Web scraping-automatizirano prikupljanje podataka

Jerčinović, Kristijan

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski studij računarstva

Završni rad

WEB SCRAPING - AUTOMATIZIRANO PRIKUPLJANJE PODATAKA

Rijeka, srpanj 2023.

Kristijan Jerčinović
0069085356

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Preddiplomski studij računarstva

Završni rad

WEB SCRAPING - AUTOMATIZIRANO PRIKUPLJANJE PODATAKA

Mentor: izv.prof.dr.sc.Mladen Tomić

Rijeka, srpanj 2023.

Kristijan Jerčinović
0069085356

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 14. ožujka 2022.

Zavod: **Zavod za računarstvo**
Predmet: **Računalne mreže**
Grana: **2.09.02 informacijski sustavi**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Kristijan Jerčinović (0069085356)**
Studij: Preddiplomski sveučilišni studij računarstva

Zadatak: **Web scraping - automatizirano prikupljanje podataka / Web Scraping - Automated Data Collection**

Opis zadatka:

Proučiti i izložiti web scraping tehnike automatiziranog prikupljanja podataka. Objasniti područja primjene. Napraviti pregled karakteristika i mogućnosti postojećih komercijalnih i besplatnih alata. Koristeći programski jezik Python, napisati prilagođenu skriptu za prikupljanje podataka. Analizirati prikupljene podatke te prezentirati rezultate.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Kristijan Jerčinović

Zadatak uručen pristupniku: 14. ožujka 2022.

Mentor:

Izv. prof. dr. sc. Mladen Tomić

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Kristijan Lenac

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradio ovaj rad.

Rijeka, srpanj 2023.

Kristijan Jerčinović

Ime Prezime

Zahvala

Zahvaljujem mentoru izv.prof.dr.sc Mladenu Tomiću na korisnim savjetima i uputama što je trebalo popraviti kod izrade rada te u kojem smjeru bi trebao sam rad ići.

Sadržaj

Popis slika

Poglavlje 1

Uvod

Godine 1989. nakon što je izumljen World Wide Web koji je bio namijenjen svakome javlja se mogućnost objavljivanja vlastitog sadržaja na internet. Kako je u to vrijeme bilo malo internet stranica, tražilice su se oslanjale na ljudske administratore za prikupljanje i formatiranje poveznica. No, to se mijenja 1993. godine nakon izrade prve internet tražilice (JumpStation) temeljene na indeksiranju, od strane Jonathona Fletchera, koja se oslanjala na rad robota i koja je omogućila indeksiranje milijuna internet stranica, čime internet postaje ogromna platforma otvorenog koda.

Zbog toga dolazi do masovnog objavljivanja podataka što je rezultiralo u velikoj količini nestrukturiranih podataka na jednom mjestu, a kako JumpStation nije bio namijenjen prikupljanju i strukturiranju podataka, sljedeći korak je bio pronaći način kako prikupiti sve te podatke i strukturirati ih po mogućnosti automatski. Zbog toga počinje razvoj procesa zvanog web struganje (web scraping). Tako su godine 2000. stvorena prva web API (engl. Application Programming Interface) sučelja, odnosno sučelja za programiranje aplikacija, koja omogućavaju programerima pristup i preuzimanje nekih podataka dostupnih javnosti. To je bio početak razvoja web struganja. [1]

No, i dalje nismo odgovorili što je web struganje. Web struganje je proces automatskog ili ručnog prikupljanja i obrade velike količine podataka s interneta korištenjem određenih algoritama ili nekog programa. Osnovna ideja web struganja je: prikupljanje HTML (engl. Hypertext Markup Language) podataka s web domene,

parsiranje prikupljenih podataka s ciljem lociranja traženih informacija, spremanje traženih informacija u pogodan oblik te prelaženje na drugu web domenu i ponavljanje procesa (opcionalno). Ono također pruža mogućnost spremanja podataka u baze podataka te mogućnost analiziranja istih čime se mogu dobiti nova znanja o raznim djelatnostima.

Najčešće korišten programski jezik za web struganje je Python jer postoji nekoliko biblioteka i okvira koje omogućavaju lakši pristup prikupljanju podataka s interneta tako da ćemo za praktične svrhe koristiti upravo Python. Za statičke HTML web stranice koristit ćemo biblioteku Beautiful Soup koja ima najlakšu krivulju učenja, a što se tiče dinamičkih JavaScript web stranica koristit ćemo Selenium. Također, spomenut ćemo još neke druge biblioteke i alate za web struganje te njihove prednosti i nedostatke. [2]

Poglavlje 2

Princip rada Web struganja

Osnovni cilj web struganja, kao već spomenuto, je prikupiti informacije s različitih nestrukturiranih web stranica, strukturirati ih te pohraniti u nekom obliku za jednostavniju pretragu, poput proračunskih tablica, baza podataka ili datoteka s vrijednostima odvojenim zarezom (csv). Na prvu ruku bismo pomislili da je to sve potrebno ručno napraviti u obliku "copy-paste" (kopiraj i zalijepi) no u našem slučaju to se izvršava automatski od strane kompjuterskog agenta, zvanog web strugač, koji je baziran na algoritmima i kodu koji smo prethodno napisali. Taj agent izvršava iste operacije koje bi čovjek radio prilikom interakcije s web stranicom (slanje zahtjeva, traženje poveznice, traženje određenih dijelova na stranici...), no za razliku od čovjeka agent ne mora razmišljati tijekom izvođenja radnji čime je puno brži i efikasniji. U današnjem vremenu web struganje ide još dalje, u razvojnom smislu, pa se povezuje s korištenjem umjetne inteligencije čime se cijeli proces podiže na sasvim novu razinu. Tehnike umjetne inteligencije znatno ubrzavaju i olakšavaju proces web struganja. Također, omogućuje i razvoj drugih dijelova tehnologije, razvoj strojnog učenja pružajući mogućnost kvalitetnije izrade i konstantnog ažuriranja njihovih modela.

No, mora se paziti da se web struganje ne koristi bez pravila ograničenja zahtjeva. Inače bi traženi poslužitelj mogao smatrati da se radi o napadu uskraćivanja usluge (engl. Denial-of-service attack), zbog velike količine zahtjeva pokrenutih u kratkom vremenu.

Kako zapravo funkcionira web struganje? Prvi korak web struganja je odabir željene web stranice s koje želimo prikupiti podatke te koje podatke želimo prikupiti. Drugi je korak je analiziranje HTML strukture ciljane web stranice kako bismo pomoću HTML koda stranice i oznaka klasa, ID-ova i ostalih HTML atributa znali pomoću kojih oznaka možemo pristupiti kojim podacima. Sljedeći korak je konstruiranje HTTP zahtjeva korištenjem određenih programskih jezika, poput Python-ovog requests, PHP-ovog CURL ili JavaScript-ovog Fetch, kako bismo poslali HTTP zahtjev našoj stranici.

Sada slijedi proces izdvajanja željenih podataka koristeći se tehnikama HTML raščlanjivanja.To nam omogućavaju različite biblioteke, poput Beautiful Soup ili xml u Pythonu. One nam omogućavaju izvlačenje i analiziranje određenih elemenata. Nakon što smo izdvojili podatke potrebno ih je preoblikovati, odnosno moramo ukloniti neželjene i nepotrebne znakove kako bismo ih mogli strukturirati u format pogodan za daljnju analizu i upotrebu. Ako je riječ o dinamičkim stranicama koristimo se alatima koji omogućuju struganje istih (Selenium) ili alatima koji nemaju JavaScript podršku kombinirajući ih sa Seleniumom, koji omogućuje raščlanjivanje HTML koda iz dinamičkih stranica.

Za potrebe struganja podataka s više stranica ili više izvora možemo koristiti automatsko pregledavanje (engl. automated crawling) ili beskonačno listanje kako bismo iterirali kroz različite URL-ove ili kroz web stranice koje imaju više stranica. No, više o tome u sljedećim poglavljima. [3]

Poglavlje 3

Gdje se koristi web struganje?

Web struganje je tijekom proteklih godina sve češće korišteno od strane mnogih tvrtki, kompanija i trgovina. Razvojem tehnologije, poput modela strojnog učenja, umjetne inteligencije, razvoj internet trgovina, sve češća upotreba društvenih mreža, javlja se potreba za prilagodbom struganja s obzirom na današnje standarde. Zbog toga se pokušava usavršiti proces web struganja primjenom umjetne inteligencije kako bi se u potpunosti automatiziralo prikupljanje podataka s interneta čime bi se znatno ubrzao proces struganja, smanjili korišteni resursi te olakšao rad korisnicima.

Web struganje ima široku primjenu u svim današnjim tvrtkama, kako državnim tako i privatnim te raznim institucijama. Ponajprije služi za struganje velike količine informacija te potom objavljivanje tih informacija drugim institucijama.

3.1 Umjetna inteligencija i web struganje

Razvoj umjetne inteligencije je jedna od ključnih stvari i za web struganje. Ona omogućava potpunu automatizaciju struganja pomoću algoritama umjetne inteligencije. Ti algoritmi omogućavaju navigaciju kroz samu web stranicu te automatsko izdvajanje istih. Također, u inteligentnom indeksiranju pomoću tehnika pojačanog učenja nudi se mogućnost puno bržeg i učinkovitijeg kretanja kroz samu stranicu. Uz to imamo i korisne tehnike obrade prirodnog jezika (NPL), računalnog vida i prepoznavanje uzoraka koje omogućavaju lakše prikupljanje podataka s HTML, XML

stranica.

No, što je s dinamičkim stranicama pisanih u JavaScriptu ili AJAX-u. Za alate koji koriste umjetnu inteligenciju to ne predstavlja problem jer mogu pokrenuti web stranice u pregledniku bez sučelja ili koristiti tehniku dinamičkog raščlanjivanja HTML -a. Također, umjetna inteligencija može koristiti tehnike rotacije IP adresa kako bi zaobišla mjere protiv struganja.

Na primjer, ChatGPT je robot koji koristi tehnike umjetne inteligencije za struganje interneta. Izrazito učinkovit i efikasan alat koji daje odgovore na gotovo sva pitanja uz uvjet da može pronaći stranice s kojih će prikupiti podatke. No, s druge strane nije uvijek pouzdan jer ne možemo biti u potpunosti sigurni da li su informacije točne. [4]

3.2 Strojno učenje

Strojno učenje je proces učenja računala, odnosno tehnika koja omogućuje računalu da uči na sličan način kao i ljudi. Odnosno, strojno učenje omogućava računalu da samo poboljšava svoj rad i ažurira informacije kojima upravlja tako što prikuplja najnovije podatke iz područja za koje je model strojnog učenja izrađen. Kako vidimo model strojnog učenja mora nekako prikupiti podatke da bi se mogao prilagoditi svakodnevnim standardima i zahtjevima. Tu nastupa web struganje.

Struganjem raznih sadržaja, poput postova na društvenim mrežama, tekstualnih podataka iz novinskih članaka, rasprava na forumima, možemo izgraditi model obrade prirodnog jezika (NLP, Neurolingvističko programiranje). Zatim, struganje slika, video i audio podataka omogućuje izradu modela za računalni vid ili obradu zvuka kao i modele za klasifikaciju i kategorizaciju istih. Nadalje, struganje se može koristiti za izvršavanje nekih zadataka strojnog učenja kao na primjer, kod e-trgovina kako bi preko prikupljenih recenzija i opisa nudila određene proizvode ili ih stavila na prvo mjesto pretrage ili kod tražilica koje bi imale poboljšani sistem pretraživanja i rangiranja traženih stranica.

Tako, na primjer, Google pomoću strojnog učenja poboljšava preciznost rezultata traženja, Facebook prikazuje postove ovisno o našim interesima, samovozeći automo-

bili prate objekte iz okruženja i koriste te podatke da bi poboljšali svoje sposobnosti u vožnji.

Umjetna inteligencija s web struganjem upotpunjuje model strojnog učenja do maksimuma jer omogućava da se model samostalno ažurira čime se model razvija i prilagođava paralelno s razvojem vanjskog svijeta. Naravno, sa što više izvora stružemo podatke i uspoređujemo ih to nam se model strojnog učenja poboljšava i generalizira čime se stvara izrazito kvalitetan i pouzdan model.

No, važno je voditi računa o pravnim i etičkim aspektima svake web stranice, poštivajući zakone o autorskim pravima te pridržavajući se uvjeta pružanja usluga stranice. [5]

3.3 Struganje društvenih mreža

Prvo bismo pomislili zašto bismo uopće htjeli strugati podatke s društvenih mreža. No, ono može biti dosta korisno, ali pritom treba paziti jer može imati zakonska ograničenja.

Pomoću web struganja možemo uzeti podatke o korisničkim profilima kao što su imena, biografije, lokacije i druge, koji nam mogu poslužiti u stvaranju sustava nekakvih preporuka. Također, to možemo primijeniti i na vlastitoj društvenoj mreži ako želimo izraditi neku. Struganje nam može poslužiti da prikupimo podatke od korisnika te na osnovu prikupljenog preporučimo istima stranice za pratiti ili neke osobe koje možda poznaju.

Nadalje, ako nas zanimaju nekakva istraživanja za tržište, praćenje robnih marki, mišljenja javnosti o određenim proizvodima ili samo praćenje trendova, možemo prikupiti razne objave, rasprave te komentare vezane za određene teme.

Web struganje može poslužiti i kao analiza zbog određene konkurencije na društvenim mrežama. Drugim riječima može nam pružiti uvid u aktivnosti na ostalim društvenim mrežama kako i objavama, uvid u ocjene i koji dijelovi mreže se korisnicima naviše sviđaju kako bismo mogli i vlastitu mrežu poboljšati i pokušati biti iznad konkurenata.

3.4 Ostale primjene

Također, web struganje se koristi od strane raznih tvrtki te za industrijske i marketinške svrhe. Mnogi agenti za nekretnine koriste struganje za dohvaćanje dostupnih nekretnina za prodaju ili iznajmljivanje. Tako stvaraju popise nekretnina koje objavljuju na svojim stranicama. Struganje može biti korisno i prilikom uspoređivanja cijena na tržištu, odnosno uspoređivanju cijena istih proizvoda ili općenito proizvoda u različitim e-trgovinama kako bismo kreirali listu određenih proizvoda ili konkurirali drugim trgovačkim lancima. Struganje omogućava spremanje podataka u bazu podataka ili neki drugi oblik poput proračunskih tablica pa pruža mogućnost da se svi ti navedeni podaci, ako nama nisu potrebni za korištenje, proslijede drugim tvrtkama. Nadalje, možemo prikupiti podatke o vremenskim prognozama za određene lokacije, najnovijim vijestima iz cijelog svijeta.

Kako smo već spomenuli korisnost umjetne inteligencije u području web struganja, tako nam i ovdje može znatno olakšati posao. Koristeći se njome možemo konstantno biti ažurni zbog čega nećemo zaostajati u odnosu na druge velike tvrtke te možemo brže dohvatiti i usporediti podatke s više izvora za kvalitetniju izgradnju liste podataka koja nam je u cilju. [6]

Poglavlje 4

Tehnike web struganja

Do sada smo naučili da web struganje može biti automatsko i ručno. Od ručnog imamo samo tehniku zvanu "copy-paste" (kopiraj i zalijepi), a automatsko dijelimo na više vrsta koje ćemo sada navesti. Razvojem tehnologije razvija se i automatsko web struganje kako bi se prilagodilo novim područjima. Također, razvojem tehnologije povećava se primjena i korisnost samog struganja.

Naravno, kod korištenja bilo koje tehnike web struganja moramo obratiti pozornost na pravila i uvjete pružanja usluga koje svaka web stranica ima. Treba paziti i na određenu količinu odgode prilikom implementacije da se server ne pretrpa sa zahtjevima zbog čega bi nas mogao blokirati i zabraniti nam pristup stranici.

4.1 Copy-paste

Kopiraj i zalijepi (engl. copy-paste) metoda je najstarija i najjednostavnija metoda web struganja kojom se u današnje vrijeme gotovo nitko više ne koristi. Metoda se služi kako i samo ime kaže doslovnim kopiranjem sadržaja s interneta te spremanjem ("paste") istog u vlastitu bazu podataka. Metoda je u neku ruku dobra jer izvođač ima direktan uvid u stranicu i kopira samo one dijelove koji su mu potrebni te je spremanje podataka vrlo jednostavno. Za tu tehniku nije potrebno nikakvo znanje kodiranja. Ona također omogućava lagano zaobilaženje obrane web stranice od robota. No, ta metoda nije učinkovita jer njezin proces može biti dugotrajan i zamoran

kada korisnik treba analizirati i pohraniti velike količine podataka. [7]

4.2 Pregled koda Hypertext Markup jezika (HTML)

Većina primjene web struganja uključuje raščlanjivanje HTML strukture, identificiranje željenih elemenata te izdvajanje relevantnih informacija za daljnju analizu i korištenje. Često korištene biblioteke za raščlanjivanje HTML-a uključuju Beautiful Soup i xml koje koriste programski jezik Python te jsoup koji koristi Javu. HTML struganje ponajprije zahtjeva poznavanje HTML-a i poznavanje njegovih tagova. Željeni elementi mogu se locirati pomoću CSS selektora, XPath izraza ili običnih izraza ovisno o složenosti HTML strukture stranice. CSS selektori nude mogućnost odabira HTML elemenata stranice preko njihovih oznaka, klasa, ID-ova ili drugih atributa. XPath je upitni jezik koji omogućava navigaciju kroz elemente u XML ili HTML dokumentu.

Nakon pronalaska elemenata, slijedi njihovo izdvajanje. Izdvajanje se može postići preko dohvaćanja tekstualnog sadržaja unutar HTML elementa, dohvaćanja određenih vrijednosti atributa kao što su "src" za slike ili "href" za linkove, raščlanjivanjem HTML tablica i izdvajanje tabličnih podataka u strukturirane formate i drugim načinima.

Problem nastaje kod modernih, dinamičkih web stranica koje je potrebno učitati preko JavaScripta ili AJAX poziva. Za struganje takvog sadržaja potrebno je koristiti preglednik bez sučelja ili Selenium kako bismo dohvatili HTML kod.

Također, mnoge stranice mogu imati sadržaj na više stranica (engl. pages) zbog čega je potrebno listati kroz stranice web stranice. Ta metoda zove se paginacija i omogućava prolaženje kroz različite stranice mijenjajući parametre url-a od web stranice. Ona nam omogućava da izdvojimo samo potrebne informacije sa svake stranice. [7]

4.3 Dohvaćanje Hypertext Transfer Protokola

Dohvaćanje HTTP-a je proces interakcije s web poslužiteljem i razmjene podataka putem interneta korištenjem HTTP protokola. Temelji se na slanju zahtjeva web poslužitelju te dohvaćanju HTML sadržaja web stranice. Tehnika se koristi modelom zahtjev-odgovor, odnosno klijent šalje zahtjev poslužitelju (najčešće GET metodom) koji odgovara traženim podacima.

Najpopularniji programski jezici koji koriste ovu tehniku su Python, s bibliotekama "requests" ili Scrapy te JavaScript s bibliotekom Puppeteer. Najbolje da to pogledamo na nekom primjeru. Prvo instalirajmo biblioteku "requests".

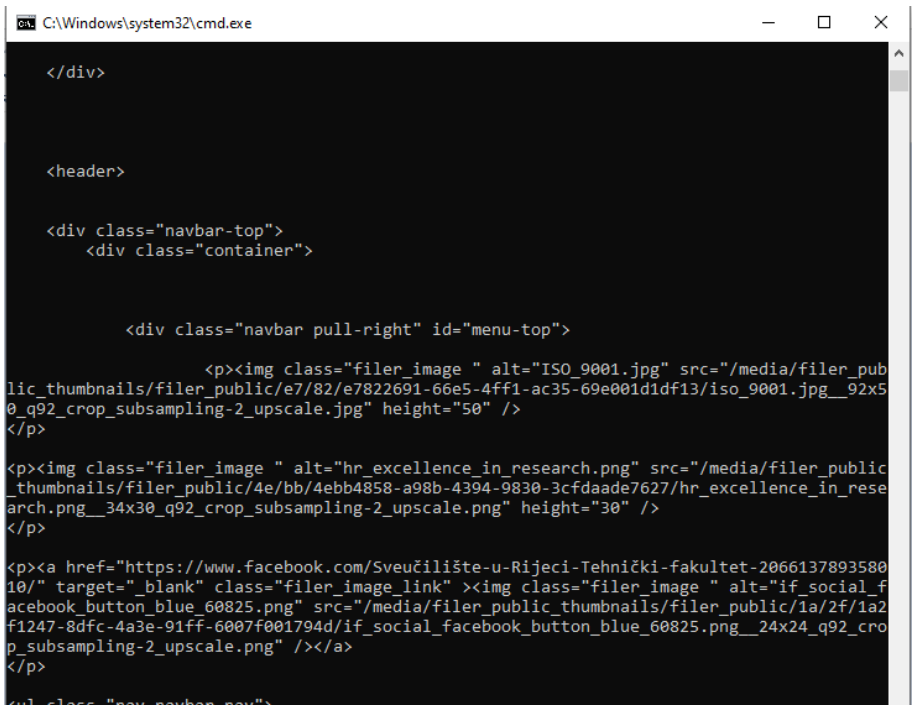
```
1    pip install requests
```

Listing 4.1 Instalacija potrebne biblioteke za zahtjeve

Sljedeći korak je odabrati stranicu čijem poslužitelju šaljemo zahtjev. U našem primjeru neka to bude stranica Tehničkog fakulteta. Zahtjev šaljemo naredbom "get" na ciljanu stranicu. Poslužitelj nakon što dobije zahtjev, procesuira ga te potom šalje odgovor (engl. response). Nakon što poslužitelj pošalje stranicu može se krenuti sa struganjem podataka.

```
1    import requests
2    url = 'http://www.riteh.uniri.hr/'
3    response = requests.get(url)
4
5    print(response.text)
```

Listing 4.2 Slanje zahtjeva naredbom "get"



Slika 4.1 HTML kod stranice Tehničkog fakulteta

Sada se koristimo prethodnom tehnikom pregleda HTML koda te prelazimo na raščlanjivanje HTML-a pomoću biblioteke "HTML parser", korištenjem Beautiful Soup. Dohvatimo samo sve "div" tagove:

```
1  import requests
2  from bs4 import BeautifulSoup
3
4  url = 'http://www.riteh.uniri.hr/'
5  response = requests.get(url)
6
7  soup = BeautifulSoup(response.text, 'html.parser')
8  tagovi = soup.find_all('div')
9  print(tagovi)
```

Listing 4.3 HTTP dohvaćanje

```
Banka: Erste &amp;amp; Steiermärkische Bank d.d.</p>
<p><a class="btn btn-primary" href="/kontakti">Popis djelatnika</a></p>
</div>, <div id="copyright">
<div class="container">
<p>© Riteh 2019., Sva prava pridržana</p>
<p>          </p>
<p><a class="" href="/media/filer_public/46/74/46743b31-bfa3-472d-a6b0-e8d9b3058545/i
o pristupačnosti</a></p>
<p>Programiranje: RitehWebTeam</p>
</div>
</div>, <div class="container">
<p>© Riteh 2019., Sva prava pridržana</p>
<p>          </p>
<p><a class="" href="/media/filer_public/46/74/46743b31-bfa3-472d-a6b0-e8d9b3058545/i
o pristupačnosti</a></p>
<p>Programiranje: RitehWebTeam</p>
</div>]
Press any key to continue . . .
```

Slika 4.2 Div tagovi stranice Tehničkog fakulteta

Kako vidimo unutar "div" tagova imamo "p" tagove koje možemo naknadno dohvatiti daljnjom implementacijom koda.

Ako su podaci na većem broju stranica ili zahtijevaju označavanje stranica potrebno je uključiti slanje dodatnih zahtjeva za njihovo dohvaćanje. Također, HTTP zahtjev sam nije dovoljan za učitavanje sadržaja dinamičkih stranica zbog čega je potrebno kao i kod pregleda HTML-a koristiti preglednik bez sučelja, Puppeteer ili Selenium. [7]

4.4 Pristup API-u

API pristup (engl. Application Programming Interface access) ili pristup sučelju za programiranje aplikacija koristi struganje za dohvaćanje podataka s web usluga ili API-ja, odnosno nije potrebno struganje HTML sadržaja već se može direktno poslati zahtjev API-ju stranice za izravno dohvaćanje podataka.

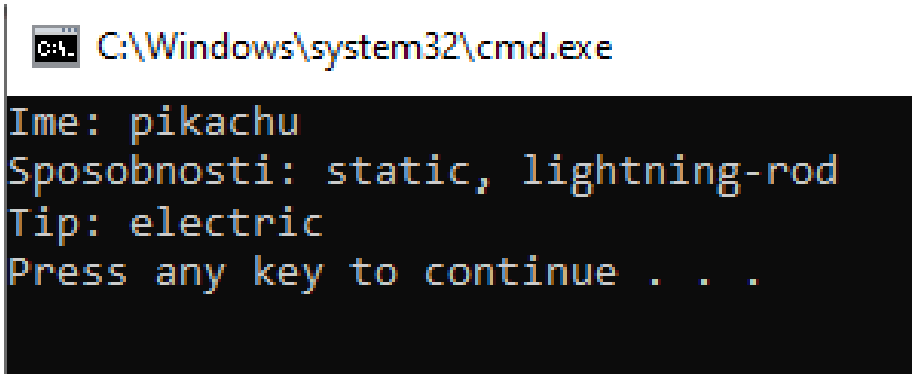
Potrebno je odrediti API koji nam je u cilju za izdvajanje podataka. Zatim, je nužno osigurati API ključ zbog provjere autentičnosti. Za neke stranice bit će nužno napraviti račun te se prijaviti u njega kako bismo dobili API ključ. Nadalje, moramo razumjeti parametre i ostale dijelove koje pruža API kako bismo konstruirali zahtjeve te odabrati biblioteku za slanje zahtjeva ("requests" u Pythonu ili Axios u JavaScriptu). Za slanje zahtjeva koristi se najčešće HTTP tehnika naredbom "get" ili "post" nakon čega dobivamo odgovor od strane API-ja koji može biti u različitim

formatima, poput JSON, XML ili CSV, koje je potom potrebno raščlaniti te spremiti u željeni oblik. Također, ne smijemo prekršiti pravila pružanja usluga koja API nudi. [8]

Uzmimo primjer API pristupa. Radi jednostavnosti uzet ćemo stranicu koja ne zahtijeva API ključ. Ako zahtjeva ključ samo je potrebno napraviti račun i dobiti ključ kao što su Twitterov ili Amazonov API. Stranica na kojoj ćemo demonstrirati API pristup je "Poke API", gdje možete ako ste ljubitelj Pokemona saznati nešto o sposobnostima, napadima i ostalim pojedinostima o svakom Pokemonu.

```
1    import requests
2
3    url = "https://pokeapi.co/api/v2/pokemon/pikachu"
4    response = requests.get(url)
5
6    #provjera ispravnosti zahtjeva
7    if response.status_code == 200:
8        #izdvajanje JSON sadrzaja
9        json_data = response.json()
10
11       #izdvajanje podataka iz JSON strukture
12       name = json_data["name"]
13       abilities = [ability["ability"]["name"] for ability in
     json_data["abilities"]]
14       types = [type_info["type"]["name"] for type_info in
     json_data["types"]]
15
16       print("Ime:", name)
17       print("Sposobnosti:", ", ".join(abilities))
18       print("Tip:", ", ".join(types))
19   else:
20       #u slucaju greske
21       print("Error:", response.status_code)
```

Listing 4.4 Kod za API pristup



Slika 4.3 Podaci o Pokemonu Pikachu

4.5 Proxy rotacija

Proxy rotacija je postupak maskiranja IP adrese kako bi se povećala anonimnost i omogućilo zaobilaženje ograničenja i zabrana od strane nekih stranica. Korisna tehnika u web struganju jer pospješuje učinkovito i neprekidno izdvajanje podataka. Na primjer, neke stranice mogu zabraniti pristup ako utvrde često slanje zahtjeva s jedne IP adrese (posumnjaju na struganje). Proxy nudi mogućnost mijenjanja IP adrese što stranicama otežava otkrivanje i blokiranje zahtjeva koje uključuju struganje.

Kako bismo implementirali proxy rotaciju unutar web struganja treba nam skup proxyja kojeg možemo dobiti od proxy pružatelja usluga ili sami izraditi vlastitu proxy strukturu. On bi se trebao sastojati od različitih IP adresa s različitim lokacijama radi veće učinkovitosti. Neke biblioteke pružaju funkcije koje omogućuju rukovanje proxy rotacijom, ali možemo ga i sami integrirati u kod tako da naš strugač šalje zahtjeve putem proxyja i izvršava određeno implementiranu proxy rotaciju, na primjer napravimo da se preusmjeri na drugi proxy svakih nekoliko minuta ili sati čime se IP adresa redovito mijenja.

No, treba konstantno voditi računa o uspješnosti proxy strategije jer uvijek postoji mogućnost da dođe do blokiranja ili neuspješnog struganja. Tada je potrebno promijeniti proxy skup ili prilagoditi proxy strategiju. [9]

Poglavlje 5

Alati za web struganje

5.1 Beautiful Soup

Beautiful Soup ja biblioteka koja omogućava jednostavno raščlanjivanje HTML i XML datoteka. Nudi jednostavno sučelje za rad s HTML elementima stranice. Beautiful Soup koristi funkcije poput find(), find_all() i select() kojima možemo odabrati grupe elemenata s obzirom na njihove oznake, atribute ili nazive klasa. Nakon što smo pronašli elemente koje želimo izdvojiti, korištenjem svojstava metoda .text(), .get() ili .string() možemo dohvatiti tekstualni sadržaj i vrijednosti atributa pojedinog elementa te možemo očistiti podatke od neželjenih znakova kako bismo ih mogli pohraniti u nama pogodnom obliku. Što se tiče složenosti, Beautiful Soup je najjednostavniji i najbolji alat za početnike i svatko tko bi se htio upoznati s web struganjem trebao bi započeti njime te kasnije prijeći na složenije alate poput Scrapyja ili Seleniuma. Zahtjeva samo poznavanje HTML-a i CSS-a, a funkcije koje nudi su jednostavne za naučiti. Međutim, Beautiful Soup nije pogodan za struganje velikih količina podataka zbog spore obrade podataka te ne podržava struganje dinamičkih web stranica pisanih u JavaScript-u zbog čega se mora koristiti sa Selenium-om ako bismo htjeli i dalje koristiti Beautiful Soup.

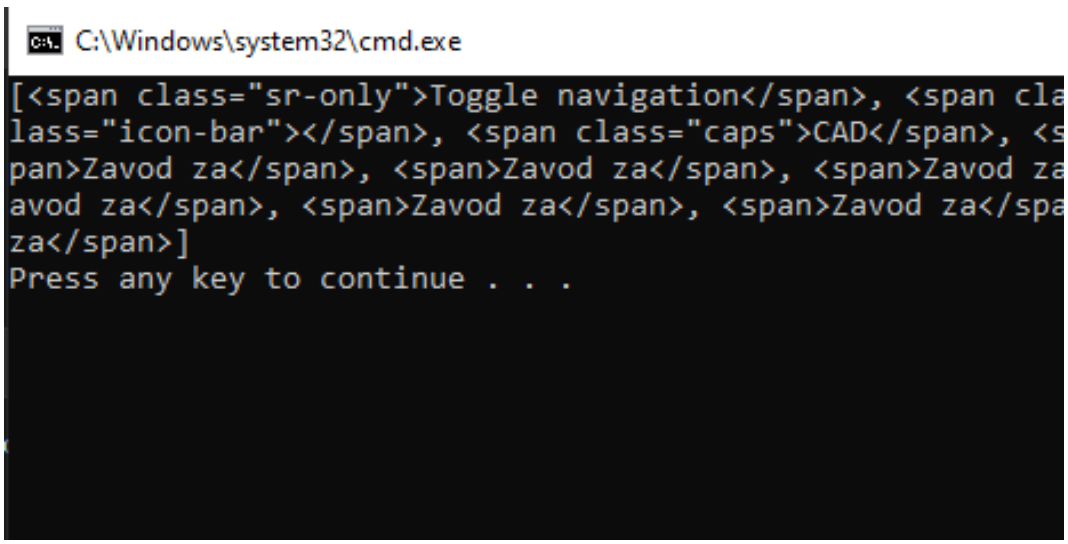
Ponajprije ga koriste tvrtke za e-trgovine za dohvaćanje informacija o cijenama, recenzijama i detaljima proizvoda, medijske tvrtke za izdvajanje novinskih članaka te podataka o vijestima te istraživačke organizacije za prikupljanje podataka o znans-

tvenim radovima i bazama podataka raznih istraživačkih područja. [10]

Pokažimo na jednostavnom primjeru. Potrebno je instalirati biblioteke requests te uvesti samu Beautiful Soup biblioteku kao što vidimo u prvim linijama koda. Sljedeće, jednostavnom naredbom "get" šaljemo zahtjev poslužitelju te potom kreiramo "soup" objekt pomoću kojeg raščlanjujemo HTML kod metodom "find_all" te naposljetku printamo podatke. U poglavlju "Praktični dio koda" pokazat ćemo detaljniju primjenu Beautiful Soup biblioteke te načine pohrane podataka.

```
1  import requests
2  from bs4 import BeautifulSoup
3
4  url = requests.get('http://www.riteh.uniri.hr/').text
5  soup = BeautifulSoup(url, 'html.parser')
6
7  info = soup.find_all('span')
8  print(info)
```

Listing 5.1 Primjer koda pisanog pomoću Beautifu Soup biblioteke



Slika 5.1 Svi "span" tagovi sa stranice Tehničkog fakulteta

Prednosti Beautiful Soup-a naspram drugim alatima je to što ima izrazito jednostavno sučelje za korištenje zbog čega je pogodan za početnike. Zbog toga ga svatko

može isprobati i razvijati vještinu struganja bez potrebe za velikim znanjem programiranja. Pogodan je za jednostavne zadatke i struganje male količine podataka zbog jednostavnosti koda. Alat je otovorenog koda. Jedini nedostaci su mu to što ima izrazito sporu obradu podataka i nije preporučljiv za izvršavanje složenijih zadataka.

Primjer tvrtki koje koriste Beautiful Soup su BuzzFeed i ScrapingBee. BuzzFeed je tvrtka za digitalne medije te koristi struganje za prikupljanje raznih novosti, sadržaja te za analizu trendova kako bi im omogućio pružanje svojoj publici ažurnog i zanimljivog sadržaja.

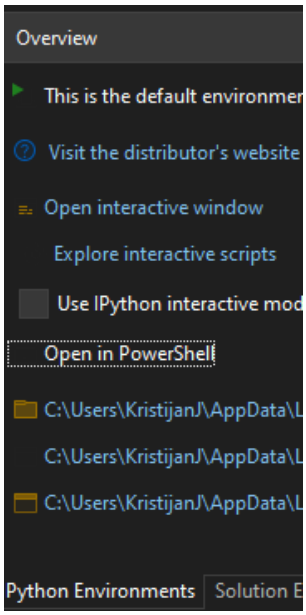
5.2 Scrapy

Scrapy sadrži mnogo funkcija indeksiranja weba, ekstrakcije podataka te obrade, zbog čega je jedan od najmoćnijih i najfleksibilnijih alata web struganja. Za razliku od Beautiful Soup-a ima puno veću brzinu obrade podataka, stoga ako nam je za cilj struganje velikih količina podataka koristit ćemo Scrapy.

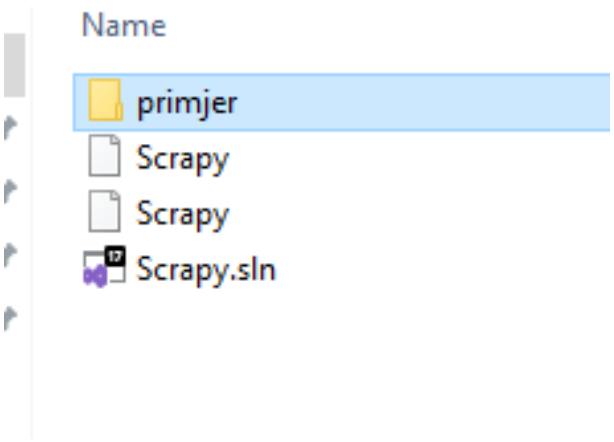
Scrapy je jedinstven po tome što se u kodu koriste tzv. spider klase kojima definiramo kako ćemo strugati stranicu. One određuju url-ove s kojih ćemo strugati te podatke koje namjeravamo strugati. Klasa spider (scrapy.Spider) mora sadržavati atribut 'name' (ime) koje mora biti jedinstveno kako bi se referenciralo na njega unutar projekta, 'start_urls' da spider zna s koje stranice izvršava struganje, metodu 'parse' za procesuiranje odgovora dobivenog od ciljanog URL-a, selektore za prikupljanje HTML, XML ili CSS tagova, atributa i ostalog, 'yield' iteme (iteme za prinos) koji predstavljaju prikupljene podatke koje procesuiramo i spremamo kasnije. Klasa spider podržava paginaciju odnosno može navigirati kroz druge stranice na web stranici, ako ih ima, koristeći 'parse' metodu i 'yield' zahtjev za sljedeću stranicu. [10]

Scrapyjem se služe tvrtke koje izrađuju tražilice ili koje zahtijevaju složene procese struganja te najviše e-trgovinske tvrtke za analizu konkurencije i praćenje cijena. Može se kombinirati s tehnikama umjetne inteligencije te je pogodan za stvaranje modela strojnog učenja zbog mogućnosti dohvaćanja velike količine podataka u kratkom vremenu.

Prvo je potrebno instalirati Scrapy naredbom "pip install scrapy". Slijedi kreiranje novog Python projekta "Scrapy" unutar kojeg kreiramo "New file" i napravimo još jednu Python datoteku koju ćemo nazvati "primjer". No, "primjer kreiramo koristeći se Pythonovom "PowerShell" naredbom "scrapy startproject imeProjekta".



Slika 5.2 Otvaranje PowerShella



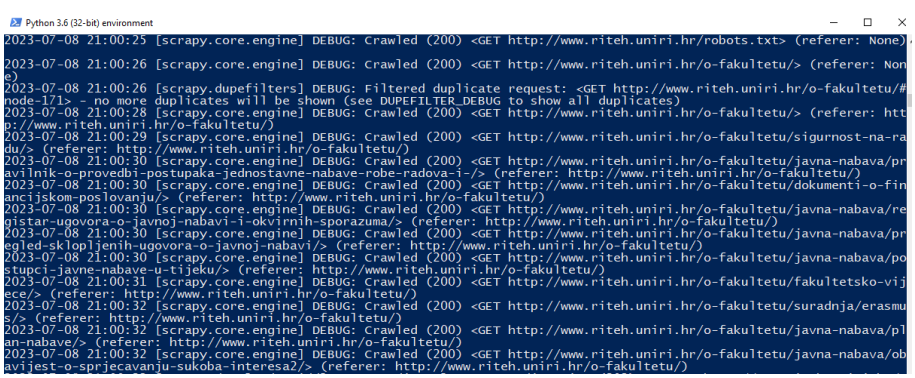
Slika 5.3 Kreiranje scrapy datoteka.

Slijedi pisanje koda. Njega ćemo smjestiti unutar našeg primjer

```
    from scrapy.spiders import CrawlSpider, Rule
    from scrapy.linkextractors import LinkExtractor

    class CrawlingSpider(CrawlSpider):
        name = "crawler"

        allow_domains = ["uniri.hr"]
        start_urls = ["http://www.riteh.uniri.hr/o-fakultetu/"]

        rules = (
            Rule(LinkExtractor(allow="/o-fakultetu/")),    )
```

Listing 5.2 Primjer koda pisanog pomoću BeautifuSoup biblioteke

Prvo uvodimo "CrawlSpider" pomoću kojeg definiramo klasu spidera te "Rule" koji određuje koje stranice su nam u cilju. Potom "linkextractors" za dohvaćanje url-a stranica. Definiramo klasu spider te ju imenujemo. Potom koristimo naredbe "allow_domains" nam omogućuje da samo ciljamo sve stranice koje sadrže "uniri.hr" unutar svojih linkova i "start_urls" koji sa "Rule" naredbom ograničava odabir stranica s kojih stružemo (radi jednostavnosti, uzet ćemo stranicu Tehničkog fakulteta Rijeka), odnosno omogućava struganje stranice unutar "uniri.hr" domene koje imaju nastavak "/o-fakultetu/". Da smo, na primjer, ciljali neke e-trgovine onda bismo se mogli ograničiti na neke određene proizvode. Npr. "/https://www.sanctadomenica.hr/bijela-tehnika.html" čime bismo se ograničili na struganje podataka vezanih za kućanske uređaje. Nakon napisanog koda vraćamo se u PowerShell i koristimo naredbu "scrapy crawl imeSpidera", odnosno u našem slučaju "crawler" pri čemu moramo pripaziti da se nalazimo unutar direktorija u kojem se nalazi projekt.



Slika 5.4 Podaci prikupljeni korištenjem Scrapyja.

Scrapy je alat koji nije pogodan za početnike jer uključuje dosta znanja i snalažljivosti u samom kodu. Na ovako jednostavnom primjeru se može vidjeti zahtjevnost te velika mogućnost sitnih pogrešaka koje nam se mogu pojaviti unutar koda. Preporučljivo bi bilo da se svatko tko ima namjeru koristiti Scrapy dobro upozna i uvježba na Beautiful Soup biblioteci kako bi stekao brži način prilagodbe i lakše razumio sam princip rada Scrapyja.

Prednosti Scrapyja su te što ima izrazito brzu obradu podataka, lagano se prilagođava i proširuje te ima ugrađenu podršku za web protokole. Alat otvorenog koda je što znači da ga uvijek možete isprobati i uvježbavati kod kuće. S druge strane, ima limitiranu podršku za dinamičke web stranice zbog čega bismo morali prijeći na korištenje Seleniuma ako nam je u cilju struganje istih.

5.3 Selenium

Selenium je alat koji omogućuje programerima interakciju s web preglednicima, odnosno možemo oponašati interakcije poput pritiskanja linkova, gumbova, ispunjavanje sadržaja, slanje podataka te navigacija kroz preglednik. Selenium podržava gotovo sve poznate preglednike od Chrome-a, Firefox-a, Safari-ja do Edge-a. Nudi WebDriver API koji omogućava kodiranje automatiziranog upravljanja preglednikom što je korisno jer napisani kod možemo primijeniti i na drugim preglednicima uz male izmjene. Također, ima sposobnost pokretanja web preglednika bez sučelja (engl. headless mode), odnosno mogućnost rada preglednika bez vidljivog sučelja što je korisno u web struganju gdje nije potrebno vizualno prikazivanje stranice. Jedan je od najjačih alata web struganja jer podržava u potpunosti JavaScript te nema nikakvih poteškoća kod rukovanja s dinamičkim stranicama. Uz JavaScript podržava i ostale jezike kao što su Python, Ruby, Java i C#. Podržava sve načine lociranja elemenata kao i prethodno spomenuti alati, poput dohvaćanja ID-a, klasa, CSS selektora, XPath-a i druge. Jedini nedostaci Seleniuma su spora obrada podataka, ograničena podrška protiv robota što može znatno otežavati automatizaciju, korištenje velike količine resursa te slaba podrška za mobilne aplikacije.

Mnoge tvrtke poput Facebook-a, Twitter-a, Microsoft-a, Amazon-a i PayPal-a

koriste upravo Selenium. Facebook, Twitter i Microsoft pomoću Selenium-a testiraju svoje usluge i web aplikacije kako bi im poboljšali kvalitetu i performanse. PayPal-u je od koristi za bolji rad transakcija i bolje korisničko iskustvo te za provjeru utjecaja ažuriranja na trenutno stanje, odnosno da li se ažuriranjem javljaju neke nove greške i problemi. [10]

Primjer koda i objašnjenje pojedinih dijelova koda razradit ćemo u poglavlju "Praktični dio koda".

5.4 Puppeteer

Puppeteer pruža API visoke razine za upravljanje web preglednicima. Nudi pokretanje preglednika bez sučelja (Chrome-a ili Chromium-a) što znači da radi bez vidljivog sučelja. Takav način rada znatno ubrzava proces struganja te troši manje resursa. No, podržava i pokretanje preglednika s vidljivim sučeljem prilikom uklanjanja grešaka i provjere automatiziranog rada. Uključuje navigaciju do web stranice, pritiskanje na elemente te ispunjavanje obrazaca ili unošenje tekstualnog sadržaja kod stranica koje sadrže tražilicu. Puppeteer nudi API za interakciju s preglednikom te mogućnost manipuliranja s web stranicama, odnosno izmjene sadržaja elemenata. Ima potpunu JavaScript podršku zbog čega može samostalno, bez kombiniranja s drugim alatima, strugati sadržaj s dinamičkih stranica. Još jedna karakteristika Puppeteer-a je to da može modificirati i lažirati zahtjeve te omogućava simuliranje različitih mrežnih uvjeta što je korisno kod struganja kako bismo zaobišli blokiranje od strane poslužitelja. Alat je besplatan, no jedini nedostatak je to što nije pogodan za rad s drugim programskim jezicima osim JavaScript-om, zahtjeva puno resursa i česta ažuriranja. [11]

Puppeteer najviše koriste Google i Microsoft. Google je ipak napravio Puppeteer, pa je logično da će ga upravo on koristiti. Google-u je koristan za poboljšanje tražilice i indeksiranje web stranica te praćenje i analizu performansi web stranica. Microsoft-u je koristan alat za kreiranje modela strojnog učenja, ažuriranje vlastitih usluga, testiranje performansi svojih softverskih rješenja temeljenih na internetu te za razna istraživanja i analizu različitih domena.

5.5 Octoparse

Octoparse je alat za koji nije potrebno nikakvo programersko znanje jer sadrži vizualno sučelje koje korisnicima omogućuje navigaciju kroz web stranice i dohvaćanje elemenata jednostavnim pritiskom na njih (engl. "point-and-click" metoda). Nudi alate prepoznavanja i odabira elemenata te mogućnost podešavanja pravila za izdvajanje istih, odnosno definiranja strukture izlaznih podataka. Također, podržava i HTML raščlanjivanje podataka, XPath odabir kao i dohvaćanje sadržaja s dinamičkih web stranica. Ima ugrađene funkcije za paginaciju i kretanje kroz stranice te rad sa složenim web stranicama. Octoparse podržava proxy rotaciju, odnosno korisnici mogu mijenjati IP adrese tijekom struganja za izbjegavanje blokiranja. Nudi i dohvaćanje koje se temelji na oblaku (engl. cloud) čime omogućava da korisnici ne moraju trošiti vlastite računalne resurse prilikom struganja. Izrazito je pogodan i za početnike zbog male količine znanja potrebnog za rukovanje njime. Jedini nedostatak je što alat nije besplatan, no nudi 14 dana besplatne probe kako biste isprobali sve alate koje nudi te se mogli opredijeliti za daljnje korištenje ili ne. [12]

Poglavlje 6

Praktični dio koda

Praktični dio će se sastojati od prilagođenog koda za određenu web stranicu, napisanog u programskom jeziku Pythonu, koristeći biblioteku Beautiful Soup.

6.1 Jednostavan primjer

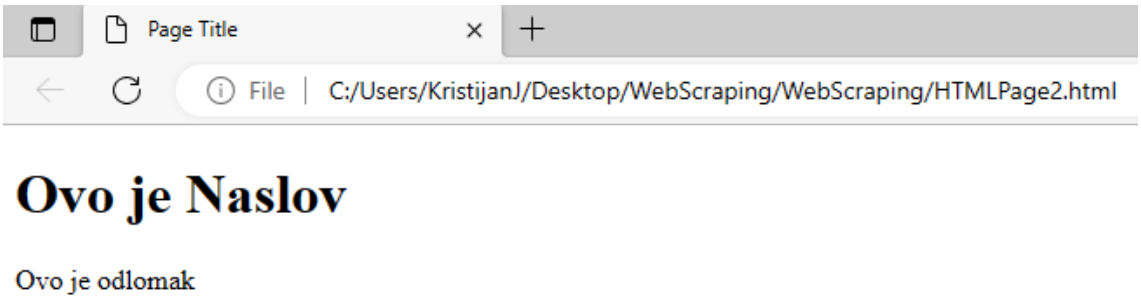
Za početak promotrimo kako uopće funkcionira web struganje na jednom jednostavnom primjeru.

Neka nam sljedeće bude neka HTML web stranica.

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>
<body>
    <h1>Ovo je Naslov</h1>
    <p>Ovo je odlomak</p>
</body>
</html>
```

Listing 6.1 HTML datoteka

Odnosno, stranica će izgledati ovako.



Slika 6.1 HTML stranica

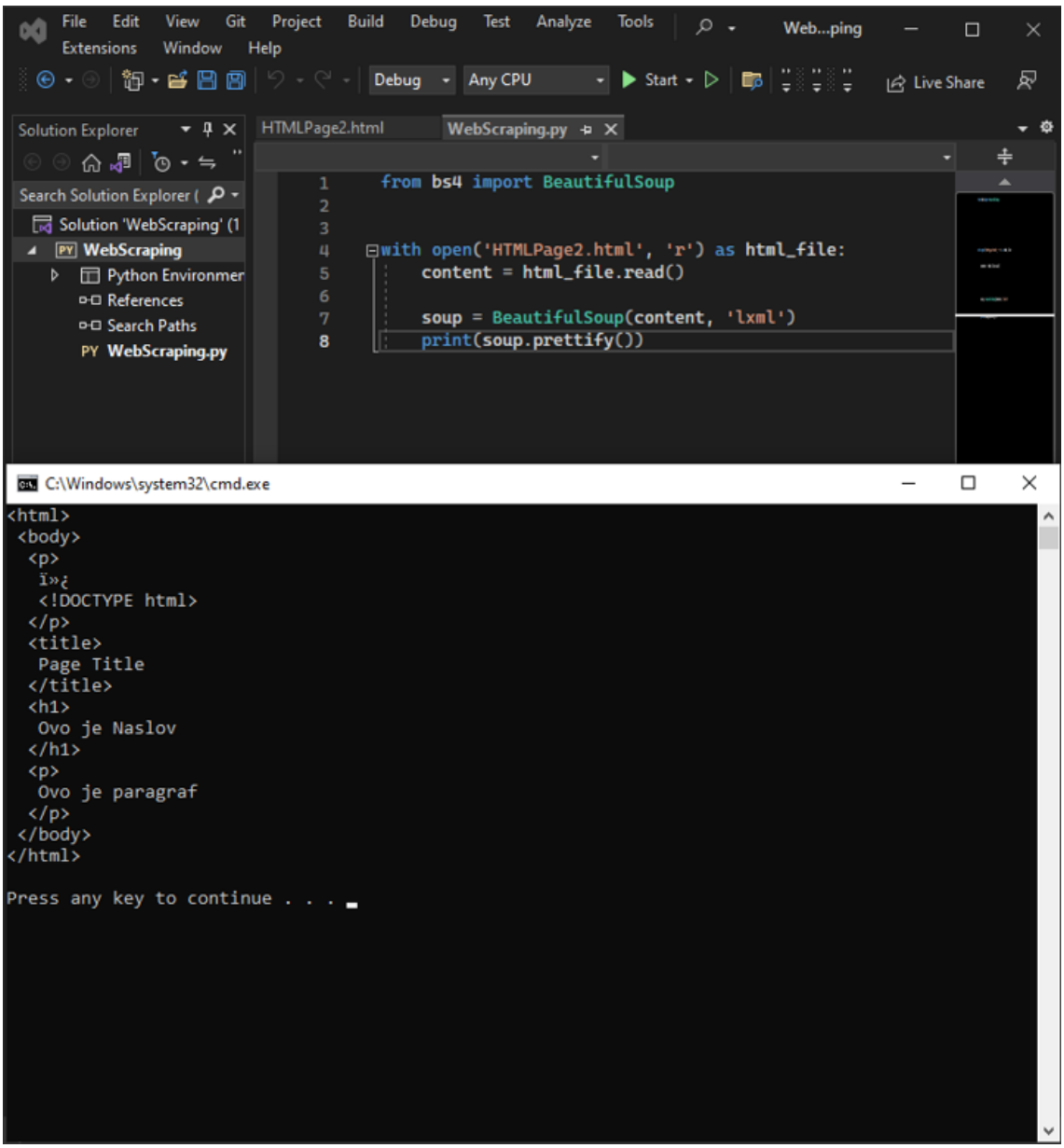
6.2 Struganje koristeći Beautiful Soup

Sada napišimo prilagođen kod za web struganje te stranice. Izgledao bi ovako.

```
from bs4 import BeautifulSoup

with open('HTMLPage2.html', 'r') as html_file:
    content = html_file.read()

    soup = BeautifulSoup(content, 'lxml')
    print(soup.prettify())
```

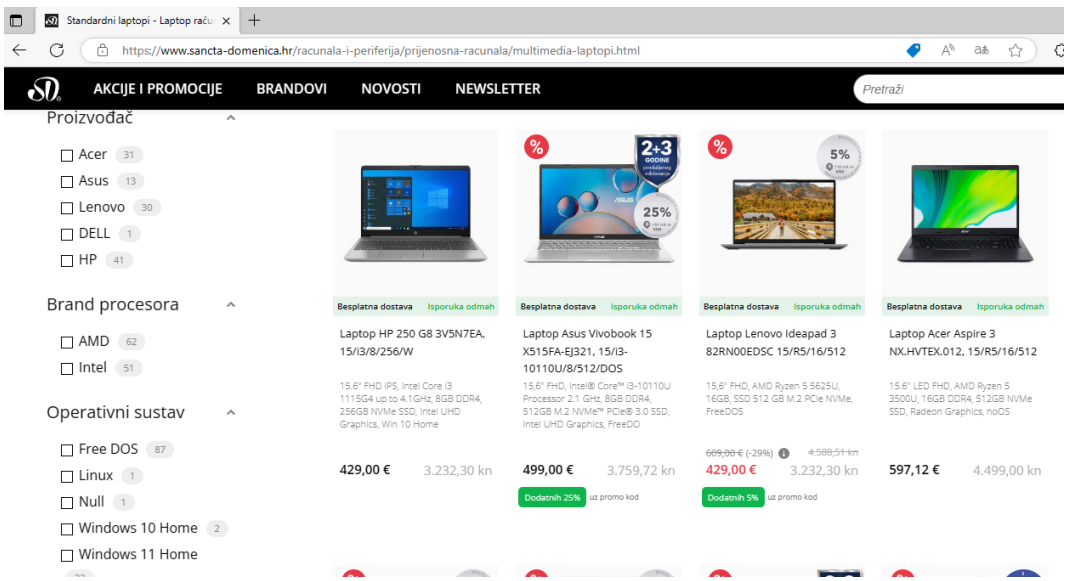
Listing 6.2 Primjer struganja

Potrebno je prvo unijeti biblioteku Beautiful Soup čemu nam služi prva linija koda. Potom, otvaramo HTML dokument imenom "HTMLPage2.html" samo za čitanje "r". Učitavamo podatke koje smo sastrugali te ih printamo. (prettify() nije nužna naredba, već služi samo za ljepši ispis)



Slika 6.2 Ispis HTML koda naše stranice pomoću web struganja

Uglavnom, vidimo da jednostavno možemo prikupiti HTML kod od neke stranice. Pokušajmo sada isto to s nekom postojećom web stranicom. Kao primjer uzmimo web trgovinu Sancta-Domenice te pokušajmo prikupiti podatke o laptopima koji su trenutno aktualni.



Slika 6.3 Web trgovina Sancta-Domenice Laptopi

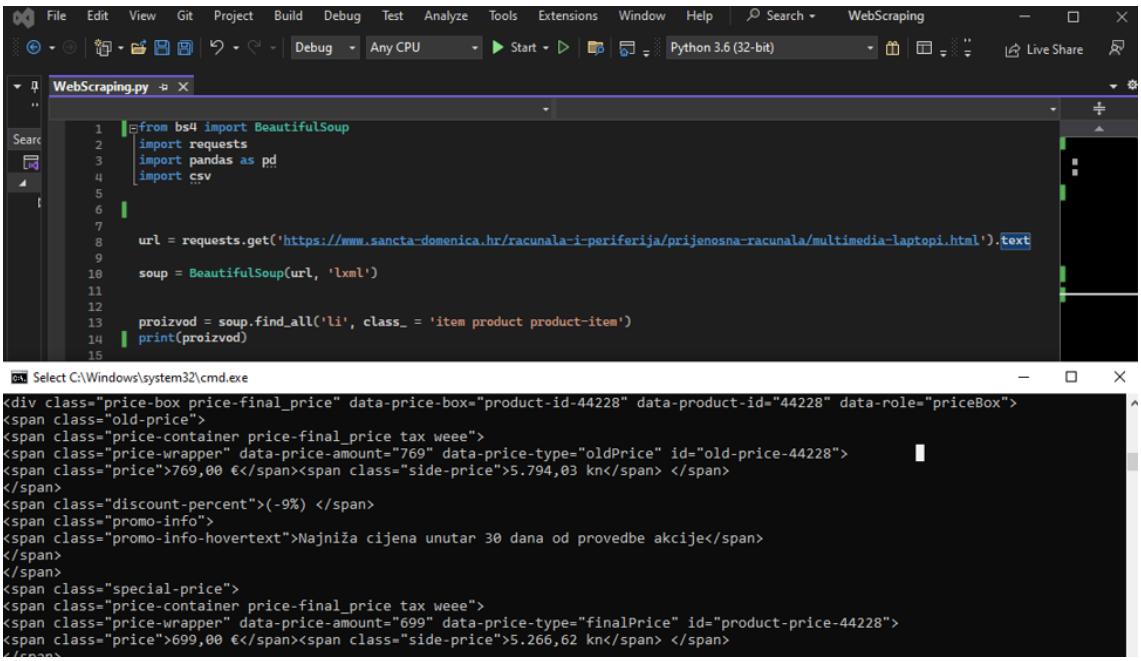
Sada napišimo kod kojim ćemo prikupiti podatke od svih laptopa i prijenosnih računala s te poveznice.

```
1   from bs4 import BeautifulSoup
2   import requests
3   import pandas as pd
4   import csv
5
6   url = requests.get('https://www.sancta-domenica.hr/racunala
    -i-periferija/prijenosna-racunala/multimedia-laptopi.html').
    text
7   soup = BeautifulSoup(url, 'lxml')
8
9   proizvod = soup.find_all('li', class_ = 'item product
    product-item')
```

```
10      print(proizvod)
```

Listing 6.3 Hvatanje koda

Vidimo da se kod malo razlikuje od prethodnog. Na početku moramo unijeti requests biblioteku kako bismo mogli poslati zahtjev stranici. Potom definiramo varijablu url koja će imati za zadaću poslati request i prikupiti HTML skriptu stranice. ".text" nam omogućava da samo uzmemo one dijelove stranice koje su pod poljem teksta. Sa "soup.find-all" tražimo sve HTML elementa s određenom oznakom i određene klase. Sada je već jasnije kako funkcionira princip rada web struganja. Krenimo malo dublje u razradu.



Slika 6.4 Ispis HTML koda web stranice Sancta-Domenice

Sada ćemo prikupiti samo određene podatke od svih laptopa. To će biti naziv laptopa, cijena u eurima i kunama te podaci o dostavi. Sve ćemo sortirati od najveće cijene do najmanje.

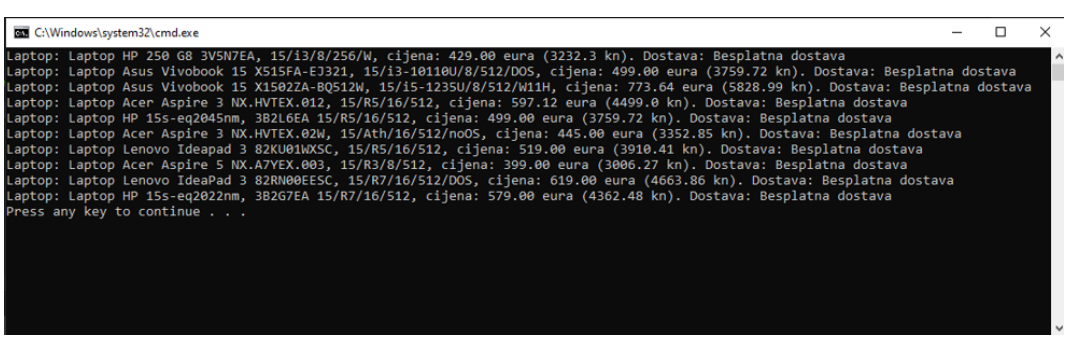
```
1       from bs4 import BeautifulSoup
2       import requests
3       import pandas as pd
```

```
4    import csv
5
6    url = requests.get('https://www.sancta-domenica.hr/racunala
    -i-periferija/prijenosna-racunala/multimedia-laptopi.html').
    text
7    soup = BeautifulSoup(url, 'lxml')
8    proizvod = soup.find_all('li', class_ = 'item product
    product-item')
9
10   naslovL = []
11   Cijena = []
12   Cijenakn = []
13   dostava = []
14
15   for a in proizvod[:10]:
16
17       naslov = a.find('a', class_ = 'product-item-link').text
    .replace('\n', '').strip(' \n, ')
18       naslovL.append(naslov)
19
20       cijena = a.find('span', class_='price').text.strip(' \n
    , ').replace(',', '.').encode('ascii', 'ignore').decode()
21       Cijena.append(cijena)
22
23       cijenakn = a.find('span', class_ = 'side-price').text.
    strip(' \n ').strip('kn').replace('.', '').replace(',', '.')
    .encode('ascii','ignore').decode()
24       Cijenakn.append(cijenakn)
25
26       dost = a.find('div', class_ = 'free-shipping').text
27       dostava.append(dost)
28
29   #sort od najvece cijene ka najmanjoj
30   for i in range(len(naslovL)):
31       for j in range(i, len(naslovL)):
```

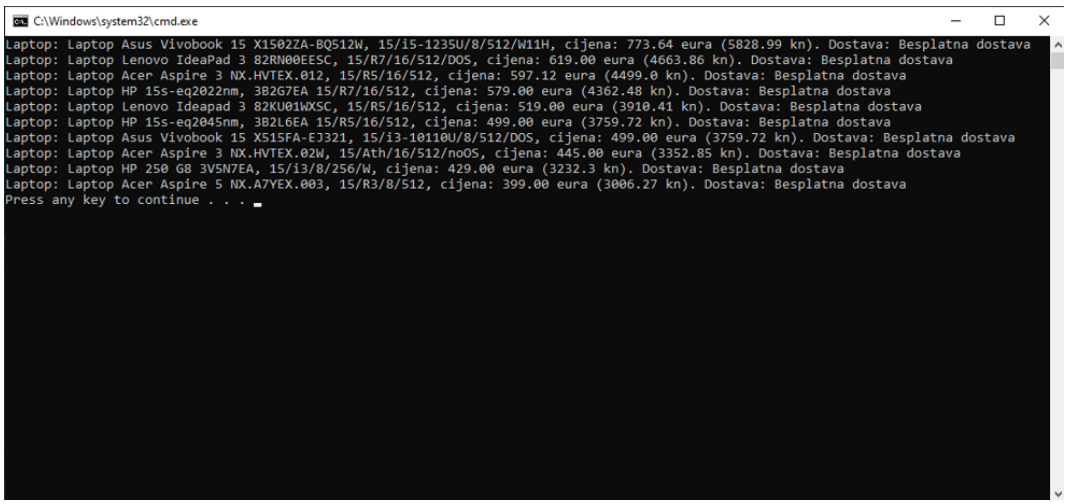
```
            if float(Cijenakn[i]) <= float(Cijenakn[j]):
                tmpKN = Cijenakn[i]
                Cijenakn[i] = Cijenakn[j]
                Cijenakn[j] = tmpKN

                tmpE = Cijena[i]
                Cijena[i] = Cijena[j]
                Cijena[j] = tmpE

                tmpN = naslovL[i]
                naslovL[i] = naslovL[j]
                naslovL[j] = tmpN

    print("Laptop: {}, cijena: {} eura ({} kn). Dostava: {}" .
format(naslovL[i], Cijena[i], float(Cijenakn[i]), dostava[i
]))
```

Listing 6.4 Primjer struganja

Za svaki podatak posebno definiramo listu u koju ćemo spremati podatke. Ograničili smo se samo na prvih 10 artikala s dijelom koda "for a in proizvodi[:10]" pomoću "[:10]". Razne naredbe ".replace(...), .endoce()..." služe za rješavanje neželjenih znakova te ujedno omogućuju pretvorbu string tipa podataka u float kako bismo mogli izvršiti sortiranje. Ispis koda je sljedeći:



Slika 6.5 Printanje prvih deset artikala



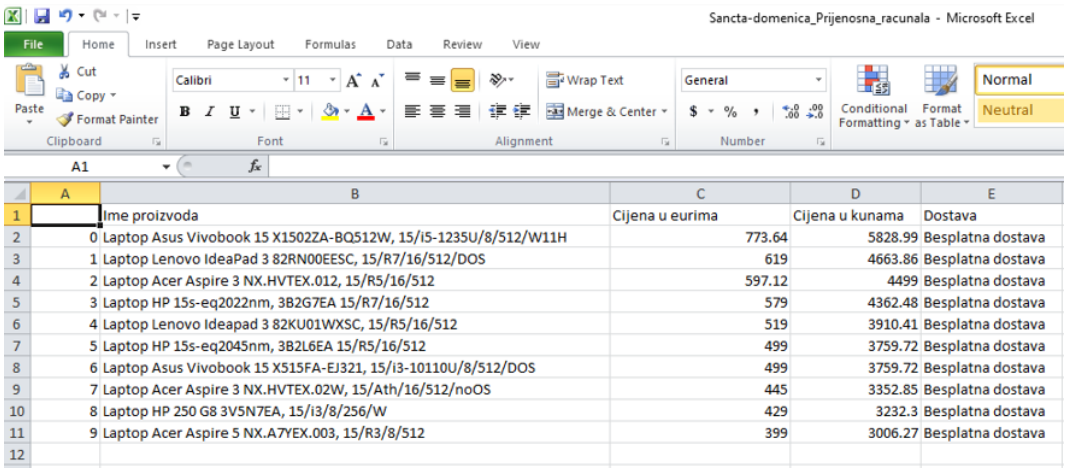
Slika 6.6 Printanje nakon sortiranja

Naposljetku želimo da te prikupljene podatke imamo spremljene kako bismo ih mogli koristiti za daljnje upotrebe. Spremamo ih u .csv datoteku u koju možemo pokrenuti u Excel-u. Za to će nam pomoći sljedeći dio koda koji nadopisujemo na prethodno prikazani kod nakon posljednje for petlje.

```
    df = pd.DataFrame()

    df['Ime proizvoda'] = naslovL
    df['Cijena u eurima'] = Cijena
    df['Cijena u kunama'] = Cijenakn
    df['Dostava'] = dostava

    print(df)
    df.to_csv('Sancta-domenica_Prijenosna_racunala.csv')
```

Listing 6.5 Hvatanje koda

Koristeći pandas biblioteku možemo koristiti gore navedene naredbe što nam olakšava postupak pohrane podataka. Spremamo ih pod imenom "Sancta-domenica-Prijenosna-racunala.csv"



	A	B	C	D	E
1		Ime proizvoda	Cijena u eurima	Cijena u kunama	Dostava
2	0	Laptop Asus Vivobook 15 X1502ZA-BQ512W, 15/i5-1235U/8/512/W11H	773.64	5828.99	Besplatna dostava
3	1	Laptop Lenovo IdeaPad 3 82RN00EESC, 15/R7/16/512/DOS	619	4663.86	Besplatna dostava
4	2	Laptop Acer Aspire 3 NX.HVTEX.012, 15/R5/16/512	597.12	4499	Besplatna dostava
5	3	Laptop HP 15s-eq2022nm, 3B2G7EA 15/R7/16/512	579	4362.48	Besplatna dostava
6	4	Laptop Lenovo Ideapad 3 82KU01WXSC, 15/R5/16/512	519	3910.41	Besplatna dostava
7	5	Laptop HP 15s-eq2045nm, 3B2L6EA 15/R5/16/512	499	3759.72	Besplatna dostava
8	6	Laptop Asus Vivobook 15 X515FA-EJ321, 15/i3-10110U/8/512/DOS	499	3759.72	Besplatna dostava
9	7	Laptop Acer Aspire 3 NX.HVTEX.02W, 15/Ath/16/512/noOS	445	3352.85	Besplatna dostava
10	8	Laptop HP 250 G8 3V5N7EA, 15/i3/8/256/W	429	3232.3	Besplatna dostava
11	9	Laptop Acer Aspire 5 NX.A7YEX.003, 15/R3/8/512	399	3006.27	Besplatna dostava
12					

Slika 6.7 CSV datoteka sa prikupljenim podacima

6.3 Struganje dinamičkih stranica

Preostaje nam pokazati još struganje podataka s dinamičkih web stranica koje koriste JavaScript za renderiranje stranice. Za primjer ćemo uzeti društvenu mrežu Facebook. Kako smo naučili da Beautiful Soup nema podršku za struganje s dinamičkih stranica, koristit ćemo Selenium. Prvi korak za korištenje Seleniuma je instaliranje drivera za browser koji ćemo koristiti. U našem primjeru neka to bude "Edge driver". Također, vodite računa o tome da se nalazi u istom folderu u kojem se nalazi i naš Selenium projekt. Za početak uvedimo neke ključne biblioteke. Redom na narednoj slici su biblioteke za rad s našim prethodno instaliranim web driverom, mogućnost korištenja tipki tipkovnice, korištenje uvjeta tijekom odabira elemenata, odabir vrste tagova koje dohvaćamo te za čekanje kako ne bi došlo do greške prilikom učitavanja.

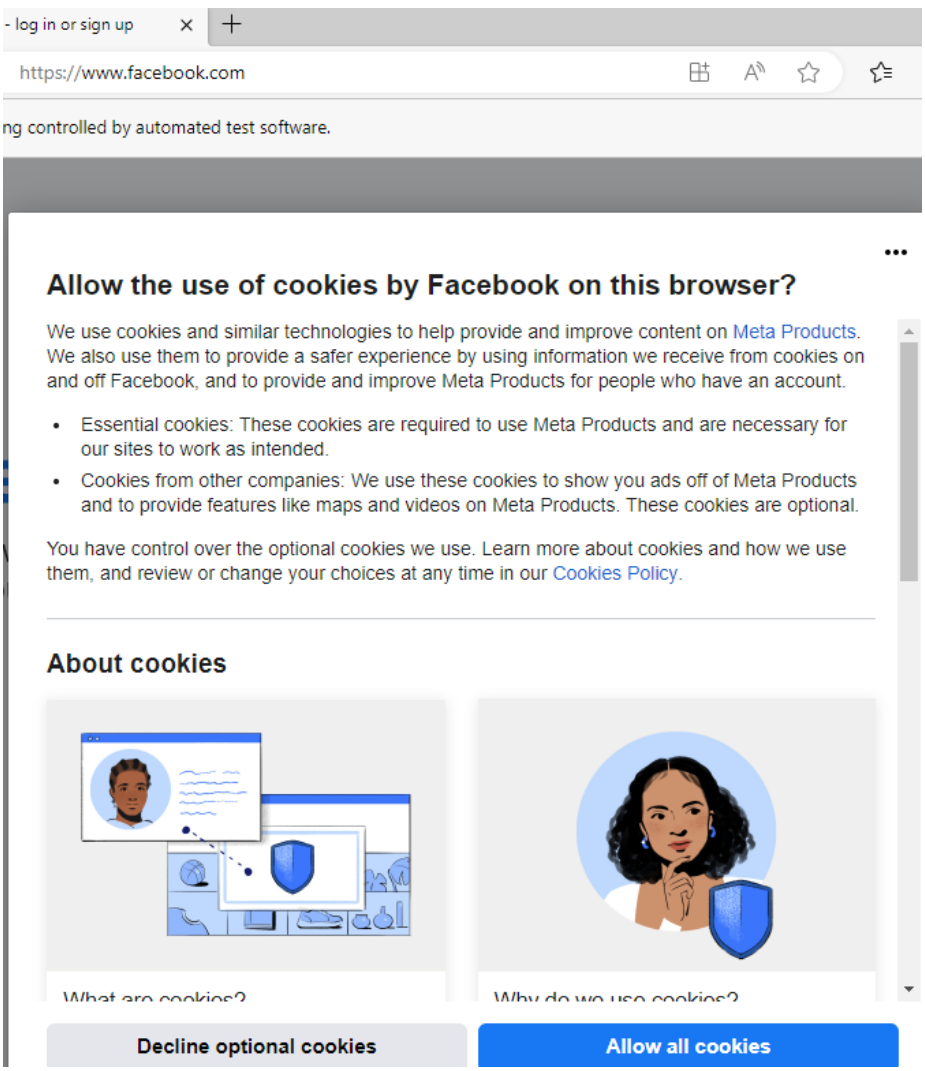
```
1 from selenium import webdriver
2 from selenium.webdriver.common.keys import Keys
3 from selenium.webdriver.support import expected_conditions as
    EC
4 from selenium.webdriver.common.by import By
5 from selenium.webdriver.support.wait import WebDriverWait
```

Listing 6.6 Selenium biblioteke koje ćemo koristit

Sljedeći korak je definiranje "driver" varijable koja za argument uzima put do našeg Edge drivera te potom dohvaćanje ciljanog url-a pomoću iste.

```
1 driver = webdriver.Edge('C:/Users/KristijanJ/Desktop/Selenium/
    Selenium/msedgedriver.exe')
2 driver.get('https://www.facebook.com/')
```

Listing 6.7 Definiranje driver varijable.



Slika 6.8 Početna stranica Facebook-a

Nakon pokretanja, otvara nam se početna stranica Facebooka. Kako vidimo naila-

zimo na neke sitne probleme, odnosno moramo se prvo riješiti prozora za kolačiće. Otvorimo "inspect" od stranice te pronađimo tag od gumba koji sadrži tekst "Allow all cookies". Koristit ćemo XPATH dohvaćanje zbog toga što je dosta specifično definiran gumb, a gumb sam po sebi je previše općenit te koristimo naredbe ".click()" i "WebDriverWait".

```
accept_cookies = WebDriverWait(driver, 10).until(EC.
    element_to_be_clickable((By.XPATH, "//button[contains(text()
    , 'Allow all cookies')]"))).click()
```

Listing 6.8 Dohvaćanje gumba za prihvaćanje kolačića

Isti proces ponovimo za polja "username" i "password" u koja potom s naredbom ".send_keys()" upisujemo email i lozinku s kojom se želimo ulogirati. Nakon toga, kao i za prethodni gumb, dohvaćamo gumb s textom "Log in".

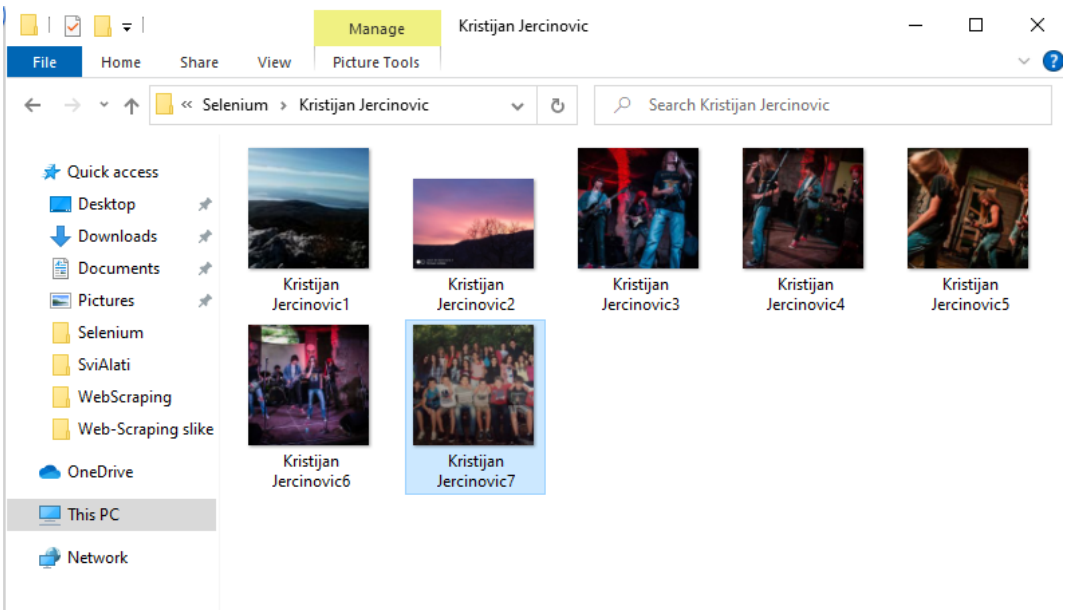
Slika 6.9 Dohvaćanje gumba za "Log in"

Nakon što smo se uspješno ulogirali, direktno ćemo otvoriti stranicu s koje ćemo dohvatiti sve slike (možete i upisati u tražilicu ime stranice te potom se navigirati do fotografija). Pokazat ću na primjeru vlastitih fotografija kako ne bi narušavao privatnost nekih stranica. Unutar for petlje, dohvatit ćemo nove url-ove na kojima se nalaze objavljene fotografije. Pomoću for petlje mogu se dohvatiti i slike s proširenih url-ova tako dodavanjem nastavaka za url. Naredba "window.scrollTo" omogućava kretanje po stranici, sa "path" definiramo putanju gdje ćemo spremiti slike (u našem slučaju u isti direktorij) te s "os.mkdir" kreiramo novi direktorij, ako ne postoji, u koji ćemo pohraniti slike. "time.sleep()" omogućuje čekanje kako ne bi došlo do pretrpavanja te s "wget.download" spremamo slike pod imenom koje smo definirali prije (to će biti "keyword.jpg"). Naposljetku dodajmo još naredbu "driver.quit()" kako bismo zatvorili naš web preglednik.

```
1  from selenium import webdriver
2  from selenium.webdriver.common.keys import Keys
3  from selenium.webdriver.support import expected_conditions as
       EC
4  from selenium.webdriver.common.by import By
5  from selenium.webdriver.support.wait import WebDriverWait
6  import wget
7  import os
8  import time
9
10 driver = webdriver.Edge('C:/Users/KristijanJ/Desktop/Selenium/
       Selenium/msedgedriver.exe')
11 driver.get('https://www.facebook.com/')
12
13 accept_cookies = WebDriverWait(driver, 10).until(EC.
       element_to_be_clickable((By.XPATH, "//button[contains(text()
       , 'Allow all cookies')]"))).click()
14
15 username = WebDriverWait(driver, 10).until(EC.
       element_to_be_clickable((By.CSS_SELECTOR, "input[name='email
       ']")))
16 password = WebDriverWait(driver, 10).until(EC.
       element_to_be_clickable((By.CSS_SELECTOR, "input[name='pass
       ']")))
17
18 username.clear()
19 password.clear()
20 username.send_keys('kristijanjercinovic@gmail.com')
21 password.send_keys('************')
22
23 log_in = WebDriverWait(driver, 20).until(EC.
       element_to_be_clickable((By.CSS_SELECTOR, "button[type='
       submit']"))).click()
24 keyword = "Kristijan Jercinovic"
25
```

```
for i in ['photos']:
    driver.get("https://www.facebook.com/kristijan.jercinovic.9/" + i + "/")
    time.sleep(2)
    n_scroll = 2
    for j in range(1, n_scroll):
        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(2)
        images = driver.find_elements_by_tag_name('img')
        images = [image.get_attribute('src') for image in images]

        path = os.getcwd()
        path = os.path.join(path, keyword)
        if not os.path.exists(path):
            os.mkdir(path)

        counter = 0;
        for image in images:
            save_as = os.path.join(path, keyword + str(counter) + '.jpg')
            time.sleep(2)
            wget.download(image, save_as)
            counter += 1
driver.quit()
```

Listing 6.9 Cjelokupni Selenium kod



Slika 6.10 Slike dohvaćene struganjem pomoću Seleniuma

Bibliografija

[1] A brief history of web scraping. , s Interneta, https://webscraper.io/blog/brief-history-of-web-scraping#:~:text=Brief20History20of20Web20Scraping20120The20birth/

[2] R.Mitchell, *Web Scraping with Python: Collecting More Data from the Modern Web*.

[3] M. Khder. Web scraping or web crawling: State of art, techniques, approaches and application.

[4] S.Aqa. What is ai web scraping. , s Interneta, https://spenceraqason.medium.com/what-is-ai-web-scraping-e32dee3a643b#:~:text=While%20regular%20web%20scraping%20

[5] How web scraping is shapping the future of machine learning. , s Interneta, https://builtin.com/machine-learning/web-scraping-machine-learning

[6] M.Perez. What is web scraping used for. , s Interneta, https://www.parsehub.com/blog/what-is-web-scraping/

[7] The best web scraping techniques. , s Interneta, https://metrow.com/blog/web-scraping-techniques/

[8] C.Gillen. How to use an api: Guide + tutorial. , s Interneta, https://zapier.com/blog/how-to-use-api/

[9] How to rotate proxies in web scraping. , s Interneta, https://scrapfly.io/blog/how-to-rotate-proxies-in-web-scraping/

[10] G.Hajba, *Website Scraping with Python Using BeautifulSoup and Scrapy*.

[11] Puppeteer. , s Interneta, https://learn.microsoft.com/en-us/microsoft-edge/puppeteer/

[12] Octoparse. , s Interneta, https://www.octoparse.com/blog/what-is-octoparse

Sažetak

Web struganje je tehnika prikupljanja podataka s internet stranica te spremanje tih podataka za daljnje korištenje. Može se koristiti ručno ili automatizirano, iako ručni način više nitko ne koristi, jer se sve više pokušava razviti potpuno automatizirani način struganja korištenjem umjetne inteligencije. Automatizacija olakšava struganje omogućujući jednostavno raščlanjivanje HTML koda stranice i dohvaćanje podataka koje želimo, jednostavan pristup API-u te konstantno mijenjanje IP adresa, prilikom korištenja proxy rotacije, tijekom zaobilaženja ograničenja od stranica. Struganje ima široku primjenu od korištenja za uspoređivanje cijena na tržištu, praćenje vremenskih prognoza, dohvaćanja objava, komentara, rasprava, lokacija na društvenim mrežama, pa do razvoja modela strojnog učenja. Također, treba voditi računa koji alat za struganje koristimo, jer ako želimo strugati podatke s dinamičkih web stranica, onda ćemo koristiti Selenium, Puppeteer ili Beautiful Soup sa Selenium-om od besplatnih alata te Octoparse od alata koji se plaćaju. S druge strane, ako su nam u cilju samo statičke stranice možemo koristiti Beautiful Soup ili Scrapy, vodeći računa o količini podataka koje želimo strugati.

U ovom radu pokazali smo karakteristike pojedinih tehnika struganja i dali primjere kako izgledaju u praksi. Također, demonstrirali smo na konkretnim primjerima upotrebu alata za struganje, kao što su Beautiful Soup, Selenium te nešto malo o Scrapy-ju.

Ključne riječi **— web struganje, Beautiful Soup, Scrapy, Selenium, HTML zahtjevi, JavaScript**

Abstract

Web scraping is a technique of collecting data from the internet and saving it for further use. It can be used manually or automatically, although the manual mode is no longer used, because the weight nowadays is to make web scraping fully automated using artificial intelligence. Automation makes scraping more productive by making HTML parsing and the retrieving of the data we want fast and easy, making APIs

more accessable and constantly changing IP addresses when using proxy rotation, while bypassing site restrictions. Scraping has a wide range of applications from using it to compare prices, tracking weather forecasts, retrieving posts, comments, discussions, locations on social media and even developing machine learning models. Also, we should take care which scraping tool we use, because if we want to scrape data from dynamic web sites, then we will use Selenium, Puppeteer or Beautiful Soup with Selenium from the free tools and Octoparse from paid ones. On the other hand, if our goal is to scrape data only from static sites, we can use Beautiful Soup or Scrapy, taking into account the amount of data we want to scrape.

In this work, we have shown the characteristics of certain scraping techniques and given examples of how they look in practical use. Also, we demonstrated on examples the use of scraping tools such as Beautiful Soup, Selenium and a little bit of Scrapy.

Keywords **— web scraping, Beautiful Soup, Scrapy, Selenium, HTML requests, JavaScript**