Proyecto Control de Gastos Móvil

Yenner Snyder Alayon Benavides

Jonathan Andrés García Rodríguez

Docente:

Edwin Albeiro Ramos Villamil

Universidad Minuto de Dios

Computación Móvil

Bogotá, D.C

2025

Contenido

Resumen 3
Objetivo general 3
Objetivos específicos 3
Alcance 4
Arquitectura de la solución 4
Componentes principales 4
Modelo de datos (propuesto) 5
Casos de uso 5
UC-01 – Iniciar sesión 5
UC-02 – Autoregistro 5
UC-03 – Registrar transacción 6
UC-04 – Listar y filtrar gastos 7
UC-05 – Editar y eliminar transacción 7
UC-06 – Reportes y visualizaciones 8
Requerimientos funcionales (RF) 9
Requerimientos no funcionales (RNF) 9
Cronograma General 10
Cronograma Desarrollo 11
Conclusiones 15
Referencias 16

Resumen

Este documento presenta la propuesta y el plan de construcción de una aplicación móvil de gastos personales. La solución contempla un frontend en React Native y un backend en Node.js con una API REST. Se incluyen el objetivo general, objetivos específicos, alcance, arquitectura, casos de uso, requerimientos funcionales y no funcionales, cronograma, conclusiones y referencias.

Objetivo general

Desarrollar un sistema de información móvil que permita a los usuarios registrar, consultar y analizar sus ingresos y gastos personales, con visualizaciones y reportes, garantizando seguridad, accesibilidad y experiencia de uso óptima.

Objetivos específicos

- Implementar el módulo de autenticación (registro, inicio de sesión y recuperación de contraseña).
- Desarrollar el módulo de gastos (crear, editar, eliminar y listar ingresos/gastos).
- Implementar categorización, filtros por fechas y reporte mensual/anual con visualizaciones.
- Diseñar y exponer una API REST segura en Node.js para el consumo desde el frontend.
- Persistir datos en una base de datos relacional e implementar migraciones y validaciones.
- Realizar pruebas unitarias y de integración básicas y configurar el despliegue de pruebas.
- Usar Azure Devops o Jira o Asana para aplicar la metodología Scrum en el desarrollo del proyecto.
- Usar GitFlow para el versionamiento de nuestro aplicativo mobile

Alcance

- Plataforma objetivo: Android (emulador y dispositivo físico).
- Usuarios finales: personas interesadas en controlar sus finanzas personales.
- Funciones cubiertas en el MVP: autenticación, gestión de gastos, categorías, filtros y reportes básicos.
- Fuera de alcance (versión inicial): multi-moneda avanzada, OCR de facturas, sincronización multi-dispositivo.

Arquitectura de la solución

La solución adopta una arquitectura cliente–servidor. El cliente móvil se construirá con React Native (Expo) y gestionará estado, navegación . El servidor se implementará con Node.js y Express, exponiendo una API REST con autenticación basada en JWT. La base de datos recomendada para el backend es PostgreSQL por su robustez transaccional (alternativamente puede usarse MongoDB o MySql). Se contemplan capas: presentación (RN), negocio (servicios/validaciones en el backend) y datos (ORM).

Componentes principales

- Frontend (React Native): Expo Router, React Navigation, componentes accesibles, almacenamiento local (AsyncStorage/SQLite).
- Backend (Node.js/Express): controladores, servicios, validaciones Joi/Zod, autenticación JWT y protección de rutas.
- Base de datos (PostgreSQL o MYSQL): tablas para usuarios, categorías y gastos; migraciones con Prisma/Sequelize.
- Servicios transversales: manejo de errores, logs, pruebas, CI de desarrollo (GitHub Actions opcional).

Modelo de datos (propuesto)

Entidades claves: Usuario(id, nombre, email, hash_password, creado_en), Categoría(id, nombre, tipo[ingreso|gasto], color), Transacción(id, usuario_id, categoría_id, tipo[ingreso|gasto], monto, fecha, nota, creado_en, actualizado_en).

Casos de uso

UC-01 – Iniciar sesión

Actor principal	Usuario
Precondiciones	Usuario registrado y con conexión a internet.
Flujo básico	1) El usuario ingresa email y contraseña. 2) El sistema valida credenciales. 3) Se genera JWT y se redirige al inicio.
Flujos alternos	Credenciales inválidas, campos obligatorios vacíos, errores de red.
Postcondiciones	Usuario autenticado con sesión activa.
Reglas de negocio	Campos validados; contraseñas cifradas; límites razonables de monto y longitud de texto.

UC-02 – Autoregistro

Actor principal	Visitante
Precondiciones	Correo y contraseña válidos; aceptación de términos.

Flujo básico	1) El visitante ingresa datos requeridos. 2) El sistema valida y crea la cuenta. 3) Se envía confirmación y se inicia sesión.
Flujos alternos	Credenciales inválidas, campos obligatorios vacíos, errores de red.
Postcondiciones	Cuenta creada y usuario autenticado.
Reglas de negocio	Campos validados; contraseñas cifradas; límites razonables de monto y longitud de texto.

UC-03 – Registrar transacción

Actor principal	Usuario autenticado
Precondiciones	Sesión activa; al menos una categoría disponible.
Flujo básico	1) El usuario selecciona tipo (ingreso/gasto) y categoría. 2) Ingresa monto, fecha y nota. 3) Guarda; el sistema persiste la transacción.
Flujos alternos	Credenciales inválidas, campos obligatorios vacíos, errores de red.

Postcondiciones	Transacción registrada y visible en el historial y reportes.
Reglas de negocio	Campos validados; contraseñas cifradas; límites razonables de monto y longitud de texto.

UC-04 – Listar y filtrar gastos

Actor principal	Usuario autenticado
Precondiciones	Transacciones existentes.
Flujo básico	1) El usuario abre el historial. 2) Aplica filtros por rango de fechas, categoría y tipo. 3) Visualiza resultados y totales.
Flujos alternos	Credenciales inválidas, campos obligatorios vacíos, errores de red.
Postcondiciones	Listado filtrado y totales calculados.
Reglas de negocio	Campos validados; contraseñas cifradas; límites razonables de monto y longitud de texto.

UC-05 – Editar y eliminar transacción

Actor principal	Usuario autenticado

Precondiciones	Transacción válida perteneciente al usuario.
Flujo básico	1) Selecciona transacción. 2) Edita campos o ejecuta eliminación. 3) El sistema actualiza/borra y recalcula totales.
Flujos alternos	Credenciales inválidas, campos obligatorios vacíos, errores de red.
Postcondiciones	Transacción actualizada/eliminada correctamente.
Reglas de negocio	Campos validados; contraseñas cifradas; límites razonables de monto y longitud de texto.

UC-06 – Reportes y visualizaciones

Actor principal	Usuario autenticado
Precondiciones	Transacciones suficientes para análisis.
Flujo básico	1) El usuario abre reportes. 2) El sistema calcula totales por categoría/mes. 3) Muestra gráficos y resumen.
Flujos alternos	Credenciales inválidas, campos obligatorios vacíos, errores de red.

Postcondiciones	Reporte generado con indicadores clave.
Reglas de negocio	Campos validados; contraseñas cifradas; límites razonables de monto y longitud de texto.

Requerimientos funcionales (RF)

- RF-01: El sistema debe permitir el registro e inicio de sesión con email y contraseña.
- RF-02: El sistema debe permitir CRUD de gastos(ingresos/gastos).
- RF-03: El sistema debe permitir CRUD de categorías.
- RF-04: El sistema debe listar y filtrar transacciones por fecha, categoría y tipo.
- RF-05: El sistema debe generar reportes/resúmenes mensuales y por categoría.
- RF-06: La API debe exponer endpoints REST seguros (JWT) para todas las operaciones.
- RF-07: El frontend debe funcionar sin conexión para registrar transacciones y sincronizar al volver la red (básico).
- RF-08: El sistema debe validar montos numéricos, fechas y campos obligatorios.

Requerimientos no funcionales (RNF)

- RNF-01 (Usabilidad): Interfaz consistente y accesible (contraste AA, tamaños dinámicos, labels para lectores de pantalla).
- RNF-02 (Rendimiento): Respuesta de API < 500 ms en operaciones típicas (promedio).
- RNF-03 (Seguridad): Contraseñas cifradas (bcrypt); JWT con expiración; validación de entrada en servidor.

- RNF-04 (Confiabilidad): Manejo de errores y reintentos en sincronización; logs de errores.
- RNF-05 (Portabilidad): Aplicación Android 8.0+; compatible con emulador y dispositivo físico.
- RNF-06 (Mantenibilidad): Código en TypeScript/JS con linter; arquitectura modular; pruebas unitarias básicas.
- RNF-07 (Privacidad): Cumplimiento de buenas prácticas de protección de datos; mínimos datos personales.

Cronograma General

Fechas clave proporcionadas por la asignatura:

Hito / Fecha	Entregable y actividades
UNIDAD 01 – 14 Sep	Documento de proyecto: metodología, requisitos (RF/RNF), arquitectura y cronograma. Configuración de repositorio y entorno.
UNIDAD 02 – 05 Oct	Interfaces de la aplicación: diseño en Figma/Mockups. Flujo de navegación y validaciones de UI.
UNIDAD 03 – 18 Oct	Desarrollo móvil (interfaz) + API (autenticación, categorías, transacciones) + base de datos. Pruebas y demo.

Cronograma Desarrollo

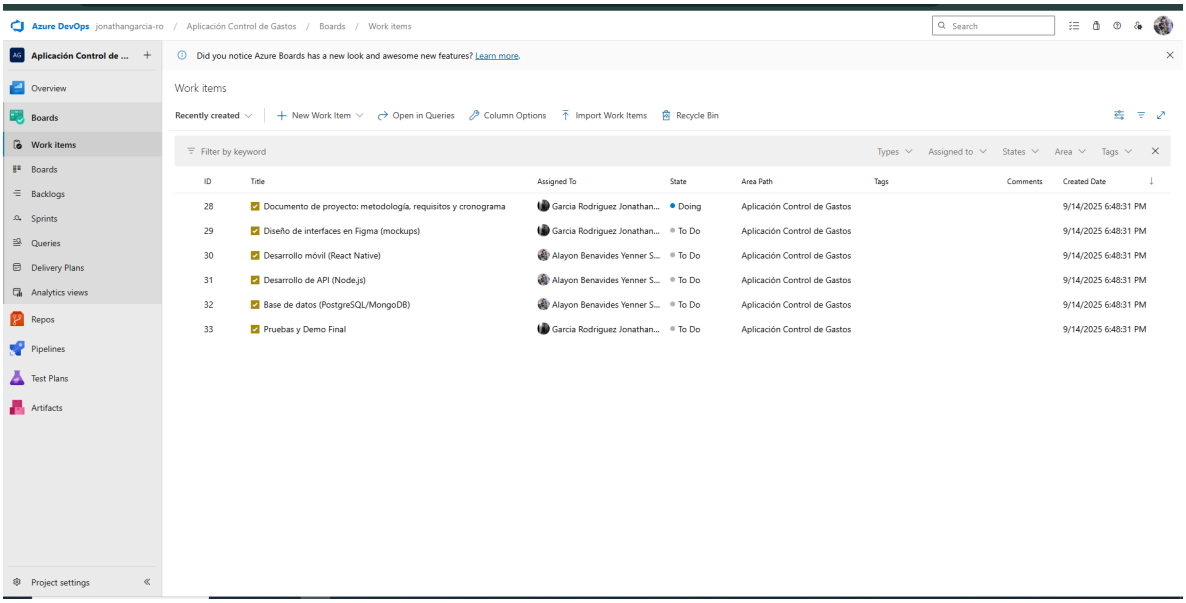

Figura 1. Tareas de Azure DevOps.

