultados al equipo de desarrollo y cómo se evalúan los resultados. En el capítulo 9 se presentan las conclusiones de este trabajo de investigación y se esbozan los trabajos futuros que se pueden desprender de éste.

2. Conceptos clave

Prácticas de DevOps

Version Control Workflow

En los procesos de desarrollo de software es de suma importancia definir el flujo de trabajo de control de versiones que se utilizará; estos flujos de trabajo nos permiten guiar de manera controlada los cambios de código entre las diferentes etapas del proceso de desarrollo. Existen flujos de trabajo muy conocidos como Gitflow y Trunk-based Development. Utilizar un flujo de control de versiones, el cual debe ser conocido y utilizado por todo el equipo de desarrollo, ayuda a reducir los reprocesos que se crean cuando se despliegan nuevas versiones que no estaban probadas en un determinado ambiente.

Por ejemplo, en el proyecto VariaMos ***[14]*** se utilizaba principalmente el flujo de control de versiones denominado Forking Workflow ***[15]***. Este flujo de control de versiones fue utilizado en las 3 versiones previas del proyecto. La adopción de otras prácticas de DevOps en la última versión del proyecto, como la utilización de ambientes independientes para cada etapa del proceso (por ejemplo, Dev, QA y Producción), nos lleva a utilizar un flujo de control de versiones diferente. El flujo de trabajo de control de versiones, denominado GtiFlow ***[15]*** ha sido seleccionado como el nuevo flujo de control de versiones de nuestro caso de aplicación puesto que nos permite relacionar los cambios y su ubicación en el respectivo ambiente.

En la **Figura 2** podemos observar el flujo de control de versiones de Variamos; el cual es una variación del flujo Gitflow. En este nuevo flujo identificamos 3 tipos de ramas: “main”, “feature” y “release”. La rama main contiene el código que está en producción, las ramas feature contienen el código de las nuevas características, las ramas de tipo release contienen el código certificado por el equipo de pruebas. Las ramas se relacionan con los ambientes, por ejemplo, las ramas feature sólo se pueden desplegar en ambientes de desarrollo, las ramas de release sólo se pueden desplegar en ambientes de análisis de la calidad y por último, la rama main es la que se debe desplegar en producción. La utilización del workflow de control de versiones nos permite relacionar los cambios en el código que están en la respectiva rama y el ambiente donde están publicados.

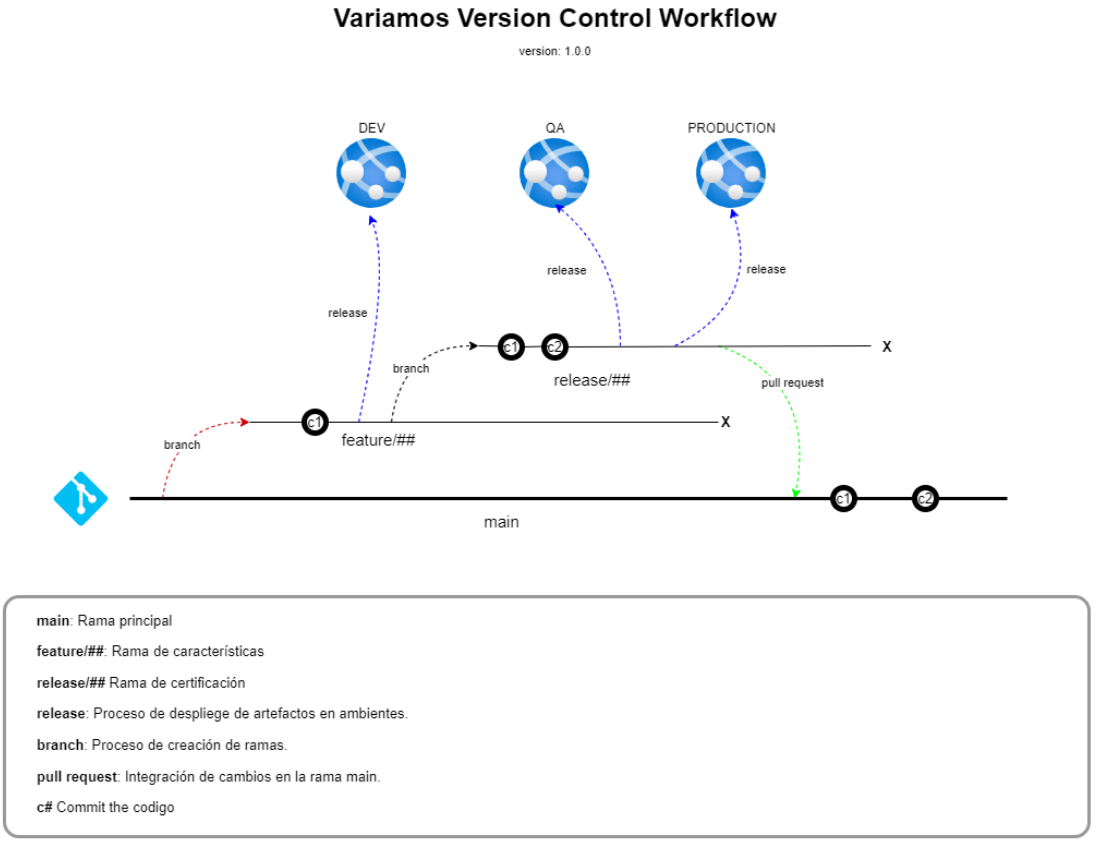

Figura 2 Flujo de control de versiones para Variamos.

Continuous Testing

Las pruebas continuas son una serie de buenas prácticas y estándares los cuales se integran con el proceso general de adopción de DevOps. Las pruebas continuas abarcan diferentes tipos de pruebas. Estos son dos ejemplos de pruebas continuas: las pruebas unitarias, las cuales son aplicadas a nivel de código en las tuberías de compilación; y las pruebas de seguridad, las cuales pueden ser aplicadas a nivel de código en tuberías de compilación o en los entornos de ejecución después de realizar los despliegues.

CI/CD Pipelines

Las tuberías de integración continua (CI) y entrega continua (CD) son herramientas que se han utilizado, desde hace mucho tiempo, en los procesos de DevOps. Estas están conformadas por un conjunto de pasos, principalmente automatizados. Generalmente están definidas en formatos de texto declarativos las cuales se ejecutan desde herramientas de DevOps como Jenkins, Cicle CI, Azure DevOps, entre otros.

No solamente existen estos dos tipos de tubería, también existen otras como las tuberías de despliegue y las tuberías de infraestructura como código. Las tuberías de despliegue son aquellas que nos permiten desplegar los artefactos compilados en los ambientes de

ejecución; es decir, en los ambientes de desarrollo, de certificación y de producción. Las tuberías de infraestructura como código nos permiten aprovisionar y actualizar los ambientes de ejecución.

Para mejorar los procesos de diseño y desarrollo de tuberías de automatización, se pueden implementar DSLs. Debido a que, la mayor la mayoría de las herramientas, que se utilizan para crear estas tuberías, requieren de un conocimiento específico sobre la estructura de los objetos, propiedades y clases que componen la infraestructura. Esto conlleva a una gran generación de errores y de incidentes de seguridad, debido a que pueden quedar puertos abiertos, o configuraciones incorrectas que no siguen los patrones arquitectónicos definidos.

Despliegue continuo

El despliegue continuo es la técnica que nos permite desplegar de manera automatizada los cambios en los ambientes productivos. Utilizando el tipo de implementación azul/verde ***[5]*** podemos desplegar en algunas de las instancias de producción las versiones nuevas (las verdes) y la versión anterior está representada por las instancias de color azul. Después de realizar las diferentes pruebas se puede tomar la decisión de desplegar la nueva versión (verde), en todas las instancias o simplemente devolver los cambios.

Infraestructura como código

Mediante esta práctica se puede definir la infraestructura de los ambientes de ejecución de la aplicación en plantillas o scripts. A estos scripts a los cuales se les implementan las diferentes prácticas que también se adoptan para el código fuente como: Flujo de trabajo de Control de Versiones ***[5]***, Administración de artefactos ***[5]***, Etiquetado ***[5]***, SemVer versioning ***[1]***, pruebas unitarias ***[5]***, etc. Adicionalmente se pueden desarrollar DSLs, para hacer más productivo el desarrollo de los entornos de ejecución. En la **figura 3** vemos un diagrama tipo .bicep ***[16]***, el cual es un DSL para modelar infraestructura en la nube Azure del vendedor: Microsoft. En el diagrama se visualizan, el ASP o servicio de computo donde se ejecutan servicios REST y las aplicaciones Web de la aplicación VariaMos.

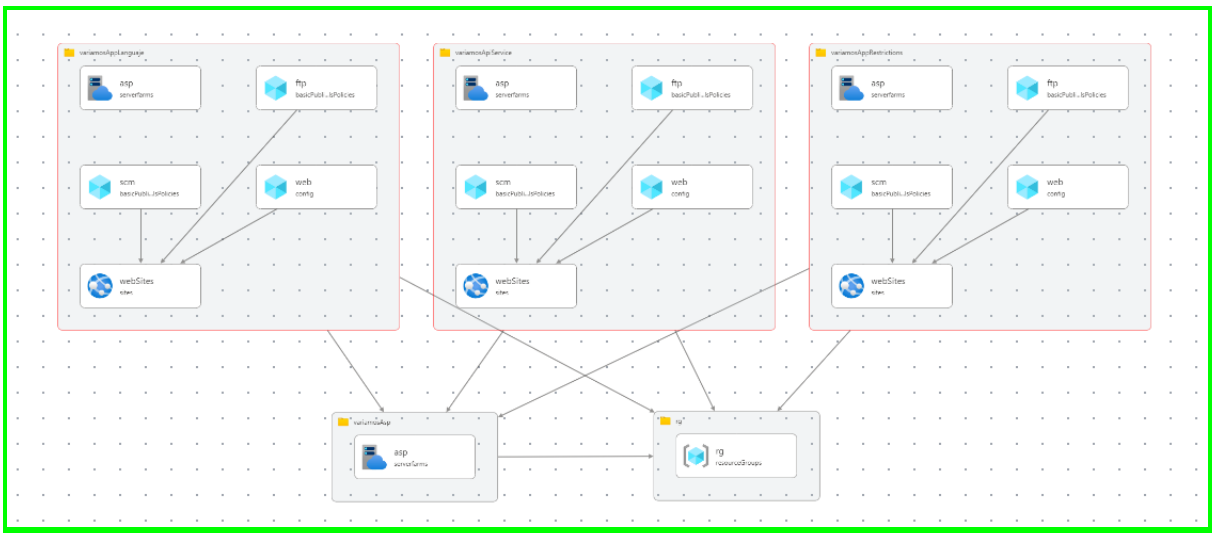

Figura 3. Representación gráfica de una plantilla .bicep para aprovisionamiento de infraestructura en la nube de Azure.

