UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

FACULTAD DE INGENIERÍA

PROGRAMA ACADÉMICO DE INGENIERÍA DE SISTEMAS

Sistema para automatizar pruebas de regresión mediante el uso de lenguaje de programación typescript

TRABAJO DE SUFICIENCIA PROFESIONAL

Para optar el título profesional de Ingeniero de Sistemas

AUTOR(ES)

Quispe Irpanocca, Delia Juana	0000-0002-5793-829X
Puchulan Madrid, Cesar Ruben	0000-0003-2073-0193

ASESOR(ES)

Burga Durango, Daniel Wilfredo	0000-0003-0312-727X

Lima, 13 de noviembre de 2023

DEDICATORIA

Dedico este trabajo a mi madre Carmen, quien siempre me ha apoyado y alentado en esforzarme cada día.

Cesar Puchulán

Se lo dedico este trabajo a estas personas Blanca F. Yolanda y Belén por siempre creer en mi y motivarme a seguir adelante. A mi hijito que siempre me robarme una sonrisa y por ser mi fuerza.

Delia Quispe

AGRADECIMIENTOS

Agradecemos a nuestro asesor por el apoyo brindado en la elaboración de este documento y a las personas que con su experiencia se pudo realizar este trabajo.

RESUMEN

En el presente trabajo se realizó la automatización de las pruebas de regresión para un banco extranjero con sede en varios países. El banco ofrece sus productos financieros de manera digital, donde el cliente puede realizar sus operaciones bancarias a través de un aplicativo móvil tanto en Android, IOS y web. Debido a que se ha empezado a desplegar más funcionalidades a producción y el sistema se está volviendo más complejo, las validaciones de los casos de prueba de regresión están tomando más tiempo de lo debido. Esto genera que los pases a producción se tengan que retrasar y a su vez ocasiona que el equipo de pruebas tenga que hacer retrabajo sobre casos de prueba validados.

Por esta razón, se desarrolló un sistema para la automatización de pruebas de regresión con el fin de reducir el tiempo de las ejecuciones de los casos de prueba.

Para poder realizar la automatización de las pruebas de regresión, primero se instaló software como Visual Studio Code, Node, JDK, Python y GIT para el funcionamiento del sistema. Luego se procedió con la redacción de los casos de prueba en base a Gherkins, después se realizó el desarrollo de los scripts de prueba con TypeScript y como parte final del trabajo se validó que el sistema pueda ejecutar de manera automática los casos de prueba. El sistema permite ejecutar de manera más rápida los casos de prueba y se pueda detectar errores en el aplicativo móvil de manera más rápida.

El resultado que se obtuvo con el sistema fue que se pueda reducir el tiempo para las pruebas de regresión y la carga de trabajo se reduzca para el equipo de pruebas, de manera que puedan realizar un trabajo más eficiente.

Palabras claves**:** Banca; automatización; pruebas de regresión; TypeScript

ABSTRACT

In this work, the automation of regression tests was carried out for a foreign bank based in several countries. The bank offers its financial products digitally, where the client can carry out their banking operations through a mobile application on Android, IOS and the web. Because more functionalities have begun to be deployed to production and the system is becoming more complex, validations of regression test cases are taking longer than they should. This means that production passes must be delayed and in turn causes the testing team to have to do rework on validated test cases. For this reason, a system was developed for the automation of regression tests to reduce the time of test case executions.

In order to automate regression testing, software such as Visual Studio Code, Node, JDK, Python and GIT were first installed for the system to function. Then we proceeded with writing the test cases based on Gherkins, then the development of the test scripts was carried out with TypeScript and finally it was validated that the system can automatically execute the test cases. The system allows test cases to be executed more quickly and errors can be detected in the mobile application more quickly.

The result obtained with the system was that the time for regression testing can be reduced and the workload is reduced for the testing team, so that they can do more efficient work.

Keywords: Bank; automation; Regression tests; TypeScript

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1: DEFINICIÓN DEL PROYECTO

En este capitulo se abarca los antecedentes, la problemática, los objetivos e indicadores planteados para la elaboración del proyecto. También, este capítulo contiene el diagrama de Gantt detallado con sus actividades y los costos referenciales del proyecto.

Antecedentes

La forma de manejar las finanzas personales cambio en la última década. La transformación digital del sistema financiero se inició en épocas previas a la pandemia del COVID 19, pero esto forzó a que la digitalización sea una necesidad. Los canales, billeteras y la banca digital empezaron a tomar protagonismo y transformaron la forma de manejar sus finanzas. Según Asbanc, el número total de operaciones por canales virtuales a octubre 2022 fue de 2,611 millones y representa un crecimiento de 108 %, respecto al acumulado alcanzado a octubre 2021. Por su parte, Julio Velarde, presidente del Banco Central de Reserva del Perú (BCRP), afirmó que los pagos digitales en Perú crecieron un 270% (Banco Bilbao Vizcaya Argentaria [BBVA], 2023).

Por otra parte, en América latina ha aumentado el crecimiento de las empresas que ofrecen servicios financieros de forma digital. El BCRP menciona que el incremento del tamaño, cantidad y alcance de las empresas que ofrecen servicios financieros de manera digital responde a una tendencia que se observa desde hace varios años a nivel global. Entre estas compañías se incluye a los neobancos y challengerbanks, cuya presencia también ha estado desarrollándose en la región (Banco Central De Reserva Del Perú [BCRP], 2022).

Tabla 1

Definición de bancos digitales

Modelo	Definición	Requiere licencia bancaria
Neobanco	Estas empresas brindan servicios financieros a través de canales digitales, en asociación con una institución financiera tradicional.	No
Challengerbank	Son bancos que brindan de forma digital el total (o la mayor parte) de sus servicios.	Si

	Elaboran servicios financieros creativos y se apoyan mediante desarrollos tecnológicos.	

Nota. Información al 06 de diciembre de 2023. Adaptado de "Reporte de Estabilidad Financiera", por Banco Central De Reserva Del Perú, 2022. (*https://www.bcrp.gob.pe/docs/Publicaciones/Reporte-Estabilidad-Financiera/2022/mayo/ref-mayo-2022-recuadro-4.pdf*).

De acuerdo con un informe de Payments & Commerce Market Intelligence se menciona que hoy, más del 80% de los latinoamericanos tienen acceso a una cuenta de depósito, lo que los convierte en participantes oficiales de la economía financiera digital. P2P actúa como una herramienta poderosa para ganar masa crítica y evolucionar hacia plataformas de pago más exitosas. Yape en Perú tiene más de 10 millones de usuarios, equivalente al 40% de los adultos peruanos (Payments & Commerce Market Intelligence [PCMI], 2023).

Las empresas están adoptando por automatizar sus procesos y optimizar sus recursos y tiempos de producción. Por otra parte, el diario peruano menciona que en América Latina las empresas están utilizando incluso inteligencia artificial en sus datos y un 34% afirman que está usando la automatización incluso para la toma de decisiones (El Peruano, 2023).

Según una investigación realizado por Fernández (2018) recalca que automatizar los procesos reducen el tiempo y aumenta la productividad para mayor beneficio del Banco de crédito, confirmando así que la automatización de procesos mejora las pruebas de software en el área de calidad del Banco de Crédito, reduciendo el tiempo de generación de data de prueba en un 56.78%, aumentando la cantidad de data generada en un 7.71%, por lo tanto, se concluye que la automatización de procesos permitió la mejora de manera significativa las pruebas de software.

(Lucero, 2022) de acuerdo con un estudio realizado en la implementación de un framework de automatización en Jenkins para las pruebas de regresión permitió que se despliegue las historias de usuario en el entorno de pruebas, se realicen las pruebas manuales propios de los cambios realizados y se ejecuten el Job de Jenkins para las pruebas de regresión. Reduciendo el tiempo de ejecución, logrando una mayor cobertura de pruebas y garantizando la calidad necesaria del producto. Por otro lado, Acosta (2023), cita según un análisis realizado

por la empresa Deloitte la automatización crece en un 40.6% cada año. Entre los principales beneficios encontrados se resaltó la reducción de esfuerzo y tiempo en las pruebas de regresión.

Descripción de la Organización

El banco es una institución financiera con presencia en los principales mercados como España, Colombia, México y Perú. En el Perú fue fundado en el año 1951 y ofrece productos financieros a sus clientes como cuentas de ahorro, tarjetas de crédito, prestamos financieros e inversiones. El banco se ha visto en la necesidad de digitalizar sus procesos debido a la pandemia y así poder atender las necesidades de sus clientes. En la actualidad más de 3.3 millones de clientes digitales usan el aplicativo del banco. Pueden realizar transacciones en línea como transferencias, pago de servicios, consultar el movimiento de sus cuentas, configuración de tarjetas de crédito, apertura cuentas de ahorro, solicitar préstamos, etc.

Misión

Entidad del rubro financiero, tiene como compromisos prioritarios poder atender las necesidades de los clientes y prestar valor al patrimonio de los accionistas, al mismo tiempo que piensa en el desarrollo en las sociedades en las que está presente.

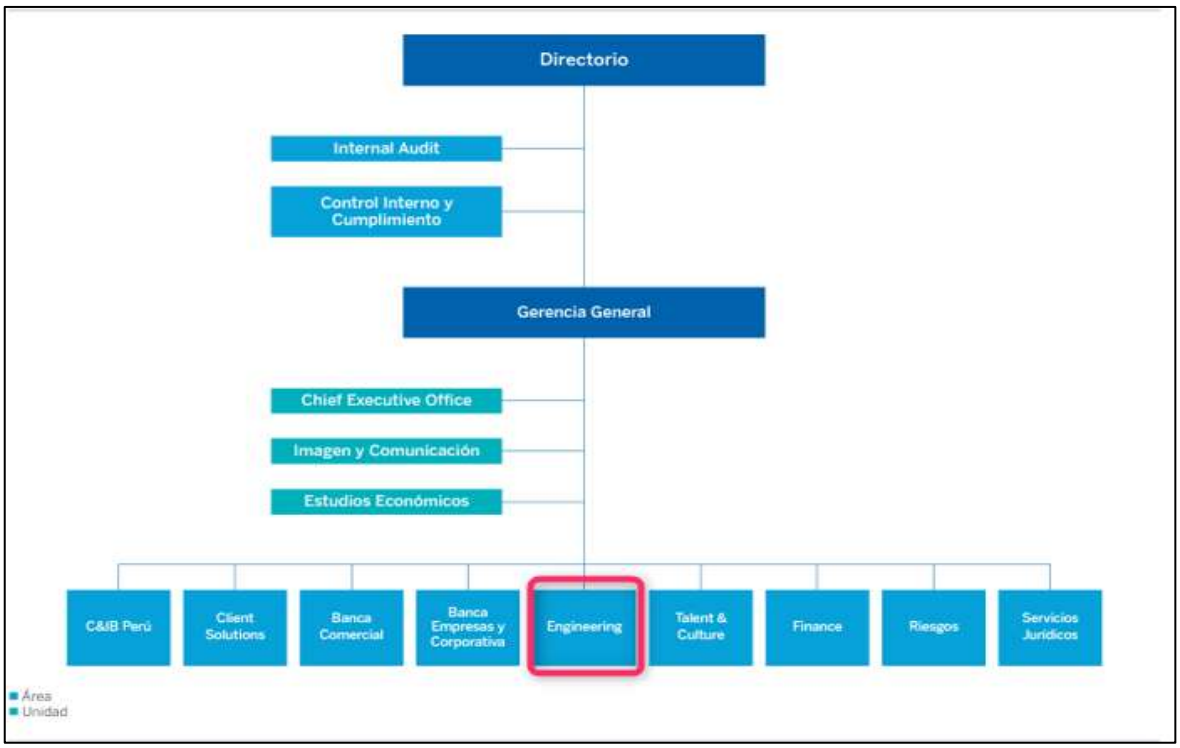
Visión

El banco busca una mejora continua para las personas, en base a los siguientes pilares;

- Integridad
- Prudencia
- Transparencia

A continuación, en la Figura 1 se muestra el organigrama del banco, el estudio del proyecto se centra en el área de Engineering, que esta resaltado con un marco en rojo.

Figura 1
Organigrama general del banco en estudio



Nota. La figura muestra el organigrama del BBVA. De "Organigrama y estructura de gobierno", por BBVA, 2022. (https://www.bbva.pe/content/dam/public-web/peru/documents/personas/memoria-anual/Memoria-Anual-2022.pdf)

Análisis del Problema

El proceso de aseguramiento de calidad comienza con el análisis de los requerimientos de software, de acuerdo con ello se elabora casos de pruebas para validar las nuevas funcionalidades del sistema en desarrollo. Luego que la nueva funcionalidad del aplicativo se encuentra instalada en el ambiente de calidad, se procede a ejecutar los casos de prueba para validar esta nueva funcionalidad. Una vez cerrado el ciclo de pruebas para validar la nueva funcionalidad se procede a ejecutar los casos de pruebas de no impacto o regresión para validar que las demás funcionalidades existentes no hayan sido impactadas por el nuevo desarrollo.

Uno de los principales problemas detectados en el área de calidad es el tiempo para ejecutar las pruebas de no impacto, las cuales dedican más tiempo de lo estimado. Esto sucede debido a que el aplicativo se vuelve más complejo porque se va desplegando nuevas funcionalidades a producción y se tiene que realizar el mantenimiento de estas funcionalidades

existentes. Además, se genera un retrabajo para el personal de pruebas, debido que esto implica validar de nuevo las funcionalidades que ya han pasado a producción. Como consecuencia de ello se tiene que emplear más tiempo para la validación de las pruebas de no impacto y esto también genera que se pospongan las fechas de pase a producción.

Se presenta en la tabla 2 los antecedentes de la problemática.

Tabla 2

Problema y antecedentes del caso de estudio

Problema	Antecedentes
Necesidad de automatizar los casos de pruebas de regresión en los sistemas	• Según un informe del Banco Central de Reserva del Perú (BCRP) las operaciones a través de banca móvil y páginas web entre junio de 2020 y junio de 2021 se incrementaron en un 136,7% y 18,2%. Asimismo, cada vez más usuarios prefieren usar soluciones digitales (BCRP, 2021). • De acuerdo con un informe de la empresa Deloitte, el 74% de los directores de sistema de información (CIO) que lograron implementar un sistema automatizado para las tareas repetitivas permitió que el trabajo de los colaboradores sea más eficiente. Asimismo, se indica que el 59% de los CIO reportaron una reducción de costos de producción de hasta un 30% (Deloitte, 2022).

Objetivos

Para poder efectuar el desarrollo de proyecto se planteó el objetivo general y 4 objetivos específicos que serán como pasos para poder concluir este trabajo. A continuación, se muestran los objetivos propuestos:

General

Implementar un sistema que permita automatizar casos de prueba mediante el uso de lenguaje de programación TypeScript para las pruebas de regresión de un banco.

Específicos

OE1: Analizar sobre tecnologías libres relacionadas a la automatización de pruebas.

OE2: Diseñar la arquitectura física y lógica del sistema automatizado de pruebas.

OE3: Validar que el sistema ejecute los casos de prueba de manera automatizada.

OE4: Realizar un plan de continuidad para el sistema automatizado de pruebas.

Indicadores de Éxito

Para poder cumplir con los objetivos propuestos se establece los siguientes indicadores como se visualiza en la tabla 3.

Tabla 3

Indicadores de éxito para los objetivos propuestos

Indicadores de éxito		Objetivo
IE01	Acta de aprobación sobre las tecnologías que existen en el mercado validado por el asesor.	OE01
IE02	Acta de aprobación en base a la arquitectura elaborada validado por parte del asesor.	OE02
IE03	Acta de conformidad que valide el funcionamiento del sistema de parte del asesor.	OE03
IE04	Acta de conformidad que valide el plan de continuidad por parte del asesor.	OE04

Planificación del Proyecto

El tiempo planificado para la ejecución del proyecto tuvo una duración de 3 meses y se trabajó aplicando el marco ágil de Scrum. El proyecto se dividió en 6 Sprint con una duración de 2 semanas cada uno. Teniendo en cuenta los eventos, roles y artefactos para el proyecto implementado.

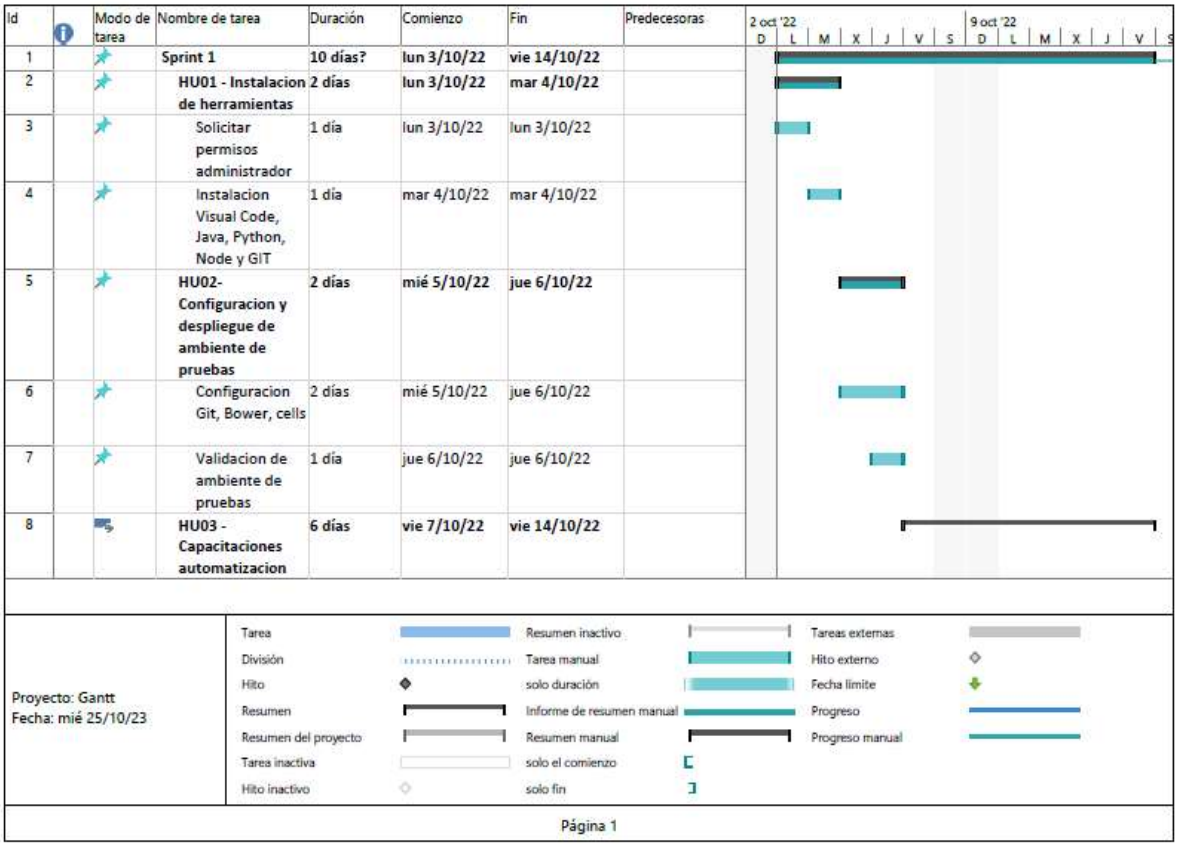
Diagrama de Gantt

En las siguientes figuras se detallan las actividades realizadas en los 6 Sprints destinados para el proyecto.

