Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Polona Štefanič

Interaktivna aplikacija za večdotično površino

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentorica: viš. pred. dr. Alenka Kavčič

Ljubljana, 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Interaktivna aplikacija za večdotično površino.

V okviru diplomske naloge izdelajte interaktivno spletno aplikacijo za večdotično površino, ki deluje kot informacijska točka na fakulteti. Glavni namen informacijske točke je prikaz seznamov zaposlenih in prostorov ter virtualnega kažipota do iskanega prostora. Poleg tega naj aplikacija omogoča tudi prikaz novic in splošnih fakultetnih informacij. Pri razvoju aplikacije upoštevajte omejitve in zahteve večdotične površine, a naj bo razvita aplikacija dovolj prilagodljiva, da bo uporabna tudi na večjih mobilnih napravah (tablice) in bo nemoteno delovala tudi brez povezave v splet. Podatki naj bodo shranjeni v primerni nerelacijski podatkovni bazi NoSQL, za katero poiščite ustrezno odprtokodno implementacijo za delovanje z večdotično površino.

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Polona Štefanič sem avtorica diplomskega dela z naslovom:

Interaktivna aplikacija za večdotično površino.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom viš. pred. dr. Alenke Kavčič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu prek univerzitetnega spletnega arhiva.

V Ljubljani, dne 10. septembra 2015 Podpis avtorice:

Zahvaljujem se svoji mentorici, viš. pred. dr. Alenki Kavčič, za pomoč, razumevanje, spodbudo, usmerjanje in koristne nasvete pri izdelavi diplomskega dela.

Zahvala gre tudi moji družini za vso podporo med študijskim časom in nastajanjem diplomskega dela ter vsem prijateljem in kolegom za pomoč ter vzpodbudne besede.

Diplomsko delo posvečam svoji družini.

Kazalo

Povzetek

Cilj diplomskega dela je bil izdelati interaktivno spletno informacijsko točko FRI za delovanje na večdotični površini, razviti na Fakulteti za računalništvo in informatiko (FRI). Razvita interaktivna spletna aplikacija *InfoFRI točka* predstavlja virtualni kažipot po fakulteti do vseh prostorov, in sicer z izrisovanjem interaktivnih poti na tlorisih zgradbe FRI in objekta X od začetne točke v avli fakultete vse do iskanega prostora. Za lažje in enostavnejše urejanje podatkovne baze *InfoFRI točke* je bila razvita še zaledna spletna aplikacija *InfoFRI admin*. Obe aplikaciji sta narejeni v spletnih tehnologijah HTML5, CSS3 in JavaScript (jQuery). Z namenom predstaviti novejši tip nerelacijskih podatkovnih baz in upoštevajoč zahteve večdotične površine za shranjevanje podatkov v dokumentu JSON, smo implementirali podatkovno bazo CouchDB, v diplomskem delu pa poleg CouchDB predstavili še nerelacijske podatkovne baze NoSQL.

Ključne besede: senzor Kinect, HTML5, CSS3, JavaScript, jQuery, NoSQL, CouchDB, nerelacijska podatkovna baza.

Abstract

The goal of this thesis is the implementation of an interactive web application, an info point, which could run on a multitouch surface created at the ViCoS lab at Faculty of Computer and Information Science. The developed interactive web application *InfoFRI point* represents a virtual signpost for the faculty's rooms by drawing an interactive paths on the floor plan of the faculty building and the building of object X, starting at the faculty's lobby in the 1st floor and continuing up to the specific room. Another web application was created in the scope of the thesis in order to ease the manipulation and editing of the *InfoFRI point* database. Both web applications were implemented in web technologies such as HTML5, CSS3 and JavaScript (jQuery). In order to comply with the requirements of the multitouch surface for storing data in JSON and to introduce lightweight databases, a document-oriented database CouchDB was implemented.

Keywords: Sensor Kinect, HTML5, CSS3, JavaScript, jQuery, NoSQL, CouchDB, lightweight database.

Poglavje 1

Uvod

Živimo v informacijski dobi, ko nas digitalna tehnologija spremlja na vsakem koraku in se je njeni uporabi pravzaprav nemogoče izogniti. Skoraj vsak posameznik ima že vsaj eno pametno mobilno napravo, ki postaja nepogrešljiv del njegovega vsakdanjika. Večnamenske pametne mobilne naprave omogočajo lažji dostop do informacij in interakcijo z njimi. Zaradi tega nastaja čedalje več elektronskih storitev ali pa se te selijo na svetovni splet. Različne institucije in podjetja torej ponujajo svoje vsebine uporabnikom prek svetovnega spleta, pa naj gre za predstavitvene, storitvene ali druge vsebine, kamor lahko štejemo tudi virtualne interaktivne zemljevide ali načrte stavb in vodnike po zgradbah.

Fakulteta za računalništvo in informatiko (FRI) se je julija 2014 preselila v novo zgradbo na Večni poti. Tedaj se je utrnila ideja o postavitvi informacijske točke oz. virtualnega spletnega kažipota za večdotično površino, razvito v Laboratoriju za umetne vizualne in spoznavne sisteme (ViCoS) na FRI, ki bi, postavljena v avli fakultete, nudila študentom, gostujočim učiteljem in vsem obiskovalcem lažjo orientacijo po fakulteti, pregled in iskanje po zaposlenih in prostorih ter izris interaktivnih poti do prostorov na fakulteti.

Cilj diplomskega dela je izdelati spletno aplikacijo *InfoFRI točka* za omenjeno večdotično površino, ki omogoča iskanje po zaposlenih in prostorih

na FRI ter izris interaktivnih poti na tlorisih zgradbe FRI in objekta X. Kot dodatek omogoča aplikacija še prikazovanje novic s fakultetnimi informacijami javnega značaja. Za urejanje razvite spletne aplikacije je bila v okviru diplomskega dela razvita še druga spletna aplikacija *InfoFRI admin*, ki omogoča urejanje *InfoFRI točke* prek spletnega uporabniškega vmesnika.

Vsebina diplomskega dela je razdeljena na osem poglavij, začenši s podrobnejšim opisom uporabniških zahtev in obeh razvitih aplikacij v poglavju 2 ter z opisom delovanja večdotične površine v poglavju 3. Ker smo pri razvoju obeh aplikacij posegli predvsem na področje spletnih tehnologij, zajema prvi del diplomskega dela tudi predstavitev spletnih tehnologij HTML, CSS, JavaScript (jQuery) in PHP v poglavju 4, splošen opis nerelacijskih podatkovnih baz NoSQL v poglavju 5 in obširnejšo predstavitev podatkovne baze CouchDB v poglavju 6. Zadnji dve poglavji sta empirične narave in sta namenjeni izključno opisu in prikazu implementacije obeh spletnih aplikacij, predstavljenih po komponentah ter podprtih z odseki programske kode in zaslonskimi posnetki.

Poglavje 2

Informacijska točka FRI

2.1 Uporabniške zahteve

Glavne smernice načrtovanja implementacije so izhajale iz spodaj naštetih funkcionalnih uporabniških zahtev, izgled aplikacije pa smo prilagodili predvsem (1) optimalni implementaciji uporabniških zahtev in upoštevajoč (2) sodobne smernice razvoja odzivnih (responsive) spletnih aplikacij ter (3) interese uporabnikov, ki jim je virtualni spletni kažipot namenjen. Pri razvoju smo zaradi lažjega prilagajanja uporabniškim zahtevam uporabili agilne metode.

Funkcionalne uporabniške zahteve so:

- razviti virtualni kažipot kot spletno aplikacijo za večdotično površino,[1]
- uporabiti spletne tehnologije HTML5, CSS3 in JavaScript,
- omogočiti iskanje po seznamu zaposlenih in seznamu prostorov,
- izrisati interaktivne poti od določene točke do ustreznega prostora na tlorisih zgradbe FRI in objekta X,
- shraniti podatke v nerelacijski podatkovni bazi – omejitev večdotične površine,
- implementirati še aplikacijo za urejanje spletnega virtualnega kažipota.

[1] Več o omenjeni večdotični površini je zapisano v poglavju 3.

2.2 Predstavitev razvite aplikacije

V okviru diplomskega dela smo izdelali spletno aplikacijo *InfoFRI točka*, ki predstavlja interaktivni virtualni kažipot do vseh prostorov na fakulteti in hkrati omogoča tudi prikaz informacij javnega značaja, kot so fakultetne splošne informacije (vpis na fakulteto, študijski programi, opisi laboratorijev ...) ali napovedi dogodkov, vabljenih predavanj ipd., v obliki krajših novic, pri katerih je mogoče poleg naslova objaviti še krajši povzetek in daljše besedilo ter dodati manjšo in večjo sliko. Poleg novic so na prvi strani še premikajoče prosojnice z osnovnimi informacijami o fakulteti.

Aplikacija je namenjena delovanju na večdotični površini in omogoča sprotno iskanje brez povezave do podatkovne baze (live search), ločen in skupen izpis po seznamih vseh zaposlenih ter prostorov in filtracijo s spustnimi meniji po glavnih razvrstitvenih kategorijah in pri iskanju v tabeli s sprotnimi vnosi. Pri zaposlenih so te kategorije profesorji, asistenti, raziskovalci in ostali (strokovni delavci), pri prostorih pa predavalnice, računalniške učilnice, laboratoriji, pisarne ter ostali prostori.

Pri izboru posameznega vnosa v tabeli se začne na tlorisu ustreznega nadstropja izrisovati interaktivna pot od začetne točke v avli fakultete do iskanega prostora. Hkrati so levo zgoraj prikazane še informacije o iskanem nizu, desno pa mini 3D-model zgradbe FRI in objekta X z obarvanim nadstropjem, v katerem se trenutno na tlorisu zgradb izrisuje pot. Podrobnejše delovanje in opis implementacije *InfoFRI točke* sta opisana v poglavju 7.

Med razvijanjem spletne aplikacije *InfoFRI točka* se je pokazala potreba po implementaciji še dodatne spletne aplikacije *InfoFRI admin*, ki bi omogočala urejanje podatkovne baze aplikacije *InfoFRI točka* prek prijaznega grafičnega uporabniškega vmesnika. V *InfoFRI adminu* lahko dodajamo, brišemo ali spreminjamo vnose pri novicah, osebah in prostorih ter prosojnice z informacijami na prvi strani. Kot vsako podobno administratorsko orodje zahteva tudi *InfoFRI admin* prijavo v sistem za urejanje aplikacije *InfoFRI točka*.

Poleg tega lahko podatkovno bazo izvozimo kot dokument JSON in jo prenesemo v *InfoFRI točko*. Več o implementaciji zaledne spletne aplikacije *InfoFRI admin* in njeni povezavi z nerelacijsko podatkovno bazo CouchDB je mogoče prebrati v poglavju 8.

Implementacija virtualne spletne informacijske točke *InfoFRI točka* je bila prilagojena trenutnemu stanju večdotične površine, razvite na Fakulteti za računalništvo in informatiko. Ker večdotična površina podpira shranjevanje podatkov v tekstovnih datotekah, kot je JSON, smo, upoštevajoč njene zmogljivosti, podatke shranili v nerelacijsko podatkovno bazo CouchDB. Več o večdotični površini je zapisano v poglavju 3. Vsa nadaljnja poglavja vsebujejo še opise uporabljenih spletnih tehnologij, vključno s podatkovnimi bazami NoSQL in CouchDB, ter opise implementacij obeh aplikacij.

Poglavje 3

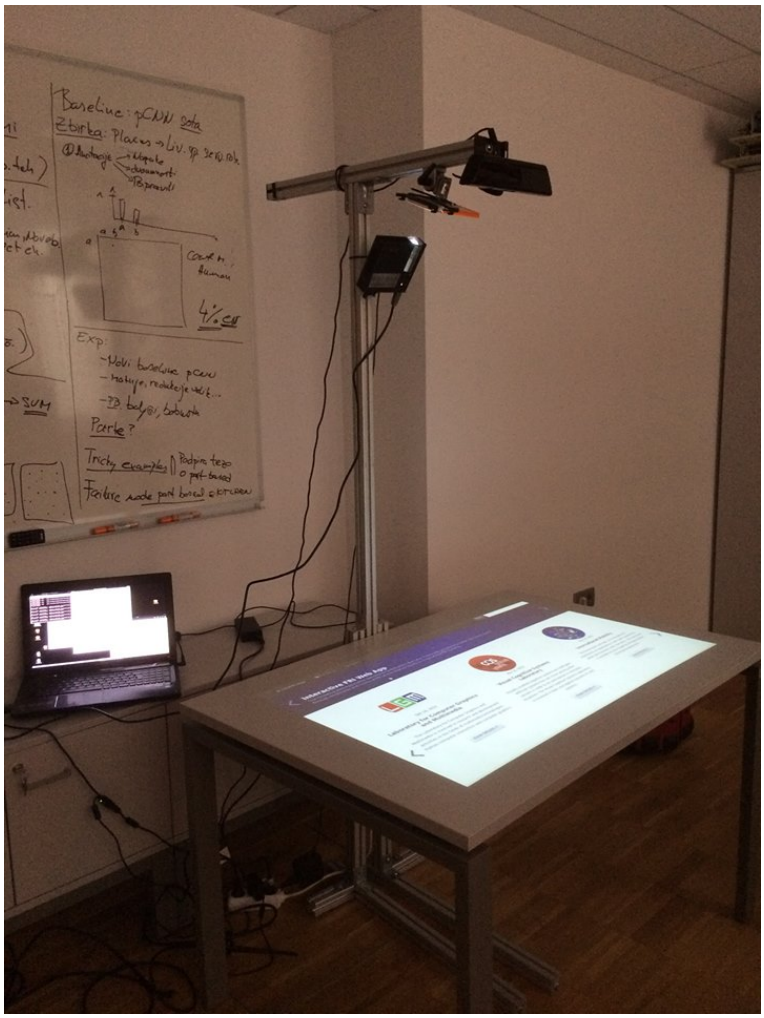
Večdotična površina

V tem poglavju je opisana večdotična površina, razvita v Laboratoriju za umetne vizualne spoznavne sisteme (ViCoS) na Fakulteti za računalništvo in informatiko. Nastajati je začela kot del raziskovalnega dela v omenjenem laboratoriju, kasneje pa jo je do trenutnega delovanja nadgradil Klemen Istenič v okviru svojega diplomskega dela [2]. Predstavitev in delovanje večdotične površine so Klemen Istenič, Luka Čehovin in Danijel Skočaj objavili tudi v znanstvenem članku [3].

3.1 Opis večdotične površine

Funkcionalnost večdotične površine omogoča več skupaj povezanih komponent. Na dolgem navpičnem stojalu so pritrjeni Microsoftov senzor Kinect, projektor in ogledalo. Ob stojalu stoji miza, ob njej ali pod njo računalnik, ki je povezan s senzorjem Kinect. Projektor je nameščen pod ostrim kotom in obrnjen navzgor, tako da je snop žarkov usmerjen neposredno v ogledalo, od katerega se svetloba iz projektorja odbije in tako je slika posredno projicirana na mizo, ki je spodaj (slika 3.1). Uporabnik lahko s premikanjem prstov in dotiki na površini mize upravlja z aplikacijo. Miza dobi tako lastnosti večdotične površine. Zaradi namestitve projektorja pod ostrim kotom in odboja svetlobe od ogledala – prav tako pod nepravim kotom – je lahko slika popačena (trapezne oblike), zato jo je treba pred uporabo aplikacije na

večdotični površini s premikom leče pri projektorju še ustrezno popraviti.



Slika 3.1: Na dolgem stojalu pritrjeni projektor, Kinect, ogledalo in spodaj stoječa miza ter računalnik ob strani skupaj delujejo kot večdotična površina, zgrajena in implementirana v Laboratoriju za umetne vizualne in spoznavne sisteme (ViCoS) na FRI.