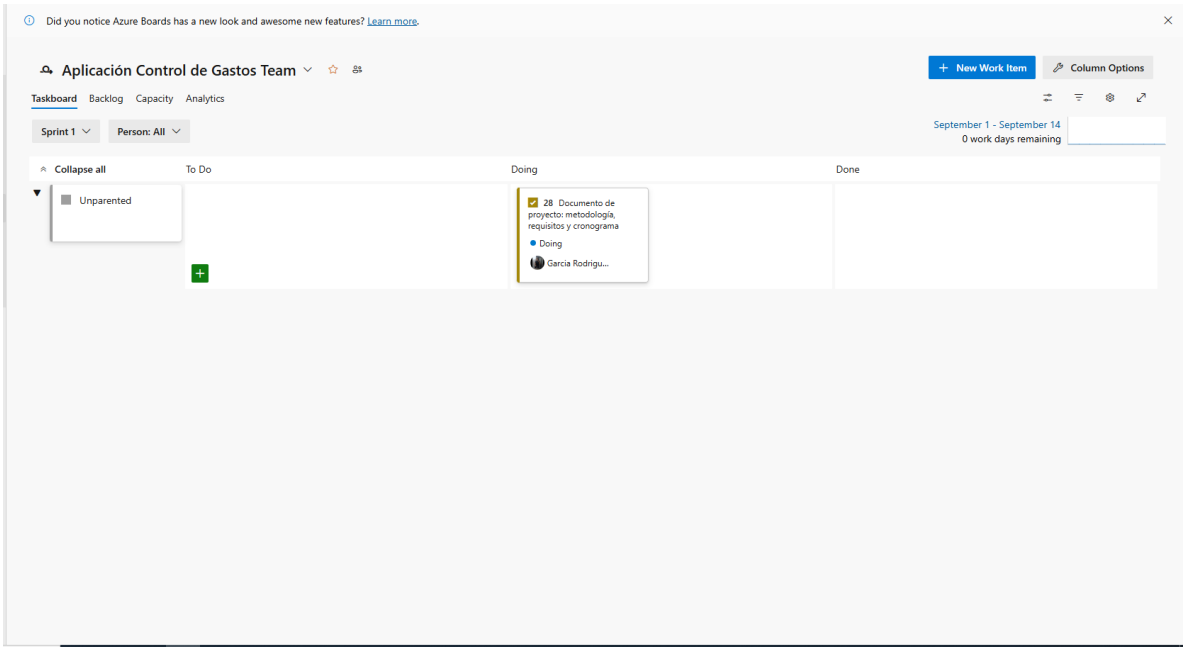

Figura 2. Sprint 1 en Azure DevOps.

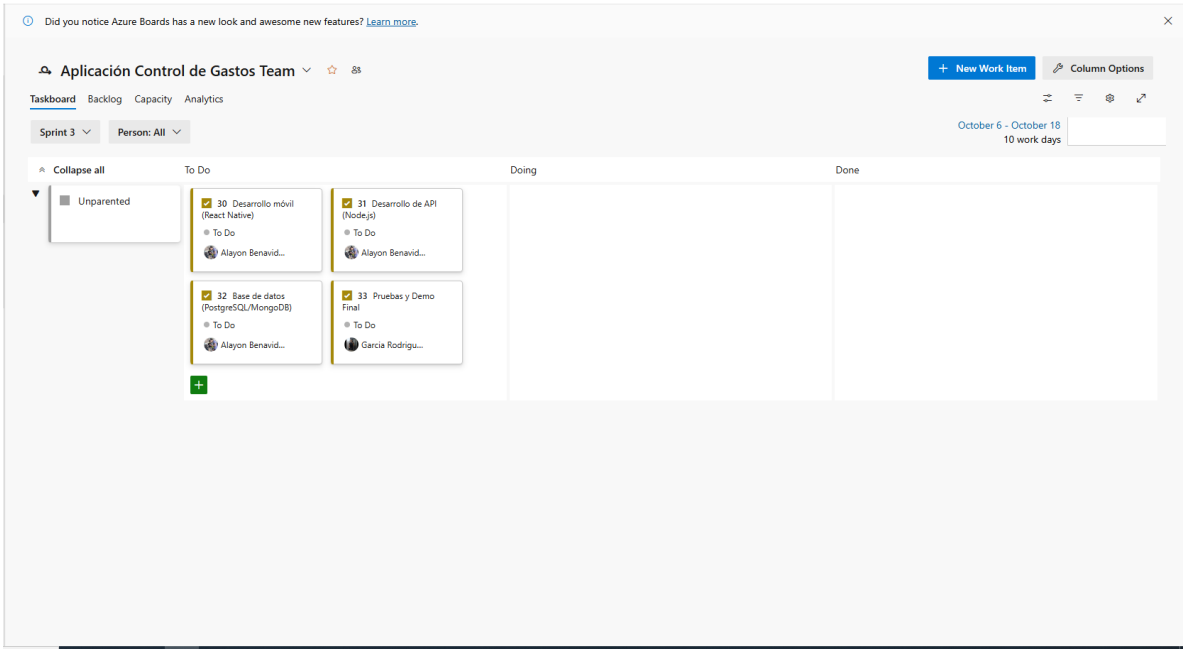

Figura 3. Sprint 3 Azure DevOps.

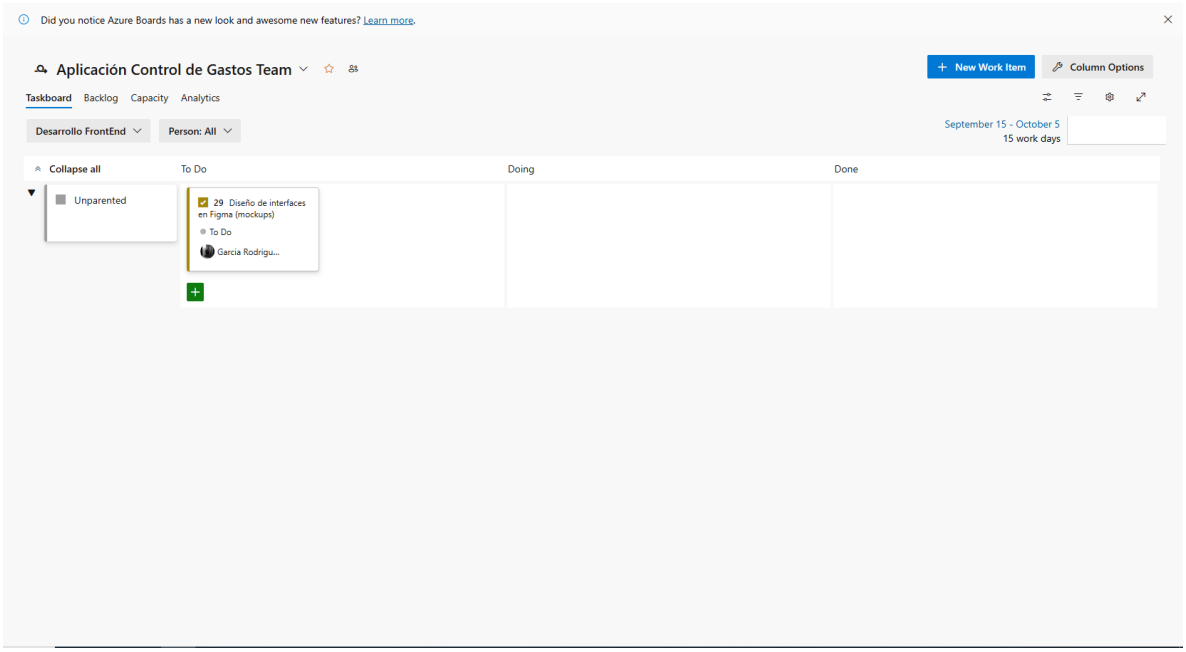

Figura 4. Sprint Desarrollo Frontend Azure DevOps.

