INSTITUTO TECNOLÓGICO SUPERIOR DE ÁLAMO TEMAPACHE

TITULACIÓN POR TESIS

SOFTWARE QUALITY ASSURANCE

PRESENTA

MIGUEL HUGO ALCÁNTARA PÉREZ

PARA OBTENER EL TITULO DE:

INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

ASESOR

YURIDIA EUSEBIA SANTOS CRUZ

XOYOTITLA, ÁLAMO TEMAPACHE, VER. FEBRERO 2023

DEDICATORIA

Le dedico esta tesis de manera especial a mis padres por haberme forjado como la persona que soy en la actualidad; muchos de mis logros se debe a que ustedes me han apoyade de una manera especial. Ellos son los que con su cariño me han impulsado siempre a perseguir mis metas y nunca abandonarlas frente a las adversidades, me formaron con reglas y con algunas libertades, pero, me motivaron constantemente para alcanzar mis anhelos.

AGRADECIMIENTOS

A mi tutor

“Le agradezco muy profundamente a mi tutor por su dedicación y paciencia, sin sus palabras y correcciones precisas no hubiese podido lograr llegar a esta instancia tan anhelada. Gracias por su guía y todos sus consejos, los llevaré grabados para siempre en la memoria en mi futuro profesional”.

A todos mis docentes

“Son muchos los docentes que han sido parten de mi camino universitario, y a todos ellos les quiero agradecer por transmitirme los conocimientos necesarios para hoy poder estar aquí. Algunos maestros se recuerdan con admiración por su forma de enseñar, otros maestros se recuerdan con cariño y gratitud por su forma de tocar los corazones de sus alumnos. La huella que han dejado, quedará en nuestra mente y corazón. Nos impulsará en el camino de la vida y nuestros pasos siempre estarán guiados por sus enseñanzas”.

Además, a mis compañeros

“Agradecerles a todos mis compañeros los cuales algunos de ellos se han convertido en mis amigos. Gracias por las horas compartidas, los trabajos realizados en conjunto y las historias vividas”

Gracias a la casa de estudios

“Por último, pero no menos importante; agradecer a la universidad que me ha exigido tanto, pero al mismo tiempo me ha permitido obtener mi tan ansiado título. Agradezco a cada directivo por su trabajo y por su gestión, sin lo cual no estarían las bases ni las condiciones para aprender conocimientos, agradezco su gran apoyo y dedicación que tienen con todos los alumnos”.

RESUMEN.

Con la definición e implementación de los conceptos que nos da ISTQB (Intertional Software Testing Qualifications Board o El Comité Internacional de Certificación de Pruebas de Software) basados en los conocimientos básicos de CTFL (Certified Tester Foundation Level o Certificación de probador Nivel Básico) de un proceso de Quality Assurance en una empresa, en este proyecto se abarcará en relación al desarrollo de software, evita a gran medida que el cliente detecte fallos y pierda la confianza del producto. Siempre es más rentable económicamente y en cuestiones temporales, detectar defectos en fases tempranas de desarrollo y la corrección del mismo.

En este trabajo, definimos e implementamos un proceso de QA tanto Automatizado como Manual, dentro de un entorno real, consta de la automatización de la página web buscador de hoteles, que tienen como finalidad de obtener los diversos sitios donde el cliente se pueda hospedar, para asegurar la calidad del software, se valida que; de acuerdo a la fecha de entrada y de salida tome el intervalo de tiempo y arroje el costo total por el hospedaje.

Se analiza la página web para comprender el comportamiento del mismo, ejecutando valores de entrada de manera aleatoria para ver la ruta crítica y así mismo saber qué código se necesita para este proyecto.

Esta problemática se soluciona con diferentes herramientas de programación que nos ayudará a ejecutar diversas pruebas de una manera automática y un sistema de control de versiones tales como; Git y GitHub. Se toma en cuenta que el código implementado será con el lenguaje de programación Java, framework tales como; Cucumber, Serenity y Selenium junto con el IDE Intellij. Se necesita tener conocimientos esenciales sobre un Quality Assurance tanto manual como automatizado.

ABSTRACT.

With the definition and implementation of the concepts given to us by ISTQB (Intertional Software Testing Qualifications Board) based on the basic knowledge of CTFL (Certified Tester Foundation Level or Basic Level Tester Certification) of a Quality Assurance process in a company, in this project will be covered in relation to software development, largely prevents the customer from detecting failures and losing product confidence. It is always more profitable economically and in temporal matters, to detect defects in early stages of development and the correction of it.

In this work, we define and implement a QA process both Automated and Manual, within a real environment, consists of the automation of the hotel search website, which aim to obtain the various sites where the customer can host, to ensure the quality of the software, it is validated that; according to the date of entry and exit take the time interval and throw the total cost for hosting.

We analyze the website to understand its behavior, running input values randomly to see the critical path and also know what code is needed for this project.

This problem is solved with different programming tools that will help us to run various tests in an automatic way and a version control system such as; Git and GitHub. Note that the code implemented will be with the Java programming language, framework such as; Cucumber, Serenity and Selenium along with the IDE Intellij. You need to have essential knowledge about both manual and automated Quality Assurance.

ÍNDICE.

DEDICATORIA ii

AGRADECIMIENTOS ii

RESUMEN. iv

ABSTRACT. v

ÍNDICE. vi

CAPÍTULO I. GENERALIDADES DEL PROYECTO 9

1. Introducción 9

1.1 Antecedentes 9

1.2 Planteamiento del problema 11

1.3 Justificación 12

1.4 Hipótesis 12

1.5 Objetivo 13

- Objetivo general 13
- Objetivo particular 13

CAPÍTULO II. MARCO TEÓRICO 14

CAPÍTULO III. ESTADO DEL ARTE 23

CAPÍTULO IV. METODOLOGÍA 25

CAPÍTULO V. ANÁLISIS Y DISCUSIÓN DE RESULTADOS 45

CAPÍTULO VI. CONCLUSIONES 48

FUENTES DE INFORMACIÓN 49

ANEXOS 53

Índice de Figuras

Figura 1.1 Planteamiento del problema (SpeedUpTech, 2022) 25
3.2 Planteamiento del problema 26
Figura 2.1 Patrón de diseño ScreenPlay (Villa, A. 2020.) 28
3.3.1 Control de versiones con Git y GiHub 28
Figura 2.1.1 Creación del repositorio 29
Figura 2.1.2 Clonar el repositorio 29
3.3.2Estructura del proyecto 30
Figura 2.2 Implementación del ScreenPlay (Elaboración propia, 2022.) 30
3.3.3 Configuración del *serenity* 30
Figura 2.2.1 Configuración del serenity.properties (Elaboración propia, 2022.) 30
3.3.4 Dependencias y otras propiedades 31
Figura 2.2.2 Configuración de las dependencias (Elaboración propia, 2022.) 31
3.3.5 Feature 32
Figura 2.2.3 Creación de la Feature (Elaboración propia, 2022.) 32
3.4 Creación del código 32
Figura 2.2.4 Creación del runner (Elaboración propia, 2022.) 33
Figura 2.2.5 Creación de los Step definition (Elaboración propia, 2022.) 33
Figura 2.2.6 Configuración de los step definition(Elaboración propia, 2022.) 34
Figura 2.2.7 Creación del cierre del escenario (Elaboración propia, 2022.) 34
Figura 2.2.8 Creación de la tarea encargada de abrir el navegador (Elaboración propia, 2022.) 35
Figura 2.2.9 Implementación de la tarea en el step definition (Elaboración propia, 2022.) 35
Figura 2.2.10 Comparación de la feature con el navegador de buscar hoteles (Elaboración propia, 2022.) 36
Figura 2.2.11 Implementación de los localizadores (Elaboración propia, 2022.) 36
Figura 2.2.12 Implementación de los localizadores (Elaboración propia, 2022.) 37

Figura 2.2.13 Implementación los constructores y getters de acuerdo a los datos requeridos de la feature (Elaboración propia, 2022.) 37
Figura 2.2.14 Creación de la tarea encargada de hacer la búsqueda de los hoteles, recordando el día de entrada y de salida (Elaboración propia, 2022.) 38
Figura 2.2.15 Creación del método retornando la misma clase (Elaboración propia, 2022.) 39
Figura 2.2.16 Creación de una utilidad que nos sirve para obtener el resultado de la fecha de entrada y de salida en un valor entero (Elaboración propia, 2022.) 39
Figura 2.2.17 Implementación de la tarea en el setep definition (Elaboración propia, 2022.) 40
Figura 2.2.18 Creación de una tarea para que almacene todos los valores de los hoteles obtenidos en la primera página (Elaboración propia, 2022.) 41
Figura 2.2.19 Implementación de la tarea en el step definition (Elaboración propia, 2022.) 41
Figura 2.2.20 Creación de una question (Elaboración propia, 2022.) 42
Figura 2.2.20.1 Creación de una question (Elaboración propia, 2022.) 42
Figura 2.2.20.2 Creación de una question (Elaboración propia, 2022.) 42
Figura 2.2.21 Creación de una question que remplaza valores (Elaboración propia, 2022.)43
Figura 2.2.22 Implementación de las question en el step definition (Elaboración propia, 2022.) 43
Figura 2.2.23 Cierre del escenario (Elaboración propia, 2022.) 44
Figura 4.1 Resultados de las pruebas (Elaboración propia, 2022) 46
Figura 4.2 Resultados de las pruebas (Elaboración propia, 2022) 47

Índice de Tabla

Tabla 1.2 Casos de prueba (Elaboración propia, 2022).....27

CAPÍTULO I. GENERALIDADES DEL PROYECTO

1. Introducción

1.1 Antecedentes

La motivación para asegurar al cliente que el software cumple con la calidad deseada es definir y depurar un proceso de aseguramiento de calidad o “Quality Assurance” (QA) que permita una vez implantado en una empresa de desarrollo de software, asegure la calidad final del producto. Cabe destacar que este proceso no solo es beneficioso para el usuario final que recibe dicho producto, sino que también lo es para la empresa, ya que se establece una metodología de trabajo en la cual existe un control permanente sobre el proceso, evitando los altos costes que pueden suponer corregir fallos en etapas avanzadas de un proyecto concreto.

En la actualidad, muchas empresas tienden a aplicar estos procesos /metodologías para llevar a cabo el desarrollo de sus productos de software, cabe mencionar que, para que esta metodología sea un éxito es necesario la colaboración de cada uno de los departamentos de la organización y con ello es necesario una definición rigurosa del proceso QA que se lleva a cabo y la metodología SCRUM.

