TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Asiman Kasõmov 154854IABB

PARKIMISE HALDAMISE ANALÜÜS JA MOBIILIRAKENDUSTE LOOMINE OÜ MERETIME GRUPP NÄITEL

Bakalaureusetöö

Juhendaja: Karin Rava
MSc. Eng

Tallinn 2019

Autorideklaratsioon

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

Autor: Asiman Kasõmov

20.05.2019

Annotatsioon

Lõputöö põhieesmärk on luua firmale OÜ Meretime Grupp tarkvaralahendus, et täita nende potentsiaalsete klientide infovajadused ja lisaks määratleda ja kirjeldada Parkimise mobiilirakenduse (edaspidi **Rakenduse**) tarkvara loomeprotsessi. Töö täpsemateks eesmärkideks on määratleda rakenduse kasutusjuhud, nende põhjal luua prototüüp ja küsitluse ning «Koridori uurimismeetodi» läbiviimisel saada potentsiaalsetelt klientidelt tagasisidet. Täpsemalt, luua rakendus, mille kaudu kliendid saavad parkimiskohta broneerida ja samuti vaadata läbi rakenduse, kui palju on vabu kohti parklas. Lisaks luua rakendus, mille kaudu parkla omanik saab jälgida parklaga seotud toiminguid (kui palju broneeringuid on antud nädalal, klientide reserveerimise info jne).

Lõputöös lahendatavaks probleemiks on klientide nõuetele vastava informatsiooni puudumine, seega tekib firma poolt nende potentsiaalsete klientide infovajaduste mitte rahuldamine. Parkla omanikul puudub ülevaade parkla tööst ja klientidest, ülevaate saamiseks ta peab alati olema parklas kohal või võtma ühendust parkla turvamehega. Kõikide selliste probleemide lahenduseks võiks olla vastava infosüsteemi loomine.

Samuti lõputöös esitatakse Parkimisrakenduse esialgne kasutajaliidese prototüüp. Pärast prototüübi testimist ning mobiilirakenduse tarkvara arendamist, esitatakse valmisolev kliendi- ja omaniku vaade Rakendus kogu funktsionaalsusega. Soovi korral saab Parkimisrakendust proovida nutitelefonis.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 43 leheküljel, 5 peatükki, 13 joonist, 3 tabelit.

Abstract

Analysis of Parking Management and Creation of Parking Mobile Application by Example of OÜ Meretime Grupp

The aim of the research is to create a software solution for the OÜ Meretime Grupp to meet the informational needs of their potential customers and to give a detailed overview of the creation process of the Parking Mobile Application software. The specific goals of the work include creating a prototype of the parking application and to get feedback from potential clients in conducting the "survey corridor method". Specifically, create an app that allows customers to book parking space and, besides, check out the amount of free parking lots available. In addition, it was necessary to create an application through which the owner of the car park can monitor the activities at the car park (number of bookings per week, customer reservation information, etc.).

The problem to be solved in the thesis is the lack of relevant information that meets the requirements of the clients, so the company fails to satisfy the informational needs of such potential clients. The owner of the car park does not have an overview of the car park's operation and of clients of the parking lot, so, to get an overview he must either be present on-site or contact a car park security officer. Therefore, the creation of a corresponding information system could provide a solution to such problems.

The preliminary user interface prototype of the Parking Application is also presented in the thesis. After testing the prototype and developing the mobile application software, the ready-to-use client and owner's view will be presented with the Parking Function, including the complete functionality. If desired, you can try the app on your smartphone.

The thesis is in Estonian and contains 43 pages of text, 5 chapters, 13 figures, 3 tables.

Lühendite ja mõistete sõnastik

ATI	TTÜ Arvutitehnika instituut
DPI	*Dots per inch*, punkti tolli kohta
API	*Application Programming Interface*, rakendusliides ehk programmiliides
Framework	Tugiraamistik, tugiprogramm
GB	Gigabyte, faili suuruse ühik
GUI	*Graphical User Interface*, graafiline kasutajaliides
Port	Andmesideühenduse lõpp-punkt
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll
HTTPS	*Hypertext Transfer Protocol Secure*, turvaline hüperteksti edastusprotokoll
IP	*Internet protocol address*, internetiaadress, arvutite ja muude arvutivõrgus olevate seadmete omavaheliseks suhtlemiseks
IDE	*Integrated Development Environment*, arenduse tarkvararakendus programmeerijatele
JSON	*JavaScript Object Notation*, lihtsustatud andmevahetusvorming, mis põhineb JavaScripti programmeerimiskeele alamhulgal
CakePHP	Tarkvararaamistik veebirakenduste loomiseks, kirjutatud PHP keeles ja loodud avaliku tarkvara printsiipidel. CakePHP realiseerib «Model-View-Controller» (MVC)
MVC	*Model View Controller*, tarkvara arhitektuurimuster kasutajaliidese rakendamiseks
OS	Operatsioonisüsteem

PHP	*Hypertext Preprocessor,* platvormist sõltumatu programmeerimiskeel
SQL	*Structured Query Language,* struktuurpäringukeel, andmebaasi päringukeel
UI	*User Interface,* kasutajaliides
UML	*Unified Modeling Language,* ühtne modelleerimiskeel
VPS	*Virtual Private Server,* virtuaalne privaatne server
Crontab	*Cron,* klassikaline demon (arvutiprogramm UNIX süsteemi klassides)
Android	*Android,* operatsioonisüsteem nutitelefonide, tahvelarvutite, elektroonraamatute, televiisorite ja teiste seadmete jaoks
Java	*Java, objekt-orienteeritud programmeerimisele.*

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Mobiilrakenduste loomine viimase kümne aasta jooksul hakkas aina rohkem kasvama ja populaarsust saama [1]. Tänu mobiilirakendustele inimestel on võimalik teostada erinevaid tehinguid kodust lahkumata, nii näiteks: teha oste, maksta arveid, broneerida reisi.

Mobiilrakendusi võib jagada kuueks grupiks: elustiili- (fitness, muusika, reisimine), sotsiaalmeedia- (facebook, instagram), üldkasulikud (meeldetuletused, kalkulaator, ilmateade), meelelahutuslikud, tootlikkuse- (docs, wallet/pay), informatiivsed (smartnews, google news) rakendused [2].

Üks üldkasulikest või tootlikustest rakendustest on OÜ Meretime Gruppi parkimise mobiilirakendus, mis annaks klientidele parkimiskoha broneerimise võimaluse. Lõputöö üheks ja peamiseks eesmärgiks on luua mobiilirakendus, läbi mille saaks parkimisfirma tegutsemist muuta efektiivsemaks nii klientide seisukohast kui ka parkla omaniku seisukohast. Nimetatud parkimisfirmal puudub infosüsteem, läbi mille kliendid võiksid saada infot parkla kohta: Kas on parklas praegusel hetkel vabu kohti? Kas võib parkimiskohta ette broneerida? Kõikide nende infovajaduste rahuldamiseks peab klient helistama parkla turvamehele, mis äri poolest ei ole nii efektiivne ja ülevaatlik.

Parkimisfirma üks põhieesmärke peab olema klientidele vajaliku informatsiooni pakkumine nii kiirelt ja arusaadavalt kui on võimalik, kuna sellest otseselt sõltub firma käive.

Läbi rakenduse loomise püütakse lihtsustada klientide ja omaniku tööd. Klientide poolest nii, et nendel tekib võimalus läbi rakenduse parkimiskohta ette broneerida ja samuti vaadata, palju on parklas vabu kohti. Omaniku poolest lihtsustatakse tööd nii, et tekib nüüd võimalus jälgida teostatavaid toiminguid, siia kuulub võimalus vaadata, palju vabu ja hõivatud kohti on parklas, hõivatuse statistikat, vaadata klientide infot (kliendi nimi, telefoninumber, email, autonumbrid).

Lõputöö eesmärkideks on analüüsida rakenduse funktsionaalsust, hankida tagasisidet, määratleda rakenduse arhitektuuri, esitada loomeprotsessi läbi infosüsteemi detailse analüüsi ja kirjeldamise, esialgse prototüübi loomine ning testimise läbi viimise abil lõpliku töötava rakenduse valmistamine.

