Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Desarrollo y Operaciones (DevOps)

Despliegue de un entorno CI/CD en una aplicación Startup utilizando Azure DevOps

Trabajo fin de estudio presentado por:	Jhonathan Lechon
Tipo de trabajo:	Tesis
Director/a:	Oscar San Juan Martinez
Fecha:	21/07/2022

Resumen

Las empresas del área de desarrollo de software constantemente van innovando su forma de trabajo y mantener una visión ágil ayuda a la empresa a tener una ventaja competitiva, por este motivo la cultura DevOps proporciona un enfoque eficiente y flexible para poder cumplir con estos requisitos, apoyándose para esto de la metodología ágil y cultura de trabajo en equipo, equilibrando las responsabilidades de manera más uniforme y adoptando la cultura de innovación y cambio. DevOps actualmente está siendo utilizado en grandes empresas con una madurez avanzada, sin embargo, gracias a la existencia de cursos y diversas herramientas, DevOps está teniendo acogida también en pequeños proyectos y Startups.

El presente trabajo tiene como objetivo la implementación de un entorno CI/CD basado en DevOps para un producto software creado por una empresa emergente (Startup). Para esto se ha utilizado una plataforma en la nube intuitiva y de fácil adopción llamada Azure DevOps, la cual nos brindó una serie herramientas para todo el ciclo de vida DevOps.

Otro de los objetivos también es presentar esta herramienta como una alternativa viable para muchas Startups o pequeños equipos de trabajo que desean empezar un proyecto software utilizando las practicas DevOps y aun no tienen claro cuál de las opciones se puede implementar en su forma de trabajo.

Palabras clave:

- DevOps
- Startups
- Azure DevOps
- CI/CD

Abstract

Companies in the software industry are constantly innovating their way of working and maintaining an agile vision lets the companies to have a competitive advantage, for this reason the DevOps culture provides an efficient and flexible approach to get these requirements supported by the agile methodology and teamwork culture, balancing responsibilities more equitably and adopting the culture of change and innovation. DevOps is currently being used in a lot of big companies with advanced maturity, however, the existence of a lot of courses and new tools, DevOps it is being welcomed in small projects and Startups.

The objective of this work is implementing a CI/CD environment based on DevOps for a software product created by a Startup. To get this goal we have used an intuitive and easy-to-adopt cloud platform called Azure DevOps, which provided us with a lot of tools for all the DevOps life cycle.

Another goal is also to present this tool as a viable alternative for many Startups or small work teams that want to start a software project using CI/CD or DevOps practices and are still not sure which of the options can be implemented in their way of working.

Keywords:

- DevOps
- Startups
- Azure
- CI/CD

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

Esta investigación surge de la necesidad que poseen muchas empresas emergentes sobre sus productos software, los cuales requieren mayor rapidez de desarrollo, capacidad de cambio y optimización de recursos, lo cual se pretende lograr con la implementación de entorno de integración y despliegue continuo (CI/CD) basado en prácticas DevOps obteniendo así los beneficios que esto con lleva.

Actualmente Muchas empresas emergentes están adoptando DevOps y la prueba de éxito se puede escuchar en muchos lugares animando más su adopción para otras empresas o tipos de proyectos. Para lograr la implementación se hará el uso de un aplicativo simple desarrollado en NetCore y con la plataforma de nube llamada Azure DevOps la cual es un entorno que contiene un conjunto de herramientas para la adopción de DevOps que actualmente existe en el mercado. En esta plataforma se hará el versionamiento del código, creación de ambientes de desarrollo, Pipelines, entorno de integración y despliegue continuo y finalmente automatización de pruebas de tal forma que podamos obtener ciclos de desarrollo reducidos incrementando la frecuencia de despliegues y finalmente entregando un producto de calidad y rápido.

Los resultados obtenidos nos permitirán conocer la viabilidad de la implementación de CI/CD aplicando la cultura DevOps en pequeños proyectos o empresas emergentes, también nos permitirá presentar a la plataforma Azure DevOps como una alternativa a utilizar para este tipo de proyectos.

1.1.Justificación del trabajo

Una de las causas principales para el fracaso de las empresas emergentes en el área del software es la falta de una metodología clara a seguir, lo cual deriva en problemas con los equipos de trabajo, ya sea por mala comunicación entre los miembros o elección errónea del equipo de desarrollo o la poca armonía entre estos miembros de equipo, lo que da como consecuencia que se entreguen productos y servicios de mala calidad como consecuencia de la falta de enfoque en el cliente y la incapacidad de entrega continua de productos al cliente. La falta de innovación y adopción de nuevas tecnologías no les permite mantenerse competitivo ante otras empresas.

Otro de los mayores desafíos que una Startups pude presentar es la falta de presupuesto para lo cual es necesario que mucho de sus empleados trabajen en dos o varias áreas de trabajo al mismo tiempo, involucrándose en distintos ciclos de vida del desarrollo de software.

Una de las prácticas de DevOps es la automatización, configuración y despliegues continuos, características que van de la mano con las necesidades de la cualquiera empresa, especialmente de las Startup, esto con el fin de estandarizar el trabajo, reducir costos y maximizar la eficiencia.

En base a esta problemática, se plantea realizar la implementación de CI/CD utilizando las practicas DevOps en un proyecto sencillo creado por una Startup, en la cual se propondrá el uso del entorno Azure DevOps de Microsoft utilizando metodologías agiles y así poder obtener sus diversas funciones y ventajas en comparación con el modelo tradicional que actualmente se maneja.

1.2. Planteamiento del problema

Hoy en día existe una gran demanda de productos tecnológicos, de la misma forma una gran cantidad de oferta por parte de la competencia, por lo cual muchos emprendedores de Startups tienen que luchar para poder sobresalir con un producto, establecerlos y así poder expandirse poco a poco. Existen muchos casos de éxitos que han logrado convertirse en empresas reconocidas y estables, sin embargo, la realidad es que, en su mayoría fracasan, ya sea por mala gestión o falta de apoyo e innovación (*El Método Lean Startup*, 2012).

Uno de los principales retos de una empresa emergente es su falta de personal especializado y poco presupuesto por lo que muchas de estas terminan fracasando, generalmente por no tener una idea clara de cómo optimizar sus recursos, estudios de mercado o un marco de trabajo que les permita manejar de mejor manera sus recursos y habilidades.

Muchas personas que empiezan con un emprendimiento y un producto software no tienen aún claro si deben o no hacer la transición a DevOps, esto puede ser porque se piensa que se necesita de un equipo amplio y recursos costosos para poder implementarlos.

En el caso propuesto, al ser una demostración de aplicación, esta será sujeta a muchos cambios, dependiendo de las necesidades específicas de cada cliente, por lo cual es fundamental contar con una cultura de cambio constante, los cambios a realizar pueden ser muy tardíos de la forma tradicional en cascada que actualmente se está realizando.

El problema que se plantea solucionar es implementar un entorno de integración y despliegue continuo (CI/CD) siguiendo en las prácticas y herramientas DevOps, de tal manera que, basándonos en esta metodología se puedan obtener y verificar sus beneficios además de que nos permitan implementar cualquier cambio de manera ágil y permitiendo formar las bases para la evolución de otros productos software, no solo a nivel del aplicativo, sino también con la forma de trabajo del equipo. La implementación de CI/CD se lo realizará en un aplicativo de tipo REST el cual está basado en microservicios, este aplicativo fue desarrollado por una Startup como un producto a ofrecer a diferentes clientes en forma de demostración, de tal manera, en base a los requerimientos y necesidades de los clientes, se puedan realizar diferentes cambios y adaptaciones necesarias.

Muchas veces debido a la dificultad o desconocimiento de cómo llevar a cabo la configuración de entornos de CI/CD y el seguimiento correcto de una metodología se prefiere optar por trabajar de manera tradicional, sin embargo, actualmente existen varias herramientas que nos permiten manejar de mejor manera todo el ciclo de vida del software sin tener que invertir grandes cantidades de dinero y largos tiempos de configuración.

Para esta implementación, se utilizará la plataforma en la nube de Microsoft llamada Azure DevOps, esto debido principalmente a los conocimientos previos de los miembros del equipo, su fácil adopción y su utilidad en este tipo de proyectos.

1.3. Estructura de la memoria

En el primer apartado se expone el resumen y la introducción, en donde se presenta de manera rápida el tema a resolver en base a las necesidades y problemas encontrados, y partiendo de eso con esto se analiza la justificación y el planteamiento del problema junto con su propuesta de solución.

Como segundo apartado revisaremos el estado del arte, para lo cual se realizó una investigación sobre los temas más relevantes referentes a DevOps y su relación con la metodología Ágil, adopción de DevOps, CI/CD, Pipelines, Azure DevOps y demás complementos que serán la base teórica para la realización de proyecto.

En el siguiente capitulo y con las bases teóricas obtenidas, se plantea los objetivos generales y específicos del proyecto, así como la metodología de estudio que nos permitirá cumplir con estos objetivos planteados.

Continuaremos con la implementación del caso, en el cual, se realizará primeramente un análisis del estado actual del aplicativo y su posterior migración. El aplicativo fue proporcionada por la Startup, la cual fue inicialmente realizada de una forma tradicional, sin seguir una metodología clara. En esta aplicación implementaremos el marco de trabajo DevOps, con el entorno de nube llamada Azure DevOps, esto debido al conocimiento previo de los integrantes del equipo de trabajo sobre tecnologías Microsoft, además que esta herramienta nos proporciona una fácil y rápida adopción y documentación.

En el siguiente capitulo continuaremos con la demostración de una implementación de un Sprint de desarrollo a tal aplicación utilizando ahora herramientas de la metodología agile, se realizará las comparaciones respectivas con su esquema tradicional y con el nuevo enfocado a DevOps. En este apartado también incluiremos la creación de pruebas automatizadas.

En el último capítulo, luego del análisis y los resultados obtenidos previamente, se expondrán las conclusiones de la investigación, junto con sus recomendaciones y prospectivas para investigaciones futuras.

2. Contexto y estado del arte

En este capítulo se investigará y profundizara sobre el conocimiento de los temas relacionados a DevOps, CI/CD, Azure DevOps y temas términos que nos servirían como fundamento teórico para el proyecto a realizar.

2.1. Desarrollo y Operaciones DevOps

Hoy en día el desarrollo de software ha tomado una importancia tan significativa que, prácticamente todo tipo de empresa utiliza algún tipo de software para su funcionamiento y operaciones comerciales. Esto ha permitido que la industria de creación de software vaya evolucionando con forme las necesidades, creándose continuamente nuevas formas optimizar y mejorar su desarrollo, esto con el fin de obtener un producto fiable, útil y seguro para todas sus operaciones.(Neeraj Mishra, 2022)

Uno de los aspectos clave para una empresa es la optimización del desarrollo de software, con el fin de cumplir con los requerimientos del cliente, en el menor tiempo posible y sin perder la calidad del producto.

En este sentido la rapidez de desarrollo de producto y su calidad son la clave del éxito del software ya que se puede responder de manera rápida ante las necesidades y requerimientos del cliente obteniendo así una ventaja competitiva.

Ante estas necesidades, ha tenido relevancia un modelo de desarrollo llamado Desarrollo y Operaciones (DevOps), este término nace en 2008 en una conferencia sobre la metodología Agile por Patrick Debois (Debois, 2008) que viene del término en inglés “Develops and Operators”, lo que quiere decir que es la representación de los desarrolladores de software, personal de control de calidad (QA) y aquellos encargados de desplegar y dar mantenimiento a los aplicativos en producción. DevOps tiene como objetivo principal, la entrega rápida del producto software, la comunicación constante con el cliente y la interacción entre los equipos de desarrollo y Operaciones. (Mishra & Otaiwi, 2020a)

2.1.1. Desarrolladores vs Operadores

Las empresas tradicionales generalmente tienen bien definidos sus equipos de trabajo de acuerdo con sus funciones y actividades y, por lo tanto, se suele tener segmentados los equipos de desarrollo, cuales son responsables del desarrollo y codificación de nuevas características aplicaciones y requerimientos del cliente, control de calidad (QA) y operaciones quienes son responsables de administrar y monitorear el software en producción. Estos equipos generalmente encuentran ubicado en diferentes instalaciones o departamentos habiendo un claro distanciamiento entre estos. La separación de estos equipos ha permitido que actualmente aún existan brechas entre este personal, siendo posible su comunicación muchas veces solo a través de un líder de proyecto o correo electrónico. Estos equipos poseen cada uno sus respectivas características, procesos, herramientas y metodologías de trabajo de forma independiente. (Leite et al., 2019)

La comunicación entre estos equipos solía ser a través de Tickets de soporte o correos electrónicos, lo cual hacía que los operadores o personal de QA exigían la resolución de tales Tickets y los desarrolladores exijan la implementación de las diferentes versiones del software que acaban de elaborar. Hay que tomar en cuenta también que sus motivaciones son distintas, el equipo de desarrollo trabaja por el cambio y evolución del aplicativo, mientras el equipo de operaciones trabaja por la estabilidad del sistema.

Esta iteración si bien ha funcionado a lo largo de estos años, también ha ocasionado largas demoras entre las actualizaciones e implementación de los nuevos cambios del software, así como problemas de comunicación y deficientes resoluciones de problemas en los que cada equipo culpa al otro por las causas ocasionadas aumentando los conflictos y distanciamiento entre estos. (Leite et al., 2019)

Debido a este problema, se planteó la idea de que ambos equipos empiecen a trabajar entre sí, involucrándose cada uno de cierta manera en los roles del otro equipo, con el fin de mejorar la comunicación entre estos y puedan brindar soluciones de una manera más rápida, apelando a la empatía y compañerismo, y para lograr esto, es necesario alinear los objetivos de estos equipos, para lo cual será necesario implementarse la metodología ágil también en el equipo de operaciones. (Hüttermann, 2012)

2.2.METODOLOGÍA ÁGIL Y DEVOPS

Las metodologías agiles de desarrollo de software tienen un origen en los años 90, luego de diversos problemas con las metodologías tradicionales de ese entonces permitiendo adaptar rápidamente el desarrollo de una aplicación de software, respondiendo a cambios y nuevas necesidades que van surgiendo durante el ciclo de desarrollo, permiten la integración de los equipos y propone la auto organización. (Garcia et al., n.d.)

Sus bases se guían del llamado Manifiesto Ágil, que propone una serie de valores entre los que destacan, la valoración del individuo y su interacción en el equipo sobre las herramientas y proceso, el enfoque en el desarrollo sobre la documentación, la colaboración del cliente con el equipo, la respuesta rápidos a cambios y nuevas necesidades, entre otros.

Existe una gran cantidad de metodologías basadas en Ágil que se guían en el manifiesto Ágil, pero cada una tiene sus propios enunciados y marcos de trabajo haciendo mayor énfasis en algunos aspectos específicos. Muchas de estas metodologías han sido utilizadas y probadas con éxito en diferentes proyectos, pero existen algunas que aún no son muy conocidas o utilizadas, esto debido a que no son muy conocidas en nuestro entorno y por falta de mayor difusión o reconocimiento, otro factor es su gran parecido con otras metodologías agiles ya conocidas(Garcia et al., n.d.).

Los marcos de trabajo basados en la metodología agiles más conocidos son:

- Kanban
- Scrum
- Extreme Programming (XP)
- Lean
- Adaptative Software Development
- Crystal Methodologies

DevOps se basa en prácticas agiles de desarrollo de software, principalmente en la Integración Continua (CI) y Despliegue Contino (CD) por lo que es evidente que existe una relación con las prácticas y principios de Agile, por esta razón, muchas de las empresas que trabajan con la metodología Ágil encuentran a DevOps muy familiar y fácil de implementar ya que están impulsados por objetivos y valores básicos similares.

En la siguiente tabla podemos ver algunas de sus características y similitudes:

Tabla 1 Tabla comparativa DevOps y Agile

Metodología Ágil	**DevOps**
Derriban las barreras entre el equipo de desarrollo y el cliente, con el fin de conocer sus necesidades y prioridades comerciales y ver una solución juntos	Requiera que se rompas los muros entre el equipo de desarrollo y el de operaciones, de tal forma que se obtenga una estrecha colaboración entre estos dos equipos
Brindan más decisiones al equipo de desarrollo sobre el cómo se debería funcionar el producto software.	Brinda decisiones sobre cómo se configura y se pone en ejecución el sistema total a nivel de desarrollo y operaciones.
Se aplica generalmente al equipo de software o equipos pequeños bien diferenciados.	Requiere que sea aplicado a todo el equipo implicado en la elaboración y despliegue de la aplicación
Se enfocan en el desarrollo rápido e incrementales, recompilan información para poder corregir cualquier problema rápidamente utilizando el trabajo de equipo y la organización como la herramienta principal.	
En ninguno de los dos casos, son objetivos finales de una empresa ya que ambos constituyen movimientos culturales que permiten a la organización utilizar métodos y herramientas que permitirán lograr los objetivos planteados de una forma ágil y rápida, sin perder calidad.	

La relación entre DevOps y Agile llega a tal punto que existen casos en los que se ha demostrado que existen mejoras en la utilización de Scrum cuando se usa DevOps. (Lwakatare et al., 2016)

Los principios y prácticas del desarrollo de software Agile son necesarias para la adopción exitosa de DevOps, sin embargo, a DevOps, a diferencia del desarrollo ágil, no cuenta con metodologías como Scrum o XP, sino que este en cuenta con un gran conjunto de prácticas a partir de las cuales se puede identificar patronos y se los puede aplicar en diferentes entornos en función de sus necesidades.(Cukier, 2013)

2.2.1. Scrum

Scrum es un marco de trabajo basado en las practicas Agile utilizado para la gestión y seguimiento de desarrollo de proyectos, se puede decir que es una estrategia flexible de desarrollo de productos (Sachdeva, 2016). Scrum tiene como objetivo principal maximizar la productividad del equipo de trabajo garantizando la comunicación e interacción entre miembros del equipo reduciendo al máximo actividades que no aporten con el objetivo principal del proyecto y permite presentar resultados en cortos periodos de tiempo.

Scrum es uno de los marcos de trabajo basados en Agile más populares que permite que los equipos participen y sean autoorganizados, permite exponer cualquier problema que estén afrontando y entre todos solucionarlo. Scrum maneja un conjunto de artefactos que ayudan a implementar este Framework y es por este motivo que existen un sin número de herramientas que facilitan su implementación en los entornos de trabajo. (RIVERA CARO, 2019)

2.2.1.1. Roles Scrum

En Scrum se define diferentes roles centrales, el cual debe estar formado por personas que estén completamente comprometidas con el proyecto ya que serán los responsables de implementar correctamente el marco de trabajo. En el equipo Scrum podemos ver los siguientes roles:

- **Product Owner**: Su responsabilidad es liderar la planificación, ejecución y desarrollo del producto maximizando el valor del producto, también es el encargado de gestionar los requisitos con el cliente, es decir, representa la voz del cliente.
- **Scrum Master:** Se encarga de facilitar, guiar y asegurar que el equipo Scrum cuente con todas las facilidades para cumplir con sus objetivos.
- **Equipo:** Se encargan de desarrollar y entregar los diferentes requisitos expuestos por el Product Owner.

2.2.1.2. Artefactos

Son elementos que nos permiten la transparencia y el registro de la información del proceso de Scrum, estos recursos incrementan y mejoran la productividad y calidad del proyecto. Los artefactos Scrum son:

- **Product Backlog:** Es un listado de los requerimientos del producto que están siendo desarrollados y que se pretende desarrollar. Se presenta como una visión panorámica de las actividades a implementar ordenado por prioridades y con su estimación respectiva.
- **Release Plan:** Describe las metas de cada despliegue, permite planificar e indicar a las demás personas cuando se pretende desplegar los elementos planificados a producción.
- **Historias de Usuario:** Es la explicación general no formal y corta de una función o requerimiento expresada desde la parte del cliente o usuario final.
- **Burndow Charts:** Expresa de manera grafica que nos permite ver el estado del progreso de un Sprint, relaciona el trabajo realizado con el tiempo faltante.
- **Épicas:** Se trata de una funcionalidad de alto nivel que aún no ha sido refinada o detallada debido a su gran tamaño, es de donde se desglosa las diferentes actividades a realizar. También puede ser visto como un agrupador de muchos cambios y requisitos a aplicar dentro de un módulo de una aplicación.
- **Feature:** Es un elemento que nos permite agrupar los diferentes cambios dentro de un módulo de una aplicación, es un nivel menor a las Épicas.

2.2.1.3. Ceremonias

En Scrum se definen las siguientes ceremonias a llevar a cabo durante el proceso de desarrollo (Toromoreno, 2021):

- **Sprint:** Son eventos de larga duración en la que todo el equipo realiza sus respectivas tareas con el fin de incrementar el valor del producto. La duración de cada Sprint depende de cada proyecto, pero generalmente es de 2 a 3 semanas y cada Sprint empieza cuando se termina el otro.

- **Sprint Planning:** Es una ceremonia que se da al inicio de cada Sprint, en este evento se definen las metas, historias de usuario, estimaciones y alcances de del trabajo a realizar.
- **Daily Scrum:** Es una ceremonia que se la realiza todos los días laborables del proyecto, en esta se expresan brevemente los avances obtenidos, las actividades en las que se encuentra trabajando cada miembro del equipo y sus posibles complicaciones.
- **Sprint Review y Retrospectiva:** Reuniones que se las realiza al finalizar cada Sprint, en estas se lleva a cabo la revisión de los avances realizados, las buenas prácticas utilizadas en el Sprint y los compromisos a realizar, esto con el fin de mantener al equipo en constante mejora.

La interacción entre Scrum y DevOps permite tener equipos multifuncionales orientados al negocio que realizan sus entregas de forma continua y permite que su interacción y trabajo fluya sin problemas entre todas las partes interesadas. Los equipos Scrum son encargados de entregar el producto software y los equipos de operaciones de mantener y desplegar el entorno que permita desplegar el aplicativo de una forma segura y optima.(West & Groll, 2017)

2.3.CULTURA DEVOPS

Desarrollo y Operaciones (DevOps) es la evolución del desarrollo Ágil, actualmente se está convirtiendo en una parte esencial de la industria del software por todos los beneficios que su implementación con lleva.