Vsako površino, na katero želimo projicirati aplikacijo, lahko uporabimo za večdotično, najsi bo vodoravna ali navpična (npr. miza ali stena) in načeloma vseh vrst materialov, z izjemo preveč odbojnih ali prosojnih. Odbojni oz. prosojni materiali prejeto (infrardečo) svetlobo prekomerno oddajajo oz. absorbirajo in tako niso primerni, saj globinska kamera pri napravi Kinect ne

more pravilno določiti oddaljenosti od površine. Prav tako površina ne sme biti obsijana s sončno svetlobo, saj lahko slednja vsebuje tudi infrardeče valovanje in prav tako lahko popači zajete podatke.

Senzor Kinect je povezan z računalnikom, na katerem sta nameščena operacijski sistem Linux Ubuntu 12.04 in aplikacija za zaznavanje prstov ter obdelavo podatkov, pridobljenih s senzorjem Kinect. Pri testiranju aplikacije na večdotični površini je bil uporabljen ultralahek prenosni projektor *Asus P2E* svetilnosti 350 lumnov (LED diode), ločljivosti HD 1280 x 800 slikovnih elementov in možnosti trapeznega popravka slike ter premika leče.

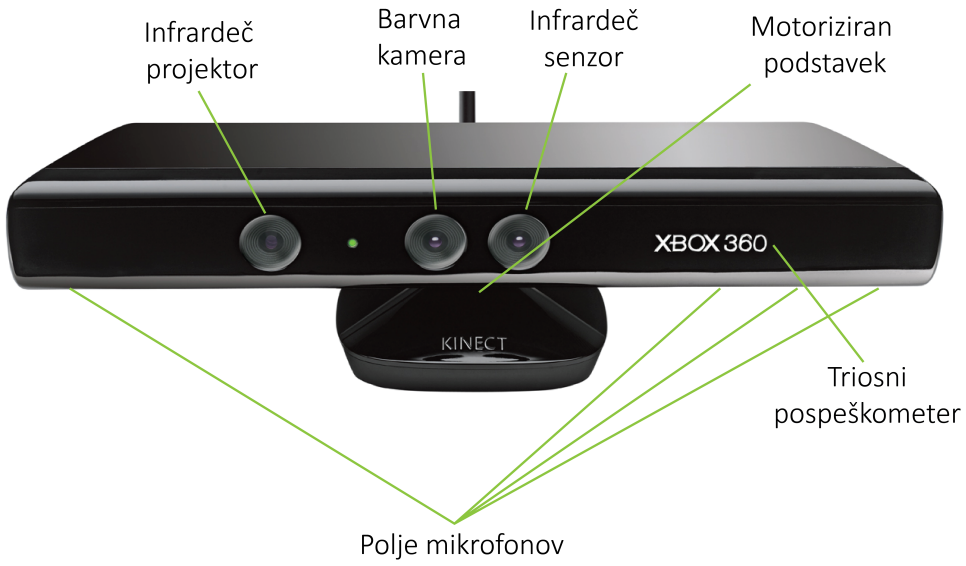
Naprava Kinect mora biti za optimalno delovanje nameščena približno 95 cm od površine, pri čemer je velikost projicirane slike na površino pri tej razdalji 80 cm x 69 cm. Povprečni odzivni čas sistema je od 100 ms do 120 ms. V času so zajeti uporabnikova akcija na večdotični površini, zajem in obdelava globinske slike, pošiljanje podatkov prek protokola TUIO in prikaz uporabnikovega dejanja na večdotični površini, pri čemer največ časa vzamejo zajem in obdelava globinske slike ter posredovanje podatkov [2, str. 49, 50]. Območje zaznavanja naprave Kinect je pri večdotični površini med 0,4 m in 6 m, pri uporabi igralne konzole Xbox 360 pa je domet med 1,2 m in 3,5 m [2, str. 14].

Slika 3.2: Zaznavanje prstov ob zagonu večdotične površine.

3.2 Kinect

Kinect (slika 3.3) je naprava z integriranim optičnim in zvočnim sistemom za zaznavanje gibanja, v osnovi namenjena upravljanju igralne konzole Xbox 360 podjetja Microsoft. Za igranje iger ga je potrebno povezati s konzolo Xbox in ga namestiti po dolžini nad ali pod video zaslon (računalniški monitor ali televizor). Pri igranju iger uporabnikom omogoča interakcijo s konzolo in računalnikom s kretnjami, gibi telesa in glasovnimi ukazi [17].

V približno polmetrskem ozkem ohišju ima Kinect[1] vgrajene različne senzorje, in sicer infrardeč (IR) projektor, barvno kamero, infrardeč senzor, motoriziran podstavek, triosni pospeškometer in polje mikrofonov. Ti senzorji omogočajo zajem zvoka, barvne in globinske slike, prepoznavanje delov telesa ter glasovno prepoznavo uporabnika [12].



Slika 3.3: Microsoftov senzor Kinect.

[1]Kinect je znan tudi pod imenom *Project Natal*.

3.2.1 Barvna kamera

Barvna kamera omogoča prepoznavanje obrazov, zajete podatke pa shrani na podlagi treh barvnih kanalov (rdečega, zelenega in modrega) v 32-bitne slike ločljivosti 640 x 480 slikovnih elementov (pikslov) pri hitrosti 30 slik na sekundo ali slike ločljivosti 1280 x 1024 slikovnih elementov pri hitrosti 15 slik na sekundo [12].

3.2.2 Polje mikrofonov in pospeškometer

Vzdolž spodnjega dela naprave Kinect so vgrajeni štirje mikrofoni – vsak kanal obdeluje podatke s 16 biti in frekvenco vzorčenja 16 kHz, vsebujejo pa tudi analogno-digitalni pretvornik za obdelavo signalov. Mikrofoni omogočajo snemanje zvoka, glasovno prepoznavanje uporabnika, prostorsko določitev zvočnega vira ter odstranjevanje odmeva in šuma. Triosni pospeškometer se uporablja za določanje trenutne orientacije senzorja Kinect [12].

3.2.3 Globinska kamera (infrardeč projektor in senzor)

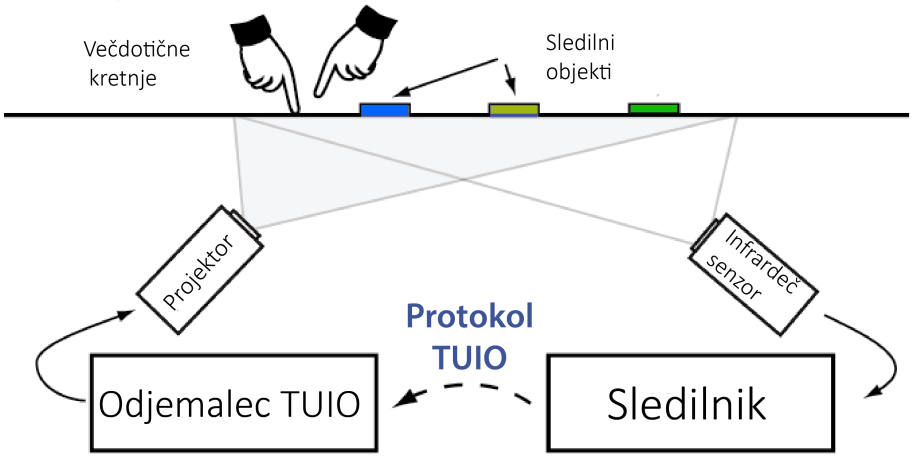
Infrardeč projektor razprši snop infrardečih žarkov v prostor, enobarvni globinski senzor pa zazna in interpretira infrardeče žarke, odbite od objektov. Na podlagi odbitih žarkov je mogoče izračunati razdaljo od objektov do senzorja, kar omogoča zajetje globinske slike (3D) pri različnih svetlobnih pogojih. Prav tako Kinect omogoča tudi samodejno kalibracijo senzorja glede na uporabnike in objekte v dometu senzorjev. Infrardeč projektor in senzor sta v napravi Kinect oddaljena drug od drugega približno 8 cm, med njima pa je barvna kamera [12].

3.2.4 Motoriziran podstavek

Motoriziran podstavek na dnu naprave Kinect omogoča samodejno sledenje kretnjam in gibom uporabnikov (slika 3.3). Senzor se lahko premika gor in dol za največ 57 stopinj ter levo in desno za največ 43 stopinj [12].

3.3 Komunikacijski vmesnik – protokol TUIO

Protokol TUIO je odprtokodno orodje, ki definira komunikacijski vmesnik za večdotične površine. Omogoča prenos podatkov med sledilnikom (ang. TUIO tracker application) in dejanskimi uporabniškimi vmesniki na površini (ang. client application). Razvit je bil za uporabo na področju večdotičnih površin in omogoča prenos abstraktnega opisa interaktivnih površin, skupaj z lastnostmi zaznanih uporabnikovih gest, dotikov in stanj objektov na površini. Protokol zakodira tok podatkov iz sledilnika, pridobljenih s pomočjo algoritmov računalniškega vida, in jih pošlje aplikaciji (odjemalcu), ki dešifrira protokol[2] [18].



Slika 3.4: Protokol TUIO [2, str. 39].

Protokol TUIO je implementiran na podlagi standarda za interaktivna okolja Open Sound Control (OSC), zaradi česar je kompatibilen z vsako platformo, ki podpira OSC. Privzet način prenosa podatkov je enkapsulacija podatkov OSC znotraj paketa UDP. Za manjšo zakasnitev paketov in lažjo implementacijo se uporablja večinoma protokol TUIO/UDP, novejše implementacije pa podpirajo tudi uporabo transportnega protokola TCP. Trenutna verzija protokola TUIO je 1.1 [18].

[2]Obstaja veliko aplikacij za zaznavanje in sledenje ter knjižnic za odjemalce za različna programska orodja in okolja.

Protokol TUIO uporablja dva tipa sporočil – SET in ALIVE. Sporočila SET se uporabljajo za prenos podatkov o stanju objektov, kot so pozicija, orientacija in vsakršni podatki za razpoznavanje stanj in sprememb lastnosti zaznanih objektov. Sporočila ALIVE prikazujejo aktivni nabor zaznanih objektov, ki so prisotni na večdotični površini in imajo enolične sejne identifikacijske številke [18].

Delovanje protokola TUIO poteka po naslednjih korakih:

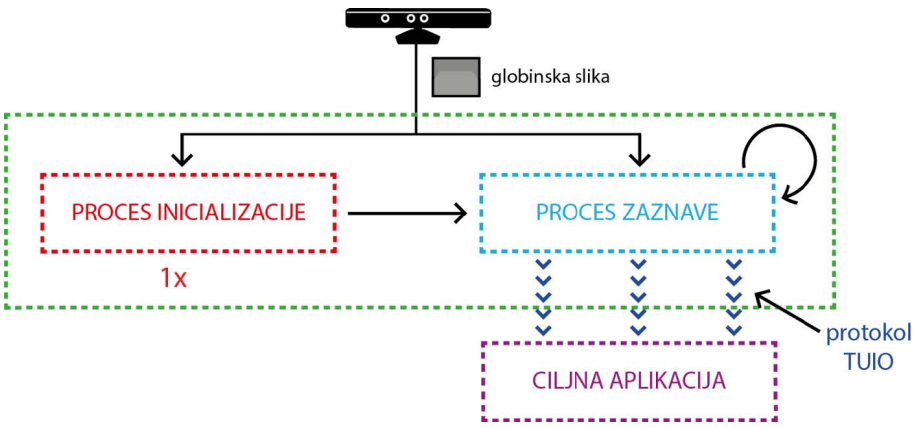
- Ob vsaki spremembi stanja (dotiki, premiki) so poslane lastnosti zaznanih objektov (sporočilo SET).
- Pri odstranitvi objekta pošlje strežnik posodobljeno sporočilo ALIVE z novim naborom trenutno aktivnih (zaznanih) objektov.
- Paketi, zajeti ob istem času oz. koraku posodobitve stanja, imajo enako identifikacijsko številko.
- Na podlagi sporočil SET in ALIVE ter identifikacijskih številk odjemalec razume, ali gre za dodajanje oz. odstranitev objekta s površine [18].

3.4 Inicializacija, kalibracija in zaznavanje

Preden začnemo uporabljati aplikacije na večdotični površini, je treba sistem najprej **inicializirati**. Pri procesu inicializacije se najprej zgradi model površine in kalibracija naprave Kinect s projektorjem. Model površine je zgrajen na podlagi globinske slike, matematične enačbe ravnine in mej površine. Na podlagi teh podatkov sistem izračuna oddaljenost in lokacijo zaznanih uporabnikovih prstov. Med zaznavanjem se model ozadja osvežuje [2, str. 21–37].

Pri **kalibraciji** se uporabniku na večdotični površini prikazujejo točke, ki jih sistem po uporabnikovi izbiri shrani in preslika iz koordinatnega sistema globinske kamere v koordinatni sistem projektorja.

Medtem ko se proces inicializacije izvrši le ob zagonu aplikacije za zaznavanje, **proces zaznave** poteka neprestano. Po zajemu globinskih slik jih sistem najprej procesira: odstrani ozadje in šume ter zaznane dotike prejšnjih globinskih slik povezuje z dotiki novejših. Podatke o lokaciji uporabnikovih dotikov preslika v koordinatni sistem aplikacije (odjemalca), ki teče na večdotični površini, in jih pošlje prek protokola TUIO [2, str. 21–37].



Slika 3.5: Prikaz delovanja protokola TUIO [2, str. 21].

Poglavje 4

Spletne tehnologije pri razvoju

V tem poglavju so predstavljene in opisane spletne tehnologije, uporabljene pri implementaciji *InfoFRI točke* in *InfoFRI admina*, začenši s HTML. Opisu HTML sledi predstavitev enega izmed ključnih elementov v HTML5 – *canvas*, v nadaljevanju pa so predstavljene še stilske predloge (CSS) skupaj z medijskimi poizvedbami, ki postajajo čedalje bolj nepogrešljiv del pri razvijanju odzivnih spletnih strani. Na koncu poglavja so opisani še skriptni programski jezik JavaScript, vtičnik jQuery in osnove strežniškega skriptnega jezika PHP. Ker se skripte v PHP izvajajo na strežniku le pri aplikaciji *InfoFRI admin*, je njihov podrobnejši opis v poglavju 8.

4.1 HTML

HTML (ang. Hypertext Markup Language) je skriptni jezik za označevanje nadbesedil [19], namenjen izdelavi spletnih strani. Predstavlja osnovo spletnih dokumentov HTML, v katerih uporablja elemente HTML[1] kot osnovne gradnike za definiranje strukture dokumenta. Spletni brskalniki znajo dokumente HTML ustrezno interpretirati, besedilne in multimedijske vsebine pa (brez elementov) prikazati na spletni strani. Na začetku dokumenta HTML brskalnik prebere deklaracijo (npr. `<!doctype>`), ki definira upo-

[1]Elemente označujemo tudi kot značke (tags), ki so definirane med znakoma `<` in `>`, v dokumentu HTML pa lahko stojijo samostojno (`<br />`) ali v parih (`<p></p>`).

rabo določene verzije jezika HTML in prikaz dokumenta HTML. Ogrodje dokumenta HTML so deklaracija dokumenta in obvezni elementi html, head, body in title. Medtem ko so v glavi zapisani naslov dokumenta, povezave do stilov in skript ter metapodatki, je v telesu vsebina, ki bo prikazana. Elementom HTML lahko priredimo še atribute za določanje lastnosti in proženje dogodkov.[2]

```
<!DOCTYPE html>
<html>
    <head>
        <title>Naslov spletne strani</title>
        <link href="style.css" rel="stylesheet">
    </head>
    <body>
        <p style="color: green;">Hello World!</p>
    </body>
</html>
```

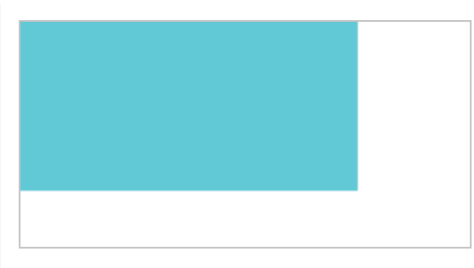
Slika 4.1: Primer osnovnega dokumenta HTML z metapodatki in atributi.

