Desafio Sympla - RPA - Documento de Design de Processo (PDD)

Resumo do Projeto

Este documento detalha o design do processo de automação desenvolvido para coletar, estruturar e analisar dados sobre estados brasileiros, suas capitais, populações e regiões. O robô, criado em Python, utiliza Selenium para scraping, pandas para manipulação de dados e SQLite para armazenamento. O objetivo é extrair insights por meio de consultas SQL e exportar os resultados em arquivos formatados.

Objetivo do Processo

Automatizar a coleta, processamento e análise de dados de estados brasileiros para otimizar o tempo e reduzir erros manuais, fornecendo insights acionáveis para decisões baseadas em dados.

Escopo do Processo

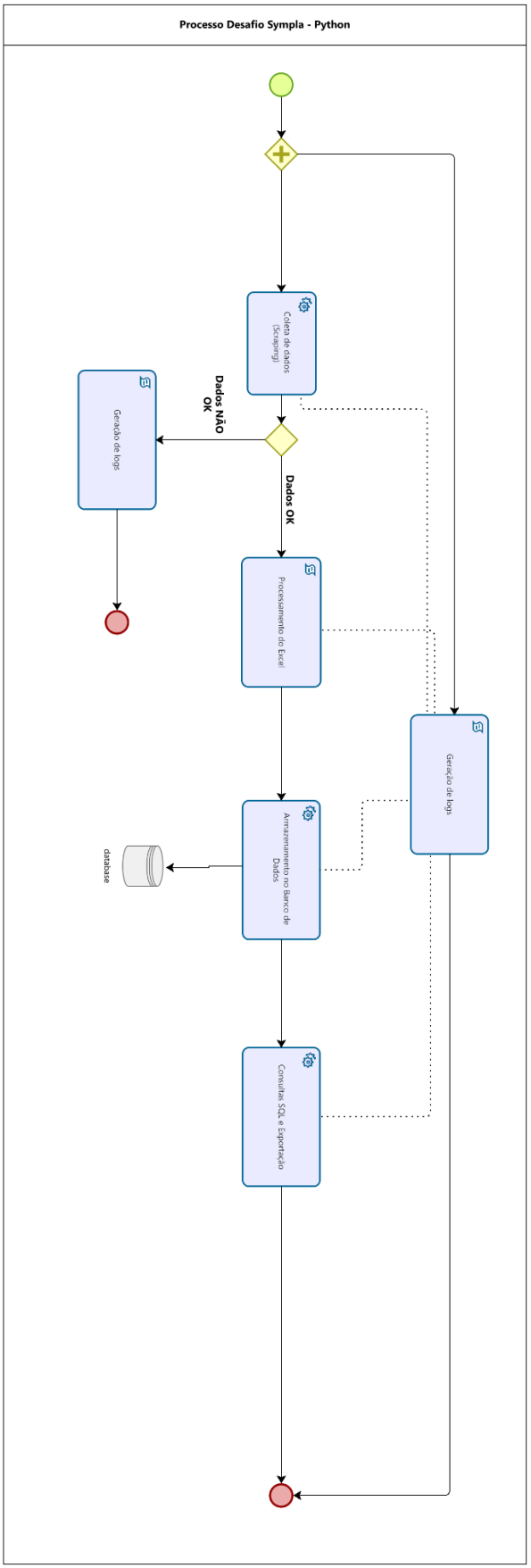
Incluído:

- Coleta de dados sobre estados, capitais e regiões em domínios web.
- Processamento de dados de populações a partir de arquivos Excel.
- Armazenamento de dados no banco SQLite.
- Consultas SQL e exportação de arquivos em CSV e XLS.
- Geração de logs para monitoramento do processo.

Excluído:

- Análise de dados além das consultas especificadas.
- Integração com sistemas externos além dos especificados.

Diagrama do Fluxo de Processo



Fluxo do Processo

1. Evento de Início

- **Ação**: O processo é iniciado manualmente pelo usuário ao executar o arquivo `main.py`.
- **Resultado Esperado**: O ambiente é carregado, e o processo avança para a etapa de coleta de dados.

