Implementación de metodologías DevOps para que MIPYMES en Guatemala puedan recuperarse de la crisis causada por el COVID-19 en el año 2020

Helmuth Andrés Nistal Urizar

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación de metodologías DevOps para que MIPYMES en Guatemala puedan recuperarse de la crisis causada por el COVID-19 en el año 2020

Trabajo de graduación en modalidad de tesis presentado por
Helmuth Andrés Nistal Urizar
para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,
2020

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación de metodologías DevOps para que MIPYMES en Guatemala puedan recuperarse de la crisis causada por el COVID-19 en el año 2020

Trabajo de graduación en modalidad de tesis presentado por
Helmuth Andrés Nistal Urizar
para optar al grado académico de Licenciado en Ingeniería en Ciencias de la Computación y Tecnologías de la Información

Guatemala,
2020

Vo. Bo. :

(f)_____

Tomás Gálvez

Tribunal Examinador:

(f)_____

Tomás Gálvez

(f) _____

Alhvi Balcarcel

(f) _____

Douglas Barrios

Fecha de aprobación: Guatemala, 15 diciembre de 2020

Prefacio

Desde pequeño me ha apasionado la tecnología y mi país, Guatemala. Me enorgullece presentar este proyecto para poder ayudar a la población guatemalteca. El año 2020 se caracterizó por los efectos de COVID-19, específicamente en el área económica de Guatemala. Gran parte del PIB depende de negocios de carácter informal y estos han sido detenidos debido a las medidas preventivas para reducir la tasa de contagio de la pandemia. Muchos negocios para subsistir han buscado alternativas tecnológicas para respaldar sus procesos de negocios, sin embargo esto representa un esfuerzo económico adicional.

Nunca me imaginé que es posible tener impacto en la economía guatemalteca al facilitar varias herramientas para varias personas. Considero que cada módulo de este proyecto representa cada uno de los conocimientos que he aprendido durante mis estudios universitarios. 2020 sin duda ha sido un año inolvidable para muchas personas de todo el mundo, espero que este recuerdo no solo sea del todo negativo. Esta situación nos ha permitido cambiar de mentalidad para mejorar en varias situaciones. Creo en que todos como equipo podemos ayudarnos unos a los otros en cualquier momento. Este trabajo es mi pequeña contribución.

Agradezco grandemente a cada persona que me ha acompañado en este camino universitario. Iniciando por mi familia que me han apoyado grandemente desde el inicio. También a amigos, equipos deportivos, compañeros de clase, compañeros Devpack Group y profesores que sin duda me han ayudado a llegar hasta aquí.

Índice

Lista de figuras

Lista de cuadros

Resumen

Este trabajo puede ser de interés para cualquiera que tenga una profesión en tecnología o que tenga influencia en el flujo de trabajo tecnológico (Gerente de producto, Control de calidad, Operaciones de IT y seguridad de la información) así como líderes de mercado y de negocios, donde inicia la demanda de la tecnología. El objetivo principal es implementar diferentes servicios tecnológicos para tercerizar tareas de IT como: administración de máquinas virtuales, monitoreo de recursos y configuración de herramientas de integración continua; para favorecer desarrollos emergentes. Toda la metodología fue basada en los principios y metodologías propuestas por la cultura DevOps. El enfoque va a proyectos emprendedores de MIPYMES en Guatemala afectadas económicamente por las consecuencias causadas por COVID-19 en el año 2020 y así poder recuperar la recesión del PIB según la predicción del Banco de Guatemala para el año 2021 rápidamente.

Se implementaron diferentes mecanismos para que una persona con pocos conocimientos tecnológicos pueda crear máquinas virtuales en la nube bajo demanda. Dichas máquinas poseen pipelines de integración continua y de entrega continua para desarrollos como: landing pages, ecommerce, apis o algún servicio web. Cada tecnología implementada fue seleccionada según las prácticas de metodologías Ágiles y DevOps. Para comprobar la funcionalidad del proyecto se implementaron todas las herramientas para un emprendimiento iniciado el 2020, además se realizaron entrevistas para identificar si existe la posibilidad de implementación para más casos.

CAPÍTULO 1

Introducción

El año 2020 destaca debido a que varias empresas han sido afectadas gravemente por las medidas preventivas por COVID-19 en Guatemala. Esta situación ha obligado a que varios negocios migren los procesos de negocio convencionales a un ambiente digital por medio de implementaciones tecnológicas. Este ambiente se caracteriza por reducir el riesgo de contacto debido a que se previene el contacto físico entre clientes y trabajadores. Esta implementación bajo la situación del año 2020 puede complicarse mucho para MIPYMES en Guatemala las cuales se caracterizan por poseer capital reducido para invertir y más aún cuando los ingresos han sido afectados.

Este proyecto demuestra cómo la cultura de DevOps y las metodologías ágiles ayudan a que la transición de sistemas de información sea eficiente para que emprendimientos pequeños puedan recuperarse de las pérdidas económicas rápidamente. Durante el proceso se reduce la curva de aprendizaje que los sistemas de información que tienen los proveedores en la nube. Como resultado se espera crear un desarrollo de alta calidad que sea de fácil acceso para el usuario y que además cumpla con altos estándares de confiabilidad, integridad y disponibilidad.

Como prueba de concepto se realiza la implementación para un emprendimiento emergente del 2020. Sin embargo, toda la infraestructura puede ser fácilmente modificada para abarcar cualquier caso de negocio emergente.Tomando en cuenta las mejores prácticas de experiencia de usuario se desarrollaron varias interfaces que facilitan a que cualquier persona con pocos conocimientos tecnológicos tenga la capacidad de administrar recursos de proveedores en la nube y a su vez, que estos soporten desarrollos para los procesos internos de una MIPYME.

CAPÍTULO 2

Objetivos

2.1. Objetivo general

Crear un conjunto de servicios basados en la cultura de DevOps que puedan soportar desarrollos tecnológicos emergentes de MIPYMES de Guatemala en el año 2020 como resultado de las medidas preventivas contra COVID-19 para los procesos de negocio de MIPYMES.

2.2. Objetivos específicos

- Realizar prueba de concepto de creación de infraestructura por medio de prácticas DevOps en un proyecto de emprendimiento
- Crear procedimientos automatizados que puedan soportar desarrollos emergentes del 2021 para recuperar las pérdidas económicas del 2020.
- Reducir la curva de aprendizaje de MIPYMES para la implementación de sistemas de información al utilizar metodologías DevOps.

CAPÍTULO 3

Justificación

En Guatemala las microempresas, pequeñas y medianas empresas constituyen un sector de importancia especial para la economía, su aporte en la generación de empleo es de 80 % aproximadamente y su atribución al producto interno bruto de alrededor del 40 % según el Informe de Situación y Evolución del Sector MIPYME de Guatemala 2015-2017 dirigido por el Ministerio de Economía de Guatemala. Adicionalmente, el Ministerio de Trabajo reporta que de los 6,9 millones de personas ocupadas, el 29,8 % trabaja en la formalidad y el 70,2 % restante lo hace en la informalidad, quienes no tienen derecho a un seguro social, ni a prestaciones laborales.

En el año 2020 se ha realizado un cambio radical para la vida de las personas. COVID-19 es un virus que ha afectado a toda la humanidad. El gobierno ha impuesto ciertas medidas preventivas para reducir la propagación de la enfermedad. Como resultado las personas deben de permanecer en casa por lo que, las empresas se vieron en la obligación de implementar soluciones tecnológicas para poder continuar con los labores empresariales. Este comportamiento debe ser realizado en un corto periodo de tiempo y tomando en cuenta las restricciones que se presentan. Esto representa un reto para muchas empresas debido a la complejidad de los sistemas. Esto no puede detener a las empresas, a su vez, varios emprendimientos han encontrado oportunidades de negocio sobre estas condiciones.

Este proyecto implementa la metodología de desarrollo y operaciones (Devops) para facilitar la interacción del recurso humano con los sistemas de información. Esta metodología se basa en controlar el riesgo, reducir esfuerzos y reducir costos en implementaciones de servicios tecnológicos para desarrollos que se encuentran en un ambiente que cambia rápidamente. Se realizará una prueba de concepto con herramientas que sean fácilmente trasladadas a casos de negocios de Guatemala. Se busca crear una estrategia para facilitar la implementación de tecnologías recientes para microempresas, pequeñas y medianas empresas. Con ello la caída del producto interno bruto del 3.6 % para el 2020, según las predicciones del Banco de Guatemala, podrá recuperarse para el año 2021 rápidamente. Dándole la oportunidad a que las empresas MIPYMES en Guatemala puedan retomar sus operaciones laborales de una manera actualizada y estable.

CAPÍTULO 4

Marco teórico

4.1. Recesión Guatemala 2020

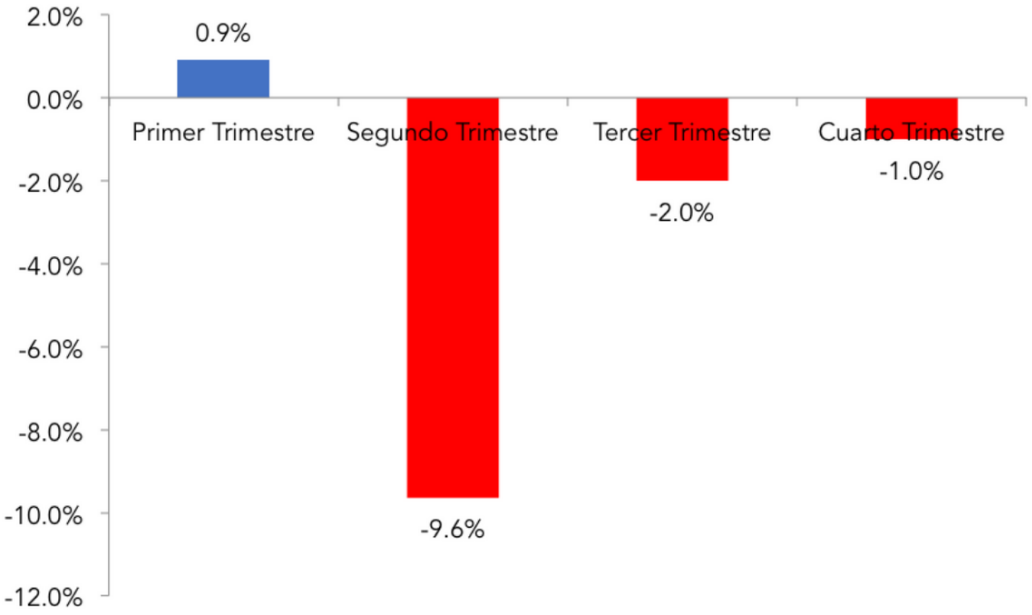
El 9 de enero de 2020 Banguat estimaba un crecimiento del producto interno bruto del 3.6 %. Luego para finales de marzo estimaba que la economía crecería solo un 1 % para todo el año 2020. La pandemia causada por COVID-19, ha perturbado en el mundo a miles de millones de vidas y medios de subsistencia, amenaza los avances en materia de desarrollo alcanzados. La economía mundial podría contraerse desde un 5 % a un 8 % en 2020 antes de recuperarse en 2021. Se espera que la economía guatemalteca, considerablemente afectada por la pandemia, se contraiga un 3.5 % en 2020. Tomando en cuenta las vulnerabilidades existentes en el país. (Banco Mundial, 2020)

- Las empresas en particular las MIPYMES, se han visto afectadas significativamente debido a las medidas de cierre, la caída de la demanda, cuellos de botella en cadenas de suministro y la disminución de liquidez a medida que los ingresos colapsan.
- Se espera que la pérdida de empleos sea mayor en los sectores de construcción, servicios, transporte y comercio, donde se concentra la mayor parte de la fuerza laboral vulnerable.
- Los ingresos de los trabajadores por cuenta propia, los trabajadores temporales y los trabajadores en sectores afectados(turismo, entretenimiento) caerán.

