Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Computación

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

System for the design of Functional Test Cases from Use Cases for the Computer Science course at the National University of Loja.

Trabajo de Integración Curricular previa a la obtención del título de Ingeniero en Ciencias de la Computación

AUTOR:

Juan Francisco Castillo Estrella.

DIRECTOR:

Ing. Wilman Patricio Chamba Zaragocín.

CODIRECTOR:

Ing. Pablo Fernando Ordoñez Ordoñez, Mg. Sc.

Loja – Ecuador

2025

Certificación de directores

Ing. Wilman Patricio Chamba Zaragocín.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Ing. Pablo Fernando Ordoñez Ordoñez, Mg. Sc.
CODIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

C E R T I F I C O:

Que hemos revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominado: **Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja**, previo a la obtención del título de Ingeniera en Ciencias de la Computación, de la autoría del estudiante **Juan Francisco Castillo Estrella**, con cédula de identidad **1105822447**, una vez que se determina que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Firmado electrónicamente por:
WILMAN PATRICIO
CHAMBA ZARAGOCIN

Ing. Wilman Patricio Chamba Zaragocín.
DIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Firmado electrónicamente por:
PABLO FERNANDO
ORDONEZ ORDONEZ

Ing. Pablo Fernando Ordoñez Ordoñez, Mg. Sc.
CODIRECTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Autoría

Yo, **Juan Francisco Castillo Estrella**, declaro ser autor del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular, en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma: 

Cédula de identidad: 1105822447

Fecha: 14-03-2025

Correo electrónico: juan.f.castillo.e@unl.edu.ec

Teléfono: 09817466394

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular

Yo, **Juan Francisco Castillo Estrella**, declaro ser el autor del Trabajo de Integración Curricular denominado: **Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja**, como requisito para optar por el título de **Ingeniero en Ciencias de la Computación**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los 14 días del mes de marzo de dos mil veinticinco.

Firma: 

Autor: Juan Francisco Castillo Estrella
Cédula de identidad: 1105822447
Dirección: La Tebaida, Loja, Ecuador.
Correo electrónico: juan.f.castillo.e@unl.edu.ec
Teléfono: 0987466394

DATOS COMPLEMENTARIOS:
Director del Trabajo de Integración Curricular: Ing. Wilman Chamba.
Codirector del Trabajo de Integración Curricular: Ing. Pablo Fernando Ordoñez Ordoñez, Mg. Sc.
.

Dedicatoria

A mi madre, mi abuelo, mi abuelita, mis hermanos, mis tías y mis primos, quienes han sido un pilar fundamental en cada etapa de este proceso. Su amor, paciencia y apoyo incondicional me han dado la fortaleza para seguir adelante, incluso en los momentos más desafiantes.

A mi madre, por ser mi guía, mi ejemplo de perseverancia y el motor que impulsa mis sueños. A mi abuelo, por su sabiduría, sus enseñanzas y por recordarme siempre el valor de la humildad y la dedicación. A mis hermanos, por ser mi compañía en este viaje, por su aliento y su confianza en mí. A mis tías y primos, por estar presentes con palabras de ánimo, consejos y gestos que han hecho la diferencia en mi vida.

Cada uno de ustedes ha dejado una huella imborrable en este logro, y por ello, les estaré eternamente agradecido.

Juan Francisco Castillo Estrella

Agradecimiento

Quiero expresar mi agradecimiento, primeramente, a mi abuelita que desde el cielo siempre estuvo presente en cada paso de este largo camino, fue mi motivación para demostrar que puedo conseguir mis objetivos, siento que estaría orgullosa de lo que soy.

Agradezco a mi madre Paulina y a mi abuelito Marco que ha sido mi padre, que ante todo siempre estuvieron presentes, mis hermanos, tías y primos que, con su cariño y amor, fueron mi sustento diario para no rendirme jamás. A mis amigos, los que hice en el camino y los que han estado antes de la etapa universitaria, todos han hecho que esta fase de mi vida sea mejor y agradable en todo sentido.

Mi Abuelito que siempre estuvo a mi lado en toda la etapa universitaria y apoyarme en todas las decisiones tomadas y ser un amigo en este proceso, y a mi novia que siempre estuvo conmigo en muchas épocas buenas y malas de la carrera universitaria, por ser parte de mis alegrías y tristezas, siempre estaré agradecido.

A mi tutor de trabajo de integración curricular, el Ingeniero Wilman Chamba que constantemente veló por colaborar en mis avances de trabajo y todos los colegas junto a él que consideraron alentar cuando era necesario, su esfuerzo como profesional es admirable y es un gran ejemplo a seguir.

Al Ingeniero Francisco Álvarez que fue el promotor del tema trabajado, fue quien impulsó las ideas iniciales y aportó con las primeras fases de este desarrollo, su apoyo fue un gran aporte para cumplir con los objetivos propuestos.

Juan Francisco Castillo Estrella

Índice de Contenidos

Índice de Tablas.

Índice de Figuras.

Índice de Anexos.

1. Título

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la Carrera de Computación de la Universidad Nacional de Loja.

Software for the design of Functional Test Cases from Use Cases for the Computer Science degree at the National University of Loja.

2. Resumen

Actualmente, el proceso de diseño de casos de prueba funcionales a partir de casos de uso en el desarrollo de software demanda una considerable inversión de tiempo y esfuerzo, especialmente cuando se realiza de manera manual o semiautomática, pues ante la carencia de herramientas especializadas que automaticen este proceso afecta la eficiencia en el desarrollo de pruebas de software; por tal motivo, el presente trabajo de titulación tiene como objetivo desarrollar una herramienta basada en el uso de inteligencia artificial generativa para el diseño automatizado de casos de prueba funcionales a partir de casos de uso, dirigida principalmente a los estudiantes de la Carrera de Computación de la Universidad Nacional de Loja y profesionales dedicados al desarrollo, por tanto, para cumplir con este objetivo se plantearon dos fases principales. En la primera fase, se desarrolló una aplicación web utilizando un stack tecnológico compuesto por: Next.js para el frontend, Nest.js para el backend, PostgreSQL como base de datos, e integrando el modelo de “Commad-r” que consume el API de Cohere. Esta implementación incluyó características de seguridad mediante Next Auth para la autenticación y Nodemailer para la gestión de correos electrónicos, todo guiado por la metodología RAD (Desarrollo Rápido de Aplicaciones). En la segunda fase, se evaluó la percepción de utilidad de la herramienta entre los estudiantes de la carrera de Computación, utilizando lineamientos del Modelo de Aceptación Tecnológica (TAM), donde los resultados de la evaluación mostraron un nivel alto en la escala de Likert para la percepción de utilidad, indicando una aceptación significativa de la herramienta por parte de los estudiantes. Estos hallazgos sugieren que la implementación de tecnologías de inteligencia artificial en el proceso de diseño de casos de prueba es percibida como una solución valiosa y útil en el contexto académico de la ingeniería de software.

Palabras Clave: *Casos de Uso, Casos de Prueba, Inteligencia Artificial, Desarrollo de Software.*

Abstract

Currently, the process of designing functional test cases from use cases in software development requires a considerable investment of time and effort, especially when done manually or semi-automatically. The lack of specialized tools to automate this process affects the efficiency of software testing. For this reason, this thesis aims to develop a tool based on generative artificial intelligence for the automated design of functional test cases from use cases. The tool is primarily intended for students of the Computer Science program at the National University of Loja and professionals dedicated to development. To achieve this objective, two main phases were proposed. In the first phase, a web application was developed using a technology stack consisting of Next.js for the frontend, Nest.js for the backend, PostgreSQL as the database, and the integration of the “Commad-r” model, which consumes the Cohere API. This implementation included security features through Next Auth for authentication and Nodemailer for email management, all guided by the Rapid Application Development (RAD) methodology. In the second phase, the perception of the tool’s usefulness among Computer Science students was evaluated using the guidelines of the Technology Acceptance Model (TAM). The evaluation results showed a high level on the Likert scale for perceived usefulness, indicating significant acceptance of the tool among students. These findings suggest that implementing artificial intelligence technologies in the test case design process is perceived as a valuable and useful solution in the academic context of software engineering.

Key Words: *Use Cases, Test Cases, Artificial Intelligence, Software Development.*

3. Introducción

El proceso de pruebas de software es una fase relevante en el desarrollo de aplicaciones, siendo fundamental para garantizar la calidad y confiabilidad del producto final, dentro de este proceso, el diseño de casos de prueba funcionales representa una tarea especialmente desafiante [1], dado que, requiere una comprensión profunda de los requisitos del sistema y una considerable inversión de tiempo y recursos; en particular, la derivación de casos de prueba a partir de casos de uso, aunque es una práctica común en la industria, puede resultar compleja y propensa a errores cuando se realiza de manera manual [2].

En este contexto, la integración de tecnologías de inteligencia artificial (IA) emerge como una solución prometedora para automatizar y optimizar el proceso de diseño de casos de prueba, pues la IA puede potencialmente reducir el tiempo dedicado a esta tarea mientras mantiene o incluso mejora la calidad de los casos de prueba generados [3], no obstante, para que estas soluciones tecnológicas sean efectivas en el ámbito académico, es necesario evaluar su percepción de utilidad entre los usuarios finales, en este caso, los estudiantes de la carrera de Computación.

El presente trabajo de investigación se centra en desarrollar y evaluar una herramienta de software que utiliza inteligencia artificial para el diseño automatizado de casos de prueba funcionales a partir de casos de uso, específicamente dirigida a los estudiantes de la carrera de Computación de la Universidad Nacional de Loja, por ello, la importancia de este estudio radica en la necesidad de facilitar y optimizar el proceso de diseño de casos de prueba en el contexto educativo, proporcionando a los estudiantes una herramienta que pueda servir como apoyo en su formación en computación y calidad de software.

Para abordar esta necesidad, se plantea la siguiente pregunta de investigación: **¿Cuál es el nivel de percepción de utilidad de una herramienta basada en inteligencia artificial para el diseño de casos de prueba funcionales a partir de casos de uso, entre los estudiantes de la carrera de Computación de la Universidad Nacional de Loja?** Para dar respuesta a esta interrogante, la investigación se estructura en dos objetivos específicos: primero, desarrollar una herramienta de software que utilice la API de una IA para diseñar casos de prueba funcionales a partir de casos de uso, empleando la metodología RAD (Desarrollo Rápido de Aplicaciones); y segundo, evaluar la percepción de utilidad de la herramienta desarrollada entre los estudiantes, utilizando como base ciertos lineamientos del modelo TAM (Modelo de Aceptación Tecnológica).

El alcance del proyecto se centra en el desarrollo e implementación de la herramienta de software, así como en la evaluación de su percepción de utilidad entre los estudiantes de

la carrera de Computación; aquí se usa la metodología RAD para el desarrollo del software debido a su enfoque ágil y su capacidad para producir resultados rápidos mientras se mantiene la calidad del producto.

El presente trabajo está organizado de la siguiente manera: en el Marco Teórico se presentan los fundamentos conceptuales sobre pruebas de software, casos de uso, inteligencia artificial y el modelo TAM; la sección de Metodología detalla el proceso de desarrollo RAD y los métodos de evaluación empleados [4]; en Resultados se exponen los hallazgos obtenidos tanto en el desarrollo de la herramienta como en su evaluación; la Discusión analiza las implicaciones de los resultados obtenidos; finalmente, las secciones de Conclusiones y Recomendaciones presentan los principales hallazgos y sugerencias para trabajos futuros.

4. Marco teórico

4.1. Antecedentes.

Los casos de prueba son componentes básicos del proceso de desarrollo de software, los cuales son utilizados para proporcionar un entorno que permita comprobar la conformidad, la calidad y la funcionalidad del sistema; según Sommerville (2011), estos validadores deben diseñarse considerando diversos escenarios para garantizar una amplia cobertura y cumplir con los requisitos del sistema [2], este proceso de diseño de casos de prueba suele ser un proceso manual, donde, a partir de los requisitos necesarios, los profesionales desarrollan casos de prueba, lo que resulta en varias ocasiones en la elaboración de tareas monótonas o difíciles, esto debido a la constante evolución de las tecnologías, por lo que, llega a ser un proceso de alto impacto, pues, es casi imposible atender a todos los aspectos al mismo tiempo.

El caso de uso es un artefacto valioso para crear casos de prueba, Cockburn (2000) destaca la interacción entre los usuarios y los sistemas, la inclusión de actores, de entradas y salidas, y flujos tanto normales como alternos [5], esta información es útil, dado que, proporciona una visión externa de la aplicación a manera de “caja negra”, lo que permite saber qué funcionalidades pueden probarse, haciendo uso de sus atributos para plantear situaciones que comprueben el correcto funcionamiento de un sistema, en este sentido la IA es una buena ayuda, pues con su evolución, esta se ha formado como herramienta eficaz en progresión para realizar tareas difíciles, ya que, estos modelos son capaces de analizar patrones y aprender de experiencias previas [6], todo con el objetivo de reducir tareas repetitivas y deducciones simples, optimizando el tiempo y recursos en el proceso de diseño.

Los estudiantes y profesionales de computación a menudo enfrentan dificultades al diseñar casos de prueba, esto se evidencia en la entrevista realizada al Ing. Francisco Álvarez de la Carrera de Computación de la Universidad Nacional de Loja (**véase Anexo 1**), estos desafíos incluyen complejidad en la comprensión, diseño de precondiciones y postcondiciones, dificultades para lograr una cobertura completa de requisitos, lo que deriva a omitir errores significativos en proyectos estudiantiles y de integración curricular [1], por tanto, esta progresiva complicación de los proyectos de software demanda un enfoque más eficiente para el diseño de casos de prueba.

Un software que consuma IA para esta tarea puede simplificar el proceso de diseño y minimizar pasos para obtener resultados, así se permite a los estudiantes concentrarse en la ejecución de otras actividades pertenecientes al área de pruebas; como sugiere Harman (2011), esta implementación no solo optimizaría recursos, sino que también fortalecería la formación de los estudiantes, permitiéndoles explorar en profundidad los aspectos teóricos y prácticos de los casos de prueba en la calidad del software [7], por lo que, dicho enfoque

moderno tiene el potencial de mejorar significativamente tanto la eficiencia en el desarrollo de software como la calidad de la educación en el campo de la Computación.

4.2. Fundamentación Teórica

El desarrollo de software requiere seguir un lineamiento para evitar ambigüedades en su avance, adjunto a esto se puede entender que la rapidez en la entrega de un producto, satisface más pronto la necesidad, por ello, se opta por una metodología que aporte a ese ideal de las entregas rápidas y con el apoyo constante de los interesados.

4.2.1. Metodología para Desarrollo de Software

4.2.1.1. Metodología RAD (Rapid Application Development)

La metodología RAD (acrónimo en inglés de Rapid Application Development), metodología ágil introducida por James Martin en 1991 como salida de metodologías tradicionales, es aquella que propone el desarrollo rápido y las entregas de calidad a corto plazo [8], en ella, se suelen ejecutar ciclos cortos e iterativos y el sistema se basa en prototipos funcionales obtenidos desde etapas tempranas del desarrollo [9]; y para facilitar la adaptación, este se encuentra hecho para trabajar en equipos pequeños, por tanto, se considera una elección acertada para el presente trabajo, puesto que, al estar enfocada en las entregas rápidas es la herramienta ideal debido al tiempo de doce semanas enfocadas en el desarrollo del software, adoptando su guía de contacto constante con el cliente y entregables de calidad, por lo que, en este sistema se encuentra presente el cliente en cada paso que se ejecuta [10]. Conocida la causa de su elección, se menciona la estructura del proceso compuesta generalmente de cuatro fases, que determinan el flujo ideal para entregar un producto en el menor tiempo. En la **Tabla 1**, se muestran las fases de RAD, expuestas por Campaña [4]:

Tabla 1. Metodología RAD *[4]*

Fase	Descripción	Actividades Principales	Entregables
1. Planificación de Requisitos	Identificación de los requisitos esenciales del sistema para comprender las necesidades del cliente y los objetivos del proyecto.	- Reuniones con partes interesadas - Recopilación y análisis de requisitos - Documentación inicial	- Documento de requisitos - Modelo conceptual
2. Diseño de Usuario	Creación de prototipos iterativos y diseño de interfaces para obtener retroalimentación constante del cliente.	- Creación de prototipos rápidos - Diseño de interfaces de usuario (GUI)	- Prototipos interactivos - Bocetos de interfaz - Ajustes según retroalimentación

Fase	Descripción	Actividades Principales	Entregables
		- Validación con usuarios finales	
3. Construcción	Fase de desarrollo donde se implementa el sistema basado en los prototipos aprobados	- Desarrollo de módulos funcionales - Pruebas unitarias e integración - Revisión y ajustes iterativos	- Código funcional - Versión Beta del sistema - Resultados de pruebas - Documentación técnica
4. Transición	Despliegue del sistema en producción, capacitación de usuarios y estabilización del entorno de operación.	- Implementación en entorno de producción - Migración de datos - Capacitación a usuarios - Resolución de problemas finales	- Sistema en producción - Manual de usuario - Documentación final y plan de mantenimiento

La metodología RAD se distingue por su énfasis en la creación rápida de prototipos y la activa participación del usuario, aspectos que son clave para el desarrollo de un sistema que demanda un alto grado de precisión y adaptabilidad, por lo que, resulta ventajoso por un menor costo, una mayor satisfacción del usuario y por la adaptabilidad a cambios, que va a ser primordial, debido al contacto tan cercano con el cliente, pues, esto lo debe hacer susceptible a modificaciones.

La naturaleza iterativa de RAD permite una retroalimentación continua, lo cual es de gran importancia en un proyecto que utiliza tecnologías que consumen inteligencia artificial, así como para el uso de tecnologías relativamente nuevas como son los frameworks de Nest.js y Next.js, particularmente focalizadas a Typescript para una estructura organizada, y dicha retroalimentación también se apoya en la comunidad académica de la Carrera de Computación de la Universidad Nacional de Loja.

4.2.2. Modelo para la Aceptación Tecnológica.

4.2.2.1. Metodología de Aceptación Tecnológica (TAM).

Para evaluar la percepción de utilidad del sistema desarrollado para el diseño de casos de prueba, se emplean lineamientos seleccionados del Modelo de Aceptación Tecnológica (TAM) en combinación con la escala de Likert, lo que permite una evaluación más flexible y adaptada a las limitaciones de tamaño de muestra del proyecto.

El Modelo de Aceptación Tecnológica (TAM) es ampliamente reconocido en la investigación sobre sistemas de información y tecnología por su capacidad para analizar la percepción de los usuarios sobre nuevas tecnologías [11], por tanto, por el contexto, se centra

en la percepción de utilidad del TAM, adaptando sus lineamientos metodológicos al contexto específico del trabajo.

La percepción de utilidad se define como el grado en que un usuario cree que el uso de un sistema particular mejoraría su desempeño en una tarea [11], en este caso, se evaluará cómo los usuarios perciben que el sistema desarrollado mejora su capacidad para diseñar casos de prueba.

Para implementar esta evaluación focalizada, se adapta los siguientes lineamientos del TAM [11]:

- **Tamaño de muestra:** Considerando las limitaciones del proyecto, se trabaja con una muestra de 20 a 30 estudiantes con conocimientos en pruebas de software de la Carrera de Computación.
- **Diseño del cuestionario:** Se elabora un cuestionario conciso, enfocado exclusivamente en ítems que evalúen la percepción de utilidad del sistema para el diseño de casos de prueba.
- **Escala de medición:** Se utiliza la escala de Likert de 5 puntos, manteniendo la consistencia con la metodología TAM original, pero aplicada únicamente a las preguntas sobre utilidad percibida.
- **Recolección de datos:** La aplicación del cuestionario se realiza de manera controlada, asegurando que los participantes comprendan el enfoque específico en la utilidad del sistema.
- **Análisis de datos:** Se lleva a cabo un análisis estadístico descriptivo centrado en las respuestas relacionadas con la percepción de utilidad.
- **Interpretación de resultados:** La interpretación se centra en cómo los usuarios perciben la utilidad del sistema para mejorar su desempeño en el diseño de casos de prueba, sin extenderse a otros aspectos del TAM.

Esta aproximación adaptada permite obtener una evaluación significativa de la percepción de utilidad del sistema, alineada con los principios del TAM, pero ajustada a las necesidades y limitaciones específicas del proyecto, puesto que, al estar enfocado en la percepción de utilidad, se profundiza en este aspecto crítico, incluso con una muestra limitada.

La **Figura 1** muestra el modelo TAM con sus diferentes factores para explicar cómo los usuarios llegan a aceptar y usar una tecnología, aquí el modelo sugiere que varios de estos influyen en la decisión de cómo y cuándo los usuarios utilizarán una nueva tecnología,

con la utilidad percibida y la facilidad de uso percibida siendo los factores clave que afectan la aceptación del usuario.

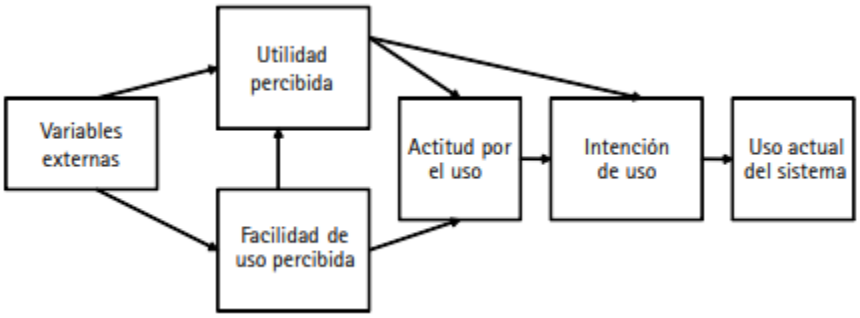


Figura 1: Modelo de aceptación tecnológica (TAM)

- **Percepción de Utilidad.**

La toma de captura del modelo TAM se efectúa para contrastar la Utilidad percibida o percepción de utilidad, pues este concepto es fundamental en la predicción de la aceptación y uso de nuevas tecnologías; esta se refiere al grado en que una persona cree que el uso de un sistema en particular mejoraría su desempeño en el trabajo o en una tarea específica [12]. El factor analizado resulta de alto impacto debido a:

- **Influencia directa en la intención de uso:** Como se muestra en el diagrama, la Utilidad percibida tiene un impacto directo en la Intención de uso, lo que sugiere que los usuarios son más propensos a adoptar una tecnología si creen que les proporcionará beneficios tangibles en sus actividades [13].
- **Efecto sobre la Actitud:** La Utilidad percibida también afecta la Actitud hacia el uso, lo que indirectamente influye en la Intención de uso, lo que implica que una percepción positiva de la utilidad puede mejorar la actitud general hacia la tecnología [14].
- **Interacción con la Facilidad de uso percibida:** Aunque son constructos separados, la Utilidad percibida y la Facilidad de uso percibida están interrelacionadas. Una tecnología que se percibe como fácil de usar puede, a su vez, ser percibida como más útil [15].
- **Mediador de variables externas:** La Utilidad percibida actúa como un mediador entre las Variables externas y la Intención de uso, lo que sugiere que factores como las características del sistema, el contexto organizacional o las diferencias individuales pueden influir en la percepción de utilidad [16].

En el contexto del trabajo, el análisis de la Utilidad percibida puede proporcionar perspectivas valiosas sobre cómo los usuarios evalúan el potencial del sistema para mejorar

su rendimiento o alcanzar sus objetivos, lo que contrasta al estudiar la implementación de una nueva tecnología o buscar mejoras en la adopción de un sistema existente.

4.2.2.2. Likert

La escala de Likert se emplea en este proyecto para medir la percepción de utilidad del sistema desarrollado para el diseño de casos de prueba, esto debido a varias razones [17], inicialmente la simplicidad y claridad, pues, la escala de Likert es fácil de entender y utilizar, lo que facilita la recopilación de datos precisos de los usuarios, también, porque permite a los encuestados expresar su grado de acuerdo o desacuerdo en un rango que va desde "totalmente en desacuerdo" hasta "totalmente de acuerdo", ofreciendo una visión detallada de sus percepciones, esto hace que pueda adaptarse a diferentes contextos y temáticas, lo que la hace ideal para evaluar aspectos específicos del sistema, como la percepción de utilidad y una vez recolectados los datos mediante esta escala, se vuelve fácil de cuantificar y analizar, dando paso a identificación de patrones y tendencias en las respuestas de manera eficiente [17].

La escala de Likert permite redactar afirmaciones específicas relacionadas con la percepción de utilidad del sistema para el diseño de casos de prueba, pues en la construcción del cuestionario se integran afirmaciones donde los participantes pueden calificar cada ítem utilizando la escala de Likert de 5 puntos [18], distribuyendo este artefacto a una muestra representativa de estudiantes que han utilizado el sistema, para luego analizar las respuestas y evaluar la percepción de utilidad del sistema, utilizando métodos estadísticos descriptivos.

En la **Figura 2**, se aprecia en una campana de distribución normal tipo gaussiana, y con cinco escalas. Para la primera escala (*), se tienen las categorías de Likert, donde los valores |0 a 1| son de totalmente acuerdo; de |1 a 2| es de acuerdo, |2 a 3| Ni de acuerdo ni desacuerdo, |3 a 4| en desacuerdo y |4 a 5| totalmente desacuerdo [18].

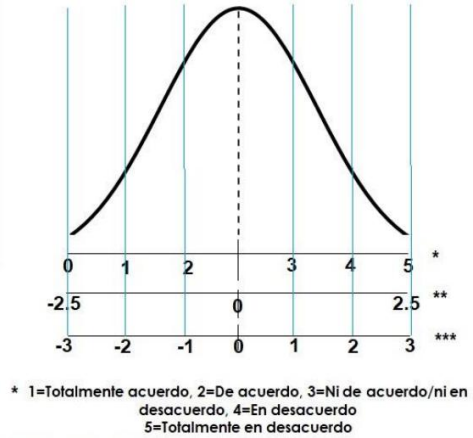


Figura 2: Distribución gaussiana de la escala de Likert.

La muestra para este estudio se selecciona de una población compuesta por estudiantes del 8vo y 9no ciclo de la carrera de Computación de la UNL, aquí se emplea un muestreo aleatorio simple para garantizar que cada estudiante tenga la misma probabilidad de ser seleccionado, asegurando así la representatividad de la muestra [19]; el tamaño de la muestra se determina considerando un nivel de confianza del 95% y un margen de error del 5%, lo que resulta en una muestra de entre 20 y 30 estudiantes, suficiente para obtener datos estadísticamente significativos en el contexto de este estudio [20].

Esta metodología permite una evaluación rigurosa y objetiva de la percepción de utilidad del sistema desarrollado, proporcionando perspectivas valiosas para su mejora y potencial adopción a largo plazo.

4.2.4. Tecnologías

4.2.4.1. TypeScript

TypeScript se emplea en este proyecto como el lenguaje de programación principal para el desarrollo del sistema de diseño de casos de prueba, dado que, se fundamenta en varias características clave que aportan significativamente a la calidad y eficiencia del desarrollo [21], primeramente porque introduce un sistema de tipos estáticos que permite la detección temprana de errores, mejorando la robustez del código y facilitando el mantenimiento a largo plazo; así mismo, este lenguaje posee ayuda en la autocompletación inteligente, la navegación de código y las herramientas de refactorización, optimizan el flujo de trabajo de los desarrolladores, lo que otorga una integración sin interferencias de librerías y frameworks existentes, brindando escalabilidad en la organización del código [22].

Este enfoque metodológico garantiza un desarrollo eficiente y mantenible del sistema de diseño de casos de prueba, aprovechando al máximo las ventajas que ofrece TypeScript en términos de calidad de código, productividad y escalabilidad.

4.2.4.2. NestJS

NestJS es un framework de Node.js lanzado en 2017 por Kamil Mysliwiec; desarrollado para construir aplicaciones del lado del servidor (backend) [23], utilizado en este proyecto para el sistema de diseño de casos de prueba, debido a que promueve una estructura de código organizada y modular, facilitando el mantenimiento y la escalabilidad, dado que, este está construido con y para TypeScript, aprovechando todas sus ventajas, y así, implementar un sistema robusto de inyección de dependencias que mejora la testabilidad y la modularidad del código, pues utiliza decoradores para simplificar la configuración y mejorar la legibilidad del código [24].

Dichas características aportan en gran medida al desarrollo del sistema, puesto que, proporcionan una base sólida y estructurada para la creación de APIs robustas [25], por tanto, para implementarlo en el proyecto se pueden seguir estos pasos:

Inicialmente, se crea un nuevo proyecto NestJS utilizando el CLI, para luego estructurar los componentes del sistema en módulos lógicos; posteriormente, se desarrollan los end-points de la API y la lógica de negocio, luego, se añaden capas adicionales para el manejo de solicitudes y validación de datos. Después, se configura la conexión y se definen los modelos de datos, y se prueba unitariamente como de forma íntegra [25]. Este enfoque proporciona una base sólida para el backend del sistema, que facilita su desarrollo, mantenimiento y escalabilidad futura.

4.2.4.3. Nodemailer

Nodemailer es un módulo para aplicaciones Node.js diseñado específicamente para permitir el envío fácil de correos electrónicos [26], este se implementó en el proyecto para el sistema de notificaciones por correo electrónico, debido a su robusta capacidad de manejo de correos electrónicos SMTP, soporte para contenido HTML y archivos adjuntos, además de su amplia compatibilidad con diversos servicios de correo electrónico [27].

Las características principales de Nodemailer benefician significativamente al proyecto, dado que proporciona una interfaz unificada para diferentes proveedores de servicios de correo electrónico, manejo de plantillas HTML personalizables, y soporte para TLS/STARTTLS para comunicaciones seguras [28]. Por consiguiente, para implementar Nodemailer en el proyecto se pueden seguir estos pasos:

Primeramente, se instala el módulo Nodemailer mediante npm, seguido de la configuración del transporter con las credenciales SMTP correspondientes; posteriormente, se crean las plantillas HTML para los correos electrónicos y se desarrolla la lógica de envío, luego, se implementan las funciones de manejo de errores y reintentos, y finalmente, se realizan pruebas de integración para verificar el correcto funcionamiento del sistema de correos [29]. Esta metodología asegura una implementación confiable del sistema de notificaciones por correo electrónico, garantizando la entrega efectiva de mensajes y una experiencia de usuario mejorada.

4.2.4.4. NextJS

Next.js es un potente framework de React desarrollado por Vercel, diseñado para optimizar la creación de aplicaciones web modernas, el cual se destaca por su capacidad de renderización híbrida, permitiendo esta tanto del lado del servidor (SSR) como del lado del cliente (CSR), lo que resulta en aplicaciones con mejor rendimiento y experiencia de usuario

[30]. Este framework se integra con TypeScript, alineándose así con las prácticas de desarrollo robustas utilizadas en el backend y contribuyendo a una estructura de proyecto más coherente y mantenible, esto no solo mejora la eficiencia en el desarrollo, sino que también optimiza aspectos importantes como el rendimiento, la navegación y la indexación en motores de búsqueda [30].

La implementación de Next.js en un proyecto sigue un flujo de trabajo estructurado y eficiente; inicialmente, se crea un nuevo proyecto utilizando el CLI de Next.js, que configura automáticamente una estructura básica y las dependencias necesarias, luego, se desarrollan las vistas y componentes de la aplicación, aprovechando la sintaxis familiar de React y las características adicionales proporcionadas por el framework, como el enrutamiento basado en el sistema de archivos [31]. Un paso relevante es la integración con el backend, que implica configurar las llamadas a la API y gestionar el estado de la aplicación, posiblemente utilizando herramientas como SWR o React Query para un manejo eficiente de los datos, al final, se organiza la estructura de carpetas del proyecto para maximizar la modularidad y facilitar la navegación entre componentes, y así, seguir las mejores prácticas recomendadas por la comunidad de Next.js [32].

4.2.4.5. Next-Auth

NextAuth.js es un sistema de autenticación y autorización de código abierto mantenido por Vercel, diseñado específicamente para aplicaciones Next.js [33], este framework se implementó en el proyecto para gestionar la autenticación de usuarios debido a su capacidad de integración con múltiples proveedores de autenticación, su robusta gestión de sesiones, y su implementación simplificada de flujos de autenticación seguros [34].

Las características fundamentales de NextAuth.js aportan significativamente, puesto que ofrece soporte integrado para JWT (JSON Web Tokens), manejo automático de cookies de sesión, y compatibilidad con diversos proveedores de autenticación como Google, GitHub y servicios personalizados de credenciales [35]; por tanto, para implementar NextAuth.js en el proyecto se pueden seguir estos pasos:

Primero, se instala NextAuth.js mediante **npm** y se configura el archivo de configuración principal; luego, se implementan los proveedores de autenticación deseados y se configuran las callbacks necesarias; se desarrollan las páginas de inicio de sesión y registro, se implementa la lógica de protección de rutas, y finalmente, se realizan pruebas de integración para verificar los flujos de autenticación [36].

4.2.4.6. Postgres

PostgreSQL es un sistema de gestión de bases de datos relacional que ha tomado relevancia en los últimos años, debido a la cantidad de características que ofrece, tales como [37]:

- Soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).
- Capacidad para manejar concurrencia de manera eficiente.
- Extensibilidad a través de tipos de datos personalizados y funciones definidas por el usuario.
- Soporte para datos JSON y otras estructuras de datos complejas.

Además, esta base de datos se destaca por su capacidad para manejar cargas de trabajo complejas y su adherencia a los estándares SQL, lo que lo hace ideal para aplicaciones que requieren integridad de datos y rendimiento consistente, por tanto, se utiliza en este proyecto como el sistema de gestión de bases de datos relacional para almacenar y gestionar los datos del sistema de diseño de casos de prueba [38].

Las características mencionadas garantizan un almacenamiento eficiente y flexible de los casos de prueba y datos relacionados, por ello, para una mayor comprensión de cómo funciona se muestra el proceso de instalación.

4.2.4.7. Prisma.

Para simplificar el proceso de consultas y conexión con bases de datos, yacen los ORM, que son herramientas que facilitan la interacción entre las aplicaciones y las bases de datos relacionales y no relacionales, estos, simplifican pasos repetitivos o extensos para obtener comunicación con una base de datos, por tanto, de acuerdo al contexto del trabajo, existen tareas que ya se hacen comúnmente, como consultas de credenciales, obtener y guardar información, etc., entonces, se opta por un ORM (Object-Relational Mapping), simple y libre de uso como lo es Prisma, este destaca en la este tiempo por su manera de integrarse con aplicaciones modernas [39].

La razón principal para la elección es por su enfoque en esquemas, que permite describir la estructura de manera concisa, pues se tipea los diferentes tipos de datos u objetos con un lenguaje de modelado de datos que mejora visualmente la comprensión semántica de los mismos y sus relaciones [40]. Este ORM tiene la característica de generar automáticamente tipos Typescript desde sus propios esquemas, aumentando la productividad del desarrollador y minimizando la cantidad de errores en ejecución [41].

La arquitectura de prisma ofrece ventajas sobre las consultas tradicionales, en la **Figura 3**, se aprecia una visión clara del flujo de datos y las interacciones entre el cliente

Prisma, el motor de consultas y la base de datos, dicha arquitectura demuestra cómo Prisma abstrae las complejidades de las interacciones directas con la base de datos, proporcionando una interfaz más intuitiva y segura para los desarrolladores, aquí, la capa del motor de consultas actúa como un intermediario eficiente, optimizando las conexiones y transformando las operaciones de alto nivel en consultas SQL eficientes [42].

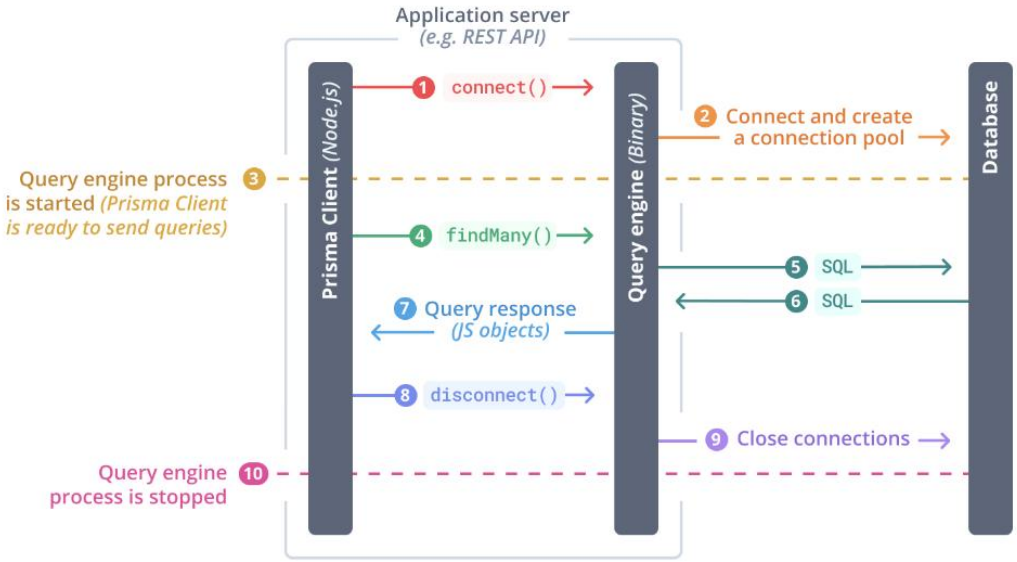


Figura 3: Funcionamiento de Prisma como ORM *[42]*

La **Figura 3** contrasta varios beneficios en la aplicación de este ORM, tales como [43]:

- Abstracción eficiente de la complejidad de la base de datos
- Manejo optimizado de conexiones a través de un pool
- Transformación transparente entre objetos JavaScript y registros de base de datos
- Ciclo de vida claro y gestionado de las operaciones de base de datos

Estas ventajas refuerzan la aplicación de un ORM para simplificar las consultas a una base de datos en este tipo de proyectos, dado que, simplifica el trabajo y acelera el desarrollo.

4.2.4.8. API Cohere.

La API de Cohere se implementa en este proyecto como el motor de inteligencia artificial central para asistir en el diseño de casos de prueba, puesto que, su avanzado procesamiento de lenguaje natural la convierte en una herramienta ideal para interpretar requisitos y generar casos de prueba detallados y relevantes [44]. Esta versatilidad permite una integración en múltiples componentes del sistema, facilitando el procesamiento y generación de contenido en diversos formatos, lo cual es de alto impacto para manejar la variedad de entradas y salidas inherentes al proceso de diseño de pruebas [45]; así mismo, la capacidad de Cohere para gestionar eficientemente cargas de trabajo variables, se alinea

perfectamente con las demandas cambiantes del proyecto, permitiendo una respuesta ágil a picos de actividad en el diseño de casos de prueba [46], dichas características contribuyen significativamente a la automatización y mejora del proceso, elevando la calidad y eficiencia del diseño de casos de prueba.

Estas características aportan significativamente a la automatización y mejora del proceso de diseño de casos de prueba, no obstante, este conocimiento queda escueto ante su manipulación, por tanto, se requiere saber la interacción que tendrá el sistema con dicha API, por ello, en la figura 4, se aprecia un ejemplo de cómo API Cohere puede actuar como un intermediario inteligente entre los usuarios y los sistemas de información, mejorando significativamente la experiencia del usuario al proporcionar interfaces conversacionales naturales para acceder a datos y servicios complejos. Demuestra varios aspectos clave de la integración de Cohere en aplicaciones, como, su capacidad para entender y procesar consultas en lenguaje natural, así como, la habilidad para determinar qué funciones o servicios externos son necesarios para responder a una consulta; también, la integración fluida entre Cohere, la aplicación host y servicios externos, y, su idoneidad para generar respuestas coherentes en lenguaje natural basadas en datos estructurados.

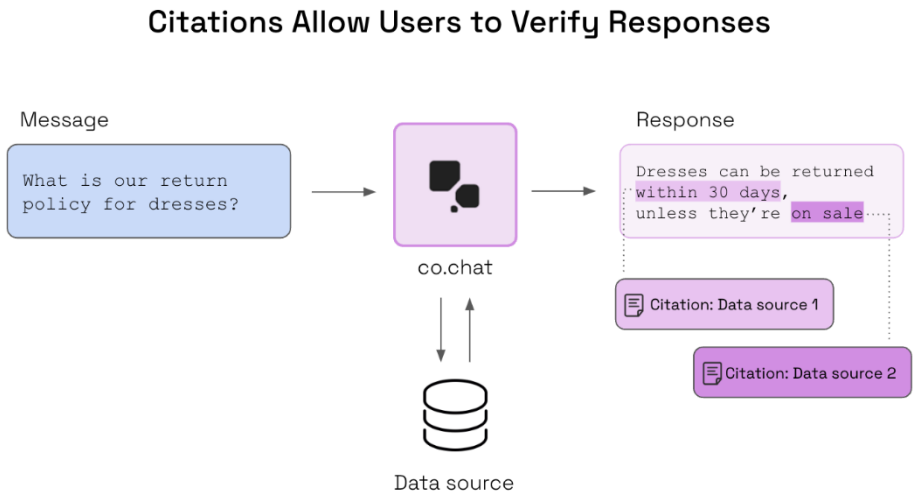


Figura 4: Cohere ToolKit intermediario para respuestas con IA *[44]*

4.2.4.9. Visual Studio Code

Para codificar todo el proyecto se requiere un editor de código o un IDE, donde este segundo refiere a un entorno de desarrollo integrado que proporciona servicios integrales para facilitar la programación a un desarrollador de software; por excelencia y simplicidad destaca Visual Studio Code, pues es eficaz y ligero, una aplicación multiplataforma que cuenta con varias extensiones que unifican todo el desarrollo en un solo entorno, mejorando la productividad y la ejecución de diferentes aplicativos en distintos lenguajes, ofreciendo incluso una personalización variada que se adecua al programador [47].

El IDE de Microsoft es un software gratuito que posee un compacto diseño que mejora el codeo, entre sus características se encuentra su autocompletado con IntelliSense, así como una depuración integrada y el control de versiones con Git [48].

4.2.4.10. GitHub

La incorporación de un repositorio alojado en la nube facilita el control de versiones y cambios en un proyecto de software, GitHub se encarga de ello y por eso lo utiliza en este proyecto, pues este es una plataforma de control de versiones y colaboración para el desarrollo de sistemas [49], la cual debido a si comportamiento y características es idóneo para controlar el código y sus cambios, entre los aspectos que se destacan de esta plataforma están los siguientes:

- **Control de versiones distribuido:** En base a Git, GitHub permite un seguimiento detallado y eficiente de los cambios en el código, esta capacidad facilita la colaboración entre desarrolladores, permitiendo trabajar en paralelo y fusionar cambios de manera controlada [50].
- **Gestión de proyectos integrada:** La plataforma ofrece herramientas como Issues y Projects, las cuales permiten organizar tareas y seguir el progreso del desarrollo, lo que gestionar eficazmente los flujos de trabajo del equipo, promoviendo una mejor organización y transparencia en el proceso de desarrollo [51].
- **Integración continua y despliegue continuo (CI/CD): GitHub** facilita la implementación de pipelines de CI/CD a través de GitHub Actions, permitiendo automatizar pruebas, compilaciones y despliegues, esto se vuelve crucial para mantener la calidad del código y agilizar el proceso de entrega de software [51].

La adopción de GitHub en este proyecto no solo facilita el control de versiones, sino que también promueve prácticas de desarrollo colaborativo, mejora la transparencia del proceso de desarrollo y proporciona herramientas para mantener un alto estándar de calidad en el código, por lo que, estas características hacen que GitHub sea una elección óptima para proyectos de software que requieren una gestión eficiente y una colaboración fluida entre desarrolladores.

4.2.4.11. DBeaver

DBeaver es una solución universal de gestión de bases de datos que se ha consolidado como una herramienta crítica en el ecosistema de desarrollo de software y administración de datos. Fundada en 2010, esta plataforma de código abierto ha evolucionado significativamente, transformándose de una herramienta básica a una solución integral para profesionales de bases de datos, desarrolladores y administradores de sistemas.

La herramienta destaca por su arquitectura multiplataforma, ofreciendo soporte robusto para diversos sistemas operativos como Windows, macOS y Linux, así mismo, su diseño modular permite la integración con una amplia gama de sistemas de gestión de bases de datos, tanto relacionales (SQL) como no relacionales (NoSQL), incluyendo MySQL, PostgreSQL, MongoDB, Redis, Azure SQL, etc.

Entre las funcionalidades de esta herramienta se menciona la administración de Bases de Datos, Capacidades de Consulta y Manipulación, Gestión de Datos Funcionalidades, Visualización y Modelado de Datos.

4.2.4.12. Postman

Postman se ha consolidado como una herramienta crítica en el ecosistema de desarrollo de software, proporcionando una solución integral para la gestión, prueba y documentación de interfaces de programación (APIs). Desde su concepción, Postman ha evolucionado significativamente, transformándose de una simple extensión de navegador a una plataforma completa para desarrolladores y equipos de tecnología [52].

La herramienta destaca por su arquitectura multiplataforma, ofreciendo soporte robusto para diversos sistemas operativos como Windows, macOS y Linux [53]. Su diseño modular permite una integración fluida en diversos entornos de desarrollo, facilitando la colaboración entre equipos de frontend y backend.

Postman ofrece un conjunto completo de herramientas que comprenden el testeo de colecciones de APIs, organización estructurada de servicios web, gestión de entornos de desarrollo (calidad, desarrollo, producción), generación automática de documentación, y colaboración en entorno cloud [53]. Adicionalmente, soporta múltiples protocolos y formatos como REST, SOAP, GraphQL, Webhooks, y diversos métodos de autenticación, cubriendo el ciclo de vida completo de las APIs desde su conceptualización y definición, pasando por su desarrollo y pruebas, hasta su monitoreo continuo y mantenimiento.

4.2.4.13. GitHub Actions

GitHub Actions representa una evolución significativa en las herramientas de integración continua y despliegue continuo (CI/CD), transformándose en una plataforma integral de automatización para equipos de desarrollo de software [54]. Su diseño innovador permite a los desarrolladores crear flujos de trabajo personalizados que automatizan procesos críticos en el ciclo de desarrollo de software.

La plataforma se distingue por su versatilidad multiplataforma, proporcionando máquinas virtuales para Linux, Windows y macOS, lo que facilita la ejecución de flujos de

trabajo en diversos entornos computacionales [55]. Esta flexibilidad se extiende más allá de las tradicionales prácticas de DevOps, permitiendo la automatización de procesos basados en eventos dentro de los repositorios, como la gestión automática de etiquetas en propuestas nuevas o la ejecución de pruebas en solicitudes de cambios [56].

4.2.4.14. Jest

Jest se ha consolidado como una plataforma de testing líder en el ecosistema de desarrollo JavaScript, especialmente en el entorno de desarrollo de React [57]. Mantenida por Facebook, esta herramienta ha revolucionado la forma en que los desarrolladores abordan las pruebas de software, ofreciendo un conjunto de características que simplifican y optimizan el proceso de testing [58].

La herramienta destaca por su enfoque en la eficiencia y la facilidad de uso, proporcionando una experiencia de testing intuitiva y poderosa. Su capacidad distintiva para realizar mock functions (funciones simuladas) representa un avance significativo en la metodología de pruebas de software, permitiendo a los desarrolladores simular comportamientos complejos de componentes que dependen de servicios externos, bases de datos y otras interacciones [59].

4.2.4.15. Cypress

Cypress es un framework moderno para la automatización de pruebas, especialmente diseñado para pruebas end-to-end (e2e), aunque también permite realizar pruebas unitarias y de integración [60]. A diferencia de otros frameworks tradicionales como Selenium, Cypress se construyó desde cero con una arquitectura completamente nueva, que le permite ejecutar comandos en el mismo ciclo de ejecución que la aplicación que se está probando. Esto elimina problemas comunes de sincronización entre el navegador y las pruebas, optimizando el flujo de trabajo para desarrolladores y testers.

Cypress es un framework moderno para pruebas automáticas que destaca por su arquitectura innovadora, diseñada desde cero para ejecutar pruebas en el mismo ciclo de ejecución que la aplicación. Esto lo diferencia de herramientas tradicionales como Selenium, al operar directamente dentro del navegador y sincronizarse con un proceso en Node.js que facilita la comunicación entre el frontend y el backend [61]. Su integración con librerías como Mocha, Chai y Sinon permite realizar pruebas unitarias, de integración y end-to-end (e2e) de manera eficiente. Además, ofrece acceso nativo a elementos del DOM, funciones, y respuestas del servidor, permitiendo manipular estados, simular errores y evitar acciones repetitivas como logins, todo mediante pruebas escritas en JavaScript [61].

4.3. Trabajos Relacionados

Al buscar trabajos que estén relacionados con el diseño de casos de prueba a partir de casos de uso, se ha podido encontrar algunos artículos que tratan este tema, no como se presenta en este proyecto, pero con temas que contribuyen a ello. La **Tabla 2** muestra los trabajos relacionados encontrados mientras se investigaba el tema del presente TIC, junto con un código que se utiliza para referirse a estos trabajos; los cuales van desde artículos que poseen un enfoque similar al tema hasta otros que sirven de modelo o base para el mismo.

Tabla 2: Trabajos relacionados

Código	Título	Descripción
TR001	Generación de casos de prueba a partir de casos de uso en las pruebas de software.	En el presente artículo se muestra un enfoque sistemático para la generación de casos de prueba a partir de estos casos de uso, lo que permite asegurar que el software cumpla con los requisitos especificados y funcione correctamente en diferentes escenarios; aquí los autores discuten cómo este método no solo mejora la cobertura de las pruebas, sino que también optimiza el proceso de validación y verificación del software, contribuyendo a la calidad del producto final. Además, se presentan ejemplos y metodologías que pueden ser aplicadas en la práctica para facilitar la implementación de este enfoque en proyectos de desarrollo de software [62].
TR002	Automatic Generation of Acceptance Test Cases from Use Case Specifications: an NLP-based Approach	El artículo presenta una temática que se centra en la generación automática de casos de prueba de aceptación a partir de especificaciones de casos de uso, aquí se utiliza técnicas de procesamiento de lenguaje natural (NLP) para extraer información estructurada de documentos escritos en lenguaje natural, lo que permite transformar requisitos en casos de prueba ejecutables. El objetivo principal es reducir el esfuerzo manual en la creación de pruebas y mejorar la cobertura de los requisitos, especialmente en sistemas críticos donde la validación es esencial. El artículo también discute el contexto industrial, los métodos utilizados y los resultados de validaciones empíricas realizadas en estudios de caso [63].
TR003	Desarrollo de un chatbot de apoyo a la terapia basada en reminiscencia	Este proyecto de titulación sirve como base para la ejecución e incorporación de la IA en el proyecto, pues esta muestra un chatbot que consume la API de Gemini para el apoyo a la terapia basada en reminiscencia, el cual no sirve exactamente por el problema, sino por la estructura y forma en como obtiene información de la IA y como idea el prompt ideal para enviar en la petición [64].
TR004	Towards Integrating Emerging AI Applications in SE Education	El presente artículo muestra un análisis de la integración de la IA en la ingeniería de software destacando tanto los desafíos que enfrentan los educadores como las oportunidades para adaptar conceptos didácticos, aquí, se busca categorizar los hallazgos en relación con diversas actividades y áreas de SE (Software Engineering), utilizando taxonomías como el Software Engineering Body of Knowledge

Código	Título	Descripción
		(SEBoK), así mismo, los resultados iniciales y sus aplicaciones potenciales se discuten en secciones específicas del documento [65].
TR005	AI based Software Testing	Este artículo presenta una revisión de técnicas de pruebas de software basadas en IA que aprovechan el aprendizaje automático, el procesamiento del lenguaje natural y otras técnicas de IA; ofrece una visión general de las pruebas de software basadas en IA, incluidas sus y limitaciones, y se analizan diversas técnicas y herramientas utilizadas en este campo. En el documento también destaca algunos de los esfuerzos actuales de investigación y pruebas de software basadas en IA, así como las direcciones y retos futuros [3].

En conclusión, de los trabajos relacionados, se menciona que los casos de uso han demostrado ser una base sólida y eficaz para la elaboración de casos de prueba en el desarrollo de software, no obstante, el proceso tradicional de derivar casos de prueba a partir de casos de uso a menudo requiere una inversión significativa de tiempo y recursos, lo que puede limitar la productividad de los equipos de desarrollo. En este contexto, la integración de la Inteligencia Artificial (IA) emerge como una solución prometedora, pues esta ofrece el potencial de simplificar y optimizar tareas repetitivas o rutinarias, permitiendo a los desarrolladores enfocarse en aspectos complejos y creativos del proceso de pruebas.

Entre las tecnologías de IA disponibles, el modelo de Cohere [66] se destaca como una opción particularmente atractiva para esta aplicación, dado que, ha demostrado capacidades superiores en tareas de razonamiento según los datos proporcionados por su sitio oficial, superioridad que se ve respaldada por los resultados observados en algunos trabajos relacionados. En adición, ofrece ventajas significativas en términos de implementación, puesto que, en comparación con otras APIs disponibles, su integración resulta directa y sencilla, lo que facilita su adopción en proyectos de desarrollo de software.

La combinación de la eficacia probada de los casos de uso con la potencia y facilidad de implementación promete un enfoque innovador para la generación automatizada de casos de prueba, por tanto, este método tiene el potencial de mejorar significativamente la eficiencia del proceso de pruebas, manteniendo al mismo tiempo la calidad y exhaustividad necesarias para garantizar la robustez del software desarrollado.

5. Metodología

Esta sección aborda aspectos clave para la ejecución del TIC, abarcando tanto la metodología empleada como las herramientas esenciales. En consonancia con el objetivo establecido en la **sección 5.1**, se detalla el área de estudio utilizada en el diseño del proyecto. En la **sección 5.2**, se describe el procedimiento que permitió alinear las acciones con los objetivos planteados, mientras que la **sección 5.3** proporciona un análisis de los recursos necesarios para llevar a cabo el proyecto de manera efectiva.

5.1. Área de estudio

El escenario para la ejecución del presente Trabajo de Integración Curricular (TIC) titulado "**Sistema para diseñar casos de prueba funcionales a partir de casos de uso en la carrera de Computación de la Universidad Nacional de Loja**", se constituye por estudiantes de la Carrera de Computación de la Facultad de Energía y Recurso no Renovables de la UNL, en la **Figura 5**, se muestra la locación donde se aplicará tanto el primer como el segundo objetivo, a fin de evaluar la percepción de utilidad de la aplicación, permitiendo cumplir con la planificación realizada y acatando el cronograma establecido.

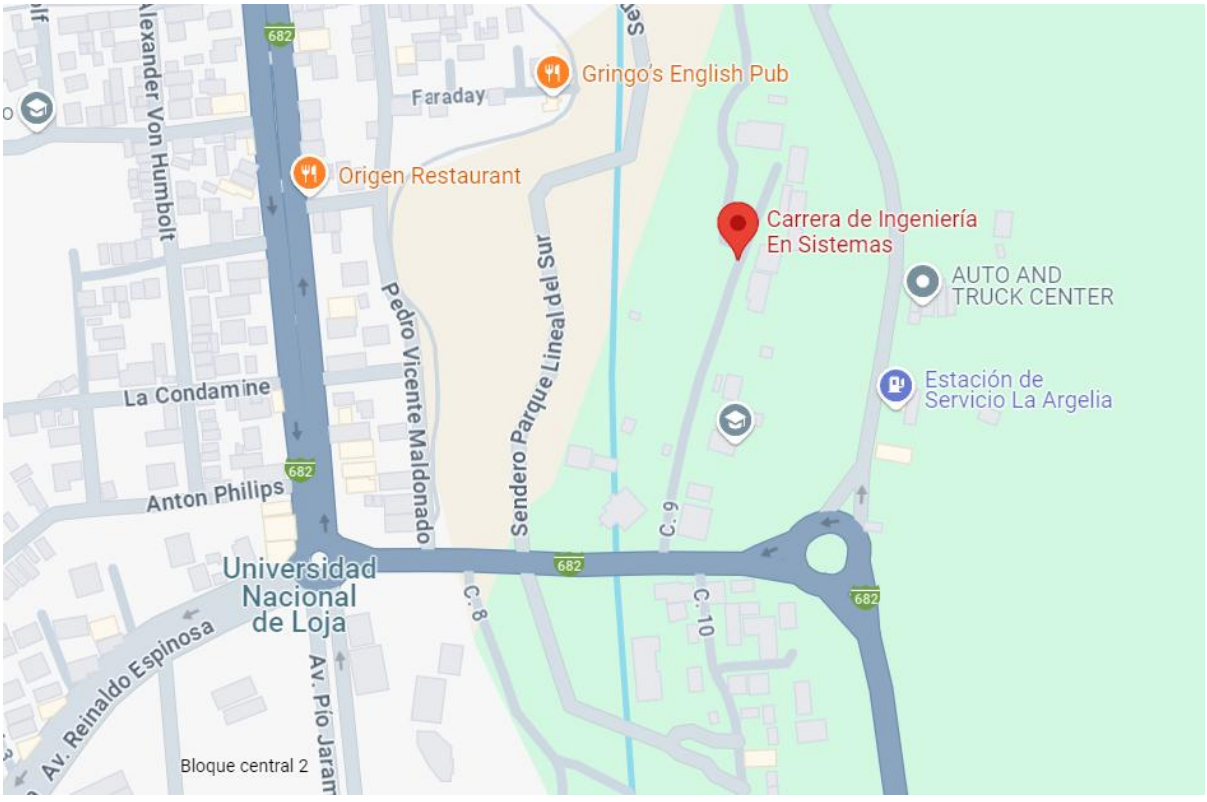


Figura 5: Carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja

5.2. Procedimiento

A continuación, describe el procedimiento establecido para alcanzar los objetivos planteados en el Proyecto TIC.

5.2.1. Objetivo 1: Desarrollar una herramienta de software que utilice la API de una IA para diseñar casos de prueba funcionales a partir de casos de uso, utilizando la metodología RAD (Desarrollo Rápido de Aplicaciones).

Para alcanzar la meta planteada en el objetivo 1 del TIC se hizo uso de la metodología de desarrollo de software RAD (Rapid Application Development) ajustando sus fases y tareas para adaptarlas a los requisitos del objetivo planteado.

5.2.1.1. Fase de Planificación de Requisitos

- Se utilizó la técnica de *Brainstorm* durante una reunión con las partes interesadas (estudiantes de la carrera de computación), para identificar y recopilar los requisitos funcionales y no funcionales del sistema **(véase sección 6.1.1)**.
- Se generó la lista priorizada de Funcionalidades, clasificada según su importancia y viabilidad, proveniente de la técnica aplicada y el Documento de Especificación de Requisitos el cual contiene las necesidades específicas del cliente y los objetivos del sistema **(véase sección 6.1.1).**

5.2.1.2. Fase de Diseño de Usuario

- Se desarrollaron diagramas de casos de uso **(véase sección 6.1.2)** que incluyen sus narraciones de las principales funcionalidades de la aplicación.
- Posteriormente, la arquitectura de la aplicación fue presentada mediante diagramas de componentes y despliegue para visualizar cómo los módulos interactúan entre sí y estructurar la aplicación de manera eficiente **(véase sección 6.1.2).**
- A partir de los casos de uso, se creó un modelo del dominio, representado mediante un diagrama de dominio y un diagrama de clases, para detallar la organización lógica de los datos; así mismo, para modelar la interacción general de la aplicación, se creó un diagrama de secuencia que ilustra el flujo lógico de las operaciones principales del sistema **(véase sección 6.1.2).**
- Se diseñaron prototipos no funcionales utilizando la herramienta Figma, enfocándose en las ventanas principales del sistema, dichas maquetas detalladas representaron la estructura visual y disposición de elementos en la interfaz **(véase sección 6.1.2).**

5.2.1.3. Fase de Construcción

- En esta etapa se desarrollaron los módulos del sistema en base a la fase de planificación de requisitos **(véase sección 6.1.3).**
- La codificación se llevó a cabo con tecnologías de desarrollo seleccionadas, tales como:
  - **Base de datos Relacional:** Postgres
  - **Backend:** Nest.js
  - **Frontend:** Next.js
- La interacción del sistema con la API se hizo a través de una consulta que posee un prompt, para este, se aplicó la estructura ROSES con la documentación de los casos de uso y caso de prueba, en conjunto a los estándares de calidad para que reforzar la petición a la IA **(véase sección 6.1.3).**

5.2.1.4. Fase de Transición

- Para supervisar la correcta integración entre módulos y el comportamiento tanto a nivel de componentes como de sistema, se realizaron pruebas unitarias y funcionales. Las pruebas unitarias se llevaron a cabo utilizando Jest para verificar que cada módulo cumpliera con los requisitos establecidos y respondiera adecuadamente **(véase en sección 6.1.4).**
- Adicionalmente, las pruebas funcionales se realizaron con herramientas como **Cypress** (para pruebas E2E), lo que permitió validar la interacción del usuario en el sistema en un entorno controlado **(véase sección 6.1.4).**
- El despliegue se lo realizó mediante archivos de configuración de Docker para ser ejecutados en el servidor de la Universidad Nacional de Loja. **(véase sección 6.1.4).**

5.2.2. Objetivo 2: Evaluar la percepción de utilidad de la herramienta desarrollada entre los estudiantes, utilizando como base ciertos lineamientos del modelo TAM (Modelo de Aceptación Tecnológica).

5.2.2.1. Fase de Planificación de evaluación

- Para evaluar la percepción de utilidad se realizó una encuesta focalizada en la percepción de utilidad, basada en el método TAM **(véase sección 6.2.1).**

5.2.2.2. Fase de Visualización de resultados

- Se realizó la separación en matrices de la información obtenida **(véase sección 6.2.2)**

5.2.2.3. Fase de Análisis de Resultados

- Se evaluó el análisis y evaluación de los resultados **(véase sección 6.2.3)**

5.2.2.4. Fase de Ciclo de mejora

- Se aplica una encuesta de satisfacción para evaluar la aplicación y a partir de dichos puntos, refinar la aplicación antes de su entrega final **(véase sección 6.2.4)**

5.3. Recursos

En esta sección, se detallan los recursos empleados en el presente TIC. En la subsección **5.3.1** se exponen los recursos científicos, mientras que en la **5.3.2** se incluyen los recursos técnicos utilizados en el proyecto. Por último, se enumeran tanto los recursos de software como los de hardware en la subsección **5.3.3**.

5.3.1. Recursos Científicos

5.3.1.1. Estudios de Caso

Se analizó el proceso de diseño de casos de prueba en la carrera de Computación de la UNL, a fin de determinar el flujo general, validando estos métodos mediante estándares como el de ISTQB o IEEE, encontrando el flujo ideal que se ajuste al contexto del desarrollo.

5.3.1.2. Investigación Bibliográfica

Revisión de casos similares en un contexto más amplio, así como revisión de soluciones existentes **(véase sección 4.3)**

5.3.1.3. Observación directa

Se observó el uso de la aplicación en los diferentes entornos controlados que se plantearon, tomando nota de los comportamientos de los grupos de muestra, con el objetivo de apreciar la percepción de utilidad de cada individuo.

5.3.2. Recursos Técnicos

5.3.2.1. Brainstorm

Permitieron obtener información valiosa sobre el proceso de diseño de casos de prueba funcionales, aplicando la técnica de brainstorm para la recolección de requisitos **(véase sección 6.1.1).**

5.3.2.2. Encuestas

Se aplicaron a los estudiantes de la carrera de Computación, con el fin de identificar la percepción de utilidad y encontrar posibles puntos para el ciclo de mejora **(véase sección 6.2.1).**

5.3.2.3. Entrevistas

Se aplicó al especialista de calidad de la carrera de Computación, a fin de reforzar las ideas propuestas por los estudiantes en la técnica de Brainstorm, pues el punto de vista de una persona con mayor experiencia y conocimiento proporciona un mejor desarrollo de software (**véase Anexo 1**).

5.3.3. Recursos de Hardware y Software

5.3.3.1. Hardware

- **Laptop Gigabyte**: El uso de este dispositivo fue indispensable tanto para realizar las diversas investigaciones necesarias en este proyecto como para llevar a cabo el desarrollo de la solución informática.
- **Teléfono Móvil**: El uso de este fue necesario para la grabación y toma de evidencias para la entrevista y encuesta.

5.3.3.2. Software

- **Visual Studio Code:** Editor de código utilizado para la implementación de la Aplicación Web.
- **Nest.js:** Framework backend utilizado para estructurar y organizar la lógica del servidor.
- **Next.js:** Framework frontend utilizado para la creación de la interfaz de usuario y renderizado del lado del servidor.
- **Postgres:** Sistema de gestión de bases de datos relacional utilizado para almacenar la información de la aplicación.

5.4. Participantes

El trabajo de titulación se realizó por el estudiante Juan Francisco Castillo Estrella bajo la dirección del Ing. Wilman Chamba, docente de la Universidad Nacional de Loja y con la participación de estudiantes de la carrera de Computación de la misma institución.

6. Resultados

En la presente sección, se muestran los resultados conseguidos durante la elaboración del TIC. En la sección **6.1** se presentan los resultados del primer objetivo, el cual estuvo focalizado en el desarrollo del software para el diseño de casos de prueba funcionales a partir de casos de uso, utilizando la metodología RAD, mientras que, en la sección **6.2** se muestra la evaluación de la percepción de la utilidad del software desarrollado, tratado mediante la metodología TAM y escalas de Likert.

6.1. Objetivo 1: Desarrollar una herramienta de software que utilice la API de una IA para diseñar casos de prueba funcionales a partir de casos de uso, utilizando la metodología RAD (Desarrollo Rápido de Aplicaciones).

Para cumplir con el primer objetivo se siguieron las fases de la metodología RAD antes definidas:

6.1.1. Fase de Planificación de Requisitos.

- **Técnica de Brainstorm.**

En esta fase se aplicó la técnica de Brainstorm que consiste en la recoger ideas a partir de ciertas preguntas, para así poder obtener requisitos funcionales y no funcionales provenientes de una muestra de estudiantes en específico (Para ver todo el proceso de la técnica vea el **Anexo 5**).

En la **Tabla 3**, se muestra el resultado inicial de la sesión, evidenciando la cantidad de ideas proporcionadas en esta primera fase y la cual es relevante debido a que los requisitos que posteriormente se elaboraron.

Tabla 3: Ideas Iniciales de Brainstorm

Nro. Idea	Ideas
1	Debería tener los stakeholders con sus respectivos casos de uso
2	Mostrar reportes detallados respecto a las actividades hechas
3	Debe tener una navegación sencilla y de fácil uso
4	Que muestre la o las sugerencias de casos de prueba
5	Tener un método de autenticación para mayor seguridad al ingresar a la plataforma
6	Limitar los tiempos de sesiones para reducir el costo computacional de la aplicación
7	Formato estándar o formato dependiendo la metodología que usa
8	Debe tener limitado la cantidad de información que pueden subir los usuarios a la aplicación para no ocupar un gran almacenamiento
9	Por cada caso de estudio se puedan añadir imágenes
10	Debe tener una manera de recuperar la contraseña en caso de que pierda
11	Generar casos de prueba con una técnica de pruebas
12	Subir un grupo de casos de uso
13	Las respuestas deben ser rápidas
14	Que los casos de prueba que ya están listos tengan un checklist para verificar cuales están ejecutados o una matriz
15	Que permita agregar casos de uso, por comando de voz
16	Brindar ejemplos sobre cómo crear casos de prueba para un resultado eficiente
17	Ver los cambios que han hecho los demás compañeros
18	Generación de casos de prueba
19	Los casos de prueba listos para integrarse en Jira
20	Resultados en formato PDF
21	Validación de resultados esperados
22	Edición y personalización de casos
23	Gestionar Perfil
24	Ofrecer recomendaciones automáticas para mejorar la calidad del diseño de casos de prueba
25	Integrarse con otras plataformas comunes como Trello, Github y GitLab.

A partir de la reunión se realizó un análisis basado en prioridad o necesidad en la aplicación, para ello se usó del método MoSCoW, para así convertir las ideas potenciales en requisitos potenciales y poder refinarlos en una fase posterior, para un mayor detalle y evidenciar lo ejecutado se puede visualizar el **Anexo 2.**

En la **Tabla 4**, se puede visualizar la lista de requisitos potenciales que se extrajo a partir de la aplicación de la técnica de Brainstorm.

Tabla 4: Requisitos Potenciales

Código	Requisitos
R001	El sistema debe implementar autenticación mediante correo y contraseña.
R002	El sistema debe cerrar sesiones después de 15 minutos de inactividad.
R003	El sistema debe permitir crear usuarios.
R004	El sistema debe permitir actualizar usuarios.
R005	El sistema debe permitir buscar usuarios por nombre.
R006	El sistema debe permitir dar de baja usuarios.
R007	El sistema debe permitir la asignación de roles (Administrador, Usuario).
R008	El sistema debe mostrar una guía de cómo utilizar la herramienta.
R009	El sistema debe permitir compartir proyectos con otros usuarios.
R010	El sistema debe implementar un diseño responsive para diferentes dispositivos.
R011	El sistema debe implementar un modo oscuro/claro.
R012	El sistema debe exportar reportes en formato PDF IEEE 829.
R013	El sistema debe mantener tiempos de respuesta menores a 5 segundos para operaciones comunes.

