# Web scraping project – scraping the page legimi.pl

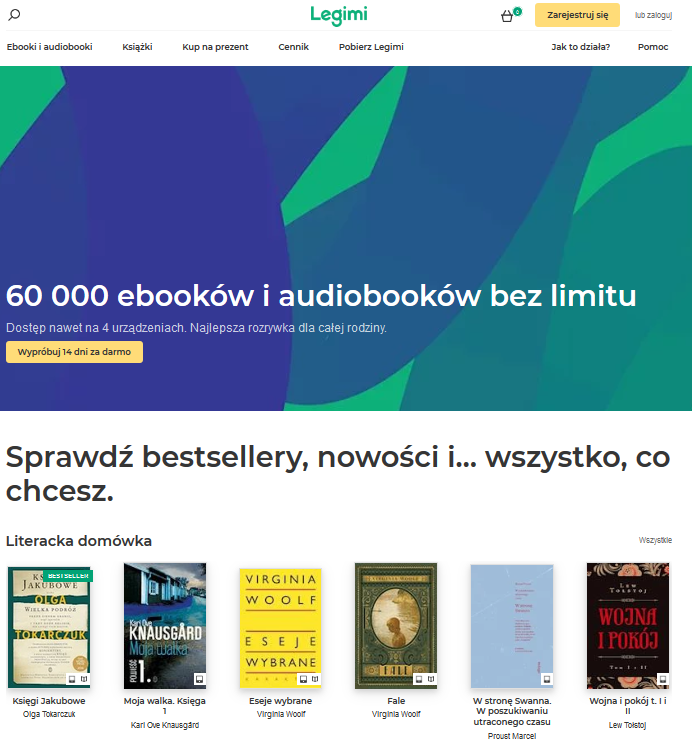
Authors: Maria Kubara (384628) & Mateusz Majak (384912)

Site: <https://www.legimi.pl/>

## Page description:

Legimi is a book rental service that allows its subscribers to access unlimited ebooks and audiobooks for a monthly fee. Besides the subscription service, it also offers a standard purchase of a book in a paperback, online or audio form. Right now one can find there more than 60.000 different books (see Fig. 1). Robots.txt file of the page allows for scraping individual book’s pages.

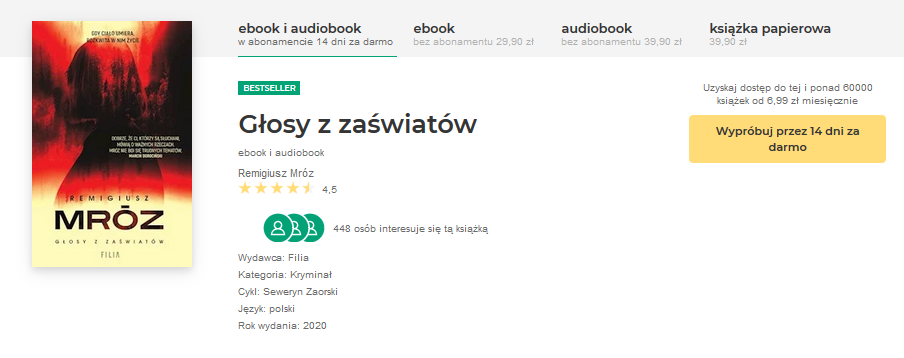
**Figure 1.** Welcome page of the Legimi service



Source: <https://www.legimi.pl/>

Each book available at the webpage is well described and includes numerous information about the author, publisher and novel’s type (see Fig. 2). Having the data about the price, popularity and readers’ rating one can use these pieces of information to prepare analysis about the current reading trends (more in: Basic data analysis).

