API de Gestão de Cheque Especial - Node.js

📋 Visão Geral

Esta é uma API completa para gestão de cheque especial bancário desenvolvida em **Node.js** com as seguintes funcionalidades:

- **Autenticação entre aplicações** via JWT
- **Consulta de limites** por CPF/CNPJ
- **Painel administrativo** para gestão de clientes
- **Sistema de logs** para auditoria
- **Validações de segurança** robustas

🏗️ Arquitetura

Tecnologias Utilizadas

- **Backend**: Node.js + Express.js
- **Banco de Dados**: MySQL (produção) / SQLite (desenvolvimento)
- **ORM**: Sequelize
- **Autenticação**: JWT + bcrypt
- **Frontend**: HTML5 + CSS3 + JavaScript (Vanilla)
- **Containerização**: Docker + Docker Compose

Estrutura do Projeto

```
cheque-especial-nodejs/
├── src/
│   ├── config/
│   │   └── database.js          # Configuração do banco
│   ├── models/
│   │   └── index.js             # Modelos Sequelize
│   ├── routes/
│   │   ├── auth.js              # Rotas de autenticação
│   │   ├── overdraft.js         # API de cheque especial
│   │   └── admin.js             # Rotas administrativas
│   ├── middleware/
│   │   └── auth.js              # Middleware de autenticação
```

```
│   ├── utils/
│   │   └── validators.js        # Validadores (CPF/CNPJ)
│   └── app.js                   # Aplicação principal
├── public/
│   └── index.html               # Frontend integrado
├── docker-compose.yml           # Configuração Docker
├── Dockerfile                   # Imagem da aplicação
├── init.sql                     # Script de inicialização MySQL
├── populate-db.js               # Script para popular banco
├── package.json                 # Dependências Node.js
└── .env                         # Variáveis de ambiente
```

🚀 Funcionalidades Implementadas

1. Autenticação entre Aplicações

- **Endpoint**: `POST /api/auth/token`
- **Método**: Client Credentials (OAuth2-like)
- **Segurança**: JWT com expiração configurável
- **Rate Limiting**: Proteção contra ataques

2. API de Consulta de Limite

- **Endpoint**: `POST /api/overdraft/check`
- **Parâmetros**: `document` (CPF/CNPJ)
- **Resposta**: `account_limit`, `updated_date`, `status`
- **Validações**: CPF/CNPJ válidos

3. Painel Administrativo

- **Login seguro** com sessões
- **Busca de clientes** por documento
- **Listagem paginada** de clientes
- **Atualização de limites** com senha de operação
- **Visualização de logs** em tempo real

4. Sistema de Logs

- **Auditoria completa** de alterações
- **Rastreamento de usuários** responsáveis
- **Timestamps precisos** de operações
- **Status de operações** (sucesso/erro)

🔧 Instalação e Configuração

Pré-requisitos

- **Node.js** 18+
- **npm** ou **yarn**
- **Docker** e **Docker Compose** (opcional)
- **MySQL** 8.0+ (para produção)

Instalação Local

1. **Clone o projeto**:

```
git clone <repository-url>
cd cheque-especial-nodejs
```

1. **Instale as dependências**:

```
npm install
```

1. **Configure as variáveis de ambiente**:

```
cp .env.example .env
# Edite o arquivo .env com suas configurações
```

1. **Configure o banco de dados**:

Para **desenvolvimento** (SQLite):

```
# O banco SQLite será criado automaticamente
npm run populate-db
```

Para **produção** (MySQL):

```
# Inicie o MySQL via Docker
docker-compose up mysql -d

# Aguarde o MySQL inicializar e execute
npm run populate-db
```

1. **Inicie a aplicação**:

```
npm start
# ou para desenvolvimento com auto-reload
npm run dev
```

Instalação com Docker

1. **Inicie todos os serviços**:

```
docker-compose up -d
```

1. **Acesse a aplicação**:
2. **Frontend**: http://localhost:3000
3. **API**: http://localhost:3000/api
4. **Health Check**: http://localhost:3000/health

📚 Documentação da API

Autenticação de Aplicações

Obter Token de Acesso

```
POST /api/auth/token
Content-Type: application/json

{
  "client_id": "bank_app_001",
  "client_secret": "secret_key_123"
}
```

Resposta de Sucesso:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

API de Cheque Especial

Consultar Limite do Cliente

```
POST /api/overdraft/check
Content-Type: application/json
Authorization: Bearer {access_token}

{
  "document": "12345678901"
}
```

Resposta de Sucesso:

```
{
  "status": "success",
  "account_limit": 5000.00,
  "updated_date": "2025-06-17T16:30:00Z"
}
```

Resposta de Erro:

```
{
  "status": "error",
  "error": "Cliente não encontrado",
  "code": "CLIENT_NOT_FOUND"
}
```

API Administrativa

Login do Administrador

```
POST /api/admin/login
Content-Type: application/json

{
  "username": "admin",
  "password": "admin123"
}
```

Buscar Cliente

```
POST /api/admin/client/search
Content-Type: application/json
```

```
Cookie: session_id={session_cookie}

{
  "document": "12345678901"
}
```

Atualizar Limite

```
POST /api/admin/client/update-limit
Content-Type: application/json
Cookie: session_id={session_cookie}

{
  "document": "12345678901",
  "new_limit": 7500.00,
  "operation_password": "12345678"
}
```

Listar Clientes

```
GET /api/admin/clients?page=1&per_page=10
Cookie: session_id={session_cookie}
```

Obter Logs

```
GET /api/admin/logs?limit=10
Cookie: session_id={session_cookie}
```

🔒 Segurança

Medidas Implementadas

1. **Autenticação JWT** para APIs externas
2. **Sessões seguras** para painel administrativo
3. **Rate Limiting** para prevenir ataques
4. **Validação rigorosa** de entrada
5. **Hash bcrypt** para senhas
6. **CORS configurado** adequadamente
7. **Headers de segurança** (Helmet.js)

Configurações de Segurança

```
// Rate Limiting
app.use('/api', rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutos
  max: 100 // máximo 100 requests por IP
}));

// Helmet para headers de segurança
app.use(helmet());

// CORS configurado
app.use(cors({
  origin: process.env.ALLOWED_ORIGINS?.split(',') || ['http://
localhost:3000'],
  credentials: true
}));
```

🧪 Testes

Credenciais de Teste

Administrador: - Usuário: `admin` - Senha: `admin123` - Senha de Operação: `12345678`

API: - Client ID: `bank_app_001` - Client Secret: `secret_key_123`

Clientes de Exemplo: - CPF: `12345678901` (João Silva Santos) - Limite: R$ 5.000,00 - CPF: `98765432100` (Maria Oliveira Costa) - Limite: R$ 3.000,00 - CNPJ: `12345678000195` (Empresa ABC Ltda) - Limite: R$ 15.000,00

Testando a API

1. **Obter token**:

```
curl -X POST http://localhost:3000/api/auth/token \
  -H "Content-Type: application/json" \
  -d
'{"client_id":"bank_app_001","client_secret":"secret_key_123"}'
```

1. **Consultar limite**:

```
curl -X POST http://localhost:3000/api/overdraft/check \
  -H "Content-Type: application/json" \
```

```
  -H "Authorization: Bearer {token}" \
  -d '{"document":"12345678901"}'
```

📊 Monitoramento

Health Check

```
GET /health
```

Retorna status da aplicação e conexão com banco de dados.

Logs da Aplicação

A aplicação gera logs estruturados para: - Requisições HTTP - Erros de aplicação - Operações de banco de dados - Tentativas de autenticação

🔧 Configuração de Produção

Variáveis de Ambiente

```
# Banco de Dados
DB_HOST=mysql-server
DB_PORT=3306
DB_NAME=cheque_especial
DB_USER=root
DB_PASSWORD=your-secure-password

# Aplicação
PORT=3000
NODE_ENV=production

# Segurança
JWT_SECRET=your-super-secret-jwt-key-256-bits
SESSION_SECRET=your-super-secret-session-key-256-bits
BCRYPT_ROUNDS=12

# Rate Limiting
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=100
```

Deploy com Docker

1. **Configurar variáveis de produção**:

```
cp .env.example .env.production
# Editar com valores de produção
```

1. **Build e deploy**:

```
docker-compose -f docker-compose.prod.yml up -d
```

Deploy Manual

1. **Instalar dependências**:

```
npm ci --only=production
```

1. **Configurar processo manager**:

```
npm install -g pm2
pm2 start src/app.js --name "cheque-especial-api"
pm2 startup
pm2 save
```

🐛 Troubleshooting

Problemas Comuns

1. Erro de Conexão com Banco

```
Error: getaddrinfo ENOTFOUND mysql
```

Solução: Verificar se o MySQL está rodando e as credenciais estão corretas.

2. Token JWT Inválido

```
{"error": "Token inválido"}
```

Solução: Verificar se o token não expirou e se está sendo enviado corretamente no header Authorization.

3. Senha de Operação Incorreta

```
{"error": "Senha de operação inválida"}
```

Solução: Verificar se a senha tem exatamente 8 dígitos numéricos.

4. Cliente Não Encontrado

```
{"error": "Cliente não encontrado"}
```

Solução: Verificar se o CPF/CNPJ está correto e se o cliente existe no banco de dados.

Logs de Debug

Para habilitar logs detalhados:

```
NODE_ENV=development npm start
```

📈 Performance

Otimizações Implementadas

1. **Connection Pooling** no Sequelize
2. **Rate Limiting** para prevenir sobrecarga
3. **Índices de banco** otimizados
4. **Compressão gzip** habilitada
5. **Cache de sessões** em memória

Métricas Recomendadas

- **Tempo de resposta** da API
- **Taxa de erro** por endpoint
- **Uso de CPU e memória**
- **Conexões ativas** no banco
- **Rate limit hits** por IP

🔄 Backup e Recuperação

Backup do Banco de Dados

```
# MySQL
mysqldump -u root -p cheque_especial > backup.sql

# SQLite
cp database.sqlite backup_$(date +%Y%m%d_%H%M%S).sqlite
```

Restauração

```
# MySQL
mysql -u root -p cheque_especial < backup.sql

# SQLite
cp backup_20250617_162500.sqlite database.sqlite
```

🤝 Contribuição

Estrutura de Desenvolvimento

1. **Fork** o repositório
2. **Crie** uma branch para sua feature
3. **Implemente** as mudanças
4. **Teste** localmente
5. **Submeta** um Pull Request

Padrões de Código

- **ESLint** para linting
- **Prettier** para formatação
- **Conventional Commits** para mensagens
- **Testes unitários** obrigatórios

📄 Licença

Este projeto está licenciado sob a **MIT License**.

📞 Suporte

Para suporte técnico:

- **Email**: suporte@flcbank.com.br
- **Documentação**: [Wiki do Projeto]
- **Issues**: [GitHub Issues]

📋 Checklist de Implementação

✅ Funcionalidades Principais

- [x] Autenticação JWT entre aplicações
- [x] API de consulta de limite por CPF/CNPJ
- [x] Painel administrativo com login
- [x] Busca e listagem de clientes
- [x] Atualização de limites com senha de operação
- [x] Sistema de logs de alterações
- [x] Validação de CPF/CNPJ
- [x] Rate limiting e segurança

✅ Tecnologias

- [x] Node.js + Express.js
- [x] MySQL com Sequelize ORM
- [x] JWT para autenticação de API
- [x] Sessões para painel admin
- [x] bcrypt para hash de senhas
- [x] Docker e Docker Compose
- [x] Frontend integrado

✅ Segurança

- [x] Validação de entrada rigorosa
- [x] Rate limiting configurado
- [x] Headers de segurança (Helmet)
- [x] CORS configurado adequadamente
- [x] Senhas hasheadas com bcrypt
- [x] JWT com expiração
- [x] Sessões seguras

✅ Qualidade

- [x] Código bem estruturado e documentado
- [x] Tratamento de erros robusto
- [x] Logs estruturados
- [x] Health check endpoint
- [x] Configuração via variáveis de ambiente
- [x] Scripts de população de dados
- [x] Documentação completa

Desenvolvido com ❤️ para FLC Bank