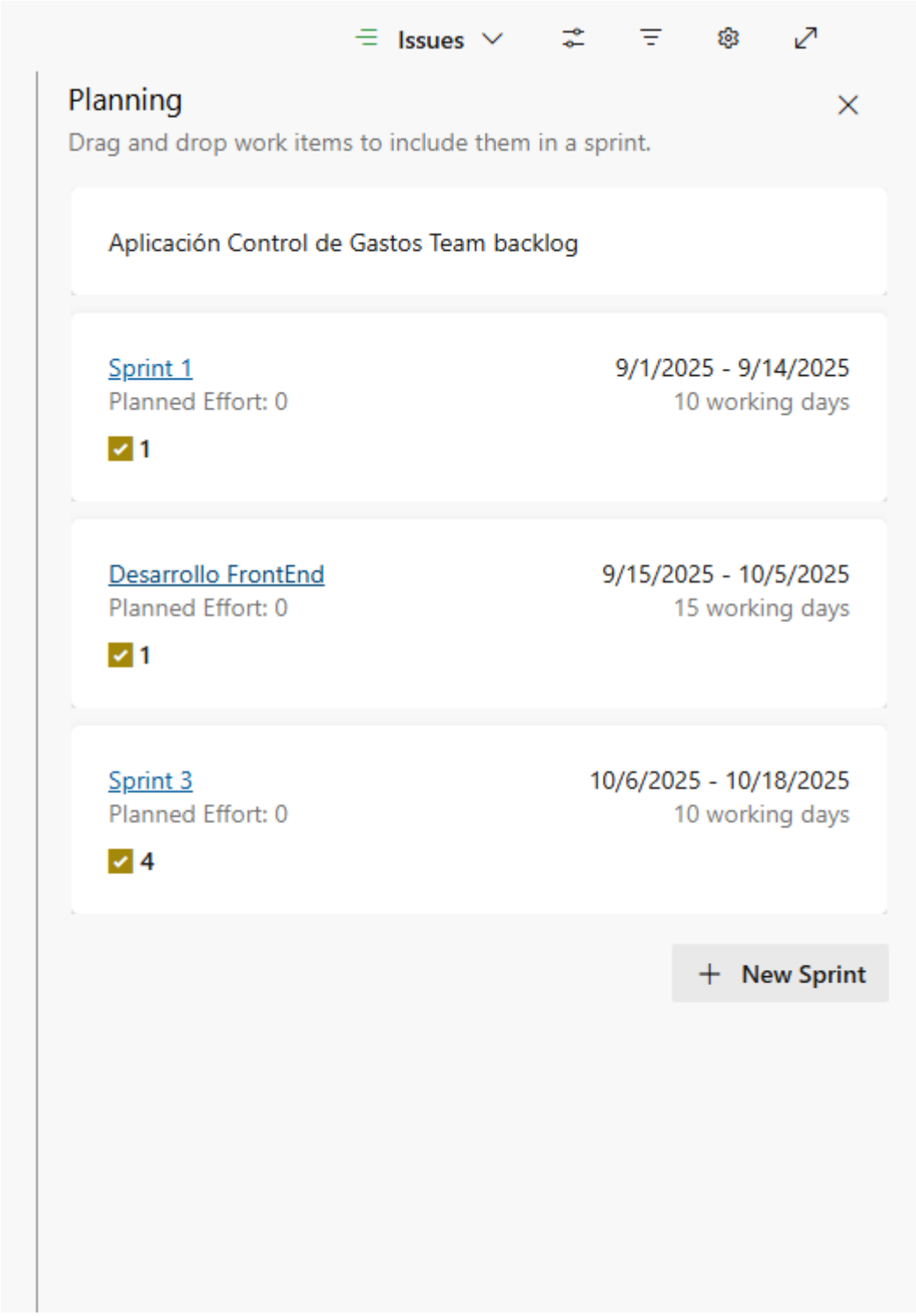

Figura 5. Planning Azure DevOps.

