Programsko rješenje web sustava za prilagodljivi višekriterijski prikaz vijesti iz različitih izvora

Maras, Filip

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:508678

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij

PROGRAMSKO RJEŠENJE WEB SUSTAVA ZA PRILAGODLJIVI VIŠEKRITERIJSKI PRIKAZ VIJESTI IZ RAZLIČITIH IZVORA

Diplomski rad

Filip Maras

Osijek, 2020.

SADRŽAJ

1. UVOD 1
1.1. Zadatak diplomskog rada 2
2. PREGLED TRENUTNOG STANJA U PODRUČJU SUSTAVA ZA PRILAGODLJIVI PRIKAZ VIJESTI 3
2.1. Prilagodljivi prikaz vijesti iz različitih izvora 3
2.2. Prikaz postojećih rješenja 3
3. ZAHTJEVI, MODEL I IDEJNO RJEŠENJE 7
3.1. Zahtjevi i model aplikacije 7
3.1.1 Funkcionalni i nefunkcionalni zahtjevi 7
3.1.2 Model aplikacije 8
3.2. Višekriterijski prikaz vijesti 8
3.3. Idejno rješenje sustava na strani poslužitelja 9
3.4. Idejno rješenje sustava na strani korisnika 10
4. PROGRAMSKO RJEŠENJE WEB SUSTAVA ZA PRILAGODLJIVI PRIKAZ VIJESTI 12
4.1. Korišteni jezici, alati i okoline za implementaciju web sustava za prilagodljivi prikaz vijesti 12
4.1.1. Visual Studio Code 12
4.1.2 Python 12
4.1.3 Flask 12
4.1.4 HTML i CSS 13
4.2. Programsko rješenje na strani poslužitelja 15
4.2.1 Dohvaćanje podataka s različitih izvora 15
4.2.2. Funkcija get feed from multiple sources 16
4.2.3. Funkcija *get image* 16
4.2.4. Funkcija get description 17
4.2.5. Funkcija get urls from filter 18
4.2.6. Ruta *home* 19
4.2.7. Ruta *content* 19
4.2.8. Ruta *search* 20
4.2.9. Rute *all* i sport 20
4.3. Programsko rješenje na strani korisnika 22
4.3.1. Početna stranica 22
4.3.2. Stranica za prikaz vijesti 23
4.3.3. CSS stiliziranje 24
4.4. Analiza i testiranje programskog rješenja 26
4.4.1. Funkcionalno testiranje 26
4.4.2. Rezultati provedbe funkcionalnog testiranja 27

4.4.3. Nefunkcionalno testiranje 29
4.4.4. Rezultati nefunkcionalnog testiranja 30
5. NAČIN KORIŠTENJA SUSTAVA I ANALIZA REZULTATA 35
5.1. Opis načina rada sustava 35
5.2. Analiza rezultata razvoja i testiranja programske podrške 37
6. ZAKLJUČAK 38
LITERATURA 39
SAŽETAK 42
ABSTRACT 43
ŽIVOTOPIS 44
PRILOZI 45

1. UVOD

Na početku ovog diplomskog rada dan je pregled rješenja web sustava za prilagodljivi i višekriterijski prikaz vijesti iz različitih izvora. Prikazana su postojeća rješenja i njihove karakteristike, prednosti i nedostaci. Svrha ovakvih rješenja je olakšati korisniku pregled vijesti, odnosno uklanjanje potrebe da korisnik ide na više različitih vjesnika, jer se sve vijesti mogu vidjeti na jednom mjestu. Cilj ovog diplomskog rada je stvaranje rješenja za prikaz vijesti koje će korisniku omogućiti pregledavanje vijesti iz različitih izvora uz mogućnost odabira izvora, te pretraživanje vijesti iz ponuđenih vjesnika prema više kriterija.

Web sustav prilagodljivi višekriterijski prikaz vijesti iz različitih izvora treba imati mogućnost pretraživanja vijesti što korisniku smanjuje vrijeme potrebno za pronalazak željene vijesti. Taj sustav, kao web rješenje, podijeljen je na sustav na poslužiteljskoj strani i sustav na korisničkoj strani. Sustav na poslužiteljskoj strani ostvaren je pomoću tehnologija Python i Flask, te radi dohvaćanje vijesti s odabranih vjesnika i njihovo parsiranje kako bi se omogućio optimalan prikaz na korisničkoj strani. Također, ovisno o korisnikovim željama, sustav vrši parsiranje i pretraživanje. Činjenica da korisnik sam bira što želi čitati, tj. korisnik sam kreira svoj prikaz vlastitim odabirom vjesnika ili pretraživanjem, čini sustav prilagodljivim i višekriterijskim. Sustav na korisničkoj strani sastoji se od dvije stranice koje će biti ostvarene uz korištenje tehnologija HTML i CSS. Početna stranica je stranica koja uz naslov ima i formu za odabir vijesti. Druga stranica je stranica prikaz odabranog sadržaja i sastoji se od naslova, poveznice koja vodi na početku stranicu, te od prikazanog sadržaja. Nakon završetka izrade cjelokupnog sustava, sustav je testiran u sklopu drugog diplomskog rada. Testiranjem su dobivena izvješća o potencijalnim funkcionalnim greškama, te izvješća o testu opterećenja sustava. Te informacije korištene su unaprjeđenje ostvarenog sustava ispravljanjem grešaka i nedostataka.

U drugom poglavlju dan je pregled postojećih rješenja uz primjere. Treće poglavlje obuhvaća zahtjeve za sustav, model sustava i idejno rješenje sustava. Nakon definiranih zahtjeva i razvijenog idejnog rješenja, u četvrtom poglavlju prikazano je programsko rješenje. Četvrto poglavlje također obuhvaća postupak testiranje, te daje uvid u moguće izmjene i popravke sustava u ovisnosti o rezultatima testiranja. U petom poglavlju dane su korisničke upute i načinjena je analiza cjelokupnog sustava.

1.1. Zadatak diplomskog rada

U diplomskom radu treba opisati postojeća slična rješenja i zahtjeve i izazove u dohvaćanju i prilagodljivom prikazu vijesti iz različitih on-line izvora, te razraditi model i arhitekturu web sustava koji bi na strani poslužitelja omogućio pretraživanje, sortiranje, filtriranje, parsiranje i višekriterijsku prilagodbu vijesti, a na korisničkom sučelju odgovarajući oblik prikaza vijesti. Nadalje, treba opisati programske tehnologije, jezike i okvire koje će se koristiti za ostvarenje sustava. U praktičnom dijelu rada treba programski ostvariti opisani web sustav na strani poslužitelja i klijenta, te ga unaprijediti uzimajući u obzir rezultate testiranja.

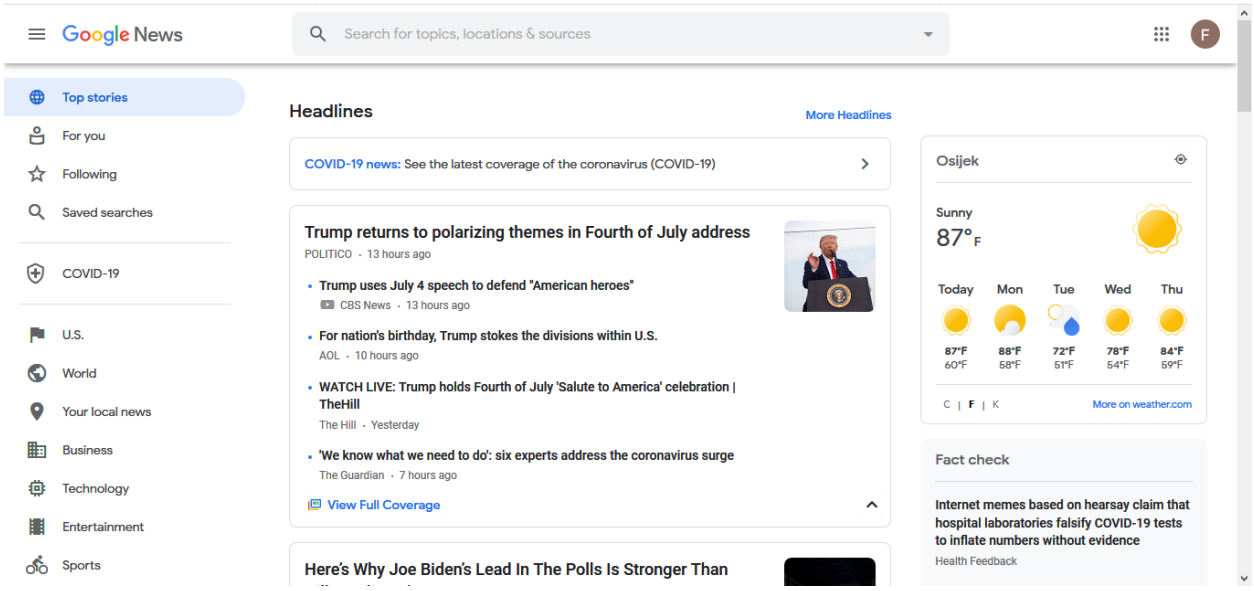
2. PREGLED TRENUTNOG STANJA U PODRUČJU SUSTAVA ZA PRILAGODLJIVI PRIKAZ VIJESTI

2.1. Prilagodljivi prikaz vijesti iz različitih izvora

U današnjem svijetu okruženi smo velikom količinom raznih informacija. Te informacije je potrebno nekako filtrirati, sortirati i kategorizirati kako bi pronalaženje potrebnih informacija bilo što lakše i bezbolnije. Ono što tržište zahtjeva je upravo alat koji će imati mogućnost dohvaćanja raznih vijesti s različitih izvora te njihovo sortiranje, kategoriziranje i filtriranje [1]. Takav alat će korisniku dati mogućnost da na jednom mjestu vidi sve vijesti s više različitih portala, da pretražuje te vijesti ili da jednostavno odabere kategoriju koja ga najviše zanima. Iako se ovaj rad bavi prikazom vijesti, ta rješenja se često koriste i u druge svrhe. Primjerice, developeri često koriste takva rješenja za praćenje noviteta u svom području što uvelike olakšava posao i smanjuje vrijeme potrebno za pronalazak informacija [2].

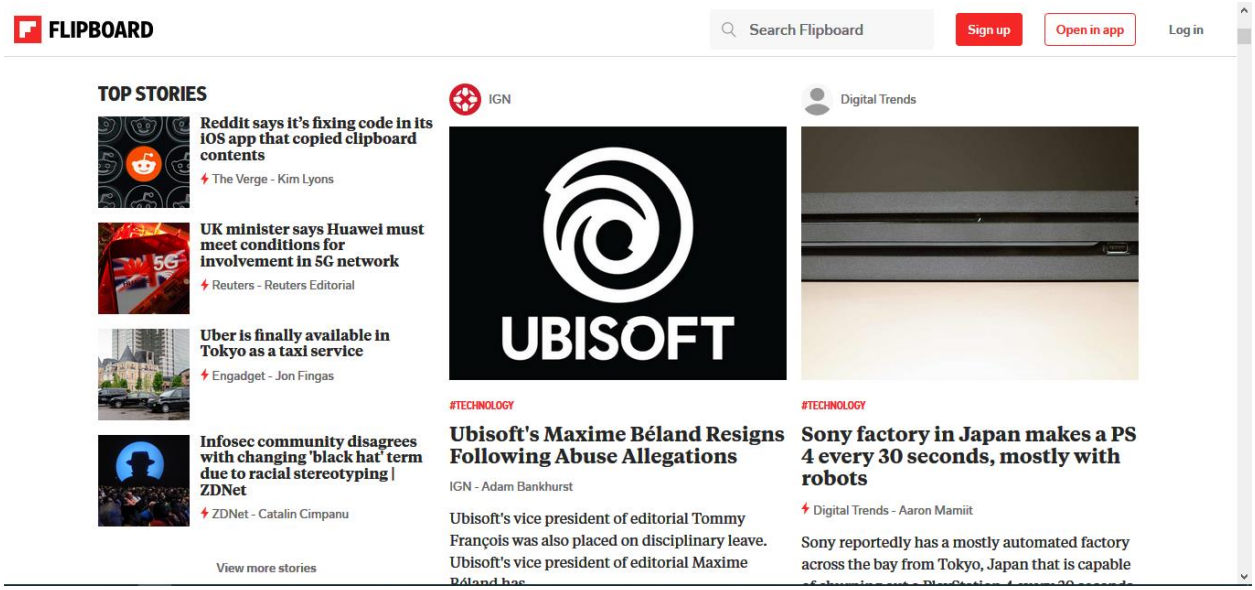
2.2. Prikaz postojećih rješenja

Danas već postoje mnoga rješenja koja nude mogućnost prilagođavanja sadržaja koji se gleda. Neka od rješenja su: *Google News*, *Flipboard* i *News Clusters* [3]. Prvim pogledom na *Google News* lako se zamijeti koliko su ponuđene vijesti zanimljive. To je zato što *Google News* personalizira vijesti svakom pojedincu koristeći sve podatke o osobi koje ima na raspolaganju. Ti podaci su primjerice povijest *Google* pretraživanja, povijest gledanja sadržaja na *YouTube*-u i mnogi drugi koji su vezani za korišteni *Google* račun. Na slici 2.1 vidi se *web* prikaz *Google News*-a. Zamjetno je da postoje različite kategorije vijesti koje se mogu birati kao što su vijesti iz svijeta tehnologije, iz svijeta sporta, lokalne vijesti koje se biraju ovisno o geografskoj lokaciji korisnika i drugo. Također se mogu gledati vijesti koje je *Google* algoritam odabrao za nas klikom na gumb *„For you“*. Ono što se odmah vidi jest da *Google News* naglasak daje tekstu, a ne slikama. *Google News* također ima i rješenje za mobilne uređaje koje je moguće skinuti putem trgovine *Google play* za Android ili *App store* za iPhone uređaje.



