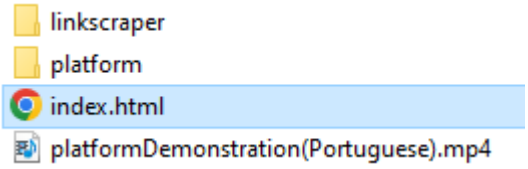
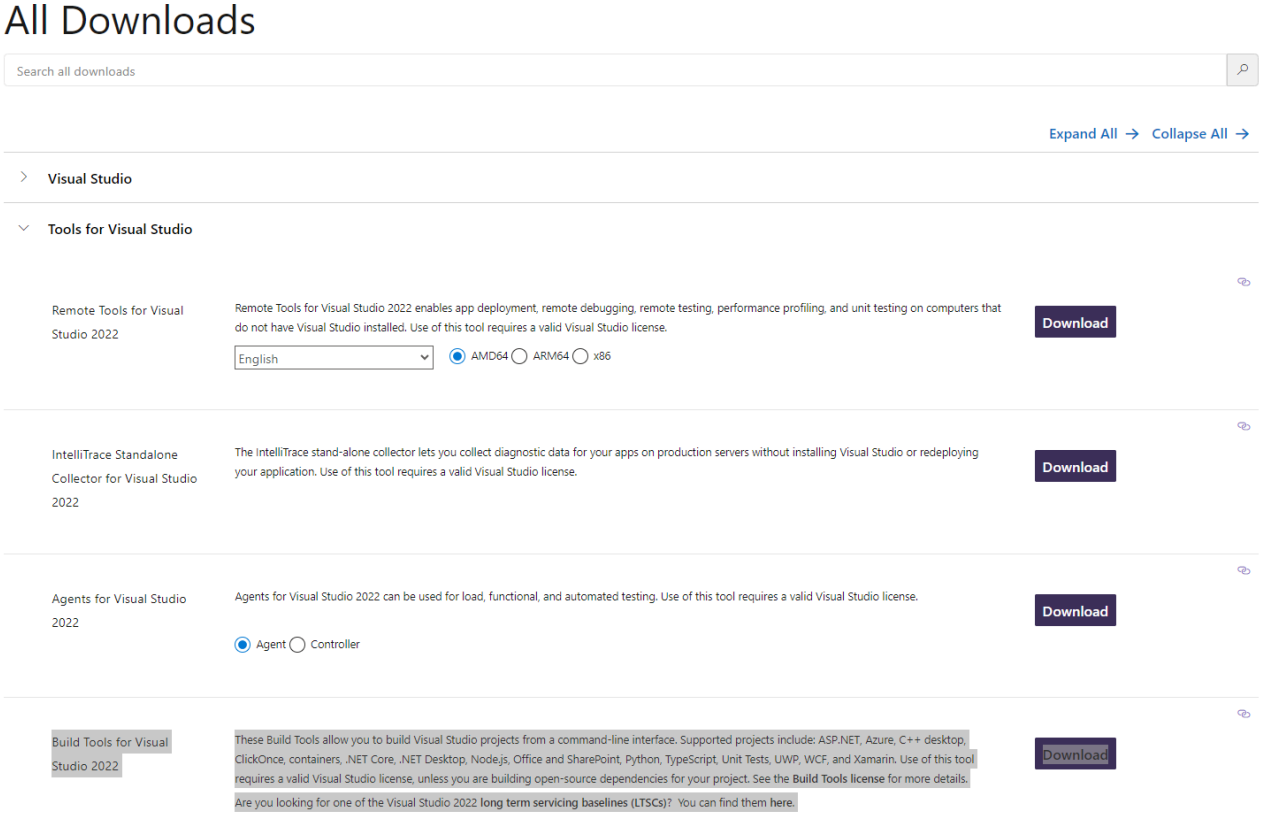
Bibliometric Data Gathering Tool

