**Especificación de Requerimientos de Software (ERS)**

**Capstone**

**Sección-003D**

**App de Finanzas Personales o Pymes**

***Integrantes:***

*Cristopher Arredondo*

*Angel Cea*

**Página de Actualizaciones del Proyecto**

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre del Integrante | Fecha | Actualización / Avance | Observaciones |
| Angel Cea | 03-09-2025 | Desarrollo completo del informe |  |

Notas:

• Nombre del Integrante: Persona que realiza la actualización.

• Fecha: Día en que se registra el avance o modificación.

• Actualización / Avance: Descripción breve y precisa del trabajo realizado.

• Observaciones: Comentarios adicionales, dificultades, próximos pasos o retroalimentación.

Contenido

[Introducción 4](#_Toc207815990)

[Propósito 4](#_Toc207815991)

[Alcance del Sistema 4](#_Toc207815992)

[Definiciones, siglas y abreviaturas 5](#_Toc207815993)

[Descripción General 6](#_Toc207815994)

[Perspectiva del producto 6](#_Toc207815995)

[Funciones Principales 6](#_Toc207815996)

[Características de usuarios 7](#_Toc207815997)

[Restricciones Generales 8](#_Toc207815998)

[Suposiciones y dependencias 8](#_Toc207815999)

[Requerimientos específicos 9](#_Toc207816000)

[Requerimientos funcionales 9](#_Toc207816001)

[Requerimientos no funcionales 11](#_Toc207816002)

[Tecnologías 12](#_Toc207816003)

[Diagramas 13](#_Toc207816004)

[Diagrama de casos de usos 13](#_Toc207816005)

[Diagrama de clases 14](#_Toc207816006)

[Atributos del sistema 15](#_Toc207816007)

[Pruebas 16](#_Toc207816008)

[Alcance de pruebas 16](#_Toc207816009)

[Datos de prueba (base mínima) 16](#_Toc207816010)

[Matriz de casos de prueba funcionales 17](#_Toc207816011)

[Pruebas negativas y de validación 20](#_Toc207816012)

[Pruebas de rendimiento percibido (en la app) 21](#_Toc207816013)

[Pruebas de usabilidad básica (dentro de la app) 21](#_Toc207816014)

[Funciones mínimas para el lanzamiento inicial 21](#_Toc207816015)

# Introducción

## Propósito

El propósito de este documento es especificar en detalle los requerimientos del sistema de software denominado *“Aplicación de Finanzas Personales y para Pymes”,* cuyo objetivo es entregar a personas naturales y microempresas una herramienta integral de gestión financiera.  
Este documento servirá como referencia principal para:

* El **equipo de desarrollo**, que utilizará los requerimientos aquí definidos como guía para la construcción del software.
* Los **usuarios y beneficiarios**, quienes podrán validar que las funcionalidades propuestas responden a sus necesidades.
* Los **evaluadores y revisores**, quienes verificarán que el sistema cumpla con lo comprometido en cuanto a funcionalidades, calidad, seguridad y usabilidad.

## Alcance del Sistema

La aplicación es un sistema móvil híbrido, desarrollado en Angular e Ionic, con almacenamiento seguro en Firebase/Firestore, que permitirá:

* Registrar de manera manual ingresos, gastos, deudas e inversiones.
* Clasificar los movimientos financieros mediante categorías personalizadas.
* Gestionar deudas y gastos compartidos entre varios usuarios.
* Presentar la información consolidada en reportes claros, exportables a CSV/JSON.

El sistema está orientado a:

* **Personas naturales**, que requieren mayor control sobre sus gastos diarios, planificación de ahorros y visualización de compromisos financieros.
* **Microempresas (pymes)**, que necesitan unificar información dispersa de distintas fuentes (bancos, Transbank, SII, registros internos) para mejorar la gestión de caja, controlar deudas y optimizar la relación con proveedores.

La aplicación **no depende de convenios con bancos** ni de integraciones externas en su primera versión, lo que asegura independencia y facilidad de adopción.

