Razvoj Python aplikacije za sistematiziranu ekstrakciju sadržaja s web-stranica

Šarić, Jakov

Sveučilište u Zadru

Odjel za informacijske znanosti

Stručni prijediplomski studij

Informacijske tehnologije

Jakov Šarić

Razvoj Python aplikacije za sistematiziranu ekstrakciju sadržaja s web-stranica

Završni rad

Zadar, 2024.

Sveučilište u Zadru
Odjel za informacijske znanosti
Stručni prijediplomski studij
Informacijske tehnologije

Razvoj Python aplikacije za sistematiziranu ekstrakciju sadržaja s web-stranica

Završni rad

Student/ica: Jakov Šarić

Mentor/ica: Doc. dr. sc. Marko Šarlija

Zadar, 2024.

Izjava o akademskoj čestitosti

Ja, **Jakov Šarić,** ovime izjavljujem da je moj **završni** rad pod naslovom **Razvoj Python aplikacije za sistematiziranu ekstrakciju sadržaja s web-stranica** rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na izvore i radove navedene u bilješkama i popisu literature. Ni jedan dio mojega rada nije napisan na nedopušten način, odnosno nije prepisan iz necitiranih radova i ne krši bilo čija autorska prava.

Izjavljujem da ni jedan dio ovoga rada nije iskorišten u kojem drugom radu pri bilo kojoj drugoj visokoškolskoj, znanstvenoj, obrazovnoj ili inoj ustanovi.

Sadržaj mojega rada u potpunosti odgovara sadržaju obranjenoga i nakon obrane uređenoga rada.

Zadar, 27. rujna 2024.

Sadržaj

1. UVOD ... 8
2. SISTEMATIZIRANA EKSTRAKCIJA SADRŽAJA S WEB-STRANICA ... 9
   2.1. OPĆENITO ... 9
   2.2. VRSTE SISTEMATIZIRANE EKSTRAKCIJE ... 10
      2.2.1. Statička web ekstrakcija ... 10
      2.2.2. Dinamička web ekstrakcija ... 11
      2.2.3. DOM Parsing ... 11
      2.2.4. Ekstrakcija putem API-ja ... 11
      2.2.5. Headless Browser Scraping ... 12
      2.2.6. Ekstrakcija putem RSS Feeda ... 12
      2.2.7. OCR ekstrakcija ... 13
      2.2.8. Ekstrakcija sadržaja s proxy serverima i rotacijom IP adresa ... 13
   2.3. TIPIČNI PROBLEMI ... 13
3. ALATI ZA SISTEMATIZIRANU EKSTRAKCIJU ... 15
   3.1. BeautifulSoup ... 15
   3.2. Lxml ... 15
   3.3. Scrapy ... 16
   3.4. Selenium ... 16
   3.5. Puppeteer ... 16
   3.6. Playwright ... 16
   3.7. XPath ... 16
   3.8. Feedparser ... 17
   3.9. Tesseract ... 17
   3.10. Octoparse ... 17
4. PROBLEMI I CILJEVI ... 18
5. RAZVOJ APLIKACIJE ... 20
   5.1. POSTAVLJANJE RADNOG OKRUŽENJA ... 20
   5.2. DIZAJN BAZE PODATAKA ... 22
   5.3. RAZVOJ KODA ZA SISTEMATIZIRANU EKSTRAKCIJU ... 23
   5.4. PRIKAZ EKSTRAHIRANIH PODATAKA ... 30
   5.5. VREMENSKO POKRETANJE ... 34
      5.5.1. Vremensko pokretanje na Windows operativnom sustavu ... 34
      5.5.2. Vremensko pokretanje na Linux operativnom sustavu ... 35
6. ZAKLJUČAK ... 37

LITERATURA ... 38

POPIS SLIKA ... 40

GITHUB.. 41

SAŽETAK

Ovaj završni rad bavi se sistematiziranom ekstrakcijom sadržaja s web-stranica, metodama i alatima koji omogućuju automatizirano prikupljanje podataka s interneta. U uvodu se daje pregled problematike te kako će se ti problemi riješiti. Detaljno su opisane različite vrste ekstrakcije, uključujući statičku i dinamičku ekstrakciju, DOM *parsing*, ekstrakciju putem API-ja, *Headless Browser Scraping*, RSS *feedovi*, OCR tehnologiju te korištenje *proxy* servera i rotacija IP adresa. Osim različitih vrsta ekstrakcije, opisani su ukratko najpopularniji alati za ekstrakciju kao što su BeautifulSoup, Scrapy, Selenium. Tesseract i ostali, zajedno s njihovim karakteristikama i primjenama. Opisani su problemi koji se žele riješiti te postavljeni ciljevi koje će projekt ostvariti.

Veći dio rada detaljno opisuje razvoj aplikacije za sistematiziranu ekstrakciju, od postavljanja radnog okruženja i dizajna baze podataka, do implementacije koda i vremenskog pokretanja skripti na različitim operativnim sustavima. Rad završava zaključkom koji sumira najbitnije stavke. Rad pruža sveobuhvatan pregled tehnika i alata potrebnih za efikasnu automatizaciju prikupljanja podataka te nudi praktično rješenje kroz razvoj prilagođene aplikacije.

Ključne riječi: ekstrakcija, web, scraping, Python, Flask, baza podataka, programski alati, vremensko pokretanje skripti, automatizacija prikupljanja podataka

POPIS KORIŠTENIH KRATICA

- API – Application Programming Interface
- CAPTCHA – Completely Automated Public Turing test to tell Computers and Humans Apart
- CDF – Change Data Feed
- CPU – Central Processing Unit
- CSS – Cascading Style Sheets
- CSV – Comma-separated Values
- DOM – Document Object Model
- HTML – HyperText Markup Language
- HTTP – HyperText Transfer Protocol
- IP – Internet Protocol address
- JSON – JavaScript Object Notation
- OCR – Optical Character Recognition
- RSS – Really Simple Syndication
- SQL – Structured Query Language
- SVG – Scalable Vector Graphics
- URL – Uniform Resource Locator
- VPN – Virtual Private Network
- XML – Extensible Markup Language

1. UVOD

Početkom 21. stoljeća računala i internet postali su neizostavni dio svakodnevnog života. S ubrzanim razvojem tehnologije, upotreba interneta kontinuirano raste te danas ima ključnu ulogu u obavljanju različitih aktivnosti, kao što su informiranje, komunikacija, konzumacija medijskih sadržaja te online kupovina. Način na koji obavljamo svakodnevne zadatke značajno se promijenio uvođenjem digitalnih tehnologija. Aktivnosti koje su nekada zahtijevale fizičku prisutnost, poput odlaska na kiosk radi kupovine novina ili odlaska u kino, danas su zamijenjene digitalnim alternativama poput pregledavanja internetskih portala i streaming servisa.

Online kupovina, kao jedan od najistaknutijih primjera digitalne transformacije, omogućila je korisnicima značajnu uštedu vremena i povećala dostupnost proizvoda. Umjesto fizičkog obilaska trgovina, korisnici sada imaju mogućnost naručivanja proizvoda iz udobnosti vlastitog doma. Međutim, uz prednosti koje donosi digitalizacija, javljaju se i izazovi, osobito u pogledu navigacije kroz veliku količinu dostupnih informacija. Inflacija i rast cijena dodatno su povećali važnost pronalaska najpovoljnijih ponuda, no mnoštvo informacija i nepouzdanih izvora na internetu otežava korisnicima donošenje informiranih odluka.

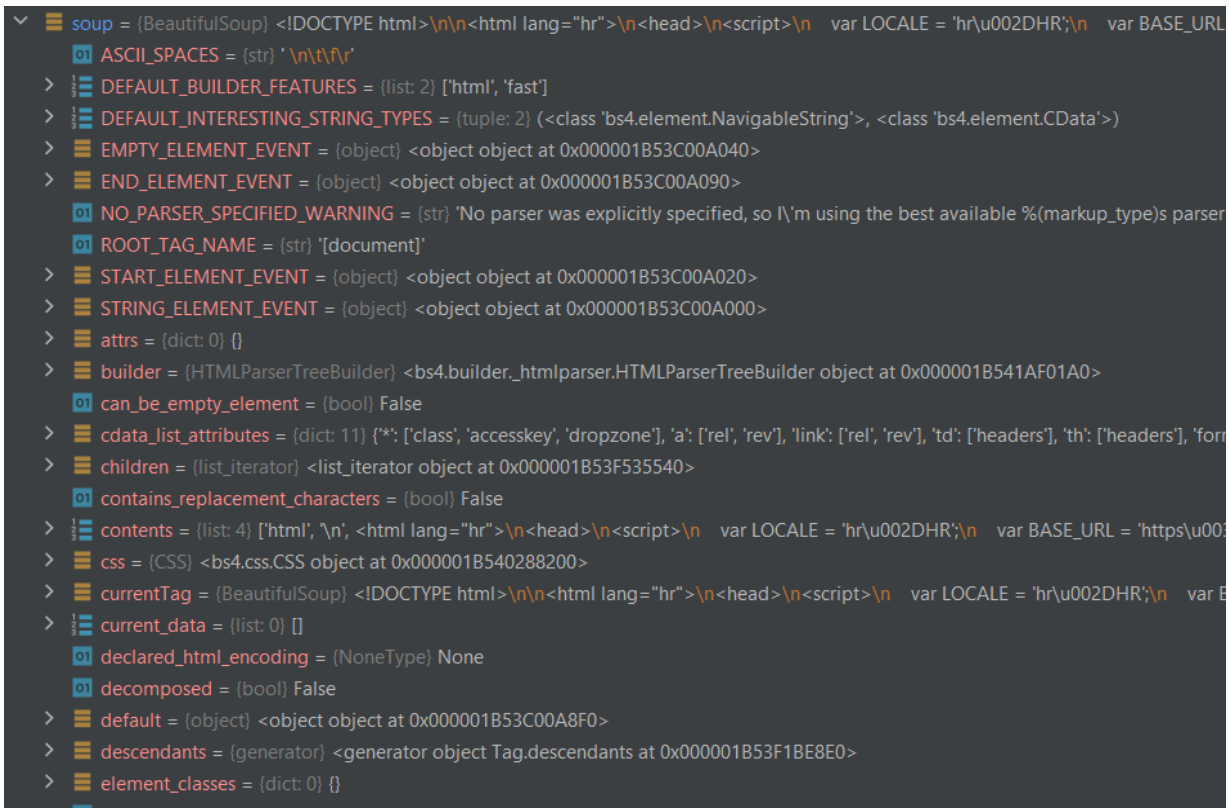
Ovaj završni rad fokusira se na rješenje tog problema kroz razvoj alata za web *scraping*, s ciljem usporedbe cijena proizvoda u različitim internetskim trgovinama. Projekt se trenutno temelji na analizi ponude televizora, no planirano je proširenje na druge kategorije proizvoda. Primarni cilj ovog rada je prikazati alat pomoću kojeg potencijalni korisnici mogu sigurno i učinkovito pretraživati online trgovine, dok alat uspoređuje cijenu istih proizvoda te pronalazi najpovoljnijih ponuda.

Projekt se sastoji od Python programa za *web scraping*, svaki program ekstrahira podatke s web lokacije koja mu je dodijeljena. Podaci se zatim spremaju u bazu podataka iz koje posebni Python program učitava sve podatke o proizvodima te ih uspoređuje, odnosno traži iste proizvode u različitim trgovinama s različitim cijenama. Novi podaci spremaju se u novu tablicu unutar baze podataka, koje potom Flask web aplikacija prikazuje u vizualnom obliku preko korisničkog sučelja.

2. SISTEMATIZIRANA EKSTRAKCIJA SADRŽAJA S WEB-STRANICA

2.1. OPĆENITO

Sistematizirana ekstrakcija sadržaja s web-stranica metoda je kojom se automatski mogu preuzeti određeni podaci s interneta. Ti podaci su nestrukturirani te su u HTML formatu. Uz pomoć određenih alata pretvaraju se u strukturirane podatke koji poprimaju određeno značenje [1], [2]. Na slici ispod (Slika 1) prikazano je kako izgledaju nestrukturirani podaci, u pregledniku varijabli unutar PyCharm-a, nakon izvršene ekstrakcije uz pomoć Python biblioteke pod imenom Beautiful Soup.



Slika 1 Nestrukturirani podaci nakon ekstrakcije

Postoji mnogo različitih načina za izvođenje sistematizirane ekstrakcije za dobivanje podataka s web-stranica. To uključuje korištenje mrežnih usluga, određenih API-ja ili čak stvaranje koda za sistematiziranu ekstrakciju od nule. Mnoge velike web stranice, poput Googlea, Twittera, Facebooka, StackOverflowa itd. imaju API-je koji omogućuju pristup njihovim podacima u strukturiranom formatu. Taj način je najbolji jer takvi ekstrahirani podaci su jednostavni za upravljanje. *Web Scraper-i* , programski alati za ekstrakciju sadržaja s web-stranica, odnosno

web strugači, mogu izdvojiti sve podatke na određenim stranicama ili specifične podatke koje korisnik želi. U idealnom slučaju, najbolje je navesti željene podatke kako bi web strugač brzo izvukao samo te podatke. Dakle, kada alat za struganje weba treba ekstrahirati web mjesto, prvo se daju URL-ovi. Zatim se učitava sav HTML kod stranice, a napredniji alat za struganje mogao bi čak izdvojiti i sve CSS i Javascript elemente. Zatim strugač dobiva tražene podatke iz tog HTML koda i ispisuje te podatke u formatu koji odredi korisnik. Uglavnom je to u obliku Excel proračunske tablice ili CSV datoteke, no podaci se mogu spremiti i u drugim formatima, poput JSON datoteke [2], [3], [4].

2.2. VRSTE SISTEMATIZIRANE EKSTRAKCIJE

Postoji nekoliko metoda ili tipova „web *scrapinga*“, koje zavise od načina prikupljanja podataka i složenosti strukture stranice. Glavni tipovi „web *scrapinga*“ koji će biti objašnjeni u sljedećim potpoglavljima:

- Statički web scraping
- Dinamički web scraping
- DOM Parsing
- Scraping putem API-ja
- Headless Browser Scraping
- XPath Scraping
- Scraping putem RSS Feeda
- OCR Scraping
- Web Scraping s *proxy* serverima i rotacijom IP adresa.

2.2.1. Statička web ekstrakcija

Prilikom statičkog ekstrahiranja web stranica sadržaj se preuzima sa statičkih web stranica, što znači da takve stranice ne koriste JavaScript ili slične tehnologije za ažuriranje sadržaja. Svi podaci su unaprijed definirani unutar HTML koda te su zbog toga lako dostupni. Ovakav tip jednostavan je za implementiranje te omogućuje brzo preuzimanje podataka. Nedostatci ovog pristupa su neažuriranje podataka bez ponovnog preuzimanja stranice te je pogodan isključivo za statične stranice [5].

Pogodni alati su: BeautifulSoup, lxml, Scrapy.

Primjer: Preuzimanje podataka sa statične stranice poput https://zadarskilist.novilist.hr/. Svaki članak je definiran unutar HTML koda.

2.2.2. Dinamička web ekstrakcija

Dinamička ekstrakcija sadržaja upotrebljava se kada stranica koristi JavaScript za generiranje sadržaja koji se učitava nakon što je stranica učitana. Kako bi to bilo moguće, potrebna je emulacija ili izvršavanje JavaScript koda kako bi se dobio potpuni sadržaj stranice. Osim toga, prednost ovog tipa ekstrahiranja sadržaja je rad s interaktivnim web stranicama. Uz određene prednosti dolaze i neki nedostatci. Preuzimanje podataka s dinamičke stranice bit će usporeno u odnosu na statičku stranicu, zbog potrebe za izvršavanje JavaScript-a. Sve to zahtijeva i više resursa (CPU, memorija). Prilikom korištenja emulacije preglednika moguća je blokada pristupa zbog „neovlaštenog pristupa“ [6], [7], [8].

Pogodni alati su: Selenium, Puppeteer, Playwright.

Primjer: Preuzimanje podataka sa stranica poput X-a (bivši Twitter) ili Facebooka.

2.2.3. DOM Parsing

„DOM Parsing“ uključuje rad s DOM strukturom HTML dokumenta. DOM predstavlja hijerarhiju elemenata na stranici (oznake, klase, atributi), omogućujući vam navigaciju kroz stranice i odabir određenih elemenata iz HTML-a. Navedeni tip izuzetno je koristan kada želimo ekstrahirati podatke iz određenog dijela web stranice, pogotovo ako radimo sa statičnim stranicama. Izazovi s kojima se susreće ovakav pristup pojavljuju se kod stranica s vrlo složenom DOM strukturom što može dovesti do otežane identifikacije traženih elemenata [9], [10].

Pogodni alati su: BeautifulSoup, lxml, XPath.

Primjer: Ekstrakcija cijena određenog proizvoda iz web shopa poput Sancta Domenica Webshop.

2.2.4. Ekstrakcija putem API-ja

Neke web stranice pružaju API-je (Application Programming Interfaces) kako bi omogućile pristup strukturiranim podacima u JSON ili XML formatu. API omogućava direktnu komunikaciju sa serverima i obično su najpouzdaniji način za preuzimanje podataka. Velika

prednost ovog pristupa su strukturirani podaci koje dobijemo i s kojima je lakše upravljati. Koristeći ekstrakciju putem API-ja smanjujemo mogućnost blokiranja pristupa web stranici jer nema „neautoriziranog“ pristupa. Problem može nastati prilikom korištenja API-ja koji ograničava pristup ili se plaća te je potrebno poznavanje API koncepta poput RESTful-a ili GraphQL-a [11].

Pogodni alati su: requests, http.client, Postman.

Primjer: Dohvaćanje vremenskih podataka s OpenWeather API-ja.

2.2.5. Headless Browser Scraping

Ovakav način ekstrakcije omogućava učitavanje web stranica i JavaScript-a bez otvaranja prozora preglednika, odakle mu potječe naziv „headless“ (u prijevodu „bez grafičkog sučelja“). Time omogućava ekstrakciju dinamičkih stranica bez potrebe za prikazivanjem stranice. Međutim, zahtjeva mnogo resursa te je znatno sporiji od prethodnika, a imate veće šanse da budete blokirani [12], [13].

Pogodni alati su: Selenium, Puppeteer, Playwright.

Primjer: Ekstrakcija stranice koja ima „beskonačni scroll“, odnosno sadržaj se učitava kako se korisnik pomiče prema dolje.

2.2.6. Ekstrakcija putem RSS Feeda

Mnoge web stranice nude RSS *feed-ove* koji omogućavaju korisnicima da automatski dobijaju ažurirane informacije. RSS *feed-ovi* su obično dostupni u strukturiranom XML formatu. Omogućen je jednostavan pristup ažuriranim strukturiranim podacima te su umanjene šanse da budemo blokirani. Nedostatak ovog pristupa nalazi se u ograničenim podacima koji su dostupni u RSS *feed-u* [14], [15].

Pogodni alati su: Feedparser, requests.

Primjer: Ekstrakcija blogova, vijesti ili ažuriranih članaka s web stranica koje pružaju RSS feedove.

2.2.7. OCR ekstrakcija

Podaci na stranicama mogu biti prikazani kao slike. Kako bi ekstrahirali takve podatke, potrebni su nam pristupi poput OCR (Optical Character Recognition) pristupa. OCR alati analiziraju slike te ih pretvaraju u tekstualne podatke kojima je jednostavnije upravljati. Ovakav pristup omogućuje i ekstrakciju podataka s CAPTCH-a ili skeniranih dokumenata. Problem nastaje kada slike nije dovoljno velike kvalitete ili sadrži velike količine teksta [16].

Pogodni alati su: Tesseract, OpenCV.

2.2.8. Ekstrakcija sadržaja s proxy serverima i rotacijom IP adresa

Neke web stranice imaju zaštitne mehanizme koje detektiraju i blokiraju IP adrese koje pokušavaju ekstrahirati previše podataka u kratkom vremenskom periodu. Upotreba *proxy* servera i rotacija IP adresa omogućava izbjegavanje ovih blokada. Mana je potreba za *proxy* serverima te kompleksna implementacija [17].

Pogodni alati su: ProxyMesh, ScraperAPI, Luminati.

2.3. TIPIČNI PROBLEMI

Web scraping može biti moćan alat za automatsko prikupljanje podataka s internetskih stranica, ali u procesu *scrapinga* često nailazimo na različite probleme i izazove. Neke web stranice imaju zaštitne mehanizme protiv prekomjernog *scrapinga* koji detektiraju i blokiraju IP adrese koje šalju previše zahtjeva u kratkom vremenskom periodu. Ovo se radi kako bi se smanjilo opterećenje na serverima i spriječilo neovlašteno prikupljanje podataka. Problem se može riješiti korištenjem proxy servera, korištenjem VPN-a za rotaciju IP adresa, ograničavanjem brzine zahtjeva te uključivanjem vremenskih pauza između svakog zahtjeva.

Neki serveri imaju datoteku „robots.txt“ koja regulira što web *scraperi* smiju prikupljati, a što je zabranjeno. Iako ovaj problem nije tehničke prirode, mogući su pravni postupci protiv prekršitelja stoga je nužno provjeriti pravila web lokacije.

Mnoge moderne stranice koriste JavaScript za dinamičko učitavanje podataka, što znači da podaci nisu odmah dostupni u HTML kodu stranice kada se stranica učita. Sadržaj se može prikazivati tek nakon što JavaScript izvrši dodatne zahtjeve prema serveru (npr. AJAX pozive). Upotrebom alata za izvršavanje JavaScript-a, ovaj problem je rješiv, ako ne postoji jednostavnije rješenje poput pristupa putem API-ja.

Paginacija je problem s kojim se najčešće možemo susresti ekstrahirajući podatke s web shopova. Paginacijom se podaci prikazuju na više stranica, što otežava ekstrakciju jer svaka stranica mora biti ekstrahirana odvojeno. Rješenje se nalazi u automatskom prolaženju kroz stranice putem programskih petlji ili korištenjem alata poput Selenium-a za simulaciju „beskonačnog scroll-a“.

CAPTCHA i anti-bot mehanizmi dodatno otežavaju *scraping*. Stranice koriste ove mehanizme kako bi se zaštitile od automatiziranih alata, tražeći od korisnika da unesu tekst sa slike ili prepoznaju objekte na slici. Rješenje može biti korištenje OCR tehnologije za automatsko prepoznavanje teksta ili servis koji omogućava zaobilaženje CAPTCHA izazova.

Pored ovih tehničkih prepreka, *scraping* se suočava s problemima nesređenih podataka i promjenjive HTML strukture. Stranice sa složenim i neurednim HTML-om ili čestom promjenom dizajna mogu otežati *scraping*, jer skripte koje funkcioniraju danas, sutra mogu biti neupotrebljive. Zato se koriste robusni alati poput BeautifulSoup-a koji mogu raditi s nevažećim HTML-om i fleksibilne metode za prepoznavanje elemenata.

„Rate limiting“ još jedna je zaštita koju koriste web stranice. Ograničava broj zahtjeva koje jedan korisnik ili IP adresa može poslati u određenom vremenskom razdoblju. Kako bi se to zaobišlo, *scraperi* uvode pauze između zahtjeva ili koriste proxy poslužitelje za distribuciju zahtjeva s različitih IP adresa.

Također, česte promjene u HTML strukturi ili URL shemama na stranicama mogu uzrokovati neuspjeh skripti za ekstrakciju. Web stranice se često ažuriraju, što znači da se programski kod *scrapera* mora redovito ažurirati kako bi bio u toku s tim promjenama.

Pravni problemi također su česti u *scrapingu*, budući da mnoge web stranice izričito zabranjuju prikupljanje podataka u svojim uvjetima korištenja. Ignoriranje ovih zabrana može dovesti do zakonskih sankcija. Osim pravnih, tu su i etička pitanja, poput prikupljanja osobnih podataka korisnika bez njihova dopuštenja. Konačno, struganje može naići na tehničke prepreke kao što su komprimirani podaci ili enkripcija, gdje su podaci skriveni ili šifrirani kako bi im se teško pristupilo. U takvim situacijama potrebno je koristiti alate za dekompresiju ili dešifriranje podataka prije nego što budu dostupni za ekstrakciju.

3. ALATI ZA SISTEMATIZIRANU EKSTRAKCIJU

Alati za automatizaciju ekstrakcije omogućuju efikasnije prikupljanje, analiziranje i upravljanje informacijama. Ova cjelina istražuje najistaknutije alate za sistematiziranu ekstrakciju podataka, svaki sa specifičnim prednostima i područjima primjene. Opisani alati pružaju niz funkcionalnosti – od analize strukture HTML i XML dokumenata, preko automatizacije web preglednika, do prepoznavanja teksta na slikama.

U ovom dijelu obrađeni su alati poput BeautifulSoup, jednostavne biblioteke za raščlanjivanje HTML i XML datoteka, te lxml, snažan alat za rad s XML-om, koji kombinira brzinu i fleksibilnost. Zatim slijede alati poput Scrapy i Selenium, koji omogućuju ekstrakciju i automatizaciju preglednika, te Puppeteer i Playwright, koji su osmišljeni za kontrolu i automatizaciju web preglednika na visokoj razini. Također se razmatraju i specijalizirani alati poput Feedparser za preuzimanje i analizu sindiciranih feedova te Tesseract, koji služi za prepoznavanje teksta iz slika. Na kraju, opisani su alati kao što je Octoparse, koji omogućuje ekstrakciju podataka bez potrebe za kodiranjem.

3.1. BeautifulSoup

BeautifulSoup je Python biblioteka koja omogućuje lako izdvajanje podataka iz HTML i XML datoteka. Radeći u kombinaciji s različitim *parserima*, omogućava jednostavno pretraživanje, navigaciju i modifikaciju stabla strukture dokumenta. Programerima često skraćuje vrijeme koje bi inače utrošili na ručno analiziranje podataka. Biblioteka nudi brojne mogućnosti, uključujući pretragu, navigaciju i manipulaciju podacima, te može olakšati rješavanje problema prilikom rada s neočekivanim formatima podataka [18].

3.2. Lxml

Lxml je Python alat razvijen za rad s XML-om, a koristi C biblioteke „libxml2“ i „libxslt“ za pružanje brze i učinkovite obrade XML dokumenata. Kombinira brzinu ovih C biblioteka s intuitivnim i fleksibilnim Python sučeljem, što ga čini snažnijim u odnosu na standardne alate kao što je „ElementTree“. Kompatibilan je s različitim verzijama Pythona, a namijenjen je korisnicima koji trebaju naprednu obradu XML-a, uz jednostavnost rada u Pythonu [19].

3.3. Scrapy

Scrapy je alat visoke razine razvijen za brzo i efikasno struganje web stranica i prikupljanje strukturiranih podataka. Osim što omogućuje indeksiranje web stranica, može se koristiti za širok spektar zadataka, uključujući rudarenje podataka, praćenje web stranica te automatizirano testiranje. Scrapy je popularan alat u domenama gdje je potrebno kontinuirano prikupljanje i analiza podataka iz različitih izvora na webu [20].

3.4. Selenium

Selenium je projekt koji omogućuje automatizaciju web preglednika kroz skup alata i biblioteka. Ovaj okvir pruža podršku za simulaciju korisničkih interakcija u različitim preglednicima, omogućujući testiranje, automatizaciju i kontrolu preglednika na različitim platformama. Jezgra Seleniuma je WebDriver, sučelje koje omogućava pisanje skripti koje se mogu pokretati u više preglednika. Osim automatizacije preglednika, projekt okuplja stručnjake i entuzijaste koji kontinuirano unapređuju ovu tehnologiju [21].

3.5. Puppeteer

Puppeteer je JavaScript biblioteka koja pruža API visoke razine za kontrolu preglednika poput Chromea i Firefoxa putem protokola DevTools ili WebDriver BiDi. Zadano radi u "bezglavnom" načinu rada, što znači da preglednik nije vidljiv korisniku, ali se može podesiti da radi u standardnom ("s glavom") načinu. Puppeteer je koristan alat za automatizaciju zadataka u preglednicima, kao što su testiranje web aplikacija ili prikupljanje podataka [22].

3.6. Playwright

Playwright je razvijen s naglaskom na end-to-end testiranje web aplikacija, ali se može koristiti i za opću automatizaciju preglednika. Pruža podršku za sve glavne preglednike kao što su Chromium, WebKit i Firefox, te omogućuje testiranje na različitim operativnim sustavima, uključujući Windows, Linux i macOS. Nudi snažne API-je koji omogućuju detaljnu kontrolu nad preglednicima, kao i emulaciju mobilnih uređaja, što ga čini fleksibilnim alatom za različite svrhe [23].

3.7. XPath

XPath je jezik dizajniran za navigaciju kroz XML dokumente i omogućuje selektiranje i prepoznavanje dijelova dokumenta na temelju definiranih putanja. Iako je prvenstveno razvijen

za rad s XML-om, može se koristiti i za druge formate koji koriste sličnu hijerarhijsku strukturu, poput HTML-a i SVG-a. XPath je često ključan alat u XSLT transformacijama, ali se također može koristiti za efikasnije upravljanje elementima u DOM-u, u usporedbi s tradicionalnim metodama kao što su „getElementById“ ili „querySelectorAll“ [24].

3.8. Feedparser

Feedparser je Python modul namijenjen preuzimanju i parsiranju različitih vrsta sindiciranih feedova, uključujući RSS i Atom. Ovaj alat može rukovati brojnim verzijama tih formata, kao i popularnim proširenjima poput Dublin Corea i iTunes modula. Feedparser je jednostavan za korištenje i dizajniran je da bude dio većeg Python programa, gdje preuzima feedove putem URL-a ili lokalnih datoteka te ih pretvara u lako dostupne podatke [25].

3.9. Tesseract

Tesseract je open-source OCR (optical character recognition) alat koji omogućuje prepoznavanje teksta iz slika. Razvijen pod licencom Apache 2.0, Tesseract podržava širok raspon jezika i može se koristiti kroz naredbeni redak ili API. Iako nema vlastito grafičko korisničko sučelje, postoji niz dostupnih vanjskih alata i dodataka. Tesseract je visoko prilagodljiv i može se koristiti u različitim okruženjima, uključujući mobilne uređaje, što ga čini moćnim rješenjem za digitalizaciju tiskanog teksta [26].

3.10. Octoparse

Octoparse je alat za skrapiranje podataka s web stranica bez potrebe za programiranjem. Korisnicima omogućuje da jednostavnim klikovima pretvore sadržaj web stranica u strukturirane podatke. Njegovo intuitivno sučelje čini ga pristupačnim alatima za korisnike bez tehničkog znanja, omogućujući brzo i efikasno prikupljanje i organizaciju podataka s interneta [27].

4. PROBLEMI I CILJEVI

Jedan od ključnih problema s kojima se suvremeni potrošači suočavaju prilikom online kupovine je pronalazak najpovoljnijih cijena proizvoda u uvjetima visoke inflacije i rastućih životnih troškova. Internetske trgovine nude veliki broj proizvoda, ali i cijene koje variraju od trgovine do trgovine. Iako potrošači teoretski imaju mogućnost istražiti i usporediti cijene na različitim platformama, taj proces može biti izrazito vremenski zahtjevan i složen zbog prekomjerne količine podataka, nepouzdanih izvora te potencijalnih prevara.

Osim toga, dinamična priroda internetskih cijena, gdje one mogu često fluktuirati, dodatno otežava korisnicima praćenje i usporedbu ponuda. U tom kontekstu, javlja se potreba za automatiziranim sustavom koji će olakšati korisnicima pretragu, usporedbu i pronalazak najboljih ponuda na siguran i učinkovit način.

Cilj ovog projekta je izraditi sustav koji putem web *scrapinga* omogućuje automatsko prikupljanje podataka o cijenama proizvoda iz različitih web trgovina te usporedbu tih cijena u realnom vremenu. Projekt je trenutno usmjeren na analizu ponude televizora, no sustav je dizajniran kao skalabilan te ga je moguće proširiti na druge kategorije proizvoda. Za prikupljanje podataka koriste se tri web *scrapera* implementirana u programskom jeziku Python, pri čemu svaki *scraper* prikuplja podatke s različite internetske trgovine.

Uz prikupljanje podataka, projekt uključuje i razvoj web aplikacije temeljene na okviru Flask, koja omogućuje korisnicima jednostavan i intuitivan pregled prikupljenih informacija. Aplikacija korisniku prikazuje proizvode koji su dostupni u različitim trgovinama, uspoređuje njihove cijene te identificira trgovinu s najpovoljnijom ponudom. Time se korisnicima omogućuje brže i jednostavnije donošenje informiranih odluka prilikom online kupovine.

Glavni ciljevi ovog projekta su:

- Razviti zasebne funkcionalne web *scrapere* za prikupljanje podataka iz različitih trgovina, unutar jednog računalnog programa.
- Dizajnirati i kreirati bazu za pohranu podataka.
- Implementirati web aplikaciju koja će podatke prezentirati na pregledan i korisnicima prihvatljiv način.

- Omogućiti usporedbu cijena istih proizvoda na različitim platformama te identificirati najpovoljnije ponude.

- Osigurati skalabilnost sustava kako bi se u budućnosti mogao proširiti na druge kategorije proizvoda i dodatne trgovine.

Planirane tehnike u ovom projektu:

- Statička web ekstrakcija – projekt se oslanja na statičku web ekstrakciju jer se podaci prikupljaju direktno iz HTML-a web stranica pomoću HTTP zahtjeva (requests.get), bez interakcije s dinamičkim elementima koji bi zahtijevali izvršavanje JavaScript-a.
- DOM Parsing – korištenje biblioteke BeautifulSoup za parsiranje HTML strukture web stranice označava tehniku DOM parsinga. Koristit će se funkcije kao što su find i find_all za pretraživanje specifičnih HTML elemenata (npr. article elementi za proizvode).

Planirani alati koji će se koristiti u projektu:

- Requests – Python biblioteka requests koristi se za slanje HTTP zahtjeva prema web stranicama i preuzimanje njihovog HTML sadržaja. Ovo je standardni alat za dohvaćanje statičkih podataka.
- BeautifulSoup – Python biblioteka BeautifulSoup koristi se za parsiranje HTML-a. Ona olakšava rad s DOM-om (struktura dokumenta) i omogućuje pretraživanje specifičnih elemenata na stranici pomoću različitih metoda (find, find_all).
- SQLite – za spremanje prikupljenih podataka koristi se SQLite baza podataka. U ovom slučaju, prikupljeni podaci o proizvodima (npr. ime, cijena, veličina zaslona) pohranjuju se u SQLite bazu putem SQL upita.
- RegEx (Regularni izrazi) – regularni izrazi (pomoću Python modula re) koriste se za izdvajanje specifičnih informacija iz tekstualnih stringova, kao što su cijene ili veličine zaslona. Na primjer, u funkciji extract_screen_size, koristi se regularni izraz za pronalazak brojeva koji označavaju veličinu zaslona.

5. RAZVOJ APLIKACIJE

Aplikacija se sastoji od *backend* dijela i *frontend* dijela. U pozadini se pokreću Python skripte koje prikupljaju podatke o proizvodima s unaprijed definiranih web lokacija te Python skripta koja uspoređuje dobivene podatke te traži iste proizvode na temelju šifre proizvoda. Kada se utvrdi da su proizvodi iz različitih trgovina isti, prioritet ide trgovini gdje je proizvod jeftiniji.

Korisničko sučelje realizirano je programskim jezicima Python uz pomoć alata Flask te HTML-a. Učitavaju se podaci o najjeftinijim proizvodima te se prikazuju u obliku kartica, po tri u svakom redu. Moguće ih je sortirati po određenom atributu ili filtrirati samo željenog proizvođača. Svaka kartica sadrži sliku i naziv proizvoda, opis proizvoda, cijenu te popis trgovina s većom cijenom, ako je isti dostupan u njima.

Ovakav dizajn omogućava neovisnost između *backend-a* i *frontend-a*, što znači da se svaki od tih dijelova može uređivati odvojeno. Prilikom razvoja koda, pazilo se da aplikacija bude skalabilna, odnosno da se novi *scraperi* za nove trgovine mogu jednostavno implementirati bez potrebe za izmjenom postojećeg koda ili dizajna baze podataka.

5.1. POSTAVLJANJE RADNOG OKRUŽENJA

Za početak rada na projektu potrebno je postaviti radno okruženje, odnosno instalirati Python, Python razvojno okruženje i PowerShell.

Python je interpretirani programski jezik, visoke razine, orijentiran na objekte, poznat po svojoj dinamičkoj semantici. Njegove napredne ugrađene strukture podataka, uz podršku za dinamičko tipkanje i vezivanje, čine ga idealnim za brzi razvoj aplikacija te za upotrebu kao programski jezik za integraciju postojećih komponenti. Zbog jednostavne i lako razumljive sintakse, Python olakšava čitljivost koda i smanjuje troškove održavanja softvera. Također podržava module i pakete, omogućujući modularni pristup razvoju i ponovnu uporabu koda. Python tumač i njegova bogata standardna biblioteka dostupni su besplatno na svim glavnim platformama [28].

Link za preuzimanje: https://www.python.org/downloads/

Dokumentacija: https://docs.python.org/3/

Python razvojno okruženje koje je korišteno prilikom izrade ovog projekta je PyCharm Community Edition 2021.3.3. „PyCharm Community Edition besplatan je projekt koji je

izgrađen na otvorenom kodu i podržava osnovni razvoj Pythona,…“[29] Navedeni alat pogodan je za ovaj projekt zbog sljedećih značajki:

- mogućnost naprednog uređivanja koda
- ugrađeni alat za otklanjanje pogrešaka
- virtualna okruženja i upravljanje paketima
- ugrađena podrška za testiranje
- podrška za rad s bazama podataka
- praćenje verzija (VCS/Git integracija)
- strukturiranje i organizacija projekta

Link za preuzimanje: https://www.jetbrains.com/pycharm/download/other.html

Dokumentacija: https://www.jetbrains.com/help/pycharm/getting-started.html

„PowerShell je višeplatformsko rješenje za automatizaciju zadataka sastavljeno od ljuske naredbenog retka, skriptnog jezika i okvira za upravljanje konfiguracijom. PowerShell radi na sustavima Windows, Linux i macOS.“[30] Pogodan je za ovaj projekt zbog jednostavnog upravljanja virtualnim okruženjem koristeći jednostavne naredbe za kreiranje, aktiviranje i deaktiviranje okruženja.

Kreiranje virtualnog okruženja:

```
python -m venv env
```

Navedena naredba kreira Python virtualno okruženje pod nazivom „env“.

Aktiviranje virtualnog okruženja:

```
.\env\Scripts\Activate
```

Navedena naredba aktivira virtualno okruženje.

Link za preuzimanje: https://learn.microsoft.com/en-us/powershell/scripting/install/installing-powershell-on-windows?view=powershell-7.4

Dokumentacija: https://learn.microsoft.com/en-us/powershell/

Kako bi radno okruženje bilo postavljeno do kraja i spremno za realizaciju projekta, potrebno je instalirati Python pakete, „requests“ za slanje HTTP zahtjeva i dobivanje HTML sadržaja

stranice te „BeautifulSoup“ za parsiranje (analizu) HTML-a i izdvajanje relevantnih podataka. Najjednostavniji način je pomoću PowerShell-a pozicionirat se u radnom direktoriju te unijeti sljedeću naredbu:

```
pip install requests beautifulsoup4
```

„Unutar naredbe, „pip“ predstavlja upravitelja Python paketima. Paket sadrži sve datoteke koje su potrebne za modul. Moduli su biblioteke Python koda koje je moguće uključiti u projekt.“[31]

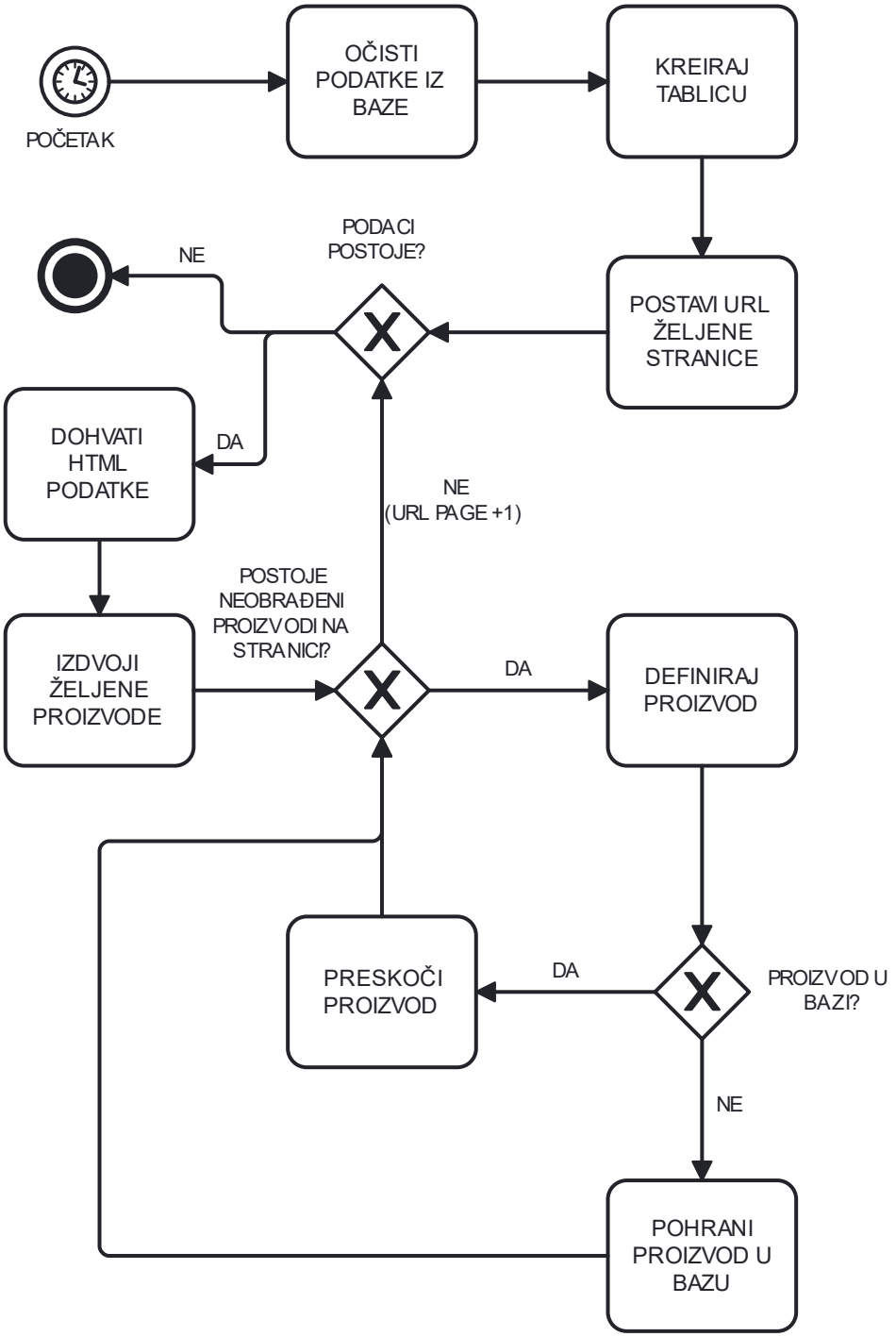
5.2. DIZAJN BAZE PODATAKA

Prije početka pisanja koda, potrebno je isplanirati bazu podataka. U projektu je korištena SQLite baza podataka. Baza podataka sastoji se od tablica. U svaku tablicu spremaju se podaci o televizijama s određene stranice.

Entitet koji se sprema je TV, a atributi su sljedeći:

- id (Integer, PRIMARY KEY) – jedinstveni identifikator svakog proizvoda
- name (String, UNIQUE NOT NULL) – ime proizvoda
- price (Float, NOT NULL) – cijena proizvoda
- screen_size (Integer, NOT NULL DEFAULT 0) – veličina zaslona u inčima
- manufacturer (String, NOT NULL DEFAULT 'Other') – ime proizvođača (npr. Samsung, LG, itd.)
- screen_type (String, NOT NULL DEFAULT 'Other') – vrsta zaslona (npr. LED, OLED, itd.)
- tv_code (String, NOT NULL DEFAULT 'Unknown') – jedinstvena oznaka koja se koristi u trgovinama za identifikaciju proizvoda
- product_link (String, NOT NULL DEFAULT 'Unknown') – link na stranicu proizvoda
- image_link (String, NOT NULL DEFAULT 'Unknown') – link na sliku proizvoda
- store (String, NOT NULL DEFAULT {ime trgovine}) – ime trgovine (npr. Sancta Domenica, Instar, itd.)

5.3. RAZVOJ KODA ZA SISTEMATIZIRANU EKSTRAKCIJU



Slika 2 Dijagram toka programa

Nakon dizajna baze podataka potrebno je definirati funkciju, unutar Python koda za *scraping*, koja će kreirati tablicu unutar baze podataka.

Prilikom kreiranja tablice važno je definirati ograničenja da bi se zadržao integritet baze podataka. Neka od ograničenja su:

- PRIMARY KEY – svaka tablica u SQLite-u može imati najviše jedan PRIMARY KEY. Pomoću njega može se pristupiti podacima određenog zapisa jer je jedinstven. Uglavnom je to „id“.[32]
- UNIQUE – sprječava da dva zapisa imaju identične vrijednosti u određenom stupcu.[32]
- NOT NULL – prema zadanim postavkama, stupac može sadržavati NULL vrijednosti. Kako bi se spriječilo, treba definirati takvo ograničenje za ovaj stupac navodeći da NULL sada nije dopušteno za taj stupac. NULL nije isto što i „nema podataka“, već predstavlja nepoznate podatke.[32]
- DEFAULT – daje zadanu vrijednost stupcu kada izjava INSERT INTO ne daje određenu vrijednost.[32]
- CHECK – omogućuje uvjet za provjeru vrijednosti koja se unosi u zapis. Ako se uvjet ocijeni netočnim, zapis krši ograničenje i ne unosi se u tablicu.[32]

productsV3 | Table name: productsSanctaDomenica | WITHOUT ROWID | STRICT

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	id	INTEGER								*NULL*
2	name	TEXT								*NULL*
3	price	FLOAT								*NULL*
4	screen_size	INTEGER								0
5	manufacturer	TEXT								Other
6	screen_type	TEXT								Other
7	tv_code	TEXT								Unknown
8	product_link	TEXT								Unknown
9	image_link	TEXT								Unknown
10	store	TEXT								Sancta Domenica

Slika 3 Tablica za trgovinu Sancta Domenica

Prvi korak prilikom ekstrakcije sadržaja je slanje HTTP zahtjeva i dohvaćanje HTML-a stranice. Tu se pojavio prvi problem ovog projekta. Lokacija koja je bila meta ekstrakcije sadržaja je Pevex.hr[1]. Međutim ova web lokacija zabranila je ekstrakciju te su svi HTTP zahtjevi bili blokirani. Jedino preostalo rješenje bilo je pronaći novu web lokaciju za ekstrakciju sadržaja.

[1] https://www.pevex.hr/elektronika/televizori-i-dodatci/televizori

Nova lokacija bila je Sancta Domenica[2]. Prilikom slanja zahtjeva nije bilo nikakvih problema te je *scraper* imao jednostavan pristup svim HTML podacima. Ispis koda može se vidjeti ispod.

```
def fetch_page(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99 Safari/537.36"
    }

    response = requests.get(url, headers=headers)

    if response.status_code == 200:
        return BeautifulSoup(response.text, "html.parser")
    else:
        print(f"Error: Unable to fetch page (status code
{response.status_code})")
        return None


# Testiraj s URL-om
url = "https://www.sancta-domenica.hr/televizori/led-tv.html"
soup = fetch_page(url)
print(soup.prettify())  # Ispisuje strukturirani HTML stranice
```

Sada je cijeli HTML kod preuzet i pohranjen u varijablu „soup“.

Pomoću BeautiflSoup-a iz varijable „soup“ izvlače se svi željeni podaci. Podaci se pronalaze na temelju HTML klasa koje sadrže željene podatke. Za to je zadužena funkcija *extract_products(soup)*.

Funkcija prima BeautifulSoup objekt, iz kojeg prvo pronalazi sve proizvode unutar <li> tagova s klasom "item product product-item". Zatim iterira kroz svaki od tih elemenata i iz njih izdvaja ključne informacije poput imena proizvoda, cijene, linkova na proizvod i sliku, kao i dodatne karakteristike poput veličine ekrana, proizvođača, tipa ekrana i koda proizvoda.

Prilikom izdvajanja neke podatke bilo je jednostavno pronaći i zapisati, poput „name“, „product_link“ i „image_link“, jer je potrebno točno u onakvom obliku u kakvom se nalazi u

[2] https://www.sancta-domenica.hr/televizori/led-tv.html

HTML-u. Novi problem nastao je prilikom izvlačenja cijena jer su u HTML-u zapisane u formatu koji SQLite baza podataka ne prihvaća, pogotovo za cijene iznad 1000€. Cijena je zapisana u obliku „1.000,00 €“, dok SQLite baza zahtjeva format „1000.00 €“. Iz navedenog može se zaključiti da problem stvaraju točka, koja označuje tisućice, i zarez, koji označuje cente (decimale). Zbog toga je bilo potrebno napraviti funkciju koja će preoblikovati format cijene na način da ukloni točku za tisućice te zarez zamijeni decimalnom točkom.

```
# Function to parse the price string into a float
def parse_price(price_str):
    # Match price with or without thousands separator and with a decimal
comma
    match = re.search(r'\d{1,3}(?:\.\d{3})*(?:,\d+)?', price_str)
    if match:
        cleaned_price = match.group().replace(".", "").replace(",", ".")
        return float(cleaned_price)
    else:
        raise ValueError(f"Could not parse price string: {price_str}")
```

Podaci za veličinu zaslona, tip zaslona, proizvođača te TV kod (šifra proizvoda) izvučeni su iz imena TV-a (npr. iz „TV 65" Samsung QLED 65Q60D“).

Funkcija za izvlačenje veličine zaslona pokušava izvući veličinu zaslona (dijagonalu u inčima) iz imena proizvoda. Koristi regularne izraze (re.search()) kako bi pronašla broj koji predstavlja veličinu zaslona u inčima. Prvo se traži uzorak broja koji je praćen znakom " (dva okomita navodnika), npr. "55" za TV dijagonale 55 inča. Ako prvi uzorak ne pronađe ništa, traži isti broj, ali uzorak koji koristi dva apostrofa (' '), jer se u nekim zapisima inči označavaju s dva apostrofa, npr. "55' ' ". Ako se pronađe podudaranje, broj se vraća kao int, a ako ne, vraća se -1, što označava da nije pronađena veličina zaslona.

Funkcija za izvlačenje proizvođača pokušava identificirati proizvođača TV-a na temelju imena proizvoda. Sadrži popis poznatih proizvođača (manufacturers), poput "Samsung", "LG", "Sony", itd. Prolazi kroz ovaj popis i provjerava je li bilo koji od tih proizvođača spomenut u imenu. Ako pronađe proizvođača, vraća njegovo ime. Ako ne pronađe nijednog proizvođača s popisa, funkcija vraća "Other", što označava da proizvođač nije prepoznat.

Funkcija za izvlačenje vrste zaslona identificira tip zaslona (npr. OLED, QLED) iz imena proizvoda. Funkcija ima popis različitih vrsta zaslona, kao što su "FULL HD", "OLED",

"QLED", "LED", itd. Prolazi kroz ovaj popis i provjerava je li bilo koji tip zaslona spomenut u imenu proizvoda (pretvarajući ime u velika slova kako bi pretraga bila neosjetljiva na mala/velika slova). Ako pronađe tip zaslona, vraća ga. Ako ne pronađe nijedan prepoznatljiv tip zaslona, funkcija vraća "Other" kako bi označila da nije prepoznat zaslon.

Funkcija za izvlačenje tv koda pokušava izvući kod televizora (TV code) iz imena proizvoda, koristeći ime proizvođača. Funkcija koristi regularni izraz (re.search) kako bi našla dio imena koji dolazi nakon imena proizvođača. To obično uključuje modelni broj televizora. Ako se pronađe podudaranje, izdvoji tekst nakon imena proizvođača, koji predstavlja modelni kod. Zatim, koristi .split() da razdvoji String na dijelove koristeći praznine (space) kao razdjelnike. Filtrira dijelove tako da uklanja one koji su samo slova, kraći su od 4 znaka, ili sadrže zareze. Ovo pomaže u izdvajanje stvarnog modelnog broja. Ako nakon filtriranja postoje neki dijelovi, oni se spajaju natrag u jedan string koji predstavlja modelni broj televizora, odnosno tv kod. Ako nema pronađenog modelnog broja, funkcija vraća "Unknown".

Prije nego što se novi podaci pohrane, često je korisno obrisati stare podatke iz baze, kako bi se izbjeglo dupliciranje ili rad s neaktualnim informacijama. Funkcija „clear_data()“ briše sve podatke iz tablice productsSanctaDomenica. Otvara vezu prema bazi podataka, briše sve postojeće zapise iz tablice koristeći SQL naredbu DELETE te nakon što su svi podaci obrisani, funkcija zatvara vezu prema bazi podataka.

Funkcija insert_data() unosi podatke o pojedinom proizvodu u bazu podataka. Funkcionira na način da prvo otvara vezu prema bazi podataka "productsV3.db". Nakon toga pokušava unijeti podatke o proizvodu u tablicu *productsSanctaDomenica*. Podaci uključuju ime proizvoda, cijenu, veličinu ekrana, proizvođača, tip zaslona, model (TV kod), URL proizvoda, URL slike i naziv trgovine ("Sancta Domenica"). Ako je proizvod već prisutan (jer je stupac name označen kao UNIQUE), dogodit će se sqlite3.IntegrityError, čime se funkcija neće srušiti, već će preskočiti unos tog proizvoda i ispisati poruku: "Product '{name}' already exists in the database. Skipping.". Kada je unos završen ili kada je proizvod već postojao, funkcija zatvara vezu prema bazi podataka.

Kada su definirane sve potrebne funkcije, može se započeti razvoj finalnog koda *scrapera*. Kod se izvršava unutar „while“ petlje koja se ponavlja sve dok varijabla „current_url“ ima vrijednost. Ova varijabla sadrži URL trenutne stranice proizvoda koju skripta obrađuje. Ako ne postoji više stranica za obradu, „current_url“ postaje „None“, čime se petlja zaustavlja. Funkcija „fetch_page(current_url)“ koristi biblioteku „requests“ za preuzimanje HTML

sadržaja trenutne stranice s URL-om koji se nalazi u „current_url“. Zatim se koristi BeautifulSoup za parsiranje tog HTML sadržaja i kreira se objekt „soup“ koji olakšava navigaciju i pretraživanje HTML-a. Funkcija „extract_products(soup)“ analizira parsirani HTML sadržaj (proslijeđen kao „soup“) i pronalazi sve proizvode na stranici. Funkcija vraća popis proizvoda, gdje je svaki proizvod predstavljen kao „tuple“ s podacima o:

- nazivu proizvoda,
- cijeni,
- veličini ekrana,
- proizvođaču,
- tipu zaslona,
- kodu proizvoda (TV kod),
- URL-u proizvoda,
- URL-u slike proizvoda.

Petlja prolazi kroz svaki proizvod iz liste „products“. Za svaki proizvod izdvajaju se vrijednosti koje predstavljaju informacije o imenu, cijeni, veličini ekrana, proizvođaču, tipu zaslona, TV kodu, linku na proizvod i linku na sliku. Funkcija „product_exists(name)“ provjerava postoji li već zapis u bazi podataka za proizvod s tim imenom. Ako proizvod ne postoji u bazi („if not“), nastavlja se proces unosa podataka. Ako proizvod ne postoji u bazi, poziva se funkcija „insert_data()“ koja unosi podatke o proizvodu u SQLite bazu. Svi relevantni podaci (ime, cijena, veličina ekrana itd.) šalju se funkciji, a ona zatim unosi te podatke u tablicu productsSanctaDomenica.

Ako je proizvod uspješno unesen u bazu, podaci o njemu ispisuju se u konzolu. Ispisuje se ime, cijena (formatirana na dvije decimale), veličina ekrana, proizvođač, tip zaslona, TV kod, URL proizvoda i URL slike. Ako je proizvod već prisutan u bazi podataka (tj. ako funkcija „product_exists(name)“ vrati „True“), ispisuje se poruka koja ukazuje da je proizvod preskočen. Nakon što su svi proizvodi s trenutne stranice obrađeni, koristi se BeautifulSoup objekt „soup“ kako bi se pronašao link na sljedeću stranicu. Traži se HTML element <a> s klasom „action next“, koja obično označava dugme ili link za sljedeću stranicu u paginaciji.

Ako je pronađen link na sljedeću stranicu („next_page_link“ nije „None“), „current_url“ se ažurira novim URL-om koji pokazuje na sljedeću stranicu proizvoda. Tako se ciklus ponavlja i preuzima se sljedeća stranica. Ako link na sljedeću stranicu ne postoji (tj. lociran je na zadnjoj stranici), „current_url“ se postavlja na „None“, čime se petlja zaustavlja.

Kada više nema stranica za procesuiranje, petlja završava, a ispisuje se poruka koja potvrđuje da su svi podaci uspješno pohranjeni u bazu podataka.

Ovaj kod automatizira prikupljanje proizvoda s web-stranice (paginirane strukture), obrađuje ih i pohranjuje u bazu podataka. Kroz više stranica proizvoda prolazi se pomoću linka na sljedeću stranicu, a provjerava se i dupliciranje kako bi se spriječilo više unosa istog proizvoda. Sve ove operacije se odvijaju unutar beskonačne „while“ petlje, koja se zaustavlja kad više ne postoji sljedeća stranica za obradu. Ista logika korištena je za ekstrakciju sadržaja s ostalih stranica, uz minimalne promjene zbog drugačije strukture HTML-a.

Kako bi ova aplikacija dobila smisao, potrebno je sve podatke iz različitih tablica u bazi usporediti kako bi se pronašli isti televizori u različitim trgovinama. Nakon što su pronađeni, prioritet dobivaju trgovine u kojima je televizor jeftiniji. Skuplje trgovine će također biti prikazane, ali neće biti istaknute kao najjeftinija. Za početak potrebno je definirati funkcije.

Funkcija „similarity()“ koristi Pythonov modul „difflib“ i njegovu funkciju „SequenceMatcher“ da izračuna sličnost između dvije tekstualne vrijednosti. Ova funkcija vraća vrijednost između 0 i 1, gdje je 1 potpuna sličnost, a 0 potpuna različitost.

```
# Function to calculate similarity between two strings
def similarity(a, b):
    return SequenceMatcher(None, a, b).ratio()
```

Funkcija „clear_data()“ briše sve postojeće podatke iz tablice „productsCompared“ u bazi podataka. To se radi kako bi se osiguralo da pri novom pokretanju koda uvijek radimo sa svježim podacima.

Nakon spajanja na bazu podataka potrebno je kreirati novu tablicu, ako nije kreirana, pod nazivom „productsCompared“ koja će biti identična prethodno spomenutim tablicama, uz dva nova stupca. U stupac „another_stores“ spremaju se podaci o ostalim trgovinama gdje je televizor nešto skuplji, a u stupac „stores_links“ spremaju se linkovi na te skuplje ponude televizora.

Nakon toga kreira se lista „tables“ s imenima svih tablica te prazna lista „all_tvs“ gdje će se spremiti svi televizori iz različitih stranica. Poziva se „clear_data()“ kako bi se izbrisali

prethodni podaci iz „productsCompared“ te se proizvodi iz svake tablice dohvaćaju naredbom SELECT * i spremaju u listu „all_tvs“.

Kada su svi televizori učitani može započeti uspoređivanje. Prvo se definira „processed“, skup koji prati indekse televizora koji su već obrađeni. Petlja prolazi kroz sve televizore iz liste „all_tvs“. Za svaki televizor (tv1):

- Ako je „tv_code“ tog televizora nepoznat ("Unknown"), odmah ga upisuje u „productsCompared“ bez daljnje usporedbe.
- Zatim traži televizore koji imaju isti ili sličan „tv_code“ koristeći uvjet da jedan „tv_code“ može biti podniz drugog ili da je njihova sličnost 100% (koristeći funkciju „similarity()“).
- Pronalazi sve televizore s istim kodom, sortira ih po cijeni (koristeći „key=lambda x: x[2]“ gdje je „x[2]“ cijena), te uzima najjeftiniji televizor („main_tv“).
- Zatim kreira listu trgovina („another_stores“) i linkova („stores_links“) gdje su pronađeni ostali televizori s istim kodom.
- Na kraju, najjeftiniji televizor i informacije o njemu upisuju se u tablicu „productsCompared“, zajedno s popisom trgovina i linkova.

Sve promjene u bazi podataka se potvrđuju naredbom „commit()“, a veza s bazom se zatvara naredbom „close()“. Nakon toga, ispisuje se poruka kako bi se potvrdilo da je proces usporedbe završen.

5.4. PRIKAZ EKSTRAHIRANIH PODATAKA

Ovaj projekt koristi Flask, Python web framework, za prikaz proizvoda iz SQLite baze podataka u web aplikaciji. Podaci se prikupljaju, filtriraju, sortiraju i prikazuju na stranici, a korisnici mogu odabrati različite opcije sortiranja i filtriranja pomoću HTML obrasca. Web aplikacija za prikaz podataka sastoji se od python dokumenta „app.py“ te pripada „backend“ logici, dok je za „frontend“ logiku zadužen „index.html“.

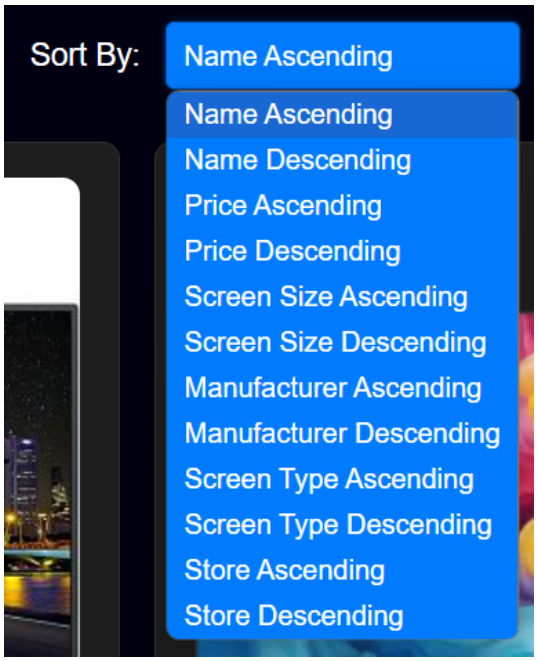
Prvi korak je inicijalizacija Flask aplikacije, što je moguće realizirati unutar jednog retka. Ovaj redak pokreće Flask aplikaciju. Flask(__name__) je instanca aplikacije koja će se koristiti za definiranje ruta i komunikaciju s korisnikom putem HTTP zahtjeva.

Zatim je potrebno kreirati funkciju za spajanje na bazu podataka. Funkcija „get_db_connection()“ otvara vezu prema SQLite bazi podataka definiranoj u varijabli

DATABASE. Korištenjem „sqlite3.Row“ kao „row_factory“, omogućuje se da rezultati SQL upita budu dostupni kao objekti tipa „dictionary“ (npr. row['name']), umjesto običnih tuplea.

Nakon toga kreirana je ruta „index()“. Ova funkcija obrađuje HTTP GET zahtjev na root URL-u ("/"). Na početku, iz URL parametara (request.args.get) dohvaća se korisnički odabir sortiranja (sort_option, s zadanim redoslijedom po ID-u) i proizvođača za filtriranje (selected_manufacturer, s zadanim "All" - tj. svi proizvođači).

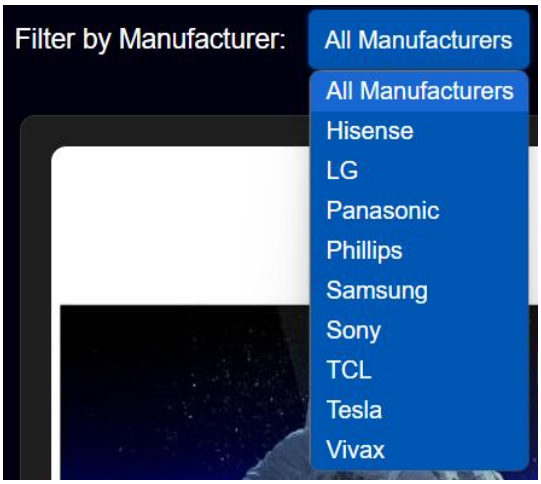
Jedna od važnih značajki je mogućnost sortiranja. „sort_options“ je „dictionary“ koji povezuje svaki parametar sortiranja s odgovarajućim SQL poljem i redoslijedom ("asc" za uzlazno, "desc" za silazno). Nakon toga, prema odabranoj opciji („sort_option“), dohvaćaju se odgovarajuće vrijednosti za sortiranje.



Slika 4 Mogućnosti sortiranja

Kada su funkcije definirane, može započeti spajanje na bazu podataka. Otvara se veza s bazom pomoću „get_db_connection()“ i dohvaćaju se svi jedinstveni proizvođači iz tablice „productsCompared“, koji će se prikazati u padajućem izborniku za filtriranje.

Da bi izvršili sortiranje i filtriranje potrebno je izvršiti upite prema bazi. SQL upit započinje sa selektiranjem svih proizvoda. Ako je korisnik odabrao određenog proizvođača, dodaje se uvjet WHERE manufacturer = ?. Ako ne, prikazuju se svi proizvodi. Također, dodaje se sortiranje na temelju odabranog parametra.



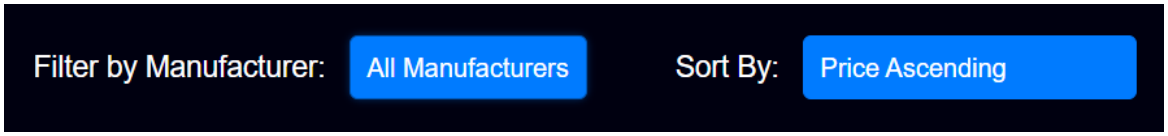
Slika 5 Mogućnost filtriranja po proizvođaču

Svi dohvaćeni podaci pohranjuju se u varijablu „rows“. Zatim se podaci za svaki proizvod pretvaraju u oblik „dictionary“ („row_dict“), čime je moguće lakše raditi s njima u HTML-u. Dodatno, „another_stores“ i „stores_links“ (ako postoje) dijele se i uparuju (npr. za prikaz linkova na alternativne prodavaonice).

Na kraju, koristi se funkcija „render_template()“ kako bi se generirala HTML stranica koristeći „index.html“ predložak. Uz to, prosljeđuju se procesirani podaci („rows“), trenutno odabrane opcije za sortiranje i proizvođače.

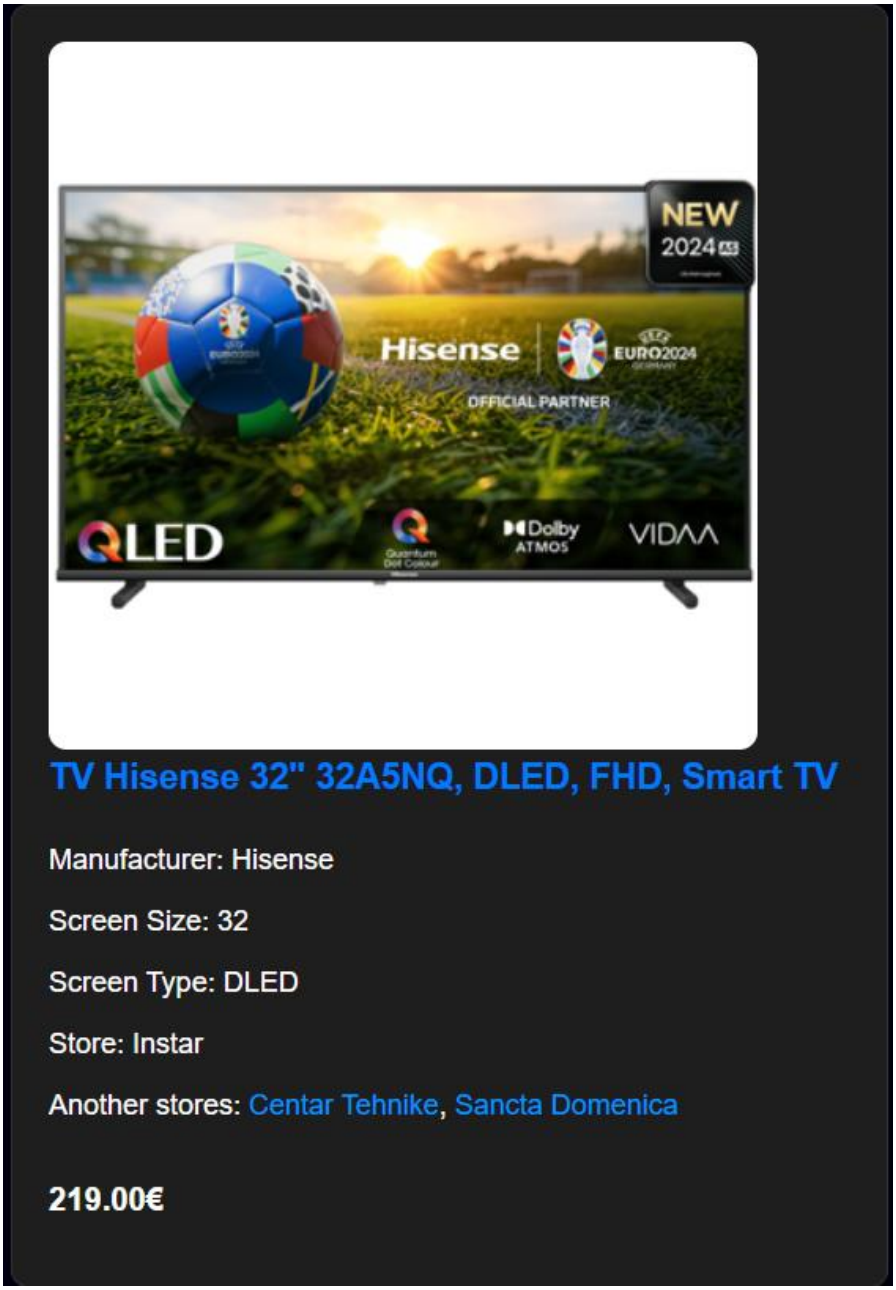
Paralelno s razvojem pozadinske logike aplikacije, razvija se i vizualni dio. Za to je zadužen „index.html“. HTML struktura počinje deklaracijom doctype-a i osnovnim postavkama za kodiranje i prikaz stranice. Uključuje se CSS za stilizaciju iz statičkih datoteka te JavaScript funkcija „sortProducts()“, koja omogućava ažuriranje URL-a prilikom promjene sortiranja ili proizvođača.

Ispod naslova nalaze se dva padajuća izbornika, jedan za filtriranje proizvođača, a drugi za sortiranje kartica s televizorima. Ovdje su definirana dva „select“ elementa za filtriranje proizvoda po proizvođaču i sortiranje po različitim atributima (ID, cijena, veličina ekrana itd.). Svaka promjena odabira automatski poziva funkciju „sortProducts()“, koja ažurira stranicu s novim parametrima URL-a.



Slika 6 Mogućnosti filtriranja i sortiranja

Svi proizvodi iz baze podataka prikazuju se u div elementima sa stilom "product-card". Svaki proizvod uključuje sliku, naziv (s linkom na izvorni proizvod), detalje poput veličine zaslona, proizvođača, tipa zaslona i cijene. Ako postoje dodatne trgovine, njihov popis s linkovima također se prikazuje.



Slika 7 Prikaz proizvoda u karticama

Kod za prikaz podataka iz baze o alternativnim trgovinama radi na principu postavljanja parova. Svaki par se sastoji od imena trgovine i linka na proizvod u toj trgovini. Parovi se formiraju na temelju podataka iz baze. Ti podaci zapisani su u stupcima „another_stores“ i „stores_links“. Spremljeni su na način da su unutar istog stupca podaci spremljeni u isto polje, ali odvojeni zarezom. Prilikom učitavanja podataka učitava se cijeli veliki string te se dijeli na manje

stringove separatorom zarez(,). Takav princip omogućava kasnije proširenje aplikacije gdje će biti dovoljno dodati nove trgovine, umjesto mijenjanja strukture tablice .

5.5. VREMENSKO POKRETANJE

Postavljanje vremenskog pokretanja skripti i aplikacija koje rade s Pythonom može biti ključan korak u automatizaciji procesa, posebice u projektima kao što je ovaj, gdje *scraperi* i aplikacija za usporedbu televizora moraju raditi u točno određenim intervalima jer se cijene televizora mijenjaju pa je potrebno redovito ažurirati podatke kako bi bili relevantni. Ovdje ćemo analizirati kako se ovaj proces može implementirati na dva sustava – lokalno na Windows operativnom sustavu i na Linux serveru – koristeći dostupne alate za automatizaciju zadataka. Kako bi se to olakšalo, sve Python datoteke za ekstrakciju sadržaja te Python datoteka za uspoređivanje televizora pokreću se unutar jedne Python datoteke pod nazivom „runApp.py“. Ovaj kod koristi paket „subprocess“ kako bi mogao pokrenuti Python datoteke.

```
import subprocess

# List of python files to run
python_files = ['scrapeCentarTehnikeWithDBv4.py',
                'scrapeINSTARwithDBv3.py',
                'scrapeSanctaDomenicaWithDBv3.py',
                'comparingTVs.py']

# Running each python file one by one
for file in python_files:
    subprocess.run(['python', file])
```

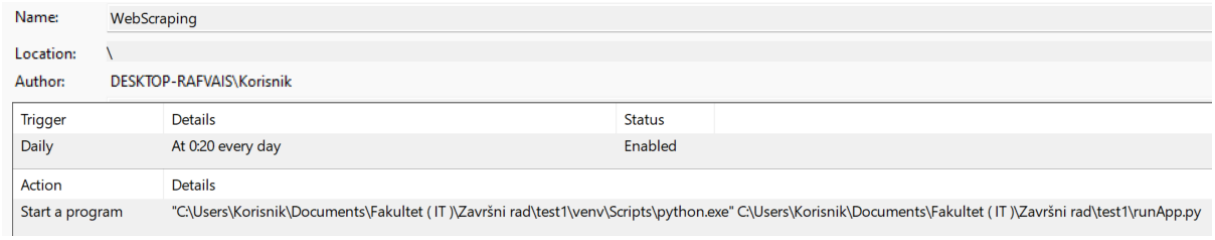
5.5.1. Vremensko pokretanje na Windows operativnom sustavu

Na Windows sustavu, najčešći način za postavljanje automatiziranih zadataka je korištenje ugrađenog alata Task Scheduler. Task Scheduler omogućuje kreiranje zadataka koji se izvršavaju u unaprijed definiranim intervalima ili u određenim uvjetima. Za konfiguraciju Python skripti koje se pokreću unutar "runApp.py" datoteke, potrebno je napraviti nekoliko koraka:

- Otvaranje Task Schedulera: Task Scheduler se može pronaći unutar izbornika Start ili pomoću pretraživanja.

- Kreiranje novog zadatka: U Task Scheduleru, korisnik mora kreirati novi zadatak, gdje će definirati osnovne informacije kao što su naziv zadatka, korisnički račun s kojeg će se zadatak izvršavati, i hoće li se zadatak pokretati čak i ako je korisnik odjavljen.
- Dodavanje okidača (Trigger): Okidač određuje kada će se zadatak pokrenuti. U ovom slučaju, može se postaviti interval izvršavanja (npr. jednom dnevno, svaki sat, ili na određene datume).
- Definiranje akcije: Akcija je stvarni proces koji se izvršava kada okidač bude aktiviran. Ovdje je potrebno definirati pokretanje Python interpretatora i prosljeđivanje puta do "runApp.py" datoteke kao argumenta. Na primjer: python C:\Users\Korisnik\Documents\Fakultet (IT)\Završni rad\test1\runApp.py.
- Praćenje i dijagnostika: Task Scheduler omogućuje pregled stanja zadataka te u slučaju greške, korisniku pruža detaljne informacije o problemu.

Ova metoda omogućava korisnicima na Windows sustavu da automatski pokrenu skup Python skripti u unaprijed određenom vremenskom okviru, osiguravajući da se procesi odvijaju kontinuirano bez ručne intervencije.



Slika 8 Vremensko pokretanje na Windows 11

5.5.2. Vremensko pokretanje na Linux operativnom sustavu

Na Linux serverima, standardni način za vremensko pokretanje zadataka je korištenje „cron“ sustava, koji je iznimno fleksibilan alat za automatizaciju periodičnih zadataka. „Cron“ koristi konfiguracijsku datoteku poznatu kao „crontab“, gdje se definiraju zadaci koji će se izvršavati u specifičnim vremenskim intervalima.

Za postavljanje vremenskog pokretanja skripti na Linux serveru, slijede koraci:

1. Pristup crontab-u: Otvara se crontab konfiguracija pomoću naredbe crontab -e. To će omogućiti uređivanje liste zadataka specifičnih za korisnički račun s kojeg se uređuje crontab.

2. Dodavanje novog zadatka: Zadatak unutar crontaba definira se pomoću specifične sintakse koja omogućuje preciznu kontrolu nad vremenom izvršavanja. Primjer zadatka koji pokreće "runApp.py" svaki sat izgledao bi ovako:

   ```
   0 * * * * /usr/bin/python3 /path/to/runApp.py
   ```

   Ovdje se koristi Python interpretator smješten u /usr/bin/, a zadatak će se pokretati svaki sat. Crontab podržava fleksibilno podešavanje intervala (svake minute, sata, dana u tjednu, itd.).

3. Praćenje i zapisivanje izlaza: Cron zadaci mogu biti konfigurirani tako da pohranjuju izlaz (stdout i stderr) u log datoteke radi lakšeg praćenja i dijagnostike potencijalnih grešaka. Ovo je važno za osiguravanje pravilnog rada aplikacije tijekom vremena.

4. Upotreba virtualnih okruženja: Ako se projekt oslanja na specifično virtualno okruženje (virtualenv), tada crontab mora uključivati aktivaciju tog okruženja prije pokretanja skripte. To se može postići dodavanjem linija unutar crontaba koje prvo aktiviraju virtualno okruženje, a zatim izvršavaju skriptu:

   ```
   0 * * * * /bin/bash -c 'source /path/to/venv/bin/activate && python
   /path/to/runApp.py'
   ```

   Ova metoda osigurava da svi zadaci na Linux serveru budu pokrenuti u preciznim intervalima, čineći cron idealnim alatom za dugoročne i kontinuirane projekte poput web *scrapinga* ili analize podataka.

6. ZAKLJUČAK

U današnjem digitalnom dobu, internet i računalne tehnologije imaju ključnu ulogu u olakšavanju svakodnevnih aktivnosti, poput informiranja i online kupovine. Međutim, s rastućom dostupnošću proizvoda i informacijama iz različitih izvora, korisnici se suočavaju s izazovima pri donošenju informiranih odluka, osobito kada je riječ o pronalasku najpovoljnijih cijena. Web *scraping*, kao tehnika automatskog prikupljanja podataka s web stranica, postaje neizostavan alat u prevladavanju tih izazova. Ovaj projekt usmjeren je na razvoj sustava za usporedbu cijena proizvoda, konkretno televizora, korištenjem *scrapera* implementiranih u Pythonu.

Glavna svrha ovog projekta bila je izgraditi pouzdano i skalabilno rješenje koje korisnicima omogućuje pregledavanje ponuda iz više online trgovina te jednostavnu usporedbu cijena. Kombinacija Python biblioteka poput Requests i BeautifulSoup omogućila je preciznu ekstrakciju podataka iz HTML strukture stranica, dok su prikupljeni podaci pohranjeni u SQLite bazu podataka. Regularni izrazi korišteni su za analizu specifičnih atributa proizvoda, kao što su cijene i dimenzije zaslona. U projekt je uključen razvoj web aplikacije temeljene na okviru Flask, koja korisnicima nudi intuitivno sučelje za pregled prikupljenih podataka.

Vremensko pokretanje sustava osigurava kontinuirano prikupljanje i ažuriranje podataka, a može se postaviti na različitim platformama. Na Windows operativnom sustavu to se postiže putem Task Schedulera, dok se na Linux serveru koristi alat *cron*. Oba alata omogućuju postavljanje periodičnog izvršavanja Python skripti, čime se automatizira proces prikupljanja podataka. Task Scheduler i cron nude fleksibilne opcije za definiranje intervala izvršavanja te omogućuju pregled i dijagnostiku zadataka, što ih čini idealnim rješenjima za ovakve projekte.

Na temelju implementiranih tehnologija i metoda, ovaj projekt predstavlja značajan korak prema optimizaciji procesa online kupovine, omogućujući korisnicima brže i jednostavnije donošenje odluka u digitalnom okruženju. Automatizacija prikupljanja podataka putem web *scrapinga* ima potencijal proširiti se i na druge kategorije proizvoda, čime bi sustav postao još korisniji za širu populaciju korisnika.

LITERATURA

[1] „What Is Scraping | About Price & Web Scraping Tools | Imperva“, Learning Center. Pristupljeno: 08. svibanj 2024. [Na internetu]. Dostupno na: https://www.imperva.com/learn/application-security/web-scraping-attack/

[2] „What is Web Scraping and How to Use It?“, GeeksforGeeks. Pristupljeno: 08. svibanj 2024. [Na internetu]. Dostupno na: https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/

[3] „What is Web Scraping and What is it Used For? | ParseHub“, Web Scraping Blog (Tips, Guides + Tutorials) | ParseHub. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.parsehub.com/blog/what-is-web-scraping/

[4] „What Are The Different Types Of Web Scraping Approaches?“ Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.linkedin.com/pulse/what-different-types-web-scraping-approaches-rajat-thakur

[5] „Scrapfly Web Scraping API | Academy - Scraping Static HTML“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://scrapfly.io/academy/static-scraping

[6] „Scrapfly Web Scraping API | Academy - Scraping Dynamic Pages“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://scrapfly.io/academy/dynamic-scraping

[7] „Dynamic Web Pages Scraping With Python: Guide to Scrape All Content - ZenRows“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.zenrows.com/blog/dynamic-web-pages-scraping-python#how-to

[8] „Scraping Dynamic Websites with Python: Step-by-Step Tutorial“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://oxylabs.io/blog/dynamic-web-scraping-python

[9] „How to parse HTML DOM with DOMDocument ?“, GeeksforGeeks. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.geeksforgeeks.org/how-to-parse-html-dom-with-domdocument/

[10] „DOMParser - Web APIs | MDN“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/API/DOMParser

[11] „API scraping | Academy | Apify Documentation“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://docs.apify.com/academy/api-scraping

[12] „What Is a Headless Browser and Best Ones for Web Scraping - ZenRows“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.zenrows.com/blog/headless-browser-scraping

[13] „How to Scrape Dynamic Websites Using Headless Web Browsers“, ScrapFly Blog. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://scrapfly.io/blog/scraping-using-browsers/

[14] „RSS Feed Scraping or How to Get Even More Content“, DataHen Blog. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.datahen.com/blog/rss-feed-scraping-get-even-more-content/

[15] „Scraping RSS Data Feed and Automated Web Crawling“, PromptCloud. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.promptcloud.com/crawl-scrape-and-extract-rss-feeds-blog-forum-news-content-aggregator/

[16] „OCR Screen Scraping with built-in OCR. Free RPA software for Windows, Mac and Linux.“ Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://ui.vision/rpa/x/desktop-automation/screen-scraping

[17] „Best Proxies for Web Scraping: 2024 Guide“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://oxylabs.io/blog/web-scraping-proxies

[18] „Beautiful Soup Documentation — Beautiful Soup 4.12.0 documentation“. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[19] „lxml - Processing XML and HTML with Python“. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://lxml.de/
[20] „Scrapy 2.11 documentation — Scrapy 2.11.2 documentation“. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://docs.scrapy.org/en/latest/
[21] „The Selenium Browser Automation Project“, Selenium. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://www.selenium.dev/documentation/
[22] „What is Puppeteer? | Puppeteer“. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://pptr.dev/guides/what-is-puppeteer
[23] „Installation | Playwright Python“. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://playwright.dev/python/docs/intro
[24] „XPath | MDN“. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/XPath
[25] „Introduction — feedparser 6.0.11 documentation“. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://feedparser.readthedocs.io/en/latest/introduction.html
[26] „Tesseract User Manual“, tessdoc. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://tesseract-ocr.github.io/tessdoc/
[27] „Web Scraping Tool & Free Web Crawlers“, Octoparse. Pristupljeno: 15. rujan 2024. [Na internetu]. Dostupno na: https://www.octoparse.com/
[28] „What is Python? Executive Summary“, Python.org. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.python.org/doc/essays/blurb/?external_link=true
[29] „JetBrains Products Comparison“, JetBrains. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.jetbrains.com/products/compare/
[30] sdwheeler, „What is PowerShell? - PowerShell“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.4
[31] „Python PIP“. Pristupljeno: 21. rujan 2024. [Na internetu]. Dostupno na: https://www.w3schools.com/python/python_pip.asp
[32] „SQLite - Constraints“. Pristupljeno: 23. rujan 2024. [Na internetu]. Dostupno na: https://www.tutorialspoint.com/sqlite/sqlite_constraints.htm

POPIS SLIKA

Slika 1 Nestrukturirani podaci nakon ekstrakcije 9
Slika 2 Dijagram toka programa 23
Slika 3 Tablica za trgovinu Sancta Domenica 24
Slika 4 Mogućnosti sortiranja 31
Slika 5 Mogućnost filtriranja po proizvođaču 32
Slika 6 Mogućnosti filtriranja i sortiranja 32
Slika 7 Prikaz proizvoda u karticama 33
Slika 8 Vremensko pokretanje na Windows 11 35

GITHUB

https://github.com/jakovsaric/ZavrsniRadV1

DEVELOPMENT OF A PYTHON APPLICATION FOR WEB SCRAPING

SUMMARY

This final thesis deals with the systematized extraction of content from web pages, methods and tools that enable automated data collection from the Internet. The introduction provides an overview of the problem and how these problems are solved. Various types of extraction are explained in detail, including static and dynamic extraction, DOM parsing, API extraction, Headless Browser Scraping, RSS feeds, OCR technology, and the use of proxy servers and IP address rotation. In addition to different types of extraction, the most popular extraction tools such as BeautifulSoup, Scrapy, Selenium, Tesseract and others are briefly described, along with their characteristics and applications. The problems to be solved are described and the goals that the project will achieve are set.

The bulk of the paper will detail the development of a systematized extraction application, from setting up the work environment and database design, to implementing the code and running the scripts at runtime on different operating systems. The paper ends with a conclusion that summarizes the most important points. The paper provides a comprehensive overview of the techniques and tools required for efficient automation of data collection and offers a practical solution through the development of a customized application.

Keywords: extraction, web, scraping, Python, Flask, database, programming tools, runtime scripts, data collection automation