

# Sprint 4 – Apartado de Inversiones

---

**Periodo:** 20 octubre – 2 noviembre 2025 **Estado:** ☒ COMPLETADO

---

## Objetivo

Incorporar la gestión de inversiones y su rendimiento en la aplicación de finanzas personales "Ahorro".

---

## Entregables Completados

### 1. ☒ Interfaz para registrar inversiones

**Ubicación:** `public/views/newInvestment.html`

**Formulario implementado con los siguientes campos:**

- **Nombre:** Identificador de la inversión (ej: "Acciones Tesla")
- **Tipo de Inversión:** Categoría (ej: Acciones, Bonos, Criptomonedas)
- **Monto Inicial:** Capital invertido originalmente
- **Valor Actual:** Valor presente de la inversión
- **Fecha de Inicio:** Cuándo se realizó la inversión

**Características:**

- Validación de campos requeridos
  - Formato de entrada numérico con decimales
  - Selector de fecha
  - Botones de guardar y cancelar
  - Mensajes de error integrados
- 

### 2. ☒ Base de datos de inversiones

**Ubicación:** `src/models/investment.model.js`

**Modelo MongoDB (Mongoose Schema):**

```
{
  userId: ObjectId (referencia a User),
  name: String (requerido),
  type: String (requerido),
  initialAmount: Number (requerido),
  currentValue: Number (requerido),
  startDate: Date (requerido),
  timestamps: true (createdAt, updatedAt automáticos)
}
```

**Controlador de API:** `src/controllers/investment.controller.js`

### Endpoints implementados:

- `GET /api/investments` - Obtener todas las inversiones del usuario
- `GET /api/investments/:id` - Obtener una inversión específica
- `POST /api/investments` - Crear nueva inversión
- `PUT /api/investments/:id` - Actualizar inversión existente
- `DELETE /api/investments/:id` - Eliminar inversión

### Seguridad:

- Autenticación con JWT requerida
- Verificación de propiedad (solo el usuario puede ver sus inversiones)
- Validación de datos en servidor

---

## 3. ☒ Cálculo automático del rendimiento

**Ubicación:** `public/js/app.js` (función `displayInvestmentsCards`)

### Métricas calculadas:

#### Rendimiento Absoluto:

```
Rendimiento = Valor Actual - Monto Inicial
```

#### Rendimiento Porcentual:

```
Porcentaje = (Rendimiento / Monto Inicial) × 100
```

### Visualización:

- ☒ Rendimiento positivo → Color verde
- ☒ Rendimiento negativo → Color rojo
- Formato: `$1,500 (+30%)`

### Integración con Dashboard:

- Cálculo de inversiones totales
- Suma de `currentValue` de todas las inversiones
- Visualización en tarjeta amarilla del dashboard

---

## 4. ☒ Pruebas de funcionamiento y registro de fallos

### Casos de prueba ejecutados:

#	Caso de Prueba	Estado	Resultado
1	Crear inversión con datos válidos	<input checked="" type="checkbox"/> Pasó	Inversión creada correctamente
2	Crear inversión sin campos requeridos	<input checked="" type="checkbox"/> Pasó	Error de validación mostrado
3	Ver lista de inversiones	<input checked="" type="checkbox"/> Pasó	Listado correcto con cálculos
4	Eliminar inversión	<input checked="" type="checkbox"/> Pasó	Eliminación exitosa
5	Acceso no autorizado a inversión ajena	<input checked="" type="checkbox"/> Pasó	Error 401 retornado
6	Cálculo de rendimiento positivo	<input checked="" type="checkbox"/> Pasó	Porcentaje correcto (verde)
7	Cálculo de rendimiento negativo	<input checked="" type="checkbox"/> Pasó	Porcentaje correcto (rojo)
8	Visualización en dashboard	<input checked="" type="checkbox"/> Pasó	Total mostrado correctamente

Fallos detectados y resueltos:

- **✗ Fallo #1:** Formato de fecha no validado → ☒ Solucionado con validación HTML5
- **✗ Fallo #2:** Rendimiento con decimales irregulares → ☒ Solucionado con `.toFixed(2)`

Fallos pendientes: Ninguno

5. ☒ Documentación de mejoras

Archivos de código documentados:

- ☒ `src/models/investment.model.js` - Esquema con comentarios
- ☒ `src/controllers/investment.controller.js` - Endpoints con descripciones JSDoc
- ☒ `public/views/investments.html` - Estructura HTML comentada
- ☒ `public/js/app.js` - Funciones documentadas (initInvestments, displayInvestmentsCards)

Mejoras implementadas:

1. **Separación de responsabilidades:** Backend (API) y Frontend (visualización)
2. **Reutilización de código:** Funciones modulares en `app.js`
3. **Validación dual:** Cliente (HTML5) y Servidor (Mongoose)
4. **Feedback visual:** Colores para rendimiento positivo/negativo
5. **Responsividad:** Grid de Bootstrap para adaptación móvil

Arquitectura Técnica

Stack utilizado:

- **Backend:** Node.js + Express.js
- **Base de datos:** MongoDB + Mongoose
- **Frontend:** HTML5 + Bootstrap 5 + JavaScript Vanilla
- **Autenticación:** JWT (JSON Web Tokens)

Flujo de datos:

Usuario → Formulario HTML → API REST → MongoDB → Respuesta JSON → Actualización UI

## Capturas de Funcionalidad

### Vista de Inversiones:

- Lista de inversiones con tarjetas
- Cálculo de rendimiento en tiempo real
- Botones de acción (Eliminar)

### Formulario de Nueva Inversión:

- Campos de entrada validados
- Selector de fecha
- Mensajes de error

### Dashboard:

- Tarjeta "Inversiones Totales" con suma actualizada
- Integración con otras métricas financieras

## Conclusiones

- ☒ Todos los entregables del Sprint 4 fueron completados exitosamente.
- ☒ El módulo de inversiones está **completamente funcional** e integrado con el sistema.
- ☒ Se realizaron pruebas exhaustivas sin fallos pendientes.
- ☒ La documentación técnica está completa y actualizada.

## Próximos pasos

- Implementar edición de inversiones (actualizar valor actual)
- Agregar historial de cambios de valor
- Conectar inversiones con transacciones (opcional)

**Desarrollado por:** [Tu nombre] **Fecha de entrega:** 2 noviembre 2025 **Versión:** 1.0