Manual de Build - EPI Guard

Sumário

Pré-requisitos de Desenvolvimento

Software Necessário

1. **Node.js 16+**: `bash # Verificar versão node --version npm --version`
2. **Git**: `bash git --version`
3. **Python 3.8+** (para API): `bash python --version # ou python3 --version`

Ferramentas Opcionais

1. **Wine** (para build Windows no Linux): `bash # Ubuntu/Debian sudo apt install wine`
2. **Docker** (para builds isolados): `bash docker --version`

Configuração do Ambiente

Passo 1: Clonar o Repositório

```
# Clonar o projeto
git clone <repository-url>
cd epi-guard-electron

# Ou extrair do zip
```

```
unzip epi-guard-electron.zip
cd epi-guard-electron
```

Passo 2: Instalar Dependências

```
# Instalar dependências do projeto
npm install

# Verificar instalação
npm list --depth=0
```

Passo 3: Configurar Variáveis de Ambiente

1. **Criar arquivo `.env`** (opcional): `bash # .env REACT_APP_API_URL=http://localhost:8000 ELECTRON_IS_DEV=true`
2. **Configurar scripts personalizados**: `json // package.json { "scripts": { "start:api": "cd ../helmet_detection_system/api && python main.py", "dev:full": "concurrently \"npm run start:api\" \"npm run electron-dev\"" } }`

Build de Desenvolvimento

Servidor de Desenvolvimento React

```
# Iniciar servidor webpack
npm start

# Acesso: http://localhost:3000
```

Desenvolvimento com Electron

```
# Iniciar Electron + React em modo dev
npm run electron-dev

# Ou separadamente:
# Terminal 1:
npm start

# Terminal 2 (aguardar servidor iniciar):
npm run electron
```

Hot Reload

O ambiente de desenvolvimento inclui: - **Hot Reload** para React - **Auto-restart** para Electron - **DevTools** habilitado - **Source Maps** para debug

Debug e Inspeção

```
# Abrir DevTools automaticamente
ELECTRON_IS_DEV=true npm run electron

# Debug do processo main
npm run electron -- --inspect=9229

# Debug com breakpoints
npm run electron -- --inspect-brk=9229
```

Build de Produção

Passo 1: Build do React

```
# Gerar build otimizado
npm run build

# Verificar saída
ls -la build/
```

Saída esperada:

```
build/
├── index.html          # HTML principal
├── bundle.js           # JavaScript minificado (~313 KB)
├── bundle.js.map       # Source map
└── assets/             # Recursos estáticos
```

Passo 2: Testar Build Local

```
# Servir build localmente
npx serve build

# Ou testar com Electron
npm run electron
```

Passo 3: Otimizações

1. **Análise do Bundle**: ```bash # Instalar analisador npm install --save-dev webpack-bundle-analyzer

Analisar bundle npx webpack-bundle-analyzer build/bundle.js ```

1. **Otimizações de Performance**: `javascript // webpack.config.js module.exports = { optimization: { splitChunks: { chunks: 'all', }, usedExports: true, sideEffects: false } };`

Geração de Instaladores

Configuração do Electron Builder

O arquivo `package.json` já contém a configuração:

```
{
  "build": {
    "appId": "com.epiguard.app",
    "productName": "EPI Guard",
    "directories": {
      "output": "dist"
    },
    "files": [
      "build/**/*",
      "main.js",
      "preload.js",
      "package.json"
    ],
    "linux": {
      "target": "AppImage",
      "category": "Utility"
    },
    "win": {
      "target": "nsis"
    }
  }
}
```

Build para Linux

```
# Gerar AppImage
npm run dist-linux
```

```
# Ou apenas empacotar
npm run pack
```

Saída:

```
dist/
├── EPI Guard-1.0.0.AppImage     # Instalador (~103 MB)
├── linux-unpacked/              # Versão descompactada
└── builder-debug.yml            # Log de build
```

Build para Windows

```
# No Linux (requer Wine)
npm run dist-win

# No Windows
npm run dist
```

Saída:

```
dist/
├── EPI Guard Setup 1.0.0.exe    # Instalador NSIS
├── win-unpacked/                # Versão descompactada
└── builder-debug.yml            # Log de build
```

