IMPLEMENTACIÓN DE PRUEBAS DE SOFTWARE AUTOMATIZADAS EN AMBIENTES WEB Y MÓVIL

Mateo Cano Bermúdez

Informe de práctica para optar al título de Ingeniero de Sistemas

Asesor

Jaime Humberto Fonseca Espinal, Ingeniero de Sistemas Especialista (Esp) en Ciencias electrónicas e informáticas

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín, Antioquia, Colombia

2022

Cita	Cano Bermúdez [1]
Referencia Estilo IEEE (2020)	[1] M. Cano Bermúdez, “Implementación de pruebas de software automatizadas en ambientes web y móvil”, Trabajo de grado profesional, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2022.

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

QA	Quality Assurance
BDD	Behavior Driven Development
BD	Base de Datos
OTP	One-Time Password
IMAPS	Internet Message Access Protocol
API	Application Programming Interface
CT	Continuous Testing

RESUMEN

La calidad del software, ha sido siempre un tema de atención en el ciclo del desarrollo del software, con un único objetivo de entregar sistemas de calidad y generar confianza ante el cliente final; ante tal necesidad, se ha evidenciado que el ciclo de pruebas tiene cierto protagonismo en un proyecto de software y que en ocasiones dependiendo de la magnitud del sistema puede tomar demasiado tiempo y ser muy costoso. Con el fin de reducir estos tiempos y costos, sin sacrificar la calidad, se propone implementar pruebas automatizadas a un Sistema de banca para empresas, realizando diferentes análisis e identificando casos de pruebas claves para ser candidatas a ejecutar de manera automática. Se logra obtener implementación de un set de pruebas automatizadas estratégicas, que reducen de manera significativa el proceso de ejecución manual, sin reducir la calidad del sistema final, de esta manera evitando errores, con la posibilidad de identificarlos de una forma más rápida para futuros reléase.

Palabras clave — **Casos de prueba, Calidad del software, Pruebas automatizadas, Script de automatización.**

ABSTRACT

Software Quality Assurance has always been a topic of consideration in the software development cycle, with the only objective of delivering quality systems and generating confidence to the final customer; given this need, it has become evident that the testing cycle has a certain role in a software project and that sometimes, depending on the magnitude of the system, it can take too much time and be very costly.

In order to reduce these times and costs, without sacrificing quality, it is proposed to implement automated testing to a business banking system, performing different analyses and identifying testing cases, crucial to be candidates to be executed automatically. The implementation of a strategic set of automated testing is achieved, which significantly reduces the manual execution process, without reducing the quality of the final system, avoiding errors and with the possibility of identifying them in a faster way for future release.

Keywords — **Test Cases, Software quality assurance, Automated testing, Automated script.**

I. INTRODUCCIÓN

El proceso de pruebas en el ciclo de vida del desarrollo de una aplicación o de un sistema, tienen como objetivo principal asegurar la calidad del software. Esto se logra inicialmente tras la elaboración de un plan o estrategia de pruebas y por consiguiente la implementación y ejecución de esta misma, a través de casos de prueba previamente definidos y que componen el sistema o funcionalidad a trabajar; identificando así casos de éxito o de fallo, con los que se pretende medir la calidad del desarrollo y poder tomar medidas correctivas.

Con la adopción de metodologías ágiles en el desarrollo de software, cada vez es más frecuente la liberación de una nueva versión de un producto, lo cual también conlleva una inversión en el tiempo en pruebas y se vuelven cada vez más repetitivas. Ante esta situación que se evidencia en un cliente de un Sistema de banca para empresas[1], cuyo sistema se encuentra desplegado en ambientes productivos, pero que de igual manera realiza despliegue de nuevas versiones de manera continua, surge la necesidad de poder diseñar scripts automatizados de pruebas, que puedan ejecutar estos casos sin ningún tipo de intervención humana, y que, a su vez puedan arrojar resultados para identificar fallos de manera temprana.

El Sistema de Banca para empresas en mención, actualmente permite realizar diferentes transacciones como transferencias, pagos de nómina y pagos a terceros, en donde cada vez se implementan nuevas funcionalidades y, en consecuencia, se obtiene un sistema bastante robusto, el cual debe mantener su estabilidad y calidad. Basados en lo anterior y teniendo en cuenta que desde la perspectiva del área de QA el fin es asegurar la calidad no solo de los nuevos desarrollos, sino también de las funcionalidades desarrolladas en momentos anteriores, se tiene un alcance de pruebas mucho más grande y al mismo tiempo más escenarios de prueba para una regresión[2].

Con motivo de mantener la calidad del sistema, se pretende implementar una automatización de pruebas funcionales a la aplicación, con el fin de realizar una regresión de pruebas tanto a las funcionalidades previamente desarrolladas como a las funcionalidades nuevas, esto con el fin de agilizar el proceso de pruebas que se vuelven repetitivas, generando confianza y valor en el producto.

[1] Es el producto de software insumo al cual se le implementará la automatización de pruebas, es un sistema bancario para empresas.
[2] Son un tipo de pruebas que se realizan cuando existe un cambio o evolución en el sistema, con el fin de encontrar inyección de errores.

II. PLANTEAMIENTO DEL PROBLEMA

En el proyecto del Sistema de Banca para empresas, el cual ya se encuentra en ambientes productivos, pero que continúa en evolución, se descubre que, al liberar nuevas versiones de la aplicación, se realizan pruebas de regresión, con el objetivo de prevenir inyección de errores y garantizar la calidad del sistema en ambientes productivos sin afectar al cliente.

Como es de saber al estar previamente desarrollado cuenta con varias funcionalidades, siendo así un sistema bastante robusto, al igual que la ejecución de una regresión de pruebas lleva bastante esfuerzo y tiempo.