En el primer sprint se realizó las instalaciones de las herramientas de trabajo y la configuración del ambiente de pruebas automatizado. También, se llevó a cabo una capacitación a los QA para que puedan desarrollar los scripts automatizados. Al culminar el sprint se tuvo como entregable el ambiente instalado y el personal QA capacitado.

Figura 2

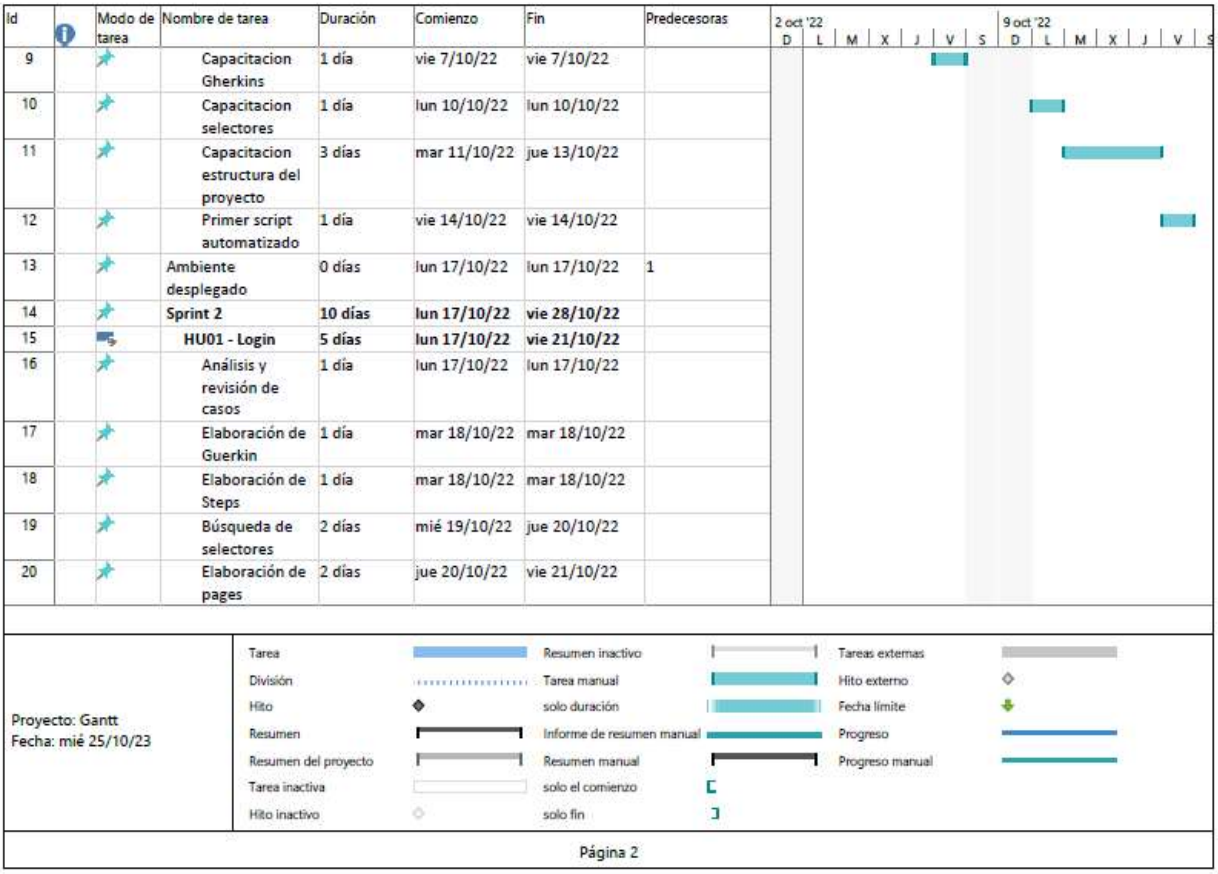
Diagrama de Gantt - Parte 1



En el segundo sprint se elaboró los scripts de automatización de las funcionalidades de Loguin y consultar detalle de producto. Al culminar el sprint se tuvo como entregable las funcionalidades automatizadas.

Figura 3

Diagrama de Gantt - Parte 2

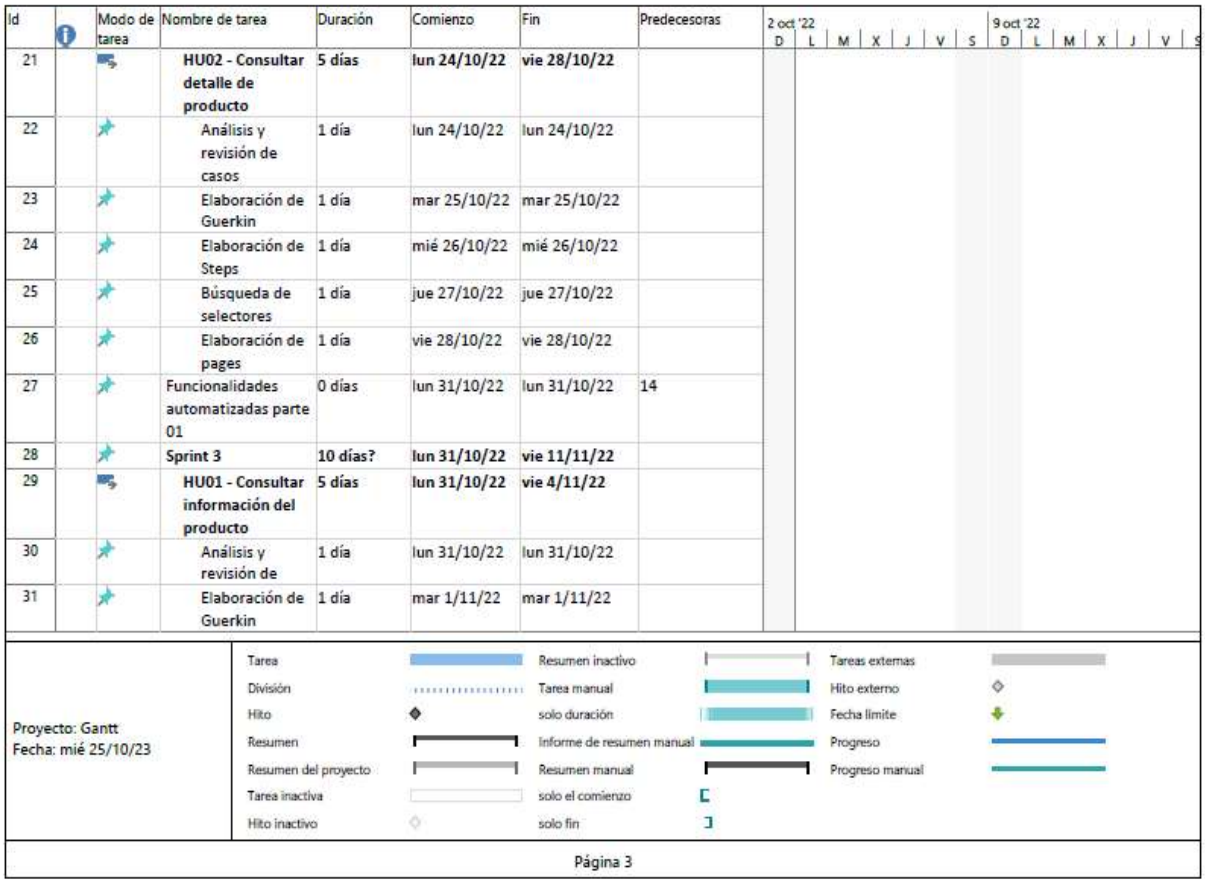


Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
9		Capacitacion Gherkins	1 día	vie 7/10/22	vie 7/10/22	
10		Capacitacion selectores	1 día	lun 10/10/22	lun 10/10/22	
11		Capacitacion estructura del proyecto	3 días	mar 11/10/22	jue 13/10/22	
12		Primer script automatizado	1 día	vie 14/10/22	vie 14/10/22	
13		Ambiente desplegado	0 días	lun 17/10/22	lun 17/10/22	1
14		**Sprint 2**	**10 días**	**lun 17/10/22**	**vie 28/10/22**	
15		**HU01 - Login**	**5 días**	**lun 17/10/22**	**vie 21/10/22**	
16		Análisis y revisión de casos	1 día	lun 17/10/22	lun 17/10/22	
17		Elaboración de Guerkin	1 día	mar 18/10/22	mar 18/10/22	
18		Elaboración de Steps	1 día	mar 18/10/22	mar 18/10/22	
19		Búsqueda de selectores	2 días	mié 19/10/22	jue 20/10/22	
20		Elaboración de pages	2 días	jue 20/10/22	vie 21/10/22	

En el tercer sprint se elaboró los scripts de automatización de las funcionalidades de consultar información del producto y validación del producto. Al culminar el sprint se tuvo como entregable las funcionalidades automatizadas.

Figura 4

Diagrama GANTT - Parte 3



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
21		**HU02 - Consultar detalle de producto**	**5 días**	**lun 24/10/22**	**vie 28/10/22**	
22		Análisis y revisión de casos	1 día	lun 24/10/22	lun 24/10/22	
23		Elaboración de Guerkin	1 día	mar 25/10/22	mar 25/10/22	
24		Elaboración de Steps	1 día	mié 26/10/22	mié 26/10/22	
25		Búsqueda de selectores	1 día	jue 27/10/22	jue 27/10/22	
26		Elaboración de pages	1 día	vie 28/10/22	vie 28/10/22	
27		Funcionalidades automatizadas parte 01	0 días	lun 31/10/22	lun 31/10/22	14
28		**Sprint 3**	**10 días?**	**lun 31/10/22**	**vie 11/11/22**	
29		**HU01 - Consultar información del producto**	**5 días**	**lun 31/10/22**	**vie 4/11/22**	
30		Análisis y revisión de	1 día	lun 31/10/22	lun 31/10/22	
31		Elaboración de Guerkin	1 día	mar 1/11/22	mar 1/11/22	

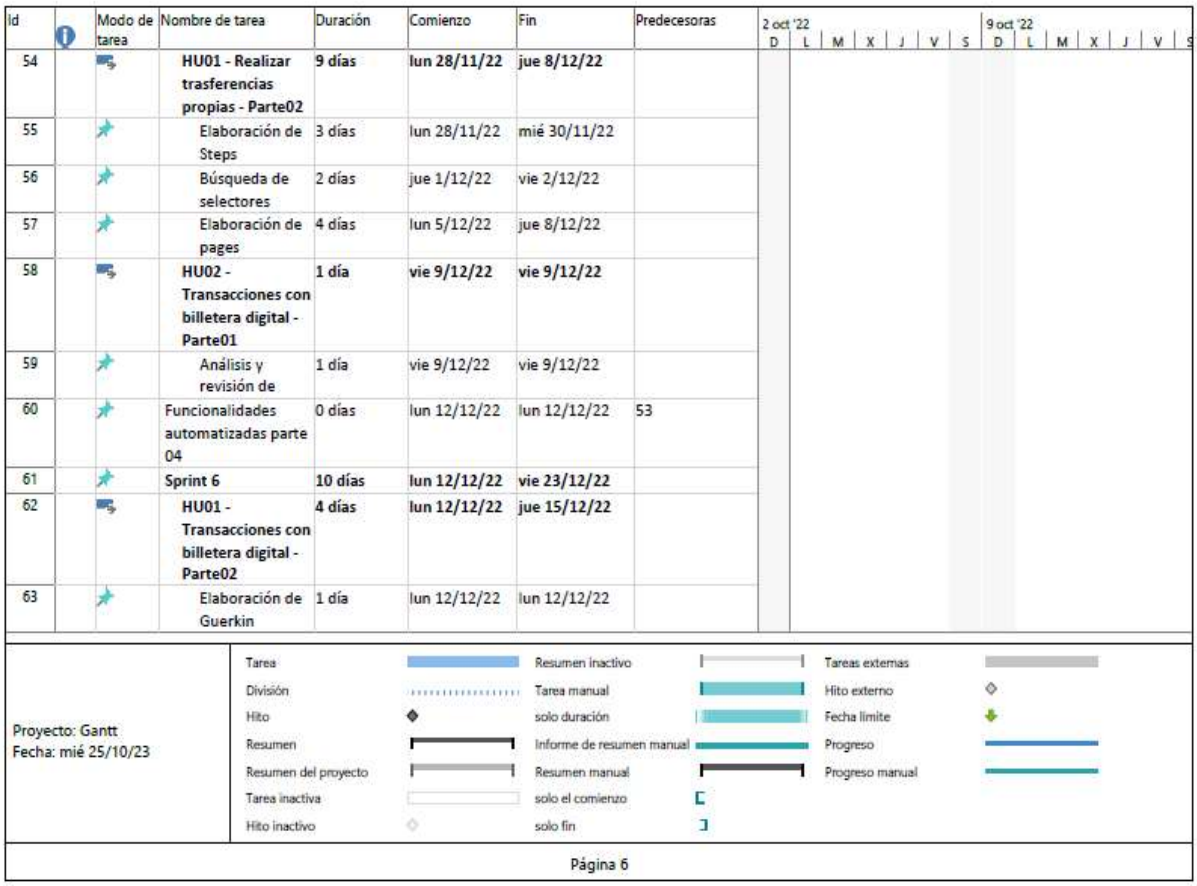
En el cuarto sprint se elaboró los scripts de automatización de la funcionalidad de pagar tarjeta propia y parte de la funcionalidad de realizar transferencias propias. Al culminar el sprint se tuvo como entregable una funcionalidad automatizada.

En el quinto sprint se continuo con la elaboración de los scripts de automatización de la funcionalidad de realizar transferencias propias y parte de la funcionalidad de transacciones con billetera digital. Al culminar el sprint se tuvo como entregable una funcionalidad automatizada.

En el sexto y último sprint se continuo con la elaboración de los scripts de la funcionalidad de realizar transacciones con billetera digital y se culminó con la funcionalidad de cambio de venta y compra de moneda. Al culminar el sprint se tuvo como entregable las funcionalidades automatizadas.

Figura 5

Diagrama GANTT - Parte 4



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
54		**HU01 - Realizar trasferencias propias - Parte02**	**9 días**	**lun 28/11/22**	**jue 8/12/22**	
55		Elaboración de Steps	3 días	lun 28/11/22	mié 30/11/22	
56		Búsqueda de selectores	2 días	jue 1/12/22	vie 2/12/22	
57		Elaboración de pages	4 días	lun 5/12/22	jue 8/12/22	
58		**HU02 - Transacciones con billetera digital - Parte01**	**1 día**	**vie 9/12/22**	**vie 9/12/22**	
59		Análisis y revisión de	1 día	vie 9/12/22	vie 9/12/22	
60		Funcionalidades automatizadas parte 04	0 días	lun 12/12/22	lun 12/12/22	53
61		**Sprint 6**	**10 días**	**lun 12/12/22**	**vie 23/12/22**	
62		**HU01 - Transacciones con billetera digital - Parte02**	**4 días**	**lun 12/12/22**	**jue 15/12/22**	
63		Elaboración de Guerkin	1 día	lun 12/12/22	lun 12/12/22	

Para mayor detalle del diagrama de Gantt se puede revisar en el anexo1.

Costo del proyecto

En cuanto al costo del proyecto solo se indica de manera referencial debido que esta información es de carácter sensible y no puede ser evidenciado en este trabajo. Sin embargo, se detalla en la tabla 4 los costos estimados, tiempo de duración y roles asignados.

Tabla 4
Costos referenciales por Rol

Recurso requerido	Costo por mes	Meses	Costo total
QA Lead	S/. 7,000	3	S/. 21,000
Scrum Master	S/. 6,500	3	S/. 19,500
QA funcional	S/. 4,000	3	S/. 12,000
QA Automatizador de prueba 1	S/. 6,000	3	S/. 18,000
QA Automatizador de prueba 2	S/. 6,000	3	S/. 18,000
		Total	S/. 88,500

También se considera los costos de los equipos, servicio en la nube (repositorio de versiones) y gastos administrativos. Teniendo en cuenta que solo se refleje en este informe lo que se utilizó para el desarrollo del proyecto.

Tabla 5
Otros costos del equipo de trabajo

Equipos	Costo x Unidad	Cantidad	Depreciación del equipo x 1	Costo total
Marca de laptop Lenovo Procesador: Intel Core i7-4.7 GHz 11va Generación Memoria RAM: 20 GB DDR4 / Disco sólido SSD 512GB PCIe. Pantalla LED 17.3	S/. 3,500	3	1,680	S/. 10,500
			Total	S/. 10,500

En la tabla 6 se visualiza el costo de servicio en la nube para el repositorio versiones en Bitbucket

Tabla 6
Costo de servicio en la nube

Equipos	Costo x mes	# usuario	Costo total
Servicio en la nube	S/. 56	3	S/. 168
		Total	S/. 168

Solo se tuvo en cuenta tres laptops debido que tres han tenido una participación de un 80% en el proyecto. Cabe recalcar también que los costos son referenciales, dando como resultado total de S/. 99,168.

CAPÍTULO 2: MARCO TEÓRICO

Este capítulo tiene como objetivo presentar conceptos teóricos acerca de herramientas utilizadas en el sistema automatizado con el fin de poder comprender y poder desarrollar temas específicos del proyecto en estudio. Asimismo, se presenta framework y lenguajes de programación que se utilizan para la automatización de pruebas. También, se define las bases legales y marco normativos que está sujeto el banco en estudio en la entrega de sus productos a sus clientes o usuarios.

Marco Conceptual

A continuación, se presenta conceptos básicos utilizados para el desarrollo del proyecto.

Pruebas de software

International Business Machines Corporation (IBM, s.f.) se refiere a las pruebas de software como el proceso de evaluar y revisar que un producto o aplicación de software haga lo que está considerado que deba hacer. Entre los beneficios abarca la prevención de errores, la disminución de los costos de desarrollo y la mejora de rendimiento.

Se realizan diversos tipos de pruebas de software, cada una con una finalidad y planificación específicos:

- Prueba de aceptación: Verificamos que el sistema funcione según el requerimiento del cliente.
- Pruebas de integración: Certifica que los componente o aplicaciones del software se ejecuten de manera conjunta.
- Pruebas de unidad: Valida que cada unidad de software funcione según el requerimiento del negocio o cliente.
- Pruebas funcionales: Revisa funciones a través de la emulación de escenarios de negocio, en base a los requerimientos funcionales.

- Pruebas de rendimiento: Prueba de cómo funciona el software bajo diferentes cargas de trabajo.
- Pruebas de regresión: Verifica que las nuevas características no rompan o degraden las funcionalidades existentes.
- Pruebas de estrés: Prueba cuanta estrés puede mantener el sistema antes que colapse.
- Pruebas de usabilidad: Valida que tan bien el cliente puede usar un sistema o aplicación para realizar una tarea (IBM, s.f.).

Visual Studio Code

Es la herramienta para editar código fuente, ligero pero eficaz que se ejecuta en el escritorio, cuenta con una consola integrada y atajos. "Está disponible para Windows, macOS y Linux. Además, incluye compatibilidad integrada con JavaScript, TypeScript y Node.js, y tiene un amplio ecosistema de extensiones para otros lenguajes (como C++, C#, Java, Python, Go, .NET)" (Microsoft Visual Studio Code, 2023, párr. 1).

Java JDK (Java Development Kit)

IBM (2021):

> Java Development Kit o JDK es un software para los desarrolladores de Java. Incorpora el intérprete Java, clases Java y herramientas de desarrollo Java (JDT): compilador, depurador, desensamblador, visor de applets, generador de archivos de apéndice y generador de documentación.
>
> El JDK permite escribir aplicaciones que se desarrollan una sola vez y se ejecutan en cualquier lugar de cualquier máquina virtual Java. Las aplicaciones Java desarrolladas con el JDK en un sistema se pueden usar en otro sistema sin tener que cambiar ni recompilar el código. Los archivos de clase Java son portables a cualquier máquina virtual Java estándar (párr. 1-2).

Node

Node.Js es una tecnología de código abierto y multiplataforma, que permite ejecutar y crear a los desarrolladores herramientas por el lado del servidor y aplicaciones en JavaScript. El cual permite una ejecución en tiempo real que puede ser ejecutado en una computadora o en

un sistema operativo del servidor. Pero el ambiente omite las ApIs de JavaScript específicas de los exploradores webs, añadiendo soporte APIs para sistemas operativos más tradicionales como HTTP y bibliotecas del sistema fichero (MDN Web Docs, 2023).

Ventajas:

Alto rendimiento, Node ha sido diseñado para optimizar el rendimiento y la escalabilidad en aplicaciones web y es un buen complemento para muchos problemas comunes de desarrollo web (ej, aplicaciones web en tiempo real).

El gestor de paquetes de Node (NPM del inglés: Node Packet Manager) que proporciona acceso a cientos o miles de paquetes reutilizables. Además, la mejor en su clase resolución de dependencias y puede usarse para automatizar la mayor parte de la cadena de herramientas de compilación.

Es portable, con versiones que funcionan en Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS, y NonStop OS. Además, soporta muchos proveedores de servidores webs, que proporcionan infraestructura específica y documentación. Tiene un ecosistema y comunidad de desarrolladores muy activa (MDN Web Docs, 2023, párr. 6, 9-11).

Python

Es un lenguaje de programación de código abierto, gratuito y se utiliza para el desarrollo de aplicaciones webs, desarrollo de software, sin embargo, es más empleada en ciencias de datos y machine learning (ML). Es de fácil de aprendizaje, eficiente y sobre todo se puede ejecutar en diversas plataformas (Amazon Web Services, s.f.).

Algunos beneficios que incluye son:

- Python posee una gran biblioteca estándar que comprende de códigos reutilizables para casi cualquier actividad. Cual facilita a los desarrolladores de no tener la necesidad de escribir el código desde cero.
- Python se pueden integrar fácilmente con otros lenguajes de programación como Java, C y C++.
- Python cuenta con una gran comunidad de soporte que brindan su apoyo. Cuenta con muchos recursos en internet desde manuales, videos y blogs.

- Cabe mencionar que Python es compatible con la mayoría de sistema operativos tales como Windows, macOS, Linux y Unix.

GIT

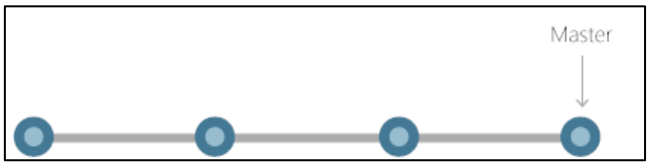
Software para el manejo del control de versiones que ayuda a gestionar el registro de cambios almacenados en un repositorio local o en la nube. Estos repositorios se pueden trabajar de manera remota o sin conexión. Esta herramienta funciona en diferentes sistemas operativas y entornos de desarrollo integrado (IDE). Además, cuenta con una arquitectura distribuida DVCS (sistema de control de versiones distribuido), lo que significa que cada copia del código de cada desarrollador es un registro que contiene el historial completo de todos los cambios del proyecto (Atlassian, s.f.).

Conceptos básicos de Git

Cada vez que se guarda el proyecto, GIT genera una confirmación. La vinculación de cada confirmación forma un historial de desarrollo. Esto permite revertir a una confirmación anterior para verificar los cambios que se realizaron de una confirmación a otra y poder revisar donde, cuando y como se efectuaron los cambios (Microsoft Learn, 2023).

Figura 6

Confirmarion GIT

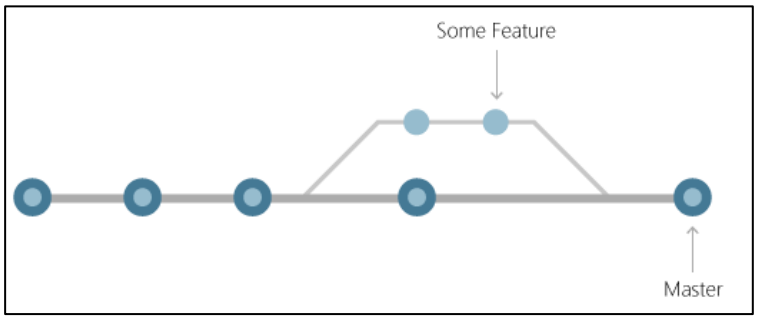


Nota. Adaptada De" Configuración de Git", por Microsoft, 2023. (https://learn.microsoft.com/es-es/devops/develop/git/what-is-git)

Ramas

Son versiones de código fuente de un proyecto, ya que cada desarrollador guarda sus cambios como resultado de los cambios realizados o confirmación. Mediante git podemos ramificar versiones como en máster, developer, test y subramas que al final siempre se va a unificar en la rama principal. Asimismo, Git proporciona herramientas para aislar los cambios y volver a combinarlos de manera posterior (Microsoft Learn, 2023).

Figura 7
Ramas GIT

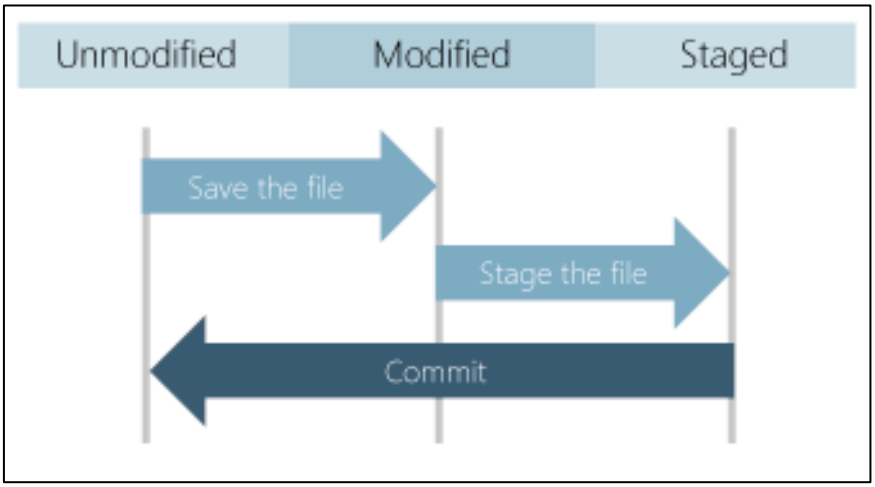


Nota, Adaptada De "Ramas Git", por Microsoft, 2023 (https://learn.microsoft.com/es-es/devops/develop/git/whatis-git)

Archivos y confirmaciones

Git maneja tres estados para los archivos: Modificado, almacenado temporal y confirmado. Si se realiza una modificación por primera vez, los cambios están de forma local y no forma parte de una confirmación o del historial de desarrollo. El almacenado temporal abarca todos los cambios que se introducirán en la siguiente confirmación. Cuando el desarrollador está conforme con los archivos alojados de forma temporal, los archivos se guardan como confirmación con un mensaje que detalla lo que se ha cambiado y esta confirmación pertenece al historial de cambio (Microsoft Learn, 2023).

Figura 8
Estados de archivos GIT



Nota. De "Archivos y confirmaciones", por Microsoft, 2023 (https://learn.microsoft.com/es-es/devops/develop/git/what-is-git)

Algunas ventajas que cuenta GIT son:

Desarrollo simultáneo

Cada desarrollador cuenta con una copia local de código y se puede trabajar de manera simultaneo en la rama que le corresponde. GIT puede funcionar sin conexión a internet, ya que las operaciones que se realizan son de manera local (Microsoft Learn, 2023).

Versiones más rápidas

Al separar por ramas el desarrollo se puede trabajar de manera flexible y simultánea. La rama principal abarca el código permanente y de alta calidad. Las subramas contienen el trabajo en desarrollo, la cuales se unen a la rama principal al finalizar. Es más fácil de administrar el código estable y enviar actualizaciones de manera más rápida al separar la rama de versión del desarrollo en curso (Microsoft Learn, 2023).

Integración incorporada

> Git puede integrarse con la mayoría de las herramientas y productos. Las IDE cuentan con compatibilidad integrada y diversas herramientas que permiten la integración continua, implementación continua, pruebas automatizadas, seguimiento de elementos de trabajo, métricas e integración de características de informes con GIT. Esto simplifica el flujo de trabajo diario (Microsoft Learn, 2023, párr. 11).

TypesScript

JavaScript es considerado uno de los lenguajes de programación más usados del mundo y se ha convertido oficial para desarrollo Web. Los desarrolladores lo usan para escribir aplicaciones multiplataforma que se pueden ejecutar en cualquier plataforma y explorador.

Aunque JavaScript se usa para crear aplicaciones multiplataforma, no está concebido para aplicaciones grandes que impliquen miles o incluso millones de líneas de código. JavaScript carece de algunas de las características de los lenguajes más maduros que se emplean en las sofisticadas aplicaciones de hoy en día. Para los editores de desarrollo integrado (IDE), puede ser todo un desafío administrar JavaScript y mantener estas grandes bases de código.

TypeScript aborda las limitaciones de JavaScript sin poner en peligro la propuesta de valor clave de JavaScript: la capacidad de ejecutar el código en cualquier lugar y en cualquier plataforma, explorador o host. Además, TypeScript es un lenguaje de código abierto desarrollado por Microsoft. Se trata de un supra conjunto de JavaScript, lo que significa que puede usar sus conocimientos actuales sobre JavaScript que ya ha desarrollado junto con determinadas características que antes no estaban disponibles.

La característica principal de TypeScript es su sistema de tipos. En TypeScript, puede identificar el tipo de datos de una variable o un parámetro mediante una sugerencia de tipo. Con las sugerencias de tipo, se describe la forma de un objeto, lo que proporciona una documentación mejor y permite a TypeScript validar que el código funciona correcto.

Mediante la comprobación de tipos estáticos, TypeScript al principio del desarrollo detecta problemas de código que JavaScript no puede detectar hasta que el código se ejecuta en el explorador. Los tipos también permiten describir lo que debe hacer el código (Microsoft Learn, s.f.).

Gherkins

Es un Lenguaje Específico de Dominio (DSL), que son lenguajes diseñados en concreto para resolver un problema muy específico. Como objetivo tiene resolver problema de comunicación entre los perfiles de negocio y los perfiles técnicos. Trabaja bajo el modelo de desarrollo dirigido por comportamiento (BDD), Además se comprende un aseria de pasos, escenarios y características (Profile Software Services, 2021).

Los elementos más utilizados son:

- **Feature:** Describe la funcionalidad a desarrollar. Se trata de un elemento de tipo característica en Gherkin, indica, los elementos que son de más alto nivel.
- **Escenario:** Una funcionalidad puede tener uno o varios escenarios, que no son otra cosa que distintas características (que pueden estar relacionadas o no). Los escenarios son elementos de tipo comportamiento.
- **Given:** aquí se expresa las precondiciones para que puedan ejecutar x acciones. Es un elemento de tipo acción.
- **When:** Se trata de las condiciones de las acciones que se van a ejecutar y es un elemento de tipo acción.

- **Then:** Es el resultado esperado de las acciones ejecutadas, también es un elemento de tipo acción (PSS, 2021, párr, 7-12).

Estándares, frameworks y buenas prácticas

A continuación, se menciona las principales herramientas de automatización, framework y lenguajes de programación que utilizan para agilizar las pruebas de regresión que existen en el mercado.

WebDriverIO

Es un framework utilizado para la automatización de pruebas progresivos de aplicaciones web y móvil, se puede trabajar de modo asíncrona y síncrona. Además, su configuración es flexible mediante línea de comando. Solo está disponible para JavaScript (TypeScript), debido a que se ofrece un conjunto de extensiones que contribuyen en crear una suite de pruebas estable, robusta y escalable. Es compatible con aplicaciones web modernas desarrolladas en Angular, React, Vue, Svelte u otros frameworks, aplicaciones web hibridas y aplicaciones móviles nativas. WebdriverIO puede ejecutarse en el Protocolo WebDriver para poder realizar pruebas verdaderas entre navegadores, así como Protocolo de Chrome DevTools para automatización basada en Chromium utilizando Puppeteer. Es de software libre, basado en estándares de web y cuenta con canal de comunidad (WebdriverIO,2023).

Playwright

Es un framework de open source desarrollado por Microsoft y fue lanzado en el año 2020, pero tuvo mucha acogida por parte de los desarrolladores. Es una herramienta que permite automatizar pruebas de extremo a extremo. Soporta navegadores como Chrome, Firefox, Chromium, WebKit Safari e incluso se puede realizar pruebas de aplicaciones Api Rest. Se destaca de otros framework porque en JavaScript su sintaxis es intuitiva y la flexibilidad con la que puede interactuar con el navegador en múltiples páginas y dominios. Además, ya viene con integración nativa. Por ejemplo, Playwright tiene imágenes Docker, lo que le permite ejecutar pruebas rápidas en un entorno aislado y controlado. Hay integraciones nativas disponibles para las mejores herramientas de CI/CD, incluidas GitHub Actions, Azure Pipelines, CircleCI, Jenkins y GitLab. Es compatibles con los ejecutores de pruebas de JavaScript existentes, como Jest/Jasmine, AVA y Mocha, lo cual es útil si está migrando desde

una base de código existente. Además, es compatible con sistemas operativos como Windows, Linux y macOS (Playwright, s.f.).

Cypress

Es una herramienta para automatización de pruebas en las aplicaciones webs modernas desarrollados como React, Angular y Vue.js. Permite escribir pruebas más rápido, más fácil y confiable a nivel de front-end.

Los tipos de pruebas que se pueden escribir utilizando cypress son:

- Configurar pruebas
- Escribir pruebas
- Ejecutar pruebas
- Pruebas de depuración

Ya que las aplicaciones modernas están desarrolladas con librerías de JavaScript, cual favorece para realizar todo tipo de pruebas.

- Pruebas de un extremo a otro
- Prueba de componente
- Prueba de integración
- Pruebas unitarias

Cypress es una aplicación gratuita, de código abierto e instalada de manera local y Cypress Cloud para registrar sus pruebas. Solo es compatible con los navegadores Chrome, Edge, Firefox y Safari por ahora, debido que se ejecuta de forma directa en el navegador (Cypress,2023).

SerenityDBB

Es una librería de código abierto para automatizar pruebas de regresión. Se puede utilizar para pruebas de aplicaciones webs, móviles tanto Android e IOS. Además, se puede trabajar con ScreenPlay que es un patrón de diseño para automatización de calidad, su comportamiento está orientado al formato Behaviour Driven Development (BDD).

Se mencionan algunas características:

- Principales características:
- Condigo abierto
- trabajar con DBB
- Mobile web, pruebas de App
- Pruebas de aplicaciones web
- Reporte automático
- documentación en ejecución
- Compatibilidad
- Cucumber
- Soporte para Maven, Gradle
- Soporta Multi Browser (Chrome, Firefox)
- Appium
- Permite la integración continua con Jenkins/Gitlab
- Integración con BrowserSctack/SauceLabs (SerenityBDD,s.f)

SeleniumBase

Es un framework de Python para automatizar pruebas en las aplicaciones web. Permite automatizar las pruebas de aceptación con calidad y rápido. Con flexibilidad y fácil de conservar las pruebas. Las pruebas de SeleniumBase se ejecutan mediante el pytest para pruebas de un extremo a otro, pruebas visuales, informes, gráficos,

- Utiliza de forma predeterminada el navegador Chrome si no se especifica un navegador.
- Las pruebas están estructuradas utilizando 23 formatos de sintaxis únicos.
- Las pruebas se pueden ejecutar con múltiples opciones de línea de comandos.
- Pruebas de interfaz de usuario más rápidas y expresivas: Escribe pruebas de IU limpias y expresivas más rápido con Serenity BDD y Selenium.
- Código robusto y reutilizable: Crea código de automatización sólido y reutilizable utilizando clases de acción y objetos de página, lo que reduce el tiempo de mantenimiento para pruebas inestables.
- Informes dinámicos: Genera informes completos y dinámicos desde la ejecución de su prueba, reduciendo el tiempo de documentación en cada versión.
- Informes más completos: Logre una mayor cobertura e informes más completos con pruebas basadas en datos en Serenity (BDD) (SerenityDBB, s.f.).

Bases Legales y Marco Normativo

Ley N° 26702 – Ley general del sistema financiero y del sistema de seguros y orgánica de la superintendencia de banca y seguros.

La presente ley constituye la norma para regular y controlar a las entidades que laboran en el sistema financiero y de seguros, asimismo en aquellas que efectúan actividades vinculadas o complementarias al objeto social de dichas personas (Congreso de la República del Perú, 1996, Ley N° 26702, Art. 1).

Ley N° 29733 – Ley de protección de datos personales

La vigente ley establece el propósito de garantizar el derecho primordial a salvaguardar los datos personales, fijado en el artículo 2 numeral 6 de la Constitución Política del Perú, a través de su adecuado tratamiento, en el contexto de respeto de los demás derechos esenciales que en ella se reconocen (Congreso de la República del Perú, 2011, Ley N. ª 29733, Art. 1).

CAPÍTULO 3: DESARROLLO DEL PROYECTO

El objetivo de este capítulo es abordar el análisis de las tecnologías y librerías utilizadas para la automatización de pruebas. Asimismo, se presentará la arquitectura física y lógica, diseño e implementación de la solución. Así como la instalación de herramientas y configuración para el desarrollo del proyecto. También se incluye los resultados obtenidos al finalizar la automatización de los pruebas y reporte.

Diseño de la Solución

A continuación, se presenta el diseño de la solución del sistema automatizado de pruebas.

Análisis de las herramientas de automatización

Uno de los objetivos propuesto es investigar y analizar sobre las tecnologías para automatizar pruebas de software que existe en el mercado actual. Para ello, se consideró tres frameworks con mayor popularidad y tres lenguajes de programación (LP) los más utilizados para automatizar pruebas de regresión. Como principales caracteríscas se considera la compatibilidad con los navegadores, código abierto, fácil de integración para automatizar

pruebas de integración, automatizar pruebas E2E en las aplicaciones web. A continuación, se presenta la Tabla 7, donde se detalla las características de las herramientas de automatización de software.