2. Coleta de Dados (Scraping)

- **Ação**: Utiliza Selenium para acessar o site de referência, localizar a tabela HTML e extrair dados sobre estados, capitais, regiões e siglas.
- **Logs Gerados**: Confirmação do início e conclusão da coleta ou erro em caso de falha na conexão.
- **Resultado Esperado**: Dados estruturados em listas Python para processamento posterior.

3. Processamento do Excel

- **Ação**: Utiliza Pandas para ler o arquivo Excel com informações de populações, tratar os dados (como remoção de caracteres e conversão de tipos) e combiná-los com os dados coletados.
- **Logs Gerados**: Registro do processamento do arquivo, incluindo possíveis erros de estrutura.
- **Resultado Esperado**: DataFrame consolidado contendo informações completas sobre estados, capitais, regiões e populações.

4. Armazenamento no Banco de Dados

- **Ação**: Insere ou atualiza os dados no banco de dados SQLite, garantindo a consistência e eliminando duplicatas.
- **Logs Gerados**: Informações sobre a criação ou atualização do banco de dados e erros, caso ocorram.
- **Resultado Esperado**: Dados salvos no arquivo `estados.db` no diretório `database/`.

5. Consultas SQL e Exportação

Ação: Executa consultas SQL no banco de dados para gerar insights, como:

- Top 3 regiões mais populosas.
- Número de capitais por região.
- Dois estados com as capitais mais populosas. Exporta os resultados em formatos CSV e XLS.

Logs Gerados: Sucesso ou erro para cada consulta e exportação.

Resultado Esperado: Arquivos gerados no diretório `outputs/`.

6. Geração de Logs

- **Ação**: Registra informações detalhadas sobre cada etapa do processo, incluindo sucessos, erros e exceções.
- **Logs Gerados**: Arquivos de log no diretório `logs/`, contendo rastreabilidade completa do processo.
- **Resultado Esperado**: Logs acessíveis para diagnóstico e auditoria.

7. Evento de Finalização

O processo é concluído com sucesso, e os seguintes outputs são gerados e armazenados em seus respectivos diretórios:

Arquivos de Resultados (diretório *outputs/*):

```
Unset

    top3_regioes_populosas.csv: Arquivo contendo as 3 regiões mais populosas e suas populações totais.

    regioes_n_capitais.xls: Arquivo XLS listando as regiões e a quantidade de capitais em cada uma.

    estados_mais_populosos.xls: Arquivo XLS com os dois estados cujas capitais possuem as maiores populações.
```

Banco de Dados (diretório *database/*):

```
Unset

    estados.db: Banco de dados SQLite contendo os dados estruturados dos estados, capitais, regiões e populações.
```

Logs do Processo (diretório logs/):

```
Unset

    Arquivos de log contendo o registro detalhado de todas as etapas executadas no processo, incluindo sucessos e falhas.
```

O evento de finalização assegura que todos os outputs estejam devidamente gerados, organizados e prontos para uso.

Regras de Negócio

- Apenas dados com formatação correta são processados.
- Caso um erro seja encontrado, ele é registrado nos logs, e o processo continua.

Exceções e Tratamento de Erros

O processo foi projetado para identificar e registrar erros em todas as etapas críticas, garantindo rastreabilidade e simplicidade no tratamento de exceções. Segue o detalhamento:

Erro de Conexão ao Site:

Em caso de falha na conexão durante o scraping, o erro é capturado e registrado nos logs com o nível **ERROR**. O processo é encerrado imediatamente.

Erro no Banco de Dados:

Se houver falha ao acessar ou manipular o banco de dados SQLite (ex.: arquivo corrompido), o erro é registrado nos logs com o nível **ERROR**, e o processo é encerrado. Isso evita causar danos ao banco e permite que o administrador corrija o problema antes de reiniciar.

Requisitos Funcionais:

- Coleta e processamento automático de dados.
- Exportação em formatos padronizados especificados pelo negócio.