Este proyecto se ha desarrollado con la colaboración de un ecosistema empresarial que están dedicada en el desarrollo de productos y/o servicios de software donde se implementa una metodología ágil SCRUM siendo este un marco de trabajo que se adapta a diversos proyectos, el cual nos ayuda a entregar productos de máximo valor posible, y la estrategia de Desarrollo Dirigido por Comportamiento (BDD), donde en cada una de las entregas al cliente se le mostrará el valor agregado o un avance del proyecto, todo esto está definido por una serie de unidades de trabajo que contiene un conjunto de pruebas, tales como; Pruebas de sistema, Pruebas de integración, Pruebas de aceptación, entre otros. Cada unidad de trabajo, siguiendo la metodología SCRUM, donde se invierten distintos roles para realizar las diversas actividades que conforman dicha metodología (Desarrolladores, QA, Produc Owner, entre otros).

Estos roles y actividades, así como la gestión de diversas pruebas que conforman cada unidad de trabajo, hacen necesario definir de una manera exhaustiva un proceso que asegure la calidad del producto entregado al cliente.

El proyecto consta de la automatización de la página web buscador de hoteles, que tienen como finalidad de obtener los diversos sitios donde el cliente se pueda hospedar, para asegurar la calidad del software, se valida que; de acuerdo a la fecha de entrada y de salida tome el intervalo de tiempo y arroje el costo total por el hospedaje, teniendo como objetivo, validar que la página buscadora de hoteles esté funcionando correctamente y haciendo que los clientes tengan la seguridad de utilizar dicha página.

Por lo tanto, el punto de partida no es iniciar desde cero, ya que dicho ecosistema de empresas ha implementado la metodología donde tiene gran peso en el aseguramiento de calidad. Es importante destacar que la automatización de la página buscador de hoteles es para darle el aseguramiento de calidad a los clientes, haciendo pruebas con diversos datos de entrada y precondiciones, llevando todo esto al cliente para que le dé el visto bueno y decida si el software solicitado cumpla con los requisitos establecidos.

1.2 Planteamiento del problema

A nivel internacional, podemos observar que las empresas están implementando el aseguramiento de calidad en relación al desarrollo del software, esto quiere decir que la actualidad hay mucha demanda para aquellos que son QA y mucho más si están certificados por el ISTQB, dándole al cliente la mejor calidad del software cumpliendo con las normas y requerimientos contractuales.

En la actualidad es una problemática comprar boletos en tiendas de manera presencial, ya que en algunas ocasiones hay muchas filas para la compra de boletos y en otras no hay disponibilidad de boletos, es por eso que el cliente solicitó el software de una página web relacionada al servicio de hoteles que están situados en diferentes partes del país.

El alcance de este proyecto abarca en relación a las historias de usuario que conforma la épica: Se realizará casos de prueba tanto validos como inválidos de cada historia de usuario. Se realizará pruebas manuales, pruebas funcionales, pruebas de integración, pruebas de sistema, pruebas de aceptación, pruebas de caja negra y caja blanca. Dichas pruebas se harán con el fin de validad que el sistema y sus componentes estén funcionando correctamente, con ello podremos verificar que el sistema está libre de defectos.

Se realizará pruebas automatizadas porque tenemos el software como tal, con ello se utilizan varios métodos, se utilizará diversas herramientas. Con ello, se realizarán los casos de prueba de acuerdo a los criterios de aceptación que están en las historias de usuario, generando un reporte de cada uno de ellos, validando y verificando la funcionalidad del mismo.

Las limitaciones que tendremos son; No se realizará el código para la creación de dicho producto ya que este proceso lo engloba el equipo de desarrolladores. Otra de nuestras limitaciones son que en el área de QA no se realizan pruebas de aceptación, ya que estas pruebas la realizan el cliente final y él tomará la decisión si satisface las necesidades y los requerimientos contractuales. Las actualizaciones también será una limitación porque posiblemente pueden cambiar los localizadores y la estructura de la página web.

1.3 Justificación

La importancia de este proyecto es dar a conocer que las empresas necesitan implementar el aseguramiento de calidad para que los clientes tengan esa seguridad de que sus productos cumplirán con los requerimientos contractuales, esta implementación hace que las empresas tengan mayor nivel en cuanto a la calidad del software.

Al llevar a cabo un adecuado aseguramiento de calidad permite detectar defectos o desviaciones en el producto o servicio, corregirlos de manera oportuna y mejorar de manera continua, garantizando la satisfacción de los clientes y del mercado.

Todo esto engloba a La norma ISO/IEC 9126 que se encarga de evaluar los productos del software, indicando las características de calidad y sus métricas asociadas.

1.4 Hipótesis

Si nos basamos en el aseguramiento de la calidad del software, junto la implementación de las herramientas necesarias tales como: Cucumber, Selenium, Serenity, con IDE Intellij y aplicando el lenguaje de programación Java, esto hace función a la hora de evaluar la calidad de la página web buscador de hoteles, me da como resultado el hacer pruebas automatizadas de dicha página web con diversos valores de entradas y precondiciones, con esto, nos da como resultado la validación y la verificación de que la página web está funcionando de cierta manera, tales como: la página web está funcionando correctamente y no se detectó alguna falla, y el otro resultado sería que, en la página web se detectó alguna falla y se tiene que corregir lo antes posible. Sí alguna de estas fallas llega a producción que es cuando se lanza el producto a todos los clientes, este sufrirá una gran catástrofe, ya que no se hizo un buen uso de las normas y el aseguramiento de calidad que requiere, obteniendo pérdida de dinero, pérdida de clientes y tener una mala reputación empresarial, por ende, es necesario hacer las iteraciones o los sprints necesarios para eliminar los errores que muestra la página web.

Otra alternativa sería analizar la documentación de los antecedentes de otros proyectos para poder predecir los posibles errores, todo esto puede depender de la administración que

maneje la empresa, el ambiente laboral, y la experiencia que tengan los trabajadores de las diferentes áreas. Considere adecuando que los trabajadores tengan un status de experiencia en el área abordad y analizar el cómo puede solucionar la problemática planteada.

1.5 Objetivo

- **Objetivo general**

  Asegurar la calidad de una página web busca hoteles con diversas herramientas, tanto manuales como automatizadas y metodologías de desarrollo ágil incremental que se implementarán en el área de QA Automation de la empresa SpeedUpTech.

- **Objetivo particular**

  1- Analizar la ruta de la página que se va automatizar, tomando en cuenta el comportamiento del mismo.
  2- Crear historias de usuario junto con sus criterios de aceptación con la finalidad de implementar la documentación de los casos de prueba para pasarlos al lenguaje Gherkin que extiende de Cucumber y que se puedan implementar en las Features del proyecto.
  3- Realizar el código usando herramientas tales como: el lenguaje de programación Java, con el IDE (entorno de desarrollo integrado) Intellij, usando los principios SOLID y el patrón ScreenPlay, subiendo los avances y/o actualizaciones al repositorio remoto utilizando Git/GitHug.
  4- Ejecutar todos los steps definition en la terminal del IDE implementando el comando necesario para que arroje el informe y/o reporte.

CAPÍTULO II. MARCO TEÓRICO

2.1 International Software Testing Qualifications Board

El International Software Testing Qualifications Board nos sirve para darle valor en el campo del aseguramiento de calidad.

Según *Gandarillas, A. (ISTQB, 2002) nos dice que "Nació con el objetivo de definir un esquema de certificación internacional para la calidad del software. Permite formarse y certificarse en tres niveles: Foundation, Advanced y Expert."*

Esta certificación ayuda a darle una mayor fuerza al software ya que esta certificación se obtiene al hacer el examen mostrando las capacidades y conocimientos que se tiene del área de aseguramiento de calidad, esto le da un valor agregado al cliente y se le da la seguridad de que su producto lo están haciendo personas capacitadas y certificadas en el área.

2.1.1 Probar

A la hora de probar el producto estamos evaluando la calidad del software, con ello se reduce el riesgo de que falle en un ambiente de producción.

Como menciona *Gandarillas, A. (ISTQB, 2017) probar es el "proceso que consiste en todas las actividades del ciclo de vida software, tanto estáticas como dinámicas, concernientes con la planificación, preparación y evaluación determinar que éstos satisfacen los requerimientos especificados, para demostrar que se ajustan al propósito y para detectar defectos."*

Es importante destacar que es necesario probar la página web de hoteles porque con esto se asegura que el producto está funcionando correctamente, cumpliendo los requisitos contractuales, con ello se garantiza que esté libre de defectos.

2.1.2 Tipos de pruebas

Los tipos de prueba son las diferentes formas de probar un producto de software que tiene diferentes formas de implementar, pero teniendo un fin en concreto que es asegurar la calidad del producto.

Como menciona *Gandarillas, A. (ISTQB, 2014) los tipos de prueba son un "grupo de actividades de pruebas dirigidas a probar un componente o sistema orientado a un objetivo de prueba específico, por ejemplo, pruebas funcionales, pruebas de sistema, etc. Un tipo de prueba puede tener lugar en uno o más niveles de prueba o fases de prueba."*

Con estos tipos de pruebas podemos probar en diferentes áreas o componentes del software para probar; ya sea de un componente por uno o con todos los componentes juntos para tener un mayor control y probar con diferentes valores de entradas y salidas de acuerdo a las historias de usuarios y los criterios de aceptación.

2.1.2.1 Pruebas de integración

Las pruebas que se van a realizar son las de integración que se encargar de implementar diversos componentes.

Como menciona *Gandarillas, A. (ISTQB, 2017) las pruebas de integración son "pruebas realizadas con el objeto de poner en evidencia defectos en las interfaces e interacciones entre componentes o sistemas integrados."*

Estas pruebas de integración se realizan cuando hay varios componentes que depende uno del otro, en este caso el componente de la localización dependerá de qué lugar se va a escoger y con ello mostrar los hoteles que están disponibles en esa localización. Con ello podemos aclarar que un componente depende del otro y que se puede automatizar en conjunto.

2.1.2.2 Pruebas de aceptación

Estas pruebas de aceptación las realizaremos para validar que los requisitos contractuales estén cumpliendo correctamente, y el usuario dará la aprobación si se cumple correctamente.

Como menciona *Gandarillas, A. (ISTQB, 2017) las pruebas de aceptación son "pruebas formales con respecto a las necesidades del usuario, requisitos y procesos de negocio dirigidos a determinar si el sistema satisface o no los criterios de aceptación y a habilitar al usuario, cliente u otra entidad."*

Con estas pruebas el cliente utiliza el software para verificar que su producto se haya creado de la manera en la que él lo desea, dando el visto bueno en la corroboración de que todos sus requisitos contractuales se cumplan correctamente. Así el cliente decide si se lleva el producto o no.

2.1.2.3 Pruebas de sistema

Las pruebas de sistema son pruebas que se le hacen al sistema general para validar que está libre de defectos y que cumple con los requisitos establecidos.

Como menciona *Gandarillas, A. (ISTQB, 2017) "probar un sistema integrado para verificar que cumple con los requisitos especificados."*

2.1.3 Casos de prueba

Los casos de prueba son un conjunto de precondiciones que se necesita en el software para que satisfaga las necesidades del cliente.

