

Requisitos para App de Gestión de Gastos del Hogar con React Native y NestJS

Requisitos Funcionales y No Funcionales para una Aplicación Móvil de Gestión de Gastos del Hogar con React Native y NestJS

En este contexto, el objetivo de este informe es detallar de forma exhaustiva los requisitos funcionales y no funcionales para una aplicación móvil de gestión de gastos del hogar, desarrollada con React Native para el frontend y NestJS para el backend. La aplicación estará orientada a la colaboración entre uno o mas usuarios que comparten una lista de compras, permitiendo marcar productos como comprados y sumando automáticamente su costo al gasto total del hogar. Además, se incluyen funcionalidades adicionales y propuestas de valor agregado para maximizar la utilidad y la experiencia del usuario.

El documento está estructurado en secciones que abordan los requisitos funcionales esenciales, funcionalidades adicionales, requisitos no funcionales y propuestas de valor agregado. Se emplean tablas para organizar las funcionalidades por categoría y prioridad, seguidas de análisis detallados que contextualizan y justifican cada requisito. El enfoque se basa en las mejores prácticas de desarrollo móvil, usabilidad, seguridad y escalabilidad, integrando referencias actualizadas del sector fintech, guías de desarrollo y ejemplos de aplicaciones líderes en el mercado.

1. Requisitos Funcionales Esenciales

1.1. Gestión de Listas de Compras Compartidas

La funcionalidad central de la aplicación es permitir que dos o mas usuarios gestionen de manera colaborativa una o varias listas de compras, con la posibilidad de agregar, editar, eliminar y marcar productos como comprados. Esta característica es

fundamental para la organización eficiente de las compras del hogar y la reducción de gastos innecesarios.

Características clave:

- Creación, edición y eliminación de listas de compras.
- Compartición de listas entre dos o mas usuarios del mismo hogar.
- Sincronización en tiempo real de los cambios realizados por ambos usuarios.
- Visualización del estado de cada producto (pendiente/comprado).
- Ordenación y filtrado de productos por categorías, prioridad o fecha de compra.

La colaboración en tiempo real es esencial para evitar duplicidades y garantizar que ambos usuarios tengan acceso a la información más actualizada. La posibilidad de gestionar múltiples listas (por ejemplo, compras semanales, eventos especiales, compras de emergencia) añade flexibilidad y se adapta a diferentes necesidades familiares.

1.2. Registro y Seguimiento de Gastos del Hogar

La aplicación debe registrar automáticamente el costo de cada producto marcado como comprado, sumando el importe al gasto total del hogar. Además, debe permitir el seguimiento detallado de los gastos por categoría, fecha y usuario.

Características clave:

- Registro automático del gasto al marcar un producto como comprado.
- Visualización del gasto total acumulado y desglose por categorías (alimentos, limpieza, etc.).
- Historial de compras y gastos con filtros por periodo, categoría y usuario.
- Gráficas y reportes visuales para facilitar el análisis de los hábitos de consumo.

El seguimiento detallado de los gastos es fundamental para identificar patrones de consumo, detectar excesos y fomentar una cultura de ahorro en el hogar. La integración de reportes visuales mejora la comprensión y facilita la toma de decisiones financieras.

1.3. Marcado de Productos como Comprados y Cálculo Automático

Cada producto de la lista puede marcarse como comprado, lo que desencadena el cálculo automático del gasto y la actualización del estado del producto. Esta funcionalidad debe ser intuitiva y rápida, permitiendo a los usuarios registrar compras sobre la marcha.

Características clave:

- Botón o gesto rápido para marcar productos como comprados.
- Actualización instantánea del gasto total y del estado del producto.
- Registro de la fecha y hora de compra para cada producto.
- Opción de deshacer el marcado en caso de error.

La facilidad para registrar compras en tiempo real es clave para la adopción y el uso constante de la aplicación. La posibilidad de deshacer acciones previene errores y mejora la experiencia de usuario.

