Universidad de Matanzas

Facultad de Ciencias Técnicas

Departamento de Informática

PLATAFORMA PARA LA SIMULACIÓN DE SOLICITUD DE CRÉDITOS BANCARIOS DEL BPA MATANZAS

TRABAJO DE DIPLOMA EN OPCIÓN DEL TÍTULO DE INGENIERO INFORMÁTICO

Autor: Ediesky Rivero Pérez

Tutores: MsC. Maily Estopiñan Lantigua

Matanzas, 2024

Resumen

El avance de las tecnologías de la información y comunicación en nuestro país ha ampliado significativamente las posibilidades para llevar a cabo tareas de gran impacto en la economía nacional de manera más eficiente, aprovechando las herramientas modernas que estas tecnologías ofrecen. Este estudio se centra en aprovechar estas ventajas al desarrollar una herramienta web para facilitar la solicitud de créditos en el Banco Popular de Ahorro Matanzas, aportando una simulación crediticia. Para lograr estos objetivos, se emplean métodos científicos tanto teóricos como empíricos para recopilar información. El desarrollo se lleva a cabo utilizando la metodología ágil XP, mientras que para la implementación se utiliza el lenguaje de programación Python y se emplea MySQL como gestor de base de datos. La herramienta se evalúa continuamente para asegurar que cumpla con los estándares de calidad establecidos, obteniendo resultados satisfactorios. Esta solución proporciona a la empresa y a sus clientes independencia, comodidad y seguridad, mejorando significativamente la eficiencia en las actividades comerciales. Enmarcadas en el proceso de bancarización implementado por nuestro país.

Palabras claves: Simulación, crédito, herramienta web, banco, Python.

Abstract

The advancement of information and communication technologies in our country has significantly expanded the possibilities to carry out tasks of great impact on the national economy more efficiently, taking advantage of the modern tools that these technologies offer. This study focuses on taking advantage of these advantages by developing a web tool to facilitate the application for loans at the Banco Popular de Ahorro Matanzas, providing a credit simulation. To achieve these objectives, both theoretical and empirical scientific methods are used to collect information. The development is carried out using the agile XP methodology, while the Python programming language is used for implementation and MySQL is used as the database manager. The tool is continually evaluated to ensure that it meets established quality standards, obtaining satisfactory results. This solution provides the company and its clients with independence, comfort and security, significantly improving efficiency in commercial activities. Framed in the banking process implemented by our country.

Keywords: Simulation, credit, web tool, bank, Python.

Contenido

Introducción

Desde la Revolución Industrial el mundo ha experimentado una carrera vertiginosa hacia la generación permanente de información diversa. La economía tradicional concentrada en la creación de bienes ha sido desplazada paulatinamente hacia una economía de los servicios, basada en la gestión del conocimiento. La toma de decisiones financieras tanto para personas naturales como jurídicas sufren cada día la urgencia de la inmediatez.

Como nunca antes el popular consejo de Benjamín Franklin *time is money* impulsa a la dinamización de las estrategias comerciales, que en el presente solo se logra mediante la digitalización de los procesos claves: una compleja simplificación de procedimientos con el único propósito de ampliar la participación empresarial en los diferentes nichos de mercados. Quien más simple y atractiva realice esta metamorfosis tendrá la ventaja competitiva en un entorno socioeconómico signado por la proliferación de actores que ofrecen productos y servicios similares.

Uno de los sectores que ha migrado con éxito hacia estas tendencias es precisamente el financiero. Con una historia ancestral, la banca ha evolucionado año tras año y se ha insertado en la digitalización de las sociedades contemporáneas con un protagonismo innegable. Tarjetas magnéticas, variedad de créditos y productos de ahorro o inversión a corto, mediano y largo plazo transitan día a día por las secuencias de 0 y 1, en los más disímiles escenarios y plataformas, desde la web, los móviles hasta la alucinante nube.

Dentro de las estrategias más novedosas del sector financiero para ahorrar tiempo no solo a clientes, sino también al personal comercial, se encuentra la simulación de financiamientos, una suerte de análisis previo que ofrece al posible cliente una idea a priori sobre la cuantía que puede solicitar según sus ingresos y el tiempo en que desee amortizar el crédito.

En Cuba, el sistema financiero ha transitado por diferentes etapas en su evolución, determinadas por la economía nacional y la situación sociopolítica imperante en cada momento. Los bancos comerciales del actual sistema financiero presentan un mayor grado de especialización, pues consideran la segmentación de mercados existentes, tanto por agentes como por tipo de moneda (Melgarejo, 2019).

Históricamente los financiamientos han sustentado las economías de las grandes, medianas y pequeñas empresas, y así se reconoce en el VI Congreso del Partido, cuyos Lineamientos de la Política Económica y Social del Partido y la Revolución significaron un reto para el sistema bancario cubano.

En el lineamiento No. 2 se plantea: “El modelo de gestión reconoce y promueve, además de la empresa estatal socialista, que es la forma principal en la economía nacional, las modalidades de inversión extranjera previstas en la ley (empresas mixtas, contratos de asociación económica internacional, entre otras), las cooperativas, los agricultores pequeños, los usufructuarios, los arrendatarios, los trabajadores por cuenta propia y otras formas de gestión no estatal, todas las que, en conjunto, deben contribuir a elevar la eficiencia ” (PCC a, 2011).

Por ende, la política monetaria del país debe garantizar dicho modelo. Y así se pauta en el lineamiento No. 53: Prestar los servicios bancarios necesarios, que incluyan el otorgamiento de créditos al sector que opera bajo formas de gestión no estatal, para contribuir a su adecuado funcionamiento, estudiando la creación de cuentas de capitalización para la adquisición de equipamiento y otros destinos (PCC a, 2011).

Finalizando el 2011 se aprobó el Decreto-Ley No. 289 y tres resoluciones complementarias del Banco Central de Cuba (BCC), en esta dirección. A partir de esta fecha se inició la concesión de créditos a personas naturales para la realización de acciones constructivas por esfuerzo propio en sus viviendas, para los Pequeños Agricultores (PA), las Cooperativas no Agropecuarias (CNA) y los Trabajadores por Cuenta Propia (TCP) (BCC, 2011).

Dichos cambios han aumentado la solicitud de patentes para ejercer la actividad por cuenta propia y el surgimiento de un nuevo segmento de mercado al cual se ha denominado pequeños negocios. El proceso de concesión de créditos a los pequeños negocios exige a la banca cubana desarrollar una estrategia que permita penetrar este mercado y actualizar adecuadamente el proceso de otorgamiento de créditos y evaluación de riesgo (Melgarejo, 2019).

De acuerdo con Melgarejos (2019) si bien las cifras se han incrementado desde 2015 aún resulta insuficiente la cantidad de clientes TCP captados, a pesar del reconocimiento y promoción de las nuevas formas de propiedad y la experticia de los especialistas que se relacionan con la temática para el otorgamiento de financiamientos a personas naturales como nueva forma de propiedad no estatal y otros actores económicos.

Aunque variadas pudieran ser las causas que determinan el bajo nivel de otorgamiento de financiamientos a TCP en el BPA, los estudios precedentes realizados en la institución arrojaron que existen limitaciones relacionadas con lo establecido en el Manual de Instrucciones y Procedimientos, pues tornan el proceso muy engorroso y complejo, por lo general desestimulante para los posibles clientes.

En su tesis de maestría Melgarejos (2019) propone la digitalización del proceso de financiamiento con la inclusión de un módulo para simular los créditos, a fin de facilitar al cliente el proceso, lo que disminuiría considerablemente el tiempo de espera entre el análisis previo y la solicitud.

Actualmente ninguno de los bancos integrantes del sistema bancario cubano cuenta con el servicio de simulación de créditos dentro de sus plataformas digitales. De ahí que teniendo en cuenta la relevancia del tema en medio de la campaña nacional por la Bancarización de la sociedad, la presente investigación defina como **problema científico:**

¿Cómo contribuir a la simulación de créditos a los diferentes actores económicos en el Banco Popular de Ahorro de Matanzas?

Como hipótesis se plantea:

Si se diseña e implementa un módulo para la simulación de créditos del BPA en una aplicación web, se dinamizaría la digitalización de este proceso clave dentro de la organización y facilitaría el acceso de los clientes a estos productos financieros.

Variables:

- Independiente: Simulación de créditos.
- Dependiente: Otorgamiento de créditos.

Se asume como **objeto de estudio**: El proceso de otorgamiento de créditos a los diferentes actores económicos en el Banco Popular de Ahorro de Matanzas.

De acuerdo con lo planteado anteriormente, se establece como **objetivo general**: desarrollar una aplicación web para la simulación de créditos a los diferentes actores económicos en el Banco Popular de Ahorro y como **objetivos específicos**:

- Realizar un estudio del arte sobre las tendencias actuales de la simulación de financiamientos en la gestión de la banca a nivel internacional.
- Diseñar el módulo para la simulación de créditos del BPA de Matanzas en una aplicación web.
- Implementar la plataforma para la simulación de solicitud de créditos del BPA de Matanzas.
- Seleccionar las herramientas adecuadas para el desarrollo del sistema.

Validar el módulo para la simulación de créditos del BPA de Matanzas en una aplicación web. Para el desarrollo de la investigación se utilizaron diversos métodos tales como:

Métodos Teóricos:

- **Analítico-sintético:** para el análisis de la bibliografía necesaria y el estudio de los principales elementos que componen la simulación de financiamientos, lo cual, permite una mejor comprensión del problema a resolver y así arribar a conclusiones que sustenten la necesidad de realizar esta investigación.
- **Inductivo-deductivo:** para realizar el estudio de las principales herramientas y tecnologías existentes para los sistemas de simulación de créditos, revisando sus características propias, y según las mismas, definir las que debe cumplir el sistema que se propone en el presente trabajo de diploma.
- **Histórico-lógico:** para realizar un análisis de las tendencias actuales y soluciones similares en los sistemas informáticos para la simulación de créditos bancarios.
- **Modelación:** para la elaboración de múltiples diagramas, lo que permite un mejor entendimiento del problema y solución que se propone.

Métodos Empíricos:

- **Consulta bibliográfica:** para consultar las fuentes de información relacionadas con los diversos tipos de sistemas informáticos para la simulación de créditos bancarios.
- **Generalización:** permite estructurar los elementos más significativos en cada capítulo de la investigación y arribar a conclusiones más objetivas y explícitas.

Técnicas para la obtención de información:

- **La entrevista:** para realizar encuentros planificados con el cliente y de esta forma obtener información referente solicitudes y tipos de financiamientos, fomentando una buena comunicación con el equipo de desarrollo

La presente investigación tiene un aporte metodológico al proponer el perfeccionamiento o la mejora del Manual de Instrucciones y Procedimiento que permita viabilizar el proceso de determinación del riesgo crediticio, para que incida en un mayor nivel de otorgamiento de financiamientos en el Banco Popular de Ahorro en Matanzas.

El informe de investigación se estructura en tres capítulos, distribuidos de la siguiente forma:

Capítulo 1. Fundamentación teórica: en este capítulo se precisan las tendencias actuales de la simulación crediticia a nivel internacional y los retos de la bancarización en Cuba. Se elabora el marco teórico de la investigación. Se analizan y se exponen las características principales que argumentan la propuesta y permiten un acercamiento al objeto de estudio como las principales tendencias, tecnologías, metodologías y softwares utilizados en la actualidad.

Capítulo 2. Análisis y diseño del sistema: en este capítulo se presentan los fundamentos que sostienen al sistema propuesto, se describen sus funcionalidades y características esenciales, proceso que será guiado por la metodología de desarrollo seleccionada. En el mismo se realiza el modelo de dominio para describir las principales entidades que intervienen, se obtienen los requisitos funcionales y no funcionales, se define además la arquitectura y los patrones de diseño que se utilizan para el desarrollo del sistema informático.

Capítulo 3. Implementación y pruebas: en este capítulo se describe la fase de implementación del sistema, según la metodología propuesta. Se realizan además una serie de pruebas con el objetivo de entregarle al cliente un producto funcional, que cumpla con los requisitos demandados, verificándose así, el cumplimiento de los objetivos trazados al inicio de la investigación.

Este trabajo ofrece conclusiones derivadas de la investigación y se expresan las recomendaciones que pueden ser de utilidad en la digitalización de la concesión de financiamientos al sector no estatal, además de la bibliografía consultada y los anexos necesarios.

Capítulo I Fundamentación Teórica

Introducción

En este capítulo se muestra la elaboración del marco teórico de la investigación. Se analizan y se exponen las características principales que argumentan la propuesta y permiten un acercamiento al objeto de estudio como las principales tendencias, tecnologías, metodologías y softwares utilizados en la actualidad para el desarrollo de sistemas de financiamiento.

Principales elementos para el otorgamiento de créditos

Sistemas Financieros

El sistema financiero comprende, tanto los instrumentos o activos financieros, como las instituciones o intermediarios y los mercados financieros donde se compran y venden los activos financieros. Cumple por tanto con la misión fundamental en una economía de mercado, de captar el excedente de los ahorradores (unidades de gastos con superávit) y canalizarlo hacia los prestatarios públicos (unidades de gastos con déficit).

Asimismo, el sistema financiero debe contribuir al logro de la actividad financiera y permitir el desarrollo de una política monetaria activa por parte de la autoridad monetaria. La medida en que se cumplan ambas opciones será una muestra de su eficacia.

Mercados Financieros

Los mercados financieros constituyen uno de los elementos que conforman el sistema financiero. Los mercados financieros son mecanismos mediante los cuales oferentes y demandantes de recursos, canalizan el ahorro hacia la inversión con la participación de intermediarios financieros que las ponen en contacto a través de activos financieros.

Así, los mercados financieros se identifican con los pasos, los procedimientos, el modo de actuación para comprar y vender activos financieros.

Por último, un mercado es tanto más flexible cuanta más facilidad existe para la rápida reacción de los agentes ante la aparición de cambios en los precios de los activos u otras condiciones del mercado. Esta reacción se manifiesta en la pronta aparición de nuevas órdenes de compra y ventas de activos y requiere la existencia de una gran transparencia y rapidez en el suministro de la información sobre las cotizaciones.

Instituciones financieras

Las instituciones financieras son aquellas instituciones cuya razón u objeto social es la compra, venta o administración de activos financieros (Borrás, 2013).

Los intermediarios financieros o instituciones financieras canalizan los ahorros de diferentes personas en préstamos o inversiones. El proceso por medio del cual se acumulan los ahorros en las instituciones financieras y después se prestan o invierten, generalmente se llama intermediación. Un intermediario financiero debe operar dentro de ciertas restricciones legales que se imponen al tipo de préstamos y/o inversiones que se pueden hacer.

La función de prestar y pedir prestados fondos no basta para definir estas instituciones, puesto que tal función es desarrollada por otros muchos agentes, sino que habría que precisar que dicha actividad constituye el eje básico de su actuación, y que, por tanto, están siempre dispuestos a recibir todos los fondos que deseen depositarse en ellos a los tipos de interés anunciados.

Políticas crediticias

Las políticas crediticias definen el mercado objetivo y los productos y servicios financieros provistos para cada segmento de clientes. Además, especifican los criterios y parámetros concretos para la selección de los clientes, evaluación de clientes y finalmente la aprobación de cada uno de los productos crediticios ofrecidos a estos. Por lo tanto, tienen como objetivo controlar el riesgo crediticio: primero, excluyendo a aquellos clientes considerados muy riesgosos para los productos y servicios ofrecidos, y segundo, estableciendo parámetros y límites para la asignación del crédito, manteniendo la calidad de la cartera conforme a las expectativas.

Simulación

La simulación crediticia es una técnica poderosa que le permite evaluar el impacto de diferentes escenarios crediticios en sus resultados financieros. Al utilizar herramientas de simulación de crédito, puede probar y mejorar su estrategia crediticia experimentando con varios factores, como:

- Su puntaje crediticio e informe crediticio.
- Su utilización de crédito y relación deuda-ingresos.
- Su historial de pagos y combinación de crédito.
- Las consultas de crédito y nuevas cuentas.
- Sus objetivos y planes de crédito.

La simulación crediticia puede proporcionarle información y comentarios valiosos sobre su situación y comportamiento crediticio. Algunos de los beneficios de la simulación crediticia son:

- Puede ayudarle a comprender cómo se calculan su puntaje y su informe crediticio y qué factores influyen en ellos.
- Puede ayudarle a identificar sus fortalezas y debilidades crediticias y áreas de mejora.

- Puede ayudarle a establecer objetivos y planes crediticios realistas y alcanzables y a realizar un seguimiento de su progreso.
- Puede ayudarle a comparar diferentes opciones y escenarios de crédito y elegir el que mejor se adapte a sus necesidades y preferencias.
- Puede ayudarle a prepararse para futuros eventos y oportunidades crediticias y evitar posibles dificultades y riesgos.

Antecedentes

La simulación ha sido una herramienta importante en muchos campos de la investigación y la toma de decisiones desde hace décadas. La simulación es una herramienta cada vez más utilizada en el mundo empresarial para prever situaciones y tomar decisiones más acertadas.

La primera empresa que utilizó la simulación fue la compañía de petróleo Standard Oil en la década de 1950. Buscaban simular la capacidad de producción de sus campos de petróleo y gas en diferentes escenarios. Otra empresa que utilizó por primera vez la simulación fue la compañía de automóviles General Motors, en la década de 1960. Querían simular el funcionamiento de sus fábricas y mejorar el flujo de producción, lo que les permitió ahorrar costos y mejorar la calidad de sus productos. En la década de 1970, la compañía de electrónica IBM también comenzó a utilizar la simulación para prever el rendimiento de sus sistemas informáticos y mejorar su diseño.

Desde entonces, muchas otras empresas han adoptado la simulación como una herramienta esencial en sus operaciones, desde empresas de energía, transporte hasta empresas de finanzas y seguros como los bancos para el otorgamiento de créditos.

Características de las herramientas de simulación de créditos

Las herramientas de simulación de crédito son aplicaciones poderosas que le permiten crear y probar diferentes escenarios de su estrategia crediticia. Al utilizar estas herramientas, puede evaluar el impacto de varios factores, como tasas de interés, montos de pago, utilización del crédito, combinación de créditos y más, en su puntaje crediticio y salud financiera. También puedes comparar diferentes opciones y encontrar la mejor para tus objetivos y situación.

Algunas de las características clave de las herramientas de simulación de crédito son:

Personalización: Puedes personalizar tu simulación de crédito eligiendo el tipo y número de cuentas de crédito, el saldo y límite de cada cuenta, el historial y frecuencia de pagos, y la duración de la simulación. También podrás ajustar los parámetros de la simulación según tus preferencias y necesidades.

Visualización: Puede visualizar los resultados de su simulación de crédito mediante el uso de gráficos, tablas y otros formatos. Puede ver cómo cambia su puntaje

crediticio con el tiempo, cómo la utilización de su crédito afecta su puntaje, cómo su historial de pagos afecta su puntaje y cómo su combinación de crédito influye en su puntaje. De esta manera, podrá comprender fácilmente los factores que afectan su puntaje crediticio y cómo optimizarlos.

Comparación: puede comparar diferentes escenarios de su simulación de crédito utilizando la función de comparación. También puede comparar los pros y los contras de cada escenario y ver las ventajas y desventajas de cada opción.

Recomendación: Puede obtener sugerencias sobre cómo mejorar su puntaje crediticio, cómo reducir la utilización de su crédito, cómo diversificar su combinación de crédito y cómo administrar sus cuentas de crédito. De esta manera, podrá seguir las mejores prácticas y estrategias para su situación crediticia.

Existen diferentes tipos de herramientas de simulación de crédito que puede utilizar para probar y mejorar su estrategia crediticia. Algunos de los más comunes y populares son:

Simuladores de puntaje crediticio: Son herramientas que le permiten estimar cómo cambiará su puntaje crediticio en función de diferentes acciones y eventos crediticios, como liquidar su deuda, abrir una nueva cuenta o no realizar un pago.

Algunos ejemplos de simuladores de puntaje crediticio son Credit Karma, Experian y NerdWallet.

- **Credit Karma:** herramienta gratuita de monitoreo de crédito puede ayudarle a mantenerse al tanto de su crédito y detectar cualquier error que pueda afectar sus puntajes. Utiliza análisis de datos y modelos financieros para determinar las mejores tarjetas de crédito, préstamos y oportunidades financieros de cada usuario.
- **Experian:** se utiliza para la gestión de riesgo de crédito. Recopilan información crediticia de diferentes fuentes de reportes, como prestamistas en un informe de crédito.
- **NerdWallet:** es un startup de finanzas personales que ofrece herramientas y consejos que facilitan al cliente pagar una deuda, seleccionar los mejores servicios y productos financieros y enfrentar metas importantes, como la compra de vivienda o ahorros para la jubilación.

Simuladores de informes crediticios: son herramientas que le permiten estimar cómo cambiará su informe crediticio en función de diferentes acciones y eventos crediticios, como disputar un error, eliminar un elemento negativo o agregar un elemento positivo.

Algunos ejemplos de simuladores de informes crediticios son AnnualCreditReport, Credit Sesame y Credit.com.

Algunos ejemplos de simuladores de informes crediticios son:

- **AnnualCreditReport:** es el sitio oficial para obtener sus informes crediticios anuales gratuitos. Este sitio es mantenido por Central Source, LLC. Central

Source, LLC está patrocinado por Equifax, Experian y TransUnion, por lo que tiene un único sitio donde puede solicitar sus tres informes crediticios gratuitos.

- **Credit Sesame:** es un servicio en línea gratuito que lo ayuda a monitorear y mejorar su puntaje crediticio, administrar su deuda y proteger su identidad. Le ofrece una variedad de herramientas y funciones que pueden ayudarlo a alcanzar sus objetivos financieros y ahorrar dinero.
- **Credit.com:**proporciona su puntaje de crédito y su tarjeta de informe de crédito de forma totalmente gratuita (Otal, S.; Serrano, G.; Serrano, R., 2007)**..**

Simuladores de escenarios crediticios: son herramientas que le permiten comparar y contrastar diferentes escenarios y resultados crediticios, como refinanciar su hipoteca, consolidar su deuda o solicitar una nueva tarjeta de crédito o préstamo. Por lo general, utilizan la información de su puntaje e informe crediticio, así como sus ingresos, gastos y otros datos financieros.

Algunos ejemplos de simuladores de escenarios crediticios son Bankrate y LendingTree

- **Bankrate:** es un sitio web de finanzas personales que tiene una larga trayectoria ayudando a las personas a tomar decisiones financieras inteligentes. Desde el principio, ha proporcionado información sobre tipos de interés. A partir de 2004, Bankrate también comenzó a ofrecer contenidos de educación financiera, cotizaciones de seguros y ofertas de tarjetas de crédito.
- **LendingTree:** es una plataforma en línea en la que puede adquirir y comparar préstamos. La empresa cubre diferentes préstamos en sus servicios, incluyendo préstamos para la vivienda, préstamos personales y para las empresas. Al ofrecer múltiples ofertas de varios prestamistas ayuda a obtener la mejor opción posible en sus préstamos(Otal, S.; Serrano, G.; Serrano, R., 2007)**.**

En resumen, las herramientas de simulación crediticia desempeñan un papel crucial a la hora de probar y mejorar su estrategia crediticia. Ofrecen información, permiten probar escenarios, evaluar riesgos, optimizar la asignación de recursos y mejorar la toma de decisiones. Al aprovechar estas herramientas, puede tomar decisiones informadas, mitigar riesgos y alcanzar sus objetivos crediticios de manera más efectiva.

Los servicios ofrecidos en estos sitios web se proporcionan "tal cual", sin garantías de ningún tipo. Los informes que se obtienen no podemos garantizar que sean completos y precisos.

Metodologías para el desarrollo del software

Las metodologías guían todo el proceso de desarrollo de un software con el fin de hacerlo más eficiente. Una elección incorrecta de la misma propicia la obtención de resultados impredecibles, la detección tardía de errores y la introducción de herramientas que afectan perjudicialmente el proceso. “Un proceso de desarrollo de

software define quién está haciendo qué, cuándo y cómo para alcanzar un determinado objetivo." (James Rumbaugh,2000). En la actualidad existe un gran número de metodologías, las cuales se clasifican en ágiles o robustas. Entre las metodologías ágiles se encuentra XP (*Extreme Programing*) y las robustas RUP (*Rational Unified Process*).

Rational Unified Process(RUP)

El Proceso Unificado de Desarrollo constituye un proceso configurable que se adapta a través de los proyectos variados en tamaños y complejidad como los desarrollados en el departamento de Señales Digitales. RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Características:

- **Dirigido por casos de uso (CU):** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requisitos.
- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción.
- **Iterativo e Incremental:** Aunque pueda parecer que los flujos de trabajo se desarrollan en cascada, RUP propone que cada fase se desarrolle en iteraciones.

Extreme Programing (XP)

La metodología XP consiste en una programación rápida o extrema caracterizada por tener como parte del equipo al cliente o usuario final. Está diseñada para ser usada con equipos de desarrollos pequeños. XP no permite introducir funcionalidades antes de ser necesarias y las mismas son añadidas con retroalimentación continua.

Características:

- La comunicación se establece entre los usuarios y los desarrolladores.
- Establece la retroalimentación constante del equipo de desarrollo.
- El cliente decide lo que se implementa.

Elección de la metodología de desarrollo de software

A partir de las características mencionadas anteriormente se opta por la elección de la metodología de desarrollo ágil XP, debido a que el cliente tiene una participación activa y directa en la solución propuesta. Para el desarrollo de la presente investigación se opta por utilizar una metodología de desarrollo ágil porque, en primer lugar, son más flexibles y adaptables a cambios, lo que significa que pueden

responder de una forma eficaz a requisitos nuevos o cambiantes del cliente. Además, las metodologías ágiles promueven una colaboración más estrecha entre el equipo de desarrollo y el cliente, lo que permite una mejor comprensión de las necesidades y requerimientos del proyecto. Esto, a su vez, conduce a un producto final más útil y satisfactorio para el cliente. Otra ventaja es que permiten un desarrollo más rápido y frecuente de prototipos y versiones del software. Esto significa que los problemas o errores pueden detectarse y corregirse más temprano en el ciclo de vida del proyecto, lo que ahorra tiempo y dinero en correcciones tardías.

Además, la entrega temprana y regular de funcionalidades permite al cliente tener un mayor control sobre el resultado final del proyecto, y debido a esto es necesario una mayor flexibilidad a los cambios de requisitos, lo que reduce el riesgo de desarrollar un producto que no satisfaga las necesidades del cliente. Por último, como el desarrollo es llevado a cabo por un equipo es necesario la comunicación y la transparencia entre todos los miembros del equipo fomentando el trabajo colaborativo y la creatividad.

Tecnologías y herramientas a utilizar

Lenguaje de modelado

El Lenguaje de Modelado Unificado (UML, por sus siglas en inglés) es un lenguaje visual utilizado para modelar sistemas de software. Se utiliza para representar gráficamente diferentes aspectos de un sistema de software, como su estructura, comportamiento y relaciones. El propósito principal de UML es facilitar la comunicación y comprensión entre los diferentes stakeholders y miembros del equipo de desarrollo de software. UML consta de diferentes tipos de diagramas, como diagramas de casos de uso, diagramas de clases, diagramas de secuencia, entre otros, cada uno con su propia sintaxis y semántica. Estos diagramas se pueden utilizar para modelar diferentes aspectos del sistema, como su arquitectura, comportamiento o interacciones entre los componentes (Budgen, 2013).

Herramienta CASE

Son distintas aplicaciones informáticas con el objetivo de aumentar la productividad y de reducir los tiempos de trabajo y costos. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software. Como es conocido, los estados en el ciclo de vida de desarrollo de un software son: Modelación del Negocio, Requerimientos, Análisis y Diseño, Construcción, Pruebas y Despliegue. Las herramientas CASE son un elemento muy importante, que le permite al administrador de un proyecto informático llevar adelante un proyecto de forma eficaz y eficiente. También es un hecho que estas mismas herramientas, como toda Tecnología de la Información se encuentren en continua evolución y exista además una gran variedad de proveedores y productos, cada uno de ellos con sus diferentes aplicaciones y especificaciones. Actualmente existen varias herramientas CASE, entre ellas, Rational Rose, Visual Paradigm y ArgoUML.

ArgoUML

ArgoUML es una herramienta CASE gratuita y de código abierto que se utiliza para modelar sistemas de software utilizando UML (Lenguaje de Modelado Unificado). Es compatible con diferentes plataformas y permite la creación de diferentes tipos de diagramas UML, como diagramas de casos de uso, diagramas de clases, diagramas de secuencia, entre otros. La herramienta también permite la generación de código a partir de los modelos creados y la importación y exportación de modelos en diferentes formatos, como XMI y SVG (ArgoUML, 2023).

A pesar de que ArgoUML es una herramienta gratuita y de código abierto que ofrece muchas funcionalidades útiles, también tiene algunas desventajas que la desechan como alternativa para la presente investigación, entre estas se encuentran:

- **Problemas de compatibilidad:** puede presentar problemas de compatibilidad con algunas versiones de sistemas operativos o con otros programas de software, lo que puede dificultar su uso,
- **Interfaz desactualizada:** Aunque cuenta con una interfaz gráfica de usuario intuitiva, esta se encuentra desactualizada en comparación con otras herramientas CASE disponibles en el mercado,
- **Problemas con el guardado de archivos:** En ocasiones los archivos se corrompen producto a fallas del Sistema Operativo y no pueden ser recuperados.

Rational Rose

Rational Rose es una herramienta CASE utilizada para modelar sistemas de software utilizando UML. La herramienta fue desarrollada inicialmente por Rational Software Corporation y posteriormente adquirida por IBM. Rational Rose permite la creación de diferentes tipos de diagramas UML, como diagramas de casos de uso, diagramas de clases, diagramas de secuencia, entre otros. La herramienta también permite la generación de código a partir de los modelos creados y la importación y exportación de modelos en diferentes formatos.

Rational Rose es compatible con diferentes plataformas y cuenta con una interfaz gráfica de usuario intuitiva y fácil de usar. La herramienta es ampliamente utilizada en la industria de software para el modelado de sistemas de software complejos y para la generación de código a partir de los modelos creados (IBM, 2023).

Sin embargo, esta herramienta no posee licencia gratuita, por lo que se descarta como alternativa para el desarrollo de la presente investigación.

Visual Paradigm

Es una herramienta multiplataforma que facilita la diagramación visual y el diseño de proyectos de un sistema. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. También proporciona abundantes tutoriales de UML, demostraciones interactivas

de UML y proyectos UML. Dentro de sus principales beneficios se encuentran: la navegación intuitiva entre código y el modelo, superior entorno de modelado visual, soporte completo de notaciones UML y diagramas de diseño automático sofisticado. Es una herramienta colaborativa pues soporta múltiples usuarios trabajando sobre el mismo proyecto y a la vez permite realizar el control de versiones.

Además, también cuenta con una comunidad activa de usuarios y desarrolladores que pueden proporcionar soporte y actualizaciones para la herramienta (Paradigm, 2023a; b), por lo que teniendo en cuenta sus características se selecciona como herramienta CASE para la presente investigación.

Marco de Trabajo (Framework)

Los frameworks en el proceso de desarrollo de software representan una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta.

Lado del servidor

Teniendo en cuenta que en la presente investigación será necesario desarrollar algoritmos para el análisis de marcos de trabajos por el lado del servidor basados en el lenguaje de programación Python. Esta elección se debe a que si se emplea un marco de trabajo que no es basado en Python, se hace necesario investigar cómo realizar las integraciones correspondientes de las librerías necesarias al marco de trabajo seleccionado, con lo cual, se pueden presentar dificultades durante este proceso, retrasando así el desarrollo de la investigación. Entre los diferentes marcos de trabajos por el lado del servidor basados en *Python Django, Flask y Pyramid*. A continuación, se describen de manera breve algunas de sus características:

Django

Django es un framework de desarrollo web en Python que facilita la creación de aplicaciones seguras, escalables y mantenibles. Proporciona un conjunto de herramientas para el manejo de bases de datos, el manejo de URLs, la administración de sesiones, el caché y la seguridad, las mismas aceleran y facilitan el desarrollo de aplicaciones web (Foundation, 2023b). Es gratuito y de código abierto. Este marco de trabajo se caracteriza porque permite construir aplicaciones de forma rápida, segura y con menos código (en comparación con otros marcos de trabajos). Además, cuenta con una comunidad de colaboradores muy grande a nivel mundial que se encarga de realizar las debidas actualizaciones tanto del marco de trabajo como de su documentación de forma diaria (Gómez García, 2018).

Pyramid

Pyramid es un framework web de Python que facilita el desarrollo de aplicaciones web de forma flexible y escalable. Se basa en el enfoque minimalista y permite a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a sus

necesidades. Pyramid es conocido por su simplicidad y su capacidad para construir aplicaciones web robustas. Es gratuito y de código abierto, orientado a objetos y usa un sistema de configuración basado en decoradores que permite registrar las vistas, los eventos, los permisos y otros componentes de forma clara y modular. Además, es compatible con varios sistemas de plantillas y se puede integrar con diferentes sistemas de gestión de bases de datos (Team, 2023b).

Flask

Flask es un framework web de Python que permite crear aplicaciones web de manera rápida y sencilla. Proporciona una estructura minimalista pero poderosa que se centra en la simplicidad y la extensibilidad. Flask se utiliza ampliamente en el desarrollo de aplicaciones web y servicios API. Este se caracteriza por ser ligero, sencillo y flexible (Pallets, 2023). Algunas de sus características clave incluyen: enrutamiento y manejo de URL sencillo y flexible, soporte para el uso de plantillas para la generación dinámica de contenido, integración con bases de datos mediante extensiones como SQLAlchemy, manejo de sesiones y cookies, y soporte para la creación de APIs.

Flask se basa en el principio de "hacerlo sencillo permite a los desarrolladores elegir las herramientas y bibliotecas que mejor se adapten a sus necesidades (Grinberg, 2018).

Por las características que se mencionan anteriormente se opta por seleccionar a Django como marco de trabajo por el lado del servidor.

Lado del cliente

Un framework de desarrollo por el lado del cliente es un conjunto de herramientas, bibliotecas y estructuras que facilitan la creación de aplicaciones web interactivas en el navegador del usuario. Estos se centran en la programación del lado del cliente, es decir, en el desarrollo de la interfaz de usuario y la lógica que se ejecuta en el navegador (Mikowski y Powell, 2013). Entre estos se encuentran *Vue.js*, *React* y *Angular*. A continuación, se describen de manera breve algunas de sus características:

Vue.js

Vue.js es un framework de JavaScript de código abierto utilizado para construir interfaces de usuario interactivas y reactivas en aplicaciones web. Se caracteriza por ser progresivo, lo que significa que puede ser adoptado gradualmente en proyectos existentes sin necesidad de reescribir el código por completo. Sus principales características incluyen su enfoque progresivo, lo que permite su adopción gradual en proyectos existentes; su sistema de reactividad, que actualiza automáticamente la interfaz de usuario cuando los datos cambian; el uso de componentes reutilizables que facilitan la creación de interfaces modulares; un conjunto de directivas que amplían la funcionalidad HTML; y una comunidad activa que proporciona soporte, complementos y recursos adicionales. Vue.js se destaca por su simplicidad, flexibilidad y rendimiento, lo que lo convierte en una opción

popular para el desarrollo ágil de aplicaciones web modernas y reactivas (Team, 2023d).

React

React es una biblioteca de JavaScript desarrollada por Facebook que se utiliza ampliamente para crear interfaces de usuario interactivas y eficientes en aplicaciones web. Se basa en un modelo de programación declarativo y componentes reutilizables. Sus principales características incluyen el uso de componentes reutilizables, que permiten la creación de interfaces modulares y escalables; el concepto de Virtual DOM, que mejora el rendimiento al aplicar solo los cambios necesarios en el DOM del navegador; la unidireccionalidad de datos, que simplifica el seguimiento de los cambios y el mantenimiento del estado de la aplicación; JSX, una sintaxis que combina JavaScript y HTML para facilitar la composición de componentes; y una comunidad activa que proporciona soporte, bibliotecas y recursos adicionales (Team, 2023c).

Angular

Angular es un framework de desarrollo de aplicaciones web de código abierto creado por Google. Ofrece un enfoque completo para construir aplicaciones web robustas y escalables. Sus características principales incluyen una arquitectura MVC que permite una separación clara de la lógica de negocio y la representación visual, el uso de componentes reutilizables para construir interfaces modulares, un sistema de inyección de dependencias que mejora la modularidad y testabilidad, directivas integradas que facilitan la manipulación del DOM de manera declarativa, y un enrutador que permite la navegación fluida entre diferentes vistas de la aplicación (Team, 2023a).

Por las características que se mencionan se opta por seleccionar a Vue.js como marco de trabajo por el lado del cliente.

Lenguajes de Programación

Los lenguajes de programación constituyen un conjunto de símbolos, sintaxis y reglas semánticas que permiten establecer una comunicación entre las personas y el ordenador. A través de ellos la persona puede dar instrucciones a la computadora. Existe un gran número de lenguajes de programación, unos del lado del cliente y otros del servidor.

Un lenguaje del lado del cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. Un lenguaje que se ejecuta en el lado del servidor es independiente del cliente, siendo mucho menos rígido respecto al cambio de un navegador a otro.

Python

Es el lenguaje seleccionado para el desarrollo del sistema debido a que es el lenguaje que maneja el marco de trabajo Django. es un lenguaje de programación interpretado, orientado a objetos y de alto nivel, con una semántica dinámica. Sus

estructuras de datos integradas de alto nivel, combi- nadas con tipado dinámico y enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de script o de unión para conectar componentes existentes entre sí. La sintaxis simple y fácil de aprender de Python enfatiza la legibilidad y, por lo tanto, reduce el costo del mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización de código. El intérprete de Python y la extensa biblioteca estándar están disponibles en forma de código fuente o binario sin cargo para todas las plataformas principales, y se pueden distribuir libremente (Foundation, 2023a).

HTML

Es un lenguaje sencillo que permite definir documentos de hipertexto a base de ciertas etiquetas que marcan partes del documento, dándoles una estructura o jerarquía. Presenta el texto de una manera estructurada y agradable, con enlaces que conducen a otros documentos o fuentes de información y con inserciones multimedia (gráficos, sonido, video). El lugar donde se encuentra esta información puede ser el mismo documento o cualquier otro lugar de Internet (Miranda Perea,2008).

Características:

- Es un potente lenguaje orientado a objetos y multiplataforma.
- Consta con una biblioteca de clases base para el desarrollo, pero en aplicaciones de envergadura requiere de frameworks externos.
- Las aplicaciones resultantes necesitan gran cantidad de memoria en las computadoras clientes.

CSS

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para describir la presentación visual y el diseño de un documento HTML. Permite definir estilos, como colores, fuentes, márgenes y posicionamiento, para aplicar a elementos individuales o conjuntos de elementos en una página web (Docs, 2023a).

JavaScript

Java Script constituye un lenguaje interpretado que posibilita añadir a las páginas web dinamismo. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario. Este lenguaje se puede utilizar para crear páginas interactivas en programas como calculadoras y agendas.