En aras de analizar la situación se evidenció que las pruebas de regresión son repetitivas e incrementales, lo que hace pensar que en el futuro podrían volverse inmantenibles realizando estas mismas ejecuciones de pruebas de forma manual. Por tal razón el cliente se ve en la necesidad de buscar una estrategia más eficaz para estas pruebas.

III. JUSTIFICACIÓN

No todo tipo de pruebas pueden ser automatizadas, y tampoco es cierto que todos los casos de prueba de una funcionalidad sean candidatos para implementar una automatización. Sin embargo, para el presente caso de estudio se identificaron factores claves que lograron determinar que en su mayoría el caso de prueba del Sistema de banca para empresas cumplía con los requisitos para ser automatizado.

Como políticas de aseguramiento de calidad del software, al realizar un nuevo pase a producción, era necesario realizar pruebas de regresión, que no son más que una ejecución de los mismos casos de prueba abarcando las funcionalidades previas como las nuevas funcionalidades. Se evidencio que este trabajo se volvía repetitivo y aumentaba el tiempo de ejecución tras la inclusión de nuevas funcionalidades al sistema, al ser un sistema tan robusto, las regresiones previas a un pase a producción podrían tomar hasta 2 días en ejecución de pruebas manuales.

Se tienen inicialmente dos entradas claves para analizar una posible implementación de automatización de pruebas: la constante repetición de ejecución de casos de pruebas y el tiempo de un recurso humano para ejecutarlas (que se incrementa a través del tiempo). Ahora bien, las funcionalidades del sistema a abordar, en su mayoría no es necesaria la intervención humana (temas de biometría) o interacción con diferentes aplicativos por lo cual también es posible realizar la automatización.

Dadas las anteriores premisas y con el fin de reducir tiempos y esfuerzos para el aseguramiento de la calidad del software, se encuentra altamente provechoso realizar un análisis detallado y poder implementar un set de pruebas automatizado, el cual ahorrará temas de ejecución repetitivo de procesos.

IV. OBJETIVOS

A. Objetivo general

Implementar la automatización de pruebas en el ciclo de vida del desarrollo de software, con el fin de reducir tiempos en la ejecución manual, generando confianza y calidad en el producto.

B. Objetivos específicos

- Entender el modelo del negocio en los proyectos a través de documentación funcional, técnica y prototipos del software en cuestión.
- Analizar el contexto de los escenarios de prueba, con el fin de identificar aquellos que cumplan con las características para ser automatizables, y definir si se requiere una ruta básica o una regresión.
- Codificar scripts de automatización en entornos web y móvil, haciendo uso de frameworks según necesidad.
- Ejecutar el set de pruebas automatizado y analizar el resultado obtenido.

V. MARCO TEÓRICO

En la actualidad, el proceso de aseguramiento de la calidad de software está cada vez más inmerso en el ciclo de desarrollo, puesto el Agile Testing posibilita reducir el tiempo en los periodos de prueba, dicho proceso conocido como: metodología ágil; dicha metodología se ha convertido de gran utilidad y necesidad en la automatización de pruebas, pasando de meses de ejecución a días de desarrollo; como bien lo menciona Guzmán "Las pruebas automatizadas permiten, a través del uso de diferentes herramientas y scripts robotizados, otorgar un sinfín de ventajas para las empresas que las utilizan." [1, p. 1]. Por tal motivo la innovación en pruebas automatizadas se concibe en productividad, eficacia y solidez para las compañías. De igual manera cabe mencionar que no solo da respuesta a la rapidez en automatización, de igual manera posibilita mayor efectividad en tiempo y costo, implementación de integración continua y aseguramiento de la calidad en el desarrollo del software. "Las pruebas automatizadas requieren un menor tiempo de ejecución que las pruebas manuales. Si bien poseen un costo inicial mayor, en el tiempo dicho costo se amortigua respecto de las ganancias que trae consigo para las compañías" [1, p. 1] .

Ahora bien, dicho algunos de los beneficios de las pruebas automatizadas, en necesario reconocer como la implementación de esta metodología reduce significativamente el trabajo y la operatividad. Kumar y Mishra afirman:

> A diferencia de las pruebas automatizadas, las pruebas manuales son aquellas que son diseñadas y al mismo tiempo ejecutadas por un individuo, y que además en ocasiones son repetitivas y requieren esfuerzo y tiempo. Las pruebas automatizadas tienen varias ventajas y siempre van en pro al aseguramiento de la calidad de la aplicación, ya que brindan cobertura de pruebas formales, evitan errores humanos y aceleran el proceso de ejecución de pruebas. Además, dado que acelera el proceso de ejecución, es la solución más eficaz para cumplir con los estrictos plazos. [2, pp. 8-15]

En este orden de ideas las pruebas automatizadas se convierten en el mecanismo de ejecución y productividad, donde se reconocen las ventajas expuestas anteriormente, y como estas compiten con la calidad esperada en la automatización, sin olvidar que este tipo de metodología

ágil requiere una serie de lineamientos para su ejecución, como bien lo menciona Serna, Martínez y Tamayo "Para obtener beneficios con la automatización las pruebas deben ser cuidadosamente seleccionadas y aplicadas, es necesario contar con un checklist para identificar escenarios candidatos para una automatización y que agreguen valor al proceso de pruebas, porque la calidad de este proceso es independiente de la calidad de la prueba" [3, pp. 337-352]

Por último, cabe mencionar que esta metodología ha permeado en los escenarios y que no solo es un recurso de innovación, sino que aporta un sinfín considerable de recursos para su desarrollo, por ejemplo, de menciona en la revista digital.ai que "Existen en el mercado un sinfín de herramientas que permiten diseñar estos scripts, un ejemplo es Serenity BDD, el cual es un Framework que admite múltiples soluciones de pruebas de aceptación automatizadas. Puede usar la herramienta junto con JUnit para escribir rápidamente criterios aceptables limpios y fáciles de mantener o integrarlo con WebDriver para probar aplicaciones web en Selenium" [4]. Es así como este tipo de metodología aporta un acercamiento conceptual y practico a la ejecución de esta propuesta de implementación, que consolida su mirada en la ejecución y desarrollo de este modelo alternativo para el desarrollo de pruebas de software.

