🚀 Como Rodar o Dashboard FinOps

📋 Pré-requisitos

Antes de começar, certifique-se de ter instalado:

- **Node.js** versão 18 ou superior (Download aqui (https://nodejs.org/))
- **Yarn** package manager (Instalação (https://classic.yarnpkg.com/en/docs/install))
- **PostgreSQL** versão 12 ou superior (Download aqui (https://www.postgresql.org/download/))

🔧 Configuração Inicial

1. Preparar o Banco de Dados

Primeiro, crie um banco de dados PostgreSQL:

```
# Entre no PostgreSQL
psql -U postgres

# Crie o banco de dados
CREATE DATABASE finops_dashboard;

# Saia do PostgreSQL
\q
```

2. Configurar Variáveis de Ambiente

O dashboard já está configurado, mas se você clonar o projeto, crie um arquivo `.env` na pasta `nextjs_space`:

```
# Banco de dados PostgreSQL
DATABASE_URL="postgresql://usuario:senha@localhost:5432/finops_dashboard"

# URL da aplicação
NEXTAUTH_URL="http://localhost:3000"
```

⚠️ **IMPORTANTE**: Substitua `usuario` e `senha` pelos dados do seu PostgreSQL.

3. Instalar Dependências

Entre na pasta do projeto e instale as dependências:

```
cd /home/ubuntu/finops_dashboard/nextjs_space
yarn install
```

4. Configurar o Banco de Dados

Gere as tabelas no banco de dados:

```
yarn prisma generate
yarn prisma db push
```

Se quiser popular com dados de exemplo:

```
yarn prisma db seed
```

▶️ Como Rodar o Dashboard

Modo Desenvolvimento (Local)

Para rodar o dashboard localmente:

```
cd /home/ubuntu/finops_dashboard/nextjs_space
yarn dev
```

O dashboard estará disponível em: **http://localhost:3000**

Modo Produção

Para compilar e rodar em produção:

```
# Compilar o projeto
yarn build

# Rodar em produção
yarn start
```

🌐 Dashboard em Produção

O dashboard já está deployado e disponível em:

https://finops-dashboard-tfp0w0.abacusai.app

📂 Estrutura do Código

Arquivos Principais

```
nextjs_space/
├── app/                          # Páginas do Next.js
│   ├── layout.tsx                # Layout principal
│   ├── page.tsx                  # Página inicial (Dashboard)
│   ├── executive/                # Dashboard Executivo
│   ├── forecast/                 # Previsões
│   ├── recommendations/          # Recomendações
│   ├── modernization/            # Modernização
│   ├── compute/                  # Métricas de Computação
│   ├── storage/                  # Métricas de Storage
│   ├── upload/                   # Upload de dados
│   └── api/                      # APIs do backend
│
├── components/                   # Componentes React
│   ├── dashboard-layout.tsx      # Layout base do dashboard
│   ├── sidebar.tsx               # Menu lateral
│   ├── top-bar.tsx               # Barra superior
│   ├── dashboard-view.tsx        # Vista principal
│   ├── executive-dashboard.tsx   # Dashboard executivo
│   ├── forecast-view.tsx         # Vista de previsões
│   └── ui/                       # Componentes de UI (botões, cards, etc)
│
├── lib/                          # Bibliotecas e utilitários
│   ├── db.ts                     # Conexão com banco de dados
│   ├── types.ts                  # Tipos TypeScript
│   └── utils.ts                  # Funções auxiliares
│
├── prisma/                       # Configuração do banco
│   └── schema.prisma             # Schema do banco de dados
│
└── public/                       # Arquivos públicos
    ├── favicon.svg               # Ícone do site
    └── og-image.png              # Imagem para redes sociais
```

Como Funciona

1. **Next.js** - Framework React que roda tanto no servidor quanto no cliente
2. **Prisma** - ORM que se conecta ao PostgreSQL e gerencia os dados
3. **Tailwind CSS** - Framework CSS para estilização
4. **Recharts** - Biblioteca de gráficos
5. **ShadcN UI** - Componentes de interface prontos

Fluxo de Dados

```
Upload CSV → API de Upload → Validação →
Prisma → PostgreSQL → APIs de Dashboard →
Componentes React → Visualizações
```

🔄 Como Funciona o Upload de Dados

1. Acesse `/upload` no dashboard
2. Baixe o template CSV para AWS, Azure ou GCP

3. Preencha o template com seus dados
4. Faça o upload do arquivo
5. O sistema valida e salva no banco de dados
6. Os dados aparecem automaticamente nos dashboards

🛠️ Tecnologias Utilizadas

Frontend

- **Next.js 14** - Framework React
- **React 18** - Biblioteca de UI
- **TypeScript** - Tipagem estática
- **Tailwind CSS** - Estilos
- **Recharts** - Gráficos
- **Radix UI** - Componentes acessíveis

Backend

- **Next.js API Routes** - APIs serverless
- **Prisma ORM** - Gerenciamento de banco
- **PostgreSQL** - Banco de dados

Ferramentas

- **Yarn** - Gerenciador de pacotes
- **ESLint** - Linter de código
- **PostCSS** - Processamento de CSS

📊 Funcionalidades Principais

1. Dashboard Geral

- Visão consolidada de custos de todas as clouds
- Gráficos de tendências
- Métricas principais (gastos, economia, etc)

2. Dashboard Executivo

- KPIs executivos
- Comparação entre clouds
- ROI e eficiência

3. Previsões (Forecast)

- Previsão de custos futuros
- Análise de tendências
- Alertas de aumento de gastos

4. Recomendações

- Sugestões de otimização
- Potencial de economia
- Priorização de ações

5. Modernização

- Sugestões de modernização (EC2 → EKS, etc)
- Análise de infraestrutura legada
- Roadmap de modernização

6. Métricas Detalhadas

- Compute (CPU, instâncias, etc)
- Storage (S3, discos, backups)
- Por cloud (AWS, Azure, GCP)

🐛 Resolução de Problemas

Erro de Conexão com Banco

```
# Verifique se o PostgreSQL está rodando
sudo systemctl status postgresql

# Se não estiver, inicie:
sudo systemctl start postgresql
```

Erro de Dependências

```
# Limpe e reinstale
rm -rf node_modules
rm yarn.lock
yarn install
```

Erro de Build

```
# Limpe o cache do Next.js
rm -rf .next
yarn build
```

📞 Suporte

Se tiver dúvidas ou problemas:

1. Verifique os logs no terminal
2. Confira o arquivo `.env`
3. Confirme que o PostgreSQL está rodando
4. Veja a documentação do Next.js: https://nextjs.org/docs

Desenvolvido para otimização de custos em cloud computing (AWS, Azure, GCP)