4.1.1 Canvas

Element *canvas* je del HTML5 in se uporablja za risanje geometrijskih likov in prikazovanje slik. V obliki pravokotne površine ga na spletno stran postavimo kot element HTML in deluje kot (slikarsko) platno brez robov ali vsebine. jCanvas je JavaScript knjižnica, ki vsebuje različne metode za enostavnejše risanje likov, teksta, dodajanja slik itd. Poleg pripravljenih metod iz omenjene knjižnice lahko z JavaScriptom tudi sami dostopamo do elementa *canvas* in sprogramiramo risanje poljubnih geometrijskih likov. Element *canvas* dodamo v telo dokumenta HTML in mu določimo poljubno velikost, tako kot prikazuje slika 4.2.

[2]S skriptnim jezikom JavaScript napišemo funkcije, ki se izvajajo ob proženju dogodka.

```
<canvas id="myCanvas" width="200px" height="200px"></canvas>

<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    ctx.fillStyle = "#00F0F0";
    ctx.fillRect(0,0,150,75);
</script>
```



Slika 4.2: V telo dokumenta HTML postavimo element *canvas*, znotraj elementa *script* v dokumentu HTML (ali pa v ločeni datoteki .js) pa lahko na primer sprogramiramo risanje geometrijskih likov.

4.2 CSS

CSS (ang. Cascading Style Sheets) so stilske predloge, s katerimi lahko elementom HTML določimo različne oblikovne lastnosti [19], npr. barvo (ozadja), pozicijo, velikost, animacije itd. Pravila CSS lahko dodamo v dokument HTML (1) elementom HTML z uporabo atributa `style`, (2) v glavi dokumenta HTML znotraj elementa `<style>` ali (3) v ločeni datoteki .css, ki jo vključimo v dokument HTML. Zadnji način je najbolj eleganten in priporočljiv, saj dobimo tako večjo sistematičnost, fleksibilnost in pregled nad določanjem oblikovnih lastnosti, prav tako pa lahko tudi več spletnih strani uporablja isto datoteko. Pravila CSS lahko določimo posameznim selektorjem, ki predstavljajo elemente HTML, in njihovim atributom (id, class).

Trenutno je v uporabi zadnja verzija CSS3, ki omogoča tudi animacije, tranzicije, gradiente in implementacijo odzivnih (responsive) spletnih strani s pomočjo medijskih poizvedb (ang. media queries). Več o odzivnih spletnih straneh in medijskih poizvedbah je zapisano v podpoglavju 4.2.1.

```
body {
    background-color: #fff;
    font-family: sans-serif, "Calibri Light";
}
.all-submit-buttons {
    border: 1px solid grey;
}
a#link-id {
    text-decoration: none;
}
a#link-id:hover {
    color: green;
}
```

Slika 4.3: Primer selektorjev v CSS.

4.2.1 Medijske poizvedbe

Medijske poizvedbe (ang. media queries) so modul najnovejših stilskih predlog CSS3 in standard W3C, ki omogoča prilagajanje vsebine na spletni strani glede na ločljivost različnih zaslonov (pri pametnih telefonih, tablicah ali prenosnikih itd.). Medijska poizvedba je sestavljena iz medijskega tipa (ang. media type), nič, enega ali več izrazov (pogojev) ter (neobveznega) logičnega operatorja (and, or, only itd.). Medijske poizvedbe so logični izrazi, ki so pravilni ali napačni, implementirani pa so kot del stilskih predlog v CSS [11].

Rezultat medijske poizvedbe je pravilen, če se tip medija v medijski poizvedbi ujema s tipom izhodne naprave (npr. z ločljivostjo monitorja ali pametnega telefona), na kateri je prikazana spletna stran, in če so vsi izrazi v medijski poizvedbi pravilni. Takrat se izvedejo pravila CSS znotraj medijske poizvedbe. Pri napačni medijski poizvedbi ali če se ta ne nanaša na pravo izhodno napravo, jo brskalnik ignorira [11]. Pri spodnji medijski poizvedbi mora biti izhodna naprava monitor z največjo ločljivostjo 1280 slikovnih elementov.

```
@media screen and (max-width: 1280px) { … }
```

Medijske poizvedbe definirajo naslednje medijske tipe (našteti so le osnovni) [11]:

- **all** – vse vrste medijev,
- **screen** – računalniški zasloni,
- **print** – tiskalniki,
- **braille** – braillove naprave,
- **handheld** – prenosne naprave (pametni telefoni, tablice),
- **tv** – televizorji in
- **tty** – terminali.

Brez medijskih poizvedb je pravzaprav težko narediti odzivno (ang. responsive) spletno stran, zato je v zadnjem času nastalo nekaj orodij, ki omogočajo enostavnejšo implementacijo odzivnih spletnih strani. Eno izmed takšnih je **Bootstrap** – knjižnica JavaScript in jQuery, ki omogoča prilagodljivejšo implementacijo odzivnih spletnih strani in spletnih vmesnikov[3] (ang. frontend). Z orodjem Bootstrap lahko zastavimo osnovno ogrodje spletnih strani, saj vključuje različne stilske predloge, sodoben dizajn in interakcijo za elemente HTML (npr. tabele, obrazci, gumbi, navigacija itd.) ter nekatere že pripravljene knjižnice, predloge (ang. templates) in vtičnike v jeziku JavaScript (npr. spustni meniji, animacija slik, interaktivni navigacijski meniji itd.).

Ker smo se odločili, da bo aplikacija *InfoFRI točka* prilagojena za tablice, večdotično površino in kot spletna aplikacija, je bilo treba implementirane stvari v aplikaciji (zemljevide, slike, prosojnice, pisavo itd.) ustrezno prilagoditi glede na različne ločljivosti naprav. Čeprav smo uporabili knjižnico *bootstrap.js* z nekaterimi že vsebovanimi medijskimi poizvedbami, smo pri *InfoFRI točki* implementirali kar nekaj dodatnih medijskih poizvedb, da je njen izgled na različnih napravah (večdotična površina, tablice, monitor) z različnimi ločljivostmi enak. Medijske poizvedbe so pri *InfoFRI točki* implementirane v datoteki *main.css*. Na sliki 4.4 je predstavljenih nekaj primerov

[3]Odzivni razvoj spletnih strani (ang. responsive web design) omogoča izdelavo in oblikovanje spletnih strani, ki so prilagodljive zaslonom različnih ločljivosti (od pametnih telefonov in tablic do velikih namizij). Bootstrap je implementiran za najnovejšo verzijo – HTML5.

poizvedb iz omenjene datoteke.

```
@media (max-width: 1280px) and (min-width: 979px) {
    .navbar-wrapper { height: 15px; }
    .navbar-inner { height: 10px; }
    .meni_links { font-size: 0.8em; }
    .carousel-inner { height: 200px; }
    .carousel-caption > h1 { font-size: 1.7em; }
    .carousel-caption > .lead { font-size: 1.0em; }
    .about_news { font-size: 0.9em; }
    .span4 h2 { font-size: 1.4em; line-height: 1.2em; }
    #news_list { height: 350px; }
    .span4 { height: 350px; }
    .news_read_more { margin-top: 10px; }
    #news_container, #news_container_inner  { height: 100%; }
    .headline { line-height: 20px; font-size: 1.5em; }
    .big_img { width: 550px; height: 400px; margin-left: 10px; }
    div#map_model { width: 320px; height: 210px; }
    #search_input { width: 210px; }
    #canvas { margin-top: -70px; }
    #big-picture { max-width: 600px; margin-left: 10px; }
    table.dataTable { font-size: 0.75em; }
    #news_container_inner_data .text p { text-align: justify; padding-right: 40px; }
    #news_container_inner_data .summary p { text-align: justify; padding-right: 40px; }
}
```

Slika 4.4: Del medijskih poizvedb iz datoteke *main.css*, ki prilagodijo *InfoFRI točko* različni ločljivosti naprav: prikazan primer pomeni prilagodljivost glede na širino zaslona od 979 do največ 1280 slikovnih elementov.

4.3 JavaScript

JavaScript je visokonivojski dinamični skriptni programski jezik, narejen za (interaktivno) spletno programiranje na strani odjemalca. Medtem ko sta HTML in CSS namenjena prikazu vsebine oz. oblike spletne strani, lahko z JavaScriptom implementiramo vedenje vsebine (animacije, efekti) spletne strani, prožimo dogodke na spletni strani ali dinamično dostopamo in posodabljamo vsebino ter strukturo spletnih strani (dinamično kreiranje kode HTML) [19]. Ko se spletna stran naloži, ustvari brskalnik model DOM, prek katerega lahko dostopa JavaScript do vseh elementov HTML, njihovih atributov in stilskih predlog (CSS) ter dinamično omogoča njihovo spreminjanje, dodajanje ali odstranjevanje. Večina spletnih strani je implementiranih z JavaScriptom, prav tako ga podpirajo tudi vsi novejši brskalniki, ki imajo vgrajene prevajalnike za JavaScript.

4.3.1 jQuery

Za naprednejše spletno programiranje je bilo implementiranih veliko JavaScript knjižnic in orodij. Eno izmed najbolj popularnih in razširjenih je odprtokodna knjižnica **jQuery**, ki uporablja selektorje CSS za dostop in manipulacijo elementov HTML na spletni strani. Primer dogodka jQuery je na sliki 4.5.

Z uporabo knjižnice jQuery je nastalo več kot tisoč odprtokodnih (in plačljivih) vtičnikov, ki omogočajo implementacijo že pripravljenih interaktivnih efektov ali animacij. Zaradi tega je raba knjižnic jQuery v zadnjem času narasla. jQuery poenostavlja sintakso JavaScripta in ponuja boljšo interakcijo med JavaScriptom in ostalimi spletnimi programskimi jeziki. Zagotavlja lažji dostop do modela DOM in lažje kreiranje dinamičnih spletnih segmentov (Ajax) v primerjavi z JavaScriptom.

Pri implementaciji aplikacije *InfoFRI točka* smo uporabili tudi dva vtičnika jQuery za implementacijo virtualne tipkovnice in manipulacijo s tabelami,

in sicer *jQuery On-Screen Keyboard* in *DataTables*. Obema smo izgled prilagodili *InfoFRI točki* in dodali še funkcionalnosti. Več o vtičnikih je mogoče prebrati v podpoglavjih 7.3 in 7.4.

```
<!-- index.html -->
<head>
    <script type='text/javascript' src='js/jquery-latest.min.js'></script>
    <script type='text/javascript' src='js/myFunctions.js'></script>
</head>

// myFunctions.js
$(document).ready(function() {
    $("#my-div").click(function() {
        $(this).hide();
    });
});
```

Slika 4.5: Primer dogodka v jQuery, pri katerem se skrije element z id-jem $\#my-div$, ko kliknemo nanj.

4.4 PHP

PHP (ang. PHP: Hypertext Preprocessor) je odprtokodni skriptni programski jezik, namenjen programiranju na strani strežnika. Z njim je mogoče generirati dinamično vsebino spletnih strani, pošiljati in prejemati piškotke, neposredno dostopati, spreminjati in kriptirati podatke v datotekah na strežniku ter v podatkovni bazi, urejati nadzor za dostop uporabnikov do strežnika itd. [19]

Čeprav obe aplikaciji, razviti v okviru tega diplomskega dela, temeljita predvsem na prej opisanih spletnih tehnologijah, smo potrebovali tudi PHP. Prva skripta *imgToBase64.php* je namenjena pretvorbi slik v nize po standardu Base64 (podpoglavje 8.4), druga za izvoz dokumenta JSON iz podatkovne baze CouchDB in prenos na strežnik v imenik aplikacije *InfoFRI admin*, tretja skripta pa za premik dokumenta JSON iz imenika aplikacije *InfoFRI admin* v imenik *InfoFRI točke* na strežniku (podpoglavje 8.5).

Poglavje 5

NoSQL

5.1 Opis podatkovnih baz NoSQL

NoSQL je sistem za upravljanje z nerelacijskimi podatkovnimi bazami, ki zagotavlja mehanizem za shranjevanje in iskanje podatkov. Izraz NoSQL je leta 1998 prvi uporabil Carlo Strozzi, ko je poimenoval svojo odprtokodno relacijsko podatkovno bazo *Strozzi NoSQL*, ki sicer ni podpirala standarda SQL, kljub temu pa je bila relacijska. Zaradi tega je izraz NoSQL najprej pomenil "non SQL (ne SQL)", kasneje pa se je zanj uveljavil pojem **Not only SQL**, kar poudari dejstvo, da podpira tudi poizvedbene jezike SQL [15].

V zadnjih nekaj letih je raba nerelacijskih podatkovnih baz NoSQL zelo narasla, vsekakor pa tradicionalnih, relacijskih podatkovnih baz vendarle ne bo izpodrinila. Medtem ko lahko v tabelah relacijskih podatkovnih baz shranjujemo le strukturirane podatke, podpirajo nerelacijske podatkovne baze NoSQL shranjevanje tako strukturiranih kot tudi nestrukturiranih podatkov. Primeri nestrukturiranih podatkov so na primer vsebine medijskih ali dnevniških datotek, sporočila na družbenih omrežjih ali blogih [14].

Lastnosti podatkovnih baz NoSQL:

- (horizontalna) skalabilnost: lahko upravljajo z veliko količino podatkov; možnost delovanja na gručah računalnikov (ang. clusters);
- učinkovita performanca: tudi pri visokih obremenitvah so dovolj zmogljive in hitre pri obdelavi podatkov;
- visoka razpoložljivost: nekatere podatkovne baze NoSQL (npr. Cassandra) imajo vgrajene mehanizme, ki rešujejo zaplete pri odpovedih strojne opreme;
- brezshematskost: nimajo tabel, vrstic ali stolpcev – dokumentno usmerjena podatkovna baza vsebuje dokumente, v katerih so shranjeni podatki po modelu ključ-vrednost;
- hitra prilagojenost spremembam pri razvoju;
- enostavnost dostopa prek spletnega uporabniškega vmesnika [14].

Ker so nerelacijske podatkovne baze NoSQL zmožne upravljati z veliko količino podatkov[1] pri visoki obremenitvi, jih uporabljajo tudi velika svetovna podjetja, kot so Facebook, Google, Amazon[2] ipd., pri katerih se dnevno procesira več sto milijonov branj in pisanj v manj kot sekundi [16].

Uporaba nerelacijskih podatkovnih baz NoSQL je torej priporočljiva pri:

- upravljanju z velikimi količinami podatkov,
- hkratnih pogostih dostopih do podatkov,
- delu s hitro rastočimi seznami elementov: sporočila na družbenih omrežjih, dnevniški zapisi, blogi ...,
- nestrukturiranih podatkih oz. podatkih s spreminjajočo se strukturo,
- omejitvah (ang. constraints), ki niso zahtevane pri načrtovanju podatkovne baze [15].

[1] Pri ogromnih količinah podatkov lahko govorimo o pentabajtih (10^{15}) bajtov.

[2] Kot zanimivost je na tem mestu navedenih nekaj primerov nerelacijskih podatkovnih baz, ki jih uporabljajo svetovno znana podjetja: Cassandra (Facebook – pri iskanju po sporočilih), BigTable (Google), Voldemort (LinkedIn). Nekatera podjetja so svoje podatkovne baze od začetka implementirala za svoje interese (BigTable).

Znotraj podatkovnih baz NoSQL obstajajo štiri kategorije, ki se razlikujejo po načinu shranjevanja, pridobivanja in obdelave shranjenih podatkov.

- **Zbirka ključ-vrednost (ang. Key-Value Stores)**: podatki so shranjeni v slovarju (Hash table) z enoličnim ključem in kazalcem na določeno vrednost. Prek ključev je mogoče shranjevanje in pridobivanje podatkov. Primeri: Riak, Voldemort, Redis, Memcached, HamsterDB [13].

