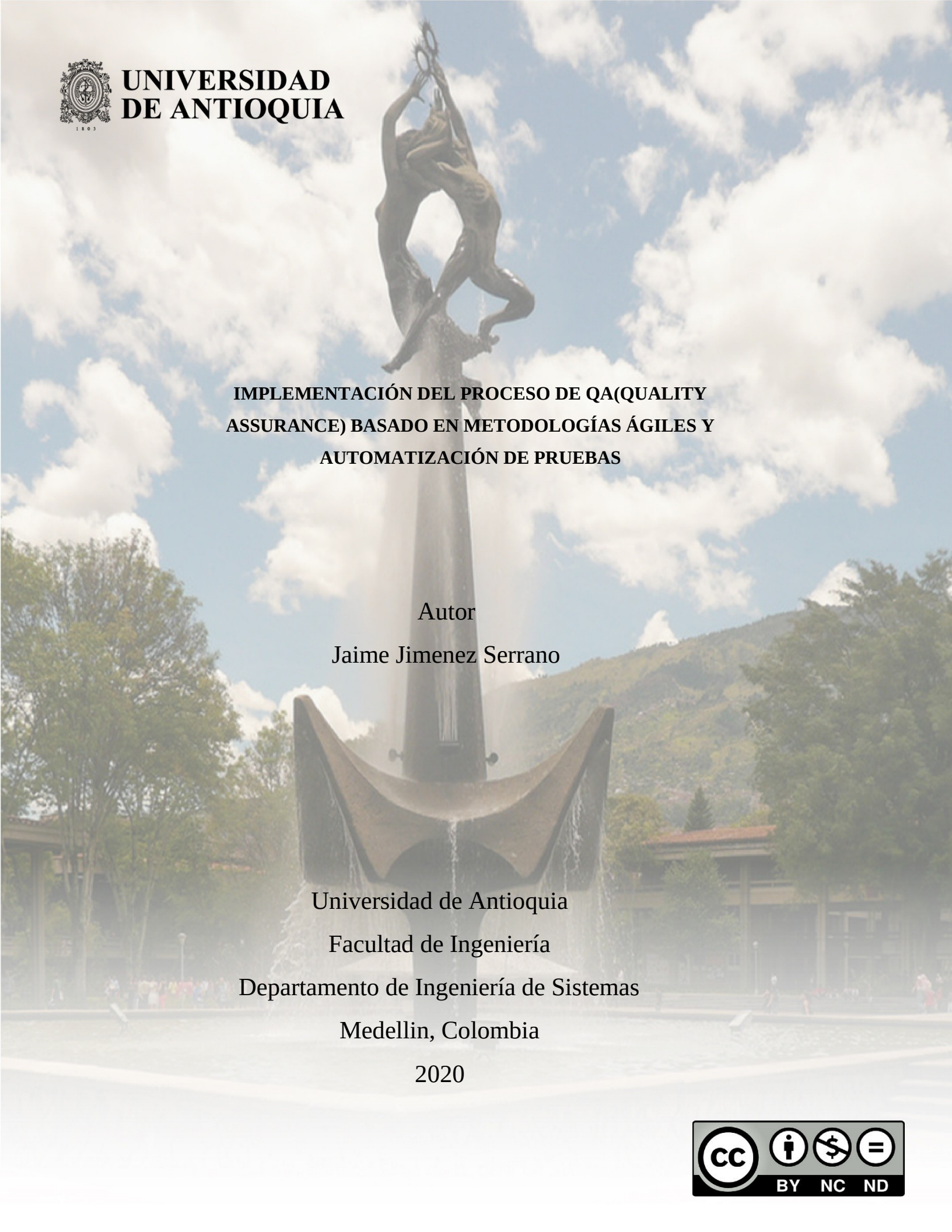

UNIVERSIDAD DE ANTIOQUIA

IMPLEMENTACIÓN DEL PROCESO DE QA(QUALITY ASSURANCE) BASADO EN METODOLOGÍAS ÁGILES Y AUTOMATIZACIÓN DE PRUEBAS

Autor

Jaime Jimenez Serrano

Universidad de Antioquia

Facultad de Ingeniería

Departamento de Ingeniería de Sistemas

Medellin, Colombia

2020

Implementación del proceso de QA(Quality assurance) basado en metodologías ágiles y automatización de pruebas