Slika 2.1. Google News

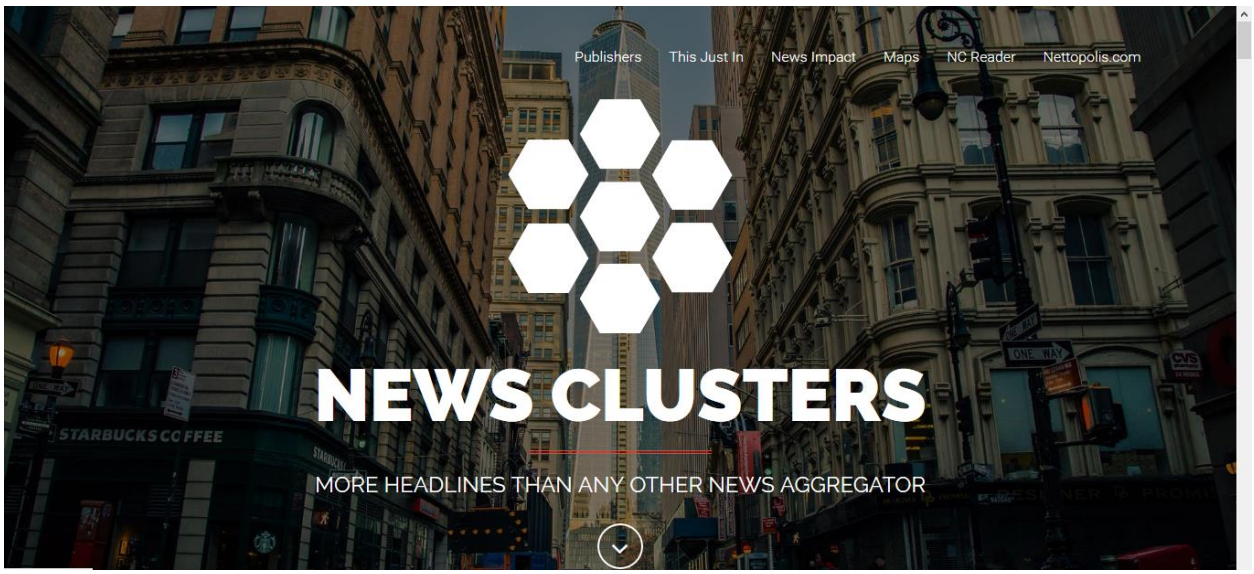
Drugo spomenuto rješenje je *Flipboard* koje se može vidjeti na slici 2.2. Za razliku od *Google News*, *Flipboard* naglašava slike. *Flipboard* je vrlo prilagodljiv, pa tako korisnik može sam birati koje će izdavače pratiti ili koje najdraže teme želi čitati. Također se članci mogu označiti sa „sviđa mi se“ ili „ne sviđa mi se“ te će *Flipboard* vremenom sam naučiti koja tematika zanima korisnika te mu pomoću tih informacija prilagoditi prikazani sadržaj. *Flipboard* kao i *Google News* se mogu koristiti kao mobilno i web rješenje.



Slika 2.2. Flipboard

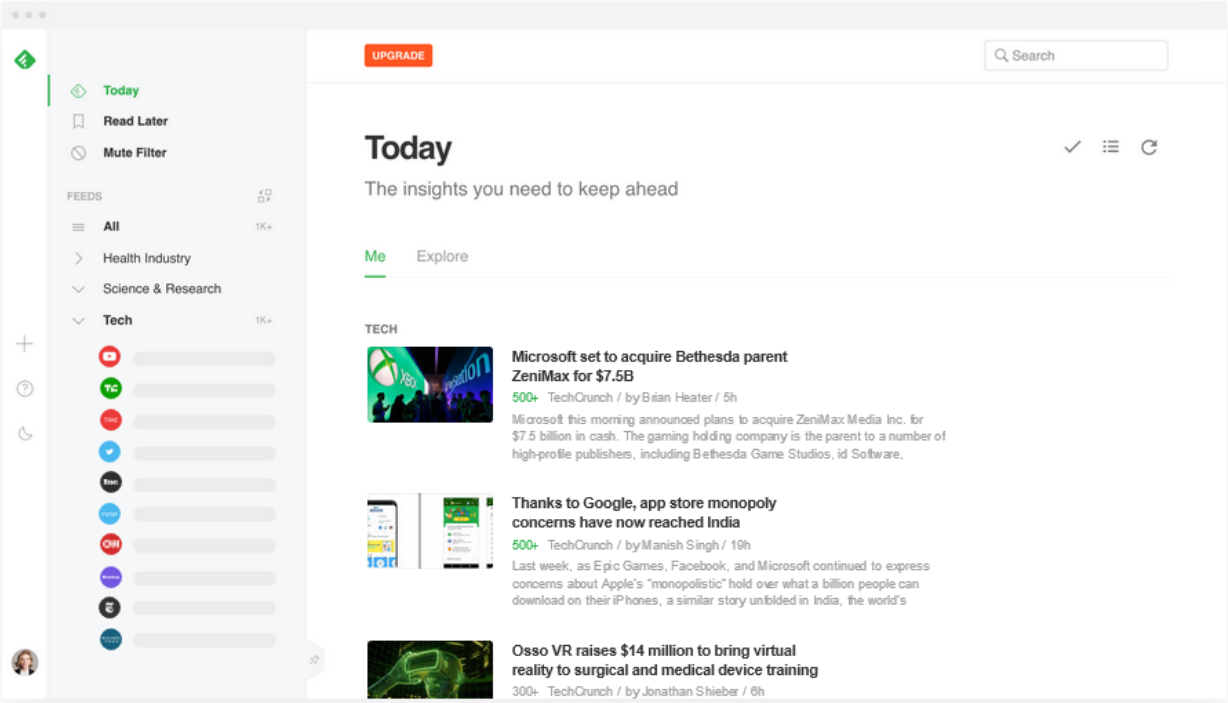
Treće spomenuto rješenje je *News Clusters* koje se može vidjeti na slici 2.3. Za razliku od gore dva navedena rješenja, *News Clusters* se može gledati samo pomoću web preglednika, odnosno

mobilne aplikacije ne postoje. *News Clusters* nudi razne načine za pronalazak vijesti. Mogu se birati razni izdavači kao što su: *Forbes, CNN, BBC* i drugi. Također se može odabrati iz koje zemlje svijeta se želi vidjeti vijesti. Postoji i kategorizacija vijesti, pa tako mogu vidjeti samo vijesti iz sporta, zabave, poslovne vijesti i druge.



Slika 2.3. News Clusters

Feedly (slika 2.4) je još jedan primjer rješenja sustava za prilagodljivi višekriterijski prikaz vijesti iz različitih izvora. On korisniku omogućava kreiranje vlastite liste omiljenih vjesnika s velikim izborom tema [4]. Kao i gore navedeni primjeri *Feedly* također ima implementiran mehanizam preporuka korisniku na temelju dosadašnjih odabira tema i vjesnika. *Feedly* je dostupan kao mobilna aplikacija i kao web rješenje. *Feedly* ima jedno zanimljivo rješenje, a to je Leo, virtualni pomagač koji koristi umjetnu inteligenciju kako bi korisniku prikazao što njemu relevantnije teme i članke. Uz to što korisnik ima mogućnost sam Leu zadati željene teme i članke, korisnik također može prikazane teme i članke označiti kao željene ili neželjene. Svakim korisnikovim odabirom Leo bolje razumije korisnikove potrebe i samim tim prikazuje relevantniji sadržaj.



Slika 2.4. *Feedly*

Iz prethodno navedenih rješenja da se zaključiti da bez obzira na to što svi oni imaju vrlo sličnu svrhu, a to je što bolje personalizirati vijesti i članke za korisnika, sva rješenja imaju različit pristup. Tako neka od rješenja stavljaju naglasak na slike, neka rješenja naglašavaju tekst, a neki su implementirali strojno učenje za dodatno personaliziranje sadržaja. Svima je zajedničko da korisniku daju mogućnost višekriterijskog prikaza vijesti koje se dohvaćaju iz različitih izvora kako bi se smanjilo vrijeme potrebno za posjećivanje svakog izvora zasebno [5]. U sljedećem poglavlju bit će opisano idejno rješenje takvog sustava, definirat će se model i opisati kako će sustav raditi na strani poslužitelja, a kako na strani korisnika.

3. ZAHTJEVI, MODEL I IDEJNO RJEŠENJE

U ovom poglavlju bit će objašnjeni model aplikacije i funkcionalni i nefunkcionalni zahtjevi aplikacije. Također, bit će opisano idejno rješenje web aplikacije.

3.1. Zahtjevi i model aplikacije

3.1.1 Funkcionalni i nefunkcionalni zahtjevi

Prije izrade aplikacije potrebno je jasno definirati zahtjeve. Zahtjevi služe kao vodič prilikom izrade aplikacije. Oni definiraju sve ono što će korisnik moći raditi prilikom korištenja aplikacije. Iako se zahtjevi mogu mijenjati prilikom izrade aplikacije, njihova srž mora ostati netaknuta.

U nastavku će biti objašnjeni funkcionalni zahtjevi web sustava za prilagodljivi prikaz vijesti. Prvo mora biti omogućeno dohvaćanje vijesti iz različitih izvora. Zatim je potrebno sve te podatke parsirati i prilagoditi spomenutoj aplikaciji kako bi se omogućio prikaz vijesti. Nakon što su vijesti prikazane, korisniku se treba omogućiti pretraživanje vijesti, pregled vijesti po kategorijama, sortiranje vijesti i filtriranje vijesti, odnosno prilagoditi prikaz vijesti u ovisnosti s kojih portala korisnik želi vidjeti vijesti. Kako bi aplikacija bila u potpunosti korisna, također se treba korisniku omogućiti čitanje odabrane vijesti. Osim funkcionalnih zahtjeva, nefunkcionalni su također bitni. Jedan od nefunkcionalnih zahtjeva su performanse koje su vrlo bitne i potrebno je omogućiti većem broju korisnika pristup ovom web sustavu. Svi navedeni zahtjevi bit će testirani i ovisno o rezultatima donosit će se odluke o popravku ili poboljšanju dijela ili cijelog sustava. Funkcionalni zahtjevi bit će testirani ručno dok će se za nefunkcionalne zahtjeve koristiti tri različita alata i to su: JMeter, BlazeMeter i VisualStudio. U tablici 3.1. je navedeno pretpostavljeno vrijeme odziva sustava u odnosu na broj korisnika koji koriste sustav.

Tablica 3.1. Nefunkcionalni zahtjevi

Broj korisnika	Odziv (sekunde)
1	1
10	1
80	1
400	4

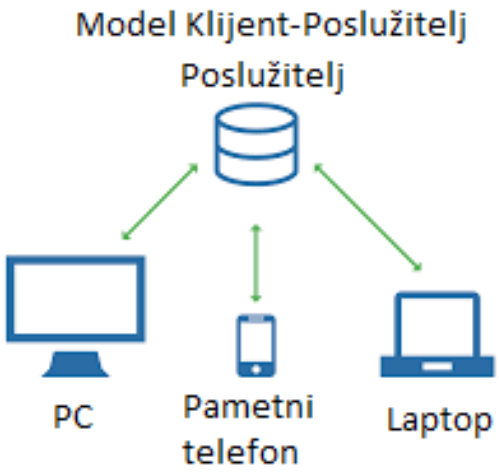
Tablica se temelji na pretpostavki da će sustav imati trenutni odziv za do 100 korisnika, a da će za više od 100 korisnika odziv linearno rasti formulom „broj korisnika/100=odziv“. U tablici 3.2. Navedeni su i opisani funkcionalni zahtjevi sustava.

Tablica 3.2. Osnovni funkcionalni zahtjevi

Zahtjev	Opis
Odabir vjesnika	Omogućiti korisniku odabir jednog ili više vjesnika
Odabir svih vjesnika	Omogućiti korisniku izravan odabir svih vjesnika
Odabir samo sportskih vijesti	Omogućiti korisniku izravan odabir samo sportskih vijesti
Pretraživanje	Omogućiti korisniku pretraživanje po ključnoj riječi
Sortiranje	Prikazati korisniku sortirani sadržaj

3.1.2 Model aplikacije

U izradi web rješenja sustava za prilagodljivi prikaz vijesti bit će korišten klijent-poslužitelj model kao na slici 3.1 [6]. To znači da će aplikacija biti distribuirana i za rad iste će biti potreban klijent (web preglednik) i udaljeni poslužitelj (server). Aplikacija koja koristi ovaj model radi tako da klijent šalje zahtjev poslužitelju, poslužitelj obrađuje zahtjev te klijentu šalje odgovor koji sadrži tražene informacije ili ako je došlo do pogreške, odgovarajuću poruku pogreške. Poslužitelj je uglavnom realiziran nekim od sljedećih programskih jezika: C#, Python, PHP, Ruby itd. Klijent za prikaz dobivenog sadržaja koristi HTML, CSS i JavaScript.