VI. METODOLOGÍA

Inicialmente se parte desde el patrón de Screenplay, el cual permite realizar la definición de las interacciones en lenguaje natural (Gherkin) y orientado a BDD, este patrón integra la librería de Serenity BDD, con la cual se podrá escribir los test de una manera más rápida y mantenible en el tiempo; de igual manera permite crear una arquitectura de desarrollo de las pruebas de modo tal que sea un código simple, legible y altamente reutilizable. Se compone de 5 pilares fundamentales que podemos observar en la **Figura 1.** los **Actores**, quien representa el usuario final y el ejecutor de la prueba; las **Habilidades**, que representan las capacidades que tiene el actor de interactuar con un sistema; las **Interacciones** que son aquellas acciones que permiten manipular o recorrer el sistema (clicks, lectura, ingreso de datos); las **Tareas** que son la descripción de la acción a realizar por el actor en una abstracción a nivel de lenguaje del negocio; y por último las **Questions** que son las validaciones que se realizan a la prueba, el resultado esperado.

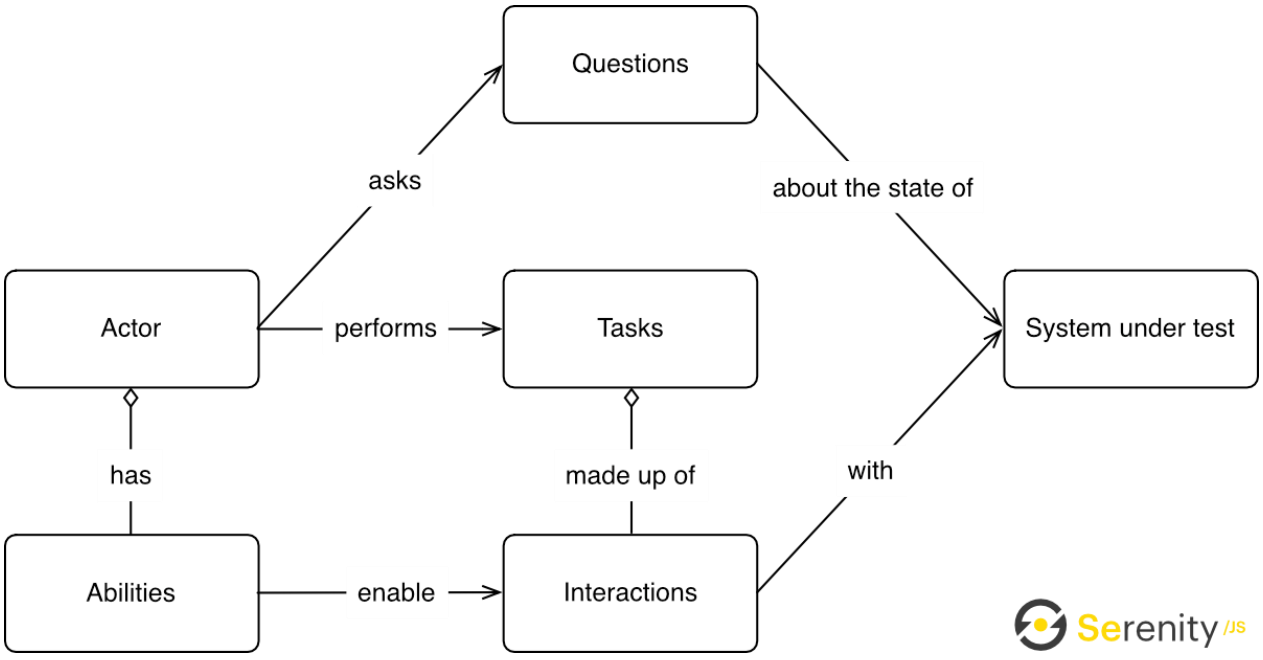

Figura 1. El patrón Screenplay

Nota: fuente https://serenity-js.org/handbook/design/images/the-screenplay-pattern.png

Tomando como referente los parámetros expuestos anteriormente, se toma como base la Pirámide de Cohn, donde enfoca las pruebas automatizadas en el pico de la pirámide. Al ser unas pruebas a nivel de interfaz de usuario se sitúan casi al final de esta, donde lo ideal es que la

ejecución de pruebas tome menos esfuerzo y esto se logra a través del uso del patrón Screenplay, contando con un nivel estable de pruebas automatizadas, y así obteniendo una mayor cobertura de pruebas.

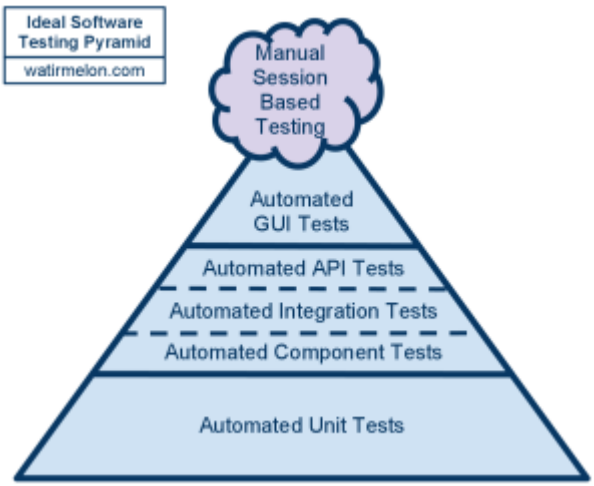

Figura 2. Pirámide de pruebas de Cohn

Nota: fuente https://www.javiergarzas.com/2015/01/automatizacion-pruebas.html

Al visualizar el enfoque, se inicia con la contextualización del sistema a tratar, el cual es una aplicación web de banca para empresas; dicho aplicativo se encuentra previamente desarrollado y en ambiente de producción. Como antecedentes se encontró que el sistema no contaba con pruebas automatizadas, y que, en casos de requerir algún tipo de prueba, todas se realizaban de manera manual.