DevSecOps

DevSecOps es una metodología que abarca las buenas prácticas de seguridad que se deben adoptar en los procesos de desarrollo de software. Algunas de esas buenas prácticas son: Modelado de amenazas ***[17]***, Seguridad como código ***[17]***, Supervisión y registro centrados en la seguridad ***[17]***.

Modelado de Amenazas

El modelado de amenazas es una metodología de seguridad que se utiliza en el área de DevSecOps, para identificar amenazas de seguridad en las aplicaciones o servicios desde el diseño de las aplicaciones ***[18]***. En la ***figura 4*** vemos el diagrama de una aplicación que consume tres Apis y las Api hacen llamados a la base de datos para persistir los datos. Con la herramienta de DSL llamada “Microsoft Threat Modeling”, diseñamos el diagrama y luego generamos el reporte, el cual genera la lista de amenazas basadas en el enfoque STRIDE ***[27]***, y posibles acciones de mitigación. Luego de identificar las amenazas se crean las historias de usuario en las cuales se van a solucionar cada una de estas, ya sea implementaciones por código o por adopción de prácticas de DevOps y/o DevSecOps.

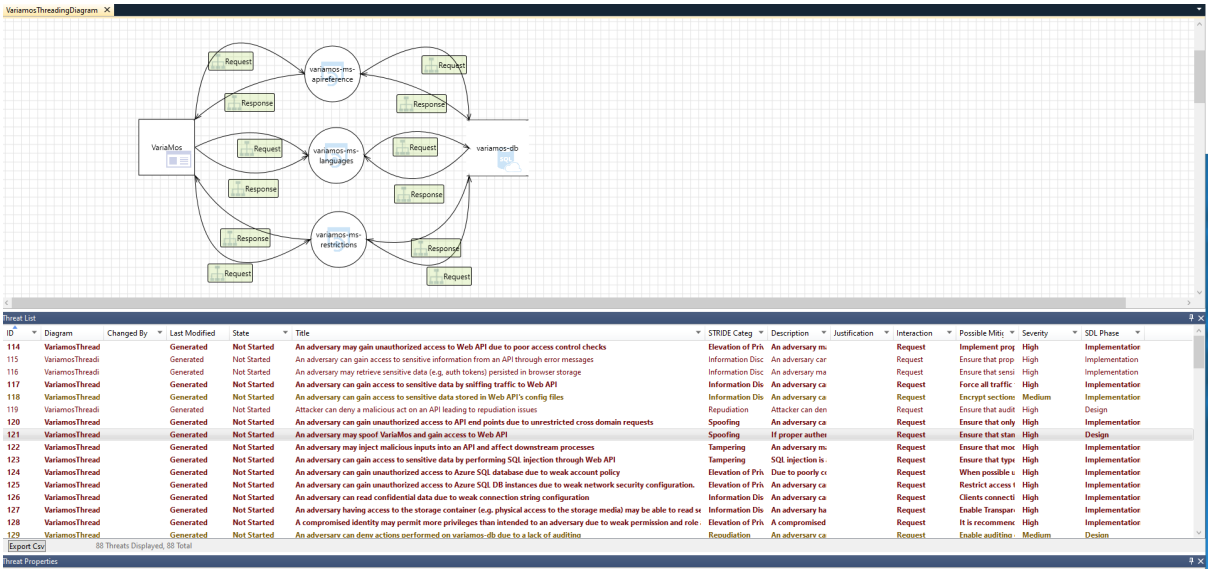

Figura 4. Diagrama de Amanzas de la aplicación web Variamos

Prácticas de Ingeniería de Confiabilidad del sitio (SRE)

Monitoreo Continuo

El monitoreo continuo nos permite recolectar toda la información que genera la aplicación; por ejemplo, trazas, logs y page requests. También podemos recolectar la información de

los proyectos de implementación desde la herramienta AGIL; por ejemplo, plazo de entrega (Lead Time), velocidad, diagrama de flujo acumulativo, tiempo de ciclo. En los tableros de monitoreo (como el que se presenta en la Figura 5) podemos incluir las diferentes métricas con las que evaluaremos el sistema.

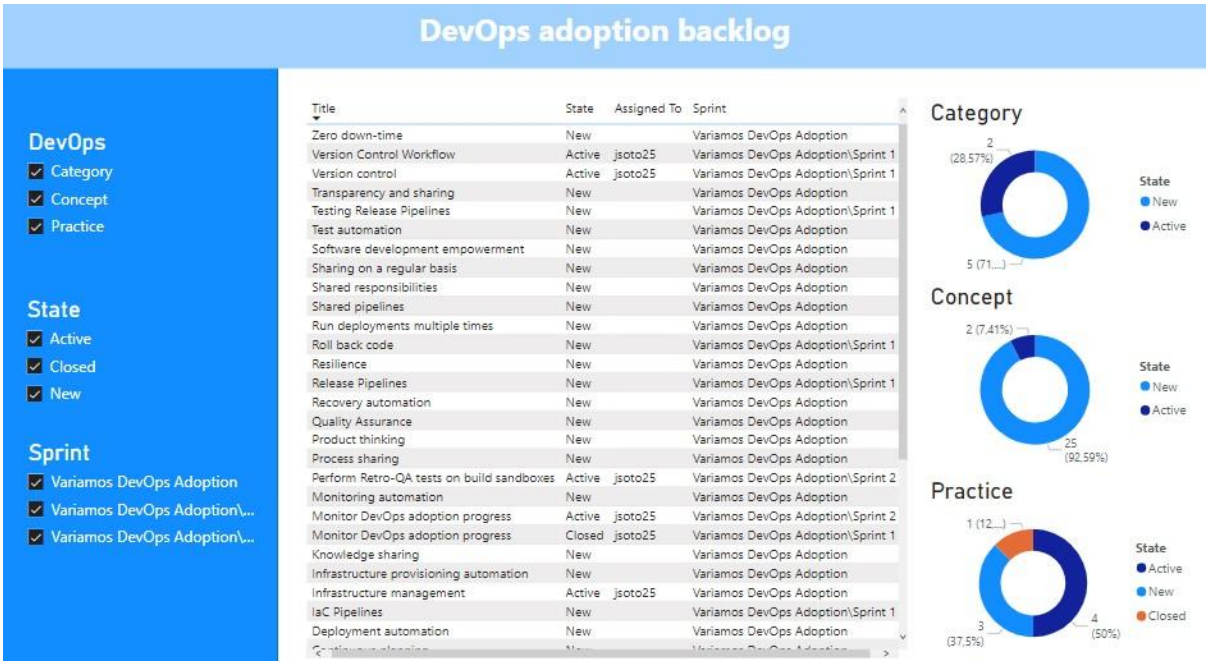

Figura 5. Tablero de monitoreo de avance del proceso ADAM aplicado a la herramienta Variamos

Mediante técnicas y herramientas de monitoreo podemos verificar el “estado de salud” del servicio o aplicación y visualizar los SLI, SLO y SLAs, los cuales ayudan a determinar cuales son las siguientes historias de usuario a implementar de una manera programada de acuerdo a las necesidades de la organización. Se utilizan tableros de Inteligencia de negocio para visualizar la información de logs, trazas, alertas y cualquier otro tipo de evento que se registre en los tableros.

Definición de los SLOs, SLIs y SLAs

La SRE nos invita a definir los indicadores, los objetivos y los acuerdos con las siguientes definiciones. Service Level Objetives - SLO, Service Level Indicators - SLIs y Service Level Agreements - SLAs. Los SLOs son los objetivos que quiere alcanzar la organización, por ejemplo que el 90% de las transacciones sean correctas. Los SLIs son los indicadores que nos ayudan a validar los SLOs. Los indicadores son métricas como por ejemplo Transacciones/Hora. Los SLAs hacen referencia al nivel valor mínimo que los SLOs deben cumplir y si están por debajo de éstos podrían haber consecuencias por el incumplimiento de los SLOs.

Estos indicadores se ayudan de herramientas de BI para visualizar el resultado. En la ***figura 6*** vemos como es el comportamiento de la aplicación y como en ciertos momentos los SLAs no se han logrado, lo que incurre en incumplimiento de los contratos.

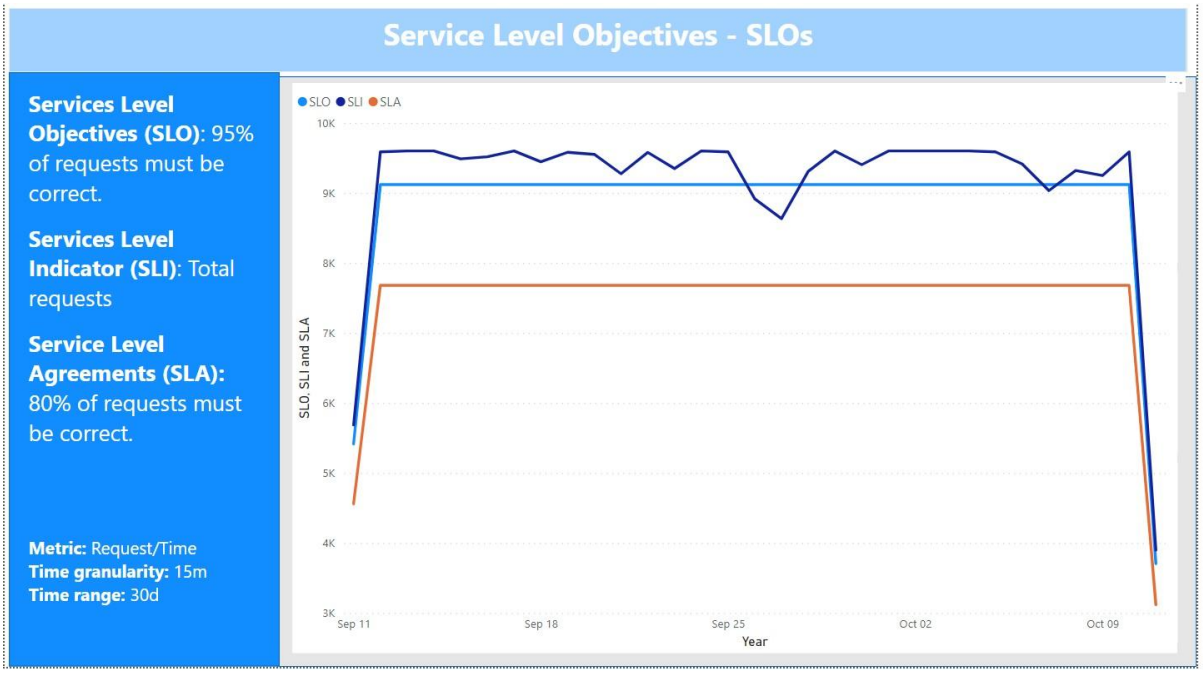

Figura 6: **Tablero en Power BI que permite visualizar y comparar el comportamiento de los SLOs.**

3. Estado del arte

Para identificar las buenas prácticas de DevOps que deben ser consideradas en el método presentado en esta tesina, se realizaron búsquedas en diferentes bases de datos de artículos: IEEE Xplore, ICEIS, Science Direct y Google Scholar. Adicionalmente se realizaron búsqueda de material técnico en portales comerciales como O'Reilly, OverDrive y otros portales de empresas de como Google, Microsoft, AWS, y Puppet. Se seleccionaron más de 150 artículos relacionados con el tema de DevOps. Luego estos artículos fueron categorizados por tema, herramientas, vendedores, fecha de publicación.