(Dirkmaat, Fernandez, 2020)

La recesión en los Estados Unidos afectará las exportaciones y las remesas, las dos principales fuentes de ingresos en divisas y agravará la desaceleración del consumo privado.(Dirkmaat ,Fernandez, 2020)

Figura 4.1: Pronóstico de PIB del año, según Banguat y proyecciones de Fundación Libertad y Desarrollo



(Boteo, 2020)

Se prevé que aproximadamente un millón de personas caigan en pobreza, aumentando la tasa de pobreza del país hasta un 6 %, dependiendo de la profundidad y duración de la crisis, así como la velocidad de la recuperación económica. (Guatemala Panorama general, 2020)

4.2. Micro empresa

Según el Ministerio de Economía de Guatemala, la microempresa, pequeña y mediana empresa en Latinoamérica representa el 99 % del tejido industrial, generando así la mayoría de los empleos. En Guatemala las micros, pequeñas y medianas empresas constituyen un sector de importancia especial, su aporte en la generación de empleo es de alrededor del 80 %, su contribución al producto interno bruto de alrededor del 40 % y su papel en el incremento de la competitividad de los territorios representan una gran oportunidad de desarrollo económico y humano. (MINECO, 2017)

La MIPYME en Guatemala se ha caracterizado por tener importantes brechas que cerrar para la generación de oportunidades como la alta incidencia de informalidad, los procesos de producción básica y las limitadas oportunidades de acceso a crédito por mencionar algunos ejemplos. Pero aún más importante, la Mipyme se considera por ser un espacio de oportunidad de empleo y autoempleo, un mecanismo para generar más desarrollo económico incluyente y sostenible. (MINECO, 2017)

Es relevante establecer a qué empresas comprenden el sector MIPYME. De acuerdo con el Acuerdo Gubernativo 211-2015 de fecha 21 de septiembre de 2015 el tamaño de las empresas tendrá como variables el número de trabajadores y las ventas anuales expresadas en salarios mínimos de actividades no agrícolas. (MINECO, 2017)

Tabla 4.1: Lineamientos para consideración de empresas como microempresas, pequeñas y medianas empresas

Tamaño de la empresa	Número de empleados	Ventas anuales en salarios mínimos mensuales de actividades no agrícolas
Micro	1-10	1-190
Pequeña	11-80	191-3,700
Mediana	81-200	3,701-15,420

(MINECO, 2017)

Durante el periodo comprendido entre 2015-2017 el número de empresas activas totales en Guatemala aumentó pasado de 372,779 en 2015 a 481,570 en 2017. Los datos referentes a esta evolución se presentan a continuación. (MINECO, 2017)

Tabla 4.2: Composición del parque activo empresarial guatemalteco por tamaño de empresa 2015-2017

Tamaño de la empresa	2015		2016		2017	
	%	#	%	#	%	#
Microempresas	88.73	330,750	90.17	393,730	90.34	425,043
Pequeñas	9.76	36,398	8.55	37,352	8.42	40,568
Medianas	1.08	4,027	0.92	3,999	0.90	4,328
Grandes	0.43	1,602	0.36	1,591	0.34	1,631
Total empresas	372,779		436,672		481,570	

(MINECO, 2017)

Esto representó un aumento interanual promedio de 14 %. El aumento más importante se dio en las microempresas donde existe un crecimiento de 108,791 empresas en los dos años de observación. Esto a su vez tuvo como resultado que el porcentaje de empresas en Guatemala catalogadas como microempresas aumentara pasando de representar el 88.73 % en el año 2015 a representar el 90.34 % en el año 2017. Es importante notar que dado el nivel de concentración que existe en esta categoría de empresas, los aumentos porcentuales son por su naturaleza limitados. Por otro lado, las pequeñas y medianas empresas evidenciaron niveles de crecimiento por debajo del promedio global observado durante el período comprendido entre 2015-2017. (MINECO, 2017)

El porcentaje de microempresarios dentro del universo total de empresas se ha incrementado y en términos demográficos, una proporción cada vez mayor de la PEA (Población Económicamente Activa) se desempeña en este tipo de empresas, por lo que es importante comprender y asistir a estas empresas a desarrollarse para poder coadyuvar al desarrollo económico nacional. (MINECO, 2017)

4.3. ¿Por qué las empresas buscan servicios de tecnología?

Technology-as-a-service playbook, afirma que modelos de negocio que utilizan servicios como Software como Servicios (SaaS) y servicios administrados son muy rentables. En el año 2013 23 % realizaron un estudio donde 23 % de las compañías manejan un flujo ingresos basados en Servicios Administrados. A finales de 2015, 46 % de las compañías reportaron flujos de ingresos por Sistemas Administrados(MS). Existen ciertas tendencias que provocan la demanda de MS:

- Reducen la complejidad operacional: Los clientes ya no quieren manejar operaciones de IT, especialmente si no miran estas como una competencia clave del negocio

- Capacidad bajo demanda: Clientes ya no quieren pagar por la capacidad de un equipo de IT que no necesitan. Modelos MS son mucho más flexibles que permiten al cliente comprar la capacidad que necesitan
- Gastos operativos(OpEx) contra gastos de capital(CapEx): Algunos clientes prefieren ver la tecnología como un gasto de operación en lugar de invertir capital. CapEx representa pagos de grandes cantidades y luego comprar activos que deben ser depreciados durante el tiempo. Al migrar los gastos de IT a OpEx normalmente se maneja de manera fluida y con costos más predecibles durante el tiempo.
- Valor sobre tecnología: Clientes miran a los proveedores de la tecnología para aplicar hallazgos únicos que ayuden a maximizar el impacto de la tecnología en su negocio. Una proposición de valor que ofrece MS es la capacidad acelerada de adoptar tecnología en lugar de hacerla por sus propios medios.
- Economías de escala: Proveedores de tecnología pueden crear ambientes, herramientas y procesos que soportan múltiples clientes. Estas economías permiten al proveedor entregar ambientes de tecnología con costos efectivos.
- Estrategia contra táctica: la revista CIO publicó un artículo citando el incremento en la demanda de servicios administrados. La revista reportó el interés de las compañías por tercerizar sus operaciones de IT para poder enfocar los esfuerzos en sus procesos de negocio internos.

(Wood, Lah, 2016)

4.4. Modelos de desarrollo de software

Los modelos de desarrollo de software son conformados por varias metodologías y procedimientos, seleccionados para desarrollar un proyecto acorde a un propósito y objetivos específicos. Existen varios modelos para desarrollar software sin embargo a pesar de que cada uno posee diferentes características tienen cierta base común. Según el estándar internacional para SDLC(ciclos de vida de desarrollo de software), ISO/IEC 12207 define las fases comunes de un modelo de desarrollo para software. (Stoica, M. 2013)

Figura 4.2: Fases generales del ciclo de vida para desarrollo de software

(Boteo, 2020)

- Fase 1: Análisis de requerimientos y planificación:

  Es la fase más importante para un SDLC. En esta miembros del equipo de desarrollo obtienen datos de clientes, departamentos de ventas, investigaciones de mercado y expertos en industrias. Esta información es utilizada para la realización de un plan y un estudio factibilidad según el lado económico, operacional y técnico. Además, en este paso se definen el control de calidad y se identifican los riesgos del proyecto. Este esfuerzo resulta en plantear un acercamiento para implementar el proyecto con el mínimo riesgo posible.

- Fase 2: Definición de requerimientos:

  Luego de que los requerimientos son analizados y el producto ha sido claramente definido y documentado. El cliente o el mercado deben de aprobarlo por medio del SRS(Especificación de requerimientos de software).

- Fase 3: Diseño de arquitectura del proyecto:

  Tomando en cuenta la lista de SRS uno o varios arquitectos crean la mejor arquitectura del producto. Por lo menos una propuesta debe de ser realizada y documentada en un DDS(Documento de diseño de especificaciones). Esta propuesta es validada por todos los interesados basado en parámetros como: evaluación de riesgo, robustez del producto, método de diseño, presupuesto y tiempo.

- Fase 4: Implementación del producto o desarrollo:

  Esta fase se basa en la implementación de código fuente. Si el diseño fue realizado de una manera detallada y organizada este paso puede realizarse sin complicaciones. Los desarrolladores deben de seguir las guías de sus organizaciones para la creación del producto.

- Fase 5: pruebas del producto:

  Esta fase es una subvención en todas las demás fases debido a que puede estar relacionada con todas las demás fases del ciclo de vida de desarrollo de software. Esta toma en cuenta la situación donde son reportadas, rastreadas, arregladas y analistas las fallas del producto hasta que cumpla con los requisitos de calidad del SRS.

- Fase 6: operaciones y mantenimiento

  Una vez el producto haya sido probado, está listo para salir al mercado. Puede ser lanzado a un público limitado para que el producto sea probado en un escenario real del ambiente laboral y después basado en la retroalimentación recibida puede ser lanzado a todo el mercado tomando en cuenta los cambios de los clientes. Luego de ello es necesario realizar mantenimiento para que los clientes puedan seguir utilizándolo.

4.5. Metodologías tradicionales

Los ciclos de desarrollo de software tradicionales están basados en un acercamiento productivo. Los equipos trabajan con un plan detallado y una lista de todas las características que deben ser completadas para el ciclo del producto. Estos complementos dependen del análisis de requerimientos y planificación al inicio del ciclo. Sin embargo, cada cambio debe ser incluido por medio de un proceso estricto de manejo de control de cambios. Adicionalmente los métodos tradicionales al inicio se enfocan en documentación de orientación y clarificación del proyecto para el equipo de desarrollo. Esto facilita que hayan malos entendimientos durante el desarrollo del proyecto. (Toica, M. 2013)

Estas son algunas ventajas de modelos tradicionales:

- La documentación y la estructura es de beneficio para los nuevos integrantes.
- Fácil de entender y usar.
- Fácil de coordinar debido a que es esperado los resultados en cada fase del proceso.
- Las fases son implementadas una a la vez.

Desventajas de estos modelos:

- Algunos requerientes surgen después de iniciado el proyecto por lo que impactó negativamente el desarrollo.
- Algunos requerimientos adicionales por el cliente puede resultar en mayores costos.
- No todos los problemas son solucionados en cada fase.
- No existe prototipo hasta el final del ciclo.
- Existe un gran riesgo y una gran inseguridad.

¿Cuándo implementar metodologías tradicionales?

- Los requerimientos están bien claros y definidos.
- La definición del producto es estable.
- La tecnología es comprendida.

- No hay requerimientos ambiguos.

Incrementar la agilidad de las operaciones internas es un comportamiento que buscan todas las organizaciones. Esto permite una rápida y eficiente adaptación a los cambios del mercado para ganar una mejor ventaja estratégica. Este comportamiento depende de la disminución de tiempo para nuevos procesos de desarrollo y también un incremento de flexibilidad a los procesos existentes donde la modificación e implementación es necesaria. Todo esto resulta en reducción de tiempo para suplir las demandas de los clientes, manejar mayor cantidad de clientes y disminuir los costos de adaptación que resulta en más ganancias. Actualmente este comportamiento no es una característica en las empresas sino una condición para acceder y permanecer en el mercado. Este comportamiento es algo que se puede obtener con cualquier metodología, sin embargo las metodologías ágiles se crean en base a estos principios. (Stoica, M. 2013)

4.6. Metodologías ágiles

En un modelo incremental los requerimientos son divididos en varias interacciones. Este método involucra múltiples ciclos de desarrollo lo cual puede verse como un modelo de múltiples cascadas. Cada ciclo es dividido en varios ciclos pequeños para que puedan ser manejados fácilmente. Sin embargo, cada uno de los módulos pasa por análisis de requerimientos, diseño, implementación y pruebas. Se busca que desde la primera iteración se cree una primera versión del software. Cada integración adicional busca agregar nuevas funcionalidades y características al ciclo anterior. Este proceso se repite hasta que el sistema es completado. (Stoica, M. 2013)

Estas son algunas ventajas de modelos tradicionales:

- Se entrega un producto funcional al cliente en cada ciclo.
- La retroalimentación del cliente es distribuida durante el proceso de desarrollo.
- Es más flexible, el cambio de requerimientos requiere menos esfuerzo comparado con un modelo tradicional.
- En una integración pequeña es fácil de probar.
- Es más fácil administrar el riesgo debido a que los riesgos son identificados y manejados en cada iteración.
- Nuevos requerimientos pueden ser integrados constantemente.

Desventajas de estos modelos:

- Necesita un claro y completo entendimiento de la definición del sistema antes que pueda ser dividido en múltiples ciclos.
- El costo total tiende a ser mayor que el modelo tradicional.
- Este acercamiento puede resultar en “desarrollar y arreglar”.
- El cliente puede ver lo que se ha hecho e incrementar los requerimientos.

¿Cuándo implementar metodologías tradicionales?

- Los requerimientos son finales y algunos detalles pueden cambiar durante el desarrollo.

- Es necesario un prototipo funcional lo antes posible.
- Se utiliza una nueva tecnología.
- Existe una gran cantidad de riesgos y objetivos.

