

УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ



Зорана Гајић

САВРЕМЕНИ АЛАТИ ЗА ПРИКУПЉАЊЕ
ПОДАТАКА СА ВЕБ-СТРАНИЦА

мастер рад

Београд, 2023.

Ментор:

др Милена ВУЛОШЕВИЋ ЈАНИЧИЋ, ванредни професор
Универзитет у Београду, Математички факултет

Чланови комисије:

др Ана АНИЋ, ванредни професор
University of Disneyland, Недођија

др Лаза ЛАЗИЋ, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 15. јануар 2016.

Порогици

Наслов мастер рада: Савремени алати за прикупљање података са веб-страница

Резиме:

Кључне речи: прикупљање података са веб-страница

Садржај

1	Увод	1
2	Прикупљање података са веб-страница	2
2.1	Изазови	3
2.2	Примењене технологије	5
3	Преглед алата за прикупљање података са веб-страница	10
3.1	Библиотека <i>BeautifulSoup</i>	10
3.2	Библиотека Selenium	20
3.3	Библиотека Scrapy	20
4	Анализа резултата	21
5	Закључак	22
	Библиографија	23

Глава 1

Увод

TODO:

Глава 2

Прикупљање података са веб-страница

Веб (енг. *World Wide Web*, *WWW* ¹) представља највећи извор података у историји човечанства, али се већина ових података састоји од неструктурираних информација, што може отежати њихово прикупљање [4]. На многим веб-сајтовима забрањено је копирање и преузимање података, али на сајтовима на којима је преузимање података дозвољено, ручно копирање може потрајати данима или недељама.

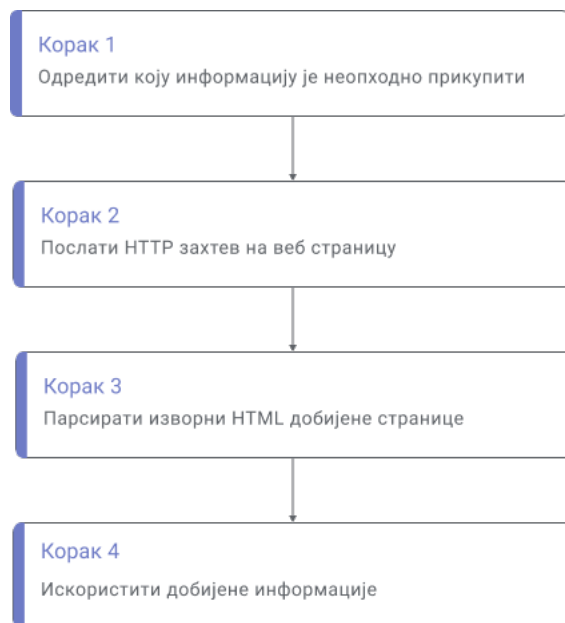
Веб скрејпинг (енг. *Web scraping*) представља аутоматизовани процес који омогућава издвајање података са различитих веб-страница и њихово чување у структурираном формату ради тренутне употребе или касније анализе. Постоје различити програмски језици који пружају подршку за имплементацију Веб скрејпинга, од којих су најпопуларнији: Пајтон (енг. *Python*), Јава (енг. *Java*) и Руби (енг. *Ruby*).

Поступак прикупљања информација састоји се од неколико фаза, које су приказане на слици 2.1. Прва фаза је проналажење одговарајуће веб-странице за прикупљање података (детаљније објашњено у одељку 2.1) и одређивање информација које су потребне за прикупљање. Након тога, потребно је послати *HTTP* (енг. *Hypertext Transfer Protocol* ²) захтев на жељену веб-страницу и преузети изворни код *HTML* странице. Пре него што се парсира *HTML* код, потребно је пронаћи најбољи начин за индексирање жељених

¹Светска мрежа, познатија као Веб, систем је међусобно повезаних, хипертекстуалних докумената који се налазе на интернету.

²*HTTP* је мрежни протокол који припада слоју апликације референтног модела ОСИ, представља главни и најчешћи метод преноса информација на Вебу.

елемената, а затим парсирати изворни код *HTML* странице и извршити неопходну радњу са добијеним информацијама.



