Automatización de pruebas canales de recaudos TUYA S.A

Juan José Ramírez Acosta

Informe final de práctica para optar al título de:

Ingeniero de Sistemas

Asesora Interna
Catalina María Céspedes Toro
Especialista en Gerencia de Proyectos

Asesora Externa
Leidys Martinez Vega
Especialista TI

Universidad de Antioquia - Ciudad Universitaria
Facultad de Ingeniería
Ingeniería de sistemas - Pregrado
2024

Cita	Acosta Ramírez, Juan José [1]
Referencia Estilo IEEE (2020)	[1] Acosta Ramírez, Juan Jose, "Automatización de pruebas canales de recaudos TUYA S.A, 2024", Prácticas académicas, Pregrado UdeA, Universidad de Antioquia, Ciudad Universitaria UdeA, 2024.

Tabla de contenido

RESUMEN 7
ABSTRACT 7
I. INTRODUCCIÓN 8
II. OBJETIVOS 9
A. Objetivo general 9
B. Objetivos específicos 9
III. MARCO TEÓRICO 10
IV. METODOLOGÍA 13
V. RESULTADOS Y ANÁLISIS 17
VII. CONCLUSIONES 20
REFERENCIAS 21

LISTA DE FIGURAS

Fig. 1. Diagrama ejemplo del proceso de desarrollo y pruebas en Tuya S.A 14
Fig. 2. Estructura del proyecto 18
Fig. 3. Resultados del test de Serenity 19
Fig. 4. Cobertura de las pruebas realizadas 19

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

UdeA	Universidad de Antioquia
S.A	Sociedad Anonima
ISO	Organización Internacional de Normalización
IEC	Comisión Electrotécnica Internacional
IEEE	Institute of Electrical and Electronics Engineers
QA	Aseguramiento de la calidad
CI/CD	integración continua/distribución continua
OWASP	Open Web Application Security Project
API	Interfaz de programación de aplicaciones

RESUMEN

En respuesta a la creciente demanda de desarrollo de software y con el propósito de elevar la calidad de los productos de TUYA S.A, esta propuesta tiene como objetivo implementar una solución de automatización de pruebas de software en el equipo de Canales de Recaudo de Tuya S.A, Mediante la adopción de estrategias de ejecución de pruebas utilizando herramientas de automatización. Se espera optimizar el proceso de pruebas, reducir el tiempo de entrega, asegurando la confiabilidad y disponibilidad de los productos que se construyen desde el equipo de Canales de Recaudo. Se utilizarán prácticas y herramientas de automatización de pruebas modernas para lograr una mayor eficiencia en el desarrollo y liberación de software.

Palabras clave **— Eficiencia, Calidad del Software, Herramientas de Automatización de Pruebas, Proceso de Pruebas.**

ABSTRACT

In response to the growing demand for software development and with the aim of improving the quality of TUYA S.A.'s products, this proposal aims to implement a software testing automation solution within the TUYA S.A. Revenue Channels team. This will be achieved through the adoption of test execution strategies using automation tools. The goal is to optimize the testing process, reduce delivery time, and ensure the reliability and availability of the products developed by the Revenue Channels team. Modern test automation practices and tools will be used to achieve greater efficiency in software development and release.

Keywords **—, Efficiency, Software Quality, Automation Testing Tools, Testing Process.**

I. INTRODUCCIÓN

En un entorno empresarial altamente competitivo y en constante evolución, la eficiencia en el desarrollo de software es esencial para mantener la satisfacción del cliente y la posición en el mercado. En este proyecto, el equipo de Canales de Recaudo de la empresa Tuya S.A enfrenta desafíos específicos en la construcción de software que pueden resolverse de manera más eficaz mediante la automatización de pruebas. La automatización de pruebas no solo contribuye a optimizar los costos del proceso de desarrollo, sino que también brinda un mayor control, calidad y reduce los tiempos de ejecución agregando mayor valor al producto final.