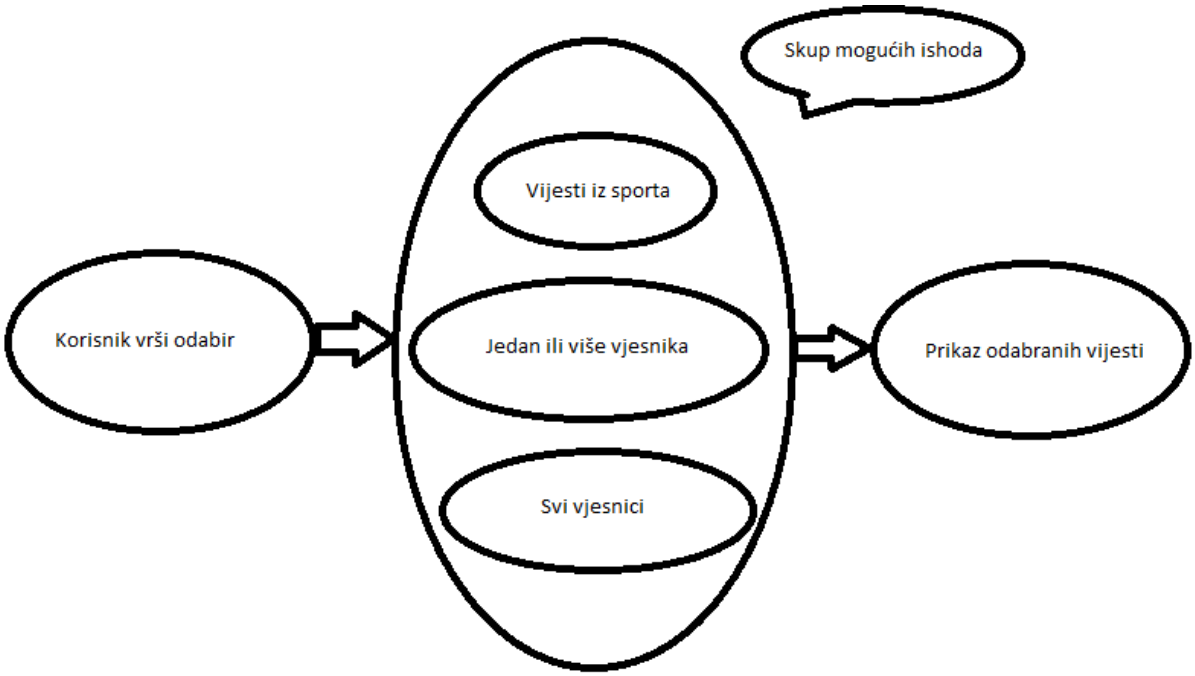
Slika 3.1. Model klijent – poslužitelj [7]

Kao što se na slici 3.1 vidi, jedan poslužitelj može posluživati više različitih korisnika, jer poslužitelj samo obrađuje zahtjeve i šalje podatkovne pakete, dok se sam prikaz primljenih podataka realizira na klijentskoj strani.

3.2. Višekriterijski prikaz vijesti

Općenito, višekriterijsko odlučivanje je proces određivanja ishoda iz skupa ishoda. Većina postojećih rješenja koristi neku vrstu preporuka koje se temelje uglavnom na strojnom učenju te njihov prikaz osim o korisnikovu odabiru ovisi i o preporukama koji generira sustav preporuka [8]. U ovom sustavu za prilagodljivi višektiterijski prikaz vijesti iz različitih izvora nije korišten

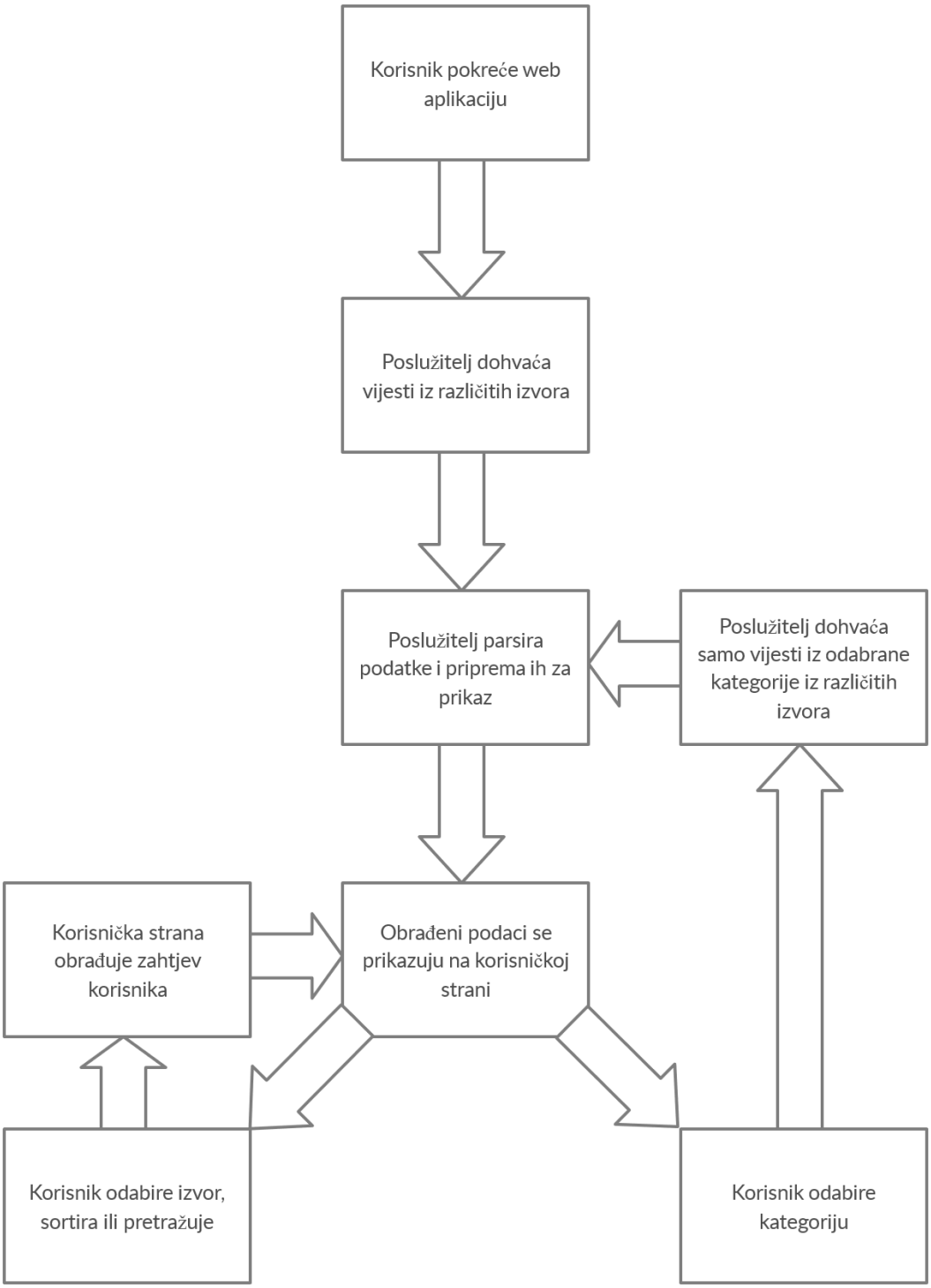
mehanizam preporuka, ali korisnik svoj prikaz može prilagoditi svojim potrebama. Višekriterijski prikaz ovisi o odabiru korisnika, a korisnik bira ishod iz ranije određenog skupa ishoda. Tako korisnik ima mogućnost odabira jednog ili više vjesnika, odabir svih vjesnika i odabir vijesti iz sporta (Slika 3.2). Nakon što se korisniku prikažu vijesti, korisnik ima mogućnost pretraživanja, što bi bio još jedan kriterij prikaza, ali takav prikaz bi svakako pripadao jednom od tri navedena ishoda.



Slika 3.2. Višekriterijski prikaz vijesti

3.3. Idejno rješenje sustava na strani poslužitelja

Prilikom rada sustava za prilagodljivi prikaz vijesti poslužitelj će biti taj koji će prikupljati podatke iz različitih izvora pomoću tehnologije rss. Te sirove podatke će zatim parsirati kako bi se iz svih podataka dobilo samo ono najbitnije. Poslužitelj će dohvaćati i kategorizirane vijesti iz različitih izvora te ih spajati u jedan skup. To će se događati na zahtjev, kada klijent primjerice klikne na kategoriju Sport. Poslužitelj će dohvatiti sportske vijesti sa svih odabranih portala, parsirati podatke, spremiti u jedan skup te ih proslijediti klijentu za prikaz. Takav tok programa može se vidjeti na slici 3.3 Na taj način, klijent će uvijek imati sve najnovije vijesti iz odabrane kategorije.



Slika 3.3. Dijagram toka web sustava za prilagodljivi prikaz vijesti iz različitih izvora

3.4. Idejno rješenje sustava na strani korisnika

S korisničke strane moći će se vidjeti stranica vizualno i sastojat će se od izbornika s kategorijama, filtera, polja za upisivanje riječi za pretragu te liste vijesti. U pozadini će se izvršavati operacije nad podacima. Te operacije će biti pretraživanje, filtriranje i sortiranje. Tako će korisnik moći

birati želi li vidjeti sve vijesti ili samo određenu kategoriju. Također će moći sortirati vijesti po vremenu nastanka, ali i pretraživati. Pretraživanje će biti vezano za kategoriju vijesti koje korisnik odabere. Ako korisnik odabere kategoriju vijesti sport i pretražuje određenu ključnu riječ, pretraživat će se samo vijesti u kategoriji sport.

Na slici 3.4 može se vidjeti prototip dizajna web stranice. Sastoji se od uzglavlja i tijela. Uzglavlje sadrži naslov stranice, polja za pretraživanje i padajućeg izbornik za izbor kategorije vijesti. Tijelo stranice sadrži skup elemenata koji prikazuju pojedinu vijest. Svaki element se sastoji od slike, naslova vijesti i njenog opisa. Klikom na vijest link nas vodi na izvornu vijest na izvornom portalu.

Slika 3.4. Prototip dizajna web stranice

4. PROGRAMSKO RJEŠENJE WEB SUSTAVA ZA PRILAGODLJIVI PRIKAZ VIJESTI

4.1. Korišteni jezici, alati i okoline za implementaciju web sustava za prilagodljivi prikaz vijesti

4.1.1. Visual Studio Code

Visual Studio Code je besplatni editor programskog koda koji je razvila tvrtka *Microsoft* za operacijske sustave *Windows, macOS* i *Linux* [9]. Nudi razne mogućnosti među kojima su otklanjanje grešaka (*debugging*), inteligentno dovršavanje koda i refaktoriranje koda. Također podržava instaliranje raznih tema, dodataka te dodavanje prečaca tipkovnice koji dodatno olakšavaju rad. Visual Studio Code podržava mnoge programske jezika kao što su Java, C++, Python, Go, JavaScript, F# i druge. VS Code također podržava HTML, CSS, SQL, XML i JSON. Još jedna vrlo korisna mogućnost VS Code-a je korištenje git naredbi iz samog editora što dodatno olakšava verzioniranje koda.

4.1.2 Python

Python je interpreterski programski jezik opće namjene. Stvorio ga je Guido van Rossum s idejom stvaranja programskog jezika u kojem se lako razvija kod koji je ujedno i lako čitljiv [10]. Za razliku od mnogih drugih jezika Python ne koristi točka zarez na kraju linije koda. Također umjesto uglatih zagrada za definiranje blokova koda koristi tabulator. Kao i mnogi drugi jezici za nazive ključnih riječi koristi riječi iz Engleskog jezika. Sintaksa Pythona je uglavnom slična sintaksi programskih jezika Java i C iako postoje razlike. Neke od razlika su kod *bool* operatera gdje se za razliku od Jave i C-a ne koriste znakovi „&&“, „||“ i „!“ nego riječi Engleskog jezika „*and*“, „*or*“ i „*not*“. Vjerojatno najveća razlika između C-a i Python-a su tipovi podataka koje se u Pythonu ne mora navoditi za razliku od C-a gdje se za svaku varijablu mora točno reći kojeg je tipa i toga se mora strogo pridržavati prilikom pisanja koda. Python podržava strukturno i objektno orijentirano programiranje, a mnoge stavke Pythona podržavaju i funkcionalno programiranje. Python podržava mnoge biblioteke za razne namjene. Neke od korištenijih su biblioteke za: strojno učenje, umjetnu inteligenciju, obradu slika, baze podataka, matematičku obradu, razvoj web sustava. Python se koristi za rad na velikim i malim sustavima, a koriste ga i velike organizacije kao što su *Google, CERN, NASA, Facebook, Amazon* i mnogi drugi.