**Figure 2.** A book description - sample page

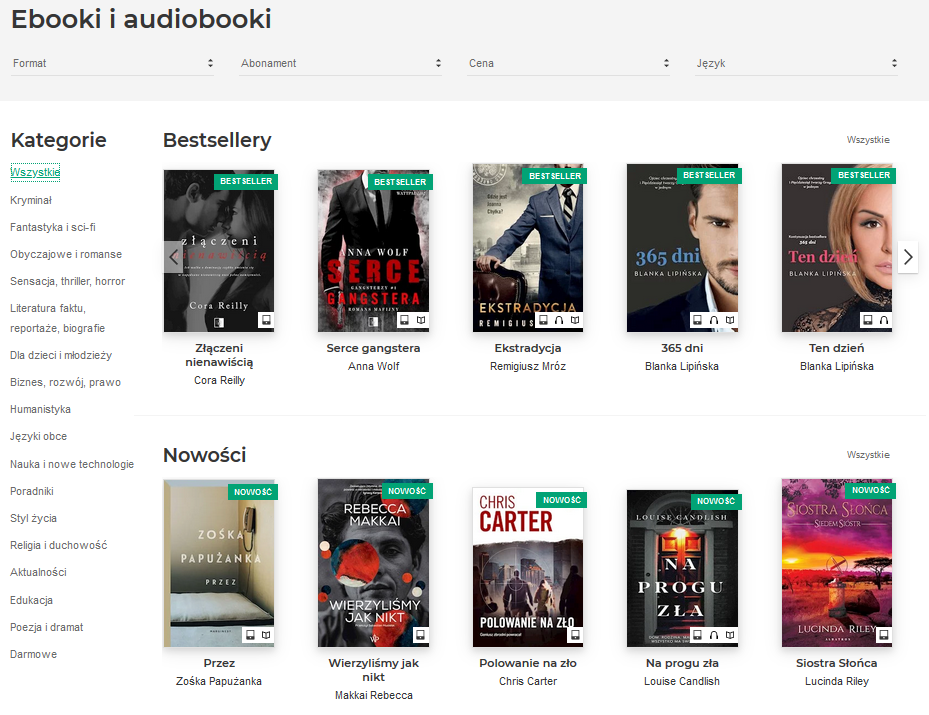


Source: <https://www.legimi.pl/>

## Interactive site character

Each book site is built as a static page, but to get there one needs to look for it on a main browse page, build with JavaScript dynamic functions (see Fig. 3).

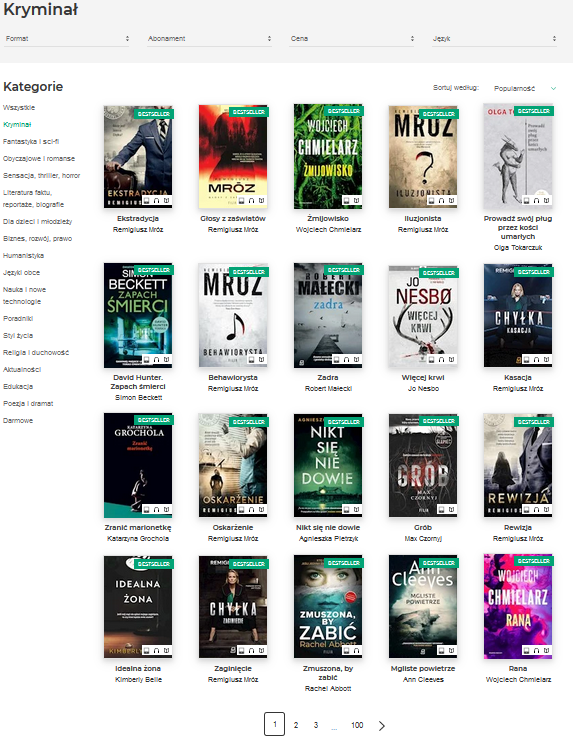
**Figure 3.** Main browse page: JS carousels, dynamic content loading and more



Source: <https://www.legimi.pl/>

Our aim was to get all information about criminals available on the Legimi site (2363 books on the 6th of April). To get there we limited the search area to the site dedicated to this kind of novels (see Fig. 4).

