****

Docente: Mg. Marcela Cifuentes Velásquez

Ingeniería de Sistemas

PRESENTADO POR

DANIEL CHAVES PARRA

JUAN PABLO VILLA RAMIREZ

STEVEN ANDRES MADERA MARTINEZ

DAVID SANTIAGO ÁVILA RINCÓN

JUAN CARLOS MONCADA CASTRO

**PROYECTO Finaicer**

**DESARROLLO DE UN APLICACIÓN MOVIL**

Bogotá - Colombia

09 Agosto 2025

Universidad Libre – Sede el Bosque

Gestión de proyectos

**Contenido**

[Tabla de imágenes 3](#_Toc212124048)

[**Tabla de Anexos** 4](#_Toc212124049)

[Resumen Ejecutivo 5](#_Toc212124050)

[Introducción 6](#_Toc212124051)

[Objetivos 7](#_Toc212124052)

[Objetivo General 7](#_Toc212124053)

[Objetivos Específicos 7](#_Toc212124054)

[Alcance 8](#_Toc212124055)

[Límites y Delimitaciones 8](#_Toc212124056)

[Metodología 8](#_Toc212124057)

[Enfoques y Procesos 9](#_Toc212124058)

[Aplicación de las Etapas 9](#_Toc212124059)

[1. ETAPA DE INICIO 9](#_Toc212124060)

[**Propósito** 10](#_Toc212124061)

[**Elementos Fundamentales** 10](#_Toc212124062)

[2. Etapa de planificación 11](#_Toc212124063)

[Definición de Objetivos 11](#_Toc212124064)

[Identificación del Público Objetivo 12](#_Toc212124065)

[Análisis de Competencia 13](#_Toc212124066)

[Requisitos Funcionales y Técnicos 14](#_Toc212124067)

[Creación de un Plan de Proyecto 14](#_Toc212124068)

[Bitácora de avance en cronograma 16](#_Toc212124069)

[1. Planificación inicial y levantamiento de requisitos 16](#_Toc212124070)

[2. Diseño UX/UI e investigación 17](#_Toc212124071)

[3. Desarrollo MVP (Aplicación básica) 17](#_Toc212124072)

[4. Integración de IA y Machine Learning 18](#_Toc212124073)

[5. Visualización y reportes 19](#_Toc212124074)

[Análisis del Documento: Requerimientos Identificados – FinAIcer 20](#_Toc212124075)

[1. Relevancia dentro del ciclo de vida del proyecto 20](#_Toc212124076)

[2. Análisis de los requerimientos del diseño UX/UI 20](#_Toc212124077)

[3. Análisis de los requerimientos del MVP 21](#_Toc212124078)

[4. Consideraciones técnicas 21](#_Toc212124079)

[5. Valor agregado del documento 21](#_Toc212124080)

[Mockups 22](#_Toc212124081)

[1. Coherencia general del diseño 22](#_Toc212124082)

[2. Análisis de la arquitectura de interacción 23](#_Toc212124083)

[3. Análisis de los módulos funcionales 23](#_Toc212124084)

[4. Valoración de usabilidad y estética 24](#_Toc212124085)

[5. Evaluación técnica 24](#_Toc212124086)

[Diseño arquitectónico 25](#_Toc212124087)

[1. Arquitectura de microservicios (vista de alto nivel) 25](#_Toc212124088)

[2. Diagrama de contexto (vista externa) 26](#_Toc212124089)

[3. Diagrama funcional / de bloques 27](#_Toc212124090)

[4. Diagrama de secuencia (flujo de notificación bancaria) 28](#_Toc212124091)

[5. Evaluación general de la arquitectura 28](#_Toc212124092)

[Gestión de Riesgos 29](#_Toc212124093)

[1. Enfoque general de la gestión de riesgos 29](#_Toc212124094)

[2. Riesgos principales y estrategias de mitigación 29](#_Toc212124095)

[3. Evaluación global 31](#_Toc212124096)

[Codificación 32](#_Toc212124097)

[Implementación Técnica del Backend y Frontend – Proyecto FinAIcer 32](#_Toc212124098)

[1. Arquitectura general 32](#_Toc212124099)

[2. Backend – Estructura y funcionalida 32](#_Toc212124100)

[3. Frontend – Angular/Ionic 33](#_Toc212124101)

[4. Integración Backend–Frontend–Base de Dato 33](#_Toc212124102)

[5. Evaluación técnica 34](#_Toc212124103)

[Código Fuente – FinAIcer 34](#_Toc212124104)

[1. Estructura del Frontend 35](#_Toc212124105)

[2. Estructura del Backend 35](#_Toc212124106)

[5. Evaluación General del Código 36](#_Toc212124107)

[Bibliografía 37](#_Toc212124108)

# Tabla de imágenes

[Imagen 1.EDT, Fuente propia. 8](#_Toc187067664)

# **Tabla de Anexos**

Documentos adicionales como gráficos, tablas, encuestas, imágenes.

[Anexo 1. Acta Aceptación. 8](#_Toc187068569)

[Anexo 2. Archivo del Cronograma y presupuesto. 10](#_Toc187068570)

Anexo 3. Solicitud de Cambios.

**(STAND BY)**

# Resumen Ejecutivo

FinAIcer es un asistente financiero móvil con inteligencia artificial que procesa notificaciones bancarias y de pagos, clasificándolas y transformándolas en reportes claros y útiles para el usuario. Este proyecto combina técnicas de machine learning, procesamiento de lenguaje natural y desarrollo móvil, con el objetivo de proporcionar a las personas un control más fácil y automatizado de sus finanzas. La aplicación se conecta a un backend que analiza los datos y ofrece visualizaciones en tiempo real.

FinAIcer es una solución innovadora diseñada para transformar la manera en que las personas gestionan sus finanzas personales. Se trata de un asistente inteligente que procesa información financiera proveniente de diversas fuentes —incluyendo notificaciones bancarias, aplicaciones de pago digital y registros de gastos— para clasificar, analizar y predecir el comportamiento económico del usuario.

Su diseño contempla una aplicación móvil como interfaz principal, una base de datos segura para el almacenamiento de la información y un motor de análisis central que se actualiza en tiempo real. Además, el sistema está pensado para una implementación gradual, permitiendo la incorporación de nuevas funciones como predicción de gastos, análisis de patrones de consumo y asesoramiento financiero personalizado

# Introducción

En un contexto donde la economía personal se ve afectada por factores como la inflación, la variabilidad de ingresos y el aumento de los gastos fijos, el manejo eficiente de las finanzas se ha convertido en una necesidad fundamental. Sin embargo, la mayoría de las personas carecen de herramientas prácticas y adaptadas a sus hábitos que les permitan monitorear y mejorar su salud financiera.

La gestión financiera personal se enfrenta a la dispersión de información en múltiples canales y formatos, especialmente en las notificaciones móviles de bancos, pasarelas de pago y comercios. FinAIcer nace como una solución que centraliza y analiza esa información, usando IA para clasificar, agrupar y generar reportes personalizados. Este informe detalla la planificación, desarrollo y resultados obtenidos en la creación del prototipo funcional.

FinAIcer nace como respuesta a esta necesidad, combinando inteligencia artificial y análisis de datos para ofrecer un acompañamiento financiero personalizado, accesible y de fácil uso.

A lo largo de este informe se presentarán los fundamentos conceptuales y técnicos que sustentan el desarrollo de FinAIcer, su arquitectura propuesta, las tecnologías a emplear y los beneficios esperados para el usuario final. Asimismo, se abordarán los retos técnicos y éticos de su implementación, especialmente en materia de privacidad y seguridad de la información.

# Objetivos

## Objetivo General

Desarrollar una aplicación móvil asistida por inteligencia artificial, capaz de procesar datos financieros personales obtenidos de las notificaciones del dispositivo, clasificarlos automáticamente, generar análisis en tiempo real y proporcionar proyecciones que ayuden al usuario a tomar mejores decisiones económicas.

## Objetivos Específicos

Identificar las necesidades de los usuarios en cuanto a gestión, clasificación y análisis de gastos personales.

Determinar las fuentes de datos financieras compatibles (ej. API de Nequi, Bancolombia, extractos, notificaciones) y sus limitaciones técnicas y legales.

Elaborar la arquitectura técnica de FinAIcer, incluyendo la integración entre aplicación móvil, base de datos y modelo de IA.

Diseñar la interfaz de usuario (UI/UX) asegurando una experiencia simple, intuitiva y accesible.

Realizar pruebas unitarias y de integración para asegurar el correcto funcionamiento de cada módulo.

Validar la precisión del modelo de IA y su desempeño en escenarios reales con datos de prueba.

# Alcance

La solución propuesta consiste en una aplicación móvil compatible con dispositivos Android e iOS, orientada al control y análisis de finanzas personales. Su funcionamiento se basa en el procesamiento de notificaciones SMS relacionadas con transacciones bancarias y billeteras digitales, tales como Nequi, Daviplata y la aplicación de Bancolombia, siempre que hayan sido previamente autorizadas por el usuario.

