Tercer iteración:

Listado actualizado de endpoints de la API REST:

Método	URL	Descripción
GET	/api/auth/me	Obtener usuario identificado
POST	/api/auth/register	Registrar un nuevo usuario en el sistema
POST	/api/auth/login	Iniciar sesión con email y contraseña
POST	/api/auth/reset-password	Solicitar el cambio de contraseña
GET	/api/movements	Obtener la lista de movimientos del usuario
POST	/api/movements	Registrar un nuevo movimiento
PUT	/api/movements/{id}	Editar un movimiento existente
DELETE	/api/movements/{id}	Eliminar un movimiento
GET	/api/budget	Obtener el presupuesto mensual del usuario
POST	/api/budget	Crear un nuevo presupuesto mensual
GET	/api/statistics/income-vs-expense	Obtener datos para grafico de ingresos y gastos
GET	/api/statistics/category-distribution	Obtener datos para grafico de distribución de gastos
GET	/api/statistics/monthly-balance	Obtener datos para grafico de evolución del balance mensual
GET	/api/categories	Obtener lista de categorías
POST	/api/categories	Crear nueva categoría
PUT	/api/categories/{id}	Editar una categoría existente
DELETE	/api/categories/{id}	Eliminar una categoría
GET	/api/dashboard	Obtener resumen mensual: total de ingresos, gastos y balance.

Requerimientos Funcionales (actualizado):

RF1. El usuario debe poder ingresar un nuevo movimiento.

RF2. El usuario debe poder detallar en un movimiento el monto, tipo (ingreso/gasto), categoría, fecha y descripción.

RF3. El usuario debe poder crear una nueva categoría.

RF4. El usuario debe poder eliminar o editar una categoría anteriormente creada.

RF5. El usuario debe poder editar los datos de un movimiento ya registrado.

RF6. El usuario debe poder eliminar un movimiento ya creado.

RF7. El sistema muestra una lista de todos los movimientos en orden cronológico.

RF8. El usuario puede filtrar la lista de movimientos por mes, tipo y categoría.

RF9. El sistema debe mostrar en el centro de control, para un mes determinado, el total de ingresos, gastos y el balance.

RF10. El usuario debe poder crear presupuestos mensuales y ser notificado cuando esté cerca de superarlo o lo haya superado.

RF11. El sistema debe poder generar y mostrar los siguientes gráficos de un periodo determinado:
a. Un gráfico de barras que muestre el total de ingresos y gastos.
b. Un gráfico de torta que muestre la distribución de gastos por categoría.

RF12. El sistema debe generar y mostrar un gráfico de línea que represente el balance mensual a lo largo del tiempo.

RF13. Al agregarse, modificarse o eliminarse un movimiento, el sistema debe actualizar la lista de movimientos automáticamente.

RF14. El sistema debe guardar automáticamente todos los cambios y registros realizados por el usuario en la base de datos.

RF15. El sistema, al finalizar el mes, debe restablecer los gastos e ingresos para iniciar el nuevo periodo.

RF16. El usuario debe poder crearse una cuenta para el gestor de gastos con su email y contraseña.

RF17. El usuario debe poder ingresar a su cuenta con el email y contraseña previamente creados.

RF18. El sistema debe verificar que no existan dos cuentas con el mismo email.

RF19. El sistema debe permitir al usuario cambiar la contraseña si se la olvida.

Requerimientos no funcionales:

RNF1: Autenticación basada en JWT, para garantizar que solo usuarios autenticados accedan a recursos privados.

RNF2: Control de acceso por usuario, para evitar que un usuario pueda modificar lo de otro.

RNF3: Validación de datos: El sistema debe validar todos los datos recibidos desde el cliente en el backend, asegurando que cumplen con las reglas de formato, tipo y contenido esperadas. Esto incluye validaciones de campos obligatorios, tipos de dato correctos, valores dentro de rangos válidos, y control de longitud de texto, con el fin de garantizar la integridad de los datos y prevenir ataques como inyecciones o fallos lógicos.

Implementación:

En esta iteración se realizó el desarrollo del backend y frontend del gestor de gastos aplicando las tecnologías mencionadas anteriormente. Se implementaron los siguientes requerimientos funcionales:

RF1. El usuario debe poder ingresar un nuevo movimiento.

RF2. El usuario debe poder detallar en un movimiento el monto, tipo (ingreso/gasto), categoría, fecha y descripción.

RF3. El usuario debe poder crear una nueva categoría.

RF5. El usuario debe poder editar los datos de un movimiento ya registrado.

RF6. El usuario debe poder eliminar un movimiento ya creado.

RF7. El sistema muestra una lista de todos los movimientos en orden cronológico.

RF9. El sistema debe mostrar en el centro de control, para un mes determinado, el total de ingresos, gastos y el balance.

RF10. El usuario debe poder crear presupuestos mensuales y ser notificado cuando esté cerca de superarlo o lo haya superado.

RF13. Al agregarse, modificarse o eliminarse un movimiento, el sistema debe actualizar la lista de movimientos automáticamente.

RF16. El usuario debe poder crearse una cuenta para el gestor de gastos con su email y contraseña.

RF17. El usuario debe poder ingresar a su cuenta con el email y contraseña previamente creados.

RF18. El sistema debe verificar que no existan dos cuentas con el mismo email.

Y también se implementaron los requerimientos no funcionales:

RNF1: Autenticación basada en JWT, para garantizar que solo usuarios autenticados accedan a recursos privados.

RNF2: Control de acceso por usuario, para evitar que un usuario pueda modificar lo de otro.

RNF3: Validación de datos: El sistema debe validar todos los datos recibidos desde el cliente en el backend, asegurando que cumplen con las reglas de formato, tipo y contenido esperadas. Esto incluye validaciones de campos obligatorios, tipos de dato correctos, valores dentro de rangos válidos, y control de longitud de texto, con el fin de garantizar la integridad de los datos y prevenir ataques como inyecciones o fallos lógicos.

El hosting de la página se realizó en Versel para el frontend y en Render para el backend y la base de datos.