Como menciona la *norma IEEE 610 (2017) son un "conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y postcondiciones de ejecución, desarrollado con un objetivo en particular o condición de prueba, tales como probar un determinado camino de ejecución o para verificar el cumplimiento de un requisito determinado."*

Estos casos de prueba se definen entre el Produc Owner y el cliente ya que son las precondiciones que debe cumplir el sistema, dichas precondiciones se pasan al lenguaje gherkin y así poder convertirlo en un lenguaje más cotidiano y entendible para cualquier persona. Estos casos de prueba son esenciales porque nos ayuda a tener mapeado el alcance de la historia de usuario.

2.1.4 Ciclo de vida del desarrollo del software y sus normas

El ciclo de vida de desarrollo del software es el ciclo o recorrido que debe realizar el producto antes de entregarlo al cliente final, consta de diversas fases de acuerdo a qué modelo esté implementando la empresa.

Como menciona *Milano, P. (2007) "contempla las fases necesarias para validar el desarrollo del software y así garantizar que este cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo, asegurándose de que los métodos usados son apropiados."*

Según la *norma ISO 12270 (ISTQB, 2017) "Es el estándar para los procesos del ciclo de vida del software."*

Este ciclo se usa para crear las historias de usuario, casos de prueba y con base a eso se hará la automatización, una vez que se haya terminado esa iteración se procede a implementar las otras historias de usuario, ya que la metodología es de desarrollo ágil e incremental, si uno de nuestras historias de usuario no se finaliza en el tiempo programado se espera para el siguiente sprint porque en SCURM se entrega el software más alto para el negocio en el mejor precio.

2.2 Quality Assurance

Un QA es aquel que se encarga del aseguramiento de calidad de un producto implementando diversas metodologías de acuerdo al marco de trabajo de la empresa.

Como menciona *Hiberus, (2021) El aseguramiento de la calidad (Quality assurance, QA) es el "conjunto de actividades planificadas y sistemáticas aplicadas en un sistema de gestión de calidad para que los requisitos de calidad de un producto o servicio sean satisfechos."*

Siendo una persona encargada del aseguramiento de calidad, se tiene como objetivo hacer diversas pruebas al software con diversas herramientas para darle calidad al producto, En el área de QA manual, se realiza toda la documentación tales como; plan de prueba, historias de usuario, casos de prueba, reporte de posibles bugs, entre otros.

2.2.1 Quality Assurance Automation

Un QA Automation se encarga de realizar pruebas de manera automatizada con el fin de optimizar tiempo, haciendo la iteración de manera ágil.

Como menciona *Tripathy, P., & Naik, K. (2011). Se refiere a "una pieza separada de software para correr pruebas sobre una aplicación que tu equipo esté desarrollando."*

Es lo mismo que un QA manual donde se tiene que implementar cierta documentación, pero en el QA Automation ya se tienen que realizar código y/o ejecución iterativa de ciertos componentes que se requiera asegurar la calidad.

2.2.2 SCRUM

SCRUM es un marco de trabajo ágil que implementan las empresas para entregar un producto de calidad en un corto periodo de tiempo.

Como menciona *Rodríguez, C., & Dorado, R. (2015) "Es un marco de trabajo para desarrollo ágil de software que se ha expandido a otras industrias. Adopta una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto."*

En SCRUM que es la metodología es de desarrollo ágil e incremental, se implementa de la siguiente manera; si uno de nuestras historias de usuario no se finaliza en el tiempo programado se espera para el siguiente sprint porque en SCURM se entrega el software más alto para el negocio en el mejor precio. Si la historia de usuarios completó de acuerdo con el

tiempo establecido se procede a realizar el otro sprint y de esa manera se hace de forma iterativa e incremental

2.2.3 Requisitos

Los requisitos son las condiciones que pide un cliente para la funcionalidad del software.

Como menciona *Gandarillas, A. (ISTQB, 2017) los requisitos son una "condición o capacidad necesaria para un usuario con el objeto de solucionar un problema o lograr un objetivo que debe ser alcanzado o poseído por un sistema o componente de un sistema, para satisfacer un contrato, estándar, especificación u otro documento impuesto formalmente."*

2.2.4 Feature o Característica

La feature es, como su nombre lo indica, una característica o funcionalidad que tiene un conjunto de casos de prueba, la cual nos ayudará para realzar la ruta que se requiere al sistema.

Como menciona *Caroli, P. (2020) una característica (feature) "es una unidad funcional de un sistema de software que satisface un requisito, representa una decisión de diseño, y puede generar una opción de implementación."*

De acuerdo a nuestros casos de prueba se convierte al lenguaje gherkin que esto nos da una estructura de palabras clave para poder describir con lenguaje más cotidiano, las funcionalidades que se requiere realizar, dicho lenguaje busca ser declarativo y entendible por todas las personas.

2.3 Software

Un software es un sistema informático que se encarga de realizar tareas determinadas de acuerdo a las necesidades del cliente.

Como menciona *Pérez, P. J. y Gardey, A. (2021), un software es un "sistema informático que comprende del conjunto de los componentes lógicos necesarios que hacen posible la relación de tares específicas."*

El software puede ser desde una aplicación, un programa, pero en este caso, el software será de una página web dedicada la búsqueda de hoteles.

2.3.1 Variable de entorno

Una variable de entorno es una herramienta que afecta el comportamiento de los procesos ya sea a nivel de sistema o a nivel de usuario.

Como menciona *Merino, M. (2020) "es una variable dinámica que puede afectar al comportamiento de los procesos en ejecución en un ordenador. Son parte del entorno en el que se ejecuta un proceso."*

Esta variable de entorno es necesaria porque a la hora de crear el proyecto para la automatización, es necesario que nuestro IDE identifique que tenemos una variable llamada Maven y Java para que el IDE pueda implementar la funcionalidad, y así poder acceder a los recursos de dichas variables de entorno.

2.3.2 *API*

Una *API* es un conjunto de funciones y procedimiento que ofrecen ciertas bibliotecas para ser utilizadas.

Según *AWS (2022) las API "son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos."*

2.3.3 Principios *SOLID*

Los principios *SOLID* son una serie de reglas básicas de la programación orientada a objetos con el fin de obtener un código más eficiente y fácil.

Como menciona *Norberto, A. M. (2020). SOLID es un acrónimo acuñado por Robert C.Martin en el cual se representan los cinco principios básicos de la programación orientada a objetos. La intención de seguir estos principios es eliminar malos diseños, evitar la refactorización y construir un código más eficiente y fácil de mantener.*

Estos principios son de suma importancia porque nos ayuda a tener una mejor estructura, las carpetas están bien distribuidas y el código hace que tenga responsabilidades únicas de acuerdo al interfaz que se está automatizando. La S significa que las clases deben tener una y sólo una única responsabilidad. La O indica que las clases deben estar abiertas para extenderse, pero cerradas para modificarse. La L significa que las clases derivadas deben poder sustituirse por sus clases base. La I indica que se tiene que hacer interfaces para una

finalidad concreta y la letra D significa que las clases depende de la abstracción, no de las clases concretas.

2.3.4 Entorno de desarrollo integrado *IDE*

El *IDE* es un entorno de desarrollo integrado que nos permite implementar la programación con diversos lenguajes, dándonos una interfaz agradable y entendible.

Como menciona *Luna, E. (2019) "es una herramienta de software que proporciona un entorno de programación completo para los desarrolladores de software. Este conjunto de herramientas es utilizado para ayudar al desarrollo de software desde un mismo techo."*

2.3.5 *Selenium*

Selenium es un software que nos permite realizar pruebas automatizadas basadas en la web.

Como menciona *Anaya, A. (2020, May 20) "Es una herramienta de código abierto para la automatización de pruebas, de navegadores web."*

2.3.6 *Serenity BDD*

Serenity BDD es una herramienta que nos ayuda a realiza r pruebas de aceptación para un producto automatizable.

Como menciona *Villa, A. (2020) "Es una librería de código abierto que ayuda a escribir pruebas de aceptación automatizadas de mayor calidad y de manera más eficiente."*

2.3.7 *BDD*

BDD es una metodología dirigida por comportamiento del sistema y el usuario, usando el lenguaje *Cucumber*, se combinan aspectos técnicos como de negocio.

Como menciona Sergio, V. (2019) es una estrategia de desarrollo dirigido por comportamiento, Para definir los casos BDD para una historia de usuario se deben definir bajo el patrón 'Given-When-Then', que se define como:

- *Given 'dado': Se especifica el escenario, las precondiciones.*
- *When 'cuando': Las condiciones de las acciones que se van a ejecutar.*
- *Then 'entonces': El resultado esperado, las validaciones a realizar.*

2.3.8 *Cucumber*

Cucumber (Pepinillo) Es una herramienta basada en BDD, para la escritura de pruebas de aceptación, permite escribir el lenguaje en un formato legible y comprensible, para cualquier analista de negocio, probador, desarrollador, etc.

Como menciona *Mari, L. (2020) "es una herramienta de software que apoya el desarrollo impulsado por el comportamiento. En el enfoque de Cucumber BDD es fundamental su analizador de lenguaje ordinario llamado Gherkin. Permite especificar los comportamientos esperados del software en un lenguaje lógico que los clientes puedan entender."*

2.3.9 Git y GitHub

Git es un sistema de control de versiones que nos permite controlar y gestionar los avances de un proyecto, GitHub es una herramienta que nos ayuda al almacenamiento de repositorios en la nube.

Como menciona *Castellanos, E. (2020, febrero 14) "Git es el control de versiones, se refiere al proceso de guardar diferentes archivos o «versiones» a lo largo de las diferentes etapas de un proyecto. Esto permite a los desarrolladores hacer un seguimiento de lo que se ha hecho y volver a una fase anterior si deciden que quieren revertir algunos de los cambios que han hecho. GitHub facilita la colaboración con Git. Es una plataforma que puede mantener repositorios de código en almacenamiento basado en la nube para que varios desarrolladores puedan trabajar en un solo proyecto y ver las ediciones de cada uno en tiempo real."*

2.3.10 Xpath y localizadores

Los *Xpath* son una forma de localizar objetos, funciones o atributos de una página web.

Como menciona *Javier, M. V. (2009), "Es el sistema que se utiliza para navegar y consultar los elementos y atributos contenidos en la estructura de un documento XML."*

Por las buenas prácticas y la obtención de un código decente, se busca tener los Xpath con menor carácter, que sean únicos y que la forma de localizarlo sea eficiente.

2.3.11 Maven

Maven es una variable de entorno y gestor de dependencia que se requiere para la funcionalidad de diversas herramientas y códigos que se implementará a lo largo de la automatización.