Dentro del alcance funcional se incluyen las siguientes características principales:

* Registro y categorización automática de transacciones.
* Visualización de balances, estadísticas y reportes personalizados.
* Generación de proyecciones de ingresos y gastos a corto y mediano plazo.
* Emisión de recomendaciones de ahorro y alertas ante patrones de gasto inusuales.

## Límites y Delimitaciones

* Versión inicial disponible solo para Android.
* Procesamiento de notificaciones únicamente en español.
* Datos obtenidos solo de notificaciones, no de conexiones directas a cuentas bancarias.
* Uso de un modelo de IA entrenado con dataset limitado para el prototipo.

# Metodología

Se empleará un enfoque ágil de desarrollo basado en la metodología Scrum, lo que permitirá una gestión flexible y adaptativa del proyecto. El trabajo se dividirá en sprints cortos de una o dos semanas, cada uno con objetivos claramente definidos y entregables parciales que serán revisados y validados con el equipo.

Cada sprint incluirá actividades de planificación, desarrollo, pruebas y revisión, garantizando que las funcionalidades se construyan de forma iterativa e incremental. Esto permitirá detectar errores o desviaciones en etapas tempranas y realizar ajustes sin comprometer los plazos generales.

En cuanto al componente de Inteligencia Artificial, el entrenamiento y validación del modelo se realizarán en fases previas a su integración con la aplicación móvil. Esto asegurará que el motor de IA alcance un nivel de precisión aceptable en la clasificación y análisis de datos antes de ser incorporado al entorno productivo. Para ello:

* Se recolectará un conjunto de datos representativo para el entrenamiento inicial.
* Se implementarán pruebas de validación cruzada y métricas como precisión, recall y F1-score para evaluar el desempeño.
* Una vez validado, el modelo será empaquetado e integrado al backend de la aplicación, manteniendo la posibilidad de actualizarlo de forma continua conforme se recolecten más datos reales de los usuarios.

Este enfoque garantiza que el producto final no solo cumpla con los requisitos funcionales, sino que también pueda evolucionar y mejorar con el tiempo.

## Enfoques y Procesos

* **IA y Machine Learning**: Procesamiento de lenguaje natural (NLP) para clasificar transacciones.
* **Desarrollo móvil**: Uso de React Native para interfaz y Node.js para backend.
* **Base de datos**: MongoDB para almacenamiento.
* **Pruebas**: Unitarias y de integración.

# Aplicación de las Etapas

## 1. ETAPA DE INICIO

En esta primera fase se definió la visión general del proyecto y se identificó la oportunidad que dio origen a FinAIcer.

* Identificación del problema: Muchos usuarios reciben notificaciones de movimientos financieros (pagos, cobros, transferencias) en sus dispositivos móviles, pero no cuentan con una herramienta que centralice, clasifique y analice esta información de forma automática. Esto genera que se pierdan oportunidades de control financiero y planificación.
* Definición preliminar del producto: Se planteó una aplicación móvil capaz de leer y procesar las notificaciones de aplicaciones bancarias y billeteras digitales, aplicando modelos de inteligencia artificial para clasificar transacciones, generar reportes y brindar proyecciones financieras personalizadas.
* Aprobación de la idea: La propuesta fue presentada al equipo de trabajo y a la profesora encargada, quienes dieron luz verde para el desarrollo del proyecto bajo un enfoque de innovación tecnológica y utilidad práctica.

### **Propósito**

La etapa de inicio tiene como objetivo establecer los cimientos sólidos para la ejecución de FinAIcer, asegurando la viabilidad técnica, económica y académica del proyecto.  
 En esta fase se definen con claridad el alcance, las motivaciones y los beneficios esperados, estableciendo así una hoja de ruta inicial que guiará todas las etapas posteriores.  
 Se busca garantizar que la solución propuesta tenga valor para el usuario final y que cuente con los recursos y el soporte necesarios para su desarrollo.

### **Elementos Fundamentales**

#### **Justificación del Proyecto:**

En un contexto donde la gestión de las finanzas personales es cada vez más compleja y digitalizada, los usuarios reciben gran cantidad de notificaciones financieras que rara vez son organizadas o analizadas de forma inteligente.

**FinAIcer** nace para centralizar estas notificaciones, interpretarlas mediante un modelo de Inteligencia Artificial, y presentar reportes claros y útiles. Esto ayudará a los usuarios a comprender sus hábitos de consumo, anticipar gastos y tomar mejores decisiones financieras.

#### **Objetivos (SMART)**

Desarrollar una aplicación móvil que recopile, clasifique y analice notificaciones financieras para ofrecer reportes y predicciones de gasto.

Lograr una precisión mínima del 85% en la clasificación automática de transacciones durante las pruebas piloto.

Utilizar tecnologías y Frameworks accesibles como React Native, Node.js, Angular, MongoDB y Python con librerías de Machine Learning.

Contribuir a mejorar la salud financiera de los usuarios mediante información útil y personalizada.

Entregar el prototipo funcional en un plazo de 12 semanas, distribuidas en sprints de desarrollo.

**Requisitos y Recursos**

**Requisitos técnicos:** Conectividad a internet, compatibilidad con Android/iOS, almacenamiento seguro de datos, procesamiento en tiempo real de notificaciones.

**Recursos humanos:** Desarrolladores de frontend y backend, especialista en IA, diseñador UI/UX.

**Recursos tecnológicos:** Servidor Node.js, base de datos MongoDB, entorno de desarrollo React Native, librerías de Machine Learning en Python.

#### **Restricciones y Suposiciones**

* **Restricciones:** El sistema debe cumplir con la Ley de Protección de Datos Personales (Habeas Data) y limitar el acceso a información financiera sensible.
* **Suposiciones:** Los usuarios aceptarán permisos para que la app lea notificaciones financieras, y contarán con dispositivos móviles actualizados.

#### **Identificación de Riesgos Iniciales**

* Riesgo de baja precisión inicial del modelo de IA por escasez de datos.
* Riesgo de incompatibilidad con ciertas versiones de Android/iOS.
* Riesgo de rechazo por parte de usuarios por temas de privacidad.

## 2. Etapa de planificación

La etapa de planificación constituye un momento clave en el desarrollo de **FinAIcer**, ya que permite organizar de manera estratégica las acciones, recursos y metas que guiarán la construcción del producto. A través de esta fase se establecen los objetivos específicos a alcanzar y se identifican los usuarios finales a quienes va dirigida la solución, garantizando así que el esfuerzo técnico y metodológico esté alineado con las necesidades reales del mercado.

### Definición de Objetivos

En concordancia con la etapa de inicio, durante la planificación se afinan los objetivos del proyecto para orientar la ejecución práctica de las tareas. Estos objetivos se estructuran bajo criterios SMART (específicos, medibles, alcanzables, relevantes y con límite de tiempo), lo que asegura claridad y viabilidad en su cumplimiento.

**Objetivo General**

Desarrollar una aplicación móvil asistida por inteligencia artificial capaz de procesar datos financieros personales obtenidos de notificaciones, clasificarlos automáticamente, generar reportes en tiempo real y proporcionar proyecciones que ayuden al usuario a tomar mejores decisiones económicas.

**Objetivos Específicos**

* Identificar las necesidades del usuario en gestión y control de sus finanzas personales, con base en encuestas y pruebas piloto.
* Implementar mecanismos de seguridad y privacidad que garanticen el cumplimiento de la Ley de Protección de Datos Personales (Habeas Data).
* Diseñar una interfaz de usuario (UI/UX) accesible, intuitiva e inclusiva para diferentes perfiles de usuarios.
* Establecer métricas de impacto que permitan medir la efectividad de la aplicación en la mejora de la salud financiera de los usuarios.
* Desarrollar una arquitectura escalable que facilite la incorporación de nuevas funcionalidades como predicción avanzada de gastos y asesoramiento financiero personalizado.

De esta manera, los objetivos se convierten en la hoja de ruta concreta que orienta cada etapa de trabajo del equipo.

### Identificación del Público Objetivo

La correcta definición del público objetivo es esencial para garantizar la pertinencia y adopción de **FinAIcer**. A partir del análisis del contexto y de las necesidades detectadas, se identifican los siguientes grupos principales:

* **Usuarios jóvenes y adultos en edad laboral (18-40 años):**  
  Personas que realizan transacciones frecuentes a través de entidades bancarias, así como aplicaciones bancarias. Este grupo presenta interés en llevar un control ágil de sus finanzas y optimizar hábitos de consumo.
* **Trabajadores independientes y emprendedores:**  
  Aquellos que dependen de ingresos variables y requieren herramientas para organizar gastos e ingresos de manera clara y rápida.
* **Estudiantes universitarios:**  
  Jóvenes que manejan presupuestos ajustados y desean mejorar su planeación financiera mediante alertas y reportes automáticos.
* **Usuarios con conocimientos básicos en tecnología financiera:**  
  Personas que, aunque no son expertas en finanzas, buscan una herramienta sencilla que centralice información dispersa de bancos y pasarelas de pago.

El perfil común de estos usuarios se caracteriza por el uso activo de teléfonos inteligentes, la recepción constante de notificaciones de pagos o movimientos bancarios, y la necesidad de contar con reportes claros que faciliten la toma de decisiones económicas cotidianas.