4.7. Comparación metodologías ágiles y tradicionales

Los métodos ágiles se basan en métodos de desarrollo de software que buscan adaptación, mientras que los modelos tradicionales están basados en un acercamiento productivo. En un modelo SDLC tradicional, los equipos deben de realizar un plan detallado, lleno de características y tareas que deben ser completadas durante todo el ciclo de vida de desarrollo del producto. Los métodos predictivos dependen completamente en el análisis de requerimiento y planificación al inicio de cada ciclo. Cualquier cambio que debe ser incluido debe de ser aprobado luego de un proceso de administración de control y priorización. La metodología ágil utiliza un acercamiento adaptativo donde no existe una planificación detallada, donde las tareas futuras se basan únicamente en las características que deben desarrollarse. El equipo se adapta a constantes cambios en los requerimientos de producto. El producto está en constantes escenarios de prueba para minimizar el riesgo de problemas mayores en el futuro. La iteración con clientes es el punto fuerte de la metodología ágil pero algunas tendencias son manejar documentación mínima. (Stoica, M. 2013)

Las SDLC ágiles son mejor aprovechadas por proyectos pequeños y medianos, mientras que las tradicionales se aplican a gran escala. Por lo tanto, es importante que el equipo de desarrollo seleccione la metodología que mejor se acople a las características del proyecto. Algunos criterios que se pueden utilizar para identificar la dimensión del proyecto pueden ser: tamaño del equipo, locación geográfica, tamaño y complejidad del software, tipo de proyecto, estrategia de negocio y capacidades de ingeniería. Además es importante que el equipo tenga claro las ventajas y desventajas de cada SDLC antes de tomar una decisión. El equipo también debe de estudiar el contexto del negocio, requerimientos de la industria y estrategia de negocio para poder proponer una SDLC. Es importante esta evaluación previa para maximizar las posibilidades de crear un software exitoso. (Stoica, M. 2013)

Tabla 4.3: Comparación entre metodologías ágiles y tradicionales

	Desarrollo tradicional	Desarrollo ágil
Hipótesis fundamental	Los sistemas están completamente especificados, predecibles y son desarrollados por un plan detallado y extendido.	Un software adaptativo de alta calidad es desarrollado por equipos pequeños que utilizan el principio de mejora continua del diseño y pruebas basadas en retroalimentación rápida y cambio.
Estilo de administración	Controlar y comandar.	Liderazgo y colaboración.
Manejo de conocimiento	Explícita.	Tácita.
Comunicación	Formal.	Informal.
Modelos de desarrollo	Modelo de ciclo de vida (cascada, espiral).	Modelo de entrega evolucional.
Estructura organizacional	Mecánica (burocrática, muy formal) apunta a organizaciones grandes.	Orgánica (flexible, participativa y promueve la participación social) apunta a pequeñas y medianas organizaciones.
Control de calidad	Planificación estratégica y estricto control. Pruebas complejas y tardía.	Control permanente o requerimientos, diseño y soluciones. Pruebas permanentes.
Requerimiento de usuarios	Detallada y definida antes de la implementación.	Constante aportación.
Costo de reiniciar	Alto.	Bajo.
Dirección de desarrollo	Rígido.	Fácil de cambiar.
Pruebas	Luego de que el código es completado.	Cada iteración.
Participación del cliente	Poca.	Mucha.
Habilidades adicionales requeridas por los desarrolladores	Nada en particular.	Habilidades interpersonales y básico conocimiento del negocio.
Escala apropiada del proyecto	Grande escala.	Pequeña y mediana escala.
Desarrolladores	Orientación al plan con habilidades adecuadas y acceso a conocimiento externo.	Ágil, con conocimiento avanzado y cooperativo.
Clientes	Con acceso al conocimiento, cooperativo, representativo y empoderado.	Dedicado, experto, cooperativo, representativo y empoderado.
Requerimientos	Estables y conocidos con antelación.	Emergentes y cambios rápidos.
Arquitectura	Diseño actual y requerimientos predecibles.	Diseño para requerimientos actuales.
Objetivos primarios	Alta seguridad.	Valor rápido.

4.8. Ejemplos de métodos ágiles

Existen varios métodos ágiles de desarrollo de software. Cada uno con su propio acercamiento pero tomando en cuenta los valores principios del manifiesto ágil. Todos mencionan comunicación permanente, planificación, pruebas e integración. Estos ayudan a desarrollar software, pero lo que los define como ágiles es la característica que les permite tomar buenas decisiones rápidamente. A continuación se mencionan varios métodos ágiles:

Tabla 4.4: Descripción sintetizada de ejemplos de metodologías ágiles.

Metodo agil	Descripción
Metodologías Crystal	Es una familia de métodos para equipos de varios tamaños: clear, yellow, orange, red, blue. El método más ágil, Crystal Clear, se concentra en la comunicación entre pequeños equipos que realizan software no crítico. El desarrollo tiene siete características: entrega constante, mejora reflexiva, comunicación osmótica, seguridad personal, enfoque, fácil acceso a usuarios expertos y requerimientos para un ambiente técnico
Método de desarrollo dinámico de software(DSDM)	Divide proyectos en tres fases: antes del proyecto, ciclo de vida del proyecto y luego del proyecto. DSDM está basado en nueve principios: intervención de usuarios, empoderamiento del equipo de proyecto, entrega continua, acercamiento a las necesidades del negocio, iterativo y de desarrollo incremental, permite revertir cambios, metas a largo plazo son establecidas antes de que inicie el proyecto, pruebas durante el ciclo de vida y comunicación eficiente.
Feature-driven development	Combina un modelo dirigido a desarrollo con una metodología ágil, se enfoca en el desarrollo de un modelo inicial lo antes posible, el trabajo se enfoca en características del producto y en diseño iterativo de cada característica. Afirma ser la mejor metodología para el desarrollo de software crítico. Cada iteración tiene dos fases: diseño y desarrollo
Scrum	Se enfoca en la administración de proyectos donde la planificación es difícil. Utiliza un mecanismo para "control de procesos empíricos" donde los ciclos de retroalimentación son el elemento principal. El software es desarrollado por un equipo (que se organiza por sí mismo) en fases(sprints) que empiezan con planificación y terminan en evaluación. Las características que deben de ser implementadas son registradas en una lista de órdenes pendientes. El cliente decide cual de esas órdenes deben ser implementadas en el siguiente sprint. Miembros del equipo coordinan la actividad de reuniones diarias. Donde un miembro es responsable de resolver problemas que interfieren con el trabajo del equipo.
Extreme programming(XP)	Se concentra en las mejores prácticas de desarrollo y consiste en doce actividades: planificar el juego, pequeños lanzamientos, metáforas, planificación simple, pruebas, refactorización, programación cooperativa, propiedad colectiva, integración continua, 40 horas a la semana, clientes en sitio y estándares de codificación.
	La versión revisada , XP2, consiste en las siguientes prácticas principales: involucrar a todo el equipo, espacio de trabajo informativo, trabajar bajo presión, historias de programación cooperativa, ciclos semanales, ciclos trimestrales, desarrollo de minuto 10, integración continua, diseño incremental y 11 prácticas adicionales a XP.

4.9. Manifiesto Ágil

El Manifiesto Ágil es un documento que identifica cuatro valores clave y doce principios que los autores creen que los desarrolladores de software deben de utilizar para su trabajo. Formalmente lo llaman Manifiesto para desarrollo ágil de software. Fue creado por 17 desarrolladores del 11 al 13 de Febrero en el año 2001 en la estación de ski The Lodge at SnowBird en Utah.

Los desarrolladores que crean el manifiesto se llaman a sí mismos como la Alianza Ágil. Ellos estaban buscando una alternativa a los procesos de desarrollo de software existentes los cuales ellos mismos los describen como complicados, no adaptativos y muy enfocados en documentación de requerimientos. Este grupo de personas expresan su deseo para encontrar un balance entre las maneras de desarrollo existentes y las nuevas alternativas. Ellos aceptan modelar y documentar pero solo si tiene un beneficio claro.

Cuatro valores:

- Individuos e interacciones sobre procedimientos y herramientas.
- Trabajar software sobre una documentación exhaustiva.
- Colaboración del cliente sobre negociaciones por contrato.
- Responder al cambio sobre seguir el plan.

Doce principios:

- Bienvenidos son los cambios de requerimientos, incluso en desarrollo avanzado. Procesos ágiles aprovechan el cambio para que el cliente pueda tener una ventaja competitiva.
- Entrega continua de software funcional frecuentemente, desde un par de semanas a un par de meses, con preferencia en una escala de tiempo reducida.
- Personas de negocio y desarrolladores deben de trabajar juntos diariamente durante el proyecto.
- Crear proyectos sobre individuos motivados. Darles el ambiente y apoyo que necesitan y confiar en ellos para hacer el trabajo.
- El método más eficiente y efectivo de transmitir información hay entre el equipo de desarrollo son las conversaciones cara a cara.
- El software funcional es una medición primaria de progreso
- Los procesos ágiles promueven el desarrollo sustentable. Los patrocinadores, desarrolladores y usuarios deben de poder mantener un paso constante indefinido.
- Atención continua a excelencia técnica y buenos diseños promueven agilidad.
- Simplicidad, el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requerimientos y diseños emergen de equipos autoorganizados.
- En intervalos regulares, el equipo refleja en como pueden ser más efectivos luego se afinan y ajustan su comportamiento acorde.

(agile manifesto, 2001)

4.10. Desarrollo y operaciones (DevOps)

En la mayoría de organizaciones el departamento de IT tiene un constante conflicto entre desarrollo y operaciones de tecnologías de información lo que aumenta el tiempo para sacar nuevos productos y características al mercado, reduce su calidad y aumenta costos. Para evitar esta situación el departamento de IT debe de responder rápidamente a ambientes competitivos cambiantes y proveer un servicio al cliente estable, confiable y seguro. En caso contrario, el equipo tecnológico termina en una situación que que resulta en mala calidad de servicio y software, además en malos resultados al cliente y también se ve reflejado en reuniones para solucionar problemas.

El libro *The DevOps HandBook how to Create World-Class Agility, Reliability, and Security in Technology Organizations* define tres secciones que resumen las áreas donde pueden haber problemas. La primera se refiere a que varios de los problemas son debido a que las aplicaciones e infraestructura son muy complejas, mal documentadas y muy frágiles. El equipo técnico está en una deuda permanente de tiempo donde cuando haya tiempo solucionará los problemas pero nunca llegará esta situación. El segundo escenario se debe a que debido a fallas del sistema a los clientes los equipos comerciales tratan de compensarlo por medio de promesas que aumentan la funcionalidad de los productos. El problema de ello es que estas nuevas características no son planificadas correctamente y son de alta urgencia afectando la producción. Y por último se refiere a que cada vez el conjunto de todos los sistemas es más complejo de manejar. Cada miembro del equipo está más ocupado, la comunicación es menos, el trabajo lleva más tiempo, y las colas de trabajo tienden a ser más largas. (Kim, Humble, Debois, Willis, 2016)

The DevOps Handbook plantea tres principios que involucran prácticas de DevOps para evitar los escenarios negativos mencionados anteriormente. Desde 2013 al 2016 se recolectó información de más de 25 mil profesionales de tecnologías, con el objetivo de mejorar el entendimiento de los hábitos de las organizaciones en todos los pasos en la adopción de DevOps. Entre lo destacado se encuentra:(Kim, Humble, Debois, Willis, 2016)

- Métricas de rendimiento.
- Cambios de producción treinta veces más frecuente.
- Tiempo de espera de cambios de producción doscientas veces más rápido.
- Métricas de confiabilidad.
- Seis veces más probable de tener éxito en desarrollos de producción.
- Tiempo promedio para recuperar un servicio 168 veces más rápido.
- Métricas del desempeño de la empresa.
- Crecimiento en la capitalización de mercado 50 % arriba durante tres años.
- Productividad y metas rentables dos veces más probables de tener éxito.

Adicionalmente en 2015 el reporte del estado de DevOps describe la métrica de no solo despliegues por día si nó despliegues por día por desarrollador. Empresas como Google, Amazon y Netflix han logrado aumentar la cantidad de despliegues según el aumento de la cantidad de trabajadores. Estas métricas afectan la administración de producto, desarrollo, gestión de calidad, operaciones de IT, seguridad de la información, una ventaja en el mercado(Kim, Humble, Debois, Willis, 2016)

DevOps es el resultado de prácticas técnicas, arquitectónicas y culturales representan la convergencia de varios movimientos filosóficos y administrativos. Hay décadas de lecciones

aprendidas en: procesos de manufacturación, organizaciones de alta confianza, manejo de modelos de alta confianza y otros que han moldeado las prácticas de DevOps a lo que se conoce actualmente. DevOps resulta de conocimientos de la teoría de restricciones, el sistema de producción de Toyota, ingeniería de residencia, organizaciones de aprendizaje, cultura segura, factores humanos, y otros. El resultado es calidad, confiabilidad, estabilidad y seguridad a bajo costo y esfuerzo. Esto ha acelerado el proceso y aumentado la confiabilidad de la corriente de tecnología que incluye administración de producto, desarrollo, control de calidad, operaciones de IT.(Kim, Humble, Debois, Willis, 2016)

Principio del flujo:

Este principio establece entregarle valor a los clientes rápidamente. Esto inicia por crear un flujo visible, reducir el tiempo de transiciones y los intervalos de trabajo, aumentar la calidad y prevenir que los defectos sean trasladados al cliente. Esto también incluye reducir el tiempo interno y externo de atender las solicitudes de los clientes. Adicionalmente es necesario identificar y analizar las restricciones que tienen los sistemas. Por último una de las principales amenazas al negocio es el desperdicio. Esto puede ser catalogado en las siguientes categorías:

- Trabajo parcialmente realizado: esto incluye trabajo en la corriente de valor que no ha sido completado, este pierde valor mientras el tiempo pasa.
- Procesos adicionales: cualquier trabajo adicional que está siendo realizado que no agrega valor al cliente . Esto incluye documentación que no se utiliza en el día a día, aprobaciones que no agregan valor al resultado final. Estos procesos extra agregan esfuerzo e incrementa los tiempos de transición.
- Funcionalidades adicionales: Funcionalidades que no son necesitadas por el negocio o por el cliente. Estas agregan complejidad y esfuerzo para probar y administrar el desarrollo.
- Cambiar de tareas: Cuando las personas tienen muchos proyectos asignados, esto requiere cambiar de contexto para asimilar las dependencias de cada trabajo. Esto representa esfuerzo y tiempo adicional a la corriente de valor.
- Espera: Cualquier retraso en el trabajo por recursos para completar su trabajo. Los retrasos aumentan el ciclo de vida de los proyectos y previene al usuario de obtener valor.
- Movimiento: Es necesario cierta cantidad de esfuerzo para mover información o material de un centro de trabajo a otro. El desperdicio de movimiento es creado cuando personas que deben de comunicarse no están relacionadas.
- Defectos: Información incorrecta, faltante o no clara, materiales, productos crean desperdicio debido a que representa un esfuerzo resolver estos problemas. Mientras mayor sea el tiempo entre la creación del defecto y la detección del defecto es más difícil resolver el defecto
- Trabajo manual o no estandarizado: La dependencia en trabajo no estandarizado como no utilizar ambientes de prueba y configuraciones correctamente. Idealmente cualquier dependencia en operaciones debe ser automatizada, auto sostenible y disponible bajo demanda.
- Heroicas: Para que la organización pueda completar sus metas individuos y equipos deben de realizar actos no razonables lo cual puede volverse parte de su vida laboral diaria.