Como menciona, Alarcón. J. M. (2022) *"Es una potente herramienta de gestión de proyectos que se utiliza para gestión de dependencias, como herramienta de compilación e incluso como herramienta de documentación. Es de código abierto y gratuita."*

CAPÍTULO III. ESTADO DEL ARTE

Para entender mejor el contexto nos enfocaremos en la descripción de diferentes trabajos y documentos que están directamente relacionados con un método de aseguramiento de la calidad para el desarrollo de software, de esta manera, se intenta recabar información de diferentes procesos que de una u otra manera intentan asegurar la calidad en el transcurso del desarrollo de un sistema.

El trabajo presentado por (E. Diez, 2013), presenta un conjunto de acciones y métodos de ACS. El modelo general de ACS, se documenta en un plan general de aseguramiento de la calidad, flexibilidad que a través de diferentes fases y módulos ayuda a desarrollar un producto de calidad, permitiendo la adaptación del modelo genérico o plan general de aseguramiento de la calidad del software, a todo tipo de proyectos. Se formaliza a través de los planes específicos de aseguramiento de la calidad del software para cada proyecto. Este enfoque no presenta ser exclusivo y en ningún caso limita o inhibe la aplicación de otras acciones, métodos o modelos, sino que podría ser su complemento, adaptándolo convenientemente. Cabe destacar, que esta propuesta identifica un equipo especializado en el aseguramiento de calidad dentro de la empresa, encargado de realizar todas las tareas de SQA.

Por una parte, el trabajo de (G. Aguirre, 2015), a pesar de que no es un método o plan de aseguramiento de la calidad, su objetivo es el mismo: satisfacer los requerimientos del usuario a través de una calidad esperada. Este marco metodológico, está basado en diferentes estándares de calidad, que a su vez le proporcionar una serie de características para poder guiar un desarrollo de software de forma exitosa. Cabe destacar que este marco está desarrollado pensando en lo complicado que es para las pymes aplicar métodos o estándares de calidad, debido principalmente a aspectos relacionados con su complejidad, costos, restricciones etc. Es por ello, que se crea este método de V&V para beneficiar a las empresas desarrolladoras de software en Perú, el desarrollo de sus proyectos de software.

Por otra parte, el trabajo de (G. Paladines, 2014), describe un plan de aseguramiento de la calidad a través de diferentes actividades de verificación con checklist, de las diferentes etapas del desarrollo del proyecto, que no hace distinción entre el método de desarrollo y la estrategia de aseguramiento de la calidad. Este plan de gestión de la calidad, solo se encarga de controlar si se cumplen con los requisitos al final de cada etapa del mismo y no es una guía de aseguramiento de la calidad para el desarrollo del proyecto, por lo tanto, solo es un control de calidad.

Y, por último, en (G. Gómez, 2014) se presenta una revisión sistemática de la literatura acerca del tópico: adaptaciones en la mejora del proceso software (SPI) en las MiPyMEs, en el período comprendido de 1995 a diciembre de 2013. Su objetivo es, presentar información actualizada sobre las tendencias de este tópico como son países y sectores que abordan el tema, así como modelos, metodologías, estándares, y procesos de soporte del área de calidad reportados en este tipo de empresas. Esta revisión sistemática se centra en aportaciones reportadas sobre procesos de soporte del área de calidad del ciclo de vida del software.

CAPÍTULO IV. METODOLOGÍA

3.1 Análisis de la problemática

Lo primero que se debe analizar es el problema planteado en dichos proyectos, ya que estamos hablando de *front-end* se interactúa con una página web, todo esto se implementará utilizando la metodología SCRUM con el ciclo de vida incremental, el planteamiento del problema consta de lo siguiente:

Carolina es la jefe de ventas en la agencia de viajes DX Hotels: https://js.devexpress.com/Demos/DXHotels/#home y como parte de las tareas que tiene que hacer, está el de verificar las tarifas cobradas a los clientes; de manera que ella hace una búsqueda de los hoteles con la tarifa más económica en los diferentes destinos ofrecidos y valida que el precio total sea correcto. El problema es que Carolina maneja muchos clientes a nivel nacional de manera que requiere de un analista de automatización que le ayude a validar esta información antes de sacar los planes a PDN.

Cree un robot que permita seleccionar el hotel con la tarifa más económica para cada uno de los destinos ofrecidos y que valide que el precio total a pagar (TOTAL TO PAY NOW) sea el correspondiente con la siguiente fórmula: No días x Tarifa día.

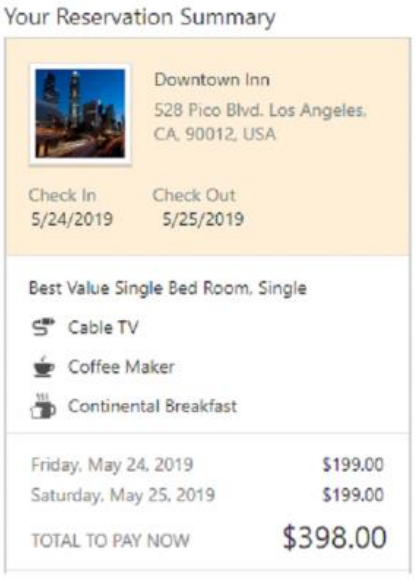


Figura 1.1 Planteamiento del problema (SpeedUpTech, 2022)

3.2 Planteamiento del problema

En este proyecto se realiza la historia de usuario que son la representación de un requisito escrito utilizando el lenguaje común del usuario, llevando una estructura y recalcando en la descripción el cómo, quiero, para; dicha estructura será de la siguiente manera:

HotelDx

Título: Buscar Hotel

Descripción:

COMO jefa de ventas

QUIERO verificar las tarifas cobradas a los clientes

PARA validar que el precio total sea correcto.

Asignación: Miguel Hugo

Prioridad: Alta

Estimación: 6

Criterios de Aceptación:

- Seleccionar uno de las localizaciones del hotel que aparecen en la lista desplegable.
- Seleccionar o escribir la fecha de entrada y de salida, dicha fecha se debe escribir con formato MM-DD-YYYY.
- Seleccionar cuántos adultos y niños estarán en el hospedaje, teniendo en cuenta que el límite a ingresar es de 4 personas en cada uno.
- Crear un robot que identifique cuál es la tarifa más barata y seleccionarla

Una vez que se tenga listo la historia de usuario, se procede a crear los casos de prueba cumpliendo con los criterios de aceptación de la historia de usuario, estos casos serán tanto válidos, llevando una estructura como la siguiente tabla:

Tabla 1.2 Casos de prueba (Elaboración propia, 2022)

No.	dentificador	Escenario	Descripción	Pasos	Resultado Esperado	Resultados Obtenidos	Probador	Estado
1	CP_01	[Frontend] Buscar hotel	Buscar un hotel ingresando los datos idóneos	1. ingresar a la página https://js.devexpress.com/Demos/DXHotels/#home 2. Llenar los campos solicitados, seleccionando una de las opciones de la lista desplegable de la localización, seleccionar o escribir la fecha de ida y vuelta teniendo el formato MM-DD-YYYY, seleccionar cuántos adultos y niños van a hospedarse. 3. Dar click en el botón de buscar.	El sistema debe mostrar los resultados de acuerdo a la localización solicitada, mostrando una lista de los hoteles disponibles, mostrando el precio, dirección y una breve descipción del mismo	El sistema sí muestra lo solicitado de manera correcta	Miguel H.	IN PROCESS
2	CP_02	[Frontend] Buscar hotel	Buscar un hotel ingresando la fecha incorrecta	1. ingresar a la página https://js.devexpress.com/Demos/DXHotels/#home 2. Llenar los campos solicitados, seleccionando una de las opciones de la lista desplegable de la localización, seleccionar o escribir la fecha de ida y vuelta teniendo el formato YYYY-MM-DD, seleccionar cuántos adultos y niños van a hospedarse. 3. Dar click en el botón de buscar.	El sistema debe mostrar una alerta en relación a que se colocó mal la fecha, haciendo que el botón de seach no se pueda pulsar.	El sistema si muestra una alerta con el mensaje de color rojo en el dominio que está erroneo.	Miguel H.	IN PROCESS

Con los casos de prueba ya planteados se procede a convertirlos en lenguaje *gherkin*.

Scenario Outline: Hotel Search in home

Given Go to the official website of the hotel

When we fill in the data required to search for the hotel

| location | checkIn | checkOut | numberAdult | numberChildren |

| <location> | <checkIn> | <checkOut> | <numberAdult> | <numberChildren> |

And select the hotel with the lowest price

Then he will see the "<totalPagar>"

Examples:

| location | checkIn | checkOut | numberAdult | numberChildren | totalPagar |

| Nassau | 10/11/2022 | 10/20/2022 | 2 | 2 | 2490.00 |

3.3 Desarrollo del problema

Una vez creada toda la documentación, se procede a crear la estructura del proyecto con el entorno de desarrollo integrado (IDE) *Intellij*, implementando las variables de entorno tanto de Java como de Maven, la creación del proyecto tiene que ser con la estructura Maven implementando el patrón de diseño *ScreenPlay* y los principios SOLID dando como resultado una buena creación del proyecto implementando las buenas prácticas.

La arquitectura *ScreenPlay* consta de la siguiente estructura, que se denominarán como carpetas y dentro de cada carpeta tendrá las actividades o las clases que se necesiten implementar cumpliendo con los principios SOLID.

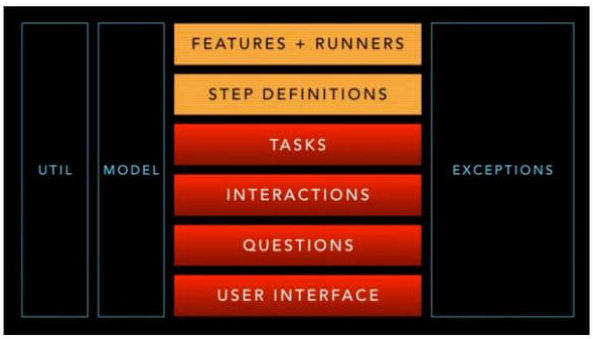


Figura 2.1 Patrón de diseño ScreenPlay (Villa, A. 2020.)

3.3.1 Control de versiones con Git y GiHub

Antes de crear el proyecto, es necesario crear un repositorio en GitHub para poder tener un sistema de control de versiones, le ponemos el nombre que pueda ser identificado de acuerdo a lo que se está realizando, dicho repositorio tendrá tres ramas base, siendo estas; Main/Master, Develop, Release. Cada una de ellas se puede crear de manera remota.