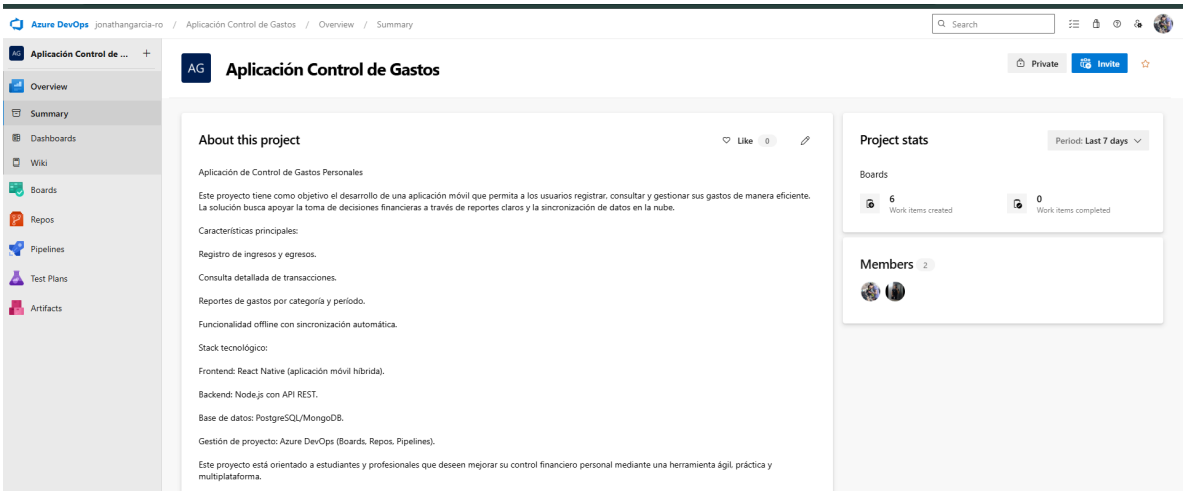

Figura 6. About this Project Azure DevOps.

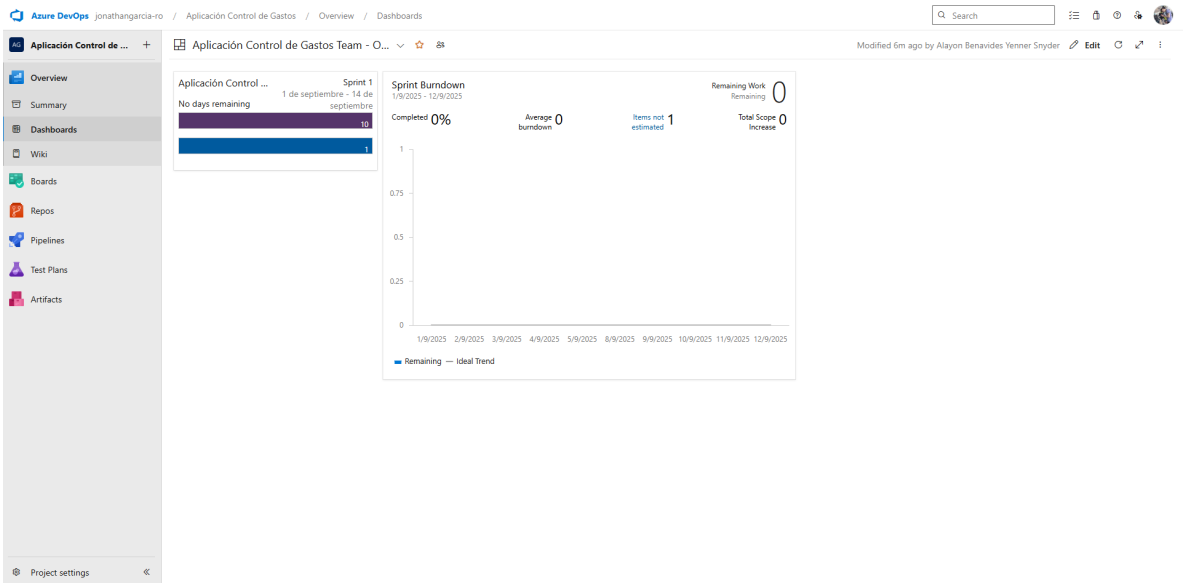

Figura 7. Dashboards Azure DevOps.