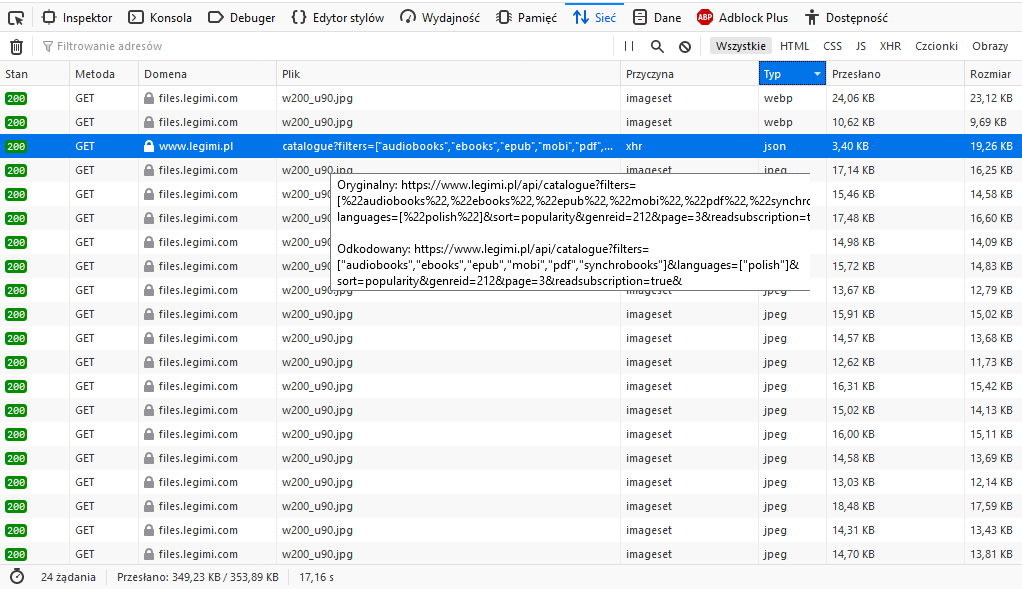
**Figure 4.** Browse page dedicated to criminal novels



Source: <https://www.legimi.pl/>

Anyhow, the page content is still automatically generated on the viewer demand - click on the next page button forces the server to put new information on the site (see Fig. 5).

**Figure 5.** Tracker of network flow, after clicking the next page button (json files imputed to the site content)



Source: <https://www.legimi.pl/>

## Scraper mechanics

We’ve decided to build two scrapers. First build with Beautiful Soup and Selenium and the other with the Selenium and Scrapy usage. Both are shortly characterized below.

1. **BeautifulSoup + Selenium**

Starting with the site <https://www.legimi.pl/ebooki/kryminal,g212/?sort=popularity> (allowing for browsing the criminal novels, available at the Legimi website) BeautifulSoup was used to get reference links to all books dedicated to this category. As each browse page included only the first twenty books (sorted by popularity), Selenium was used to simulate a click to the next page. Then the updated html document was fed to the Beautiful Soup and the links to consecutive books were obtained. The same procedure was repeated until the last browse page.

After obtaining all links to individual books (2363 links to book’s webpages scraped on the 6th of April) BeautifulSoup was used to retrieve information about each of them (detailed explanation in the Output description). The data was cleaned with some regex expressions (to get only the interesting part of information) and was finally stored in a csv file in UTF-8 formatting, ready to be used for the next analysis.

Source code for this scraper is available in the *combineBS.py* file.

The analysis was focused only on the criminal novels (which already gave a lot of data to be processed), but the solution was built to be flexible. If one was interested in other type of books only the first link needs to be changed. For example, if we would like to scrap self-help books the first *url* variable needs to be changed to <https://www.legimi.pl/ebooki/poradniki,g214/?sort=popularity> – all the other parts of the code will adapt and give a resulting csv file with information about these books. To get data about all books available in the Legimi service one simply needs to start with the link <https://www.legimi.pl/ebooki/?sort=popularity>.

1. **Scrapy + Selenium**

This solution follows similar steps as the previous scraper, but different technical solutions were used.