DevOps propone un conjunto de prácticas Ágiles con el fin de permitir una entrega continua de software en ciclos cortos y automatizados, propone también romper las brechas entre los desarrolladores y operadores y de esta forma hacer que trabajen colaborativamente, todo esto con el fin de entregar software seguro, confiable y de alta calidad. Para lograr todos estos objetivos, se hace un seguimiento en prácticamente todo el ciclo de desarrollo del software desde el desarrollo, implementación y mantenimiento del software hasta el monitoreo constante del estado del aplicativo. (Hüttermann, 2012)

Entre las principales características que engloban la implementación de DevOps (Willis, 2010) podemos ver:

- **Cultura**: Mayor importancia de las personas sobre los procesos y herramientas.
- **Automatización**: Su importancia para optimizar los tiempos, reducir redundancias y obtener retrospectivas.
- **Medición y Calidad**: Si no se puede medir, no es posible mejorarlo, por este motivo en DevOps es necesario realizar mediciones a todos los procesos, personas y rendimiento en general.
- **Comunicación**: Crea una cultura en la cual todas las personas involucradas puedan compartir sus puntos de vista, ideas y conocimiento entre todos los miembros del equipo de trabajo.
- **Despliegue e integración continua**: Una continua retrospectiva, despliegue y pruebas del producto ayuda a aprender de los errores y responder con mayor eficiencia a cambios.

Algunas de las ventajas al implementar DevOps en las Empresas son:

- Software Optimo y de calidad, ya que se facilita el trabajo entre los diferentes equipos y se tiene mejor retrospectiva del cliente y sus requerimientos.
- Integración de los equipos, comprendiendo las necesidades de cada equipo, compartir sus experiencias y conocimientos, se combinan procesos, habilidades, metodologías y técnicas. De igual forma se pretende sacar el mayor provecho a los conocimientos y destrezas de cada miembro del equipo.
- Optimiza y automatiza procesos repetitivos, de tal forma que se pueda aprovechar de una mejor manera los recursos humanos y tecnológicos.

2.3.1. Adopción de DevOps

La adopción de DevOps puede realizarse de distintas formas y no existe un manual especifico de cómo realizar este proceso, sin embargo, existen muchas guías y casos de éxito que pueden ayudarnos como guía de su implementación, es fundamental contar con todo el equipo involucrado y además que tengan conocimiento de lo que se va a realizar y los resultados esperados.

Algunas de estas estrategias según (Redacción KeepCoding, 2022) y los pasos a considerar pueden ser:

1. **Planificación**. – Se necesita realizar un plan de los diferentes pasos a seguir, su estimación de tiempos, estrategias y acciones.
2. **Definición de herramientas**. - Dependiendo las capacidades del equipo, las necesidades del proyecto y el presupuesto, en esta fase se seleccionan las herramientas y aplicaciones a utilizar teniendo como eje la automatización de procesos.
3. **Capacitación de los equipos de trabajo. –** Es muy importante que el personal este al tanto de una manera general de esta implementación, deben conocer sus beneficios, objetivos y los resultados esperados. En esta fase se crean los diferentes equipos de trabajo y se la capacidad dependiendo sus funciones asignadas, pero además se necesita que compartan sus conocimientos, experiencias con los otros los miembros del equipo. La idea es que cada persona tenga conocimientos solidos de su área de trabajo y también tenga nociones sobre las actividades que los otros equipos realizan. Para lograr estos objetivos, se podría hacer la implementación de un Dojo DevOps, en el cual se plantea el aprendizaje continuo y técnicas para elevar las habilidades del equipo.
4. **Feedback. -** Se plantea una constante retroalimentación (FeedBack) de cada proceso entre los diferentes equipos de tal forma que se pueda notar los efectos de cada acción que se está realizando, esto flexibiliza la comunicación entre los equipos y se espera una adopción al cambio y mayor eficacia.
5. **Automatización de Pruebas. -** La automatización de pruebas permitirá acelerar el proceso de verificación del software, haciendo posible encontrar y corregir de manera más rápida los errores.
6. **Métricas. –** Obtener métricas de cómo funciona el producto, desde su desarrollo hasta su implementación en producción nos permitirá detectar problemas de rendimiento y seguridad, permitiéndonos también poder evaluar los procesos y obtener Feedback.
7. **Cultura de confianza y aprendizaje. –** Se basa en la motivación y recompensa a nuestro equipo de trabajo, se pretende mantener una colaboración y compañerismo de todo el equipo de trabajo. También se motiva el aprendizaje, capacitaciones, experiencias obtenidas y la resolución de conflictos.

8. **Fallos controlados. –** Se sugiere la inyección de fallos controlados en producción, ya sea de rendimiento o seguridad, esto con el fin de ver su comportamiento y aprender el cómo solucionarlo para estar preparados en casos reales.

2.3.2. Desafíos de la implementación de DevOps

La implementación de DevOps dependiendo del tipo de proyecto y empresa, puede requerir de mucho tiempo y ser muy costoso, sin embargo, la inversión se puede justificar con todos los beneficios que se obtendrán con su correcta adopción. Existen muchos retos en la adopción de DevOps, los cuales van desde el cambio de mentalidad, herramientas y procesos y metodologías a la que una empresa ya estaba acostumbrada.

Otro aspecto que puede interferir en su correcta adopción es la falta de personal debidamente capacitado ya que es necesario que tengan comprensión de la infraestructura, configuración, implementación y monitoreo, es decir, todo el ciclo de vida del producto software. Si bien esto tiene solución simplemente con practica y capacitaciones, se debe tomar en cuenta que suele existir resistencia al cambio e incertidumbre, por lo que es fundamental la motivación y liderazgo. (Senapathi et al., 2018)

2.4. DEVOPS Y LAS STARTUPS

Se les llama Startups (también conocidas como Empresas Emergentes) a aquellas empresas pequeñas que inician un emprendimiento que crean un producto en condiciones de alta incertidumbre, es decir no se sabe si va a funcionar o no (*El Método Lean Startup*, 2012).

El enfoque en su producto, su posicionamiento en el mercado, la calidad y la innovación de estas empresas emergentes son piezas clave para su éxito. Las Startups suelen afrontar los siguientes retos al momento de realizar su emprendimiento:

- Baja calidad de código y falta de experiencia.
- No existe tiempo suficiente para realizar las diversas pruebas requeridas para garantizar la calidad del código.
- Ausencia de documentación y falta de una metodología.
- Presión sobre su negocio o aplicativo por parte de los clientes o inversores impacientes.

Uno de los mayores desafíos que presenta una Startup es el presupuesto limitado y la falta de personal para cada área específica, por lo que ven la necesidad de que una persona interactúe en diferentes etapas del ciclo de vida del software. Otro requerimiento importante es la necesidad de entregar y mostrar rápidamente sus productos, ya que, debido a la competencia, si no lo hacen, alguien más lo hará en su lugar. Es aquí donde se puede notar la necesidad de la implementación de DevOps, gracias a su enfoque en la optimización de procesos y la necesidad de que el personal tenga un conocimiento general de todo el ciclo de desarrollo, de esta forma, DevOps da el poder de sostenerse en el mercado y la empresa puede mantener una ventaja competitiva frente a otras empresas, aunque parezca difícil de implementar, los beneficios obtenidos a largo plazo justifican su implementación. (Chaturvedi, 2021)

Se puede creer que muchas de estas empresas emergentes aún tienen incertidumbre sobre si fuera necesario o no la implementación de DevOps, pero actualmente cada vez más personas están optando por su implementación, esto debido a las facilidades de aprendizaje y acceso a herramientas que existen hoy en día (Neeraj Mishra, 2022).

Para poder obtener los beneficios de la adopción de DevOps en startups se exponen los siguientes concejos (Hordashnyk Yevhenii, 2020) a considerar:

- **Capacitación**. - Se debe realizar una correcta implementación y configuración de las herramientas a utilizar, de lo contrario, se puede perder mucho tiempo obteniendo una deuda técnica. Una deuda técnica es el costo que implica la corrección y refactorización de acciones causadas por escoger una solución incorrecta.
- **Adoptar la tecnología de Nube.** - Actualmente existen varios proveedores de nube que ofrecen una amplia gama de servicios probados y listos para su puesta en marcha, de esta manera evitarse los costos y tiempos de implementación. De igual manera existen una infinidad de guías, tutoriales y buenas prácticas.
- **Automatizar la infraestructura**. - Utilizar la IAC (Infraestructura como Código) esto con el fin de evitar los errores humanos y evitar las tareas repetidas. Si bien esto al inicio puede costar mucho tiempo y dedicación, este proceso valdrá la pena a futuro.
- **Utilizar Contenedores**. – Los contenedores nos permitirán obtener portabilidad de nuestros proyectos, así como manejar diferentes entornos de desarrollo y pruebas similares o idénticos a los de producción haciendo el paso de versión mucho más rápido y confiable.

- **Monitoreo y Métricas.** – La implementación de herramientas de monitoreo son de gran utilidad con el fin de saber el estado de nuestra aplicación, detectando errores de seguridad, cuellos de botella entre otros, lo cual permite a los emprendedores tomar decisiones sobre acciones a tomar.

2.5.INTEGRACIÓN Y DESPLIEGUE CONTINUO (CI, CD)

Para mantenerse competitivo ante otras empresas, actualmente no solo es necesario contar con la capacidad de ofrecer aplicaciones de calidad y segura, sino que también es necesario que la entrega de este producto se lo pueda realizar de una forma rápida y correcta manteniendo al cliente constantemente al tanto de los avances realizados y que él pueda presenciar los avances directamente y dar una retroalimentación del producto. Para poder cubrir estas necesidades, existen practicas diseñadas para ayudar a las organizaciones a mejorar y acelerar la entrega de características y funcionalidades al producto software de forma automática, evitando la interacción humana y sin perder su calidad, las cuales son conocidas como Entrega Continua, por sus siglas CD (Continuos Delivery) e Integración Continua, por sus siglas CI (Continous Integration) o también denotado por CI/CD. (Ståhl & Bosch, 2014)

La Integración y Despliegue Continuo permite a las empresas reducir el tiempo de ciclo de vida del producto software esto con el fin de obtener rápidamente Feedback de los usuarios o clientes y bajar el riesgo comercial y el costo de las implementaciones, esto lo logra gracias a que las decisiones e implementaciones se basan en los Feedbacks del software en funcionamiento y no en hipótesis o supuestos.

En relación con riesgos, esta metodología es más madura, eficiente y efectiva para poder aplicar los controles de cambio y cumplir con los requisitos de negocio y reglamentos internos, de tal forma que se puede obtener mayor confianza en cambios de última hora realizados en producción (Patrick Debois et al., 2011).

En la siguiente figura, se puede observar a breve rasgo el proceso de un entorno CI/CD y sus diferentes componentes

Ilustración 1 Proceso CI/CD

Fuente:(Montoya & Ocampo, 2020)

La implementación de DevOps facilita la creación de CD en las empresas y de esta forma obtener sus diferentes beneficios (Axelsson, 2015). CD se logra gracias a la cultura de colaboración entre todos los miembros del equipo, medición de métricas de los procesos y costes, compartir conocimientos, uso de herramientas, automatización de los procesos y retrospectivas constantes, aspectos clave de la cultura DevOps. Uno de los primeros pasos para poder implementar el CD es la integración continua.

La integración continua es una práctica de desarrollo en la cual, cada desarrollador continuamente integra el código de las funcionalidades que trabajan en un reportorio o rama central, generalmente en un sistema de control de versiones. La integración de código se lo realizara periódicamente, preferiblemente a diario, esto con el fin de poder realizar pruebas constantemente y aplicar las correcciones necesarias inmediatamente.

Existen un gran número de herramientas de nos permiten la integración que son básicas para su implementación y en consecuencia permiten el correcto funcionamiento de la CD ya que facilitan la colaboración e interacción entre los desarrolles y su trabajo. Algunas de estas herramientas son Git, Jenkings, Azure Devops, GitLab y muchas otras herramientas que cada día van evolucionando(Mishra & Otaiwi, 2020b). El uso de estas herramientas depende del tipo de proyecto a lanzar, las políticas de la empresa y experiencia de su equipo. A estas herramientas también se suman aquellas que nos permitan la automatización de procesos y pruebas funcionales.

Si bien la CD se preocupa por liberar constantemente versiones a producción bajo demanda del cliente, de manera rápida y eficiente, el CI se preocupa por la integración constante del trabajo de los desarrolladores. El objetivo de CI/CD es hacer que el proceso de liberación de cambios al cliente sea sencillo y rápido, reducir el riesgo y optimizar el ciclo de vida del producto software.

2.5.1. Herramientas para la Implementación de CI/CD

Los equipos de desarrollo deben comprender mejor el valor de las herramientas de integración y entrega continua ya que el uso de herramientas es fundamental para su implementación, y el proceso de configuración depende mucho del sistema que estén elaborando ya que muchas veces necesitaran integrar diferentes herramientas para cumplir con el objetivo.

Actualmente existen una gran cantidad de herramientas para poder realizar la implementación de CI/CD con DevOps, y por esto es importante comprender primeramente las capacidades técnicas y del personal que se posee sobre el aplicativo a implementar ya que gracias a su conocimiento se podrá saber que herramienta es la que más le conviene al proyecto. Algunas herramientas y aplicaciones que nos permiten implementar las practicas DevOps son Azure DevOps, Azure Portal, Jenkins, SonarQube, Pingdom, Akami, Jira, entre otros(Taylor, n.d.).

En el siguiente cuadro comparativo, se pueden observar alguna de sus características y diferencias tomando en cuenta la capa gratuita de cada una de estas herramientas.

Tabla 2 Cuadro comparativo de Herramientas CI/CD DevOps

Nombre	**Funciones General**	**Capa Gratuita**
Aws CodePipeline Amazon Web Services	• Permite automatizar Pipelines de lanzamiento • Junto con otras herramientas de AWS como CodeCommit, CodeDeploy, CodeStart brindan soluciones para todo el ciclo de vida DevOps • Entorno en la nube listo para utilizar	Ofrece un Pipeline gratuito activo al mes.
Azure DevOps Microsoft	• Ejecución de una sola Pipeline al mismo tiempo, desarrollada en YAML o interfaz de usuario	Gratuitos para proyectos de máximo 5 personas.

	• Combina la CI/CD para compilar y probar automáticamente los cambios en el producto software • Ofrece herramientas integradas para todo el ciclo de vida DevOps • Entorno en la nube listo para usar	Limitación en la ejecución de Pruebas Automatizadas.
Jenkins Java	• Compatible con múltiples sistemas de control de versiones, es la herramienta de CI/CD más conocida • Permite obtenerlo localmente, si se cumple con los requisitos en las maquinas o servidores • Permite la ejecución de múltiples Pipelines al mismo tiempo	Totalmente gratuito al ser código abierto. Necesita de complementos para poder explotar toda su capacidad.
GitLab CI/CD Gitlab	• Incluido en GitLab, se encarga de la implementación continúa permitiendo actualizar cambios dentro del producto en producción. • Compilación automatizada • Permite la integración diaria del código, permitiendo identificar errores fácilmente • Plataforma en la nube, permite la ejecución de múltiples pipelines al mismo tiempo.	Gratis para uso individual con ciertas limitaciones, permite 400 minutos de CI/CD al mes.
Travis CI Travis	• Ofrece múltiples opciones de integración continúa automatizadas • Elimina la necesidad de un servidor dedicado al estar en la nube	Gratuita para repositorios Github públicos y BitBuket,

	• Permite realizar pruebas en diferentes entornos en varias máquinas y distintos sistemas operativos. • 5 pipelines al mismo tiempo en su versión Open Source	posee limitaciones en su capa gratuita
Circle CI	• Ejecuta procesos CI/CD automáticamente con una máquina virtual o contenedores limpios, esto permite facilitar la ejecución de pruebas. • Permite notificar al equipo en caso de errores. • Se integra con Bitbucket y Github para crear los Pipelines • Enfoque especial en la seguridad aislando las máquinas virtuales y con su gestión de usuarios.	Mas de 6 000 minutos para poder construir los aplicativos al mes. Permite la ejecución de 30 Jobs al mismo tiempo, además de permitir la ejecución de pruebas en paralelo.

Fuente: (Montoya & Ocampo, 2020)

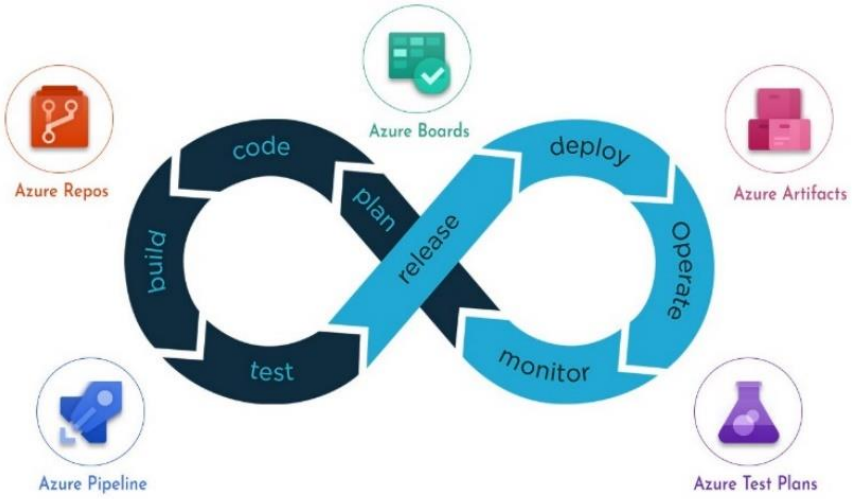
Para cumplir con todos los objetivos que busca DevOps, muchas veces será necesario combinar el uso de varias herramientas al mismo tiempo, de aquí la importancia en conocer también que tan compatibles son cada una de estas herramientas con otras.

El motivo de la selección de una u otra herramienta depende de varios factores, como la experiencia o manejo de los miembros del equipo, el presupuesto y acceso a documentación. Como se vio anteriormente, se recomienda el uso de tecnologías de nube o plataformas maduras y robustas, esto con el fin de evitar perder el tiempo en implementaciones desde cero sin la necesidad de adquirir equipos costosos y montar una nueva arquitectura.

2.6. AZURE DEVOPS

Azure DevOps es una plataforma de Software como Servicio (Saas) y es la evolución de lo que se conocía como Visual Studio Team Fundation Server (TFS) desarrollada por Microsoft que ofrece múltiples herramientas o servicios que abarcan toda la amplitud del ciclo de vida de DevOps. Todas estas herramientas permiten, por ejemplo, manejar el versionamiento de código fuente, crear diferentes entornos de desarrollo, administrar el ciclo de vida de software, implementación de CI/CD, creación de Pipelines, manejo eficiente del equipo de desarrollo, métricas del rendimiento y estado del aplicativo y avances del equipo de trabajo, también cuenta una amplia gama de extensiones en su Marketplace para poder, por ejemplo, conectarse con terceros. Azure DevOps nos permite acelerar el proceso de CI/CD, ya que flexibiliza la compilación y la construcción de los aplicativos a través de Azure Pipelines y sus otras herramientas. (Taylor, n.d.)

Ilustración 2 Azure DevOps en el ciclo de vida DevOps



Fuente:(Valencia, 2021).

Azure DevOps soporta cualquier lenguaje de programación y la integración de diferentes plataformas incluyendo nubes como AWS o Google Cloud. Gracias a su interfaz de usuario y funcionalidades preconfiguradas, permiten una implementación rápida e intuitiva a través de su interfaz Web, de tal forma que es fácil familiarizarse con su entorno pues se basa en el marco de trabajo ágil. Mucho de los conceptos relacionados al entorno de desarrollo de sistemas son manejados también en Azure DevOps.

En la siguiente tabla se observa brevemente los conceptos comúnmente conocidos con los servicios de Azure DevOps.

Tabla 3 Tabla de Servicios de Azure DevOps

Conceptos Generales	**Servicios de Azure DevOps**	**Descripción**
Construcción y Despliegue	Azure Pipelines	Entorno CI/CD configurable, trabaja con cualquier lenguaje de programación y plataformas incluyendo la nube.
Código	Azure Repos	Repositorio de código fuente basado en Git
Trabajos	Azure Boards	Seguimiento de trabajo a través de tableros Kanban, Backlogs, Dashboards y otras herramientas de seguimiento.
Test	Azure Test Plans	Pruebas automatizadas y reportes
Paquetes	Azure Artifacts	Repositorio de paquetes como Maven o Npm

2.6.1. Azure DevOps Services

Azure DevOps proporciona un conjunto de herramientas que nos permiten llevar el proceso de DevOps y la metodología Agile, entre las herramientas más destacadas que nos proporciona Azure DevOps (*Azure DevOps Services | Microsoft Azure*, n.d.) podemos ver:

2.6.1.1. Azure Repos

Es el almacenamiento de código a través de repositorios privados ilimitados de GIT hospedados en la nube. Permite la colaboración y la integración de código de las diferentes funciones implementadas por los desarrolladores, así como la capacidad de poder volver a estados anteriores de desarrollo en forma de historial y verificar el código que se está implementando.

2.6.1.2. Azure Boards:

Herramienta Agile que permite la visualización del estado de las actividades de los equipos en un tablero similar a Kanban, en el cual se pueden definir tareas a realizar, tareas realizadas y el estado de cada una de estas. Azure Boards permite planificar, debatir y hacer el seguimiento del trabajo a realizar.

2.6.1.3. Azure Test Plans

Herramienta para la definición, automatización y ejecución de pruebas manuales y exploratorias, proporciona herramientas completas y potentes para la ejecución de pruebas en las que cada miembro del equipo puede probar sus funcionalidades y mejorar la calidad de su desarrollo. Las pruebas que se pueden ejecutar son, por ejemplo, pruebas de carga, aceptación y funcionalidad, pudiendo hacerles seguimientos a través de tableros y estadísticas que nos proporciona Azure DevOps (Microsoft, 2020).