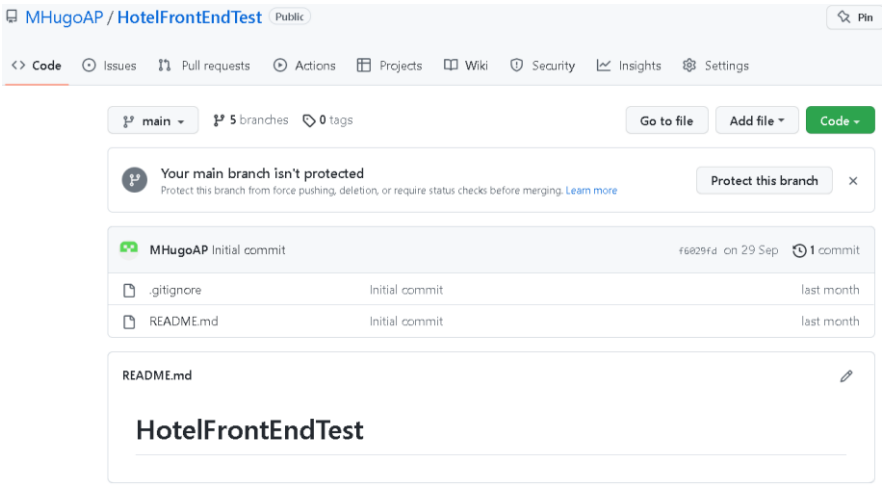


Figura 2.1.1 Creación del repositorio

Después de crear el repositorio se procede a clonarlo en nuestro local, se abre el Git Bash (que es una ventana que nos sirve para ejecutar diversos comandos) ponemos la línea de comandos para clonarlo con la llave SSH para que sea de manera segura.

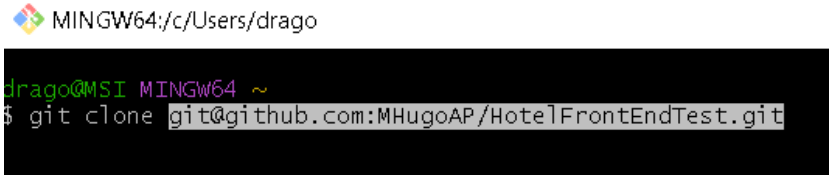


Figura 2.1.2 Clonar el repositorio

3.3.2Estructura del proyecto

Abrimos el IDE que vamos a utilizar y creamos un nuevo proyecto de tipo Maven, para tenemos nuestro sistema de gestor de versiones es necesario situarnos en la ruta donde de clonó dicho repositorio. Para ejecutar nuestro proyecto es necesario descargar el *ChromeDriver* de acuerdo a la versión que tenga el navegador de la máquina. Este drive se pondrá en la carpeta raíz

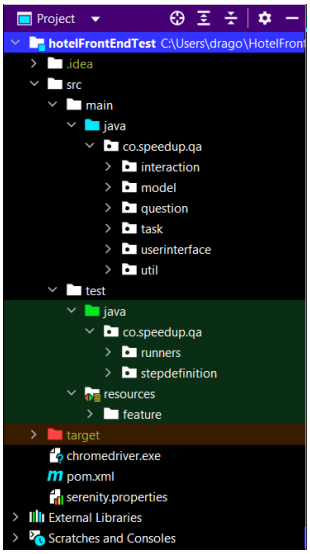


Figura 2.2 Implementación del ScreenPlay (Elaboración propia, 2022.)

3.3.3 Configuración del *serenity*

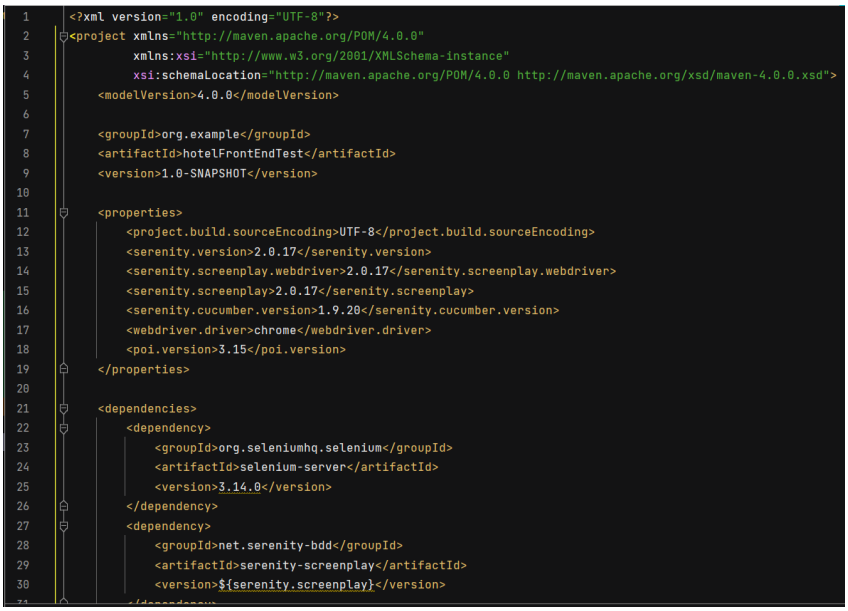
Creamos un archivo en la raíz llamado *serenity.properties* para que podamos definir las propiedades para que nuestro navegador abra como se requiere. Usando el lenguaje de *serenity* le decimos que abra el navegar en modo incognito y maximice la ventana.

```
1  webdriver.driver=chrome
2  chrome.switches=--incognito,--start-maximized
```

Figura 2.2.1 Configuración del serenity.properties (Elaboración propia, 2022.)

3.3.4 Dependencias y otras propiedades

En el archivo pom.xml pondremos todas las dependencias que vamos a necesitar para que nuestros test ejecuten correctamente. Estas dependencias van a variar de acuerdo al proyecto que se esté realizando.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>hotelFrontEndTest</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <serenity.version>2.0.17</serenity.version>
        <serenity.screenplay.webdriver>2.0.17</serenity.screenplay.webdriver>
        <serenity.screenplay>2.0.17</serenity.screenplay>
        <serenity.cucumber.version>1.9.20</serenity.cucumber.version>
        <webdriver.driver>chrome</webdriver.driver>
        <poi.version>3.15</poi.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-server</artifactId>
            <version>3.14.0</version>
        </dependency>
        <dependency>
            <groupId>net.serenity-bdd</groupId>
            <artifactId>serenity-screenplay</artifactId>
            <version>${serenity.screenplay}</version>
```

Figura 2.2.2 Configuración de las dependencias (Elaboración propia, 2022.)

3.3.5 Feature

Una vez que se haya creado la estructura de acuerdo al patrón *ScreenPlay* se procede a realizar la *feature* que requeriremos, eso se hará de acuerdo al caso de prueba que se creó convirtiéndolo al lenguaje *Gherkin*, esta *feature* va dentro de la carpeta de *src* > *test* > *resourse* > *feature* > *home.feature*. En el *When* crearemos una lista la cual estará vinculada a los *Examples*.

```
#language: en
  #autor: miguel hugo AP

Feature: Hotel Search
  As AQ Automation
  I want to find a hotel
  To stay in that hotel

  Scenario Outline: Hotel Search in home

    Given Go to the official website of the hotel
    When we fill in the data required to search for the hotel
      | location   | checkIn   | checkOut   | numberAdult   | numberChildren   |
      | <location> | <checkIn> | <checkOut> | <numberAdult> | <numberChildren> |
    And select the hotel with the lowest price
    Then he will see the "<totalPagar>"

    Examples:
      | location | checkIn    | checkOut   | numberAdult | numberChildren | totalPagar |
      | Nassau   | 10/11/2022 | 10/20/2022 | 2           | 2              | 2490.00    |
      | London   | 10/15/2022 | 10/30/2022 | 1           | 3              | 2384.00    |
      | New York | 10/10/2022 | 10/11/2022 | 1           | 1              | 238.00     |
      | Honolulu | 12/12/2022 | 12/30/2022 | 2           | 0              | 2109.00    |
```

Figura 2.2.3 Creación de la Feature (Elaboración propia, 2022.)

3.4 Creación del código

De ahí se crea una clase el cuál se va a encargar de convertir el lengaje *Gherkin* a Java, denominado runners de acuerdo al patrón de diseño implementado. En esta clase vamos a crear una anotación de tipo *@RunWith* que nos permite ejecutar prubas unitarias sobre el *Cucumber* con el *Serenity*, para la anotación de *@CucumberOptions* llamaremos a las features poniendo su ruta de localización, el glue que nos ayuda a localizar el archivo de los pasos definidos (*StepDefinition*), y para el apartado de snippets se pondrá un estilo de esccritura para las frases o palabras compuestas. Esta clase va dentro de *src* > *test* > *java* > *co.speedup.qa* > *runners* > *runner.class*

```java
package co.speedup.qa.runners;

import cucumber.api.CucumberOptions;
import cucumber.api.SnippetType;
import net.serenitybdd.cucumber.CucumberWithSerenity;
import org.junit.runner.RunWith;

Miguel Hugo *
@RunWith(CucumberWithSerenity.class)
@CucumberOptions(features = "src/test/resources/feature/home.feature"
        , glue = "co/speedup/qa/stepdefinition"
        , snippets = SnippetType.CAMELCASE)
public class HomeRunner {

}
```

Figura 2.2.4 Creación del runner (Elaboración propia, 2022.)

Depués de ejecuatr nuestra clase runner se tiene que copiar el resultado que nos arroja la consola que es la conversión del lenguaje Gherkin a Java, y se procede a pegarlo en una nueva clase que estará en la ruta *src* > *test* > *java* > *co.speedup.qa* > *stepdefinition* > *HomeStepDefinition.class*, el cual que va a encargar de llamar a los actores y métodos que se implementará en el proyecto.

```java
1 usage  Miguel Hugo *
@Given("^Go to the official website of the hotel$")
public void goToTheOfficialWebsiteOfTheHotel() {

}

1 usage  Miguel Hugo *
@When("^we fill in the data required to search for the hotel$")
public void weFillInTheDataRequiredToSearchForTheHotel(List<HomeModel> models) {
}

1 usage  new *
@And("^select the hotel with the lowest price$")
public void selectTheHotelWithTheLowestPrice() {

}

1 usage  new *
@Then("^he will see the \"([^\"]*)\"$")
public void heWillSeeThe(String totalPage) {

}
```

Figura 2.2.5 Creación de los Step definition (Elaboración propia, 2022.)

Lo que sigue es configurar nuestro entorno, para ello vamos a llamar a nuestro *webDriver* usando la anotación de *@Managed*, una anotación *@Before* para que se ejecute antes que

nuestros *Given*, el ese método *setUpOnstage* le decimos que cunado pueda abrir el navegador llame a nuestro actor, esto es para las buenas prácticas.

```
27  public class HomeStepDefinition {
        1 usage
28      @Managed
29      private WebDriver myBrowser;
30
        Miguel Hugo
31      @Before
32      public void setupOnstage() {
33          setTheStage(Cast.whereEveryoneCan(BrowseTheWeb.with(myBrowser)));
34          theActorCalled( requiredActor: "Miguel Hugo");
35      }
```

Figura 2.2.6 Configuración de los step definition(Elaboración propia, 2022.)

Y por último le tenemos que decir que termine la escena para que el cierre sea limpio después de cada iteración.

