UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Magistrsko delo

Spletna aplikacija za hranjenje, urejanje in iskanje metapodatkov o spletnih povezavah

(Web application for storing, editing and searching of web link metadata)

Ime in priimek: Gregor Ambrožič
Študijski program: Računalništvo in informatika, 2. stopnja
Mentor: doc. dr. Peter Rogelj

Koper, januar 2017

Ključna dokumentacijska informacija

Ime in PRIIMEK: Gregor AMBROŽIČ

Naslov magistrskega dela: Spletna aplikacija za hranjenje, urejanje in iskanje metapodatkov o spletnih povezavah

Kraj: Koper

Leto: 2017

Število listov: 83 Število slik: 20 Število tabel: 6

Število prilog: 2 Število strani prilog: 8 Število referenc: 10

Mentor: doc. dr. Peter Rogelj

UDK: 004.774(043.2)

Ključne besede: spletna aplikacija, povezave, zaznamki, socialni zaznamki, poslovni zaznamki, hranjenje, urejanje, metapodatki

Izvleček:
Namen magistrskega dela je bil razviti sodobno spletno aplikacijo, ki bi služila kot shramba podatkov o spletnih povezavah s specifičnega področja zanimanja. Določeno področje uporabe lahko združuje podatke na spletu, ki so na zelo različnih lokacijah, zato je njihovo iskanje in urejanje težavno. Aplikacija kot organizirana zbirka metapodatkov rešuje ta problem. Poleg osnovnih podatkov lahko vsaka povezava vsebuje tudi poljubno hierarhično strukturo, s katero bolj natančno opišemo vsebino za povezavo. Uporabnik tako lažje najde podatke, ki ga zanimajo, nato pa s pomočjo povezave dostopa do dejanske vsebine. Ena izmed možnosti uporabe je hranjenje spletnih povezav do medicinskih primerov s področja radiologije. V sklopu magistrskega dela je razvita spletna aplikacija, kar vključuje funkcionalno in tehnično dokumentacijo, programiranje podatkovnega, strežniškega in odjemalnega dela aplikacije, programiranje avtomatiziranih testov aplikacije ter namestitev aplikacije na testnem strežniku.

Key words documentation

Name and SURNAME: Gregor AMBROŽIČ

Title of master thesis: Web application for storing, editing and searching of web link metadata

Place: Koper

Year: 2017

Number of pages: 83 Number of figures: 20 Number of tables: 6

Number of appendices: 2 Number of appendix pages: 8 Number of references: 10

Mentor: Assist. Prof. Peter Rogelj, PhD

UDK: 004.774(043.2)

Keywords: web application, web links, bookmarking, social bookmarking, enterprise bookmarking, storing, editing, metadata

Abstract: The purpose of paper is to develop a modern web application for storing web link metadata from a specific field of interest. A certain field of use might contain web data, that is distributed in various locations which makes searching and editing difficult. This application is solving the problem by functioning as an organized metadata collection. Besides general data every link can contain a configurable hierarchic structure used to more accurately describe the content linked. The user can find the data in an easier way and access it directly. One of the ways to use the application is storing web links of medical cases from the field of radiology. Through master thesis a web application was developed, including functional and technical documentation, programming of data, server and client parts of application, developing automated tests and installation on a test server.

Zahvala

Zahvaljujem se dr. Petru Roglju za mentorstvo in predvsem potrpežljivost.

Posebna zahvala gre mojemu dekletu in družini za neskončen vir spodbude in pozitivne energije.

Kazalo vsebine

Kazalo tabel

Kazalo slik

Kazalo prilog

Seznam kratic

API	programski vmesnik (application programming interface)
GUI	uporabniški vmesnik (graphical user interface)
JSON	standard (format) za izmenjavo podatkov (JavaScript Object Notation)
JDK	razvojna distribucija Jave (Java development kit)
MVC	razvojni vzorec model–pogled–kontroler (model-view-controller)
HTML	jezik za označevanje nadbesedila (hypertext markup language)
CSS	slogovni jezik za oblikovanje spletnih strani (cascading style sheets)
JDBC	Java programski vmesnik za dostop do podatkovnih baz (java database connectivity)
SQL	jezik za povpraševanje podatkovnih baz (structured query language)
JSP	tip spletnih predlog v Javi (java server pages)
HTTP	protokol za prenos spletnih podatkov (hypertext transfer protocol)
GET	tip HTTP zahtevka za branje podatkov
POST	tip HTTP zahtevka za pošiljanje podatkov

1 Uvod

V uvodu bomo opredelili problem, ki ga rešujemo, in opisali namen magistrskega dela.

1.1 Opredelitev problema

Informacije so na spletu dosegljive iz različnih virov in skupaj tvorijo bolj ali manj zaokrožene sklope, vezane na specifično tematiko. Uporabnik mora tako sam skrbeti, da organizira svojo množico spletnih povezav na podatke z izbranega področja uporabe. Kot primer naj navedemo medicinske podatke, ki so, predvsem v izobraževalne namene, dosegljivi v različnih spletnih aplikacijah (nekaj primerov je navedenih na [1]). Vsaka izmed aplikacij ima malo drugačen način organizacije in prikaza podatkov. Bolj ko je področje uporabe specifično, težje je organizirati relevantne povezave do gradiv, saj v tem primeru tudi spletni iskalniki ne nudijo zadostne možnosti omejitve na specifično področje zanimanja.

Iščemo torej rešitev, ki bi olajšala pregledovanje in spremljanje objav na specifičnih področjih, v obliki zaokrožene množice povezav na različne spletne vire. Pri tem izhajamo iz problemskega področja radioterapije, kjer bi želeli spremljati objave radioterapevtskih slikovnih podatkov, opremljenih z dodatnimi razlagami, kot so na primer opisi in segmentacije anatomskih struktur.

1.2 Namen magistrskega dela

Namen magistrskega dela je razviti spletno aplikacijo Linkpoint, ki bo služila kot shramba podatkov o spletnih povezavah s specifičnega področja zanimanja. Spletne povezave so lahko povezave na različne spletne strani, zato bodo njihovi podatki shranjeni zelo splošno. Vsaka povezava mora biti opremljena z osnovnimi podatki o ciljni strani, lastniku, imenom, opisom, oznakami in predstavitveno sliko ter hierarhično opisno strukturo, ki opisuje vsebino ter je prilagojena področju.

2 Analiza zahtev

Aplikacija mora podpirati shranjevanje, branje, iskanje spletnih povezav po lastnostih ter po elementih hierarhične strukture. Na ta način so podatki standardno organizirani in bolje razumljivi. Vnesena povezava je lahko tudi poenostavljena in vsebuje samo osnovne podatke.

Za razvoj je potrebno uporabiti zgolj odprtokodne rešitve. Aplikacija mora večjemu številu uporabnikov omogočati hkratno delo. Iskanje podatkov mora biti uteženo, tako da so rezultati razvrščeni po pomembnosti glede na število pojavitev iskanega niza ter njihove lokacije. Vsa gesla v aplikaciji morajo biti enosmerno kriptirana z enim od standardnih algoritmov. Aplikacija mora podpirati dva načina dela – z registracijo/prijavo ter brez.

2.1 Neprijavljen uporabnik

Za neprijavljenega uporabnika bo na voljo pregled vseh javno dostopnih podatkov, to je podatkov o spletnih povezavah in splošnih podatkov o lastniku povezave. Poleg tega bo imel neprijavljen uporabnik možnost iskanja povezav. Neregistrirani uporabniki bodo tako lahko dostopali do aplikacije le preko spletnega vmesnika, medtem ko bodo imeli registrirani uporabniki še dodatno možnost uporabe programskega vmesnika. V splošnem bosta imela oba vmesnika enake funkcije (branje in urejanje podatkov), s tem da bo spletni vmesnik vseboval še iskanje po vsebini. Spletni vmesnik bo tako služil predvsem pregledu in urejanju manjše količine podatkov, medtem ko bo programski vmesnik namenjen branju in vnosu večje količine podatkov. Namen spletnega vmesnika je omogočiti, da lahko spletna mesta s specifičnega področja uporabe samodejno objavljajo svoje povezave. Spletni vmesnik bo zaščiten z uporabo varnostnega gesla, programski vmesnik pa z naključno generiranim ključem.

2.1.1 Tok dogodkov

Pred prijavo v sistem ima vsak uporabnik možnost pregleda javno dostopnih podatkov. Neprijavljen uporabnik lahko izvaja naslednje aktivnosti:

Registracija uporabnika Uporabnik mora ob registraciji vnesti ime, naslov elektronske pošte in geslo. Po pravilnem vnosu prejme elektronsko sporočilo s povezavo,

s katero aktivira svoj račun. Po aktivaciji se lahko prijavi v aplikacijo.

Prijava Uporabnik mora ob prijavi vnesti naslov elektronske pošte in geslo. Po pravilnem vnosu (uporabnik z naslovom elektronske pošte še ne obstaja, vsa polja so izpolnjena) in preverjanju gesla lahko uporabnik uporablja dodatne funkcionalnosti (glej razdelek 2.2).

Pregled povezave Uporabnik lahko pregleduje podrobnosti vsake povezave – ime, opis, sliko, priloge, oznake. Podatkov ne more spreminjati. Pri vsaki povezavi lahko uporabnik vidi tudi lastnika povezave.

Pregled uporabnika Uporabnik lahko pregleda osnovne, javne podatke drugega registriranega uporabnika aplikacije na podlagi njegovih objav. Vidi lahko njegovo ime, sliko in opis ter pripadajoče povezave. Nobenega od podatkov ne more spreminjati.

Iskanje Uporabnik vnese iskalno geslo in prikažejo se mu vse najdene povezave, ki ustrezajo iskalnemu geslu. Povezave, ki so bolj relevantne, se pokažejo na začetku. Na relevantnost vpliva mesto pojavitve iskalnega gesla v podatkih povezave (najbolj pomembno je ime povezave, nato oznake, nato elementi drevesne strukture in nazadnje opis) in število pojavitev iskalnega gesla v podatkih povezave.

2.2 Prijavljen uporabnik

Za registrirane in prijavljene uporabnike bo poleg omenjenih funkcionalnosti omogočena še možnost dodajanja povezav in opisov svojih lastnih podatkov in urejanje teh povezav.

2.2.1 Tok dogodkov

Prijavljen uporabnik lahko izvaja vse akcije neprijavljenega uporabnik z izjemo prijave v sistem in registracije.

Pregled in urejanje lastnih podatkov Uporabnik lahko pregleda in ureja svoje lastne javne (ime, slika, opis) ter tajne podatke, ki so ostalim in neprijavljenim skriti. Med tajne podatke spadata naslov elektronske pošte in ključ za dostop prek programskega vmesnika.

Dodajanje, urejanje in brisanje povezav Uporabnik lahko doda neomejeno število novih povezav. Vsako izmed svojih povezav lahko tudi ureja. Spreminja

lahko ime, URL, opis, sliko, oznake ter ureja strukturo podatkov v okviru hierarhije tipov. Če pogledamo specifičen primer uporabe v radiologiji, lahko uporabnik npr. doda nov obris "delineation" in podobno. Katero koli izmed svojih povezav lahko uporabnik izbriše.

2.3 Področja uporabe

Za lažjo predstavo bomo predstavili delovanje aplikacije na področju radioterapije, kljub temu da je aplikacija načrtovana splošno in jo lahko uporabimo tudi na poljubnem drugem področju uporabe. Aplikacija Linkpoint v radiologiji predstavlja zbirko podatkov o medicinskih primerih. V ta namen bo hierarhija dodatnih podatkov povezave prilagojena ključnim podatkom tega področja, torej opisom medicinskih primerov. Tipi hierarhičnih podatkov bodo slike, segmentacije struktur, dodatni slikovni sloji in priloge. Ostali podatki bodo generično obravnavani kot priloge.

2.3.1 Tipi podatkov

Linkpoint je aplikacija, zasnovana kot zbirka podatkov o povezavah na splošno. Za čim boljši opis podatkov, prisotnih na spletni povezavi, uporablja hierarhično strukturo, pri čemer ima vsak element v hierarhiji tip in ime. V naslednjih razdelkih bomo navedli nekaj primerov tipov podatkov.

2.3.2 Primer hierarhične strukture v radiologiji

Linkpoint lahko podpira hranjenje podatkov o medicinskih primerih s področja radiologije. Ti so označeni s pripadajočimi identifikacijskimi številkami, ki jih bomo uporabili v demonstracijski različici aplikacije:

1 case;

2 image;

3 structure folder;

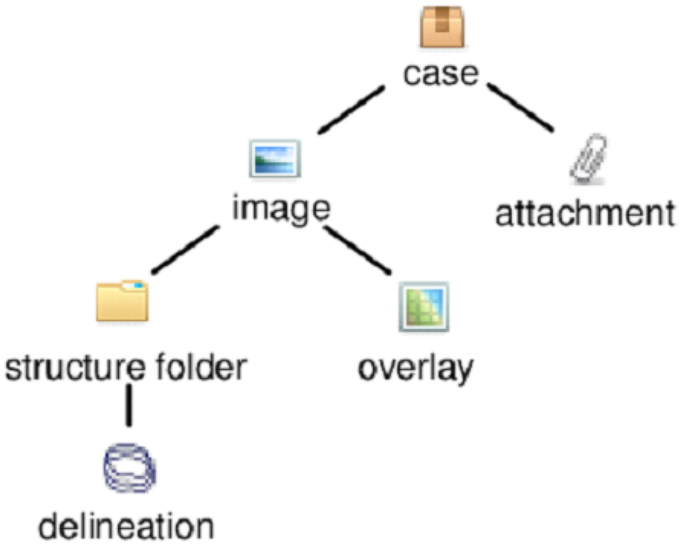
4 delineation;

10 overlay;

100 attachment.

Navedeni tipi podatkov so hierarhično urejeni kot prikazuje naslednja slika – npr. tip "case" lahko vsebuje samo podtipa "image" in "attachment".

Slika 1 prikazuje hierarhijo tipov podatkov.



Slika 1: Tipi podatkov (vir slike: [8]).

Razlaga uporabljenih tipov podatkov v radiologiji

Case Medicinski primer, zbirka podatkov o posameznem pacientu. Opis primera lahko shranimo v opis povezave, saj je v primeru radioloških slik lahko primer na povezavi samo en.

Image Medicinska slika, del medicinskega primera. Vsak medicinski primer lahko vsebuje več medicinskih slik, ki so lahko zajete z različnimi slikovnimi tehnikami, različnimi projekcijami, v različnem času ali pod kakšnimi drugimi različnimi pogoji.

Overlay Maska medicinske slike, to je polje vrednosti za vsak slikovni element slike, ki podaja neko dodatno informacijo o vsebini slike ali rezultatih obdelav in analiz vezanih na to sliko. Namen mask je lahko zelo različen, od vrisovanja za zdravljenje pomembnih podatkov, do prikaza rezultatov različnih analiz, kot na primer prejeta doza obsevanja v radioterapiji ali ovrednotenje pričakovanih razlik pri različnih medicinskih posegih.

Structure folder logična enota zbirke obrisov. Tipično gre za obrise iste (anatomske ali funkcionalne) strukture na sliki. V primeru, ko imamo več segmentiranj iste slike, bi to lahko predstavili v skupni strukturni mapi (structure folder), v kateri bi bila dva obrisa. Uporabimo jo lahko tudi za druge načine logičnega ločevanja med različnimi objekti.

Delineation Obris (anatomske ali funkcionalne) strukture na sliki. Obrisi tipično sestojijo iz kontur, ki so vezane na posamezne rezine slike, skupaj pa predstavljajo 3D področje posamezne strukture na sliki. Ime obrisa določuje pomen obrisa. Obrisi tipično označujejo področja posameznih organov oziroma področja, ključna za načrtovanje zdravljenja ali analizo zdravstvenega stanja.

Attachment Priloga v obliki datoteke v poljubnem datotečnem formatu, ki dopolnjuje slikovne podatke, npr. v obliki predstavitev, dokumentov, člankov, video posnetkov in podobno.

Primer opisa povezave na medicinski primer v radiologiji

Tukaj bomo z LinkPoint modelom opisali primer, ki je dostopen na [9]. Stran je brezplačno dostopna z registracijo uporabnika. Primer podatkov:

URL povezave http://econtour.org/cases/34

Ime povezave 65 y/o M with intermediate risk prostate cancer (eContour case 34)

Opis povezave He was treated with IMRT alone. The prostate and low SV received 79.2 Gy in 44 fractions. Patient was instructed to have a comfortably full bladder and empty rectum for treatment. IGRT included daily CBCT ...