4.1.3 Flask

Flask je mikro programski okvir pisan u programskom jeziku Python. Danas je jedan od najpopularnijih Python web programskih okvira. Flask ne nameće nikakve ovisnosti ili izgled

projekta, to je sve na volju razvojnog programera. Programer također sam bira koje će biblioteke koristiti. Temelj Flask-a su dva projekta Werkzeug i Jinja [11]. Werkzeug sadrži metode za realizaciju zahtjeva, odgovora i druge uslužne funkcije dok Jinja obrađuje predloške. Neka od Flask-ovih svojstava su integrirana podrška za unit testove, zasnovan je na unicode-u, dostupna je velika količina dodataka koje povećavaju Flask-ove mogućnosti i druga[12]. Jedna od temeljnih značajki Flask-a su rute. Rute su dekoratori koji Flask-u govore koji *url* će pokrenuti funkciju ispod dekoratora. Tom značajkom je omogućeno lako spajanje poslužiteljske i korisničke strane. Za prikaz korisničkog sučelja koristit će se funkcija *render_template*. Funkcija kao parametre prima lokaciju dokumenta pisanog u HTML-u koji se treba prikazati i opcionalno podatke koje treba proslijediti na korisničko sučelje. Predlošci (eng. *templates*) su dokumenti koji sadrže statičke podatke i trebaju se nalaziti u direktoriju *templates* kako bi ih Flask prepoznao. Flask koristi Jinja template library biblioteku za renderiranje predložaka [13].

4.1.4 HTML i CSS

HTML ili *Hyper Text Markup Language* je jezik koji služi za izradu web stranica [14]. On nije programski jezik već opisni ili prezentacijski jezik. Služi za definiranje vizualnog izgleda stranice uz CSS. HTML čine dvije osnovne jedinice, a to su oznake i atributi. Oznake služe za definiranje početka i kraja HTML elementa. HTML ima nekoliko osnovnih oznaka a to su *DOCTYPE* kojom se označava točna inačica HTML dokumenta, *head* koji sadrži meta informacije o HTML dokumentu kao što su naslov stranice i oznaka *body* koja služi kao kontejner za sve vidljive dokumente. Atributi su dio HTML dokumenta koji pobliže opisuju element. Primjerice atribut *width* definirat će širinu elementa, atribut *height* definirat će širinu elementa, a atribut *style* definirat će boju, font i slično [15]. Svakom HTML dokumentu može se dodijeliti id i/ili ime klase. Id je jedinstvena oznaka karakteristična samo za taj odabrani element,dok ime klase označava skupinu elemenata. Oba identifikatora kao i HTML oznake koriste se prilikom definiranja stila elementa pomoću CSS-a.

CSS ili *Cascading Style Sheets* je opisni jezik koji definira vizualni izgled HTML elementa. CSS-om se može primjerice definirati boja, veličina i pozicija HTML elementa. CSS se može pisati na tri načina. U redu, a to je upravo onaj atribut *style* HTML elementa, može ga se pisati unutar *head* oznake ili u vanjskom dokumentu [16]. Najčešće se koristi pisanje u vanjskom dokumentu zbog veće preglednosti koda i jednostavnijih budućih izmjena. Sintaksa CSS-a je jednostavna. Prvo se piše selektor kojim se definira na koji element ili skupinu elemenata se želi utjecati s tim blokom. Zatim se otvaraju uglate zagrade u koje se upisuju deklaracije odvojene točka-zarezom. Svaka deklaracija se sastoji od imena svojstva i vrijednosti. Selektor može označavati jedan ili skupinu

elemenata. Pomoću toga može se vrlo lako i uredno organizirati koji elementi trebaju poprimiti koje svojstvo. Uz HTML oznake, kao selektore se može definirati i koristiti klase i id-eve. Klasa se koristi kada se želi da više elemenata poprimi ista CSS svojstva.Id se koristi za obilježavanje samo jednog elementa.

Na slici 4.1 se vidi primjer HTML koda s CSS-om pisanim unutar *head* oznaka. Otvori li se taj dokument pomoću internet preglednika vidi se jedan centrirani naslov pisan bijelom bojom na plavoj pozadini i centrirani paragraf ispod naslova pisan plavom bojom. Da se zamijetiti da je za identificiranje naslova korišten id koji se označava oznakom „ljestve“, a za identificiranje paragrafa korištena je klasa koja se označava točkom [17].

```
doc.html ×
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <style>
5
6   #headerOne {
7     background-color: blue;
8     color: white;
9     padding: 60px;
10    text-align: center;
11  }
12
13  .clName {
14    color: blue;
15    text-align: center;
16    padding: 10px;
17  }
18  </style>
19  </head>
20  <body>
21
22  <h1 id="headerOne">Header One</h1>
23
24  <p class = "clName">Paragraph</p>
25
26  </body>
27  </html>
```

Slika 4.1. Primjer HTML i CSS koda

4.2. Programsko rješenje na strani poslužitelja

Programsko rješenje na strani poslužitelja realizirano je korištenjem programskog jezika python i mikro programskog okvira flask uz dodatno korištenje nekoliko biblioteka koje će u nastavku biti navedene i objašnjene. Poslužiteljski dio rješenja podijeljen je u dva dokumenta. Glavni dokument u kojem su definirane sve rute zove se *routes.py*. Sporedni dokument sadrži pomoćne funkcije potrebne za rad sustava i zove se *feed.py*. Oba dokumenta s pripadajućim rutama i funkcijama bit će objašnjena u nastavku. U tablici 4.1 navedene su sve rute i pomoćne funkcije korištene za realizaciju poslužiteljske strane sustava za prilagodljivi višekriterijski prikaz vijesti iz različitih izvora.

Tablica 4.1. Tablica ruta i pomoćnih funkcija

Rute	Funkcije
home	get_raw_feed
content	get_feed_from_multiple_sources
search	get_image
all	get_description
sport	get_urls_from_filter

Sve navedene rute bit će detaljno objašnjene u nastavku, također će biti prikazan izvorni kod istih na slikama.

4.2.1 Dohvaćanje podataka s različitih izvora

Za dohvaćanje vijesti s različitih izvora prvo je potrebno prikupiti *url*-ove s kojih podatke možemo dohvatiti. Za potrebe ovog rješenja prikupljeni su i korišteni *url*-ovi sapet različitih izvora (vjesnika) i to su index.hr, rtl.lu, bbc.com, cnn.com i nytimes.com. Pomoću python programskog jezika realizirana je funkcija koja dohvaća sirove podatke s predanog *url*-a te ju se može vidjeti na slici 4.2.

```
def get_raw_feed(url):
    http = urllib3.PoolManager()
    raw_feed = fp.parse(url)
    return raw_feed.entries
```

Slika 4.2. Funkcija koja dohvaća sirove podatke

Na slici se vidi da se u varijablu *raw_feed* spremi ono što *feedparser* (na slici fp) dohvati s predanog *url*-a. Funkcija vraća ne parsirane vijesti.

4.2.2. Funkcija get feed from multiple sources

Kako bi sustav ispravno funkcionirao iz sirovih podataka je potrebno dobiti iskoristive podatke koji će se moći uredno prikazati na klijentskoj strani sustava. Funkciju *get_raw_feed* poziva funkcija *get_feed_from_multiple_sources* koja kao ulazni parametar prima listu *url*-ova s kojih se treba dohvatiti vijesti što se vidi na slici 4.3.

```
def get_feed_from_multiple_sources(urls):
    dictionaty_list = []
    for url in urls:
        raw_feed = get_raw_feed(url)
        for item in raw_feed:
            dictionary = {
                "title" : item.title,
                "description" : get_description(item, url),
                "link" : item.link,
                "image" : get_image(item, url),
                "published" : get_published(item, url)
            }
            dictionaty_list.append(dictionary)
    return dictionaty_list
```

Slika 4.3. Funkcija koja dohvaća podatke sa različitih izvora

Kao što se vidi na slici 4.3 funkcija get_feed_from_multiple_sources za svaki url iz liste poziva funkciju get_raw_feed koja joj vraća sirove podatke.U sljedećoj for petlji svaki item predstavlja jednu vijest. Iz vijesti se uzima pet najbitnijih značajki, a to su: naslov, opis, link na vijest, link slike i vrijeme objave. Svaka vijest se sprema u python tip podatka imenik (eng. Dictionary) zbog daljnjeg lakšeg korištenja. Svaki, sada imenik koji sadrži najbitnije značajke vijesti se sprema u listu. Funkcija get_feed_from_multiple_sources vraća upravo tu listu, listu imenika s najbitnijim značajkama svake pojedine vijesti. Unutar get_feed_from_multiple_sources funkcije poziva se tri dodatne funkcije, a to su get_description, get_image i get_published. Te funkcije se koriste zbog različitog formatiranja podataka od strane vjesnika i bit će objašnjene u nastavku.

4.2.3. Funkcija *get image*

Ova funkcija ima jednu funkcionalnost, a to je dohvaćanje slike iz sirovih podataka vijesti. Kao parametre prima *item* odnosno vijest i *url* s kojeg dolazi vijest. *Url* se koristi za razlikovanje vijesti i ovisno o njemu različito se dohvaća slika. Primjerice rtl sliku sadrži unutar atributa *links* na

drugom mjestu. U slučaju da dohvaćanje slike ne uspije kao slika se postavlja odgovarajuća predefinirana slika. U slučaju bbc-a slika ne postoji unutar dohvaćenih vijesti stoga se postavlja predefinirana slika. Funkciju *get_image* može se pogledati na slici 4.4.

```
def get_image(item, url):
    if 'rtl' in url:
        try:
            return item.links[1].href + ' '
        except:
            return 'static/rtl.png'
    if 'index' in url:
        return item.description[10:item.description.index('"', 12)] + ' '
    if 'bbc' in url:
        return 'static/bbc.png'
    if 'nytimes' in url:
        try:
            temp = str(item.media_content)
            image_url = temp[(temp.index('url') + 7):temp.index('jpg') + 3]
            return image_url + ' '
        except:
            return 'static/nytimes.jpg'
    if 'cnn' in url:
        try:
            temp = str(item.media_content)
            image_url = temp[(temp.index('url') + 7):temp.index('jpg') + 3] + ' '
            return image_url
        except:
            return 'static/cnn.png'
```

Slika 4.4. Funkcija za dohvaćanje slike iz vijesti

4.2.4. Funkcija get description

Za dohvaćanje opisa vijesti koristi se funkcija *get_description* koju se može vidjeti na slici 4.5. Funkcija prima iste parametre kao funkcija *get_image* ali kao rezultat vraća opis vijesti. U slučaju opisa jedino index.hr i cnn.com imaju različiti format spremanja, za ostala tri izvora format je isti te se opis nalazi unutar atributa *description*. Index.hr unutar svog opisa sadrži i link slike koji nam u opisu ne treba te ga rezanjem stringa izbacujemo. Cnn.com na kraju svog opisa ima nepotrebne informacije koje su također uklonjene rezanjem stringa.

```
def get_description(item, url):
    if 'index' in url:
        return item.description[item.description.index(' ', 12) + 1:]
    if 'cnn' in url:
        return item.description[:item.description.index('<')]
    else:
        return item.description
```

Slika 4.5. Funkcija koja dohvaća opis vijesti

4.2.5. Funkcija get urls from filter

Funkcija kao parametar prima filtare koji su lista imena vjesnika s kojih se želi dohvatiti vijesti. Za svaki filter se provjerava pripada li nekom od predefiniranih url-ova što se može vidjeti na slici 4.6. Ako pripada nekom od predefiniranih url-ova dodaje ga se u listu url-ova koje će funkcija vratiti te s kojih će se daljnjom obradom dohvatiti vijesti..

```
def get_urls_from_filter(filter):
    urls_to_get = []
    with open('url.json', 'r') as json_urls:
        data = json_urls.read()
    urls = json.loads(data)
    for item in filter:
        if item in urls['index']['news']:
            urls_to_get.append(urls['index']['news'])
        elif item in urls['rtl']['news']:
            urls_to_get.append(urls['rtl']['news'])
        elif item in urls['other']['bbc']:
            urls_to_get.append(urls['other']['bbc'])
        elif item in urls['other']['cnn']:
            urls_to_get.append(urls['other']['cnn'])
        elif item in urls['other']['nytimes']:
            urls_to_get.append(urls['other']['nytimes'])
        elif item == 'sport':
            urls_to_get.append(urls['rtl']['sport'])
            urls_to_get.append(urls['index']['sport'])
    return urls_to_get
```

Slika 4.6. Funkcija za dohvaćanje *url*-ova na temelju filtera

4.2.6. Ruta *home*

Ruta home ima samo jednu funkciju i to je funkcija *render_template* kojom se prikazuje *home.HTML* dokument na korisničkoj strani. Funkcija se vidi na slici 4.7 gdje se vidi i lokacija dokumenta kao parametar funkcije *render_template.*

```
@app.route('/home', methods=['GET', 'POST'])
def home():
    return render_template("home.html")
```

Slika 4.7. Ruta *home*

4.2.7. Ruta *content*

Korisniku se daje na izbor iz kojih vjesnika želi gledati vijesti. To radi na način da na korisničkom sučelju odabere željene vjesnike i podnese formu klikom na gumb *submit*. Forma odabrane vrijednosti šalje na poslužiteljsku stranu pozivom funkcije *content*. Na slici 4.8 vidi se funkcija *content* koja prima vrijednosti od forme te ih sprema u varijablu *filter*.

```
@app.route('/content', methods=['GET', 'POST'])
def content():
    urls_to_get = []
    filter = request.form
    urls_to_get = get_urls_from_filter(filter)
    content = get_feed_from_multiple_sources(urls_to_get)
    if content == []:
        return render_template("home.html")
    return render_template("content.html", content = content)
```

Slika 4.8. Funkcija za prikaz vijesti s odabranih vjesnika

Nakon što funkcija primi vrijednosti od forme poziva funkciju *get_urls_from_filter* koja kao rezultat vraća *url*-ove željenih vjesnika. Funkcija *get_urls_from_filter* bit će objašnjena u nastavku. Nakon što funkcija *content* ima *url*-ove predaje ih funkciji *get_feed_from_multiple_sources* koja joj kao što je ranije objašnjeno, vraća vijesti s odabranih vjesnika. Ako funkcija *get_feed_from_multiple_sources* vrati praznu listu znači da nema vijesti iz odabranih vjesnika ili je došlo do greške. U tom slučaju korisnik ostaje na početnoj stranici. Ako

je funkcija *get_feed_from_multiple_sources* vratila listu s vijestima, korisnik se preusmjerava na stranicu s prikazanim vijestima funkcijom *render_template* s parametrima ime stranice na koju se preusmjerava i podacima za prikaz.