- **Zbirka dokumentov (ang. Document Stores)**: podatke se shranjuje in obdeluje v dokumentih (npr. JSON). Dokumenti so lahko različno strukturirani, prav tako lahko vsebujejo druge dokumente. Podatki so znotraj dokumentov shranjeni po modelu ključ-vrednost. Primeri: MongoDB, **CouchDB**, Cloudant [13].

- **Zbirke stolpcev (ang. Column Family Stores)**: vsaka vrstica ima enolično identifikacijsko številko in vsebuje več stolpcev, v katerih so podatki shranjeni kot ključ-vrednost. Imena stolpcev morajo biti vnaprej določena (predstavljajo ključe), podatki v posameznih stolpcih pa predstavljajo vrednosti. Ta kategorija je primerna za obdelavo velike količine podatkov. Primeri: Cassandra, HBase, DynamoDB, BigTable [13].

- **Grafne podatkovne baze (ang. Graph Databases)**: podatki so shranjeni v grafnih strukturah. Entitete so shranjene kot vozlišča, povezave med njimi (lastnosti) pa kot robovi. Ta tip podatkovne baze je primeren za analizo podatkov, organizacija grafa pa omogoča interpretacijo na podlagi različnih razmerij in vzorcev. Primeri: Neo4J, Infinite Graph, OrientDB [13].

5.1.1 JSON

JSON (ang. JavaScript Object Notation) je minimalen, berljiv tekstovni format, prvotno namenjen prenosu podatkov med strežnikom in odjemalcem (spletno aplikacijo) kot alternativa XML. Ima hierarhično strukturo, glavni sestavni deli pa so ključi (keys) in vrednosti (values), ki so povezani v pare ključ-vrednost. Ključ zmeraj predstavlja niz med dvema narekovajema. Vrednost pa je lahko niz, število, logični tip (boolean), seznam ali objekt. Vrednosti so od pripadajočih ključev ločene z dvopičjem, pari ključ-vrednost pa z vejicami. Celoten dokument obdajata zavita oklepaja { }. Zaradi takšne strukture je JSON enostaven za razčlenjevanje (parsanje) in neodvisen od programskega jezika. Končnica datoteke je .json. Na sliki 5.1 je primer zapisa v formatu JSON [10].

```
{
  "_id": "personnel_101",
  "_rev": "1-e565a97d5cf25851ae92f9068a4de7de",
  "id": 101,
  "position": 2,
  "title": "Profesor",
  "name": "Janez",
  "surname": "Novak",
  "room": "R3.26",
  "roomId": 94,
  "lab": "LGM",
  "description": "Dekan",
  "field": "personnel"
}
```

Slika 5.1: Primer dokumenta JSON iz podatkovne baze aplikacije *InfoFRI točka*. Podatki so izmišljeni, saj je namen prikazati izključno strukturo dokumenta JSON.

Poglavje 6

CouchDB

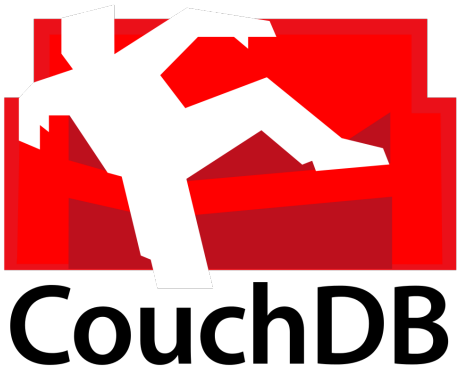
Celotna tematika diplomskega dela posega predvsem na področje spletnih tehnologij, zato smo se – da bi predstavili razmeroma novo ogrodje za delo s podatkovnimi bazami – pri implementaciji zalednega dela (backend) odločili za uporabo nerelacijske podatkovne baze CouchDB. Vsekakor CouchDB ni edina tovrstna nerelacijska podatkovna baza, ima pa kot platforma oz. orodje na področju računalništva dovolj dolgo tradicijo in reference, ki so nas prepričale v uporabo pri realizaciji vseh potrebnih zahtev, povezanih s shranjevanjem, pridobivanjem in urejanjem podatkov v podatkovni bazi *InfoFRI točke*.

6.1 Opis in zgodovina CouchDB

CouchDB je odprtokodna dokumentno usmerjena nerelacijska podatkovna baza NoSQL, ki uporablja format JSON za shranjevanje podatkov v dokumentih, skriptni jezik JavaScript za pisanje poizvedb in spletni grafični uporabniški vmesnik Fauxton za dostop in urejanje podatkovne baze.

Začetek razvoja podatkovne baze CouchDB sega v leto 2005, ko je nekdanji razvijalec za Lotus Notes pri IBM Damien Katz na svojem blogu napovedal razvoj novega jedra nerelacijske podatkovne baze. Označil ga je kot sistem za shranjevanje skalabilne objektne podatkovne baze, poimenovavši

ga CouchDB – pri čemer gre za kratico **C**luster **O**f **U**nreliable **C**ommodity **H**ardware. Njegov cilj je bil, da bi CouchDB postal podatkovna baza spleta, prilagojena spletnim aplikacijam. Po dveh letih razvoja ga je objavil kot odprtokodni projekt pod licenco GNU [4]. Na sliki 6.1 je prikazan logo CouchDB.



Slika 6.1: Logo podatkovne baze CouchDB. Obris zleknjenega človeka na kavču lahko razumemo kot metaforo za prilagodljivost CouchDB.

Pri razvoju CouchDB se je Katz odločil za programski jezik C++, prav tako je bil CouchDB že na začetku razvoja mišljen kot brezshematska in indeksirana podatkovna baza. Na razvoj CouchDB so zagotovo vplivale tudi Katzove izkušnje s platformo Lotus Notes, saj je pri implementaciji CouchDB omenil, da bo naredil CouchDB od začetka kot nekakšen Lotus Notes za splet. Tako je decembra 2005 na svojem blogu objavil glavne funkcije, ki jih bo imel CouchDB, in sicer shranjevanje podatkov v dokumentih, distribuirana arhitektura, obojestranska replikacija in dostop do podatkovne baze (tudi) brez internetne povezave [4].

Leto dni po začetku razvoja se je Katz odločil za novo, boljšo implementacijo jedra podatkovne baze CouchDB v programskem jeziku Erlang,[1] saj naj bi

[1]Erlang je programski jezik, ki se uporablja za gradnjo množičnih razširljivih mehkih sistemov, delujočih v realnem času z zahtevami po visoki razpoložljivosti. Njegova raba sega na področja telekomunikacij, bančništva, računalniške telefonije, takojšnjega sporočanja (ang. instant messaging). Ima vgrajeno podporo za sočasnost, distribucijo in toleranco pri odpovedih strojne ali programske opreme (ang. fault tolerance) [7].

bil omenjeni programski jezik bolj primeren za realizacijo vseh zastavljenih ciljev pri implementaciji CouchDB. Zaradi specifičnih lastnosti je njegova raba namreč visoko zastopana pri razvoju telekomunikacijskih tehnologij, s poudarkom na nadzoru sočasnosti (ang. concurrency control), toleranci odpovedi strojne opreme (ang. fault tolerance) in distribuiranih aplikacijskih vmesnikih (ang. distributed applications), kar je kajpak ustrezalo Katzovi viziji razvoja CouchDB [4].

Februarja 2008 je postal CouchDB del projektov podjetja Apache – sprva v okviru Apache inkubatorja, čez nekaj mesecev pa kot del njihove fundacije (Apache Software Foundation). Julija 2010 je izšla prva stabilna verzija CouchDB, zaščitena z licenco Apache Licence. Damien Katz je kasneje opustil razvoj projekta CouchDB in se osredotočil na razvoj projekta Couchbase Server. Podjetje Apache je nadaljevalo z razvojem podatkovne baze CouchDB in vsako nadaljnje leto objavilo novejšo verzijo CouchDB. Trenutna veljavna verzija CouchDB je 1.6.1, izšla pa je septembra 2014 [4].

6.2 CouchDB in B+ drevesa

Osnovo podatkovne baze CouchDB predstavljajo B+ drevesa, podatkovna struktura, ki zagotavlja jedro mehanizma za shranjevanje, vstavljanje, iskanje in brisanje podatkov ter kreiranje dokumentov in pogledov (ang. views) v logaritmičnem času O(log n) [1, str. 13, 14].

Ker je CouchDB brezshematska podatkovna baza brez primarnih in tujih ključev, je za poljubna razmerja in povezave med dokumenti treba ustvariti poglede. Rezultati tovrstnih pogledov so izračunani na podlagi modela MapReduce,[2] ki je primeren za procesiranje in ustvarjanje velikih količin po-

[2]Model MapReduce deluje po naslednjem principu: glavno vozlišče v verigi porazdeljenega računanja vzame dokument, razdeli problem na več manjših podproblemov ter jih razporedi med ostala vozlišča v porazdeljenem sistemu. Vsako izmed vozlišč ustrezno opravi svoje delo in vrne rešitve glavnemu vozlišču. V zadnji fazi glavno vozlišče zlije prejete podatke od porazdeljenih vozlišč in jih sestavi v celovito rešitev.

datkov s pomočjo porazdeljenega računanja. Funkcije MapReduce pri CouchDB kreirajo pare ključ-vrednost, ki jih CouchDB nato vstavi v B+ drevesa in uredi po ključih. To omogoča učinkovito iskanje po ključih in izboljšano delovanje znotraj B+ dreves. Iskanje dokumentov v B+ drevesu poteka torej prek ključev [1, str. 233–235].

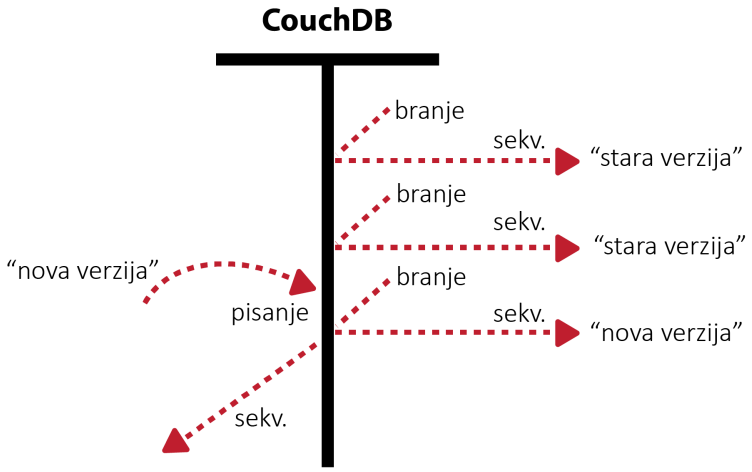
B+ drevo je iskalno uravnoteženo drevo, pri katerem so vsi podatki shranjeni v listih drevesa, vmesna vozlišča pa so pravzaprav le odločitve oz. kazalci, do katerih listov se je potrebno usmeriti, da pridemo do pravega podatka. V CouchDB shranjujejo B+ drevesa podatkovne baze in poglede. Vsakokrat, ko posodobimo podatkovno bazo oz. dokument, pripne CouchDB dokumente zmeraj na koncu (v liste) B+ drevesa, prav tako pa se posodobijo tudi informacije v korenu drevesa. Starejše verzije dokumentov se arhivirajo, saj do njih še zmeraj obstajajo povezave. Prek starejšega korena je mogoče dostopati do prejšnjih verzij dokumentov oz. podatkovne baze, do starejšega korena pa kaže kazalec iz novega korena [1, str. 233–235].

6.3 Nadzor sočasnih dostopov v CouchDB

Tradicionalne relacijske podatkovne baze uporabljajo sistem za zaklepanje pri sočasnih dostopih za pisanje. Če poskuša več virov hkrati dostopati in zapisovati v podatkovno bazo, relacijske podatkovne baze omogočajo sistem zaklepanja. Pri tem lahko operacijo pisanja zmeraj izvaja le en vir naenkrat (upoštevavši vse vire, poteka pisanje v podatkovno bazo zmeraj zaporedno). Pridobivanje pravic za spreminjanje gre po principu vrste FIFO. Ko poskuša več virov hkrati spreminjati podatke v podatkovni bazi, dobi prvi vir pravico za pisanje in najprej zaklene vsebino. Medtem ko spreminja vsebino v bazi, morajo ostali viri čakati tako dolgo, da zaključi z operacijo pisanja, odklene vsebino in tako naslednjemu viru omogoči spreminjanje vsebine. Čeprav gre za učinkovit sistem razporejanja in preprečevanja hkratnih dostopov, lahko relacijska baza pri visokih obremenitvah porabi zelo veliko procesorske moči za koordinacijo zaklepanja vsebin.

Za koordiniranje sočasnih dostopov in spreminjanja vsebin v bazi uporablja CouchDB metodo multiverzijskega sočasnega nadzora (ang. Multiversion Concurrency Control – MVCC), pri katerem je vsakemu odjemalcu za spreminjanje na voljo zadnja verzija dokumenta. Dokumenti so v CouchDB določeni z verzijami. Ko odjemalec spreminja določen dokument, naredi CouchDB novo verzijo tega dokumenta in jo shrani v bazo. Tako obstajata v CouchDB starejša in novejša verzija istega dokumenta. Multiverzijski sočasni nadzor omogoča več odjemalcem, da lahko hkrati naložijo in urejajo isti dokument, deluje pa podobno kot sistem za nadzor različic Subversion [1, str. 15].

Denimo, da ima prvi odjemalec bralni dostop do poljubnega dokumenta v podatkovni bazi, takisti dokument pa želi spreminjati drugi odjemalec. Ker prvi odjemalec ničesar ne spreminja, ureja drugi odjemalec najnovejšo verzijo dokumenta, zato bo CouchDB brez konfliktov uveljavil njegove spremembe v dokumentu – ne da bi pri tem čakal zaključitev bralnih dostopov. Ko želi naslednji odjemalec bralni dostop do istega dokumenta, bo CouchDB ponudil najnovejšo verzijo dokumenta, ki je bila posodobljena nazadnje. Postopek je prikazan na sliki 6.2.



Slika 6.2: Nadzor sočasnih dostopov pri CouchDB [1, str. 15].

Tak način sočasnega nadzora omogoča več odjemalcem sočasno spreminjanje istega dokumenta. Če pa poskušajo odjemalci sočasno uveljavljati spre-

membe, prejmejo obvestilo o konfliktu, da je treba najprej naložiti najnovejšo verzijo dokumenta in šele nato ponovno poskusiti uveljaviti spremembe.

6.4 Dokumenti v CouchDB

Dokumenti JSON so osrednja podatkovna struktura za shranjevanje podatkov v CouchDB. Vsak dokument ima lahko nedoločeno število polj, ki pri drugih dokumentih niso nujno definirana. Polja lahko vsebujejo različne tipe podatkov (nizi, številke ...) in niso omejena z velikostjo. Vsako polje mora imeti v dokumentu enolično ime. Ko shranimo dokument v bazo, doda CouchDB dokumentu dve metaoznaki, in sicer identifikacijsko (_id) in revizijsko (_rev) številko.

6.4.1 Identifikacijska številka (_id)

Identifikacijske številke so nizi, ki jih lahko sami poljubno izberemo. Pri *Info-FRI točki* smo sestavili nize identifikacijskih številk iz kategorij "zaposlenih" in "prostorov", v for zanki pa smo jim priredili še številske vrednosti glede na število vnosov. CouchDB omogoča še avtomatsko prirejanje identifikacijskih številk na podlagi univerzalnega enoličnega identifikatorja (ang. Universally Unique IDentifier – UUID), ki jih zgenerira zgoščevalna funkcija in predstavljajo naključna števila z nizko možnostjo ponavljanja [1, str. 38–40].

6.4.2 Revizijska številka (_rev)

Ko želimo spreminjati določene vrednosti v dokumentu, je treba najprej z zahtevo GET pridobiti celoten dokument iz CouchDB, spremeniti vrednosti v dokumentu ali objektu in shraniti novo revizijo (verzijo) dokumenta v CouchDB. Vsaka revizija je označena z vrednostjo _rev.