Características:

- Compatible con la mayoría de los navegadores modernos.
- Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez.

- JavaScript permite ejecutar instrucciones como respuesta a las acciones del usuario.

Servidor Web

Un servidor web constituye un programa que se ejecuta de manera continua en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un usuario de Internet. Es decir, un servidor sirve de contenido estático a un navegador, carga un archivo y lo brinda a través de la red al navegador del usuario. Los servidores web son fundamentales en las aplicaciones de al lado del servidor que se implementen, pues ellos funcionarán sobre estas aplicaciones. A partir de la elección del marco de trabajo Django, servidores web como *Apache o Nginx* son alternativas viables para el despliegue de la solución.

Ngnix

Nginx es un servidor proxy inverso HTTP y también un email proxy escrito. A la fecha ya lleva más de 5 años corriendo en sitios de tráfico alto. Nginx es una aplicación muy potente que brinda balanceo de carga y estabilidad. Ha sido probado en distintos sistemas operativos como son: Linux y Solaris.

Características:

- Presenta gran escalabilidad.
- Es de configuración básica.
- La mayoría de su documentación se encuentra en ruso.

Apache

Constituye un servidor de código abierto desarrollado en febrero de 1995. Este servidor es construido con parches de código y piezas preexistentes. Apache es un servidor web flexible, robusto, estable y continuamente actualizado. Debido a sus características únicas es uno de los servidores más utilizados en todo el mundo.

Características:

- Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas y parches.

Sistema Gestor de Base de Datos

El Sistema de Gestión de Base de Datos (SGBD) es un conjunto de programas que se encargan de garantizar la seguridad, integridad de los datos y la interacción con el sistema operativo. Los SGBD permiten manejar de una forma clara y sencilla grandes volúmenes de datos.

Un Sistema de Gestión de Base de Datos permite a los usuarios crear y mantener una Base de Datos (BD), por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones. Pueden ser de propósito general o específico (Gómez Ballester,2001).

PostgreSQL

PostgreSQL es un sistema gestor de bases de datos objeto-relacionales. PostgreSQL ha demostrado ser un gestor de bases de datos de código abierto muy avanzado, ofreciendo control de concurrencia multiversión, soportando casi toda la sintaxis SQL (incluyendo transacciones, tipos y funciones definidas por el usuario), contando también con un amplio conjunto de enlaces con lenguajes de programación como pueden ser C, C++, Java, Perl, TCL y Python.

Características:

Su licencia es libre, por tanto, puede ser utilizado, modificado y distribuido de forma gratuita para cualquier fin, ya sea privado, comercial o académico.

- Cuenta con un rico conjunto de tipos de datos incluyendo el array.
- Su administración se basa en usuarios y privilegios.
- Incluye herencia de tablas.
- Incorpora estructura de datos array.

MySQL

Es un SGBD muy conocido y usado por su rendimiento, rapidez, facilidad para ser aprendido y sencillez al ser utilizado, siendo frecuente su utilización en aplicaciones web y plataformas. MySQL es un programa de intercambio que permite conectarse a servidores MySQL, para la realización de consultas y obtención de los resultados. Suele ser combinado para trabajar con PHP, siendo esta mezcla bastante segura (Gónzalez, M. H. Á,2009).

Características:

- Es una aplicación sencilla de instalar y configurar en servidores tanto Windows como Linux.
- Presenta un tamaño del registro sin límite.
- Posibilita un buen control de acceso entre usuario y tablas.
- Lento en el trabajo con base de datos grandes.
- Un gran porcentaje de utilidades no están documentadas.

SQLite

SQLite es un sistema de gestión de bases de datos relacional de código abierto que se caracteriza por ser ligero, autónomo y de uso sencillo. A diferencia de otros

sistemas de gestión de bases de datos. SQLite se implementa como una biblioteca embebida en la aplicación, lo que facilita su integración en una amplia variedad de proyectos y plataformas.

Características:

- SQLite es compatible con la mayoría de las características estándar de SQL y es utilizado en una amplia gama de aplicaciones, desde dispositivos móviles hasta aplicaciones de escritorio.
- Ofrece soporte para transacciones ACID, lo que garantiza la integridad de los datos, y permite la ejecución eficiente de consultas y operaciones en entornos con grandes volúmenes de datos (Consortium, 2023).

Elección del sistema gestor de base de datos

Al analizar las características de los diferentes gestores de bases de datos anteriormente mencionados se determina que PostgreSQL posee un elevado consumo de memoria RAM, este aspecto en ocasiones reduce los tiempos de respuesta del sistema y en computadoras de prestaciones limitadas, además, para realizar su despliegue de forma local se requieren de conocimientos avanzados, por lo cual no es una opción viable, Por otro lado SQLite tiene limitaciones en cuanto al tamaño de datos, variables y su rendimiento puede verse afectado negativamente a medida que la base de datos crece en tamaño y complejidad, además, carece de funciones de seguridad y administración de usuarios, con lo cual se desecha como alternativa viable. Por último, MySQL es fácil de usar, es ligero, es potente y viene integrado en los servidores libres. Posee mayor rendimiento, buenas utilidades de administración y está preparado para manejar grandes volúmenes de datos, además, es altamente personalizable por lo que su despliegue es fácil de realizarse en entornos locales. Por estos motivos se decide seleccionar a MySQL como el gestor de base de datos de la presente investigación.

Entornos de Desarrollo

Un IDE es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o varios de estos. Hoy en día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado. Para la elaboración de la solución se utilizará PyCharm como IDE.

PyCharm

PyCharm es un IDE (Entorno de Desarrollo Integrado) específicamente diseñado para el desarrollo de aplicaciones Python. Proporciona un conjunto de herramientas y características que ayudan a los desarrolladores a escribir, depurar y ejecutar código Python de manera eficiente. Algunas de las características de PyCharm son:

- **Editor de código inteligente:** PyCharm ofrece un editor de código inteligente con resaltado de sintaxis, finalización automática de código, refactorización y navegación avanzada.
- **Depurador integrado:** Permite depurar paso a paso el código Python, establecer puntos de interrupción y realizar inspecciones para detectar y solucionar errores.
- **Administración de proyectos:** PyCharm facilita la creación y gestión de proyectos Python, con herramientas para la organización de archivos, administración de dependencias y configuración de entornos virtuales.
- **Integración con herramientas de control de versiones:** Soporta sistemas de control de versiones como Git, SVN y Mercurial, lo que facilita el seguimiento y la colaboración en proyectos de desarrollo.
- **Testing y cobertura de código:** PyCharm ofrece soporte integrado para la ejecución de pruebas unitarias y pruebas de integración, así como para el análisis de cobertura de código.
- **Análisis estático y detección de errores:** Proporciona herramientas de análisis estático que ayudan a identificar posibles errores, problemas de rendimiento y prácticas de codificación no óptimas.

Conclusiones del capítulo

En el presente capítulo se identificaron soluciones anteriores que se encuentran dentro del campo de acción de la investigación, pero al analizarlos se confirma que no son los adecuados para la entidad por sus características y necesidades particulares. Se analizaron los diferentes grupos de metodologías de desarrollo de software y se optó por emplear la metodología ágil, específicamente, Programación Extrema (XP). De acuerdo al estudio realizado se decide utilizar las siguientes tecnologías y herramientas: el lenguaje de modelado UML y el Visual Paradigm como herramienta CASE para visualizar, construir y documentar los artefactos del sistema, Django como marco de trabajo de desarrollo, Python como lenguaje de programación, MySql como Sistema de Gestión de Bases de Datos y PyCharm como entorno de desarrollo integrado. En sentido general se contribuye a mejorar la comprensión del objeto de estudio y se establecen las bases para las siguientes fases de la investigación.

Capítulo II Análisis y diseño del sistema

Introducción

En el presente capítulo se reflejan las actividades realizadas en los procesos de análisis y diseño de la solución propuesta guiado por la metodología de desarrollo XP, donde se recogen las historias de usuario, costos, plan de iteraciones y plan de entrega. Se realiza la captura de requisitos funcionales y no funcionales, con la cual se provee a los desarrolladores de las principales funcionalidades del sistema.

Propuesta del Sistema

Se propone un sistema llamado Plataforma para la Simulación de Solicitud de Créditos Bancarios del BPA Matanzas, utilizando para su desarrollo herramientas libres. El sistema debe ser capaz de brindar las funcionalidades requeridas para la gestión de los créditos en el Banco Popular de Ahorro.

Roles del Sistema

- **Administrador**: define los parámetros por los que se rige la plataforma para el cálculo de la capacidad de pago, cuadro amortizativo o flujo de caja, lista usuarios y detalles de las simulaciones generadas en la plataforma.
- **Cliente:** genera simulaciones, capacidades de pago o flujos de caja, lista sus simulaciones.

Descripción del Negocio

El objetivo del modelo del negocio es describir los procesos, existentes u observados, con el propósito de comprenderlos. Se especifican qué procesos del negocio soportará el sistema. Además de identificar los objetos del negocio implicados, este modelo establece las competencias que se requieren de cada proceso: sus actores, sus trabajadores, sus responsabilidades y las operaciones que llevan a cabo. Consiste en el conjunto de elecciones hechas por la empresa y el conjunto de consecuencias que se derivan de dichas elecciones.

Para lograr el modelo de negocio se debe comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema, conjuntamente con los problemas actuales de la organización e identificar las mejoras potenciales, asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización y derivar los requerimientos del sistema que va a soportar la organización.

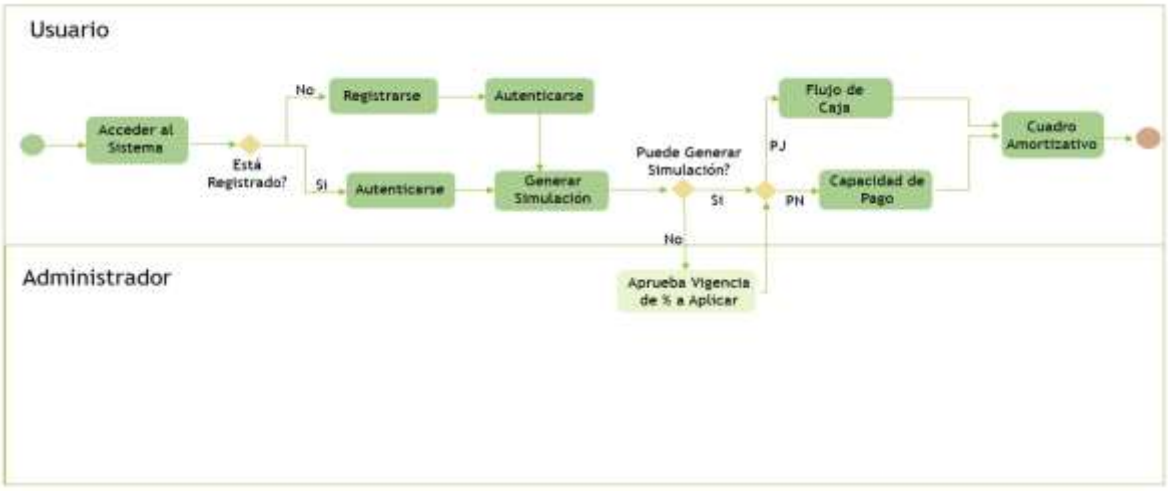

Ilustración 1: Diagrama de Proceso del Negocio

Modelo de la Base de Datos

Un Modelo Conceptual es un componente, ya sea lógico o físico, que contribuye a la comprensión de un problema. Basándose en lo mencionado y considerando el análisis de la información recopilada durante las entrevistas con los clientes sobre los datos, sus características y el flujo de información a abordar, se establece el siguiente Modelo Conceptual de la Base de Datos:

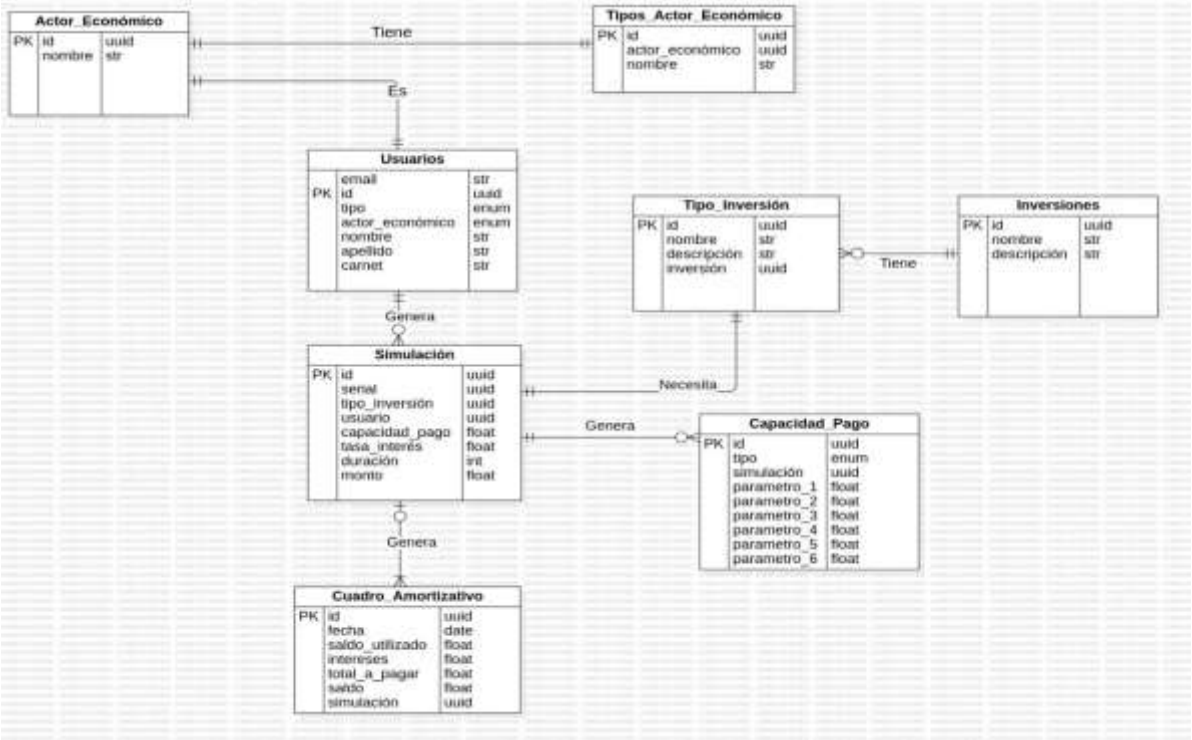

Ilustración 2: Modelo conceptual de la base de datos

Explicación de la solución propuesta

Se propone el desarrollo de una herramienta web que permita a la Dirección Provincial del BPA en Matanzas informatizar la simulación crediticia en la asistencia a sus clientes. La aplicación provee al cliente de una herramienta capaz de facilitar la planificación crediticia. Esto busca mejorar la eficacia de las operaciones en un aspecto fundamental de la entidad y brindar a los clientes un servicio de mayor calidad, comodidad y seguridad.

Fase de Exploración y Planificación

Es la fase inicial de la metodología XP, donde se establece una comunicación continua entre el equipo de desarrollo y el cliente, para obtener los requisitos del sistema. Los requisitos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar. Con el conocimiento de los conceptos asociados al objeto de estudio del problema se comenzará a modelar el sistema que se desea construir. Esto se consigue mediante una descripción de los requisitos funcionales y no funcionales del sistema.

Con el análisis de las características del sistema, los componentes y funcionalidades se obtuvo como principal resultado una primera versión del listado de requisitos. Posteriormente fueron refinados para verificar la existencia de requisitos consistentes y que no se omitieran funcionalidades. Este listado incluye los requisitos funcionales y no funcionales, los cuales se citan a continuación.

Requisitos funcionales

Los requisitos funcionales son las funcionalidades o acciones que el sistema debe ser capaz de cumplir. A continuación, se definen los siguientes:

RF 1: Autenticar usuario en el sistema:

a) RF-1.1: Iniciar sesión.

b) RF-1.2: Cerrar sesión.

RF 2: Gestionar usuarios en el panel de administración

a) RF-2.1: Crear usuario en el sistema.

b) RF-2.2: Listar usuarios existentes en el sistema.

c) RF-2.3: Actualizar un usuario en el sistema.

d) RF-2.4: Eliminar un usuario en el sistema.

RF 3: Gestionar Simulación.

a) RF-3.1 Listar simulación

b) RF-3.2 Adicionar simulación

RF 4: Generar Capacidad de Pago.

a) RF-4.1: Calcular la capacidad de pago

RF 5: Generar Flujo de Caja.

a) RF-5.1: Calcular Flujo de caja

RF 6: Generar Amortización.

a) RF-6.1: Reporte de cuadro amortizativo

b) RF-6.2: Listar amortizaciones.

Requisitos no funcionales

Los requisitos no funcionales son las cualidades que el sistema debe poseer. Estas cualidades se ven como las características que hacen al sistema agradable, usable y confidencial. A continuación, se definen los siguientes:

RNF1: Apariencia o interfaz de usuario

La interfaz gráfica de la aplicación debe concebirse con un ambiente sencillo y de navegación fácil para el usuario. Los colores serán verde claro y blancos por ser los representativos de la entidad bancaria.

Es conveniente no mostrarle al usuario muchos elementos funcionales al mismo tiempo. Se deben tener en cuenta los principios fundamentales de diseño.

RNF 2: Estabilidad

El sistema debe ser capaz de manejar los errores ocurridos durante la ejecución de la misma y avisando de la naturalidad del error.

RNF 3: Usabilidad

Las funcionalidades deben ser claras y se debe mostrar la información de forma lógica y correctamente estructurada.

La configuración debe tener el número mínimo e imprescindible de parámetros de ajustes.

La plataforma debe ser auto gestionable y dinámica.

RNF 4: Seguridad

Se debe garantizar la integridad, disponibilidad y confidencialidad de la información manipulada por la plataforma.

RNF 5: Ayuda y documentación

Se brindarán manuales de ayuda que documenten cómo trabajar de forma adecuada con el sistema.

RNF 6: Hardware

Para el servidor como mínimo una computadora con un procesador de 2.0 GHz o más rápido de x64 bits, 4 GB de memoria RAM y al menos 50 GB de espacio

disponible en el disco duro. Para el cliente como mínimo una computadora con un procesador de 1.0 GHz o más rápido de x64 bits, 2 GB de memoria RAM y al menos 30 GB de espacio disponible en el disco duro.

RNF7: Software

El sistema se ejecutará en los navegadores web Mozilla Firefox, Google Chrome, Opera y Microsoft Edge. Es recomendable actualizar los navegadores a la versión más reciente.

Historias de Usuario

Las historias de usuarios son lo más parecido a los requisitos del producto, sistema o proyecto que se quiere diseñar, construir y poner en marcha. Son descripciones breves y simples de una característica contada desde la perspectiva de la persona que dese la nueva capacidad, generalmente un usuario o cliente del sistema (Cohn, 2004b).

Abarcan o contienen tres aspectos:

- Una descripción de la historia que se usa para la planificación y como recordatorio.
- Conversaciones sobre las historias que sirven para ir extrayendo sus detalles.
- Test que documentan los detalles y permiten validar que la historia está terminada.

A continuación, se muestra un ejemplo de las Historias de Usuarios del sistema:

Tabla 2.1: Descripción de Historia de usuario

HISTORIA DE USO			
Orden	HU_3	**Nombre**	Gestionar usuarios en panel de administración
Riesgo	Bajo	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	0.5
Descripción	El sistema debe brindar la funcionalidad de listar los usuarios, además, adicionar, actualizar y eliminar un usuario en el sistema, todo esto a través del panel de administración del sistema. Se realizará un sistema para la búsqueda de usuarios por nombre de usuario y correo electrónico.		
Observación	El listado debe contener una paginación de registros.		

Descripción de los campos que componen las Historias de Usuario:

- **Orden:** está constituido por dos partes. La primera está referido al nomenclador HU (Historia de Usuario) y la segunda corresponde el número de la funcionalidad que representa.
- **Nombre:** nombre que identificará a la Historia de Usuario.
- **Riesgo:** es el grado de incertidumbre en el desarrollo que se asocia a la Historia de Usuario. Determina la posibilidad real de implementarse o no con las condiciones previstas por el equipo de desarrollo (tiempo, recursos, personal). Puede ser Bajo, Medio o Alto.
- **Prioridad:** la prioridad la define el cliente, y es el grado de importancia que le concede a la funcionalidad.
- **Iteración:** es el número de la fase en la cual se define la Historia de Usuario.
- **Puntos estimados:** es un número entero que representa la cantidad de semanas que se dispone para el desarrollo de la Historia de Usuario. Las Historias de Usuario con altos puntos estimados deben ser separadas en varias tareas. Un punto es una semana efectiva de desarrollo.
- **Descripción:** se escribe una fundamentación de lo que hace la funcionalidad.
- **Observación:** se escribe los elementos o detalles que se deben tener en cuenta para la implementación de la misma.

A partir de la solución propuesta se identificaron 12 requisitos funcionales agrupados en las siguientes Historias de Usuario:

Tabla 2.2: Historias de usuario

NÚMERO	HISTORIA DE USUARIO
1	Diseño y creación de la base de datos
2	Autenticar usuario en el sistema
3	Gestionar usuarios en panel de administración
4	Gestionar Simulación
5	Generar Capacidad de Pago
6	Generar Flujo de Caja
7	Generar amortización

Estimación de esfuerzo por Historias de Usuario y Plan de iteraciones

La estimación de esfuerzo juega un papel fundamental en la actualidad, debido que es de gran importancia para una organización determinar sus necesidades, tiempos de desarrollo y recursos necesarios que le permita tener una planificación aproximada de los proyectos. Así mismo, la estimación de esfuerzo proporciona un apoyo para la toma de decisiones y su fortaleza está basada en la fiabilidad del modelo de estimación y el ciclo de vida del sistema.

Adicional a esto que el desarrollo de software ágil reemplaza eficientemente los métodos tradicionales y es un enfoque iterativo e incremental que mejora los productos y proyectos exitosos gracias al uso de los comentarios de los clientes y la frecuente actualización de la documentación de los proyectos.

Para lograr una mejor organización del trabajo y proporcionar un desarrollo iterativo e incremental, se crea el plan de iteraciones donde se planifica el orden de desarrollo de las Historias de Usuario. Se definió realizar 4 iteraciones, su orden está determinada según las prioridades de las Historias de Usuario y las dependencias existentes entre ellas. La duración total de cada iteración dependerá de los puntos estimados de las Historias de Usuario que en él se desarrollan. Teniendo en cuenta lo expuesto anteriormente la estimación de esfuerzo de las Historias de Usuario y el plan de iteraciones queda como se muestra en la siguiente tabla:

Tabla 2.3: Plan de iteraciones

Iteración	Historia de Usuario	Estimación de esfuerzo
1	Diseño y creación de la base de datos Sistema de autenticación Gestionar usuarios	2 semanas
2	Generar Simulación	1 semanas
3	Generar Capacidad de Pago Generar Flujo de Caja	2 semanas
4	Generar Cuadro Amortizativo	3 semanas
Total: 4	**Total: 7**	**Total: 8 semanas**

A partir de la suma de los puntos de estimación de esfuerzo por cada Historia de Usuario, se calcula que el desarrollo del sistema tendrá una duración de 8 semanas.

Plan de entrega

El plan de entrega en la metodología XP (Extreme Programming) se refiere a la planificación de los incrementos de funcionalidad y las fechas de entrega durante el desarrollo del proyecto. Es un componente esencial para gestionar el flujo de trabajo y garantizar la entrega de software funcional de manera continua (Beck, 2004).

Tabla 2.4: Plan de entrega

Iteración	Historia de Usuario	Fecha de entrega
1	3	7 de mayo del 2024
2	1	21 de mayo del 2024
3	2	28 de mayo de 2024
4	1	12 de junio de 2024

Estimación del Costo

La investigación se realizará entre el 7 de mayo del 2024 al 2 de julio de 2024, por lo tanto, han sido 8 semanas de trabajo. Se realiza un desglose del coste de la investigación, en el cual se incluye el coste de personal, hardware y software. Teniendo en cuenta una jornada laboral de 8h de trabajo se tiene un total de 320 horas de trabajo, distribuidas entre diferentes tareas y diferentes roles profesionales que las llevan a cabo.

Coste de Personal

La metodología de software escogida propone un equipo de desarrollo pequeño donde cada integrante tiene su rol y funciones bien definidas.

Tabla 2.5: Coste de Personal

Fase	Categoría	Horas	Coste/hora	Coste
Documentación	Analista	33	100.00 CUP	3300.00 CUP
Análisis	Analista	40	100.00 CUP	4000.00 CUP
Diseño	Diseñador	57	75.00 CUP	4275.00 CUP
Diseño gráfico	Diseñador gráfico	22	75.00 CUP	1650.00 CUP

Implementación	Programador	146	50.00 CUP	7300.00 CUP
Pruebas	Programador	22	50.00 CUP	1100.00 CUP
Total		**320**		**21625.00 CUP**

Coste de Hardware

Tabla 2.6: Coste de hardware

Equipo	**Coste**	**Coste de amortizado**
Computadora portátil	45200.00 CUP	25764.10 CUP
Teléfono Celular	18540.00 CUP	11712.34 CUP
Total	**63740.00 CUP**	**37476.44 CUP**

Coste de Software

En la realización de esta investigación no existe costo asociado al software debido a que se opta por utilizar Software Libre.

Coste total

A partir del coste de cada uno de los elementos necesarios para la investigación se puede llegar al coste total, como se aprecia en la siguiente tabla:

Tabla 2.7: Coste total

Tipo de coste	**Total**
Coste de personal	21625.00 CUP
Coste de hardware	37476.44 CUP
Coste de software	0 CUP
Total	**59111.44 CUP**

Patrones de Arquitectura

Según Christopher Alexander en el libro “The Timeless Way of Building” define al patrón en la siguiente manera:

“Como elemento de un lenguaje, un patrón es una instrucción que muestra como puede ser usada esta configuración espacial una y otra vez para resolver el sistema de fuerzas, siempre que el contexto lo haga relevante.” “Cada patrón describe un problema que ocurre una y otra vez en el entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo siquiera dos veces de la misma forma” (Alexander, Christopher).

Existen diferentes tipos de patrones, entre ellos se pueden encontrar:

- Patrones arquitectónicos.
- Patrones de diseño.

Un patrón arquitectónico no es más que un patrón de software que ofrece soluciones a problemas de arquitecturas comunes surgidos a la hora de construir un nuevo sistema. Su objetivo fundamental es facilitar la tarea del diseño de tal sistema. La utilización de patrones mejora la concepción y entendimiento de los sistemas, por estas razones se decide aplicarlos en el desarrollo del presente trabajo.

Patrón de Arquitectura Cliente-Servidor

El patrón de arquitectura cliente-servidor es un enfoque de diseño ampliamente utilizado en el desarrollo de sistemas distribuidos. En este patrón, la aplicación se divide en dos componentes principales:

- **Cliente:** Es la parte de la aplicación que interactúa directamente con el usuario. El cliente se encarga de enviar solicitudes al servidor y presentar los resultados al usuario de manera adecuada. Puede ser una interfaz de usuario, una aplicación móvil, un navegador web, entre otros.
- **Servidor:** Es la parte centralizada de la aplicación que procesa las solicitudes del cliente y envía las respuestas correspondientes. El servidor gestiona la lógica de negocio, el acceso a los datos y otros aspectos relacionados con el funcionamiento de la aplicación. Puede ser un servidor físico, una máquina virtual o un servicio en la nube.

El patrón cliente-servidor permite una arquitectura escalable y flexible, ya que varios clientes pue- den conectarse simultáneamente a un servidor centralizado. Además, permite la separación de preocupaciones y la reutilización de la lógica de negocio en el servidor, lo que facilita el mantenimiento y la evolución del sistema (Thies y Fuhrmannek, 2019).

Arquitectura basada en capas

Es definido como una organización jerárquica, donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Las restricciones topológicas del patrón pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos de la misma; se supone que si esta exigencia se relaja, el patrón deja de ser puro y pierde algo de su capacidad heurística.

Pueden implementarse sistemas con dos, tres, n capas, sin embargo, el ejemplo más ilustrativo y usado lo constituye el modelo de tres capas. (Ilustración 3)

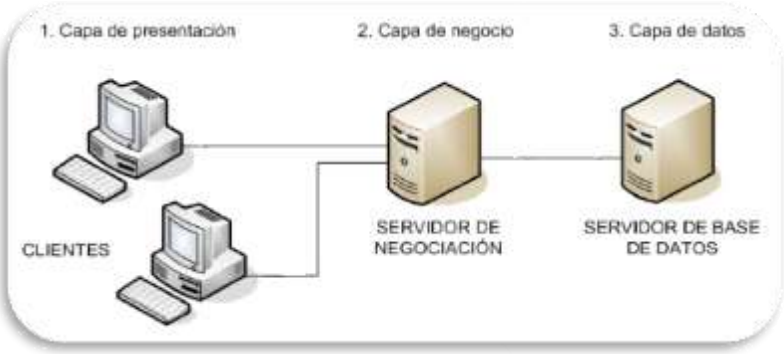

Ilustración 3: Modelo Tres Capas

Las tres capas representadas son:

- **Capa de presentación:** reúne los aspectos del software relacionados con las interfaces y la interacción con los diferentes tipos de usuarios.
- **Capa de negocio:** reúne los aspectos del software que se tienen que automatizar. Incluye las tareas, reglas y las restricciones que se aplican.
- **Capa de datos:** constituye el puente entre las dos capas anteriores. Encapsula la lógica de acceso a datos.

La arquitectura basada en capas facilita la escalabilidad y el mantenimiento del sistema, ya que permite la sustitución o actualización de una capa sin afectar a las demás. También permite la re- utilización de componentes, ya que las capas están bien definidas y se pueden utilizar en diferentes contextos (Buschmann y col., 2020).

Patrón de Arquitectura de la solución

En el patrón MVC, el modelo representa los datos y la lógica de negocio de la aplicación. La vista se encarga de la presentación visual de los datos y la interacción con el usuario. El controlador actúa como intermediario entre el modelo y la vista, gestionando las solicitudes del usuario, actualizando el modelo y coordinando la actualización de la vista (Gammack y Kopec, 2017).

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. La finalidad del modelo es mejorar la reusabilidad por medio del desacople entre la vista y el

modelo. La característica fundamental de este modelo es que al separar como entidades independientes sus elementos, un cambio en el modelo se refleja automáticamente en las vistas.

En la ilustración 4 se refleja el funcionamiento de un sistema implementando el patrón arquitectónico Modelo Vista Controlador.

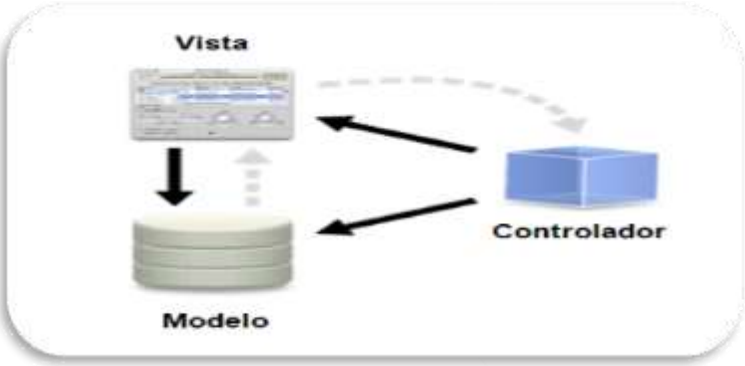

Ilustración 4: Modelo Vista Controlador

Los tres elementos representados en esta figura son:

- **Modelo:** maneja los datos y controla todas sus transformaciones. El sistema es el encargado de notificar a las Vistas los cambios ocurridos en el modelo.
- **Vista:** genera una representación visual del Modelo y muestra los datos al usuario.
- **Controlador:** proporciona significado a las órdenes del usuario, actuando sobre los datos representados en el modelo. Entra en acción cuando se produce algún cambio en alguno de los elementos anteriores.

Patrones de Diseño

"Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto específico" (Larman, 2008).

Fábrica abstracta (Abstract Factory)

Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas. La idea principal del patrón es proporcionar un mecanismo para crear objetos relacionados sin acoplar el código cliente a clases concretas. Esto permite que el código cliente sea independiente de la implementación concreta y se pueda intercambiar fácilmente entre diferentes familias de objetos.

Tiene como principales características:

- Aísla las clases concretas: Ayuda a controlar las clases de objetos que crea una aplicación.
- Facilita el intercambio de familias de productos: La clase de una fábrica concreta sólo aparece una vez en una aplicación (cuando se crea).

- Promueve la consistencia entre productos: Cuando se diseñan objetos productos en una familia para trabajar juntos, es importante que una aplicación use objetos de una sola familia a la vez. Fábrica abstracta facilita que se cumpla esta restricción (Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J., 2003).

Método de la fábrica (Factory Method)

Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar. Permite que una clase delegue en subclases la creación de objetos.

- Encapsula la creación de objetos y promueve la flexibilidad al permitir que las subclases tomen decisiones sobre la creación de objetos.
- El cliente simplemente invoca el Factory Method de la clase base y recibe una instancia del objeto deseado sin tener conocimiento de la clase concreta que se está creando.
- Permite extender o personalizar el proceso de creación sin modificar el código cliente (Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J., 2003).

Repositorio (Repository)

El patrón de diseño Repository es un patrón de diseño arquitectónico que se utiliza en el desarrollo de software para gestionar el acceso y la persistencia de los datos. Se basa en la idea de tener una interfaz o clase abstracta llamada "Repository" que define métodos para acceder, crear, actualizar y eliminar objetos de un origen de datos, como una base de datos. Luego, se implementan clases concretas que utilizan tecnologías específicas para interactuar con el almacenamiento de datos, ya sea mediante consultas SQL, llamadas a API u otras formas de acceso.

- Proporciona una capa de abstracción entre la lógica de negocio de una aplicación y el almacenamiento de datos, permitiendo un acoplamiento más débil y una mayor flexibilidad.
- Proporciona una abstracción de alto nivel para el acceso a los datos, lo que permite a la lógica de negocio trabajar con entidades y objetos en lugar de tener que lidiar directamente con los detalles de almacenamiento y persistencia.
- Permite centralizar la lógica de acceso a los datos, lo que facilita su mantenimiento y evita la duplicación de código (Eisenberg, 2020).

Conclusiones del segundo capítulo

Durante el desarrollo de este capítulo se elaboró el plan principal de iteraciones, siempre teniendo en cuenta que las mismas pueden sufrir modificaciones durante el proceso de implementación. También se plasmaron las historias de usuario para el desarrollo del software.

A partir de la descripción de la solución propuesta en este capítulo se llega a un mejor entendimiento de los requerimientos funcionales para que el software cumpla de manera satisfactoria con las necesidades del cliente, siendo la comunicación con el mismo el factor más importante para lograrlo.

Siguiendo esta planificación inicial se procedió a construir el sistema que constituye la propuesta de solución. Se concretaron los patrones de diseño y arquitectura aplicados a la propuesta con el objetivo de satisfacer la necesidad por parte del cliente de un software extensible y flexible a los cambios.

Capítulo III Análisis de los resultados

Mediante la implementación de tecnologías, herramientas y una metodología de desarrollo específica, se creó una herramienta web que cautivó a los clientes por su atractivo, comodidad y seguridad. Esta aplicación no solo fomentó la fidelidad de los usuarios, sino que los convenció de utilizar este software debido a los múltiples beneficios que les ofrece. Además, contribuye significativamente en la solicitud de créditos, permitiendo a la entidad objeto de estudio realizar esta tarea de manera más eficiente y ventajosa.

Entre las ventajas y beneficios que esta solución aportó a la entidad se incluyen:

- Provee a la entidad de una herramienta capaz de facilitar la solicitud de los créditos convirtiendo un proceso que hoy es engorroso y complejo en algo sencillo, seguro y al alcance de todos los clientes.
- Permite la personalización de productos bancarios en dependencia de las necesidades de mayor coincidencia entre los tipos de clientes.

Vistas del software

Ilustración 5: Pantalla de Autenticación

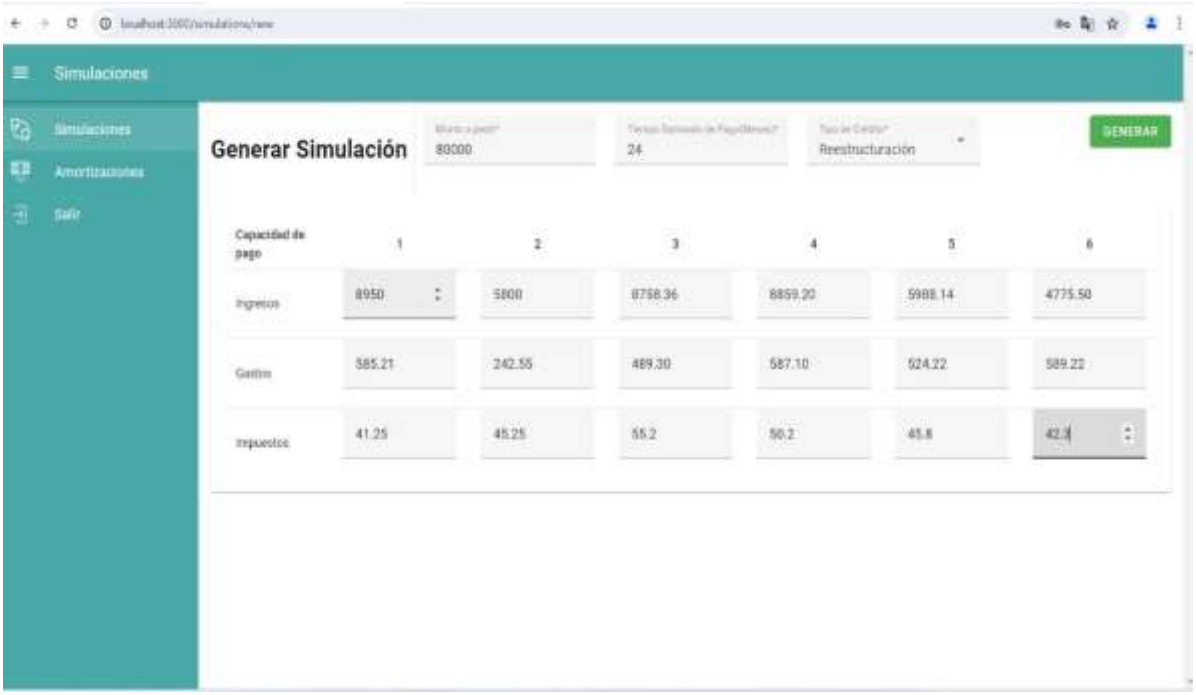

Ilustración 6: Gestionar Simulación

Ilustración 7: Detalles de la Amortización

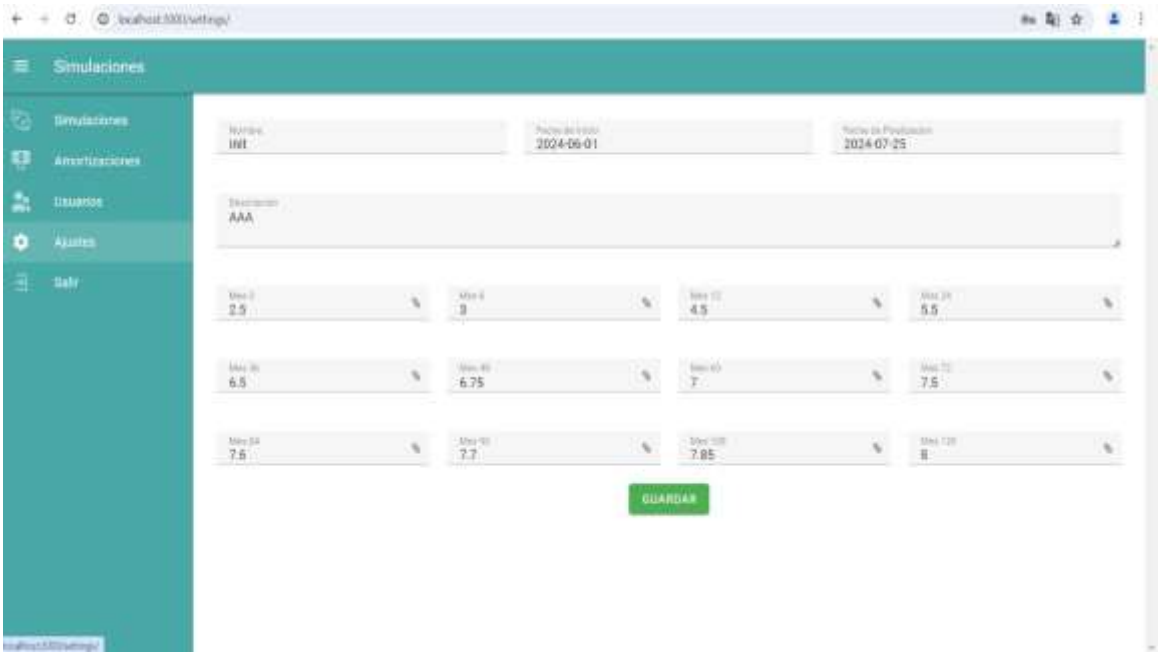

Ilustración 8: Panel de Administración

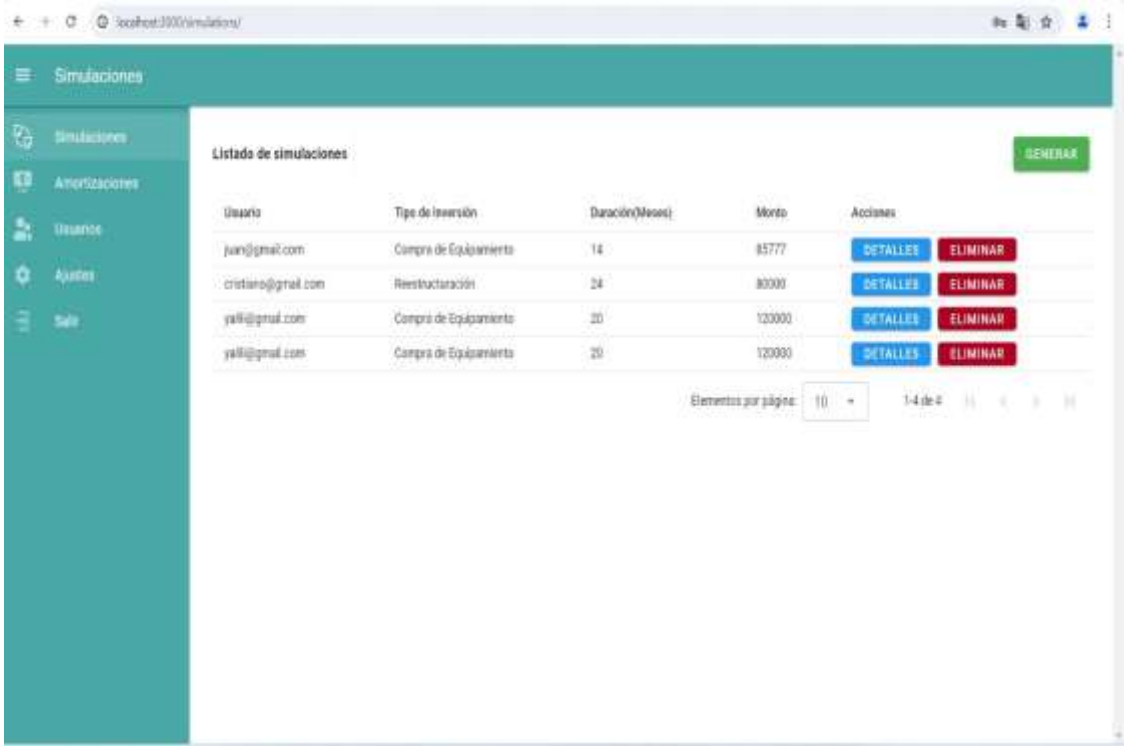

Ilustración 9: Listar Simulación

Pruebas de Aceptación

Las pruebas de aceptación son un recurso fundamental durante el desarrollo de un sistema basado en la metodología ágil. Con base en los principios propuestos en ésta, es posible aplicar los “tests” durante las diferentes iteraciones. Permiten evaluar el grado de satisfacción del cliente con el producto que se está desarrollando. Se elaboran en paralelo con el desarrollo del sistema, y adaptándose a sus cambios. De esta manera, las pruebas de aceptación se ejecutan ágilmente para corregir los errores oportunamente.

Campos de la tabla de prueba:

- **Código:** identificador del caso de prueba. Dividido en dos partes. La primera representa la inicial del artefacto y la segunda representa el número con que se identifica la prueba.
- **Descripción:** es una breve descripción del propósito de la prueba.
- **Condiciones de ejecución:** condiciones especiales que deben tenerse en cuenta para ejecutar el caso de prueba,
- **Entradas / pasos de ejecución:** entradas o funciones que deben ejecutarse para realizar el caso de prueba,
- **Resultado esperado:** salida u objetivo que debe cumplir la funcionalidad a la que se le realiza el caso de prueba,
- **Evaluación:** evaluación de éxito del caso de prueba. Prueba satisfactoria en caso de éxito o prueba insatisfactoria en caso de fallo.

A continuación, se muestra una de las pruebas realizadas:

Tabla 3.1: Caso de Prueba Autenticar al sistema con errores I

Caso de prueba de aceptación	
Código: PA01_HU2	**Historia de Usuario:** HU_2
Nombre: Autenticar al sistema con errores I	
Descripción: Se intentará autenticar al sistema dejando los campos de correo y clave vacíos.	
Condiciones de ejecución: El sistema no debe haber iniciado sesión o tenerla cerrada en el servidor.	

Pasos de ejecución: Se ingresa a la pantalla de autenticación mediante la url en el navegador. Una vez que aparezca la vista inicial, se intenta presionar el botón Aceptar del cuadro de diálogo con los campos correo y clave vacíos.
Resultados esperados: El sistema inhabilita el botón Aceptar. Muestra mensaje de error de correo y/o contraseña requerida.
Evaluación de la prueba: Satisfactoria

Tabla 3.2: Caso de Prueba Autenticar al sistema con errores II

Caso de prueba de aceptación	
Código: PA02_HU2	**Historia de Usuario:** HU_2
Nombre: Autenticar al sistema con errores II	
Descripción: Se intentará autenticar al sistema con el correo y/o clave incorrecta.	
Condiciones de ejecución: El sistema no debe haber iniciado sesión o tenerla cerrada en el servidor.	
Pasos de ejecución: Se ingresa a la pantalla de autenticación mediante la url en el navegador. Una vez que aparezca la vista inicial, se introduce correo y/o clave incorrecta. Se presiona el botón Aceptar	
Evaluación de la prueba: Satisfactorio	

Tabla 3.3: Caso de Prueba Generar Capacidad de Pago I

Caso de prueba de aceptación	
Código: PA03_HU5	**Historia de Usuario:** HU_5
Nombre: Generar Capacidad de Pago I	

Descripción: Se intentará introducir ingresos, gastos e impuestos al sistema de forma incorrecta (letras).
Condiciones de ejecución: El sistema debe haber generado una simulación.
Pasos de ejecución: Se ingresa a la pantalla de Generar simulación. Una vez que aparezca la tabla de capacidad de pago, se introduce ingresos, gastos y/o impuestos de forma incorrecta.
Resultados esperados: Muestra mensaje de solicitud de campo requerido.
Evaluación de la prueba: Satisfactorio

Tabla 3.4: Caso de Prueba Generar Capacidad de Pago II

Caso de prueba de aceptación	
Código: PA04_HU5	**Historia de Usuario:** HU_5
Nombre: Generar capacidad de pago II	
Descripción: Se intentará dejar los campos de cantidad de meses y/o monto a solicitar vacíos	
Condiciones de ejecución: El sistema debe haber generado una simulación.	
Pasos de ejecución: Se ingresa a la pantalla de Generar simulación. Una vez que aparezca la tabla de capacidad de pago, se intenta dejar los campos de cantidad de meses y/o monto a solicitar en blanco.	
Resultados esperados: Muestra mensaje de solicitud de campo requerido.	
Evaluación de la prueba: Satisfactorio	

Tabla 3.5: Caso de Prueba Generar flujo de caja I

Caso de prueba de aceptación	
Código: PA05_HU6	**Historia de Usuario:** HU_6
Nombre: Generar flujo de caja	
Descripción: Se intentará introducir ingresos, gastos, deudas e impuestos al sistema de forma incorrecta (letras).	
Condiciones de ejecución: El sistema debe haber generado una simulación.	
Pasos de ejecución: Se ingresa a la pantalla de Generar flujo de caja. Una vez que aparezca la tabla de flujo de caja, se introduce ingresos, gastos, deudas y/o impuestos de forma incorrecta.	
Resultados esperados: Muestra mensaje de solicitud de campo requerido.	
Evaluación de la prueba: Satisfactorio	

Conclusiones del tercer capítulo

La implementación exitosa de la herramienta web con tecnología avanzada y metodología específica facilita a los clientes la solicitud de créditos en el BPA, ayudando en la planificación de sus ingresos y gastos. Sus ventajas notables incluyen una presentación atractiva, además de dotar a la entidad de una herramienta para mejorar la gestión de productos bancarios. Las vistas del sistema respaldan su diseño intuitivo, reforzando visualmente estos beneficios. En conjunto, esta herramienta se ha convertido en una inversión estratégica que impulsa la eficiencia operativa y la satisfacción del cliente de manera significativa.

Conclusiones

Con la realización de la presente investigación se establecieron los fundamentos teórico-metodológicos de los principales elementos que componen la Simulación de Solicitud de Créditos de la Dirección Provincial del BPA Matanzas.

Se definió la metodología de desarrollo ágil XP para el desarrollo del sistema. El estudio de las bases teóricas permitió, además, la elección de las herramientas adecuadas para el desarrollo de este tipo de sistemas, se especificaron los requisitos funcionales y no funcionales, lo que permitió identificar las funcionalidades que dan respuesta a las necesidades del usuario.

Se implementó el sistema web Plataforma para la Simulación de Solicitud de Créditos Bancarios del BPA Matanzas según los presupuestos y requisitos previamente definidos, haciendo uso de herramientas de software libre y código abierto.

Se validó su funcionamiento mediante la realización de las pruebas necesarias, según la metodología XP y posteriormente se realizó el análisis de los resultados que arrojaron las mismas.

Recomendaciones

Dentro de las posibles mejoras y recomendaciones que se le hacen al sistema desarrollado están:

1. Continuar con la investigación con el objetivo de perfeccionar o adicionar nuevas funcionalidades al sistema.

Bibliografía

Alexander, Christopher.(2024). *The Timeless Way of Building*. Oxford University Press: s.n. 0195024028.

ArgoUML (2023). *ArgoUML: Open source UML modeling tool.* Consultado: 2024-05-09. URL: http://argouml.org/.

Beck, K. (2004). *Extreme Programming Explained: Embrace Change*. 2nd. Addison-Wesley Professional.

Borrás, F. (2013). La Banca Comercial. *Productos y Servicios.* La Habana: Félix Varela., 2013.

Buschmann, F. y col. (2020). *Pattern-Oriented Software Architecture: A System of Patterns*. Hobo- ken, NJ: John Wiley y Sons. ISBN: 978-0471958697.

Budgen, D. (2013). *Software design*. Pearson Education.

Cohn, M. (2004a). "An Introduction to CRC Cards". En: *Better Software* 6.1, págs. 19-22. URL: https://www.mountaingoatsoftware.com/papers/crc-cards.pdf (visitado 21-05-2023).

Docs, M. W. (2023a). *CSS (Cascading Style Sheets)*. Consultado: 2023-05-17. URL: https : //developer.mozilla.org/en-US/docs/Web/CSS.

Eisenberg, J. (2020). "Repository Design Pattern". En: Microsoft Docs. URL: https://docs.microsoft.com/enus/dotnet/architecture/microservices/microserviceddd-cqrs-patterns/infrastructure-persistence-layer-design (visitado 20-05-2023).

Foundation, P. S. (2023a). *What is Python? Executive Summary*. Consultado: 2024-05-08. URL: https://www.python.org/doc/essays/blurb/.

Gammack, J. y D. Kopec (2017). *Learning MVC Architecture with ASP.NET Core 2*.Birmingham, UK: Packt Publishing. ISBN: 978-1786465354.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (2003) *Patrones de Diseño. Elementos del software orientado a objetos reutilizable.* ISBN:84-7829-059-1

Gómez García, D. E. (2018). *"Desarrollo del sistema de requisiciones para la empresa hidroeléctrica Abanico SA Aplicando el entorno de programación Node. js"*. B.S. thesis. Escuela Superior Politécnica de Chimborazo.

González, M. H. Á. (2009) *Diseño de la Base de Datos del Sistema de Información de Perforación de Pozos.* Habana, Cuba: s.n., 2009.

Grinberg, M. (2018). *Flask Web Development with Python Tutorial*. O'Reilly Media.ISBN: 978- 1449372620.

IBM (2023). *IBM Rational Rose Enterprise*. Consultado: 2024-05-08. URL:https://www.ibm.com/products/rational-rose-enterprise.

Larman, Craig. (2008). *Introducción al Análisis y Diseño Orientado a Objetos*. Uruguay: s.n., 2008.

Melgarejo, M. (2019). *Perfeccionamiento del Manual de Instrucciones y Procedimientos para la concesión de créditos a Trabajadores por Cuenta Propia en el Banco Popular de Ahorro.*

Mikowski, T. y J. Powell (2013). *Single Page Web Applications: JavaScript end-to-end.* Manning Publications. ISBN: 978-1617290756.
Otal, S.; Serrano, G.; Serrano, R. (2007) *Simulación Financiera* ISBN: 84-7978-782-1
Pallets (2023). Flask. Consultado: 2023-05-16. URL: https://flask.palletsprojects.com/.
Paradigm, V. (2023a). Visual Paradigm. Consultado: 2023-05-15. URL:https://www.visual- paradigm.com/.
Paradigm, V. (2023b). *Visual Paradigm Product Overview*. Consultado: 2023-03-24.URL:https://www.visualparadigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.
Team, T. A. (2023a). *Angular*. Consultado: 2023-05-16. URL: https://angular.io/.
Team, T. P. (2023b). *Welcome to Pyramid, a Python Web Framework*. Consultado:2023-05-16. URL: https://trypyramid.com/.
Team, T. R. (2023c). *React*. Consultado: 2023-05-16. URL: https://reactjs.org/.
Team, T. V. (2023d). *Vue.js*. Consultado: 2023-05-16. URL: https://vuejs.org/.
Thies, F. y R. Fuhrmannek (2019). *Software Architecture for Developers*. Heidelberg, Germany: dpunkt.verlag. ISBN: 978-3864906468.

Anexos

Anexo A: Historias de Usuarios

Descripción de las Historias de Usuarios definidas para el sistema.

HISTORIA DE USO			
Orden	HU_1	**Nombre**	Diseño y creación de la base de datos
Riesgo	Alto	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	1
Descripción	Se realizará el diseño de la base de datos y será definida por medio de migraciones con el ORM del marco de trabajo.		
Observación	El modelo de base de datos será escrito en el idioma inglés. El nombre de las tablas será en plural y el nombre de los atributos en singular.		

HISTORIA DE USO			
Orden	HU_2	**Nombre**	Autenticar usuario en el sistema
Riesgo	Bajo	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	0.5
Descripción	El sistema debe brindar la funcionalidad de acceso por medio de correo electrónico y clave de acceso. Las claves de acceso podrán ser modificadas solamente por el administrador del sistema. El sistema deberá permitir al usuario cerrar su sesión una vez el mismo lo decida. Si		

	el usuario olvida su clave de acceso se le generará una de forma automática y se le enviará a su correo electrónico.
Observación	El sistema no debe permitir autenticar o cerrar sesión al usuario más una vez. El sistema debe permitir eliminar y restaurar todos los usuarios.

HISTORIA DE USO			
Orden	HU_3	**Nombre**	Gestionar usuarios en panel de administración
Riesgo	Bajo	**Prioridad**	Alta
Iteración	1	**Puntos estimados**	0.5
Descripción	El sistema debe brindar la funcionalidad de listar los usuarios, además, adicionar, actualizar y eliminar un usuario en el sistema, todo esto a través del panel de administración del sistema. Se realizará un sistema para la búsqueda de usuarios por nombre de usuario y correo electrónico.		
Observación	El listado debe contener una paginación de registros.		

HISTORIA DE USO			
Orden	HU_4	**Nombre**	**Gestionar Simulación**
Riesgo	Alto	**Prioridad**	Alta
Iteración	2	**Puntos estimados**	1

Descripción	El sistema debe brindar la funcionalidad de listar las simulaciones realizadas previamente por el usuario, además de crear una nueva simulación (Generar).
Observación	El sistema debe ser capaz de mostrar la tabla de solicitud de datos para la simulación según el tipo de cliente.

HISTORIA DE USO			
Orden	HU_5	**Nombre**	**Generar Capacidad de Pago**
Riesgo	Alto	**Prioridad**	Alta
Iteración	3	**Puntos estimados**	1
Descripción	El sistema debe brindar la funcionalidad de calcular la capacidad de pago teniendo en cuenta el tipo de cliente. Para este cálculo los usuarios deben introducir los ingresos, gastos e impuestos de los últimos 6 meses.		
Observación	El sistema debe ser capaz de mostrar la capacidad de pago.		

HISTORIA DE USO			
Orden	HU_6	**Nombre**	**Generar Flujo de Caja**
Riesgo	Alto	**Prioridad**	Alta

Iteración	3	**Puntos estimados**	1
Descripción	El sistema debe brindar la funcionalidad de calcular el flujo de caja teniendo en cuenta el tipo de cliente. Para este cálculo los usuarios deben introducir los ingresos, gastos, deudas e impuestos de los últimos 6 meses.		
Observación	El sistema debe ser capaz de mostrar la solvencia económica.		

HISTORIA DE USO			
Orden	HU_7	**Nombre**	Generar Amortización
Riesgo	Alto	**Prioridad**	Alta
Iteración	4	**Puntos estimados**	3
Descripción	El sistema debe brindar la funcionalidad de calcular el cuadro amortizativo de la solicitud de crédito y permite listar las amortizaciones previamente realizadas por el usuario.		
Observación	El sistema debe mostrar el reporte de cuadro amortizativo		

Anexo B: Diagrama de Despliegue

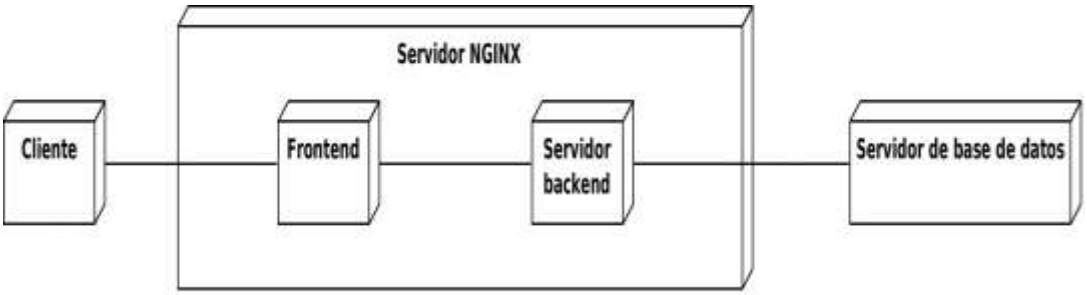

Ilustración 8: Diagrama de despliegue

- **Cliente:** el nodo representa un dispositivo cliente desde el cual se podrá acceder al sistema por medio de un navegador web e interactuar con todas las funcionalidades que este brinda.
- **Servidor NGINX:** el nodo representa el servidor web donde estarán alojados:
- **Frontend:** servidor donde se encuentra la interfaz que interactúa con los usuarios.
- **Servidor *Backend*:** servidor que se encarga de procesar toda la lógica que se encuentra en la página web.
- **Servidor de base de datos:** el nodo representa el servidor de base de datos MySQL en el que estará alojada la base de datos del sistema.

Anexo C: Diagrama de Paquetes

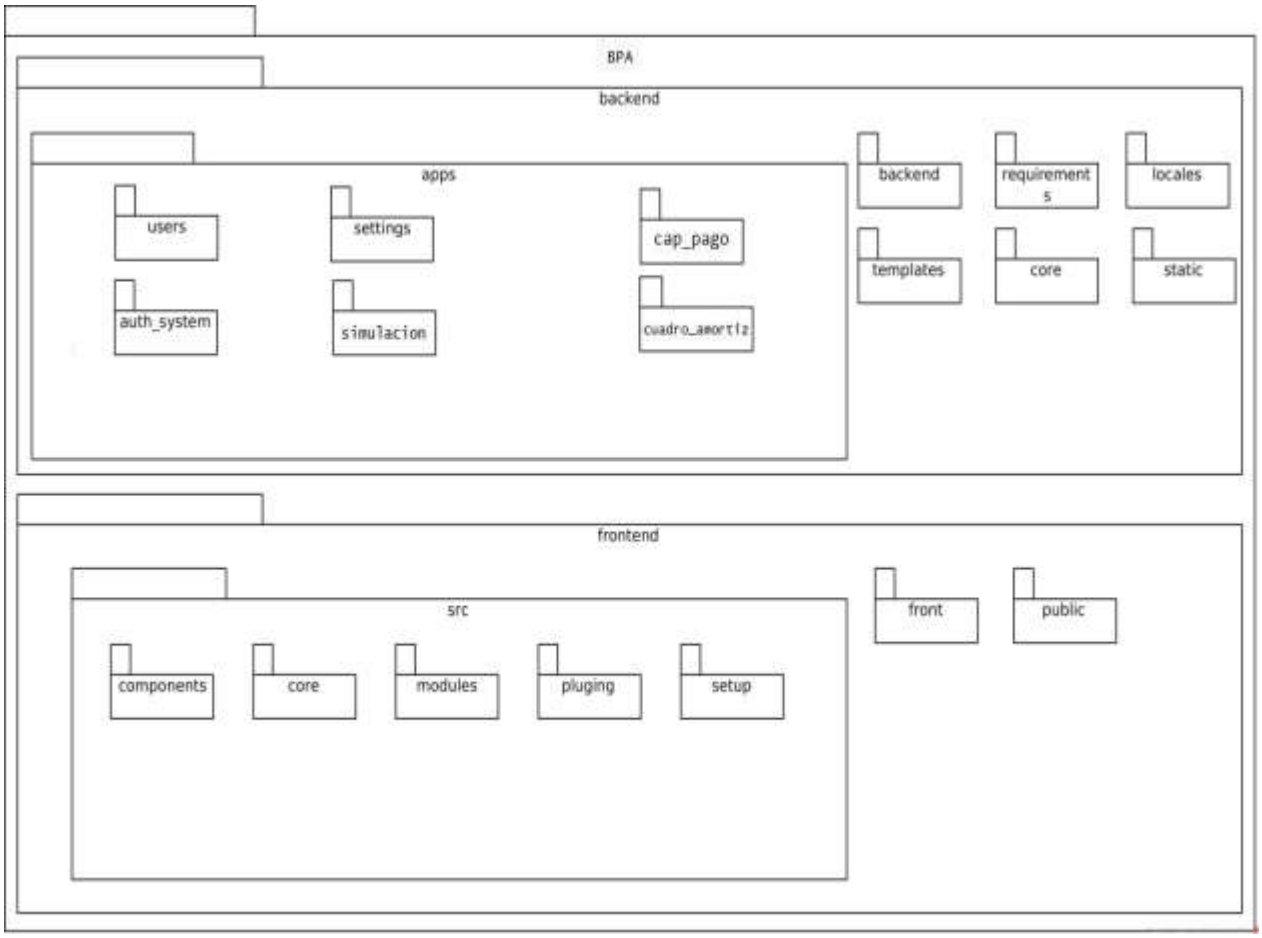

Ilustración 9: Diagrama de Paquetes

Backend:

- **apps:** cada uno de los paquetes contenidos dentro de este paquete contiene los modelos, la lógica y las validaciones correspondientes a la app con el mismo nombre.
- **backend:** es la carpeta raíz del proyecto. Se encuentran los ficheros que manejan toda la configuración del proyecto.
- **templates:** se encuentran las plantillas visuales de los correos electrónicos.
- **requirements:** se encuentran todas las dependencias utilizadas para el desarrollo del sistema.
- **locales:** contiene archivos de traducción para el sitio o aplicación web.
- **static:** contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, cadenas de texto, css y algunos ficheros que contienen la programación lógica de las vistas.
- **core:** se almacenan las bases para la arquitectura principal del proyecto.

Frontend:

- **public**: se almacenan los archivos estáticos del sitio.
- **front:** donde se almacenará el sitio compilado y listo para producción,
- **src:** es la carpeta raíz del proyecto.
- **components:** se almacenan los componentes genéricos y compartidos del proyecto, core: se almacenan las bases para la arquitectura principal del proyecto.
- **modules:** contiene cada módulo del sistema, contiene sus componentes, estilos y sus archivos de typescript.
- **pluing:** carpeta donde se instalan los pluings del proyecto.
- **setup:** configuraciones globales del proyecto y de los pluings.