User's Test Guide

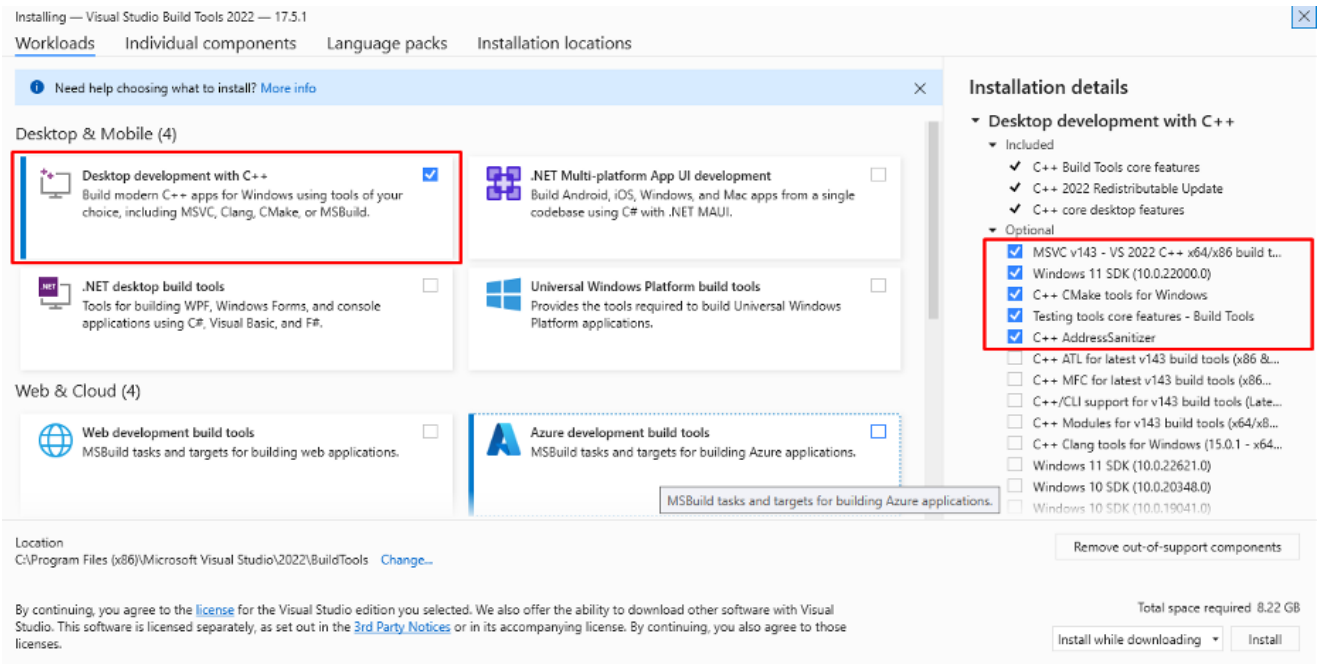
1. ***Platform:***
   - On the root folder double-click the "index.html" file.



   - A video demonstration of the platform can be visualized by double-clicking the "platformDemonstration(Portuguese).mp4" file.
2. ***System Dependencies Installation (Windows Users):***
   - Microsoft Visual C++ Build Tools:
     i. Access the following link: "https://visualstudio.microsoft.com/downloads/".
     ii. Scroll Down to the "All Downloads Section".
     iii. Expand the "Tools for Visual Studio" Tab.
     iv. Click "Download" on the "Build Tools for Visual Studio" option.



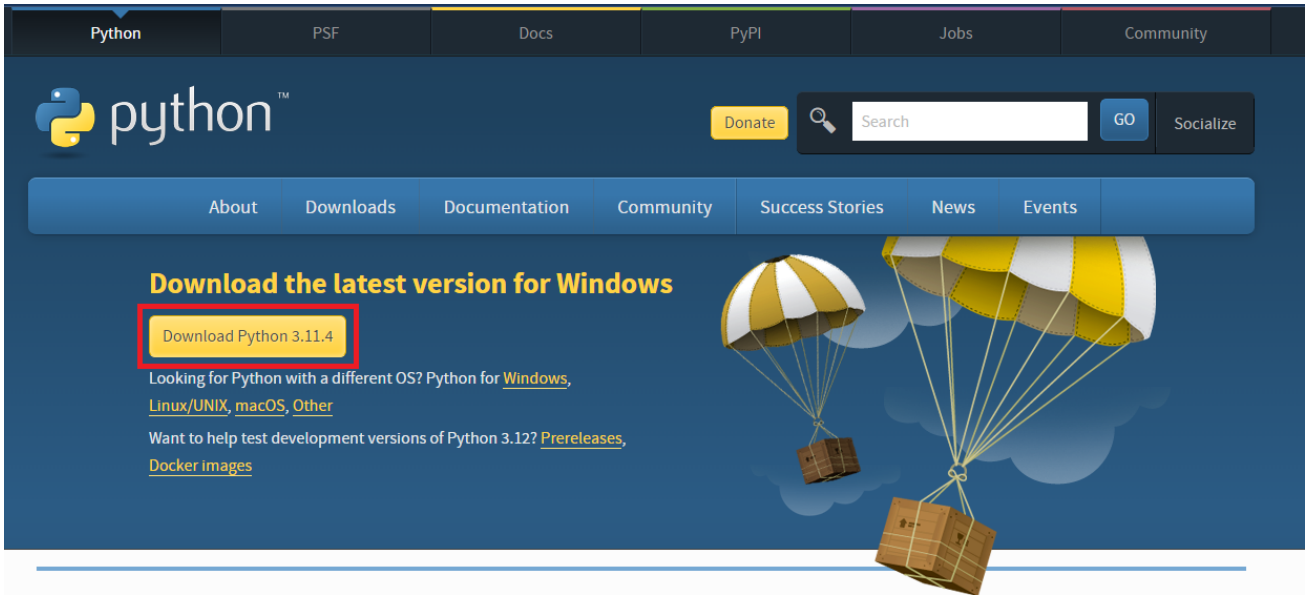
     v. Execute the extracted executable file.
     vi. At the initial installation screen, select the "Continue" button.
     vii. Select the "Desktop development with C++" option and ensure the optional features are selected too.



viii. Select the “Install” button to start downloading Microsoft Visual Build tools on the Windows system.

