Guia de Instalação e Configuração - Sistema de Estoque para Peças de Carro

Versão: 1.0
Data: Junho 2025
Autor: Manus AI

Sumário

Requisitos do Sistema

Requisitos de Hardware

Para uma instalação adequada do sistema, é necessário um servidor ou computador que atenda aos seguintes requisitos mínimos de hardware. Para ambientes de desenvolvimento ou empresas pequenas com até 1000 produtos e baixo volume de transações, um servidor com 2 CPU cores, 4GB de RAM e 20GB de espaço em disco é suficiente.

Para ambientes de produção com maior volume de dados e transações, recomenda-se um servidor com pelo menos 4 CPU cores, 8GB de RAM e 50GB de espaço em disco SSD para garantir performance adequada. O espaço em disco deve considerar não apenas o sistema e banco de dados, mas também armazenamento de imagens de produtos e logs do sistema.

A conectividade de rede é crucial para o funcionamento adequado da integração com o Mercado Livre. É necessária uma conexão de internet estável com velocidade mínima de

10 Mbps para garantir sincronização em tempo real. Para ambientes com alto volume de transações, recomenda-se conexão de 50 Mbps ou superior.

Requisitos de Software

O sistema foi desenvolvido utilizando tecnologias modernas e requer um ambiente de execução específico. Para o backend, é necessário Python 3.11 ou superior, que pode ser instalado através do gerenciador de pacotes do sistema operacional ou baixado diretamente do site oficial do Python.

O frontend requer Node.js versão 18 ou superior para compilação e execução. O Node.js inclui o npm (Node Package Manager) que é utilizado para gerenciar as dependências do projeto. Recomenda-se também a instalação do pnpm como gerenciador de pacotes alternativo para melhor performance.

Para o banco de dados, o sistema suporta SQLite para desenvolvimento e PostgreSQL para produção. SQLite está incluído no Python e não requer instalação adicional, enquanto PostgreSQL deve ser instalado separadamente. Para ambientes de produção, recomenda-se PostgreSQL 12 ou superior.

Sistema Operacional

O sistema é compatível com os principais sistemas operacionais, incluindo Linux (Ubuntu 20.04+, CentOS 8+, Debian 10+), macOS 10.15+, e Windows 10+. Para ambientes de produção, recomenda-se fortemente o uso de distribuições Linux devido à maior estabilidade, segurança e performance.

Ubuntu Server LTS é a distribuição recomendada para produção devido ao suporte de longo prazo, ampla documentação disponível, e compatibilidade testada com todas as dependências do sistema. A instalação em Ubuntu é simplificada através dos gerenciadores de pacotes apt e snap.

Para desenvolvimento local, qualquer sistema operacional moderno é adequado, permitindo que desenvolvedores utilizem seus ambientes preferidos. O sistema foi testado extensivamente em Windows 10/11, macOS Big Sur/Monterey, e várias distribuições Linux.

Instalação do Backend

Preparação do Ambiente

Antes de iniciar a instalação do backend, é necessário preparar o ambiente de desenvolvimento ou produção. Comece atualizando o sistema operacional e instalando as dependências básicas. Em sistemas Ubuntu/Debian, execute os seguintes comandos:

```
sudo apt update && sudo apt upgrade -y
sudo apt install python3.11 python3.11-venv python3.11-dev
python3-pip git curl wget -y
```

Para sistemas CentOS/RHEL, utilize:

```
sudo yum update -y
sudo yum install python3.11 python3.11-venv python3.11-devel
python3-pip git curl wget -y
```

Crie um usuário específico para executar a aplicação, seguindo as melhores práticas de segurança:

```
sudo useradd -m -s /bin/bash estoque
sudo usermod -aG sudo estoque
su - estoque
```

Download e Configuração

Clone ou copie os arquivos do sistema para o diretório apropriado. Se você recebeu os arquivos em um pacote compactado, extraia-os para o diretório home do usuário:

```
cd /home/estoque
# Se recebeu via git:
git clone <repository-url> sistema_estoque
# Ou se recebeu arquivos compactados:
tar -xzf sistema_estoque.tar.gz
cd sistema_estoque/sistema_estoque_backend
```