```
        Miguel Hugo *
59      @After
60      public void closeTheStage() {
61          OnStage.drawTheCurtain();
62      }
63  }
```

Figura 2.2.7 Creación del cierre del escenario (Elaboración propia, 2022.)

Una vez creada los *steps* procedemos a crear las clases para que los actores realizan las tareas correspondientes, ya sea; *interactions*, *models*, *questions*, *tasks*, *userinterface*, *utils*. De acuerdo al patrón de diseño *ScreenPlay*.

Lo primero que se tiene que hacer es decirle al programa que abra el navegador, para esto se creará una nueva clase en la ruta *src* > *main* > *java* > *co.speedup.qa* > *Task* > *OpenBrowser.class*

Se implementa la *Task* de *Srenity BDD* en nuestra clase, ingresamos el método que se requiere, creamos al actor y le decimos que intente abrir el navegador, con una espera de tres segundos. Creamos otro método estático y que retorne las tareas solicitadas.

```
4 usages  Miguel Hugo
public class OpenBrowser implements Task {
    Miguel Hugo
    @Override
    public <T extends Actor> void performAs(T actor) {
        actor.attemptsTo(Open.url( targetUrl: "https://js.devexpress.com/Demos/DXHotels/#home"), WaitInteraction.waitFor( seconds: 3));
    }
    1 usage  Miguel Hugo
    public static OpenBrowser browser(){
        return Tasks.instrumented(OpenBrowser.class);
    }
}
```

Figura 2.2.8 Creación de la tarea encargada de abrir el navegador (Elaboración propia, 2022.)

Para llamar a esa clase creada, la cual se encarga de abrir el navegados, la tendremos que llamar en los *step definition* que son los pasos que debe realizar la automatización, esta va en el *Given* ya que será lo primero que se requiere para ejecutarlo.

Le decimos que, en el escenario, el actor está en escena y que fue capaz de abrir el navegador, llamando a la clase *OpenBrowser.class*

```
1 usage  Miguel Hugo *
@Given("^Go to the official website of the hotel$")
public void goToTheOfficialWebsiteOfTheHotel() {
    OnStage.theActorInTheSpotlight().wasAbleTo(OpenBrowser.browser());
}
```

Figura 2.2.9 Implementación de la tarea en el step definition (Elaboración propia, 2022.)

Lo que sigue es ingresar los datos de acuerdo a los *examples* de la *feature* para que busque el hotel requerido, para ello, tenemos que buscar los localizadores necesarios para que nuestro actor realice las acciones requeridas.

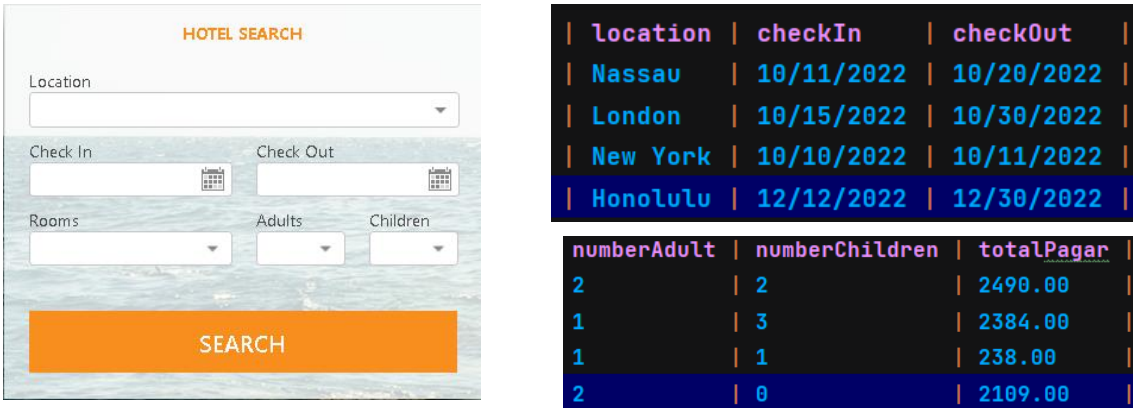


Figura 2.2.10 Comparación de la feature con el navegador de buscar hoteles (Elaboración propia, 2022.)

Una vez encontrado los xPath, se procede a crear una nueva clase para almacenarlo de acuerdo a la interfaz que se está visualizadon, esta clase va dentro de *src* > *main* > *java* > *co.speedup.qa* > *userinterface* > *HomeUserInterface.class* Se le pone static final ya que el valor no cambiará.

```
1 usage
public static final Target LOCATION_HOME = Target
        .the( targetElementName: "Select the locate")
        .locatedBy( cssOrXPathSelector: "(//div[@class='dx-dropdowneditor-icon'])[1]");

1 usage
public static final Target LOCATION_HOME_NASSAU = Target
        .the( targetElementName: "Nassau")
        .locatedBy( cssOrXPathSelector: "//div[@class='dx-item-content dx-list-item-content'][normalize-space()='{0}']");

1 usage
public static final Target CHECK_IN_CLICK = Target
        .the( targetElementName: "Entry date")
        .locatedBy( cssOrXPathSelector: "(//input[@class='dx-texteditor-input'])[2]");
```

Figura 2.2.11 Implementación de los localizadores (Elaboración propia, 2022.)

Debemos tener en cuanta que no hay *xPath* únicos por lo cual debemos buscar el más apto y con lo más reducido que sea posible.

```
2 usages
public static final Target CHECK_OUT_CLICK = Target
        .the( targetElementName: "Check Out")
        .locatedBy( cssOrXPathSelector: "/html/body/div/div/div/div[2]/div[1]/div/div/div[1]/div/div[1]/div[2]/div[1]/div/div/div/div/div/di

1 usage
public static final Target SEARCH_BUTTON = Target
        .the( targetElementName: "Search button")
        .locatedBy( cssOrXPathSelector: "//div[@aria-label='SEARCH']//div[@class='dx-button-content']");

1 usage
public static final Target PEOPLE = Target
        .the( targetElementName: "{0}")
        .locatedBy( cssOrXPathSelector: "//input[contains(@id,'_{0}')]");

1 usage
public static final Target VARIABLE = Target
        .the( targetElementName: "{0}")
        .locatedBy( cssOrXPathSelector: "//*[@class='dx-overlay-wrapper dx-dropdowneditor-overlay dx-popup-wrapper dx-dropdownlist-popup-wra

}
```

Figura 2.2.12 Implementación de los localizadores (Elaboración propia, 2022.)

Creamos una clase para los constructores junto con el *get* para llamarlo en las *task* y que escriba de acuerdo a los datos de la *feature*. Esta clase estará en la ruta *src* > *main* > *java* > *co.speedup.qa* > *model* > *HomeModel.class*

```
Miguel Hugo *
public class HomeModel {
    2 usages
    private String location;
    2 usages
    private String checkIn;
    2 usages
    private String checkOut;

    2 usages
    private String numberAdult;

    2 usages
    private String numberChildren;

    Miguel Hugo *
    public HomeModel(String location, String checkIn, String checkOut, String numberAdult, String numberChildren) {
        this.location = location;
        this.checkIn = checkIn;
        this.checkOut = checkOut;
        this.numberAdult = numberAdult;
        this.numberChildren = numberChildren;
    }

    1 usage  Miguel Hugo
    public String getLocation() {
        return location;
    }
```

```
2 usages  Miguel Hugo
public String getCheckIn() {
    return checkIn;
}

2 usages  Miguel Hugo
public String getCheckOut() {
    return checkOut;
}

1 usage  new *
public String getNumberAdult() {
    return numberAdult;
}

1 usage  new *
public String getNumberChildren() {
    return numberChildren;
}
}
```

Figura 2.2.13 Implementación los constructores y getters de acuerdo a los datos requeridos de la feature (Elaboración propia, 2022.)

Continuamos con la creación de una nueva *Task* para indicarle que llene los datos de acuerdo a la *feature* y a los localizadores. Esta clase se creará en la ruta *src* > *main* > *java* > *co.speedup.qa* > *Task* > *HomeTask.java*

Implementamos las *Task* y añadimos el método del *Task*. Creamos la instancia de la clase creado para los constructores, creamos el constructor que almacene la instancia del constructor previamente creado.

Creamos al actor para que intente dar clic y que selecciones de acuerdo a los datos del *Model* que está ligado a la *feature*, en otros se le dirá al actor que escriba con el *Enter.theValue* de acuerdo al *Model* usando los *getters*. Para que actor sepa dónde hacer la interacción, se manda a llamar las *user interface* que es donde los *xpath* se crearon.

```
16      public class HomeTask implements Task {
17
            8 usages
18          private HomeModel homeModelInstance;
19
            1 usage   Miguel Hugo
20          public HomeTask(HomeModel homeModel){
21              this.homeModelInstance = homeModel;
22          }
23
            Miguel Hugo *
24          @Override
25  O↑ @    public <T extends Actor> void performAs(T actorHome) {
26              actorHome.attemptsTo(Click.on(LOCATION_HOME));
27              actorHome.attemptsTo(Click.on(LOCATION_HOME_NASSAU.of(homeModelInstance.getLocation())));
28              actorHome.attemptsTo(Enter.theValue(homeModelInstance.getCheckIn()).into(CHECK_IN_CLICK));
29              actorHome.attemptsTo(Click.on(CHECK_OUT_CLICK), RobotDelete.on() ,
30                      Enter.theValue(homeModelInstance.getCheckOut()).into(CHECK_OUT_CLICK));
31              actorHome.attemptsTo(AddPeople.people( type: "adults", homeModelInstance.getNumberAdult()),
32                      AddPeople.people( type: "children", homeModelInstance.getNumberChildren()));
33              actorHome.attemptsTo(Click.on(SEARCH_BUTTON));
34
35              try {
36                  actorHome.remember("days", Quantity.ofDays(homeModelInstance.getCheckIn(),
37                          homeModelInstance.getCheckOut()));
38              }catch (ParseException exception){
39                  exception.printStackTrace();
40              }
41          }
```

Figura 2.2.14 Creación de la tarea encargada de hacer la búsqueda de los hoteles, recordando el día de entrada y de salida (Elaboración propia, 2022.)

En el try le estamos diciendo que actor recuerde la palabra "*days*" en relación a la fecha de entrada y de salida de acuerdo a la *feature*, y el catch es por si hay una excepción sea controlada.

Creamos otro método estático donde obtendrá el constructor del paquete *Model*, y retornará la nueva clase con el constructor como parámetro.

