Ransomwatch

Esse projeto acessa os links onlines do site https://cti.fyi/ e checa informações publicadas em um dado período de tempo.

Estrutura de Diretórios e Arquivos

```
/tor-docker
  └── Dockerfile

.env
.gitignore
get_html.py
get_url.py
groups.json
grupos.py
main.py
ransomscraper.pdf
README.md
requirements.txt
save_pdf.py
send_email.py
```

Classes

Main

A classe `Main` é responsável por orquestrar o fluxo de execução do processo de scraping de dados, geração de relatórios em PDF e envio de e-mails. Ela gerencia a integração entre diferentes componentes do sistema, configurando e controlando a execução das tarefas principais.

Atributos da Classe

- `BASE_URL` (`str`): A URL base para a realização do scraping, carregada a partir do arquivo `.env`.
- `DAYS` (`int`): O número de dias a serem utilizados como critério de filtragem nos dados raspados, obtido através de uma variável de ambiente.
- `GERAR_PDF` (`str`): Define se o PDF será gerado no final do processo. O valor é extraído da variável de ambiente `GERAR_PDF`.
- `url_scraper` (`GetOnlineUrls`): Instância da classe `GetOnlineUrls`, utilizada para coletar URLs online.
- `grupos` (`Grupos`): Instância da classe `Grupos`, utilizada para processar dados de diferentes grupos de acordo com o intervalo de dias especificado.
- `pdf_generator` (`DataFrameToMarkdownPDF`): Instância da classe `DataFrameToMarkdownPDF`, responsável pela geração do arquivo PDF contendo os dados processados.
- `get_html` (`GetHtml`): Instância da classe `GetHtml`, usada para extrair o conteúdo HTML das URLs obtidas.

Métodos da Classe

1. `__init__(self)`
   - Inicializa a classe carregando as variáveis de ambiente e configurando os atributos principais, como `BASE_URL`, `DAYS` e `GERAR_PDF`. Além disso, inicializa as instâncias das classes auxiliares necessárias para o fluxo de execução.
2. `load_days(self)`
   - Carrega a variável `DAYS` do ambiente e tenta convertê-la para um inteiro. Caso a conversão falhe, um erro é registrado no log e uma exceção é levantada.
3. `run(self)`
   - Método principal que executa todo o fluxo da aplicação:
     - Chama o método `scrape_urls()` para coletar e salvar URLs online.
     - Se `GERAR_PDF` estiver configurado como "YES", extrai dados HTML, prepara o DataFrame, processa os grupos de dados, gera o PDF e envia o e-mail.

    - Caso contrário, registra um log informando que as URLs foram salvas no arquivo `groups.json`.

4. `scrape_urls(self)`
    - Obtém as URLs online utilizando a instância `url_scraper` e as salva. Se `BASE_URL` não estiver configurada, um erro é registrado no log.
5. `prepare_dataframe(self, df)`
    - Filtra o DataFrame para incluir apenas os dados disponíveis (onde `infos` é `True`), remove duplicatas nas colunas `html` e `grupo`, e reorganiza os índices.
6. `processar_grupos(self, df_to_analyze)`
    - Itera sobre o DataFrame analisado e chama o método `scrape_group()` correspondente para cada grupo, coletando os resultados em uma lista. Registra erros no log caso ocorram durante o processamento.
7. `scrape_group(self, grupo, html_content)`
    - Chama o método específico de raspagem para o grupo correspondente, utilizando um dicionário que mapeia grupos a seus métodos de raspagem. Registra um aviso no log se o grupo não tiver um método de raspagem definido.
8. `send_email(self)`
    - Envia um e-mail com os arquivos PDF e JSON em anexo, utilizando a classe `SendEmail`. Inclui um atraso opcional de 10 segundos antes do envio.

Fluxo de Execução

1. **Inicialização**: A classe é instanciada, carregando as variáveis de ambiente e inicializando as classes auxiliares.
2. **Execução do Fluxo Principal**: O método `run()` é chamado, iniciando a sequência de coleta de URLs, processamento de dados e, se necessário, geração de PDF e envio de e-mail.
3. **Coleta de URLs**: O método `scrape_urls()` busca e salva as URLs online, com validações para garantir que `BASE_URL` esteja disponível.
4. **Preparação e Processamento de Dados**: Os dados coletados são filtrados e processados, utilizando os métodos de scraping adequados para cada

grupo.

