Diseño y desarrollo de un prototipo funcional para una aplicación web enfocada en la gestión de gastos personales

Johan Fabian Fornari Rios y Jonathan Vanegas Lesmes

Facultad de Ingeniería

Universidad Distrital Francisco José de Caldas

Especialización en Ingeniería de Software

PhD. Beitmantt Geovanni Cardenas Quintero

Octubre 24, 2023

Introducción

Actualmente, muchas personas tienen problemas al administrar adecuadamente su dinero, por un escaso control de sus finanzas y falta de educación financiera, lo que ocasiona que sus egresos superen los ingresos, por lo que surge la necesidad de endeudarse para poder cubrir sus gastos, lo que a largo plazo podría generar problemas financieros. El presente proyecto pretende brindar una solución práctica mediante la implementación de una herramienta tecnológica al problema de la administración del dinero desde la perspectiva de las finanzas personales.

Para que la aplicación sea de acceso masivo, se propone implementar una arquitectura web y adicionalmente para la gestión del proyecto, se trabajará con metodología iterativa e incremental. Esta metodología permite el desarrollo y la entrega de funcionalidades en ciclos cortos y manejables, facilitando la incorporación de mejoras y adaptaciones en el proyecto.

Si bien existen varias alternativas de aplicaciones de gestión de finanzas personales en el mercado, la propuesta diferencial de este proyecto consiste en permitir al usuario generar presupuestos sobre módulos personalizados según la necesidad y/o patrones de ingresos y gastos de cada usuario. Se espera que el prototipo implementado se ajuste a las necesidades de la mayoría de los usuarios que tengan ingresos económicos, enmarcado dentro de las leyes y buenas prácticas en materia de finanzas personales.

Cada uno de los aspectos involucrados en el desarrollo del presente proyecto se describen en este documento, iniciando con la contextualización de la investigación, seguido del desarrollo de la investigación y finalizando con el cierre, resultados obtenidos y conclusión.

Parte I. Contextualización de la Investigación

Capítulo 1. Descripción de la Investigación

En este capítulo se expondrá el problema planteado, el cual se busca solucionar con el desarrollo del presente proyecto. Para esto se plantean los objetivos generales y específicos, se realiza la justificación de la investigación planteada generando así una hipótesis. Se realiza el marco referencial, la metodología de la investigación y la organización del trabajo.

1.1. Planteamiento / Identificación del Problema

1.1.1. Planteamiento del problema

Un alto porcentaje de la población tienen problemas a la hora de administrar sus ingresos, específicamente cuando hablamos de los gastos que tiene el individuo, lo que implica un déficit importante en sus finanzas. Analizando las causas de este comportamiento se observa que esto se genera por una escasa educación financiera y por no llevar un control y registro de sus finanzas personales en una herramienta informática especializada que permita analizarlas personalizadas y ajustadas a sus necesidades.

1.1.2. Formulación del problema

¿Cómo se puede diseñar e implementar un prototipo funcional escalable para una aplicación web destinada a la gestión de gastos personales?

1.1.3. Sistematización del problema

¿Existen herramientas tecnológicas que ayuden a gestionar los gastos personales?

¿Qué modelos de generación de código automático se pueden emplear en una aplicación de gestión de gastos personales?

¿Qué tipo de modelos de bases de datos se pueden utilizar para la creación de una aplicación de gestión de finanzas personales?

¿Qué arquitecturas de construcción de software existen para la implementación de aplicaciones web?

¿Cómo diseñar y construir reportes en una aplicación web para permitir la visualización de los ingresos y gastos?

1.2. Objetivos

1.2.1. Objetivo general

Diseñar e implementar un prototipo funcional para una aplicación web destinada a la gestión de gastos personales.

1.2.2. Objetivos específicos

- Diseñar un modelo de datos escalable que permita la adición de atributos en las tablas de la base de datos a medida que se creen gastos personalizados por parte de los usuarios.
- Desarrollar un prototipo funcional de la aplicación haciendo uso de un modelo de generación automática de código aplicado a los CRUDS del sistema.
- Desarrollar reportes de presupuestos y gastos.

1.3. Justificación del trabajo/investigación

A continuación, se describen los aspectos principales que ayudan a sustentar la justificación del trabajo de investigación planteado.

1.3.1. Justificación teórica

En este proyecto se quiere desarrollar una aplicación web enfocada en la gestión de gastos personales que permita un manejo flexible para su uso, según las necesidades particulares de cada usuario. Esto se logra mediante la implementación de un modelo de gestión de gastos personales que permita la creación de categorías de gastos personalizadas.

1.3.2. Justificación metodológica

Nos inclinamos por la metodología iterativa incremental para la gestión durante el desarrollo de nuestro proyecto porque permite la generación de entregables funcionales de la aplicación rápida y ágil. Para la aplicación se propone la implementación del modelo evolutivo que especifica un desarrollo incremental e iterativo, que permite generar un producto de software inicial con parte de la funcionalidad requerida, y a medida que se realizan iteraciones se añaden nuevas funciones que mejoran las existentes y se adaptan según la retroalimentación recibida. El producto final podrá ser usado como referencia para futuros desarrollos de arquitecturas de software.

1.3.3. Justificación práctica

Con el desarrollo de este proyecto se busca brindar una herramienta de software alternativa basada en una aplicación web de tipo cliente servidor que implemente un modelo de

generación de código automático para los cruds del sistema y un modelo de gestión de gastos personales, que se pueda adaptar a las necesidades de cada usuario.

1.4. Hipótesis

El desarrollo de un prototipo de una aplicación web enfocada en un modelo de gestión de gastos personales permitirá que los usuarios puedan registrar y administrar sus gastos y así tener un control más inteligente del consumo del dinero. También se les brindará a los usuarios la funcionalidad para que puedan descargar informes desde el prototipo desarrollado, los cuales les permitan revisar sus patrones de gastos según los ingresos que estos tengan.

1.5. Marco Referencial

A continuación, se incluyen todos los elementos del marco referencial que incluyen un marco teórico, marco conceptual y el marco legal.

1.5.1. Marco Teórico

Hoy en día las ciencias de la administración y las finanzas estudian diferentes aspectos relacionados con la gestión del dinero. Estos aspectos suelen abordarse desde un enfoque empresarial y no a nivel personal. Es así como se puede ver al individuo como una pequeña empresa que tiene la necesidad de administrar sus finanzas de la mejor forma posible y para lograr este fin se requieren herramientas que le ayuden a la persona a tener un control en sus finanzas personales de una forma práctica y sencilla. Actualmente hay sin fin de herramientas tecnológicas que dan soluciones relacionadas con la administración de recursos a nivel empresarial, pero en el caso de las personas estas optan por organizar sus finanzas con algunos elementos dados por experiencia, vivencias, recomendaciones y otras mediante herramientas ofimáticas como Excel y, en menor medida, con apps para ello.

El éxito en la administración de las finanzas personales puede venir de diversos elementos, pero hay un elemento clave que nos permite evaluar los caminos para tomar decisiones de la forma más acertada posible, este elemento es el presupuesto.

1.5.1.1. Paradigmas Finanzas Personales. A través de la planeación financiera de un individuo, hay aspectos a considerar al realizar un presupuesto, mediante decisiones financieras acertadas que permitan alcanzar las metas deseadas. A continuación, se presentan cuatro paradigmas de las finanzas personales extraídos de la Revista Escuela de Administración de Negocios, donde se abordan creencias erróneas en el manejo del dinero. (Sarmiento, 2005).

- **El paradigma del progreso**

Se sostiene que la acumulación de bienes se asocia con un mayor progreso económico, pero es fundamental mantener un equilibrio entre los bienes, sus beneficios y los pasivos, representados por las deudas. Según datos de la DANE, aproximadamente el 50% de la población no se ajusta a este modelo, ya que sus deudas superan sus activos o no cuentan con reservas financieras, lo que sugiere que poseen más pasivos que activos.

- **El paradigma financiero**

Consiste en un principio financiero que plantea que entre mayores activos se tengan, mayor es la cantidad de ingresos que se deben generar. Este paradigma tiene una gran aplicación en el sector empresarial, sin embargo, en las finanzas personales se encuentran escenarios donde los activos solo generan gastos y no beneficios, por ejemplo, un vehículo y en ocasiones los bienes raíces.

- **El paradigma del ingreso bajo**

Mediante la encuesta de ingresos y egresos del DANE los investigadores han encontrado que los trabajadores promedio sienten que necesitan como mínimo un incremento adicional del 35% en el salario mensual para alcanzar sus metas y sueños.

- **El paradigma de la buena suerte**

Este paradigma se centra en una variable que no tiene mucha credibilidad en el mundo financiero, esta variable es la suerte. A menudo las personas pueden incluirla como un factor que les ayude a cumplir sus metas y sueños o que puedan solucionar sus problemas financieros. Un ejemplo de esto es jugar la lotería, estadísticamente la probabilidad de ganar la lotería es entre 1 a más de 10.000.000.

De acuerdo con la tabla 1 se puede determinar que, entre los individuos con niveles altos de ingresos, se mantiene un hábito de vivir endeudados o al día, mientras que la diferencia en el ahorro de los que poseen niveles de ingreso bajos y altos no es muy representativa. Lo anterior demuestra que el problema no se encuentra en el ingreso sino en la forma de administrarlo.

Tabla 1

Hábitos financieros de Colombia.

Nivel de Ingresos (salarios mínimos)	Ahorran habitualmente	Viven "al día"	Viven "al fiado"
Menos de 1	N.A.	54,75%	44,80%
entre 1 y 2	2.7%	45,37%	51,90%
entre 2 y 3	7,52%	40,06%	52,40%
entre 3 y 4	17,88%	4,47%	77,70%
entre 4 y 5	22,71%	33,22%	44,10%
entre 5 y 6	23,96%	21,10%	54,90%
entre 6 y 7	28,32%	25,86%	45,80%
entre 7 y 8	26,68%	15,81%	57,50%
entre 8 y 9	30,82%	1,57%	67,60%
entre 9 y 10	26,81%	14,87%	58,30%
entre 10 y 15	17,29%	46,04%	36,70%
Más de 15	8,90%	N.A.	N.A.
PROMEDIO	14,73%	18,01%	31,13%

Fuente: encuesta de Ingresos y Egresos DANE

Nota. Datos tomados de revista de finanzas personales. Olmedo (2009, p.128).

1.5.1.2. Modelo Clasificación de Gastos. Al momento de crear un presupuesto personal se puede observar que existen diversas herramientas en el mercado que se basan en los ingresos y gastos, buscando dejar espacio para el ahorro y futuras inversiones. Para lograr una correcta administración de gastos el autor Olmedo (2009) propone el siguiente análisis y modelo:

- Adopte un sistema o mecanismos para la elaboración del presupuesto que le ayude a organizar sus gastos mensuales, estos pueden ser clasificados por meses, o por ítem, o por importancia.
- Registre todos los ingresos y los gastos durante el mes y tenga cuidado de no gastar más de la cuenta, para esto se presenta un análisis que se pueden seguir, para una correcta gestión.

Averigüe de dónde viene el dinero (ingresos) y en qué se utiliza (gastos) en cada mes.

- Los ingresos pueden incluir: Salarios, pensión, dividendos, otros ingresos como comisiones, rendimientos y arrendamientos.
- Los gastos pueden incluir:

Gastos fijos: aquellos cuyo valor es constante o fijo durante un largo período de tiempo, por ejemplo, arriendo, hipoteca, créditos, servicios públicos, mercado, impuestos, transporte y educación.

Gastos flexibles: aquellos que pueden variar basados en sus actividades, por ejemplo, ahorros, servicios privados, comidas esporádicas en restaurantes, gastos vacacionales y gastos personales.

Para la clasificación de los gastos personales el autor Olmedo (2009). Propone la matriz de dos entradas por cada lado (Figura 1). Este modelo nos ayuda como punto de referencia para la elaboración del diseño del modelo para la arquitectura propuesta en este trabajo.

Figura 1

Matriz clasificatoria de gastos en Colombia.

MATRÍZ CLASIFICATORIA DE GASTOS[1,2]

	URGENTE	NO URGENTE
Importante	• Arriendo o cuota de vivienda. • Servicios públicos, administración. • Mercado. • Colegios. • Salud. • Gasolina. **I**	• Ahorro pensión. • Base para compra de vivienda. • Fondo para emergencias. • Ahorro para automóvil. • Seguros. • Ahorro para esparcimiento y otros (vacaciones). • Impuestos. **II**
No Importante	• Gastos excesivos en tecnología y comunicaciones. • Consumos de tarjeta de crédito. • Gastos altos en cuidado de personal. **III**	• Vacaciones “apalancadas”. • Carros lujosos y a crédito. • Ropa costosa. • Vivienda costosa. • Demasiados restaurantes lujosos. **IV**

Nota. Datos tomados de Hábitos Financieros de Colombia. Olmedo (2009, p.138).

1.5.1.3. Antecedentes. Rodríguez (2021) creó un prototipo de aplicación móvil para el análisis y control de finanzas personales mediante la incorporación de análisis predictivo, haciendo uso de técnicas de Machine Learning para la gestión, análisis y control de gastos. El método usado para esta investigación fue de tipo deductivo, El desarrollo de la propuesta se trabajó con una arquitectura orientada a servicios SOA, donde se construyó un prototipo móvil con ayuda de

Python, Django Rest Framework y React Native. Como resultado la aplicación permite llevar un control de los gastos personales por categorías, que utiliza los datos financieros (ingresos y gastos) para dar un panorama más claro de los gastos mensuales apoyándose en gráficas resultado de un análisis descriptivo.

Martínez D. (2014) realizó un trabajo de investigación que busca simplificar el proceso de desarrollo de software, teniendo como punto de partida la abstracción de un problema para construir modelos que permitan una representación desde una solución general. El investigador emplea modelos para la elaboración de un lenguaje determinado y los módulos para la generación de código. Como resultado se desarrolla un programa bajo la arquitectura modelo vista controlador que se encarga de brindar al desarrollador un mecanismo para la generación de código fuente basado en un metamodelo, mejorando la productividad, calidad, mantenibilidad y la reutilización. Este trabajo nos permite evaluar los modelos para la generación de código automático y evaluar si se pueden aplicar o adaptar al problema que se quiere solucionar.

1.5.1.4. Aplicaciones Existentes. Existen en el mercado aplicaciones creadas para brindar soluciones a la gestión de gastos personales a partir de presupuestos. Algunas de las más usadas son las siguientes:

- **Mint**

Mint es una aplicación móvil disponible para los sistemas operativos Android y iOS, diseñada para la gestión eficiente de gastos y presupuestos personales, facilita la gestión de gastos, permite a los usuarios crear presupuestos de manera sencilla y categoriza de manera automática las transacciones a medida que se ingresan en la aplicación. También ofrece recomendaciones personalizadas que ayudan al usuario a elaborar presupuestos más

efectivos, basado en tendencias de gastos versus ingresos de meses anteriores, lo que puede resultar en un ahorro significativo de dinero y la reducción de gastos innecesarios. (Mint,2023)

La aplicación brinda la posibilidad de establecer un límite por cada gasto y notificar al usuario cuando estén cerca de exceder el presupuesto.

En cuanto a la seguridad, Mint ofrece funcionalidades como la autenticación multifactor, que adicional a la validación de la contraseña, verifica la identidad del usuario a través de SMS o correo electrónico. Además, en caso de robo del dispositivo móvil, la aplicación permite borrar de forma remota toda la información de la cuenta de Mint, garantizando la protección de los datos sensibles del usuario. Finalmente ofrece encriptación de datos SSL de 128 bits.

- **YNAB (You Need a Budget)**

YNAB es una aplicación disponible en múltiples plataformas, incluyendo iPad, iPhone, Apple Watch, Alexa y Android. Una de sus características destacadas es la capacidad del sistema de actualizar la información en tiempo real. En cuanto al seguimiento de metas de ahorro y gastos, esta aplicación ofrece una representación visual intuitiva mediante el uso de colores para indicar el progreso de cada una de estas metas. Por ejemplo, cuando un gasto supera el presupuesto asignado, se resalta en rojo, y se proporciona información sobre el monto excedido. Asimismo, se muestra el presupuesto restante a medida que se asigna a cada categoría.

La aplicación también ofrece la posibilidad de generar reportes detallados que comparan los gastos con los ingresos mes a mes con fechas de búsqueda parametrizables. Además, se puede acceder a un desglose de gastos por categoría, lo que facilita la comprensión de los patrones de gasto. En cuanto a la seguridad de la información, la aplicación

incorpora medidas robustas, como el cifrado de datos mediante SSL de 256 bits y la autenticación de dos factores (2FA). (YNAB,2023)

- **Pocket Guard**

Esta aplicación está diseñada para ser usada sobre sistemas operativos Android e IOs y permite clasificar los gastos en categorías y asignarles un presupuesto, adicionalmente permite agrupar en una misma categoría gastos relacionados. Por ejemplo: pago de servicios públicos, mercado y gastos del hogar en general en la categoría hogar. Además, permite notificar al usuario cuando se haya gastado un porcentaje de dicho presupuesto. Permite también crear un plan de pagos de una deuda para de esta manera ayudar a que los usuarios no se endeuden con facilidad. En cuanto al aspecto de seguridad, los datos de los usuarios están protegidos con un algoritmo SSL de 256 bits (Pocket Guard,2023).

- **GoodBudget**

Una de las principales características de esta aplicación es dividir los ingresos mensuales en “sobres”. Esto significa que automáticamente divide el presupuesto y lo asigna para cada categoría creada en la aplicación. En este sentido, solo se permite gastar el dinero asignado a cada sobre a uno o más usuarios con acceso. Otra de las funcionalidades soportadas es permitir crear gastos compartidos con actualización en tiempo real. Esta aplicación se ejecuta sobre web, IPhone y Android (GoodBuget,2023)

1.5.2. Marco Conceptual

El marco conceptual nos brinda la información necesaria de los conceptos mas importantes para el entendimiento del presente documento abordando los temas relevantes de

finanzas e ingeniería de software que se deben aplicar para la generación de un prototipo de aplicación.

1.5.2.1. Finanzas. Finanzas, según el diccionario, se refiere a hacienda, caudal y negocios. Se le define como el conjunto de actividades mercantiles relacionadas con el dinero de los negocios, de la banca y de la bolsa; y como el grupo de mercados o instituciones financieras de ámbito nacional o internacional. Así, la palabra “finanzas” se puede definir como el conjunto de actividades que, a través de la toma de decisiones, mueven, controlan, utilizan y administran dinero y otros recursos de valor. “Finanzas” es una palabra que provoca reacciones distintas, las cuales pueden ir desde el interés y la curiosidad hasta el temor y la angustia. Para la mayoría de las personas las finanzas se perciben como algo complejo, poco accesible y que solo es manejado por los estudiosos; en realidad, estas pueden ser tan sencillas, interesantes y útiles si se cuenta con los conocimientos financieros esenciales (García, 2014).

1.5.2.2. Finanzas Personales. Se puede definir el concepto de finanzas personales como “La forma como las personas planean, organizan, dirigen y controlan los recursos económicos para satisfacer sus necesidades.” López (2016).

También pueden verse las finanzas personales como una herramienta de apoyo en la vida cotidiana de los individuos y las familias que les permite a estos controlar los ingresos y egresos, y tomar decisiones adecuadas para mantener un equilibrio económico.

“Manejo de las finanzas personales y su impacto en el desempeño laboral de los trabajadores de la empresa financiera independencia S. A. de C. V. en Coatzacoalcos Ver”, Alvarado Nolasco (2013)

Otra definición que aborda el concepto de finanzas personales desde otro punto de vista consiste en el registro de manera cotidiana de hechos económicos por las personas o por sus familias y que son de vital importancia para la relación con entidades financieras y servicios que determinan dicha relación tales como cuentas, créditos, tarjetas de crédito, entre otros.

Lo anterior muestra que las finanzas personales no afectan únicamente al individuo, sino que también su relación con el entorno. Por lo tanto, una buena educación financiera permite el manejo adecuado de las finanzas personales y además blinda a los individuos contra la toma de malas decisiones financieras y, en consecuencia, protege al sistema financiero y al mercado global contra los malos resultados por efecto de los malos hábitos financieros de las personas.

1.5.2.3. Presupuesto. El presupuesto es el documento en el que se ingresa de manera adecuada y por escrito todos los gastos e ingresos correspondientes a una unidad familiar durante determinado período de tiempo. Igualmente, permite planear el destino del dinero de las personas, calculando los ingresos monetarios de cualquier tipo y los egresos monetarios de cualquier tipo. (López,2016)

El presupuesto se puede clasificar en tres tipos:

- Excedente: cuando se anticipan las ganancias.
- Equilibrado: Si los ingresos y los egresos son iguales.
- Deficitario: Si los egresos son mayores que los ingresos.

Los presupuestos permiten lo siguiente a las personas:

- Conocer en qué se gasta el dinero y detectar gastos innecesarios.
- Ajustar y priorizar los gastos.

- Controlar los gastos.
- Planificar ahorros.
- Ajustar el nivel de vida según los ingresos.

1.5.2.4. Metodología Iterativa Incremental. La metodología iterativa incremental es un enfoque de desarrollo de software que se centra en la entrega temprana y frecuente de funcionalidades, a través de ciclos iterativos de desarrollo. Según Sommerville (2016), este enfoque permite una adaptación continua a los cambios en los requisitos del cliente, al tiempo que proporciona una mayor visibilidad del progreso del proyecto.

Según Pressman (2014), la metodología iterativa incremental se basa en la división del proyecto en pequeñas partes incrementales, que se desarrollan en ciclos repetitivos. Cada iteración produce un incremento en la funcionalidad del sistema, lo que permite una entrega gradual y una mayor flexibilidad para adaptarse a los cambios.

Las características principales de la metodología iterativa incremental incluyen la planificación flexible, la entrega continua, el feedback constante y el enfoque en riesgos (Boehm, 1988). Estas características permiten una adaptación ágil a los cambios y una mayor eficiencia en el desarrollo del software.

Según Schwaber y Beedle (2002), la metodología iterativa incremental proporciona varios beneficios, como una entrega temprana de valor, una mayor visibilidad del progreso del proyecto y una mayor adaptabilidad a los cambios. Sin embargo, también presenta desafíos, como la necesidad de una comunicación y colaboración efectiva entre el equipo de desarrollo y los stakeholders.

1.5.2.5. Modelo Evolutivo de Desarrollo de Software. El modelo evolutivo de desarrollo de software es un enfoque iterativo e incremental que se basa en la construcción progresiva y gradual del producto. A diferencia de los modelos tradicionales, el modelo evolutivo permite adaptarse y modificar el producto a medida que se van obteniendo retroalimentaciones y se van identificando nuevas necesidades del usuario (Pressman & Maxim, 2015). En este modelo, el desarrollo del software se divide en pequeñas fases o iteraciones, donde cada iteración produce un incremento en la funcionalidad del sistema (Sommerville, 2016). Durante cada ciclo, se planifica, diseña, desarrolla, prueba e implementa una porción del software lo que permite que el producto se entregue gradualmente y que los clientes puedan ver resultados tangibles de manera temprana.

El modelo evolutivo es especialmente útil en proyectos donde los requisitos no están completamente definidos desde el principio o donde pueden cambiar a lo largo del tiempo. Al permitir una planificación flexible y adaptativa, este enfoque facilita la incorporación de cambios y ajustes en el producto final, garantizando su alineación con las necesidades del usuario y su capacidad para satisfacer los objetivos del proyecto (Larman, 2004).

1.5.2.6. Arquitectura de Software ISO 42010. La arquitectura de software, definida según la norma ISO/IEC 42010, abarca la estructura fundamental de un sistema de software, incluyendo sus componentes, relaciones y principios guía para su diseño y evolución (ISO/IEC, 2011). En el contexto del desarrollo de aplicaciones, esta arquitectura juega un papel crucial en la organización y gestión de la complejidad del software, facilitando la comunicación y colaboración entre los miembros del equipo de desarrollo (Bass, Clements, & Kazman, 2012).

Conforme a la norma ISO/IEC 42010, proporciona una estructura para describir los sistemas de software desde diferentes perspectivas, incluyendo la estructura, el

comportamiento, las funciones y las propiedades de calidad. Esta norma establece un marco para la arquitectura de software que incluye cuatro puntos clave: los stakeholders, las vistas, los puntos de vista y los modelos (ISO/IEC, 2011).