(Kim, Humble, Debois, Willis, 2016)

Principio de retroalimentación:

El objetivo es crear un sistema de trabajo más seguro y resiliente. Esta sección se refiere a los principios para recibir retroalimentación rápidamente y constantemente en todas las fases de la corriente de valor. Esto permite detectar y arreglar problemas cuando son pequeños y fáciles de arreglar. Los sistemas tecnológicos pueden tender a ser definidos como complejos debido al alto grado de interconectividad de múltiples componentes con múltiples comportamientos. Es aquí donde los fallos son inevitables por lo que es necesario diseñar un sistema de trabajo seguro donde el trabajo pueda realizarse sin miedo, cualquier error debe ser detectado rápidamente antes de que ocasione resultados catastróficos. Dr. Steven Spear analiza el caso del sistema de productos Toyota y propone cuatro condiciones para crear un sistema de trabajo seguro para sistemas complejos:

- Para trabajos complejos se busca revelar problemas en el diseño y las operaciones del mismo para que se genere conocimiento que luego arregle estos problemas.
- El conocimiento local es aprovechado globalmente por toda la organización.
- Líderes pueden crear más líderes con las mismas capacidades.

En un sistema seguro de trabajo es necesario probar el diseño con sus respectivas suposiciones previas al desarrollo. Un ambiente de alto rendimiento que tiene un flujo de información rápido, frecuente y de alta calidad debe medir y monitorear cada operación para que cualquier defecto sea rápidamente identificado y corregido. Un comportamiento de esta manera permite una mejora de calidad, seguridad y de aprendizaje continuo. En contraste, cuando la retroalimentación es poco frecuente o muy lenta pueden haber resultados no deseados. Pero para que exista retroalimentación es necesario que haya una metodología para buscar problemas debido a las siguientes razones:

- Previene que los problemas sean transmitidos a la corriente de valor donde el esfuerzo y costo puede incrementar exponencialmente.
- Previene que el equipo de trabajo realice esfuerzos adicionales, los cuales pueden producir nuevos errores en el sistema.
- Si el problema no es identificado el equipo de trabajo puede repetir la falla en operaciones futuras, representado el doble de trabajo para arreglarlo.

Sin embargo en sistemas complejos existen casos donde se aumenta la posibilidad de fallos al aumentar la cantidad de pasos de inspección y aprobación. Por ello es necesario que todo el equipo esté coordinado de manera estratégica. Algunos ejemplos de controles de calidad no efectivos:

- Necesitar de otro equipo que complete tareas manuales, esto puede producir mayor cantidad de errores. En lugar de automatizar procesos según el equipo lo necesite.
- Necesitar aprobaciones de personas muy ocupadas o que están distantes del trabajo, forzándola a tomar decisiones sin el conocimiento necesario o de las implicaciones potenciales que pueden tener sus decisiones.
- Crear grandes volúmenes de documentación que terminan siendo obsoletas luego de que se escriben.
- Manejar grandes cantidades de tareas para que pueda ser aprobado y procesado un trabajo puede producir tiempo de espera en los resultados.

(Kim, Humble, Debois, Willis, 2016)

Principio de aprendizaje continuo y experimentación:

En los flujos de valor de tecnología se busca tener una cultura de confianza, tomando en cuenta que siempre estamos en constante aprendizaje y debemos tomar riesgos en nuestro trabajo diario. Se busca aprender de nuestros fallos y éxitos mejorando las ideas que no funcionan y completando las que sí. Este conocimiento personal se puede volver rápidamente una mejora global para desarrollar nuevas técnicas que pueden ser utilizadas por toda la organización. Cuando se trabaja con sistemas complejos por definición es imposible predecir todos los resultados de las acciones que tomamos. Esto contribuye a resultados inesperados o catastróficos sin importar que hayamos tomado precauciones para evitarlo. Cuando estos resultados afectan a nuestros clientes probablemente sea por algo que haya realizado un recurso humano. La administración convencional señala al recurso culpable y probablemente lo castiga. Esto puede guiar a una cultura de miedo donde los problemas no son reportados. El Dr. Ron Estrum fue uno de los primeros en observar la importancia de una cultura organizacional segura. Por medio de la observación de organizaciones de cuidados de salud pudo encontrar patrones para diferenciar tres tipos de culturas

- Organizaciones patológicas caracterizadas por miedo y amenazas. Las personas guardan información o prefieren distorsionarla para dar una mejor apariencia. Normalmente los errores se esconden.
- Organizaciones burocráticas caracterizadas por reglas y procesos. Las fallas son procesadas por un sistema de juicio y según el resultado del proceso se crean castigos, justicia o piedad.
- Las organizaciones generativas están caracterizadas por constantemente buscar y compartir información para mejorar la organización. La responsabilidad es compartida durante el flujo de valor y las fallas son estudiadas.

El flujo de valor tecnológico debería regirse de una cultura generativa para crear sistemas de trabajo seguros. Cuando sucedan accidentes en lugar de buscar al culpable se debería de buscar cómo rediseñar el sistema para prevenir que ese accidente vuelva a suceder. Existen equipos que no buscan implementar mejoras para los procesos que implementan. Esto resulta en que continúan sufriendo de los problema actuales y que cada vez los problemas se vuelven más difíciles de resolver. En el área tecnológica cuando se busca una solución verdadera a los problemas estos pueden acumularse hasta que exista el caso donde las tareas únicamente sean resolver problemas o resolver desastres. En contraste se busca que los equipos arreglen problemas en sus tareas diarias en cualquier momento como parte de su trabajo diario. Como equipo pueden resolver los problemas fácilmente y a bajo costo con consecuencias pequeñas. Por otro lado, cuando se descubre nuevo conocimiento debe haber algún mecanismo en las organizaciones para que ese conocimiento sea de beneficio para todos. Por ejemplo, se puede crear reportes accesibles para todos los equipos que resuman las soluciones que se hayan encontrado para que sea fácilmente utilizado por las organizaciones. Otra práctica que ayuda a que el conocimiento de los procesos pueda mejorar a favor de la organización es implementar prácticas bajo control que pongan a prueba los límites de los sistemas. Esta tensión puede ayudar a buscar nuevos diseños que tomen en cuenta casos extremos o que puedan soportar más carga de trabajo. Tradicionalmente existen líderes de proyecto los cuales son responsables por plantear objetivos en combinación con incentivos. Ellos deben elevar el valor del aprendizaje para resolver los problemas. El personal puede ser motivado por medio de preguntas como:

- ¿Qué has aprendido?
- ¿Qué es lo que está pasando ahora?
- ¿A dónde quieres llegar?
- ¿Qué problema tienes ahora?

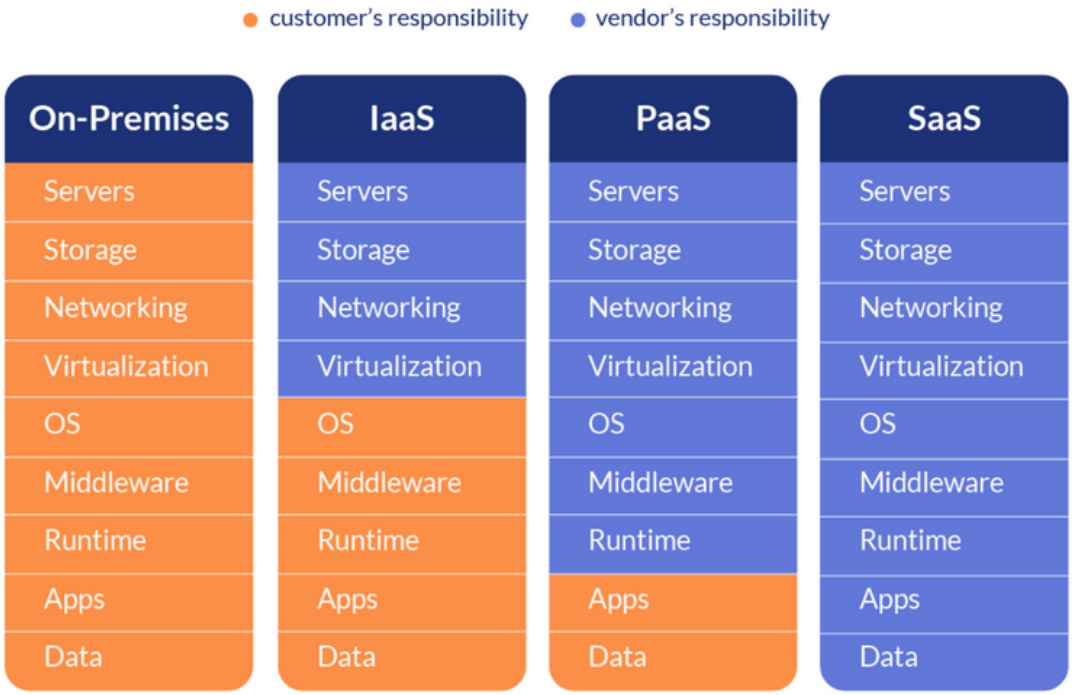
- ¿Qué esperas?

(Kim, Humble, Debois, Willis, 2016)

4.11. As a Service

La nube es un concepto que absorbe diferentes tipos de servicios en línea. Para los que consideran utilizar servicios de nube es necesario diferenciar entre IaaS, Paas y SaaS. Los acrónimos -aaS son confusos, pero se refieren a "algo" como servicio. Para facilitar la comprehensivo de estos tres términos se puede realizar la siguiente comparación con On-Premises el cual hace referencia a la administración total de la tecnología: (Sakovich, 2018)

Figura 4.3: Comparación entre On-Premise Infraestructura como servicio, plataforma como servicio y software como servicio



(Boteo, 2020)

Infraestructura como Servicio(IaaS):

IaaS, este es equivalente a los tradicionales centros de datos. Los proveedores de infraestructura en la nube utilizan ambientes virtuales para ofrecer recursos de computación escalables a los clientes, estos ofrecen servidores, redes, almacenamiento. Esto es beneficioso para los clientes porque no deben comprar dispositivos y manejar sus componentes. Ellos pueden acceder a estas plataformas e instalar los sistemas que deseen en las máquinas que ofrecen los proveedores. El proveedor es responsable por toda la infraestructura pero los usuarios tienen el total control sobre esta. Algunas soluciones de este estilo son: Microsoft Azure, Google Compute Engine, Amazon Web Services, Cisco MetaPod, DigitalOcean, Linode and Rackspace. (Sakovich, 2018)

Características clave de IaaS:

- Recursos altamente escalables.
- Infraestructura a nivel empresarial.
- El costo depende de la demanda.
- Arquitectura multi arrendamiento, se comparte dispositivos físicos entre varios usuarios.

Ventajas de utilizar IaaS:

- El modelo más flexible y dinámico.
- Costo según se requiera.
- Fácil de utilizar debido al hardware automatizado.
- Las tareas de administración son virtualizadas.

Desventajas de utilizar IaaS:

- Pueden haber problemas de seguridad debido a la arquitectura multi arrendamiento.
- Si el proveedor falla los usuarios no pueden acceder a su información.
- Es necesario entrenar el equipo para utilizar esta infraestructura.

Cuando es recomendable utilizar IaaS:

- Al ser una compañía pequeña o un emprendimiento que no tiene presupuesto para crear su propia infraestructura.
- Al ser una compañía que crece rápidamente y la demanda son inestables y cambiantes.
- Al ser una compañía grande que quiere tener un control efectivo sobre su infraestructura y pagar únicamente lo que se utiliza.

Plataforma como servicio (PaaS):

PaaS en computación en la nube es un marco de trabajo para la creación de software entregada por internet. Esta ofrece una plataforma creada con software, componentes y herramientas que los desarrolladores pueden utilizar para crear, personalizar, probar y lanzar aplicaciones. Proveedores de PaaS administran servidores, sistemas operativos, actualizaciones, actualizaciones de seguridad y backups. Los clientes se enfocan en el desarrollo de aplicaciones sin preocuparse de la infraestructura o el mantenimiento del sistema operativo. Algunos ejemplos de proveedores de PaaS Google App Engine, Amazon AWS, Windows Azure Cloud services, Heroku, AWS Elastic BeanStalk, Apache Stratos and OpenShift. (Sakovich, 2018)

Características clave de PaaS:

- Permite desarrollar, probar y almacenar aplicaciones en el mismo ambiente.
- Los recursos pueden escalar según las necesidades del negocio.
- Varios usuarios pueden acceder a la misma aplicación en desarrollo.

