**Requerimientos identificados FinAIcer**

**Requerimientos identificados (Diseño UX/UI e investigación)**

1. Realizar un **estudio de usuarios** (jóvenes profesionales, estudiantes universitarios y adultos que desean gestionar sus finanzas).
2. Recopilar **datos sobre hábitos financieros** y patrones de gasto.
3. Diseñar un **flujo de experiencia de usuario (UX)** simple y ágil para registrar ingresos y gastos.
4. Crear **mockups y prototipos de UI** para validar la navegación de la app.
5. Desarrollar una **interfaz intuitiva y visualmente atractiva** que incluya dashboards y reportes básicos.
6. Definir **requerimientos funcionales y no funcionales** para el MVP.
7. Elaborar la **arquitectura técnica inicial** (conexión frontend–backend–base de datos).
8. Probar la **usabilidad del prototipo** con retroalimentación de usuarios iniciales.

**Tipos de requerimientos**

**Funcionales**

* Registro, autenticación y perfil de usuario.
* Registro manual de ingresos y gastos.
* Clasificación de gastos por categorías (alimentación, transporte, entretenimiento).
* Visualización de reportes gráficos básicos (barras, pastel, líneas).
* Notificaciones simples (ej. gasto inusual).

**No Funcionales**

* **Usabilidad**: navegación intuitiva, con máximo 3 clics para registrar un gasto.
* **Rendimiento**: la app debe cargar datos en menos de 2 segundos.
* **Seguridad**: credenciales encriptadas y conexión SSL/TLS.
* **Compatibilidad**: funcionar en móviles Android (mínimo versión 10) y web responsive.
* **Escalabilidad**: permitir la incorporación de módulos futuros (IA, predicciones).

**Técnicos**

* **Frontend**: Framework híbrido (ej. React Native o Ionic).
* **Backend**: API REST con Node.js/Express.
* **Base de datos**: PostgreSQL (relacional) con posibilidad de migrar a nube (AWS RDS).
* **Integración IA**: librerías Python (Scikit-learn, TensorFlow) conectadas vía API.
* **Infraestructura**: despliegue en AWS (EC2, S3, Lambda) o equivalente en Azure.

**Requerimientos del proyecto FinAIcer (Desarrollo MVP)**

**Requerimientos identificados**

1. Implementar el **módulo de registro de ingresos y gastos** (crear, editar, eliminar).
2. Desarrollar la **base de datos relacional** para almacenar transacciones, usuarios y categorías.
3. Configurar el **entorno de desarrollo y despliegue básico** (backend, frontend, BD).
4. Crear **pantallas iniciales**: login, dashboard, formulario de ingresos y gastos.
5. Incorporar **autenticación y seguridad básica** (usuario y contraseña).
6. Validar los datos antes de guardarlos (formato de montos, categorías, fechas).
7. Probar las funcionalidades principales con **tests unitarios**.
8. Documentar el flujo y las funciones desarrolladas en el MVP.

**Tipos de Requerimientos del MVP**

**Funcionales**

* Registro de usuarios con autenticación.
* Ingreso manual de transacciones (ingresos/gastos).
* Categorización básica de los gastos.
* Consulta de transacciones por fecha y categoría.
* Eliminación y edición de registros.
* Pantalla de resumen financiero (total ingresos vs total gastos).

**No Funcionales**

* **Usabilidad**: el usuario debe poder registrar un gasto en menos de 1 minuto.
* **Rendimiento**: consultas de transacciones deben responder en < 2 segundos.
* **Seguridad**: cifrado de contraseñas (ej. bcrypt) y sesiones seguras.
* **Disponibilidad**: el MVP debe estar disponible al menos 95% del tiempo durante pruebas.
* **Compatibilidad**: funcionar en dispositivos Android gama media y navegadores web actuales.

**Técnicos**

* **Frontend**: Ionic o React Native para desarrollo móvil híbrido.
* **Backend**: Node.js/Express con API REST.
* **Base de datos**: PostgreSQL (estructura básica con tablas: usuarios, transacciones, categorías).
* **Infraestructura**: despliegue en entorno de pruebas (Heroku, AWS Free Tier o Azure estudiante).
* **Control de versiones**: Git/GitHub con ramas para desarrollo y pruebas.