Backend - Sistema de Control de Activos

Backend del sistema desarrollado con Node.js, Express y TypeScript.

🛠️ Stack Tecnológico

- **Runtime:** Node.js 18+
- **Framework:** Express.js
- **Lenguaje:** TypeScript
- **Base de Datos:** SQL Server
- **ORM:** Sequelize
- **Autenticación:** JWT + bcrypt

📁 Estructura de Carpetas

```
backend/
├── src/
│   ├── config/          # Configuraciones (DB, JWT)
│   ├── models/          # Modelos de Sequelize
│   ├── controllers/     # Controladores de rutas
│   ├── services/        # Lógica de negocio
│   ├── routes/          # Definición de rutas
│   ├── middlewares/     # Middlewares personalizados
│   ├── utils/           # Utilidades
│   ├── types/           # Tipos TypeScript
│   └── app.ts           # Archivo principal
├── uploads/             # Archivos subidos
├── logs/                # Logs de la aplicación
├── package.json
├── tsconfig.json
└── .env                 # Variables de entorno (crear desde .env.example)
```

🚀 Instalación

Paso 1: Instalar dependencias

```bash
npm install
```

Paso 2: Configurar variables de entorno

Copia el archivo `.env.example` a `.env`:

```bash
copy .env.example .env
```

Edita el archivo `.env` con tus configuraciones:

```
PORT=5000
NODE_ENV=development

# Base de Datos
DB_HOST=localhost
DB_PORT=1433
DB_NAME=SistemaActivos
DB_USER=sa
DB_PASSWORD=tu_password_aqui

# JWT
JWT_SECRET=cambia_esto_por_una_clave_segura
JWT_EXPIRES_IN=8h

# Frontend
FRONTEND_URL=http://localhost:5173
```

Paso 3: Configurar SQL Server

1. Asegúrate de tener SQL Server instalado y corriendo
2. Crea la base de datos:

```sql
CREATE DATABASE SistemaActivos;
```

3. Ejecuta los scripts de creación de tablas (en la carpeta database/)

Paso 4: Crear usuario administrador inicial

Ejecuta este SQL en tu base de datos:

```sql
```

```
INSERT INTO Usuarios (Nombre, Email, Password, Rol, Estado, FechaCreacion)
VALUES (
  'Administrador',
  'admin@sistema.com',
  '$2b$10$ejemplo_hash_bcrypt',  -- Se hasheará automáticamente
  'Admin',
  1,
  GETDATE()
);
```

O usa el endpoint de registro cuando esté disponible.

🏃 Ejecución

Desarrollo (con hot reload)

```
bash

npm run dev
```

Compilar TypeScript

```
bash

npm run build
```

Producción

```
bash

npm start
```

📡 Endpoints Disponibles (Fase 1)

Autenticación

- `POST /api/auth/login` - Login de usuario
- `GET /api/auth/me` - Obtener usuario actual (requiere autenticación)
- `POST /api/auth/logout` - Logout

Health Check

- `GET /` - Info de la API

- `GET /health` - Estado del servidor

🔐 Autenticación

El sistema usa JWT (JSON Web Tokens) para autenticación:

1. Login con email y contraseña
2. Recibe token JWT
3. Incluye token en header: `Authorization: Bearer <token>`

Ejemplo:

```javascript
// Login
POST /api/auth/login
{
  "email": "admin@sistema.com",
  "password": "tu_password"
}

// Respuesta
{
  "success": true,
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "usuario": {
    "UsuarioID": 1,
    "Nombre": "Administrador",
    "Email": "admin@sistema.com",
    "Rol": "Admin"
  }
}

// Usar token en siguiente request
GET /api/auth/me
Headers: {
  "Authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

👥 Roles de Usuario

- **Admin:** Acceso total
- **Contador:** Acceso a reportes financieros/tributarios
- **Operador:** Registro de activos y gastos

- **Consulta:** Solo lectura

🐛 Troubleshooting

Error: Cannot connect to database

- Verifica que SQL Server esté corriendo
- Confirma las credenciales en `.env`
- Verifica el puerto (1433 por defecto)

Error: Module not found

```bash
npm install
```

Error: Port already in use

Cambia el puerto en `.env` o mata el proceso:

```bash
# Windows
netstat -ano | findstr :5000
taskkill /PID <PID> /F
```

📝 Próximos Pasos (Fase 2)

- CRUD de vehículos
- CRUD de propiedades
- Sistema de gastos
- Carga de archivos

Estado: ✅ Fase 1 - Semanas 1-2 completada