- Para algunos casos se pueden integrar servicios web y bases de datos.
- Equipos remotos pueden trabajar fácilmente.

Ventajas de utilizar PaaS:

- El software de PaaS es altamente escalable, disponible y puede soportar varios usuarios.
- El proceso de desarrollo es agilizado y simplificado.
- Se reducen los costos por crear, probar y lanzar aplicaciones.

Desventajas de utilizar PaaS:

- Problemas con la seguridad de los datos.
- Problemas con compatibilidad en infraestructura propia.
- Se depende de la velocidad, confianza y soporte del proveedor.

Cuando es recomendable utilizar PaaS:

- Es rentable para desarrolladores que quieren enfocar sus esfuerzos en desarrollar, probar y lanzar aplicaciones.
- Varios desarrolladores pueden trabajar en el mismo proyecto.
- Más proveedores deben ser incluidos.

Software como servicio (SaaS):

Usuarios pueden acceder a software en la nube. Los usuarios no necesitan descargar e instalar el software en sus dispositivos locales, pero hay veces que necesitan extensiones. El acceso de SaaS software es por medio de redes remotas que pueden ser accesadas por servicios web o APIs. En estas aplicaciones los clientes pueden colaborar en proyectos, almacenar información y analizarla. En esta modalidad el proveedor se encarga de manejar todo el hardware y la funcionalidad de la aplicación. Los clientes no son responsables de algo en este modelo, únicamente utilizan los programas para realizar sus tareas. Algunos ejemplos son: Google Apps, Dropbox, gmail, salesforce, Cisco WebEx, Office365, GoToMeeting. (Sakovich, 2018)

Características clave de SaaS:

- Para utilizarlas se utilizan suscripciones.
- No es necesario descargar, instalar o actualizar el software.
- Los recursos pueden ser escalables dependiendo de los requerimientos.
- Las aplicaciones son accesibles para cualquier dispositivo conectado.
- El proveedor es responsable por todo.

Ventajas de utilizar SaaS:

- No existen costos en hardware.

- No costos por configuraciones iniciales.
- Actualizaciones automáticas.
- Compatibilidad con múltiples dispositivos.
- Accesible en cualquier locación.
- Modelo de paga según se utiliza.
- Escalabilidad.
- Fácil personalización.

Desventajas de utilizar SaaS:

- Falta de control.
- Soluciones limitadas.
- Es necesario conectividad.

Cuando es recomendable utilizar SaaS:

- Si la compañía necesita un software listo en cualquier momento.
- Para proyectos pequeños que necesitan colaboración.
- Para aplicaciones que necesitan acceso web y acceso en móviles.

4.12. Herramientas de CI/CD

Las herramientas de integración continua(CI) y distribución continua(CD) sirven para distribuir aplicaciones de forma periódica, mediante el uso de la automatización de las etapas del desarrollo de aplicaciones. Algunos ejemplos de ellas son: Jenkins, Travis CI, Circleci, Team City y Bamboo.

Un Pipeline es un conjunto de prácticas para incorporar la automatización continua y el control permanente en todo el ciclo de vida de un proyecto. Esto comprende las etapas de integración, pruebas, distribución e implementación. Estas son algunas características de un Pipeline CI/CD

- Integración y verificación: en un flujo de trabajo de software estandarizado, se espera que varios desarrolladores trabajen en sus propias ramas. Estas deben integrarse periódicamente. Antes de que una rama se integre con las demás es necesario verificar que no exista algún conflicto con las funcionalidades existentes.
- Automatización: para agilizar los procesos es necesario que los pasos estén automatizados. Este paso reduce el tiempo en que los desarrolladores se dedican a tareas repetitivas y elimina los tiempos de espera en etapas de desarrollo.
- Cultura DevOps: el equipo de desarrollo tendrá la capacidad de detectar y solucionar problemas con anticipación antes que sean problemas mayores. Podrán realizar pruebas con mayor frecuencia.
- Utilizar contenedores: esta capacidad permite reducir la complejidad de compatibilidad con la infraestructura. Además facilita el transporte de la aplicación.

Algunos de los beneficios de implementar estas prácticas son:

- Definir controles de calidad minuciosos como que la compilación no deben poseer alertas ni errores o que todas las pruebas unitarias deben ser exitosas.
- Pruebas continuas automatizadas, esto comprende pruebas unitarias, análisis de seguridad, pruebas de rendimiento y luego generar reportes sobre los resultados de estas fases.
- Mecanismos para identificar vulnerabilidades en las dependencias.
- Despliegue controlado por medio de contenedores, esta capacidad permite implementar condiciones o esperar por la aceptación de una persona para que la aplicación pueda ser expuesta en producción cuando sea conveniente.

(Vergara, 2019)

4.13. ¿Qué es un contenedor de Docker?

Un contenedor es una unidad estándar de software que almacena el código y todas sus dependencias para que la aplicación pueda correr rápidamente y de manera consistente de un ambiente de cómputo a otro. Una imagen de un contenedor de Docker es ligera, autónoma, un paquete ejecutable de software que incluye todo lo necesario para correr la aplicación: tiempo de ejecución, herramientas de sistema, librerías de sistema y configuraciones. (Docker, 2020)

Una imagen es la definición de un contenedor antes de ser ejecutado y que además está disponible para cualquier sistema operativo, el software contenerizado siempre se ejecutará de la misma manera, sin importar la infraestructura. Los contenedores separan el software de su ambiente y aseguran que trabajará uniformemente sin importar las diferencias entre las instancias de desarrollo y producción.(Docker, 2020)

4.14. Comparación contenedores y máquinas virtuales

Contenedores y máquinas virtuales tienen una separación y asignación de recursos con beneficios similares pero funcionan diferente iniciando por los contenedores los cuales virtualiza el sistema operativo, por otro lado las máquinas virtuales virtualiza el hardware. Los contenedores son más portables y eficientes. Las máquinas virtuales son una abstracción de los dispositivos físicos convirtiendo un servidor en varios servidores. Múltiples MV pueden correr en una misma máquina física mediante el uso de un hypervisor. Cada Vm incluye una copia de un sistema operativo, la aplicación, necesariamente binarios y librerías ocupan alrededor de diez GB. Por otro lado, un contenedor puede ser ejecutado desde una misma máquina y compartir el kernel del sistema operativo con otro contenedor pero cada uno corriendo procesos separados en el espacio del usuario. Los contenedores pueden ocupar menos espacio que las máquinas virtuales(las imágenes de contenedores ocultan alrededor de 10 MB), pueden manejar más aplicaciones y necesitan menos máquinas virtuales y sistemas operativos. (Bauer, 2018)

CAPÍTULO 5

Marco metodológico

Previo a la planificación de la implementación de tecnologías fue necesario definir y explicar cuáles son las características que se esperan de las partes involucradas. A continuación se describe qué es lo que puede esperar cada individuo según el rol que tiene en lo posibles ambientes donde tendrá interacción con el proyecto y como este aporta según sus necesidades. Adicionalmente se muestra diagramas de la interacción que tiene el individuo con el sistema en sus diferentes etapas.

5.1. Características del proyecto

Así como lo menciona la Alianza Ágil en 2001, para poder ser exitoso en la economía electrónica es necesario tomar decisiones agresivas en la era de negocios y comercios electrónicos (e-business e e-commerce) más aún cuando una pandemia como COVID-19 ha pausado repentinamente varias industrias y deben de recuperarse de las pérdidas del 2020 lo antes posible. Varias empresas han descubierto las capacidades que tiene la tecnología para mejorar sus negocios y poder lograr una ventaja competitiva en la industria. En este tipo de situaciones las metodologías ágiles son ideales para desarrollar soluciones tecnológicas con pocos recursos y en cortos periodos de tiempo. Muchas de las soluciones para implementación de tecnologías en los negocios implican el uso de servicios en la nube para manejar costos eficientes. Sin embargo la configuración e instalación de servicios en la nube puede ser muy complicada para los desarrolladores. Por esa razón se decidió crear una solución que automatice la creación y configuración de máquinas virtuales, ambientes de ejecución de software en la nube y sistemas de integración y entrega contínua. Con la idea de que un desarrollador junior pueda realizar la configuración de forma más simple y enfocar su tiempo en desarrollar la solución específica para el cliente.

5.2. Usuario objetivo

Debido a la crisis múltiples emprendedores han tenido que cambiar metodologías tradicionales por implementaciones tecnologías. Este proyecto espera apoyar a cualquier equipo de trabajo interesado en implementar herramientas de CI/CD para administrar su proyecto pero que no tiene los conocimientos para instalarlas y configurarlas.

Roles:

- Cliente o dueño del proyecto(stakeholder): persona dispuesta a invertir económicamente en desarrollos que soporten los procesos de negocio vitales para producir ganancias. Consta de un corto presupuesto y necesita generar ganancias lo antes posible para recuperarse de la crisis causada por COVID-19.
- Desarrollador: recurso humano contratado por el cliente para implementar soluciones tecnológicas que soporten los procesos de negocio. Este recurso es limitado debido al presupuesto por lo que se espera un perfil de desarrollador junior dispuesto a aprender sobre la implementación de servicios en la nube.
- Usuario final: conjunto de personas que ya no pueden tener acceso a los servicios del cliente tradicionales. Están dispuestos a cambiar la interacción del beneficio aportado por el cliente a soluciones tecnológicas de vanguardia y aportar económicamente según el servicio que necesite.

5.3. Descripción general del proyecto

Este proyecto es catalogado como PaaS debido a que se implementaron diferentes configuraciones para habilitar una plataforma capaz de soportar varios tipos de desarrollo tecnológico. Por medio de dos archivos de configuraciones y el código fuente de un proyecto esta plataforma tiene la capacidad de crear una máquina virtual personalizada según características(tamaño, capacidad, servicios) especificadas por el cliente. Además esta máquina virtual genera automáticamente un ambiente para administrar el ciclo de vida del proyecto(CI/CD) según sea especificado en los archivos de configuraciones. Esta plataforma administra los servicios en la nube por el usuario, para que sin importar sus conocimientos del usuario en servicios en la nube pueda desarrollar su proyecto y pueda generar valor rápidamente al usuario final.

5.4. Desarrollo del proyecto

Este proyecto fue realizado siguiendo los principios de una metodología ágil principalmente para obtener retroalimentación constante para que el resultado final pueda ser utilizado por un caso real. Para ello se planteó tres fases generales para el proyecto: inicialización, integración e interfaz.

Fase inicialización

Se inició por crear definición del proyecto. Según esto se realizó la selección de tecnologías según la funcionalidad definida. Además se realizaron pruebas unitarias de cada tecnología para validar la factibilidad de la automatización. La lista de tecnologías seleccionadas acorde a su aporte es la siguiente:

- React: Interfaz web para interacción humana.
- Node: Creación de servicios para administrar recursos en la nube de Google Cloud.
- Docker: Crea el ambiente para los desarrollos sin importar la infraestructura.
- Scripts de Bash Administra toda la instalación y configuración de las herramienta de CI/CD y docker en las maquinas virtuales.
- Jenkins: el plugin Jenkins As Code permite definir configuraciones en archivos para que la herramienta pueda crear pipelines sin la interacción humana.

- Git: Almacenamiento de archivos de configuración y el código fuente de cada proyecto.

Además se realizaron tres diagramas. La Figura 5.1 representa la interacción general que tiene cada tecnología y muestra a cuales de ellas es accedida por humanos. La Figura 5.2 muestra todos los pasos que fueron automatizados por medio de scripts y diferentes configuraciones, las cuales en casos tradicionales deben realizarse por personas. La Figura 5.3 muestra una perspectiva general de cómo interactúa cada rol con el proyecto.