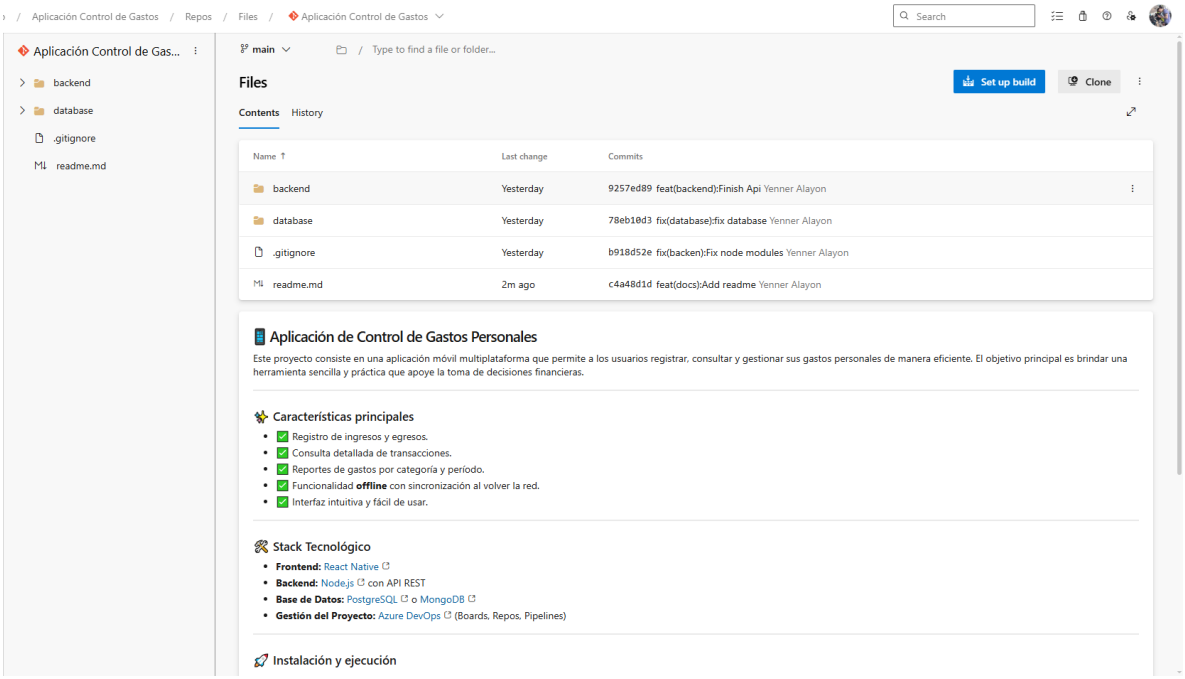

Figura 8. Repositorio Git.

Conclusiones

La aplicación de gastos personales propuesta ofrece una base sólida para gestionar finanzas individuales desde dispositivos Android. La combinación de React Native y Node.js permite un desarrollo ágil, mientras que la arquitectura modular y los requerimientos definidos facilitan la escalabilidad. La priorización de accesibilidad y seguridad contribuye a una solución inclusiva y confiable. El cronograma planteado viabiliza la entrega incremental de valor a lo largo del semestre.

Referencias

- Expo. (2025). Documentación de Expo. https://docs.expo.dev/
- Meta. (2025). React Native: documentación. https://reactnative.dev/
- Express.js. (2025). Guía oficial. https://expressjs.com/
- PostgreSQL Global Development Group. (2025). Documentación oficial. https://www.postgresql.org/docs/
- ISO/IEC 25010:2011. (2011). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE).
- Nielsen, J. (1994). 10 Usability Heuristics for User Interface Design. Nielsen Norman Group.