Guia de Instalação - ClientManager

📋 Índice

💻 Requisitos do Sistema

Requisitos Mínimos

- **Node.js**: 18.0.0 ou superior
- **npm**: 9.0.0 ou superior
- **PostgreSQL**: 12.0 ou superior
- **Memória RAM**: 2GB mínimo, 4GB recomendado
- **Espaço em Disco**: 1GB livre
- **Sistema Operacional**: Linux, macOS ou Windows

Requisitos Recomendados

- **Node.js**: 20.0.0 LTS
- **PostgreSQL**: 15.0 ou superior

- **Memória RAM**: 8GB
- **CPU**: 2 cores ou mais
- **SSD**: Para melhor performance

🏠 Instalação Local

Passo 1: Preparação do Ambiente

Ubuntu/Debian

```
# Atualizar sistema
sudo apt update && sudo apt upgrade -y

# Instalar Node.js 20 LTS
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -y nodejs

# Instalar PostgreSQL
sudo apt install postgresql postgresql-contrib -y

# Instalar Git
sudo apt install git -y
```

macOS

```
# Instalar Homebrew (se não tiver)
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Instalar Node.js
brew install node

# Instalar PostgreSQL
brew install postgresql
brew services start postgresql

# Instalar Git
brew install git
```

Windows

1. Baixe e instale Node.js do site oficial
2. Baixe e instale PostgreSQL do site oficial
3. Baixe e instale Git do site oficial

Passo 2: Clone do Repositório

```
# Clone o repositório
git clone https://github.com/seu-usuario/client-manager.git

# Entre no diretório
cd client-manager

# Verifique a versão do Node.js
node --version  # Deve ser 18+

# Verifique a versão do npm
npm --version   # Deve ser 9+
```

Passo 3: Instalação das Dependências

```
# Instale as dependências
npm install

# Verifique se não há vulnerabilidades
npm audit

# Se houver vulnerabilidades, corrija
npm audit fix
```

🗄️ Configuração do Banco de Dados

PostgreSQL Local

Ubuntu/Debian

```
# Iniciar serviço PostgreSQL
sudo service postgresql start

# Acessar PostgreSQL como usuário postgres
sudo -u postgres psql

# Dentro do PostgreSQL, execute:
CREATE DATABASE client_manager;
CREATE USER clientmanager WITH PASSWORD 'sua_senha_segura';
GRANT ALL PRIVILEGES ON DATABASE client_manager TO clientmanager;
\q
```

macOS

```
 # Iniciar PostgreSQL
brew services start postgresql

# Criar banco de dados
createdb client_manager

# Acessar PostgreSQL
psql client_manager

# Criar usuário (opcional)
CREATE USER clientmanager WITH PASSWORD 'sua_senha_segura';
GRANT ALL PRIVILEGES ON DATABASE client_manager TO clientmanager;
\q
```

Windows

1. Abra o pgAdmin ou SQL Shell
2. Conecte-se ao servidor PostgreSQL
3. Execute os comandos SQL acima

PostgreSQL em Nuvem

Supabase

1. Acesse supabase.com
2. Crie uma nova conta/projeto
3. Vá em Settings > Database
4. Copie a connection string

Neon

1. Acesse neon.tech
2. Crie uma nova conta/projeto
3. Copie a connection string

Railway

1. Acesse railway.app
2. Crie um novo projeto

3. Adicione PostgreSQL
4. Copie a connection string

⚙️ Configuração das Variáveis de Ambiente

Passo 1: Criar Arquivo .env

```
# Copie o arquivo de exemplo
cp .env.example .env

# Edite o arquivo .env
nano .env  # ou use seu editor preferido
```