```
                1 usage   Miguel Hugo
43    @         public static HomeTask withData(HomeModel homeModel){
44                  return new HomeTask(homeModel);
45              }
46          }
```

Figura 2.2.15 Creación del método retornando la misma clase (Elaboración propia, 2022.)

Creamos otra nueva clase ubicado en la ruta *src* > *main* > *java* > *co.speedup.qa* > *útil* > *Quantity.class* la cual servirá para implementar un método y convertir un Dato de Fecha a un tipo entero.

```
7   public class Quantity {
8
9       private Quantity() {
10      }
11
            1 usage
12      public static int ofDays(String startDate, String endDate) throws ParseException {
13
14          SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "MM/dd/yyyy");
15          Date fechaIni = dateFormat.parse(startDate);
16          Date fechaFin = dateFormat.parse(endDate);
17          return (int) ((fechaFin.getTime() - fechaIni.getTime()) / 86400000) + 1;
18      }
19  }
```

Figura 2.2.16 Creación de una utilidad que nos sirve para obtener el resultado de la fecha de entrada y de salida en un valor entero (Elaboración propia, 2022.)

Todas estas operaciones, métodos y actores implementados se llamarán a los *step definition* para que el actor esté en escena y haga el procedimiento requerido. Recibirá como parámetro una lista de la clase *Model* que tiene un constructor y los *getters*, le indicamos que el actor está en escena y que intente realizar dichas operacines que está en el índice cero.

```
1 usage   Miguel Hugo
42      @When("^we fill in the data required to search for the hotel$")
43  @   public void weFillInTheDataRequiredToSearchForTheHotel(List<HomeModel> models) {
44          OnStage.theActorInTheSpotlight().attemptsTo(HomeTask.withData(models.get(0)));
45      }
```

Figura 2.2.17 Implementación de la tarea en el setep definition (Elaboración propia, 2022.)

Se procede a crear una nueva clase en la *task*, dicha tarea tendrá como objetivo seleccionar el hotel con el menor precio. Crearemos un *ArrayLsit* para que guarde los datos de la primera página que se muestre de acuerdo a la localización seleccionada. Y crearemos otro *ArrayList* de *ArrayList* para que almacene la lista creada anteriormente.

Vamos a crear un actor que recuerdo una llave denominada "*cheaper*" y que hará es reemplazar los datos almacenados en la lista, los remplazará de un "$" a un "" para que solo tome los números.

De ahí va a esperar hasta que el localizados ingresado sea remplazado por "$" a un "" y si ese localizador hace el emparejamiento para que sea visible sin que pase de un determinado tiempo proceda a dar clic en el localizador, remplazando de nuevo por un "$" a un "" para obtener los valores numéricos. Se crea un método para que retorne esa tarea de dicha clase.

```java
4 usages
public class ChooseHotelBy implements Task {

    3 usages
    private ArrayList<String> values = new ArrayList<>();

    2 usages
    ArrayList<ArrayList<String>> list = new ArrayList<>();

    @Override
    public <T extends Actor> void performAs(T actorChoose) {
        for (int i = 0; i < ResultHotel.HOTEL_INFO.resolveAllFor(actorChoose).size(); i++) {
            values.add(ResultHotel.HOTEL_COST.of(String.valueOf( i + 1)).resolveFor(actorChoose).getText());
            list.add(i, values);
            values = new ArrayList<>();
        }
        List<String> result = Calculate.withLowerPrice(list);

        actorChoose.remember("cheaper", result.get(1).replace( target: "$",  replacement: ""));
        actorChoose.attemptsTo(WaitUntil.the(ResultHotel.BTN_BEST_PRICE.of(result.get(1).replace( target: "$",  replacement: "")),
                WebElementStateMatchers.isVisible()).forNoMoreThan( amount: 20).seconds(), WaitInteraction.waitFor( seconds: 2),
                Click.on(ResultHotel.BTN_BEST_PRICE.of(result.get(1).replace( target: "$",  replacement: ""))),
                WaitInteraction.waitFor( seconds: 2));
    }

    1 usage
    public static ChooseHotelBy lowerPrice(){
        return Tasks.instrumented(ChooseHotelBy.class);
    }
}
```

Figura 2.2.18 Creación de una tarea para que almacene todos los valores de los hoteles obtenidos en la primera página (Elaboración propia, 2022.)

Procedemos a llamarlo en nuestro step *definition*, diciéndole que el actor está en escena y que intente realizar las operaciones de dicha clase.

```java
1 usage   new *
@And("^select the hotel with the lowest price$")
public void selectTheHotelWithTheLowestPrice() {
    OnStage.theActorInTheSpotlight().attemptsTo(ChooseHotelBy.lowerPrice());
}
```

Figura 2.2.19 Implementación de la tarea en el step definition (Elaboración propia, 2022.)

Procedemos a crear otra clase para validar que el valor total a pagar sea la misma que la *feature*, creamos una clase en la ruta *src > main > java > co.speedup.qa > question > TotalForQuestion,class*

Dicha clase va a implementar las *question* con el tipo genérico *String* y procedemos a implementar el método requerido.

En esta parte le decimos al actor que obtenga el recuerdo de las llaves creadas previamente almacenándolo en un tipo de dato entero y en el otro de tipo *double*, retornará el método decimal que posteriormente se va a crear y remplazará una “,” por un “.” Para que el tipo de dato *double* lo tome como válido.

```
4 usages
8   public class TotalForQuestion implements Question<String> {
9
10      @Override
11  @   public String answeredBy(Actor actor) {
12          int days = actor.recall( key: "days");
13          double total = Double.parseDouble(actor.recall( key: "cheaper"));
14          total = total * days;
15          return decimals(total).replace( target: ",",  replacement: ".");
16      }
```

Figura 2.2.20 Creación de una question (Elaboración propia, 2022.)

Creamos el método decimal de tipo *String* que recibirá como parámetro un *double*, dicho método nos servirá para darle un formato de cuantos decimales puede almacenar y que retorne el mismo valor *double*.

```
1 usage
18      private String decimals(double total) {
19          DecimalFormat df = new DecimalFormat( pattern: "0.00");
20          df.setMinimumIntegerDigits(2);
21          return df.format(total);
22      }
```

Figura 2.2.20.1 Creación de una question (Elaboración propia, 2022.)

Se crea otro método para que retorne la misma clase.

```
1 usage
24  @   public static TotalForQuestion days() {
25          return new TotalForQuestion();
26      }
27  }
```

Figura 2.2.20.2 Creación de una question (Elaboración propia, 2022.)

Creamos otra *question* en la ruta *src > main > java > co.speedup.qa > question > ValueQuestion,class*

Nos servirá para obtener el localizador de los precios y remplazar el valor “$” por un “”

```
4 usages
public class ValueQuestion implements Question<String> {

    @Override
    public String answeredBy(Actor actor) {
        return PageHotel.TOTAL_TO_PAGE.resolveFor(actor).getText().replace( target: "$", replacement: "")
                .trim();
    }

    1 usage
    public static ValueQuestion toPay(){
        return new ValueQuestion();
    }
}
```

Figura 2.2.21 Creación de una question que remplaza valores (Elaboración propia, 2022.)

Procedemos a llamas las clases creadas a nuestro *step definition*, diciendo que el actor está en escena y que debería ver lo que tiene en esa clase y hacer un emparejamiento para comparar el resultado de nuestra *feature* con el resultado con el total a pagar ubicado con nuestro localizador.

```
1 usage   new *
@Then("^he will see the \"([^\"]*)\"$")
public void heWillSeeThe(String totalPage) {
    theActorInTheSpotlight().should(GivenWhenThen.seeThat(TotalForQuestion.days(),
            Matchers.equalTo(totalPage)), GivenWhenThen.seeThat(ValueQuestion.toPay(),
            Matchers.equalTo(totalPage)));
}
```

Figura 2.2.22 Implementación de las question en el step definition (Elaboración propia, 2022.)

Por último, le diremos en el *step definition* que cierre el telón o que dibuje la cortina para finalizar con el proceso y/o la iteración. Para ello creamos un método con la anotación de *@After*, dicho método será vacío.

