

Pedro Vinicius Almeida de Freitas  
Matrícula: 1921163

# **Projeto Final de Programação: Video Dataset Creator**

Rio de Janeiro, Brasil

2020

Pedro Vinicius Almeida de Freitas  
Matrícula: 1921163

## **Projeto Final de Programação: Video Dataset Creator**

Trabalho apresentado ao coordenador do programa de pós-graduação em informática da PUC-Rio como requisito para obtenção de nota na disciplina INF2102-Projeto Final de Programação

Pontifícia Universidade Católica do Rio de Janeiro  
Departamento de Informática  
Programa de Pós-Graduação em Informática

Orientador: Sérgio Colcher

Rio de Janeiro, Brasil  
2020

# Sumário

<b>Sumário</b>	<b>2</b>
<b>1 ESPECIFICAÇÃO DO PROGRAMA</b>	<b>3</b>
1.1 <b>Objetivo</b>	<b>3</b>
1.2 <b>Escopo</b>	<b>3</b>
1.3 <b>Requisitos</b>	<b>3</b>
1.3.1 Requisitos Funcionais	3
1.3.2 Requisitos Não-Funcionais	5
<b>2 ARQUITETURA</b>	<b>6</b>
2.0.1 Youtube Search	7
2.0.2 Youtube Downloader from CSV	7
2.0.3 RNP Crawler	9
<b>3 TESTES</b>	<b>11</b>
3.1 <b>Youtube Search</b>	<b>11</b>
3.2 <b>Youtube Downloader from CSV</b>	<b>11</b>
3.3 <b>RNP Crawler</b>	<b>11</b>
3.3.1 Resultados dos testes	12
<b>4 DOCUMENTAÇÃO PARA O USUÁRIO</b>	<b>13</b>
4.1 <b>Executando as ferramentas para coleta no Youtube</b>	<b>14</b>
4.2 <b>Executando a ferramenta para coleta na plataforma Video@RNP</b>	<b>15</b>
<b>5 CÓDIGO FONTE</b>	<b>18</b>

# 1 Especificação do Programa

## 1.1 Objetivo

Com a crescente quantidade de dados disponibilizados na internet, uma tarefa recorrente é a coleta automática de dados, sejam eles vídeos, imagens, ou texto. O objetivo deste projeto é desenvolver ferramentas para coleta automática (também conhecidos como *crawlers* ou *bots*) de vídeos em plataformas de hospedagem de vídeo. Essas ferramentas podem auxiliar na criação de datasets para aplicação de técnicas de machine learning, datasets para sistemas de recomendação, e análise exploratória de dados em projetos de data science.

## 1.2 Escopo

O escopo desse projeto de programação de final é o design, desenvolvimento e documentação de *crawlers*, com funções como continuar downloads caso interrompidos, e com capacidade para execução por longos períodos de tempo. Para isso, serão criadas três variações com fins específicos:

- Busca e coleta de vídeos a partir de termos de pesquisa.
- Coleta de vídeos a partir de links.
- Coleta de vídeos a partir de uma API.

Serão suportadas duas plataformas de hospedagem de vídeos: Youtube<sup>1</sup> e Video@RNP<sup>2</sup>. Esses *crawlers* também terão interação através de interface em linhas de comando, dadas suas funções pontuais e execução majoritariamente em servidores *headless* (sem interface gráfica).

## 1.3 Requisitos

### 1.3.1 Requisitos Funcionais

Para determinar os Requisitos Funcionais (RF) desse projeto foram identificados três casos de uso:

---

<sup>1</sup> <https://www.youtube.com/>

<sup>2</sup> <http://www.videoaula.rnp.br/portal/home>

- **Caso-01 Busca e coleta de vídeos no Youtube a partir de termos de pesquisa:** Existem casos em que há necessidade de coleta automática de vídeos sobre determinado assunto, para isso se pode usar o próprio motor de busca do Youtube. A ferramenta deve coletar automaticamente vídeos a partir dos resultados dessa busca. Esses vídeos são salvos em suas respectivas pastas de acordo com o termo de busca para futura anotação e verificação manual.
- **Caso-02 Coleta de vídeos do Youtube a partir de links:** Nesse caso de uso o usuário fez a seleção de quais vídeos baixar em uma ferramenta externa, por exemplo, obtenção de URLs (Uniform Resource Locator, ou "localizador uniforme de recursos") a partir do dataset Youtube8M, que é um conjunto de dados composto de cerca de 8 milhões de vídeos manualmente anotados. Então a ferramenta teria como entrada um arquivo com varias URLs, primeiro há de ocorrer a verificação de quais vídeos já foram baixados com base em seus ids, então, iniciar o download automatico dessa lista de vídeos a partir da lista de vídeos ainda não baixados.
- **Caso-03 Coleta de vídeos a partir da API Video@RNP:** Existe também a possibilidade de acesso a API (Application Programming Interface ou em português, Interface de Programação de Aplicação) de uma plataforma de hospedagem de video, especificamente o acesso a API da plataforma Video@RNP, da Rede Nacional de Pesquisa. Essa plataforma hospeda majoritariamente vídeos educativos em português. A ferramenta também precisa checar quais vídeos já foram baixados e continuar a coleta de onde parou caso interrompido.

Com esses casos de uso em mente, foram definidos quatro Requisitos Funcionais:

- **RF-01 Busca por videos:** Uma das ferramentas deve ser capaz de buscar videos na plataforma Youtube.
- **RF-02 Coleta a partir de arquivos contendo URLs:** Uma das ferramentas deve ter a função de baixar videos a partir de um arquivo CSV (Comma Separated Values).
- **RF-03 Coleta sequencial de vídeos a partir de uma API:** Uma das ferramentas deve ter a capacidade de descobrir e solicitar qualquer numero de videos em uma API.
- **RF-04 Download e armazenamento de um video a partir de uma URL ou link:** Todas a ferramentas devem ter a capacidade de baixar quaisquer videos encontrados durante sua execução (durante o *crawling*).

Requisitos como RF-01, RF-02 e RF-3 devem ser atendidos por pelo menos uma das ferramentas, enquanto o requisito RF-04 deve ser atendido por todas.

### 1.3.2 Requisitos Não-Funcionais

Os Requisitos Não Funcionais (RNF) desse projeto foram identificados como:

- **RNF-01 Usabilidade:** As ferramentas precisam ter ajuda para um usuário, com uma função de *help* na linha de comando, que descreve os argumentos e utilização da ferramenta.
- **RNF-02 Recuperação de falhas:** As ferramentas precisam ter capacidade de continuar sua execução caso interrompidas, dada sua natureza de longa execução.
- **RNF-03 Estabilidade.** As ferramentas devem ser capazes de executar por longos períodos de tempo sem erros inerentes da ferramenta, sem acúmulo de uso de memória, e devem tratar erros de conexão de rede sem interromper a fila de downloads.
- **RNF-04 Padrões.** Conformidade com os padrões e normas a serem seguidas no desenvolvimento do sistema.

## 2 Arquitetura

Cada ferramenta foi implementada utilizando a linguagem Python (Versão 3.6.12), utilizando o gerenciador de pacotes Pipenv (versão 2020.11.15), que torna a instalação das ferramentas e dependências mais conveniente.

O projeto contém dois diretórios: *docs*, que contém os arquivos para geração da documentação do projeto, e *crawlers*, esta, por sua vez é dividida em três módulos:

- *rnp*: Contém o crawler focado na plataforma Video@RNP.
- *tests*: Contém os testes unitários automatizados para todos os crawlers.
- *youtube*: Contém os crawlers para download de videos na plataforma youtube a partir de busca por termo ou a partir de arquivos com URLs.

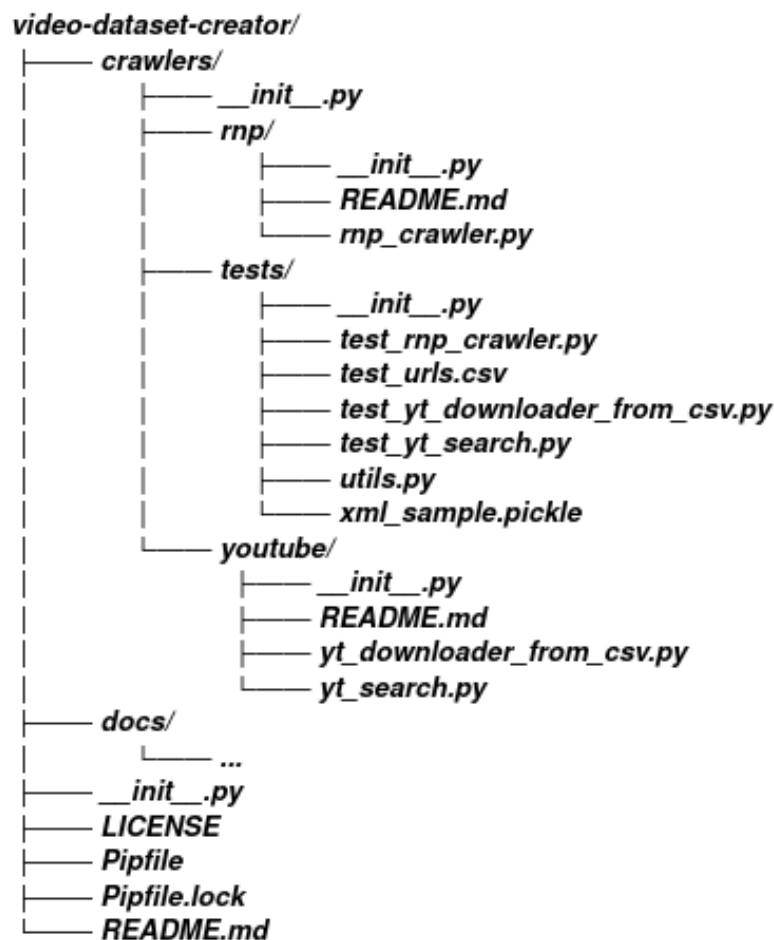


Figura 1 – Arvore de arquivos do projeto.

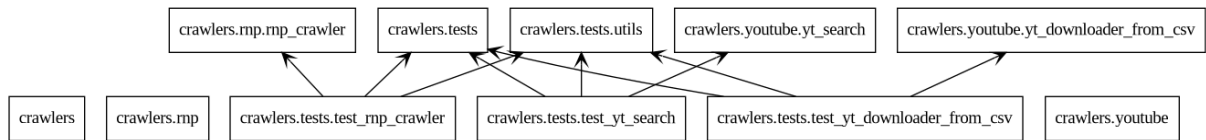


Figura 2 – Diagrama de dependência entre os pacotes.

### 2.0.1 Youtube Search

```
$ python yt_search.py Dança ~/ --number 5 --wait 5
```

A ferramenta *Youtube Search* depende principalmente da biblioteca *youtube-dl*<sup>1</sup> para fazer as requisições e transferência de vídeos do youtube, pois lida com multiplos agentes de navegador e com extração de informação de conteúdo por vezes protegido.

Para se defender de ataques Denial of Service (DoS), muitos sites modernos implementam técnicas como limitar o número requisições por ip, por usuário, por serviço e por geolocalização.

Para evitar que as requisições sejam recusadas, é necessário respeitar o limite de requisições por minuto de cada plataforma de hospedagem. Com isso em mente, é definido um tempo padrão, que pode ser modificado, entre requisições de download em todas as ferramentas implementadas.

A organização da ferramenta *Youtube Search* consiste basicamente em uma função principal que testa a existência da pasta de destino, que em caso negativo cria e nomeia automaticamente a pasta de destino, e processa os argumentos a serem passados para a biblioteca *youtube-dl*. Primeiro o termo de busca é pesquisado no youtube, a biblioteca *youtube-dl* retorna uma playlist com o número de vídeos requisitados resultantes da pesquisa, que então entram em uma fila para download. A ferramenta também evita baixar vídeos que já estão na pasta de destino através de comparação do id do vídeo com o id a URL fornecida.