4.2.8. Ruta *search*

Nakon što je korisnik preusmjeren na stranicu s prikazanim vijestima ima mogućnost pretraživanja vijesti. U tekstualno polje unosi tekst koji želi pretraživati i potvrđuje pritiskom tipke *enter*. Polje za pretragu je također forma koja nakon potvrde vrijednosti šalje na poslužiteljsku stranu definiranom rutom koja zatim poziva funkciju za pretragu. Funkcija *search* (slika 4.9) prvo zaprima vrijednost koju treba pretražiti. Zatim dohvaća *url*-ove svih vjesnika te poziva funkciju *get_feed_from_multiple_sources* koja dohvaća sve vijesti iz kojih će se vršiti pretraga. Prolaskom kroz sve vijesti provjerava se nalazi li se traženi pojam u naslovu ili opisu pojedine vijesti. Ako se traženi pojam nalazi u naslovu ili opisu, ta vijest se dodaje u novu listu koja će se predati kao parametar funkciji *render_template* kako bi se rezultati pretrage prikazali korisniku.

```
@app.route('/search', methods=['GET', 'POST'])
def search():
    urls_to_get = []
    filtered_content = []
    response = request.form.to_dict()
    searched = response['searched']
    all_list = []
    with open('url.json', 'r') as json_urls:
        data = json_urls.read()
    urls = json.loads(data)
    urls_to_get.append(urls['index']['news'])
    urls_to_get.append(urls['other']['bbc'])
    urls_to_get.append(urls['other']['cnn'])
    urls_to_get.append(urls['other']['nytimes'])
    urls_to_get.append(urls['rtl']['news'])
    content = get_feed_from_multiple_sources(urls_to_get)
    for item in content:
        if(searched in item['title'] or searched in item['description']):
            filtered_content.append(item)
    return render_template("content.html", content = filtered_content)
```

Slika 4.9. Funkcija za pretragu

4.2.9. Rute *all* i sport

Kako korisnik ne bi morao označiti sve vjesnike ako želi vidjeti vijesti sa svih njih, uveden je gumb *all* koji poziva istoimenu rutu (slika 4.10). Funkcija *all* funkciji

get_feed_from_multiple_sources predaje *url*-ove svih vjesnika i kao rezultat dobiva sve vijesti koje prikazuje pomoću funkcije *render_template* na *content* stranici.

```
@app.route('/all', methods=['GET'])
def all():
    urls_to_get = []
    all_list = []
    with open('url.json', 'r') as json_urls:
        data = json_urls.read()
    urls = json.loads(data)
    urls_to_get.append(urls['index']['news'])
    urls_to_get.append(urls['other']['bbc'])
    urls_to_get.append(urls['other']['cnn'])
    urls_to_get.append(urls['other']['nytimes'])
    urls_to_get.append(urls['rtl']['news'])
    content = get_feed_from_multiple_sources(urls_to_get)
    return render_template("content.html", content = content)
```

Slika 4.10. Funkcija za prikaz svih vijesti

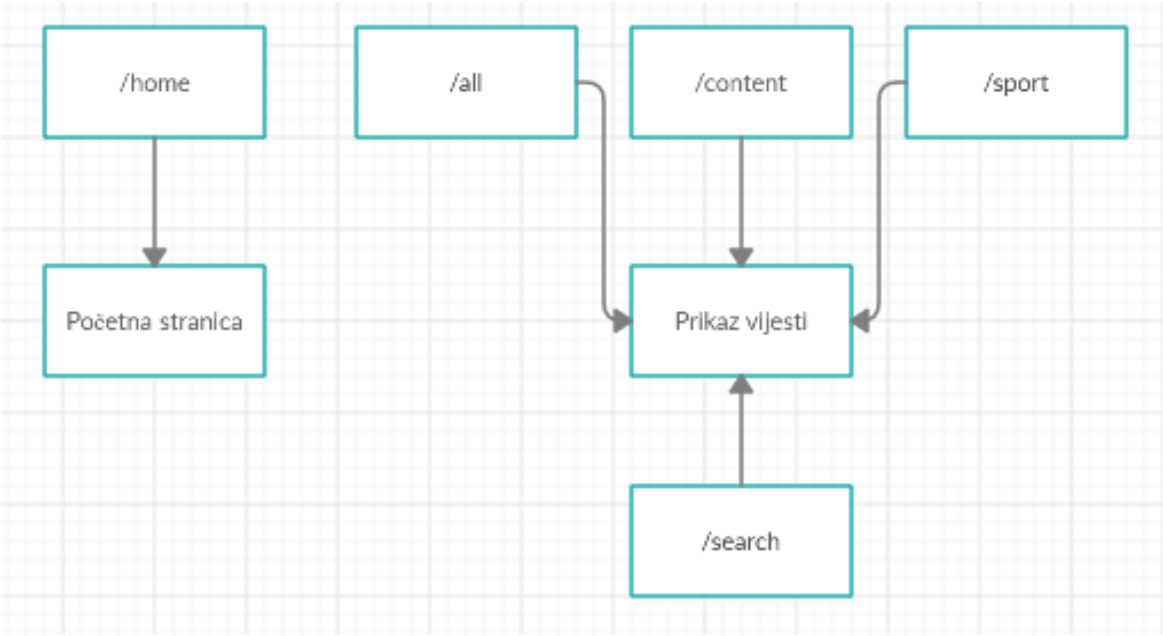
Na isti način kreirana je i ruta sport (slika 4.11), jer je sport jedna od najpopularnijih kategorija vijesti.

```
@app.route('/sport', methods=['GET'])
def sport():
    urls_to_get = []
    sport_list = []
    with open('url.json', 'r') as json_urls:
        data = json_urls.read()
    urls = json.loads(data)
    urls_to_get.append(urls['index']['sport'])
    urls_to_get.append(urls['rtl']['sport'])
    content = get_feed_from_multiple_sources(urls_to_get)
    return render_template("content.html", content = content)
```

Slika 4.11. Funkcija za prikaz vijesti iz sporta

4.3. Programsko rješenje na strani korisnika

Programsko rješenje na strani korisnika realizirano je tehnologijama HTML i CSS uz korištenje poslužiteljske strane kao izvora podataka i sustava za obradu podataka. Korisnički prikaz sastoji se od dvije stranice, početne i stranice za prikaz vijesti. Na slici 4.12 dijagram pokazuje koja će se stranica prikazivati pozivom određene rute.



Slika 4.12. Ovisnost prikaza stranice o pozivanju rute

4.3.1. Početna stranica

Početna stranica je prva stranica koju korisnik vidi dolaskom na ovaj web sustav. Stranica se sastoji od naslova stranice, forme za korisnički odabir vjesnika, gumba za podnošenje forme i dva gumba, prvi za prikaz svih vijesti i drugi za prikaz samo vijesti iz sporta. Forma se sastoji od šest kontejnera unutar kojeg se nalaze slika i potvrdni okvir (eng. Checkbox) te gumba za podnošenje forme. Svaki potvrdni okvir ima svoje ime i vrijednost koji se, ako je potvrdni okvir označen, kao uređeni par šalju na poslužiteljsku stranu prilikom podnošenja forme. Klikom na gumb za podnošenje forme aktivira se akcija koja šalje podatke na definirani *url* što se može vidjeti na slici 4.13 u drugom redu.

```html
<div id = "form_container">
    <form id = "grid_container" action = "{{url_for('content')}}" method = "post" name = "select_for">
        <div class = "site">
            <input class = "check" id = "check1" type="checkbox" name = "index" value = "index">
            <span class="checkmark_filler"></span>
            <label class = "checkimg" for="check1"><img class = "checkimg" src="../static/index.png" /></label>
        </div>
        <div class = "site">
            <input class = "check" id = "check2" type="checkbox" name = "rtl" value = "rtl">
            <span class="checkmark_filler"></span>
            <label class = "checkimg" for="check2"><img class = "checkimg" src="../static/rtl.png" /></label>
        </div>
        <div class = "site">
            <input class = "check" id = "check3" type="checkbox" name = "bbc" value = "bbc">
            <span class="checkmark_filler"></span>
            <label class = "checkimg" for="check3"><img class = "checkimg" src="../static/bbc.png" /></label>
        </div>
        <div class = "site">
            <input class = "check" id = "check4" type="checkbox" name = "cnn" value = "cnn">
            <span class="checkmark_filler"></span>
            <label class = "checkimg" for="check4"><img class = "checkimg" src="../static/cnn.png" /></label>
        </div>
        <div class = "site">
            <input class = "check" id = "check5" type="checkbox" name = "nytimes" value = "nytimes">
            <span class="checkmark_filler"></span>
            <label class = "checkimg" for="check5"><img class = "checkimg" src="../static/nytimes.jpg" /></label>
        </div>
        <div class = "site">
            <input class = "check" id = "check6" type="checkbox" name = "sport" value = "sport">
            <span class="checkmark_filler"></span>
            <label class = "checkimg" for="check6"><img class = "checkimg" src="../static/sport.png" /></label>
        </div>
        <input id = "submit" type="submit" value = "Submit">
    </form>
</div>
```

Slika 4.13. HTML rješenje forme za odabir vjesnika

Ispod forme se nalaze dva gumba (slika 4.14). Gumb za prikaz svih vijesti i gumb za prikaz vijesti iz sporta. Oba gumba se nalaze u jednom kontejneru. Klikom na željeni gumb poziva se odgovarajuća ruta te se izvršava funkcija ispod rute koja korisnika vodi na stranicu za prikaz vijesti uz prikaz odgovarajućeg sadržaja.

```html
<div class = "buttons">
    <a href="{{url_for('all')}}" >
        <input id = "button_all" type="button" value="All" />
    </a>
    <a href="{{url_for('sport')}}" >
        <input id = "button_sport" type="button" value="Sport" />
    </a>
</div>
```

Slika 4.14. HTML rješenje za prikaz svih vijesi ili samo sportskih vijesti

4.3.2. Stranica za prikaz vijesti

Podnošenjem forme ili klikom na gumbe *all* ili sport korisnik se preusmjerava na stranicu za prikaz vijesti uz odgovarajući sadržaj. Stranica za prikaz vijesti sastoji se od naslova, linka na početnu stranicu, polja za pretraživanje i liste prikazanih vijesti. Svaka vijest se sastoji od naslova, opisa

vijesti i slike. Svaka slika u sebi sadrži link te se klikom na sliku korisnika vodi na izvornu vijest. Naslov i link na početnu stranicu nalaze se unutar jednog kontejnera te se nalaze na naslovnoj traci. Polje za pretraživanje nalazi se ispod naslovne trake i unutar je vlastitog kontejnera. Polje za pretraživanje realizirano je kao forma koja unutar sebe ima polje za tekstualni unos. Ova forma nema gumb za podnošenje forme već se forma podnosi klikom na gumb *enter*. HTML rješenje naslovne trake i polja za pretraživanje može se vidjeti na slici 4.15.

```
<div id = "title_container">
    <h1 id = "news">News</h1>
    <a class = "home_button" href="{{url_for('home')}}" >
    <input class = "home_button" type="button" value="Home" />
 </a>
</div>

<div id = "search_container">
    <form id = "search_form" action = "{{url_for('search')}}" method = "post" name = "search">
        <input id = "text_field" type="text" id="search_input" name = "searched" placeholder="Search..">
    </form>
</div>
```