Código	Requisitos
R014	El sistema debe permitir diseñar casos de prueba funcionales a partir de casos de uso.
R015	El sistema debe permitir la edición y personalización de los casos de prueba funcionales diseñados.
R016	El sistema debe proporcionar ejemplos sobre cómo crear casos de prueba funcionales.
R017	El sistema debe permitir crear proyectos.
R018	El sistema debe permitir actualizar proyectos
R019	El sistema debe permitir buscar proyectos por nombre
R020	El sistema debe permitir dar de baja proyectos
R021	El sistema debe permitir crear casos de uso.
R022	El sistema debe permitir actualizar casos de uso
R023	El sistema debe permitir buscar casos de uso por nombre
R024	El sistema debe permitir dar de baja casos de uso
R025	El sistema debe permitir visualizar una explicación paso a paso de cómo se diseñaron los casos de prueba funcionales.
R026	El sistema debe permitir cerrar sesión
R027	El sistema debe permitir recuperar contraseña.
R028	El sistema debe permitir cifrar las claves de cuentas de usuario.
R029	El sistema deberá consumir a la API de Cohere.
R030	El sistema debe permitir personalizar y actualizar el perfil del usuario.
R031	El sistema debe limitar a 10 el número máximo de proyectos a crear por usuario

- **Requisitos Funcionales.**

Como resultado del análisis de las ideas recopiladas durante la sesión de *Brainstorming*, se generó una lista de requisitos potenciales que fueron refinados y redactados de manera más precisa. En la **Tabla 5** se presentan los requisitos funcionales del sistema que determinan las funcionalidades que posee el software, para un detalle completo de los requisitos funcionales, consulte el **Anexo 3.**

Tabla 5: Requisitos Funcionales.

Código	Nombre	Descripción	Actor	Prioridad
RF001	Autenticar usuario	Debe permitir la autenticación (iniciar sesión) a usuarios mediante validación de credenciales (correo y contraseña).	Todos los usuarios	Alta
RF002	Registrarse	Debe permitir a usuarios generales registrarse con un correo electrónico único y la siguiente información: • Nombre • Apellido • Contraseña Y su rol por defecto será Tester.	Usuario No registrado	Alta
RF003	Recuperar contraseña	Debe permitir la recuperación de contraseña en caso de olvido.	Todos los usuarios	Alta
RF004	Actualizar el perfil del usuario	Debe permitir la modificación de su perfil, se puede modificar:	Todos los usuarios	Baja

Código	Nombre	Descripción	Actor	Prioridad
		• Nombre • Apellido • Imagen		
RF005	Actualizar roles de usuario	Debe permitir la actualización de roles de usuarios	Administrador	Alta
RF006	Buscar usuarios por correo electrónico	Debe permitir buscar la información de usuarios mediante correo electrónico	Administrador	Media
RF007	Desactivar usuarios	Permitir desactivar usuarios registrados.	Administrador	Alta
RF008	Crear proyectos	Debe permitir el registro de nuevos proyectos con la siguiente información: • Nombre • Descripción • Imagen	Tester	Alta
RF009	Actualizar proyectos	Debe permitir la actualización de proyectos con la siguiente información: • Nombre • Descripción • Imagen	Tester	Alta
RF010	Buscar Proyectos por Nombre	Debe permitir visualizar la información de proyectos (Nombre, Descripción, Imagen)	Tester	Media
RF011	Eliminar proyectos	Permitir eliminar sus proyectos del sistema.	Tester	Alta
RF012	Compartir proyectos	Debe ofrecer la opción de colaboración a través de compartir proyectos con otros usuarios testers mediante una invitación al correo electrónico	Tester	Media
RF013	Limitar a 10 el número máximo de proyectos a crear por usuario	Debe controlar el número de proyectos que un usuario Tester puede crear	Sistema	Media
RF014	Crear casos de uso	Debe permitir el registro de nuevos casos de uso a un proyecto con la siguiente información: • Nombre • Descripción • Entradas • Precondiciones • Postcondiciones • Flujo Normal • Flujo Alterno	Tester	Alta
RF015	Actualizar casos de uso	Debe permitir la actualización de casos de uso de un proyecto con la siguiente información: • Nombre • Descripción • Entradas • Precondiciones • Postcondiciones • Flujo Normal • Flujo Alterno	Tester	Alta

Código	Nombre	Descripción	Actor	Prioridad
RF016	Buscar Casos de uso	Debe permitir visualizar la información de casos de uso de un proyecto (Nombre, Descripción, Entradas)	Tester	Media
RF017	Eliminar casos de uso	Permitir la eliminación de casos de uso de un proyecto del sistema.	Tester	Alta
RF018	Crear casos de prueba funcionales a partir de casos de uso	Debe permitir generar casos de prueba funcionales en base a un caso de uso.	Tester	Alta
RF019	Actualizar casos de prueba funcionales diseñados	Actualizar la información de un caso de prueba funcional, con la siguiente información: • Nombre • Descripción • Pasos • Entradas • Resultado Esperado	Tester	Alta
RF020	Eliminar Casos de prueba funcionales	Permitir la eliminación de un caso de prueba funcional de un caso de uso en el sistema.	Tester	Alta
RF021	Buscar casos de prueba funcionales por Nombre	Debe permitir visualizar la información de los casos de prueba funcionales (Nombre, Descripción)	Tester	Media
RF022	Generar reportes de proyectos en formato PDF	Debe ofrecer la funcionalidad de exportación de reportes en formato PDF	Tester	Media

- **Requisitos No Funcionales.**

En la **Tabla 6** se muestran los requisitos no funcionales de la aplicación web (Detalle de requisitos no funcionales véase en **Anexo 3**).

Tabla 6: Requisitos No Funcionales.

Categoría	Código	Descripción
Seguridad	**RNF001**	Cierre automático de sesión tras 15 minutos de inactividad.
	RNF002	Cifrar las claves de cuentas de usuario
Usabilidad	**RNF003**	El diseño del sistema debe ser de tipo Responsive y se adapte a todo tamaño de pantalla
	RNF004	Mostrar una guía de cómo crear casos de prueba funcionales con la herramienta
Interfaz Gráfica de Usuario	**RNF005**	La aplicación web debe contar con interfaces amigables e intuitivas, que permitan realizar el proceso de estimación de costos de manera sencilla.
	RNF006	Modo oscuro y claro.

Categoría	Código	Descripción
Eficiencia	**RNF007**	Optimizar el rendimiento para que las operaciones más comunes se ejecuten entre 5 a 10 segundos
Disponibilidad	**RNF008**	La aplicación web debe estar disponible dentro del laboratorio de software, siempre y cuando el estado de la infraestructura de los servidores de la carrera de Computación sea óptimo y estén en funcionamiento.

6.1.2. Fase de Diseño de Usuario

Una vez finalizada la recopilación de información mediante la técnica de Brainstorming y la especificación de los requisitos funcionales y no funcionales, se procedió a estructurar la información obtenida en un Modelado de Casos de Uso; luego para en un Modelado de la arquitectura donde se presentan diagramas de despliegue y componentes; también para representar el Modelado del Dominio se creó diagrama de clases UML y para el flujo de la aplicación se creó un diagrama de secuencia, para que finalmente se elaboren los prototipos de la aplicación.

- **Modelado de Casos de Uso.**

Se creó un diagrama de casos de uso que describe en términos generales todas las interacciones del sistema con el usuario, en la **Figura 6** se muestra el diagrama de casos de uso general con las distintas acciones de cada tipo de usuario; para mayor detalle se muestran los diagramas de casos de uso completos con su narración en el **Anexo 4**.

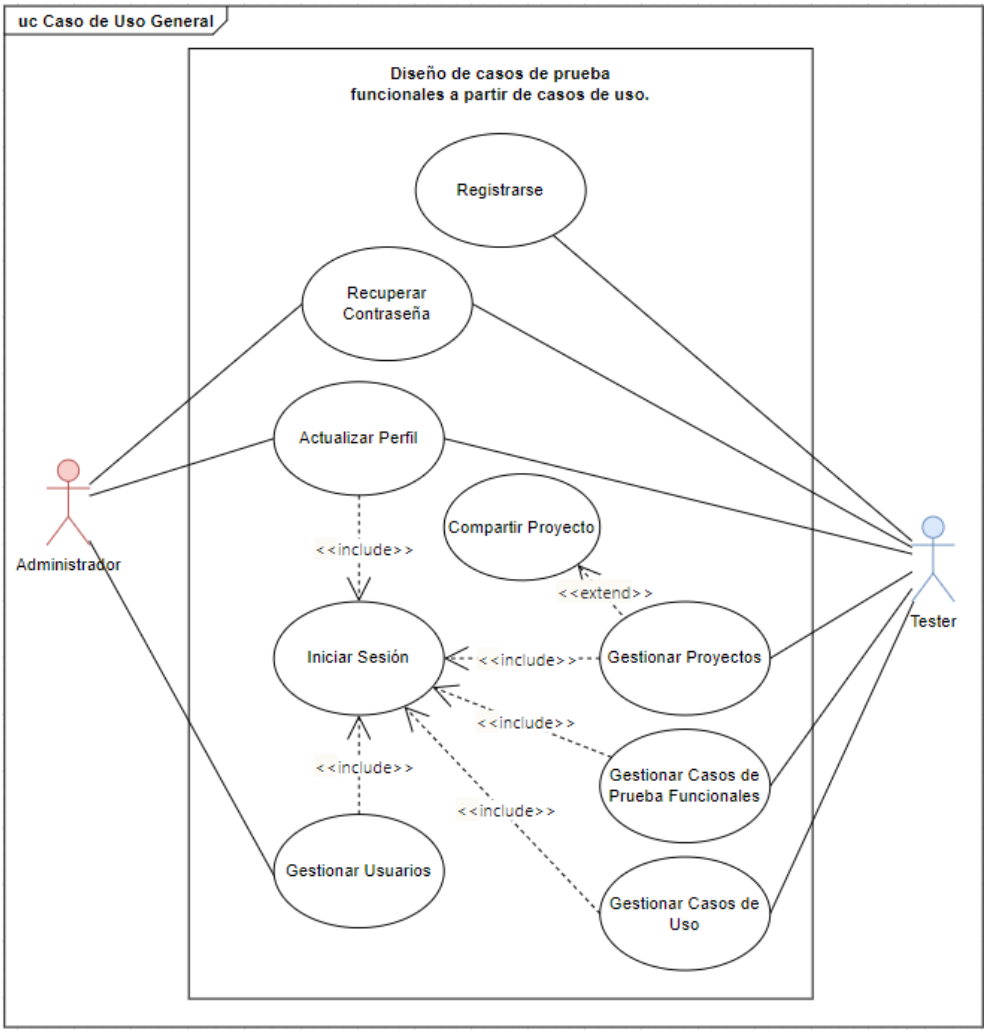


Figura 6: Diagrama de Casos de Uso - General

En la **Tabla 7,** se presenta la descripción de caso de uso iniciar sesión, que aplica a todos los usuarios y para todas las acciones, con la excepción de Recuperar contraseña y Registrarse, en esos casos no será necesario haber accedido al sistema con credenciales de usuario.

Tabla 7: Descripción de Caso de Uso - Iniciar Sesión

Identificador	**CU01**
Nombre	Iniciar sesión
Actores	Todos los usuarios del sistema
Descripción	Autenticación de usuario en el sistema
Precondiciones	El usuario debe estar registrado. Estar ubicado en la página de “Iniciar sesión”
Postcondiciones	El usuario accede a la interfaz principal del sistema.
Flujo Principal	El usuario ingresa sus credenciales de acceso (correo y contraseña) El usuario pulsa el botón Iniciar Sesión El sistema verifica las credenciales. El sistema autentica al usuario y accede al sistema.

Identificador	CU01
Flujo Alterno	**Credenciales Incorrectas** A1: Si el usuario ingresa una contraseña incorrecta, el sistema muestra un mensaje de error. **Intentos Permitidos** A2: Si el usuario excede el número de intentos permitidos, el sistema bloquea la cuenta temporalmente. **Campos Vacíos** A3: Si el usuario pulsa el botón Iniciar Sesión sin llenar los campos, muestra un mensaje indicando que hay campos vacíos en el formulario

Posteriormente, se muestran los casos de uso de cada modelo, con una descripción que ejemplifica la forma de manejar dicha información. En la **Figura 7** se muestra el caso de Uso de Usuarios, el cual únicamente tiene acceso el Usuario con Rol de Administrador.

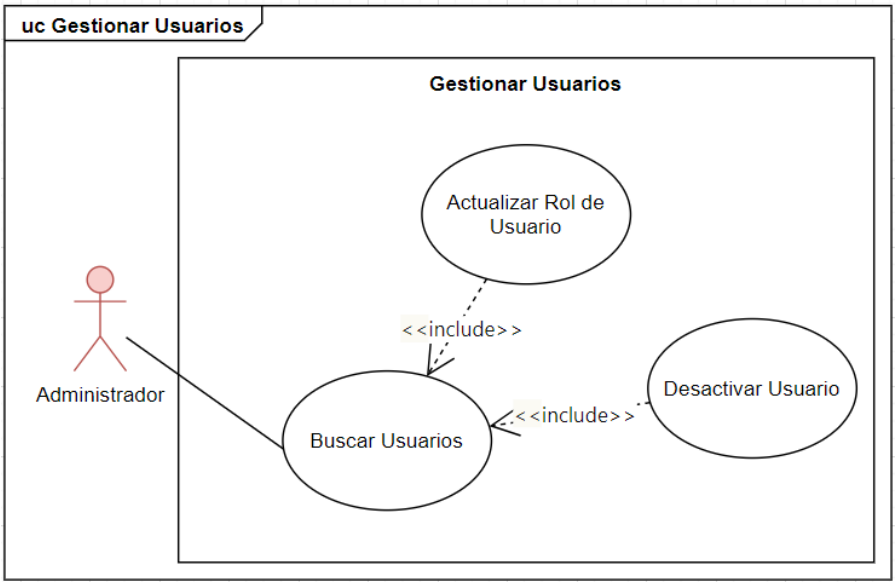


Figura 7: Diagrama de Casos de Uso - Usuarios

En la **Tabla 8,** se presenta la descripción de caso de uso Actualizar roles de usuario, que aplica al usuario con rol de Administrador.

Tabla 8: Descripción de Casos de Uso - Actualizar roles de usuario

Identificador	CU05
Nombre	Actualizar roles de usuario
Actores	Administrador
Descripción	Modificar el rol de un usuario
Precondiciones	El administrador debe estar autenticado Ubicado en la página de Listar Usuarios Buscar usuario
Postcondiciones	Los datos del usuario son actualizados.
Flujo Principal	El administrador selecciona el usuario a modificar. El administrador accede a la opción de editar

Identificador	CU05
	El administrador actualiza el rol El sistema valida datos El sistema guarda los cambios y confirma la actualización.

Posteriormente, en la **Figura 8** se adjunta el caso de uso de Proyectos, que presenta la gestión de los proyectos desde la vista del Usuario con Rol de Tester.

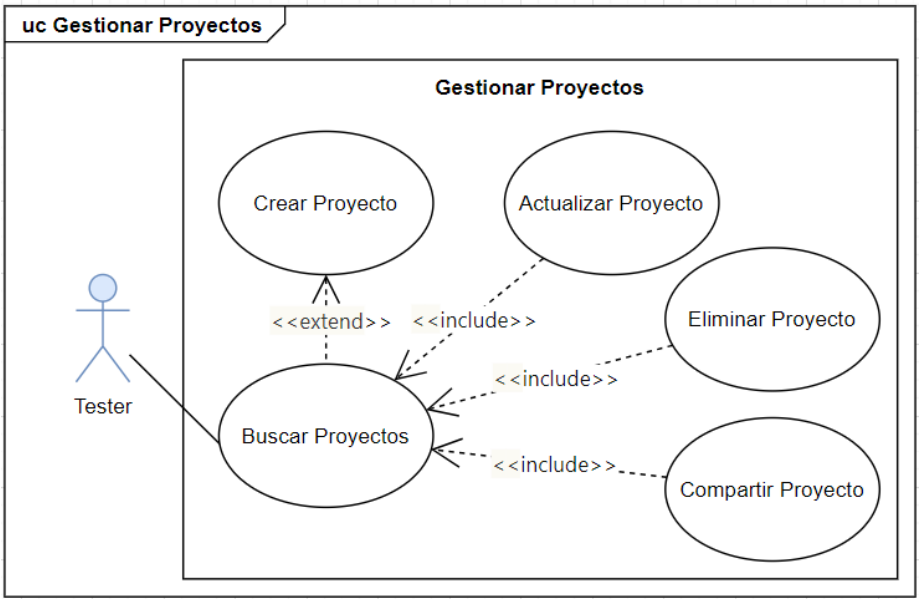


Figura 8: Diagrama de Casos de Uso - Proyectos

En la **Tabla 9,** se presenta la descripción de caso de uso Crear proyecto, que aplica al usuario con rol de Tester.

Tabla 9: Descripción de Casos de Uso - Crear Proyecto

Identificador	CU08
Nombre	Crear proyecto
Actores	Tester
Descripción	Crear un nuevo proyecto.
Precondiciones	El Tester debe estar autenticado. Ubicado en la página de Listar Proyectos
Postcondiciones	El proyecto se agrega a la lista de proyectos del usuario.
Flujo Principal	El Tester pulsa el botón de “Crear proyecto” El sistema muestra el formulario de crear proyecto El Tester ingresa la información del nuevo proyecto (nombre, descripción, imagen) El Tester pulsa el botón de Crear El sistema valida los datos ingresados. El sistema guarda el nuevo proyecto y muestra un mensaje de confirmación.
Flujo Alterno	**Límite de proyectos** A1: Si el Tester excede el límite de proyectos, el sistema muestra un mensaje de advertencia. **Campos vacíos** A2: El Tester no ingresa todos los campos requeridos, el sistema muestra un mensaje de error.

Posteriormente, en la **Figura 9** se adjunta el caso de uso de Casos de Uso, que presenta la gestión de los casos de uso de un proyecto perteneciente a un Usuario con Rol de Tester.

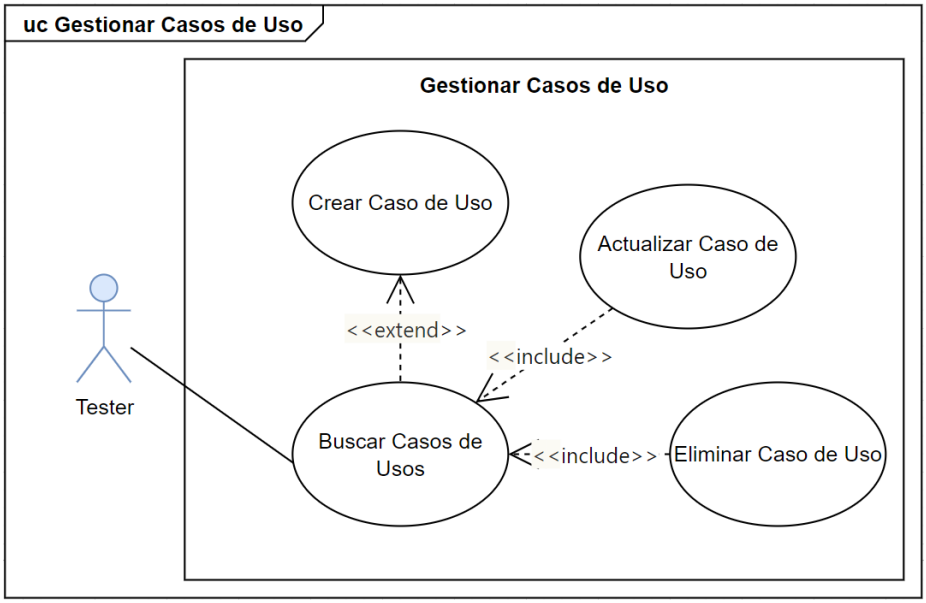


Figura 9: Diagrama de Casos de Uso - Casos de Uso

En la **Tabla 10** se adjunta la narración del caso de uso de Casos de Uso, que presenta la gestión de los casos de uso de un proyecto perteneciente a un Usuario con Rol de Tester.

Tabla 10: Descripción de Casos de Uso - Crear Caso de Uso

Identificador	**CU013**
Nombre	Crear Caso de uso
Actores	Tester
Descripción	Crear un nuevo Caso de uso en un proyecto
Precondiciones	El Tester debe estar autenticado. Ubicado en la página de Listar Casos de usos
Postcondiciones	El Caso de uso se agrega a la lista de proyectos del Tester.
Flujo Principal	El Tester pulsa el botón de “Crear Caso de uso” El sistema muestra el formulario de crear Caso de uso El Tester ingresa la información del nuevo Caso de uso (nombre, descripción, entradas, flujo normal, flujo alterno) El Tester pulsa el botón de Crear El sistema valida los datos ingresados. El sistema guarda el nuevo Caso de uso y muestra un mensaje de confirmación.
Flujo Alterno	**Campos vacíos** A1: El Tester no ingresa todos los campos requeridos, el sistema muestra un mensaje de error.

Posteriormente, en la **Figura 10** se adjunta el caso de uso de gestionar Casos de Prueba Funcionales que lo pueden ejecutar Usuarios con Rol de Tester.

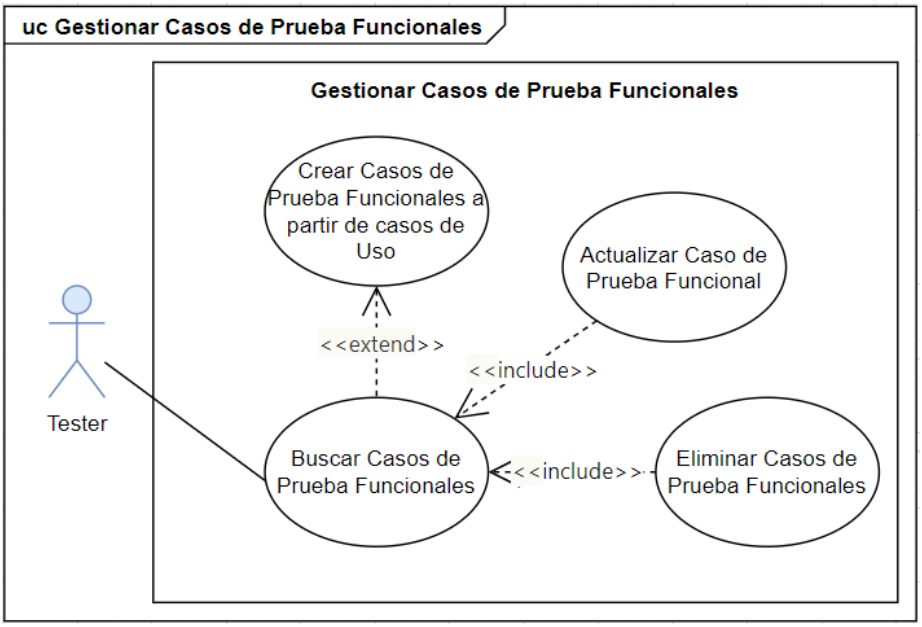


Figura 10: Diagrama de Casos de Uso - Casos de Prueba Funcionales

En la **Tabla 11,** se muestra un ejemplo de la descripción de los casos de uso el cual lleva de nombre “Crear casos de prueba funcionales a partir de casos de uso”, y en el **Anexo 6** se incluye la descripción detallada de cada caso de uso.

Tabla 11: Descripción de Casos de Uso – Crear Casos de prueba funcionales a partir de casos de uso.

Identificador	**CU017**
Nombre	Crear casos de prueba funcionales a partir de casos de uso
Actores	Tester
Descripción	Diseñar casos de prueba funcionales en base a un caso de uso.
Precondiciones	El Tester debe estar autenticado El Tester debe tener casos de uso registrados.
Postcondiciones	Se genera el caso de prueba funcional en el sistema.
Flujo Principal	El Tester selecciona un caso de uso. El Tester selecciona la opción crear Caso de Prueba Funcional El sistema crea los casos de prueba funcionales basado en el caso de uso. Los casos de prueba funcionales se almacenan y se muestran al Tester en la tabla correspondiente a Casos de Prueba Funcionales
Flujo Alterno	**Error en la petición** A1: Si existe un error en la comunicación de la aplicación con la API de la IA, no se puede generar el caso de prueba, el sistema muestra un mensaje de error. **Caso de Uso Inválido** A2: Si el caso de uso posee información inválida, el sistema muestra un mensaje de error

- **Modelado de la Arquitectura.**

Se elaboraron diagramas UML de despliegue y de componentes para representar la arquitectura seleccionada en el desarrollo de la aplicación. Estos diagramas incorporan diversos estilos arquitectónicos, tales como:

- Para el despliegue se utilizó el estilo arquitectónico cliente servidor cuya distribución se basa en dos monolitos, para separar los servicios de la vista.
- Para la estructura se utilizó el estilo arquitectónico Basado en Componentes cuya distribución se basa en la estructura del repositorio del proyecto.
- Para la Comunicación se utilizó el estilo arquitectónico ROA (resource oriented architecture) para que las peticiones por parte del frontend se hicieran mediante peticiones REST a través de HTTP.
- Para el Dominio se utilizó el estilo arquitectónico Domain Driven Design para que el software desarrollado se focalice en la situación problemática y en sus relacionados, haciendo que desarrolladores posteriores puedan usar dicho software especializado en el tema.

Para mayor detalle de los diagramas y proceso realizado en la arquitectura del software véase el **Anexo 8**.

- **Diagrama de Despliegue.**

  En la **Figura 11,** se muestra el diagrama de despliegue de la aplicación:

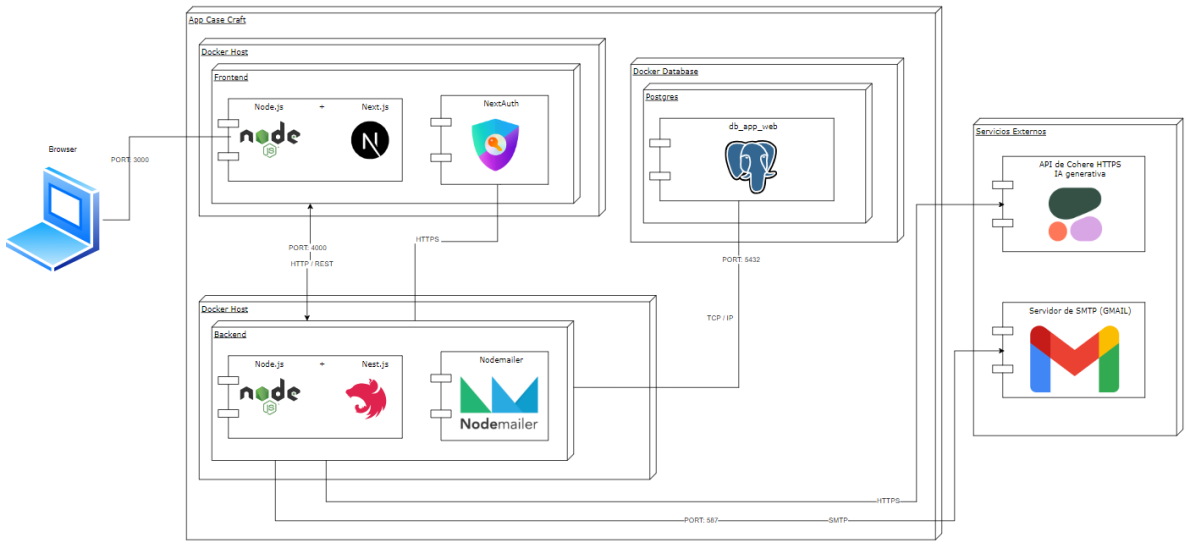


Figura 11: Diagrama de Despliegue.

- **Diagrama de Componentes.**

En la **Figura 12**, se presenta el diagrama de componentes de la aplicación:

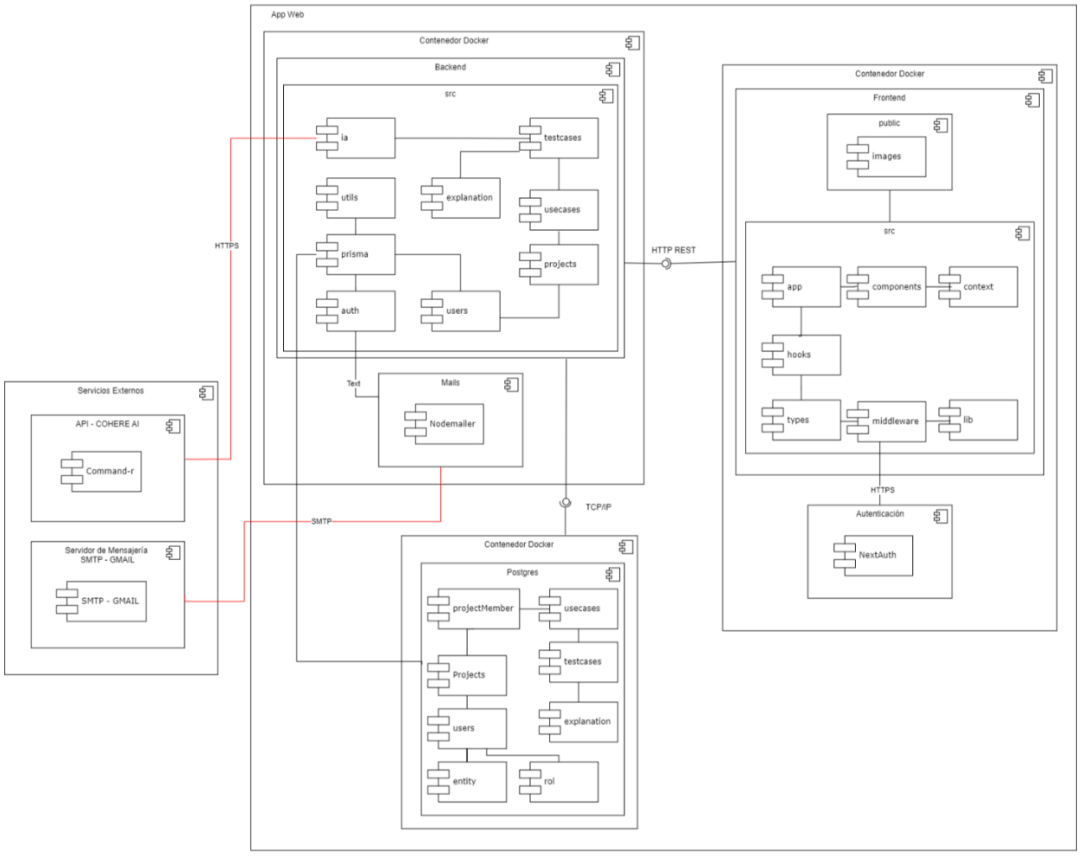


Figura 12: Diagrama de Componentes.

- **Modelado del Dominio.**

Se realizaron diagramas UML de acuerdo con la arquitectura seleccionada para plasmar el dominio de la aplicación web. A continuación, se presentan los principales resultados.

- **Diagrama de Clases UML.**

En la **Figura 13**, se presenta el diagrama de clases UML de la aplicación web.

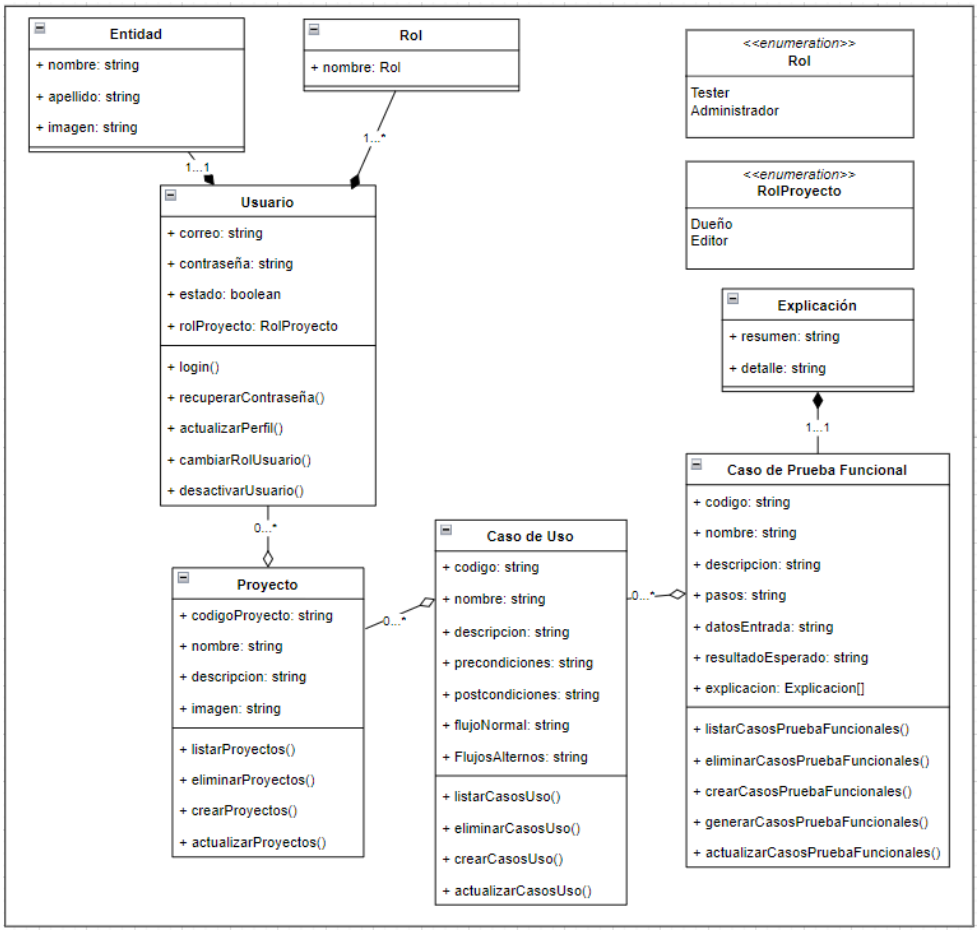


Figura 13: Diagrama de Clases UML.

- **Diagramas de Secuencia**

En la **Figura 14** se muestra el diagrama de secuencia UML del Inicio de sesión.

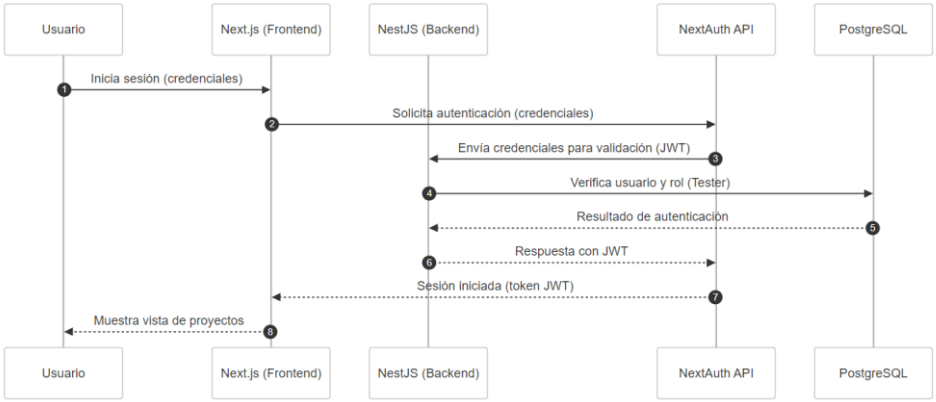


Figura 14: Diagrama de Secuencia para el funcionamiento general de la Aplicación Web para el escenario – Iniciar sesión

En la **Figura 15** se muestra el diagrama de secuencia UML de Crear Proyecto.

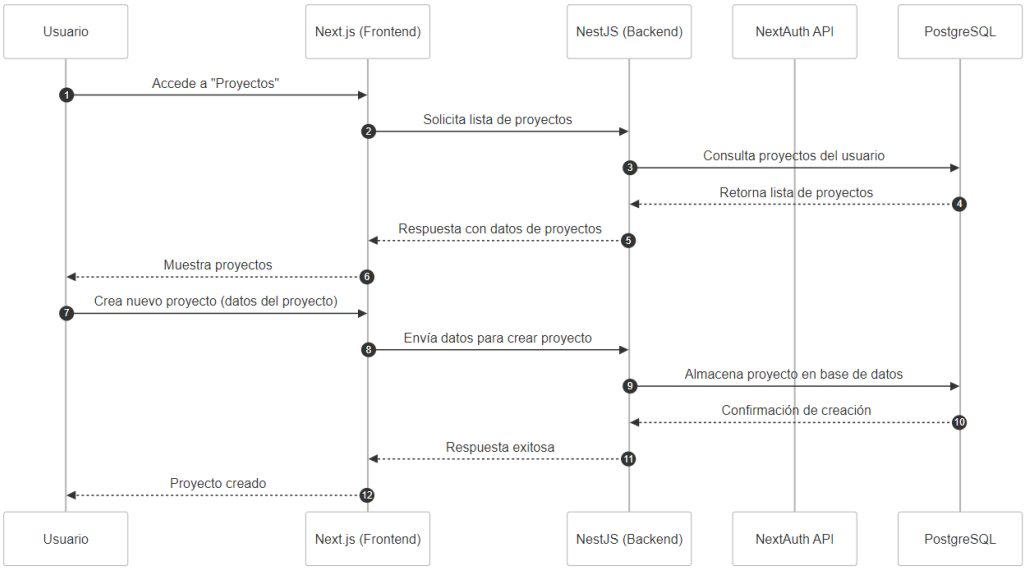


Figura 15: Diagrama de Secuencia para el funcionamiento general de la Aplicación Web para el escenario – Crear proyecto

En la **Figura 16** se muestra el diagrama de secuencia UML de Crear caso de uso.

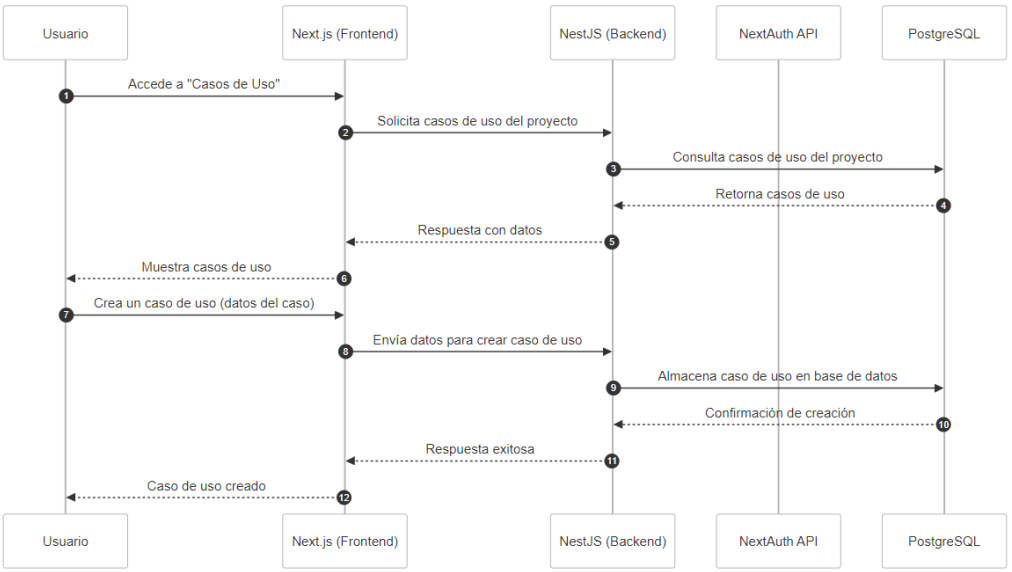


Figura 16: Diagrama de Secuencia para el funcionamiento general de la Aplicación Web para el escenario – Crear Caso de Uso

En la **Figura 17** se muestra el diagrama de secuencia UML de Generar casos de prueba funcionales.

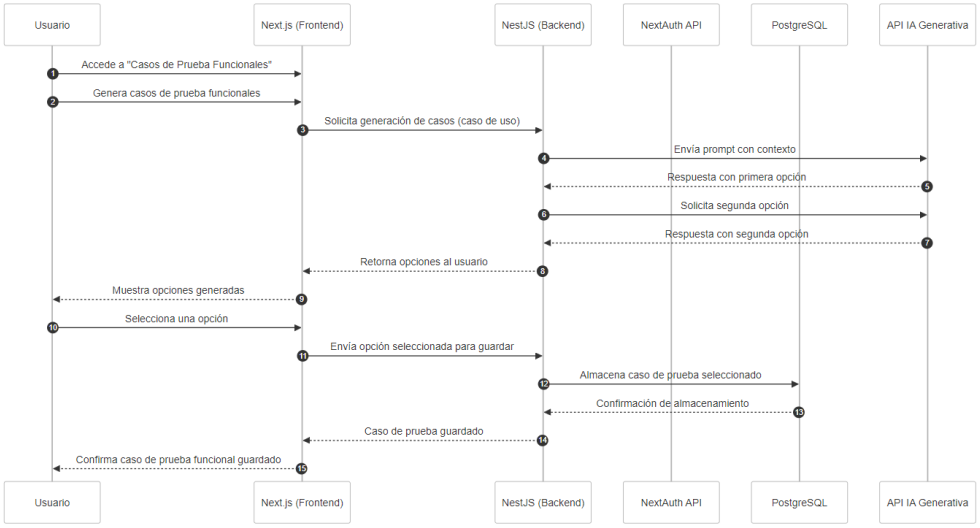


Figura 17: Diagrama de Secuencia para el funcionamiento general de la Aplicación Web para el escenario – Generar casos de prueba funcionales

En la **Figura 18** se muestra el diagrama de secuencia UML de Exportar PDF de Proyecto.

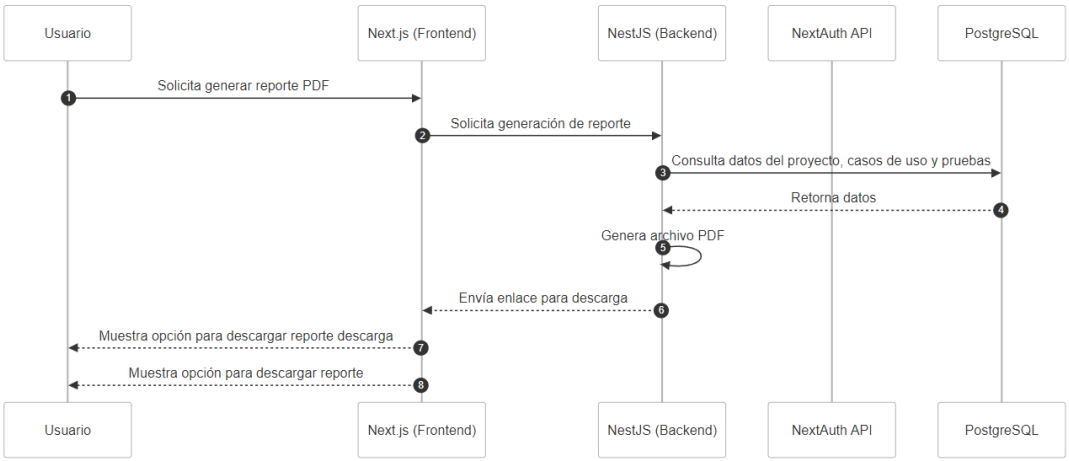


Figura 18: Diagrama de Secuencia para el funcionamiento general de la Aplicación Web para el escenario – Exportar PDF de Proyecto

- **Prototipos**

Se diseñaron prototipos de interfaces de usuario con el objetivo de representar de manera eficiente los componentes a desarrollar. En la **Figura 19** se muestra un ejemplo del prototipo de la pantalla inicial, mientras que el resto de los prototipos se encuentran detallados en el **Anexo 5.**

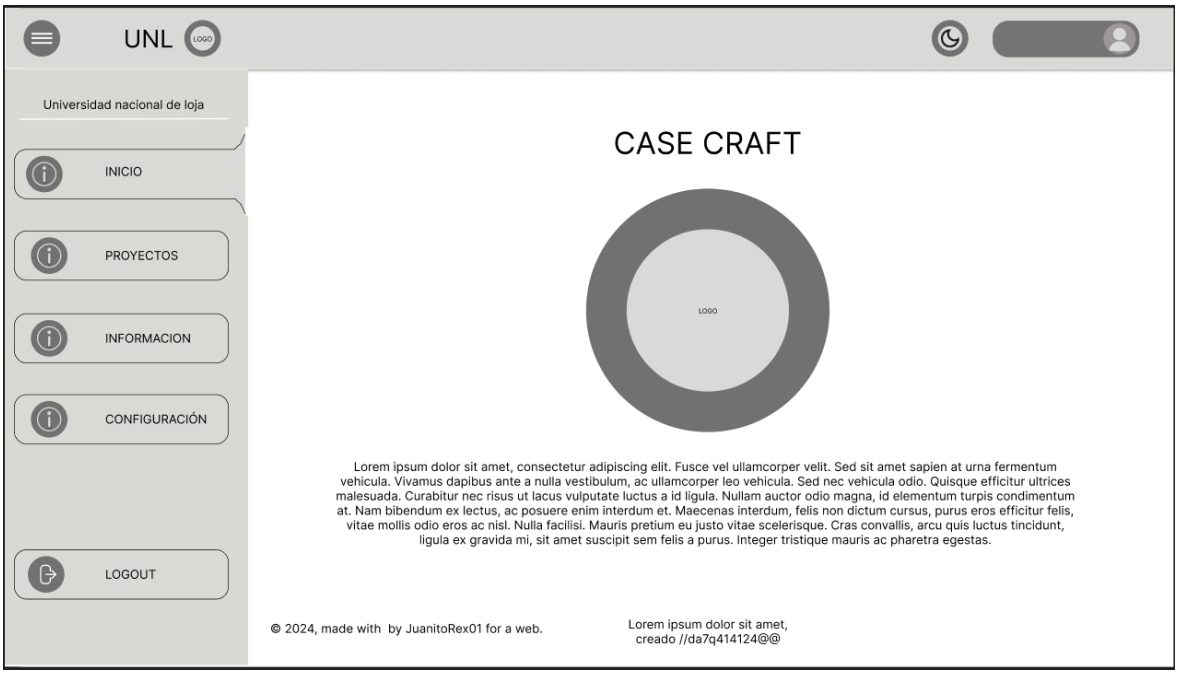


Figura 19: Prototipo de Pantalla Inicial del Sistema

6.1.3. Fase de Construcción

En esta fase de construcción se utilizó las siguientes tecnologías, en la **Tabla 12** se muestran los entornos de desarrollo que se utilizaron para el desarrollo de la aplicación web. Para conocer todo el contenido de la fase de construcción se puede visualizar el desarrollo de la metodología en el **Anexo 8**.

Tabla 12: Entornos de Desarrollo

Categoría	Especificación	Versión	Descripción
Lenguaje de Programación	Typescript	^5	Lenguaje principal para el desarrollo frontend y backend, proporcionando tipado estático y mejores prácticas de desarrollo
Framework Frontend	Next.js	14.2.11	Framework de React para desarrollo con funcionalidades de SSR y API routes
Framework Backend	Nest.js	^10.0.0	Framework de Express para desarrollo con funcionalidades de SSR y API routes
Runtime	Node.js	v20.9.0	Entorno de ejecución para JavaScript/TypeScript en el servidor

Además, en la **Tabla 13** se muestro los Frameworks, Toolkit y librerías que se utilizaron para el desarrollo de la aplicación web.

Tabla 13: Frameworks y Librerías Principales.

Librería/Framework	Versión	Propósito	Dependencias Críticas
React	18.x	UI Framework	Ninguna
Tailwind CSS	3.x	Framework de estilos	PostCSS
Prisma	5.x	ORM para base de datos	Ninguna
Shadcn/ui	1.x	Componentes de UI	Tailwind CSS, RadixUI

La construcción de la aplicación web cuenta con dos entornos, en la **Figura 20** se muestra el primero de backend, en el cual se enfocó a la parte del servidor y definir las funciones, modelos, esquemas, rutas, middlewares, librerías y controladores, de acuerdo con el diagrama de componentes.

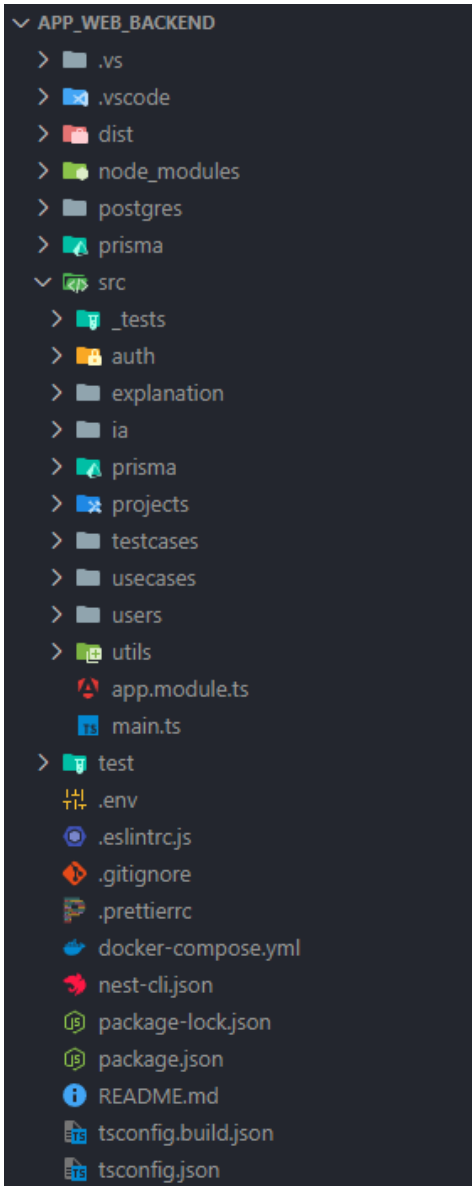


Figura 20: Entorno de Backend de la aplicación web

Por otro lado, en la **Figura 21**, se muestra el lado del frontend, que se realizó utilizando NextJs, aquí se definió la comunicación con el backend, los componentes de las diferentes interfaces, los contextos para lograr tener persistencia de datos y funciones de las principales entidades de la aplicación, las páginas que se muestran al usuario final y finalmente diversos archivos de configuración en donde se define la estructura y características del servidor de frontend.

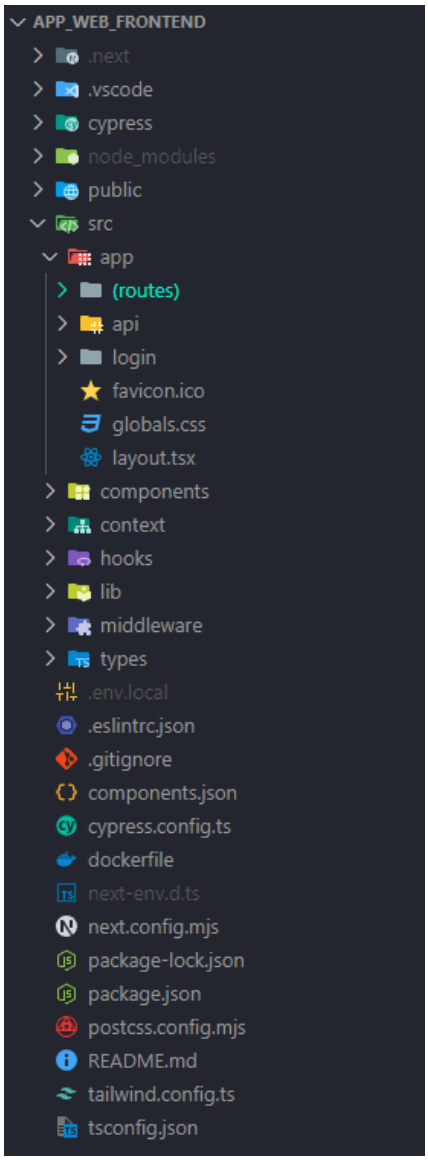


Figura 21: Entorno de Frontend de la aplicación web

Por otro lado, la base de datos se diseñó en Postgres. Desde la **Figura 22** hasta la **Figura 25**, se presenta el esquema elaborado con prisma para la aplicación web.

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model Entity {
  id            String  @id @default(uuid())
  firstName     String
  lastName      String
  imageEntity   String?
  user          User?
}

model User {
  id            String    @id @default(uuid())
  email         String    @unique
  password      String
  status        Boolean   @default(true)
  entityId      String    @unique
  entity        Entity    @relation(fields: [entityId], references: [id])
  roleId        String
  role          Role      @relation(fields: [roleId], references: [id])
  projects      ProjectMember[]
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt
}

enum Roles {
  Tester
  Administrator
}
```

Figura 22: Esquema de la Base de Datos en Prisma.

```
model Role {
  id    String @id @default(uuid())
  name  Roles @unique
  users User[]
}

enum ProjectRoles {
  Owner
  Editor
  Viewer
}

model Project {
  id          String      @id @default(uuid())
  code        String      @unique
  name        String
  description String?
  image       String?
  useCases    UseCase[]   @relation("ProjectUseCases")
  members     ProjectMember[]
  createdAt   DateTime    @default(now())
  updatedAt   DateTime    @updatedAt
}

model ProjectMember {
  id         String       @id @default(uuid())
  userId     String
  user       User         @relation(fields: [userId], references: [id], onDelete: Cascade)
  projectId  String
  project    Project      @relation(fields: [projectId], references: [id], onDelete: Cascade)
  role       ProjectRoles
  createdAt  DateTime     @default(now())
  updatedAt  DateTime     @updatedAt

  @@unique([userId, projectId])
}
```

Figura 23: Esquema de la Base de Datos en Prisma.

```
model UseCase {
  id              String     @id @default(uuid())
  code            String
  name            String
  description     String
  preconditions   String
  postconditions  String
  mainFlow        String
  alternateFlows  String?
  testCases       TestCase[] @relation("UseCaseTestCases")
  projectId       String
  project         Project    @relation("ProjectUseCases", fields: [projectId], references: [id], onDelete: Cascade)
  createdAt       DateTime   @default(now())
  updatedAt       DateTime   @updatedAt
}

model TestCase {
  id              String      @id @default(uuid())
  code            String
  name            String
  description     String
  steps           String
  inputData       String
  expectedResult  String?
  useCaseId       String
  useCase         UseCase     @relation("UseCaseTestCases", fields: [useCaseId], references: [id], onDelete: Cascade)
  explanation     Explanation?@relation("TestCaseExplanation")
  createdAt       DateTime    @default(now())
  updatedAt       DateTime    @updatedAt
}
```

Figura 24: Esquema de la Base de Datos en Prisma.

```
model Explanation {
  id          String    @id @default(uuid())
  summary     String
  details     String
  testCaseId  String    @unique
  testCase    TestCase  @relation("TestCaseExplanation", fields: [testCaseId], references: [id], onDelete: Cascade)
  createdAt   DateTime  @default(now())
  updatedAt   DateTime  @updatedAt
}
```

Figura 25: Esquema de la Base de Datos en Prisma.

En la **Figura 26** se presenta el diagrama Entidad-Relación, que ilustra de forma visual la estructura del modelo de la base de datos en PostgresDB utilizada en este TIC. Este diagrama se alinea con la estructura previamente definida en Prisma.

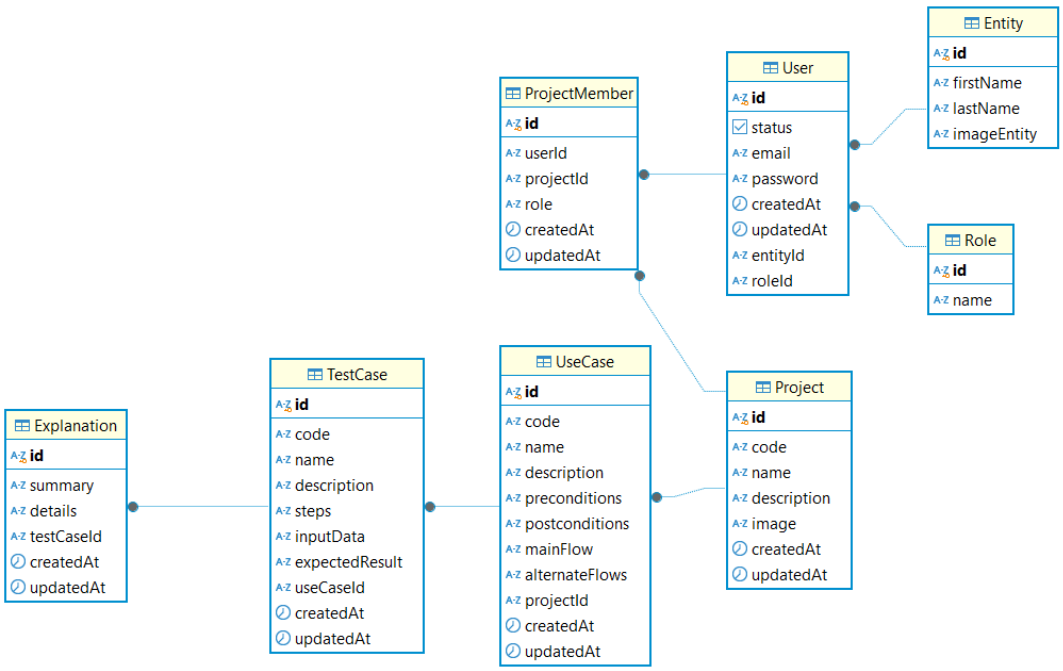


Figura 26: Diagrama Entidad Relación de la Base de Datos en Postgres.

El proceso de construcción del sistema se desarrolló en base a un plan de iteraciones separado en 4 sprints, para conocer todo el plan debe dirigirse al **Anexo 9**.

Además, el desarrollo de la fase de construcción y la metodología RAD se puede visualizar en el **Anexo 10**, no obstante, el repositorio con el código fuente de la aplicación con sus entornos de backend y frontend se los muestra a continuación:

- Para el **Backend:** https://github.com/Computacion-UNL/testcasecraft/tree/backend
- Para el **Frontend:** https://github.com/Computacion-UNL/testcasecraft/tree/frontend

- **Desarrollo del Prompt.**

Se presenta el prompt establecido se envia a la IA en cada consulta a su API, misma en la que se utilizó una estructura ROSES: Role, Objetive, Scenario, Examples, Steps; este detalle del proceso para obtener el prompt ideal se lo puede ver en el **Anexo 7.**

"Eres un ingeniero de software de QA especializado en Pruebas de Software, certificado por ISTQB, con más de 10 años de experiencia. Tu objetivo es realizar un análisis riguroso de casos de uso y generar casos de prueba funcionales de alta calidad. Se te proporcionará un caso de uso que debes evaluar y, si es válido, generar casos de prueba funcionales para el mismo, siguiendo estos pasos:
Para validar el caso de uso, Verifica Si los parametros del caso de uso son una frase/palabra con significado o si es solo un conjunto de letras sin sentido.
Establece "response" como true si tiene sentido y como false si no. Ignora las etiquetas HTML
Ejemplos:
1. "ingresar al sistema" → "response": true
2. "asdfghjkl" → "response": false
3. "permiso de acceso" → "response": true
4. "qwertyuiop" → "response": false
Cuando "response" sea false, No generes casos de prueba, Llena la sección "error" con las mejoras necesarias para el caso de uso.
Cuando "response" sea true Genera casos de prueba funcionales utilizando las técnicas de ISTQB:
- Partición de equivalenci, Análisis de valores límite, Transición de estados, Tablas de decisión
Para cada caso de prueba funcional:
Proporciona una explicación detallada en "explanationDetails" con los pasos de generación, el razonamiento y las técnicas aplicadas en el proceso de generación del caso de prueba.
Ejemplo de explanationDetails:
- Partición de equivalencia: En la explicación, se detallaría cómo se identificaron los valores válidos e inválidos, cómo se agruparon en clases de equivalencia, y cómo estos se utilizaron para generar casos de prueba representativos de cada clase.
- Análisis de valores límite: La explicación debería detallar cómo se seleccionaron estos valores límite (por ejemplo, el valor mínimo válido, el valor justo por debajo del mínimo, el valor máximo válido, el

valor justo por encima del máximo) y cómo se utilizaron para crear casos de prueba que verifiquen el comportamiento del sistema en estos puntos críticos.
- Transición de estados: La explicación debería describir cómo se identificaron los diferentes estados del sistema y las transiciones entre ellos. Se busca que se detalle cómo se crearon casos de prueba para verificar:Transiciones válidas entre estados, Intentos de transiciones inválidas, Secuencias de transiciones que cubran todos los estados y transiciones posibles, Condiciones iniciales y finales para cada estado
- Tablas de decisión: Se espera una explicación de cómo se identificaron las condiciones y acciones relevantes para el caso de uso. La explicación debería detallar: Cómo se construyó la tabla de decisión, listando todas las combinaciones posibles de condiciones, Cómo se determinaron las acciones esperadas para cada combinación de condiciones, Cómo se generaron casos de prueba a partir de cada fila de la tabla de decisión, Cómo se aseguró que todas las combinaciones relevantes fueran cubiertas.
5. Genera la respuesta en formato JSON siguiendo la estructura proporcionada:

```
{
  "response": boolean,
  "error": {
    "message": string,
    "improvements": [
      {
        "issue": string,
        "suggestion": string,
        "example": string
      }
    ]
  },
  "testCases": [
    {
      "code": string,
      "name": string,
      "description": string,
      "steps": string,
      "inputData": string,
      "expectedResult": string,
      "explanationSummary": string,
      "explanationDetails": string
    }
  ]
}
```

6. Asegúrate de que todas las explicaciones sean detalladas, claras y técnicas, basadas en ISTQB.

7. Responde únicamente con el formato JSON especificado y en español.
A continuación, el caso de uso a evaluar:”

- **Consumo de IA con el Prompt.**

En la **Figura 27**, se muestra la sección de código que consuma la IA “command-r”, mediante la librería de Cohere que simplifica la cantidad de líneas de código para la comunicación con la API que se necesita para obtener respuesta mediante el prompt especificado.

```
import { Injectable } from '@nestjs/common';
import { Usecase } from 'src/usecases/entities/usecase.entity';
import { prompt, api_key } from '../utils/constants';
import { CohereClient } from "cohere-ai";

const cohere = new CohereClient({
  token: api_key,
});
@Injectable()
export class IaService {

  async getCompletion(useCaseData: Partial<Usecase>) {
    const useCaseJson = JSON.stringify(useCaseData, null, 2);
    const promptData = `${prompt} \n${useCaseJson}`;

    console.log(useCaseJson)
    try {
      const response = await cohere.v2.chat({
        model: 'command-r',
        messages: [
          {
            role: 'user',
            content: promptData,
          },
        ],
      });
      return response.message.content[0].text
    } catch (error) {
      throw new Error('Error al obtener respuesta de Cohere AI');
    }
  }
}
```

Figura 27: Consumo de IA.

6.1.4. Fase de Transición.

La fase de transición se focaliza en el desarrollo de pruebas previo al despliegue de la aplicación, por tanto, para contemplar tanto el entorno del backend y frontend, se diseñaron pruebas unitarias y funcionales respectivamente.

- **Pruebas Unitarias**

A continuación, en la **Tabla 14**, se presenta el resumen de las pruebas unitarias realizadas en la herramienta JEST. (Para más detalle revisar el **Anexo 9**).

Tabla 14: Pruebas Unitarias de la aplicación web.

Código	Componente	Descripción de la Prueba	Resultado
PU-01	Iniciar Sesión	Comprueba la autenticación de un usuario con credenciales válidas. La respuesta del servidor debe de tener un código 200 y devolver una comprobación de credenciales correctas	Aprobado
PU-02	Iniciar Sesión	Comprueba la autenticación de un usuario con credenciales inválidas. La respuesta del servidor debe de tener un código 401 y devolver un mensaje de 'No se encontró una cuenta con ese correo electrónico'	Aprobado
PU-03	Recuperar Contraseña	Comprueba el envío de una OTP para la recuperación de la contraseña de un usuario. La respuesta del servidor debe de tener un código 200 y devuelve un token con el OTP	Aprobado
PU-04	Validar OTP	Valida la OTP ingresada por el usuario para recuperar su contraseña. La respuesta del servidor debe de tener un código 200.	Aprobado
PU-05	Registrarse	Un usuario se pueda registrar con datos válidos. La respuesta del servidor debe de tener un código 200 y devuelve un mensaje de 'Registrado correctamente'	Aprobado
PU-06	Cambiar rol de usuario	Valida que un administrador pueda cambiar el rol de un usuario. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Usuario Actualizado'	Aprobado
PU-07	Listar Usuarios	Comprueba que se devuelva una lista de usuarios registrados en el sistema. La respuesta del servidor debe de tener un código 200 y devuelve la lista de todos los usuarios'	Aprobado
PU-08	Crear Proyecto	Valida que un proyecto se cree correctamente con datos válidos. La respuesta del servidor debe de tener un código 200	Aprobado
PU-09	Actualizar Proyecto	Valida que un proyecto se actualice correctamente con datos válidos. La respuesta del servidor debe de tener un código 200 y devuelve un mensaje de 'Proyecto Actualizado Correctamente'	Aprobado
PU-010	Eliminar Proyecto	Comprueba la eliminación lógica exitosa de un proyecto. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Proyecto Eliminado Correctamente.	Aprobado
PU-011	Listar Proyectos	Comprueba que se devuelva una lista de proyectos de un usuario registrado en el sistema. La respuesta del servidor debe de tener un código 200 y devolver la lista de proyectos del usuario	Aprobado
PU-012	Compartir Proyecto	Comprueba que un código de proyecto exista y a este, un usuario sin registro previo al proyecto se una. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Se ha unido al proyecto correctamente'	Aprobado

Código	Componente	Descripción de la Prueba	Resultado
PU-013	Crear Caso de Uso	Valida que un caso de uso se cree correctamente con datos válidos dentro de un proyecto. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Uso Creado Correctamente'	Aprobado
PU-014	Actualizar caso de uso	Valida que un caso de uso se actualice correctamente con datos válidos. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Uso Actualizado Correctamente'.	Aprobado
PU-015	Eliminar caso de uso	Comprueba la eliminación lógica exitosa de un caso de uso. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Uso Eliminado Correctamente'	Aprobado
PU-016	Listar casos de uso	Comprueba que se devuelva una lista de casos de uso de un proyecto en el sistema. La respuesta del servidor debe de tener un código 200 y devolver la lista de casos de uso del proyecto	Aprobado
PU-017	Generar casos de prueba funcionales	Valida que se generen N casos de prueba funcionales a partir del id de un caso de uso que se envíe. La respuesta del servidor debe de tener un código 200 y devolver la lista de casos de prueba funcionales generados	Aprobado
PU-018	Actualizar caso de prueba funcional	Valida que se actualice un caso de prueba funcional con datos válidos. La respuesta del servidor debe de tener un código 200 y devolver un mensaje 'Caso de Prueba Funcional Actualizado Correctamente'	Aprobado
PU-019	Eliminar caso de prueba funcional	Comprueba la eliminación lógica exitosa de un caso de prueba funcional. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Prueba Funcional Eliminado Correctamente'	Aprobado
PU-020	Listar casos de prueba funcionales	Comprueba que se devuelva una lista de casos de prueba funcionales de un caso de uso en el sistema. La respuesta del servidor debe de tener un código 200 y devolver la lista de casos de prueba funcionales de un caso de uso	Aprobado

- **Pruebas Funcionales**

En la **Tabla 15**, se presenta el resumen de las pruebas de funcionalidad o funcionales ejecutadas en la herramienta CYPREES. (Para más detalle revisar el **Anexo 10**).

Tabla 15: Pruebas Funcionales de la aplicación web.

Código	Componente	Resultado
PF-01	Iniciar Sesión	Aprobado
PF-02	Iniciar Sesión – Credenciales Inválidas	Aprobado
PF-03	Recuperar Contraseña	Aprobado
PF-04	Registrarse	Aprobado
PF-05	Crear Proyecto	Aprobado
PF-06	Actualizar Proyecto	Aprobado

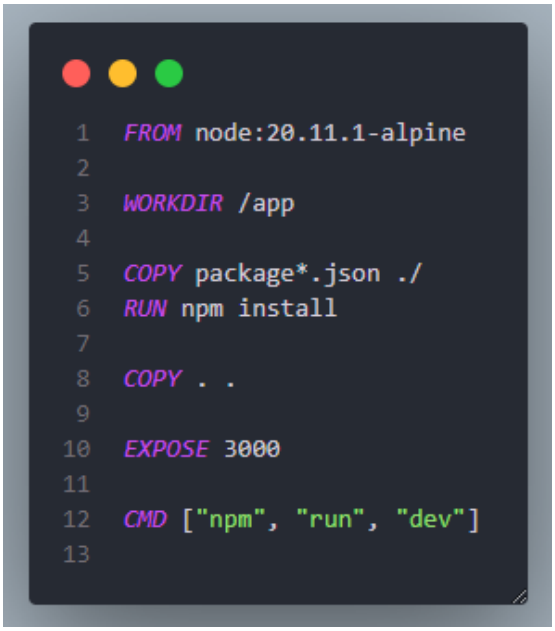
Código	Componente	Resultado
PF-07	Eliminar un Proyecto	Aprobado
PF-08	Listar Proyectos	Aprobado
PF-09	Creación de un caso de uso	Aprobado
PF-010	Actualizar caso de uso	Aprobado
PF-011	Eliminar Caso de Uso	Aprobado
PF-012	Generación de casos de prueba funcionales	Aprobado
PF-013	Eliminar caso de prueba funcional	Aprobado
PF-014	Compartir Proyecto	Aprobado
PF-015	Unirse a Proyecto	Aprobado
PF-016	Activar Usuario	Aprobado
PF-017	Desactivar Usuario	Aprobado
PF-018	Cambiar Rol de Usuario	Aprobado
PF-019	Generar PDF de Proyecto	Aprobado

Las pruebas denotan un porcentaje del 100% de aprobación tanto en unitarias como en funcionales, lo que demuestra un control propicio de errores y manejo correcto de las funcionalidades básicas del software. Este acierto no dice que todo fue perfecto, al contrario, los errores se hicieron presentes durante todo el desarrollo y cada iteración del plan, sin embargo, se controló y corrigió cada inconveniente que surgió para obtener un resultado ideal.

- **Despliegue**

Una vez realizadas las pruebas de software, se puede dar paso a su despliegue en un servidor remoto, para que, con ello, los usuarios puedan adentrarse y hacer uso de la aplicación otorgando así una forma de validar la respuesta de la misma, ante la interacción con individuos reales, dejando de manera libre un entorno de producción para el sistema y que así se prosiga con el objetivo secuencial del trabajo integración curricular.

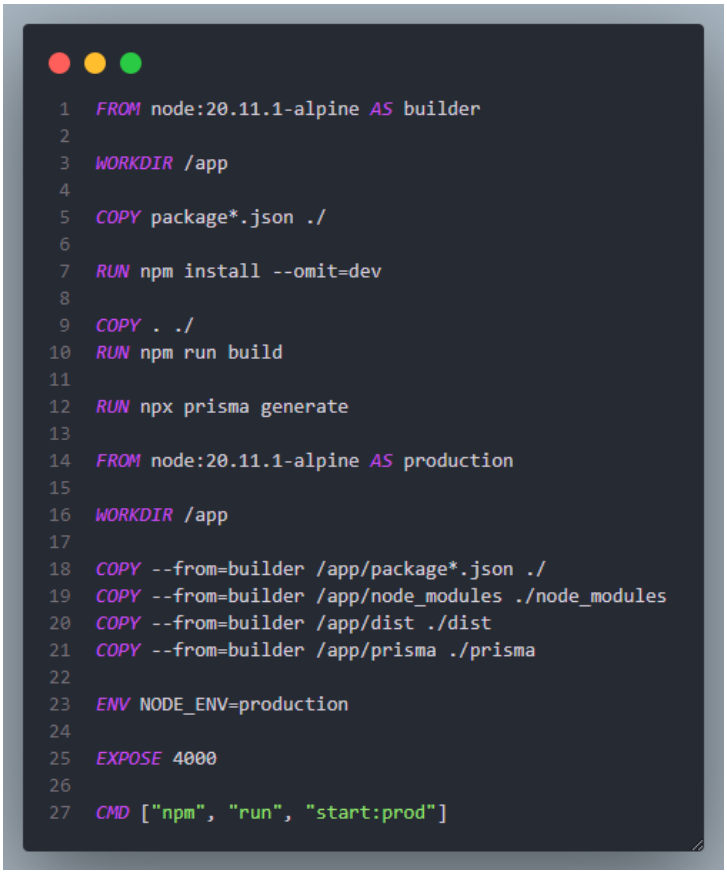
Por ello, para el despliegue se utilizó contenedores de docker separados para cada entorno, por lo que se inició “dockerizando” el servicio de frontend en un archivo dockerfile ubicado en la raíz del proyecto. En la **Figura 28** se muestra el archivo dockerfile del servicio de frontend.