## Definiciones, siglas y abreviaturas

* ERS: Especificación de Requerimientos de Software.
* RF: Requerimiento Funcional.
* RNF: Requerimiento No Funcional.
* UI: Interfaz de Usuario.
* CLP: Peso chileno (moneda por defecto del sistema).
* Firebase Authentication: Servicio de autenticación de usuarios de Firebase.
* Firestore: Base de datos en la nube proporcionada por Firebase.
* Ionic/Capacitor: Framework y bridge nativo para el desarrollo de aplicaciones móviles híbridas.

# Descripción General

## Perspectiva del producto

La aplicación de finanzas personales y para pymes se concibe como un sistema móvil híbrido, desarrollado con Angular e Ionic y soportado en Firebase/Firestore como backend-as-a-service (“backend como servicio”).

El sistema no depende de convenios con bancos ni de integraciones externas para su funcionamiento inicial, lo que lo hace independiente y accesible a un público amplio.

Actualmente, las personas y las pymes suelen organizar sus finanzas mediante cartolas bancarias, aplicaciones de bancos, planillas Excel o registros manuales. Estos métodos presentan limitaciones como la falta de consolidación de la información y errores manuales.

El producto propuesto cubre estas carencias mediante:

* Unificación de registros en un único panel.
* Clasificación flexible y personalizada de movimientos.
* Módulo básico para registrar deudas y gastos compartidos.
* Reportes simples y claros sobre los movimientos registrados.

## Funciones Principales

La primera versión de la aplicación se enfocará en un conjunto reducido pero esencial de funcionalidades. El núcleo del sistema será el **re**gistro manual de operaciones financieras, lo que incluye ingresos, gastos, deudas e inversiones. Cada operación podrá clasificarse en categorías personalizadas, lo que da al usuario la libertad de organizar su información según sus propias necesidades y prioridades.

La aplicación mostrará toda la información consolidada en un panel único, de manera que el usuario no necesite revisar varias fuentes en paralelo. Además, se incorpora un módulo para la gestión de deudas y gastos compartidos, que permitirá dividir montos entre varias personas y asignar responsabilidades de forma clara, evitando confusiones y cálculos externos.

Finalmente, el sistema permitirá generar reportes básicos que resumen los movimientos registrados. Dichos reportes estarán disponibles en formatos exportables (CSV/JSON), lo que facilitará su respaldo o su uso en otras plataformas de análisis financiero.

## Características de usuarios

El público objetivo de la aplicación se divide en dos grupos principales.

Por un lado, están las personas naturales, quienes en muchos casos no cuentan con formación financiera formal y suelen depender del saldo disponible en sus cuentas bancarias o de simples anotaciones en planillas. Para este grupo, la aplicación representa una herramienta accesible y clara que les permitirá visualizar sus gastos, organizar mejor sus recursos y tomar decisiones con más información.

Por otro lado, se encuentran las microempresas (pymes), que enfrentan el desafío de consolidar información dispersa entre bancos, proveedores y registros internos. La app les ofrecerá una solución práctica para llevar un control más estricto de sus flujos de caja, organizar compromisos financieros y fortalecer la relación con proveedores al contar con un registro claro y actualizado de su situación económica.

## Restricciones Generales

El sistema debe funcionar en dispositivos móviles con el sistema operativo Android. Toda la gestión de datos se realizará en la nube mediante los servicios de Firebase (Authentication, Firestore y Storage), lo que permite seguridad, persistencia y respaldo de la información.

Se establecen restricciones de calidad que la aplicación debe cumplir desde su primera versión: la disponibilidad mínima del servicio debe ser del 99%, el tiempo de respuesta en operaciones comunes no debe superar los tres segundos y la interfaz debe ser lo suficientemente intuitiva como para que pueda ser utilizada por personas con conocimientos técnicos básicos.

Asimismo, se define como restricción que la primera versión se limita al registro manual de operaciones y la generación de reportes simples. No contempla integración automática con bancos ni el uso de algoritmos avanzados de proyección o inteligencia artificial.

## Suposiciones y dependencias

El desarrollo de este sistema parte de varias suposiciones. Se asume que los usuarios dispondrán de un acceso básico a internet para utilizar la aplicación, aunque el almacenamiento de datos y autenticación dependerán completamente de los servicios de Firebase. También se asume que los usuarios estarán dispuestos a ingresar la mayoría de la información manualmente, al menos en la etapa inicial del proyecto, ya que no se consideran integraciones externas en esta primera entrega.

