## Parte 1: Importancia del software y la calidad

En la sociedad contemporánea, el software ha evolucionado hasta convertirse en un elemento omnipresente que permea prácticamente todas las esferas de la existencia humana. Su influencia se extiende desde los dispositivos móviles que se utilizan diariamente hasta los complejos sistemas que controlan infraestructuras críticas, redefiniendo cómo las personas interactúan con la tecnología y entre sí mismas. Esta transformación digital ha generado un ecosistema donde la tecnología software ya no es una simple herramienta auxiliar, sino un componente esencial para el funcionamiento de la civilización actual. [1]

Esta creciente dependencia del software ha puesto de manifiesto la vital importancia de garantizar su calidad. Un error en un sistema bancario puede paralizar transacciones financieras en todo el mundo; una falla en equipos médicos podría comprometer vidas humanas; un defecto en sistemas de control industrial podría desencadenar catástrofes ambientales. Las implicaciones económicas de estas fallas son igualmente alarmantes, con estimaciones que sitúan las pérdidas anuales en decenas de miles de millones de dólares. [2]

Más allá del impacto financiero inmediato, el software defectuoso genera una cascada de consecuencias negativas: ciclos de desarrollo que se alargan indefinidamente por la necesidad constante de corregir errores; la erosión gradual pero implacable de la confianza de los usuarios; la frustración de los clientes traducida en abandono; y en casos extremos, disputas legales que pueden amenazar la supervivencia misma de las organizaciones. La detección temprana de defectos no es simplemente una buena práctica, sino una necesidad económica, pues el costo de corrección se multiplica exponencialmente conforme avanza el ciclo de desarrollo. [3]

## Parte 2: Aseguramiento de la calidad y pruebas automatizadas

El aseguramiento de la calidad del software representa un enfoque integral y proactivo que atraviesa todo el ciclo de vida del desarrollo. Lejos de ser una fase aislada que se ejecuta al final del proceso, constituye una filosofía orientada a prevenir defectos desde las etapas iniciales de conceptualización hasta el despliegue y mantenimiento. Este paradigma establece que la calidad no se puede "probar" posteriormente en un producto, sino que debe "construirse" desde su concepción. [1]

En este contexto, las pruebas de software emergen como una disciplina fundamental cuyo objetivo principal no es demostrar la ausencia de errores, sino descubrir su presencia. Esta distinción, aunque sutil, representa un cambio fundamental en el enfoque: las pruebas eficaces requieren una mentalidad orientada a la detección de problemas más que a la validación de funcionalidades ya implementadas. [2]

La automatización de pruebas ha revolucionado esta disciplina, permitiendo la ejecución sistemática, consistente y repetible de casos de prueba sin intervención humana directa. Esta aproximación transformadora proporciona beneficios que van más allá de la simple reducción de esfuerzo manual. La automatización permite:

Ejecutar conjuntos extensos de pruebas en periodos reducidos, ampliando significativamente la cobertura del producto y detectando problemas que podrían pasar desapercibidos en pruebas manuales limitadas. [3]

Establecer mecanismos de verificación continua que aseguran que funcionalidades previamente validadas no se deterioran con la introducción de nuevas características, protegiendo así la integridad del sistema a lo largo de su evolución. [2]

Eliminar la variabilidad inherente a la ejecución humana, garantizando que cada caso de prueba se ejecute exactamente de la misma manera en cada iteración, independientemente de factores como el cansancio o la distracción. [3]

Optimizar recursos a largo plazo, pues aunque la inversión inicial en automatización puede ser considerable, el retorno se materializa a través de ciclos de desarrollo más cortos, mayor calidad del producto y reducción de costos de mantenimiento. [1]

No obstante, sería erróneo considerar la automatización como una panacea universal. Las pruebas manuales conservan un valor irreemplazable en aspectos como la evaluación de usabilidad, las pruebas exploratorias y aquellas situaciones donde la intuición y experiencia humanas resultan insustituibles. La estrategia óptima consiste en una combinación equilibrada de ambos enfoques, aprovechando las fortalezas de cada uno para crear un ecosistema de calidad robusto y eficiente. [3]

## Parte 3: Introducción a la problemática y enfoque

La transición hacia metodologías ágiles y DevOps en la industria del desarrollo de software ha evidenciado la necesidad crítica de integrar pruebas automatizadas como parte fundamental del ciclo de desarrollo. Sin embargo, múltiples organizaciones siguen operando sin incorporar estas prácticas de manera sistemática en sus proyectos, lo que genera ineficiencias operativas y riesgos potenciales para la calidad del producto final. [4]

Este es precisamente el caso de Avangenio, una empresa de desarrollo de software que, hasta hace poco, llevaba a cabo sus proyectos sin contar con equipos dedicados a la automatización de pruebas. Los procesos de verificación y validación se realizaban predominantemente de forma manual, lo que implicaba limitaciones significativas en términos de cobertura, consistencia y capacidad de regresión. Esta situación, común en muchas empresas del sector, suele derivar de factores como la percepción de alto costo inicial de implementación, la escasez de personal especializado o la subestimación del retorno de inversión de las pruebas automatizadas. [5]

La problemática se hizo especialmente evidente cuando un cliente solicitó explícitamente la inclusión de pruebas automatizadas para el módulo "Carga-Descarga" de un nuevo proyecto. Esta exigencia externa actuó como catalizador para el cambio organizacional, poniendo de manifiesto la necesidad de desarrollar capacidades internas en el ámbito de la automatización de pruebas. La situación planteó desafíos importantes: conformar un equipo especializado, seleccionar las herramientas adecuadas, establecer metodologías de trabajo y garantizar la adecuada integración con los procesos de desarrollo existentes.

