🚀 Guia Completo de Deploy do DELIVEREI no Render

Este guia fornece instruções passo a passo para fazer o deploy do backend do DELIVEREI no Render (plano gratuito) e configurar o frontend no Netlify para se conectar ao backend em produção.

📋 Índice

🎯 Visão Geral

O DELIVEREI é um sistema multi-tenant de delivery composto por:

- **Frontend**: React + TypeScript (já deployado no Netlify)
- **Backend**: NestJS + PostgreSQL (será deployado no Render)
- **Banco de Dados**: PostgreSQL (será criado no Render)

Arquitetura Atual

```
Frontend (Netlify) → Backend (Render) → PostgreSQL (Render)
```

✅ Pré-requisitos

Antes de começar, você precisa:

- [ ] Conta no GitHub (com acesso ao repositório `nerdrico2025/deliverei-v1`)
- [ ] Conta no Netlify (já configurada com o frontend)
- [ ] Conta no Render (criaremos neste guia)
- [ ] Git instalado localmente (opcional, mas recomendado)

🔑 Passo 1: Criar Conta no Render

1.1 Acessar o Render

1. Acesse: https://render.com (https://render.com)
2. Clique em **“Get Started for Free”** (ou **“Sign Up”**)
3. Escolha uma das opções para criar sua conta:
   - **GitHub** (recomendado - facilita o deploy)
   - **GitLab**
   - **Email**

1.2 Conectar com GitHub

Se você escolheu GitHub:

1. Clique em **“Sign Up with GitHub”**
2. Autorize o Render a acessar sua conta do GitHub
3. Selecione os repositórios que o Render pode acessar:
   - Recomendação: Selecione apenas **“nerdrico2025/deliverei-v1”**
4. Clique em **“Install & Authorize”**

1.3 Confirmar Email

1. Verifique seu email
2. Clique no link de confirmação enviado pelo Render
3. Você será redirecionado para o Dashboard do Render

🗄️ Passo 2: Criar Banco de Dados PostgreSQL

2.1 Acessar a Criação de Database

1. No Dashboard do Render, clique em **“New +”** (canto superior direito)
2. Selecione **“PostgreSQL”**

2.2 Configurar o Database

Preencha os seguintes campos:

Campo	Valor	Descrição
Name	`deliverei-db`	Nome do banco de dados
Database	`deliverei`	Nome do schema principal
User	`deliverei_user`	Usuário do banco (automático)
Region	`Oregon (US West)`	Escolha a região mais próx-ima
PostgreSQL Version	`16`	Versão mais recente
Plan	**Free**	Plano gratuito (suficiente para testes)

2.3 Criar o Database

1. Clique em **“Create Database”**
2. Aguarde alguns segundos/minutos enquanto o Render provisiona o banco
3. Quando pronto, você verá a página de detalhes do database

2.4 Copiar as Credenciais

⚠️ **IMPORTANTE**: Guarde essas informações em um local seguro!

Na página do database, você encontrará:

Internal Database URL (para conexões dentro do Render)

```
postgres://deliverei_user:SENHA@dpg-XXXXX/deliverei
```

External Database URL (para conexões de fora do Render)

```
postgres://deliverei_user:SENHA@dpg-XXXXX-a.oregon-postgres.render.com/deliverei
```

Campos Individuais

- **Hostname**: `dpg-XXXXX-a.oregon-postgres.render.com`
- **Port**: `5432`
- **Database**: `deliverei`
- **Username**: `deliverei_user`
- **Password**: `XXXXXXXXXXXXXXXXX` (gerado automaticamente)

📝 **Copie a “Internal Database URL”** - você precisará dela no Passo 4!

🚀 Passo 3: Deploy do Backend

3.1 Criar um Web Service

1. No Dashboard do Render, clique em **“New +”**
2. Selecione **“Web Service”**

3.2 Conectar o Repositório

1. Se você já conectou o GitHub no Passo 1, verá seus repositórios
2. Procure por **“deliverei-v1”** ou **“nerdrico2025/deliverei-v1”**
3. Clique em **“Connect”** ao lado do repositório

3.3 Configurar o Web Service

Preencha os seguintes campos:

Campo	Valor	Descrição
Name	`deliverei-backend`	Nome do serviço (será usado na URL)
Region	`Oregon (US West)`	Mesma região do database
Branch	`main`	Branch a ser deployado
Root Directory	`backend`	⚠️ **IMPORTANTE**: Pasta do backend!
Runtime	`Node`	Detectado automaticamente
Build Command	`npm install && npm run build && npx prisma generate && npx prisma migrate deploy`	Comando para build
Start Command	`npm run start:prod`	Comando para iniciar o servidor
Plan	**Free**	Plano gratuito

3.4 Configurações Avançadas (antes de criar)

NÃO clique em “Create Web Service” ainda! Antes, role para baixo e expanda **“Advanced”**:

Environment Variables (adicionar depois)

Por enquanto, apenas anote que você precisará adicionar variáveis de ambiente no próximo passo.

Auto-Deploy

- ✅ Deixe marcado: **“Yes”** para “Auto-Deploy”
- Isso fará deploy automático quando você fizer push para a branch `main`

3.5 Criar o Web Service

1. Revise todas as configurações
2. Clique em **“Create Web Service”**
3. O Render começará o primeiro deploy automaticamente
4. **Aguarde** - o primeiro deploy pode levar de 5 a 10 minutos

3.6 Acompanhar o Deploy

Você verá logs em tempo real mostrando:
- Download das dependências (`npm install`)
- Build do código (`npm run build`)
- Geração do Prisma Client (`npx prisma generate`)
- Execução das migrações (`npx prisma migrate deploy`)
- Inicialização do servidor

⚠️ **IMPORTANTE**: O primeiro deploy provavelmente falhará porque ainda não configuramos as variáveis de ambiente! Isso é normal, continue para o Passo 4.

⚙️ Passo 4: Configurar Variáveis de Ambiente

4.1 Acessar Environment Variables

1. Na página do seu Web Service (`deliverei-backend`)
2. No menu lateral esquerdo, clique em **“Environment”**
3. Clique em **“Add Environment Variable”**

4.2 Adicionar Variáveis Obrigatórias

Adicione as seguintes variáveis uma por uma:

🔐 Database (usando Internal Database URL)

Key	Value	Descrição
`DATABASE_URL`	`[Internal Database URL do Passo 2.4]`	URL de conexão com o banco
`DIRECT_URL`	`[Internal Database URL do Passo 2.4]`	URL direta (mesmo valor)

Exemplo:

```
DATABASE_URL=postgresql://deliverei_user:SENHA@dpg-XXXXX/deliverei
DIRECT_URL=postgresql://deliverei_user:SENHA@dpg-XXXXX/deliverei
```

⚠️ **Use a “Internal Database URL”** (sem `-a` no hostname), não a “External”!

🔒 JWT Secrets

Key	Value	Descrição
`JWT_SECRET`	`[Gere uma string aleatória segura]`	Chave para assinar tokens JWT
`JWT_EXPIRES_IN`	`15m`	Tempo de expiração do access token
`JWT_REFRESH_SECRET`	`[Gere outra string aleatória segura]`	Chave para refresh tokens
`JWT_REFRESH_EXPIRES_IN`	`7d`	Tempo de expiração do refresh token

Como gerar strings seguras:
- Opção 1: Use um gerador online: randomkeygen.com (https://randomkeygen.com/)
- Opção 2: Execute no terminal:

```
bash
  node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"
```

Exemplo:

```
JWT_SECRET=a8f5f167f44f4964e6c998dee827110c03e9e4c5e3e6e7e3e7e3e7e3e7e3e7e3
JWT_EXPIRES_IN=15m
JWT_REFRESH_SECRET=b9g6g268g55g5075f7d009fef938220d04f0f5d6f4f7f8f4f8f4f8f4f8f4f8f4
JWT_REFRESH_EXPIRES_IN=7d
```

🌐 CORS e Frontend

Key	Value	Descrição
`CORS_ORIGIN`	`https://deliver-ei.netlify.app,https://de-liverei.com.br,http://localhost:5173`	URLs permitidas para CORS
`FRONTEND_URL`	`https://deliver-ei.netlify.app`	URL principal do frontend

🔧 Node Environment

Key	Value	Descrição
`NODE_ENV`	`production`	Ambiente de produção
`PORT`	`10000`	Porta (Render usa 10000 por padrão)

4.3 Variáveis Opcionais (Adicionar Depois)

Você pode adicionar essas variáveis depois, quando for configurar integrações:

💳 Stripe (Pagamentos com Cartão)

```
STRIPE_SECRET_KEY=sk_live_... ou sk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...
STRIPE_PRICE_BASICO=price_...
STRIPE_PRICE_PROFISSIONAL=price_...
STRIPE_PRICE_ENTERPRISE=price_...
```

💰 Asaas (Pagamentos PIX/Boleto)

```
ASAAS_API_KEY=your_asaas_api_key
ASAAS_WEBHOOK_TOKEN=your_webhook_token
```

