TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Eduard Bortkevitš 134459IAPB

REISIMARSRUUDI PLANEERIMISE VEEBIRAKENDUS

Bakalaureusetöö

Juhendaja: Gert Kanter

Tehnikateaduse magister

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Eduard Bortkevitš

21.05.2019

Annotatsioon

Antud bakalaureusetöö eesmärk on luua reisimarsruudi planeerimise veebirakendus. Rakendus pakub kasutajale võimalust luua ja hallata reisiplaane, näha ülevaadet varem loodud reisiplaanidest ning automatiseerida reisiplaani sihtpunktide vaheliste marsruutide arvutust.

Veebirakenduse arendamisel on kasutatud Angular 7 raamistikku, Java 8 programmeerimiskeelt koos Spring Boot raamistikuga ja PostgreSQL 11 andmebaasi.

Töö esimeses osas analüüsib autor olemasolevaid reisiplaneerimisega seotud veebi- ja mobiilirakendusi. Teises osas toob autor välja eesmärkide saavutamiseks valitud tehnoloogiad ja tehtud valikute põhjused. Kolmandas osas kirjeldab autor rakenduse teostamist ja selle funktsionaalsust. Neljandas osas analüüsib autor rakenduse kasutajate poolt teostatud valideerimise tulemusi ja toob välja peamised täiendused, mida tulevikus rakenduse funktsionaalsuses teha.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 6 peatükki, 11 joonist.

Abstract

Travel Itinerary Planning Web Application

The goal of this bachelor's thesis is to create a travel itinerary planning web application. Application allows user to create and manage travel itineraries, see an overview of previously created itineraries and automate calculation of routes between itinerary waypoints.

The web application was developed using Angular 7 framework, Java 8 programming language with Spring Boot framework and PostgreSQL 11 database.

In the first part of the thesis, author analyses existing travel itinerary web and mobile applications. In the second part, author outlines the technologies chosen to achieve goals and the reasons for the choices made. In the third part, author describes development of the application and its functionality. In the fourth part, author analyses the results of validation of the application by users and outlines the main improvements to application functionality to be done in the future.

The thesis is in Estonian and contains 25 pages of text, 6 chapters, 11 figures.

Lühendite ja mõistete sõnastik

Google Maps	Google poolt arendatav veebipõhine kaarditarkvara
Angular	*TypeScript* baasil töötav esitluskihi raamistik
TypeScript	*JavaScript* keele ülemhulk, kus on lubatud staatilised tüübid
JavaScript	Esitluskihi objektorienteeritud programmeerimiskeel
Java	Platvormist sõltumatu objektorienteeritud programmeerimiskeel
Spring	*Java* platvormi raamistik
Spring Boot	*Spring* raamistiku osa, mis on eelseadistatud parimate tavade järgi
PostgreSQL	Objekt-relatsiooniline andmebaasi haldamise süsteem
SPA	*Single-page application.* Dünaamiline veebirakendus, millel puudub vajadus lehe uuesti laadimiseks
HTML	*Hypertext Markup Language*. Märgistuskeel veebilehtede ja veebirakenduste loomiseks
React	*JavaScript* teek esitluskihi arendamiseks
API	*Application Programming Interface*. Rakendusliides
PHP	*Hypertext Preprocessor*. Skriptimiskeel
Gradle	Rakenduse ehitamise automatiseerimise süsteem
Docker	Virtualiseerimise tarkvara
Liquibase	Andmebaasi haldamise teek
DAO	*Data Access Object*. Objekt, mille abil saab ligi andmetele
CSS	*Cascading Style Sheets*. Märgistuskeeles kirjutatud dokumendi kuvamise stiili kirjeldav keel

Sisukord

Jooniste loetelu

1 Sissejuhatus

1.1 Motivatsioon ja probleem

Reisimine on paljude inimeste jaoks lahutamatu osa elust. Põhjusi reisimiseks on mitmeid: kultuur, sport, puhkus, töö. Kui varem tegelesid reiside planeerimisega valdavalt reisibürood, siis tänapäeval on üha populaarsemaks muutumas iseseisev reisiplaanide koostamine ja broneerimine[1] . Nii on inimesel võimalik endal valida, kus ja kuidas ta aega veedab, sõltumata reisikorraldaja piirangutest. Tihtipeale on selline reisimine ka odavam. Samuti ei pruugi kohalikud reisibürood pakkuda reise eksootilisematesse sihtkohtadesse.

Reisiplaanide koostamiseks on loodud palju rakendusi nutiseadmetele. Vähesemal määral on olemas ka veebirakendusi, mis taolist funktsionaalsust pakuvad. Siiski ei võimalda isegi olemasolevad lahendused reisiplaane piisavalt personaliseerida. Näiteks võib inimesel olla reisiks eraldatud kindel eelarve, ajaline graafik või jõudlus – kui suure vahemaa ta soovib mingi aja vältel läbida. Kui jätta kõik need parameetrid reisi planeerimisel välja, ei pruugi tulemus olla meelepärane. Inimesele jääb plaane tehes võimalus oma võimalustega arvestada, kuid see nõuab palju aega ja tööd.

1.2 Töö eesmärk

Käesoleva bakalaureusetöö eesmärgiks on luua veebirakendus, mis suudab reisi ja selle marsruudi planeerimisel arvestada inimese personaalsete vajadustega ning pakkuda kasutajale võimalust neid vajadusi sätestada.

Rakenduse kaudu on võimalik sisestada soovitavat eelarvet, mida arvesse võttes kuvatakse kasutajale, kas tema plaanitud reis jääb selle piiridesse. Vajadusel võib kasutaja teha muudatusi oma eelarves või kulutustes.

Samuti saab kasutaja lisada kellaaegu, millal ta reisida ei soovi. Näiteks teab kasutaja, et kolmel esimesel puhkuse päeval on ta 14:00 – 16:00 hõivatud. Rakendus võimaldab seda määrata ning neid kellaaegu marsruudi koostamisel arvesse ei võeta.

Plaanide koostamisel on tähtis osa ka inimese jõudlusel. Rakendus pakub võimalust määrata maksimaalset läbitavat vahemaad ühe päeva vältel. Kahe sihtpunkti vahelise vahemaa arvutamisel on võimalik valemisse lisada ka tõusumeetrid – kõrguse kasvamisel ei pruugi inimese jõudlus samaks jääda.

Valmiv rakendus on põhiliselt liidestatud *Google Maps* süsteemiga. Reisi sihtkohtade kohta jääb kasutajale võimalus lisada isiklikke kommentaare ja täpsustusi.

1.3 Töö teostamise metoodika

Töö peamiseks tulemiks on toimiv veebirakendus. Valminud rakendusel on olemas andmebaas, teenuskiht ja esitluskiht.

Rakenduse esitluskiht on loodud kasutades *Angular 7* raamistikku. *Angular* on avatud lähtekoodiga front-end arenduse raamistik, mis töötab *TypeScript* keele baasil[2] . Rakendus töötab populaarsemates veebibrauserites: Mozilla Firefox, Google Chrome, Microsoft Edge, Internet Explorer 11.

Rakenduse teenuskiht on realiseeritud keeles *Java*, kasutades *Spring* back-end raamistikku. *Spring* raamistikust on eesmärgi täitmiseks valitud *Spring Boot*, mis lubab ilma suuremahulise konfigureerimiseta luua töötavaid rakendusi[3] .

Andmebaasi tehnoloogiaks on valitud *PostgreSQL*. Tegemist on objekt-relatsiooniline andmebaasi haldamise süsteemiga. Nagu ka eelnevad raamistikud, on *PostgreSQL* avatud lähtekoodiga[4] .

Veebirakendus töötab „*single-page application*” põhimõtte järgi. See tähendab, et veebilehe sisu ei laeta iga kord uuesti. Selle asemel küsitakse andmeid serverilt dünaamiliselt ja kuvatakse neid juba laetud veebilehe sees.

1.4 Töö struktuur

Käesoleva töö teises peatükis on välja toodud erinevad probleemi lahendused, mis juba olemas on. Töö autor võrdleb neid lahendusi ning toob välja nende nõrgad ja tugevad küljed.

Kolmandas peatükis kirjeldab autor valitud tehnoloogiaid ja nende sobivust probleemi lahendamiseks. Samuti on välja toodud võrdlus alternatiivsete tehnoloogiatega.

Neljandas peatükis kirjeldab autor töö raames valminud rakenduse teostamist ja selle põhilisemat funktsionaalsust.

Viiendas peatükis räägib autor töö valideerimisest ja selle tulemustest. Samuti on välja toodud täiendused, mida on võimalik rakendusele luua.