Jaime Jimenez Serrano

Trabajo de practicas presentado como requisito parcial para optar al título de:
Ingeniero de Sistemas

Asesores:
Roberto Florez Rueda, Ingeniero Civil
Mateo García Córdoba, Ingeniero de Sistemas

Universidad de Antioquia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas
Medellin, Colombia
2020

Introducción

En el desarrollo de un producto o la prestación de un servicio, se requieren procesos que permitan controlar y asegurar estándares de calidad establecidos, los productos de software no están exentos a la implementación de estos procesos, con el fin de producir software capaz de satisfacer las necesidades del usuario.

En el modelo de desarrollo tradicional, está definida la fase de pruebas como una de las fases finales del desarrollo, en donde un conjunto de pruebas establecidas, son ejecutadas por un equipo de testers y con el fin de encontrar posibles fallos en el producto desarrollado, este modelo, sin embargo, no permite encontrar fallos en previas fases al desarrollo, lo que se traduce en muchos casos en un incremento de costos y tiempo.

En los últimos años con el nacimiento de nuevas metodologías de desarrollo ágiles, enfocadas a entregas iterativas e incrementales, buscando entregar en cada una las funcionalidades que más valor produzcan al software desarrollado; han requerido también nuevas formas de asegurar la calidad, como, la transversalidad de las pruebas en todo el ciclo de desarrollo, la automatización de las pruebas y la implantación de métodos de medición tales como la cobertura de código que tienen las pruebas.

En la actualidad, la organización Grupo Creativo Habla S.A.S., presenta un modelo de desarrollo ágil, no obstante, el proceso de pruebas y aseguramiento de la calidad del software, sigue usando lineamientos tradicionales, lo que conlleva a retrasos en las entregas, así como una posterior corrección de defectos no detectados, además de la posible inserción nuevos defectos al no contar con mecanismo que permita llevar una clara trazabilidad de los defectos eliminados.

Por lo anterior, se propone la implementación de un nuevo proceso de QA, basado en metodologías ágiles y la automatización de pruebas que permita mejorar los tiempos de entrega y reducir las correcciones posteriores a las entregas.

Objetivos

Objetivo general

Mejorar los procesos de pruebas con la implementación de metodologías de pruebas enfocados a desarrollos ágiles y automatización de pruebas, con el fin de reducir tiempos de entrega y el número de defectos presentes posteriormente.

Objetivos específicos

- Seleccionar el enfoque ágil que ofrezca procesos que permitan la automatización de pruebas.
- Especificar las herramientas a usar para el desarrollo de pruebas a usar para las tecnologías usadas en la organización.
- Definir el tipo de pruebas que se realizaran, así como el orden y buenas prácticas de las mismas.
- Seleccionar la plataforma para realizar integración continua y despliegue continuo, con base en sus costos y las funcionalidades ofrecidas para manejo de cobertura de código.
- Desplegar aplicación con la implementación de automatización de pruebas.

Proceso

Para la implementación del proceso de QA, se realizo la selección de la metodología de testing ágil Test Driven Development (TDD), frente a otras dos opciones las cuales eran Behavioral Driven Development (BDD) y Acceptance Test Driven Development (ATDD).

Parametros	**TDD**	**BDD**	**ATDD**
Definición	Es una metodología enfocada principalmente en la implementación de una caracteristica	Metodología enfocada en el comporatiento del sistema	Es una metodología similar a TDD, pero con mayor enfoque en la captura de requerimientos
Principales participentes	Desarrolladores	Desarrolladores, Cliente	Desarrolladores, Cliente
Lenguaje usado	El lenguaje de la plataforma	Gherkin (generalmente un ingles simple)	Gherkin (generalmente un ingles simple)
Principapal enfoque	Pruebas unitarias	Entendimiento de los requisitos	Escribir pruebas de aceptación

La selección de la metodología se realizo bajo los criterios siguientes selección:

- Tener una rápida implementación.
- Permitir obtener una retroalimentación en poco tiempo.
- Reducir el tiempo refactorizando código defectuoso.
- Facilitar el hallar defectos y errores prontamente.
- Dar flexibilidad y fácil mantenimiento futuro.

