Informe Móvil I

Mi PistoHN

Versiones del documento

Fecha	Versión	Descripción	Autor
Junio 19, 2025	1.0	Informe técnico de MIPISTOHN	1. Andrea Galo 2. Cesar Reyes 3. Christopher Enamorado 4. Paola Deras

Contenido

Informe Desarrollo Móvil I

1. DESCRIPCIÓN DE LA SOLUCIÓN

1.1 Resumen gerencial

MiPistoHN es una aplicación móvil de gestión financiera personal diseñada específicamente para el contexto hondureño. La aplicación permite a los usuarios llevar un control detallado de sus ingresos y gastos de manera sencilla e intuitiva. El sistema cuenta con funcionalidades de registro de gastos por categorías, gestión de presupuestos, visualización de estadísticas y un sistema de autenticación seguro. La aplicación busca democratizar el acceso a herramientas de control financiero, ofreciendo una interfaz amigable que no requiere conocimientos previos en finanzas.

1.2 Glosario

Término	Definición
Presupuesto	Monto de dinero establecido por el usuario para un período determinado
Gasto	Registro de una salida de dinero categorizada y documentada
Categoría	Clasificación de gastos (alimentación, transporte, entretenimiento, etc.)

Restricciones y metas de arquitectura

Restricciones técnicas:

- La aplicación debe ser compatible con dispositivos Android (versión mínima API 21)
- Requiere conexión a internet para sincronización de datos con API REST
- El backend debe soportar autenticación JWT con middleware de verificación
- La base de datos MySQL debe mantener integridad referencial entre usuarios, gastos, categorías y presupuestos
- Manejo de archivos de imágenes limitado a formatos estándar (JPG, PNG)
- Configuración de zona horaria América Central (UTC-6) para consistencia de timestamps

Restricciones de seguridad:

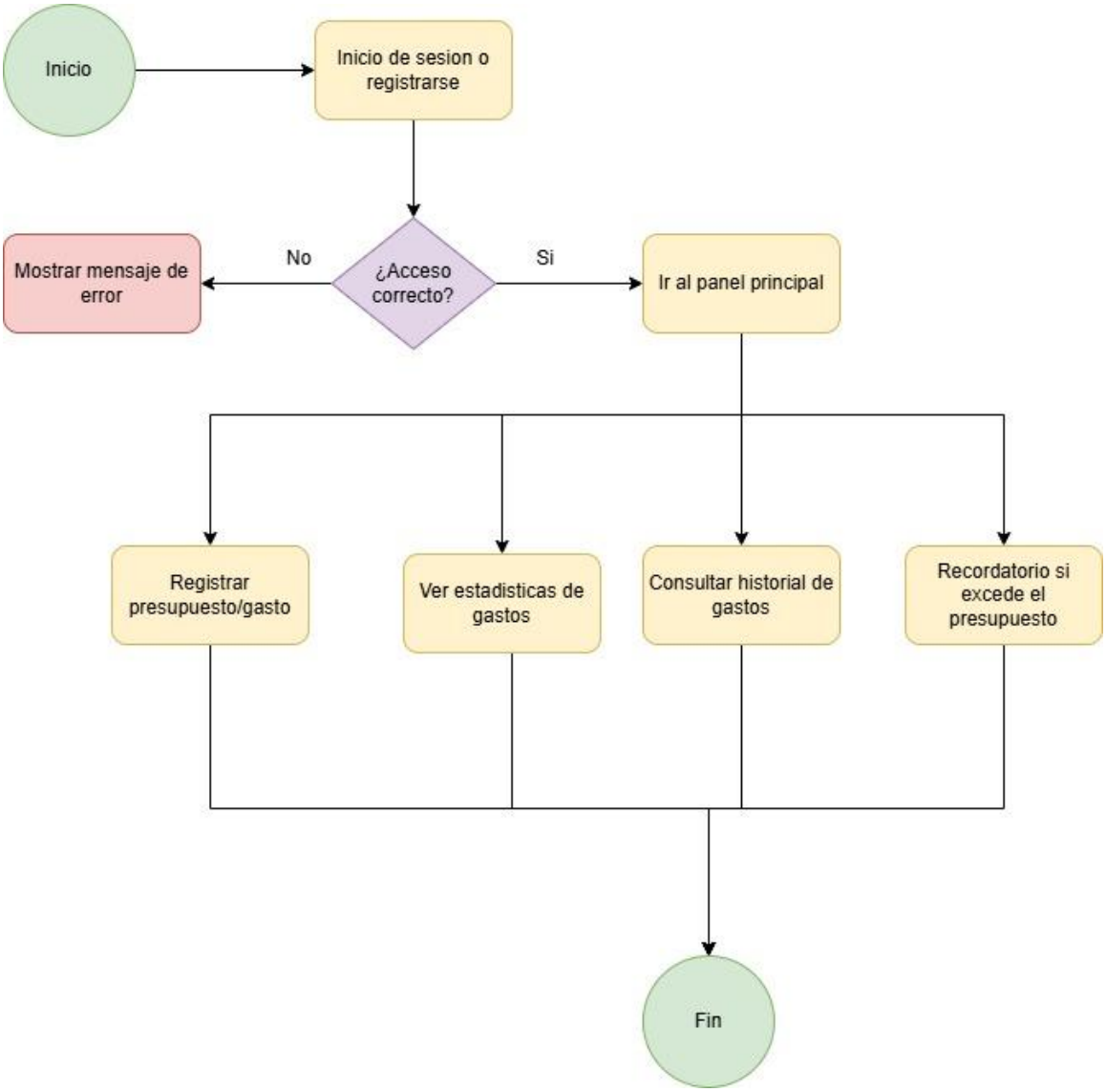
- Encriptación obligatoria de contraseñas con bcrypt
- Validación de formato de correo electrónico en frontend y backend
- Tokens JWT con tiempo de expiración configurado
- Middleware de autenticación obligatorio para todos los endpoints protegidos
- Validación de tipos de archivo en subida de imágenes de perfil