Slika 4.15. HTML rješenje naslovne trake i polja za pretraživanje

Sve vijesti se prikazuju ispod gore navedenih elemenata te imaju vlastiti kontejner. Svaka pojedina vijest također ima vlastiti kontejner u kojem se naslov, opis i slika vijesti. Kontejneri vijesti se kreiraju s obzirom na to koliko vijesti je predano na korisničko sučelje i zbog toga stranica izgleda jednako bez obzira bila jedna vijest ili njih pedeset. Na slici 4.16 može se vidjeti da je kreiranje kontejnera vijesti i sadržaja unutra realizirano for petljom.

```
<div id = "master_container">
{% for it in content %}
    <div class="row">
        <div class="column">
          <h2 class = "title">{{it.title}}</h2>
          <p class = "description">{{it.description}}</p>
          <a class = "image" href = {{it.link}}><img class = "image" src = {{it.image}}/></a>
        </div>
    </div>
{% endfor %}
</div>
```

Slika 4.16. HTML rješenje prikaza vijesti

4.3.3. CSS stiliziranje

Za stiliziranje web rješenja korišten je CSS. Pomoću njega definirane su sve boje, margine, fontovi i drugo. Iako se CSS može pisati unutar samog HTML dokumenta, u svrhu ovog rada pisan je u vanjski dokument. Kako flask traži poštivanje određenih pravila prilikom kreiranja strukture

projekta, svi CSS dokumenti moraju se nalaziti u direktoriju *static* [18]. U nastavku, na slici 4.17 može se vidjeti dio CSS dokumenta koji je korišten za stiliziranje ovog web sustava. U prvom redu može se vidjeti da je kao selektor korišten HTML oznaka dok je za ostala tri elementa kao oznaka korišten id. Na slici se može vidjeti da je za tijelo (eng. *body*) stranice definirana boja s hex vrijednošću. Za definiranje boje još se može koristiti rgb, rgba, hsl, hsla vrijednosti i predefinirana imena [19]. U svrhu ovoga rada korištene su hex vrijednosti i predefinirana imena boja. Za stiliziranje elementa koji ima id *news* korišteno je CSS svojstvo za font, boju fonta, poziciju, prikaz i drugi.

```
1   body {
2       background-color: #e0e0eb;
3   }
4
5   #title_container {
6       position: relative;
7       left: 0px;
8       right: 0px;
9       margin-left: 0px;
10      margin-right: 0px;
11      margin-top: 0px;
12      height: 100px;
13      background-color:  #C70039 ;
14
15  }
16
17  #news {
18      position: relative;
19      display: inline;
20      top: 5%;
21      left: 2%;
22      text-align: left;
23      font-size: 80px;
24      font-weight: normal;
25      color: #e0e0eb;
26      font-family: serif;
27      border: none;
28      background-color: inherit;
29  }
30
31  #title_holder {
32      position: relative;
33      text-decoration: none;
34  }
```

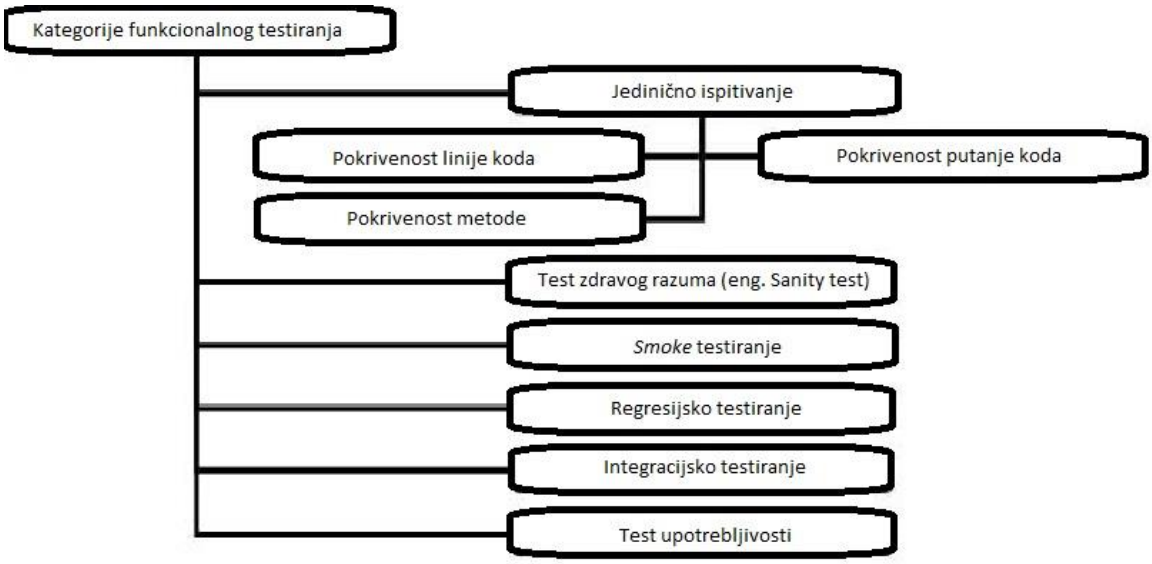
Slika 4.17. Dio CSS dokumenta korišten za stiliziranje web sustava.

4.4. Analiza i testiranje programskog rješenja

U sljedećih nekoliko potpoglavlja bit će navedene i opisane dvije osnovne vrste testiranja, a to su funkcionalno testiranje i nefunkcionalno testiranje. Također će biti opisana provedba tih testova na programskom rješenju web sustava za prilagodljivi višekriterijski prikaz vijesti iz različitih izvora te rezultati istih. Na kraju će biti komentirani rezultati te eventualne izmjene programskog rješenja u svrhu popravka i poboljšanja performansi.

4.4.1. Funkcionalno testiranje

Funkcionalno testiranje je vrsta testiranja koje ispituje ispunjava li aplikacija sve ranije definirane zahtjeve. Te zahtjeve definira klijent i oni su osnovna vodilja pri izradi aplikacije. Ovaj tip testiranja ne bavi se tim kako će aplikacija obaviti određeni posao već što će biti rezultat toga posla [20]. Primjerice, klijent je kao jedan od zahtjeva definirao da aplikacija mora imati mehanizme registracije i prijave u aplikaciju. Funkcionalno testiranje testirat će može li se korisnik registrirati i prijaviti u aplikaciju, ali neće gledati na koji način aplikacija to radi. Funkcionalno testiranje ima nekoliko tipova, a neki od njih su: *sanity* testiranje, *smoke* testiranje, integracijsko testiranje i regresijsko testiranje [21]. Cijela podjela funkcionalnog testiranja može se vidjeti na slici 4.17. Funkcionalno testiranje programske podrške je vrlo važan dio razvoja. Može se provoditi za vrijeme razvoja aplikacije i nakon završetka razvoja. Ako se provodi ispravno i redovito uvelike smanjuje mogućnost pojave pogrešaka u produkcijskim aplikacijama.



Slika 4.17. Tipovi funkcionalnog testiranja [22]

4.4.2. Rezultati provedbe funkcionalnog testiranja

Prilikom provođenja funkcionalnog testiranja pronađene su tri pogreške. U nastavku će svaka pojedina pogreška biti navedena i detaljno razjašnjena. Također će biti naveden način na koji se svaka može otkloniti. U tablici 4.1 navedene su pogreške pronađene prilikom provođenja funkcionalnog testiranja.

Tablica 4.1. Prikaz pogrešaka pronađenih provedbom funkcionalnog testiranja

Redni broj	Naziv	Prioritet	Blokirajuća greška
1.	Nedostaje tekst za nula rezultata	Srednji	Ne
2.	Nedostaje pretraživana riječ	Srednji	Ne
3.	Označeni tekst	Niski	Ne

Na slici 4.18 može se vidjeti greška nedostajanja teksta za nula rezultata pretraživanja.

Slika 4.18. Greška nedostaje tekst za nula rezultata pretraživanja

Ako se u polje za pretraživanje unese tekst koji se ne nalazi u niti jednom naslovu ili opisu vijesti tada se javlja greška nedostaje tekst za nula rezultata pretraživanja. Ova greška nije blokirajuća, odnosno ne onemogućava korisniku korištenje jedne ili više funkcionalnosti i zato je njen prioritet srednji. Kako bi se ova greška popravila potrebno je korisniku prikazati određenu

poruku kako bi korisnik znao o čemu se radi. U ovom slučaju poruka bi korisniku trebala dati do znanja da nema niti jednog rezultata za njegovu pretragu.

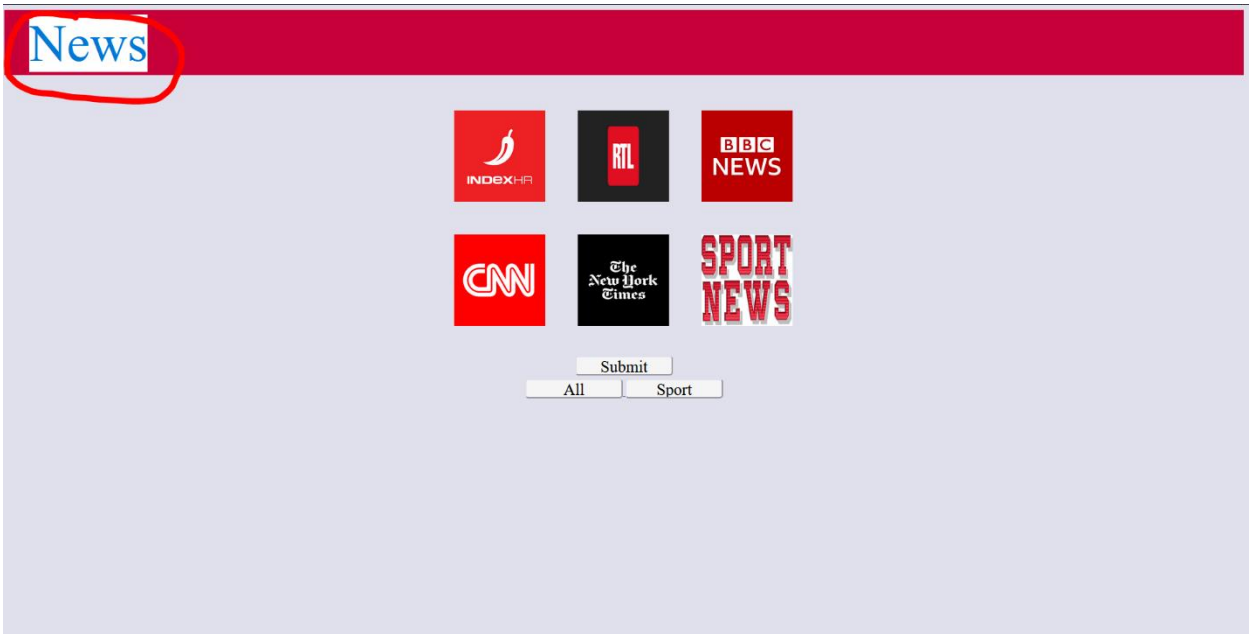
Greška nedostaje pretraživana riječ također nije blokirajuća greška (Slika 4.19). Javlja se također pri pretraživanju i otkrivena je tako da se pretraživala željena riječ, rezultati su se pojavili, korisnik je kliknuo na jednu od vijesti što ga je odvelo na vijest (korisnik je napustio stranicu), korisnik se klikom na nazad vratio na stranicu te mu više nisu bili prikazani rezultati pretraživanja nego sve vijesti s odabranih vjesnika. Iako ova greška nije blokirajuća uvelike utječe na korisnikov dojam o sustavu.

Greška nedostaje pretraživana riječ može se riješiti na više načina. U svrhu ova rada bit će objašnjena dva. Prvi način bi bio spremanje riječi za pretraživanje u kolačić. Kada bi se korisnik vratio s vijesti koju je čitao na stranicu bilo bi samo potrebno pročitati riječ iz kolačića i ponovno obaviti pretraživanje. Međutim, kako ovaj sustav pretraživanje obavlja na poslužiteljskoj strani ovaj način bi nepotrebno opteretio poslužitelja. Drugi način bi bio pamćenje stanja na stranici gdje bi se rezultati pretraživanja privremeno spremili te bi se povratkom na stranicu samo ponovno prikazali. Ovaj način je puno jednostavniji i manje opterećujući za sustav jer nije potrebna ponovna komunikacija s poslužiteljem jer su rezultati pretraživanja za tu riječ već obavljeni.

Slika 4.19. Greška nedostaje pretraživana riječ

Greška naslov se može označavati ima najmanji prioritet od sve tri greške zato što nije blokirajuća i samo je vizualnog karaktera (slika 4.20). Ova greška se javlja kada korisnik dva puta klikne na tekst naslova što rezultira označavanjem teksta. Ova greška se može lako popraviti dodavanjem CSS atributa *„user-select: none;“* što će korisniku onemogućiti odabiranje teksta naslova. Kada

bi se ovaj atribut isprobao, dalo bi se primijetiti da se greška ne bi popravila u svim preglednicima. Taj problem bi se riješio dodavanjem još nekoliko CSS atributa koji su prilagođeni pojedinom pregledniku.



Slika 4.20. Greška naslov se može označavati

4.4.3. Nefunkcionalno testiranje

Nefunkcionalno testiranje je vrsta testiranja koja odgovara na pitanje kako sustav radi. Ono testira performanse sustava, upotrebljivost sustava, sigurnost sustava, integritet podataka i drugo (slika 4.21) [23]. Iako je ranije nefunkcionalno testiranje bilo rijetko provođeno, danas ga se čak smatra najvažnijom vrstom testiranja. Razlog je jednostavan, ako primjerice sustav ne može poslužiti dovoljan broj korisnika te se korisnički zahtjevi ne izvršavaju u dovoljno kratkom vremenu, korisnici zasigurno neće biti zadovoljni upotrebljivošću sustava. Tada više nije važno rade li funkcionalni zahtjevi ispravno jer primjerice korisnik zbog preopterećenosti sustava više nije u mogućnosti pristupiti sustavu. Također je vrlo bitno da se sustav jednako ponaša na različitim operacijskim sustavima, da se sustav može oporaviti od kvara te da je sigurnost sustava i korisnika održana. U svrhu ovog rada proveden je test opterećenja stranice te će rezultati istog biti prikazani i razjašnjeni u sljedećem potpoglavlju.

Slika 4.21. Tipovi nefunkcionalnog testiranja [24]

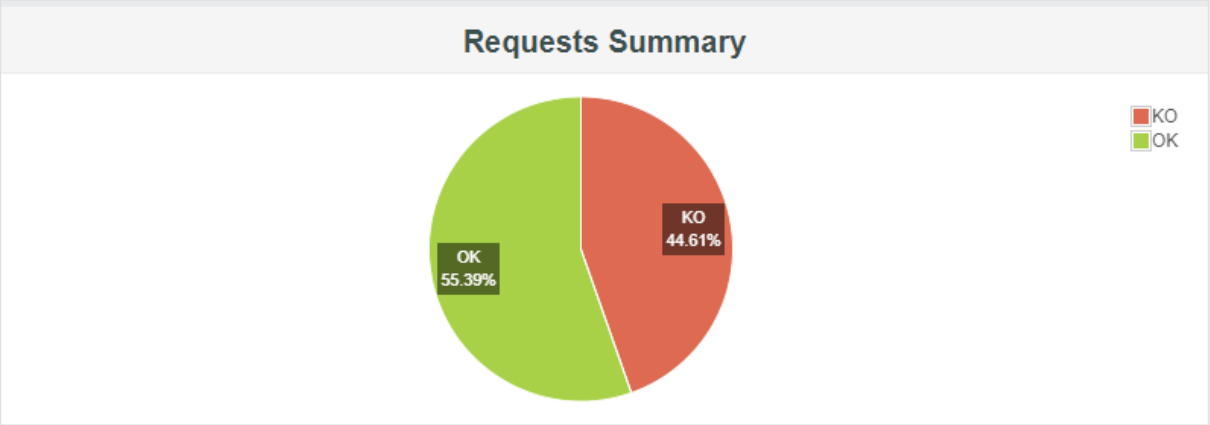
4.4.4. Rezultati nefunkcionalnog testiranja

Test opterećenja sustava proveden je s 1, 10, 80 i 400 korisnika. Prije provedbe testa opterećenja bilo je potrebno definirati testne scenarije i isti će biti navedeni u tablici 4.2. Također je navedeno vrijeme odziva za pojedini scenarij. Testiranje je provedeno pomoću tri različita alata, JMeter s proxy poslužiteljem, BlazeMeter i VisualStudio. U nastavku će biti prikazani i objašnjeni rezultati sva tri upotrijebljena alata. Rezultati provedbe testa pomoću proxy poslužitelja opterećenja nalaze se u tablici 4.2.

Tablica 4.2. Dobiveni rezultati s poslužiteljem Proxy [24]

Broj korisnika / Testni scenariji	1	10	80	400
Odabir jednog portala	0.9 sec	1.7 sec	3.2 sec	8.7 sec
Otvaranje više portala i pretraživanje	1.5sec	2.2 sec	6.7 sec	11.6 sec
Odabir opcije prikaza svih portala (ALL)	1.5 sec	2.3 sec	6.6 sec	9.1 sec
Odabir tri vijesti pa opciju svih portala (ALL)	0.9 sec	1.7 sec	2.2 sec	10.2 sec
Odabir tri vijesti pa opciju svih portala (ALL)	1.5 sec	2.3 sec	5.6 sec	10.4 sec
UKUPNO	6.3 sec	10.2 sec	24.3 sec	46 sec

Iako se na prvi pogled čini da je sustav dobro izdržao test opterećenja izvedenim sa proxy poslužiteljem, to ipak nije slučaj. Detaljnijim pregledom rezultata vidi se da između 40 i 50 posto zahtjeva uopće nije prošlo što se vidi na slici 4.22. Prikazani rezultati su za 10 korisnika istovremeno. Vrlo su slični za jednog, 80 i 400 korisnika.



Slika 4.22. Grafički prikaz uspješnosti zahtjeva korištenjem proxy poslužitelja [25]

Korištenjem alata BlazeMeter rezultati su puno realniji. Također uspješan broj poslanih zahtjeva je puno bolji. Tek s 400 korisnika BlazeMeter ima jednak broj neuspjelih zahtjeva kakav je proxy poslužitelj imao kroz cijelo testiranje. U tablici 4.3 mogu se vidjeti rezultati testa opterećenja korištenjem alata BlazeMeter.

Tablica 4.3. Dobiveni rezultati s BlazeMeter-om [25]

Broj korisnika	1	10	80	400
Testni scenariji				
Odabir jednog portala	7.4 sec	10.2 sec	8.9 sec	14.6 sec
Otvaranje više portala i pretraživanje	9.1 sec	18.1 sec	37.2 sec	31.2 sec
Odabir opcije prikaza svih portala (ALL)	7.3 sec	5.3 sec	8.8 sec	22.5 sec
Odabir tri vijesti pa opciju svih portala (ALL)	7.5 sec	8.4 sec	9.2 sec	21.7 sec
Odabir tri vijesti pa opciju svih portala (ALL)	6 sec	5.3 sec	8.7 sec	90.1 sec
UKUPNO	31.3 sec	47.3 sec	72.8 sec	180.1 sec

U ovom slučaju rezultati su puno realniji, jer je primjerice jednom korisniku potrebno malo više od 7 sekundi da pristupi stranici, odabere jedan portal i potvrdi formu. Za razliku od rezultata proxy poslužitelja gdje je jedan korisnik u istim uvjetima to napravio za 0.9 sekundi, što nije realna vrijednost. U tablici 4.4 prikazan je broj neuspjelih zahtjeva korištenjem BlazeMeter alata za 1, 10, 80 i 400 korisnika.

Tablica 4.4. Broj neuspjelih zahtjeva koristeći BlazeMeter alat

Broj korisnika	Broj neuspjelih zahtjeva u postocima
1	0
10	0
80	18.9
400	44.61

Iz tablice se jasno vidi da je sustav bez problema posluživao jednog i 10 korisnika. S 80 korisnika sustav je imao 18.9 posto neuspjelih zahtjeva, što znači da bi skoro petini korisnika bio onemogućen pristup djelu ili svim funkcionalnostima sustava.

Zadnji korišteni alat je VisualStudio. On ne daje rezultate u obliku kao JMeter već ih daje u obliku tablice (slika 4.23) i grafa (4.24).

Load Test Summary

Test Run Information

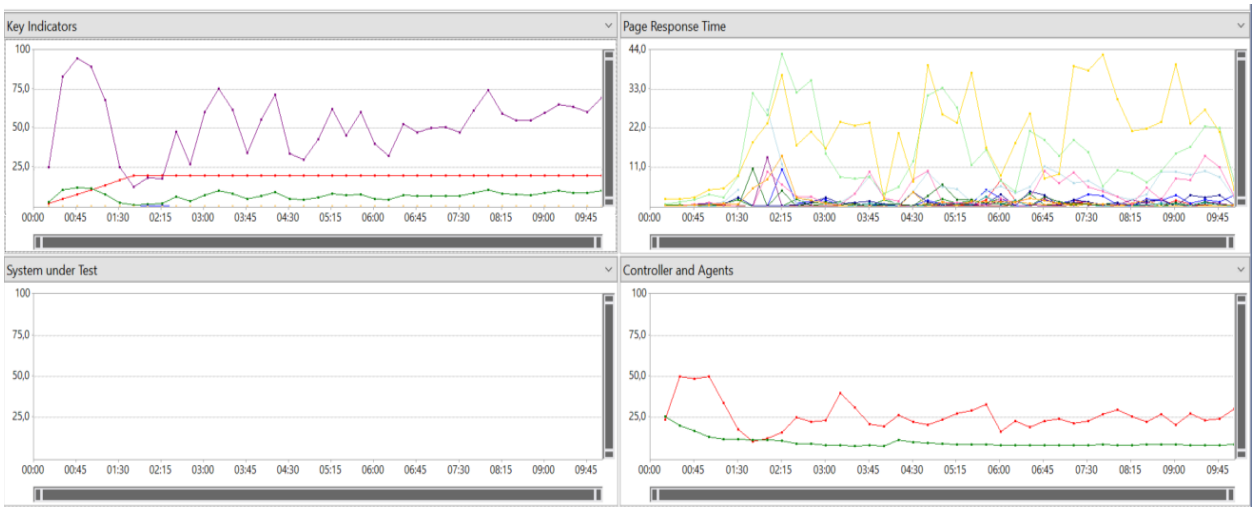
Load test name	LoadTest1
Description	
Start time	12.9.2020. 22:32:01
End time	12.9.2020. 22:42:01
Warm-up duration	00:00:00
Duration	00:10:00
Controller	Local run
Number of agents	1
Run settings used	Run Settings1

Overall Results

Max User Load	200
Tests/Sec	2,61
Tests Failed	1.566
Avg. Test Time (sec)	42,4
Transactions/Sec	0
Avg. Transaction Time (sec)	0
Pages/Sec	72,0
Avg. Page Time (sec)	2,00
Requests/Sec	242
Requests Failed	31.633
Requests Cached Percentage	43,8
Avg. Response Time (sec)	1,44
Avg. Content Length (bytes)	9.567

Slika 4.23. Tablični rezultati testa opterećenja iz alata VisualStudio [25]

Iz tablice se može iščitati vrijeme trajanja testa opterećenja, maksimalni broj korisnika, srednje vrijeme odgovora poslužitelja i drugo.



Slika 4.24. Grafički prikaz rezulata testa opterećenja iz alata *VisualStudio* [25]

Gledajući sva tri testa opterećenja da se zamijetiti da se povećanjem broja korisnika vrijeme odziva također povećava. Takvo ponašanje sustava je očekivano i normalno. Najbolje rezultate je dao alat BlazeMeter jer je imao najmanje neuspjelih zahtjeva. Zatim bi sljedilo alat VisualStudio. Nedostatak alata VisualStudio je taj što se zahtjevi potrebni za provedbu testa opterećenja mogu

snimiti samo pomoću preglednika Internet Explorer. Koristeći proxy poslužitelj dobiveni su najgori rezultati jer je veliki broj zahtjeva bio neuspješan.

Općenito, rezultati testa opterećenja su dobri s obzirom na to da sustav nije bio hostan na udaljenom poslužitelju nego je sustav koristio lokalne računalne resurse. Daljnjim povećavanjem broja korisnika došli bi do maksimuma opterećenja, odnosno postoji mogućnost da bi poslužitelj prestao funkcionirati i bilo bi ga potrebno ponovno pokrenuti ako se ne bi sam oporavio od kvara.

Kako bi se poboljšale performanse sustava bilo bi potrebno sustav hostati na udaljenom poslužitelju. Jedan od odabira je Amazon Web Services (AWS) [26] koji nudi razne dodatne mogućnosti kao što su prostor za pohranu podataka, strojno učenje i drugo. Jedna od prednosti ovakvog modela je skalabilnost. Povećanjem broja korisnika neće doći do prestanka rada poslužitelja već će se stranici alocirati dodatni resursi i ona će nastaviti normalno raditi.