Antud bakalaureusetöö teises peatükis esitatakse Rakenduse üldine funktsionaalsus, esialgne kasutajaliidese prototüüp ning esitletakse kasutusjuhtude- ja tegevusdiagrammid. Samuti teises peatükis fokuseerib autor uurimismeetodite kasutamise tähtsuses ja eelistustes tagasiside saamiseks, täpsemalt räägitakse mis uurimismeetodid olid kasutatud ja mis tulemused olid saadud. Kolmandas peatükis esitletakse Rakenduse lõplikud vaated pärast algset prototüüpi testimist ja tagasiside saamist ning tarkvara arendamist, andmebaasi füüsiline mudel ning tuuakse välja arhitektuuri joonis. Neljanda peatüki fookuseks on serverisüsteemi valiku ja kasutatavate tööristade kasutuse kirjeldamine koos koodinäidetega, andmebaasisüsteemi kirjeldamine, andmevahetuse ning samm-sammuline loomeprotsessi lahti kirjutamine.

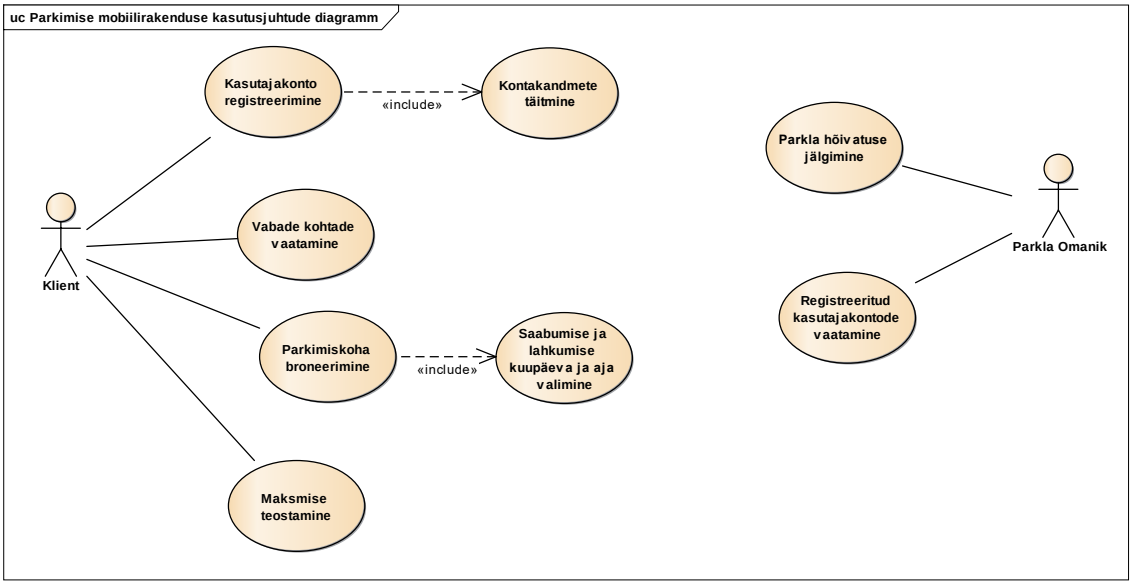
2 Parkimise mobiilirakenduse analüüs

Käesolevas peatükis analüüsib autor rakenduste kasutajate funktsionaalsust läbi üldise funktsionaalsuse kirjelduse ja kasutusjuhtude ja tegevusdiagrammide.

Rakenduse üldise funktsionaalsuse kohaselt, mille baasil on loodud ka kasutuslood, saab klient kasutajakontot luua. Pärast konto loomist kliendile antakse võimalus vaadata parklas olevate vabade kohtade arvu ja teostada parkimiskoha broneeringut. Süsteem salvestab klientide andmeid ja broneeringute ajalugu. Kliendil on võimalik teenuse eest tasuda mobiiliga. Kliendil on võimalik igal ajal oma broneeringut lõpetada ja tulevast broneeringut tühistada. Parkla omanikul on võimalus vaadata kasutajate kasutajakontosid (nende andmeid) ja jälgida parklas hõivatuse statistikat nädalate kaupa.

2.1 Rakenduse UML diagrammid

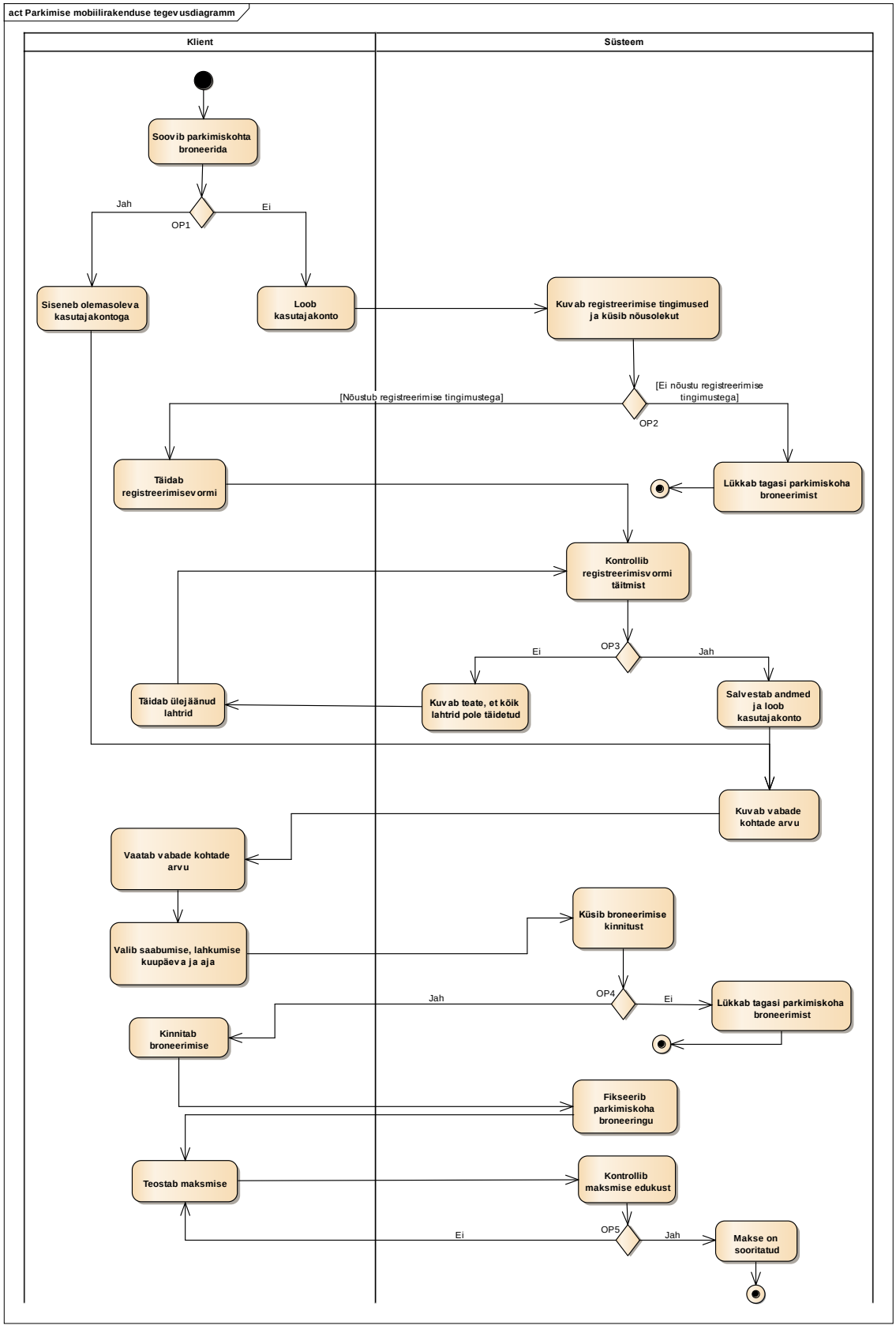
Käesolevas alapeatükis esitab autor kasutusjuhtude diagrammi (Joonis 1) ning tegevusdiagrammi (Joonis 2). Diagrammid on koostatud Enterprise Architect'i tarkvara abil. Kasutusjuhtude diagrammi dokumentatsioon on esitatud Lisas 1 Tabel 4 ning tegevusdiagrammi dokumentatsioon asub Lisas 2 Tabel 5.



Joonis 1. Parkimise mobiilirakenduse kasutusjuhtude diagramm

Joonisel 1 esitatud rakenduse kasutusjuhtude diagramm esitab süsteemi tööd kliendi ja parkla omaniku vahel. Kliendil tuleb parkimiskoha broneerimiseks teostada mitu tehingut, parkla omanikul on funktsionaalsus parklas toimuvate tehingute jälgimiseks.