Con base en estos criterios, la metodología seleccionada fue TDD, ya que se basa en el proceso red-green-refactor, el cual consiste en un proceso cíclico en el que se elaboran primero las pruebas para que fallen, posteriormente añadir el código necesario para hacer que las pruebas sean exitosas, añadir mas escenarios a la prueba, volver a ejecutar las pruebas y refactorizar el código hasta que cumpla con todos los escenarios descritos para la prueba y que cumplan con la totalidad de la funcionalidad probada.

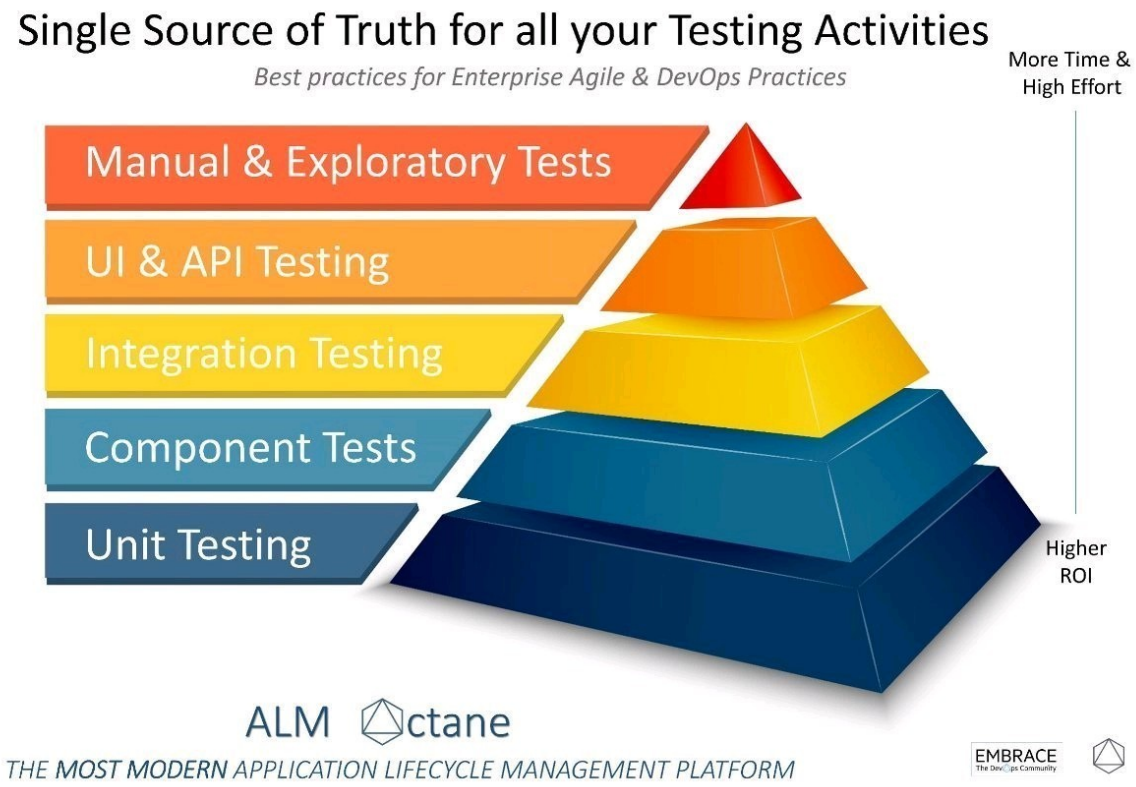

Figura 1: Resource : https://embrace-devops.com/2018/10/21/track-all-phases-of-the-agile-testing-pyramid/

Posterior a la selección de la metodología de testing, se prosiguió con la definición de las pruebas a realizar. Con el fin de buscar cubrir la mayor cantidad escenarios posibles en la aplicación se decidió seguir la convención de la pirámide de testing (figura 1).