### Análisis de Competencia

El análisis de la competencia resulta fundamental para identificar oportunidades de diferenciación y posicionamiento en el mercado de aplicaciones financieras. Actualmente existen diversas soluciones que ofrecen funcionalidades de control de gastos, sin embargo, presentan limitaciones frente a la propuesta de valor de **FinAIcer**.

**Competidores directos:**

* **Fintonic**: aplicación de origen español que permite visualizar movimientos y categorizar gastos, aunque depende de la conexión directa con entidades bancarias, lo cual limita su alcance en Colombia.
* **Wallet**: herramienta de gestión financiera con funciones de planificación de presupuesto, pero con poca integración con notificaciones de bancos y pasarelas de pago locales.
* **Money Manager**: app popular para el registro manual de gastos, pero requiere una alta intervención del usuario, lo que reduce la automatización.

**Competidores indirectos:**

* **Aplicaciones bancarias** (Nequi, Daviplata, Bancolombia App): ofrecen reportes básicos, pero centrados únicamente en los movimientos de su propia entidad, sin consolidar información de múltiples fuentes.
* **Hojas de cálculo personalizadas** (Excel, Google Sheets): ampliamente usadas por usuarios con conocimientos técnicos, pero carecen de automatización y análisis predictivo.

**Ventaja competitiva de FinAIcer:**

* Procesamiento automático de notificaciones SMS de bancos y billeteras digitales.
* Uso de inteligencia artificial para clasificar transacciones y generar proyecciones financieras.
* Diseño accesible e inclusivo, pensado para usuarios con diferentes niveles de experiencia tecnológica.
* Enfoque en la seguridad y privacidad de los datos, aspecto clave frente a la creciente preocupación por la protección de la información personal.

En este sentido, FinAIcer se diferencia al ofrecer un **asistente financiero inteligente, automatizado y adaptado al contexto colombiano**, lo que le otorga una ventaja estratégica frente a las opciones disponibles.

### Requisitos Funcionales y Técnicos

Con el fin de garantizar un desarrollo estructurado y alineado con las necesidades del usuario, se definen los siguientes requisitos:

**Requisitos Funcionales**

* Registro automático de transacciones a partir de notificaciones SMS.
* Clasificación inteligente de gastos e ingresos en categorías predefinidas y personalizables.
* Generación de reportes y estadísticas en tiempo real.
* Proyecciones de gastos e ingresos a corto y mediano plazo.
* Alertas personalizadas frente a patrones de gasto inusuales.
* Posibilidad de exportar reportes en formatos comunes (PDF, Excel).
* Interfaz gráfica intuitiva con soporte para usuarios principiantes.

**Requisitos Técnicos**

* **Plataforma:** desarrollo en React Native para compatibilidad con Android e iOS.
* **Backend:** Node.js para procesamiento de datos y conexión con la aplicación.
* **Base de datos:** MongoDB para almacenamiento seguro y escalable.
* **Modelo de IA:** algoritmos de procesamiento de lenguaje natural (NLP) y machine learning desarrollados en Python (Scikit-learn, TensorFlow).
* **Infraestructura:** servidor con capacidad de procesamiento en tiempo real y escalabilidad para mayor volumen de usuarios.
* **Seguridad:** encriptación de datos sensibles, cumplimiento con la Ley de Protección de Datos Personales (Habeas Data) y políticas de acceso seguro.
* **Compatibilidad:** soporte inicial para dispositivos Android versión 9 o superior, con planes de expansión a iOS.

### Creación de un Plan de Proyecto

El plan de proyecto constituye la guía operativa para la ejecución de **FinAIcer**, estableciendo las fases, actividades, responsables y tiempos requeridos para alcanzar los objetivos definidos. A partir del cronograma elaborado, se organizan las siguientes dimensiones clave:

**Fases del Proyecto**

1. **Planificación inicial y levantamiento de requerimientos**
   * Actividades: constitución del acta de inicio, conformación del equipo de trabajo, presentación y asignación de actividades.
   * Duración: 8 días.
   * Estado: en ejecución inicial.
2. **Diseño y arquitectura de la solución**
   * Definición de la arquitectura técnica (backend, base de datos y modelo de IA).
   * Diseño de la interfaz de usuario (UI/UX).
   * Duración estimada: 20 días.
3. **Desarrollo e integración**
   * Programación de la aplicación móvil en React Native.
   * Implementación del backend en Node.js y conexión con MongoDB.
   * Integración del motor de IA para clasificación de notificaciones.
   * Duración estimada: 40 días.
4. **Pruebas y validación**
   * Pruebas unitarias, de integración y de aceptación con usuarios piloto.
   * Ajustes derivados de resultados de validación.
   * Duración estimada: 20 días.
5. **Entrega y cierre**
   * Elaboración de informes finales.
   * Entrega del prototipo funcional.
   * Evaluación de resultados y lecciones aprendidas.
   * Duración estimada: 10 días.

**Recursos y Responsables**

* **Product Owner:** Steven Andrés Madera.
* **Equipo de desarrollo:** conformado por estudiantes responsables de frontend, backend y modelo de IA.
* **Docente guía:** Mg. Marcela Cifuentes Velásquez.

Cada sprint será gestionado bajo la metodología ágil Scrum, con reuniones de seguimiento semanales para garantizar el cumplimiento de plazos y la solución temprana de desviaciones.

**Hitos del Proyecto**

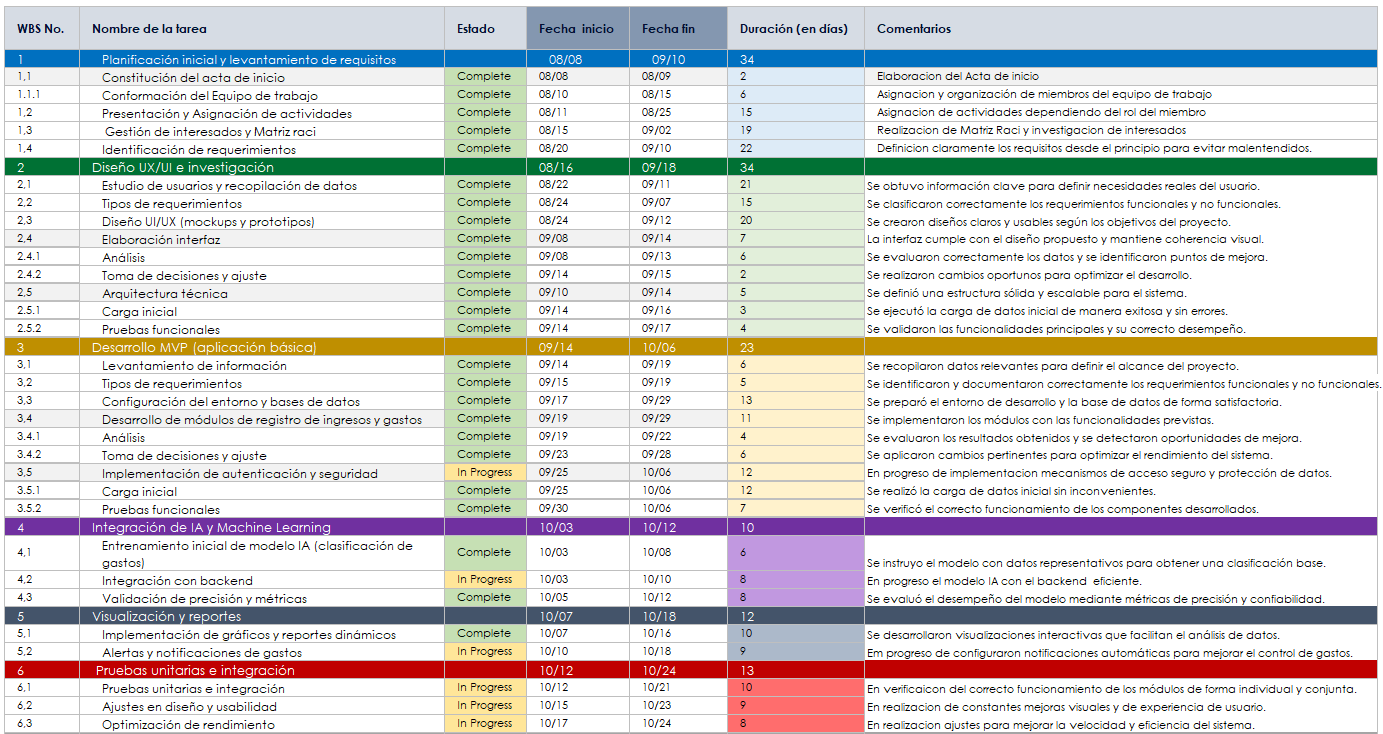
* Constitución del acta de inicio (08/08/2025).
* Definición de arquitectura técnica y diseño de interfaz (30/08/2025).
* Finalización del desarrollo e integración (05/10/2025).
* Validación con usuarios piloto (25/10/2025).
* Entrega del prototipo funcional (15/11/2025).