A Figura 3 ilustra o processo de comunicação entre a ferramenta e os servidores do Youtube.

### 2.0.2 Youtube Downloader from CSV

A ferramenta *Youtube Downloader from CSV* também depende da biblioteca *youtube-dl*<sup>2</sup> para fazer as requisições e transferência de vídeos do youtube, principalmente para resolução de urls e extração de dados para o download do vídeo.

A organização da ferramenta *Youtube Downloader from CSV* consiste em três funções principais. A função *read\_csv\_and\_download\_videos* lê o arquivo CSV, separa

<sup>1</sup> [https://pypi.org/project/youtube\\_dl/](https://pypi.org/project/youtube_dl/)

<sup>2</sup> [https://pypi.org/project/youtube\\_dl/](https://pypi.org/project/youtube_dl/)



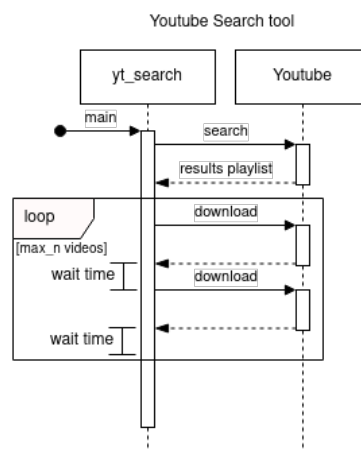


Figura 3 – Diagrama de sequência entre a ferramenta Youtube Search e os servidores do Youtube.

as URLs, testa a existência da pasta de destino, que em caso negativo cria e nomeia automaticamente a pasta de destino, e processa os argumentos a serem passados para a biblioteca *youtube-dl*. A função *already\_downloaded* é chamada para cada URL no arquivo CSV de entrada e retorna se o vídeo proveniente dessa URL já foi baixado ou não. Por fim, a função *download*, cuida da chamada para a função de transcrição da biblioteca *youtube-dl*. Uma vez determinadas quais vídeos ainda não foram baixados, uma fila de *download* é executada para coletar os vídeos remanescentes.

A Figura 4 ilustra o processo de comunicação entre a ferramenta e os servidores do Youtube.

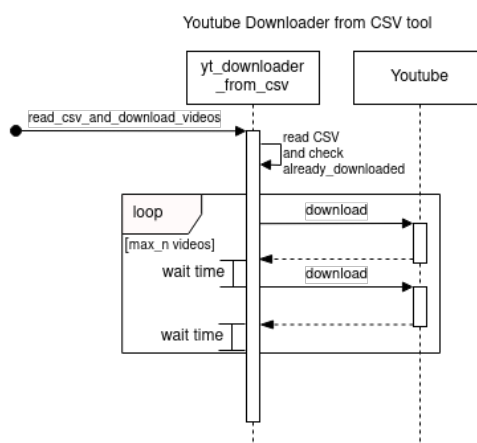


Figura 4 – Diagrama de sequência entre a ferramenta Youtube Downloader from CSV e os servidores do Youtube.

A ferramenta também evita baixar vídeos que já estão na pasta de destino através de comparação do id do vídeo com o id a URL fornecida.

### 2.0.3 RNP Crawler

Para executar a ferramenta para coleta de vídeos através da API da plataforma Video@RNP é necessário que o usuário primeiro tenha uma chave. Detalhes sobre a como obter a chave de cliente são fornecidos na Seção 4.2.

A ferramenta *RNP Crawler* é composta por uma função principal *crawl\_and\_download*, que faz requisições à API do site Video@RNP, primeiro requisitando uma lista de vídeos na plataforma de acordo com o tamanho determinado em um dos argumentos. Em seguida, para cada vídeo na lista, requisita dados de quais as versões disponíveis na plataforma e seleciona a versão com maior qualidade. Por fim, para cada URL selecionada usa a função *download\_file* para baixar o vídeo selecionado. Nesse caso, a biblioteca youtube-dl não foi necessária pois as limitações de download e acesso no Youtube não acontecem nessa plataforma.

Essa ferramenta conta com outras funções de utilidade como uma função de *logging* para arquivos, formatação de tamanho de arquivos e *debugging*. Todas essas funções são detalhadas na SubSeção 3.3.

Assim como as outras ferramentas, *RNP Crawler* também faz intervalos de 50 segundos entre as requisições para download (Tal tempo é definido pela plataforma). Esse tempo de espera entre os downloads serve com um meio de evitar que o algoritmo ultrapasse o limite de requisições por minuto da plataforma, o que causaria um bloqueio temporário das requisições e impossibilitaria o download contínuo de vídeos.

A Figura 5 ilustra o processo de comunicação via API entre a ferramenta e os servidores da plataforma Video@RNP.

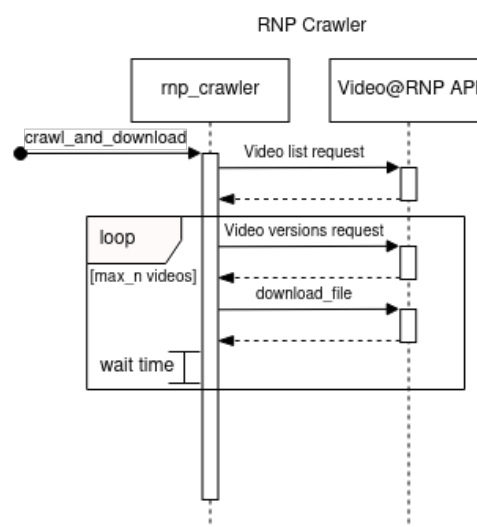


Figura 5 – Diagrama de sequência entre a ferramenta RNP Crawler e os servidores Video@RNP.