Tipos de Pruebas compatibles

Azure Test Plans admite los siguientes objetivos de prueba:

- **Pruebas manuales y Exploratorias**: Pruebas manuales desarrolladas por los miembros del equipo de trabajo, para esto se Integra una interfaz que nos permite registrar el resultado cada prueba.
- **Pruebas de aceptación de usuario:** En estas pruebas se asigna un usuario que verifica si el requisito cumple o no con los criterios de aceptación.
- **Pruebas Automatizadas:** Son pruebas implementadas dentro del código a forma de pruebas unitarias o de integración. Estas pruebas se ejecutan generalmente después de finalizado el desarrollo y generalmente dependiendo el resultado de estas, se procede con la siguiente etapa de desarrollo. Gracias a los Pipelines estas pruebas pueden ser ejecutadas automáticamente siendo parte del CI/CD y forman parte de los pasos al momento de desplegar una aplicación a producción. El resultado de las pruebas puede ser observados a través de los informes de progreso y ejecución de los Pipelines.
- **Trazabilidad:** Van de la mano con los criterios de aceptación de las historias de usuario ya que son pruebas que se definen en su aceptación. Permiten realizar el seguimiento de la calidad de los requisitos. Estas pruebas pueden ser agregadas directamente desde el tablero Kanban o directamente desde el apartado de Azure Test Plans.

2.6.1.4. Azure Artifacts

Permite crear, hospedar y compartir paquetes de origen público y privado con todo el equipo de desarrollo, estos paquetes pueden ser de tipo Maven, Npm, Nuget, entre otros. Una vez finalizado el Pipeline de construcción, se generan archivos que son guardados en este directorio y se los conoce como Artefacto

2.6.1.5. Azure Pipelines

Sistema de integración y entrega continua que funciona con cualquier lenguaje, plataforma o nube, conectándose a cualquier tipo de repositorio GIT, permitiendo realizar los procesos CI/CD. Permite automatizar las diferentes fases del desarrollo, compilación, pruebas y su despliegue en los diferentes ambientes de desarrollo.

La compilación es el proceso en el que el código fuente, sus dependencias y ficheros son convertidos en un software funcional. Utilizando los Pipelines podemos compilar y probar el código automáticamente, al finalizar el proceso de un Pipeline, se generan Artefactos y se los prepara para que, de esta forma, implementando CD, se los consuma y se los despliegue de forma automáticamente, con eso se pueden obtener nuevas versiones de software o nuevas funcionalidades de forma mucho más rápida.

Azure Pipelines es quizá una de las herramientas más conocidas de Azure DevOps, de tal forma que muchos desarrolladores la combinan con sus otros ambientes DevOps como Jenkins o Jira. Algunas de las razones para utilizar Azure Pipeline (Rossberg, 2019) pueden ser:

- Funciona en cualquier lenguaje de programación.
- Puede Desplegar el aplicativo en los diferentes ambientes de desarrollo, como por ejemplo QA y Producción al mismo tiempo.
- Se puede desarrollar e implementar en sistemas operativos de Windows, Linux o Mac.
- Proporciona una gran integración con repositorios de código como Github
- Es una gran opción para proyectos Open Source.

Pipeline se desarrolla en archivos Yaml, y se encuentran generalmente compuestos de la siguiente estructura:

- **Trigger**: Desencadenan la ejecución de la Pipeline, ubicadas al principio de todo. El detonante del Trigger puede ser por ejemplo un Commit o Merge a un Branch especifico.

- **Stages**: Indica las diferentes etapas de ejecución del Pipeline.
- **Jobs**: Compuesto de una serie de pasos que necesitamos que se vayan ejecutando paso a paso.

2.6.1.6. Release Pipeline

Permiten al equipo de desarrollo la implementación de entrega continua (CD) del software a los clientes de forma mucho más rápida y reduciendo el riesgo de errores, optimizando los tiempos al automatizar procesos repetitivos. Permite la automatización completa de las pruebas y despliegue del software a su etapa en producción. También permite configurar procesos semi automáticos como aprobaciones o revisiones de código por parte de otros miembros del equipo y la implementación de despliegues bajo demanda o programadas.(Microsoft, 20222)

Para poder utilizar el Release Pipeline se necesita primero generar un Artefactos, estos pueden ser generados desde los Pipeline de construcción o a través de una amplia variedad de origen de artefactos como pueden ser de Jenkins o Team City.

Una vez definido el origen de los artefactos se procede al despliegue mediante etapas, estas etapas pueden ser definidas a través de los Trabajos (Jobs), por lo que, muchos de los conceptos hablados anteriormente en Pipelines son aplicables. Los Jobs deben de ser configurados de tal manera que se configure el despliegue del aplicativo en el servidor de aplicaciones que necesitemos. El uso de variables nos permite manejar diferentes entornos de desarrollo y los Trigers permiten saber cuándo debe ejecutarse el despliegue.

2.6.2. Azure DevOps vs Otras alternativas

Para la implementación de DevOps existe una gran cantidad de herramientas que nos facilitan su adopción de una forma fácil, rápida e intuitiva. Una de las primeras acciones que se debe tener en cuenta al momento de adoptar DevOps es las herramientas necesarias, pero para esto hay que considerar muchos aspectos y alternativas. Jenkins actualmente es una de las herramientas preferidas por los desarrolladores debido principalmente a que es de código abierto, sin embargo, últimamente Azure DevOps está tomando mayor notoriedad en este ámbito gracias a su interfaz sencilla y fácil de implementar y utilizar, también es gratuito para equipos pequeños y fácilmente escalable para cuando vayamos evolucionando.

El uso de una u otra herramienta entonces depende mucho de las necesidades y preferencias de cada equipo y a continuación, se muestra un cuadro comparativo con diversas ventajas y desventajas de Azure DevOps con Jenkins que podrían guiarnos al momento de elegir que herramienta utilizar (Sarma, n.d.):

Tabla 4 Azure DevOps vs Jenkins

Azure DevOps	**Jenkins**
Permite la configuración del entorno en la nube e instalado en un servidor o computador local.	Es necesario descargarse el aplicativo e instalarlo en un servidor o computador local.
Permite simplificar en una sola tarea una serie de procesos y trabajos ya definidas gracias a los Pipeline	Generalmente las tareas se las realizan manualmente, una por una lo que puede generar problemas de seguimiento o responsabilidad.
Gracias a los ficheros YALM es posible codificar y configurar los Pipelines para tareas de CI/CD	No soporta una interfaz YALM y sus Pipelines son programadas con Groovy, un lenguaje propio de Jenkins
Es posible desplegar varias aplicaciones de diferentes lenguajes de programación, y luego subirlas a diferentes instancias de la nube de forma automática.	Permite la integración con diferentes lenguajes de programación, gestores de paquetes o servidores de aplicaciones gracias a su comunidad que constantemente le permiten la compatibilidad con las nuevas tecnologías.
Proporciona métricas de análisis como la tasa y duración de la ejecución de forma gráfica y estadística.	No posee ningún tipo de métricas de forma nativa.

Posee una tienda llamada Azure DevOps Marketplace en donde podemos encontrar Extensiones y Plugins.	Posee una gran amplia variedad de Plugins proporcionados por la comunidad.
La integración de Azure Pipeline con productos Microsoft puede ser muy fácil, pero para poder integrarse con terceros, requiere una configuración adicional.	Puede integrarse fácilmente con otras plataformas, pudiendo ser fácilmente modificado y expandido.
Soporte por parte de Microsoft, al ser un producto relativamente nuevo actualmente no existe tanta ayuda por parte de la comunidad.	Debido a que Jenkins es un proyecto Open Sourse, existe una gran cantidad de soporte por diferentes equipos y la comunidad.
Posee muchas herramientas de forma nativa, es decir vienen incluidas en su ecosistema y son fácilmente manejables.	Depende mucho de los Plugins para poder obtener nuevas funcionalidades, muchas veces su instalación y administración suelen ser complejos.

En el mercado actualmente existen una gran cantidad de herramientas que presentan soluciones similares a las de Azure DevOps, cada una expone herramientas similares y funcionalidades exclusivas, entonces queda a criterio de cada equipo utilizar, otras de las más conocidas son las siguientes:

- Gitlab
- Copado CI/CD
- TeamCity
- AWS CodePipeline
- CloudBee CI
- Bamboo
- Red Hat Ansible Automation Platform
- Google Cloud Build

3. Objetivos y metodología de trabajo

En este apartado definiremos los objetivos del proyecto y la metodología utilizada para poder cumplir con estos objetivos.

3.1. Objetivo general

Creación de un entorno de integración y despliegue continúo basado en DevOps de un aplicativo desarrollado por una Startup utilizando el entorno Azure DevOps de Microsoft.

3.2. Objetivos específicos

I. Definir e Implementar el entorno CI/CD basado en DevOps y las herramientas del entorno Azure DevOps a ser utilizada en el proyecto.
II. Implementar la Integración Continúa utilizando para esto el gestor de código fuente y crear los diferentes entornos de desarrollo.
III. Establecer y configurar los Pipelines de construcción y despliegue del aplicativo.
IV. Implementar pruebas unitarias y de integración al aplicativo y automatizar su ejecución.
V. Evaluar y verificar el estado del aplicativo y la implementación del entorno CI/CD basándonos en las métricas obtenidas en el entorno Azure Devops.

3.3. Metodología del trabajo

Para poder realizar el trabajo propuesto, fue necesario primero conocer como fue desarrollado el aplicativo en primera instancia, las herramientas, tecnologías utilizadas, así como su estado actual.

Tipo de Investigación

Una vez conocido el estado del aplicativo, se procedió a realizar la investigación de las herramientas que se utilizan para la creación de entornos de integración y despliegue continuo (CI/CD) basados en prácticas DevOps y para esto, fue necesario primeramente realizar una investigación general del tema DevOps y sus herramientas, en la cual vimos las diferentes ventajas, desventajas y marcos de referencia para su adopción en proyectos similares a los expuestos en este trabajo, debido a esto, se puede decir que la investigación se ha calificado como descriptiva, aplicada y mixta.

- **Descriptiva. –** Se califico como descriptiva pues se tuvo que describir las diferentes ventajas y características de la herramienta de implementación de un entorno CI/CD basado en DevOps seleccionada y la diferencia con las otras herramientas existentes esto con el fin de brindar una visión general de las herramientas que existen actualmente y las ventajas que tiene el haber escogido la herramienta seleccionada y en base a esto proponer la implementación de dicha herramienta en un aplicativo simple.
- **Aplicada. –** Es aplicada ya que se realizó la implementación del entorno de integración y despliegue continúo utilizando la tecnología expuesta en un aplicativo previamente desarrollado de manera tradicional.
- **Mixta. –** Es Mixta debido a que, a través del análisis e investigación realizada, se pudo conseguir información que permitió tener un conocimiento breve de la situación de la implementación de entornos CI/CD en Startups y con esto aplicarla a la aplicación expuesta, la cual trata de un aplicativo de demostración.

Marco de trabajo

Una vez obtenida toda la información y fundamentos teóricos necesarios, se procedió con la implementación del entorno de integración y despliegue continuo, así como la creación y automatización de pruebas, utilizando para esto las herramientas que nos brinda el entorno Azure DevOps, haciendo uso también de la metodología ágil.

Sprint Cero

El primer sprint será utilizado para definir el esquema de trabajo a seguir, la definición de las historias de usuario y la arquitectura general del aplicativo y nos servirá para poder empezar a familiarizarse con la plataforma.

Al finalizar esta reunión se obtendrá el backlog y las historias de usuario, todo esto ya implementado en el entorno Azure DevOps, utilizando las épicas, historias de usuario, ramas el tablero Kanban y demás herramientas.

Definición de Sprint

El sprint fue definido con una duración de dos semanas, en el cual se incluyeron todas las actividades que corresponden con la migración del aplicativo a la plataforma en la nube de Azure. Para seleccionar el equipo y el manejo de sesiones según la metodología, se ha

considerado el tiempo disponible de cada desarrollador, su experiencia de desarrollo y las condiciones iniciales del proyecto. El equipo cuenta de 4 personas que están constantemente activas para el desarrollo de estas se definieron los roles de cada uno y se procedió con la planificación.

El procedimiento de la implementación de los cambios necesarios se ha definido inicialmente en las siguientes iteraciones:

Tabla 5 Definición de Épicas

Iteración	Nombre	Features	Historias
1	**EP01** Migración del aplicativo	**F01** Migración General del api	• **H01** Migración del aplicativo al Repositorio • **H02** Creación de Branchs y entorno de desarrollo • **H03** Creación de Pipelines de construcción
2	**EP02** Implementación de Pruebas y Actualización del Api	**F02** Implementación de pruebas **F03** Actualización y mejoras del Api	• **H04** Creación de Pruebas de Integración • **H05** Creación de Pruebas Unitarias • **H06** Documentación de Api con Swagger • **H07** Implementación de la automatización de las pruebas • **H08** Creación de Pipelines de despliegue para el despliegue del aplicativo.

Arquitectura y estado del aplicativo

El presente proyecto se lo realizó sobre un aplicativo desarrollado en la tecnología NetCore y está basado en servicios Rest a manera de microservicios, dichos servicios proporcionan a un aplicativo móvil desarrollado en ReactJS, todo lo relacionado con el BackEnd, es decir la lógica de negocio y persistencia a datos.

El aplicativo fue desarrollado por una empresa emergente y fue brindado para poder implementar este proyecto sobre él, esto con el fin de evaluar el resultado final y dependiendo los resultados, poder implementarlo oficialmente o en cualquiera de los otros proyectos que esta empresa maneja.

El aplicativo actualmente se lo tiene en producción como beta para su demostración a diferentes clientes interesados, los cuales, en base a sus necesidades, lo adaptaran con nuevas funcionalidades y requerimientos.

De acuerdo con las necesidades del proyecto, el equipo de trabajo y el entorno DevOps a implementar, se ha definido que el aplicativo contara con:

- **Ambiente de Desarrollo. -** En este ambiente se realizarán las implementaciones de los diferentes requisitos y nuevas funcionalidades**.** Cada desarrollador contara con este entorno en sus respectivas computadoras, de tal forma que puedan evaluar el comportamiento inicial del código y arreglar cualquier error que se presente en ese instante.
- **Ambiente de Pruebas y Control de Calidad (QA).** - En este apartado será para las pruebas en un entorno muy parecido al de producción, en el cual se aprobarán las diferentes funcionalidades expuestas por Desarrollo, y serán aprobados por parte del líder del proyecto.
- **Ambiente de Producción**. – Entorno en el cual será expuesto el aplicativo para su uso en entornos reales.

4. DESARROLLO ESPECÍFICO DE LA CONTRIBUCIÓN

En este episodio se describirá el procedimiento realizado para la elaboración del proyecto, para lo cual se hará un breve análisis del aplicativo, su migración, la implementación del entorno CI/CD, la elaboración de pruebas automatizadas y los resultados obtenidos de este procedimiento.

4.1.Análisis del aplicativo a migrar y requisitos

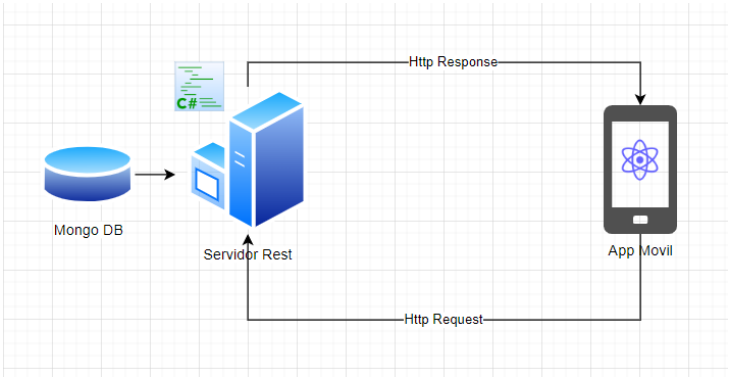
El Aplicativo tiene el nombre de "SafeLife", y permite a los usuarios enviar alertas de situaciones o eventos inusuales en un barrio, conjunto residencial o vecindario. Las situaciones inusuales pueden ser, personas desconocidas merodeando el sector, avistamiento de robos, situaciones anormales, entre otras. El principal objetivo de la aplicación es poder enviar alertas en forma de texto, imágenes, ubicación a todos los miembros del barrio con el fin de tomar alguna acción al respecto.

Se compone actualmente de un apartado de API Rest y el Apartado Móvil. En este trabajo, se utilizará únicamente el API para la implementación del entorno CI/CD.

Arquitectura: Se compone de 2 apartados claramente diferenciados

- **Móvil**. - Desarrollado en ReactJs es la parte visual del aplicativo, en el cual se maneja toda la interacción con el usuario.
- **Backend**: Esta desarrollado en NetCore basado en servicios Rest, estos servicios proporcionan a un aplicativo móvil todo lo relacionado con el Backend (lógica de negocio, persistencia a datos).

Ilustración 3 Arquitectura del Aplicativo



4.1.1. Metodología de desarrollo

"SafeLife" fue desarrollado de manera tradicional por un equipo corto de 4 desarrolladores pertenecientes por una empresa que se ubica dentro de la categoría de StartUp, la cual actualmente se la tiene en producción como beta, para su demostración a diferentes clientes interesados. El proyecto tardo alrededor de 6 meses, entre la creación de la idea y su despliegue en producción.

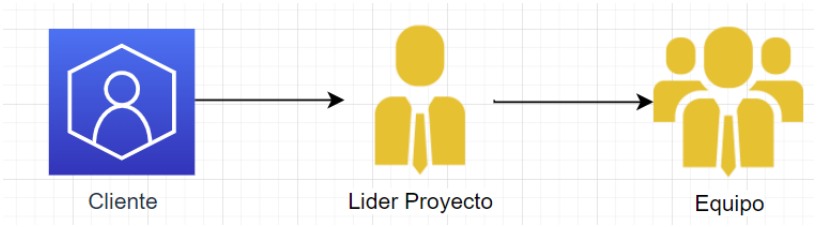
4.1.2. Interacción del equipo de desarrollo y el cliente

Las peticiones y requerimientos venían de una reunión por parte del cliente y líder de proyecto, el cual después de un breve análisis de los requisitos, distribuía las tareas a cada miembro del equipo.

Al final de cada semana, se tenían q enviar los respectivos cambios al líder del proyecto en forma empaquetada para ser validado y dar su respectiva retro alimentación o FeedBack.

Una vez aprobado el cambio, el líder subía manualmente los cambios al entorno de producción que disponía y se procedían a realizar las pruebas correspondientes con usuarios de prueba que disponía en tal ambiente.

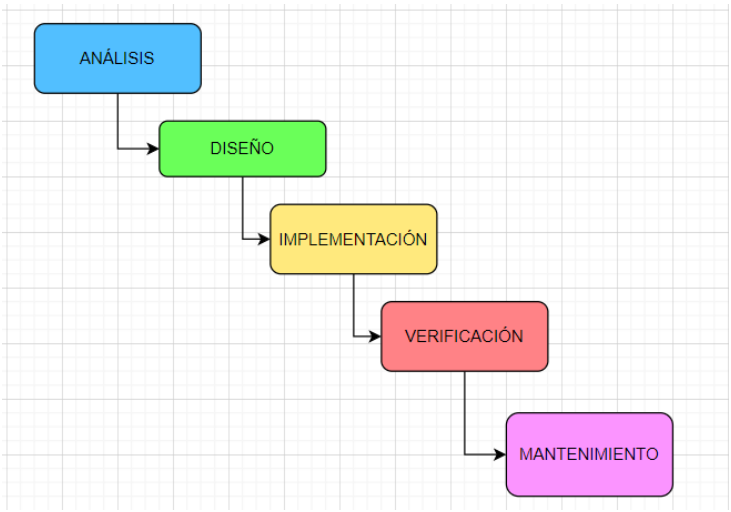
Ilustración 4 Flujo antiguo de desarrollo del aplicativo



4.1.3. Ciclo de vida de desarrollo

Como se puede observar en la ilustración 5, la metodología utilizada era muy similar a la de Cascada, partiendo desde el análisis de los requisitos, continuando con el diseño, desarrollo, pruebas y finalmente su implementación. Actualmente se encuentra en mantenimiento constante debido a que el aplicativo debe estar sujeto a cambios que el cada cliente requiera implementar.

Ilustración 5 Metodología de desarrollo

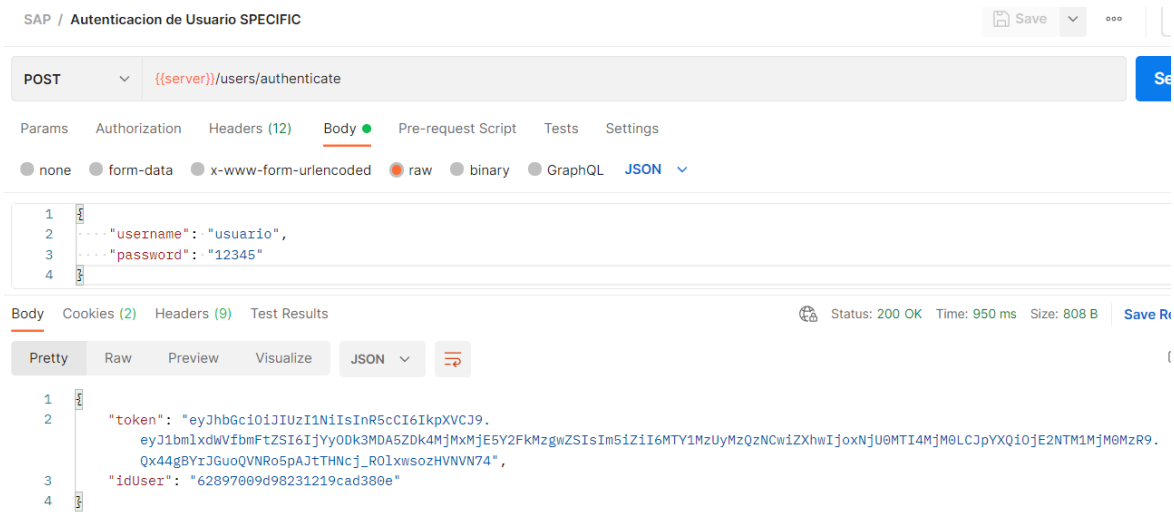


4.1.4. Estado actual del Aplicativo

El aplicativo cuenta con un conjunto de módulos que permiten la comunicación entre los diferentes componentes, estos módulos son:

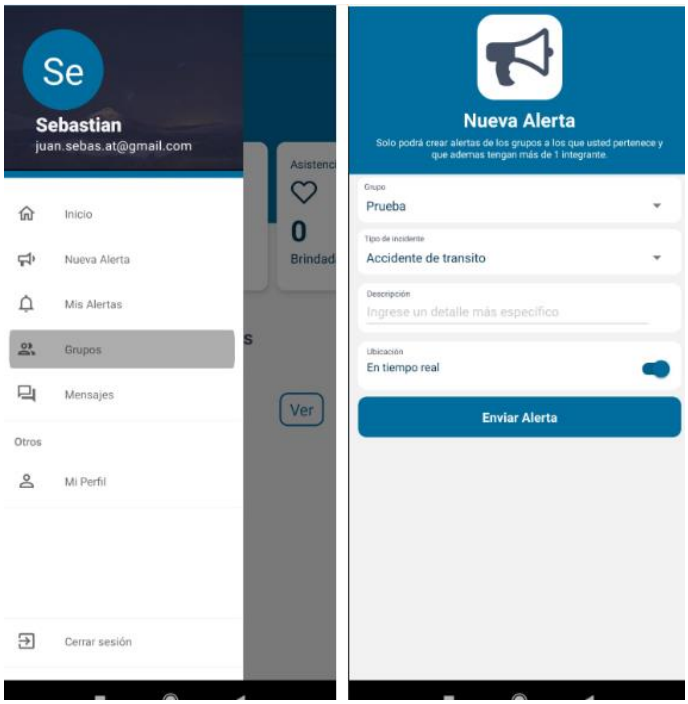
Api: Conjunto de servicios Rest similares a microservicios que exponen los servicios por el protocolo HTTP. Para verificar el funcionamiento de los servicios, se utiliza Postman que es una herramienta que nos permite realizar pruebas de APIs.