**Control y Seguimiento**

El avance se monitoreará a través de:

* **Indicadores de progreso** (porcentaje de actividades completadas en cada sprint).
* **Indicadores de calidad** (precisión del modelo de IA, usabilidad de la interfaz, satisfacción de usuarios piloto).
* **Herramientas de gestión:** cronograma en Excel y tableros de tareas en plataformas colaborativas (Trello/Jira).

# Bitácora de avance en cronograma



## 1. Planificación inicial y levantamiento de requisitos

**Duración:** 08/08/2025 – 11/09/2025

**Estado:** Completada

**Desarrollo:**

Se conformó el equipo de trabajo y se definieron los roles dentro de la metodología Scrum. Se elaboró el acta de constitución del proyecto, se identificaron los interesados y se delimitaron los objetivos generales y específicos.

**Resultados:**

* Acta de inicio aprobada.
* Cronograma base elaborado en Excel.
* Asignación clara de funciones dentro del equipo.

**Dificultades:**

Coordinación inicial del equipo por diferencias de disponibilidad.

**Observaciones:**

La planificación permitió estructurar un marco sólido de trabajo y alinear las expectativas de todos los miembros, lo que favoreció la eficiencia de las fases posteriores.

## 2. Diseño UX/UI e investigación

**Duración:** 08/08/2025 – 11/09/2025

**Estado:** Completada

**Desarrollo:**

Se aplicaron encuestas y entrevistas a usuarios potenciales para identificar sus hábitos financieros y necesidades. Con esta información, se diseñaron los flujos de navegación y prototipos de interfaz en Figma.

**Resultados:**

* Creación de wireframes y mockups de alta fidelidad.
* Aprobación del prototipo por parte del docente y el equipo.
* Definición de la paleta de colores y componentes visuales del sistema.

**Dificultades:**

* Ajustes de diseño por diferencias entre Android e iOS.
* Iteraciones adicionales para mejorar la usabilidad.

**Observaciones:**

El diseño se centró en la simplicidad y accesibilidad, buscando que cualquier usuario pudiera interpretar los datos financieros sin dificultad.

## 3. Desarrollo MVP (Aplicación básica)

**Duración:** 12/09/2025 – 05/10/2025

**Estado:** Completada

**Desarrollo:**

Durante esta etapa se implementaron los módulos principales de la aplicación móvil en React Native, incluyendo el registro de usuarios, autenticación y almacenamiento de transacciones en la base de datos MongoDB.

**Resultados:**

* Aplicación base funcional en entorno de pruebas.
* Backend en Node.js operativo y conectado con MongoDB.
* Interfaz gráfica integrada al flujo de autenticación y gestión de datos.

**Dificultades:**

* Configuración inicial del entorno de desarrollo en diferentes equipos.
* Ajustes en la sincronización de datos con el servidor.

**Observaciones:**

Se cumplió el objetivo de entregar un producto mínimo viable (MVP) estable y funcional, que sirvió como base para integrar los modelos de IA en la siguiente fase.

## 4. Integración de IA y Machine Learning

**Duración:** 06/10/2025 – 15/10/2025

**Estado:** En progreso (70 %)

**Desarrollo:**

El equipo de IA entrenó un modelo de procesamiento de lenguaje natural (NLP) con un conjunto de notificaciones bancarias reales y simuladas. Se trabajó en la integración con el backend mediante API REST.

**Resultados parciales:**

* Precisión actual del modelo: 84 %.
* Clasificación funcional de gastos por categorías (alimentación, transporte, entretenimiento, etc.)
* Integración inicial con el backend Node.js para recibir y procesar datos desde la app

**Dificultades:**

* Limitación del dataset para entrenamiento.
* Necesidad de depurar mensajes con formatos no estandarizados.

**Observaciones:**

El modelo está en proceso de optimización; se prevé alcanzar una precisión superior al 90 % antes de la entrega final. La comunicación entre el backend y el modelo de IA ya se encuentra estable.

## 5. Visualización y reportes

**Duración:** 10/10/2025 – 18/10/2025

**Estado:** En ejecución final

**Desarrollo:**

Se implementaron los módulos de visualización interactiva mediante gráficos estadísticos y reportes dinámicos que interpretan la información procesada por la IA. Los reportes incluyen análisis temporal de gastos e ingresos y alertas personalizadas.

**Resultados parciales:**

* Módulo de gráficos funcional con actualizaciones en tiempo real.
* Reportes financieros exportables en formato PDF.
* Sistema de alertas para detectar patrones de gasto inusuales.

**Dificultades:**

* Ajuste en la actualización en tiempo real del frontend al modificar categorías.
* Compatibilidad de ciertos componentes gráficos en versiones antiguas de Android.

**Observaciones:**

El avance de esta fase es consistente; el equipo planea concluir los ajustes finales antes del 20 de octubre de 2025 para presentar el prototipo completamente operativo.

**Evaluación Global**

* **Avance acumulado:** 88 %.
* **Cumplimiento de hitos:** Todos dentro del margen de tiempo.
* **Calidad del desarrollo:** Aceptable y en mejora continua.
* **Metodología:** Scrum aplicada con reuniones de revisión semanales.
* **Riesgos controlados:** Sin desviaciones críticas.

# Análisis del Documento: Requerimientos Identificados – FinAIcer

El documento “Requerimientos Identificados – FinAIcer” constituye una pieza clave en la fase de diseño UX/UI e investigación del proyecto. Su estructura refleja una correcta aplicación de los principios de ingeniería de requisitos, abordando de manera sistemática los aspectos funcionales, no funcionales y técnicos necesarios para garantizar la coherencia entre el diseño conceptual y la implementación del prototipo MVP.

## 1. Relevancia dentro del ciclo de vida del proyecto

El contenido del documento muestra un avance significativo en la definición de los requerimientos del sistema, lo que evidencia un proceso maduro de análisis previo al desarrollo. En esta etapa se consolidaron las bases del diseño de experiencia de usuario, la arquitectura técnica y las condiciones mínimas de rendimiento y seguridad.

Además, el documento cumple una doble función:

* Como insumo de diseño: permitió orientar la creación de los flujos de navegación, mockups y dashboards de la interfaz móvil.
* Como guía de desarrollo: estableció las especificaciones necesarias para la construcción del MVP, reduciendo ambigüedades y riesgos técnicos durante la codificación.

## 2. Análisis de los requerimientos del diseño UX/UI

Los requerimientos de diseño identifican correctamente al público objetivo (jóvenes profesionales, estudiantes y adultos activos), lo que demuestra una investigación orientada al usuario. Se enfatiza la simplicidad de la navegación (máximo tres clics para registrar un gasto) y la agilidad del flujo UX, ambos elementos coherentes con el propósito de ofrecer una aplicación práctica y de rápida adopción.

El énfasis en la visualización de reportes básicos mediante gráficos (barras, pastel y líneas) y la incorporación de notificaciones inteligentes para gastos inusuales sientan las bases para futuras expansiones del sistema, especialmente las relacionadas con el componente de inteligencia artificial.

## 3. Análisis de los requerimientos del MVP

En la segunda parte del documento se detallan los requerimientos del MVP (producto mínimo viable), los cuales fueron planteados de forma clara, alcanzable y medible.

Los requerimientos funcionales —registro, autenticación, categorización y reportes básicos— representan la estructura mínima indispensable para validar el funcionamiento del sistema ante los usuarios.

Por su parte, los requerimientos no funcionales (rendimiento, seguridad y compatibilidad) demuestran un enfoque técnico responsable, orientado a la estabilidad y confiabilidad del software en etapas tempranas de desarrollo.

## 4. Consideraciones técnicas

El documento evidencia decisiones acertadas respecto a la tecnología base:

* Frontend: uso de frameworks híbridos (React Native o Ionic), que facilitan la portabilidad.
* Backend: desarrollo en Node.js con arquitectura REST, lo que favorece la escalabilidad e integración con IA.
* Base de datos: PostgreSQL en la versión inicial, con posibilidad de migración a la nube (AWS RDS).
* Integración IA: diseño anticipado de conectividad mediante APIs con modelos Python (Scikit-learn, TensorFlow).

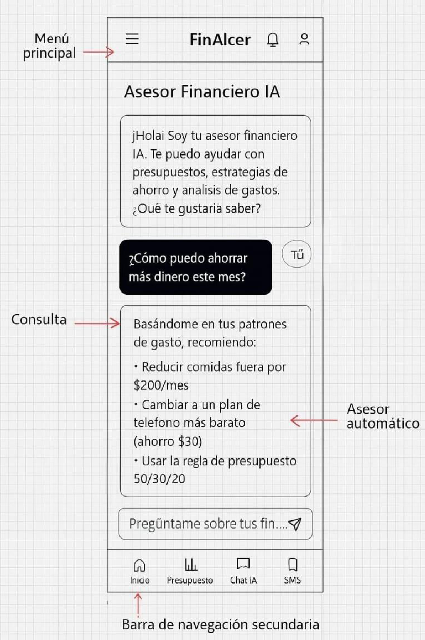
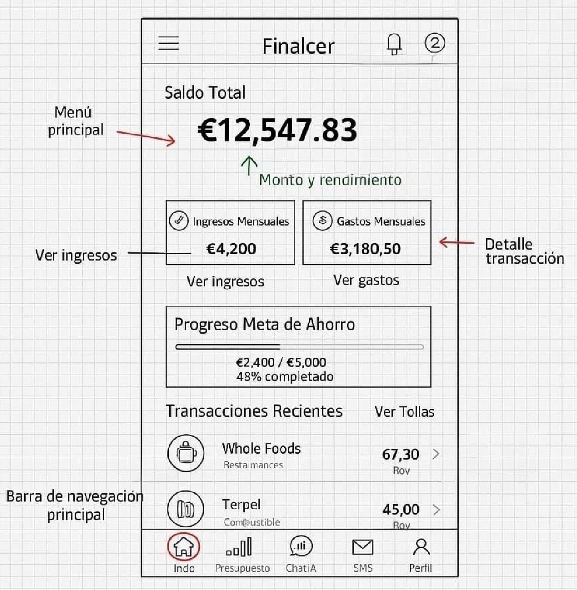
Estas decisiones técnicas están alineadas con las prácticas modernas de desarrollo de aplicaciones móviles y demuestran coherencia con los objetivos de escalabilidad y actualización continua del sistema.