1.4. Autenticación y Gestión de Usuarios

La seguridad y la privacidad de los datos requieren un sistema robusto de autenticación y gestión de usuarios. La aplicación debe permitir el registro, inicio de sesión y recuperación de cuenta, así como la gestión de perfiles y preferencias.

Características clave:

- Registro de usuarios con correo electrónico y contraseña segura.
- Inicio de sesión y cierre de sesión.
- Recuperación de contraseña mediante correo electrónico.
- Autenticación de dos factores (2FA) opcional para mayor seguridad.
- Gestión de perfiles de usuario (nombre, foto, preferencias).
- Asociación de usuarios a un hogar (Puede ser mas de dos por hogar).

La autenticación segura es imprescindible para proteger la información financiera y personal de los usuarios. La opción de 2FA añade una capa adicional de protección, alineándose con las mejores prácticas de seguridad actuales.

1.5. Sincronización en Tiempo Real y Manejo de Conflictos

La colaboración efectiva requiere que los cambios realizados por un usuario se reflejen instantáneamente en el dispositivo del otro usuario. La aplicación debe implementar sincronización en tiempo real y mecanismos para resolver conflictos cuando ambos usuarios modifican la misma información simultáneamente.

Características clave:

- Sincronización instantánea de listas y gastos mediante WebSockets o tecnologías similares.
- Resolución automática o manual de conflictos (por ejemplo, último cambio prevalece o confirmación del usuario).
- Indicadores visuales de sincronización y estado de conexión.
- Soporte para funcionamiento offline y sincronización posterior.

La sincronización en tiempo real mejora la experiencia colaborativa y evita inconsistencias en los datos. El manejo adecuado de conflictos es esencial para mantener la integridad de la información y prevenir pérdidas de datos.

1.6. Notificaciones y Alertas (Push y Locales)

Las notificaciones son fundamentales para mantener a los usuarios informados sobre cambios relevantes, recordatorios de compras y alertas de gastos. La aplicación debe soportar notificaciones push y locales, personalizables según las preferencias del usuario.

Características clave:

- Notificaciones push para avisar sobre productos comprados, cambios en la lista o nuevos gastos.
- Recordatorios programados para realizar compras o revisar el presupuesto.
- Alertas de sobrepaso de presupuesto o gastos inusuales.
- Configuración de preferencias de notificación por usuario.

La personalización de las notificaciones permite adaptar la experiencia a las necesidades de cada usuario, evitando la saturación y mejorando la relevancia de los avisos.

1.7. Historial y Reportes de Gastos

El acceso al historial de compras y gastos es esencial para el análisis financiero y la toma de decisiones. La aplicación debe ofrecer reportes detallados y exportables, con filtros avanzados y visualizaciones gráficas.

Características clave:

- Historial completo de compras y gastos con filtros por fecha, categoría y usuario.
- Reportes visuales (gráficas de barras, pastel, líneas) para analizar tendencias.
- Exportación de datos en formatos CSV y PDF.
- Comparativas mensuales y alertas de variaciones significativas.

La capacidad de exportar datos facilita la integración con otras herramientas y la consulta externa, mientras que los reportes visuales mejoran la comprensión de la información financiera.

1.8. Gestión de Productos y Categorías

La organización eficiente de las listas de compras requiere una gestión flexible de productos y categorías. La aplicación debe permitir la creación, edición y eliminación de productos y categorías personalizadas.

Características clave:

- Catálogo de productos frecuentes y favoritos.
- Creación y edición de categorías personalizadas (alimentos, limpieza, etc.).
- Asignación de productos a categorías y subcategorías.
- Sugerencias automáticas de productos basadas en historial y frecuencia de compra.

La personalización de categorías y productos facilita la adaptación de la aplicación a diferentes estilos de vida y preferencias de consumo.