Slika povezave (glej sliko 2)

Lastnik povezave econtour.org

Hierarhična struktura :

- Case 65 y/o M with intermediate risk prostate cancer

- - Image GTV (Prostate)

- - - Structure folder OARs

- - - - Delineation Bladder

- - - - Delineation Body

- - - - Delineation Penile Bulb

- - - - Delineation Rectum SV (proximal 1cm)

- - - Structure folder Targets

- - - - Delineation GTV (Prostate)

- - - - Delineation CTV 79.2

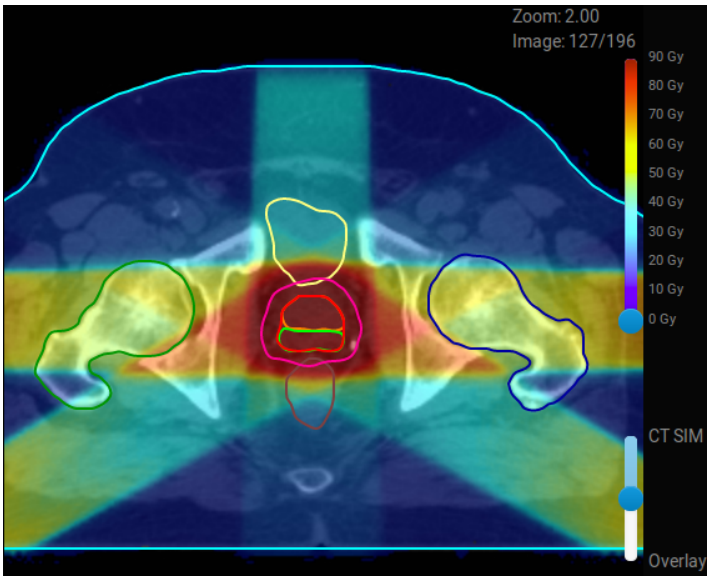
- - - - Delineation PTV 79.2

- - - Structure folder Anatomy

- - - - Delineation Ischial Tuberosity

- - - Overlay RT Dose

- - Attachment Step-by-step contouring guide



Slika 2: Primer slike, ki ilustrira povezavo na medicinski primer v radiologiji (vir slike: [9]).

2.3.3 Primer hierarhične strukture pri filmih

Linkpoint bi lahko bil uporabljen tudi za hranjenje podatkov o filmih. Na osnovnem nivoju lahko navedemo ime režiserja (tip "režiser") in scenarista (tip "scenarist") ter leto izida (tip "leto"). Nato definiramo nov tip "igralec" za glavne igralce. Vsak igralec bi lahko imel podtip "lik" z liki, ki so jih igrali.

Povezava na primer filma

Tukaj bomo z LinkPoint modelom opisali film, ki je dostopen na [10]. Stran je brezplačno dostopna. Primer podatkov:

URL povezave http://www.imdb.com/title/tt0133093

Ime povezave The Matrix (IMDb)

Opis povezave A computer hacker learns from mysterious rebels about the true nature of his reality and his role in the war against its controllers.

Lastnik povezave IMDb

Hierarhična struktura :

- **Režiser** Lana Wachowski, Lilly Wachowski

- **Scenarist** Lana Wachowski, Lilly Wachowski

- **Leto** 1999

- **Igralec** Keanu Reeves

- - **Lik** Neo

- - **Lik** Mr Anderson

- **Igralec** Laurence Fishburne

- - **Lik** Morpheus

- **Igralec** Carrie-Anne Moss

- - **Lik** Trinity

2.4 Pregled obstoječih rešitev

Organizirano shranjevanje podatkov o spletnih povezavah je prisotno že dalj časa. Ena izmed prvih tovrstnih spletnih storitev za shranjevanje zaznamkov je bila itList [2]. Uporabnikom je omogočala shranjevanje, urejanje in organiziranje zaznamkov, ki jih je bilo mogoče tudi deliti.

S spletno aplikacijo Delicious [3] se je področje bolj razvilo, pojavil se je tudi izraz socialni zaznamek ("social bookmark") [4]. Poleg organizacije zaznamkov jih je bilo mogoče tudi opremiti z oznakami, ki pa so poljubne in ne sledijo strogemu redu označevanja – različni uporabniki lahko določeno povezavo različno označijo, oznake niso predpisane.

Iz socialnih zaznamkov so se razvili poslovni zaznamki ("enterprise bookmark") [5], kar je prenos shranjevanja in označevanja zaznamkov na poslovno področje, kjer podjetja lahko na enem mestu zberejo podatke o vsebinah, dostopnih na distribuiranih podatkovnih bazah ali datotečnih strežnikih. Primera takih aplikacij sta ApexKB [6] in Knowledge Plaza [7].

Odločili smo se za razvoj nove aplikacije, saj med najdenimi nismo našli take, ki bi bila odprtokodna ter podpirala definicijo hierarhične strukture za boljšo definicijo povezave ter možnost povezave aplikacije s podatkovnimi skladišči na danem področju uporabe, preko dodatnega programskega vmesnika.

2.5 Obseg

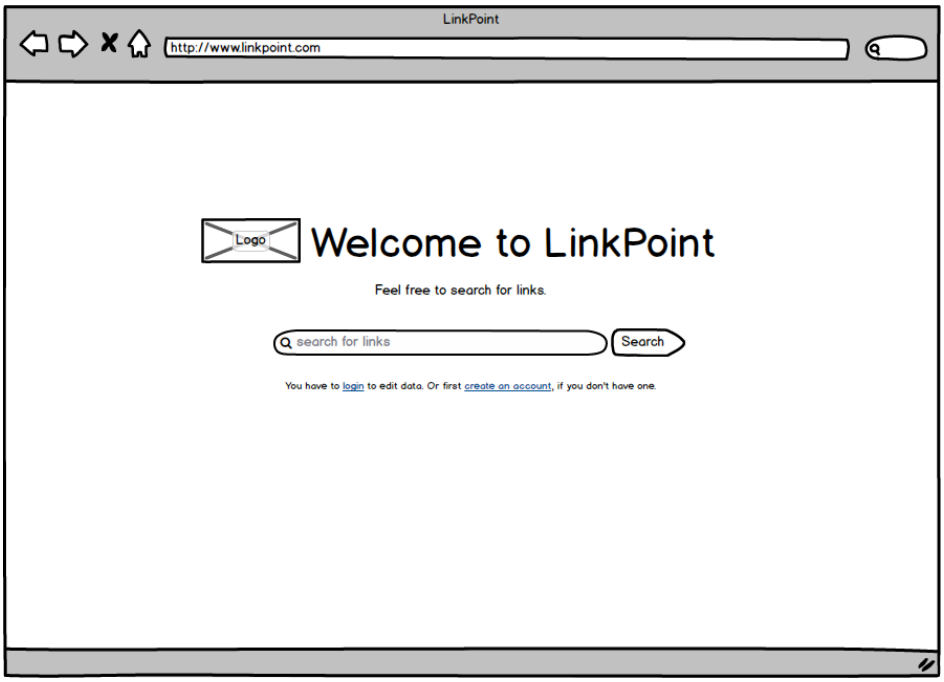
V sklopu magistrskega dela bomo razvili spletno aplikacijo, kar vključuje funkcionalno in tehnično dokumentacijo, implementacijo sistema, testno dokumentacijo, navodila za namestitev ter namestitev aplikacije na testnem strežniku.

2.6 Zahteve uporabniškega vmesnika

Spletni vmesnik z uporabo brskalnika bo namenjen končnim uporabnikom, ki želijo ročno dostopati do podatkov. Naslednji razdelki vsebujejo skice uporabniškega vmesnika, ki so nastale kot eden izmed produktov faze načrtovanja.

2.6.1 Domača stran

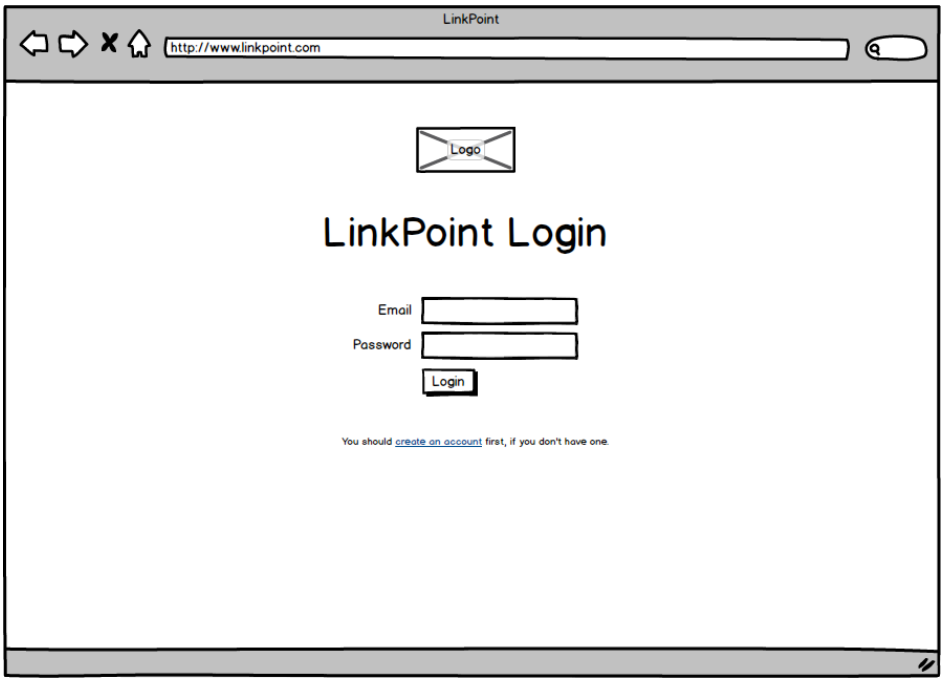
Je prva stran, ki se prikaže v spletnem vmesniku. Vsebuje iskalno polje, ki išče po celotni vsebini javnih podatkov. Glej sliko 3.



Slika 3: Domača stran.

2.6.2 Stran za prijavo

Stran se prikaže, ko uporabnik izbere možnost prijave v sistem. Če je prijava neuspešna, mora sistem uporabnika obvestiti. Vsebuje polji za vnos uporabniškega imena in gesla, gumb za prijavo ter prostor za izpis morebitnega sporočila o napaki. Pod polji je povezava do strani za registracijo novega uporabnika. Slika 4 prikazuje stran za prijavo.



Slika 4: Stran za prijavo.

2.6.3 Stran za kreiranje novega uporabnika

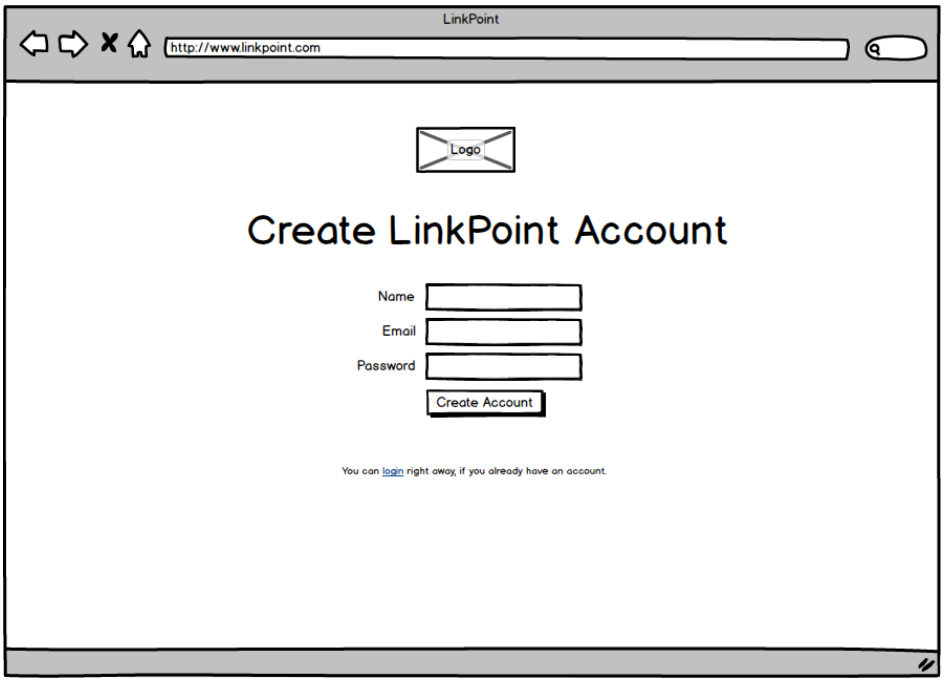
Stran se prikaže, ko uporabnik izbere možnost za kreiranje oz. registracijo novega uporabnika aplikacije. Potrebno je vnesti ime, naslov elektronske pošte in geslo ter potrditi. V primeru, da pride do napake, se sporočilo izpiše na isti maski. Slika 5 prikazuje stran za kreiranje novega uporabnika. Pri kreiranju uporabnika se temu generira tudi API ključ, ki je viden pri ogledu lastnega profila.

2.6.4 Osnovna stran prijavljenega uporabnika

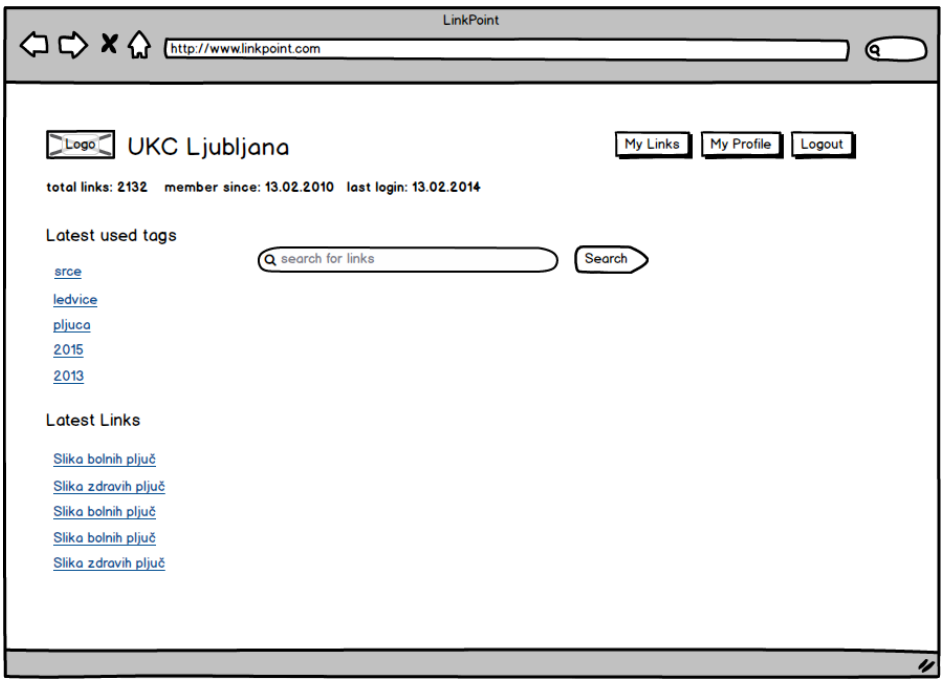
Stran se prikaže po uspešni prijavi. Uporabnik ima na voljo bližnjice do svojih povezav, iskanja in urejanja lastnih podatkov. Prikazane so zadnje uporabljene oznake in zadnje dodane povezave. Slika 6 prikazuje osnovno stran prijavljenega uporabnika.

2.6.5 Stran s podatki o uporabniku

Stran vsebuje vse javno dostopne podatke o uporabniku aplikacije. Na desni strani je seznam zadnjih dodanih povezav.