Metas de Arquitectura:

- Escalabilidad: El sistema debe soportar múltiples usuarios concurrentes mediante pool de conexiones MySQL
- Seguridad: Implementación robusta de autenticación JWT y protección de datos financieros personales
- Usabilidad: Interfaz intuitiva que requiera mínima curva de aprendizaje para gestión financiera básica
- Performance: Tiempos de respuesta menores a 3 segundos en operaciones CRUD con optimización de consultas
- Mantenibilidad: Código modular con separación clara de responsabilidades (MVC), ORM para abstracción de datos
- Confiabilidad: Uso de transacciones de base de datos para operaciones críticas (registro de usuario, creación de gastos)
- Consistencia: Manejo centralizado de respuestas de API y formato uniforme de errores

1.2 Diagrama de Proceso

Flujo principal de la aplicación:



1.3 Requerimientos del sistema

Requerimientos funcionales:

- Sistema de autenticación con registro y login de usuarios
- Encriptación de contraseñas con bcrypt
- Gestión completa de gastos personales con categorización (15 categorías predefinidas)
- Sistema de presupuestos con cálculo automático de saldo disponible
- Historial completo de gastos con ordenamiento cronológico
- Estadísticas de gastos agrupadas por categoría y rango de fechas
- Gestión de perfil de usuario con subida de imágenes
- API REST con endpoints para todas las operaciones CRUD
- Persistencia segura de tokens de autenticación

Requerimientos no funcionales:

- Compatibilidad: Android 5.0+ (API 21), Node.js 14+, MySQL 8.0+
- Interfaz: Responsiva y adaptable con React Native
- Performance: Tiempo de respuesta < 3 segundos para operaciones de base de datos
- Disponibilidad: 99.5% con manejo robusto de errores y reconexión automática
- Seguridad: Autenticación JWT, encriptación bcrypt, validación de entrada de datos
- Escalabilidad: Arquitectura modular con ORM para facilitar crecimiento de usuarios
- Usabilidad: Interfaz intuitiva en español con iconografía clara para categorías

