

Fix: Validación Mejorada para Creación de Préstamos

Fecha: 31 de Octubre, 2025

Tipo: Corrección y Mejora

Módulo: Préstamos (Loans)

Problema Reportado

El usuario reportó un error al intentar crear un préstamo en la ruta `/admin/loans/new`.

Análisis del Problema

Al revisar el código, se identificaron las siguientes áreas de mejora:

- Validaciones Insuficientes:** El API route no validaba adecuadamente los campos vacíos o inválidos
- Mensajes de Error Genéricos:** Los errores no proporcionaban información específica sobre qué campo era problemático
- Falta de Logs:** No había logging detallado para debugging
- Validación de Tipos:** No se validaban correctamente los tipos de datos antes de procesarlos

Cambios Implementados

1. API Route Mejorado (`/app/api/loans/route.ts`)

Se agregaron las siguientes mejoras:

a) Logging Detallado

```
console.log('Datos recibidos para crear préstamo:', body);
console.log('Creando préstamo con datos validados:', { ... });
console.log('Préstamo creado exitosamente:', loan.id);
```

b) Validación de ClientId Vacío

```
if (typeof clientId === 'string' && clientId.trim() === '') {
  console.error('clientId está vacío');
  return NextResponse.json(
    { error: 'El ID del cliente no puede estar vacío' },
    { status: 400 }
  );
}
```

c) Validación de Campos Numéricos

```
const principal = parseFloat(principalAmount.toString());
const term = parseInt(termMonths.toString());
const rate = parseFloat(interestRate.toString());
const payment = parseFloat(monthlyPayment.toString());

if (isNaN(principal) || principal <= 0) {
  return NextResponse.json(
    { error: 'El monto principal debe ser un número positivo' },
    { status: 400 }
  );
}

// Similar validaciones para term, rate, y payment
```

d) Validación de Fechas

```
const start = new Date(startDate);
const end = new Date(endDate);

if (isNaN(start.getTime())) {
  return NextResponse.json(
    { error: 'La fecha de inicio no es válida' },
    { status: 400 }
  );
}

if (end <= start) {
  return NextResponse.json(
    { error: 'La fecha de fin debe ser posterior a la fecha de inicio' },
    { status: 400 }
  );
}
```

e) Validación de Enums










```
const validLoanTypes = ['PERSONAL', 'BUSINESS', 'MORTGAGE', 'AUTO', 'EDUCATION'];
if (!validLoanTypes.includes(loanType)) {
  return NextResponse.json(
    { error: 'Tipo de préstamo no válido' },
    { status: 400 }
  );
}

const validStatuses = ['ACTIVE', 'PAID_OFF', 'DEFAULTED', 'CANCELLED'];
if (!validStatuses.includes(status)) {
  return NextResponse.json(
    { error: 'Estado de préstamo no válido' },
    { status: 400 }
  );
}
```

f) Mensajes de Error Específicos

```
if (error.message.includes('Unique constraint')) {  
  return NextResponse.json(  
    { error: 'Ya existe un préstamo con este número' },  
    { status: 409 }  
  );  
}  
if (error.message.includes('Foreign key constraint')) {  
  return NextResponse.json(  
    { error: 'Referencia inválida. Verifica que el cliente existe.' },  
    { status: 400 }  
  );  
}
```

Validaciones Agregadas

1.  Validación de campos requeridos (no null/undefined)
2.  Validación de clientId no vacío
3.  Validación de valores numéricos positivos
4.  Validación de fechas válidas
5.  Validación de fechas lógicas (fin > inicio)
6.  Validación de enums (loanType, status)
7.  Validación de existencia de cliente
8.  Mensajes de error específicos y descriptivos
9.  Logging detallado para debugging

Beneficios







1. **Mejor UX:** Mensajes de error claros y específicos
2. **Debugging Facilitado:** Logs detallados en consola del servidor
3. **Prevención de Errores:** Validaciones robustas antes de operaciones de DB
4. **Mantenibilidad:** Código más legible y fácil de mantener
5. **Seguridad:** Validación exhaustiva de entrada de usuario

Archivos Modificados

- `/app/api/loans/route.ts` - Mejorado con validaciones robustas y logging

Pruebas Recomendadas

Después de este fix, se debe probar:

1.  Crear préstamo con datos válidos
2.  Intentar crear préstamo sin seleccionar cliente
3.  Intentar crear préstamo con montos inválidos (0, negativos, texto)
4.  Intentar crear préstamo con fechas inválidas
5.  Intentar crear préstamo con fecha de fin anterior a inicio
6.  Intentar crear préstamo con tipo inválido

7. ☒ Verificar que los logs aparecen correctamente en consola

Próximos Pasos

1. Desplegar cambios
 2. Monitorear logs del servidor al crear préstamos
 3. Reportar cualquier error específico para refinamiento adicional
 4. Considerar agregar validaciones similares en otros endpoints
-

Estado: ☒ Completado

Commit: Pendiente

Requiere Deploy: Sí