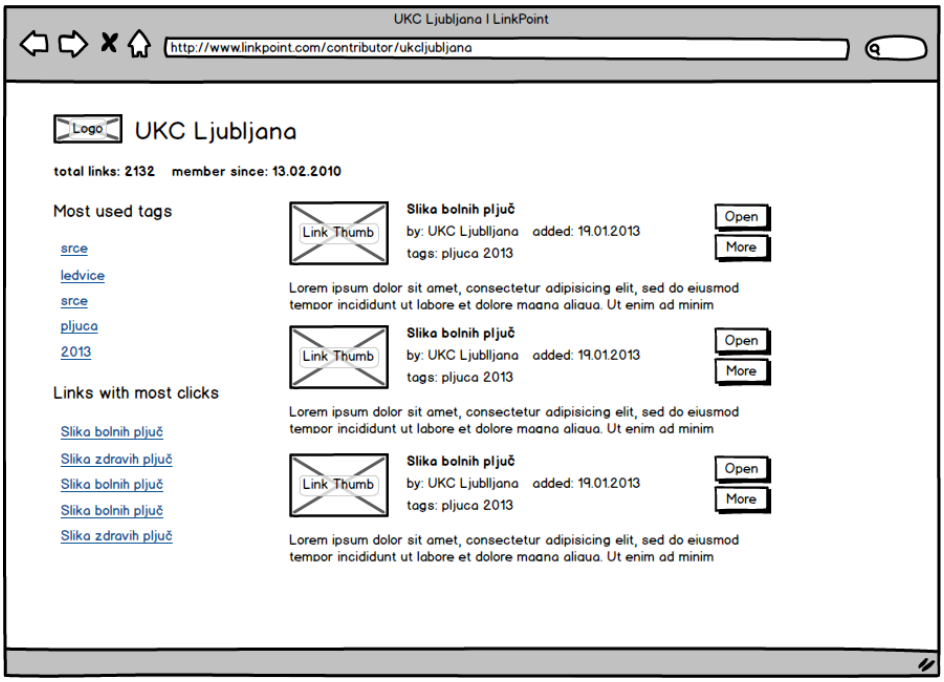


Slika 5: Stran za kreiranje novega uporabnika.



Slika 6: Osnovna stran prijavljenega uporabnika.

Na levi strani so uporabnikove oznake in povezave, ki so najbolj pogosto uporabljane. Zgoraj so navedene oznake (tags) največkrat prikazanih povezav, spodaj pa največkrat prikazane povezave. Slika 7 prikazuje stran s podatki o uporabniku.



Slika 7: Stran s podatki o uporabniku.

2.6.6 Stran z rezultati iskanja

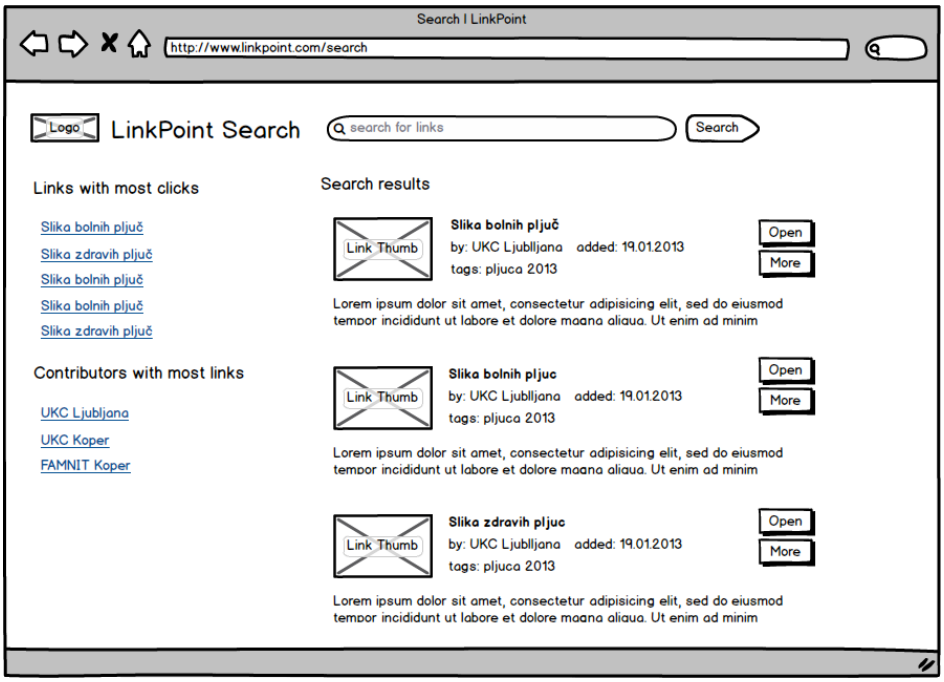
Ko uporabnik vnese iskalno vsebino, se na tej strani prikaže seznam najdenih povezav. Če je uporabnik prijavljen, se poleg javnih povezav prikažejo tudi njegove lastne. Na vrhu je iskalno polje in gumb, s katerim lahko uporabnik zažene novo iskanje. Slika 8 prikazuje stran z rezultati iskanja.

2.6.7 Stran s podrobnostmi povezave

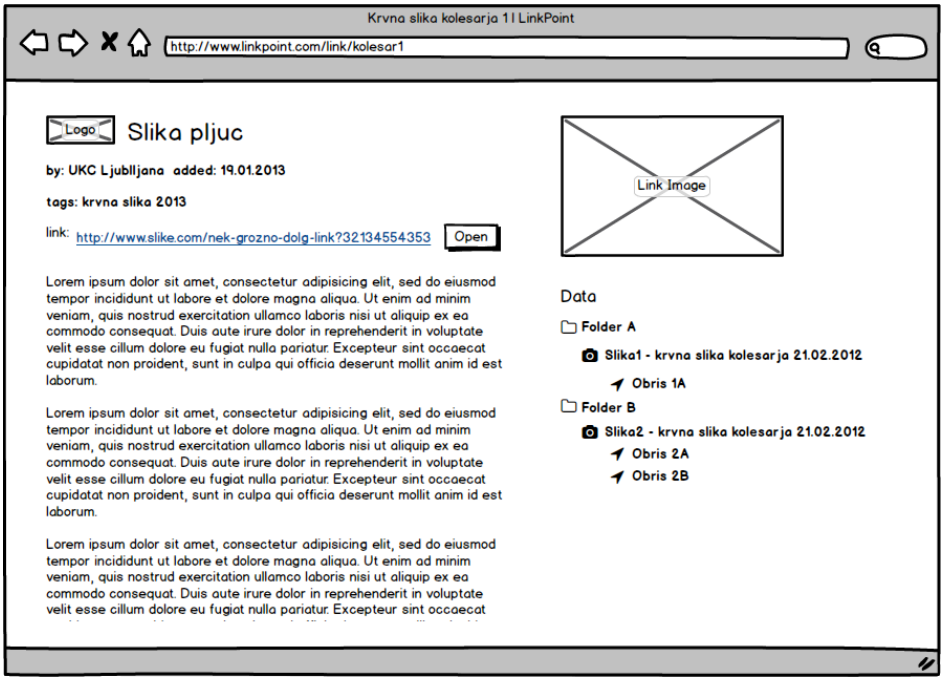
Prikazuje podrobne podatke o posamezni povezavi. Vidni so slika povezave, opis in ostali podatki. Pod osnovnimi podatki je opis linka. Na desni strani je večja slika, ki predstavlja vsebino za linkom. Pod sliko je drevesna struktura, ki opisuje hierarhijo podatkov za linkom. Podatki so lahko različnih tipov, vsak je označen z ikono tipa ter imenom ali opisom. Slika 9 prikazuje stran s podrobnostmi povezave.

2.6.8 Stran za vnos povezave

Stran je podobna strani za urejanje podatkov, na začetku je prazna. Na voljo je samo prijavljenemu uporabniku. Slika 10 prikazuje stran za vnos povezave.



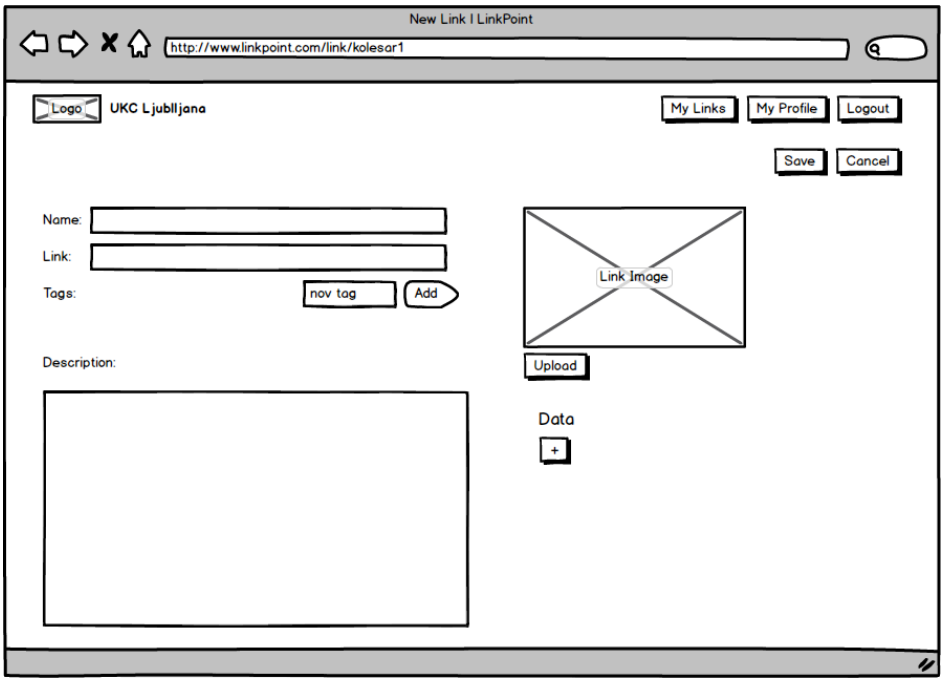
Slika 8: Stran z rezultati iskanja.



Slika 9: Stran s podrobnostmi povezave.

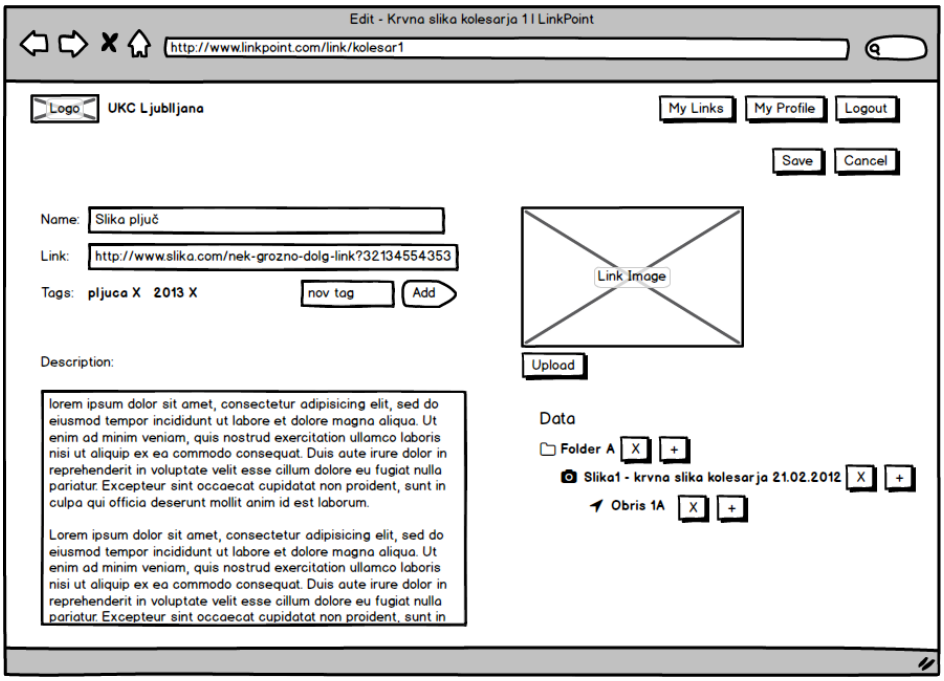
2.6.9 Stran za urejanje povezave

Je podobna strani za prikaz podatkov, s tem da vsebuje vnosna in izbirna polja za vse atribute povezave. Na voljo je samo prijavljenemu uporabniku, ki pregleduje svojo



Slika 10: Stran za vnos nove povezave.

povezavo. Slika 11 prikazuje stran za urejanje povezave.



Slika 11: Stran za urejanje obstoječe povezave.

2.7 Nefunkcijske zahteve

V tem razdelku bomo opisali nefunkcijske zahteve za izdelavo aplikacije. Ta morajo podpirati določene varnostne standarde in uporabljati odprtokodne programske rešitve.

2.7.1 Varnost

Uporabniki lahko dostopajo in urejajo samo svoje lastne podatke. Do javnih podatkov drugih uporabnikov imajo samo dostop za branje. V primeru spletnega vmesnika so uporabnikovi podatki zaščiteni z geslom, pri programskem vmesniku pa z API ključem. Uporabnik lahko z uporabo API ključa bere in ureja podatke samo o svojih povezavah, ne more pa urejati lastnih podatkov (ime, naslov elektronske pošte, geslo, API ključ).

Vsa gesla v aplikaciji so shranjena v šifrirani obliki MD5. Za varnostne potrebe skrbi ločen modul aplikacije.

2.7.2 Uporaba odprtokodne programske opreme

Aplikacija naj zaradi omejevanja stroškov uporablja odprtokodno tehnologijo. Kljub temu je potrebno izbrati programsko opremo, ki ima dovolj veliko bazo uporabnikov ter je dovolj preizkušena.

2.7.3 Jezik

Aplikacija naj zaradi širše dostopnosti vsebuje uporabniški vmesnik v angleškem jeziku. Izbrana tehnologija za implementacijo naj omogoča enostavno prevajanje v druge jezike po potrebi.

3 Primeri uporabe

Opisi primerov uporabe so ločeni glede na vrsto uporabnika. Uporabnik je lahko GUI uporabnik, ki je človeški uporabnik s spletnim brskalnikom, ali pa API uporabnik – sistem, ki uporablja JSON vmesnik, kar je namenjeno možnosti avtomatizirane objave podatkov drugih podatkovnih zbirk.

Slika 12 prikazuje diagram primerov uporabe aplikacije s strani GUI (uporabnik s spletnim brskalnikom) in API (sistem z uporabo programskega vmesnika) uporabnika.

3.1 Primeri uporabe GUI uporabnika

3.1.1 Iskanje povezav

Akterji

- GUI Uporabnik

Zahteve

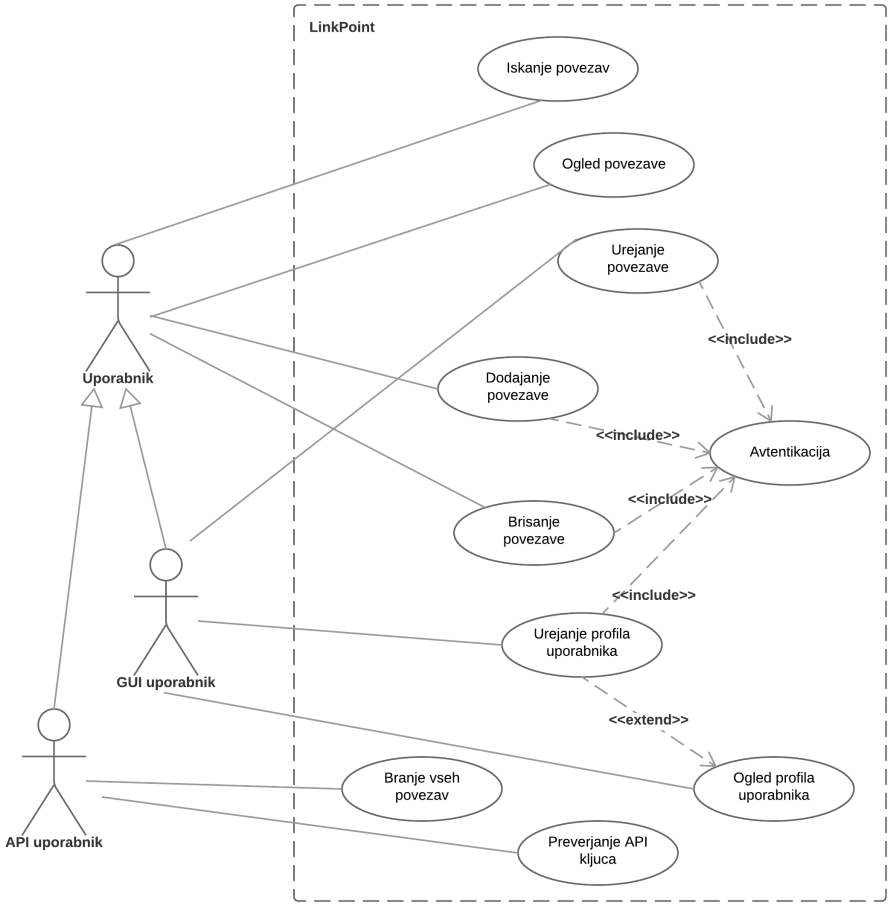
Uporabnik želi poiskati povezave glede na njihovo ime, opis, oznako ali ime uporabnika.