El equipo de canales de recaudos, está inmerso en un importante proceso de desarrollo ágil, caracterizado por la urgencia en la entrega de requisitos que deben ser desarrollados y probados en lapsos de tiempo notablemente reducidos. Esta dinámica ágil y la demanda de rapidez en la implementación ha llevado al equipo a reconocer la necesidad imperativa de fortalecer sus prácticas de pruebas y avanzar hacia la automatización de las mismas.

Al adoptar la automatización de pruebas, no solo se logra aumentar la velocidad del ciclo de desarrollo y asegurar una detección temprana de errores, sino que también se liberan recursos valiosos que pueden concentrarse en actividades más estratégicas. Al explorar este enfoque, será importante considerar cómo puede adaptarse y personalizarse para las necesidades específicas del equipo de Canales de Recaudo de Tuya S.A, manteniendo un enfoque en la calidad y en la optimización de los recursos.

En el contexto de automatización de pruebas, el rol como tester se vuelve fundamental durante la fase de contextualización. El tester colabora a la par con el equipo de desarrollo y de negocios para así comprender los requisitos del software, participa en la planificación y ejecución de pruebas de calidad, selecciona y supervisa herramientas de automatización adecuadas y cobertura integral. También mencionar que la gestión de datos de prueba, monitoreo continuo, optimización y análisis de resultados son responsabilidades fundamentales del tester, este no solo se limita a la ejecución de pruebas automatizadas, sino que desempeña un papel clave en garantizar la calidad y eficacia del proceso de pruebas en su conjunto.

II. OBJETIVOS

A. Objetivo general

Automatizar las pruebas del software desarrollado por el equipo de Canales de Recaudo en la empresa Tuya S.A para mejorar la eficiencia y la calidad en el proceso de desarrollo de este equipo.

B. Objetivos específicos

- Identificar los casos de prueba más adecuados para la automatización.
- Familiarización con herramientas y configuración del entorno de pruebas automatizadas.
- Diseñar y desarrollar conjuntos de pruebas automatizadas para los principales componentes del software.
- Construir pruebas automatizadas y dejarlas disponibles para su integración en el proceso CI/CD.
- Establecer métricas para evaluar la eficacia de la automatización de pruebas en términos de ahorro de tiempo y mejora de la calidad.

III. MARCO TEÓRICO

Introducción

La automatización de pruebas surge como un componente esencial en los equipos de desarrollo de software para garantizar la calidad y la eficiencia en la entrega de productos tecnológicos. este al reducir la intervención manual de la fase de pruebas, minimiza los errores humanos, acelera los tiempos de entrega y asegura una mayor consistencia en los resultados. Además facilita la repetición de pruebas, permitiendo una detección temprana de posibles fallos y proporcionando una retroalimentación inmediata a los desarrolladores. en este proyecto para garantizar una mayor coherencia y calidad a la hora de elaborar casos de prueba se hará una adopción de normas reconocidas, como ISO/IEC 12207 e IEEE 829 Estas normas proporcionan un marco estructurado para planificar, ejecutar y gestionar pruebas, asegurando las mejores prácticas internacionales.

1. ISO/IEC 12207 - Proceso del Ciclo de Vida del Software:

Esta norma es un estándar internacional que establece los procesos y actividades necesarios para el ciclo de vida del software. en cuanto a la calidad del software, esta norma aborda temas como lo son:

1. **Proceso de aseguramiento de calidad(QA):** donde tiene como objetivo garantizar que se establezcan y se mantengan estándares y procedimientos para la gestión de calidad, definiendo actividades relacionadas con la planificación y ejecución de revisiones, auditorías y pruebas para asegurar que el producto cumple con los requisitos de calidad especificados.

2. **Proceso de desarrollo:** abordando la calidad de software en el proceso de desarrollo, donde nos muestra la importancia de la gestión de la configuración, la gestión de la calidad, la verificación y validación. también proporciona directrices para la

identificación, aplicación de estándares y procedimientos de calidad durante las etapas del ciclo de vida del software.