Слика 2.1: Фазе прикупљања и употребе података

2.1 Изазови

Веб скрејпинг се сматра корисним процесом за добијање увида у податке. Међутим потребно је пазити на правне аспекте, како би се избегли легални проблеми, јер ниједан веб-сајт не жели да дозволи крађу података. Да би прикупљање података било успешно, од суштинског значаја је квалитет добијених података. Како би се добили квалитетни подаци, потребно је да је сам веб-сајт исправан, односно да не садржи неисправне линкове, јер се веб скрејпинг обично изводи преко целог веб-сајта, а не само преко одређених страница.

Када се ради о пројектима великих размера и обимних база података, један од честих изазова јесте складиштење података. Овај изазов је повезан са ефикасним прикупљањем, обрадом и анализом велике количине података који се могу прикупити путем веб скрејпинга са различитих извора. Овај проблем може бити решен употребом већ постојећих платформи за складиштење.

Када се ради о динамичким веб-сајтовима, решење може бити приказивање у претраживачу без заглавља (енг. *Headless Chromium* [6]), што омогућава да се претраживач покреће у окружењу сервера. Пријављивање корисника на веб-страницу може представљати велики изазов, али се то може решити уз помоћ библиотеке као што су *Selenium* и *Scrapy*.

У наставку ће бити описане најчешће заштите од напада на веб-странице који могу ометати процес веб скрејпинга:

1. *CAPTCHA* (енгл. *Completely Automated Public Turing test to tell Computers and Humans Apart*)

CAPTCHA је технологија која се користи за проверу и потврду да је корисник веб-странице заиста човек, а не компјутерски програм. То се постиже приказивањем изазова, на пример слике са текстом или бројевима, које је лако за људе да га виде и реше, али тешко за компјутерске програме да га аутоматски реше. Корисници обично морају да унесу решење изазова како би потврдили да су људи и дозволили приступ подацима на веб страницама.

2. Захтеви за аутентификацију

Пре приступања подацима, кориснику може бити захтевано да унесе своје корисничко име и лозинку или да потврди своју електронску пошту.

3. Блокирање *IP* (енгл. *Internet Protocol address*) адреса.

Веб странице могу блокирати *IP* адресе које се повезују са прекомерним бројем захтева или са ботовима који су идентификовани као нежељени. Ово може бити привремено или трајно.

4. Провера корисничког агента.

Сваки *HTTP* захтев у заглављу шаље корисничког агента (енг. *user agent*). Коришћењем овог подешавања веб-сајт идентификује претраживач који му приступа: његову верзију и платформу. Уколико се користи исти кориснички агент у сваком захтеву, веб-сајт може лако да открије да је у питању аутоматизовани приступ страници.

5. Праћење учесталости прикупљања података.

Да би се спречило преузимање садржаја са веб-странице, веб-сајт може увести ограничење фреквенције за ботове. Циљ је спречити преузимање садржаја са веб-странице у превеликој количини или превеликој брзини. Ово ограничење може укључивати број захтева по јединици времена, максималну брзину преузимања.

6. Постојање *robots.txt* фајла.

robots.txt фајл представља политику веб-сајта и најчешће се проналази на нивоу основног директоријума. Уколико фајл садржи линије попут ових приказаних у наставку, то значи да веб-сајт не жели да се прикупљају подаци са њега

```
User-agent: *  
Disallow:/
```

7. Законске заштите

У неким случајевима, веб странице се могу заштитити законима о ауторским правима, заштитом података или другим релевантним законима који спречавају копирање или употребу података са веб-странице без дозволе власника.

2.2 Примењене технологије

Веб скрејпинг технологије подразумевају различите методе и алате за издвајање података са веб-страница. У оквиру ових технологија користе се: регуларни изрази (енг. *Regular Expressions*, *RegEx*), тагови (енг. *tags*), *CSS* (енг. *Cascading Style Sheets*) селектори и *XPath* (енг. *XML Path Language* [1]).

Препоручени редослед идентификације елемената у оквиру *HTML* кода током веб скрејпинга може бити следећи:

1. Преко идентификатора - ако елементи имају јединствени идентификатор, најбрже и најпоузданије је користити овај начин идентификације.
2. По имену класе - ако се елементи налазе у истој класи, могу се идентификовати преко имена класе. Ово је корисно када је потребно издвојити групу елемената са заједничким стилом или функционалношћу.