5. **Geração e Envio de Relatório**: Se configurado, um PDF é gerado a partir dos dados processados e um e-mail é enviado com os relatórios em anexo.

GetOnlineUrls

A classe `GetOnlineUrls` é responsável por extrair URLs de grupos online a partir de uma página base e salvar os resultados em um arquivo JSON. O fluxo envolve acessar o conteúdo da página, extrair links específicos de grupos marcados como online, e armazenar essas informações de forma organizada.

Atributos da Classe

- `base_url` (`str`): A URL base da qual serão extraídos os grupos e seus respectivos links online.
- `logger` (`logging.Logger`): Configurado para registrar logs de informações e erros ao longo do processo.

Métodos

1. `__init__(self, base_url)`
   - Inicializa a classe com a URL base e configura o logger.
   - Define o nível de logging como `INFO` e formata as mensagens de log.
2. `fetch_page_content(self, url)`
   - Faz uma requisição HTTP para obter o conteúdo HTML da página fornecida.
   - Utiliza `requests` para acessar o URL e retorna o conteúdo em caso de sucesso.
   - Se houver um erro na requisição, registra o erro no log e retorna `None`.
3. `generate_groups_dict(self)`
   - Gera um dicionário contendo os nomes dos grupos e seus URLs de acesso.
   - Faz uma requisição à `base_url` e utiliza BeautifulSoup para localizar as linhas de tabela (`<tr>`).

- Para cada linha, verifica se há um grupo marcado como "🟢" (online) e extrai seu nome e link.
- Retorna um dicionário com os grupos e seus respectivos links.

4. `extract_online_links(self, groups)`
   - Para cada grupo no dicionário gerado, acessa o link do grupo e extrai os links online específicos.
   - Faz uma nova requisição ao URL de cada grupo e usa BeautifulSoup para encontrar links associados ao grupo marcado como online.
   - Adiciona os links online ao dicionário de cada grupo.
   - Retorna o dicionário atualizado com os links online.
5. `save_to_json(self, data, file_path='groups.json')`
   - Salva o dicionário de grupos e seus links em um arquivo JSON.
   - Tenta abrir o arquivo especificado e grava os dados nele.
   - Em caso de erro ao salvar, registra uma mensagem de erro no log.
6. `get_online_urls_and_save(self)`
   - Método principal que executa o processo completo:
     1. Gera o dicionário de grupos.
     2. Extrai os links online para cada grupo.
     3. Salva os dados em um arquivo JSON.
   - Este método organiza todas as etapas em uma única chamada.

Fluxo de Execução

1. **Inicialização**: A classe é instanciada com a URL base carregada de um arquivo `.env`.
2. **Extração de Grupos**: A função `generate_groups_dict()` faz a requisição inicial à URL base e encontra grupos marcados como "online", extraindo seus links.
3. **Extração de Links**: A função `extract_online_links()` acessa a página de cada grupo e busca pelos links online associados.
4. **Armazenamento**: O dicionário resultante é salvo em um arquivo JSON usando o método `save_to_json()`.

GetHtml

A classe
`GetHtml` tem como objetivo principal realizar a extração de HTMLs de uma lista de URLs, extraindo datas relevantes dentro de um intervalo de dias e armazenando os resultados em um DataFrame do Pandas. Além disso, ela realiza logging de eventos importantes como sucessos e erros nas requisições HTTP.

Atributos da Classe

- **`json_file`** (`str`): Arquivo JSON contendo os dados dos grupos e URLs para scraping. Por padrão, usa `groups.json`.
- **`days`** (`int`): Número de dias de intervalo para filtrar as datas extraídas. Padrão é 7 dias.
- **`df`** (`pd.DataFrame`): DataFrame para armazenar os dados carregados do JSON e processados.

Métodos

1. **`__init__(self, json_file='groups.json', days=7)`**
   - Inicializa a classe com o arquivo JSON e o intervalo de dias.
   - Prepara um DataFrame vazio para armazenar os dados.
2. **`load_data(self)`**
   - Carrega os dados do arquivo JSON especificado.
   - Cria um DataFrame a partir do dicionário de dados e ajusta suas colunas.
   - Inicializa colunas adicionais (`response_status_code`, `html`, `infos`, `datas`, `qtd_datas`) para armazenar resultados das requisições e processamento.
3. **`extract_dates(html, days)`**
   - Método estático que extrai datas de um HTML fornecido usando expressões regulares.