3. **Proceso de mantenimiento:** donde destaca la importancia de gestionar la calidad durante las actividades de mantenimiento, proporcionandonos pautas para evaluar y mejorar la calidad del software existente, incluyendo la gestión de cambios y retroalimentación continua.

4. **Medición de la calidad:** donde se reconoce la importancia de la medición de la calidad del software y nos orienta sobre la planificación de mediciones para evaluar la eficacia de los procesos y la calidad del producto.

5. **Documentación de la calidad:** donde establece requisitos para la documentación de la calidad del software, incluyendo la necesidad de mantener registros de actividades relacionadas con la calidad como revisiones, pruebas y auditorías.

6. **Gestión de riesgos:** Están relacionados con la calidad del software, indicando la importancia de identificar, evaluar y mitigar los riesgos que puedan afectar la calidad del producto.

2. IEEE 829 - Estructura y Propósito de Documentos:

Por otro lado, aunque la IEEE 829 no es exclusivamente una norma de calidad del software, desempeña un papel crucial en garantizar la calidad mediante la documentación detallada de actividades de prueba. algunos de estos puntos clave relacionados con la calidad de software en el contexto de la IEEE 829 son:

1. **Plan de pruebas:** donde establece los requisitos para la creación de pruebas para garantizar la calidad del software a partir de una estrategia general de pruebas, los recursos necesarios, la programación y los objetivos de las pruebas. Un plan de pruebas bien elaborado contribuye directamente a la gestión de la calidad.

2. **especificaciones de pruebas:** la norma requiere la elaboración de especificaciones de pruebas, estas describen las condiciones de prueba, los procedimientos de prueba y los criterios de aceptación. Esto nos ayuda a evaluar si el software cumple con los requisitos establecidos.

3. **informes de pruebas:** Los informes son vitales para la gestión de la calidad, ya que proporcionan una visión clara del estado del software en términos de calidad y conformidad con los requisitos.

4. **rastreabilidad:** Ayuda a identificar qué casos de prueba están asociados con un requisito particular y viceversa.

5. **gestión de configuración de pruebas:** ayudan a mantener la integridad de las pruebas y contribuye a la calidad al garantizar que las versiones correctas de los elementos de prueba se utilicen en cada fase.

IV. METODOLOGÍA

La metodología a emplear será cualitativa, centrándose en la implementación de prácticas de automatización de pruebas en un entorno de desarrollo ágil. Se utilizarán herramientas como Azure Pipelines, SonarQube, Kiuwan, OWASP, Cucumber, SonarLint, Serenity y Selenium para habilitar un proceso integral de desarrollo y pruebas automatizadas. El proceso se dividirá en las siguientes fases:

1. Identificación de casos de prueba adecuados para la automatización.
2. Familiarización con Herramientas y Configuración del Entorno de Pruebas Automatizadas.
3. Diseño y desarrollo de conjuntos de pruebas automatizadas.
4. Construcción de pruebas automatizadas y dejarlas disponibles para su integración en el proceso CI/CD.
5. Establecimiento de métricas y evaluación de resultados.

Como analista de calidad en el equipo de la compañía, es imperativo seguir una serie de procesos apropiados y estratégicos específicos de la organización. Algunas de las responsabilidades clave incluyen:

- Garantizar el correcto funcionamiento de la automatización en los navegadores pertinentes, en caso de que sea aplicable.
- Documentar de manera exhaustiva todas las actividades realizadas durante el proceso.
- Adherirse rigurosamente a las buenas prácticas establecidas en los arquetipos definidos.
- Verificar que la automatización realizada cumpla con los requisitos de auditoría gubernamental.

Siguiendo estas normas los pasos a seguir para la implementación de una automatización de pruebas es la siguiente:

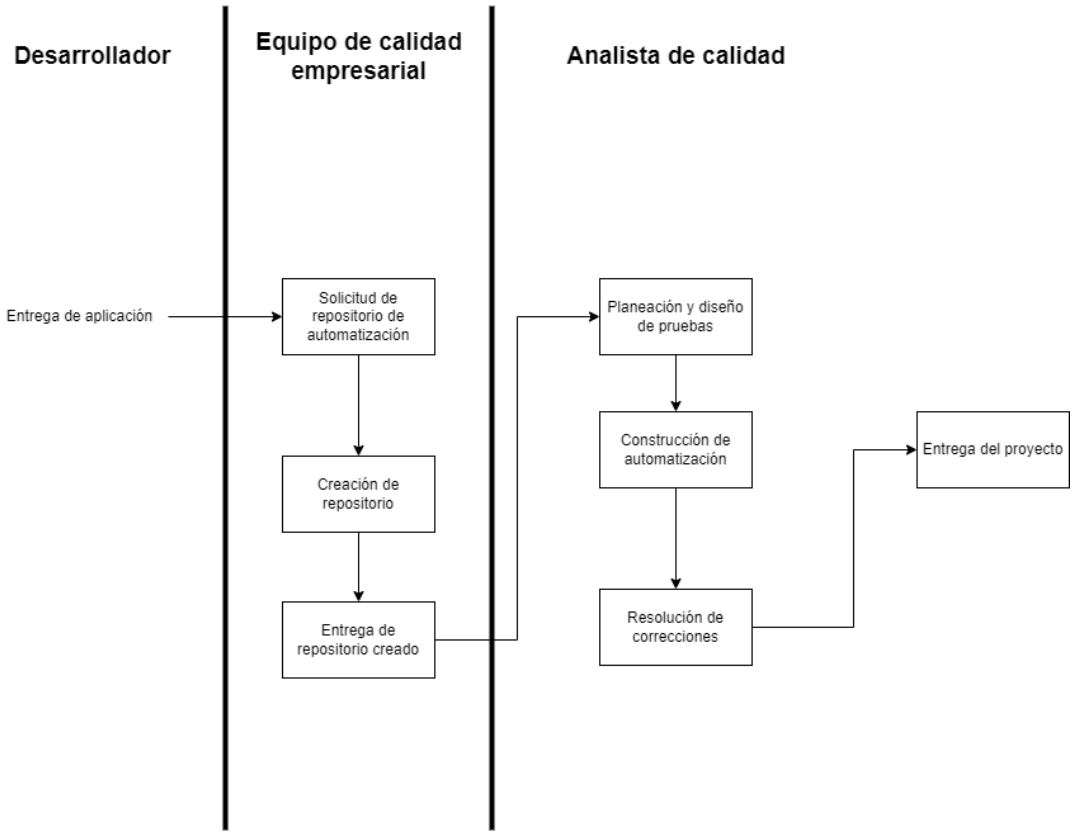


Fig. 1. Diagrama ejemplo del proceso de desarrollo y pruebas en Tuya S.A

En el contexto de la elaboración de esta propuesta, se llevará a cabo la elaboración de un plan de pruebas para dos interfaces de programación de aplicaciones (APIs). Estas APIs están diseñadas con el propósito de permitir la visualización de información esencial para nuestros clientes. En particular, se enfocarán en dos aspectos clave: la capacidad de listar los productos asociados a un cliente, incluyendo toda la información relevante de dichos productos, y la habilidad de presentar de manera detallada los valores de pago asociados al crédito.

El primer conjunto de pruebas se centrará en la API **"Listar Producto"**, específicamente en la funcionalidad que permite listar los productos vinculados al cliente. El segundo conjunto de pruebas se orientará hacia la API de **"Detalle del Producto"**. Aquí, el objetivo principal será garantizar la capacidad de listar detalladamente los valores de pago asociados al crédito del cliente.

El propósito principal de las pruebas funcionales será verificar que las APIs satisfacen las necesidades y expectativas de nuestros usuarios, del mismo modo en proporcionar a nuestros clientes una experiencia de pago mejorada y libre de inconvenientes al interactuar con sus tarjetas de crédito.