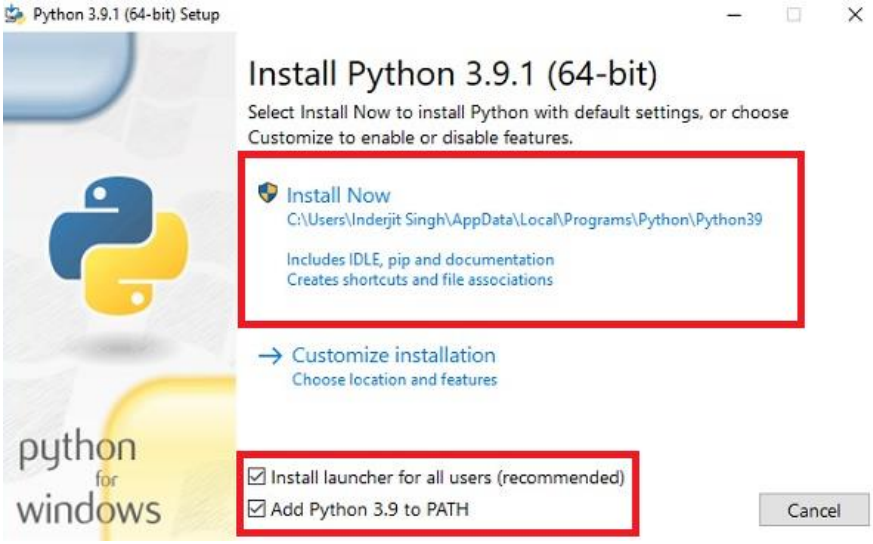
ix. Wait until the Build tools installation is completed and then restart the computer.

- Python:
  i. Access the following link: https://www.python.org/downloads/.
  ii. Click on the “Download Python” button.



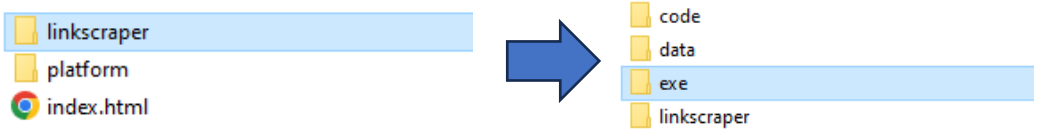
iii. Execute the extracted download file.

iv. Click on the “Install Now" button and ensure that the optional features are selected too.

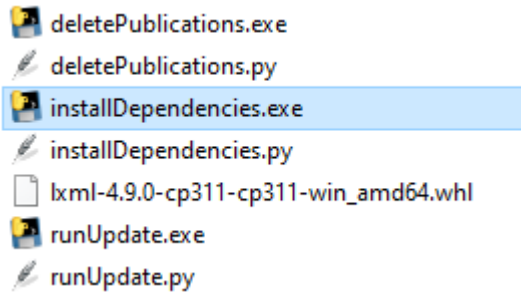


v. Wait until the installation is completed.

- Remaining Dependencies:
  i. On the tool's root folder navigate to the "linkscraper" -> "exe" folders.



