

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2499

**Informacijski sustav za
prikupljanje javno dostupnih
podataka za zadani skup osoba**

Tomislav Kravaršćan


Zagreb, lipanj 2021.

SADRŽAJ

1. Uvod	1
2. Opis problema	2
3. Arhitektura sustava	3
3.1. <i>Web scraping</i> i <i>web crawling</i>	4
3.1.1. <i>Robots.txt</i> datoteka	5
3.1.2. Izvlačenje poveznica na članke s popisa vijesti	5
3.1.3. Izvlačenje informacija sa stranice članka	7
3.2. <i>Named Entity Recognition API</i>	10
3.2.1. Izvlačenje imena pomoću CLASSLA cjevovoda	10
3.2.2. Poboljšavanje rezultata NER-a pomoću baze hrvatskih imena i prezimena	12
3.3. Spremanje članaka s pronađenim imenovanim entitetima u bazu	16
3.4. Periodičko izvršavanje procesa <i>scrapinga</i>	18
3.4.1. <i>NodeJS cron job</i> za periodičko pokretanje <i>scrapera</i>	19
3.5. Web prikaz i filtriranje članaka	20
3.5.1. Web poslužitelj	22
4. Korištene tehnologije	25
4.1. <i>JavaScript</i>	25
4.2. <i>NodeJS</i>	25
4.3. <i>Puppeteer</i>	26
4.4. <i>MongoDB</i>	27
4.5. <i>Python</i>	27
4.6. <i>Flask</i>	28
4.7. <i>Vue.js</i>	29

5. Ocjena sustava	32
5.1. Ocjena <i>scraper</i> komponente	32
5.2. Ocjena komponente NER API	33
6. Moguća unaprjeđenja i smjernice za budući razvoj	35
6.1. <i>Scrape</i> proces	35
6.2. NER API	35
6.3. Web stranica	37
7. Legalna napomena	38
8. Zaključak	39
Literatura	40

1. Uvod

Ljudi već dugo vremena javnosti na internetu dobrovoljno daju svoje podatke - imena, prezimena, datume rođenja i sl. Često bismo htjeli za određenu grupu ljudi pronaći što više toga možemo na internetu i trenutno to većinom radimo ručno. Tako na primjer, ako za bivše studente FER-a želimo pronaći informacije o tome gdje sada rade, jesu li postigli neka značajna postignuća u životu ili o bilo kojem događaju u čijem se kontekstu spominju, za to ne postoji jednostavan način. Jedna mogućnost bila bi da, ako slučajno naletimo na neku osobu u nekom novinskom portalu, znanstvenom radu ili časopisu - spremimo poveznicu na taj resurs i tako ručno radimo posao kako bi netko koga potencijalno zanima osoba koja se u resursu spominje mogla imati sva njegova javna postignuća i događaje na jednom mjestu. 

2. Opis problema



Kako bismo prestali po internetu tražiti događaje i postignuća zadanog skupa ljudi, možemo si olakšati taj postupak automatizacijom dijelova postupka koje radimo ručno. Konkretno, prvo moramo odrediti koje izvore ćemo koristiti i na koji način, a zatim bismo u tim izvorima trebali pronaći koja se sve imena spominju te da li se spominje neko od onih koje mi tražimo.

Za potrebe testiranja ovog automatiziranog sustava kojim ćemo pronalaziti i filtrirati imena koja tražimo, odabrali smo nekoliko glavnih hrvatskih internetskih portala. Kroz sve te portale ćemo periodički prolaziti i tražiti sve članke koji se pojavljuju. Pri tom ćemo morati paziti da ne uzimamo u obzir članke koje smo već u prošlosti prošli, odnosno da ne stvaramo duplikate članaka u bazi.

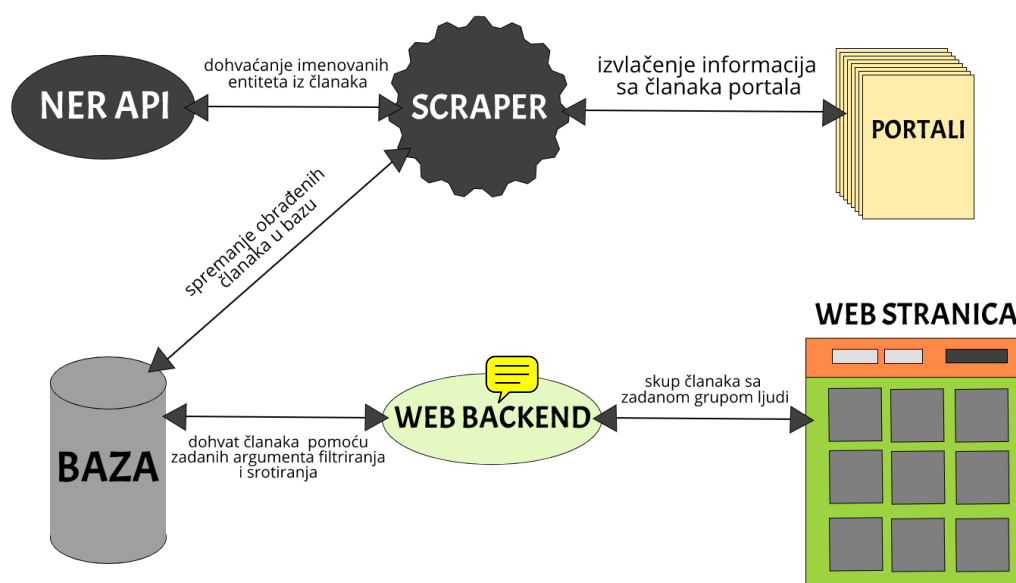
Traženje članaka koji se nalaze na stranicama portala obavljat ćemo postupcima koji se nazivaju *web scraping* i *web crawling*. Zatim ćemo pomoću programa za pronalazak imenovanih entiteta (engl. *named entity recognition*, *NER*) pronaći sva imena osoba koja se pojavljuju u tekstu te ćemo tako zaključiti da li se osoba koju tražimo nalazi u tekstu članka.

Na kraju ćemo za sve pronađene članke omogućiti pristup na web stranici. Na toj stranici korisnici će moći filtrirati pronađene članke biranjem grupe željenog skupa ljudi.

Cijeli postupak ćemo na kraju ocijeniti kako bismo imali pregled uspješnosti.

3. Arhitektura sustava

Sustav će se sastojati od nekoliko komponenti koje kontinuirano rade.



Slika 3.1: Arhitektura sustava za prikupljanje javno dostupnih imena sa hrvatskih portala

Na slici 3.1 nalazi se pregled svih komponenti koje će sudjelovati u procesu prikupljanja javno dostupnih imena sa stranica hrvatskih portala. *Scraper*, odnosno proces za izvlačenje podataka s web stranica, prikupljat će članke koje pronađe na stranicama portala. Ulazne točke za portale bit će mu stranice koje su na neki način popisi svih nedavnih članaka za zadani portal.

Nakon toga, *scraper* će slati izvučene članke NER API-ju koji će pomoću teksta dobivenih članaka pronaći imenovane entitete. Konkretno, API će pronalaziti imena i prezimena osoba koje se spominju u danim tekstovima.

Kada je NER API pronašao entitete, vratit će ih *scaperu* koji će napraviti dodatnu pripremu članaka, dodatno dobivene entitete te takve članke spremiti u bazu.

Na kraju ćemo imati sažeti prikaz članaka u obliku web stranice. Ta stranica neće

imati izravnu komunikaciju s bazom, već će se oslanjati na web poslužitelj. Zadaća tog web poslužitelja bit će filtriranje i sortiranje članaka iz baze po argumentima koje dobiva preko korisničkih zahtjeva s web stranice. U nastavku ćemo sve navedene komponente sustava i komunikaciju među njima detaljnije opisati.

3.1. *Web scraping i web crawling*

Web scraping i *web crawling* su usko povezani pojmovi. *Web crawling* je postupak dohvaćanja HTML (engl. *HyperText Markup Language*) strukture stranica i praćenja poveznica koje se spominju u njima. Pod pojmom *web scraping* podrazumijeva se izvlačenje samih podataka iz HTML sadržaja dobivenog *crawlanjem*. Ovakva vrsta izvlačenja podataka nije idealna, već bi bilo bolje kada bi portali s kojih tražimo tekstove članaka izdavali nekakvo aplikacijsko programsko sučelje (engl. *application programming interface, API*), preko kojeg bi izravno mogli dohvaćati tekst.

Glavna prednost takvog izvlačenja bila bi ta da ne trebamo brinuti o potencijalnim promjenama strukture stranica. Naime, jedan od glavnih nedostataka *scrapinga* je osjetljivost na bilo kakvu promjenu u HTML strukturi. Kako se prilikom izvlačenja informacija oslanjamo na meta-podatke koji se nalaze u HTML-u stranice, svaka promjena tih meta-podataka natjerat će nas da promijenimo i logiku našeg *scraper*a. Dobivanje tih informacija izravno od API znatno bi poboljšalo robusnost sustava jer on ne bi bio toliko sklon greškama prilikom navedenih promjena. Još jedan nedostatak postupka *scrapinga* u odnosu na API je brzina kojom se podaci dohvaćaju. Prilikom obrađivanja stranice, u sam postupak čak dodajemo umjetnu pauzu kako nas poslužitelj na kojem se nalazi stranica koju obrađujemo ne bi izbacio zbog prevelikog broja zahtjeva u kratkom vremenu.

Portali koje ćemo koristiti u ovome radu nemaju svoje API-je, tako da ćemo se morati okrenuti postupku *scrapinga*.

Pronalazak podataka u strukturama dobivenim *crawlerom* znatno je olakšao XPath [1] (engl. *XML Path Language*). XPath je upitni jezik za adresiranje dijelova XML-a (engl. *Extensible Markup Language*). Pomoću zadanog teksta nalik URL (engl. *universal resource locator*) putanjama možemo navigirati po hijerarhijskoj strukturi DOM-a (engl. *document object model*).¹ To je velika prednost u odnosu na izvorne funkcije browsera za dohvaćanje HTML čvorova kao što su *Document.getElementById()* ili *Document.querySelector()*. Tako na primjer samo pomoću idućeg XPath izraza možemo dohvatiti sve čvorove *naslov* za knjige u knjižari koje koštaju više od 130 kuna:

3.1.1. *Robots.txt* datoteka

Jedna bitna stvar koju bismo trebali provjeriti prije početka samog *scraping* procesa je provjera *robots.txt* datoteke u vršnom dijelu stranice, na URL-u `<url-stranice>/robots.txt`. To je datoteka koja postoji specifično za *web crawlere*. Pomoću nje stranica može zabraniti pristup nekim svojim dijelovima radi zaštite - u prošlosti je to bilo bitno zbog *DDoS* (engl. *Distributed Denial of Service*) napada koji bi se mogli dogoditi ili namjerno, ili kao uzrok loše napravljenog *web crawlera*. Danas postoji puno mehanizama kojima se stranice brane od takvih napada, tako da je *robots.txt* postao dodatan način na koji web stranice izjavljuju da ne žele automatiziranu pretragu nekog njihovog dijela.

```
User-agent: *
Disallow: /administrator/
Disallow: /bin/
Disallow: /cache/
Disallow: /cli/
Disallow: /components/
Disallow: /includes/
Disallow: /installation/
Disallow: /language/
Disallow: /layouts/
Disallow: /libraries/
Disallow: /logs/
Disallow: /modules/
Disallow: /plugins/
Disallow: /tmp/
Disallow: /component/com_mailto/
Disallow: /article-preview/
```