Teniendo en cuenta el contexto anteriormente descrito, se realizaron planes de prueba detallados para la ejecución de las APIs asociadas al servicio de crédito. Estos planes de prueba se estructuraron con el propósito de validar la funcionalidad y el cumplimiento de requisitos en las dos categorías principales, A continuación, se presenta un resumen general de las metodologías de validación empleadas:

Validaciones para **Detalle de Crédito:**

- Se realizaron pruebas exhaustivas para verificar la consulta de detalles de productos asociados a créditos Credicompras, así como la consulta de productos sin asociación.
- Se implementaron métodos de validación específicos para tarjetas encriptadas erróneas, tarjetas existentes, tarjetas con longitud mayor a 18 caracteres, tarjetas con caracteres letras, y tarjetas con caracteres especiales.
- Se evaluaron escenarios de consulta de detalles de productos con valores de pago no disponibles, valores de pago mínimo y total en 0, sin valores de pago mínimo, sin valores de pago total, y con fecha límite de pago.

Validaciones para **Listar Productos**:

- Se realizaron pruebas detalladas para la consulta de clientes con productos, así como la consulta de clientes sin productos y con cédulas encriptadas erróneas.
- Se aplicaron métodos de validación específicos para cédulas existentes, cédulas con longitud mayor a 16 caracteres, cédulas con caracteres letras, y cédulas con caracteres especiales.
- Se evaluaron validaciones específicas para clientes con dos o más productos de crédito, considerando diferentes combinaciones de negocios, tipos de productos y estados.
- En ambas categorías, se emplearon enfoques consistentes y métodos de validación estandarizados para garantizar la integridad y confiabilidad de las operaciones en las APIs de crédito, contribuyendo así a una experiencia de usuario sin inconvenientes al interactuar con el servicio de tarjeta de crédito.

V. RESULTADOS Y ANÁLISIS

Teniendo en cuenta los casos de prueba planteados para este proyecto, se crea el proyecto en Java aplicando el patrón de diseño Screenplay junto con la integración de Cucumber y Serenity BDD. Este enfoque combina la claridad y legibilidad de los casos de prueba escritos en Gherkin, facilitados por Cucumber, con la generación de informes detallados y visuales proporcionados por Serenity BDD. El patrón de diseño Screenplay se utiliza para estructurar las interacciones entre actores y tareas, brindando un marco sólido para la automatización de pruebas que se alinea con las necesidades específicas del proyecto.

En la figura 4 podemos ver que este proyecto está constituido por:

- **Enums:** Contiene un grupo de constantes agrupadas por clase que representan un comportamiento.
- **Exceptions:** Contiene la tipología de las excepciones que se quieren propagar en la ejecución de una automatización.
- **Interactions:** Contiene las actividades de bajo nivel que requiere el actor para interactuar con el sistema.
- **Modelos:** Contiene la representación de objetos del mundo real con sus características.
- **questions:** contiene aquellas clases que nos permiten realizar validaciones acerca del estado de los elementos.
- **Task:** contiene en cada clase un conjunto de interacciones que le permiten al actor realizar una acción completa en el sistema.
- **Utilities:** Contiene en cada clase funciones transversales de lógica que permiten completar algunos de los procesos.
- **Runners:** Contiene las clases que permiten ejecutar los features con relación uno a uno entre feature y clase.
- **Stepdefinitions:** Contiene las clases que implementan los pasos utilizados en los features a partir de los cuales se pueden invocar tareas.
- **Features:** Contiene las historias de usuario en lenguaje gherkin con narrativa declarativa en términos de negocio.

- **Data:** Contiene archivos que pueden ser insumo de datos para ejecutar algunas pruebas.