Joonisel 2 esitatud rakenduse kasutuse tegevusdiagramm esitab parkimiskoha broneerimist algusest lõpuni kliendi seisukohast. Kliendi ja rakenduse ehk süsteemi tegevused on eraldi välja toodud.

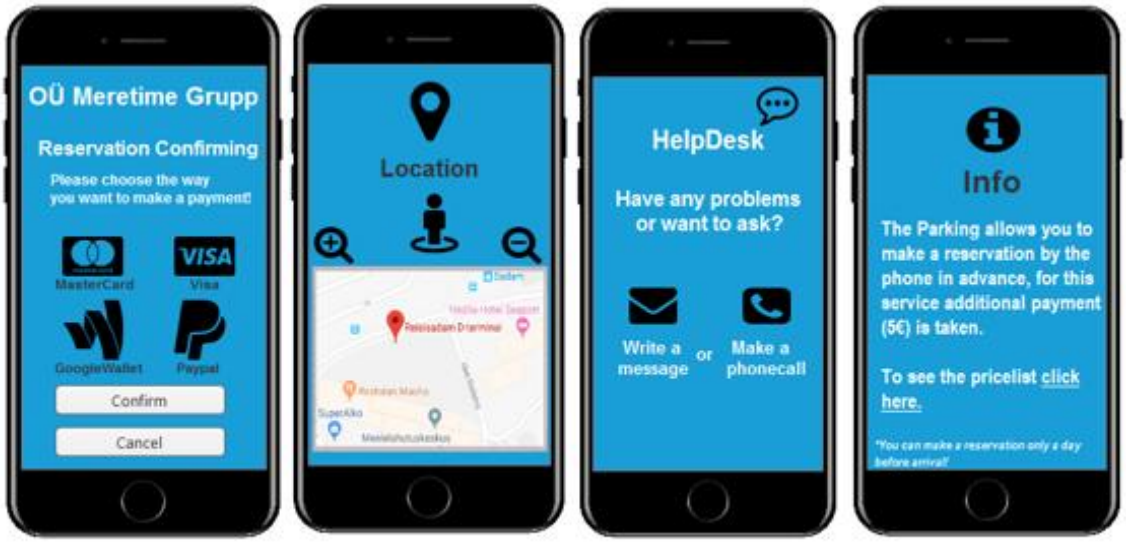


Joonis 2. Parkimise mobiilirakenduse tegevusdiagramm

2.2 Rakenduste kasutajaliidese esialgne prototüüp

Antud peatükis on esitatud kliendi ja omaniku rakenduse esialgne prototüüp, mis on loodud Axure tarkvara abil. Järgnevalt on esitatud kliendirakenduse prototüübi vaated.

Joonis 3. Parkimise mobiilirakenduse neli vaadet: pealeht (a), parkimisperioodi valimine (b), parkimiskoha valimine (c), broneeringu info (d)



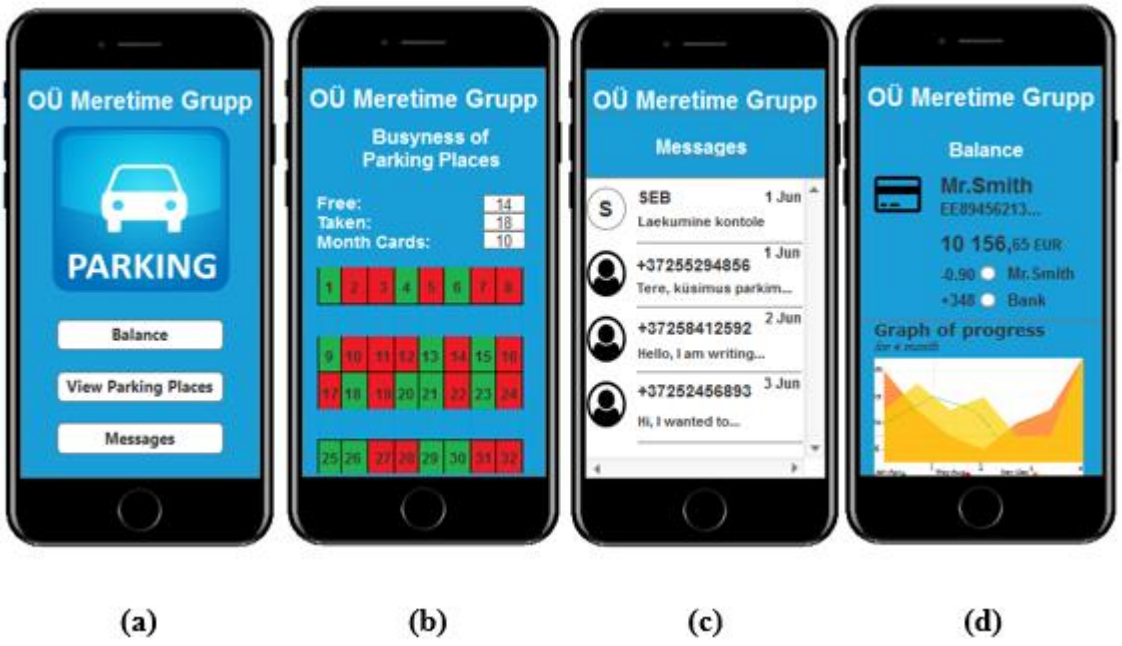
(e) (f) (g) (h)

Joonis 4. Parkimise mobiilirakenduse neli vaadet: broneerimise kinnitamine (e), asukoha leht (f), klienditufi (g), info ja hinnakirja leht (h).

Jooniselt 3 ja 4 on näha neli kliendirakenduse kasutajaliidese vaadet. Kasutajaliidese vaade (a) on esimene vaade, mida klient rakendust käivitades näeb. Teisel vaatel (b), klient valib soovitud saabumise ja lahkumise kuupäeva ning kellaja. Kolmas vaade (c), kus klient näeb, palju on parklas vabu kohti, mis kohad on hõivatud ja võib valida soovitud parkimiskohta. Neljas vaade (d), kus klient saab parkimiskoha kinnitust (valitud kuupäevade vahemik, määratud kellaeg, parkimiskoha number, hind).

Klient saab maksta broneeringu eest talle soovitud maksmisviisil, seda esitab vaade (e). Täiendava informatsiooni saamiseks klient saab vajutada kolm lisaikooni, asukoha teada saamiseks on eraldi vastav nupp, vaade (f). Vajadusel, kui klient vajab abi, tema saab helistada või esitada oma küsimust klienditoesse, seda esitab vaade (g) ja informatsiooni saamiseks parkimise kohta, on olemas info ikoon, vaade (h).

Parkla omaniku mobiilirakenduse vaade:



Joonis 5. Parkimise rakenduse omaniku kasutajaliidese vaated: peamenüü (a), parkla vabad ja hõivatud kohad (b), klientide sõnumid (c), parkimise jooksev saldo (d).

Joonisel 5 on esitatud neli omaniku rakenduse vaadet. Esimene vaade (a) on parkla omaniku rakenduse esileht, mille peal on kolm nuppu. Bilansi nupu vajutamisel kasutajale antakse võimalus jälgida parkimise jooksvat saldot, seda esitab vaade (d), parkimiskohtade hõivatuse vaatamiseks kasutaja vajutab parkimiskohtade vaate nuppu, mis samuti näitab graafiliselt, mis parkimiskohad on hõivatud, seda esitab vaade (b). Samuti parkla omanik saab kuvada klientide sõnumeid, seda esitab vaade (c).

2.3 Kasutatud uurimismeetodid tagasiside saamiseks

Mobiilirakenduse loomisel on tähtis koguda maksimaalselt palju tagasisidet potentsiaalsetelt kasutajatelt ja selleks valida sobivad meetodid. Uurimismeetodi valimisel on hea meeles pidada, et millisel moel oleks kõige efektiivsem saada tagasisidet potentsiaalsetelt klientidelt, et edaspidi rakendust parandada ja arendada vastavalt klientide soovitustele. Üks sellistest meetoditest on „Koridori meetod“, mille käigus küsitakse koridoris viibijatelt nende hinnangut antud mobiilirakenduse prototüübi kohta [3]. „Koridori meetodi“ eeliseks on võimalus anda klientidele loodud prototüüpi kohe katsetada ja otseselt anda rakenduse arendajale tagasisidet. Parkimisrakenduse lõplikuks teostamiseks viidi läbi uuring „Koridori meetod“, mille käigus intervjueeriti seitset

Tallinna Tehnikaülikooli üliõpilast. Nendele esitati mobiilirakendusega seonduvaid erinevaid küsimusi:

- Kas nad oleksid huvitatud parkimise broneerimise rakendusest?
- Kuivõrd lihtne on rakenduse kasutamine nende jaoks?
- Kas nad sooviksid näha rakenduses täiendavaid funktsioone?
- Kuidas nad hindavad rakenduse välist ilmet?