Sprožilec

Uporabnik odpre stran za iskanje s klikom na "Search" v orodni vrstici.

Osnovni tok dogodkov

1. Odpre se podstran za iskanje, kjer je aktivno polje za vnos iskalnih parametrov.
2. Uporabnik vnese iskalne pogoje in potrdi s tipko "enter" ali klikom na gumb "Search" pod vnosnim poljem.
3. Sistem najde in prikaže seznam povezav. Stran za iskanje se dopolni z rezultati iskanja, ki so razvrščeni po pomembnosti (najprej povezave, ki so bile najdene po imenu/opisu povezave, nato povezave, najdene po oznaki, ter nazadnje povezave, najdene po imenu uporabnika). Vsak rezultat iskanja vsebuje sliko za ilustracijo



Slika 12: Diagram primerov uporabe.

povezave, ime povezave, ime uporabnika ter gumb "Details", ki vodi do podrobnosti povezave. Nad seznamom povezav je oznaka "Link count", ki prikazuje število najdenih povezav.

4. Tok dogodkov se uspešno zaključi.

Alternativni tokovi dogodkov

Ni najdenih rezultatov:

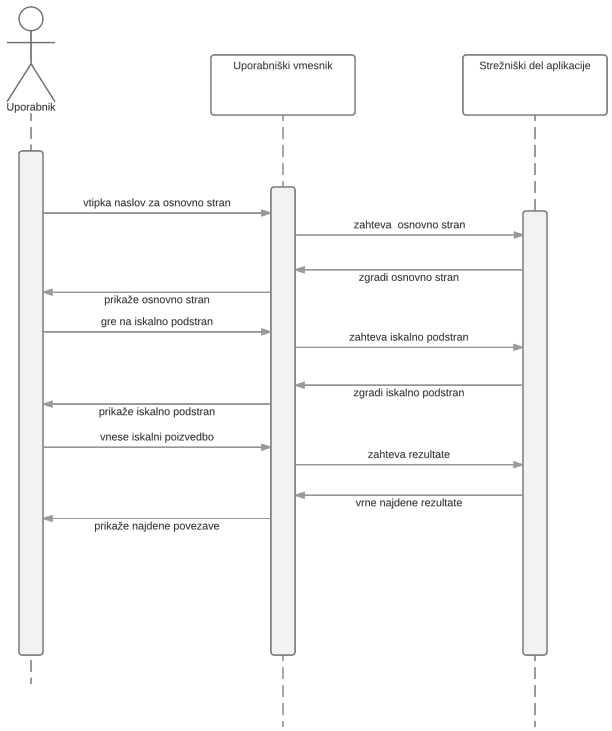
3.a1 Sistem za iskalne parametre ne najde rezultatov.

3.a2 Pod iskalnim poljem se izpiše obvestilo, da noben rezultat ne ustreza iskalnim pogojem.

Stanje po zaključku

Uporabnik lahko izbere katero od najdenih povezav in odpre njene podrobnosti. Uporabo aplikacije lahko nadaljuje tudi z izbiro drugih stalnih možnosti (prehod na začetno stran aplikacije, urejanje profila, vnos povezav ...). Podatki v podatkovni bazi se ne spremenijo.

Sekvenčni diagram



Slika 13: Sekvenčni diagram iskanja povezav.

3.1.2 Ogled povezave

Akterji

- GUI Uporabnik

Zahteve

Uporabnik želi pregledati podrobnosti o določeni povezavi.

Sprožilec

Uporabnik ima odprt seznam povezav. Lahko je to seznam lastnih povezav, seznam povezav drugega uporabnika ali pa seznam najdenih povezav. Uporabnik klikne na ime ali pa na gumb "Details" ene izmed povezav.

Osnovni tok dogodkov

1. Sistem najde povezavo za šifro povezave za ogled.
2. Odpre se stran s podrobnostmi povezave.
3. Na vrhu strani je ime povezave.
4. Na levi strani so lastnosti povezave (lastnik povezave, datum vnosa, naslov povezave, opis, seznam oznak in hierarhija strukturnih elementov).
5. Na desni strani je predstavitvena slika povezave.
6. Tok dogodkov se uspešno zaključi.

Alternativni tokovi dogodkov

Ni najdenih rezultatov:

3.a1 Sistem za šifro ne najde povezave.

3.a2 Na vrhu strani se izpiše obvestilo, da povezava ni bila najdena.

Stanje po zaključku

Uporabnik se lahko pomakne na prejšnjo stran ter izbere katero od drugih povezav in odpre njene podrobnosti.

3.1.3 Urejanje povezave

Akterji

- GUI Uporabnik

Zahteve

Uporabnik želi spremeniti določene lastnosti povezave, ki jo je predhodno vnesel.

Sprožilec

Prijavljen uporabnik odpre stran s svojimi povezavami s klikom na "Links" ter:

- odpre stran za urejanje povezave s klikom na srednji gumb "Edit", ki se nahaja na desni strani vsake navedene povezave ali
- odpre stran za pregled povezave s klikom na zgornji gumb "Details", ki se nahaja na desni strani vsake navedene povezave in nato odpre stran za urejanje s klikom na gumb "Edit", ki je na dnu strani.

Osnovni tok dogodkov

1. Odpre se stran za urejanje povezave z vnosnimi polji za naslov, ime, opis, sliko, oznakami in strukturnimi elementi.
2. Uporabnik spremeni podatke povezave in potrdi s tipko "Enter" ali s klikom na gumb "Save" pod vnosnimi polji.
3. Sistem preveri in shrani podatke povezave.
4. Sistem prikaže na novo urejene podatke obstoječe povezave.

Alternativni tokovi dogodkov

Uporabnik ne vnese nujno zahtevanih podatkov za povezavo:

3.a1 Sistem v koraku 3 osnovnega toka ugotovi, kateri podatki so manjkajoči.

3.a2 Sistem na vnosni maski prikaže sporočila o napakah. Vsako sporočilo je izpisano ob vnosnem polju, ki ima manjkajoč zahtevan podatek.

3.a3 Osnovni tok dogodkov se nadaljuje s točko 2.

Uporabnik vnese neveljavne podatke za povezavo:

3.b1 Sistem v koraku 3 osnovnega toka ugotovi, kateri podatki so neveljavni.

3.b2 Sistem na vnosni maski prikaže sporočila o napakah. Vsako sporočilo je izpisano ob vnosnem polju, ki je neustrezno izpolnjeno.

3.b3 Osnovni tok dogodkov se nadaljuje s točko 2.

3.1.4 Dodajanje povezave

Akterji

- GUI Uporabnik

Zahteve

GUI uporabnik želi shraniti novo povezavo.

Sprožilec

Prijavljen uporabnik odpre stran za dodajanje povezave s klikom na "Add Link" v meniju.

Osnovni tok dogodkov

1. Odpre se stran za dodajanje povezave z vnosnimi polji za naslov, ime, opis, sliko, oznakami in strukturnimi elementi.
2. Uporabnik vnese podatke za novo povezavo in potrdi s tipko "Enter" ali s klikom na gumb "Save" pod vnosnimi polji.
3. Sistem preveri in shrani podatke povezave.
4. Sistem prikaže podatke nove povezave.

Alternativni tokovi dogodkov

Uporabnik ne vnese nujno zahtevanih podatkov za povezavo:

3.a1 Sistem v koraku 3 osnovnega toka ugotovi, kateri podatki so manjkajoči.

3.a2 Sistem na vnosni maski prikaže sporočila o napakah. Vsako sporočilo je izpisano ob vnosnem polju, ki ima manjkajoč zahtevan podatek.

3.a3 Osnovni tok dogodkov se nadaljuje s točko 2.

Uporabnik vnese neveljavne podatke za povezavo:

3.b1 Sistem v koraku 3 osnovnega toka ugotovi, kateri podatki so neveljavni.

3.b2 Sistem na vnosni maski prikaže sporočila o napakah. Vsako sporočilo je izpisano ob vnosnem polju, ki je neustrezno izpolnjeno.

3.b3 Osnovni tok dogodkov se nadaljuje s točko 2.

3.1.5 Brisanje povezave

Akterji

- GUI Uporabnik

Zahteve

Uporabnik želi izbrisati povezavo, ki jo je predhodno vnesel.

Sprožilec

Prijavljen uporabnik odpre stran s svojimi povezavami s klikom na "Links" ter klikne na spodnji gumb "Delete", ki se nahaja na desni strani vsake navedene povezave.

Osnovni tok dogodkov

1. Odpre se dialog za potrditev brisanja povezave z gumboma "OK" in "Cancel".
2. Uporabnik potrdi z gumbom "OK".
3. Okno za potrditev se zapre.
4. Sistem preveri ukaz za brisanje in izbriše povezavo iz podatkovne baze.
5. Sistem prikaže sporočilo o uspešnem brisanju ter seznam uporabnikovih povezav brez tiste, ki je bila izbrisana.

Alternativni tokovi dogodkov

Uporabnik prekliče brisanje povezave:

2.a1 Uporabnik v koraku 2 osnovnega toka prekliče brisanje z gumbom "Cancel".

2.a2 Okno za potrditev se zapre.

2.a3 Tok dogodkov se zaključi.

3.1.6 Urejanje profila uporabnika

Akterji

- GUI Uporabnik

Zahteve

Prijavljen uporabnik želi urediti svoj profil.

Sprožilec

Uporabnik je prijavljen v aplikacijo ter klikne na meni "Profile".

Osnovni tok dogodkov

1. Sistem prebere podatke o prijavljenem uporabniku.
2. Odpre se stran s podrobnostmi o profilu ter polji za urejanje.
3. Urediti je mogoče ime ter naslov elektronske pošte uporabnika.
4. Uporabnik spremeni ime in/ali naslov elektronske pošte ter klikne na gumb "Save".
5. Sistem v podatkovni bazi spremeni podatke.
6. Prikaže se stran s profilom, ki vsebuje popravljene podatke.
7. Na vrhu strani je sporočilo o uspešnem shranjevanju podatkov.

Alternativni tokovi dogodkov

Ni prijavljenega uporabnika:

1.a1 Sistem ugotovi, da uporabnik ni prijavljen v sistem.

1.a2 Prikaže se stran za prijavo.

Neuspešno shranjevanje podatkov uporabnika:

4.b1 Sistem ugotovi, da ime ali naslov elektronske pošte nista pravilno vnesena.

4.b2 Prikaže se stran s profilom uporabnika.

4.b3 Prikaže se sporočilo o napaki.

4.b4 Pri polju za ime in/ali naslov elektronske pošte je sporočilo o napaki.

Stanje po zaključku

Uporabnik lahko nadaljuje z uporabo aplikacije.

3.1.7 Ogled profila uporabnika

Akterji

- GUI Uporabnik

Zahteve

Prijavljen uporabnik želi urediti svoj profil.

Sprožilec

Uporabnik je prijavljen v aplikacijo ter klikne na meni "Profile".

Osnovni tok dogodkov

1. Sistem prebere podatke o izbranem uporabniku.
2. Odpre se stran s podrobnostmi o uporabniku (na vrhu je ime uporabnika, pod njim število uporabnikovih povezav ter datum vnosa uporabnikovega računa).
3. Pod naslovnimi podatki so največkrat odprte povezave uporabnika ter največkrat odprte oznake uporabnikovih povezav.
4. Spodaj je prikazan seznam uporabnikovih povezav z osnovnimi podatki ter gumbi za ogled podrobnosti.

Alternativni tokovi dogodkov

Uporabnik za šifro ni bil najden:

1.a1 Sistem prikaže sporočilo o neznanem uporabniku.

Stanje po zaključku

Uporabnik lahko nadaljuje z uporabo aplikacije.

3.2 Primeri uporabe API uporabnika

API uporabnik ali uporabnik programskega vmesnika uporablja osnovne funkcije, ki jih uporablja tudi uporabnik spletnega vmesnika. Vse operacije so sestavljene iz JSON zahtevka (ki deluje kot sprožilec primera uporabe), obdelave na strežniku in JSON odgovora.

3.2.1 Branje uporabnikovih povezav

Akterji

- API Uporabnik

Zahteve

API uporabnik želi prebrati celoten seznam svojih povezav.

Primeri

Primera JSON zahtevka in odgovora sta na voljo med prilogami.

- Primer zahtevka: A.5.1
- Primer odgovora: A.5.2

Sprožilec

API uporabnik strežniku pošlje JSON zahtevek z API ključem.

Osnovni tok dogodkov

1. Strežnik prebere zahtevek.
2. Sistem preveri veljavnost API ključa.
3. Sistem iz podatkovne baze prebere seznam povezav uporabnika.
4. Sistem vrne uspešen odgovor s seznamom povezav.

Alternativni tokovi dogodkov

Strežnik ne zna razbrati zahtevka:

1.a1 Sistem vrne odgovor o neznanem zahtevku.

Sistem ugotovi, da ključ ni veljaven:

2.b1 Sistem vrne odgovor z napako o neveljavnem ključu.

Zgodi se sistemska napaka (na strežniku, podatkovni bazi ...):

1.c1 Sistem vrne odgovor z opisom napake.

3.2.2 Branje povezave

Akterji

- API Uporabnik

Zahteve

API uporabnik želi prebrati podatke o svoji povezavi.

Primeri

Primera JSON zahtevka in odgovora sta na voljo med prilogami.

- Primer zahtevka: A.4.1
- Primer odgovora: A.4.2

Sprožilec

API uporabnik strežniku pošlje JSON zahtevek z API ključem in šifro povezave.

Osnovni tok dogodkov

1. Strežnik prebere zahtevek.
2. Sistem preveri veljavnost API ključa.
3. Sistem preveri, ali povezava s šifro obstaja.
4. Sistem preveri, ali povezava s šifro pripada uporabniku.
5. Sistem pošlje odgovor o uspešnosti s podatki povezave.

Alternativni tokovi dogodkov

Strežnik ne zna razbrati zahtevka:

1.a1 Sistem vrne odgovor o neznanem zahtevku.

Sistem ugotovi, da ključ ni veljaven:

2.b1 Sistem vrne odgovor z napako o neveljavnem ključu.

Sistem ugotovi, da šifra povezave v zahtevku ni prisotna ali je neveljavna:

3.c1 Sistem vrne odgovor z napako o manjkajoči/neveljavni šifri.

Sistem ugotovi, da povezava s šifro ne obstaja:

3.d1 Sistem vrne odgovor z napako o neobstoječi povezavi.

Sistem ugotovi, da povezava s šifro ne pripada uporabniku:

4.e1 Sistem vrne odgovor z napako.

Zgodi se sistemska napaka (na strežniku, podatkovni bazi ...):

1.f1 Sistem vrne odgovor z opisom napake.

3.2.3 Dodajanje povezave

Akterji

- API Uporabnik

Zahteve

API uporabnik želi dodati novo povezavo.

Primeri

Primera JSON zahtevka in odgovora sta na voljo med prilogami.

- Primer zahtevka: A.3.1
- Primer odgovora: A.3.2

Sprožilec

API uporabnik strežniku pošlje JSON zahtevek z API ključem in podatki za novo povezavo.

Osnovni tok dogodkov

1. Strežnik prebere zahtevek.
2. Sistem preveri veljavnost API ključa.
3. Sistem preveri, ali so podatki s povezavo ustrezni.
4. Sistem shrani povezavo.
5. Sistem pošlje odgovor o uspešnosti dodajanja povezave.

Alternativni tokovi dogodkov

Strežnik ne zna razbrati zahtevka:

1.a1 Sistem vrne odgovor o neznanem zahtevku.

Sistem ugotovi, da ključ ni veljaven:

2.b1 Sistem vrne odgovor z napako o neveljavnem ključu.

Zgodi se napaka pri validaciji podatkov za novo povezavo:

3.c1 Sistem vrne odgovor z napakami v podatkih za novo povezavo.

Zgodi se sistemska napaka (na strežniku, podatkovni bazi ...):

1.d1 Sistem vrne odgovor z opisom napake.

3.2.4 Brisanje povezave

Akterji

- API Uporabnik

Zahteve

API uporabnik želi izbrisati povezavo.

Primeri

Primera JSON zahtevka in odgovora sta na voljo med prilogami.

- Primer zahtevka: A.6.1
- Primer odgovora: A.6.2

Sprožilec

API uporabnik strežniku pošlje JSON zahtevek z API ključem in šifro povezave.

Osnovni tok dogodkov

1. Strežnik prebere zahtevek.
2. Sistem preveri veljavnost API ključa.
3. Sistem preveri, ali povezava s šifro obstaja.
4. Sistem preveri, ali povezava s šifro pripada uporabniku.
5. Sistem izbriše povezave.
6. Sistem pošlje odgovor o uspešnosti brisanja povezave.

Alternativni tokovi dogodkov

Strežnik ne zna razbrati zahtevka:

1.a1 Sistem vrne odgovor o neznanem zahtevku.

Sistem ugotovi, da ključ ni veljaven:

2.b1 Sistem vrne odgovor z napako o neveljavnem ključu.

Sistem ugotovi, da šifra povezave v zahtevku ni prisotna ali je neveljavna:

3.c1 Sistem vrne odgovor z napako o manjkajoči/neveljavni šifri.

Sistem ugotovi, da povezava s šifro ne obstaja:

3.d1 Sistem vrne odgovor z napako o neobstoječi povezavi.

Sistem ugotovi, da povezava s šifro ne pripada uporabniku:

4.e1 Sistem vrne odgovor z napako.

Zgodi se sistemska napaka (na strežniku, podatkovni bazi ...):

1.f1 Sistem vrne odgovor z opisom napake.

3.2.5 Preverjanje API ključa

Akterji

- API Uporabnik

Zahteve

API uporabnik želi preveriti veljavnost API ključa ali dostopnost strežnika.

Primeri

Primera JSON zahtevka in odgovora sta na voljo med prilogami.

- Primer zahtevka: A.1.1
- Primer odgovora: A.1.2

Sprožilec

API uporabnik strežniku pošlje JSON zahtevek z API ključem.

Osnovni tok dogodkov

1. Strežnik prebere zahtevek.
2. Sistem preveri veljavnost API ključa.
3. Sistem pošlje pozitiven odgovor o veljavnosti ključa.

Alternativni tokovi dogodkov

Strežnik ne zna razbrati zahtevka:

1.a1 Sistem vrne odgovor o neznanem zahtevku.

Sistem ugotovi, da API ključ v zahtevku ni pristen:

2.b1 Sistem vrne odgovor z napako o manjkajočem ključu.

Sistem ugotovi, da ključ ni veljaven:

2.c1 Sistem vrne negativen odgovor o veljavnosti ključa.

Zgodi se sistemska napaka (na strežniku, podatkovni bazi ...):

1.d1 Sistem vrne odgovor z opisom napake.

4 Izbira tehnologije

Aplikacija bo implementirana v programskem jeziku Java na platformi Spring MVC z uporabo strežnika Apache Tomcat. Podatki bodo shranjeni v podatkovni bazi MySQL. Programski vmesnik bo realiziran s standardom JSON. Spletni del bo realiziran z uporabo HTML, CSS in knjižnic Bootstrap ter jQuery. JSP predloge, ki se bodo uporabljale, omogočajo eksternalizacijo besedila v jezikovne datoteke, ki jih je mogoče prevajati v druge jezike.

Uporabljali bomo odprtokodne rešitve za shranjevanje kode (Github) ter orodje za razvoj aplikacij, ki temelji na Eclipse (Spring Tool Suite – vsebuje ogrodje za implementacijo, testiranje ter poganjanje).

4.1 Odjemalni del

Uporabniku spletnega brskalnika je potrebno olajšati dostop in delo z aplikacijo. Kjer je mogoče, je potrebno podatke za vnos ali spremembo preveriti že znotraj brskalnika – jQuery. Kadar je mogoče, se morajo podatki nalagati prek skript Ajax.

Uporabnikom, ki uporabljajo programski vmesnik, je potrebno zagotoviti dostop do osrednjih funkcionalnosti aplikacije, ki jih ima tudi uporabnik spletnega brskalnika. Zahtevki in odgovori morajo biti v standardizirani JSON obliki.

4.2 Strežniški del

Strežnik mora omogočati dostop do spletne aplikacije velikemu številu uporabnikov naenkrat. Posamezen zahtevek na strežnik ne sme trajati več kot 3 s. Spletni strežnik mora omogočati delo preko spletnega ali programskega vmesnika.

4.3 Podatkovna baza

Podatkovna baza mora omogočati transakcije (lastnosti ACID) ter hitro branje in pisanje. Posamezna poizvedba na podatkovni bazi ne sme trajati več kot 0,5 s. Podpirati mora uporabo indeksov (za potrebe iskanja tudi “full-text” indeksov).

Za potrebe ad-hoc poizvedb na zahtevo mora biti podatkovna baza relacijska in podpirati povpraševalni jezik SQL. Ker je izbrana tehnologija Java, mora biti podatkovna baza podprta s strani JDBC knjižnice.

4.4 Shranjevanje datotek

Datoteke bodo shranjene v direktorijih na operacijskem sistemu Linux. Skladišče datotek mora omogočati shranjevanje večjega števila datotek, pri čimer morajo biti datoteke shranjene v različnih poddirektorijih glede na lastnika – uporabnika, ki jih je naložil.

5 Načrtovanje sistema

Načrtovanje sistema vključuje načrtovanje arhitekture in podrobno načrtovanje komponent.

5.1 Arhitektura sistema

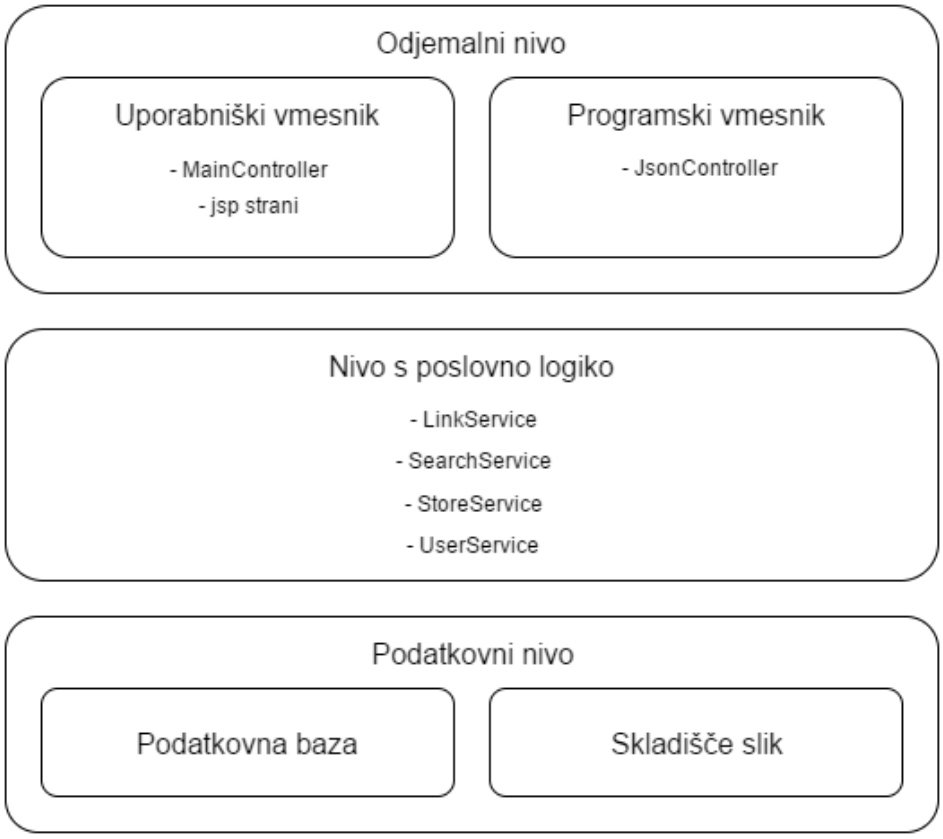
Osnovo za arhitekturne nivoje aplikacije predstavlja uporablja Spring MVC skupaj s podatkovno bazo MySQL. Podatkovna baza zagotavlja hrambo in integriteto podatkov ter predstavlja podatkovni nivo. Ogrodje Spring MVC predstavlja več nivojev– nivo za delo s podatki (JDBC template), nivo s poslovno logiko (services), odjemalni nivo (controllers) in znotraj tega uporabniški vmesnik (view). Slika 14 predstavlja hierarhijo nivojev aplikacije. Slika vsebuje tudi navedene osrednje Java razrede, ki so zastopani v posameznem nivoju in jih bomo podrobno opisali v nadaljevanju.

Pri delovanju aplikacije je vsak nivo vertikalno povezan s sosednjima nivojema. Odjemalni nivo (nivo uporabniškega/programskega vmesnika) je aplikacija kot jo vidi končni uporabnik. Vsak zahtevek (tudi JSON programski zahtevki) proti aplikaciji se sproži s tega nivoja in posreduje spletnemu nivoju. Tam se zahtevek procesira in oblikuje odgovor. Odjemalni nivo določene funkcionalnosti prepusti nivoju s poslovno logiko, ki vsebuje implementacijo poslovnih metod in ima dostop do podatkov – posredno prek dostopa do podatkovnega nivoja (in prek tega do skladišča slik ali podatkovne baze).

5.1.1 Odjemalni nivo

Ta nivo vsebuje uporabniški in programski vmesnik. Uporabniški vmesnik vsebuje "jsp" spletne strani, ki prikazujejo statične podatke skupaj z dinamično generiranimi v spletnem nivoju. Spletne strani lahko dostopajo do podatkov, ki jih je pripravil nivo s poslovno logiko – to so lahko podatki, namenjeni samo za določen zahtevek ali pa so dostopni tekom celotne seje.

Nivo predstavlja interakcijo uporabnikovega zahtevka s poslovno logiko. Nivo je realiziran s Spring MVC "controller" razredi. Metode v spletnem nivoju se sprožijo bodisi zaradi zahtevka v spletnem brskalniku ali JSON klica. Kontroler metode v spletnem



Slika 14: Nivoji aplikacije.

nivoju imajo dostop do trenutne HTTP seje, do uporabniške seje in trenutnega zahtevka. Tipičen primer uporabe je odgovor na spletni zahtevek, kjer spletni nivo pokliče poslovno logiko in dobi nazaj podatke, ki se nato uporabijo kot dinamična vsebina. V primeru programskega vmesnika nimamo "jsp" strani, saj se JSON odgovori zgradijo neposredno v "controller" razredih.

Razred MainController

Razred vsebuje implementacijo kontroler metod, ki so mapirane na URL naslove uporabniškega vmesnika. Vsaka izmed metod služi obravnavanju določenega HTTP zahtevka (GET ali POST). V GET primeru programer določi regularni izraz spletnih naslovov, ki jih bo metoda pokrivala, podatke, ki se bodo uporabili za izris spletne strani ter "jsp" stran, ki se bo na koncu izrisala. Pri POST primeru programer prav tako določi regularni izraz spletnih naslovov, akcije, ki se bodo zgodile na strežniku, ter naslov, na katerem naj se izvajanje nadaljuje (po koncu preusmerimo promet na GET naslov).

Razred JsonController

Razred vsebuje implementacijo JSON kontroler metod, ki so mapirane na URL naslove programskega vmesnika. Vsaka izmed metod služi obravnavanju določenega HTTP zahtevka (GET ali POST). V GET primeru programer določi regularni izraz spletnih naslovov, ki jih bo metoda pokrivala, podatke, ki se bodo uporabili za izpis JSON paketa ter oblikovanje JSON paketa. Pri POST primeru programer prav tako določi regularni izraz spletnih naslovov, akcije, ki se bodo zgodile na strežniku, ter podatke, ki se bodo uporabili za izpis JSON paketa ter oblikovanje JSON paketa.

5.1.2 Nivo s poslovno logiko

Nivo je realiziran s Spring MVC "service" razredi. Uporablja se kot vmesni nivo med nivojem za delo s podatki in spletnim nivojem. Tukaj se zgodi vsa poslovna logika, ki je vezana na določeno uporabniško akcijo. Akcija je lahko sprožena s strani uporabnika za spletnim vmesnikom ali pa s strani zunanjega uporabnika, ki uporablja programski JSON vmesnik – v obeh primerih se uporabijo iste metode poslovne logike.

Konsistenca podatkov

Ker uporabljamo dva različna podatkovna vira, je potrebno poskrbeti za integriteto podatkov med njima. To realiziramo s poslovno logiko v "service" razredih. Vsaka metoda v teh razredih mora delovati atomarno – če se določen del metode ne izvede uspešno, mora sprožiti izjemo, kar povrne stanje v obeh podatkovnih virih na stanje pred začetkom izvajanja metode. Najprej izvedemo ukaz na podatkovni bazi (brez ukaza "commit"), nato izvedemo ukaz na podatkovnem sistemu. Če je bil tudi drugi ukaz uspešen, sprožimo "commit", sicer "rollback".

Razred LinkService

Razred vsebuje metode za delo z povezavami:

- branje seznama tipov podatkov;
- branje podrobnosti o tipu podatkov;
- shranjevanje podatkov nove povezave;
- shranjevanje podatkov obstoječe povezave;
- shranjevanje oznak;
- shranjevanje hierarhične strukture;

- branje povezav za uporabnika;
- branje povezav za oznako,
- branje povezave po šifri;
- branje zadnjih vnesenih povezav;
- branje priljubljenih povezav;
- branje zadnjih vnesenih oznak;
- branje priljubljenih oznak;
- brisanje povezave.

Razred SearchService

Razred vsebuje metodo za iskanje povezav. Vhodni podatek je iskani znakovni niz, izhodni pa seznam najdenih povezav (ali prazen seznam, kadar ni bilo najdenih povezav).

Razred StoreService

Razred vsebuje metodi za shranjevanje in branje predstavnih slik povezav. Pri obeh operacijah metodi ustrezno obravnavata umeščanje slik v poddirektorije glede na uporabnika.

Razred UserService

Razred vsebuje metode za delo z uporabniki aplikacije:

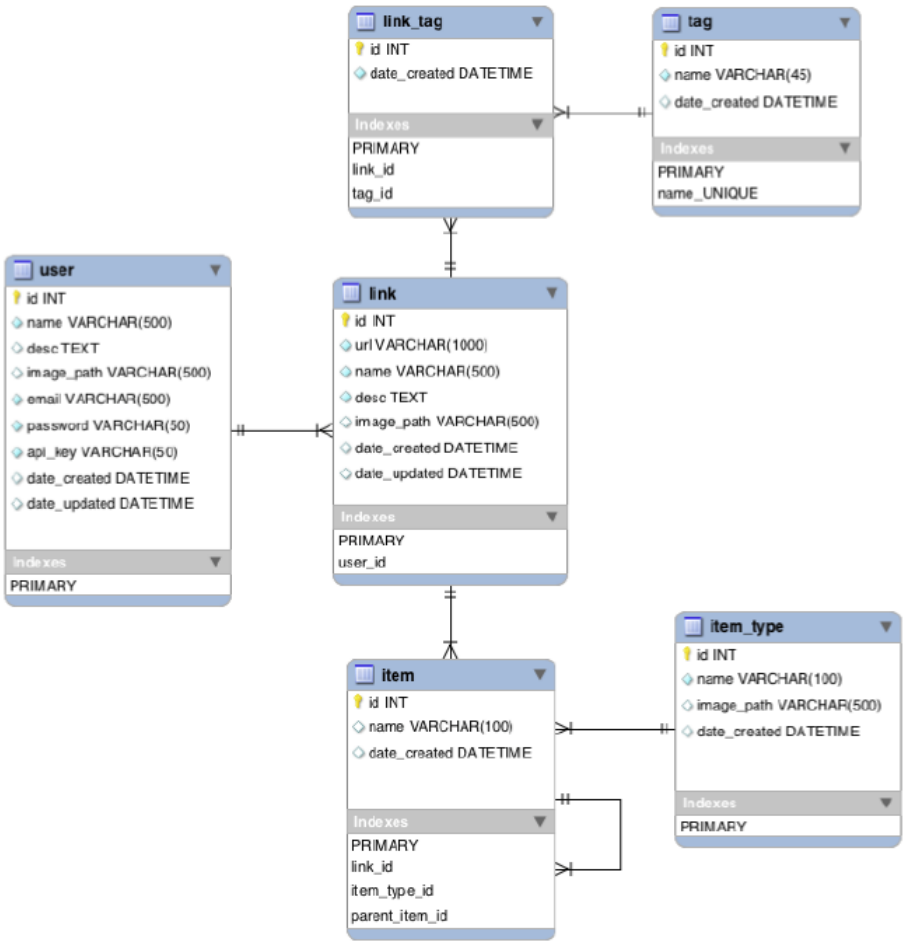
- branje podatkov uporabnika glede na šifro;
- branje podatkov uporabnika glede na API ključ;
- branje podatkov uporabnika glede na naslov elektronske pošte;
- shranjevanje podatkov novega uporabnika;
- shranjevanje podatkov obstoječega uporabnika;
- branje podatkov o uporabnikih z največ povezavami.