Crie e ative um ambiente virtual Python para isolar as dependências do projeto:

```
python3.11 -m venv venv
source venv/bin/activate
```

Instale as dependências do projeto utilizando o arquivo requirements.txt:

```
pip install --upgrade pip
pip install -r requirements.txt
```

Configuração de Variáveis de Ambiente

Copie o arquivo de exemplo de configuração e edite-o com suas configurações específicas:

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

Configure as seguintes variáveis essenciais no arquivo .env:

```
# Configurações do Mercado Livre
ML_CLIENT_ID=seu_client_id_aqui
ML_CLIENT_SECRET=seu_client_secret_aqui
ML_REDIRECT_URI=http://seu-dominio.com/api/ml/callback

# Configurações do banco de dados (para produção)
DATABASE_URL=postgresql://usuario:senha@localhost/
sistema_estoque

# Configurações gerais
FLASK_ENV=production
SECRET_KEY=sua_chave_secreta_muito_segura_aqui
```

Para gerar uma chave secreta segura, utilize:

```
python3 -c "import secrets; print(secrets.token_hex(32))"
```

Inicialização do Banco de Dados

Para desenvolvimento, o sistema utilizará SQLite automaticamente. Para produção com PostgreSQL, primeiro crie o banco de dados:

```
sudo -u postgres psql
CREATE DATABASE sistema_estoque;
CREATE USER estoque_user WITH PASSWORD 'senha_segura';
GRANT ALL PRIVILEGES ON DATABASE sistema_estoque TO
```

```
estoque_user;
\q
```

Execute a aplicação uma vez para criar as tabelas automaticamente:

```
python src/main.py
```

O sistema criará automaticamente todas as tabelas necessárias no primeiro acesso.

Teste da Instalação

Para verificar se a instalação foi bem-sucedida, execute o servidor de desenvolvimento:

```
python src/main.py
```

O sistema deve iniciar e exibir mensagens indicando que está rodando na porta 5001. Teste o acesso através de outro terminal:

```
curl http://localhost:5001/api/produtos
```

Se a resposta for um JSON com lista vazia de produtos, a instalação do backend foi bem-sucedida.

Instalação do Frontend

Preparação do Ambiente Node.js

O frontend requer Node.js e um gerenciador de pacotes para compilação e execução. Instale Node.js através do gerenciador de versões nvm para maior flexibilidade:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.0/
install.sh | bash
source ~/.bashrc
nvm install 18
nvm use 18
```

Alternativamente, instale diretamente através do gerenciador de pacotes do sistema:

```
# Ubuntu/Debian
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash
```

```
-
sudo apt-get install -y nodejs

# CentOS/RHEL
curl -fsSL https://rpm.nodesource.com/setup_18.x | sudo bash -
sudo yum install -y nodejs
```

Instale o pnpm como gerenciador de pacotes para melhor performance:

```
npm install -g pnpm
```

Configuração do Projeto Frontend

Navegue até o diretório do frontend e instale as dependências:

```
cd /home/estoque/sistema_estoque/sistema_estoque_frontend
pnpm install
```

Configure as variáveis de ambiente do frontend criando um arquivo .env.local:

```
nano .env.local
```

Adicione a configuração da URL da API:

```
VITE_API_BASE_URL=http://localhost:5001/api
```

Para produção, substitua localhost pelo domínio ou IP do servidor backend.

Compilação para Produção

Para ambientes de produção, compile o frontend para arquivos estáticos otimizados:

```
pnpm run build
```

Os arquivos compilados serão gerados no diretório `dist/` e podem ser servidos por qualquer servidor web como Nginx ou Apache.

Teste do Frontend

Para desenvolvimento, execute o servidor de desenvolvimento:

```
pnpm run dev
```

O frontend estará disponível em http://localhost:5173. Para produção, copie os arquivos do diretório `dist/` para o diretório de arquivos estáticos do servidor web.

Configuração do Banco de Dados

PostgreSQL para Produção

Para ambientes de produção, é altamente recomendado utilizar PostgreSQL devido à sua robustez, performance e recursos avançados. Instale PostgreSQL através do gerenciador de pacotes:

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

# CentOS/RHEL
sudo yum install postgresql-server postgresql-contrib -y
sudo postgresql-setup initdb
```