1.5.2.7. Java Spring Framework - Spring Boot. Java Spring Framework (Spring Framework) es un popular marco de trabajo para estructura empresarial de código abierto para crear aplicaciones independientes de nivel de producción que se ejecutan en la máquina virtual Java (JVM). Java Spring Boot (Spring Boot) es una herramienta que hace que el desarrollo de aplicaciones web y microservicios con Spring Framework sea más rápido y fácil a través de tres funcionalidades principales:

- Configuración automática
- Un enfoque obstinado de la configuración
- La capacidad de crear aplicaciones independientes

Estas características funcionan juntas para brindarle una herramienta que le permite configurar una aplicación basada en Spring con una configuración y preparación mínimas. (International Business Machines Corporation,2023)

1.5.2.8. Maven. Apache Maven es una herramienta de comprensión y gestión de proyectos de software. Basado en el concepto de modelo de objetos de proyecto (POM), Maven puede gestionar la construcción, los informes y la documentación de un proyecto desde una pieza de información central. (Maven,2023)

El objetivo principal de Maven es permitir que un desarrollador comprenda el estado completo de un esfuerzo de desarrollo en el menor tiempo posible. Para lograr este objetivo, Maven aborda varias áreas de preocupación:

- Facilitando el proceso de construcción
- Proporcionar un sistema de construcción uniforme
- Proporcionar información de calidad sobre el proyecto.
- Fomentar mejores prácticas de desarrollo

1.5.2.9. PostgreSQL. PostgreSQL es un potente sistema de base de datos relacional de objetos de código abierto que utiliza y amplía el lenguaje SQL combinado con muchas funciones que almacenan y escalan de forma segura las cargas de trabajo de datos más complicadas. PostgreSQL se ha ganado una sólida reputación por su arquitectura comprobada, confiabilidad, integridad de datos, conjunto de características sólidas, extensibilidad y la dedicación de la comunidad de código abierto detrás del software para ofrecer constantemente soluciones innovadoras y de alto rendimiento. PostgreSQL se ejecuta en todos los principales sistemas operativos, cumple con ACID desde 2001 y tiene potentes complementos, como el popular extensor de bases de datos geoespaciales PostGIS. (PostgreSQL,2023).

1.5.2.10. Apache Velocity. Apache Velocity es una biblioteca de plantillas de código abierto utilizada para la generación dinámica de contenido en aplicaciones Java. Permite la separación de la lógica de presentación del código de la aplicación, facilitando así el desarrollo de aplicaciones con alto nivel de escalabilidad y mantenibilidad. Se utiliza un lenguaje de sintaxis sencillo para definir las plantillas y ofrece funciones para integrar datos dinámicos en ellas. (Apache Software Foundation,2024)

1.5.3. Marco Legal

Documento CONPES 4005 - Política Nacional de Inclusión y Educación Económica y Financiera

El informe señala que, en materia de educación económica y financiera, la Ley 1328 de 2009 estableció como un derecho del consumidor financiero recibir una adecuada educación sobre productos y servicios, derechos y obligaciones y también definió una obligación de establecer programas de educación para sus clientes. En concreto, el Artículo 8 establece la obligación de las entidades financieras de promover programas de educación financiera a clientes y público en general. En ese mismo sentido, indica que la Superintendencia Financiera de Colombia debe promover la educación financiera. (Disposiciones analizadas por Avance Jurídico Casa,2023)

De otra parte, el decreto 457 de 2014 crea el Sistema Administrativo Nacional para la Educación Económica y Financiera que en su artículo 2 señala lo siguiente: *"Organízase el Sistema Administrativo Nacional para la Educación Económica y Financiera con el fin de coordinar las actividades estatales y de los particulares para lograr un nivel adecuado de educación económica y financiera de calidad para la población."* (DECRETO 457, 2014).

En cuanto a programas de educación en finanzas, en El Plan Nacional de Desarrollo 2010-2014, Ley 1450 de 2011 en su artículo 145, señala que se incluirá la educación financiera en el diseño de programas para el desarrollo de competencias básicas por parte del Ministerio de Educación Nacional. (Disposiciones analizadas por Avance Jurídico Casa,2023)

Finalmente, en 2014, se dio la publicación del Documento No. 26 Mi Plan, Mi Vida y Mi Futuro: Orientaciones Pedagógicas para la Educación Económica y Financiera" enmarcado dentro del Programa de Educación Económica y Financiera para educación básica y media, emitido por el Ministerio de Educación Nacional y Asobancaria, 2014.

Una de las principales entidades financieras en Colombia, el Banco Davivienda (2023) establece el programa de Mis finanzas en casa…y específicamente, ofrece una herramienta web para simulación de un presupuesto personal.

Ley 1581 de 2012: protección de datos personales

Esta ley establece las disposiciones relacionadas con la protección de datos personales.

En el contexto de las aplicaciones web, el artículo 17 establece la obligación de obtener un consentimiento expreso de los usuarios antes de recopilar y utilizar sus datos personales. Dentro del contexto de este proyecto, se debe recordar que se van a recopilar datos referentes a los gastos de los usuarios de nuestra aplicación, y posiblemente cuentas de contrato asociados a servicios públicos y privados, así como también datos referentes a suscripciones.

En cuanto a la seguridad de los datos personales, el artículo 19 se refiere a la necesidad de establecer mecanismos técnicos y administrativos para proteger los datos personales con el objetivo de garantizar la seguridad de los datos almacenados y transmitidos a través de la aplicación.

El artículo 25 indica que los usuarios tienen derecho a conocer, actualizar y rectificar su información personal. En este sentido, para nuestro proyecto, el usuario debe tener la posibilidad de actualizar sus datos personales que se encuentren almacenados en la plataforma.

1.6. Metodología de la Investigación

1.6.1. Tipo de estudio

El estudio realizado tiene como objetivo brindar una herramienta para ayudar a resolver un problema recurrente en la sociedad que es la gestión de los gastos personales, lo que requiere la implementación de diferentes enfoques metodológicos. Para la investigación se trabajará desde tres enfoques metodológicos: el área de investigación y el área ingenieril, que abarca la implementación de una metodología enfocada en el diseño del sistema y el desarrollo, dando como resultado un prototipo funcional.

El proyecto trabajará desde un enfoque investigativo exploratorio que busca los principales factores relacionados en el problema y su posible solución. Posteriormente se realizará un estudio experimental en diferentes iteraciones abordando los enfoques metodológicos basados en la investigación y el diseño, con la generación de un diseño inicial se realizará el desarrollo y, junto con lo anterior, se realizarán prototipos funcionales y se concluirán con una evaluación de usabilidad.

En primer lugar, para implementar un prototipo de software para una aplicación de finanzas personales, se deben revisar y escoger las necesidades principales de los usuarios de la aplicación para definir los requisitos funcionales y también los requisitos no funcionales.

En el mismo sentido, desde el punto de vista técnico se evaluarán los lenguajes de programación disponibles para el desarrollo y despliegue de la aplicación de tipo web, tanto en backend como para el frontend, el motor de base de datos a utilizar y así mismo se investigarán los frameworks disponibles para la generación automática de código fuente y se escogerá aquel que mejor se acomode a las necesidades del proyecto.

En el contexto del estudio experimental, se realizará el desarrollo del prototipo, usando frameworks de generación automática de código fuente como una de las principales características de la aplicación.

Finalmente, el estudio de la evaluación de la usabilidad consiste en evaluar la facilidad en la usabilidad de la aplicación y la satisfacción de los usuarios con el uso de esta y así mismo se evaluarán los tiempos de respuesta de la aplicación teniendo en cuenta que se va a implementar el modelo de CRUDS basado en técnicas de generación automática de código fuente.

1.6.2. Método de investigación

El método de investigación cuantitativo nos ayudará a realizar los respectivos procesos de análisis de los datos a partir de la información obtenida del estudio de las áreas del conocimiento aplicadas a las finanzas personales, lo que nos permite proporcionar un modelo práctico que sirva como base para el desarrollo de un prototipo de software enfocado en un sistema de gestión de gastos personales, así mismo nos permitirá medir los resultados obtenidos en el proyecto mediante un conjunto de técnicas tales como encuestas de usuario para medir el nivel de funcionalidad y usabilidad de la aplicación, realizar mediciones sobre los procesos de generación de código automático y obtener tiempos de respuesta del programa en el momento en que se esté ejecutando. Estos análisis nos brindarán información suficiente para determinar la viabilidad del proyecto.

El proyecto se trabajara desde tres enfoques metodológicos los cuales están representados por un aspecto metodológico que considera todo lo relacionado con el área de

investigación del problema, el segundo aspecto nos permite la implementación de metodologías para el desarrollo del proyecto, tales como Scrum, en donde se contempla un enfoque ingenieril, el cual está dividido por el diseño del sistema y finalmente el desarrollo, en donde se propone la realización de un prototipo funcional utilizando un modelo de desarrollo de tipo evolutivo.

1.6.3. Fuentes y técnicas para la recolección de la información

Para el estudio referente al control de finanzas personales se determina que nos debemos apoyar principalmente en fuentes documentales, así mismo para las fases de planeación, análisis, diseño y desarrollo de la arquitectura y el prototipo de software se requiere de diversidad de documentos, los cuales nos ayudará a tener un mejor entendimiento relacionado con los hábitos de gastos de las personas y así generar un modelo adecuado que permita al usuario final tener una buena experiencia de uso. Es importante para el desarrollo del proyecto el estudio de arquitecturas, metodologías, planes, modelos y métodos que ayuden a elaborar un producto de calidad, que tenga usabilidad, bajo costo y que cumpla con todos los requerimientos propuestos. Las fuentes documentales empleadas fueron las siguientes: libros tales como Introducción a las finanzas (García,2014), artículos (Finanzas Personales (Sarmiento ,2009), un Concepto que va más allá de su Estructura (López,2016)), leyes como: Ley 1328 de 2009, ley 1450 de 2011. Revistas (Las finanzas personales (Olmedo,2009)), páginas web: Guía de la metodología scrum. (Atlassian, 2023).

En última instancia se propone realizar encuestas de experiencia de usuario donde el objetivo es recibir retroalimentación para asegurar la funcionalidad y usabilidad de la aplicación, esto ayudados de un pequeño grupo de estudio.

1.6.4. Tratamiento de la información

La información generada en el proyecto será de carácter público y estará almacenada en la base de datos de la universidad Distrital, las fuentes documentales se usarán para el análisis, diseño y desarrollo de una arquitectura de software basada en un sistema de gestión de gastos personales, las encuestas generadas nos permitirán medir la usabilidad y practicidad de la aplicación donde la información obtenida solo se enfocara en extraer información importante sobre la experiencia del usuario al usar la aplicación, por lo cual no se solicitará información personal.

1.6.4.1. Técnicas de Recolección de Datos. Teniendo en cuenta que el presente proyecto se aborda desde un método de investigación cuantitativo, se realizará un piloto de encuesta a una muestra que nos permita medir la eficacia de la hipótesis planteada. La encuesta propuesta es la siguiente:

1. ¿Usa o ha usado alguna herramienta tecnológica para el control de los gastos personales?

SI o NO

2. ¿Qué tan fácil considera que es el uso de la aplicación?

Fácil

Normal

Complejo

3. Califique de 1 a 4 cada una de las siguientes características que considera más importantes para una aplicación de gestión de gastos personales, siendo 4 considero que es demasiado importante, 3 me parece importante, 2 me parece bien que tenga la función, sin embargo, considero que es prescindible y 1 me parece que no es importante. Por favor indique al final qué otras características funcionales consideran que se deberían tener en cuenta para una aplicación de administración de gastos personales.

a. Presupuesto predictivo basado en los hábitos de gastos y los ingresos de los últimos 3 meses

b. Presupuesto compartido por varios miembros de la familia

c. Alerta de gastos excesivos por cada categoría del presupuesto

d. Gráficos y análisis de gastos vs presupuesto

e. Creación de categorías y gastos personalizables

4. ¿Considera que después de probar la aplicación, mejoró el control sobre su presupuesto y sus gastos personales?

SI o NO

5. ¿Seguiría usando la aplicación de finanzas personales por lo menos durante los próximos 6 meses?

SI o NO

1.7. Organización de Trabajo de Grado

Parte I. Contextualización de la Investigación

Gestionar de manera adecuada los gastos personales es un desafío para muchos individuos en la actualidad, lo que podría en algún momento llevar a los individuos a tener problemas financieros. Este problema, derivado en parte de la escasa educación financiera y también por la falta del uso de herramientas adecuadas para la gestión de gastos, ha motivado la realización del presente proyecto de investigación. En este capítulo introductorio, se identifica y formula el problema central: ¿cómo diseñar e implementar un prototipo funcional escalable para una aplicación web destinada a la gestión de gastos personales? A partir de esta pregunta fundamental, se delinean objetivos generales y específicos, se establece una hipótesis y se justifica la relevancia teórica, metodológica y práctica de la investigación. Posteriormente se detallan los enfoques metodológicos que guiarán el estudio, los cuales incluyen aspectos exploratorios, experimentales y cuantitativos. Asimismo, se discuten las fuentes y técnicas de recolección de información, junto con el tratamiento previsto para los datos obtenidos, lo que proporcionará una base sólida para el desarrollo y evaluación del prototipo propuesto.

Parte 2. Desarrollo de la Investigación

En este capítulo se realiza el levantamiento de los requerimientos mediante el análisis y generación de historias de usuario y los casos de uso. Se hace elección del Modelo Evolutivo para el desarrollo de Software, caracterizado por su enfoque iterativo e incremental, abarcando desde la comunicación de los requerimientos y la planeación hasta el despliegue en cada iteración.

La fase de planeación determina las tecnologías a emplear y establece una hoja de ruta clara para el proyecto, definiendo tres iteraciones que incluyen las actividades esenciales de cada fase. En el modelado, se diseña el modelo de datos base relacional y la capa de servicio, así como los controladores del sistema, sentando las bases para la implementación de las funciones CRUD (Crear, Leer, Actualizar y Eliminar).

El desarrollo se inicia con la creación de un prototipo funcional que permita a los usuarios registrarse, iniciar sesión y registrar sus ingresos y gastos. Posteriormente, se procede al despliegue inicial en un entorno local para facilitar la retroalimentación de los usuarios y ajustar el prototipo según sus necesidades y comentarios. En cada iteración, se agregan nuevas funcionalidades basadas en las necesidades identificadas, lo que garantiza un proceso de desarrollo continuo y adaptable.

Parte 3. Cierre de la Investigación

Finalmente, para el cierre del presente trabajo se llevan a cabo las pruebas funcionales unitarias y se ejecutan los casos de prueba de acuerdo con las historias de usuario y casos de uso los cuales definieron los requerimientos. Una vez se realicen las pruebas y estas cumplan con los requerimientos funcionales, se llevarán a cabo pruebas de usabilidad a una muestra de usuarios a los cuales se les realizará una encuesta para medir de manera cuantitativa el resultado de nuestra investigación.

1.8. Estudio de Sistemas Previos

Rodríguez (2021) en su investigación titulada "**Diseño de prototipo para el control de finanzas personales mediante análisis predictivo en una aplicación móvil**", adoptó un enfoque descriptivo para examinar la gestión y control de gastos. En este estudio, se realizó un análisis deductivo complementado con técnicas de Machine Learning, respaldando así la naturaleza descriptiva de la investigación. La muestra consistió en un conjunto de datos aleatorios que registraron los ingresos de los últimos 12 meses. A través de un proceso predictivo, se generaron datos adicionales, resultando en una muestra final de 300 datos. Para la recolección de datos, se emplearon técnicas específicas de recolección de información, como artículos, textos y encuestas.

El objetivo general del autor fue diseñar un prototipo de aplicación móvil destinado al análisis y control de finanzas personales, mediante la integración de análisis predictivo y técnicas de machine learning, con la finalidad de facilitar la gestión, análisis y control de los gastos presentes y futuros de los usuarios. Para alcanzar este objetivo general, se propusieron los siguientes objetivos específicos: a) Investigar las técnicas de machine learning más utilizadas para el análisis predictivo, desarrollando un modelo dentro de la aplicación para proporcionar una visión clara del control de los gastos personales; b) Crear un prototipo de aplicación utilizando Python para implementar el modelo de análisis predictivo, con un componente de backend API REST construido con Django Rest Framework y un componente de frontend en React Native, compatible con dispositivos móviles Android e iOS; c) Diseñar una interfaz de usuario intuitiva y atractiva para la aplicación móvil.

Como resultado del primer objetivo específico, se realizó una investigación exhaustiva y una implementación práctica de los principales métodos de análisis predictivo, concluyendo que la técnica de árboles de decisión era la más adecuada para el modelo de análisis dentro de la aplicación. En relación con el segundo objetivo, se logró desarrollar un prototipo funcional de la aplicación móvil, construido sobre una arquitectura orientada a servicios y utilizando Python y Django Rest Framework para el backend API REST, así como React Native para el frontend. Finalmente, para cumplir con el último objetivo, se diseñó una interfaz de usuario intuitiva y atractiva. En conjunto, estos logros han permitido alcanzar el objetivo general del proyecto, ofreciendo una solución integral y efectiva para el análisis y control de finanzas personales a través de la aplicación móvil.

La relación entre el trabajo propuesto por el autor con respecto al presente trabajo se centra en el ámbito de la gestión de gastos personales y el desarrollo de una aplicación para la administración de estos, donde en ambos proyectos se busca desarrollar una herramienta efectiva para gestionar las finanzas personales de manera más eficiente y conveniente. Con respecto a los elementos utilizados se exploró el planteamiento realizado por los autores de la gestión de gastos personales y su implementación en una aplicación enfocada en una arquitectura web, en particular, su contribución se centró en los análisis estadísticos y los informes presentados al usuario dentro de la aplicación. Utilizamos parte de las estructuras proporcionadas como base para organizar y presentar los gastos en forma de gráficos.

Prieto y Madrid (2021) en su investigación titulada "**Prototipo generador de código fuente para aplicaciones de servicios**", proponen la construcción de un prototipo generador de código que pueda aportar una reducción de errores comunes en la codificación, junto con

una disminución en los tiempos de implementación, permitiendo centrar el equipo en el proceso de análisis y diseño del negocio. La investigación se enmarca en un estudio exploratorio de desarrollo de software, con un énfasis en la ingeniería de software y la mejora de procesos. Las técnicas utilizadas para la recolección de datos incluyeron la ejecución de pruebas funcionales con Swagger sobre las operaciones del CRUD del sistema. Además, se llevaron a cabo pruebas de carga utilizando Jmeter, con un total aproximado de 4000 ejecuciones en 3,59 segundos.

El objetivo general del trabajo referenciado es desarrollar un prototipo para la construcción de una arquitectura de servicios API REST, partiendo de un modelo de datos en SQL que permita parametrizar el código fuente utilizado en los productos de software orientados a servicios, utilizando como caso de estudio el proceso de reserva de libros de una biblioteca. Para lograr este objetivo. Como objetivos específicos los autores proponen: a) implementar una arquitectura de servicios API REST aplicando buenas prácticas de desarrollo de software; b) desarrollar un prototipo de un generador de código que se ajuste a la arquitectura propuesta y a las tecnologías seleccionadas, con el fin de optimizar el proceso; y c) validar la implementación del caso de estudio propuesto utilizando el generador de código para asegurar la funcionalidad propuesta.

Para las conclusiones del trabajo los autores destacan la implementación de prácticas sugeridas y herramientas que faciliten la creación automática de código, lo que contribuye a reducir tiempos y esfuerzos en la codificación de proyectos, generando así una reducción de costos para las organizaciones. Además, enfatizan la adopción de estándares en la escritura de código y la importancia de una arquitectura sólida en el proceso de análisis, diseño y definición

de proyectos de software. En resumen, concluyen que la aplicación de una correcta arquitectura de servicios API Rest, junto con el uso de buenas prácticas y herramientas adecuadas, puede mejorar significativamente la eficiencia y calidad en la creación de software, lo que contribuye al crecimiento tecnológico y competitividad de las organizaciones.

En relación con nuestro proyecto de gestión de gastos, se plantea la generación de código eficiente para implementar las operaciones CRUD del sistema. Para lograr este objetivo, se propone desarrollar un modelo de generación de código que permita automatizar este proceso. Para lo cual, se estudian algunos elementos de buenas prácticas de este trabajo para la generación de código. Este enfoque permitirá agilizar el desarrollo del software, reducir posibles errores y garantizar una mayor consistencia en la implementación de las funcionalidades relacionadas con la gestión de gastos.

Rizki y Juwono (2020) proponen el desarrollo de la aplicación móvil denominada "**Android-Based Mobile Application Development Expensify: Personal Expense Manager**". Este estudio fue realizado desde la perspectiva de una investigación aplicada, se enfoca en usuarios de dispositivos móviles con sistema operativo Android y no se especifica una muestra particular dado que sus resultados se validan mediante la verificación funcional de los casos de prueba derivados de las historias de usuario definidas en los requerimientos funcionales. Para la recopilación de la información, se empleó la revisión de literatura como instrumento principal.

El trabajo presenta como objetivo general desarrollar una aplicación móvil para Android que ayude a los usuarios a administrar sus finanzas personales basados en la metodología de Desarrollo Rápido de Aplicaciones (RAD) por sus siglas en inglés. Para lograr este objetivo, se

definieron los siguientes objetivos específicos: a) Permitir a los usuarios registrar transacciones de ingresos y gastos con detalles por categoría. b) Brindar a los usuarios la posibilidad de visualizar resúmenes financieros, informes y gráficos basados en los datos registrados. c) Permitir a los usuarios crear y administrar presupuestos y d) Facilitar a los usuarios el análisis, planificación y control de sus finanzas personales.

Las conclusiones principales del estudio destacan la adopción de la metodología de Desarrollo Rápido de Aplicaciones (RAD), que prioriza el prototipado de la aplicación y la adaptación según la retroalimentación de los usuarios. Se identificaron 14 requisitos funcionales agrupados en tres categorías principales: gestión de transacciones, gestión de categorías y gestión de presupuestos. La aplicación Expensify es compatible con dispositivos Android a partir de la versión API 21 (Android 5.0 Lollipop). Además, se recomienda la realización de investigaciones futuras para incorporar funciones adicionales, como la sincronización de datos con otras plataformas y la importación automática de transacciones bancarias.

En relación con nuestro proyecto, se destaca la elección para nuestro trabajo de la metodología Rapid Application Development (RAD) por sus siglas en inglés de manera que se realicen entregas de versiones funcionales de la aplicación cuyas funcionalidades se van refinando e incrementando con cada iteración de entrega. Otro de los aspectos a destacar es que el trabajo de estos autores nos provee una idea en cuanto al diseño de las interfaces de usuario para permitir al usuario ingresar y consultar sus gastos por categorías y adicionalmente en nuestro proyecto se adiciona el concepto de subcategoría el cual corresponde específicamente al rubro al cual se aplica el gasto. La subcategoría se asocia directamente con

la categoría. Fincalmente, vale la pena destacar el hecho que en el presente proyecto se realiza un diseño detallado y sistemático de la base de datos de tipo relacional, mediante los diagramas ER y relacional para definir el modelo de datos de la aplicación, a diferencia de lo descrito en el trabajo en mención en donde no especifican la manera en que hacen el diseño de la base de datos.

Por otra parte, Velmurugan (2020), en su investigación titulada "**Development of an Expense Manager Android Application**" plantea un estudio de investigación aplicada dirigida específicamente a usuarios de dispositivos móviles con sistema operativo Android. Para la recopilación de datos, se adopta el enfoque de revisión de la literatura existente para identificar necesidades de los usuarios y soluciones existentes en el ámbito de la gestión de gastos personales.

El objetivo general de su investigación plantea el desarrollo una aplicación móvil para Android que facilitara a los usuarios la gestión tanto de sus gastos personales como grupales. Como objetivos específicos se definen: a) Eliminar la necesidad de usar notas adhesivas y hojas de cálculo que podrían resultar confusas para el control de gastos personales. b) Permitir a los usuarios registrar transacciones diarias y/o dinero prestado y gastos grupales. c)Sugerir opciones de inversión basadas en el presupuesto del usuario. c) Proporcionar acceso a un marketplace externo a la app y permitir hacer pagos desde distintas billeteras en dicho sitio y registrar los pagos realizados en la aplicación, d) Brindar noticias financieras verificadas y e) Ofrecer cupones de descuento.

