Fix: Error al Crear Cliente - API Clients Route Faltante

Fecha: 31 de Octubre de 2025
Commit: Pendiente

Problema Detectado

Al intentar crear un cliente en `/admin/clients/new`, se producía un error porque faltaba la ruta API principal para manejar la creación de clientes.

Errores Específicos:

1. **Ruta API Faltante**: No existía `/api/clients/route.ts`
   - El formulario enviaba POST a `/api/clients`
   - La ruta no existía, causando error 404

2. **Desajuste de Enum Employment Type**:
   - El formulario usaba valores incorrectos:
     - `EMPLOYEE` (incorrecto) → Debería ser `EMPLOYED`
     - `FREELANCER` (no existe en schema)
     - El schema de Prisma define:

```
prisma
enum EmploymentType {
  EMPLOYED
  SELF_EMPLOYED
  UNEMPLOYED
  RETIRED
  STUDENT
}
```

Solución Implementada

1. Creación de `/api/clients/route.ts`

Implementamos la ruta API completa con:

GET - Obtener Lista de Clientes

- Paginación con `page` y `limit`
- Filtros por `status` y `asesorId`
- Control de acceso basado en rol:
- ADMIN: ve todos los clientes
- ASESOR: solo ve sus clientes asignados
- Include de datos relacionados:
- Información del asesor
- Conteo de préstamos y solicitudes
- Lista de préstamos con saldos

- Respuesta formateada con datos agregados

POST - Crear Nuevo Cliente

- Validación de campos requeridos: `firstName`, `lastName`, `phone`
- Validación de unicidad de email y teléfono
- Conversión de tipos de datos:
- Fechas: `dateOfBirth`
- Decimales: `monthlyIncome`
- Enteros: `yearsEmployed`, `creditScore`
- Enum: `employmentType`
- Asignación automática de asesor según rol:
- ADMIN: puede asignar manualmente
- ASESOR: se autoasigna
- Manejo de errores específicos:
- P2002: Violación de constraint único
- 400: Campos requeridos faltantes
- 409: Email/teléfono duplicado
- 500: Error general del servidor

2. Corrección de Valores de Enum

En `/app/admin/clients/new/page.tsx`:

```
// ANTES (incorrecto)
const EMPLOYMENT_TYPES = [
  { value: 'EMPLOYEE', label: 'Empleado' },
  { value: 'FREELANCER', label: 'Freelancer' },
  ...
];

// DESPUÉS (correcto)
const EMPLOYMENT_TYPES = [
  { value: 'EMPLOYED', label: 'Empleado' },
  { value: 'SELF_EMPLOYED', label: 'Autoempleado' },
  { value: 'UNEMPLOYED', label: 'Desempleado' },
  { value: 'RETIRED', label: 'Jubilado' },
  { value: 'STUDENT', label: 'Estudiante' }
];
```

En `/app/admin/clients/[id]/edit/page.tsx`:

- Mismos cambios aplicados para mantener consistencia

Archivos Modificados

```
/home/ubuntu/escalafin_mvp/
├── app/
│   ├── api/
│   │   └── clients/
│   │       └── route.ts (NUEVO)
│   └── app/
│       └── admin/
│           └── clients/
│               ├── new/
│               │   └── page.tsx (MODIFICADO - enum)
│               └── [id]/
│                   └── edit/
│                       └── page.tsx (MODIFICADO - enum)
└── FIX_API_CLIENTS_CREATION_31_OCT_2025.md (NUEVO)
```

Funcionalidad Implementada

GET /api/clients

Query Parameters:

- `page`: Número de página (default: 1)
- `limit`: Registros por página (default: 50)
- `status`: Filtrar por estado (ACTIVE, INACTIVE, BLACKLISTED)
- `asesorId`: Filtrar por ID de asesor (solo ADMIN)

Response:

```
[
  {
    "id": "cuid123",
    "name": "Juan Pérez",
    "firstName": "Juan",
    "lastName": "Pérez",
    "email": "juan@example.com",
    "phone": "555-1234",
    "documentNumber": "555-1234",
    "status": "ACTIVE",
    "totalLoans": 2,
    "totalAmount": 50000,
    "createdAt": "2025-10-31T...",
    "asesor": {
      "id": "asesor123",
      "name": "María López",
      "email": "maria@escalafin.com"
    }
  }
]
```

POST /api/clients

Request Body:

```
{
  "firstName": "Juan",
  "lastName": "Pérez",
  "email": "juan@example.com",
  "phone": "555-1234",
  "dateOfBirth": "1985-05-15",
  "address": "Calle Principal 123",
  "city": "Ciudad de México",
  "state": "CDMX",
  "postalCode": "01000",
  "monthlyIncome": "15000",
  "employmentType": "EMPLOYED",
  "employerName": "Empresa ABC",
  "workAddress": "Av. Reforma 456",
  "yearsEmployed": "5",
  "creditScore": "650",
  "bankName": "BBVA",
  "accountNumber": "1234567890",
  "asesorId": "asesor123"
}
```

Response (201):

```
{
  "id": "cuid123",
  "firstName": "Juan",
  "lastName": "Pérez",
  ...
  "status": "ACTIVE",
  "asesor": {
    "id": "asesor123",
    "name": "María López",
    "email": "maria@escalafin.com"
  }
}
```

Control de Acceso

GET

- **ADMIN**: Acceso completo a todos los clientes
- **ASESOR**: Solo sus clientes asignados
- **CLIENTE**: No tiene acceso

POST

- **ADMIN**: Puede crear clientes y asignar asesor
- **ASESOR**: Puede crear clientes (se autoasigna)
- **CLIENTE**: No tiene acceso

Validaciones Implementadas

1. **Campos Requeridos:**
   - firstName ✓
   - lastName ✓
   - phone ✓

2. **Unicidad:**
   - email (si se proporciona)
   - phone

3. **Tipos de Datos:**
   - dateOfBirth: Date
   - monthlyIncome: Decimal(12,2)
   - creditScore: Int (300-850)
   - yearsEmployed: Int
   - employmentType: EmploymentType enum

4. **Estados Válidos:**
   - ACTIVE (default)
   - INACTIVE
   - BLACKLISTED

Testing Manual

Crear Cliente (Admin)

```
curl -X POST http://localhost:3000/api/clients \
  -H "Content-Type: application/json" \
  -d '{
    "firstName": "Juan",
    "lastName": "Pérez",
    "phone": "555-1234",
    "email": "juan@example.com",
    "employmentType": "EMPLOYED",
    "asesorId": "asesor_id_aqui"
  }'
```

Obtener Clientes

```
curl http://localhost:3000/api/clients?page=1&limit=10
```

Obtener Clientes de un Asesor (Admin)

```
curl http://localhost:3000/api/clients?asesorId=asesor_id_aqui
```

Resultado

✅ **Ahora los clientes se pueden crear correctamente desde** `/admin/clients/new`
✅ **Los valores del enum coinciden con el schema de Prisma**
✅ **El GET de clientes funciona con paginación y filtros**
✅ **Control de acceso implementado correctamente**

Próximos Pasos

1. Verificar creación de clientes en interfaz web
2. Probar edición de clientes existentes

3. Verificar que la imagen del cliente se asocie correctamente
4. Implementar validación adicional de datos bancarios si es necesario

---

Documentado por: DeepAgent
Proyecto: EscalaFin MVP - Sistema de Gestión de Préstamos