Passo 2: Configurar Variáveis

```
# Database
DATABASE_URL="postgresql://usuario:senha@localhost:5432/client_manager"

# JWT
JWT_SECRET="seu-jwt-secret-super-seguro-com-pelo-menos-32-caracteres"

# Next.js
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="seu-nextauth-secret-super-seguro"

# Meta Ads API (opcional para desenvolvimento)
META_ADS_APP_ID="seu-app-id"
META_ADS_APP_SECRET="seu-app-secret"
META_ADS_ACCESS_TOKEN="seu-access-token"

# Google Ads API (opcional para desenvolvimento)
GOOGLE_ADS_CLIENT_ID="seu-client-id"
GOOGLE_ADS_CLIENT_SECRET="seu-client-secret"
GOOGLE_ADS_REFRESH_TOKEN="seu-refresh-token"
GOOGLE_ADS_DEVELOPER_TOKEN="seu-developer-token"

# Evolution API (opcional para desenvolvimento)
EVOLUTION_API_URL="https://sua-evolution-api.com"
EVOLUTION_API_KEY="sua-api-key"

# n8n (opcional para desenvolvimento)
N8N_WEBHOOK_URL="https://seu-n8n.com/webhook"
N8N_API_KEY="sua-api-key"
```

Passo 3: Gerar Secrets Seguros

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

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

🚀 Primeira Execução

Passo 1: Configurar Banco de Dados

```
# Gerar cliente Prisma
npx prisma generate

# Executar migrações
npx prisma migrate dev --name init

# Verificar se as tabelas foram criadas
npx prisma studio  # Abre interface web
```

Passo 2: Popular Banco com Dados Iniciais

```
# Executar seed
npx tsx prisma/seed.ts

# Ou se der erro, instale tsx primeiro
npm install -g tsx
tsx prisma/seed.ts
```

Passo 3: Iniciar Aplicação

```
# Modo desenvolvimento
npm run dev

# A aplicação estará disponível em:
# http://localhost:3000
```

Passo 4: Testar Login

Use um dos usuários criados pelo seed:

Email	Senha	Role
admin@clientmanager.com	admin123	Administrador
gestor@clientmanager.com	gestor123	Gestor
maria@empresaabc.com	cliente123	Cliente

🔗 Configuração das Integrações

Meta Ads API

Passo 1: Criar App no Facebook Developers

1. Acesse developers.facebook.com
2. Clique em "Meus Apps" > "Criar App"
3. Escolha "Empresa" como tipo
4. Preencha os dados do app

Passo 2: Configurar Permissões

1. Vá em "Produtos" > "Marketing API"
2. Configure as permissões necessárias:
3. `ads_read`
4. `ads_management`
5. `business_management`

Passo 3: Obter Credenciais

1. **App ID**: Disponível no dashboard
2. **App Secret**: Em Configurações > Básico
3. **Access Token**: Use a Graph API Explorer

Passo 4: Testar Conexão

```
# Teste a API
curl -G \
  -d "access_token=SEU_ACCESS_TOKEN" \
  "https://graph.facebook.com/v18.0/me/adaccounts"
```

Google Ads API

Passo 1: Configurar Projeto no Google Cloud

1. Acesse console.cloud.google.com
2. Crie um novo projeto
3. Ative a Google Ads API

Passo 2: Configurar OAuth 2.0

1. Vá em "APIs e serviços" > "Credenciais"
2. Crie credenciais OAuth 2.0
3. Configure URLs de redirecionamento

Passo 3: Obter Developer Token

1. Acesse Google Ads
2. Vá em Ferramentas > Configuração > Centro de API
3. Solicite Developer Token

Passo 4: Testar Conexão

```
# Use a biblioteca oficial do Google Ads
npm install google-ads-api
```

Evolution API (WhatsApp)

Passo 1: Instalar Evolution API

```
# Clone o repositório
git clone https://github.com/EvolutionAPI/evolution-api.git

# Configure e execute
cd evolution-api
npm install
npm start
```

Passo 2: Configurar Instância

1. Acesse a interface web da Evolution API
2. Crie uma nova instância
3. Conecte com QR Code
4. Obtenha a API Key

Passo 3: Testar Envio

```
# Teste envio de mensagem
curl -X POST \
  -H "Content-Type: application/json" \
  -H "apikey: SUA_API_KEY" \
  -d '{"number": "5511999999999", "text": "Teste"}' \
  "https://sua-evolution-api.com/message/sendText/instancia"
```

n8n (Automação)

Passo 1: Instalar n8n

```
# Instalar globalmente
npm install n8n -g

# Ou usar Docker
docker run -it --rm \
  --name n8n \
  -p 5678:5678 \
  n8nio/n8n
```

Passo 2: Configurar Workflows

1. Acesse http://localhost:5678

2. Crie workflows de automação
3. Configure webhooks para o ClientManager

🌐 Deploy em Produção

Vercel (Recomendado)

Passo 1: Preparar Repositório

```
# Commit todas as mudanças
git add .
git commit -m "Configuração inicial"
git push origin main
```

Passo 2: Deploy na Vercel

1. Acesse vercel.com
2. Conecte seu repositório GitHub
3. Configure variáveis de ambiente
4. Deploy automático

Passo 3: Configurar Banco de Dados

Use Supabase ou Neon para PostgreSQL em produção:

```
DATABASE_URL="postgresql://user:pass@host:5432/db?sslmode=require"
```

VPS/Servidor Próprio

Passo 1: Preparar Servidor

```
# Ubuntu 22.04 LTS
sudo apt update && sudo apt upgrade -y

# Instalar Node.js 20
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -y nodejs

# Instalar PostgreSQL
sudo apt install postgresql postgresql-contrib -y

# Instalar Nginx
sudo apt install nginx -y

# Instalar PM2
sudo npm install -g pm2
```

Passo 2: Configurar Aplicação

```
# Clone no servidor
git clone https://github.com/seu-usuario/client-manager.git
cd client-manager

# Instalar dependências
npm ci --only=production

# Build da aplicação
npm run build

# Configurar PM2
pm2 start npm --name "client-manager" -- start
pm2 startup
pm2 save
```

Passo 3: Configurar Nginx

```
# /etc/nginx/sites-available/client-manager
server {
    listen 80;
    server_name seu-dominio.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }
}
```

```
# Ativar site
sudo ln -s /etc/nginx/sites-available/client-manager /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx

# Configurar SSL com Certbot
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d seu-dominio.com
```

🐳 Docker

Dockerfile

```
FROM node:20-alpine

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY . .
RUN npm run build

EXPOSE 3000

CMD ["npm", "start"]
```

docker-compose.yml

```yaml
version: '3.8'

services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgresql://postgres:password@db:5432/client_manager
      - JWT_SECRET=seu-jwt-secret
    depends_on:
      - db

  db:
    image: postgres:15
    environment:
      - POSTGRES_DB=client_manager
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=password
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

volumes:
  postgres_data:
```

Executar com Docker

```bash
# Build e start
docker-compose up -d

# Ver logs
docker-compose logs -f

# Parar
docker-compose down
```

🔧 Troubleshooting

Problemas Comuns

Erro de Conexão com Banco

```
# Verificar se PostgreSQL está rodando
sudo service postgresql status

# Verificar conexão
psql -h localhost -U postgres -d client_manager

# Verificar variável DATABASE_URL
echo $DATABASE_URL
```

Erro de Permissões

```
# Dar permissões corretas
sudo chown -R $`USER:`$USER /caminho/para/client-manager
chmod -R 755 /caminho/para/client-manager
```

Erro de Porta em Uso

```
# Verificar qual processo está usando a porta 3000
sudo lsof -i :3000

# Matar processo se necessário
sudo kill -9 PID
```

Erro de Memória

```
# Aumentar limite de memória do Node.js
export NODE_OPTIONS="--max-old-space-size=4096"
npm run build
```

Logs e Debugging

```
# Ver logs da aplicação
pm2 logs client-manager

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

# Ver logs do Nginx
sudo tail -f /var/log/nginx/error.log
```

Performance

```
 # Otimizar PostgreSQL
sudo nano /etc/postgresql/15/main/postgresql.conf

# Configurações recomendadas:
# shared_buffers = 256MB
# effective_cache_size = 1GB
# work_mem = 4MB
# maintenance_work_mem = 64MB

# Reiniciar PostgreSQL
sudo service postgresql restart
```

📞 Suporte

Se encontrar problemas durante a instalação:

- **GitHub Issues**: Reportar problema
- **Discord**: Comunidade
- **Email**: suporte@clientmanager.com

Última atualização: Janeiro 2024
Versão: 1.0.0