Unit/Component tests (pruebas unitarias/componentes): Son las realizadas a componentes específicos de la aplicación de manera aislada.

Integración tests (pruebas de integración): Son las elaboradas para probar la interacción/integración de varios componentes.

UI/API tests (pruebas de UI/API): Son las encargadas de validar a la aplicación como un todo, simulando ser un usuario final.

Manual tests (pruebas manuales): Son pruebas finales de aceptación por el usuario.

La pirámide define el orden y magnitud de las pruebas a realizar, estando en la base las pruebas unitarias que deben corresponder al 70% - 80% de las pruebas de la aplicación, ya que son las que menor esfuerzo y tiempo requieren, pero mayor impacto generan.

Además de las pruebas definidas en la pirámide, se definió que se realizarían pruebas de regresión, las cuales son las correspondientes al correrse tras cambios en la aplicación.

Una vez definido el alcance de las pruebas de realizar para la plataforma, se continuo con el proceso de seleccionar las herramientas de testing a utilizar con el fin de realizar las pruebas especificadas en los lenguajes Javascript (React.js) y Ruby (Ruby on rails).

Para lograr este fin se especificaron los siguientes criterios de selección:

- Mayor cobertura por tipo de prueba.
- Una baja curva de aprendizaje para el desarrollador.
- Menor costo de implementación.
- Mantenibilidad(una comunidad activa que la soporte).
- Estabilidad y madurez en el mercado.

Con los criterios establecidos, se seleccionaron las siguientes herramientas por lenguaje y tipo de pruebas.

Pruebas Unitarias:

- Ruby on Rails: rspec, factorybot, database cleaner, email_spec.
- Javascript: jest, enzyme.

Pruebas de integración:

- Ruby on rails: rspec, factorybot, database cleaner, rspec-mock.
- Javascript: jest, react-testing-library, user-event.

Pruebas de UI/API:

- Ruby on rails(API): rspec, factorybot, database cleaner.
- Javascript(UI o E2E): selenium-webdriver.

Una vez seleccionadas las herramientas a usar para la implementación de las pruebas en los diferentes lenguajes y por tipo de prueba, se realizo la selección de la plataforma en la cual realizar la ejecución del despliegue continuo y la integración continua, debido a que se realizo una migración a la plataforma AWS(Amazon Web Services), fue seleccionada la herramienta Codepipeline ofrecida por AWS, debido a que generaba el menor costo y debido a que la aplicación web se encontraba en esta plataforma facilitaba la integración lo la herramienta ya mencionada.

Posteriormente se definió la estructura del Pipeline que la aplicación realizaría en cada despliegue de nuevas versiones (figura 2).