Estos artículos fueron reunidos en un sistema de documentos llamado SharePoint, el cual permite realizar búsquedas en texto completo y adicionar los campos para búsqueda para agrupar las búsqueda como por ejemplo: Autor, tema, vendedor, año de publicación.

En el resto de esta sección se resumen los aspectos principales de los métodos y buenas prácticas encontrados en la literatura seleccionada. En esta clasificación pondremos a las propuestas que tienen un proceso bien definido en la categorìas de métodos, y las mejoras puntuales que se usan en alguna de las etapas de un proceso de DevOps las clasificamos en la categorìa de buenas prácticas. Tanto los métodos como las buenas prácticas son, eventualmente, implementados en una o varias herramientas que también se presentarán en el método o buena práctica correspondiente.

Podríamos decir que los inicios de DevOps se encuentran entre los años 2000 y 2001. Varias tendencias de la tecnología conforman la base para el desarrollo de DevOps, como fueron los movimientos ágiles con la definición del manifiesto ágil y los movimientos lean con el mapeo del flujo de valor con tableros Kanban adoptados en la cultura organizacional como en Toyota ***[19]***.

Existen muchas estrategias de DevOps y el número sigue creciendo al pasar de los años a medida que evolucionan las nuevas tendencias tecnológicas. Se realizó una búsqueda y encontramos que una de las estrategias más adecuadas para aplicaciones web basadas en entornos de la nube es la que se conoce como "DevOps As A Service" en donde el desarrollo, las pruebas y las demás prácticas de DevOps se ejecutan en ambientes en la nube ***[20]***.

DevOps para aplicativos móviles se basa en las prácticas de DevOps generales como CI/CD pipelines, Continuous security, e.t.c. A su vez, lo que lo diferencia de DevOps para otras arquitecturas o tipos de proyectos, es la diversidad de sistemas operativos y de herramientas que se requieren para realizar las pruebas de calidad, las pruebas de performance (rendimiento), las pruebas de interfaz y también la forma como se realiza el monitoreo de las aplicaciones ***[21]***.

Las aplicaciones contenerizadas requieren estrategias especializadas de DevOps, las cuales representan retos importantes con respecto al monitoreo, el auto escalamiento y la seguridad. Por tal motivo las estrategias de adopción de DevOps son diferentes, las aplicaciones contenerizadas en cuanto a las herramientas y tecnologías utilizadas con respecto a las utilizadas en aplicación multi capas o aplicaciones móviles. Las aplicaciones

multicapas se implementan generalmente en máquinas virtuales y los microservicios se implementan en PODs. Las pruebas funcionales en una aplicación de Microservicios pueden realizarse con las mismas herramientas que se utilizan en aplicaciones monolíticas, como JMeter o Selenium, mientras para aplicaciones móviles las herramientas son diferentes y muy específicas para la plataforma, como por ejemplo TestFlight para aplicaciones en iOS y FireBase en Android. ***[16]***.

Otra tendencia de DevOps aplicada a las Machine Learning y Deep Learning se denomina MLOPS, esta tecnología permite a los equipos de ingeniería y ciencia de datos concentrarse en la construcción de los modelos, entrenamiento y actualización de los modelos, los cuales una vez validados pasan a los equipos de operaciones para ser integrados en las aplicaciones, desplegados , monitoreados y retroalimentar a los equipos. Es en este punto donde se integra DevOps con Machine learning integrando nuevas prácticas de DevOps como la denominada Continuous Model Delivery ***[22]***.

El término SRE nace en Google,***[5]*** cuando se vio la necesidad de realizar las implementaciones de los procesos de infraestructura y desarrollo de aplicaciones de manera mucho más rápida y automatizada utilizando las metodologías, buenas prácticas y herramientas enmarcadas en el término DevOps. Después de un tiempo estas dos metodologías, DevOps y SRE comenzaron a ser utilizadas en la gran mayoría de los proyectos de desarrollo a nivel industrial e investigativo.

4. Resumen de la propuesta

El método de adopción de DevOps propuesta en esta tesina se llama ADAM (Fast DevOps Adoption Method) y su construcción se basa en trabajos anteriores sobre buenas prácticas; adicionalmente su construcción capitaliza la experiencia laboral del autor de este trabajo (más de 7 años trabajando como ingeniero DevOps y otros 16 años de experiencia como director de desarrollo en múltiples proyectos a nivel industrial).

El método ADAM está compuesto de 4 grupos de actividades: (Estado de la adopción, Prácticas de Ingeniería de confiabilidad del sitio, Gestión de proyectos y Monitoreo) las cuales se recomienda implementar de manera continua como lo muestra el proceso de la ***Figura 7.*** Este método complementa la implementación de buenas prácticas de DevOps que están representadas en la ***Figura 8.*** debido a que las buenas prácticas de DevOps se implementaran de acuerdo a los objetivos definidos por el equipo, Podemos tomar como ejemplo un SLO de 99.999% esto se puede lograr implementando entre otras prácticas la auto escalabilidad, la cual se logra en el diseño e implementación de la infraestructura como código donde se define que si el uso de los recursos (CPU) no están cumpliendo con los SLO definidos por las peticiones, entonces se debe de escalar la CPU de manera horizontal o vertical de acuerdo a cómo esté diseñada la arquitectura del sistema.

En el grupo de actividades ***"Estado de la adopción"***, agrupamos los pasos y prácticas que vamos a utilizar para determinar el estado actual de adopción de las prácticas de DevOps en un proyecto de desarrollo de software. Se parte del listado de buenas prácticas de DevOps, luego buscamos en diferentes sistemas de información la evidencia que nos indique el estado de adopción de cada práctica. Y finalmente, valoramos el estado de adopción de acuerdo con los valores expresados en la ***Figura 1, Valores que caracterizan cada práctica de DevOps.*** Después de valorar el estado de adopción se determina si se logró el objetivo de la adopción. Si no se ha logrado el objetivo se asigna una nueva historia de usuario para el siguiente sprint, esto se continúa hasta lograr la valoración presupuestada o hasta tomar o hasta tomar la decisión de no continuar con la adopción de la práctica debido a que no está aportando al logro de los objetivos.

El grupo de actividades denominado ***"SRE"*** hace referencia a las prácticas de Ingeniería de Confiabilidad de Sitios que nos permitirán justificar ante el equipo de desarrollo, operaciones e interesados del proyecto, las prácticas de DevOps a adoptar. Algunas de las principales prácticas de SRE son las siguientes: definición de los Objetivos de Nivel de Servicio o SLOs y la definición de los Indicadores de Nivel de Servicio ó SLIs ***[5]***.

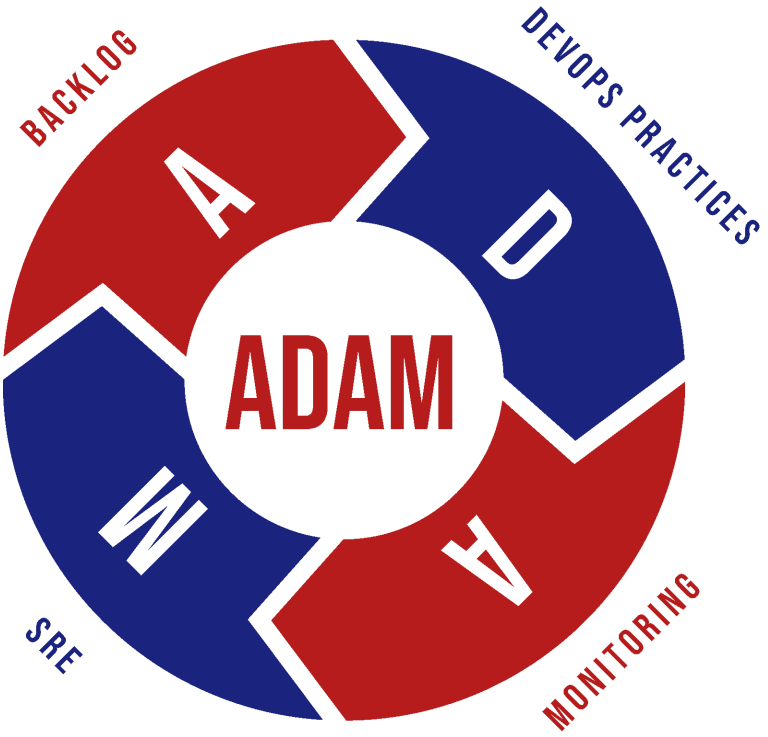

Figura 7: **Diagrama ADAM**

El grupo de prácticas denominado **“Monitoreo”** incluye las prácticas de monitoreo y visibilidad con las que visualizamos las métricas y los indicadores del proyecto. Por ejemplo, si se define un SLO que nos indique un 99.999% de disponibilidad del sistema en producción, entonces se deben implementar las prácticas de DevOps que nos ayuden a lograr el objetivo. Por ejemplo, la Infraestructura como código nos permite crear modelos de la infraestructura en los cuales se puede definir la forma en que se puede escalar un recurso como por ejemplo el tamaño de la CPU. Si el objetivo es que el uso de la CPU es mayor de 80% entonces se configura la infraestructura para que crezca de manera horizontal o vertical. Este tipo de auto escalamiento se puede aplicar a cualquier tipo de recurso como memoria, capacidad de almacenamiento entre otros.