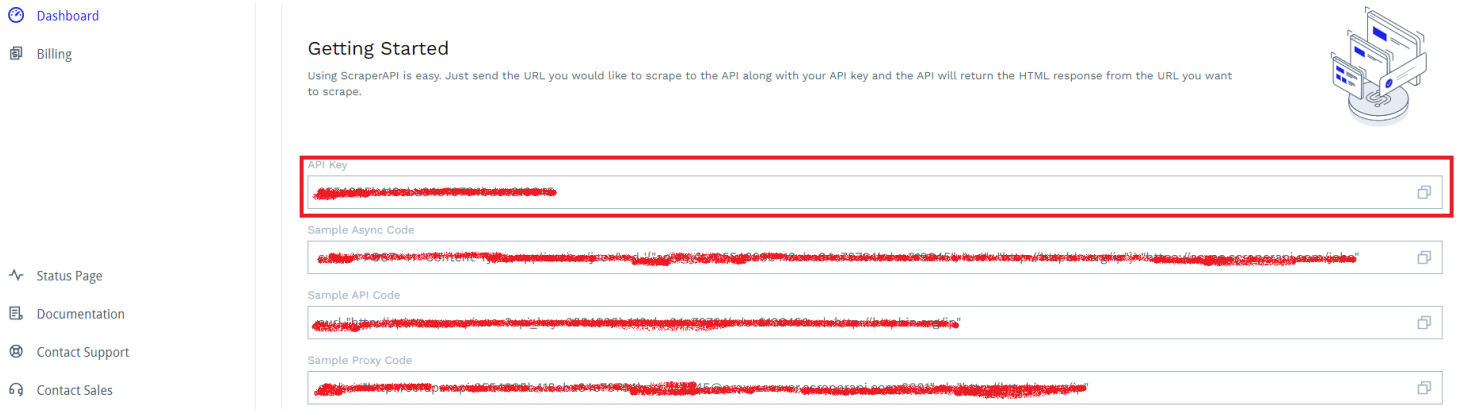
  ii. Double-click on the "installDependencies.exe" executable file (Windows Users) or run the following console command "python installDependencies.py" on the same folder as the executable file (non-Windows Users).



  iii. Wait until the "lxml", "scrapy", "scholarly" and "xlsxwriter" packages are installed.

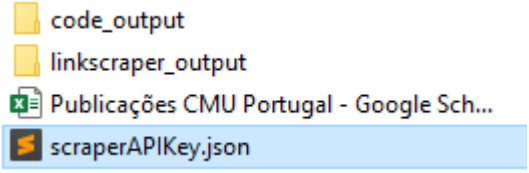
3. *ScraperAPI:*

- Create an account on the following website: https://www.scraperapi.com/.
- Access your account's dashboard.
- On the "Dashboard" tab, copy the "API Key".

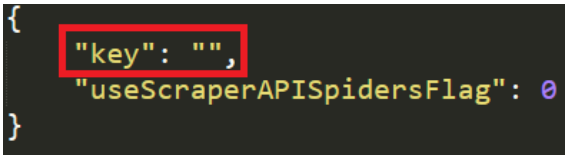


- On the tool's root folder, navigate to the "linkscraper" -> "data" folders.

- Open the "scraperAPIKey.json" file with wordpad (bloco de notas).

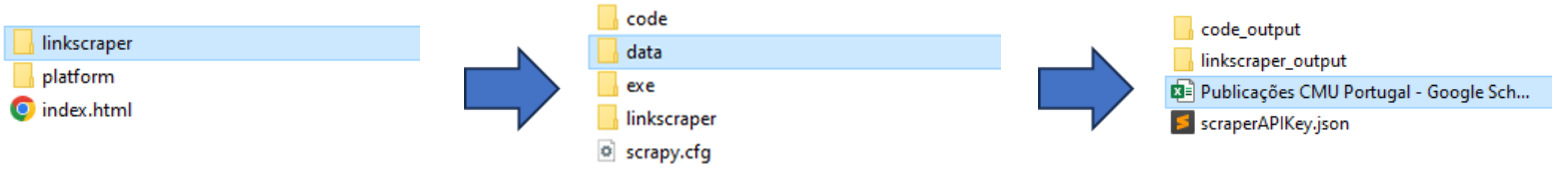


- On the field "key" paste the copied "API Key" between the quotation marks (" ").

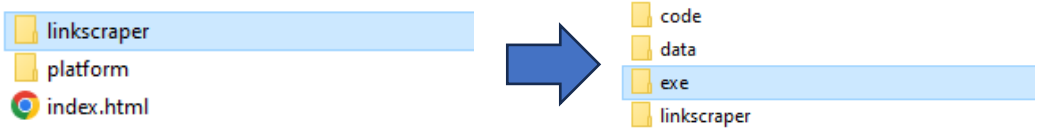


4. *Data Extraction Tool (Windows Users):*

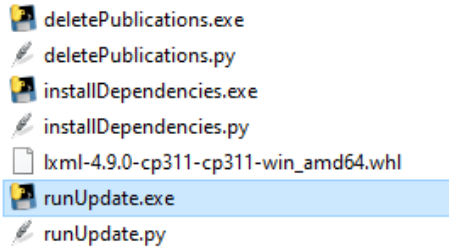
- Authors Insertion: It is possible to add authors in the "Publicações CMU Portugal - Google Scholar" file. We can navigate to this file from the tool's root folder to the "linkscraper" -> "data" folders.