5.1.3 Podatkovni nivo

Podatkovna baza hrani vse podatke, razen slik, ki so shranjene na datotečnem sistemu. Osnovni direktorij za slike se v aplikaciji uporabi kot konstanta (nastavljivo), zato se poti do slik v podatkovno bazo shranjujejo relativno. Podrobnosti so opisane v naslednjih sekcijah.

Podatkovna baza

Slika 15 vsebuje grafični prikaz podatkovnega modela aplikacije.



Slika 15: Model podatkovne baze.

Tabela: user Tabela vsebuje podatke o uporabniku, ki vnaša in ureja povezave. Uporabnik ima naziv in opis (name, desc), prikazno sliko (image_path – pot do datoteke, relativno glede na direktorij s slikami). Stolpca z naslovom elektronske

pošte in geslom se uporabljata za prijavo. Dodan je stolpec z generiranim ključem (api_key), ki ga zunanje aplikacije uporabijo za vnos in urejanje podatkov prek programskega vmesnika.

Tabela: povezava Tabela vsebuje podatke o povezavah. Vsaka povezava ima referenco na uporabnika, ki ga je vnesel ter ga lahko ureja. Stolpec s spletnim naslovom (url) vsebuje spletno povezavo. Dodani so še stolpci z imenom, opisom in demonstracijsko sliko povezave (name, desc, image_path). Dodan je še stolpec s številom ogledov povezav.

Tabela: item Tabela vsebuje podatke za drevesno strukturo podatkov, ki jih vsebuje povezava. Vsi podatki, ki so vezani na povezavo, imajo referenco na povezavo (link_id), med seboj pa so hierarhično povezani (parent_item_id). Podatki, ki so na najvišjem nivoju, imajo null vrednost v stolpcu parent_item_id. Vsak podatek ima še referenco, ki definira tip podatka (item_type_id).

Tabela: item_type Šifrant različnih tipov hierarhičnih podatkov vsebuje ime (name) in pot do datoteke z ikono, ki predstavlja tip podatka (image_path).

Tabela: tag Vsebuje različne oznake, ki jih imajo lahko povezave. Oznaka je unikatna.

Tabela: link_tag Vmesna tabela, ki določa razmerja med povezavami in oznakami. Vsaka povezava ima lahko poljubno število oznak, lahko tudi nobene. Poljubne povezave različnih uporabnikov so lahko povezane z istimi oznakami, saj gre zgolj za opisni podatek.

Dostop do podatkov v podatkovni bazi je realiziran s pomočjo knjižnice JDBC template. Ta skrbi za delo neposredno s podatkovno bazo. Nivo podpira vnos, branje, spreminjanje in brisanje podatkov. Pri branju podatkov jih s pomočjo JDBC template pretvorimo v Java razrede, ki jih uporabljajo višji nivoji. Podatki se iz teh razredov berejo tudi pred vnosom, spreminjanjem in brisanjem.

Skladišče slik

Poleg podatkovne baze aplikacija shranjuje podatke (predstavne slike povezav) tudi neposredno na datotečnem sistemu. Uporabljamo enega od novejših datotečnih sistemov na sistemu Linux (ext4 ali novejši – neomejeno število poddirektorijev; ni težav, če uporabnik naloži veliko količino slik). Slike so shranjene v hierarhiji direktorijev pod osnovnim direktorijem. Zaradi organizacije ima vsak uporabnik svoj poddirektorij pod osnovnim direktorijem s slikami. Ime poddirektorija je "id" tega uporabnika

(tabela "user", stolpec "id"). Poimenovanje slik je sestavljeno iz "id"-ja povezave, ki jo slika predstavlja (tabela "link", polje "id") ter končnice ".jpg" (sistem pri nalaganju slike pretvori vsako sliko v format JPG). Slike, ki predstavljajo tipe prilog "item" so shranjene v skupnem poddirektorju "item_types", poimenovane pa so z "id"-jem vrste priloge (table "item_type", polje "id") ter končnico ".jpg".

5.2 Programski vmesnik

Programski vmesnik je namenjen samodejnemu vnašanju podatkov, ki ga sprožijo zunanje aplikacije. Za vsako možnost uporabe mora zunanja aplikacija na določen naslov poslati podatke v formatu JSON. Vse operacije prek API vmesnika so atomarne, vedno je potreben samo en zahtevek z veljavnim ključem. V vsakem primeru sistem vrne JSON z opisom odgovora, da uporabnik ve, ali je bila operacija uspešna. Sledi opis primerov uporabe prek programskega vmesnika. Zunanja aplikacija se programskemu vmesniku Linkpoint predstavi s ključem (polje api_key), ki mora biti prisoten v vsakemu zahtevku. Ključ je mogoče generirati s pomočjo grafičnega uporabniškega vmesnika po predhodni prijavi z uporabniškim imenom in geslom. V primeru napak pri izvajanju metode programski vmesnik vrne JSON s kodo in opisom napake. Če je prisoten element "error", lahko odjemalska aplikacija meni, da zahtevek ni bil uspešen in ustrezno obravnava napako. Primer 1 prikazuje JSON odgovor ob napaki.

```
1 {"error": {
2         "code": -1,
3         "desc": "Internal error"
4 }}
```

Primer 1: Primer JSON odgovora ob napaki

5.3 Strežnik

Aplikacija deluje na strežniku, ki ima nameščen spletni strežnik in podatkovno bazo. Strežnik ima nameščeno novejšo distribucijo operacijskega sistema Linux. Datotečni sistem je ext4.

5.3.1 Aplikacijski strežnik

Uporablja se aplikacijski strežnik Apache Tomcat verzije 7 – podpira JSP 2.2 in Servlet 3.0. V direktoriju "/opt" se nahaja bližnjica "apache-tomcat", ki kaže na inštalacijo strežnika Tomcat.

5.4 Shranjevanje podatkov

Aplikacija ima ločeno hrambo podatkov in slik. Podatki so shranjeni v podatkovni bazi, predstavne slike povezav pa neposredno na datotečnem sistemu.

5.4.1 Podatkovna baza

Na strežniku je nameščena podatkovna baza MySQL 5.6 s privzetimi nastavitvami. Na podatkovni bazi je ustvarjena shema "linkpoint" in uporabniško ime ter geslo za dostop, ki ju bo uporabljal aplikacijski strežnik za delo z bazo. Podatkovna baza se redno arhivira, varnostna kopija se izdela enkrat na dan. Uporabi se Unix "crontab", ki sproži MySQL ukaz "mysqldump", s čimer se podatki izvozijo. Izvozna datoteka se nato prekopira na varno lokacijo. V primeru potrebe po povrnitvi stanja iz arhiva se uporabi ukaz "mysql" na izvozni datoteki. V primeru, ko je potrebno podatkovno bazo povrniti v začetno stanje, se zažene skripta "init_linkpoint.sql", ki je priložena izvorni kodi aplikacije.

5.4.2 Skladišče slik

Na strežniku je direktorij "/data/images", kjer bodo shranjene naložene slike. Uporabnik "tomcat"mora imeti vse pravice na tem direktoriju (read, write, execute).

5.5 Podrobno načrtovanje

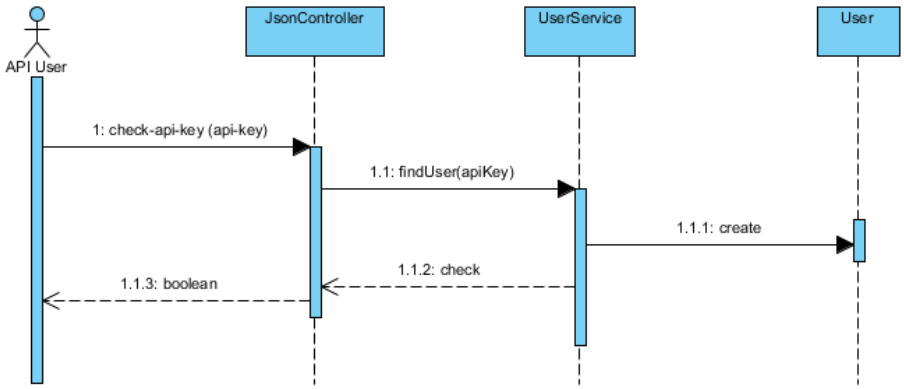
V tem razdelku bomo podrobneje opisali kompleksnejše postopke branja in shranjevanja podatkov. Sekvenčni diagrami opisujejo tok dogodkov pri izvajanju.

5.5.1 Preverjanje API ključa – JSON vmesnik

Slika 16 prikazuje diagram preverjanja API ključa s pomočjo programskega vmesnika. API uporabnik pošlje vhodni JSON, ki ga prestreže razred JsonController. Ta pokliče razred s poslovno logiko LinkService, ki iz podatkovne baze prebere informacije o uporabniku ter jih posreduje nazaj kontroler razredu. JsonController na koncu zgradi odhodni JSON in ga kot odgovor pošlje API uporabniku.

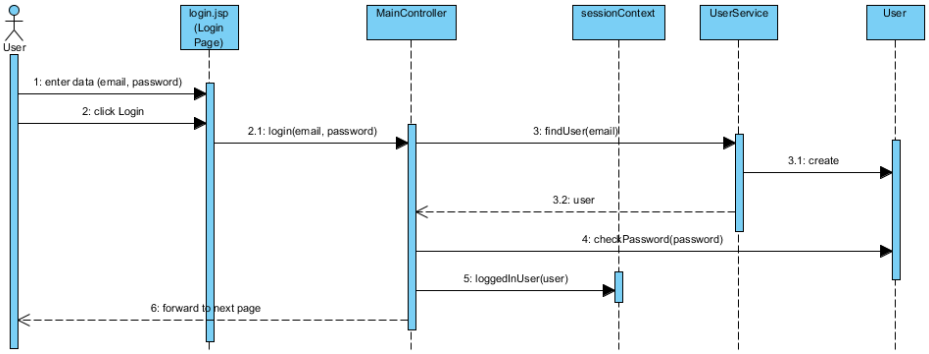
5.5.2 Prijava v aplikacijo – spletni vmesnik

Spletni brskalnik pošlje podatke za prijavo na strani "login.jsp" Controller razredu "MainController". Ta s pomočjo razreda "UserService" (razred najde uporabnika in



Slika 16: Diagram preverjanja API ključa.

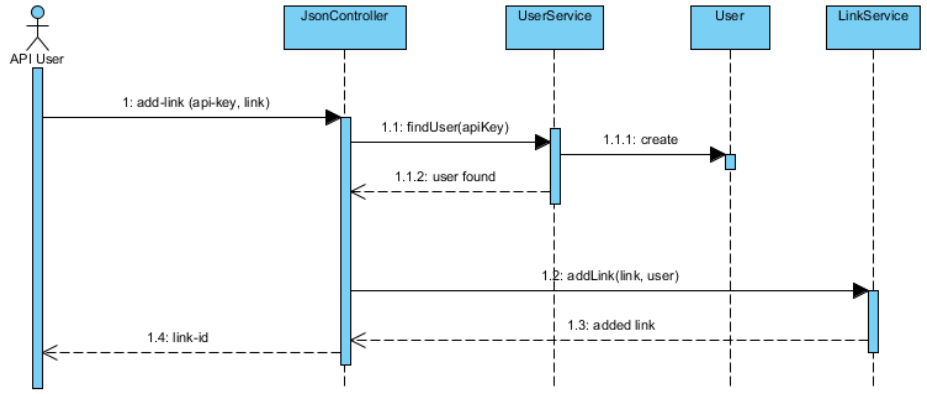
zanj ustvari novo instanco razreda "User") najde uporabnika ter mu preveri geslo. Po preverjanju Controller razred poskrbi za preusmeritev na naslednjo stran (v primeru uspešne prijave je to domača stran – "dashboard"). Pred preusmeritvijo Controller razred shrani podatke o uporabniku na sejo. Glej sliko 17.



Slika 17: Prijava v aplikacijo.

5.5.3 Dodajanje povezave – JSON vmesnik

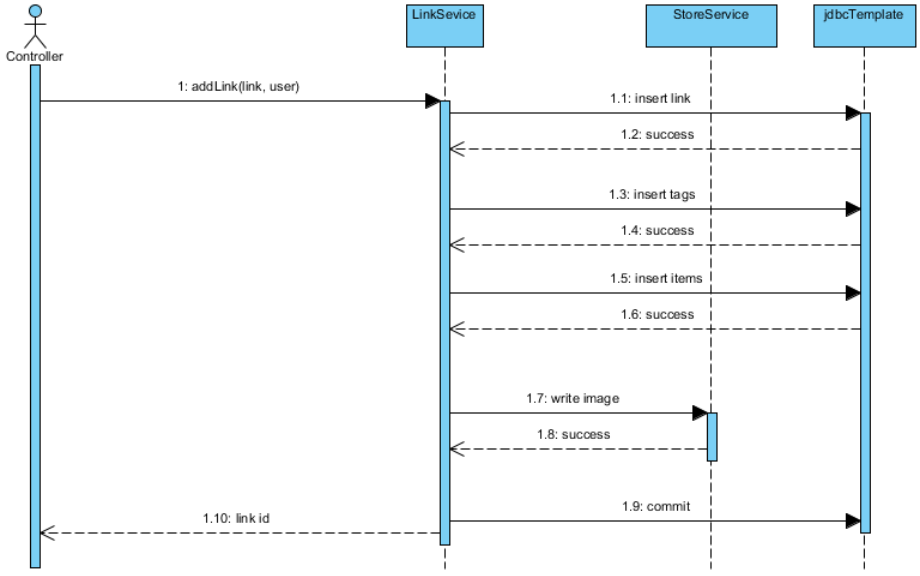
JsonController prevzame HTTP zahtevek in najprej pokliče UserService, da ta najde uporabnika. Nato pokliče LinkService, ki shrani povezavo in vrne ID nove povezave. Na koncu vrne ID povezave prek HTTP odgovora.



Slika 18: Dodajanje povezave s pomočjo programskega vmesnika.

5.5.4 Dodajanje povezave – poslovna logika

Service razred LinkService je vstopna točka, ki jo pokliče Controller razred – to je lahko JsonController v primeru JSON vmesnika ali MainController, kadar gre za uporabnika za spletnim brskalnikom. Primer prikazuje vrstni red operacij shranjevanja podatkov povezave, s katerim zagotovimo integriteto podatkov.



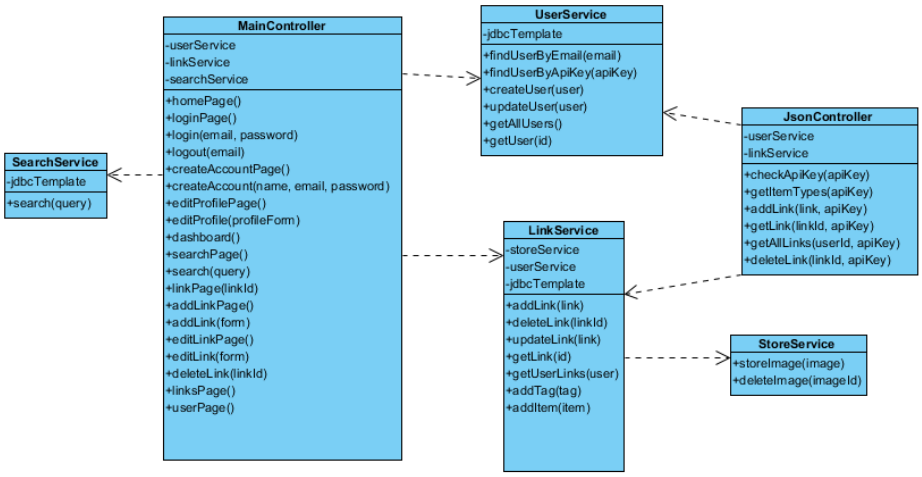
Slika 19: Poslovna logika dodajanja povezave.

6 Implementacija

V razdelku so opisani določeni postopki v aplikaciji z vidika implementacije. Prikazani so Spring MVC razredi z razrednim diagramom. Postopki so predstavljeni s sekvenčnimi diagrami.

6.1 Spletni razredi in razredi s poslovno logiko

Prikazani razredi vsebujejo vse osnovne metode za delovanje aplikacije. Kontroler razredi so vhodne točke za uporabnika, nato pa podrobne postopke prepustijo implementaciji Service razredov. Kontroler razredi včasih vsebujejo dve metodi, vezani na določeno funkcionalnost – npr. "login" in "loginPage". Metoda "loginPage" služi mapiranju GET HTTP zahtevka in se izvede pred prikazom strani za prijavo v aplikacijo. Metoda "login" je implementacija POST HTTP zahtevka in se izvede po kliku na gumb za prijavo.



Slika 20: Razredni diagram razredov s poslovno logiko.

6.2 Koncept MVC

Za osnovno arhitekturo razvoja aplikacije uporabljamo vzorec MVC. Na ta način so ločeni prezentacijski, kontrolni in servisni nivo. Ker uporabljamo ogrodje Spring MVC, nam ni potrebno razvijati svoje implementacije, ampak uporabimo Spring delitev – "pogled" predstavljajo spletne strani jsp, "kontroler" so Spring Controller razredi, "model" pa Spring Service razredi. Tako ogrodje nam omogoča razširljivost in uporabo skupnih komponent za implementacijo vmesnikov spletnega in API uporabnika.

6.2.1 Izrisovanje spletnih stran

Pri uporabniškem vmesniku, kjer uporabnik do strani dostopa prek brskalnika, mora aplikacija ustvariti HTML dokument po predlogi. Spletni nivo prejme HTTP zahtevek ter ga obdela v kontroler razredu, ki je za to zadolžen (MainController). Izvede se izpostavljena metoda, katere definicija se ujema s strukturo URI naslova. V metodi se za pripravo podatkov za izris strani po potrebi pokliče tudi nivo poslovne logike. Vsi podatki se nastavijo na model po principu ključ–vrednost. Ker sta spletni nivo in nivo poslovne logike povezana neposredno, je mogoče na model nastaviti tudi objekte domenskih razredov. Mehanizem za izris strani prebere predlogo v obliki "jsp" ter jo napolni s podatki iz modela glede na ključe. Na predlogah je mogoče uporabljati tudi programske konstrukte, kot so pogoji in zanke. Rezultat izrisa je spletna stran v obliki HTML, ki je na koncu pripravljena za transport do brskalnika.

6.2.2 Komunikacija med komponentami

Spletni nivo sprejme HTTP zahtevek (prek uporabniškega ali programskega vmesnika) ter ga obravnava. Komunikacija z nivojem poslovne logike se dogaja neposredno z uporabo klicev Java metod. Argumenti ali izhodne vrednosti so lahko šifre objektov ali pa celotni objekti domenskih razredov. Nivo poslovne logike za branje in shranjevanje podatkov uporablja enake objekte, s tem da jih je potrebno pred branjem ali pisanjem pretvoriti v podatkovne tipe, ki ustrezajo stolpcem tabel v podatkovni bazi. Kadar gre za predstavitvene slike povezav, jih je pred shranjevanjem treba serializirati oz. deserializirati pred branjem.

6.3 Uporaba zunanjih knjižnic

Za preverjanje veljavnosti seje prijavljenih uporabnikov in prijavo samo se uporablja knjižnica Spring Security. Za komunikacijo s podatkovno bazo se uporablja knjižnica

Spring JDBC ter MySQL Connector. Za pogosto uporabljane splošne operacije se uporablja knjižnica Apache Commons.

7 Testiranje sistema

V tem poglavju bomo navedli in podrobno opisali postopke za testiranje sistema. Postopki se ujemajo s primeri uporabe aplikacije.

7.1 Načrtovanje testiranja

Uporabniški vmesnik bomo testirali ročno, z uporabo testnih scenarijev. Z njimi želimo pokriti vso funkcionalnost, do katere dostopa uporabnik preko spletnega brskalnika. Testiranje programskega vmesnika smo se odločili izvesti s samodejnim funkcijskim testiranjem. Funkcijski testi morajo pokriti vse možne primere uporabe programskega vmesnika.

7.2 Testni scenariji

Vsak testni scenarij vsebuje pogoje za izvedbo, postopek in končno stanje. Kadar se postopki večkrat ponavljajo z različnimi vhodnimi podatki ter rezultati, jih bomo zaradi preglednosti organizirali v tabeli.

7.2.1 Registracija uporabnika

Pogoji

- Aplikacija LinkPoint se izvaja in je dostopna.
- GUI uporabnik ima odprto osnovno stran.
- V podatkovni bazi še ni evidentiranega uporabnika z naslovom elektronske pošte "up@linkpoint.com".

Postopek

1. Uporabnik odpre stran za registracijo (klikne na "Join").
2. Uporabnik v vnosna polja za ime, naslov elektronske pošte in geslo vnese vrednosti stolpcev "Ime", "Email naslov" in "Geslo" iz tabele ter potrdi registracijo s pritiskom na "Enter" ali klikom na gumb "Join", ki se nahaja pod vnosnimi polji.

Končno stanje

Končno stanje za določene vhodne parametre je navedeno v stolpcu "Končno stanje" v tabeli 1 na strani 49.

7.2.2 Prijava uporabnika

Pogoji

- Aplikacija LinkPoint se izvaja in je dostopna.
- Uporabnik ni prijavljen in ima odprto osnovno stran.
- V podatkovni bazi je evidentiran uporabnik z naslovom elektronske pošte "up@linkpoint.com" ter geslom "geslo9".

Postopek

1. Uporabnik odpre stran za prijavo (klikne na "Login").
2. Uporabnik v vnosni polji za naslov elektronske pošte in geslo vnese vrednosti stolpcev "Email naslov" in "Geslo" iz tabele ter potrdi prijavo s pritiskom na "Enter" ali klikom na gumb "Login", ki se nahaja pod vnosnimi polji.

Končno stanje

Končno stanje za določene vhodne parametre je navedeno v stolpcu "Končno stanje" v tabeli 2 na strani 50.

7.2.3 Iskanje povezav

Pogoji

- Aplikacija LinkPoint se izvaja in je dostopna.
- Uporabnik ni prijavljen in ima odprto osnovno stran.
- V podatkovni bazi so evidentirani uporabniki z vnesenimi povezavami. Uporabnik z imenom "Osirix 5" ima vnesene povezave z imeni "Osirix Sample - KNIX", "Osirix Sample - CENOVIX". Uporabnik z imenom "Osirix 2" ima vnešene povezave z imeni "Osirix Sample - WRIX" (opis vsebuje "Scaphoid fracture"), "Osirix Sample - VIX". Drugih uporabnikov in povezav v podatkovni bazi ni.

Tabela 1: Tabela parametrov in končnih stanj za registracijo uporabnika

Ime	**E-naslov**	**Geslo**	**Končno stanje**
			1. V podatkovni bazi ni sprememb. 2. Uporabniški vmesnik še vedno prikazuje stran za registracijo. 3. Pri vnosnem polju za ime je opozorilo, da polje ne sme biti prazno. Pri vnosnem polju za geslo je opozorilo, da polje ne sme biti prazno. Pri vnosnem polju za naslov elektronske pošte je opozorilo, da polje ne sme biti prazno.
l	lol	lol	1. V podatkovni bazi ni sprememb. 2. Uporabniški vmesnik še vedno prikazuje stran za registracijo. 3. Pri vnosnem polju za ime je opozorilo, da mora biti v polje vnesenih med 3 in 100 znakov. Pri vnosnem polju za geslo je opozorilo, da mora biti v polje vnesenih med 6 in 20 znakov. Pri vnosnem polju za naslov elektronske pošte je opozorilo, da vneseni naslov elektronske pošte ni v pravi obliki.
Uporabnik	up@linkpoint.com	geslo9	1. V podatkovni bazi je dodan nov uporabnik. 2. Uporabnik je samodejno prijavljen v aplikacijo. 3. Uporabniški vmesnik prikazuje osnovno stran prijavljenega uporabnika.

Tabela 2: Tabela parametrov in končnih stanj za prijavo uporabnika

Email naslov	**Geslo**	**Končno stanje**
		1. Uporabniški vmesnik prikazuje stran za prijavo. 2. Uporabniški vmesnik nad vnosnimi polji prikazuje opozorilo v rdeči barvi o napaki pri prijavi zaradi neveljavnega uporabniškega imena/gesla.
lol	lol	1. Uporabniški vmesnik prikazuje stran za prijavo. 2. Uporabniški vmesnik nad vnosnimi polji prikazuje opozorilo v rdeči barvi o napaki pri prijavi zaradi neveljavnega uporabniškega imena/gesla.
up@linkpoint.com	geslo8	1. Uporabniški vmesnik prikazuje stran za prijavo. 2. Uporabniški vmesnik nad vnosnimi polji prikazuje opozorilo v rdeči barvi o napaki pri prijavi zaradi neveljavnega uporabniškega imena/gesla.
up@linkpoint.com	geslo9	1. Uporabniški vmesnik prikazuje domačo stran uporabnika. 2. Prikazan je datum registracije in število povezav uporabnika ter obrazec za iskanje podatkov.

Postopek

1. Uporabnik odpre prvo stran, kjer je obrazec za iskanje. Enak obrazec za iskanje se nahaja tudi na strani za iskanje, dostopni prek menija "Search".

2. Uporabnik v vnosno polje za poizvedovanje "Query" vpiše vrednost "Poizvedba" iz tabele ter potrdi iskanje s pritiskom na "Enter" ali klikom na gumb "Search", ki se nahaja pod vnosnim poljem.

Končno stanje

Končno stanje za določene vhodne parametre je navedeno v stolpcu "Končno stanje" v tabeli 3 na strani 51.

Tabela 3: Tabela parametrov in končnih stanj za iskanje povezav

Poizvedba	Končno stanje
	1. Uporabniški vmesnik prikazuje stran za iskanje. 2. Uporabniški vmesnik nad gumbom prikazuje opozorilo v rdeči barvi o napaki pri iskanju, saj poizvedba ni bila vpisana.
lo	1. Uporabniški vmesnik prikazuje stran za iskanje. 2. Uporabniški vmesnik nad gumbom prikazuje opozorilo v rdeči barvi o napaki pri iskanju, saj je bila dolžina iskalnega niza prekratka (manj kot 3 znake).
lll	1. Uporabniški vmesnik prikazuje stran za iskanje. 2. Uporabniški vmesnik nad gumbom prikazuje obvestilo v črni barvi, ki uporabniku sporoča, da za iskani niz ni bilo najdenih povezav.

Nadaljevanje tabele 3	
Poizvedba	Končno stanje
osirix	1. Uporabniški vmesnik prikazuje stran za iskanje. 2. Pod obrazcem za iskanje je seznam rezultatov. 3. Oznaka, ki prikazuje število rezultatov, prikazuje število 4. 4. Seznam rezultatov vsebuje 4 rezultate, prikazane z imenom, povezavo, kratkim opisom, sliko, imenom uporabnika in gumbom za ogled podrobnosti povezave “Details”.
osirix 5	1. Uporabniški vmesnik prikazuje stran za iskanje. 2. Pod obrazcem za iskanje je seznam rezultatov. 3. Oznaka, ki prikazuje število rezultatov, prikazuje število 2. 4. Seznam rezultatov vsebuje 2 rezultata, prikazana z imenom, povezavo, kratkim opisom, sliko, imenom uporabnika in gumbom za ogled podrobnosti povezave “Details”.
osirix 55	1. Uporabniški vmesnik prikazuje stran za iskanje. 2. Uporabniški vmesnik nad gumbom prikazuje obvestilo v črni barvi, ki uporabniku sporoča, da za iskani niz ni bilo najdenih povezav.

Nadaljevanje tabele 3	
Poizvedba	Končno stanje
cenovix	1. Uporabniški vmesnik prikazuje stran za iskanje. 2. Pod obrazcem za iskanje je seznam rezultatov. 3. Oznaka, ki prikazuje število rezultatov, prikazuje število 1. 4. Seznam rezultatov vsebuje 1 rezultat, prikazan z imenom, povezavo, kratkim opisom, sliko, imenom uporabnika in gumbom za ogled podrobnosti povezave "Details". 5. Ime rezultata je "Osirix Sample - CENOVIX".

7.2.4 Vnos povezave

Pogoji

- Aplikacija LinkPoint se izvaja in je dostopna.
- V podatkovni bazi uporabnik nima povezav.
- Uporabnik je prijavljen in ima odprto domačo stran.

Postopek

1. Uporabnik odpre prvo stran, kjer je obrazec za dodajanje povezave dostopen prek menija "Add Link".
2. Uporabnik vnosna polja izpolni v skladu z vrednostmi "Vnesena polja" iz tabele ter potrdi vnos s pritiskom na "Enter" ali klikom na gumb "Save", ki se nahaja pod vnosnimi polji.

Končno stanje

Končno stanje za določene vhodne parametre je navedeno v stolpcu "Končno stanje" v tabeli 4 na strani 54.

Tabela 4: Tabela parametrov in končnih stanj za vnos povezave

Vnesena polja	Končno stanje
	1. Uporabniški vmesnik prikazuje stran za vnos povezave. 2. Uporabniški vmesnik pri vnosnih poljih "URL", "Name" in "Description" prikazuje opozorila v rdeči barvi. Opozorila pravijo, da polja ne smejo biti prazna.
URL: x, Name: x, Description: x	1. Uporabniški vmesnik prikazuje stran za vnos povezave. 2. Uporabniški vmesnik pri vnosnih poljih "URL", "Name" in "Description" prikazuje opozorila v rdeči barvi. Opozorila pravijo, da polje "URL" ne vsebuje veljavnega spletnega naslova, polji "Name" in "Description" pa vsebujeta prekratki vrednosti.

Nadaljevanje tabele 4

Vnesena polja	Končno stanje
• URL: http://www.famnit.si • Name: FAMNIT • Description: The Faculty of Mathematics, Natural Sciences and Information Technologies (UP FAMNIT) is a member of the University of Primorska, and was founded in 2006.	1. Povezava se uspešno shrani v podatkovno bazo. 2. Uporabniški vmesnik prikazuje pregled nove povezave. 3. Na vrhu strani je naslov, ki vsebuje ime nove povezave. 4. Med podatki je polje "Created at", ki vsebuje pravilen datum in čas vnosa. 5. Med podatki je polje "Created by", ki vsebuje ime uporabnika. 6. Polji "URL" in "Description" vsebujeta vneseni vrednosti. 7. Polji "Tags" in "Items" vsebujeta sporočili "No tags." in "No items.".

Nadaljevanje tabele 4	
Vnesena polja	Končno stanje
• URL: http://www.famnit.upr.si • Name: FAMNIT • Description: The Faculty of Mathematics, Natural Sciences and Information Technologies (UP FAMNIT) is a member of the University of Primorska, and was founded in 2006. • Tags: Test 1, Test 2 • Items: Case Name - Case 1, Image 1 Name: Image 1, Overlay 1 Name: Over 1	1. Povezava se uspešno shrani v podatkovno bazo. 2. Uporabniški vmesnik prikazuje pregled nove povezave. 3. Na vrhu strani je naslov, ki vsebuje ime nove povezave. 4. Med podatki, je polje “Created at”, ki vsebuje pravilen datum in čas vnosa. 5. Med podatki, je polje “Created by”, ki vsebuje ime uporabnika. 6. Polji “URL” in “Description” vsebujeta vneseni vrednosti. 7. Polje “Tags” vsebuje dve vneseni oznaki. 8. Polje “Items” prikazuje hierarhijo podatkov – na vrhu je “Case 1”, pod primerom “Image 1”, pod sliko “Over 1”.

7.2.5 Pregled in brisanje povezave

Pogoji

- Aplikacija LinkPoint se izvaja in je dostopna.
- Uporabnik je prijavljen in ima odprto osnovno stran.
- V podatkovni bazi ima uporabnik eno povezavo z imenom “Test link” z oznako "Test tag".

Postopek

1. Uporabnik odpre stran s povezavo “Test link”. Do povezave uporabnik pride prek menija “Links”, ki mu prikaže lastne povezave, in nato pri povezavi “Test link”

izbere možnost “Details”. Do povezave lahko pride tudi z iskanjem prek menija “Search”, v iskalno polje vnese “Test link”, potrdi iskanje ter na strani z rezultati pri ustrezni povezavi izbere “Details”.

Končno stanje

Končno stanje za določeno akcijo je navedeno v stolpcu “Končno stanje” v tabeli 5 na strani 57.

Tabela 5: Tabela akcij in končnih stanj za pregled povezave

Akcija	Končno stanje
	1. Uporabniški vmesnik prikazuje stran za pregled povezave. 2. Prikazane so vnesene lastnosti povezave. 3. Pod prikazanimi podatki je gumb za urejanje “Edit”. 4. Pod prikazanimi podatki je gumb za brisanje “Delete”.
Klik na gumb “Edit”	1. Uporabniški vmesnik prikazuje stran za urejanje povezave. 2. Uporabniški vmesnik vsebuje polja za urejanje ter gumb za shranjevanje “Save”.
Klik na gumb “Delete” in potrditev z gumbom “OK”	1. Povezava je izbrisana iz podatkovne baze. 2. Uporabniški vmesnik prikazuje stran s seznamom uporabnikovih povezav, ki ne vsebuje izbrisanih.
Klik na oznako “Test tag”	1. Uporabniški vmesnik prikazuje stran s seznamom povezav z oznako “Test tag”. 2. Seznam povezav mora vsebovati vsaj povezavo “Test link”.

Nadaljevanje tabele 5	
Akcija	Končno stanje
Klik na povezavo v polju "URL"	1. Uporabniški vmesnik prikazuje stran za pregled povezave. 2. Spletni brskalnik v novem zavihku ali oknu prikazuje stran, na katero kaže povezava.

7.2.6 Urejanje povezave

Pogoji

- Aplikacija LinkPoint se izvaja in je dostopna.
- V podatkovni bazi ima uporabnik povezavo z imenom "Test link".
- Uporabnik je prijavljen in ima odprto stran s prikazom podatkov povezave "Test link".

Postopek

1. Uporabnik izpolni vnosna polja spremeni v skladu z vrednostmi "Vnesena polja" iz tabele ter potrdi vnos s pritiskom na "Enter" ali klikom na gumb "Save", ki se nahaja pod vnosnimi polji.

Končno stanje

Končno stanje za določene vhodne parametre je navedeno v stolpcu "Končno stanje" v tabeli 6 na strani 58.

Tabela 6: Tabela parametrov in končnih stanj za urejanje povezave

Vnesena polja	Končno stanje
• URL: • Name: • Description: • Ostala polja nespremenjena.	1. Uporabniški vmesnik prikazuje stran za urejanje povezave. 2. Uporabniški vmesnik pri vnosnih poljih "URL", "Name" in "Description" prikazuje opozorila v rdeči barvi. Opozorila pravijo, da polja ne smejo biti prazna.

Nadaljevanje tabele 6	
Vnesena polja	Končno stanje
• URL: x • Name: x • Description: x • Ostala polja nespremenjena.	1. Uporabniški vmesnik prikazuje stran za urejanje povezave. 2. Uporabniški vmesnik pri vnosnih poljih "URL", "Name" in "Description" prikazuje opozorila v rdeči barvi. Opozorila pravijo, da polje "URL" ne vsebuje veljavnega spletnega naslov, polji "Name" in "Description" pa vsebujeta prekratki vrednosti.
• URL: http://www.famnit.upr.si • Name: FAMNIT • Description: The Faculty of Mathematics, Natural Sciences and Information Technologies (UP FAMNIT) is a member of the University of Primorska, and was founded in 2006.	1. Spremembe se uspešno shranijo v podatkovno bazo. 2. Uporabniški vmesnik prikazuje pregled shranjene povezave. 3. Na vrhu strani je naslov, ki vsebuje ime shranjene povezave. 4. Med podatki je polje "Changed at", ki vsebuje pravilen datum in čas spremembe. 5. Polji "URL" in "Description" vsebujeta spremenjeni vrednosti.

7.3 Funkcijsko testiranje

Funkcijsko testiranje izvedemo s pomočjo samodejnega zagona aplikacije s privzetimi testnimi podatki, izvajanjem zaporednih klicev programskega vmesnika ter preverjanja JSON vrednosti in HTTP stanja odgovorov. Funkcijski testi se nahajajo v razredu "com.linkpoint.app.JsonTests" v naslednjih metodah:

testCheckApiKey testiranje metode za preverjanje veljavnosti API ključa;

testGetItemTypes testiranje metode za branje tipov strukturnih elementov;

testAddLink testiranje metode za dodajanje povezave;

testGetLink testiranje metode za branje povezave;

testGetAllLinks testiranje metode za branje vseh povezav uporabnika;

testRemoveLink testiranje metode za brisanje povezave.

7.4 Testiranje obremenitve sistema

Testiranje obremenitve sistema (stress test) izvajamo z uporabo orodja JMeter. Simuliramo lahko vzporedno uporabo aplikacije (tako uporabniškega kot programskega vmesnika) s strani poljubnega števila uporabnikov.

7.5 Rezultati testiranja

Pri izvedbi testni scenarijev se aplikacija odzove v skladu s predpisanimi končnimi stanji. Pri izvajanju funkcijskega testiranja se aplikacija odzove v skladu s predefiniranimi odgovori. Testni scenariji in funkcijsko testiranje je pripomoglo k temu, da smo nekaj hroščev našli in rešili med razvojem.

Pri obremenitvenih testih se aplikacija odzove v skladu z zahtevami. Upočastnitve so opazne samo pri hkratnem shranjevanju večjega števila povezav, kjer imajo povezave priložene večje prikazne slike. Pri shranjevanju slik ugotavljamo, da bi bilo smiselno generirati poenostavljeno verzijo prikaznih slik (nižja ločljivost, dodatna kompresija), s čimer bi ohranili večino opazne kakovosti, obenem pa zmanjšali čas shranjevanja (ter kasneje branja) in prihranili na diskovnem prostoru.

8 Zaključek

Aplikacija je razvita, pripravljena za delovanje in objavljena v obliki odprte kode na portalu github.com. Deluje v skladu z zahtevami, navedenimi v okviru tega magistrskega dela.

Največji izziv pri razvoju je predstavljala dilema, ali bi slikovne datoteke hranili kot datoteke ali kot vnose v podatkovni bazi. V prvem primeru je slike mogoče lažje pregledovati ročno pri vzdrževanju, izognemo pa se tudi dostopu do podatkovne baze, kadar je potrebno sliko prikazati (spletni zahtevki). V drugem primeru pa imamo bolj enostavno arhitekturo (saj podatkovna baza v vsakem primeru obstaja), olajšamo migracijo med strežniki, imamo pa tudi manj dela pri implementaciji dodatnega pomnenja podatkov za pohitritev delovanja. Težav pri sami implementaciji aplikacije ni bilo, pripomogli so tudi predhodno napisani testni scenariji ter uporaba funkcijskih testov.

Trenutno je aplikacija prilagojena za hranjenje povezav za specifičen primer uporabe v ragiologiji (tipi podatkov v podatkovni bazi so primerni za specifično hierarhično strukturo). Če se pojavi zahteva, je to strukturo mogoče spremeniti – potrebne so spremembe v podatkovni bazi in uporabniškem vmesniku. Podatkovni model je fleksibilen in omogoča tudi spremembo programske logike, kjer bi bila hierarhično struktura nastavljiva – potrebno bi jo bilo zapisati v JSON ali YAML obliki, ki bi služila kot osnova za delovanje. Aplikacija ponuja poenoten zapis podatkov o določenem domenskem področju.

Tehnologija, izbrana za implementacijo, je preizkušena in primerna za tovrstno aplikacijo. Kljub temu bi bilo za odjemalni del bolje uporabiti eno izmed bolj aktualnih ogrodij za spletne aplikacije. Primer takega ogrodja je Angular 2. Implementacija t. i. koncepta aplikacije z eno spletno stranjo ("single page application") je primerna, saj je uporabniku izkušnja bolj prijazna, obenem pa bi se znebili nivoja "jsp" strani, saj bi taka stran uporabljala programski JSON vmesnik. Uporaba aplikacije z mobilnimi platformami je podprta že sedaj skozi programski vmesnik, potrebno pa bi bilo implementirati posamezne mobilne aplikacije (Android, iOS ...).

Za podporo zelo visokega števila uporabnikov priporočamo implementacijo vmesnega pomnilnika, ki zmanjša obremenitev baze podatkov. Z uporabo implementacije Memcached lahko branje podatkov pospešimo tudi za faktor 100. V primeru dodatnih

potreb po izboljšanju je mogoče aplikacijo namestiti na več aplikacijskih strežnikov, ki dostopajo do iste podatkovne baze, dostop do njih pa omogočiti prek sistemov za izravnavo obremenitve – "load balancer".

9 Literatura

[1] *NIH Data Sharing Repositories*,
`http://www.nlm.nih.gov/NIHbmic/nih_data_sharing_repositories.html`.
(Datum ogleda: 28. 5. 2015.) *(Citirano na strani 1.)*

[2] *The Scout Report*,
`http://www.mail-archive.com/scout-report@hypatia.cs.wisc.edu/msg00038.html`. (Datum ogleda: 10. 1. 2017.) *(Citirano na strani 9.)*

[3] *Delicious*,
`https://del.icio.us`. (Datum ogleda: 10. 1. 2017.) *(Citirano na strani 9.)*

[4] *Social bookmarking*,
`https://en.wikipedia.org/wiki/Social_bookmarking`. (Datum ogleda: 10. 1. 2017.) *(Citirano na strani 9.)*

[5] *Enterprise bookmarking*,
`https://en.wikipedia.org/wiki/Enterprise_bookmarking`. (Datum ogleda: 10. 1. 2017.) *(Citirano na strani 9.)*

[6] *ApexKB*,
`https://en.wikipedia.org/wiki/Social_bookmarking`. (Datum ogleda: 10. 1. 2017.) *(Citirano na strani 9.)*

[7] *Knowledge Plaza*,
`https://www.knowledgeplaza.net`. (Datum ogleda: 10. 1. 2017.) *(Citirano na strani 9.)*

[8] P. Rogelj,P. Petrič, *Towards Open Medical Image Based Collaboration*,
`http://library.ijs.si/Stacks/Proceedings/InformationSociety/IS2013_Volume_A.pdf`. (Datum ogleda: 4. 7. 2014.) *(Citirano na straneh IX in 5.)*

[9] *eContour*,
`http://econtour.org`. (Datum ogleda: 13. 1. 2017.) *(Citirano na straneh IX, 7 in 8.)*

[10] *IMDB*,
`http://www.imdb.com`. (Datum ogleda: 13. 1. 2017.) *(Citirano na strani 8.)*

Priloge

A Primeri JSON zahtevkov in odgovorov

V tem poglavju bomo navedli primere veljavnih vhodnih in izhodnih JSON objektov pri uporabi programskega vmesnika.

A.1 Preverjanje API ključa

A.1.1 Primer zahtevka

```
{
        "api_key": "hd82jfs74cvnd72ag95"
}
```

A.1.2 Primer odgovora

```
{
        "valid": true
}
```

A.2 Branje seznama tipov podatkov

A.2.1 Primer zahtevka

```
{
        "api_key": "hd82jfs74cvnd72ag95"
}
```

A.2.2 Primer odgovora

```
{"item_types": [
  {
    "id": 1,
    "name": "Case"
```

```
    },
    {
      "id": 2,
      "name": "Image"
    },
    {
      "id": 3,
      "name": "Structure folder"
    },
    {
      "id": 4,
      "name": "Delineation"
    },
    {
      "id": 10,
      "name": "Overlay"
    },
    {
      "id": 100,
      "name": "Attachment"
    }
]}
```

A.3 Dodajanje povezave

A.3.1 Primer zahtevka

```
{
  "api_key": "hd82jfs74cvnd72ag95",
  "link": {
    "description": "Test-description",
    "image": "iVBORw0KGgoAAAANSUhEUgAAASwAAAEsCAYAAAB5fY5",
    "items": [{
      "items": [{
        "name": "Image1",
        "type": "2"
      }],
      "name": "Case1",
      "type": "1"
```

```
    }],
    "name": "Test-link",
    "tags": [
      "tag1",
      "tag2"
    ],
    "url": "http://www.test.com"
  }
}
```

A.3.2 Primer odgovora

```
{
  "link_id": 83
}
```

A.4 Branje podatkov povezave

A.4.1 Primer zahtevka

```
{
        "api_key": "hd82jfs74cvnd72ag95",
        "link_id": "83"
}
```

A.4.2 Primer odgovora

```
{"link": {
  "description": "Test-description",
  "id": 83,
  "image": "iVBORw0KGgoAAAANSUhEUgAAASwAAAEsCAYAAAB5fY4=",
  "items": [],
  "name": "Test-link",
  "tags": [
    "tag1",
    "tag2"
  ],
  "url": "http://www.test.com"
}}
```

A.5 Branje podatkov o vseh povezavah uporabnika

A.5.1 Primer zahtevka

```
{
        "api_key": "hd82jfs74cvnd72ag95"
}
```

A.5.2 Primer odgovora


```
{"links": [
  {
    "description": "Test-description",
    "id": 60,
    "items": [],
    "name": "Test-link",
    "tags": [],
    "url": "http://www.test.com"
  },
  {
    "description": "Test-description",
    "id": 61,
    "image": "iVBORw0KGgoAAAANSUhEUgAAASwAAAEsCAY",
    "items": [],
    "name": "Test-link",
    "tags": [],
    "url": "http://www.test.com"
  }
]}
```


A.6 Brisanje povezave

A.6.1 Primer zahtevka

```
{
        "api_key": "hd82jfs74cvnd72ag95",
        "link_id": "83"
}
```

A.6.2 Primer odgovora

```
{
        "success": true
}
```

B Navodila za namestitev in zagon aplikacije

V tem poglavju bomo navedli postopek priprave aplikacije in namestitve na strežnik.

1. Namestitev poljubne distribucije operacijskega sistema Linux (datotečni sistem ext4).
2. Namestitev Java 1.8 (JDK). Konfiguracija okoljske spremenljivke "JAVA_HOME", ki mora kazati na lokacijo nameščene Jave.
3. Namestitev Maven 3. Konfiguracija okoljske spremenljivke "M2_HOME", ki mora kazati na lokacijo nameščenega programa Maven.
4. Namestitev aplikacijskega strežnika Tomcat 7. V direktorij "/opt" dodamo bližnjico "apache-tomcat", ki kaže na inštalacijo strežnika Tomcat.
5. Namestitev podatkovne baze MySQL 5.6 s shemo "linkpoint" ter uporabnikom s privzetim uporabniškim imenom in geslom linkpoint/linkpoint. Omogočimo samo lokalni dostop do podatkovne baze.
6. Ustvarjen direktorij "/data/images". Uporabnik "tomcat" mora imeti vse pravice na tem direktoriju (read, write, execute).
7. Kopiranje izvorne kode na strežnik z ukazom "git clone" na Git repozitoriju projekta (https://github.com/link-point/linkpoint.git).
8. Prevajanje in priprava izvorne kode z ukazom "mvn package" v direktoriju z izvorno kodo.
9. Kopiranje "war" datoteke z aplikacijo na aplikacijski strežnik v direktorij "/opt/apache-tomcat/webapps/ROOT/".
10. Inicializacija podatkovne baze s skripto "init_linkpoint.sql", ki je priložena izvorni kodi.
11. Zagon aplikacijskega strežnika Tomcat z ukazom "/opt/apache-tomcat/bin/startup.sh.

12. Če je bil postopek uspešen, je aplikacija prek spletnega brskalnika dostopna na naslovu “http://localhost:8080”.

13. Za zagon v produkcijske namene uporabimo strežnika Apache httpd in preusmerimo promet iz vrat 80 na 8080.