Configure PostgreSQL para aceitar conexões locais editando o arquivo de configuração:

```
sudo nano /etc/postgresql/14/main/pg_hba.conf
```

Adicione ou modifique a linha para permitir autenticação por senha:

```
local   all             all
md5
```

Reinicie o serviço PostgreSQL:

```
sudo systemctl restart postgresql
sudo systemctl enable postgresql
```

Configuração de Usuário e Banco

Crie um usuário específico para a aplicação com permissões limitadas:

```
sudo -u postgres psql
CREATE USER estoque_user WITH PASSWORD 'senha_muito_segura';
CREATE DATABASE sistema_estoque OWNER estoque_user;
GRANT ALL PRIVILEGES ON DATABASE sistema_estoque TO
estoque_user;
\q
```

Teste a conexão com o novo usuário:

```
psql -h localhost -U estoque_user -d sistema_estoque
```

Otimização de Performance

Para otimizar a performance do PostgreSQL, edite o arquivo de configuração principal:

```
sudo nano /etc/postgresql/14/main/postgresql.conf
```

Ajuste os seguintes parâmetros baseados nos recursos disponíveis:

```
shared_buffers = 256MB          # 25% da RAM disponível
effective_cache_size = 1GB      # 75% da RAM disponível
work_mem = 4MB                  # Para operações de ordenação
maintenance_work_mem = 64MB     # Para operações de manutenção
```

Reinicie PostgreSQL após as alterações:

```
sudo systemctl restart postgresql
```

Backup e Recuperação

Configure backups automáticos do banco de dados criando um script de backup:

```
nano /home/estoque/backup_db.sh
```

Adicione o seguinte conteúdo:

```
#!/bin/bash
BACKUP_DIR="/home/estoque/backups"
DATE=$(date +%Y%m%d_%H%M%S)
mkdir -p $BACKUP_DIR
```

```
pg_dump -h localhost -U estoque_user -d sistema_estoque >
$BACKUP_DIR/backup_$DATE.sql

# Manter apenas os últimos 7 backups
find $BACKUP_DIR -name "backup_*.sql" -mtime +7 -delete
```

Torne o script executável e configure no crontab:

```
chmod +x /home/estoque/backup_db.sh
crontab -e
```

Adicione uma linha para executar backup diário às 2h da manhã:

```
0 2 * * * /home/estoque/backup_db.sh
```

Configuração da Integração com Mercado Livre

Criação da Aplicação no Mercado Livre

Para configurar a integração com o Mercado Livre, primeiro é necessário criar uma aplicação no portal de desenvolvedores. Acesse https://developers.mercadolivre.com.br e faça login com sua conta do Mercado Livre.

No painel de desenvolvedores, clique em "Criar aplicação" e preencha as informações solicitadas:

- **Nome da aplicação**: Sistema de Estoque [Nome da sua empresa]
- **Descrição**: Sistema interno de controle de estoque com integração automática
- **URL de callback**: http://seu-dominio.com/api/ml/callback
- **Domínios autorizados**: seu-dominio.com

Após criar a aplicação, anote o Client ID e Client Secret gerados, pois serão necessários para configuração do sistema.

Configuração das Credenciais

Edite o arquivo .env do backend e adicione as credenciais obtidas:

```
nano /home/estoque/sistema_estoque/sistema_estoque_backend/.env
```

Configure as seguintes variáveis:

```
ML_CLIENT_ID=seu_client_id_aqui
ML_CLIENT_SECRET=seu_client_secret_aqui
ML_REDIRECT_URI=http://seu-dominio.com/api/ml/callback
```

Para desenvolvimento local, utilize:

```
ML_REDIRECT_URI=http://localhost:5001/api/ml/callback
```

Processo de Autorização

Após configurar as credenciais, é necessário autorizar o sistema a acessar sua conta do Mercado Livre. Acesse o sistema através do navegador e navegue até a seção de configurações da integração com Mercado Livre.