3. По таговима - ако је неопходно издвојити све елементе са одређеним тагом, овај начин идентификације је најбољи.
4. *CSS* селектори - ако постоје елементи који немају јединствен идентификатор, али имају јединствен *CSS* стил, могу се идентификовати преко *CSS* селектора.
5. Регуларни изрази - ако је неопходно издвојити елементе на основу текста који се налази у њима.
6. *XPath* - ово је најопштији начин идентификације елемената у *HTML* коду.

Регуларни изрази

Регуларни изрази представљају метод за усклађивање специфичних образаца у зависности од датих комбинација, који се могу користити као филтери за добијање жељеног резултата. У прикупљању података регуларни изрази се често користе за поређење шаблона и издвајање података, за локализовање и издвајање специфичних података из *HTML* или *XML* докумената. Једна од најзначајнијих предности регуларних израза јесте у њиховој универзалности, тј. могу се применити на било коју врсту података.

У многим програмским језицима, регуларни изрази се подржавају кроз уграђене библиотеке или модуле. Модул *re* програмског језика Пајтон пружа подршку регуларним изразима за поређење шаблона и издвајање података. У наставку је дат пример регуларног израза који би се могао искористити за претраживање и издвајање свих линкова из изворног кода *HTML* странице:

```
1 regex_pattern = r"<a\s+(?:[^\>]*?\s+)?href=\"([^\"]*)\""
```

CSS селектори

CSS селектори се могу користити у процесу сакупљања података са веб-страница како би се идентификовали и издвојили одређени елементи. Овакав приступ је посебно користан када се ради са веб-страницама које не поседују јасну структуру и организацију.

CSS селектори раде на принципу идентификације елемената према њиховом имену ознаке, имену класе или идентификатору. На пример, селектор

„`div[class='imeKlase']`” се користи за издвајање свих *div* елемената који имају класу *imeKlase*.

Тагови

Тагови играју кључну улогу у прикупљању података са веб-страница јер помажу у идентификацији и издвајању одређених информација из изворног кода *HTML* страница. Тагови у *HTML* коду се користе за дефинисање структуре веб-странице. Сваки таг представља одређени елемент или секцију странице, као што су заглавља, пасуси, слике и линкови.

У наставку су наведени тагови који се често користе:

- `<html>`

Означава почетак и крај *HTML* документа.

- `<body>`

Представља садржај документа који је видљив кориснику.

- `<h1>` до `<h6>`

Користе се за дефинисање наслова.

- `<p>`

Користи се за дефинисање параграфа текста.

- `<a>`

Ствара хиперлинк (енгл. *Hyperlink*) до друге веб-странице или ресурса.

- `` и ``

Користе се за стварање неуређене листе ставки.

- `` и ``

Користе се за стварање уређене листе ставки.

- `<div>`

Користи се за дефинисање одељка документа у сврху стилизовања.

- ``

Користи се за дефинисање малог дела текста у сврху стилизовања.

- `<input>`

Користи се за стварање поља за унос корисничких података.

XPath

XPath представља флексибилан начин адресирања различитих делова *XML* (енг. *Extensible Markup Language* ³) документа. То га чини погодним за навигацију кроз објектни модел документа (енг. *Document Object Model, DOM* ⁴) било ког документа сличног *XML*, користећи *XPath* израз (енг. *XPathExpression* ⁵). *XPath* израз дефинише образац за одабир скупа чворова и садржи преко 200 уграђених функција. Овај језик је дефинисан од стране *WWW* конзорцијума. У овом раду ће се *XPath* користити за одабир елемената са изворног кода *HTML* страница.

***XPath* синтакса**

XPath користи изразе путања за избор чворова у *XML* документу. Чвор се одабира праћењем путање или корака. Неки корисни примери израза путања су наведени у наставку:

- `//h2`

Издаваја све *h2* елементе.

- `//div//p`

Издаваја све *p* елементе који се налазе унутар *div* блока.

- `//ul/li/a`

Издаваја све линкове који се налазе унутар неуређених листи.

- `//ol/li[2]`

Издаваја други елемент уређене листе.

- `//div/*`

Издаваја све елементе који се налазе унутар *div* блокова који нису уређени.

³ *XML* представља прошириви (мета) језик за означавање (енгл. *markup*)

⁴ Објектни модел документа представља хијерархијски приказ структуре веб-сајта.

⁵ *XPath* израз дефинише образац за одабир скупа чворова.

- `//*[@id="id"]`

Издавај елемент са одређеним идентификатором.

- `//*[@class="class"]`

Издавај све елементе са одређеном класом.

- `//a[@name or @href]`

Издавај све линкове који имају атрибут *name* или *href*.

- `//a[last()]`

Издавај последњи линк.

- `//table[count(tr)=1]`

Издавај број редова у табели.

- `string()`
`number()`
`boolean()`

Врши конверзију типова.

- `contains()`
`starts-with()`
`ends-with()`

Примењује функције са нискама.

- `//*`

Издавај све елементе.

- `//section[h1[@id='section-name']]`

Издавај све *h1* елементе који се налазе унутар *section* елемента са одређеним идентификатором.

- `//a/text()`

Издавај текст линка.

- `./a`

Тачка издавај тренутни чвор.

Глава 3

Преглед алата за прикупљање података са веб-страница

TODO: Кратак увод шта ће се користити у поглављу, који програмски језик, које библиотеке, опис сајта, жељени циљеви, препреке са којима ћу се сусрести у раду

3.1 Библиотека *BeautifulSoup*

Библиотека *BeautifulSoup* представља једну од једноставнијих за коришћење Пајтон библиотека за имплементацију прикупљања података из *HTML* и *XML* текстуалних докумената [8]. Ова библиотека је намењена искључиво за статички скрејпинг, што значи да игнорише Јаваскрипт (енг. *JavaScript*) и преузима веб-страницу са сервера без коришћења претраживача. За разлику од других библиотека које ће се касније разматрати, ова библиотека не може сама да приступи веб-страници и потребни су јој помоћни модули. Њена главна функција је парсирање *HTML* и *XML* датотека. Често се користи у комбинацији са пакетом *Selenium* [5], који омогућава дохватање динамичног дела веб-странице.

Инсталација

Библиотека *BeautifulSoup* се може инсталирати користећи *pip*, алат за инсталирање пакета за програмски језик Пајтон. Приказана у наставку наредба ће преузети и инсталирати најновију верзију пакета *BeautifulSoup*:

ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА

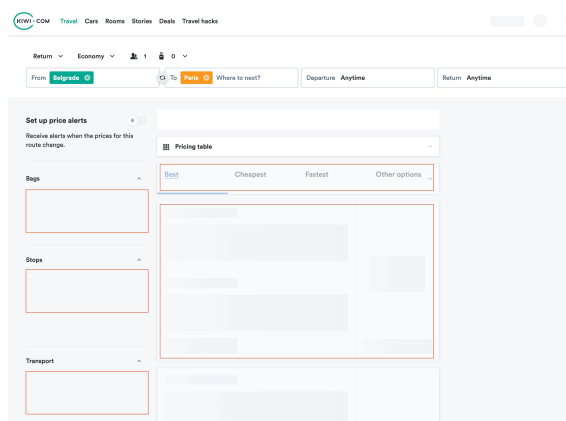
```
pip3 install bs4
```

Након што се инсталација заврши, библиотека се може укључити у Пајтон скрипту користећи следећу наредбу:

```
from bs4 import BeautifulSoup
```

Провера динамичности веб-странице

Коришћење библиотеке *BeautifulSoup* на примеру *kiwi* веб-странице је ограничено јер је страница динамична и захтева Јаваскрипт за учитавање. Овај захтев представља препреку за добијање конкретних података о летовима, што се може видети уз помоћ изгледа веб-странице са угашеним Јаваскриптом, који је приказан на слици 3.1. Како би се Јаваскрипт угасио за конкретан веб-сајт неопходно је угасити опцију коришћења Јаваскрипта унутар подешавања коришћеног претраживача, што је приказано на слици 3.2.

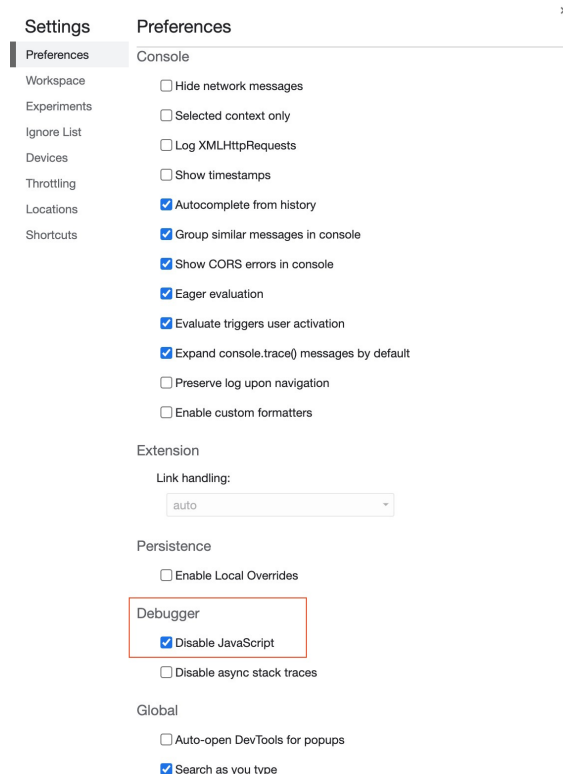


Слика 3.1: *kiwi* веб-сајт са угашеним Јаваскриптом

Прикупљање и парсирање *HTML*

Библиотека *BeautifulSoup* не представља самосталну библиотеку за прикупљање података са веб-страница. Уместо тога, она представља библиотеку која олакшава ефикасно извлачење информација из *HTML* документа. Пре тога, неопходно је да постоји *HTML* документ, који се може добити коришћењем Пајтон модула *requests* за слање *HTTP* захтева. У овом пакету постоји неколико метода који су значајни.

ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА



Слика 3.2: Подешавања Гугл кроум (енг. *Google Chrome*) претраживача

BeautifulSoap није библиотека за скрејповање података са Веба сама по себи. То је библиотека која омогућава да се ефикасно и лако извуче информација из HTML.

Дакле, за почетак, потребан је HTML документ и у ту сврху се може искористити Пајтонов пакет *requests* који омогућава слање HTTP захтева. Постоји неколико метода од значаја у овом пакету [7]:

- `get(url, params, args)`

Шаље HTTP GET захтев на наведену веб-адресу (енг. URL, Uniform Resource Locator))

- `post(url, data, json, args)`

Шаље HTTP POST захтев на наведену веб-адресу

- `put(url, data, args)`

Шаље HTTP PUT захтев на наведену веб-адресу

ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА

У наставку је приказан код који преузима веб-страницу. Треба имати у виду да при преузимању странице са Веба, може да се догоди да страница не буде пронађена на серверу (или је дошло до грешке у њеном преузимању) или да сервер није пронађен.

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 website = 'https://www.kiwi.com/en/search/results/belgrade-serbia/
    paris-france'
5 try:
6     response = requests.get(website)
7 except requests.exceptions.RequestException as err:
8     print("Error fetching page")
9     exit()
10
11 content = response.text
```

Даље, неопходно је испарсирати добијену HTML страницу тако што се креира *BeautifulSoup* конструктор који за параметре прихвата HTML страницу и парсер.

Пајтон нуди разне библиотеке за парсирање HTML, од којих су две најзаступљеније: *lxml* [3] и *html.parser* [2]. Сваки пројекат има различите захтеве на основу којих се може одабрати најпогоднији парсер, па је за потребе овог рада одабран *lxml* парсер из једног простог разлога - брзина. *lxml* је најбржи парсер HTML страница према званичној документацији *BeautifulSoup* библиотеке [8]. Једна од мана овог парсера јесте екстерна зависност, али у овом случају то не представља проблем.

Наведени код ниже креира *BeautifulSoup* објекат, који омогућава лак приступ многим корисним информацијама, као што је наслов странице, сви линкови на страници, текст са странице...