2 Olemasolevad reisiplaneerimise rakendused

Käesolevas peatükis on kirjeldatud mõned eksisteerivad lahendused. Iga lahenduse kohta tuuakse välja selle tugevad ja nõrgad küljed ning analüüsitakse selle sobivust käsitletava eesmärgi saavutamiseks.

Reisiplaneerimise eesmärgil on loodud mitmeid lahendusi, kaasaarvatud veebirakendusi. Käesoleva töö raames vaadeldava probleemi lahendamiseks analüüsis autor neist kolme:

- Itineree – http://itineree.com/
- Roadtrippers – https://roadtrippers.com/
- Travefy – https://travefy.com

Lisaks on loodud reisiplaneerimise rakendusi ka nutiseadmetele. Seetõttu valis autor neist populaarseima samuti analüüsitavate rakenduste hulka:

- Google Trips – https://get.google.com/trips/

2.1 Itineree

Itineree puhul on tegemist reisi ja selle sihtkohtade planeerimise rakendusega. Rakendus on liidestatud suure hulga väliste süsteemidega, mille hulka kuuluvad näiteks Google Maps, TripAdvisor, Yelp, ja Booking.com[5] .

Rakenduse funktsionaalsus lubab kasutajal luua mitmepäevase reisiplaani. Iga päeva juures on võimalik märkida ühe linna, kus reisija parasjagu viibib. Selle linna piires saab otsida ööbimis- ja söögikohti ning muid vaatamisväärsusi. Tänu mitmetele liidestustele on ka otsingutulemuste arv suur. Valitud sihtkohad kajastuvad kaardil, mida saab soovi korral avada. Loodud reis on sotsiaalmeedia teel jagatav ning ka teiste kasutajate loodud reise on võimalik kopeerida ja seejärel muuta.

2.2 Roatrippers

Roadtrippers on rakendus, mis võimaldab reisiplaane detailsemalt koostada. Rakenduse sihtgrupp on inimesed, kes reisivad iseseisvalt ning soovivad külastada kohti, mis ei pruugi olla tavalised turismisihtkohad. Rakendus on saadaval nii veebi- kui ka mobiilirakendusena[6].

Lisaks sihtkohtade otsingule, mis on sellist tüüpi rakenduste puhul tavaline, võimaldab Roadtrippers ka marsruuti personaliseerida. Selleks tuleb alguses valida algus- ja lõpppunkt, mille vahel rakendus joonistab esialgse marsruudi. Reisiplaanis on iga teelõigu kohta välja toodud vahemaa, ajakulu ja eeldatav kütuse maksumus, mis selle tee läbimiseks kulub. Joonistatud marsruuti saab kaardi peal liigutada ja lisada sellele vahepunkte, mille kohta samuti arvutatakse välja eelnimetatud väärtused. Nii oleks soovi korral käsitsi vahepeatusi lisades võimalik arvestada vahemaa ja ajaga, mida kasutaja soovib päevas kulutada. Rakenduse tasuta versioon lubab lisaks reisi algus- ja lõpppunktile lisada ainult 5 vahepunkti. Iga reisisihtkoha ja vaheetapi kohta saab sisseloginud kasutaja lisada isiklikke märkmeid ja kuupäevi. Transpordi liik on seadistatav, kuid valik piirdub mootorsõidukitega. Sõiduki kütusekulu on samuti muudetav.

2.3 Travefy

Travefy veebi- ja mobiilirakendus pakub võimalust luua väga detailseid reisiplaane, mis sobivad nii ühele isikule kui ka grupis reisijatele. Rakendus võimaldab koostada mitmepäevaseid plaane, lisades igale päevale tegevusi, ööbimiskohti, söögikohti, lende ja muud transporti ning täiendavat informatsiooni. Peaaegu kõikide kategooriate all saab ära märkida kellaajad ja kuupäevad, broneeringud, kulud, fotod ja muud seotud failid. Rahalised kulud võib ära jagada inimeste vahel, märgistades kes reisijatest üritusest osa võttis. Koostatud reisiplaani punktid on informatiivsed – näiteks kellaaegade kattuvust ei kontrollita ning ka mitme sihtkoha vahelise kauguse arvutus puudub. Rakenduses on olemas Google Maps toel töötav otsing, mis annab kasutajale võimaluse teda huvitavad sihtkohad kiirelt reisiplaani lisada[7].

2.4 Google Trips

Google Trips on nutiseadmetele mõeldud mobiilirakendus, mille tugevaks küljeks on selle oskus pakkuda kasutajale valmis reisiplaane. Sisestades sihtkoha soovitab rakendus huvi pakkuvaid vaatamisväärsusi, restorane, muuseume jne. Soovi korral võib kasutaja neid soovitusi muuta ja valida endale rohkem meeldivad sihtpunktid. Selleks, et rakenduse kasutamine muudaks reisimise veelgi mugavamaks, kogub Google Trips kasutaja Gmail postkastist kokku informatsiooni reisi kohta, et seejärel näidata seda ühes kohas[8].

Lisaks soovitustele ja broneeringute informatsioonile pakub rakendus kasulikku infot ka linna kohta, kuhu kasutaja plaanib minna. Näiteks võib selline info sisaldada transpordi kasutamise võimalusi ja hädaabinumbreid. Rakenduse keskseks funktsiooniks on Google Maps, kuid see ei kasuta paljusid teenuseid. Näiteks ei paku Google Trips linnadevahelise liikluse puhul vahemaa arvutamist ega rahalist kulu.

2.5 Olemasolevate rakenduste analüüs

Igal eelnevalt kirjeldatud rakendusel on oma tugevad ja nõrgad küljed. Siiski on üks funktsionaalsus, mida kõik analüüsitud rakendused pakuvad – huvi pakkuvate sihtkohtade otsing. Nimelt on kõikides rakendustes võimalik otsida vaatamisväärsusi, muuseume, restorane ja muid asutusi ning neid oma reisiplaanidesse lisada. Samuti lubavad kõik analüüsitud rakendused mingil määral personaliseerida plaane – lisada kuupäevi, mitmeid sihtkohti, anda reisiplaanile nimetus.

Itineree rakenduse tugevaks küljeks on liidestuste rohkus, kuid rakenduse analüüsimise käigus oli mitmel korral näha, et see võib tekitada jõudluse probleeme. Kui rakendus peab koguma andmeid kokku mitmest erinevast kohast, kulub sellele rohkem aega ning tööprotsess muutub kasutajale ebamugavamaks. Mitmel korral rakendus hangus otsingu käivitamise järel. Samuti ei lahenda rakendus peamist vaadeldavat probleemi – kasutaja soovi personaliseerida oma reisiplaane arvestades mitmeid erinevaid tegureid nagu vahemaad, ajalised piirangud ja rahakulu.

Roadtrippers rakendus on probleemi lahendamisele palju lähemal. Nimelt on see mõeldud just sellistele kasutajatele, kes soovivad ise otsustada, mis teed pidi ja kuidas

nad soovivad sihtkohta jõuda. On olemas võimalus käsitsi lisada piiratud arv vahepunkte, et jagada reis mitmeks osaks, kuid see on ebamugav ning vajab palju peenhäälestamist. Veel üheks puuduseks on liikumisvahendite piiratus – rakendus arvestab liikumisel ainult autode, busside ja mootorratastega. Seetõttu ei ole kütusekulu arvutamine muu transpordiliigi puhul asjakohane.

Travefy lähenemine probleemile on samuti hea. Veebirakendus lubab reisiplaani koostajal personaliseerida väga palju aspekte. Kirja saab panna nii kellaaegu, sihtkohti, hindu kui ka lisainfot. Siiski jääb lisatud info ainult informatiivseks ja mingeid arvutusi sellega ei tehta, mis tähendab lisatööd kasutajale. Lisatud info põhjal oleks võimalik arvutada ka marsruute, kuid seda funktsionaalsust ei pakuta.

Google Trips on analüüsitud rakendustest ainus, millel puudub veebirakendus. Pakutav funktsionaalsus on kasutajale mugav – suur osa reisiga seotud informatsioonist laetakse automaatselt nutiseadmest rakendusse. Soovitatavad kohad, mida külastada on vahemaa poolest üksteisele lähedal ja on loogilises järjekorras, üldinfo reisi sihtkoha kohta on kasulik. Kuid siingi puudub piisav haldamise võimalus käsitletava probleemi lahendamiseks – kas kasutaja suudab näiteks läbida päevas 20 km ning kas tema eelarve lubab ööbida pakutavas hotellis.