In this approach it was also started with the same criminal category site <https://www.legimi.pl/ebooki/kryminal,g212/?sort=popularity>. As it was mentioned before, each browse page included only the part of all books in the category. Thus, Selenium was used to get all links to the pages containing reference urls to criminal books. Similar to the previous approach Selenium simulated a click to the next page. It was iterating through the pages and appending current pages links to the list. Then, all links from the list was saved in a csv file using Pandas.

After obtain links to the pages Scrapy project was created. The first spider scraped all of the books links (links to all of the books in the criminal category) using urls scraped with Selenium and saved them into csv file. Then, second spider was used to scrap all of the chosen information about books. The data was cleaned with regular expressions just like in the previous approach.

Source code for this approach is available in the *selenium\_url\_pages\_extractor.py,*

*legimi\_criminal\_books\_urls.py* and *Legimi\_all\_pages.py* files.

This solution allows to scrap different categories as well. Also all of the books information can be obtained by starting start with the link <https://www.legimi.pl/ebooki/?sort=popularity> (just like in the BeautifulSoup + Selenium approach).

## Output description

After each scraper finishes its work, a csv file will be received. The form is as follows: information about each book is stored in an individual row. The output table will include following variables, describing each position:

* a book’s title,
* its author,
* publisher data,
* assigned category (categories),
* received score (averaged readers’ rating in scale 1-5),
* the price of an ebook (given the digital form is offered),
* the price of an audiobook (if the audio form is available),
* the price of a paper book (given that a paper cover is in stock),
* number of people interested (count of readers, that added this book to their to-be-read list).

For some books not every pricing option is available, as the service offers different books in different reading options. In those cases, missing values are generated.

**Table 1.** Outline of the first few rows in the output table

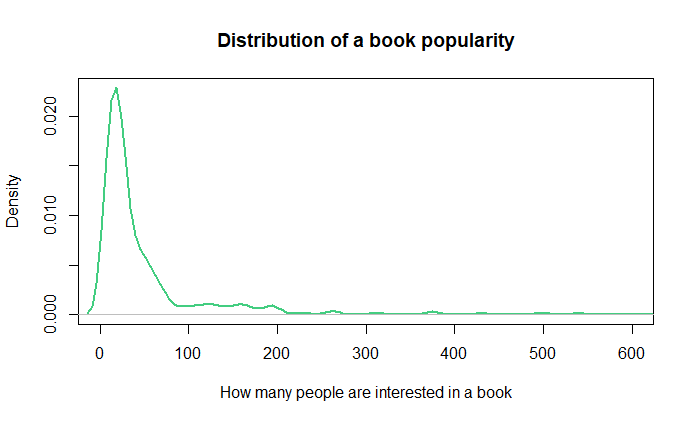
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Title | Author | Publisher | Category | Score | Ebook  Price | Audiobook  Price | Paper  Price | People  Interested |
| Przejęcie | Wojciech Chmielarz | Wydawnictwo Czarne | Kryminał | 4,4 | 27,9 | 29,9 | 39,9 | 108 |
| Zombie | Wojciech Chmielarz | Wydawnictwo Czarne | Kryminał | 4,3 | 33,9 | 29,9 | 39,9 | 51 |
| Hashtag | Remigiusz Mróz | Czwarta Strona | Kryminał | 3,9 | 31,9 | 39,9 | 14,9 | 52 |
| Listy zza grobu | Remigiusz Mróz | Filia | Kryminał | 4,2 | 29,9 | 39,9 | 39,9 | 155 |
| Rana | Wojciech Chmielarz | Marginesy | Kryminał | 4,1 | 31,91 | 39,9 | 39,9 | 119 |
| Nieodnaleziona | Remigiusz Mróz | Filia | Kryminał | 4,2 | 29,9 | 39,9 | 39,9 | 72 |

Source: Own study based on the data scraped from <https://www.legimi.pl/>

## Basic data analysis

Obtained data can be utilized in future analysis. One idea of the upcoming research could be an unfairness of the popularity distribution. A density plot representing the number of people interested in a particular book indicates a significant skewness of the popularity distribution (see Fig. 6). There are many books with relatively low popularity count, and few outliers with incomparably higher numbers of interested people.

