# Capítulo I: Preliminares

## Portada

Instituto Tecnológico José Mario Molina Pasquel y Henríquez Campus Chapala

Ing. Sistemas Computacionales



QA Automation Initiative

Pedro Solórzano Gómez

Carmen Leticia Salcedo Quevedo

El Salto Jalisco a, a 30 de octubre de 2020.

## Agradecimientos

## Resumen

## Índice

[Capítulo I: Preliminares 1](#_Toc57156953)

[Portada 1](#_Toc57156954)

[Agradecimientos 2](#_Toc57156955)

[Resumen 3](#_Toc57156956)

[Índice 4](#_Toc57156957)

[Capítulo II: Generalidades del proyecto 6](#_Toc57156958)

[Introducción 6](#_Toc57156959)

[Descripción de la empresa u organización y del puesto o área de trabajo del estudiante 7](#_Toc57156960)

[Problemas a resolver. 8](#_Toc57156961)

[Objetivos (Generales y específicos) 9](#_Toc57156962)

[Objetivo General 9](#_Toc57156963)

[Objetivos Específicos 9](#_Toc57156964)

[Justificación 10](#_Toc57156965)

[Capítulo III: Marco teórico 11](#_Toc57156966)

[Marco teórico 11](#_Toc57156967)

[ Integración Continua 11](#_Toc57156968)

[ Jenkins 11](#_Toc57156969)

[ JavaScript 11](#_Toc57156970)

[ Mocha 12](#_Toc57156971)

[ Apiary 12](#_Toc57156972)

[ Dredd 12](#_Toc57156973)

[ Perl 12](#_Toc57156974)

[ Selenium 12](#_Toc57156975)

[ DynamoDB 12](#_Toc57156976)

[ Allure Report 12](#_Toc57156977)

[ Axios 12](#_Toc57156978)

[ Chai 12](#_Toc57156979)

[Capitulo IV: Desarrollo 13](#_Toc57156980)

[Procedimiento y descripción de las actividades realizadas 13](#_Toc57156981)

[Capítulo V: Resultados 14](#_Toc57156982)

[Resultados, planos, gráficas, prototipos, manuales, programas, análisis estadísticos, modelos matemáticos, simulaciones, normatividades, regulaciones y restricciones, entre otros 14](#_Toc57156983)

[Actividades sociales realizadas en la empresa u organización 15](#_Toc57156984)

[Capítulo VI: Conclusiones 16](#_Toc57156985)

[Conclusiones de proyecto, recomendaciones y experiencia personal profesional adquirida 16](#_Toc57156986)

[Capitulo VII: Competencias desarrolladas 17](#_Toc57156987)

[Competencias desarrolladas y/o aplicadas 17](#_Toc57156988)

[Capitulo VIII: Fuentes de información 18](#_Toc57156989)

[Fuentes de información 18](#_Toc57156990)

[Capitulo IX: Anexos 19](#_Toc57156991)

# Capítulo II: Generalidades del proyecto

## Introducción

## Descripción de la empresa u organización y del puesto o área de trabajo del estudiante

Green Marimba I / O es una agencia de contratación de personal y subcontratación de Tecnologías de la Información centrada en los servicios en la nube de AWS, la automatización de pruebas y todas las áreas de ciberseguridad.

Green Marimba I / O fue fundada por un equipo experimentado de ejecutivos de Tecnologías de la Información que comprenden los desafíos de administrar el personal de TI remoto. Con sede en Boulder, Colorado, y oficinas en Los Ángeles y México.

El área en la que estuve realizando mi estancia profesional fue en el área de QA (Control de Calidad), esta área es la encargada de realizar pruebas al código desplegado por parte de los desarrolladores.

En esta área se hace el desarrollo de pruebas automatizadas, desde pruebas de Backend como pueden ser pruebas de API’s hasta pruebas automatizadas de Frontend como lo son pruebas de UI.

## Problemas a resolver.

En el transcurso de la residencia surgieron varios problemas y conforme avanzaba el tiempo y me iba familiarizando más con las herramientas y metodologías que utilizaban pude empezar a resolverlos. Los siguientes problemas fueron los que se me presentaron en la realización del proyecto de residencia:

* Dificultad con el lenguaje de programación utilizado.
* Tecnologías desconocidas para la realización del proyecto.
* Desconocimiento de las diferentes etapas del desarrollo de software.