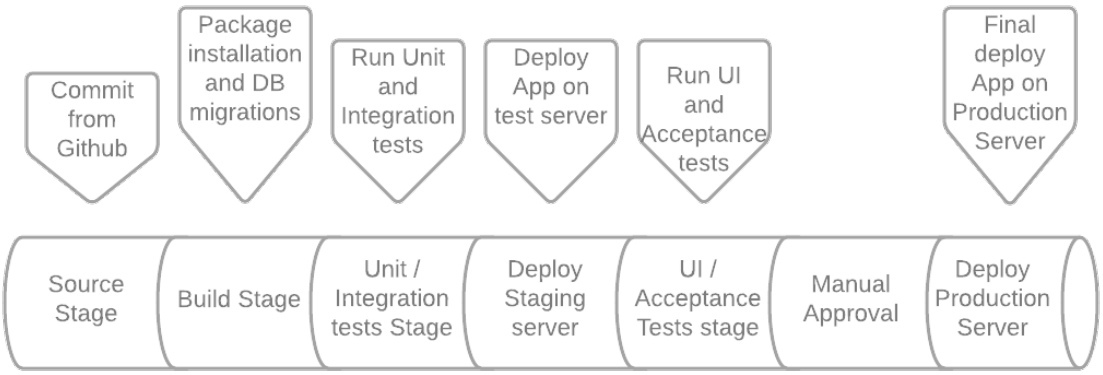

Figura 2: Pipeline definido para la aplicación

En el cual se definió que el proceso de despliegue iniciara al realizar un commit en la rama Main (Master), el cual iniciara la etapa de Build en la cual se instalaran paquetes necesarios para las posteriores etapas, así como las migraciones necesarias en la DB, tras finalizar esta etapa, se realizara la ejecución de las pruebas unitarias y de integración, si estas son exitosas se desplegara la nueva versión de la aplicación en el ambiente de staging(pruebas) una vez desplegado se realizaran las pruebas de aceptación o E2E, las cuales una vez son finalizadas con éxito, se esperara a una revisión manual y aceptación final, para finalmente realizar el nuevo despliegue en el ambiente de producción.

Una vez definidos los anteriores aspectos, se inicio la implementación del proceso de CI y CD, en el cual se estableció que para la primera entrega se implantarían un pequeño conjunto de pruebas que abarquen todos los tipos definidos, tanto en el entorno de Backend, como en el de Frontend, con las cuales se iniciaría el proceso de implementación inicial del pipeline, una vez elaborado este conjunto de pruebas, se realizo la configuración del pipeline en AWS usando Codepipeline para el orquestador del pipeline, Github como la fuente inicial del recurso para el proceso de CI/CD, Codebuild para realizar las etapas de Build y Test, ademas de brindar una herramienta que permita conocer la cobertura de código que cubren las pruebas, AWS Elastic Beanstalk para el despliegue de los entornos del backend y AWS Amplify para el despliegue de los entornos.

Posterior a este proceso, se realizo la presentación del nuevo flujo de despliegue y metodología de pruebas al equipo de trabajo, en el que se mostraron la implementación del pipeline, el proceso de elaboración de pruebas y herramientas para facilitar la adaptación y control de errores, dejando como siguiente paso el inicio de aplicar el nuevo flujo tras un proceso de aprendizaje y adaptación de un mes.

Conclusiones

El presente informe muestra todas las actividades realizadas para la elaboración de un proceso de CI/CD. En este proceso se realizo la investigación e implementación de un proceso de integración t despliegue continuo para la empresa y posterior redacción de informe final, durante el proceso se llegó a las siguientes conclusiones y recomendaciones:

1. La entidad previa a este proceso no contaba con un conjunto de pruebas lo que no permite una medición directa, sin embargo, el control y capacidad de conocer el estado del código, reduce la incertidumbre que se tenia previo a la implementación del proceso.
2. El cambio de metodología del proceso tradicional de código a la metodología TDD, implica un cambio interesante y retador en un inicio, el cual se refina con el tiempo y practica.
3. Para la implementación de la codificación mediante la metodología TDD, se parte de preparar el conjunto de pruebas que satisfagan los criterios de aceptación de una funcionalidad, para posteriormente elaborar el código necesario para cumplir con las pruebas realizadas, se refactoriza el código para su optimización y se repite el proceso, hasta que se cubran la totalidad de los criterios de aceptación de la funionalidad.
4. El proceso de CI/CD, permite librar a los desarrolladores de todo el proceso de despliegue y en su mayoría de pruebas, al automatizar estos procesos, permitiendo el ahorro de tiempo, un aumento de confiabilidad y reducción de la incertidumbre de los posibles fallos en el entorno de producción.
5. El contar con pruebas que cubran el código implementado, generan mayor confianza, seguridad y estabilidad del código, reduciendo la necesidad de llevar a cabo hotfix una vez se despliega una nueva funcionalidad en producción, así mismo evitan que nuevas funcionalidades afecten anteriores funcionalidades, ya que estas implicaciones serian detectadas previas a despliegues a producción, permitiendo así la elaboración de los cambios correspondientes.

Mateo García Córdoba
Asesor externo

Roberto Florez Rueda
Asesor Interno