📱 WhatsApp Business API

```
WHATSAPP_API_URL=https://graph.facebook.com/v17.0
WHATSAPP_PHONE_NUMBER_ID=your_phone_number_id
WHATSAPP_ACCESS_TOKEN=your_access_token
```

4.4 Salvar e Re-deploy

1. Após adicionar todas as variáveis obrigatórias, clique em **“Save Changes”**
2. O Render iniciará um novo deploy automaticamente
3. Aguarde o deploy completar (5-10 minutos)

🧪 Passo 5: Testar o Backend

5.1 Verificar Status do Deploy

1. Aguarde até que o status mude para **“Live”** (bolinha verde)
2. Copie a URL do seu backend (será algo como):
   `https://deliverei-backend.onrender.com`

5.2 Testar Endpoints Públicos

Abra o navegador ou use o terminal para testar:

Teste 1: Health Check (se existir)

```
curl https://deliverei-backend.onrender.com/api
```

Resposta esperada: Mensagem de boas-vindas ou informações da API

Teste 2: Documentação Swagger (se configurada)

```
https://deliverei-backend.onrender.com/api-docs
```

5.3 Testar Login de Super Admin

Se você já executou o seed do banco de dados, teste o login:

```
curl -X POST https://deliverei-backend.onrender.com/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "superadmin@deliverei.com",
    "senha": "super123"
  }'
```

Resposta esperada:

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "...",
    "email": "superadmin@deliverei.com",
    "nome": "Super Administrador",
    "role": "SUPER_ADMIN"
  }
}
```

5.4 Executar Seed (se necessário)

Se o banco estiver vazio, você pode executar o seed de duas formas:

Opção A: Via Render Shell

1. No Dashboard do Render, vá para seu Web Service
2. No menu superior, clique em **“Shell”**
3. Execute:
   `bash`
   `cd backend`
   `npm run prisma:seed`

Opção B: Via Script Remoto (se disponível)

```
curl -X POST https://deliverei-backend.onrender.com/api/seed
```

🎨 Passo 6: Atualizar Frontend

Agora vamos atualizar o frontend para usar variáveis de ambiente em vez de URLs hardcoded.

6.1 Clonar o Repositório Localmente

```
cd ~
git clone https://github.com/nerdrico2025/deliverei-v1.git
cd deliverei-v1
```

6.2 Criar Arquivo `.env.example` no Frontend

Crie o arquivo na raiz do projeto:

```
nano .env.example
```

Adicione o seguinte conteúdo:

```
# Backend API URL
VITE_API_URL=http://localhost:3000/api

# Modo de desenvolvimento
VITE_DEV_MODE=true
```

Salve com `Ctrl+O`, depois `Enter`, e saia com `Ctrl+X`.

6.3 Criar Arquivo `.env.production.example`

```
nano .env.production.example
```

Adicione:

```
# Backend API URL - Production
VITE_API_URL=https://deliverei-backend.onrender.com/api

# Modo de produção
VITE_DEV_MODE=false
```

6.4 Atualizar `src/services/apiClient.ts`

Abra o arquivo:

```
nano src/services/apiClient.ts
```

Substitua o conteúdo por:

```
import axios from 'axios';

// Usar variável de ambiente ou fallback para localhost
const baseURL = import.meta.env.VITE_API_URL || 'http://localhost:3000/api';

const apiClient = axios.create({
  baseURL,
  headers: {
    'Content-Type': 'application/json',
  },
});

// Interceptor para adicionar token e tenant slug
apiClient.interceptors.request.use((config) => {
  const token = localStorage.getItem('deliverei_token');
  const tenantSlug = localStorage.getItem('deliverei_tenant_slug');

  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }
  if (tenantSlug) {
    config.headers['x-tenant-slug'] = tenantSlug;
  }

  return config;
});

// Interceptor para refresh token
apiClient.interceptors.response.use(
  (response) => response,
  async (error) => {
    const originalRequest = error.config;

    if (error.response?.status === 401 && !originalRequest._retry) {
      originalRequest._retry = true;

      const refreshToken = localStorage.getItem('deliverei_refresh_token');
      if (refreshToken) {
        try {
          const { data } = await axios.post(`${baseURL}/auth/refresh`, {
            refreshToken
          });

          localStorage.setItem('deliverei_token', data.accessToken);
          originalRequest.headers.Authorization = `Bearer ${data.accessToken}`;