Saadud vastused olid järgmised: kõik osalejad vastasid, et rakendust on kerge kasutada ja parkimiskohtade broneerimisel raskusi ei tekkinud, liides on lihtne ja selge. Parkimise eest tasumiseks on mitmeid viise ja eeliseks on küsimuste tekkimise korral võimalus võtta otse ühendust klienditeenindusega, kasutades selleks rakendust.

Suurema vastajate ringi kaasamiseks viidi läbi 19 osalejaga uuring. Küsimustikus paluti vastata kahele küsimusele ja anda hinnang esialgsele prototüübile. Küsimused ja hinnangu andmise võimalused olid järgmised:

- Kas olete kasutanud parkimiskohtade broneerimise rakendust?
- Hinnake rakenduse mugavust 5-punkti skaalal?
- Hinnake prototüübi disaini.
- Hinnake rakenduse mugavust parkla omanikule.
- Hinnake rakenduse disaini parkla omaniku vaatenurgast.
- Kas nad sooviksid näha rakenduses mõnda lisafunktsiooni?

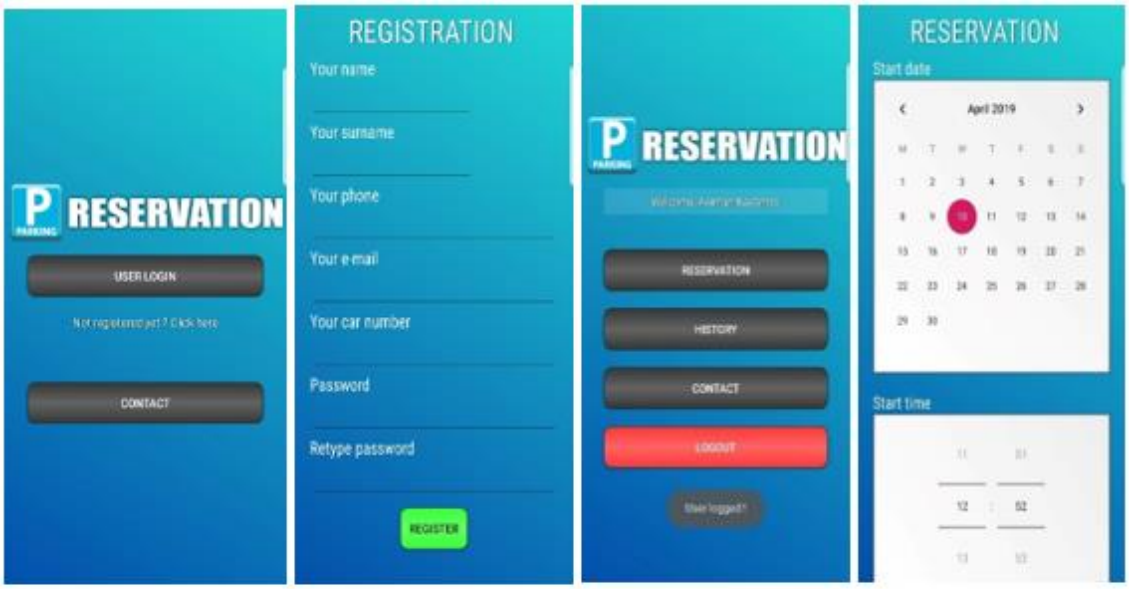
Küsimuste ja vastustega saab täpsemalt tutvuda Lisas 3. Vastused olid järgmised: enam kui pooled vastasid, et pole broneerimise rakendust kasutanud. Enamik vastanutest hindas üsna kõrgelt esialgset prototüüpi. Lisafunktsioonidena nimetati: boonussüsteemi loomine rakenduse sagedastele kasutajatele (püsikliendid) ning parkimise lõpuni jäänud aja kuvamist.

3 Parkimise mobiilirakenduse disain

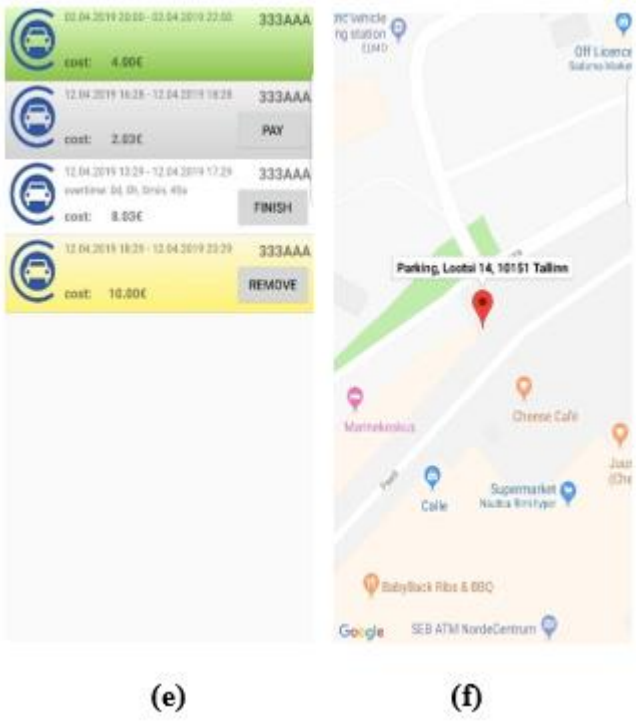
Käesolevas peatükis esitab autor Rakenduse lõplikud vaated, andmebaasimudeli ning toob illustreeriva arhitektuurse joonise erinevate kihtide andmevahetusest.

3.1 Rakenduse lõplikud vaated

Selles peatükis on esitatud Rakenduse lõplikud vaated, mida klient rakendust käivitades oma nutitelefoni ekraani peal näeb.



(a) (b) (c) (d)



Joonis 6. Lõpliku parkimise mobiilirakenduse kuus kliendivaadet: esileht (a), kasutajakonto registreerimisvorm (b), kliendivaade pärast konto loomist (c), parkimiskoha registreerimisvorm (d), broneeringute ajalugu (e), asukoha leht (f).

Joonisel 6 on esitatud kuus lõplikku kliendivaadet. Esimene vaade (a) on kliendirakenduse esileht, mida klient näeb rakendust käivitades [4]. Kliendirakendus pakub siseneda süsteemi olemasoleva kontoga või juhul kui kasutaja pole kasutajaokontot loonud üle minna registreerimislehele, vaade (b). Kolmas vaade (c) on kliendirakenduse menüü, mida klient näeb, siis kui ta on kasutajakonto loonud. Pärast kasutajakonto loomist saab klient autoriseerida enda loodud parooliga ja teostada broneerimist, seda esitab vaade (d). Kasutaja saab vaadata enda broneeringute ajalugu. Broneeringute ajaloos kuvatakse: rohelise värviga makstud broneering, halli värviga on tähistatud mitte makstud broneering, valge värviga kehtiv broneering ja kollase värviga tuleviku broneering, seda esitab vaade (e). Parkla asukoha teada saamiseks klient saab vajutada nuppu «Contact», kus talle avaneb asukoht kaardi peal, vaade (f).

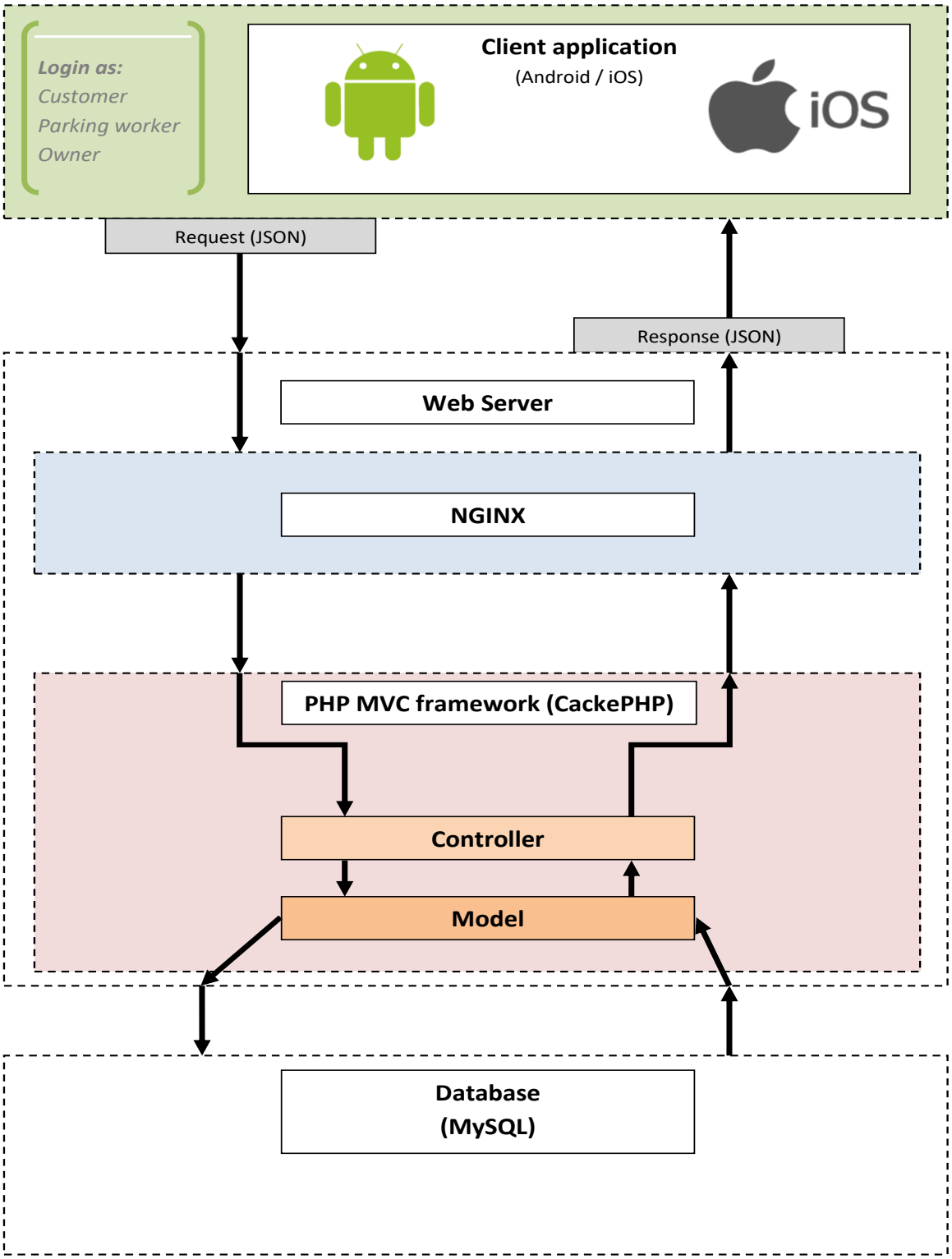
Parkla omaniku mobiilirakenduse lõplikud vaated:

Joonis 7. Lõpliku parkimise mobiilirakenduse neli omanikuvaadet: sisselogimisvorm (g), peamenüü (h), kasutajakonto registreerinud kliendid (i), hõivatuse statistika (j).

Joonisel 7 on esitatud neli lõpliku Rakenduse omaniku vaadet. Esimest sisselogimise vaadet (g) näeb nii klient kui ka parkla omanik. Omanik siseneb süsteemi enda kasutajakontoga. Pärast sisselogimist omanikule avaneb parkla halduri menüü, vaade (h). Omanikurakenduses saab näha kõikide registreerinud kasutajate infot (nimi, perekonnanimi, auto number, email, telefoninumber), seda esitab vaade (i). Omanikul on võimalus jälgida parkla tööhõivatust nädalate kaupa, seda esitab vaade (j).

3.2 Rakenduse arhitektuur

Joonisel 8 on esitatud Rakenduse arhitektuur. Rakendus on arhitektuuriliselt jagatud nelja kihti – kliendirakendus, veebiserver, loogikakiht ning andmebaasikiht.



Joonis 8. Parkimise mobiilirakenduse arhitektuur.

Kliendirakendus vormistab päringu JSON vormingus ja saadab selle spetsiaalse URL aadressi kaudu laiali küsides domeeni (meie puhul www.parking.ee) pordi 80 kaudu (kui kasutatakse kaitstud SSL ühendust, siis port 443). Serverisse on paigaldatud API, mis

töötab kliendirakendusega. Seda tööd teeb NGINX server. Porte kuulates ja päringut vastu võttes teeb server ette antud töö ära ja genereerib kliendirakendusele asjakohase vastuse.

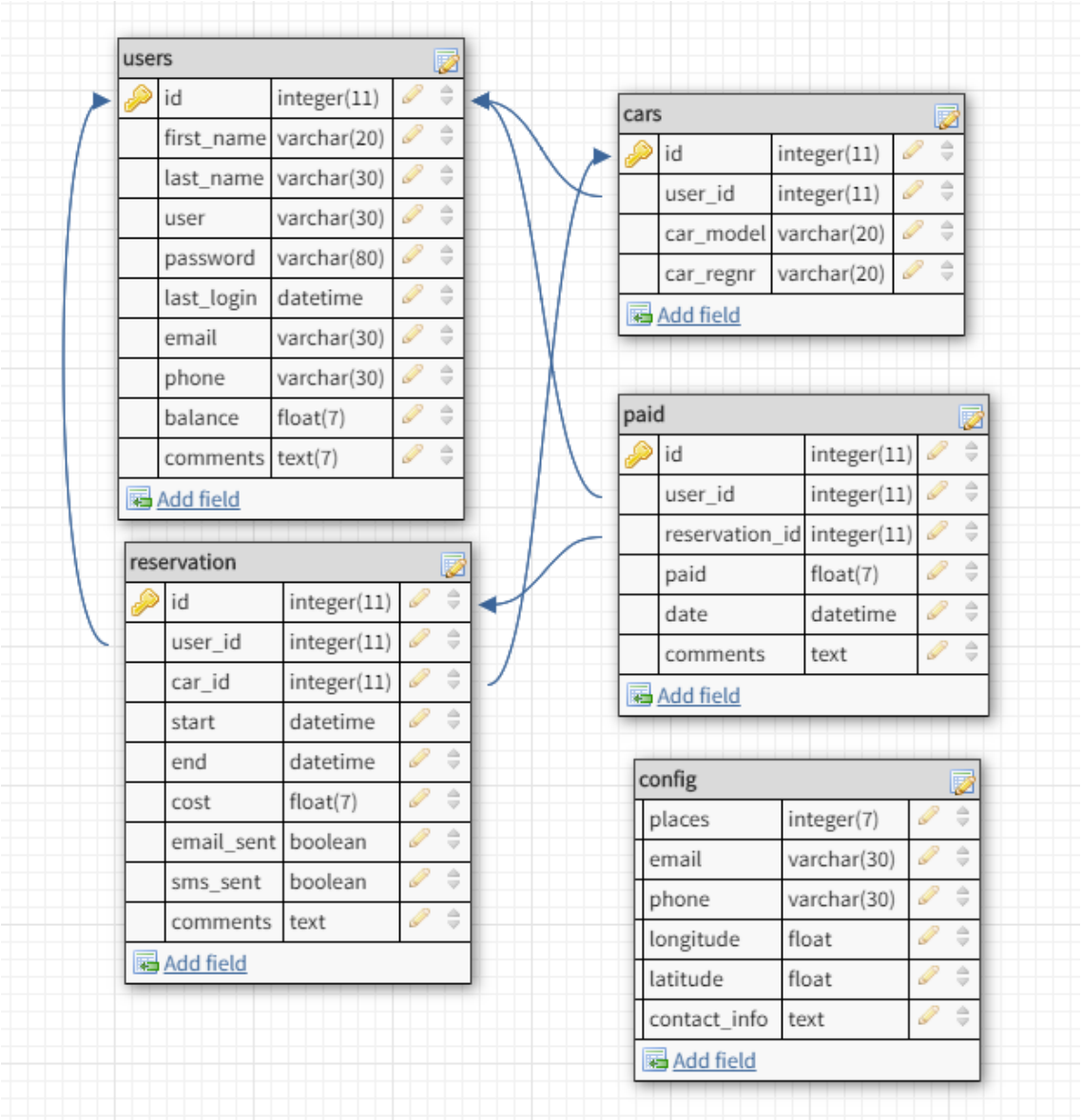
Moodustunud URL-i põhjal määrab meie API (antud juhul CakePHP raamistikus töötav) kindlaks, millist kontrollerit päringu töötlemiseks kasutada ja millist loogikat rakendada. Näiteks kui see saab päringu kujul parking.ee/login, kasutatakse registreeritud kontrollerit LoginController, mis sisaldab antud päringule vastavat loogikat.

Nii näiteks saab kontroller kontrollida sisestatud andmete paikapidavust, näiteks e-aadressi (peab sisaldama järjestikusi sümboleid) või telefoninumbrit. Kui valideerimine õnnestub, teeb kontroller vajalikud toimingud (kasutaja lisamine, tema andmete muutmine jne) ja saadab õiged andmed tagasi kliendirakendusse.

Kontroller omakorda suhtleb rakenduse andmebaasiga, mis asub serveris mudeli kaudu. Mudelid loevad, lisavad, muudavad ja eemaldavad teavet andmebaasist. Samuti kindlustavad need andmete, näiteks andmebaasi seotud tabelite võtmete tervikluse.

3.3 Rakenduse andmebaasimudel

Joonisel 9 on esitatud Rakenduse andmebaasi füüsiline mudel:



Joonis 9. Parkimise mobiilirakenduse andmebaasimudel.

Järgnevalt on esitatud Rakenduste andmetabelite kirjeldus:

Tabel 1. Parkimise mobiilirakenduse andmetabelite kirjeldus.

Tabeli nimetus	**Tabeli kirjeldus**
users	Tabel, milles hoitakse kõikide kasutajate loetelu ja andmed
cars	Autode loetelu, mis on kasutajatel kasutuses. Ühel kasutajal võib olla mitu autot.
reservation	Parkimise broneering.

paid	Kasutajate parkimiskohtade tasude tabel.
config	Parkla info tabel. Parkimiskohtade üldarv, asukoht ja üldinekirjeldus.

4 Parkimise mobiilirakenduse realisatsioon

Käesolevas peatükis kirjeldab autor olemasolevaid serverisüsteeme, toob ülevaate vajalikudest tööristadest, kasutatud andmebaasisüsteemist ning räägib andmete liikumisest rakenduse süsteemis. Samuti autor esitab Rakenduse samm-sammulist loomeprotsessi.

4.1 Serverisüsteem

Selles peatükis annab autor ülevaate Rakenduse kasutuselevõtuks vajaminevatest serverisüsteemidest ja toob teenuste pakkujate võrdluse.

4.1.1 Füüsiline server (nn püsiserver)

Füüsiline server on, nagu nimigi ütleb, server (füüsiline arvuti), mis töötab, nagu ka iga teine arvuti, nii Windows kui ka Linux operatsioonisüsteemiga. Füüsilised serverid on praktiliselt kõikjal. Tegemist on lauaarvutitega, millel on suur hulk täiustusi, mida tavalisel lauaarvutil ei ole ja millel on sellised asjad nagu koondatud toiteplokke, RAID-kontrollereid *(Redundant Array of Independent Disks)*, mitut võrgukaarti jne. Füüsilised serverid on tervikuna palju võimsamate komponentidega. Kõik osad vajavad serveriruumis eraldi kohta. Enamikul serveritest on ka kaks või enam füüsilist protsessorit, millel mõlemal on mitu südamikku [5].

4.1.2 Virtuaalserver (VPS või virtuaalne masin - VM)

Virtuaalserver on spetsiaalses tarkvaras loodud virtuaalne objekt, millel on samad omadused kui tavalisel füüsilisel serveril. Virtuaalserverit saab kasutada mistahes füüsilises serveris.

Ühes füüsilises serveris saab korraga käivitada mitut virtuaalserverit oma unikaalse konfiguratsiooniga, eraldi haldusega ja omaenda protsessidega. Iga kasutaja saab koos virtuaalserveriga ka kindla koguse süsteemiressursse, mis määravad protsessi võimsuse ja eraldatava RAM-i *(Random Access Memory)* hulga. Piirangud kehtivad ainult eraldatud süsteemiressursside, andmebaaside, veebilehtede ja nendele juurdepääsu omavate kasutajate arvu suhtes.

Virtuaalserveri rakendamise käigus saab kasutaja muuta oma seadeid, paigaldada vajalikke tarkvaratooteid [6].

4.2 Teenuste pakkujate võrdlus ja serverisüsteemi paigaldus

Lõputöö eesmärgiks on luua infosüsteem, mis hõlbustaks klientide ja parkimisteenuste koostoimet, looks mugavuse parkla omanikule ja klientidele ning tagaks kasumlikkuse. Seetõttu on üks peamistest eesmärkidest vähendada Rakenduse kulusid ja saada soovitud ressursse ja suutlikkust. Tuginedes eelnevale alapeatükile, valikuks osutus virtuaalserver, mitte füüsilise serveri üürimine. Edaspidi kaalusime ainult neid pakkumisi, mis vastasid meie poolt suutlikkusele ja ressursside määratud kriteeriumidele.

Uurides ülesande täitmiseks vajaminevat energia ja ressurssi kulu, otsustasime, et vajame umbes 1Gib andmete salvestamiseks, umbes 100Mib tarkvara jaoks, samuti Crontab'i tuge, vähemalt kaks aktiivset ülesannet.

Otsustavaks teguriks oli ka andmebaasi haldamise mugavus (Zone.ee-s kasutati PhpMyAdmini halduspaneeli) ja juurdepääs serveri failisüsteemile FTP- protokolli kaudu.

Võrdluseks võttis autor kolm Eestis populaarset VPS-teenuse pakkujat. Andmed on asjakohased 01.02.2019 seisuga.

Tabel 2. Virtuaalserveri pakkujate võrdlev tabel.

Pakkuja	Mälu veebirakenduseks	Andmebaasi mahutavus	Crontab ülesannete toetus	Maksumus EUR / month
Zone.ee [7]	512 MiB	4GiB	2	4.80 + KM
veebimajutus.ee [8]	piiramatult	5 GiB	5	5.20 + KM
Digital Ocean [9]	512 MiB	3 GiB	2	5.00 + KM

Kõik kolm teenusepakkujat vastavad meie nõuetele (vajame umbes 1 Git andmete salvestamiseks, umbes 100Mib tarkvaraosale, samuti toetust Crontabile, vähemalt kaks aktiivset ülesannet). Hetkel on kasutuses kaks süsteemi – pilve- ja füüsiline süsteem. Autor otsustas teenusepakkuja Zone.ee pilveserveri süsteemi kasuks. Pilvesüsteem on füüsilise serveri rentimisest märksa odavam ja sellesse on juba paigaldatud kõik tööks vajalikud teenused (Apache, FTP, PHP, PHPMyAdmin jne).

Seega otsustasime valida Zone Media (zone.ee), kuna nendel oli kõikidest odavaim pakett. Eeliseks on ka nende väga mugav administreerimispaneel ja 24/7 režiimis töötav tugi.

4.3 PHP tööriistad

Rakenduse serveri osa on kirjutatud PHP keeles. PHP on levinud avatud lähtekoodiga programmeerimiskeel, mis on konstrueeritud spetsiaalselt veebirakenduste arendamiseks [10].

Rakenduse põhiloogika pandud PHP koodi – näiteks transaktsioonide säilitamine ja lugemine andmebaasist.

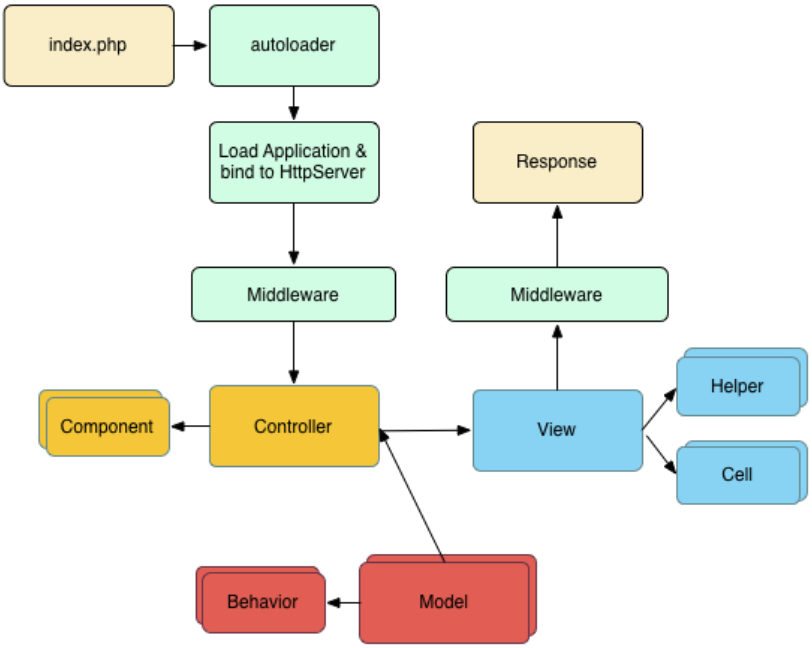
PHP-ga mugavaks töötamiseks välja töötatud erinevad raamistikud (tugiraamistik rakenduste loomiseks PHP keeles), mis lihtsustavad tööd andmebaasidega, töötlevad kasutaja päringuid ja kuvavad tulemuse genereeritud HTML lehe kujul või ettevalmistatud vastuse struktuursete andmete näol. Üheks selliseks raamistikuks on CakePHP.

CakePHP — see on tarkvararaamistik veebirakenduste loomiseks, mis on kirjutatud PHP-s ja põhineb avatud lähtekoodiga tarkvara põhimõtetel. CakePHP realiseerib mustrit „Mudel-Vaade-Kontroller“ (MVC) [11].

Vaatleme MVC kontseptsiooni:

- **Model**
  Mudel esindab rakenduse osa, mis realiseerib äriloogikat. See vastutab andmete vastuvõtmise ja nende konverteerimise eest vastavalt rakenduse kontseptsioonile. See hõlmab töötlemist, valideerimist, ühendusi ja muid andetöötluse ülesandeid.
- **View**
  Vaade loob Mudelist saadud andmete visuaalse kujutise. Eksisteerides Mudeli objektidest eraldiseisvalt, vastutab see talle kättesaadava teabe kasutamise eest kasutajaliidese loomiseks rakenduse tarbeks.
- **Controller**
  Kontroller töötleb kõiki kasutajate päringuid, olles siduvaks lüliks Vaate ja Mudeli vahel [12].

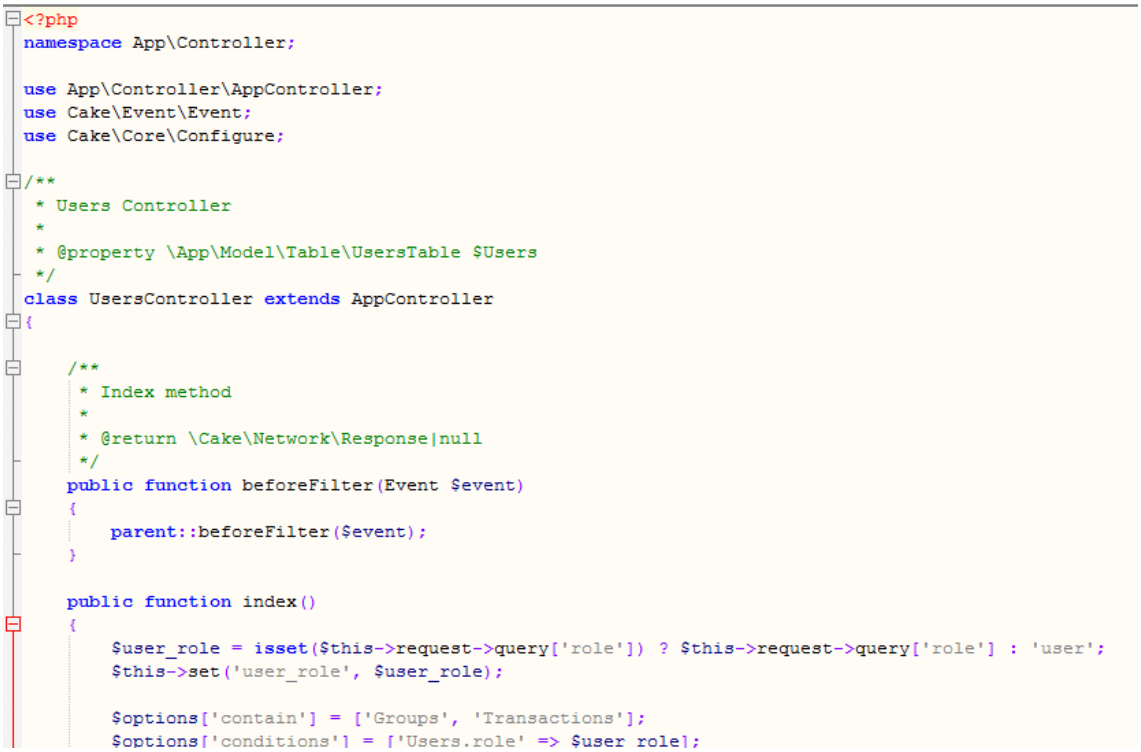
Alltoodud joonisel on esitatud täistsükkel päringute töötlemiseks CakePHP's.



Joonis 10. Päringute töötlemine CakePHP-s [13].

CakePHP-ga töötamise mugavuseks, me hakkame kasutama arendusplatvormi PhpStorm [14]. Rakenduse serveri osa on kirjutatud PHP programmeerimise keeles ja see arendusplatvorm on tihedalt sellega integreeritud. PhpStormi-s on realiseeritud hea kontekstiabi ja koodi automaattäitmise süsteem .

Alltoodud on kontrolleri koodi osa, mis vastutab andmebaasi «users» tabeli päringute eest. Selles koodis toimub ettevalmistus nimekirja loomiseks kõigist tabelis olevatest kasutajatest.



```php
<?php
namespace App\Controller;

use App\Controller\AppController;
use Cake\Event\Event;
use Cake\Core\Configure;

/**
 * Users Controller
 *
 * @property \App\Model\Table\UsersTable $Users
 */
class UsersController extends AppController
{

    /**
     * Index method
     *
     * @return \Cake\Network\Response|null
     */
    public function beforeFilter(Event $event)
    {
        parent::beforeFilter($event);
    }

    public function index()
    {
        $user_role = isset($this->request->query['role']) ? $this->request->query['role'] : 'user';
        $this->set('user_role', $user_role);

        $options['contain'] = ['Groups', 'Transactions'];
        $options['conditions'] = ['Users.role' => $user_role];
```

Joonis 11. Kontrolleri kood PHP keeles, mis vastutab andmete päringute eest tabelis "Users".

4.4 Android tööriistad

Rakenduse kliendiosa on kirjutatud Java keeles Androidi operatsioonisüsteemi [15] tarbeks Android Studio [16] arenduskeskkonna kasutamisega. Androidi operatsioonisüsteem on paigaldatud enam kui 80%-le kõikidest kaasaegsetest nutitelefonidest ja tahvelarvutitest [17], kus saab käivitada ka meie kliendirakenduse.

Kuna kliendiosa on kirjutatud Androidi operatsioonisüsteemi jaoks, arendame rakendust Android Studio arenduskeskkonna abil. Rakendus on kirjutatud Java programmeerimiskeeles ja antud keskkonnas on sellele keelele väga hea tugi. Mugava arenduse jaoks on olemas kontekstitundlik abi, koodi autotäitmine ja palju integreeritud tööriistu.

Alljärgneval joonisel on esitatud koodinäide, mis arvutab vabade kohtade arvu antud ajaperioodiks.

Joonis 12. Java kood, mis arvutab vabade kohtade arvu antud ajaperioodiks.

Ülaltoodud Android Studio keskkonnas kirjutatud Android-kood kirjeldab funktsiooni, mis otsib broneerimise kuupäevade ja kellaaegade kokkulangevusi eelmiste broneerimiste maatriksiga. Vastete leidmisel suurendab see muutujat, mis salvestab leitud vastete arvu. Funktsiooni lõpus lahutatakse see väärtus parkimiskohtade koguarvust.

4.5 Rakenduse andmebaasisüsteem

Andmebaase kasutatakse rakenduse puhul klient-server tüüpi töö korraldamiseks ja töötamiseks rakendustega, mis suhtlevad omavahel vormitud struktureeritud päringute kaudu.