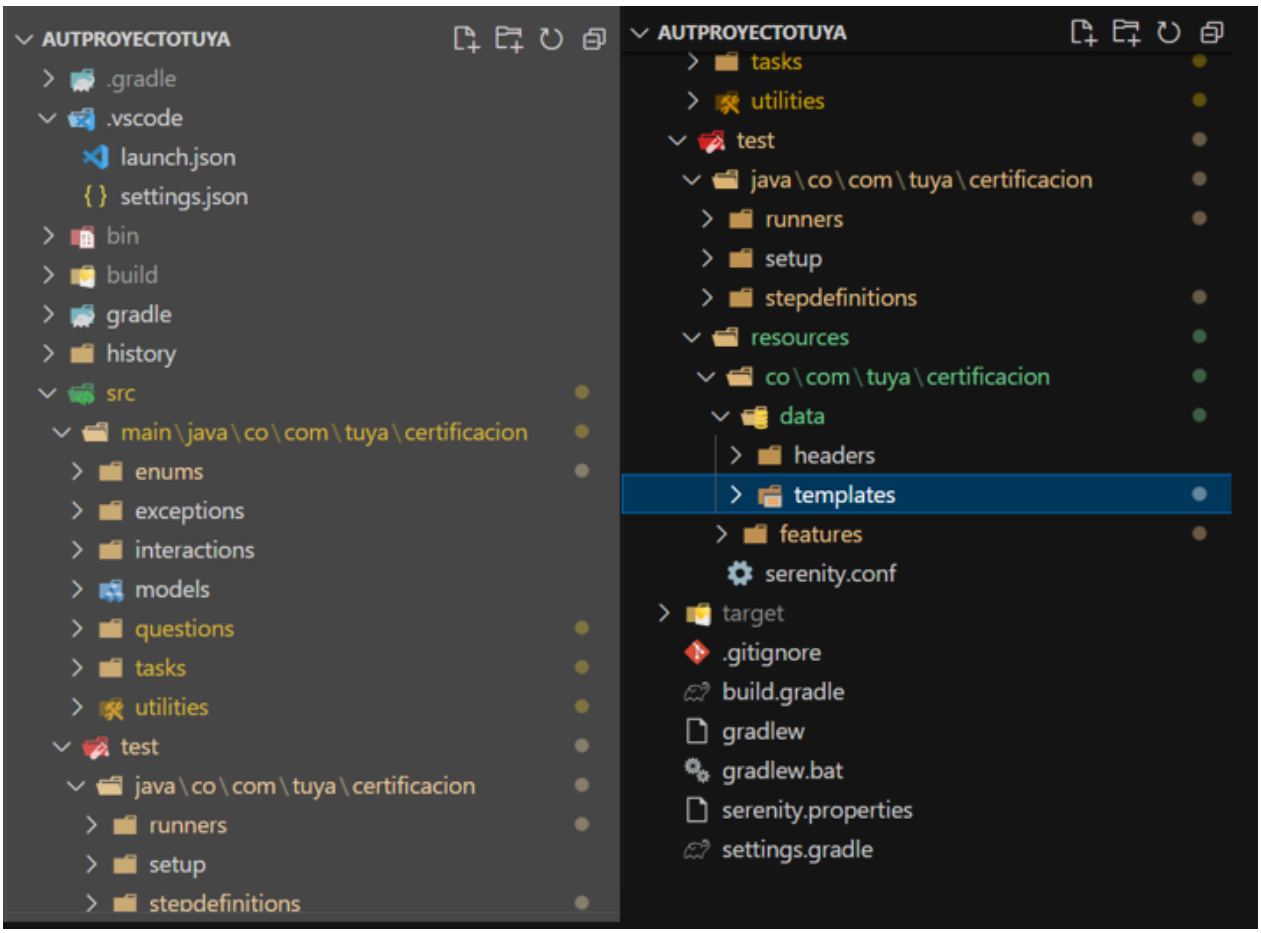