5. NAČIN KORIŠTENJA SUSTAVA I ANALIZA REZULTATA

5.1. Opis načina rada sustava

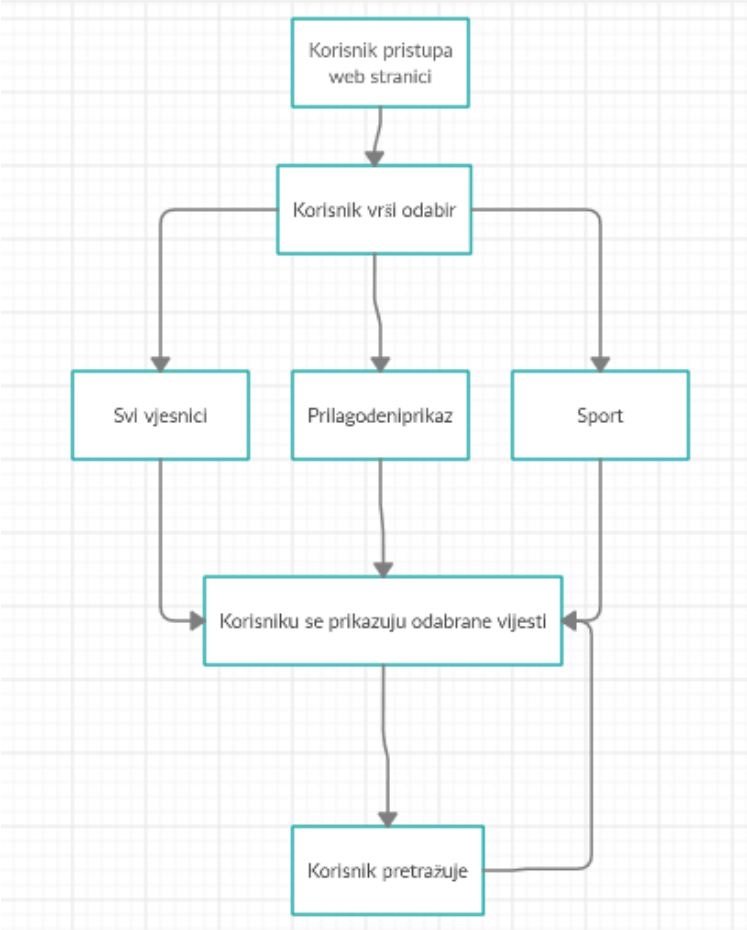
Kako je ranije rečeno, sustav se sastoji od dvije stranice. Prva, početna stranica, korisniku daje mogućnost biranja vjesnika. Korisnik može birati između pet vjesnika i vijesti iz sporta, a može odabrati sve klikom na gumb *all*. Također se može odabrati samo vijesti iz sporta. Početna stranica se može vidjeti na slici 5.1.

Slika 5.1. Početna stranica sustava za prilagodljivi višekriterijski prikaz vijesti iz različitih izvora

Nakon što korisnik odabere, sustav ga vodi na stranicu za prikaz vijesti (slika 5.2). Na stranici za prikaz vijesti korisnik može listati kroz vijesti iz odabranih vjesnika. Svaka vijest je prikazana u vlastitom okviru te se redom od gore prema dolje prikazuju naslov, opis i slika vijesti. Ako korisnik želi detaljnije pročitati vijest, klikom na sliku vijesti vodi ga se na izvornu vijest. Također postoji opcija pretraživanja vijesti gdje korisnik unosi podatak u polje za pretragu te potvrdom na uvid dobiva samo one vijesti koje u naslovu ili opisu sadrže traženi podatak.

Slika 5.2. Stranica za prikaz vijesti sustava za prilagodljivi višekriterijski prikaz vijesti iz različitih izvora

Na slici 5.3 dijagramom je prikazan tok programa koji je gore opisan.



Slika 5.3. Dijagram toka programa

5.2. Analiza rezultata razvoja i testiranja programske podrške

Nakon završenog razvoja sustav je testiran funkcionalno i nefunkcionalno i pokazao se kao pouzdan sustav. Korisnik može koristiti sve funkcionalnosti predviđene prototipom. Pa tako korisnik može odabirati vjesnike po želji i može pretraživati. Vijesti su prikazane sortirane po vjesnicima te su redom prikazane index, bbc, cnn, nytimes i na kraju rtl vijesti. Ako korisnik odabere index i nytimes vijesti će mu se prikazati upravo tim redom, index pa nytimes. Vijesti su također sortirane i po datumu objave te će se primjerice index vijesti prikazati od novijih ka starijima. Redoslijed je definiran na poslužiteljskoj strani i sortirane vijesti su proslijeđene korisničkoj strani. Sustav je moguće koristiti u svim internet preglednicima kao što su *Google Chrome*, *Firefox*, *Opera* i *Edge*. Sustav nije prilagođen za mobilne uređaje, ali to se po potrebi može lako ispraviti dodavanjem CSS pravila *media*. Kako se sustav trenutno nalazi na lokalnom računalu ne podržava istovremeno posluživanje većeg broja ljudi bez odgode. Trenutni maksimalni broj korisnika na stranici je 100 korisnika, za taj broj je odziv još uvijek vrlo dobar. Za veći broj korisnika sustav bi bio prespor i kao takav neupotrebljiv.

6. ZAKLJUČAK

Cilj ovog diplomskog rada je razvoj programskog rješenja web sustava za prilagođeni višekriterijski prikaz vijesti iz različitih izvora. Prilagođeni višekriterijski prikaz vijesti iz različitih izvora ostvaren je omogućavanjem korisniku odabir vjesnika sa kojih želi čitati vijesti te pretraživanjem istih. Za ostvarenje programskog rješenja korištene su sljedeće programske tehnologije: Python, Flask, HTML i CSS. Pomoću Pythona i Flaska razvijena je poslužiteljska strana sustava. Python je izabran, jer je pogodan za obradu stringova i ima puno biblioteka za rad s njima. Flask je izabran kao laki programski okvir koji programeru daje velik broj mogućnosti uz dovoljno dobro definiranu strukturu projekta. HTML i CSS su izabrani,jer je stranica statička i nema potrebe za nekim kompliciranijim alatima kao što su React ili Angular. Za vrijeme razvoja programskog rješenja došlo je do manjih izmjena dizajna stranice za prikaz vijesti u odnosu na prototip dizajna. Odabran je drugačiji prikaz vijesti zbog bolje preglednosti, odnosno prvotni dizajn nije bio prilagođen za osobe sa slabim vidom.

Razvijeni sustav je testiran funkcionalno i nefunkcionalno pri čemu su pronađena tri manja nedostatka. Nedostaci nisu bili visokog prioriteta i nisu utjecali na samu funkcionalnost sustava. Test opterećenja je pokazao da bi sustav na trenutnoj infrastrukturi mogao normalno posluživati oko 100 korisnika, a s većom odgodom i do 400, pa se može zaključiti da bi za optimalan rad sustava, sustav bilo potrebno postaviti na neku od ponuđenih usluga kako ne bi dolazilo do velikih odgoda ili ispada sustava. Također, bilo bi dobro sustav dodatno optimirati i smanjiti broj zahtjeva koje se šalju poslužitelju. To bi se moglo učiniti premještanjem pretraživanja s poslužiteljske strane na korisničku stranu što bi zahtjevalo korištenje dodatnih tehnologija kao što su JS i Ajax.

LITERATURA

[1] G. Paliouras, A. Mouzakidis, V. Moustakas, C. Skourlas; PNS: A personalized news aggregatoron the web; Department of Informatics, Tehnological Institute of Athens

[2] M. Aniche, C. Treude, I. Steinmacher, I. Wiese, G. Pinto, M.A. Storey, M. Aurélio Gerosa; How Modern News Aggregators Help Development Communities Shape and Share Knowledge; Delft University of Technology, University of Adelaide, Northern Arizona University Technological University of Paraná (UTFPR), Campo Mourão, University of Pará (UFPA), University of Victoria

[3] The Best News Aggregators, https://www.techlicious.com/guide/the-best-news-aggregation-sites/

Posjećeno: 2.7.2020.

[4] 9 Best News Aggregator Websites, https://www.wpbeginner.com/showcase/best-news-aggregator-websites-how-to-build-your-own/

Posjećeno: 21.9.2020.

[5] C. Grozea , D.C. Cercel , C. Onose , S. Trausan-Matu ; Atlas: News aggregation service; 2017 16th RoEduNet Conference: Networking in Education and Research

[6] Client-Server Model, https://www.geeksforgeeks.org/client-server-model/

Posjećeno: 2.7.2020.

[7] Napravljeno po uzoru na: https://techterms.com/img/lg/client-server_model_1253.png

Posjećeno: 2.7.2020.

[8] F. Hdioud, B. Frikh, B. Ouhbi, Multi-Criteria Recommender Systems based on Multi-Attribute Decision Making

[9] Visual Studio Code, https://code.visualstudio.com/

Posjećeno: 3.7.2020.

[10] Python, https://en.wikipedia.org/wiki/Python

Posjećeno: 3.7.2020.

[11] Flask, https://github.com/pallets/flask

Posjećeno: 4.7.2020.

[12] Flask (web framework), https://en.wikipedia.org/wiki/Flask_(web_framework)

Posjećeno: 4.7.2020.

[13] Flask templates, https://flask.palletsprojects.com/en/1.1.x/tutorial/templates/

Posjećeno: 13.9.2020.

[14] What is HTML, https://HTML.com/#What_is_HTML

Posjećeno: 5.7.2020.

[15] HTML Introduction, https://www.w3schools.com/HTML/HTML_intro.asp

Posjećeno: 5.7.2020.

[16] CSS Introduction, https://www.w3schools.com/CSS/CSS_intro.asp

Posjećeno: 5.7.2020.

[17] J. Ducket, HTML & CSS design and build websites, 2011, John Wiley & Sons, Inc., Indianapolis, Indiana

[18] Static files, https://flask.palletsprojects.com/en/1.1.x/tutorial/static/

Posjećeno: 14.9.2020.

[19] CSS Legal color values, https://www.w3schools.com/CSSref/CSS_colors_legal.asp

Posjećeno: 14.9.2020.

[20] Functional testing, https://softwaretestingfundamentals.com/functional-testing/

Posjećeno: 20.9.2020.

[21] Complete functional testing guide with its types and examples, https://www.softwaretestinghelp.com/guide-to-functional-testing/

Posjećeno: 20.9.2020.

[22] Napravljeno po uzoru na: https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2016/01/flow-chart.jpg

Posjećeno: 20.9.2020.

[23] A complete non-functional testing guide for beginers, https://www.softwaretestinghelp.com/what-is-non-functional-testing/

Posjećeno: 20.9.2020.

[24] Napravljeno po uzoru na: https://dlpng.com/png/7199648

Posjećeno: 20.9.2020.

[25] I. Maras, Funkcionalno testiranje i testiranje opterećenja web sustava prikaza vijesti, Diplomski rad, Fakultet elektrotehnike, računarstva i informacijskih tehnologija, 2020.

[26] Host a static website, https://aws.amazon.com/getting-started/hands-on/host-static-website/

Posjećeno: 20.9.2020.

SAŽETAK

U ovom diplomskom radu programski je ostvaren web sustava za prilagodljivi višekriterijski prikaz vijesti iz različitih izvora. Prilagodljivi višekriterijski prikaz vijesti omogućenje na način da korisnik sam bira vjesnike sa kojih želi čitati vijesti te tako sam kreira svoj prikaz. Korisnik također ima mogućnost pretraživanja što dodatno sustav čini prilagodljivijim. Web je razvijen koristeći Python, Flask, HTML i CSS. Sustav koristi model klijent-poslužitelj koji omogućuje da jedan poslužitelj poslužuje jednog ili više klijenata. Cilj je korisniku omogućiti odabir vjesnika s kojih želi vidjeti vijesti, prikazati mu vijesti prema odabranim kriterijima i omogućiti njihovo pretraživanje. Također, provedeno je funkcionalno i nefunkcionalno testiranje. Rezultat funkcionalnog testiranja su pronađene tri pogreške. Niti jedna od tri pogreške nije blokirajuća i ne onemogućuje korisniku korištenje svih funkcionalnosti sustava. Kao rezultat nefunkcionalnog testiranja dobiven je broj korisnika koji sustav može posluživati bez većih zastoja. Funkcionalni nedostaci ispravljaju se izmjenama u programskom kodu. Nedostaci opterećenja sustava mogu se ispraviti podizanjem sustava na neki od ponuđenih domaćina web sustava koji ima opciju prilagođene raspodjele resursa ovisno o opterećenju.

Ključne riječi Funkcionalno testiranje, nefunkcionalno testiranje, prilagodljivi višekriterijski prikaz vijesti, web stranica.

ABSTRACT

In this diploma thesis, a web system for adaptive multicriteria presentation of news from various sources has been programmatically realized. The customizable multi-criteria news view is enabled in such a way that the user chooses the newspapers with which he wants to read the news, thus creating his own view. The user also has the ability to search which makes the additional system customizable. The web was developed using Python, Flask, HTML and CSS. The system uses a client-server model that allows one server to serve one or more clients. The goal is to allow the user to select the newspapers from where he wants to see the news, show the news according to the selected criteria and enable their search. Also, functional and non-functional testing has been conducted. As a result of functional testing, three errors were found. None of the three errors is blocking error and does not prevent the use of any system functionalities. As a result of non-functional testing, a number of users were obtained who can operate the system without major downtime. Functional deficiencies are corrected by changes in the program code. Disadvantages of the system response can be corrected by hosting on some of the offered web system hosts that has the option of adaptive resource allocation depending on the load.

Keywords: Functional testing, non-functional testing, customizable multi-criteria news display, web page.

ŽIVOTOPIS

Filip Maras rođen je 03.03.1995 u Osijeku. U Čepinu završava osnovnu školu „Miroslav Krleža“ nakon koje 2010. upisuje Prvu gimnaziju u Osijeku. Od 2014. godine redovan je student na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer računarstvo. 2018. godine upisuje diplomski studij računarstva smjer Informacijske i podatkovne znanosti na fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI

Prilog 1: Diplomski rad u .pdf formatu

Prilog 2: Diplomski rad u .docx formatu

Prilog 3: Programsko rješenje u .py, .html i .css formatu