En cuanto a dependencias, el sistema depende de la estabilidad y disponibilidad de los servicios proporcionados por Firebase, incluyendo autenticación, base de datos y almacenamiento de archivos. Una eventual caída o limitación de dichos servicios afectaría directamente el funcionamiento de la aplicación.

# Requerimientos específicos

## Requerimientos funcionales

Los requerimientos funcionales describen lo que el sistema debe ser capaz de hacer desde la perspectiva del usuario. Cada funcionalidad se asocia a un identificador único (RF) que permitirá mantener la trazabilidad con las pruebas y la validación.

En la primera versión, el sistema contempla las siguientes funcionalidades:

|  |  |  |
| --- | --- | --- |
| **ID** | **Requerimiento Funcional** | **Detalle / Criterio de Aceptación** |
| RF-01 | Registro de usuario y autenticación | Permitir crear cuenta, iniciar/cerrar sesión y recuperar contraseña usando Firebase Authentication. |
| RF-02 | Perfil de usuario | Guardar nombre, tipo de usuario (persona/pymes), moneda (CLP por defecto) y preferencias básicas. |
| RF-03 | Ingreso manual de ingresos | Crear ingresos con fecha, monto, categoría, descripción y medio de pago. |
| RF-04 | Ingreso manual de gastos | Crear gastos con fecha, monto, categoría, descripción y medio de pago. |
| RF-05 | Registro de deudas | Crear deudas con acreedor/deudor, monto total, número de cuotas, tasa (opcional) y fechas de vencimiento. |
| RF-06 | Registro de inversiones | Registrar inversiones con tipo, monto, fecha y horizonte (sin cálculos automáticos externos). |
| RF-07 | Edición y eliminación de registros | Editar/eliminar ingresos, gastos, deudas e inversiones con confirmación. |
| RF-08 | Clasificación flexible | Crear/editar categorías personalizadas y asignarlas a cualquier movimiento. |
| RF-09 | Búsqueda y filtros | Filtrar por fecha, tipo (ingreso/gasto/deuda/inversión), categoría y rango de montos; búsqueda por texto. |
| RF-10 | Panel unificado | Mostrar en una vista consolidada saldos, totales de ingresos/gastos, deudas activas e inversiones registradas. |
| RF-11 | Gestión de gastos/deudas compartidas | Dividir un gasto/deuda entre varias personas, asignar porcentajes o partes iguales y calcular automáticamente los montos. |
| RF-12 | Moneda y formato local | Usar CLP por defecto y formatos de fecha/montos locales; permitir cambiar moneda (sin tipos de cambio automáticos). |
| RF-13 | Persistencia en Firestore | Guardar y recuperar todos los datos del usuario en colecciones seguras por UID. |
| RF-14 | Validaciones | Validar campos obligatorios, tipos de dato, montos positivos y fechas válidas; mensajes claros ante errores. |
| RF-15 | Gestión de categorías | CRUD de categorías (crear, renombrar, eliminar) y reasignación masiva al eliminar una categoría. |
| RF-16 | Copia de seguridad/exportación | Exportar todos los datos del usuario a CSV/JSON desde la app. |
| RF-17 | Eliminar cuenta (Right to Delete) | Permitir al usuario borrar su cuenta y todos sus datos asociados. |

## Requerimientos no funcionales

Además de las funciones descritas, el sistema debe cumplir con un conjunto de requerimientos de calidad, seguridad y rendimiento que aseguren la experiencia de uso y la continuidad del servicio.

