EPI Guard - Sistema de Detecção Automática de Capacetes

Visão Geral

O **EPI Guard** é um sistema completo de detecção automática de capacetes de segurança desenvolvido com tecnologias modernas. O sistema utiliza inteligência artificial (YOLOv8) para detectar pessoas sem capacete em tempo real através de câmeras IP, oferecendo uma interface desktop moderna e intuitiva.

Características Principais

- **Interface Desktop Moderna**: Desenvolvida com Electron + React
- **Detecção em Tempo Real**: Modelo YOLOv8 integrado via API FastAPI
- **Fidelidade Visual**: Interface baseada pixel-a-pixel no protótipo Figma
- **Multiplataforma**: Compatível com Windows e Linux
- **Offline First**: Funciona sem conexão com sincronização posterior

Tecnologias Utilizadas

Frontend (Interface Desktop)

- **Electron 28.3.3**: Framework para aplicações desktop
- **React 18.2.0**: Biblioteca para interface do usuário
- **Webpack 5.99.9**: Bundler e servidor de desenvolvimento
- **Axios 1.6.2**: Cliente HTTP para comunicação com API
- **Lucide React**: Biblioteca de ícones moderna

Backend (API)

- **FastAPI**: Framework web moderno para Python
- **YOLOv8**: Modelo de detecção de objetos
- **OpenCV**: Processamento de imagens
- **SQLite**: Banco de dados local

Build e Distribuição

- **Electron Builder**: Geração de instaladores
- **Babel**: Transpilação JavaScript
- **CSS Modules**: Estilização modular

Estrutura do Projeto

```
epi-guard-electron/
├── src/                    # Código fonte React
│   ├── components/         # Componentes reutilizáveis
│   ├── views/              # Páginas da aplicação
│   ├── services/           # Serviços e contextos
│   ├── assets/             # Recursos estáticos
│   └── styles/             # Estilos globais
├── build/                  # Build de produção
├── dist/                   # Builds distribuíveis
├── tests/                  # Testes da aplicação
├── main.js                 # Processo principal Electron
├── preload.js              # Script de preload
├── webpack.config.js       # Configuração Webpack
└── package.json            # Dependências e scripts
```

Funcionalidades Implementadas

🔐 Sistema de Autenticação

- Login com usuário e senha
- Controle de perfis (Admin, Supervisor, Técnico)
- Sessão persistente
- Validação de permissões

📊 Dashboard Interativo

- Estatísticas em tempo real
- Monitoramento de câmeras ativas
- Taxa de conformidade
- Histórico de detecções
- Alertas e notificações

📹 Gerenciamento de Câmeras

- Configuração de câmeras RTSP

- Visualização em tempo real
- Detecção manual e automática
- Teste de conectividade
- Histórico por câmera

📈 Relatórios e Análises

- Geração de relatórios em PDF
- Exportação para Excel e CSV
- Filtros avançados (data, câmera, tipo)
- Estatísticas detalhadas
- Gráficos de tendência

👥 Gerenciamento de Usuários

- CRUD completo de usuários
- Controle de perfis e permissões
- Histórico de login
- Busca e filtros
- Validação de dados

⚙️ Configurações Avançadas

- Configuração da API
- Parâmetros de detecção
- Sistema de notificações
- Gerenciamento de armazenamento
- Configurações de interface

Requisitos do Sistema

Mínimos

- **Sistema Operacional**: Windows 10+ ou Linux (Ubuntu 18.04+)
- **RAM**: 4 GB
- **Processamento**: Intel i3 ou equivalente
- **Armazenamento**: 500 MB livres
- **Rede**: Conexão com câmeras IP (opcional para webcam)

Recomendados

- **Sistema Operacional**: Windows 11 ou Linux (Ubuntu 22.04+)
- **RAM**: 8 GB ou mais
- **Processamento**: Intel i5 ou equivalente
- **Armazenamento**: 2 GB livres
- **GPU**: Dedicada (para melhor performance de detecção)

Scripts Disponíveis

Desenvolvimento

```
npm start              # Servidor de desenvolvimento React
npm run electron-dev   # Desenvolvimento com Electron
```

Build

```
npm run build          # Build de produção
npm run dist           # Gerar instalador (Linux)
npm run dist-win       # Gerar instalador (Windows)
npm run dist-linux     # Gerar instalador (Linux)
```

Utilitários

```
npm run pack           # Empacotar sem gerar instalador
npm test               # Executar testes
```

Arquitetura da Aplicação

Fluxo de Dados

1. **Interface React** → Componentes modulares e reutilizáveis
2. **Context API** → Gerenciamento de estado global
3. **Axios Client** → Comunicação com API FastAPI
4. **Electron Main** → Processo principal da aplicação
5. **Preload Script** → Ponte segura entre renderer e main

Padrões Utilizados

- **Component-Based Architecture**: Componentes React modulares
- **Context Pattern**: Gerenciamento de estado centralizado
- **Service Layer**: Abstração da comunicação com API
- **Responsive Design**: Interface adaptável a diferentes resoluções

Segurança

Electron Security

- **Context Isolation**: Habilitado para isolamento de contexto
- **Node Integration**: Desabilitado no renderer
- **Preload Script**: Comunicação segura entre processos
- **CSP**: Content Security Policy configurado

Autenticação

- **Session Management**: Controle de sessão local
- **Permission Validation**: Validação de permissões por funcionalidade
- **Secure Storage**: Armazenamento seguro de configurações

Performance

Otimizações Implementadas

- **Code Splitting**: Bundle otimizado (313 KB)
- **Lazy Loading**: Carregamento sob demanda
- **Memoization**: Cache de componentes React
- **Debouncing**: Otimização de chamadas à API

Métricas

- **Bundle Size**: 313 KB (minificado)
- **App Size**: 103 MB (AppImage)
- **Startup Time**: < 3 segundos
- **Memory Usage**: ~150 MB em execução

Licença

Este projeto está licenciado sob a Licença MIT - veja o arquivo LICENSE para detalhes.

Suporte

Para suporte técnico ou dúvidas sobre o sistema, entre em contato com a equipe de desenvolvimento.

EPI Guard v1.0.0 - Sistema de Detecção Automática de Capacetes
Desenvolvido com ❤️ pela equipe EPI Guard