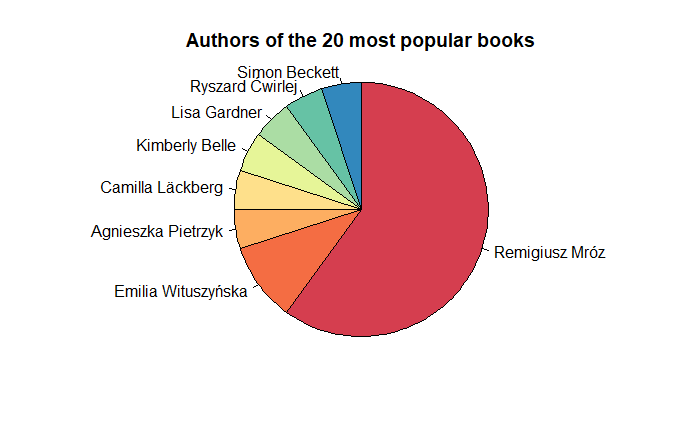
**Figure 6.** Density plot of a book popularity distribution



Source: Own study based on the data scraped from <https://www.legimi.pl/>

Similar popularity unfairness can be seen in the pie plot, representing the authors of twenty most popular criminal novels (see Fig. 7). Most of the most successful books were written by one person.

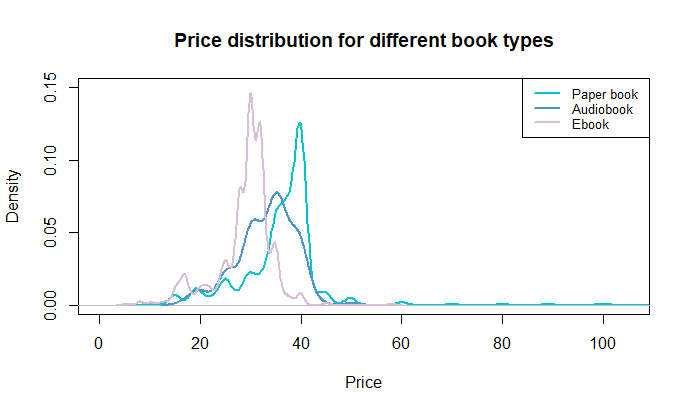
**Figure 7.** Authors of the twenty most popular criminal novels



Source: Own study based on the data scraped from <https://www.legimi.pl/>

Thanks to the pricing information one can also prepare an analysis of a book pricing policy, based on its popularity, author and readers rating. One can even prepare a model attempting to explain a particular book price. Even without digging in too much details we can make some interesting observations on the pricing policies for different book types offered in the Legimi service. As we can see in the Fig. 8 all distributions of book prices have a clear spike in the middle - most popular pricing option for a particular book type. Paper book price distribution has large number of outliers, while the books sold in digital form have a more even price distribution.

**Figure 8.** Distribution of prices for different book types



Source: Own study based on the data scraped from <https://www.legimi.pl/>

Source files:

Project was prepared in a high cooperation and with equal workload distribution (solving issues together). Some comments about the author of the codes are given in the files (as required in the project rules), but this is only an indicator of a ‘person a bit more responsible for the last outlook of that code part’, as all the problems were solved in the cooperation mode. Work on cleaning the code and final report preparation was also divided equally.

In both scrapers a Boolean value was added (‘limiter’) that allows for the limiting of the scraper to the 100 pages scraped only.

**Files:**

*combineBS.py* – full scraper made in BeautifulSoup and Selenium

**Files for Scrapy rewrite:**

*selenium\_url\_pages\_extractor.py* – get links to books with Selenium

*legimi\_criminal\_books\_urls.py* – feeding the book links to the spider

*Legimi\_all\_pages.py* – using Scrapy to get individual book’s information

**Files with .xlsx format:**

Resulting files from the scrapers’ operation