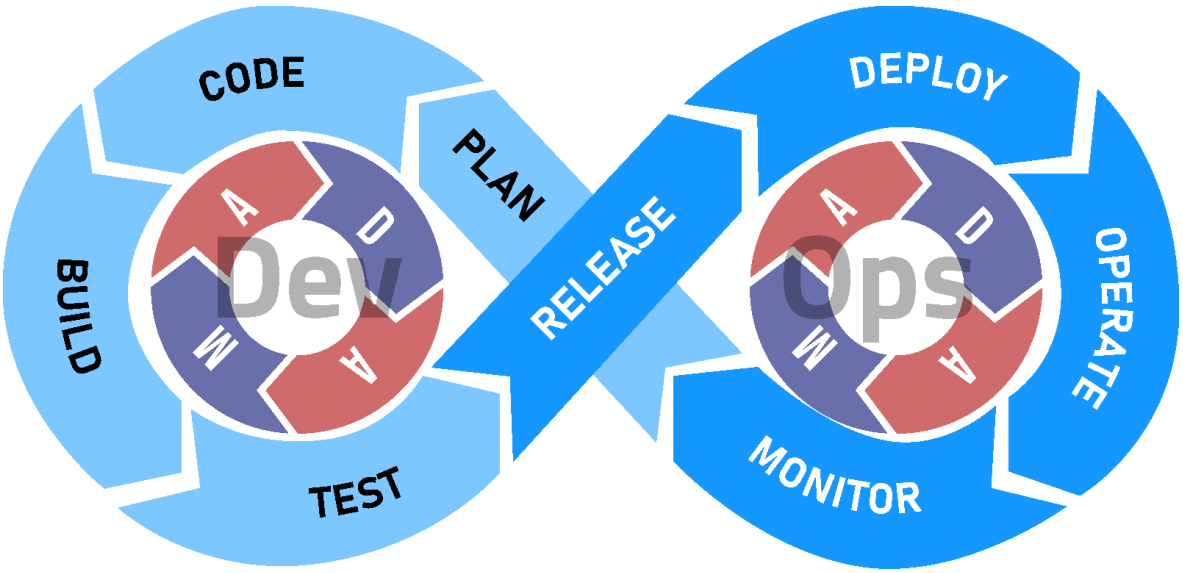

Figura 8: **Integración de de la metodología ADAM con el ciclo Infinito de DevOps**

En el último grupo de actividades de la estrategia ADAM, está "**Gestion Agil de Proyectos**". Este grupo está compuesto por las actividades de las metodologías AGILEs para la gestión de proyectos, como por ejemplo la definición del backlog de historias de usuario. En este punto se deben crear historias de usuario por cada buena práctica de DevOps a implementar ***[3]***, ***[23]***. En estas historias de usuario se definen los criterios de aceptación para la implementación de las buenas prácticas de DevOps y adicionalmente se da la prioridad entre valores de Alta, media y baja. Adicionalmente se puntúan las historias utilizando la serie fibonacci y por último se asigna el periodo de tiempo o Sprint, en que se ejecutara la historia de usuario.

5. Propuesta

Diseño

Estrategia de adopción de DevOps propuesta

En este documento proponemos el método llamado **Fast DevOps Agile Method (ADAM)**. El objetivo del método ADAM es servir de marco y de soporte para los interesados que quieran o necesiten implementar procesos de DevOps en el desarrollo de aplicaciones. El procesos general de esta estrategia está compuesto en 4 pasos principales, comenzando con la identificación del estado actual, seguido de la elaboración de un backlog con las diferentes buenas prácticas a adoptar, luego pasamos a la definición de los objetivos de niveles de servicios o SLIs y los indicadores de niveles de servicio, por último encontramos el paso en el cual se monitorean y se observa los cumplimientos de los objetivos de los niveles de servicios (SLI) utilizando técnicas de site reliability engineering (SRE). Esta estrategia se debe ejecutar de una manera cíclica y continua.

El Proceso

El proceso ADAM consiste en 4 pasos principales que se ejecutan de manera interactiva.

Estado Adopción DevOps

En el primer paso se identifica el estado actual de adopción de DevOps, se identifican las actividades que ya están automatizadas, mediante la implementación de buenas prácticas y herramientas.

SRE

Para el segundo paso nos valemos de metodologías de SRE para identificar los objetivos de nivel de servicio (SLOs), los indicadores de nivel de servicio (SLIs) y los acuerdos de niveles de servicio (SLAs), de esta manera se define el nivel mínimo de los indicadores los cuales debe mantener la aplicación para considerarse de que está ejecutando de manera satisfactoria.

Gestión Ágil de Proyectos

En el tercer paso creamos el backlog de actividades para implementar las buenas prácticas de DevOps, DevSecOps y de SRE, Este backlog se combina con el backlog de desarrollo de proyecto, con el objetivo de que tenga visibilidad en el proyecto, y los equipos puedan ver e implementar estas actividades.

Monitoreo

El último paso consiste en utilizar dashboards de BI para representar el comportamiento de los SLOs y de esta manera dar prioridad a las siguientes historias de usuario relacionadas con la adopción de DevOps, DevSecOps y SRE.

Descripción de la estrategia de adopción de DevOps

Esta estrategia consiste en lo siguientes pasos:

1. Definir las prácticas de DevOps a implementar en el proyecto [1], [24], indicando el estado actual y el estado esperado de la práctica utilizada. El valor del estado se obtiene de la ***figura 9*** [25].
2. Seleccionar las diferentes herramientas a utilizar para la adopción de las buenas prácticas. La selección de éstas se hace teniendo en cuenta los siguiente aspectos:
    - Arquitectura de la solución
    - Frameworks de desarrollo
    - Stack de ejecución
    - Normas de seguridad que debe de cumplir la aplicación
    - Herramientas para automatización utilizadas en la organización
    - Conocimiento técnico del equipo implementador
    - Modelo de licenciamiento de herramientas de desarrollo y automatización
    - Contratos con proveedores de la nube con que cuenta el proyecto u organización.
3. Utilizar la metodología de trabajo ágil en la herramienta seleccionada para la gestión del proyecto, se escoge el proceso ágil entre las posibilidades disponibles de AGIL, SCRUM o CMMI y relacionar tipos de ítems del agilismo con la categorización de de DevOps, como se muestra en la ***figura 10***.

Descripción valor	Acrónimo	Valor o rango en %
Fully Implemented	FI	100
Largely Implemented	LI	[51,99]
Partially Implemented	PI	[21,50]
Not Implemented	NI	[1,20]
Not Yet	NY	0

Figura 9. Valores que caracterizan cada práctica de DevOps.

Tipo de actividad Ágil	**Categoría DevOps**
Epica	Categoría

Feature	Concepto
Historia de usuario	Prácticas

Figure 10. Relación entre los ítems de trabajos de metodología ágil con los ítems categorizados de las estrategias de DevOps

4. Construir un backlog en una herramienta de tablero para metodología ágil, en cual se prioricen las actividades durante los sprints. Este paso es de suma importancia, porque nos permitirá hacer seguimiento del avance de la estrategia y evaluar su efectividad en la implementación en un proyecto de desarrollo.
5. Implementación de las buenas prácticas definidas por el equipo de trabajo, iniciando por la práctica denominada: PC22-Version Control Workflow [24]
6. Implementar un proceso de Ingeniería de la fiabilidad del sitio (SRE), el cual nos permite evaluar la efectividad del proceso de adopción de DevOps.

Desarrollo

¿Cómo elegir la estrategia de DevOps?

Se realizó una búsqueda de las diferentes estrategias de adopción de DevOps, en diferentes fuentes comerciales y educativas III.i.Se concluyó que una estrategia de DevOps para adoptar en un proyecto de desarrollo de software, consiste en elegir una serie de actividades o buenas prácticas de DevOps que se implementan en el proceso de desarrollo de la solución. Las prácticas se eligen teniendo en cuenta diferentes aspectos, como son: Arquitectura de la solución, Frameworks de desarrollo, experiencia de los equipos, presupuestos disponibles, fechas de entrega, y restricciones técnicas.

¿Cómo medir el progreso de adopción de prácticas de DevOps?

Se categorizó las buenas prácticas de DevOps de manera que podemos tener una relación entre las buenas prácticas y las actividades de la metodología Ágil, de tal manera que definimos una relación entre historias de usuario y buenas prácticas. Al tener esta categorías creamos un backlog con las historias de usuario las cuales se distribuyeron en diferentes sprints. Utilizando herramientas de Ágiles como Azure DevOps o Jira Software se realiza el seguimiento del día a día de los sprints. Adicionalmente se utilizan herramientas de Inteligencia de negocio como Power BI o tableau para evaluar el progreso de la implementación de las buenas prácticas.

¿Cómo medir el valor agregado que aporta el estado de adopción de DevOps al proyecto?

Para medir el valor agregado que aportan la implementación de las prácticas de DevOps de una manera objetiva decidimos implementar las metodologías de **Site Reliability Engineering** (SRE). De esta manera al definir los Service Levels Objectives (SLOs), los

Services Levels Indicators (SLI) y los Error Budgets e incorporarlos en las actividades del proceso ADAM, logramos realizar la evaluación de manera continua y repetitiva comparando el resultado de los sprint anteriores y de esta manera visualizar el grado del aporte.

DSL para Adopción de DevOps

Se desarrolló un Lenguaje específico de dominio, para diseñar la estrategia de adopción de DevOps. Este lenguaje permite definir los contenedores que conforman una aplicación, los objetivos de nivel de servicio que debe de cumplir la aplicación, los indicadores de niveles de servicio, con los cuales se miden los objetivos. Con esta información el modelo genera varios reportes con las siguientes salidas.

- Buenas prácticas a implementar
- Herramientas a utilizar para la implementación
- otros.