Desde el área de QA ya se tenían previamente un set de casos de prueba definidos, lo cual fue útil para identificar las diferentes funcionalidades y su cobertura a nivel de pruebas. Con el fin de identificar aquellos casos de prueba candidatos para ser automatizados, se elaboró un checklist con diferentes criterios a tener en cuenta para automatizar un caso de prueba y así determinar si este podía ser automatizado o no.

Para la construcción de los scripts, fue necesario contar con un ambiente de pruebas en ambiente de QA, en el cual se realizaron los mapeos de los elementos a interactuar (botones, campos de texto, checkbox…) y en el cual se ejecutaron los scripts a medida que se construían para la estabilización de los escenarios. Es importante anotar que la estabilización fue contemplada debido a los cambios que presentó el software durante el proceso de desarrollo.

Como resultado del proceso anterior se logró contar con los escenarios de prueba seleccionados automatizados y listos para ser ejecutados en uno o dos ciclos de prueba.

Al contar con la totalidad de casos de prueba automatizados, se realiza una ejecución limpia del script de automatización, es decir con la totalidad de casos de prueba exitosos; para esto fue necesario contar con una disponibilidad al 100% del ambiente de pruebas, ya que en ocasiones los test fallaban por inestabilidad en los servicios.

En el siguiente punto, se generó el reporte de ejecución de pruebas, el cual se obtiene desde la herramienta de Serenity, en este informe se pudo observar de manera clara información como el tiempo de ejecución de pruebas, el total de escenarios y casos de pruebas ejecutados, el resultado (éxito o fallo) de cada caso de prueba y las evidencias de cada prueba.

Para concluir se continuó con el análisis del reporte de ejecución generado en el paso anterior, con el fin de identificar fallas y planes correctivos.

VII. RESULTADOS

De los Casos de prueba previamente diseñados, se inicia la elaboración de un checklist de automatización, esto con el fin de identificar de aquellos escenarios de prueba, cuales son candidatos a realizar la automatización. Se toman varios criterios para determinar la estrategia de automatización, entre ellos están:

- **Requiere intervención manual:** Este criterio define si para la ejecución de la prueba es necesario que el usuario intervenga de alguna forma manual, o que al sistema se le deba realizar alguna configuración manual que no esté al alcance de la automatización. El criterio se encuentra de primero, ya que es excluyente, es decir si la prueba necesita intervención manual, no es apto para automatizar.
- **Caso de prueba repetitivo:** Determina si el Caso de prueba se debe ejecutar recurrentemente.
- **De alto riesgo para el negocio:** Se clasifica entre Alto, Medio y Bajo, donde alto es el nivel de importancia que tiene dicha funcionalidad en el negocio y la capacidad de generar un impacto alto ante una indisponibilidad de esta.
- **Aplica pruebas de regresión:** Indica si el caso de prueba es ejecutado siempre en las pruebas de regresión previas a un paso a producción.
- **Consume mucho esfuerzo:** Se puede determinar por medio del tiempo que toma realizar la ejecución de la prueba de forma manual, incluyendo toma de evidencias.

Con los criterios anteriores se generó el checklist y se evaluó cada caso de prueba; en la **TABLA I** a modo de ejemplo podemos observar que para dos grupos de funcionalidades (Administración de terceros y Transacciones comunes) existen casos de prueba que fueron excluidas de implementar la automatización.

TABLA I

CHECKLIST PARA LA AUTOMATIZACIÓN DE CASOS DE PRUEBA

Grupo	Caso de prueba	Requiere intervención manual	Caso de prueba repetitivo	Riesgo	Aplica pruebas de regresión	Consume mucho esfuerzo	Candidato automatizable
Administrativas	Eliminación de terceros mismo banco	No	Si	Medio	Si	No	**Si**
	Eliminación de terceros nómina	No	Si	Medio	Si	No	**Si**
	Eliminacion de terceros otros bancos	No	Si	Medio	Si	No	**Si**
	Edición de terceros mismo banco	No	Si	Alto	Si	Si	**Si**
	Edición de terceros nómina	No	Si	Alto	Si	Si	**Si**
	Edición de terceros otros bancos	No	Si	Alto	Si	Si	**Si**
	Administración de plantillas	Si	Si	Medio	Si	Si	**No**
	Generar reportes de auditoría	Si	Si	Alto	Si	Si	**No**
Transacciones comunes	Cambio de clave voluntario	No	No	Alto	Si	No	Si
	Login	No	Si	Alto	Si	No	Si
	Cambio obligatorio de clave	Si	Si	Medio	Si	Si	**No**

Como resultado del análisis se decidió excluir tres casos de prueba, que si miramos detalladamente la razón principal es por su dependencia a una intervención manual.

Para los casos de prueba excluidos de las transacciones administrativas, era necesario interactuar con elementos nativos del sistema operativo, como el gestor de archivos tanto para la carga y descarga, lo cual no es posible realizar la implementación por razones técnicas de la herramienta de automatización utilizada.

Para el caso de cambio obligatorio de clave, se excluye gracias a que es necesaria una intervención a nivel de base de datos para actualizar el estado del usuario, y solicitar un cambio de clave obligatorio. Si bien, se da la claridad que técnicamente se podría realizar la conexión de BD desde el script de pruebas, sin embargo, no era parte del alcance para la entrega.