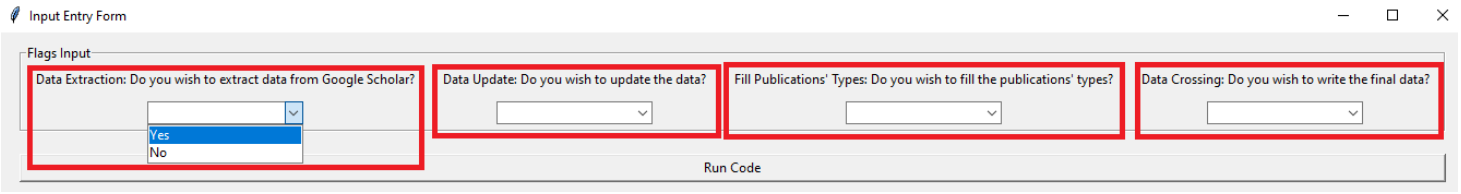
- Tool's Execution:
  i. Run Update:
    1. On the tool's root folder navigate to the "linkscraper" -> "exe" folders.



    2. Double-click on the "runUpdate.exe" executable file (Windows Users) or run the following console command "python runUpdate.py" on the same folder as the executable file (non-Windows Users)..



    3. The following window then appears:



To provide users with more flexibility and control over the execution of the code, we developed a feature that opens an interface allowing them to select the specific code sections they wish to execute.
This approach enables users to selectively execute different parts of the code based on their needs.
Not every user may require the complete functionality at once, and by offering the option to choose specific modules, they can customize the execution to suit their requirements. For example, users can choose to execute the data extraction module alone to retrieve new data without performing the data crossing or update steps. Additionally, this feature accommodates scenarios where some files have already been updated and users only need to write the final data to the platform.

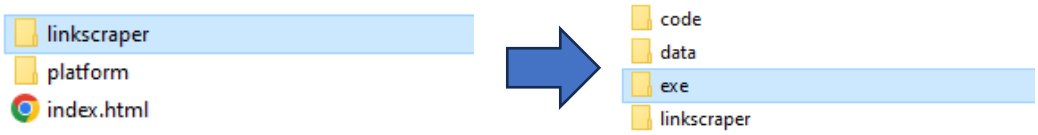
The code is separated in the following manner:

a. Data Extraction (it may or may not use ScraperAPI): This code part extracts the information of each author's Google Scholar profile file and their respective publications.
b. Data Update: Updates the code's backup file. It verifies if there is any missing publication in the most recent extracted data.
c. Fill Publication's Types (always uses ScraperAPI): This part fills any missing publication's types that were not able to be identified from the Data Extraction module.
d. Data Crossing: This module uses the extracted and updated information about authors and their publications. The extracted data is used to extract valuable insights by crossing and evaluating various data points and fields and to turn this data into new information of interest.

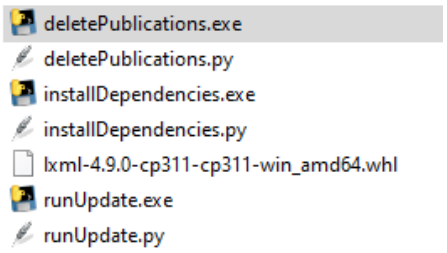
4. Select which part of the code you wish to execute by choosing "Yes" or "No" in each drop-down menu.
5. Click the "Run Code" button.
6. Wait for the execution to end (about 2 hours for 88 students).

ii. Delete Publications:

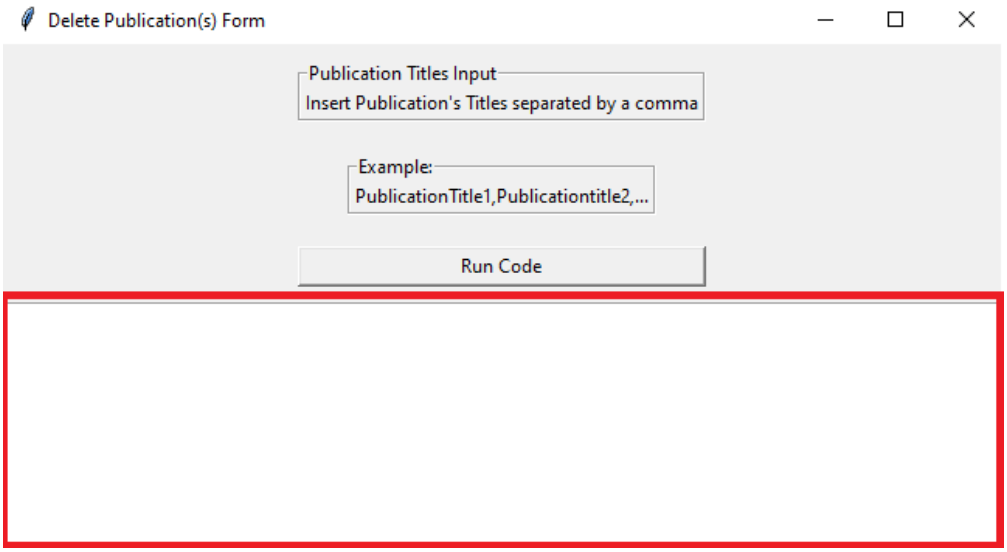
1. On the tool's root folder navigate to the "linkscraper" -> "exe" folders.