```
Miguel Hugo *
59      @After
60      public void closeTheStage() {
61          OnStage.drawTheCurtain();
62      }
63  }
```

Figura 2.2.23 Cierre del escenario (Elaboración propia, 2022.)

Ya que contamos con el sistema de control de versiones, es necesario subirlo a nuestro repositorio todo el proyecto, pero por las buenas prácticas se hará por cada uno de los archivos. Para ello, *Intellij* nos proporciona la facilidad de abrir el *git* desde el mismo programa.

Implementamos una serie de comandos, el primero será un *git add* el cual nos va a servir para agregar nuestro avance al *Staging Area* de ahí ponemos la ruta donde se encuentra el proyecto, el archivo o el documento que se va a subir. Después implementamos el comando *git commit -m "The file will be uploaded"* esto es un comentario y une breve descripción de lo que se va subir, esto tiene que estar escrito en inglés por las buenas prácticas. Y para finalizar se hace un *git push origin feature*, este comando nos sirve para subir nuestros cambios al repositorio.

CAPÍTULO V. ANÁLISIS Y DISCUSIÓN DE RESULTADOS

Al momento de desarrollar el proyecto se comenzó analizando la página web búscador de hoteles, se implementó las precondiciones y los criterios de aceptación en dicha página, se analizó la ruta y se crearon los casos de prueba necesarios para la automatización, esto se hizo en un ambiente ya de producción, porque el software ya estaba construido y listo para ser usado por los usuarios.

A la hora de realizar el proyecto se hizo la mejora e implementación del aseguramiento de calidad de la página web búsqueda de hoteles, usando diversas herramientas tales como: *Cucumber*, *Serenity*, *Selenium* el cual nos ayuda a comunicarnos con el software mediante el código usando la interfaz gráfica *Intellij* e implementando el lenguaje de programación *Java* y buenas prácticas, teniendo como resultado una excelente calidad.

Con ello, también se analizó la ruta crítica de la página búsqueda de hoteles, se puede ver que se necesita varios valores de entrada y precondiciones para que se haga la ejecución correctamente, esto quiere decir que en la *feature* es necesario cambiar la fecha de entrada y salida, ya que esas fechas se vuelven obsoletas haciendo que falle en las pruebas automatizadas, es necesario| poner las fechas actuales o futuras.

Con el cambio de las fechas también cambia el costo total de la reservación, para esto se tiene que hacer la prueba de humo que ayuda a implementar la página búsqueda de hoteles con los valores de entrada nuevos, se compara el precio nuevo con el anterior y se actualiza para que se puede hacer la comparación y que la prueba cumpla con los criterios.

La historia de usuario que se puede apreciar en el planteamiento del problema 3.1 nos dice los criterios de aceptación y las precondiciones que se debe cumplir a la hora de automatizar dicha página, con esas historias de usuario se realiza los casos de prueba para corroborar que el software cumple o no cumple con lo requerido (véase en la Tabla 1.2 Casos de prueba).

4.1 Ejecución de todas las *features*

Con todo esto finalizado procedemos a implementar un comando de Maven, lo que hará el comando es ejecutar todas nuestras pruebas de automatización para corroborar que la ruta crítica se haya realizado correctamente, libre de errores. Dicho reporte se hará con Serenity. El tiempo de ejecución puede variar dependiendo del internet, de la funcionalidad de la página y la computadora en la que se esté ejecutando los test. El código a ejecutar es el siguiente: mvn clean verify serenity:aggregate

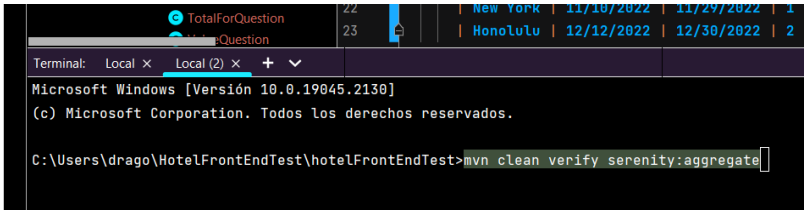


Figura 4.1 Resultados de las pruebas (Elaboración propia, 2022)

4.2 Reporte del *serenity*

Como se puede ver en la figura 4.2 nos indica que se ejecutó exitosamente con un tiempo estimado de 2 minutos con 15 segundos, dándonos un 100% de las pruebas automatizadas. Con todos estos datos podemos asegurar que la automatización de la página busca hoteles está funcionando correctamente, dando una calidad a los clientes y asegurando que su servicio funcione de manera segura.

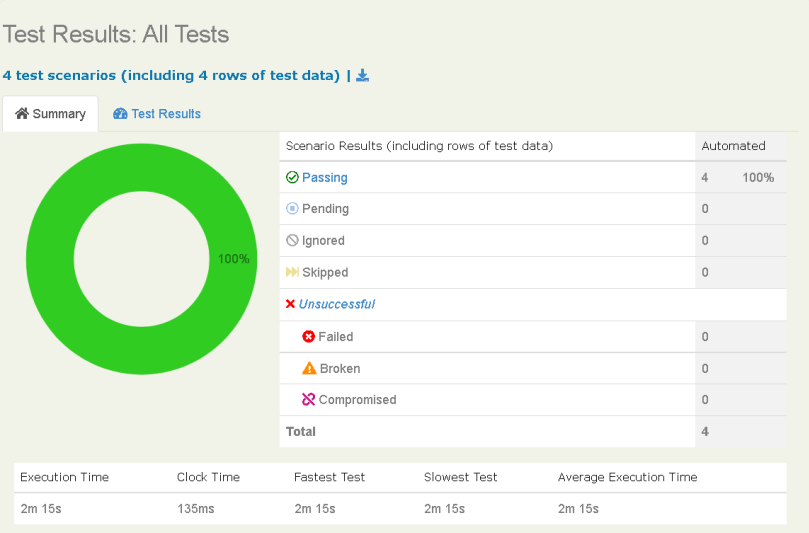


Figura 4.2 Resultados de las pruebas (Elaboración propia, 2022)

Podemos concluir que se cumplieron todos los criterios de aceptación y los requisitos contractuales con un tiempo estimado de 2 minutos y 15 segundos. Dando un 100% de cobertura en todos los escenarios utilizados, dándole al cliente una excelente calidad del software.

CAPÍTULO VI. CONCLUSIONES

Este proyecto tuvo como objetivo el demostrar que el aseguramiento de calidad es de suma importancia en las empresas, ya que les da a los clientes la certeza que su producto estará hecho de la más alta calidad posible, satisfaciendo las necesidades y los requerimientos establecidos.

Además, es de suma importancia que las empresas cuenten con trabajadores certificados por ISTQB u alguna otra organización que se encarga de impartir certificaciones de una alta calidad, aquellas personas que hayan realizado la certificación le da una mayor fuerza en el mercado laboral.

A través de la automatización de esta página web, fue posible ejecutar una ruta crítica con diferentes valores de entrada, validando que el resultado sea el mismo que en la documentación de los casos de prueba y los criterios de aceptación.

A la hora de usar la metodología de desarrollo ágil; SCRUM, me di cuenta que es una metodología en donde el producto se debe entregar con una buena calidad y en un corto periodo de tiempo, cumpliendo con una serie de normas de acuerdo al aseguramiento de calidad y cumpliendo los requisitos contractuales con el cliente.

En este y cualquier otro proyecto es importante la constante comunicación con el cliente, y con los trabajadores de diversas áreas, porque nos ayuda a tener un buen desempeño a la hora de realizar un nuevo proyecto, nos da una facilidad de reportar bugs y/o errores que se presentan antes de llegar a producción, se pueden hacer modificaciones de la interfaz, y el cliente nos puede dar diversas actualizaciones en las historias de usuario o al agregar una nueva funcionalidad. Ya que en esta empresa son flexibles al cambio.

FUENTES DE INFORMACIÓN

¿Qué es una API? - Guía sobre las API para principiantes - AWS. (2022). Retrieved October 7, 2022, from Amazon Web Services, Inc. website: https://aws.amazon.com/es/what-is/api/

Caroli, P. (2020, February 8). Scrum comienza con PBB - Caroli.org. Retrieved October 27, 2022, from Caroli.org website: https://caroli.org/es/scrum-comienza-con-pbb/#:~:text=La%20feature%20es%20una%20descripci%C3%B3n,un%20usuario%20con%20el%20producto.

contenidos. (2021, May 11). ▷ ¿Qué es el Xpath y para qué sirve? - Neo Wiki | NeoAttack. Retrieved October 11, 2022, from NeoAttack website: https://neoattack.com/neowiki/xpath/

E. Diez. "Aseguramiento de la calidad en la construcción de sistemas basados en el Conocimiento: un enfoque práctico", Revista Latinoamericana de Ingeniería de Software, 1(5): pp. 167-206, ISSN 2314-2642, 2013.

Gandarillas, A. (2017, September 14). Probar – Glosario de términos. Retrieved October 5, 2022, from Glosario.es website: https://glosario.es/istqb

G. Gómez. "Avances en las Mejoras de Procesos Software en las MiPyMes Desarrolladoras de Software: Una revisión Sistemática", Revista Latino Americana de Ingeniería de Software 2 (4): 262-268, ISSN 2314-2642, 2014.

G. Aguirre. "Desarrollo de un marco meto dológico del proceso de verificación y validación de software para pequeñas y medianas empresas". Revista de la Facultad de Ingeniería Industrial, 18(2): pp. 145-154, 2015.

G. Paladines. "Administración de la calidad en el desarrollo de un sistema de información". Facultad de ingeniería en electricidad y computación (FIEC). 2014

José Manuel Alarcón. (2022, June). Java: ¿Qué es Maven? ¿Qué es el archivo pom.xml? - campusMVP.es. Retrieved October 31, 2022, from campusMVP.es website: https://www.campusmvp.es/recursos/post/java-que-es-maven-que-es-el-archivo-pom-xml.aspx

mariluz.usero@chakray.com. (2020, August 10). Testing con Cucumber: Cómo afianzar la fiabilidad en los desarrollos. Retrieved October 19, 2022, from Chakray website: https://www.chakray.com/es/testing-cucumber-afianzar-fiabilidad-desarrollos/

Merino, M. (2020, April 5). Variables de entorno: qué son, para qué sirven y cómo podemos editarlas en Windows y Linux. Retrieved October 27, 2022, from Genbeta.com website: https://www.genbeta.com/desarrollo/variables-entorno-que-sirven-como-podemos-editarlas-windows-linux

Milano, P. (2007). Seguridad en el ciclo de vida del desarrollo de software. Argentina. www. cybsec. com/upload/cybsec_Tendencias2007_Seguridad_SDLC. pdf.

Principios SOLID en programación. (2020). Retrieved October 7, 2022, from Kata Software website: https://kata-software.com/es/publicaciones/principios-solid-en-programacion#:~:text=SOLID%20es%20un%20acr%C3%B3nimo%20acu%C3%B1ado,eficiente%20y%20f%C3%A1cil%20de%20mantener.

Rodríguez, C., & Dorado, R. (2015). ¿ Por qué implementar Scrum?. Revista Ontare, 3(1), 125-144

Romero, G. (2021, June 29). Cómo realizar pruebas automatizadas con Postman. Retrieved October 7, 2022, from Encora website: https://www.encora.com/es/blog/como-realizar-pruebas-automatizadas-con-postman#:~:text=Postman%20es%20una%20aplicaci%C3%B3n%20que,que%20posteriormente%20deber%C3%A1n%20ser%20validados.

Tripathy, P., & Naik, K. (2011). Software testing and quality assurance: theory and practice. John Wiley & Sons

Verity. (2021, June 23). La ISO/IEC 9126: 2001: Características de la calidad de software. Retrieved October 9, 2022, from Verity.cl website: https://www.verity.cl/blog/que-es-norma-iso-iec-9126-2001

Vergara, S. (2019, July 18). ¿Qué es BDD (Behavior Driven Development)? Retrieved October 11, 2022, from Blog ITDO - Agencia de desarrollo Web, APPs y Marketing en Barcelona website: https://www.itdo.com/blog/que-es-bdd-behavior-driven-development/

Villa, A. (2020). Cómo empezar con SERENITY BDD - SCREENPLAY. Retrieved October 11, 2022, from Pragma.com.co website: https://www.pragma.com.co/academia/lecciones/como-empezar-con-serenity-bdd-screenplay#:~:text=Serenity%20es%20una%20librer%C3%ADa%20de,y%20de%20manera%20m%C3%A1s%20eficiente.&text=Ya%20teniendo%20un%20poco%20m%C3%A1s,pasos%20para%20empezar%20a%20automatizar.

ANEXOS

Esta clase y métod tiene la funcionalidad de calcular el precio más bajo de acuerdo a la localización seleccionada, la fecha de entrada y salida.

Hace una iteración y remplazar el signo de pesos por un valor vacío para que se pueda usar como valor *double*. Esta clase se llamará en las *Task*, almacenándolo en otra lista, para crear una lista de listas.

```
public class Calculate {

    static double value = 0;
    private static String retun;
    static int posicion = 1;

    public static List<String> withLowerPrice(ArrayList<ArrayList<String>> list) {
        List<String> result = new ArrayList<>();
        value = Double.parseDouble(list.get(0).get(0).replace("$", ""));

        for (int j = 0; j < list.size(); j++) {
            if (Double.parseDouble(list.get(j).get(0).replace("$", "")) <= value) {
                retun = list.get(j).get(0);
                posicion = j + 1;
            }
        }
        result.add(String.valueOf(posicion));
        result.add(retun);

        return result;
    }
}
```