Ilustración 6 Consumo del Api Rest en Postman



Aplicativo Móvil: Aplicativo desarrollado en ReactJs el cual consume los servicios Rest del Backend y los muestra.

Ilustración 7 Interfaz del aplicativo móvil



4.2. Migración del aplicativo al entorno de Azure DevOps

El primer paso para migrar el aplicativo hacia este entorno DevOps es la creación de un usuario, esto se logra simplemente con un correo de Microsoft, cada miembro del equipo debe crearse una cuenta y ser invitado al proyecto a través de correo electrónico o enlace de invitación.

4.2.1. Inicio del Proyecto

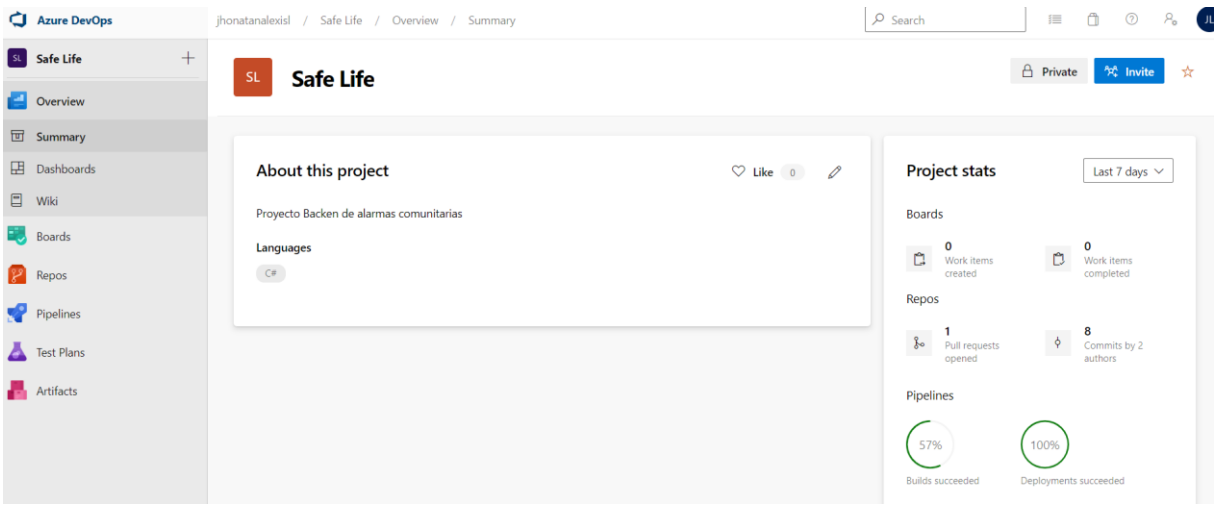
Para este caso, se utilizó el plan básico gratuito, el cual proporciona hasta 5 usuarios gratis y 6$ por cada usuario adicional. Este plan contiene:

- **Azure Pipelines**: Incluye la oferta gratis de SERVICIOS INDIVIDUALES
- **Azure Boards**: Para el seguimiento de elementos de trabajo y paneles Kanban
- **Azure Repos**: Nos brinda repositorios GIT privados e ilimitados
- **Azure Artifacts**: Nos brinda 2 GiB gratis para artefactos
- **Azure Test**: El plan básico de pruebas automáticas y manuales

Las pruebas también se las puede realizar directamente desde la configuración del aplicativo, Pipeline o Release y adjuntando cada caso de prueba en las historias de usuario como criterios de aceptación.

Una vez seleccionado el plan y hacer la configuración inicial del proyecto, tendremos la siguiente pantalla de inicio:

Ilustración 8 Página principal de Azure DevOps

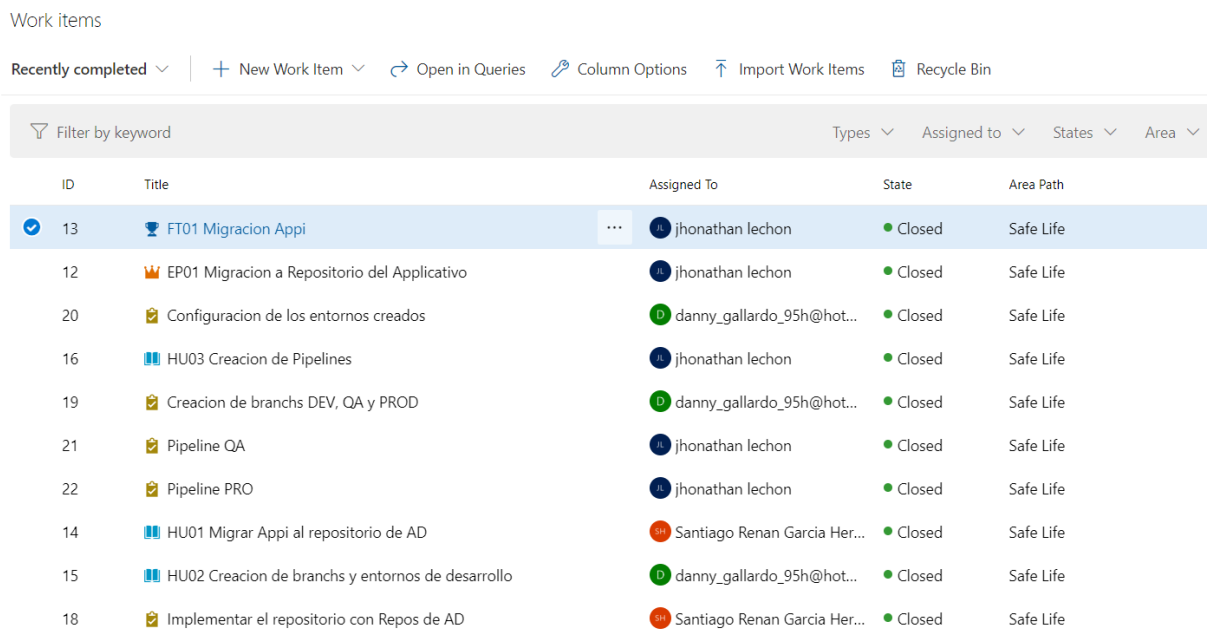


Como muestra la ilustración 8, en la página de inicio, en el lado derecho se puede observar las diferentes herramientas que nos brinda el portal Azure DevOps. En la parte central muestra una información de resumen sobre nuestro proyecto. En la parte derecha se ve un resumen general de las actividades y estado del aplicativo de manera gráfica y también se puede configurar para indicar métricas personalizadas.

4.2.2. Configuración de Boards

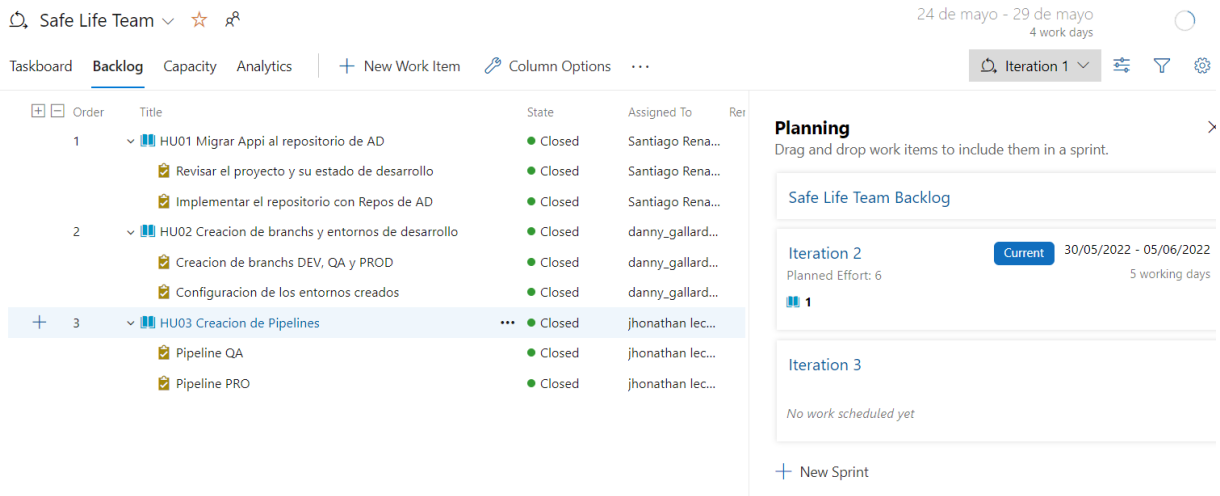
Como una breve introducción a Boards y nuevo marco de trabajo se optó por empezar a subir las actividades de la migración a cada desarrollador, incluyendo las Epicas, Features, Historias de usuario, tablero Kanban, entre otras de las muchas funcionalidades que nos brinda Azure DevOps, esto con el fin de empezar a familiarizarse con el entorno. El manejo del entorno fue relativamente fácil gracias a la interfaz gráfica intuitiva de Azure DevOps, la documentación y videos tutoriales.

Ilustración 9 Items de trabajo sobre la migración del Aplicativo



En el apartado “Sprints” podemos ver resumidamente las historias de usuario, el estado, la persona asignada, estas columnas pueden ser fácilmente personalizadas y agregar categorías que cada Scrum Máster vea necesario. En la ilustración 10 podemos ver en el lado derecho el Backlog y las iteraciones actuales y finalizadas.

Ilustración 10 Visualización del Sprint 0

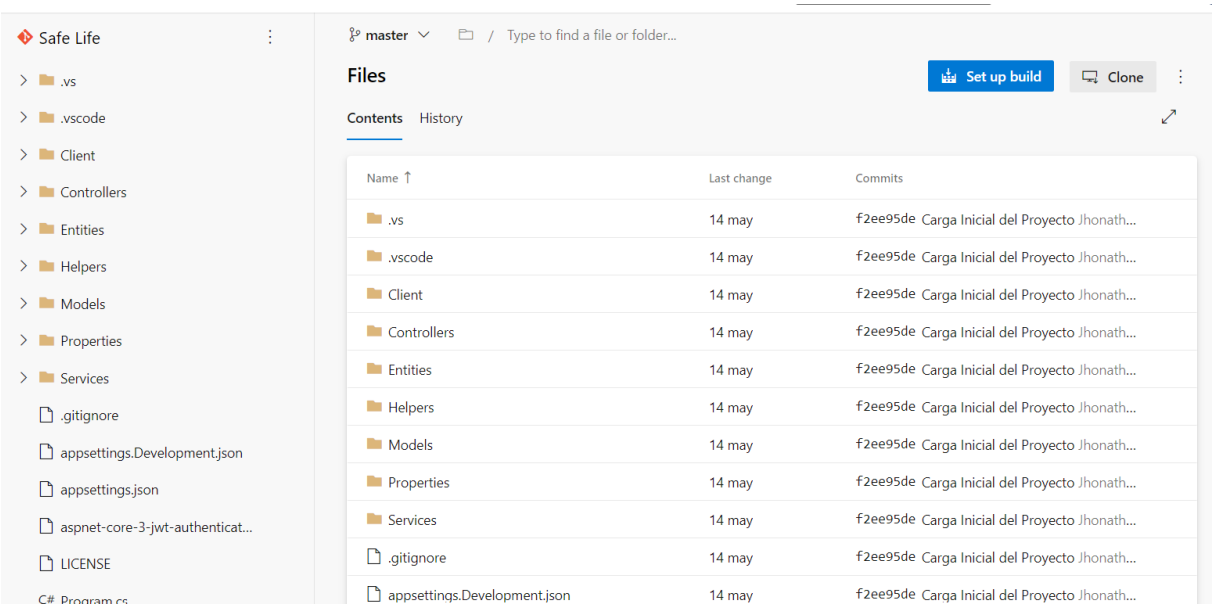


4.3. Implementación de la Integración Continua

Continuando con los pasos del proyecto, se procedió a subir el código fuente del aplicativo Api al apartado de “Repos” del portal Azure DevOps a la rama principal que viene por defecto

utilizando para esto Git. En la ilustración 11 se observa el código en nuestro repositorio en Azure DevOps.

Ilustración 11 Repositorio de código fuente del aplicativo



4.3.1. Ambientes de Desarrollo

El tener nuestro código en un repositorio, nos permitió también manejar los diferentes ambientes de desarrollo, de los cuales se definieron entre todos los miembros del equipo los 3 tipos más comunes:

- **Ambiente de Desarrollo (Dev). -** En este ambiente se realizan las nuevas implementaciones de los diferentes requisitos y funcionalidades. Cada desarrollador tendrá este ambiente en su respectiva maquina y mantendrá las configuraciones localmente**.**

  DataBase: Local

  IIS: Local

- **Ambiente de Pruebas y Control de Calidad (QA).** – Este ambiente se utiliza para las pruebas en un entorno muy parecido al de producción, su despliegue será a través de un Web App en Azure y la base de datos de Mongo en la Nube.

  DataBase: BD Atlas QA

  IIS: Azure WebApp QA

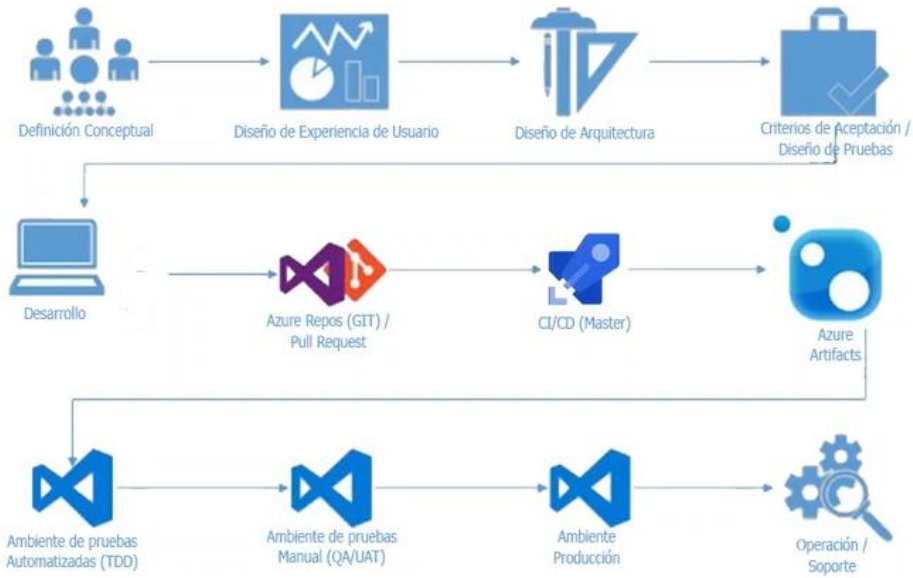
- **Ambiente de Producción (PRO)**. – Este entorno será el que esta desplegado para su utilización en producción, su despliegue será a través de un Web App en Azure y la base de datos de Mongo en la Nube. El paso a esta rama deberá ser aprobado por el líder del proyecto.

  DataBase: BD Atlas PRO

  IIS: Azure WebApp PRO

La arquitectura general de los entornos de desarrollo de nuestro aplicativo se expresa en la siguiente ilustración.

Ilustración 12 Arquitectura de despliegue



Fuente:(Microsoft, 2021)

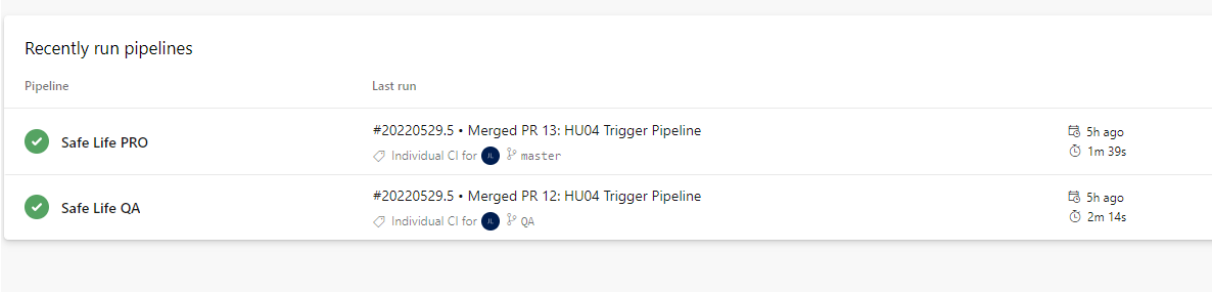
4.4. Implementación de Pipelines

Azure DevOps nos permite la creación de Pipelines de una forma fácil e intuitiva a través de su interfaz gráfica, para esto utilizamos el lenguaje Yaml, en el cual, describiremos paso a paso los pasos para la ejecución y empaquetamiento de nuestro aplicativo, así como la instalación de librerías y paquetes necesarios para esto.

Azure Pipelines posee también varias plantillas de diferentes entornos o configuraciones comunes que podríamos necesitar, además de Templates que se pueden conseguir de la comunidad o en su tienda virtual.

Para este proyecto fue necesario incluir las configuraciones en dos ficheros de formato Yaml, uno para PRO y otro para QA, cada uno apuntando a su ambiente respectivo y con sus respectivas configuraciones, los Pipelines creados podemos observarlos en la siguiente ilustración.

Ilustración 13 Pipelines de los diferentes entornos



Nuestro aplicativo, al ser desarrollado en NetCore, normalmente necesitaba de comandos que compilaba y empaquetaba los archivos necesarios para poder implementarlo en nuestro IIS. Ahora, para el pipeline, describiremos los pasos a seguir para que se pueda obtener el empaquetado, desde la instalación de las dependencias, configuraciones de entorno y empaquetamiento. Los pasos descritos son los siguientes:

Trigger: Indica cuando se va a ejecutar el Pipeline, en este caso se ejecutará cuando se haga un Push al Branch de QA y Producción

Ilustración 14 Trigger del Pipeline

```
1
2  trigger:
3  - QA
```

Pool: Es la máquina virtual que nos proporciona Azure DevOps en el cual se ejecutan las compilaciones, para nuestro caso al tratarse de un proyecto NetCore utilizaremos la máquina de Windows.

Ilustración 15 Pool del Pipeline

```
5  pool:
6    vmImage: 'windows-2019'
```

Variables: Se utilizó estos valores para poder facilitar el manejo de ambientes y tipo de construcción. También no permitió poder escribir y modificar directamente en nuestro fichero de configuración la Base de Datos en tiempo de despliegue, permitiéndonos manejar diferentes ambientes de desarrollo con un solo archivo de configuración.

Ilustración 16 Variables del Pipeline

```
8   variables:
9     appsettingsfile: appsettings.json
10    MongoDatabaseSettings.ConnectionString: 'mongodb+srv://    c:    c@cluster0.d2izz.mongodb.net/?retryWrites=true&w=
11    buildConfiguration: 'Release'
```

Steps: Los pasos o Jobs a ejecutarse, para este caso lo hace en la siguiente secuencia

1. Instalación de Net Core 5, esto con el fin de obtener todas las librerías del SDK.

Ilustración 17 Tarea de instalación

```
Settings
14  - task: UseDotNet@2
15    displayName: 'Install .Net 5 sdk'
16    inputs:
17      version: '5.0.x'
18      performMultiLevelLookup: true
```

2. Escritura de archivo de configuración, en este apartado modificaremos la cadena de conexión de la Base de Datos, para poder apuntar a los diferentes entornos de desarrollo, pues contamos con una base de datos para cada entorno.