2. Double-click on the "deletePublications.exe" executable file (Windows Users) or run the following console command "python deletePublications.py" on the same folder as the executable file (non-Windows Users).



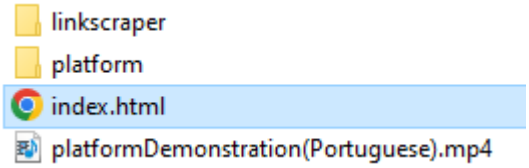
3. The following window then appears:



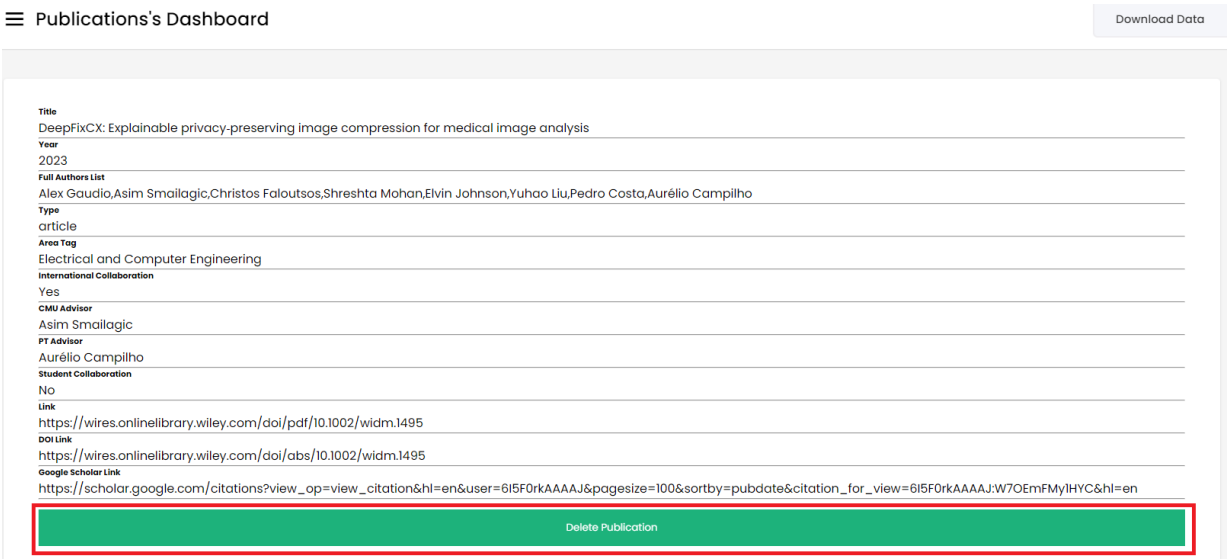
To ensure data accuracy and maintain the integrity of the CMU Portugal profile, we developed the "Delete Publications" feature. This feature allows users to indicate specific publications that should be excluded from the

CMU Portugal dataset. By providing this option, we enable users to remove publications that fall outside the intended timeframe, eliminating any potential inaccuracies or misrepresentations.

4. On the root folder double-click the “index.html” file.



5. Go to the profile of the publication that you wish to delete.
6. Click on the “Delete Publication” button and confirm the deletion.

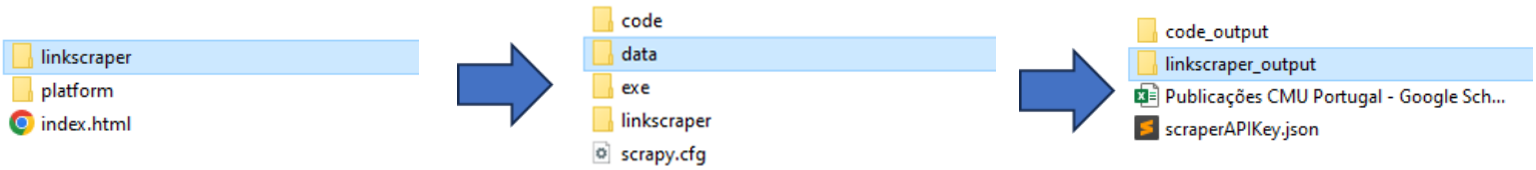


7. Open the downloaded file.
8. Copy the publication’s title from the downloaded file.

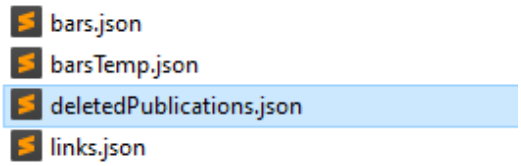
9. Paste the copied title to the “deletePublications.exe” graphical interface text box (If you wish to delete more than one publication at a time, separate the publications’ titles by a comma (example: Title1,Title2)).
10. Click the “Run Code” button.
11. Wait for the execution to end.

iii. <u>Adding Publications Back</u>:

1. On the tool’s root folder navigate to the “linkscraper” -> “data” -> “linkscraper_output” folders.



2. Double-click on the “deletedPublications.json” file and open with wordpad (bloco de notas)



3. Delete the line with the publication title that you wish to put back in the dataset.

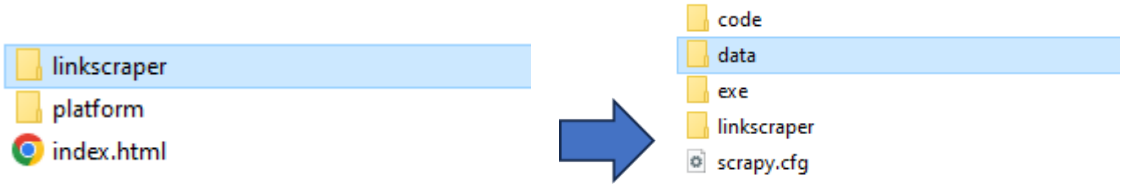
```
[
    "Title1",
    "Title2",
    "Title3",
    "Title4",
    "Title5",
]
```

4. On the next update, this publication will be added again.

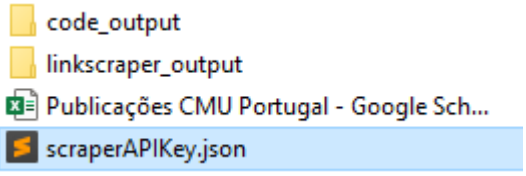
5. *Additonal Extraction Information (ScraperAPI)*

Our tool allows to either use or not use the ScraperAPI as proxy in the "Data Extraction" module. A proxy service is a server infrastructure that acts as an intermediary between a client's request for a resource and the server that provides that resource. Since we will be extracting enough data from Google Scholar to risk having our Google Scholar requests rejected and blocked, it's worth emphasizing that we'll need a proxy infrastructure. However, further in the implementation, we stopped using ScraperAPI because it has a monthly price value for limited credits for requests. We did this by masking our session to Google Scholar and customizing request headers.

- Use ScraperAPI:
  i. On the tool's root folder, navigate to the "linkscraper" -> "data" folders.



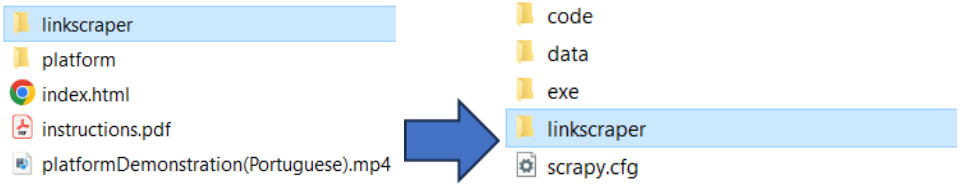
  ii. Open the "scraperAPIKey.json" file with wordpad (bloco de notas).



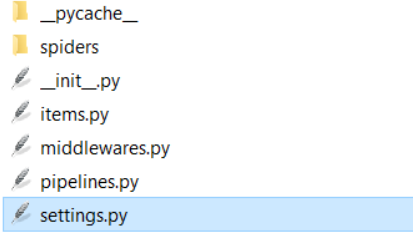
  iii. On the field "useScraperAPISpidersFlag" change the value from "0" to "1".

```
{
    "key": "",
    "useScraperAPISpidersFlag": 0
}
```

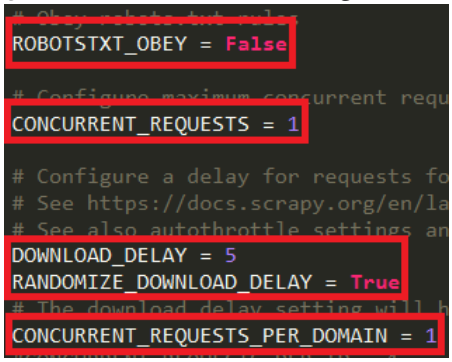
  iv. From the tool's root folder navigate to the "linkscraper" -> "linkscraper" folders



  v. Open the "settings.py" file with wordpad (bloco de notas).