1.4 Vista de Casos de uso de la aplicación

Nombre del caso de uso	Descripción de la funcionalidad	Ruta del Caso de Uso
Registro de usuario	Permite crear una nueva cuenta en la aplicación con datos básicos y validaciones de seguridad	• Validación de formato de correo electrónico • Verificación de contraseña mínimo 8 caracteres • Validación de campos obligatorios (primer_nombre, primer_apellido) • Verificación de unicidad de correo en base de datos • Encriptación de contraseña con bcrypt • Creación del usuario en la base de datos • Redirección al inicio de sesión tras registro exitoso
Inicio de sesión	Permite el acceso a usuarios registrados mediante autenticación.	• Validación de formato de correo electrónico • Comparación de contraseña con hash almacenado • Generación de token de autenticación • Validación de credenciales y acceso al panel principal • Manejo de errores por acceso inválido
Visualización de gráficos	Muestra estadísticas visuales de los gastos registrados.	• Cálculo y agrupación de gastos por categoría • Generación de gráficos a partir de los datos ingresados • Visualización dentro del panel principal
Registro de ingreso/gasto	Permite registrar nuevos ingresos o gastos clasificados por categoría.	• Validación de autenticación • Ingreso de monto y categoría • Registro del movimiento en la base de datos • Visualización en historial y gráficos

Consulta de historial de gastos	Muestra al usuario un listado cronológico de sus ingresos y egresos.	• Consulta de registros financieros del usuario • Ordenamiento por fecha descendente • Visualización de datos detallados (fecha, categoría, monto) • Requiere autenticación

2. Tecnología

2.1 Desarrollo

Frontend (Aplicación Móvil)

- **React Native:** Framework de desarrollo móvil multiplataforma que permite crear aplicaciones nativas para iOS y Android utilizando JavaScript y React. Proporciona componentes nativos optimizados y acceso a APIs del dispositivo, garantizando un rendimiento cercano al nativo mientras mantiene una base de código única.

- **Expo:** Plataforma y conjunto de herramientas que simplifica el desarrollo con React Native. Ofrece un entorno de desarrollo rápido, acceso a APIs nativas sin configuración compleja, y facilita el proceso de construcción y distribución de la aplicación.

- **TypeScript:** Superset de JavaScript que añade tipado estático al lenguaje. Mejora la calidad del código, reduce errores en tiempo de ejecución y proporciona mejor autocompletado e IntelliSense durante el desarrollo.

- **React Navigation:** Biblioteca de navegación para React Native que maneja el enrutamiento y la navegación entre pantallas. Soporta diferentes tipos de navegadores (Stack, Tab) y mantiene el estado de navegación de forma eficiente.

- **Axios:** Cliente HTTP basado en promesas para JavaScript que facilita las peticiones a la API REST. Proporciona interceptores, transformación automática de datos JSON y manejo robusto de errores.

Backend (API REST)

- **Node.js**: Entorno de ejecución de JavaScript del lado del servidor que permite construir aplicaciones web escalables y de alto rendimiento. Utiliza un modelo de E/O no bloqueante y basado en eventos.

- **Express.js:** Framework web minimalista y flexible para Node.js que proporciona un conjunto robusto de características para aplicaciones web y móviles. Facilita la creación de APIs REST con un sistema de enrutamiento sencillo y middleware extensible.

- **Sequelize ORM:** Object-Relational Mapping (ORM) para Node.js que soporta múltiples bases de datos SQL. Proporciona una abstracción de alto nivel para interactuar con MySQL, incluyendo definición de modelos, migraciones, validaciones y relaciones entre entidades.

- **MySQL:** Sistema de gestión de base de datos relacional que almacena de forma persistente toda la información de usuarios, gastos, categorías y presupuestos, garantizando integridad referencial y consistencia de datos.

- **bcryptjs:** Biblioteca de hashing para encriptar contraseñas de usuarios utilizando el algoritmo bcrypt con salt rounds configurables, garantizando la seguridad de las credenciales almacenadas. jsonwebtoken (JWT): Implementación de JSON Web Tokens para Node.js utilizada para autenticación stateless. Los tokens contienen información del usuario codificada y firmada digitalmente para verificar su autenticidad.

- **Multer:** Middleware para Express.js que maneja la subida de archivos multipart/form-data, utilizado específicamente para la gestión de imágenes de perfil de usuarios.

- **dayjs:** Biblioteca ligera para manipulación y formateo de fechas, utilizada para el manejo de rangos de fechas en las estadísticas de gastos.

2.2 Modelo Entidad Relación