A ferramenta também cria uma pasta de destino caso esta não exista e lá armazenará

os vídeos baixados. Ela também evita automaticamente coletar vídeos que já foram baixados. Essa ferramenta foi feita com a intenção de baixar todos os vídeos da plataforma. Por esse motivo precisa ser capaz de rodar por longos períodos de tempo sem cessar.

## 3 Testes

Com o objetivo de verificar as funcionalidades do sistema, foram feitos testes unitários para cada um dos crawlers. Esses testes estão descritos respectivamente nas Seções 3.1, 3.2 e 3.3.

### 3.1 Youtube Search

O crawler para buscas depende do motor de buscas do próprio youtube, para assegurar o funcionamento, um teste de busca e download para um termo de teste *sudoku* foi implementado.

*test\_search*. É testada funcionalidade de busca por vídeos com base em um termo. É testada a capacidade de buscar e baixar vídeos com base em um termo de busca.

### 3.2 Youtube Downloader from CSV

A ferramenta de download de vídeos a partir de urls (links) precisa ler um arquivo csv, processar as urls nele contidas, verificar quais vídeos já foram baixados e baixar os vídeos restantes.

- *test\_download*. Testa a capacidade da ferramenta de fazer downloads a partir de um conjunto de urls conhecidas.
- *test\_already\_downloaded*. Nesse teste, que deve ser feito após o teste de dowload, é testada a capacidade da ferramenta de checar quais das url já foram processadas e assim continuar a baixar os vídeos de um csv.
- *test\_read\_csv\_and\_download\_videos*. Nesse teste todo os processos de ler um arquivo csv, processar as urls e baixar os vídeos é verificado, ou seja, é verificada a funcionalidade global da ferramenta.

### 3.3 RNP Crawler

A ferramenta de download de vídeos da plataforma *Video@RNP* deve ser capaz de buscar e baixar vídeos disponíveis, além de verificar quais desses vídeos já foram baixados. Além disso, foram desenvolvidas funcionalidades de logging em arquivo, debugging de respostas xml e formatação de unidades de tamanho de arquivos.

- *test\_log* Esse teste verifica a funcionalidade de logging, com e sem saída para arquivo.
- *test\_scandown*. Nesse teste, é verificada a capacidade da função de debugging de respostas xml, essa capacidade não é utilizada normalmente, mas é bastante relevante para debugging da ferramenta.
- *test\_sizeof\_fmt*. Nesse teste, é verificada a função de formatação de tamanho de arquivos, pois o tamanho de cada arquivo é fornecida somente em bytes.
- *test\_download\_file*. Nesse teste, é verificada a função de download de arquivos a partir da base da plataforma *Video@RNP*.
- *test\_crawl\_and\_download*. Nesse teste, a soma de todas as capacidades da ferramenta: buscar, verificar e baixar vídeos na plataforma *Video@RNP*, são testadas.

### 3.3.1 Resultados dos testes

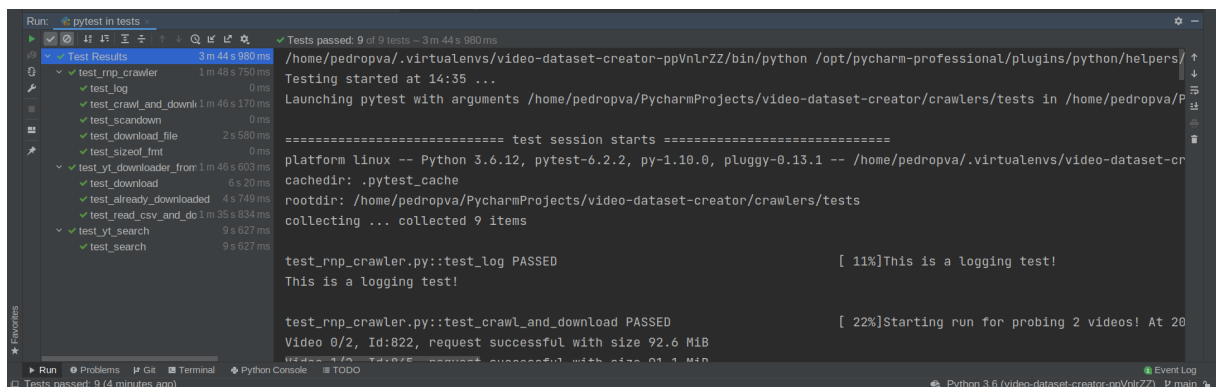


Figura 6 – Logs da execução dos testes.

Todos os testes foram executados com sucesso. Todos os crawlers executaram suas funções definidas durante os testes. A saída da plataforma de teste pode ser observada na Figura 6.

## 4 Documentação para o Usuário

Essas orientações para instalação serão majoritariamente focadas em um usuário utilizando uma distribuição Linux como sistema operacional, dado que a grande parte dos servidores, onde se é visada a execução dessas ferramentas, utilizam distribuições Linux como sistema operacional. Não obstante, essas ferramentas ainda podem ser utilizadas em qualquer sistema operacional que suporte a execução de scripts *Python*. Primeiro, o usuário deve fazer o download do projeto, que pode ser feito pelo site GitHub.<sup>1</sup> Este pode ser feito como .zip diretamente pelo site ou através do comando:

```
$ git clone https://github.com/pedropva/video-dataset-creator.git
```

Em seguida, é necessário verificar a existência de uma versão *Python* compatível (A sua versão *Python* precisa ser 3.6.12 ou mais recente):

```
$ python --version
Python 3.6.12
```

Caso não tenha uma versão *Python* mais recente que 3.6.12, o usuário pode usar o pacote *pyenv*<sup>2</sup> para instalar a versão *Python* desejada, sem interferir com a versão Python do sistema.