Nagu analüüsist selgub, ei lahenda ükski vaadeldud rakendus probleemi täielikult. Kõige lähemal on sellele Roadtrippers ja Travefy, kuid neid kasutades peaks keerulisemate marsruutide loomisel tegema palju käsitööd ja mõnel juhul maksma rakenduse tasulise versiooni eest.

3 Tehnoloogiad

Käesolevas peatükis on kirjeldatud rakenduse arendamisel kasutatud tehnoloogiad ning nende valimise põhjused.

3.1 Veebirakendus

Reisiplaneerimise rakenduse teostamise platvormiks on valitud veebirakendus. Veebirakendused on populaarsuse poolest nutiseadmetest tagapool, kuid neil on oma tugevad ja nõrgad küljed. Üheks tugevaks küljeks võib lugeda veebirakenduste uuendamise lihtsust. Uuenduste nägemiseks ei pea kasutaja ise midagi tegema ega alla laadima – kogu protsess toimub rakenduse siseselt. Samuti on veebirakenduste arendamiseks sobilike programmeerimiskeelte ja raamistike valik suurem. Erinevalt muudest rakenduste tüüpidest on veebipõhised rakendused universaalselt kasutatavad. Need ei sõltu kindlatest seadmetest ning nende kasutamiseks on vajalikud üksnes veebibrauser ja internetiühendus. Kuna kasutaja informatsiooni hoitakse tavaliselt serveris, on see kättesaadav ka teisest seadmest veebilehele sisenedes ja ennast identifitseerides. Veebirakendustel on ka miinuseid. Nende töö nõuab pidevat internetiühendust. Kuigi tänapäeval on üsna levinud tasuta juhtmevabad võrgud, ei pruugi igas kohas siiski ühendust leida. Nagu näitab aina sagenev veebilehtede häkkimine ehk omavoliline sissemurdmine, ei ole veebirakendused alati väga turvaliselt üles ehitatud. Sellisel juhul võib kasutajate isiklik info lekkida ning tekitada neile kahju.

Valiku tegemisel omab põhilist rolli reisiplaneerimise rakenduse kasutamise põhimõte. Kuigi arvutiprogrammid võivad olla kiired ja turvalised, ei ole need antud ülesande teostamisel parimaks valikuks. Planeerimise järgselt reisile minnes võib kasutajal tekkida soov oma marsruuti järgi vaadata või korrigeerida, kuid algse arvuti puudumisel ei ole see võimalik. Mobiilirakendustel on palju tugevaid külgi, kuid need on seotud ühe operatsioonisüsteemiga. Kui reisijal tekib soov kasutada muud seadet, tuleb esmalt rakendus sellele paigaldada. Piiravateks teguriteks on internetiühendus, energia

tarbimine ja turvalisus, nagu ka veebirakendustel. Veebipõhine tarkvara vajab samuti internetiühendust, kuid on kasutatav kõikidel seadmetel: arvuti, telefon, tahvelarvuti. Kasutaja infot hoitakse väljaspool tema arvutit, mille tõttu piisab uuel seadmel ainult enda autentimisest. Viimaks võimaldab tehnoloogiate mitmekesisus kiiret ja mugavat arendust, mida saab kiiresti ka kasutajani viia.

3.2 Angular

Reisiplaneerimise veebirakenduse front-end tehnoloogiaks on TypeScript baasil töötav Angular 7. TypeScript omakorda on JavaScripti ülemhulk, mis võimaldab lisada staatilist tüübikirjeldust. Vaatamata nimele on tegemist Angular raamistiku 5. versiooniga, mille eelkäijaks oli AngularJS. Hilisemaid raamistiku versioone nimetatakse kokkuvõtvalt Angular või Angular 2+.

Angulari keskseks osaks on moodulid ja nendesse kuuluvad komponendid. Igal rakendusel peab olema vähemalt üks moodul – *AppModule*. Jagades rakenduse osad erinevatesse moodulitesse vastavalt seotud funktsionaalsusele või mõnele teisele kriteeriumile, on võimalik muuta arendusprotsessi efektiivsemaks ja tagada taaskasutatavus. Iga komponent koosneb klassist, mis implementeerib rakenduse loogikat ning selle HTML mallist, mis paneb paika visuaalse poole. Igas moodulis võib olla mitu komponenti ja igal komponendil võivad olla ka alamkomponendid. Funktsionaalsus, mis ei ole seotud ühe kindla komponendiga asub teenustes, mis lisatakse vajalikes kohtades sõltuvustena. Nii on tagatud taaskasutatavus ja modulaarsus[9].

Angular on põhiliselt suunatud SPA (*single-page-application*) veebirakenduste arendamisele. Taoliste rakenduste arendamine on kiirem ning nende kasutamine on tavaliselt inimesele mugavam ja sujuvam. Uue vaate avamine ei nõua kogu sisu uuesti laadimist – piisab üksnes vajalike komponentide muutmisest. Samuti on nii võimalik säilitada informatsiooni selle taaskasutamiseks, tagades rakenduse kiire töö. SPA rakenduste miinuseks on JavaScripti sisselülitamise vajadus ja halb otsingumootorite optimeerimine.

Raamistikuna teeb Angulari sobivaks selle populaarsus, millest tulenevalt on olemas suur hulk pakutavaid teeke. Mõned populaarsemad on näiteks *Angular Material*, *ngx-bootstrap* ja *Onsen UI*[10] . Populaarsuselt Angularile väga lähedal on *React*. Kuigi tegemist ei ole raamistikuga, pakub React palju samasugust funktsionaalsust. Mõlemad kasutavad rakenduse loomiseks komponente ning React on kiirem ja paremini optimeeritud otsingumootoritele. Põhjus, miks React sobib käsitletava probleemi lahendamiseks halvemini, seisneb väärtuste sidumises. Erinevalt Angularist ei paku React kahepoolset andmete sidumist[11] . See tähendab, et muutes mõnda reisiga seotud väärtust vormil, ei kajastu see koheselt rakenduse loogikas ja mudelites. Antud funktsionaalsus on marsruutide koostamisel tähtis ning seetõttu sobib kasutajaliidese arendamiseks rohkem Angular.

3.3 Java

Rakenduse back-end ehk teenuskiht on loodud kasutades Java 8 programmeerimiskeelt. Tegemist on klassipõhise objektorienteeritud keelega. Java tugevaks küljeks on selle sõltumatus riistvarast ja operatsioonisüsteemidest – kompileeritud programmi on võimalik käivitada masinal, mis toetab Java keelt. Seda eesmärki saavutatakse Java virtuaalmasina abil (*JVM*).

Objektorienteeritud keelena on Java põhiosaks objektid. Objektid koosnevad parameetritest ehk väljadest ja meetoditest ehk protseduuridest. Tarkvara loogika on üles ehitatud objektide omavahelise koostoime abil. Objektide käitumine ja väljad määratakse klassides, millel võivad omakorda olla alam- ja ülemklassid, mis laiendavad funktsionaalsust ja võimaldavad kasutada polümorfismi. Klasside välju tihtipeale peidetakse teiste klasside eest ning tehakse kättesaadavaks üksnes meetodite abil. Sellist käitumist nimetatakse kapseldamiseks[12] .

Kaks peamist funktsionaalsust, mis lisandusid keele 8. versioonis on *Stream* ja *Optional*. Stream ehk voog võimaldab toimetada kollektsioonidega, vältides seejuures ülearuseid tsükleid ja muutes koodi loetavamaks. Optional on Java lahendus situatsioonile, kus soovitakse näidata, et väärtus võib puududa. *Null* väärtuse tagastamine sellist infot ei anna ning võib tekitada segadust ja veaolukordi. Samuti

lisandus uus aja *API*, mis on arusaadavam, mugavam ja lahendab hargtöötluse probleemi.

Alternatiivne keel back-end tarkvara loomiseks on näiteks *PHP*. Mõned põhjused PHP valimiseks on selle populaarsus, funktsionaalsuse implementeerimise kiirus ja rakenduse kiire paigaldamine. Põhjused Java valikuks on tüübi- ja mäluohutus, kiirem hargtöötlus ja turvalisus. Java tugevad küljed on eeliseks suurte rakenduste loomisel. Kuna töö käigus loodaval rakendusel on palju potentsiaalset kasvuruumi, on selle loomiseks valitud just Java.