Ilustración 18 Tarea de modificaciÓn de archivo

```
Settings
20  - task: FileTransform@2
21    displayName: "Transform Json"
22    inputs:
23      folderPath: '$(System.DefaultWorkingDirectory)/**/'
24      xmlTransformationRules: ''
25      jsonTargetFiles: '**/$(appsettingsfile)'
26  - bash: |
27      cat $(appsettingsfile)
28    displayName: "Show Json substitution"
```

3. Construcción del proyecto con las configuraciones definidas.

Ilustración 19 Tarea de construcción del proyecto

```
Settings
30  - task: DotNetCoreCLI@2
31    displayName: 'Build projects'
32    inputs:
33      command: 'build'
34      projects: '**/WebApi.csproj'
35      arguments: '--configuration $(buildConfiguration)'
```

4. Construcción de la publicación del paquete, se especifica que el paquete será un fichero Zip.

Ilustración 20 Tarea de empaquetamiento del proyecto

```
Settings
- task: DotNetCoreCLI@2
  displayName: 'Create Publis Package'
  inputs:
    command: publish
    publishWebProjects: true
    projects: '**/WebApi.csproj'
    arguments: '--configuration $(buildConfiguration) --output $(Build.ArtifactStagingDirectory)'
    zipAfterPublish: true
```

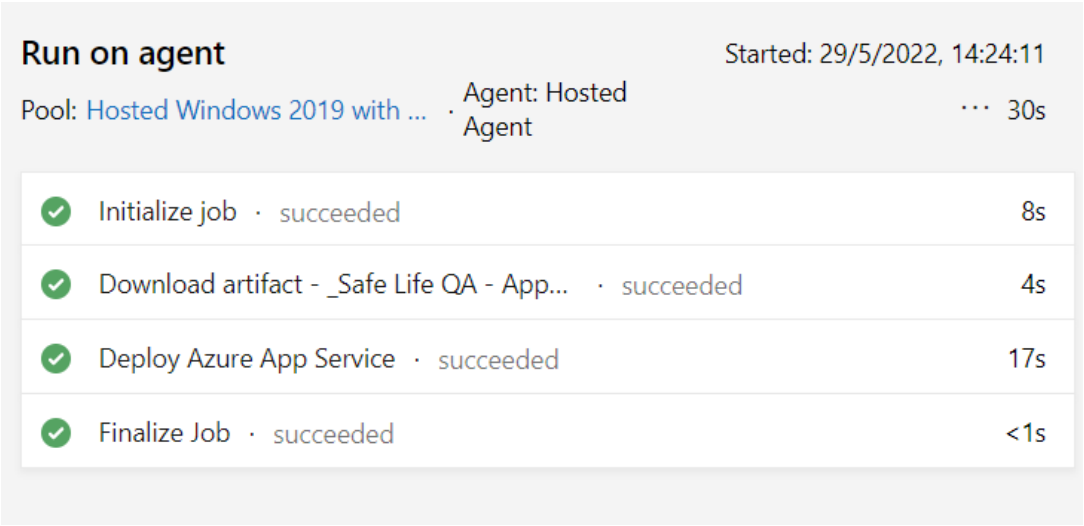
5. Publicación del Artefacto construido, este nos servirá para poder desplegarlo

Ilustración 21 Tarea de publicación

```

Settings
- task: PublishBuildArtifacts@1
  displayName: 'Publis Package'
  inputs:
    PathtoPublish: '$(Build.ArtifactStagingDirectory)'
    ArtifactName: 'AppApi'
```

Al ejecutar el Pipeline, podemos observar gráficamente el estado de cada paso ejecutado, su duración, fecha y Logs. En la ilustración 22, se observa el resultado de la ejecución del Pipeline descrito anteriormente.

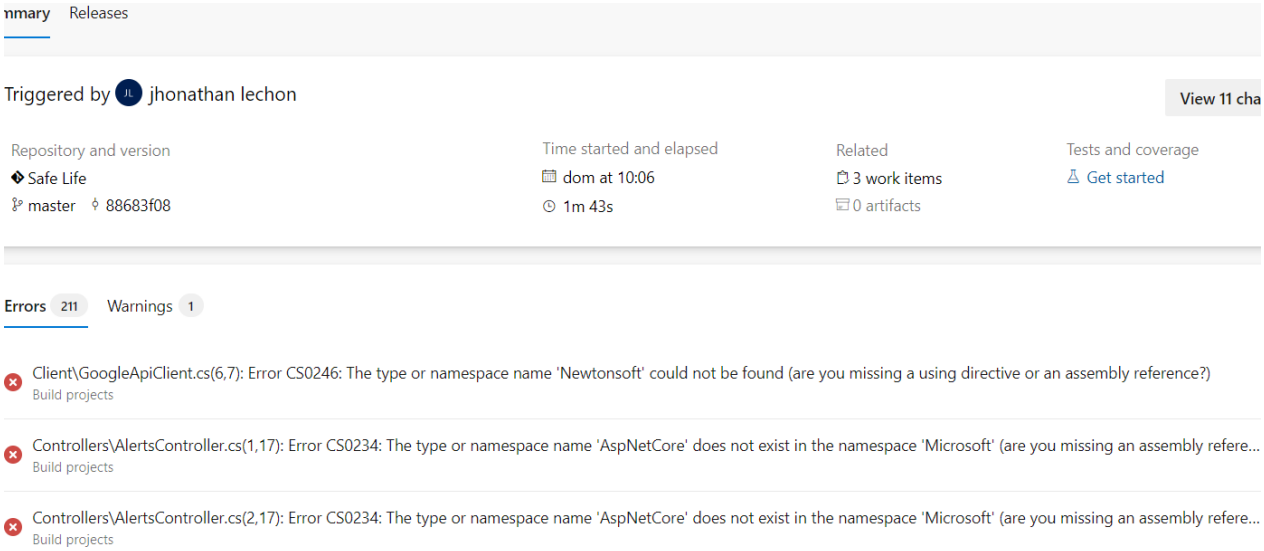
Ilustración 22 Resultado de la ejecución del Pipeline



También, nos permite ver los errores generados durante la ejecución del Pipeline. En la siguiente ilustración, se puede observar los errores presentados al momento de ejecutar un Pipeline. En este ejemplo, el error mostrado se debe a que no especificamos el proyecto a

ejecutar. Como podemos ver, esto nos permite, en la mayoría de los casos, saber directamente el motivo de tal error y así poder corregirlos inmediatamente.

Ilustración 23 Log de errores de la ejecución del Pipeline

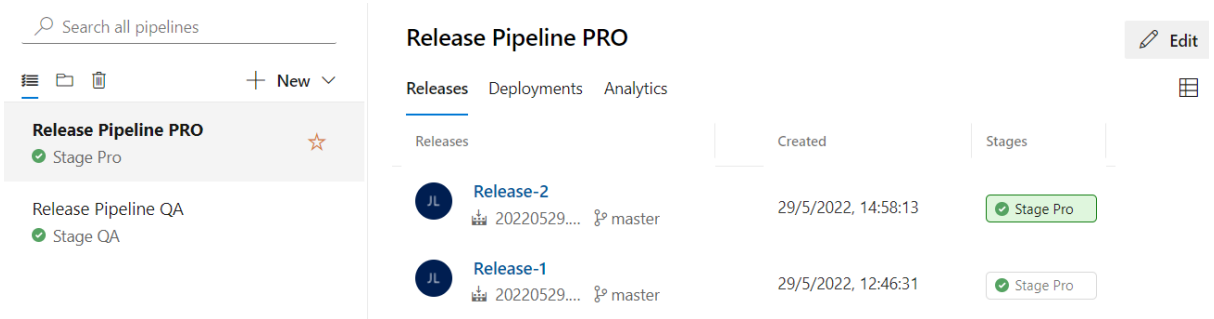


4.5. Implementación de Despliegue Continuo

Azure DevOps tiene una herramienta en Pipelines denominado Releases, en esta herramienta podremos publicar nuestro aplicativo generado por el Pipeline a nuestro servidor de aplicaciones o nube de forma automática, esto se logra gracias al concepto de Triggers visto anteriormente y a configuraciones internas.

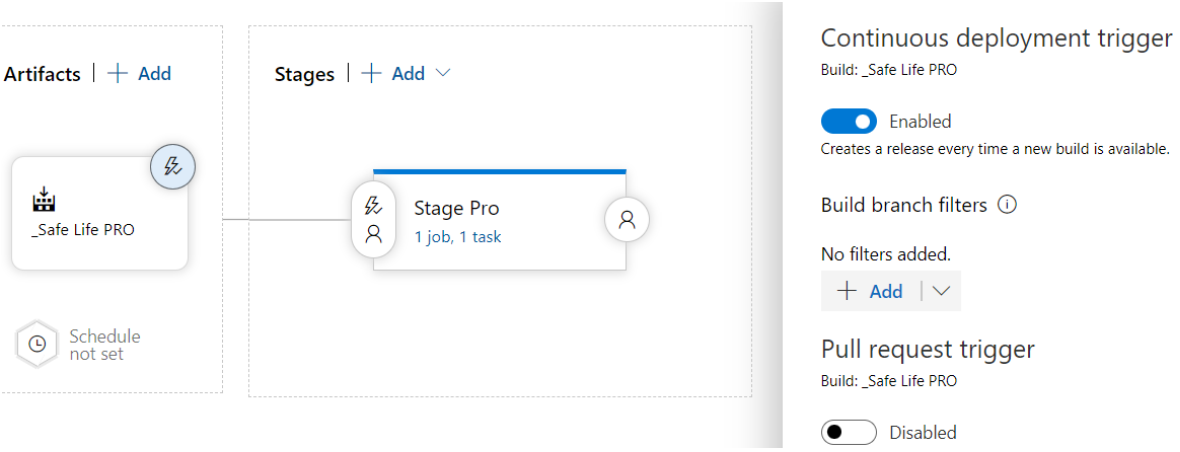
En este proyecto se crearon dos configuraciones de Release para cada uno de los Pipelines de Producción y QA elaborados anteriormente, se definió que cuando se implemente un cambio en la rama de QA o de Producción, se ejecute automáticamente el despliegue de cada uno en sus respectivos entornos. En la ilustración 24, se puede ver las publicaciones realizadas y un historial para cada ambiente, la rama utilizada y el resultado del despliegue.

Ilustración 24 Release de los ambientes de desarrollo



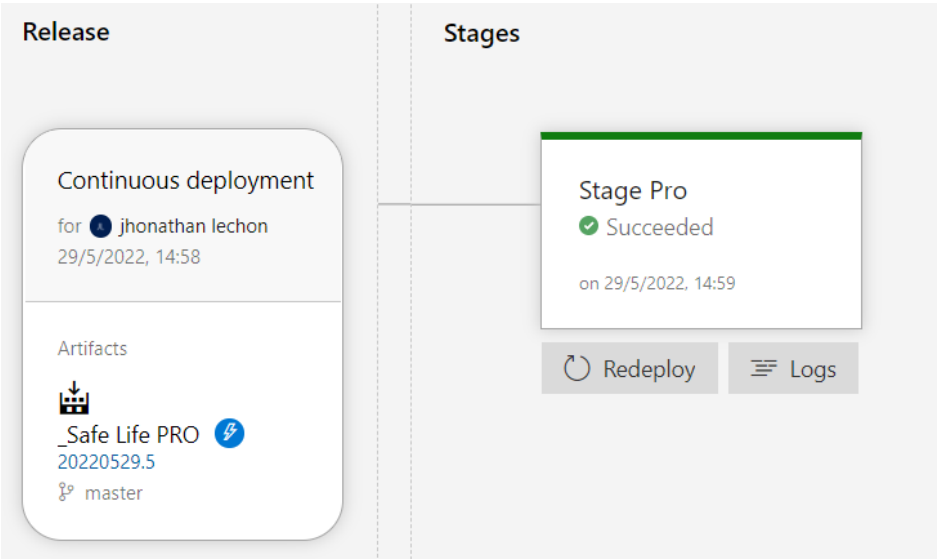
Los pasos para desplegar el aplicativo puede ser personalizado según las necesidades de cada proyecto, políticas internas o necesidades, para nuestro caso, se definió que, una vez ejecutado el Pipeline, se obtendrá el paquete como Artefacto y lo desplegará en la nube de Azure como una Web App. Esto debido a que, al ser del mismo fabricante, la comunicación entre Azure DevOps y el Portal de Azure mantienen plantillas preconfiguradas, sin embargo, es posible también con otras nubes o servidores de aplicaciones. En la ilustración 25 podemos observar la configuración global que elegimos para nuestro ambiente de producción.

Ilustración 25 Configuración del despliegue continuo



El primer resultado obtenido fue exitoso, tal y como se puede observar en la ilustración 26, podemos ver a la persona que ejecutó del Deploy, el artefacto utilizado, el resultado e incluso podríamos ver los logs generados por los Jobs del Pipeline o volver a Deployar nuestro aplicativo en el caso de ser necesario.

Ilustración 26 Resultado de CD



4.6. Lanzamiento y estimación del proyecto Ágil

Luego de tener listo todo el ambiente en la plataforma de Azure DevOps y configurado la mayor parte del ecosistema del aplicativo, procedimos con la planificación y desarrollo de la segunda iteración del proyecto, para esto se definieron las tareas y funcionalidades a implementar en este Sprint, haciendo uso de la metodología Ágil y las herramientas que nos proporciona la plataforma.

Los puntos para tratar fueron la creación e implementación de pruebas automatizadas y también la documentación de las Api utilizando Swagger.

La definición de los artefactos de Scrum se lo realizaron siguiendo las recomendaciones según metodología ágil, desde la planificación, hasta la definición de historias de usuario y ejecución de las actividades. Los pasos para seguir fueron:

4.6.1. Identificación de roles y el equipo de trabajo

El equipo cuenta con 3 Desarrolladores y 1 líder técnico, el líder técnico tendrá la función de Scrum Máster, los desarrolladores deberán realizar las pruebas y despliegue de las aplicaciones según lo que se necesite.

4.6.2. Identificación de los requisitos y elaboración de las historias de usuario

Los requisitos iniciales son puestos por el líder técnico y se tratan de mejorar iniciales del aplicativo API y como parte de adaptación a esta metodología, lo cual implica los siguientes Factures generales:

- Elaboración de Pruebas Automáticas
- Implementación de Documentación a las APIs

En base a esto se definieron las siguientes Historias de Usuario junto con la definición de cada uno, sus criterios de aceptación, puntos de historia y nivel de importancia, para lo cual se utilizaron las siguientes métricas de nivel de riesgo:

- Alto: De gran importancia
- Medio: Es importante que lo tenga, pero no fundamental
- Bajo: Es deseable que lo tenga, pero no es fundamental

4.6.3. Puntos de Historia

Los puntos se lo hicieron para los valores de 8, 16, 24 y 32, los cuales nos indican un aproximado de la estimación de esfuerzo para su implementación, el rango va desde historias muy simples de completar, hasta aquellas que son mucho más complejas de desarrollar.

4.6.4. Estimación de Puntos de Historia

Para elegir el valor de los puntos de historia de cada tarea, se utilizó la técnica llamada Planning Poker en una reunión general entre todo el equipo, esto con el fin de evaluar cada una de las historias de forma equitativa por cada uno de los desarrolladores, exponiendo sus puntos de vista y experiencia, de esta forma se pudo evaluar cada historia y su complejidad.

4.6.5. Historias de Usuario

Una vez creada y definidas las historias de usuario y luego realizar la estimación de cada una, así como su nivel de riesgo utilizando las técnicas y metodologías recomendadas, se obtuvo las siguientes historias de usuario:

Ilustración 27 Historia de Usuario HU05

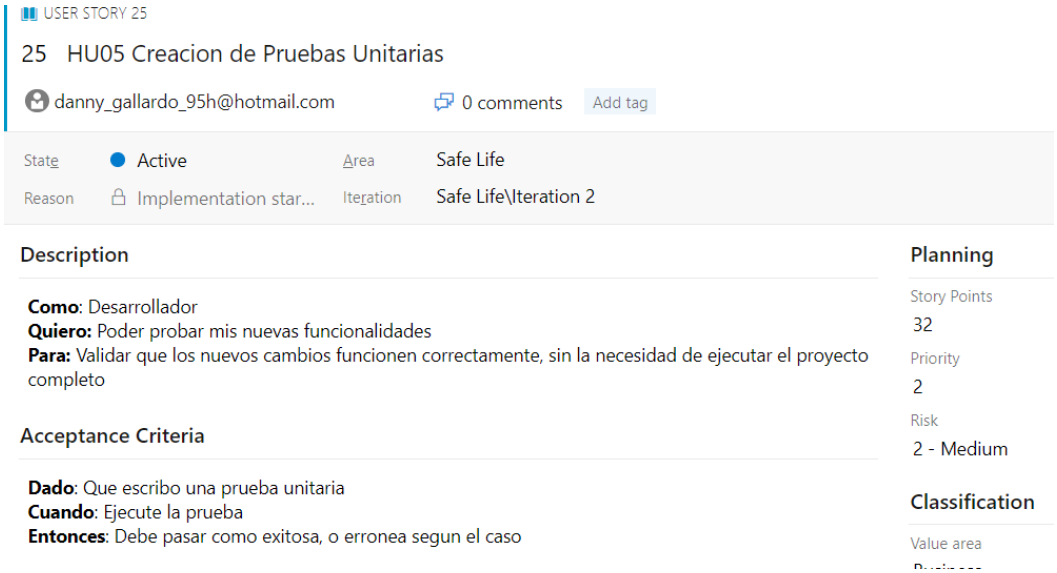
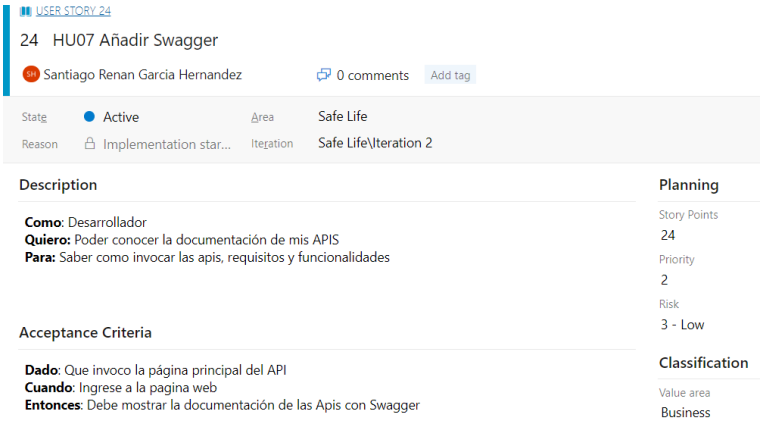


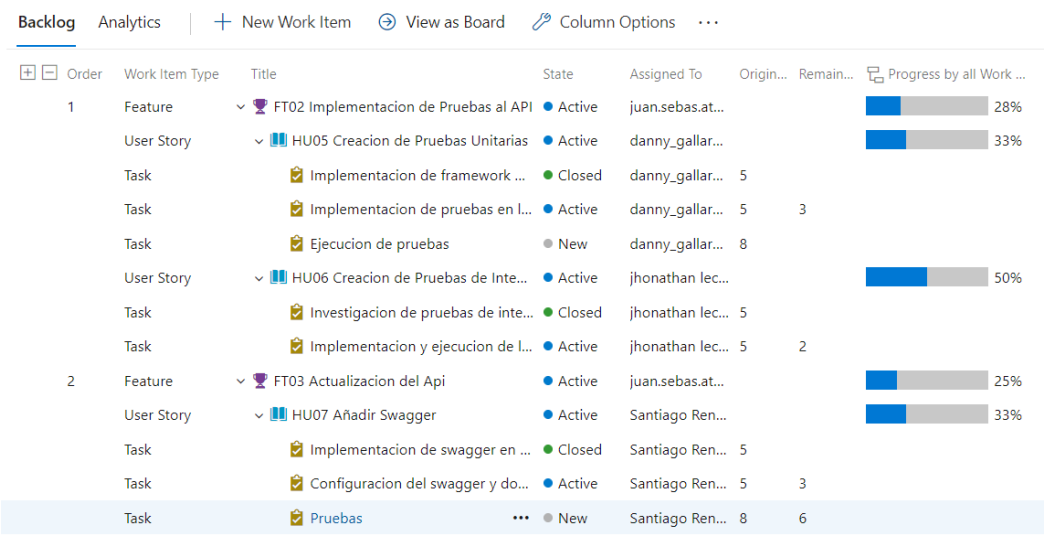
Ilustración 28 Historia de Usuario HU06

Ilustración 29 Historia de Usuario HU07



En la ilustración 30 podemos ver el Backlog de la segunda iteración, los Features, Historias de Usuario y Tareas, además, gracias a que este panel nos permite agregar o modificar columnas, se configuró la presentación del Backlog para indicar una barra de progreso de las actividades, horas estimadas y horas faltantes lo cual nos permite saber de manera más gráfica y rápida el progreso de cada desarrollador.

Ilustración 30 Backlog Segunda Iteración



Order	Work Item Type	Title	State	Assigned To	Origin...	Remain...	Progress by all Work ...
1	Feature	FT02 Implementacion de Pruebas al API	Active	juan.sebas.at...			28%
	User Story	HU05 Creacion de Pruebas Unitarias	Active	danny_gallar...			33%
	Task	Implementacion de framework ...	Closed	danny_gallar...	5		
	Task	Implementacion de pruebas en l...	Active	danny_gallar...	5	3	
	Task	Ejecucion de pruebas	New	danny_gallar...	8		
	User Story	HU06 Creacion de Pruebas de Inte...	Active	jhonathan lec...			50%
	Task	Investigacion de pruebas de inte...	Closed	jhonathan lec...	5		
	Task	Implementacion y ejecucion de l...	Active	jhonathan lec...	5	2	
2	Feature	FT03 Actualizacion del Api	Active	juan.sebas.at...			25%
	User Story	HU07 Añadir Swagger	Active	Santiago Ren...			33%
	Task	Implementacion de swagger en ...	Closed	Santiago Ren...	5		
	Task	Configuracion del swagger y do...	Active	Santiago Ren...	5	3	
	Task	Pruebas	New	Santiago Ren...	8	6	

4.7. Implementación y automatización Pruebas

Siguiendo las practicas DevOps y con la planificación creada para este Sprint se vio la necesidad de la creación de pruebas unitarias y de integración, esto con el fin de validar la funcionalidad del código y permitir evaluar y comprobar los nuevos requisitos de forma más rápida y gráfica.

4.7.1. Elaboración de Pruebas Unitarias y de Integración

Para la elaboración de las pruebas unitarias y de integración se utilizó el Framework XUnit y NUnit las cuales nos brindan un conjunto de librerías y herramientas que nos permiten el desarrollo y ejecución de pruebas.

El propósito de las pruebas unitarias será probar funcionalidades cortas y rápidas de diversos métodos, en la siguiente ilustración se puede ver la ejecución de las pruebas unitarias implementadas en este proyecto, se elaboraron 5 pruebas unitarias, validando ciertas funcionalidades de nuestro aplicativo.