```
FROM node:20.11.1-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "run", "dev"]

```

Figura 28: Archivo de configuración Docker del frontend.

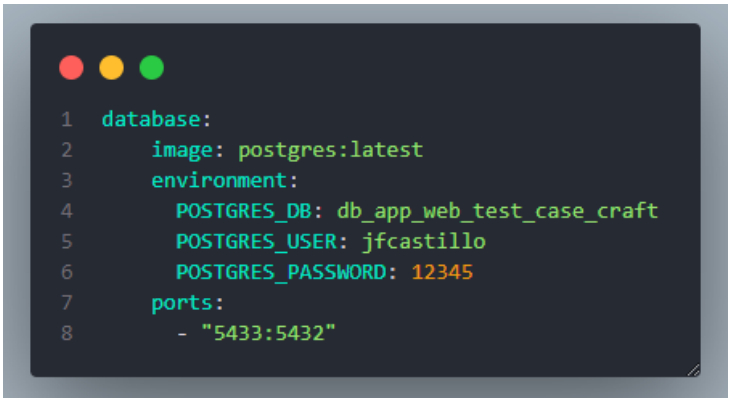
En la **Figura 29** se muestra el archivo dockerfile del servicio de backend.



```
FROM node:20.11.1-alpine AS builder

WORKDIR /app

COPY package*.json ./

RUN npm install --omit=dev

COPY . ./
RUN npm run build

RUN npx prisma generate

FROM node:20.11.1-alpine AS production

WORKDIR /app

COPY --from=builder /app/package*.json ./
COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app/dist ./dist
COPY --from=builder /app/prisma ./prisma

ENV NODE_ENV=production

EXPOSE 4000

CMD ["npm", "run", "start:prod"]
```

Figura 29: Archivo de configuración Docker del backend.

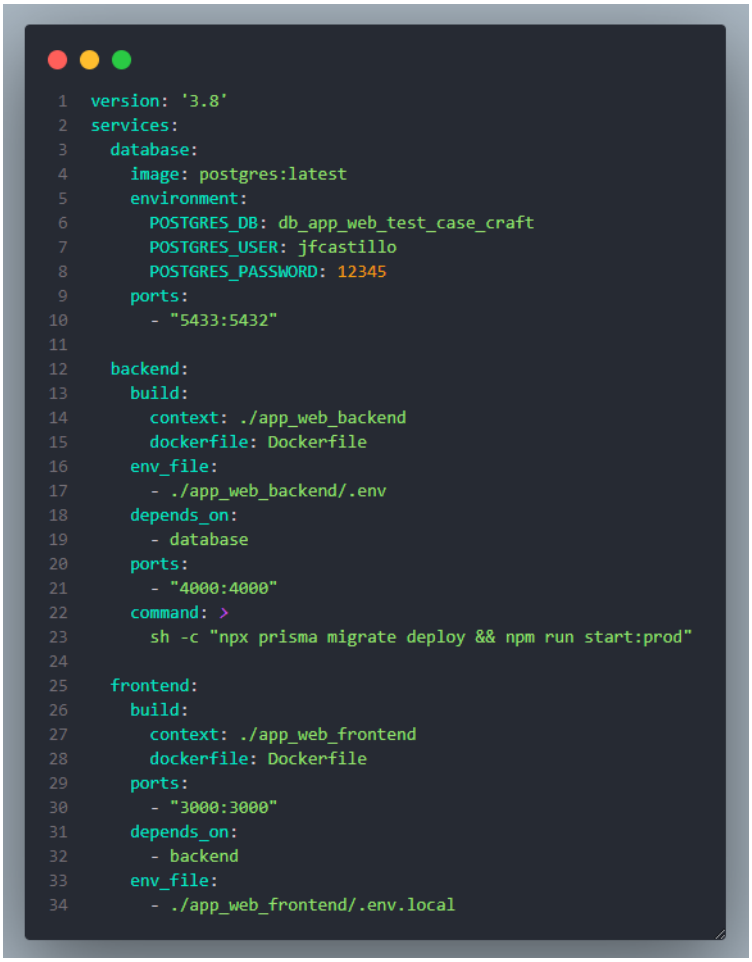
En la **Figura 30** se presenta el archivo de configuración de Docker para la base de datos.



```yaml
database:
    image: postgres:latest
    environment:
      POSTGRES_DB: db_app_web_test_case_craft
      POSTGRES_USER: jfcastillo
      POSTGRES_PASSWORD: 12345
    ports:
      - "5433:5432"
```

Figura 30: Archivo de configuración Docker para la base de datos.

Para vincular y ejecutar todos los servicios se usó Docker Compose, el archivo de configuración se muestra a continuación en la **Figura 31**.



```yaml
version: '3.8'
services:
  database:
    image: postgres:latest
    environment:
      POSTGRES_DB: db_app_web_test_case_craft
      POSTGRES_USER: jfcastillo
      POSTGRES_PASSWORD: 12345
    ports:
      - "5433:5432"

  backend:
    build:
      context: ./app_web_backend
      dockerfile: Dockerfile
    env_file:
      - ./app_web_backend/.env
    depends_on:
      - database
    ports:
      - "4000:4000"
    command: >
      sh -c "npx prisma migrate deploy && npm run start:prod"