Las principales conclusiones del estudio expresan que la aplicación desarrollada busca contribuir a eliminar la necesidad de utilizar notas adhesivas y hojas de cálculo para el seguimiento de gastos, ofreciendo en su lugar una experiencia más intuitiva, organizada y centralizada. También se sugiere la inclusión de funciones adicionales, como la integración con números de teléfono, correos electrónicos y redes sociales, para mejorar la portabilidad de la aplicación. Además, se enfatiza la eficacia de la aplicación para realizar el cálculo preciso y el registro de gastos compartidos, junto con sus respectivas fechas en las que se ejecutan.

En relación con nuestro proyecto actual, este estudio proporciona una guía en cuanto a los requisitos asociados con el registro y control de gastos personales de manera estructurada y centralizada. Asimismo, la propuesta de diseñar un modelo de datos escalable se alinea con la capacidad mencionada en el artículo para gestionar grandes cantidades de datos de manera eficiente. Además, nuestro proyecto también permitirá a los usuarios consultar informes de gastos y presupuestos, lo cual está directamente relacionado con el objetivo del artículo de ayudar a los usuarios a rastrear gastos y analizar presupuestos.

Parte II. Desarrollo de la Investigación

Capítulo 2. Marco Metodológico

En este capítulo, se presentan los principales elementos del marco metodológico que hemos seleccionado para el desarrollo de nuestro proyecto. Este enfoque se basa en una metodología iterativa e incremental, que proporciona una estructura flexible y adaptable para la construcción de nuestra aplicación web de gestión de gastos personales. Este marco metodológico nos sirve como guía para el desarrollo efectivo y eficiente de nuestra aplicación, permitiendo una adaptación ágil a los cambios en los requisitos del proyecto y una entrega gradual de funcionalidades.

2.1. Metodología Iterativa Incremental

El desarrollo de nuestro proyecto se basa en una metodología iterativa e incremental que abarca los siguientes aspectos clave:

2.1.1. Iterativo

El proceso de desarrollo de la aplicación se divide en ciclos repetitivos o iteraciones, donde cada ciclo produce un incremento en la funcionalidad del sistema. Durante cada ciclo, se lleva a cabo una planificación detallada, se desarrolla el código correspondiente, se realizan pruebas exhaustivas y se implementa cada parte de la aplicación.

2.1.2. Incremental

El proyecto se construye en pequeños incrementos funcionales. En cada iteración, se añade nueva funcionalidad o se mejora la existente. Esto permite que el producto sea entregado gradualmente, lo que permite ver resultados de manera temprana y proporciona

oportunidades para retroalimentación y ajustes continuos, los incrementos e iteraciones se planean con ayuda de las historias de usuario que dan como resultado los requerimientos del proyecto

2.1.3. Planificación Flexible

Adoptamos una planificación flexible y adaptable para el desarrollo de nuestra aplicación web, donde, inicialmente, se definen los objetivos del proyecto. Sin embargo, los detalles específicos se establecen en cada iteración. Este enfoque nos permite adaptarnos rápidamente a los cambios en los requisitos y prioridades del sistema, asegurando así un resultado satisfactorio.

2.1.4. Entrega Continua

Se entregan incrementos de software funcionales en intervalos regulares y cortos. Esta práctica de entrega continua proporciona un alto nivel de retroalimentación, lo que facilita la detección temprana de problemas y la incorporación de cambios de manera oportuna.

2.1.5. Feedback Constante

En el proyecto, mantenemos una comunicación continua que nos permite recibir retroalimentación regularmente para refinar y ajustar las funcionalidades del producto, asegurando que el desarrollo esté alineado con los objetivos planteados.

2.1.6. Enfoque en Riesgos

Se tratan de buscar los riesgos más importantes desde el inicio de cada iteración, lo que nos permite mitigar los posibles problemas que se puedan presentar.

2.1.7. Pruebas Integradas

Realizamos pruebas continuas e integradas durante todo el proceso de desarrollo para garantizar la calidad del software. Esto nos permite detectar y corregir errores de manera oportuna, asegurando que el producto final cumpla con los estándares de calidad esperados.

2.1.8. Refactorización Continua

Implementamos la refactorización de código de forma continua para mejorar su calidad, mantenibilidad y escalabilidad. Los cambios en el diseño y la arquitectura del sistema se realizan según sea necesario para adaptarse a los requisitos cambiantes y garantizar un desarrollo eficiente y sostenible.

2.2. Modelo Desarrollo Evolutivo

Hemos optado por el modelo evolutivo para el desarrollo de nuestro proyecto de grado, el cual consiste en la implementación de un prototipo de aplicación web destinado a la gestión de gastos personales, utilizando técnicas de generación automática de código fuente para los CRUDS del sistema. Este modelo nos permite adaptarnos a la metodología iterativa incremental para desarrollar la aplicación web.

Inicialmente, describiremos las cinco fases del modelo para la primera iteración, mientras que, para las siguientes, nos centraremos en detallar las funcionalidades a implementar. Cabe mencionar que, para cada iteración, seguiremos las cinco actividades del modelo incremental: comunicación, planeación, modelado, construcción y despliegue.

2.2.1. Comunicación

En cuanto a la comunicación, basaremos el primer incremento en los requisitos funcionales básicos obtenidos a través de la interacción con un grupo reducido de usuarios, centrado en la funcionalidad y la interfaz gráfica de usuario.

2.2.2. Planeación

En la etapa de planeación, durante la primera iteración, estableceremos también las tecnologías para el desarrollo de la aplicación, planificando tres iteraciones o entregables, cada uno abarcando las cinco fases del modelo incremental.

2.2.3. Modelamiento

En la fase de modelado, una vez definidos los requisitos básicos, como ingresos, gastos y presupuestos, diseñamos el modelo de datos base desde una perspectiva empresarial, sobre el cual se apoyará toda la lógica de negocio para el manejo de las finanzas personales. También modelamos la capa de servicio, los controladores del sistema y los mocks de la interfaz gráfica de usuario.

2.2.4. Desarrollo

El desarrollo se centrará en la implementación de la primera entrega funcional, que incluirá un prototipo de la aplicación con funciones básicas de registro de ingresos y gastos.

2.2.4. Despliegue

Por último, el despliegue inicial se realizará en un entorno local, donde la aplicación estará disponible a través de un navegador web dentro de la misma red local.

Este enfoque nos permitirá obtener retroalimentación del usuario y avanzar hacia las siguientes iteraciones de manera progresiva, ajustando y refinando continuamente las funcionalidades del producto.

Tabla 2

Iteraciones modelo de desarrollo evolutivo.

Iteración	Funcionalidad	Caso de Uso
1	Creación del usuario en el sistema.	Autenticación del usuario
2	Autenticación del usuario en la aplicación.	Autenticación del usuario
3	Adición y almacenamiento del ingreso monetario en el sistema.	Gestión de presupuesto
4	Creación de categorías de gastos personalizados del usuario.	Administración funcional
5	Creación de subcategorías de gastos personalizados del usuario.	Administración funcional
6	Adición y almacenamiento de un gasto monetario en el sistema.	Gestión de presupuesto
3	Generación de informe de ingresos versus gastos durante un período de tiempo	Generación de informes

Nota. Desarrollado por los autores del presente documento. Planeación de iteraciones del desarrollo de la aplicación.

2.3. Levantamiento de Requerimientos

En este apartado se realiza el levantamiento de los requerimientos necesarios para el diseño, desarrollo, pruebas e implementación de la aplicación web de gestión de gastos personales. Se detallarán los pasos llevados a cabo para definir los aspectos clave de la aplicación, desde la interfaz de usuario hasta la infraestructura tecnológica requerida. Estos requerimientos servirán como guía para el proceso de desarrollo y garantizarán que la aplicación cumpla con las expectativas del usuario final

2.3.1. Construcción de las Historias de Usuario:

Se definen las siguientes historias de usuario las cuales soportan los requerimientos funcionales establecidos para ser diseñados e implementados durante el transcurso del proyecto.

Tabla 3

Requerimiento desarrollo del registro de usuario

Historia de usuario 1	**Registro de usuario**
Como	Nuevo usuario
Quiero	Poder registrarme en la aplicación
Para	Poder empezar a gestionar mis gastos dentro de la aplicación a través de una cuenta personal.

Nota. Desarrollado por los autores del presente documento. Historia de usuario 1.

Tabla 4

Requerimiento desarrollo del inicio de sesión

Historia de usuario 2	**Inicio de sesión**
Como	Usuario registrado
Quiero	Poder loguearme en la aplicación
Para	Acceder a mi cuenta y ver mi presupuesto y gastos.

Nota. Desarrollado por los autores del presente documento. Historia de usuario 2.

Tabla 5

Requerimiento desarrollo del registro de ingresos

Historia de usuario 3	**Registro de ingresos económicos**
Como	Usuario de la aplicación
Quiero	Poder registrar mis ingresos, especificando el origen y el monto.
Para	Llevar un registro de mis ingresos y poder establecer presupuestos.

Nota. Desarrollado por los autores del presente documento. Historia de usuario 3.

Tabla 6

Requerimiento desarrollo del registro de categorias

Historia de usuario 4	**Gestión de categorías**
Como	usuario de la aplicación
Quiero	Poder adicionar o ver categorías y subcategorías personalizadas para mis gastos
Para	Organizar y clasificar mejor mis gastos según mis necesidades.

Nota. Desarrollado por los autores del presente documento. Historia de usuario 4.

Tabla 7

Requerimiento desarrollo del registro de subcategorías

Historia de usuario 5	**Ver resumen de gastos**
Como	Usuario de la aplicación
Quiero	Ver un resumen de mis gastos totales
Para	Tener una visión general de mis hábitos de gasto y controlar mis finanzas.

Nota. Desarrollado por los autores del presente documento. Historia de usuario 5.

Tabla 8

Requerimiento desarrollo del registro de gastos

Historia de usuario 6	**Registrar gastos**
Como	Usuario de la aplicación
Quiero	Poder registrar mis gastos, especificando la categoría y el monto.
Para	Llevar un registro detallado de mis gastos.

Nota. Desarrollado por los autores del presente documento. Historia de usuario 6.

Tabla 9

Requerimiento desarrollo de generación de informes

Historia de usuario 7	**Generar informes**
Como	Usuario de la aplicación
Quiero	Poder generar informes sobre ingresos y gastos durante un período de tiempo personalizado
Para	Analizar mis hábitos de gastos versus mis ingresos a lo largo del tiempo.

Nota. Desarrollado por los autores del presente documento. Historia de usuario 7.

2.3.2. Casos de uso

En el siguiente punto se muestran los diseños generados de los modelos de casos de uso que permiten describir la funcionalidad propuesta del sistema para la gestión de gastos personales de una forma simple y organizada, donde se representa la interacción entre el usuario y el sistema. Ver anexo A. Diagramas de Casos de Uso.

Capítulo 3. Diseño Componente de Aplicación

En este capítulo, explicaremos el diseño detallado del componente de aplicación donde presentamos la arquitectura de software del sistema, los diseños de la aplicación a nivel de datos y los diagramas de clases y secuencia. Estos elementos proporcionarán una comprensión profunda de cómo está organizado nuestro sistema y cómo interactúan sus partes.

3.1. Diseño Arquitectura

La arquitectura propuesta para la aplicación se compone de tres elementos principales: Backend, Frontend y Base de Datos.

3.1.1. Backend (Java SpringBoot):

El backend de la aplicación estará desarrollado utilizando Java con el framework SpringBoot. SpringBoot proporciona un entorno de desarrollo ágil y simplificado para la creación de aplicaciones Java basadas en Spring, lo que garantiza una alta eficiencia y escalabilidad en el desarrollo de la lógica empresarial y la gestión de solicitudes HTTP.

3.1.2. Frontend (React):

Para el frontend de la aplicación, se utilizará React, una biblioteca de JavaScript ampliamente utilizada y respaldada por una gran comunidad de desarrolladores. React ofrece un enfoque moderno y eficiente para la creación de interfaces de usuario interactivas y dinámicas. Con React, podremos construir una experiencia de usuario fluida y altamente receptiva para nuestros usuarios.

3.1.3. Base De Datos (PostgrestSQL):

La base de datos de la aplicación estará gestionada por PostgreSQL, un sistema de gestión de bases de datos relacional de código abierto. PostgreSQL ofrece una sólida seguridad, integridad de los datos y soporte para transacciones ACID, lo que lo convierte en una opción ideal para el almacenamiento de datos críticos de nuestra aplicación.

Esta arquitectura permite una separación clara de responsabilidades entre el backend y el frontend, lo que facilita el desarrollo modular y el mantenimiento del código. Además, la elección de tecnologías como SpringBoot, React y PostgreSQL garantiza un alto rendimiento, fiabilidad y escalabilidad de nuestra aplicación.

3.2. Diagramas de Secuencia

En el siguiente apartado se da una descripción detallada de la interacción de los objetos y el intercambio de mensajes en el sistema que se va a desarrollar, como primer elemento se evidencia la solicitud de información de un usuario mediante diferentes vistas de tipo web, que a su vez realizan el consumo de los diferentes microservicios que contienen la lógica para el

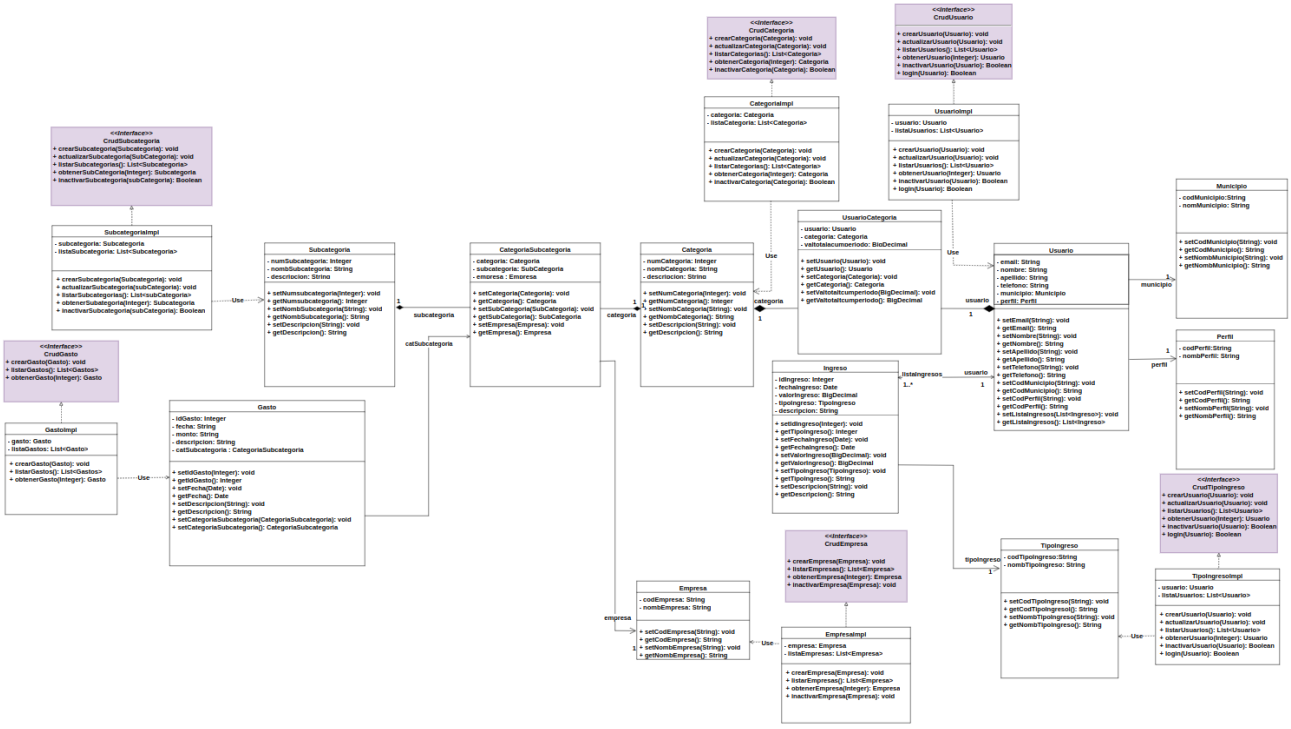
procesamiento de la información recibida por el usuario. Los diagramas desarrollados se pueden ver en el anexo B. Diagramas de secuencia.

3.3. Diagrama de Clases

El diagrama de clases es generado para incluirlo en el modelo de la aplicación, donde se crean las clases correspondientes en el paquete Models del back el cual se realiza en el Framework Spring boot. Este diagrama es obtenido en el modelamiento de los datos los cuales se realizan con un enfoque empresarial y se describen más adelante. Al ingresar el modelo de datos en el entorno de desarrollo respetando la estructura establecida y ejecutar el programa por primera vez, este realiza la creación de las tablas en la base de datos con todas las características plasmadas en el código como lo son llaves primarias, llaves foráneas y llaves compuestas. El modelo de diagrama de clases lo podemos apreciar en la figura 2.

Figura 2

Diagrama de clases de la aplicación.



Nota. Desarrollado por los autores del presente documento. Modelo de diagrama de clases.

Capítulo 4. Diseño y Construcción de la Base de Datos

En este capítulo se analiza y diseña una base de datos con enfoque empresarial que satisface la lógica de negocio y optimiza el procesamiento de información. Este análisis incluye la identificación de las misiones requeridas, definición de entidades y relaciones, y normalización de datos para asegurar integridad. El diseño se centra en eficiencia, escalabilidad y seguridad de los datos, utilizando técnicas basadas en la generación de un modelo seudomatemático seguido del modelo relacional.

4.1. Modelo de Base de datos:

4.1.1. Modelo de lógica de negocios para el diseño de la base de datos

El diseño de la capa de lógica de negocios se define a partir del enfoque empresarial para el diseño de bases de datos relacionales. En este sentido, se proponen las siguientes clases para representar el modelo de lógica de negocios del proyecto.

usuarios: Representa a los usuarios que hacen uso de la aplicación.
ingresos: Representa los ingresos asociados a los usuarios.
categorías: Representa las categorías de gastos que pueden tener los usuarios como, por ejemplo, casa (gastos asociados a la casa), carro, entre otros.
subcategorías: Representa los subtipos de gastos asociados a cada categoría. Por ejemplo, para la categoría casa se pueden tener las subcategorías servicios públicos, reparaciones, impuestos, entre otros.
gastos: Representa los gastos asociados a cada subcategoría.

4.1.2. Definición de los atributos de las clases, llaves primarias, llaves foráneas por defecto y relaciones

Partiendo de la definición de las clases del modelo de lógica de negocios del punto 1.1, se establecen las relaciones entre estas, teniendo en cuenta que la relación natural siempre va de uno a muchos y la relación inversa puede ser de uno a uno, lo que implica relaciones de tipo Dependencia Funcional Exclusiva (DFE) o de uno a muchos, lo que implica relaciones de tipo Dependencia Funcional No Exclusiva (DFNE).

Para el caso de nuestro modelo, las relaciones inversas entre las clases quedan definidas de la siguiente manera:

Tabla 10

Definición relaciones DFE y DFNE.

Clase origen	Clase destino	Relación inversa (Ri)
gastos	subcategorias	DFE
subcategorias	categorias	DFNE
categorias	usuarios	DFNE
ingresos	usuarios	DFE

Nota. Desarrollado por los autores del presente documento. Tabla relaciones.

Teniendo en cuenta que en este punto ya se tiene la definición de las clases y las relaciones que conforman nuestro modelo de lógica de negocios, se puede decir que se cumple con la Primera Forma Normal (1FN).

A partir de las clases, se asignan los atributos a cada una de estas con base en la ley de la autodeterminación. Los atributos propuestos para cada una de las clases son las siguientes:

usuarios

- idusuario PK
- documento
- nombre
- apellido
- telefono
- codiciudad FKD
- codperfil FKD

ingresos

- idingreso PK
- fechaingreso
- valoringreso
- codtipoingreso FKD
- descripcion

categorias

- numcategoria PK
- nombcategoria
- valtotacumanio
- periodoacumulado
- descripcion

subcategorias

- numsubcategoria PK
- nomsubcategoria
- codempresa FKD
- descripcion

gastos

- idgasto PK
- fecha
- monto
- descripcion

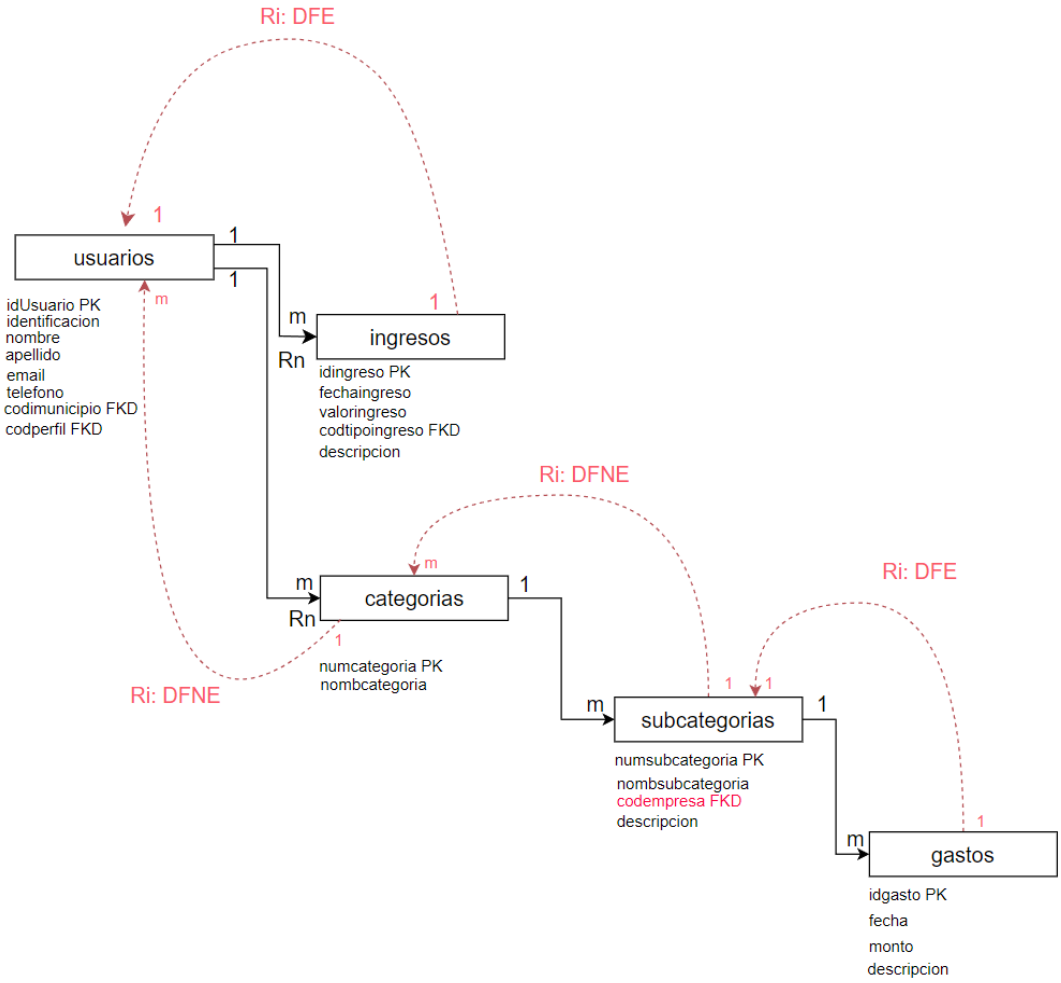
Los atributos con sufijo PK corresponden a aquellos que son llaves primarias y los que tienen FKD como sufijo son las llaves foráneas por defecto.

En este punto, se cumple con la Segunda Forma Normal (2FN), dado que se tiene la definición de los atributos pertenecientes a cada clase y además los atributos que son llaves primarias y llaves foráneas por defecto. En la figura 3 se detalla el modelo de lógica de

negocios donde se visualizan las entidades, sus atributos y los tipos de relaciones entre las entidades.

Figura 3

Modelo de lógica de negocios.