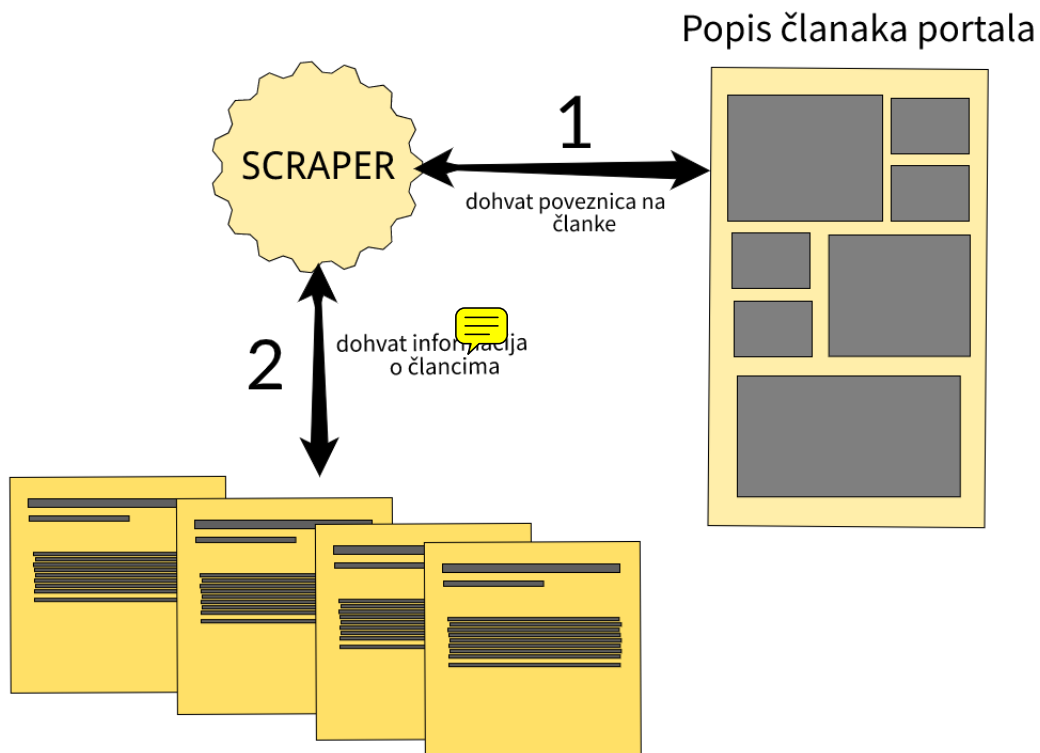
Slika 3.2: Sadržaj datoteke *robots.txt* sa stranice *jutarnji.hr* (*jutarnji.hr/robots.txt*)

Na prvoj liniji datoteke sa slike 3.2 vidimo da se zabrane odnose na sve korisničke agente (engl. *user-agent*) te su sve naredne linije relativne putanje koje ne bismo smjeli posjećivati. Znamo da prilikom ekstrakcije sadržaja nećemo posjećivati niti jednu od tih putanja i time poštujemo smjernice za *crawlere*.

3.1.2. Izvlačenje poveznica na članke s popisa vijesti

Svaka stranica ima drukčiju strukturu i nazive čvorova, što na prvu pomisao vodi na to da ćemo morati za svaki portal napisati zaseban *scraper*. No, izvlačenje sadržaja s

portala koje koristimo može se svesti na dva koraka: izvlačenje poveznica na članke s naslovnica portala te izvlačenje samih informacija o članku jednom kada dođemo na njegovu stranicu. Tako će kôd kojim se definira postupak izvlačenja informacija biti generaliziran, dok će specifičnosti za svaki portal, kao što su XPath varijable ili pomoćne funkcije, biti odvojene kako bismo izbjegli nepotrebnu duplikaciju. Na slici 3.3 nalazi se postupak izvlačenja podataka u dva koraka.



Slika 3.3: Generalizacija postupka ekstrakcije informacija o člancima

Unatoč dobro generaliziranom kôdu, svaku je stranicu trebalo pregledati i pronaći meta-podatke pomoću kojih bi se mogle izvući poveznice na sve članke. Prvo, naslovnica nije baš najbolja početna točka za traženje tih poveznica, jer ona osim poveznica ima i puno reklama i dodatnih elemenata koji su dobri za dojam korisnika, ali za pronalazak članaka nisu previše bitni. Tako da kao polazne točke odabiremo stranice koje baš ciljano sadrže popis svih članaka. Na primjer, na *jutarnji.hr*, stranica koja u jednostavnijoj strukturi, u obliku popisa, sadrži sve članke je *jutarnji.hr/vijesti*. Da stvar bude još jednostavnija, sve poveznice koje se nalaze na toj stranici, a počinju s */vijesti*, vrlo su vjerojatno poveznice na članak, pa temeljem toga možemo jednostavno upotrijebiti sljedeći XPath upit:

```
evalString = '//a[starts-with(@href, "/vijesti/")]'
```

Na slici 3.4 možemo primijetiti da *href* atribut *anchor* čvora na stranici *jutarnji.hr/vijesti* počinje sa */vijesti*.

```
▼<a class="card_article-link" href="/vijesti/svijet/milorad-dodik-od-clanstva-b  
osne-i-hercegovine-u-nato-savezu-nema-nista-15082324" title="Milorad Dodik: Od č  
lanstva Bosne i Hercegovine u NATO savezu nema ništa">
```

Slika 3.4: Primjer poveznice na članak sa stranice *jutarnji.hr/vijesti*

3.1.3. Izvlačenje informacija sa stranice članka

Na slici 3.5 vidimo generalizirani kôd koji, uz neke specifičnosti za svaku stranicu, dohvaća informacije o članku kao što su naslov, sadržaj, poveznicu na taj članak i sažetak koji se sastoji od poveznice i naslova članka.

```
57     await page.goto(postLink);  
58     await page.addScriptTag({ content: portal });  
59  
60     const eval1 = async titleString => document.querySelector(titleString).textContent;  
61     const eval2 = async contentString => document.querySelector(contentString).textContent;  
62  
63     const title = String(await page.evaluate(eval1, portal.titleString));  
64  
65     let text;  
66     if (portal.specificTextGetter) {  
67         text = await portal.specificTextGetter(page);  
68     } else {  
69         text = [String(await page.evaluate(eval2, portal.contentString))];  
70     }  
71  
72     text = text.map((paragraph) => paragraph.replace(/\s+/gm, ' '));  
73  
74     return {  
75         title,  
76         text,  
77         url: postLink,  
78         hash: crypto.createHash('md5').update(postLink + ' ' + title).digest('hex'),  
79     }
```

Slika 3.5: Generalizirani kôd pomoću kojeg se obrađuju sve stranice

Informacije se dohvaćaju tako da koristimo posebnosti stranice sa slike 3.6. Za dohvaćanje naslova članka koristimo *.item__title*, što je CSS (engl. *Cascading Style Sheets*) oznaka klase na elementu koji sadrži naslov. Isto tako za sadržaj članka koristimo oznaku *.itemFullText*.

```

25   jutarnji: {
26     url: 'http://jutarnji.hr/vijesti/',
27     evalString: '//a[starts-with(@href, "/vijesti/")]',
28     titleString: '.item__title',
29     contentString: '.itemFullText',
30     filters: [
31       containsRegex(/-\d+$/),
32       removeDuplicates
33     ],
34     cleanup(postlinks) {
35       postlinks.shift();
36     },
37     async specificTextGetter(page) {
38       const paragraphs = await page.$x('//div[@class="itemFullText"]//p');
39       let content = [];
40       for (let i = 0; i < paragraphs.length; i++) {
41         const p = await paragraphs[i].getProperty('textContent');
42         const text = p._remoteObject.value && p._remoteObject.value.trim();
43         if (text) content.push(text);
44       }
45
46       return content;
47     }
48   },

```

Slika 3.6: Kôd sa posebnostima za stranicu *jutarnji.hr*

Osim tih specifičnih *selectors*, za svaki portal definiramo i neke pomoćne funkcije kao što su *cleanup* i listu filter funkcija *filters*. *Cleanup* u ovom konkretnom slučaju postoji jer, kada dohvatimo sve poveznice na članke, dobivamo jednu poveznicu koja počinje sa */vijesti* i uvijek se nalazi na početku liste, ali nije zapravo poveznica na članak, pa je se želimo riješiti. Osim toga, može se pojaviti još takvih poveznica koje nisu stvarno poveznice. No, jedna stvar koja je svim poveznicama zajednička su brojevi na kraju koji prethode crticom, zbog čega se u jednoj od funkcija iz *filters* liste rješavamo svih poveznica koje nisu tog formata tako da koristimo *regex* (engl. *regular expression*, slijed znakova kojim definiramo obrazac za pretraživanje teksta). *Regex* koji u ovom slučaju koristimo je *-\d+\$*, koji se prevodi i nalazi sav tekst koji odgovara sljedećem opisu: crtica, popraćena jednom ili više brojki (označeno sa *\d+*), a sve to nalazi se na kraju teksta (označeno sa *\$*).

```

▼ <div class="itemFullText">
  ▶ <p>...</p>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--photo">...</div>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--onnetworktv">...</div>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--photo">...</div>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--photo">...</div>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--photo">...</div>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--photo">...</div>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--onnetworktv">...</div>
  ▶ <p>...</p>
  ▶ <div class="se-embed se-embed--onnetworktv">...</div>
</div>
<div class="position_item_textend_top"></div>
<div class="position_item_textend_bottom"></div>
<div class="row"></div> (flex)
▶ <div class="item_social-jl item_social-jl--bottom">...</div> (flex)
▶ <script>...</script>
▶ <div class="item_tags">...</div>
▶ <a href="mailto:jutarnjihr@hanzamedia.hr">...</a>
<div class="position_item_center_06_top"></div>
<div class="position_item_center_06_bottom"></div>
▶ <div class="prikazi_komentare">...</div>
</div>

```

Slika 3.7: Svi elementi koji se nalaze unutar *div* elementa sa oznakom klase *.itemFullText*

Prilikom ekstrakcije samog teksta iz pojedinih članaka, moramo paziti na neželjene elemente koji bi se mogli pojaviti izravno u tekstu. Na primjer, na stranici *jutarnji.hr*, format članka je takav da između svakog paragrafa, koji je čisti tekst, postoje slike ili video isječki vezani uz članak. Za potrebe ovog rada, pošto tražimo samo imena i prezimena ljudi iz članaka, neće nam biti potrebne slike, video isječki ili ugrađeni (engl. *embedded*) sadržaj. Osim njih, u ovom se elementu nalaze i komentari, reklame i čak dijelovi *JavaScript* kôda, koji također u našem slučaju neće biti korisni. Na slici 3.7 vidimo sve te neželjene elemente, kojih se u ovom slučaju rješavamo micanjem svega što nije paragraf, odnosno `<p>` element. *XPath* kojim označavamo takav tekst je sljedeći:

```
'//div[@class="itemFullText"]//p'
```

koji se ukratko prevodi u: svi `<p>` elementi koji se nalaze unutar `<div>` elemenata kojima je CSS klasa *itemFullText*.

3.2. *Named Entity Recognition API*

Kada je *scraper* dohvatio podatke i čisti tekst članaka s portala, taj članak se može dalje proslijediti NER (engl. *Named Entity Recognition*) API-ju. Taj API koristi CLASSLA[2] model za prepoznavanje imenovanih entiteta i pomoću njega izvlači imena i prezimena ljudi u tekstovima članaka. Model je treniran sa oko 500,000 ručno anotiranih tokena iz *hr500k 1.0*[3] te 89,104 tokena sa hrvatskih *Twitter* objava, *ReLDI-NormTagNER-hr 2.1*[4].

3.2.1. Izvlačenje imena pomoću CLASSLA cjevovoda

CLASSLA je općenitiji alat za procesiranje teksta napravljen modifikacijama *open-source* StanfordNLP[5] projekta za pokretanje različitih alata kojima se može procesirati preko 60 jezika. Tekst se može tokenizirati, lematizirati, parsirati ovisnosti, provlačiti kroz prepoznavanje imenovanih entiteta i još mnogo toga. Svi navedeni procesori te cijelokupno područje koje se bavi procesiranjem jezika naziva se NLP (engl. *Natural Language Processing*).

Za korištenje CLASSLA knjižnice potreban nam je *python 3.6+* te ga koristimo na sljedeći način:

```
nlp = classla.Pipeline('hr', processors='tokenize,ner,lemma')
```

taj cjevovod tada jednostavno možemo samo pozvati nad željenim tekstom:

```
res_doc = nlp(text)
```

res_doc je tada rječnik koji sadrži sve informacije koje je procesor jezika sakupio. Odvojene riječi u obliku tokena, zatim njihova lema, odnosno izvorni oblik te riječi (nominativ za imenice, infinitiv za glagole...) te na kraju rezultat procesora za imenovane entitete koji riječ označava oznakom prikladnom za taj entitet.

```

Damir Vrbanović, koji je u aferi Dinamo, pravomoćno osuđen na tri godine zatvora, danas se trebao javiti na
izdržavanje kazne u Remetinec, no to se neće dogoditi. Jer je 13. svibnja zatražio odgodu zbog bolesti.
{id': 1, 'text': 'Damir', 'lemma': 'damir', 'ner': 'B-PER'}
{id': 2, 'text': 'Vrbanović', 'lemma': 'Vrbanović', 'misc': 'SpaceAfter=No', 'ner': 'I-PER'}
{id': 3, 'text': ',', 'lemma': 'ak', 'ner': 'O'}
{id': 4, 'text': 'koji', 'lemma': 'koji', 'ner': 'O'}
{id': 5, 'text': 'je', 'lemma': 'on', 'ner': 'O'}
{id': 6, 'text': 'u', 'lemma': 'u', 'ner': 'O'}
{id': 7, 'text': 'aferi', 'lemma': 'aferi', 'ner': 'O'}
{id': 8, 'text': 'Dinamo', 'lemma': 'dinati', 'misc': 'SpaceAfter=No', 'ner': 'B-ORG'}
{id': 9, 'text': ',', 'lemma': 'ak', 'ner': 'O'}
{id': 10, 'text': 'pravomoćno', 'lemma': 'pravomoćan', 'ner': 'O'}
{id': 11, 'text': 'osuđen', 'lemma': 'osuđen', 'ner': 'O'}
{id': 12, 'text': 'na', 'lemma': 'ni', 'ner': 'O'}
{id': 13, 'text': 'tri', 'lemma': 'tri', 'ner': 'O'}
{id': 14, 'text': 'godine', 'lemma': 'godini', 'ner': 'O'}
{id': 15, 'text': 'zatvora', 'lemma': 'zatvor', 'misc': 'SpaceAfter=No', 'ner': 'O'}
{id': 16, 'text': ',', 'lemma': 'ak', 'ner': 'O'}
{id': 17, 'text': 'danas', 'lemma': 'danas', 'ner': 'O'}
{id': 18, 'text': 'se', 'lemma': 'se', 'ner': 'O'}
{id': 19, 'text': 'trebao', 'lemma': 'trebati', 'ner': 'O'}
{id': 20, 'text': 'javiti', 'lemma': 'javiti', 'ner': 'O'}
{id': 21, 'text': 'na', 'lemma': 'ni', 'ner': 'O'}
{id': 22, 'text': 'izdržavanje', 'lemma': 'izdržavanje', 'ner': 'O'}
{id': 23, 'text': 'kazne', 'lemma': 'kazni', 'ner': 'O'}
{id': 24, 'text': 'u', 'lemma': 'u', 'ner': 'O'}
{id': 25, 'text': 'Remetinec', 'lemma': 'remetinuti', 'misc': 'SpaceAfter=No', 'ner': 'B-LOC'}
{id': 26, 'text': ',', 'lemma': 'ak', 'ner': 'O'}
{id': 27, 'text': 'no', 'lemma': 'no', 'ner': 'O'}
{id': 28, 'text': 'to', 'lemma': 'taj', 'ner': 'O'}
{id': 29, 'text': 'se', 'lemma': 'se', 'ner': 'O'}
{id': 30, 'text': 'neće', 'lemma': 'neći', 'ner': 'O'}
{id': 31, 'text': 'dogoditi', 'lemma': 'dogoditi', 'misc': 'SpaceAfter=No', 'ner': 'O'}
{id': 32, 'text': '.', 'lemma': 'ak', 'ner': 'O'}

```

Slika 3.8: Ispis CLASSLA NLP sustava za zadani tekst

Na slici 3.8 je ispis NLP sustava u kojem vidimo sve informacije koje je NLP izvu- kao prilikom procesiranja teksta koji se nalazi na vrhu ispisa. Izvorni tekst podijeljen je na tokene, koji se nalaze u rječniku pod ključem 'text'. Izvorni oblici nalaze se pod ključem 'lemma', dok se izrazi NER procesora nalaze pod ključem 'ner'. Tako na primjer vidimo da je pod 'id': 15 sustav pronašao riječ *zatvora* te zaključio da je izvorni oblik te riječi *zatvor*. Na mjestu 'id': 8 nalazi se organizacija *Dinamo*, koju je NER označio sa 'B-ORG'.

Bitnije stavke kao što su organizacije, mjesta i osobe, označene su sa *ORG*, *LOC* i *PER*. Te se oznake nikada ne pojavljuju same za sebe nego uvijek imaju 'B', 'I' ili 'O' na početku. Takav zapis dio je IOB (engl. *Inside-outside-begginng*) formata, prvi put spomenutog u *Text Chunking using Transformation-Based Learning, 1995*[6], kojim označavamo poziciju tokena unutar pronađenog entiteta. Prefiksom 'B' označavamo početak entiteta, pomoću 'I' unutrašnjost, dok nam 'O' govori da se token nalazi *outside*, odnosno nije dio entiteta. Tako na slici 3.8 vidimo da većina tokena ima oznaku 'O', pošto nisu dio neke cjeline vrijedne razmatranja. Na primjer, u rečenici *Čekam te na Trgu žrtava fašizma*, cijeli entitet *Trgu žrtava fašizma* bio bi označen kao lokacija, i to tako da je token *Trgu* označen kao 'B-LOC', dok su tokeni *žrtava* i *fašizma* označeni kao 'I-LOC'. Svi ostali tokeni, *Čekam*, *te* i *na*, bili bi označeni oznakom 'O'.

Mi želimo pomoću tih tokena izvući imena i prezimena osoba koje se nalaze u tekstu. Ta imena mogu se pojaviti u bilo kojem obliku, tako da nam je bitno pronaći izvorni oblik tih imena, odnosno njihov nominativ. Za entitet *Tomislavu Kravarščanu* želimo generirati izvorni oblik *Tomislav Kravarščan*, za entitet *Ivanom Horvatom* želimo generirati *Ivan Horvat* itd. Pošto nam procesor za lematizaciju već pronalazi izvorni oblik pronađenog tokena, mogli bismo iskoristiti to za sve tokene entiteta osobe i tako bismo već imali neko rješenje. No, u praksi se pokazalo da procesor kojim se lematiziraju tokeni ne daje baš najbolje rezultate. To vidimo čak i na slici 3.8, gdje je procesor za token 'id' : 7, odnosno *aferi*, zaključio da se radi o izvornom obliku koji glasi *aferi*, dok mi znamo da se zapravo radi o imenici *afera*. Kada su u pitanju mjesta i organizacije te njihovi izvorni oblici, vidimo da je procesor za lematizaciju tu pogotovo podbacio. Za *Dinamo* je zaključio da je izvorni oblik *dinati*, dok je za *Remetinec* zaključio *remetinuti*. Isto tako često podbacuje na imenima, iako je u ovom slučaju - za entitet *Damir Vrbano* i oba tokena tog entiteta, dobro zaključio izvorne oblike, izuzev malog slova u lemi *damir*.

3.2.2. Poboljšavanje rezultata NER-a pomoću baze hrvatskih imena i prezimena

Kako bismo pokušali poboljšati rezultate koje će naš NER API vratiti, iskoristit ćemo rad *Leksička flektivna bazu podataka hrvatskih imena i prezimena* [7], u kojoj se nalaze sva javno dostupna hrvatska imena, sa svim njihovim oblicima koji su u skladu sa pravilima hrvatskog jezika. Sva imena i prezimena iz te baze upisana su u *hrLex 1.3* [8]. To je veliki rječnik hrvatskog jezika sa jako puno riječi i njihovih klasifikacija.

MSD sintaksa

662901	Tomislav	Tomislav	Npfsa	3
662902	Tomislav	Tomislav	Npf sd	0
662903	Tomislav	Tomislav	Npf sg	25
662904	Tomislav	Tomislav	Npf si	0
662905	Tomislav	Tomislav	Npf sl	1
662906	Tomislav	Tomislav	Npf sn	119
662907	Tomislav	Tomislav	Npf sv	0
662908	Tomislav	Tomislav	Npmsn	59941
662909	Tomislava	Tomislav	Npmsay	1576
662910	Tomislava	Tomislav	Npmsg	18907
662911	Tomislava	Tomislava	Npf sn	1668
662912	Tomislava	Tomislava	Npf sv	0
662913	Tomislave	Tomislav	Npmsv	3
662914	Tomislave	Tomislava	Npf sg	43
662915	Tomislavi	Tomislava	Npf sd	0
662916	Tomislavi	Tomislava	Npf sl	0

Slika 3.9: Skraćen dio zapisa *hrLex* datoteke

Na slici 3.9 vidimo dio zapisa obrađene *hrLex* datoteke, koja ima redom: oblik riječi u kojima bi se mogla pojaviti, izvorni oblik te riječi, njen morfosintaktički opis (engl. *morphosyntactic description*, *MSD*) te frekvenciju pojavljivanja te riječi u hrvatskom jeziku. U MSD-u su sadržane sve pojedinosti riječi koju opisuje, a svaka kategorija riječi ima svoju zasebnu specifikaciju za format MSD-a.

P	Attribute (en)	Value (en)	Code (en)
0	CATEGORY	Noun	N
1	Type	common	c
		proper	p
2	Gender	masculine	m
		feminine	f
		neuter	n
3	Number	singular	s
		plural	p
4	Case	nominative	n
		genitive	g
		dative	d
		accusative	a
		vocative	v
		locative	l
		instrumental	i
5	Animate	no	n
		yes	y

Slika 3.10: Specifikacija za hrvatske imenice

Specifikacija za imenice nalazi se na slici 3.10. Vidimo da imenice na prvom mjestu imaju kategoriju N, odnosno *Noun*, zatim tip koji označava radi li se o općoj ili posvojnoj imenici, onda informacija o tome je li imenica u jednini ili množini, na kon čega slijedi njena deklinacija. Na kraju se nalazi atribut *Animate*, koji označava odnosi li se imenica na nešto živo.

Generiranje izvornih oblika riječi pomoću *hrLex* datoteke

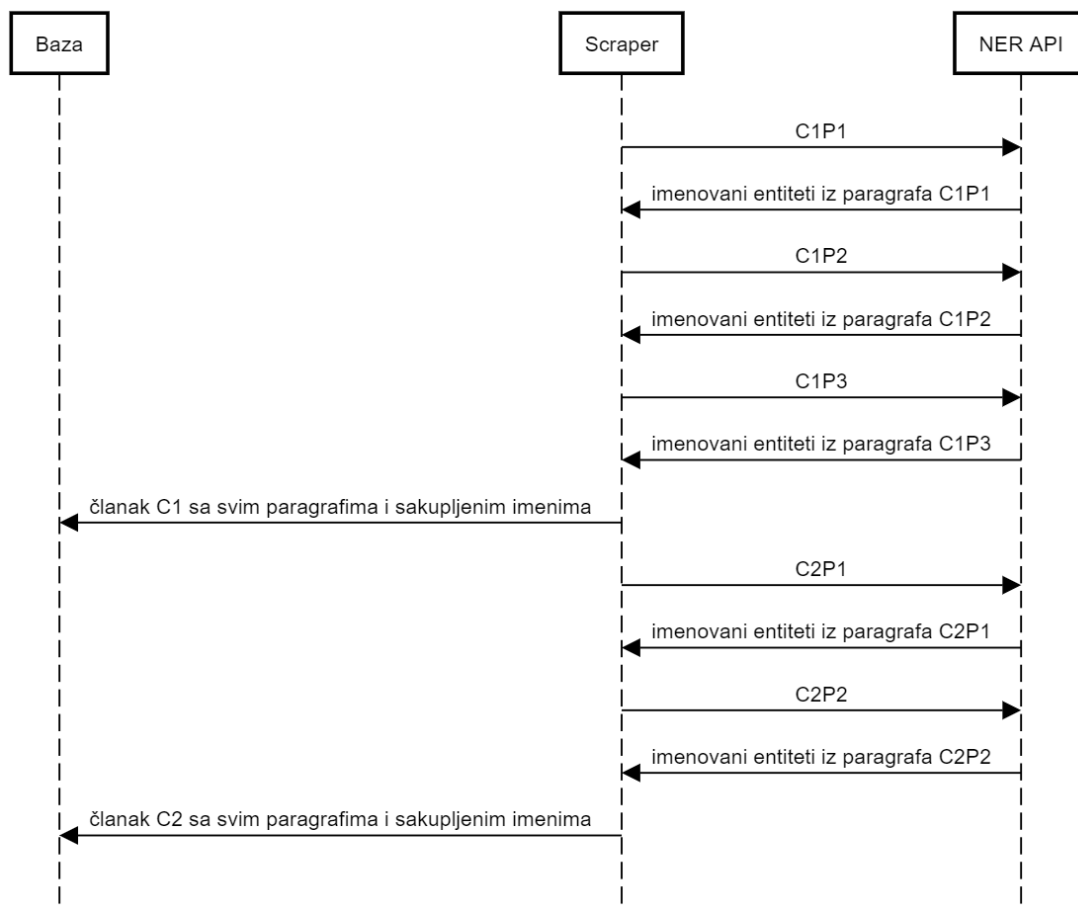
Kako bismo izvukli sve imenice koje su najvjerojatnije imena i prezimena, iz *hrLex* datoteke filtrirali smo sve što nije *proper noun*, odnosno posvojna imenica te sve što nije u jednini. Tako smo smanjili veličinu *hrLex* datoteke i optimizirali brzinu pronalaska imena.

Kako nam izlaz NER-a nije bio zadovoljavajuć, sada možemo te filtrirane imenice koristiti za pronalazak nominativa imena pronađenih u člancima. Na primjeru dijela datoteke sa slike 3.9 možemo zaključiti da, ako se u tekstu nalazi imenica u obliku *Tomislava*, izvorni oblik je najvjerojatnije *Tomislav*.

Dakle, NER procesor je u tekstu pronašao token *Tomislava* i označio ga sa B-PER. Osim toga, dao nam je i svoju pretpostavku izvornog oblika imena koju ćemo za početak ignorirati, jer znamo da je ta pretpostavka često bila pogrešna. Umjesto toga, uzimamo token *Tomislava* i tražimo ga u skraćenoj *hrLex* datoteci, iz koje smo filtrirali sve što sigurno nije ime ili prezime. Kada pronađemo sve pojave tokena u prvom stupcu *hrLex* datoteke, kojih vidimo da će, ako se u obzir uzme samo slika 3.9, biti četiri. Te redove gdje se pojavljuje tada možemo sortirati po frekvenciji pojavljivanja u hrvatskom jeziku, koja se nalazi u posljednjem stupcu. Sortiranjem primjećujemo da se najčešće pojavljuje kao izvorni oblik *Tomislav*, sa 18,907 pojava te MDN izlazom *Npmmsg*, što znači da je to posvojna imenica, muškog roda, u jednini i u genitivu. Ukoliko u *hrLex* datoteci uopće ne pronađemo imenicu koju tražimo, kao rezervu ipak ćemo uzeti ono što je NER izvorno pretpostavio. Osim toga, kako bismo bili sigurniji, u popis imena koja će NER vratiti staviti ćemo i ime, odnosno prezime osobe u deklinaciji u kojoj se zaista i pojavljuje u tekstu. Znači, za pojavu entiteta u obliku *Tomislavom Kravarščanom*, NER će zapisati sljedeće imenovane entitete: *Tomislav Kravarščan* i *Tomislavom Kravarščanom*. U tom slučaju, ako se ime u tekstu već nalazi u nominativu, a NER zaključi da je nominativ tog imena ipak nešto drugo, osigurali smo da se ipak spremanjem entiteta u obliku u kojem se pojavio u tekstu smanji pogreška NER procesora.

Vidimo da ovaj način također nije idealan, jer ćemo uvijek uzeti samo najčešću pojavu neke imenice, što neće svaki put biti dobar odabir, ali pokazalo se da je tako ipak puno bolje nego samo uzimanje pretpostavke NER procesora.

Također, pokazalo se da NER najbolje radi ako je tekst koji obrađuje koherentan i sadrži jednu smislenu cjelinu. Pošto neki članci mogu sadržavati samo popis informacija te su paragrafi članaka često nepovezani, najbolje je bilo slati NER API-ju paragraf po paragraf članaka. On tada pronađe sva imena koja se nalaze u paragrafu, a na klijentu je da sva imena iz svih paragrafa sakupi i ukloni duplikate ako ih bude.



Slika 3.11: Sekvencijski dijagram postupka pronalaska imenovanih entiteta u paragrafima članaka

Na slici 3.11 vidimo sekvencijski dijagram komunikacije između *scraper*a, NER API-ja i baze. *Scraper* za svaki paragraf članka pozove NER API, koji mu vrati sva imena koja je pronašao u tom paragrafu. Kada NER API i *scraper* obrade sve paragrafe jednog članka, *scraper* sakupi ta imena i spremi ih u jednu listu koju će, s tekstom i ostalim bitnim informacijama o članku, spremati u bazu.

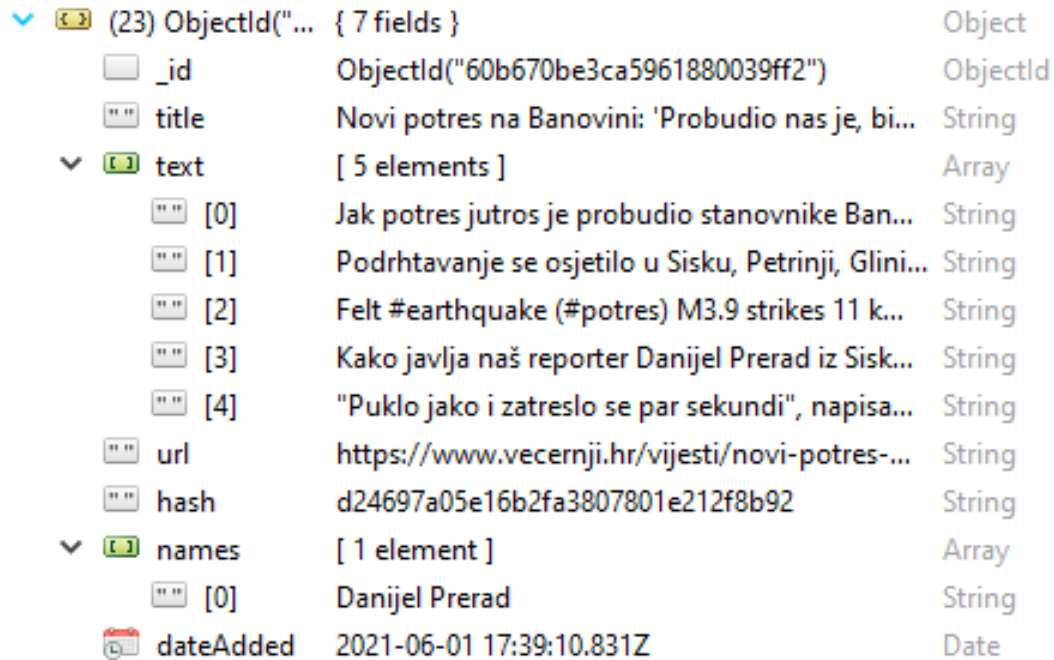
Dijagram sa slike 3.11 je pojednostavljen. *Scraper* radi optimizacije sprema u bazu tek kada sakupi sve članke s jednog portala, a u međuvremenu ih drži u memoriji.

3.3. Spremanje članaka s pronađenim imenovanim entitetima u bazu



Baza podataka koju koristimo je dokumentno orijentirana baza koja nativno radi s JSON (engl. *JavaScript Object Notation*) formatom. Odabrana je takva baza jer se

podaci primarno temelje na popisu članaka, pa je jednostavnost i brzina izrade imala prednost prilikom odabira. Baza sadrži dvije kolekcije: članci (*posts*) i grupe ljudi (*groups*).

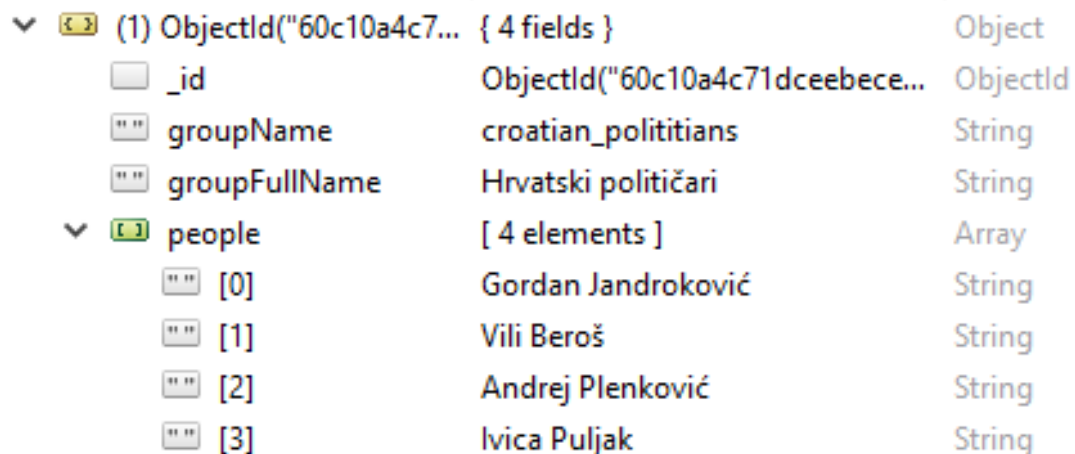


The screenshot shows a MongoDB document in the 'posts' collection. The document is an Object with 7 fields. The fields are: _id (ObjectId), title (String), text (Array of 5 elements), url (String), hash (String), names (Array of 1 element), and dateAdded (Date). The text field contains an array of 5 strings describing an earthquake in Banovina. The names field contains an array with the name 'Danijel Prerad'.

Field	Value	Type
_id	ObjectId("60b670be3ca5961880039ff2")	ObjectId
title	Novi potres na Banovini: 'Probudio nas je, bi...	String
text	[5 elements]	Array
[0]	Jak potres jutros je probudio stanovnike Ban...	String
[1]	Podrhtavanje se osjetilo u Sisku, Petrinji, Glini...	String
[2]	Felt #earthquake (#potres) M3.9 strikes 11 k...	String
[3]	Kako javlja naš reporter Danijel Prerad iz Sisk...	String
[4]	"Puklo jako i zatreslo se par sekundi", napisa...	String
url	https://www.vecernji.hr/vijesti/novi-potres-...	String
hash	d24697a05e16b2fa3807801e212f8b92	String
names	[1 element]	Array
[0]	Danijel Prerad	String
dateAdded	2021-06-01 17:39:10.831Z	Date

Slika 3.12: Zapis članka u kolekciji *posts*


Na slici 3.12 vidimo zapis jednog članka u *posts* kolekciji dokumentno orijentirane baze. Za taj članak spremamo naslov, tekst u obliku popisa paragrafa, poveznicu na članak, sažetak koji se dobiva spajanjem poveznice i naslova, popis imena koja se u njemu nalaze te datum kada je članak dodan u bazu.



The screenshot shows a MongoDB document in the 'groups' collection. The document is an Object with 4 fields. The fields are: _id (ObjectId), groupName (String), groupFullName (String), and people (Array of 4 elements). The people field contains an array of 4 strings representing names of Croatian politicians.

Field	Value	Type
_id	ObjectId("60c10a4c71dceebece...")	ObjectId
groupName	croatian_polititians	String
groupFullName	Hrvatski političari	String
people	[4 elements]	Array
[0]	Gordan Jandroković	String
[1]	Vili Beroš	String
[2]	Andrej Plenković	String
[3]	Ivica Puljak	String

Slika 3.13: Zapis grupe ljudi u kolekciji *groups*

Na slici 3.13 nalazi se zapis jedne grupe ljudi. Zapis sadrži ime grupe, puno ime grupe te popis osoba koje se nalaze u toj grupi. 

Provjera postojanja članaka u bazi

Za svaki članak za koji se pronađe poveznica na popisu članaka pojedinog portala treba provjeriti da li se on već nalazi u bazi. Postupak pronalaženja novih članaka odvija se periodički, tako da je za neke članke portala moguće da smo ih već posjetili, obradili i spremili. To možemo lako provjeriti tako da pokušamo pronaći članak u bazi. Ako se već tamo nalazi, ignoriramo ga, u suprotnom krećemo u izvlačenje podataka i pronalazak imenovanih entiteta te tako obrađen novi članak spremamo u bazu.

Pošto znamo da će ta provjera postojanja članka biti vrlo česta, nad kolekcijom članaka u bazi možemo postaviti indeks. Indeksiranjem dobivamo puno bolje performanse prilikom pretrage, jer se tako stvara struktura kojom je pretraživanje iznimno brzo. Kolekciju *posts* indeksirat ćemo po polju *hash*, odnosno po sažetku koji je kreiran sažimanjem spoja poveznice na članak i njegovog naslova:

```
post_hash = md5_hash(post.link + ' ' + post.title)
```

Indeks se u *MongoDB* bazi stvara na sljedeći način:

```
db.posts.createIndex( { hash: 1 }, { unique: true } )
```

što znači da za kolekciju *posts* stvaramo indeks nad ključem *hash* te želimo da taj ključ bude jedinstven, odnosno da sažetak dobiven spajanjem poveznice i naslova uvijek bude različit.

3.4. Periodičko izvršavanje procesa *scrapinga*

Sve navedene komponente trebale bi raditi kontinuirano. *Web poslužitelj* će na zahtjev klijenta vraćati filtrirane članke direktno iz baze, *NER API* će biti pokrenut cijelo vrijeme i čekati na upite *scrapera*, dok će *scraper* periodički posjećivati sve zadane portale, tražiti poveznice na članke i izvlačiti informacije. *Scraper* posjećuje svaki portal redom i kreće na sljedeći tek kada obradi sve članke s jednog portala.

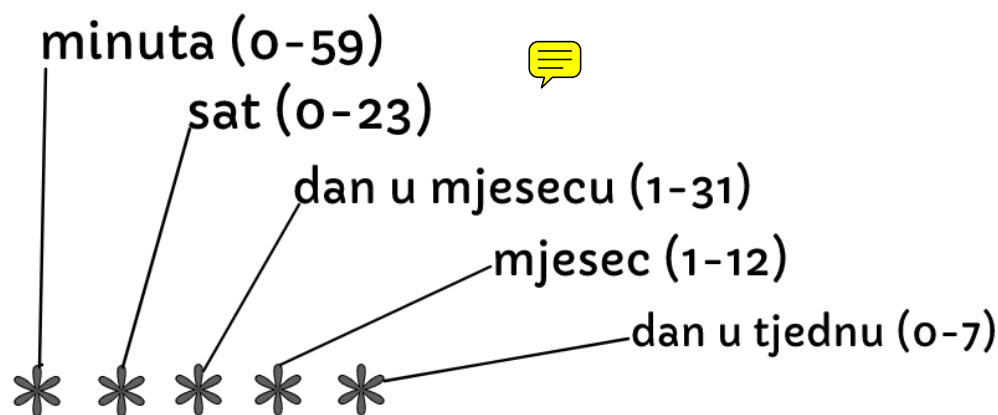
```
1 of 31 from https://www.ictbusiness.info/vijesti/hrvatska-ubrzala-
2 of 31 from https://www.ictbusiness.info/vijesti/acer-globalna-nes
3 of 31 from https://www.ictbusiness.info/vijesti/playrix-lansira-m
4 of 31 from https://www.ictbusiness.info/vijesti/idc-pc-trziste-ce
5 of 31 from https://www.ictbusiness.info/vijesti/okrsaji-najboljih
6 of 31 from https://www.ictbusiness.info/vijesti/na-virtualnom-sta
7 of 31 from https://www.ictbusiness.info/vijesti/prica-o-prvom-pla
8 of 31 from https://www.ictbusiness.info/vijesti/pc-emea-trziste-z
9 of 31 from https://www.ictbusiness.info/vijesti/it-pomaze-u-stvar
10 of 31 from https://www.ictbusiness.info/vijesti/bec-testira-nove
11 of 31 from https://www.ictbusiness.info/vijesti/do-2025-godine-t
12 of 31 from https://www.ictbusiness.info/vijesti/stagnacija-hrvat
13 of 31 from https://www.ictbusiness.info/vijesti/cak-33-7-milijar
14 of 31 from https://www.ictbusiness.info/vijesti/huawei-jeve-tehno
15 of 31 from https://www.ictbusiness.info/vijesti/pobjednici-u-bra
16 of 31 from https://www.ictbusiness.info/vijesti/ulaganja-u-istra
17 of 31 from https://www.ictbusiness.info/vijesti/novi-zakon-o-aut
18 of 31 from https://www.ictbusiness.info/vijesti/rimac-automobili
19 of 31 from https://www.ictbusiness.info/vijesti/atos-predstavlja
20 of 31 from https://www.ictbusiness.info/vijesti/hbo-go-u-optima-
21 of 31 from https://www.ictbusiness.info/vijesti/peta-esecurity-k
22 of 31 from https://www.ictbusiness.info/vijesti/zastita-radnika-
23 of 31 from https://www.ictbusiness.info/vijesti/za-10-godina-zan
```

 Slika 3.14: Rad *scrapera*

Na primjer, na slici 3.14 nalazi se ispis po kojem vidimo kako je *scrapper* pronašao ukupno 31 članak na stranici *ictbusiness.info* te obrađuje članke jedan po jedan. Kako ne bi preopteretio server portala koji posjećuje, *scrapper* čeka 15 sekundi nakon što je obradio jedan članak i tek onda kreće na idući.

3.4.1. *NodeJS cron job* za periodičko pokretanje *scrapera*

Cijeli proces dobavljanja informacija s portala izvodi se svakih pet sati. To je omogućeno *cron-job* knjižnicom za *NodeJS*. *Cron job* je poznat kao *Linux* naredba za planirano izvršavanje skripti. Ta naredba se izvršava u vremenskim intervalima koji su definirani u *crontab* datoteci. Može ju definirati korisnik, a postoji i *crontab* datoteka za cijeli sustav, koja se najčešće nalazi unutar */etc* direktorija.



Slika 3.15: Format zapisa u *crontab* datoteci

Na slici 3.15 vidimo kako sve možemo definirati *crontab*. Na primjer, zapisom 5 4 3 2 1 definirali smo da će se neka skripta pokretati u četiri sata i pet minuta, na treći dan veljače koji je ponedjeljak. U trenutku pisanja ovog rada, idući puta kada će to biti ostvareno je 3.2.2022., 04:05. Osim samih brojki, ako nam je svejedno, možemo staviti i * te možemo oznakom / definirati ponavljanje. Tako bi sa 0 0 */2 * * mogli definirati da će se skripta izvoditi u ponoć svaki drugi dan u mjesecu.

3.5. Web prikaz i filtriranje članaka

Sve pronađene članke treba sada prikazati na web stranici. U bazi se nalaze svi članci sa svim pronađenim entitetima, a zadaća web stranice će onda biti filtriranje tih članaka po želji. Konkretno, na stranici ćemo filtriranje omogućiti pomoću padajućeg izbornika kojim biramo grupu ljudi po čijim članovima želimo filtrirati.

Prikaz članaka

NAZAD

NAPRIJED

Svi postovi ▼

Grčka ukida maske na otvorenom

GRČKA je objavila da od četvrtka ujutro ukida obvezu nošenja maski na otvorenom, osim u slučaju velikih okupljanja, slijedeći isti korak koji su vezano za pandemiju koronavirusa već napravile Francuska, Španjolska i Italija. Usporedno s poboljšanjem zdravstvene situacije, Grčka će od ponedjeljka tak...

Datum: Wed, 23 Jun 2021 17:51:33 GMT

Spominju se: Nikos Hardalias, Van Papaevaggelo, Vana Papaevaggelou

Zbog vrućina Hitna ima puno posla, liječnica otkrila zašto ih sve ljudi zovu

SPLITSKA liječnica dr. Anita Belak Barišin bila je gošća Dnevnika N1 televizije te je pojasnila da je zbog velikih vrućina povećan broj intervencija. Otkrila je na što se pacijenti žale te kako se zaštititi od toplinskog udara. "S obzirom na visoke temperature, danas bilježimo značajno veći broj inte...

Datum: Wed, 23 Jun 2021 17:51:33 GMT

Spominju se: Anita Belak Barišin, Belak Barišin

Navalnijev pravni tim objavio detalje presude o ekstremizmu

PRAVNI tim pritvorenog kritičara Kremlja Alekseja Navalnog danas je objavio puni tekst presude u kojoj je politička mreža tog ruskog aktivista 9. lipnja proglašena ekstremističkom. Presudom koju su Navalniji odvjetnici proglasili gušenjem njegovih aktivnosti bez presedana njegovim se saveznicima za...

Datum: Wed, 23 Jun 2021 17:51:33 GMT

Spominju se: Kremlje Aleksej Navaln, Kremlja Alekseja Navalnog, Vladimir Putin, Vladimira Putina, Navaln, Navalnog, Putin, Joea Biden, Saveznik Navaln, Joeom Bidenom, Saveznici Navalnog, Navalni

Grlić Radman: Od Srbije se traže jasni odgovori

SRBIJA na svom putu prema EU treba pokazati suštinski i održiv napredak u temeljnim područjima i pokazati volju za ispunjavanje svih potrebnih kriterija, izjavio je u utorak u Luxembourg hrvatski ministar vanjskih poslova Gordan Grlić Radman. "Od Srbije se traže jasni odgovori, prije svega suštinski...

Datum: Wed, 23 Jun 2021 17:51:33 GMT

Spominju se: Gordan Grlić Radman, Grlić Radman

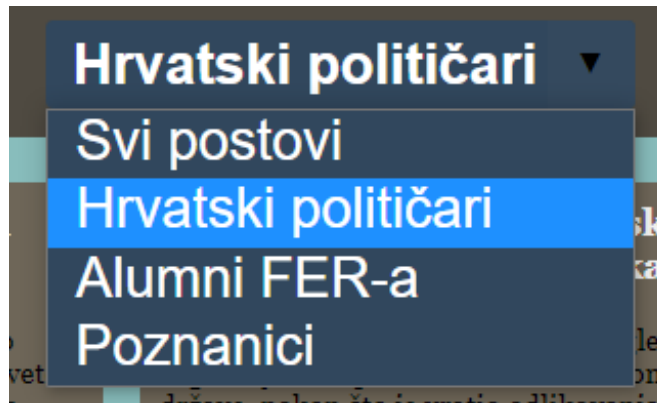
Ukinuta odluka koja je sprječavala izručenje braće Mamić

Prvi put se sastali gradonačelnici Banje Luke i Sarajeva. 26 godina nakon rata

Slika 3.16: Izgled web stranice za pregled sažetaka filtriranih članaka

Na slici 3.16 vidimo izgled web stranice koja služi kao grafičko sučelje za prikaz filtriranih članaka. Korisnik može listati kroz članke pritiskom na tipke `NAPRIJED` ili `NAZAD`. Za svaki članak postoji kartica u kojoj korisnik može vidjeti naslov, početak teksta članka, datum i vrijeme kada je članak posjećen te popis imenovanih entiteta koji se spominju u tekstu članka.

Biranje grupe za filtriranje



Slika 3.17: Odabir grupe za filtriranje članaka

Osim navedenih osnovnih funkcionalnosti pregledavanja članaka, pomoću padajućeg izbornika na slici 3.17 korisnik može odabrati grupu ljudi koju želi tražiti u tekstovima članaka. Pritiskom na neku grupu, web poslužitelj filtrira članke po članovima te grupe te vraća samo one koji se nalaze u *Spominju se* dijelu članka.

3.5.1. Web poslužitelj

Web stranica ne komunicira izravno s bazom prilikom filtriranja članaka, već svoje zahtjeve šalje web poslužitelju koji taj posao odrađuje umjesto nje. Preko poslužitelja pozivom `GET` i `POST` zahtjeva omogućene su funkcionalnosti dodavanja grupe imena po kojima će se filtrirati, dohvaćanja svih postojećih grupa, dohvaćanja filtriranih članaka te dohvaćanja pojedinih članaka iz baze.

Dodavanje grupe imena

Dodavanje grupe imena ostvarujemo pozivom `POST` metode na krajnjoj točki `/add_group`. Prilikom slanja zahtjeva u njegovom tijelu mora se nalaziti JSON objekt kojim ćemo definirati grupu. Primjer tog objekta nalazi se na slici 3.18.

```

1  {
2    "groupName": "croatian_politicians",
3    "groupFullName": "Hrvatski političari",
4    "people": [
5      "Andrej Plenković",
6      "Tomislav Tomašević",
7      "Vesna Škare Ožbolt",
8      "Gordan Jandroković",
9      "Vili Beroš"
10   ]
11 }

```

Slika 3.18: Primjer formata objekta kojim se definira grupa

Dohvaćanje svih grupa

Dohvaćanje grupa u bazi ostvaruje se pozivom krajnje točke `/groups`. Dobivamo se postojeće grupe u obliku prikazanom na slici 3.19.

```

1  [
2    {
3      "groupFullName": "Hrvatski političari",
4      "groupName": "croatian_politicians"
5    },
6    {
7      "groupFullName": "Alumni FER-a",
8      "groupName": "fer_alumni"
9    },
10   {
11     "groupFullName": "Poznanici",
12     "groupName": "people_i_know"
13   }
14 ]

```

Slika 3.19: Podaci dobiveni pozivom `/groups` krajnje točke poslužitelja

Primijetimo da se u podacima nalaze samo imena grupa po kojima možemo filtrirati. Entiteti koji su unutar tih grupa ne trebaju biti poznati klijentu, već ih samo poslužitelj na temelju tih imena dohvaća iz baze.

Dohvaćanje filtriranih članaka

Filtrirane članke možemo dohvatiti pozivanjem krajnje točke `/posts`. Ova krajnja točka može primiti i opcionalne parametre kao što su `count` - broj članaka koji želimo dohvatiti, `start` - od kuda počinjemo dohvaćati članke te `group` - ime grupe po čijim članovima želimo filtrirati članke. Tako bi jedan poziv kojim želimo filtrirati članke po

članovima grupe "Hrvatski političari" te dohvatiti njih 30, počevši od 2006^g izgledao ovako: /posts?group=croatian_politicians&count=30&start=20. U odgovoru poslužitelja dobit ćemo popis članaka u obliku JSON objekta, a primjer jednog članka u tom popisu nalazi se na slici 3.20. Početna vrijednost za atribut grupe je "all", za broj članaka "10" te za početak "0". To su vrijednosti s kojima će stranica dohvatiti članke prilikom prvog posjeta.

```
1  [
2    {
3      "dateAdded": "Wed, 23 Jun 2021 17:43:58 GMT",
4      "hash": "9108c5afecc0b730ec802d7346d4fca1",
5      "names": [
6        "Zoran Milanović",
7        "Tomislav Tomašević",
8        "Tomašević",
9        "Milanović",
10       "Tomislava Tomaševića",
11       "Tomaševića"
12     ],
13     "text": [
14       "- To se zove vladanje. To je njegova odgovornost  

15         nesposobna lijenčina - rekao je u intervjuu  

16         poteze gradonačelnika Zagreba Tomislava Toma  

17         I nakon opaske novinarkе da je Tomašević ime  

18         zapošljavanja mimo javnih natječaja, Milanov  

19       "- To nije zapošljavanje. To su čelne pozicije I
20     ],
21     "title": "Iza kulisa preokreta u retorici: Milanović  

22     "url": "https://www.jutarnji.hr/vijesti/hrvatska/  

23       iza-kulisa-preokreta-u-retorici-milanovic-podrz  

24     },
25   ],
26 }
```

Slika 3.20: Odgovor koji sadrži članak dobiven filtriranjem po članovima grupe "Hrvatski političari"

Web stranica koristeći ove informacije može ispisati naslov i tekst članka, imena koja se u njemu spominju, datum kada je članak posjećen te omogućava praćenje poveznice do izvornog članka na portalu na kojem se nalazi.

4. Korištene tehnologije

4.1. *JavaScript*

JavaScript je objektno orijentirani interpretirani jezik pomoću kojeg možemo implementirati kompleksne funkcionalnosti. To je najrašireniji jezik na svijetu jer ga koriste svi web preglednici. Po njegovoj specifikaciji često se spominje i u obliku *ECMAScript* [9]. To je dokument pomoću kojeg se pokušava standardizirati *JavaScript* tako da su njegove funkcionalnosti predefinirane. Svaki preglednik ima pravo koristiti verziju *JavaScripta* i podskup njegovih funkcionalnosti koji želi, što često stvara probleme kompatibilnosti prilikom razvoja web stranice. *JavaScript* najčešće koristimo za manipulaciju HTML i CSS elemenata web stranice, a možemo ga koristiti i izvan preglednika pomoću *NodeJSa*.

JavaScript se u ovom radu koristi na više mjesta, u sklopu *NodeJS* izvršne okoline za *scraping* te u sklopu *VueJS* radne okoline za web stranicu s prikazom filtriranih članaka.

4.2. *NodeJS*

NodeJS [10] je izvršna okolina pomoću koje se *JavaScript* kôd može izvršavati van preglednika. Pomoću njega možemo koristiti *JavaScript* prilikom izrade web poslužitelja ili kao bilo koju skriptu opće namjene. Jedna od prednosti *NodeJS* okoline je dobar mehanizam upravljanja ovisnostima. Instalacijom *NodeJS* sustava dobivamo na korištenje i `npm` naredbu kojom možemo dodavati i brisati pakete koji su nam potrebni prilikom izrade projekta. Osim što možemo dodavati i brisati pojedinačne pakete, u korijenskom direktoriju *NodeJS* projekta možemo definirati datoteku *package.json* u kojoj se nalaze sve ovisnosti i meta-podaci o našem projektu. Tako pomoću naredbe `npm install` možemo odjednom instalirati sve pakete koji su navedeni u *package.json* datoteci, dok naredbom `npm init` kreiramo praznu *package.json* datoteku. Pojedinačni

paketi mogu se instalirati naredbom `npm install <ime-paketa>`.

U ovom radu *NodeJS* se koristi prilikom *scraping* procesa. Pomoću njega i knjižnice *Puppeteer* izvlačimo članke sa zadanih portala.

4.3. *Puppeteer*

Puppeteer [11] je *NodeJS* knjižnica koja korisniku na korištenje daje API kojim se može kontrolirati bezglavi (engl. *headless*) *Chrome* preglednik pomoću *DevTools* protokola. Taj protokol koristi se za izlaganje funkcionalnosti *Chrome* preglednika za potrebe raznih razvojnih alata. *Headless* preglednik je preglednik bez grafičkog korisničkog sučelja, a koristan je za automatiziranu interakciju s web preglednikom i testiranje, pošto se preko sučelja za naredbe može renderirati cijela stranica i tako oponašati čin stvarnog posjeta stranici.

Kao i sve druge *NodeJS* pakete, možemo ga jednostavno instalirati naredbom `npm install puppeteer`, a jedan od primjera korištenja nalazi se na slici 4.1.

```
1  const puppeteer = require('puppeteer')
2  const screenshot = 'jutarnji-screenshot.png'
3  try {
4    (async () => {
5      const browser = await puppeteer.launch()
6      const page = await browser.newPage()
7      await page.setViewport({ width: 1280, height: 800 })
8      await page.goto('https://www.jutarnji.hr', { waitUntil: 'networkidle2' })
9      await page.click('nav__item item-913')
10     await page.waitForSelector('.section-pretplata')
11     await page.screenshot({ path: screenshot })
12     await browser.close()
13     console.log('Slika opcija pretplate na jutarnji.hr: ' + screenshot)
14   })()
15 } catch (err) {
16   console.error(err)
17 }
```

Slika 4.1: Primjer korištenja *Puppeteer* knjižnice za slikanje opcija pretplata na *jutarnji.hr*

Slika 4.1 prikazuje kôd kojim možemo slikati stranicu pretplata na *jutarnji.hr*. *Puppeteer* knjižnicu uključimo u modul u prvoj liniji pozivanjem `require('puppeteer')`. Nakon toga, sa `puppeteer.launch()` stvaramo novi *headless Chrome* preglednik te sa `browser.newPage()` otvaramo novu stranicu. Pozivom funkcije `page.goto` navigiramo do naslovnice, zatim funkcijom `page.click` kliknemo na gumb u navigaciji koji nas vodi do stranice pretplata. Kada se učita čvor s klasom `.section-pretplata` čiju pojavu čekamo pomoću `page.waitForSelector`, naredbom `page.screenshot` možemo usli-

kati trenutno stanje stranice.

4.4. *MongoDB*

MongoDB [12] je dokumentno orijentirana, *NoSQL* baza koja je skalabilna i jednostavna za korištenje. *NoSQL* baze se od klasičnih relacijskih baza razlikuju po tome što ne koriste tablice, već neki drugi način spremanja podataka. Mogu biti dokumentno orijentirane, tipa ključ-vrijednost, baze širokog stupca, graf baze itd. Dokumentno orijentirane baze, kao što je *MongoDB*, koriste kolekcije kao spremišta za dokumente, odnosno JSON objekte s ključevima i vrijednostima.

Glavne komponente *MongoDB* arhitekture su: *_id* - obavezno polje koje se automatski generira, jedinstveno označava svaki zapis unutar dokumenata, *collection* - skup dokumenata, ekvivalent tablicama u relacijskim bazama podataka, *cursor* - pokazivač na rezultate *MongoDB* upita, *database* - sadrži skup kolekcija, *document* - jedan zapis u bazi s atributima u obliku ključ-vrijednost te *field* - jedno polje unutar dokumenta, par ključ-vrijednost.

Unutar *MongoDB* sučelja nativno se može koristiti *JavaScript*, pomoću čega je integracija u *NodeJS* projekt vrlo jednostavna. U ovom se radu koristi na dva mjesta: unutar *NodeJS* *scrapera*, koji u njega sprema članke s pronađenim imenovanim entitetima te unutar *Python* API-ja u sklopu web poslužitelja koji podatke dohvaća iz baze i obrađuje. Za *NodeJS* *scraper* korišten je pokretač naziva `node-mongodb-native`, pomoću kojeg se unutar projekta možemo spojiti na bazu, dok je za *Python* poslužitelj korišten pokretač `pymongo`.

4.5. *Python*

Python [13] je interpretirani programski jezik opće namjene. Njegove ugrađene strukture koje su prema korisnicima izložene na visokoj razini i dinamičko tipiziranje čine ga dobrim jezikom za brz razvoj i testiranje. Vrlo je dobar za početnike jer je njegov kôd lako čitljiv, logičan i razumljiv.

U projektu ovoga rada koristi se većinom prilikom izrade *NER* API-ja i web poslužitelja, uz pomoć *Flask* radnog okruženja. Osim toga, radi jednostavnosti i brzine koristi se prilikom bilo kojeg manipuliranja tekstualnim datotekama. Na primjer, korišten je prilikom filtriranja imena i prezimena iz velike *hrLex* datoteke, u skripti za ocjenu sustava te u raznim testovima.

4.6. *Flask*

Flask je web radno okruženje (engl. *web framework*) u obliku jednostavnog *Python* modula. Jednostavan je zato što ne koristi puno ovisnosti sam po sebi, tako da uključivanje u projekt ne zahtijeva puno pisanja novog kôda. Sve dodatne funkcionalnosti koje obično imaju kompliciranija web radna okruženja možemo odabrati i dodati naknadno. Zbog toga je dobar za male projekte kojima nisu potrebni složeni moduli i knjižnice. Može se koristiti za cijele web aplikacije. Koristi sustav predložaka *jinja2* pomoću kojeg možemo stvarati HTML stranice renderane na strani poslužitelja.

U ovom radu koristimo ga samo kao okruženje za lakši razvoj API-ja. Točnije, pomoću njega razvijeni su NER API i web poslužitelj u obliku API-ja koji će koristiti web stranica. Definiranje API krajnjih točaka u *Flasku* omogućeno je jednostavnim anotacijama koje služe kao dekoratori za određene *Python* funkcije. Primjer definiranja jedne krajnje točke nad *Python* funkcijom nalazi se na slici 4.2.

```
18 @app.route('/posts', methods=['GET'])
19 def get_recent_by_group():
20     count = request.args.get('count')
21     start = request.args.get('start')
22     group_name = request.args.get('group')
```

Slika 4.2: Primjer definicije krajnje točke API-ja u *Flask* radnom okruženju

Na prvoj liniji kôda sa slike 4.2 nalazi se anotacija kojom definiramo da se na poziv krajnje točke `/posts` metodom `GET` poziva funkcija `get_recent_by_group`. Unutar funkcije nam je dostupan i `request` objekt koji sadrži sve informacije o zahtjevu koji je poslan na krajnju točku. U ovom slučaju vidimo da funkcija očekuje *query* parametre `count`, `start` i `group`. Osim *query* parametara, u anotaciju možemo dodati i *url* parametar `title_hash` tako da unutar anotacije dodamo `/post/<title_hash>`. Tako će `title_hash` biti dostupan kao argument funkcije nad kojom je definirana ruta.

```
13 @app.route('/post/<title_hash>', methods=['GET'])
14 def get_post(title_hash):
15     db_post = posts.find_one({ 'titleHash': title_hash })
16     return jsonify(to_post(db_post))
```

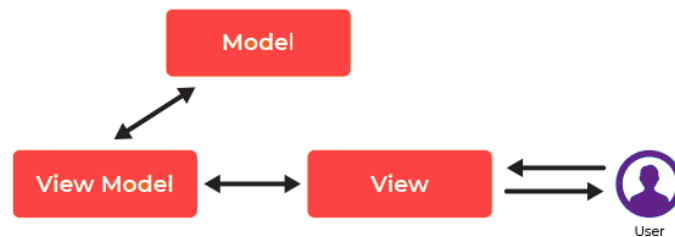
Slika 4.3: Funkcija za dohvaćanje jednog posta

Slika 4.3 prikazuje jednostavnost korištenja *Flask* okruženja. Samo pomoću ove četiri linije možemo dohvatiti članak sa zadanim sažetkom. Prvo pomoću sučelja po-

kretača *pymongo* pronađemo članak sa zadanim sažetkom. Nakon toga taj zapis pretvorimo u čitljiviji model koji se lako pretvara u JSON te ga na kraju pretvorimo u JSON i takvog pošaljemo u odgovoru zahtjeva.

4.7. *Vue.js*

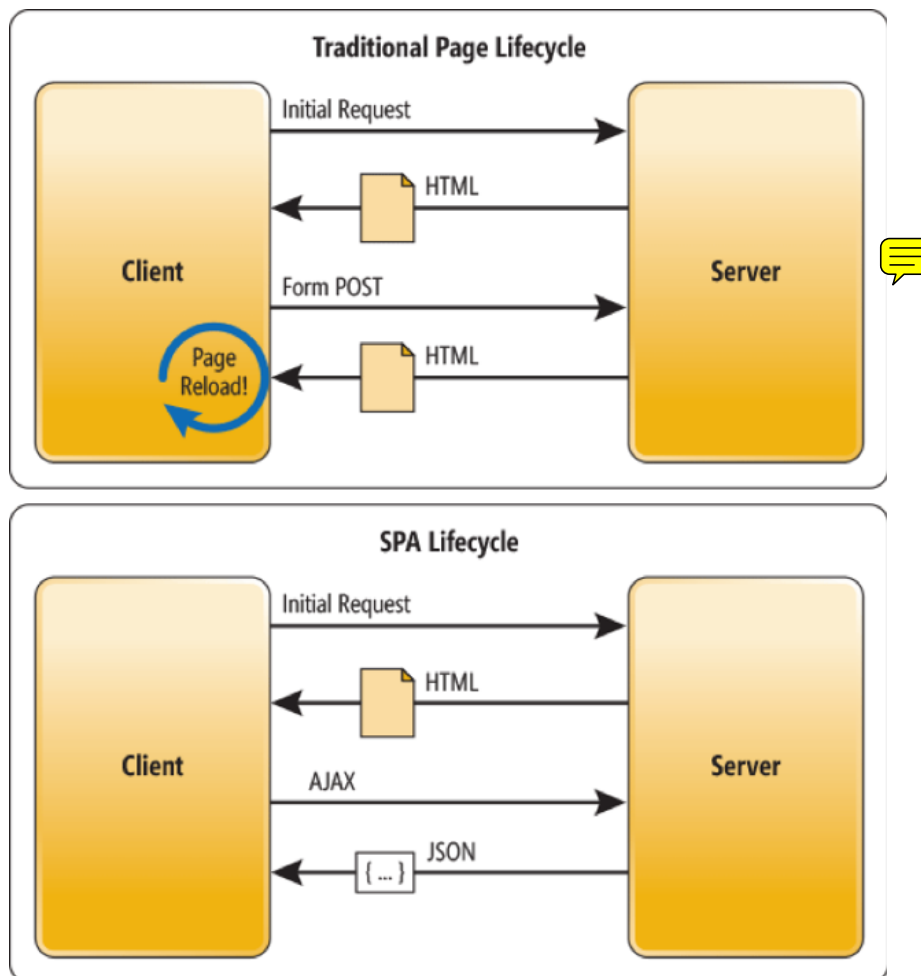
Vue.js je progresivno *JavaScript* radno okruženje otvorenog kôda. Bazirano je na obrascu *Model–View–ViewModel* (MVVM), kojim se odvaja razvoj grafičkog sučelja.



Slika 4.4: Arhitektura MVVM obrasca

Slika 4.4 pokazuje arhitekturu *model-view-viewmodel* obrasca u kojem *model* pohranjuje podatke i povezanu logiku, *view* označava komponente korisničkog sučelja, dok *viewmodel* pruža funkcije, naredbe i metode kojim se kontrolira stanje *view* komponente.

Pomoću *Vue* okruženja razvijaju se korisnička sučelja i *single-page* aplikacije (SPA). *Single-page* je pristup koji se koristi za aplikacije koje se, osim početnog učitavanja, u potpunosti izvršavaju u pregledniku. Interakcija korisnika nikada ne uzrokuje ponovno učitavanje, što znatno poboljšava korisničko iskustvo. Svaka potreba za novim podacima ostvaruje se AJAX (engl. *Asynchronous JavaScript And XML*) zahtjevima te se u pregledniku tada samo ponovno iscrta dio aplikacije koji se dohvaćanjem novih podataka promijenio.



Slika 4.5: Razlika između tradicionalne web stranice i SPA web stranice, izvor: <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>

Na slici 4.5 vidimo razliku u životnom ciklusu tradicionalne web stranice i *single-page* aplikacije. Vidimo da će na tradicionalnoj stranici svaka korisnička akcija uzrokovati dohvaćanje nove stranice u HTML obliku i time uzrokovati ponovno učitavanje. S druge strane, SPA dohvaća HTML stranicu samo na početku, dok su svaki naredni pozivi AJAX zahtjevi kojima se dohvaćaju sirovi podatci u obliku JSON objekata.

Vue aplikaciju vrlo je jednostavno napraviti, sljedeći odsječci kôda to demonstriraju:

```
<div id="app">
  {{ message }}
</div>
```

U ovom odsječku nalazi se jednostavna HTML datoteka. Sve što je potrebno je jedan vršni čvor sa `id` atributom željenog imena.

```
var app = new Vue({  
  el: '#app',  
  data: {  
    message: 'Ovo je moja prva Vue aplikacija'  
  }  
})
```

U ovom se odsječku nalazi *JavaScript* kôd. Potrebno je navesti na kojem će se DOM čvoru iscrtati aplikacija te koji su podaci potrebni predlošku.

Vue je u pozadini omogućio da je poruka `message` koja se nalazi u `data` objektu sada automatski reaktivna. Dinamička promjena *message* polja automatski će se propagirati do pozadinskih mehanizama koji će ponovno iscrtati samo taj dio stranice gdje je poruka.



5. Ocjena sustava



Komponente sustava po kojima možemo zaključiti koliko sustav dobro radi su *scraper* i NER API. Za *scraper* možemo proučiti koliko dobro izvlači tekst, da li pokupi sav bitan tekst u članku, koliko se neželjenog teksta nalazi u njemu itd. NER API možemo ocijeniti po tome koliko dobro pronalazi imena za neki zadani skup članaka. Testiranje uspješnosti NER API-ja smo obavili programski, uz pomoć anotiranih članaka, dok se ocjena *scraper*a odnosi na česte, teško ispravljive greške nađene prilikom razvoja.

5.1. Ocjena *scraper* komponente

Komponenta za izvlačenje informacija o člancima nije testirana programski, već je ručno pregledano 20 članaka - po četiri sa svakog portala koje koristimo u ovoj pokusnoj (engl. *proof of concept*) aplikaciji. Uspješnost je subjektivno određena pregledom je li sav relevantan tekst i ostali sadržaj članka *scraper* komponenta pokupila. Ako je, proces je jednostavno označen uspješnim a ako nije u potpunosti, neuspješnim. Na temelju tih **20** pregledanih članaka, **2** nisu bila uspješna, što daje ukupnu uspješnost od **90%**.

Razlog neuspjeha u oba neuspjela slučaja leži u prirodi procesa *scrapinga* i činjenici da je svaki portal malo različit. Prethodno je opisano da *scraper* generalno traži sve paragrafe unutar čvora u kojem se nalazi sadržaj članka. U jednom slučaju razlog neuspjeha bio je to što se dio teksta članka nije nalazio unutar paragrafa, već u vršnom čvoru sadržaja, a drugi je neuspjeli slučaj pokupio previše neželjenog teksta zbog kojeg bi NER API-ju bio otežan proces pronalaska imena.

```
top: 45px; .../div>
<div class="article__body--main">
  <div class="article__body--main_content" id="js_content"
    data-section-slug="vijesti" data-subsection-slug="svijet"
    data-article-id="1503193">
    <p>...</p>
    <div class="twitter-tweet twitter-tweet-rendered"
      style="display: flex; max-width: 550px; width: 100%; mar
        gin-top: 10px; margin-bottom: 10px;">...</div> flex
    <p>...</p>
    <p>...</p>
    <p>
      <div class="fb-post fb_iframe_widget" data-href="http
        s://www.facebook.com/lovci bourek/posts/406314673042161
          8" data-show-text="true" fb-xfbml-state="rendered" fb-
          iframe-plugin-query="app_id=117486881606473&container_w
            idth=507&href=https%3A%2F%2Fwww.facebook.com%2Flovci bou
              rek%2Fposts%2F4063146730421618&locale=hr_HR&sdk=joey&sh
                ow_text=true">...</div>
      </p>
    <p>...</p>
```

Slika 5.1: Primjer neželjenog sadržaja u tekstu članka

Na slici 5.1 vidimo cijelo stablo unutar paragrafa `<p>`, koje u sebi ima čak i *iframe* čvor. Zbog takvog besmislenog teksta NER API neće moći uspješno uhvatiti imena sa smislenog dijela koji se, umjesto izravno u paragrafu, nalazi na listu stabla unutar njega.

5.2. Ocjena komponente NER API

NER API najbolje je ocijeniti po tome koliko je imenovanih entiteta pronašao u člancima. To možemo napraviti ako imamo popis imenovanih entiteta koji se stvarno nalaze u člancima. Kao pripremu za testiranje NER API-ja označili smo imenovane entitete za oko 80ak članaka s portala koje koristimo u pokusnom rješenju. Prilikom postupka označavanja entiteta pokušali smo označavati imenovane entitete tako da označavamo puna imena i prezimena osoba koje se pojavljuju u tekstu bez dupliciranja. Ako neko u cijelom tekstu ime nema popratno prezime, zapišemo samo to ime. Isto vrijedi i za prezimena.

Za potrebe ovog testiranja izrađena je skripta koja za zadani popis poveznica na članke obavlja proces izvlačenja naslova i teksta članka te šalje upite NER API-ju za pronalazak imenovanih entiteta. Imenovani entiteti koji se pojavljuju u člancima zapisuju se u posebnu datoteku tako da sadrže dva stupca: u prvom se nalaze poveznice na članke, dok se u drugom nalaze svi entiteti pronađeni u tekstu tog članka. Zbog toga kako NER API radi, u tom ispisu pronađenih entiteta nalazit će se više istih imena u različitim oblicima te neke potencijalno besmislene stvari koje nam za potrebe ovog sustava ne predstavljaju problem.

```

https://www.index.hr/vijesti/clanak/... {'Ivor V.', 'Nino Čengiđ', 'Ivan G.', 'Dominik B.', 'Asja G.',
https://www.index.hr/vijesti/clanak/... {'Milanović', 'Puljak', 'Puljak Od Mihanović', 'Milanovića', 'L
https://www.index.hr/vijesti/clanak/... {'Ivor V.', 'Nino Čengiđ', 'Ivan G.', 'Dominik B.', 'Asja G.',
https://www.index.hr/vijesti/clanak/... {'Dean Prodati', 'Valtera Flege', 'Milana Bandića', 'Kajin', 'I
https://www.index.hr/vijesti/clanak/... {'Kolosti', 'Elvira Mešanović', 'Dražiti Lalić', 'Lukašenkov',
https://www.index.hr/vijesti/clanak/... {'Tesle', 'Igor Kolovrat', 'Kolovrat', 'Teslo', 'Tesla'}
https://www.index.hr/vijesti/clanak/... {}
https://www.index.hr/vijesti/clanak/... {'Bandića', 'Tomislav Tomašević', 'Jelen Pavičić Vukičević', 'V
https://www.index.hr/vijesti/clanak/... {'Tomaševića', 'Škoru', 'Tomašević', 'Tomislav Tomašević', 'Ško
https://www.index.hr/vijesti/clanak/... {}
https://www.index.hr/vijesti/clanak/... {'Grubeša', 'Saudić', 'Bisera Turković', 'Bezrazložan', 'Saud
https://www.index.hr/vijesti/clanak/... {'Janez Janša', 'Janeza Janše', 'Bojana Beović', 'Bojan Beović'
https://www.index.hr/vijesti/clanak/... {}

```

Slika 5.2: Dio datoteke u kojoj se nalazi ispis pronađenih imenovanih entiteta

Na slici 5.2 nalazi se dio ispisa pronađenih entiteta dobivenog izvođenjem skripte. U drugom stupcu nalaze se imena unutar podatkovne strukture skup (engl. *set*) - ako NER API nije pronašao ništa, taj je skup prazan. Korištenjem strukture skup osigurali smo da se u ispisu ne pojavljuju nepotrebni duplikati.

Kada imamo ispis sustava za zadan popis poveznica na članke te datoteku s ručno označenim entitetima u tekstovima članaka, možemo kvalitetno ocijeniti preciznost NER API-ja. To ćemo napraviti pomoću nove skripte koja čita te dvije datoteke te ih uspoređuje tako da ide kroz ručno označene entitete svakog članka te provjeri nalaze li se ti entiteti u skupovima entiteta koje je NER API pronašao.

```

Poveznica: https://rep.hr/vijesti/tvrtke-i-trzista/span-na-korak-do-izlaska-na-zagrebacku-burzu/7693/
P/O:      3/3 (1.00)
Pronađeni: ['Dujmović', 'Nikola Dujmović', 'Ante Mandić', 'Nikole Dujmovića', 'Aron Paulić', 'Span', 'Spana']
Označeni:  ['Ante Mandić', 'Aron Paulić', 'Nikola Dujmović']

```

Slika 5.3: Ispis za jednu poveznicu skripte za procjenu uspješnosti NER API-ja

U ispisu za poveznicu na slici 5.3 vidimo da je uspješnost **1**, odnosno **100%**. To je tako zato što se svaki od entiteta u popisu označenih nalazi i u popisu pronađenih entiteta.

```

Ukupno:
P/O:      344/418 (0.82)

```

Slika 5.4: Ukupna uspješnost NER API-ja



Na slici 5.4 nalazi se ukupan postotak pronađenih u odnosu na označene entitete. U **80** članaka, NER je pronašao **344**, dok je označeno **418** entiteta, što daje ukupno **82%** točnosti komponenti sustava za pronalazak imenovanih entiteta.

6. Moguća unaprjeđenja i smjernice za budući razvoj

Ukoliko bi se ovaj sustav razvijao na većoj skali, bilo bi dobro napraviti neke preinake koje bi učinile sustav robusnijim i boljim. Cijeli sustav mogli bismo bolje skalirati postavljanjem komponenata u neki od pružatelja *cloud* usluga kao što su *Amazon Web Services* (AWS), *Microsoft Azure*, *Google Cloud Provider*, *Oracle* itd. U njima bi pomoću raznih tehnologija kao što su *Content Delivery Network* (CDN), *serverless computing* i sličnih, mogli kreirati sustav koji je otporan na razne vrste grešaka ili napada. Potencijalna unaprjeđenja procesa komponenti u ovom radu objašnjena su u nastavku.

6.1. *Scrape* proces

Kako bi se poboljšao proces *scrapinga*, u njega bi trebalo dodati puno heuristika kojima bi se smanjila mogućnost dohvaćanja neželjenih dijelova teksta ili propust dohvaćanja željenih. Unos novog portala koji će se pretraživati trebao bi biti vrlo pojednostavljen. U idealnom slučaju, jednim generaliziranim sustavom mogao bi se pretraživati bilo koji portal. Glavna stvar prilikom izrade takvog sustava bila bi pronaći sve tipove struktura portala te za svaki tip napraviti *scraper*. To je za jednu razinu apstrakcije iznad ovog rada, koji za samo jedan tip strukture pokušava pronaći zajednička obilježja portala.

6.2. NER API

U sklopu nekog velikog sustava mogao bi se trenirati vlastiti NER procesor nad podacima o imenovanim entitetima sakupljenim baš s internetskih portala. Tako će za ovu svrhu NER u budućnosti davati najbolje rezultate.

Osim korištenja samog NER-a za pronalazak željenih imena osoba, mogli bismo koristiti neke druge načine, npr. prefiksno pretraživanje teksta.

NER često pronađe sva imena koja se nalaze u nekom članku, ali se ona u članku ne pojavljuju u nominativu. U nekim slučajevima naš sustav ne zaključi koji je pravilan nominativ tog imena, tako da dođemo do situacije gdje u bazi imamo ime u nekoj drugoj deklinaciji te ime koje je, pri pokušaju pretvorbe u nominativ, pretvoreno u nešto besmisleno. Sada, iako imamo dva oblika istog imena u bazi, ono neće biti pronađeno jer će se prilikom pretrage zapisi u bazi uspoređivati s nominativima imena. No, ako gledamo korijen imena, u kojoj god da se deklinaciji ime pojavilo, on je vjerojatno isti. Tako bismo mogli na neki način provjeravati sličnost dva teksta te ako je ta sličnost velika, s određenim postotkom vjerojatnosti zaključiti nalazi li se ime u tekstu ili ne. Jedna mogućnost ostvarivanja toga bila bi *Jaro-Winkler* sličnost. To je metoda za usporedbu sličnosti dva teksta takva da će davati veću ocjenu sličnosti tekstovima koji imaju zajednički prefiks. Formula za *Jaro-Winkler* sličnost je:

$$S_w = S_j + P * L * (1 - S_j)$$

gdje je S_w *Jaro-Winkler* sličnost, S_j *Jaro* sličnost, P faktor skaliranja i L maksimalna duljina prefiksa za mjerenje sličnosti. Pri tome s formula za *Jaro* sličnost nalazi na slici 6.1.

$$Jaro\ similarity = \begin{cases} 0, & \text{if } m=0 \\ \frac{1}{3} \left(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m} \right), & \text{for } m \neq 0 \end{cases}$$

Slika 6.1: Formula za *Jaro* sličnost

Na formuli za *Jaro* sličnost sa slike 6.1, m označava broj jednakih znakova, t je polovica broja transpozicija, a $s1$ i $s2$ su duljine tekstova.

Osim pronalaska nominativa imena iz entiteta pronađenih u člancima, mogli bismo raditi i obrnuto, tako da zadana imena koje tražimo, a imamo ih u nominativu, pretvorimo u sve druge deklinacije. Kako bi to ostvarili također bismo trebali koristiti *hrLex* datoteku tako da za sva tražena imena pronađemo nominative u drugom stupcu ispisa *hrLex* te pomoću toga, temeljem frekvencije korištenja navedenih imena ili na neki drugi način, generiramo sve ostale deklinacije.

6.3. Web stranica

Na web stranici bi se mogle dodati razne mogućnosti. Osim filtriranja odabirom grupe osoba, filtriranje bi moglo biti naprednije ako dodamo mogućnost filtriranja po ostalim imenovanim entitetima: organizacijama, lokacijama itd. Na stranici bi mogli implementirati i sustav korisničkih računa s kojim bi svaki korisnik mogao imati individualne, personalizirane filtere i tako dobivati poveznice na članke koji ga zanimaju. Tim korisnicima bi onda moglo biti omogućeno i dodavanje novih grupa ljudi ili čak dinamično dodavanje novih vrsta filtera.

7. Legalna napomena

Kako se prilikom ovog rada prikupljaju javno dostupni podaci o stvarnim osobama, u pitanje se dovodi legalna priroda takvog sustava. Prema općoj uredbi o zaštiti podataka (engl. *General Data Protection Regulation, GDPR*) Europske Unije, tko god prikuplja javno dostupne podatke u svrhu daljnjeg procesiranja, dužan je izravno obavijestiti osobu čiji se podaci prikupljaju. Čak i u slučaju da je čin sakupljanja podataka o osobama objavljen negdje na web stranici na kojoj smo prijavljeni kao korisnik, uredba o zaštiti podataka svejedno se krši ako kao korisnik nismo na neki način obaviješteni izravno ili ako je, umjesto svih korisnika, obaviješten samo određeni podskup.

Kako je implementacija sustava iz ovog rada samo pokusna i koristi nekoliko osoba iz hrvatske javnosti za potrebe testiranja, ne krši opću uredbu o zaštiti podataka.

8. Zaključak

Pretraživanje podataka pomoću sadržaja web stranica, koje su namijenjene za krajnje, ljudske korisnike, pokazao se kao mukotrpan posao. To pospješuje činjenica da se stranice u današnje vrijeme vrlo često mijenjaju te time proces koji smo prolazili prilikom izvlačenja podataka *web scraperom* moramo prolaziti ponovno. Tehnologija brzo napreduje, a programeri koji ih koriste opravdano žele svakim izlaskom nove tehnologije unaprijediti web stranice koje razvijaju kako bi se povećala sigurnost i otpornost na budućnost. Osim napretka tehnologije, svaka stranica je i sama po sebi različita, svaki novinski portal ima drukčiju strukturu i svaka je napravljena u tehnologiji koja ima svoje specifičnosti. Unatoč tim preprekama, moguće je pronaći neke zajedničke elemente skupa stranica koje iz kojih želimo izvlačiti podatke. U slučaju portala u ovom radu, taj zajednički element je činjenica da na portalima sve poveznice na članke započinju istim znakovima i imaju isti format te činjenica da na stranicama članaka, za izvlačenje naslova i sadržaja članka možemo jednostavno samo koristiti XPath upite skrojene specifično za strukturu tog portala.

Sustav za pronalaženje imenovanih entiteta, NER, pokazao se kao vrlo dobar način za ubrzanje pretrage teksta u slučaju kada tražimo specifične entitete koje tekst opisuje. Osim za traženje osoba, za što se koristi u ovom radu, bio bi dobar za traženje organizacija, lokacija i još mnogo toga. Bez njega, pretraživanje bismo morali raditi ili primitivnijim načinima pretrage, koji bi bili vrlo spori, ili nekim načinima čija bi implementacija mogla biti rad za sebe.



LITERATURA

- [1] W3C. Xpath, 2016. URL <https://www.w3.org/TR/1999/REC-xpath-19991116/>.
- [2] Nikola Ljubešić. The CLASSLA-StanfordNLP model for named entity recognition of non-standard croatian 1.0, 2020. URL <http://hdl.handle.net/11356/1340>. Slovenian language resource repository CLARIN.SI.
- [3] Nikola Ljubešić, Željko Agić, Filip Klubička, Vuk Batanović, and Tomaž Erjavec. Training corpus hr500k 1.0, 2018. URL <http://hdl.handle.net/11356/1183>. Slovenian language resource repository CLARIN.SI.
- [4] Nikola Ljubešić, Tomaž Erjavec, Vuk Batanović, Maja Miličević, and Tanja Samardžić. Croatian twitter training corpus ReLDI-NormTagNER-hr 2.1, 2019. URL <http://hdl.handle.net/11356/1241>. Slovenian language resource repository CLARIN.SI.
- [5] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python natural language processing toolkit for many human languages, 2020.
- [6] Lance A. Ramshaw and Mitchell P. Marcus. Text chunking using transformation-based learning, 1995.
- [7] Leksička flektivna baza podataka hrvatskih imena i prezimena, 2003.
- [8] Nikola Ljubešić. Inflectional lexicon hrLex 1.3, 2019. URL <http://hdl.handle.net/11356/1232>. Slovenian language resource repository CLARIN.SI.
- [9] ECMA Ecma. 262: EcmaScript language specification, 1999.
- [10] Node.js, 2021. URL <https://nodejs.org/en/>.

- [11] Puppeteer, 2021. URL <https://pptr.dev/>.
- [12] Mongodb, 2021. URL <https://www.mongodb.com/>.
- [13] Python, Jun 2021. URL <https://www.python.org/>.

Informacijski sustav za prikupljanje javno dostupnih podataka za zadani skup osoba

Sažetak

Informacijski sustav kojim se prikupljaju javno dostupni podaci o osobama koje se spominju u hrvatskim portalima u svrhu informiranja o postignućima i događajima vezanim za zadani skup osoba. Prikupljeni podaci prikazuju se na internetskoj stranici na kojoj je omogućeno filtriranje vremenski sortiranih članaka po određenim grupama ljudi. Prikupljanje članaka s hrvatskih internetskih portala ostvareno je tehnologijama za ekstrakciju podataka s HTML stranica, dok je pronalaženje imena ostvareno tehnologijom za prepoznavanje imenovanih entiteta u tekstovima.

Ključne riječi: informacijski sustav, portal, osoba, članak, ekstrakcija, podaci, grupa

Information system for gathering publicly available data for a selected group of people

Abstract

Information system for gathering publicly available data about people mentioned in croatian news sites with the goal of informing about accomplishments and events regarding a given set of people. Gathered data is displayed in a web site where you can filter sorted posts by a specific group of people. Gathering posts from croatian news sites is implemented as a scraper, while people's names are found using a named entity recognition system.

Keywords: information system, news site, person, post, scraping, data, group, NER, named entity recognition