Dependências e Pré-requisitos

- **Softwares Necessários**:
  - Python 3.11
  - Bibliotecas: Selenium, Pandas, Pyexcel, SQLite.
- **Hardware**:
  - Computador com 4GB de RAM e conexão à internet.

Métricas e KPIs

- Quantidade de dados processados por execução.

- Taxa de erros por execução.

Análise de Riscos

- **Risco**: Site com a tabela fora do ar.
    - **Mitigação**: Salvar uma cópia local para emergências.
- **Risco**: Arquivo Excel corrompido.
    - **Mitigação**: Validar formato antes de processar.

Plano de Implementação

- **Fase 1**: Configuração do ambiente.
- **Fase 2**: Desenvolvimento do código.
- **Fase 3**: Testes em diferentes ambientes.
- **Fase 4**: Implantação final.

Plano de Manutenção

- Revisão periódica do código a cada mês para mitigação de erros que podem ser causados por alterações no site de pesquisa.
- Atualizações das bibliotecas utilizadas.

Guia de Uso do Sistema

1. **Clonar o Repositório:**

```
Unset
git clone
https://github.com/Akyllesbarros/desafio-sympla-dev-rpa.git
```

2. **Configurar o Ambiente**:

- No Windows:

```
Unset
python -m venv venv
venv\Scripts\activate
pip install -r requirements.txt
```

- No Mac:

```
Unset
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

3. **Executar o Sistema**:

```
Unset
python main.py
```

4. **Verificar Resultados**:
1. Outputs na pasta `outputs/`.
2. Logs na pasta `logs/`.
3. Banco de dados em `database/estados.db`.

Estrutura de Arquivos

```
Unset
/desafio-sympla-dev-rpa
├── main.py                  # Arquivo principal para execução do processo.
├── scraping.py              # Função responsável pela coleta de dados (scraping).
├── data_processing.py       # Funções para manipulação e processamento de dados.
├── database.py              # Funções de manipulação do banco de dados SQLite.
├── logger_config.py         # Configuração e estruturação dos logs.
├── requirements.txt         # Dependências necessárias para o ambiente virtual.
├── venv/                    # Ambiente virtual
├── database/                # Diretório para o banco de dados SQLite.
|   └── estados.db           # Banco de dados contendo os dados estruturados.
├── logs/                    # Diretório para armazenar os arquivos de log.
|   └── log_YYYY-MM-DD.log   # Logs detalhados por execução.
├── outputs/                 # Diretório para exportação de resultados.
|   ├── top3_regioes_populosas.csv
|   ├── regioes_n_capitais.xls
|   └── estados_mais_populosos.xls
```

Conclusão

Este Documento de Design de Processo (PDD) detalha todas as etapas e decisões envolvidas na automação do processo de coleta, processamento, análise e exportação de dados sobre estados brasileiros, suas capitais, populações e regiões. A implementação foi estruturada com foco em eficiência, rastreabilidade e conformidade com as melhores práticas de automação.

Os resultados esperados incluem a geração de insights acionáveis a partir dos dados processados e a eliminação de esforços manuais repetitivos, garantindo maior confiabilidade e rapidez. O sistema foi desenvolvido com modularidade, permitindo futuras expansões ou integrações, e foi validado com testes em diferentes ambientes para garantir sua funcionalidade e portabilidade.

Os logs detalhados oferecem uma visão completa das execuções, permitindo diagnósticos rápidos e suporte contínuo. O banco de dados e os outputs gerados estão organizados em diretórios específicos, assegurando acessibilidade e clareza.

Próximos Passos

- Monitorar o desempenho do sistema em produção e coletar feedback dos usuários para ajustes.
- Realizar manutenções periódicas, como atualizações de dependências e revisão do código.
- Explorar possibilidades de integrações adicionais e melhorias no processo de exportação.

Com este documento, espera-se proporcionar uma referência sólida e completa para todas as partes interessadas no projeto.

Desenvolvido por:

Nome do Desenvolvedor: **Akylles Barros**

Função: **Desenvolvedor RPA**

Data: **09/12/2024**

Revisado por:

Nome do Revisor:

Função:

Data: