TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Taavi Meier 185150IABB
Gert Mänd 179157IABB
Kevin Kiil 185321IABB

Parkimiskohtade jagamise ja haldamise rakendus

Bakalaureusetöö

Juhendaja: Tarvo Treier
Magistrikraad

Tallinn 2021

Autorideklaratsioon

Kinnitame, et oleme 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.

Autorid: Taavi Meier, Gert Mänd, Kevin Kiil

18.05.2021

Annotatsioon

Käesoleva töö eesmärgiks oli luua parklakohtade jagamise ja haldamise rakendus. Rakendus on mõeldud ettevõtetele, korteriühistutele ja muudele asutustele, kus on kasutada parkla. Parklatega on nimetatud asutuste puhul tihti probleemiks kasutajate suurem arv võrreldes parklakohtadega. Sarnase probleemiga tegelesid autorid Registrite ja Infosüsteemide Keskusele sarnast rakendust luues. Sealt kasvas välja idee luua universaalsem rakendus, mis lahendaks sama probleemi ka teistes asutustes.

Lõputöö käigus valmis rakendus, kus on võimalik registreerida kasutajaid ja asutusi ning loodud asutustes üles seada asutuse kasutajate ja parklakohtade keskkond. Kasutajad saavad parklakohti vabastada ja broneerida. Administraatorid omavad ülevaadet asutuse kasutajatest, parklakohtadest ning logidest, kus on kirjeldatud kõik rakenduses tehtavad toimingud. Lisaks on võimalik nimetatud elemente muuta.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 44 leheküljel, 3 peatükki, 19 joonist, 3 tabelit.

Abstract

Parking Spot Sharing and Management Application

The aim for this project was to create an application for institutions with parking lot/parking garage. Often, the number of potential users is higher than the number of parking spots. In every-day use users who have designated spot do not use the spot 24/7. For example, during vacation or leave these spots are unused or are used without good management system.

For institution administrators authors confirmed the need for a management system, where the information about parking spots and users is provided and they can configure different aspects of these objects.

Predecessor for this thesis was a project for The Centre of Registers and Information Systems, where a similar problem was faced. From that project came the idea for more universal solution.

As a result, an application was created in which users can create accounts and institutions. Users (administrators) who create institutions can setup environment for other users in their institution by adding parking spots, users etc. Users can then free and book parking spots and administrators have overview of what is happening inside their institutions with the option to modify business-critical factors.

The thesis is in Estonian and contains 44 pages of text, 3 chapters, 19 figures, 3 tables.

Lühendite ja mõistete sõnastik

.NET-raamistik	Microsofti tarkvaraplatvorm
API	Application Programming Interface ehk liides rakenduse loogikale
ASP.NET Core	Veebirakenduste jaoks loodud raamistik
Back-end	API koos loogikaga, andmebaasi ja kasutajaliidese andmevahetusest
Docker	Andmebaasirakendus
Enterprise Architect	Rakendus modelleerimiseks ja kujundamiseks
Front-end	Kuvand kasutajale
GitHub	Tarkvaraarenduse veebimajutust pakkuv ettevõte
HTTP meetod	Päringutüüp
JSON Web Token	Krüpteeritud andmed JSON objekti kujul
Mapper	Andmemuster andmete töötlemiseks objektide vahel
Material-UI	Kasutajaliideste jaoks loodud disainikeel
Netlify	Pilveteenust pakkuv ettevõte
NUnit	Ühiktestide jaoks loodud raamistik, mis toetab C# keelt
Postman	Rakendus API testimiseks ja päringute saatmiseks
Queries	Päringud API-ga suhtlemiseks
React	JavaScripti teek kasutajaliidese loomiseks
RIK	Registrite ja Infosüsteemide Keskus
Visual Studio Code	Veebilehtede programmeerimistarkvara

Sisukord

Joonised

Tabelid

Sissejuhatus

Parklakohtade puudusega puutuvad kokku paljud ettevõtted, korteriühistud ja muud asutused, kus on parkla. Enamasti on kohtadele määratud kindlad kasutajad või lähtutakse „kuni kohti jätkub“ loogikast. See ei soosi parkla efektiivset kasutamist. Autorid peavad siinkohal silmas seda, et parkimiskoha hõivatus võiks olla ajalises raamis pikem.

Antud projekti eel-tööna ehitasid autorid sarnase probleemi lahendamiseks rakendust Registrite ja Infosüsteemide Keskusele (RIK). RIK-i administraatoritega suheldes tulid välja kindlad probleemid:

- Parklakohtadele on määratud kindlad kasutajad, kes kohta püsivalt ei kasuta (puhkus, kodukontor). See tähendas omakorda, et parklakoht seisab tühjana ning ei teeninda efektiivselt kasutajaid.
- Administraatoritel puudus ülevaade kohtade kasutamisest ja näiteks esines olukordi, kus kasutajad helistasid administraatorile ja küsisid, kas parklas on vabu kohti.
- Olukordades, kus mõni juht oli parkinud valele kohale ei olnud lihtsat viisi, kuidas teha kindlaks valel kohal parkinud autojuht.

Sellest lähtuvalt valmis esialgne prototüüp, kus kasutajad saaksid parklakohti vabastada ning broneerida. Lisaks oli suur komponent administraatorite töö lihtsustamine.

RIK-ile rakendust ehitades nägid autorid probleemi üldisemat iseloomu ning RIK-ist saadud info põhjal otsustati ehitada üldine *Software as a Service* tüüpi rakendus, kus kasutajad saavad püsti seada enda asutusele keskkonna ning seal enda asutuse parklakohti jagada ning jälgida kõike parkla kasutamisega seonduvat.

Analüüsi tarbeks suhtlesid autorid korteriühistute juhatuseliikmetega ning nendest vestlustest tuli välja, et peamine probleem on valesti parkimine ning sellele järgnev teavitustöö – kirjutatakse ühistule, kes omakorda kirjutab kõikidele elanikele ning palub tähelepanelikumalt parkida. Intervjueeritavad leidsid, et selline rakendus aitaks kindlasti

lihtsustada murede lahendamist ning oleks mugav töövahend kiirelt valesti parkijate tuvastamiseks või neile otse teavituse saatmiseks.

Rakendust ehitades said autorid täiendavat inspiratsiooni test-kasutajatelt, kes omakorda jagasid ideid, mis nende meelest rakenduse väärtust veel tõstaksid. Valminud rakendusse need funktsionaalsused veel ei jõudnud, aga analüüsi seisukohast said autorid tugeva signaali, et vajadus sellise lahenduse järele on olemas.

Täna pakub lahendust parklate efektiivsemaks kasutamiseks ka teisi ettevõtteid, näiteks Barking [1]. Loodud rakenduse suurim erinevus seisneb personaalsemas lähenemises – kasutajad saavad ise vabamalt oma kohta vabastada ning teised kasutajad kerge vaevaga vabastatud kohta broneerida. Rakendusest saab lisaks sügavama ülevaate parklas parkivatest autodest ja nende omanikest.

Rakenduse puhul tuvastasid autorid olulise kitsaskoha inimfaktori ja kolmandate osapoolte poolt pakutavate lahenduste näol. Neist esimesel juhul baseerub kogu kasutamise efektiivsus kasutajate tähelepanelikkusel ja tahtel rakendust kasutada. RIK-i puhul tuli omakorda välja keerukus kolmandate osapoolte poolt pakutud lahenduste osas, täpsemalt numbrituvastuse osas. Nimelt oli RIK-is kasutatav numbrituvastuse süsteem turvafirma hallata ning sellega meie rakenduse sidumine ei olnud võimalik. Käesoleva projekti raames valminud rakendusele koostasid autorid lihtsad liidestuse näited autode numbrimärkide ja kasutajate telefoninumbrite tuvastamiseks.

Peamised loodud funktsionaalsused:

- Asutuste lisamine
- Kasutajate lisamine/muutmine/kustutamine, sh. auto numbrimärkide info lisamine
- Parklakohtade lisamine/muutmine/kustutamine, sh. kasutajate õiguste määratlemine.
- Parklakohtade vabastamise ja broneerimise funktsionaalsus
- *Back-end* suutlikkus tuvastada numbrimärke ja telefoninumbreid.

Läbivalt oli oluline jätta võimalikult palju konfigureeritavaid kohti, mis annab kasutajatele maksimaalselt vabadust rakendust enda nägemuse järgi kohandada.

Käesolev töö annab ülevaate kasutatud tööriistadest, rakenduse arhitektuuris ja disainist. Lahti on kirjutatud tööprotsessid ning töö käigus ilmnenud kitsaskohad. Lisaks saab lugeja ülevaate kõikidest teostatud funktsionaalsustest ning töö tulemuste analüüsist. Viimase osana on välja toodud töö koostajate eneseanalüüs.

Töö jaguneb kolme suurde plokki – metoodika, tulemused ning analüüs ja järeldused.

1 Metoodika

1.1 Objekti detailne kirjeldus

Projektiga alustati RIK-is ning saadi valmis esialgne prototüüp. Rakenduses oli võimalik hallata parklakohti, loovutada neid kaastöötajatele või vabastada teistele broneerimiseks. Broneeringutele oli kontrolliks lisatud alg- ja lõppkuupäev, et vältida topelt broneerimist. Vabastatud parklakoha otsimisel leiti kasutajale parim parklakoht valitud perioodil. Otsijal polnud võimalik oma valikut muuta või vastavalt oma soovidele konfigureerida, kui ainult leitud broneering tühistada ja otsida uus.

Ülevaate tagamiseks valmistati administraatoritele paneel, kus oli võimalik näha järgnevaid andmeid:

- Parklakohad ning nende peakasutajad
- Töötajad ja nende andmed (nimi, email, osakond, parklakoht, sõiduk)
- Logid

Parklakohti oli võimalik otsida ja filtreerida, kuid puudus funktsionaalsus lisada ning hallata. Töötajate tabelist oli näha täiendavat informatsiooni isikuandmete kohta. Kasutajaid sai lisada e-posti alusel. Kuna rakendus töötas isiklikul usaldusel, siis võis tekkida probleeme kasutajate vahel, kui keegi parkis valel kohal. Parandamaks antud olukorda, oli nõutud andmete salvestamine ning tegevuste logimine.

Rakendus oli ehitatud kasutades ASP.NET Core 3.1 [2] raamistikku ning kasutajaliides kasutades Reacti [3] ja TypeScript programmeerimiskeelt [4]. Andmete salvestamiseks kasutati PostgreSQL [5] andmebaasi.

1.2 Tööriistade kirjeldus

Rakendus oli ehitatud REST-API [6] baasil. Selleks kasutati Visual Studio Enterprise 2019 tarkvara. API valmistati ASP.NET Core 3.1 raamistikus ning kirjutati C# keeles. Kasutajaliides tehti kasutades Visual Studio Code tarkvara ning JavaScripti raamistikku React. Lisaks oli TypeScript ja Redux [7]. Disainimiseks kasutati Material-UI [8]. Arenduse käigus kasutati andmete säilitamiseks PostgreSQL andmebaasi, mis asus Dockeril. Rakenduse turvalisuse tagamiseks võeti kasutaja autentimiseks kasutusele JSON Web Token.

Versiooni kontrolliks ning koodi haldamiseks võeti kasutusele veebimajutusteenus GitHub. Analüüsiks ning tagasiside saamiseks otsustasid autorid, et kogu rakendus lisatakse produktsiooni keskkonda. Kasutajaliidese pidevaks integreerimiseks kasutati nii GitHubi kui ka Netlify keskkonda. API jaoks prooviti nii SmarterAsp kui ka Azure pilveteenust. Uute muudatuste tegemisel muutus produktsioonis olev rakendus automaatselt. Koduleht esialgu laeti Netlify enda alamdomeenile. Lisaks suunati leht isiklikule domeenile, kasutades veebimajutuse teenust.

Analüüsimiseks ja diagrammide valmistamiseks kasutati visuaalse modelleerimise ja kujundamise Enterprise Architect ning Visual Studio sisseehitatud tööriistu. Testimisel kirjutati NUnit ühikteste ning kasutati ka API klienti Postman. Lisaks testimisele, oli rakendus Postman abiks ka esialgsele suhtlusele API-ga.

1.3 Tööprotsessi kirjeldus

Rakenduse tegemisel lähtuti SCRUM arendusmetoodikast. Igal nädalal kohtusid tiimiliikmed kahel korral ning üheskoos arutati tööetappe. SCRUM-i ei kasutatud üks-ühele vaid võeti sealt üleüldine ideoloogia nagu lühikesed tähtajad, tihedad kohtumised tiimiliikmetega ja tööde nimekirja omamine. Tööd jaotati peamiselt selle järgi, kus tiimiliikmed ennast tugevalt tundsid. Kuna iseloomult oli tegu RIK-ile tehtud rakenduse edasiarendusega, jaotati tegevusi sarnaselt eelmises projektid tehtud töödega. Igasugune tegevus oli konsensuslik ja arutlev ning tiimiliikmed toetasid teineteise tegevust.

Tööde tegemisel jälgisid tiimiliikmed seda, et ülesannete täitmine ei võtaks ebamõistlikult palju aega. Kui ilmnesid ohumärgid paluti teistelt liikmetelt abi ja lahendati probleemid.

Suuremate funktsionaalsuste teostamisele eelnes tiimiliikmete koosolek, kus üheskoos leiti parim lahendus ning koguti märkmeid töö alustamiseks.

Tööde ülevaate jaoks kasutati Microsoft Plannerit, Toggl-t ja GitHubi. Neist esimesse lisas iga tiimiliige enda ülesandeid ja sellele vastavalt logiti aega ja lisati *commit'*eid.

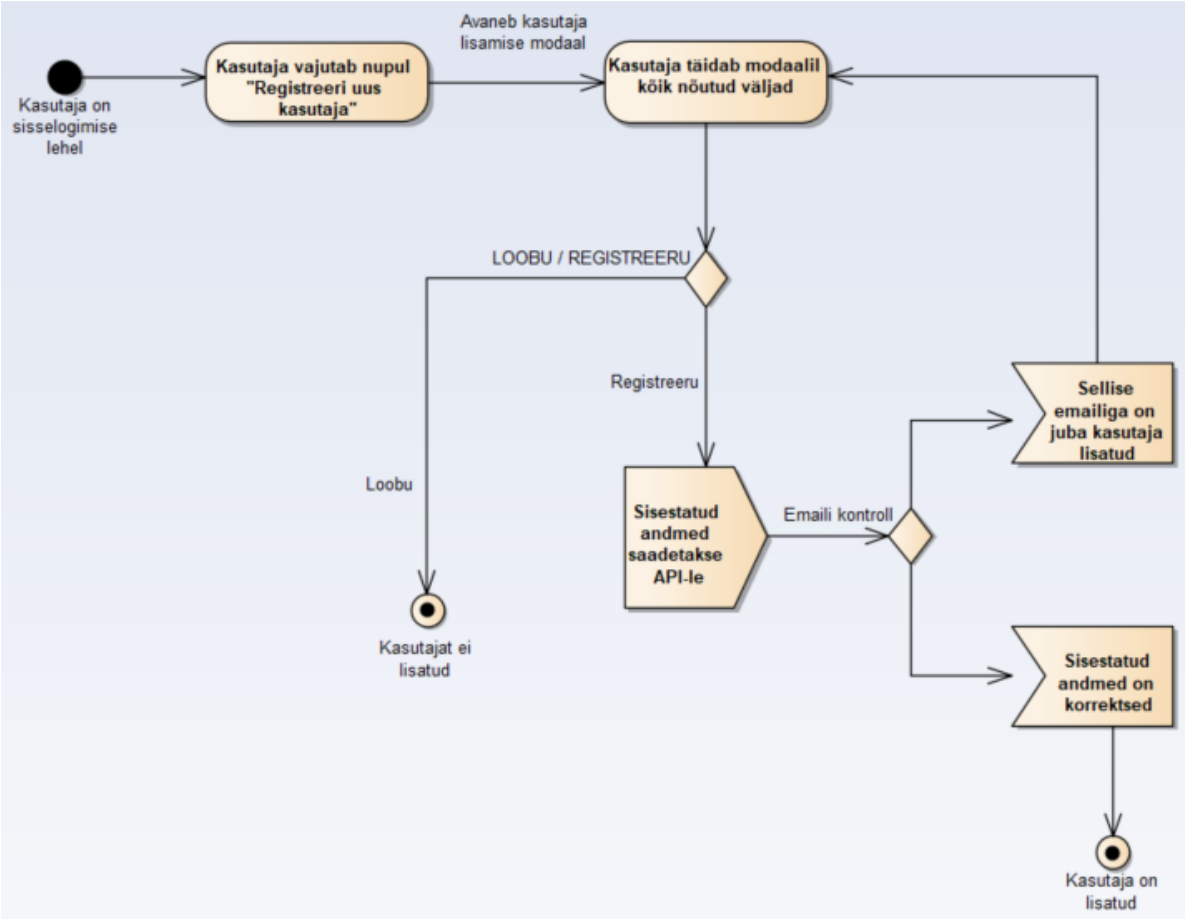
Pühapäeviti toimusid ajaliselt pikemad kohtumised, kus tihti lahendati üheskoos vahepeal ilmnenud probleeme.

2 Töö tulemused

2.1 Rakenduse kasutajad ja asutused

Rakenduse kasutamiseks on vajalik esmalt kasutajakonto loomine (Joonis 2). Seejärel on võimalik lisada asutus, kus omakorda lisatakse asutusele parklakohad ja kasutajad. Iga registreerunud kasutaja näeb enda kuvas teateid, kust on võimalik kinnitada või tagasi lükata kutseid liitumaks teiste asutustega (Joonis 1).

Joonis 1. Kutsete kinnitamise/tühistamise aken.



Joonis 2. Kasutaja registreerimise tegevusdiagramm.

Kasutuslood:

- Kasutajana saan ma lisada asutuse, et seada üles rakenduse kasutamiseks vajalik keskkond.
- Kasutajana saan ma luua endale konto, et saaksin rakendust kasutada.
- Kasutajana saan kasutada telefoninumbri kontrolli, et võimaldada kasutaja tuvastamine telefoninumbri järgi.
- Kasutajana saan kasutada auto numbrimärgi kontrolli, et teha kindlaks parkimiskohal parkiva kasutaja õigus sellel kohal parkida.
- Kasutajana saan ma kinnitada/tühistada endale saadetud kutseid asutustega liitumiseks, et kasutada rakendust vastavalt vajadusele.

Kasutusjuhud:

Kasutusjuht: Asutuse lisamine

Tegutseja: Administraator

Kirjeldus:

1. Kasutaja logib sisse
2. Süsteem kontrollib kasutaja andmed ja suunab asutuste valimise lehele
3. Kasutaja klõpsab nupul „Lisa uus“
4. Avaneb asutuse lisamise akna
5. Kasutaja täidab kõik nõutud väljad
6. Kasutaja klõpsab nupul „Lisa asutus“
7. Süsteem kontrollib kas kõik väljad on korrektselt täidetud
8. Süsteem lisab asutuse andmebaasi

Kasutusjuht: Kasutaja lisamine

Tegutseja: Kasutaja

Kirjeldus:

1. Kasutaja klõpsab nupul „Registreeri uus kasutaja“
2. Süsteem avab kasutaja registreerimise akna
3. Kasutaja täidab kõik kohustuslikud väljad
4. Kasutaja klõpsab nupul „Registreeru“
5. Süsteem kontrollib kõik väljad ja saadab pärigu API-le
6. Süsteem kontrollib emaili
7. Kasutaja lisatakse andmebaasi
8. Süsteem sulgeb kasutaja registreerimise akna ja teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Asutusega liitumise kinnitamine

Tegutseja: Kasutaja

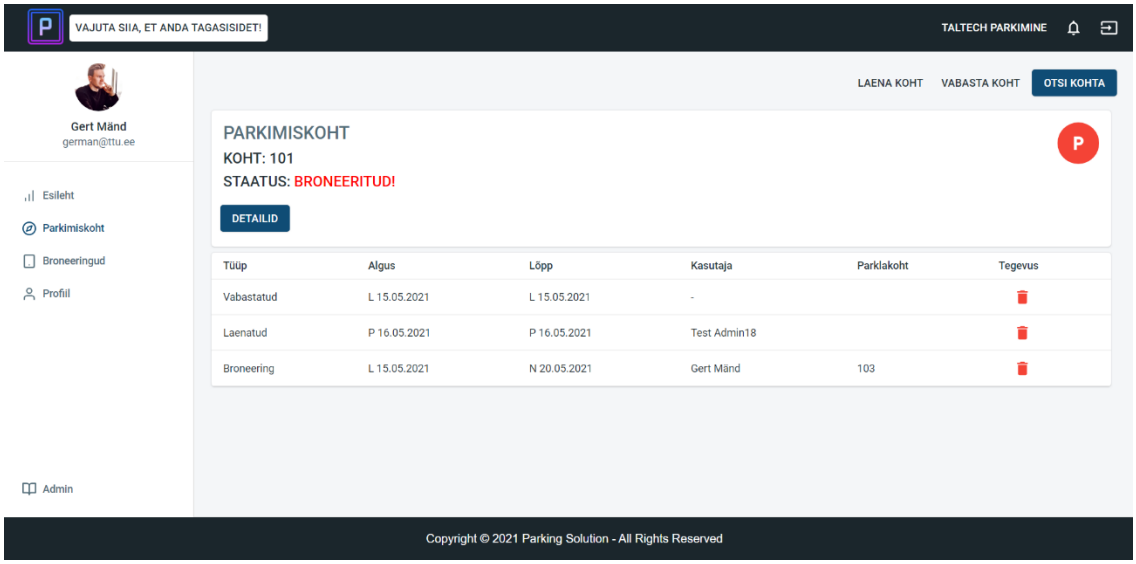
Kirjeldus:

1. Kasutaja logib sisse
2. Kasutaja klõpsab teavituste ikoonil
3. Süsteem kuvab kasutajale kõik asutused, kes soovivad kasutajat enda gruppi lisada
4. Kasutaja kinnitab valitud asutused

5. Süsteem lisab kasutaja vastava asutuse nimekirja
6. Süsteem teavitab kasutajat õnnestunud toimingust

2.2 Parklakoha vabastamise ja reserveerimise loogika

Rakenduse põhirõhk on parklakohtade haldamine (Joonis 3). Eristatakse kahte liiki kasutajaid: peakasutaja, kes on parklakoha omanik, ja (tava)kasutaja, kellel püsiv parklakoht puudub. Parklakohad määratakse kasutajatele administraatorite poolt. Igal peakasutajal on võimalik vabastada oma koht teistele broneerimiseks ning otse loovutada koht määratud kasutajale. Koha vabastamisel ja loovutamisel tuleb määrata periood. Peakasutajal on lisa õiguse puhul võimalik broneerida ajutine koht, tavakasutajal õigust vaja ei lähe. Õiguse eest hoolitseb administraator. Koha omanik võib vabastatud koha tühistada. Broneeritud ajutist kohta on võimalik tühistada ja koht tagastada peakasutajale. Esilehel kuvatakse kasutajale parkimiskoha number ning selle staatus.



Joonis 3. Parkimiskoha lehe vaade.

Kasutuslood:

- Parklakoha omanikuna saan hallata oma parklakohta.
- Parklakoha omanikuna saan kindlaks perioodiks vabastada oma koha teistele broneerimiseks.
- Parklakoha omanikuna saan kindlaks perioodiks loovutada oma koha valitud kolleegile.

- Parklakoha omanikuna saan otsida vabastatud kohti ja broneerida, kui vastav õigus on administraatorite poolt antud.
- Parklakoha omanikuna näen andmeid oma parklakoha kohta, mis perioodiks on vabastatud, mis perioodil on reserveeritud.
- Parklakoha omanikuna saan vabastatud perioodi tühistada.
- Kasutajana saan otsida vabastatud kohti ja broneerida valitud perioodiks parkimiskoht.
- Kasutajana näen oma tänast broneeringut ning parklakohta.
- Kasutajana näen oma tulevasi broneeringuid.
- Kasutajana saan oma broneeringu tühistada ning parklakoha omanikule tagastada.

Kasutusjuhud:

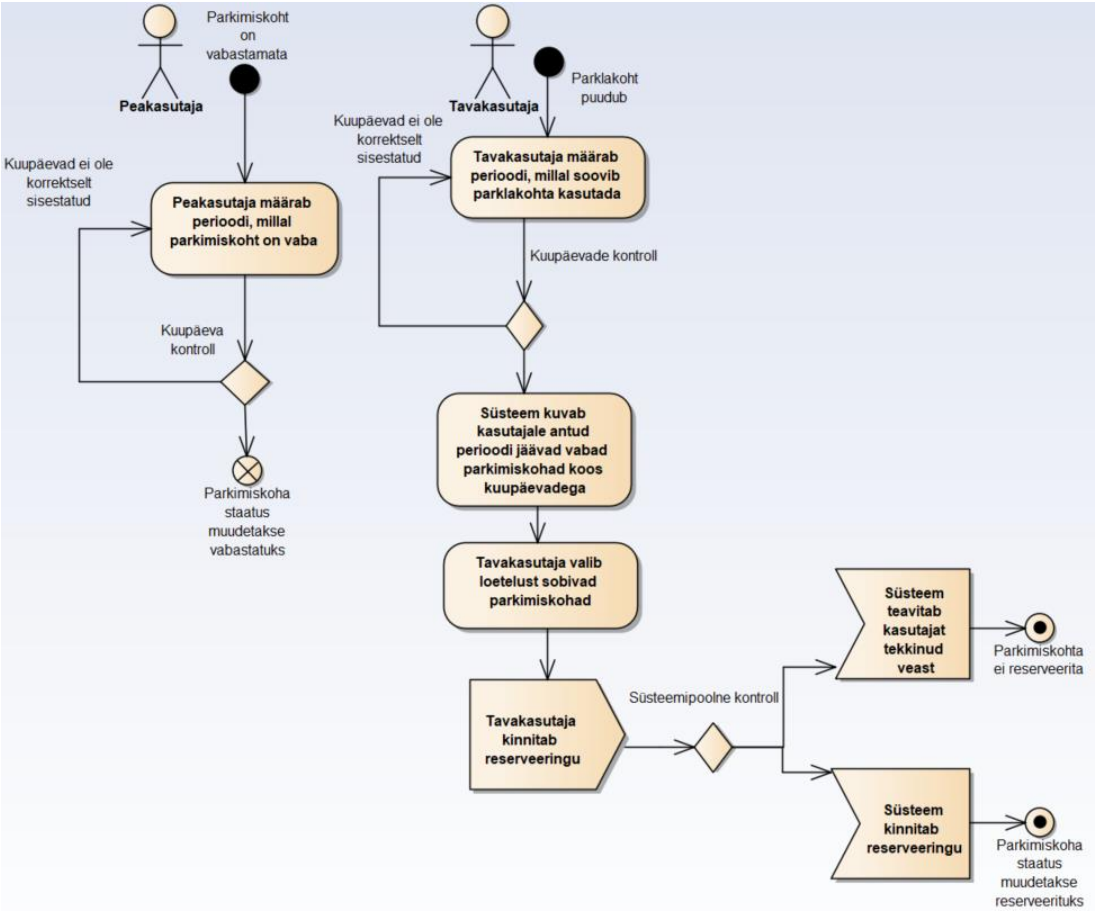
Kasutusjuht: Koha otsimine ning broneerimine (Joonis 4)

Tegutseja: Pea- ja tavakasutaja

Kirjeldus:

1. Kasutaja avab lehe „Parkimiskoht“
2. Kasutaja klõpsab esilehel nupul „OTSI KOHTA“
3. Süsteem avab akna
4. Kasutaja sisestab alguskuupäeva ning lõppkuupäeva
    a. Süsteem kontrollib kuupäevi: lõpp- ei oleks enne alguskuupäeva, alguskuupäev ei oleks varem kui tänane kuupäev, teavitab kasutajat võimalikust veast ning liigutakse tagasi punkti 4
5. Kasutaja klõpsab nupul „Otsi“
    a. Loobumisel klõpsab kasutaja nupul „Tühista“
6. Süsteem otsib kõik vabastatud kohad
    a. Vabastatuid kohti antud perioodiks ei leita, teavitatakse kasutajat
7. Süsteem eemaldab tulemusest otsija enda vabastatud kohad
8. Süsteem eemaldab juba reserveeritud perioodid antud vabastatud kohtadest
9. Süsteem tagastab võimalikud broneeritavad objektid kasutajale
10. Kasutaja valib omale sobivad parklakohad
11. Kasutaja klõpsab nupul „Broneeri“
12. Süsteem kontrollib perioodi, kas koht on veel vaba
13. Süsteem kontrollib, et perioodid ei kattuks

14. Süsteem lisab broneeringu andmebaasi
15. Süsteem teavitab kasutajat eduka broneeringu lisamise eest ning sulgeb akna
16. Süsteem uuendab esilehel tabeli



Joonis 4. Parkimiskohtade vabastamine ning broneerimine.

Kasutusjuht: Ajutise koha tühistamine

Tegutseja: Pea- ja tavakasutaja

Kirjeldus:

1. Kasutaja avab lehe „Parkimiskoht“
2. Leiab antud broneeringu tabelist
3. Kasutaja klõpsab punasel nupul „Tühista broneering“
4. Süsteem kontrollib päringut
5. Süsteem uuendab kustutamise kuupäeva (*DeletionDate)*
6. Süsteem uuendab andmebaasi
7. Süsteem teavitab kasutajat õnnestunud toimingu eest
8. Süsteem eemaldab broneeringu esilehe tabelist

Kasutusjuht: Parkimiskoha vabastamine (Joonis 4)

Tegutseja: Peakasutaja

Kirjeldus:

1. Peakasutaja avab esilehe
2. Peakasutaja klõpsab nupul „Vabasta koht“
3. Süsteem avab „Vabasta koht“ akna
4. Peakasutaja sisestab alg- ja lõppkuupäeva
    a. Süsteem kontrollib kuupäevi: lõpp- ei oleks enne alguskuupäeva, alguskuupäev ei oleks varem kui tänane kuupäev, teavitab kasutajat võimalikust veast ning liigutakse tagasi punkti 4
5. Peakasutaja klõpsab nupul „Vabasta“
    a. Loobumisel klõpsab nupul „Tühista“ ning lõpetab tegevuse
6. Süsteem kontrollib, kas vabastatud koha periood ei kattuks juba reserveeritud perioodiga
    a. Süsteem teavitab vea korral kasutajat, jätkatakse punktist 4
7. Süsteem lisab vabastatud koha andmebaasi
8. Süsteem teavitab peakasutajat edukast toimingust ning sulgeb akna
9. Süsteem uuendab esilehel tabeli

Kasutusjuht: Vabastatud koha tühistamine

Tegutseja: Peakasutaja

Kirjeldus:

1. Kasutaja avab lehe „Parkimiskoht“
2. Leiab antud vabastatud koha tabelist
3. Kasutaja klõpsab punasel nupul „Tühista toiming“
4. Süsteem kontrollib päringut
5. Süsteem uuendab kustutamise kuupäeva (*DeletionDate)*
6. Süsteem uuendab andmebaasi
7. Süsteem teavitab kasutajat õnnestunud toimingu eest
8. Süsteem eemaldab vabastatud koha esilehe tabelist

Kasutusjuht: Parkimiskoha loovutamine teisele kasutajale

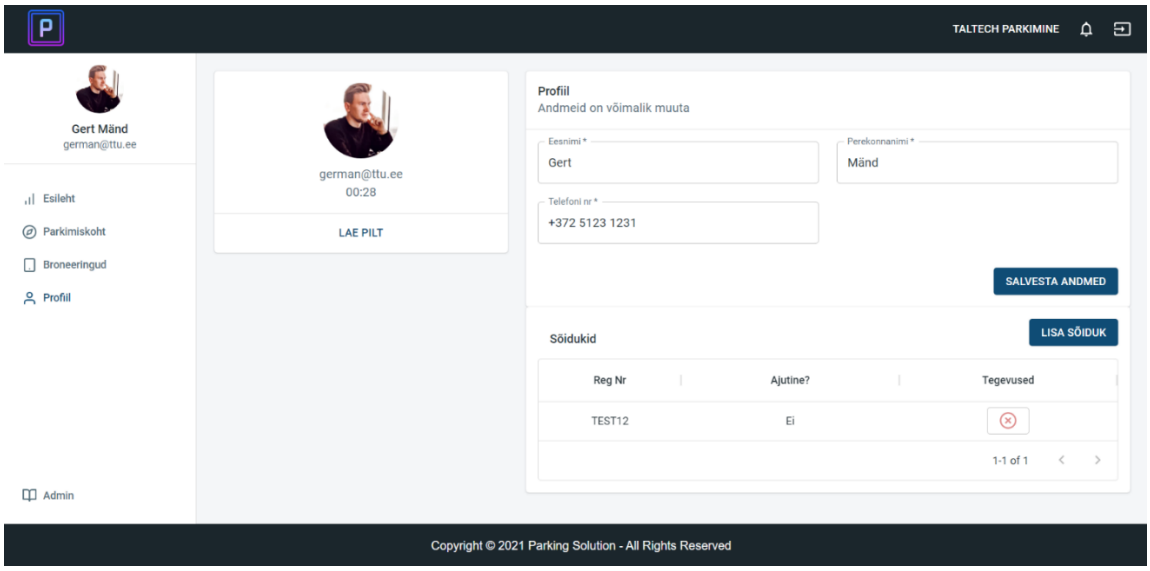
Tegutseja: Peakasutaja

Kirjeldus:

1. Peakasutaja avab lehe „Parkimiskoht“
2. Peakasutaja klõpsab nupul „Laena koht“
3. Süsteem avab „Loovuta koht“ akna
4. Süsteem otsib kõik tavakasutajad ning kuvab neid
5. Peakasutaja valib tavakasutaja, kellele oma koha loovutada soovib
    a. Süsteem kontrollib, kas tavakasutaja on määratud. Vea korral teavitatakse kasutajat ning liigutakse punkti 5
6. Peakasutaja sisestab alg- ja lõppkuupäeva
    a. Süsteem kontrollib kuupäevi: lõpp- ei oleks enne alguskuupäeva, alguskuupäev ei oleks varem kui tänane kuupäev, teavitab kasutajat võimalikust veast. Liigutakse punkti 6
7. Peakasutaja klõpsab nupul „Loovuta“
    a. Loobumisel klõpsab nupul „Tühista“ ning lõpetab tegevuse
8. Süsteem kontrollib, kas valitud periood ei kattuks juba reserveeritud perioodiga
    a. Süsteem teavitab vea korral kasutajat, jätkatakse punktist 6
9. Süsteem lisab broneeringu andmebaasi
10. Süsteem teavitab peakasutajat edukast toimingust ning sulgeb akna
11. Süsteem uuendab esilehel tabeli

2.3 Kasutaja profiil

Kasutaja profiili vaadet on võimalik avada vajutades oma profiilipildile või vasakul olevast menüüribast valides „Profiil“. Profiili vaates kuvatakse kasutajale tema kõige olulisemad andmed, milleks on eesnimi, perekonnanimi ja telefoninumber. Kasutaja andmeid näevad ainult administraatorid, et vajadusel nendega parkimisega seonduvatel küsimustel ühendust võtta ja kasutaja ise. Ainult kasutaja saab oma andmeid muuta (Joonis 5).



Joonis 5. Kasutaja profiili vaade.

Lisaks kontaktandmetele saab kasutaja profiili vaates sisestada andmed oma sõidukite kohta. Sõidukeid on võimalik lisada ja kustutada. Sõidukeid liigitatakse rakenduses ajutisteks ja mitteajutisteks. Juhul, kui kasutaja kasutab rendisõidukeid või muid variante, kus sõiduk on kasutajal ajutiseks kasutamiseks, on sõidukite lisamise vaates mõistlik teha linnuke „Ajutine?“ kastis. Peale numbrimärgi sisestamist tehakse üldine kontroll, kas numbrimärk sisaldab täpitähti või kirjavahemärke ja kui pikk on sisestatud numbrimärk oma karakterite arvult. Detailsemalt kontrolli, kas numbrimärk reaalselt maanteeametis eksisteerib ei tehta.

Kasutuslood:

- Kasutajana saan ma ligi oma andmetele, et näha mis andmeid rakendusele jagatakse.
- Kasutajana saan ma oma andmeid muuta, et hoida rakenduse jaoks enda kohta kõige värskemat infot.
- Kasutajana saan ma enda profiilile lisada sõidukeid, mida kasutan parklas parkimiseks, et vajadusel oleks võimalik numbrimärgi järgi leida vastav kasutaja.
- Kasutajana saan ma enda profiililt sõidukeid kustutada, et hoida oma profiili kohta rakenduse jaoks kõige värskemat infot.

Kasutusjuhud:

Kasutusjuht: Kasutaja andmete muutmine

Tegutseja: Kasutaja

Kirjeldus:

1. Kasutaja klõpsab nupul „Profiil“ või oma profiilipildil
2. Süsteem avab kasutaja profiili vaate ja kuvab olemasolevad andmed kasutajast
3. Kasutaja muudab väljad, mida soovib muuta
4. Kasutaja klõpsab nupul „Salvesta andmed“
5. Süsteem kontrollib muudetud väljasid
6. Süsteem uuendab muudetud väljad andmebaasis
7. Süsteem teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Kasutaja sõidukite lisamine

Tegutseja: Kasutaja

Kirjeldus:

1. Kasutaja klõpsab nupul „Profiil“ või oma profiilipildil
2. Süsteem avab kasutaja profiili vaate ja kuvab olemasolevad andmed kasutajast
3. Kasutaja klõpsab nupul „Lisa sõiduk“
4. Kasutaja sisestab numbrimärgi ja märgib vastavalt, kas tegemist on ajutise sõidukiga või mitte
5. Süsteem kontrollib numbrimärki
6. Süsteem sisestab andmed sõiduki kohta andmebaasi
7. Süsteem loob seose sõiduki ja kasutaja vahel
8. Süsteem teavitab kasutajat õnnestunud toimingust ning kuvab kasutajale uuendatud sõidukite nimekirja

Kasutusjuht: Kasutaja sõidukite kustutamine

Tegutseja: Kasutaja

Kirjeldus:

1. Kasutaja klõpsab nupul „Profiil“ või oma profiilipildil
2. Süsteem avab kasutaja profiili vaate ja kuvab olemasolevad andmed kasutajast
3. Kasutaja klõpsab sõidukite tabelis vastava sõiduki real nupul „Kustuta“

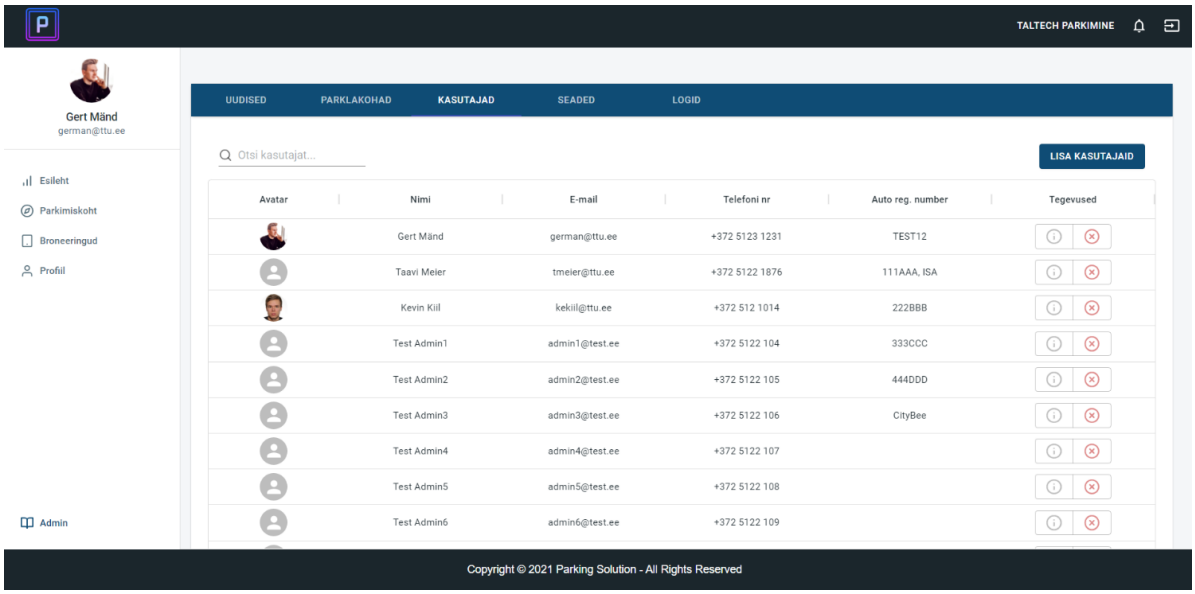
4. Süsteem kuvab kustutamise kinnitamise akna
5. Kasutaja klõpsab nupul „Kustuta“
6. Süsteem kustutab valitud sõiduki andmebaasist
7. Süsteem kustutab kasutaja ja sõiduki vahel olnud seose andmebaasist
8. Süsteem teavitab kasutajat õnnestunud toimingust ning kuvab kasutajale uuendatud sõidukite nimekirja

2.4 Administraatori vaade

Administraatori vaatele on ligipääs ainult parkla halduritel ehk administraatoritel. Administraatoritel on võimalik mainitud vaade avada klõpsates menüüriba allosas olevale “Admin”-ile. Administraatori vaates saab ligipääsu leheküljega seonduvatele uudistele, parklakohtade tabelile, liikmete tabelile, parkla seadetele ja parklaga seotud logidele. Vaikevalikuna avatakse administraatori vaates parklakohtade tabel.

2.4.1 Kasutajate tabel

Kasutajate tabeli kuva eesmärk on kuvada administraatorile kõik parkla kutse aktsepteerinud kasutajad. Kasutaja kohta kuvatakse tabelireal tema avatar, nimi, e-maili aadress, telefoninumber, kasutajale kuuluvad sõidukid ja tegevuste nupud (Joonis 6). Liikmeid on võimalik tabelist eraldi otsida sisestades otsingulahtrisse liikme nime.



Joonis 6. Asutuse kasutajate tabeli vaade.

Kasutajate tabeli “Tegevused” veerus on iga kasutaja kohta võimalus vaadata valitud kasutaja detaile või eemaldada kasutaja asutuse nimekirjast.

Tabelile on rakendatud automaatne sorteerimisvõimekus veergude kohta ja korraga kuvatakse andmed 10 kasutaja kohta. Rohkemate kasutajate korral on võimalik tabeli paremal allosas oleva linnukesega liikuda järgmisele lehele, kus kuvatakse järgmised 10 kasutajat.

Kasutuslood:

- Administraatorina saan ma kasutajate tabelist näha kõiki parkla kasutajaid, et saada ülevaade palju on parklal kasutajaid.
- Administraatorina saan ma saata kasutajatele kutseid asutusega ühinema, et nad saaksid alustada asutusele kuuluvas parklas parkimist.
- Administraatorina saan ma asutuse kasutajate nimekirjast kasutajaid eemaldada, et hoida nimekirja alati värskena ja et piirata võõraste ligipääs parklale.
- Administraatorina saan ma kasutajate tabelist eraldi vaadata üksikisikuliselt kasutaja detaile, et saada vajaminevat infot/kontaktandmeid.

Kasutusjuhud:

Kasutusjuht: Kasutajatele liitumiskutse saatmine asutusega liitumiseks
Tegutsejad: Administraator
Kirjeldus:

1. Administraator klõpsab nupul „Lisa kasutajaid“
2. Süsteem avab kasutajate lisamise akna
3. Administraator sisestab e-mailid, kellele soovib liitumiskutse saata
4. Süsteem kontrollib sisestatud e-maile
5. Süsteem sisestab aktsepteeritud e-mailide kohta kutsed andmebaasi
6. Süsteem teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Kasutaja eemaldamine asutuse nimekirjast
Tegutsejad: Administraator
Kirjeldus:

1. Administraator klõpsab kasutajate tabelis vastava kasutaja real nupul „Eemalda kasutaja nimekirjast“

2. Süsteem kuvab kinnituse akna kasutaja kustutamise kohta
3. Administraator vajutab nupul „Eemalda“
4. Süsteem kustutab kasutaja ja asutuse vahel olnud seose andmebaasist
5. Süsteem teavitab kasutajat õnnestunud toimingust ning kuvab kasutajale uuendatud kasutajate tabeli

2.4.2 Kasutaja detailvaade

Kasutaja detailvaates on administraatoril võimalik näha kasutajaga seonduvaid täpsemaid andmeid. Detailvaade jaguneb kolmeks – kasutaja kontaktandmed, broneeringud ja logid. Detailvaate avamisel kuvatakse vaikimisi kasutaja kontaktandmed.

Kontaktandmete alt näeb administraator lisaks kontaktandmetele ka sõidukite nimekirja. Administraatoril puuduvad õigused andmete muutmiseks ning omab ainult õiguseid andmete vaatamiseks.

Broneeringute valikus kuvatakse parkla haldurile info kasutajate parklakoha ja broneeringute kohta. Sarnaselt kontaktandmetele puudub ka parkimiskoha ülevaate juures administraatoril õigus teha valitud kasutaja eest toiminguid ning omab ainult õiguseid vaatamiseks.

Viimase valikuna kasutaja detailvaates on parkla halduril võimalik vaadata kasutajaga seonduvaid logisid. Kasutaja personaalsete logide alla liigituvad kõik toimingud, mis puudutavad valitud kasutajat. Logide nimekiri on väga mahukas, mistõttu on administraatorile lisatud funktsionaalsus logisid otsida, seda siis aja või logi kirjelduse järgi.

Personaalsete logide alla kuuluvad:

- Valitud kasutaja andmete muutused
- Valitud kasutaja sõidukite lisamised/kustutamised
- Valitud kasutaja parklakoha toimingud (vabastamised/laenamised/broneeringud)
- Valitud kasutaja asutuse kutse kinnitamine/tagasilükkamine
- Valitud kasutajale broneerimisõiguste muudatused
- Valitud kasutaja parklakohale peakasutajaks määramised

Kasutuslood:

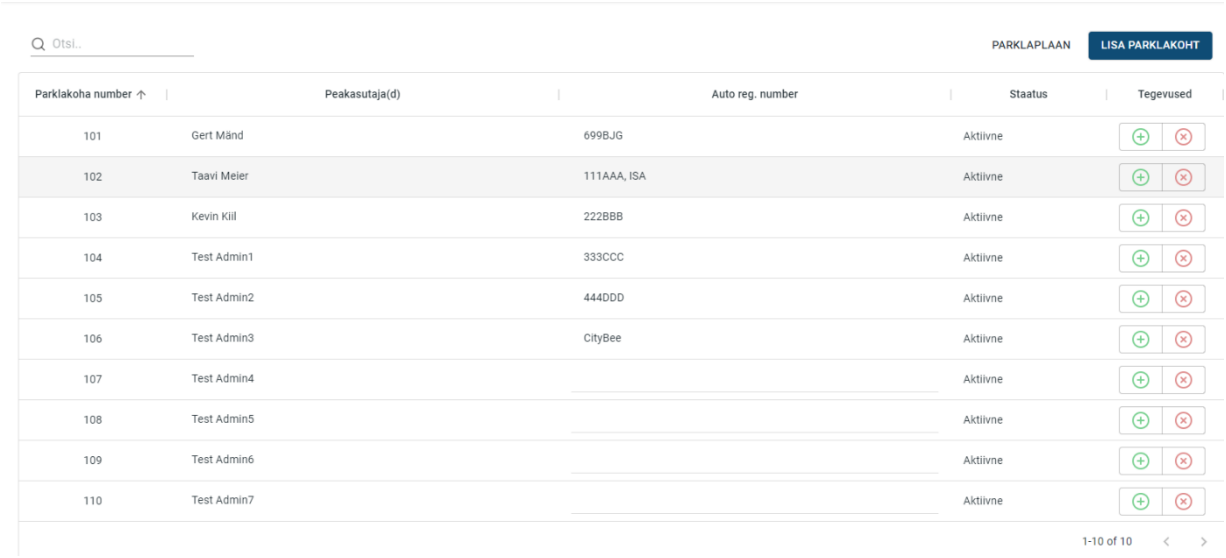
- Administraatorina saan ma kasutaja detaile vaadates näha infot tema parklakoha/broneeringute kohta, et näha millal kasutaja pargib või on parkinud.
- Administraatorina saan ma kasutaja detaile vaadates näha valitud kasutajaga seotud logisid, et teada saada, mida kasutaja on minevikus teinud (millised parkimisega seotud tegevused, milliseid andmeid muutnud).

2.4.3 Parklakohtade tabel

Parklakohtade tabeli puhul omab administraator ülevaadet kõikidest parklakohtadest, määratud peakasutajatest ja nende sõidukitest ning parklakoha staatusest (Joonis 7).

Tabelis on võimalik vaadata/muuta parklaplaani ning lisada/muuta parklakoha peakasutajaid. Kui parklakohale on lisatud peakasutaja, siis tema ei saa endale täiendavalt kohti broneerida. Kui kohale on määratud mitu kasutajat on vajalik broneerimisõiguse taastamine. Sellest tulenevalt on parklakohtade tabelis võimalik modifitseerida broneerimisõigust.

Tabelis on võimalik parklakohti lisada üksikult või exceli tabelist. Kasuteguri suurendamiseks on lisatud otsingu võimalus, mille abil kiirelt leida parklakoha peakasutaja või sellel parkiva sõiduku info.



Joonis 7. Parklakohtade tabeli vaade.

Kasutuslood:

- Administraatorina saan ma lisada parklakohti (üksikult või Exceli failist) ning neid kustutada, et omada ülevaadet parklakohtadest.
- Administraatorina saan ma kasutajale lisada/eemaldada broneerimisõigust, et vältida olukordi, kus peakasutaja kahte kohta hõivaks.
- Administraatorina saan ma otsida parklakohti/kasutajaid/sõidukite numbrimärke, et teha kindlaks parklakoha kasutaja.
- Administraatorina saan ma lisada/muuta parklaplaani, et kasutajad saaksid tutvuda parklaga ning leida oma parkimiskoht.
- Administraatorina saan ma lisada/muuta/kustutada parklakohtade peakasutajaid, et tagada ülevaade ning vajalikud õigused parkla kasutajatele.

Kasutusjuhud:

Kasutusjuht: Parklakoha lisamine üksikult

Tegutseja: Administraator

Kirjeldus:

1. Kasutaja klõpsab nupul „Lisa parklakoht“
2. Süsteem avab parklakoha lisamise akna
3. Administraator sisestab parklakoha numbri
4. Administraator klõpsab nupul „Lisa“
5. Süsteem kontrollib sisestatud numbrit
6. Süsteem lisab parklakoha andmebaasi
7. Süsteem sulgeb parklakoha lisamise akna ja teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Parklakoha lisamine Exceli tabelist

Tegutseja: Administraator

Kirjeldus:

1. Kasutaja klõpsab nupul „Lisa parklakoht“
2. Süsteem avab parklakoha lisamise akna
3. Administraator klõpsab nupul „Faili avamine“
4. Administraator valib .xlsx formaadis faili

5. Süsteem kontrollib sisestatud faili ja teavitab administraatorit, kui fail ei ole korrektne
6. Süsteem kuvab nupu „Lisa parkimiskohad“
7. Kasutaja klõpsab nupul „Lisa parkimiskohad“
8. Süsteem lisab andmebaasi kõik parkimiskohad, mida juba ei eksisteeri
9. Süsteem sulgeb parklakoha lisamise akna ja teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Parklakoha kustutamine

Tegutseja: Administraator

Kirjeldus:

1. Kasutaja klõpsab nupul „Kustuta parklakoht“
2. Süsteem kuvab kinnitamise akna
3. Administraator klõpsab nupul „Kustuta“
4. Süsteem kustutab parklakoha andmebaasist
5. Süsteem kustutab seosed kasutaja ja parklakoha vahel
6. Süsteem sulgeb parklakoha kustutamise kinnitamise akna ja teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Parklakohale peakasutaja lisamine

Tegutseja: Administraator

Kirjeldus:

1. Administraator klõpsab nupul „Lisa peakasutaja“
2. Süsteem kuvab peakasutaja lisamise akna
3. Süsteem kuvab „Valige töötaja“ nimekirjas kasutajad, kellel ei ole määratud parkimiskohta
4. Administraator valib kasutaja
5. Administraator määrab broneerimiseõiguse lisatavale peakasutajale
6. Administraator klõpsab nupul „Lisa peakasutaja“
7. Süsteem kontrollib sisestatud andmed
8. Süsteem lisab parklakoha ja kasutaja seose andmebaasi
9. Süsteem sulgeb parklakoha peakasutaja lisamise akna ja teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Parklakoha peakasutaja eemaldamine

Tegutseja: Administraator

Kirjeldus:

1. Administraator klõpsab nupul „Lisa peakasutaja“
2. Süsteem kuvab peakasutaja lisamise akna
3. Süsteem kuvab aknal kõik kehtivad peakasutajad
4. Administraator klõpsab nupul „Eemalda peakasutaja“
5. Süsteem kustutab parklakoha ja kasutaja seosed andmebaasist
6. Süsteem teavitab kasutajat õnnestunud toimingust

Kasutusjuht: Parklakoha peakasutaja broneerimisõiguse muutmine

Tegutseja: Administraator

Kirjeldus:

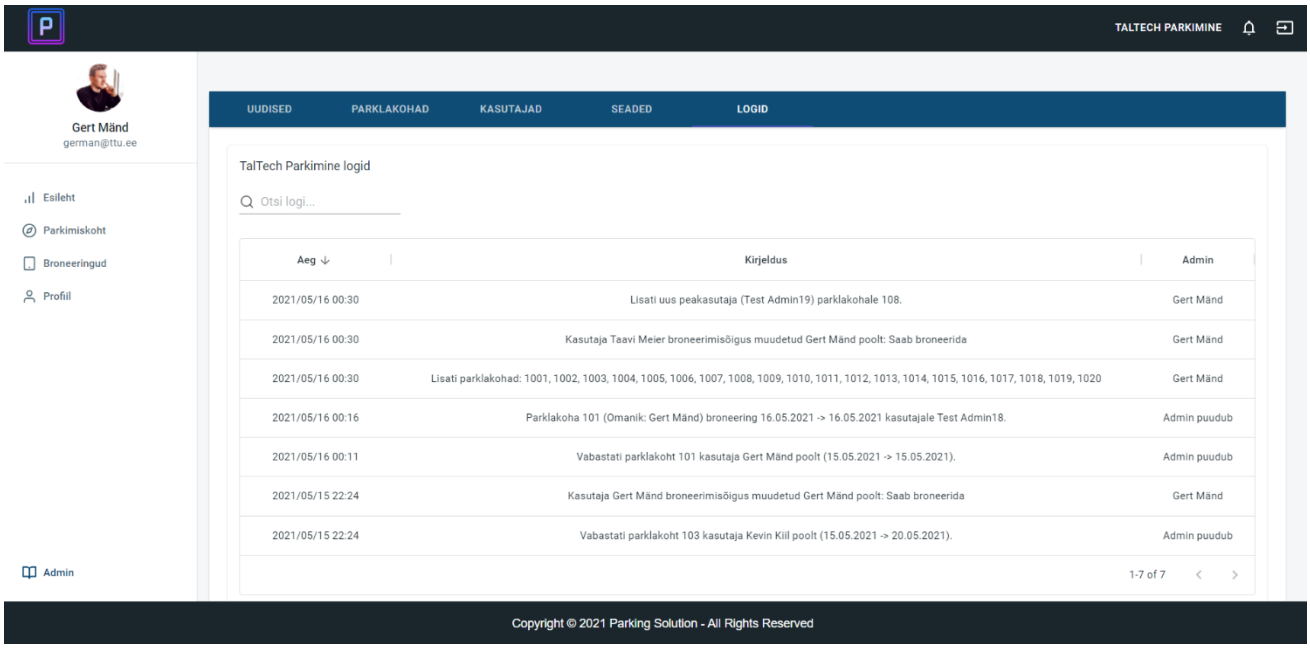
1. Administraator klõpsab nupul „Lisa peakasutaja“
2. Süsteem kuvab peakasutaja lisamise akna
3. Süsteem kuvab aknal kõik kehtivad peakasutajad
4. Administraator täidab/tühjendab *checkbox*'i „Broneerimisõigus“ tulbas peakasutaja nime ees
5. Süsteem muudab andmebaasis kasutaja õiguse broneerida vastavalt tehtud valikule
6. Süsteem teavitab kasutajat õnnestunud toimingust

2.4.4 Asutuse logid

Asutuse logide juures on oluline kuvada parkla haldurile kõik asutust puudutavad toimingud (Joonis 8).

Asutuse logide alla kuuluvad:

- Asutuse parklakohtade peakasutaja määramised
- Asutuse parklakohtade lisamised/kustutamised
- Asutuse kasutaja lisamised/kustutamised
- Asutuse parklakohtade broneeringud/vabastamised/laenamised
- Asutuse kasutajate broneerimisõiguste muudatused



Joonis 8. Asutuse logid.

Sarnaselt kasutaja detailvaate logidele on administraatoril võimalik ka asutuse logisid otsida logide aja või kirjelduse järgi. Asutuse logide puhul kuvatakse ka veerus „Admin“ administraatori ees- ja perekonnanimi, kui tegemist oli toiminguga, mida saab teha ainult administraator (näiteks parklakohtade lisamised/kustutamised, peakasutajate määramised).

Kasutuslood:

- Administraatorina saan ma asutuse logide alt näha kõiki asutuse parklaga seotud toiminguid, et saada ülevaade parklas toimuvast ja administraatorite tegevustest.

2.5 Arhitektuur ja disain

Projekti raskustase on kõrge ning selleks jälgiti autorite poolt kehtestatuid reegleid. Rakendus ehk REST-API oli kirjutatud C# keeles ning kasutati mitmekihilist arhitektuuri (Lisa 2). Malliks kasutati Boilerplate [6]. *Back-end* jaguneb mitmeks osaks. Päringud saadetakse esialgu *controller'*ile (Tabel 1) ehk kontrollerile, mille ülesandeks on vastavalt argumentidele genereerida vastus ning tagastada see kasutajale, kes päringu alustas. Loogika ning vastuse genereerimine käib läbi *service*'te (Tabel 2) ehk teenuste. Seal toimub kogu suhtlus andmebaasiga ning vajalike andmete töötlemine. Teenus kutsutakse välja otse kontrollerist. Päringute vähendamiseks kontrollerile võeti kasutusele *Data*

transfer object (DTO). Tegemist on andmeedastusobjektiga, mis pakitakse mitmest erinevast objektist kokku ning tagastatakse *front-end'*i. Automaatseks objektide kokku pakkimiseks kasutatakse andmete korrastamise teeki *AutoMapper*'it.

Arenduse käigus initsialiseeriti andmebaas näidisandmetega (Lisa 3). Mudelite valmistamisel kasutati korduvate muutujate juures klasside pärimist. Kõikidel objektidel on antud unikaalne identifikaatori, mis on päritud klassist nimega *UniqueEntityData*. Alg- ning lõppkuupäev on päritud klassist nimega *ParkingDateEntityData*. Lisaks perioodile, olid objektidel määratud *Created* ehk loomise kuupäev, *Updated* ehk uuendamise kuupäev ning *DeletionDate* ehk kustutamise kuupäev, mida päritakse klassist *DateEntityData*

Tabel 1. Kontrollerid.

EnterprisesController	Asutused, asutuste kasutajad, parklakohad, broneeringud
AccountsController	Kasutaja andmed, autentimine ning registreerimine

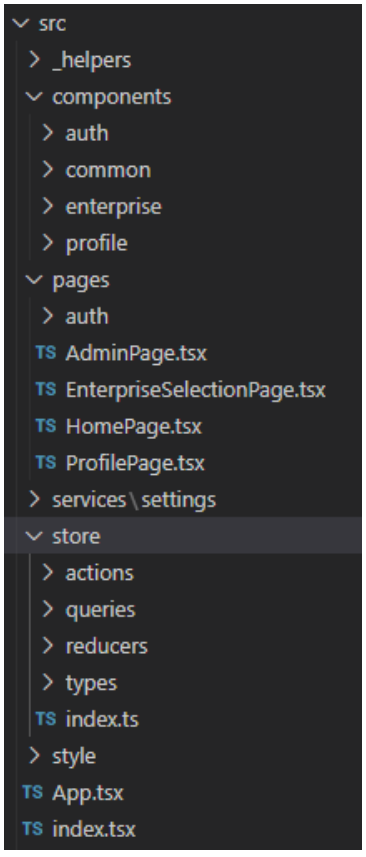
Tabel 2. *Service'*d.

AccountService	Kasutaja andmed, autentimine, registreerimine, parooli lähtestamine, profiili muutmine, numbrimärgi lisamine
EmailService	E-posti saatmine
EnterpriseService	Asutused, kasutaja asutused, asutuste kasutajad, asutuse kutsed, kasutajate eemaldamine asutustest, telefoninumbri valideerimine, administraatorite kuvamine
LogService	Asutuste logid, kasutaja logid, logide sisestamine
ParkingSpotService	Parklakohad, staatus, broneeringud, peakasutajad, tühistamised

Front-end poolel on komponendid ning failid sorteeritud eraldi kaustadesse (Joonis 9). Lisafunktsionaalsus, mida on võimalik korduvkasutada, on jaotatud eraldi failidesse, mis asuvad kaustas „_helpers“. Andmete ajutiseks salvestamiseks brauseris, kasutatakse Reduxit, mis on mõeldud kasutajaliidese oleku haldamiseks. Antud failid asuvad kaustas „store“. Et andmeid kuvada kasutajaliideses, tehakse päringud *back-end'*i, mis tagastab

vajaliku informatsiooni struktureeritud kujutl. Päringud on jaotatud eraldi failidesse ning kausta „store/queries“. Komponentide kuvamine lehel algas lehe-failist kaustas „Pages“.

Kasutajaliidese disainimisel kasutati Google loodud disainikeelt Material Design / Material UI [8]. Arendajatele disaini nõudeid ei esitatud ning lehe üldine paigutus on autorite poolt seatud. Lehe malliks kasutati „material dashboard kit“, mille autoriks on Devias-io [9]. Antud disain kasutab MIT litsentsi, millel on luba piiranguteta kasutada, kopeerida, muuta ja/või müüa, kui on täidetud eeldus, et sama autoriõiguse teave tingimuste kohta tuleb lisada uuesti publitseeritud tarkvarale. Lehel on üks värvivalik ning see on ära defineeritud eraldi teema failis. Määratud on taustavärv, põhivärv, sekundaarne värv ning tekstivärv. Veebidisain on dünaamiline ehk *responsive*, ehk lehte on võimalik kasutada nii arvutis, tahvelarvutis kui ka telefonis. Selleks kasutati Material UI loodud Grid komponenti, mis reageerib lehe suuruse ja orientatsioonile.



Joonis 9. Kasutajaliidese failide struktuur.

Kasutajakogemuse parandamiseks kasutatakse brauseris küpsiseid. Sisse logides valitakse asutus, mis salvestatakse küpsistesse ning järgmisel sisse logimisel tehakse kasutaja eest seda automaatselt. Lehele saabumisel kuvatakse kasutajale laadimise ikooni,

vältimaks ootamatuid andmete muudatusi. Niipea kui päringud on edukalt täidetud, uuendatakse kuvand vastavalt andmetele.

2.6 Kood

Front-end'is jagati lehed eraldi komponentideks ning arenduse käigus jälgiti, et pikkus ei ületaks rohkem kui 100 koodirida. [10, p. 174] Lisaks koodiridadele jälgiti, et komponentide vaheline suhtlus ei tekitaks lehel oleku probleeme. Programmi Visual Studio Code on sisseehitatud koodi ülevaatus, mis hoiatab kasutamata muutujate ning muude probleemide korral, mis lehte aeglustab. Samuti arenduskäigus tagastab brauser ootamatud vead, mis vajavad kohest parandamist. Tabeli veergudele tuleb määrata vastav identifikaator ning kui tabelis leidub sama id-ga ridu, on see potentsiaalne veakoht.

Päringutest on tehtud eraldi meetodid, mida hoitakse eraldi *queries* failides. Isiklike andmete kätte saamiseks on vaja autentida, selleks genereeritakse sisse logimisel *token* ehk kood, mis käib iga päringuga kaasas. Antud *token*'it hoitakse kasutaja brauseris küpsistes. Iga päringu tegemisel tuleb määrata selle meetod:

- HTTP GET meetod – Andmete pärimiseks, parameetrid on lisatud aadressile
- HTTP POST meetod – Andmete lisamiseks, andmed antakse kaasa eraldi objektina (Joonis 10)
- HTTP PUT – Andmete muutmiseks
- HTTP DELETE – Andmete kustutamiseks

Lisaks meetodile antakse *back-end*'ile kaasa genereeritud *token,* mis tuvastab isiku ning töötleb vastavalt kriteeriumitele andmed. Selleks, et igale päringumeetodile ei antaks HTTP meetodit ning *token'*it eraldi, on loodud *fetch-wrapper* fail. Päringute tegemiseks kasutatakse brauseri sisseehitatud funktsiooni *window.fetch*.

```
export const post = (url: any, body: any) => {
    const requestOptions = {
        method: 'POST',
        headers: { 'Content-
Type': 'application/json', 'Accept': 'application/json', ...authHeader() } as any
,
        credentials: 'include' as any,
        body: JSON.stringify(body)
    };
    return fetch(url, requestOptions).then(handleResponse);
}
```

Joonis 10. HTTP Post meetodi näide.

Lehel leidus korduvkasutatavaid elemente, millest loodi eraldi komponendid. Üks nendest on kuupäeva sisestus (Joonis 11). Sisendiks on antud *label* ehk silt, mis teksti kuvatakse enne lahtri täitmist ja *date* ehk valitud kuupäev. Lahtri muutmisel käivitatakse funktsioon *handleDateChange*, mis teavitab vanemkomponenti muudatusest ning uuendab kuvandi. Antud komponenti kasutati topelt nii koha vabastamisel, loovutamisel kui ka otsimisel.

```
<MuiPickersUtilsProvider utils={DateFnsUtils} locale={estLocale}>
        <KeyboardDatePicker
          disableToolbar
          disablePast
          variant="inline"
          format="dd.MM.yyyy"
          margin="normal"
          id={ "select-date-" + label}
          label={label}
          value={date}
          onChange={handleDateChange}
          KeyboardButtonProps={{
            'aria-label': 'muuda',
          }}
          autoOk={true}
          helperText={''}
        />
    </MuiPickersUtilsProvider>
```

Joonis 11. Kuupäeva korduvkasutatav *component*.

Back-end'is toimub üldine loogika *service* failides (Joonis 12) [11]. Meetodid olid ära jaotatud vastavalt, avalikud meetodid üleval, kinnised all. Samuti kasutati abistavaid meetodeid, mis tavaliselt sisaldas korduvkasutatavaid koodiridu või päringuid. Abistavad meetodid algasid väikse tähega.

```
public interface IAccountService
{
        AuthenticateResponse Authenticate(AuthenticateRequest model, string ipAddress);
        AuthenticateResponse RefreshToken(string token, string ipAddress);
        void RevokeToken(string token, string ipAddress);
        void Register(RegisterRequest model, string origin);
        void VerifyEmail(string token);
        void ForgotPassword(ForgotPasswordRequest model, string origin);
        void ValidateResetToken(ValidateResetTokenRequest model);
        void ResetPassword(ResetPasswordRequest model);
        IList<CarResponse> GetCarsByAccountId(int id);
        IEnumerable<AccountResponse> GetAll();
        AccountResponse GetById(int id);
        AccountResponse Create(CreateRequest model);
        AccountResponse Update(int id, UpdateRequest model);
        void Delete(int id);
        void AddCar(int id,  AddCarRequest request);
        void DeleteCar(int id, CarResponse request);
        void EditAccount(int id, EditAccountRequest request);
}
```

Joonis 12. *AccountService interface* näidis.

Kuupäevade kontrollimiseks tehti eraldi abistav fail. Andmeid andmebaasist ei kustutata ning kasutatakse objektidel DeletionDate muutujat. Kontrolliks, kas antud objekt on veel aktiivne, tehti abistav meetod, mis eemaldab listist objekti, juhul kui see on tühistatud (Joonis 13).

```
public static List<T> RemoveDeletedDates(List<T> list)
{
       for (int i = list.Count - 1; i >= 0; i--)
       {
              if (list[i].DeletionDate != null)
              {
                     if (list[i].DeletionDate.Value < DateTime.Today.Date)
                     {
                            list.RemoveAll(x => x.Id == list[i].Id);
                     }
              }
       }
       return list;
}
```

Joonis 13. Listist tühistatud objektide eemaldamise näidiskood.

Rakendusel on kasutusel *middleware*, mis on vahelüliks ning ülesandeks päringute haldamine. Esimesena kasutatakse JwtMiddleware, millega hallatakse kasutaja *token*'it. *Token*'i abil tuvastatakse päringu saatja ning *middleware* töö on kontrollida selle õigsust. Pärast kontrollimist salvestatakse kasutaja andmed mälusse ning kontrolleritest on võimalik päringu saatja andmed kätte saada. Vea korral tagastatakse *front-end*'i *HTTP*

401 – Unauthorized response. Teisena kasutatakse ErrorHandlerMiddleware. *Service*'te vigade korral *throw*'itakse *exception*, mis hallatakse läbi *middleware* tagasi kuvandisse (Joonis 14). Eeldefineeritud on kaks veakoodi:

- HTTP Bad Request – AppException
- HTTP Not Found – KeyNotFoundException

```
throw new KeyNotFoundException("message");
```

Joonis 14. Veateate saatmine kuvandi.

2.7 Testid

Rakendus on testitud kolmel erineval viisil. API päringutele on tehtud testid Postmani kasutades. NUnit raamistiku abil on API-le loodud ühiktestid. Kasutajaliidese ja API poolele on lisatud ka sisestusväljade kontrollid ning kasutajate autoriseerimise kontroll.

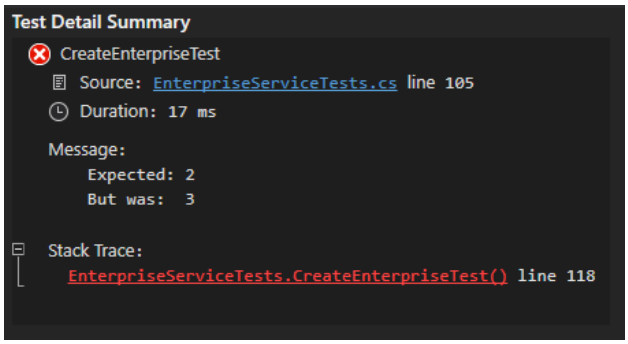
Autorid käisid läbi peamised kasutusjuhud ning kirjutasid vajalikud testid ja kontrollid tagamaks rakenduse korrektne kasutamine kasutajate poolt.

NUnit testid tehti eraldi projekti ning sorteeriti eraldi kaustadesse. Testide läbiviimiseks initsialiseeriti andmebaas näidisandmetega. Testi loomiseks tuleb klassi sisse luua meetod ning see määrata testiks (Joonis 15). Selleks lisati meetodi peale „[Test]“. Esialgu luuakse objekt ning täidetakse muutujad vastavate andmetega. Järgmisena kutsutakse välja meetod, mida soovitakse testida ning parameetrina antakse kaasa tehtud objekt. Lõpus kontrollitakse *Assert* lausete abil, kas tagastatud vastus on korrektne.

```
[Test]
public void CreateEnterpriseTest()
{
        var enterpriseCountBeforeAdd = eService.GetAll().Count();
        EnterpriseCreateRequest request = new EnterpriseCreateRequest
        {
                AcceptTerms = true,
                UserId = 1,
                Description = "kortermaja",
                Name = "Vilde tee 100",
                type = EnterpriseType.Ühistu
        };
        eService.Create(request);
        var enterpriseCountAfterAdd = eService.GetAll().Count();
        Assert.AreEqual(enterpriseCountAfterAdd, enterpriseCountBeforeAdd+1);
}
```

Joonis 15. Näidis testi kood.

Testprotseduuri käivitades jooksutakse kõik testid järjest. Vea korral tagastatakse info kasutajale, mis testis viga oli, oodatud ning meetodist saadetud väärtus. (Joonis 16)



Joonis 16. Vigase testi näide.

3 Analüüs ja järeldused

3.1 Tehnilise teostuse põhjendus

3.1.1 Nõuded

Nõuete puhul peeti oluliseks keskenduda rakenduse turvalisusele ja võimalikult iseseisvale ja kasutajakesksele toimimisele. Kasutaja erinevate toimingute juures tuleb oluliseks pidada, et kasutajale tuleb anda võimalus otsustamiseks ja et rakendus ei otsustaks kasutaja eest. Näiteks parkla kutsete puhul tuleb anda kutsesaajale võimalus kas see vastu võtta või sellest keelduda. Rakenduse peamised nõuded funktsionaalsuse osas olid kasutaja registreerimine, parkla registreerimine, kasutajate lisamine/eemaldamine ja parkla haldamine, kasutajate vahelised parklakoha broneeringud/vabastamised ja tegevuste logid.

Nagu ka varasemalt mainitud, siis antud projekti eelkäijaks oli parklakohtade rakendus RIK-ile, siis sellest tulenevalt kandusid üle ka asutuse parkimist puudutavad nõuded. Uute nõuetena funktsionaalsusele lisandusid asutuse registreerimine ja üldiselt asutuste andmete eristamine. Asutuse andmete eristamisel on oluline eristada asutuse siseseid parkimisi, parkla haldustoiminguid ja logisid.

3.1.2 Arhitektuur ja alternatiivid

Arhitektuuri puhul peeti oluliseks selle keerulisust ning rakenduse täiendamist. RIK-i projekti näitel oli näha, et kui rakenduse arendusel puuduvad kindlad reeglid ja struktuur, on uue funktsionaalsuse lisamine kordades keerulisem. Hiljem koodi täielik refaktoreerimine on aja kaotus. Suurte puuduste tõttu võtsid autorid vastu otsuse alustada antud projekti nullist. Projekti arenduse käigus jälgiti, et kui meeskonnaga ühineks uus arendaja, oleks arhitektuur, nimetused ja vorming koheselt arusaadav. *Back-end'*is jäljendatakse niinimetatud mitmekihilist arhitektuuri, aga kogu rakendus asetses ühes projektis. Kihid jaotati ära erinevatesse kaustadesse. Lisaks RIK-i projektile võeti kasutusele kontrollerite ja andmebaasi vahele *service*'d, mis kujunes loogika kihiks.

Projekti analüüsi tehes arvestati repositooriumitega, kuid ajalise kulu tõttu otsustasid autorid, et seda kasutusele ei võeta.

Rakenduse üheks miinuseks on *back-end*'i ja *front-end*'i omavaheline suhtlus. Andmed uuendatakse kuvandis juhul, kui kasutaja on seda küsinud. Selleks tehakse päring *back-end'i* ning andmed tagastatakse struktureeritul kujul tagasi. Olukorras, kui rakendust kasutab kaks või rohkem kasutajat samaaegselt, võib tekkida olukordi, kus ühe kasutaja tegevused ei jõua teiseni ning tekitab arusaamatusi. Näiteks parklakoha broneerimisel. Kasutaja A ning kasutaja B soovivad broneerida samaks perioodiks endale parkimiskohta. Mõlemale kuvatakse sama vabastatud parkimiskoht, kuid koha saab endale kiirem ning teisele kohta ei broneerita. Enne igat sooritust, kontrollitakse topelt andmed üle ning seejärel uuendatakse informatsioon andmebaasis.

Alternatiiviks leiti erinevaid tarkvarakogusid. ASP.NET-iga on võimalik seda teha, kasutades SignalR-i. Tegemist on Microsofti enda raamistikuga, mis võimaldab serverikoodil saata asünkroonseid märguandeid kliendipoolsetele veebirakendustele. Sarnaselt toimib Google arendatud platvorm Firebase. Suureks plussiks on telefonirakenduste integratsioon.

3.1.3 Disain

Rakenduse puhul on oluline turvalisus ja efektiivne ressursside kasutamine. Rakenduse kvaliteedi seisukohast on oluline kasutada üldlevinud disainimustreid, mis omakorda tõstavad rakenduse käideldavust. Sellest tulenevalt on kasutusel *service layer* [12, p. 133] ja *data mapper* [12, p. 165], *identity field* [12, p. 216], *foreign key mapping* [12, p. 236], *data transfer object* [12, p. 401]. Eesmärk oli vähendada koodisõltuvust ja koodiduplikatsiooni. Kasutajaliidese puhul kasutati võimalikult abstraktseid komponente, mida saaks korduvkasutada.

API poolel ei käi kontrollerid otse baasist informatsiooni küsimas – see tegevus käib läbi *service* privaatsete meetodite. Ka API poolel on viidud võimalikult palju koodi abstraktsele tasemele ning *service* loomisel kasutati *interface*'i.

3.1.4 Koodi kirjutamise reeglid

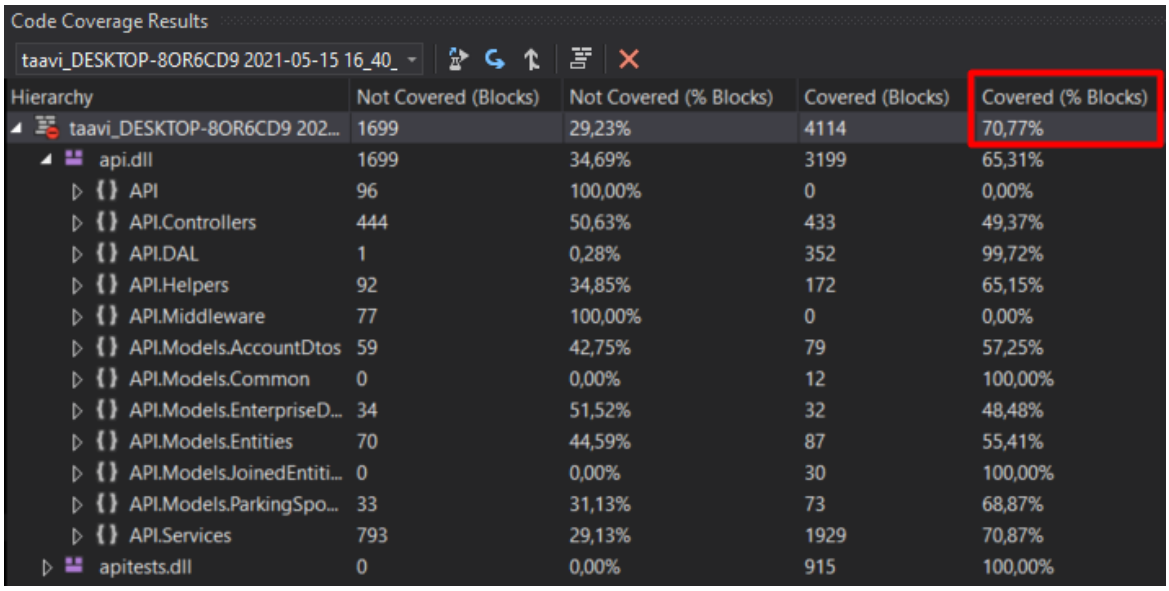
Rakenduse arendustöö jagunes kaheks, *front-end*'iks ja *back-end*'iks. *Front-end* oli kirjutatud JavaScript ja TypeScript keeles, *back-end* C# keeles. Projektist moodustas JavaScripti kood 31,9%, TypeScript 33,9% ja C# 33,9%.

Koodi kirjutamise reeglitena järgiti peamiselt *Clean Code* põhimõtteid. Tähendusrikkad nimed klassidele, meetoditele, komponentidele, muutujatele. Funktsioonid ja meetodid nimetatud tegusõnadena. Klassid, muutujad, komponendid nimisõnadena. Arendustöö käigus toimus pidev refaktoreerimine. [13, pp. 25,76] Näiteks *front-end*'i arenduses jälgiti, et Reacti komponentides oleks kasutusel ainult vajaminev ja et poleks üleliigseid importimisi. Samuti peeti oluliseks klasside, meetodite, komponentide pikkust, et need ei ületaks 100 koodirea piiri. [10, p. 174] Keerukamad komponendid jagati osadeks ja uute lahenduste juures jälgiti, et ei esineks koodikorduseid. Võimalusel kasutati varasemalt loodud lahendusi. [13, p. 48]

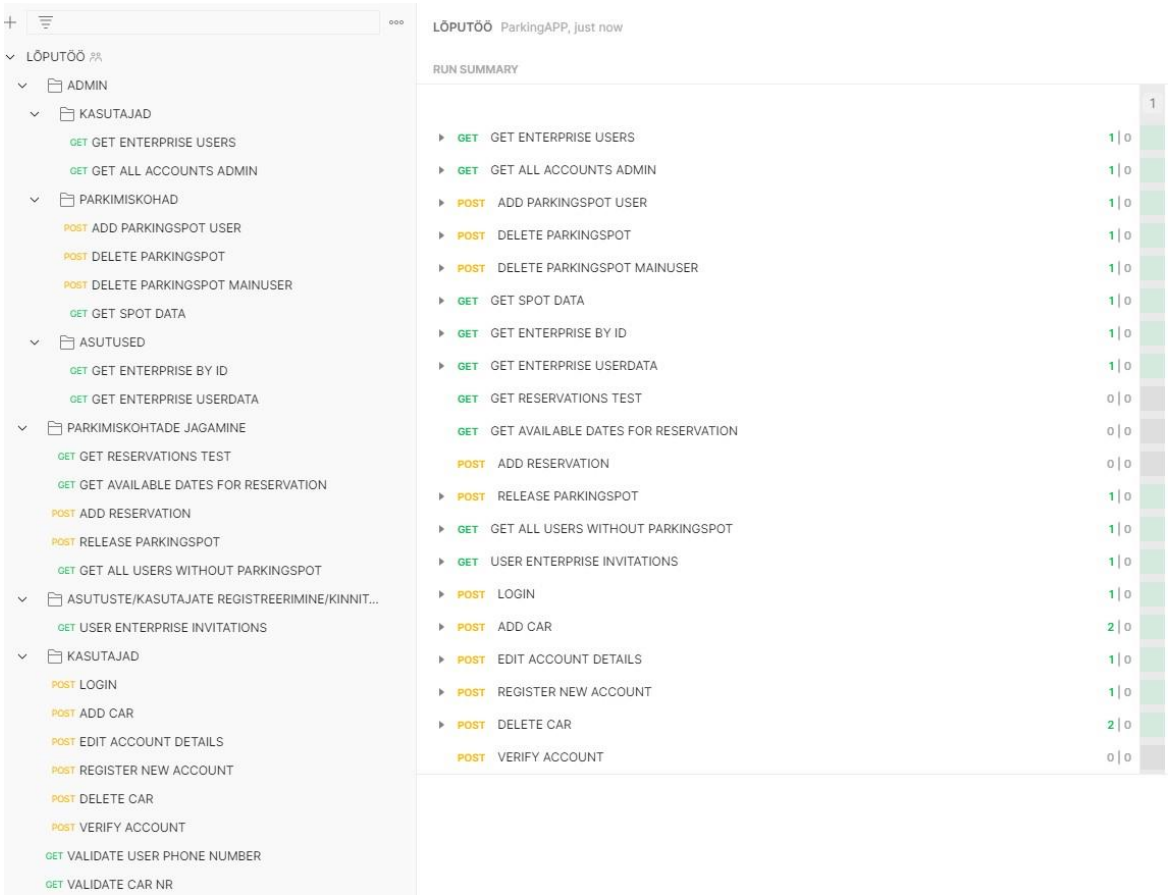
Arendustöö juures rõhutati *commit*'ite kommentaaride olulisust. Üleslaetav töö peab olema kommenteeritud saavutamaks tiimikaaslaste parem mõistmine tehtud muudatustest ja et oleks võimalik analüüsida tööprotsessi kulgu. Arendajate seas jälgiti tava, et muudatuste üleslaadimisel rakendus oleks käivitatav ja et ei esineks tõrkeid. Tõrgete ilmnemisel on võimalik GitHubist järgi vaadata, millised komponendid *commit*'iga seoses muudeti ning siinkohal väljendub *commit*'i kommentaari olulisus – kommentaar, mis annab lühidalt ja detailselt info muudatuse kohta, aitab saavutada tunduvalt kiirema vigade leidmise.

3.1.5 Testide metoodika ja kaetus

Rakenduse testimisel kasutati Postmani API-sse päringute saatmiseks. Postman oli oluliseks abivahendiks rakenduse ehitamisel, kuna enamustele päringutele teostati enne kasutajaliidese ehitamist testid Postmaniga (Joonis 18). Seeläbi tehti kindlaks API töökindlus. API on testitud 70% ulatuses ühiktestidega nende meetodite ulatuses, mida oli otstarbekas ja võimalik antud tingimustest testida (Joonis 17). Testimisel kasutati NUnit raamistikku, kuna tegu on .NET raamistikule mõeldud raamistikuga.



Joonis 17. API ühiktestidega kaetus.

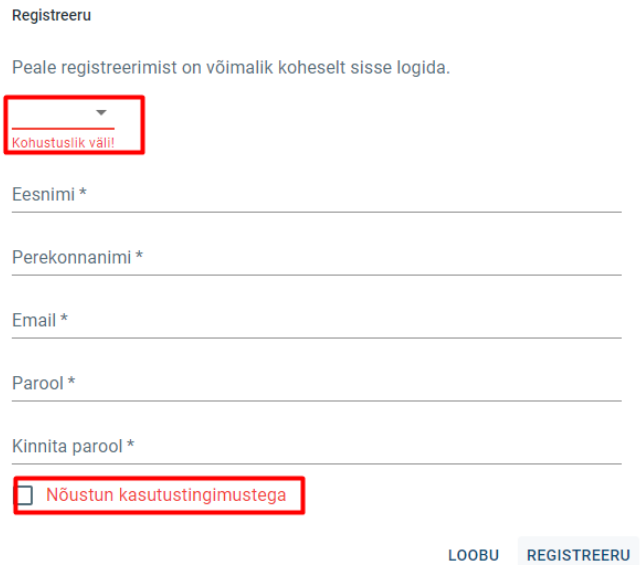


Joonis 18. Postmani testid.

Kasutajaliidese ja API poolne kontroll on lisatud väljade sisestamisele, korrektsete formaatide kasutamisele, aga ka kasutajate valideerimisele. Rakendust saavad kasutada

mitmed asutused ning seetõttu on oluline tagada andmete turvalisus samaaegselt kasutusmugavusega.

Kasutajaid teavitatakse koheselt igasugustest anomaaliatest ja valesisestusest eesmärgiga vältida rakenduse tõrkeid (Joonis 19).



Joonis 19. Kasutaja registreerumise aken. Kohustuslikud väljad.

3.2 Projekti taust ning põhjendus

Parkimine on olnud probleemiks juba aastaid ja tänasel päeval on enamus lahendusi seotud parklate maksustamise, tõkkepuude paigaldamise või mõne muu meetmega, millega vältida parklate väär/õigustamata kasutamist.

Autorite eesmärk on muuta olemasolevate „kinniste“ parklate kasutamine asutuse siseringis efektiivsemaks ja tagada seeläbi töötajatele, elanikele ja muudele kasutajatele täiendav mugavus samal ajal vähendades parklate administreerimise eest vastutavate isikute koormust probleemide lahendamisel.

Teisisõnu on need lahendused mõeldud eelkõige automatiseerimiseks ja korra tagamiseks. Analüüsi tulemusena jõuti järeldusele, et lahendust, kus asutus saaks mugavalt hoida kasutajate andmeid ning kohtade jagamine oleks võimalikult lihtne, ei ole. Autoritele teadaolevalt avalikku rakendust parklakohtade jagamiseks ei ole.

Esimene kokkupuude rakenduse vajalikkusega oli RIK-is, kus RIK-ist lähtuvalt ehitati rakendus, kus töötajad saaksid parklakohti jagada ning broneerida ja administraatorid mugavalt kohtade ja kasutajate infot vaadata ning muuta. Selle töö põhjal teostati käesoleva lõputöö teemaks olnud projekt.

Käesolev projekt on teostuselt ja arhitektuurilt täiesti erinev eelkäiast. Seda peamiselt turvalisuse ja kvaliteedi osas. Projekti raames peetud analüütilised vestlused potentsiaalsete kasutajatega on kinnitanud rakenduse vajalikkust ja andnud sisendi autoritele tuleviku plaanide osas, kuna projekt ei ole täna veel täiesti valmis.

3.3 Teostatud tööde logi

3.3.1 Commit-id

Autor	Kuupäev	Kirjeldus
Gert Mänd	16.05.2021 12:20	route probleem parandatud, parklakoha tabeli vea teated parandatud
Gert Mänd	16.05.2021 08:47	buttonitele päringute ajaks disable, disaini fixid ja täiendused
Kevin Kiil	15.05.2021 18:18	Eemalda kasutaja nimekirjast
Kevin Kiil	15.05.2021 18:01	Liikmed -> Kasutajad (nimemuudatus)
Gert Mänd	15.05.2021 20:20	broneeringu eemaldamine, vabastatud koha eemaldamine, testide lisamine, disaini fixid
Kevin Kiil	15.05.2021 17:50	Kasutajate lisamise Loobu nupp loogika
Gert Mänd	15.05.2021 17:54	table id fix
Gert Mänd	15.05.2021 16:22	viimased fixid, broneerimine, frontend, testid
Kevin Kiil	14.05.2021 12:06	Asutuse logides: Kui admin puudub, siis kuvatakse ka Admini veerus rea kohta "Admin puudub"
Kevin Kiil	14.05.2021 11:50	Asutuse logide puhul kuvatakse veerus "Admin" ka admini ees- ja perekonnanimi
Kevin Kiil	13.05.2021 13:34	Liikme eemaldamine asutuse nimekirjast
Kevin Kiil	13.05.2021 12:25	Kasutaja liitumiskutse vastus logitud.
Kevin Kiil	10.05.2021 19:36	Foto
Gert Mänd	10.05.2021 19:41	parklakoha broneerimiseks vabastatud kohtade otsimise bug fix
Taavi Meier	09.05.2021 20:39	TopBar changes. TODO: Eraldi komponent kutsete vastu võtmiseks.
Taavi Meier	09.05.2021 19:41	Warningud, naming
Kevin Kiil	09.05.2021 19:40	Liikmete lisamise front fix
Taavi Meier	09.05.2021 19:23	Commit fix
Kevin Kiil	09.05.2021 18:42	Nüüd peaks front ka mergitud olema
Taavi Meier	09.05.2021 18:48	Uuenda nupp enterpriseSelection pagel.
Kevin Kiil	09.05.2021 17:50	Invitationite lisamine APIs ja frontist vajaminevate emailide kätte saamine
Taavi Meier	09.05.2021 12:22	Invitation approval done. TODO: enterprises and invitations update after approval.
Taavi Meier	08.05.2021 14:12	Invitation acceptance modal without functionality
Taavi Meier	08.05.2021 11:22	Multiple parkingspot mainusers car nrs view fix

Taavi Meier	07.05.2021 14:27	Tests fix
Taavi Meier	07.05.2021 14:24	Warningud, Invitations view
Kevin Kiil	07.05.2021 14:51	Ühte sama autot ei saa mitu korda lisada, otsi kohta nuppu enam admin, userstable details broneeringute all pole
Kevin Kiil	07.05.2021 13:32	Logid valmis
Kevin Kiil	07.05.2021 12:25	Logid on enterprise vahel eristatavad, kuvatakse ainult käesoelva enterprisega seonduvaid logisid
Gert Mänd	07.05.2021 11:30	added moment locale to dates, weekday should be correct and fixed
Gert Mänd	07.05.2021 11:14	Update changeDate.ts
Gert Mänd	07.05.2021 11:09	give spot fix, date day letter fixed (needs live testing)
Kevin Kiil	07.05.2021 11:35	Peakasutaja tegevused logitud
Kevin Kiil	07.05.2021 10:09	Logide front-end development (status: Ready for Prod)
Taavi Meier	07.05.2021 07:07	User enterprise invitations response API .
Taavi Meier	06.05.2021 19:52	Invitation entity
Taavi Meier	06.05.2021 19:22	Testid korda + ReserveParkingSpot logging can be done only if spotaccount is not null.
Kevin Kiil	07.05.2021 07:20	Front (logid)
Kevin Kiil	05.05.2021 20:01	Olulisemad tegevused saavad logitud.
Taavi Meier	05.05.2021 14:58	Testid korda.
Kevin Kiil	02.05.2021 19:46	Panin appsettingu tagasi
Kevin Kiil	02.05.2021 19:21	fix
Kevin Kiil	02.05.2021 12:14	Logid -> sõiduki lisamine/kustutamine
Kevin Kiil	01.05.2021 15:11	LogService
Kevin Kiil	01.05.2021 11:58	Done: Kasutajate lisamine admini kasutajate tabelist
Kevin Kiil	01.05.2021 09:08	Admini kasutajatetabelis detailide alt näeb kasutajate broneeringuid
Taavi Meier	30.04.2021 21:59	ParkingSpot table carNrs
Taavi Meier	30.04.2021 10:06	Testid @ 74,93%
Taavi Meier	30.04.2021 06:21	Api structure small fix #2
Taavi Meier	30.04.2021 06:18	API structure small fix
Kevin Kiil	29.04.2021 14:51	Cars tabel design fix
Kevin Kiil	29.04.2021 14:42	adminid näevad ka nüüd autosid kasutaja detailide alt, cars 'Ajutine?' väärtus fix
Taavi Meier	29.04.2021 16:14	Tests -> 66,32%
Taavi Meier	29.04.2021 13:33	ParkingSpot mainuser selection fix.
Taavi Meier	29.04.2021 12:11	Tests -> 61,93%
Kevin Kiil	29.04.2021 10:05	quick fix -> removing id parameter for DeleteCar method
Taavi Meier	29.04.2021 12:09	Unused namespaces cleared. Tests -> 61,93%
Taavi Meier	26.04.2021 20:45	AccountServiceTests
Kevin Kiil	26.04.2021 18:13	Kasutajad saavad enda detaile muuta! Tehtud nii API kui front. Kasutajad ei saa oma emaili muuta.
Kevin Kiil	26.04.2021 11:12	Design fix front-endis
Kevin Kiil	26.04.2021 11:05	Numbrimärgi kontroll ja front muudatus, et inputfield oleks näha
Taavi Meier	26.04.2021 20:43	Testid AccountService
Taavi Meier	26.04.2021 00:02	Tagasiside vorm, parkingspots multiple add file check and only new numbers added. Adding user mr/mrs -> mees/naine
Taavi Meier	25.04.2021 22:46	Small fix
Taavi Meier	25.04.2021 20:55	LaptopWindows warning

Kevin Kiil	25.04.2021 19:45	Üks lisa error fixed
Kevin Kiil	25.04.2021 19:44	Errorite fixid
Kevin Kiil	25.04.2021 19:37	Aitasin Taavit ühe raske probleemi lahendamisel
Kevin Kiil	25.04.2021 19:26	AddUserCar & DeleteUserCar API ja FRONT valmis
Taavi Meier	25.04.2021 20:49	ParkingSpots from excel fix + refactor
Taavi Meier	25.04.2021 19:25	Testide baas.
Gert Mänd	24.04.2021 22:08	package json fix
Kevin Kiil	23.04.2021 20:06	car add ja delete API meetodid
Taavi Meier	24.04.2021 14:50	excelReader, parkingSpotsFrom excel added to parkingspot add modal
Taavi Meier	23.04.2021 02:40	parkingLotPlan fix
Taavi Meier	23.04.2021 02:26	Postman Validate car number
Taavi Meier	23.04.2021 00:33	POSTMAN-> Validate phone number
Taavi Meier	22.04.2021 19:17	TODO-s add
Taavi Meier	22.04.2021 19:05	ParkingSpots add from excel done.
Taavi Meier	22.04.2021 15:42	EnterpriseAdd done
Taavi Meier	22.04.2021 13:17	Enterprise Create back-end
Taavi Meier	21.04.2021 23:48	Enterprise Add modal front end done.
Gert Mänd	21.04.2021 21:27	sm width change
Gert Mänd	21.04.2021 21:20	warnings fix
Kevin Kiil	21.04.2021 20:45	userCars personal table UI design
Gert Mänd	21.04.2021 21:18	loading fix, give, release modal grid fix, admin parkingtable loading fix, warnings
Kevin Kiil	21.04.2021 20:22	API changes
Kevin Kiil	21.04.2021 20:00	Userstable usercars query in back-end development
Kevin Kiil	21.04.2021 19:32	Revert "warningud"
Taavi Meier	21.04.2021 13:32	warningud
Taavi Meier	21.04.2021 13:30	EnterpriseSelectionPage warning fix, enterpriseAddModal base add
Taavi Meier	20.04.2021 14:36	Register new user change
Gert Mänd	19.04.2021 17:20	Update package-lock.json
Gert Mänd	19.04.2021 17:18	register modal naming
Gert Mänd	19.04.2021 17:15	code check
Gert Mänd	19.04.2021 17:00	Update package.json
Taavi Meier	19.04.2021 14:21	userAdd modal close fix
Taavi Meier	19.04.2021 14:08	Email validation for user add
Taavi Meier	19.04.2021 13:22	userAddModal component
Gert Mänd	19.04.2021 11:03	disaini fixid, @kevkiil meetodi lisamine (ToDo'd üle vaadata) , päringud
Taavi Meier	19.04.2021 12:29	Register new user.
Kevin Kiil	18.04.2021 19:59	refaktooring
Kevin Kiil	18.04.2021 19:58	UserCars in usersTable
Taavi Meier	18.04.2021 17:56	parkingPlan update
Taavi Meier	18.04.2021 17:38	User registration form without submit logic
Taavi Meier	18.04.2021 17:07	Register new user button
Gert Mänd	18.04.2021 17:39	Update parkingSpotTableComponent.tsx
Kevin Kiil	18.04.2021 18:02	Refaktoorimine, usertables cars
Gert Mänd	18.04.2021 16:47	Update parkingSpotTableComponent.tsx
Kevin Kiil	18.04.2021 16:48	Cars front design

Kevin Kiil	18.04.2021 16:29	GetUserData
Kevin Kiil	18.04.2021 16:28	Merge
Taavi Meier	18.04.2021 16:18	MainUserDelete fix (data is removed from table)
Taavi Meier	18.04.2021 14:24	parkingSpotMainUser delete front end.
Taavi Meier	18.04.2021 13:51	delete mainuser api
Gert Mänd	18.04.2021 12:53	sitealert fix, parklakohe main user lisamise warning fix
Gert Mänd	18.04.2021 12:47	update parkingSpotTable after adding/deleting spot
Taavi Meier	18.04.2021 12:17	ParkingSpot delete will release users
Kevin Kiil	18.04.2021 16:23	Userstable, userdetailstabid, refaktoorimine, UserCars, front & back-end arendus.
Gert Mänd	18.04.2021 03:52	parklakoha staatus frontendis
Gert Mänd	18.04.2021 03:22	tühistamise kinnitus
Gert Mänd	18.04.2021 02:52	vabade parklakohtade broneerimine, kohade otsimise loogika lisad, oma kohta ei saa broneerida, enterprise kontroll
Gert Mänd	18.04.2021 02:14	additional designing fixes, modals changing, booking api needs to be done
Gert Mänd	18.04.2021 01:25	disaini fixid, parklakoha broneerimine, vabade kohtade otsimine, kõikide vabade kohtade objektid tagastus frontendi, modalis, kohtade näitamine, valimine, päevade kalkulatsioon
Gert Mänd	17.04.2021 18:20	@kevkiil not used lines commented
Taavi Meier	17.04.2021 16:08	Small update
Taavi Meier	17.04.2021 16:02	FRONT korda
Taavi Meier	17.04.2021 15:57	API korda
Taavi Meier	17.04.2021 15:54	ParkingSpotMainUsers add + canBook change done.
Taavi Meier	17.04.2021 14:35	Testandmeid juurde
Taavi Meier	17.04.2021 10:40	Warningud, change canBook status && addMainuser.
Taavi Meier	16.04.2021 18:53	Grid + ParkingLotPlanAdd Button modification
Taavi Meier	15.04.2021 22:02	Controlleris admini asjad korda
Taavi Meier	15.04.2021 21:03	Front-end error fix, parkingtable
Kevin Kiil	15.04.2021 22:28	UsersTable componentideks, UsersDetails modaali loomine.
Gert Mänd	15.04.2021 19:24	bookingmodali alustus, admin parkingtable lisa
Gert Mänd	15.04.2021 18:51	Create README.md
Taavi Meier	15.04.2021 17:37	ErrorFix + TODO: markers
Taavi Meier	14.04.2021 21:30	lot_pic_change
Taavi Meier	14.04.2021 21:20	parkingTable refactor done. ParkingSpotMainUser add TODO + seperate main users view.
Taavi Meier	14.04.2021 16:34	parkingTable table component refactoring
Taavi Meier	14.04.2021 14:58	Merge branch 'main' of https://github.com/gertmand/ParkingAPP into main
Kevin Kiil	13.04.2021 21:04	Refaktoorimine
Kevin Kiil	13.04.2021 19:12	Liikmete tabeli otsimine, avatar, lisaväljade lisamine (muudetud nii back-end kui front-end)
Taavi Meier	14.04.2021 14:57	ParkingSpotMainUserAdd/delete modal view without function
Taavi Meier	13.04.2021 15:30	ParkingSpotMainUserAdd frondis type ja query lisatud.
Taavi Meier	13.04.2021 15:22	ParkingSpotMainUser back-end done
Taavi Meier	13.04.2021 13:48	Small fix
Taavi Meier	13.04.2021 13:13	ParkingSpotMainUsers view -> ParkingTable done.
Taavi Meier	13.04.2021 12:11	getParkingSpotByUserId .SingleOrDefault change to FirstOrDefault // muidu ei saa mitu peakasutajat olla.

Taavi Meier	13.04.2021 12:06	ParkingSpotMainUsers method change -> API
Taavi Meier	12.04.2021 20:59	GetParkingSpotMainUsers API pool valmis.
Taavi Meier	12.04.2021 19:26	SmallFix
Taavi Meier	12.04.2021 19:23	ParkingSpot adding done
Taavi Meier	12.04.2021 16:51	AddParkingSpot API pool done.
Taavi Meier	10.04.2021 18:24	parkinglot plan add modal
Taavi Meier	10.04.2021 18:12	deleteconfirm for parkingTable
Taavi Meier	10.04.2021 17:03	parkingTable front paremaks. Puhastus
Taavi Meier	10.04.2021 16:10	System.Obsolete -> IHostEnvironment
Taavi Meier	10.04.2021 14:50	Saving parkinglot plan.
Taavi Meier	09.04.2021 18:32	Frondis parklapildi kuvamine, API pool tegemata
Taavi Meier	08.04.2021 21:02	parkingTable fix
Taavi Meier	08.04.2021 20:18	ParkingSpot delete töötab
Taavi Meier	08.04.2021 16:34	warningud
Taavi Meier	07.04.2021 21:29	DeleteParkingSpot; GetById parkingspot.
Taavi Meier	06.04.2021 21:23	parkingTable fix
Gert Mänd	06.04.2021 20:01	Update apiUrl.ts
Gert Mänd	06.04.2021 19:57	Update apiUrl.ts
Taavi Meier	06.04.2021 19:56	parkingTable mod.
Gert Mänd	06.04.2021 13:04	kasutaja andmete laadimine, spinner, App Loading / päringute parandused
Gert Mänd	06.04.2021 11:56	loading fix
Gert Mänd	06.04.2021 11:47	loading fix
Gert Mänd	06.04.2021 11:39	warnings
Gert Mänd	06.04.2021 11:30	Update package-lock.json
Taavi Meier	05.04.2021 22:20	DataGrid @ parkingTable.
Taavi Meier	05.04.2021 19:03	parkingTable kood puhtamaks
Taavi Meier	05.04.2021 18:53	Parkimiskohtade otsing PARKLAKOHAD lehel
Gert Mänd	05.04.2021 15:43	Update RenderView.tsx
Gert Mänd	05.04.2021 12:27	loading fix
Gert Mänd	05.04.2021 12:19	Update RenderView.tsx
Gert Mänd	05.04.2021 12:16	frontend loading fix, givespot query fix, KEVIN BLJAT
Gert Mänd	05.04.2021 11:34	fixes
Gert Mänd	05.04.2021 00:15	parklatabeli disaini muudatused
Taavi Meier	05.04.2021 18:50	Revert "Search for parkingspots @ ADMIN-> Parklakohad"
Taavi Meier	05.04.2021 18:47	Search for parkingspots @ ADMIN-> Parklakohad
Taavi Meier	05.04.2021 00:04	Enterprise parkingspots view done.
Kevin Kiil	04.04.2021 23:25	usersTable
Gert Mänd	04.04.2021 23:22	Parklatabel näita reserve samuti
Kevin Kiil	04.04.2021 23:21	Liikmete tabel v 1.0
Taavi Meier	04.04.2021 23:35	Revert "Enterprise queries -> getallparkingspots"
Taavi Meier	04.04.2021 23:29	Enterprise queries -> getallparkingspots
Kevin Kiil	04.04.2021 22:37	Arendus
Taavi Meier	04.04.2021 22:41	GetAllEnterpriseParkingSpots validation fix
Gert Mänd	04.04.2021 22:29	auto mapper
Taavi Meier	04.04.2021 22:30	GetAllEnterpriseParkingSpots API pool done.
Gert Mänd	04.04.2021 21:46	Update AdminPage.tsx
Taavi Meier	04.04.2021 21:38	parkintSpaceQueries, parkingSpotType deleted

Gert Mänd	04.04.2021 21:40	Update EnterprisesController.cs
Kevin Kiil	04.04.2021 21:40	Proovime uuesti
Kevin Kiil	04.04.2021 21:34	TakeRemote
Gert Mänd	04.04.2021 21:16	publish
Taavi Meier	04.04.2021 21:21	Datacontext -> DAL folder;
Kevin Kiil	04.04.2021 21:33	Admin queries - front & back
Gert Mänd	04.04.2021 18:28	Layout fix, pildi laadimise aken, profiil, lehe tiitlid
Gert Mänd	04.04.2021 17:28	Footer, navbar updates, logi välja / enterprise mobiili vaatesse
Gert Mänd	04.04.2021 16:20	disaini muudatused, avatari lisamine, kuvamine frontendis
Gert Mänd	04.04.2021 15:04	koha otsimise algoritmi implementeerimine, disaini parandused, clg eemaldused, admin paneeli alustused
Gert Mänd	03.04.2021 18:12	fixes, book spot modal, admin page component, design,
Gert Mänd	03.04.2021 15:11	Spot buttons
Kevin Kiil	23.03.2021 20:47	API poolelt lisatud EnterpriseUserDataResponsile isAdmin
Gert Mänd	21.03.2021 19:56	TopBar fix, lehe vahetuse fixid, enterpriseselection fix
Gert Mänd	21.03.2021 18:23	Fixid, andmete fetchimise loogika frondis, topbar, enterprise valimine frondis
Gert Mänd	19.03.2021 12:02	parklakoha vabastamine, small fixes
Gert Mänd	19.03.2021 10:56	ID asemel NIME kuvamine tabelis
Gert Mänd	19.03.2021 10:41	frontis parklakoha andmete kuvamine tabelis
Gert Mänd	19.03.2021 10:28	Parklakoha tabel
Gert Mänd	17.03.2021 18:48	Reservation POST meetod APIS
Gert Mänd	11.03.2021 10:17	frontend muudatused
Gert Mänd	10.03.2021 12:55	ProfileDetails frontend
Gert Mänd	09.03.2021 15:56	frontend basic parking data from db to frontend
Gert Mänd	09.03.2021 15:56	frontend basic parking data from db to frontend
Gert Mänd	09.03.2021 14:15	modelite transfer, parkingSpotService, kasutaja enterprise andmete tagastamine meetod, nimed, mapper
Gert Mänd	08.03.2021 23:03	Get Enterprise, kontrollib, kas User on enterprise's olemas
Gert Mänd	08.03.2021 21:44	Update EnterprisesController.cs
Gert Mänd	08.03.2021 21:41	Enterprises get meetod by userid töötab
Gert Mänd	08.03.2021 21:28	enterprise service first methods
Gert Mänd	08.03.2021 20:45	Main classes / models, dtos
Gert Mänd	25.02.2021 17:03	muudatused
Gert Mänd	25.02.2021 15:55	FrontEnd sisse logimine, andmete store lisamine frontendis, service kasutamine, helperid, routes
Kevin Kiil	19.02.2021 13:06	Peale õige kasutajanime ja parooli sisestamist saab edasi kodulehele.
Kevin Kiil	19.02.2021 12:57	Õige kasutajanime ja parooli sisestamisel lisatakse localStorage token, kuid on eraldi home page avamine tokeniga tegemata.
Taavi Meier	10.02.2021 15:51	Namespace correction
Gert Mänd	08.02.2021 13:01	frontend
Gert Mänd	07.02.2021 17:48	API Init
Gert Mänd	07.02.2021 16:58	Initial commit

3.3.2 Toggle

Tabel 3. Tiimiliikmete ajaline panus.

	Analüüs	Arendus	Enesearendus	Kokku
Taavi	63h 56min	185h 08min	6h 22min	255h 26min
Gert	54h 33min	166h 36min	9h 38min	230h 47min
Kevin	60h 34min	172h 18min	4h 36min	237h 28min

3.4 Hinnang projekti teostamise protsessi kohta

3.4.1 Projekti juhtimine ning teostamine

Projekti alguses pani tiim kokku eesmärgid, kuhu enam-vähem projektiga plaaniti jõuda. Tööd jaotati ühise arutelu tulemusena lähtudes tiimiliikmete oskustest. Iga-nädalaselt toimusid koosolekud, kus vaadati ühiselt tööd üle ja lahendati ilmnenud probleeme. Kiireloomuliste probleemidega tegeles tiim koheselt.

Projekti käigus toimusid koosolekud juhendajaga, kellele tutvustati tehtud töid ning kes omakorda andis nõu ja ideid, millele võiks tähelepanu täiendavalt pöörata. Tiimisisene juhendamine toimis vastastikuselt ning peamiselt saadi abi teistelt liikmetelt ja internetis leitavast dokumentatsioonist.

Peamine suhtluskanal oli MS Teams. Tihti kasutas tiim videovõimalust ning paarisprogrammeerimist.

3.4.2 Projekti teostuse positiivsed ja negatiivsed tähelepanekud

Kuna meeskond oli eelnevalt koos töötanud RIK-i projekti puhul, siis oli väga sujuv keskkondade ülesseadmine ning ühiste otsuste vastu võtmine. Suhtlus tiimis toimis hästi ja sellest tulenevalt suudeti palju eesmärke täita plaanipäraselt.

Koostöö juhendajaga toimis samuti hästi – infoliikumine oli kiire ja asjakohane. Kogu tiimi tööd iseloomustab konkreetsus ja vähene aja raiskamine.

Esialgsete plaanide täitmine jäi umbes 25% ulatuses täitmata – peamiselt seetõttu, et tiim ei osanud hinnata tehtavatele töödele kuluvat aega. Mõne funktsionaalsuse juures kulus rohkem aega, kuna palju oli vaja tähelepanu pöörata detailidele ja erinevatele seostele koodis.

Teine oluline kitsaskoht tuli välja tööde planeerimise ja retrospektiivide kohalt. Kuigi tööde planeerimine suhtluse seisukohast toimis ei pandud selle käigus paika detailsemaid plaane. See viis selleni, et tehtud töödele oli hiljem tiimisiseselt erinevaid arvamusi ja see tekitas vaidluseid, mida oleks saanud vältida paremini planeerides. Tiimis puudusid ka hästi planeeritud retrospektiivid, mis oleks aidanud vältida tüüpvigade esinemist.

Tiimiliikmete taust ja oskused olid erinevad nagu ka arusaamad koodikirjutamisest. See tekitas samuti vastuolusid ja vajadust aega kulutada koodi refaktoreerimisele.

Hilisema retrospektiivi käigus leidis tiim, et kõik need kitsaskohad oleks saanud välistada projekti algul paremini tegevust planeerides.

3.4.3 Hinnang üldisele projekti teostamise protsessile

Üldine hinnang projekti teostamisele on hea. Tiim jäi väga rahul juhendaja panusega ning üleüldine taustajõud (TalTech ja juhendaja) soodustas oluliselt projekti täideviimist. Tiimis töötasid liikmed erinevas tempos ja kuigi kuude lõikes ajalised panused erinesid oli tervikuna panustamine võrdne.

Juhendajalt saadi asjakohast tagasisidet ja suunavaid ideid, mis aitas keerulisematele probleemidele kiiremini lahendusi leida. Meeskonna-siseselt jaotusid rollid vastavalt oskustele ning see muutis tiimisisese juhendamise lihtsamaks ning efektiivsemaks.

Projekti käigus tehtud tööle teostas tiim jooksvalt nelja-silma kontrolli, selle käigus tuvastati koodis palju väiksemaid ja suuremaid vigu, mis üheskoos ära parandati. Seatud eesmärgid olid vastavuses oskustega ning ebareaalseid eesmärke tiim endale projekti alguses ei seadnud.

Retrospektiivi käigus selgus, et peamine töökvaliteedi vähendaja oli projekti alguses tehtud planeerimine, mida oleks pidanud tegema detailsemalt.

3.5 Olemasolevad lahendused

Eesti turul pakub sarnast lahendust Barking [1]. Nad pakuvad digitaalset lahendust klientidele oma parklat manageerida samal ajal teenides tulu. Kinnised parklad, mille parkimiskohad on tihti tühjad või kasutamata tehakse avalikuks kogu Eestile. Sisuliselt antakse kasutamata parkimiskoht soodsama hinnaga, kui keskmised parkimiskohad, autojuhile, kes parasjagu parkimiskohta selles regioonis otsib. Lisaks sellele jääb kogu otsustamine kliendile, valides päevad ning kellaaja. Nende sõnul võib aastane tulu tõusta kuni 3000 euroni, sõltuvalt ajast, kaua parkimiskohta renditakse. Samuti kasutatakse parklates võimalusel nende endi väravat, mis läbi mobiilirakenduse või helistamise avatakse, tehes parkla kasutamise turvaliseks ning kindlaks.

3.6 Meeskondlik konsensuslik hinnang

	Gert	Taavi	Kevin
Gert	**X**	0	0
Taavi	0	**X**	0
Kevin	0	0	**X**

Kokkuvõte

Lõputöö eesmärgiks oli luua parklakohtade rakendus, kus kasutajad saavad parklakohti vabastada ning broneerida. Lisaks oli rakenduse eesmärk luua abivahend administraatoritele, kust on võimalik näha ning muuta kasutajate ja parklakohtadega seonduvat informatsiooni.

Ettevõtete, korteriühistute, koolide jms asutuste puhul on parklates üldiselt kasutajaid rohkem kui kohti. Esimest korda puutusid probleemiga autorid kokku RIK-is, kus sooviti parklat efektiivsemalt kasutada ja suurendada parklakoha kasutuse aega.

RIK-ile tehtud projekti pealt analüüsiti probleemi üldisemalt ning tehti valmis universaalne rakendus, kus kasutajad saavad luua enda asutustele keskkonna, lisada sinna kasutajaid ja parklakohti ning seejärel on võimalik parklakohtade kasutajatel endale määratud kohti vabastada. Teised kasutajad saavad seeläbi neid kohti otsida ja broneerida. Administraatoritel on võimalik teha kõiki *CRUD* toiminguid kasutajate ja parklakohtadega.

Tiim töötas SCRUM ideoloogiale tuginedes ning kasutades koodi kirjutamise häid tavasid. Suur osa oli nelja-silma kontrollil ja koodi kvaliteedi pideval refaktoreerimine ja parandamisel. Kasutajaliidese jaoks kasutati React raamistikku ning JavaScript ja TypeScript programmeerimiskeeli. API tehti ASP.NET Core 3.1 raamistikul C# keelt kasutades. Produktsioonis kasutab rakendus InMemory andmebaasi, lokaalselt kasutati Dockeri abil PostgreSQL andmebaasi.

Projekti teostus sujus hästi ja ainukese olulise kitsaskohana kaardistati retrospektiivi käigus ära esmase analüüsi kitsaskohad, mis tingisid ajalise ressursi raiskamist ning probleeme koodis. Tiimis toimus pidev suhtlus ning tööd jaotati liikmete oskusi ning eelistusi silmas pidades.

Kasutatud materjalid

[1] „Barking,“ [Võrgumaterjal]. Available: https://barking.ee/.

[2] A. documentation. [Võrgumaterjal]. Available: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0.

[3] Facebook, „React,“ [Võrgumaterjal]. Available: https://reactjs.org/.

[4] Microsoft, „Typescript,“ [Võrgumaterjal]. Available: https://www.typescriptlang.org/.

[5] The PostgreSQL Global Development Group. [Võrgumaterjal]. Available: https://www.postgresql.org/.

[6] J. Watmore, „Aspnet core 3 boilerplate api with email sign up verification authentication,“ [Võrgumaterjal]. Available: https://jasonwatmore.com/post/2020/07/06/aspnet-core-3-boilerplate-api-with-email-sign-up-verification-authentication-forgot-password.

[7] Dan Abramov and the Redux documentation authors, „React-redux,“ [Võrgumaterjal]. Available: https://react-redux.js.org/.

[8] Material UI, „Material UI,“ [Võrgumaterjal]. Available: https://material-ui.com/.

[9] D. IO. [Võrgumaterjal]. Available: https://github.com/devias-io/material-kit-react.

[10] S. McConnell, Code Complete: A Practical Handbook of Software Construction, Second Edition, Redmond: Microsoft Press, 2004.

[11] Coding Homework, „RESTful API with ASPNET Core,“ [Võrgumaterjal]. Available:

https://www.youtube.com/results?search_query=RESTful+API+with+ASP.NET+Core+
3.0++-+Creating+Album+Entity.

[12] M. Fowler, D. Rice, M. Foemmel, E. Hieatt, R. Mee ja R. Stafford, Patterns of Enterprise Application Architecture, 2002.

[13] R.C.Martin, Clean Code: A Handbook of Agile Software Craftsmanship, New Jersey: Prentice Hall, 2008.

[14] ParkIT. [Võrgumaterjal]. Available: https://parkit.ee/meist/.

[15] TheCodeBuzz. [Võrgumaterjal]. Available: https://www.thecodebuzz.com/unit-test mock-automapper-asp-net-core-imapper/.

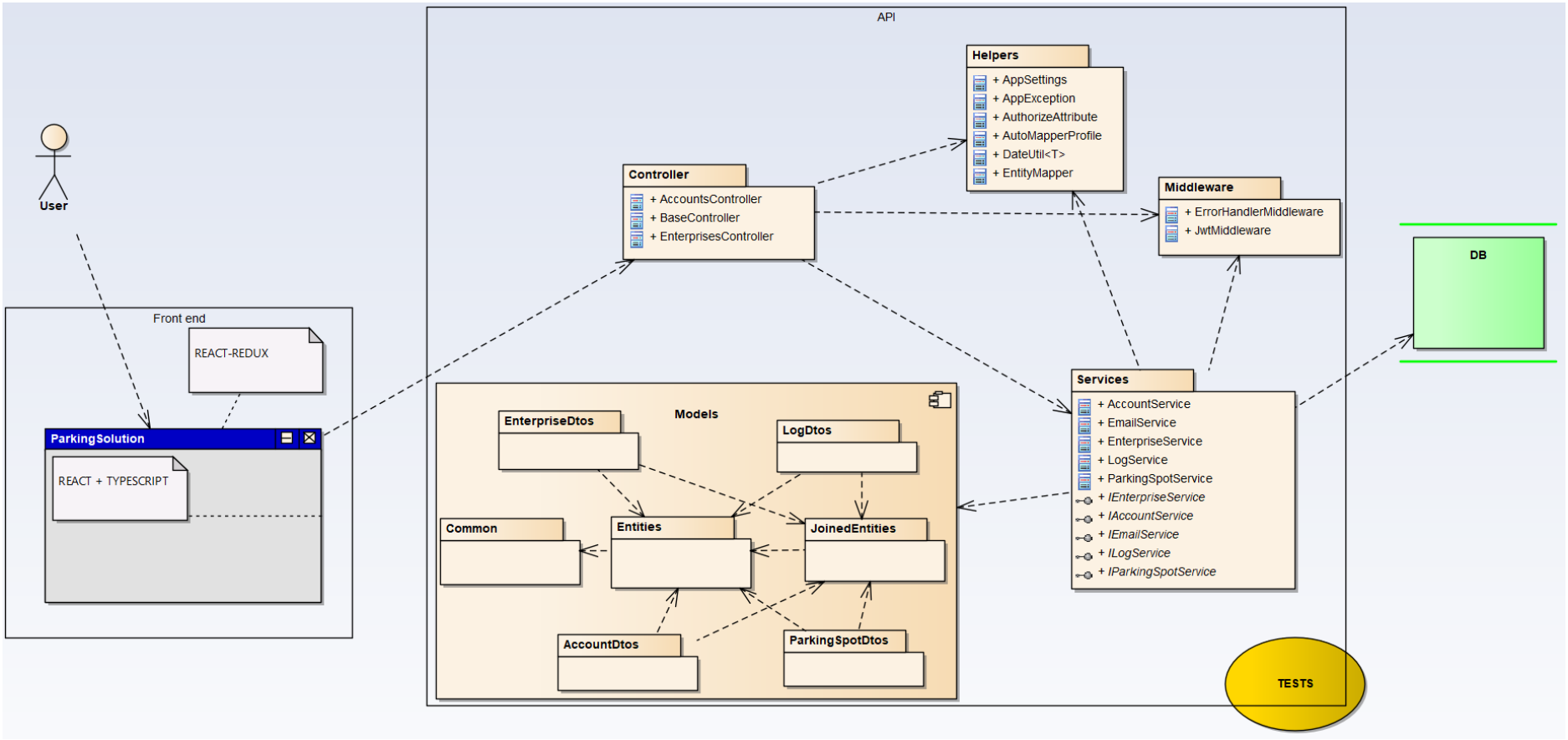
Lisa 1 - Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Autorid Taavi Meier, Kevin Kiil ja Gert Mänd

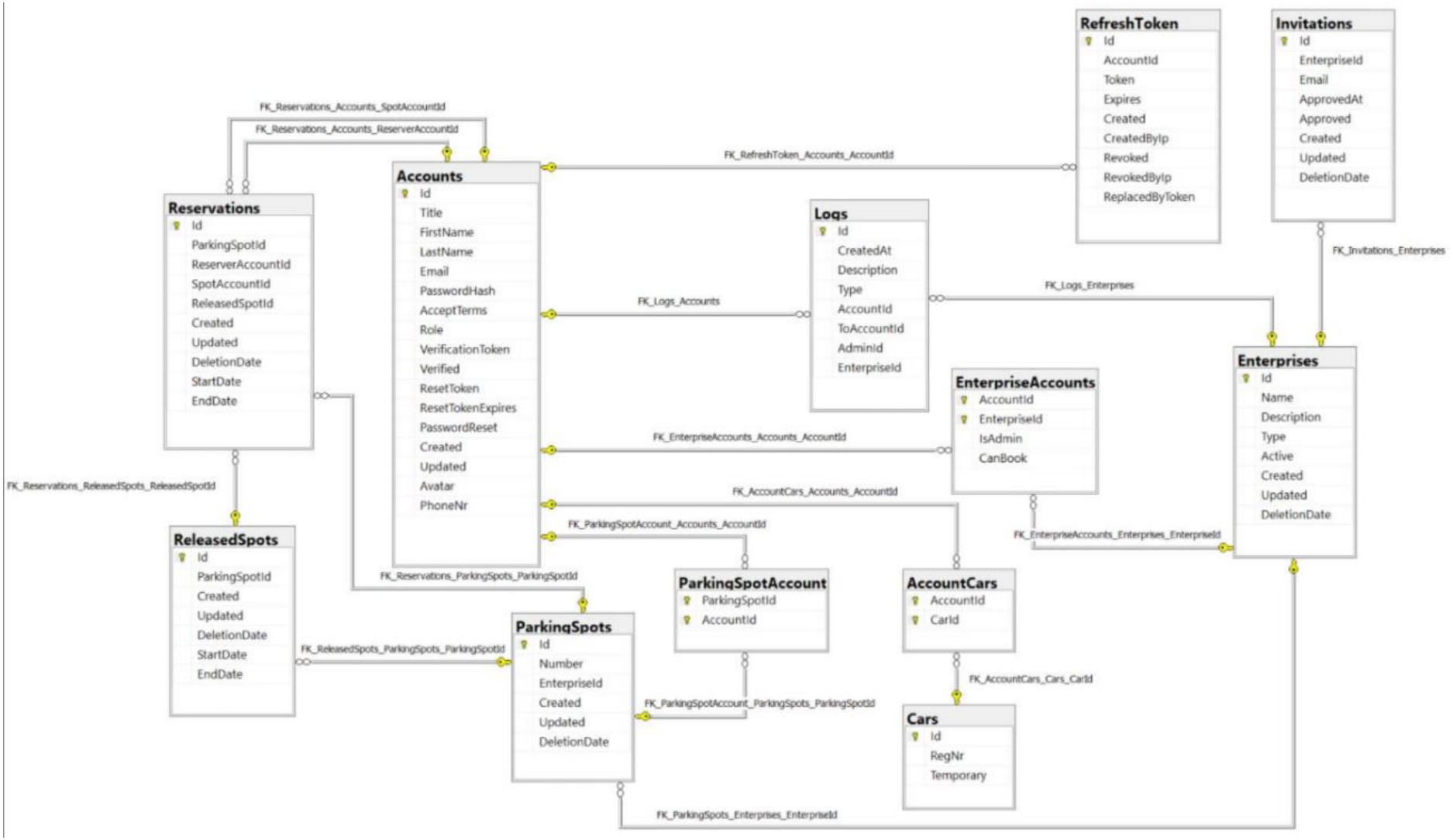
1. Annavad Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Parkimiskohtade jagamise ja haldamise rakendus," mille juhendaja on Tarvo Treier
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Autorid on teadlikud, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autoritele.
3. Autorid kinnitavad, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

18.05.2021

Lisa 2 - Rakenduse arhitektuur



Lisa 3 – Andmebaasi diagramm



Lisa 4 - Taavi tehtud tööd ja eneseanalüüs

Tehtud tööd projektis

Projekti juures töötasin front-endi ja API-ga ning lisaks seadsin üles suhtluskeskkonnad tiimi töökorralduse edendamiseks.

Minu peamine aeg läks administraatori vaates parklakohtade tabeli realiseerimisele, kus ehitasin üles järgnevad funktsionaalsused:

- Otsing parklakoha numbri järgi. Eesmärgid kiirelt tuvastada parkimiskohaga seotud info olukorras, kus parkimiskohti on palju.
- Parklakohtade tabelis on võimalik lisada parklaplaani, et kasutajatel oleks võimalik pildilt vaadata parklakohti. Hetkel realiseeritud lokaalses rakenduses.
- Parklakohtade lisamisel tabelisse on võimalik seda teha üksikult või otse Exceli tabelist. Mõlemal juhul on kasutajatele ette kirjutatud lihtsalt jälgitav juhend, kuidas õigesti toimida.
- Parklakohtade kustutamine, mis omakorda eemaldab ka seoses parklakoha ja kasutajate vahel.
- Parklakohtadele peakasutaja määramine, muutmine.
- Rakenduses üldine broneerimise loogika näeb ette, et kasutajad kellel on määratud koht olemas, täiendavalt broneerida kohti ei saa. Kuna ühel kohal võib olla mitu peakasutajat, siis on vajalik võimaldada neile ka broneerimisõigust olukordades, kus kasutajad samaaegselt parklat kasutada soovivad. Selle jaoks lisasin eraldi õiguse muutmise funktsionaalsuse.

Lisaks:

- Kasutajakontode registreerimine. Sellega koos töötasin välja ka eraldi e-maili aadressi kontrolli. Kasutajate kontroll toimub e-maili järgi, ning ühe e-mailiga

saab eksisteerida vaid üks konto. Konto registreerimisel saadetakse e-mail API-poolsesse kontrolli, kust tagastatakse tõeväärtus, mille põhjal viiakse registreerimine lõpuni või katkestatakse ning palutakse kasutajal andmeid kontrollida.

- Peale kasutajakonto registreerimist saab kasutaja sisse logida, misjärel suunatakse ta asutuste lehele, kus on võimalik lisada asutus. Asutuse lisamine on esimene samm oma keskkonna loomiseks. Kui asutus on lisatud on võimalik edasi liikuda administraatoritele mõeldud toimingutega.
- Tiim otsustas realiseerida kasutajate lisamise asutusse kutse-põhise loogikaga, mis näeb välja nii:
    1. Administraator sisestab e-mailid, keda soovid asutusse lisada
    2. Kui kasutaja sisse logib, kuvatakse talle menüüribal teateid, kust on võimalik kutseid kinnitada/tagasi lükata.

  Töötasin välja teadete kuvamise vastavalt kutsetele ning loogika kutsete kinnitamisest/tühistamisest lähtuvalt.
- Arutelust juhendajaga saime sisendi töötada välja ka API-poolne kontroll kasutajate autode numbrimärkide ja telefoninumbrite kontrolliks. Seeläbi on võimalik realiseerida näiteks tõkkepuu avamist telefoninumbriga või teistel kasutajatel teha kindlaks/teavitada teisi kasutajaid, kui on toimunud valele kohale parkimine. Telefoninumbri kontrolli korral tagastab süsteem tõeväärtuse kasutaja eksisteerimise kohta ning numbrikontroll tagastab info, kas sisestatud numbrimärk tohib asutuse parklas parkida või mitte.

Rakendusele kirjutasin ühiktestid ja testisin API-t Postman tarkvaraga. Projektis täitsin jooksvalt ka projektijuhi rolli ning sarnaselt teiste liikmetega olin tiimiliikmetele juhendaja rollis ning andsin oma panuse puhta koodi kirjutamisele ning refaktoreerimisele.

Eneseanalüüs

Senised omandatud teadmised koolis on loonud suhteliselt hea teadmistepagasi, mille pealt oli projekti teostamine kindlasti kergem. Pean väga oluliseks tiimi sisekliima tervislikkust ning arvan, et meie projekti tiimis oli sisekliima väga hea. Toimus konkreetne ja lahendustele suunatud arutelu ning aja raiskamist oli vähe. Tagantjärele

tarkusena tunnen, et projektijuhtimise ja tarkvara loomise detailsem *know-how* soodustanuks veel efektiivsemat koodikirjutamist ning rakendusele täiendavate funktsionaalsuste juurde loomist.

Arendaja rollis tundsin ennast hästi, oli kohti, kus jäin liiga pikalt mingite probleemidega maadlema ja oli ka hetki, mil oleks võinud mõned otsused teistega läbi arutada. Selle viimase tähelepaneku seostan sarnaselt kogu tiimiga projekti esialgse analüüsi ja planeerimise puudujääkidega.

Olulise tähelepanekuna ei läinud projekti algus nii nagu lootsin – asjad võtsid oodatust rohkem aega ning esimesel kuul oli keeruline leida motivatsiooni töö tegemiseks – seda seetõttu, et lähteülesande koostas tiim endale ise ja täna tagasi vaadates ei osanud alguses näha, kui mahukas tegelikult antud töö on.

Kogemuste ja oskuste kohalt tunnen, et selle töö raames arenesin väga palju ja sain nii juhtimise kui arendusprotsessi osas vajalikke teadmisi, mida edasises karjääris rakendada.

Lisa 5 - Kevini tehtud tööd ja eneseanalüüs

Minu roll projekti juures sarnanes eelmisele projektile RIK-is ja võtsin vastutuse arendada administratiivne pool, kasutaja andmete töötlemine ja kasutajate tegevuste logimine. Arendus hõlmas endast tööd nii API-s kui front-end-i kallal.

Tehtud tööd projektis

Kõige rohkem väljakutseid esitas administraatori kasutajate tabeli arendamine. Kasutajate tabelisse vajamineva info kättesaamine läbi erinevate seoste nõudis põhjalikku arusaamist andmebaasis loodud tabelite arhitektuurist ja päringute loomisest API-s.

- Kasutajate tabeli kohta loodud funktsionaalsused:
- Asutuse kasutajate kuvamine
- Asutuse kasutajate otsimine nime järgi
- Kutsete saatmine e-mailidele asutusega liitumiseks
- Kasutajate eemaldamine asutuse nimekirjast
- Kasutaja detailide kuvamine
- Kasutaja kontaktandmete ja sõidukite nime kirja kuvamine
- Kasutaja broneeringute ja parklakohaga seonduva info kuvamine
- Kasutaja logide kuvamine
- Kasutaja logide otsimine aja või kirjelduse järgi

Samuti nõudis suurt tähelepanu ka logide arendamine. Kuna RIK-i projektiga erineb antud projekt selle poolest, et rakendust saavad kasutada samaaegselt mitmed erinevad asutused koos oma kasutajatega, siis kriitilist tähtsust omab andmete eristamine ja et andmed ei satuks valede inimeste kätesse. Kvaliteedi tagamiseks tuleb panustada rohkelt aega erinevate olukordade läbimängimiseks ja testimiseks, et soovitud tulemuses kindel olla.

Toimingud, mida logitakse:

- Rakendusele uue kasutaja registreerimine
- Kasutaja andmete muudatused
- Kasutaja sõidukite lisamised/kustutamised

- Kasutaja broneerimisõiguste muudatused
- Asutuste registreerimised
- Parkimiskoha vabastamised
- Parkimiskoha broneeringud
- Asutuse parkimiskohtade lisamised/kustutamised
- Asutuse parkimiskohtadele peakasutajate määramised/eemaldamised
- Asutusele liikmete lisamised/kustutamised

Kasutaja andmeid saab muuta ainult kasutaja. Administraatoril puuduvad õigused nende andmete muutmiseks ja omab ainult õiguseid kasutajate andmete vaatamiseks. Antud rakenduses on administraatoril kasutaja andmete suhtes ülevaatlik roll. Võrreldes RIK-i projektiga, siis varasemalt sai administraator kasutajaid luua, nende kohta andmeid sisestada ja omas täielikku võimu kasutaja andmete üle. Võttes eelnevalt mainitud nõudeid arvesse, arendasin ma profiililehe, kus kasutaja saab muuta/vaadata oma kontaktandmeid ja sisestada infot sõidukite kohta.

Eneseanalüüs

Kindlasti oli suureks eeliseks antud projekti puhul varasem kogemus ja teadmistepagas kasutatud tööriistadest, raamistikest ja programmeerimiskeeltest. RIK-i projekt andis hea ülevaate võimalikest kitsaskohtadest, mis aitasid saavutada paremat ajaplaneerimist ja mõistmist arendusprotsessist.

Olen väga rahul meeskonnaga, kus koosolekud ja arutelud analüüsi üle olid väga sisukad ja enne arendustöid oli hea nõu küsida, kuidas teised probleemile läheneksid. Olenemata asjaolust, et rakenduse arendus algas täiesti nullist, siis meeskonnana oli alati tunda sihikindlust ja eesmärgipõhist arengut. Millegi uue ettevõtmine nõuab julgust ja mugavustsoonist välja tulemist, seda eriti kohtades, kus tuleb selgeks õppida mõni uus raamistik või kohaneda uuendustega. Infotehnoloogia kiiret arengut sai ka selle projekti puhul tunda ja seda eriti *front-end*'i Reacti raamistiku puhul, kus kasutasime uusimaid disainilahendusi. Projekti algusfaasis oli tunda kiusatus jätkata vana projektiga, kuid õnneks taipasime kiirelt, et mõistlik on alustada puhtalt lehelt võttes arvesse eelmisest

projektist saadud õppetunnid. Puhtalt lehelt alustamine soodustas ka uute lahenduste kasutusele võttu nagu seda on *front-end*'is kasutatav Material-UI disain.

Olen lõpptulemusega väga rahul, võtan saadud kogemused ja teadmised endaga kaasa ja loodan selle projektiga jätkata, et leida reaalseid kliente, kellele see rakendus võiks abiks olla.

Lisa 6 - Gerdi tehtud tööd ja eneseanalüüs

Meeskonnas töötasin nii *back-end'*is ning *front-end'*is. Pealmisteks ülesanneteks oli mõlema projekti ülesseadmine ning struktuuri valik. Arendusest oli minu teha kogu parkimise loogika ning disaini pool.

Tehtud tööd projektis

Minu hooldada oli kogu lehe disain. Veenduda, et asi oleks dünaamiline ning töötaks kõikides seadmetes. Komponentide paigutus oleks korras ning läbi mõeldud ja lahtrid oleksid samal joonel. Idee ning teostus, kuidas administraatori paneel võiks välja näha: kasutada erinevaid vahelehti. Väljakutseks sai kogu lehe ülesseadmine ning turvalisus. Kasutajad ei saaks ligi andmetele, millele neil õiguseid ei ole. Määrata ära vastavad lõpp-punktid ehk *endpoint'*id, mis aadressil, mis info kuvatakse ning kellel sellele vastav ligipääs on. Olulisel kohal oli lehe laadimine ning andmete kuvamine. Esimesel korral, kui leht avatakse, näidatakse kasutajale laadimise ikooni kuniks päringud on edukalt tehtud. Et parandada kasutajakogemust, võtsin kasutusele brauseris küpsised, et jätta meelde kasutaja vaikimisi väärtused. Näiteks sisse logides peab kasutaja esmalt valima asutuse, mis salvestatakse küpsistesse ning järgmisel logimisel tehakse kasutaja eest juba automaatselt.

Spetsiifilisemalt oli minul arendada kogu parkimise loogika, sh vabastamine, koha loovutamine ning broneerimine. Raskeim osa nendest on broneerimise algoritm, mis pidi jälgima vabastatuid kohti ning olemasolevaid broneeringuid. Lisaks kontrolliti, kas vabastatud koht või broneering pole juba tühistatud. Lisaks algoritmidele, tuli kogu informatsioon kuvada kasutajale ning komponentide paigutus teha vastavalt. Vabastamisel, loovutamisel, parklakoha otsimisel ja broneerimisel kasutati erinevaid dialooge, mis tegi lehe üldise disaini puhtamaks. Koha broneerimine on tehtud vastavalt kasutajale konfigureeritavaks, tavakasutajal, kellel koht puudub, pole vaja lisaõigust, kuid peakasutajal, kellel koht olemas, vajab lisaõigust, mille määrab administraator.

Parklakohaga seonduvad tööd:

- Parklakoha vabastamine, loovutamine ning broneerimine
    - Vabastamine – Kontrollitakse, kas parklakoht pole juba antud perioodi vältel broneeritud / vabastatud, kui jah, teavitatakse kasutajat vastavalt olukorrast
    - Loovutamine – Lisaks vabastamise loogikale, kontrollitakse, kas antud kasutajal, kellele parklakoht antakse, pole aktiivseid broneeringuid tehtud
    - Broneerimine – Vabastatud parklakohtade otsimine, nendest juba broneeritud perioodide eemaldamine
- Parklakoha staatus: aktiivne, vabastatud, broneeritud
- Parklakoha ning kasutaja andmete kuvamine tabelis
    - Parklakohaga seonduv informatsioon, kas parklakoht on vabastatud, loovutatud või broneeritud, periood ning vastav kasutaja
    - Kasutaja info kuvamine, tulevased broneeringud

Eneseanalüüs

Meeskonnas töötamine oli super, toimusid pidevad koosolekud ning arutelud, mille tõttu paranes ka meie analüüsi võimekus. Üldiselt arutasime läbi oma tegevused ning mõtted, kuidas asja lahendada. Seejärel jagas igaüks oma ideid kuniks leidsime parima lahenduse, mille valmis tegime. Liikmed teadsid hästi arendusest ning probleemide korral piisas väiksest juhendamisest. Alguses oli raske startida, ei olnud piisavalt motivatsiooni, aga selle eest kui see leiti, tehti kaotatud aeg tagasi.

Projekti suuruse tajumine oli esialgu kehv, seda näitas ka RIK-ile arendatud rakendus. Sellest hoolimata võtsin oma teadmised ning õpitu kokku ja pühendasin seda antud rakenduses. Arenduse käigus järgisin põhimõtet, et mu kood oleks kohe refaktoreeritud ning lisa ajakulu sellele panustama ei pea. Suureks plussiks oli see, et antud rakendus on tehtud täiesti nullist, ehk meie teha oli nii ärianalüüs kui ka arendus. Antud projekti käigus täitsin ka projektijuhi rolli, juhendasin koodis teisi ning aitasin probleemide korral. Olen rahul lõpptulemusega, kus sain kasutada oma õpitut ja teadmisi ning saada aru üldisest arendusprotsessist.