El enfoque adoptado para abordar esta problemática se basó en la formación de un equipo dedicado de aseguramiento de calidad (QA) con énfasis en automatización, del cual el presente informe documenta la experiencia desde una perspectiva práctica. La estrategia implementada se centró en la capacitación intensiva del nuevo equipo mediante un periodo de familiarización con un módulo previamente desarrollado ("Añejamiento"), lo que permitió adquirir las competencias necesarias antes de afrontar el desarrollo del módulo principal objeto de las pruebas automatizadas.

En cuanto a la selección tecnológica, se optó por Playwright como framework principal de automatización, una decisión fundamentada en su creciente adopción en la industria, su compatibilidad con pytest y la disponibilidad de funcionalidades avanzadas en su versión estándar que otras herramientas solo ofrecen en versiones premium. Esta elección representa un elemento estratégico para la resolución de la problemática, pues facilita la implementación de pruebas robustas con una curva de aprendizaje optimizada. [6]

## Parte 4: Objetivos

### Objetivo General

Implementar un proceso sistemático de pruebas automatizadas para el módulo "Carga-Descarga" del proyecto en desarrollo, que permita mejorar la calidad del producto software, reducir los tiempos de verificación y establecer las bases metodológicas para la incorporación de prácticas de automatización en futuros proyectos de Avangenio.

### Objetivos Específicos

1. Desarrollar habilidades técnicas en el framework Playwright para la automatización de pruebas, mediante su estudio y aplicación práctica en el módulo "Carga-Descarga", aprovechando su integración con Python y sus capacidades avanzadas frente a otras herramientas del mercado.
2. Diseñar e implementar un conjunto de casos de prueba automatizados para el módulo "Carga-Descarga" que garanticen la correcta verificación de sus funcionalidades críticas, siguiendo patrones de diseño que favorezcan la reutilización y el mantenimiento del código.
3. Evaluar y optimizar los scripts de prueba existentes para mejorar su eficiencia, escalabilidad e independencia de datos, incorporando técnicas de parametrización y gestión de datos externos que faciliten su ejecución en diferentes entornos y escenarios.
4. Explorar la integración de herramientas de inteligencia artificial en el proceso de automatización de pruebas, con énfasis en las capacidades de auto reparación de scripts, para reducir el esfuerzo de mantenimiento y aumentar la robustez de las pruebas ante cambios en la interfaz de usuario.
5. Demostrar la efectividad de las pruebas automatizadas implementadas para el módulo "Carga-Descarga" mediante la detección temprana de defectos y la validación continua de nuevas funcionalidades, contribuyendo así a la mejora de la calidad del producto final.

Para el estado del arte:

Ejecutar conjuntos extensos de pruebas en periodos reducidos, ampliando significativamente la cobertura del producto y detectando problemas que podrían pasar desapercibidos en pruebas manuales limitadas[7].

Establecer mecanismos de verificación continua que aseguran que funcionalidades previamente validadas no se deterioran con la introducción de nuevas características, protegiendo así la integridad del sistema a lo largo de su evolución[8].

Eliminar la variabilidad inherente a la ejecución humana, garantizando que cada caso de prueba se ejecute exactamente de la misma manera en cada iteración, independientemente de factores como el cansancio o la distracción[7].

Optimizar recursos a largo plazo, pues aunque la inversión inicial en automatización puede ser considerable, el retorno se materializa a través de ciclos de desarrollo más cortos, mayor calidad del producto y reducción de costos de mantenimiento[3, 9].

Referencias principales:

1. Pressman, R.S., *Software engineering: a practitioner's approach*. 2005: Palgrave macmillan.

2. Myers, G.J., C. Sandler, and T. Badgett, *The art of software testing*. 2011: John Wiley & Sons.

3. Stapp, L., A. Roman, and M. Pilaeten, *ISTQB® Certified Tester Foundation Level: A Self-Study Guide Syllabus v4.0*. 2023: Springer Nature Switzerland.

4. Garousi, V. and M.V. Mäntylä, *When and what to automate in software testing? A multi-vocal literature review.* Information and Software Technology, 2016. **76**: p. 92-117.

5. Rafi, D.M., et al. *Benefits and limitations of automated software testing: Systematic literature review and practitioner survey*. in *2012 7th international workshop on automation of software test (AST)*. 2012. IEEE.

6. Leotta, M., et al., *Approaches and tools for automated end-to-end web testing*, in *Advances in Computers*. 2016, Elsevier. p. 193-237.

7. Pargaonkar, S., *A comprehensive research analysis of software development life cycle (SDLC) agile & waterfall model advantages, disadvantages, and application suitability in software quality engineering.* International Journal of Scientific and Research Publications (IJSRP), 2023. **13**(08): p. 345-358.

8. Carter, D.O., *Advancing Software Quality: A Comprehensive Exploration of Code Quality Metrics, Static Analysis Tools, and Best Practices.* Journal of Science & Technology, 2024. **5**(1): p. 69-81.

9. Black, R. and J.L. Mitchell, *Advanced Software Testing - Vol. 3: Guide to the ISTQB Advanced Certification as an Advanced Technical Test Analyst*. 2011: Rocky Nook.