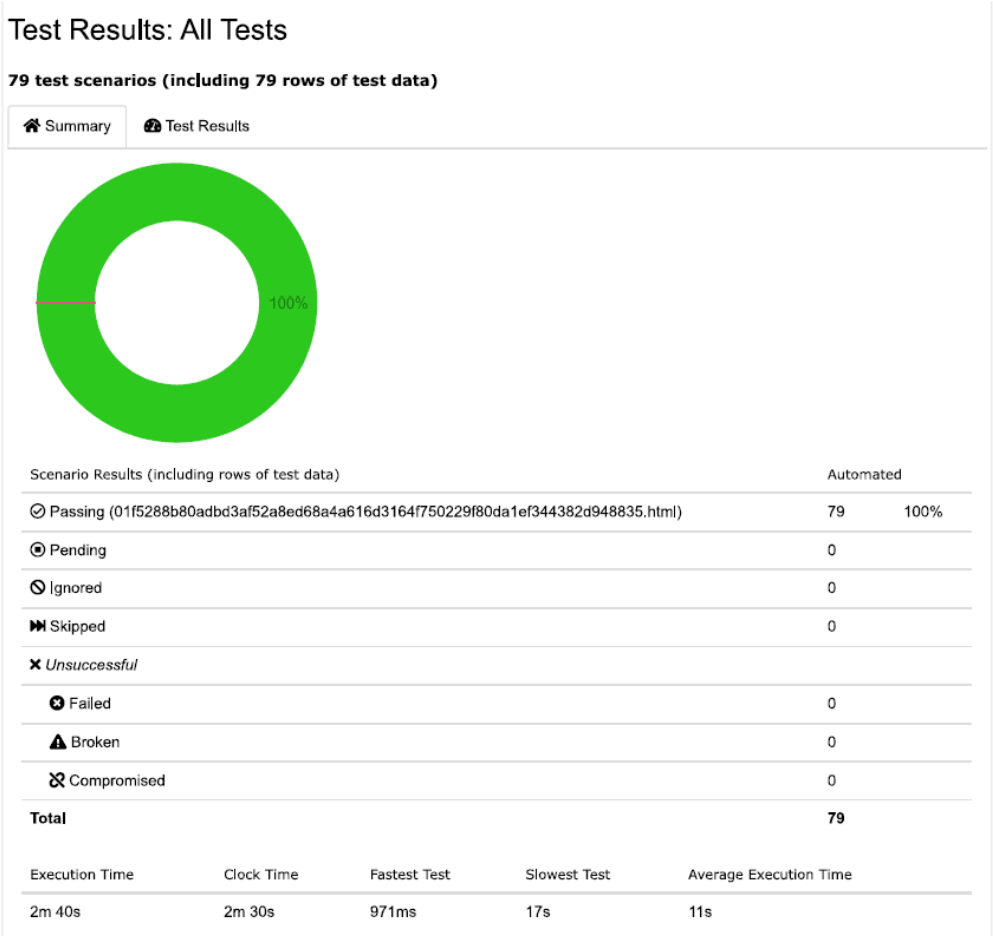