Tras identificar dentro del set de casos de pruebas aquellos que no eran candidatos a ser automatizados por dependencias manuales, se analizaron los otros criterios definidos en el Checklist, y tomando también en cuenta que se trata de un Sistema de banca para empresas, y la prioridad y el nivel de riesgo que representan las funcionalidades a las que pertenecen, se eligieron los siguientes escenarios y casos de pruebas a los que se implementó la automatización:

TABLA II
CASOS DE PRUEBA SELECCIONADOS PARA LA AUTOMATIZACIÓN

Funcionalidad	**Caso de prueba**
Administración de terceros	Inscripción de cuenta de terceros (otros bancos)
	Inscripción de cuentas de nómina de terceros
	Inscripción de cuentas de tercero mismo banco
	Edición de cuentas de terceros (otros bancos)
	Edición de cuentas de nómina de terceros
	Edición de cuentas de tercero mismo banco
	Eliminación de cuentas de terceros (otros bancos)
	Eliminación de cuentas de nómina de terceros
	Eliminación de cuentas de tercero mismo banco
Consulta de movimientos	Consulta de transacciones completadas
	Consulta de transacciones pendientes
Transacciones comunes	Login y autenticación
	Cambio de clave voluntario

Al aplicar la metodología bajo el modelo de Screenplay, se obtiene un desarrollo del script de pruebas, donde se dividen en escenarios y casos de prueba. Los escenarios corresponden a una funcionalidad completa, como ejemplo una inscripción de cuentas, y los casos de prueba corresponden a un caso en específico dentro de ese escenario, por ejemplo, inscripción de cuentas

de terceros o inscripción de cuentas propias. Por consiguiente, para un Escenario obtuvimos uno o más casos de prueba automatizados.

En la Figura 3, se puede observar un ejemplo de la forma en la que se construyeron los escenarios de prueba a través del lenguaje Gherkin, siendo un lenguaje más enfocado al negocio y que permitió realizar un diseño de prueba basado en BDD determinando tres bloques importantes: el Dado (precondiciones), el Cuándo (acciones a realizar para la prueba), y el Entonces (Resultado esperado y validación).

```
Esquema del escenario: realizar eliminacion exitosa de terceros otros bancos
  Dado que User ingresa a Coltefinanciera empresas con usuario usuario y clave
    | UsuarioPreparador   | ClavePreparador   |
    | <UsuarioPreparador> | <ClavePreparador> |
  Cuando deseo eliminar una cuenta terceros
    |grupoEmpresarial    | nombreAlias   |  numeroCuenta |
    |<grupoEmpresarial>  | <nombreAlias> |  <numeroCuenta>|
  Entonces puedo ver el mensaje de eliminacion exitosa
    | mensajeConfirmacion     |
    | <mensajeConfirmacion>   |
  Ejemplos:
    | UsuarioPreparador   | ClavePreparador | grupoEmpresarial         | nombreAlias    | numeroCuenta
    | EJECUTORISMAR1      | Todo0517        | COLTE2                  |diegotestalias3   | 454545454
    | EJECUTORISMAR1      | Todo0517        | COLTE2                  |Prueba interbancaria 17   | 0023453
```

Figura 3. Diseño de escenario de prueba en lenguaje Gherkin

Luego de generar el artefacto anterior que se define como Feature, se realizó el diseño de los StepDefinitions, que no es más que la traducción del lenguaje Gherkin, a un lenguaje de programación, que en este caso se realizó con Java.

Ahora para el mismo escenario, en la Figura 4 se plasma el StepDefinition para la eliminación de terceros otros bancos, se puede observar el Cuándo y el Entonces definidos en el Feature anterior, en el StepDefinition, se realiza un llamado de cada Tarea que fue implementada para poder llevar a cabo la ejecución del caso de prueba.

```java
package co.com.todo1.certificacion.stepdefinitions.administrativas;

import ...

public class AdministrarTercerosStepDefinition {
    @Cuando("deseo eliminar una cuenta terceros")
    public void deseoEliminarUnaCuentaTerceros(List<Map<String, String>> datos) {
        theActorInTheSpotlight().attemptsTo(
                EliminarTercero.inscrito(
                        datos.get(0).get("grupoEmpresarial"),
                        datos.get(0).get("nombreAlias"),
                        datos.get(0).get("numeroCuenta")
                )
        );
    }

    @Entonces("puedo ver el mensaje de eliminacion exitosa")
    public void puedoVerElMensajeDeEliminacionExitosa(List<Map<String, String>> datos) {
        theActorInTheSpotlight().should(GivenWhenThen.seeThat(
                EliminacionDeTerceros.exitoso(), Matchers.containsString(datos.get(0).get("mensajeConfirmacion"))
        ));

    }
```

Figura 4.Clase StepDefinition

En el StepDefinition que se observa en la Figura 4, se realiza en el **@Cuando** el llamado a la Tarea de *EliminarTercero.inscrito()*, en esta clase es donde se implementa la interacción paso a paso de la ejecución del caso de prueba Figura 5.

```java
public class EliminarTercero implements Task {

    String grupoEmpresarial, nombreAlias, numeroCuenta;

    public EliminarTercero(String grupoEmpresarial, String nombreAlias, String numeroCuenta) {
        this.grupoEmpresarial = grupoEmpresarial;
        this.nombreAlias = nombreAlias;
        this.numeroCuenta = numeroCuenta;
    }

    @Override
    public <T extends Actor> void performAs(T actor) {
        actor.attemptsTo(
                WaitUntil.the(LBL_CARGANDO, isNotVisible()),
                Click.on(OPCION_MODULO.of(ADMINISTRACION)),
                Click.on(OPCION_MENU.of(TERCEROS)),
                WaitUntil.the(LBL_CARGANDO, isNotVisible()),
                Click.on(BTN_FILTRO),
                Enter.theValue(grupoEmpresarial).into(TXT_GRUPO_EMPRESARIAL),
                Enter.theValue(nombreAlias).into(TXT_NOMBRE_PERSONALIZADO),
                Enter.theValue(numeroCuenta).into(TXT_CUENTA),
                Click.on(BTN_ELIMINAR.of(nombreAlias)),
                WaitUntil.the(BTN_CONTINUAR_TERCEROS,isVisible()),
                Click.on(BTN_CONTINUAR_TERCEROS)
                );
    }
```

Figura 5. Clase que implementa la clase Task para la descripción del paso a paso del caso de prueba.