|  |  |  |
| --- | --- | --- |
| **ID** | **Requerimiento No Funcional** | **Detalle / Justificación** |
| RNF-01 | Tecnología base: la aplicación debe desarrollarse en Angular e Ionic. | Permite construir una app híbrida, desplegable en Android en V1 y con posibilidad de iOS en futuras versiones. |
| RNF-02 | Base de datos: debe usar Firebase/Firestore. | Asegura almacenamiento en la nube, sincronización y seguridad en el manejo de datos. |
| RNF-03 | Seguridad de datos: toda la información debe transmitirse con cifrado. | Protege la información sensible de usuarios frente a ataques o intercepciones. |
| RNF-04 | Confidencialidad: los datos registrados por el usuario no podrán compartirse sin autorización explícita. | Refuerza la confianza de usuarios y cumple con buenas prácticas de privacidad. |
| RNF-05 | Disponibilidad: la aplicación debe estar disponible al menos el 99% del tiempo. | Para garantizar continuidad de uso, especialmente en pymes que dependen de ella a diario. |
| RNF-06 | Rendimiento: las consultas y registros deben responder en un máximo de 3 segundos. | Asegura fluidez en la experiencia del usuario. |
| RNF-07 | Usabilidad: la interfaz debe ser intuitiva, clara y simple, orientada a usuarios sin conocimientos técnicos. | Permite adopción masiva tanto en personas naturales como pymes. |
| RNF-08 | Escalabilidad: la arquitectura debe permitir la incorporación de futuras funcionalidades. | La arquitectura debe permitir la incorporación de futuras funcionalidades |
| RNF-09 | Mantenibilidad: el código debe estar documentado y estructurado para facilitar actualizaciones. | Reduce costos de mantenimiento y asegura continuidad del proyecto. |

## Tecnologías

|  |  |  |
| --- | --- | --- |
| **Capa / Área** | **Tecnología / Servicio** | **Rol en el proyecto** |
| Frontend móvil (UI) | Angular 20.1.5 | Base del front: componentes, enrutamiento, estados y lógica de presentación. |
| UI Móvil / Bridge nativo | Ionic 7.2.1 | Componentes UI móviles y empaquetado híbrido para Android/iOS. |
| Acceso a hardware/dispositivo | Capacitor (Ionic) | Plugins nativos (filesystem, network, device info) si se requieren. |
| Lenguaje | TypeScript | Tipado estático para reducir errores y mejorar mantenibilidad. |
| Entorno runtime | Node.js 22.14.0 + npm 10.9.2 | Gestión de dependencias, ejecución de scripts de build y entorno de desarrollo. |
| Backend-as-a-Service | Firebase | Plataforma administrada: autenticación, base de datos, reglas de seguridad, almacenamiento. |
| Autenticación | Firebase Authentication | Registro, login, recuperación de contraseña. |
| Base de datos | Cloud Firestore | Persistencia de ingresos, gastos, deudas, inversiones, categorías, perfiles, etc. |
| Reglas de seguridad | Firestore Security Rules | Control de acceso a documentos/colecciones por UID. |
| Control de versiones | Git + GitHub | Versionado, PRs, manejo de ramas y revisiones de código. |
| Lenguaje adicional | Python 3.13.2 | Uso en análisis de datos, generación de reportes y posibles módulos de procesamiento offline. |

# Diagramas

## Diagrama de casos de usos

## Diagrama de clases

# Atributos del sistema

El sistema deberá cumplir con un conjunto de atributos de calidad que garanticen seguridad, confiabilidad, facilidad de uso, rendimiento y capacidad de crecimiento. Estos atributos complementan los requerimientos funcionales y no funcionales, y constituyen criterios de aceptación transversales a todo el proyecto.

**Seguridad.** Toda la información registrada en la aplicación deberá almacenarse en Firebase/Firestore bajo reglas de acceso restringidas por identificador de usuario (UID). Asimismo, se deberá garantizar la correcta implementación de procesos de autenticación y recuperación de credenciales a través de Firebase Authentication, asegurando que ningún usuario pueda acceder a información de terceros.

**Usabilidad.** La interfaz de usuario debe diseñarse con un enfoque en la simplicidad y claridad, de modo que pueda ser utilizada sin dificultad por personas sin conocimientos técnicos avanzados. Los flujos de registro de operaciones financieras deberán ser intuitivos, con formularios claros, mensajes de error comprensibles y retroalimentación inmediata en las acciones más comunes (registro, edición, eliminación).

**Fiabilidad y disponibilidad.** La aplicación deberá mantener una disponibilidad mínima del 99%, apoyándose en la infraestructura de Firebase, que garantiza redundancia y tolerancia a fallos. Esto resulta crítico en el contexto de las pymes, que dependen del sistema para su gestión diaria.

**Rendimiento.** Todas las operaciones básicas (registro de un movimiento, consulta de lista filtrada, edición o eliminación) deberán ejecutarse con un tiempo de respuesta inferior a 3 segundos en condiciones de conexión a internet estable. Este rendimiento deberá mantenerse incluso con volúmenes de datos medianos (hasta varios miles de registros por usuario).

**Escalabilidad y mantenibilidad.** La arquitectura de la aplicación se basará en componentes modulares construidos en Angular e Ionic, lo que permitirá extender el sistema en futuras versiones con nuevas funcionalidades, como integraciones externas o módulos adicionales. El código deberá estar documentado y estructurado, de modo que un equipo de desarrollo distinto pueda realizar mejoras sin dificultad.

# Pruebas

## Alcance de pruebas

* **Funcional**: Ingresos, gastos, deudas, inversiones, clasificación, filtros/búsquedas, panel unificado, reportes.
* **No funcional**: Usabilidad básica (claridad de formularios), rendimiento percibido (≤3 s en operaciones comunes), seguridad de datos (no exponer información entre usuarios).

## Datos de prueba (base mínima)

* **Personas naturales**
* Ingresos: sueldo $900.000 (mensual), ingreso extra $120.000 (una vez)
* Gastos: alimentación $180.000, transporte $60.000, ocio $70.000, servicios $80.000
* Deuda: celular $360.000 en 6 cuotas
* Inversión: depósito $100.000 (registro simple, sin cálculo externo)
* **Pyme**
* Ingresos (ventas): 10 registros entre $25.000 y $180.000
* Gastos: proveedor A $250.000, proveedor B $180.000, servicios $90.000
* Deuda: compra insumos $600.000 en 4 cuotas.
* Inversión: $300.000 (registro simple)

## Matriz de casos de prueba funcionales

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Módulo** | **Caso** | **Pasos (en la app)** | **Resultado esperado (Criterio de aceptación)** |
| PF-01 | Registro/Acceso | Crear cuenta de usuario | Abrir app → “Crear cuenta” → ingresar datos → Guardar | Se crea la cuenta en Firebase Auth; el usuario queda autenticado y redirigido al panel inicial. |
| PF-02 | Autenticación | Iniciar sesión | Abrir app → ingresar correo/contraseña → Ingresar | Acceso concedido; se muestra el panel con datos del usuario. |
| PF-03 | Perfil de usuario | Configurar perfil | Menú → “Perfil” → nombre, tipo (persona/pyme), moneda (CLP) → Guardar | Datos almacenados en Firestore y visibles al reabrir la app. |
| PF-04 | Ingresos | Crear ingreso manual (persona) | Abrir “Nuevo ingreso” → completar (monto, fecha, categoría, medio, descripción) → Guardar | El ingreso aparece en la lista; el panel unificado actualiza totales sin recargar la pantalla. |
| PF-05 | Gastos | Crear gasto manual (persona) | “Nuevo gasto” → completar campos → Guardar | Lista y panel actualizan totales; se refleja el nuevo saldo. |
| PF-06 | Deudas | Registrar deuda con cuotas (persona) | “Nueva deuda” → monto total, número de cuotas, fechas de vencimiento → Guardar | Se registran las cuotas programadas; el panel refleja deuda activa y futuras obligaciones. |
| PF-07 | Inversiones | Registrar inversión (persona) | “Nueva inversión” → tipo, monto, fecha → Guardar | La inversión se lista y el panel la distingue de ingresos/gastos. |
| PF-08 | Clasificación | Crear categoría personalizada | “Categorías” → “Nueva categoría” → nombre/color → Guardar | Disponible en selectores; al reclasificar movimientos, reportes reflejan la nueva categoría. |
| PF-09 | Filtros/Búsqueda | Filtrar por rango de fechas y categoría | Vista de movimientos → configurar filtros → aplicar | Se muestran solo los movimientos que cumplen; Exportar CSV respeta el filtro activo. |
| PF-10 | Panel unificado | Ver totales consolidados | Abrir panel principal | Ver totales de ingresos/gastos/deudas/inversiones coherentes con lo ingresado. |
| PF-11 | Deudas compartidas | Dividir gasto entre 3 personas | “Nuevo gasto compartido” → agregar 3 participantes → partes iguales → Guardar | Monto por persona correcto; total = suma de partes; si se edita un valor, el cálculo se actualiza correctamente. |
| PF-12 | Edición | Editar gasto (monto/fecha) | Abrir gasto → Editar → Guardar | Totales se recalculan; el cambio queda persistido en Firestore. |
| PF-13 | Eliminación | Eliminar ingreso | Abrir ingreso → Eliminar → Confirmar | Se elimina el registro; panel/reportes se actualizan acorde. |
| PF-14 | Exportación | Exportar CSV del mes | Reportes → seleccionar período → “Exportar CSV” | Archivo CSV con columnas y formato esperado; cantidad de filas coincide con el filtro aplicado. |
| PF-15 | Persistencia | Cerrar sesión y volver a ingresar | Registrar algunos movimientos → Cerrar sesión → Iniciar sesión | Todos los datos permanecen disponibles para el usuario autenticado. |
| PF-16 | Categorías | Eliminar categoría usada (con reasignación) | Menú “Categorías” → eliminar categoría → seleccionar categoría destino para movimientos afectados | La categoría se elimina y los movimientos quedan correctamente reasignados. |
| PF-17 | Respaldo | Exportar respaldo completo | Menú → “Exportar” → “Exportar todo (CSV/JSON)” | Archivo con todos los registros del usuario, legible y con consistencia de conteos. |
| PF-18 | Cuenta | Eliminar cuenta y datos | Menú → “Configuración” → “Eliminar cuenta” → Confirmar | Se elimina la cuenta de Firebase Auth y todos los datos del usuario en Firestore. |

## Pruebas negativas y de validación

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Caso** | **Entrada inválida** | **Resultado esperado** |
| PN-01 | Monto vacío | Monto = “” | Mensaje claro: “El monto es obligatorio”; no guarda el registro. |
| PN-02 | Monto negativo | Monto = -1000 | Mensaje: “El monto debe ser positivo”; no guarda. |
| PN-03 | Fecha inválida | Vencimiento anterior a la fecha de origen | Mensaje: “La fecha de vencimiento no es válida”; no guarda. |
| PN-04 | Categoría inexistente | Eliminar categoría que está en uso | Forzar reasignación a otra categoría antes de eliminar; impedir eliminación directa. |
| PN-05 | Duplicado accidental | Crear 2 veces el mismo gasto en 1 minuto | Mostrar alerta no bloqueante: “¿Seguro? Parece duplicado”. |

## Pruebas de rendimiento percibido (en la app)

* **Acciones objetivo:** crear/editar/eliminar movimiento; abrir panel; filtrar/buscar; abrir reportes.
* **Criterio:** tiempo de respuesta ≤ 3 s (RNF-06) con los datasets base definidos en el informe.
* **Medición:** cronómetro manual y logs de tiempo en consola (solo desarrollo), sin herramientas externas.

## Pruebas de usabilidad básica (dentro de la app)

**Objetivo:** que una persona sin experiencia registre 5 movimientos y vea un reporte sin pedir ayuda.

**Criterios:**

* Completa el flujo en ≤ 8 minutos.
* No comete más de 2 errores corregidos por validación.
* Comprende dónde ver totales en el panel y cómo exportar un reporte.

## Funciones mínimas para el lanzamiento inicial

Marcar **✅** si se cumple en la app:

* Puedo registrar ingresos, gastos, deudas e inversiones y verlos unificados.
* Puedo clasificar y luego filtrar/buscar por categoría/fecha.
* Puedo dividir un gasto/deuda con otras personas y el cálculo es correcto.
* Puedo exportar un reporte a CSV (y respaldo completo CSV/JSON)
* Las acciones comunes responden en ≤ 3 s.
* Ningún dato mío aparece en otra cuenta (aislamiento por usuario).