Para fazer a instalação das ferramentas e das suas dependências basta usar o comando *Pipenv*<sup>3</sup>.

```
$ pipenv install
```

Você também pode especificar o caminho para uma versão do python com *--python*.

```
$ pipenv install --python ~/.pyenv/versions/3.6.12/bin/python
```

Uma vez instalados o ambiente e dependências, basta utilizar o comando *pipenv run* para executar as ferramentas. Também é possível criar uma instancia de um terminal com esse ambiente com *pipenv run*, todos os comandos *Python* executados dentro desse terminal implicação no uso do interpretador *Python* selecionado, juntamente com as dependencias instaladas.

---

<sup>1</sup> <https://github.com/pedropva/video-dataset-creator>

<sup>2</sup> <https://github.com/pyenv/pyenv>

<sup>3</sup> <https://pypi.org/project/pipenv/>

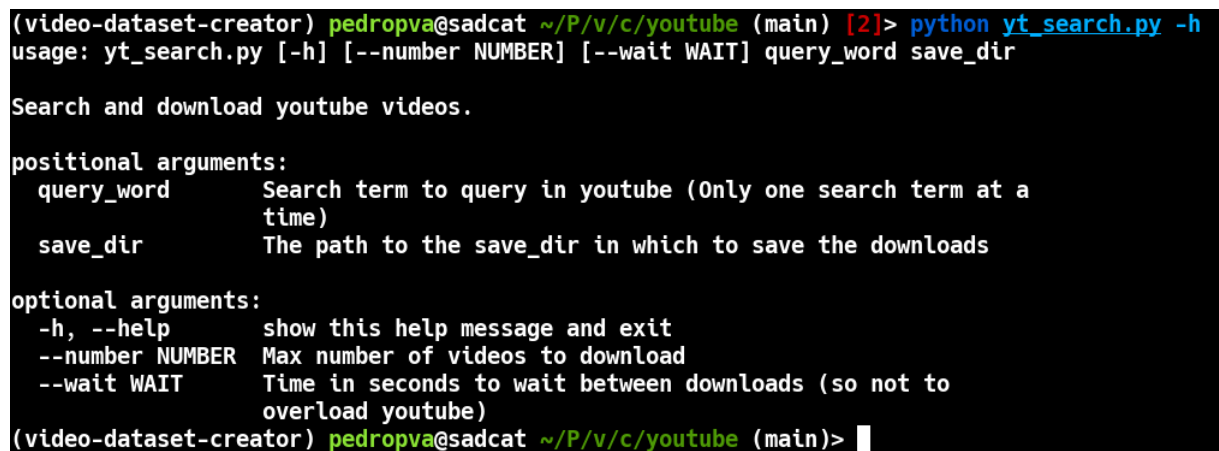
## 4.1 Executando as ferramentas para coleta no Youtube

Para executar a ferramenta para busca e coleta de vídeos, basta usar o seguinte comando enquanto dentro de um terminal *pipenv* (*pipenv shell*) ou utilizando o prefixo *pipenv run*:

```
$ python yt_search.py Dança ~/ --number 5 --wait 5
```

Com esse comando a ferramenta buscará pela palavra "Dança" e coletará 5 vídeos, com intervalo de tempo entre 5 segundos entre as requisições para download. Esse tempo de espera entre os downloads serve com um meio de evitar que o algoritmo ultrapasse o limite de requisições por minuto do Youtube, o que causaria um bloqueio temporário das requisições e impossibilitaria o download contínuo de vídeos. A ferramenta também criará uma pasta chamada *Dança\_videos/* na pasta *home* do usuário (Pois a pasta de destino especificada foi *" /"*) e lá armazenará os vídeos baixados.

Há uma opção para ajuda, caso o usuário tenha alguma dúvida sobre quais argumentos usar, como mostra a Figura 7



```
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main) [21]> python yt_search.py -h
usage: yt_search.py [-h] [--number NUMBER] [--wait WAIT] query_word save_dir

Search and download youtube videos.

positional arguments:
  query_word          Search term to query in youtube (Only one search term at a
                      time)
  save_dir            The path to the save_dir in which to save the downloads

optional arguments:
  -h, --help          show this help message and exit
  --number NUMBER     Max number of videos to download
  --wait WAIT         Time in seconds to wait between downloads (so not to
                      overload youtube)
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main)>
```

Figura 7 – Página de ajuda da ferramenta Youtube Search.

Em casos de termos de busca compostos por mais de uma palavra é necessário que o usuário utilize aspas duplas para demarcar o termo. Em termos de busca simples (com apenas uma palavra), as aspas não são necessárias. A Figura 8 mostra um exemplo de execução da ferramenta, buscando por um termo de busca composto ("Sphinx cat").

O uso da ferramenta *Youtube Downloader from CSV* é similar, basta usar o seguinte comando enquanto dentro de um terminal *pipenv* (*pipenv shell*) ou utilizando o prefixo *pipenv run*:

```
$ python yt_downloader_from_csv.py ../tests/test_urls.csv ~/yt_csv_downloads/
```

```
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main) [1]> python yt_search.py "Sphinx cat" /home/pedropva/Desktop/ --number 5
Downloading 5 videos of Sphinx cat videos!
[download] Downloading playlist: Sphinx cat
[youtube:search] query "Sphinx cat": Downloading page 1
[youtube:search] playlist Sphinx cat: Downloading 5 videos
[download] Downloading video 1 of 5
[youtube] _Na_DUc6iLM: Downloading webpage
_Na_DUc6iLM
[download] Sleeping 10 seconds...
[download] Destination: /home/pedropva/Desktop//Sphinx cat_videos/_Na_DUc6iLM.mp4
[download] 100% of 22.71MiB in 00:05
[download] Downloading video 2 of 5
[youtube] IAUz5bGKzic: Downloading webpage
IAUz5bGKzic
[download] Sleeping 10 seconds...
[download] Destination: /home/pedropva/Desktop//Sphinx cat_videos/IAUz5bGKzic.mp4
[download] 100% of 39.63MiB in 00:08
[download] Downloading video 3 of 5
[youtube] umzjMX80M1Q: Downloading webpage
umzjMX80M1Q
[download] Sleeping 10 seconds...
[download] Destination: /home/pedropva/Desktop//Sphinx cat_videos/umzjMX80M1Q.mp4
[download] 100% of 38.29MiB in 00:10
[download] Downloading video 4 of 5
[youtube] VJYombwohAM: Downloading webpage
VJYombwohAM
[download] Sleeping 10 seconds...
[download] Destination: /home/pedropva/Desktop//Sphinx cat_videos/VJYombwohAM.mp4
[download] 100% of 38.17MiB in 00:07
[download] Downloading video 5 of 5
[youtube] hrbMJ2Uj6_A: Downloading webpage
hrbMJ2Uj6_A
[download] Sleeping 10 seconds...
[download] Destination: /home/pedropva/Desktop//Sphinx cat_videos/hrbMJ2Uj6_A.mp4
[download] 100% of 33.16MiB in 00:07
[download] Finished downloading playlist: Sphinx cat
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main)> █
```