Durante la codificación del Script se detectó que la mayoría de funcionalidades utilizan un método de generación y obtención de código OTP, al ser un sistema de banca, es necesario para brindar un mayor nivel de seguridad al finalizar una transacción. Este código OTP es enviado al correo asociado a la cuenta del usuario, por lo que fue necesario implementar una interacción en la automatización y poderse conectar vía protocolo IMAPS al email asociado, esto se logró a través del API de JavaMail.

Una vez contando con el Script de pruebas automatizado y con un alto porcentaje en su estabilidad, se realizó un ciclo limpio de ejecución[3] el cual se tomó para generar el insumo del reporte de ejecución de Serenity, el cual arrojó información clave sobre el estado de cada caso de prueba, como el tiempo de ejecución, el caso de éxito o falla y las respectivas evidencias en la interfaz gráfica.

En la **TABLA III** se puede apreciar el tiempo de ejecución arrojado en el reporte para cada caso de prueba, los cuales en promedio no tomaron más de 2 minutos en una ejecución automatizada, a su vez, se puede observar el tiempo de ejecución manual que se invertía para los mismos casos de prueba.

TABLA III
TIEMPO DE EJECUCIÓN AUTOMATIZADA PARA CADA CASO DE PRUEBA

Funcionalidad	Caso de prueba	Tiempo (en Minutos)
Administración de terceros	Inscripción de cuenta de terceros (otros bancos)	2,92
	Inscripción de cuentas de nómina de terceros	1,29
	Inscripción de cuentas de tercero mismo banco	1,37
	Edición de cuentas de terceros (otros bancos)	2,27
	Edición de cuentas de nómina de terceros	1,12
	Edición de cuentas de tercero mismo banco	1,25
	Eliminación de cuentas de terceros (otros bancos)	2,27
	Eliminación de cuentas de nómina de terceros	1,04
	Eliminación de cuentas de tercero mismo banco	1,2
Consulta de movimientos	Consulta de transacciones completadas	1,2

[3] Al hablar de un ciclo de pruebas limpio es cuando la totalidad de casos de prueba son exitosos.

	Consulta de transacciones pendientes	0,87
Transacciones comunes	Login y autenticación	0,54
	Cambio de clave voluntario	1,65
	TOTAL	18,99 min

En total se puede observar que la ejecución del set de pruebas automatizadas, el cual comprende 16 casos de prueba, se toma un tiempo aproximado de 19 minutos, es decir un promedio de 1,5 min por caso de prueba que es menor a la ejecución de pruebas manuales, análisis que será desarrollado en la Tabla IV de la Discusión.

El reporte que es generado gracias al framework de Serenity, es almacenado en la raíz del proyecto de automatización, es una interfaz web local la cual recopila el estado de cada prueba realizada como las evidencias y la captura de errores.

En las Figuras 6, 7 y 8, se puede observar más detalladamente, la interfaz gráfica y la forma de muestra de resultados dada por el reporte.