Pri posodabljanju dokumenta pričakuje CouchDB vključitev revizijske številke (_rev) verzije dokumenta, ki jo želimo spreminjati. Ko CouchDB uveljavi novo verzijo spremenjenega dokumenta, z zgoščevalno funkcijo (tudi na pod-

lagi tega, kolikokrat je bil dokument posodobljen) zgenerira novo revizijsko številko za ta dokument. Odjemalci lahko sočasno spreminjajo isti dokument, pri shranjevanju pa morajo dodati v dokument še revizijsko številko. Če se revizijska številka ne ujema s tisto na strežniku, pomeni, da odjemalec nima najnovejše verzije dokumenta, zato vrne CouchDB sporočilo o konfliktu. Najprej je treba poslati zahtevo za pridobitev najnovejšega dokumenta, šele nato je mogoče uveljaviti spremembe. Kadar se revizijski številki ujemata, bo CouchDB sprejel spremembe, saj je bila spreminjana najnovejša verzija dokumenta. Če pri uveljavljanju sprememb ne pošljemo revizijske številke _rev, CouchDB razume, da želimo posodabljati starejšo verzijo dokumenta (revizijska številka se v tem primeru prav tako ne ujema). Na sliki 6.3 so prikazani postopki (z ukazi v terminalu) ustvarjanja CouchDB in dokumentov, pridobivanja UUID ter uporabe revizijskih številk za posodabljanje sprememb [1, str. 38–40]. CouchDB uporablja protokol HTTP brez stanj (ang. stateless). Za pridobitev dokumenta pošlje odjemalec zahtevo GET, s čimer odpre povezavo, pridobi podatke in zapre povezavo, kar CouchDB omogoča lažjo koordinacijo in manjšo izrabo procesorske moči pri več sočasnih povezavah [1, str. 40].

```
(1) curl -X PUT 194.249.0.123:5984/test_database
{"ok":true}
(2) curl -X GET 194.249.0.123:5984/_uuids
{"uuids":["ac5dc83eae5f8a312c285efbd6005e06"]}
(3) curl -X PUT 194.249.0.123:5984/test_database/person_id -d '{"Name":"Janez", "Surname":"Novak"}'
{"ok":true,"id":"person_id","rev":"1-4f2518bf19c3ec916e09f99dcce71ff1"}
(4) curl -X PUT 194.249.0.123:5984/test_database/person_id -d '{"Name":"Peter", "Surname":"Novak"}'
{"error":"conflict","reason":"Document update conflict."}
(5) curl -X GET 194.249.0.123:5984/test_database/person_id
{"_id":"person_id","_rev":"1-4f2518bf19c3ec916e09f99dcce71ff1","Name":"Janez","Surname":"Novak"}
(6) curl -X PUT 194.249.0.123:5984/test_database/person_id -d
'{"_rev":"1-4f2518bf19c3ec916e09f99dcce71ff1", "Name":"Peter", "Surname":"Novak"}'
{"ok":true,"id":"person_id","rev":"2-8e558a9618023cc18322f3faf81cba3b"}
```

Slika 6.3: (1) Kreiranje CouchDB. (2) Pridobitev UUID. (3) Kreiranje in shranjevanje dokumenta, CouchDB vrne še revizijsko številko. (4) Če pri spreminjanju dokumenta ne vključimo polja _rev, vrne CouchDB obvestilo o konfliktu. (5) Z zahtevo GET izpiše CouchDB vsebino dokumenta. (6) Pri spreminjanju vključimo revizijsko številko, ki jo CouchDB shrani, dokumentu pa priredi novo rev. številko.

6.5 Replikacija

Replikacija naredi kopijo CouchDB ali sinhronizira kopiji iste podatkovne baze CouchDB, pri čemer uporabnikom zagotavlja nizko latenco za dostop do podatkov. Replikacija je mogoča znotraj istega ali prek oddaljenega strežnika.

CouchDB podpira dva načina replikacije: navadno (ang. triggered replication) in stalno replikacijo (ang. continuous replication), ki je pravzaprav sinhronizacija v pravem pomenu besede. Medtem ko je treba pri navadni replikaciji med podatkovnimi bazami CouchDB sprožiti sinhronizacijo, pri stalni replikaciji podatkovne baze stalno poslušajo in spremembe sproti med seboj sinhronizirajo. Če se spremeni ena kopija podatkovne baze, bo mehanizem replikacije avtomatsko poslal spremembe ostalim podatkovnim bazam. CouchDB ima vgrajen kompleksen algoritem, ki izbere idealen trenutek za sinhronizacijo podatkovnih baz. Tako je delovanje CouchDB zmeraj optimalno [1, str. 152].

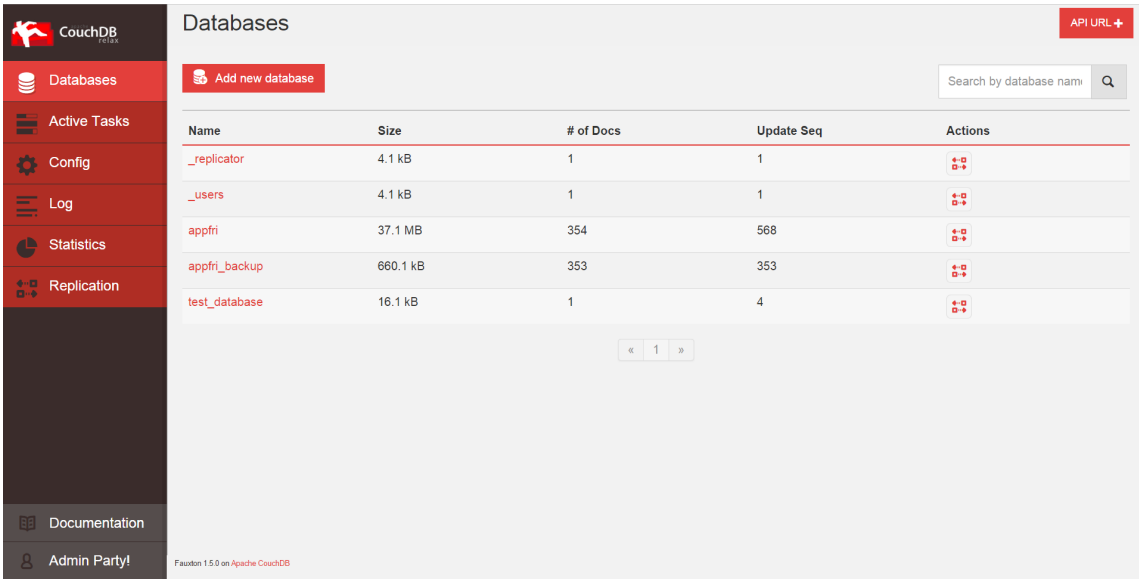
Ko se odločimo za navadno replikacijo, izberemo izvorno in ciljno podatkovno bazo bodisi na lokalnem ali oddaljenem strežniku. CouchDB med seboj primerja obe podatkovni bazi, nato pa podatke sinhronizira v obeh bazah. Pri posodabljanju dokumentov se prenesejo le novejše verzije dokumentov. Pri prekinitvi procesa replikacije (npr. izpad internetne povezave) pusti CouchDB obe podatkovni bazi v nekonsistentnem stanju. Ko po odpravljenih problemih ponovno zaženemo replikacijo, se proces nadaljuje, kjer se je ustavil. CouchDB vzdržuje še zgodovino replikacij [1, str. 8].

6.6 Fauxton – spletni uporabniški vmesnik

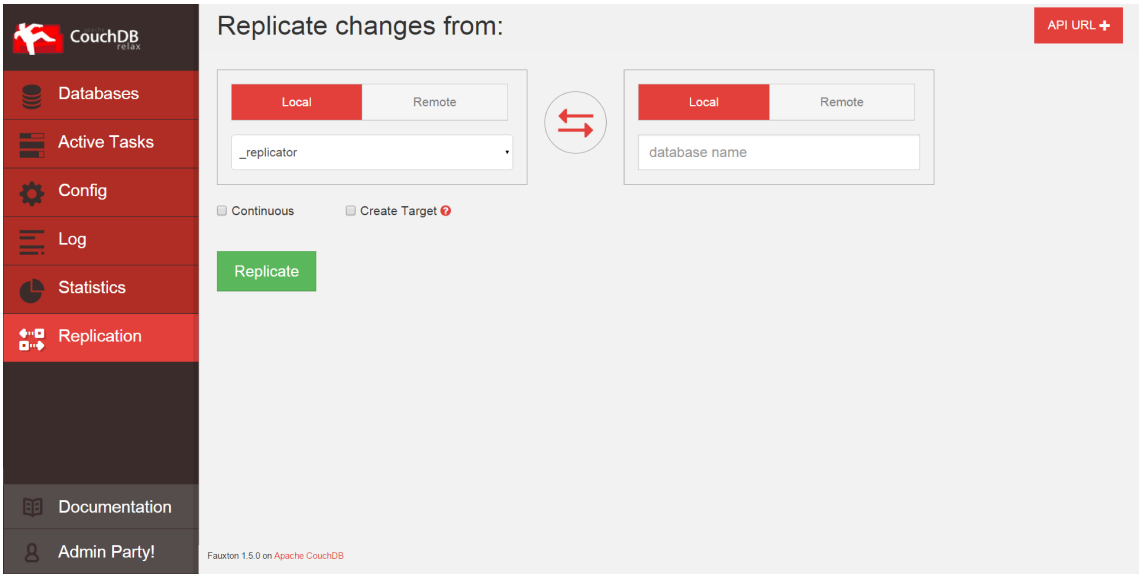
Fauxton je spletni uporabniški vmesnik – administratorsko orodje, vgrajeno v CouchDB za urejanje in dostop do podatkovne baze CouchDB. Z verzijo CouchDB 1.5 je nadomestil starejši spletni uporabniški vmesnik Futon. V osnovi Fauxton zagotavlja kreiranje, posodabljanje, brisanje podatkovne baze, dokumentov, pogledov in replikacijo ter statistično analizo in možnost administratorske prijave. Na slikah 6.4 in 6.5 sta primera uporabniškega vmesnika Fauxton.

Fauxton ima na levi strani navigacijsko vrstico s povezavami:

- **Databases:** prikaže seznam vseh podatkovnih baz, njihovo velikost, število dokumentov in revizij (posodobitev). Vsebuje še povezavo do sistema za replikacijo.
- **Active Tasks:** prikaže seznam procesov, ki tečejo na strežniku.
- **Config:** omogoča konfiguracijo namestitve CouchDB.
- **Replication:** omogoča povezavo do replikacijskega sistema CouchDB.
- **Log:** vsebuje dnevniške zapise.
- **Statistics:** statistični podatki o prijavah, bralnih, pisalnih dostopih ...
- **Replication:** omogoča kopiranje in sinhronizacijo podatkovnih baz CouchDB na lokalnem ali oddaljenem strežniku.
- **Documentation:** ponuja povezavo do spletne strani z dokumentacijo.
- **Admin Party:** omogoča dodajanje novih uporabnikov in spreminjanje gesla uporabnikom. Po privzetem načinu imajo vsi uporabniki enake administratorske privilegije (ni prijave). Kakor hitro določimo administratorja, ostali uporabniki izgubijo administratorske privilegije [8].



Slika 6.4: Spletni uporabniški vmesnik Fauxton.



Slika 6.5: Replikacija podatkovnih baz v Fauxtonu.

Poglavje 7

InfoFRI točka

7.1 Uvod v opis implementacije

Naj na kratko povzamemo opis spletne aplikacije *InfoFRI točka*. Kot interaktivni virtualni spletni kažipot po fakulteti omenjena aplikacija omogoča še prikazovanje novic, premikajočih se prosojnic z osnovnimi informacijami o fakulteti, iskanje po seznamih zaposlenih in prostorov, razvrščenih v tabelah, kot glavno funkcionalnost pa izrisovanje interaktivnih poti na prečiščenih arhitekturnih načrtih (tlorisih) zgradbe FRI in objekta X. Njeno implementacijo smo prilagodili za delovanje na večdotični površini, kot spletna aplikacija pa brezhibno deluje tudi na vseh mobilnih tablicah ali drugih računalnikih – kot večdotična ali navadna spletna aplikacija.

Sedmo poglavje predstavlja torej začetek empiričnega dela, v katerem bomo s tehničnega vidika po delih predstavili in opisali implementacijo *InfoFRI točke* z vsemi pomembnimi implementiranimi elementi, podprtimi s programsko kodo in zaslonskimi posnetki (slikami) – začenši s premikajočimi se novicami in prosojnicami. Sledil bo opis dveh vtičnikov jQuery za implementacijo virtualne tipkovnice (jQuery On-Screen KeyBoard) in tabel s podatki (DataTables). Opisan bo tudi razvoj vnosnih polj in spustnih menijev s pomočjo elementov div ter njihovo delovanje, ki je bilo razvito posebej za delovanje *InfoFRI točke* na večdotični površini, saj vgrajeni elementi HTML za vnosna

polja in spustne menije na večdotični površini ne delujejo. Sledil bo še opis priprave arhitekturnih načrtov zgradbe FRI in objekta X. V zadnjem poglavju je opisan še algoritem za risanje interaktivnih poti na tlorisih zgradbe FRI in objekta X.

7.2 Prikaz in premikanje novic in prosojnic

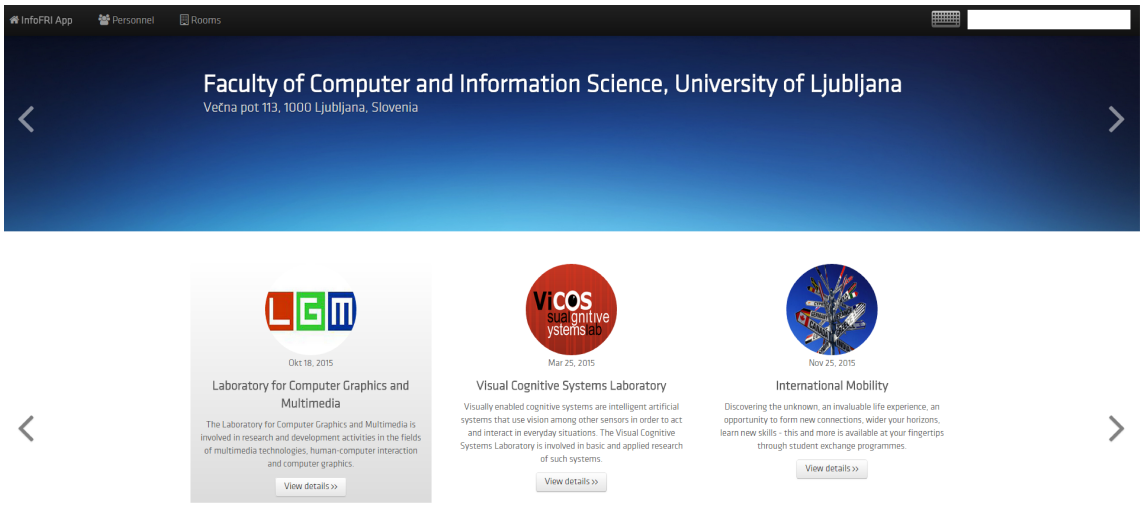
Na prvi strani *InfoFRI točke* so v ločenih razdelkih hkrati prikazane tri novice (slika 7.1), in sicer z manjšo sliko, naslovom, datumom, povzetkom in povezavo, ki preusmeri uporabnika na podrobnejše branje novice. Pri podrobnejšem branju sta uporabniku na voljo poleg naslova še večja slika (ta je lahko drugačna od manjše, prikazane na prvi strani) in daljše besedilo z informacijami. Vsaka novica vsebuje torej naslov, datum, povzetek in daljše besedilo ter manjšo in večjo sliko.

Novice je mogoče prek *InfoFRI admina* (poglavje 8) vstavljati v CouchDB, kjer so določene tudi omejitve pri vnašanju besedila, in sicer: 70 znakov za naslov, 40 znakov za datum,[1] 235 znakov za povzetek in 1500 znakov za daljše besedilo. Če uporabnik preseže omejeno število znakov, aplikacija odvečne znake odreže. Za manjše slike (ang. thumbnail) je priporočena velikost 260 x 190 slikovnih elementov, večja slika pa se glede na implementirane medijske poizvedbe samodejno prilagodi različnim ločljivostim zaslonov. Priporočeno razmerje večje slike je 9 : 6.

V *InfoFRI točko* je mogoče vnesti neomejeno število novic, hkrati pa so prikazane zmeraj le po tri (slika 7.1). Do vseh vnešenih novic se premikamo s puščicama levo ali desno, zmeraj po eno novico. Na prvi strani *InfoFRI točke* se privzeto prikažejo zmeraj prve tri novice iz CouchDB. Zaradi implementacije premikanja po novicah v levi in desni smeri se zdi, kot da se novice vrtijo v krogu.

[1] Zaželen format je npr. 6. september 2015.

Prosojnice na prvi strani predstavljajo nekakšen ohranjevalnik zaslona z uporabnimi informacijami o fakulteti in se premikajo samodejno ali s klikom na levo oz. desno puščico. Pri implementaciji smo uporabili vtičnik Bootstrap Carousel (datoteka *bootstrap-carousel.js*), prosojnicam pa smo prilagodili izgled in funkcionalnost. V aplikacijo je mogoče vnesti neomejeno število prosojnic, ki so lahko prazne (samo slika) ali pa vsebujejo še naslov (80 znakov) in krajše besedilo (450 znakov).



Slika 7.1: Zaslonski posnetek prve strani *InfoFRI točke* s prosojnicami in novicami.

7.3 Virtualna tipkovnica

Ker na večdotični površini ne moremo uporabljati računalniške tipkovnice, prav tako večdotična površina nima vgrajene svoje lastne (virtualne) tipkovnice, jo je bilo torej treba implementirati v *InfoFRI točki*.

Za virtualno tipkovnico smo izbrali obstoječ vtičnik jQuery On-Screen Keyboard in ga spremenili ter prilagodili aplikaciji tako, da ima odziven izgled in je primeren za različne ločljivosti. Poleg tega smo dodali še manjkajoče tipke (slika 7.2) za šumnike (velike in male črke) ter tipke za števila, piko, vejico, presledek in vračalko (ang. backspace). Odziven dizajn je v datoteki *virtual-keyboard.css*, implementacija in funkcionalnosti tipk pa v datotekah *master-run.js* in *jsKeyboard.js*. Tipkovnica se prikaže oz. skrije s klikom na ikono tipkovnice v aplikaciji desno zgoraj. Prav tako se tipkovnica prikaže, če izberemo vnosna polja za iskanje. Virtualna tipkovnica iz aplikacije je prikazana na sliki 7.3.

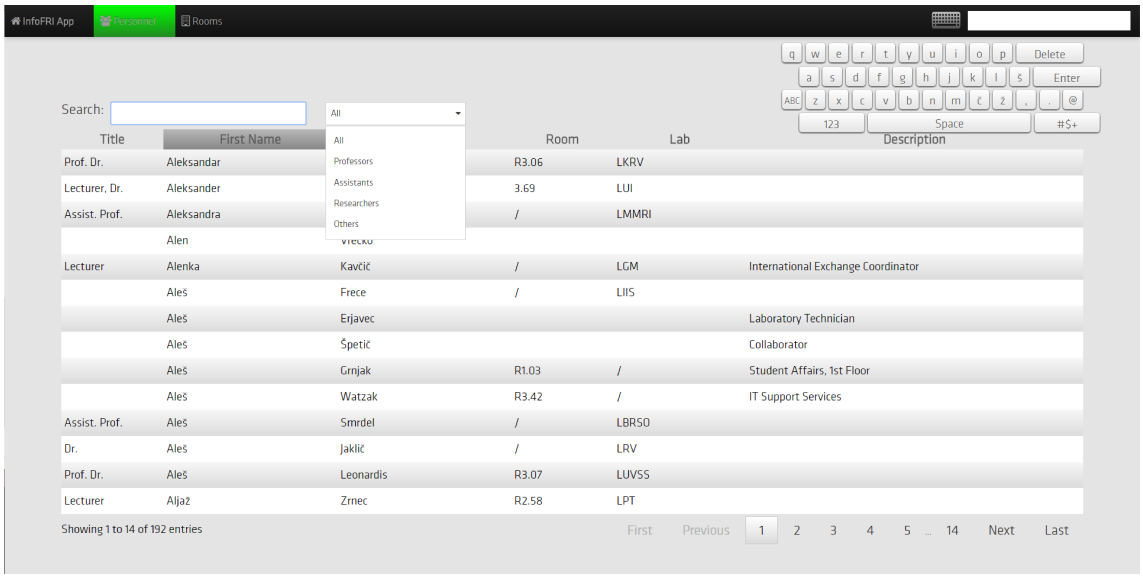
```
capitalLetter:  [
// 2nd row
       { value: 65, buttonClass: "button button_a" },{ value: 83 },{ value: 68 },{ value: 70 },
       { value: 71 },{ value: 72 },{ value: 74 },{ value: 75 },{ value: 76 },
       { value: "Š", isChar: "false", onclick: "jsKeyboard.writeDirect('Š');"},
       { value: "Enter", isChar: "false", buttonClass: "button button_enter",
       onclick: "jsKeyboard.enter();", keyClass: "key key_enter" },
// 3rd row
       { value: "abc", isChar: "false", buttonClass: "button button_smallletter",
       onclick: "jsKeyboard.changeToSmallLetter();", keyClass: "key key_smallletter" },
       { value: 90 },{ value: 88 },{ value: 67 },{ value: 86 },{ value: 66 },{ value: 78 },
       { value: 77 },{ value: "Č", isChar: "false", onclick: "jsKeyboard.writeDirect('Č');"},
       { value: "Ž", isChar: "false", onclick: "jsKeyboard.writeDirect('Ž');"},
       { value: 44 }, { value: 46 },{ value: 64 }
    ],
smallLetter: [
// 1st row
        { value: 113 },{ value: 119 },{ value: 101 },{ value: 114 },{ value: 116 },
        { value: 121 },{ value: 117 },{ value: 105 },{ value: 111 },{ value: 112 },
        { value: "Delete", isChar: "false", onclick: "jsKeyboard.del()",
        buttonClass: "button button_del", keyClass: "key key_del" }.
// 2nd row
        { value: 97, buttonClass: "button button_a" },{ value: 115 },{ value: 100 },{ value: 102 },
        { value: 103 },{ value: 104 },{ value: 106 },{ value: 107 },{ value: 108 },{ value: "š", isChar: "false",
        onclick: "jsKeyboard.writeDirect('š');"},
        { value: "Enter", isChar: "false", buttonClass: "button button_enter",
        onclick: "jsKeyboard.enter();", keyClass: "key key_enter" },
// 3rd row
        { value: "ABC", isChar: "false", buttonClass: "button button_capitalletterleft",
        onclick: "jsKeyboard.changeToCapitalLetter();",
        keyClass: "key key_capitalletterleft" },
        { value: 122 },{ value: 120 },{ value: 99 },{ value: 118 },{ value: 98 },
        { value: 110 },{ value: 109 },{ value: "č", isChar: "false",
        onclick: "jsKeyboard.writeDirect('č');"},
        { value: "ž", isChar: "false", onclick: "jsKeyboard.writeDirect('ž');"},
        { value: 44 },{ value: 46 },{ value: 64 },
    ],
```

Slika 7.2: Del kode, kjer so dodani še znaki za šumnike. Celotna koda je v datoteki *jsKeyboard.js*.

Slika 7.3: Virtualna tipkovnica v *InfoFRI točki*.

7.4 Izpis zaposlenih in prostorov v tabeli

Za izpis vnosov smo uporabili prilagodljivo ogrodje – jQuery vtičnik *DataTables*, ki omogoča napredno interakcijo tabel HTML. Med drugim omogoča številčenje strani (paginacijo), sprotno iskanje po tabeli (ang. live search) in razvrstitev od najmanjše do največje vrednosti (in obratno) po vseh stolpcih v tabeli (slika 7.4).



Slika 7.4: Izpis podatkov v tabeli. Prikazan je tudi spustni meni, ki omogoča filtracijo podatkov po glavnih kategorijah.

Tudi pri implementiranih tabelah smo prilagodili njihovo funkcionalnost in izgled za različne ločljivosti, tako da funkcija *this.numOfTableRows* dinamično preračuna, koliko vrstic naj se prikaže v tabeli (slika 7.5) glede na ločljivost zaslona. Število vrstic izračunamo tako, da od višine spletne strani odštejemo višino elementa div (header) in delimo z višino celice v tabeli. Rezultat zaokrožimo navzgor.

```
// Number of Table Rows According to Screen Resolution
this.numOfTableRows = function() {
    var row_number;
    row_number = Math.ceil(($(window.document).height() - 290) / 43);
    //alert(row_number);
    return row_number;
};
```

Slika 7.5: Prilagajanje velikosti tabele glede na ločljivost zaslona.

7.5 Implementacija vnosnih polj in spustnih menijev

Pri testiranju aplikacije na večdotični površini se je izkazalo, da dogodki ob kliku oz. dotiku pri vnosnih poljih (ang. input fields) in spustnih menijih (ang. dropdown menus) ne delujejo, zato je bilo treba njihovo delovanje simulirati z implementacijo elementov div (razdelkov). Namesto navadnih vnosnih polj in spustnih menijev smo torej oboje implementirali z elementi div. Razdelke, ki simulirajo vnosna polja, smo tudi ustrezno povezali z virtualno tipkovnico in tabelami, tako da se podatki iz baze ob vnosu črk prek virtualne tipkovnice v tabeli neposredno sproti filtrirajo. Najprej je bilo potrebno razdelke za vnosna polja in spustne menije dinamično ustvariti, njihovo funkcionalnost pa ločeno sprogramirati. Implementacija elementov div je pri vnosnih poljih v datotekah *div-input.js* in *jquery.dataTables.js*, implementacija spustnih menijev z elementi div pa v *select-input.js*. Programska koda implementacije razdelkov za vnosna polja je prikazana na sliki 7.6.

Za pristno obnašanje simuliranih vnosnih polj smo naredili še simulacijo kur-

zorja, in sicer tako, da navpična črta (|) utripa na pol sekunde (funkcija *_p.blink*). Funkcija *_p.updateText* vedno postavi kurzor (tj. pokončna črtica) za zadnjim znakom v div (tj. simulirano vnosno polje), v katerega se zapisuje besedilo iz spremenljivke. Kurzor torej utripa, ko je fokus v vnosnem polju (funkcija *_p.focus*). Kadar element div ni fokusiran, kurzor izgine (funkcija *_p.blur*). Vse funkcije so implementirane v datoteki *div-input.js*, prikazane pa na sliki 7.6.

```
var divInput = function (selector) {
    var _p = this;
    _p.text = '';
    _p.cursor = false;
    _p.cursorText = '<span class="blink">|</span>';
    _p.subDiv = document.createElement('div');
    _p.selector = selector;
    _p.subSelector = $(_p.subDiv);
    _p.events = {};
    _p.getText = function () { return _p.text; };
    _p.setText = function (text) { _p.text = text; };
    _p.updateText = function (cursor) {...};
    _p.focus = function () {...};
    _p.keydown = function (e) {...};
    _p.blur = function () {...};
    _p.click = function () {...};
    _p.blink = function () {...};
    _p.subscribeEvent = function (event, func, args, scope) {...};
    _p.callEvent = function (e) {...};
    selector.attr('tabindex', 0);
    selector.html(_p.subSelector);
    selector.focus(function () {...});
    selector.blur(function () {...});
    selector.click(function () {...});
    selector.keydown(function (e) {...});
    _p.subSelector.addClass('divInputSubDiv');
};
```

Slika 7.6: Funkcije, ki omogočajo simulacijo vnosnih polj in kurzorja.

Pri simulaciji spustnih menijev so vse opcije zapisane v elementih div, prikazan je le razdelek z izbrano opcijo. Celotno delovanje in integracijo spustnih menijev omogoča več funkcij: *this.init* dinamično kreira elemente div in jih prikaže v dokumentu HTML. Funkcija *this.addOption* v polje opcij doda vse opcije, ki bodo na izbiro pri spustnem meniju. Vsaka opcija ima svojo identifikacijsko številko in vrednost. Za izris vseh elementov div in opcij ter puščice navzdol (s tem ponazorimo, da gre za spustni meni) se pokliče

funkcija *this.render*. V funkciji *this.initClicks* je implementirana ključna funkcionalnost simuliranih spustnih menijev. Ob kliku na element div se odpre seznam opcij (ostali razdelki), z izborom posamezne opcije iz seznama pa se njena vrednost zapiše v prikazan razdelek, vsi ostali razdelki z opcijami se skrijejo. Puščica se pojavi zmeraj v prikazanem razdelku. Del programske kode implementiranih spustnih menijev z razdelki je na sliki 7.7.

```
function selectInput (id, parentelement) {
    this.id = id;
    this.parentelement = parentelement;
    this.value = null;
    this.options = [];
    this.valueDiv = null;
    this.optionsDiv = null;
    this.change_function = null;
    this.html = "";
    this.arrow = '<i style="float: right;" class="fa fa-sort-desc"></i>';

    this.init = function() {…};
    this.setValue = function(id) {…};
    this.getValue = function() {…};
    this.addOption = function(id, text) {…};
    this.render = function() {…};
    this.initClicks = function() {…};
    this.change = function(funct) {…};
}
```

Slika 7.7: Del definiranih funkcij za simulacijo spustnih menijev iz datoteke *select-input.js*.

7.6 Implementacija načrtov zgradbe FRI in objekta X

Pred implementacijo načrtov zgradbe FRI in objekta X smo v programu Adobe Illustrator iz arhitekturnih načrtov odstranili preveč podrobne informacije, nato pa smo na preoblikovane tlorise narisali imena oz. številke kabinetov, predavalnic itd. in jih izvozili kot slike različnih dimenzij formata .png s prosojnim ozadjem (slika 7.8). Oblikovali smo tudi mini 3D-model zgradbe FRI in objekta X, ki uporabniku pri izrisovanju poti ves čas prikazuje nadstropje v stavbi, v katerem se izrisuje pot.

Slika 7.8: Tloris 1. nadstopja zgradbe FRI z narisanimi številkami prostorov.

Z JavaScriptom smo element *canvas* najprej dinamično postavili v dokument HTML (slika 7.9), in sicer v takšni velikosti, kakršna je velikost tlorisov pri različnih ločljivostih.[2] Pri tem funkcija *FriMapPath.getMapScreenSize* (slika 7.10) preračuna, kako velik *canvas* in slika načrta se glede na ločljivost zaslona naložita na spletno stran. V Objektu *FriMapPath.maps* (slika 7.10) so shranjene slike vseh načrtov za različne ločljivosti in mini 3D-model. Na *canvasu* smo nato z JavaScriptom dinamično narisali interaktivne poti. Programska koda za interakcijo z zemljevidi vključno z zgoraj omenjeno funkcijo in objektom je v datoteki *master-run.js*.

```
$('#map_model').html('<img id="floor_img" src="' + floorImg + '" />');
$('#maps').html('<canvas id="canvas" width="' + new String(mapWidth) +
'" height="' + new String(mapHeight) + '"></canvas>');
$('#canvas').drawImage({
    source: mapSrc,
    fromCenter: false, x: -0, y: -0
});

$('#maps').css('margin-top', marginTop);

var data = getData();
var html = mapInfo(data);
$('#query_info').html('');
$('#query_info').html(html);
```

Slika 7.9: Dinamična postavitev *canvasa* na spletno stran.

[2]Aplikacijo *InfoFRI točka* smo testirali na tablici z ločljivostjo 1280 x 800 px, večdotični površini 1280 x 700 px in računalniških monitorjih ločljivosti najmanj 1920 x 1080 px.

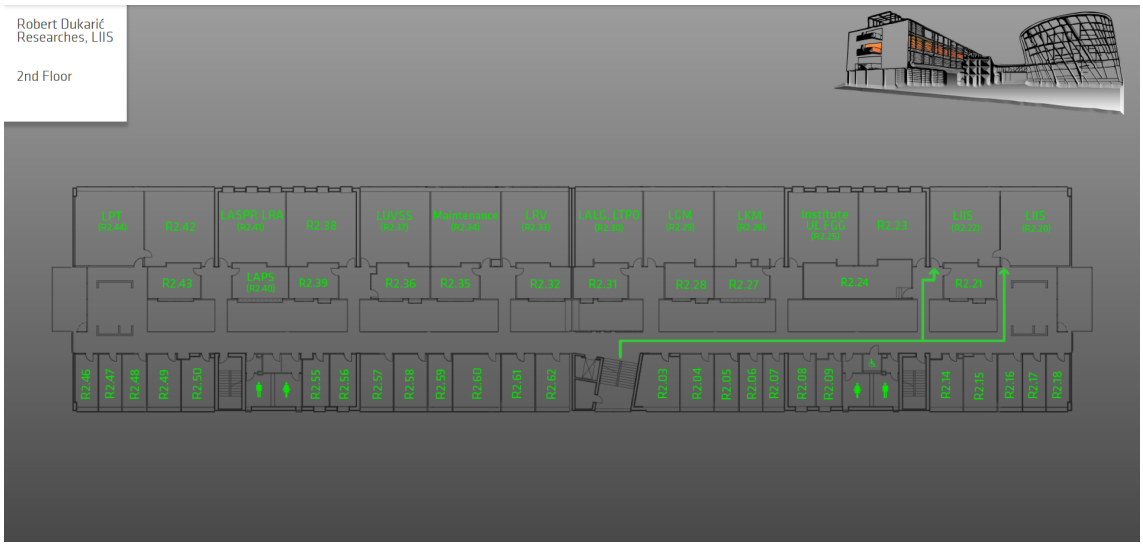
```
FriMapPath.maps = {
    map1: {
        small: { src: 'img/fri_1_res1280.png', width: 1281, height: 467, marginTop: -40 },
        medium: { src: 'img/fri_1_res1600.png', width: 1600, height: 583, marginTop: -10 },
        large: { src: 'img/fri_1_res1900.png', width: 1902, height: 693, marginTop: -10 },
        floorImg: 'img/fri_1.png'
    },
    map2: {
        small: { src: 'img/fri_2_res1280.png', width: 1281, height: 291, marginTop: 220 },
        medium: { src: 'img/fri_2_res1600.png', width: 1600, height: 363, marginTop: 160 },
        large: { src: 'img/fri_2_res1900.png', width: 1902, height: 431, marginTop: 120 },
        floorImg: 'img/fri_2.png'
    },
    map3: {...},
    map4: {...},
    map5: {...},
    map6: {...}
};

FriMapPath.getMapScreenSize = function () {
    if ($(window.document).width() < 1600) {
        return 'small';
    }
    if ($(window.document).width() >= 1600 && $(window.document).width() < 1900) {
        return 'medium';
    }
    if ($(window.document).width() >= 1900) {
        return 'large';
    }
};
```

Slika 7.10: Objekt *FriMapPath.maps* in funkcija *FriMapPath.getMapScreenSize*.

7.7 Risanje interaktivne poti na tlorisih

Ko pri zaposlenih ali prostorih izberemo poljuben vnos (vrstico) v tabeli, nas aplikacija preusmeri na zemljevid in začne izrisovati interaktivno pot, ki vodi do iskanega prostora. Ne glede na to, kateri prostor iščemo, je začetek izrisovanja interaktivne poti zmeraj pri ovalnem rdečem objektu v avli fakultete FRI na tlorisu 1. nadstropja stavbe FRI.

Pri vseh prostorih v 1. nadstropju se pot izrisuje samo na enem tlorisu (slika 7.11). Kadar iščemo prostor v višjem nadstropju zgradbe FRI, se najprej izriše pot do stopnic v prvem nadstropju in nadaljuje od stopnic naprej na tlorisu višjega nadstropja. Če pa je prostor v objektu X, se naložijo tlorisi objekta X, v katerem je iskani prostor.



Slika 7.11: Primer izrisa (razcepljene) interaktivne poti.

Interaktivno pot smo implementirali tako, da smo na eni velikosti vseh načrtov določili koordinate vogalnih točk (tj. točk, kjer pot zavije za 90 stopinj). Točke, ki označujejo vogale pri vsaki poti, so definirane v objektu *FriMapPath.paths*. Ker je pripravljenih več slik načrtov za različne ločljivosti, je bilo treba koordinate točk ustrezno povečati/zmanjšati (skalirati), da bi se na tlorisih (slikah) različnih ločljivosti prikazovale v enakem razmerju. To je implementirano v funkciji *FriMapPath.scaleMap*, ki je prikazana na sliki 7.12.

```
FriMapPath.scaleMap = function (point) {
    var mapName = FriMapPath.mapName;
    var pointSize = 'small';
    var pointWidth = FriMapPath.maps[mapName][pointSize].width;
    var pointHeight = FriMapPath.maps[mapName][pointSize].height;
    var actualSize = FriMapPath.getMapScreenSize();
    var actualWidth = FriMapPath.maps[mapName][actualSize].width;
    var actualHeight = FriMapPath.maps[mapName][actualSize].height;

    point.x = (point.x * actualWidth) / pointWidth;
    point.y = (point.y * actualHeight) / pointHeight;

    return point;
};
```

Slika 7.12: Funkcija *FriMapPath.scaleMap* preračuna, za koliko je treba točke transformirati, da bo pri različni ločljivosti tlorisov razmerje enako.

Funkcije, ki omogočajo interaktivno izrisovanje poti:

- *calculateAllPoints*: funkcija izračuna in vrne vse točke med dvema vogalnima točkama in jih shrani v seznam.
- *drawline*: definira črto in ji določi lastnosti (debelino, barvo), zadnji točki pa nariše puščico.
- *timeOutFunc*: med vogalnimi točkami riše točke[3]. Ko smo pri predzadnji točki, pri zadnji točki nastavi pogoj za izris puščice. Tukaj se preverja, v katerem nadstropju je cilj. Če iščemo prostore v višjem nadstropju, kot je začetna točka poti, se požene del kode, ki naloži naslednji tloris in risanje poti se nadaljuje po njem.
- *candrawPath*: ob kliku na gumb nazaj se risanje poti ustavi, sicer bi se ob drugem izboru vnosa v tabeli še zmeraj izrisovala prva pot.
- Funkcija *setFromButton* si zapomni, za kateri vnos iz tabele se je izrisovala pot, da se lahko pri kliku nazaj vrnemo na pravo tabelo (zaposleni, prostori ali skupno).

[3]Med vogalnimi točkami riše črtice dolžine 1 px, kar so pravzaprav točke.

Razcepljene poti

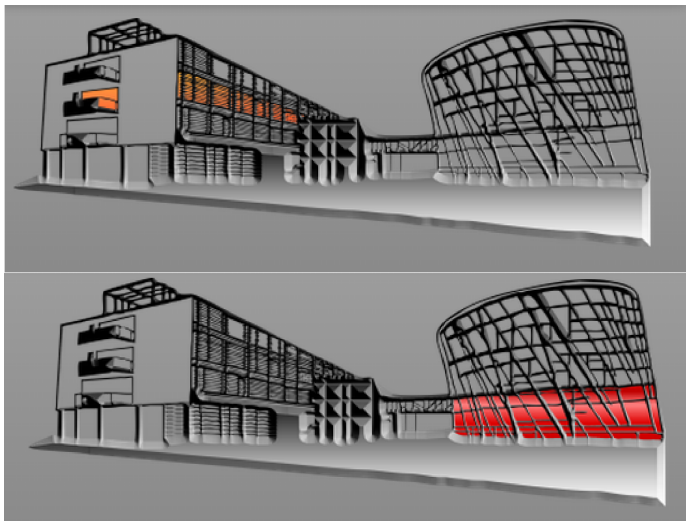
Do vsakega prostora gre od začetka do konca navadna celovita interaktivna pot, kar je implementirano v funkciji *showOnePath*. Kadar je mogoče več prostorov združiti zaradi podobnih interesnih dejavnosti, se pot pri izrisu razcepi (slika 7.11). Za takšno implementacijo smo se odločili, ker ima na primer nekaj laboratorijev dva prostora oz. je v vsakem nadstropju več stranišč. Algoritem razcepljenih poti je implementiran v funkciji *showMultiPaths*.

Podatki o iskanem nizu

Pri izrisovanju poti so levo zgoraj prikazani še podatki o iskanem nizu (slika 7.11). Podatke o osebah in prostorih pridobita funkciji *getRoomData* in *getPersonnelData*, izpišeta pa jih funkciji *mapInfoRoom* in *mapInfoPersonnel*.

Mini 3D-model

Za boljšo orientacijo po zgradbi FRI in objektu X je pri izrisovanju poti desno zgoraj prikazan mini 3D-model celotne stavbe. Vsakokorat, ko se zamenja tloris nadstropja, se zamenja tudi mini 3D-model (slika 7.13).



Slika 7.13: Primer dveh mini 3D-modelov, ki prikazujeta nahajanje v 2. nadstropju stavbe FRI in pritličju objekta X.

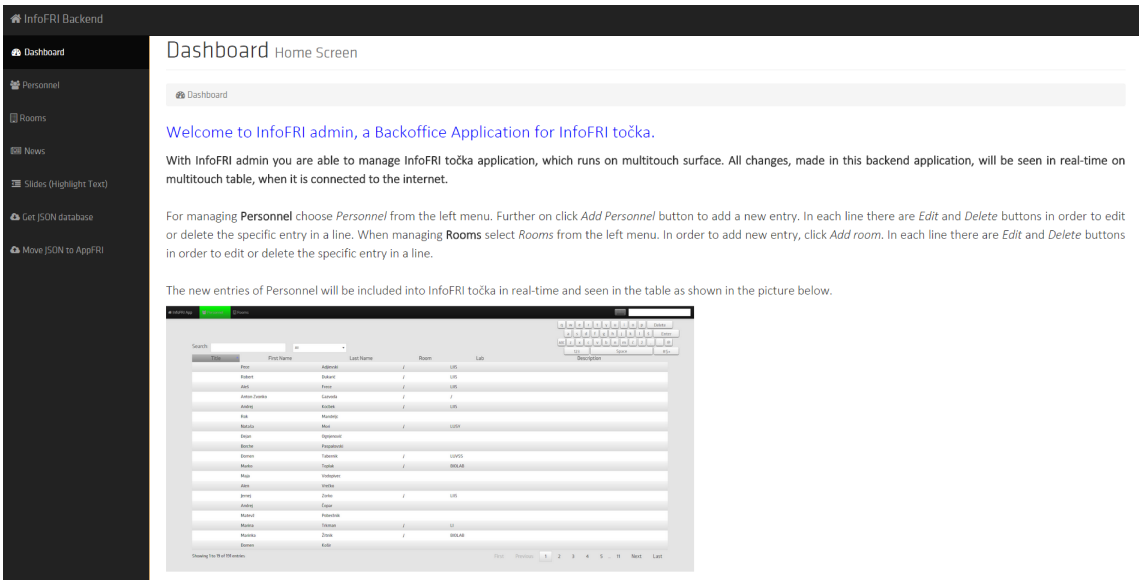
Poglavje 8

InfoFRI admin

8.1 Opis zaledne spletne aplikacije

V osmem poglavju je opisana zaledna (ang. backend) spletna aplikacija, ki omogoča urejanje *InfoFRI točke* prek modernega uporabniškega vmesnika. Za osnovo smo vzeli predlogo, ki uporablja knjižnico Bootstrap in ki smo jo preuredili in ji dodali funkcionalnosti. Na levi strani spletne aplikacije (slika 8.1) je navigacijski meni s povezavami do posameznih podstrani s podatki v tabelah (zaposleni, prostori, novice in prosojnice), na vseh podstraneh je tudi označeno, kje lahko vstavimo, brišemo ali urejamo vsebine. Za prikaz ikon v navigacijskem meniju smo uporabili knjižnico Font Awesome [9].

Ker je *InfoFRI admin* namenjen interni uporabi na fakulteti, je nabor njegovih funkcionalnosti osredotočen na vstavljanje, brisanje in spreminjanje vnosov pri zaposlenih, prostorih, novicah in prosojnicah. Pri avtentikaciji administratorjev smo se zaradi tega odločili za enostavno avtentikacijo z datoteko *.htaccess*, ki jo omogoča spletni strežnik Apache. Avtentikacija je opisana v podpoglavju 8.2. V nadaljevanju sledijo še podpoglavja z opisom in primeri implementacije pri vstavljanju vnosov, opis skripte PHP za pretvarjanje slik v nize po standardu Base64 (podpoglavje 8.4) in še opis dveh skript PHP, ki omogočata izvoz dokumenta JSON iz CouchDB (podpoglavje 8.5).



Slika 8.1: Uvodna stran pri *InfoFRI adminu*.

8.2 Avtentikacija administratorjev s .htaccess

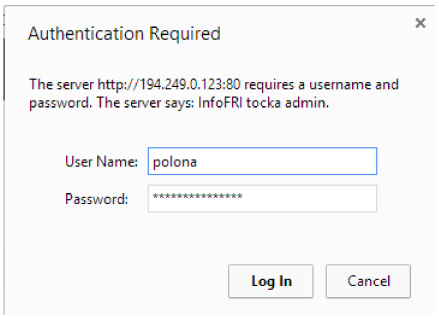
Kot večina administratorskih orodij za urejanje spletnih aplikacij tudi *InfoFRI admin* omogoča avtentikacijo administratorjev pred samim spreminjanjem vsebin v podatkovni bazi *InfoFRI točke*.

Ker je na fakultetnem strežniku nameščen Apache, smo se odločili za nastavitve dostopov do spletnih imenikov in dokumentov s konfiguracijsko datoteko *.htaccess* (hypertext access), ki med drugim omogoča enega izmed bolj enostavnih načinov strežniškega preverjanja dostopov oz. zaščito določene mape na strežniku z uporabniškim imenom in geslom [5]. Na strežniku smo v imeniku aplikacije *InfoFRI admin* ustvarili datoteko *.htaccess* (slika 8.2) in vanjo zapisali administratorje, ki lahko dostopajo in urejajo vsebine v *InfoFRI točki*. V *.htaccess* je določena tudi pot do uporabniških gesel, ki smo jih kriptirana[1] shranili v datoteki *passwords*. Administratorji so določeni v zadnji vrstici datoteke *.htaccess* (slika 8.2), gesla pa priredimo administra-

[1] *.htaccess* zahteva shranjevanje gesel z zgoščevalno funkcijo MD-5.

torjem v ukazni vrstici[2] ali jih prek spletne storitve kriptiramo in priredimo posameznim uporabnikom v datoteki *passwords*, upoštevavši naslednjo skladnjo: *uporabnisko_ime:kriptirano_geslo*. Pri dostopanju do spletne aplikacije *InfoFRI admin* se pojavi prijavno okno z zahtevo po uporabniškem imenu in geslu, kot je prikazano na desni strani slike 8.2.

```
AuthType Basic
AuthName "InfoFRI tocka admin"
# (Following line optional)
AuthBasicProvider file
AuthUserFile /var/www/passwords
Require user polona
```



Slika 8.2: Na levi strani slike je prikazana vsebina datoteke *.htaccess*. Z AuthUserFile določimo pot do datoteke z gesli, v zadnji vrstici pa administratorje. Na desni strani je prijavno okno, ki se pojavi pri dostopanju do *InfoFRI admina*.

[2]V ukazni vrstici okolja Unix z ukazom *htpasswd datoteka_z_gesli ime_uporabnika* (*htpasswd gesla polona*) uporabniku priredimo kriptirano geslo.

8.3 Prikazovanje vsebine iz CouchDB

InfoFRI admin omogoča operacije vstavljanja, brisanja in urejanja. Način povezave in komunikacije s CouchDB ter prikazovanja podatkov je podoben pri vseh operacijah,[3] zato bo poleg razlage v nadaljevanju prikazan le del programske kode (slika 8.3) za eno od operacij. Vsa programska koda za vstavljanje, brisanje, urejanje je v datoteki *couchdb.js*.

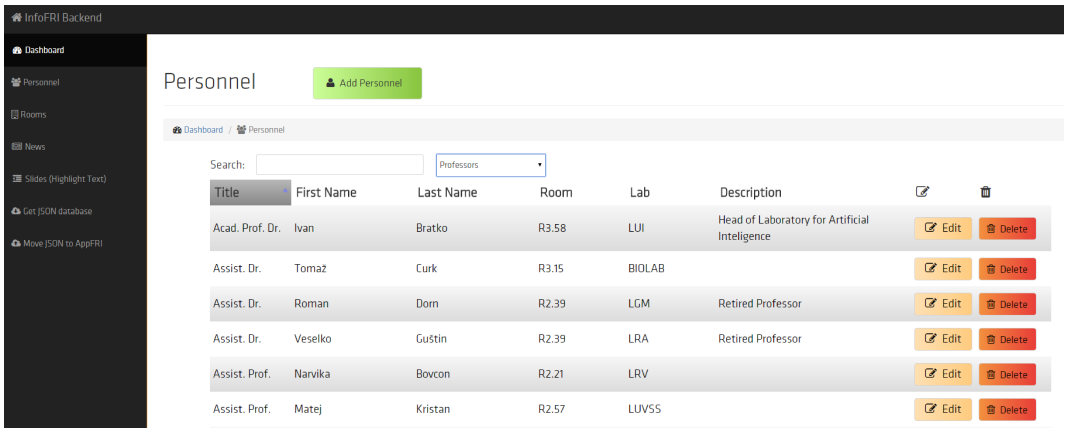
```
function insert_news(title, text, summary, smallImg, bigImg, date) {
    var data = JSON.parse(FRI_DATA);
    var id = getMaxId(data.news) + 1;
    var object = {
        id: id,
        title : title,
        text : text,
        summary : summary,
        smallImg : smallImg,
        bigImg : bigImg,
        date : date,
        field: 'news',
        _id: ("news_" + id)
    };
    $("body").addClass("loading");
    rpdb.put(object).then(function (response) {
        selectAllData(function() {
            FriInfo.DataTable.populateNews()
            $("#data_news").show();
            $("#news-input-form").hide();
            FriInfo.Form.News.clear();

            $("body").removeClass("loading");
        });
    }).catch(function (err) {
        alert(err);
    });
}
```

Slika 8.3: Primer vstavljanja novic v CouchDB.

V primeru **vstavljanja** zaposlenih izberemo gumb *Add Personnel* (slika 8.4), nahajajoč se na podstrani *Personnel*, ki je dostopna iz navigacijskega menija. Ob kliku se prikažejo prazna vnosna polja z možnostjo vnašanja dotičnih podatkov glede na vrsto vnosa (zaposleni, prostori ...). Shranjevanje vnosa potrdimo s klikom na gumb *Submit*. Vnos lahko **zbrišemo** s

[3]Prav tako velja tudi pri vseh vnosih: pri zaposlenih, prostorih, novicah in prosojnicah.

klikom na gumb *Delete*, ki je v vsaki vrstici vnosa na desni strani v tabeli (slika 8.4). Prav tako je pri vsakem vnosu na desni strani v tabeli še gumb *Edit*, ki posamezne podatke enega vnosa prikaže v vnosnih poljih, kjer jih lahko **urejamo** (slika 8.4).



Slika 8.4: Pri podstrani o zaposlenih so iz CouchDB v tabeli izpisani podatki. S klikom na gumb *Add Personnel* dodamo nov vnos v tabelo, na koncu vsake vrstice pa imamo gumba *Edit* in *Delete*, ki sta namenjena urejanju oz. brisanju vnosov.

Pridobivanje in shranjevanje podatkov v CouchDB predstavlja jedro in najtežji del implementacije v tem diplomskem delu. Ko v brskalniku odpremo spletno aplikacijo, se podatki sproti kličejo (prenesejo) iz CouchDB v JavaScript objekt *FRI_DATA*[4] in se naložijo v posamezne tabele. Pri shranjevanju sprememb ob posodobitvi, brisanju ali vstavljanju nove vsebine s klikom na gumb *Submit* poskrbimo, da se spremembe shranijo v CouchDB, nato pa se podatki ponovno kličejo in prenesejo v JavaScript objekt in šele zatem se posodobljeni izrišejo v tabeli. Pri zaledni aplikaciji se podatki v tabeli vsakokrat ob kliku na gumb *Submit* ponovno naložijo iz CouchDB v JavaScript objekt, iz tega pa v tabele. Zaradi tega se zdi, da se spletna stran samodejno osvežuje ob potrditvi vnesenih sprememb. Primer vstavljanja v CouchDB je na sliki 8.3.

Pri *InfoFRI točki* se podatki prav tako naložijo iz CouchDB, shranijo v JavaScript objekt in iz slednjega prikažejo v razdelkih in tabelah, vendar tukaj

[4]JavaScript objekt FRI_DATA vsebuje vse podatke v obliki formata JSON.

osveževanje ne poteka samodejno, temveč je za prikaz najnovejših sprememb treba ponovno naložiti oz. osvežiti aplikacijo (oz. ustvariti nov dokument JSON in ga prenesti v imenik *InfoFRI točke*).

8.4 Pretvorba slik v nize po standardu Base64

V CouchDB so vsi podatki (tudi slike) shranjeni kot nizi. Pri nalaganju na strežnik ter shranjevanju prosojnic in slik pri novicah v CouchDB je bilo torej treba vse slike iz binarnega zapisa pretvoriti v nize in jih šele nato shraniti v CouchDB. V ta namen smo napisali skripto PHP *imgToBase64.php* (slika 8.5), ki je v imeniku *InfoFRI admina* na fakultetnem strežniku in se zažene vsakokrat, ko želimo naložiti in shraniti sliko v CouchDB.

Base64 je format zapisa kodiranja za pretvarjanje binarnih podatkov v tekstovne nize. Kodirani podatki po standardu Base64 so za tretjino bolj obsežni od originalnih. Prednost pretvorbe je kompatibilnost z večino operacijskih sistemov, saj imajo v kodirni formi omejen nabor znakov. Algoritem deluje po naslednjem postopku:

- Base64 kodira po tri bajte vhodnih podatkov hkrati.
- Vse tri bajte razvrsti v 24-bitne vrednosti.
- 24-bitne vrednosti organizira v štiri skupine po šest bitov.
- 6-bitne skupine pretvori v desetiška števila.
- Dobljene desetiške vrednosti poiščemo v tabeli Base64 in jih pretvorimo v znake [6].

Pri nalaganju *InfoFRI točke* v brskalniku se shranjeni podatki kličejo iz CouchDB, pretvorjene slike v Base64 pa zna brskalnik samodejno pretvoriti nazaj v prvotno obliko (tj. prikazati sliko).

```php
<?php
header("Access-Control-Allow-Origin: *");

if(isset($_FILES[0])){
    $path = $_FILES[0]['tmp_name'];
    $type = $_FILES[0]['type'];
    $data = file_get_contents($path);
    $base64 = 'data:' . $type . ';base64,' . base64_encode($data);
    echo $base64;
}
```

Slika 8.5: Skripta PHP za pretvarjanje slik v nize po standardu Base64, ki se izvrši pri nalaganju slik v CouchDB.

8.5 Izvoz dokumenta JSON iz CouchDB

Namen *InfoFRI točke* je predvsem delovanje na večdotični površini, zato smo se pri implementaciji obeh aplikacij že v osnovi osredotočili in prilagajali zmogljivostim večdotične površine. Ker je ta zaradi svoje enostavnosti pri postavitvi in uporabi precej mobilna in jo je mogoče postaviti praktično kamor koli, morajo vse aplikacije delovati tudi brez povezave v svetovni splet.

Prav zaradi tega smo pri obeh aplikacijah implementirali dostop do podatkov neodvisno od internetne povezave. Kadar sta obe implementirani aplikaciji povezani v svetovni splet, se vsi podatki v *InfoFRI točki* kličejo in prenesejo neposredno iz CouchDB v JavaScript objekt *FRI_DATA*. Če pa *InfoFRI točka* ni povezana v internet in posledično nima neposrednega dostopa do CouchDB, se podatki berejo iz lokalnega dokumenta JSON znotraj imenika *InfoFRI točke*. Zaradi tega omogoča *InfoFRI admin* tudi ustvarjanje in izvoz podatkov v dokument JSON. Skripta *create.php* (slika 8.6) iz podatkov v CouchDB ustvari in izvozi dokument JSON ter ga odloži na strežnik v imenik *InfoFRI admina*. Skripto sprožimo s klikom na povezavo *Get JSON database* v *InfoFRI adminu* (slika 8.4).

Za posodobitev podatkov pri *InfoFRI točki* moramo izvožen dokument JSON še prestaviti iz imenika, kjer smo ga odložili (pri *InfoFRI adminu*), v imenik *InfoFRI točke*, ki mora imeti v času prenosa dostop do interneta. S klikom na

povezavo *Move JSON to AppFRI* (slika 8.4) pokličemo skripto *move.php*, ki v imeniku *InfoFRI točke* ustvari nov dokument JSON (če ga ni) oz. zamenja obstoječega (slika 8.6).

```
** create.php **

<?php
$db = "http://194.249.0.123:5984/appfri";
$fileName = "/var/www/html/appfri_backend/data/db.json";

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $db."/_all_docs?include_docs=true");
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

$output = curl_exec($ch);
curl_close($ch);

file_put_contents($fileName, $output);

header("Location: index.html");
exit;
~


**  move.php **

<?php
$fileNameSrc = "/var/www/html/appfri_backend/data/db.json";
$fileNameDest = "/var/www/html/appfri/data/db.json";

rename($fileNameSrc, $fileNameDest);
```

Slika 8.6: Skripti *create.php* in *move.php* za ustvarjanje in izvoz dokumenta JSON ter premik v imenik *InfoFRI točke*.

Poglavje 9

Sklepne ugotovitve

V okviru diplomskega dela smo upoštevaje uporabniške zahteve realizirali vse zastavljene cilje in razvili spletno aplikacijo *InfoFRI točka* za večdotično površino, razvito v Laboratoriju za umetne vizualne in spoznavne sisteme (ViCoS) na Fakulteti za računalništvo in informatiko (FRI). V osnovi je *InfoFRI točka* namenjena izrisovanju interaktivnih poti na tlorisih zgradbe FRI in objekta X od začetne točke v avli fakultete do vseh prostorov na fakulteti. *InfoFRI točka* je svojevrsten virtualni kažipot, ki omogoča tudi ločen in skupen izpis v tabeli po seznamih zaposlenih in prostorov na FRI ter postreže z informacijami javnega značaja v obliki krajših novic na prvi strani, zaradi česar jo lahko poimenujemo informacijska točka – *InfoFRI točka*. Deluje na večdotični površini, ki bo postavljena v avli fakultete in tako študentom, gostujočim učiteljem ali obiskovalcem omogočala lažjo orientacijo po fakulteti. Med razvojem *InfoFRI točke* se je pokazala potreba po implementaciji še zaledne spletne aplikacije *InfoFRI admin*, ki kot spletna aplikacija na fakultetnem strežniku v okviru interne rabe omogoča urejanje *InfoFRI točke*. Pri razvoju obeh spletnih aplikacij smo poleg spletnih tehnologij HTML5, CSS3 in JavaScript ter jQuery za implementacijo podatkovne baze izbrali in podrobneje predstavili nerelacijsko podatkovno bazo CouchDB, ki shranjuje podatke v formatu JSON, za pisanje poizvedb uporablja JavaScript ter za dostop in urejanje podatkovne baze spletni uporabniški vmesnik Fauxton. Interaktivne poti v *InfoFRI točki* se izrisujejo na elementu *canvas*, na

katerega smo postavili slike tlorisov zgradbe FRI in objekta X. Iz arhitekturnih načrtov (tlorisov) smo odstranili odvečne informacije in nanje narisali imena oz. številke prostorov. Na podlagi slik tlorisov smo pridobili koordinate vogalnih krajišč, kjer pot spremeni smer. Koordinate vogalnih točk smo vnesli v algoritem, ki preračunava skalabilnost vogalnih točk (krajišč) glede na velikosti slik in izrisuje točke med krajišči na platno (*canvas*). Pri morebitnem nadaljnjem razvoju in nadgradnji *InfoFRI točke* bi bilo smiselno implementirati dodatne funkcionalnosti, kot so dinamično spreminjanje napisov pri tlorisih načrtov z JavaScriptom, dinamično preračunavanje vogalnih točk ter vnos vseh podatkov v podatkovno bazo, kjer bi jih bilo mogoče urejati. Za boljšo uporabniško izkušnjo pa bi bilo lahko izrisovanje poti podprto še z glasovnim vodenjem (podobno kot pri navigacijskih aplikacijah), nenazadnje pa bi lahko *InfoFRI točka* prikazovala tudi virtualnega inteligentnega pomočnika, ki bi znal odgovarjati na vprašanja v naravnem jeziku.

Literatura

[1] J. C. Anderson, J. Lehnardt in N. Slater, *CouchDB. The Definite Guide*, Kalifornija: O'Reilly Media, 2010.

[2] K. Istenič, *Razvoj večdotične površine na osnovi barvno-globinske kamere*, Ljubljana: Fakulteta za računalništvo in informatiko, diplomsko delo, 2013, str. 21–37.

[3] K. Istenič, L. Čehovin in D. Skočaj, *Multi-touch surface based on RGBD camera*, Ljubljana: Fakulteta za računalništvo in informatiko, 2014.

[4] J. Lennon, *Beginning CouchDB*, New York: Springer-Verlag, 2009, str. 3–6.

[5] .htaccess-Guide [Online]. Dostopno na: http://htaccess-guide.com/.

[6] Base64 [Online]. Dostopno na: https://blogs.oracle.com/rammenon/entry/base64_explained.

[7] Erlang [Online]. Dostopno na: http://www.erlang.org/.

[8] Fauxton Visual Guide [Online]. Dostopno na: http://michellephung.github.io/Fauxton-Visual-Guide/.

[9] Font Awesome [Online]. Dostopno na: https://fortawesome.github.io/Font-Awesome/.

[10] JSON [Online]. Dostopno na:
http://developers.squarespace.com/what-is-json/.

[11] Media Queries [Online]. Dostopno na:
http://www.w3.org/TR/html4/types.html#h-6.13.

[12] Microsoft Developer Network [Online]. Dostopno na:
https://msdn.microsoft.com/en-us/library/jj131033.aspx.

[13] NoSQL Database Solutions [Online]. Dostopno na:
http://blog.flux7.com/blogs/nosql/nosql-database-solutions-types-and-examples.

[14] NoSQL Databases Overview [Online]. Dostopno na:
http://www.thoughtworks.com/insights/blog/nosql-databases-overview.

[15] NoSQL Seminar [Online]. Dostopno na:
http://www.slideshare.net/Shreyash007/nosql-seminar.

[16] Outsourcing partners [Online]. Dostopno na:
http://blog.outsourcing-partners.com/2012/10/why-nosql-database-is-used-by-facebook-google-and-linkedin-applications/.

[17] Seattlepi.com [Online]. Dostopno na:
http://blog.seattlepi.com/digitaljoystick/2009/06/01/e3-2009-microsoft-at-e3-several-metric-tons-of-press-releaseapalloza/.

[18] TUIO [Online]. Dosegljivo:
http://www.tuio.org/?specification.

[19] W3Schools [Online]. Dostopno na:
http://www.w3schools.com/html/default.asp.