QA Dashboard App - Manual do Usuário

Visão Geral

O QA Dashboard App é um aplicativo desktop desenvolvido em Python para análise automatizada de métricas de Quality Assurance (QA) a partir de arquivos PDF. O aplicativo oferece uma interface web interativa para visualização de dados e funcionalidades de processamento automático.

Características Principais

✅ Funcionalidades Implementadas

- **Extração Robusta de PDFs**: Suporte a múltiplas técnicas de extração
- PyPDF2 para PDFs textuais
- pdfplumber para tabelas estruturadas
- tabula-py para tabelas complexas
- OCR com Tesseract para PDFs escaneados
- **Dashboard Interativo**: Interface web moderna com Streamlit
- Upload de arquivos via drag-and-drop
- Visualizações interativas com Plotly
- KPIs em tempo real
- Exportação para CSV
- **Processamento Automático**: Sistema de agendamento
- Monitoramento de pasta de entrada
- Processamento diário agendado

- Organização automática de arquivos
- **Análise de Métricas**: Cálculos automáticos de KPIs
- Total de casos de teste
- Percentual de execução
- Percentual de sucesso
- Distribuição por status

Requisitos do Sistema

Pré-requisitos Obrigatórios

- **Python 3.8+**: Linguagem de programação principal
- **Java Runtime Environment**: Necessário para tabula-py
- **Sistema Operacional**: Windows 10/11, Linux, macOS

Pré-requisitos Opcionais

- **Tesseract OCR**: Para processamento de PDFs escaneados
- **poppler-utils**: Para conversão PDF para imagem

Dependências Python

```
 pandas>=1.4.0
streamlit>=1.28.0
plotly>=5.15.0
PyPDF2>=3.0.0
pdfplumber>=0.7.0
tabula-py>=2.5.0
pytesseract>=0.3.10
pdf2image>=3.1.0
schedule>=1.2.0
```

Instalação

Método 1: Pacote Portável (Recomendado)

1. Extraia o arquivo `QA_Dashboard_Portable.zip`
2. Execute `Instalar_Dependencias.bat` (Windows) ou `pip install -r requirements.txt`
3. Use os arquivos `.bat` para executar as funcionalidades

Método 2: Instalação Manual

```
# Clone ou baixe o projeto
cd qa_dashboard_app

# Instale as dependências
pip install -r requirements.txt

# Execute o aplicativo
python app.py
```

Guia de Uso

Iniciando o Dashboard

```
# Método 1: Comando direto
python app.py dashboard

# Método 2: Windows batch file
Iniciar_Dashboard.bat
```

O dashboard será aberto automaticamente no navegador em `http://localhost:8501`

Upload e Análise de PDFs

1. **Acesse o Dashboard**: Abra o navegador em localhost:8501
2. **Upload do Arquivo**: Use a área de upload na barra lateral

3. **Visualização Automática**: Os dados são processados e exibidos automaticamente
4. **Exportação**: Use o botão "Baixar CSV" para exportar os dados

Processamento Automático

Configuração

1. Coloque os PDFs na pasta `input_pdfs/`
2. Execute o agendador:

```
python app.py scheduler
```

Funcionamento

- **Monitoramento**: Verifica a pasta `input_pdfs/` diariamente às 09:00
- **Processamento**: Extrai dados e gera CSVs na pasta `processed_data/`
- **Organização**: Move PDFs processados para `input_pdfs/processed/`

Teste de Funcionalidade

```
# Testa o processamento imediatamente
python app.py test
```

Estrutura do Projeto

```
qa_dashboard_app/
├── app.py                    # Script principal
├── dashboard.py              # Interface Streamlit
├── scheduler.py              # Agendador automático
├── build.py                  # Script de empacotamento
├── requirements.txt          # Dependências Python
├── src/                      # Módulos de processamento
│   ├── pdf_extractor.py      # Extração de PDFs
│   └── data_processor.py     # Processamento de dados
├── input_pdfs/               # Pasta de entrada (automático)
├── processed_data/           # Pasta de saída (automático)
└── docs/                     # Documentação
```

Formatos de PDF Suportados

PDFs Textuais

- Documentos com texto selecionável
- Relatórios gerados por ferramentas de QA
- Exports de sistemas de teste

PDFs com Tabelas

- Tabelas estruturadas com bordas
- Dados tabulares organizados
- Relatórios de métricas formatados

PDFs Escaneados

- Documentos digitalizados (requer OCR)
- Imagens de relatórios impressos
- Screenshots de dashboards

Formato Esperado dos Dados

O aplicativo procura por tabelas com a seguinte estrutura:

```
 Status          | Total
Passou          | 100
Falhou          | 10
Bloqueado       | 5
Não Executado   | 20
```

Solução de Problemas

Erro: "Java não encontrado"

Problema: tabula-py requer Java **Solução**: - Windows: Instale Java JRE/JDK - Linux: `sudo apt-get install openjdk-11-jre` - macOS: `brew install openjdk`

Erro: "Tesseract não encontrado"

Problema: OCR não está instalado **Solução**: - Windows: Baixe do GitHub oficial do Tesseract - Linux: `sudo apt-get install tesseract-ocr` - macOS: `brew install tesseract`

Erro: "Nenhuma tabela encontrada"

Problema: PDF não contém dados estruturados **Soluções**: 1. Verifique se o PDF contém tabelas visíveis 2. Teste com OCR se for PDF escaneado 3. Verifique o formato dos dados

Dashboard não carrega

Problema: Porta 8501 ocupada **Solução**:

```
# Use porta alternativa
streamlit run dashboard.py --server.port 8502
```

Personalização

Modificando KPIs

Edite o arquivo `src/data_processor.py` na função `calculate_kpis()`:

```
def calculate_kpis(df_status):
    # Adicione novos cálculos aqui
    kpis["Novo_KPI"] = sua_formula
    return kpis
```

Adicionando Gráficos

Edite o arquivo `dashboard.py` na função `display_dashboard()`:

```
# Adicione novos gráficos Plotly
fig_novo = px.scatter(df_status, x='Status', y='Total')
st.plotly_chart(fig_novo)
```

Configurando Agendamento

Edite o arquivo `scheduler.py` para alterar horários:

```
# Altere o horário de execução
schedule.every().day.at("14:30").do(self.process_pdfs)
```

Limitações Conhecidas

1. **Formato de Dados**: Requer estrutura específica de tabelas
2. **Idioma OCR**: Otimizado para português/inglês
3. **Tamanho de Arquivo**: PDFs muito grandes podem ser lentos
4. **Dependências**: Requer Java e Python instalados

Suporte e Manutenção

Logs de Erro

Os logs são exibidos no console durante a execução. Para debug:

```
# Execute com verbose
python app.py dashboard --verbose
```

Atualizações

Para atualizar dependências:

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

Backup de Dados

Recomenda-se backup regular das pastas: - `processed_data/` - Dados processados - `input_pdfs/processed/` - PDFs processados

Conclusão

O QA Dashboard App oferece uma solução completa para análise automatizada de métricas de QA, combinando extração robusta de dados, visualização interativa e processamento automático. A arquitetura modular permite fácil personalização e extensão conforme necessidades específicas.

Para suporte adicional ou relatório de bugs, consulte a documentação técnica ou entre em contato com a equipe de desenvolvimento.