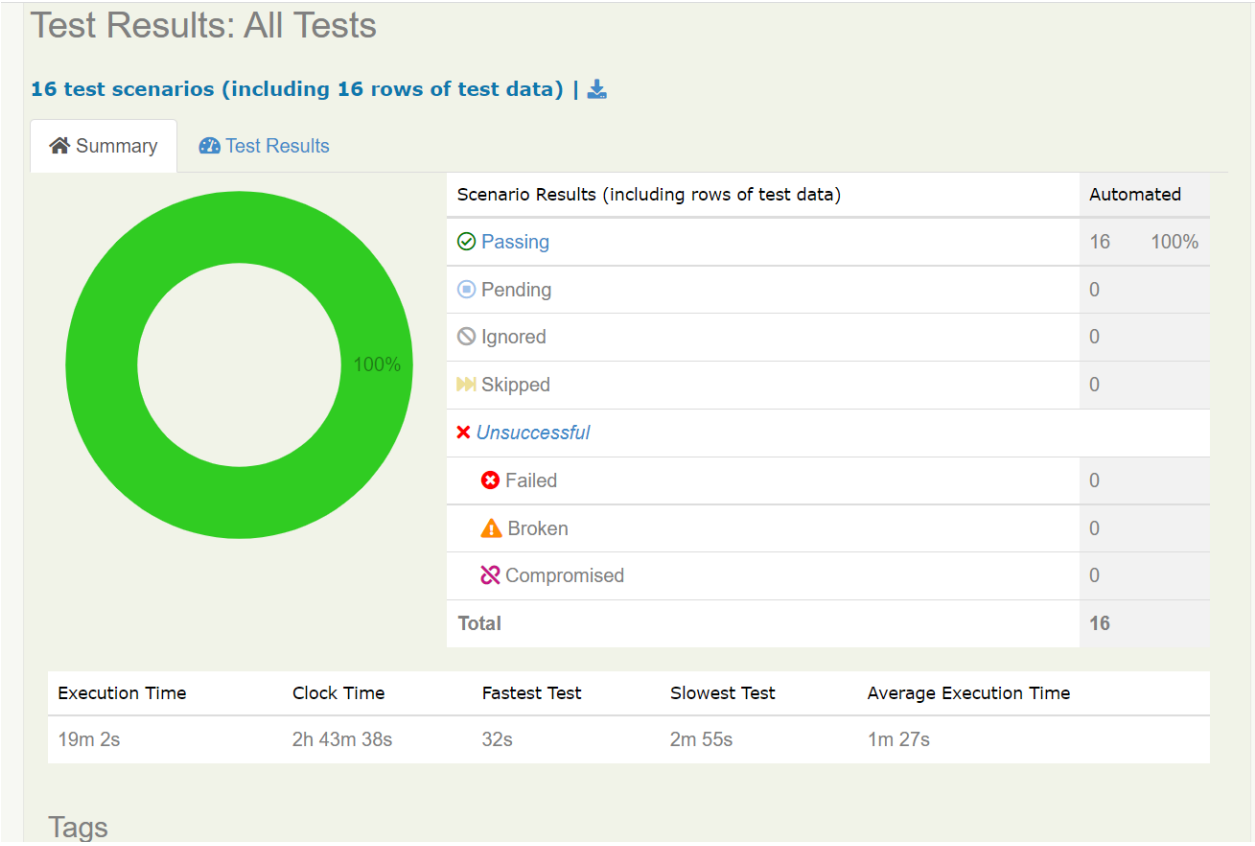

Figura 6. Dashboard del reporte de Serenity

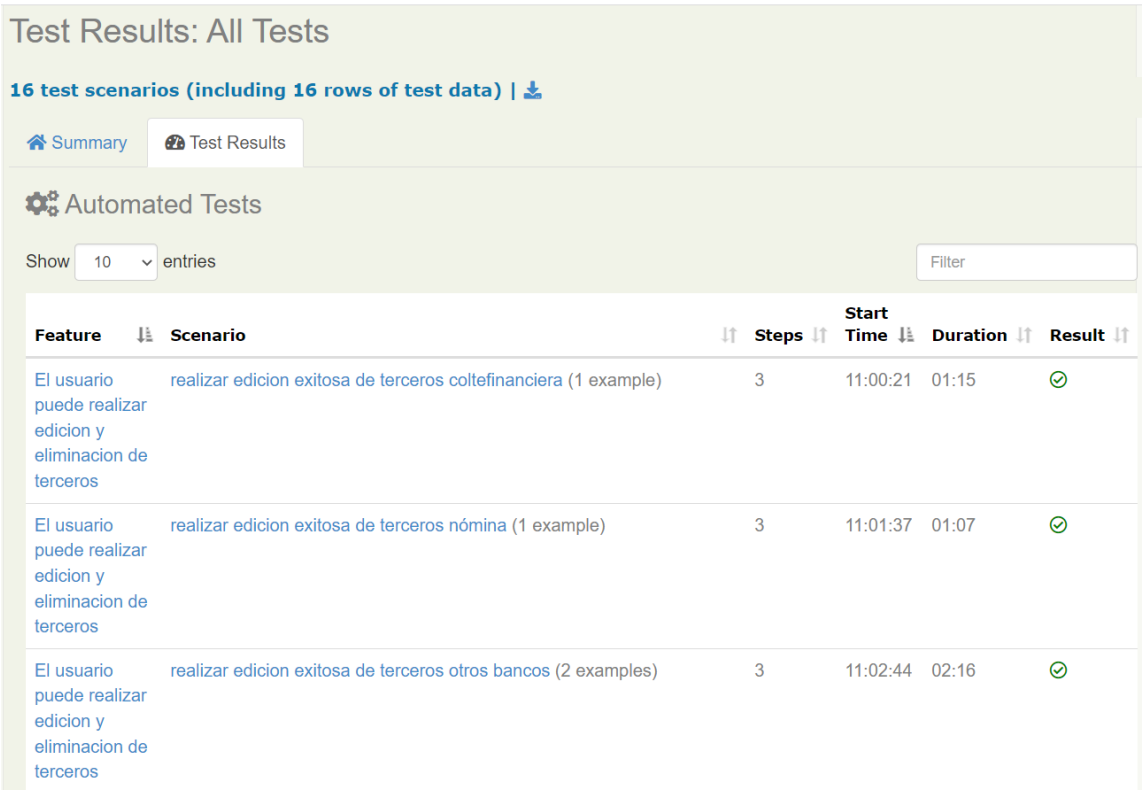

Figura 7. Resultado desglosado para cada escenario de prueba.

Figura 8. Resultado detallado de un caso de prueba, donde se observa la descripción del lenguaje natural para cada paso y las evidencias en cada uno.

Al diseñar los escenarios de prueba en lenguaje Gherkin, se puede ver en la Figura 9, que los pasos están detallados en un lenguaje natural, de tal modo que este reporte pueda leerlo y analizarlo una persona con conocimientos técnicos como también un analista funcional o de negocio, que no tenga bases técnicas. Se encuentra un muy buen nivel de detalle en el reporte, donde se puede identificar cada acción (clicks, entradas de valores…) que realiza la automatización para completar un flujo de prueba.

De igual forma es importante aclarar que los tiempos fueron tomados en referencia a un único reporte, puede ser variable el tiempo de ejecución y esto depende a diferentes factores como la capacidad de procesamiento de la máquina donde se realiza la automatización, la velocidad de la red o de la estabilidad del ambiente de pruebas.

VIII. DISCUSIÓN

Los resultados arrojados tras la ejecución automatizada de pruebas exhibieron claramente tiempos muy bajos de ejecución, donde pudimos observar que el caso de prueba con más tiempo empleado fue de casi 3 minutos y el más rápido de tan sólo 32 segundos.

Como se observa en la **TABLA IV**, añadiéndole el tiempo de ejecución de pruebas anuales, podemos observar una reducción significativa.