1.9. Offline-First y Sincronización Posterior

La aplicación debe ser funcional incluso sin conexión a Internet, permitiendo a los usuarios consultar y modificar listas y gastos offline, con sincronización automática al recuperar la conexión.

Características clave:

- Acceso completo a listas y gastos en modo offline.
- Almacenamiento local de cambios y sincronización automática al reconectar.
- Indicadores de estado offline/online y sincronización pendiente.
- Resolución de conflictos tras la sincronización.

El enfoque offline-first mejora la confiabilidad y la usabilidad en entornos con conectividad limitada, aumentando la satisfacción del usuario.

1.10. Seguridad y Privacidad de Datos

La protección de la información financiera y personal es prioritaria. La aplicación debe implementar cifrado de datos en tránsito y en reposo..

Características clave:

- Cifrado de datos en tránsito (TLS/HTTPS) y en reposo (AES u otro estándar).
- Gestión segura de claves y contraseñas.
- Eliminación segura de datos a solicitud del usuario.

El cumplimiento de estándares de seguridad y privacidad es esencial para generar confianza y evitar riesgos legales y reputacionales.

1.11. División de Gastos

Para facilitar la gestión financiera compartida, la aplicación debe permitir la división automática de gastos entre los usuarios.

Características clave:

- Registro de pagos realizados por cada usuario.
- División automática o manual de gastos (porcentaje, monto fijo, alternancia).
- Historial de pagos y deudas pendientes entre usuarios.

La automatización de la división de gastos reduce conflictos y simplifica la administración financiera del hogar.

2. Requisitos Funcionales Adicionales

2.1. Presupuestos y Metas de Ahorro

La aplicación puede ofrecer la posibilidad de establecer presupuestos mensuales y metas de ahorro, con seguimiento automático y alertas de cumplimiento.

Características clave:

- Definición de presupuestos por categoría y periodo.
- Seguimiento del progreso hacia metas de ahorro.
- Alertas de sobrepaso de presupuesto o cumplimiento de metas.
- Visualización de avances y recomendaciones para optimizar el ahorro.

El establecimiento de metas fomenta la disciplina financiera y motiva a los usuarios a mejorar sus hábitos de consumo.

2.7. Exportación e Importación de Datos

La posibilidad de exportar e importar datos facilita la migración, el respaldo y la integración con otras herramientas financieras¹⁸.

Características clave:

- Exportación de listas, gastos e historial en formatos CSV y PDF.
- Sincronización con servicios en la nube para respaldo automático.

La interoperabilidad y la portabilidad de los datos son cada vez más valoradas por los usuarios avanzados.

3. Requisitos No Funcionales

3.1. Rendimiento y Escalabilidad

La aplicación debe ofrecer un rendimiento óptimo, tiempos de respuesta bajos y capacidad para escalar según el crecimiento de usuarios y datos.

Características clave:

- Tiempos de carga rápidos y navegación fluida.
- Optimización de recursos en dispositivos móviles (memoria, batería).

- Escalabilidad horizontal del backend mediante clustering o contenedores.
- Uso eficiente de bases de datos y almacenamiento local.

El rendimiento es crítico para la satisfacción del usuario y la retención a largo plazo.

3.2. Disponibilidad y Tolerancia a Fallos

La aplicación debe garantizar alta disponibilidad y mecanismos de recuperación ante fallos, minimizando el impacto de caídas o errores.

Características clave:

- Arquitectura redundante y balanceo de carga en el backend.
- Monitoreo y alertas automáticas ante incidencias.
- Recuperación automática de servicios y datos tras fallos.
- Soporte para actualizaciones sin interrupciones.

La disponibilidad continua es esencial para aplicaciones que gestionan información crítica y colaborativa.

3.3. Seguridad y Cumplimiento

El cumplimiento de estándares de seguridad y normativas legales es obligatorio para proteger la información y evitar sanciones.