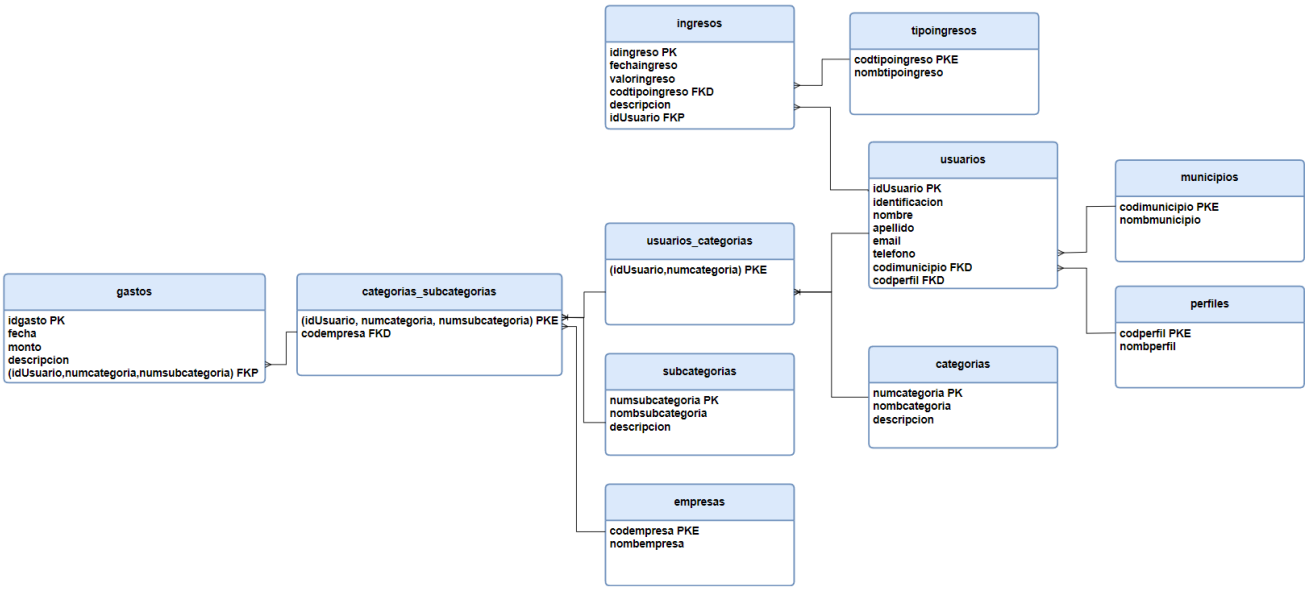
Nota. Desarrollado por los autores del presente documento. Modelo de lógica de negocios.

4.2. Modelo Relacional

El modelo relacional que se representa en la figura 4 es generado a partir del modelo de lógica de negocios y el modelo pseudomatemático. Este modelo nos proporciona una representación estructurada y detallada de los datos y sus relaciones, reflejando los procesos y reglas del negocio. Incluye la definición de tablas, atributos, claves primarias y foráneas, así como las relaciones entre estas entidades, garantizando la integridad referencial y la coherencia de los datos.

Figura 4

Modelo relacional de base de datos.



Nota. Desarrollado por los autores del presente documento. Modelo Relacional

Capítulo 5. Construcción Componente de Aplicación

En este capítulo, nos sumergimos en el proceso de construcción del componente de aplicación de nuestro proyecto, siguiendo las iteraciones generadas por el modelo evolutivo de desarrollo de software. Aquí, presentamos los avances y mejoras realizadas en cada etapa del desarrollo, reflejando el progreso incremental y continuo de nuestra aplicación. Desde la implementación de funcionalidades básicas hasta la incorporación de características más complejas, cada iteración representa un paso hacia adelante en la evolución de nuestro producto. A través de este capítulo, proporcionamos una visión detallada de cómo nuestro proyecto cobra vida a medida que avanzamos en su construcción, demostrando el valor y la efectividad de adoptar un enfoque iterativo en el desarrollo de software.

5.1. Generación Código Automático CRUDS

Para la generación de código automático se realiza un análisis del modelo de datos, junto con el código necesario para la construcción del servicio de cada entidad, donde se determina que para cada objeto se requiere las clases que representan el modelo, el repositorio, el controlador y por último el servicio. Para la entrada de datos del algoritmo de generación de código automático encontramos que los datos más importantes son el nombre de la entidad y una lista compuesta de el nombre y tipo de cada atributo. Teniendo como restricción que solo se pueden crear atributos con tipos de datos nativos.

La herramienta en la cual nos apoyamos para la generación de código automático es una librería llamada Apache Velocity la cual es un motor de plantillas para código java, permitiéndonos modelar una plantilla para cada una de las clases que se requiere generar para la creación del CRUD.

5.9.1. Generación Modelo de datos

Teniendo como restricción que solo se pueden crear atributos con tipos de datos nativos se crea una plantilla capaz de crear la entidad de forma general, esta plantilla carga las librerías necesarias, crea la etiqueta @Entity para asignar la clase creada a un modelo de datos que posteriormente va a ser creado de forma automática en la base de datos, crea los atributos con sus respectivos tipos de datos nativos, crea los constructores necesarios y por último los getters y Setter de las variables. En la figura 5 se puede observar el modelo para los datos.

Figura 5

Plantilla de generación de código Model.vm.

Model.vm

```
package ${paquete};

#foreach($libreria in $libreriasBasicas)
import $libreria;
#end

@Entity
public class ${entidad} {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    #foreach($atributo in $atributos)
    private ${atributo.tipo} ${atributo.nombre};
    #end

    public ${entidad}(#set($firstAttribute = true)#foreach($atributo in $atributos)${atribut
    #foreach($atributo in $atributos)
        this.${atributo.nombre} = ${atributo.nombre};
    #end
    }

    public ${entidad}() {
        super();
    }

    #foreach($atributo in $atributos)
    #set($firstChar = $atributo.nombre.substring(0, 1).toUpperCase())
    #set($restOfWord = $atributo.nombre.substring(1))
    public ${atributo.tipo} get${firstChar}${restOfWord}() {
        return this.${atributo.nombre};
    }

    public void set${firstChar}${restOfWord}(${atributo.tipo} ${atributo.nombre}) {
        this.${atributo.nombre} = ${atributo.nombre};
    }
    #end
}
```

Nota. Desarrollado por los autores del presente documento. Modelo Model.VM.

La figura 6 nos muestra un ejemplo de la clase Perfil creada a partir de la plantilla Model.vm donde se puede observar que tiene una estructura adecuada.

Figura 6

Clase generada a partir de la plantilla Model.vm.

Perfil.java

```java
package com.gestiongastos.models;

import jakarta.persistence.Entity;

@Entity
public class Perfil {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int idPerfil;
    private String nombre;
    private int privilegio;

    public Perfil(int idPerfil,String nombre,int privilegio){
        this.idPerfil = idPerfil;
        this.nombre = nombre;
        this.privilegio = privilegio;
    }

    public Perfil() {
        super();
    }

    public int getIdPerfil() {
        return this.idPerfil;
    }

    public void setIdPerfil(int idPerfil) {
        this.idPerfil = idPerfil;
    }
    public String getNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getPrivilegio() {
        return this.privilegio;
    }

    public void setPrivilegio(int privilegio) {
        this.privilegio = privilegio;
    }
}
```

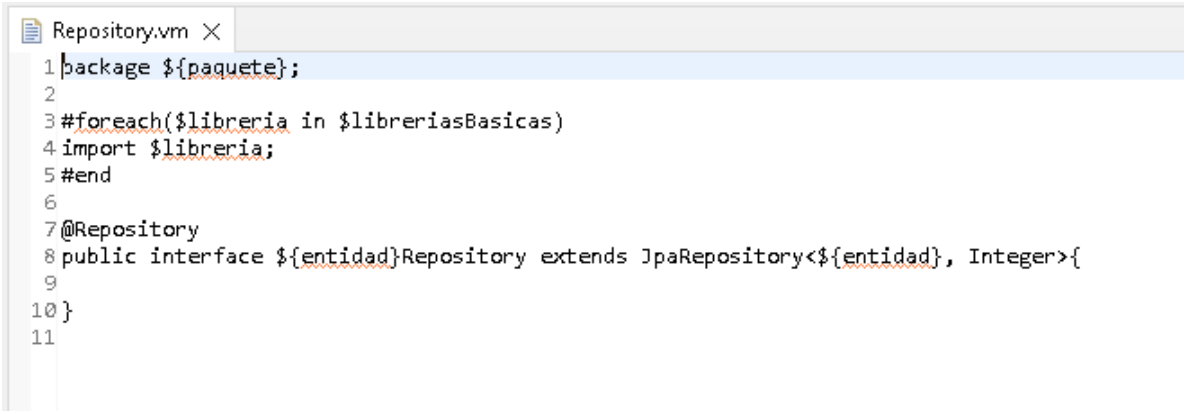
Nota. Desarrollado por los autores del presente documento. Clase generada a partir Model.vm.

5.9.2. Generación Repositorio

Para la creación del repositorio se crea una plantilla que contiene las librerías, una etiqueta @Repository, el nombre de la clase y un extends a la clase JpaRepository la cual contiene todos los métodos necesarios para la gestión del CRUD en la base de datos. Se puede observar el modelo de la generación de la clase repositorio en figura 7.

Figura 7

Plantilla de generación de código Repository.vm.

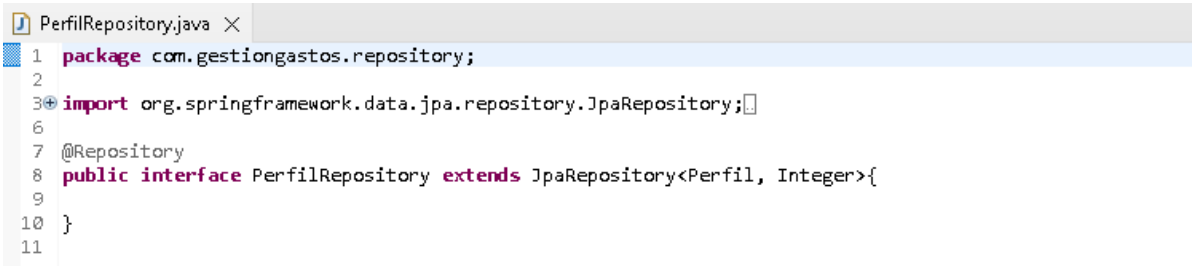


Nota. Desarrollado por los autores del presente documento. Modelo Repository.vm

A continuación, se muestra en la figura 8 un ejemplo de la clase PerfilRepository generada a partir de la plantilla Repository.vm.

Figura 8

Clase generada a partir de la plantilla Repository.vm.



Nota. Desarrollado por los autores del presente documento. Clase generada de Repositopry.vm

5.9.3. Generación Servicio

Para la creación del servicio se tiene en cuenta que esta clase va a contener todas las operaciones del CRUD, estas operaciones van a ser llamadas desde un objeto de la clase repositorio, para lograr esto se crea una plantilla que contiene las librerías, una etiqueta de tipo @Service, el nombre de la clase, un objeto de tipo Repository que contiene los métodos necesarios para las operaciones requeridas, y los métodos del CRUD los cuales son crear, actualizar, listar, obtener y eliminar. La figura 9 contiene la lógica para la generación de la clase servicio.

Figura 9

Plantilla de generación de código Service.vm.

Service.vm

```
package ${paquete};

#foreach($libreria in $libreriasBasicas)
import $libreria;
#end

@Service
public class ${entidad}Service {

    @Autowired
    private ${entidad}Repository ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Repository;

    // Crear ${entidad}
    public ${entidad} crear${entidad}(${entidad} ${entidad.toLowerCase()}) {
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Repository.save(${entidad.toLowerCase()});
    }

    // Listar ${entidad}
    public List<${entidad}> listar${entidad}() {
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Repository.findAll();
    }

    // Obtener ${entidad} por su ID
    public Optional<${entidad}> obtener${entidad}ById(int id) {
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Repository.findById(id);
    }

    // Actualizar ${entidad}
    public ${entidad} actualizar${entidad}(${entidad} ${entidad.toLowerCase()}) {
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Repository.save(${entidad.toLowerCase()});
    }

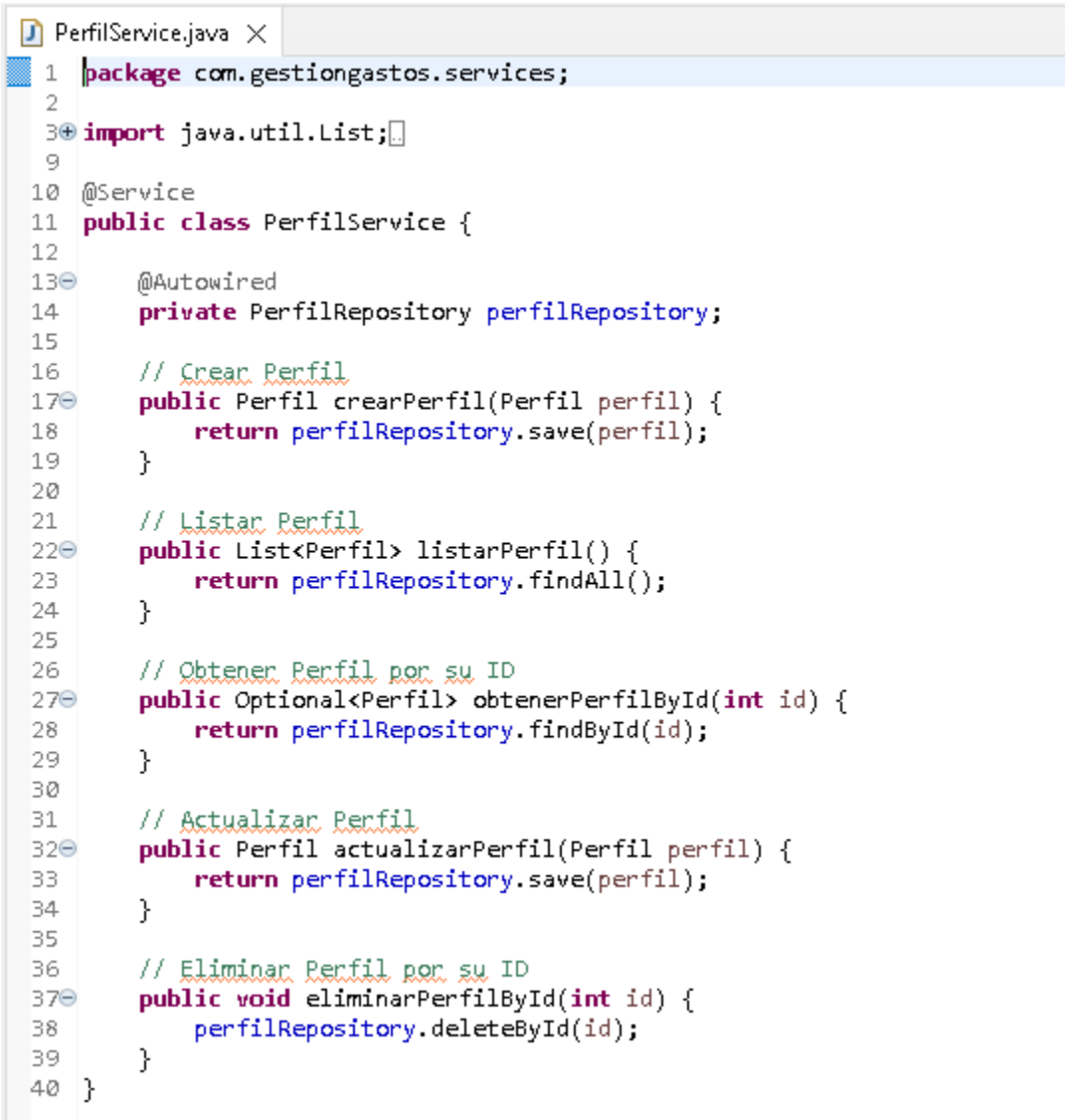
    // Eliminar ${entidad} por su ID
    public void eliminar${entidad}ById(int id) {
        ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Repository.deleteById(id);
    }
}
```

Nota. Desarrollado por los autores del presente documento. Modelo Service.vm

A continuación, se muestra un ejemplo de la clase PerfilService generada a partir de la plantilla Repository.vm. Como se puede observar se realiza la generación automática de la clase en la figura 10.

Figura 10

Clase generada a partir de la plantilla Model.vm.



PerfilService.java

```java
package com.gestiongastos.services;

import java.util.List;

@Service
public class PerfilService {

    @Autowired
    private PerfilRepository perfilRepository;

    // Crear Perfil
    public Perfil crearPerfil(Perfil perfil) {
        return perfilRepository.save(perfil);
    }

    // Listar Perfil
    public List<Perfil> listarPerfil() {
        return perfilRepository.findAll();
    }

    // Obtener Perfil por su ID
    public Optional<Perfil> obtenerPerfilById(int id) {
        return perfilRepository.findById(id);
    }

    // Actualizar Perfil
    public Perfil actualizarPerfil(Perfil perfil) {
        return perfilRepository.save(perfil);
    }

    // Eliminar Perfil por su ID
    public void eliminarPerfilById(int id) {
        perfilRepository.deleteById(id);
    }
}
```

Nota. Desarrollado por los autores del presente documento. Clase generada de Service.vm.

5.9.4. Generación Controlador

Para la creación del controlador se requiere crear el complemento del encabezado html que hace el llamado al servicio, así mismo, se hace el llamado de la clase Service, para crear la plantilla del controlador se crean las librerías necesarias, una etiqueta de tipo @Controller, el nombre de la clase, un objeto de tipo Service que contiene los métodos para las operaciones requeridas, y por último se crean los métodos del CRUD los cuales son crear, actualizar, listar, obtener y eliminar. Este modelo lo podemos visualizar en la figura 11.

Figura 11

Plantilla de generación de código Controller.vm.

Controller.vm

```
package ${paquete};

#foreach($libreria in $libreriasBasicas)
import $libreria;
#end


@RestController
@RequestMapping("/${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}")
public class ${entidad}Controller {

    @Autowired
    private ${entidad}Service ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Service;

    //Crear ${entidad}
    @PostMapping("")
    public ${entidad} crear${entidad}(@RequestBody ${entidad} ${entidad.substring(0, 1).toLowerCase()}${entid
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Service.crear${entidad}(${entid
    }

    //Listar ${entidad}
    @GetMapping("")
    public List<${entidad}> getAll${entidad}() {
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Service.listar${entidad}();
    }

    //Obtener ${entidad} por su ID
    @GetMapping("/{id}")
    public Optional<${entidad}> obtener${entidad}ById(@PathVariable int id) {
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Service.obtener${entidad}ById(i
    }

    //Actualizar ${entidad}
    @PutMapping("/{id}")
    public ${entidad} actualizar${entidad}(@PathVariable int id, @RequestBody ${entidad} ${entidad.substring(
        ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}.setId${entidad}(id);
        return ${entidad.substring(0, 1).toLowerCase()}${entidad.substring(1)}Service.actualizar${entidad}(${
    }

```

Nota. Desarrollado por los autores del presente documento. Modelo Controller.vm

La figura 12 nos proporciona un ejemplo de la clase PerfilController generada a partir de la plantilla Controller.vm.

Figura 12

Clase generada a partir de la plantilla Controller.vm.

PerfilController.java

```java
package com.gestiongastos.controllers;

import java.util.List;


@RestController
@RequestMapping("/perfil")
public class PerfilController {

    @Autowired
    private PerfilService perfilService;

    //Crear Perfil
    @PostMapping("")
    public Perfil crearPerfil(@RequestBody Perfil perfil) {
        return perfilService.crearPerfil(perfil);
    }

    //Listar Perfil
    @GetMapping("")
    public List<Perfil> getAllPerfil() {
        return perfilService.listarPerfil();
    }

    //Obtener Perfil por su ID
    @GetMapping("/{id}")
    public Optional<Perfil> obtenerPerfilById(@PathVariable int id) {
        return perfilService.obtenerPerfilById(id);
    }

    //Actualizar Perfil
    @PutMapping("/{id}")
    public Perfil actualizarPerfil(@PathVariable int id, @RequestBody Perfil perfil) {
        perfil.setIdPerfil(id);
        return perfilService.actualizarPerfil(perfil);
    }

    //Eliminar Perfil por su ID
    @DeleteMapping("/{id}")
    public void eliminarPerfilById(@PathVariable int id) {
        perfilService.eliminarPerfilById(id);
    }
}

```

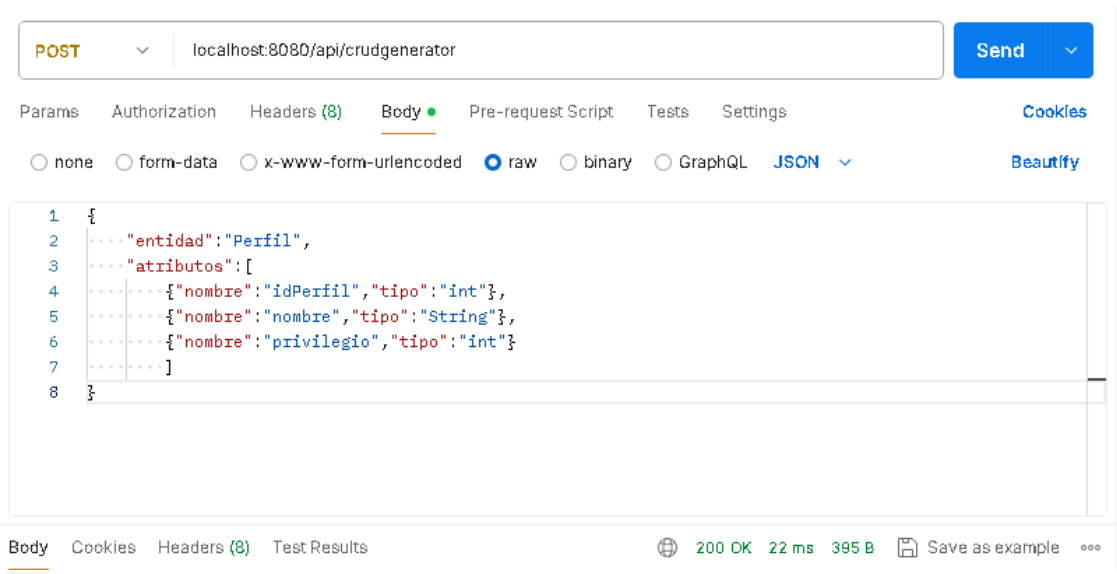
Nota. Desarrollado por los autores del presente documento. Clase generada de Controller.vm

5.9.4. Creación Servicio y Prueba de los CRUDS

para la generación del servicio del perfil de un usuario se crea un objeto de tipo JSON que contiene los atributos “entidad” donde se coloca el nombre de la entidad, que para nuestro caso es Perfil, y atributo que contiene una lista de datos que representan el nombre del atributo y el tipo del atributo. Para la ejecución se crea un servicio back el cual se ejecuta y crea de forma automática las clases correspondientes. En la figura 13 se muestra la ejecución del servicio de generación de código automático para la creación del crud del sistema para los perfiles del usuario.

Figura 13

Creación crud para la clase perfil en postman.

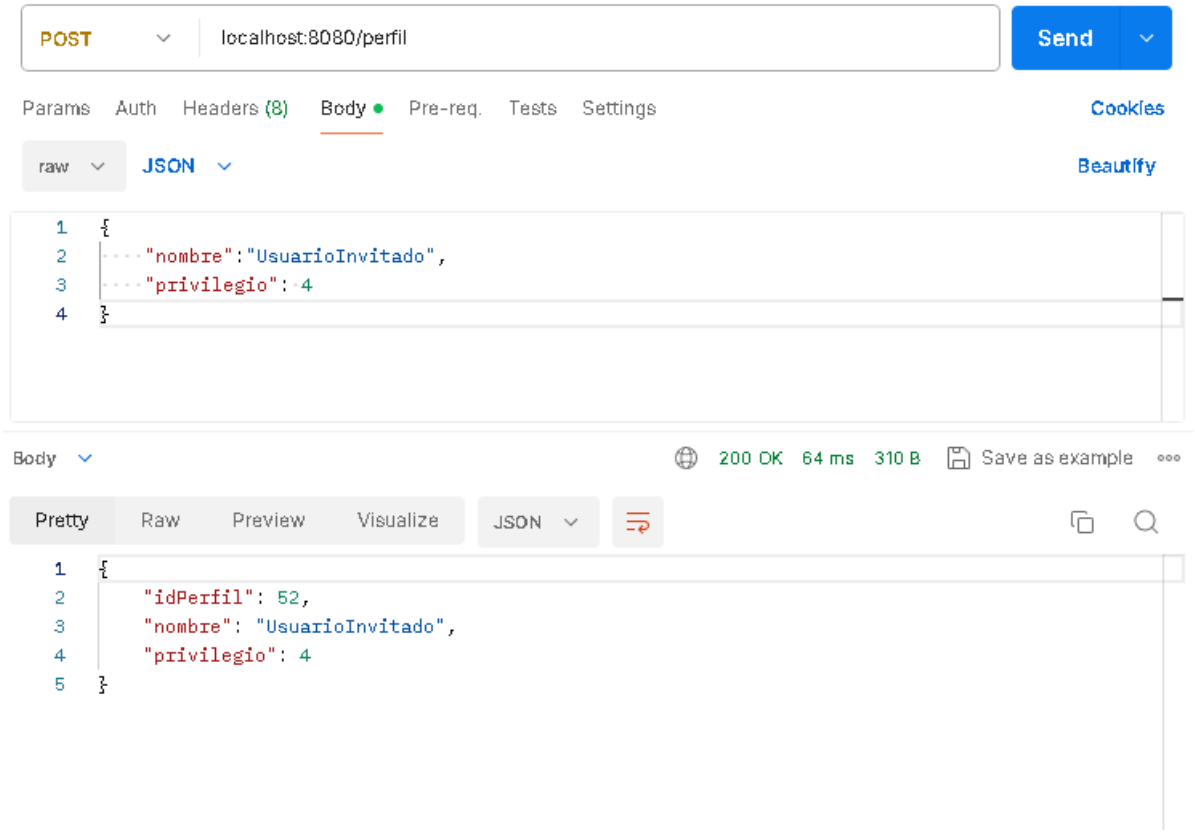


Nota. Desarrollado por los autores del presente documento. Generación crud Perfil.