## Objetivos (Generales y específicos)

## Objetivo General

* Desarrollar un Framework de control de calidad automatizado para los lenguajes Java y JavaScript que soporte la generación de reportes además de la integración de procesos de CI/CD.

## Objetivos Específicos

* Desarrollar una infraestructura en la nube, así como un Framework que soporte los lenguajes de programación: Java y JavaScript.
* Integrar procesos de desarrollo de tipo CI/CD a la infraestructura anteriormente mencionada.
* Automatizar la generación de reportes acorde a los resultados de ejecución de pruebas realizadas.

## Justificación

El presente proyecto permitirá a los usuarios integrar pruebas complejas de Control de Calidad en procesos de CI/CD en dos lenguajes de programación diferentes (Java y JavaScript). Con la posibilidad de generar reportes automatizados con los resultados de dichas pruebas ejecutadas.

Esta herramienta podrá ser implementada en cualquier proyecto con fines de automatizar las pruebas de calidad siempre y cuando cumpla con el tipo de lenguaje anteriormente mencionado.

Integrando desde pruebas de Backend utilizando un framework de pruebas como lo es Mocha. Hasta pruebas de Frontend utilizando Selenium web driver.

# Capítulo III: Marco teórico

## Marco teórico

En esta sección se describirán las herramientas o practicas utilizadas a lo largo del desarrollo de dicho proyecto, se añadirá una pequeña descripción de la herramienta además de una explicación del uso de la herramienta en el proyecto.

### Integración Continua

La integración continua es una práctica de desarrollo de software mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. La integración continua se refiere en su mayoría a la fase de creación o integración del proceso de publicación de software y conlleva un componente de automatización. Los objetivos clave de la integración continua consisten en encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones de software. (Amazon Web Services, 2020).

### Jenkins

Jenkins es un servidor automatizado de integración continua de código abierto y actualmente uno de los más utilizados por su fácil manejo y extensa integración con diferentes tecnologías. Además de su amplio catálogo de complementos que se pueden descargar directamente de su web.

En el proyecto se utilizó Jenkins como servidor en el proceso de Integración Continua; Se creo un pipeline para integrar desde la obtención del código fuente del proyecto desde el repositorio de control de versiones en GitHub como las pruebas realizadas para Frontend y Backend, hasta el despliegue de la aplicación en AWS.

### JavaScript

JavaScript es un lenguaje de programación de lado del cliente, utilizado mayormente para el desarrollo web, no requiere de compilación ya que los navegadores son los encargados de interpretar el código lo que lo hace multiplataforma.

JavaScript fue el lenguaje que se utilizo para las pruebas automatizadas realizadas, se utilizo este lenguaje debido a su fácil integración con el proyecto y a otras herramientas necesarias para la elaboración de las pruebas, como el framework de pruebas y a sus diferentes estilos de aserción, así como también su sencilla integración con la herramienta utilizada para el reporte de las pruebas.

### Mocha

Mocha es un framework de pruebas para JavaScript ejecutado en Node.js y el navegador, con este framework es posible hacer pruebas asíncronas y síncronas de una manera simple. Además de ofrecer muchas utilidades para la ejecución y reporte de las pruebas.

Este fue el framework de pruebas utilizado para la construcción de las pruebas, se utilizó este framework por su fácil integración con otras librerías de JavaScript además de su capacidad para crear pruebas asíncronas, y su integración con la librería de aserción utilizada.

### Apiary

Apiary es un editor para APIs que permite usar tanto API Blueprint como Swagger, los cuales son lenguajes que permiten describir nuestras APIs, y gracias a ellos se puede realizar algunas tareas como simular una API de forma sencilla y rápida.

Apiary no solo permite editar texto como cualquier otro editor, sino que también ayuda a pasar pruebas a nuestra API empleando Dredd. También analiza las peticiones con su inspector, adema permite previsualizar el documento final y compila constantemente con los últimos cambios dejando ver errores al validarlo todo. (Campos Romero, 2020)

Apiary fue la herramienta que se utilizo para hacer la documentación de la API. Se utilizo esta herramienta por su facilidad para documentar, y por la facilidad que tiene esta para integrarse con otra herramienta de pruebas de documentación de APIs. Otra de las razones por la que se utilizo esta herramienta es por su facilidad de simular el comportamiento de la API sin tener que programarla antes.

### Dredd

Dredd es una herramienta para la validación de documentos de descripción de APIs. Dredd revisa el documento de descripción de la API y paso a paso valida que funcione de la manera en que se especifica en el documento de descripción. Dredd soporta dos formatos de documentos de descripción: API Blueprint y Open API 2.

Se utilizo Dredd como herramienta para verificar el documento de descripción de API por su gran compatibilidad con Apiary y por su fácil ejecución de pruebas, tan solo utilizando unos comandos.

### Perl

Perl es un lenguaje de programación, utilizado mayormente para procesamiento de texto, para escanear archivos de texto y extraer información de ellos.

En la elaboración del proyecto se utilizo Perl para realizar un script, puesto que se necesitaba un pre procesador de texto que inyectara texto obtenido de un fichero en especifico dentro de otro fichero.

### Selenium

Selenium es un conjunto de herramientas disponibles en varios lenguajes de programación (Java, Python, C#, Ruby, JavaScript, Kotlin). Estas herramientas sirven para controlar el navegador web, simulando la interacción de un usuario con una página web. Dichas herramientas son utilizadas para la realización de pruebas automatizadas.

Para el desarrollo de las pruebas automatizadas de la Interfaz de Usuario se utilizó Selenium WebDriver, ya que esta herramienta es la mas utilizada para dicho propósito, además de su buena integración con otras herramientas necesarias para la correcta ejecución de las pruebas.

### DynamoDB

Amazon DynamoDB es una base de datos de clave-valor y documentos que ofrece rendimiento en milisegundos de un solo dígito a cualquier escala. Se trata de una base de datos duradera de varias regiones y con varios maestros, completamente administrada, que cuenta con copia de seguridad, restauración y seguridad integradas, y almacenamiento de caché en memoria para aplicaciones a escala de Internet. DynamoDB puede gestionar más de 10 billones de solicitudes por día y puede admitir picos de más de 20 millones de solicitudes por segundo. (Amazon Web Services, 2020)

Este fue la base de datos elegida para la elaboración del proyecto puesto que Amazon Web Services es el servicio de computación en la nube preferido de la empresa.

### Allure Report

Allure Report es una herramienta flexible, ligera y multilenguaje para el proceso de elaboración de los reportes de pruebas, ofrece reportes gráficos muy claros sobre la ejecución de las pruebas.

Allure fue la herramienta para la publicación de los reportes de pruebas utilizada durante el proyecto, por su sencilla incorporación con Jenkins el servidor de integración continúa elegido para la ejecución de este proyecto.

### Axios

Axios es una librería JavaScript que puede ejecutarse en el navegador y que nos permite hacer sencillas las operaciones como cliente HTTP basada en promesas, por lo que podremos configurar y realizar solicitudes a un servidor y recibiremos respuestas fáciles de procesar. (Baquero García, 2020).

Se utilizo Axios como la librería específica para la ejecución de peticiones HTTP a los diferentes endpoints de la API a la que se le realizaron las pruebas, se eligió esta librería por su compatibilidad con promesas.

### Chai

Chai.js es una librería de JavaScript que ofrece tres tipos diferentes de aserciones para la captura de resultados en las pruebas, además de ofrecer compatibilidad con cualquier framework de pruebas de JavaScript.

Para las aserciones en los casos de pruebas utilizamos esta librería puesto que ofrece tres diferentes tipos de aserción y una excelente compatibilidad con el framework de pruebas utilizado.

### Git

Git es una herramienta de control de versiones, el propósito de esta herramienta es llevar un registro de los cambios que se realizaron en cada fichero del proyecto y quien realiza dichos cambios.

Para los proyectos de tecnologías de la información es indispensable utilizar una herramienta para llevar acabo el control de versiones, puesto que es importante tener un control de cada cambio que se realiza en el proyecto. Por esta razón se utilizo Git para el control de versiones además de ser la herramienta mas utilizada mundialmente para esta finalidad.

### GitHub

GitHub

### Allure

# Capitulo IV: Desarrollo

## Procedimiento y descripción de las actividades realizadas

# Capítulo V: Resultados

## Resultados, planos, gráficas, prototipos, manuales, programas, análisis estadísticos, modelos matemáticos, simulaciones, normatividades, regulaciones y restricciones, entre otros

## Actividades sociales realizadas en la empresa u organización

# Capítulo VI: Conclusiones

## Conclusiones de proyecto, recomendaciones y experiencia personal profesional adquirida

# Capitulo VII: Competencias desarrolladas

## Competencias desarrolladas y/o aplicadas

# Capitulo VIII: Fuentes de información

## Fuentes de información

# Capitulo IX: Anexos