Características clave:

- Cifrado de datos en tránsito y en reposo.
- Autenticación robusta y gestión segura de sesiones.
- Protección contra ataques comunes (XSS, CSRF, inyección SQL/NoSQL).

La seguridad debe ser una prioridad transversal en todo el ciclo de vida de la aplicación.

3.4. Mantenibilidad y Arquitectura

La aplicación debe estar diseñada para facilitar el mantenimiento, la evolución y la incorporación de nuevas funcionalidades.

Características clave:

- Arquitectura modular y desacoplada (microservicios, monorepo, etc.).
- Documentación clara y actualizada del código y la API.
- Pruebas automatizadas (unitarias, de integración, end-to-end).
- Control de versiones y despliegue continuo.

La mantenibilidad reduce costos a largo plazo y agiliza la respuesta a cambios y mejoras.

3.5. Usabilidad y Accesibilidad

La experiencia de usuario debe ser intuitiva, accesible y adaptada a diferentes perfiles y necesidades.

Características clave:

- Interfaz limpia, minimalista y coherente con las guías de diseño de Android.
- Feedback visual y háptico inmediato ante acciones del usuario.

La usabilidad y la accesibilidad son factores diferenciadores en el éxito de aplicaciones financieras.

3.7. Monitoreo y Analítica

El monitoreo continuo y la analítica de uso permiten detectar problemas, optimizar la experiencia y tomar decisiones basadas en datos.

Características clave:

- Analítica de uso en el frontend (eventos, flujos, retención).

La analítica es fundamental para la mejora continua y la adaptación a las necesidades reales de los usuarios.

5. Tabla Resumen de Funcionalidades por Categoría y Prioridad

Categoría	Funcionalidad Esencial	Funcionalidad Adicional	Valor Agregado	Prioridad
Listas de compras				Alta

compartidas				
Seguimiento de gastos				Alta
Marcado de productos/compras				Alta
Autenticación y usuarios			Seguridad avanzada	Alta
Sincronización en tiempo real				Alta
Notificaciones y alertas				Alta
Historial y reportes		Exportación/importación	Insights IA	Alta
Gestión de productos/categorías				Media
Offline-first				Alta
Seguridad y privacidad			Seguridad avanzada	Alta
Presupuestos y metas de ahorro			Gamificación	Media
Exportación/importación de datos				Media

7. Consideraciones Técnicas para React Native y NestJS

7.1. React Native (Frontend)

- **Reutilización de código:** Permite compartir hasta el 90% del código entre Android e iOS, optimizando tiempos y costos de desarrollo.
- **Gestión de estado:** Uso de Redux o Context API para manejar el estado global de listas, gastos y usuarios.

- **Sincronización en tiempo real:** Implementación de WebSockets para comunicación bidireccional y actualizaciones instantáneas⁷.
- **Notificaciones:** Integración con Firebase Cloud Messaging para push notifications y manejo de notificaciones locales⁸.
- **Accesibilidad y usabilidad:** Cumplimiento de las guías de accesibilidad de React Native y Material Design²².
- **Internacionalización:** Uso de librerías como i18next para soporte multilingüe y localización¹².
- **Optimización de listas:** Uso de FlatList y técnicas de virtualización para manejar grandes volúmenes de datos sin afectar el rendimiento²⁶.
- **Integración con asistentes de voz y widgets:** Implementación de bridges nativos para funcionalidades avanzadas en Android e iOS¹⁷.

7.2. NestJS (Backend)

- **Arquitectura modular:** Organización en módulos independientes para usuarios, listas, gastos, notificaciones, etc.
 - **Seguridad:** Implementación de JWT para autenticación, cifrado de datos y protección contra ataques comunes.
 - **Escalabilidad:** Uso de docker para escalar horizontalmente según la demanda.
 - **Sincronización en tiempo real:** Integración de WebSockets (Socket.io) para comunicación instantánea con el frontend.
 - **Persistencia de datos:** Uso de base de datos relacional postgres.
-