Como se observa en la figura 14 el método Post nos permite guardar la información correspondiente de la entidad creada, en la siguiente prueba ingresamos los datos del perfil en formato Json y nos da como resultado la entidad con su respectivo id.

Figura 14

Creación perfil de usuario método post en postman.

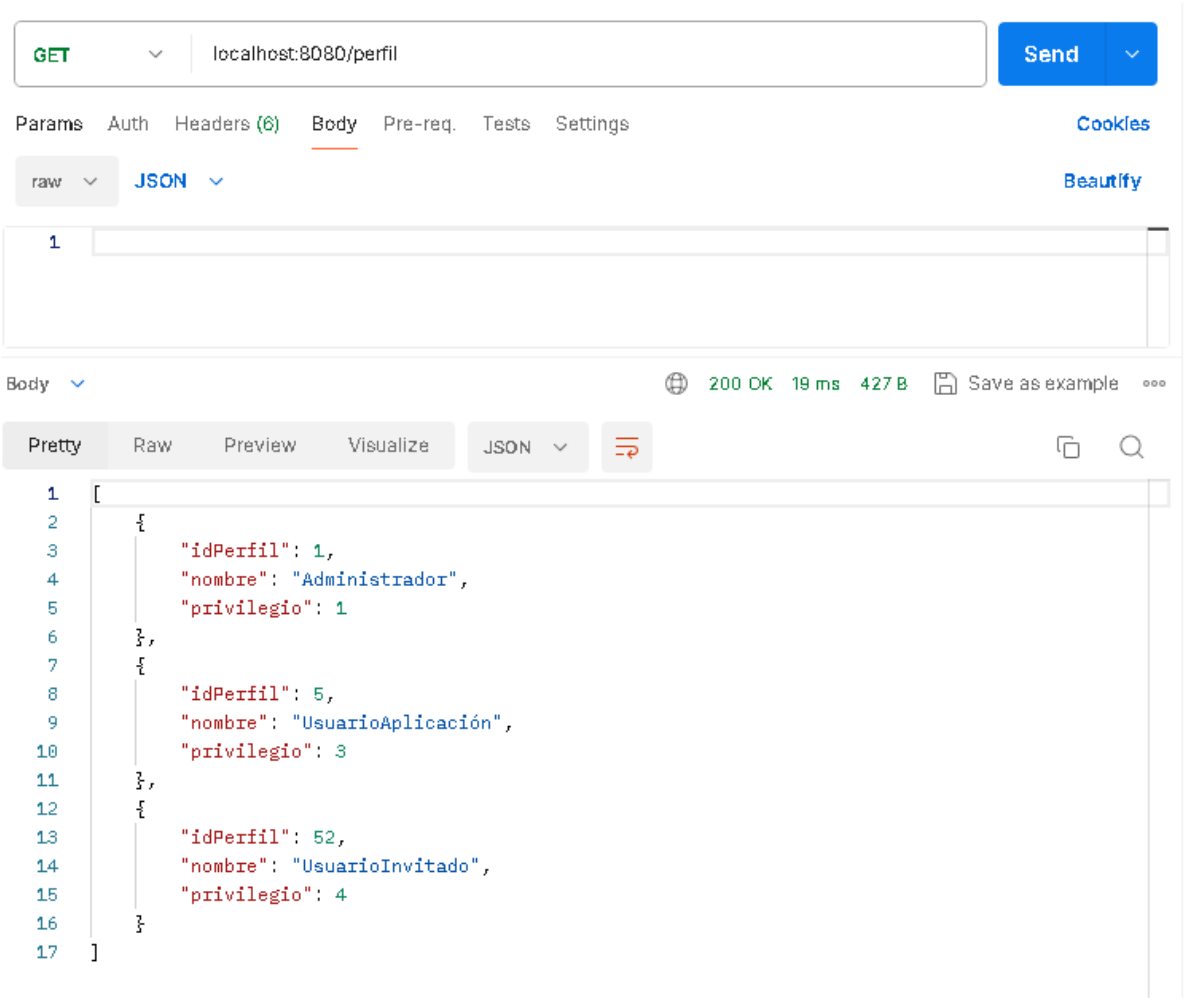


Nota. Desarrollado por los autores del presente documento. Metodo Post

El método Get nos permite listar todos los datos de la entidad que se está trabajando, como se muestra en la figura 15, se guardan varios datos y se ejecuta el método get en donde se devuelve un listado en formato Json de la información almacenada.

Figura 15

Obtener todos los perfiles de usuarios método get en postman.

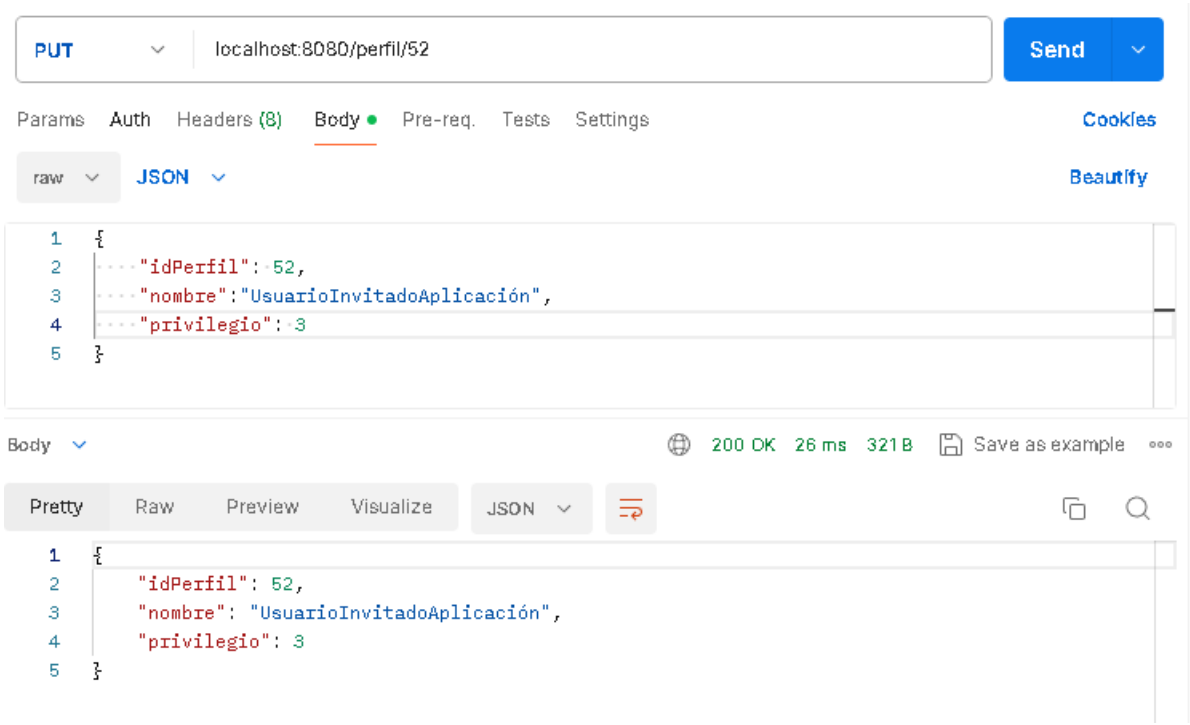


Nota. Desarrollado por los autores del presente documento. Metodo Get

El método Put nos permite modificar cualquier atributo de la tabla diferente a la llave primaria, para esto debemos de ingresar el id de la entidad e ingresar los nuevos datos. La figura 16 ilustra como es posible la actualización.

Figura 16

Actualizar perfil método put en postman.

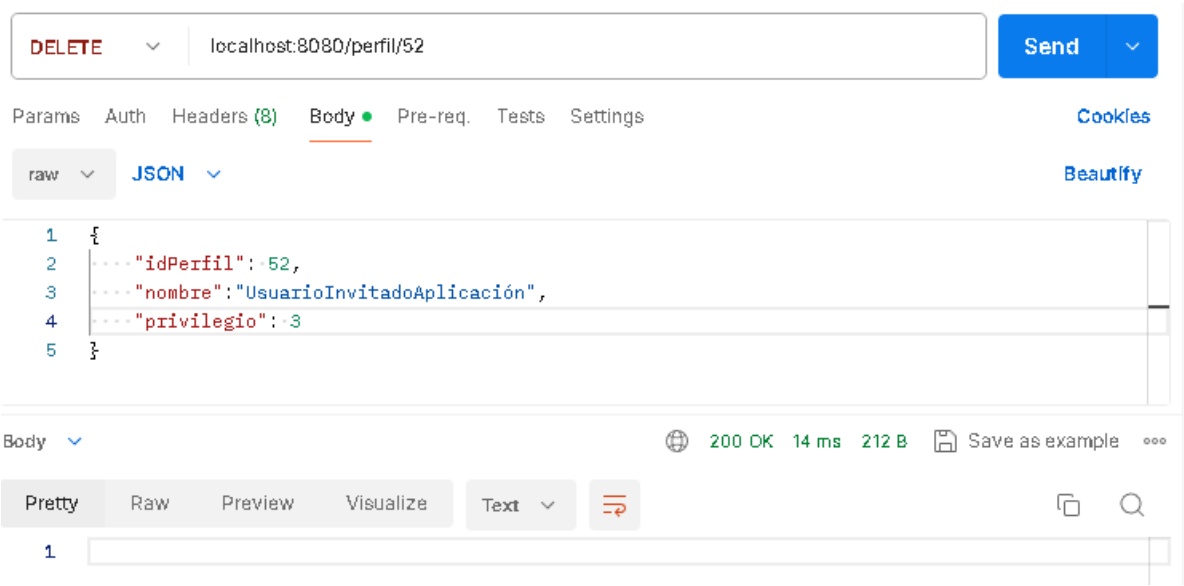


Nota. Desarrollado por los autores del presente documento. Metodo Put

En la figura 17 se muestra el método Delete el cual nos permite eliminar un dato de la tabla ingresando la llave primaria del dato que se requiere eliminar.

Figura 17

Eliminar perfil método delete en postman.



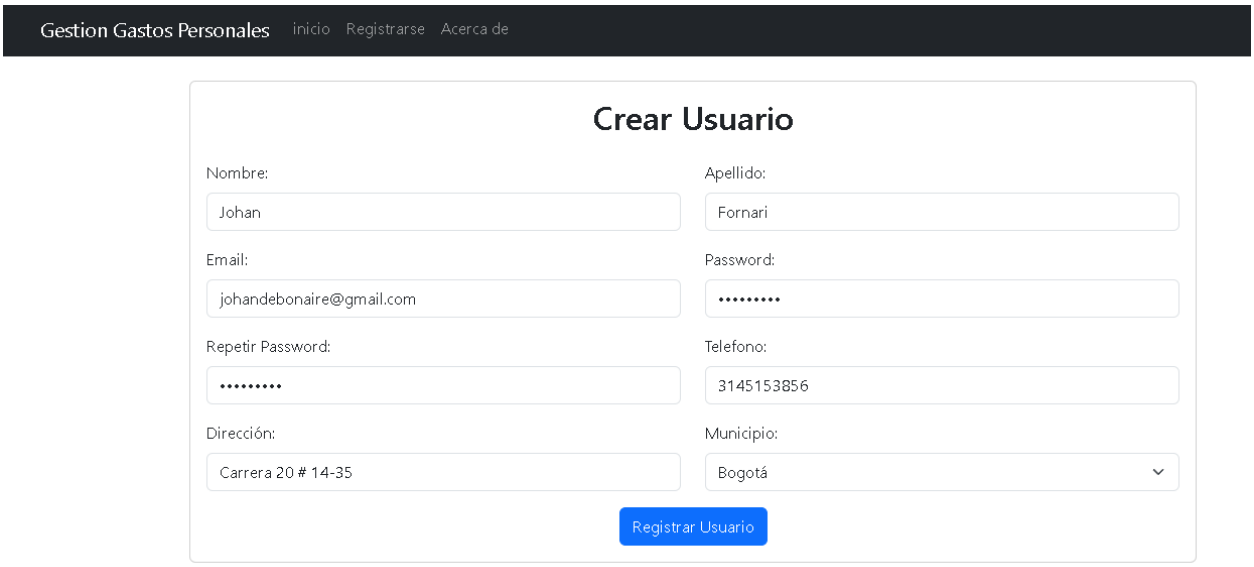
Nota. Desarrollado por los autores del presente documento. Metodo Delete

5.2. Registro Usuario

El registro de los usuarios de la aplicación es el primer paso que debemos realizar si queremos empezar a realizar la gestión de gastos en la aplicación, para realizar el registro nos dirigimos en la parte superior en el botón Registrarse, los datos son almacenados según el modelo de datos establecido, en términos de seguridad utilizamos la librería Argon2 para almacenar la contraseña del usuario en un hash que encripta este dato.

Figura 18

Crear usuario en la aplicación.



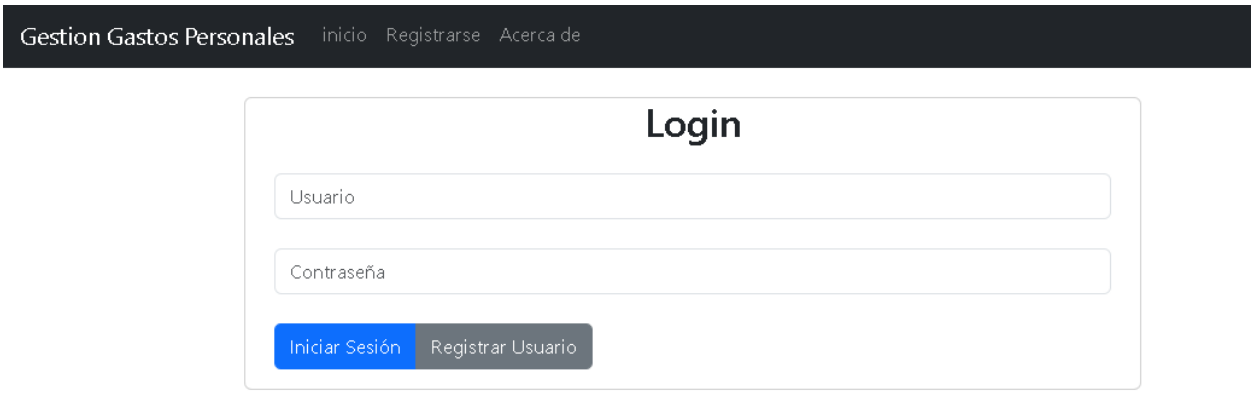
Nota. Desarrollado por los autores del presente documento. Crear usuario.

5.3. Autenticación Usuario

Al iniciar la aplicación, podemos ingresar a nuestro usuario si ya lo tenemos creado o también tenemos la posibilidad de crear un usuario nuevo dando clic en el botón gris llamado Registrar Usuario. Para el inicio de sesión debemos ingresar con la dirección del correo registrado en la aplicación y la contraseña.

Figura 19

Crear usuario en la aplicación.

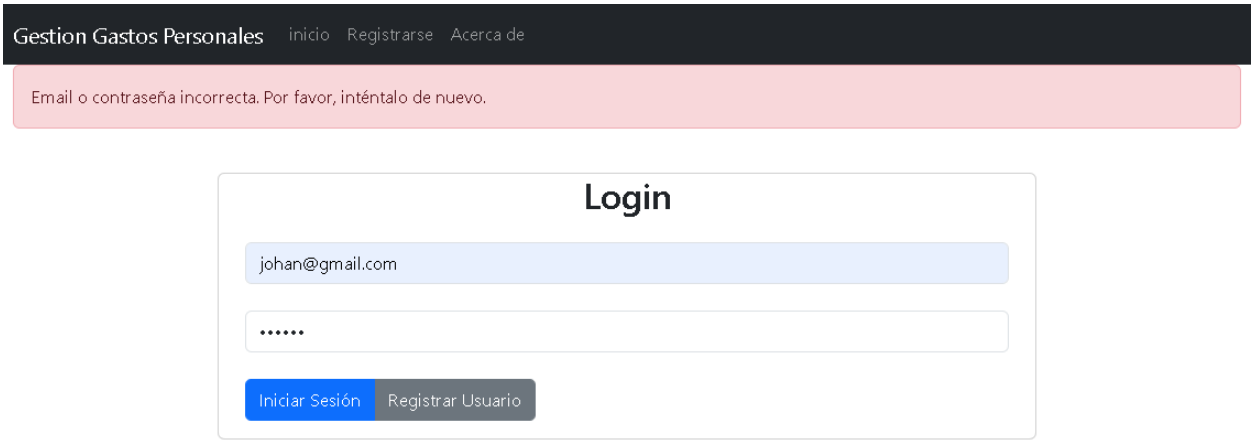


Nota. Desarrollado por los autores del presente documento. Login.

En caso de que alguno de los datos ingresados no corresponda a la hora de iniciar sesión con nuestro correo registrado y contraseña nos aparecerá la siguiente imagen correspondiente a la figura 20.

Figura 20

Login Fallido aplicación.



Nota. Desarrollado por los autores del presente documento. Login fallido

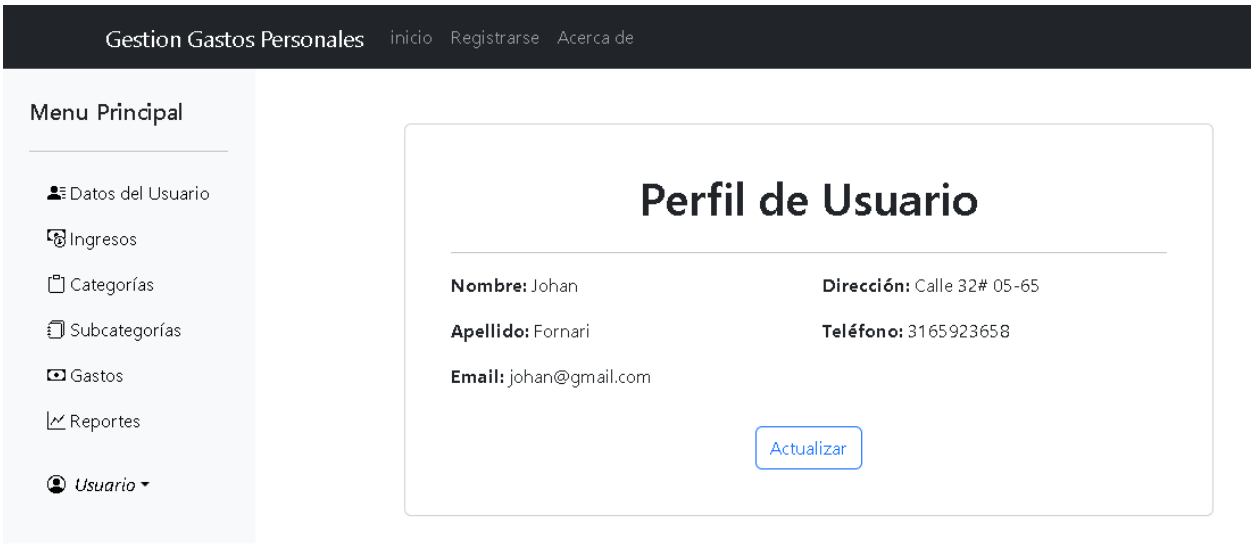
5.4. Ingreso Usuario

Al ingresar a la aplicación con un usuario registrado, tenemos la posibilidad de acceder a diferentes funcionalidades del menú lateral el cual contiene todos los datos del usuario, como lo son los ingresos, las categorías, las subcategorías, los gastos y un apartado de reportes.

El menú de datos de usuario nos muestra un detalle de los datos del usuario y un botón de actualizar en caso de que queramos modificar algún dato, como podemos observar en la figura 21.

Figura 21

Datos de usuario aplicación.



Nota. Desarrollado por los autores del presente documento. Perfil del usuario

5.5. Actualizar Datos Usuario

Para la actualización de datos del usuario se consideran 2 atributos que consideramos que se pueden modificar respetando la integridad de los datos del usuario, para este caso modificamos la dirección de residencia del usuario y el teléfono, tal como se muestra en la figura 22.

Figura 22

Actualizar usuario en la aplicación.



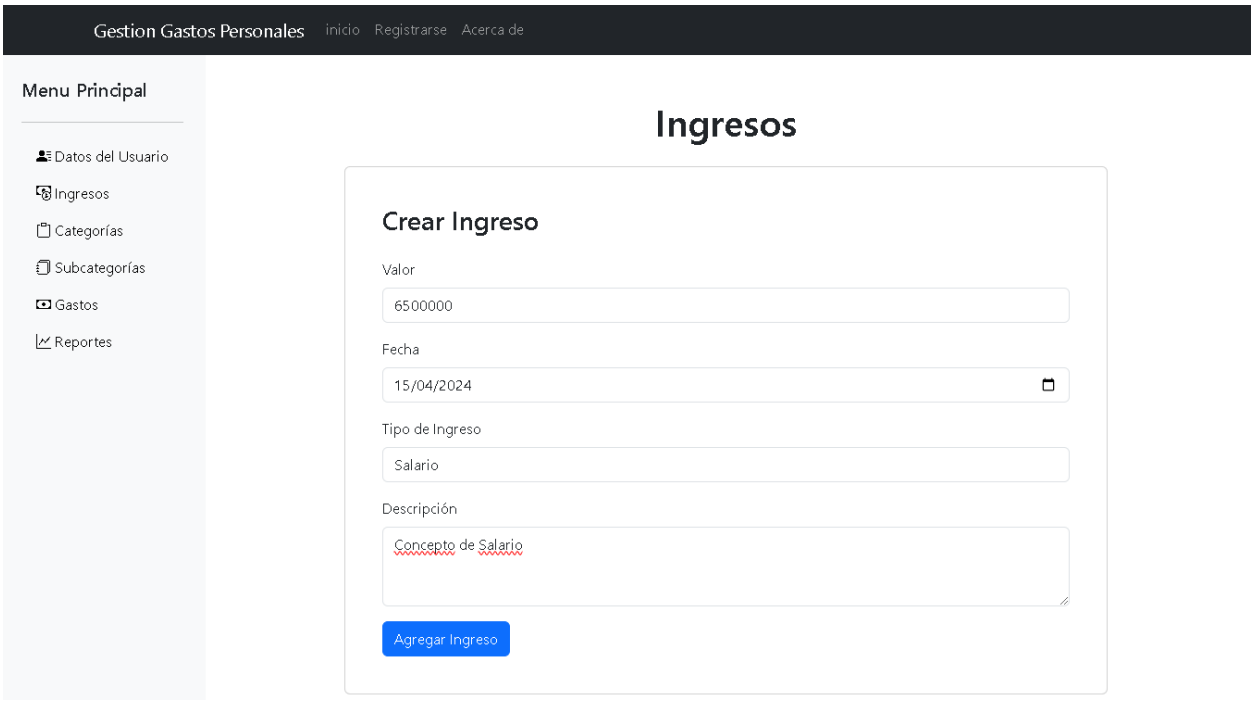
Nota. Desarrollado por los autores del presente documento. Actualizar Usuario

5.6. Registro de Ingresos Aplicación

El registro de ingresos del usuario lo realizamos ingresando en el menú de Ingresos en donde hay dos apartados, uno que corresponde al registro de los ingresos del usuario y una lista de ingresos que el usuario ha registrado. Para registrar un nuevo ingreso debemos agregar el valor, la fecha, el tipo de ingreso y por último una descripción de este, como se muestra a continuación en la figura 23.

Figura 23

Crear ingresos en la aplicación.



Nota. Desarrollado por los autores del presente documento. Registro Ingresos

En el segundo apartado vemos en detalle todos los ingresos del usuario incluyendo el valor, la fecha, el tipo de Ingreso y por último se muestra la descripción, este detalle se visualiza en la figura 24.

Figura 24

Ingresos del usuario en la aplicación.

Valor	Fecha	Tipo de Ingreso	Descripción
$ 5.000.000,00	2024-04-16	Arriendo	5m
$ 65.822.000,00	2024-04-17	Otros	venta carro
$ 600.500,00	2024-03-19	Otros	Concepto de venta articulo deportivo
$ 96.000,00	2024-03-05	Salario	3500000
$ 950.000,00	2024-01-08	Otros	Venta Bicicleta
$ 50.000.000,00	2024-05-03	Otros	Venta de Carro
$ 856.000,00	2024-03-01	Otros	Venta de una figura de colección
$ 6.000.000,00	2024-05-14	Salario	Salario

Nota. Desarrollado por los autores del presente documento. Tabla ingresos aplicación

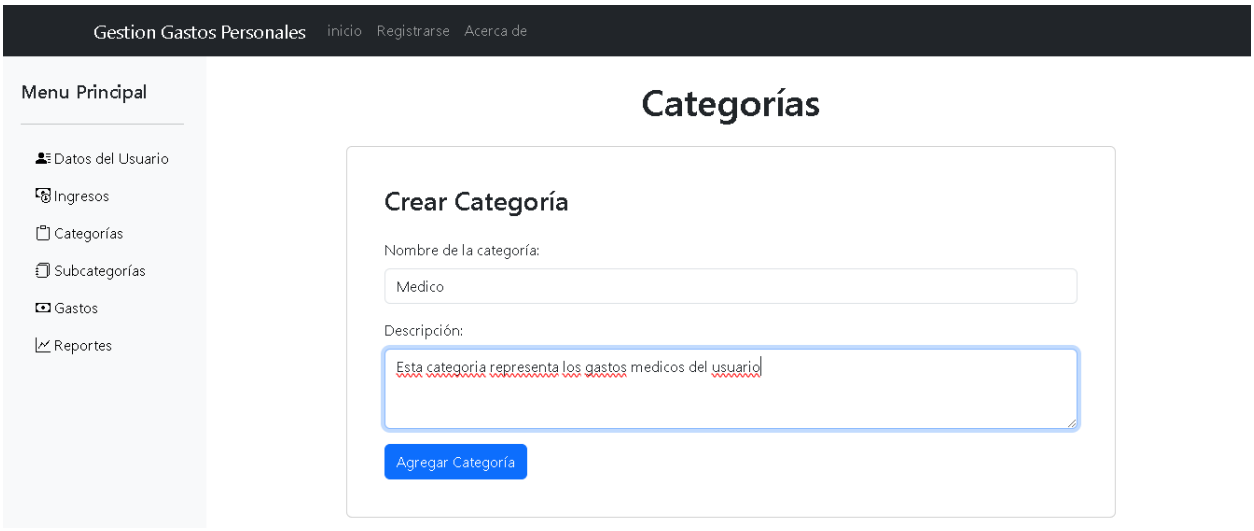
5.6. Registro de Categorías Aplicación

Las categorías sirven para crear un apartado que permite separar los gastos del usuario de una forma personalizada, ofreciendo la posibilidad de clasificar los gastos en categorías generales o específicas. Esto ayuda a organizar y visualizar mejor las finanzas de las personas que usen la aplicación.

Para la creación de una categoría nueva, se ingresa el nombre de la categoría y la descripción correspondiente. Esto facilita una referencia clara y detallada de cada tipo de gasto, proporcionando una visión estructurada de los hábitos financieros del usuario y ayudando a la toma de deciciones.

Figura 25

Registro de categorias en la aplicación.



Nota. Desarrollado por los autores del presente documento. Crear categorías.

Las categorías creadas se pueden observar en la figura 26, donde se detallan el nombre de la categoría y su descripción. Esta información se presenta en la parte inferior con un formato de tabla, facilitando una visualización clara y ordenada de las diferentes categorías y sus respectivas descripciones. Esto permite al usuario tener un control más eficiente y organizado de sus gastos, asegurando una gestión financiera más efectiva.

Figura 26

Categorias del usuario en la aplicación.

Nombre de la Categoría	Descripción
Vivienda	Concepto de vivienda
Vehiculo	Gastos asociados al vehiculo
Medico	Esta categoria representa los gastos medicos del usuario
Personal	Esta categoria se refiere a los gastos personales del usuario.
Mascota	mascota
Salidas	Gastos de salidas
Finca	Gastos de la finca
Viajes	viajes
Tecnología	Gastos Tencologicos

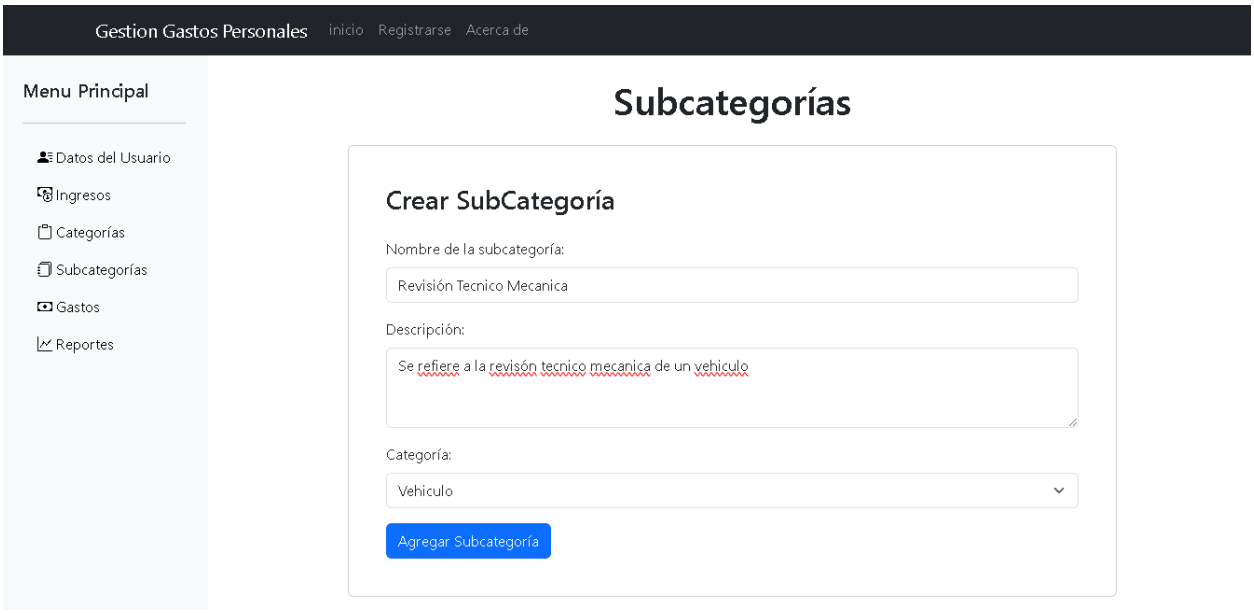
Nota. Desarrollado por los autores del presente documento. Tabla categorías aplicación

5.7. Registro de Subcategorías Aplicación

Las subcategorías son tipos de gastos más específicos relacionados con una categoría principal, lo que permite al usuario segmentar los gastos de una manera más detallada y realizar un control y seguimiento más preciso. Para la creación de una subcategoría, se ingresan el nombre de la subcategoría, la descripción correspondiente, y la categoría a la cual pertenece. Estas subcategorías se muestran en la figura 27, donde se puede apreciar la relación entre cada subcategoría y su categoría principal, así como las descripciones asociadas, todo presentado en un formato de tabla para facilitar su comprensión y manejo.

Figura 27

Registro de subcategorías en la aplicación.



Nota. Desarrollado por los autores del presente documento. Registro subcategorías

Para validar que la creación de subcategorías se realiza de forma correcta, se agrega una tabla que contiene los datos de todas las subcategorías del usuario. Esta tabla detalla el nombre, la descripción y la categoría asociada a cada subcategoría, lo que facilita el control y la segmentación de los gastos. La información se presenta en la parte inferior del panel, como se observa en la imagen 28, permitiendo al usuario verificar y gestionar sus subcategorías de manera eficiente. La implementación de categorías y subcategorías en el modelo propuesto permite una mejor organización de los gastos, lo que a su vez facilita un control más efectivo de las finanzas del usuario. Esta estructura organizada ayuda a segmentar y rastrear los gastos de manera más precisa, proporcionando una visión clara y detallada del uso del dinero.

Figura 28

Registros de subcategorías del usuario en la aplicación.

Nombre de la Subcategoría	Descripción	Categoría
Recibo Gas	Recibo Gas	Vivienda
Recibo Agua	agua	Vivienda
Recibo Luz	Recibo Luz	Vivienda
Compra Vehiculo	Compra de cualquier tipo de vehiculo	Vehiculo
Revisión Tecnico Mecanica	Se refiere a la revisón tecnico mecanica de un vehiculo	Vehiculo
Bonos	Este concepto representa los bonos que se requieren para acceder a servicios medicos de la EPS.	Medico
Restaurantes	Gastos relacionados con salidas a restaurantes y bares	Personal

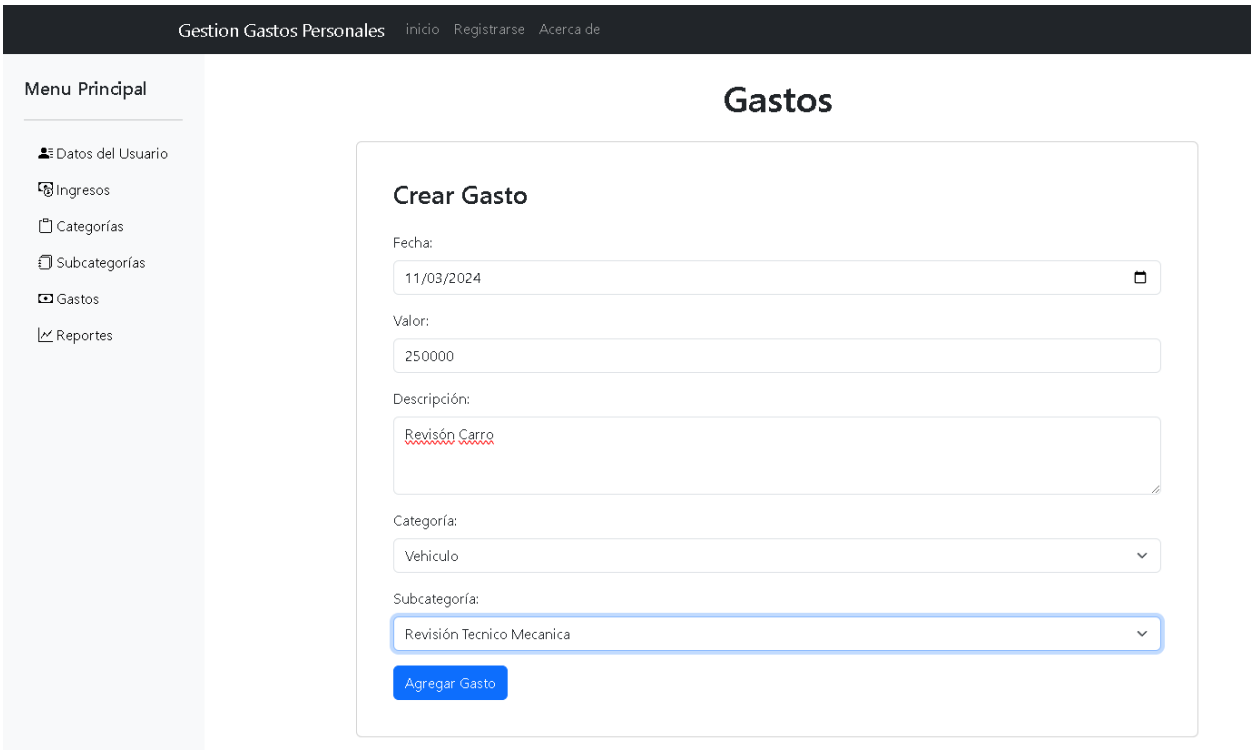
Nota. Desarrollado por los autores del presente documento. Tabla subcategorías Aplicación

5.8. Registro de Gastos Aplicación

Los gastos que realiza el usuario se registran en el panel de Gastos, donde se registran la fecha, el valor, la descripción, la categoría y la subcategoría correspondiente. Un ejemplo del registro de un gasto de un usuario se muestra en la figura número 29. Esta información detallada permite una gestión más organizada y precisa de los gastos personales del usuario. Al incluir estos campos, el usuario puede categorizar y subcategorizar cada gasto, lo que facilita el seguimiento y análisis de sus finanzas. Además, tener registros completos y organizados mejora la capacidad del usuario para identificar patrones de gasto, ajustar su presupuesto y tomar decisiones financieras más informadas.

Figura 29

Registro de Gastos en la aplicación.



Nota. Desarrollado por los autores del presente documento. Registro gastos

Como podemos observar en la figura 30 se genera la información de los gastos en una tabla que se encuentra en la parte inferior con el compendio de datos de los gastos registrados en la aplicación para el usuario. Los datos que se presentan en la tabla incluyen la fecha, el valor, la descripción, la categoría y la subcategoría. Esta tabla permite una visualización clara y estructurada de los gastos del usuario, facilitando la revisión y el análisis detallado de cada transacción registrada.

Figura 30

Registros de gastos del usuario en la aplicación.

Fecha	Valor	Descripción	Categoría	Subcategoría
2024-04-19	$ 5.800.000,00	Pago de recibo	Vivienda	Recibo Luz
2024-04-19	$ 54.552.000,00	Recibo	Vivienda	Recibo Gas
2024-04-25	$ 95.000,00	Pago bono para cita medica y reclamar medicamentos	Medico	Bonos
2024-03-11	$ 250.000,00	Revisón Carro	Vehiculo	Revisión Tecnico Mecanica
2024-02-19	$ 35.000,00	Almuerzo	Personal	Restaurantes

Nota. Desarrollado por los autores del presente documento. Tabla de gastos.

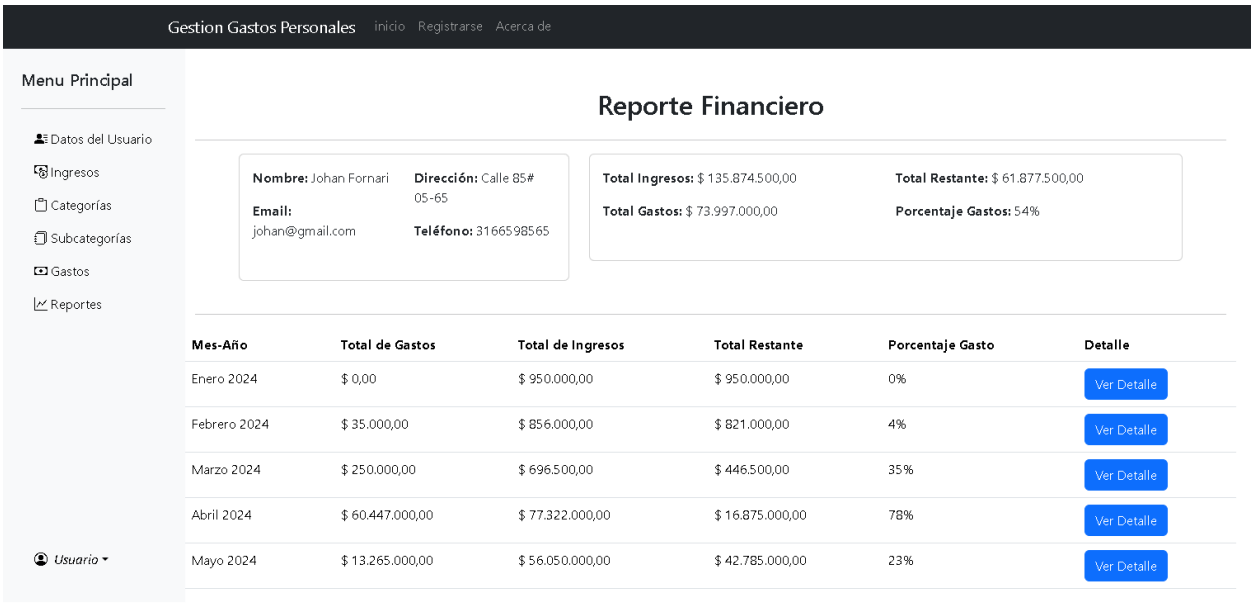
5.10. Generación de Reportes

Para el desarrollo de los reportes, se obtiene la información del usuario y se coloca en un panel en la parte superior izquierda. Esta información incluye detalles relevantes como el nombre, la dirección y el teléfono del usuario, proporcionando un contexto claro y accesible. Luego, mediante la implementación de un query personalizado en la base de datos aplicado al backend de la aplicación, se obtiene un balance de los ingresos y gastos que ha tenido el usuario a lo largo del tiempo. Este balance general permite al usuario tener una visión clara de su situación financiera global.

Adicionalmente, se realiza una segmentación mensual de la información financiera. Esto permite ver en detalle los ingresos y gastos que ha tenido el usuario en cada periodo específico. De esta manera, el usuario puede analizar y comparar su comportamiento financiero mes a mes, identificando patrones, variaciones y áreas de mejora. Un ejemplo de los reportes generados se puede apreciar en la figura 31.

Figura 31

Reporte de ingresos y gastos del usuario en la aplicación.



Nota. Desarrollado por los autores del presente documento. Reportes Totales

Cuando se selecciona el botón "Ver Detalle" en la tabla de balance mensual, se despliega un panel que contiene la información específica de todos los ingresos y gastos que ha tenido el usuario a lo largo del mes. Este panel proporciona un desglose detallado, permitiendo al usuario revisar cada transacción individual. Además, se ha implementado una funcionalidad que permite exportar el reporte en un archivo PDF. Esto facilita que el usuario pueda descargar y guardar la información desde el programa, asegurando así un acceso offline y una documentación más organizada de sus finanzas. El resultado del detalle se visualiza en la figura 32.

Figura 32

Detalle de reporte de ingresos y gastos del usuario en la aplicación.



Reporte Mayo 2024

Total Ingresos: $ 56.050.000,00

Total Restante: $ 42.785.000,00

Total Gastos: $ 13.265.000,00

Porcentaje Gastos: 23%

Ingresos

Valor	Fecha	Tipo de Ingreso	Descripción
$ 50.000.000,00	2024-05-03	Otros	Venta de Carro
$ 6.000.000,00	2024-05-14	Salario	Salario
$ 50.000,00	2024-05-13	Otros	venta camisa

Gastos

Fecha	Valor	Descripción	Categoría	Subcategoría
2024-05-02	$ 65.000,00	Compra de Pantalaon	Personal	Ropa
2024-05-03	$ 12.300.000,00	Compra de moto	Vehiculo	Compra Vehiculo
2024-05-07	$ 900.000,00	Compra celular	Tecnología	Celular

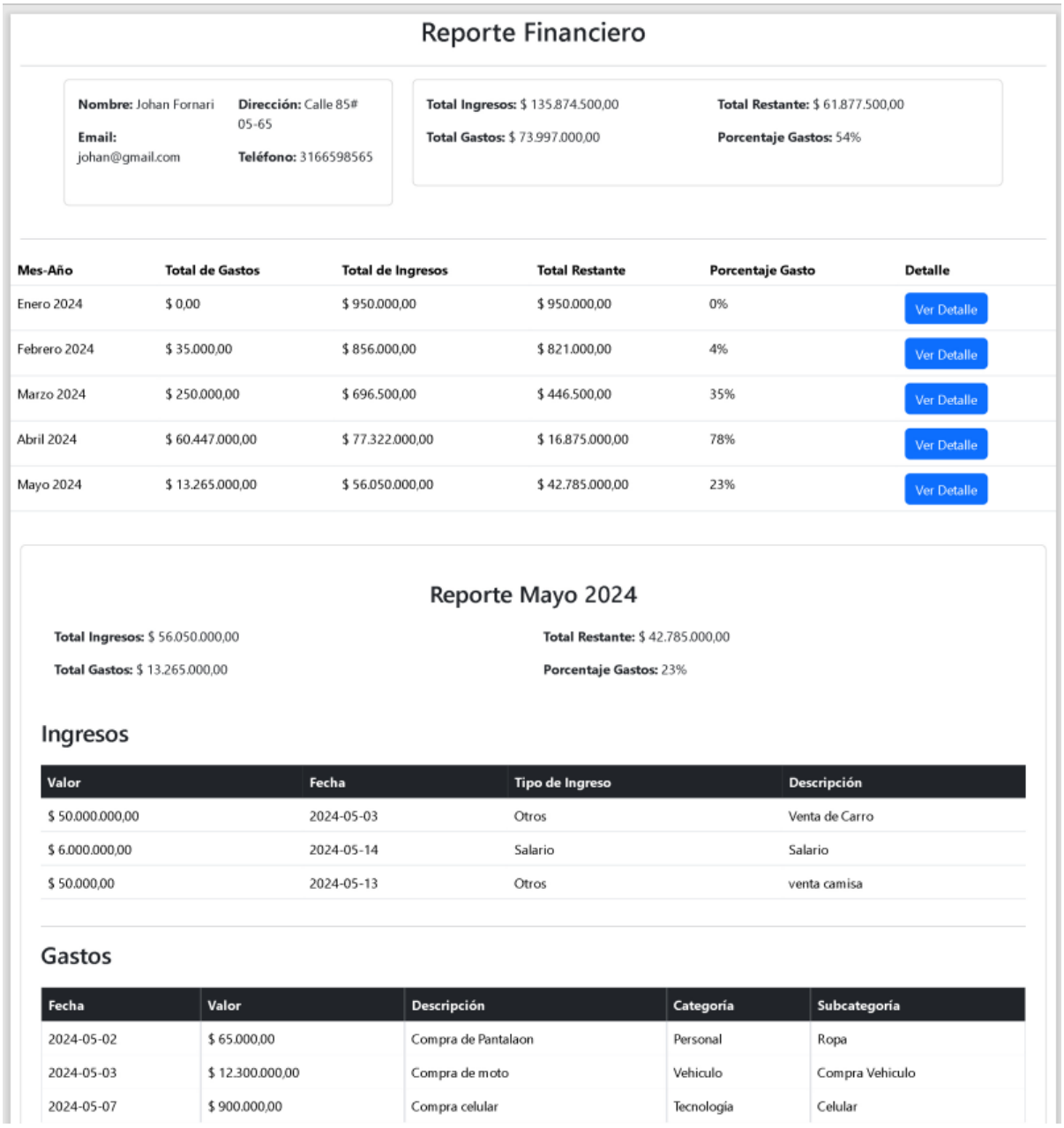
Generar Reporte

Nota. Desarrollado por los autores del presente documento. Detalle reportes por mes.

Al seleccionar el botón generar reporte se descarga un archivo con formato pdf que contiene el detalle del mes. En la figura 33 podemos visualizar el resultado obtenido en el archivo descargado. La información desplegada, incluyendo el desglose detallado de ingresos y gastos ilustra claramente cómo se presenta la información al usuario y cómo puede interactuar con ella para un mejor control financiero.

Figura 33

Reporte de ingresos y gastos del usuario en formato pdf.



Reporte Financiero

Nombre: Johan Fornari **Dirección:** Calle 85# 05-65
Email: johan@gmail.com **Teléfono:** 3166598565

Total Ingresos: $ 135.874.500,00 **Total Restante:** $ 61.877.500,00
Total Gastos: $ 73.997.000,00 **Porcentaje Gastos:** 54%

Mes-Año	Total de Gastos	Total de Ingresos	Total Restante	Porcentaje Gasto	Detalle
Enero 2024	$ 0,00	$ 950.000,00	$ 950.000,00	0%	Ver Detalle
Febrero 2024	$ 35.000,00	$ 856.000,00	$ 821.000,00	4%	Ver Detalle
Marzo 2024	$ 250.000,00	$ 696.500,00	$ 446.500,00	35%	Ver Detalle
Abril 2024	$ 60.447.000,00	$ 77.322.000,00	$ 16.875.000,00	78%	Ver Detalle
Mayo 2024	$ 13.265.000,00	$ 56.050.000,00	$ 42.785.000,00	23%	Ver Detalle

Reporte Mayo 2024

Total Ingresos: $ 56.050.000,00 **Total Restante:** $ 42.785.000,00
Total Gastos: $ 13.265.000,00 **Porcentaje Gastos:** 23%

Ingresos

Valor	Fecha	Tipo de Ingreso	Descripción
$ 50.000.000,00	2024-05-03	Otros	Venta de Carro
$ 6.000.000,00	2024-05-14	Salario	Salario
$ 50.000,00	2024-05-13	Otros	venta camisa

Gastos

Fecha	Valor	Descripción	Categoría	Subcategoría
2024-05-02	$ 65.000,00	Compra de Pantalaon	Personal	Ropa
2024-05-03	$ 12.300.000,00	Compra de moto	Vehiculo	Compra Vehiculo
2024-05-07	$ 900.000,00	Compra celular	Tecnologia	Celular

Nota. Desarrollado por los autores del presente documento. Reporte descargado formato pdf

Parte III. Cierre de la Investigación

Capítulo 6. Resultados y Discusión

Con la implementación del proyecto, se logró construir un prototipo de software que brinda una solución a la problemática de la gestión de gastos personales, ayudando a las personas con ingresos económicos a organizar sus finanzas.