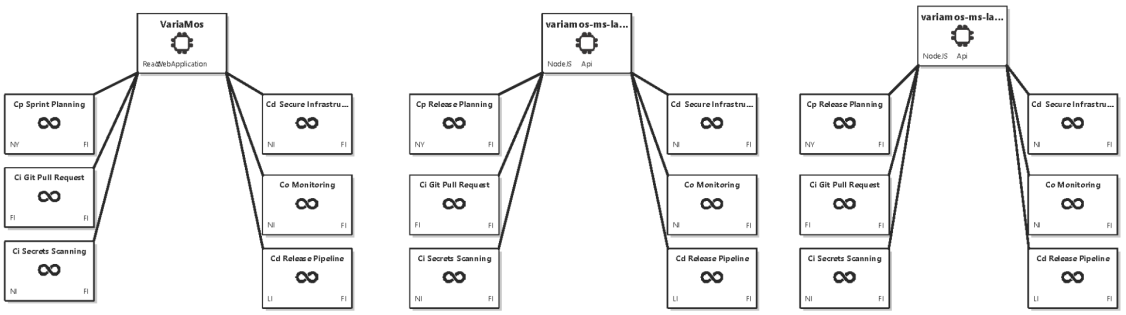

***Figura 11*: DSL Para la adopción de DevOps [25]**

Variamos DevOps adoption spected status

Application: Variamos
Cloud Provider: Azure
Architectural Style: NTierApplication
Model Version: 0.0.1
Date: 11/16/2022 20:12:53

DevOps Practices

VariaMos	Current	Target
Ci Secrets Scanning	NI	FI
Cp Sprint Planning	NY	FI
Ci Git Pull Request	FI	FI
Cd Release Pipeline	LI	FI
Cd Secure Infrastructure Deployment	NI	FI
Co Monitoring	NI	FI

variamos-ms-languages	Current	Target
Cp Release Planning	NY	FI
Ci Git Pull Request	FI	FI
Ci Secrets Scanning	NI	FI
Cd Secure Infrastructure Deployment	NI	FI
Co Monitoring	NI	FI
Cd Release Pipeline	LI	FI

variamos-ms-languages1	Current	Target
Cp Release Planning	NY	FI
Ci Git Pull Request	FI	FI
Ci Secrets Scanning	NI	FI
Cd Release Pipeline	LI	FI
Co Monitoring	NI	FI
Cd Secure Infrastructure Deployment	NI	FI

***Figura 12*: Reporte generado con el DSL para la adopción de DevOps**

Estado actual de la adopción de DeVOps

Comenzamos identificando los diferentes procesos automatizados o las buenas prácticas implementadas. Generalmente lo primero tratamos de identificar es cómo se gestiona el control de versiones de la aplicación, que tipo de sistema de control de versiones se utiliza. En la siguiente lista de buenas prácticas seleccionamos cuales están implementadas, las buenas prácticas relacionadas con el control de versiones las colocaremos bajo la categoría de Integración continua.

Integración continua

- Control de Versiones
- Estrategia de control de versiones
- Metodología de números de versiones
- Etiquetado de versiones
- Estrategia de pull request

El siguiente grupo de buenas prácticas que podemos identificar en los proyectos de desarrollo de software, son las que hacen parte Despliegue continuo. Las buenas prácticas que hacen parte del despliegue continuo son:

- Compilación automática
- Tubería de despliegue
- Gestión de artefactos

Ahora continuemos identificando el estado de adopción de las buenas prácticas relacionadas con DevSecOps, las cuales están relacionadas con los aspectos de seguridad que se deben de tener en cuenta en los procesos de desarrollo de aplicaciones. El siguiente es un lista de buenas practicas de DevSecOps, que podemos evaluar:

- Análisis estático de código
- Modelado de amenazas
- Identificación de amenazas
- Gestión de secretos

El siguiente grupo de buenas prácticas a identificar están en la categoría de Infraestructura como código, El siguiente es un listado de prácticas a identificar.

- Plantillas de infraestructura
- Tuberías de infraestructura

Practice Id	Value
PC1-Create development sandboxes for minimum code deployment	FI
PC2-Automate sandboxes deployment through the development pipeline	LI
PC3-Provide continuous collaboration system in real-time	PI
PC4-Automate testing sandboxes to run in conjunction with development sandboxes	NI
PCN-##############	

Figura 13: **Evaluación de la adopción de DevOps, [25],[24], [10]**

DevOps Adoption Backlog

En este paso se crea Backlog de historias de usuario, en los cuales se detallan los criterios de aceptación que se deben cumplir para su aceptación. Se recomienda que el equipo de desarrollo tenga acceso al backlog para garantizar la visibilidad de las actividades a realizar. Es común que los equipos de TI tengan backlogs separados, pero esto va en contradicción de los marcos de trabajos Ágiles. En la figura 8 vemos un backlog de historias de actividades de DevOps.

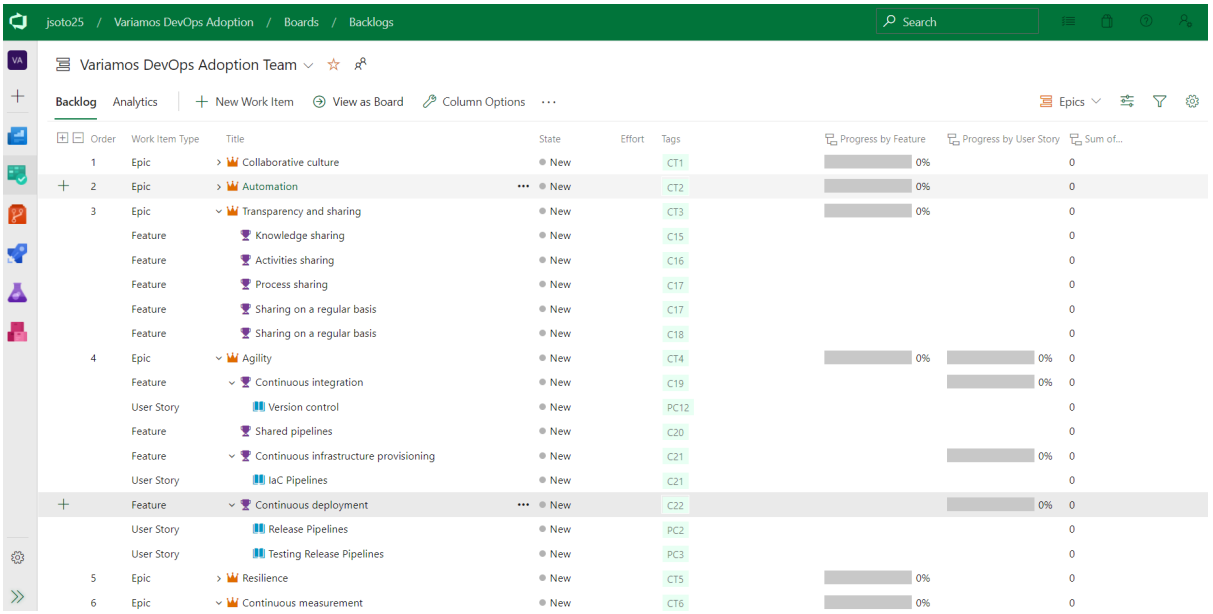

Figura 14: Backlog de historias de usuario de buenas prácticas de DevOps

Definición de los SLOs y los SLIs

En el proceso ADAM las actividades relacionadas con la Ingeniería de Confiabilidad de Sitios (SRE), en las cuales se definen los Indicadores de Nivel de Servicio (SLIs) y los Objetivos de Nivel de Servicio (SLOs) que nos permiten determinar si la aplicación está ejecutándose entre los rangos definidos por los SLOs. Si la aplicación funciona en los rangos no válidos de estos niveles, se hace necesario definir historias de usuario habilitadoras las cuales se deben desarrollar y estas deben de tener el objetivo de cumplir con los SLOs.

Los SLOs básicos que deben alcanzar las aplicaciones web son ***[5]***:

- Disponibilidad
- Latencia
- Presupuesto de errores (máximo número permitido de errores o tiempo de inactividad del sistema) ***[5]***
- Sostenibilidad
- Escalabilidad

Existen algunas buenas prácticas que nos ayudan a seleccionar los SLIs que se van a utilizar para definir los SLOs de una aplicación, de las cuales presentamos 6 de manera concisa ***[5]***:

1. Definir claramente los niveles deseados de: Disponibilidad, desempeño y fiabilidad de la aplicación en términos de SLOs.
2. Determinar los flujos de usuario más críticos para el éxito de la aplicación, debido a que tienen mayor impacto sobre los SLOs.
3. Seleccionar los SLIs que miden precisamente los flujos críticos y los SLOs de la aplicación.
4. Asegurarse de que los SLIs están basados en datos que representan los flujos críticos y los SLOs de la aplicación.
5. Automatizar la recolección de los datos y la medición de los SLIs para asegurar la consistencia y precisión de los datos.
6. Monitorear continuamente los SLIs para asegurar que reflejan acertadamente el desempeño del sistema y el logro de los SLOs.

Aplicación de monitoreo y observabilidad

Cada plataforma de ejecución de aplicaciones, ya sea en la nube o en sitio, requiere de servicios de captura de métricas, logs, trazas y otras, lo cual permite determinar el estado de cada aplicación y obtener información detallada de los errores. Los proveedores de la nube generalmente brindan servicios de logs y de registro de métricas, pero también existen proveedores de servicios de terceros que permiten implementar el monitoreo de las aplicaciones.

En nuestro caso, se implementó el monitoreo de la aplicación web VariaMos con el proveedor de servicios de monitoreo llamado New Relic. Inicialmente se estaba utilizando el servicio de monitoreo del proveedor en la nube Azure, denominado Application Insights, pero luego de evaluar los costos, se tomó la decisión de utilizar New Relic.

Con estas herramientas de monitoreo se puede capturar los siguientes tipos de información de monitoreo: Logs, Traces, Events y Métricas. Es necesario configurar las aplicaciones para recopilar la información de monitoreo. En los servicios de Variamos se implementó la configuración a nivel de los archivos de definición de los contenedores o Docker file.

En la ***Figura 15***, observamos un tablero de monitoreo, para la aplicación VariaMos, en donde se hace seguimiento de las siguientes métricas: Errores de cliente, errores de servidor, peticiones , conecciones a base de datos fallidas, uso de CPU de la plataforma de cómputo y del servicio de base de datos.

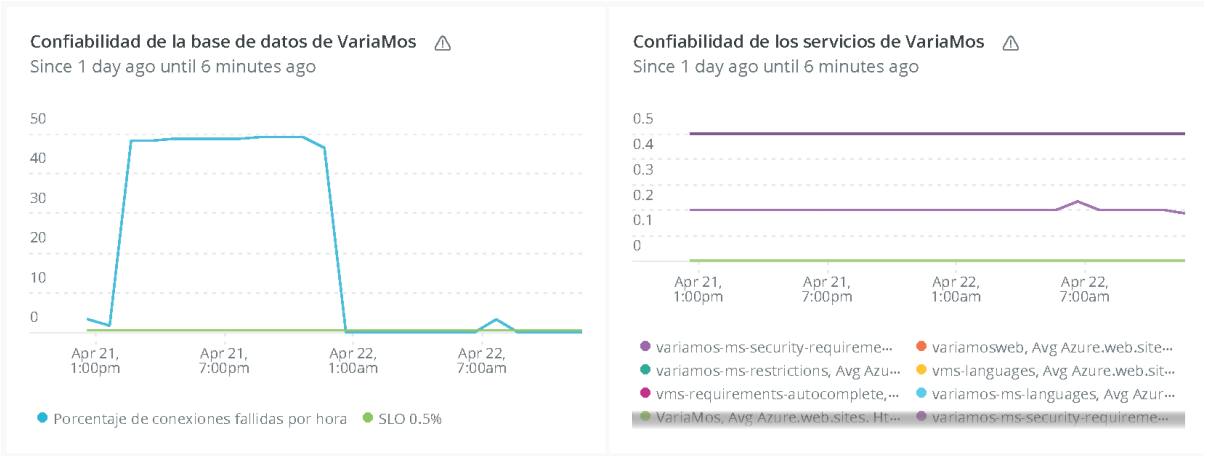

Figura 15: **Tablero de monitoreo implementado en New Relic para la aplicación VariaMos**

6. Evaluación de la propuesta

Para la evaluación del método ADAM utilizado en la adopción de prácticas de de DevOps, nos valemos de la implementación de las prácticas de Ingeniería de Confiabilidad de Sitios (SRE), las cuales nos permiten definir los SLOs, basados en logro de los SLIs (en los niveles requeridos para garantizar la ejecución óptima de la aplicación). A continuación describimos las prácticas de DevOps implementadas y las prácticas de SRE utilizadas para la evaluación.

Las actividades que se implementan para medir la efectividad de la adopción son las siguientes:

1. Seleccionar las buenas prácticas de DevOps que se pueden implementar para solucionar errores, alcanzar objetivos o implementar nuevas funcionalidades.
2. Crear un backlog con las historias de usuario que representan las buenas prácticas definidas de DevOps a implementar definidas en el paso anterior.
3. Definir los Indicadores de nivel de servicio o SLIs, definir los objetivos de nivel de servicio o SLOs. De nivel de servicio.
4. Análisis de los indicadores los vamos a analizar, antes y después de la implementación de las historias de usuario definidas para la implementación de las buenas prácticas de DevOps.
5. Elaboración de recomendaciones que resulten de acuerdo al análisis de la adopción en las buenas prácticas de DevOps.

A continuación presentamos los detalles de cada una de las actividades realizadas para evaluar el método ADAM propuesto en esta tesina.

1. Selección de las buenas prácticas de DevOps

Para la selección de las capacidades y buenas prácticas de DevOps a implementar tomamos como cómo base la lista de capacidades la referencia [23]. Esta lista nos da una visión general de las capacidades y prácticas disponibles para implementar proyectos de desarrollo y abarca diferentes aspectos; por ejemplo, las pruebas, la seguridad, las operaciones y las mejoras al sistema y al proceso.

Inicialmente se seleccionan dos prácticas a implementar: Desarrollo de Contenedores y de Microservicios e Infraestructura como Código (IaC).

Se selecciona la práctica de desarrollo de contenedores y Microservicios, orquestados con Kubernetes debido a que la versión actual de VariaMos, aunque está contenerizada, está siendo orquestada de manera monolítica. Esto último significa que se por cada nuevo servicio se despliega un sitio en un servidor, al cual se le deben de configurar los siguientes aspectos: Acción de GitHub para despliegue, aprovisionamiento del servicio de aplicaciones. Esto toma un tiempo considerable el cual se repite cada vez que se requiere la implementación de un nuevo servicio. Con la propuesta de implementar orquestación con Kubernetes se busca lograr los siguientes objetivos:

- Tener un ambiente de desarrollo local similar al de producción en el cual los desarrolladores puedan probar sus implementaciones antes de pasar a producción.
- Disminuir en un alto porcentaje el tiempo de despliegue de nuevos servicios.
- Disminuir la dependencia de personal o conocimiento requerido en operaciones para los despliegues y aprovisionamiento de nuevos servicios.
- Aprovechar los beneficios generales de la utilización de Kubernetes como orquestador como son: Escalabilidad, la resiliencia y la portabilidad.

2. Definición del backlog

Una vez definidas las buenas prácticas a implementar, procedemos a crear el backlog en la herramienta Azure DevOps utilizada para la gestión ágil de proyectos. Por cada actividad definimos las historias de usuario a implementar y se determinan las siguientes propiedades: Sprint en el que se implementaran, puntuación del esfuerzo, criterios de aceptación los cuales nos ayudan a determinar el estado de la actividad, también se define la prioridad. A medida que se va implementando la historia de usuario, se debe ir actualizando el estado de esta.

La definición de las historias de usuario nos ayudan a medir el plazo de entrega o Lead Time. El cual a su vez se utiliza como SLI para definir el SLO para el Plazo de Entrega.

La definición del SLO que incluye el Plazo de Entrega es:

Métrica: **Plazo de Entrega**
SLO: El 90% de las historias de usuarios se deben de completar en 1 sprint.
Periodo de tiempo: Sprint

3. Implementación de Ingeniería de Fiabilidad de Sitios (SRE)

Debido a que la aplicación Variamos está hospedada en en una plataforma en la nube, se adopta SRE, lo cual nos permite garantizar la fiabilidad y la disponibilidad de la aplicación. Al monitorear y medir el desempeño de la aplicación, se logran resolver las fallas de la aplicación antes de que se vuelvan críticas.

Al implementar SRE, con la definición de los SLO y SLIs, se busca definir un proceso de gestión de incidentes y revisión posterior a estos , garantizando de manera oportuna, la mejora del desempeño de la aplicación, la disminución del tiempo de inactividad y la mejora de la satisfacción del cliente (Investigadores en el area de Ingenieria de Lineas de Productos.).

En el proceso de implementación de SRE, comenzamos a definir los siguientes SLIs: **Latencia**, el cual nos permite medir el tiempo en que una petición es respondida correctamente y se mide en porcentajes; por ejemplo 98% de latencia. **Ratio de error**, este

SLI nos permite medir el porcentaje de peticiones que respondieron con error. **Tráfico**, este SLI permite medir el tráfico de la aplicación.

Continuamos con la definición de los objetivos de nivel de servicio. Los SLOs definidos en el proyecto Variamos son los siguientes: **Disponibilidad**, el nos muestra el tiempo que el servicio debe de estar disponible, con los parámetros deseados. **El presupuesto de error** mide el ratio de error o la latencia permitida en el sistema antes de que se viole el SLO. El **desempeño** mide que tan rápido responde el sistema y se mide en porcentajes de latencia.

La siguiente figura nos muestra los SLIs y los SLOs definidos en el proyecto web evaluado.

Nombre	**SLI**	**SLO**	**Valor**
Latencia	98%, porcentaje de latencia	250 ms	El 98% de la latencia del servicio se espera esté por debajo de 250 milisegundos.
Ratio de error	Porcentaje de peticiones que se hacen sobre los servicios de la API Rest que entregan un código de error (e.g., 400: Código de respuesta de error al acceder recursos o falta de permisos. 500: Código de respuesta con error en el servidor o backend). Código de respuesta diferente de [200..210]	1%	El ratio de error del sistema se espera que esté por debajo del 1%.
Disponibilidad	Porcentaje del tiempo que el sistema está disponible	99.5%	Se espera que el sistema esté disponible el 99.5% del tiempo.
Plazo de entrega de las historias de usuario	Porcentaje de historias de usuario completadas en el sprint	90%	Se espera que más del 90% de las historias de usuario se completen en el sprint.
Confiabilidad de la base de datos	Porcentaje de conexiones a base de datos fallidas por hora.	0.5 %	Se espera que las conexiones a base de datos que se ejecutan en el sistema fallidas estén por debajo del 5% por hora.
Confiabilidad del servidor	Porcentaje de errores de servidor 5xx por hora	0.5 %	Se espera que las respuestas de los servicios de backend que se ejecutan en el sistema fallidas estén por debajo del 5% por hora.

Figura 16: Indicadores de Nivel Servicio, Objetivos de Nivel de Servicio, definidos para monitorear la aplicación evaluada (VariaMos).

4. Implementación de Monitoreo y Telemetría

Para la correcta evaluación de esta propuesta, nos valemos de herramientas de monitoreo y telemetría. Se descartó la utilización de Azure Application Insights, debido a los costos que incurre para el proyecto, sin descartar que es una herramienta que se puede utilizar debido a que el Stack de la aplicación que evaluamos (Variamos) está implementado sobre la plataforma de Azure. Seleccionamos entonces, New Relic, una plataforma de monitoreo y observabilidad que permite recolectar datos de tipo Métricas, Eventos, Logs y Trazas. Adicionalmente permite utilizar el lenguaje para la creación de sentencias denominado NRQL (con características muy similares a un lenguaje de sentencias tipo SQL) el cual puede ser usado para crear queries sobre las diferentes tablas que contienen las métricas recolectadas de los sistemas monitoreados. Los resultados de estas sentencias se pueden visualizar como tablas, registros, gráficos de barras y líneas. Además nos permite monitorear aplicaciones implementadas en los principales proveedores de servicios en la nube y también aplicaciones móviles entre otras.

El proceso general para la implementación de la recolección de datos con la herramienta New Relic, utilizado en la aplicación Variamos es el siguiente:

1. Creación de una cuenta gratuita en New Relic, la cual tiene las siguientes características:
    - Límite de 100 GB
    - 1 usuario completo de plataforma
    - 100 usuarios básicos gratis
    - Alertas ilimitadas
2. Adicionar la la cuenta de Azure donde está hospedada la aplicación web Varamos, y selección de los servicios a monitorear. Se seleccionan los siguientes servicios:
    - Databases for PostgreSQL
    - PostgreSQL Flexible Server
    - App Services
    - Functions
    - Virtual Machines
3. Creación del Dashboard donde se visualizan los gráficos que representan:
    - Errores 4xx
    - Errores 5xx
    - Total requests
    - Comunicaciones a bases de datos fallidas
    - Conexiones activas
    - Uso de CPU
    - SLOs
4. Creación de las alertas para monitorear las fallas en la aplicación y de esta manera proceder a su solución de manera temprana.

Con los pasos implementados en esta propuesta se recolectan las métricas generadas en el proceso de desarrollo, las métricas generadas por los diferentes servicios de la aplicación. Estas métricas se utilizan para implementar los diferentes objetivos e indicadores que ayudaron a medir la fiabilidad de la aplicación, y a definir las prácticas de DevOps, que se

van a implementar para alcanzar dichos objetivos. Adicionalmente se utilizaron prácticas de gestión de proyectos ágiles que ayudaron a visibilizar en el equipo las prácticas implementadas.

7. Resultados

Métricas de medición del proceso de adopción

Se utilizaron las 5 métricas más utilizadas para la gestión de proyectos ágiles las cuales son para monitorear y analizar el progreso de la adopción de las prácticas de DevOps. Estas métricas son:

- Velocidad
- Plazo de entrega o *Lead time*
- Tiempo de ciclo o *Cycle time*
- Gráfico de evolución
- Diagrama de flujo acumulativo

La métrica de velocidad se utiliza para pronosticar cuántos puntos de esfuerzo se pueden implementar en un sprint. Se puntúan todas las historias de usuario de implementación de prácticas de DevOps. Por ejemplo para la adopción de Infraestructura como Código (IaC) en el proyecto la velocidad por sprint es de 11 puntos aproximadamente. Esto se logró determinar después de evaluar más de 20 sprints.

Adopción de prácticas de SRE

Los resultado obtenidos al implementar las buenas prácticas de SRE son los siguientes:

- Mejoramiento de la fiabilidad del sistema. Se definieron dos objetivos de nivel de servicio para verificar este indicador, ver ***Figura 16***, en el cuál el máximo porcentaje de conexiones a base de datos erróneas y los errores de servidor tipo 5.xx deben de estar por debajo del 0.5 % en una hora. Estos indicadores se implementaron en el tablero de monitoreo. ver ***Figura 15***
- Reducción de los tiempos para la solución de incidentes. Esto se logró al implementar las alertas en el sistema de monitoreo, las cuales notifican a los equipos de soporte via correo electrónico, cuando se incumple un Objetivo de Nivel de Servicio.
- Mejoramiento de la información sobre la toma de decisiones. Esto debido a que ya no es necesario que el equipo de desarrollo tenga que informar el error por medio de canales de comunicación con grupos en WhatsApp, Slack, Team. En su defecto el sistema de alertas envía notificaciones al equipo de soporte, cuando ocurren errores o los indicadores u objetivos son sobrepasados en sus límites permitidos. ver ***Figura 17***

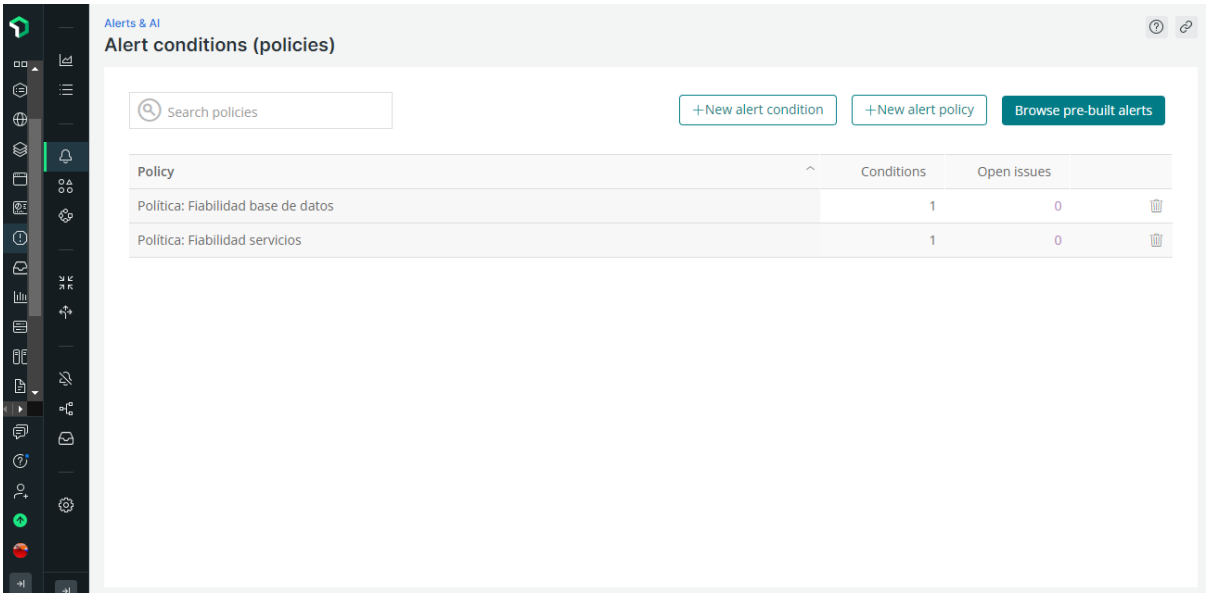

Figura 17: Definición de alertas para cuando se sobrepasan los valores permitidos de los objetivos de nivel de servicio.

Las principales prácticas de SRE que se adoptaron en el proyecto usado en este trabajo de investigación son:

- Monitoreo: Se recolectaron las métricas que representan el comportamiento y el desempeño del sistema.
- Se definieron los objetivos de nivel de servicio, de manera que representen una medición de la fiabilidad del sistema. Uno de los SLO definidos en la aplicación es el siguiente: 95% de las transacciones deben ser correctas. ver ***Figura 16***
- Se definieron las alertas: Se crearon alertas, las cuales se configuran para que notifiquen a buzones de correo, o ejecuten acciones correctivas automatizadas, cuando se sobrepasan valores permitidos en las métricas recolectadas y en los objetivos de nivel de servicio definidos. Fueron creadas alertas para informar cuando ocurrieran errores de conexión a base de datos, o para cuando uno de los servicios no estuviera disponible. ver ***Figura 17***
- Respuesta a incidentes: Se definió un proceso y los respectivos responsables encargados de solucionar los incidentes que afectan la disponibilidad de la aplicación. Con el análisis del presupuesto de errores y las alertas tempranas disparadas por el sistema, el equipo encargado de los incidentes, logra reaccionar de una manera más rápida y oportuna.

Adopción de Infraestructura como código

La adopción de esta práctica trae varias ventajas para el proyecto: Creación de infraestructura repetible y consistente entre los diferentes ambientes, reducción de los errores humanos derivados de procesos manuales de configuración de la infraestructura, colaboración entre los equipos que intervienen en la definición, diseño e implementación de la infraestructura, debido a que se trabaja bajo el mismo codebase. Adicionalmente esta adopción presenta principalmente una desventaja que se evidencia en la curva de

aprendizaje: es lenta debido a las herramientas, lenguajes y gran cantidad de prácticas que se deben implementar.

Para la implementación de esta práctica en el proyecto se siguieron los siguientes pasos:

- Se seleccionó la herramienta para Infraestructura como Código , para este proyecto evaluamos entre ARM, Bicep y Terraform. De estas tres se seleccionó Bicep debido a que es un DSL para infraestructura basada en ARM, que simplifica la sintaxis y mejora la mantenibilidad y la lectura de los templates.
- Desarrollo del código de Infraestructura como código, el cual implicó la creación del cluster, los nodos y los servicios. Este código se almacena en un repositorio de código de Github. Ver ***Figura 3*** con la representación gráfica de la infraestructura en formato .bicep. Ver Código de Infraestructura como código ***[13]***
- Despliegue de la infraestructura utilizando canalizaciones de Infraestructura como código basados en GitHub Workflows.

Adopción de Metodologías Ágiles para gestión del proyecto

Las prácticas que se implementaron fueron:

- Creación de un backlog de historias de usuario con las prácticas de DevOps.
- Implementación de sprints.
- Priorización de la implementación de las actividades y cálculo del esfuerzo requerido para cada actividad.

Los resultados obtenidos fueron los siguientes:

- El equipo se mantuvo informado del proceso de desarrollo mediante el uso de la herramienta Azure DevOps, que permitió gestionar el backlog, los sprints y el registro del tiempo dedicado a cada tarea.
- Se logró una mayor eficiencia y calidad en la entrega del producto final, al seguir los principios y valores de las metodologías ágiles y aplicar las prácticas de DevOps para integrar y desplegar el código continuamente.

Se encontró un reto en el momento de elegir la herramienta ágil adecuada para el proyecto, debido a que los equipos de investigación trabajan con herramientas open source y gratuitas, pero éstas son muy limitadas y se recomienda trabajar con herramientas de pago. Azure DevOps fue la herramienta seleccionada por ofrecer una capa gratuita que incluye todos los componentes que se requieren como: épicas, features, historias de usuario, puntos de historia, registro de tiempo entre otros. Además, se puede integrar con los repositorios de código en GitHub.

Implementación de arquitectura de Microservicios con Kubernetes

Se implementa una arquitectura de Microservicios sobre un orquestador de Kubernetes del proveedor de la nube de Azure. Esta implementación se hace debido a que el stack de

cómputo de la versión anterior estaba implementado sobre unos servicios de aplicaciones a los cuales se les desplegaron las imágenes de Docker con las nuevas versiones. Estos tipos de servicios basadas en Docker son propietarios y los proveedores en la nube tienen servicios similares como por ejemplo AWS Beanstalk y Azure App Services. Estos servicios son muy fáciles de implementar pero tienen muchas desventajas en el momento de probar en ambientes de desarrollo. Por tal motivo se opta por utilizar una solución estándar y de grado industrial, como es utilizar orquestadores basados en Kubernetes, los cuales llevan gran tiempo en la industria a nivel productivo y adicionalmente para los ambientes de desarrollo se puede utilizar Minikube el cual permite tener un ambiente de desarrollo con características similares al de los ambientes productivos.

8. Discusión y validez de los resultados

Existen muchas definiciones de buenas prácticas de DevOps, definidas en la literatura y en la industria las cuales se pueden implementar en proyectos de desarrollo de tecnología de la información. Las cuales sin la utilización de metodologías de ingeniería, imposibilitan la adopción de estas prácticas.

La adopción de la práctica de DevOps, denominada infraestructura como código, ayudó a materializar las buenas prácticas de DevOps en el repositorio de control de versiones. Dicha materialización se usó para mejorar el despliegue de la aplicación usada como caso de prueba de conceptos, además de ser usada para mantener y actualizar el stack tecnológico de acuerdo a las necesidades del proyecto VariaMos. En el proyecto que se evaluó periódicamente se están implementando nuevos servicios, los cuales requieren de actualización o creación de nuevos servicios de aplicaciones en el proveedor de servicios de la nube de Microsoft Azure. La creación de estos nuevos servicios implica actualización de la plantilla .bicep, con los nuevos servicios definidos, y luego mediante comandos de despliegue en Power Shell CLI o Azure Cloud Shell, se implementan los cambios.

La adopción de la práctica de DevOPs de Monitoreo Continuo con la herramienta New Relic y adicionalmente con la implementación de los Objetivos de Nivel de Servicio y los Indicadores de Nivel de servicio, ayudaron a tener un control más apropiado para el manejo de incidentes de la aplicación. Esto debido principalmente a que el equipo de soporte está enterrado en el instante que ocurre los errores y si a esto le sumamos las estrategias que se definieron como por ejemplo los límites mínimos permitidos antes de que se generen las alertas, de esta manera comenzó a optimizar el recursos de soporte de la aplicación. Evitando gastar tiempo en falsos positivos que se pueden generar en el momento que se actualiza una aplicación y no está disponible por tiempo indeterminado.