Tabla 7

Principales características de las herramientas de automatización de software

Descripción	**WebdriverIO**	**Playwright**	**Cypress**
Lenguajes que soporta	Java, Python, C #, PHP, Ruby, Perl y .Net y TypeScript	JavaScript, TypeScript, Python, Java y NET	Java, C#, Ruby, JavaScript, Golang, TypeScript
Flexibilidad de uso	Fácil de descargar y ejecutar los comandos para comenzar su uso	Tiene un solo controlador que es fácil de descargar ejecutar playwright install. Mediante comandos	Es muy fácil de instalar y usar.
Costo	Gratuito	Gratuito	Gratuito
Compatibilidad de navegadores	Chromium, WebKit y Firefox	Chrome, Microsoft Edge (con Chromium), Safari (con WebKit) y Firefox.	Chrome, Firefox, Edge y Safari.
Open source	SI	SI	SI
Sistema operativo	MacOS, Windows, Safari.	Windows, macOS y Linux	Es compatible con todos los sistemas operativos.
Soporte	Soporte de comunidad de 24/7 gratuito	Hay menos comunidad ya que es nuevo en el mercado.	Cuenta con munidad y hay bastante información
Para pruebas basado	UI y End to End	UI y End to End	UI y End to End

Análisis de las tecnologías para el sistema

Como parte del análisis de benchmarking que se presenta en la Tabla 8 se expone los criterios a evaluar que se debe considerar para el sistema de automatización de pruebas y los principales requisitos que debe poseer las tecnologías para la implementación.

Tabla 8
Criterios de evaluación que se considero

N°	Criterio	Descripción
1	El sistema debe ser fácil de uso	El sistema debe ser fácil de uso para el equipo de pruebas.
2	Facilidad de integración con frameworks y herramientas de automatización	El sistema debe ser compatible con otra librería y frameworks para automatizar pruebas funcionales y E2E.
3	Compatibilidad con todos los navegadores	El sistema debe ser compatible con varios navegadores de internet como Google, Edge, etc.,
4	Fácil de mantenimiento	Modificación y corrección de errores deben ser rápidas, para mantener disponibilidad del sistema.
5	Velocidad de ejecución	El tiempo de ejecución de las pruebas es prioridad para cubrir mayor cantidad pruebas.
6	Licencia de código abierto herramientas	El banco tiene como política trabajar con software libre.
7	Soporte	Las herramientas y framework seleccionadas cuenten con comunidad de soporte.

Se procede a evaluar 3 tecnologías de lenguaje de programación más populares y orientado a objeto que los QA automatizadores de prueba utilizan en la actualidad. También sea considerado el patrón de diseño modelo vista controlador (MVC) para el desarrollo de software. Ya que la automatización de pruebas implica desarrollar un sistema que ayude a la ejecución de pruebas de regresión. Asimismo, para poder evaluar la tecnología adecuada se va a tener en cuenta que cumpla con los criterios descritos en la Tabla 8.

Tabla 9

Comparación de características de lenguaje de programación

Clasificación	Puntaje	Descripción
Muy alto	4	LP que cumple con las características requeridas para el desarrollo del sistema de automatización definidos en la tabla 7.
Alto	3	LP que cumple con la mayoría de las características definidos en la tabla7, para el desarrollo del sistema de automatización.
Medio	2	LP que cumple con algunas características definidos en la tabla7, para el desarrollo del sistema de automatización.
Bajo	1	LP que apenas cumple con dos o ningunas características definidas en la tabla7, para el desarrollo del sistema de automatización.

Tabla 10

Comparación de lenguajes de programación para automatización Pruebas de Software

Criterio	TypeScript	Puntaje	Java	Puntaje	Python	Puntaje
Detección de error- antes	Detecta un error de sintaxis de código antes de ejecutar el programa.	4	Los errores de sintaxis se detectan en la fase de compilación del programa.	1	Los errores de sintaxis son detectados por el intérprete de Python antes de ejecutar el programa.	1
Integración con otros Framework y/o herramientas automatización	Selenium, Puppeteer, Playwright, Testcafe, Cypress, NodeJS, WebdriverIO, etc.	3	Playwright, Cypress, WebdriverIO, Selenium, Cucumber, Junit, Serenity	2	Playwright, Selenium	1
Compilación	El código fuente compila a JavaScript	3	El código fuente se compila en una Máquina virtual Java.	2	El código fuente se compila python -m compileall archivo.py	2

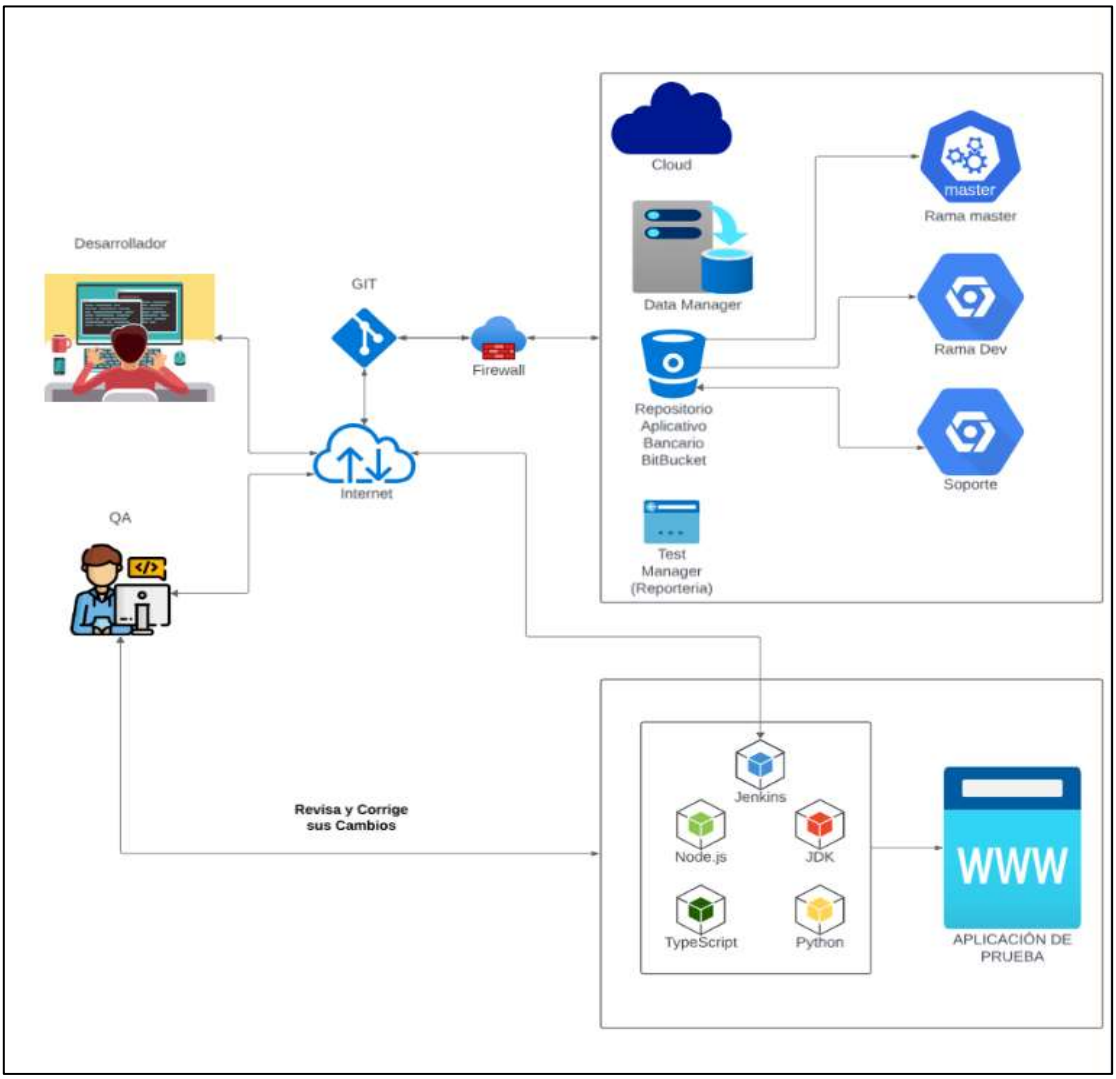
Compatibilidad con Navegadores	Es compatible con todos los navegadores.	3	Mozilla Firefox, Internet Explorer, Safari (Mac OS X),	3	Mozilla Firefox, Chrome, Apple Safari y Edge.	3
Comunidad soporte	Hay actualizaciones, existente documentos y comunidad	3	Hay bastante documentación, gran comunidad	3	Hay bastante documentación, gran comunidad	3
Mantenimiento	Es fácil debido incluso permite identificar los bugs en tiempo real de ejecución. Es un lenguaje tipificado	3	Es un lenguaje de programación simple y legible de comprender.	2	Es claro, fácil de leer y está bien estructurado.	2
Resultado		**19**		**13**		**12**

De acuerdo con el análisis realizada entre las tecnologías la que obtuvo el mayor puntaje con 19 puntos es TypeScript, debido a que es un lenguaje de programación de alto nivel. Supersety posee un compilador super veloz y es de código abierto. Es una tecnología que se puede integrar con facilidad con librerías y frameworks para autorizar pruebas E2E, front-end y back-end y con otros leguajes de programación. Además, permite identificar los errores en la sintaxis del código antes de ejecutar el programa. Por todo lo, mencionado y otras características se seleccionó TypeScript como la tecnología más idónea para el sistema implementado.

Arquitectura física

Para el objetivo 2 se elaboró las arquitecturas del sistema, en la figura 8 se tiene la arquitectura física.

Figura 9
Arquitectura física



El analista de calidad o QA valida desde su máquina de trabajo que la aplicación de prueba este operativo, para ello debe contar con las herramientas de Node, JDK y Python, instaladas y configuradas en su equipo de trabajo. El QA ejecuta los scripts de prueba, que están elaborados en lenguaje Typescript, el sistema compila de manera local los scripts en conjunto con las herramientas y ejecuta de manera automática los casos de prueba en un navegador de internet. El sistema genera un reporte de pruebas donde se indica que casos de prueba pasaron con éxito y que casos fallaron en la prueba. Además, el QA solo tiene acceso a la rama Dev.

El desarrollador desde su máquina de trabajo local se conecta al repositorio en la nube para subir la implementación de una nueva funcionalidad al repositorio de la rama Máster y Dev, este mismo proceso realiza para subir los cambios que realizados en el código del sistema.

Asimismo, es importante mencionar que para subir las versiones de código se utiliza Git y Bitbuckt como repositorio en la nube ramificados categorías como Máster, Dev y Soporte.

Data Manager es donde se almacena la data de prueba en la nube a ello se accede mediante una aplicación.

Los componentes están configurados en espacio de trabajo local, las tecnologías y librerías, así como TypeScript, Node JS, JDK y Jenkins.

El QA tiene que levantar el ambiente de prueba en calidad y validar que se pueda navegar en la aplicación, sobre todo en las funcionalidades que se va a probar mediante el sistema automatizado.

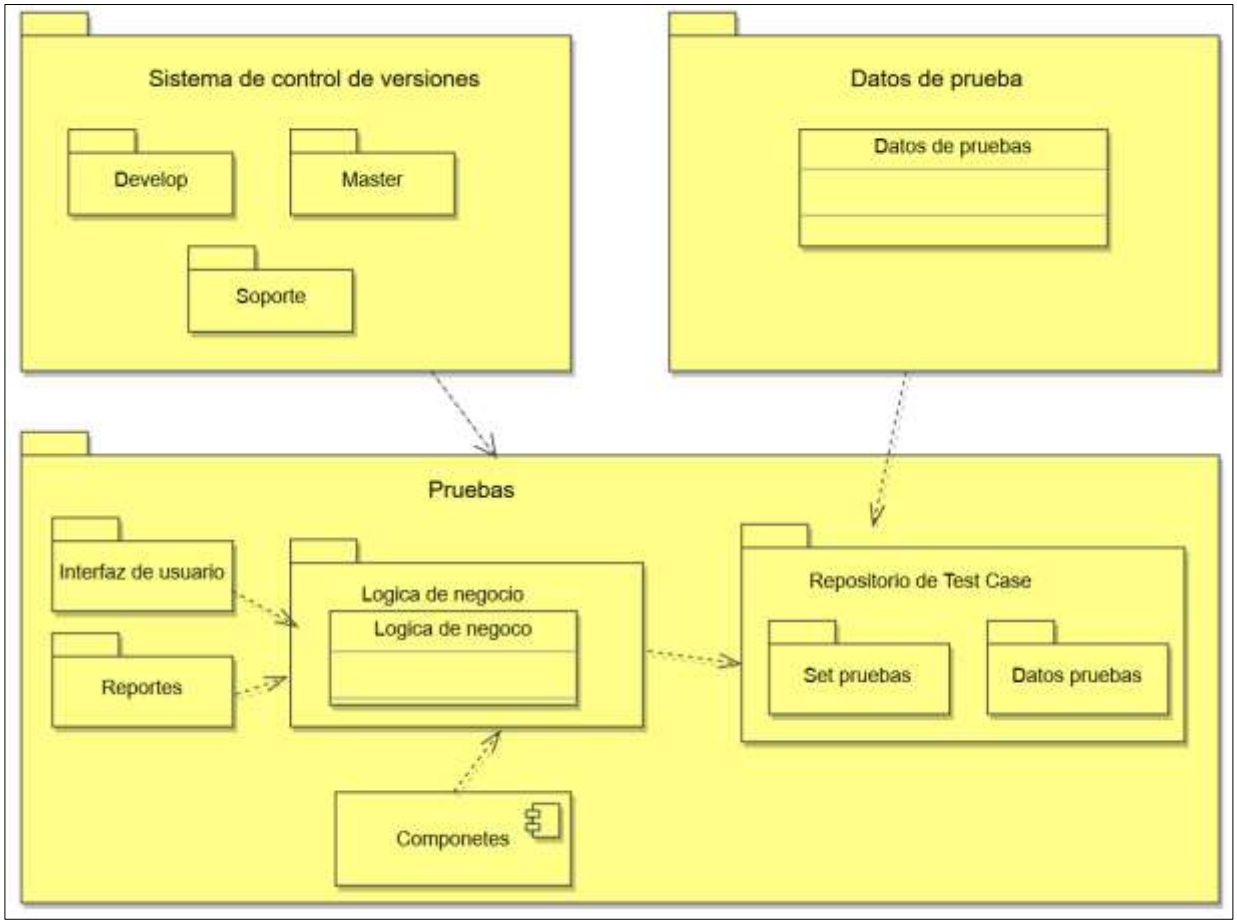
En la rama soporte es repositorio subcategorizado en feature que refleja los cambios en test, reales refleja los cambios en calidad y hotfit es usado para parchear versiones productivas.

Arquitectura lógica

La arquitectura lógica se encuentra compuesta por 3 capas: Pruebas, datos de prueba y sistema de control de versiones. En la Figura 10 se puede visualizar la interrelación de las capas del sistema.

Figura 10

Arquitectura Lógica



A continuación, se procede a detallar los servicios y sus respectivos componentes.

Capa pruebas

En esta capa, la interfaz del usuario y reportes interactúa con la lógica del negocio para poder seguir un flujo definido de acuerdo a la funcionalidad a validar. A su vez, la lógica de negocio se comunica con los componentes donde están alojados las herramientas de automatización como java, node, Python y Typescript. También se comunica con el repositorio de Test case para poder obtener los casos de pruebas automatizados y los datos de prueba almacenados en cache de manera temporal.

Capa datos de prueba

Esta capa se relaciona con la capa pruebas para poder brindar diferentes datos de prueba como DNI de clientes, cuentas bancarias e interbancarias, líneas de crédito, tarjetas de crédito, etc.

Capa de sistema de control de versiones

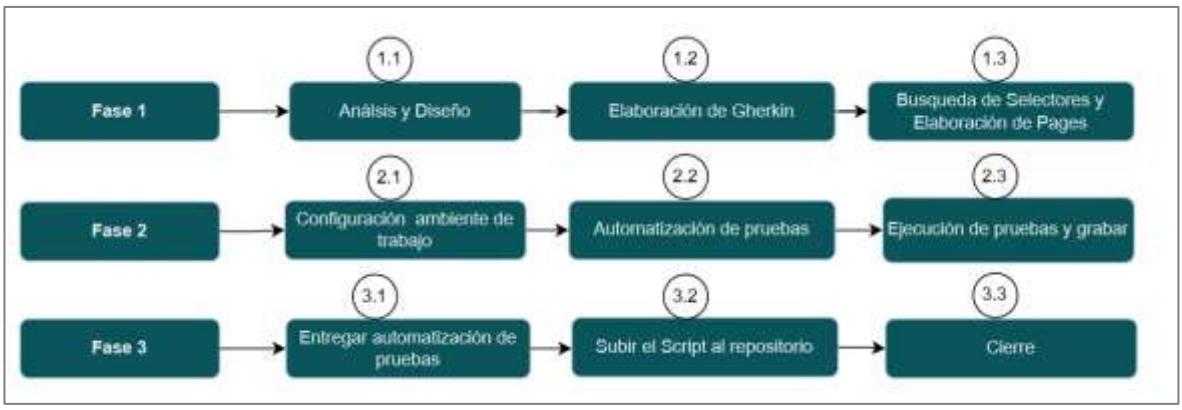
Esta capa se comunica con la capa de pruebas para que se pueda descargar o cargar cambios realizados en los scripts de negocio y se lleve un correcto control de los cambios realizados.

Desarrollo de la Solución

En esta sección describe el desarrollo de la solución implementado mediante las diferentes fases, análisis, configuración del ambiente, redacción de casos en Gherkin, Feture y la estructura del proyecto.

Figura 11

Fases de Desarrollo del sistema



Análisis y Diseño de casos de pruebas

Mediante el análisis se determinó las principales funcionalidades aptas a automatizar y que no requieran un token digital.

El alcance de la automatización es para 9 funcionalidades o características:

- Loguin
- Consultar detalle del producto
- Movimiento del producto
- Información de productos
- Pagar tarjeta de crédito
- Transferencias
- Línea de Crédito

- Plin
- Cambio de venta y compra de moneda

En la tabla11 se indica los escenarios y los respectivos números de casos de pruebas identificados.