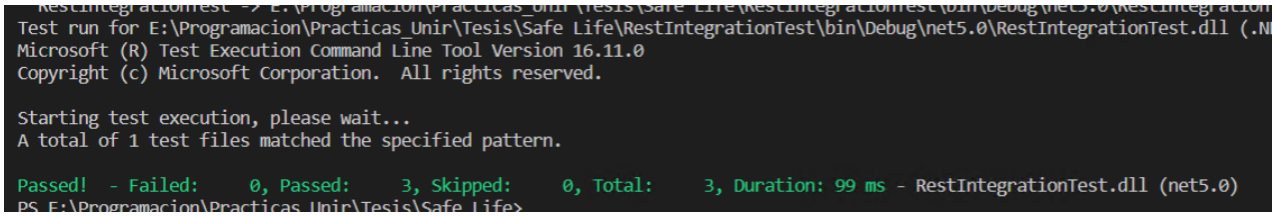
Ilustración 31 Ejecución de Pruebas Unitarias

```
  WebApi -> C:\Users\JhonathanAsus\Documents\VSCode\Safe Life\bin\Debug\net6.0\WebApi.dll
  ApiTest -> C:\Users\JhonathanAsus\Documents\VSCode\Safe Life\ApiTest\bin\Debug\net6.0\ApiTest.dll
Serie de pruebas para C:\Users\JhonathanAsus\Documents\VSCode\Safe Life\ApiTest\bin\Debug\net6.0\ApiTest.dll
Herramienta de línea de comandos de ejecución de pruebas de Microsoft(R), versión 17.2.0 (x64)
Copyright (c) Microsoft Corporation. Todos los derechos reservados.

Iniciando la ejecución de pruebas, espere...
1 archivos de prueba en total coincidieron con el patrón especificado.

Correctas! - Con error:     0, Superado:     5, Omitido:     0, Total:     5, Duración: 11 ms - ApiTest.dll
PS C:\Users\JhonathanAsus\Documents\VSCode\Safe Life>
```

Por otro lado, las pruebas de integración nos permitirán ver el flujo parcial o completo de la aplicación, para este proyecto se utilizaron para probar las respuestas de la API, es decir comprobar la respuesta que nos brinda al ejecutar una petición Http.

En la ilustración 32 se observa la ejecución de estas pruebas de Integración, para este caso elaboramos 3 pruebas que verifican el contenido de la petición a nuestra API.

Ilustración 32 Ejecución de Pruebas de Integración



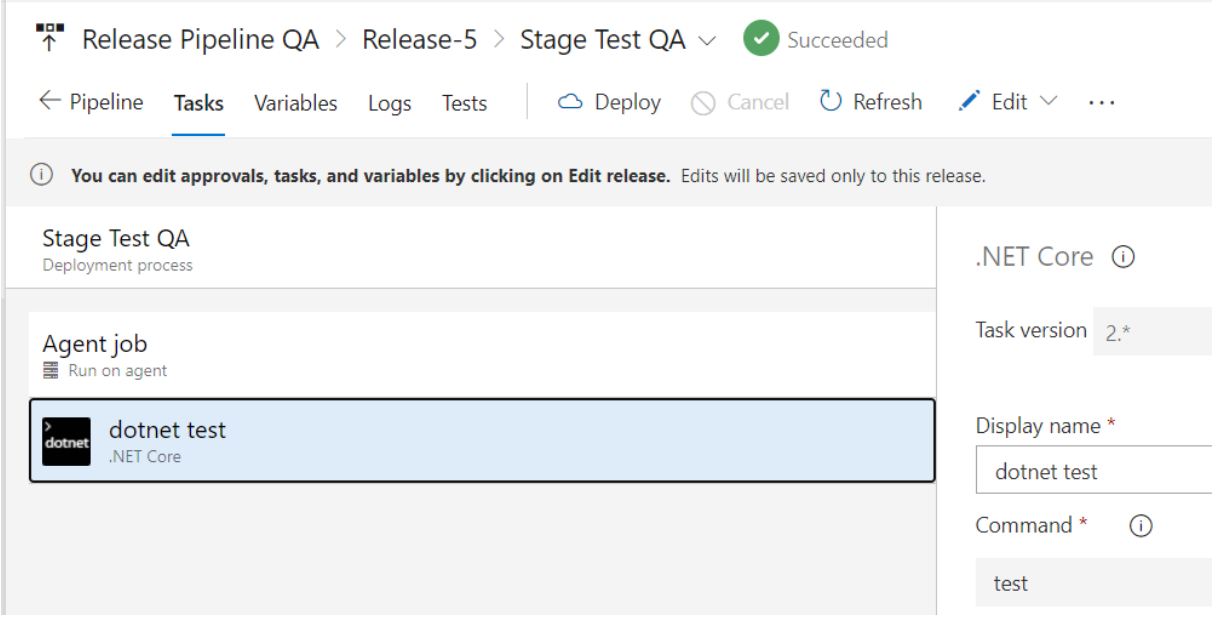
4.7.2. Automatización de Pruebas en Azure DevOps

Una vez configurada las pruebas en el aplicativo, se procedió a realizar la configuración de la ejecución de las pruebas en Azure DevOps. Para esto se tienen dos opciones:

- **Implementación de Pruebas en el Pipeline**: Esto nos permite realizar la ejecución de las pruebas internas como un Job en Pipeline, de esta forma podremos detener la ejecución del Pipeline en el caso de que alguna de estas pruebas falle.
- **Automatización de Pruebas en el Release Pipeline:** Estas pruebas se lo realizan una vez compilado y empaquetado el proyecto, se lo puede hacer como una parte del despliegue del aplicativo.

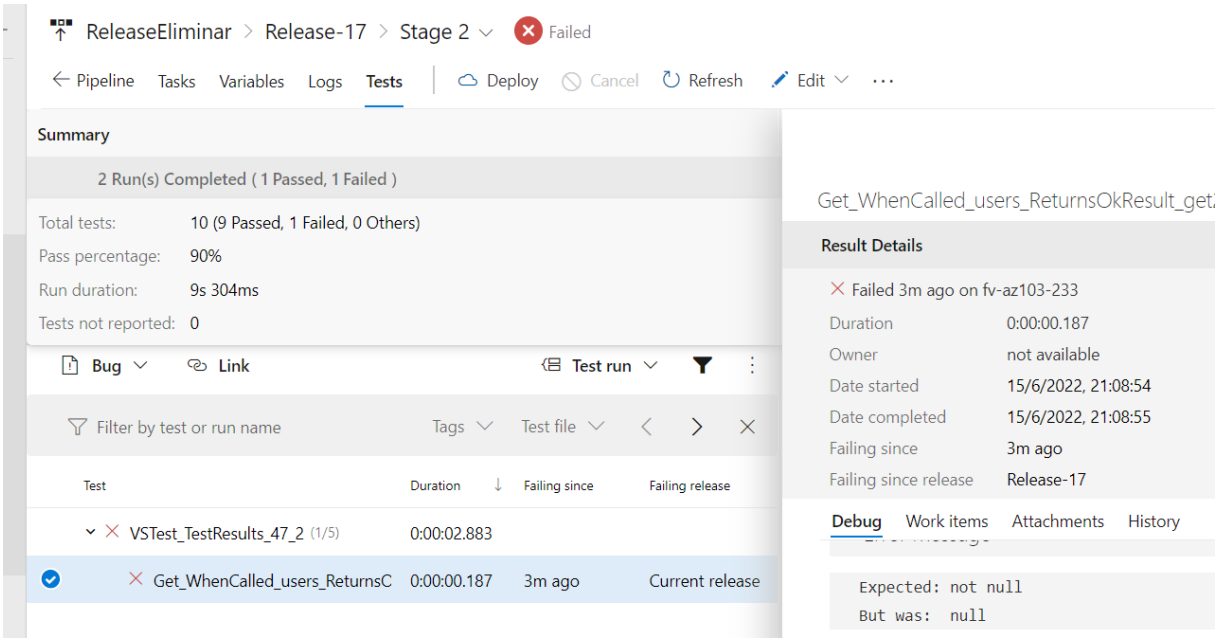
En este proyecto se optó por implementar las pruebas en el Release e implementar políticas de acuerdo con el éxito o fallo de estas, para ello, fue necesario realizar la configuración en el pipeline creado anteriormente en el cual se implementó un nuevo Stage encargado exclusivamente para la ejecución de las pruebas unitarias y de integración. Para la ejecución de pruebas se utilizó el Agente de DotNet como se puede observar en la figura 33, sin embargo, se pueden utilizar muchas otras herramientas de Testing.

Ilustración 33 Configuración de pruebas en Azure DevOps



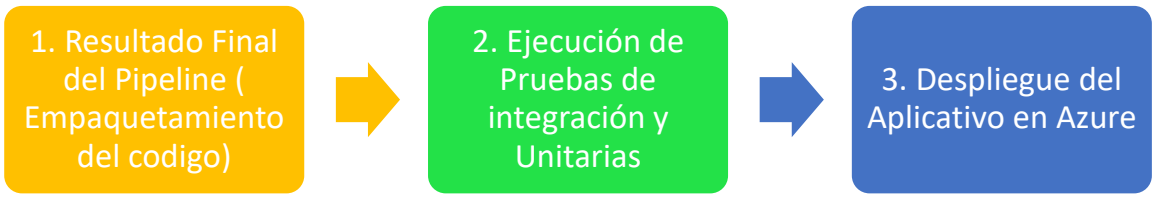
Inicialmente se ejecutaron pruebas tanto exitosas como erróneas para validar el correcto comportamiento de este entorno según las configuraciones, por ejemplo, una de las políticas definidas es que se pare el despliegue del aplicativo en el caso de encontrarse un error, tal y como se puede observar en la siguiente ilustración.

Ilustración 34 Caso de Error en una prueba



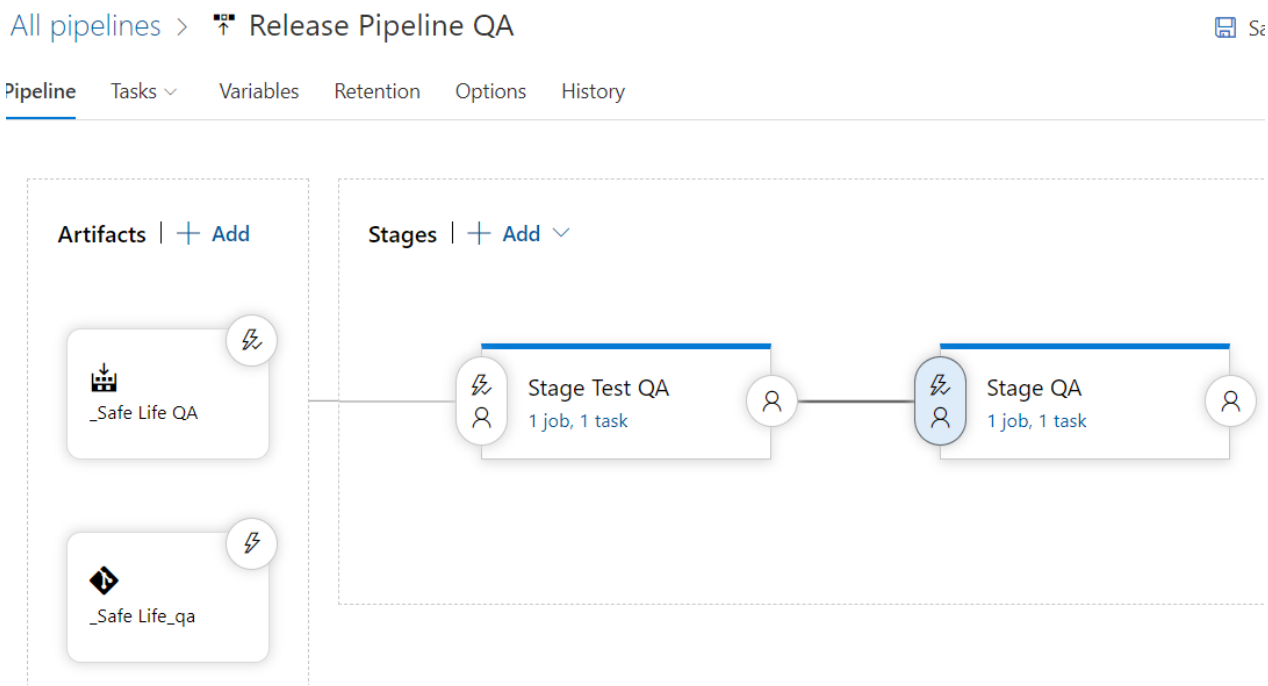
Una vez comprobado el correcto funcionamiento de las pruebas, se procedió a diseñar los pasos a realizar para el despliegue y se lo realizó de la siguiente manera:

Ilustración 35 Proceso de Despliegue Continuo



En la ilustración 36 se puede observar la configuración utilizada para la elaboración del despliegue del aplicativo y de la ejecución de las pruebas ejecutado por etapas, cabe mencionar que la configuración indicada se lo utilizó tanto para el ambiente de Producción como para QA ya que se ambos ambientes presentan configuraciones similares y la diferencia se encuentra en las configuraciones dentro de su servidor de aplicaciones, para nuestro caso fue desplegado como una Web App en Azure.

Ilustración 36 Diseño Final del Release Pipeline

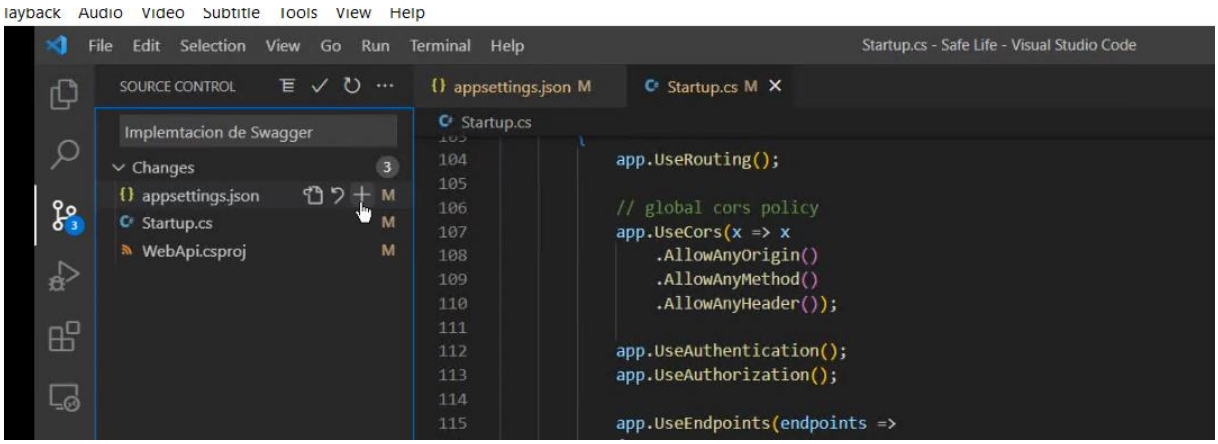


4.8. Prueba General del Entorno

Una vez configurado todo, y de acuerdo con la planificación del Sprint, se procede a probar todo el entorno de desarrollo y entrega continua, para ello haremos la demostración de la implementación de Swagger, en nuestro aplicativo empezamos con:

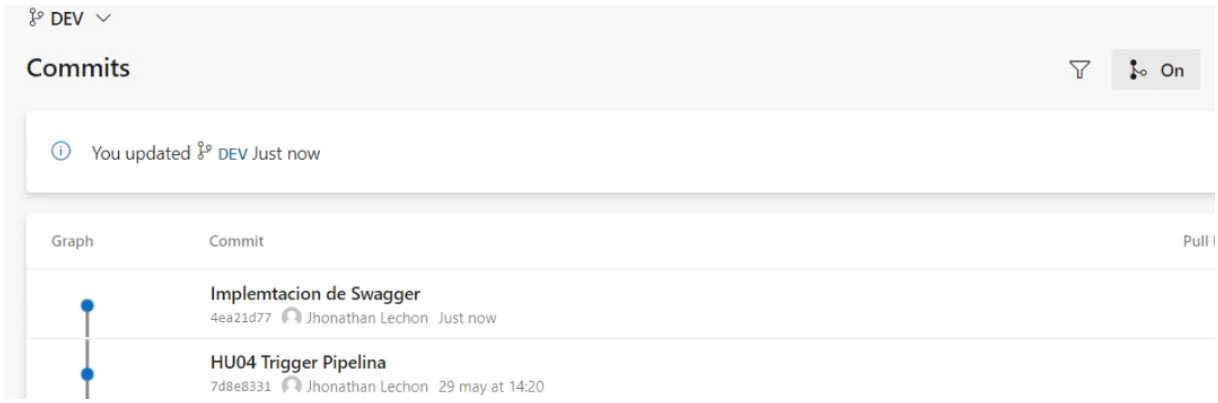
1. Elaboración del código y Commit de los cambios en la rama de Desarrollo desde nuestro editor de código.

Ilustración 37 Commit de los cambios en el IDE

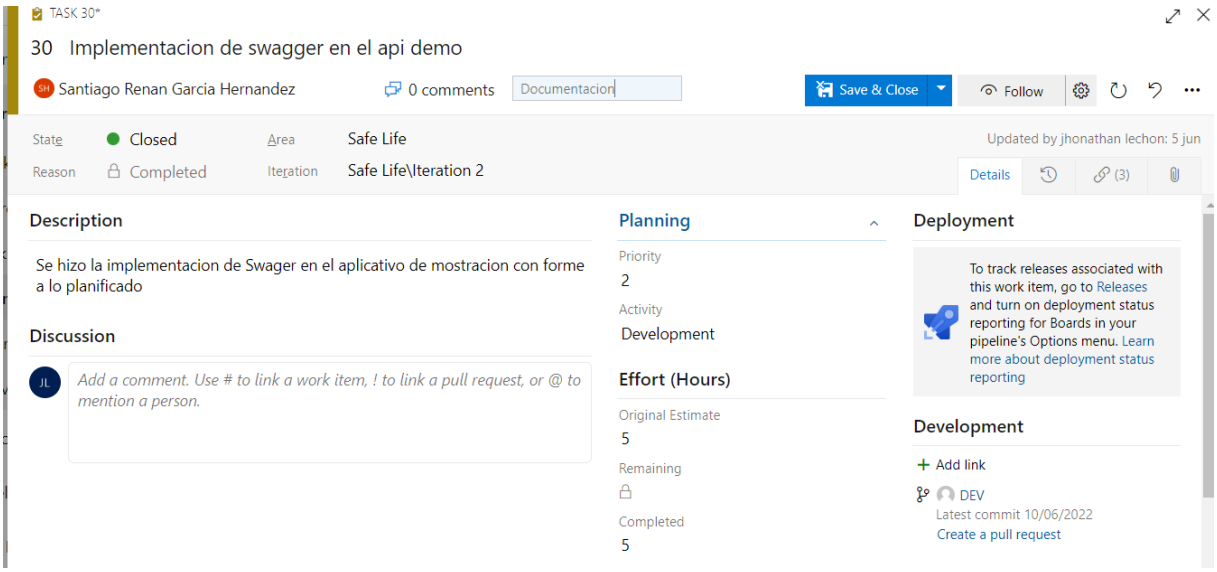


2. Push del código a nuestro repositorio en la rama de Desarrollo, Git nos permite la integración continua de Código.

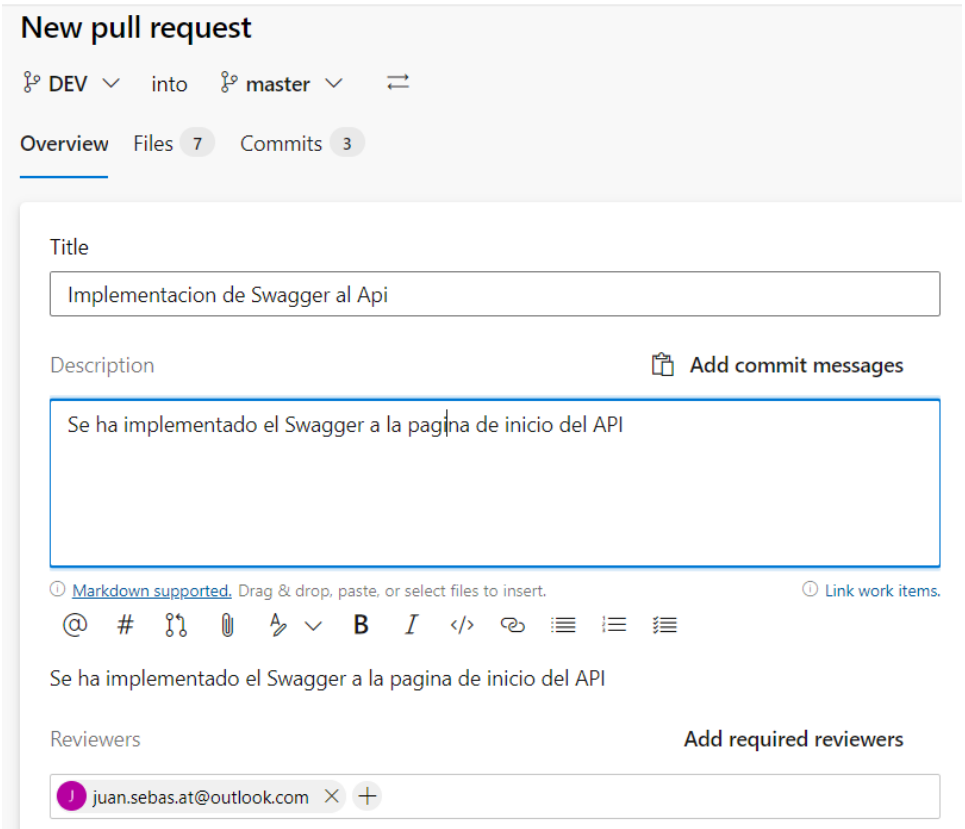
Ilustración 38 Verificación del código en la Rama Dev



3. Modificación de la tarjeta de tarea en el Product Backlog para indicar que la tarea se encuentra cerrada para su verificación junto con el enlace a la rama utilizada.

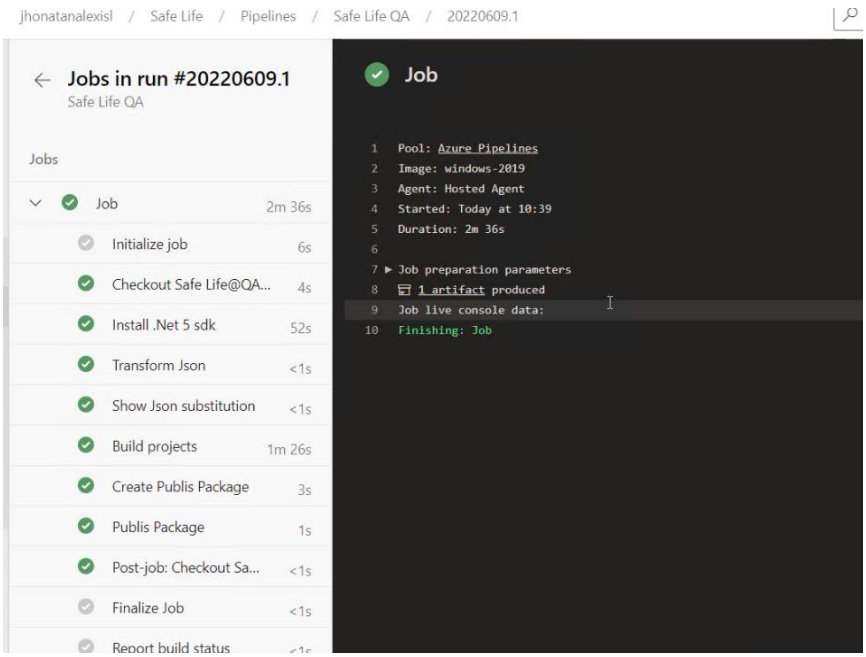


4. Una vez revisado los cambios, se procede a integrar los cambios a la rama de QA, esto se hace a través de un Pull Request a esta rama, en la cual se incluye al personal que será encargado de verificar el código antes de aceptar el cambio.



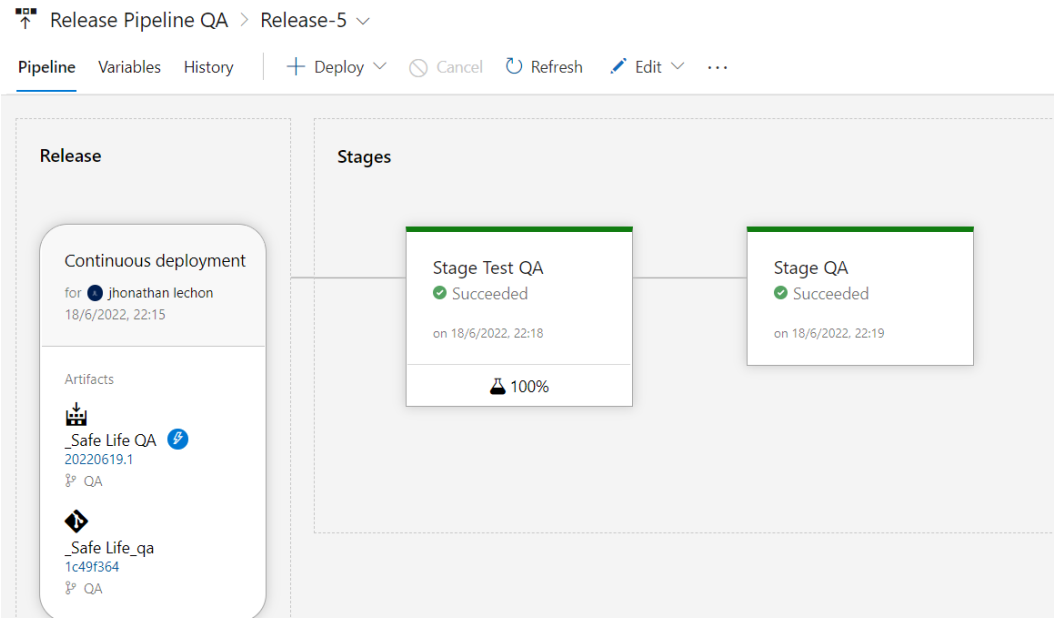
5. Una vez aprobado el Pull Request y validado los cambios, justo al momento de hacer el Merge en la Rama, se procede a lanzar el Trigger del Pipeline, es decir automáticamente se ejecuta el empaquetado de nuestro aplicativo. El resultado de la ejecución del Pipeline se puede observar en la siguiente ilustración.

Ilustración 39 Resultado del Pipeline



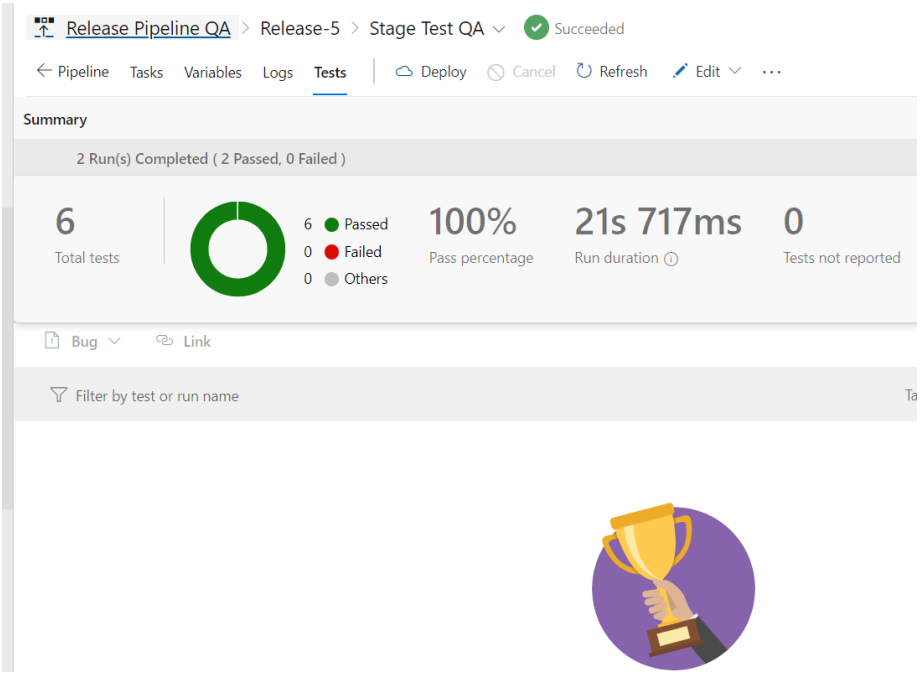
6. Gracias a la configuración de nuestro Release Pipeline se ejecuta automáticamente la ejecución automática de pruebas y posteriormente despliegue de nuestro aplicativo en Azure.

Ilustración 40 Resultado del despliegue del Aplicativo



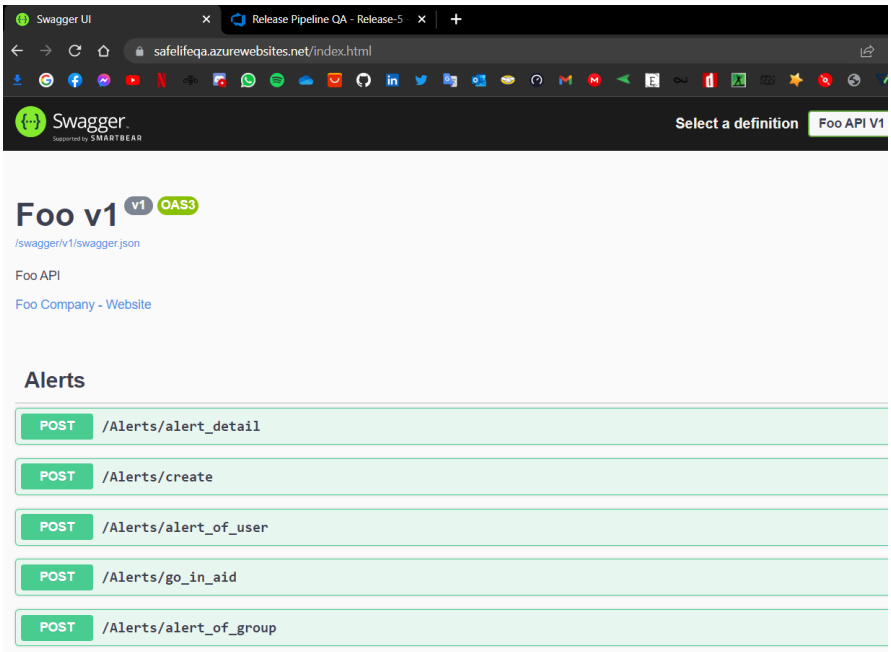
En la ilustración 41 se puede observar que el 100% de las pruebas fueron realizadas exitosamente, pero en el caso de ser necesario, también se puede ver a mayor detalle en el resumen y los registros de log de nuestro despliegue.

Ilustración 41 Resultado de la ejecución de Pruebas



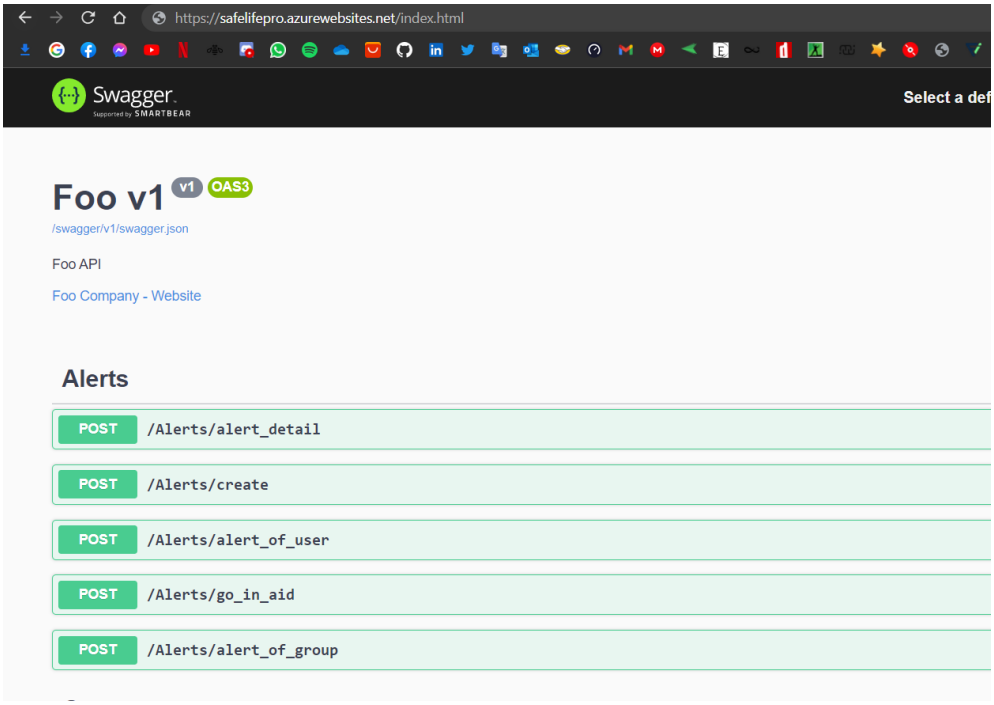
7. Visualizar los cambios realizados directamente en la aplicación, ya que ahora se encuentra lista y desplegada en nuestro ambiente de QA en el cual podremos realizar las diferentes pruebas internas.

Ilustración 42 Visualización de los cambios en QA



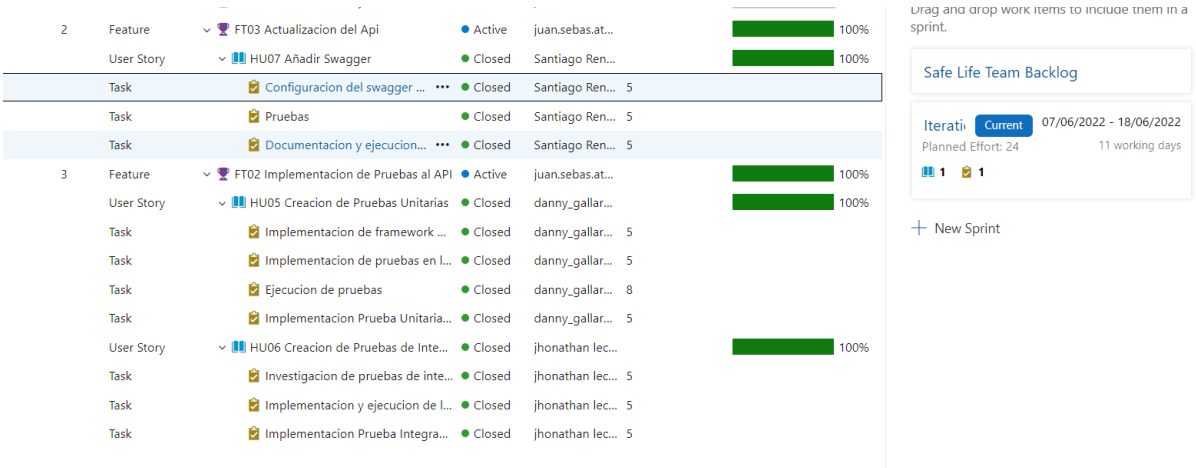
8. Una vez verificado todos los cambios se procede a realizar los mismos pasos descritos en el paso 4, es decir simplemente se realizar el Pull Request desde QA a nuestra rama de Producción, una vez aprobado el cambio e integrado los cambios, automáticamente se desplegará en nuestro ambiente de Producción.

Ilustración 43 Visualización de los cambios en Producción



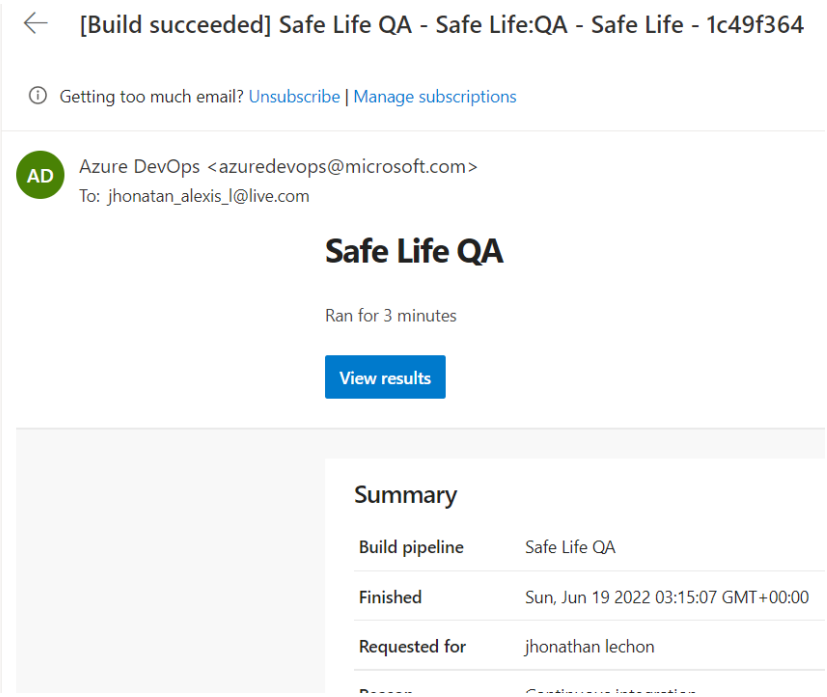
9. Finalmente, y conforme a nuestra metodología de trabajo, se procede a dar por terminado las historias de usuario relacionadas con esta fase del proyecto y se continua con el siguiente Sprint.

Ilustración 44 Backlog de las tareas completadas del Sprint 2



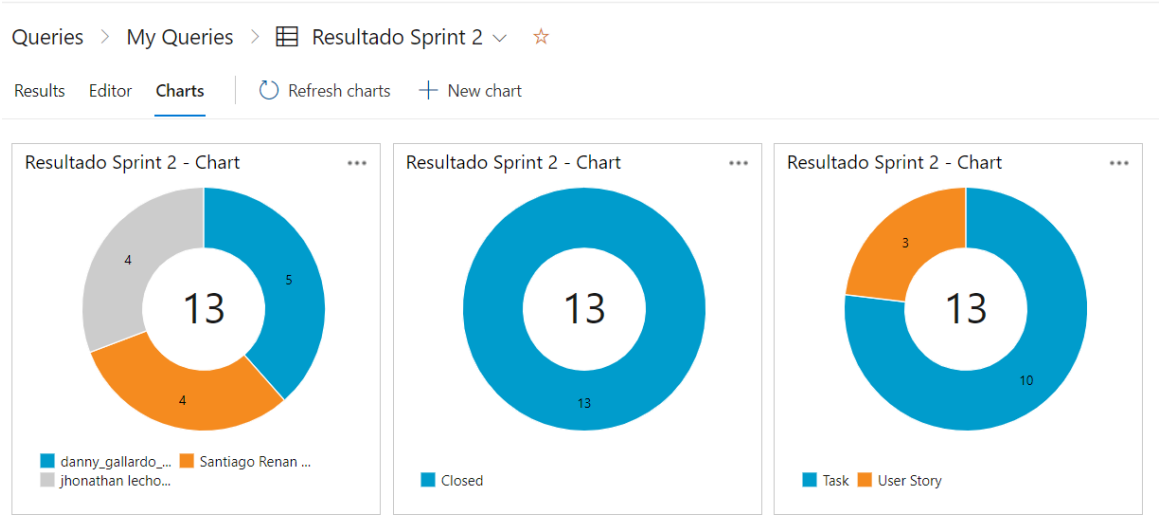
Incluso la herramienta nos permite notificar a la persona encargada a travéz de un correo electrónico, el cual se le puede configurar de acuerdo con las necesidades y políticas. En la siguiente ilustración observamos un ejemplo de la notificación obtenida al momento de desplegar nuestro aplicativo.

Ilustración 45 Ejemplo de Correo Electrónico recibido



10. Tal y cual se indica en la metodología, para finalizar, se realiza el Sprint Review, para esto se puede ayudar de métricas y estadísticas que nos proporciona el propio sitio de Azure DevOps y con esto dar por finalizado el Sprint e iniciar el ciclo nuevamente.

Ilustración 46 Cuadro de resultado del Sprint



4.9. Presentación de los Resultados

En este capítulo se presentarán los resultados que se obtuvieron en la migración de este proyecto y el análisis de los datos obtenidos utilizando para esto el mismo entorno de Azure DevOps a través de los Querys y Graficas que nos proporciona. También se mostrarán las mejoras que se obtuvieron con los procesos automatizados y la metodología aplicada, obtenida esta información se hará el análisis con los objetivos planteados.

4.9.1. Dashboard

En este tablero se muestra a breve rasgos el desempeño del equipo, las actividades realizadas, pendientes, lanzamientos y resultado de las pruebas ejecutadas, este tablero es personalizable así que se puede ir ajustando a las necesidades futuras del proyecto.

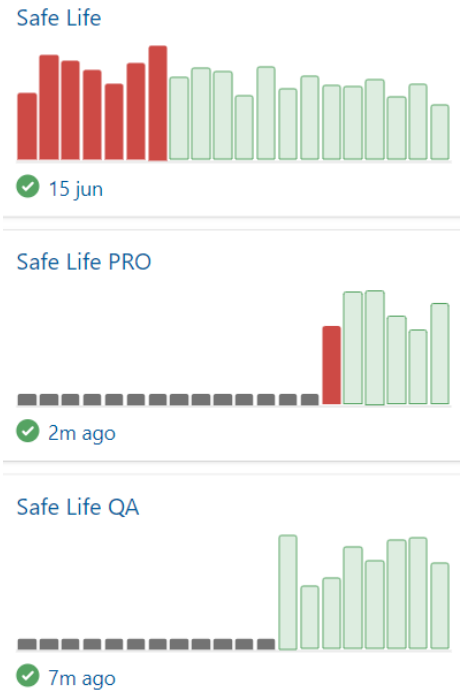
En las siguientes ilustraciones se muestran los tableros que fueron configurados para este proyecto y se explicaran a detalle los resultados obtenidos dentro de cada tablero mostrado:

Tablero de Ejecución de Pipelines

En este tablero se muestran las ultimas ejecuciones del pipeline y su resultado. Las barras en color verde representan a los pipelines ejecutados correctamente, las barras en color rojo muestran aquellos que tuvieron un error al momento de su ejecución, también se muestra el orden cronológico de derecha a izquierda desde el más reciente hasta el más antiguo, la altura de las barras corresponde a la etapa en la que ocurrió el error, por lo que entre más alta la barra más cerca estuvo de finalizar su ejecución.

En la ilustración 47 se muestra un historial de las ejecuciones realizadas en los diferentes ambientes creados para este proyecto, y se agrega una gráfica de un pipeline utilizado para realizar las pruebas iniciales para estabilizar ambiente debido a eso presenta más fallas iniciales. Este tablero al presentarse inicialmente es muy útil pues podemos observar se forma global rápidamente a cualquier miembro del equipo si se presenta un error y de esta forma dirigirse rápidamente al log a verificar el motivo.

Ilustración 47 Historial de ejecución del Pipeline en los diferentes ambientes

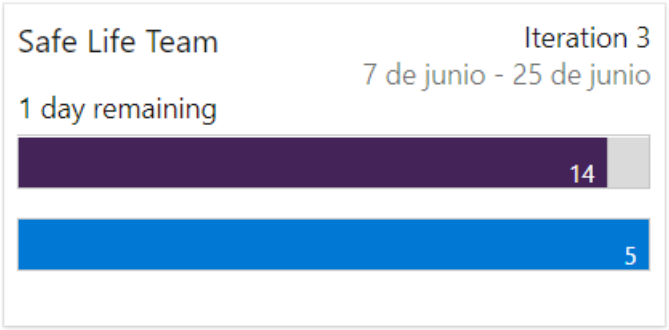


Tablero de desempeño del equipo de trabajo por Sprint

Indica el desempeño logrado por cada miembro del equipo en base a los ítems de trabajo o puntos de historia realizado en el sprint correspondiente, de tal manera que cada líder de equipo puede tener un entendimiento claro del desempeño del sprint actual, y tomar decisiones al respecto.

En la siguiente ilustración se muestra el grafico de desempeño correspondiente al nuevo Sprint y muestra las tareas finalizadas en esos días de trabajo.

Ilustración 48 Grafico de desempeño del Equipo por Sprint



Descripción General de Release Pipeline

En esta vista podemos ver el estado del Release Pipeline, en el cual podremos ver resumidamente los procesos y plataformas utilizadas para su despliegue, así como el estado con el que concluyo cada uno, este panel es configurable y se puede seleccionar el ambiente que deseemos y agregar más información que creamos necesaria.

En la siguiente ilustración se muestra de forma resumida el Release Pipeline del entorno de QA y sus diferentes lanzamientos y estados indicando en mayor tamaño el último lanzamiento realizado.

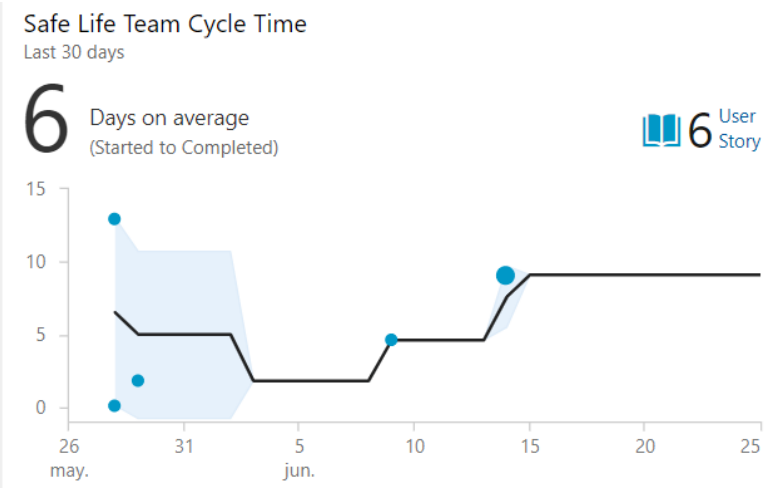
Ilustración 49 Resultado del Release Pipeline en el ambiente de QA

Tiempo del Ciclo y Plazo

Se puede ver este grafico para ver el promedio y las desviaciones estándar para estimar los tiempos de entrega. Siempre que se genere un elemento de trabajo se puede utilizar este grafico para poder estimar en cuanto tiempo el equipo de trabajo podrá finalizar sus tareas, por otra parte, la desviación estándar del equipo nos indica la variabilidad de la estimación, de esta manera podremos mejorar nuestra estimación de tiempos de entrega y de trabajo.

En la siguiente ilustración el tiempo de ciclo promedio es de seis días, la desviación estándar es aproximadamente cuatro días. Con esta información podremos estimar que el equipo de trabajo completara sus respectivas historias de usuario entre 2 y 10 días después de empezar con su tarea. Mientras más estrecha sea la desviación estándar se espera que sean más predecibles las estimaciones.

Ilustración 50 Tiempo de Ciclo del Equipo



Burndown Chart

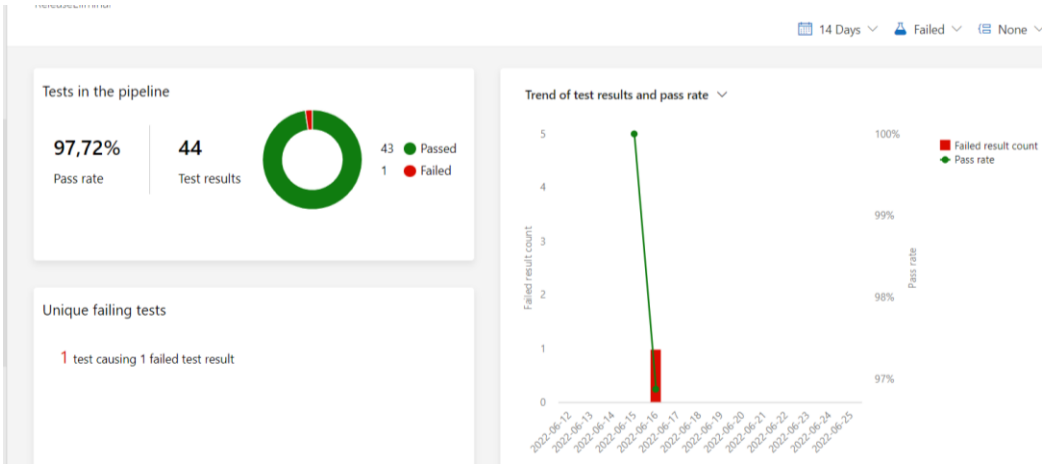
Esta herramienta nos permite visualizar el avance del equipo de trabajo y su tendencia a cumplir con el objetivo del Sprint, le permite al equipo auto organizarse y ajustar su plan de trabajo cada día. En el siguiente grafico podemos observar el trabajo pendiente del Sprint, también el avance y el ritmo de progreso. Podemos ver que el equipo se trabaja a una velocidad fija, es decir en promedio tardan en realizar su actividad el mismo tiempo, pero debido a que aún se está trabajando en implementar la metodología, se observa que aún falta madurar la forma de trabajar a nivel de la planificación de las historias de usuario.

Ilustración 51 Burdown Chart

4.9.2. Pruebas en los Despliegues

Cada Release Pipeline creado nos permite visualizar métricas de los diferentes componentes ejecutados, para nuestro proyecto en el Stage Test QA podremos observar un reporte de las pruebas ejecutadas en cada ejecución del Release tal cual se muestra en la siguiente ilustración.

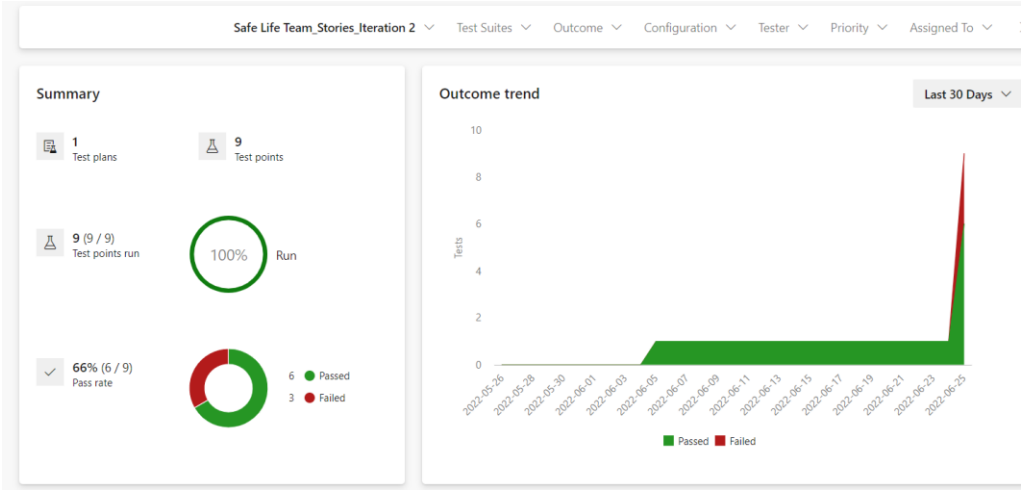
Ilustración 52 Reporte de Pruebas del Release Pipeline



4.9.3. Plan de Pruebas

En este apartado se pueden ver a forma de resumen y con grafica el progreso de todas las pruebas asignadas a cada historia de usuario que se ejecutaron en este Sprint. En la siguiente ilustración se puede observar en el cuadro resumen de todas las pruebas ejecutadas manualmente, y un gráfico con el número de pruebas exitosas y erróneas ocurridas en función del tiempo.

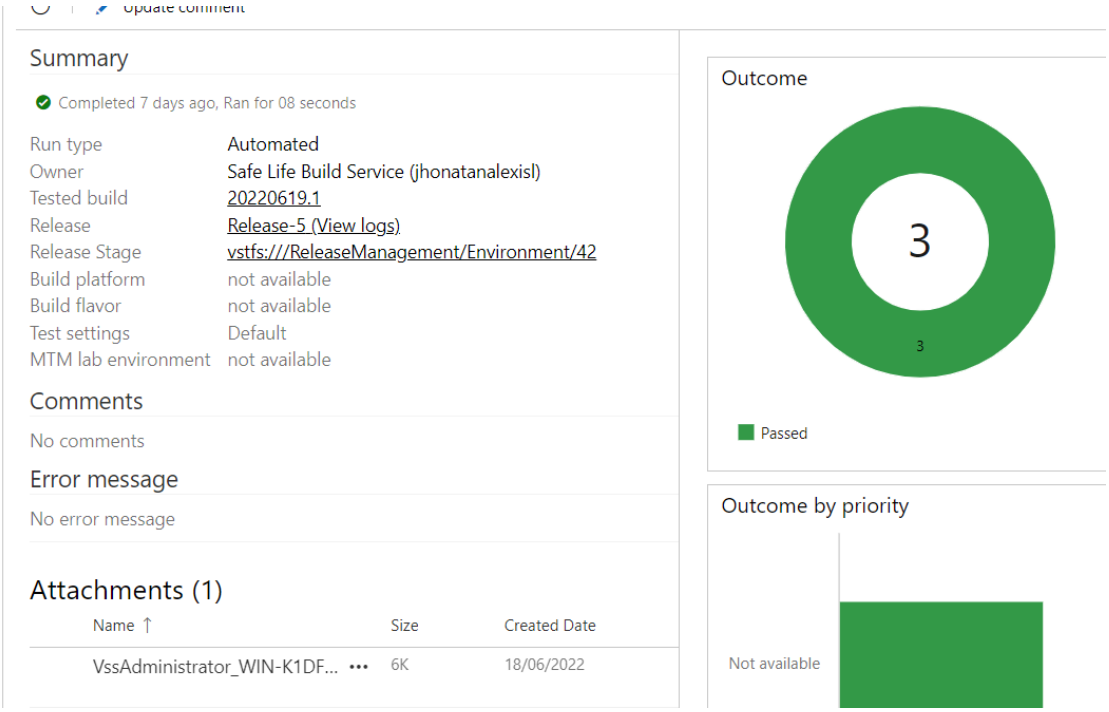
Ilustración 53 Reporte de Progreso del plan de Pruebas



Es posible también observar el resumen de las pruebas ejecutadas en nuestro Pipeline Release en cada despliegue, esto gracias a un historial que la herramienta nos proporciona, gracias a esto podremos ir revisando aquellos puntos en los que nuestro aplicativo tiene mayor problema.

En la siguiente ilustración podemos observar el resultado de las pruebas de integración al ser ejecutadas en uno de los Ralease.

Ilustración 54 Resumen de ejecución de pruebas automáticas



5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1.Conclusiones

- El uso de prácticas DevOps puede ser aplicado diferentes proyectos sin importar el tamaño del equipo o presupuesto ya que depende más que todo en la iniciativa del equipo por querer sobresalir del resto, su sed de conocimiento y agilidad para adaptarse a las nuevas tecnologías y herramientas.
- El uso de herramientas y plataformas ayudan y permiten una adopción rápida de fácil de DevOps, sin embargo, hay que tener en cuenta que no depende en su totalidad de estas herramientas, sino de la cultura y motivación del equipo para adaptarse a esta nueva forma de trabajo.
- La plataforma Azure DevOps ha permitido implementar el marco de trabajo Agile basado en Scrum de forma fácil e intuitiva gracias a su conjunto de herramientas dentro de su entorno basado en la nube. El entorno Azure DevOps nos brindó este conjunto de herramientas listas para usar y gratuitas, durante este proyecto se mostró la facilidad de su implementación y adopción que nos permitieron la implementación de la integración y despliegue continuos, además de pruebas automatizadas y seguimiento del proyecto.
- Las métricas obtenidas y configuradas en el Dashboard de nuestro proyecto nos permitieron monitorear durante el proceso de desarrollo, desde el rendimiento de nuestro equipo de trabajo hasta el estado del aplicativo, la integración continua, el despliegue y estado de las pruebas manuales y automáticas. Todas estas métricas obtenidas nos permitirán ir mejorando continuamente.
- La implementación de CI/CD en aplicativos permite optimizar el tiempo al automatizar procesos repetitivos, nos brinda mayor fiabilidad y nos permite corregir de forma rápida cualquier error presentado durante el despliegue del aplicativo. Al implementar los diferentes entornos de desarrollo ayudándonos del repositorio de código fuente Git se pudo separar las distintas etapas de desarrollo brindándonos la facilidad de versionamiento y el control de cambios, permitiéndonos tener un mejor control de errores que se pueden presentar en los diferentes entornos de trabajo.

- Se ha migrado el Backend del aplicativo desarrollado inicialmente por una Startup de manera tradicional a un marco de trabajo Agile basado en DevOps de forma correcta, y satisfactoria, por lo cual ahora queda a disposición de la empresa su implementación en sus otros proyectos.

5.2. Recomendaciones

- Se recomienda la creación de diferentes entornos de desarrollo, esto con el fin de prepararse para diferentes circunstancias, para esto igual es necesaria la creación de diferentes pipelines de construcción y despliegue para su implementación. Todo esto nos permitirá tener un mejor control de errores y nos permitirá tener mayor fiabilidad en cada entrega realizada.
- Se recomienda investigar y estudiar con mayor profundidad todas las herramientas que nos proporciona Azure DevOps, ya que se va actualizando continuamente, agregando nuevas funcionalidades e integraciones con terceros. Todo esto con el fin de explotar al máximo sus capacidades y aportar de mejor manera con la metodología y marco de trabajo.
- Se recomienda la implementación de métricas basadas en logs, como por ejemplo Logstash, esto con el fin de monitorear el estado y rendimiento del aplicativo en producción.

5.3. Líneas de trabajo futuro

- Una posible línea de trabajo a investigar podría ser la integración del entorno Azure DevOps con terceros, esto debido a que puede ser de gran importancia para proyectos que utilicen otras tecnologías y ambientes de desarrollo.
- Durante el desarrollo del proyecto se pudo observar que en el apartado de los Release Pipeline, existe una gran cantidad utilitarios que podrían ser de ayuda para las diferentes necesidades de despliegue de los aplicativos, por lo que se podría investigar sobre esos utilitarios y en qué tipo de proyectos pueden ser utilizados.

6. REFERENCIAS BIBLIOGRÁFICAS

Axelsson, J. (2015). Architectural allocation alternatives and associated concerns in cyber-physical systems: A case study. *ACM International Conference Proceeding Series*, *07-11-September-2015*. https://doi.org/10.1145/2797433.2797448

Azure DevOps Services | Microsoft Azure. (n.d.). Retrieved April 23, 2022, from https://azure.microsoft.com/es-es/services/devops/#overview

Chaturvedi, S. (2021, December 22). *What is DevOps, Why a Startup Needs It?* https://www.finextra.com/blogposting/19718/what-is-devops-why-a-startup-needs-it

Cukier, D. (2013). DevOps patterns to scale web applications using cloud services. *SPLASH 2013 - Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, and Applications: Software for Humanity*, 143–152. https://doi.org/10.1145/2508075.2508432

Debois, P. (2008). Agile infrastructure and operations: How infra-gile are you? *Proceedings - Agile 2008 Conference*, 202–207. https://doi.org/10.1109/AGILE.2008.42

El método Lean Startup. (2012).

Garcia, J., Plat, B. J., & Salazar, P. (n.d.). *Métodologías Ágiles en el Desarrollo de Soware*. Retrieved April 30, 2022, from www.agileuniverse.com.

Hordashnyk Yevhenii. (2020, October 10). *【DevOps for Startups】| Bitter Truth and Useful Tips 2020 - ALPACKED | DevOps for startups*. https://alpacked.io/blog/devops-for-startups/

Hüttermann, M. (2012). Beginning DevOps for Developers. *DevOps for Developers*, 3–13. https://doi.org/10.1007/978-1-4302-4570-4_1

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A Survey of DevOps Concepts and Challenges. *ACM Computing Surveys (CSUR)*, *52*(6). https://doi.org/10.1145/3359981

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016). Relationship of DevOps to Agile, Lean and Continuous Deployment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10027 LNCS*, 399–415. https://doi.org/10.1007/978-3-319-49094-6_27

Microsoft. (2020). *What is Azure Test Plans?* https://docs.microsoft.com/en-us/azure/devops/test/overview?view=azure-devops

Microsoft. (2021, August 11). *Create a CI/CD pipeline for GitHub repo using Azure DevOps Starter*. https://docs.microsoft.com/en-us/azure/devops-project/azure-devops-project-github

Microsoft. (20222). *Release pipelines*. https://docs.microsoft.com/en-us/azure/devops/pipelines/release/?view=azure-devops

Mishra, A., & Otaiwi, Z. (2020a). DevOps and software quality: A systematic mapping. *Computer Science Review*, *38*, 100308. https://doi.org/10.1016/J.COSREV.2020.100308

Mishra, A., & Otaiwi, Z. (2020b). DevOps and software quality: A systematic mapping. *Computer Science Review*, *38*, 100308. https://doi.org/10.1016/J.COSREV.2020.100308

Montoya, S., & Ocampo, A. (2020, May 12). *Conoce 5 herramientas para integración y entrega continua con DevOps*. https://www.pragma.com.co/blog/conoce-5-herramientas-para-integracion-y-entrega-continua-con-devops

Neeraj Mishra. (2022). *Does a Startup Really Need DevOps? - The Crazy Programmer*. https://www.thecrazyprogrammer.com/2021/10/does-a-startup-really-need-devops.html

Patrick Debois, by, Jez Humble, by, Molesky, J., Eric Shamow, by, Lawrence Fitzpatrick, by, Dillon, M., Bill Phifer, by, & Dominica DeGrandis, by. (2011). Devops: A Software Revolution in the Making? Opening Statement Why Enterprises Must Adopt Devops to Enable Continuous Delivery Devops at Advance Internet: How We Got in the Door The Business Case for Devops: A Five-Year Retrospective Next-Generation Process Integration: CMMI and ITIL Do Devops Devops: So You Say You Want a Revolution? *The Journal of Information Technology Management Cutter IT Journal*, *24*(8). www.cutter.com

Redacción KeepCoding. (2022, February 16). *¿Cómo implementar DevOps en tu empresa?* https://keepcoding.io/blog/como-implementar-devops-en-tu-empresa/

RIVERA CARO, E. H. (2019). *IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL EN ANDROID PARA MOSTRAR LOS ATRACTIVOS TURÍSTICOS DE LA RESERVA NACIONAL DE LACHAY, APLICANDO LA METODOLOGÍA SCRUM YHERRAMIENTAS DEVOPS* [UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA]. http://repositorio.untels.edu.pe/jspui/bitstream/123456789/278/1/Rivera_Ederr_Trabajo_Suficiencia_2019.pdf

Rossberg, J. (2019). An Overview of Azure DevOps. *Agile Project Management with Azure DevOps*, 37–66. https://doi.org/10.1007/978-1-4842-4483-8_2

Sachdeva, S. (2016). Scrum Methodology. *International Journal Of Engineering and Computer Science (IJECS) International Journal Of Engineering And Computer Science*, *5*, 16793–16799. https://doi.org/10.18535/ijecs/v5i6.11

Sarma, S. (n.d.). *Azure DevOps vs Jenkins: Who wins the battle? - Aspire Systems*. Retrieved April 23, 2022, from https://blog.aspiresys.com/infrastructure-managed-services/azure-devops-vs-jenkins-who-wins-the-battle/

Senapathi, M., Buchan, J., & Osman, H. (2018). DevOps capabilities, practices, and challenges: Insights from a case study. *ACM International Conference Proceeding Series*, *Part F137700*. https://doi.org/10.1145/3210459.3210465

Ståhl, D., & Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, *87*(1), 48–59. https://doi.org/10.1016/J.JSS.2013.08.032

Taylor, R. (n.d.). *Introduction to Test Automation and DevOps: A Case Study*. Retrieved April 23, 2022, from http://uploads.pnsqc.org/2019/papers/Taylor-Introduction-to-Test-Automation-and-DevOps.pdf

Toromoreno, S. (2021). *Demostración de proceso devops para el desarrollo de una aplicación web: sistema de donación de recursos.* [EPN]. https://bibdigital.epn.edu.ec/handle/15000/21948

Velencia, H. (2021, August 5). *Azure DevOps para automatizar tus áreas y procesos internos*. https://www.inbest.cloud/comunidad/azure-devops-para-automatizar-tus-%C3%A1reas-y-procesos-internos

West, D., & Groll, J. (2017). The Convergence of Scrum and DevOps for an Agile IT Organization. *Scrum.Org*. https://puppet.com/resources/whitepaper/state-of-devops-report.

Willis, J. (2010, July 16). *What Devops Means to Me - Chef Blog | Chef*. What Devops Means to Me. https://www.chef.io/blog/what-devops-means-to-me

Anexo A. Código Fuente de Pipeline utilizado

```
trigger:
- azure-pipelines-delete
pool:
  vmImage: 'windows-2019'
variables:
  appsettingsfile: appsettings.json
  MongoDatabaseSettings.ConnectionString: 'mongodb+srv://saac:saac@cluster0.d2izz.mongodb
.net/?retryWrites=true&w=majority'
  buildConfiguration: 'Release'
steps:
- task: UseDotNet@2
  displayName: 'Install .Net 5 sdk'
  inputs:
    version: '5.0.x'
    performMultiLevelLookup: true
- task: FileTransform@2
  displayName: "Transform Json"
  inputs:
    folderPath: '$(System.DefaultWorkingDirectory)/**/'
    xmlTransformationRules: ''
    jsonTargetFiles: '**/$(appsettingsfile)'
- bash: |
    cat $(appsettingsfile)
  displayName: "Show Json substitution"
- task: DotNetCoreCLI@2
  displayName: 'Build projects'
  inputs:
   command: 'build'
   projects: '**/WebApi.csproj'
   arguments: '--configuration $(buildConfiguration)'
- task: DotNetCoreCLI@2
  displayName: 'Create Publis Package'
  inputs:
   command: publish
   publishWebProjects: true
   projects: '**/WebApi.csproj'
   arguments: '--configuration $(buildConfiguration) --
output $(Build.ArtifactStagingDirectory)'
   zipAfterPublish: true
- task: PublishBuildArtifacts@1
  displayName: 'Publis Package'
  inputs:
   PathtoPublish: '$(Build.ArtifactStagingDirectory)'
   ArtifactName: 'AppApi'
```