Sistema de Gestão de Telecom

Apresentação

Guilherme, desenvolvedor full stack com 21 anos, focado em React, TypeScript e tecnologias modernas de front-end. Apesar de pouca vivência com .NET e C#, demonstra grande disposição para aprender e evoluir nessas tecnologias. O projeto foi desenvolvido em React, mas com a estrutura pensada para futura integração com .NET Core e PostgreSQL, conforme proposto no desafio. O sistema é funcional, organizado, com interface limpa, boas práticas e demonstra o compromisso do desenvolvedor com aprendizado constante e entrega de valor.

Descrição do Projeto

Sistema completo de gestão de telecomunicações para controle de contratos de operadoras, faturas mensais e notificações automáticas por e-mail sobre vencimentos. Inclui CRUDs, dashboards interativos com gráficos, painel de notificações e arquitetura preparada para produção.

Funcionalidades

Funcionalidades Principais

- **Cadastro de Operadoras**: CRUD completo (Vivo, Claro, TIM etc.)
- **Gestão de Contratos**: Associações com operadoras e controle de status
- **Registro de Faturas**: Cadastro mensal, com cálculo automático de gastos
- **Dashboard Interativo**: Gráficos de pizza e barras com filtros dinâmicos
- **Sistema de Notificações**: Verificação de vencimentos e alertas via e-mail

Sistema de Notificações

- Verificação automática a cada hora de contratos vencendo nos próximos 5 dias.
- Controle inteligente que evita envios duplicados.
- Painel interativo com histórico, contador de pendentes e envio manual ou em lote.
- Templates HTML com todos os dados do contrato.
- Sistema pronto para integração real com serviços de e-mail (SMTP).

Tecnologias Utilizadas

Front-End

- **React 18** + **TypeScript**
- **Tailwind CSS** (estilização responsiva)
- **Recharts** (gráficos nativos para React)
- **Lucide React** (ícones)
- **date-fns** (manipulação de datas)

Dados e Persistência

- **LocalStorage** para dados entre sessões (ideal para protótipos).
- Hooks customizados (`useLocalStorage`) para persistência reativa.
- Serialização JSON com suporte a objetos complexos e datas.
- Mock de dados pré-carregado, com estrutura pronta para migração a PostgreSQL.

Configuração de E-mail

Sistema preparado para integração com: - **SMTP via Nodemailer** - **SendGrid**, **Mailgun**, **AWS SES**

Exemplo de configuração:

```
interface EmailConfig {
  smtpHost: string;
  smtpPort: number;
  smtpUser: string;
  smtpPassword: string;
  fromEmail: string;
  fromName: string;
}
```

Integração para Produção

O serviço está preparado para integração com: - **Nodemailer** para SMTP - **SendGrid** para e-mails transacionais - **AWS SES** para alta disponibilidade - **Mailgun** para APIs robustas

Gráficos e Indicadores

A biblioteca **Recharts** foi utilizada por ser ideal para aplicações React. Ela oferece:

- Alta performance
- Responsividade
- Suporte completo ao TypeScript
- Baseada em D3.js

Gráficos implementados:

- **Pizza**: Distribuição das faturas por status (Paga, Pendente, Vencida)
- **Barras**: Evolução mensal de faturas emitidas e pagas
- **Indicadores (Cards)**: Total de faturas, total faturado, status atuais

Estratégia de Persistência

Enquanto o backend .NET não está implementado, os dados são persistidos localmente no **LocalStorage**:

- Acesso instantâneo e sem dependência de banco
- Ideal para protótipos rápidos e testes offline
- Serialização segura e validação de dados

Preparado para migração futura para:

- **PostgreSQL** com **EF Core**
- APIs REST com **.NET Core**
- Suporte a autenticação, backup, sincronização e CI/CD

Como Executar

Pré-requisitos

- Node.js 18+
- npm ou yarn

Instalação

```
# Clonar o repositório
git clone https://github.com/guiborges77/sistema-gestao-telecom.git

# Instalar dependências
npm install

# Executar em modo desenvolvimento
npm run dev

# Build para produção
npm run build
```

Acesso

- **Desenvolvimento**: http://localhost:5173
- **Produção**: Após build, servir pasta `dist`

Funcionalidades por Tela

Dashboard

- Cards com estatísticas principais
- Gráfico de pizza com distribuição de faturas

- Gráfico de barras com evolução mensal
- Indicadores visuais e cores consistentes

Operadoras

- Lista com busca e filtros
- Formulário de cadastro/edição
- Validação de campos obrigatórios
- Exclusão com confirmação

Contratos

- Associação com operadoras
- Controle de status (Ativo/Inativo)
- Datas de início e vencimento
- Valores mensais

Faturas

- Vinculação com contratos
- Status (Paga/Pendente/Vencida)
- Cálculo automático de totais
- Filtros por período e status

Notificações

- Painel flutuante com contador
- Lista de notificações pendentes
- Histórico de e-mails enviados
- Ações de envio individual ou em lote

Arquitetura do Código

Estrutura de Pastas

```
src/
├── components/          # Componentes React
│   ├── ui/             # Componentes base (Button, Input, etc.)
│   ├── layout/         # Layout e navegação
│   ├── dashboard/      # Componentes do dashboard
│   ├── operators/      # Gestão de operadoras
│   ├── contracts/      # Gestão de contratos
│   ├── invoices/       # Gestão de faturas
│   └── notifications/  # Sistema de notificações
├── hooks/              # Hooks customizados
├── services/           # Serviços (e-mail, API)
├── utils/              # Utilitários e helpers
├── types/              # Definições TypeScript
└── data/               # Dados mock e iniciais
```

Próximos Passos

Para Produção

1. **Backend**: Implementar API REST com .NET Core
2. **Banco de Dados**: Migrar para PostgreSQL
3. **Autenticação**: Sistema de login e permissões
4. **E-mail Real**: Integrar com serviço de e-mail
5. **Deploy**: Configurar CI/CD e hospedagem