  frontend:
    build:
      context: ./app_web_frontend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    depends_on:
      - backend
    env_file:
      - ./app_web_frontend/.env.local
```

Figura 31: Archivo Docker Compose de los servicios de la aplicación web.

Luego se ejecutó estos contenedores en los servidores de la carrera de computación, mediante el comando:

docker-compose up -d

Finalmente se tuvo como resultado la aplicación web funcionando en la siguiente ruta: ***https://computacion.unl.edu.ec/tcc/.*** En la **Figura 32** se muestra la aplicación web desplegada en la ruta descrita dentro de los servidores de la carrera de computación.

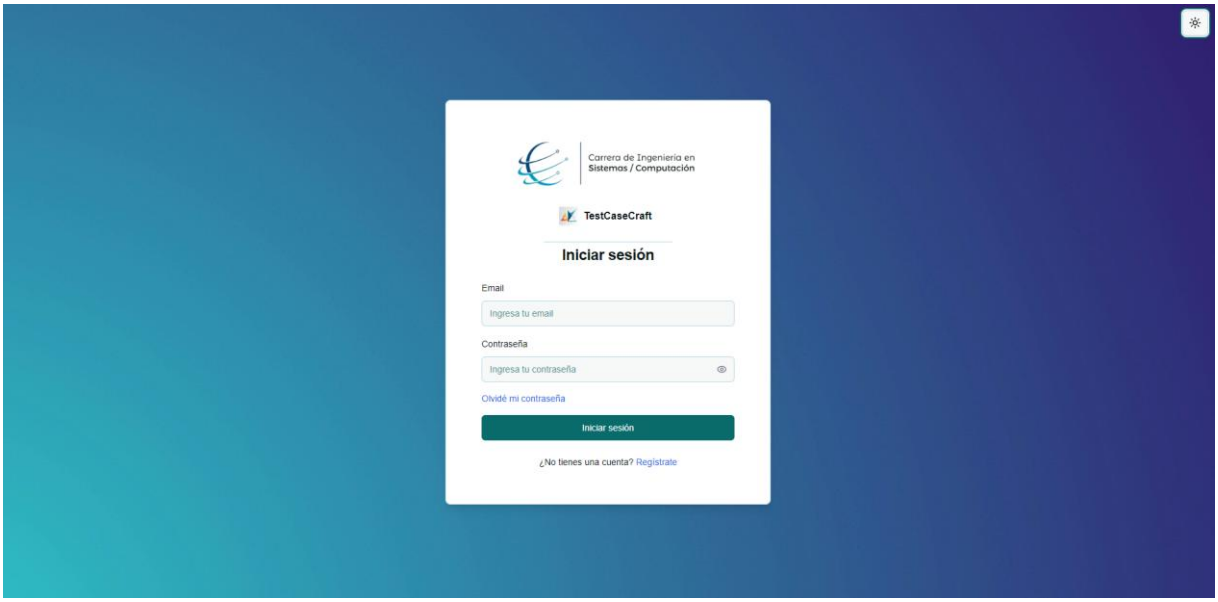


Figura 32: Aplicación Web desplegada.

Posterior a las pruebas y despliegue de la aplicación se muestra un manual de usuario para que las personas que utilicen el sistema posean una guía de cómo deberían hacer uso de la aplicación, así estos individuos, se capacitan y aprovechan el software, en el **Anexo 11** se muestra el manual con sus detalles.

6.2. Objetivo 2: Evaluar la percepción de utilidad de la herramienta desarrollada entre los estudiantes, utilizando como base ciertos lineamientos del modelo TAM (Modelo de Aceptación Tecnológica).

A continuación, se describe los resultados obtenidos en la ejecución del objetivo específico dos, de acuerdo con las fases definidas.

6.2.1. Fase 1: Planificación de evaluación

La planificación de evaluación se muestra en la **Tabla 16**, en ella se detalla las subsecciones a seguir para evaluar la percepción de utilidad de la aplicación de diseño de casos de prueba funcionales a partir de casos de uso.

Tabla 16: Planificación de Evaluación.

Sección	Descripción	Duración	Entregable
1	Elaboración de instrumento de evaluación	3 día	Cuestionario Base
2	Elaboración de escala de Likert	1 día	Escala Definida
3	Determinar muestra	2 días	Documento de criterios de selección
4	Traspaso a formulario en Google Forms	1 día	Formulario digital
5	Evaluación de Instrumento	7 días	Resultados recopilados

- **Elaboración de instrumento de evaluación**

El cuestionario realizado con respecto a la variable de percepción de utilidad del modelo TAM se describe en la **Tabla 17.**

Tabla 17: Instrumentos de evaluación para medir la variable de percepción de utilidad del modelo TAM.

Identificador	Pregunta
PU1	¿El sistema reduce significativamente el tiempo necesario para crear casos de prueba funcionales?
PU2	¿El sistema me ayuda a identificar escenarios de prueba que podría haber pasado por alto manualmente?
PU3	¿Los casos de prueba generados son relevantes para los casos de uso proporcionados?
PU4	¿La calidad de los casos de prueba generados es comparable o superior a los creados manualmente?
PU5	¿El sistema proporciona una cobertura adecuada de los casos de uso?
PU6	¿El uso de este sistema mejora la calidad de mi trabajo en Testing?
PU7	¿Esta herramienta me ayuda a realizar mi trabajo de manera más efectiva?
PU8	¿El sistema se integra bien en mi flujo de trabajo actual de pruebas?
PU9	¿Recomendaría este sistema a otros profesionales del Testing?
PU10	¿Es útil la generación automática de casos de prueba funcionales a partir de casos de uso?
PU11	¿Es útil el formato y estructura de los casos de prueba funcionales generados?
PU12	¿Es útil la personalización de los casos de prueba funcionales generados?

Identificador	Pregunta
PU13	¿En qué porcentaje estima que el sistema ha reducido el tiempo de creación de casos de prueba?
PU14	¿Qué porcentaje de los casos de prueba generados ha requerido de modificación manual?
PU15	¿Qué aspectos del sistema podrían mejorarse?

- **Elaboración de escala de Likert**

Este cuestionario será evaluado con la escala de Likert la cual describe cinco niveles de respuestas, la **Tabla 18** presenta la escala a evaluar de la pregunta 1 a la 12 debido a su respuesta cualitativa que va desde Totalmente en desacuerdo a Totalmente de acuerdo, lo cual plantea un valor cuantitativo en esta escala de 0 a 5.

Tabla 18: Escala de Likert

Respuesta	Valor
Totalmente en desacuerdo	0 – Nada
En desacuerdo	1 – Bajo
Ni de acuerdo ni en desacuerdo	2 – Normal
De acuerdo	3 – Medio
Totalmente de acuerdo	4 – Alto

En la **Tabla 19,** se presenta una escala Likert ponderada por porcentaje para evaluar la pregunta 13 y 14, esto debido al tipo de respuesta que reflejan dichas preguntas, planteando un mapeo de escala Likert a un rango porcentual.

Tabla 19: Escala de Likert ponderada por porcentaje.

Respuesta		Valor
Pregunta 13	**Pregunta 14**	
0-20%	81-100%	0 – Nada
21-40%	61-80%	1 – Bajo
41-60%	41-60%	2 – Normal
61-80%	21-40%	3 – Medio
81-100%	0-20%	4 – Alto

- **Determinar muestra**

La muestra de estudio es por conveniencia pues comprendió a 21 estudiantes de octavo y noveno ciclo del itinerario de ingeniería de software de la carrera de computación, considerando que estos participantes cuentan con formación y experiencia previa en pruebas de software (testing), lo cual es fundamental para una evaluación significativa de la percepción de utilidad del sistema.

Posteriormente, se transfirió la información de la encuesta a un formulario en Google Form para poder evaluar el instrumento con la muestra indicada, se puede visualizar el formulario en el siguiente enlace: https://docs.google.com/forms/d/e/1FAIpQLSew9G5FRba5KJfyG_c8Lt0Mo6PFg_6r-ogkNFtpSOCXsDrUdg/viewform?usp=sf_link

6.2.2. Fase 2: Visualización de resultados.

En total, 21 estudiantes participaron en la encuesta y respondieron satisfactoriamente todas las preguntas planteadas, como se muestra en la **Figura 33.**

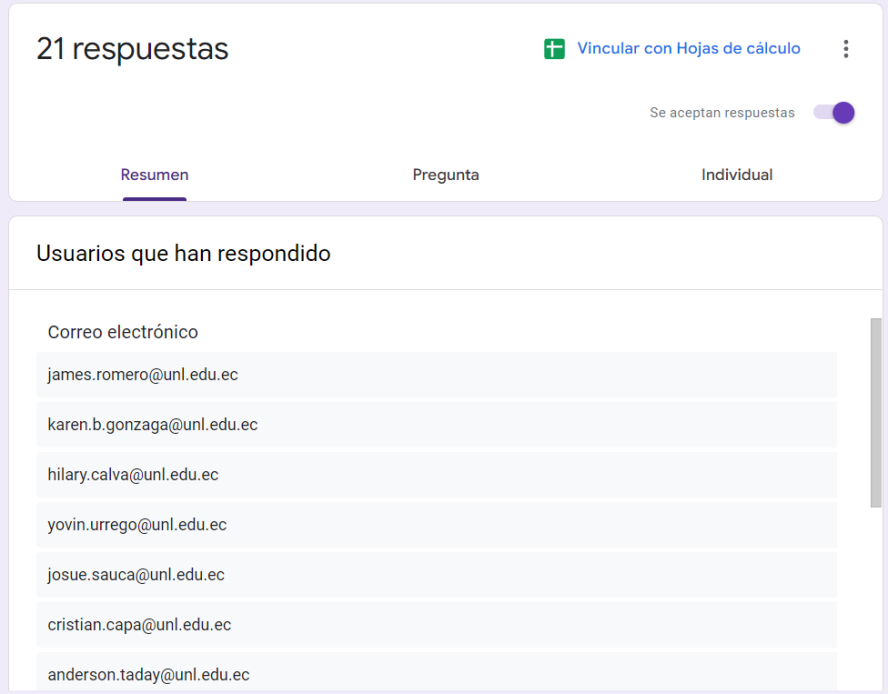


Figura 33: Encuesta de Evaluación de Percepción de Utilidad: Resultados.

En la **Tabla 20**, se presentan los resultados de la ejecución de los instrumentos de evaluación de la variable de percepción de utilidad de TAM de la pregunta 1 a la 12, mediante escala de Likert.

Tabla 20: Tabulación de resultados de acuerdo a la variable de percepción de utilidad.

Indicador	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Totalmente de acuerdo
P1	0%	0%	4.8%	33.3%	61.9%
P2	0%	0%	4.8%	47.6%	47.6%
P3	0%	0%	0%	61.9%	38.1%
P4	0%	0%	9.5%	52.4%	38.1%
P5	0%	0%	4.8%	38.1%	57.1%
P6	0%	0%	9.5%	28.6%	61.9%
P7	0%	0%	0%	33.3%	66.7%
P8	0%	0%	9.5%	28.6%	61.9%
P9	0%	0%	0%	42.9%	57.1%
P10	0%	0%	0%	19%	81%
P11	0%	0%	0%	38.1%	61.9%
P12	0%	0%	0%	23.8%	76.2%

En la **Tabla 21**, se presentan los resultados de la ejecución de los instrumentos de evaluación de la variable de percepción de utilidad de TAM de la pregunta 13 y 14, mediante escala de Likert ponderada en porcentaje.

Tabla 21: Tabulación de resultados de acuerdo a la variable de percepción de utilidad.

Indicador	0-20%	21-40%	41-60%	61-80%	81-100%
P13	0%	4.8%	14.3%	52.4%	28.6%
P14	52.4%	19%	9.5%	14.3%	4.8%

6.2.3. Fase 3: Análisis de Resultados.

A continuación, se presenta el análisis de los resultados obtenidos al evaluar la variable de percepción de utilidad del modelo TAM. En la **Tabla 22**, se muestra el análisis realizado por cada pregunta de la variable de percepción de utilidad.

Tabla 22: Resumen Análisis - Percepción de Utilidad

Indicador	Resultado	Análisis
PU1	El 61.9% de los encuestados están totalmente de acuerdo, y el 33.3% están de acuerdo en que el sistema reduce significativamente el tiempo necesario para crear casos de prueba funcionales.	Esto demuestra que más del 90% de los usuarios perciben una mejora en la eficiencia del proceso de creación de casos de prueba mediante el uso del sistema.
PU2	El 47.6% están totalmente de acuerdo, y otro 47.6% están de acuerdo en que el sistema ayuda a identificar escenarios de prueba que podrían haber pasado por alto manualmente.	Esto sugiere que más del 95% los usuarios valoran la capacidad del sistema para mejorar la exhaustividad en la identificación de escenarios de prueba.
PU3	El 38.1% están totalmente de acuerdo, y el 61.9% están de acuerdo en que los casos de prueba generados son relevantes para los casos de uso proporcionados.	Esto muestra que más del 90% de los usuarios consideran que los casos de prueba funcionales generados son pertinentes y alineados con los casos de uso.
PU4	El 38.1% están totalmente de acuerdo, y el 52.4% están de acuerdo en que la calidad de los casos de prueba generados es comparable o superior a los creados manualmente.	Esto indica que más del 90% de los usuarios obtuvieron una percepción positiva sobre la calidad de los casos de prueba funcionales generados automáticamente.
PU5	El 57.1% están totalmente de acuerdo, y el 38.1% están de acuerdo en que el sistema proporciona una cobertura adecuada de los casos de uso.	Esto muestra que más del 80% de los encuestados confían en la cobertura proporcionada por el sistema.
PU6	El 61.9% están totalmente de acuerdo, y el 28.6% están de acuerdo en que el uso del sistema mejora la calidad de su trabajo en Testing.	Más del 80% de los usuarios perciben que el sistema contribuye positivamente a la calidad de su trabajo.
PU7	El 66.7% están totalmente de acuerdo, y el 33.3% están de acuerdo en que la herramienta les ayuda a realizar su trabajo de manera más efectiva.	Esto evidencia que más del 90% los encuestados encuentran la herramienta útil para mejorar la eficacia en su trabajo.
PU8	El 61.9% están totalmente de acuerdo, y el 28.6% están de acuerdo en que el sistema se integra bien en su flujo de trabajo actual de pruebas.	Esto sugiere que más del 85% de los encuestados perciben una buena adaptabilidad del sistema al flujo de trabajo existente de los usuarios.
PU9	El 57.1% están totalmente de acuerdo, y el 42.9% están de acuerdo en que recomendarían el sistema a otros profesionales del Testing.	La alta recomendación indica que más del 90% de los usuarios tienen una aceptación positiva y satisfacción con el sistema.

Indicador	Resultado	Análisis
PU10	El 81% están totalmente de acuerdo, y el 19% están de acuerdo en que la generación automática de casos de prueba funcionales a partir de casos de uso es útil.	Esto indica que más del 95% considera útil esta funcionalidad, destacando su importancia en el proceso de Testing.
PU11	El 61.9% están totalmente de acuerdo, y el 38.1% están de acuerdo en que el formato y estructura de los casos de prueba funcionales generados son útiles.	Más del 90% de los usuarios valoran positivamente el formato y estructura de los casos de prueba funcionales generados.
PU12	El 76.2% están totalmente de acuerdo, y el 23.8% están de acuerdo en que la personalización de los casos de prueba funcionales generados es útil.	Al obtener más del 90% de aprobación esto destaca la importancia de la personalización en la generación de casos de prueba para satisfacer necesidades específicas.
PU13	El 52.4% de los encuestados estiman que el sistema ha reducido el tiempo de creación de casos de prueba entre un 61-80%, y el 28.6% estiman una reducción entre el 81-100%.	Esto muestra que más del 80% de los usuarios perciben una reducción entre el 61 al 100% en el tiempo requerido para crear casos de prueba funcionales, lo que evidencia la eficacia del sistema en la optimización del tiempo.
PU14	El 52.4% de los encuestados indican que solo el 0-20% de los casos de prueba generados han requerido modificación manual, mientras que un 19% señala modificaciones en el 21-40% de los casos.	Más del 70% de los usuarios encuentran que los casos de prueba generados requieren 0-40% de modificación, lo que sugiere que el sistema es efectivo en producir resultados de alta calidad que cumplen con las expectativas de los usuarios sin necesidad de ajustes significativos.

6.2.4. Fase 4: Ciclo de mejora.

En la **Figura 34,** se presenta la pregunta número 15 del instrumento de evaluación que, adquirió 21 respuestas, focalizadas en sugerencias o aspectos a mejorar que propongan los usuarios finales y así ampliar la utilidad de la app web.

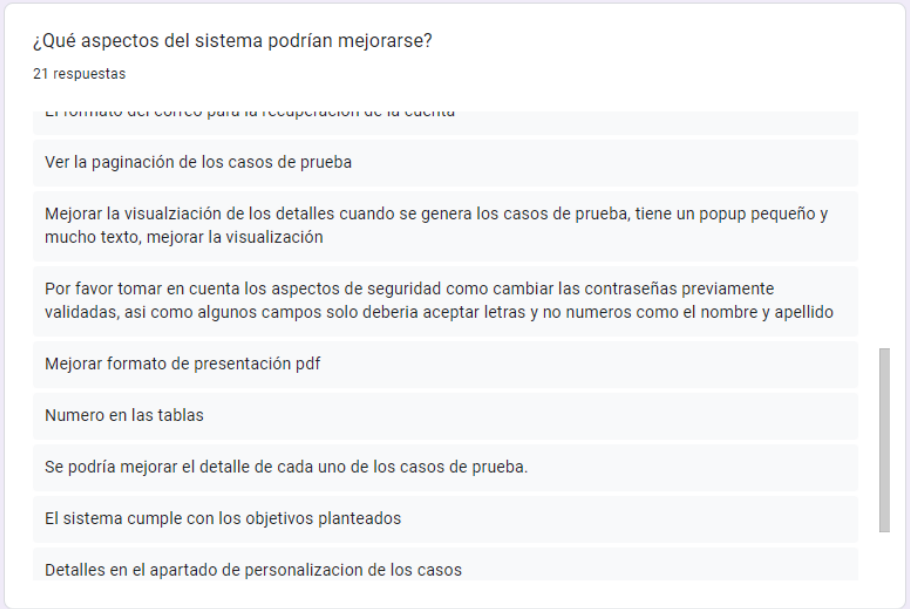


Figura 34: Pregunta de Ciclo de Mejora: Resultados

A partir de dichos aspectos, en la **Tabla 23** se determinan 6 aspectos a mejorar, que se consideran de urgencia moderada.

Tabla 23: Aspectos a mejorar - Ciclo de Mejora

Código	Aspectos a Mejorar
R1	Mejorar la interfaz de usuario de la sección de Explicaciones
R2	Mejorar la vista del correo para recuperar la contraseña
R3	Añadir paginación en las tablas
R4	Mejorar la visualización de los detalles cuando se genera los casos de prueba
R5	Mejorar aspectos de seguridad como cambiar las contraseñas previamente validadas, así como algunos campos solo debería aceptar letras y no números como el nombre y apellido
R6	Mejorar formato de presentación PDF

Las mejoras identificadas corresponden principalmente al Frontend, lo que permitió implementar los cambios de manera eficiente en un plazo de dos días. Esta rápida implementación demuestra que la arquitectura del software es flexible y robusta, facilitando modificaciones sin afectar la estructura base del sistema. A continuación, se presenta una comparación entre el estado inicial del sistema y las modificaciones realizadas según los requerimientos del usuario final.

En la **Figura 35** se presenta el cambio de la interfaz de Explicaciones, que proviene del aspecto a mejorar R1.

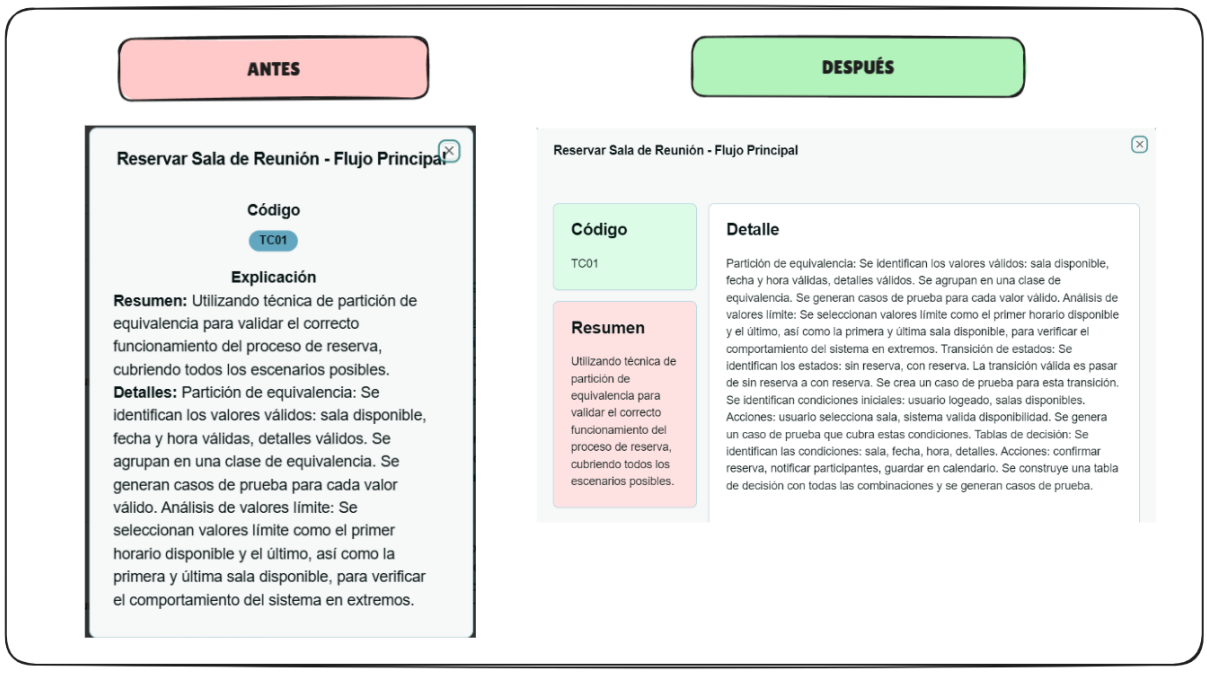


Figura 35: Antes y Después de R1.

En la **Figura 36** se presenta el cambio del correo para recuperar contraseña, que proviene del aspecto a mejorar R2.

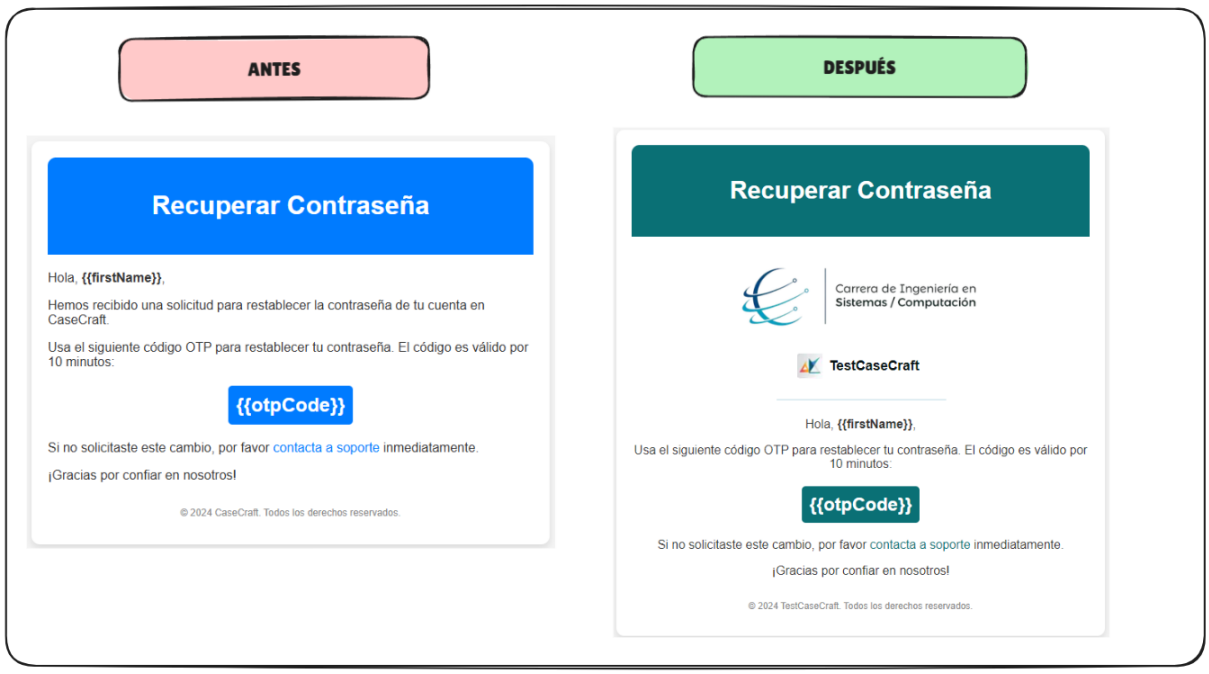


Figura 36: Antes y Después de R2.

En la **Figura 37** se presenta el cambio a tener paginación en las tablas, que proviene del aspecto a mejorar R3.

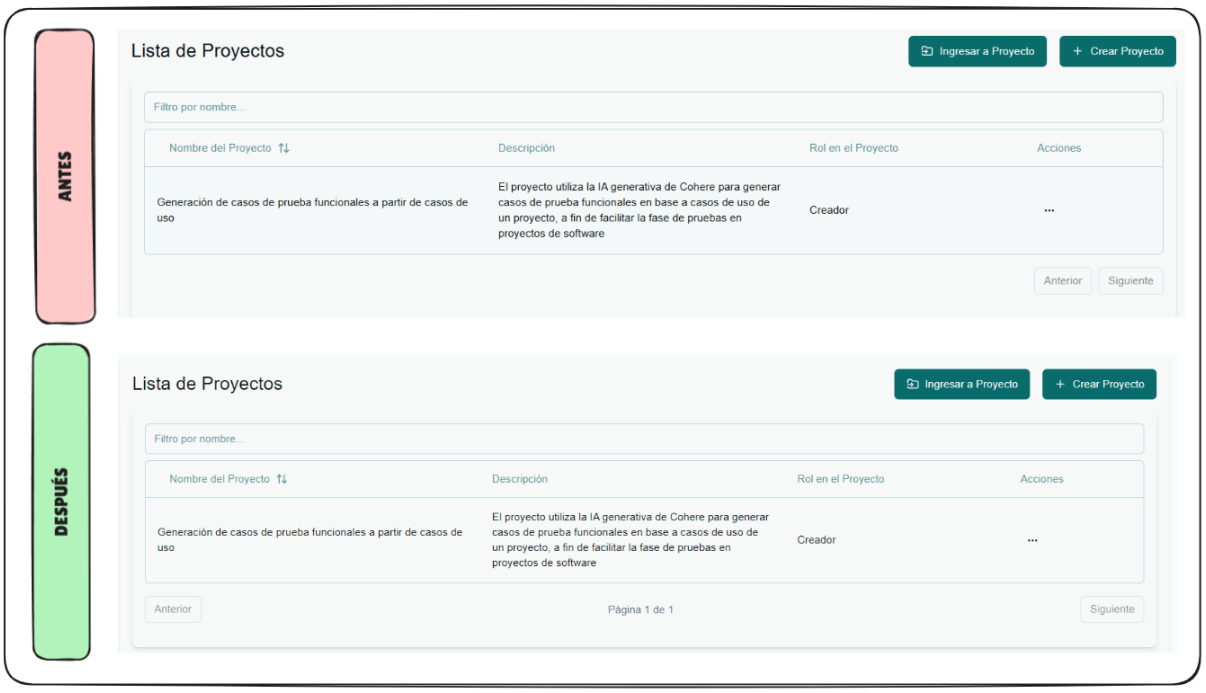


Figura 37: Antes y Después de R3.

En la **Figura 38** se presenta el cambio de la interfaz de detalles del caso de prueba funcional cuando es generado, que proviene del aspecto a mejorar R4.

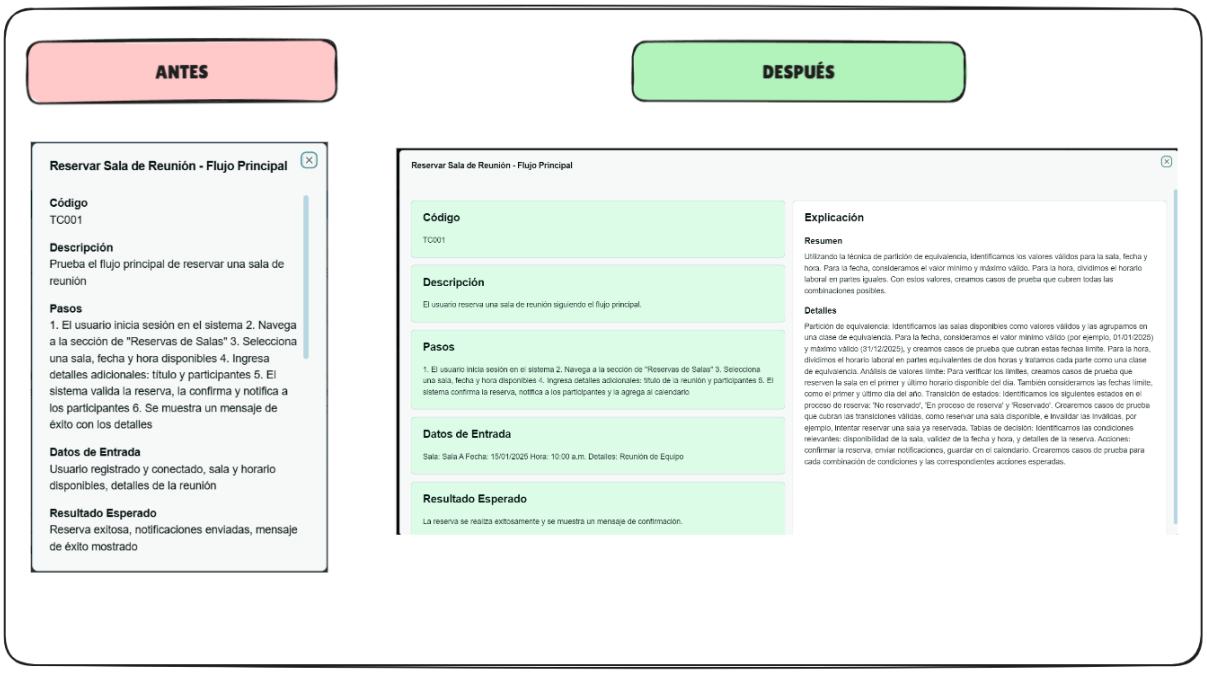


Figura 38: Antes y Después de R4.

En la **Figura 39** se presenta el cambio de la interfaz para que pueda cambiar contraseña el usuario desde la configuración de su perfil, que proviene del aspecto a mejorar R5.

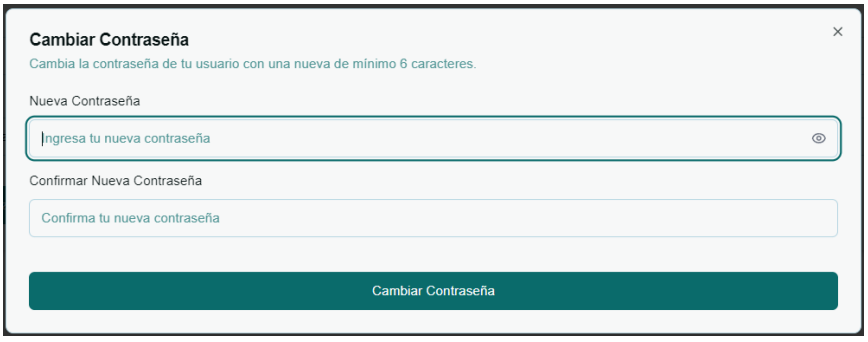


Figura 39: Después de R5.

En la **Figura 40** se presenta el cambio formato del PDF que se exporta del proyecto, que proviene del aspecto a mejorar R6.

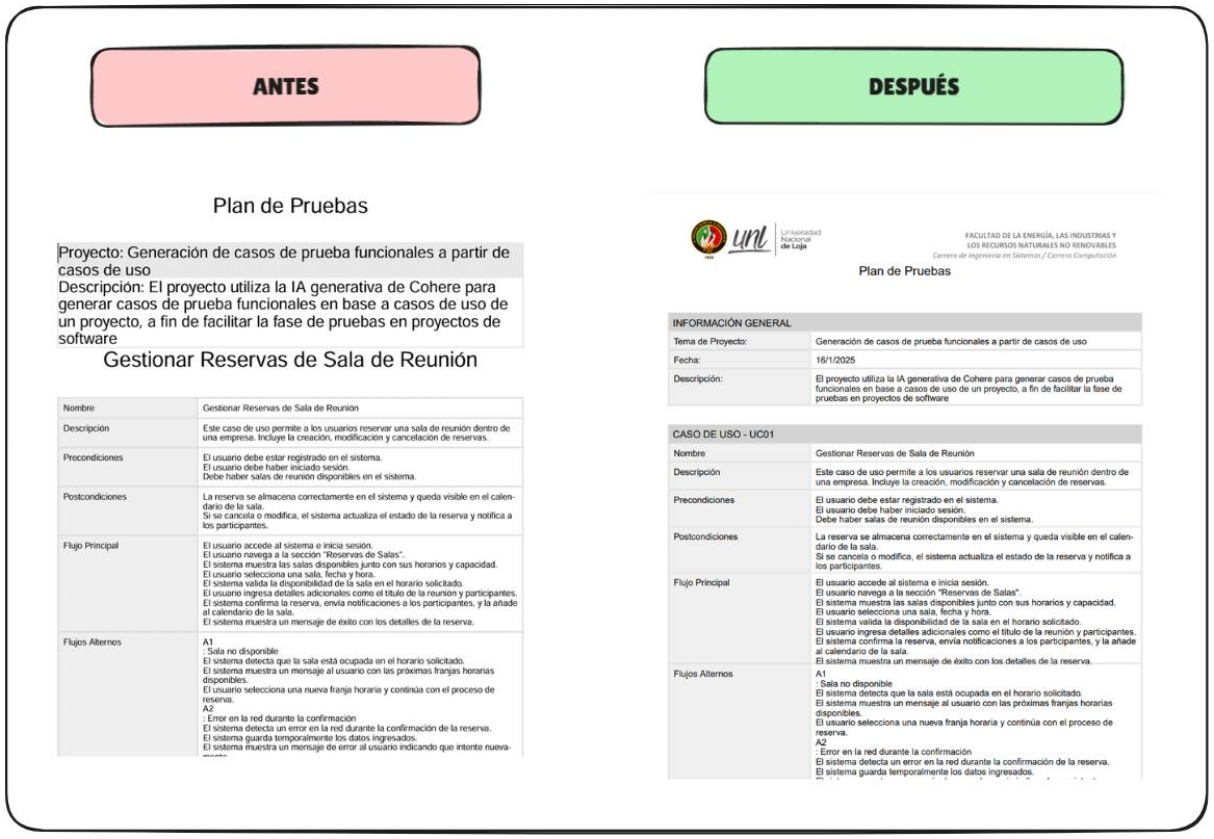


Figura 40: Antes y Después de R6.

7. Discusión

7.1 Primer objetivo: Desarrollar una herramienta de software que utilice la API de una IA para diseñar casos de prueba funcionales a partir de casos de uso, utilizando la metodología RAD (Desarrollo Rápido de Aplicaciones).

Este objetivo se centró en el desarrollo de una aplicación que automatiza la generación de casos de prueba funcionales a partir de casos de uso, integrando inteligencia artificial y aplicando la metodología RAD.

El desarrollo de una aplicación web con la combinación de inteligencia artificial y la metodología RAD ofrece una solución efectiva para la generación de casos de prueba funcionales, pues la decisión de adoptar dicha metodología permitió iteraciones rápidas y ajustes continuos basados en retroalimentación frecuente, lo cual fue fundamental para responder a las necesidades cambiantes de los usuarios, esto concuerda con Campaña [4], que presenta una metodología con lineamientos libres que aporta a la agilidad de un proyecto que depende mucho de un tiempo limitado, mientras que la IA aportó a obtener los mejores resultados posibles de los casos de uso que se le propicien y controlando la respuesta que provenga de este, manteniendo un formato y formalidad que mejoraría el trabajo de los usuarios.

El desafío de recopilar los requisitos del sistema, se le dio cobertura mediante entrevistas con potenciales usuarios y la aplicación de la técnica de Brainstorm, lo que ayudó a alinear los objetivos del proyecto con las expectativas de los clientes, incluyendo también, la definición de casos de uso que sirvieron como base para la generación de casos de prueba, determinando en gran medida las funcionalidades que se requiere para un usuario final, lo que concuerda con la estructura base que debe poseer un caso de prueba, temática tratada en las clases de calidad de software impartidas por el Ing. Francisco Álvarez de la UNL.

El estilo arquitectónico cliente-servidor, separado por monolitos con responsabilidades especificas (vista y servicios) aplicados en la aplicación web, facilitó la integración con tecnologías para backend y frontend, pues, para el backend, se optó por NestJS, aprovechando librerías como Nodemailer para la gestión de correos electrónicos y la inclusión de API Keys en las variables de entorno para controlar la conexión con la IA, por otro lado, el frontend se desarrolló utilizando Next.js, complementado con Next Auth para la autenticación de usuarios y los componentes de Shadcn para agilizar el desarrollo de la interfaz. La conexión entre estos entornos se le dio cobertura mediante una API REST, facilitando la integración con tecnologías como la API de Cohere para la IA, también, se almacena todos los datos necesarios con ayuda de Prisma, pues mediante este ORM se maneja una base de datos

PostgreSQL simplificando la gestión y asegurando la consistencia en el control de la información.

En la fase de codificación surgieron dificultades en el manejo de respuestas asíncronas, la interpretación de los resultados generados por la IA y problemas con los tiempos de respuesta en la conexión SMTP de Nodemailer. Estos obstáculos se superaron mediante pruebas iterativas con Jest [59], Cypress [61] y ajustes en los algoritmos de procesamiento, logrando una integración exitosa.

La estructura ROSES (Role, Objective, Scenario, Examples, Steps) ayudó a elaborar el prompt para la comunicación con la IA, esta permitió especificar los requerimientos necesarios para obtener respuestas esquematizadas en formato JSON, incluyendo la validación de las entradas para evitar retrasos en las respuestas y asegurar la precisión del contenido generado.

El uso de contenedores a través de Docker para el despliegue de la aplicación facilitó la portabilidad y la replicación del entorno de desarrollo en servidores de producción, no obstante, aunque se enfrentaron problemas iniciales relacionados con configuraciones de red y versiones de contenedores, estos fueron superados mediante la consulta de documentación oficial [67] y soporte técnico.

Este proyecto, al igual que estudios previos en el campo, subraya la importancia de las metodologías ágiles y las tecnologías modernas en el desarrollo de herramientas de software innovadoras. Sin embargo, a diferencia de investigaciones anteriores que se centraron en la optimización de procesos tradicionales, este trabajo se distingue por su enfoque en la integración de inteligencia artificial para la automatización y mejora de la calidad en el desarrollo de software. La combinación de IA con RAD no solo acelera el proceso de creación de casos de prueba, sino que también mejora su calidad y relevancia, marcando un avance significativo en las prácticas de aseguramiento de calidad en el desarrollo de software.

7.2 Segundo objetivo: Evaluar la percepción de utilidad de la herramienta desarrollada entre los estudiantes, utilizando como base ciertos lineamientos del modelo TAM (Modelo de Aceptación Tecnológica).

Se corrobora que la elección del Modelo de Aceptación Tecnológica (TAM) para evaluar la percepción de utilidad de la aplicación web de diseño de casos de prueba funcionales a partir de casos de uso fue acertada y proporcionó un marco sólido para este propósito, en concordancia con lo planteado por Cataido [11], TAM proporcionó una guía estructurada y bien definida para comprender cómo los usuarios perciben y adoptan la herramienta de diseño de casos de prueba funcionales, dado que el enfoque en la percepción de utilidad minimizó el área de cuestionamiento y, apoyó a mejorar y comprender áreas que son útiles para los usuarios.

A pesar del tamaño limitado de la muestra, que proviene de un muestreo por conveniencia focalizado en personas de 8vo y 9no ciclo con conocimiento en pruebas de la materia de calidad de software impartida en la UNL, lo que concuerda con la información de Cataldo [11] que menciona que la muestra se puede ver afectada por limitaciones o requerimientos, los datos obtenidos son de gran valor al aplicarse a una población específica y relevante para esta investigación, por tanto, en lo que respecta a la variable de percepción de utilidad del modelo TAM, más del 90% de los usuarios indicaron estar de acuerdo con las afirmaciones realizadas en cada una de las preguntas, mientras que un 4.8% manifestó tener una opinión neutra. Esto permitió conocer que los usuarios de la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso perciben utilidad de la herramienta en el proceso de generación de casos de prueba funcionales.

La percepción de utilidad del Modelo de Aceptación de Tecnología (TAM) reflejada en la evaluación a los usuarios de la aplicación sobrepasa el 90% de utilidad percibida, según la evaluación realizada en la **sección 6.2.2**, lo que sugiere que los usuarios divisan la herramienta como útil. En respuesta a la pregunta de investigación planteada: **¿Cuál es el nivel de percepción de utilidad de una herramienta basada en inteligencia artificial para el diseño de casos de prueba funcionales a partir de casos de uso, entre los estudiantes de la carrera de Computación de la Universidad Nacional de Loja?**, y basado en los datos recopilados, se concluye que la percepción de utilidad de la aplicación para el diseño de casos de prueba funcionales a partir de casos de uso, que implementa el consumo de IA instructiva, está en un nivel **alto**, según lo definido en la escala de Likert, lo que sugiere que es una solución viable y bien recibida para las necesidades de generación de casos de prueba funcionales a partir de casos de uso.

8. Conclusiones

Tras finalizar el presente TIC, se ha llegado a las siguientes conclusiones:

- La implementación de una herramienta para la generación de casos de prueba funcionales a partir de casos de uso ha demostrado ser una solución eficiente y adecuada para los proyectos académicos; este enfoque simplifica la creación de pruebas al automatizar procesos que usualmente requieren un alto grado de experiencia en análisis y diseño de pruebas, reduciendo así el tiempo y esfuerzo necesarios, así mismo, la segmentación del proceso en fases ha permitido un desarrollo más ágil y organizado.
- La metodología RAD ha resultado ser idónea para este proyecto de integración curricular, gracias a su enfoque en la rápida iteración y la obtención de retroalimentación constante, este método permitió la creación de prototipos en lapsos cortos, lo que facilitó la identificación temprana de ajustes necesarios y mejoró la alineación del producto con las expectativas de los usuarios. La flexibilidad de RAD, junto con sus ciclos de desarrollo rápidos, contribuyó a la entrega de un software funcional dentro del tiempo establecido.
- La integración de tecnologías como NestJS, Nodemailer, Next.js, TailwindCss y Cohere, así como la utilización de Prisma y PostgreSQL, demostró ser una elección acertada para la arquitectura del sistema, dado que, estas herramientas no solo facilitaron la implementación y el manejo de datos, sino que también garantizaron un rendimiento robusto y una fácil escalabilidad del sistema, aquí también, la integración de una API de IA fue clave para automatizar tareas complejas, ofreciendo un valor añadido significativo en la generación de casos de prueba.
- El despliegue mediante contenedores Docker en los servidores del laboratorio de software ha resultado en un proceso eficiente y efectivo, puesto que este enfoque no solo permitió un entorno controlado y replicable, sino que también garantizó la portabilidad del sistema, simplificando su mantenimiento y asegurando una configuración uniforme, esto es especialmente relevante en contextos académicos y colaborativos, donde la facilidad de despliegue y la estabilidad del entorno son importantes.
- El modelo de aceptación tecnológica TAM, ha permitido evaluar la percepción de utilidad de los usuarios al software de diseño de casos de prueba funcionales a partir de casos de uso desarrollado. La combinación de TAM, la escala de Likert y de los instrumentos de evaluación realizados, han permitido tener métricas y valores para conocer la utilidad percibida alcanzada en los usuarios de la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso.

- Por último, tras desarrollar los dos objetivos específicos de esta investigación, se concluye que la percepción de utilidad alcanzada de la aplicación para diseño de casos de prueba funcionales a partir de casos de uso, es positiva con más del 90% de aceptación. Los usuarios han mostrado una utilidad percibida que varía entre niveles medios y altos, reflejando una respuesta favorable en todas las variables evaluadas. No obstante, existen complementos que se podrían desarrollar en la herramienta que no se contemplan en el alcance, pero a nivel de percepción de utilidad de los usuarios resultaría útil o informativo, lo que sugiere que podría ser un foco de mejora en la aplicación web.

9. Recomendaciones

Al finalizar el presente TIC, se enlistan las siguientes recomendaciones:

- Si bien la herramienta desarrollada ha demostrado ser eficiente en la automatización de pruebas funcionales a partir de casos de uso, se recomienda explorar la incorporación de técnicas adicionales, como la generación de casos de prueba basados en historias de usuario o a partir de requisitos. Esto permitiría una cobertura y robustez diferente de las pruebas, ofreciendo un análisis distinto del comportamiento del software evaluado.
- Optimizar periódicamente el prompt utilizado en la integración de la IA, dentro de la aplicación web, con el objetivo de generar una mayor variedad de casos de prueba. Actualmente, la herramienta se enfoca en la generación de pruebas funcionales a partir de casos de uso, pero ajustar y enriquecer el prompt podría permitir la extracción de otros tipos de pruebas, como de integración, de regresión, de carga y estrés y de seguridad.
- Evaluar nuevos modelos de Ias generativas más especializados, con la finalidad de optimizar su desempeño o explorando alternativas que permitan mayor precisión en la generación de casos de prueba.
- Ampliar la muestra de evaluación utilizada en el modelo TAM, involucrando a usuarios con diferentes niveles de experiencia en pruebas de software. Esto permitirá obtener métricas más robustas y representativas, fortaleciendo el análisis de aceptación y facilitando futuras mejoras en la aplicación.
- La elaboración de una guía que documente las mejores prácticas en el desarrollo y despliegue de herramientas similares dentro del entorno académico. Esto proporcionará un marco de referencia para futuras investigaciones y proyectos que busquen mejorar la calidad y eficiencia en el diseño de casos de prueba funcionales.
- A los encargados de los servidores de aplicaciones de la Carrera Computación, documentar un procedimiento estandarizado para la carga de las aplicaciones en dichos servidores y así garantizar una replicabilidad efectiva en futuros proyectos, facilitando la adopción y el mantenimiento del sistema por otros equipos de desarrollo en el ámbito universitario.

Trabajos Futuros.

La integración de un sistema de autenticación centralizado Single Sign-On (SSO), alineado con las soluciones de autenticación ya utilizadas en la carrera de Computación para permitir un acceso más ágil y seguro a la herramienta.

10. Bibliografía

[1] F. Alvarez, Interviewee, *Entrevista sobre dificultades en el diseño de casos de prueba..* [Interview]. 07 2024.

[2] I. Sommerville, "Ingeniería de Software," Pearson Educación, 9, 2011.

[3] N. T. O. M. M. M. M. N. I. A. D. M. H. Saquib Ali Khan, "AI based Software Testing," 2024.

[4] R. Campaña, "El proceso de desarrollo rápido de aplicaciones de software," 2010.

[5] A. Cockburn, "Writing Effective Use Cases," Addison-Wesley, 2000.

[6] A. C. Mathieu Nancel, "CAUSALITY – A Conceptual Model of Interaction History," *HAL Open Science,* vol. 1, no. 10, 2017.

[7] M. Harman, "Software engineering meets evolutionary computation," *Computer,* vol. 44, no. 10, pp. 31-39, 2011.

[8] I. K. y. T. Pachidis, "Evolution towards Hybrid Software Development Methods and Information Systems Audit Challenges," *Software,* vol. 1, no. 3, pp. 316-363, 2022.

[9] N. A. y. B. A. J.K. Wiredu, "Design and Implementation of Online Crime Report System using Rapid Application Development (RAD) Methodology,," *Asian Journal of Science and Technology,* 2024.

[10] Z. H. y. A. P. M.L.H. Hamzah, "The Implementation of Rapid Application Development Method in Designing E-Learning based on Learning Management System Moodle at Universitas Islam," *ICoSEEH,* 2019.

[11] A. Cataldo, "Limitaciones y oportunidades del Modelo de Aceptación Tecnológica (TAM)," *Depto. de Ingeniería Informática y Cs. de la Computación,* vol. 1.

[12] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology.," *MIS Quarterly,* vol. 13, no. 3, pp. 319-340, 1989.

[13] V. &. D. F. D. Venkatesh, "A theoretical extension of the technology acceptance model: Four longitudinal field studies.," *Management Science,* vol. 46, no. 2, pp. 186-204., 2000.

[14] W. R. &. H. J. King, "A meta-analysis of the technology acceptance model.," *Information & Management,* vol. 43, no. 6, pp. 740-755., 2006.

[15] D. &. S. D. W. Gefen, "The relative importance of perceived ease of use in IS adoption: A study of e-commerce adoption.," *Journal of the Association for Information Systems,* vol. 1, no. 1, p. 8, 2000.

[16] P. I. J. &. C. P. Legris, "Why do people use information technology? A critical review of the technology acceptance model.," *Information & Management,* vol. 40, no. 3, pp. 191-204, 2003.

[17] V. C.-A. D. T.-J. S. F.-S. Norka Bedregal-Alpaca, "Evaluation of the student perception in relation to the use of the Moodle platform from the TAM perspective," *Ingeniare. Revista chilena de ingeniería,* vol. 27, no. 4, 2019.

[18] D. W. E. S. G. D. J. B. O. I. J. E. C. D. A. S. F. Lic. Ángela Guadalupe Canto de Gante, "Escala de Likert: Una alternativa para elaborar e interpretar un," *Academia Journals,* vol. 12, no. 1, pp. 1-8, 2020.

[19] I. M. S. A. &. A. R. S. Etikan, "Comparison of convenience sampling and purposive sampling.," *American journal of theoretical and applied statistics,* vol. 5, no. 1, pp. 1-4, 2016.

[20] H. Taherdoost, "Determining sample size; how to calculate survey sample size.," *International Journal of Economics and Management Systems,* vol. 2, 2017.

[21] G. A. M. &. T. M. Bierman, "Understanding TypeScript.," *European Conference on Object-Oriented Programming,* pp. 257-281, 2014.

[22] S. Fenton, "Pro TypeScript: Application-Scale JavaScript Development.," *Apress,* 2020.

[23] nestjs, "nestjs," 2023. [Online]. Available: https://docs.nestjs.com. [Accessed 14 10 2024].

[24] M. Kamil, "NestJS: A Progressive Node.js Framework.," in *In Enterprise Node.js Projects*, Berkeley, CA., Apress, 2021, p. 124.

[25] R. Perkins, "Hands-On RESTful Web Services with TypeScript 3: An end-to-end guide to building RESTful APIs using TypeScript and NestJS.," *Packt Publishing Ltd.,* 2021.

[26] A. Reinman, "Nodemailer - Module for Node.js to send emails," *GitHub Repository,* 2021.

[27] M. Williams, "Email Automation with Node.js and Nodemailer," *Journal of Web Development,* vol. 8, no. 3, pp. 45-52, 2023.

[28] S. Johnson, "Modern Email Integration Patterns with Nodemailer," *Node.js Best Practices Guide,* 2023.

[29] D. Martinez, "Implementing Secure Email Systems with Node.js," *Web Development Solutions,* vol. 4, pp. 78-85, 2024.

[30] Next.js, "Documentación oficial de Next.js.," Next.js, 2023. [Online]. Available: https://nextjs.org/docs.

[31] A. &. O. T. Wathan, " Modern Web Development with Next.js," *Tailwind Labs Inc.,* 2020.

[32] T. Takahashi, Hands-On Next.js: Build and deploy fullstack React applications with Next.js 12 and TypeScript, Packt Publishing., 2021.

[33] B. Orbán, "NextAuth.js Documentation and Implementation Guide," *NextAuth.js Official Documentation,* 2023.

[34] R. Anderson, "Modern Authentication Patterns with NextAuth.js," *Web Security Journal,* vol. 5, no. 2, pp. 123-135, 2023.

[35] K. Thompson, "Authentication Best Practices in Next.js Applications," *Next.js Development Handbook,* vol. 3, pp. 67-82, 2024.

[36] L. García, "Implementing Secure Authentication Flows with NextAuth.js," *Web Development Security Patterns,* vol. 6, 2024.

[37] P. G. D. Group., "PostgreSQL: The World's Most Advanced Open Source Relational Database," 2023. [Online]. Available: https://www.postgresql.org/.

[38] R. &. H. L. Obe, "PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database," *O'Reilly Media.,* 2020.

[39] Prisma, "Prisma - Next-generation Node.js and TypeScript ORM.," 2023. [Online]. Available: https://www.prisma.io.

[40] Npm, "NPM," 2024. [Online]. Available: https://www.npmjs.com/package/prisma.

[41] D. Saray, "Fullstack Development with Node.js, React, and Prisma," *Packt Publishing.,* 2022.

[42] Prisma, "Prisma Architecture," Prisma, 2023. [Online]. Available: https://prisma-client-py.readthedocs.io/en/v0.4.2/contributing/architecture/. [Accessed 15 10 2024].

[43] M. Fowler, "Patterns of Enterprise Application Architecture.," Addison-Wesley Professional., 2022.

[44] Cohere, "Cohere Docs," Cohere, 2024. [Online]. Available: https://docs.cohere.com/v2/docs/the-cohere-platform.

[45] S. D. A. H. Philippe Besnard, Computational Models of Argument: Proceedings of COMMA 2008, 2008.

[46] W. A. L. S. Karin Sim Smith, "Cohere: A Toolkit for Local Coherence," *European Language Resources Association (ELRA),* 2016.

[47] Microsoft, "Microsoft - Visual Studio Code," 2024. [Online]. Available: https://visualstudio.microsoft.com/es/#vscode-section. [Accessed 15 10 2024].

[48] G. C. Cuadrado, "OpenWebinars," 2022. [Online]. Available: https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/. [Accessed 15 10 2024].

[49] L. S. C. T. J. &. H. J. Dabbish, "Social coding in GitHub: transparency and collaboration in an open software repository," *In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work,* pp. 1277-1286, 2012.

[50] J. D. D. E. R. &. W. G. Blischak, " A quick introduction to version control with Git and GitHub.," *PLoS computational biology,* vol. 12, no. 1, p. e1004668, 2016.

[51] E. G. G. B. K. S. L. G. D. M. &. D. D. Kalliamvakou, " The promises and perils of mining GitHub.," *In Proceedings of the 11th working conference on mining software repositories,* pp. 92-101, 2014.

[52] Postman, "Postman," 2024. [Online]. Available: https://learning.postman.com/docs/introduction/overview/. [Accessed 14 12 2024].

[53] R. &. K. S. Patel, "API Development and Testing Methodologies: A Comprehensive Review," *International Journal of Software Engineering,* vol. 12, no. 3, 2021.

[54] X. &. R. M. Chen, "Continuous Integration and Deployment: Emerging Trends in DevOps Automation," *IEEE Software,* 2022.

[55] R. Kumar, "Comparative Analysis of CI/CD Platforms in Modern Software Development," *ACM Transactions on Software Engineering and Methodology,* vol. 32, no. 2, pp. 178-201, 2023.

[56] S. J. Patel, "Event-Driven Automation in Software Development Workflows," *International Journal of Software Engineering Innovation,* vol. 15, no. 3, 2022.

[57] X. &. R. M. Chen, "Modern JavaScript Testing Frameworks: A Comparative Analysis," *IEEE Software,* vol. 39, no. 5, 2022.

[58] R. Kumar, "Evolution of Testing Methodologies in Front-end Development," *ACM Transactions on Software Engineering and Methodology,* 2023.

[59] S. &. J. L. Patel, "Mock Functions and Simulation in JavaScript Testing Environments," *International Journal of Software Testing and Validation,* vol. 16, no. 2, 2022.

[60] Cypress, "Cypress, un framework de pruebas todo en uno," [Online]. Available: https://www.paradigmadigital.com/dev/cypress-un-framework-de-pruebas-todo-en-uno/.

[61] Cypress, "Cypress Docs," [Online]. Available: https://docs.cypress.io/app/get-started/why-cypress.

[62] O. Jordán Enriquez and O. Vázquez Ruiz, "GENERACIÓN DE CASOS DE PRUEBA A PARTIR DE CASOS DE USO EN LAS PRUEBAS DE," *Ingeniería Industrial,* vol. XXVII, no. 1, pp. 1-5, 2005.

[63] F. P. A. G. a. L. C. B. Chunhui Wang, "Automatic Generation of Acceptance Test Cases," *UMTG,* vol. 2, 2019.

[64] M. V. Navarro, *Desarrollo de un chatbot de apoyo a la terapia,* Madrid, 2024.

[65] I. G. T. A. ,. C. S. Michael Vierhauser, "Towards Integrating Emerging AI Applications in SE Education," *PREPRINT ,* no. 36, 2024.

[66] G. AI., "Gemini API Overview.," 2023. [Online]. Available: https://ai.google.dev/docs/gemini_api_overview. [Accessed 15 10 2024].

[67] Docker, "Docker Docs," 2024. [Online]. Available: https://docs.docker.com/get-started/get-docker/.

[68] R. e. a. Bommasani, "On the Opportunities and Risks of Foundation Models," in *On the Opportunities and Risks of Foundation Models*, arXiv preprint arXiv:2108.07258v2., 2023.

[69] Y. B. Y. &. H. G. LeCun, "Deep learning.," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[70] Gemini, "Codelabs," Codelabs, 2023. [Online]. Available: https://codelabs.developers.google.com/codelabs/gemini-java-developers?hl=zh-tw. [Accessed 15 10 2024].

11. Anexos

Anexo 1. Entrevista realizada al especialista de calidad de la carrera de Computación de la Universidad Nacional de Loja.

FACULTAD DE LA ENERGÍA, LAS INDUSTIRAS Y LOS RECURSOS NATURALES NO RENOVABLES

INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

Entrevistador: Juan Francisco Castillo Estrella

Entrevistado: Ing. Francisco Álvarez

Fecha: 8 de julio del 2024

El propósito de esta entrevista es explorar las experiencias y perspectivas del Ing. Francisco Álvarez, docente de la carrera de Computación en la UNL, en relación con el proceso de diseño de casos de prueba basados en casos de uso. Se busca identificar los desafíos actuales, la eficiencia del proceso y las posibles mejoras mediante la implementación de un sistema automatizado basado en inteligencia artificial (IA). Las respuestas obtenidas serán valiosas para comprender mejor la situación actual y evaluar la viabilidad de adoptar nuevas tecnologías para optimizar este proceso.

1. **¿Cuál es el proceso actual para el diseño de casos de prueba en los proyectos de software del 8vo ciclo de la UNL?**

El proceso actual lo estamos manejando en base al ISTQB, y básicamente es un proceso estándar que permite a los estudiantes conocer cómo se debe realizar el proceso de pruebas. Entonces, esa es nuestra base principal. Existen otras formas de trabajo, otros procesos, pero ya se sale un poquito del alcance. Con que conozcamos uno que se puede aplicar a todos los proyectos, pues ya nos ayuda muchísimo.

2. **Desde su experiencia en la docencia ¿Cuáles son los principales desafíos o problemas que enfrenta el proceso en la fase de diseño de casos de prueba?**

Bien, no solo en la docencia sino en la parte profesional, el diseño de casos de pruebas es crítico, ya que en base a eso se detectarán o no errores en evaluación de un producto. Temas que se tienen ahí, básicamente que los alumnos puedan diseñar casos de prueba eficientes, es decir, que detecten errores, por un lado y por el otro, que ayuden a validar los requisitos del sistema. Esos son los retos más fuertes hasta ahora.

3. **¿Con qué frecuencia se presentan retrasos o errores en el diseño de casos de prueba debido al proceso manual?**

Este proceso manual requiere de conocimientos del negocio, por ejemplo, por parte de las personas que van a diseñar casos de prueba este tema suele obviarse mucho cuando tú diseñas casos de prueba por primera vez, digamos, que es el acercamiento que tenemos en la asignatura. Entonces, esa partecita es la que cuesta implicar en los alumnos que no solo tienen que centrarse en la funcionalidad que están evaluando los requerimientos que están

evaluando, sino que tiene que conocer todo el negocio para diseñar casos de prueba eficientes. Ese es uno de los temas fuertes que tenemos acá.

4. **¿Cuál es su percepción sobre la eficiencia del proceso de diseño de casos de prueba actual en ISTQB en términos de tiempo y recursos empleados?**

De lo que he visto, por ejemplo, en algunos trabajos no generan mucha eficiencia en cuanto a ahorro de tiempo. Puede haber varios factores ahí. Pueden ser, si es con los alumnos, por lo que están empezando en la parte de pruebas, si es con los tesistas, quizás por el desconocimiento o falta de refrescarse un poquito al momento de ejecutar sus pruebas en las tesis y si es en los profesionales, básicamente es una mala aplicación de las técnicas de prueba. No diseñan en base a las técnicas, sino lo diseñan en base a su experiencia y se olvidan de temas básicos que debemos considerar en el diseño de las pruebas. Entonces, claro, hay retrasos en tiempo, en ejecución y en la efectividad de estos casos

5. **La mayoría de procesos de pruebas incluyen una fase de diseño de casos de prueba. Sus estudiantes ¿Le han compartido preocupaciones sobre el diseño de casos de prueba? ¿Cuáles?**

Más que compartirme directamente se ha notado una falencia en ese sentido, porque es complejo diseñar casos de prueba. Hemos diseñado laboratorios, por ejemplo, para que apliquen un poco el diseño y se complica un poco. Se complica el diseñar, el entender, el armar precondiciones, postcondiciones, los datos de entrada, datos de salida, entonces se ha visto esa necesidad. Con algunos, más que comentarme los alumnos personalmente, lo hemos analizado en clases, de tal manera que todos aprendamos sobre las dudas que tiene cada uno. Y sí, genera algunas dudas, ¿no? Desde cómo suscribe el caso de prueba hasta pensar en el diseño, en cómo se va a ejecutar el caso del problema. Entonces, sí, sí hay algunos temas que se podrían mejorar por esa parte.

6. **Desde su experiencia de docencia ¿Cómo ha evaluado la efectividad de los casos de prueba diseñados por sus estudiantes?**

Más que en los estudiantes, podría hablar de las tesis. Entonces, por ejemplo, he visto que en las tesis se diseñan casos de prueba sólo por cumplir con los requerimientos que tiene una metodología, mas no están orientados a una validación real del producto que se quiere hacer. Entonces, más bien están por cumplir con algo que se solicita en el documento y se diseñan unos casos de pruebas esporádicas. Es más, he visto que los diseñan ya al final de la tesis, ¿no? Cuando se ejecutaron las pruebas ni antes de valorar el producto, sino al final. Entonces, claro que algo está mal. En cuanto a los alumnos, básicamente los alumnos de octavo, como están iniciando en este proceso, no se podría evaluar la eficacia real en un proyecto, pero sí en la construcción, en el diseño, que sí les cuesta un poquito definir los casos de prueba. Entonces, si es una limitante porque es una parte clave para la ejecución de las pruebas.

7. **Desde su experiencia de docencia ¿Ha habido casos en los que los proyectos de los estudiantes se han visto afectados negativamente debido a su diseño manual?**

Sí, la nota directamente. Que tienen el abono actual. Hay algunos temas, por ejemplo, una incorrecta utilización del diseño de los casos de correo. Entonces, claro, como es un proceso manual, se obvia en algunas partes, se dejan de lado requisitos, no se amplía la visión general de todo lo que se debe probar. Entonces, sí, hay algunas falencias que se podrían corregir quizás con el acompañamiento tecnológico

8. **En su opinión, ¿cuáles serían los beneficios de implementar un sistema automatizado basado en IA para la generación de casos de prueba en base a casos de uso?**

Bueno, en el ámbito educativo, pues facilitar la comprensión del diseño en los casos de prueba de los alumnos. Ese es el principal factor porque podríamos comparar los resultados manuales con los resultados que nos muestra la herramienta. Y eso ayuda mucho a corregir por parte de los alumnos cualquier falencia que tengan. La independencia del profesor también, porque podrían ejecutar la herramienta en cualquier momento sin necesidad de que esté el docente y podrían determinar qué salió bien y qué salió mal. En el tema profesional, claro, ayudaría muchísimo en la reducción de tiempos, en la reducción de tiempos, en aprovechar más los recursos. Y, claro, nos daría mucho más tiempo para hacer otro tipo de pruebas.

9. **¿Considera que el uso de técnicas de NLP (Procesamiento del Lenguaje Natural) para la generación de casos de prueba podría mejorar la calidad y eficiencia del proceso?**

Sí, yo pienso que podría iniciarse por esta parte, pero también habría que complementar el estudio para ver si hay algunas otras técnicas. Pero esta podría utilizarse como punto de partida para diseñar algún tipo de acercamiento con este diseño de casos de prueba, que es lo que se pretende. Sin embargo, sí recomendaría yo realizar una búsqueda un poquito formal, aunque no sea una revisión sistemática, pero sí una búsqueda bibliográfica para ver si es que hay algunas otras técnicas, pero esta, inicialmente, estaría bastante bien para el diseño de casos de prueba

10. **¿Qué mejoras considera pertinentes que se pueden implementar al proceso actual de la fase de diseño de casos de prueba? ¿Cuáles?**

Básicamente la cobertura de los casos de prueba a los requisitos, que realmente los casos de prueba cobran todo lo necesario y generar los casos de prueba negativos. Eso es algo que se olvidan mucho las pruebas. Por lo general, se concentran en evaluar los casos de prueba válidos y muchas veces los no controlados, los que no están especificados, los complementarios, los negativos que les comento son los que suelen generar errores. Entonces, eso ayudaría mucho.

11. **En la elaboración de casos de prueba, existe la fase del diseño ¿Considera que, si se llegara a implementar un sistema automatizado basado en IA, sería necesario alterar el flujo actual de su diseño o reemplazarlo por completo?**

Eso tendríamos que analizarlo en el camino. Conforme se desarrolle tu tesis y con los requisitos que arme el proyecto, tendríamos que ver si nos toca modificar algún procedimiento del análisis y diseño, o si podemos mantener el mismo, como ya se involucra una herramienta automática, es posible que algunas fases manuales se dejen de lado y es posible que la herramienta automática también requiera de algún otro tipo de información para poder trabajar. Entonces, más bien esta parte, yo diría que la iremos descubriendo en el camino ya como resultado de tu proyecto.

12. **¿Cuál es su percepción sobre la aceptación de un sistema automatizado por parte de los estudiantes y personal docente?**

Como comentaba en una pregunta anterior, a nivel educativo nos ayudaría mucho al docente para explicar mediante la tecnología cómo se diseñan los casos de prueba y al estudiante para aprender mediante esa tecnología cómo mejorar su diseño. Entonces nos ayudaría mucho en esta parte. En cuanto al ámbito profesional, como comentaba realmente el tema de ahorro de tiempo, de reutilización, de tener un padre revisor que, si bien no necesariamente debo hacer pruebas automáticas que me ayude a contrastar las pruebas manuales y sería el diseño, pero en el caso de las pruebas manuales sería bastante bueno. Así, si algo se me escapó, pues lo incorporo, y si todo está correcto, pues ya puedo ir confiando en esa herramienta automática para dedicarme a ver temas que tienen más riesgo en el proyecto.

13. **¿Estaría dispuesto a colaborar en la fase de implementación y prueba de un sistema automatizado basado en IA para elaboración de casos de prueba en base a casos de uso?**

Sí, desde lo que conocemos, podemos aportar alguna parte metodológica o lo que se requiera para desarrollar el proyecto. Aunque se tiene poco conocimiento en la parte de calidad, pero creo que se podría aportar mutuamente en conjunto contigo con el taxista para ir desarrollando algo.

- **Link del drive con el audio de la entrevista:** https://drive.google.com/drive/folders/1ME5grtQJzCc0omDfmGw8Y2fU4bEvzur3?usp=sharing

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Francisco Álvarez	Firmado electrónicamente por: FRANCISCO JAVIER ALVAREZ PINEDA

Anexo 2. Aplicación de Técnica de Brainstorm para la obtención de requisitos.

[Técnica Brainstorm para la obtención de requisitos]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Documento de Evidencia: Aplicación de Técnica de Brainstorming para Obtención de Requisitos

Información General			
Proyecto:	Software para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la Carrera de Computación de la Universidad Nacional de Loja.		
Duración:	30 minutos	**Fecha de la sesión:**	22-10-2024
Anfitrión:	Juan Francisco Castillo Estrella		
Participantes:	• Jean Carlo Martínez Herrera • Josué Alejandro Sauca Pucha • William Enrique Cueva Rivas • Danny Augusto Martínez Granda • Bryan Jeremy Encalada Mejía • José Andrés Macas Vélez • Cristian Ramiro Capa Rodríguez • Jimmy Alexander Cajamarca Escaleras • Jennifer Elizabeth Quizhpe Hurtado		

1. Objetivo de la Sesión

- Identificar y aclarar los requisitos para el software de diseño de casos de prueba funcionales basado en casos de uso, utilizando IA de Cohere, específicamente para la Carrera de Computación de la Universidad Nacional de Loja.

2. Metodología

2.1 Técnica utilizada

Brainstorming clásico con enfoque en la obtención de requisitos de software

2.2 Reglas establecidas

- Todas las ideas son bienvenidas, sin juicios inmediatos
- Construir sobre las ideas de los demás
- Buscar cantidad de ideas inicialmente, la calidad se refinará después
- Respetar el tiempo de habla o espacio de trabajo de cada participante.
- Mantener el enfoque en el objetivo de la sesión

2.3 Materiales utilizados

- Pizarra virtual para anotar ideas (Miro).
- Notas adhesivas (digitales en la herramienta de Miro).
- Documento compartido para registrar ideas en tiempo real (Pizarra en Miro).
- Timer para controlar los tiempos de las actividades.

3. Desarrollo de la Sesión

3.1 Preguntas de enfoque

- Si tuviera una herramienta para diseñar casos de prueba funcionales a partir de casos de uso, **¿qué características o funciones no podrían faltar?**
- Imagine que tiene una aplicación que asiste en la creación de casos de prueba, **¿qué tipo de ayuda te gustaría que te ofreciera en el proceso?**
- Imagine que esta herramienta es utilizada para enseñar a otros sobre pruebas funcionales. **¿Qué funcionalidades cree que podría hacer el aprendizaje más efectivo?**
- Si pudiera visualizar o comunicar mejor su trabajo con profesores o colegas, **¿cómo le gustaría hacerlo?**

3.2 Resumen de ideas propuestas

En la Figura 1 se muestran las ideas propuestas por los participantes de la sesión de brainstorm, las cuales se dieron a partir de la lectura de las preguntas de enfoque; para que aporten con las ideas, se habilitó un enlace hacia la herramienta MIRO, donde quedó registro de las mismas.

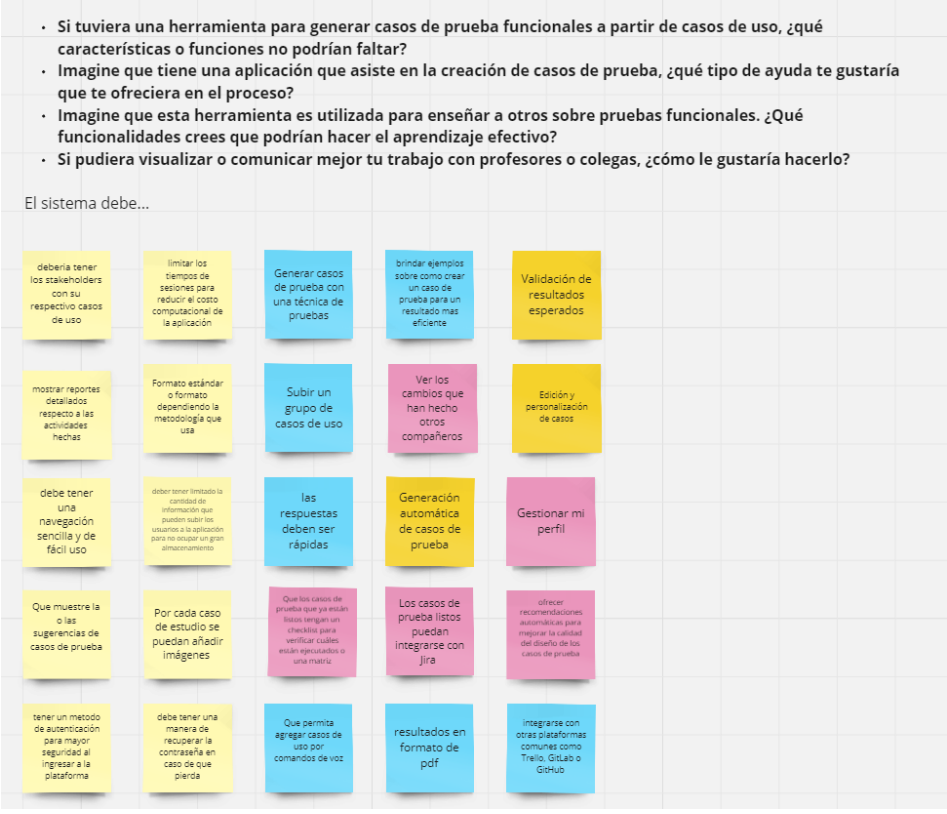


Figura 1: Pizarra de MIRO de ideas propuestas.

Estas ideas se las trasladó a la Tabla 1, la cual, muestra de forma organizada y con mejor redacción los aportes de los participantes y para un mejor control se los etiquetó con un código.

Tabla 1: Ideas Propuestas.

Código	Ideas Propuestas
IP001	Generación de casos de prueba a partir de una técnica de pruebas.
IP002	Subir un grupo de casos de prueba para evaluarlos en conjunto.
IP003	Validación de resultados esperados automáticamente.
IP004	Edición y personalización de los casos generados.
IP005	Gestión del perfil de usuario para personalización y control.
IP006	Generación automática de casos de prueba, facilitando el proceso.
IP007	Integración con Jira para sincronizar los casos de prueba con herramientas de gestión.
IP008	Exportación de resultados en formato PDF para compartir o almacenar.
IP009	Lista de stakeholders con su respectivo caso de uso.
IP010	Limitación de tiempos de sesión para optimizar recursos computacionales.
IP011	Ejemplos de cómo crear casos de prueba eficaces.
IP012	Visualización de cambios realizados por otros compañeros en los casos.
IP013	Respuestas rápidas del sistema al crear o procesar casos.
IP014	Reportes detallados sobre actividades y resultados de las pruebas.
IP015	Formato estándar o flexible según la metodología empleada (Agile, Waterfall, etc.).
IP016	Navegación sencilla y amigable para mejorar la experiencia del usuario.
IP017	Sugerencias automáticas de casos de prueba para optimizar el diseño.
IP018	Posibilidad de añadir imágenes a los casos de prueba.
IP019	Limitación del almacenamiento si los recursos son limitados.
IP020	Checklist para verificar la calidad y consistencia de los casos de prueba.
IP021	Autenticación segura para acceder a la plataforma.
IP022	Recuperación de contraseñas en caso de olvido.
IP023	Ingreso de casos de uso por comandos de voz para una experiencia más dinámica.

3.3 Categorías identificadas

Posteriormente, en la Tabla 2 se procedió a una categorización de las ideas para sectorizar enfoques y obtener una gestión controlada de las mismas, en dicha tabla se presentan las categorías de las ideas encontradas y se marcó su trazabilidad mediante el código antes impuesto.

Tabla 2: Categorización de Ideas Propuestas

Categoría	Idea Generada
Generación y gestión de casos de prueba	IP001, IP002, IP004, IP020, IP017, IP011, IP006
Interacción y colaboración	IP012, IP005, IP007
Validación y reportes	IP003, IP013, IP014, IP015, IP008
Seguridad y acceso	IP021, IP022, IP010
Usabilidad y experiencia de usuario	IP016, IP023, IP009
Almacenamiento y rendimiento	IP019
Visualización y soporte multimedia	IP018

4. Resultados

4.1 Lista de requisitos potenciales

Al ya tener el aporte de las ideas propuestas en la sesión y con su debida categorización, de forma individual, como analista de esta técnica, se dio paso a definir los posibles requisitos que serán aplicados en la aplicación, por tanto, se inició ajustando las ideas para convertirlos en requisitos, con una lista de requisitos potenciales que se evidencian en la Tabla 3:

Tabla 3: Lista Requisitos Potenciales.

Código	Requisitos Potenciales
RP001	El sistema debe diseñar casos de prueba a partir de los casos de uso ingresados.
RP002	El sistema debe permitir subir y gestionar un grupo de casos de prueba para su evaluación conjunta.
RP003	El sistema debe permitir la edición y personalización de los casos de prueba generados.
RP004	El sistema debe proporcionar ejemplos sobre cómo crear casos de prueba eficaces.
RP005	El sistema debe sugerir casos de prueba basados en los casos de uso.
RP006	El sistema debe incluir un checklist para verificar la calidad y consistencia de los casos de prueba.
RP007	El sistema debe permitir visualizar los cambios realizados por otros compañeros en los casos de prueba.
RP008	El sistema debe permitir la gestión de perfiles de usuario para personalización y control.

RP009	El sistema debe integrarse con herramientas como Jira para sincronizar los casos de prueba con la gestión de proyectos.
RP010	El sistema debe permitir la colaboración entre múltiples usuarios en proyectos compartidos.
RP011	El sistema debe validar automáticamente los resultados esperados de los casos de prueba.
RP012	El sistema debe proporcionar respuestas rápidas en la generación y procesamiento de casos de prueba.
RP013	El sistema debe generar reportes detallados sobre los casos de uso y resultados de los casos de pruebas realizadas.
RP014	El sistema debe ofrecer opciones de formato estándar o flexible para adaptar los casos de prueba a diferentes metodologías.
RP015	El sistema debe exportar los resultados en formatos como PDF, Excel, y Word para facilitar su distribución.
RP016	El sistema debe ofrecer autenticación segura para el acceso a la plataforma.
RP017	El sistema debe proporcionar un método de recuperación de contraseñas en caso de olvido.
RP018	El sistema debe limitar los tiempos de sesión para optimizar los recursos computacionales y evitar inactividad prolongada.
RP019	El sistema debe ofrecer una navegación sencilla e intuitiva para mejorar la experiencia del usuario.
RP020	El sistema debe permitir el ingreso de casos de uso mediante comandos de voz para facilitar el proceso.
RP021	El sistema debe mostrar claramente los stakeholders asociados con sus respectivos casos de uso.
RP022	El sistema debe limitar la cantidad de información almacenada por cada caso de uso para optimizar el uso de los recursos.
RP023	El sistema debe optimizar el costo computacional mediante la limitación de sesiones o recursos innecesarios.
RP024	El sistema debe permitir a los usuarios añadir imágenes a los casos de prueba para complementar la información.
RP025	El sistema debe mostrar ejemplos visuales o plantillas predefinidas para facilitar la creación de casos de prueba.

5.2. Priorización inicial

Como siguiente paso, para la priorización se aplica el método MoSCoW con el cual se establece la prioridad de necesidad de los requisitos, y así separar cuales son los más relevantes para ejecutarlos y cuales quedan para una siguiente versión o trabajos futuros.

En la Tabla 4 se muestra una priorización mediante el método MoSCoW, que determina cuales son los requisitos potenciales que se consideran indispensables y los que directamente no son necesarios como primera versión de la aplicación.

Tabla 4: Método MoSCoW.

Método MoSCoW	
MUST HAVE (Críticos, indispensables para el funcionamiento básico del sistema)	RP001, RP003, RP004, RP011, RP018, RP016, RP019, RP015, RP012, RP008, RP010, RP017, RP025
SHOULD HAVE (Importantes, pero no esenciales para la primera versión)	RP013, RP021, RP024
COULD HAVE (Deseables, pero menos urgentes)	RP002, RP022, RP020, RP023, RP014, RP007, RP006
WON'T HAVE THIS TIME (No se priorizan para la versión inicial, se pueden dejar para futuras versiones)	RP005, RP009, RP022

Existen áreas de los requisitos potenciales que no quedaron claras, por tanto, se requiere una explicación más a detalle de cómo se llevarán a cabo.

4.3 Áreas que requieren clarificación

- **Formato de los casos de prueba:** Se adoptará el formato estándar ISTQB, usando JSON para enviar la estructura a la IA.
- **Validación de resultados:** La IA validará automáticamente comparando tres resultados para mejorar la precisión.
- **Integración con Jira:** La integración se centrará en casos de uso y resultados, pero no es una prioridad inmediata.
- **Reportes y análisis:** Se espera generar reportes siguiendo el formato IEEE 829.
- **Limitación de sesiones:** Se gestionará mediante JWT y Next Auth, controlando la duración de la sesión para reducir el costo computacional.

- **Generación de ejemplos:** El sistema proporcionará guías y sugerencias, con posibles tips al iniciar sesión, pero no es prioridad inmediata.
- **Gestión de perfiles y permisos:** Los usuarios tendrán roles como administrador, usuario, etc., con permisos de lectura y escritura asignados por el creador del proyecto.
- **Colaboración entre usuarios:** Los cambios se visualizarán con la opción de añadir un historial en versiones futuras, pues no es una prioridad inmediata.
- **Recuperación de contraseñas:** Será mediante correo electrónico, enviando una contraseña temporal para cambiarla luego.
- **Comandos de voz:** La entrada por voz no es una prioridad y podría implementarse en el futuro.

Para un complementar el análisis, se contrasta los insights clave, desafíos y lecciones, que resultan de generar estos requisitos potenciales, y que, aportan en el refinamiento de estos.

4.4. Insights clave

- **Automatización como clave:** El sistema automatiza la creación y gestión de casos de prueba funcionales a partir de los casos de uso, validación de resultados y exportación en un formato, reduciendo el trabajo manual y mejorando la eficiencia.
- **Personalización y flexibilidad:** Aunque automatizado, los usuarios pueden editar y personalizar los casos de prueba funcionales.
- **Enfoque en la experiencia de usuario:** Se prioriza la facilidad de uso con una navegación simple y un ejemplo claro, para que sea accesible a usuarios con diferentes niveles técnicos.
- **Seguridad y optimización:** La autenticación segura y la gestión eficiente de sesiones protegen la información y optimizan el rendimiento del sistema.

4.5 Desafíos encontrados

- **Automatización vs. personalización:** Aunque la automatización es clave, es importante permitir que los usuarios personalicen los casos de prueba sin sentirse limitados.
- **Validación automática:** Validar automáticamente los resultados de los casos de prueba funcionales es un reto técnico, especialmente en entornos diversos y dinámicos.
- **Rendimiento:** Se busca generar y procesar casos de prueba rápidamente, pero mantener la velocidad en proyectos grandes puede ser un desafío, afectando recursos y experiencia.
- **Seguridad:** Garantizar autenticación segura sin complicar la usabilidad es clave para proteger los datos sin afectar la experiencia del usuario.

- **Optimización de recursos:** Limitar tiempos de sesión para optimizar recursos debe hacerse sin interrumpir el flujo de trabajo, lo que requiere una gestión cuidadosa.

4.6 Lecciones aprendidas

- **Automatización flexible:** La automatización por sí sola no es suficiente; los usuarios necesitan personalizar los casos de prueba funcionales. Las herramientas deben ser flexibles para ajustarse a sus necesidades.
- **Simplicidad en la experiencia de usuario:** Una interfaz intuitiva es clave. Aunque el sistema sea avanzado, si no es fácil de usar, los usuarios se frustrarán.
- **Seguridad y usabilidad:** La seguridad es esencial, pero los procesos de autenticación no deben ser tan complejos que afecten la eficiencia del usuario.

A partir de los requisitos priorizados, así como de los insights, lecciones y desafíos identificados, se ha decidido agruparlos en las categorías de **Must have** y **Should have**. Destacando los puntos clave de cada sección, se pueden refinar los requisitos para mejorar la gestión de aquellos con mayor potencial, pues con este enfoque se define con mayor precisión los requisitos que se implementarán en el desarrollo de la aplicación web.
Por ello, los requisitos de forma general quedan como se muestra en la Tabla 5, la cual muestra los requerimientos con ajustes y una redacción más concisa.

Tabla 5: Requisitos Potenciales.

Código	Requisitos
R001	El sistema debe implementar autenticación mediante correo y contraseña.
R002	El sistema debe cerrar sesiones después de 15 minutos de inactividad.
R003	El sistema debe permitir crear usuarios.
R004	El sistema debe permitir actualizar usuarios.
R005	El sistema debe permitir buscar usuarios por nombre.
R006	El sistema debe permitir dar de baja usuarios.
R007	El sistema debe permitir la asignación de roles (Administrador, Usuario).
R008	El sistema debe mostrar una guía de cómo utilizar la herramienta.
R009	El sistema debe permitir compartir proyectos con otros usuarios.
R010	El sistema debe implementar un diseño responsive para diferentes dispositivos.
R011	El sistema debe implementar un modo oscuro/claro.
R012	El sistema debe exportar reportes en formato PDF IEEE 829.
R013	El sistema debe mantener tiempos de respuesta menores a 5 segundos para operaciones comunes.

R014	El sistema debe permitir diseñar casos de prueba funcionales a partir de casos de uso.
R015	El sistema debe permitir la edición y personalización de los casos de prueba funcionales diseñados.
R016	El sistema debe proporcionar ejemplos sobre cómo crear casos de prueba funcionales.
R017	El sistema debe permitir crear proyectos.
R018	El sistema debe permitir actualizar proyectos
R019	El sistema debe permitir buscar proyectos por nombre
R020	El sistema debe permitir dar de baja proyectos
R021	El sistema debe permitir crear casos de uso.
R022	El sistema debe permitir actualizar casos de uso
R023	El sistema debe permitir buscar casos de uso por nombre
R024	El sistema debe permitir dar de baja casos de uso
R025	El sistema debe permitir visualizar una explicación paso a paso de cómo se diseñaron los casos de prueba funcionales.
R026	El sistema debe permitir cerrar sesión
R027	El sistema debe permitir recuperar contraseña.
R028	El sistema debe permitir cifrar las claves de cuentas de usuario.
R029	El sistema deberá consumir a la API de Cohere.
R030	El sistema debe permitir personalizar y actualizar el perfil del usuario.
R031	El sistema debe limitar a 10 el número máximo de proyectos a crear por usuario

5. Aprobaciones

Actor	Nombre:	Firma
Facilitador:	Ing. María del Cisne Ruilova	Firmado electrónicamente por: MARIA DEL CISNE RUILOVA SANCHEZ
Líder del Proyecto:	Juan Francisco Castillo Estrella.	Firmado electrónicamente por: JUAN FRANCISCO CASTILLO ESTRELLA
Fecha de aprobación: 22-10-2024		

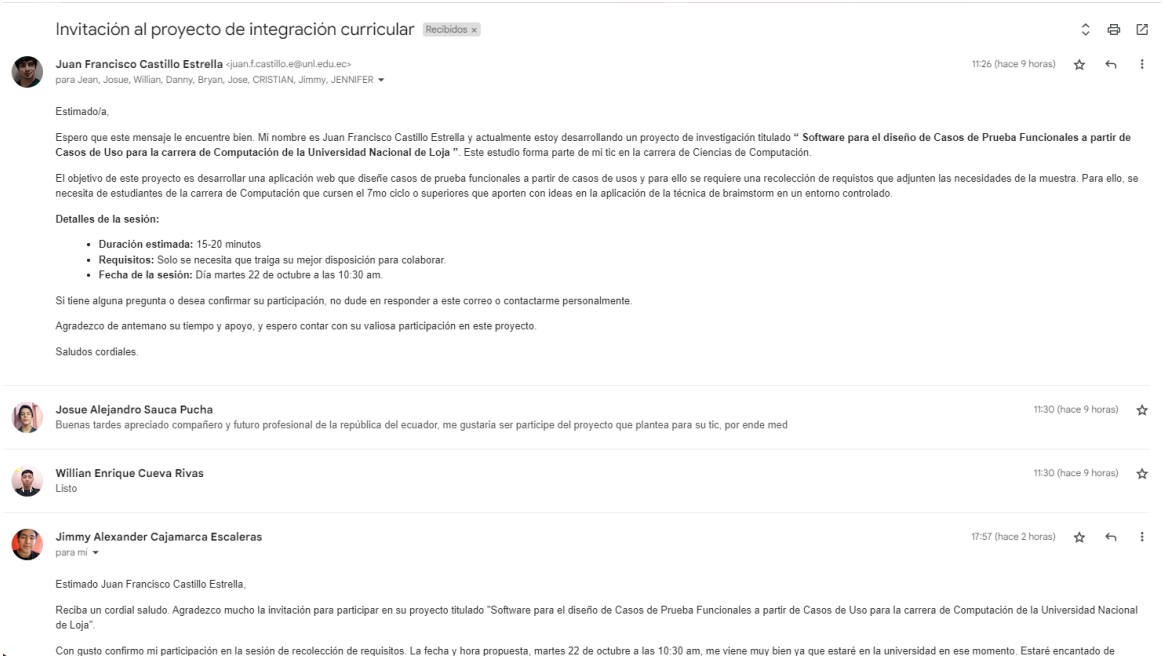
6. Anexos

Anexo 1.

Constancia de participación de sesión de Brainstorm.

ASISTENCIA Y CONSTANCIA DE SESIÓN DE BRAINSTORM	
Lugar de sesión:	Universidad Nacional de Loja.
Fecha de sesión:	22 de octubre del 2024
Nombres y apellidos del anfitrión:	Juan Francisco Castillo Estrella
Tiempo de sesión:	30 minutos

Se envió un correo a todos los participantes para solicitar su asistencia en la sesión de la técnica Brainstorm.



Invitación al proyecto de integración curricular Recibidos ×

Juan Francisco Castillo Estrella <juan.f.castillo.e@unl.edu.ec> 11:26 (hace 9 horas)
para Jean, Josue, Willian, Danny, Bryan, Jose, CRISTIAN, Jimmy, JENNIFER

Estimado/a,

Espero que este mensaje le encuentre bien. Mi nombre es Juan Francisco Castillo Estrella y actualmente estoy desarrollando un proyecto de investigación titulado " **Software para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera de Computación de la Universidad Nacional de Loja** ". Este estudio forma parte de mi tic en la carrera de Ciencias de Computación.

El objetivo de este proyecto es desarrollar una aplicación web que diseñe casos de prueba funcionales a partir de casos de usos y para ello se requiere una recolección de requistos que adjunten las necesidades de la muestra. Para ello, se necesita de estudiantes de la carrera de Computación que cursen el 7mo ciclo o superiores que aporten con ideas en la aplicación de la técnica de braimstorm en un entorno controlado.

Detalles de la sesión:

- **Duración estimada:** 15-20 minutos
- **Requisitos:** Solo se necesita que traiga su mejor disposición para colaborar.
- **Fecha de la sesión:** Día martes 22 de octubre a las 10:30 am.

Si tiene alguna pregunta o desea confirmar su participación, no dude en responder a este correo o contactarme personalmente.

Agradezco de antemano su tiempo y apoyo, y espero contar con su valiosa participación en este proyecto.

Saludos cordiales.

Josue Alejandro Sauca Pucha 11:30 (hace 9 horas)
Buenas tardes apreciado compañero y futuro profesional de la república del ecuador, me gustaria ser participe del proyecto que plantea para su tic, por ende med

Willian Enrique Cueva Rivas 11:30 (hace 9 horas)
Listo

Jimmy Alexander Cajamarca Escaleras 17:57 (hace 2 horas)
para mí

Estimado Juan Francisco Castillo Estrella,

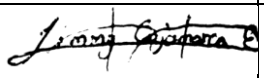
Reciba un cordial saludo. Agradezco mucho la invitación para participar en su proyecto titulado "Software para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera de Computación de la Universidad Nacional de Loja".

Con gusto confirmo mi participación en la sesión de recolección de requisitos. La fecha y hora propuesta, martes 22 de octubre a las 10:30 am, me viene muy bien ya que estaré en la universidad en ese momento. Estaré encantado de

Figura 1: Evidencia del correo enviado a los participantes.

Al finalizar la reunión se llevó constancia de la participación, mediante una recolección de firmas.

Tabla 1: Firma de Constancia de asistencia de los participantes.

FECHA	HORA DE INGRESO	HORA DE SALIDA	ESTUDIANTE	FIRMA (ESTUDIANTE)	DEPARTAMENTO
22-10-24	10:30	11:00	Jean Carlo Martinez		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	Josue Alejandro Sauca Pucha		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	William Cueva		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	Danny Martinez		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	Bryan Jeremy Encalada		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	José Andrés Macas		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	Cristian Ramiro Capa		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	Jimmy Cajamarca		Aula 323 – Carrera de Computación UNL
22-10-24	10:30	11:00	Jennifer Quizhpe		Aula 323 – Carrera de Computación UNL

Anexo 2.

Fotos de la sesión de Brainstorm realizada en las aulas de la Carrera de Computación de la Universidad Nacional de Loja.

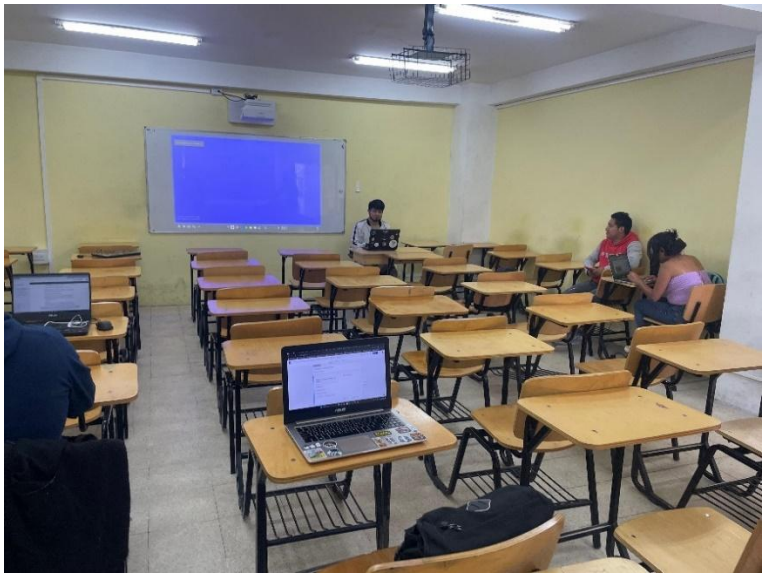


Figura 2: Evidencia de la sesión de Brainstorm.

Figura 3: Evidencia de la sesión de Brainstorm.

Figura 4: Evidencia de la sesión de Brainstorm.

Link de Audio de grabación de la sesión:
https://drive.google.com/file/d/1UCHG0ruHEPykXMMHgywopj15auZB-l80/view?usp=sharing

Anexo 3. Documento de Especificación de Requisitos Norma IEEE 830

[Requisitos funcionales y no funcionales]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

1. Introducción

El presente documento es la especificación de los requisitos funcionales, no funcionales y las plataformas de desarrollo del sistema que automatiza el diseño de casos de prueba funcionales basándose en casos de uso. Esta herramienta, desarrollada como proyecto de titulación en la Carrera de Computación de la Universidad Nacional de Loja, está diseñada para automatizar y facilitar la creación de casos de prueba funcionales a partir de la información contenida en los casos de uso de proyectos de software.

2. Objetivo

Los requisitos funcionales, no funcionales y plataformas de desarrollo son un entregable vital en el desarrollo de un software, es por eso que el objetivo de este documento es brindar una descripción amplia y detallada de las características requeridas en las funcionalidades y limitaciones del sistema, así como las plataformas que se utilizarán para el desarrollo del sistema. Esto garantizará que cada consideración crítica dentro del sistema sea evaluada y manejada apropiadamente durante el proceso del desarrollo, y por lo tanto asegurar una implementación exitosa y efectiva del sistema para diseñar casos de prueba funcionales a partir de casos de uso.

3. Actores

Esta sección describe los actores que interactúan con el sistema y sus roles. Cada actor puede representar un usuario final, un sistema externo o cualquier entidad que interactúe de alguna manera con el sistema.

Tabla 1: Actores del sistema.

Actor	Descripción	Ejemplo de Interacción
Administrador	Usuario con rol Administrador con privilegios para gestionar usuarios del sistema.	Asignar roles de usuario
Tester	Usuario con rol de Tester que interactúa con el sistema y ejecuta las funcionalidades propósito del sistema.	Gestión de casos de uso y prueba

4. Listado de Requisitos Funcionales y No Funcionales.

A continuación, se detallan los requerimientos generados:

- **Requisitos Funcionales**

Tabla 2: Requisitos Funcionales.

Código	Nombre	Descripción	Actor	Prioridad
RF001	Autenticar usuario	Debe permitir la autenticación (iniciar sesión) a usuarios mediante validación de	Todos los usuarios	Alta

		credenciales (correo y contraseña).		
RF002	Registrarse	Debe permitir a usuarios generales registrarse con un correo electrónico único y la siguiente información: • Nombre • Apellido • Contraseña Y su rol por defecto será Tester.	Usuario No registrado	Alta
RF003	Recuperar contraseña	Debe permitir la recuperación de contraseña en caso de olvido.	Todos los usuarios	Alta
RF004	Actualizar el perfil del usuario	Debe permitir la modificación de su perfil, se puede modificar: • Nombre • Apellido Imagen	Usuario Estándar / Administrador	Baja
RF005	Actualizar roles de usuario	Debe permitir la actualización de roles de usuarios	Administrador	Alta
RF006	Buscar usuarios por correo electrónico	Debe permitir buscar la información de usuarios mediante correo electrónico	Administrador	Media
RF007	Desactivar usuarios	Permitir desactivar usuarios registrados.	Administrador	Alta
RF008	Crear proyectos	Debe permitir el registro de nuevos proyectos con la siguiente información: • Nombre • Descripción • Imagen	Tester	Alta
RF009	Actualizar proyectos	Debe permitir la actualización de proyectos con la siguiente información: • Nombre • Descripción • Imagen	Tester	Alta
RF010	Buscar Proyectos por Nombre	Debe permitir visualizar la información de proyectos (Nombre, Descripción, Imagen)	Tester	Media
RF011	Eliminar proyectos	Permitir eliminar sus proyectos del sistema.	Tester	Alta
RF012	Compartir proyectos	Debe ofrecer la opción de colaboración a través de compartir proyectos con otros usuarios testers	Tester	Media

		mediante una invitación al correo electrónico		
RF013	Limitar a 10 el número máximo de proyectos a crear por usuario	Debe controlar el número de proyectos que un usuario Tester puede crear	Sistema	Media
RF014	Crear casos de uso	Debe permitir el registro de nuevos casos de uso a un proyecto con la siguiente información: • Nombre • Descripción • Entradas • Precondiciones • Postcondiciones • Flujo Normal • Flujo Alterno	Tester	Alta
RF015	Actualizar casos de uso	Debe permitir la actualización de casos de uso de un proyecto con la siguiente información: • Nombre • Descripción • Entradas • Precondiciones • Postcondiciones • Flujo Normal • Flujo Alterno	Tester	Alta
RF016	Buscar Casos de uso	Debe permitir visualizar la información de casos de uso de un proyecto (Nombre, Descripción, Entradas)	Tester	Media
RF017	Eliminar casos de uso	Permitir la eliminación de casos de uso de un proyecto del sistema.	Tester	Alta
RF018	Crear casos de prueba funcionales a partir de casos de uso	Debe permitir generar casos de prueba funcionales en base a un caso de uso.	Tester	Alta
RF019	Actualizar casos de prueba funcionales diseñados	Actualizar la información de un caso de prueba funcional, con la siguiente información: • Nombre • Descripción • Pasos • Entradas • Resultado Esperado	Tester	Alta
RF020	Eliminar Casos de prueba funcionales	Permitir la eliminación de un caso de prueba	Tester	Alta

		funcional de un caso de uso en el sistema.		
RF021	Buscar casos de prueba funcionales por Nombre	Debe permitir visualizar la información de los casos de prueba funcionales (Nombre, Descripción)	Tester	Media
RF022	Generar reportes de proyectos en formato IEEE 829	Debe ofrecer la funcionalidad de exportación de reportes en el formato estándar IEEE 829 en PDF	Tester	Media

- **Requisitos No Funcionales**

Tabla 3: Requisitos No Funcionales

Categoría	Código	Descripción
Seguridad	RNF001	Cierre automático de sesión tras 15 minutos de inactividad.
	RNF002	Cifrar las claves de cuentas de usuario
Usabilidad	RNF003	El diseño del sistema debe ser de tipo Responsive y se adapte a todo tamaño de pantalla
	RNF004	Mostrar una guía de cómo crear casos de prueba funcionales con la herramienta
	RNF005	La aplicación web debe contar con interfaces amigables e intuitivas, que permitan realizar el proceso de estimación de costos de manera sencilla.
Interfaz Gráfica de Usuario	RNF006	Modo oscuro y claro.
Eficiencia	RNF007	Optimizar el rendimiento para que las operaciones más comunes se ejecuten en menos de 5 segundos
Disponibilidad	RNF008	La aplicación web debe estar disponible dentro del laboratorio de software, siempre y cuando el estado de la infraestructura de los servidores de la carrera de Computación sea óptimo y estén en funcionamiento.

5. Plataformas de Desarrollo

La siguiente sección describe las plataformas de desarrollo seleccionadas para implementar el sistema. Cada tecnología cumple un rol específico en el manejo de diferentes partes del sistema, con el objetivo de asegurar una estructura organizada, eficiente y

compatible entre sus componentes. Esto permitirá una integración armoniosa que facilitará el mantenimiento y la escalabilidad del sistema.

5.1. Entorno de Desarrollo.

Tabla 4: Entorno de Desarrollo.

Categoría	Especificación	Versión	Descripción
Lenguaje de Programación	Typescript	^5	Lenguaje principal para el desarrollo frontend y backend, proporcionando tipado estático y mejores prácticas de desarrollo
Framework Frontend	Next.js	14.2.11	Framework de React para desarrollo con funcionalidades de SSR y API routes
Framework Backend	Nest.js	^10.0.0	Framework de Express para desarrollo con funcionalidades de SSR y API routes
Runtime	Node.js	v20.9.0	Entorno de ejecución para JavaScript/TypeScript en el servidor

5.2. Frameworks y Librerías Principales.

Tabla 5: Frameworks y Librerías Principales.

Librería/Framework	Versión	Propósito	Dependencias Críticas
React	18.x	UI Framework	Ninguna
Tailwind CSS	3.x	Framework de estilos	PostCSS
Prisma	5.x	ORM para base de datos	Ninguna
Shadcn/ui	1.x	Componentes de UI	Tailwind CSS, RadixUI

5.3. Herramientas de Desarrollo.

Tabla 6: Herramientas de Desarrollo.

Herramienta	Versión	Propósito	Configuración Requerida
VSCode	Latest	IDE	Extensiones recomendadas: ESLint, Prettier
Git	2.x	Control de Versiones	Configuración de usuario
Npm	8.x	Gestor de paquetes	Configuración personalizada
Prettier	3.x	Formateo del Código	Configuración personalizada

5.4. Infraestructura y Despliegue

Tabla 7: Infraestructura y Despliegue.

Herramienta	Versión	Propósito	Configuración Requerida
Base de Datos	18.x	UI Framework	Ninguna
Servidor de Backend	3.x	Framework de estilos	PostCSS
Servidor de Frontend	5.x	ORM para base de datos	Ninguna

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 4. Diagrama de Casos de Uso y Narración.

[Casos de Uso]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

1. Introducción

El presente documento es el detalle de los casos de uso del sistema que automatiza el diseño de casos de prueba funcionales basándose en casos de uso. Esta herramienta, desarrollada como proyecto de titulación en la Carrera de Computación de la Universidad Nacional de Loja, está diseñada para automatizar y facilitar la creación de casos de prueba funcionales a partir de la información contenida en los casos de uso de proyectos de software.

2. Objetivo

Los casos de uso son un entregable vital en el desarrollo de un software, es por eso que el objetivo de este documento es brindar una descripción amplia y detallada de las características requeridas en las funcionalidades y limitaciones del sistema. Esto garantizará que cada consideración crítica dentro del sistema sea evaluada y manejada apropiadamente durante el proceso del desarrollo, y por lo tanto asegurar una implementación exitosa y efectiva del sistema para diseñar casos de prueba funcionales a partir de casos de uso.

3. Diagrama de Casos de Uso.

A continuación, se detallan los casos de uso generados:

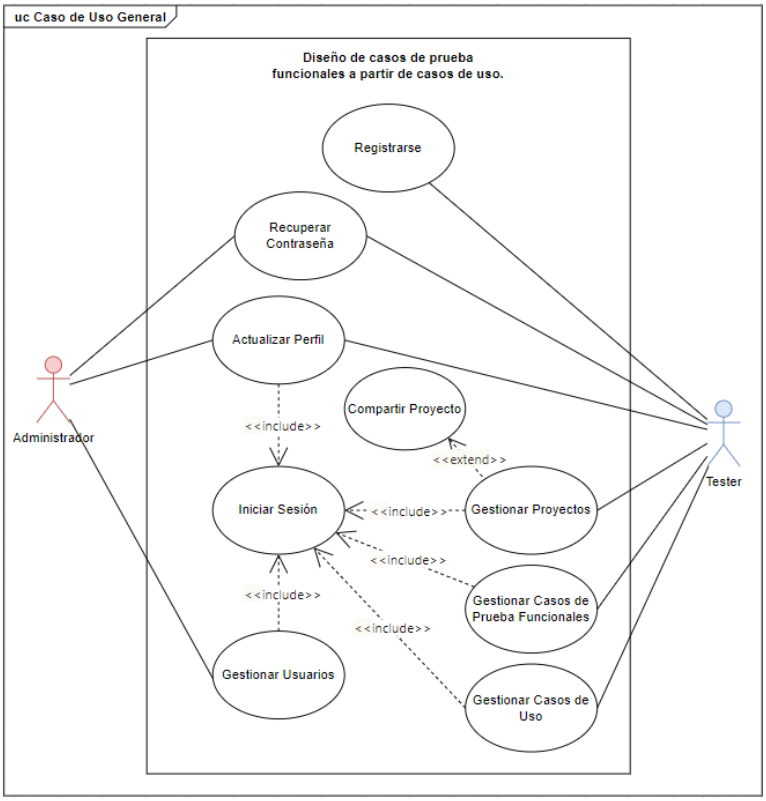


Figura 1: Casos de Uso General.

4. Narración de Casos de Uso.

Tabla 1: Caso de Uso 1.

Identificador	**CU01**
Nombre	Iniciar sesión
Actores	Todos los usuarios del sistema
Descripción	Autenticación de usuario en el sistema
Precondiciones	El usuario debe estar registrado. Estar ubicado en la página de “Iniciar sesión”
Postcondiciones	El usuario accede a la interfaz principal del sistema.
Flujo Principal	El usuario ingresa sus credenciales de acceso (correo y contraseña) El usuario pulsa el botón Iniciar Sesión El sistema verifica las credenciales. El sistema autentica al usuario y accede al sistema.
Flujo Alterno	**Credenciales Incorrectas** A1: Si el usuario ingresa una contraseña incorrecta, el sistema muestra un mensaje de error. **Intentos Permitidos** A2: Si el usuario excede el número de intentos permitidos, el sistema bloquea la cuenta temporalmente. **Campos Vacíos** A3: Si el usuario pulsa el botón Iniciar Sesión sin llenar los campos, muestra un mensaje indicando que hay campos vacíos en el formulario

Tabla 2: Caso de Uso 2.

Identificador	**CU02**
Nombre	Registrarse
Actores	Usuario No Registrado
Descripción	Un usuario no registrado registra sus datos
Precondiciones	Ubicado en la página de Registro
Postcondiciones	El usuario es registrado en el sistema y accede al sistema.
Flujo Principal	El usuario no registrado ingreso los datos necesarios (Nombre, Apellido, Correo, Contraseña) El sistema valida datos El sistema guarda los datos y confirma el registro.
Flujo Alterno	**Campos incorrectos**

	A1: Si el usuario no registrado intenta ingresar con datos incorrectos, el sistema muestra un mensaje de error. **Campos obligatorios vacíos** A2: Si el usuario no registrado pulsa el botón Registrar sin llenar los campos, muestra un mensaje indicando que hay campos vacíos en el formulario **Correo existente** A3: Si el usuario no registrado ingresa un correo ya existente, el sistema emite un mensaje de error indicando que ya existe un usuario con ese correo

Tabla 3: Caso de Uso 3.

Identificador	**CU03**
Nombre	Recuperar contraseña
Actores	Todos los usuarios del sistema
Descripción	Permite al usuario recuperar su contraseña mediante una contraseña generada aleatoriamente y enviada al correo del usuario del que se solicitó la recuperación.
Precondiciones	El usuario debe proporcionar el correo registrado. El usuario debe estar registrado
Postcondiciones	Se envía una contraseña temporal (OTP) al correo para usarla para el ingreso a la cuenta
Flujo Principal	El usuario ingresa su correo en la opción de recuperación de contraseña. El sistema envía un OTP al correo del usuario El usuario coloca la OTP en la ventana modal mostrada El usuario inicia sesión en el sistema y es redirigido a la página de personalización de perfil
Flujo Alterno	**Correo no registrado** A1: Si el correo no está registrado, el sistema muestra un mensaje de error. **OTP incorrecta** Al ingresar una OTP que no corresponda a la enviada en el correo, se muestra un mensaje de error

Tabla 4: Caso de Uso 4.

Identificador	**CU04**
Nombre	Actualizar perfil de usuario
Actores	Todos los usuarios del sistema
Descripción	Permite a los usuarios personalizar y actualizar su perfil en el sistema.
Precondiciones	El usuario debe estar autenticado.
Postcondiciones	Los cambios en el perfil son guardados.
Flujo Principal	El usuario selecciona la opción de personalizar perfil. El usuario realiza los cambios deseados, en cualquier campo (Nombre, Apellido, Imagen, Contraseña) El sistema valida los cambios El sistema guarda los cambios en el perfil.
Flujo Alterno	**Datos Incorrectos** A1: El usuario ingresa datos incorrectos, se muestra un mensaje de error. **Campos Vacíos** A2: El usuario pulsa el botón de actualizar datos, con los campos del formulario vacíos, se muestra un mensaje de error.

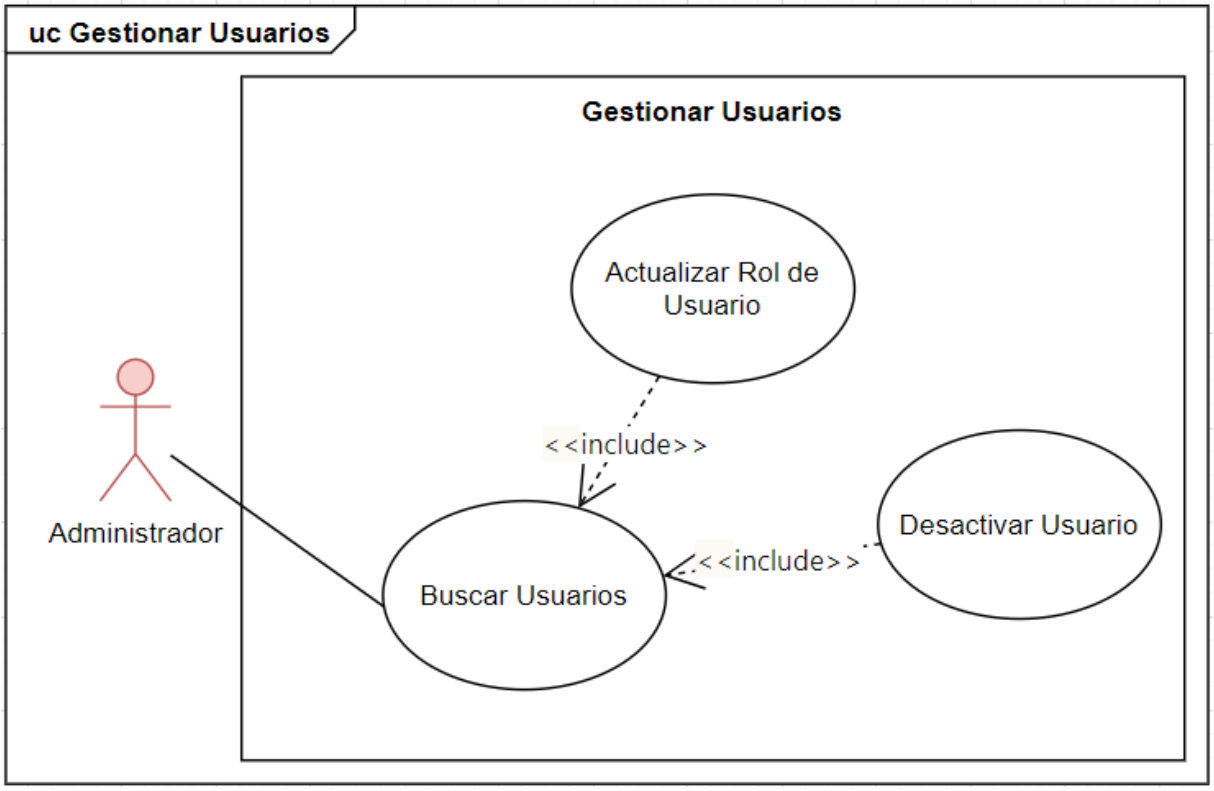


Figura 2: Caso de Uso Usuarios.

Tabla 5: Caso de Uso 5.

Identificador	**CU05**
Nombre	Actualizar roles de usuario
Actores	Administrador
Descripción	Modificar el rol de un usuario
Precondiciones	El administrador debe estar autenticado Ubicado en la página de Listar Usuarios Buscar usuario
Postcondiciones	Los datos del usuario son actualizados.
Flujo Principal	El administrador selecciona el usuario a modificar. El administrador accede a la opción de editar El administrador actualiza el rol El sistema valida datos El sistema guarda los cambios y confirma la actualización.

Tabla 6: Caso de Uso 6.

Identificador	**CU06**
Nombre	Buscar usuarios
Actores	Administrador
Descripción	El sistema permite buscar usuarios por correo
Precondiciones	El administrador debe estar autenticado.
Postcondiciones	Se visualiza la información de los usuarios
Flujo Principal	El administrador ingresa el correo en el campo ubicado en la parte superior de la lista de usuarios El sistema busca y muestra la información del usuario
Flujo Alterno	**No existen Usuarios** A1: Si el correo no existe, el sistema muestra un mensaje indicando que no existen datos.

Tabla 7: Caso de Uso 7.

Identificador	**CU07**
Nombre	Desactivar usuario
Actores	Administrador
Descripción	Desactivar a un usuario.
Precondiciones	El administrador debe estar autenticado. Ubicado en la página de Listar Usuarios
Postcondiciones	El usuario queda desactivado del sistema.
Flujo Principal	El administrador selecciona al usuario a desactivar. El sistema selecciona la opción desactivar

	El sistema muestra una confirmación de acción. El administrador pulsa el botón “Acepto” El sistema desactiva el usuario y muestra un mensaje confirmando la eliminación.
Flujo Alterno	**Cancelar Operación** A1: Si el administrador pulsa el botón de Cancelar, el sistema cancela la operación.

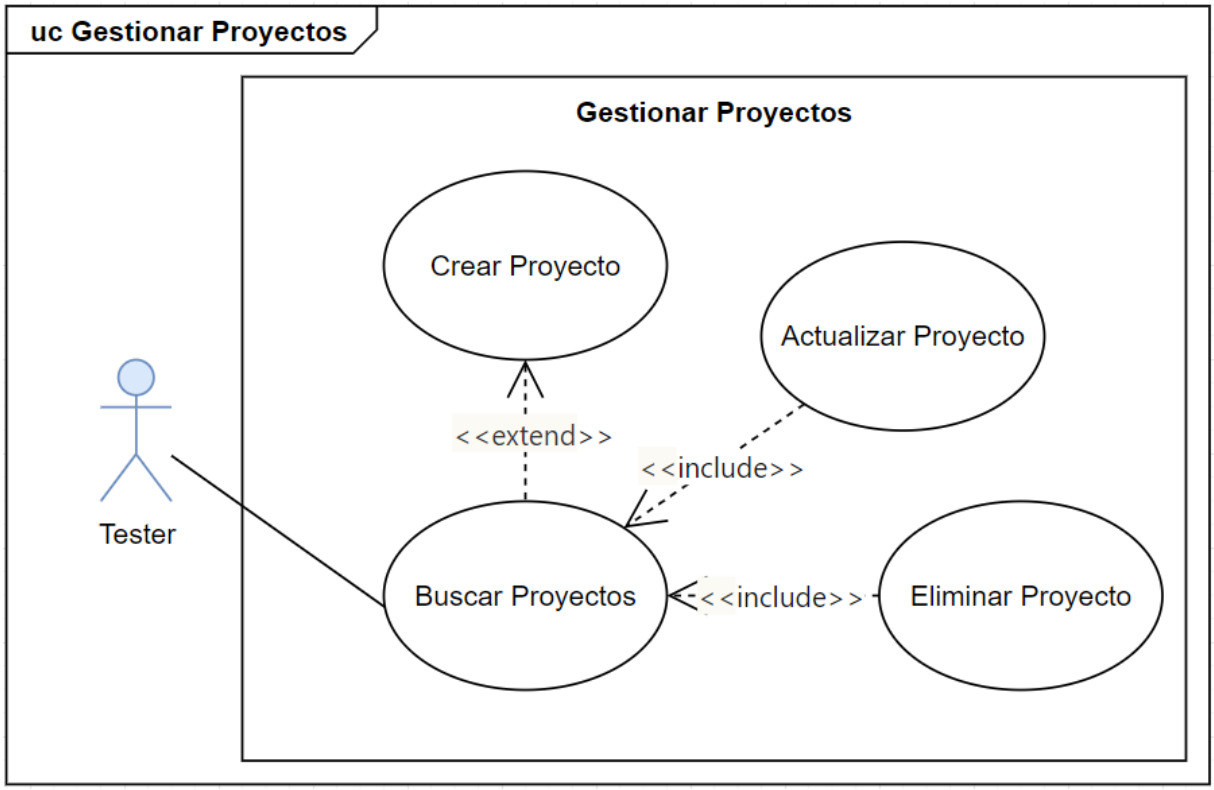


Figura 3: Caso de Uso Proyectos.

Tabla 8: Caso de Uso 8.

Identificador	**CU08**
Nombre	Crear proyecto
Actores	Tester
Descripción	Crear un nuevo proyecto.
Precondiciones	El Tester debe estar autenticado. Ubicado en la página de Listar Proyectos
Postcondiciones	El proyecto se agrega a la lista de proyectos del usuario.
Flujo Principal	El Tester pulsa el botón de “Crear proyecto” El sistema muestra el formulario de crear proyecto El Tester ingresa la información del nuevo proyecto (nombre, descripción, imagen)

	El Tester pulsa el botón de Crear El sistema valida los datos ingresados. El sistema guarda el nuevo proyecto y muestra un mensaje de confirmación.
Flujo Alterno	**Límite de proyectos** A1: Si el Tester excede el límite de proyectos, el sistema muestra un mensaje de advertencia. **Campos vacíos** A2: El Tester no ingresa todos los campos requeridos, el sistema muestra un mensaje de error.

Tabla 9: Caso de Uso 9.

Identificador	**CU09**
Nombre	Actualizar proyecto
Actores	Tester
Descripción	Modificar la información de un proyecto ya existente.
Precondiciones	Ubicado en página Listar Proyectos
Postcondiciones	El proyecto se actualiza con los datos ingresados.
Flujo Principal	El Tester selecciona el proyecto a modificar. El sistema muestra el formulario de actualización del proyecto. El Tester actualiza los datos necesarios (cualquiera de estos parámetros: Nombre, Descripción, Imagen) El sistema valida los datos ingresados El sistema guarda los cambios en el proyecto.
Flujo Alterno	**Campos incorrectos** A1: Si hay errores en los datos, el sistema muestra un mensaje de error. **Campos vacíos** A2: Si algún campo del formulario está vacío, se muestra un mensaje de error.

Tabla 10: Caso de Uso 10.

Identificador	**CU010**
Nombre	Buscar proyectos
Actores	Tester
Descripción	Permite buscar proyectos por nombre
Precondiciones	El usuario debe estar autenticado.
Postcondiciones	Se visualiza la información de los proyectos

Flujo Principal	El Tester ingresa el nombre en el campo ubicado en la parte superior de la lista de proyectos El sistema busca y muestra la información del proyecto
Flujo Alterno	**No existen resultados** A1: Si no se encuentran coincidencias, el sistema muestra un mensaje indicando la falta de resultados.

Tabla 11: Caso de Uso 11.

Identificador	**CU011**
Nombre	Eliminar proyecto
Actores	Tester
Descripción	Permite la eliminación de un proyecto
Precondiciones	Debe existir un proyecto registrado. Ubicado en la página de Listar proyectos
Postcondiciones	Se elimina el proyecto
Flujo Principal	El usuario selecciona un proyecto para eliminar El usuario pulsa la opción Eliminar El sistema muestra una confirmación de acción El usuario selecciona Acepto El sistema elimina el proyecto
Flujo Alterno	**Cancelar Operación** A1: El usuario pulsa el botón de cancelar en la confirmación de acción, el sistema cancela la operación.

Tabla 12: Caso de Uso 12.

Identificador	**CU012**
Nombre	Compartir proyectos
Actores	Tester
Descripción	Compartir proyectos con otros usuarios testers.
Precondiciones	El usuario debe tener un proyecto creado. Ubicado en página de Listar Proyectos
Postcondiciones	Se genera un código de acceso para el usuario invitado.
Flujo Principal	El usuario selecciona un proyecto para compartir. El usuario selecciona la opción de “Compartir Proyecto”

	El sistema abre un modal para seleccionar los usuarios con los que se desea compartir el proyecto. El usuario busca y selecciona el usuario al que desea compartir el proyecto El usuario pulsa el botón compartir El sistema envía una confirmación a los usuarios seleccionados
Flujo Alterno	**Usuario no seleccionado** A1: El usuario no selecciona un usuario con quien compartir el proyecto, y pulsa el botón compartir, por tanto, el sistema muestra un mensaje de error. **Cancelar Operación** A2: El usuario pulsa el botón de cancelar en el modal de seleccionar los usuarios, el sistema cancela la operación.

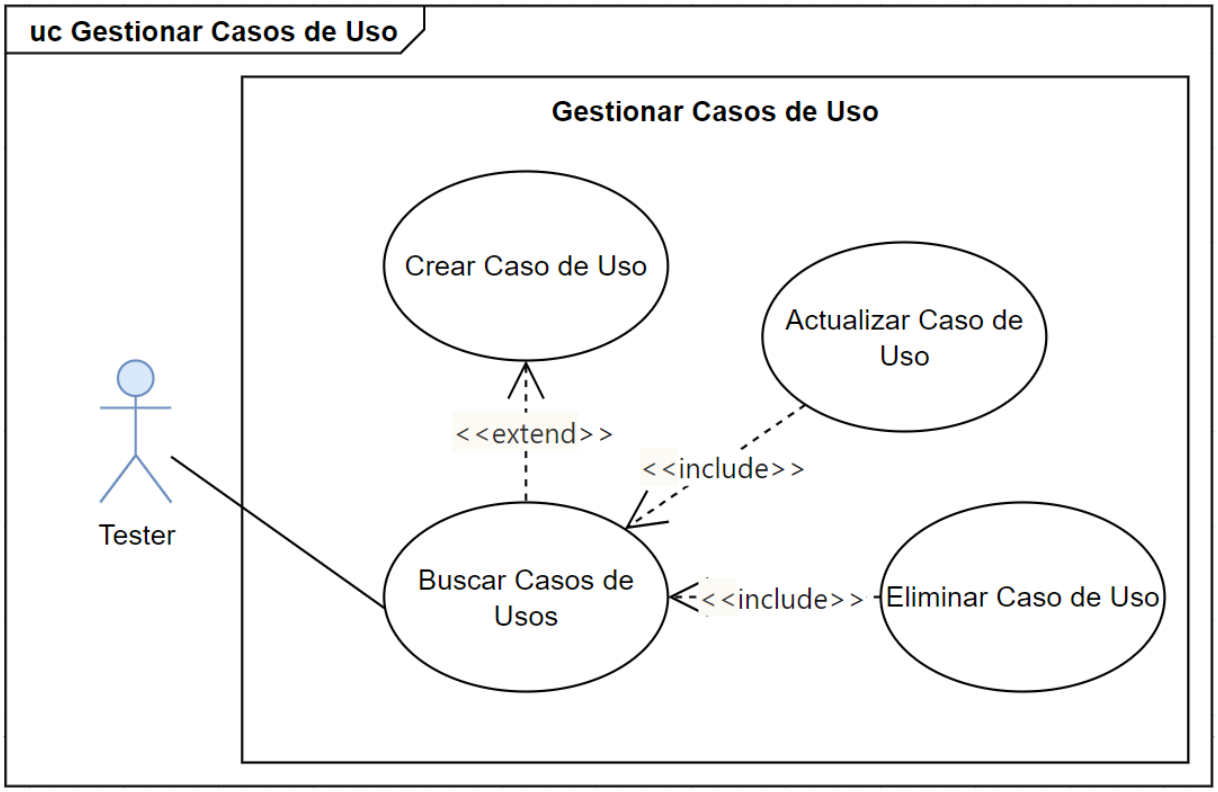


Figura 4: Caso de Uso de Casos de Uso.

Tabla 13: Caso de Uso 13.

Identificador	**CU013**
Nombre	Crear Caso de uso
Actores	Tester
Descripción	Crear un nuevo Caso de uso en un proyecto

Precondiciones	El Tester debe estar autenticado. Ubicado en la página de Listar Casos de usos
Postcondiciones	El Caso de uso se agrega a la lista de proyectos del Tester.
Flujo Principal	El Tester pulsa el botón de “Crear Caso de uso” El sistema muestra el formulario de crear Caso de uso El Tester ingresa la información del nuevo Caso de uso (nombre, descripción, entradas, flujo normal, flujo alterno) El Tester pulsa el botón de Crear El sistema valida los datos ingresados. El sistema guarda el nuevo Caso de uso y muestra un mensaje de confirmación.
Flujo Alterno	**Campos vacíos** A1: El Tester no ingresa todos los campos requeridos, el sistema muestra un mensaje de error.

Tabla 14: Caso de Uso 14.

Identificador	**CU014**
Nombre	Actualizar caso de uso
Actores	Tester
Descripción	Modificar la información de un caso de uso ya existente.
Precondiciones	Debe existir un caso de uso Ubicado en página Listar casos de usos
Postcondiciones	El caso de uso se actualiza con los datos ingresados.
Flujo Principal	El Tester selecciona el caso de uso a modificar. El sistema muestra el formulario de actualización del caso de uso El Tester actualiza los datos necesarios (cualquiera de estos: Nombre, Descripción, entradas, flujo normal, flujo alterno) El sistema valida los datos ingresados El sistema guarda los cambios en el caso de uso
Flujo Alterno	**Campos incorrectos** A1: Si hay errores en los datos, el sistema muestra un mensaje de error. **Campos vacíos**

	A2: Si algún campo del formulario está vacío, se muestra un mensaje de error.

Tabla 15: Caso de Uso 15.

Identificador	**CU015**
Nombre	Buscar casos de usos
Actores	Tester
Descripción	Permite buscar casos de usos por nombre
Precondiciones	El Tester debe estar autenticado.
Postcondiciones	Se visualiza la información de los casos de usos
Flujo Principal	El Tester ingresa el nombre en el campo ubicado en la parte superior de la lista de casos de uso El sistema busca y muestra la información del usuario
Flujo Alterno	**No existen resultados** A1: Si no se encuentran coincidencias, el sistema muestra un mensaje indicando la falta de resultados.

Tabla 16: Caso de Uso 16.

Identificador	**CU016**
Nombre	Eliminar caso de uso
Actores	Tester
Descripción	Permite la eliminación de un caso de uso
Precondiciones	Debe existir un proyecto registrado. Ubicado en la página de Listar casos de usos
Postcondiciones	Se elimina el caso de uso
Flujo Principal	El Tester selecciona un caso de uso para eliminar El Tester pulsa la opción Eliminar El sistema muestra una confirmación de acción El Tester selecciona Acepto El sistema elimina el caso de uso
Flujo Alterno	**Cancelar Operación** A1: El Tester pulsa el botón de cancelar en la confirmación de acción, el sistema cancela la operación.

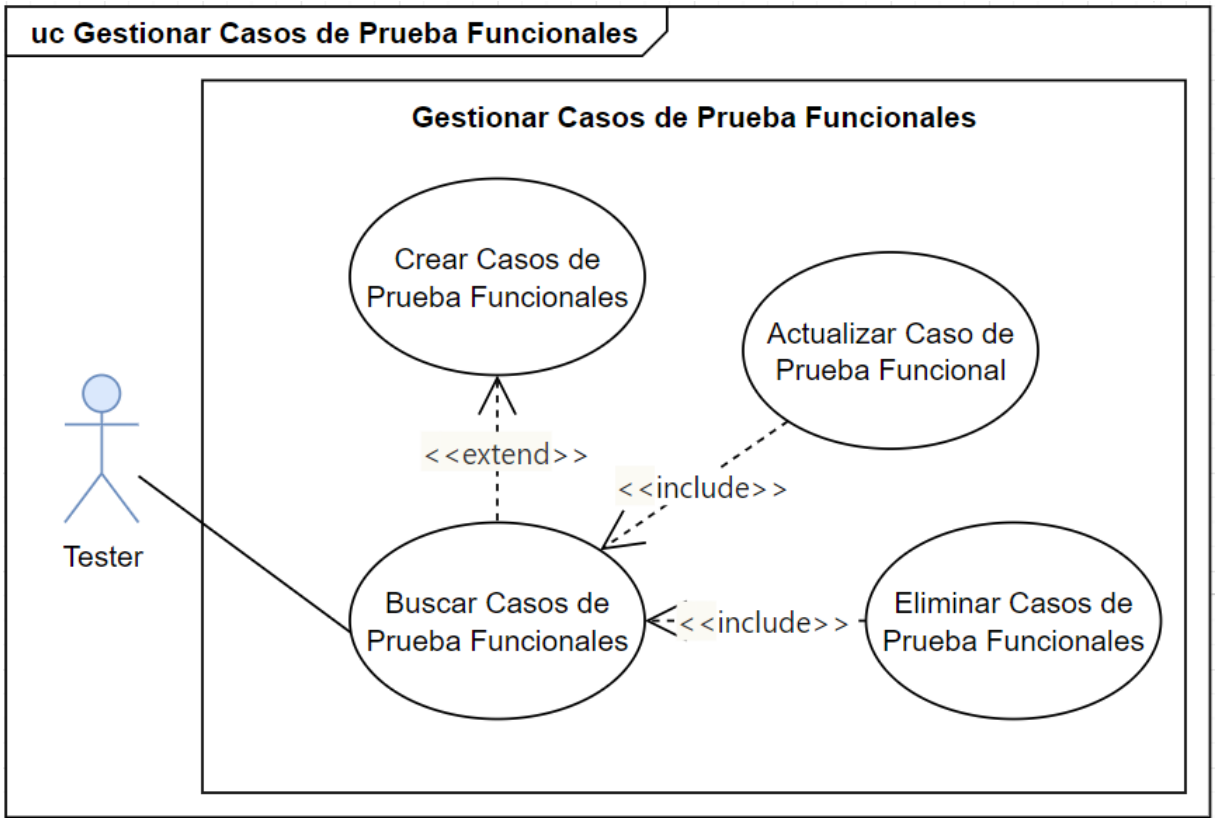


Figura 5: Caso de Uso de Casos de Prueba Funcionales.

Tabla 17: Caso de Uso 17.

Identificador	**CU017**
Nombre	Crear casos de prueba funcionales a partir de casos de uso
Actores	Tester
Descripción	Diseñar casos de prueba funcionales en base a un caso de uso.
Precondiciones	El Tester debe estar autenticado El Tester debe tener casos de uso registrados.
Postcondiciones	Se genera el caso de prueba funcional en el sistema.
Flujo Principal	El Tester selecciona un caso de uso. El Tester selecciona la opción crear Caso de Prueba Funcional El sistema crea los casos de prueba funcionales basado en el caso de uso. Los casos de prueba funcionales se almacenan y se muestran al Tester en la tabla correspondiente a Casos de Prueba Funcionales
Flujo Alterno	**Error en la petición**

	A1: Si existe un error en la comunicación de la aplicación con la API de la IA, no se puede generar el caso de prueba, el sistema muestra un mensaje de error. **Caso de Uso Inválido** A2: Si el caso de uso posee información inválida, el sistema muestra un mensaje de error

Tabla 18: Caso de Uso 18.

Identificador	**CU018**
Nombre	Actualizar caso de prueba funcional
Actores	Tester
Descripción	Permite la edición y personalización de los casos de prueba previamente diseñados.
Precondiciones	Debe existir un caso de prueba registrado.
Postcondiciones	Se guardan los cambios realizados en el caso de prueba.
Flujo Principal	El Tester selecciona un caso de prueba para editar. El Tester realiza las modificaciones. El sistema valida los datos ingresados El sistema guarda los cambios.
Flujo Alterno	**Campos incorrectos** A1: Si los cambios contienen errores, el sistema muestra un mensaje indicando los campos a corregir.

Tabla 19: Caso de Uso 19.

Identificador	**CU019**
Nombre	Eliminar caso de prueba funcional
Actores	Tester
Descripción	Permite la eliminación de un caso de prueba funcional
Precondiciones	Debe existir un caso de prueba funcional registrado. Ubicado en la página de Listar Casos de prueba funcionales
Postcondiciones	Se elimina el caso de prueba funcional del sistema.
Flujo Principal	El Tester selecciona un caso de prueba para eliminar El Tester pulsa la opción Eliminar El sistema muestra una confirmación de acción

	El Tester selecciona Acepto El sistema elimina el caso de prueba funcional.
Flujo Alterno	**Cancelar Operación** A1: El Tester pulsa el botón de cancelar en la confirmación de acción, el sistema cancela la operación.

Tabla 20: Caso de Uso 20.

Identificador	**CU020**
Nombre	Buscar casos de prueba funcionales
Actores	Tester
Descripción	El sistema permite buscar casos de prueba funcionales
Precondiciones	El Tester debe estar autenticado.
Postcondiciones	Se visualiza la información de los casos de prueba funcionales
Flujo Principal	El Tester ingresa el nombre en el campo ubicado en la parte superior de la lista de casos de pruebas funcionales El sistema busca y muestra la información del usuario
Flujo Alterno	**No existen casos de prueba funcionales** A1: Si existe fallo de comunicación con la base de datos y no encuentra casos de prueba funcionales, el sistema muestra un mensaje indicando que no existen datos.

Tabla 21: Caso de Uso 21.

Identificador	**CU021**
Nombre	Generar reportes de proyectos en formato IEEE 829
Actores	Tester
Descripción	El sistema permite generar reportes de los proyectos
Precondiciones	El Tester debe estar autenticado.
Postcondiciones	El Tester obtiene el reporte en un pdf descargado
Flujo Principal	El Tester pulsa el botón de generar reporte de proyecto El sistema genera un reporte del proyecto y envia a descargar un pdf al Tester

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 5. Prototipos.

[Prototipos]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

1. Introducción

El presente documento es el detalle de los prototipos del sistema que automatiza el diseño de casos de prueba funcionales basándose en casos de uso. Esta herramienta, desarrollada como proyecto de titulación en la Carrera de Computación de la Universidad Nacional de Loja, está diseñada para automatizar y facilitar la creación de casos de prueba funcionales a partir de la información contenida en los casos de uso de proyectos de software.

2. Objetivo

Los prototipos son un entregable vital en el desarrollo de un software, es por eso que el objetivo de este documento es brindar una descripción amplia y detallada de las características requeridas en las funcionalidades y limitaciones del sistema. Esto garantizará que cada consideración crítica dentro del sistema sea evaluada y manejada apropiadamente durante el proceso del desarrollo, y por lo tanto asegurar una implementación exitosa y efectiva del sistema para diseñar casos de prueba funcionales a partir de casos de uso.

3. Listado de Prototipos realizados por iteración:

- **Primera Iteración**

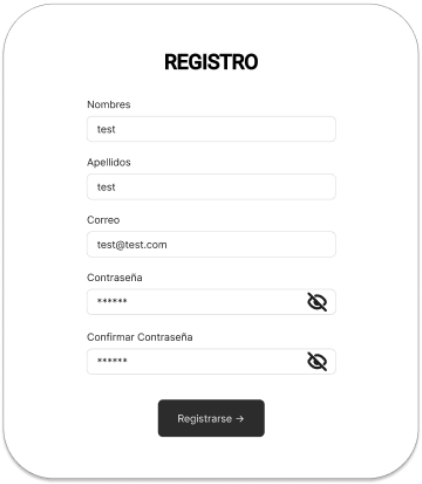


Figura 1. Prototipo registro de usuario

Figura 2. Prototipo inicio de sesión de usuario

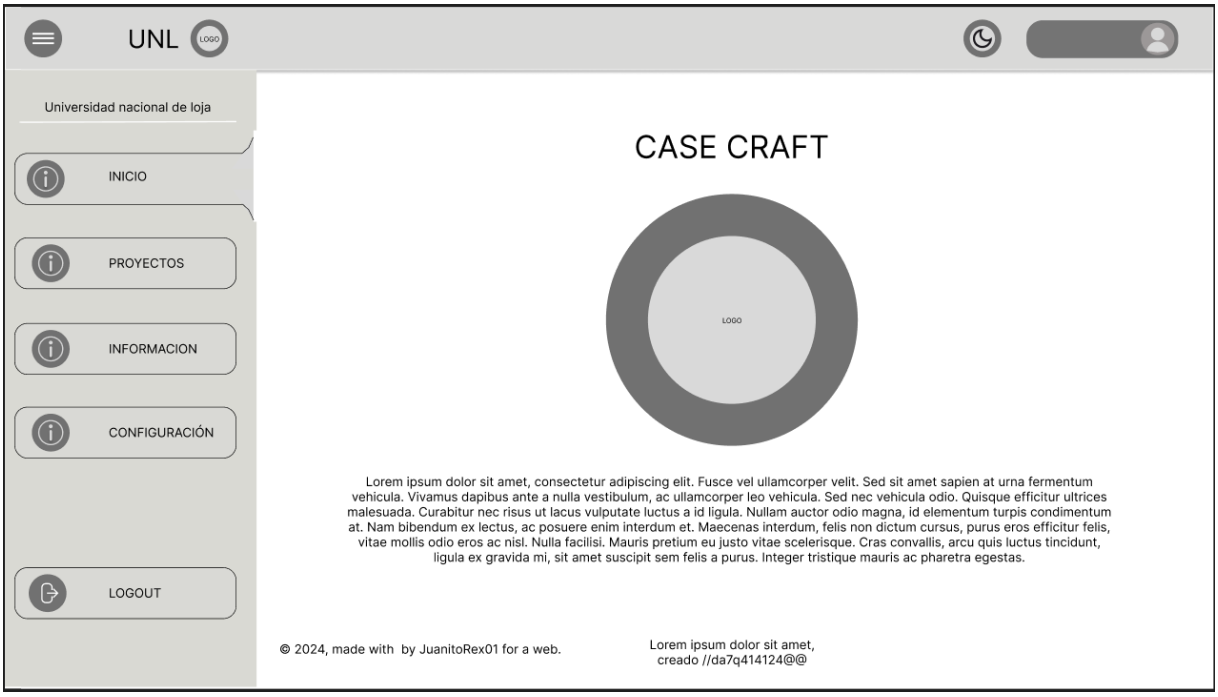


Figura 3. Prototipo página inicial del usuario

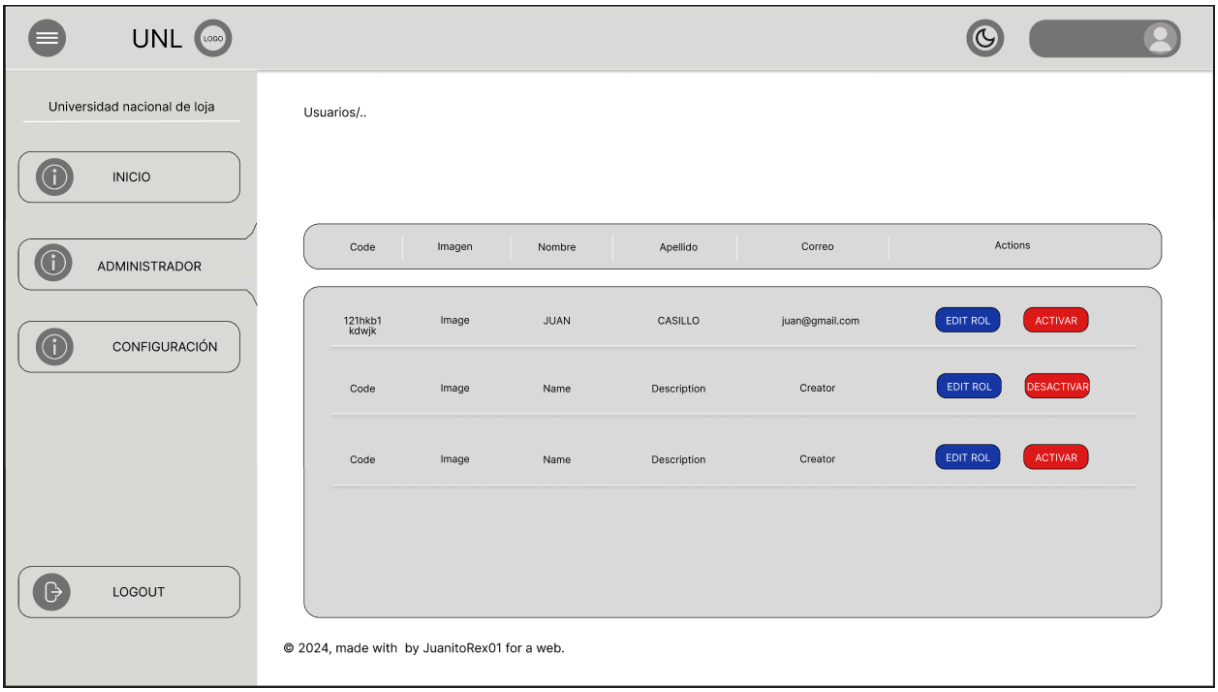


Figura 4. Prototipo página usuarios

Figura 5. Prototipo página cambiar rol del usuario

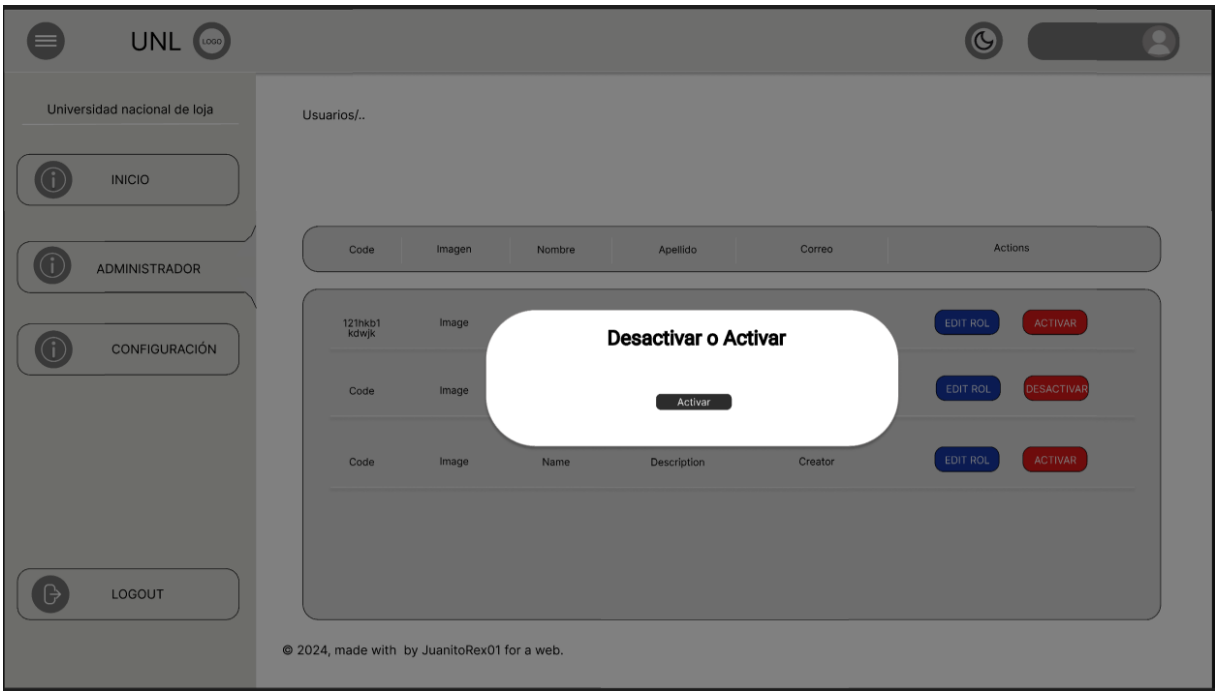


Figura 6. Prototipo página desactivar o activar el usuario

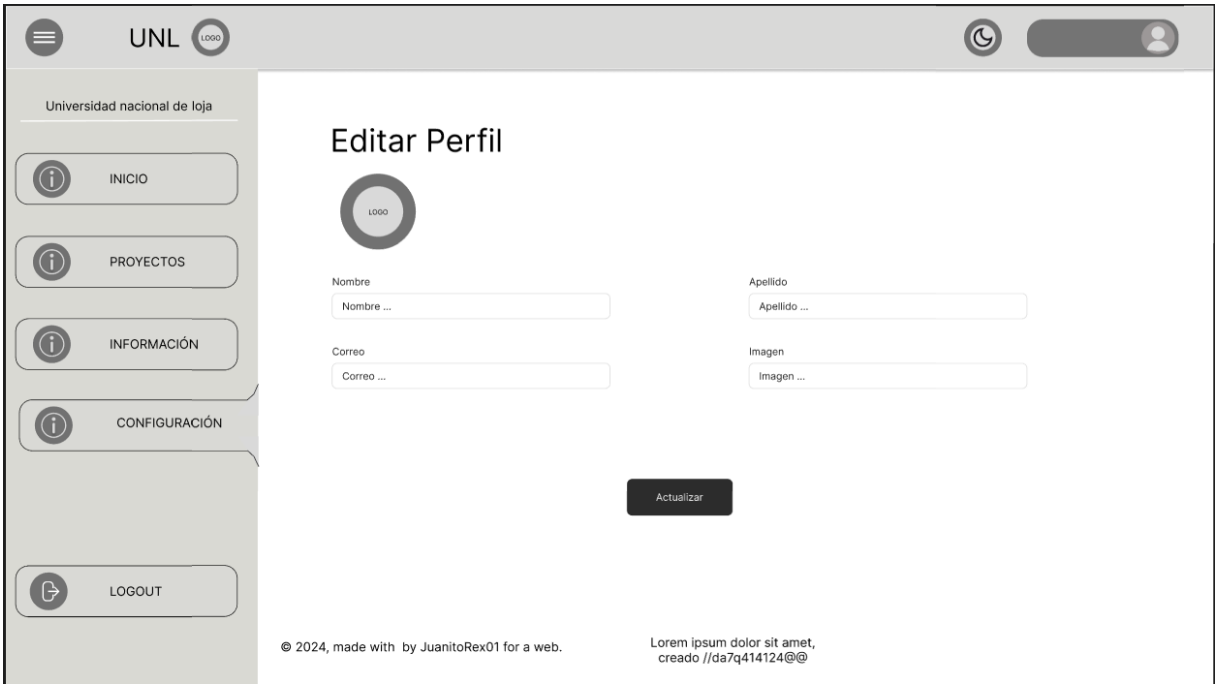


Figura 7. Prototipo página Editar Perfil

- **Segunda Iteración**

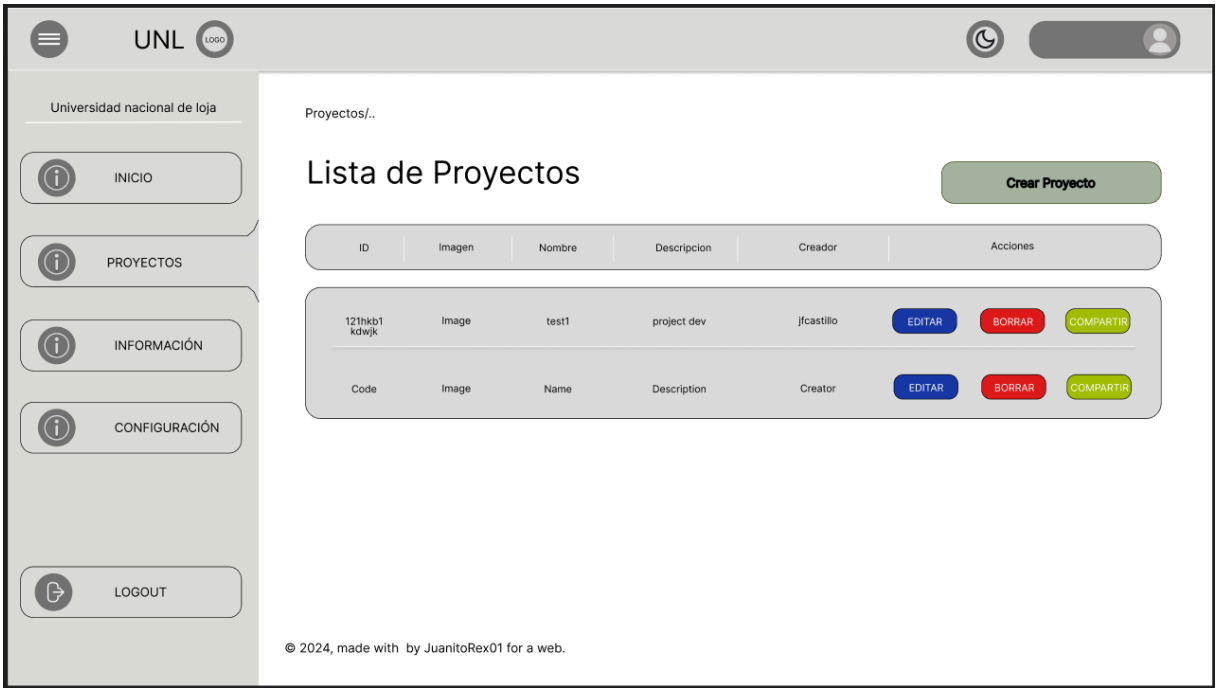


Figura 8. Prototipo página de proyectos del usuario

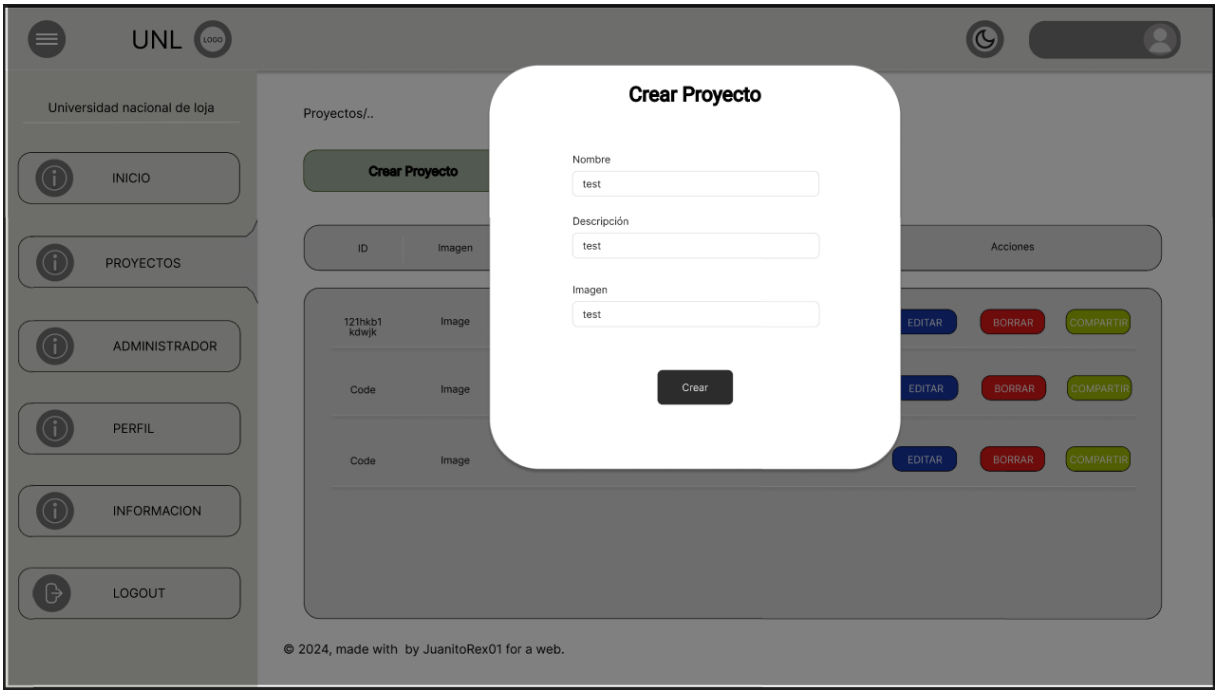


Figura 9. Prototipo página Crear Proyecto

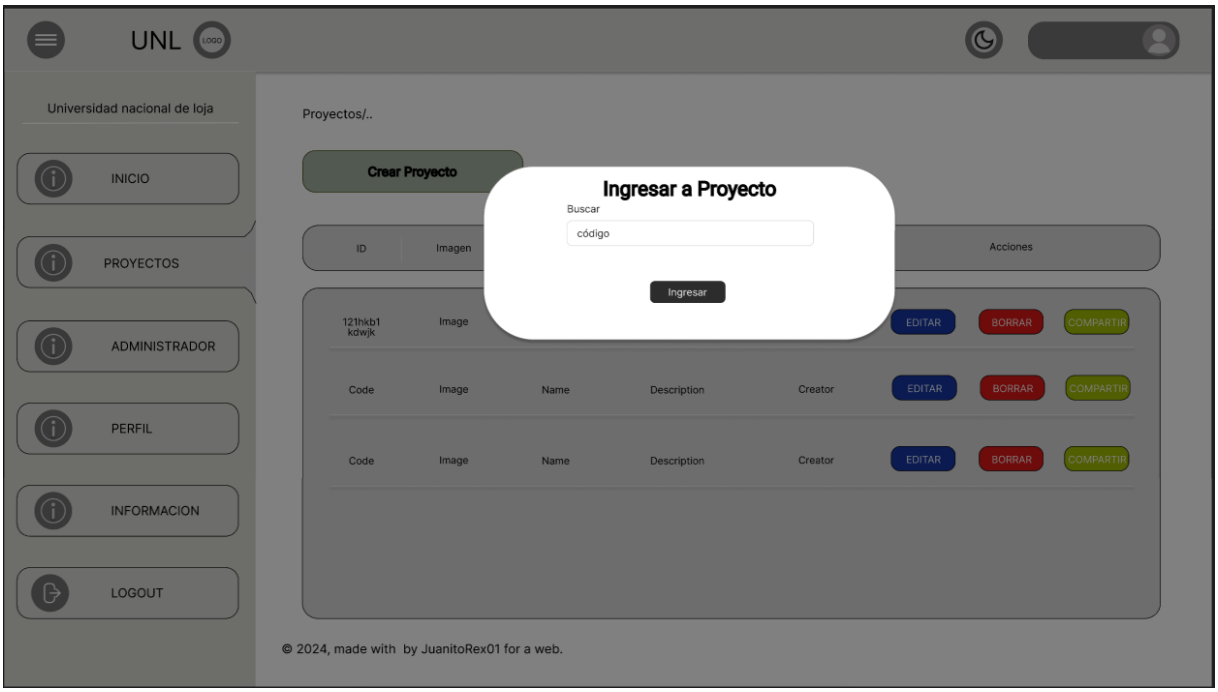


Figura 10. Prototipo página ingresar a Proyecto de un usuario

- **Tercera Iteración**

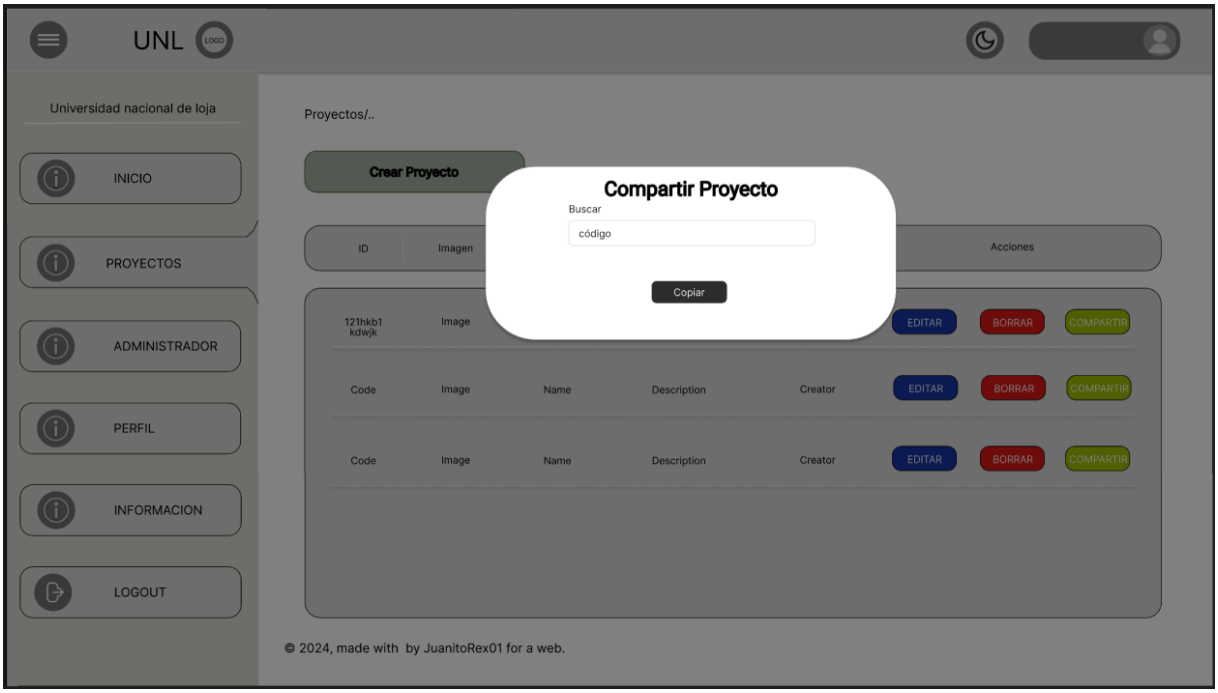


Figura 11. Prototipo página compartir proyecto del usuario

Figura 12. Prototipo página Casos de Uso de un proyecto

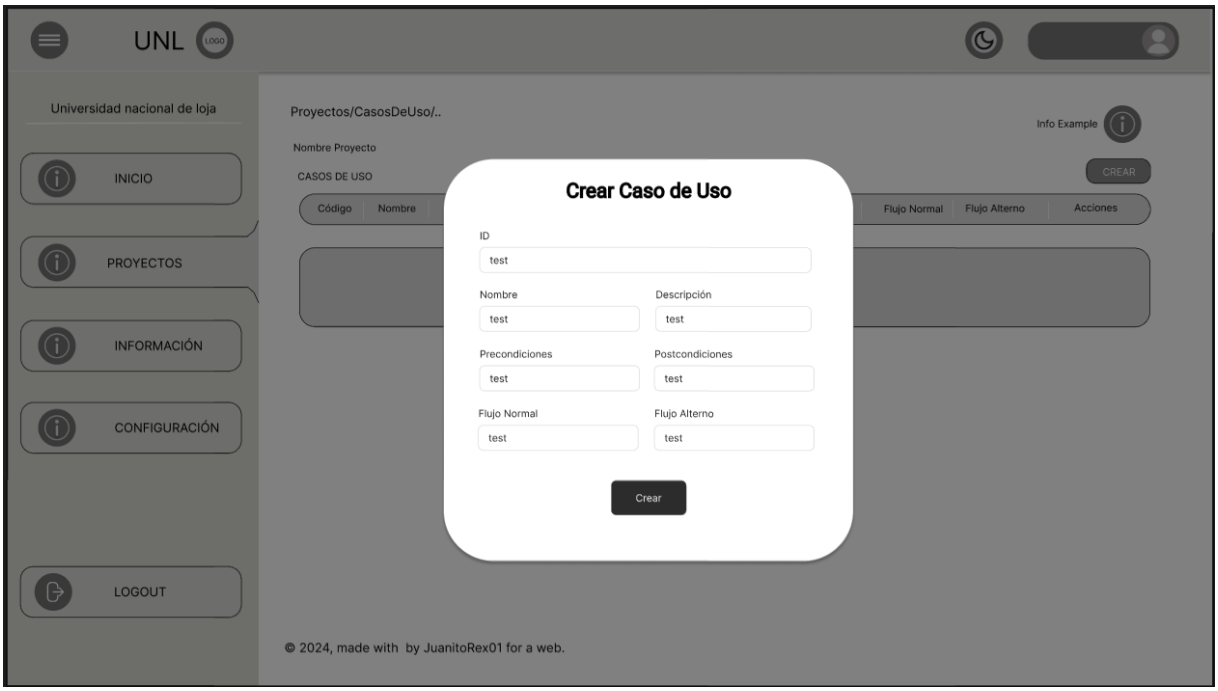


Figura 13. Prototipo página Crear Caso de Uso

- **Cuarta Iteración**

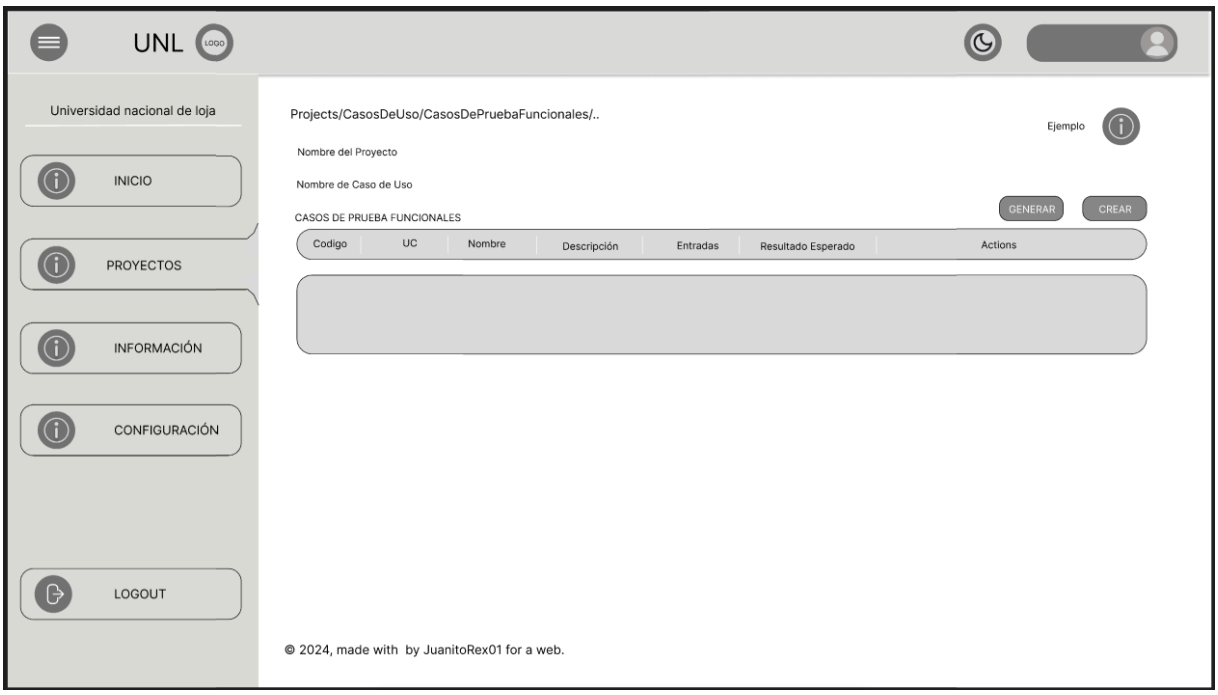


Figura 14. Prototipo página Casos de Prueba Funcionales

Figura 15. Prototipo página Generar casos de prueba Funcionales

Figura 16. Prototipo página Crear Caso de prueba Funcional

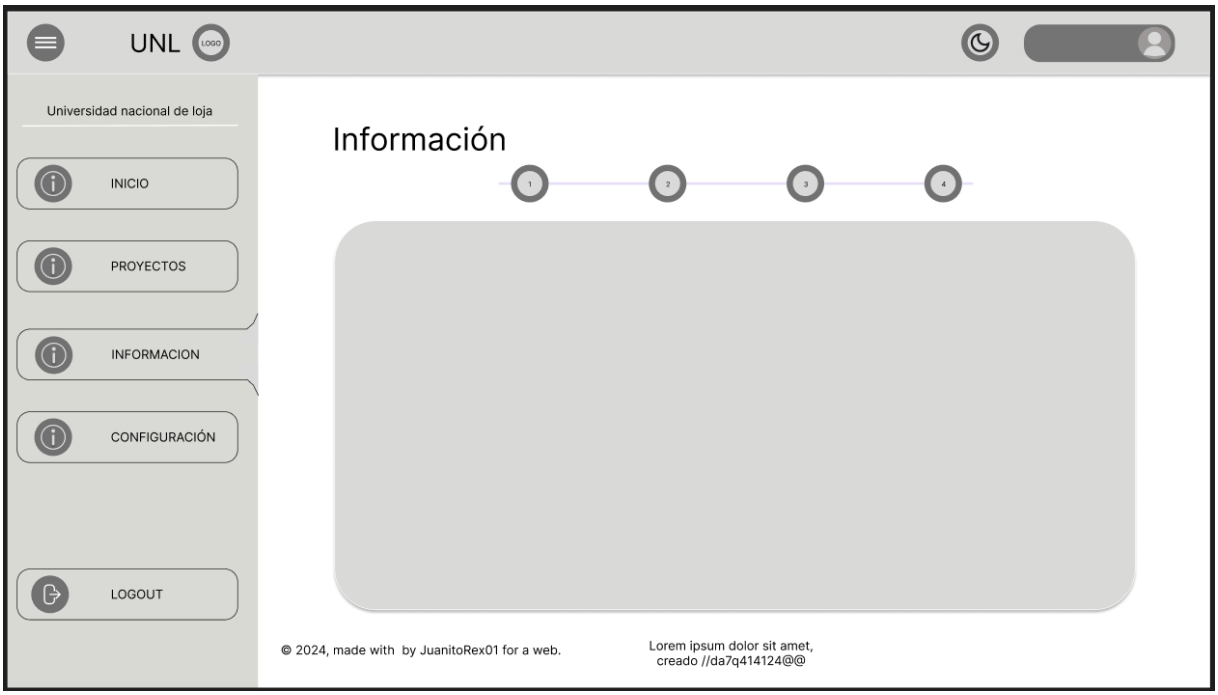


Figura 17. Prototipo página Información

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 6. Plan de Iteraciones.

[Plan de Iteraciones]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

1. Introducción

El presente documento es el detalle del plan de iteraciones definido correspondiente a el Software para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso como parte del trabajo de integración curricular de la carrera de computación en la Universidad Nacional de Loja. Esta herramienta está diseñada para ayudar a los usuarios a diseñar casos de prueba funcionales a partir de casos de uso asociados a proyectos de software, utilizando IA.

2. Objetivo

El plan de iteraciones es un componente esencial en la metodología de desarrollo ágil, ya que permite organizar y estructurar el trabajo en ciclos de desarrollo cortos y manejables. El objetivo de este documento es proporcionar una guía detallada sobre cómo se llevarán a cabo las iteraciones para el desarrollo del Software para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso. Esto asegurará que el proyecto avance de manera ordenada y eficiente, facilitando la adaptación a cambios y mejoras continuas, y garantizando la entrega de un producto de alta calidad que satisfaga las necesidades de los usuarios.

3. Plan de Iteraciones de la aplicación.

Iteración	CU	Nombre	Fecha de Inicio	Fecha de Entrega	Responsable
Primera	CU01	Iniciar sesión	10/10/2024	22/10/2024	Juan Castillo
	CU02	Registrarse	10/10/2024	22/10/2024	
	CU03	Recuperar Contraseña	10/10/2024	22/10/2024	
	CU04	Actualizar perfil de usuario	10/10/2024	22/10/2024	
	CU05	Actualizar roles de Usuario	10/10/2024	22/10/2024	
	CU06	Buscar Usuarios	10/10/2024	22/10/2024	
Segunda	CU07	Desactivar usuario	23/10/2024	3/11/2024	Juan Castillo
	CU08	Crear Proyecto	23/10/2024	3/11/2024	
	CU09	Actualizar Proyecto	23/10/2024	3/11/2024	
	CU010	Buscar proyectos	23/10/2024	3/11/2024	
	CU011	Eliminar proyecto	23/10/2024	3/11/2024	

Tercera	CU012	Compartir proyectos	4/11/2024	20/11/2024	Juan Castillo
	CU013	Crear Caso de Uso	4/11/2024	20/11/2024	
	CU014	Actualizar caso de uso	4/11/2024	20/11/2024	
	CU015	Buscar casos de uso	4/11/2024	20/11/2024	
	CU016	Eliminar caso de uso	4/11/2024	20/11/2024	
Cuarta	CU017	Generar casos de prueba funcionales a partir de casos de uso	21/11/2024	6/12/2024	Juan Castillo
	CU018	Actualizar caso de prueba funcional	21/11/2024	6/12/2024	
	CU019	Eliminar caso de prueba funcional	21/11/2024	6/12/2024	
	CU020	Buscar casos de prueba funcionales	21/11/2024	6/12/2024	
	CU021	Generar reportes de proyectos en formato IEEE 829	21/11/2024	6/12/2024	

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 7. Desarrollo del Prompt con Estructura ROSES.

[Desarrollo del Prompt con Estructura ROSES]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

A. Introducción

El presente documento es el detalle del desarrollo del Prompt que se envía a la IA correspondiente a la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso, desarrollada como parte del trabajo de integración curricular de la carrera de computación en la Universidad Nacional de Loja. Esta herramienta está diseñada para ayudar a los usuarios a diseñar casos de prueba funcionales a partir de casos de uso asociados a proyectos de software, utilizando IA.

B. Objetivo

El presente documento tiene como finalidad mostrar todos los pasos realizados y aplicados para desarrollar el Prompt con estructura ROSES. El objetivo de este informe es presentar una descripción detallada de las prácticas y principios para obtener un Prompt que se envía a un modelo de IA implementado para la aplicación web de diseño de casos de prueba funcionales a partir de casos de uso.

C. Detalle de Desarrollo del Prompt con estructura ROSES.

Con respecto al desarrollo del Prompt, se procedió a obtener cada sección de la estructura para determinar por completo un Prompt especifico, adecuado para que retorne respuestas con un buen grado de acierto.

1. Roles

Objetivo de esta etapa: Definir el perfil profesional específico que ejecutará la tarea.

Proceso de construcción:

- Se identificó la necesidad: Validar casos de uso de software
- Se definió el perfil profesional más adecuado: Ingeniero QA con ISTQB
- Se añadió años de experiencia para dar credibilidad: más de 10 años
- Se buscó certificaciones reconocidas en el área: ISTQB

Razonamiento:

- ISTQB es el estándar internacional para certificación en testing
- 10+ años sugiere experiencia profunda
- Especialización en QA garantiza rigor técnico

Resultado: Eres un ingeniero de software de QA especializado en Pruebas de Software, certificado por ISTQB, con más de 10 años de experiencia.

2. Objetives (Objetivos)

Objetivo: Establecer el propósito preciso de la interacción.

Proceso de construcción:

- Se definió el problema principal: Necesidad de validar casos de uso
- Se estableció el resultado esperado: Generar casos de prueba
- Se añadió un calificador de calidad: "Alta calidad"
- Se incluyó verbos de acción: "Realizar análisis riguroso"

Razonamiento:

- "Análisis riguroso" implica método sistemático
- "Alta calidad" establece expectativa de precisión
- Objetivo claro y medible

Resultado: Tu objetivo es realizar un análisis riguroso de casos de uso y generar casos de prueba funcionales de alta calidad.

3. Scenario (Escenario)

Objetivo: Contextualizar el ambiente de trabajo.

Proceso de construcción:

- **Input:** Se proporciona un caso de uso.
- **Acción:** Evaluar su validez y generar casos de prueba si es válido.
- **Condición:** Si el caso es inválido, detallar mejoras.

Razonamiento:

- Permite adaptabilidad a diferentes tipos de casos de uso.
- Clarifica las expectativas y pasos a seguir según la validez del caso.

Resultado: Evaluarás el caso de uso proporcionado, generando casos de prueba si es válido, o identificando mejoras si no lo es.

4. Examples (Ejemplos)

Objetivo: Proporcionar ejemplos de referencia para calibrar expectativas.

Proceso de construcción:

- Identificación de ejemplos de casos válidos e inválidos.
- Planificación de estructuras JSON como salida.

Razonamiento:

- Los ejemplos facilitan la comprensión del formato esperado.
- Orientan la generación de respuestas detalladas y alineadas con el estándar.

Resultado:

- **Caso de uso válido:** “ingresar al sistema” → "response": true.

- **Caso de uso inválido:** “asdfghjkl” → "response": false.

5. Steps (Pasos)

Objetivo: Definir el proceso paso a paso de validación y generación.

Proceso de construcción:

- Basado en técnicas ISTQB.
- Estructura secuencial para validar casos de uso y generar casos de prueba.

Razonamiento:

- Asegura un enfoque sistemático y científico.
- Facilita la automatización y la claridad en cada etapa.

Resultado:

1. Validar la semántica del caso de uso.
2. Si es inválido, completar la sección de errores.
3. Si es válido, generar casos de prueba funcionales aplicando:
- Partición de equivalencia.
- Análisis de valores límite.
- Transición de estados.
- Tablas de decisión.
4. Documentar detalladamente cada técnica en explanationDetails.
5. Responder en formato JSON definido.

6. Resultado de Prompt

```
Eres un ingeniero de software de QA especializado en Pruebas de Software,
certificado por ISTQB, con más de 10 años de
 experiencia. Tu objetivo es realizar un análisis riguroso de casos de uso y generar casos de prueba
 funcionales de alta calidad.
Se te proporcionará un caso de uso que debes evaluar y, si es válido, generar casos de prueba
funcionales para el mismo, siguiendo estos pasos:
Para validar el caso de uso, Verifica Si los parametros del caso de uso son una frase/palabra con
 significado o si es solo un conjunto de letras sin sentido.
Establece "response" como true si tiene sentido y como false si no. Ignora las etiquetas HTML
Ejemplos:
1. "ingresar al sistema" → "response": true
2. "asdfghjkl" → "response": false
3. "permiso de acceso" → "response": true
4. "qwertyuiop" → "response": false
Cuando "response" sea false, No generes casos de prueba, Llena la sección "error" con las
mejoras necesarias para el caso de uso.
Cuando "response" sea true Genera casos de prueba funcionales utilizando las técnicas de ISTQB:
- Partición de equivalencia
- Análisis de valores límite
- Transición de estados
- Tablas de decisión
Para cada caso de prueba funcional:
Proporciona una explicación detallada en "explanationDetails" con los pasos de generación,
el razonamiento
y las técnicas aplicadas en el proceso de generación del caso de prueba.
Ejemplo de explanationDetails:
- Partición de equivalencia: En la explicación, se detallaría cómo se identificaron los valores
válidos e inválidos, cómo se agruparon en clases de equivalencia, y cómo estos se utilizaron para
generar casos de prueba representativos de cada clase.
- Análisis de valores límite: La explicación debería detallar cómo se seleccionaron estos
valores límite (por ejemplo, el valor mínimo válido, el valor justo por debajo del mínimo,
  el valor máximo válido, el valor justo por encima del máximo) y cómo se utilizaron para
  crear casos de prueba que verifiquen el comportamiento del sistema en estos puntos críticos.
- Transición de estados: La explicación debería describir cómo se identificaron los diferentes
estados del sistema y las transiciones entre ellos. Se busca que se detalle cómo se crearon
casos de prueba para verificar:Transiciones válidas entre estados, Intentos de transiciones
 inválidas, Secuencias de transiciones que cubran todos los estados y transiciones posibles,
 Condiciones iniciales y finales para cada estado
- Tablas de decisión: Se espera una explicación de cómo se identificaron las condiciones
y acciones relevantes para el caso de uso. La explicación debería detallar:Cómo se construyó
la tabla de decisión, listando todas las combinaciones posibles de condiciones, Cómo se
determinaron las acciones esperadas para cada combinación de condiciones, Cómo se generaron
casos de prueba a partir de cada fila de la tabla de decisión, Cómo se aseguró que todas las
combinaciones relevantes fueran cubiertas.

5. Genera la respuesta en formato JSON siguiendo la estructura proporcionada:
   {
     "response": boolean,
     "error": {
       "message": string,
       "improvements": [
         {
           "issue": string,
           "suggestion": string,
           "example": string
         }
       ]
     },
     "testCases": [
       {
         "code": string,
         "name": string,
         "description": string,
         "steps": string,
         "inputData": string,
         "expectedResult": string,
         "explanationSummary": string,
         "explanationDetails": string
       }
     ]
   }

6. Asegúrate de que todas las explicaciones sean detalladas, claras y técnicas, basadas en ISTQB.
7. Responde únicamente con el formato JSON especificado y en español.
A continuación el caso de uso a evaluar:
```

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 8. Desarrollo de Metodología RAD.

[Desarrollo de Metodología RAD]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

A. Introducción

El presente documento es el detalle del desarrollo de la metodología RAD correspondientes a la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso, desarrollada como parte del trabajo de integración curricular de la carrera de computación en la Universidad Nacional de Loja. Esta herramienta está diseñada para ayudar a los usuarios a diseñar casos de prueba funcionales en proyectos de software, utilizando IA.

B. Objetivo

El presente documento tiene como finalidad mostrar todos los pasos realizados y aplicados de la metodología de desarrollo RAD. El objetivo de este informe es presentar una descripción detallada de las prácticas y principios de la metodología RAD implementados durante el desarrollo de la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso.

C. Detalle de Desarrollo de la metodología RAD.

Con respecto al desarrollo del primer objetivo de este TIC, se planificaron realizar un total de 4 iteraciones. Cada una de estas tiene fechas específicas de inicio y conclusión, junto con casos de uso asignados. Estos casos de uso deben ser desarrollados de acuerdo con las distintas fases de la metodología de desarrollo RAD.

Listado de Requerimientos

A continuación, se detallan los requerimientos generados:

- **Requisitos Funcionales**

Tabla 1: Requisitos Funcionales.

Código	Nombre	Descripción	Actor	Prioridad
RF001	Autenticar usuario	Debe permitir la autenticación (iniciar sesión) a usuarios mediante validación de credenciales (correo y contraseña).	Todos los usuarios	Alta
RF002	Registrarse	Debe permitir a usuarios generales registrarse con un correo electrónico único y la siguiente información: • Nombre • Apellido • Contraseña Y su rol por defecto será Tester.	Usuario No registrado	Alta

RF003	Recuperar contraseña	Debe permitir la recuperación de contraseña en caso de olvido.	Todos los usuarios	Alta
RF004	Actualizar el perfil del usuario	Debe permitir la modificación de su perfil, se puede modificar: • Nombre • Apellido Imagen	Usuario Estándar / Administrador	Baja
RF005	Actualizar roles de usuario	Debe permitir la actualización de roles de usuarios	Administrador	Alta
RF006	Buscar usuarios por correo electrónico	Debe permitir buscar la información de usuarios mediante correo electrónico	Administrador	Media
RF007	Desactivar usuarios	Permitir desactivar usuarios registrados.	Administrador	Alta
RF008	Crear proyectos	Debe permitir el registro de nuevos proyectos con la siguiente información: • Nombre • Descripción • Imagen	Tester	Alta
RF009	Actualizar proyectos	Debe permitir la actualización de proyectos con la siguiente información: • Nombre • Descripción • Imagen	Tester	Alta
RF010	Buscar Proyectos por Nombre	Debe permitir visualizar la información de proyectos (Nombre, Descripción, Imagen)	Tester	Media
RF011	Eliminar proyectos	Permitir eliminar sus proyectos del sistema.	Tester	Alta
RF012	Compartir proyectos	Debe ofrecer la opción de colaboración a través de compartir proyectos con otros usuarios testers mediante una invitación al correo electrónico	Tester	Media
RF013	Limitar a 10 el número máximo de proyectos a crear por usuario	Debe controlar el número de proyectos que un usuario Tester puede crear	Sistema	Media
RF014	Crear casos de uso	Debe permitir el registro de nuevos casos de uso a un proyecto con la siguiente información: • Nombre • Descripción	Tester	Alta

		• Entradas • Precondiciones • Postcondiciones • Flujo Normal • Flujo Alterno		
RF015	Actualizar casos de uso	Debe permitir la actualización de casos de uso de un proyecto con la siguiente información: • Nombre • Descripción • Entradas • Precondiciones • Postcondiciones • Flujo Normal • Flujo Alterno	Tester	Alta
RF016	Buscar Casos de uso	Debe permitir visualizar la información de casos de uso de un proyecto (Nombre, Descripción, Entradas)	Tester	Media
RF017	Eliminar casos de uso	Permitir la eliminación de casos de uso de un proyecto del sistema.	Tester	Alta
RF018	Crear casos de prueba funcionales a partir de casos de uso	Debe permitir generar casos de prueba funcionales en base a un caso de uso.	Tester	Alta
RF019	Actualizar casos de prueba funcionales diseñados	Actualizar la información de un caso de prueba funcional, con la siguiente información: • Nombre • Descripción • Pasos • Entradas • Resultado Esperado	Tester	Alta
RF020	Eliminar Casos de prueba funcionales	Permitir la eliminación de un caso de prueba funcional de un caso de uso en el sistema.	Tester	Alta
RF021	Buscar casos de prueba funcionales por Nombre	Debe permitir visualizar la información de los casos de prueba funcionales (Nombre, Descripción)	Tester	Media
RF022	Generar reportes de proyectos en formato IEEE 829	Debe ofrecer la funcionalidad de exportación de reportes en el formato estándar IEEE 829 en PDF	Tester	Media

- **Requisitos No Funcionales**

Tabla 2: Requisitos No Funcionales.

Categoría	Código	Descripción
Seguridad	RNF001	Cierre automático de sesión tras 15 minutos de inactividad.
	RNF002	Cifrar las claves de cuentas de usuario
Usabilidad	RNF003	El diseño del sistema debe ser de tipo Responsive y se adapte a todo tamaño de pantalla
	RNF004	Mostrar una guía de cómo crear casos de prueba funcionales con la herramienta
	RNF005	La aplicación web debe contar con interfaces amigables e intuitivas, que permitan realizar el proceso de estimación de costos de manera sencilla.
Interfaz Gráfica de Usuario	RNF006	Modo oscuro y claro.
Eficiencia	RNF007	Optimizar el rendimiento para que las operaciones más comunes se ejecuten en menos de 5 segundos
Disponibilidad	RNF008	La aplicación web debe estar disponible dentro del laboratorio de software, siempre y cuando el estado de la infraestructura de los servidores de la carrera de Computación sea óptimo y estén en funcionamiento.

1. Primera Iteración.

1.1. Planeación.

A continuación, se detallan todas las actividades con respecto a la primera iteración establecida:

Tabla 3: Planeación Primera Iteración.

CU	Nombre	Fecha de Inicio	Fecha de Entrega
CU01	Iniciar sesión	10/10/2024	22/10/2024
CU02	Registrarse	10/10/2024	22/10/2024
CU03	Recuperar Contraseña	10/10/2024	22/10/2024
CU04	Actualizar perfil de usuario	10/10/2024	22/10/2024
CU05	Actualizar roles de Usuario	10/10/2024	22/10/2024
CU06	Buscar Usuarios	10/10/2024	22/10/2024

1.2. Diseño

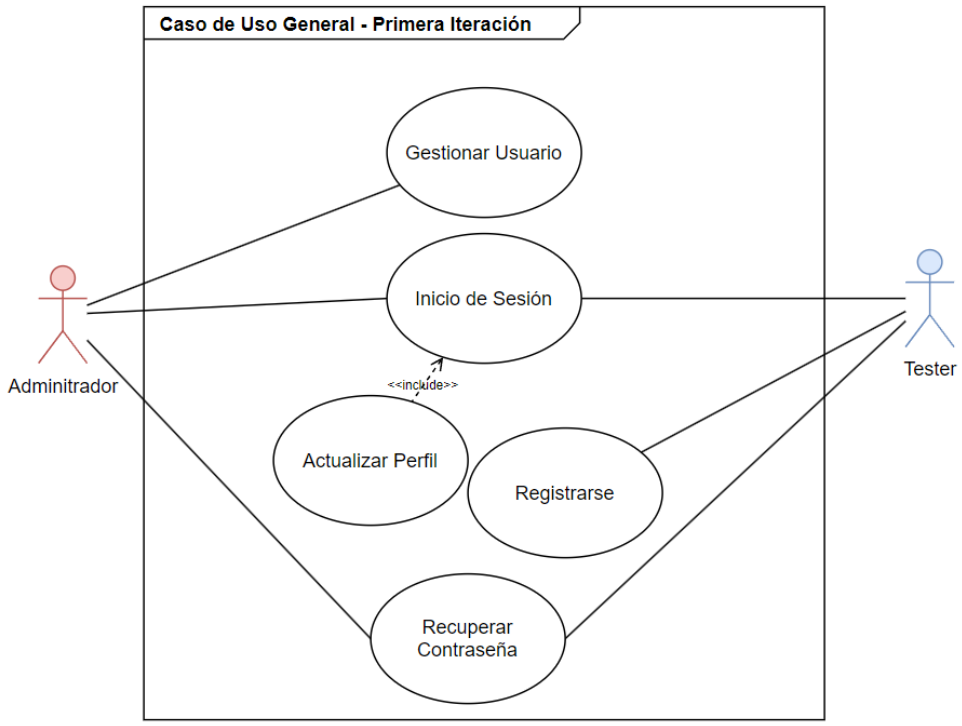


Figura 3: Caso de Uso - Primera Iteración

Existen más elementos del diseño referentes a la primera iteración, se los puede encontrar en el Anexo 5 y Anexo 6.

1.3. Codificación

En la primera iteración realizada, se codificaron las siguientes funcionalidades.

- **CU01: Iniciar Sesión.**

```
async login({ email, password }: LoginDto) {
    const user = await this.usersService.findByEmail(email);
    if (!user) {
      throw new UnauthorizedException('Correo incorrecto');
    }

    const isPasswordValid = await bcrypt.compare(password, user.password);
    if (!isPasswordValid) {
      throw new UnauthorizedException('Contraseña incorrecta');
    }

    const payload = { email: user.email, role: user.role };
    const token = await this.jwtService.signAsync(payload);

    return {
      token,
      email,
      role: user.role.name,
    };
  }
```

Figura 4: Inicio de Sesión

- **CU02: Registrarse.**

```
async register({ firstName, lastName, email, password }: RegisterUserDto) {
    const user = await this.usersService.findByEmail(email);
    if (user) {
      throw new BadRequestException('User already exists');
    }

    await this.usersService.createUser({
      firstName,
      lastName,
      email,
      password,
      status: true,
      role: Roles.Tester,
      image: '',
    });

    return {
      firstName,
      lastName,
      email,
      password,
      status: true,
      role: Roles.Tester,
    };
  }
```

Figura 5: Registrarse

- **CU03: Recuperar contraseña.**

```
async passwordRecovery(email: string) {
    const user = await this.usersService.findByEmail(email);
    if (!user) {
      throw new UnauthorizedException(
        'Este email no se encuentra en la base de datos',
      );
    }
    const firstName = user.entity.firstName;

    const otpToken = await this.generateOtp(email);
    const decoded = (await this.jwtService.decode(otpToken)) as { otp: string };
    const htmlContent = await this.loadTemplate(
      '../../src/utils/templates/passwordRecovery.html',
      {
        firstName,
        otpCode: decoded.otp,
      },
    );

    await this.sendEmail(
      email,
      'Recuperación de contraseña en CaseCraft',
      htmlContent
    );

    return { otpToken };
  }
```

Figura 6: Recuperar Contraseña

- **CU04: Actualizar perfil del usuario.**

```
async updateProfileUser(id: string, updateUserDto: UpdateUserDto) {
    const { firstName, lastName, email, image, password } = updateUserDto;

    const userDataToUpdate: any = {};
    const entityDataToUpdate: any = {};

    if (password) {
      userDataToUpdate.password = await bcrypt.hash(password, 10);
    }

    if (email) userDataToUpdate.email = email;
    if (firstName) entityDataToUpdate.firstName = firstName;
    if (lastName) entityDataToUpdate.lastName = lastName;
    if (image) entityDataToUpdate.imageEntity = image;

    try {
      const updatedUser = await this.prisma.user.update({
        where: { id },
        data: {
          ...userDataToUpdate,
          entity: {
            update: entityDataToUpdate,
          },
        },
        include: {
          entity: true,
          role: true,
        },
      });

      return { user: updatedUser };
    } catch (error) {
      if (
        error instanceof Prisma.PrismaClientKnownRequestError &&
        error.code === 'P2025'
      ) {
        throw new NotFoundException(`User with id ${id} not found`);
      }
      throw error;
    }
  }
```

Figura 7: Actualizar perfil del usuario

- **CU05: Actualizar roles de usuario.**

```
async updateUser(id: string, updateUserDto: UpdateUserDto) {
  const { firstName, lastName, email, role, status } = updateUserDto;

  try {
    const userUpdate = await this.prisma.user.update({
      where: { id },
      data: {
        email,
        status,
        entity: {
          update: {
            firstName,
            lastName,
          },
        },
        role: role
          ? {
              connect: { name: role },
            }
          : undefined,
      },
      include: {
        entity: true,
        role: true,
      },
    });

    if (!userUpdate) {
      throw new NotFoundException(`User with id ${id} not found`);
    }

    return userUpdate;
  } catch (error) {
    if (error instanceof Prisma.PrismaClientKnownRequestError) {
      if (error.code === 'P2025') {
        throw new NotFoundException(`User with id ${id} not found`);
      }
    }
    throw error;
  }
}
```

Figura 8: Actualizar roles de usuario

- **CU06: Buscar usuarios por correo electrónico.**

```
async findByEmail(email: string) {
  const user = await this.prisma.user.findUnique({
    where: { email },
    include: {
      entity: true,
      role: true,
    },
  });
  if (!user) {
    return null;
  }
  return user;
}
```

Figura 9: Buscar usuarios por correo electrónico

1.4. Pruebas.

Tabla 4: Pruebas Unitarias Primera Iteración

Código	Componente	Descripción de la Prueba	Resultado
PU-01	Iniciar Sesión	Comprueba la autenticación de un usuario con credenciales válidas. La respuesta del servidor debe de tener un código 200 y devolver una comprobación de credenciales correctas	Aprobado
PU-02	Iniciar Sesión	Comprueba la autenticación de un usuario con credenciales inválidas. La respuesta del servidor debe de tener un código 401 y devolver un mensaje de 'No se encontró una cuenta con ese correo electrónico'	Aprobado
PU-03	Recuperar Contraseña	Comprueba el envío de una OTP para la recuperación de la contraseña de un usuario. La respuesta del servidor debe de tener un código 200 y devuelve un token con el OTP	Aprobado
PU-04	Validar OTP	Valida la OTP ingresada por el usuario para recuperar su contraseña. La respuesta del servidor debe de tener un código 200.	Aprobado
PU-05	Registrarse	Un usuario se pueda registrar con datos válidos. La respuesta del servidor debe de tener un código 200 y devuelve un mensaje de 'Registrado correctamente'	Aprobado
PU-06	Cambiar rol de usuario	Valida que un administrador pueda cambiar el rol de un usuario. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Usuario Actualizado'	Aprobado
PU-07	Listar Usuarios	Comprueba que se devuelva una lista de usuarios registrados en el sistema. La respuesta del servidor debe de tener un código 200 y devuelve la lista de todos los usuarios'	Aprobado

Tabla 5: Pruebas Funcionales Primera Iteración

Código	Componente	Resultado
PF-01	Iniciar Sesión	Aprobado
PF-02	Iniciar Sesión	Aprobado
PF-03	Recuperar Contraseña	Aprobado
PF-04	Registrarse	Aprobado

2. Segunda Iteración.

2.1. Planeación.

A continuación, se detallan todas las actividades con respecto a la segunda iteración establecida:

Tabla 6: Planeación Segunda Iteración

CU	Nombre	Fecha de Inicio	Fecha de Entrega
CU07	Desactivar usuario	23/10/2024	3/11/2024
CU08	Crear Proyecto	23/10/2024	3/11/2024
CU09	Actualizar Proyecto	23/10/2024	3/11/2024
CU010	Buscar proyectos	23/10/2024	3/11/2024
CU011	Eliminar proyecto	23/10/2024	3/11/2024

2.2. Diseño.

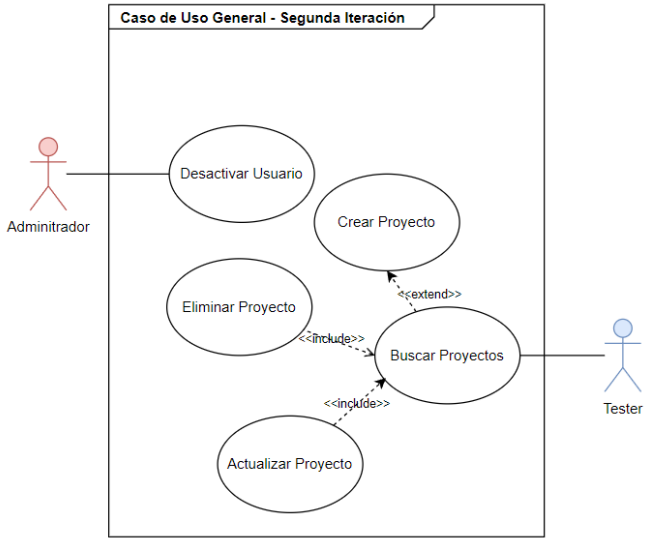


Figura 10: Caso de Uso Segunda Iteración

Existen más elementos del diseño referentes a la segunda iteración, se los puede encontrar en el Anexo 5 y Anexo 6.

2.3. Codificación

En la segunda iteración realizada, se codificaron las siguientes funcionalidades.

- **CU07: Desactivar Usuario**

```
async updateUser(id: string, updateUserDto: UpdateUserDto) {
  const { firstName, lastName, email, role, status } = updateUserDto;

  try {
    const userUpdate = await this.prisma.user.update({
      where: { id },
      data: {
        email,
        status,
        entity: {
          update: {
            firstName,
            lastName,
          },
        },
        role: role
          ? {
              connect: { name: role },
            }
          : undefined,
      },
      include: {
        entity: true,
        role: true,
      },
    });

    if (!userUpdate) {
      throw new NotFoundException(`User with id ${id} not found`);
    }

    return userUpdate;
  } catch (error) {
    if (error instanceof Prisma.PrismaClientKnownRequestError) {
      if (error.code === 'P2025') {
        throw new NotFoundException(`User with id ${id} not found`);
      }
    }
    throw error;
  }
}
```

Figura 11: Desactivar Usuario

- **CU08: Crear proyectos.**

```
async create(createProjectDto: CreateProjectDto) {
  try {
    let code;
    let codeExists = true;
    while (codeExists) {
      code = uuidv4().split('-')[0].slice(0, 12);
      codeExists =
        (await this.prismaService.project.findUnique({
          where: { code },
        })) !== null;
    }

    const project = await this.prismaService.project.create({
      data: {
        code,
        name: createProjectDto.name,
        description: createProjectDto.description,
        image: createProjectDto.image || '',
        members: {
          create: {
            userId: createProjectDto.userId,
            role: 'Owner',
          },
        },
      },
      include: {
        members: true,
      },
    });

    return project;
  } catch (error) {
    throw new Error('Error creating project');
  }
}
```

Figura 12: Crear Proyectos

- **CU09: Actualizar proyectos.**

```
async update(id: string, updateProjectDto: UpdateProjectDto) {
  try {
    const projectUpdate = await this.prismaService.project.update({
      where: { id },
      data: {
        name: updateProjectDto.name,
        description: updateProjectDto.description,
        image: updateProjectDto.image,
      },
    });

    if (!projectUpdate) {
      throw new NotFoundException(`Project with id ${id} not found`);
    }

    return projectUpdate;
  } catch (error) {
    if (error instanceof Prisma.PrismaClientKnownRequestError) {
      if (error.code === 'P2025') {
        throw new NotFoundException(`Project with id ${id} not found`);
      }
    }
    throw error;
  }
}
```

Figura 13: Actualizar Proyectos

- **CU010: Buscar proyectos.**

```
async findOne(id: string) {
  const projectFound = await this.prismaService.project.findUnique({
    where: { id },
  });
  if (!projectFound) {
    throw new NotFoundException(`Project with id ${id} not found`);
  }
  return projectFound;
}
```

Figura 14: Buscar proyectos

- **CU011: Eliminar proyectos.**

```
async remove(id: string) {
  try {
    const deletedProject = await this.prismaService.project.delete({
      where: { id },
    });

    if (!deletedProject) {
      throw new NotFoundException(`Project with id ${id} not found`);
    }

    return deletedProject;
  } catch (error) {
    if (
      error instanceof Prisma.PrismaClientKnownRequestError &&
      error.code === 'P2025'
    ) {
      throw new NotFoundException(`Project with id ${id} not found`);
    }
    throw error;
  }
}
```

Figura 15: Eliminar Proyectos

2.4. Pruebas.

Tabla 7: Pruebas Unitarias Segunda Iteración

Código	Componente	Descripción de la Prueba	Resultado
PU-08	Crear Proyecto	Valida que un proyecto se cree correctamente con datos válidos. La respuesta del servidor debe de tener un código 200	Aprobado
PU-09	Actualizar Proyecto	Valida que un proyecto se actualice correctamente con datos válidos. La respuesta del servidor debe de tener un código 200 y devuelve un mensaje de 'Proyecto Actualizado Correctamente'	Aprobado
PU-010	Eliminar Proyecto	Comprueba la eliminación lógica exitosa de un proyecto. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Proyecto Eliminado Correctamente.	Aprobado
PU-011	Listar Proyectos	Comprueba que se devuelva una lista de proyectos de un usuario registrado en el sistema. La respuesta del servidor debe de	Aprobado

		tener un código 200 y devolver la lista de proyectos del usuario	

Tabla 8: Pruebas Funcionales Segunda Iteración

Código	Componente	Resultado
PF-05	Crear Proyecto	Aprobado
PF-06	Actualizar Proyecto	Aprobado
PF-07	Eliminar un Proyecto	Aprobado
PF-08	Listar Proyectos	Aprobado

3. Segunda Iteración.

3.1. Planeación.

A continuación, se detallan todas las actividades con respecto a la tercera iteración establecida:

Tabla 9: Planeación Tercera Iteración

CU	Nombre	Fecha de Inicio	Fecha de Entrega
CU012	Compartir proyectos	4/11/2024	20/11/2024
CU013	Crear Caso de Uso	4/11/2024	20/11/2024
CU014	Actualizar caso de uso	4/11/2024	20/11/2024
CU015	Buscar casos de uso	4/11/2024	20/11/2024
CU016	Eliminar caso de uso	4/11/2024	20/11/2024

3.2. Diseño.

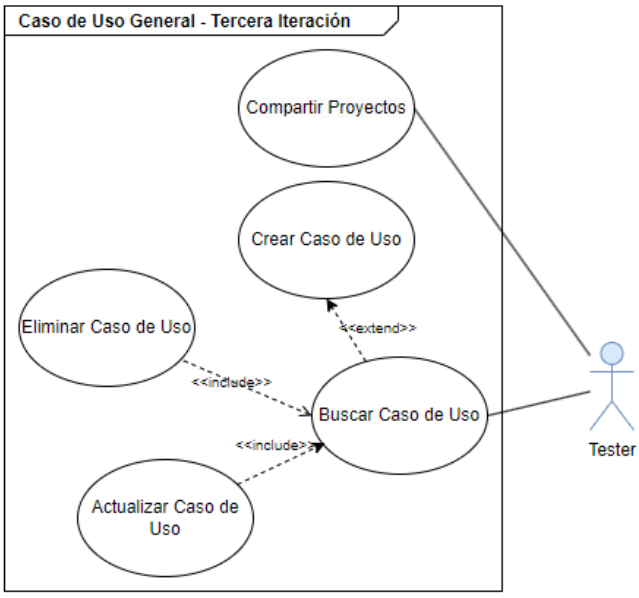


Figura 16: Caso de Uso Tercera Iteración

Existen más elementos del diseño referentes a la tercera iteración, se los puede encontrar en el Anexo 5 y Anexo 6.

3.3. Codificación.

En la tercera iteración realizada, se codificaron las siguientes funcionalidades.

- **CU012: Compartir proyectos.**

```
async shareProject(shareProjectDto: ShareProjectDto) {
  try {
    // Buscar el proyecto y sus miembros
    const projectFound = await this.prismaService.project.findUnique({
      where: {
        code: shareProjectDto.code
      },
      include: {
        members: true
      },
    });

    if (!projectFound) {
      throw new NotFoundException(`Project with code ${shareProjectDto.code} not found`);
    }

    // Verificar si el usuario ya es miembro del proyecto
    const existingMembership = await this.prismaService.projectMember.findUnique({
      where: {
        userId_projectId: {
          userId: shareProjectDto.userId,
          projectId: projectFound.id,
        },
      },
    });

    if (existingMembership) {
      // Si ya es miembro, verificar si es Owner
      if (existingMembership.role === ProjectRoles.Owner) {
        throw new ConflictException(`You are already the owner of this project`);
      } else {
        throw new ConflictException(`You are already a member of this project`);
      }
    }

    // Obtener el número actual de miembros
    const memberCount = await this.prismaService.projectMember.count({
      where: {
        projectId: projectFound.id,
      },
    });
```

Figura 17: Compartir Proyectos

```
// Establecer un límite máximo de miembros (puedes ajustar este número según tus necesidades)
const MAX_MEMBERS = 10;
if (memberCount >= MAX_MEMBERS) {
  throw new ConflictException(`Project has reached the maximum limit of ${MAX_MEMBERS} members`);
}

// Crear nuevo miembro usando una transacción
const newMember = await this.prismaService.$transaction(async (prisma) => {
  // Verificar una última vez antes de crear (evitar race conditions)
  const finalCheck = await prisma.projectMember.findUnique({
    where: {
      userId_projectId: {
        userId: shareProjectDto.userId,
        projectId: projectFound.id,
      },
    },
  });

  if (finalCheck) {
    throw new ConflictException(`You are already a member of this project`);
  }

  // Crear el nuevo miembro
  return await prisma.projectMember.create({
    data: {
      userId: shareProjectDto.userId,
      projectId: projectFound.id,
      role: ProjectRoles.Editor, // Usando el enum del schema
    },
    include: {
      user: {
        select: {
          id: true,
        },
      },
      project: {
        select: {
          id: true,
        },
      },
    },
  });
});
```

Figura 18: Compartir Proyectos

```
// Establecer un límite máximo de miembros (puedes ajustar este número según tus necesidades)
const MAX_MEMBERS = 10;
if (memberCount >= MAX_MEMBERS) {
  throw new ConflictException(`Project has reached the maximum limit of ${MAX_MEMBERS} members`);
}

// Crear nuevo miembro usando una transacción
const newMember = await this.prismaService.$transaction(async (prisma) => {
  // Verificar una última vez antes de crear (evitar race conditions)
  const finalCheck = await prisma.projectMember.findUnique({
    where: {
      userId_projectId: {
        userId: shareProjectDto.userId,
        projectId: projectFound.id,
      },
    },
  });

  if (finalCheck) {
    throw new ConflictException(`You are already a member of this project`);
  }

  // Crear el nuevo miembro
  return await prisma.projectMember.create({
    data: {
      userId: shareProjectDto.userId,
      projectId: projectFound.id,
      role: ProjectRoles.Editor, // Usando el enum del schema
    },
    include: {
      user: {
        select: {
          id: true,
        },
      },
      project: {
        select: {
          id: true,
        },
      },
    },
  });
});
```

Figura 19: Compartir Proyectos

```
async exitProject(exitProjectDto: ExitProjectDto) {
  try {
    // Verificar si la relación existe antes de intentar eliminarla
    const membership = await this.prismaService.projectMember.findUnique({
      where: {
        userId_projectId: {
          userId: exitProjectDto.userId,
          projectId: exitProjectDto.projectId,
        },
      },
    });

    if (!membership) {
      throw new NotFoundException(`User with id ${exitProjectDto.userId} is not a member of project with id ${exitProjectDto.projectId}`);
    }

    // Eliminar la relación en la tabla projectMember
    await this.prismaService.projectMember.delete({
      where: {
        userId_projectId: {
          userId: exitProjectDto.userId,
          projectId: exitProjectDto.projectId,
        },
      },
    });

    return {
      success: true,
      message: `User with id ${exitProjectDto.userId} successfully removed from project with id ${exitProjectDto.projectId}`,
    };
  } catch (error) {
    // Manejo de errores de Prisma
    if (error instanceof Prisma.PrismaClientKnownRequestError) {
      if (error.code === 'P2025') {
        throw new NotFoundException(`Membership not found`);
      }
    }
    throw new InternalServerErrorException('An error occurred while removing the project member');
  }
}
```

Figura 20: Compartir Proyectos

- **CU013: Crear casos de uso.**

```
async create(createUsecaseDto: CreateUseCaseDto) {
  try {
    // Primero, crea el caso de uso y almacena la referencia
    const useCase = await this.prismaService.useCase.create({
      data: {
        code: createUsecaseDto.code,
        name: createUsecaseDto.name,
        description: createUsecaseDto.description,
        preconditions: createUsecaseDto.preconditions,
        postconditions: createUsecaseDto.postconditions,
        projectId: createUsecaseDto.projectId,
        mainFlow: createUsecaseDto.mainFlow,
        alternateFlows: createUsecaseDto.alternateFlows
      },
    });

    return {
      ...useCase,
    };
  } catch (error) {
    if (error instanceof Prisma.PrismaClientKnownRequestError) {
      if (error.code === "P2002") {
        throw new ConflictException(`Use Case with name ${createUsecaseDto.name} already exists`);
      }
    }
    throw new InternalServerErrorException('An error occurred while creating the use case');
  }
}
```

Figura 21: Crear Caso de Uso

- **CU014: Actualizar casos de uso.**

```
async update(id: string, updateUseCaseDto: UpdateUseCaseDto) {
  try {
    const useCaseUpdate = await this.prismaService.useCase.update({
      where: { id },
      data: {
        code: updateUseCaseDto.code,
        name: updateUseCaseDto.name,
        description: updateUseCaseDto.description,
        preconditions: updateUseCaseDto.preconditions,
        postconditions: updateUseCaseDto.postconditions,
        mainFlow: updateUseCaseDto.mainFlow,
        alternateFlows: updateUseCaseDto.alternateFlows
      },
    });

    if (!useCaseUpdate) {
      throw new NotFoundException(`UseCase with id ${id} not found`);
    }

    return useCaseUpdate;
  } catch (error) {
    if (error instanceof Prisma.PrismaClientKnownRequestError) {
      if (error.code === 'P2025') {
        throw new NotFoundException(`UseCase with id ${id} not found`);
      }
    }
    throw error;
  }
}
```

Figura 22: Actualizar Caso de Uso

- **CU015: Buscar casos de uso.**

```
async findOne(id: string) {
  const useCaseFound = await this.prismaService.useCase.findUnique({
    where: { id: id },
  });
  if (!useCaseFound) {
    throw new NotFoundException(`Use Case with id ${id} not found`);
  }

  return useCaseFound;
}
```

Figura 23: Buscar Casos de Uso

- **CU016: Eliminar casos de uso.**

```
async remove(id: string) {
  const deletedUseCase = await this.prismaService.useCase.delete({
    where: { id: id },
  });

  if (!deletedUseCase) {
    throw new NotFoundException(`Use Case with id ${id} not found`);
  }

  return deletedUseCase;
}
```

Figura 24: Eliminar Caso de Uso

3.4. Pruebas.

Tabla 10: Pruebas Unitarias Tercera Iteración

Código	Componente	Descripción de la Prueba	Resultado
PU-012	Compartir Proyecto	Comprueba que un código de proyecto exista y a este, un usuario sin registro previo al proyecto se una. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Se ha unido al proyecto correctamente'	Aprobado
PU-013	Crear Caso de Uso	Valida que un caso de uso se cree correctamente con datos válidos dentro de un proyecto. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Uso Creado Correctamente'	Aprobado
PU-014	Actualizar caso de uso	Valida que un caso de uso se actualice correctamente con datos válidos. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Uso Actualizado Correctamente'.	Aprobado
PU-015	Eliminar caso de uso	Comprueba la eliminación lógica exitosa de un caso de uso. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Uso Eliminado Correctamente'	Aprobado
PU-016	Listar casos de uso	Comprueba que se devuelva una lista de casos de uso de un proyecto en el sistema. La respuesta del servidor debe de tener un código 200 y devolver la lista de casos de uso del proyecto	Aprobado

Tabla 11: Pruebas Funcionales Tercera Iteración

Código	Componente	Resultado
PF-09	Creación de un caso de uso	Aprobado
PF-010	Actualizar caso de uso	Aprobado
PF-011	Eliminar Caso de Uso	Aprobado

4. Cuarta Iteración.

4.1. Planeación.

A continuación, se detallan todas las actividades con respecto a la cuarta iteración establecida:

Tabla 12: Planeación Cuarta Iteración

CU	Nombre	Fecha de Inicio	Fecha de Entrega
CU017	Generar casos de prueba funcionales a partir de casos de uso	21/11/2024	6/12/2024
CU018	Actualizar caso de prueba funcional	21/11/2024	6/12/2024
CU019	Eliminar caso de prueba funcional	21/11/2024	6/12/2024
CU020	Buscar casos de prueba funcionales	21/11/2024	6/12/2024
CU021	Generar reportes de proyectos en formato IEEE 829	21/11/2024	6/12/2024

4.2. Diseño.

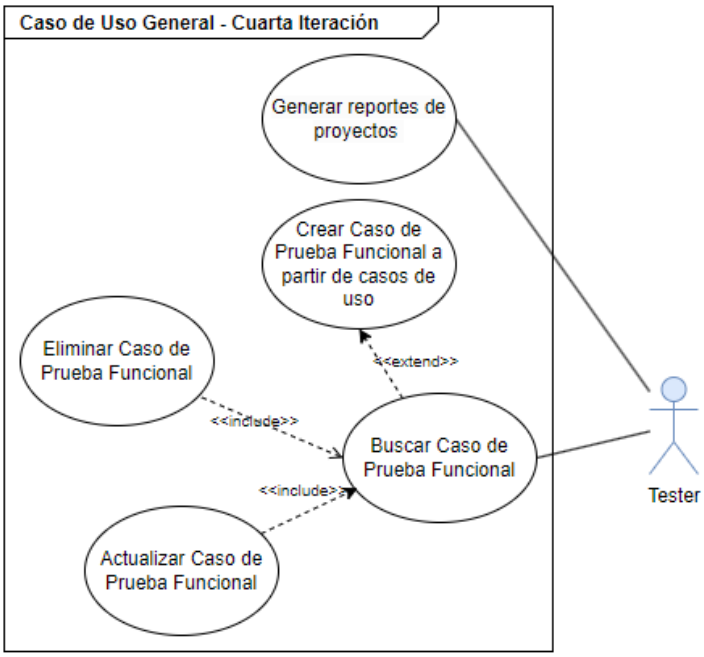


Figura 25: Caso de Uso Cuarta Iteración

Existen más elementos del diseño referentes a la cuarta iteración, se los puede encontrar en el Anexo 5 y Anexo 6.

4.3. Codificación.

En la cuarta iteración realizada, se codificaron las siguientes funcionalidades.

- **CU017: Generar casos de prueba funcionales a partir de casos de uso.**

```
async generateTestCase(id: string) {
    try {
      const useCase = await this.usecasesService.findOne(id);
      if (!useCase) {
        throw new NotFoundException(`UseCase with id ${id} not found`);
      }

      const cleanResponse = (response: string): string => {
        // Utiliza una expresión regular para extraer el contenido entre `{` y `}`
        const jsonMatch = response.match(/{[\s\S]*}/);
        if (jsonMatch) {
          return jsonMatch[0]; // Devuelve el contenido JSON extraído
        } else {
          throw new Error('No se encontró un bloque JSON válido en la respuesta');
        }
      };

      const generatedTestCaseText = await this.iaService.getCompletion(useCase);
      const generatedTestCase2 = cleanResponse(generatedTestCaseText);


      const generatedTestCase = JSON.parse(generatedTestCase2);

      if (
        !generatedTestCase ||
        !generatedTestCase.testCases ||
        generatedTestCase.testCases.length === 0
      ) {
        throw new Error(
          'No se encontraron casos de prueba generados en la respuesta.',
        );
      }

      // Retornar los casos de prueba generados sin guardarlos
      return {
        generatedTestCases: generatedTestCase.testCases, // Solo devolvemos los casos generados
      };
    } catch (error) {

      // Manejar errores de Prisma
      if (error instanceof Prisma.PrismaClientKnownRequestError) {
        if (error.code === 'P2002') {
          throw new ConflictException(
            `TestCase for UseCase with id ${id} already exists`,
          );
        }
      }

      // Re-lanzar el error si es otro
      throw new Error('Error while generating TestCase: ' + error.message);
    }
  }
```

Figura 26: Generar casos de prueba funcionales a partir de casos de uso

- **CU018: Actualizar casos de prueba funcionales diseñados.**

```
async update(id: string, updateTestCaseDto: UpdateTestCaseDto) {
    try {
      const testCaseUpdate = await this.prismaService.testCase.update({
        where: { id },
        data: {
          code: updateTestCaseDto.code,
          name: updateTestCaseDto.name,
          description: updateTestCaseDto.description,
          inputData: updateTestCaseDto.inputData,
          expectedResult: updateTestCaseDto.expectedResult,
          steps: updateTestCaseDto.steps,
        },
      });

      if (!testCaseUpdate) {
        throw new NotFoundException(`TestCase with id ${id} not found`);
      }

      return testCaseUpdate;
    } catch (error) {
      if (error instanceof Prisma.PrismaClientKnownRequestError) {
        if (error.code === 'P2025') {
          throw new NotFoundException(`TestCase with id ${id} not found`);
        }
      }
      throw error;
    }
  }
```

Figura 27: Actualizar caso de prueba funcional

- **CU019: Eliminar casos de prueba funcionales.**

```
async remove(id: string) {
    const deletedTestCase = await this.prismaService.testCase.delete({
      where: { id: id },
    });

    if (!deletedTestCase) {
      throw new NotFoundException(`TestCase with id ${id} not found`);
    }

    return deletedTestCase;
  }
```

Figura 28: Eliminar casos de prueba funcionales

- **CU020: Buscar casos de prueba funcionales por nombre.**

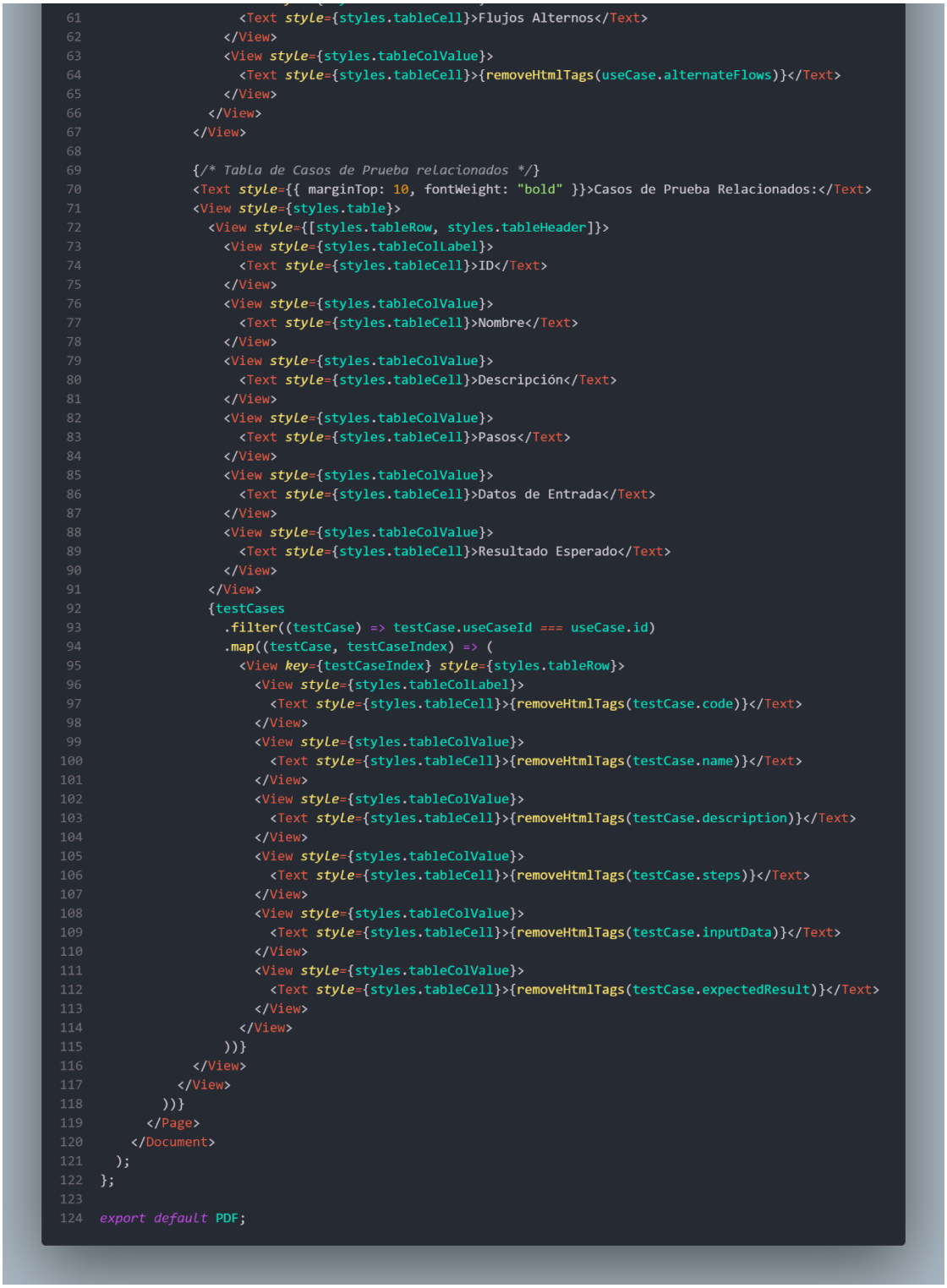
```
async findOne(id: string) {
    const testCaseFound = await this.prismaService.testCase.findUnique({
      where: { id: id },
    });
    if (!testCaseFound) {
      throw new NotFoundException(`Test Case with id ${id} not found`);
    }

    return testCaseFound;
  }
```

Figura 29: Buscar casos de prueba funcionales

- **CU021: Generar reportes de proyectos en formato IEEE 829**

```
const PDF = ({ project, useCases, testCases }: pdfProps) => {
  return (
    <Document>
      <Page style={styles.page}>
        <Text style={styles.title}>Plan de Pruebas - Formato IEEE 829</Text>

        {/* Información del Proyecto */}
        <View style={styles.section}>
          <Text style={styles.title}>Proyecto: {removeHtmlTags(project.name)}</Text>
          <Text style={styles.section}>Descripción: {removeHtmlTags(project.description)}</Text>
        </View>

        {/* Casos de Uso y Casos de Prueba */}
        {useCases.map((useCase, useCaseIndex) => (
          <View key={useCaseIndex} style={styles.section}>
            <Text style={styles.title}>{removeHtmlTags(useCase.name)}</Text>
            {/* Tabla de Información del Caso de Uso */}
            <View style={styles.table}>
              <View style={styles.tableRow}>
                <View style={styles.tableColLabel}>
                  <Text style={styles.tableCell}>Nombre</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>{removeHtmlTags(useCase.name)}</Text>
                </View>
              </View>
              <View style={styles.tableRow}>
                <View style={styles.tableColLabel}>
                  <Text style={styles.tableCell}>Descripción</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>{removeHtmlTags(useCase.description)}</Text>
                </View>
              </View>
              <View style={styles.tableRow}>
                <View style={styles.tableColLabel}>
                  <Text style={styles.tableCell}>Precondiciones</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>{removeHtmlTags(useCase.preconditions)}</Text>
                </View>
              </View>
              <View style={styles.tableRow}>
                <View style={styles.tableColLabel}>
                  <Text style={styles.tableCell}>Postcondiciones</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>{removeHtmlTags(useCase.postconditions)}</Text>
                </View>
              </View>
              <View style={styles.tableRow}>
                <View style={styles.tableColLabel}>
                  <Text style={styles.tableCell}>Flujo Principal</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>{removeHtmlTags(useCase.mainFlow)}</Text>
                </View>
              </View>
              <View style={styles.tableRow}>
                <View style={styles.tableColLabel}>
                  <Text style={styles.tableCell}>Flujos Alternos</Text>
```

Figura 30: Generar reportes de proyectos en formato IEEE 829



```
                  <Text style={styles.tableCell}>Flujos Alternos</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>{removeHtmlTags(useCase.alternateFlows)}</Text>
                </View>
              </View>
            </View>

            {/* Tabla de Casos de Prueba relacionados */}
            <Text style={{ marginTop: 10, fontWeight: "bold" }}>Casos de Prueba Relacionados:</Text>
            <View style={styles.table}>
              <View style={[styles.tableRow, styles.tableHeader]}>
                <View style={styles.tableColLabel}>
                  <Text style={styles.tableCell}>ID</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>Nombre</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>Descripción</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>Pasos</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>Datos de Entrada</Text>
                </View>
                <View style={styles.tableColValue}>
                  <Text style={styles.tableCell}>Resultado Esperado</Text>
                </View>
              </View>
              {testCases
                .filter((testCase) => testCase.useCaseId === useCase.id)
                .map((testCase, testCaseIndex) => (
                  <View key={testCaseIndex} style={styles.tableRow}>
                    <View style={styles.tableColLabel}>
                      <Text style={styles.tableCell}>{removeHtmlTags(testCase.code)}</Text>
                    </View>
                    <View style={styles.tableColValue}>
                      <Text style={styles.tableCell}>{removeHtmlTags(testCase.name)}</Text>
                    </View>
                    <View style={styles.tableColValue}>
                      <Text style={styles.tableCell}>{removeHtmlTags(testCase.description)}</Text>
                    </View>
                    <View style={styles.tableColValue}>
                      <Text style={styles.tableCell}>{removeHtmlTags(testCase.steps)}</Text>
                    </View>
                    <View style={styles.tableColValue}>
                      <Text style={styles.tableCell}>{removeHtmlTags(testCase.inputData)}</Text>
                    </View>
                    <View style={styles.tableColValue}>
                      <Text style={styles.tableCell}>{removeHtmlTags(testCase.expectedResult)}</Text>
                    </View>
                  </View>
                ))}
            </View>
          </View>
        ))}
      </Page>
    </Document>
  );
};

export default PDF;
```

Figura 31: Generar reportes de proyectos en formato PDF

4.4. Pruebas.

Tabla 13: Pruebas Unitarias Cuarta Iteración

Código	Componente	Descripción de la Prueba	Resultado
PU-017	Generar casos de prueba funcionales	Valida que se generen N casos de prueba funcionales a partir del id de un caso de uso que se envíe. La respuesta del servidor debe de tener un código 200 y devolver la lista de casos de prueba funcionales generados	Aprobado
PU-018	Actualizar caso de prueba funcional	Valida que se actualice un caso de prueba funcional con datos válidos. La respuesta del servidor debe de tener un código 200 y devolver un mensaje 'Caso de Prueba Funcional Actualizado Correctamente'	Aprobado
PU-019	Eliminar caso de prueba funcional	Comprueba la eliminación lógica exitosa de un caso de prueba funcional. La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Prueba Funcional Eliminado Correctamente'	Aprobado
PU-020	Listar casos de prueba funcionales	Comprueba que se devuelva una lista de casos de prueba funcionales de un caso de uso en el sistema. La respuesta del servidor debe de tener un código 200 y devolver la lista de casos de prueba funcionales de un caso de uso	Aprobado

Tabla 14: Pruebas Funcionales Cuarta Iteración

Código	Componente	Resultado
PF-012	Generación de casos de prueba funcionales	Aprobado
PF-013	Eliminar caso de prueba funcional	Aprobado
PF-014	Compartir Proyecto	Aprobado
PF-015	Unirse a Proyecto	Aprobado
PF-016	Activar Usuario	Aprobado
PF-017	Desactivar Usuario	Aprobado
PF-018	Cambiar Rol de Usuario	Aprobado
PF-019	Generar PDF de Proyecto	Aprobado

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 9. Pruebas Unitarias.

[Pruebas Unitarias]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

1. Introducción

El presente documento es el detalle de las pruebas unitarias generadas correspondientes a la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso, desarrollada como parte del trabajo de integración curricular de la carrera de computación en la Universidad Nacional de Loja. Esta herramienta está diseñada para ayudar a los usuarios a diseñar casos de prueba funcionales a partir de casos de uso asociados a proyectos de software, utilizando el IA.

2. Objetivo

El presente documento detalla las pruebas unitarias generadas para asegurar la funcionalidad y la calidad de la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso. El objetivo de este informe es proporcionar una descripción de las pruebas unitarias implementadas, incluyendo los casos de prueba, los resultados esperados y los resultados obtenidos. Esto garantizará que cada componente de la aplicación funcione correctamente de manera aislada, contribuyendo a la fiabilidad y estabilidad del sistema en su conjunto.

3. Listado de pruebas unitarias realizadas:

Las pruebas unitarias se realizaron con las funcionalidades alojadas en el entorno del backend, pues en ellas radican la correcta funcionalidad individual de cada endpoint o función; para ello, se utilizó la herramienta de Jest que por defecto se integra en los proyectos de NestJs, optimizando la configuración de su entorno y agilizando las pruebas con archivos .spec.ts.

Tabla 1: PU-1

Prueba Unitaria		
Identificador	PU-01	
Descripción	Comprueba la autenticación de un usuario con credenciales válidas	
Componente	Usuario	
Método	Iniciar Sesión	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Email • Password	Status 200 (OK) que representa comprobación de credenciales correctas.	APROBADO

```
it('Debe devolver un token de acceso', async () => {
      const loginDto: LoginDto = { email: '@example.com', password: '12345' };
      const result = await authController.login(loginDto);
      expect(result).toEqual({ accessToken: 'fake-jwt-token' });
      expect(authService.login).toHaveBeenCalledWith(loginDto);
    });
```

Figura 1: Detalle prueba Unitaria 1

Tabla 2: PU-2

Prueba Unitaria		
Identificador	PU-02	
Descripción	Comprueba la autenticación de un usuario con credenciales inválidas	
Componente	Usuario	
Método	Iniciar Sesión	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Email • Password	Status 401 (Error) que representa que no puede acceder a los recursos restringidos y devolver un mensaje de 'Credenciales Incorrectas.	APROBADO

```
it('Debe lanzar un error por credenciales incorrectas', async () => {
      const loginDto: LoginDto = {
        email: 'incorrect@example.com',
        password: 'wrongpassword',
      };
      jest
        .spyOn(authService, 'login')
        .mockRejectedValueOnce(
          new UnauthorizedException(
            'No se encontró una cuenta con ese correo electrónico',
          ),
        );
      try {
        await authController.login(loginDto);
      } catch (error) {
        expect(error.response).toEqual({
          message: 'No se encontró una cuenta con ese correo electrónico',
          error: 'Unauthorized',
          statusCode: 401,
        });
        expect(error.status).toEqual(401);
      }
    });
```

Figura 2: Detalle prueba Unitaria 2

Tabla 3: PU-3

Prueba Unitaria		
Identificador	PU-03	
Descripción	Comprueba el envío de una OTP para la recuperación de la contraseña de un usuario	
Componente	Usuario	
Método	Recuperar Contraseña	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Email	Status 200 (OK) y devuelve un token con el OTP.	APROBADO

```
it('Debe enviar correo electrónico de recuperación de contraseña', async () => {
      const email = 'test@example.com';
      const result = await authController.passwordRecovery(email);
      expect(result).toEqual({ email, status: 'email sent' });
      expect(authService.passwordRecovery).toHaveBeenCalledWith(email);
    });
```

Figura 3: Detalle prueba Unitaria 3

Tabla 4: PU-4

Prueba Unitaria		
Identificador	PU-04	
Descripción	Valida la OTP ingresada por el usuario para recuperar su contraseña	
Componente	Usuario	
Método	Validar OTP	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• OTP	Status 200 (OK) y status TRUE.	APROBADO

```
it('Debe verificar el OTP', async () => {
      const verifyOtpDto: VerifyOtpDto = {
        token: 'tasfasfe124198yuw9d1n2d1',
        enteredOtp: '123456',
      };
      const result = await authController.verifyOtp(verifyOtpDto);
      expect(result).toEqual({ success: true });
      expect(authService.verifyOtp).toHaveBeenCalledWith(verifyOtpDto);
    });
```

Figura 4: Detalle prueba Unitaria 4

Tabla 5: PU-5

Prueba Unitaria		
Identificador	PU-05	
Descripción	Un usuario se pueda registrar con datos válidos.	
Componente	Usuario	
Método	Registrarse	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• FirstName • LastName • Email • Password	Status 200 (OK) y devuelve un mensaje de 'Registrado correctamente'.	APROBADO

```
it('Debe registrar un usuario', async () => {
      const registerDto: RegisterUserDto = {
        firstName: 'test',
        lastName: 'test',
        email: 'test@example.com',
        password: '12345',
      };
      const result = await authController.register(registerDto);
      expect(result).toEqual({ ...registerDto, id: 1 });
      expect(authService.register).toHaveBeenCalledWith(registerDto);
    });
```

Figura 5: Detalle prueba Unitaria 5

Tabla 6: PU-6

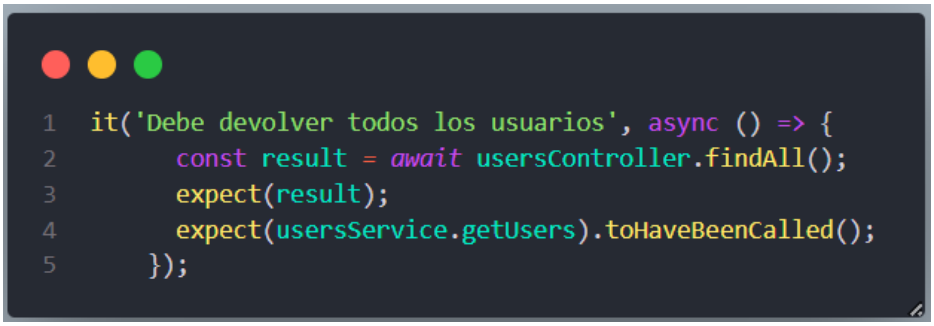
Prueba Unitaria		
Identificador	PU-06	
Descripción	Valida que un administrador pueda cambiar el rol de un usuario	
Componente	Usuario	
Método	Cambiar rol de usuario	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UserId • Role	Status 200 (OK) y devolver un mensaje de 'Usuario Actualizado'.	APROBADO

```
it('Debe cambiar el rol de usuario', async () => {
      const updateUserDto: UpdateUserDto = {
        role: 'Administrator',
      };
      const updatedUser = {
        id: '1',
        role: { name: updateUserDto.role },
      };
      mockUsersService.updateUser.mockResolvedValue(updatedUser);
      const result = await usersController.update('1', updateUserDto);
      expect(result).toEqual(updatedUser);
      expect(usersService.updateUser).toHaveBeenCalledWith('1', updateUserDto);
    });
```

Figura 6: Detalle prueba Unitaria 6

Tabla 7: PU-7

Prueba Unitaria		
Identificador	PU-07	
Descripción	Comprueba que se devuelva una lista de usuarios registrados en el sistema.	
Componente	Usuario	
Método	Listar Usuarios	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• NA	Status 200 (OK) y devuelve la lista de todos los usuarios'.	APROBADO



```
1  it('Debe devolver todos los usuarios', async () => {
2        const result = await usersController.findAll();
3        expect(result);
4        expect(usersService.getUsers).toHaveBeenCalled();
5      });
```

Figura 7: Detalle prueba Unitaria 7

Tabla 8: PU-8

Prueba Unitaria		
Identificador	PU-08	
Descripción	Valida que un proyecto se cree correctamente con datos válidos.	
Componente	Proyectos	
Método	Crear Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UserId • Name • Description	Status 200 (OK) y status TRUE.	APROBADO

```
it('Debe crear un proyecto', async () => {
      const createProjectDto: CreateProjectDto = {
        name: 'New Project',
        description: 'Project description',
        userId: '1',
      };
      const result = await projectsController.create(createProjectDto);
      expect(result).toEqual({ ...createProjectDto, id: '1' });
      expect(projectsService.create).toHaveBeenCalledWith(createProjectDto);
    });
```

Figura 8: Detalle prueba Unitaria 8

Tabla 9: PU-9

Prueba Unitaria		
Identificador	PU-09	
Descripción	Valida que un proyecto se actualice correctamente con datos válidos.	
Componente	Proyectos	
Método	Actualizar Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Name • Description	Status 200 (OK) y devuelve un mensaje de 'Proyecto Actualizado Correctamente'.	APROBADO

```
it('Debe actualizar un proyecto', async () => {
      const id = '1';
      const updateProjectDto: UpdateProjectDto = { name: 'Updated Project' };
      const result = await projectsController.update(id, updateProjectDto);
      expect(result).toEqual({ projectId: id, ...updateProjectDto });
      expect(projectsService.update).toHaveBeenCalledWith(id, updateProjectDto);
    });
```

Figura 9: Detalle prueba Unitaria 9

Tabla 10: PU-10

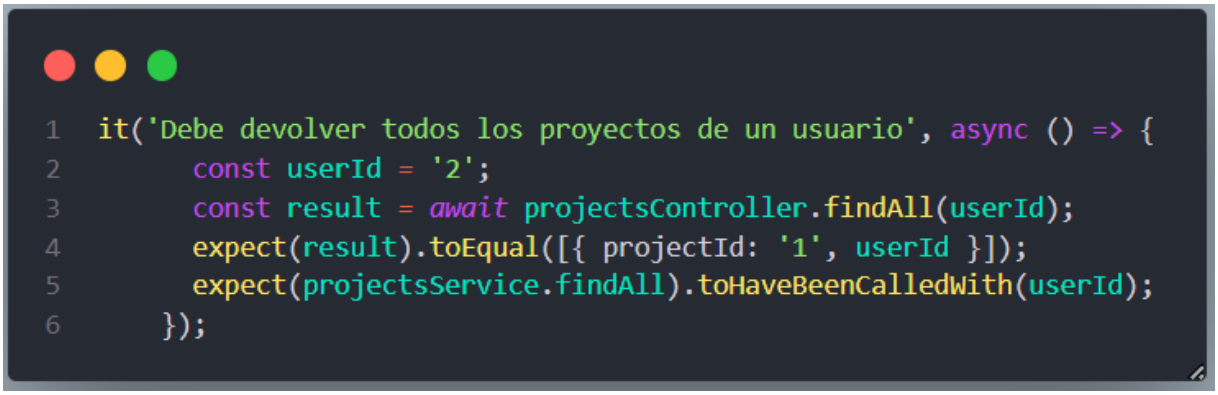
Prueba Unitaria		
Identificador	PU-010	
Descripción	Comprueba la eliminación lógica exitosa de un proyecto.	
Componente	Proyecto	
Método	Eliminar Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• NA	Status 200 (OK) y devolver un mensaje de 'Proyecto Eliminado Correctamente.	APROBADO

```
1  it('Debe eliminar un proyecto', async () => {
2        const id = '1';
3        const result = await projectsController.remove(id);
4        expect(result).toEqual({ projectId: id, removed: true });
5        expect(projectsService.remove).toHaveBeenCalledWith(id);
6      });
```

Figura 10: Detalle prueba Unitaria 10

Tabla 11: PU-11

Prueba Unitaria		
Identificador	PU-011	
Descripción	Comprueba que se devuelva una lista de proyectos de un usuario registrado en el sistema.	
Componente	Proyecto	
Método	Listar Proyectos	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UserId	La respuesta del servidor debe de tener un código 200 y devolver la lista de proyectos del usuario.	APROBADO



```
it('Debe devolver todos los proyectos de un usuario', async () => {
      const userId = '2';
      const result = await projectsController.findAll(userId);
      expect(result).toEqual([{ projectId: '1', userId }]);
      expect(projectsService.findAll).toHaveBeenCalledWith(userId);
    });
```

Figura 11: Detalle prueba Unitaria 11

Tabla 12: PU-12

Prueba Unitaria		
Identificador	PU-012	
Descripción	Comprueba que un código de proyecto exista y a este, un usuario sin registro previo al proyecto se una.	
Componente	Proyecto	
Método	Compartir Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Code • UserId	Status 200 (OK) y devolver un mensaje de 'Se ha unido al proyecto correctamente'.	APROBADO

```
it('Debe compartir un proyecto', async () => {
      const shareProjectDto: ShareProjectDto = { code: '1', userId: '2' };
      const result = await projectsController.shareProject(shareProjectDto);
      expect(result).toEqual({ ...shareProjectDto, shared: true });
      expect(projectsService.shareProject).toHaveBeenCalledWith(
        shareProjectDto,
      );
    });
```

Figura 12: Detalle prueba Unitaria 12

Tabla 13: PU-13

Prueba Unitaria		
Identificador	PU-013	
Descripción	Valida que un caso de uso se cree correctamente con datos válidos dentro de un proyecto.	
Componente	Casos de Uso	
Método	Crear Caso de Uso	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Code • Name • Description • Preconditions • Postconditions • MainFlow • AlternateFlows	Status 200 (OK) y devolver un mensaje de 'Caso de Uso Creado Correctamente'.	APROBADO

```
it('Debe crear un caso de uso', async () => {
      const createUseCaseDto: CreateUseCaseDto = {
        code: '123',
        name: 'New Usecase',
        description: 'Usecase description',
        preconditions: 'test',
        postconditions: 'test',
        mainFlow: 'inicia y termina',
        projectId: '1',
        alternateFlows: 'termina e inicia',
      };
      const result = await usecasesController.create(createUseCaseDto);
      expect(result).toEqual({ ...createUseCaseDto, id: '1' });
      expect(usecasesService.create).toHaveBeenCalledWith(createUseCaseDto);
    });
```

Figura 13: Detalle prueba Unitaria 13

Tabla 13: PU-13

Prueba Unitaria	
Identificador	PU-014
Descripción	Valida que un caso de uso se actualice correctamente con datos válidos.

Componente	Caso de Uso	
Método	Actualizar caso de uso	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Code • Name • Description • Preconditions • Postconditions • MainFlow • AlternateFlows	Status 200 (OK) y devolver un mensaje de 'Caso de Uso Actualizado Correctamente'.	APROBADO

```
it('Debe actualizar un caso de uso', async () => {
      const id = '1';
      const updateUseCaseDto: UpdateUseCaseDto = { name: 'Updated Usecase' };
      const result = await usecasesController.update(id, updateUseCaseDto);
      expect(result).toEqual({ usecaseId: id, ...updateUseCaseDto });
      expect(usecasesService.update).toHaveBeenCalledWith(id, updateUseCaseDto);
    });
```

Figura 14: Detalle prueba Unitaria 14

Tabla 15: PU-15

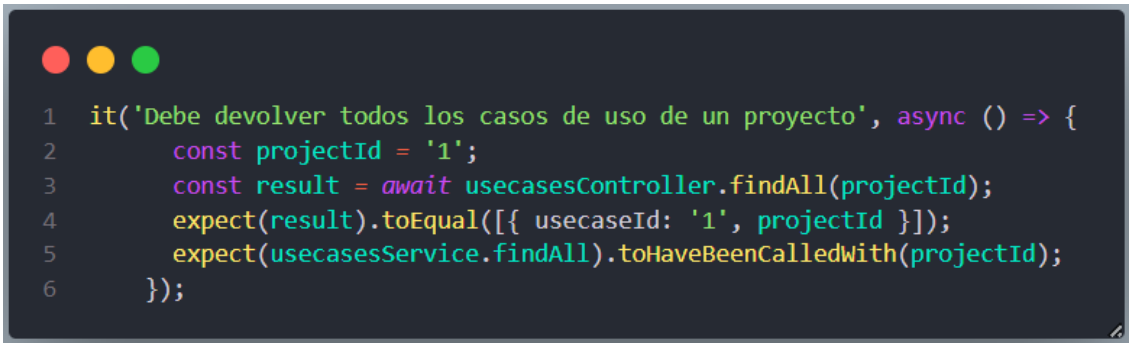
Prueba Unitaria		
Identificador	PU-015	
Descripción	Comprueba la eliminación lógica exitosa de un caso de uso.	
Componente	Caso de Uso	
Método	Eliminar caso de uso	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UseCaseId	Status 200 (OK) y devolver un mensaje de 'Caso de Uso Eliminado Correctamente'.	APROBADO

```
it('Debe eliminar un caso de uso', async () => {
      const id = '1';
      const result = await usecasesController.remove(id);
      expect(result).toEqual({ usecaseId: id, removed: true });
      expect(usecasesService.remove).toHaveBeenCalledWith(id);
    });
  });
```

Figura 15: Detalle prueba Unitaria 15

Tabla 16: PU-16

Prueba Unitaria		
Identificador	PU-016	
Descripción	Comprueba que se devuelva una lista de casos de uso de un proyecto en el sistema.	
Componente	Caso de Uso	
Método	Listar casos de uso	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• ProjectId	Status 200 (OK) y devolver la lista de casos de uso del proyecto	APROBADO



```
it('Debe devolver todos los casos de uso de un proyecto', async () => {
      const projectId = '1';
      const result = await usecasesController.findAll(projectId);
      expect(result).toEqual([{ usecaseId: '1', projectId }]);
      expect(usecasesService.findAll).toHaveBeenCalledWith(projectId);
    });
```

Figura 16: Detalle prueba Unitaria 16

Tabla 17: PU-17

Prueba Unitaria		
Identificador	PU-017	
Descripción	Valida que se generen N casos de prueba funcionales a partir del id de un caso de uso que se envíe	
Componente	Caso de Prueba Funcional	
Método	Generar casos de prueba funcionales	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UseCaseId	Status 200 (OK) y devolver la lista de casos de prueba funcionales generados.	APROBADO

```
it('Debe generar casos de prueba funcionales', async () => {
      const id = '1';
      const result = await testcasesController.generate(id);
      expect(result).toEqual({ id, generated: true });
      expect(testcasesService.generateTestCase).toHaveBeenCalledWith(id);
    });
```

Figura 17: Detalle prueba Unitaria 17

Tabla 18: PU-18

Prueba Unitaria		
Identificador	PU-018	
Descripción	Valida que se actualice un caso de prueba funcional con datos válidos	
Componente	Caso de Prueba Funcional	
Método	Actualizar caso de prueba funcional	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Code • Name • Description • Steps • ExpectedResult	Status 200 (OK) y devolver un mensaje 'Caso de Prueba Funcional Actualizado Correctamente'	APROBADO

• Input data		

```
it('Debe actualizar un caso de prueba funcional', async () => {
      const id = '1';
      const updateTestCaseDto: UpdateTestCaseDto = {
        name: 'Updated Test Case',
      };
      const result = await testcasesController.update(id, updateTestCaseDto);
      expect(result).toEqual({ testCaseId: id, ...updateTestCaseDto });
      expect(testcasesService.update).toHaveBeenCalledWith(
        id,
        updateTestCaseDto,
      );
    });
```

Figura 18: Detalle prueba Unitaria 18

Tabla 19: PU-19

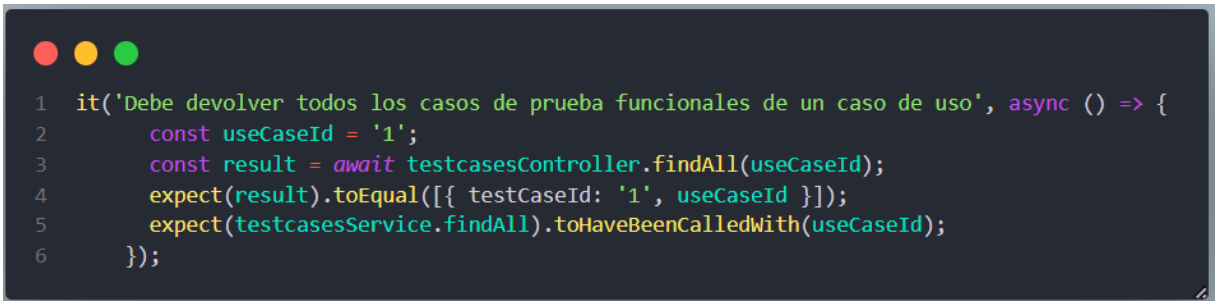
Prueba Unitaria		
Identificador	PU-019	
Descripción	Comprueba la eliminación lógica exitosa de un caso de prueba funcional.	
Componente	Caso de Prueba Funcional	
Método	Eliminar caso de prueba funcional	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
TestCaseId	Status 200 (OK) y devolver un mensaje de 'Caso de Prueba Funcional Eliminado Correctamente'.	APROBADO

```
it('Debe eliminar un caso de prueba funcional', async () => {
      const id = '1';
      const result = await testcasesController.remove(id);
      expect(result).toEqual({ testCaseId: id, removed: true });
      expect(testcasesService.remove).toHaveBeenCalledWith(id);
    });
```

Figura 19: Detalle prueba Unitaria 19

Tabla 20: PU-20

Prueba Unitaria		
Identificador	PU-020	
Descripción	Comprueba que se devuelva una lista de casos de prueba funcionales de un caso de uso en el sistema.	
Componente	Caso de Prueba Funcional	
Método	Listar casos de prueba funcionales	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
UseCaseId	Status 200 (OK) y devolver la lista de casos de prueba funcionales de un caso de uso	APROBADO



```
it('Debe devolver todos los casos de prueba funcionales de un caso de uso', async () => {
      const useCaseId = '1';
      const result = await testcasesController.findAll(useCaseId);
      expect(result).toEqual([{ testCaseId: '1', useCaseId }]);
      expect(testcasesService.findAll).toHaveBeenCalledWith(useCaseId);
    });
```

Figura 20: Detalle prueba Unitaria 20

Para constancia de la valides de este documento:

Acción	**Funcionario**	**Firma**
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 10. Pruebas Funcionales.

[Pruebas Funcionales]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

1. Introducción

El presente documento es el detalle de las pruebas funcionales generadas correspondientes a la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso, desarrollada como parte del trabajo de integración curricular de la carrera de computación en la Universidad Nacional de Loja. Esta herramienta está diseñada para ayudar a los usuarios a diseñar casos de prueba funcionales a partir de casos de uso asociados a proyectos de software, utilizando el IA.

2. Objetivo

El presente documento detalla las pruebas funcionales generadas en la sección del Frontend para asegurar la funcionalidad y la calidad de la aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso. El objetivo de este informe es proporcionar una descripción y muestra de las pruebas funcionales implementadas, incluyendo los casos de prueba, los resultados esperados y los resultados obtenidos. Esto garantizará que cada componente de la aplicación funcione correctamente de manera aislada, contribuyendo a la fiabilidad y estabilidad del sistema en su conjunto.

3. Listado de pruebas funcionales realizadas:

Las pruebas funcionales se elaboraron en el entorno del Frontend, utilizando la librería de Cypress, la cual permite generar scripts que manejen de manera dinámica la vista del sistema, simulando un funcionamiento natural o adverso.

Tabla 1: PF-1

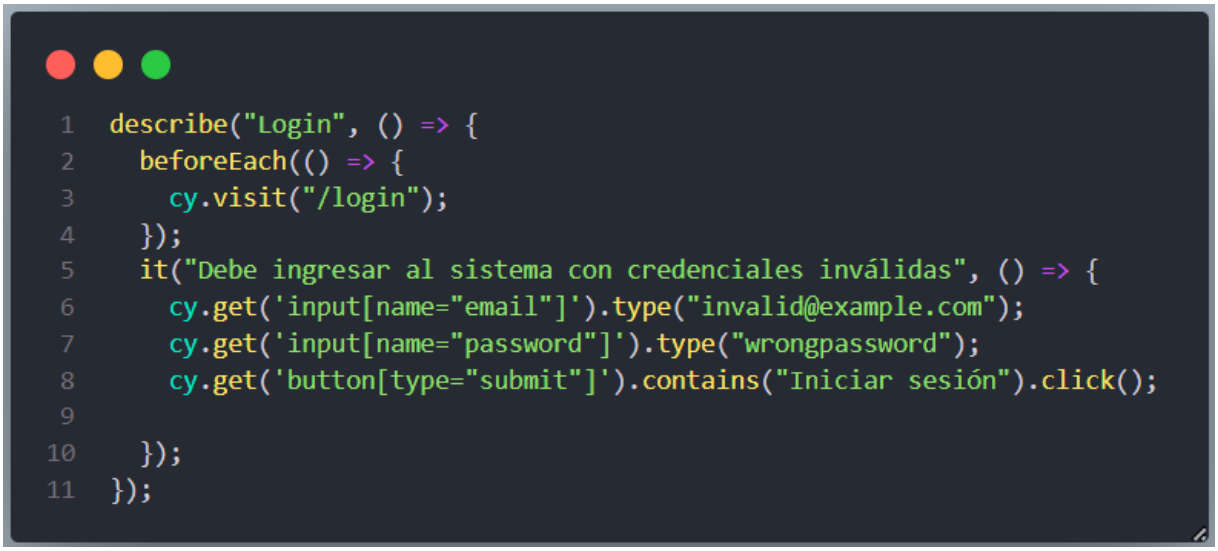
Prueba Funcional		
Identificador	PF-01	
Descripción	Valida el ingreso al sistema mediante autenticación de credenciales	
Componente	Usuario	
Método	Inicio de Sesión	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Email • Password	El sistema debe iniciar sesión con credenciales válidas al Dashboard de la aplicación web	APROBADO

```
describe("Login", () => {
  beforeEach(() => {
    cy.visit("/login");
  });

  it("Debe ingresar al sistema con credenciales válidas", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]').should("be.visible").type("juan1@gmail.com");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
  });
});
```

Figura 1: Detalle prueba Funcional 1

Tabla 2: PF-2

Prueba Funcional		
Identificador	PF-02	
Descripción	Valida el no ingreso al sistema mediante autenticación con credenciales inválidas	
Componente	Usuario	
Método	Inicio de Sesión	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Email • Password	El sistema debe mostrar un mensaje de “credenciales incorrectas” al intentar iniciar sesión con credenciales inválidas	APROBADO



```
describe("Login", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Debe ingresar al sistema con credenciales inválidas", () => {
    cy.get('input[name="email"]').type("invalid@example.com");
    cy.get('input[name="password"]').type("wrongpassword");
    cy.get('button[type="submit"]').contains("Iniciar sesión").click();

  });
});
```

Figura 2: Detalle prueba Funcional 2

Tabla 3: PF-3

Prueba Funcional		
Identificador	PF-03	
Descripción	Comprueba que el usuario pueda recuperar su contraseña al olvidarla	
Componente	Usuario	
Método	Recuperar Contraseña	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Email	El sistema debe permitir cambiar la contraseña de un usuario mediante una validación de una OTP enviada al correo del usuario	APROBADO

```
describe("Recuperar Contraseña", () => {
    beforeEach(() => {
      cy.visit("/login");
    });

it('Recuperar Contraseña', () => {
    cy.contains('Olvidé mi contraseña').click();
    cy.get('form').within(() => {
      cy.get('input[name="emailOTP"]').should('be.visible').type("juan.f.castillo.e@unl.edu.ec");
      cy.get('button[type="submit"]')
        .contains("Enviar correo de recuperación")
        .should("be.visible")
        .click();
    });
  });
});
```

Figura 3: Detalle prueba Funcional 3

Tabla 4: PF-4

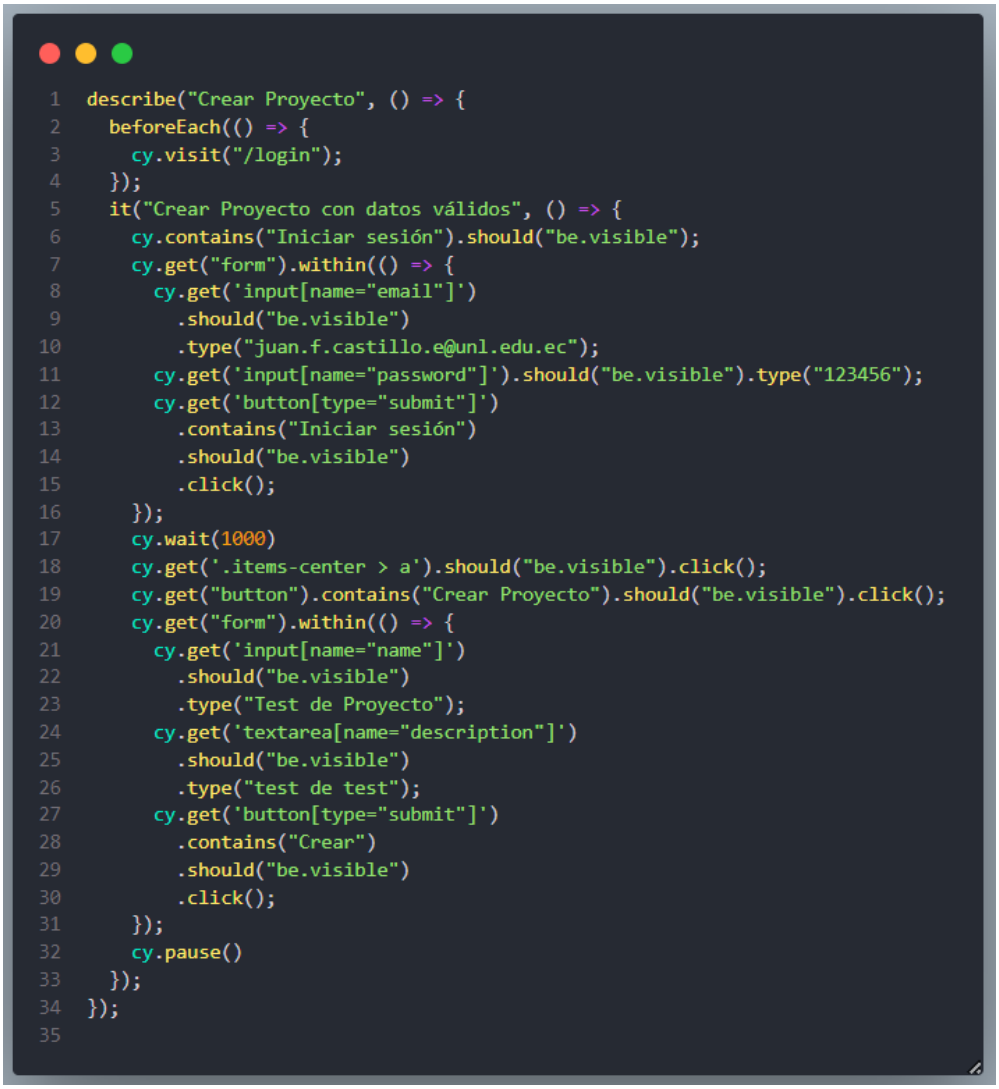
Prueba Funcional		
Identificador	PF-04	
Descripción	Comprueba que el usuario pueda registrarse con datos válidos	
Componente	Usuario	
Método	Registrarse	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• FirstName • LastName • Email • Password	El sistema debe permitir iniciar sesión en el sistema inmediatamente después de haberse registrado en el sistema	APROBADO

```
describe("Registro", () => {
    beforeEach(() => {
      cy.visit("/login");
    });

it('Registrarse con credenciales válidas', () => {
    cy.contains('Regístrate').click();
    cy.get('form').within(() => {
      cy.get('input[name="firstName"]').should('be.visible').type("Juan ");
      cy.get('input[name="lastName"]').should('be.visible').type("Castillo");
      cy.get('input[name="email"]').should('be.visible').type("juan1@gmail.com");
      cy.get('input[name="password"]').should('be.visible').type("123456");
      cy.get('input[name="confirmPassword"]').should('be.visible').type("123456");
      cy.get('button[type="submit"]')
        .contains("Registro")
        .should("be.visible")
        .click();
    });
  });
});
```

Figura 4: Detalle prueba Funcional 4

Tabla 5: PF-5

Prueba Funcional		
Identificador	PF-05	
Descripción	Comprueba la creación de un proyecto en la cuenta de un usuario	
Componente	Proyecto	
Método	Crear Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Name • Description	El sistema debe permitir crear un proyecto y listarlo en la página de proyectos de un usuario	APROBADO



```
describe("Crear Proyecto", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Crear Proyecto con datos válidos", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000)
    cy.get('.items-center > a').should("be.visible").click();
    cy.get("button").contains("Crear Proyecto").should("be.visible").click();
    cy.get("form").within(() => {
      cy.get('input[name="name"]')
        .should("be.visible")
        .type("Test de Proyecto");
      cy.get('textarea[name="description"]')
        .should("be.visible")
        .type("test de test");
      cy.get('button[type="submit"]')
        .contains("Crear")
        .should("be.visible")
        .click();
    });
    cy.pause()
  });
});

```

Figura 5: Detalle prueba Funcional 5

Tabla 6: PF-6

Prueba Funcional		
Identificador	PF-06	
Descripción	Comprueba la actualización de un proyecto en una cuenta de un usuario	
Componente	Proyecto	
Método	Actualizar Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Name • Description	El sistema debe permitir actualizar un proyecto y mostrar un mensaje de proyecto actualizado.	APROBADO

```
describe("Actualizar Proyecto", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Actualizar Proyecto", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

    cy.contains("Editar").click();

    cy.get("form").within(() => {
      cy.get('button[type="submit"]')
        .contains("Actualizar")
        .should("be.visible")
        .click();
    });
  });
});

```

Figura 6: Detalle prueba Funcional 6

Tabla 7: PF-7

Prueba Funcional		
Identificador	PF-07	
Descripción	Comprueba la eliminación lógica de un proyecto	
Componente	Proyecto	
Método	Eliminar un Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
NA	La respuesta del servidor debe de tener un código 200 y devolver un mensaje de 'Caso de Prueba Funcional Eliminado Correctamente'.	APROBADO

```
describe("Eliminar Proyecto", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Eliminar Proyecto", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000)
    cy.get('.items-center > a').should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

    cy.contains("Eliminar").click();
    cy.contains("Eliminar").click();
  });
});

```

Figura 7: Detalle prueba Funcional 7

Tabla 8: PF-8

Prueba Funcional		
Identificador	PF-08	
Descripción	Comprueba la obtención de proyectos de un usuario	
Componente	Proyecto	
Método	Listar Proyectos	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UserId	El sistema debe listar los proyectos de un usuario	APROBADO

```
describe("Obtener Proyectos", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Obtener Proyectos", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000)
    cy.get('.items-center > a').should("be.visible").click();
  });
});

```

Figura 8: Detalle prueba Funcional 8

Tabla 9: PF-9

Prueba Funcional		
Identificador	PF-09	
Descripción	Comprueba la creación de un caso de uso en un proyecto	
Componente	Caso de Uso	
Método	Creación de un caso de uso	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Code • Name • Description • Preconditions • Postconditions • MainFlow • AlternateFlows	El sistema agrega un caso de uso a un proyecto de un usuario.	APROBADO

```
describe("Crear Caso de uso", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Crear caso de uso", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000)
    cy.get('.items-center > a').should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

    cy.contains("Ir al detalle").click();
    cy.get("button").contains("Crear Caso de Uso").should("be.visible").click();
    cy.get("form").within(() => {
      cy.get('input[name="code"]').should("be.visible").type("UC01");
      cy.get('textarea[name="name"]')
        .should("be.visible")
        .type("Test de Proyecto");
      cy.get('textarea[name="description"]')
        .should("be.visible")
        .type("test de test");
      cy.get('[data-testid="preconditions"]')
        .should("be.visible")
        .type("Precondiciones de prueba");
      cy.get('[data-testid="postconditions"]')
        .should("be.visible")
        .type("Postcondiciones de prueba");
      cy.get('[data-testid="mainFlow"]')
        .should("be.visible")
        .type("MainFlow de prueba");
      cy.get('[data-testid="alternateFlows"]')
        .should("be.visible")
        .type("AlternateFlows de prueba");

      cy.get('button[type="submit"]')
        .contains("Crear")
        .should("be.visible")
        .click();
    });
  });
});

```

Figura 9: Detalle prueba Funcional 9

Tabla 10: PF-10

Prueba Funcional		
Identificador	PF-010	
Descripción	Comprueba la actualización de un caso de uso en un proyecto	
Componente	Caso de Uso	
Método	Actualizar caso de uso	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Code • Name • Description • Preconditions • Postconditions • MainFlow • AlternateFlows	El sistema actualiza un caso de uso con los nuevos datos que se hayan modificado.	APROBADO

```
describe("Actualizar Caso de Uso", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Actualizar caso de uso", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

    cy.contains("Ir al detalle").click();

    cy.get('button[name="AbrirMenu"]').should("be.visible").click();

    cy.contains("Editar").click();

    cy.get("form").within(() => {
      cy.get('button[type="submit"]')
        .contains("Actualizar")
        .should("be.visible")
        .click();
    });
  });
});

```

Figura 10: Detalle prueba Funcional 10

Tabla 11: PF-11

Prueba Funcional		
Identificador	PF-011	
Descripción	Comprueba la eliminación lógica exitosa de un caso de uso.	
Componente	Caso de Uso	
Método	Eliminar Caso de Uso	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UseCaseId	El sistema permite eliminar un caso de uso y mostrar un mensaje de 'Caso de Uso Eliminado Correctamente'.	APROBADO

```
describe("Eliminar Caso de Uso", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Eliminar caso de uso", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

    cy.contains("Ir al detalle").click();

    cy.get('button[name="AbrirMenu"]').should("be.visible").click();

    cy.contains("Eliminar").click();
    cy.contains("Eliminar").click();
  });
});

```

Figura 11: Detalle prueba Funcional 11

Tabla 12: PF-12

Prueba Funcional		
Identificador	PF-012	
Descripción	Comprueba la generación exitosa de casos de prueba funcionales.	
Componente	Caso de Prueba Funcional	
Método	Generación de casos de prueba funcionales	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• UseCaseId	El sistema consume una IA y devuelve N casos de prueba para que el usuario escoja y guarde los que considere más útiles	APROBADO

```
describe("Generar Casos de Prueba Funcionales", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Generar casos de prueba funcionales", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

    cy.contains("Ir al detalle").click();

    cy.get('button[name="AbrirMenu"]').should("be.visible").click();

    cy.contains("Ir al detalle").click();

    cy.get("button")
      .contains("Generar Casos de Prueba Funcionales")
      .should("be.visible")
      .click();

    cy.get("button")
      .contains("Generar casos de prueba")
      .should("be.visible")
      .click();

    cy.wait(5000);

    cy.get("button").contains("Seleccionar").should("be.visible").click();

    cy.get("button")
      .contains("Guardar casos de prueba seleccionados (1)")
      .should("be.visible")
      .click();
  });
});
```

Figura 12: Detalle prueba Funcional 12

Tabla 13: PF-13

Prueba Funcional		
Identificador	PF-013	
Descripción	Comprueba la eliminación lógica exitosa de un caso de prueba funcional.	
Componente	Caso de Prueba Funcional	
Método	Eliminar caso de prueba funcional	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• TestCaseId	El sistema debe eliminar el caso de prueba funcional y devuelve un mensaje de 'Caso de Prueba Funcional Eliminado Correctamente'.	APROBADO

```
describe("Eliminar Caso de Prueba Funcional", () => {
  beforeEach(() => {
    cy.visit("/login");
  });

  it("Eliminar Caso de Prueba Funcional", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });

    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();
    cy.contains("Ir al detalle").click();
    cy.get('button[name="AbrirMenu"]').should("be.visible").click();
    cy.contains("Ir al detalle").click();

    cy.get('button[name="Open"]').eq(0).should("be.visible").click();

    cy.wait(2000);
    cy.contains("Eliminar", { timeout: 10000 })
      .eq(0)
      .should("be.visible")
      .click(); //
    cy.contains("Eliminar").eq(0).should("be.visible").click();
  });
});

```

Figura 13: Detalle prueba Funcional 13

Tabla 14: PF-14

Prueba Funcional		
Identificador	PF-014	
Descripción	Comprueba la funcionalidad de compartir un proyecto	
Componente	Proyecto	
Método	Compartir Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• Proyecto Creado	El sistema devuelve el código para copiar y compartir	APROBADO

```
describe("Compartir Proyecto", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Compartir Proyecto", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();

    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

    cy.contains("Compartir").click();

    cy.get('button[name="copy"]').should("be.visible").click();

  });
});

```

Figura 14: Detalle prueba Funcional 14

Tabla 15: PF-15

Prueba Funcional		
Identificador	PF-015	
Descripción	Comprueba el ingreso correcto a un proyecto	
Componente	Proyecto	
Método	Unirse a Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• NA	El sistema debe permitir unirse al proyecto y listarlo en el listado de proyectos	APROBADO

```
describe("Unirse a Proyecto", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Unirse a Proyecto", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan.f.castillo.e@unl.edu.ec");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").eq(0).should("be.visible").click();
    cy.get("button")
      .contains("Ingresar a Proyecto")
      .should("be.visible")
      .click();
    cy.get('input[name="codeProject"]').should("be.visible").type("7b7b0e22");
    cy.get("button")
      .contains("Unirme al Proyecto")
      .should("be.visible")
      .click();
    // cy.get('[y1="12"]').should("be.visible").click();
  });
});

```

Figura 15: Detalle prueba Funcional 15

Tabla 16: PF-16

Prueba Funcional		
Identificador	PF-016	
Descripción	Activar Usuario	
Componente	Usuario	
Método	Activar Usuario	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• NA	El sistema debe permitir cambiar el estado de un usuario a activo	APROBADO

```
describe("Activar Usuario", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Activar Usuario", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan1@gmail.com");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();
    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();
    cy.contains("Activar").click();
    cy.contains("Activar").eq(0).should("be.visible").click();
  });
});

```

Figura 16: Detalle prueba Funcional 16

Tabla 17: PF-17

Prueba Funcional	
Identificador	PF-017
Descripción	Desactivar Usuario
Componente	Usuario
Método	Desactivar Usuario
Responsable	Juan Francisco Castillo Estrella

Datos de entrada	Salida Esperada	Resultado
• NA	El sistema debe permitir cambiar el estado de un usuario a desactivado	APROBADO

```
describe("Desactivar Usuario", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Desactivar Usuario", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan1@gmail.com");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();
    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();
    cy.contains("Desactivar").click();
    cy.contains("Desactivar").eq(0).should("be.visible").click();
  });
});

```

Figura 17: Detalle prueba Funcional 17

Tabla 18: PF-18

Prueba Funcional		
Identificador	PF-016	
Descripción	Cambiar Rol de Usuario	
Componente	Usuario	
Método	Cambiar Rol de Usuario	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• NA	El sistema debe permitir cambiar el rol de un usuario	APROBADO

```
describe("Cambiar Rol", () => {
  beforeEach(() => {
    cy.visit("/login");
  });
  it("Cambiar Rol de Usuario", () => {
    cy.contains("Iniciar sesión").should("be.visible");
    cy.get("form").within(() => {
      cy.get('input[name="email"]')
        .should("be.visible")
        .type("juan1@gmail.com");
      cy.get('input[name="password"]').should("be.visible").type("123456");
      cy.get('button[type="submit"]')
        .contains("Iniciar sesión")
        .should("be.visible")
        .click();
    });
    cy.wait(1000);
    cy.get(".items-center > a").should("be.visible").click();
    cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();
    cy.contains("Cambiar Rol").click();
    cy.get('[role="combobox"]').should("be.visible").click();

    cy.get('[role="option"]').contains("Tester").click();
    cy.get('[role="combobox"]').should("contain", "Tester");
    cy.contains("Guardar").should("be.visible").click();
  });
});

```

Figura 18: Detalle prueba Funcional 18

Tabla 19: PF-19

Prueba Funcional		
Identificador	PF-019	
Descripción	Generar PDF de Proyecto	
Componente	Proyecto	
Método	Unirse a Proyecto	
Responsable	Juan Francisco Castillo Estrella	
Datos de entrada	**Salida Esperada**	**Resultado**
• NA	El sistema debe permitir generar el pdf de un proyecto	APROBADO

```
describe("Generar PDF Proyecto", () => {
    beforeEach(() => {
      cy.visit("/login");
    });
    it("Generar PDF Proyecto", () => {
      cy.contains("Iniciar sesión").should("be.visible");
      cy.get("form").within(() => {
        cy.get('input[name="email"]')
          .should("be.visible")
          .type("juan.f.castillo.e@unl.edu.ec");
        cy.get('input[name="password"]').should("be.visible").type("123456");
        cy.get('button[type="submit"]')
          .contains("Iniciar sesión")
          .should("be.visible")
          .click();
      });
      cy.wait(1000)
      cy.get('.items-center > a').should("be.visible").click();

      cy.get('button[name="Abrir Menu"]').eq(0).should("be.visible").click();

      cy.contains("Exportar en PDF").click();

    });
  });

```

Figura 19: Detalle prueba Funcional 19

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 11. Manual de Usuario.

[Manual de Usuario]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

1. Introducción

El presente documento es el manual de usuario de la Aplicación web para la estimación de costos mediante el método de puntos de función, desarrollada como parte del proyecto de integración curricular de la carrera de computación en la Universidad Nacional de Loja. Esta herramienta está diseñada para ayudar a los usuarios a calcular de manera eficiente los costos asociados a proyectos de software, utilizando el método de puntos de función.

2. Objetivo

En este manual, se describen las tareas y responsabilidades de dos tipos de usuarios principales: Administrador y Estudiante. Cada rol tiene acceso a diferentes funcionalidades de la aplicación, diseñadas para facilitar el proceso de estimación de costos y la gestión de proyectos de manera integral.

3. Dirigido a

Este manual de usuario está dirigido a los estudiantes y administradores del software web para la estimación de costos mediante el método de puntos de función en proyecto de software.

4. Tareas

- **Ingreso a la aplicación web.**

  Para acceder a la aplicación web, debe de dirigirse a su navegador de confianza, e ingresar la siguiente url: https://computacion.unl.edu.ec/testCaseCraft/

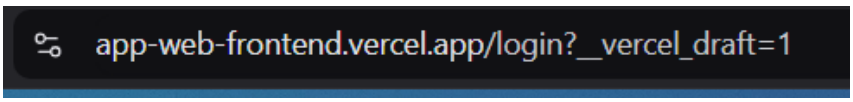


Figura 1: Accediendo a la aplicación web

Luego, se presentará la aplicación web.

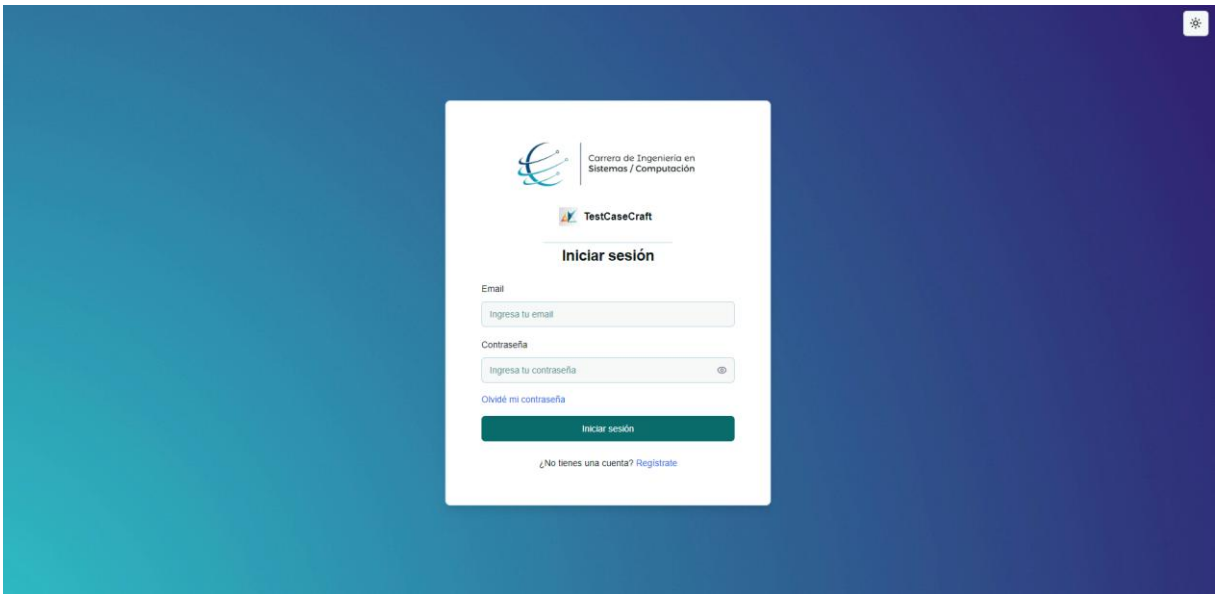


Figura 2: Aplicación web TestCaseCraft

Tareas rol: Tester

Registro de usuarios

Para acceder al formulario de registro se tiene que dirigir a la parte inferior del formulario de inicio de sesión en la opción “¿No tienes cuenta? Regístrate”.

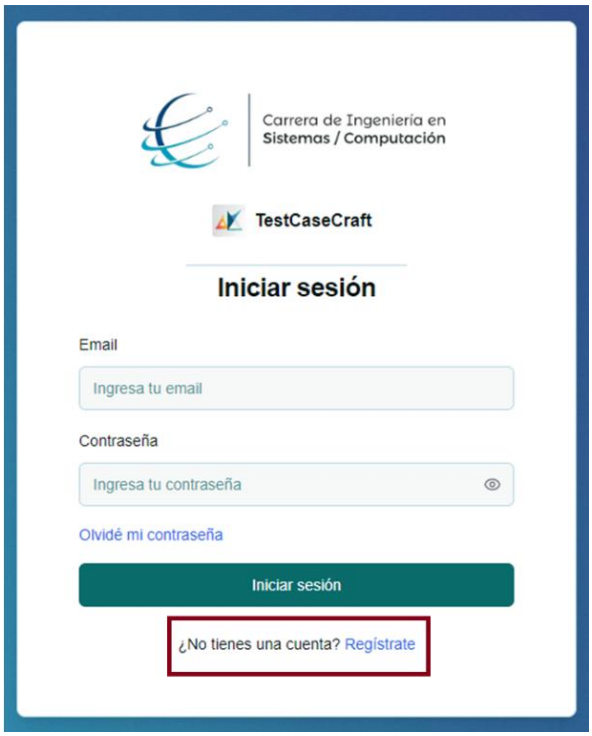


Figura 3: Accediendo a registro de usuarios

Luego de acceder se visualizará un formulario de registro, debe llenar todos los campos solicitados con la información correspondiente:

Figura 4: Formulario de Registro

Una vez ingresada la información solicitada, se procede a pulsar el botón de Registro:

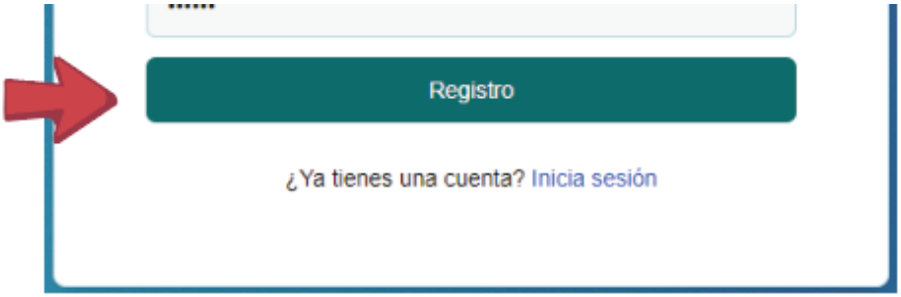


Figura 5: Registro en el sistema

Automáticamente cuando se registra se inicia sesión en la aplicación e ingresa a la página inicial del software.

Modo Claro y Oscuro

Al iniciar el sistema se presenta la pantalla de Bienvenida del aplicativo web TestCaseCraft

Figura 6: Página Inicial del sistema

Aquí se puede manipular el sistema para cambiar al gusto del usuario entre un modo claro y oscuro, haciendo click en el botón con un sol en la parte superior derecha de la pantalla; al pulsar este, el tema de la página cambia a modo oscuro, pues el modo claro viene por defecto.

Figura 7: Página Inicial del sistema en Modo Oscuro

Gestión de Proyectos

Para acceder a proyectos se puede hacer de dos formas:

- Pulsando el botón ¡Vamos Allá!
- Pulsando el item del SideBar en la parte izquierda de la pantalla que dice “Proyectos”.

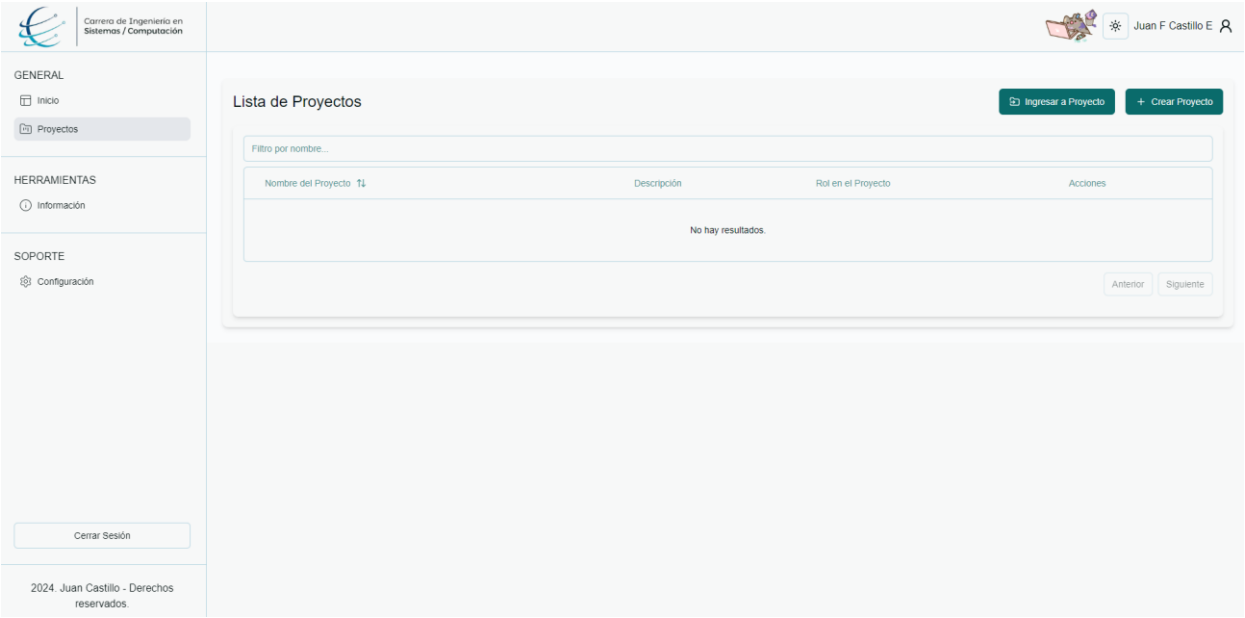


Figura 8: Página de Proyectos

Para crear un proyecto se pulsa el botón de la parte superior derecha que se llama “Crear Proyecto”

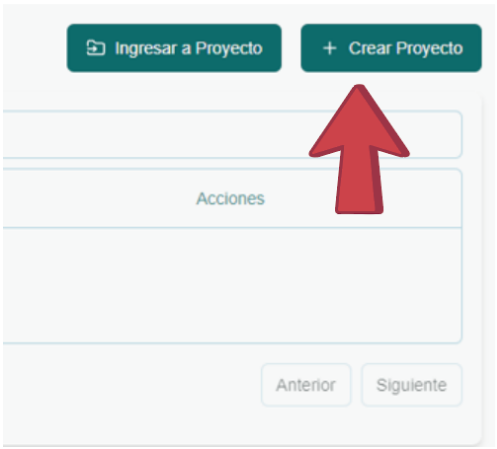


Figura 9: Opción Crear Proyecto

Al pulsar el botón se despliega un modal para ingresar la información del proyecto a crear

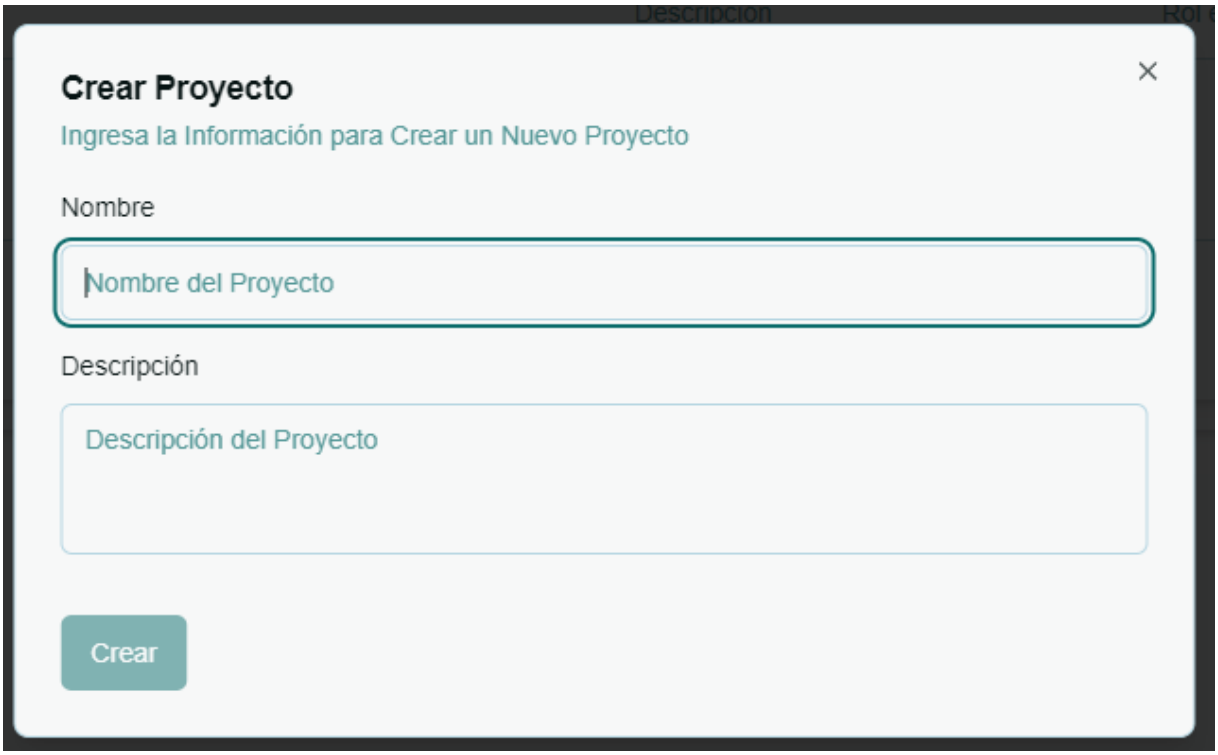


Figura 10: Modal de Formulario Crear Proyecto

Se puede llenar con información como la que se muestra a continuación:

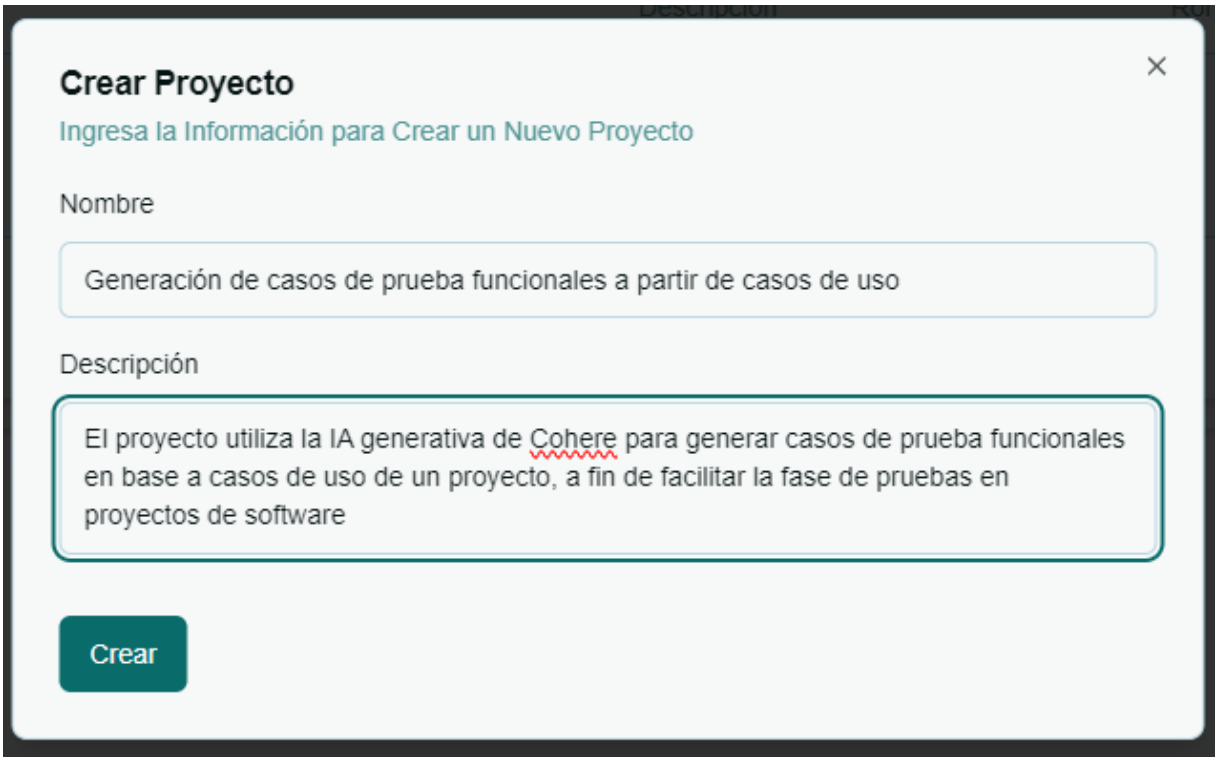


Figura 11: Formulario Crear Proyecto

Luego se pulsa el botón de Crear para que se guarde la información ingresada a un nuevo proyecto

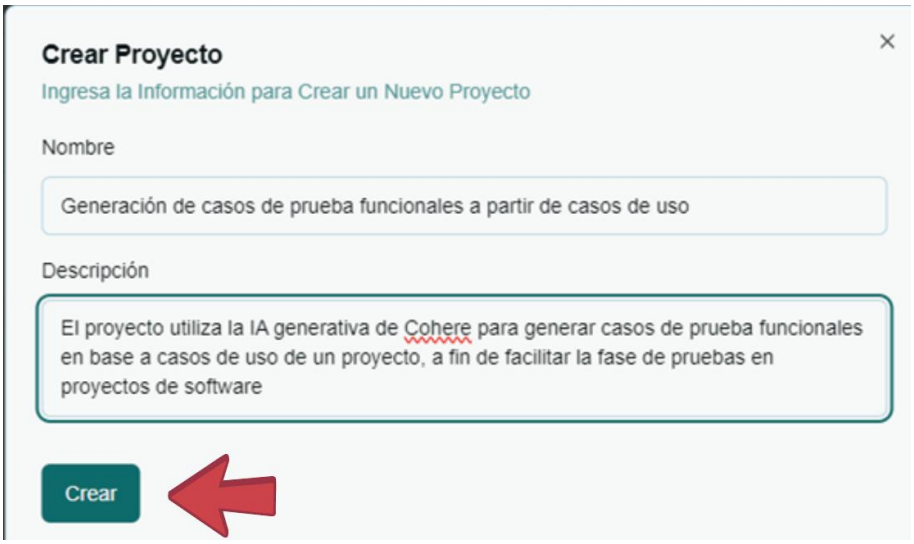


Figura 12: Crear Proyecto

Posteriormente se mostrará de nuevo la lista de proyectos con el proyecto ya creado y con la información manipulable

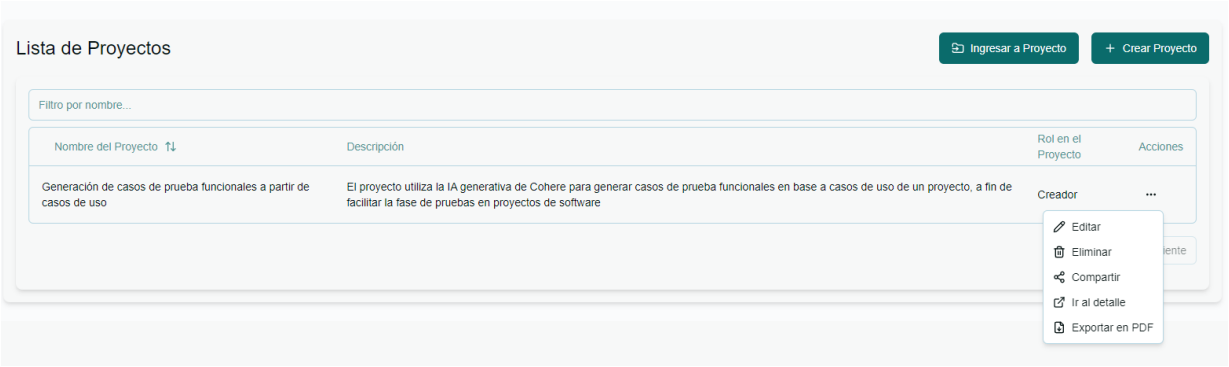


Figura 13: Lista de Proyectos con Proyecto Creado

Editar Proyecto

Para ingresar a las opciones del proyecto, se debe pulsar el botón con los tres puntos en la fila del proyecto que requerimos.

Figura 14: Opciones de proyecto

Entre las opciones que se le permite al usuario con el proyecto, se encuentra “editar” que permite modificar la información del proyecto.

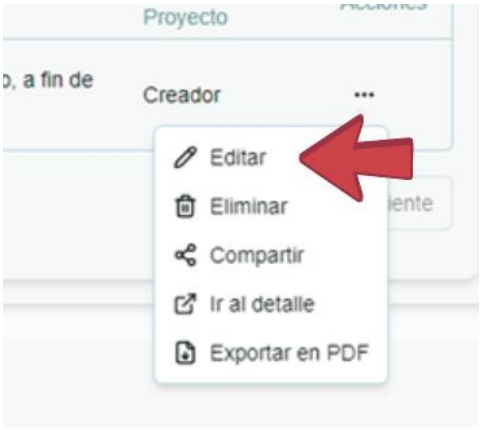


Figura 15: Ingreso a opción Editar

A continuación, se muestra el formulario con la información de proyecto a editar.

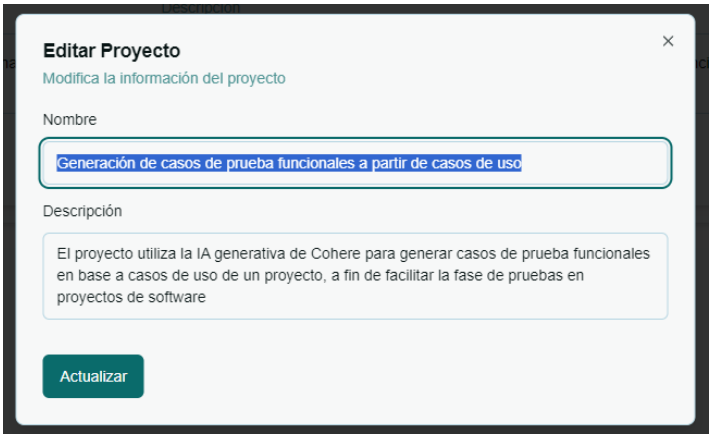


Figura 16: Formulario de Editar

Posterior a ello, se pulsa el botón actualizar y se guarda el proyecto con las modificaciones realizadas.

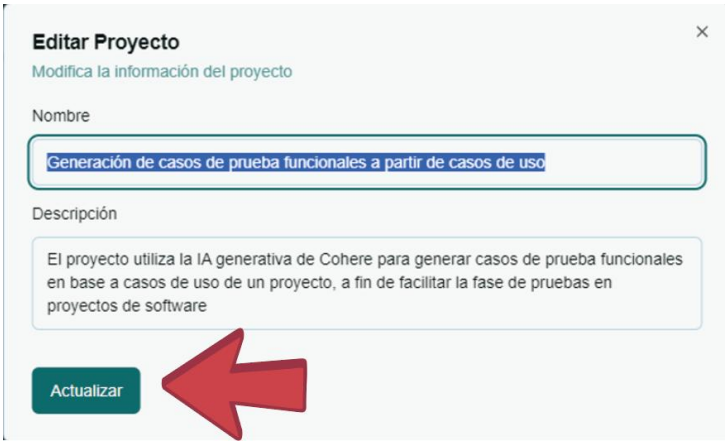


Figura 17: Formulario de Editar

Compartir Proyecto

Para compartir proyecto se debe ingresar a las opciones del proyecto

Figura 18: Opciones de proyecto

Para luego escoger la opción de Compartir Proyecto

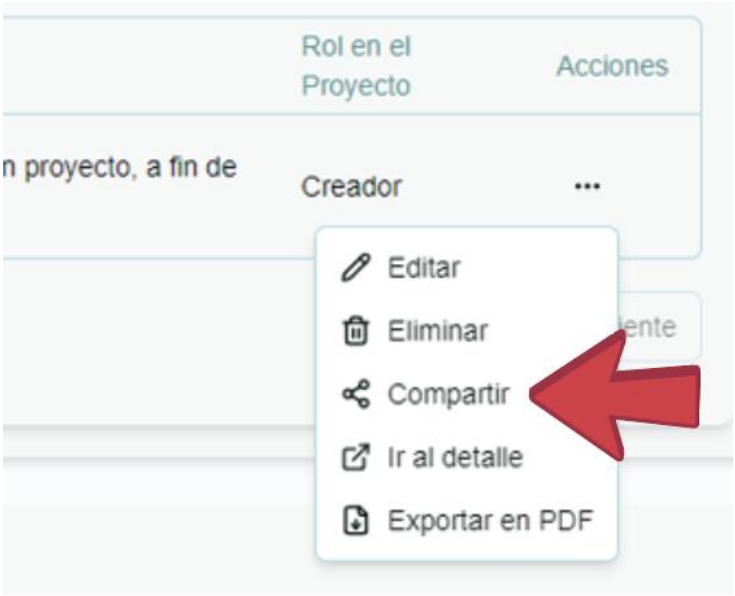


Figura 19: Opción Compartir

Cuando se pulse el botón de Compartir se abrirá un modal con el código de proyecto que se puede copiar para enviárselo al resto de colaboradores y se puedan unir.

Figura 20: Modal con código para compartir

Unirse a Proyecto

Para unirse a un proyecto se requiere que el dueño del proyecto al que se desea unir le comparta el código del proyecto, por tanto, con el código se puede proseguir con los siguientes pasos.

Primeramente, se dirige a la lista de proyectos y en la parte superior se ubica un botón de “Ingresar a Proyecto”

Figura 21: Ingresar a Proyecto

Luego, se abrirá un Modal para ingresar el código del proyecto al cual nos podemos unir:

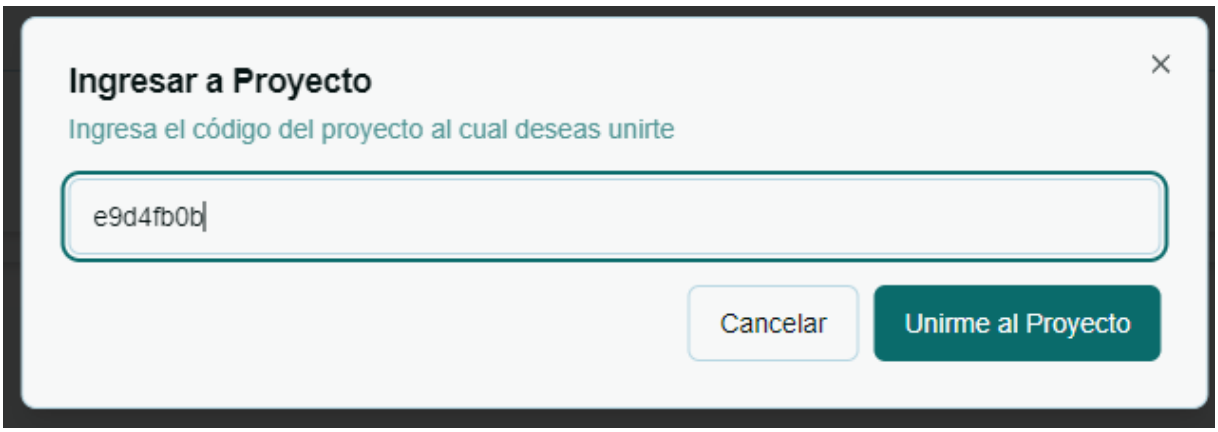


Figura 22: Modal Ingresar a Proyecto

Se pulsa el botón Unirme al Proyecto:

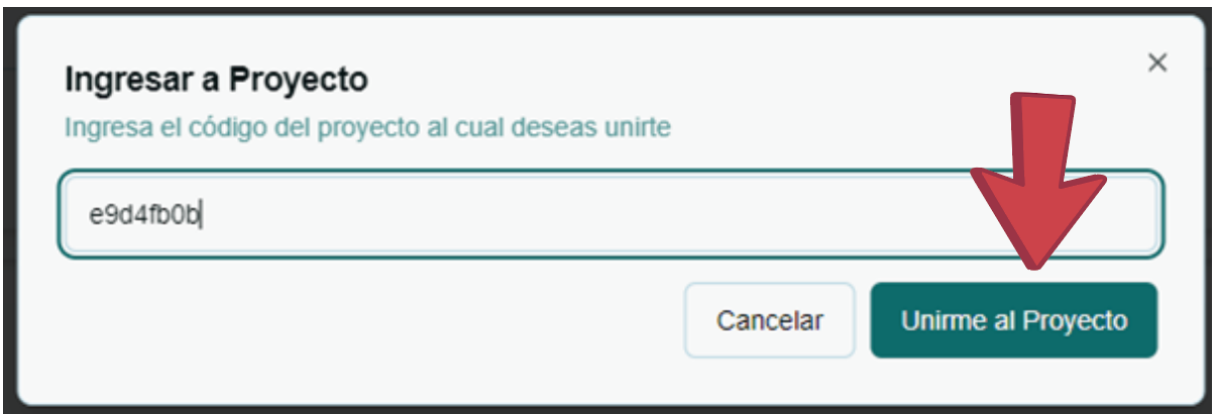


Figura 23: Modal Ingresar a proyecto

Al ingresar el código de un proyecto que se nos ha haya compartido y pulsar el botón de unirme al proyecto, automáticamente, el proyecto se agregará a la lista de proyectos, con la particularidad de que el rol del proyecto es “Invitado”.

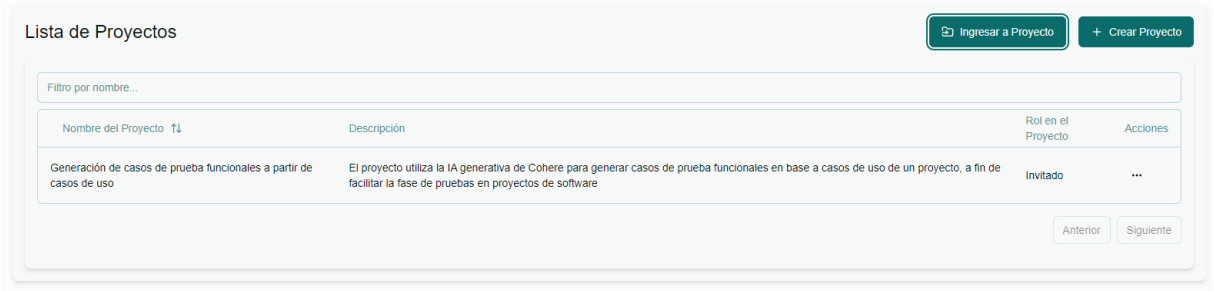


Figura 24: Lista de Proyecto con rol Invitado

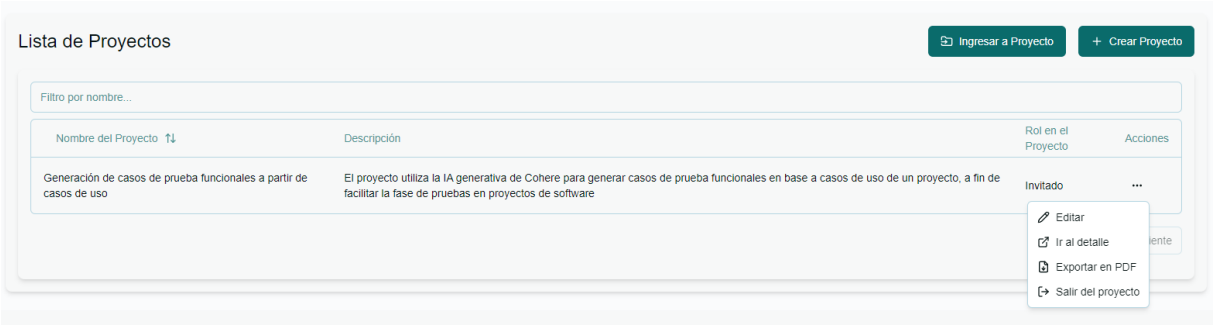


Figura 25: Lista de Proyecto con rol Invitado

Salir de Proyecto

Para salir de un proyecto, hay que tener en cuenta que esto solo se puede hacer cuando el usuario tiene rol de Invitado en el Proyecto.

Primeramente, se ingresa a las opciones del proyecto:

Figura 26: Opciones de proyecto

Para luego escoger la opción de Salir del Proyecto

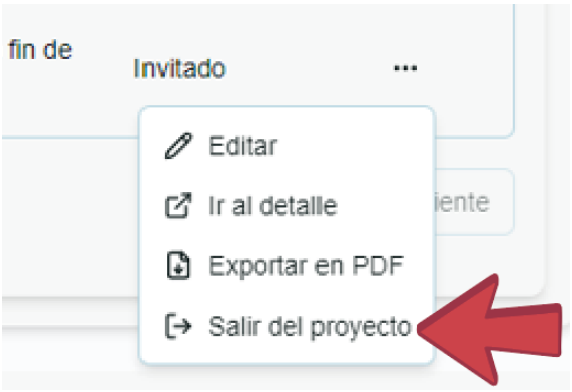


Figura 27: Opción Salir del Proyecto

Cuando se pulse el botón de Salir del Proyecto se abrirá un modal para confirmar la acción.

Figura 28: Modal de Salir del Proyecto

Ahora se pulsa el botón Salir para confirmar la acción.

Figura 29: Opción de Salir para confirmar acción de Salir del Proyecto

Aquí ya se elimina de la lista de proyectos, saliendo exitosamente del proyecto

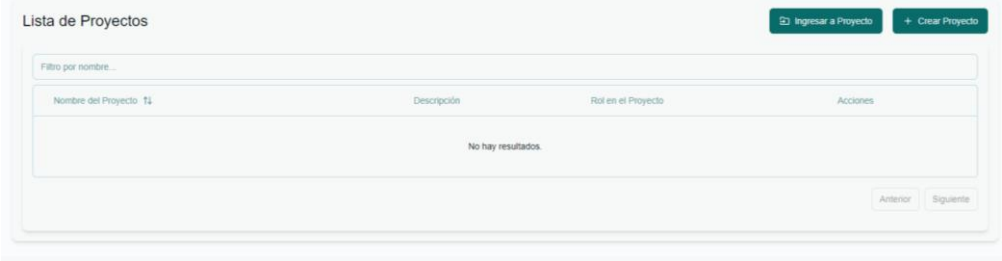


Figura 30: Lista de Proyectos saliendo del proyecto

Eliminar Proyecto

Para eliminar proyecto, se debe tener un proyecto creado, ser el creador y acceder a las opciones del proyecto.

Figura 31: Opciones de proyecto

Para luego escoger la opción de Eliminar Proyecto

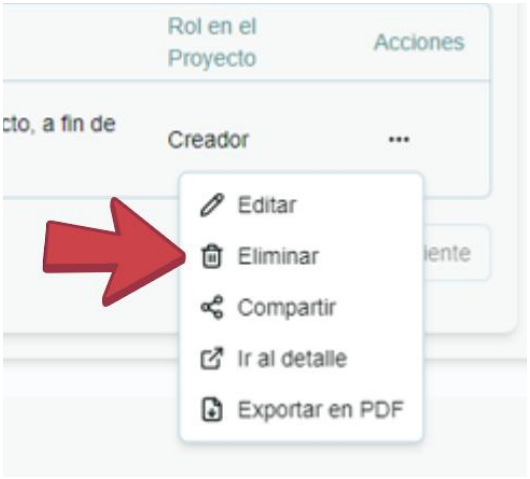


Figura 32: Opción Eliminar Proyecto

Una vez pulsada la opción eliminar, se abre un modal de confirmación para validar si realmente la acción quiere ser ejecutada.

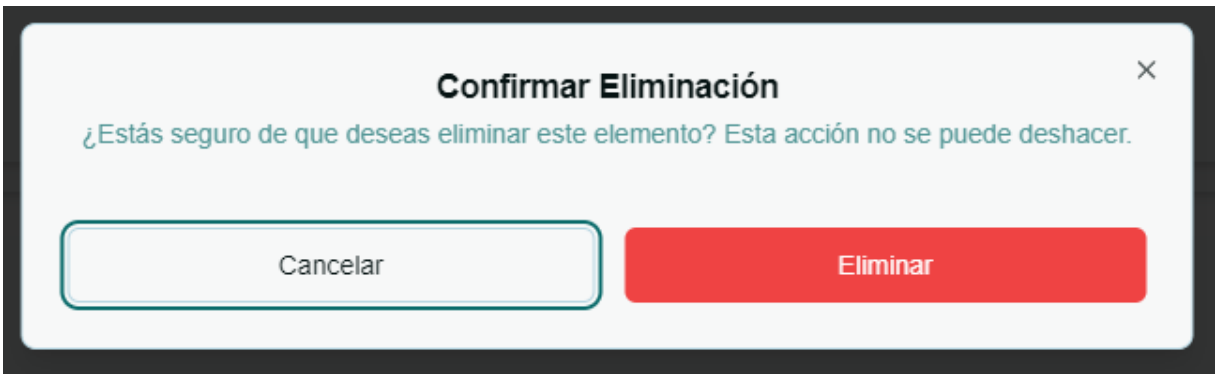


Figura 33: Modal Eliminar Proyecto

Debe pulsar el botón eliminar para que el proyecto se elimine de la lista de proyectos del usuario

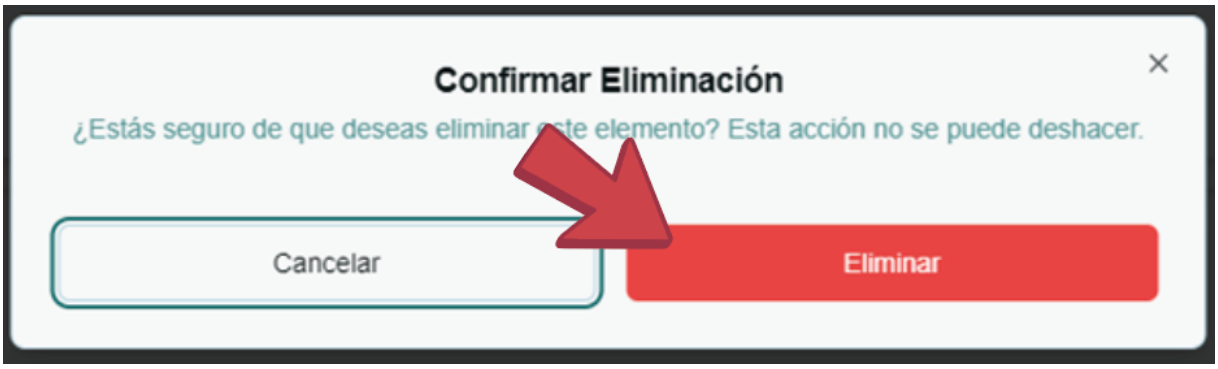


Figura 32: Opción Eliminar Proyecto

Ir al Detalle del Proyecto

Para ingresar a ver el detalle del proyecto se ingresa a las opciones del proyecto.

Figura 33: Opciones de proyecto

Se pulsa la opción de “Ir al detalle” del proyecto

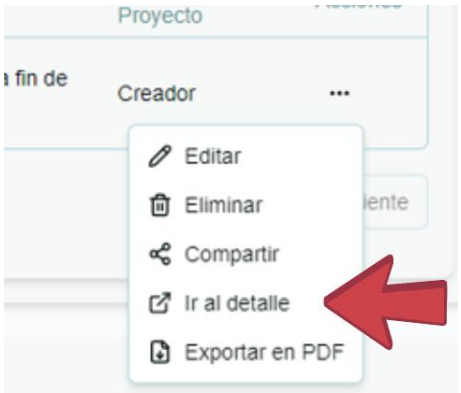


Figura 34: Opción de Ir al detalle del proyecto

Luego se abre la sección de los casos de uso, en la cual se muestra la lista de casos de uso.

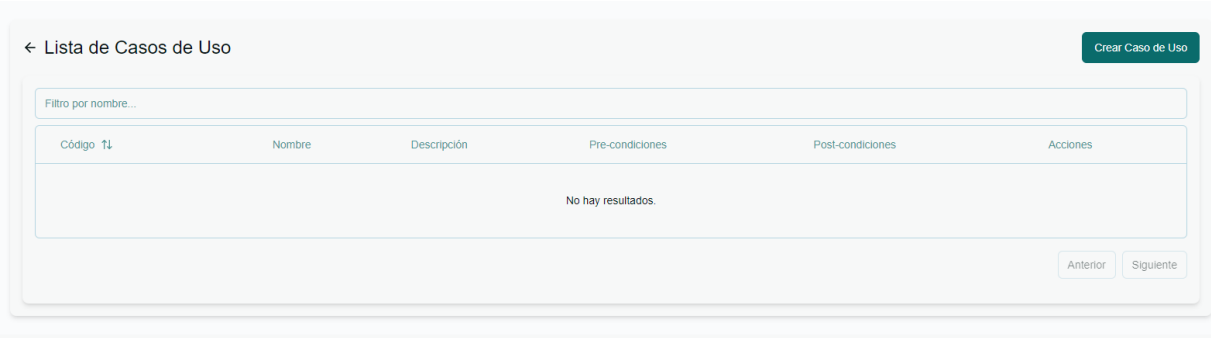


Figura 35: Lista de Casos de Uso de un proyecto

Gestión de Casos de Uso

Crear Caso de Uso

Para crear un caso de uso se debe pulsar el botón de “Crear caso de uso”

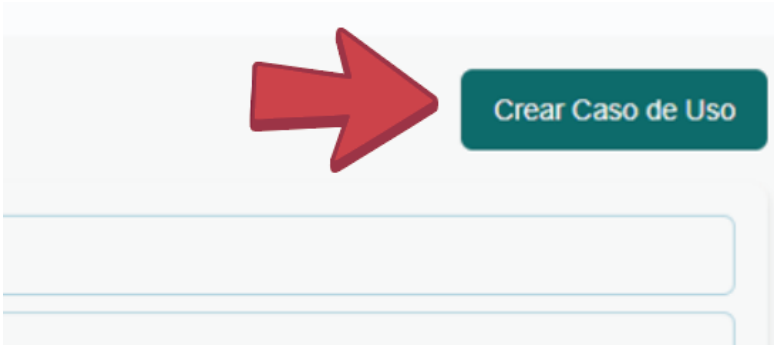


Figura 36: Botón para Crear Caso de Uso

Luego de presionarlo se abre el modal de Crear Caso de Uso

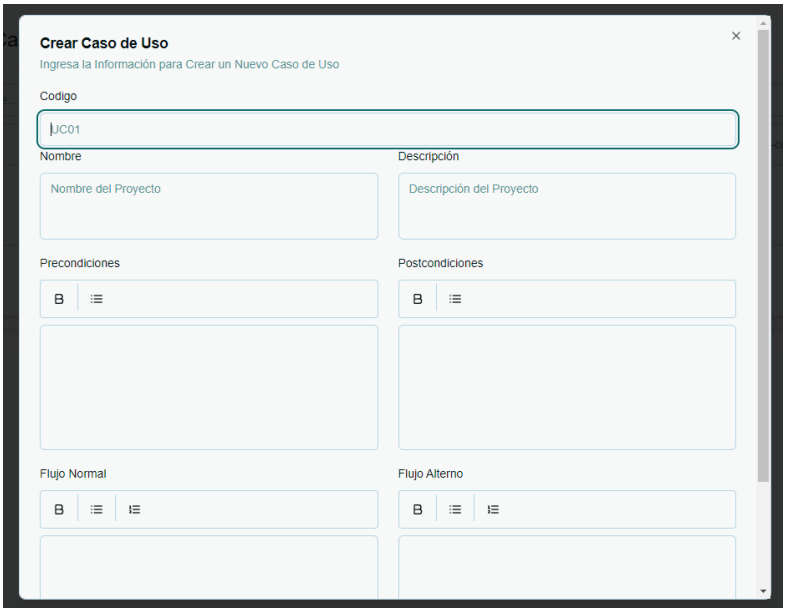


Figura 37: Modal para Crear Caso de Uso

Se llena los campos solicitados con información que corresponda al proyecto que requiera crear

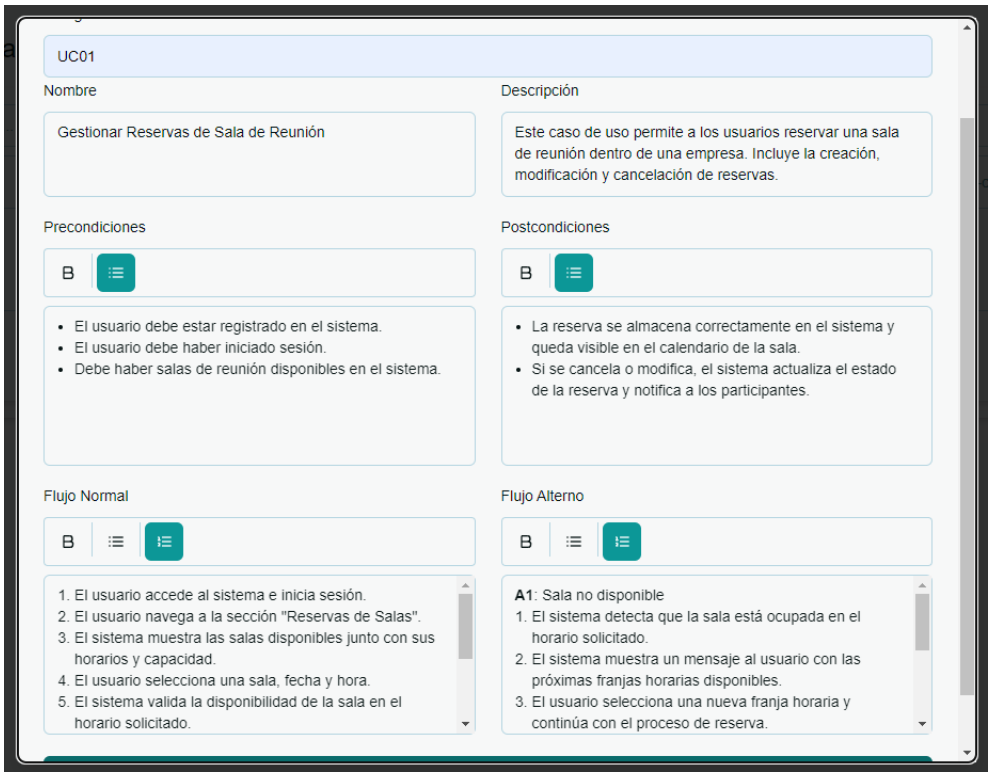


Figura 38: Modal para Crear Caso de Uso

Luego se debe pulsar el botón de Crear ubicado en la parte inferior del Modal

Figura 39: Botón Crear para Guardar Caso de Uso

Se mostrará el caso de uso en el listado inicial de casos de uso del proyecto

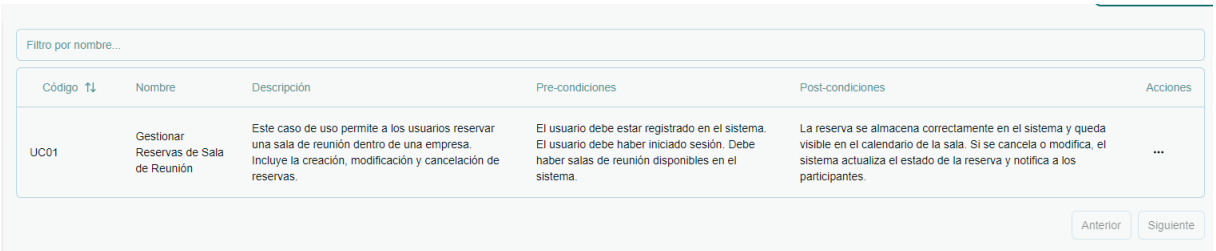


Figura 40: Lista de Casos de Uso

Editar Caso de Uso

Para editar el caso de uso, se debe tener un caso de uso creado y acceder a las opciones del proyecto.

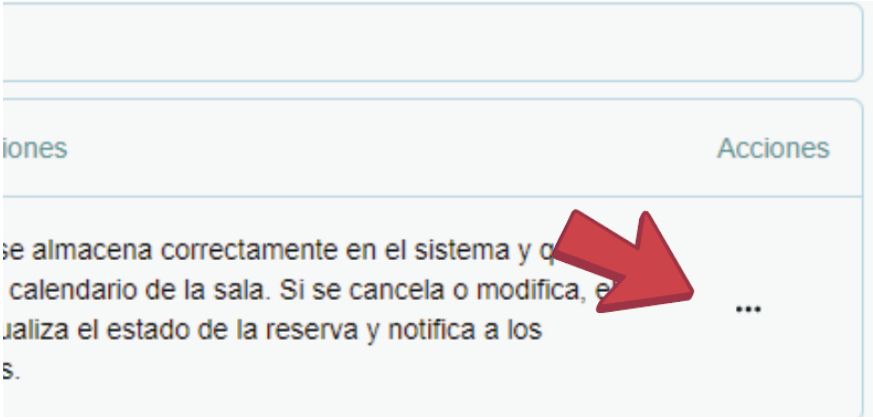


Figura 41: Opciones de Casos de Uso

Para luego escoger la opción de Editar Caso de Uso

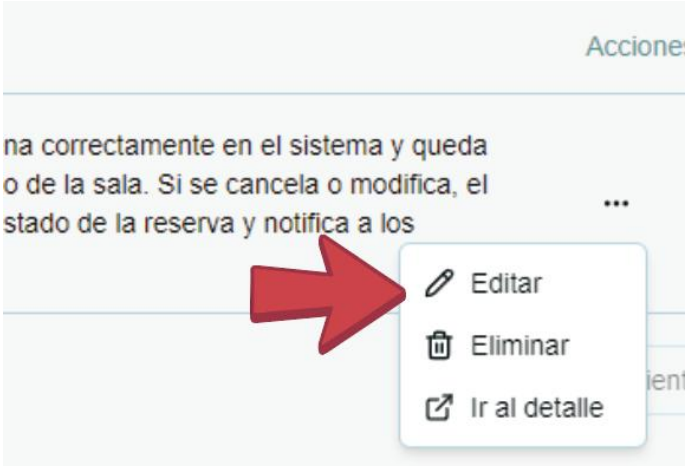


Figura 42: Opción Editar de Caso de Uso

Se abre el modal de edición del caso de uso

Figura 43: Modal Editar de Caso de Uso

Aquí se hacen los cambios que deba hacer y se pulsa el botón de actualizar

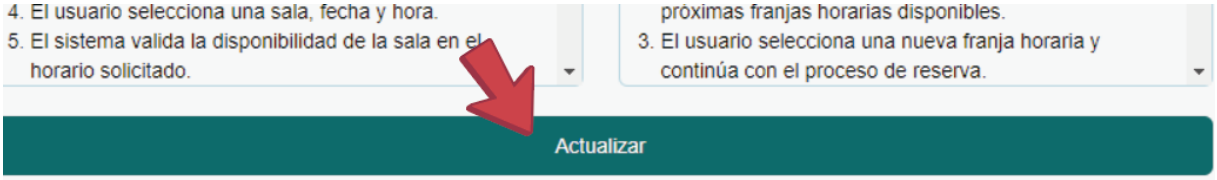


Figura 44: Botón Actualizar Caso de Uso

Eliminar Caso de Uso

Para eliminar el caso de uso, se debe tener un caso de uso creado y acceder a las opciones del proyecto.

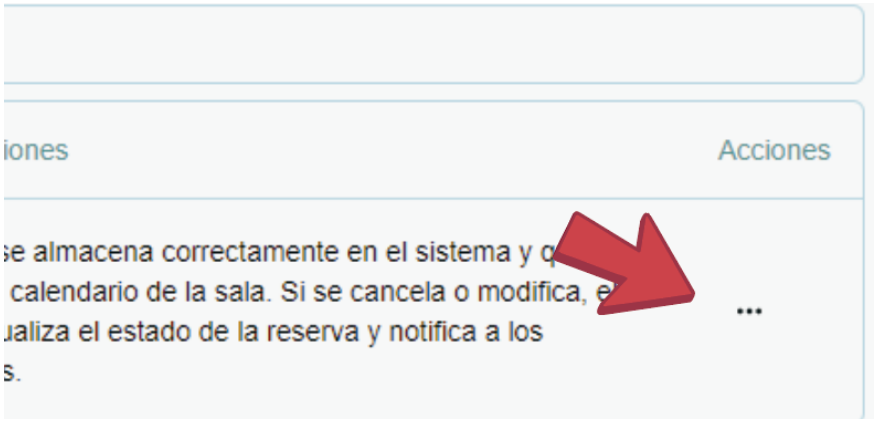


Figura 45: Opciones de Casos de Uso

Para luego escoger la opción de Editar Caso de Uso

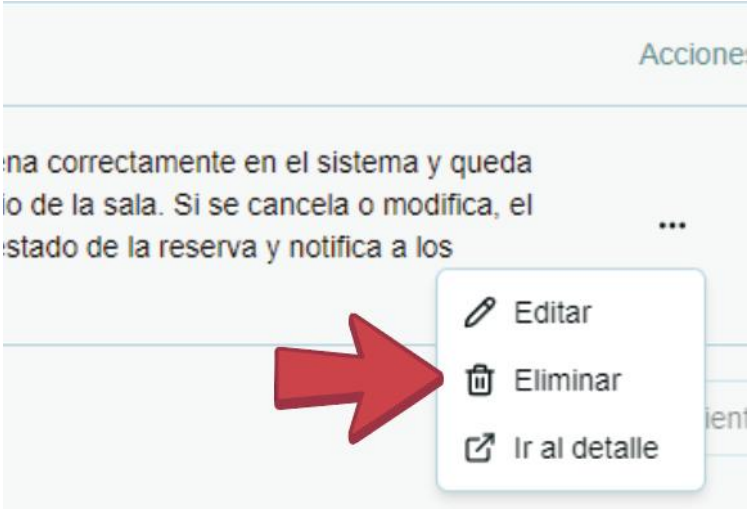


Figura 46: Opción Eliminar Caso de Uso

Una vez pulsada la opción eliminar, se abre un modal de confirmación para validar si realmente la acción quiere ser ejecutada.

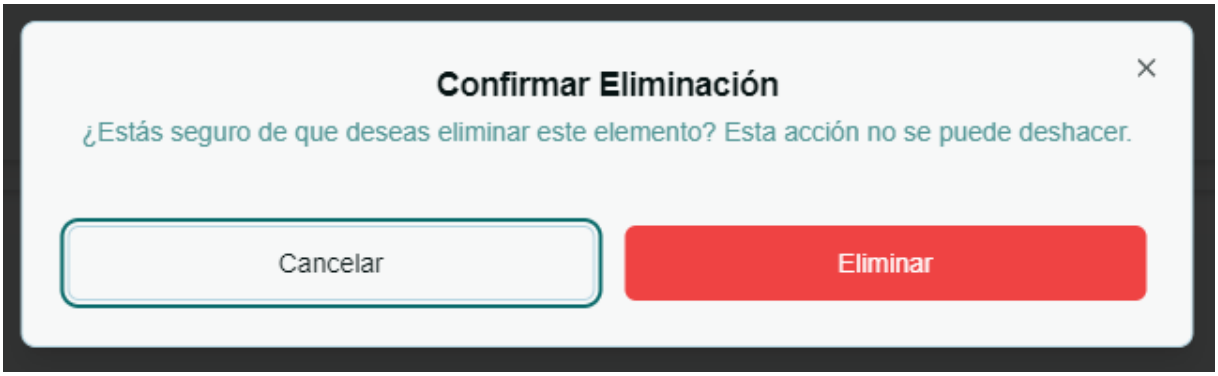


Figura 47: Modal Eliminar Caso de Uso

Debe pulsar el botón eliminar para que el caso de uso se elimine de la lista de casos de uso del proyecto

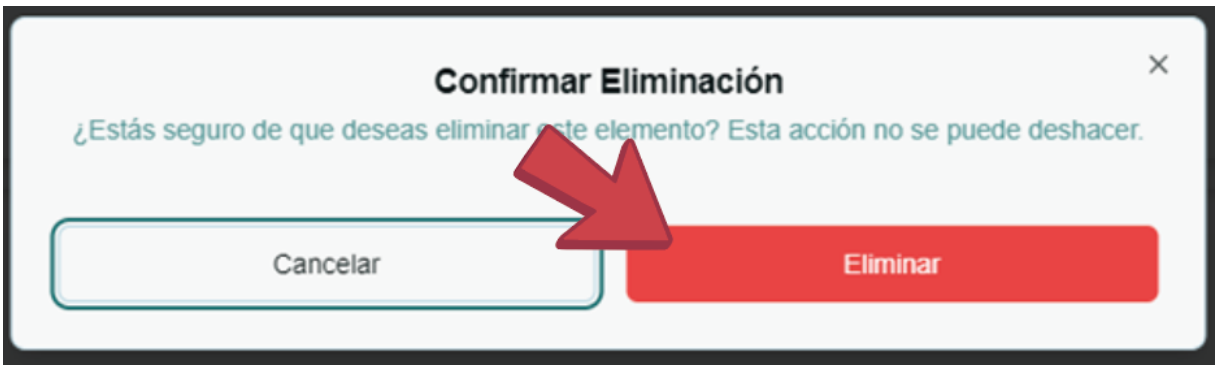


Figura 48: Opción Eliminar Caso de Uso

Ir al detalle del caso de uso

Para ingresar a ver el detalle del caso de uso se ingresa a las opciones del proyecto.

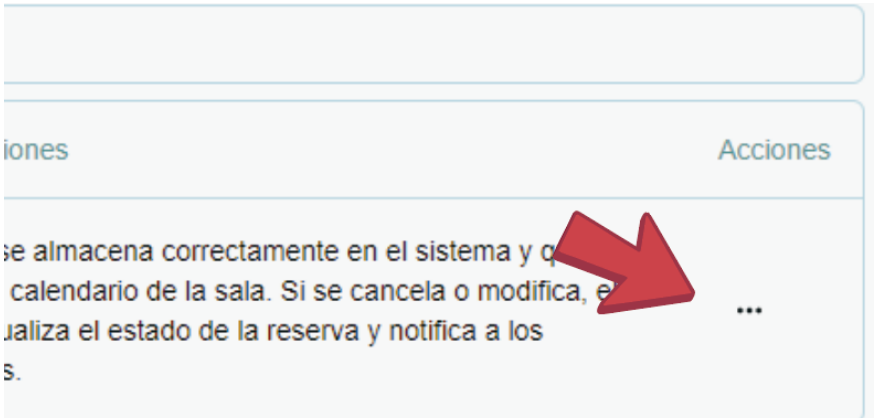


Figura 49: Opciones de Casos de Uso

Se pulsa la opción “Ir al detalle” del caso de uso

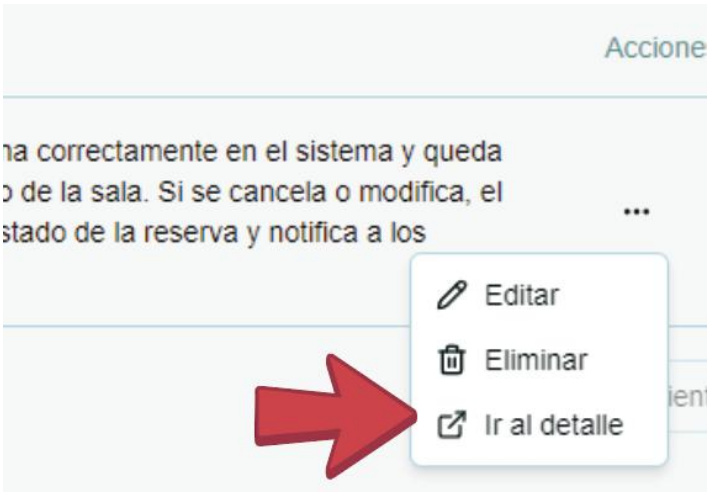


Figura 50: Opción de Ir al detalle del caso de uso

Luego se abre la sección de los casos de prueba funcionales, en la cual se muestra la lista de casos de prueba funcionales

Figura 51: Lista de Casos de Prueba Funcionales de un Caso de Uso

Gestión de Casos de Prueba Funcionales

Generar casos de prueba funcionales

Para generar casos de prueba funcionales a partir de un caso de uso, se debe estar ubicado en la lista de casos de prueba funcionales de un caso de uso.

Aquí se debe pulsar el botón de “Generar casos de prueba funcionales”

Figura 52: Botón Generar Casos de prueba funcionales

Luego se despliega un modal para generar casos de prueba funcionales, en este debe pulsar el botón “Generar casos de prueba”.

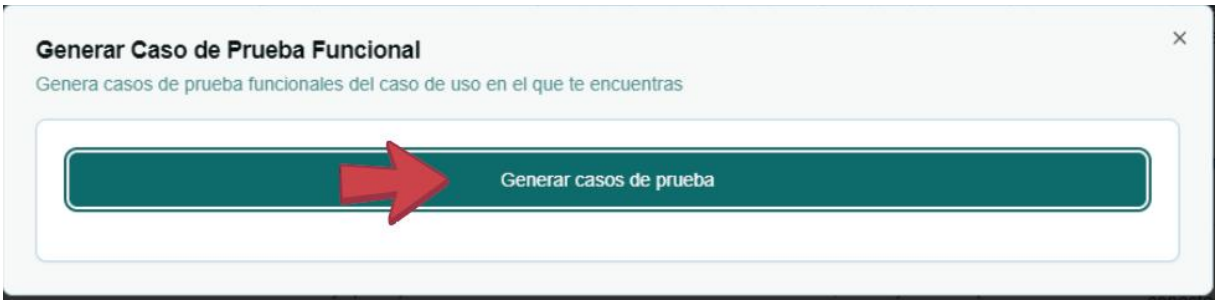


Figura 53: Botón Generar Casos de prueba funcionales

Aquí se empezarán a generar los casos de prueba funcionales, para ello se debe esperar la respuesta de la IA entre 5 a 10 segundos por una respuesta, se mostrará una pantalla de carga mientras espera.

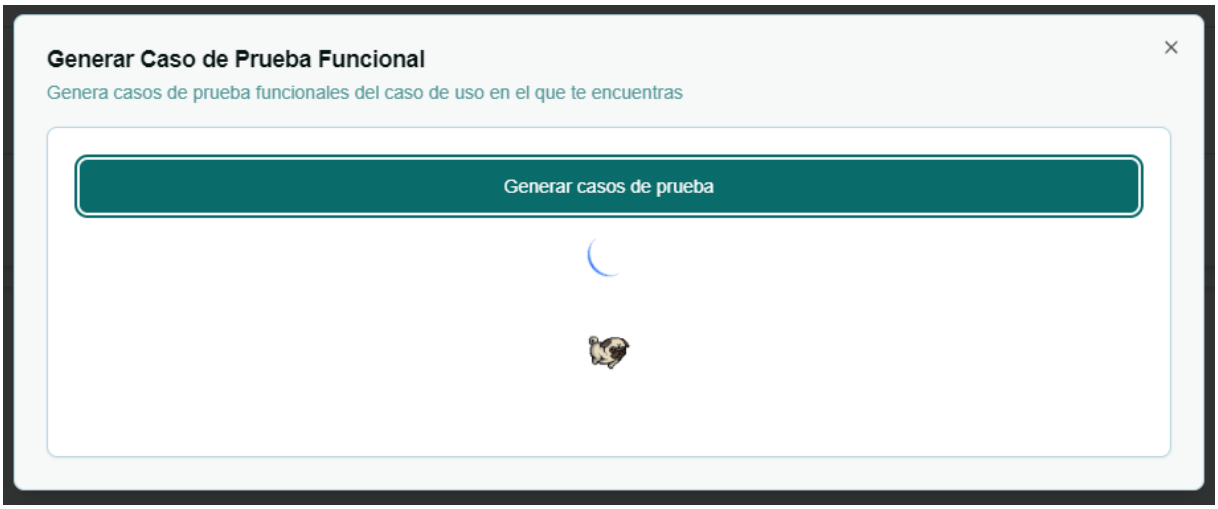


Figura 54: Pantalla de carga para generar casos de prueba funcionales

Luego de la espera se mostrarán N opciones de casos de prueba que usted podrá visualizar y revisar para saber si son aquellos los más adecuados.

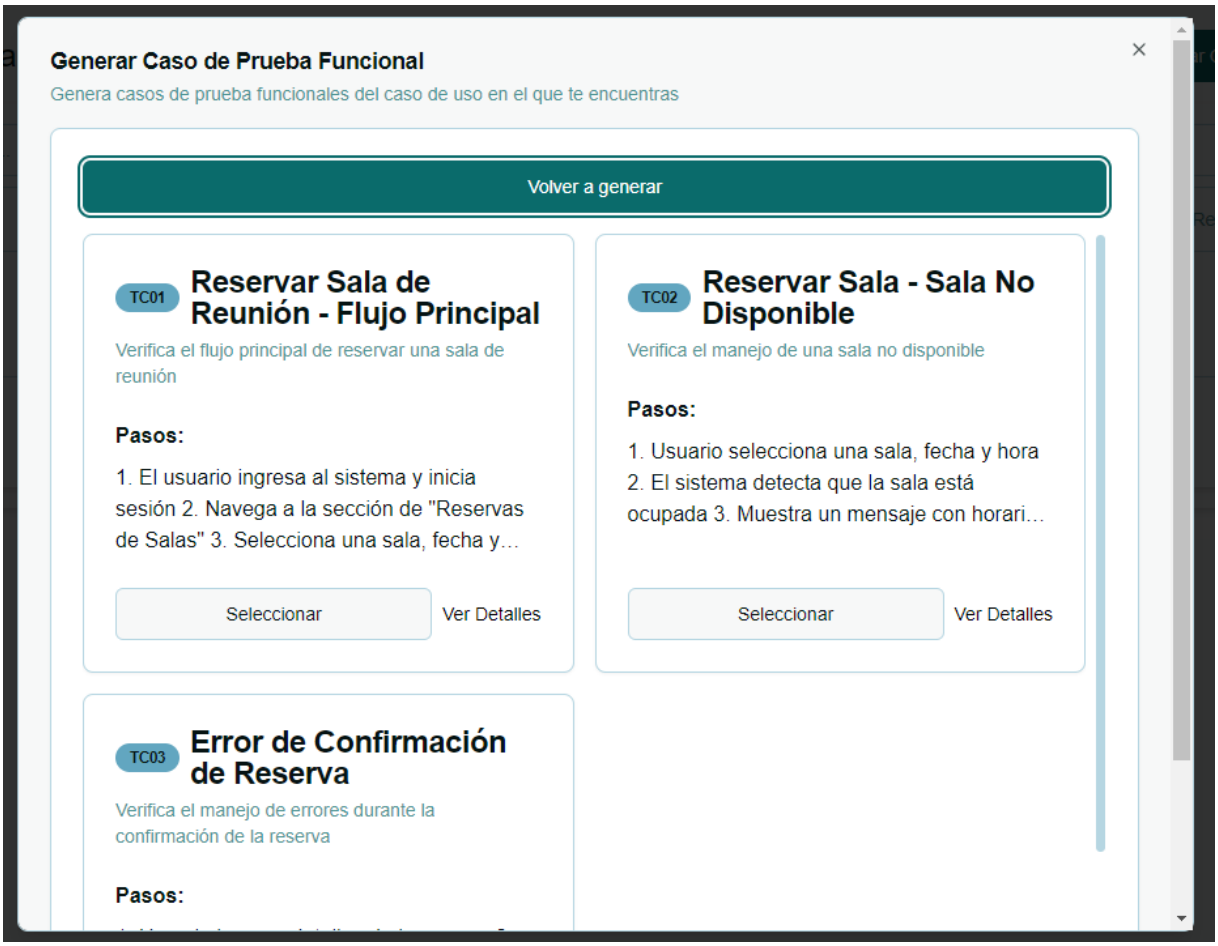


Figura 55: Casos de prueba funcionales generados

Si no existe alguno que requiera, puede enviar a generar nuevamente.

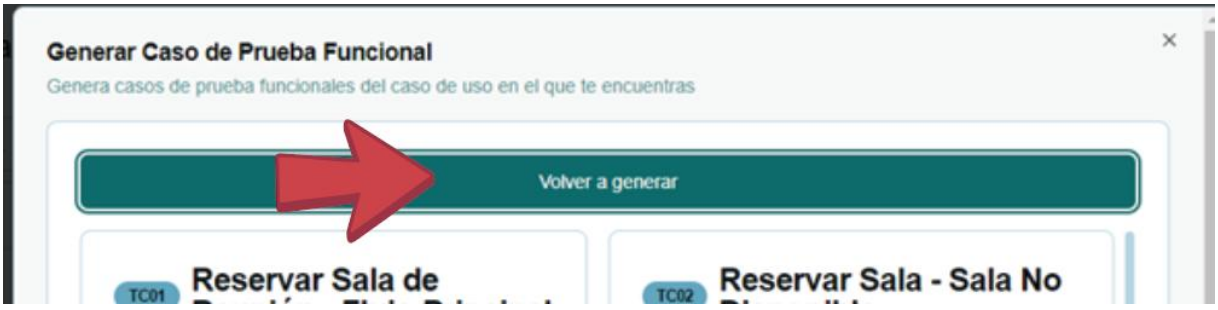


Figura 56: Pantalla de carga para generar casos de prueba funcionales

Puede acceder al detalle de cada caso de prueba funcional y ver sus especificaciones, así se comprueba cada parámetro si es adecuado o no para el contexto que se plantea, para ello se pulsa en el link ubicado a la derecha de cada botón de seleccionar de caso de prueba funcional.

Figura 57: Detalle de Caso de prueba funcional.

Ahora para almacenarlos se debe pulsar en “Seleccionar” en el caso de prueba funcional que requiera y este se pintara de color cambiando a “Seleccionado”

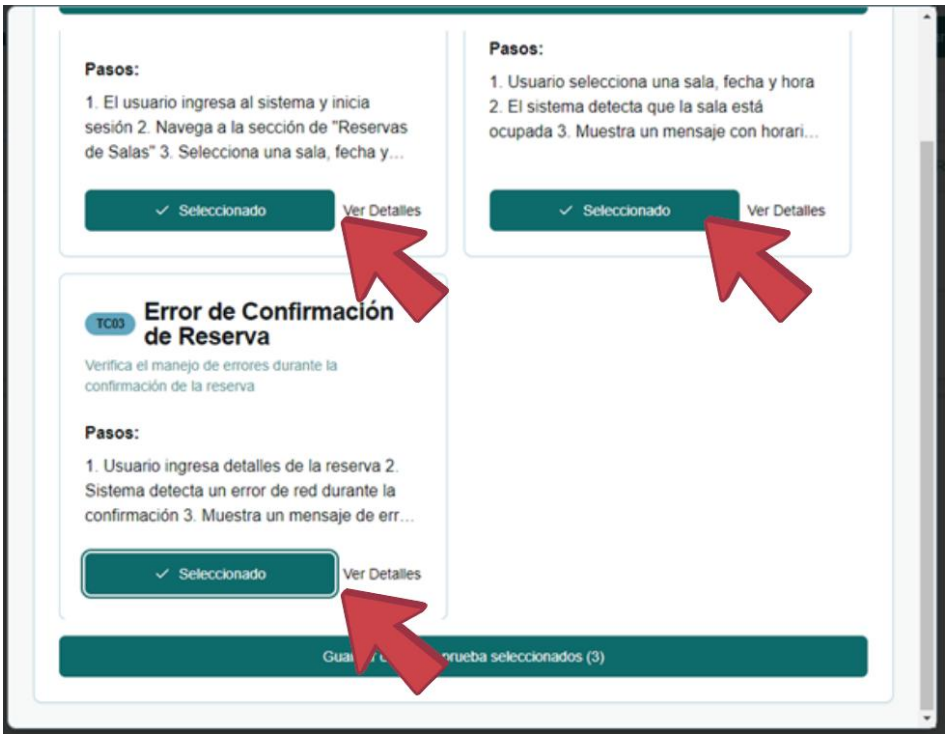


Figura 58: Casos de prueba funcionales seleccionados

Luego se pulsa Guardar casos de prueba seleccionados

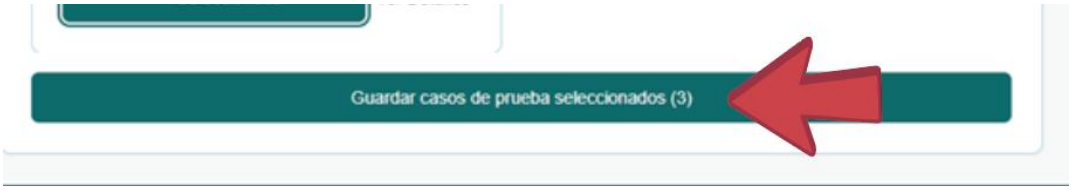


Figura 59: Guardar Casos de prueba funcionales seleccionados

A continuación, se regresa a la lista de casos de prueba funcionales

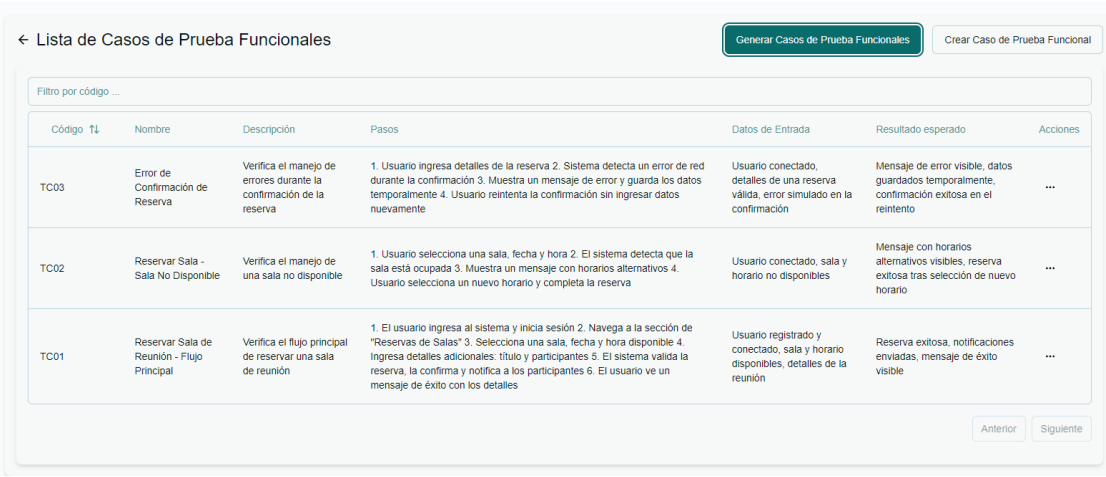


Figura 60: Lista de Casos de prueba funcionales

Editar Caso de Prueba Funcional

Para editar un caso de prueba funcional se debe poseer un caso de prueba funcional y acceder a las opciones del mismo

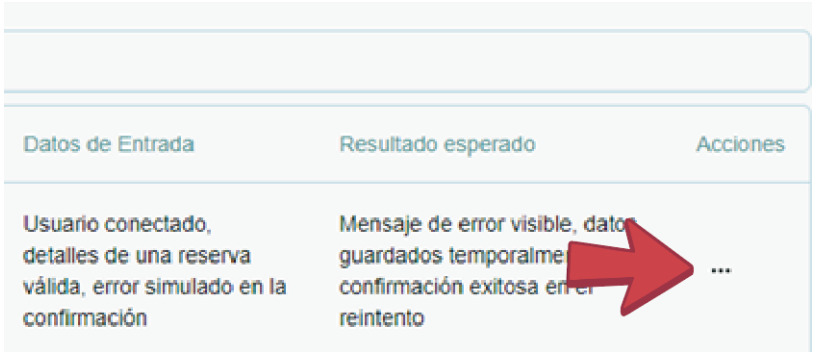


Figura 61: Opciones de caso de prueba funcional

Al desplegarse las opciones se escoge editar

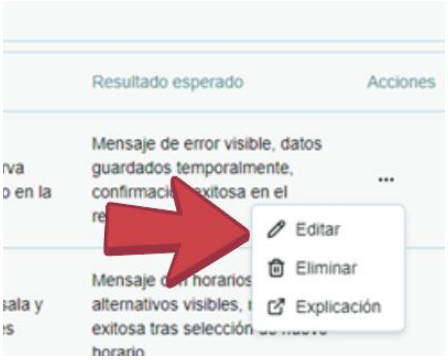


Figura 62: Opciones de caso de prueba funcional

A partir de pulsar dicha opción, se despliega el modal de edición del caso de prueba funcional, para que sus campos sean modificados

Figura 63: Modal Editar caso de prueba funcional

Luego de ello, se pulsa actualizar y los cambios aplicados al caso de prueba funcional se guardarán

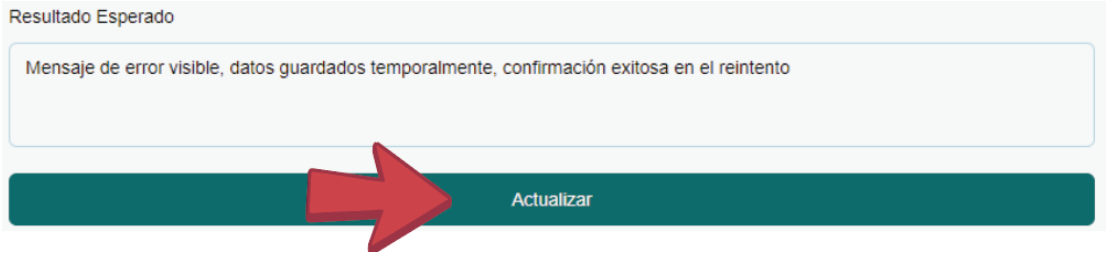


Figura 64: Botón actualizar Caso de prueba funcional

Explicación Caso de Prueba Funcional

Para visualizar la explicación de la generación de un caso de prueba funcional se debe poseer un caso de prueba funcional y acceder a las opciones del mismo, además de que este haya sido generado por la IA.

Primero se ingresa a las opciones del caso de prueba funcional y se escoge “Explicación”

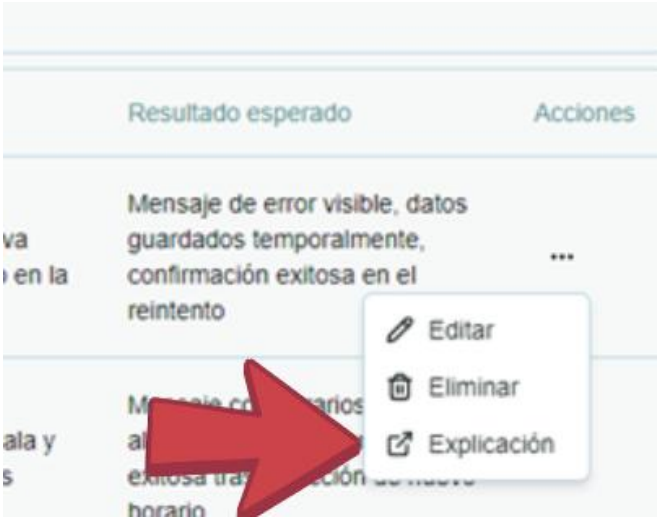


Figura 65: Opción Explicación de Caso de prueba funcional

Al pulsar la opción se abre un modal con un resumen y detalle del proceso para generar dicho caso de prueba funcional

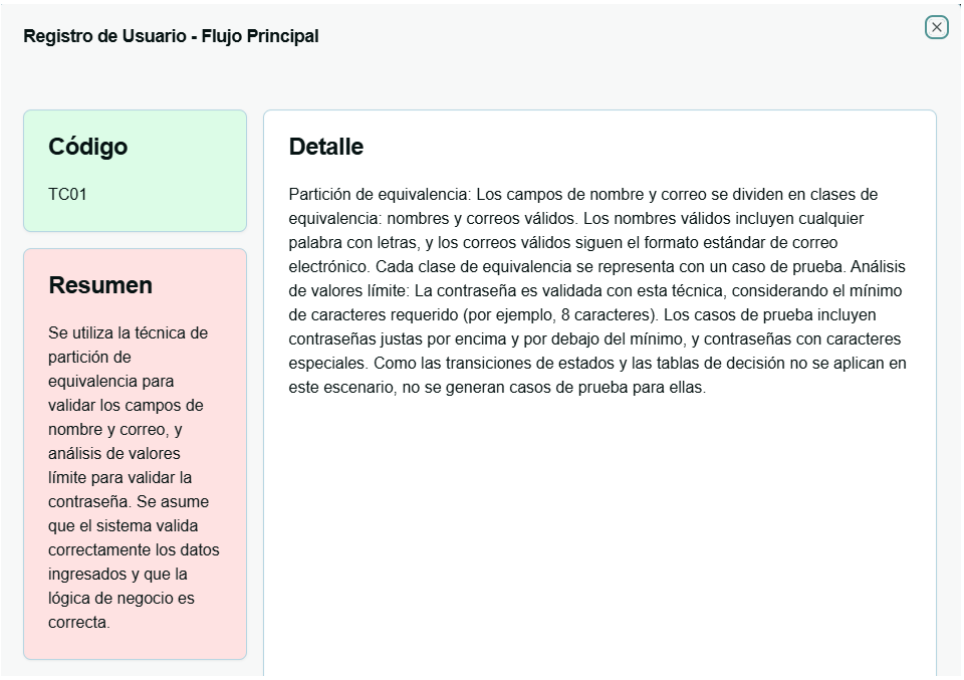


Figura 66: Explicación de Caso de prueba funcional

Eliminar Caso de Prueba Funcional

Para eliminar un caso de prueba funcional se debe poseer un caso de prueba funcional y acceder a las opciones del mismo, aquí se ingresa a las opciones del caso de prueba funcional y se escoge “Eliminar”

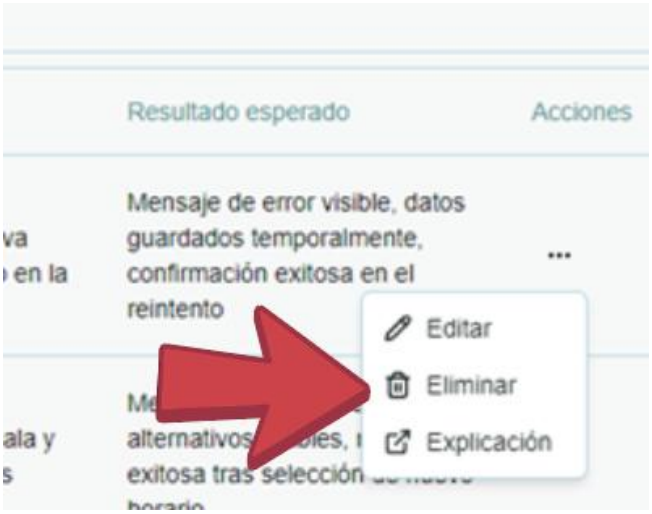


Figura 67: Eliminar Caso de prueba funcional

Al pulsar la opción de eliminar se despliega un modal para confirmar la acción

Figura 68: Modal de Eliminar Caso de prueba funcional

Aquí se pulsa la opción eliminar, para que el caso de prueba funcional se elimine.

Figura 69: Modal de Eliminar Caso de prueba funcional

Crear Caso de Prueba Funcional

A parte de la generación de casos de prueba funcionales por IA, el usuario también puede crear uno manualmente, para ello se selecciona la opción “Crear caso de prueba funcional”

Figura 70: Botón crear Caso de prueba funcional

A partir de ello, se despliega un Modal para crear caso de prueba funcionales

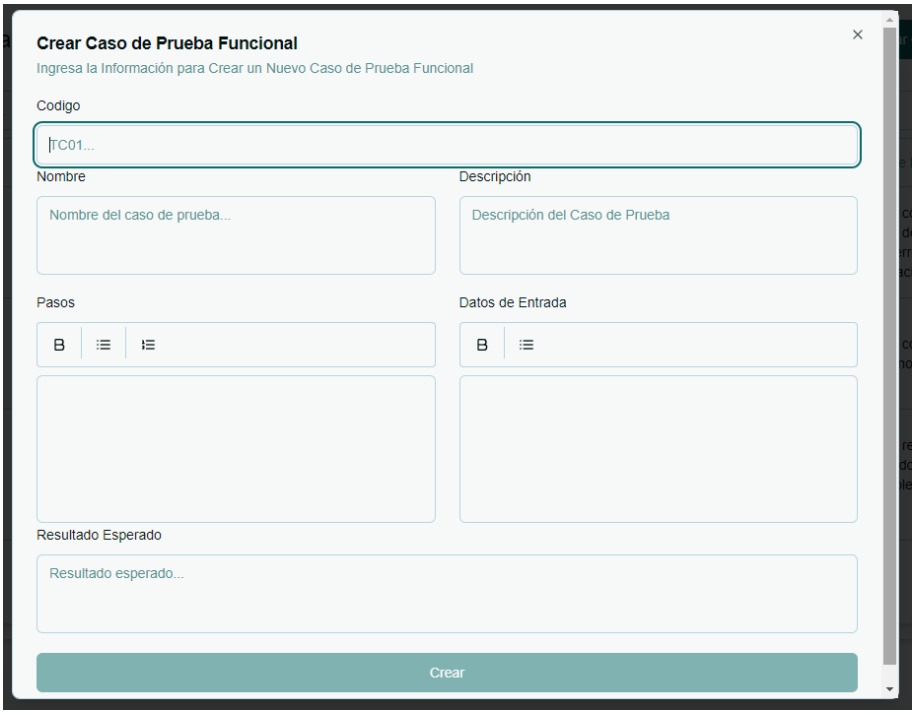


Figura 71: Modal de Crear Caso de prueba funcional

Se llena los campos solicitados con la información requerida.

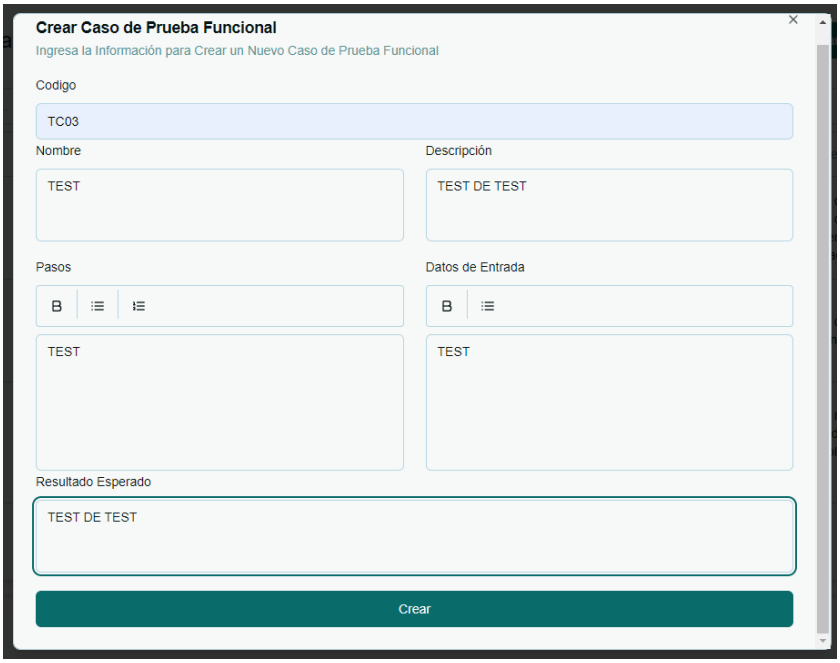


Figura 72: Modal de Crear Caso de prueba funcional

Luego se pulsa el botón de crear para que se guarde el caso de prueba funcional.

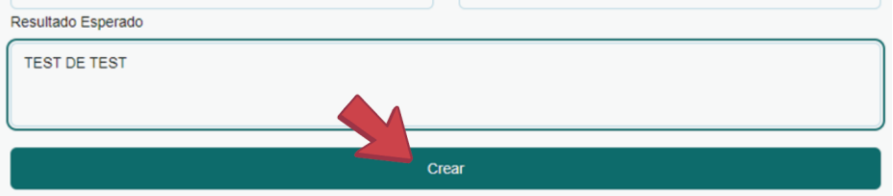


Figura 73: Botón para guardar Caso de prueba funcional

Se almacena en la lista de casos de prueba funcionales del caso de uso

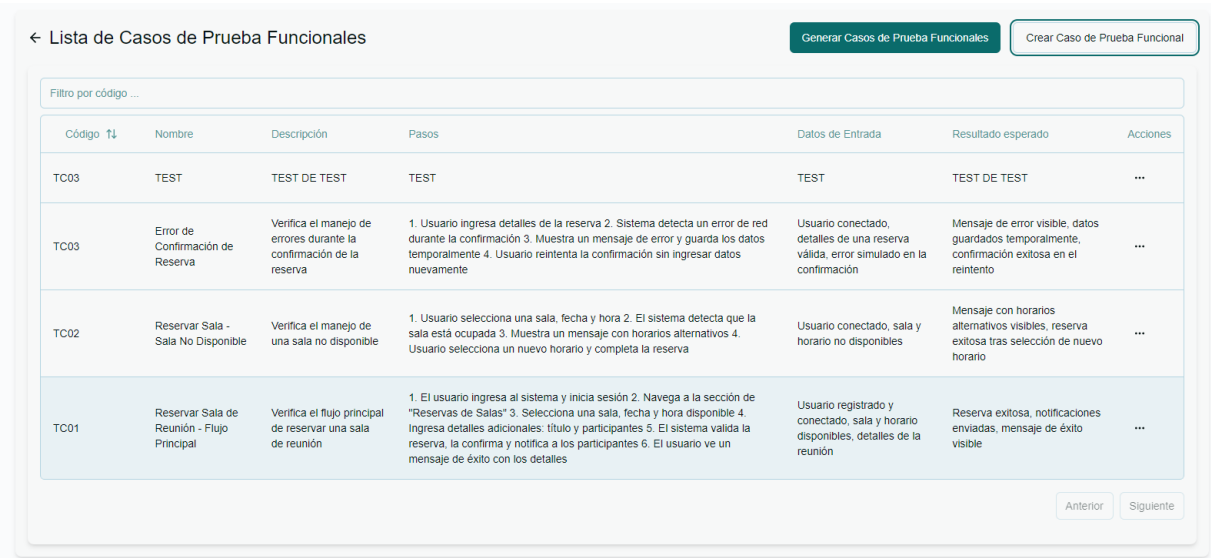


Figura 74: Lista de Casos de prueba funcionales.

Exportar PDF de Proyecto

Cuando se llena toda la información de un proyecto, casos de uso y casos de prueba se puede exportar esta información en formato pdf

Para exportar el pdf del proyecto, se debe acceder a las opciones del proyecto.

Figura 75: Opciones de proyecto

Para luego escoger la opción de Exportar en PDF del Proyecto

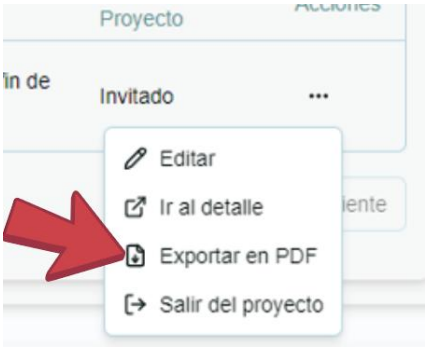


Figura 76: Opción de exportar en PDF

Aquí se despliega la visualización del PDF del proyecto, el cual puede observar, imprimirlo o guardarlo.

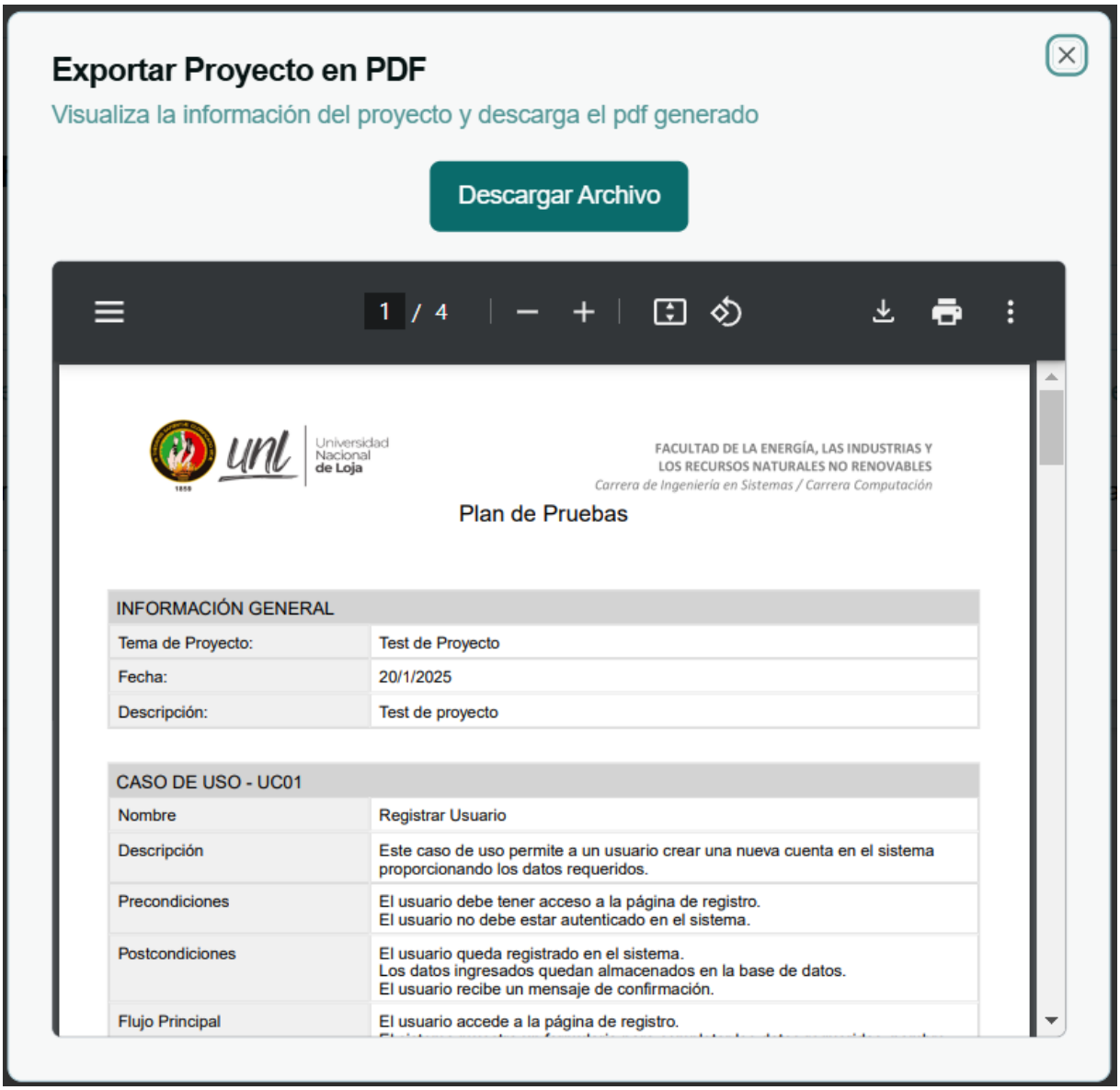


Figura 77: Modal de exportar en PDF.

Tareas rol: Administrator

Gestionar Usuarios

Para ingresar a la lista de usuarios desde el rol de Administrador se puede hacer de dos formas

- Pulsando el botón ¡Vamos Allá!
- Pulsando el item del SideBar en la parte izquierda de la pantalla que dice “Gestión de Roles”.

Figura 78: Página Inicial rol Administrador

Cuando se pulsa una de las dos opciones se redirige a la página de gestión de roles

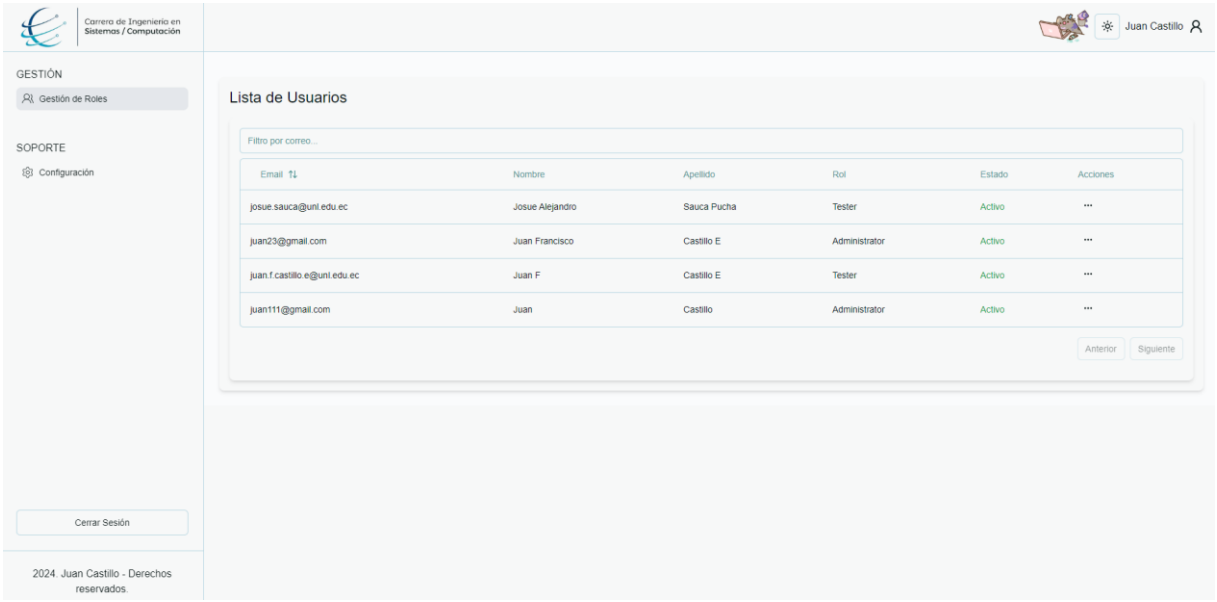


Figura 79: Página Usuarios rol Administrador

Cambiar Rol

Para cambiar rol de un usuario, se debe ir a las opciones de usuario

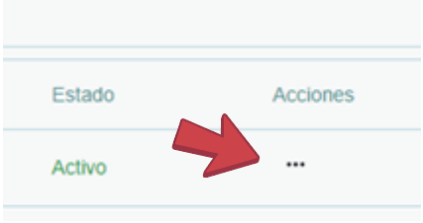


Figura 80: Opciones de Usuario

Aquí se ingresa a las opciones del usuario y se escoge “Cambiar Rol”

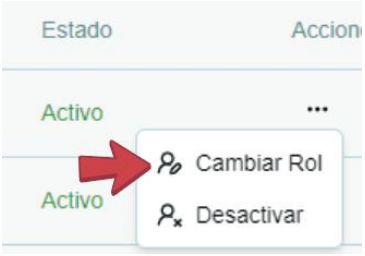


Figura 81: Opciones de Usuario

Se despliega un modal para cambiar el rol del usuario

Figura 82: Opciones de Usuario

Se pulsa guardar para actualizar el rol del usuario

Figura 83: Botón guardar rol de Usuario

Activar Usuario

El usuario debe estar desactivado para ello e ingresar a las opciones de usuario

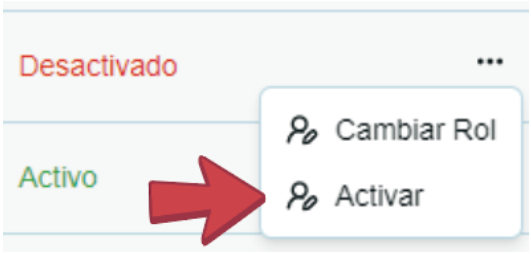


Figura 84: Botón guardar rol de Usuario

Al pulsar el botón de Activar se despliega un modal para confirmar la acción

Figura 85: Modal activar Usuario

Para confirmar la acción se pulsa Activar y el estado del usuario cambia a activo

Figura 86: Modal activar Usuario

Desactivar Usuario

El usuario debe estar activado para ello e ingresar a las opciones de usuario

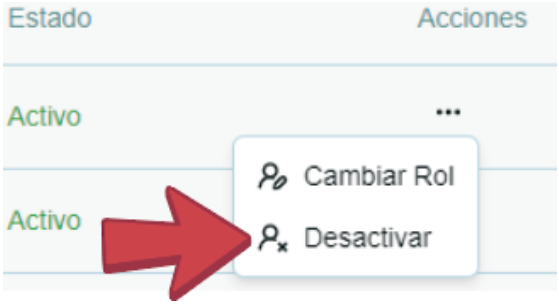


Figura 87: Opción desactivar Usuario

Al pulsar el botón de Desactivar se despliega un modal para confirmar la acción

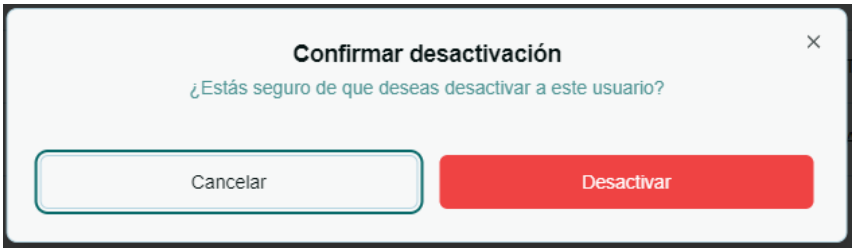


Figura 88: Modal desactivar Usuario

Para confirmar la acción se pulsa Desactivar y el estado del usuario cambia a desactivado

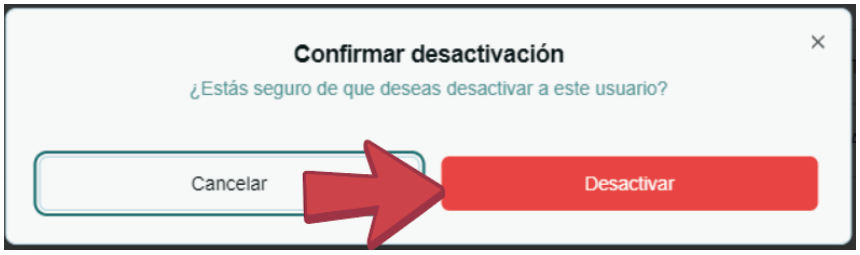


Figura 89: Modal desactivar Usuario

Información

Para ingresar a ver información sobre la aplicación y contextualización sobre el ámbito del mismo, se debe pulsar en el Sidebar de la izquierda que posee la opción “Información”

Figura 90: Información de la Aplicación

Configuración

La configuración radica en el perfil del usuario, para acceder a ella se debe pulsar en el Sidebar de la izquierda que posee la opción “Configuración”

Existen diferentes páginas para modificar el perfil de acuerdo al rol

Rol Tester:

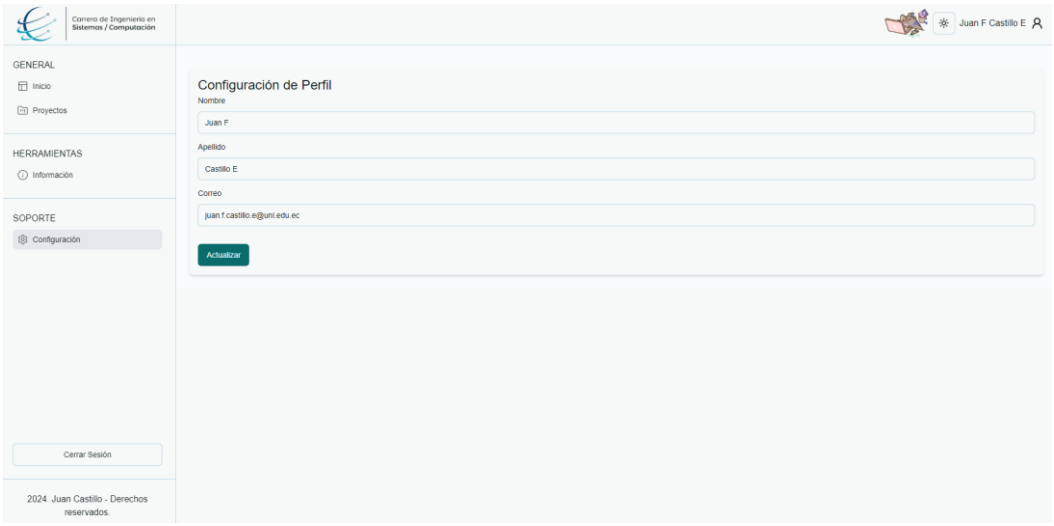


Figura 91: Configuración de Perfil – Vista Tester

Rol Administrador:

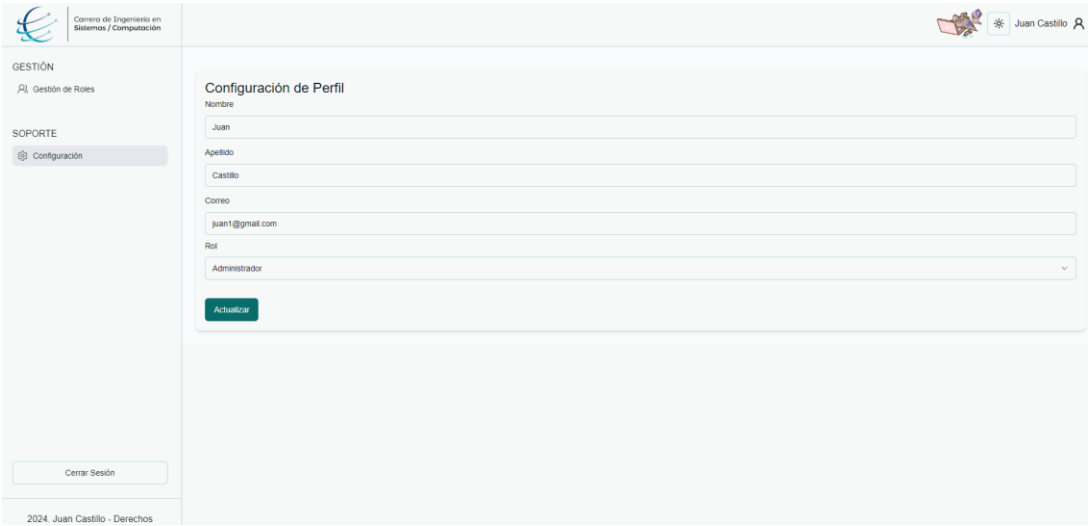


Figura 92: Configuración de Perfil – Vista Administrador

En los dos roles se puede actualizar el perfil, modificando los campos solicitados, sin embargo, en el rol de administrador puede modificar su propio rol.

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 12. Pruebas de Percepción de utilidad TAM

[Pruebas de Percepción de utilidad TAM]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Juan Francisco Castillo Estrella

Revisado y aprobado por:
Ing. Wilman Chamba

2024

Historial de Cambios

Versión	Fecha	Responsable
1.0	25/10/2024	Juan Francisco Castillo Estrella

RESULTADOS DE LA EVALUACIÓN DE LA PERCEPCIÓN DE UTILIDAD MEDIANTE TAM

Datos de la evaluación:

- **Población:** Estudiantes de octavo y noveno ciclo del itinerario de Ingeniería de Software de computación de la carrera de computación.
- **Fecha:** 12/02/2024.
- **Producto probado:** Aplicación web para el diseño de casos de prueba funcionales a partir de casos de uso para la carrera de Computación de la Universidad Nacional de Loja.

Preguntas:

Tabla 1: Cuestionarios de variable de percepción de utilidad del método TAM

Identificador	Pregunta
PU1	¿El sistema reduce significativamente el tiempo necesario para crear casos de prueba funcionales?
PU2	¿El sistema me ayuda a identificar escenarios de prueba que podría haber pasado por alto manualmente?
PU3	¿Los casos de prueba generados son relevantes para los casos de uso proporcionados?
PU4	¿La calidad de los casos de prueba generados es comparable o superior a los creados manualmente?
PU5	¿El sistema proporciona una cobertura adecuada de los casos de uso?
PU6	¿El uso de este sistema mejora la calidad de mi trabajo en Testing?
PU7	¿Esta herramienta me ayuda a realizar mi trabajo de manera más efectiva?
PU8	¿El sistema se integra bien en mi flujo de trabajo actual de pruebas?
PU9	¿Recomendaría este sistema a otros profesionales del Testing?
PU10	¿Es útil la generación automática de casos de prueba funcionales a partir de casos de uso?
PU11	¿Es útil el formato y estructura de los casos de prueba funcionales generados?
PU12	¿Es útil la personalización de los casos de prueba funcionales generados?
PU13	¿En qué porcentaje estima que el sistema ha reducido el tiempo de creación de casos de prueba? 0-20%, 21-40%, 41-60%, 61-80%, 81-100%
PU14	¿Qué porcentaje de los casos de prueba generados ha requerido de modificación manual? 0-20%, 21-40%, 41-60%, 61-80%, 81-100%

Tabulación de los resultados mediante la escala de Likert

Tabla 2: Cuestionarios de variable de percepción de utilidad del método TAM

Indicador	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Totalmente de acuerdo
P1	0%	0%	4.8%	33.3%	61.9%
P2	0%	0%	4.8%	47.6%	47.6%
P3	0%	0%	0%	61.9%	38.1%
P4	0%	0%	9.5%	52.4%	38.1%
P5	0%	0%	4.8%	38.1%	57.1%
P6	0%	0%	9.5%	28.6%	61.9%
P7	0%	0%	0%	33.3%	66.7%
P8	0%	0%	9.5%	28.6%	61.9%
P9	0%	0%	0%	42.9%	57.1%
P10	0%	0%	0%	19%	81%
P11	0%	0%	0%	38.1%	61.9%
P12	0%	0%	0%	23.8%	76.2%

Tabla 3: Cuestionarios de variable de percepción de utilidad del método TAM

Indicador	0-20%	21-40%	41-60%	61-80%	81-100%
P13	0%	4.8%	14.3%	52.4%	28.6%
P14	52.4%	19%	9.5%	14.3%	4.8%

- **Elaboración de escala de Likert**

Este cuestionario fue evaluado con la escala de Likert la cual describe cinco niveles de respuestas, la **Tabla 4** presenta la escala a evaluar de la pregunta 1 a la 12 debido a su respuesta cualitativa que va desde Totalmente en desacuerdo a Totalmente de acuerdo, lo cual plantea un valor cuantitativo en esta escala de 0 a 5.

Tabla 4: Escala de Likert

Respuesta	Valor
Totalmente en desacuerdo	0 – Nada
En desacuerdo	1 – Bajo
Ni de acuerdo ni en desacuerdo	2 – Normal
De acuerdo	3 – Medio
Totalmente de acuerdo	4 – Alto

En la **Tabla 5,** se presenta una escala Likert ponderada por porcentaje para evaluar la pregunta 13 y 14, esto debido al tipo de respuesta que reflejan dichas preguntas, planteando un mapeo de escala Likert a un rango porcentual.

Tabla 5: Escala de Likert

Respuesta		**Valor**
Pregunta 13	**Pregunta 14**	
0-20%	81-100%	0 – Nada
21-40%	61-80%	1 – Bajo
41-60%	41-60%	2 – Normal
61-80%	21-40%	3 – Medio
81-100%	0-20%	4 – Alto

Análisis de Resultados.

En la **Figura 1**, se muestra el ítem PU1, el 61.9% de los encuestados están totalmente de acuerdo, y el 33.3% están de acuerdo en que el sistema reduce significativamente el tiempo necesario para crear casos de prueba funcionales. Esto demuestra que más del 90% de los usuarios perciben una mejora en la eficiencia del proceso de creación de casos de prueba mediante el uso del sistema.

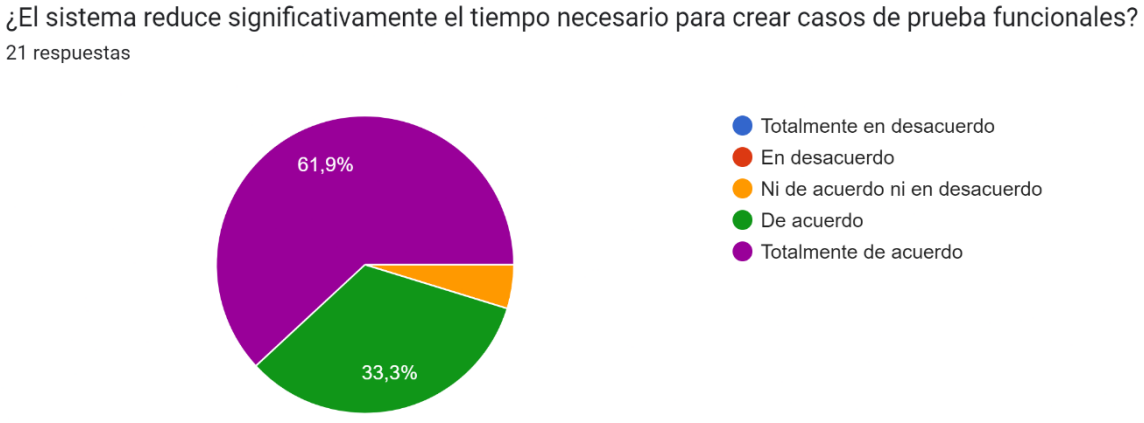


Figura 1: PU1

En la **Figura 2**, se muestra el ítem PU2, el 47.6% están totalmente de acuerdo, y otro 47.6% están de acuerdo en que el sistema ayuda a identificar escenarios de prueba que podrían haber pasado por alto manualmente. Esto sugiere que más del 95% los usuarios valoran la capacidad del sistema para mejorar la exhaustividad en la identificación de escenarios de prueba.

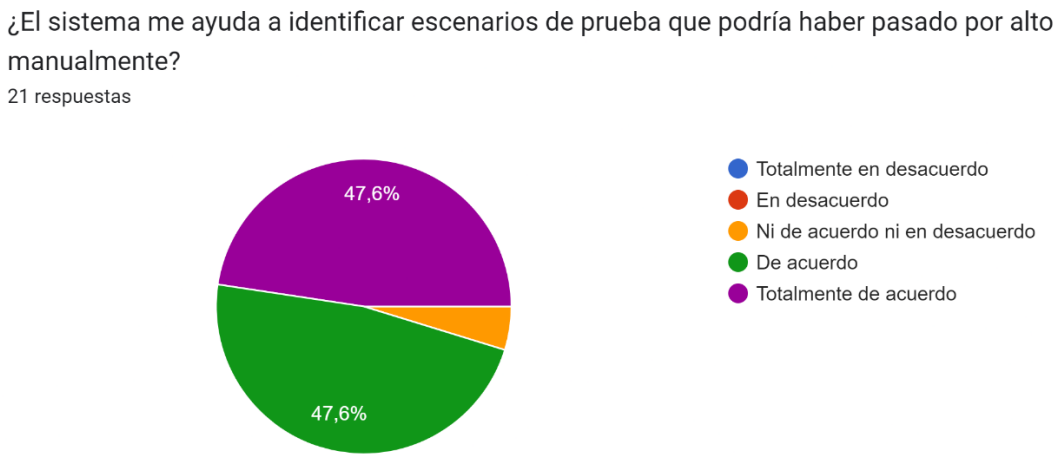


Figura 2: PU2

En la **Figura 3**, se muestra el ítem PU3, el 38.1% están totalmente de acuerdo, y el 61.9% están de acuerdo en que los casos de prueba generados son relevantes para los casos de uso proporcionados. Esto muestra que más del 90% de los usuarios consideran que los casos de prueba funcionales generados son pertinentes y alineados con los casos de uso.

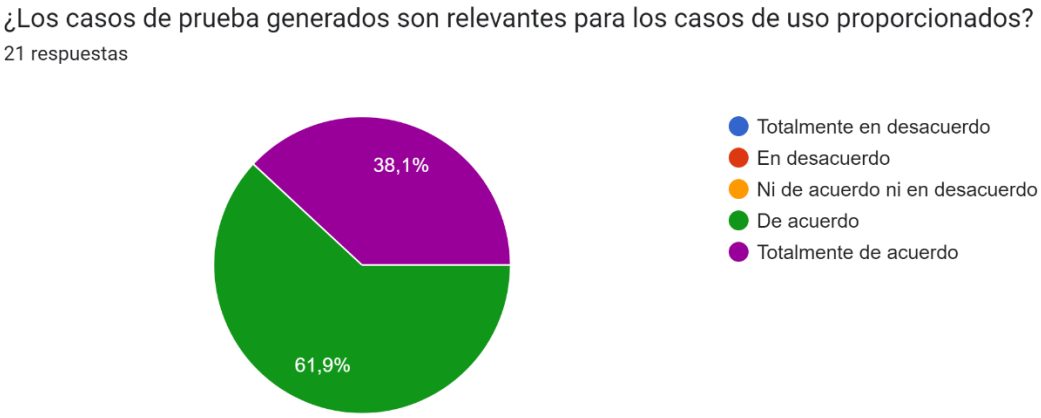


Figura 3: PU3

En la **Figura 4**, se muestra el ítem PU4, el 38.1% están totalmente de acuerdo, y el 52.4% están de acuerdo en que la calidad de los casos de prueba generados es comparable o superior a los creados manualmente. Esto indica que más del 90% de los usuarios obtuvieron una percepción positiva sobre la calidad de los casos de prueba funcionales generados automáticamente.

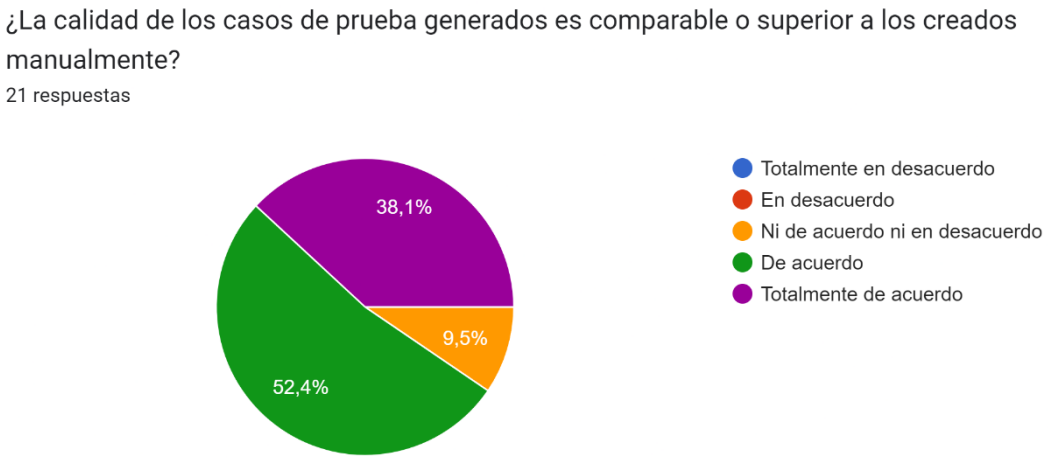


Figura 4: PU4

En la **Figura 5**, se muestra el ítem PU5, el 57.1% están totalmente de acuerdo, y el 38.1% están de acuerdo en que el sistema proporciona una cobertura adecuada de los casos de uso. Esto muestra que más del 80% de los encuestados confían en la cobertura proporcionada por el sistema.

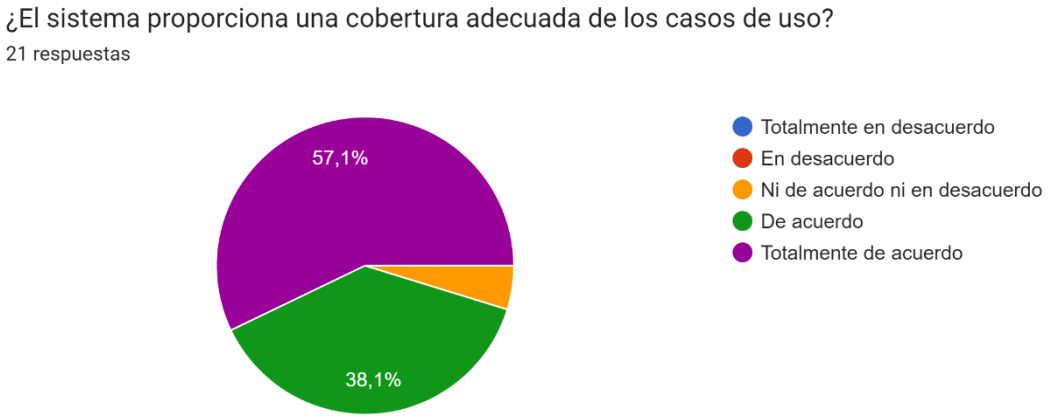


Figura 5: PU5

En la **Figura 6**, se muestra el ítem PU6, el 61.9% están totalmente de acuerdo, y el 28.6% están de acuerdo en que el uso del sistema mejora la calidad de su trabajo en Testing. Más del 80% de los usuarios perciben que el sistema contribuye positivamente a la calidad de su trabajo.

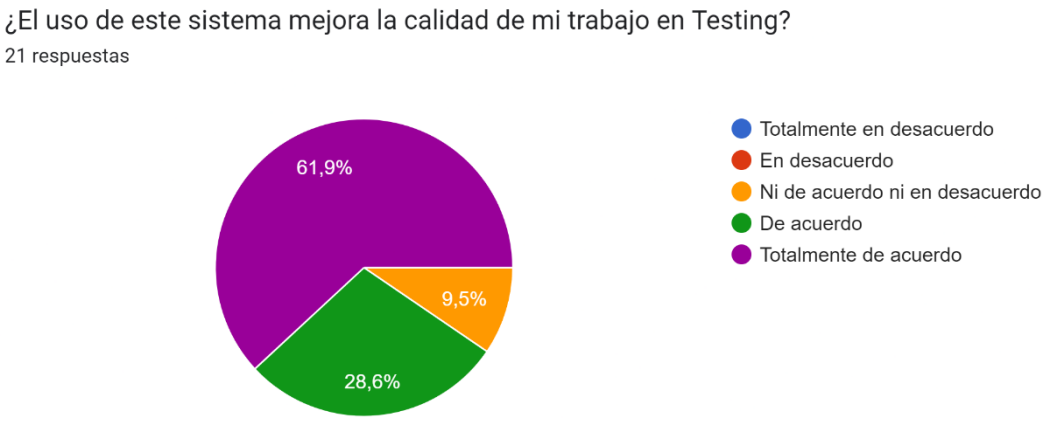


Figura 6: PU6

En la **Figura 7**, se muestra el ítem PU7, el 66.7% están totalmente de acuerdo, y el 33.3% están de acuerdo en que la herramienta les ayuda a realizar su trabajo de manera más efectiva. Esto evidencia que más del 90% los encuestados encuentran la herramienta útil para mejorar la eficacia en su trabajo.

Figura 7: PU7

En la **Figura 8,** se muestra el ítem PU8, El 61.9% están totalmente de acuerdo, y el 28.6% están de acuerdo en que el sistema se integra bien en su flujo de trabajo actual de pruebas. Esto sugiere que más del 85% de los encuestados perciben una buena adaptabilidad del sistema al flujo de trabajo existente de los usuarios.

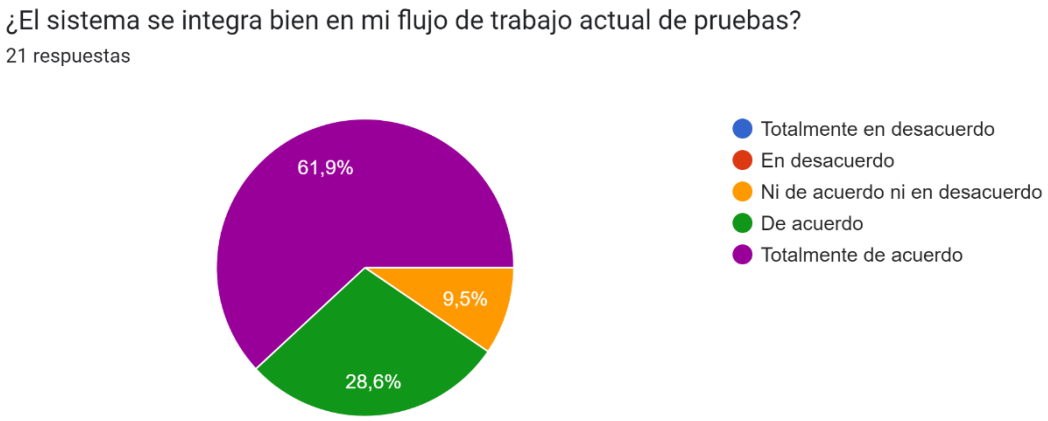


Figura 8: PU8

En la **Figura 9**, se muestra el ítem PU9, el 57.1% están totalmente de acuerdo, y el 42.9% están de acuerdo en que recomendarían el sistema a otros profesionales del Testing. La alta recomendación indica que más del 90% de los usuarios tienen una aceptación positiva y satisfacción con el sistema.

Figura 9: PU9

En la **Figura 10**, se muestra el ítem PU10, el 81% están totalmente de acuerdo, y el 19% están de acuerdo en que la generación automática de casos de prueba funcionales a partir de casos de uso es útil. Esto indica que más del 95% considera útil esta funcionalidad, destacando su importancia en el proceso de Testing.

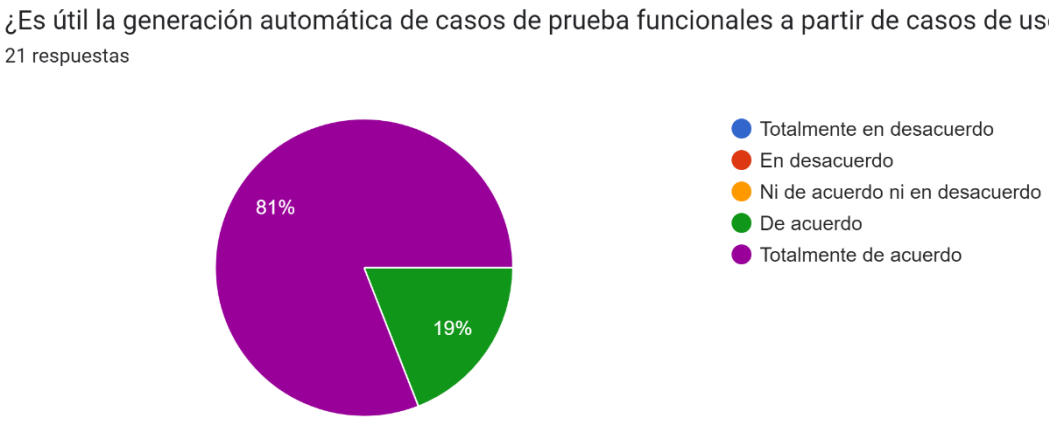


Figura 10: PU10

En la **Figura 11**, se muestra el ítem PU11, el 61.9% están totalmente de acuerdo, y el 38.1% están de acuerdo en que el formato y estructura de los casos de prueba funcionales generados son útiles. Más del 90% de los usuarios valoran positivamente el formato y estructura de los casos de prueba funcionales generados.

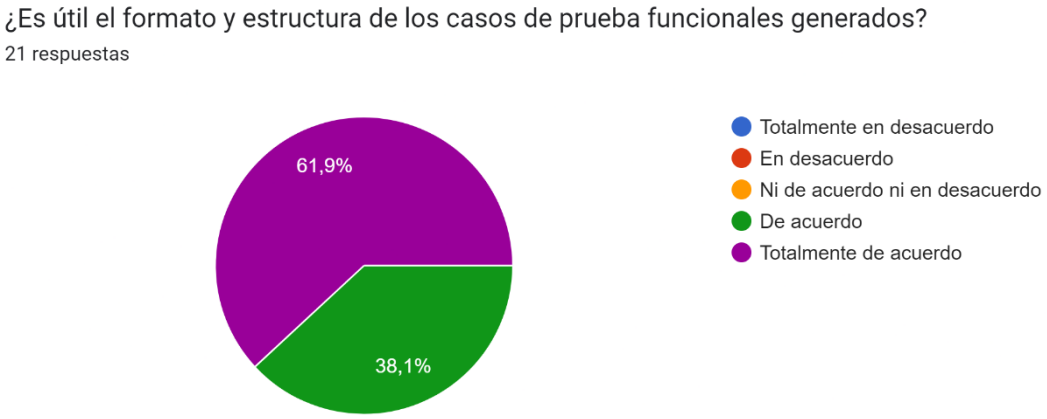


Figura 11: PU11

En la **Figura 12**, se muestra el ítem PU12, el 76.2% están totalmente de acuerdo, y el 23.8% están de acuerdo en que la personalización de los casos de prueba funcionales generados es útil. Al obtener más del 90% de aprobación esto destaca la importancia de la personalización en la generación de casos de prueba para satisfacer necesidades específicas.

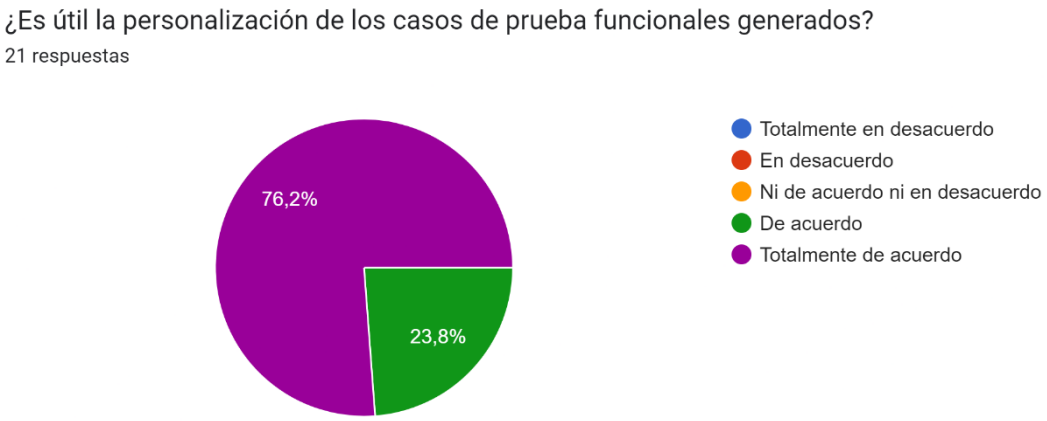


Figura 12: PU12

En la **Figura 13**, se muestra el ítem PU13, el 52.4% de los encuestados estiman que el sistema ha reducido el tiempo de creación de casos de prueba entre un 61-80%, y el 28.6% estiman una reducción entre el 81-100%. Esto muestra que más del 80% de los usuarios perciben una reducción entre el 61 al 100% en el tiempo requerido para crear casos de prueba funcionales, lo que evidencia la eficacia del sistema en la optimización del tiempo.

Figura 13: PU13

En la **Figura 14**, se muestra el ítem PU14, el 52.4% de los encuestados indican que solo el 0-20% de los casos de prueba generados han requerido modificación manual, mientras que un 19% señala modificaciones en el 21-40% de los casos. Más del 70% de los usuarios encuentran que los casos de prueba generados requieren 0-40% de modificación, lo que sugiere que el sistema es efectivo en producir resultados de alta calidad que cumplen con las expectativas de los usuarios sin necesidad de ajustes significativos.

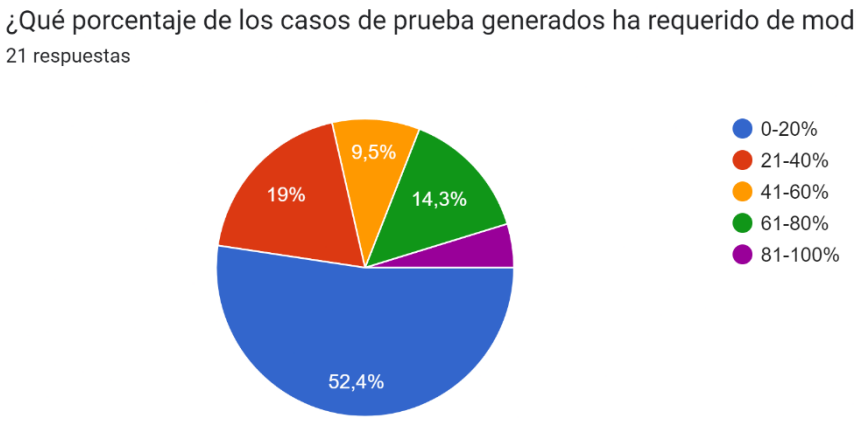


Figura 14: PU14

Ponderación de Resultados.

Para obtener un resultado preciso de los individuos que otorgaron respuestas positivas, se plantea una sumatoria de escalas superiores a la media. Por tanto, se escoge desde De acuerdo hasta Totalmente de acuerdo, no obstante, también se considera el nivel medio de Ni de acuerdo, ni en desacuerdo, para conocer las personas que se encuentran en este nivel intermedio y su respuesta apoye a la respuesta de la pregunta de investigación planteada.

Tabla 6: Evaluación de percepción de utilidad Nivel medio y alto

Pregunta	Ni de acuerdo, ni en desacuerdo	De acuerdo	Totalmente de acuerdo
1	4.8%	33.3%	61.9%
2	4.8%	47.6%	47.6%
3	0%	61.9%	38.1%
4	9.5%	52.4%	38.1%
5	4.8%	38.1%	57.1%
6	9.5%	28.6%	61.9%
7	0%	33.3%	66.7%
8	9.5%	28.6%	61.9%
9	0%	42.9%	57.1%
10	0%	19%	81%
11	0%	38.1%	61.9%
12	0%	23.8%	76.2%

Para las preguntas 13 y 14, se implementó una escala de Likert modificada con rangos porcentuales que van desde 0-20% hasta 81-100%. En el caso de la pregunta 13, los niveles se establecen de manera ascendente, donde un mayor porcentaje indica una valoración más

positiva. Sin embargo, la pregunta 14 sigue una lógica inversa, donde los porcentajes menores representan una respuesta más favorable, ya que esta pregunta evalúa tiempos de respuesta, siendo más positivo un menor tiempo de ejecución.

Tabla 7: Evaluación de percepción de utilidad Nivel medio y alto

Pregunta	**0-20%**	**21-40%**	**41-60%**	**61-80%**	**81-100%**
13	0%	4.8%	14.3%	52.4%	28.6%
14	52.4%	19%	9.5%	14.3%	4.8%

Como sumatoria de estos niveles altos, resulta en la Tabla 5.

Tabla 8: Total de percepción de utilidad Nivel medio y alto

Pregunta	**Total (De acuerdo + Totalmente De acuerdo, 0-20 + 21-40 y 61- 80 + 81-100)**
1	95,2
2	95,2
3	100
4	90,5
5	95,2
6	90,5
7	100
8	90,5
9	100
10	100
11	100
12	100
13	81
14	71,4
TOTAL	**86,464%**

Por tanto, como resultado de la evaluación se menciona que el **86,464%** de personas encuestadas perciben una utilidad entre media y alta, que corrobora la implementación de la herramienta en el entorno educativo y de desarrollo de software.

Tabla 9: Total de percepción de utilidad Ni de acuerdo, ni en desacuerdo.

Pregunta	**Total (Ni de acuerdo, ni en desacuerdo)**
1	4.8
2	4.8
3	0
4	9.5
5	4.8

6	9.5
7	0
8	9.5
9	0
10	0
11	0
12	0
13	14.3
14	9.5
TOTAL	**4,764%**

El análisis reveló que únicamente las preguntas 13 y 14 presentaron valoraciones por debajo del nivel promedio, con porcentajes de 4,8% y 19,1% respectivamente. Estas valoraciones representan apenas el 1,707% del total de la muestra evaluada, un porcentaje estadísticamente poco significativo que no afecta la valoración global positiva del sistema.

Para constancia de la valides de este documento:

Acción	Funcionario	Firma
Revisado por:	Ing. Wilman Chamba	Firmado electrónicamente por: WILMAN PATRICIO CHAMBA ZARAGOCIN

Anexo 13. Informe de Similitud de TIC

[Informe de Similitud de TIC]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Ing. Wilman Chamba

2025

Juan Francisco Castillo Estrella

TIC-JuanCastillo.pdf

PT-2 - Evaluación

SWE Models 2024-2025

Universidad Nacional de Loja

Detalles del documento

Identificador de la entrega

trn:oid:::1:3179967288

Fecha de entrega

11 mar 2025, 9:10 a.m. GMT-5

Fecha de descarga

12 mar 2025, 10:05 a.m. GMT-5

Nombre de archivo

TIC-JuanCastillo.pdf

Tamaño de archivo

2.8 MB

92 Páginas

22,411 Palabras

129,911 Caracteres

0% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 28 palabras)

Exclusiones

- N.º de coincidencias excluidas

Fuentes principales

0% Fuentes de Internet
0% Publicaciones
0% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

Caracteres reemplazados
38 caracteres sospechosos en N.º de páginas
Las letras son intercambiadas por caracteres similares de otro alfabeto.

Texto oculto
7 caracteres sospechosos en N.º de página
El texto es alterado para mezclarse con el fondo blanco del documento.

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.

Fuentes principales

0% Fuentes de Internet
0% Publicaciones
0% Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1	Trabajos del estudiante	
uazuay		<1%

Anexo 14. Certificado de Traducción.

[Certificado de Traducción]

Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja.

Elaborado por:
Lic. Viviana Valdivieso Loyola Mg. Sc.

2025

CERTIFICACIÓN DE TRADUCCIÓN

Loja, 14 de marzo de 2025

Lic. Viviana Valdivieso Loyola Mg. Sc.
DOCENTE DE INGLÉS

A petición verbal de la parte interesada:

CERTIFICA:

Que, desde mi legal saber y entender, como profesional en el área del idioma inglés, he procedido a realizar la traducción del resumen, correspondiente al Trabajo de Integración Curricular titulado **Sistema para el diseño de Casos de Prueba Funcionales a partir de Casos de Uso para la carrera Computación de la Universidad Nacional de Loja,** de la autoría de: **Juan Francisco Castillo Estrella,** portador de la cédula de identidad número **1105822447**

Para efectos de traducción se han considerado los lineamientos que corresponden a un nivel de inglés técnico, como amerita el caso.

Es todo cuanto puedo certificar en honor a la verdad, facultando al portador del presente documento, hacer uso del mismo, en lo que a bien tenga.

Atentamente. -

Firmado electrónicamente por:
VIVIANA DEL CISNE
VALDIVIESO LOYOLA

Lic. Viviana Valdivieso Loyola Mg. Sc.
1103682991

N° Registro Senescyt 4to nivel **1031-2021-2296049**
N° Registro Senescyt 3er nivel **1008-16-1454771**