⚙ Guia de Configuração - Voz do Cliente

Variáveis de Ambiente

Arquivo .env

O arquivo `.env` contém todas as configurações necessárias para a aplicação. Copie o arquivo `.env.example` e configure as variáveis conforme necessário.

```
cp .env.example .env
```

Configurações Obrigatórias

1. Banco de Dados

```
# PostgreSQL Connection String
DATABASE_URL="postgresql://usuario:senha@host:porta/nome_banco"

# Exemplo para desenvolvimento local
DATABASE_URL="postgresql://voc_user:senha123@localhost:5432/voz_cliente"

# Exemplo para produção
DATABASE_URL="postgresql://voc_user:senha_segura@db.empresa.com:5432/voz_cliente_prod"
```

2. Next.js Authentication

```
# Chave secreta para JWT (gerar uma chave aleatória segura)
NEXTAUTH_SECRET="sua_chave_secreta_muito_longa_e_aleatoria_aqui"

# URL da aplicação
NEXTAUTH_URL="http://localhost:3000"  # Desenvolvimento
NEXTAUTH_URL="https://voz-cliente.empresa.com"  # Produção
```

Configurações Opcionais (APIs Externas)

3. Twitter API v2

```
# Bearer Token do Twitter API v2
TWITTER_BEARER_TOKEN="seu_bearer_token_aqui"
```

4. Reddit API

```
# Credenciais da aplicação Reddit
REDDIT_CLIENT_ID="seu_client_id_aqui"
REDDIT_CLIENT_SECRET="seu_client_secret_aqui"
REDDIT_USER_AGENT="VozDoCliente/1.0"
```

Configuração do Banco de Dados PostgreSQL

1. Instalação e Configuração Inicial

```
# Ubuntu/Debian
sudo apt update
sudo apt install postgresql postgresql-contrib

# Iniciar serviço
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

2. Criar Banco e Usuário

```
-- Conectar como superusuário
sudo -u postgres psql

-- Criar banco de dados
CREATE DATABASE voz_cliente;

-- Criar usuário
CREATE USER voc_user WITH ENCRYPTED PASSWORD 'senha_muito_segura_aqui';

-- Conceder privilégios
GRANT ALL PRIVILEGES ON DATABASE voz_cliente TO voc_user;
ALTER USER voc_user CREATEDB;

-- Sair
\q
```

3. Configurar Acesso Remoto (Produção)

Editar postgresql.conf:

```
sudo nano /etc/postgresql/15/main/postgresql.conf
```

```
# Permitir conexões de qualquer IP (cuidado em produção)
listen_addresses = '*'

# Ou especificar IPs específicos
listen_addresses = 'localhost,192.168.1.100'
```

Editar pg_hba.conf:

```
sudo nano /etc/postgresql/15/main/pg_hba.conf
```

```
# Adicionar linha para permitir conexões
host    voz_cliente    voc_user    192.168.1.0/24    md5
```

```
# Reiniciar PostgreSQL
sudo systemctl restart postgresql
```

4. Executar Migrações

```
# Navegar para o diretório da aplicação
cd app

# Gerar cliente Prisma
npx prisma generate

# Aplicar schema ao banco
npx prisma db push

# Verificar se as tabelas foram criadas
npx prisma studio  # Interface web para visualizar dados
```

Configuração de Produção

1. Variáveis de Ambiente para Produção

```
# Banco de dados de produção
DATABASE_URL="postgresql://voc_user:senha_super_segura@db-prod.empresa.com:5432/
voz_cliente_prod"

# URLs de produção
NEXTAUTH_URL="https://voz-cliente.empresa.com"
NEXTAUTH_SECRET="chave_secreta_muito_longa_e_complexa_para_producao"

# Node.js environment
NODE_ENV="production"

# APIs externas (se disponíveis)
TWITTER_BEARER_TOKEN="bearer_token_producao"
REDDIT_CLIENT_ID="client_id_producao"
REDDIT_CLIENT_SECRET="client_secret_producao"
```

2. Configurações de Segurança

Firewall (UFW)

```
# Permitir apenas portas necessárias
sudo ufw allow 22    # SSH
sudo ufw allow 80    # HTTP
sudo ufw allow 443   # HTTPS
sudo ufw allow 5432  # PostgreSQL (apenas se necessário)
sudo ufw enable
```

SSL/TLS

```
# Instalar Certbot
sudo apt install certbot python3-certbot-nginx

# Obter certificado
sudo certbot --nginx -d voz-cliente.empresa.com
```

3. Configurações de Performance

Otimizações do PostgreSQL