```
1 soup = BeautifulSoup(content, 'lxml')
2
3 print(soup.title)
4 # <title>Belgrade-Paris trips</title>
5
6 print(soup.find_all('a'))
7 # [<a class="Logo_Link-sc-1uwlkrd-6 kcwlYv" data-test="Logo" href="/
    en/?destination=paris-france&origin=belgrade-serbia">...]
8
```

```
9 print(soup.get_text())
10 # Belgrade Paris trips
11 # TravelCarsRoomsStoriesDealsTravel hacksloadingManage your trips, set
    up price alerts, use...
```

Циљање DOM елемената

У оквиру пакета BeautifulSoup постоје два метода која омогућавају једноставно и брзо филтрирање HTML странице за проналазак жељених информација [7]:

- `find(tag, attributes, recursive, text, keywords)`

проналази и враћа жељени елемент, где су аргументи редом: назив тага или листа тагова, Пајтон речник атрибута и одговарајућих тагова које садрже било који од тих атрибута, булова вредност која представља да ли се претрага врши по глобалним таговима или и по унутрашњим таговима, подударност на основу текстуалног садржаја ознака, атрибут на основу којих се бира таг

- `findAll(tag, attributes, recursive, text, limit, keywords)`

проналази и враћа листу жељених елемената, где су аргументи слични наведеним изнад са разликом што је додат аргумент лимит, који представља максималан број подударних елемената

Да би приказане информације биле занимљивије, HTML страница ће бити преузета коришћењем Selenium пакета из разлога што Selenium подржава Јаваскрипт. Детаљније о самом пакету Selenium може да се пронађе у наставку рада 3.2.

Одабрана је страница са летовима на релацији Београд-Малага <https://www.kiwi.com/en/search/results/belgrade-serbia/malaga-spain> и коришћењем Selenium пакета је направљен Пајтон скрипт који ће у одвојени фајл експортирати изворни HTML дате странице. У наставку је приказан код који најпре затвара прозор за *колачиће* (енг. cookies ¹) који се отвара кад се приступи веб-страници, затим чека 10 секунди да би се страница учитала

¹Колачић представља текстуалну датотеку која се чува у веб прегледачу док корисник прегледа неку веб-страницу.

ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА

и цео изворни HTML експортује у одвојени фајл. И овим је решен проблем нединамичности пакета BeautifulSoup.

```
1 import time
2
3 from selenium import webdriver
4 from selenium.common import exceptions
5 from selenium.webdriver.chrome.service import Service
6 from selenium.webdriver.common.by import By
7
8 website = 'https://www.kiwi.com/en/search/results/belgrade-serbia/
           malaga-spain'
9 path = '/usr/local/bin/chromedriver_mac64/chromedriver'
10 service = Service(executable_path=path)
11 driver = webdriver.Chrome(service=service)
12 driver.get(website)
13 driver.maximize_window()
14
15 try:
16     # Close cookies dialog
17     cookie_dialog_close_btn = driver.find_element(By.XPATH, value="//
section[@id='cookie_consent']//button[@aria-label='Close']")
18     cookie_dialog_close_btn.click()
19
20     time.sleep(10)
21
22     with open('flights-belgrade-malaga-page-source.txt', 'w') as file:
23         file.write(driver.page_source)
24
25     driver.quit()
26
27 except exceptions.StaleElementReferenceException as e:
28     print(e)
29     pass
```

У наставку ће бити приказан комплетан код који извлачи информације из необрађеног HTML који је претходно био експортован у фајл: датум поласка, информације о одласку, датум повратка, информације о повратку, број ноћи, цена лета.

Најпре је неопходно установити како извући целе блокове са информацијама о лету. Када се детаљно прегледа структура HTML, може се приметити

ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА

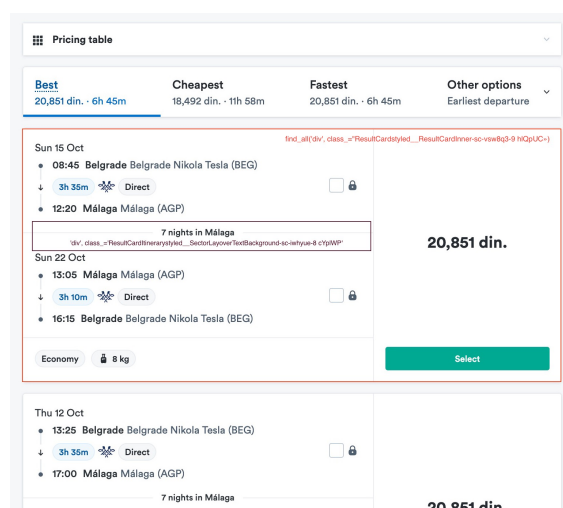
да је сваки лет издвојен у *div*² са дугачким именом класе што представља једино по чему се тај блок може идентификовати. Пошто на страници има више летова, најпре се користи функција која проналази све инстанце по потпуној подударности имена класе. На исти начин се извршио проналазак броја ноћи касније. 3.3.