Figura 5.1: Arquitectura del proyecto

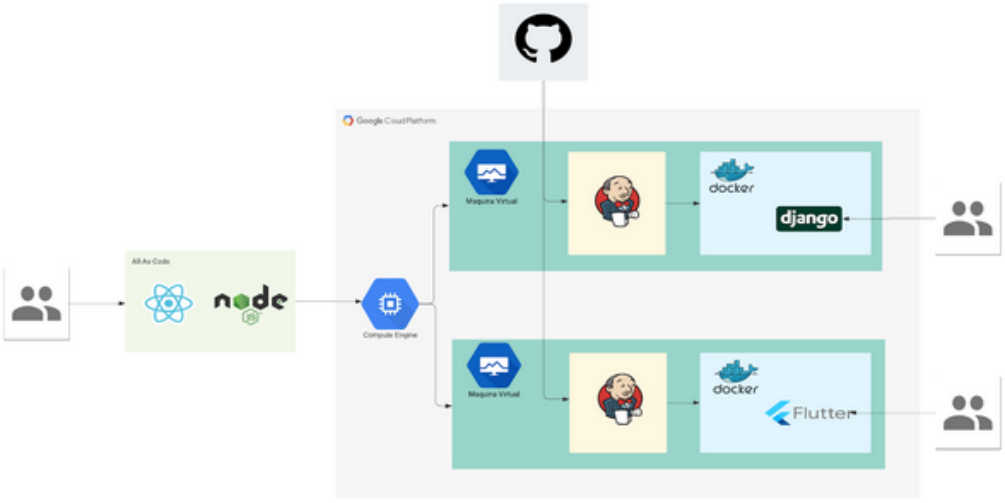


Figura 5.2: Caso de uso de inicialización de una máquina virtual

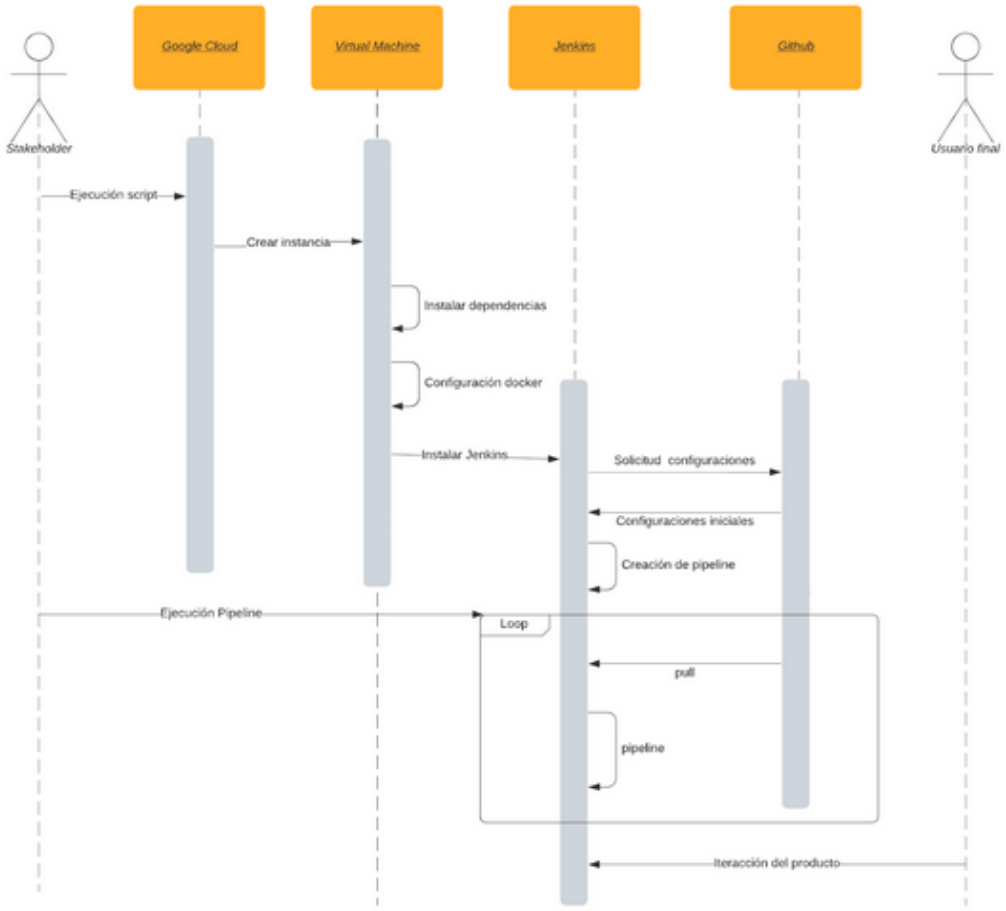
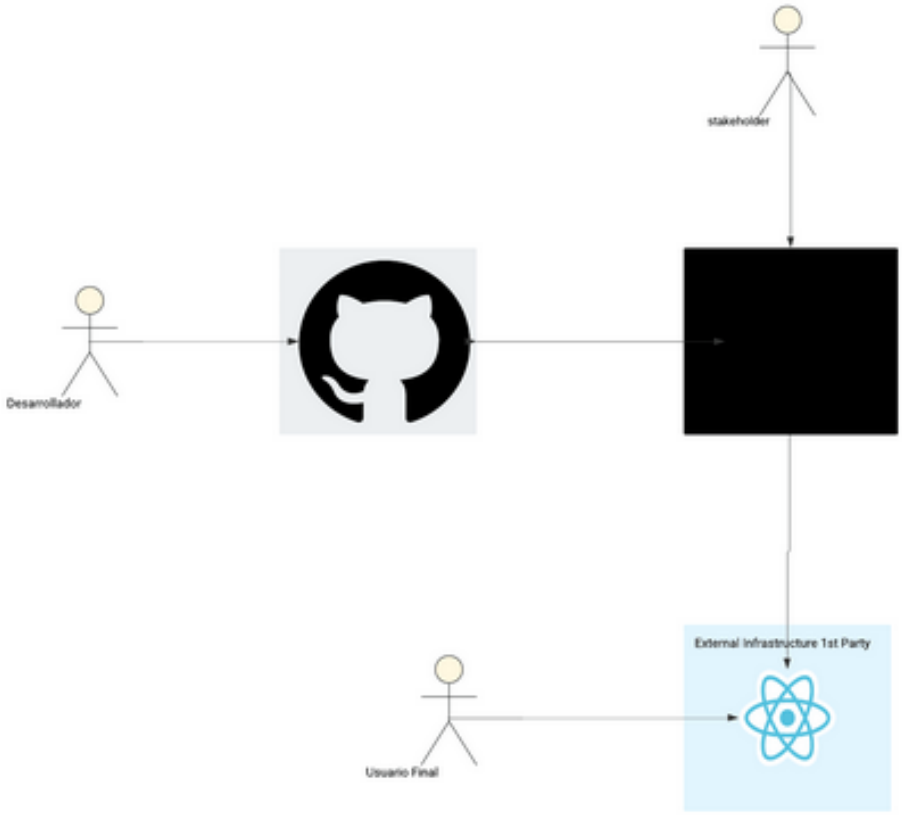


Figura 5.3: Perspectiva como desarrollador



Fase integración:

Esta fase implementó todas las tecnologías seleccionadas en el paso anterior para que ellas puedan trabajar en conjunto para desarrollar una plataforma que pudiera soportar cualquier desarrollo. En esta fase fue crítica la retroalimentación de un stakeholder desarrollo emergente del 2020 para validar la factibilidad del proyecto. Luego de que cada tecnología estuviera lista fue necesario modificar diferentes configuraciones en cada tecnología para que cada tecnología pudiera trabajar automáticamente y que el usuario no tuviera que realizar configuraciones adicionales. Además se realizaron varias entrevistas para comprobar que un desarrollador pudiera utilizar la plataforma.

Fase interfaz:

Esta fase se enfocó en implementar las interfaces necesarias que facilitaran la interacción de personas con el proyecto. Para ello se desarrolló una plataforma en React y se implemento el plugin de Blue Ocean para Jenkins el cual posee una interfaz dinámica y colorida para mejorar la interacción humana con las pipelines de desarrollo. Dichas interfaces se pueden observar a continuación:

Figura 5.4: Captura de pantalla de una pipeline luego de ejecutarse

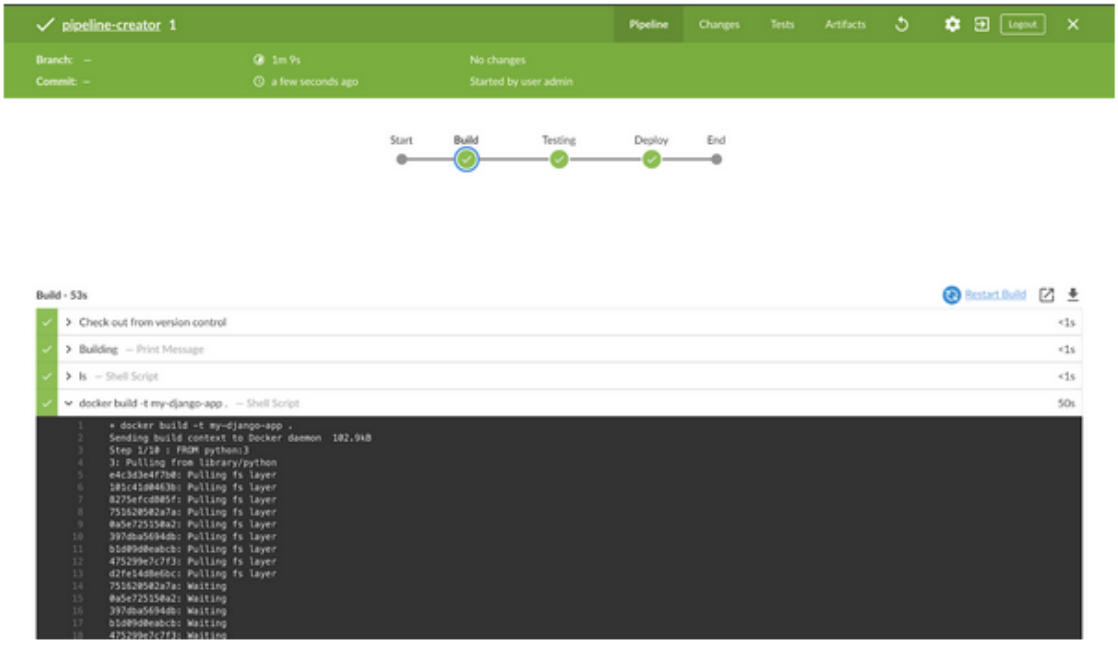
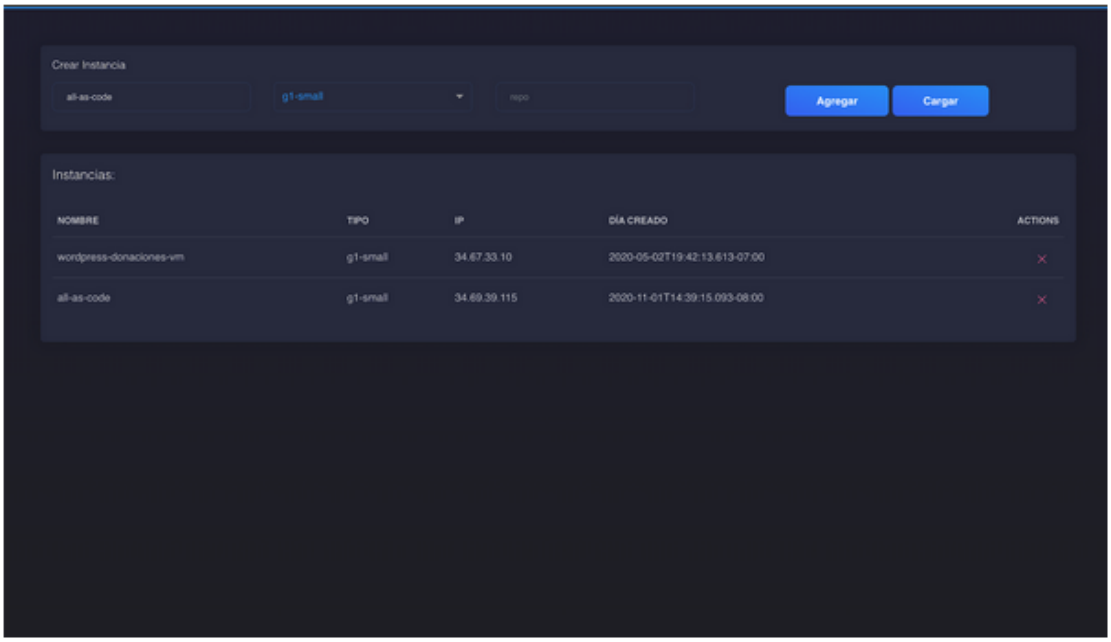


Figura 5.5: Interfaz para crear máquinas virtuales



Al finalizar las tres fases se obtuvo una herramienta gráfica en la cual un desarrollador junior puede crear automáticamente una máquina virtual en la nube y configurar un ambiente de ejecución específico para su proyecto por medio de scripts. La aplicación también configura automáticamente el sistema de integración y entrega continua, para que sea más fácil para el desarrollador actualizar su aplicación en la nube.

El código fuente de la aplicación desarrollada puede ser visto en el repositorio de Github especificado en la sección de anexos.

Teniendo la aplicación a su disposición, el desarrollador tendrá que crear un repositorio, que además de contener el código fuente de su solución, debe tener algunos archivos de configuración para poder desplegar el proyecto en la nube y configurar el sistema de integración y entrega contínua. Los pasos que el desarrollador debe seguir para utilizar la solución se detallan en la sección de anexos.

En el cuadro que se muestra a continuación se puede ver una comparación entre el flujo de trabajo tradicional y el flujo de trabajo con la solución desarrollada.

Tabla 5.1: Comparación de flujo tradicional con la plataforma desarrollada

Caso tradicional desde cero	Caso implementando el proyecto all-as-code
Seleccionar un proveedor de nube(AWS, GoogleCloud, DigialOcean, Azure). Crear cuenta en el sistema del proveedor seleccionado. Investigar cómo utilizar los servicios del proveedor seleccionado Investigar cómo funciona la infraestructura como servicio para el servidor seleccionado(s2, Google Compute Engine). Verificar que la cuenta tenga los permisos necesarios para administrar el recurso necesario. Crear una instancia según el tamaño del proyecto(verificar reglas de firewall). Ingresar a la instancia creada. Puede ser necesario actualizar el sistema operativo e instalar herramientas de linea de comando. Descargar e instalar herramienta de CI/CD. Descargar e instalar herramienta de Docker. Ejecutar y configurar herramienta de CI/CD. Investigar configuraciones de la herramienta de CI/CD para que sea compatible con github. Investigar e implementar configuraciones para que la herramienta de CI/CD sea compatible con Docker. Investigar y configurar pipeline automatizada. Realizar pruebas de integración. *Esto para cada proyecto.	Descargar he instalar proyecto all-as-code(https://github.com/nistalhelmuth/all-as-code.git). Siguiendo la documentación. Preparar proyecto según el framework de trabajo necesario. Siguiendo la documentación en Anexos o el repositorio de ejemplo: (https://github.com/nistalhelmuth/pipelines-sandbox). Ejecutar el proyecto all-as-code. Realizar pruebas de integración.

CAPÍTULO 6

Resultados

Para comprobar el alcance que tiene el proyecto respecto al tercer objetivo se realizó una encuesta para identificar debilidades y mejoras para la implementación de este proyecto. Esta encuesta fue realizada para obtener retroalimentación de personas que se encuentran familiarizadas con desarrollos tecnológicos. Se buscan perfiles similares a un desarrollador junior que esté familiarizado con lenguajes de programación de vanguardia y que por lo menos tenga conocimientos básicos sobre servicios en la nube. La entrevista fue realizada por medio del servicio de Encuestas de Google Drive para facilitar la interacción. A continuación se muestra los resultados obtenidos:

Figura 6.1: Pregunta 1. ¿Qué tan familiarizado está con los servicios de la nube?

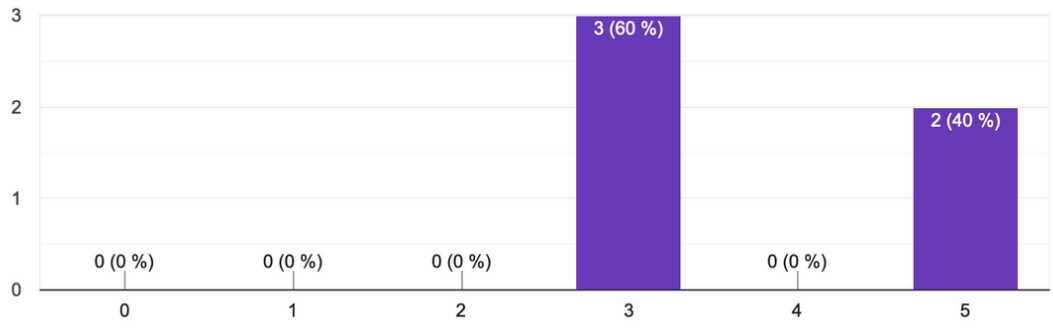


Figura 6.2: Pregunta 2. ¿Qué tan familiarizado está con herramientas de CI/CD?

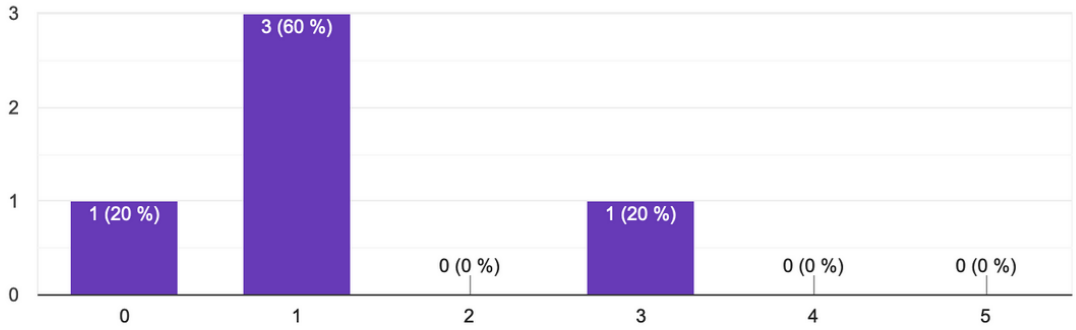


Figura 6.3: Pregunta 3. ¿Qué tan difícil considera usted que es la curva de aprendizaje para los servicios en la nube?

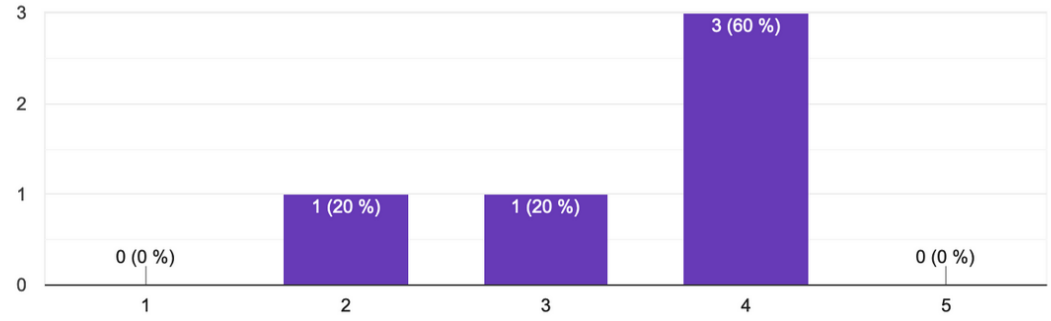


Figura 6.4: Pregunta 4. ¿Qué tan familiarizado está con Docker?

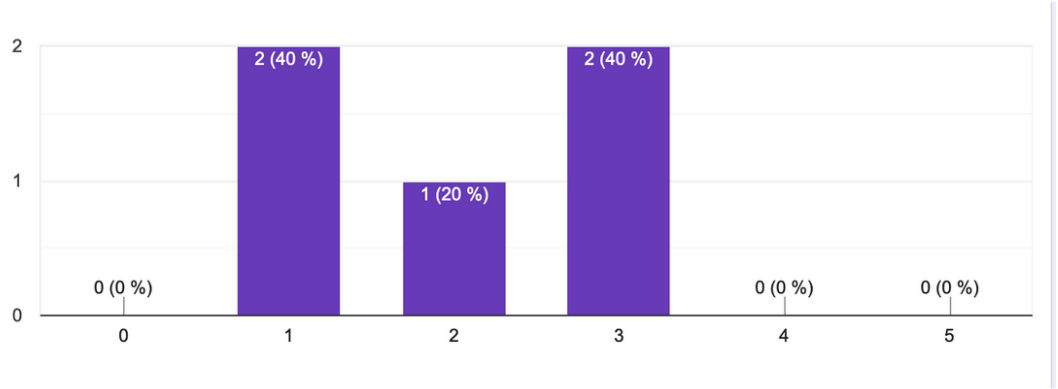


Figura 6.5: Pregunta 5. ¿Con cuáles frameworks de programación está familiarizado?

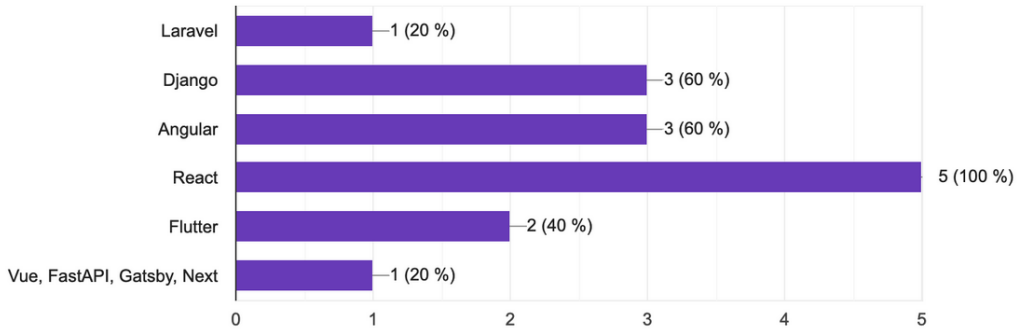


Figura 6.6: Pregunta 6. ¿Qué tan difícil es implementar el prototipo?

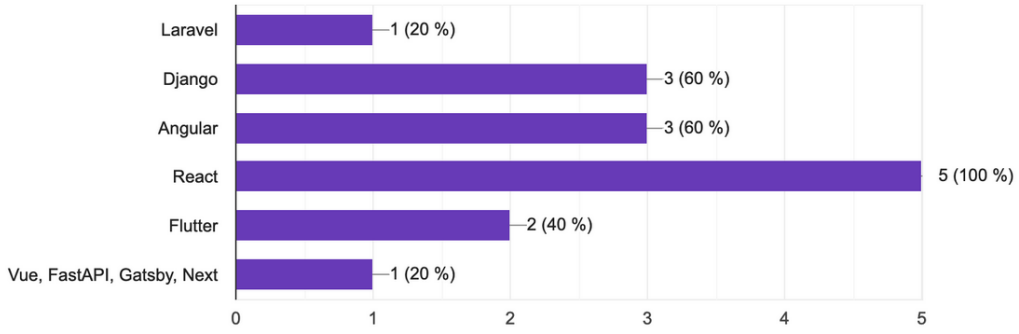


Figura 6.7: Pregunta 7. ¿La documentación qué tanto facilita la implementación?

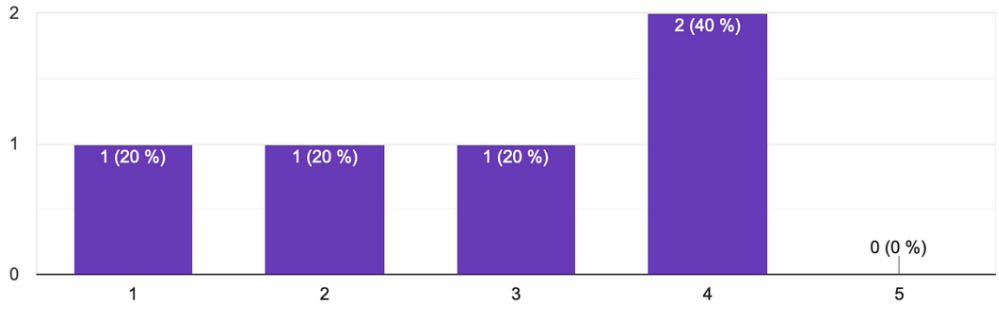


Figura 6.8: Pregunta 8. ¿Qué tan útil sientes la herramienta?

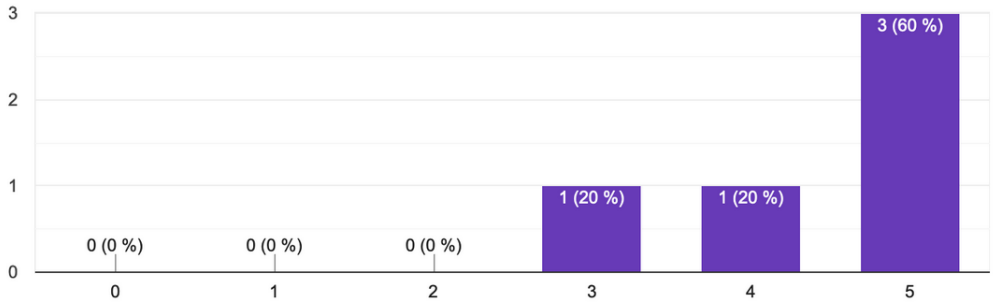


Figura 6.9: Pregunta 9. ¿Estás dispuesto a recomendarla a tus conocidos?

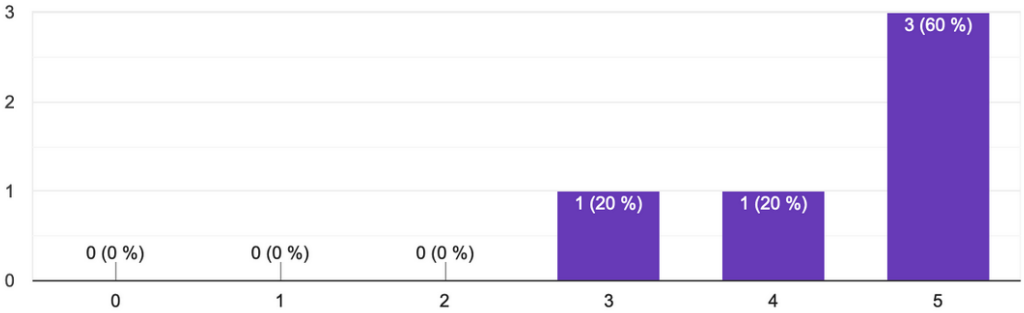
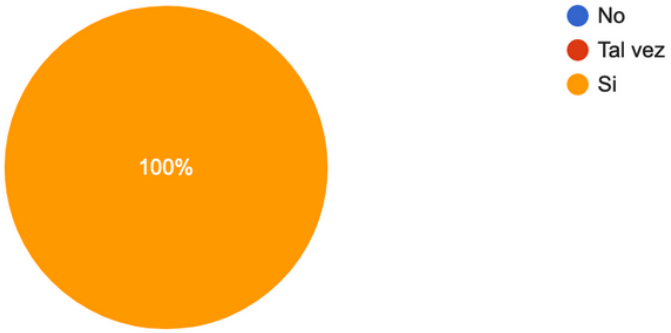


Figura 6.10: Pregunta 7. ¿La documentación qué tanto facilita la implementación?