```
-- Conectar ao banco como superusuário
sudo -u postgres psql

-- Configurações de performance (ajustar conforme hardware)
ALTER SYSTEM SET shared_buffers = '256MB';
ALTER SYSTEM SET effective_cache_size = '1GB';
ALTER SYSTEM SET maintenance_work_mem = '64MB';
ALTER SYSTEM SET checkpoint_completion_target = 0.9;
ALTER SYSTEM SET wal_buffers = '16MB';
ALTER SYSTEM SET default_statistics_target = 100;

-- Recarregar configurações
SELECT pg_reload_conf();
```

Configurações do Node.js

```
# Variáveis de ambiente para otimização
export NODE_OPTIONS="--max-old-space-size=2048"
export UV_THREADPOOL_SIZE=4
```

Configuração de Monitoramento

1. Logs da Aplicação

```
# Criar diretório de logs
sudo mkdir -p /var/log/voz-cliente
sudo chown $USER:$USER /var/log/voz-cliente
```

2. Configurar Logrotate

```
sudo nano /etc/logrotate.d/voz-cliente
```

```
/var/log/voz-cliente/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 voc-user voc-user
    postrotate
        systemctl reload voz-cliente
    endscript
}
```

Backup e Recuperação

1. Configurar Backup Automático

```
# Criar script de backup
sudo nano /usr/local/bin/backup-voz-cliente.sh
```

```
#!/bin/bash
BACKUP_DIR="/backup/voz-cliente"
DATE=$(date +%Y%m%d_%H%M%S)
DB_NAME="voz_cliente"
DB_USER="voc_user"

# Criar diretório se não existir
mkdir -p $BACKUP_DIR

# Backup do banco de dados
pg_dump -h localhost -U $DB_USER -d $DB_NAME > $BACKUP_DIR/db_backup_$DATE.sql

# Backup dos arquivos de configuração
tar -czf $BACKUP_DIR/config_backup_$DATE.tar.gz /path/to/app/.env

# Remover backups antigos (manter últimos 7 dias)
find $BACKUP_DIR -name "*.sql" -mtime +7 -delete
find $BACKUP_DIR -name "*.tar.gz" -mtime +7 -delete

echo "Backup concluído: $DATE"
```

```
# Tornar executável
sudo chmod +x /usr/local/bin/backup-voz-cliente.sh

# Configurar cron para backup diário
sudo crontab -e
```

```
# Backup diário às 2:00 AM
0 2 * * * /usr/local/bin/backup-voz-cliente.sh >> /var/log/backup-voz-cliente.log 2>&1
```

Configuração de Desenvolvimento

1. Ambiente de Desenvolvimento

```
# .env.development
DATABASE_URL="postgresql://voc_user:senha123@localhost:5432/voz_cliente_dev"
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="chave_desenvolvimento"
NODE_ENV="development"

# APIs podem usar dados simulados em desenvolvimento
# TWITTER_BEARER_TOKEN=""  # Deixar vazio para usar dados simulados
# REDDIT_CLIENT_ID=""      # Deixar vazio para usar dados simulados
```

2. Hot Reload e Debug

```
# Executar em modo de desenvolvimento com debug
DEBUG=* npm run dev

# Ou apenas logs específicos
DEBUG=prisma:* npm run dev
```

Validação da Configuração

1. Verificar Conexão com Banco

```
# Testar conexão
npx prisma db pull

# Verificar schema
npx prisma studio
```

2. Verificar APIs Externas

```
# Testar configuração fazendo uma análise de teste
curl -X POST http://localhost:3000/api/analyze \
  -H "Content-Type: application/json" \
  -d '{"companyName": "Teste"}'
```

3. Verificar Logs

```
# Verificar logs da aplicação
tail -f /var/log/voz-cliente/app.log

# Verificar logs do PostgreSQL
sudo tail -f /var/log/postgresql/postgresql-15-main.log
```

Próximos Passos

Após configurar a aplicação:

1. Consulte o Guia de Deploy (./DEPLOY.md) para colocar em produção
2. Veja APIs Externas (./EXTERNAL_APIS.md) para configurar Twitter e Reddit
3. Leia Troubleshooting (./TROUBLESHOOTING.md) para resolver problemas comuns

Suporte

Para dúvidas sobre configuração:

1. Verifique os logs da aplicação
2. Consulte a documentação de troubleshooting
3. Entre em contato com a equipe de desenvolvimento