Clique em "Conectar com Mercado Livre" para iniciar o processo de autorização OAuth. Você será redirecionado para o site do Mercado Livre onde deve fazer login e autorizar o acesso do sistema às suas informações de vendas e produtos.

Após autorizar, você será redirecionado de volta para o sistema, que automaticamente receberá e armazenará os tokens de acesso necessários. Estes tokens são renovados automaticamente pelo sistema quando necessário.

Configuração de Webhooks

Para receber notificações automáticas de novos pedidos, configure webhooks no Mercado Livre. Acesse o painel de desenvolvedores e navegue até a seção de notificações da sua aplicação.

Configure um webhook com as seguintes informações:

- **URL de notificação**: http://seu-dominio.com/api/ml/webhook
- **Tópicos**: orders_v2, payments
- **Status**: Ativo

O sistema está preparado para receber e processar automaticamente estas notificações, atualizando pedidos e estoque em tempo real.

Teste da Integração

Para verificar se a integração está funcionando corretamente, utilize o endpoint de teste disponível no sistema:

```
curl http://localhost:5001/api/ml/status
```

A resposta deve mostrar informações sobre o status da conexão, tokens válidos, e número de produtos configurados para sincronização.

Teste também a sincronização de estoque criando um produto no sistema com ML Item ID correspondente a um anúncio existente no Mercado Livre e verificando se as alterações de estoque são refletidas na plataforma.

Deploy em Produção

Configuração do Servidor Web

Para produção, recomenda-se utilizar um servidor web como Nginx como proxy reverso para o backend Flask. Instale e configure Nginx:

```
sudo apt install nginx -y
sudo systemctl enable nginx
```

Crie um arquivo de configuração para o site:

```
sudo nano /etc/nginx/sites-available/sistema_estoque
```

Adicione a seguinte configuração:

```
server {
    listen 80;
    server_name seu-dominio.com;

    # Frontend estático
    location / {
        root /home/estoque/sistema_estoque/
sistema_estoque_frontend/dist;
        try_files $uri $uri/ /index.html;
    }

    # API Backend
    location /api/ {
        proxy_pass http://127.0.0.1:5001;
        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;
        }
    }
```

Ative o site e reinicie Nginx:

```
sudo ln -s /etc/nginx/sites-available/sistema_estoque /etc/
nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

Configuração de SSL/HTTPS

Para segurança em produção, configure SSL utilizando Let's Encrypt:

```
sudo apt install certbot python3-certbot-nginx -y
sudo certbot --nginx -d seu-dominio.com
```

O Certbot configurará automaticamente SSL e renovação automática dos certificados.

Configuração de Serviço Systemd

Crie um serviço systemd para executar o backend automaticamente:

```
sudo nano /etc/systemd/system/sistema-estoque.service
```

Adicione a seguinte configuração:

```
[Unit]
Description=Sistema de Estoque Backend
After=network.target

[Service]
Type=simple
User=estoque
WorkingDirectory=/home/estoque/sistema_estoque/
sistema_estoque_backend
Environment=PATH=/home/estoque/sistema_estoque/
sistema_estoque_backend/venv/bin
ExecStart=/home/estoque/sistema_estoque/sistema_estoque_backend/
venv/bin/python src/main.py
Restart=always
```

```
[Install]
WantedBy=multi-user.target
```

Ative e inicie o serviço:

```
sudo systemctl daemon-reload
sudo systemctl enable sistema-estoque
sudo systemctl start sistema-estoque
```

Configuração de Firewall

Configure o firewall para permitir apenas as portas necessárias:

```
sudo ufw enable
sudo ufw allow ssh
sudo ufw allow 'Nginx Full'
sudo ufw status
```

Monitoramento e Logs

Configure rotação de logs para evitar acúmulo excessivo:

```
sudo nano /etc/logrotate.d/sistema-estoque
```

Adicione:

```
/home/estoque/sistema_estoque/logs/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 estoque estoque
}
```

Configure monitoramento básico com um script de verificação:

```
nano /home/estoque/monitor.sh
```

Adicione:

```
#!/bin/bash
if ! systemctl is-active --quiet sistema-estoque; then
    echo "Sistema de Estoque está inativo, reiniciando..."
    sudo systemctl restart sistema-estoque
fi
```

Configure no crontab para execução a cada 5 minutos:

```
crontab -e
*/5 * * * * /home/estoque/monitor.sh
```

Manutenção e Backup

Backup Completo do Sistema

Além do backup do banco de dados, é importante realizar backup completo dos arquivos do sistema. Crie um script de backup abrangente:

```
nano /home/estoque/backup_completo.sh
```

Adicione:

```
#!/bin/bash
BACKUP_DIR="/home/estoque/backups"
DATE=$(date +%Y%m%d_%H%M%S)
SYSTEM_DIR="/home/estoque/sistema_estoque"

mkdir -p $BACKUP_DIR

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

# Backup dos arquivos do sistema
tar -czf $BACKUP_DIR/system_backup_$DATE.tar.gz -C /home/
estoque sistema_estoque

# Backup das imagens de produtos
tar -czf $BACKUP_DIR/images_backup_$DATE.tar.gz -C $SYSTEM_DIR/
sistema_estoque_backend/src/static images

# Limpeza de backups antigos (manter 30 dias)
find $BACKUP_DIR -name "*backup_*.sql" -mtime +30 -delete
```

```
find $BACKUP_DIR -name "*backup_*.tar.gz" -mtime +30 -delete

echo "Backup completo realizado em $DATE"
```

Configure para execução semanal:

```
chmod +x /home/estoque/backup_completo.sh
crontab -e
0 3 * * 0 /home/estoque/backup_completo.sh
```

Atualizações do Sistema

Para atualizar o sistema, primeiro faça backup completo, depois siga estes passos:

```
# Parar o serviço
sudo systemctl stop sistema-estoque

# Backup dos arquivos atuais
cp -r /home/estoque/sistema_estoque /home/estoque/
sistema_estoque_backup_$(date +%Y%m%d)

# Aplicar atualizações (substitua pelos novos arquivos)
# ... copiar novos arquivos ...

# Atualizar dependências se necessário
cd /home/estoque/sistema_estoque/sistema_estoque_backend
source venv/bin/activate
pip install -r requirements.txt

# Recompilar frontend se necessário
cd /home/estoque/sistema_estoque/sistema_estoque_frontend
pnpm install
pnpm run build

# Reiniciar serviço
sudo systemctl start sistema-estoque
sudo systemctl status sistema-estoque
```

Monitoramento de Performance

Configure monitoramento básico de recursos do sistema:

```
nano /home/estoque/monitor_performance.sh
```

Adicione:

```
#!/bin/bash
LOG_FILE="/home/estoque/logs/performance.log"
DATE=$(date '+%Y-%m-%d %H:%M:%S')

# CPU e Memória
CPU=$(top -bn1 | grep "Cpu(s)" | awk '{print $2}' | cut -d'%' -
f1)
MEM=$(free | grep Mem | awk '{printf("%.1f", $3/$2 * 100.0)}')

# Espaço em disco
DISK=$(df -h / | awk 'NR==2{printf "%s", $5}')

# Status do serviço
if systemctl is-active --quiet sistema-estoque; then
    STATUS="ATIVO"
else
    STATUS="INATIVO"
fi

echo "$DATE - CPU: ${CPU}% | MEM: ${MEM}% | DISK: $DISK |
STATUS: $STATUS" >> $LOG_FILE
```

Configure para execução a cada hora:

```
chmod +x /home/estoque/monitor_performance.sh
crontab -e
0 * * * * /home/estoque/monitor_performance.sh
```

Limpeza e Otimização

Crie um script de limpeza para manter o sistema otimizado:

```
nano /home/estoque/limpeza.sh
```

Adicione:

```
#!/bin/bash
# Limpeza de logs antigos
find /home/estoque/logs -name "*.log" -mtime +90 -delete

# Limpeza de arquivos temporários
find /tmp -name "*estoque*" -mtime +1 -delete

# Otimização do banco de dados PostgreSQL
sudo -u postgres psql -d sistema_estoque -c "VACUUM ANALYZE;"
```

```
# Limpeza de cache do sistema
sudo apt autoremove -y
sudo apt autoclean

echo "Limpeza e otimização concluída em $(date)"
```

Configure para execução mensal:

```
 chmod +x /home/estoque/limpeza.sh
crontab -e
0 2 1 * * /home/estoque/limpeza.sh
```

Solução de Problemas Técnicos

Problemas de Conectividade

Se o sistema não conseguir se conectar ao Mercado Livre, primeiro verifique a conectividade básica:

```
 # Teste de conectividade com a API do ML
curl -I https://api.mercadolibre.com

# Verificar DNS
nslookup api.mercadolibre.com

# Verificar portas de saída
telnet api.mercadolibre.com 443
```

Verifique se não há firewall ou proxy bloqueando as conexões HTTPS de saída. O sistema precisa acessar os domínios do Mercado Livre na porta 443.

Problemas de Performance

Para diagnosticar problemas de performance, monitore os recursos do sistema:

```
 # Monitorar CPU e memória em tempo real
htop

# Verificar uso de disco
df -h
du -sh /home/estoque/sistema_estoque/*

# Monitorar conexões de rede
```

```
netstat -tulpn | grep :5001

# Verificar logs do sistema
journalctl -u sistema-estoque -f
```

Se o sistema estiver lento, verifique se há processos consumindo recursos excessivos e considere otimizar consultas ao banco de dados ou aumentar recursos do servidor.

Problemas de Banco de Dados

Para problemas relacionados ao PostgreSQL:

```
# Verificar status do PostgreSQL
sudo systemctl status postgresql

# Verificar logs do PostgreSQL
sudo tail -f /var/log/postgresql/postgresql-14-main.log

# Testar conexão manual
psql -h localhost -U estoque_user -d sistema_estoque

# Verificar bloqueios no banco
sudo -u postgres psql -d sistema_estoque -c "SELECT * FROM
pg_locks WHERE NOT granted;"
```

Para corrigir problemas de conexão, verifique as configurações de autenticação no arquivo pg_hba.conf e reinicie o serviço se necessário.

Recuperação de Desastres

Em caso de falha crítica do sistema, siga este procedimento de recuperação:

1. **Avalie o dano**: Determine se é um problema de hardware, software, ou dados corrompidos.

2. **Restaure do backup mais recente**:

```
# Parar o serviço
sudo systemctl stop sistema-estoque

# Restaurar banco de dados
sudo -u postgres psql -d sistema_estoque < /home/estoque/
backups/db_backup_YYYYMMDD_HHMMSS.sql

# Restaurar arquivos do sistema
```

```
cd /home/estoque
tar -xzf backups/system_backup_YYYYMMDD_HHMMSS.tar.gz
```

1. **Verificar integridade**:

```
# Testar conexão com banco
psql -h localhost -U estoque_user -d sistema_estoque -c "SELECT
COUNT(*) FROM produtos;"

# Testar API
curl http://localhost:5001/api/produtos
```

1. **Reiniciar serviços**:

```
sudo systemctl start sistema-estoque
sudo systemctl restart nginx
```

Logs e Diagnóstico

Para diagnóstico avançado, configure logs detalhados:

```
# Logs do sistema
journalctl -u sistema-estoque --since "1 hour ago"

# Logs do Nginx
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log

# Logs do PostgreSQL
sudo tail -f /var/log/postgresql/postgresql-14-main.log
```

Configure alertas automáticos para erros críticos:

```
nano /home/estoque/monitor_errors.sh
```

Adicione:

```
#!/bin/bash
ERROR_COUNT=$(journalctl -u sistema-estoque --since "5 minutes
ago" | grep -i error | wc -l)

if [ $ERROR_COUNT -gt 5 ]; then
    echo "ALERTA: $ERROR_COUNT erros detectados no sistema de
```

```
estoque" | mail -s "Alerta Sistema Estoque" admin@empresa.com
fi
```

Este guia fornece uma base sólida para instalação, configuração e manutenção do sistema em ambiente de produção. Para situações específicas não cobertas, consulte a documentação adicional ou entre em contato com o suporte técnico.

Versão do Guia: 1.0
Última Atualização: Junho 2025
Desenvolvido por: Manus AI