Build Multiplataforma

```
# Todos os targets configurados
npm run dist

# Targets específicos
npx electron-builder --linux --win
```

Distribuição

Preparação para Distribuição

1. **Verificar Builds**: ```bash # Testar AppImage chmod +x "dist/EPI Guard-1.0.0.AppImage" ./dist/EPI\ Guard-1.0.0.AppImage

Verificar assinatura (se configurada) gpg --verify dist/*.AppImage.sig ```

1. **Gerar Checksums**: `bash # SHA256 sha256sum dist/*.AppImage > dist/checksums.txt sha256sum dist/*.exe >> dist/checksums.txt`

2. **Compactar para Distribuição**: `bash # Criar arquivo de distribuição zip -r epi-guard-v1.0.0-dist.zip dist/`

Estrutura de Release

```
release/
├── linux/
│   ├── EPI Guard-1.0.0.AppImage
│   └── checksums.txt
├── windows/
│   ├── EPI Guard Setup 1.0.0.exe
│   └── checksums.txt
├── source/
│   └── epi-guard-electron-v1.0.0.zip
└── README.md
```

Versionamento

1. **Atualizar Versão**: `bash # Incrementar versão npm version patch # 1.0.0 -> 1.0.1 npm version minor # 1.0.0 -> 1.1.0 npm version major # 1.0.0 -> 2.0.0`

2. **Tag de Release**: `bash git tag v1.0.0 git push origin v1.0.0`

Troubleshooting

Problemas Comuns

1. **Erro: "Module not found"**: `bash # Limpar cache e reinstalar rm -rf node_modules package-lock.json npm install`

2. **Build falha no Windows**: ```bash # Instalar Wine (Linux) sudo apt install wine

Ou usar Docker docker run --rm -v $(pwd):/project electronuserland/builder:wine ```

1. **Bundle muito grande**: ```bash # Analisar dependências npm install --save-dev webpack-bundle-analyzer npx webpack-bundle-analyzer build/bundle.js

Remover dependências desnecessárias npm uninstall ```

1. **Electron não inicia**: ```bash # Debug verbose DEBUG=electron* npm run electron

Verificar logs tail -f ~/.config/EPI\ Guard/logs/main.log ```

Logs de Build

1. **Webpack Logs**: ```bash # Build verbose npm run build -- --verbose

Analisar stats npm run build -- --json > stats.json ```

1. **Electron Builder Logs**: ```bash # Debug completo DEBUG=electron-builder npm run dist

Verificar arquivo de debug cat dist/builder-debug.yml ```

Performance de Build

1. **Cache do Webpack**: `javascript // webpack.config.js module.exports = { cache: { type: 'filesystem', }, };`
2. **Build Paralelo**: `bash # Usar todos os cores npm run dist -- --parallel`
3. **Build Incremental**: `bash # Apenas arquivos alterados npm run build -- --watch`

Scripts Úteis

Package.json Scripts

```
{
  "scripts": {
    "clean": "rm -rf build dist node_modules",
    "prebuild": "npm run clean:build",
    "clean:build": "rm -rf build",
    "clean:dist": "rm -rf dist",
    "build:analyze": "npm run build && npx webpack-bundle-
analyzer build/bundle.js",
    "dist:all": "npm run dist-linux && npm run dist-win",
    "postdist": "npm run checksums",
    "checksums": "cd dist && sha256sum *.AppImage *.exe >
checksums.txt"
```

```
  }
}
```

Automação com Makefile

```
# Makefile
.PHONY: install build dist clean

install:
    npm install

build:
    npm run build

dist: build
    npm run dist

clean:
    rm -rf node_modules build dist

release: dist
    zip -r epi-guard-v$(shell node -p "require('./
package.json').version").zip dist/
```

CI/CD

GitHub Actions

```
# .github/workflows/build.yml
name: Build and Release

on:
  push:
    tags: ['v*']

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
      - run: npm install
      - run: npm run dist
      - uses: actions/upload-artifact@v3
```

```
        with:
          name: dist
          path: dist/
```

EPI Guard v1.0.0 - Manual de Build
Última atualização: Junho 2025