3.3.1 Spring Boot ja Gradle

Spring Boot on Java raamistik, mis võimaldab kiirelt luua töövalmis rakendusi. Erinevalt üldisest *Spring* raamistikust ei ole Spring Boot puhul vaja kirjutada keerulisi konfiguratsioone – rakenduse seadistamine toimub automaatselt kasutades *@SpringBootApplication* annotatsiooni. Muuhulgas hõlbustab Spring Boot sõltuvuste haldamist, rakenduse skaleerimist ja paigaldamist[13].

Gradle on rakenduse ehitamise automatiseerimise süsteem. Gradle abil on võimalik lihtsamini hallata mitmeosalisi projekte, automatiseerida rakenduse ehitamise ja käivitamisega seotud protsesse ja hallata sõltuvusi[14].

3.4 PostgreSQL

PostgreSQL on objekt-relatsiooniline andmebaasi haldamise süsteem. Andmebaasi tarkvara on tasuta ning selle lähtekood on avatud. Tähtsamad PostgreSQL poolt pakutavad funktsionaalsused on vaated, reeglid, alampäringud ja pesastatud transaktsioonid. Andmebaasi haldaja võib ise määrata uusi andme- ja indeksitüüpe ning implementeerida muudes programmeerimiskeeltes kirjutatud funktsioone[15].

Alternatiiv PostgreSQL süsteemile on MySQL, mis on samuti üks populaarsemaid andmebaasi haldamise süsteeme. Mõlemad pakuvad sarnast funktsionaalsust. Erandiks on näiteks *FULL OUTER JOIN*, mis on olemas ainult PostgreSQL'is. Erinevalt PostgreSQL'ist pakub MySQL ka ärilitsentsi. Andmebaasi haldamise süsteemi valikul

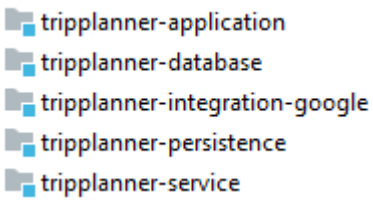
said otsustavaks peamiselt kasutusmugavus, töökindlus, keerulisemate päringute kiirus ning eelnev kogemus PostgreSQL kasutamisel.

4 Reisimarsruudi planeerimise veebirakenduse arendus

Käesolevas peatükis on kirjeldatud valminud rakenduse teostus ja selle põhilisem funktsionaalsus. Lisatud on rakenduse andmebaasi skeem (Lisa 1) ja arhitektuuri skeem (Lisa 2).

4.1 Back-end üldloogika

Rakenduse back-end projekt on jagatud eraldi alamprojektidesse ehk moodulitesse. Iga moodul vastutab ühe rakenduse osa eest (Joonis 1).



Joonis 1. Rakenduse teenuskihi moodulid.

- „*tripplanner-application*" vastutab rakenduse üldloogika ja front-end'i poolt saadetavate andmete vastuvõtmise eest. Siin on loodud rakenduse konfiguratsioonid, kontrollerid ning nende poolt kasutatavad andmeobjektid ja teenused. Samuti on selles moodulis rakenduse peamine klass – *TripPlannerApplication* (Lisa 3).

- „*tripplanner-database*" sisaldab endas andmebaasi *Dockeri* ja andmebaasi enda loomise skripte. Andmebaasi skriptid käivitatakse *Liquibase* teegi abil.

- „*tripplanner-integration-google*" sisaldab endas teenuseid, mille abil toimub suhtlus erinevate Google Maps API'dega.

- „*tripplanner-persistence*" vastutab suhtluse eest andmebaasiga. Siin on kirjeldatud andmeobjektid, mis vastavad andmebaasi tabelitele, *RowMapper*'id (andmebaasi päringu ridade Java objektideks teisendamisega tegelevad klassid) ja *DAO* klassid (andmebassi päringuid sooritavad klassid).

- „*tripplanner-service*” koosneb rakenduse põhilisest äriloogikast. Moodulis on loodud teenused, mis vahendavad suhtlust kontrollerite ja DAO klasside vahel, tehes arvutusi ja vajalikke lisapäringuid.

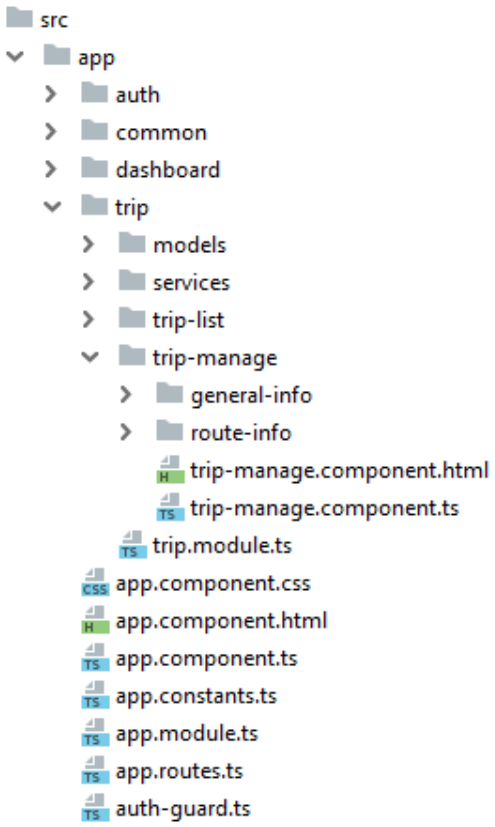
Andmebaasi ühenduse parameetrid on kirjeldatud failis *tripplanne-api.properties*. Rakenduse parameetrid on määratud käivitamise ajal:

- *config.dir* – täielik tee kaustani, kus asub *tripplanne-api.properties* fail.
- *google.integration.api-key* – Google Maps API unikaalne võti.

Rakendus on käivitatav kasutades Apache Tomcat serverit.

4.2 Front-end üldloogika

Rakenduse front-end projekt on loodud kasutades *Bootstrap 4 CSS* raamistikku. Ühe rakenduse komponendi moodustavad tema TypeScript, HTML ja mõnikord CSS failid. Komponendid on funktsionaalsuse järgi jagatud erinevatesse kaustadesse (Joonis 2):



Joonis 2. Rakenduse esitluskihi kaustade struktuur

- „*auth*” kaust sisaldab endas komponente ja teenuseid, mis vastutavad rakenduse autentimise eest.
- „*common*” kaust sisaldab rakenduse taaskasutatavaid komponente nagu tabelid ja sõnumid. Need on lisatud *SharedModule* moodulisse, mida kasutavad kõik teised moodulid.
- „*dashboard*” kaust sisaldab veebilehe raamiga seotud komponente.
- „*trip*” kaustas asuvad rakenduse põhifunktsionaalsusega seotud komponendid. Need tegelevad kasutaja reiside kuvamise ja haldusega.

Iga eelnimetatud kaust sisaldab moodulit, kus on kirjeldatud komponendid ja teenused, mis selles kasutas on. Iga moodul on omakorda kirjeldatud põhimoodulis *AppModule*.

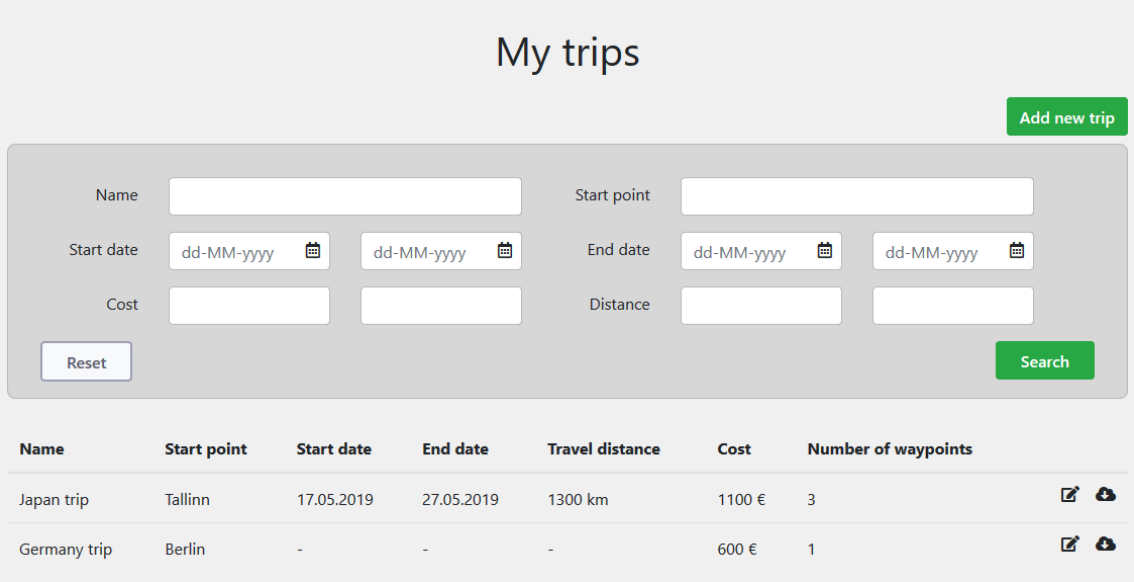
Rakenduse veebiaadressid on loetletud failis *app.routes.ts*. Mõned veebiaadressid vajavad avamiseks autentimist – sellisel juhul kontrollitakse kasutaja privileege *auth-guard.ts* abil. Kuna rakenduse front-end ja back-end asuvad erinevate portide peal, on päringuid vaja ümber suunata. Vajalikud sätted on kirjeldatud failis *proxy.dev.conf.json*.

Lisaks Bootstrap 4’le on rakenduses kasutusel *ngx-bootstrap* ja *Moment.js* teegid. ngx-bootstrap pakub kasutamiseks mitmeid komponente. Näiteks on rakenduses kasutusel kuupäeva ja kellaaja valikud, rippmenüüd ja hüpikdialoogid. Moment.js võimaldab palju paremini opereerida TypeScript kuupäevade ja kellaaegadega.

4.3 Reiside nimekiri

Peale sisselogimist avaneb rakenduse kasutajale leht, kus on loetletud tema eelnevalt loodud reisid (Joonis 3). Lisaks nimekirjale on lehel otsinguvorm, mille abil on võimalik reiside nimekirja täpsustada.

Soovi korral on võimalik täpsustada reiside nimekirja kasutades otsinguvormi. Väärtused, mille abil on võimalik reisi otsida: nimi, alguspunkt, algkuupäeva vahemik, lõppkuupäeva vahemik, maksumuse vahemik ja vahemaa vahemik. Vajutades nupule „Reset” väljad tühjendatakse ja näidatakse uuesti kõiki reise.



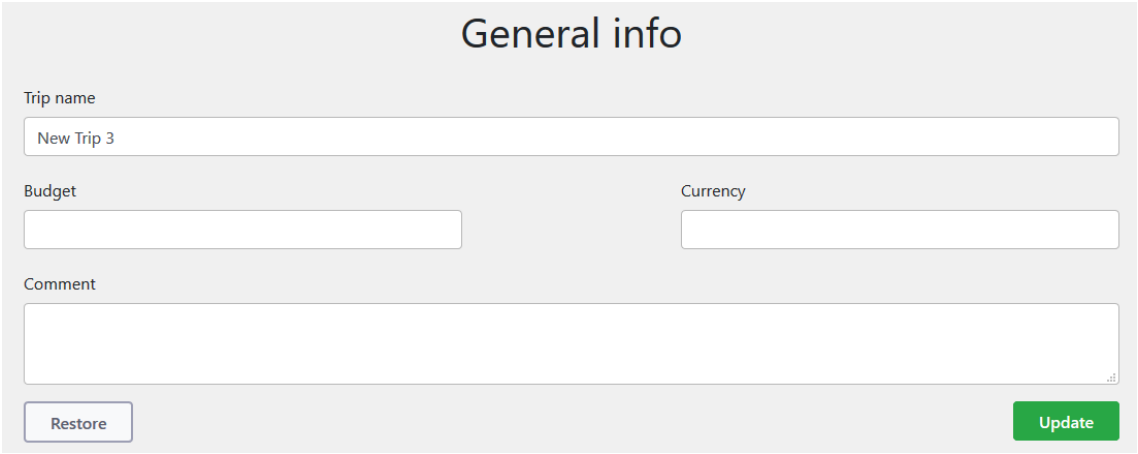
Joonis 3. Kasutaja reiside nimekirja vaade.

Esialgses reiside loetelus on välja toodud kõik kasutaja reisid. Iga reisi kohta näidatakse selle üldinfot: nimi, alguspunkt, algkuupäev, lõppkuupäev, läbitav vahemaa, maksumus koos valuutaga ja reisi sihtpunktide koguarv. Väärtuse puudumisel näidatakse selle asemel „-”. Iga rea lõpus on nupud reisi muutmise vaatesse liikumiseks ja reisi allalaadimiseks. Otsinguvormi kohal on nupp uue reisi lisamiseks. Sellele vajutades luuakse uus reis ning avatakse selle muutmise vaade.

4.4 Reisi muutmine

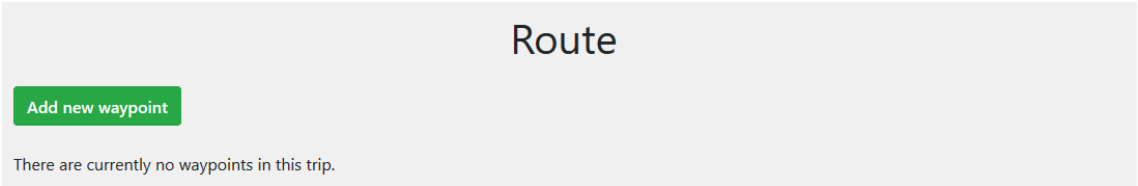
Peale uue reisi lisamist või vana reisi muutmise nupule vajutamist avaneb reisi muutmise vaade. Samuti võib avada kasutajaga seotud reisi vaate veebiaadressi järgi, teades reisi unikaalset identifikaatorit UUID v4 formaadis.

Muutmise vaate ülemises osas on vorm reisi üldinfoga: nimi, eelarve, valuuta ja kommentaar (Joonis 4). Uut reisi luues määrab rakendus vaikimisi nimeks „New Trip X”, kus X = kasutaja reiside arv + 1. Soovi korral võib seda ja teisi väärtusi muuta. Salvestamiseks tuleb vajutada nuppu „Update”. Juhul kui päring õnnestub, ilmub roheline teade vastava sisuga. Päringu ebaõnnestumise korral on teade punane. Kui muutmise käigus kasutaja otsustab, et ei soovi muudatusi salvestada, on tal võimalik taastada eelnevalt salvestatud seisu vajutades nuppu „Restore”.



Joonis 4. Reisi üldinfo muutmise vaade.

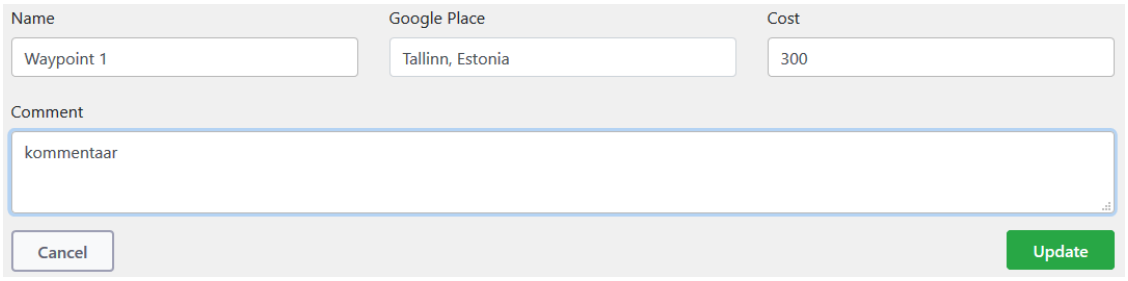
Muutmise vaate alumises osas on kirjeldatud reisi marsruut. Uue reisi puhul marsruuti veel ei ole ning selle asemel kuvatakse informatiivset teksti sihtpunktide puudumise kohta (Joonis 5). Selle koha on nupp uue sihtkoha lisamiseks. Vajutades nupule luuakse andmebaasis sihtpunkt viitega avatud reisile ning nupu alla tekib nimekiri reisiga seotud sihtpunktidest ja nende vahelistest marsruutidest (Joonis 6). Loodud sihtpunktile antakse vaikimisi nimetus „Waypoint X", kus X = sihtpunkti järjekord marsruudis.



Joonis 5. Uue reisi marsruudi muutmise vaade.

Joonis 6. Nimekiri reisi sihtpunktidest.

Iga sihtpunkti rea lõpus on nupud selle muutmiseks või kustutamiseks. Vajutades kustutamise nupule tekib kõrvale kinnitusdialoog. Kustutamise soovi kinnitades eemaldatakse andmebaasist vastav sihtpunkt ja kõik sellega seotud marsruudid. Vajutades muutmise nupule avaneb nimekirja all vorm selle sihtpunkti andmetega (Joonis 7). Muuta on võimalik nime, maksumust ja kommentaari. Samuti on võimalik teostada Google Places API otsingut, mille tulemuse valimisel salvestatakse sihtpunktile koordinaadid ja Google Place ID. ID järgi kuvatakse nimekirjas sihtpunkti tegelikku nimetust.



Joonis 7. Reisi sihtpunkti muutmise vorm.

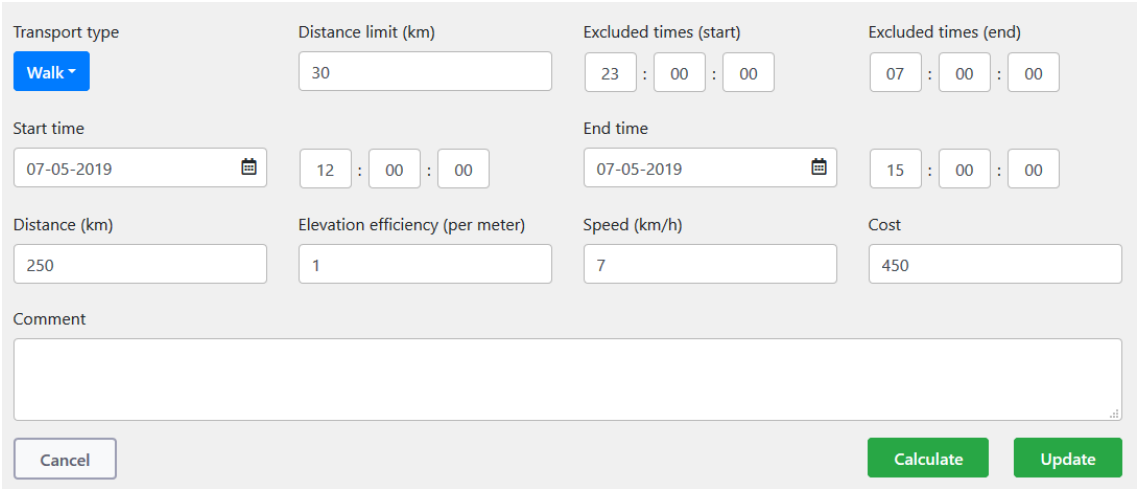
Lisades reisile veel ühe sihtpunkti, luuakse andmebaasis automaatselt marsruut uue ja eelneva sihtpunktide vahel (Joonis 8). Marsruuti on võimalik ainult muuta – kustutamine toimub algus- või lõpppunkti kustutamisel automaatselt.

Name	Google Place	Start	End	Cost	Distance	
Waypoint 1	Tallinn			300 €		
		07.05.2019 12:00:00	07.05.2019 15:00:00	450 €	250 km	
Waypoint 2	Tartu			300 €		
				1050 €	**250 km**	

Joonis 8. Nimekiri reisi sihtpunktidest ja marsruutidest.

Vajutades marsruudi muutmise nupule, avaneb selle muutmise vorm (Joonis 9). Eelnevatest vormidest eristab seda „Calculate" nupp, mis eeltäidab vormi vastavalt juba sisestatud andmetele. Arvutuste eelduseks on, et nii algus- kui ka lõpppunktil on määratud Google Place ID. Transpordivahendite rippmenüü täidetakse andmebaasi tabeli *transport* väärtustega.

Esialgu arvutatakse kahe punkti vaheline vahemaa kilomeetrites. Juhul, kui marsruudil on määratud ainult algusaeg, arvutatakse vahemaa läbimiseks kuluv aeg. Aja arvutamisel võetakse kiiruseks salvestatud kiirus või selle puudumisel valitud transpordivahendi eelseadistatud kiirus. Auto puhul on see 70 km/h, jalgratta puhul 15 km/h ja kõndimise puhul 5 km/h. Lõppaja arvutamisel võetakse arvesse tõusumeetri koefitsient, päevane läbitava vahemaa piirang ja kellaajalised piirangud. Juhul, kui marsruudil on määratud hoopis lõppaeg, toimub arvutamine sarnaselt, kuid liitmise asemel toimub aja lahutamine.

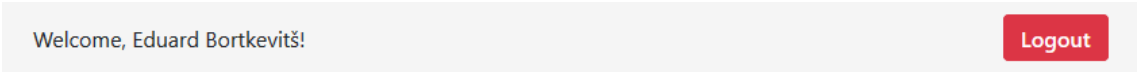


Joonis 9. Kahe sihtkoha vahelise marsruudi muutmise vorm.

Kõikide reisi sihtkohtade ja nendevaheliste marsruutide maksumused liidetakse kokku ja näidatakse nimekirja all. Kui reisi eelarve on määratud, võrreldakse saadud summat selle arvuga. Eelarve ületamise korral on näidatav summa punane. Vastasel juhul on see roheline. Samuti liidetakse informatiivsel eesmärgil kokku kõikide marsruutide vahemaad.

4.5 Veebilehe raam

Veebilehe raam on nähtav igal rakenduse lehel ning koosneb lehekülje päisest (Joonis 10).

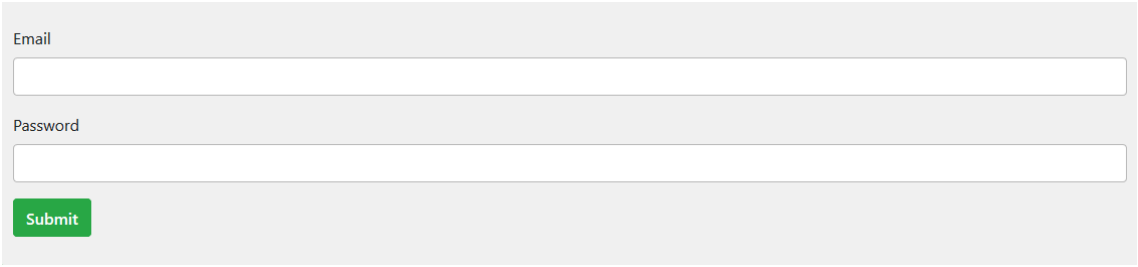


Joonis 10. Veebilehe päis.

Päises on kuvatud sisseloginud kasutaja ees- ja perekonnanimi koos tervitusega. Paremas ääres paikneb väljalogimise nupp. Vajutades nupule avaneb kinnitusdialoog. Kinnitades oma soovi välja logida, logitakse kasutaja välja ning suunatakse sisselogimise lehele.

4.6 Sisselogimine

Rakenduse avamisel ilma sisse logimata, avaneb kasutajale sisselogimise leht (Joonis 11).



Joonis 11. Sisselogimise vaade.

Kasutajal on võimalik sisestada enda email ja parool, mis saadetakse autentimise eesmärgil teenuskihti. Juhul, kui kasutaja info on korrektne, logitakse ta sisse ja suunatakse reiside nimekirja vaatesse.

5 Reisimarsruudi planeerimise veebirakenduse analüüs

Käesolevas peatükis on analüüsitud valminud rakendust ja selle valideerimist. Samuti on välja toodud võimalikud täiendused, mida tulevikus rakendusele teha.

5.1 Rakenduse valideerimine

Rakenduse valideerimine toimus selle sihtgruppi kuuluvate kasutajate poolt. Testimise järgselt oli kasutajatel võimalik jätta rakendusele hinnang järgnevate kriteeriumite järgi: kasutusmugavus, funktsionaalsus, töökindlus, võrdlus teiste taoliste rakendustega.

5.1.1 Kasutusmugavus

Üldiselt olid kasutajad rakenduse mugavusega rahul. Positiivsena toodi välja nuppude värvuse sõltuvus funktsionaalsusest, kustutamise puhul kinnituse küsimine ja rakenduse kiire töö. Rakenduse vaated nägid välja korrektsed olenemata brauserist ja ekraani suurusest.

Negatiivsena toodi välja rakenduse keelevaliku puudumise – rakendust on võimalik kasutada ainult inglise keeles. Lisaks arvasid mitmed kasutajad, et rakenduse stiilid on vanamoodsad.

5.1.2 Funktsionaalsus

Rakenduses realiseeritud funktsionaalsusega olid kasutajad rahul. Eriti meeldis neile muutujate rohkus marsruudi eeltäitmisel. Positiivne oli ka võimalus jätta pika kommentaari nii reisile üldiselt, sihtkohale kui ka sihtkohtade vahelisele marsruudile.

Segadust tekitas vajadus seostada algus- ja lõppsihtkoht Google Maps asukohaga, et seejärel automaatselt arvutada nendevahelist marsruuti. Kasutajate sõnul oleks mugavam, kui rakendus kuvaks vastavasisulist infoteadet. Lisaks oli arusaamatu, kuidas rakendati tõusumeetri koefitsienti. Puuduvast funktsionaalsusest toodi välja võimalus

muuta sihtpunktide järjekorda ning lisada sihtkoha koordinaadid ilma Google Maps otsinguta.

5.1.3 Töökindlus

Testimise käigus jälgisid kasutajad, et rakendus töötaks nii nagu peaks. Suurem osa kasutajaid jäid sellega rahule. Mõnel juhul ei toiminud kellaaja väljad õigesti – väärtus muutus peale uuendamise või arvutamise nupu vajutamist. Seda ei peetud kriitiliseks veaks, kuna arvutused toimusid õigesti ja tulemused olid korrektsed.

5.1.4 Võrdlus teiste rakendustega

Kasutajate arvates täitis rakendus oma rolli sihtkohtade vaheliste marsruutide arvutamisel, pakkudes funktsionaalsust, mida teistes sarnastes rakendustes ei ole. Sisendina võetavate muutujate rohkust loeti väga positiivseks. Siiski on loodud rakenduse funktsionaalsus piiratud ning teised rakendused suudavad pakkuda teenuseid, mida antud rakenduses realiseeritud ei ole.

5.1.5 Valideerimise kokkuvõte

Kokkuvõttes olid rakendust testivad kasutajad selle tööga rahul. Esines mõningaid väiksemaid vigu, kuid need ei takistanud rakenduse tööd. Eriti olid kasutajad rahul reisimarsruutide automaatse arvutamisega. Peamised välja toodud puudujäägid olid keelevaliku puudumine, ja ebapiisav info arvutuse parameetrite kohta.

5.2 Rakenduse analüüs

Rakenduse valideerimine potentsiaalsete kasutajate poolt näitas, et töö käigus loodud rakendus töötab ilma suuremate probleemideta ning pakub funktsionaalsust, mis esialgselt plaanitud oli. Funktsionaalsuse poolest on rakendusel potentsiaali – teised reisiplaneerimise veebirakendused pakuvad sarnaseid haldusvahendeid ja arvutusi, kuid ei ühenda endas korraga salvestatavate andmete rohkust ja marsruutide arvutamise automatiseerimist.

Rakenduses on realiseeritud kasutaja poolt loodud reiside nimekiri ja otsing. Nimekirjas kuvatakse piiratud hulka andmeid, mis peaks andma hea ülevaate reisist. Kuvatavad andmed saadakse otse andmebaasist või arvutuste käigus. Otsinguvormi abil on võimalik piirata kuvatavate reiside hulka ning leida just need, mis kasutajat huvitavad. Lisaks on rakenduses realiseeritud individuaalse reisi haldamine. Haldusvormil saab muuta reisi üldinfot ning vaadata ja muuta sihtkohti ja nendevahelisi marsruute. Sihtpunktide ja marsruutide nimekirjas kuvatakse samuti ainult osa andmeid – täieliku komplekti nägemiseks tuleb avada selle muutmise vorm. Marsruudi vormi on võimalik teatud tingimustel eeltäita.

Autori arvates on rakendus oma eesmärgi saavutanud ning on soovi korral kasutatav. Kuna alternatiivseid reiside ja nende marsruutide planeerimisega tegelevaid rakendusi on palju, nõuab laialdasem kasutamine mitmeid täiendusi rakenduse funktsionaalsuses, et olla konkurentsivõimeline ja pakkuda seejuures unikaalseid teenuseid.

5.3 Rakenduse täiendused

Rakenduses on realiseeritud esialgselt plaanitud funktsionaalsus, kuid selleks, et see oleks laialdaselt kasutatav, on funktsionaalsust vaja täiendada. Järgnevalt on kirjeldatud, kuidas võiks rakendust edasi arendada:

- Marsruutide näitamine kaardil – rakenduses on küll realiseeritud marsruutide arvutamine sisestatud andmete põhjal, kuid tulemust näidatakse tekstilisel kujul. Visuaalselt annaks kaart parema ülevaate sellest, kuidas marsruut päriselt välja näeb.

- Sihtkohtade laadimine liidestustest – sihtkohti on hetkel võimalik lisada ainult käsitsi, sisestades andmed ja otsides Google Maps API kaudu. Mugavuse eesmärgil peaks rakendus olema võimeline laadima reisi infot mõnest liidestusest, näiteks Booking.com ja Airbnb.

- Marsruudi loomine sihtkohtade vahel, millel puudub Google Place ID – rakenduses realiseeritud arvutused nõuavad, et algus- ja lõpppunktid oleks seotud Google Place kohaga läbi selle identifikaatori. Funktsionaalsuse

laiendamiseks peaks olema võimalik teostada arvutusi ka siis, kui on määratud ainult koordinaadid. Samuti peaks olema võimalik lisada koordinaate sõltumatult Google Maps API'st.

- Sihtkohtade järjekorra muutmine – see funktsionaalsus lubaks kasutajal igal ajal luua uue sihtpunkti ning liigutada see reisi keskele. Loodud rakenduses selline võimalus puudub, ning sihtkohad peavad olema sisestatud õiges järjekorras algusest peale.

- Keelevalik – rakenduse valideerimise käigus kurtsid mitmed kasutajad, et veebilehe sisu on nähtav ainult inglise keeles. Keelte arvu tuleks laiendada, lisades eesti ja vene keele.

Loetletud täiendused on kõige suurema potentsiaalse mõjuga rakenduse mugavusele ja funktsionaalsusele. Nende realiseerimisel tõuseks rakenduse väärtus kasutajatele ning suureneks konkurentsivõime.

6 Kokkuvõte

Bakalaureusetöö eesmärgiks oli luua reisimarsruudi planeerimise veebirakendus. Töös oli analüüsitud olemasolevaid reisiplaneerimise rakendusi, põhjendatud tehnoloogilisi valikuid, kirjeldatud valminud rakenduse funktsionaalsust ja selle teostamist, analüüsitud kasutajate poolt valideerimise tulemusi ning välja toodud potentsiaalsed funktsionaalsuse täiendused.

Rakenduse kasutajaliides oli realiseeritud kasutades Angular 7 raamistikku. Teenuskiht oli loodud Java 8 programmeerimiskeeles, kasutades Spring Boot raamistikku ja Gradle kompileerimise automatiseerimise süsteemi. Rakendusele on loodud PostgreSQL 11 andmebaas.

Valminud rakendus lubab kasutajal luua reisiplaane ja näha nimekirja juba varem loodud reisidest. Nimekirja on võimalik otsinguvormi abil filtreerida. Reisiplaani alla võib kasutaja lisada sihtpunkte, sisestades lisainfot ja seostades Google Maps API asukohaga. Kahe sihtpunkti vahele tekib rakenduses marsruut, mida on võimalik käsitsi uuendada või eeltäita juhul, kui algus- ja lõpppunktid on seotud reaalse asukohaga. Eeltäitmise käigus arvutatakse marsruudi puuduvad väärtused võttes arvesse sisestatud kellaaja ja vahemaa piirangud ning tõusumeetri koefitsiendi. Lisaks on sihtkohtadele ja marsruutidele võimalik lisada maksumused, mida pidevalt võrreldakse reisi eelarvega.

Kasutajate poolt valideerimine näitas, et töö käigus valmis töötav rakendus, mis täidab kõiki seatud põhieesmärke ja võimaldab kasutajal luua reisiplaane ja neid hallata.

Kasutatud kirjandus

[1] Holiday Habits Report 2018 – *Association of British Travel Agents*, 2018, 6. [WWW] https://www.abta.com/sites/default/files/2018-10/Holiday%20Habits%20Report%202018%20011018.pdf (18.03.2019)

[2] Angular – What is Angular? [WWW] https://angular.io/docs (17.03.2019)

[3] Spring Boot. [WWW] https://spring.io/projects/spring-boot (17.03.2019)

[4] PostgreSQL: About. [WWW] https://www.postgresql.org/about/ (17.03.2019)

[5] About – Itineree. [WWW] http://itineree.com/about/ (25.03.2019)

[6] About | Road Trip Route Planner, Map and Trip Guides. [WWW] https://maps.roadtrippers.com/about (26.03.2019)

[7] Travefy – Trip Itinerary. [WWW] https://travefy.com/ (26.03.2019)

[8] Google Trips. [WWW] https://get.google.com/trips/ (27.03.2019)

[9] Angular – Architecture Overview. [WWW] https://angular.io/guide/architecture (11.04.2019)

[10] Angular – Explore Angular Resources. [WWW] https://angular.io/resources (11.04.2019)

[11] React vs. Angular: The Complete Comparison – Programming with Mosh. [WWW] https://programmingwithmosh.com/react/react-vs-angular/ (10.04.2019)

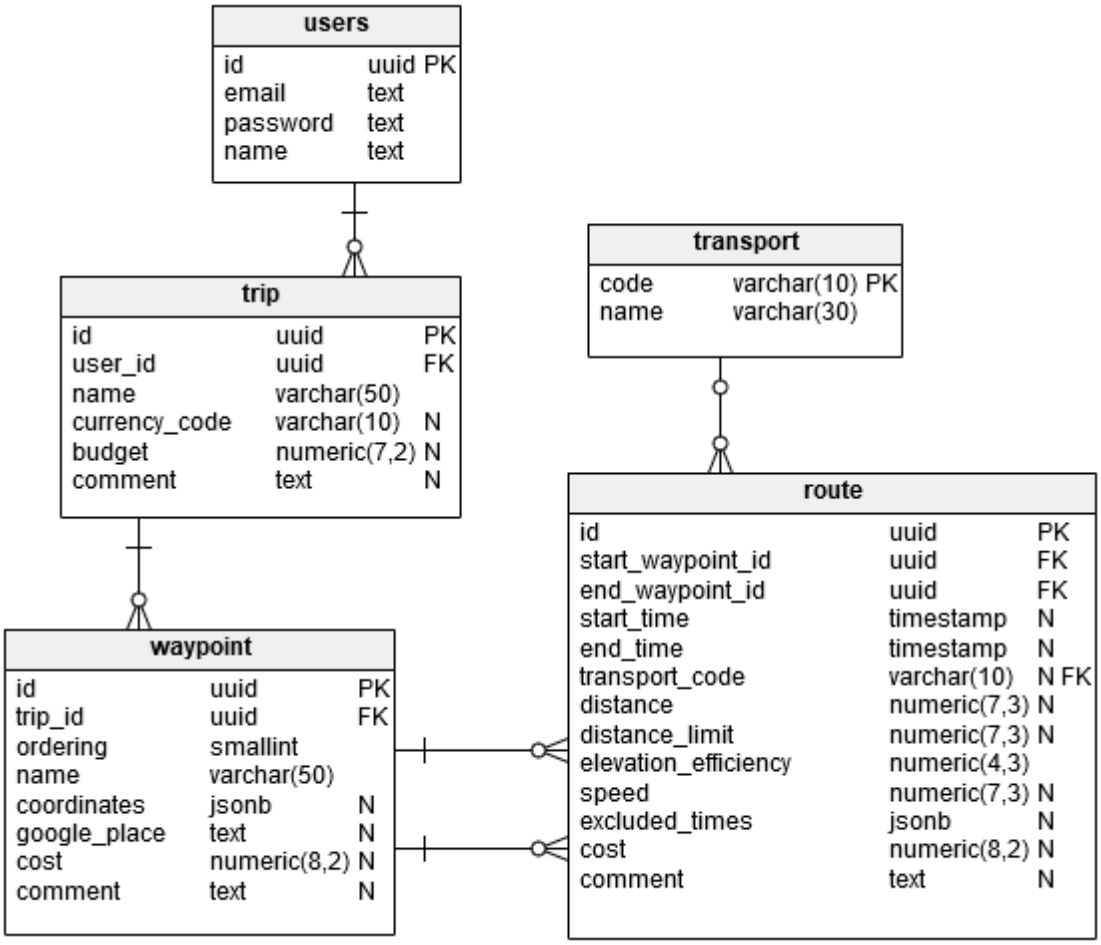
[12] Java objektorienteeritud programmeerimise kontsept. [WWW] https://ained.ttu.ee/javadoc/oopGeneral.html (15.04.2019)

[13] Spring Boot Introduction. [WWW] https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm (16.04.2019)

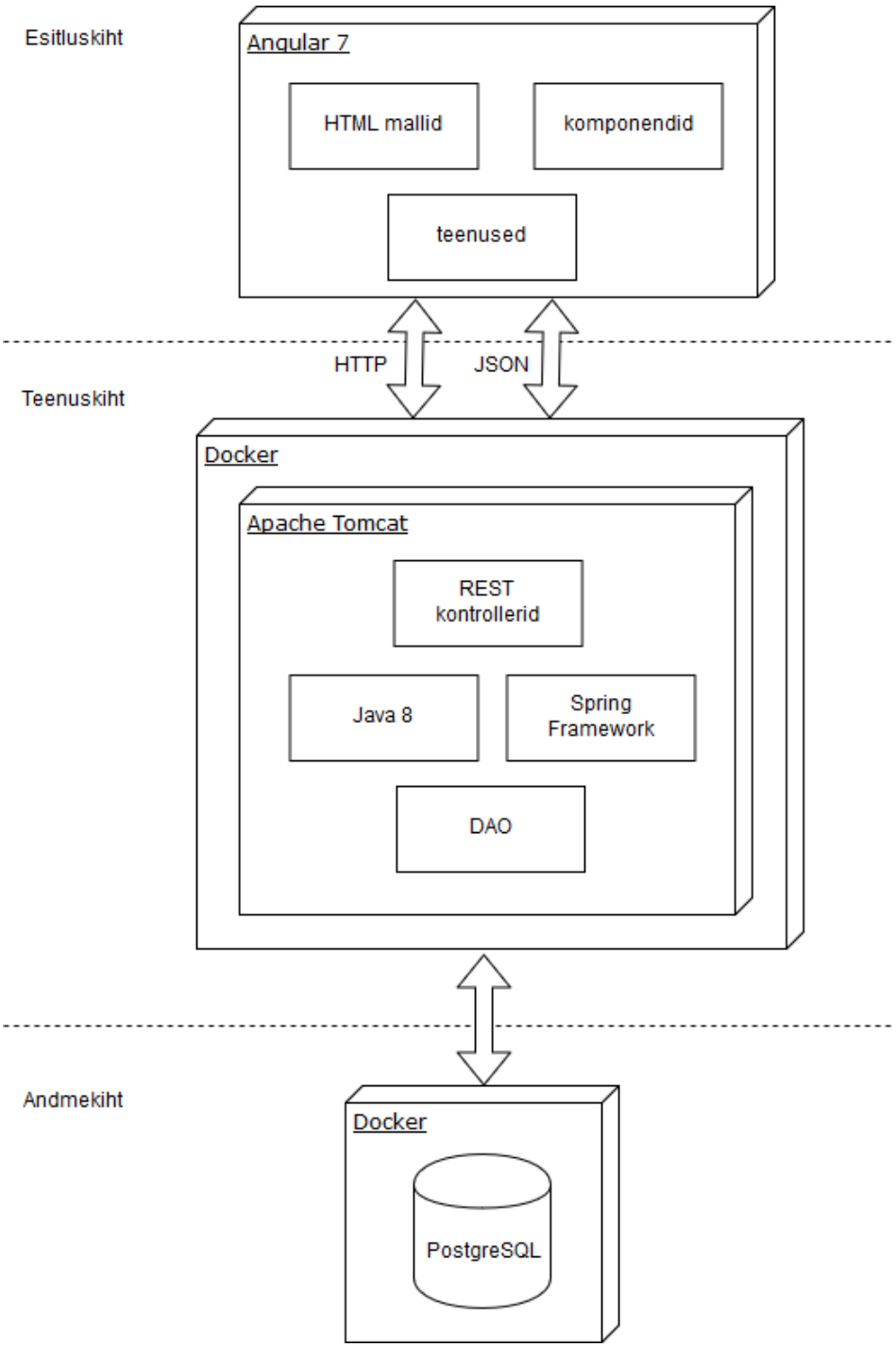
[14] Gradle Overview. [WWW] https://www.tutorialspoint.com/gradle/gradle_overview.htm (16.04.2019)

[15] What is PostgreSQL. [WWW] http://www.postgresqltutorial.com/what-is-postgresql/ (18.04.2019)

Lisa 1 – Andmebaasi skeem



Lisa 2 – Arhitektuuri skeem



Lisa 3 – TripPlannerApplication klass

```
@PropertySource(value = "file:${config.dir}/tripplanner-api.properties",
ignoreResourceNotFound = true)
@SpringBootApplication
public class TripPlannerApplication extends SpringBootServletInitializer
implements WebApplicationInitializer {
  @Override
  protected SpringApplicationBuilder configure(SpringApplicationBuilder
builder) {
    return super.configure(builder);
  }

  public static void main(String[] args) {
    SpringApplication.run(TripPlannerApplication.class, args);
  }
}
```