Figura 8 – Exemplo de execução da ferramenta para busca e coleta de vídeos no Youtube.

Com esse comando a ferramenta irá baixar todos os vídeos no arquivo *test\_urls.csv*, que vem com as ferramentas, como parte do módulo de testes. Além disso, se a pasta *yt\_csv\_downloads/* ainda não existir na pasta *home* do usuário, a ferramenta a criará e salvará os vídeos coletados nela.

Há uma opção para ajuda e consulta, como mostra a Figura 9

```
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main) [2]> python yt_downloader_from_csv.py -h
usage: yt_downloader_from_csv.py [-h] [--wait WAIT] csv_path save_dir

positional arguments:
  csv_path      Path to the csv file containing a youtube url per line
  save_dir      The path to the save_dir in which to save the downloads

optional arguments:
  -h, --help    show this help message and exit
  --wait WAIT   Time in seconds to wait between downloads (so not to overload
               youtube)
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main)> █
```

Figura 9 – Página de ajuda da ferramenta Youtube Downloader from CSV.

A ferramenta também tentará fazer download de quaisquer vídeos na lista que ainda não tenham sido baixados, como mostra a Figura 10

## 4.2 Executando a ferramenta para coleta na plataforma Video@RNP

Para executar a ferramenta para coleta de vídeos através da API da plataforma Video@RNP é necessário que o usuário primeiro tenha uma chave. Para obter uma chave é



```
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main)> python yt_downloader_from_csv.py ../tests/test_urls.csv ~/Desktop/yt_csv_downloads/
2 Videos already downloaded!
Downloading from url: https://www.youtube.com/watch?v=t3nx8axVx1k
[youtube] t3nx8axVx1k: Downloading webpage
[download] Destination: /home/pedropva/Desktop/yt_csv_downloads/t3nx8axVx1k.mp4
[download] 100% of 36.63MiB in 00:05
downloaded: /home/pedropva/Desktop/yt_csv_downloads/t3nx8axVx1k.mp4
There are 3 out of 3 videos downloaded!
(video-dataset-creator) pedropva@sadcat ~/P/v/c/youtube (main)>
```

Figura 10 – Exemplo de execução da ferramenta para coleta de vídeos no Youtube a partir de um arquivo CSV.

necessário entrar em contato através da página de contato do [site](#).<sup>4</sup>

Uma vez de posse da chave de cliente para acesso à API, basta usar o seguinte comando enquanto dentro de um terminal *pipenv* (*pipenv shell*) ou utilizando o prefixo *pipenv run*:

```
$ python rnp_crawler.py ~/rnp_videos/ --key $CLIENT_KEY --limit 3
```

Com esse comando a ferramenta coletará 3 vídeos, com intervalo de tempo entre 50 segundos entre as requisições para download (Tal tempo é imposto pela plataforma). Esse tempo de espera entre os downloads serve com um meio de evitar que o algoritmo ultrapasse o limite de requisições por minuto da plataforma, o que causaria um bloqueio temporário das requisições e impossibilitaria o download contínuo de vídeos. A ferramenta também criará uma pasta chamada *rnp\_videos/* (Caso não exista) no diretório *home* do usuário e lá armazenará os vídeos baixados.

Os seguintes argumentos podem ser usados para ajustar fatores como por qual vídeo começar a coleta, se e onde salvar a saída e a quantidade de vídeos baixados:

- **key** (obrigatório): É a chave de acesso à API. Deve ser fornecida pelo suporte da plataforma.
- **save\_dir** (obrigatório): O caminho para a pasta onde salvar os vídeos coletados, se não existir a ferramenta criará a pasta nesse caminho.
- **limit** (opcional): Define o número de vídeos a serem coletados.
- **start\_id** (opcional): Define o id do vídeo pelo qual começar, caso o usuário deseje vídeos específicos.
- **start\_index** (opcional): Define de que ponto começar a coleta, para casos onde a ferramenta parou e precisa iniciar de certo ponto.
- **log\_path** (opcional): Se fornecido, a ferramenta criará um arquivo chamado "probing.log" onde escreverá a saída padrão e outros dados de sua execução.

<sup>4</sup> <http://www.videoaula.rnp.br/portal/contact-render.action>

Há uma opção para ajuda e consulta de argumentos, como mostra a Figura 11

```
(video-dataset-creator) pedropva@sadcat ~/P/v/c/rnp (main) [2]> python rnp_crawler.py --help
usage: rnp_crawler.py [-h] [--key KEY] [--start_id START_ID] [--limit LIMIT]
                    [--start_index START_INDEX] [--log_path LOG_PATH]
                    save_dir

positional arguments:
  save_dir              The path to the save_dir in which to save the
                        downloads

optional arguments:
  -h, --help            show this help message and exit
  --key KEY             Your Video@RNP API access key. You can also provide it
                        by putting it in the start of this script.
  --start_id START_ID  The Downloader will start the downloads from this
                        video id
  --limit LIMIT         Limit of videos to download
  --start_index START_INDEX
                        The Downloader will start the downloads from a said
                        number of videos
  --log_path LOG_PATH  Path to save the logs. e.g. './'
(video-dataset-creator) pedropva@sadcat ~/P/v/c/rnp (main)> █
```

Figura 11 – Página de ajuda da ferramenta RNP Crawler.

A ferramenta foi feita com a intenção de baixar todos os vídeos da plataforma. Ela também evita automaticamente coletar vídeos que já foram baixados. A Figura 12 mostra um exemplo de execução da ferramenta, coletando 3 vídeos, sendo que dois deles já estavam baixados.

```
(video-dataset-creator) pedropva@sadcat ~/P/v/c/rnp (main)> python rnp_crawler.py ~/Desktop/rnp_videos/ --key $CLIENT_KEY --limit 3
Starting run for probing 3 videos! At 2021-02-06 16:56:49.809014
Already downloaded, skipping!
Video 0/3, Id:822, request successful with size 92.6 MiB
Already downloaded, skipping!
Video 1/3, Id:845, request successful with size 91.1 MiB
Video 2/3, Id:95681, request successful with size 141.6 MiB
Total size: 341149890
Total size: 325.3 MiB
Number of successful requests:3
Number of denied requests:0
Number of failed requests:0
```

Figura 12 – Exemplo de execução da ferramenta para coleta de vídeos da plataforma Video@RNP.

## 5 Código Fonte

Todo o código fonte deste projeto de programação final está disponível no GitHub.<sup>1</sup> Além disso, foi gerada uma documentação de todo o código fonte, presente no fim deste documento.

---

<sup>1</sup> <https://github.com/pedropva/video-dataset-creator>

---

# **Video Dataset Creator**

**Pedro Vinicius Almeida de Freitas**

**Feb 06, 2021**



**CONTENTS:**

<b>1</b>	<b>crawlers</b>	<b>1</b>
1.1	crawlers package . . . . .	1
<b>2</b>	<b>Indices and tables</b>	<b>7</b>
	<b>Python Module Index</b>	<b>9</b>
	<b>Index</b>	<b>11</b>



## CRAWLERS

### 1.1 crawlers package

#### 1.1.1 Subpackages

##### crawlers.rnp package

##### Submodules

##### crawlers.rnp.rnp\_crawler module

VideoAtRNP Crawler

Author: Pedro Vinicius Almeida de Freitas

Created in: 10/01/2021

This tool uses the VideoAtRNP API to search for all videos in the platform and download them. The user can select any number of videos to download from the platform. The VideoAtRNP site is mainly a educational videos hosting platform.

This tool requires *requests* to be installed within the Python environment you are running this tool in.

This file can also be imported as a module and contains the following functions:

- `sizeof_fmt` - Formats number of bytes to a human readable string.
- `scandown` - Scan and print a xml tree.
- `log` - Rudimentary logging function.
- `download_file` - Downloads a file from the provided url.
- `crawl_and_download` - Crawls the VideoAtRNP API, collecting and downloading video data.

```
crawlers.rnp.rnp_crawler.crawl_and_download(client_key: str, save_dir: str, start_id: Optional[int] = None, start_index: int = 0, max_n: int = 10, log_file_path=None)
```

Crawls the [Video@RNP](#) API, collecting and downloading video data.

##### Parameters

- **client\_key** (*str*) – The client key provided by the API Admin ([Video@RNP](#)).
- **save\_dir** (*str*) – Path to where the file will be saved.
- **start\_id** (*str, optional*) – The Downloader will start the downloads from this video id.



- **start\_index** (*str, optional*) – The Downloader will start the downloads from a said number of videos
- **max\_n** (*int, optional*) – Max number of videos to download.
- **log\_file\_path** (*int, optional*) – Path to save the logs. Optional. e.g. “.”

**Returns** None, It automatically saves the videos to the save\_dir.

**Return type** None

`crawlers.rnp.rnp_crawler.download_file` (*url: str, save\_dir: str, local\_filename: Optional[str] = None, verbose: bool = True*)

Downloads a file from the provided url.

**Parameters**

- **url** (*str, optional*) – Url for the file.
- **save\_dir** (*str, optional*) – Path to where the file will be saved.
- **local\_filename** (*str, optional*) – Local equivalent to the file referenced in the url, the function will try to redownload the file if it has less than 150 bytes.
- **verbose** (*bool, optional*) – Boolean to toggle whether or not to print additional content.

**Returns** The size in bytes of the downloaded content, if it failed, size will be zero.

**Return type** int

`crawlers.rnp.rnp_crawler.log` (*string: str, file=None*)  
Rudimentary logging function.

**Parameters**

- **string** (*str, optional*) – What string to log.
- **file** – Reference to a open file where to write.

**Returns** It just prints or writes the string to a file.

`crawlers.rnp.rnp_crawler.scandown` (*elements, indent=0*)  
Scan and print a xml tree.

**Parameters**

- **elements** – a Xml.dom node or a list of nodes.
- **indent** (*int, optional*) – How much to indent in each branch.

**Returns** It just prints all nodes.

`crawlers.rnp.rnp_crawler.sizeof_fmt` (*n\_bytes: int, suffix: str = 'B'*)  
Formats number of bytes to a human readable string. Function by: Fred Cirera and Wai Ha Lee Sources: <https://stackoverflow.com/questions/1094841/get-human-readable-version-of-file-size> [https://web.archive.org/web/20111010015624/http://blogmag.net/blog/read/38/Print\\_human\\_readable\\_file\\_size](https://web.archive.org/web/20111010015624/http://blogmag.net/blog/read/38/Print_human_readable_file_size)

**Parameters**

- **n\_bytes** (*int, optional*) – Number of bytes to format.
- **suffix** (*str, optional*) – Suffix to put after each Unit (e.g. B: KiB, MiB, ...)

**Returns** A string with human readable representation of a set amount of bits.

**Return type** str

## Module contents

### crawlers.youtube package

#### Submodules

#### crawlers.youtube.yt\_downloader\_from\_csv module

Youtube downloader from csv

Author: Pedro Vinicius Almeida de Freitas

Created in: 05/01/2021

This tool takes a Comma Separated Values file with youtube videos URLs as input and downloads them to the specified directory. It avoids downloading videos that are already in the destination folder.

This tool requires *youtube\_dl* to be installed within the Python environment you are running this tool in.

This file can also be imported as a module and contains the following functions:

- `already_downloaded` - Tests if the video was already downloaded.
- `download` - Downloads a video from a URL.
- `read_csv_and_download_videos` - Collects URLs from a csv file and downloads them.

`crawlers.youtube.yt_downloader_from_csv.already_downloaded(url: str, save_dir: str)`

Checks (from the url) if a video was already downloaded in the save\_dir.

##### Parameters

- `url (str, optional)` – Youtube Url for the video.
- `save_dir (str, optional)` – Path to where the videos are saved.

**Returns** A boolean, True if the video already exists in the save\_dir, False otherwise.

**Return type** bool

`crawlers.youtube.yt_downloader_from_csv.download(url: str, save_dir: str)`

Downloads a video from the provided url.

##### Parameters

- `url (str, optional)` – Youtube Url for the video.
- `save_dir (str, optional)` – Path to where the videos are saved.

**Returns** A boolean, True if it had success downloading the video, False otherwise.

**Return type** bool

`crawlers.youtube.yt_downloader_from_csv.read_csv_and_download_videos(csv_path: str, save_dir: str, wait_time: int = 30)`

Downloads a video from a csv containing youtube urls. One url per line.

##### Parameters

- `csv_path (str, optional)` – Path to the csv file with the urls to youtube.

- **save\_dir** (*str*, *optional*) – Path to where the videos are saved.
- **wait\_time** (*int*, *optional*) – Wait time between requests, use it to not get blocked for too many requests.

**Returns** A boolean, True if the it had success downloading the video, False otherwise.

**Return type** bool

### crawlers.youtube.yt\_search module

Youtube Search

Author: Pedro Vinicius Almeida de Freitas

Created in: 09/01/2021

This tool takes a string as input and uses the google search engine to look for videos related to the input. Additionally it also downloads the videos in the search results.

This tool requires *youtube\_dl* to be installed within the Python environment you are running this tool in.

This file can also be imported as a module and contains the following functions:

- **search** - Performs a query in youtube then downloads every video to the save *save\_dir*.
- **main** - The main function of the script.

`crawlers.youtube.yt_search.main` (*query\_word: str*, *save\_dir: str*, *max\_n*, *wait\_time: int = 10*)

Performs a query in youtube then downloads every video to the save *save\_dir*.

#### Parameters

- **query\_word** – The query string, can only be a single string (e.g. “beach”).
- **save\_dir** (*str*, *optional*) – Path to “save” *save\_dir* (e.g. “./beach\_videos/”), where the videos will be stored. If any of the directories in the path do not exists then they will be created.
- **max\_n** (*Any*, *optional*) – Max number of videos to download. It can be a number or simply ‘all’
- **wait\_time** (*int*, *optional*) – Wait time between requests, use it to not get blocked for too many requests.

**Returns** None, It automatically saves the videos to the save *save\_dir*.

**Return type** None

`crawlers.youtube.yt_search.search` (*query: list*, *save\_dir: str*, *max\_n*, *wait\_time: int = 10*)

Performs a query in youtube then downloads every video to the save *save\_dir*.

#### Parameters

- **query** (*list*, *optional*) – The query strings list, can contain a single string (e.g. [‘beach’]) or multiple strings (e.g. [‘boxing’, ‘MMA’]).
- **save\_dir** (*str*, *optional*) – Path to “save” *save\_dir* (e.g. “./beach\_videos/”), where the videos will be stored.
- **max\_n** (*Any*, *optional*) – Max number of videos to download. It can be a number or simply ‘all’
- **wait\_time** (*int*, *optional*) – Wait time between requests, use it to not get blocked for too many requests.

**Returns** None, It automatically saves the videos to the save save\_dir.

**Return type** None

## Module contents

### 1.1.2 Module contents



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### C

- `crawlers`, 5
- `crawlers.rnp`, 3
- `crawlers.rnp.rnp_crawler`, 1
- `crawlers.youtube`, 5
- `crawlers.youtube.yt_downloader_from_csv`,  
3
- `crawlers.youtube.yt_search`, 4





## INDEX

### A

`already_downloaded()` (in module `crawlers.youtube.yt_downloader_from_csv`), 3

### C

`crawl_and_download()` (in module `crawlers.rnp.rnp_crawler`), 1

`crawlers`  
module, 5

`crawlers.rnp`  
module, 3

`crawlers.rnp.rnp_crawler`  
module, 1

`crawlers.youtube`  
module, 5

`crawlers.youtube.yt_downloader_from_csv`  
module, 3

`crawlers.youtube.yt_search`  
module, 4

### D

`download()` (in module `crawlers.youtube.yt_downloader_from_csv`), 3

`download_file()` (in module `crawlers.rnp.rnp_crawler`), 2

### L

`log()` (in module `crawlers.rnp.rnp_crawler`), 2

### M

`main()` (in module `crawlers.youtube.yt_search`), 4

module

`crawlers`, 5

`crawlers.rnp`, 3

`crawlers.rnp.rnp_crawler`, 1

`crawlers.youtube`, 5

`crawlers.youtube.yt_downloader_from_csv`,  
3

`crawlers.youtube.yt_search`, 4

### R

`read_csv_and_download_videos()` (in module `crawlers.youtube.yt_downloader_from_csv`), 3

### S

`scandown()` (in module `crawlers.rnp.rnp_crawler`), 2

`search()` (in module `crawlers.youtube.yt_search`), 4

`sizeof_fmt()` (in module `crawlers.rnp.rnp_crawler`),  
2