```
1 all_flights = soup.find_all('div', class_="
    ResultCardStyled__ResultCardInner-sc-vsw8q3-9 h1QpUC")
```

Осим потпуне подударности по имену класе, елемент се може пронаћи по свом јединственом *id*³:

```
1 soup.find(id="specific_id")
```

Сада када су пронађени сви блокови са летовима, могу да се извуку информације о конкретно приказаном лету проласком кроз *for* петљу кроз дате инстанце.



Слика 3.3: Комплетно поклапање класа

```
1 import re
2
3 import pandas as pd
4 from bs4 import BeautifulSoup
5
6 with open('flights-belgrade-malaga-page-source.txt', 'r') as file:
7     soup = BeautifulSoup(file, 'lxml')
```

²Ознака `<div>` дефинише поделу или одељак у HTML документу.

³`id` атрибут се користи за одређивање јединственог идентификатора за HTML елемент.

ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА

```
8
9     all_flights = soup.find_all('div', class_="
ResultCardstyled__ResultCardInner-sc-vsw8q3-9 hlQpUC")
10
11     departure_date_regex = re.compile('.*DepartureDate.*')
12     departure_info_regex = re.compile('.*
ResultCardItineraryPlacestyled__StyledResultCardItineraryPlace.*')
13
14     flights = []
15     departure_dates = []
16     departure_info = []
17     return_dates = []
18     return_info = []
19     nights = []
20     prices = []
21
22     for flight in all_flights:
23         nights_spent = flight.find('div', class_='
ResultCardItinerarystyled__SectorLayoverTextBackground-sc-iwhyue-8
cYplWP').text.split(' ')[0]
24
25         dates = flight.find_all("p", {"class" : departure_date_regex})
26
27         departure_dates.append(dates[0].find('time').text)
28         return_dates.append(dates[1].find('time').text)
29
30         departure_time = flight.find_all('div', {"class" :
departure_info_regex})[0].find('time').text
31         departure_airport = flight.find_all('div', {"class" :
departure_info_regex})[0].find('div').text
32         departure_info.append(departure_time + ' ' + departure_airport
33         )
34
35         return_time = flight.find_all('div', {"class" :
departure_info_regex})[2].find('time').text
36         return_airport = flight.find_all('div', {"class" :
departure_info_regex})[2].find('div').text
37         return_info.append(return_time + ' ' + return_airport)
38
39         nights.append(nights_spent)
40         prices.append(flight.find('strong', {"data-test" : '

```

```
ResultCardPrice'}}).text)
41
42
43 df_flights = pd.DataFrame({'Departure date': departure_dates, '
Departure info': departure_info, 'Return date': return_dates, '
Return info': return_info, 'Nights' : nights, 'Prices': prices})
44 df_flights.to_csv('flights-belgrade-malaga-results.csv', index=
False)
```

Оно што треба приметити јесте да се блок са информацијама о одласку са веб-сајта идентификовао коришћењем поклапања по регексу (енг. regex).

Проблем прикупљања података са више страница

TODO: уколико буде превише страница, ова секција се може обрисати јер се пример морао радити коришћењем другог сајта.

У претходно наведеном примеру су вађени подаци из HTML једне веб-странице. Такође је могуће динамично преузети податке и то ће се приказати на примеру веб-страница које не зависе од Јаваскрипта, коришћењем BeautifulSoup и requests пакета. У питању је веб-страница која садржи транскрипте филмова. Да би се успешно извршило преузимање података кроз више страница, треба да се разуме на који је начин то постигнуто на конкретном веб-сајту.

На веб-адреси https://subslikescript.com/movies_letter-A уколико се изврши навигација на следећу страницу, може да се примети да се додаје и мења параметар *страница* (енг. page) у веб-адреси и онда она изгледа овако https://subslikescript.com/movies_letter-A?page=2. Наведени код ниже са претходно стеченим знањем, проналази блок са нумерацијом страница, извлачи укупан број страница, пролази кроз сваку од њих, поставља вредност изворног HTML на нову тако што креира нову веб-адресу и уз помоћ requests пакета извлачи HTML нове странице, креира нови BeautifulSoup објекат за парсирање, проналази све линкове ка филмовима на датој страници, прелази на конкретан линк и преузима транскрипт филма. Кључни моменат јесте да се сваки пут иницијализује BeautifulSoup објекат за парсирање.