- Filtra as datas para incluir apenas aquelas que estão dentro do intervalo de dias definido.
- Retorna uma lista de datas encontradas.

4. `fetch_data(self)`
   - Faz requisições HTTP para os links presentes no DataFrame.
   - Para cada URL, registra o status da requisição e, se bem-sucedida, salva o HTML e as datas extraídas.
   - Utiliza proxies SOCKS5 definidos pelas variáveis de ambiente para realizar as requisições.
   - Em caso de erro, registra no log e marca a requisição como falha.
5. `run(self)`
   - Método principal que executa o fluxo completo da classe:
     1. Carrega os dados do arquivo JSON.
     2. Realiza as requisições HTTP.
     3. Retorna o DataFrame final com os resultados do scraping e processamento

Fluxo de Execução

1. O método `run()` inicia o processo, carregando os dados do arquivo JSON.
2. Para cada URL no arquivo, a classe faz uma requisição HTTP e tenta obter o HTML.
3. As datas são extraídas do HTML usando expressões regulares e filtradas por um intervalo de dias.
4. Todas as informações são armazenadas no DataFrame para análise ou exportação posterior.

Grupos

A classe `Grupos` é responsável por processar conteúdos HTML ou JSON e extrair informações de grupos, filtrando-as com base em um intervalo de datas

definido. Ela é usada para processar dados de diferentes fontes e formatos, como JSON e HTML, e retorna os resultados filtrados de acordo com a data.

Atributos da Classe

- `days` (`int`): O número de dias anteriores a partir da data atual para definir o intervalo de filtragem.
- `hoje` (`datetime`): A data atual (momento em que a classe é instanciada).
- `limite` (`datetime`): A data limite inferior, que é a data atual menos o número de dias especificado.

Métodos da Classe

1. `__init__(self, days)`
   - Inicializa a classe com o número de dias para filtrar os dados.
   - Define o atributo `hoje` com a data atual e `limite` com a data resultante da subtração do número de dias informado.
2. `_parse_date(self, date_str, date_format='%Y-%m-%d')`
   - Método auxiliar que tenta converter uma string de data em um objeto `datetime` utilizando um formato de data especificado (por padrão, `%Y-%m-%d`).
   - Se a conversão falhar, retorna `None`.
3. `_is_date_within_range(self, date_obj)`
   - Método auxiliar que verifica se um objeto `datetime` está dentro do intervalo de datas entre `limite` e `hoje`.
   - Retorna `True` se a data estiver dentro do intervalo, e `False` caso contrário.
4. `ransomhouse(self, html_content)`
   - Processa um conteúdo JSON e extrai informações sobre o grupo `ransomhouse`, filtrando-as com base na data de ação (`actionDate`).
   - Para cada entrada no campo `data`, converte a data da ação no formato `%d/%m/%Y` e verifica se está no intervalo de dias definido.
   - Retorna uma lista filtrada de dicionários contendo o título, o site, e a data de cada grupo.

5. `monti(self, html_content)`
    - Processa o conteúdo HTML do grupo `monti` e busca por divs com a classe `col-auto published` para extrair datas de publicação.
    - Verifica se as datas de publicação estão dentro do intervalo e, se estiverem, extrai o título associado e adiciona aos resultados.
    - Retorna uma lista de dicionários com título, site (vazio nesse caso), e a data.
6. `play(self, html_content)`
    - Processa o conteúdo HTML da página do grupo `play` e busca por elementos `th` com a classe `News`.
    - Extrai o título, o link associado e a data de publicação (caso encontrada), verificando se a data está dentro do intervalo.
    - Retorna uma lista de dicionários com o título, o site, e a data da publicação.
7. `handala(self, html_content)`
    - Processa o conteúdo HTML do site do grupo `handala` e extrai informações de grupos dentro de itens de lista (`li`) com a classe `wp-block-post`.
    - Para cada item, extrai o título, a data da publicação (usando a tag `<time>`), e tenta extrair o site a partir do texto da descrição.
    - Verifica se a data está no intervalo, e adiciona os resultados à lista retornada.
8. `blackbyte(self, html)`
    - Processa o conteúdo HTML do site do grupo `blackbyte` e extrai informações de grupos a partir de uma tabela, buscando datas nos elementos `<td>`.
    - Verifica se as datas estão dentro do intervalo e, se estiverem, extrai o nome da empresa a partir do elemento `<caption>` da tabela.
    - Retorna uma lista de dicionários com o título (nome da empresa), o site (vazio nesse caso), e a data.