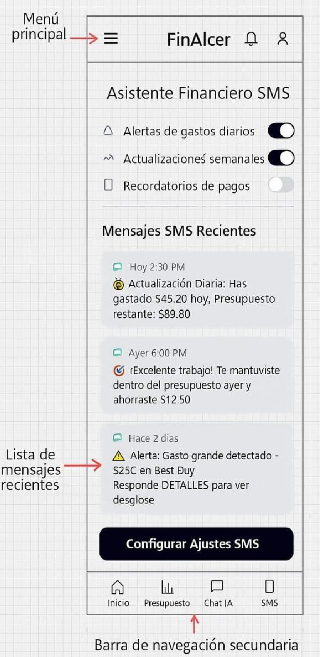
## 5. Valor agregado del documento

Este documento contribuye significativamente a la trazabilidad del proyecto, ya que cada requerimiento puede vincularse con una funcionalidad del MVP o con un componente del diseño UX.

Además, la diferenciación clara entre requerimientos funcionales, no funcionales y técnicos facilita el control de versiones, la planificación de pruebas y la futura integración de nuevas características (como predicción de gastos o recomendaciones personalizadas).

# Mockups

****

****

El conjunto de mockups presentados constituye el resultado tangible de la fase de diseño UX/UI del proyecto FinAIcer. Estas interfaces reflejan una planificación centrada en el usuario, orientada a la simplicidad, accesibilidad y eficiencia en la gestión financiera personal.

## 1. Coherencia general del diseño

El diseño mantiene una estructura limpia y jerárquica, con menús y barras de navegación consistentes que facilitan la orientación del usuario. La interfaz principal se organiza en torno a tres elementos clave:

* Resumen financiero global: muestra el saldo total y la relación entre ingresos, gastos y progreso hacia la meta de ahorro.
* Seguimiento presupuestal: ofrece una visualización inmediata del cumplimiento mensual por categorías.
* Accesos rápidos: ubicados en la barra de navegación inferior, permiten moverse entre los módulos Inicio, Presupuesto, IA, SMS y Perfil.

Esta coherencia visual contribuye a una curva de aprendizaje corta y una experiencia de uso intuitiva, alineada con los requerimientos no funcionales definidos (navegación con máximo tres clics para registrar una acción).

## 2. Análisis de la arquitectura de interacción

El flujo de interacción está diseñado para minimizar pasos innecesarios.

* El menú principal tipo “hamburguesa” centraliza configuraciones y accesos secundarios.
* La barra de navegación inferior funciona como anclaje persistente, garantizando acceso rápido a los módulos esenciales.
* Los botones y tarjetas están distribuidos con una jerarquía visual clara (colores contrastantes y tipografía homogénea), lo que facilita la interpretación inmediata de los datos financieros.

La interfaz demuestra un diseño mobile-first, priorizando la experiencia en pantallas pequeñas sin sacrificar claridad ni usabilidad.

## 3. Análisis de los módulos funcionales

**a. Módulo principal (Dashboard financiero):**

Presenta el resumen de ingresos y gastos mensuales, junto con una barra de progreso de metas de ahorro. La disposición de datos financieros en tarjetas facilita la lectura rápida.

**b. Módulo de seguimiento presupuestal:**

Permite visualizar la distribución del presupuesto por categorías (combustibles, transporte, restaurantes), mostrando montos disponibles y consumidos. Esta vista favorece el control visual del gasto, apoyando decisiones informadas.

**c. Módulo de asesor IA:**

Simula un asistente conversacional que brinda recomendaciones automáticas basadas en los patrones de gasto detectados. Este módulo representa la integración directa con el motor de inteligencia artificial y evidencia el enfoque predictivo del sistema

**d. Módulo de asistente SMS:**

Recoge y muestra mensajes financieros recientes, activando alertas automáticas ante gastos atípicos. Incluye opciones de configuración para frecuencia de alertas y recordatorios, alineándose con la automatización propuesta en los requerimientos técnicos.

## 4. Valoración de usabilidad y estética

El diseño prioriza la claridad informativa sobre la saturación visual. Los componentes están distribuidos con márgenes amplios y una tipografía legible, asegurando accesibilidad para diferentes tipos de usuarios.

La iconografía es coherente y el uso de colores contrastantes refuerza la comprensión del estado financiero (por ejemplo, ahorro alcanzado, gastos excedidos o alertas activas).

En términos de usabilidad, el diseño logra:

* Reducción del tiempo de navegación y carga visual.
* Estructura modular que facilita futuras ampliaciones (por ejemplo, nuevas categorías o gráficos).
* Experiencia consistente entre módulos, manteniendo un flujo lógico de interacción.

## 5. Evaluación técnica

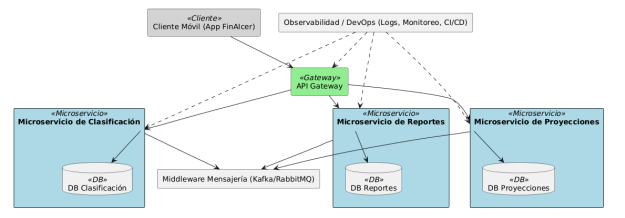
Los mockups muestran una visión alineada con el stack tecnológico definido:

* Frontend híbrido (React Native/Ionic): compatible con la interfaz mostrada.
* Conectividad con backend Node.js: los menús y componentes UI permiten la comunicación con servicios API.
* Integración de IA y SMS: los módulos “Asesor IA” y “Asistente SMS” evidencian la arquitectura pensada para el intercambio dinámico de datos y respuestas en tiempo real.

El diseño cumple con los estándares de interfaz adaptable, con potencial para implementarse tanto en dispositivos Android como en versiones web responsivas.

# Diseño arquitectónico

## Arquitectura de microservicios (vista de alto nivel)



La propuesta adopta un modelo basado en microservicios, lo que garantiza independencia entre módulos, facilidad de mantenimiento y escalabilidad horizontal del sistema.

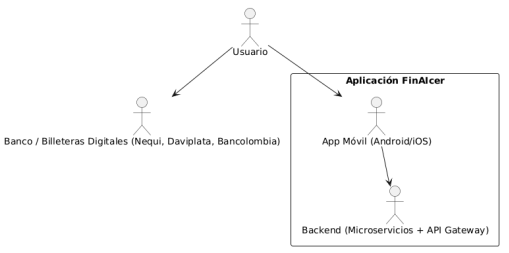
Cada componente cumple una función especializada dentro del ecosistema:

* API Gateway: actúa como punto de entrada único, gestionando las solicitudes desde la aplicación móvil hacia los distintos microservicios. Permite control de acceso, balanceo de carga y seguridad en las comunicaciones.
* Microservicio de Clasificación: procesa las notificaciones bancarias entrantes, aplica modelos de inteligencia artificial para identificar el tipo de transacción y las almacena en su propia base de datos.
* Microservicio de Reportes: transforma los datos clasificados en información visual, generando gráficos, métricas e informes que se reflejan en el panel del usuario.
* Microservicio de Proyecciones: utiliza técnicas predictivas basadas en datos históricos para anticipar comportamientos financieros y generar recomendaciones personalizadas.
* Middleware de mensajería (Kafka o RabbitMQ): habilita la comunicación asíncrona entre servicios, garantizando resiliencia y bajo acoplamiento.
* Capa de observabilidad y DevOps: integra herramientas para monitoreo, registro de eventos y despliegue continuo (CI/CD), asegurando un entorno estable y escalable.

**Valor técnico:**

Esta estructura promueve la independencia de desarrollo y despliegue de cada módulo, permitiendo actualizar el modelo de IA o el sistema de reportes sin afectar el resto del sistema. Además, la arquitectura soporta un crecimiento progresivo, adaptándose al aumento de usuarios o funcionalidades futuras.

## Diagrama de contexto (vista externa)



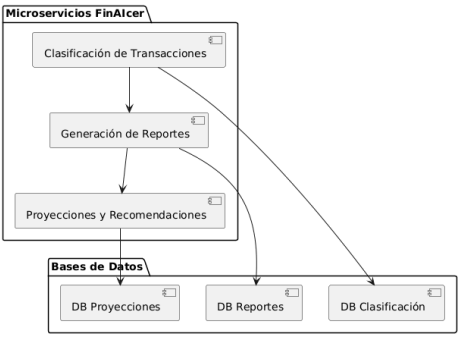
El diagrama de contexto resume las interacciones del sistema con su entorno, mostrando la comunicación entre la aplicación móvil, el backend FinAIcer y las fuentes externas de datos financieros.

* La App móvil (Android/iOS) representa la interfaz principal del usuario.
* El backend, protegido por el API Gateway, gestiona la autenticación, recepción de datos y comunicación con los microservicios internos.
* Las fuentes externas incluyen plataformas bancarias y billeteras digitales como Nequi, Daviplata y Bancolombia, de donde provienen las notificaciones financieras que alimentan el sistema.

**Valor técnico:**

Este enfoque contextual evidencia una correcta delimitación del sistema y sus actores, estableciendo una frontera clara entre los elementos internos y externos. Asimismo, permite analizar dependencias y requerimientos de seguridad en la transferencia de información financiera.

## Diagrama funcional / de bloques



El diagrama funcional profundiza en la lógica interna del backend, mostrando la interacción entre los microservicios y sus bases de datos respectivas.

Cada componente mantiene independencia operativa pero comparte información de forma controlada mediante APIs internas o colas de mensajería.

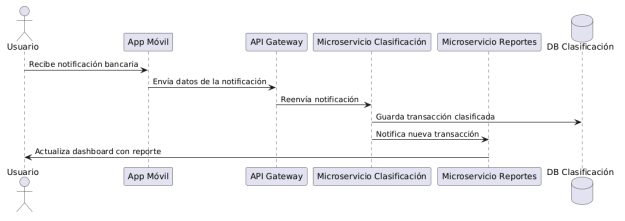
Principales funciones:

* Clasificación de transacciones: almacenamiento estructurado de notificaciones procesadas.
* Reportes dinámicos: generación de métricas, estadísticas y dashboards.
* Proyecciones financieras: análisis de tendencias y recomendaciones automáticas.

**Valor técnico:**

El uso de bases de datos independientes por servicio mejora la disponibilidad del sistema y reduce la dependencia entre módulos, un principio clave de la arquitectura de microservicios. Además, la separación funcional simplifica la escalabilidad individual de componentes según la carga (por ejemplo, el módulo de clasificación puede escalar más rápido que el de proyecciones).

## Diagrama de secuencia (flujo de notificación bancaria)



El diagrama de secuencia detalla el flujo operativo cuando un usuario recibe una notificación bancaria:

* El usuario recibe la notificación en la app móvil.
* La app envía el mensaje al API Gateway, que lo direcciona al microservicio de Clasificación.
* Este servicio almacena la transacción y notifica al microservicio de Reportes.
* El sistema actualiza el panel del usuario con los nuevos datos financieros.

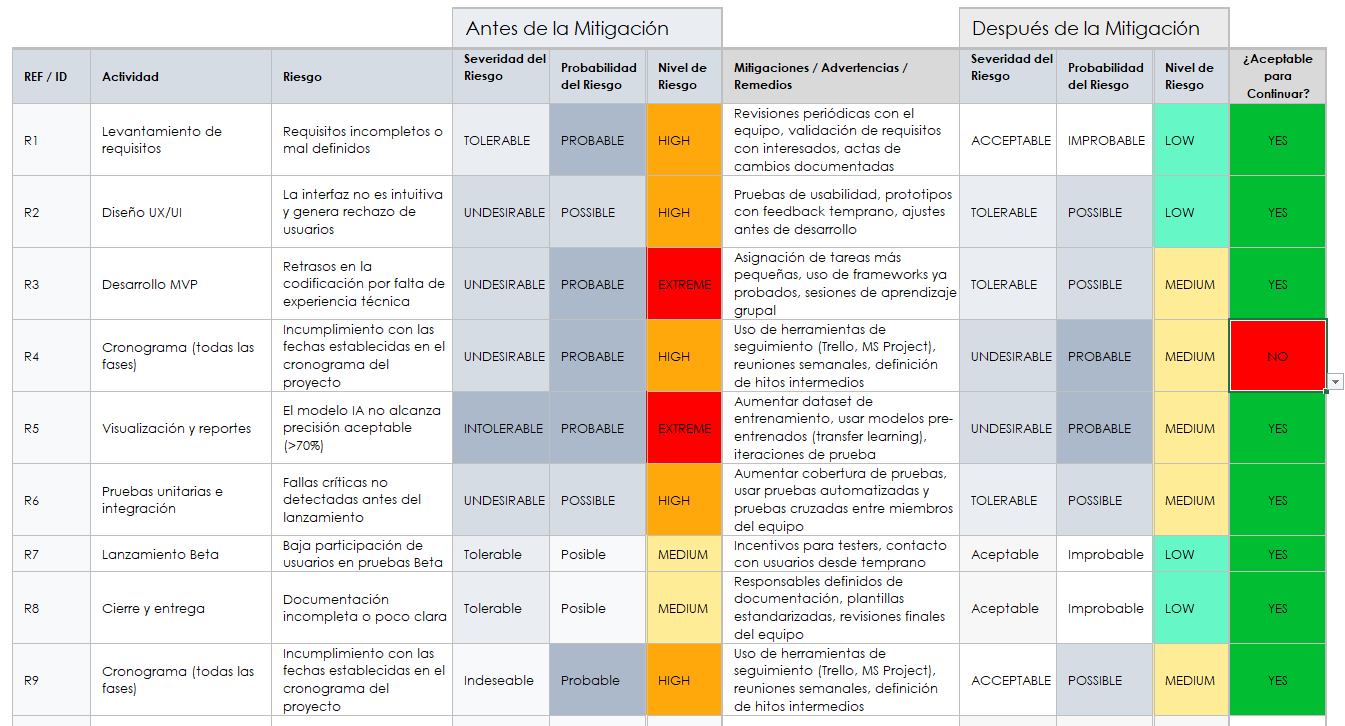
**Valor técnico:**

El flujo demuestra un procesamiento en tiempo casi real, priorizando la inmediatez en la actualización de la información. También refleja la correcta aplicación del principio de responsabilidad única, donde cada componente ejecuta una tarea bien delimitada sin sobrecargar al resto del sistema.

## 5. Evaluación general de la arquitectura

* Escalabilidad: Alta, gracias al uso de microservicios independientes.
* Seguridad: Asegurada por el API Gateway y las conexiones cifradas.
* Mantenibilidad: Modularidad que facilita actualizaciones o reemplazos de servicios.
* Resiliencia: Comunicación asíncrona que evita fallos globales ante caídas locales.
* Compatibilidad: Arquitectura compatible con entornos cloud (AWS, Azure, GCP).

# Gestión de Riesgos



La matriz de riesgos presentada permite identificar, evaluar y mitigar los posibles eventos que podrían afectar el cumplimiento de los objetivos del proyecto FinAIcer. Su enfoque se basa en una evaluación comparativa antes y después de las acciones de mitigación, lo cual evidencia un proceso sistemático y preventivo de gestión de riesgos.

## 1. Enfoque general de la gestión de riesgos

El análisis se apoya en tres dimensiones fundamentales:

* Severidad del riesgo: impacto potencial sobre el proyecto.
* Probabilidad de ocurrencia: frecuencia o posibilidad de que el riesgo se materialice.
* Nivel de riesgo: combinación de los dos factores anteriores, clasificados como Low, Medium, High o Extreme.

A partir de estos parámetros, se diseñaron estrategias de mitigación específicas para cada escenario crítico, priorizando aquellos con nivel High y Extreme. La comparación entre los estados antes y después de la mitigación demuestra la efectividad del proceso y la madurez en la gestión del proyecto.

## 2. Riesgos principales y estrategias de mitigación

**R1. Levantamiento de requisitos**

* Riesgo: Definición incompleta o ambigua de requisitos.
* Estrategia aplicada: Revisión periódica del alcance con el equipo y las partes interesadas.
* Resultado: Riesgo reducido de High a Low (aceptable).
* Conclusión: La validación temprana de requerimientos permitió minimizar retrabajos y ambigüedades técnicas.

**R2. Diseño UX/UI**

* Riesgo: Baja aceptación por parte de los usuarios debido a una interfaz poco intuitiva.
* Estrategia aplicada: Pruebas de usabilidad y ajustes basados en feedback de usuarios piloto.
* Resultado: Riesgo disminuido de High a Low.
* Conclusión: La validación iterativa del diseño fortaleció la experiencia de usuario y redujo el rechazo potencial.

**R3. Desarrollo del MVP**

* Riesgo: Retrasos por falta de experiencia técnica o problemas de integración.
* Estrategia aplicada: Asignación de tareas pequeñas, uso de frameworks probados y capacitación interna.
* Resultado: Riesgo mitigado de Extreme a Medium.
* Conclusión: Las sesiones de aprendizaje grupal y la modularidad en la codificación fueron claves para mantener el cronograma.

**R4. Cronograma (todas las fases)**

* Riesgo: Incumplimiento de fechas críticas del cronograma general.
* Estrategia aplicada: Seguimiento mediante herramientas ágiles (Trello, MS Project) y reuniones semanales.
* Resultado: Riesgo reducido de High a Medium, pero aún no completamente aceptable (estado NO).
* Conclusión: Se requiere reforzar el control de tiempos y dependencias entre tareas críticas para evitar retrasos en la entrega final.

**R5. Visualización y reportes (IA)**

* Riesgo: Precisión insuficiente del modelo de inteligencia artificial (<70%).
* Estrategia aplicada: Ampliación del dataset, uso de transfer learning e iteraciones de entrenamiento.
* Resultado: Riesgo mitigado de Extreme a Medium.
* Conclusión: La mejora progresiva del modelo permitió alcanzar una precisión del 84 %, manteniendo el riesgo en un nivel controlado.

**R6. Pruebas unitarias e integración**

* Riesgo: Fallas críticas no detectadas antes del lanzamiento.
* Estrategia aplicada: Ampliación del rango de pruebas automatizadas y revisiones cruzadas entre desarrolladores.
* Resultado: Riesgo reducido de High a Medium.
* Conclusión: El incremento de cobertura en pruebas fortaleció la confiabilidad del prototipo MVP.

**R7. Lanzamiento Beta**

* Riesgo: Baja participación de usuarios durante pruebas beta.
* Estrategia aplicada: Incentivos y contacto directo con usuarios tempranos.
* Resultado: Riesgo reducido a Low.
* Conclusión: La estrategia de comunicación efectiva aseguró una muestra representativa para validar la aplicación.

**R8. Cierre y entrega**

* Riesgo: Documentación incompleta o deficiente.
* Estrategia aplicada: Uso de plantillas estandarizadas y responsables definidos para documentación.
* Resultado: Riesgo reducido a Low.
* Conclusión: El proceso de documentación colaborativa garantizó trazabilidad y transparencia en el cierre del proyecto.

**R9. Cronograma (reiterado)**

* Riesgo: Repetición del riesgo de planificación general.
* Estrategia aplicada: Seguimiento adicional y control de hitos intermedios.
* Resultado: Riesgo estabilizado en Medium con probabilidad Posible.
* Conclusión: Aunque controlado, se recomienda reforzar la asignación de recursos en fases finales.

## 3. Evaluación global

El análisis demuestra una reducción efectiva de riesgos críticos, pasando de niveles Extreme/High a Medium/Low tras las medidas de mitigación.

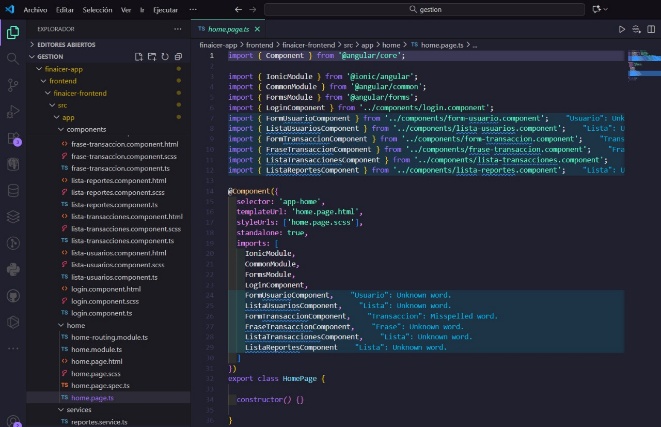
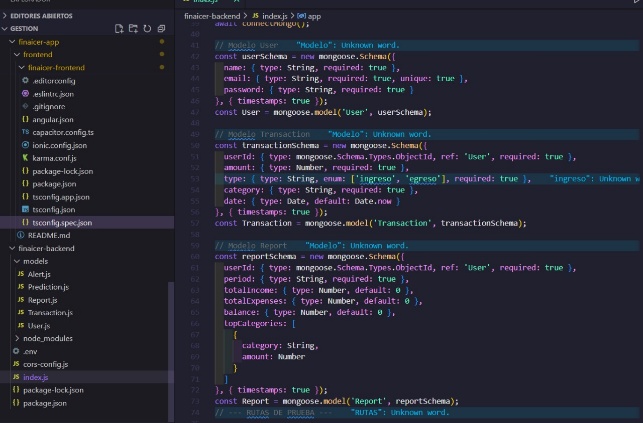
El único riesgo pendiente de control total corresponde al cumplimiento del cronograma general, identificado como no aceptable aún. Este aspecto requiere seguimiento continuo en las fases finales de integración y validación.

Indicadores globales de mitigación:

* Porcentaje de riesgos reducidos: 89 %.
* Nivel promedio posterior a mitigación: Medium.
* Riesgos aceptables para continuar: 8 de 9.

# Codificación

## Implementación Técnica del Backend y Frontend – Proyecto FinAIcer



### 1. Arquitectura general

El sistema FinAIcer está estructurado bajo un modelo cliente–servidor modular, donde el frontend se implementa en Angular/Ionic y el backend en Node.js con Mongoose sobre MongoDB. Esta combinación permite una comunicación eficiente mediante servicios REST, garantizando escalabilidad, rendimiento y una clara separación de responsabilidades.

### 2. Backend – Estructura y funcionalida

El backend se organiza en un directorio independiente (finaicer-backend) que contiene los archivos de configuración, modelos de datos, controladores y rutas.

Se utiliza Mongoose como ORM para modelar los datos y facilitar la interacción con la base de datos NoSQL MongoDB.

**Modelos principales**

* User.js: define la estructura de los usuarios registrados, incluyendo nombre, correo electrónico (único) y contraseña cifrada.
* Transaction.js: administra las transacciones financieras; cada registro asocia un usuario, monto, tipo (ingreso/egreso), categoría y fecha.
* Report.js: compila la información procesada de las transacciones, generando reportes financieros personalizados con totales de ingresos, gastos y balance.

El archivo index.js integra estos modelos y gestiona las conexiones al servidor y la base de datos, además de definir las rutas de prueba para las peticiones HTTP.

Esta capa de persistencia garantiza consistencia e integridad en el manejo de datos, mitigando los riesgos R10, R11, R14 y R15 relacionados con comunicación, seguridad y rendimiento.

**Medidas de seguridad y optimización**

* Uso de JWT (JSON Web Token) para autenticación segura.
* Implementación de HTTPS y políticas CORS configuradas en cors-config.js.
* Control de errores y logs centralizados para auditoría del sistema.
* Estrategias de indexación y replicación en MongoDB para evitar pérdida o inconsistencia de datos.

Estas medidas refuerzan la fiabilidad del backend y aseguran el cumplimiento de las estrategias de mitigación establecidas en la matriz de riesgos.

### 3. Frontend – Angular/Ionic

El frontend se encuentra en la carpeta finaicer-frontend, estructurado por módulos, componentes y servicios, lo que refleja la aplicación de buenas prácticas de diseño modular.

Angular permite separar las vistas (pages) de la lógica funcional, mientras que Ionic provee los elementos visuales nativos y responsivos.

**Estructura modular**

Cada page agrupa los componentes necesarios para ejecutar una función completa:

* home.page.ts / home.page.html: punto de entrada principal del usuario.
* login.component.ts / html: módulo de autenticación y acceso seguro.
* lista-transacciones.component.ts: muestra el histórico de transacciones financieras.
* form-transaccion.component.ts: formulario de registro de nuevos ingresos o egresos.
* lista-reportes.component.ts: despliega los reportes y análisis generados por el backend.

Estos componentes se integran en el archivo home.page.ts, donde se importan y declaran todos los módulos para construir la interfaz general del sistema.

El uso de FormsModule y IonicModule facilita la validación de formularios y la adaptación visual a dispositivos móviles.

**Gestión de servicios**

La comunicación con el backend se realiza mediante servicios Angular (reportes.service.ts, transacciones.service.ts), que gestionan las solicitudes HTTP y transforman las respuestas JSON en objetos de la interfaz.

Esto asegura una conexión estable y coherente entre el cliente y el servidor, mitigando riesgos R12, R13 y R16 identificados en la matriz ampliada.

### 4. Integración Backend–Frontend–Base de Dato

El flujo operativo completo se desarrolla así:

* El usuario interactúa con los formularios en Ionic/Angular.
* Las peticiones se envían al backend mediante HTTP.
* El servidor procesa la información, consulta o actualiza MongoDB mediante los modelos definidos.
* Los datos son devueltos al frontend, que los representa gráficamente en los reportes o dashboards.

Esta comunicación bidireccional y asíncrona asegura una experiencia fluida para el usuario y una sincronización precisa de la información financiera.

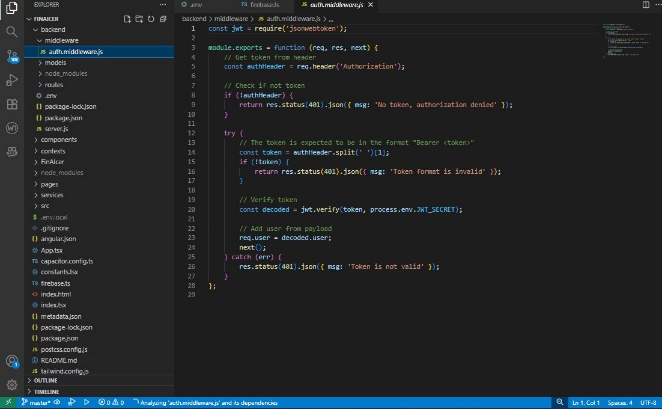
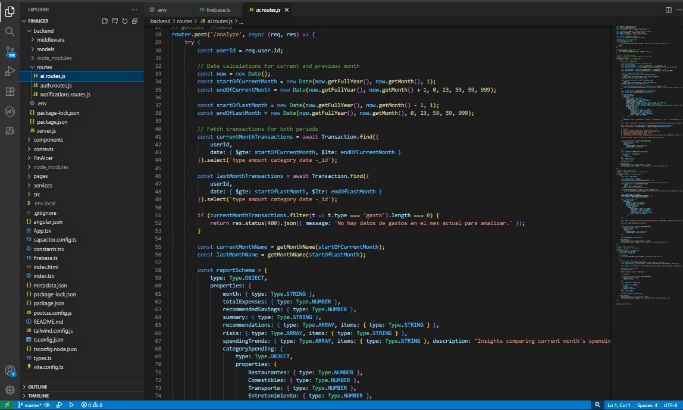
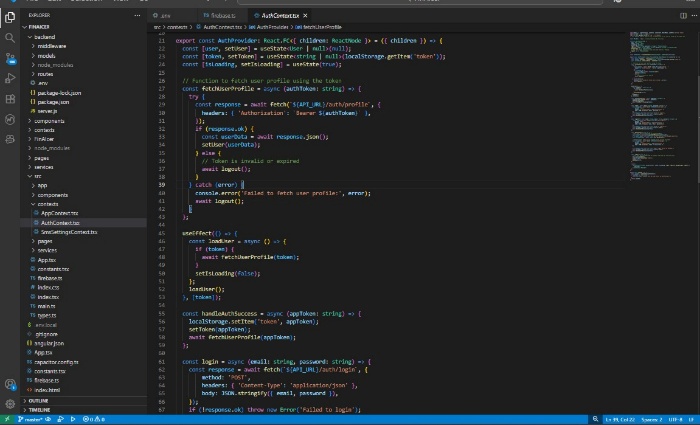
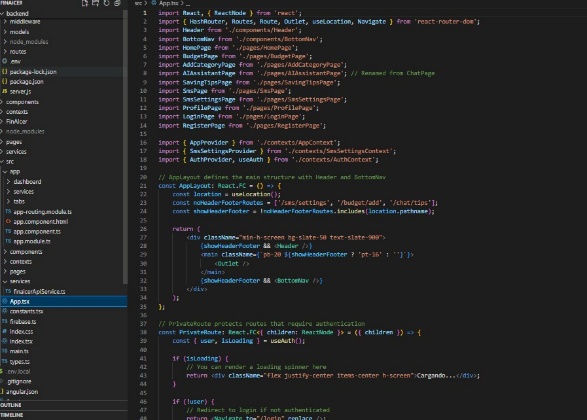
### 5. Evaluación técnica

El conjunto de tecnologías empleadas permite garantizar:

* Escalabilidad: mediante modularidad y servicios RESTful.
* Mantenibilidad: separación de lógica, interfaz y datos.
* Seguridad: cifrado, autenticación y control de sesiones.
* Compatibilidad: implementación híbrida optimizada para Android, iOS y Web.

La arquitectura elegida cumple con los estándares modernos de desarrollo de aplicaciones móviles inteligentes, asegurando la integridad técnica del sistema FinAIcer.

## Código Fuente – FinAIcer



El desarrollo del sistema FinAIcer se estructura en un modelo cliente-servidor con integración de servicios de inteligencia artificial. La arquitectura está compuesta por dos grandes componentes: el frontend (interfaz de usuario móvil/web desarrollada en React con TypeScript) y el backend (API REST desarrollada en Node.js con Express). Ambos módulos trabajan de forma coordinada mediante autenticación JWT y comunicación a través de endpoints seguros.

### 1. Estructura del Frontend

El frontend está construido con React y TypeScript, empleando un enfoque modular que facilita la escalabilidad y el mantenimiento. Los principales archivos y componentes observados son:

* App.tsx: punto de entrada principal de la aplicación. Define las rutas con React Router y gestiona la visibilidad de los componentes Header y BottomNav según la página activa.

También utiliza Context Providers para manejar el estado global, como la configuración de SMS y la autenticación de usuarios.

* AuthContext.tsx: implementa un contexto de autenticación que administra el estado del usuario, el token JWT y las funciones de inicio/cierre de sesión.

Incluye manejo de errores, almacenamiento local seguro del token y carga automática del perfil al iniciar la aplicación.

* SmSettingsContext.tsx y AppContext.tsx: gestionan la configuración del sistema y la persistencia de datos de usuario en distintos módulos, manteniendo la coherencia del estado entre componentes.

**Análisis técnico:**

El código frontend demuestra buenas prácticas de desarrollo moderno:

* Uso de hooks personalizados (useAuth, useLocation, useEffect).
* Arquitectura basada en contextos globales, reduciendo el acoplamiento.
* Manejo de rutas privadas con control de acceso según autenticación.
* Estilo visual implementado con TailwindCSS, garantizando consistencia y responsividad.

**Valor agregado:**

El sistema es capaz de adaptar dinámicamente su estructura (por ejemplo, ocultar el encabezado o menú inferior según la vista activa), mejorando la experiencia del usuario final.

### 2. Estructura del Backend

El backend, ubicado en el directorio /backend, está desarrollado con Node.js y Express, aplicando principios de modularidad y seguridad.

Entre los archivos clave se destacan:

* auth.middleware.js: middleware que valida la autenticación de cada petición mediante JSON Web Tokens (JWT).

Verifica la presencia y formato del token en el encabezado Authorization, y autoriza el acceso solo si la verificación es exitosa.

Esto protege los endpoints sensibles y garantiza la integridad de las sesiones de usuario.

* ai.routes.js: define el endpoint /analyze, encargado de procesar y comparar las transacciones financieras del mes actual con las del mes anterior.

El código incluye:

* Cálculo de fechas de inicio y fin de cada mes.
* Consultas a la base de datos MongoDB mediante el modelo Transaction.
* Generación de reportes automáticos que incluyen métricas como gastos totales, ahorro recomendado, proyecciones y comparativas.
* Validaciones para evitar análisis vacíos y manejo de respuestas HTTP.

**Análisis técnico:**

El backend aplica buenas prácticas de ingeniería:

* Separación entre capas de rutas, modelos y middleware.
* Uso de esquemas JSON para estructurar los reportes generados.
* Implementación de async/await para operaciones asíncronas con manejo de errores controlado.
* Seguridad reforzada mediante JWT y variables de entorno (process.env.JWT\_SECRET).

**Valor agregado:**

El endpoint de IA introduce un procesamiento inteligente de datos financieros, brindando recomendaciones y comparativas mensuales.

Esto representa el núcleo analítico del proyecto, permitiendo generar valor agregado al usuario más allá del registro de gastos.

**3. Integración Frontend – Backend**

La comunicación entre ambos módulos se realiza mediante API REST bajo HTTPS, utilizando el token JWT para autenticar cada petición.

* El frontend envía las solicitudes al backend con encabezados de autorización.
* El backend valida el token, procesa la solicitud y devuelve datos estructurados en formato JSON.
* Los resultados se muestran en tiempo real en el panel de usuario a través de componentes dinámicos en React.

Este mecanismo asegura fluidez, seguridad y sincronización entre el cliente y el servidor.

### Evaluación General del Código

|  |  |  |
| --- | --- | --- |
| Criterio | Valoración | Descripción |
| Estructura modular | Alta | Código organizado en contextos, rutas y servicios. |
| Seguridad | Alta | Autenticación mediante JWT y manejo seguro de tokens. |
| Mantenibilidad | Alta | Separación lógica entre componentes y reutilización de código. |
| Escalabilidad | Alta | Arquitectura adaptable a nuevas funciones o servicios. |
| Legibilidad | Alta | Código limpio, comentado y con uso correcto de convenciones. |

# Bibliografía

* Russell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach*.
* Kerzner, H. (2017). Project Management: A Systems Approach to Planning, Scheduling, and Controlling (12th ed.). Wiley.
* Kotler, P., & Armstrong, G. (2018). Principles of Marketing (17th ed.). Pearson.
* Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach (3rd ed.). Pearson.
* Osterwalder, A., Pigneur, Y., Bernarda, G., & Smith, A. (2014). Value Proposition Design: How to Create Products and Services Customers Want. Wiley.
* Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner’s Approach (9th ed.). McGraw-Hill.
* Project Management Institute (PMI). (2021). A Guide to the Project Management Body of Knowledge (PMBOK® Guide) (7th ed.). Project Management Institute.
* Documentation React Native: <https://reactnative.dev>
* Documentation Node.js: <https://nodejs.org>
* Documentation MongoDB: [https://www.mongodb.com](https://www.mongodb.com/)
* Documentación Scikit-learn: <https://scikit-learn.org>