          return apiClient.request(originalRequest);
        } catch (refreshError) {
          // Logout
          localStorage.removeItem('deliverei_token');
          localStorage.removeItem('deliverei_refresh_token');
          localStorage.removeItem('deliverei_tenant_slug');
          localStorage.removeItem('deliverei_auth');
          window.location.href = '/login';
          return Promise.reject(refreshError);
        }
      }
    }

    return Promise.reject(error);
  }
);
```

```
export default apiClient;
```

6.5 Commitar as Mudanças

```
git add .
git commit -m "feat: adicionar suporte a variáveis de ambiente para API URL"
git push origin main
```

🌐 Passo 7: Configurar Netlify

7.1 Acessar o Netlify

1. Acesse: https://app.netlify.com (https://app.netlify.com)
2. Faça login
3. Selecione seu site do DELIVEREI

7.2 Configurar Environment Variables

1. No menu lateral, clique em **“Site configuration”**
2. Clique em **“Environment variables”**
3. Clique em **“Add a variable”** → **“Add a single variable”**

Adicione as seguintes variáveis:

Key	Value	Descrição
`VITE_API_URL`	`https://deliverei-backend.onrender.com/api`	URL da API em produção
`VITE_DEV_MODE`	`false`	Modo de produção

7.3 Re-deploy do Frontend

Existem duas formas de fazer re-deploy:

Opção A: Trigger Deploy no Netlify

1. No Netlify, vá em **“Deploys”**
2. Clique em **“Trigger deploy”** → **“Clear cache and deploy site”**

Opção B: Push no GitHub (recomendado)

Como você já fez push das mudanças no Passo 6.5, o Netlify fará deploy automaticamente!

1. Aguarde alguns minutos
2. Verifique o status do deploy em **“Deploys”**

7.4 Verificar o Deploy

1. Quando o deploy estiver completo, acesse seu site: `https://deliverei.netlify.app`
2. Abra o **DevTools** do navegador (`F12`)
3. Vá para a aba **“Network”**

4. Tente fazer login ou qualquer ação que chame a API
5. Verifique se as requisições estão indo para:
   `https://deliverei-backend.onrender.com/api/...`

🔀 Passo 8: Merge do PR #27

8.1 Acessar o Pull Request

1. Acesse: https://github.com/nerdrico2025/deliverei-v1/pulls (https://github.com/nerdrico2025/deliverei-v1/pulls)
2. Clique no **PR #27** (melhorias no dashboard)

8.2 Revisar as Mudanças

1. Verifique os arquivos alterados
2. Leia a descrição do PR
3. Certifique-se de que não há conflitos

8.3 Fazer o Merge

Se você é o Administrador do Repositório:

1. Clique em **“Merge pull request”**
2. Escolha o tipo de merge:
   - **Create a merge commit** (recomendado)
   - Squash and merge
   - Rebase and merge
3. Clique em **“Confirm merge”**
4. Opcionalmente, delete a branch após o merge

Se você NÃO é o Administrador:

1. Solicite ao administrador para fazer o merge
2. Ou peça permissões de merge

8.4 Aguardar Deploy Automático

Após o merge:

1. O Netlify detectará o push na branch `main`
2. Iniciará o deploy automaticamente
3. Aguarde 2-5 minutos
4. Verifique o site atualizado

🔧 Solução de Problemas

Problema 1: Backend não inicia (falha no deploy)

Sintoma: Deploy falha com erros de build ou runtime

Soluções:

1. **Verificar logs do deploy**:
   - No Render, vá em **“Logs”**
   - Procure por erros em vermelho

2. **Variáveis de ambiente faltando**:
   - Verifique se todas as variáveis obrigatórias foram adicionadas
   - Especialmente `DATABASE_URL`, `JWT_SECRET`, etc.

3. **Erro nas migrações do Prisma**:
   `Error: P1001: Can't reach database server`
   - Verifique se a `DATABASE_URL` está correta
   - Use a **Internal Database URL**, não a External

4. **Build Command incorreto**:
   - Certifique-se de que o **Root Directory** está como `backend`
   - Build Command: `npm install && npm run build && npx prisma generate && npx prisma migrate deploy`

Problema 2: Frontend não conecta ao Backend

Sintoma: Erros de CORS ou conexão recusada

Soluções:

1. **Verificar variável `VITE_API_URL` no Netlify**:
   - Deve ser: `https://deliverei-backend.onrender.com/api`
   - **SEM barra no final!**

2. **Verificar CORS no backend**:
   - Certifique-se de que a variável `CORS_ORIGIN` inclui a URL do Netlify
   - Exemplo: `https://deliverei.netlify.app,https://deliverei.com.br`

3. **Limpar cache do navegador**:
   `bash`
   `# Ou no navegador: Ctrl+Shift+Delete`

4. **Re-deploy do frontend**:
   - No Netlify: **“Trigger deploy” → “Clear cache and deploy site”**

Problema 3: Erro 401 (Não autorizado)

Sintoma: Todas as requisições retornam 401

Soluções:

1. **Verificar JWT_SECRET**:
   - Certifique-se de que está configurado no Render
   - Token deve ser longo e aleatório

2. **Limpar localStorage**:
   `javascript`
   `// No console do navegador (F12)`
   `localStorage.clear();`

3. **Fazer login novamente**

Problema 4: Banco de Dados vazio

Sintoma: Não consegue fazer login, empresas não aparecem

Soluções:

1. **Executar seed**:
   ```
   bash
   # Via Render Shell
   cd backend
   npm run prisma:seed
   ```

2. **Verificar se as migrações foram executadas**:
   ```
   bash
   npx prisma migrate status
   ```

Problema 5: Render Free Tier "hiberna"

Sintoma: Primeira requisição demora muito ou falha

Explicação:
- O plano Free do Render hiberna o serviço após 15 minutos de inatividade
- A primeira requisição após hibernação pode demorar 30-60 segundos

Soluções:

1. **Upgrade para plano pago** (US$ 7/mês)
   - Não hiberna
   - Mais recursos

2. **Usar um serviço de "keep-alive"**:
   - UptimeRobot (https://uptimerobot.com/) (gratuito)
   - Faz ping no backend a cada 5 minutos

3. **Aceitar o comportamento** (para testes é aceitável)

Problema 6: Erros de conexão do Database

Sintoma: `Error: Can't reach database server`

Soluções:

1. **Verificar se o database está "Live"**:
   - No Dashboard do Render, vá para o PostgreSQL
   - Status deve estar verde

2. **Regenerar Internal Database URL**:
   - Às vezes, o Render muda o hostname
   - Copie a URL atualizada e atualize a variável `DATABASE_URL`

3. **Verificar se Web Service e Database estão na mesma região**

📚 Recursos Adicionais

Documentação Oficial

- Render Docs (https://render.com/docs)
- Render Deploy Guide - Node.js (https://render.com/docs/deploy-node-express-app)
- Prisma with Render (https://www.prisma.io/docs/guides/deployment/deployment-guides/deploying-to-render)
- Netlify Environment Variables (https://docs.netlify.com/environment-variables/overview/)

Comandos Úteis

Verificar logs do backend (Render):

1. Acesse o Dashboard do Render
2. Clique no Web Service
3. Vá em **“Logs”**

Acessar Shell do backend (Render):

1. Acesse o Dashboard do Render
2. Clique no Web Service
3. No menu superior, clique em **“Shell”**

Ver status das migrações:

```
npx prisma migrate status
```

Executar seed manualmente:

```
npm run prisma:seed
```

Testar conexão com o banco:

```
npx prisma studio
# Abre interface web para visualizar/editar dados
```

🎉 Conclusão

Parabéns! Se você seguiu todos os passos, seu backend DELIVEREI agora está rodando no Render e seu frontend no Netlify está conectado a ele.

Próximos Passos

1. **Configurar domínio customizado** (opcional):
   - No Render: Adicionar domínio customizado para o backend
   - No Netlify: Já está configurado (deliverei.com.br)

2. **Configurar integrações de pagamento**:
   - Adicionar credenciais Stripe
   - Adicionar credenciais Asaas

3. **Configurar WhatsApp Business API**:
   - Obter credenciais da Meta
   - Adicionar variáveis de ambiente

4. **Monitoramento**:
   - Configurar alertas no Render
   - Configurar UptimeRobot para keep-alive

5. **Testes**:
   - Testar todos os fluxos da aplicação
   - Testar multi-tenancy (várias empresas)
   - Testar checkout e pagamentos

💡 Dicas Importantes

- 🔒 **Nunca** commite arquivos `.env` para o Git
- 🔑 **Sempre** use variáveis de ambiente para credenciais
- 📊 **Monitore** os logs regularmente
- 💾 **Faça backup** do banco de dados periodicamente
- 🧪 **Teste** em ambiente local antes de fazer deploy
- 📝 **Documente** mudanças importantes

📞 Suporte

Se você encontrar problemas não cobertos neste guia:

1. Verifique os logs no Render e Netlify
2. Consulte a documentação oficial
3. Abra uma issue no GitHub
4. Entre em contato com o suporte do Render/Netlify

Última atualização: 12 de Outubro de 2025
Versão do Guia: 1.0.0
Autor: DeepAgent by Abacus.AI