La creación del backlog de actividades con las herramientas de gestión de proyectos ágiles, ayudó a relacionar las métricas e indicadores de gestión del proyecto, con las métricas e indicadores del comportamiento del sistema en ejecución. Inicialmente las actividades de aprovisionamiento de recursos de ejecución de las aplicaciones como por ejemplo implementar el sitio para un nuevo servicio web, pasó de demorarse unas 4 horas sin ADAM a 30 minutos en la actualidad usando ADAM. Esto debido a que inicialmente este aprovisionamiento se hacía manual, desde el panel de administración, y luego se redujo el procedimiento al solo tener que modificar el script Infraestructura y desplegarlo nuevamente. Estas actividades se registran en el backlog y luego se pudo evidenciar la reducción del tiempo de implementación.

9. Conclusiones y perspectivas

Se evidencio que la definición de las prácticas de DevOps no están estandarizadas, por lo general se encuentran definiciones que resultan de proyectos de investigación o de proveedores de servicios con Microsoft, Google o AWS. Por tal motivo se hace necesario en los proyectos tomarse el tiempo para definir las prácticas de DevOps a implementar y sus definiciones o en su defecto seleccionar algunas de las recomendaciones de los vendedores de servicios.

Utilizar los backlogs de gestión ágil de proyectos ayuda a visibilizar en el equipo estas actividades relacionadas a la adopción de buenas prácticas de DevOps y de Ingeniería de fiabilidad de sitios, con frecuencia los equipos de desarrollo son muy dados a no implementar estas buenas prácticas o también las pueden estar implementando de manera tácita. De tal manera que se vuelve a caer en el problema de la visibilidad que directamente afecta los tiempos de entrega y los presupuestos del proyecto.

Considero que el diseño de la arquitectura de una aplicación a nivel de proveedor de servicios en la nube, utilizando la practica de Infraestructura como código, podría ser candidata a que se le adopte la metodología de línea de productos, principalmente por que se tiene la definición de la infraestructura en código y por que generalmente las aplicaciones en la nube para múltiples clientes o inquilinos, requieren la creación de infraestructura por cada cliente y con diferentes capacidades de acuerdo al valor y beneficios de la subscripción. Lo cual nos da un aspecto de variabilidad que puede ser abordado por las estrategias de derivación que se utilizan para las líneas de productos.

10. Términos en Inglés/Español

Administración de artefactos (Artifacts Management)

La administración de artefactos, se define como el proceso de administrar y almacenar artefactos de compilación en un repositorio central, donde se pueden versionar, rastrear y compartir entre equipos. ***[5]***

Bicep

Bicep is an open-source Domain Specific Language (DSL) for defining and deploying Azure resources in Microsoft Azure. Bicep is designed to simplify the process of creating and managing Azure resources by providing a more concise and readable syntax than Azure Resource Manager (ARM) templates, the previous standard for defining Azure infrastructure. Bicep is also designed to be more flexible and extensible, allowing for the creation of custom modules and templates that can be shared across teams. ***[16]***

Desempeño

La medida de la capacidad de un sistema para responder a una carga de trabajo determinada dentro de un marco de tiempo aceptable, sin experimentar retrasos o errores significativos.***[5]***

Disponibilidad

La proporción de tiempo que un sistema está operativo y funcionando correctamente, durante un período determinado. ***[5]***

Entrega Continua (CD)

Enfoque de ingeniería de software en el que los cambios de software se construyen, prueban y preparan automáticamente para su lanzamiento a producción, en cualquier momento y con una intervención humana mínima ***[5]***.

El etiquetado (Tagging)

Se define como la práctica de asignar etiquetas o marcadores a una versión específica del código, un artefacto de compilación o un entorno de implementación, para permitir un seguimiento, identificación y administración sencillos de estos componentes. ***[5]***

Fiabilidad (reliability)

La confiabilidad se define como la probabilidad de que un sistema realice su función prevista sin fallas, durante un período de tiempo específico, bajo condiciones específicas.

En otras palabras, la confiabilidad es la medida de cuán consistentemente un sistema puede proporcionar la funcionalidad deseada sin experimentar ningún error o comportamiento inesperado. ***[5]***

Flujo de trabajo de control de versiones (Version Control Workflow)

Es un enfoque sistemático para administrar los cambios en una base de código o archivos de proyecto. Implica el uso de un sistema de control de versiones (VCS) para realizar un seguimiento de los cambios realizados en los archivos y administrar estos cambios de manera estructurada. ***[27]***

Infraestructura como Código (IaC)

Es un enfoque para la automatización de la infraestructura donde el aprovisionamiento y la administración de la infraestructura se realiza a través del código y se versiona en un repositorio. Esto permite que la infraestructura se trate como código, de manera similar a cómo se trata el código de la aplicación, lo que permite una mejor colaboración e implementaciones más rápidas y confiables. ***[5]***

La implementación Azul/Verde (blue/green deployment)

La implementación Azul/Verde, se define como una estrategia de implementación que permite la implementación de una nueva versión de una aplicación junto con la versión existente, sin afectar el tráfico de producción. ***[5]***

Integración Continua (CI)

Práctica de ingeniería de software en la que los desarrolladores combinan regularmente sus cambios de código en un repositorio compartido, donde se ejecutan compilaciones y pruebas automatizadas en el código integrado.***[5]***

Latencia

El tiempo transcurrido entre la iniciación de una solicitud y la recepción de una respuesta. ***[5]***

Modelado de amenazas (Threat Modeling)

Es una técnica de seguridad utilizada para identificar posibles amenazas y vulnerabilidades de seguridad en una aplicación de software. El modelado de amenazas implica analizar los diferentes componentes y procesos de una aplicación para identificar las posibles amenazas y riesgos asociados con cada componente.***[17]***

Pipeline

Una serie de pasos automatizados por los que pasan los cambios de código para compilarse, probarse e implementarse en producción. Las canalizaciones son un componente central de la entrega continua y proporcionan un enfoque estructurado y automatizado para la entrega de software. ***[5]***

Plazo de Entrega

Plazo de Entrega o Lead Time, es una medida del tiempo desde una solicitud de cambio hasta la entrega del cambio.

Presupuesto de errores

La cantidad máxima de errores aceptables o tiempo de inactividad que un sistema puede experimentar durante un período de tiempo específico, sin violar el acuerdo de nivel de servicio (SLA) o las expectativas del cliente. ***[5]***

Pruebas unitarias (Unit Testing)

Son un tipo de prueba de software que implica probar unidades o componentes individuales de una aplicación de forma aislada para garantizar que funcionen correctamente. ***[5]***

Ratio de error

El número de solicitudes que producen errores (como códigos de estado HTTP 5xx) dividido por el número total de solicitudes recibidas durante un período de tiempo determinado. ***[5]***

STRIDE

Significa: ***[27]***

- Suplantación de identidad del usuario
- Manipulación de datos
- Repudiabilidad
- Divulgación de información (violación de la privacidad)
- Denegación de servicio (D.o.S.)
- Elevación del privilegio

SRE

La ingeniería de confiabilidad del sitio (SRE) es una disciplina que incorpora aspectos de la ingeniería de software y los aplica a problemas de infraestructura y operaciones. Los principales objetivos de SRE son crear sistemas de software escalables y altamente confiables, aumentar la eficiencia operativa y proporcionar un puente entre los equipos de desarrollo y los equipos de operaciones. Los SRE son responsables de la disponibilidad, latencia, rendimiento, eficiencia, gestión de cambios, supervisión, respuesta a emergencias y planificación de la capacidad de los servicios que admiten. Utilizan un conjunto de prácticas, como automatización, supervisión y autopsias irreprochables, para garantizar que los servicios cumplan sus objetivos de nivel de servicio (SLO) y acuerdos de nivel de servicio (SLA). SRE no es un título de trabajo, sino un conjunto de principios y prácticas que se pueden aplicar a cualquier organización, independientemente de su tamaño o industria. ***[5]***

Tráfico

La cantidad de datos transmitidos entre un cliente y un servidor durante un período de tiempo determinado ***[5]***

Referencias

[1] Ghantous, Georges Bou and Gill, Asif, "DevOps: Concepts, Practices, Tools, Benefits and Challenges" (2017). PACIS 2017 Proceedings. 96. https://aisel.aisnet.org/pacis2017/96

[2] Introduction to DevOps with Chocolate, LEGO and Scrum Game Dana Pylayeva ,Brooklyn, New York, ISBN-13 (pbk): 978-1-4842-2564-6 ISBN-13

[3] Soto Jairo, DevOps Adoption Backlog

[4] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger & Samir Chatterjee (2007) A Design Science Research Methodology for Information Systems Research, Journal of Management Information Systems, 24:3, 45-77, DOI: 10.2753/MIS0742-1222240302

[5] Site Reliability Engineering, Betsy Beyer, Chris Jones, Niall Richard Murphy, Jennifer Petoff

[6] The Art of Agile Development, 2nd Edition

[7] Repositorio de código fuente en GitHub variamosple/DevOps, Jairo Alberto Soto Velásquez

[8] DevOps Adoption Strategy Wiki, Jairo Alberto Soto Veláquez

[9] Artículo en LinkedIn: Adopción de estrategias de DevOps en un aplicación Web de modelado de líneas de productos

[10] Canal de YouTube, DevOps Central, Jairo Alberto Soto Velásquez

[11] VariaMos Monitor Dashboard

[12] VariMos, Service Level Objectives, Dashboard.

[13] VariaNos, DevOps git repository

[14] Variamos

[15] Atlassian Git Tutorials

[16] Bicep documentation

[17] DevSecOps: Building a Secure Continuous Delivery Pipeline

[18] Exam AZ-500 Microsoft Azure Security Technologies (Video)

[19] The DevOps Handbook

[20]-Devops, A New Approach To Cloud Development & Testing

[21]-Mobile DevOps

[22] Beginning MLOps with MLFlow

[23] Introduce the foundation pillars of DevOps Culture and Lean Product

[24] DevOps Best Practices

[25] Upgrade Team, S. (2011). *Standard CMMI ® Appraisal Method for Process Improvement (SCAMPI SM) A, Version 1.3: Method Definition Document*. http://www.sei.cmu.edu

[26] PRINCIPLES OF DEVOPS

[27] Pro Git, Second Edition

[28] Semantic Versioning 2.0.0

[29] Kohnfelder, Loren, and Praerit Garg. "The threats to our products." Microsoft Interface, Microsoft Corporation 33 (1999).

Lista de figuras