Fluxo de Execução

1. **Inicialização**: A classe é instanciada com o número de dias a ser utilizado para definir o intervalo de datas.
2. **Parsing de Datas**: Cada método que processa o conteúdo HTML ou JSON utiliza `_parse_date()` para converter as strings de data em objetos `datetime`, que são então verificadas pelo método `_is_date_within_range()`.
3. **Processamento por Fonte**: Cada método específico (`ransomhouse`, `monti`, `play`, etc.) processa o conteúdo HTML ou JSON para extrair os dados de interesse (título, link e data) e verificar se estão dentro do intervalo de datas.
4. **Filtragem e Retorno**: Os dados que passam pela verificação de intervalo são retornados em forma de lista de dicionários, contendo as informações filtradas.

DataFrameToMarkdownPDF

A classe `DataFrameToMarkdownPDF` tem como objetivo gerar um relatório em PDF a partir de dados estruturados em um DataFrame, exibindo-os em formato de tabela, agrupados e ordenados. O relatório inclui informações sobre grupos e links, além de uma descrição geral dos dados.

Atributos da Classe

- `pdf_filename` (`str`): O nome do arquivo PDF que será gerado. O valor padrão é `'report.pdf'`.

Métodos da Classe

1. `__init__(self, pdf_filename='report.pdf')`
   - Inicializa a classe com o nome do arquivo PDF a ser gerado.
   - Define o atributo `pdf_filename` com o nome do arquivo.
2. `_group_and_sort_data(self, data)`
   - Método auxiliar que agrupa os dados por grupo, ordena cada grupo por data (de forma decrescente), e depois ordena os grupos pelo número de itens que possuem (também de forma decrescente).

- Retorna uma lista de tuplas, onde cada tupla contém o nome do grupo e uma lista de itens ordenados por data.

3. `generate_pdf(self, data, links_df)`
   - Gera um arquivo PDF contendo uma descrição dos dados e uma tabela para cada grupo, com informações de título, site e data de cada item.
   - **Parâmetros**:
     - `data` (`list`): Lista de dicionários contendo os dados a serem incluídos no relatório. Cada dicionário deve ter as chaves `'grupo'`, `'title'`, `'site'` e `'date'`.
     - `links_df` (`DataFrame`): DataFrame contendo informações sobre os links online e seus status de resposta HTTP.

Detalhamento do Método `generate_pdf`

1. **Agrupamento e Ordenação**:
   - O método começa chamando o método auxiliar `_group_and_sort_data()` para agrupar e ordenar os dados conforme descrito. Os dados são organizados em grupos e, dentro de cada grupo, os itens são ordenados pela data de forma decrescente.
2. **Configuração do Documento**:
   - Um objeto `SimpleDocTemplate` é criado para definir o layout do documento PDF, utilizando o tamanho de página `A4`.
   - Uma lista `elements` é usada para armazenar os elementos (textos, tabelas, espaçadores) que serão incluídos no PDF.
3. **Estilos de Parágrafo**:
   - São definidos vários estilos de parágrafo utilizando `ParagraphStyle` para controlar o visual do PDF, como a formatação do título, a descrição, os grupos e os itens. Por exemplo:
     - `title_style`: Usado para o título do relatório.
     - `description_style`: Usado para as descrições de texto.
     - `group_style`: Usado para os cabeçalhos dos grupos.
     - `item_style`: Usado para os itens dentro de cada grupo.
4. **Inserção de Informações Descritivas**:

- O método insere uma descrição geral no início do PDF, incluindo o número total de links encontrados e a quantidade de requisições que obtiveram status HTTP 200.
- Essas informações são extraídas do DataFrame `links_df`, que contém os links online e os códigos de status HTTP.

5. **Tabelas de Dados por Grupo**:
   - Para cada grupo de dados, o método cria um cabeçalho com o nome do grupo e o número de itens encontrados.
   - Em seguida, cria uma tabela contendo as colunas "Title", "Site" e "Date", com os valores de cada item do grupo.
   - A tabela é estilizada com cores de fundo e grades, além de alinhamento centralizado e tamanhos de fonte ajustados.
6. **Criação do PDF**:
   - O método monta o PDF com os elementos adicionados (textos e tabelas) e o salva com o nome especificado no atributo `pdf_filename`.
   - Ao final do processo, uma mensagem de sucesso é registrada no log.

Fluxo de Execução

1. **Inicialização**: A classe é instanciada com o nome do arquivo PDF a ser gerado.
2. **Geração de PDF**:
   - O método `generate_pdf()` é chamado com os dados e um DataFrame de links.
   - Os dados são agrupados, ordenados e processados.
   - As tabelas e descrições são adicionadas ao documento PDF.
3. **Salvamento**: O PDF é gerado e salvo no caminho especificado.
4. **Logging**: Uma mensagem é exibida no log indicando o sucesso da operação.

SendEmail

A classe `SendEmail` tem como propósito enviar um email contendo dois arquivos anexados: um PDF e um JSON. Ela utiliza o protocolo SMTP (Simple Mail Transfer Protocol) para enviar emails de maneira segura através do servidor do Gmail. As informações de login, como o usuário e a senha, são carregadas de um arquivo `.env` por meio da biblioteca `dotenv`, garantindo que dados sensíveis sejam mantidos seguros.

Atributos da Classe

- `username` (`str`): O nome de usuário (endereço de email) que será usado para autenticar no servidor SMTP.
- `password` (`str`): A senha do email ou um token de aplicação para autenticação segura no servidor SMTP.
- `sender_email` (`str`): O endereço de email do remetente.
- `recipient_email` (`str`): O endereço de email do destinatário.

Os valores desses atributos são carregados a partir de variáveis de ambiente no arquivo `.env`.

Métodos da Classe

1. `__init__(self)`
   - Este método inicializa a classe e carrega as variáveis de ambiente necessárias usando `dotenv`. Ele define os atributos de email (`username`, `password`, `sender_email`, `recipient_email`) a partir das variáveis armazenadas no arquivo `.env`.
2. `send_email(self, pdf_file, json_file)`
   - Método principal que realiza o envio do email com os arquivos PDF e JSON anexados.
   - **Parâmetros**:
     - `pdf_file` (`str`): Caminho para o arquivo PDF que será anexado ao email.
     - `json_file` (`str`): Caminho para o arquivo JSON que será anexado ao email.

   Detalhamento do Método `send_email`:

   1. **Criação da Mensagem**:

- Um objeto `MIMEMultipart` é instanciado para criar a estrutura da mensagem, que permite o envio de texto e anexos.
- Os campos de cabeçalho da mensagem são preenchidos, como o remetente (`From`), o destinatário (`To`) e o assunto (`Subject`).
- O corpo do email é adicionado como texto simples, explicando que o PDF e o JSON estão anexados.

2. **Anexação dos Arquivos**:
   - O arquivo PDF é aberto em modo de leitura binária (`'rb'`) e anexado à mensagem como um `MIMEApplication` com o subtipo `'pdf'`. O cabeçalho de disposição do conteúdo é configurado para indicar que o arquivo é um anexo.
   - O arquivo JSON é processado de maneira semelhante, sendo anexado como um `MIMEApplication` com o subtipo `'json'`.
3. **Conexão com o Servidor SMTP**:
   - Um contexto SSL padrão é criado usando `ssl.create_default_context()` para garantir uma conexão segura com o servidor SMTP.
   - O método se conecta ao servidor SMTP do Gmail (`smtp.gmail.com`) na porta 587, que é usada para comunicações seguras via TLS.
   - O processo de autenticação é realizado utilizando o nome de usuário e a senha armazenados nas variáveis de ambiente.
   - A mensagem é enviada para o destinatário especificado.
4. **Tratamento de Exceções**:
   - Caso ocorra algum erro durante o processo de envio (por exemplo, se a autenticação falhar ou a conexão não for estabelecida), uma mensagem de erro será registrada no log, e a exceção será capturada e exibida.
5. **Logging**:
   - Se o email for enviado com sucesso, uma mensagem de sucesso será registrada no log. Em caso de falha, uma mensagem de erro será registrada com detalhes da exceção.