# SGCA Backend - Sistema de Gestão de Calendário Acadêmico

Backend em Node.js/Express com TypeScript para o Sistema de Gestão de Calendário Acadêmico.

## 🚀 Tecnologias

- **Node.js** - Runtime JavaScript
- **Express.js** - Framework web
- **TypeScript** - Tipagem estática
- **PostgreSQL** - Banco de dados
- **pg** - Driver PostgreSQL para Node.js
- **CORS** - Cross-Origin Resource Sharing
- **Helmet** - Middleware de segurança
- **Morgan** - Logger HTTP

## 📋 Pré-requisitos

- Node.js 18+
- PostgreSQL 12+
- npm ou yarn

## 🔧 Instalação

1. **Clone o repositório e navegue para a pasta do backend:**
2. **Instale as dependências:**
3. **Configure as variáveis de ambiente:**
4. **Configure o banco de dados:**
    - Certifique-se de que o PostgreSQL está rodando
    - Execute o script SQL do banco de dados:
5. **Compile o projeto:**

## 🏃 Executando

### Desenvolvimento (com hot reload):

```bash
npm run dev
```

### Produção:

```bash
npm start
```

O servidor estará disponível em `http://localhost:3001`

## 📚 API Endpoints

### Health Check

- `GET /api/health` - Verificar status da API

### Anos Letivos (Períodos Acadêmicos)

- `GET /api/anos-letivos` - Listar todos os anos letivos
- `GET /api/anos-letivos/:id` - Buscar ano letivo por ID
- `POST /api/anos-letivos` - Criar novo ano letivo
- `PUT /api/anos-letivos/:id` - Atualizar ano letivo
- `DELETE /api/anos-letivos/:id` - Excluir ano letivo

### Calendários (Planos de Ensino)

- `GET /api/calendarios` - Listar todos os calendários
- `GET /api/calendarios/:id` - Buscar calendário por ID
- `POST /api/calendarios` - Criar novo calendário
- `PUT /api/calendarios/:id` - Atualizar calendário
- `DELETE /api/calendarios/:id` - Excluir calendário

## 📁 Estrutura do Projeto

```plain text
```

```
src/
├── config/          # Configurações (banco de dados, etc.)
├── controllers/     # Controladores da API
├── middleware/      # Middlewares customizados
├── models/          # Tipos TypeScript e modelos
├── routes/          # Definição das rotas
├── utils/           # Utilitários
└── index.ts         # Arquivo principal
```

## 🔒 Segurança

- **Helmet** - Headers de segurança HTTP
- **CORS** - Controle de origem cruzada
- **Validação de entrada** - Validação básica nos controllers
- **Tratamento de erros** - Middleware centralizado de erros

## 🧪 Testando a API

### Usando curl:

#### Health Check:

Bash

```bash
curl http://localhost:3001/api/health
```

#### Listar Anos Letivos:

Bash

```bash
curl http://localhost:3001/api/anos-letivos
```

#### Criar Ano Letivo:

Bash

```bash
curl -X POST http://localhost:3001/api/anos-letivos \
  -H "Content-Type: application/json" \
  -d '{
    "id_periodo": 2025,
    "nome_periodo": "Período Acadêmico 2025",
    "data_inicio": "2025-02-01",
```

```
    "data_fim": "2025-12-15"
  }'
```

## 🐛 Troubleshooting

### Erro de conexão com banco:

1. Verifique se o PostgreSQL está rodando
2. Confirme as credenciais no arquivo `.env`
3. Teste a conexão manualmente:

### Erro de CORS:

1. Verifique se `CORS_ORIGIN` no `.env` está correto
2. Para desenvolvimento, pode usar `*` (não recomendado para produção)

### Porta já em uso:

1. Altere a `PORT` no arquivo `.env`
2. Ou mate o processo que está usando a porta:

## 📝 Logs

Os logs são exibidos no console usando Morgan. Em produção, considere usar um sistema de logs mais robusto como Winston.

## 🚀 Deploy

Para deploy em produção:

1. Configure as variáveis de ambiente adequadas
2. Use um gerenciador de processos como PM2:

## 🤝 Contribuição

1. Faça um fork do projeto
2. Crie uma branch para sua feature
3. Commit suas mudanças
4. Push para a branch

5. Abra um Pull Request

## 📄 Licença

Este projeto está sob a licença ISC.