🚀 Guia de Deploy - Plataforma de Revenda IPTV

📋 Índice

---

1. Configuração do Supabase

Passo 1: Criar Projeto no Supabase

1. Acesse supabase.com
2. Crie um novo projeto
3. Anote as credenciais:
   - `SUPABASE_URL`
   - `SUPABASE_ANON_KEY`
   - `SUPABASE_SERVICE_KEY`

Passo 2: Executar Scripts SQL

1. Acesse o SQL Editor no Supabase
2. Execute o script completo do arquivo `schema.sql`
3. Verifique se todas as tabelas foram criadas:
   - usuarios
   - planos
   - pedidos
   - notificacoes
   - logs_auditoria
   - configuracoes

- mensagens

Passo 3: Configurar Autenticação

1. Vá em **Authentication → Settings**
2. Configure:
   - Enable Email confirmations (opcional)
   - Configure Email templates
   - Desabilite signup público se necessário

Passo 4: Configurar Storage (Opcional)

Se precisar armazenar arquivos:

1. Crie um bucket chamado `documentos`
2. Configure políticas de acesso

---

2. Deploy do Backend

Opções de Deploy

Opção 1: Vercel (Recomendado para API simples)

```bash
# Instalar Vercel CLI
npm i -g vercel

# Fazer deploy
vercel --prod
```

Opção 2: Railway

1. Conecte seu repositório no Railway
2. Configure as variáveis de ambiente
3. Deploy automático a cada push

Opção 3: DigitalOcean App Platform

1. Conecte seu repositório
2. Configure build command: `npm install`

3. Configure run command: `npm start`

Opção 4: VPS (Ubuntu)

```bash
# Instalar Node.js
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Instalar PM2
sudo npm install -g pm2

# Clonar repositório
git clone seu-repositorio
cd seu-repositorio

# Instalar dependências
npm install

# Configurar variáveis de ambiente
nano .env

# Iniciar com PM2
pm2 start server.js --name iptv-api
pm2 save
pm2 startup
```

Configurar Nginx (VPS)

```nginx
server {
    listen 80;
    server_name api.seudominio.com;

    location / {
        proxy_pass http://localhost:3001;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Instalar SSL

```bash
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d api.seudominio.com
```

---

3. Integração com Mercado Pago

Passo 1: Criar Conta no Mercado Pago

1. Acesse mercadopago.com.br
2. Crie uma conta Business

Passo 2: Obter Credenciais

1. Acesse **Seu negócio → Configurações → Credenciais**
2. Modo Produção:
   - `Public Key`
   - `Access Token`

Passo 3: Configurar Webhooks

1. No painel do Mercado Pago: **Seu negócio → Webhooks**
2. Adicionar URL: `https://api.seudominio.com/api/webhooks/mercadopago`
3. Selecionar eventos:
   - Pagamentos
   - Chargebacks

Passo 4: Testar em Sandbox

Use as credenciais de teste primeiro:

```javascript
// No .env
MP_ACCESS_TOKEN=TEST-xxxxx
MP_PUBLIC_KEY=TEST-xxxxx
```

Cartões de teste:

- **Aprovado**: 5031 4332 1540 6351
- **Recusado**: 5031 4332 1540 6351 (com nome APRO/OTHE)

---

4. Configuração de E-mails

Opção 1: Gmail (Desenvolvimento)

```
env

EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=seu-email@gmail.com
EMAIL_PASS=senha-de-app
```

Importante: Criar senha de app no Google:

1. Acessar myaccount.google.com/security
2. Ativar verificação em 2 etapas
3. Criar senha de app

Opção 2: SendGrid (Produção)

```
env

EMAIL_HOST=smtp.sendgrid.net
EMAIL_PORT=587
EMAIL_USER=apikey
EMAIL_PASS=SG.xxxxx
```

Opção 3: AWS SES (Econômico)

```
env

EMAIL_HOST=email-smtp.us-east-1.amazonaws.com
EMAIL_PORT=587
EMAIL_USER=AKIAXXXXXXX
EMAIL_PASS=sua-senha-smtp
```

---

5. Integração com Lovable (Frontend)

Estrutura de Integração

O frontend no Lovable precisará:

1. Configurar API Client

typescript

```typescript
// lib/api.ts
const API_URL = import.meta.env.VITE_API_URL;

export const api = {
  // Auth
  register: (data) => fetch(`${API_URL}/api/auth/register`, {...}),
  login: (data) => fetch(`${API_URL}/api/auth/login`, {...}),

  // Planos
  getPlanos: () => fetch(`${API_URL}/api/planos`),

  // Pedidos
  createPedido: (plano_id, token) => fetch(`${API_URL}/api/pedidos`, {
    method: 'POST',
    headers: {
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
    },
    body: JSON.stringify({ plano_id })
  }),

  // Pagamento
  criarPreferencia: (pedido_id, token) =>
    fetch(`${API_URL}/api/pagamento/criar-preferencia`, {...})
};
```

2. Variáveis de Ambiente no Lovable

env