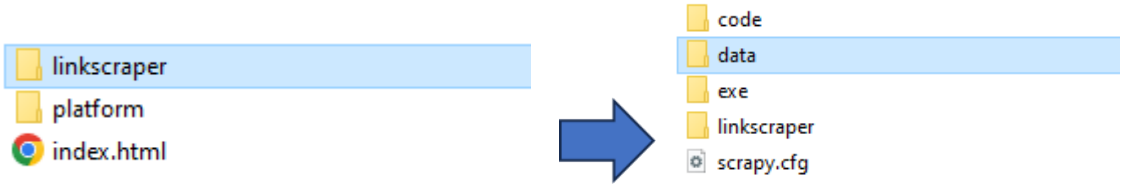


vi. Change the following fields:
    1. ROBOTSTXT_OBEY: “False” to “True”.
    2. CONCURRENT_REQUESTS: “1” to “16”.
    3. DOWNLOAD_DELAY: “5” to “0”.
    4. RANDOMIZE_DOWNLOAD_DELAY: “True” to “False”.
    5. CONCURRENT_REQUESTS_PER_DOMAIN: “1” to “8”.

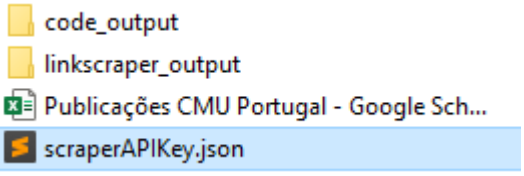


vii. With ScraperAPI it takes 2 to 5 minutes to extract to total amount of data. On average, it takes 11 000 credits for 88 students (125 credits per author).

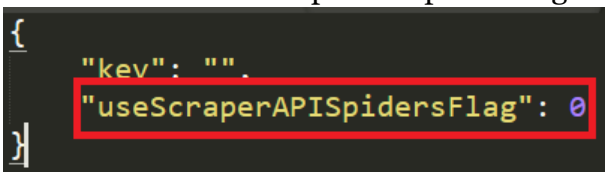
- <u>Not use ScraperAPI</u>:
    i. On the tool’s root folder, navigate to the “linkscraper” -> “data” folders.



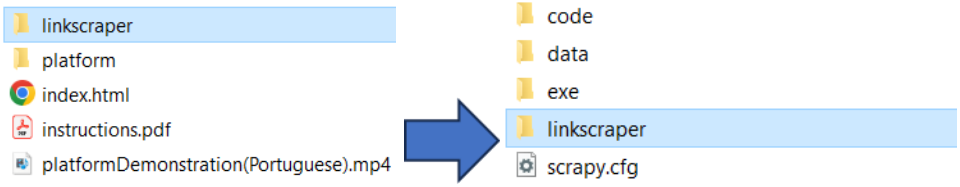
    ii. Open the “scraperAPIKey.json” file with wordpad (bloco de notas).



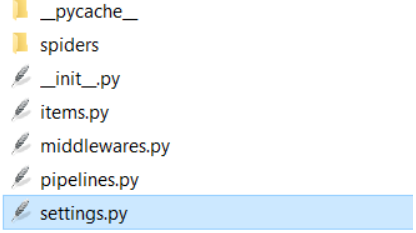
    iii. On the field “useScraperAPISpidersFlag” change the value from “1” to “0”.



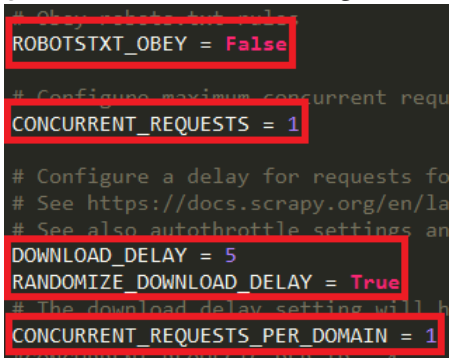
    iv. From the tool’s root folder navigate to the “linkscraper” -> “linkscraper” folders



    v. Open the “settings.py” file with wordpad (bloco de notas).



vi. Change the following fields:
    1. ROBOTSTXT_OBEY: “True” to “False”.
    2. CONCURRENT_REQUESTS: “16” to “1”.
    3. DOWNLOAD_DELAY: “0” to “5”.
    4. RANDOMIZE_DOWNLOAD_DELAY: “False” to “True”.
    5. CONCURRENT_REQUESTS_PER_DOMAIN: “8” to “1”.



vii. Without ScraperAPI it takes around 2 hours to extract to total amount of data (1 minute per author).

- Scholarly library:
    i. The scholarly library always uses ScraperAPI as a proxy.
    ii. On average, it costs 100 credits to fill out a publication type.
    iii. We need to fill a publication type in 16% of publications.
    iv. On average, each student has 11 publications, meaning that on average, it costs (11*100)*0.16 credits to fill the publications’ types for each student (176 credits).
    v. For 88 students it costs 16000 credits to fill all publications with missing types.
- ScraperAPI Monthly Plans:
    i. The monthly plans of ScraperAPI are available in the following link: https://www.scraperapi.com/pricing/.