Guia de Integração - SGCA Frontend + Backend

Este guia explica como integrar o frontend React com o backend Node.js/Express que desenvolvemos.

📋 Visão Geral da Arquitetura

Frontend (React/Next.js) ⟷ Backend (Node.js/Express) ⟷ PostgreSQL

Porta 3000 Porta 3001 Porta 5432

🔧 Configuração do Ambiente

1. Configurar o Banco de Dados

Pré-requisitos:

- PostgreSQL instalado e rodando
- Acesso de administrador ao PostgreSQL

Passos:

1. **Conectar ao PostgreSQL:**

   psql -U postgres

2. **Executar o script do banco:**

   psql -U postgres -f trabalho_2_melhorado.sql

3. **Verificar se o banco foi criado:**

   \c sistema_academico_ext

   \dt

2. Configurar o Backend

Localização: /home/ubuntu/sgca-backend/

1. **Instalar dependências:**

cd sgca-backend

npm install

2. **Configurar variáveis de ambiente:**

   cp .env.example .env

   Editar .env com suas configurações:

   PORT=3001

   NODE_ENV=development

   DB_HOST=localhost

   DB_PORT=5432

   DB_NAME=sistema_academico_ext

   DB_USER=postgres

   DB_PASSWORD=SUA_SENHA_AQUI

   CORS_ORIGIN=http://localhost:3000

3. **Compilar e iniciar:**

   npm run build

   npm run dev

4. **Testar se está funcionando:**

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

3. Configurar o Frontend

Localização: /home/ubuntu/sgca-frontend/

1. **Instalar dependências:**

cd sgca-frontend

npm install

2. **Configurar variável de ambiente da API:**

   Criar arquivo .env.local:

   NEXT_PUBLIC_API_URL=http://localhost:3001/api

3. **Iniciar o frontend:**

   npm run dev

4. **Acessar no navegador:**

   http://localhost:3000

🔗 Integração Frontend ↔ Backend

Modificações Necessárias no Frontend

O frontend já possui uma estrutura de API em /lib/api.ts, mas está configurado para usar dados mockados. Precisamos ativá-la:

1. Verificar configuração da API

O arquivo /lib/api.ts já está configurado para usar a variável de ambiente:

```
const API_BASE_URL = process.env.NEXT_PUBLIC_API_URL ||
'http://localhost:3001/api'
```

2. Modificar os componentes para usar a API real

Exemplo - Página de Anos Letivos:

Localizar o arquivo /app/anos-letivos/page.tsx e modificar:

```
// Substituir dados mockados por chamadas reais à API

import { api } from '@/lib/api';
```

```
// No useEffect:
useEffect(() => {
  const fetchData = async () => {
    try {
      const response = await api.getAnosLetivos();
      setAnosLetivos(response.data);
    } catch (error) {
      console.error('Erro ao carregar anos letivos:', error);
    }
  };

  fetchData();
}, []);
```

3. Implementar tratamento de erros

Adicionar tratamento de erros em todos os componentes:

```
const [loading, setLoading] = useState(false);
const [error, setError] = useState<string | null>(null);
const handleApiCall = async (apiFunction: () => Promise<any>) => {
  setLoading(true);
  setError(null);
  try {
```

```
    const result = await apiFunction();

    return result;

  } catch (err) {

    setError(err instanceof Error ? err.message : 'Erro desconhecido');

    throw err;

  } finally {

    setLoading(false);

  }

};
```

🧪 Testando a Integração

1. Teste Manual

1. **Iniciar Backend:**

   ```
   cd sgca-backend

   npm run dev
   ```

2. **Iniciar Frontend:**

   ```
   cd sgca-frontend

   npm run dev
   ```

3. **Testar no navegador:**

   - Acessar http://localhost:3000
   - Navegar para "Anos Letivos"
   - Tentar criar, editar e excluir registros

2. Teste via API

Testar endpoints diretamente:

```
# Health check

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

# Listar anos letivos

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

# Criar ano letivo

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"

  }'
```

🚨 Problemas Comuns e Soluções

1. Erro de CORS

Sintoma: Erro no console do navegador sobre CORS

Solução:

- Verificar se `CORS_ORIGIN` no backend está correto
- Para desenvolvimento, pode usar `CORS_ORIGIN=*`

2. Erro de conexão com banco

Sintoma: Backend não inicia ou erro de conexão

Solução:

- Verificar se PostgreSQL está rodando
- Confirmar credenciais no .env
- Testar conexão manual: psql -U postgres -d sistema_academico_ext

3. Frontend não carrega dados

Sintoma: Páginas vazias ou dados mockados

Solução:

- Verificar se NEXT_PUBLIC_API_URL está configurado
- Verificar se backend está rodando na porta correta
- Verificar console do navegador para erros de rede

4. Erro 404 nas rotas da API

Sintoma: Erro 404 ao chamar endpoints

Solução:

- Verificar se as rotas estão implementadas no backend
- Confirmar se a URL base está correta
- Verificar logs do backend

📝 Próximos Passos

Endpoints ainda não implementados:

1. **Categorias de Datas** (/api/categorias-datas)
2. **Datas** (/api/datas)
3. **Equalização** (/api/equalizacao)
4. **Eventos Obrigatórios** (/api/eventos-obrigatorios)
5. **Prazos e Eventos** (/api/prazos-eventos)

Melhorias sugeridas:

1. **Validação de dados** - Implementar validação robusta com Joi ou Zod
2. **Autenticação** - Adicionar sistema de login se necessário
3. **Paginação** - Implementar paginação para listas grandes

4. **Cache** - Adicionar cache Redis para melhor performance
5. **Logs** - Implementar sistema de logs mais robusto
6. **Testes** - Adicionar testes unitários e de integração

🔧 Comandos Úteis

Backend:

```
# Desenvolvimento

npm run dev

# Produção

npm run build && npm start

# Verificar logs

tail -f logs/app.log
```

Frontend:

```
# Desenvolvimento

npm run dev

# Build para produção

npm run build

# Iniciar produção

npm start
```

Banco de Dados:

```
# Conectar ao banco

psql -U postgres -d sistema_academico_ext

# Backup

pg_dump -U postgres sistema_academico_ext > backup.sql
```

Restore

psql -U postgres -d sistema_academico_ext < backup.sql

📞 Suporte

Se encontrar problemas:

1. Verificar logs do backend e frontend
2. Confirmar se todas as dependências estão instaladas
3. Verificar se as portas não estão em conflito
4. Consultar a documentação dos endpoints na rota /api