Fig. 2. Estructura del proyecto

Durante la fase de pruebas, se evidenció que las APIs tenían algunos errores comunes que retrasaron un poco el desarrollo de las pruebas, se identificaron errores como la dificultad a la hora de validar datos, ya que algunos de ellos no estaban disponibles, o carecían de la información necesaria para la ejecución de las pruebas. Otra de las dificultades que se tuvieron fue la necesidad de datos de pruebas.

Luego de haber solucionado los impedimentos anteriormente mencionados, continuamos con la automatización de pruebas, la cual se implementó caso por caso evidenciando que los datos de prueba recopilados se ejecutarán correctamente en las pruebas asignadas. a continuación en las siguientes figuras se muestra como las pruebas pasaron de forma completa.



Test Results: All Tests

79 test scenarios (including 79 rows of test data)

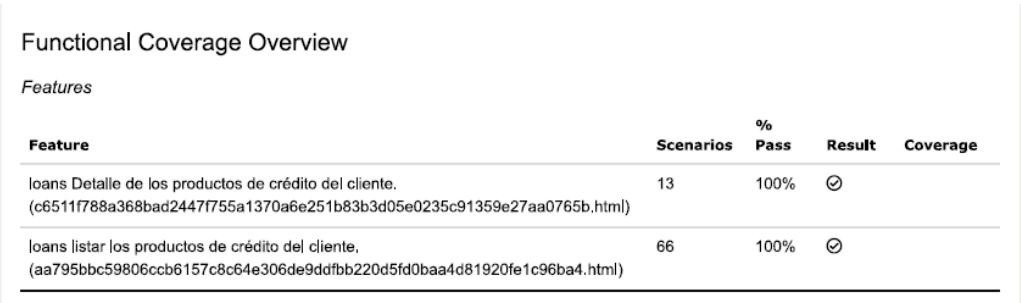
Summary | Test Results

100%

Scenario Results (including rows of test data)	Automated	
Passing (01f5288b80adbd3af52a8ed68a4a616d3164f750229f80da1ef344382d948835.html)	79	100%
Pending	0	
Ignored	0	
Skipped	0	
Unsuccessful		
Failed	0	
Broken	0	
Compromised	0	
Total	**79**	

Execution Time	Clock Time	Fastest Test	Slowest Test	Average Execution Time
2m 40s	2m 30s	971ms	17s	11s

Fig. 3. Resultados del test de Serenity



Functional Coverage Overview

Features

Feature	Scenarios	% Pass	Result	Coverage
loans Detalle de los productos de crédito del cliente. (c6511f788a368bad2447f755a1370a6e251b83b3d05e0235c91359e27aa0765b.html)	13	100%	✓	
loans listar los productos de crédito del cliente. (aa795bbc59806ccb6157c8c64e306de9ddfbb220d5fd0baa4d81920fe1c96ba4.html)	66	100%	✓	

Fig. 4. Cobertura de las pruebas realizadas

Como podemos observar en detalle en la Figura 3 y Figura 4, se presenta visualmente la cantidad de pruebas ejecutadas conforme al riguroso plan de pruebas establecido. En este análisis, se destaca que se llevaron a cabo un total de 79 escenarios, divididos en 13 escenarios de detalle de productos y 66 de lista de producto, los cuales tienen una cobertura completa en su ejecución.

VII. CONCLUSIONES

La automatización de pruebas del software desarrollado para el equipo de canales de Recaudo se llevó a cabo con el objetivo de mejorar tanto la eficiencia como la calidad en el proceso de desarrollo de este equipo, garantizando una cobertura exhaustiva de escenarios. Durante este proceso, se logró cumplir con lo desarrollado, respaldado por un reporte de SonarLint que demostró un 80% de cobertura. A pesar de enfrentar algunas dificultades, que impidieron cumplir con todos los casos de prueba previstos, la implementación de esta automatización de pruebas se presenta como una estrategia clave para optimizar el ciclo de desarrollo, contribuyendo así a resultados más eficientes y a una mayor calidad en el producto final.

Se reconoce la importancia de tener una buena fase de análisis con la cual seleccionar cuidadosamente los casos de prueba que se van a desarrollar, con el fin de agilizar el proceso de pruebas garantizando un enfoque eficiente y efectivo.

En mi experiencia personal durante este periodo de prácticas, he adquirido diversas habilidades, como un aprendizaje constante sobre metodologías ágiles, la implementación de pruebas y el desarrollo de habilidades en equipo. Lo que más agradezco es que me brindaron una base sólida para continuar creciendo en mi carrera profesional, tanto como analista de software como persona.

REFERENCIAS

[1] IEEE 12207-2-2020—ISO/IEC/IEEE International Standard—Systems and Software Engineering—Software life Cycle Processes—Part 2: Relation and Mapping between ISO/IEC/IEEE 12207:2017 and ISO/IEC 12207:2008. 2020. Available online: https://www.iso.org/standard/73446.html (accessed on 14 January 2024).

[2] IEEE 829: IEEE Standard for Software Test Documentation. (2011), http://www.coleyconsulting.co.uk/IEEE829.htm (accessed on 14 January 2024).