Comentarios de las encuestas:

- No quemes tus credenciales.
- Explicar ventajas y desventajas de implementación.
- Extender el proyecto para ser utilizado en cualquier nube (Azure, GCP, AWS, DigitalOcean).
- ¿Qué sucede con las implementaciones en la play store?

Datos adicionales sobre el proyecto en sí:

- Toda la creación y conexión de las tecnología tarda menos de 10 minutos luego de presionar "crear" desde la aplicación web desarrollada.
- El desarrollador en ningún momento debe configurar o instalar. alguna dependencias de la instancia.
- El desarrollador únicamente debe configurar dos archivos: jenkinsfile con la especificación de los pasos para el pipeline y un dockerfile con la especificación de las dependencias para compilar el proyecto.
- El stakeholder tiene la capacidad de publicar una versión estable en cualquier momento.
- Pueden eliminarse todas las instancias sin afectar los avances del proyecto.

Figura 6.11: Costos durante la elaboración del proyecto por mes del año 2020

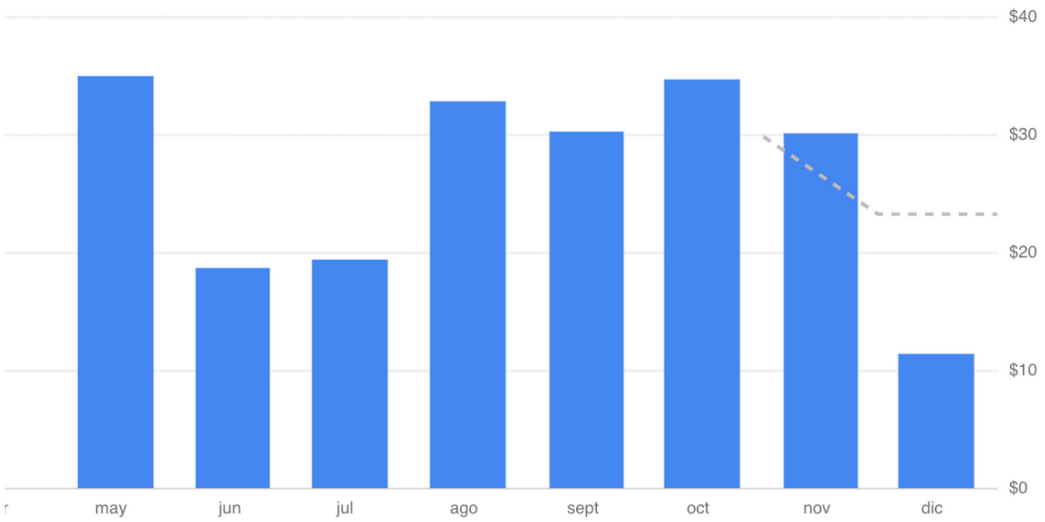


Figura 6.12: Mínima venta por mes por una microempresa

$$\frac{Q2{,}825.00\ (s.minimo)}{12\ meses * 7.8\ {}^{Q}/_{\$}} = \boxed{30.18\$\ mensuales}$$

CAPÍTULO 7

Discusión resultados

En casos reales muy pocas empresas están en la disponibilidad de implementar herramientas basadas en las prácticas de DevOps. La implementación de estas prácticas resultan en un esfuerzo adicional que no impacta directamente la percepción que tiene el usuario del producto. Sin embargo, los resultados se ven reflejados tanto en la rentabilidad del negocio como en la calidad del producto. Esto puede dar una ventaja competitiva en el mercado respecto a empresas que no poseen tecnologías de servicios administrados. Respecto a las MIPYMES esto representa una oportunidad para que puedan retomar sus operaciones laborales que han sido suspendidas durante el año 2020 debido a la pandemia causada por COVID-19.

Este proyecto fue una prueba de concepto de cómo las herramientas basadas en las prácticas de DevOps pueden reducir la curva de aprendizaje que poseen los servicios de proveedores en la nube. Se utilizó el caso de una idea emprendedora por consecuencias de COVID-19 pero el conjunto de herramientas pueden aplicarse a cualquier otro desarrollo. Adicionalmente todo el desarrollo de este proyecto fue realizado basándose en metodologías ágiles con el objetivo de administrar el riesgo de manera eficiente y así, implementar una alta cantidad de tecnologías con recursos limitados.

Respecto a las entrevistas, las primeras 5 preguntas fueron estructuradas con el fin de corroborar qué tantos conocimientos poseía la persona. En base a estas respuestas se puede saber que a pesar que los desarrolladores ya posee conocimientos básico para implementar servicios en la nube(pregunta 1) esto representa un gran esfuerzo(pregunta 3). Adicionalmente existen varias tecnologías como Docker y las herramientas de CI/CD que no son muy conocidas por las personas encuestadas(preguntas 2 y 4) a pesar que si hay varias tecnologías de desarrollo que se conocen como las mencionadas en las respuestas de la pregunta 5.

Las últimas 5 preguntas fueron planteadas para entender la aceptación del proyecto por parte de las personas. Iniciando por la pregunta 6 la cual expresa que el proyecto necesita un esfuerzo notable para su implementación. Sin embargo las preguntas 7 y 9 muestra que la aceptación del proyecto es positiva. Los desarrolladores muestran interés por implementar este tipo de tecnologías debido a que representan algún tipo de beneficio para sus tareas como desarrolladores. La pregunta 8 permite demostrar que la curva de aprendizaje es reducida por medio del proyecto y su respectiva documentación

CAPÍTULO 8

Conclusiones

- Se crean instancias bajo demanda con scripts que crean ambientes de entrega continua, en menos de 10 minutos, que a su vez cuentan con pipelines configuradas automáticamente.
- Se implementa pipelines automatizados para entrega continua para un desarrollo emergente del 2020 debido a medidas preventivas COVID-19 pero que también funciona para cualquier tipo de desarrollo que pueda ser ejecutado por Docker.
- Esta implementación de servicios en la Nube permite manejar una economía de escala accesible para que MIPYMES puedan generar dinero sin mucha inversión.
- Se elaboró una documentación que reduce la curva de aprendizaje de la servicios en la nube y metodologías DevOps por medio de un caso práctico.
- Se desarrolló una interfaz que reduce la complejidad de administrar máquinas virtuales de los servicios de Google Cloud.

CAPÍTULO 9

Recomendaciones

- Este proyecto solo fue una prueba de concepto por lo que se recomienda implementarlo en más casos para comprobar su funcionalidad.
- Docker es una herramienta que restringe el acceso a los usuarios maliciosos sin embargo no asegura un ambiente completamente seguro. Es necesario implementar prácticas de seguridad como: administrar secrets, encripción de datos y pruebas de vulnerabilidades.
- Jenkins es una herramienta que está en constante desarrollo por lo no se puede asegurar que soporte todos los casos de desarrollo. Adicionalmente el plugin JasC fue publicado el 2019 y aún tiene muchas áreas para mejorar, como lo es su documentación.
- Los scripts de Node y React implementados no toman en cuenta buenas prácticas para manejar secretos ni utilizan certificados TLS para evitar ataques estilo man in the middle.
- Este proyecto implementa crear máquinas virtuales en Google Cloud únicamente de Ubuntu por lo que se recomienda tenerlo en cuenta para la compatibilidad del desarrollo.

CAPÍTULO 10

Bibliografía

- Agile manifesto. (2001, February). Retrieved from Manifiesto for Agil Software Development: http://agilemanifesto.org/
- Bauer, R. (2018, Junio 18). *What's the Diff: VMs vs Containers*. Retrieved from Backblaze: https://www.backblaze.com/blog/vm-vs-containers/
- Belagatti, P. (209, February 10). *The State of DevOps Adopotion and Trends in 2017*. Retrieved from DevOps.com: https://devops.com/state-devops-adoption-trends-2017/
- Bocetta, S. (2020, June 18). *How DevOps Can Save Your Business from COVID-19*. Retrieved from DevOps.com: https://devops.com/how-devops-can-save-your-business-from-covid-19/
- Boleto, Boteo, P. (2020). *Perspectivas económicas 2020 ¿Será un buen año? Reporte de Actividad Económica- Noviembre 2020*.
- Chandu, V. (2019, Febrero 5). *Top 5 Reasons Why DevOps Is Important*. Retrieved from DZone: https://dzone.com/articles/top-5-reasons-why-devops-is-important
- Dirkmaat, O., Fernandez, D. (2020, Abril 4). *El impacto del COVID-19 en Guatemala: PIB podría caer 16 % en T2 2020*. Retrieved from UFM Market Trends: https://trends.ufm.edu/articulo/pib-impacto-covid/
- DOERRFELD, B. (2020, Agosto 4). *Top Pressing Concerns for CI/CD in 2020*. Retrieved from devops.com: https://devops.com/top-pressing-concerns-for-ci-cd-in-2020/
- Guatemala Panorama general. (2020, Octubre 9). Retrieved from banco mundial: https://www.bancomundial.org/es/country/guatemala/overview1
- Kim, G., Humble, J., Debois, P., Willis, J. (2016). *The DevOps Handbook: how to create world-class agility, reliability, security in technology organizations. Estados Unidos: IT revolution Press, LLC.*
- MINECO. (2017). *Informe de Situación y Evolución Del Sector MIPYME de Guatemala 2015-2017*. Retrieved from Ministerio de Economía de Guatemala: https://www.mineco.gob.gt/desarrollo-de-la-mipyme
- Palao, C. (2020). *Las estrategias de DevOps son importantes, ahora más que nunca. COMPUTERWORLD, 34-51.*

- Sakovich, N. (2018, Abril 14). *IaaS vs. PaaS vs. SaaS: What's the Difference?* Retrieved from Sam Solutions: https://www.sam-solutions.com/blog/iaas-vs-paas-vs-saas-whats-the-difference/
- Stoica, M. (2013). *Software Development: Agil vs Traditional. Bucharest University of Economic studies.*
- Vergara, S. (2019, Mayo 28). *¿Implementas algún pipeline CI/CD en tu organización?.* Retrieved from ITDO: https://www.itdo.com/blog/implementas-algun-pipeline-ci-cd-en-tu-organizacion/
- What is a Container? (2020). Retrieved from: https://www.docker.com/resources/what-container
- Wilsenach, R. (2015, July 9). *DevOpsCulture.* Retrieved from martinFowler.com: https://martinfowler.com/bliki/DevOpsCulture.html
- Wood, J., Lah, T. (2016). *San Diego: Technology Services Industry Association.*

CAPÍTULO 11

Anexos

11.1. Anexo 1

- Código y documentación del proyecto: https://github.com/nistalhelmuth/all-as-code
- Este proyecto unicamente es compatible con github así como se muestran en los siguientes repositorios de ejemplo:
  - Django: https://github.com/nistalhelmuth/pipelines-sandbox
  - WordPress(Con mysql): https://github.com/nistalhelmuth/pipeline-wordpress

11.2. Anexo 2

- Identificar el framework de programación para el proyecto (Django, Angular, Flutter, Wordpress, React, etc)
- Localizar: imagen de Docker compatible con el framework identificado. Este puede ser buscado en: https://hub.docker.com/
- Verificar que todos los archivos de configuración de dependencias sean parte del repositorio que contendrá el desarrollo, como referencia:
- Django utiliza requirements.txt para listar las dependencias de python necesarias para el proyecto.
- Node utiliza package.json para listar las dependencias de node necesarias para el proyecto.
- Crear Dockerfile para compilar el proyecto con las dependencias y configuraciones necesarias para el proyecto específico. Para los ejemplos mencionados anteriormente se utilizó de guía la documentación oficial de Docker:
  - Django: https://docs.docker.com/compose/django/
  - WordPress: https://docs.docker.com/compose/wordpress/
- Realizar pruebas locales para validar que el Dockerfile esté funcionando correctamente antes de utilizar el proyecto all-as-code

- Definir cuáles pasos son necesarios para una pipeline de desarrollo según las características del proyecto (Compilación, Pruebas Unitarias, Pruebas de Integración, Despliegue).
- Crear un Jenkinsfile representativo a los pasos ideados anteriormente, puede tomar como referencia los repositorios anteriores o utilizar la documentación Oficial de jenkins https://www.jenkins.io/doc/book/pipeline/jenkinsfile/
- Agregar el Archivo de Dockerfile y Jenkinsfile al repositorio.
- Ejecutar el proyecto all-as-code y especificar el nombre del repositorio en github (usuario/repositorio).

11.3. Anexo 3

- Entrevista realizada: https://docs.google.com/forms/d/e/1FAIpQLSfWxkLmkXQskEYGusBPHiIJqpa-jr2DfLCxXICM7Qs52EdN2g/viewform?usp=sf$_link$