Tabla 11

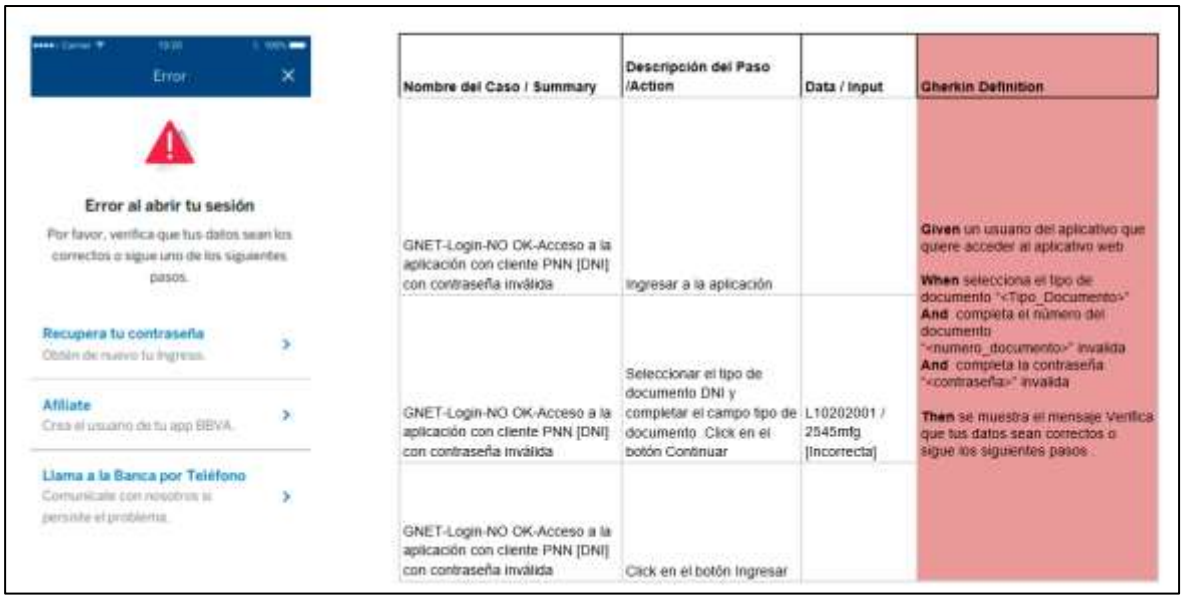
Escenarios y casos de prueba

Funcionalidades	Número de casos de pruebas
Login	34
Consultar el detalle del producto	20
Movimiento del producto	25
Información de productos	26
Pagar tarjeta propia	25
Realizar trasferencias propias	82
Solicitar línea de crédito	30
Transacciones con billetera digital	24
Realizar cambio de venta y compra de moneda	30

Previó a la implementación estos estos escenarios de casos pruebas se deben escribir en el lenguaje Gherkin para un mejor entendimiento y siguiendo las buenas prácticas.

Figura 12

Tipos de escenarios definido en el lenguaje Gherkin



Instalación y configuración de herramientas

Para el desarrollo del proyecto se procedió a instalar y configurar las siguientes herramientas que se mencionan a continuación.

- TypeScript
- Python
- Java – JDK 8
- Node JS
- GIT

En el anexo 2 se puede encontrar de manera detallada los pasos a seguir para la correcta instalación cada una de las tecnologías empleadas.

Configuración del sistema de pruebas

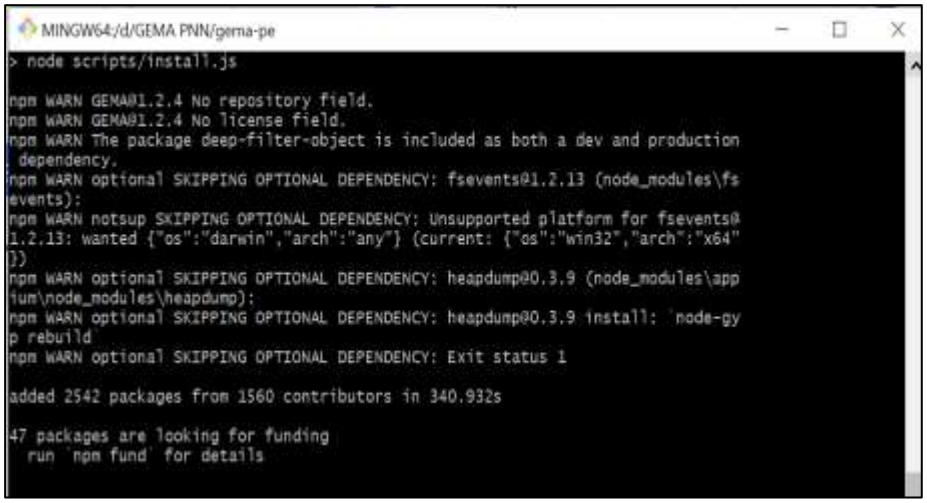
Luego de haber realizado las instalaciones de las herramientas se procede con la configuración del sistema para poder levantar el ambiente de pruebas en calidad de manera local. En el anexo 3 se puede encontrar de manera detallada los pasos a seguir para la correcta configuración.

Despliegue del ambiente de pruebas

Abrir la consola de GitBash y ubicarse en la carpeta donde se descargó el proyecto. Crear una nueva rama y ejecutar el siguiente comando "*git checkout feature/nombre_de_la_rama*" para ubicarse en la rama creada. Luego ejecutar el siguiente comando "*npm install*" para la descarga de las dependencias del proyecto

Figura 13

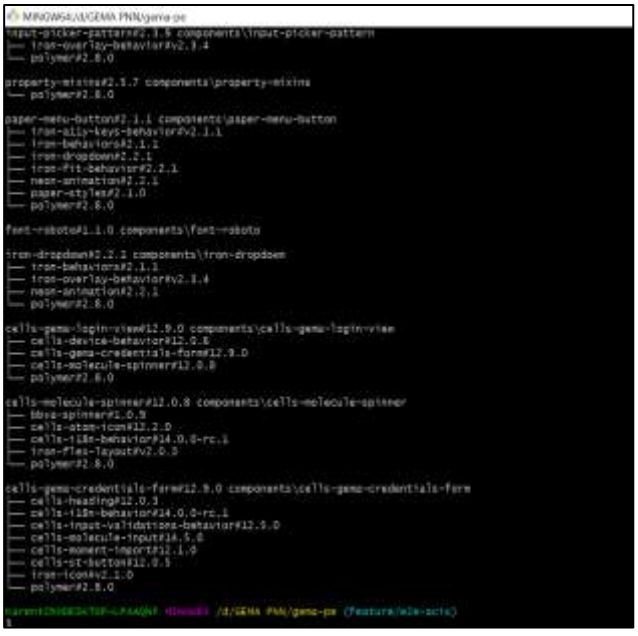
Instalación de dependencias



Luego ejecutar el siguiente comando "*bower i -F*" para la instalación de los componentes web.

Figura 14

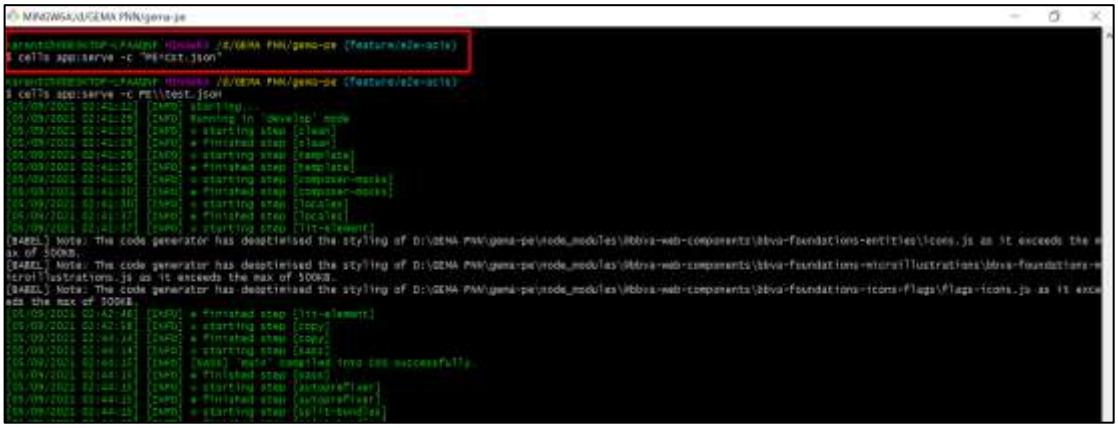
Instalación de componentes Web



Una vez finalizado la instalación, ejecutar el siguiente comando "*cells app:serve -c PE\\test.json*" para levantar el ambiente de pruebas de calidad de manera local.

Figura 15

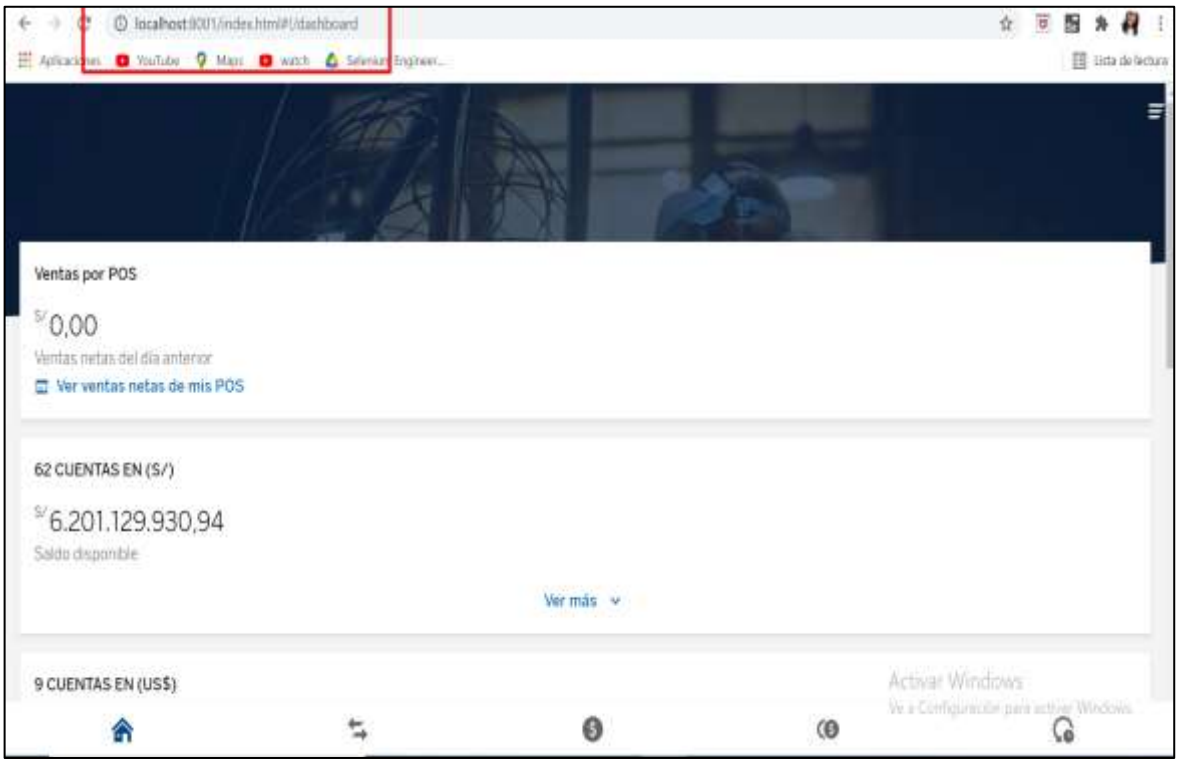
Validación de ambiente de prueba – Parte 01



Una vez que empiece a cargar el sistema y finalice se genera un URL local. Se debe ingresar al localhost con un usuario de prueba y validar que se pueda navegar por el aplicativo de prueba como se observa en la Figura 16

Figura 16

Validación de ambiente de prueba – Parte 02



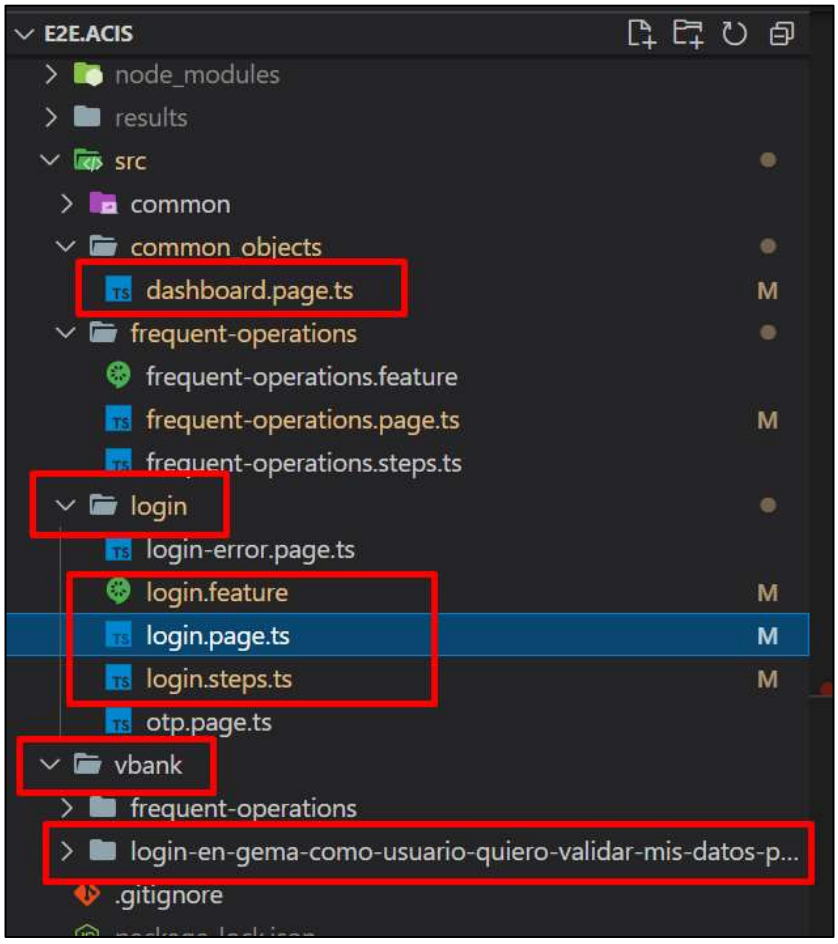
Estructura del proyecto

La estructura del proyecto se elaboró de la siguiente manera:

- Common objects: Carpeta donde se ubican los elementos en común del aplicativo como la consulta de cuentas, tarjetas o préstamos.
- Login: Carpeta donde se ubica la funcionalidad de acceso al aplicativo.
- Funcionalidad por automatizar: Carpeta donde se ubica la funcionalidad que va a ser automatizada. Esto debe contar con 3 archivos funcionalidad.feature, funcionalidad.ts y funcionalidad.steps.
- Vbank: Es esta carpeta se guarda las respuestas de los servicios en cada escenario creado.

Figura 17

Estructura del Proyecto



Para poder realizar la automatización de una funcionalidad se debe crear un archivo feature, step y page, donde se detalla a continuación:

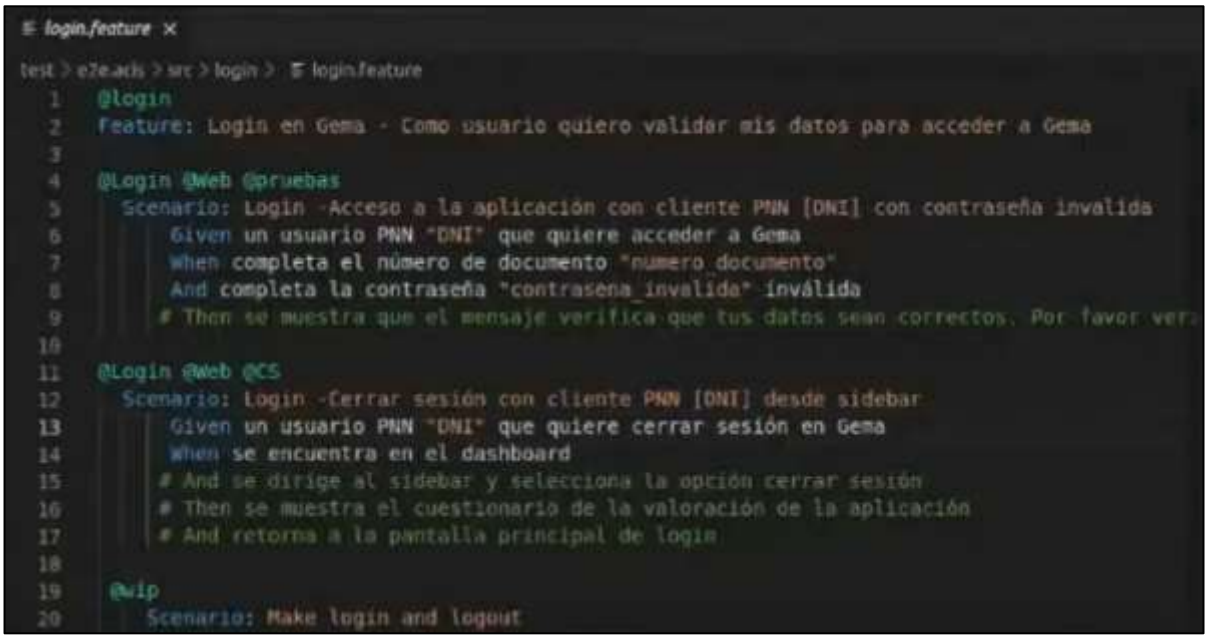
Archivo feature

Este archivo se encuentra la redacción los casos de prueba en cucumber en el lenguaje Gherkins. Para ello sea utilizado las palabras reservadas:

- Feature: Titulo con una descripción de alto nivel de la funcionalidad que se va a probar.
- Scenario: Lista de pasos de la prueba que se quiere especificar.
- Given: Contexto y precondiciones del caso del caso de prueba.
- When: Acciones que se van a ejecutar o interacciones del usuario con el sistema.
- Then.: Resultado esperado y validaciones a realizar.

Figura 18

Archivo login Feature



Asimismo, en este archivo se ingresan los diferentes escenarios que una funcionalidad pueda tener, por ejemplo:

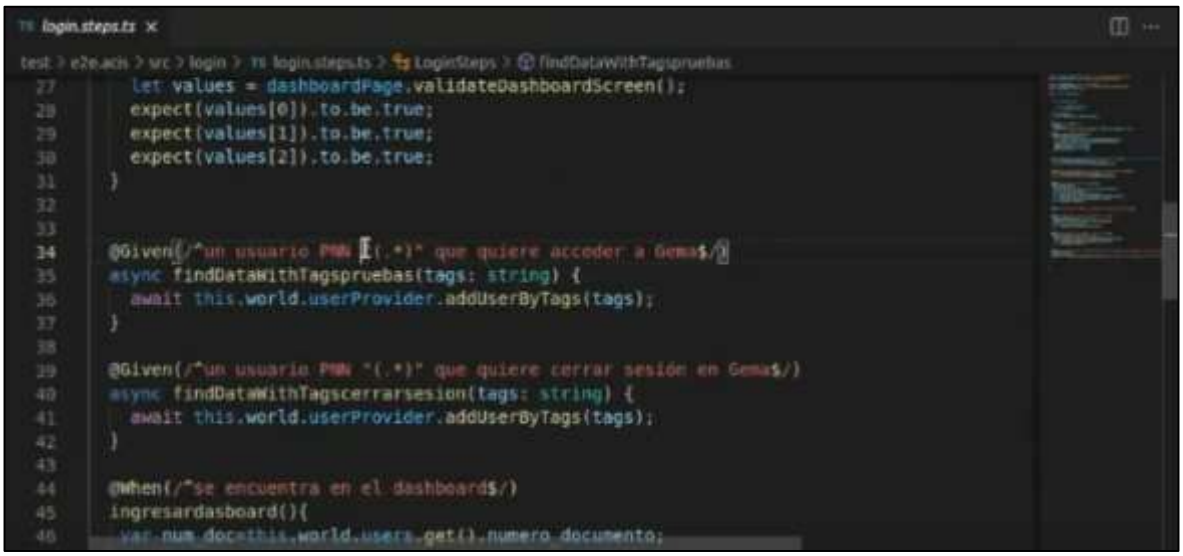
- Ingresar al aplicativo con una contraseña invalida
- Cerrar sesión de un usuario que se encuentra en sesión en el aplicativo

Cada escenario de prueba se puede agregar tags la cual inician con arroba. Esto ayuda para que se puede ejecutar un lote de casos o algún caso en específico.

Archivo step

En este archivo que está desarrollado en lenguaje TypeScript se implementan los pasos de una funcionalidad descritas en el archivo feature. Asimismo, se realiza la importación de librerías y del page para poder acceder a los elementos web.

Figura *19*
Archivo feature



Archivo pages

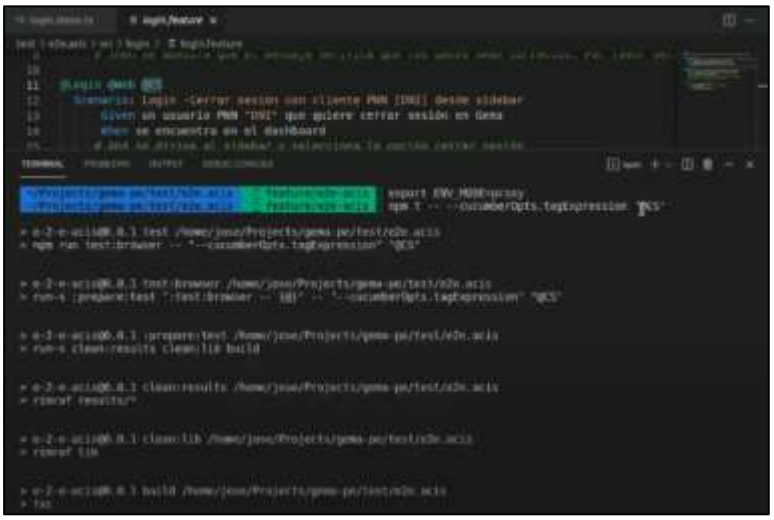
En este archivo que se encuentra desarrollado en lenguaje TypeScrpit se encuentra la ruta de la página a probar y los selectores que son el elemento base de la página de cual se van a derivar los demás elementos. Asimismo, en este archivo se describen funciones principales y comunes como addValue() o click().

Figura 20

Archivo page

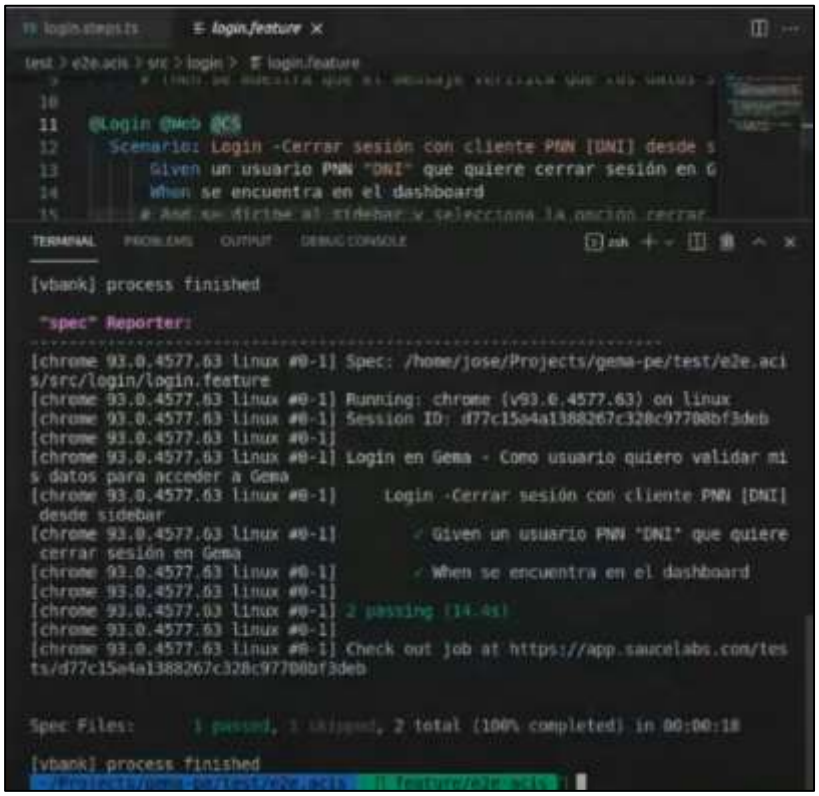
Una vez concluido con el desarrollo de los casos, se procede a validar si lo desarrollado funciona de manera correcta, por ello se ejecuta el caso específico mediante su tag definido. Primero se debe poner el VBANK en modo proxy para que ejecute el caso sin grabar, por lo cual se ejecuta el comando "*export ENV_MODE=proxy*". Luego se procede con la ejecución mediante el siguiente comando "*npm t -- --cucumberOpts.tagExpression '@CS'*"

Figura 21
Ejecución de caso de prueba en modo proxy



Se verifica en la siguiente figura que la prueba culmino de manera satisfactoria

Figura 22
Ejecucion exitosa de caso de prueba

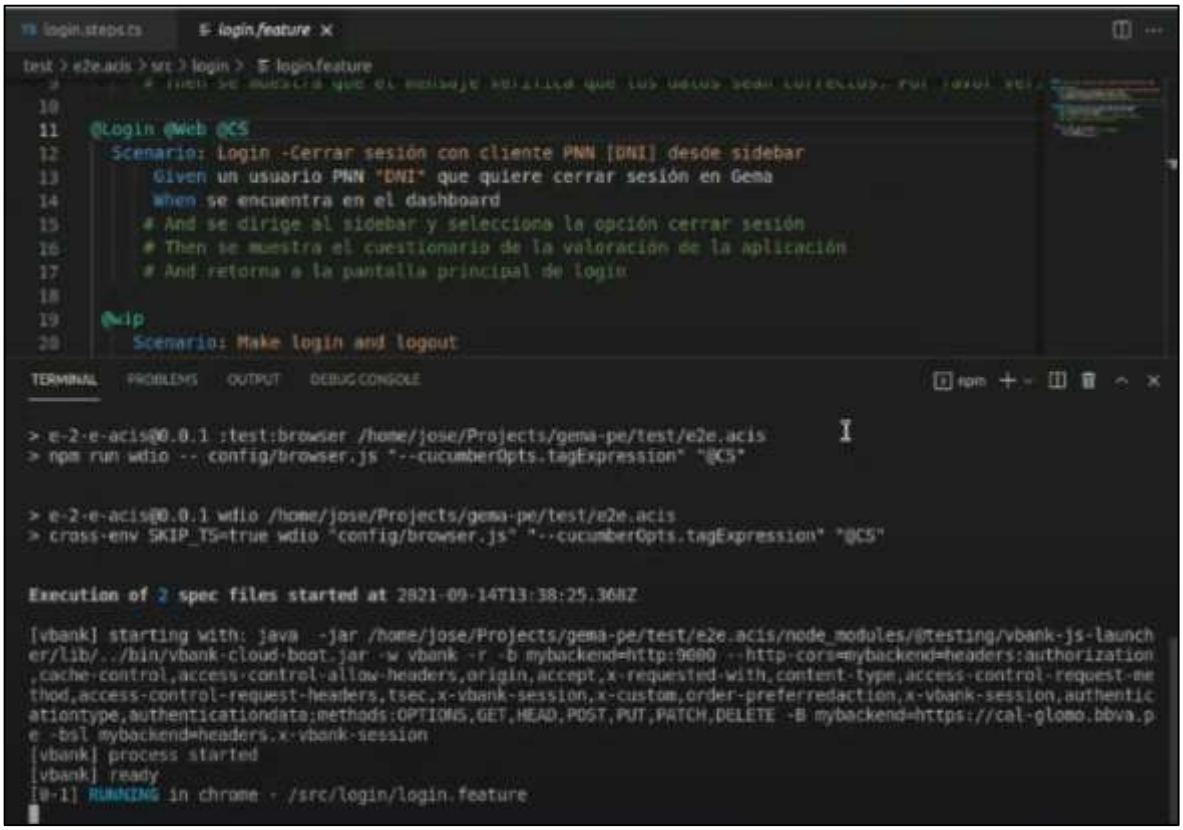


Ejecución de casos de pruebas automatizados

Para proceder con la ejecución de casos de prueba, primero se debe poner el VBANK en modo grabación, por lo cual se debe ejecutar el comando "*export ENV_MODE=rec*". Esto permite que el caso o los casos ejecutados se graben y puedan ser visualizados más adelante. Luego se procede con la ejecución del caso de prueba automatizado mediante el siguiente comando "*npm t -- --cucumberOpts.tagExpression '@CS'*". Como se ve en la siguiente figura procede con la ejecución del caso de prueba automatizado.

Figura 23

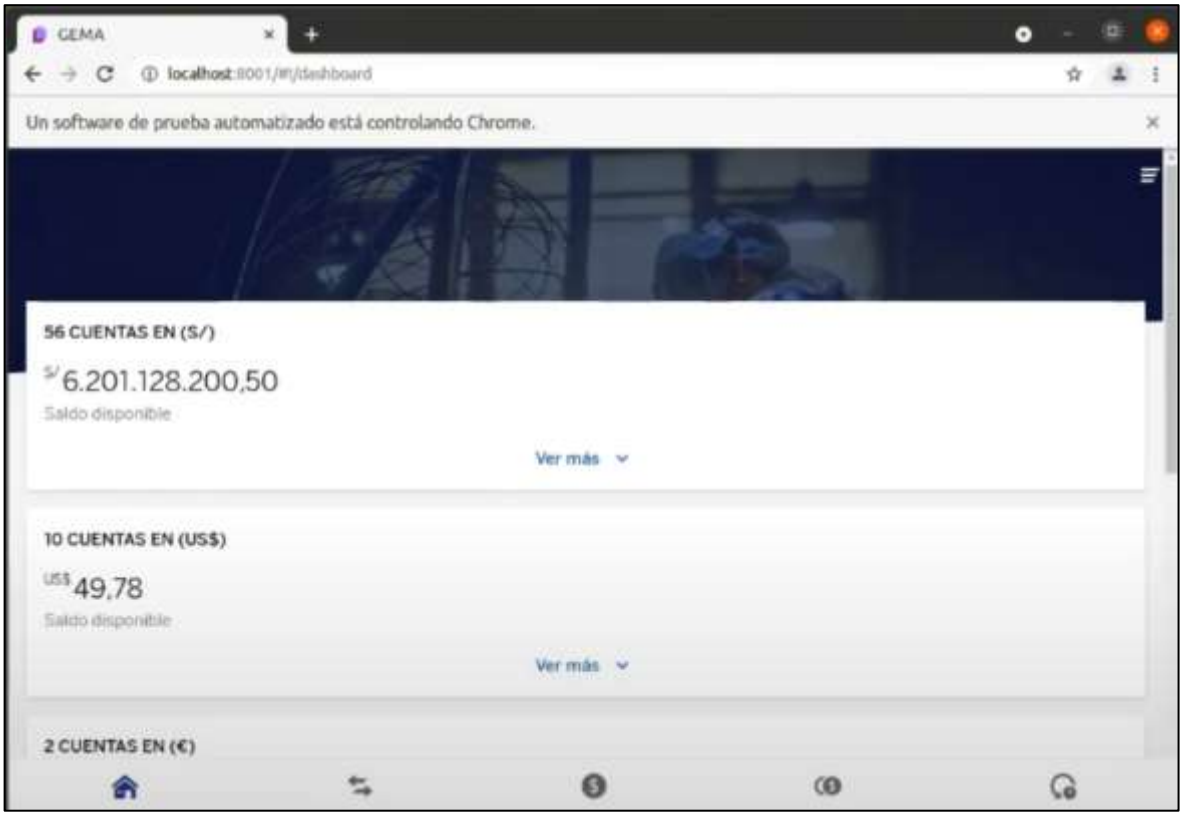
Ejecución de caso de prueba en modo rec



El sistema abre de manera automática el navegador Chrome y procede con la ejecución de pasos, como se observa en la Figura 24.

Figura 24

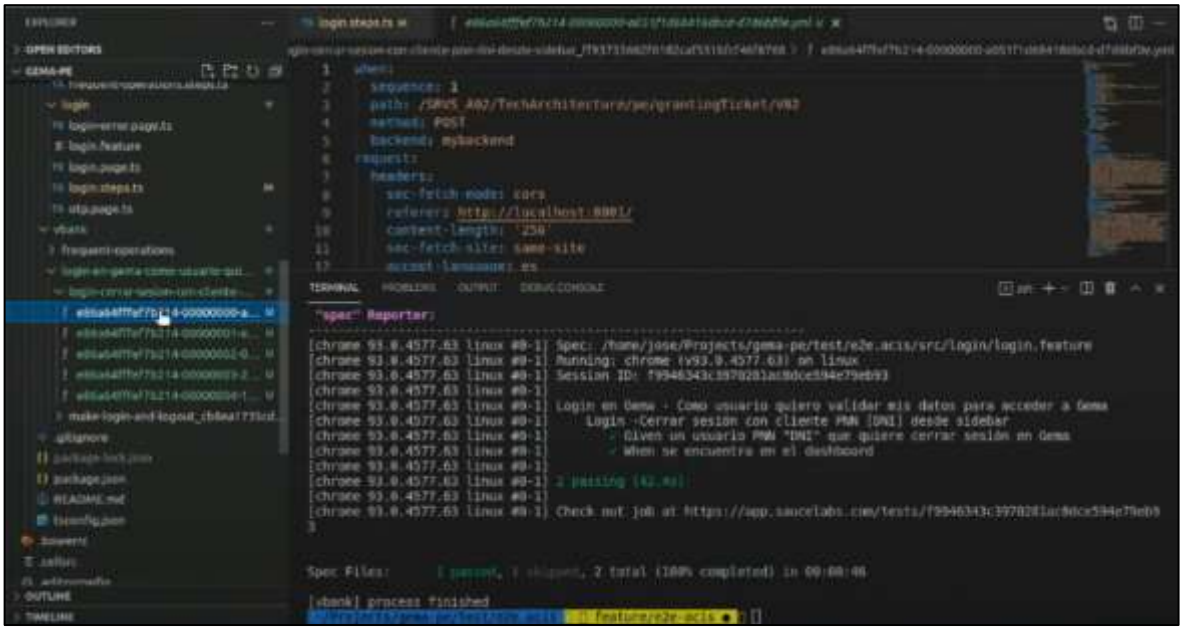
Ejecución de caso de prueba automatizado



Se verifica que el caso prueba termino de manera satisfactoria ya que se encuentra en estado “passed” y con un tiempo de ejecución de 46 segundos. Asimismo, la grabación del caso también culmino de manera correcta ya que se ve en la carpeta vbank de color verde.

Figura 25

Ejecución de caso de prueba en modo rec

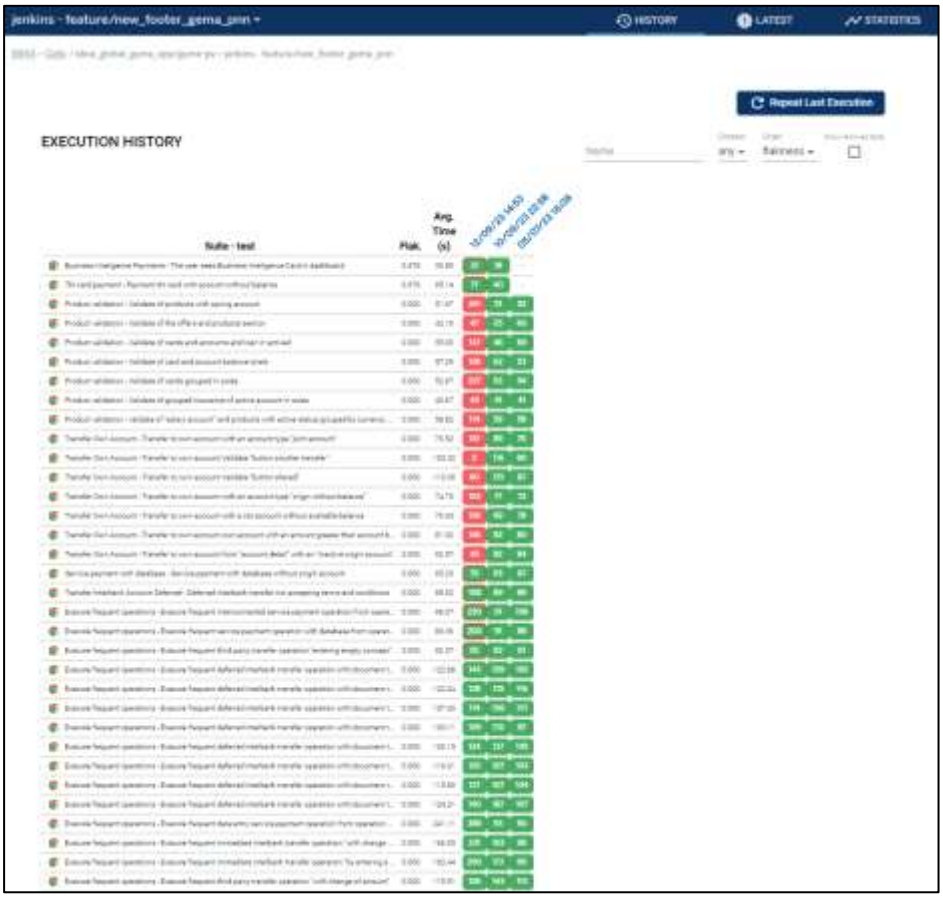


Reporte de casos ejecutados

Para poder visualizar los casos ejecutados se debe ingresar a Test Manager. Este aplicativo web que se encuentra alojado en la nube permite visualizar los casos que se han ejecutado. Como se ve en la siguiente figura cada fila es un caso ejecutado, también se visualiza la fecha y hora que ha sido ejecutado el caso de prueba. Si el caso ha salido satisfactorio se verá en color verde, caso contrario se verá en color rojo.

Figura 26

Visualización de casos ejecutados por fecha



Para poder ver los casos ejecutados en conjunto se debe dar clic en la fecha y hora que se desea consultar. En la siguiente figura se muestra un resumen más detallado de todos los casos ejecutados esa fecha, también se muestra la cantidad de casos que pasaron de manera satisfactoria en color verde y los que fallaron en color rojo.

- Se planifico un total de **133** casos de pruebas para correr en el sistema automatizado.

El resultado obtenido al finalizar la ejecución

- 118 casos de pruebas se ejecutaron con éxito
- 14 casos de pruebas terminaron fallidos
- Pendiente de ejecución
- El tiempo ejecución es 4h y 15 minutos

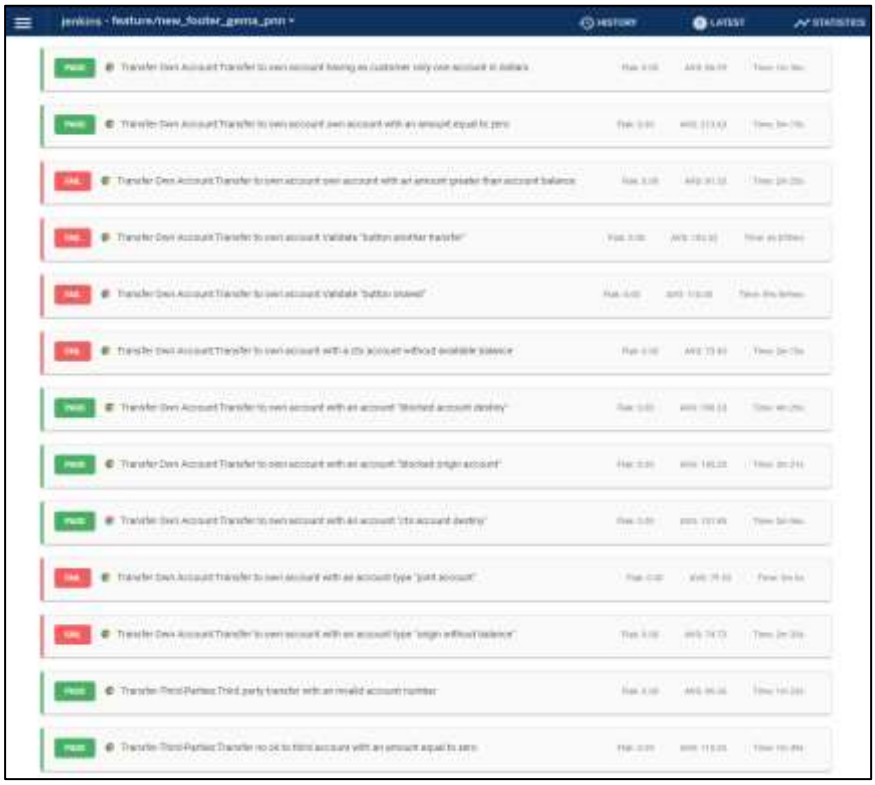
Figura *27*

Visualización de casos ejecutados por fecha

Para poder visualizar el caso correcto ejecutado se debe ir a la pestaña “Tests” y dar clic sobre el caso satisfactorio.

Figura *28*

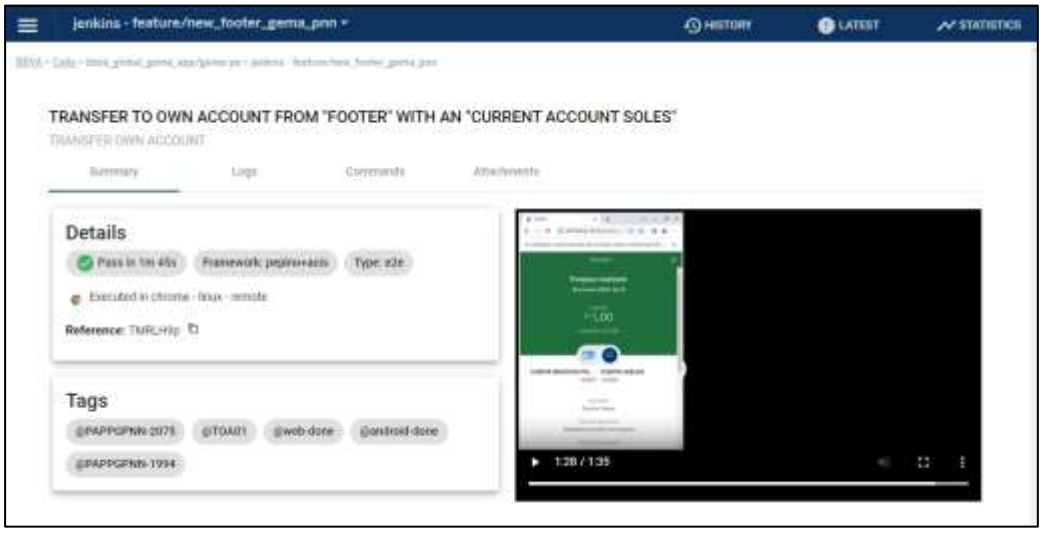
Visualización de casos ejecutados por estado



El caso ejecutado se puede revisar en un video donde se visualiza todos los pasos que ha hecho el sistema de pruebas automatizado y el tiempo que demoro en ejecutar el caso.

Figura *29*

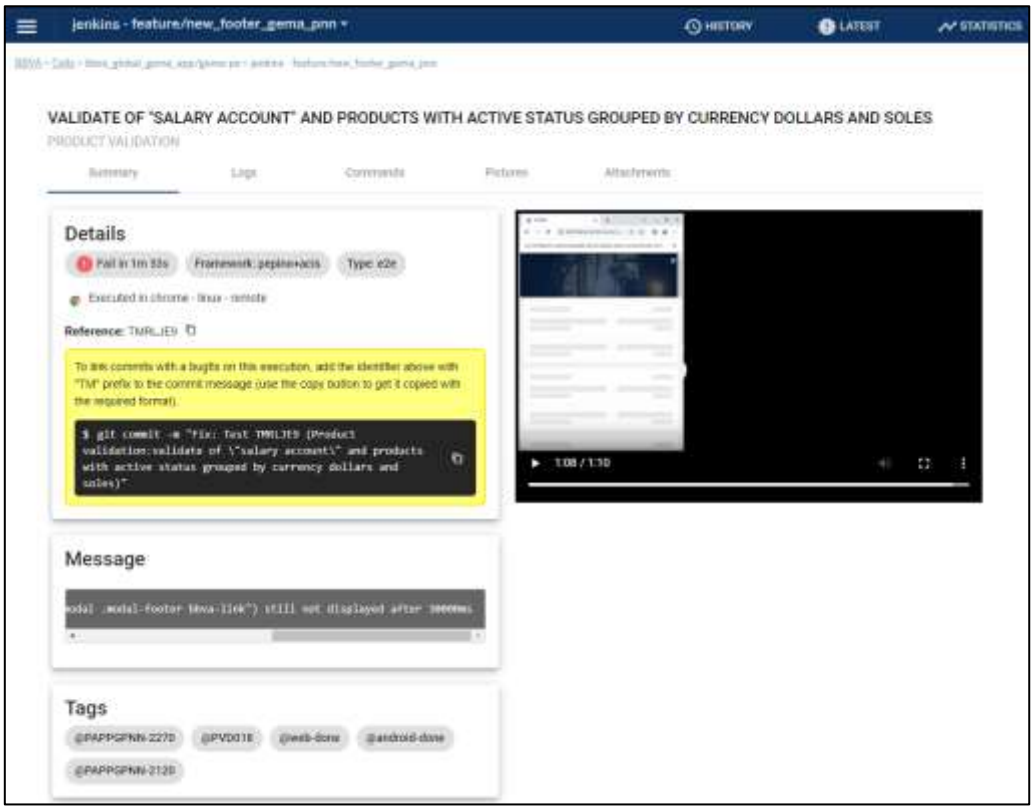
Visualización de un caso ejecutado satisfactorio



De igual modo en caso erróneo se puede visualizar en que paso fallo, el tiempo en que fallo el caso y se muestra un mensaje donde indica la posible falla del caso.

Figura 30

Visualización de un caso ejecutado fallido



Validación del Proyecto

En este capítulo se hará presente la validación del proyecto desarrollado. Así como las técnicas utilizadas para el sustento de este proyecto.

Casos manuales VS casos automatizados

Para probar los casos de pruebas de regresión de manera manual se contó con 3 QA para todos los escenarios identificado al momento de implementación. En el primer ciclo de pruebas se demoró 10 días en ejecutar los 296 casos existentes, como se muestra a continuación.

Tabla 12

1er ciclo de pruebas de regresión y personal asignados

	Personal de prueba			
Fecha de ejecución	**QA01**	**QA02**	**QA03**	**Nro. casos ejecutados**
05/09/2022	10	10	11	31
06/09/2022	10	10	10	30
07/09/2022	10	10	11	31
08/09/2022	10	11	10	31
09/09/2022	10	11	10	31
12/09/2022	10	10	10	30
13/09/2022	10	11	10	31
14/09/2022	11	11	10	32
15/09/2022	12	10	11	33
16/09/2022	5	4	7	16
			Total	296

El resumen de resultados en el primer ciclo de pruebas fue el siguiente.

Tabla 13

Resumen de 1er ciclo de pruebas de regresión

Resumen Resultados	**Nro. Casos**	**%**
Satisfactorio	215	84.80%
Insatisfactorio	81	15.20%
Pendiente Ejecución	0	0.00%
TOTAL	**296**	**100.00%**

En el segundo ciclo de pruebas de regresión se revalidaron los casos 81 casos erróneos de igual manera contando como recursos 3 QA, lo cual tomo 3 días para poder terminar con las revalidaciones de manera manual.

Tabla 14

2do ciclo de pruebas de regresión y personal asignados

	Personal de prueba			
Fecha de ejecución	**QA01**	**QA02**	**QA03**	**Nro. casos ejecutados**
19/09/2022	10	10	11	31
20/09/2022	9	10	11	30
21/09/2022	6	8	6	20
			Total	81

En total se empleó 13 días para poder culminar las pruebas de regresión en su totalidad. Con la implementación del sistema de pruebas automatizados se ejecutó la misma cantidad de casos de prueba. Se visualiza que el primer día se pudo realizar la ejecución de 133 casos de prueba, el segundo día se pudo ejecutar 123 casos y el tercer día se culminó los 40 casos restantes al primer ciclo de pruebas.

Figura 31

Resumen de 1r día de pruebas de regresión automatizado – 1er ciclo

Figura 32

Resumen de 2do día de pruebas de regresión automatizado – 1er ciclo

Figura 33

Resumen de 3er día de pruebas de regresión automatizado – 1er ciclo

Tabla *15*
1er ciclo de pruebas de regresión automatizados

Fecha de ejecución	Nro. casos ejecutados
06/03/2023	133
07/03/2023	123
08/03/2023	40
Total	296

En el primer ciclo de pruebas con el sistema automatizado se obtuvo el siguiente resumen.

Tabla *16*
Resumen de 1er ciclo de pruebas de regresión automatizados

Resumen Resultados	**Nro. Casos**	**%**
Satisfactorio	239	80.74%
Insatisfactorio	57	19.26%
Pendiente Ejecución	0	0.00%
TOTAL	**296**	**100.00%**

En el segundo ciclo de pruebas automatizados se validaron los 57 casos erróneos, lo cual tomo 1 día para terminar con las revalidaciones.

Figura 34

Resumen 2do ciclo de prueba de regresión automatizado

Tabla *17*
2do ciclo de pruebas de regresión automatizados

Fecha de ejecución	Nro. casos ejecutados
13/03/2023	57

En total para culminar el ciclo de pruebas automatizados se empleó 4 días.

Encuesta de satisfacción

Para medir el nivel de satisfacción de los usuarios que usan el sistema automatizado se elaboró un cuestionario con Google Forms en base a la escala de Likert. El cuestionario consta de 10 preguntas y se realizó a 5 personas del área de calidad. A continuación, se muestran las preguntas elaboradas en Google Forms:

Figura 35
Encuesta de satisfacción – Parte 01

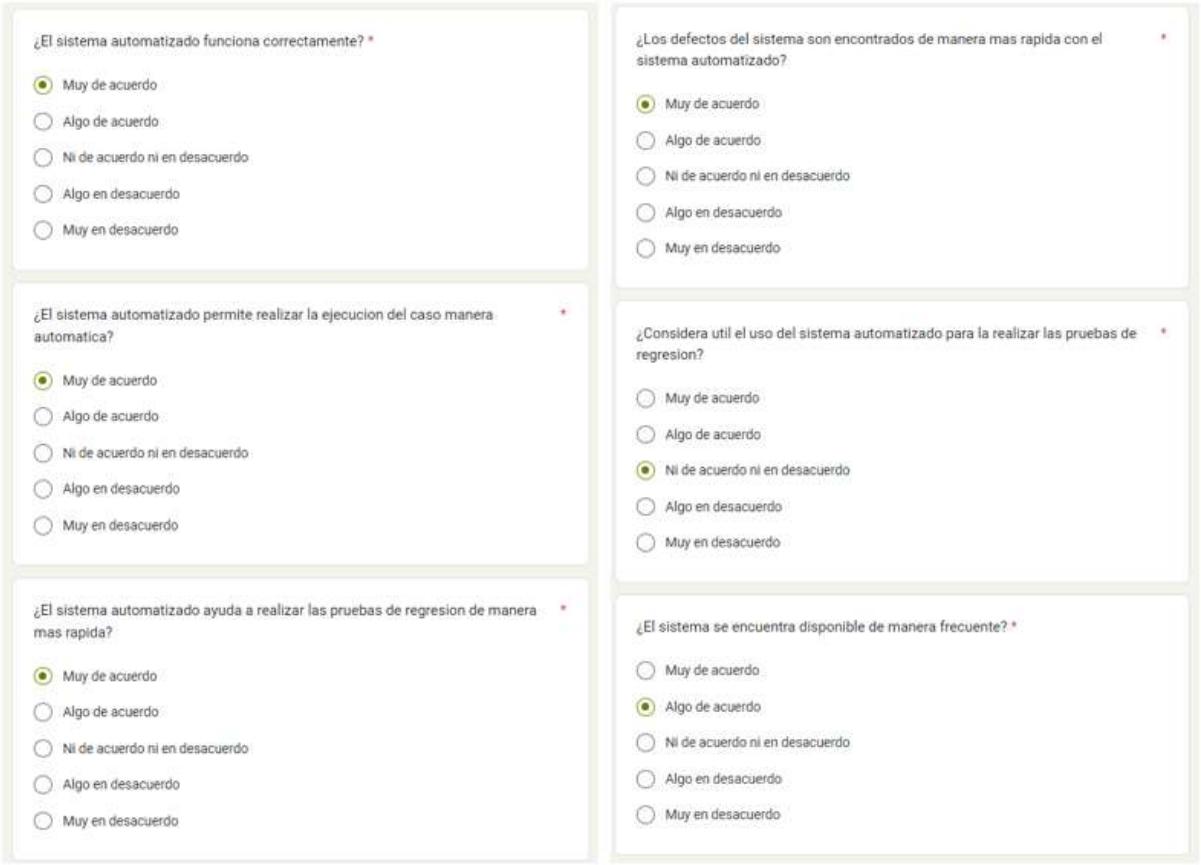
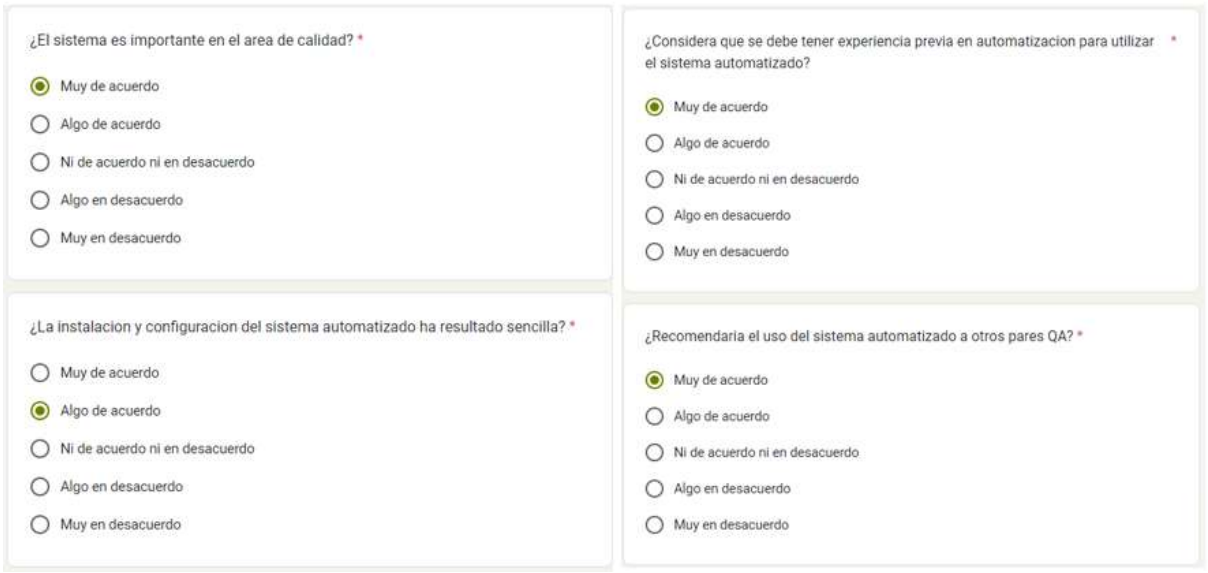


Figura 36

Encuesta de satisfacción – Parte 02



Para la medición de las respuestas se utilizó los siguientes niveles de puntaje.

Tabla *18*

Niveles de medición y su puntaje

5	4	3	2	1
Muy de acuerdo	Algo de acuerdo	Ni de acuerdo ni en desacuerdo	Algo en desacuerdo	Muy en desacuerdo

Luego de haber recopilado la información de los 5 encuestado se procede a mostrar los resultados en la siguiente tabla.

Tabla *19*

Resumen de encuesta de satisfacción

Encuestado	Preguntas									
	1	2	3	4	5	6	7	8	9	10
Persona 1	5	5	5	5	5	5	5	4	5	5
Persona 2	5	5	5	5	5	5	5	4	5	5
Persona 3	5	5	5	4	4	5	5	3	4	5

Persona 4	4	4	5	4	5	4	4	3	4	4
Persona 5	5	5	5	5	5	4	5	4	3	5

Con los datos recopilados se calcula el alfa de Cronbach, donde K es el número de pregunta, Vi es la varianza de cada pregunta y Vt es la varianza total como se muestra en la siguiente figura.

Figura 37

Alfa de Cronbach

$$\alpha = \frac{k}{k-1}\left[1 - \frac{\sum V_i}{V_t}\right]$$

α : Alfa de Cronbach
k : Número de ítems
V_i: Varianza de cada ítem
V_t: Varianza del total

Nota. Adaptada De. Calcular Alfa de Cronbach con excel y confiabilidad del instrumento de investigación FACIL, 2020. (https://www.youtube.com/watch?v=wCFpTCSdnWE

Con los datos obtenidos el alfa de Cronbach da como resultado 0.85.

Tabla 20

Resultados total de los encuestados

Variables	Resultados
K	10
Vi	2.08
Vt	8.8
α	0.85

Interpretación de los Resultados

Con la comparativa realizada de la ejecución de los casos manuales VS automatizados se obtuvo los siguientes resultados:

Tabla *21*
Resultados obtenidos

Indicador	Resultado obtenido
Tiempo de ejecución de casos de prueba	Con la ejecución de casos manuales se dedicó 13 días para la culminación total de casos de manera satisfactoria. Mientras que con la ejecución de casos automatizados se destinó 4 días para la culminación total de casos de prueba. Esto representa una mejora en el tiempo donde se redujo a 9 días para la ejecución de los casos.
Número de defectos encontrados	Se pudo realizar una temprana detección de defectos en el sistema para que los desarrolladores puedan revisarlos y solucionarlos en etapas más tempranas.
Número de casos ejecutados	Se puedo realizar la misma cantidad de ejecución de casos en un menor tiempo con el sistema automatizado, con lo que se mejora la eficiencia del equipo

De acuerdo con la encuesta de satisfacción se puede mencionar que el 80% optaron por la opción “Muy de acuerdo” con respecto al funcionamiento del sistema, lo que evidencia que el sistema ha sido construido de manera correcta y cumple con los requerimientos definidos. Por otro lado, el 100% de los encuestados eligieron la opción “Muy de acuerdo” acerca de que el sistema ayuda a realizar las pruebas de regresión de manera más rápida, lo que se puede constatar de que el sistema ayuda a que las pruebas de regresión sean más rápidas de realizar. A su vez el 80% de los encuestados escogieron la opción “Muy de acuerdo” en relación con que el sistema es importante para el área de calidad, lo que se puede hacer ver que el sistema es una necesidad en el área de calidad ya que aporta para la automatización de pruebas y así evitar el retrabajo. Por otra parte, se calculó el alfa de Cronbach dando como resultado 0.85, con este resultado se puede demostrar que los datos de la encuesta realizadas son de una alta consistencia.

Plan de continuidad

Realizar un plan de continuidad para garantizar la continuidad y la operatividad del sistema automatizado, así como para gestión de mantenimiento e implementación y automatización de nuevos escenarios en el área de Calidad.

Corregir y Actualizar los cambios

Para realizar la corrección y actualización de los cambios primero se debe crear la rama en el repositorio de la nube. Luego se debe procede a recuperar este archivo desde el repositorio de Bitbucket. El siguiente paso es validar el ambiente y ejecutar los casos de prueba. después de realizar los cambios, para validar se debe ejecutar las pruebas sí todo está "OK" entonces se procede a grabar los casos. Como último paso, se procede a subir los archivos modificados al repositorio en la nube. En caso de que persista el error entonces se procede a comunicar al desarrollador para la respectiva revisión.

Soporte de Primer Nivel

- En caso de que presente error el aplicativo de prueba se debe registrar Gestión de Incidencia en Jira detallando según las políticas establecidas.
- En caso de que persista el problema en el aplicativo se debe reportar al equipo de desarrollo según corresponda.

  Según el SLA el tiempo de respuesta a esperar es:

- Para funcionalidades críticos 2 horas
- Para funcionalidades Controlado 4 horas
- Para funcionalidades riesgos bajo hasta 2 días

Disponibilidad del aplicativo

La disponibilidad del aplicativo bajo prueba debe estar disponible las 24 horas y 7 días a la semana. Por esto, es importante revisar la matriz de riesgos de la funcionalidad para determinar la criticidad de la funcionalidad. Como se puede observar en la siguiente tabla el factor de riesgo representa cada funcionalidad, los cuales se catalogan de riesgo extremo, riesgo controlado y riesgos bajo.

Tabla 22
Resultados obtenidos

	Evaluación de riesgos					
Funcionalidad	**Impacto**	**Probabilidad**	**Nivel de Control**	**Nivel de Confianza**	**Factor de Riesgo**	**Observación**

Funcionalidad 1	4	5	3	3	**9.3**	Riesgo Extremo
Funcionalidad 2	5	4	3	3	**9.3**	Riesgo Extremo
Funcionalidad 3	4	5	4	1	**9.0**	Riesgo Extremo
Funcionalidad 4	4	3	4	4	**3.6**	Riesgo Controlado
Funcionalidad 5	4	3	4	5	**3.0**	Riesgo Controlado
Funcionalidad 6	4	2	4	3	**2.8**	Riesgo Bajo
Funcionalidad 7	3	3	4	5	**2.3**	Riesgo Bajo
Funcionalidad 8	3	3	4	5	**2.3**	Riesgo Bajo

Implementación de nueva funcionalidad

Para la implementación de una nueva funcionalidad se deben seguir los siguientes alineamientos conforme designado en el área de calidad en la organización. Para la implementación se debe seguir las siguientes fases:

Fase 1:

- ✓ Se solicita la reunión con el Scrum y QA para definir antes que finalice el desarrollo para tener el alcance y revisar los casos pruebas propuestos.
- ✓ Luego se procede a solicitar la reunión con QE y Automation Team (AT) para determinar los casos de pruebas a automatizar.

Fase 2:

- ✓ Automation Team debe revisar la viabilidad técnica de la propuesta de los casos.

- ✓ Luego debe solicitar reunión con la partición de QE, Scrum y QA como resultado de este se deben listar los casos de a automatizar, tiempo estimado y numero de desarrolladores que participaran en la implementación. Una condición de prioridad es importante el tiempo estimado debe ser acorde al pase a producción.

Fase3:

- ✓ El Automation Team comienza con las automatizaciones de los casos una vez finalizado el desarrollo. El equipo de AT debe solicitar la reunión con Scrum y QA para entregar los casos automatizados.
- ✓ El desarrollo y las automatizaciones deberán finalizar a la par al pase a producción.
- ✓ Se debe realizar el traspaso de las automatizaciones al equipo de QA.

Principales roles que intervienen el proceso son lo siguiente:

- QE es la persona encargada de ver aspectos relacionados a desarrollo y los procesos de calidad.
- QA es la persona encargada del aseguramiento de la calidad del producto en desarrollo.
- Scrum conformado por el equipo de desarrollo.

Capacitación

La capacitación se realizará para todos los colaboradores que estén involucrados en los diferentes proyectos del banco para el área de calidad. Esto incluye a PO y Scrum master, el objetivo es que todos tenga conocimiento del sistema automatizado a alto nivel y asimismo que funcionalidades estén automatizados.

Para los QA la capacitación se debe centrar en el flujo del proceso que deben seguir para identificar los casos de pruebas si deben ejecutar con el sistema automatizado o manual.

Si la funcionalidad ya está en producción entonces deben proceder revisar los casos de prueba en el repositorio de jira, luego debe solicitar la capacitación a su líder a cargo mediante un correo.

El QA debe realizar el análisis de los casos de pruebas que podrían ser automatizados y luego comunicar al QE y líder técnico.

Los materiales de apoyo será el manual de usuario y configuración de ambiente de trabajo, videos de capacitación y material de conocimiento. Un repositorio en donde este la lista de las funcionalidades ya automatizados y matriz de riesgo.

El Qa automatizado se encarga de las capacitaciones a sus pares, debido a que él tiene conocimientos sobre el sistema y los pasos a seguir para que se pueda realizar las pruebas con el sistema de manera satisfactoria.

CONCLUSIONES Y RECOMENDACIONES

Luego de haber desarrollado el proyecto se llega a una seria de conclusiones para comprender de manera abreviada lo que sucedió en el proyecto. Asimismo, se brinda una seria de recomendaciones de acuerdo con el análisis de los resultados obtenidos en la elaboración del trabajo de suficiencia profesional.

Conclusiones

En virtud de lo propuesto para el objetivo 1, mediate la revisión se identificaron una serie de librerías y herramientas para la automatización de pruebas de software. Al analizar se descubre que cada herramienta posee sus propios beneficios tales como que son de código abierto, compatibilidad con diferentes navegadores, de fácil integración para automatizar pruebas E2E y también tiene algunas limitaciones. Es así como se determina que las tecnologías como Node JS, JDK y TypeScrip cumple con la mayoría de las características para el proyecto bajo estudio. Por lo que, se concluye que TypeScrip es el lenguaje de programación que obtuvo un puntaje 19 la que más se adecuada para el diseño e implementación del sistema automatización de pruebas.

En vista de la necesidad identificada en el área de calidad, se concluye que requiere una solución tecnológica. Por lo que, se procedió a diseñar el diagrama fisca y lógica que mostrará la implementación del sistema automatización de pruebas. En el cual se refleja cómo está se encuentra conformado el sistema y cómo interactúan las capas de interfaz de usuario y componentes que lo conforman. Así como los servicios en la nube que sirve para guardar los cambios de versión del Script automatizado en las ramas correspondientes como Master y DEV y los datos de prueba almacenados. A nivel local se realizó la configuración de los componentes necesarios y las tecnologías; tales como node JS, JDK y TypeScritp. Por lo tanto, se puede concluir que el diagrama planteado muestra el flujo visual de cómo funciona la solución desarrollada.

Para realizar el desarrollo de la solución, primero se estableció las fases de desarrollo del sistema. Luego se procedió con el análisis y diseño de los casos de prueba aptos a automatizar y que no requieran un token digital que para este caso se optó por 9 funcionalidades. Estos casos se redactaron en lenguaje Gherkins para una fácil comprensión. Después se realizó la instalación y configuración de los componentes para poder desplegar el ambiente de pruebas. Asimismo, se elaboró la estructura del proyecto donde se desarrollaron los scripts automatizados. Para poder validar el correcto funcionamiento del sistema se ejecutaron 118 casos de pruebas y como resultado de la ejecución se obtuvo 118 casos ejecutados en tiempo de duración de 4 horas. Con esto se generó un reporte para su posterior revisión por parte de los desarrolladores en caso tenga que revisar los defectos encontrados. Para poder realizar la validación del sistema se realizó una encuesta de satisfacción a 5 usuarios finales que utilizan el Script automatizado y el resultado fue que un 80% de los encuestados escogió la opción “Muy de acuerdo” con respecto al funcionamiento del sistema. Asimismo, para poder medir la fiabilidad de la encuesta se utilizó el método Cronbach dando como resultado 0.85. Por lo tanto, se concluye que se cumplió con el plan de trabajo para la automatización de las funcionalidades asignadas para el proyecto, asimismo con la validación del proyecto se evidencio que los usuarios finales se encuentran satisfechos con el sistema automatizado.

Al culminar con el sistema automatizado se continuo con la elaboración del plan de continuidad con el fin de garantizar la continuidad y operatividad del sistema automatizado. Para el cual se establecieron tres pilares importantes de corrección y actualización de cambios, implementación de una nueva funcionalidad y capacitación QA. Al realizar un cambio en el Script automatizado, primero debe configurar al espacio de trabajo de manera correcta y recuperar el Script desde el repositorio de la nube utilizando Git. Una vez realizado el cambio se debe ejecutar y grabar si la ejecución concluye con éxito, entonces se debe subir al repositorio en la rama DEV los cambios. Para el caso de la implementación de una nueva funcionalidad es importante primero determinar y evaluar el riesgo que presenta a nivel funcional y la factibilidad controlada. Todos los casos de prueba a automatizar deben ser aprobados por el QE y Scrum Master. Asimismo, este debe pasar por revisión técnica rigurosa por el QA automatizado para determinar la factibilidad de desarrollo. Se deben seguir las fases propuestas para el desarrollo y garantizar disponibilidad del sistema automatizado. En conclusión, mientras se siga las pautas expuestas para la continuidad del sistema automatizado se garantiza la disponibilidad de esté.

Se logro cumplir con los objetivos establecidos para el presenta trabajo de investigación en gran parte. El análisis de las tecnologías permitió seleccionar TypeScrip y librerías con node JS, además se diseñó la arquitectura lógica y física. Por lo tanto, se puede concluir que se logró el diseño y desarrollo del sistema automatizado para las pruebas de regresión. Asimismo, al ejecutar los casos de pruebas en el sistema automatizado se consiguió optimizar el tiempo de ejecución de 10 días ejecución a 4 días.

Recomendaciones

- Se recomienda seguir implementando el sistema a más funcionalidades existentes y que han pasado a producción, con esto poder abarcar más casos de pruebas que puedan ser automatizados y seguir realizando un trabajo más eficiente.
- También se recomienda implementar el sistema automatizado a otras aplicaciones que cuenta el banco, ya que se ha demostrado que ayuda en la ejecución de casos de prueba de regresión y así evitar realizar tareas repetitivas de validación.
- Seguir capacitando al personal QA para que pueda conocer más sobre las herramientas de automatización y se pueda mejorar el sistema de automatización en caso lo requiera.
- Se sugiere llevar un control de las funcionalidades ya automatizadas para aprovechar el uso del sistema automatizado y seguir optimizando el tiempo de ejecución.
- Se recomienda realizar un buen análisis herramientas de automatización en base a la necesidad requerida, debido a que un correcto análisis brindara beneficios al sistema implementado.
- Con la implementación del sistema automatizado se recomienda asignar el personal de pruebas a otro tipo de actividades que no abarca el sistema como pruebas de estrés o de rendimiento.

REFERENCIAS

Acosta, M. (2023) *Automatización de pruebas (software testing): ¿Por qué son importantes?* Recuperado el 30 de agosto de 2023, de https://canvia.com/automatizacion-pruebas-software/

Atlassian. (s.f.). *Qué es Git. Atlassian*. Recuperado el 4 de noviembre de 2023, de https://www.atlassian.com/es/git/tutorials/what-is-git

Amazon Web Services (s.f.). *¿Qué es Python? - Explicación del lenguaje Python*. Recuperado el 02 de septiembre de 2023, de https://aws.amazon.com/es/what-is/python/

Banco Bilbao Vizcaya Argentaria (2023, 25 de enero). *Tendencias 2023: ¿Reemplazará la banca digital a la atención presencial?* Recuperado el 30 de agosto de 2023, de https://www.bbva.com/es/pe/tendencias-2023-reemplazara-la-banca-digital-a-la-atencion-presencial/

Banco Bilbao Vizcaya Argentaria (2022, 30 de marzo). *Memoria Anual 2022 Informe Integrado*. https://www.bbva.pe/content/dam/public-web/peru/documents/personas/memoria-anual/Memoria-Anual-2022.pdf

Banco Central de Reserva del Perú (2021, noviembre). *Reporte de Estabilidad Financiera*. Recuperado el 30 de agosto de 2023, de https://www.bcrp.gob.pe/docs/Publicaciones/Reporte-Estabilidad-Financiera/2021/noviembre/ref-noviembre-2021.pdf

Banco Central de Reserva del Perú (2022, mayo). *Reporte de Estabilidad Financiera*. Recuperado el 30 de agosto de 2023, de https://www.bcrp.gob.pe/docs/Publicaciones/Reporte-Estabilidad-Financiera/2022/mayo/ref-mayo-2022-recuadro-4.pdf

Congreso de la República del Perú. (1996). *Ley N° 26702 de 1996. Por lo cual se expide Ley general del sistema financiero y del sistema de seguros y orgánica de la superintendencia de banca y seguros*.

Congreso de la República del Perú. (2011). *Ley N. ª 29733 de 2011. Por lo cual se expide Ley de Protección de Datos Personales*.

Cypress. (2023, 15 de junio). *Testing your app Cypress Documentation*. Recuperado el 03 de setiembre de 2023, de https://docs.cypress.io/guides/end-to-end-testing/testing-your-app

Deloitte. (2022, mayo). *La automatización de las pruebas de regresión: Uno de los retos del CIO*. Recuperado el 30 de agosto de 2023, de https://www2.deloitte.com/content/dam/Deloitte/pe/Documents/technology/Beneficios%20de%20automatizacion%20de%20pruebas%20para%20los%20CIO_SLA.pdf

El peruano (2023, 24 de abril). *Datos e inteligencia artificial incentivan el crecimiento empresarial*. Recuperado el 30 de agosto de 2023, de https://www.elperuano.pe/noticia/210910-datos-e-inteligencia-artificial-incentivan-el-crecimiento-empresarial.

Fernández, J. L. M. (2018). *Automatización de Procesos para mejorar las Pruebas de Software en el área de calidad del Banco de Crédito* [Tesis de maestría, Universidad Cesar Vallejo]. Repositorio Institucional UCV. https://repositorio.ucv.edu.pe/bitstream/handle/20.500.12692/23871/Fernandez_AJLM.pdf?sequence=1&isAllowed=y

International Business Machines Corporation. (2021, 14 de abril). *IBM documentation*. Recuperado el 02 de septiembre de 2023, de https://www.ibm.com/docs/es/i/7.3?topic=platform-java-development-kit

International Business Machines Corporation. (s.f.) *¿Qué es la prueba de software y cómo funciona?* Recuperado el 02 septiembre de 2023, de https://www.ibm.com/es-es/topics/software-testing.

Lucero, J. (2022). *Implementación de un framework de automatización de pruebas en un marco de trabajo ágil para mejorar el proceso de calidad de software en una compañía de seguros*. [Tesis de Ingeniería, Universidad Nacional Mayor de San Marcos]. Repositorio institucional Cybertesis UNMSM. https://hdl.handle.net/20.500.12672/18814

MDN Web Docs. (2023, 21 de julio). *Introducción a Express/Node - Aprende Desarrollo Web MDN*. Recuperado el 02 de septiembre de 2023, de https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

Microsoft Learn. (2023, 27 de julio). *¿Qué es Git?* Recuperado el 02 de septiembre de 2023, de https://learn.microsoft.com/es-es/devops/develop/git/what-is-git.

Microsoft Learn. (s.f.). *Información general de TypeScript*. Recuperado el 03 septiembre de 2023, de https://learn.microsoft.com/esmx/training/modules/typescript-get-started/2-typescript-overview.

Microsoft Visual Studio Code (2023, 8 agosto). *Visual Studio: IDE y editor de código para desarrolladores de software y teams*. Recuperado el 02 de septiembre de 2023, de https://visualstudio.microsoft.com/es/#vscode-section

Payments & Commerce Market Intelligence (2023). *Tendencias de pagos digitales en Latinoamérica 2023*. Recuperado el 9 de diciembre de 2023, de https://downloads.ctfassets.net/51xdmtqw3t2p/BFaypUWHwedamc5nQK9lK/bdbdfec3280480f234204bd7f5e36952/Ebook_-_Tendencias_de_pagos_digitales_2023_-_Espan__ol__1_.pdf

Playwright (s. f.). *Installation*. Recuperado el 03 de Setiembre de 2023, de https://playwright.dev/docs/intro

Profile Software Services (2021, 13 de octubre). *Qué es Gherkin: cómo usarlo y cuáles son sus elementos*. Recuperado el 03 de septiembre de 2023, de https://profile.es/blog/que-es-gherkin/.

SerenityBDD. (s. f.). *Your first web test Serenity BDD Users Manual*. Recuperado el 03 de setiembre de 2023, de https://serenity-bdd.github.io/docs/tutorials/first_test

WebdriverIO (2023, 01 de enero) *¿Por qué elegir WebdriverIO?* Recuperado el 04 de septiembre de 2023, https://webdriver.io/es/docs/gettingstarted

ANEXOS

ANEXO 1: Diagrama Gantt

Diagrama_Gantt.mpp

ANEXO 2: Instalación y configuración de herramientas

Instalación_y_configuración_de_herram

ANEXO 3: Configuración del sistema de pruebas

Configuración_del_sistema_de_pruebas