TABLA IV

PORCENTAJE DE REDUCCIÓN EN EL TIEMPO DE EJECUCIÓN MANUAL VS AUTOMATIZADA

Funcionalidad	**Caso de prueba**	**Tiempo de ejecución (en minutos)**		**% De reducción de tiempo**
		Manual[4]	**Automatizada**	
Administración de terceros	Inscripción de cuenta de terceros (otros bancos)	10	2,92	70,8%
	Inscripción de cuentas de nómina de terceros	6	1,29	78,5%
	Inscripción de cuentas de tercero mismo banco	6	1,37	77,17%
	Edición de cuentas de terceros (otros bancos)	8	2,27	71,63%
	Edición de cuentas de nómina de terceros	7	1,12	84%
	Edición de cuentas de tercero mismo banco	7	1,25	82,14%
	Eliminación de cuentas de terceros (otros bancos)	6	2,27	62,17%
	Eliminación de cuentas de nómina de terceros	4	1,04	74%
	Eliminación de cuentas de tercero mismo banco	4	1,2	70%
Consulta de movimientos	Consulta de transacciones completadas	4	1,2	70%

[4] La medición de los tiempos en la ejecución manual fue tomada como referencia de estimaciones y ejecuciones previas que se venían realizando de forma periódica.

	Consulta de transacciones pendientes	3	0,87	71%
Transacciones comunes	Login y autenticación	3	0,54	82%
	Cambio de clave voluntario	5	1,65	67%
	TOTAL	73 min	18,99 min	74%

Realizando un análisis más detallado, se puede observar que mientras la ejecución manual de los casos de prueba toma un total de 73 minutos, en los cuales está comprendido la prueba del flujo en cuestión y la toma de evidencias, en la ejecución automatizada tarta solo 19 minutos, contando las mismas acciones realizadas en una ejecución manual. Cabe mencionar que los tiempos de ejecución manual, enfatizan únicamente en la ejecución y no en actividades adicionales como tomas de evidencia, análisis de casos de falla o éxito, entre otros; al igual que los factores de comportamiento del analista de pruebas (velocidad, atención, concentración, diligencia) influenciando en la variabilidad de los tiempos, los cuales pueden incrementar.

Si no enfocamos en el porcentaje de reducción de tiempo, notamos que todas las funcionalidades perciben una disminución de más del 60% y en algunos casos supera el 80% de disminución en el tiempo de ejecución. Al mirar los resultados de una forma global, la comparativa entre los tiempos totales entre la ejecución manual y automática, vemos una reducción final del 74% del tiempo total de ejecución de todo el set de pruebas.

IX. CONCLUSIONES

Se logró obtener resultados positivos en cuanto a la ejecución de pruebas automatizadas, se evidenció una reducción significativa en los tiempos de ejecución de prueba, con esta implementación se da soporte y mantenibilidad a través del tiempo a las funcionalidades que componen el sistema, sin ocupar sobreesfuerzos y reprocesos al ejecutar recurrentemente los mismos casos de pruebas.

Una reducción promedio de 74% en la ejecución de pruebas son de ganancia para el negocio, ya que al reducir el tiempo empleado en pruebas de regresión se pueden utilizar en otros temas que añadan valor al negocio, de igual manera, esta reducción de tiempos, reducen costos sin afectar la calidad del sistema, teniendo a la mano una ejecución de pruebas automática que puede identificar fallas de forma más temprana.

En casos que se presenten nuevas funcionalidades para el sistema, no va a ser un tema tedioso aumentar los casos de prueba por tema de esfuerzos o tiempo, puesto que se debería continuar con la automatización de futuras funcionalidades, para así tener un alcance y una cobertura suficiente para futuras ejecuciones.

X. RECOMENDACIONES

Ya contando con un script automatizado de pruebas, es posible mirar hacia un posible Continuous Testing (CT), el cual se trata de que dichos scripts hagan parte de cada reléase o entrega de un nuevo versionamiento del sistema, esto se logra a través de herramientas DevOps, las cuales permiten que antes de realizar un despliegue en ambientes productivos se realice de forma automática la ejecución de pruebas, de manera que se puedan detectar fallas o riesgos previamente a realizar el despliegue, o en el mejor de los casos entregar una ejecución de pruebas limpia.

Es importante también que la cultura de automatización de pruebas continúe a través del tiempo con la inclusión de nuevas funcionalidades, lo ideal es realizar nuevos casos de pruebas o flujos a automatizar cuando se desarrolle un nuevo componente o funcionalidad, esto con el fin de darle cobertura a nuevos desarrollos.

La mantenibilidad del script de automatización es también un tema de interés, ya que, con el tiempo se pueden presentar inyección de código en el desarrollo y con esto afectar de alguna manera los mapeos de los elementos a interactuar, por lo que es recomendable ejecutar de manera periódica la automatización e identificar de manera temprana el tipo de falla, para tomar medidas rápidas.

Para concluir el script de automatización es algo que se debe ir alimentando para aumentar la cobertura de pruebas y con esto realizar un aseguramiento de la calidad del software más exhaustivo, también se deben identificar oportunidades de mejora o inclusión de nuevos escenarios o casos de prueba.

REFERENCIAS

[1] C. Guzmán, 3 Octubre 2018. [En línea]. Available: https://medium.com/karibu-blog/ventajas-de-la-automatizaci%C3%B3n-de-pruebas-2a51a3690814. [Último acceso: 22 01 2022].

[2] D. Kumar y K. Mishra, «The Impacts of Test Automation on Software's Cost, Quality and Time to Market,» *Procedia Computer Science,* vol. 79, nº 1, pp. 8-15, 2016.

[3] E. Serna M., R. Martínez M. y P. A. Tamayo O., «Un modelo para determinar la madurez de la automatización de las pruebas del software como área de investigación y desarrollo,» *Comp. y Sist. [online],* vol. 21, nº 2, pp. 337-352, 2017.

[4] G. Arieli, «digital.ai,» 16 08 2020. [En línea]. Available: https://digital.ai/catalyst-blog/working-with-serenity-bdd-framework-an-overview. [Último acceso: 22 01 2022].