```
VITE_API_URL=https://api.seudominio.com
VITE_SUPABASE_URL=sua-url-supabase
VITE_SUPABASE_ANON_KEY=sua-chave-anonima
VITE_MP_PUBLIC_KEY=sua-chave-publica-mp
```

3. Integração Supabase Auth

typescript

```typescript
import { createClient } from '@supabase/supabase-js';

const supabase = createClient(
  import.meta.env.VITE_SUPABASE_URL,
  import.meta.env.VITE_SUPABASE_ANON_KEY
);

// Login
const { data, error } = await supabase.auth.signInWithPassword({
  email,
  password
});

// Pegar token
const token = data.session?.access_token;
```

4. Integração Mercado Pago (Checkout Pro)

typescript

```typescript
// Instalar SDK
npm install @mercadopago/sdk-react

// Component
import { initMercadoPago, Wallet } from '@mercadopago/sdk-react';

initMercadoPago('SUA_PUBLIC_KEY');

function CheckoutButton({ preferenceId }) {
  return (
    <Wallet
      initialization={{ preferenceId }}
      customization={{ texts:{ valueProp: 'smart_option'}}}
    />
  );
}
```

---

6. Segurança e Boas Práticas

Variáveis de Ambiente

✅ **NUNCA** commitar `.env`

✅ Usar variáveis diferentes para dev/prod
✅ Rotacionar tokens regularmente

HTTPS Obrigatório

- Backend deve usar HTTPS em produção
- Configure SSL/TLS corretamente

Rate Limiting

Adicionar ao server.js:

```javascript
const rateLimit = require('express-rate-limit');

const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutos
  max: 100 // máximo de requisições
});

app.use('/api/', limiter);
```

Validação de Dados

```bash
npm install joi
```

```javascript
const Joi = require('joi');

const pedidoSchema = Joi.object({
  plano_id: Joi.string().uuid().required()
});

// Middleware de validação
const validate = (schema) => (req, res, next) => {
  const { error } = schema.validate(req.body);
  if (error) {
    return res.status(400).json({ error: error.details[0].message });
  }
  next();
};
```

CORS

Configurar domínios permitidos:

```javascript
app.use(cors({
  origin: [
    'https://seu-frontend.lovable.app',
    'https://seudominio.com'
  ],
  credentials: true
}));
```

Logs

```bash
npm install winston
```

```javascript
const winston = require('winston');

const logger = winston.createLogger({
  level: 'info',
  format: winston.format.json(),
  transports: [
    new winston.transports.File({ filename: 'error.log', level: 'error' }),
    new winston.transports.File({ filename: 'combined.log' })
  ]
});
```

---

7. Testes

Checklist de Testes

Autenticação

- [ ] Registro de novo usuário
- [ ] Login com credenciais válidas
- [ ] Login com credenciais inválidas
- [ ] Recuperação de senha

Planos

- [ ] Listar planos ativos
- [ ] Criar novo plano (admin)
- [ ] Editar plano (admin)
- [ ] Desativar plano (admin)

Fluxo de Compra

- [ ] Criar pedido
- [ ] Gerar preferência Mercado Pago
- [ ] Simular pagamento aprovado
- [ ] Webhook recebido corretamente
- [ ] Notificação enviada ao admin
- [ ] Admin envia credenciais
- [ ] Cliente recebe credenciais

E-mails

- [ ] E-mail de boas-vindas
- [ ] E-mail de confirmação de pagamento
- [ ] E-mail com credenciais
- [ ] E-mail de aviso de expiração

Segurança

- [ ] Acesso negado sem token
- [ ] Cliente não pode acessar área admin
- [ ] RLS funcionando corretamente

📞 Suporte

Documentações Oficiais

- Supabase Docs
- Mercado Pago Docs
- Express.js Guide
- Node.js Best Practices

Monitoramento

Considere usar:

- **Sentry** para tracking de erros
- **LogRocket** para sessões de usuário
- **Google Analytics** para métricas

🎯 Próximos Passos

1. ✅ Configurar Supabase
2. ✅ Deploy do backend
3. ✅ Configurar Mercado Pago
4. ✅ Testar fluxo completo
5. ✅ Configurar e-mails
6. ✅ Conectar frontend Lovable
7. ✅ Testes em produção
8. ✅ Go Live! 🚀

⚠️ Checklist Pré-Produção

- [ ] Todas as variáveis de ambiente configuradas
- [ ] SSL instalado e funcionando
- [ ] Backup automático do banco configurado
- [ ] Logs sendo salvos corretamente
- [ ] Monitoramento de erros ativo
- [ ] Termos de uso e política de privacidade publicados
- [ ] Testes de carga realizados
- [ ] Plano de backup e recuperação documentado
- [ ] Contato com fornecedor IPTV estabelecido
- [ ] Processo de envio de credenciais definido