```
1 import requests
2 from bs4 import BeautifulSoup
3
4 root = 'https://subslikescript.com'
```

ГЛАВА 3. ПРЕГЛЕД АЛАТА ЗА ПРИКУПЉАЊЕ ПОДАТАКА СА ВЕБ-СТРАНИЦА

```
5 website = f'{root}/movies_letter-A'
6 result = requests.get(website)
7 content = result.text
8 soup = BeautifulSoup(content, 'lxml')
9
10 # pagination
11 pagination = soup.find('ul', class_='pagination')
12 pages = pagination.find_all('li', class_='page-item')
13 last_page = pages[-2].text
14
15 links = []
16
17 for page in range(1, int(last_page) + 1):
18     result = requests.get(f'{website}?page={page}')
19     content = result.text
20     soup = BeautifulSoup(content, 'lxml')
21
22     box = soup.find('article', class_='main-article')
23
24     for link in box.find_all('a', href=True):
25         links.append(link['href'])
26
27     for link in links:
28         try:
29             result = requests.get(f'{root}/{link}')
30             content = result.text
31             soup = BeautifulSoup(content, 'lxml')
32             box = soup.find('article', class_='main-article')
33
34             title = box.find('h1').get_text()
35             transcript = box.find('div', class_='full-script').
get_text(strip=True, separator=' ')
36
37             print(link)
38             with open(f'movies/{title}.txt', 'w') as file:
39                 file.write(transcript)
40
41         except:
42             print('Link not working')
```


3.2 Библиотека Selenium

TODO: Инсталација, сајтови покренути са JS, драјвери, лоцирање елемената, пагинација, имплицитно и експлицитно чекање, попуњавање формулара, логин, кретање између страница, скрејповање странице са бесконачним скролом, како променити корисничког агента и зашто

3.3 Библиотека Scrapy

TODO: Инсталација, scrapy шаблони, креирање паука, стругање са више линкова, стругање са више страница, стругање АПИ, попуњавање формулара, логин, како променити корисничког агента, најосновније потребне функције LUA програмског језика (неопходно за SPLASH), SPLASH, шта представља SPLASH, зашто је неопходан, како превазићи Captcha)

Глава 4

Анализа резултата

TODO: Поређење перформанси, дијаграм односа времена стругања сајта и коришћене библиотеке, како оценити добијене резултате, табела са поређеним карактеристикама коришћених библиотека, препоруке, смернице за унапређење коришћених технологија

Глава 5

Закључак

TODO:

Библиографија

- [1] Keio) 1999 W3C® (MIT, INRIA. XPath. on-line at: <https://www.w3.org/TR/1999/REC-xpath-19991116/>.
- [2] Python Software Foundation 2001-2023. `html.parser` — Simple HTML and XHTML parser. on-line at: <https://docs.python.org/3/library/html.parser.html>.
- [3] Stefan Behnel and Martijn Faassen. Parsing XML and HTML with `lxml`. on-line at: <https://lxml.de/parsing.html>.
- [4] Osmar Castrillo-Fernández. Web scraping: Applications and tools.
- [5] 2023 Software Freedom Conservancy. Selenium. on-line at: <https://www.selenium.dev/documentation/>.
- [6] Gitiles. Headless Chromium. on-line at: <https://chromium.googlesource.com/chromium/src/+lkgr/headless/README.md>.
- [7] Ryan Mitchell. Web scraping with python. In *Web Scraping with Python*, 2015.
- [8] Leonard Richardson. BeautifulSoup Documentation, 2004-2023. on-line at: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>.

Биографија аутора

Зорана Гајић, рођена 04.11.1997 у Москви, где је ... TODO: