Razvoj sustava za automatizirano prikupljanje i posluživanje tekstualnih podataka

Tipurić, Svan

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Svan Tipurić

Razvoj sustava za automatizirano prikupljanje i posluživanje tekstualnih podataka

Završni rad

Mentor: prof. dr. sc. Ana Meštrović

Rijeka, 24.06.2024.

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija

Rijeka, 3. lipnja 2024.

Zadatak za završni rad

Pristupnik/ica: Svan Tipurić

Naziv završnog rada: Razvoj sustava za automatizirano prikupljanje i pripremu tekstualnih podataka

Naziv završnog rada na engleskom jeziku: Development of a system for automated collection and preparation of textual data

Sadržaj zadatka:

Zadatak završnog rada je razviti sustav za prikupljanje tekstualnih objava s online portala (web scraping) te pripremu tekstova za daljnje korištene. U radu je potrebno prezentirati tehnologije i alata koji su korišteni te opisati na koji način su implementirane faze prikupljanja i pripreme tekstualnih podataka.

Mentor/ica
Prof. dr. sc. Ana Meštrović

Voditelj za završne radove
Izv. prof. dr. sc. Miran Pobar

Zadatak preuzet: 3.6.2024.

(potpis pristupnika/ice)

Radmile Matejčić 2,
51000 Rijeka, Hrvatska

T: +385 (0)51/584-700
E: ured@inf.uniri.hr
www.inf.uniri.hr

OIB: 64218323816
IBAN: HR1524020061400006966

Sažetak

Ovaj završni rad bavi se razvojem sustava za automatizirano prikupljanje i posluživanje tekstualnih podataka s interneta. U radu je korišten programski jezik Python te biblioteke BeautifulSoup, newspaper3k, SQLite i FastAPI za prikupljanje, obradu, pohranu i posluživanje podataka te GitHub Actions za automatizaciju postupka prikupljanja članaka. Sustav je dizajniran da preuzima članke s internetskog portala N1, obrađuje ih, pohranjuje u bazu podataka te omogućava njihovo posluživanje putem API-ja. Glavni ciljevi rada uključuju dizajn i implementaciju učinkovitog i skalabilnog sustava za web struganje, testiranje performansi sustava te analiza prikupljenih podataka. U radu se također raspravlja o pravnim i etičkim aspektima web struganja te predlaže moguća unapređenja sustava.

Ključne riječi: završni rad; web struganje (web scraping); BeautifulSoup; newspaper3k; FastAPI; automatizacija; prikupljanje podataka; GH Actions

SADRŽAJ

1. Uvod

U današnje digitalno doba, količina dostupnih informacija raste eksponencijalno. Na internetu se svakodnevno stvara i objavljuje ogromna količina podataka, uključujući vijesti, blogove, znanstvene radove i druge vrste tekstualnog sadržaja. Prikupljanje i analiza tih podataka može pružiti vrlo važne informacije korisne za razne sektore, uključujući poslovanje, istraživanje, razvoj te medije.

Automatizirano prikupljanje podataka s weba (engl. web scraping) postao je neizostavan alat za istraživače i profesionalce koji koriste velike količine podataka u određene svrhe. Web struganje omogućava automatsko preuzimanje sadržaja s web stranica, što olakšava prikupljanje i obradu podataka u realnom vremenu.

Ovaj završni rad fokusira se na razvoj sustava za automatizirano prikupljanje i posluživanje podataka koristeći programski jezik Python uz napredne alate i tehnologije kao što su BeautifulSoup (bs4), newspaper3k, SQLite, i FastAPI.

Cilj ovog rada bio je dizajnirati i implementirati sustav koji može automatski preuzeti članke s interneta, obraditi tekstualne podatke, pohraniti ih u bazu podataka te omogućiti njihovo posluživanje putem API-ja (Application Program Interface). Sustav je zamišljen da bude skalabilan, fleksibilan i jednostavan za korištenje. Sustav je dio veće cjeline, platforme za analizu trendova u online medijima. Analiza online medija daje bolji uvid u trendove i teme koje su aktualne [1, 2, 3].

U daljnjem tekstu, detaljno je opisana metodologija, alati i tehnologije koje su korištene za razvoj navedenog sustava te u konačnici i rezultati koji su postignuti. Nadalje, temeljito su objašnjeni i svi koraci koji su provedeni u procesu razvoja web scrapera, uključujući prikupljanje podataka, obradu i pohranu te posluživanje podataka korisnicima.

2. Teorijski okvir

2.1. Što je web struganje

Mreža World Wide Web sadrži sve vrste informacija različitog podrijetla te su one dostupne u različitim formatima i putem različitih pristupnih sučelja. Stoga, obrada podataka može biti zamorna, a web struganje je tehnika pomoću koje se taj problem može riješiti [4].

Web struganje je tehnika kojom se automatski prikupljaju podaci s internetskih stranica pomoću softverskih robota koji oponašaju ljudsku interakciju. Pisanjem jednostavnog automatiziranog programa moguće je izdvojiti nama relevantne informacije [5].

Iako je web struganje relativno novi pojam, te se taj proces nazivao drugačije kroz godine postojanja interneta, u ovom se radu u nastavku koristi ovaj naziv procesa. Metode web struganja uključuju preuzimanje HTML (Hyper Text Markup Language) kôda stranice i izdvajanje informacija u strukturiranom [6]. Ova tehnika omogućava korisnicima da prikupljaju velike količine podataka s interneta za daljnju analizu i obradu, primjerice kada web servisi ili API-ji nisu dostupni te u biomedicinskim istraživanjima za prikupljanje i integraciju podataka [6].

Dakle, web struganje uključuje bilo kakvo sakupljanje podataka koje se ne odvija preko API-a. Primjene web struganja su raznolike, od poslovne analitike do znanstvenih istraživanja.

Postoji nekoliko pristupa web struganju, neki od njih su korištenje biblioteka za parsiranje HTML-a te alata za automatizaciju web preglednika. Svaki od pristupa ima svoje prednosti i mane, ovisno o složenosti web stranice i vrsti podataka koje je potrebno prikupiti [7].

2.2. Povijest i razvoj web struganja

Web struganje se razvijalo paralelno s razvojem interneta. Prvi alati za struganje pojavili su se uz prve web preglednike i jednostavne HTML stranice. Kako su web stranice postajale složenije, razvijeni su napredniji alati za struganje koji mogu obraditi dinamički sadržaj i API-je. U početku su alati za web struganje bili ograničeni na statičke HTML stranice, ali danas mogu prikupljati podatke iz kompleksnih i interaktivnih web aplikacija [5].

Prvi alati za web struganje bili su jednostavne skripte napisane u programskim jezicima poput Perla i Pythona [5].

S porastom potreba za struganjem, razvijeni su specijalizirani alati i okviri poput Scrapy i BeautifulSoup, koji omogućavaju učinkovitije i skalabilnije prikupljanje podatak [8].

2.3. Teorija o Pythonu i zašto se koristi za struganje

Python je jedan od najpopularnijih programskih jezika za web struganje zbog svoje jednostavne sintakse i bogatstva biblioteka kao što su BeautifulSoup, Scrapy i Selenium. Ove biblioteke omogućuju jednostavno preuzimanje i parsiranje web stranica, čineći Python idealnim izborom za projekte web struganja [9].

Python također nudi izvrsnu podršku za rad s podacima, uključujući biblioteke kao što su pandas i NumPy, koje omogućuju lako rukovanje i analizu prikupljenih podataka [10].

Jednostavnost korištenja i velika zajednica korisnika čine Python vrlo pristupačnim za početnike, dok njegova snaga i fleksibilnost omogućuju naprednim korisnicima da izgrade kompleksne i skalabilne sustave web struganja. Osim toga, Python se integrira s mnogim alatima za strojno učenje i analizu podataka, što omogućuje daljnju obradu i analizu prikupljenih podataka [11].

2.4. Legalnost i etika web struganja

Iako je web struganje vrlo koristan alat za prikupljanje podataka, potreban je oprez jer postoje značajna etička i pravna pitanja koja se moraju uzeti u obzir koristeći ovu tehniku [12].

Legalnost web struganja varira ovisno o zakonodavstvu pojedinih zemalja i uvjetima korištenja web stranica. U mnogim slučajevima, struganje može kršiti uvjete korištenja stranice, što može rezultirati pravnim posljedicama. Neke web stranice izričito zabranjuju web struganje, pa provođenjem istog dolazi do kršenja ugovora i uvjeta web stranica. Također, ponekad struganje uključuje reprodukciju značajnih dijelova zaštićenog sadržaja bez dopuštenja što dovodi do kršenja autorskih prava [12].

Etika web struganja također je važna, uključujući poštivanje privatnosti korisnika i izbjegavanje preopterećenja servera. Jedan od glavnih etičkih pitanja je potencijalno kršenje privatnosti zato što se podaci prikupljaju i koriste na načine koje korisnici nisu predvidjeli [12].

Važno je napomenuti da mnoge web stranice imaju postavljena pravila u robots.txt datoteci koja definira koji dijelovi stranice smiju biti strugani i time upravljale automatskim pristupom sadržaju te pružale upute web robotima o tome koji dijelovi web stranice smiju, a koji ne smiju biti posjećivani. Iako nije zakonski obavezno slijediti ove upute, isto je preporučljivo te

društvene norme nalažu da se prilikom korištenja tehnike web struganja poštuju pravila postavljena u robot.txt datoteci. Navedeno se smatra etičkim postupanjem i može pomoći u sprječavanju pravnih problema. Ignoriranje ovih uputa može uzrokovati nenamjernu štetu korisnicima i vlasnicima web stranice [12].

Međutim, neki oblici struganja, kao što su prikupljanje osjetljivih podataka ili preopterećenje servera s previše zahtjeva, mogu biti nezakoniti.

Korištenje web struganja u skladu s etičkim i pravnim normama ključno je za održavanje povjerenja i suradnje u digitalnom ekosustavu. To uključuje jasno navođenje izvora podataka, poštivanje autorskih prava i osiguranje da aktivnosti web struganja ne narušavaju funkcionalnost web stranica [12].

3. Alati i tehnologije

3.1. Pregled korištenih alata i biblioteka

3.1.1. BeautifulSoup

BeautifulSoup je biblioteka u Pythonu koja služi za parsiranje HTML i XML dokumenata, omogućavajući jednostavno izvlačenje podataka.

Ova biblioteka pretvara kompleksne HTML dokumente u strukturirane objekte poput stabala, što olakšava navigaciju i pretragu elemenata. BeautifulSoup podržava različite parsere, uključujući Pythonov ugrađeni parser i brži lxml parser. Njegova sposobnost da se nosi s nesavršeno formatiranim HTML-om čini ga izuzetno korisnim za struganje stvarnih web stranica, koje često sadrže nepravilnosti u kodu [13].

Primjer korištenja BeautifulSoup:

```
from bs4 import BeautifulSoup

with open("index.html") as fp:
    soup = BeautifulSoup(fp, 'html.parser')

soup = BeautifulSoup("<html>a web page</html>", 'html.parser')
soup.find_all('a')


# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
# <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
# <a class="sister" href="“http://example.com/tillie“ id="link3">Tillie</a>]
```

3.1.2. newspaper3k

Newspaper3k je specijalizirana Python biblioteka dizajnirana za jednostavno prikupljanje vijesti s web stranica. Omogućuje preuzimanje i analizu članaka, izvlačenje naslova, autora, datuma objave, ključnih riječi, slika i samog teksta članka.

Newspaper3k automatski prepoznaje strukturu vijesti, olakšavajući ekstrakciju podataka. Također uključuje funkcionalnosti za prevođenje članaka, prepoznavanje jezika i detekciju duplikata [14].

Primjer korištenja newspaper3k:

```
from newspaper import Article

url = "http://example.com/news_article"
article = Article(url)
article.download()
article.parse()

print(article.title)
print(article.authors)
print(article.publish_date)
print(article.text)
```

3.1.3. SQLite

SQLite je lagana, ugrađena baza podataka koja ne zahtijeva posebni server za rad. SQLite je popularan zbog svoje jednostavnosti i performansi, posebno za manje projekte i aplikacije.

Podaci se pohranjuju u jednoj datoteci, što olakšava upravljanje i distribuciju baze podataka. SQLite podržava većinu SQL standarda, omogućavajući kompleksne upite, transakcije i integritet podataka [15].

Primjer korištenja SQLite:

```
import sqlite3

conn = sqlite3.connect('example.db')
c = conn.cursor()

c.execute('''CREATE TABLE articles (title TEXT, author TEXT, content TEXT)''')

c.execute("INSERT INTO articles VALUES ('Naslov članka', 'Autor', 'Sadržaj članka')")

c.execute("SELECT * FROM articles")
print(c.fetchall())

conn.commit()
conn.close()
```

3.1.4. FastAPI

FastAPI je moderni web okvir za izradu brzih i performansno učinkovitih API-ja u Pythonu. Omogućava brzu implementaciju RESTful API-ja s minimalnim kodom, pružajući ugrađenu

podršku za validaciju podataka, automatsku dokumentaciju putem Swagger-a i OpenAPI-a, te sigurnosne funkcionalnosti.

FastAPI koristi tipizaciju iz Pythona 3.6+ za provjeru valjanosti podataka, što značajno poboljšava pouzdanost i čitljivost koda [16].

Primjer korištenja okvira FastAPI:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}

@app.get("/articles/{article_id}")
def read_article(article_id: int):
    return {"article_id": article_id}
```

3.1.5. Github Actions

GitHub Actions je dinamičan alat za automatizaciju integriran u platformu GitHub koji omogućuje programerima definiranje prilagođenih tijekova rada za razne zadatke.

Ključna značajka je njegova sposobnost pokretanja zakazanih zadataka koristeći cron sintaksu, što omogućuje automatizaciju redovitih aktivnosti kao što su pokretanje dijelova programskog koda u zakazano vrijeme ili preko noći, rutinsko testiranje, sigurnosne kopije i periodične implementacije.

Ovi zakazani tijekovi rada mogu se postaviti da se izvode u određenim intervalima, kao što su svaki sat, dnevno ili tjedno, osiguravajući da se ključni zadaci održavanja obavljaju bez potrebe za ručnom intervencijom [17].

Korištenjem zakazanih zadataka unutar GitHub Actions timovi mogu održavati bazu koda bez pogreški održavati ažurirane ovisnosti i osigurati neprekinuti tijek implementacije, čime se povećava ukupna produktivnost i pouzdanost.

Ova automatizacija smanjuje ručno radno opterećenje i osigurava da se ključni procesi provode pouzdano i na vrijeme [17].

Primjer korištenja alata GH Actions u projektu:

```
name: Daily Scraper
•
```

```yaml
on:
  schedule:
    - cron: "0 0 * * *" # This runs the job daily at midnight
  workflow_dispatch: # Allows manual trigger

jobs:
  run_daily_scraper:
    runs-on: self-hosted
    timeout-minutes: 14400

    steps:
      # - name: Checkout repository
      #   uses: actions/checkout@v2

      # - name: Set up Python
      #   uses: actions/setup-python@v2
      #   with:
      #     python-version: "3.11.9"

      - name: Install dependencies
        run: |
          source /home/milky/Envs/n1_scraper/bin/activate
          pip install -r /home/milky/N1Scraper/requirements.txt

      - name: Create data directories
        run: |
          mkdir -p /home/milky/N1Scraper/data_temp

      - name: Run scraper script
        run: python /home/milky/N1Scraper/scraper/n1_scraper.py

      - name: Echo job status
        run: echo "Daily scraper job status is ${{ job.status }}."

      - name: Push to GH
        run: |
          cd /home/milky/N1Scraper/
          git config url."https://oauth2:${{ secrets.GH_TOKEN
}}@github.com".insteadOf https://github.com
          git pull origin main
          git add data/articles.db data/last_scraped_datetime.txt
data/duplicates.json
          git commit -m "data_update"
          git push origin main
```

3.2. Usporedba s drugim alatima

3.2.1. Scrapy

Scrapy je moćan i fleksibilan okvir za web struganje u Pythonu.

Dizajniran je za brzu i efikasnu izradu scraping projekata. Scrapy omogućava ekstrakciju podataka iz web stranica, navigaciju kroz web stranice i pohranu prikupljenih podataka.

Pogodan je za složene projekte gdje je potrebno prikupljati podatke s velikog broja stranica ili kada je potrebna interakcija s dinamičkim sadržajem [18].

3.2.2. Selenium

Selenium je alat za automatizaciju web preglednika koji se često koristi za web struganjee dinamičkih stranica koje zahtijevaju interakciju s JavaScriptom.

Selenium omogućava simulaciju korisničkih akcija kao što su klikanje, popunjavanje obrazaca i navigacija kroz web stranice.

Idealno je rješenje za struganje stranica koje se dinamički učitavaju ili za testiranje web aplikacija [19].

3.2.3. Razlozi za odabir korištenih alata

Scrapy i Selenium su moćni alati s mnogim prednostima, no nisu korišteni u ovom radu iz nekoliko razloga:

1. Jednostavnost i specifičnost zadatka

   BeautifulSoup i newspaper3k su jednostavniji za korištenje i pogodniji su za prikupljanje statičnih stranica kao što su vijesti, što je u skladu s ciljevima ovog rada. Njihova jednostavnost i fokus na specifične zadatke čine ih prikladnijim za brzi razvoj i implementaciju.

2. Performanse i resursi

   Scrapy je vrlo moćan, ali za jednostavnije zadatke može biti previše kompleksan. BeautifulSoup i newspaper3k su lakši i zahtijevaju manje resursa, što je važno za održavanje učinkovitosti sustava.

3. Statički sadržaj

   Iako Selenium može upravljati dinamičkim sadržajem, njegov rad je sporiji zbog potrebe za simulacijom cijelog preglednika. U ovom je radu fokus bio na statičnom sadržaju vijesti gdje BeautifulSoup i newspaper3k pružaju dovoljno funkcionalnosti.

4. Integracija s postojećim alatima

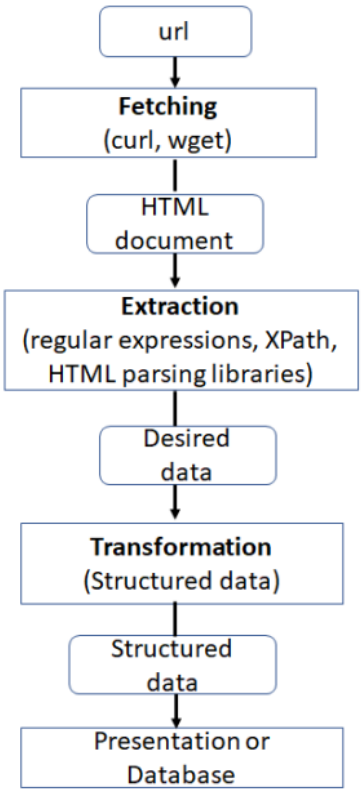
   Alati korišteni u ovom radu (BeautifulSoup, newspaper3k, SQLite, FastAPI) dobro se integriraju međusobno, omogućujući jednostavan tijek rada od prikupljanja do posluživanja podataka.

Ovim se pristupom postigla jednostavnost, učinkovitost i specifičnost potrebna za prikupljanje i posluživanje vijesti objavljenih na webu.

4. Metodologija

Proces prikupljanja vijesti s web stranica sastoji se od nekoliko ključnih koraka. Ovaj proces uključuje automatsko dohvaćanje, ekstrakciju i transformaciju podataka s web stranica u organizirani format za daljnju analizu ili arhiviranje. Detaljno su opisane glavne faze web struganja: dohvaćanje, ekstrakcija i transformacija podataka te posluživanje prikupljenih podataka putem API-ja. Svaka od ovih faza ima svoje specifičnosti i tehnike koje su objašnjene i ilustrirane kroz primjere koda.

Na slici 1 prikazana su tri koraka u procesu struganja web stranica za prikupljanje vijesti.



Slika 1: proces web struganja

4.1. Faza dohvaćanja

Faza dohvaćanja je prvi korak u procesu prikupljanja vijesti s web stranica. Ključna je jer postavlja temelje za daljnje faze koje slijede.

U fazi dohvaćanja, prvo se mora posjetiti željena web stranica koja sadrži potrebne informacije.

To se radi pomoću osnovnog alata za komunikaciju između klijenta i web poslužitelja, odnosno internetskog protokola nazvanog HTTP (Hypertext Transfer Protocol). HTTP je standardni

protokol koji se koristi za slanje i primanje zahtjeva s web poslužitelja. Web preglednici koriste slične tehnike za pristup sadržaju stranice. Kada koristimo web preglednik, šalje se HTTP GET zahtjev i prima se HTML odgovor od poslužitelja. HTTP GET zahtjev je jedan od najčešće korištenih HTTP metoda koji se koristi za dohvaćanje podataka s poslužitelja [20].

Svaki HTTP zahtjev sadrži nekoliko ključnih elemenata [20]:

- HTTP verzija: Verzija HTTP protokola koja se koristi (npr. HTTP/1.1, HTTP/2).
- URL: Uniform Resource Locator, jedinstveni identifikator resursa na internetu.
- HTTP metoda: Definira akciju koju klijent želi da poslužitelj izvrši (npr. GET, POST, PUT, DELETE).
- HTTP zaglavlja: Metapodaci vezani za zahtjev, kao što su tip sadržaja koji klijent prihvaća, informacije o autentifikaciji itd.
- Opcionalno tijelo zahtjeva: Koristi se u nekim metodama poput POST ili PUT za slanje podataka poslužitelju.
- Primanje HTML Odgovora

Kako bi dobili HTML stranicu kao odgovor, izvodimo HTTP GET zahtjev na ciljnu adresu, tj. URL (Uniform Resource Locator). Poslužitelj tada vraća HTTP odgovor koji sadrži [20]:

- HTTP statusni kod: Informacija o rezultatu zahtjeva (npr. 200 OK, 404 Not Found).
- HTTP zaglavlja odgovora: Metapodaci vezani za odgovor, poput tipa sadržaja koji se vraća.
- Opcionalno tijelo odgovora: Sadržaj koji je zatražen, poput HTML stranice.

Ovaj proces je ključan jer omogućava dohvaćanje potrebnih informacija s web stranica koje će se kasnije koristiti u fazama ekstrakcije i transformacije podataka [20]. HTTP protokol pruža strukturu i standarde za učinkovitu komunikaciju između klijenata i poslužitelja, čime se osigurava točnost i pouzdanost prenesenih podataka. [21].

Primjer koda za slanje HTTP zahtjeva i dobivanje odgovora:

```
import requests

def fetch_article_data(url):
    response = requests.get(url)
    return response.text
```

4.2. Faza ekstrakcije i transformacije

Ključne se informacije moraju izvaditi iz HTML stranice nakon što se dohvate.

Faza ekstrakcije je dio kada se koriste regularni izrazi, biblioteke za raščlanjivanje HTML-a ili XPath upiti [21].

Regularni izraz, poznat i kao racionalni izraz, je niz znakova koji označavaju uzorak podudaranja u tekstu. Može se skratiti kao regex ili regexp. Tipično, algoritmi za pretraživanje nizova koriste ove obrasce za provjeru valjanosti unosa ili za radnje "pronađi" ili "pronađi i zamijeni" nizove [22].

Unutar sustava korišteni su regularni izrazi kako bi se izdvojile relevantne informacije i maknuo bespotreban tekst. Omogućujući spremanje isključivo informacija koje služe našoj svrsi u potrebnom formatu za spremanje u bazu podatka

Primjer koda za ekstrakciju podataka:

```
import requests
from newspaper import Article
from bs4 import BeautifulSoup

def get_text_from_article(article_source):
    try:
        # Using article object to parse the page content
        article = Article(article_source, language="hr")
        article.download()
        article.parse()
        text = article.text

        # Processing the text so that it leaves only raw data
        modified_text = text.replace(
  'N1 pratite putem aplikacija za Android | iPhone/iPad i mreža       Twitter | Facebook |
Instagram | TikTok.', '')
        modified_text = modified_text.replace('Podijeli :', '')
        modified_text = modified_text.replace('\n', ' ')
        pattern = r'\b\S+\s?\S*\/\S+\b'
        modified_text = re.sub(pattern, '', modified_text)
        pattern = r'\b[\w\s]+\s?\/\s?\w+\b'
        modified_text = re.sub(pattern, '', modified_text)
        pattern = r'\b.+?\s?\/\s?.+?\b'
        modified_text = re.sub(pattern, '', modified_text)
        pattern = r'\b\w+\s+via\s+REUTERS\b'
        modified_text = re.sub(pattern, '', modified_text)
        modified_text = modified_text.replace('Pexels', '')
        modified_text = modified_text.replace('N1', '')
        modified_text = modified_text.replace('via REUTERS', '')
        modified_text = modified_text.replace('/', '')
        modified_text = modified_text.lstrip()
        return modified_text
    except Exception as e:
        logger.error(f"Error occurred while parsing article: {e}")
        return ""
```

```
def get_tags_from_article(article_source):
    response = requests.get(article_source, headers={
                            'Accept-Charset': 'UTF-8'})
    content = response.content.decode('utf-8')
    soup = BeautifulSoup(content, 'html.parser')
    tags_elements = soup.find_all(rel="tag")
    tags = [element.get_text() for element in tags_elements]
    return tags
```

4.3. Spremanje podataka

Sada kada su ostali samo relevantni podaci, oni se mogu spremiti u organizirani format za prikaz.

Iz pohranjenih podataka mogu se dobiti relevantne informacije i dalje ih obrađivati [21].

Podaci se mogu transformirati i pohraniti u SQLite bazu podataka za jednostavan pristup i manipulaciju. SQLite podržava različite SQL operacije koje olakšavaju rad s podacima a neke od njih korištene unutar primjera su:

- CREATE TABLE: Ova naredba se koristi za kreiranje nove tablice unutar SQLite baze podataka. Tablica se kreira samo ako već ne postoji, čime se izbjegavaju greške prilikom ponovnog pokretanja koda.

- INSERT INTO: Ova naredba omogućava umetanje novih zapisa u tablicu. U primjeru, koristi se pripremljena izjava sa zamjenskim znakovima (?) kako bi se omogućilo umetanje više zapisa odjednom koristeći executemany.

- Nošenje sa error-ima: SQLite podržava hvatanje i rukovanje greškama kako bi se osiguralo da se baza podataka ne korumpira i da se greške pravilno evidentiraju.

Primjer koda za transformaciju podataka:

```
import sqlite3

# Spremanje u datotečni sustav
directory = os.path.join(data_dir, '../data_temp',article.date)
create_directory_if_not_exists(directory)
file_name = f"{article.article_id}.json"
file_path = os.path.join(directory, file_name)
with open(file_path, "w", encoding="utf-8") as file:
json.dump(article.to_dict(), file, ensure_ascii=False, indent=4)

# Spremanje u bazu podataka (SQLite)
connection = sqlite3.connect(os.path.join(data_dir, 'articles.db'))
cursor = connection.cursor()
try:
sqlite_insert_query = """INSERT INTO articles (article_id, title, date, time, hashtags,
text, source, category) VALUES (?, ?, ?, ?, ?, ?, ?, ?)"""
```

```
records_to_insert = [(article.article_id, article.title, article.date,
article.time,json.dumps(article.hashtags), article.text, article.source, article.category)
for article in article_list]

      cursor.executemany(sqlite_insert_query, records_to_insert)
      connection.commit()
except sqlite3.Error as error:
logger.error(f"Failed to insert records into articles table:
{error}")
finally:
      cursor.close()
      connection.close()
```

Tablica 1: Prikaz atributa tablice baze podataka

Article
- **article_id**
- **title**
- **date**
- **time**
- **hashtags**
- **text**
- **source**
- **category**

4.4. Posluživanje podataka putem API-ja

Nakon što su podaci prikupljeni i pohranjeni, potrebno ih je učiniti dostupnima putem API-ja. FastAPI omogućava brzu i jednostavnu izradu RESTful API-ja, pružajući korisnicima mogućnost pristupa podacima na strukturiran način.

API omogućava pretragu i filtriranje podataka prema različitim kriterijima kao što su naslov, datum i kategorija članka.

Koraci za posluživanje podataka unutar rada uključuju:

1. Definiranje API endpointa

   Definiranje API endpointa obuhvaća kreiranje ruta koje omogućavaju pristup podacima. Endpointi su točke pristupa kroz koje korisnici mogu dohvatiti ili poslati podatke. U kontekstu članka, ovo bi moglo uključivati rute za dohvaćanje svih članaka, dohvaćanje pojedinačnih članaka na temelju njihovog ID-a ili pretragu članaka na temelju određenih kriterija kao što su naslov, datum i kategorija. Svaki

endpoint mora biti jasno definiran kako bi korisnici znali koje podatke mogu očekivati i koje parametre mogu koristiti za filtriranje rezultata.

2. Implementacija funkcionalnosti za pretragu

   Implementacija funkcionalnosti za pretragu omogućuje korisnicima pretraživanje i filtriranje podataka. Ova funkcionalnost omogućava korisnicima da unesu specifične kriterije pretrage kako bi dobili samo one podatke koji su im relevantni. Na primjer, korisnici mogu pretraživati članke prema naslovu, što im omogućava da brzo pronađu sve članke koji sadrže određene ključne riječi. Također mogu filtrirati članke prema datumu objave ili prema kategoriji kako bi dobili ciljanije rezultate. Ova fleksibilnost pretrage i filtriranja čini API vrlo korisnim za krajnje korisnike.

3. Osiguranje API-ja

   Osiguranje API-ja sastoji se od implementacije sigurnosnih mjera kako bi se osigurao pristup samo ovlaštenim korisnicima. Ovo uključuje provjeru autentičnosti i autorizacije korisnika. Autentifikacija osigurava da su korisnici koji pristupaju API-ju oni za koje tvrde da jesu, dok autorizacija osigurava da ti korisnici imaju prava za pristup ili izmjenu podataka. FastAPI podržava različite metode autentifikacije i autorizacije, uključujući OAuth2, JWT (JSON Web Tokens), API ključeve i osnovnu HTTP autentifikaciju. Implementacija ovih sigurnosnih mjera ključna je za zaštitu osjetljivih podataka i sprječavanje neovlaštenog pristupa.

Kombinacija definiranja jasnih API endpointa, implementacije funkcionalnosti za pretragu i osiguravanja API-ja osigurava da korisnici mogu jednostavno i sigurno pristupiti podacima. Korištenje FastAPI-ja omogućava brzu izradu učinkovitih i sigurnih API-ja koji mogu zadovoljiti različite potrebe korisnika, od pretrage podataka do zaštite privatnosti i sigurnosti. Ovaj proces omogućava da se iskoriste podaci na najefikasniji način, pružajući korisnicima točne i relevantne informacije kad god su im potrebne.

Primjer koda za kreiranje API-ja:

```
from fastapi import FastAPI, HTTPException
import sqlite3

app = FastAPI()

@app.get("/articles/{article_id}")
def read_article(article_id: int):
    conn = sqlite3.connect('articles.db')
    c = conn.cursor()

    c.execute("SELECT * FROM articles WHERE id = ?", (article_id,))
    article = c.fetchone()
    conn.close()
```

```
    if article is None:
        raise HTTPException(status_code=404, detail="Article not found")

    return {
        'id': article.id,
        'title': article.title,
        'publish_date': article.date,
        'text': article.text
    }
```

5. Sustav za automatizirano prikupljanje, spremanje i posluživanje podataka

Ovim radom razvijen je sustav koji automatizirano pomoću okvira GitHub Actions prikuplja nama relevantne informacije svih članaka N1 portala, zatim ih sprema u bazu podataka i servira pomoću API-a.

Cilj je bio stvoriti učinkovito i skalabilno rješenje koje može kontinuirano pratiti, prikupljati i obrađivati podatke s web stranica, čime se osigurava ažuriranost i dostupnost informacija.

5.1. Arhitektura sustava

Arhitektura sustava uključuje više komponenti koje koherentno rade. Zajedno sve ove komponente čine jedan skalabilan i robustan sustav.

Osnovne komponente sustava su:

1. Scraper

   Modul za prikupljanje podataka s web stranica koristeći BeautifulSoup i newspaper3k. Ovaj modul je odgovoran za preuzimanje i parsiranje HTML-a te ekstrakciju relevantnih informacija. Također provjerava informacije i izbacuje nepotrebne znakove. Sve obrađene podatke zatim sprema u bazu podataka brinući se o jedinstvenim identifikatorima. Također, koristi python biblioteku tdqm koja omogućuje kreiranje trake napretka pomoću koje možemo vidjeti na koliko je posto strugatelj tokom izvršavanja i koliko mu treba da završi svoje izvođenje.

2. Baza podataka

   SQLite baza podataka u koju se pohranjuje prikupljene i obrađene podatke. Baza podataka omogućava jednostavan pristup i manipulaciju podacima.

3. API

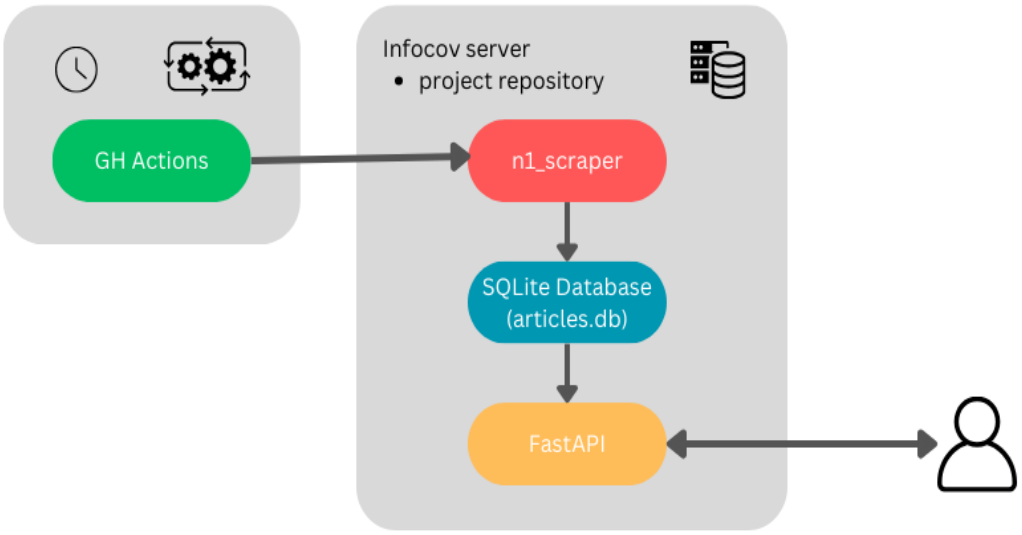
   FastAPI aplikacija koja omogućuje pristup podacima putem RESTful API-ja. API omogućava pretragu i filtriranje podataka te njihovo posluživanje korisnicima. Koristi kako bi se podaci mogli dijeliti do drugih sustava.

4. Logger

Logger zapisuje stvari koje se događaju u programu kako bi u slučaju da sustav „pukne“ znali u čemu je bio problem te dobili više informacija i kako taj problem rješiti.

5. GitHub actions skripta

   Pomoću GH actions skripte omogućavamo automatizaciju prikupljanja podataka. Unutar CRON sintakse definirano je vrijeme izvršavanja skripte te prije same skripte postavljanje okruženja i instaliranje zahtjeva.



Slika 2: model arhitkture razvijenog sustava

5.2. Testiranje i validacija sustava

Testiranje se provodilo kako bi se osiguralo da sustav pravilno funkcionira pod različitim uvjetima i opterećenjima.

Testiranje je uključivalo:

1. Funkcionalno testiranje

   Provjera ispravnosti funkcionalnosti sustava, uključujući prikupljanje podataka, obradu i pohranu te pristup putem API-ja.

2. Performansno testiranje

Evaluacija performansi sustava pod različitim opterećenjima kako bi se osigurala skalabilnost i učinkovitost.

3. Validacija podataka

Provjera točnosti prikupljenih podataka, uključujući usporedbu s izvornim izvorima podataka i provjeru cjelovitosti baze podataka.

6. Rezultati

1.1. Što je N1 portal i kako interpretirati rezultate struganja

Fokus ovog sustava bio je na prikupljanju podataka s portala N1 (https://n1info.hr/), s kojeg su prikupljeni različiti podaci uključujući naslove, autore, datume objave i sadržaj članaka.

N1 je informativna multiplatforma na kojoj osobe mogu pratiti vijesti onda kada se događaju. Osim programa, na kojemu se reporteri N1 javljaju uživo, postoji i portal kojemu se može pristupiti preko mobilnog uređaja i računala. Ovaj portal omogućuje da osobe uživo prate najnovije vijesti iz Hrvatske, Srbije, Bosne i Hercegovine i Slovenije. Vijesti koje objavljuju su raznolike, od vijesti iz države, međunarodnih vijesti, ekonomije, politike, sporta sve do kategorija zdravlje, crna kronika, a sadrži i razne intervjue [23].

Kategorije koje postoje na platformi N1 uključuju [23]:

- vijesti (iz svijeta i regije)
- sport
- vrijeme
- auto
- magazin (uključuje znanost, tehnologiju, zdravlje, kuhanje, lifestyle, kulturu, showbiz i ljubimce)
- crnu kroniku
- kolumne

Na portalu N1 objavljuje se velik broj članaka dnevno, između 50 i 100 članaka.

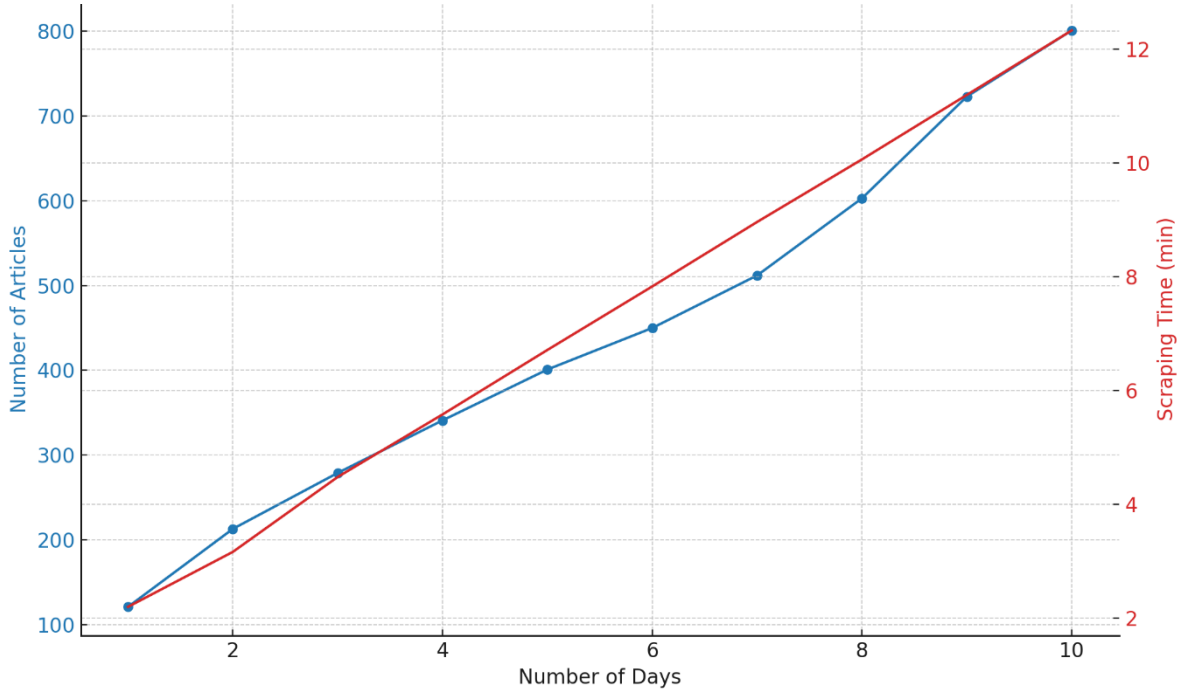
6.1. Učinkovitost sustava

Učinkovitost sustava evaluirana je na temelju nekoliko ključnih kriterija, uključujući brzinu prikupljanja podataka, točnost ekstrakcije informacija i pouzdanost sustava. Pouzdanost sustava mjerena je testiranjem rada sustava u različitim vremenskim dobima i radom s različitom količinom informacija. Sustav je pokazao visoku razinu učinkovitosti u prikupljanju i obradi podataka s portala N1, omogućujući brzo i precizno preuzimanje velikih količina članaka.

Brzina prikupljanja podataka mjerena je vremenom potrebnim za preuzimanje i obradu stranica s portala N1. Sustav je bio sposoban preuzeti i obraditi stotine članaka u kratkom vremenskom periodu, što demonstrira njegovu sposobnost za rad s velikim količinama podataka.

Tablica 2: brzina prikupljanja članaka sustava

Period struganja	*Broj članaka*	*Brzina struganja*
1 dan (28.06)	121	2:12 min
2 dana (26.06 – 28.06)	213	3:10 min
3 dana (25.06 – 28.06)	279	4:29 min
4 dana (24.06 – 28-06)	341	5:35 min
5 dana (23.06 - 28.06)	401	6:43 min
6 dana (22.06 - 28.06)	450	7:50 min
7 dana (21.06 - 28.06)	512	8:58 min
8 dana (20.06 - 28.06)	603	10:04 min
9 dana (19.06 - 28.06)	723	11:12 min
10 dana (18.06 - 28.06)	801	12:20 min



Slika 3: Grafički prikaz usporedbi brzine struganja

Također, točnost ekstrakcije informacija procijenjena je usporedbom prikupljenih podataka s originalnim člancima na portalu N1, pokazavši visoku razinu preciznosti sa čak 100%. Svaki

članak unutar vremenskog razdoblja struganja uspješno je prikupljen te ima točan sadržaj i formu.

6.2. Testiranje rada sustava

Kako bi se demonstrirala praktična primjena sustava, provedeno je opsežno testiranje koje je uključivalo prikupljanje i analizu vijesti s portala N1. Cilj testiranja bio je pokazati kako sustav može učinkovito prikupljati, obrađivati i posluživati podatke u stvarnom vremenu, osiguravajući točnost i ažuriranost prikupljenih informacija.

U sklopu testiranja, prikupljeni su članci s portala N1 tijekom perioda od jednog mjeseca (28.05. – 28.06.). Tijekom tog razdoblja sustav je automatski prikupljao sve objavljene članke, izvukao relevantne informacije te ih pohranjivao u bazu podataka. Nakon prikupljanja podataka, provedena je detaljna analiza kako bi se utvrdili ključni trendovi i obrasci u vijestima.

Analiza prikupljenih podataka uključuje pregled i interpretaciju informacija prikupljenih putem sustava za struganje.

Analiza učestalosti objavljivanja pokazala je da portal N1 ima konzistentan ritam objavljivanja vijesti s većim brojem članaka objavljenih tijekom radnih dana u odnosu na vikende. Dnevni prosjek objavljenih članaka bio je oko 50, s naglašenim vrhuncima tijekom ključnih događaja ili važnih vijesti.

Sustav je uspješno kategorizirao prikupljene članke prema različitim tematskim područjima kao što su politika, gospodarstvo, sport, tehnologija i zabava. Najveći broj članaka bio je iz kategorije politike, što je odražavalo aktualne političke događaje i rasprave. Kategorije gospodarstva i sporta također su bile značajno zastupljene.

Analiza sadržaja vijesti otkrila je nekoliko dominantnih tema koje su se ponavljale tijekom promatranog razdoblja. Primjerice, politička situacija u zemlji, ekonomski pokazatelji i sportski događaji često su bili u fokusu. Sustav je također omogućio identificiranje ključnih riječi i fraza koje su se često pojavljivale, pružajući dodatne uvide u glavne teme i interese čitatelja.

1.2. Prednosti i nedostaci razvijenog sustava

Razvijeni sustav za automatizirano prikupljanje i posluživanje tekstualnih podataka s portala N1 ima nekoliko značajnih prednosti.

Prvo, sustav je visoko učinkovit u prikupljanju i obradi velikih količina podataka, omogućujući brzo i precizno preuzimanje informacija.

Drugo, sustav je fleksibilan i može se prilagoditi različitim potrebama korisnika, omogućujući prikupljanje podataka s različitih izvora i njihovo posluživanje putem API-ja.

Prednosti sustava uključuju:

1. Učinkovitost - sustav može brzo prikupljati i obrađivati velike količine podataka, što je ključno za analizu vijesti u stvarnom vremenu.
2. Preciznost - sustav pokazuje visoku točnost u ekstrakciji podataka, što je potvrđeno usporedbom prikupljenih informacija s originalnim člancima.
3. Fleksibilnost - sustav je dizajniran tako da može prilagoditi različite izvore podataka i omogućiti korisnicima jednostavan pristup putem API-ja.

Međutim, sustav ima i određene nedostatke. Jedan od glavnih nedostataka je promjena strukture podataka na portalu N1, nažalost ne možemo utjecati na takvo što nego se sustav konstantno mora održavati i unaprjeđivati.

Isto tako sustav koristi samo jedan portal kao izvor podataka što može rezultirati u pristranosti portala ka određenoj političkoj stranci te narušavanju rezultata predviđanja.

7. Zaključak

Ovaj rad opisuje razvoj sustava za automatizirano prikupljanje i posluživanje tekstualnih podataka, s fokusom na vijesti s portala N1. Sustav koristi alate kao što su BeautifulSoup, newspaper3k, SQLite i FastAPI za prikupljanje, obradu, pohranu i posluživanje podataka. Testiranje i evaluacija sustava pokazali su visoku učinkovitost i preciznost, omogućujući brzo i točno prikupljanje velikih količina podataka.

Razvijeni sustav pruža učinkovito rješenje za prikupljanje i analizu vijesti, omogućujući korisnicima da brzo i jednostavno pristupe relevantnim informacijama. Sustav je fleksibilan te lako prilagodljiv, pružajući visoku razinu performansi i pouzdanosti. Iako postoje određeni izazovi, uključujući promjenu strukture stranice i pravne aspekte web struganja, sustav predstavlja značajan napredak u automatizaciji prikupljanja podataka.

Ovaj je rad dio šireg projekta u kojem sudjeluju i drugi članovi tima te su zadaci podijeljeni kako bi se postigao sveobuhvatan cilj. Svrha i zadatak ovog rada bio je prikupljanje, obrada i posluživanje podataka. Ostali su članovi tima odgovorni za modele strojnog učenja koji analiziraju podatke kako bi identificirali političare i kontekst u kojem se spominju, te za vizualizaciju rezultata na web stranici.

Razvoj sustava može se dalje unaprijediti na nekoliko načina. Dodatne nadogradnje mogu uključivati integraciju s društvenim mrežama, poboljšanje performansi i implementaciju sigurnosnih mjera. Integracija s društvenim mrežama, odnosno proširenje sustava za prikupljanje podataka s društvenih mreža može pružiti širi uvid u popularne teme i trendove. Nadalje, poboljšanje performansi, odnosno njihova optimizacija može osigurati veću pouzdanost i skalabilnost, što omogućuje sustavu da obradi još veće količine podataka brže. I na kraju, implementacija dodatnih sigurnosnih mjera može zaštititi prikupljene podatke i osigurati da samo ovlašteni korisnici imaju pristup sustavu i njegovom API-ju.

Proširenje sustava za prikupljanje podataka s društvenih mreža i drugih izvora može povećati opseg i korisnost prikupljenih informacija. Implementacija dodatnih sigurnosnih mjera i optimizacija performansi može osigurati veću pouzdanost i skalabilnost sustava.

Budući rad može uključivati istraživanje novih izvora podataka za prikupljanje i nadogradnju sustava da radi na drugim strukturama stranica.

Literatura

[1] K. Babić, M. Petrović, S. Beliga, S. Martinčić-Ipšić, M. Matešić, I. Petrijevčanin Vuksanović i A. Meštrović, "Characterisation of COVID-19-related tweets in the Croatian language: framework based on the Cro-CoV-cseBERT model. ," *Applied Sciences*, svez. 11, br. 21, 2021.

[2] S. Beliga, S. Martinčić-Ipšić, M. Matešić, I. Petrijevčanin Vuksanović i A. Meštrović, "Infoveillance of the Croatian online media during the COVID-19 pandemic: one-year longitudinal study using natural language processing," *JMIR public health and surveillance*, svez. 7, br. 12, 2021.

[3] V. Orešković, A. Meštrović i S. Beliga, "Towards Computational Content Analysis of Crises-Related News in Electronic Media," *Central European Conference on Information and Intellingent Systems*, pp. 407-416, 2023.

[4] S. C. M. d. S. Sirisuriya, "A Comparative Study on Web Scraping," u *Proceedings of the 8th International Research Conference, KDU, Ratmalana, Sri Lanka*, Ratmalana, Sri Lanka, 2015.

[5] M. A. Kdher, "Web Scraping or Web Crawling: State of Art,," *International Journal of Advances in Soft Computing & Its Applications*, svez. 13, 2021.

[6] D. Glez-Peña, . A. Lourenço, H. López-Fernández, M. Reboiro-Jato i F. Fdez-Riverola, "Web scraping technologies in an API world," *Briefings in Bioinformatics*, svez. 15, br. 5, pp. 788-797, 2014.

[7] R. Mitchell, Web Scraping with Python, 2nd ur., O'Reilly Media, Inc., 2018.

[8] Scraper.do., "Explore the History of Web Scraping," n.d.. [Mrežno]. Dostupno: https://scrape.do/blog/explore-the-history-of-web-scraping/. [Pokušaj pristupa 22. Lipanj 2024].

[9] ZenRows, "Best Language for Web Scraping," n.d.. [Mrežno]. Dostupno: https://www.zenrows.com/blog/best-language-web-scraping#top-languages-web-scraping. [Pokušaj pristupa 22. Lipanj 2024].

[10] M. Ahamad, "Exploratory Data Analysis with NumPy, pandas, Matplotlib, and Seaborn," 2. Travanj 2018. [Mrežno]. Dostupno: https://www.freecodecamp.org/news/exploratory-

data-analysis-with-numpy-pandas-matplotlib-seaborn/. [Pokušaj pristupa 22. Lipanj 2024].

[11] Netguru, "Python Pros and Cons," 29 October 2020. [Mrežno]. Dostupno: https://www.netguru.com/blog/python-pros-and-cons. [Pokušaj pristupa 22. Lipanj 2024].

[12] V. Krotov, L. Johnson i L. Silva, "Tutorial: Legality and Ethics of Web Scraping," *Communications of the Association for Information Systems,* svez. 47, pp. 555-581, 2020.

[13] L. Richardson, "Beautiful Soup Documentation," n.d.. [Mrežno]. Dostupno: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Pokušaj pristupa 22. Lipanj 2024].

[14] A. Barde i J. Tops, "Newspaper3k: Article scraping & curation," n.d. n.d. n.d.. [Mrežno]. Dostupno: https://newspaper.readthedocs.io/en/latest/. [Pokušaj pristupa 22. Lipanj 2024].

[15] D. R. Hipp, "SQLite," n.d.. [Mrežno]. Dostupno: https://www.sqlite.org/. [Pokušaj pristupa 22. Lipanj 2024].

[16] S. Tiangolo, "FastAPI," n.d.. [Mrežno]. Dostupno: https://fastapi.tiangolo.com/. [Pokušaj pristupa 22. Lipanj 2024].

[17] GitHub, "GitHub Actions," n.d.. [Mrežno]. Dostupno: https://docs.github.com/en/actions. [Pokušaj pristupa 22. Lipanj 2024].

[18] Scrapy, "Scrapy Documentation," n.d.. [Mrežno]. Dostupno: https://docs.scrapy.org/en/latest/index.html. [Pokušaj pristupa 22. Lipanj 2024].

[19] Selenium, "Selenium Documentation," n.d.. [Mrežno]. Dostupno: https://www.selenium.dev/documentation/. [Pokušaj pristupa 22. Lipanj 2024].

[20] Infobip, "What is HTTP (Hypertext Transfer Protocol)?," *Infobip,* 2022.

[21] E. Persson, "Evaluating Tools and Techniques for Web Scraping," 2019.

[22] A. V. Aho, "Algorithms for finding patterns in strings," *Handbook of Theoretical Computer Science,* svez. A: Algorithms and Complexity, p. 255–300, 1990.

[23] N1info.hr, "N1: Najnovije, točne i nezavisne vijesti," 2024. [Mrežno]. Dostupno: https://n1info.hr/. [Pokušaj pristupa 6. Srpanj 2024].

Popis tablica

Popis slika

Popis priloga

Prilog 1: Github repozitorij projekta