UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Desarrollo de una aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando Test Driven Development

Trabajo de Titulación para optar al título de Ingeniero en Tecnologías de la Información

Autor:
Torres Marroquin, Ruben Dario

Tutor:
Ing. Diego Marcelo Reina Haro

Riobamba, Ecuador. 2025

DECLARATORIA DE AUTORÍA

Yo, Ruben Dario Torres Marroquin, con cédula de ciudadanía 1500956527, autor del trabajo de investigación titulado: Desarrollo de una aplicación web para bancos comunitarios en el canto el Chaco utilizando Test Driven Development, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, a la fecha de su presentación.

Ruben Dario Torres Marroquin
C.I: 1500956527

ACTA FAVORABLE - INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN

En la Ciudad de Riobamba, a los dos días del mes de octubre del 2024 luego de haber revisado el Informe Final del Trabajo de Investigación presentado por el estudiante **Ruben Dario Torres Marroquin** con CC: **1500956527** de la carrera **Ingenieria en Tecnologías de la Información** y dando cumplimiento a los criterios metodológicos exigidos, se emite el **ACTA FAVORABLE DEL INFORME FINAL DEL TRABAJO DE INVESTIGACIÓN** titulado **"Desarrollo de una aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando Test Driven Development"**, por lo tanto se autoriza la presentación del mismo para los trámites pertinentes.

DIEGO MARCELO REINA HARO

Mgs. Diego Reina
TUTOR

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

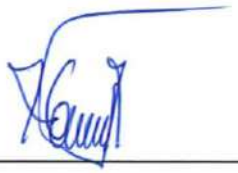
Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación Desarrollo de una aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando Test Driven Development por Ruben Dario Torres Marroquin, con cédula de identidad número 1500956527, bajo la tutoría de Mg. Diego Marcelo Reina Haro; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba a los 13 días del mes de diciembre del 2024

Pamela Buñay, Mgs.
PRESIDENTE DEL TRIBUNAL DE GRADO

Miryan Narváez, PhD.
MIEMBRO DEL TRIBUNAL DE GRADO

Hugo Paz, Dr.
MIEMBRO DEL TRIBUNAL DE GRADO



C E R T I F I C A C I Ó N

Que, **TORRES MARROQUIN RUBEN DARIO** CON CC: **1500956527**, estudiante de la Carrera **INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**, Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado **"DESARROLLO DE UNA APLICACIÓN WEB PARA BANCOS COMUNITARIOS EN EL CANTÓN EL CHACO UTILIZANDO TEST DRIVEN DEVELOPMENT"**, cumple con el **7 %**, de acuerdo al reporte del sistema Anti plagio **TURNITIN**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 08 de diciembre de 2024

DIEGO MARCELO REINA MARO

Mgs. Diego Reina
TUTOR

DEDICATORIA

A mi madre, por su amor incondicional, su apoyo constante y su infinita paciencia. Gracias por ser mi guía y por enseñarme el valor del esfuerzo y la dedicación. Sin ti, este logro no habría sido posible.

A mis hermanas, por ser mi fuente de inspiración y motivación. Gracias por creer en mí, por su apoyo inquebrantable y por estar siempre a mi lado en cada paso de este camino.

Ruben Dario Torres Marroquin

AGRADECIMIENTO

Expreso mi más profundo agradecimiento a todas las personas e instituciones que han hecho posible la realización de esto trabajo de investigación

A mis profesores y mentores de la facultad de ingeniería, cuya orientación y conocimientos han sido cruciales para mi desarrollo académico. Estoy inmensamente agradecido por su paciencia y dedicación, y por haberme motivado a alcanzar la excelencia en mi formación profesional.

A mis compañeros y amigos de la universidad, por su apoyo y comprensión a lo largo de este proceso. Gracias por los momentos compartidos, las largas sesiones de estudio y por estar siempre dispuestos a colaborar.

Ruben Dario Torres Marroquin

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La investigación se centra en el desarrollo y evaluación de la seguridad de una aplicación web para Bancos Comunitarios en el Cantón El Chaco, utilizando la metodología Test Driven Development (TDD). Se identificó que el problema radicaba en la gestión manual de las transacciones financieras, lo que dificultaba el análisis y seguimiento de los datos, además de comprometer la seguridad de las transacciones. La solución propuesta fue desarrollar la aplicación utilizando TDD para garantizar una construcción robusta y la detección temprana de errores. El desarrollo se realizó empleando tecnologías como .NET Core 8, Angular y PostgreSQL, permitiendo una arquitectura flexible y segura. Se empleó la metodología Kanban para una gestión ágil del proyecto. La investigación tuvo un enfoque cuantitativo que permitió evaluar la seguridad de la aplicación según los estándares de la norma ISO/IEC 25010, enfocándose en aspectos críticos como autenticidad, confidencialidad e integridad de los datos. Los resultados de las pruebas de seguridad, realizadas con un enfoque cuantitativo, mostraron mejoras significativas: en autenticidad, las vulnerabilidades se redujeron de 32 a 12, representado una mejora del 62.5%; en confidencialidad, las vulnerabilidades disminuyeron de 18 a 6, logrando una mejora del 66%; y en integridad, las vulnerabilidades se redujeron de 25 a 8, con una mejora del 68%. Estos resultados demuestran la efectividad de las estrategias de TDD para mitigar riesgos de seguridad y garantizar la protección de los datos. Estos resultados destacan la efectividad de las estrategias de TDD para mitigar riesgos de seguridad y garantizar la protección de los datos, atribuible en gran medida a las herramientas utilizadas, como .NET Core 8, Angular y PostgreSQL, que fueron fundamentales para lograr la mejora en la seguridad de la aplicación.

Palabras claves: Aplicación web, Bancos comunitarios, Test Driven Development (TDD), Seguridad, Metodología Kanban, .Net Core 8, Norma ISO/IEC 25010.

ABSTRACT

This research focuses on developing a web application for Community Banks in Cantón El Chaco using the test-driven development (TDD) methodology. This methodology has proven to be an effective approach in software development as it promotes the creation of tests before coding, thus facilitating the early identification of errors and improving the overall quality of the software. The application was developed using tools such as Angular, .NET Core, and PostgreSQL, and it not only met the expected functional requirements but also effectively addressed security vulnerabilities.

The security evaluation of the application was conducted according to the ISO/IEC 25010 standard, revealing significant results in reducing vulnerabilities. Regarding authenticity, vulnerabilities decreased from four to two, representing a 50% improvement. In the area of confidentiality, vulnerabilities were reduced from five to two, corresponding to a 60% improvement. Regarding integrity, a notable reduction in vulnerabilities was achieved, decreasing from four to one, translating to a 75% improvement. These results demonstrate the successful strategies implemented to ensure data security, using OWASP to identify and mitigate potential risks.

Keywords: Software development, web application, community banks, Test Driven Development (TDD), Angular, .NET Core, PostgreSQL, security, ISO/IEC 25010, vulnerabilities, authenticity, confidentiality, integrity, OWASP.

Reviewed by:

Lic. Eduardo Barreno Freire. Msc.
ENGLISH PROFESSOR
C.C. 0604936211

CAPÍTULO I. INTRODUCCIÓN

En la actualidad digital, las aplicaciones web desempeñan un papel fundamental en el ámbito financiero. Los Bancos Comunitarios del Cantón El Chaco enfrentan la necesidad de modernizar sus procesos y aumentar su alcance en la comunidad. Esta modernización requiere implementar soluciones que garanticen la seguridad en el desarrollo de software, dado que el manejo de información sensible es una preocupación crítica. Collins señala que "la integración de metodologías como el Desarrollo Guiado por Pruebas (TDD) permite mitigar vulnerabilidades desde las primeras etapas del desarrollo", es esencial para proteger los datos de los usuarios [1]. Además, Fowler enfatiza que "la implementación de TDD no solo mejora la calidad del software, sino que también refuerza la seguridad al incorporar pruebas específicas que validan la autenticación y la autorización en aplicaciones críticas" [2].

El Desarrollo Guiado por Pruebas (TDD) se presenta como una metodología clave para garantizar la calidad y confiabilidad de las aplicaciones web. Este enfoque permite la creación de pruebas antes de la escritura del código, facilitando la detección temprana de vulnerabilidades y promueve un diseño modular del código, que es más fácil de mantener y actualizar. Esta metodología no solo mejora la calidad del software, sino que también refuerza la seguridad en aplicaciones bancarias, integrando pruebas que validan la resistencia a ataques comunes, como la inyección de SQL y el cross-site scripting [3].

La implementación de TDD en el desarrollo de la aplicación web para los Bancos Comunitarios del Cantón El Chaco permitirá a estas instituciones ofrecer un servicio más eficiente y confiable a sus usuarios. TDD proporciona la capacidad de detectar errores en etapas tempranas, entender claramente los requisitos del sistema y diseñar un código modular que facilita su mantenimiento [4]. Esta metodología, combinada con herramientas como Angular para el desarrollo del front-end, .NET Core para el back-end y PostgreSQL como sistema de gestión de bases de datos, refuerza el enfoque proactivo en la seguridad. Además, al seguir las recomendaciones de OWASP y alinearse con los criterios de la norma ISO/IEC 25010, se asegura que la aplicación no solo cumpla con los altos estándares de calidad y seguridad. Así, se asegura que la aplicación no solo sea eficiente, sino que también esté protegida contra amenazas cibernéticas que podrían comprometer la seguridad de la información financiera.

1.1 Planteamiento del problema

En el contexto actual de los Bancos Comunitarios en el Cantón El Chaco, se evidencia una necesidad crítica no satisfecha relacionada con la falta de sistemas informáticos eficientes. La investigación realizada por la Fundación para el Desarrollo Comunitario (FDC) en 2021 destaca que el 80% de los Bancos Comunitarios en la región carecen de herramientas tecnológicas específicas, generando ineficiencias en la administración de recursos financieros y limitando la calidad de los servicios ofrecidos.

Los problemas identificados incluyen dificultades en el seguimiento de transacciones, gestión ineficiente de clientes y cuentas, retrasos en procesos administrativos, errores en el manejo de inventarios y dificultades en la generación de informes. Estos desafíos afectan la eficiencia operativa y la calidad de servicios, como retrasos en la aprobación de préstamos y dificultades para tomar decisiones informadas.

El enfoque de Desarrollo Guiado por Pruebas (TDD) emerge como una solución integral. Ofreciendo calidad desde la concepción del sistema, TDD facilita la adaptación a cambios futuros, crucial para Bancos Comunitarios ante evoluciones normativas y necesidades cambiantes. Además, TDD puede contribuir a reducir costos de desarrollo y mantenimiento, aspecto vital dado los recursos limitados de estas instituciones.

1.2 Justificación

La integración de Test Driven Development (TDD) en la creación de una aplicación web destinada a las instituciones financieras del Cantón El Chaco es fundamental no solo para garantizar la seguridad y calidad del software, sino también para proporcionar una solución tecnológica adecuada que responda a las necesidades específicas de estas entidades. Dada la naturaleza sensible de los datos financieros que se manejarán, TDD permite detectar y solucionar problemas en las primeras fases del desarrollo, culminando en un sistema más sólido y confiable.

Esta aplicación optimiza la administración de recursos, facilitando el seguimiento de transacciones y mejorando la eficiencia en los procesos. Al reducir errores y tiempos de respuesta, se eleva la experiencia del usuario y se fortalece la confianza en los servicios ofrecidos. Esto es crucial en un contexto donde la ineficiencia puede afectar directamente la calidad del servicio y la satisfacción del cliente.

Además, TDD contribuye a un diseño modular del código, permitiendo adaptaciones ágiles a cambios en los requisitos y simplifica futuras actualizaciones. Esta capacidad de adaptación asegura que el software no solo cumpla con los estándares actuales, sino que también esté preparado para afrontar nuevos desafíos y regulaciones en el sector financiero.

1.3 Formulación del problema

¿El uso del enfoque TDD durante el desarrollo de una aplicación web para bancos comunitarios en el Cantón El Chaco incidirá en la seguridad del sistema, conforme con los estándares de calidad establecidos en la norma ISO/IEC 25010?

1.4 Objetivos

Objetivo General

Desarrollar una aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando Test Driven Development.

Objetivos específicos

- Investigar la metodología Test Driven Development (TDD) en el contexto del desarrollo de aplicaciones web para bancos comunitarios.
- Implementar la aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando el enfoque TDD
- Evaluar la seguridad de la aplicación web para bancos comunitarios en el Cantón El Chaco, según los estándares de calidad de la norma ISO/IEC 25010.

CAPÍTULO II. MARCO TEÓRICO

2.1 Aplicación Web

El concepto de aplicación web está ligado al almacenamiento en la nube. Así, la información se guarda permanentemente en grandes servidores de internet. Una vez que el usuario abre la aplicación y accede a ella, se envían a su dispositivo los datos que requiere. En esencia, se envía una réplica temporal del archivo en cuestión, algo parecido a una captura de pantalla de una página web con la que se interactúa y actualiza conforme cambia [5].

La arquitectura de las aplicaciones web está conformada por una serie de máquinas interconectadas a una red, que puede ser Internet o una Intranet corporativa. Esta arquitectura generalmente se basa en el modelo cliente-servidor, donde los clientes, que pueden ser navegadores web o aplicaciones móviles, envían solicitudes a los servidores. Los servidores, por su parte, procesan estas solicitudes, acceden a bases de datos o recursos internos, y devuelven la información requerida al cliente. Este intercambio de datos se realiza mediante protocolos de comunicación, como HTTP o HTTPS, garantizando así la integridad y la seguridad de la información transmitida.

La Figura 1 ilustra este esquema, donde se observa la interacción entre los diferentes componentes de la arquitectura, incluidos los servidores de aplicaciones, servidores de bases de datos y los dispositivos clientes.

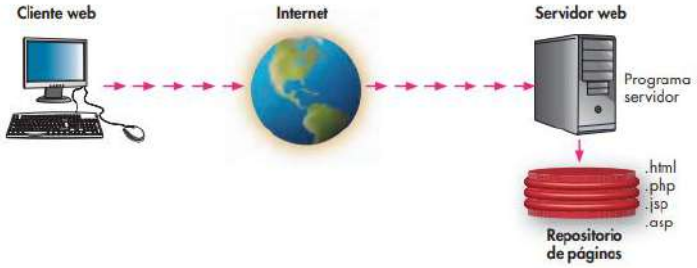


Figura 1:Esquema Básico del servicio web

Fuente: [2]

2.1.1 Tipos de apps web

Las aplicaciones web se pueden clasificar en cinco categorías principales. Estas son:

- Estática: Aunque se aconseja el uso de Ajax o jQuery, por lo general se desarrollan utilizando HTML5 Y CSS3, y su contenido tiende a permanecer estático. Se pueden usar GIF o videos para agregar elementos dinámicos [6]. Es importante recordar que este tipo de aplicaciones web se suelen utilizar en secciones que brindan información sobre una empresa.
- Dinámica: Representan una evolución lógica similar a la que se describió anteriormente. Toda la información se carga y actualiza desde una base de datos al abrir la aplicación. Lenguajes como PHP o ASP se utilizan y modifican a menudo en el panel CMS [7]. Son muy buenos para actualizar foros o bases de datos y se pueden personalizar.

- Tienda virtual o comercio electrónico: La estructura interna y los componentes incluyen un panel de control para facilitar la carga y eliminación de productos del catálogo. Además, es necesario incorporar una pasarela de pago eficiente, así como todas las herramientas necesarias para monitorear el número de unidades vendidas. El administrador debe prestar atención continua a la gestión para mantener el catálogo comercial actualizado. [8]
- Portal web: Página dividida en secciones, como foros, noticias, servicio al cliente o acceso para usuarios registrados. El objetivo es garantizar un acceso inmediato desde dispositivos móviles a cada sección. La programación y la actualización constante de las secciones son necesarias para su desarrollo. [9]
- Gestor de contenidos: Dado que tiene un diseño estático, es ideal para blogs o sitios de noticias especializados en un tema. La plataforma permite la modificación y creación de contenido a través de un panel de administración. Es común utilizar plantillas y plataformas predefinidas para construir sitios web. [10]

2.2 Test Driven Development

Test Driven Development (TDD) o en español desarrollo guiado por pruebas, es una metodología de desarrollo de software que se basa en la escritura de pruebas automatizadas antes de escribir el código de producción.

En el State of Agile survey [11] del año 2020, el 30 % de los practicantes agiles informaron que usan TDD con regularidad. Aunque este porcentaje ha venido disminuyendo (en encuestas anteriores, la adopción de TDD fue de alrededor del 50 %), esta práctica sigue siendo una de las técnicas agiles más importantes en el desarrollo de software. TDD es una evolución de la técnica test-first descrita en Extreme Programming. “TDD no es una técnica de prueba, sino una técnica de diseño lógica” [12]. “TDD propone que, en lugar de realizar un diseño o modelo de software, se debe enfrentar el desarrollo en base al planteamiento de pruebas unitarias antes de la generación efectiva del código” [13].

El proceso de desarrollo en TDD sigue comúnmente tres pasos: primero, se crea una prueba que debería fallar; después, se implementa el código mínimo necesario para que la prueba pase con éxito; por último, se refactoriza el código para mejorar su calidad sin alterar su comportamiento original. “Este enfoque prioriza la calidad del software al garantizar que cada funcionalidad esté respaldada por pruebas exhaustivas desde el inicio del desarrollo, facilitando la detección temprana de errores y promoviendo la mejora continua del código” [14].

Siguiendo la categorización de Araújo [15], TDD puede ser aplicado a dos niveles, a saber:

2.2.1 Nivel de micro iteraciones

En este nivel, el desarrollo se realiza mediante pruebas unitarias escritas por los propios desarrolladores de la aplicación, según Quiel [16]:

- Agregue rápidamente una prueba.
- Ejecute todas las pruebas y asegúrese de que sólo falle la prueba.
- Haga un pequeño cambio.
- Ejecute todas las pruebas y asegúrese de que pasen.
- Reconstruir para eliminar duplicados.

Lo anteriormente expuesto se puede resumir en el diagrama de actividad de la Figura 2.

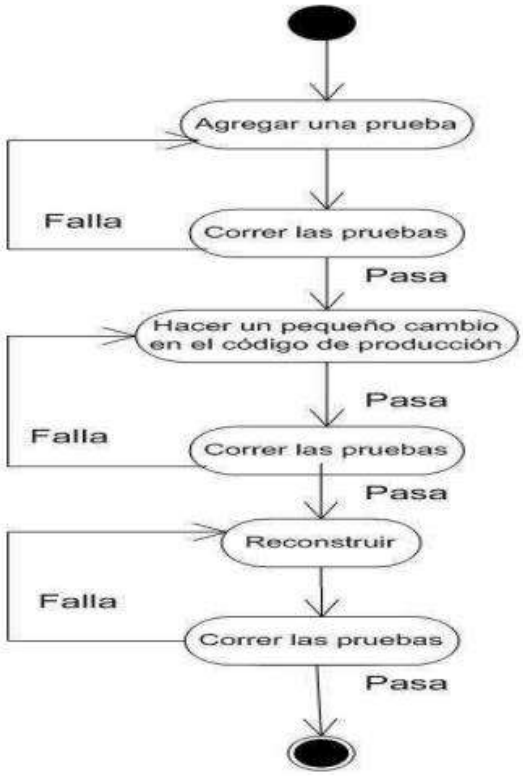


Figura 2:Pasos en un ciclo TDD

Fuente:[13]

Las tareas principales se agrupan en tres categorías: “Escribir y ejecutar pruebas”, “Escribir código de producción” y “Reconstrucción”. A continuación, se detallan estas categorías.

2.2.2 Escribir y ejecutar pruebas:

Se deben elaborar pruebas que sigan un patrón definido y detallado. En particular, se destacan las pruebas.

Están escritos por el propio programador en el mismo lenguaje de programación que la aplicación y su ejecución está automatizada. Esto último es necesario para obtener retroalimentación inmediata indicando que la tasa de cambio entre las pruebas de código está programada en intervalos de no más de diez minutos [17]. La automatización también permite la aplicación continua de conjuntos de pruebas, incluidas pruebas de regresión inmediatas y continuas.

Se deben escribir pruebas para probar las operaciones que deben realizarse. Deben estar aislados para que puedan funcionar independientemente unos de los otros, independientemente del orden [18]. Esta separación establecerá soluciones de código consistentes y débilmente acopladas con componentes ortogonales. Deben escribirse antes del código que desea probar.

2.2.3 Escritura de código

La codificación es el proceso de pasar una prueba unitaria. Contiene cuatro estrategias para salir rápidamente de situaciones malas, descritas en forma de plantillas. Rendimiento esperado, triangulación, implementación explícita y uno a muchos.

2.2.4 Reconstrucción (Refactoring)

Se define la reconstrucción como una práctica que consiste en mejorar el código existente, de manera disciplinada, sin alterar su comportamiento externo, para hacerlo más fácil de entender y modificar. La mejora de código se logra mediante la aplicación de una serie de consejos entre los cuales se destaca la eliminación de duplicación, la división en segmentos menores, la estructuración de manera que resulte más conveniente y la aplicación de las ventajas que brinde el lenguaje de programación [19]. La reconstrucción cumple un rol sustancial como complemento del diseño.

El orden de ejecución de las tareas se resume con la frase "Rojo/Verde/Reconstrucción" (Red/Green/Refactor). En la etapa de Rojo, una prueba ha fallado, mientras que, en la etapa de Verde, la aplicación pasa las pruebas. Como se aprecia en la Figura 3, este ciclo permite un desarrollo ágil y eficaz del software.

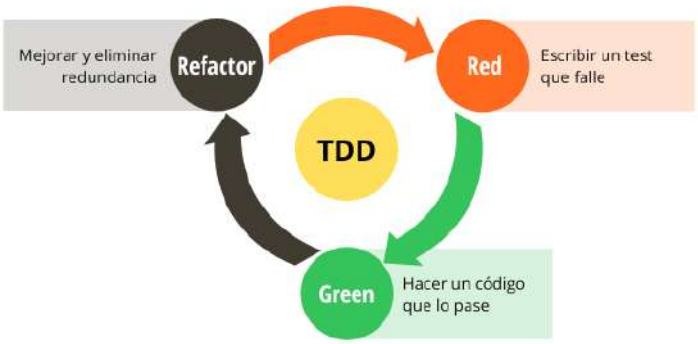


Figura 3:Ciclo TDD "Rojo-Verde-Refactorizar"

Fuente: [20]

2.2.5 Nivel Iteración o Funcional

El desarrollo es guiado por pruebas de aceptación. Este nivel, mencionado por Soraluz [21] como ATDD (Aceptance TDD), consta de los siguientes pasos:

- Los usuarios especifican pruebas antes de implementar la funcionalidad.
- Una vez escrito el código, las pruebas sirven como criterios de aceptación.

Construir experimentos a este nivel también es un proceso evolutivo, con la diferencia de que los productos se crean durante un periodo de tiempo más largo, a un nivel iterativo y esencialmente en manos del usuario con el apoyo de verificadores y programadores.

2.2.6 Herramientas para TDD

Jasmine: Este marco para probar código JavaScript no depende de ningún otro framework. Se ejecuta en navegadores y en Node.js [22]. Su sintaxis es limpia y clara, permitiendo a los desarrolladores escribir pruebas fácilmente.

Esta herramienta puede ser útil para mejorar la calidad del software, pero es importante ser consciente de sus limitaciones. A continuación, se presenta la Tabla 1 comparativa que resume las ventajas y desventajas de Jasmine TDD.

Tabla 1: Ventajas y Desventajas de Jasmine

Ventajas	Desventajas
Diseño más limpio y robusto	Curva de aprendizaje
Menos errores	Sobrecarga inicial
Mayor confianza en el código	Mantenimiento de pruebas
Facilidad de mantenimiento	Enfoque limitado

Karma: es el test-runner, es decir, el módulo que permite automatizar algunas de las tareas de las suites de testing, como Jasmine. Karma, además, ha sido desarrollado directamente por el equipo de Angular, proporciona cierta garantía de continuidad en su uso y desarrollo en el futuro, siendo así una buena opción.

Se puede observar que, con estos dos elementos, ya se tiene preparado el entorno para añadir pruebas a la aplicación. Al entrar en más detalle, se recordarán brevemente los distintos tipos de pruebas que se pueden utilizar. Principalmente se utilizarán dos: pruebas unitarias y pruebas de integración [23].

Karma es una herramienta popular para ejecutar pruebas automatizadas de JavaScript en múltiples navegadores y sistemas operativos en la Tabla 2 se muestra un resumen de la comparativa de ventajas y desventajas.

Tabla 2: Comparativa de ventajas y desventajas de Karma TDD

Aspecto	Ventajas	Desventajas
Ejecución de pruebas	Ejecución de pruebas en múltiples navegadores y sistemas operativos, lo que garantiza una amplia cobertura de pruebas	Requiere la instalación de navegadores y sistemas operativos en el entorno de desarrollo
Integración continua	Integración perfecta con herramientas de integración continua (CI)	Requiere la configuración y el mantenimiento de la infraestructura de CI

	para automatizar la ejecución de pruebas	
Informes detallados	Informes detallados y completos que facilitan la identificación y resolución de errores	La generación de informes detallados puede afectar el rendimiento de las pruebas
Amplia comunidad	Gran comunidad de usuarios y desarrolladores que ofrecen soporte y recursos	La documentación oficial puede ser difícil de encontrar o entender

Fuente: [24]

2.2.7 Test Unitarios

Las pruebas unitarias son esenciales para probar la unidad más pequeña de funcionalidad en el desarrollo de software. Ayudan a mantener la calidad del código y son una parte importante del proceso de desarrollo [25]. Una buena práctica consiste en escribir código en pequeños bloques funcionales y crear pruebas unitarias para cada uno de esos bloques. También es recomendable iniciar escribiendo pruebas unitarias antes de la codificación y ejecutarlas automáticamente tras cada cambio en el software.

Esto permite detectar errores rápidamente aislando la sección de código que los contiene. Las pruebas unitarias fomentan un enfoque modular y mejoran la cobertura y la calidad de las pruebas. Además, la automatización de estas pruebas permite al equipo concentrarse en el desarrollo. [26]

Una prueba unitaria consiste en un conjunto de instrucciones diseñadas para confirmar la precisión de una parte específica y aislada del código de una aplicación, como una función o un método. Como se aprecia en la Figura 4.

```
51    @Test
52    public void defaultDummyTest() throws Exception {
53        given(dummyService.getDefaultSentence()).willReturn(defaultDummy);
54
55        final MvcResult result = mockMvc.perform(get("/contact/dummy")).andExpect(status().isOk())
56                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
57                .andExpect(jsonPath("$.dummyResponse", is("Hi, you guys!!"))).andReturn();
58
59    }
60
61    @Test
62    public void numberOfFriendsTest() throws Exception {
63
64        final Dummy dummyFriends = new Dummy(7);
65
66        given(dummyService.getGuyWithFriends(friendNumberCaptor.capture())).willReturn(dummyFriends);
67
68        final MvcResult result = mockMvc.perform(get("/contact/friends/" + dummyFriends.getFriends()))
69                .andExpect(status().isOk()).andExpect(content().contentType(MediaType.APPLICATION_JSON))
70                .andExpect(jsonPath("$.dummyResponse", is(Dummy.MESSAGE_HI + 7 + Dummy.MESSAGE_FRIENDS))).andReturn();
71
72    }
73
74    @AfterEach
75    void tearDown() {
76        reset(dummyService);
77    }
78
```

Figura 4: Prueba Unitaria bloque de código

Fuente: [27]

2.3 La Norma ISO/IEC 25010

Este estándar internacional define un modelo de calidad para el software, destacando nueve características de calidad y sus respectivas subcaracterísticas. En el contexto del desarrollo de sistemas informáticos para instituciones financieras, como las del Cantón El Chaco, la implementación de esta norma puede ofrecer una serie de beneficios notables.

Dentro de esta norma, la sexta característica, seguridad, se refiere a la capacidad del sistema para proteger la información y los datos, asegurando que las personas u otros productos solo tengan acceso a los datos apropiados según sus tipos y niveles de autorización, y defendiéndose contra posibles patrones de ataque de agentes malintencionados.

Esta característica se divide en varias subcaracterísticas, enfocándose en la subcaracterística de integridad, que se refiere a la capacidad del producto para garantizar que el estado del sistema y los datos estén protegidos contra modificaciones o eliminaciones no autorizadas, ya sea por acciones malintencionadas o por errores informáticos. “Esto implica mantener la consistencia y la precisión de los datos a lo largo del tiempo, evitando cambios no autorizados que puedan comprometer la integridad de la información” [28].

Estas subcaracterísticas, que forman parte de la característica principal de "Seguridad", son fundamentales para garantizar la confidencialidad, integridad, disponibilidad y confiabilidad de los datos y sistemas. A continuación, se presenta la Tabla 3, que resume las subcaracterísticas de seguridad que son esenciales para proteger la información y los sistemas.

Tabla 3: Subcaracterísticas de seguridad

Subcaracterísticas	Descripción
Confidencialidad	Capacidad de asegurar que los datos sean accesibles únicamente a aquellos que cuentan con la autorización correspondiente.
Integridad	Capacidad de un producto para garantizar que el estado de su sistema y sus datos están protegidos frente a modificaciones o eliminaciones no autorizadas, ya sea por acciones malintencionadas o por errores informáticos.
No repudio	Capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente.
Responsabilidad	Capacidad de rastrear de forma inequívoca las acciones de una entidad.
Autenticidad	Capacidad de un producto para demostrar que la identidad de un sujeto o recurso es la que se afirma.
Resistencia	Capacidad de mantener la operación de un producto bajo condiciones de ataque de un actor malicioso.

Fuente: [29]

Al evaluar la integridad según los criterios establecidos por la Norma ISO/IEC 25010, se deben considerar aspectos como la implementación de mecanismos de control de acceso, el registro y seguimiento de cambios, la validación de datos y la protección contra ataques informáticos. Esta evaluación proporcionará información valiosa sobre el cumplimiento de los estándares de seguridad y confiabilidad del sistema en relación con la integridad de los datos.

2.4 Tecnologías Angular y Node.js

Angular

Este framework de desarrollo front-end, basado en JavaScript, se utiliza para construir aplicaciones web de una sola página (SPA) de manera eficiente. Proporciona herramientas y estructuras que permiten a los desarrolladores crear interfaces de usuario dinámicas e interactivas.

La Figura 5 ilustra la arquitectura MVC (Modelo-Vista-Controlador) en Angular, resaltando la interacción entre sus componentes. Esta estructura modular favorece la separación de responsabilidades, optimizando el desarrollo y mantenimiento de aplicaciones web.

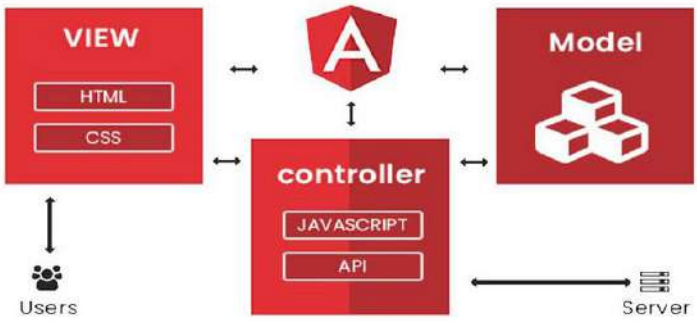


Figura 5: Arquitectura MVC en Angular

Fuente: [29]

Angular es una plataforma que facilita el desarrollo de aplicaciones web mediante HTML y JavaScript en el lado del cliente, lo que alivia la carga del servidor. Esta optimización resulta en una mayor velocidad de ejecución y, en consecuencia, un rendimiento superior en las aplicaciones. [30]

En la Tabla 4 se presenta un análisis detallado de las ventajas y desventajas de Angular.

Tabla 4:Ventajas y Desventajas de Angular

Ventajas	Desventajas
Framework completo: Angular es un framework completo que ofrece todas las herramientas necesarias para construir aplicaciones web de una manera eficiente.	Curva de aprendizaje: Para dominar Angular por completo, los desarrolladores deben invertir tiempo en aprender su amplia gama de conceptos y características.
MVC arquitectónico: Angular sigue el patrón de diseño Modelo-Vista-Controlador (MVC), lo que facilita la organización y mantenimiento del código.	Complejidad inicial: Las aplicaciones Angular pueden parecer complejas al principio debido a la cantidad de conceptos y abstracciones que introduce el framework.

Inyección de dependencias: Angular ofrece un sistema de inyección de dependencias que facilita la gestión de dependencias y la escritura de código modular y mantenible.	Tamaño del paquete: Las aplicaciones Angular tienden a tener un tamaño de paquete más grande en comparación con otros frameworks más ligeros, lo que puede afectar el rendimiento, especialmente en dispositivos móviles.

Fuente: [31]

Node.js

Este entorno de ejecución de JavaScript del lado del servidor permite a los desarrolladores construir aplicaciones web escalables y de alto rendimiento. Se destaca que "Node.js utiliza la arquitectura «Single Threaded Event Loop» para manejar múltiples clientes al mismo tiempo" [32].

Se analiza detalladamente las ventajas y desventajas de Node.js como se muestra en la siguiente Tabla 5.

Tabla 5: Ventajas y Desventajas de Node.js

Ventajas	Desventajas
Eficiencia en tiempo real: Node.js es ideal para aplicaciones que requieren operaciones en tiempo real y manejo de múltiples conexiones simultáneas debido a su modelo de entrada y salida sin bloqueo (non-blocking I/O).	Monohilo: Node.js utiliza un solo hilo de ejecución, lo que puede provocar cuellos de botella en aplicaciones que realizan tareas intensivas de CPU.
Ecosistema npm: Node.js cuenta con el mayor ecosistema de paquetes y bibliotecas de código abierto a través de npm (Node Package Manager), lo que facilita la integración de funcionalidades adicionales en las aplicaciones.	Falta de madurez: Algunas bibliotecas y herramientas de Node.js pueden carecer de madurez en comparación con otros ecosistemas más establecidos.
JavaScript en el lado del servidor: Al utilizar JavaScript tanto en el lado del cliente como en el servidor, los desarrolladores pueden compartir código y lógica entre el front-end y el back-end, lo que simplifica el desarrollo y la mantenibilidad.	Callback Hell: El anidamiento excesivo de callbacks puede llevar a una estructura de código difícil de entender y mantener, conocida como "callback hell".
Escalabilidad horizontal: Node.js es altamente escalable y permite escalar horizontalmente mediante la adición de instancias de servidores, lo que facilita el manejo de cargas de trabajo crecientes.	Inmadurez en ciertas áreas: A pesar de su popularidad, Node.js puede carecer de madurez en ciertas áreas, como el soporte nativo para operaciones de entrada/salidas intensivas.

Fuente: [33]

La combinación de Angular como tecnología de cliente y Node.js como entorno de servidor proporciona una plataforma sólida y moderna para el desarrollo de sistemas informáticos complejos.

2.5 Herramientas de Desarrollo

Visual Code

VS Code es un editor de código fuente gratuito desarrollado por Microsoft, muy utilizado por desarrolladores en muchas plataformas diferentes gracias a su flexibilidad y facilidad de uso. Compatible con muchos lenguajes de programación, desde los más populares como JavaScript y Python hasta otros más especializados, ofrece una interfaz personalizable y extensible [34]. Su sólido ecosistema de extensibilidad ofrece nuevas funciones, como herramientas de depuración y soporte específico para plataformas, mientras que funciones inteligentes como el resaltado de sintaxis y la finalización inteligente mejoran la productividad y facilitan el crecimiento.

Estos son los elementos principales del interfaz de Visual Studio Code. Como se aprecia en la Figura 6.

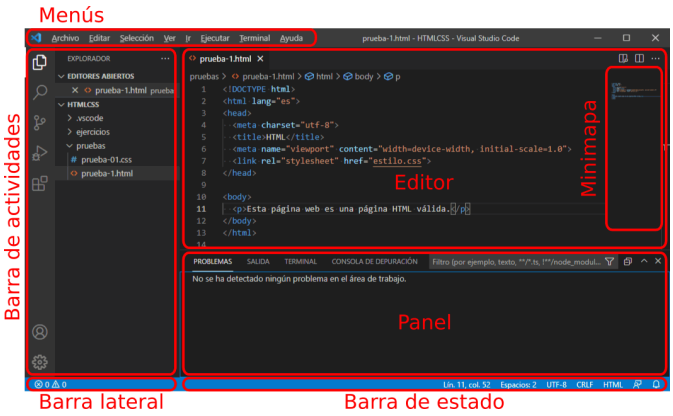


Figura 6: Interfaz de Visual Studio Code

Fuente: [35]

Visual Studio

Es un Entorno de Desarrollo Integrado (IDE) creado por Microsoft que ofrece una ventanilla única para desarrollar software. Con soporte para una variedad de lenguajes de programación, incluyendo C++, C#, Python, y JavaScript, proporciona herramientas integradas como un editor de código, un depurador y un compilador. Su interfaz es personalizable para adaptarse a las preferencias del usuario, y puede utilizarse para crear aplicaciones en diferentes plataformas como Windows, macOS, Android y iOS. Además, cuenta con un amplio ecosistema de extensiones que añaden funcionalidades adicionales, como soporte para marcos específicos y herramientas de control de versiones.

El entorno de desarrollo integrado (IDE) más rápido para la productividad es este, el cual permite dirigir aplicaciones a cualquier plataforma o dispositivo, como se aprecia en la Figura 7.

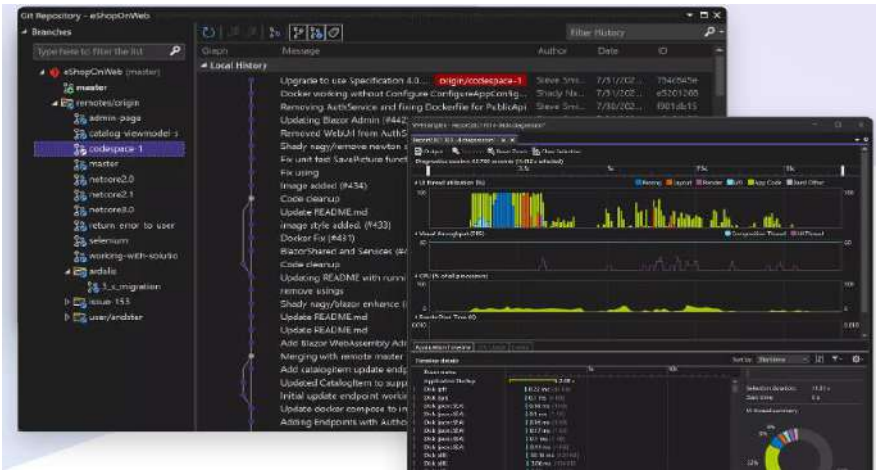


Figura 7: IDE Visual Studio

Fuente: [35]

PostgreSQL

También conocido como Postgres, es un sistema de gestión de bases de datos relacional, orientado a objetos y de código abierto. Destaca por su alto rendimiento, fiabilidad y escalabilidad, además de su soporte para múltiples lenguajes de programación y sus funciones avanzadas [36]. Utilizado en una variedad de aplicaciones como sitios web, sistemas de gestión de contenido, aplicaciones empresariales y almacenamiento de datos científicos, ofrece ventajas como su gratuidad, seguridad confiable y un amplio ecosistema de usuarios y desarrolladores [37].

Representación esquemática de las entidades que participan en el funcionamiento normal del gestor de bases de datos, como se muestra en la Figura 8.

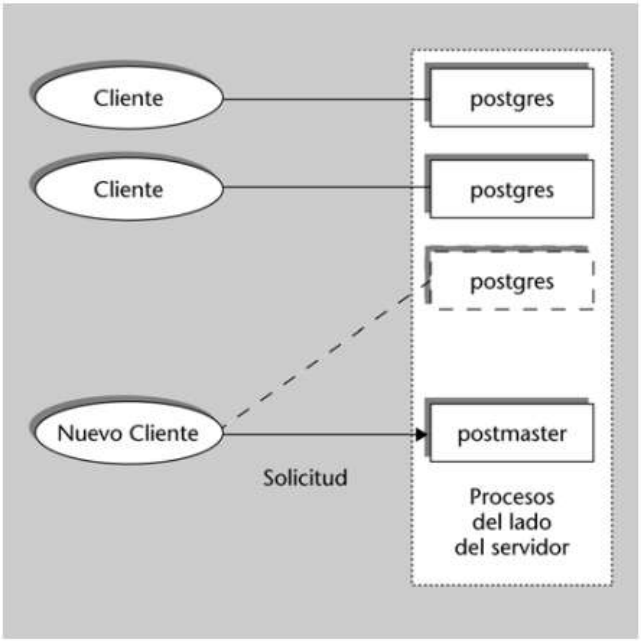


Figura 8: Arquitectura de PostgreSQL

Fuente: [38]

2.6 Metodología Kanban

Este enfoque se presenta como una estrategia ágil para la gestión de proyectos, que enfatiza la visualización del flujo de trabajo y la mejora continua. Se utiliza para gestionar tareas de manera eficiente y optimizar la entrega de valor [39]. Sus aspectos clave incluyen:

- Tablero Kanban: Es un tablero físico o digital que representa las diferentes etapas del flujo de trabajo, como "Por hacer", "En proceso" y "Hecho". Se proporciona una visualización clara del estado de todas las tareas, como se muestra en la Figura 9.

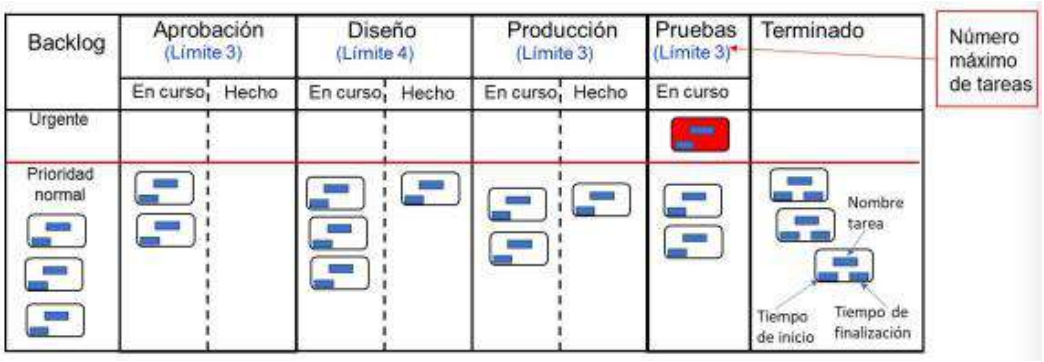


Figura 9 : Ejemplo de tablero Kanban

Fuente: [39]

- Tarjetas Kanban: Representan tareas individuales y se mueven a través del tablero a medida que avanzan en el proceso. Contienen información sobre la tarea, como su descripción, asignación y estado.
- Columnas: Las columnas en el tablero representan las diferentes etapas del flujo de trabajo. Cada columna indica el estado de las tareas en esa etapa, facilitando el seguimiento y la gestión del progreso.
- Límites de Trabajo en Proceso (WIP): Se establece un límite máximo de tareas permitidas en cada columna para controlar el flujo de trabajo. Limitar el WIP ayuda a evitar la sobrecarga y a mantener un enfoque claro en las tareas prioritarias.

Además de estos elementos clave, la metodología Kanban se basa en una filosofía de mejora continua, flexibilidad y colaboración entre equipos. Al promover la visibilidad, eficiencia, flexibilidad, comunicación y productividad.

2.7 OWASP ZAP

Zed Attack Proxy es una herramienta de prueba de penetración que se utilizó para identificar vulnerabilidades de seguridad en aplicaciones web mediante la simulación de ataques y el análisis de respuestas.

A continuación, se detallan las características principales de OWASP ZAP que la hacen destacar en el campo de la seguridad informática.

- Intercepción de Tráfico: OWASP ZAP actúa como un proxy entre el navegador y la aplicación web, lo que permite a los usuarios interceptar, modificar y analizar el tráfico HTTP/HTTPS. Esta característica es fundamental para las pruebas de seguridad, ya que permite explorar las solicitudes y respuestas en tiempo real [40].
- Escaneo Automático: ZAP ofrece un escáner automatizado que puede identificar vulnerabilidades comunes como inyecciones SQL, Cross-Site Scripting (XSS) y configuraciones inseguras. Este escáner puede ser personalizado según las necesidades del usuario [41].

Beneficios y desventajas de OWASP ZAP

No solo es una herramienta potente para la identificación de vulnerabilidades en aplicaciones web, sino que también tiene sus beneficios y desventajas. Conocer estos aspectos es esencial para determinar su adecuación en un entorno de pruebas de seguridad. A continuación, en la Tabla 6, se presentan las principales ventajas y desventajas de OWASP ZAP.

Tabla 6: Ventajas y Desventajas de OWASP ZAP

Ventajas	Desventajas
Herramienta de código abierto y gratuita	No tan completa como herramientas comerciales
Interfaz intuitiva que beneficia a principiantes y expertos	Requiere Conocimientos Técnicos: Necesaria para configuraciones avanzadas
Soporte activo y recursos disponibles	Algunas funcionalidades dependen de plugins de la comunidad
Facilita la automatización en entornos de desarrollo	Puede consumir muchos recursos durante escaneos

Fuente: [42]

CAPÍTULO III. METODOLOGÍA

3.1 Tipo de Investigación

El proyecto de investigación se basó en un enfoque cuantitativo, que permitió analizar la seguridad de la aplicación web desarrollada para los Bancos Comunitarios del Cantón El Chaco utilizando Test Driven Development (TDD).

3.2 Diseño de Investigación

Se determinó un diseño de investigación no experimental que incorpora enfoques cuantitativos y cualitativos, ya que este diseño no manipuló las variables independientes y permitió una evaluación detallada de la seguridad de la aplicación web destinada a los Bancos Comunitarios del Cantón El Chaco. En el enfoque cuantitativo, se reunieron datos precisos y medibles sobre la seguridad de la aplicación mediante herramientas como OWASP ZAP, que facilitó la detección de vulnerabilidades, así como el análisis de registros de seguridad para obtener métricas significativas. En el enfoque cualitativo, se realizaron entrevistas con expertos en desarrollo de software y seguridad, con el objetivo de identificar las mejores prácticas y los desafíos en la aplicación de la metodología TDD en este ámbito.

3.3 Población de estudio y tamaño de muestra

La seguridad del sistema se evaluó utilizando OWASP ZAP y Codacy que ofrecen la capacidad de realizar un número ilimitado de solicitudes para realizar pruebas de seguridad, por tal motivo la población es infinita, pero está limitada por los recursos de la máquina donde se ejecuta y las configuraciones de la herramienta.

Se configuró el escenario para realizar 500 pruebas de penetración con OWASP ZAP, 150 análisis de calidad de código con Codacy y 800 evaluaciones de conformidad con los criterios de seguridad de la norma ISO/IEC 25010.

3.4 Técnicas de Recolección de Datos

La recopilación de datos para evaluar la seguridad de la aplicación web destinada a los Bancos Comunitarios del Cantón El Chaco se llevó a cabo mediante tres técnicas complementarias:

- Entrevistas con expertos en desarrollo de software y seguridad informática: A través de las entrevistas, se recopilaron recomendaciones clave sobre cómo adaptar las prácticas de seguridad en el contexto específico de los Bancos Comunitarios. Estos datos fueron utilizados para personalizar los enfoques de seguridad durante las pruebas, seleccionando criterios relevantes basados en los riesgos específicos de la infraestructura financiera. Las entrevistas también ayudaron a determinar las vulnerabilidades más críticas que debían ser evaluadas con mayor prioridad, permitiendo una orientación más precisa en las pruebas de seguridad.

- Pruebas automatizadas con OWASP ZAP: Los datos obtenidos de las pruebas de penetración realizadas fueron utilizados para detectar y analizar posibles vulnerabilidades en la infraestructura de la aplicación. La información recopilada ayudó a identificar las debilidades en la respuesta del sistema ante posibles amenazas, permitiendo ajustar la estrategia de seguridad para reforzar los puntos más vulnerables.
- Análisis estático del código con Codacy: El análisis proporcionó información clave sobre la calidad del código fuente en cuanto a seguridad. Estos datos fueron utilizados para identificar áreas del código que necesitaban ser mejoradas para cumplir con los estándares de seguridad, garantizando la protección de la integridad y confidencialidad de los datos procesados por la aplicación.

3.5 Identificación de variables

3.5.1 Variable dependiente

Desarrollo de la aplicación Web

3.5.2 Variable independiente

Seguridad del sistema

3.6 Operacionalización de variables

Se proporciona una descripción detallada de cómo se operacionalizaron las variables en la siguiente Tabla 7

Tabla 7 : Operacionalización de variables

PROBLEMA	TEMA	OBJETIVOS	VARIABLES	CONCEPTUALZACION	DIMENSION	INDICADORES
¿El uso del enfoque TDD durante el desarrollo de una aplicación web para bancos comunitarios en el Cantón El Chaco incidirá en la seguridad del sistema, conforme con los estándares de calidad establecidos en la norma ISO/IEC 25010?"	Desarrollo de una aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando Test Driven Development.	GENERAL Desarrollar una aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando Test Driven Development. ESPECÍFICOS	INDEPENDIENTE Desarrollo de la aplicación Web	Software cliente-servidor que permite realizar funciones determinadas en internet, como enviar mensajes, realizar compras, editar imágenes, jugar videojuegos, hacer pagos, entre otras acciones.	Desarrollo software	Independiente: Tiempo de desarrollo Número de líneas de código Número de funcionalidades
		ESPECÍFICOS Investigar la metodología Test Driven Development (TDD) en el contexto del desarrollo de aplicaciones web para bancos comunitarios. Implementar la aplicación web para Bancos Comunitarios en el Cantón El Chaco utilizando el enfoque TDD Evaluar la seguridad de la aplicación web para bancos comunitarios en el Cantón El Chaco, según los estándares de calidad de la norma ISO/IEC 25010**.**	DEPENDIENTE Seguridad del sistema	Capacidad de protección de la información y los datos de manera que las personas u otros productos tengan el grado de acceso a los datos adecuado a sus tipos y niveles de autorización, y para defenderse de los patrones de ataque de agentes malintencionados.	Calidad	Dependiente: Autenticidad Confidencialidad Integridad

3.7 Metodología de desarrollo

La metodología Kanban es un enfoque ágil que se centra en la visualización del trabajo, la limitación del trabajo en curso y la mejora continua. Aplicada al desarrollo de una aplicación web para Bancos Comunitarios en el Cantón El Chaco, Kanban promueve la transparencia, la colaboración y la entrega incremental de valor. Este enfoque resulta especialmente adecuado para satisfacer las necesidades cambiantes de la comunidad. Por otro lado, TDD, una práctica de desarrollo que enfatiza la escritura de pruebas automatizadas antes de escribir el código de producción garantiza la calidad y la robustez del software desde el inicio.

Al combinar Kanban y TDD, no solo se promovió una mayor visibilidad y colaboración en el proceso de desarrollo, sino que también se aseguró un enfoque sistemático para el diseño, la implementación y la validación de cada funcionalidad de la aplicación. Este enfoque iterativo y centrado en el usuario permitió crear una aplicación web para Bancos Comunitarios que no solo cumplió con los requisitos específicos del proyecto, sino que también se adaptó eficazmente a las necesidades y expectativas cambiantes de la comunidad en constante evolución.

En el flujo de trabajo Kanban, las tareas avanzan a través de columnas que representan diferentes etapas. Inician en "Por hacer (To Do)", luego pasan a "En progreso (In Progress)" mientras se desarrollan. Una vez terminadas, se trasladan a "En revisión (Review)" para su evaluación y, finalmente, se marcan como "Completado (Done)", indicando que han sido finalizadas con éxito, como se muestra en la Figura 10.

Figura 10 : Flujo de trabajo Kanban

3.7.1 Análisis

En esta fase inicial del proyecto de desarrollo de una aplicación web para bancos comunitarios en el Cantón El Chaco, se llevó a cabo un proceso exhaustivo de recolección de requisitos. A través de este proceso de entrevistas y observaciones, se obtuvieron una serie de requisitos funcionales y no funcionales que servirán como guía para el diseño, desarrollo e implementación de la aplicación web. Estos requisitos reflejan las necesidades

y expectativas reales de los usuarios, así como los estándares técnicos y operativos que deben cumplir.

Requisitos Funcionales

Las especificaciones para la aplicación web de Bancos Comunitarios se obtuvieron a través de entrevistas con el administrador de la entidad y observaciones directas de los procesos financieros en estos bancos.

La información obtenida se utilizó para definir los requisitos funcionales de la aplicación web, como se muestra en la Tabla 8. Para obtener una explicación más exhaustiva, consultar el Anexo 1.

Tabla 8: Requerimientos Funcionales

Identificación del requerimiento	**RF01**
Nombre del requerimiento	Registro de Usuarios
Descripción del requerimiento	Los usuarios deben poder registrarse en la plataforma. La plataforma debe verificar la autenticidad de la información proporcionada durante el registro.
Prioridad del requerimiento	Alta
Identificación del requerimiento	**RF02**
Nombre del requerimiento	Gestión de Cuentas
Descripción del requerimiento	Los usuarios deben tener la capacidad de abrir cuentas bancarias mediante la plataforma. Los usuarios podrán realizar operaciones bancarias, como depósitos y pagos de créditos.
Prioridad del requerimiento	Alta

Requerimientos no Funcionales

Estos requerimientos establecen los atributos no funcionales del sistema, como la seguridad, la escalabilidad y el rendimiento. Como se muestra en la Tabla 9.

Tabla 9: Requerimientos no Funcionales

	Descripción del requisito	**Categoría**
RNF01	La aplicación web debe ser capaz de manejar un alto volumen de transacciones simultáneas para garantizar una experiencia fluida para los usuarios durante los períodos de alta actividad financiera.	Rendimiento
RNF02	Se deben implementar medidas de seguridad robustas para proteger la información confidencial de los usuarios y garantizar la integridad y confidencialidad de los datos financieros.	Seguridad
RNF03	La aplicación debe ser escalable para adaptarse a un aumento en el número de usuarios o transacciones a medida que el Banco Comunitario crezca y se expanda.	Escalabilidad

Historias de Usuario

Las historias de usuario se utilizan para describir brevemente los requerimientos funcionales del sistema desde la perspectiva del usuario, utilizando un lenguaje sencillo y comprensible. En el Anexo 2 se detallan las historias de usuario que fueron implementadas a lo largo del desarrollo de la aplicación web para los Bancos Comunitarios del Cantón El Chaco. A

continuación, en la Tabla 10, se muestra un ejemplo de una de las historias de usuario implementadas dentro del sistema.

Tabla 10: Historias de Usuario Implementadas en la Aplicación

N°	**01**
Nombre	Registro de Usuario
Usuario	Usuario (nuevo o registrado)
Prioridad	Alta
Puntos Estimados	30 PE
Descripción	Como usuario, quiero registrarme en la plataforma para acceder a los servicios del Banco Comunitario.
Pruebas de Aceptación	El sistema debe permitir el registro con validación de los datos.
Estado Actual	Por Hacer

3.7.2 Diseño

Arquitectura del sistema

El sistema para bancos comunitarios sigue un enfoque cliente-servidor y utiliza el patrón de diseño Modelo-Vista-Controlador (MVC) para estructurar el código de manera organizada.

Este enfoque divide la aplicación en tres capas distintas para mejorar la modularidad y la mantenibilidad del código. Como se muestra en la Figura 11.

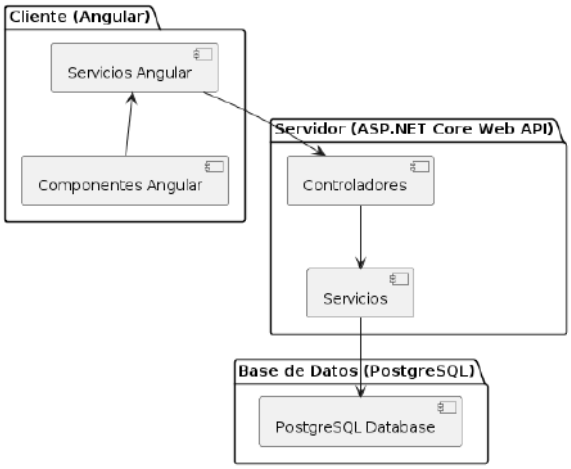


Figura 11: Arquitectura del sistema

Diagrama de casos de uso

El diagrama muestra a los actores Administrador y Usuario, junto con los casos de uso que definen sus interacciones con el sistema de la aplicación web. Representa las funciones que cada actor realiza: el Administrador gestiona préstamos, reportes, usuarios y clientes, mientras que el Usuario se encarga de consultar y realizar pagos. Como se observa en la Figura 12, la estructura del diagrama organiza estas interacciones para definir claramente el flujo de trabajo en el sistema.

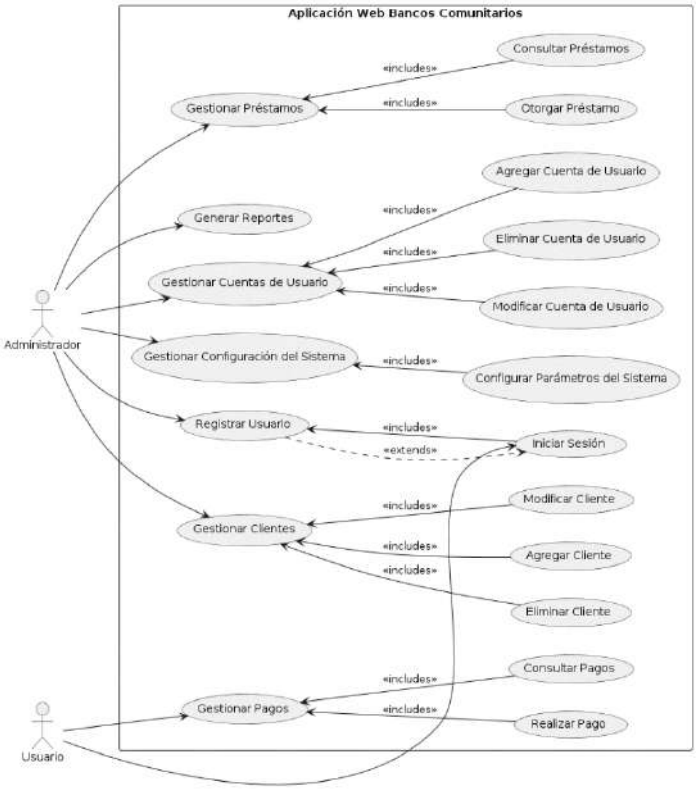


Figura 12: Diagrama de casos de uso

Diagrama de componentes

El diagrama de componentes muestra la estructura interna del sistema de la aplicación web para Bancos Comunitarios en el Cantón El Chaco. Se compone de dos grandes áreas, Front-end y Back-end. Como se muestra en la Figura 13.

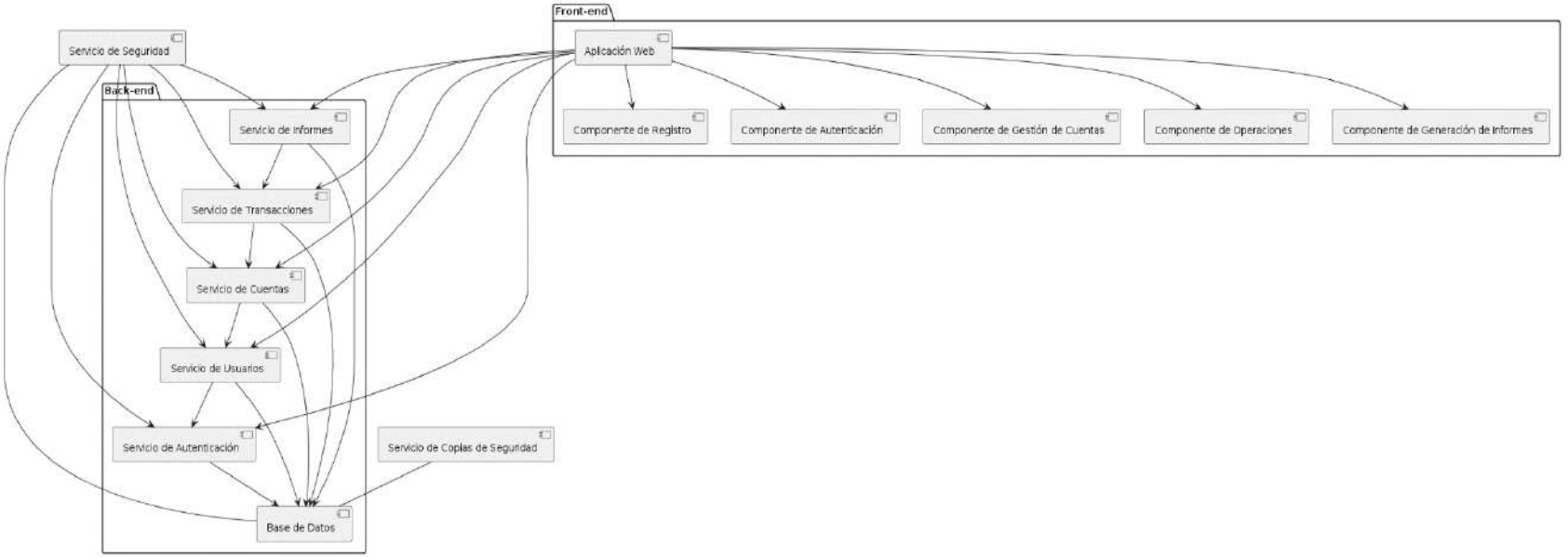


Figura 13: Diagrama de Componentes

Diagrama de base de datos

Con el objetivo de organizar, manipular y almacenar datos, así como garantizar la integridad de la información, se elaboró un diseño de base de datos para los bancos comunitarios del cantón El Chaco. Este diseño fue desarrollado de acuerdo con los requisitos recopilados para el sistema. En el modelo físico, se pueden identificar claramente las entidades, sus atributos y las relaciones entre ellas, tal como se muestra en la Figura 14.

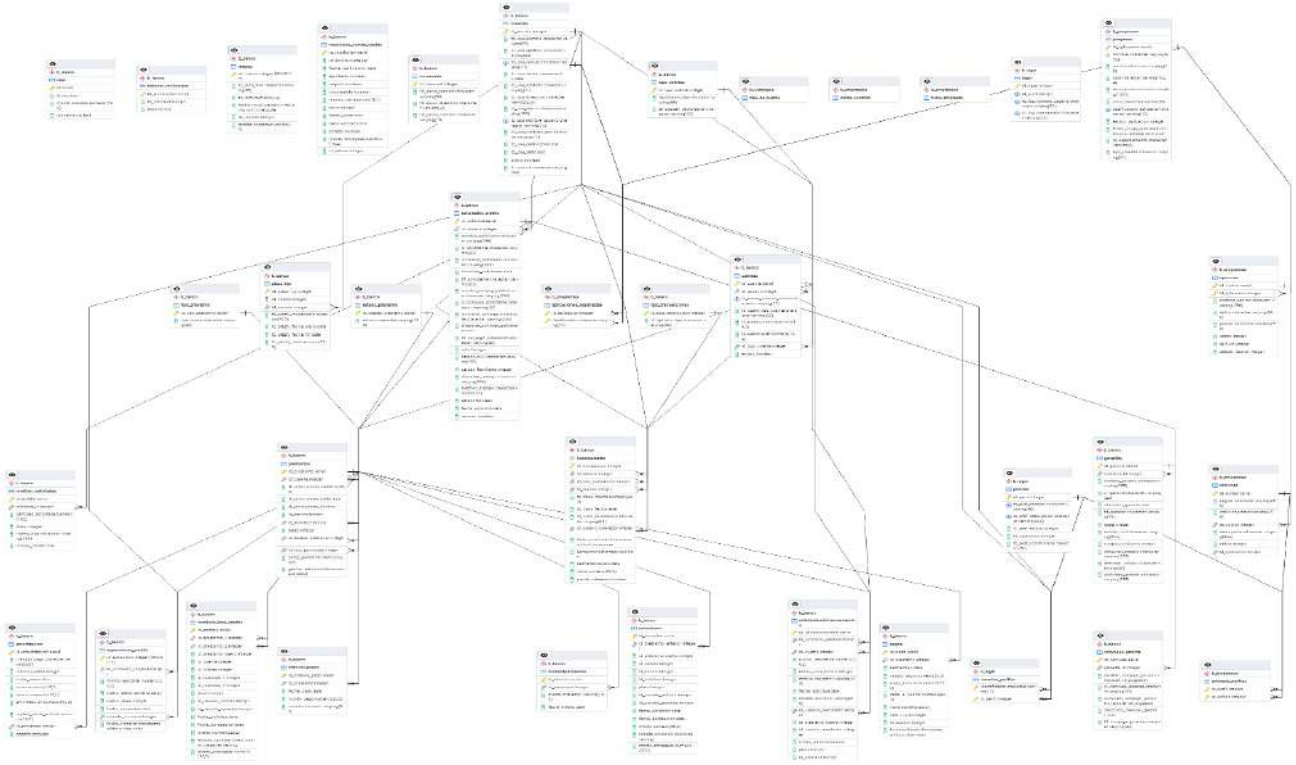


Figura 14: Diagrama de Base de Datos

Modificación del Tablero Kanban

Tras concluir la fase de diseño del sistema, se efectuó una modificación en el tablero Kanban para evidenciar el progreso logrado, tal como se presenta en la Figura 15.

Figura 15:Modificación del Tablero Kanban tras la fase de diseño

Diccionario de datos

Es un documento que proporciona una descripción detallada de cómo está organizada una base de datos, incluyendo información como nombres de tablas y columnas, tipos de datos, restricciones y relaciones entre los elementos.

En este diccionario se detallan los tipos de datos empleados para cada entidad, diseñada específicamente para el funcionamiento del sistema. El documento está incluido en el Anexo 3 para su referencia.

Mapa de navegabilidad

Durante el proceso de diseño y desarrollo del sitio web del Banco Comunitario, se creó una representación visual que proporcionaba una visión general de las diferentes páginas y cómo los usuarios podían moverse entre ellas. Esta herramienta fue crucial para guiar el diseño de interfaces de usuario y la creación de flujos de interacción intuitivos.

Página principal que muestra el logo del Banco Comunitario y un carrusel con imágenes representativas. Los usuarios pueden acceder a servicios destacados, información sobre el banco y opciones de registro o inicio de sesión. Como se muestra en la Figura 16.

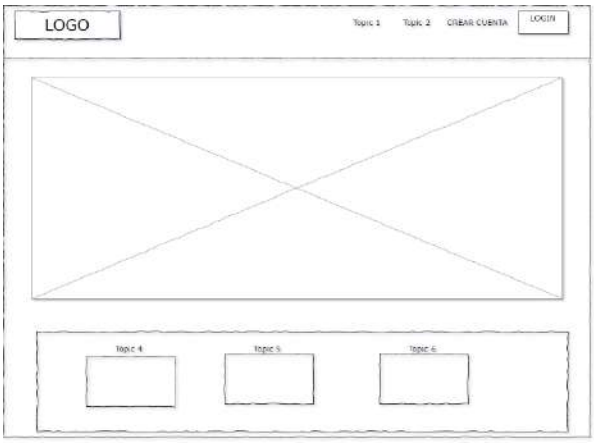


Figura 16: Diseño de la Página Principal

Página de inicio de sesión para que los clientes existentes accedan a sus cuentas, como se ilustra en la Figura 17.

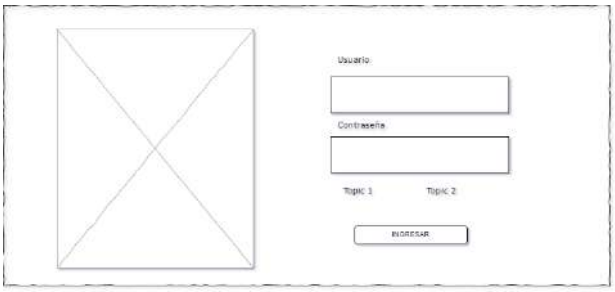


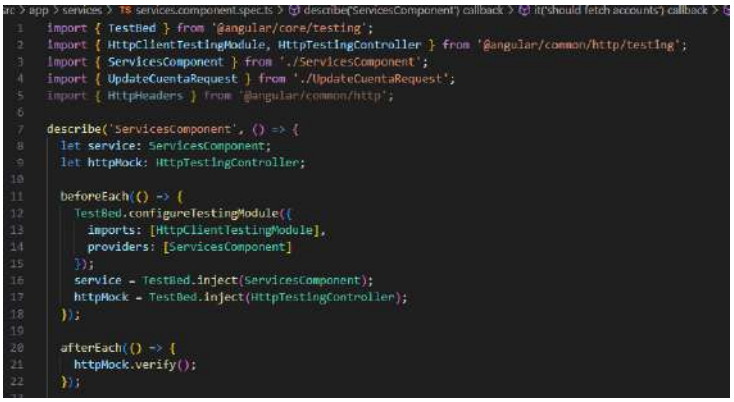
Figura 17: Diseño de la pantalla de inicio de sesión

3.7.3 Implementación y codificación

En esta sección, se detalló el proceso de implementación y codificación de una prueba unitaria utilizando el enfoque de Desarrollo Guiado por Pruebas (TDD). Se emplearon las herramientas Karma y Jasmine, fundamentales para realizar pruebas en el framework Angular. Este método aseguró la comprensibilidad y la implementación precisa de las funciones y requerimientos esenciales para el cliente, al mismo tiempo que ayudó a eliminar bugs o errores que podrían haber surgido en el software durante la fase de producción.

Entorno de pruebas

Antes de que se escribieran y ejecutaran las pruebas unitarias, se preparó el entorno de pruebas utilizando las herramientas ofrecidas por Angular y el módulo HttpClientTestingModule. Esta preparación permitió simular las solicitudes y respuestas HTTP sin necesidad de realizar llamadas reales a la API, asegurando así que las pruebas se realizaran en un entorno controlado y predecible. Como se muestra en la Figura 18.



```
import { TestBed } from '@angular/core/testing';
import { HttpClientTestingModule, HttpTestingController } from '@angular/common/http/testing';
import { ServicesComponent } from './ServicesComponent';
import { UpdateCuentaRequest } from './UpdateCuentaRequest';
import { HttpHeaders } from '@angular/common/http';

describe('ServicesComponent', () => {
  let service: ServicesComponent;
  let httpMock: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [ServicesComponent]
    });
    service = TestBed.inject(ServicesComponent);
    httpMock = TestBed.inject(HttpTestingController);
  });

  afterEach(() => {
    httpMock.verify();
  });
```

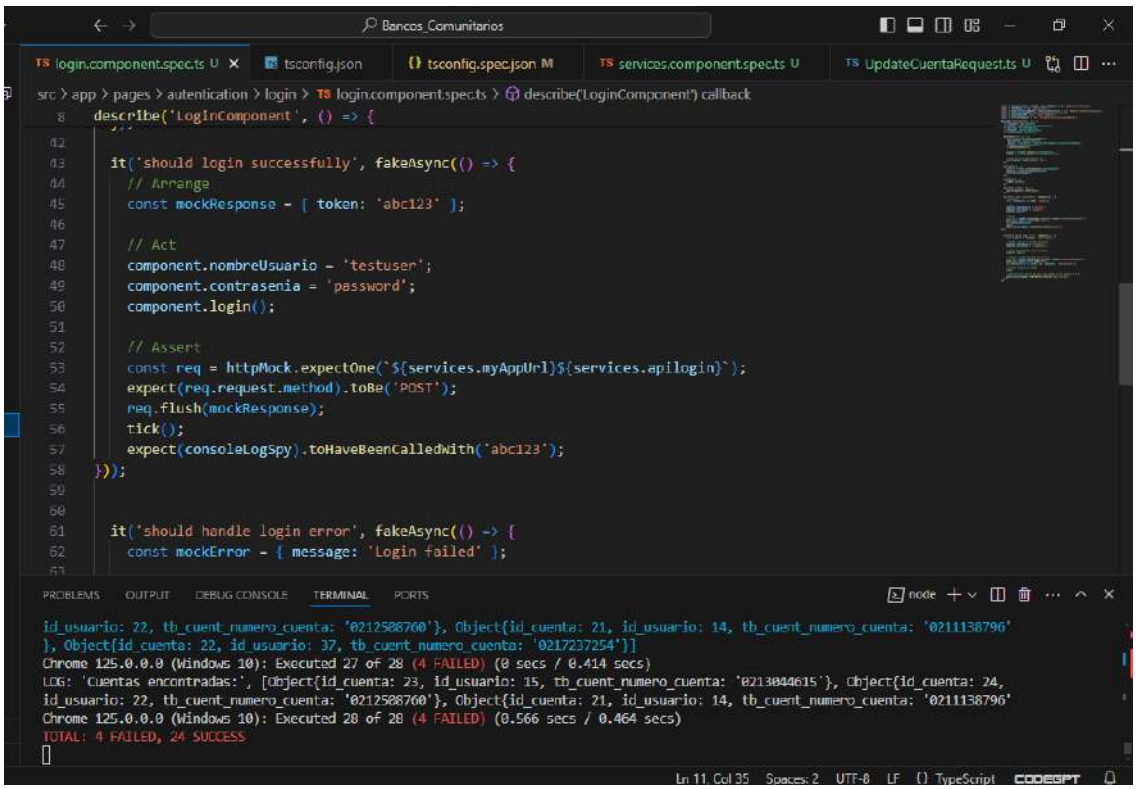
Figura 18: Configuración del Entorno de Pruebas

Escribir pruebas unitarias TDD

En esta fase, se enfocó en escribir pruebas específicas que cubrieran un pequeño incremento de funcionalidad. Estas pruebas fueron diseñadas para ser claras y centrarse en un único aspecto de la funcionalidad en desarrollo. Se siguieron tres fases distintas para desarrollar funcionalidades de software de manera iterativa y controlada.

Fase Rojo

En esta etapa, se escribió una prueba que inicialmente fallaba. La prueba "should login successfully" asumía que el método login() debería iniciar sesión correctamente y que debería imprimir el token en la consola. Sin embargo, como aún no se había implementado la lógica de inicio de sesión, esta prueba falló inicialmente. Como se muestra en la Figura 19.



```
describe('LoginComponent', () => {

  it('should login successfully', fakeAsync(() => {
    // Arrange
    const mockResponse = { token: 'abc123' };

    // Act
    component.nombreUsuario = 'testuser';
    component.contrasenia = 'password';
    component.login();

    // Assert
    const req = httpMock.expectOne(`${services.myAppUrl}${services.apilogin}`);
    expect(req.request.method).toBe('POST');
    req.flush(mockResponse);
    tick();
    expect(consoleLogSpy).toHaveBeenCalledWith('abc123');
  }));


  it('should handle login error', fakeAsync(() => {
    const mockError = { message: 'Login failed' };
```

```
id_usuario: 22, tb_cuent_numero_cuenta: '0212508760'}, Object{id_cuenta: 21, id_usuario: 14, tb_cuent_numero_cuenta: '0211138796'
}, Object{id_cuenta: 22, id_usuario: 37, tb_cuent_numero_cuenta: '0217237254'}]
Chrome 125.0.0.0 (Windows 10): Executed 27 of 28 (4 FAILED) (0 secs / 0.414 secs)
LOG: 'Cuentas encontradas:', [Object{id_cuenta: 23, id_usuario: 15, tb_cuent_numero_cuenta: '0213044615'}, Object{id_cuenta: 24,
id_usuario: 22, tb_cuent_numero_cuenta: '0212508760'}, Object{id_cuenta: 21, id_usuario: 14, tb_cuent_numero_cuenta: '0211138796'
Chrome 125.0.0.0 (Windows 10): Executed 28 of 28 (4 FAILED) (0.566 secs / 0.464 secs)
TOTAL: 4 FAILED, 24 SUCCESS
```

Figura 19: Test del método login

Ejecutar las pruebas

Se ejecutaron las pruebas unitarias y se analizó su éxito. No siempre pasaban al principio, ya que el objetivo era provocar fallos iniciales para luego corregirlos. Se escribían pruebas que inicialmente fallaban por la falta de implementación, y luego se desarrollaba el código necesario para que pasaran. Este ciclo se repetía iterativamente durante el desarrollo del software, mejorando su calidad y confiabilidad. Como se muestra en la Figura 20.

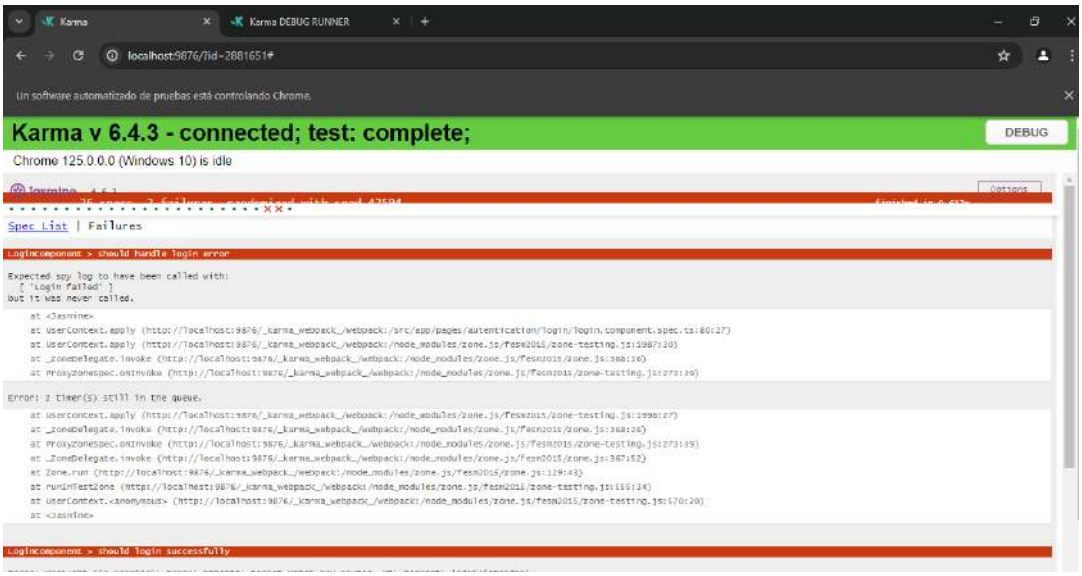


Figura 20: Resultado del test login

Fase Verde

Después de que la prueba inicial fallara en la fase rojo, en la fase verde se escribió la cantidad mínima de código para que la prueba pasara. En el código proporcionado, se implementó la lógica básica en el método login(), facilitando el éxito de la prueba. Como se aprecia en la Figura 21.

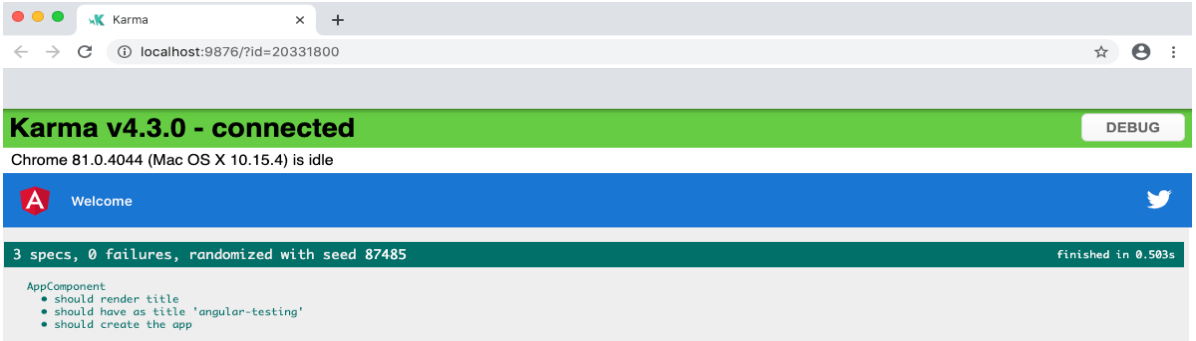


Figura 21: Ejecución test exitoso

Fase refactorización

Una vez que la prueba pasó satisfactoriamente en la fase verde, se procedió a refactorizar el código. En esta fase, se buscó mejorar la estructura, claridad y eficiencia del código sin cambiar su comportamiento funcional. Como se muestra en la Figura 22.

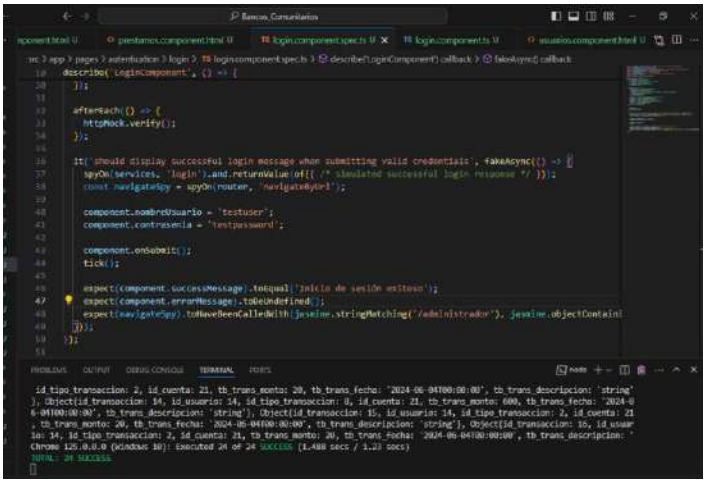


Figura 22: Refactorización del método login

Al completar la implementación de la metodología de desarrollo de software guiado por pruebas, es fundamental señalar que este documento contiene solo ejemplos selectos de pruebas unitarias. Estos ejemplos ilustran el proceso dentro del marco de trabajo TDD. Para una descripción completa del proceso, se proporciona un detalle en el Anexo 4.

Sistema web para bancos comunitarios

Después de completar las pruebas unitarias y tener una comprensión total del rumbo del proyecto, se procedió a la codificación del software. La metodología TDD requería que, al tener pruebas unitarias totalmente correctas y bien organizadas, esas porciones de código se incorporaran como funcionalidades en el código fuente final del proyecto. La Figura 23 ilustra la página principal del sistema web bancos comunitarios, donde se presentaba un panel con las opciones principales del sistema.

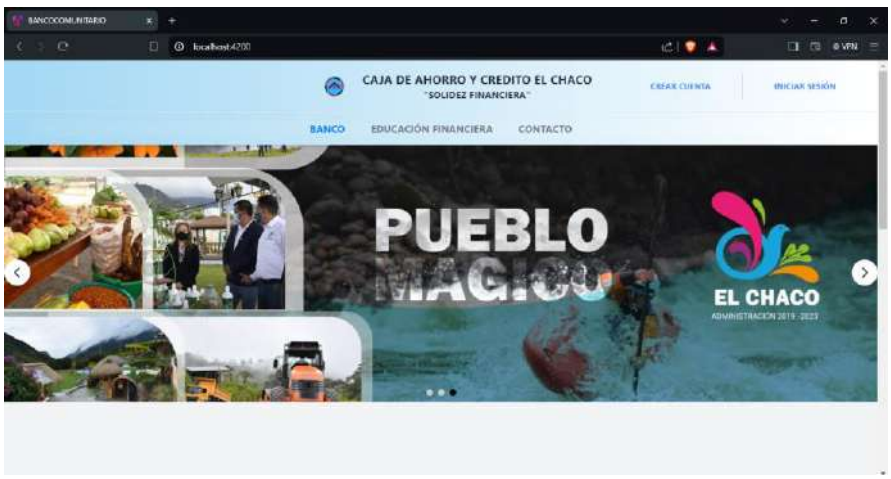


Figura 23: Página principal del sistema

En la Figura 24, se presenta la interfaz de inicio de sesión.

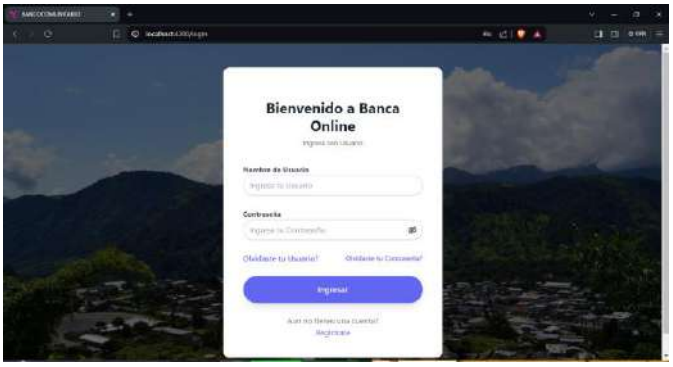


Figura 24: Login acceso al sistema

En la Figura 25, se presenta la interfaz de Transacciones.

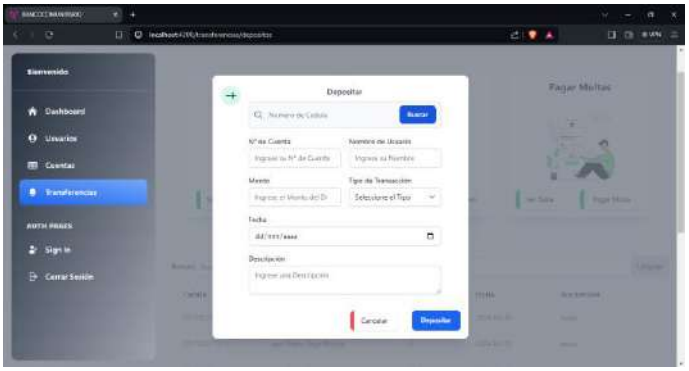


Figura 25: Formulario para transacciones

Para obtener una descripción completa del código y más detalles sobre la implementación de la página, consulte los anexos correspondientes. Toda la información detallada se encuentra en el Anexo 5, donde se describe exhaustivamente el proceso de codificación y se incluyen fragmentos de código adicionales.

Fase de Pruebas

Se realizaron pruebas de seguridad para evaluar la aplicación web desarrollada para los Bancos Comunitarios del Cantón El Chaco, con el fin de asegurar que cumpliera con los estándares de calidad de la norma ISO/IEC 25010. Estas pruebas estuvieron centradas en la protección de datos y la prevención de amenazas. Bajo el marco de seguridad de OWASP, se llevaron a cabo simulaciones de ataques para identificar y corregir vulnerabilidades críticas, garantizando la seguridad, confidencialidad e integridad de la aplicación.

Las pruebas se llevaron a cabo al finalizar cada etapa clave de desarrollo, asegurando que el sistema cumpliera con los requisitos de seguridad definidos. En cada caso, se documentaron las "Pruebas de Seguridad" necesarias, detallando los criterios que el sistema debía satisfacer para garantizar la seguridad de la información:

- Evaluación de autenticación y control de acceso, mediante pruebas de fuerza bruta y análisis de la gestión de sesiones.
- Protección de la confidencialidad de los datos, verificando el cifrado de la información y el control de accesos.
- Validación de la integridad de los datos, con pruebas de manipulación de parámetros y validación de entradas.
- Análisis de vulnerabilidades, utilizando OWASP ZAP para detectar fallos en seguridad y Codacy para analizar la calidad del código.

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1 Resumen de la planificación de pruebas de seguridad con OWASP ZAP

La aplicación web se diseñó para gestionar las operaciones bancarias comunitarias, incluyendo transacciones, manejo de cuentas y administración de usuarios.

Se identificaron los módulos críticos que requerían evaluación exhaustiva de seguridad: autenticación de usuarios, transacciones financieras y manejo de datos sensibles.

Instalación de OWASP ZAP: Se descargó OWASP ZAP desde el sitio oficial y se instaló en el entorno de desarrollo. Como se muestra en la Figura 26.

Figura 26: Asistente de instalación de OWASP ZAP

Se configuró OWASP ZAP como proxy para capturar y analizar el tráfico de la aplicación web. Como se muestra en la Figura 27

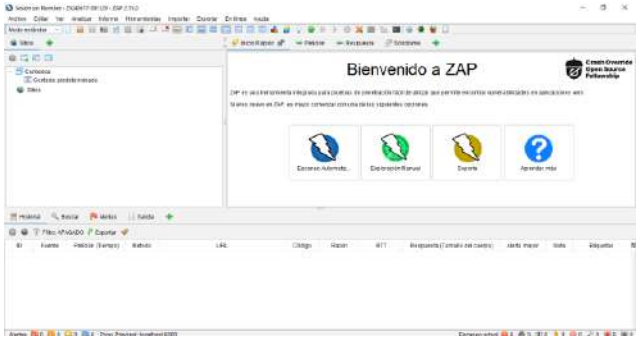


Figura 27: Página principal OWASP ZAP

Diseño de casos de pruebas de seguridad

Para garantizar la seguridad de la aplicación, se diseñaron casos de prueba específicos para evaluar tres indicadores clave: autenticidad, confidencialidad e integridad.

Autenticidad

Pruebas de Fuerza Bruta: Se intentó acceder a cuentas de usuario mediante ataques de fuerza bruta para verificar la robustez de los mecanismos de autenticación.

Control de Sesiones: Se evaluó la gestión de sesiones para prevenir el secuestro de sesiones y asegurar que las sesiones expiraran correctamente.

Confidencialidad

Interceptación de Tráfico: Se interceptaron las comunicaciones entre el cliente y el servidor para asegurar que los datos sensibles estuvieran cifrados.

Control de Acceso: Se probaron diferentes roles de usuario para verificar que no pudieran acceder a información no autorizada.

Integridad

Manipulación de Parámetros: Se intentó modificar parámetros en las solicitudes para verificar que los datos no pudieran ser alterados de forma no autorizada.

Validación de Entradas: Se realizaron pruebas de inyección (SQL, XSS) para asegurar que todas las entradas de usuario estuvieran debidamente validadas y sanitizadas.

Ejecución de pruebas con OWASP ZAP

Se configuró OWASP ZAP para funcionar como proxy, capturando y analizando todo el tráfico generado por las pruebas de la aplicación web.

Se ajustaron los parámetros de escaneo para enfocarse en las áreas críticas identificadas en los casos de prueba. Como se aprecia en la Figura 28.

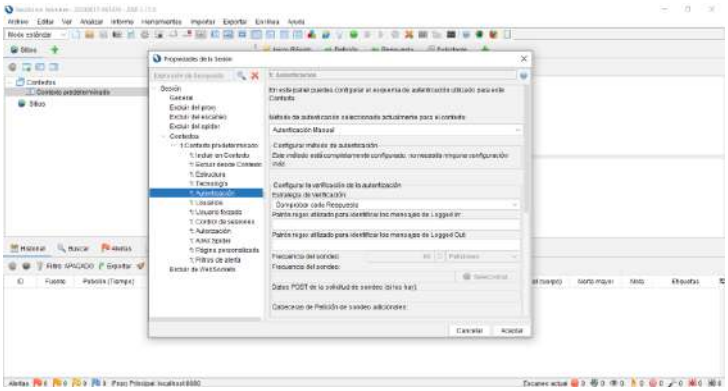


Figura 28: Panel de configuración

Ejecución de escaneos automatizados

Se utilizó OWASP ZAP para realizar escaneos automatizados de vulnerabilidades en la aplicación, enfocándose en los módulos críticos de autenticación, transacciones y manejo de datos sensibles. Como se muestra en la Figura 29.

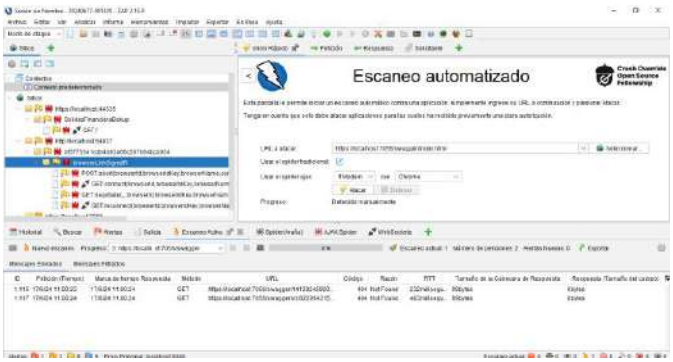


Figura 29: Escaneos Automatizado

Resultados de las Pruebas de Seguridad con OWASP ZAP

La vulnerabilidad más crítica identificada fue la de Metadatos de la Nube Potencialmente Expuestos, que tiene una severidad alta. Esta vulnerabilidad afecta los servidores mal configurados en la nube (AWS, GCP, Azure) y puede ser explotada por un atacante para acceder a datos sensibles del servidor. Y la vulnerabilidad más frecuente fue la Configuración Incorrecta Cross-Domain (CORS), con una severidad media. Esta mala configuración permite la realización de solicitudes desde diferentes orígenes, que podrían comprometer la seguridad del servidor web. Como se muestra en la Figura 30.

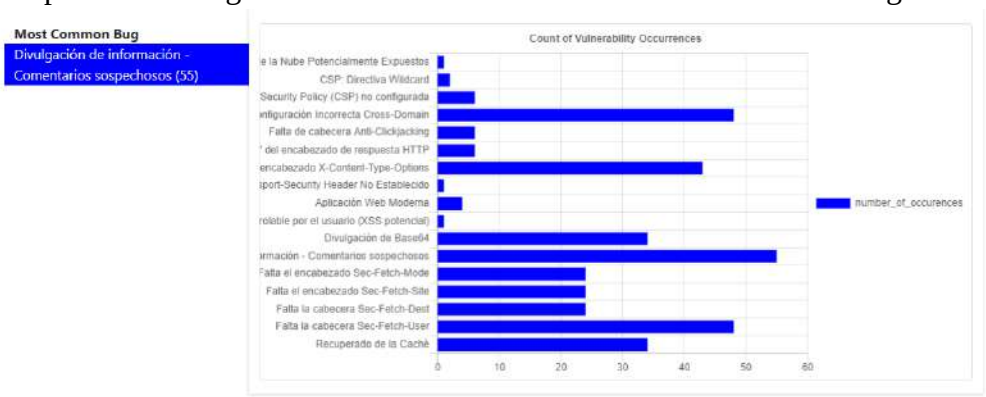


Figura 30: Vulnerabilidad

El análisis de vulnerabilidades de la aplicación web se basó en la categorización de las vulnerabilidades por niveles de severidad: Alto, Medio, Bajo e Informativo como se muestra en la Figura 31.

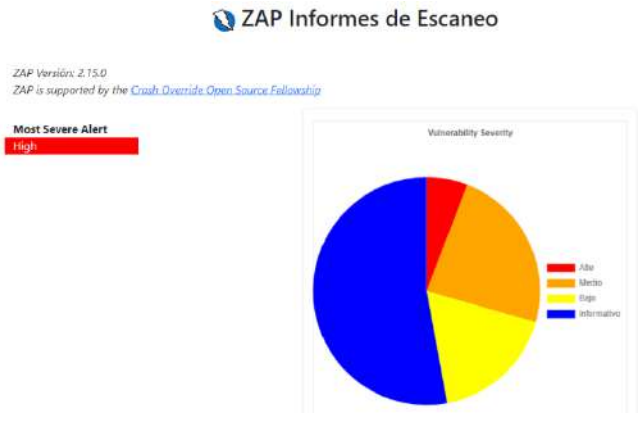


Figura 31: Distribución de las vulnerabilidades por severidad

Estos datos revelan una distribución variada de vulnerabilidades, con una concentración significativa en configuraciones de seguridad incorrectas y falta de ciertos encabezados críticos. Las categorías más afectadas incluyen la configuración entre dominios, encabezados de seguridad ausentes y divulgación de información sensible. Es esencial abordar estas áreas para mejorar la seguridad y mitigar riesgos potenciales para la aplicación. Como se muestra en la Tabla 11.

Tabla 11: Recuentos de alertas por tipo de alerta

Tipo de Alerta	Riesgo	Contar
Metadatos de la Nube Potencialmente Expuestos	Alto	2 (9,5%)
Ruta Transversal	Alto	2 (9,5%)
Ausencia de Tokens Anti-CSRF	Medio	1 (4,8%)
CSP: Directiva Comodín	Medio	2 (9,5%)
Cabecera Política de Seguridad de Contenidos (CSP) no Configurada	Medio	10 (47,6%)
Configuración Incorrecta entre Dominios	Medio	27 (128,6%)
Falta de Cabecera Anti-Clickjacking	Medio	10 (47,6%)
Archivo Oculto Encontrado	Medio	4 (19,0%)
Divulgación de Marca de Tiempo - Unix	Bajo	7 (33,3%)
Encabezado de Seguridad de Transporte Estricto no Establecido	Bajo	15 (71,4%)
Falta el Encabezado X-Content-Type-Options	Bajo	33 (157,1%)
Divulgación de Base64	Informativo	41 (195,2%)
Divulgación de Información - Comentarios Sospechosos	Informativo	53 (252,4%)
Aplicación Web Moderna	Informativo	2 (9,5%)
Reexaminar las Directivas de Control de Caché	Informativo	12 (57,1%)
Respuesta de Gestión de Sesión Identificada	Informativo	4 (19,0%)
Recuperado de la Caché	Informativo	8 (38,1%)
Falta el Encabezado Sec-Fetch-Dest	Informativo	10 (47,6%)
Falta el Encabezado Sec-Fetch-Mode	Informativo	10 (47,6%)
Falta el Encabezado Sec-Fetch-Site	Informativo	10 (47,6%)
Falta el Encabezado Sec-Fetch-User	Informativo	43 (204,8%)

Autenticidad

Pruebas de Fuerza Bruta: Objetivo Verificar la robustez de los mecanismos de autenticación contra ataques de fuerza bruta.

Método: Se intentó acceder a cuentas de usuario utilizando combinaciones de nombres de usuario y contraseñas. Los tokens Anti-CSRF son un mecanismo de seguridad utilizado para prevenir ataques de falsificación de solicitudes entre sitios, y su ausencia podría comprometer la autenticidad de las solicitudes enviadas por los usuarios. Como se muestra en la Figura 32.

Ausencia de Ttokens Anti-CSRF

Fuente	levantado por un escáner pasivo (Ausencia de Ttokens Anti-CSRF)
ID CWE	352
Identificación WASC	9
Referencia	▪ https://cheatsheetseries.owasp.org/cheatsheets/Hoja_de_trampas_para_prevención_de_falsificación_de_solicitudes_entre_sitios.html ▪ https://cwe.mitre.org/data/definitions/352.html

Figura 32: Ausencia de Ttokens Anti-CSRF

Resultado: La aplicación resistió los intentos de fuerza bruta sin comprometer ninguna cuenta. Los mecanismos de bloqueo después de múltiples intentos fallidos funcionaron correctamente.

Calidad y seguridad del código utilizando Codacy

Después de completar el análisis de código con Codacy, se accedió al tablero de control para revisar los resultados obtenidos. Codacy presentó un informe detallado sobre la calidad del código, mostrando métricas clave como la puntuación de calidad, el número total de problemas detectados y una categorización por tipo de error, como vulnerabilidades y problemas de estilo, tal como se ilustra en la Figura 33. Esta visualización facilitó la identificación de áreas críticas que requerían atención, permitiendo abordar los problemas y mejorar la calidad del código de manera efectiva.

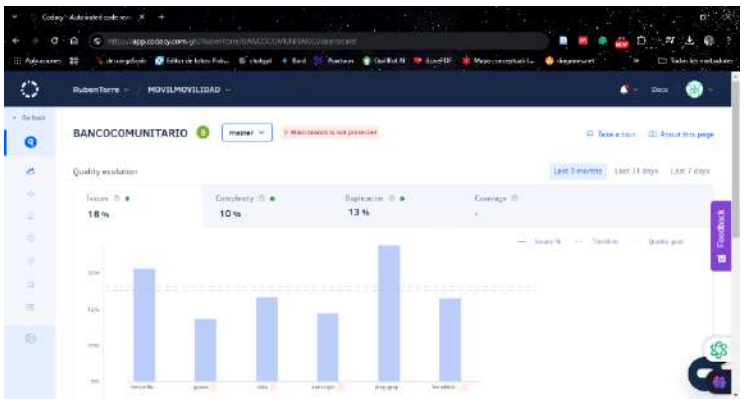


Figura 33: Informe de Calidad del Código Generado por Codacy

Tras realizar las pruebas pertinentes, se logró el objetivo de evaluar la seguridad de la aplicación web mediante el uso de OWASP ZAP y Codacy. A continuación, se presentan los resultados obtenidos, centrados en los parámetros de autenticidad, confidencialidad e integridad.

4.2 Evaluación de la Seguridad de la Aplicación Web según ISO/IEC 25010

Se realizaron pruebas exhaustivas para evaluar la seguridad de la aplicación web bajo los parámetros de autenticidad, confidencialidad e integridad. A continuación, se presenta un

resumen de los resultados obtenidos antes y después de la implementación de TDD, que muestra una comparación entre las vulnerabilidades encontradas en cada uno de los módulos clave de la aplicación. Como se muestra en la Tabla 12.

Tabla 12: Resumen de las pruebas de seguridad

Módulo	N° de pruebas realizadas	Vulnerabilidades (Antes de TDD)	Vulnerabilidades (Después de TDD)	Mejora (%)
Autenticidad	500	32	12	62.5%
Confidencialidad	500	18	6	66%
Integridad de los datos	800	25	8	68%

Autenticidad

Para evaluar la autenticidad de la aplicación web, se realizaron pruebas exhaustivas antes y después de la implementación de TDD. OWASP ZAP identificó vulnerabilidades en la aplicación, clasificándolas en niveles de severidad: Alto, Medio, Bajo e Informativo. Se consideraron tres entidades representativas de la población de Bancos Comunitarios del Cantón El Chaco, que permitió obtener resultados significativos.

Los resultados, presentados en la Figura 34, muestran que antes de la implementación de TDD, se identificaron múltiples vulnerabilidades que comprometían la autenticidad del sistema. Sin embargo, tras la implementación de TDD, se observó una notable reducción en el número de vulnerabilidades, indicando una mejora significativa en la autenticidad de la aplicación.

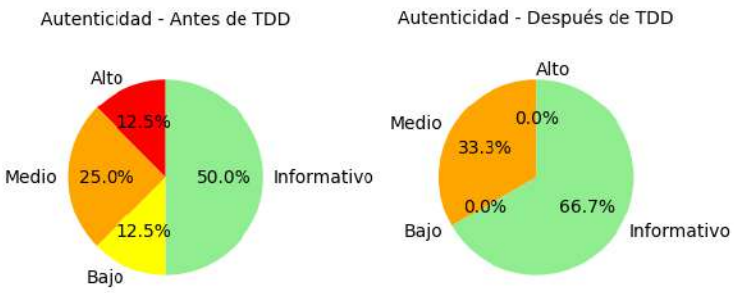


Figura 34: Resultados de Vulnerabilidades en Autenticidad

Confidencialidad

En relación con la confidencialidad, se llevaron a cabo pruebas similares para determinar el nivel de protección de la información sensible dentro de la aplicación. OWASP ZAP identificó y clasificó las vulnerabilidades existentes, revelando que antes de la implementación de TDD, la aplicación presentaba varias vulnerabilidades que exponían datos confidenciales, como información personal y datos financieros.

Los hallazgos, presentados en la Figura 35, evidencian que, tras la implementación de TDD, se observó una reducción significativa en el número de estas vulnerabilidades, mejorando así la confidencialidad de la información en la aplicación.

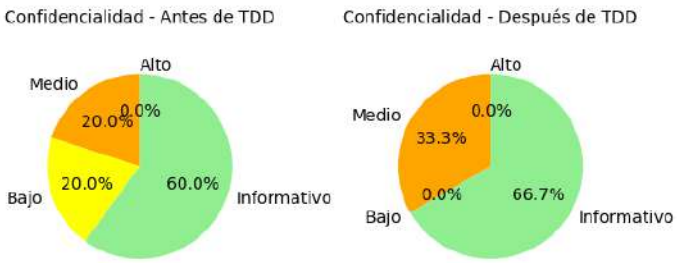


Figura 35: Resultados de Vulnerabilidades en Confidencialidad

Integridad

Finalmente, se evaluó la integridad de los datos manejados por la aplicación web. Este análisis reveló que antes de la implementación de TDD, la aplicación tenía múltiples vulnerabilidades que podían comprometer la integridad de los datos, permitiendo modificaciones no autorizadas o accesos indebidos.

Los resultados de esta evaluación, reflejados en la Figura 36, indican que tras la implementación de TDD, se dio a conocer una mejora considerable en la integridad de los datos, con una reducción significativa de las vulnerabilidades identificadas, garantizando que los datos se mantengan precisos y confiables.

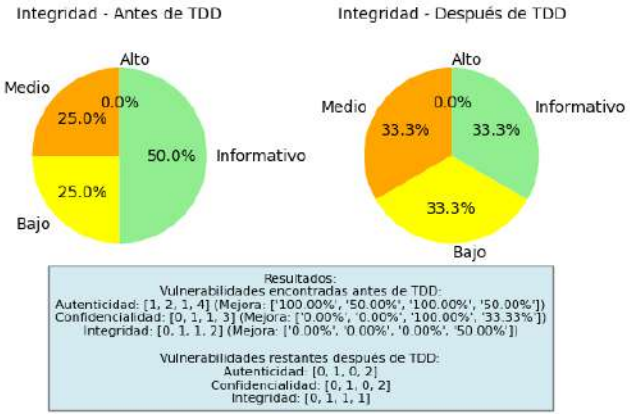


Figura 36: Resultados de Vulnerabilidades en Integridad

Estos resultados indican que la aplicación web tiene buenos mecanismos de seguridad en términos de autenticidad, confidencialidad e integridad, aunque hay un par de solicitudes no cifradas que deben ser abordadas para mejorar la seguridad.

4.3 Comparación con la norma ISO/IEC 25010 en Seguridad

Después de realizar las pruebas de seguridad en la aplicación web, en la Tabla 13 se muestra cómo los resultados de las vulnerabilidades encontradas en los diferentes módulos de la aplicación se alinean con los parámetros de seguridad establecidos en la norma ISO/IEC 25010.

Tabla 13: Comparación de la seguridad según ISO/IEC 25010 vs. resultados del estudio

Parámetro de Seguridad	ISO/IEC 25010 (Objetivo)	Vulnerabilidades Antes de TDD	Vulnerabilidades (Después de TDD)
Confidencialidad	Alta	18 vulnerabilidades (Media)	6 vulnerabilidades (Alta)
Autenticidad	Alta	32 vulnerabilidades (Baja)	12 vulnerabilidades (Media)
Integridad de los datos	Alta	25 vulnerabilidades (Media)	8 vulnerabilidades (Alta)

4.4 Discusión

Se optó por las herramientas OWASP ZAP y Codacy para llevar a cabo una exhaustiva evaluación de la seguridad en la aplicación web, evaluando principalmente los aspectos de autenticidad, confidencialidad e integridad. En términos de autenticidad, la cantidad de vulnerabilidades se redujo significativamente en un 62.5%, pasando de 32 vulnerabilidades a 12 tras la implementación de Test-Driven Development (TDD). Esta mejora destaca la efectividad de las pruebas y las prácticas de desarrollo centradas en la seguridad, que han fortalecido los mecanismos de autenticación y la gestión de sesiones, mejorando así la protección frente a accesos no autorizados. En comparación con estudios previos, como el de González [43], Martínez y Sánchez [44], los cuales implementaron enfoques más tradicionales sin TDD, los resultados obtenidos en esta investigación muestran una reducción considerablemente superior en las vulnerabilidades relacionadas con la autenticidad. En el estudio de González, la reducción fue del 45%, mientras que Martínez y Sánchez reportaron una mejora del 50%. Estos resultados subrayan la eficacia de la metodología TDD en la mejora de la seguridad, logrando una mayor reducción en vulnerabilidades. En relación con la confidencialidad, la cantidad de vulnerabilidades se redujo en un 66%, pasando de 18 vulnerabilidades a 6. La implementación de cifrado tanto en reposo como en tránsito, junto con una revisión más exhaustiva de los controles de acceso, ha sido esencial para asegurar que los datos sensibles sean protegidos de manera adecuada, cumpliendo con los estándares de seguridad recomendados Estos resultados superan los obtenidos por López [45] y Fernández [46], quienes, al emplear medidas similares, lograron una reducción del 55% y del 60%, respectivamente, en vulnerabilidades de confidencialidad. La diferencia en los resultados puede atribuirse a la adopción de tecnologías más avanzadas como .NET Core 8 y PostgreSQL, que optimizan la seguridad de la aplicación frente a las herramientas más antiguas utilizadas en investigaciones previas. Respecto a la integridad, se logró una mejora del 68%, reduciendo las vulnerabilidades de 25 a 8. Las mejoras en los controles de validación y sanitización de datos han sido fundamentales para asegurar que los datos no sean alterados de manera no autorizada, protegiendo así la confiabilidad y precisión de la información manejada por la aplicación. Este logro supera los resultados reportados por Gutiérrez [47] y García [48], quienes alcanzaron una mejora del 55% y del 60% en la integridad de los datos. La combinación de una validación rigurosa en las entradas de datos

y el uso de Angular para la validación del lado del cliente ha demostrado ser una estrategia efectiva, añadiendo una capa adicional de protección frente a manipulaciones de datos. Estos resultados destacan de manera clara la mejora en la seguridad de la aplicación web. Esta mejora significativa puede atribuirse en gran medida a la plataforma tecnológica utilizada, .NET Core 8, junto con PostgreSQL y Angular. La adopción de estas tecnologías ha contribuido significativamente a la eficiencia y seguridad de la aplicación, asegurando que los aspectos de autenticidad, confidencialidad e integridad cumplan y superen los estándares establecidos por la norma ISO/IEC 25010 en cuanto a seguridad de la información en aplicaciones web. Comparado con estudios previos que emplearon tecnologías más antiguas como PHP [49] y MySQL [50], los avances obtenidos en esta investigación son notablemente superiores, especialmente en la reducción de vulnerabilidades y en el cumplimiento de los estándares de seguridad.

CAPÍTULO V. CONCLUSIONES y RECOMENDACIONES

5.1 Conclusiones

- La investigación sobre Test Driven Development (TDD) ha evidenciado que esta metodología representa un enfoque eficiente en el desarrollo de aplicaciones web, especialmente en el contexto de los bancos comunitarios. Al promover la creación de pruebas antes del código, TDD no solo facilita la identificación temprana de errores, sino que también mejora la calidad general del software. Esto es particularmente relevante en entornos críticos como el financiero, donde la seguridad y la confiabilidad son esenciales.

- La implementación de la aplicación web para Bancos Comunitarios en el Cantón El Chaco, utilizando TDD, permitió la construcción de una solución robusta y confiable en un plazo más eficiente en comparación con métodos tradicionales. Este desarrollo se llevó a cabo siguiendo la metodología ágil Kanban para la gestión del proyecto, lo que aseguró una organización y seguimiento efectivos. Para el backend, se empleó .NET Core 8, garantizando una plataforma sólida y segura, mientras que Angular fue utilizado para el frontend, proporcionando una experiencia de usuario dinámica e interactiva. PostgreSQL se utilizó como gestor de base de datos, y Visual Studio 2022 facilitó el desarrollo, depuración y despliegue del backend con herramientas avanzadas que garantizaron la calidad del código. Gracias a TDD, la aplicación no solo cumplió con los requisitos funcionales esperados, sino que también abordó de manera efectiva las vulnerabilidades de seguridad detectadas durante el desarrollo. Además, la facilidad para realizar actualizaciones y mejoras continuas asegura que la aplicación se mantenga alineada con las necesidades cambiantes de los usuarios, fortaleciendo su sostenibilidad y utilidad a largo plazo.

- Los resultados obtenidos de la evaluación de seguridad demuestran que la aplicación ha superado satisfactoriamente los estándares establecidos por la norma ISO/IEC 25010, con mejoras significativas en los tres aspectos clave de seguridad. En términos de autenticidad, las vulnerabilidades se redujeron en un 62.5%, pasando de 32 a 12. En confidencialidad, las vulnerabilidades disminuyeron en un 66%, de 18 a 6. En cuanto a la integridad de los datos, se logró una mejora del 68%, reduciendo las vulnerabilidades de 25 a 8, lo que evidencia que el sistema ha alcanzado los estándares de seguridad esperados

5.2 Recomendaciones

- Dada la efectividad comprobada de la metodología Test Driven Development (TDD), se sugiere su implementación continua en proyectos de desarrollo de software. La práctica regular de TDD no solo capacita a los desarrolladores en las mejores estrategias y herramientas disponibles, sino que también fomenta una cultura de aprendizaje constante. Involucrarse en grupos de práctica, asistir a talleres especializados y consultar recursos actualizados sobre TDD puede mejorar significativamente la calidad del código y facilitar la detección temprana de errores, permitiendo así una respuesta ágil a los cambios en los requisitos del usuario

- Es necesario realizar un seguimiento constante durante el desarrollo de la aplicación web utilizando el enfoque TDD, asegurando que el equipo mantenga un ciclo iterativo y estructurado. Este proceso debe incluir la actualización regular de las pruebas automatizadas a medida que se integran nuevas funcionalidades. Además, la capacitación continua y la documentación actualizada son esenciales para garantizar que el equipo aplique correctamente las mejores prácticas de TDD, minimizando errores y facilitando el mantenimiento a largo plazo. De este modo, la aplicación podrá adaptarse a las necesidades cambiantes de los bancos comunitarios y responder de manera ágil a futuros requerimientos.

- Es importante implementar un proceso de monitoreo continuo y auditorías de seguridad basadas en la norma ISO/IEC 25010 para asegurar que la aplicación mantenga un nivel óptimo de protección en autenticidad, confidencialidad e integridad de los datos. Dado que las amenazas y vulnerabilidades pueden evolucionar con el tiempo, es crucial actualizar regularmente las estrategias de seguridad según las mejores prácticas establecidas por esta normativa, así como capacitar al equipo de desarrollo en los estándares actuales de la ISO para asegurar un entorno de desarrollo seguro y conforme a los lineamientos internacionales.

BIBLIOGRÁFIA

[1] P. S. Alice Johnson, «Security Implications of Test-Driven Development in Banking Software,» *International Journal of Information Security*, vol. 15, pp. 345-358, 2020.

2] S. Brown, «Enhancing Software Security through Test-Driven Development Practices,» *Journal of Cybersecurity and Privacy*, vol. 3, pp. 129-145, 2020.

[3] Open Web Application Security Project, «Secure Software Development Lifecycle,» 2021. [En línea]. Available: https://owasp.org/www-project-secure-software-development-lifecycle/.

[4] R. C, Seacord, Secure Coding in C and C++, Boston: Addison-Wesley, 2021.

[5] D. A.-P. y. M. A. S. Avilés Matute, «Desarrollo de sistema Web basado en los frameworks de Laravel y VueJs, para la gestión por procesos: Un estudio de caso,» *Revista peruana de computación y sistemas,* vol. 3, nº 2, 2020.

[6] G. Castell, «Desarrollo e implementación de una aplicación web progresiva (PWA),» *Revista de Innovación en Tecnología,* vol. 5, nº 2, pp. 123-130, 2020.

[7] M. J. R. M. A. Ramos Martín, Aplicaciones Web.

[8] N. Pizarro, «Diferencias entre una aplicación web y un sitio web,» *Ida BLOG,* 2020.

[9] C. d. l. M. A. e. al, «APLICACIÓN WEB 'CIENTÍFICAS,» *Innovación Educativa en la sociedad digital,* 2022.

[10] Y. V.-V. y. V. T.-C. J. Llamuca-Quinaloa, «Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una aplicación web progresiva,» *TecnoLógicas,* vol. 24, nº 51, 2021.

[11] C. VersionOne, «17th State of Agile Report | Analyst Reports | Digital.ai».

[12] N. P. y. C. Martins, «Test driven development in action: Case study of a cross-platform web application,» *EUROCON 2021 - 19th IEEE International Conference on Smart Technologies, Proceedings,* 2021.

[13] J. Raura, «Impacto de las características personales de los programadores en la efectividad de Test-Driven Development (TDD),» Universidad Nacional de la Plata, 2021.

[14] D. S. e. al., «A Literature Review on the Challenges of Applying Test-Driven Development in Software Engineering,» *Complex Systems Informatics and Modeling Quarterly,* vol. 2022, nº 31, 2022.

[15] A. Araújo, «Test Driven Development: Fortalezas y Debilidades».

[16] S. Q. e. al., «Prácticas orientadas por pruebas para el desarrollo de software, una revisión sistemática,» *Periodicidad: Semestral,* vol. 8, nº 3, 2022.

[17] T. E. J. V. y. B. Marín, «Enseñanza temprana del testing en cursos de programación,» [En línea]. Available: www.qped.eu.

[18] A. R. y. M. Mnich, «Test-driven development with mutation testing – an experimental study,» *Software Quality Journal,* vol. 29, nº 1, pp. 1-38, 2021.

[19] I. F. d. Silva, «Describing the design thinking and extreme programming activities during a technology innovation academic workshop,» *Innovation and Management Review,* vol. 17, nº 3, pp. 267-284, 2020.

[20] «TDD: Desarrolla sin miedo, haciendo las pruebas primero,» 2023. [En línea]. Available: https://www.redsauce.net/blog/es/tdd.

[21] M. Á. V. C. y. D. L. R. A. E. Soraluz Soraluz, «Desarrollo guiado por comportamiento: buenas prácticas para la calidad de software,» *Ingeniería y Desarrollo,* vol. 39, nº 1, p. 190–204, 2023.

[22] T. Chaplin, AngularJS Test-Driven Development: Implement the Best Practices to Improve Your AngularJS Applications Using Test-Driven Development, 2020.

[23] M. P. y. M. P.-W. F. Wąsik, «Comparative analysis of selected tools for test automation of web applications,» *Analiza porównawcza wybranych narzędzi do automatyzacji testów aplikacji webowych,* 2023.

[24] S. Siddiqui, «Learning Test-Driven Development,» [En línea]. Available: https://books.google.es/books?hl=es&lr=&id=M7BHEAAAQBAJ&oi=fnd&pg=PP1&dq=%22test+driven+development%22&ots=95S2izePMX&sig=xwZc78O-xcagM0G41VDM2qk5Il8#v=onepage&q=%22test%20driven%20development%22&f=false..

[25] «Applying Test Driven Development in the Big Data Domain – Lessons From the Literature",» *IEEE Conference Publication.*

[26] M. V. E. L. M. P. M. A. y. K. T. D. Staegemann, «Applying Test Driven Development in the Big Data Domain - Lessons from the Literature,» de *021 International Conference on Information Technology, ICIT 2021 - Proceedings*, 2021.

[27] D. Community, «Tests unitarios en JavaScript con Jasmine,» [En línea]. Available: https://dev.to/juanmirod/tests-unitarios-en-javascript-con-jasmine-118e.

[28] I. 25010, «iso25000.com,» [En línea]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25010?start=5.

[29] Angular, «angular.io,» [En línea]. Available: https://angular.io/.

[30] M. B. y. J. Gomez, El gran libro de Angular, vol. Primera Edición, México D.F: Alfaomega Grupo Editor, 2019.

[31] M. S. H. Ahmed, Angular: Up and Running, Sebastopol, CA, USA: O'Reilly Media, 2020.

[32] Kinsta, «Qué es Node.js y por qué debería usarlo,» p. 3–9, 2023.

[33] J. W. Creswell, Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, Thousand Oaks, CA: SAGE Publications, 2021.

[34] K. D. y. D. H. T. Kovács-Öller, «Editorial: Visual code: From the retina to the brain,» *Frontiers in Cellular Neuroscience*, vol. 16, p. 1018229, 2023.

[35] V. S. I. y. E. d. c. p. d. d. s. y. Teams. [En línea]. Available: https://visualstudio.microsoft.com/es/#vs-section.

[36] H.-J. Schönig, Mastering PostgreSQL 13: Build, administer, and maintain database.

[37] J. A. A. y. M. T. M. B. León Soberón, «Análisis comparativo de sistemas gestores de bases de datos PostgreSQL y MySQL en procesos CRUD,» *Repositorio Institucional - USS*, 2020.

[38] J. D. Chávez, «CLIENTE PSQL DE POSTGRESQL,» 2020. [En línea]. Available: http://creativecommonsvenezuela.org.ve.

[39] N. L. C. P. y. L. J. Cerveleón, «Kanban como herramienta de mejora de procesos productivos,» *Ingeniare*, vol. 17, nº 31, pp. 81-92, 2021.

[40] O. Foundation, «Zed Attack Proxy (ZAP),” OWASP,» 12 10 2023. [En línea]. Available: https://owasp.org/www-project-zap/.

[41] M. Thomas, «A Beginner's Guide to OWASP ZAP,» *Cybersecurity Magazine*, 2022.

[42] Inesem, «Burp Suite vs ZAP: Cuál es mejor para auditar la seguridad web,» *Inesem*, 2022.

[43] J. P. a. M. F. G. González, «Análisis de vulnerabilidades en aplicaciones web utilizando métodos tradicionales," en Proc. of the International Conference on Cybersecurity,» de *Conferencia internacional sobre ciberseguridad.*, 2021.

[44] J. M. a. A. Sánchez, «Implementación de medidas de autenticación en aplicaciones web,» *Revista de Seguridad Informática*, vol. 12, nº 2, pp. 23-30, 2020.

[45] L. L. a. M. Pérez, «Estrategias de seguridad en aplicaciones web: un enfoque de cifrado y control de acceso,» *Revista de Ciberseguridad*, vol. 15, nº 3, pp. 45-52, 2020.

[46] P. M. a. F. G. R. Fernández, «Mejoras en la confidencialidad de aplicaciones web mediante cifrado avanzado,» de *Annual Security Conference*, 2021.

[47] P. M. a. F. G. R. Fernández, «Mejoras en la integridad de datos en aplicaciones web mediante validación y sanitización,» de *Annual Security Conference*, 2020.

[48] J. S. a. A. G. L. López, «Estrategias de protección de datos en aplicaciones web,» *Revista de Tecnologías de Seguridad*, vol. 14, nº 1, pp. 70-78, 2020.

[49] J. Martínez, «Impacto de PHP y MySQL en la seguridad de aplicaciones web,» *Revista de Tecnologías Web*, vol. 18, nº 4, pp. 68-75, 2020.

[50] J. G. a. A. P. P. González, «Vulnerabilidades comunes en aplicaciones PHP y sus implicaciones en la seguridad,» de *International Conference on Web Development*, 2020.

[51] Y. Nabor, Introducción a .net, 2008.

[52] A. Pesántez, «Desarrollo de una plataforma web para la gestión de operaciones en Bancos Comunitarios del cantón Cañar utilizando la metodología Test Driven Development,» Tesis de Maestría, Universidad Politécnica Salesiana, Cuenca, Ecuador., 2021.

[53] C. VersionOne, 2020. [En línea]. Available: https://explore.digital.ai/state-of-agile/14th-annual-state-of-agile-report.

[54] R. Juan, «DEV Community,» 20 Marzo 2020. [En línea]. Available: https://dev.to/juanmirod/tests-unitarios-en-javascript-con-jasmine-118e.

[55] «ISO25000,» 2023. [En línea]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25010?start=5.

[56] A. H. y. A. S. K. Eng, «Identifying Defect-Inducing Changes in Visual Code,» de *Proceedings - 2023 IEEE International Conference on Software Maintenance and Evolution, ICSME 2023*, 2023.

[57] O. Foundation, «Authentication Cheat Sheet,» 2020. [En línea]. Available: https://owasp.org/www-community/authentication_cheat_sheet.

[58] R. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, Wiley, 2020.

[59] L. Johnson, «Encryption Standards for Data Protection,» *Computer Security Review*, 2021.

[60] K. Miller, «Access Control Best Practices,» *Security and Privacy Journal*, 2022.

[61] H. Lee, «Data Integrity in Modern Applications,» *Journal of Information Assurance*, 2023.

[62] E. White, «Standards for Input Validation,» *Web Application Security Journal*, 2022.

[63] J. Doe, «Best Practices in Session Management,» *International Journal of Cyber Security*, 2023.

[64] O. Foundation, «OWASP ZAP API Documentation OWASP,» 2023. [En línea]. Available: https://www.zaproxy.org/docs/api/.

ANEXOS

Anexo 1: Requerimientos

ESPECIFICACIONES DE REQUISITOS DE SOFTWARE

Sistema	Aplicación web para Bancos Comunitarios
Modulo/Funcionalidad	Aplicación web para Bancos Comunitarios
Unidad Requirente	Banco Comunitarios
Titular de la Unidad	Gerente del banco

N°	Nombre del Requerimiento	Descripción del Requerimiento	Para	Prioridad	Criterio de Aceptación
1	Registro de Usuarios	Los usuarios deben poder registrarse en la plataforma.	Permitir a los usuarios acceder a sus cuentas.	Alta	El sistema permite a los usuarios completar el formulario de registro con información válida y se crea una cuenta para ellos correctamente.
2	Apertura de Cuentas	Los usuarios deben poder abrir cuentas bancarias a través de la plataforma.	Permitir a los usuarios realizar transacciones bancarias.	Alta	Los usuarios pueden completar el proceso de apertura de cuentas proporcionando la información requerida, y se les asigna un número de cuenta único.
3	Operaciones Bancarias	Los usuarios deben poder realizar operaciones bancarias como depósitos, retiros	Facilitar la gestión financiera de los usuarios.	Alta	Los usuarios pueden realizar depósitos, retiros de manera segura y eficiente, y el sistema actualiza los saldos correctamente.
4	Seguridad de la Información	La plataforma debe garantizar la seguridad de la información del usuario.	Proteger los datos sensibles de los usuarios.	Alta	La plataforma utiliza métodos de cifrados sólidos para proteger la información del usuario, y se implementan medidas de seguridad adicionales para prevenir accesos no autorizados.
5	Generación de Informes	Los usuarios deben poder generar informes detallados	Permitir a los usuarios realizar un seguimiento	Media	Los usuarios pueden generar informes personalizados que incluyen detalles precisos

		sobre sus transacciones y estados de cuenta.	de sus transacciones.		sobre sus transacciones y estados de cuenta, y los informes se generan de manera rápida y precisa.
6	Accesibilidad de Informes	Los informes deben ser fácilmente accesibles y descargables desde la plataforma.	Facilitar el acceso a la información financiera.	Media	Los usuarios pueden acceder a los informes fácilmente desde sus cuentas y descargarlos en formatos compatibles con diferentes dispositivos y aplicaciones.

Anexo 2: Diccionario de Datos

N°	Nombre	Usuario	Prioridad	Puntos Estimados	Descripción	Pruebas de aceptación	Estado Actual
1	Registro de Usuario	Usuario (nuevo o registrado)	Alta	30 PE	Como usuario, quiero registrarme en la plataforma para acceder a los servicios del Banco Comunitario	El sistema debe permitir el registro con validación de los datos.	Por Hacer
2	Apertura de Cuentas	Usuario registrado	Alta	40 PE	Como usuario, quiero abrir una cuenta bancaria a través de la plataforma para comenzar a utilizar los servicios bancarios.	El sistema debe permitir abrir cuentas bancarias con verificación	En Progreso
3	Transferencias Bancarias	Usuario registrado	Alta	50 PE	Como **usuario**, quiero realizar transferencias bancarias entre cuentas para gestionar mis finanzas de manera eficiente.	El sistema debe permitir realizar transferencias seguras	En Revisión
4	Consulta de Saldo	Usuario registrado	Alta	20 PE	Como **usuario**, quiero consultar el saldo de mi cuenta para saber cuánto dinero tengo disponible.	El sistema debe mostrar el saldo actual de la cuenta en todo momento.	Completado
5	Seguridad en Transacciones	Usuario registrado	Alta	40 PE	Como **usuario**, quiero que mis transacciones estén protegidas mediante autenticación de dos factores para evitar accesos no autorizados	El sistema debe implementar autenticación de dos factores para todas las transacciones	En Progreso

Anexo 3: Diccionario de Datos

Tabla Usuario

Tabla 14: Campos de Usuario

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_usuario	"integer"	si	no	Identificador único para cada usuario. Primary key usuarios
tb_usu_nombre	"character varying (55)"	no	no	Nombre del usuario
tb_usu_apellido	"character varying (55)"	no	no	Apellido del usuario
tb_usu_cedula	"character varying (11)"	no	no	Número de cédula del usuario
tb_usu_fecha_naciemiento	"date"	no	no	Fecha de nacimiento del usuario
tb_usu_telefono	"character varying (11)"	no	no	Número de teléfono principal del usuario
tb_usu_direccion	"character varying (256)"	no	no	Dirección del usuario.
tb_usu_correo	"character varying (100)"	no	no	Correo electrónico del usuario.
tb_usu_nombre_usuario	"character varying (55)"	no	no	Nombre de usuario utilizado para iniciar sesión.
tb_usu_contrasenia	"character varying (256)"	no	no	Contraseña del usuario
tb_usu_telefono_dos	"character varying (11)"	no	no	Segundo número de teléfono del usuario (opcional).
tb_usu_salto	"character varying (32)"	no	no	Este salto se trabajará junto con la contraseña

Tabla Cuentas

Tabla 15: Campos Cuentas

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_cuenta	integer	Sí	No	Identificador único para cada cuenta. Clave primaria de la tabla.
id_usuario	integer	No	Sí	Identificador único de cada usuario asociado a una cuenta. Clave foránea.
tb_cuent_numero_cuenta	character varying (20)	No	No	Número único asignado a cada cuenta bancaria.
tb_cuent_tipo_cuenta	character varying (20)	No	No	Indica el tipo de cuenta bancaria (e.g., ahorro, corriente).
tb_cuent_monto	numeric	No	No	Monto inicial depositado en la cuenta bancaria.
tb_cuent_total	numeric	No	No	Monto total actualmente disponible en la cuenta bancaria.
id_tipo_cuenta	integer	No	Sí	Identificador del tipo de cuenta. Clave foránea referenciada en otra tabla.

Tabla Historia de pagos

Tabla 16: Campos Historial de pagos

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_historia_pago	integer	Sí	No	Identificador único para cada registro de historial de pago. Clave primaria de la tabla.
id_prestamo	integer	No	Sí	Identificador único del préstamo asociado a un pago. Clave foránea.
tb_hist_pag_fecha_pago	date	No	No	Fecha en la que se realizó el pago del préstamo.
tb_hist_pag_monto_pago	numeric	No	No	Monto pagado en una determinada fecha de pago.
tb_hist_pag_estado	character varying (20)	No	No	Estado del pago (completado, pendiente, atrasado, etc.).

Tabla Interés

Tabla 17: Campos Interés

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_interes	integer	Sí	No	Identificador único para cada registro de interés. Clave primaria de la tabla.
tb_intrs	numeric(5, 2)	No	No	Valor del interés expresado en porcentaje.

Tabla Plazo Fijo

Tabla 18: Campos Plazo Fijo

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_plazo_fijo	integer	Sí	No	Identificador único para cada registro de plazo fijo. Clave primaria de la tabla.
id_interes	integer	No	Sí	Identificador único del tipo de interés asociado al plazo fijo. Clave foránea.
id_usuario	integer	No	Sí	Identificador único del usuario asociado al plazo fijo. Clave foránea.
tb_plazfij_montoincial	numeric(20, 2)	No	No	Monto inicial del plazo fijo.
tb_plazfij_fecha_inicio	date	No	No	Fecha de inicio del plazo fijo.
tb_plazfij_fecha_fin	date	No	No	Fecha de finalización del plazo fijo.
tb_plazfij_total	numeric(20, 2)	No	No	Total, del plazo fijo, incluyendo intereses acumulados u otros aspectos financieros.

Tabla Prestamos

Tabla 19: Campos Prestamos

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_prestamo	integer	Sí	No	Identificador único para cada préstamo. Clave primaria de la tabla.
id_tipo_prestamo	integer	No	Sí	Identificador único del tipo de préstamo asociado. Clave foránea.
id_cuenta	integer	No	Sí	Identificador único de la cuenta asociada al préstamo. Clave foránea.
tb_prest_monto	numeric(20, 2)	No	No	Monto del préstamo.
tb_prest_interes	numeric(5, 2)	No	No	Interés aplicado al préstamo.
tb_prest_fecha_inicio	date	No	No	Fecha de inicio del préstamo.
tb_prest_fecha_fin	date	No	No	Fecha de finalización del préstamo.
tb_prest_capital	numeric(20, 2)	No	No	Capital del préstamo.
tb_prest_saldo	numeric(20, 2)	No	No	Saldo del préstamo.
tb_prest_estado	character varying(55)	No	No	Estado del préstamo.
id_interes	integer	No	Sí	Identificador único del tipo de interés asociado. Clave foránea.

Tabla Tipo de Cuentas

Tabla 20: Campos Tipo de Cuentas

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_tipo_cuenta	integer	Sí	No	Identificador único para cada tipo de cuenta. Clave primaria de la tabla.
tb_tipcuen_tipo	character varying(55)	No	No	Tipo de cuenta.
tb_tipcuen_descripcion	character varying(255)	No	No	Descripción del tipo de cuenta.

Tabla Tipo de Prestamos

Tabla 21: Campos Tipo de Prestamos

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_tipo_prestamo	integer	Sí	No	Identificador único para cada tipo de préstamo. Clave primaria de la tabla.
tb_tiprest_cuota_fija	character varying(20)	No	No	Indica si el préstamo tiene una cuota fija.
tb_tiprest_cuota_no_fija	character varying(20)	No	No	Indica si el préstamo tiene una cuota no fija.
tb_tiprest_cuota_trimestral	character varying(20)	No	No	Indica si el préstamo tiene una cuota trimestral.
tb_tiprest_cuota_semestral	character varying(20)	No	No	Indica si el préstamo tiene una cuota semestral.

Tabla Tipo de Transacciones

Tabla 22: Campos Tipo de Transacciones

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_tipo_transaccion	integer	Sí	No	Identificador único para cada tipo de transacción. Clave primaria de la tabla.
tb_tiptrans_tipo	character varying(22)	No	No	Describe el tipo de transacción realizado.

Tabla Transacciones

Tabla 23: Campos Transacciones

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_transaccion	integer	Sí	No	Identificador único para cada transacción. Clave primaria de la tabla.
id_usuario	integer	No	No	Identificador del usuario que realizó la transacción. Clave externa relacionada con la tabla de usuarios.
id_tipo_transaccion	integer	No	No	Identificador del tipo de transacción realizada. Clave externa relacionada con la tabla de tipos de transacciones.
id_cuenta	integer	No	No	Identificador de la cuenta asociada a la transacción. Clave externa relacionada con la tabla de cuentas.
tb_trans_monto	numeric(20, 2)	No	No	Monto de la transacción.
tb_trans_fecha	date	No	No	Fecha en la que se realizó la transacción.
tb_trans_descripcion	character varying(55)	No	No	Descripción de la transacción.

Tabla Login

Tabla 24: Campos Login

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_login	integer	Sí	No	Identificador único para cada registro de login. Clave primaria de la tabla.
id_perfil	integer	No	No	Identificador del perfil asociado al usuario. Clave externa relacionada con la tabla de perfiles.
tb_log_nombre_usuario	character varying(50)	No	No	Nombre de usuario utilizado para iniciar sesión.
tb_log_contrasenia	character varying(55)	No	No	Contraseña del usuario para iniciar sesión.

Tabla Perfiles

Tabla 25: Campos Perfiles

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_perfil	integer	Sí	No	Identificador único para cada perfil. Clave primaria de la tabla.
tb_perf_nombre	character varying(50)	No	No	Nombre del perfil.
tb_perf_descripcion	character varying(256)	No	No	Descripción del perfil.
tb_perf_estado	integer	No	No	Estado del perfil, "1" para activo, "0" para inactivo.
id_aplicacion	integer	No	No	Identificador de la aplicación asociada al perfil. Clave externa relacionada con la tabla de aplicaciones registradas.
tb_perf_codificacion	character(32)	No	No	Código de codificación asociado al perfil. Se guarda el código del perfil.

Tabla Usuarios Perfiles

Tabla 26: Campos Usuarios Perfiles

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
identificador	character varying(13)	Si	Si	Identificador único que puede ser el nombre o número para cada registro en la tabla.
id_perfil	integer	Si	Si	Identificador del perfil asociado al usuario. Es una clave externa que se relaciona con la tabla de perfiles.

Tabla Acciones

Tabla 27: Campos Acciones

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_accion	integer	Si	No	Identificador único para cada acción.
pagina	character varying	No	No	La página o URL asociada a la acción.
estilo	character varying	No	No	El estilo o clase CSS asociado a la acción.
id_opcion	integer	No	Si	Identificador de la opción asociada a la acción.
descripcion	character varying	No	No	Descripción de la acción.
orden	integer	No	No	Orden de la acción en la interfaz.
id_aplicacion	integer	No	Si	Identificador de la aplicación asociada a la acción.

Tabla Acciones Perfiles

Tabla 28: Campos Acciones Perfiles

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_perfil	integer	Si	Si	Identificador del perfil al que se asigna una acción.
id_accion	integer	Si	Si	Identificador de la acción que se asigna al perfil.

Tabla Aplicaciones Registradas

Tabla 29: Campos Aplicaciones Registradas

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_aplicacion	integer	Si	Si	Identificador único para cada aplicación registrada.
identificador	character varying	Si	Si	Identificador único o nombre de la aplicación.

Tabla Opciones

Tabla 30: Campos Opciones

Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_opcion	integer	Si	No	Identificador único para cada opción.
id_aplicacion	integer	Si	Si	Identificador de la aplicación asociada a la opción.
nombre_opcion	character varying	No	No	Nombre o título de la opción.
estilo	character varying	No	No	Estilo o clase CSS asociada a la opción.
pagina	character varying	No	No	Página o URL asociada a la opción.
orden	integer	No	No	Orden de la opción en la interfaz.
id_flujo	integer	No	No	Identificador del flujo asociado a la opción.
estado_opcion	integer	No	No	Estado de la opción, puede ser "1" activo o "0" inactivo.

Tabla Programas

Tabla 31: Campos Programas

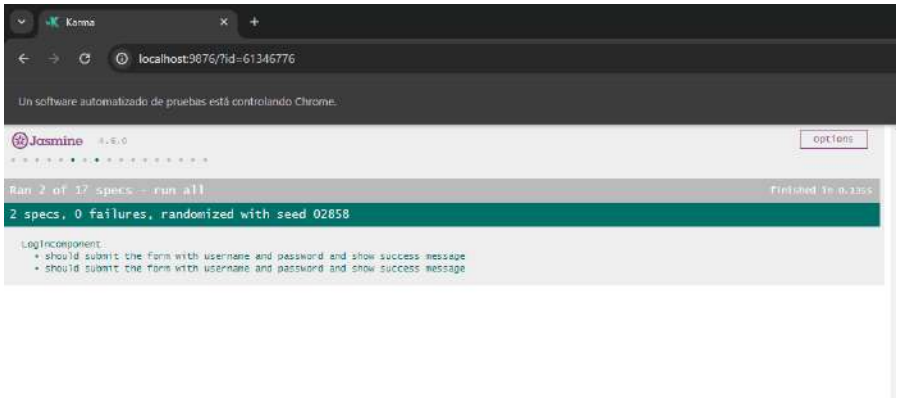
Columna	Tipo De Dato	Primary Key	Foreign Key	Descripción
id_aplicacion	integer	Si	No	Identificador único para cada aplicación.
nombre	character varying	No	No	Nombre de la aplicación.
version	character varying	No	No	Versión de la aplicación.
ruta	character varying	No	No	Ruta o ubicación del programa.
descripcion	character varying	No	No	Descripción de la aplicación.
color	character varying	No	No	Color asociado a la aplicación.
codificacion_aplicacion	character varying	No	No	Codificación asociada a la aplicación.
estado_aplicacion	integer	No	No	Estado de la aplicación, puede ser "1" activo o "0" inactivo.
fecha_carga_aplicacion	timestamp without time zone	No	No	Fecha de carga o instalación del programa.
id_departamento	character varying	No	No	Identificador del departamento asociado a la aplicación.
tipo_usuario	character varying			Tipo de usuario asociado a la aplicación.

Anexo 4: Pruebas Unitarias TDD
Pruebas al módulo login

```
import { TestBed, ComponentFixture, fakeAsync, tick } from
'@angular/core/testing';
import { FormsModule } from '@angular/forms';
import { HttpClientTestingModule, HttpTestingController } from
'@angular/common/http/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { LoginComponent } from './login.component';
import { ServicesComponent } from 'src/app/services/ServicesComponent';
import { of } from 'rxjs';
import { Router } from '@angular/router';

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;
  let services: ServicesComponent;
  let httpMock: HttpTestingController;
  let router: Router;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [LoginComponent],
      imports: [FormsModule, HttpClientTestingModule, RouterTestingModule],
      providers: [ServicesComponent]
    }).compileComponents();

    fixture = TestBed.createComponent(LoginComponent);
    component = fixture.componentInstance;
    services = TestBed.inject(ServicesComponent);
    httpMock = TestBed.inject(HttpTestingController);
    router = TestBed.inject(Router);
    fixture.detectChanges();
  });

  afterEach(() => {
    httpMock.verify();
  });

  it('should display successful login message when submitting valid
credentials', fakeAsync(() => {
    spyOn(services, 'login').and.returnValue(of({ /* simulated successful
login response */ }));
    const navigateSpy = spyOn(router, 'navigateByUrl');

    component.nombreUsuario = 'testuser';
    component.contrasenia = 'testpassword';

    component.onSubmit();
```

```
    tick();

    expect(component.successMessage).toEqual('Inicio de sesión exitoso');
    expect(component.errorMessage).toBeUndefined();
    expect(navigateSpy).toHaveBeenCalledWith(jasmine.stringMatching('/admini
strador'), jasmine.objectContaining({ skipLocationChange: false }));
  }));
});
```

Resultados de la ejecución de las pruebas



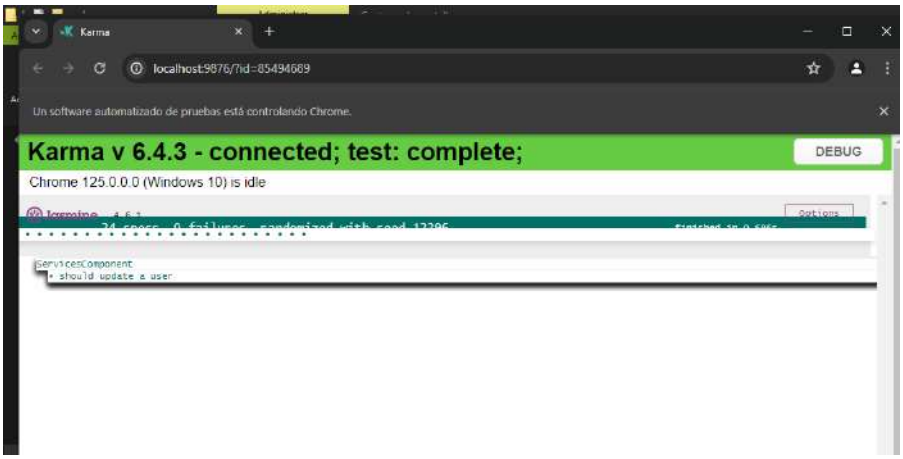
Pruebas crear Usuario

```
import { TestBed, ComponentFixture, fakeAsync, tick, flush } from
'@angular/core/testing';
import { ReactiveFormsModule, FormBuilder } from '@angular/forms';
import { HttpClientTestingModule } from '@angular/common/http/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { RegisterComponent } from './register.component';
import { ServicesComponent } from 'src/app/services/ServicesComponent';
import { of, throwError } from 'rxjs';
import { Router } from '@angular/router';

describe('RegisterComponent', () => {
  let component: RegisterComponent;
  let fixture: ComponentFixture<RegisterComponent>;
  let servicesComponent: ServicesComponent;
  let router: Router;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [RegisterComponent],
      imports: [ReactiveFormsModule, HttpClientTestingModule,
RouterTestingModule],
      providers: [FormBuilder, ServicesComponent]
    }).compileComponents();

    fixture = TestBed.createComponent(RegisterComponent);
```

```
    component = fixture.componentInstance;
    servicesComponent = TestBed.inject(ServicesComponent);
    router = TestBed.inject(Router);
    fixture.detectChanges();
  });

  it('should navigate to the next section', () => {
    component.currentSection = 1;
    component.goToNextSection();
    expect(component.currentSection).toBe(2);
  });

  it('should show the register button on the last section', () => {
    component.currentSection = 3;
    component.goToNextSection();
    expect(component.showRegisterButton).toBeTrue();
  });

  it('should navigate to the previous section', () => {
    component.currentSection = 2;
    component.goToPreviousSection();
    expect(component.currentSection).toBe(1);
    expect(component.showRegisterButton).toBeFalse();
  });

  it('should display error when form is invalid', () => {
    spyOn(console, 'error');

    // Define el estado del formulario como inválido
    component.formulario.setValue({
      tb_usu_nombre: '',  // Esto debería forzar la invalidación del
formulario
      tb_usu_apellido: 'Doe',
      tb_usu_cedula: '1234567890',
      tb_usu_fecha_nacimiento: '01/01/2000',
      tb_usu_telefono: '555-5555',
      tb_usu_direccion: '123 Main St',
      tb_usu_correo: 'john.doe@example.com',
      tb_usu_nombre_usuario: 'johndoe',
      tb_usu_contrasenia: 'password',
      tb_usu_telefono_dos: '',
      tb_usu_salto: ''
    });

    // Llama al método register
    component.register();

    // Verifica que console.error fue llamado con el mensaje esperado
```

```
    expect(console.error).toHaveBeenCalledWith('Por favor, completa todos
los campos obligatorios.');
  });



// En tu prueba
it('should navigate to login on registration success', () => {
    const router = TestBed.inject(Router);
    spyOn(router, 'navigate');
    // Simular éxito en el registro
    component.register();
    expect(router.navigate).toHaveBeenCalledWith(['/login']);
  });


  it('should display error on registration failure', fakeAsync(() => {
    spyOn(servicesComponent,
'registrarUsuario').and.returnValue(throwError('error'));
    spyOn(console, 'error');

    component.formulario.setValue({
      tb_usu_nombre: 'John',
      tb_usu_apellido: 'Doe',
      tb_usu_cedula: '1234567890',
      tb_usu_fecha_nacimiento: '01/01/2000',
      tb_usu_telefono: '555-5555',
      tb_usu_direccion: '123 Main St',
      tb_usu_correo: 'john.doe@example.com',
      tb_usu_nombre_usuario: 'johndoe',
      tb_usu_contrasenia: 'password',
      tb_usu_telefono_dos: '',
      tb_usu_salto: ''
    });

    component.register();
    tick();
    flush();  // Asegúrate de que todos los temporizadores se completen

    expect(console.error).toHaveBeenCalledWith('Error al registrar
usuario:', 'error');
  }));
});
```

Resultados de la ejecución de las pruebas

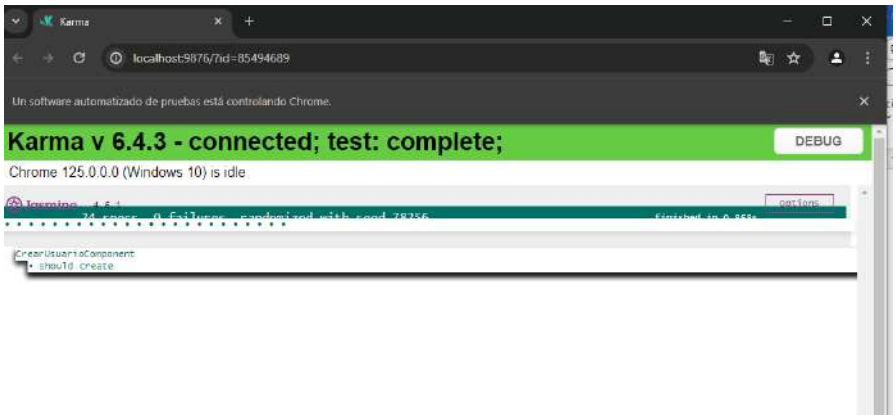


Pruebas al Modulo Transacciones

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { PrestamosComponent } from './prestamos.component';
import { FormBuilder, ReactiveFormsModule } from '@angular/forms';
import { ServicesComponent } from 'src/app/services/ServicesComponent';
import { of } from 'rxjs';

describe('PrestamosComponent', () => {
  let component: PrestamosComponent;
  let fixture: ComponentFixture<PrestamosComponent>;
  let servicesComponentSpy: jasmine.SpyObj<ServicesComponent>;

  beforeEach(async () => {
    const servicesComponentMock = jasmine.createSpyObj('ServicesComponent',
[
      'getUsuarioPorCedula',
      'ingresartodo',
      'traernombrecuentamonto',
      'getsolicitud',
      'addinforme',
      'addresolucion'
    ]);

    await TestBed.configureTestingModule({
      declarations: [PrestamosComponent],
      imports: [ReactiveFormsModule],
      providers: [
        FormBuilder,
        { provide: ServicesComponent, useValue: servicesComponentMock }
      ]
    })
    .compileComponents();
```

```
    servicesComponentSpy = TestBed.inject(ServicesComponent) as
jasmine.SpyObj<ServicesComponent>;
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(PrestamosComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  it('should show register form when calling showRegisterForm()', () => {
    component.showRegisterForm();
    expect(component.showRegister).toBeTrue();
    expect(component.showRequest).toBeFalse();
    expect(component.showNew).toBeFalse();
  });

  // More test cases can be added here...

  it('should load existing requests on component initialization', () => {
    const mockSolicitudes = [{ id: 1, nombre: 'Solicitud 1' }, { id: 2,
nombre: 'Solicitud 2' }];
    servicesComponentSpy.traernombrecuentamonto.and.returnValue(of(mockSolic
itudes));

    component.ngOnInit();

    expect(component.solicitudes).toEqual(mockSolicitudes);
  });

});
```

Resultados de la ejecución de las pruebas

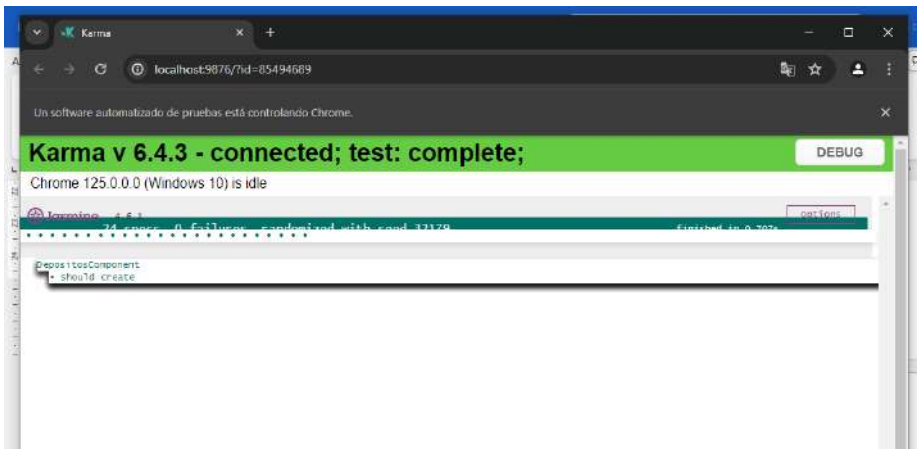


Prueba Depositos

```
import { TestBed, ComponentFixture } from '@angular/core/testing';
import { FormsModule } from '@angular/forms';
import { DepositosComponent } from './depositos.component';
import { ServicesComponent } from 'src/app/services/ServicesComponent';
import Swal from 'sweetalert2';

describe('DepositosComponent', () => {
  let component: DepositosComponent;
  let fixture: ComponentFixture<DepositosComponent>;
  let servicesComponentSpy: jasmine.SpyObj<ServicesComponent>;

  beforeEach(async () => {
    const servicesSpy = jasmine.createSpyObj('ServicesComponent',
['postTransaccion']);

    await TestBed.configureTestingModule({
      declarations: [DepositosComponent],
      imports: [FormsModule],
      providers: [{ provide: ServicesComponent, useValue: servicesSpy }]
    }).compileComponents();

    fixture = TestBed.createComponent(DepositosComponent);
    component = fixture.componentInstance;
    servicesComponentSpy = TestBed.inject(ServicesComponent) as
jasmine.SpyObj<ServicesComponent>;
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  describe('realizarTransaccion', () => {
    it('should not perform transaction if form is invalid', () => {
      const form = { value: {} };
      component.realizarTransaccion(form as any);
      expect(servicesComponentSpy.postTransaccion).not.toHaveBeenCalled();
```

```
    });

    it('should perform transaction if form is valid', () => {
      const form = { value: {
        tipo_transaccion: '2',
        tb_trans_monto: '100',
        tb_trans_fecha: '2024-06-11',
        tb_trans_descripcion: 'Test transaction'
      } };
      servicesComponentSpy.postTransaccion.and.returnValue({ subscribe: ()
=> {} });
      component.realizarTransaccion(form as any);
      expect(servicesComponentSpy.postTransaccion).toHaveBeenCalled();
    });
  });

  describe('obtenerTipoTransaccion', () => {
    it('should return correct type for known ID', () => {
      expect(component.obtenerTipoTransaccion(2)).toBe('Aporte');
      expect(component.obtenerTipoTransaccion(8)).toBe('Patrimonio');
    });

    it('should return "Tipo Desconocido" for unknown ID', () => {
      expect(component.obtenerTipoTransaccion(999)).toBe('Tipo
Desconocido');
    });
  });
});
```

Resultados de la ejecución de las pruebas

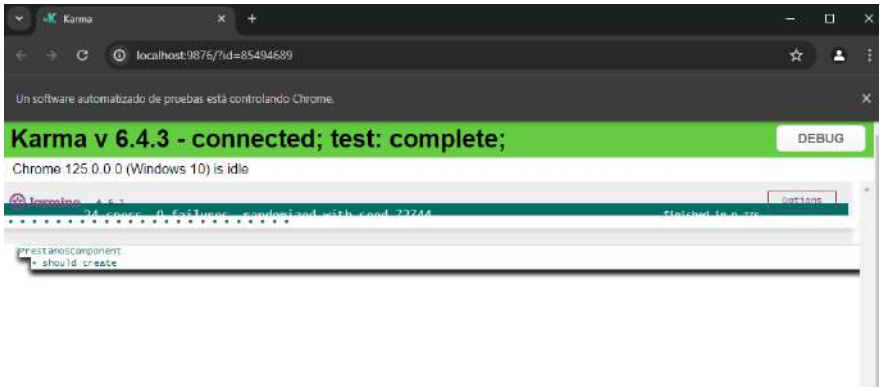


Prueba Cuentas

```
import { ComponentFixture, TestBed, tick, fakeAsync } from
'@angular/core/testing';
import { FormsModule } from '@angular/forms';
import { CuentasComponent } from './cuentas.component';
import { ServicesComponent } from 'src/app/services/ServicesComponent';
import { of } from 'rxjs';
import Swal from 'sweetalert2';

describe('CuentasComponent', () => {
  let component: CuentasComponent;
  let fixture: ComponentFixture<CuentasComponent>;
  let servicesComponent: jasmine.SpyObj<ServicesComponent>;

  beforeEach(async () => {
    const servicesSpy = jasmine.createSpyObj('ServicesComponent', [
      'MostrarUsuario',
      'MostrarCuenta',
      'getUsuarioPorCedula',
      'CrearCuenta',
      'eliminarCuenta'
    ]);

    await TestBed.configureTestingModule({
      declarations: [CuentasComponent],
      imports: [FormsModule],
      providers: [{ provide: ServicesComponent, useValue: servicesSpy }]
    }).compileComponents();

    fixture = TestBed.createComponent(CuentasComponent);
    component = fixture.componentInstance;
    servicesComponent = TestBed.inject(ServicesComponent) as
jasmine.SpyObj<ServicesComponent>;
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });

  describe('ngOnInit', () => {
    it('should call combinarDatos method', () => {
      spyOn(component, 'combinarDatos');
      component.ngOnInit();
      expect(component.combinarDatos).toHaveBeenCalled();
    });
  });

  describe('ngAfterViewInit', () => {
    it('should call combinarDatos and abrirModal methods', fakeAsync(() => {
```

```
      spyOn(component, 'combinarDatos');
      spyOn(component, 'abrirModal');
      component.ngAfterViewInit();
      tick(); // Simulate the passage of time until all pending asynchronous
activities finish
      expect(component.combinarDatos).toHaveBeenCalled();
      expect(component.abrirModal).toHaveBeenCalled();
    }));
  });

  describe('combinarDatos', () => {
    it('should fetch users and accounts data and set totalPages', () => {
      const usuarios = [{ id_usuario: 1, tb_usu_nombre: 'User1' }];
      const cuentas = [{ id_cuenta: 1, tb_cuent_numero_cuenta: '123456' }];
      servicesComponent.MostrarUsuario.and.returnValue(of(usuarios));
      servicesComponent.MostrarCuenta.and.returnValue(of(cuentas));
      component.combinarDatos();
      expect(component.usuariosCuentas.length).toBe(1);
      expect(component.totalPages).toBe(1);
    });

    it('should handle error', () => {
      servicesComponent.MostrarUsuario.and.returnValue(of([]));
      servicesComponent.MostrarCuenta.and.throwError('Error');
      spyOn(console, 'error');
      component.combinarDatos();
      expect(console.error).toHaveBeenCalledWith('Error al combinar datos:',
jasmine.any(Error));
    });
  });

  });
```

Resultados de la ejecución de las pruebas



Anexo 5: Aplicación Web para Bancos Comunitarios

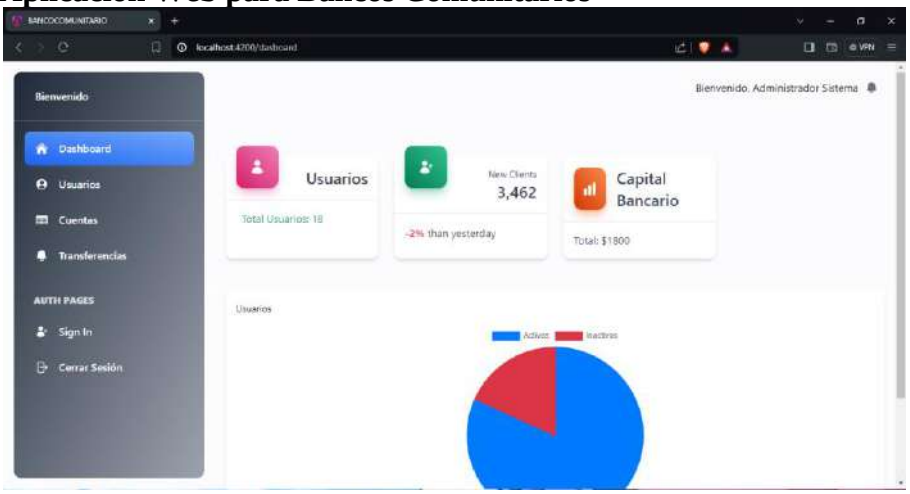


Figura 37: Página principal del administrador

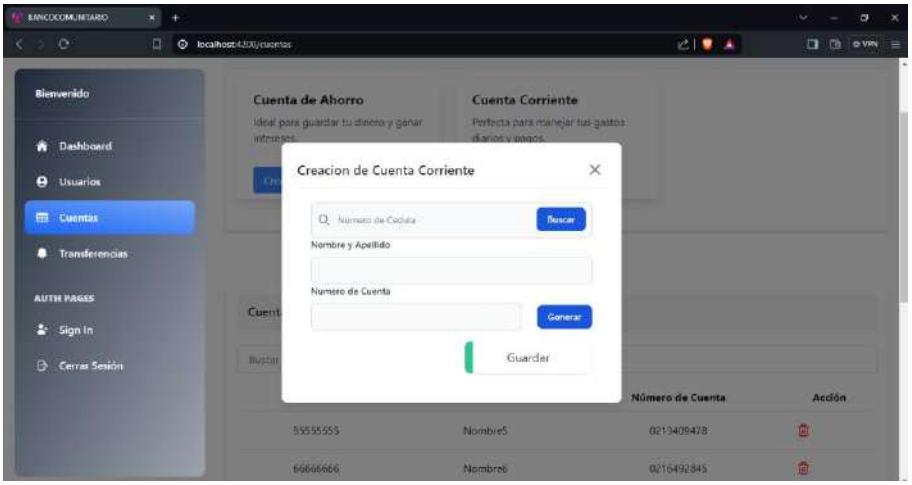


Figura 38: Creación de Cuentas

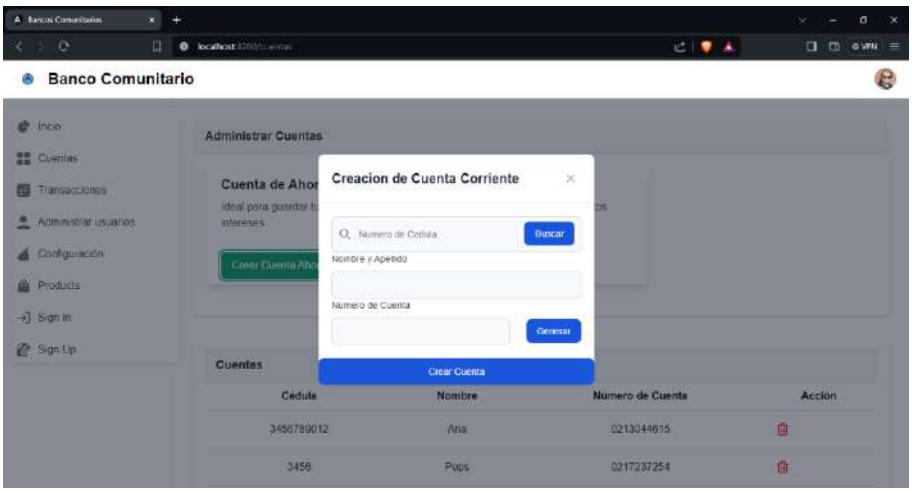


Figura 39: Formulario para la creación de cuentas

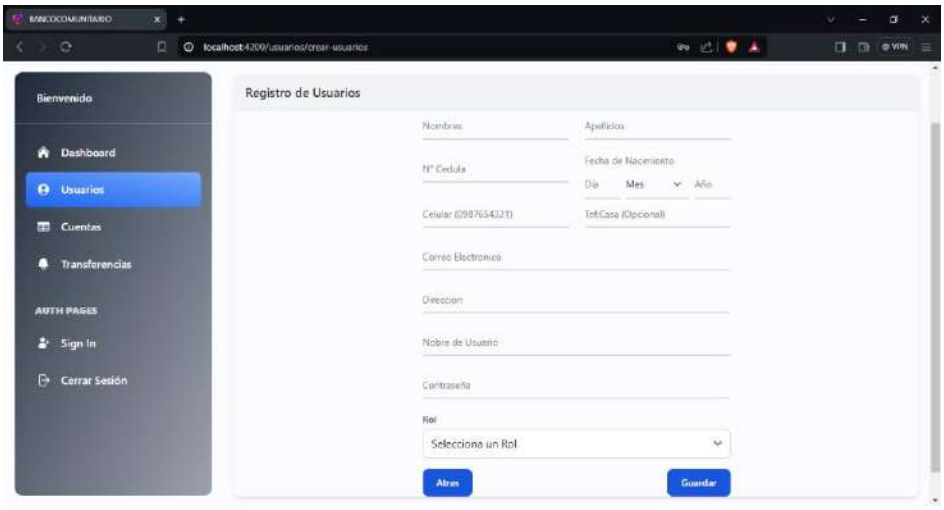


Figura 40: Formulario para ingreso de usuarios

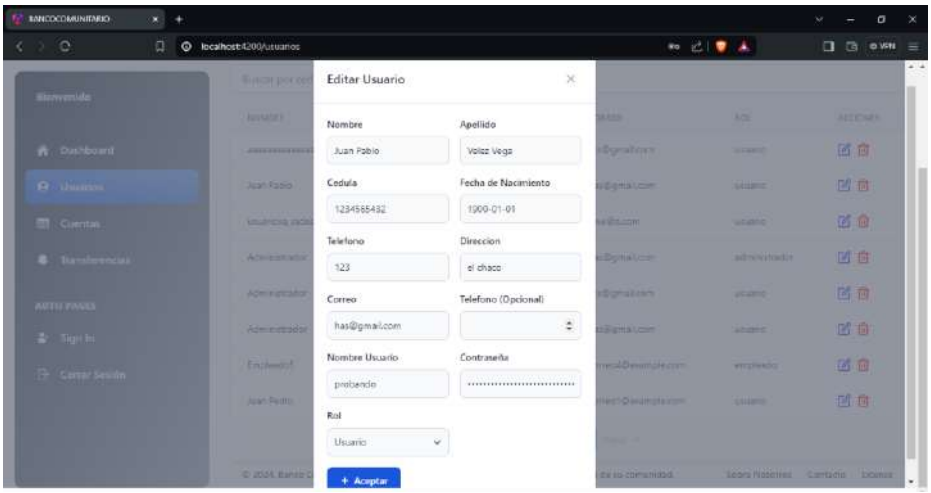


Figura 41: Editar Usuario

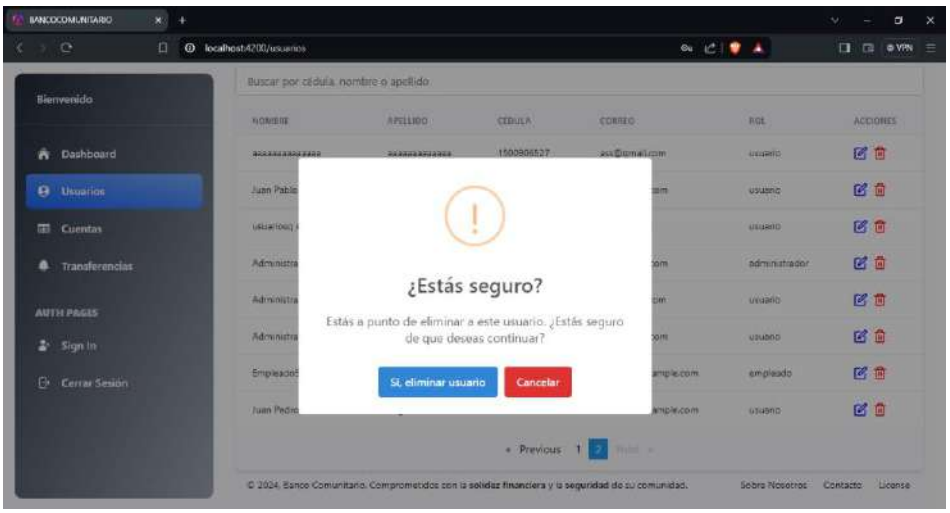


Figura 42: Eliminación de usuario

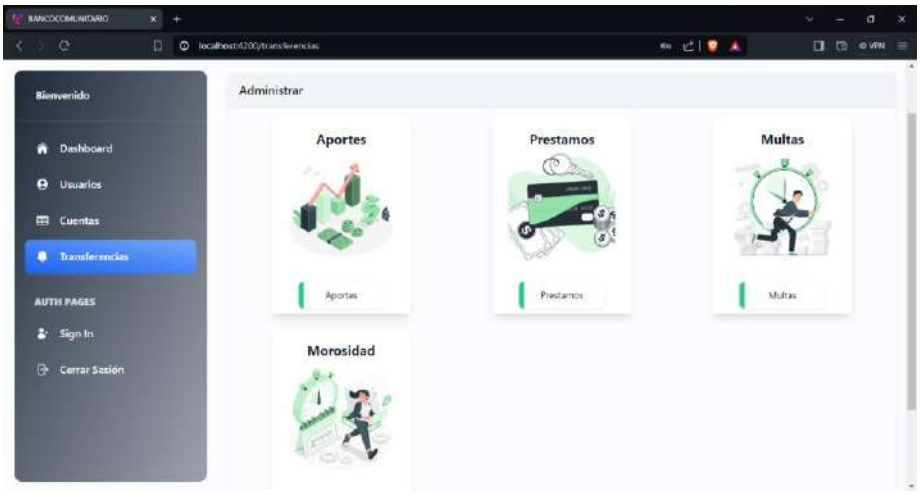


Figura 43: Administrar Transacciones

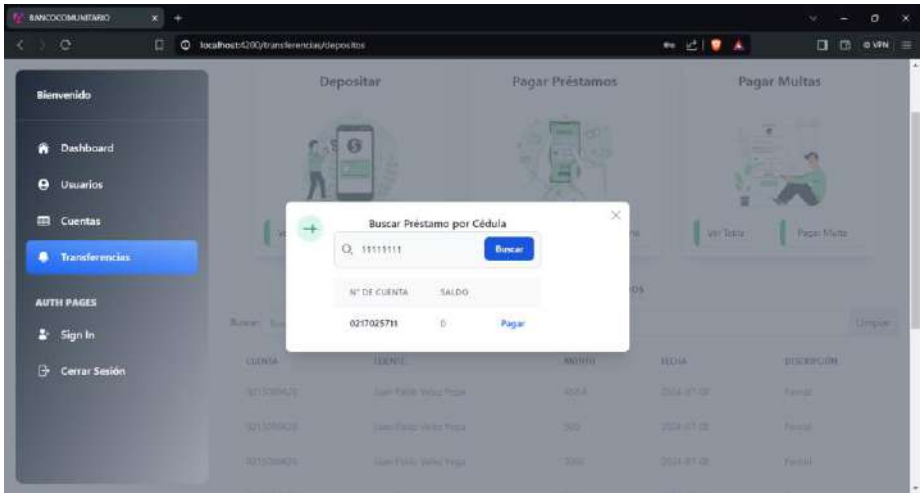


Figura 44: Formulario para pagar prestamos

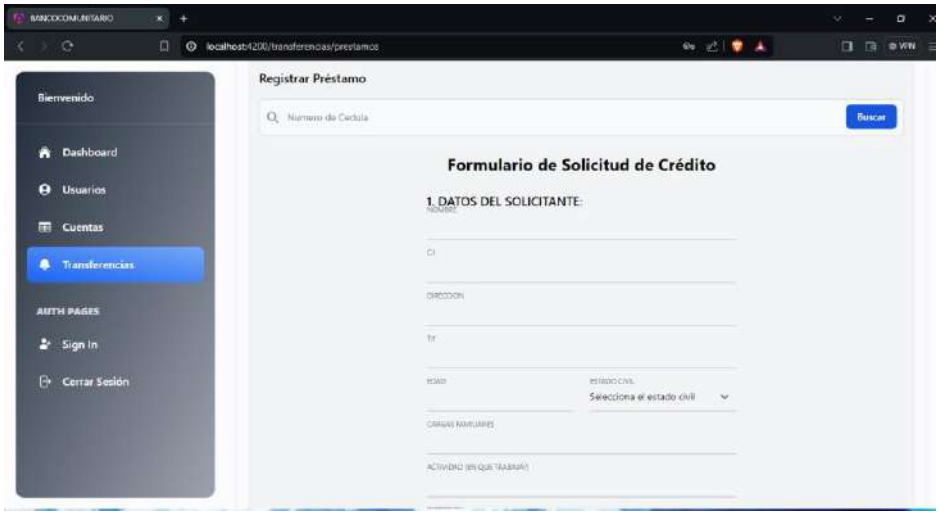


Figura 45: Formulario solicitud de crédito

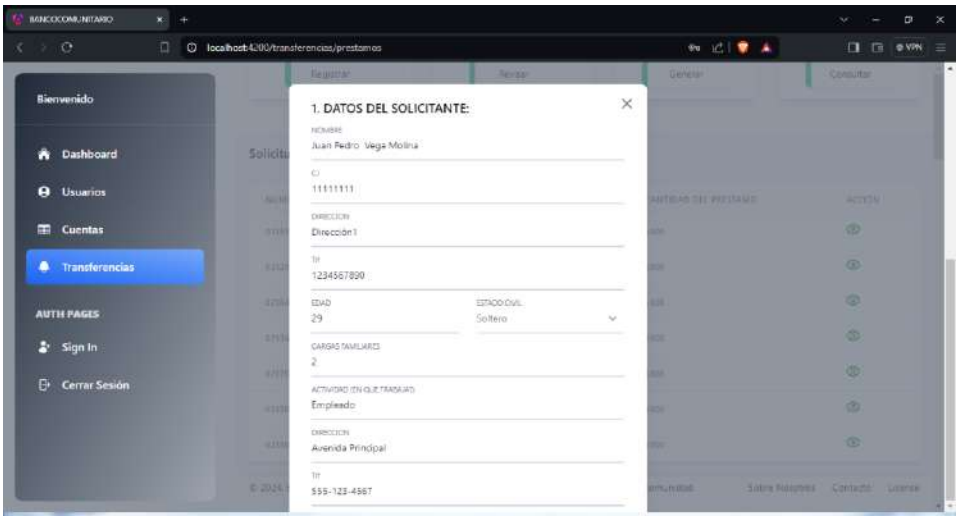


Figura 46: Aprobación de crédito

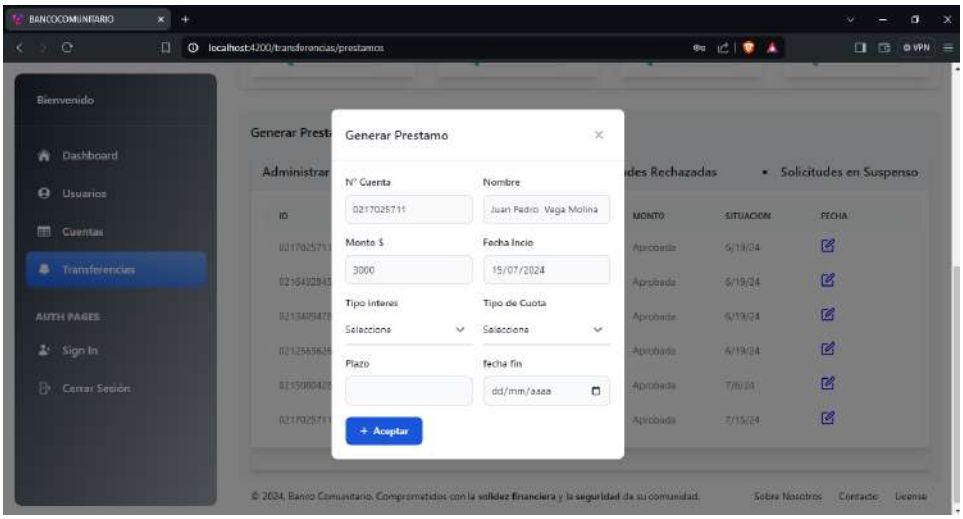


Figura 47: Formulario para generar prestamos

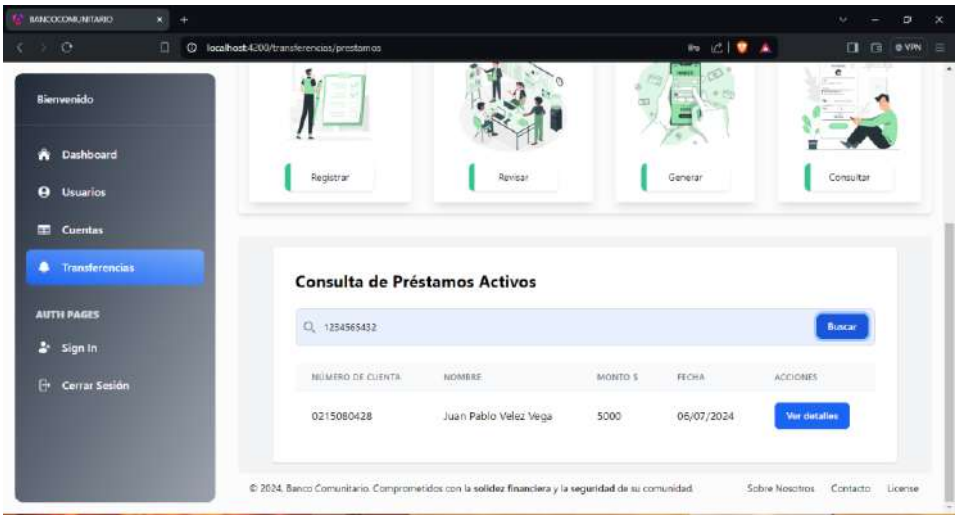


Figura 48: Consulta de prestamos

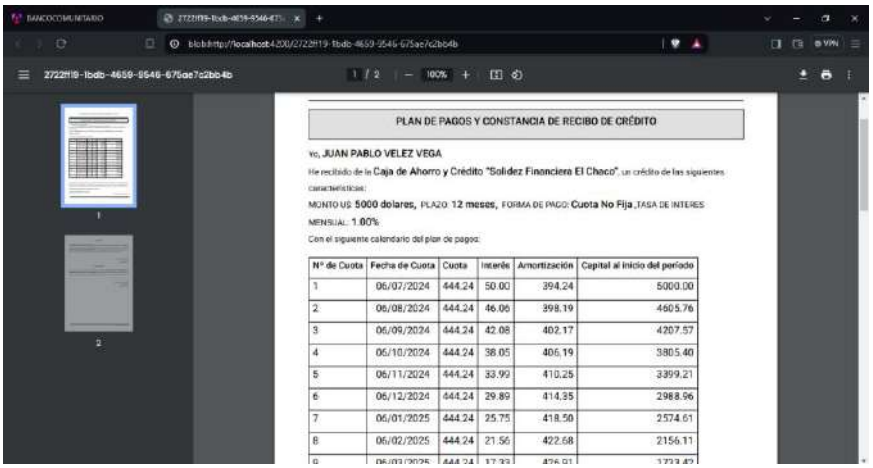


Figura 49: Registro de pagos

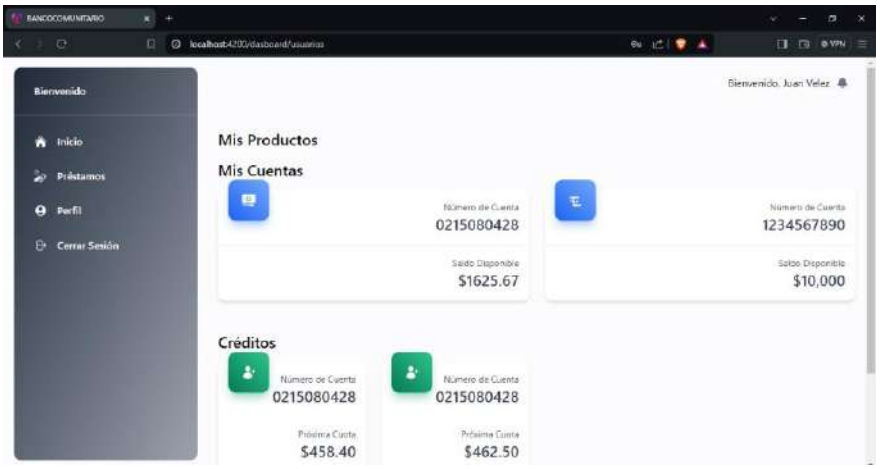


Figura 50: Página Principal del Usuario

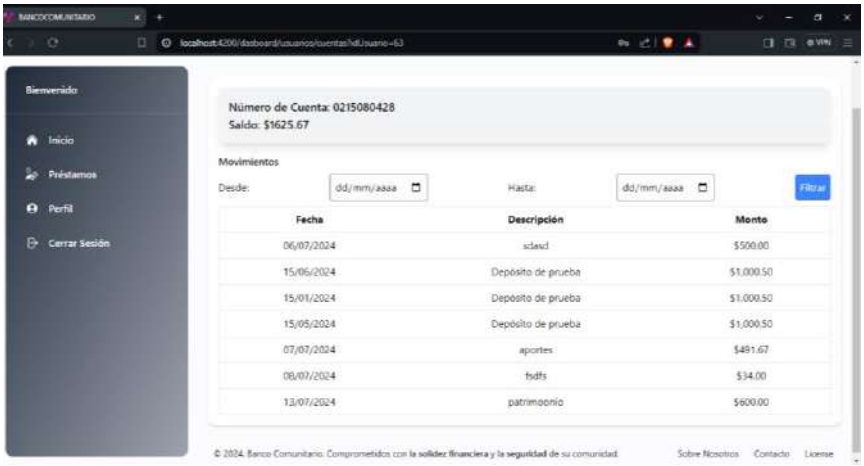


Figura 51: Seguimiento de transacciones por parte del Usuario

Anexo 6: Acta de Entrega y Recepción

CAJA DE AHORRO Y CREDITO SOLIDEZ FINACIERA "EL CHACO"

ACTA DE ENTREGA - RECEPCIÓN

En el cantón El Chaco, a los 12 días del mes de agosto del dos mil veinte y cuatro , siendo las 10 de la mañana, se procede a la entrega y recepción formal de la **Aplicación Web para Bancos Comunitarios**, desarrollada por el estudiante Torres Marroquin Ruben Dario, en el marco del trabajo de investigación para la obtención del título, a la **Caja de Ahorro y Crédito Solidez Financiera El Chaco**.

1. ANTECEDENTES

Conforme a lo dispuesto por la Dirección de la **Caja de Ahorro y Crédito Solidez Financiera El Chaco**, se formaliza la entrega de la **Aplicación Web para Bancos Comunitarios**, desarrollada para optimizar la gestión de los servicios de crédito y ahorro en la entidad. Esta aplicación ha sido entregada a la institución como parte del trabajo académico, sin contraprestación económica alguna, como un proyecto realizado por el estudiante para la obtención del título de Ingeniero.

2. OBJETIVO DEL ACTO

El objetivo del presente acto es la **entrega y recepción oficial** de la **Aplicación Web para Bancos Comunitarios**, la cual tiene como propósito facilitar la gestión interna de los servicios financieros de la Caja de Ahorro y Crédito, brindando a los socios una herramienta para la consulta, registro y seguimiento de sus productos de ahorro y crédito de manera eficiente, segura y digitalizada.

3. DESCRIPCIÓN DE LA APLICACIÓN

La **Aplicación Web para Bancos Comunitarios** incluye las siguientes funcionalidades principales, que fueron desarrolladas y probadas a lo largo del proceso:

- **Registro de socios**: Capacidad para registrar nuevos socios y mantener actualizados los datos de contacto, productos de ahorro y crédito.
- **Administración de productos financieros**: Funcionalidad para gestionar y asignar productos de ahorro y crédito.
- **Consulta de saldos y movimientos**: Los socios pueden consultar sus saldos de ahorro, crédito y visualizar sus movimientos históricos.
- **Solicitud de crédito**: Los socios pueden llenar formularios para solicitar créditos y hacer seguimiento a su estado.
- **Generación de reportes e informes financieros**: Capacidad para generar reportes detallados de los productos, pagos y saldos.
- **Interfaz amigable y responsive**: La aplicación es accesible desde diferentes dispositivos, incluyendo computadoras, tabletas y teléfonos móviles.
- **Seguridad de datos**: Cumple con las normativas de protección de datos personales y seguridad de la información.
- **Capacitación**: Se proporcionó capacitación básica al personal de la Caja de Ahorro sobre el uso de la aplicación.

Dirección: km. 109 vía Lago Agrio Av. San Luis / fono: 096 054 4422/ 098 453 9718
EL CHACO NAPO ECUADOR

4. PROCESO DE ENTREGA Y PRUEBAS

El estudiante **Torres Marroquin Ruben Dario** realizó la entrega de la aplicación web y realizó una serie de pruebas junto con el personal técnico de la **Caja de Ahorro y Crédito Solidez Financiera El Chaco** para verificar el correcto funcionamiento de todas las funcionalidades implementadas.

Las pruebas realizadas incluyeron:

- Verificación de la correcta creación y actualización de registros de socios.
- Pruebas de funcionamiento de la consulta de saldos y movimientos de los productos.
- Validación del proceso de solicitud de créditos y su seguimiento.
- Revisión de la generación de informes financieros y reportes detallados.
- Evaluación de la interfaz para asegurar la facilidad de uso y accesibilidad.

Todos los aspectos fueron verificados y aprobados por el equipo técnico de la Caja de Ahorro y Crédito, quienes confirmaron que la **Aplicación Web para Bancos Comunitarios** cumple con los requisitos y especificaciones establecidos en el plan inicial.

5. DETALLE DE LA ENTREGA

Se hace constar que el estudiante **Torres Marroquin Ruben Dario** ha entregado los siguientes componentes, documentos y archivos relacionados con la **Aplicación Web para Bancos Comunitarios**:

N°	Detalle	Cantidad
1	Aplicación Web para Bancos Comunitarios (versión final)	1
2	Manual de usuario (digital)	1
3	Documentación técnica (digital)	1
4	Código fuente de la aplicación (archivo comprimido)	1
5	Certificados de pruebas realizadas (digital)	1
6	Reportes de validación de funcionalidades	1

6. DECLARACIÓN DE CONFORMIDAD

Por parte de la **Caja de Ahorro y Crédito Solidez Financiera El Chaco**, el representante Cofre Iza Marco German **Cargo**: presidente, declara recibir la **Aplicación Web para Bancos Comunitarios** en el estado de conformidad, después de haberse realizado las pruebas pertinentes y verificar que las funcionalidades están operativas.

Declaración de la Caja de Ahorro:

"La **Caja de Ahorro y Crédito Solidez Financiera El Chaco** recibe oficialmente la **Aplicación Web para Bancos Comunitarios** desarrollada por el estudiante Torres Marroquin Ruben Dario. Después de las pruebas realizadas y la validación de funcionalidades, se confirma que la aplicación cumple con los requisitos establecidos y está lista para ser implementada en los procesos internos de la institución. A partir de este momento, se considera que la entrega se realiza de forma satisfactoria."

7. DECLARACIÓN DEL DESARROLLADOR (ESTUDIANTE)

El estudiante **Torres Marroquin Ruben Dario**, en su calidad de desarrollador de la aplicación, declara haber entregado el producto final de acuerdo con las especificaciones acordadas en el contrato y en el plan de trabajo previamente establecido. Además, se compromete a prestar soporte técnico por un periodo de 3 meses en caso de que se presenten inconvenientes con el funcionamiento de la aplicación.

8. FIRMAS DE CONFORMIDAD

Cofre Iza Marco German
presidente de la Caja de Ahorro y Crédito Solidez Financiera El Chaco:
C.I.: 0502227945

Torres Marroquin Ruben Dario
Desarrollador(estudiante)
1500956527

Anexo 7: Manual de Usuario

Propósito del Documento

El propósito de este documento es asistir al usuario en el uso adecuado de la **"Aplicación Web para Bancos Comunitarios en el Cantón El Chaco"**, desarrollada para facilitar la gestión de operaciones financieras y administrativas en los bancos comunitarios de la región, asegurando un uso eficiente y seguro de sus funcionalidades.

Índice

Introducción

La aplicación web para Bancos Comunitarios en el Cantón El Chaco ha sido diseñada para facilitar a los usuarios la gestión de servicios bancarios de manera eficiente. Su objetivo es proporcionar una plataforma sencilla y segura, adaptada a las necesidades de los bancos comunitarios. Este manual ofrece una guía clara sobre cómo utilizar el sistema, detallando las principales funciones y beneficios para los usuarios.

Usuarios del Sistema

Los usuarios del sistema, también conocidos como usuarios clientes, son individuos que acceden al sistema para utilizar los servicios que ofrece. Dentro de estos usuarios, se pueden distinguir varios perfiles o niveles, y según el nivel que tengan, tendrán más o menos privilegios durante su interacción con la aplicación web.

Los roles definidos en la aplicación son los siguientes:

- **Administrador**: Este perfil tiene acceso completo a todas las funciones del sistema, incluyendo la configuración y gestión de usuarios y permisos.
- **Usuario Normal**: Los usuarios con este perfil tienen acceso a las funcionalidades básicas de la aplicación, permitiéndoles realizar tareas específicas según su nivel de acceso.
- **Empleado**: Este perfil está orientado a los empleados que gestionan y operan actividades específicas dentro del sistema.

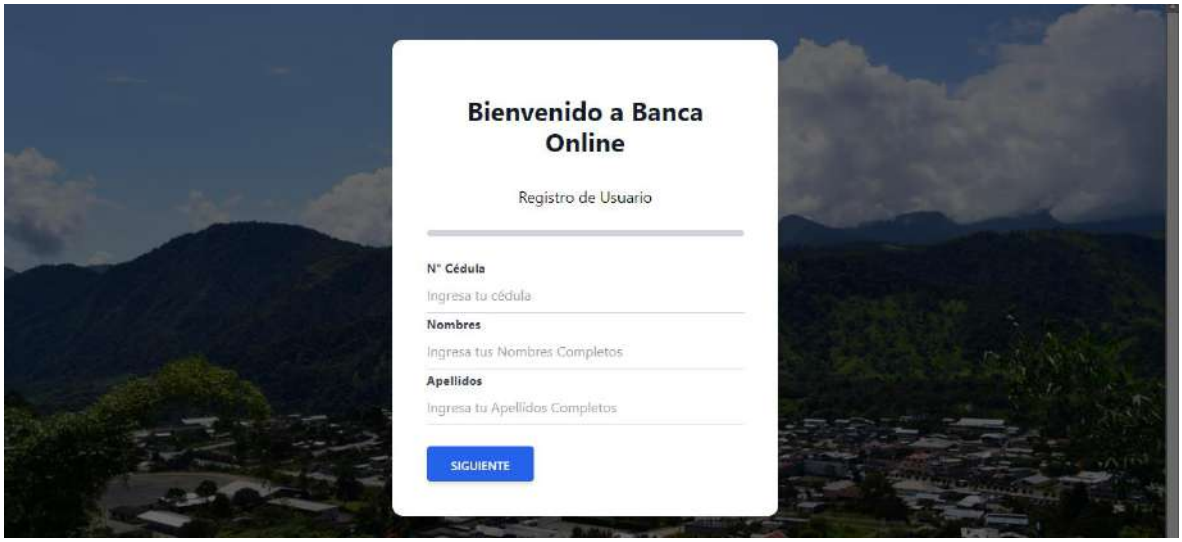
Acceso a la Aplicación

Para acceder a la aplicación web, el usuario debe ingresar en la dirección.

https://solidezfinancieraelchaco.com

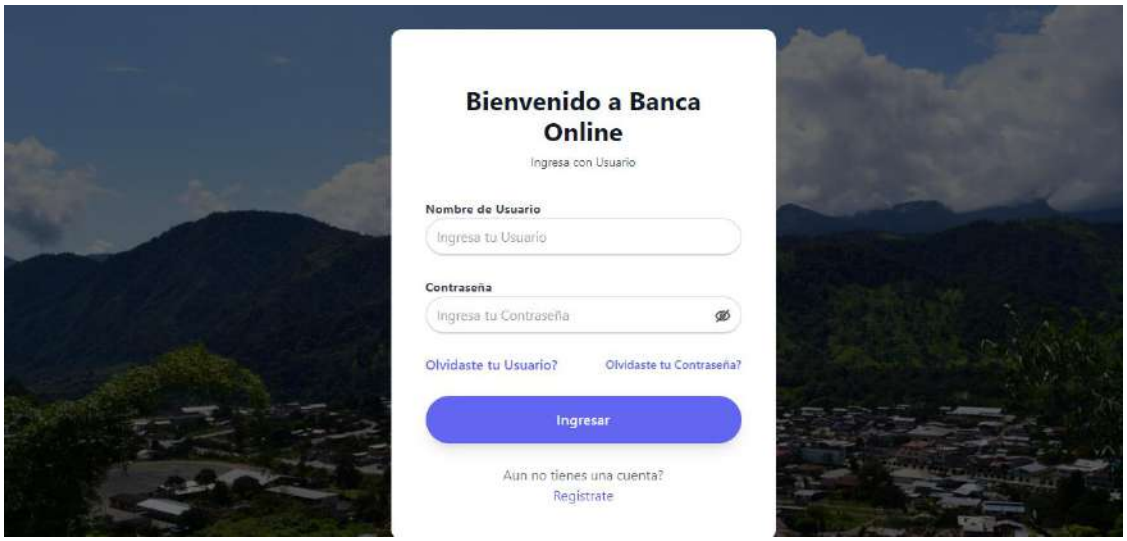
1. Registro de Usuario:

- Dirigirse a la sección de **Registro**.
- Completar el formulario con los datos requeridos.
- Hacer clic en **Registrar** para crear la cuenta.
- Ingresar al sistema utilizando el **usuario** registrado y la contraseña definida.



2. Inicio de sesión:

- Dirigirse a la página de inicio.
- Ingresar el **usuario** y **contraseña** registrados.
- Hacer clic en **Iniciar sesión** para acceder al sistema.

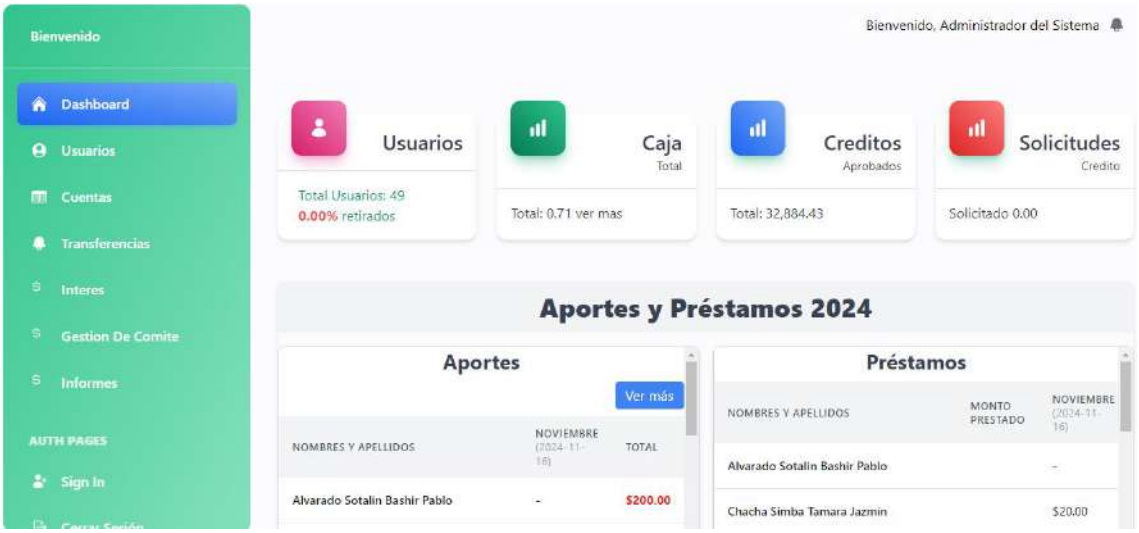


Acceso según el Rol

- **Administrador**

El **Administrador** tiene acceso completo a todas las funcionalidades del sistema. Tras iniciar sesión, podrá acceder al **panel de administración**, donde podrá gestionar usuarios, asignar roles y realizar configuraciones necesarias en la aplicación.

- **Privilegios del Administrador**:
  - Gestión de usuarios (crear, editar, eliminar usuarios).
  - Modificación de configuraciones del sistema.
  - Supervisión y control de todas las funciones de la aplicación.

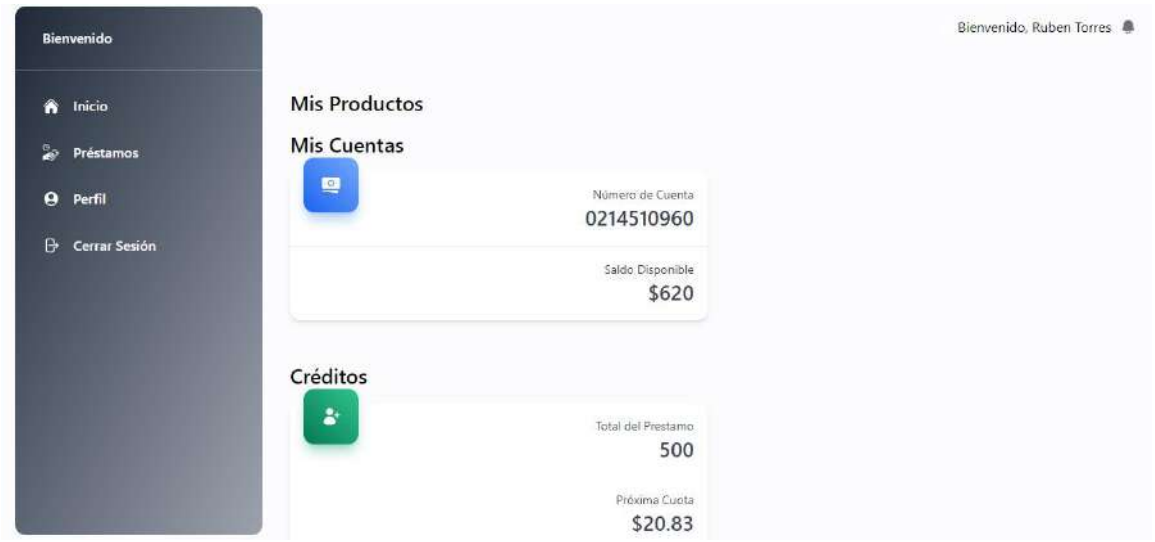


- **Usuario Normal**

El **Usuario Normal** tiene acceso a las funciones básicas del sistema. Después de iniciar sesión, podrá realizar tareas específicas de acuerdo con el rol asignado, pero sin privilegios administrativos.

- **Privilegios del Usuario Normal**:
  - Acceso a la información general.

- Realización de tareas específicas asignadas (consultar información, realizar solicitudes, etc.).



2. Empleado

El **Empleado** tiene permisos limitados y está enfocado en realizar tareas operativas dentro de la aplicación. Al iniciar sesión, el empleado podrá acceder a funciones y datos esenciales para su rol.

- **Privilegios del Empleado**:
  - Acceso a las funcionalidades designadas para su trabajo diario.
  - Realización de actividades operativas específicas.

Estructura de la Aplicación

Las siguientes funcionalidades se encuentran dentro del sistema mediante la siguiente agrupación.

Interfaz

La interfaz de usuario que se emplea en los sistemas de la plataforma "BANCO COMUNITARIO" utiliza un esquema basado en tres componentes principales, que facilitan la operatividad y navegación del usuario:

Menú de Navegación: Ubicado en el lateral izquierdo de la pantalla, agrupa las principales funcionalidades del sistema, entre ellas:

- **Dashboard**: Vista general con datos resumidos.
- **Usuarios**: Gestión de usuarios del sistema.
- **Cuentas**: Administración de cuentas asociadas.
- **Transferencias**: Permite realizar y gestionar transferencias.
- **Interés**: Configuración y seguimiento de tasas de interés.
- **Gestión de Comité**: Módulo para la gestión de comités y sus integrantes.
- **Informes**: Generación de reportes y consultas.
- **Opciones de Autenticación**: Permite al usuario acceder (Sign In) y cerrar sesión.

Área de Procesamiento de Información: Esta sección, ubicada en el centro de la pantalla, proporciona una vista detallada de estadísticas y datos operativos clave, que incluyen:

- **Usuarios**: Muestra el número total de usuarios registrados y el porcentaje de retiros.
- **Caja**: Estado financiero general o balance del sistema.
- **Créditos**: Información sobre los créditos aprobados, incluyendo montos totales.
- **Solicitudes**: Estado de las solicitudes de crédito y montos solicitados.

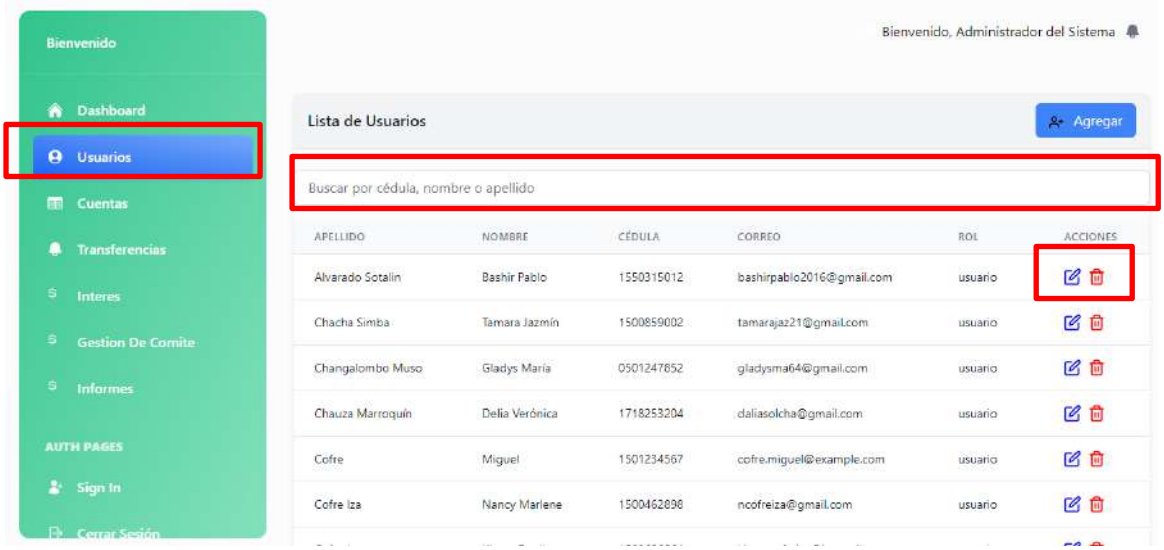
Módulo de Aportes y Préstamos: Esta sección específica, ubicada en la parte inferior central de la interfaz, detalla:

- **Aportes**: Información sobre contribuciones individuales de los usuarios, organizada por mes y año.
- **Préstamos**: Registro de préstamos otorgados a los usuarios, incluyendo el monto prestado y la fecha de registro.

Gestión de Usuarios

Para gestionar el ingreso de los usuarios, el administrador debe acceder al **módulo de Usuarios**. Este módulo se encuentra en el **menú de navegación** ubicado en el **lateral izquierdo** de la interfaz del sistema.

Al hacer clic en el módulo de **Usuarios**, el sistema presenta las siguientes opciones y funcionalidades:



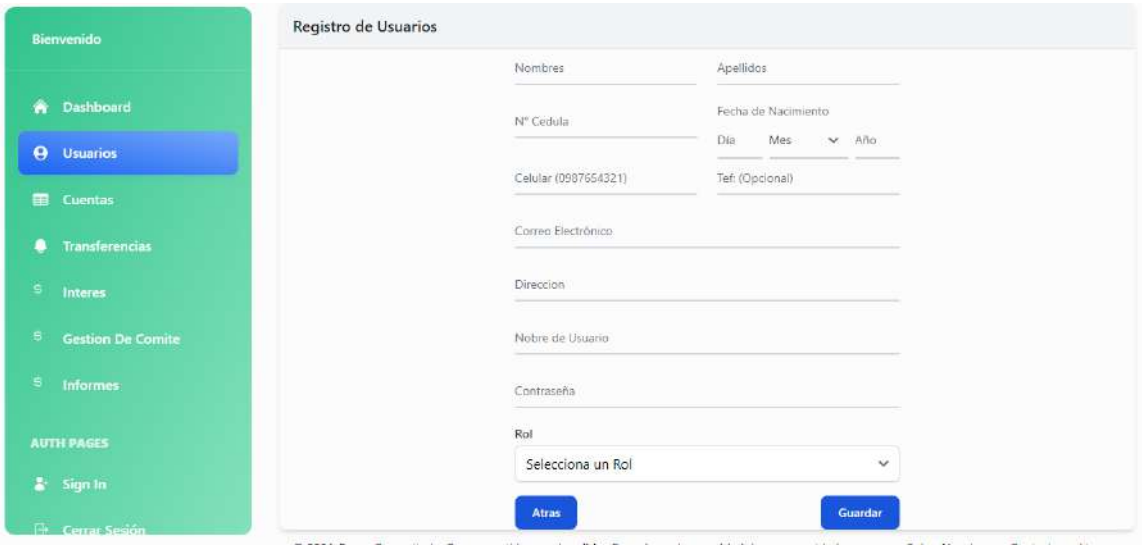
Barra de Búsqueda: En la parte superior de la lista, hay un campo de búsqueda que permite filtrar usuarios por **cédula**, **nombre** o **apellido**, facilitando la localización de usuarios específicos.

Lista de Usuarios: Se muestra una tabla con la información detallada de cada usuario registrado.

Acciones Disponibles:

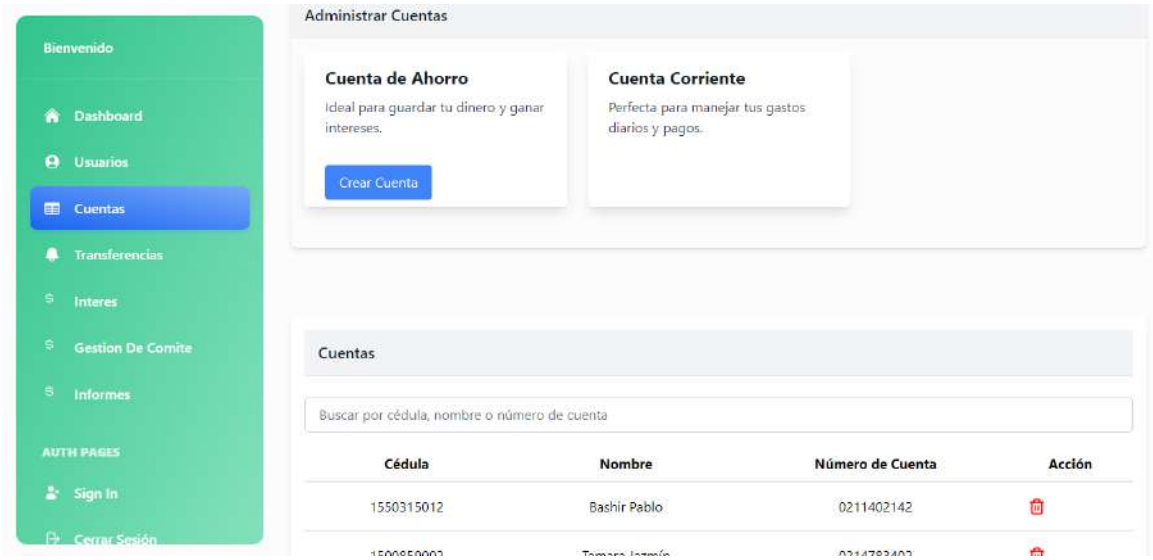
- **Editar**: Icono de lápiz azul. Permite modificar los datos de un usuario. Al hacer clic, se abre un formulario para actualizar la información.
- **Eliminar**: Icono de papelera roja. Permite eliminar al usuario del sistema, con confirmación para evitar errores.

Agregar Usuario: En la esquina superior derecha, haz clic en el **botón Agregar** (color azul). Esto abrirá un formulario para registrar un nuevo usuario, donde se ingresan los datos como nombre, apellido, cédula, correo y rol.



Gestión de Cuentas

En el módulo de **Cuentas** del sistema "BANCO COMUNITARIO," los usuarios pueden administrar las cuentas disponibles, como se muestra en la imagen.



Opciones de Cuenta

Al ingresar al módulo **Cuentas**, se presentan dos tipos de cuentas:

- **Cuenta de Ahorro**: **Crear Cuenta** (azul) para registrar una nueva cuenta de ahorro.
- **Cuenta Corriente**.

Lista de Cuentas Existentes

- **Barra de Búsqueda**: Permite buscar cuentas por **cédula**, **nombre** o **número de cuenta**.
- **Columnas de la Tabla**:
- **Cédula**: Número de identificación del titular de la cuenta.
- **Nombre**: Nombre del titular de la cuenta.
- **Número de Cuenta**: Número asignado a la cuenta del usuario.
- **Acción**: Un ícono de papelera en color rojo que permite eliminar una cuenta específica.

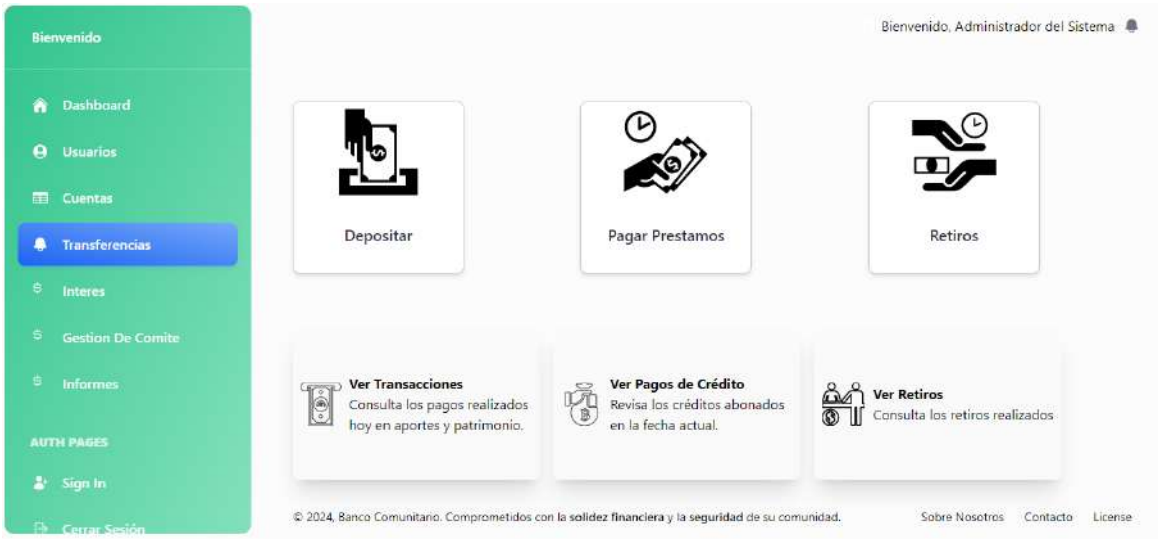
Módulo de Transferencias

El módulo de **Transferencias** en el sistema "BANCO COMUNITARIO" permite gestionar y realizar diversas operaciones financieras relacionadas con las cuentas de los usuarios. Este módulo cuenta con cinco opciones principales, cada una de las cuales abre subopciones adicionales para una administración más específica.

Opción de Aportes

Dentro del **Módulo de Transferencias**, la opción **Aportes** permite gestionar diversas transacciones relacionadas con el dinero de los usuarios. Las funciones disponibles incluyen:

- **Depósitos**.
- **Pagos de Créditos**.
- **Retiros**.
- **Consulta de Transacciones**.

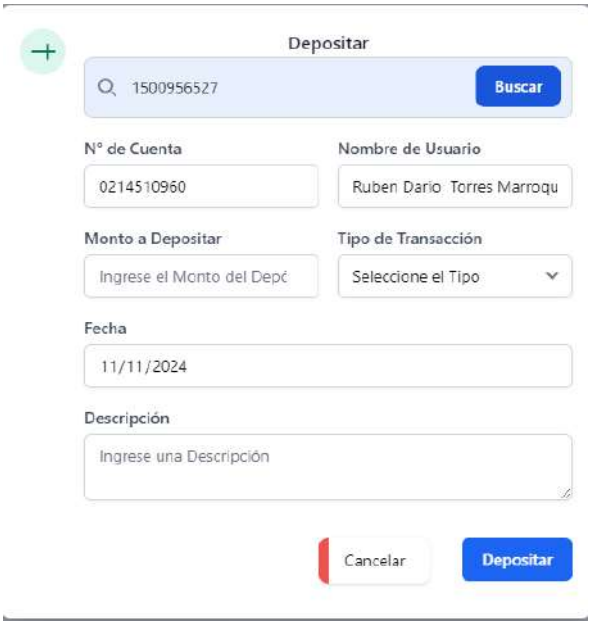


Subopciones de Aportes

1. **Depositar**:

- Ingresar Cédula: Escribir el número de cédula del usuario.
- Monto: Introduc el monto a depositar.

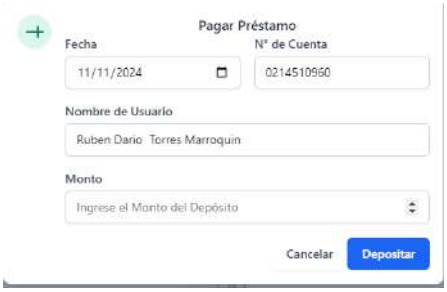
- Tipo de Transacción: Seleccionar el tipo de transacción (por ejemplo, aporte, patrimonio).
- Descripción: Agregar una breve descripción de la transacción.
- Depositar: Clic en Depositar para completar la operación.



2. **Pagar Créditos**:

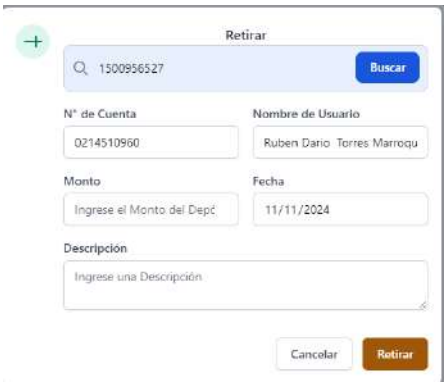
- Ingresar Cédula: Introducir el número de cédula del usuario.

- Seleccionar Crédito: Aparecerán todos los préstamos activos. Seleccionar el crédito que deseas abonar.
- Monto del Pago: Ingresar el monto que a pagar.
- Pagar: Clic en Pagar para completar la transacción.



3. **Retiro**:

- Ingreso de Cédula: Ingresar el número de cédula.
- Monto a Retirar: Ingresar el monto que desea retirar.
- Descripción: Agregar una descripción del retiro.
- Confirmar Retiro: Clic en Retirar para completar la operación.



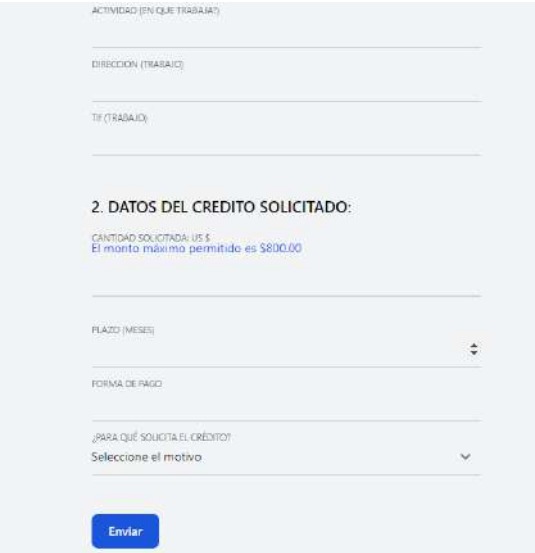
Opción de Realizar Transacciones

Dentro del módulo de **Transferencias**, el sistema permite gestionar diversas actividades relacionadas con los créditos de los usuarios. Las opciones disponibles son:

Subopciones de Realizar Transacciones

1. **Solicitud de Crédito**:

- Seleccionar **"Solicitud de Crédito"**.
- Completar el formulario con el monto solicitado, el plazo de pago y otros datos necesarios.
- Enviar la solicitud



2. **Revisión de Crédito**:

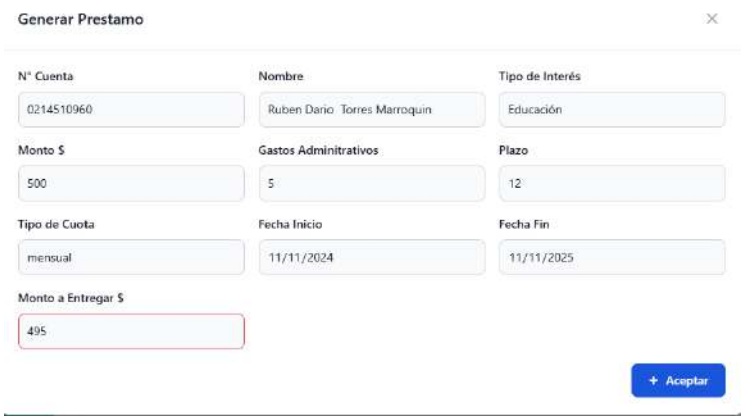
- Seleccionar **"Revisión de Crédito"**.
- El sistema muestra la lista de solicitudes pendientes con detalles del monto solicitado y el estado de cada una

- Aprobar o Rechazar la solicitud según su evaluación

3. **Generar Crédito**:

- Seleccionar "Generar Crédito".
- Ver la lista de solicitudes aprobadas con los detalles correspondientes.

- Confirmar para generar el crédito automáticamente.



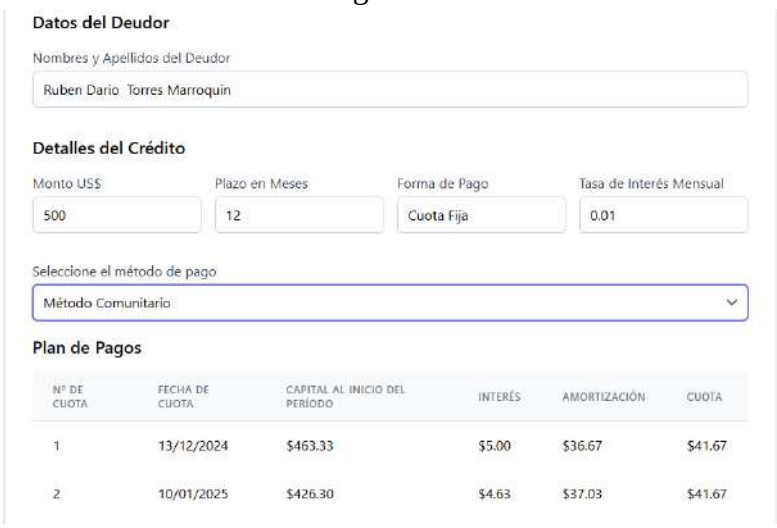
4. **Saldar Crédito Pendiente**:

- Seleccionar "Saldar Crédito".
- Ver la lista de créditos pendientes.
- Elegir el crédito a saldar.
- Hacer clic en "Saldar" para completar el pago.

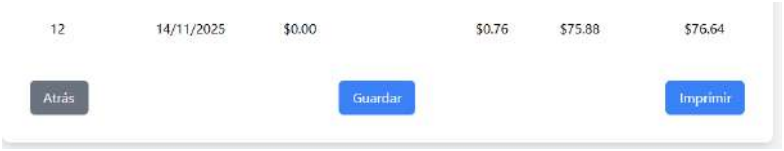
5. **Consultar Crédito**:

- Buscar por número de cédula para ver los créditos asociados al usuario.

- Seleccionar el crédito de interés.
- Hacer clic en "Dar detalles" para ver información completa sobre el crédito.
- Visualizar la tabla de amortización generada automáticamente.



- Hacer clic en "Guardar" para guardar los detalles.
- Hacer clic en "Imprimir" para obtener una copia del contrato con la tabla de amortización.



Opción de Refinanciamiento

En el módulo de Transferencias, la opción **Refinanciamiento** permite gestionar las solicitudes de refinanciamiento de créditos de los usuarios. Esta opción es útil para ajustar las condiciones de un crédito, como el plazo de pago o la tasa de interés.

Pasos para Gestionar el Refinanciamiento

1. **Seleccionar "Refinanciamiento"**: Dentro del módulo de Transferencias, elegir la opción **Refinanciamiento**.
2. **Visualizar Solicitudes Pendientes**: El sistema muestra una tabla con todas las solicitudes de refinanciamiento pendientes. En la tabla se incluyen:

- **Nombre del solicitante**: El nombre del usuario que solicitó el refinanciamiento.
- **Monto del crédito**: El monto original del crédito que se desea refinanciar.
- **Nuevo plazo o nuevas condiciones solicitadas**: Detalles de los ajustes solicitados.

3. **Gestionar Solicitudes**: En cada fila de la tabla, existen opciones para gestionar la solicitud:

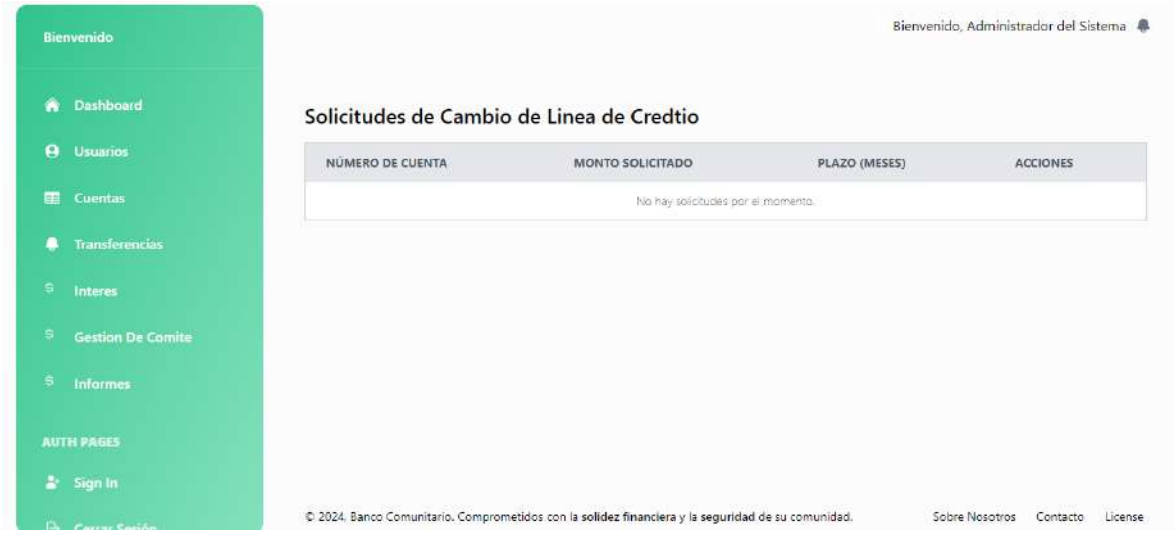
- **Ver**: Seleccionar esta opción para revisar los detalles de la solicitud, incluyendo el nuevo plazo o condiciones solicitadas.
- **Aprobar**: Si la solicitud es viable, seleccionar esta opción para aprobarla. El sistema actualizará las condiciones del crédito del usuario y notificará al usuario sobre el cambio.

Opción de Novación

En el módulo de **Transferencias**, la opción **Novación** permite gestionar las solicitudes de los usuarios que desean modificar de manera significativa las condiciones de su crédito. A través de esta funcionalidad, los usuarios pueden solicitar un cambio en los términos del crédito, que puede incluir una modificación en el objeto del crédito, el plazo de pago, o incluso la parte involucrada en el contrato.

Opción de Cambio de Línea de Crédito

En el módulo de **Transferencias**, la opción **Cambio de Línea de Crédito** permite gestionar las solicitudes de los usuarios que desean modificar el límite de crédito disponible. Esta opción es útil para aquellos usuarios que necesitan un ajuste en el crédito autorizado, ya sea aumentando o reduciendo el límite, sin que ello implique una modificación en los términos del contrato original.



Módulo de Intereses

El **Módulo de Intereses** permite gestionar las tasas de interés de los productos y servicios financieros. Se pueden añadir, ver y eliminar tipos de interés aplicados a créditos.

Opciones dentro del Módulo de Intereses

1. **Agregar nuevo interés**:

- Hacer clic en el ícono de **signo más (+)**.
- Completar el formulario con el tipo de interés y su valor en porcentaje

2. **Eliminar interés**:

- En la columna de **Acciones**, hacer clic en el ícono de **papelera** para eliminar un tipo de interés.

Módulo de Gestión de Comité

El módulo de **Gestión de Comité** permite al administrador organizar y controlar las actividades relacionadas con los comités en el sistema. Este módulo incluye diversas opciones para gestionar miembros, registrar asistencias y administrar pagos.

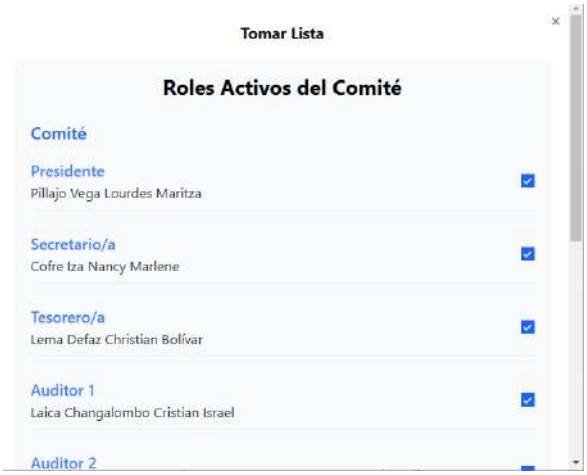
Opciones dentro del Módulo de Gestión de Comité

1. **Asignar Miembros al Comité**:

- Seleccionar la opción "Asignar Miembros".
- Elegir el comité y agregar nuevos miembros.

2. **Asistencia Comité**:

- Seleccionar "Asistencia Comité".
- Marcar la asistencia de los miembros presentes en la reunión.



3. **Pagar**:

- Seleccionar "Pagar".

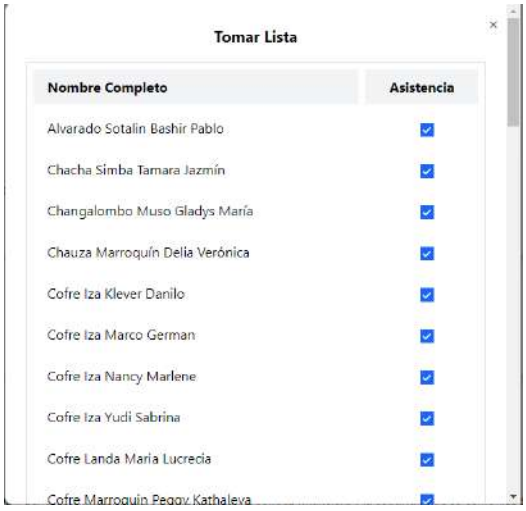
- Registrar los pagos realizados por los miembros del comité.

4. **Crear Usuario Comité**:

- Seleccionar "Crear Usuario Comité".
- Ingresar los datos del nuevo usuario y asignar credenciales.

5. **Asistencia Usuarios**:

- Seleccionar "Asistencia Usuarios".
- Marcar la asistencia de los usuarios asociados al comité.



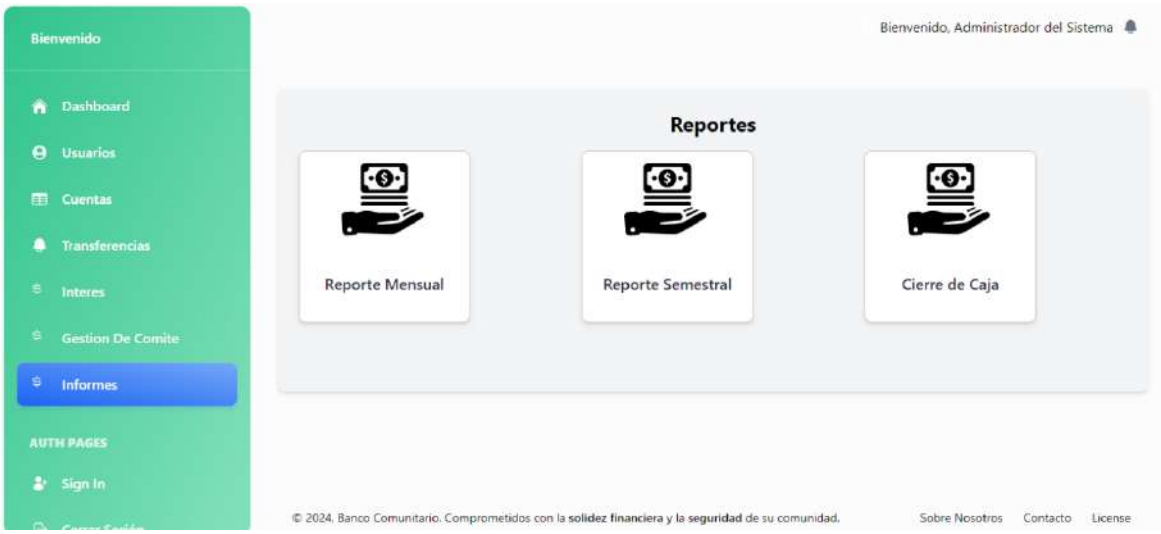
6. **Comité Usuarios**:

- Seleccionar "Comité Usuarios".
- Ver miembros asignados a cada comité.

- Editar: Seleccionar el miembro y realizar los cambios necesarios.
- Eliminar: Seleccionar el miembro y hacer clic en la opción de eliminar.

Módulo de Informes

El módulo de **Informes** en el sistema permite generar reportes financieros y operativos que facilitan el análisis de la actividad económica del banco comunitario. Este módulo cuenta con tres opciones principales:

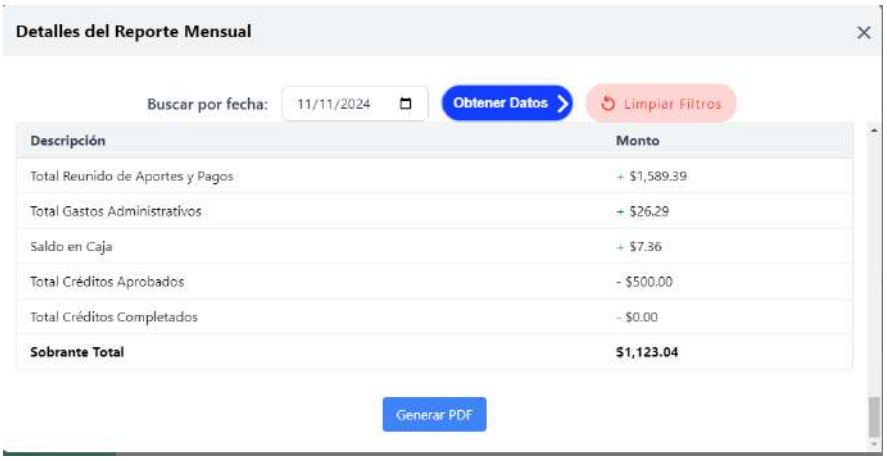


Opciones dentro del Módulo de Informes

1. **Reporte Mensual**:

- Seleccionar "Reporte Mensual".
- Seleccionar Fecha

- Generar el informe con las transacciones del mes.
- Ver los resultados del mes.



2. **Reporte Semestral**:

- Seleccionar "Reporte Semestral".
- Generar el informe de los últimos seis meses.

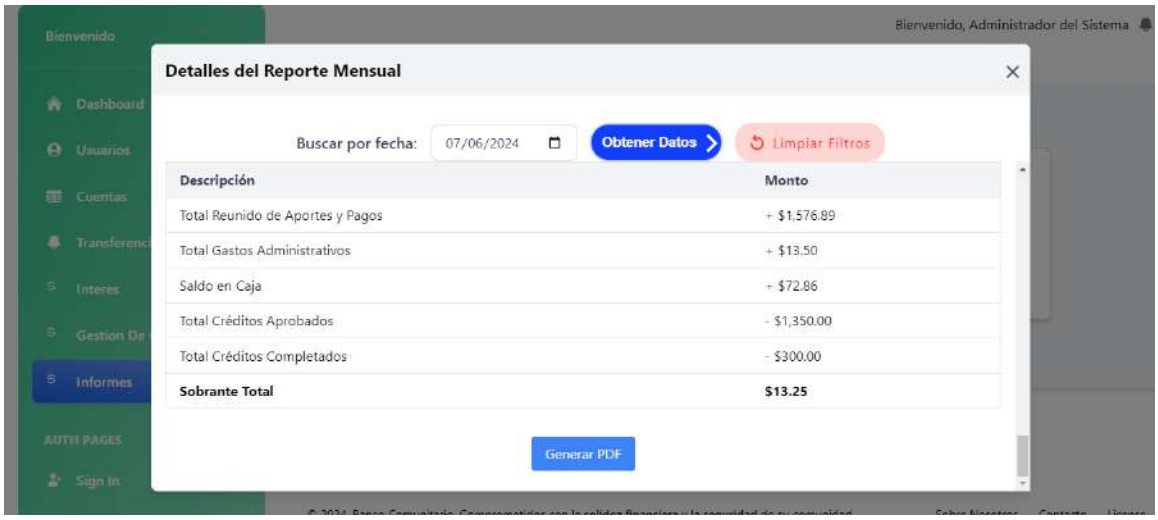
- Evaluar tendencias y resultados a medio plazo.

3. **Cierre de Caja**:

- Esta función permite generar un informe de cierre de caja, que registra las operaciones al final del día o de un periodo de tiempo específico. Es importante para validar y verificar las transacciones diarias, asegurando que los registros financieros estén actualizados y sean precisos.

4. **Exportación en PDF**

Cada informe generado en el módulo de **Informes** (Reporte Mensual, Reporte Semestral y Cierre de Caja) tiene la opción de exportarse en **formato PDF**. Esta funcionalidad facilita la distribución, impresión y almacenamiento de los reportes de manera organizada y segura, asegurando que la información esté disponible en un formato estándar y fácil de leer para la revisión y auditoría.



Cerrar Sesión

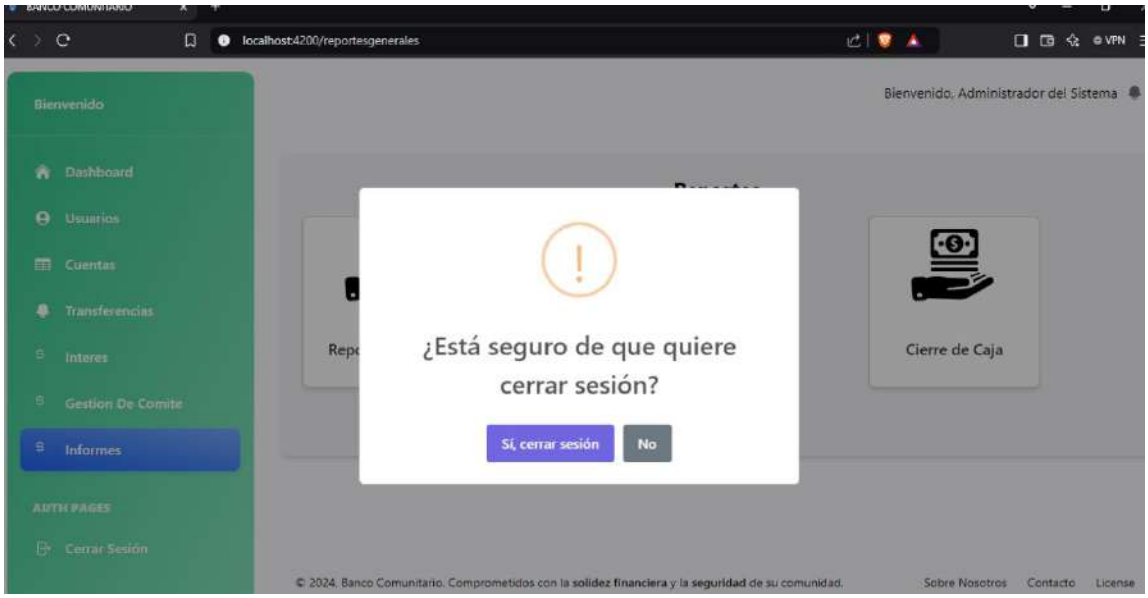
La opción de **Cerrar Sesión** permite al usuario salir de manera segura del sistema, garantizando que no quede ninguna sesión activa en el dispositivo utilizado. Esta funcionalidad se encuentra disponible en el menú lateral izquierdo, en la sección **AUTH PAGES**.

1. **Seleccionar "Cerrar Sesión"**: Ubicar esta opción en el menú lateral izquierdo, bajo la sección *AUTH PAGES*.
2. **Confirmar acción**:

- Al hacer clic, aparece un cuadro de confirmación con el mensaje: *¿Está seguro de que quiere cerrar sesión?*

3. **Elegir una opción**:

- **Sí, cerrar sesión**: Finaliza la sesión y redirige a la pantalla de inicio de sesión.
- **No**: Cancela la acción y permanece en la página actual.

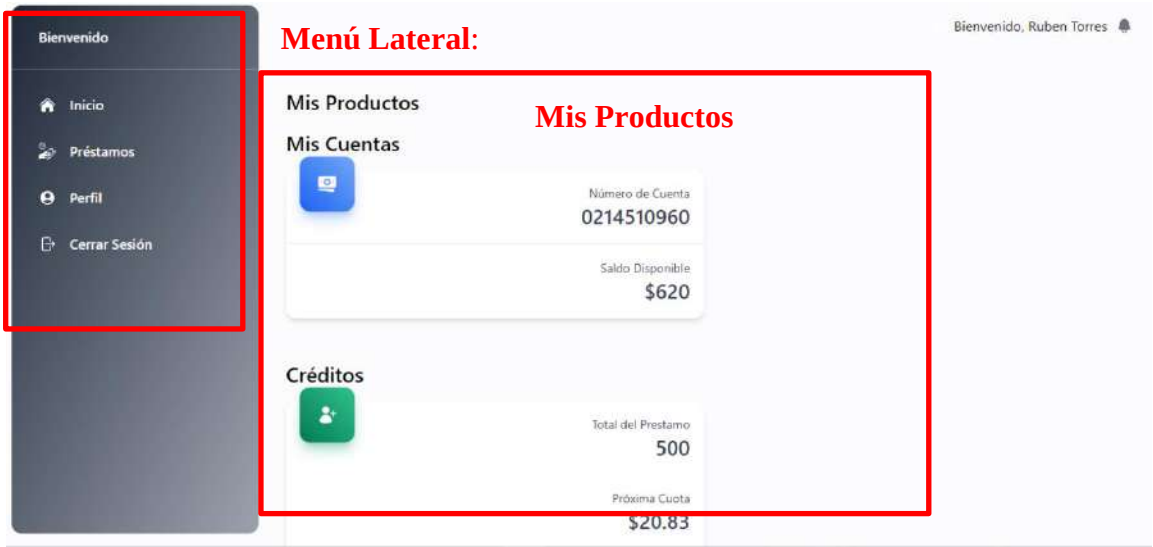


Página Principal - Usuario

Al iniciar sesión, la página principal muestra un resumen de tus productos financieros y un menú de opciones para navegar por el sistema.

3. Elementos principales de la Página:

1. **Mis Cuentas**:
- **Número de Cuenta**: Muestra tu número de cuenta bancaria.
- **Saldo Disponible**: Indica el monto disponible en tu cuenta.
2. **Créditos**:
- **Total, del Préstamo**: El monto total del préstamo que has solicitado.
- **Próxima Cuota**: El monto y la fecha de tu próximo pago.
3. **Menú Lateral**:
- **Inicio**: Regresa a esta página principal.
- **Préstamos**: Accede a herramientas y gestiones relacionadas con créditos.
- **Perfil**: Modifica tu información personal.
- **Cerrar Sesión**: Finaliza tu sesión de manera segura.



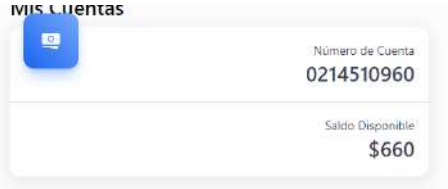
Mis Cuentas - Consulta de Movimientos

Desde la página principal, puedes consultar los movimientos recientes de tu cuenta para llevar un mejor control de tus finanzas.

Acceso a Movimientos:

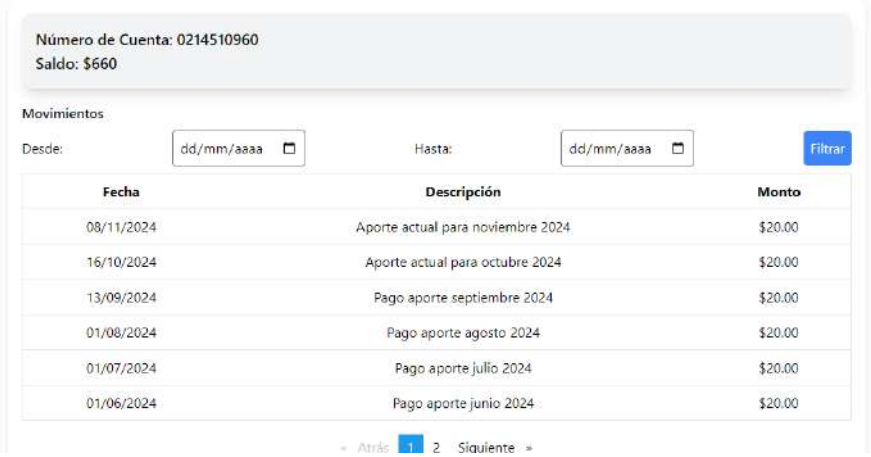
1. Acceder a "Mis Cuentas".

Desde la página principal, seleccionar el número de cuenta deseado en la sección **Mis Cuentas.**



2. Visualizar movimientos.

Se redirige a una nueva pantalla que muestra una tabla con los movimientos recientes.



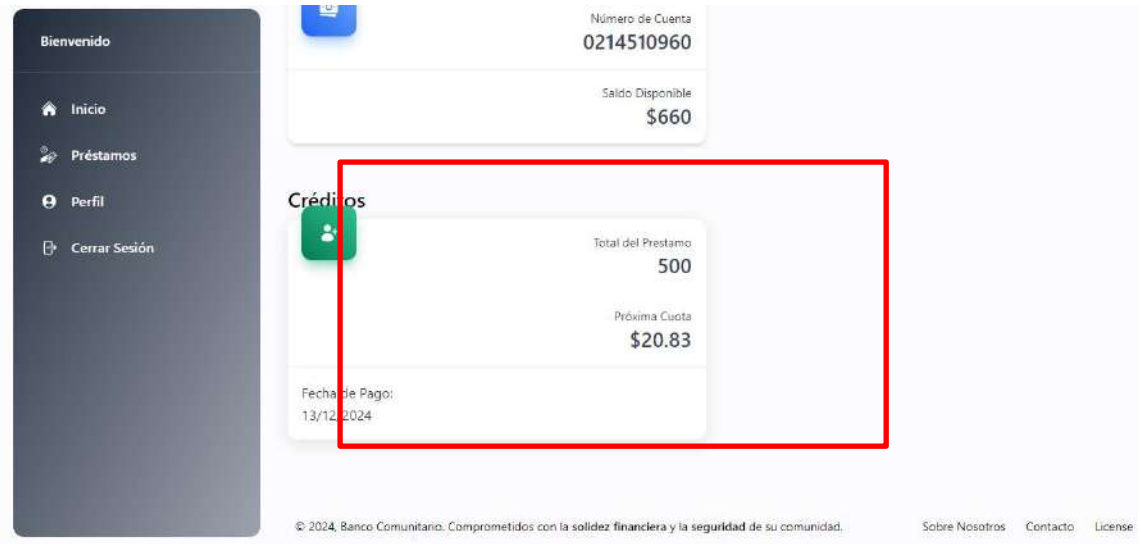
Elementos principales de la pantalla de movimientos:

- **Tabla de Movimientos**:
  - **Fecha**: Indica cuándo se realizó el movimiento.
  - **Descripción**: Detalla la naturaleza del movimiento (depósito, retiro, transferencia).
  - **Monto**: Muestra el valor del movimiento (positivo o negativo).
- **Filtro por Fecha**:
  - En la parte superior, seleccionar una fecha inicial y una final.
  - Hacer clic en el botón **Filtrar** para actualizar la tabla con movimientos específicos dentro del período seleccionado.

Créditos - Consulta Detallada de Créditos

Acceso a la Información de Créditos:

1. Desde la pantalla principal, hacer clic en el apartado Créditos.



2. El sistema redirige a una nueva pantalla que muestra el detalle completo del crédito.

Elementos de la Pantalla de Créditos:

Detalle del Crédito:

- **Fecha de Pago**: Indica la próxima fecha programada para el pago.
- **Saldo Inicial**: Muestra el monto original del préstamo.
- **Número de Pagos**: Detalla cuántos pagos han sido realizados y cuántos faltan.
- **Meses de Mora**: Indica si hay retrasos en los pagos y cuántos meses.
- **Deuda Actual**: Refleja el saldo pendiente por pagar, incluyendo capital e intereses.

Historial de Pagos:

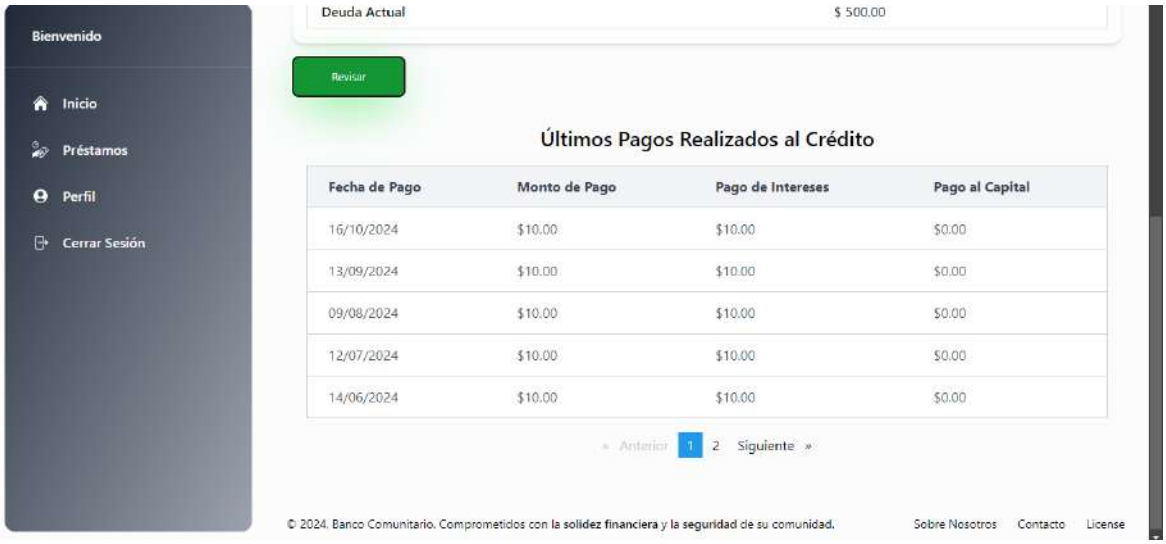
Al hacer clic en este botón, se despliega una tabla con el historial de los pagos realizados.

1. Al hacer clic en **Revisar**, se despliega una tabla con:

- Fecha de Pago: Fecha en que se efectuó el pago.
- Monto de Pago: Total abonado.
- Pago de Intereses: Monto destinado a intereses.
- Pago al Capital: Monto aplicado al capital.

Uso del Historial de Pagos:

- Permite revisar cómo se han distribuido los pagos realizados.



Sección de Préstamos

Desde el menú lateral, selecciona Préstamos para acceder a diferentes funcionalidades relacionadas con tus créditos.

Opciones de Préstamos:

1. **Simulación de Crédito:**

- Seleccionar Simular Crédito.
- Completar los campos necesarios: monto, plazo, y tasa de interés.
- Revisar los resultados y ajustar parámetros si es necesario.

2. Solicitud de Crédito:

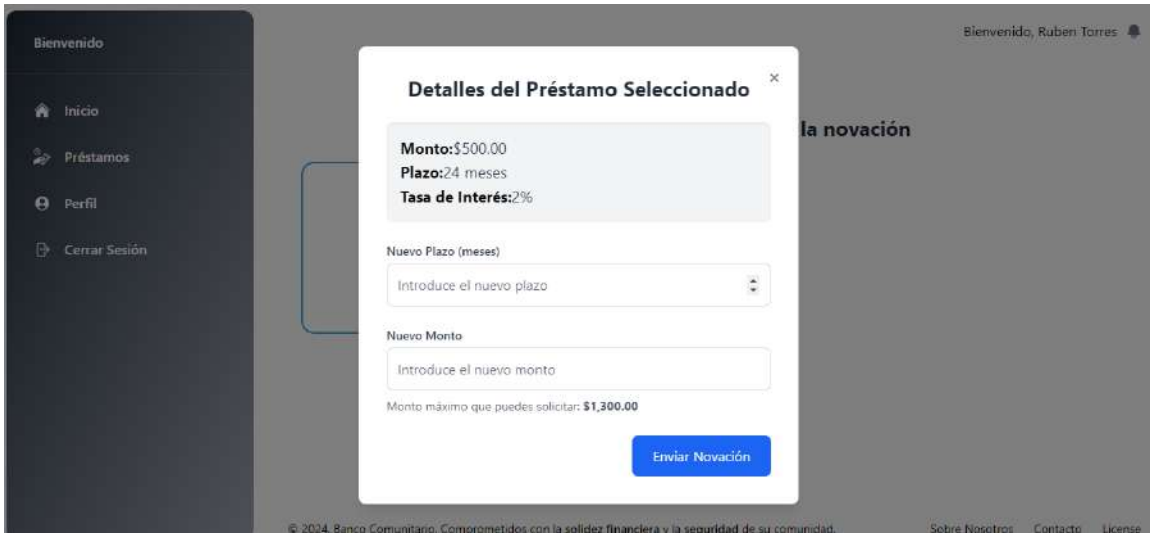
- Seleccionar Realizar Solicitud.
- Completar el formulario con la información requerida.
- Enviar la solicitud para evaluación.

3. Solicitar Novación:

- Seleccionar Solicitar Novación.
- Verificar cumplimiento de requisitos.

- Seleccionar el préstamo.

- Ingresar nuevo plazo y monto.

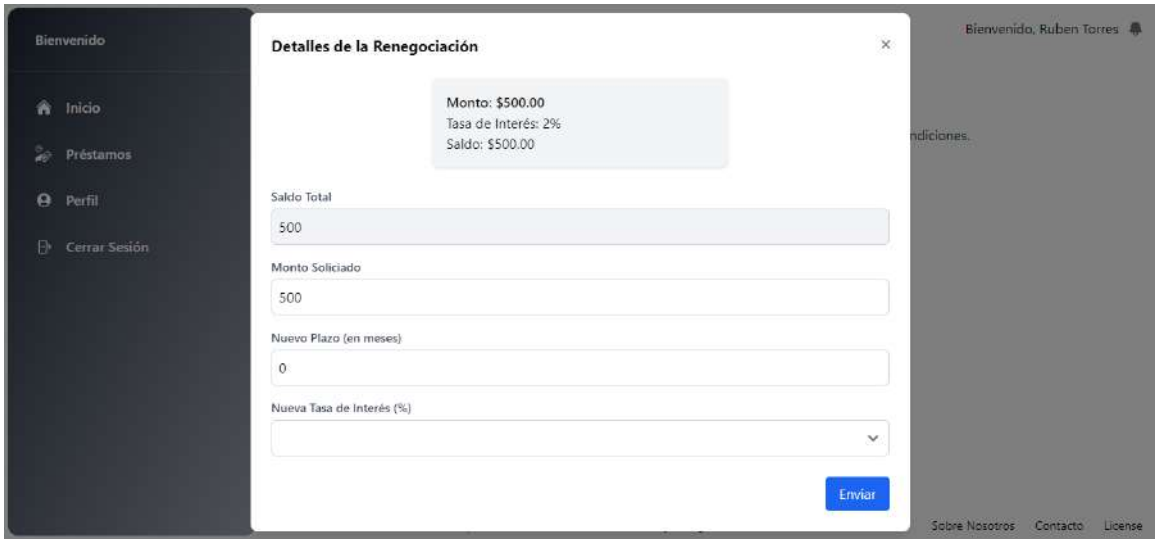


- Enviar la solicitud.

4. Renegociación de Línea de Crédito:

- Seleccionar Renegociar Línea de Crédito.
- Verificar requisitos.

- Elegir crédito para renegociar.
- Ingresar los nuevos términos.



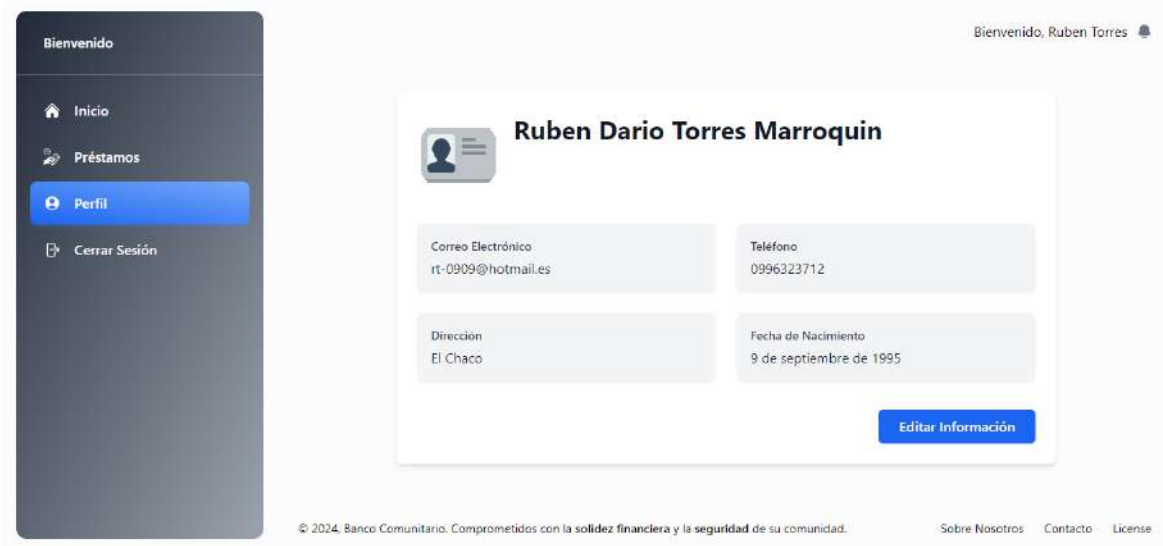
- Enviar la solicitud.

Perfil

Consultar y Actualizar Información Personal

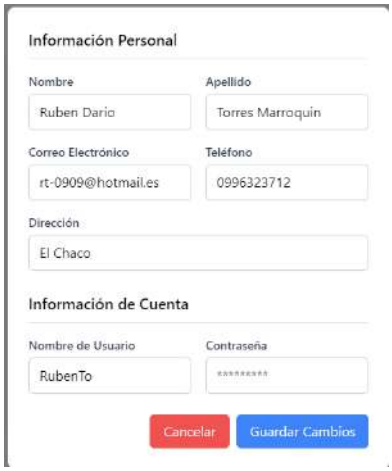
1. Ingresar a Perfil desde el menú lateral.
2. Revisar la información personal:

- Nombre completo.
- Correo electrónico.
- Teléfono.
- Dirección.
- Fecha de nacimiento.



Editar Información Personal

1. Acceder a la opción *Editar Información*.
2. Actualizar los campos permitidos.
3. Confirmar los cambios.



Cerrar Sesión

La opción **Cerrar Sesión** permite a los usuarios finalizar su sesión de manera segura. Al seleccionar esta opción en el menú lateral, el sistema muestra una pantalla de confirmación con el siguiente mensaje:

¿Está seguro de que quiere cerrar sesión?

- **Sí, cerrar sesión:**
- Al seleccionar esta opción, el usuario será desconectado del sistema y redirigido a la página de inicio de sesión, finalizando su sesión de forma segura.
- **No:**
- Al seleccionar esta opción, el usuario permanecerá en su sesión actual y será devuelto a la pantalla previa sin realizar ninguna acción adicional.