Mediante un modelo de desarrollo iterativo y la adopción de una arquitectura empresarial, se desarrolló una solución escalable, con una lógica de negocio flexible y fácilmente modificable, que permite agregar nuevas funcionalidades de manera sencilla y adaptarse rápidamente a los requerimientos del usuario.

Asimismo, la aplicación de un modelo de generación automática de código facilita la escalabilidad de la aplicación, permitiendo añadir nuevas funcionalidades con mayor eficiencia.

El prototipo creado para gestionar los gastos personales es una herramienta útil que nos permite llevar un control detallado de los ingresos y gastos. Además, gracias a su interfaz intuitiva hace que el seguimiento de las finanzas sea sencillo y accesible para la mayoría de los individuos que tengan un nivel medio de experiencia con la tecnología. Sin embargo, la efectividad de esta herramienta depende en gran medida del compromiso del usuario. Es fundamental que los usuarios registren sus ingresos y gastos de manera constante y precisa para obtener una representación fiel de sus finanzas. En conclusión, la aplicación para el control de gastos personales es una solución útil y práctica para administrar las finanzas, maximizando su efectividad con el compromiso de los usuarios.

Capítulo 7. Conclusiones

7.1. Verificación, contraste y evaluación de los objetivos

En esta sección se analizarán las tareas cuyo desarrollo permitieron cumplir el objetivo principal. También se analizarán los resultados obtenidos de la encuesta de usabilidad propuesta en la metodología del proyecto dado el enfoque cuantitativo adoptado.

- *Diseñar un modelo de datos escalable que permita la adición de atributos en las tablas de la base de datos a medida que se creen gastos personalizados por parte de los usuarios.*

Para validar este objetivo, se diseñó una aplicación web utilizando el lenguaje de programación Java y una arquitectura empresarial JEE. Esta arquitectura soporta la implementación de una solución flexible y altamente escalable, permitiendo la adición de nuevos componentes de lógica de negocio sin necesidad de modificar el código fuente de la aplicación. Además, se adoptó un modelo de desarrollo iterativo, que facilita la incorporación de atributos en las tablas de la base de datos conforme los usuarios crean gastos personalizados. Esta metodología garantiza que la aplicación pueda adaptarse dinámicamente a los requerimientos cambiantes de los usuarios, asegurando la escalabilidad y flexibilidad del modelo de datos diseñado.

- *Desarrollar un prototipo funcional de la aplicación haciendo uso de un modelo de generación automática de código aplicado a los CRUDS del sistema.*

Se desarrolló un prototipo funcional de la aplicación cuyo backend fue desarrollado en Java mediante SpringFramework, proporcionando diversos servicios de la capa de lógica de negocio y que son consumidos por el componente de frontend desarrollado en React.

La generación automática de código, realizada con la ayuda de la librería Apache Velocity, facilitó la adición de atributos al modelo de lógica de negocios de manera automatizada, adaptándose a las necesidades cambiantes de los usuarios de la aplicación.

Además, se realizó un modelo de generación de código con ayuda de la misma librería que permitió crear de forma automática clases que contienen la programación necesaria para presentar todas las funcionalidades de un crud a nivel de backend, estas operaciones son crear, actualizar, obtener y eliminar en la base de datos.

- *Desarrollar reportes de presupuestos y gastos.*

Se crea un modelo de reportes de gastos que contiene la información general del usuario, se construye un apartado dedicado a los ingresos y gastos que el usuario ha presentado y una segmentación de este presentado mes a mes. El usuario tiene la posibilidad de ver el detalle de los gastos y adicionalmente descargar estos reportes en formato PDF.

Adicionalmente, de acuerdo con la técnica de recolección de datos, teniendo en cuenta el enfoque cuantitativo de este trabajo, basándonos en los resultados de la encuesta realizada a personas que probaron el prototipo funcional de la aplicación, tomando como base una muestra de 15 personas encuestadas, se puede concluir que la mayoría de los participantes han utilizado herramientas tecnológicas para el control de sus gastos personales, y la gran mayoría de ellos considera que la aplicación de gastos es fácil de usar (10 de 15). También, 12 de las 15 personas

encuestadas expresaron su intención de seguir usando la aplicación durante al menos los próximos 6 meses. Sin embargo, es importante destacar que un número significativo de participantes no percibió una mejora en el control de su presupuesto y gastos personales después de probar la aplicación. Lo anterior sugiere que, si bien la aplicación puede resultar fácil de usar, podría no satisfacer las necesidades o expectativas de los usuarios en términos de mejorar el control de sus finanzas personales. Para los desarrolladores de la aplicación, sería recomendable considerar de manera más atenta y específica las necesidades de los usuarios que permitan ajustar o mejorar funcionalidades de la aplicación para aumentar la efectividad y satisfacción general de los usuarios al usar la aplicación.

7.2. Síntesis del modelo propuesto

7.3. Aportes originales

Como principal aporte, se destaca la incorporación automatizada de atributos relacionados con la clasificación de gastos a la aplicación, junto con la presentación de una interfaz de usuario intuitiva y sencilla. Esto permite a los usuarios llevar un control detallado de sus ingresos y gastos asociados a distintas categorías y subcategorías. Esta funcionalidad posibilita una visualización más clara y oportuna del destino de los ingresos, contribuyendo así a una mejor planificación financiera.

7.4. Trabajos o Publicaciones derivadas

Este proyecto ofrece una base para futuras investigaciones y publicaciones, abriendo posibilidades en áreas como la ampliación de los reportes para reflejar hábitos de consumo específicamente en la población colombiana, la aplicación de técnicas de machine learnng para la predicción de gastos comunes que permitan mejorar el control financiero de las personas.

Además, se sugiere investigar técnicas de inteligencia artificial para la adición automática de atributos a las entidades que conforman la lógica de negocios de la aplicación, lo que permitiría brindar una experiencia mucho más personalizada que se adapte a las necesidades cambiantes de los usuarios.

Capítulo 8. Prospectiva del Trabajo de Grado

8.1. Líneas de investigación futuras

Como parte de las líneas de investigación se podría profundizar en la identificación de hábitos de consumo de la población colombiana haciendo uso de encuestas y de estadísticas oficiales para poder identificar aquellas necesidades que no fueron contempladas en este proyecto y que en últimas permitan desarrollar nuevas funcionalidades que sirvan aún más en el control de gastos a la población en general.

Se propone también la creación de un modelo de Machine Learning que se alimente de las estadísticas relacionadas con hábitos de consumo de las personas con ingresos económicos y que permita predecir cuales son aquellos gastos más comunes en la población y con base en el análisis de estos datos se puedan proponer nuevas funcionalidades para mejorar el control de gastos.

8.2. Trabajos de Investigación futuros

Se proponen como futuros trabajos de investigación la creación de grupos que faciliten la gestión de presupuestos compartidos entre múltiples usuarios de la aplicación, así como la integración con entidades bancarias y billeteras virtuales. Además, se plantea que el modelo propuesto para la adición automatizada de atributos pueda aplicarse a aquellos atributos que formen parte de las claves primarias y/o foráneas de los componentes que integran la lógica de negocios de la aplicación. Se sugiere también la incorporación de técnicas de inteligencia artificial para la adición automática de atributos a las diferentes entidades del modelo desarrollado.

Bibliografía

Olmedo L. (2009). Las finanzas personales. Revista Escuela de Administración de Negocios, 123-144. ISSN: 0120-8160. Recuperado de: https://www.redalyc.org/articulo.oa?id=20612980007

Sarmiento J. (2005). Finanzas personales. Profesor Pontificia Universidad Javeriana. - PDF Free Download. (s/f). Recuperado de: https://docplayer.es/11231528-Finanzas-personales-julio-a-sarmiento-sabogal-profesor-pontificia-universidad-javeriana-www-javeriana-edu-co-decisiones-julio.html

García, V. (2014). Introducción a las finanzas. © 2014, Grupo Editorial Patria, https://dmd.unadmexico.mx/contenidos/DCSA/MODULOS/CFP/M1_VPEFP/U0/descargables/Introduccion_a_las_finanzas.pdf

Rodríguez, N. (2021). *Diseño de prototipo para el control de finanzas personales mediante análisis predictivo en una aplicación móvil* [Tesis de Especialización, Universidad Distrital Francisco José de Caldas]. Repositorio Institucional Universidad Distrital - RIUD.

Martínez Acosta, D. (2014). *Herramienta para la generación automática del código fuente para aplicaciones con arquitectura modelo vista controlador (MVC) bajo desarrollo dirigido por modelos textuales (MDD)*. [Tesis de Maestría, Universidad Nacional de Colombia]. Repositorio Institucional - UNAL.

López, J. (2016). L*as finanzas personales, un concepto que va más allá de su estructura. estado del arte período* 2006-2016. Universidad EAFIT.

Boehm, B. (1988). A spiral model of software development and enhancement. ACM SIGSOFT Software Engineering Notes, 11(4), 14-24.

Pressman, R. S. (2014). Ingeniería del software: Un enfoque práctico (7ma ed.). McGraw-Hill Education.

Schwaber, K., & Beedle, M. (2002). Agile Software Development with Scrum. Prentice Hall.

Sommerville, I. (2016). Ingeniería del software (10ma ed.). Pearson.

Pressman, R. S., & Maxim, B. R. (2015). Ingeniería del Software: Un enfoque práctico. McGraw-Hill.

Larman, C. (2004). Agile and Iterative Development: A Manager's Guide. Addison-Wesley.

Prieto Rodriguez, C. A. & Madrid Caballero, D. A. (2021). Prototipo generador de código fuente para aplicaciones de servicios. Caso de estudio: proceso de alquiler de libros de una biblioteca. http://hdl.handle.net/11349/29569

García, J. (2018). Gestión financiera personal: Estrategias para el manejo efectivo de ingresos y gastos. Editorial Finanzas y Economía.

Disposiciones analizadas por Avance Jurídico Casa Editorial Ltda.©

"*Leyes desde 1992 - Vigencia Expresa y Sentencias de Constitucionalidad*"

ISSN [1657-6241 (En línea)]. Última actualización: 20 de octubre de 2023 - (Diario Oficial No. 52.535 - 1 de octubre de 2023).

http://www.secretariasenado.gov.co/senado/basedoc/ley_1328_2009.html#10

Disposiciones analizadas por Avance Jurídico Casa Editorial Ltda.©

"*Leyes desde 1992 - Vigencia Expresa y Sentencias de Constitucionalidad*"

ISSN [1657-6241 (En línea)]. Última actualización: 20 de octubre de 2023 - (Diario Oficial No. 52.535 - 1 de octubre de 2023)

http://www.secretariasenado.gov.co/senado/basedoc/ley_1450_2011_pr002.html#132

DECRETO 457 DE 2014, *"Por el cual se organiza el Sistema Administrativo Nacional para la Educación Económica y Financiera, se crea una Comisión Intersectorial y se dictan otras disposiciones.".* Publicado en el Diario Oficial. N. 49083. 5 de marzo de 2014.

https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=66298

Referencias Web

International Business Machines Corporation. (12 de septiembre de 2023). *¿Qué es Java Spring Boot?*. IBM. https://www.ibm.com/mx-es/topics/java-spring-boot

Apache Maven Project (14 de septiembre de 2023). *What is Maven?.* Apache Maven project. https://maven.apache.org: https://maven.apache.org/what-is-maven.html

PostgreSQL: The World's Most Advanced Open Source Relational Database. (14 de septiembre de 2023). *What is PostgreSQL?.* The World's Most Advanced Open Source Relational Database. https://www.postgresql.org/about/

Atlassian. (18 de septiembre de 2023). *Qué es scrum y cómo empezar: Guía de la metodología scrum: qué es, cómo funciona y cómo empezar*. Atlassian. https://www.atlassian.com/es/agile/scrum

Mint. (25 de septiembre de 2023). *Managing money, made simple.* Mint. https://mint.intuit.com/

Ynab. (25 de septiembre de 2023). *Change Your Relationship With Money*. Ynab. https://www.ynab.com/

pocketguard. (25 de septiembre de 2023) *pocketguard* https://pocketguard.com/

Goodbudget. (28 de septiembre de 2023)) *Budget with a why.* Goodbudget. .https://goodbudget.com/

Davivienda. (15 de octubre de 2023). *Simulador de presupuesto*. Finanzas en casa. https://misfinanzasencasa.davivienda.com/finanzas-personales/simuladores/simulador-de-presupuesto

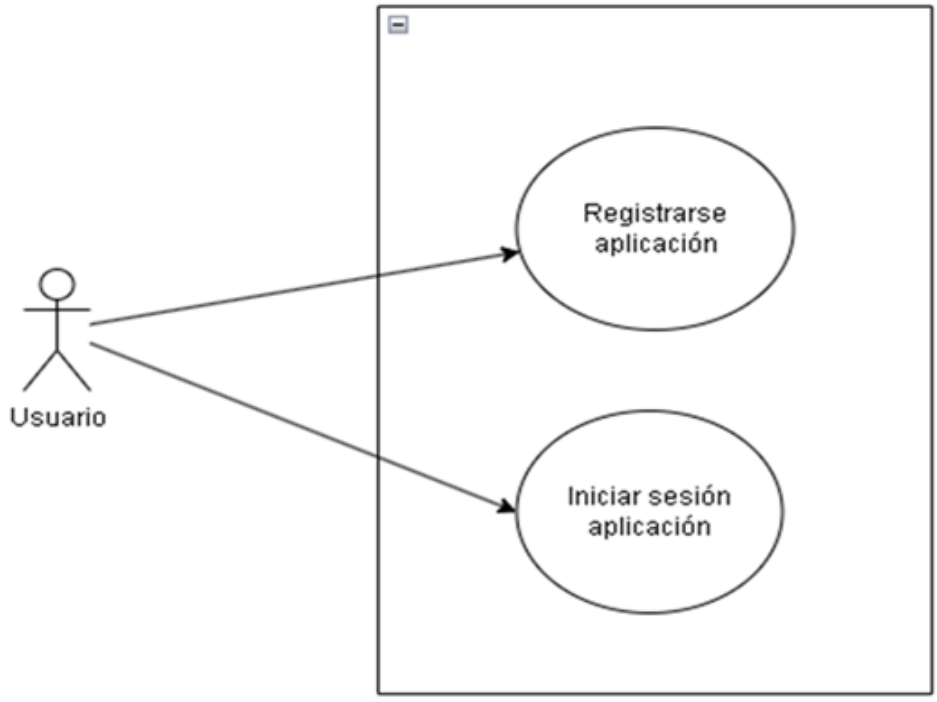
Apache Software Foundation. (09 de abril 2024). Apache Velocity. https://velocity.apache.org/

Anexos

A. Diagramas de Casos de Uso

Figura 2

Caso de uso para registro e inicio sesión usuario



Nota. Desarrollado por los autores del presente documento. Casos de Uso 1.

Caso de uso: Registrarse

Actor principal: Administrador, usuario aplicación.

Objetivo en contexto: El usuario realiza el registro de su cuenta proporcionando sus datos personales relevantes y asociándose a un correo, nombre de usuario y contraseña.

Precondiciones: El aplicativo debe estar corriendo en un servidor al cual el usuario se pueda registrar.

Caso de uso: Iniciar sesión

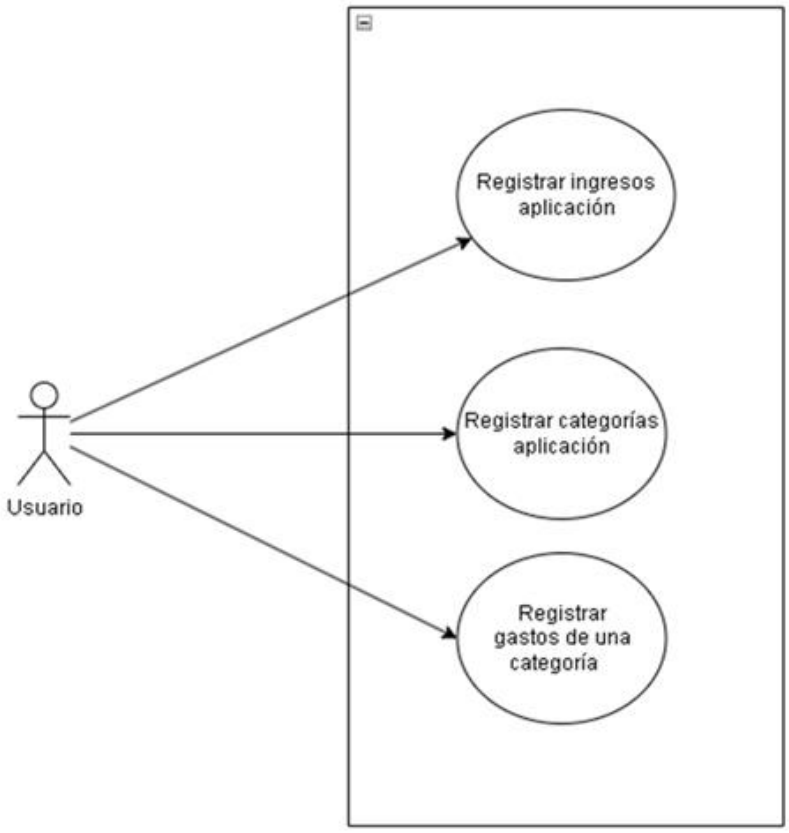
Actor principal: Administrador, usuario aplicación.

Objetivo en contexto: El usuario realiza el inicio de sesión en su cuenta el correo y contraseña correspondiente.

Precondiciones: El usuario debe estar registrado.

Figura 3

Casos de uso para registro de ingresos, categorias y gastos



Nota. Desarrollado por los autores del presente documento. Casos de Uso 2.

Caso de uso: Registrar Ingresos

Actor principal: usuario aplicación.

Objetivo en contexto: El usuario realiza el registro de sus ingresos disponibles en el aplicativo, los cuales servirán como base para realizar el reporte de gastos vs presupuesto.

Precondiciones: haber iniciado sesión con un usuario registrado y tener ingresos económicos para registrar.

Caso de uso: Registrar categorías

Actor principal: usuario aplicación, administrador.

Objetivo en contexto: El usuario realiza el registro de las categorías que requiera las cuales corresponden a los diferentes tipos de gastos, en caso de que la categoría no exista el usuario tiene la posibilidad de crearla.

Precondiciones: haber registrado el presupuesto.

Caso de uso: Registrar gastos de una categoría

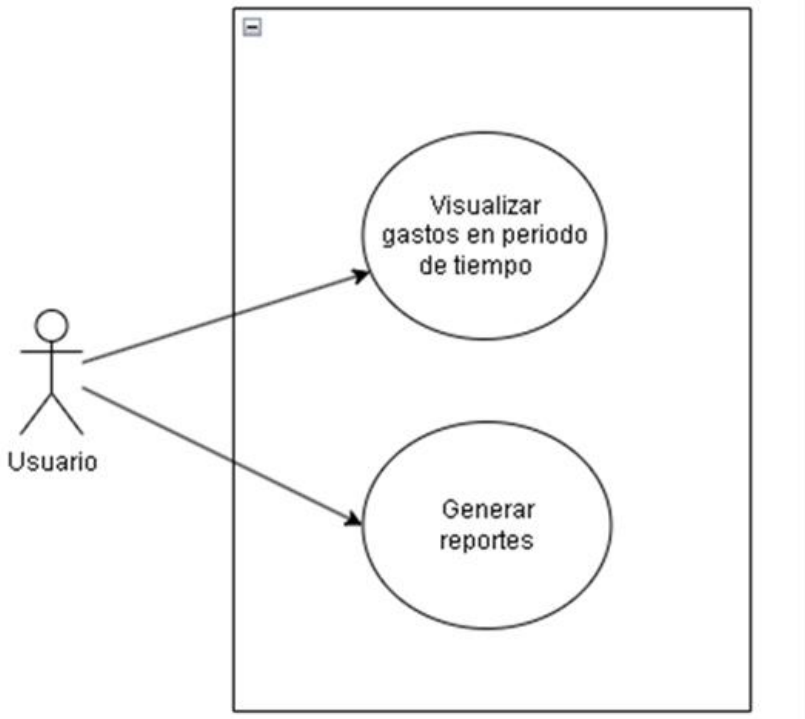
Actor principal: usuario aplicación

Objetivo en contexto: El usuario tiene la posibilidad de realizar una distribución de gastos sobre una categoría específica y por ende crear subcategorías.

Precondiciones: haber registrado las categorías que se requieren distribuir.

Figura 4

Caso de uso para la generación de reportes



Nota. Desarrollado por los autores del presente documento. Casos de Uso 3.

Caso de uso: Visualizar gastos en un periodo de tiempo

Actor principal: usuario aplicación

Objetivo en contexto: El usuario tiene la posibilidad de generar un resumen de los gastos que realizó y registró en un periodo de tiempo que configure.

Precondiciones: haber registrado los gastos de periodo solicitado en las categorías y subcategorías.

Caso de uso: Generar reportes

Actor principal: usuario aplicación

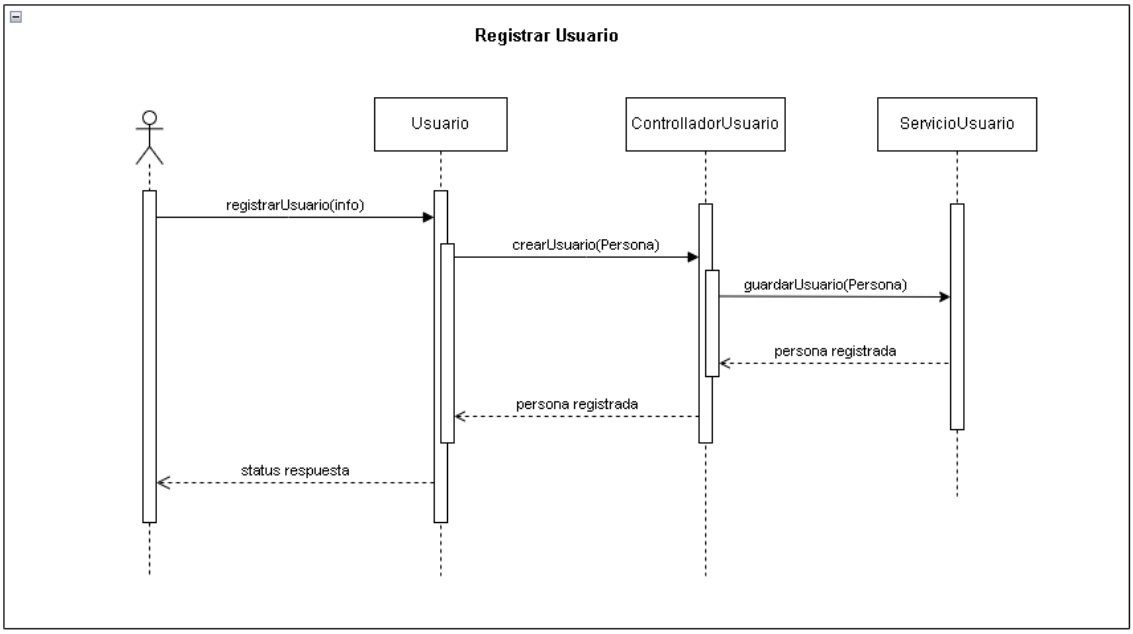
Objetivo en contexto: El usuario tiene la posibilidad de generar un reporte de los gastos y presupuestos escogiendo un periodo de tiempo que configure.

Precondiciones: haber registrado los gastos de periodo solicitado en las categorías y subcategorías.

B. Diagramas de secuencia

Figura 5

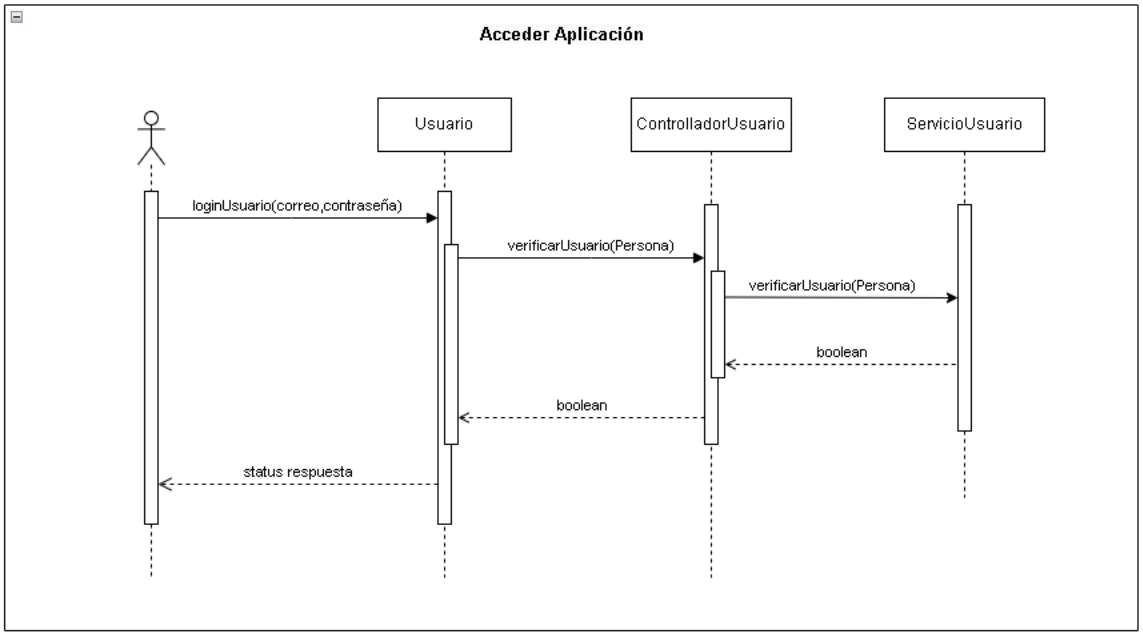
Diagrama de secuencia registro usuario



Nota. Desarrollado por los autores del presente documento. Diagrama registrar usuario.

Figura 6

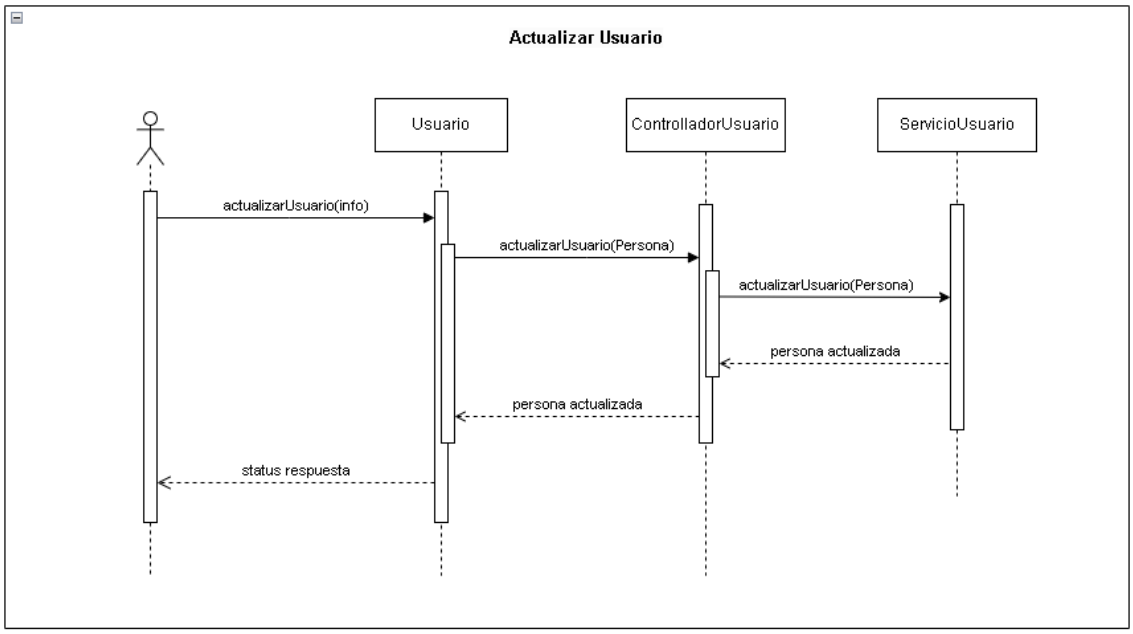
Diagrama de secuencia acceso a la aplicación



Nota. Desarrollado por los autores del presente documento. Diagrama acceder aplicación.

Figura 7

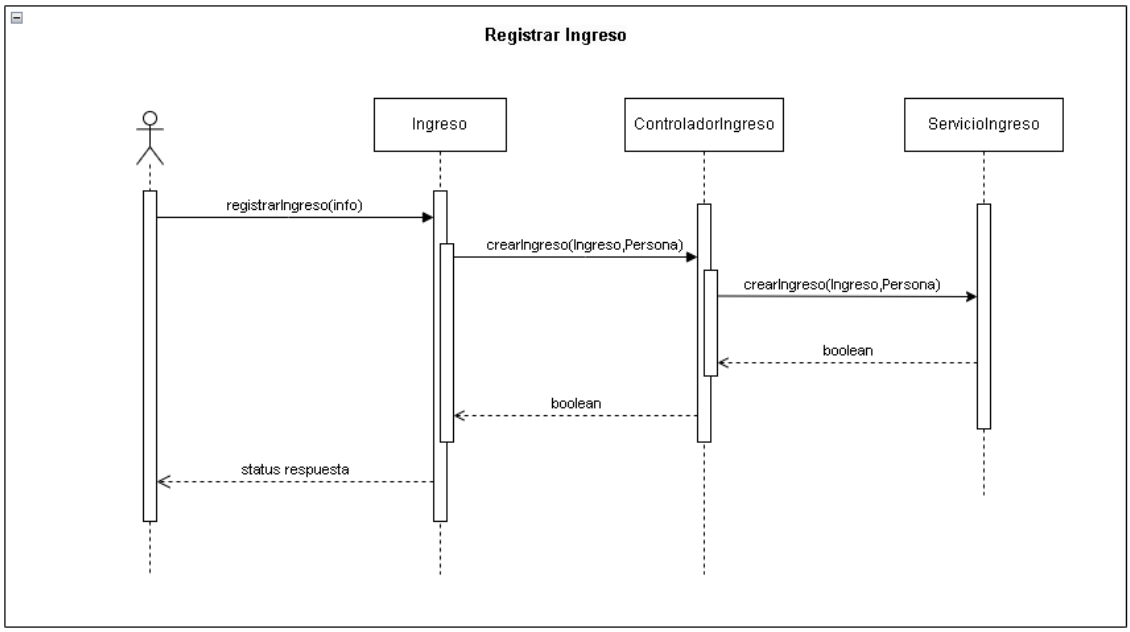
Diagrama de secuencia actualizar usuario



Nota. Desarrollado por los autores del presente documento. Diagrama actualizar usuario.

Figura 8

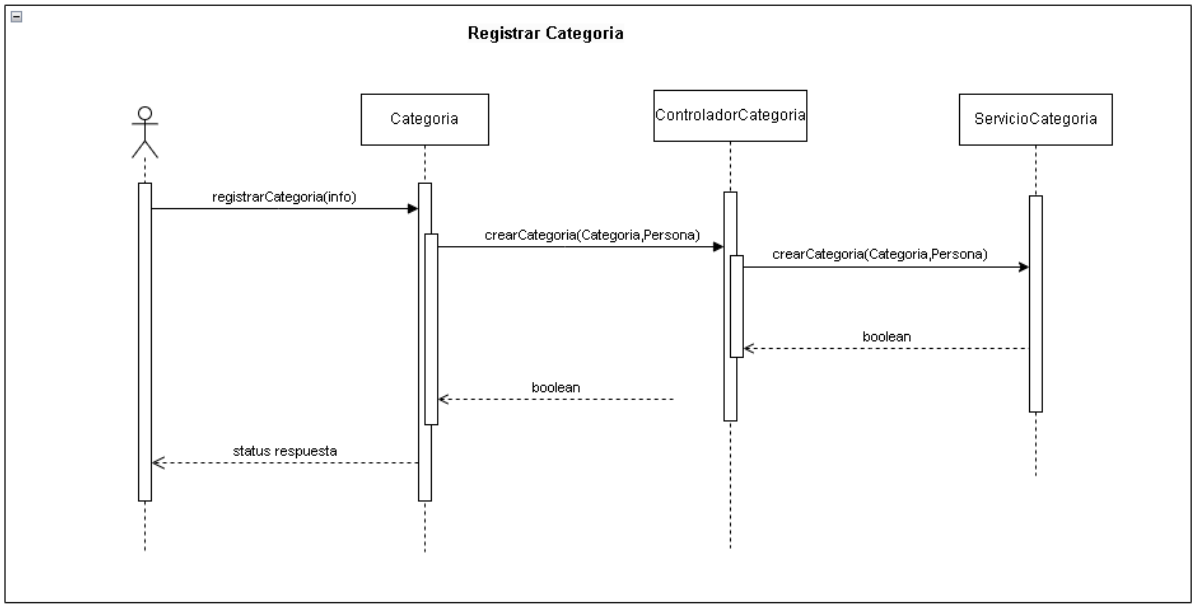
Diagrama de secuencia registro de ingreso



Nota. Desarrollado por los autores del presente documento. Diagrama registrar ingreso.

Figura 9

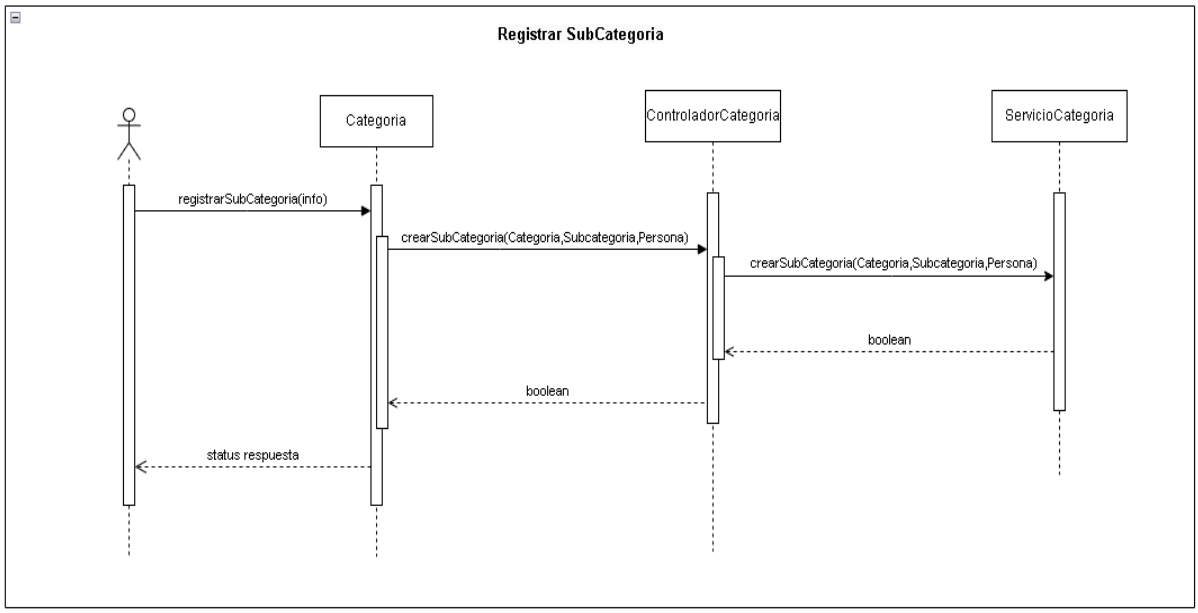
Diagrama de secuencia registro categoría



Nota. Desarrollado por los autores del presente documento. Diagrama registrar categoría.

Figura 10

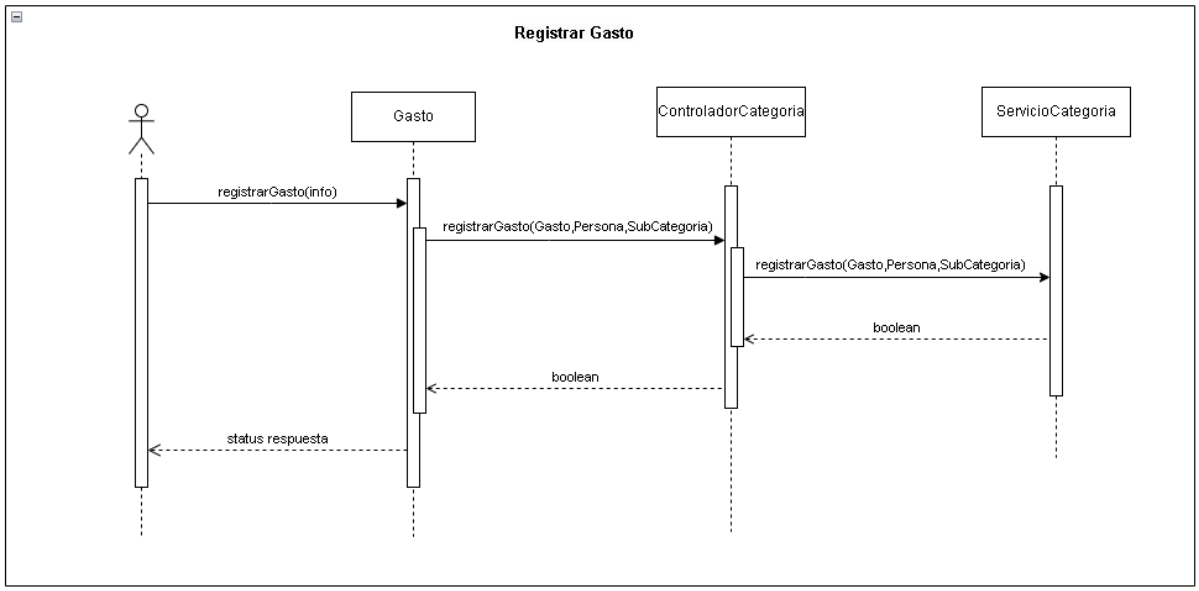
Diagrama de secuencia registro de subcategoría.



Nota. Desarrollado por los autores del presente documento. Diagrama registrar subcategoría.

Figura 11

Diagrama de secuencia registro de gasto

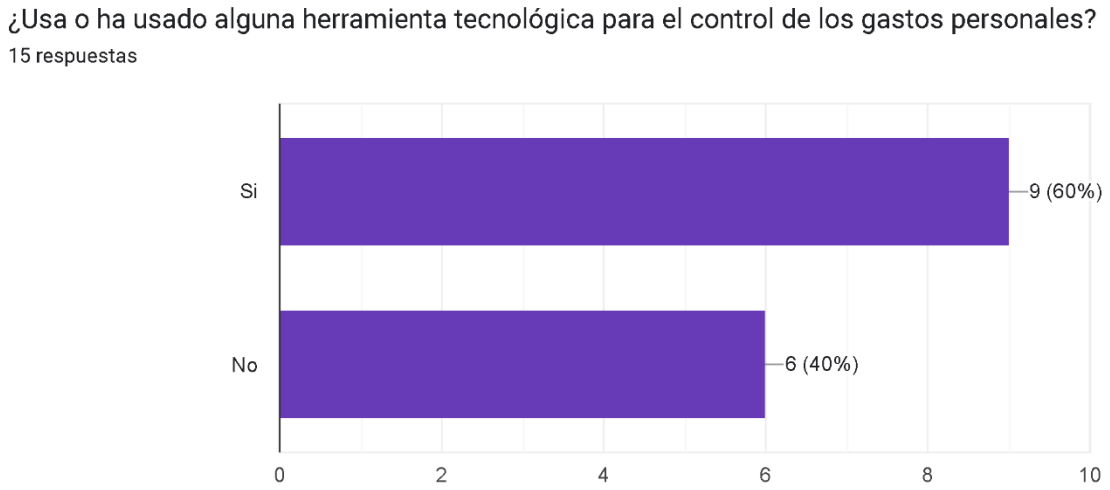


Nota. Desarrollado por los autores del presente documento. Diagrama registrar gasto.

C. Encuesta de Usabilidad Aplicación Finanzas Personales

Figura 11

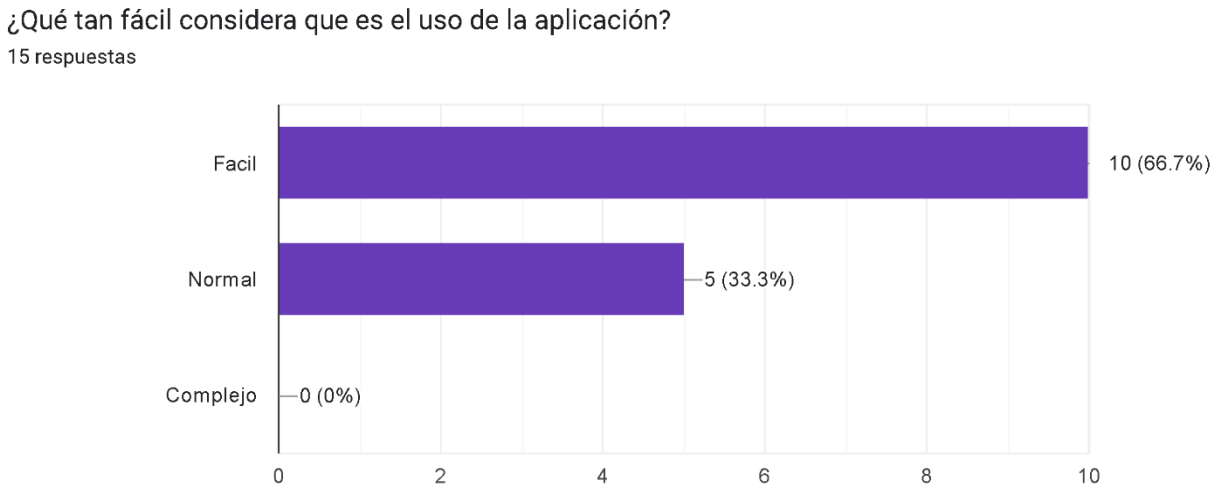
Gráfico respuesta pregunta 2 encuesta de usabilidad



Nota. Grafica extraída de Google Forms, resultado encuesta usabilidad aplicación, 2024 https://docs.google.com/forms/d/e/1FAIpQLSe09kv0Wr5wCvM9x_02zHPw47216Cs3F9Yrg6GF1i0ltqczWw/viewform

Figura 11

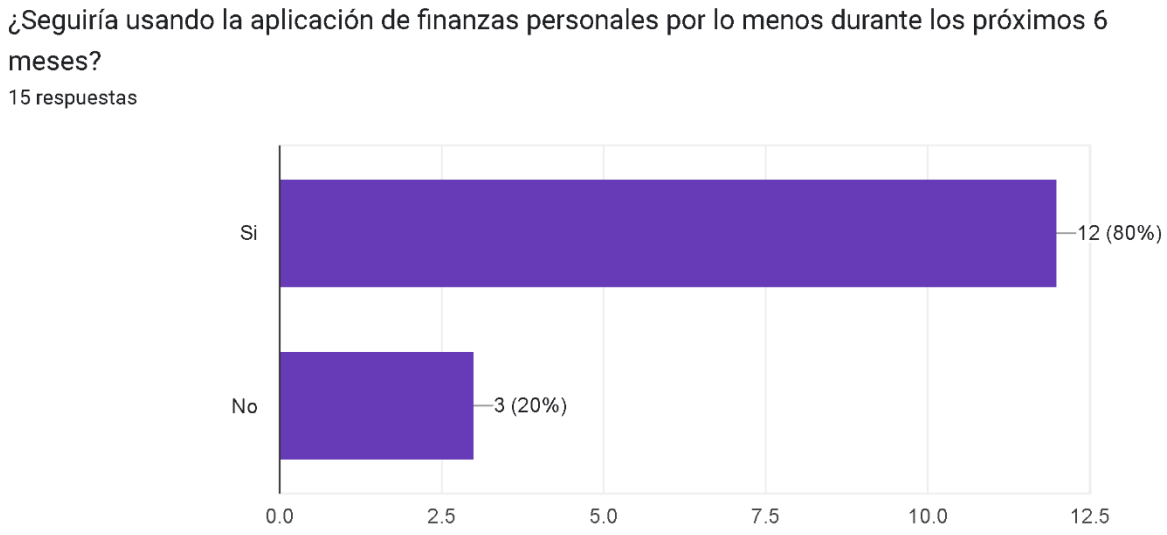
Gráfico respuesta pregunta 2 encuesta de usabilidad



Nota. Grafica extraída de Google Forms, resultado encuesta usabilidad aplicación, 2024 https://docs.google.com/forms/d/e/1FAIpQLSe09kv0Wr5wCvM9x_02zHPw47216Cs3F9Yrg6GF1i0ltqczWw/viewform

Figura 11

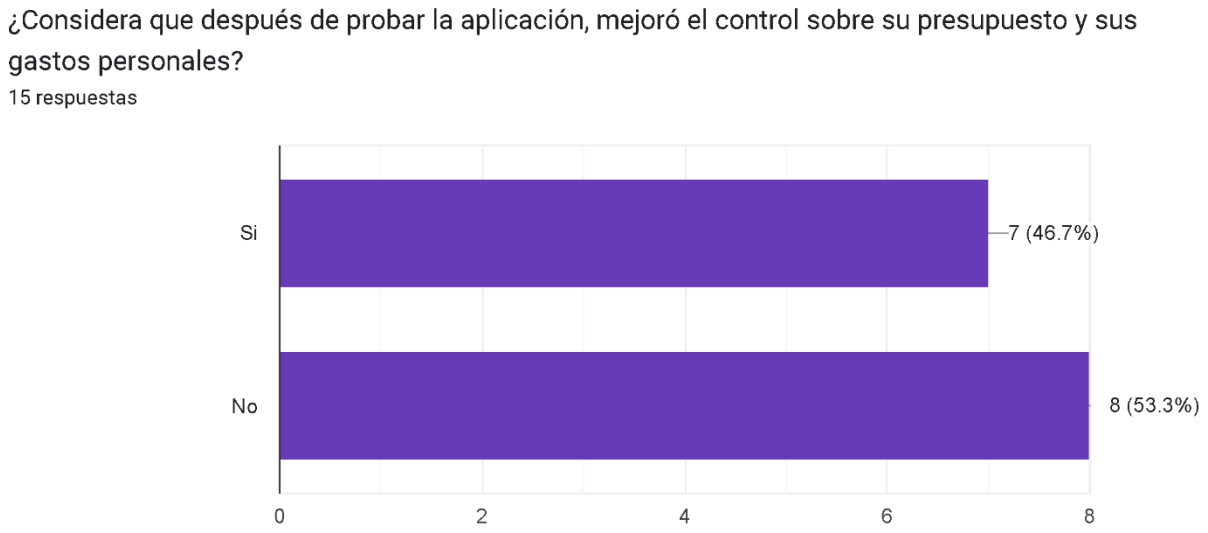
Gráfico respuesta pregunta 2 encuesta de usabilidad



Nota. Grafica extraída de Google Forms, resultado encuesta usabilidad aplicación, 2024 https://docs.google.com/forms/d/e/1FAIpQLSe09kv0Wr5wCvM9x_02zHPw47216Cs3F9Yrg6GF1i0ltqczWw/viewform

Figura 11

Gráfico respuesta pregunta 2 encuesta de usabilidad



Nota. Grafica extraída de Google Forms, resultado encuesta usabilidad aplicación, 2024 https://docs.google.com/forms/d/e/1FAIpQLSe09kv0Wr5wCvM9x_02zHPw47216Cs3F9Yrg6GF1i0ltqczWw/viewform

D. Repositorio Desarrollo

D1. FrontEnd React

https://github.com/JohanFornari/GestionGastosFrontEnd.git

D2. BackEnd SpringBoot

https://github.com/JohanFornari/GestionGastosNew.git