Andmebaasi aluseks valis autor andmebaaside haldamise süsteemi MySQL, mille eelisteks on lai levik, väga hea tehniline tugi, optimeeritud kiirus ja kasutamise lihtsus vähese teabemahu puhul.
MySQL on kõige paremini kohandatud kasutamiseks DBMS-is (andmebaasi ohjesüsteem). Kliendi rakenduste käivitamiseks pakub enamus serveriteenuse pakkujaid nii arvuti kui ketta ressursse väikeses koguses. Seepärast on antud kasutuseks vaja ülitõhusat ja samas töökindlat DBMS-i (enamus veebirakendusi ja -lehekülgi peab töötama ööpäevaringsel režiimil).

MySQL peamised eelised:

- Lõimtöötlus, mitme samaaegse päringu tugi;

- Kirjutatud madala taseme, masinakoodile võimalikult lähedastes keeltes;
- Paindlik privileegide ja paroolide süsteem;
- Paindlik numbrivormingute, muutuva pikkusega stringide ja ajatemplite tugi;
- Liides keeltes C, Perl ja PHP;
- Töötab paljudel erinevatel platvormidel;
- Kiire töö, mastaabitavus;
- Enamusel juhtudest tasuta;
- Korralik majutusteenuste pakkujate tugi [18].

Rakenduses kasutatakse vajalikke funktsioone tööks andmebaasidega (SELECT, INSERT, UPDATE, DELETE jne) andmete lugemiseks, värskendamiseks ja hoolduseks, näiteks:

Tabel 3. Andmebaasi funktsioonide tabel.

Funktsiooni nimetus	**Näide, kirjeldus**
SELECT	SELECT * FROM users WHERE id=1 *Tagastab kõik andmete read kasutaja kohta koos identifikaatoriga number 1.*
INSERT	INSERT INTO cars (user_id, car_model, car_regnr) VALUES (1, 'Citroen c4', '111 AAA') *Lisab salvestuse tabelisse cars kasutaja jaoks identifikaatoriga number 1 ja väärtusega: automudel: Citroen c4 ja registrinumber: 111 AAA*
UPDATE	UPDATE users SET car='345 ABC' WHERE id=1 *Määrab autonumbri kasutajale indetifikaatoriga number 1*
DELETE	DELETE FROM users WHERE id=1 *Kustutab salvestuse kasutaja kohta identifikaatroiga 1*

4.6 Andmete liikumine rakenduse süsteemis

Andmete edastamiseks kliendi-serveri rakenduste vahel kasutatakse laialdaselt tekstivorminguid nagu loetavad JSON ja XML toetades loomist, lugemist ja dekodeerimist reaalsetes rakendustes. Mõlemad on hierarhilised ja sõltumatud andmevahetuse teksti sümbolitest. JSON on JavaScripti põhine teksivahetuse formaat. Nagu ka paljud teised tekstivormingud, on JSON-i lihtne lugeda ka inimestel.

Parkimise mobiilirakenduses edastatakse ja võetakse vastu kõik andmed JSON-vormingus. Antud formaati on lihtne lugeda nii JavaScripti kui ka PHP-ga.

Tänu oma kompaktsusele võrreldes XML-ga, võib JSON-formaat olla sobivam keerukamate struktuuride (nimi-väärtus paaride kogum ja järjestatud väärtuste nimekiri) realiseerimiseks. Kui rääkida veebirakendustest, on see antud kujul sobilik brauseri ja serveri vaheliseks andmevahetuseks (AJAX), nagu ka serverite vaheliseks vahetuseks (HTTP programmiliidesed) [19].

Järgnevalt on esitatud Rakenduse JSON-i kasutajate ja reserveerimise koodi näited:

```
user JSON sample
{
        'id':1,
        'name':'Mark',
        'surname':'Tamm',
        'phone':'+372 5553311',
        'email':'mark@hot.ee',
        'car':'222AAA',
        'password':'123456',
        'role':'user'
}

reservation JSON sample
{
        'id':0,
        'userid':1,
        'start':1554224400000,
        'end': 1554231600000,
        'cost':4,
        'car':'222AAA',
        'finish':true,
        'payd':true
}
```

Joonis 13. JSON-i kasutajate ja reserveerimise kood.

4.7 Rakenduse samm-sammuline loomiseprotsess

Rakenduse elluviimine nõuab erinevate programmeerimistehnoloogiate ja tööriistade integreerimist. Erinevatest rakendustest loodud kliendi-serveri süsteem peab omavahel suhtlema ja tegema eksimatult kõik õigeks omavaheliseks tööks vajalikud operatsioonid. Edasi esitatakse samm-sammult, kuidas projekt loodi ning milliseid tehnoloogiaid ja tööriistu kasutati.

1. Rakenduse väljatöötamise esimene etapp on domeeni registreerimine ja virtuaalserveri genereerimine. Harilikult käib see lihtsalt ja võtab aega umbes tunni. Virtuaalserveri loob internetiteenuse pakkuja automaatselt.

2. Edasi tuleb virtuaalserveri haldusmoodulis lisada kasutaja FTP protokolli kasutamiseks, et vajalikud failid virtuaalserverisse üles laadida. Määrame sellele kasutajale kaugkausta, mis on protokolliga töötamise ajal tema jaoks juurkaust. Määrata tuleb, et kasutaja hakkab töötama kaustaga httpdocs, mis sisaldab virtuaalserveris indeksfaili (index.php), mida teenusepakkuja vaikimisi kasutab.
3. Lisame andmebaasi ja maksimaalsete õigustega kasutaja baasi haldamiseks.
4. Siseneme halduskonsoolist PhpMyAdmin loodud kasutaja alt ning loome tabelid ja ühendused kooskõlas meie andmebaasi MySQL mudeliga. Tabelid luuakse visuaalses redaktoris, mis on palju mugavam kui MySQL käskude käsitsi kirjutamine. PhpMyAdmin tõlgib visuaalse kujutise automaatselt MySQL-i päringukäskudesse.
5. Seejärel paigaldame rakenduse serveriosa kirjutamiseks PHP koodi IDE redaktori nimega **PhpStorm**. See arenduskeskkond on väga mugav PHP projektide jaoks. See toetab enamust populaarseid PHP raamistikke ja selles on realiseeritud hea kontekstiabi ja koodi automaattäitmise süsteem. Kasutatakse tasuta prooviversiooni, mis töötab 30 päeva piiranguga.
6. PhpStormi installeeritud kolmandate arendajate poolt pluginate paigaldamise süsteemi abil lisame **CakePHP** raamistiku toe, mida hakatakse kasutama rakenduse serveriosa kirjutamiseks. Laadime alla CakePHP raamistiku vajalikud failid ja paigutame need FTP kliendi FTP protokolli kaudu virtuaalserverisse juurkausta, vahetades failid selles kaustas vaikimisi failide vastu, mis kuuluvad CakePHP raamistikku.
7. Seadistame PhpStormi tööks meie virtuaalserveriga. Selleks märgime projekti seadetesse meie serveri aadressi ja FTP parooliga kasutaja. Nüüd saab töötada projekti koodiga, mis laaditakse automaatselt serveri töökausta.
8. Laadime alla ja paigaldame arenduse **AndroidStudio** keskkonna kliendirakenduse kirjutamiseks. Kuna AndroidStudio on peamiselt ette nähtud mobiilsete rakenduste kirjutamiseks Java keeles Android platvormile, on samuti vaja alla laadida arenduseks vajalikud SDK-d.
9. Klientide mobiilirakenduse testimiseks kasutatakse kas platvormi Android sisseehitatud emulaatorit, mis on arenduskeskkonda AndroidStudio juba installeeritud, või reaalset nutitelefoni, mis on USB kaabli kaudu arvutiga ühendatud. Samuti käib USB kaabli kaudu rakenduse silumine, mis kuvatakse

arenduskonsoolis AndroidStudio. Selle abil on mugav jälgida rakenduse kulgemist ja kuvada programmi vahetulemusi või muutujate ja mälu seisundit.

5 Kokkuvõte

Käesoleva bakalaureusetöö eesmärkideks oli teostada parkimise mobiilirakenduse detailset analüüsi, vastavalt analüüsi tulemusele luua kasutajaliidese prototüüp, viia läbi testimine ning küsitlus. Nende põhjal arendada prototüüpi edasi ning luua lõplik rakendus.

Parkimise mobiilirakenduse loomeprotsessi kirjeldamisel tõi autor välja rakenduse analüüsi ja arhitektuuri joonise, süsteemi platvormi analüüsi, andmebaasi mudeli ning kirjelduse. Samuti esitas autor vastavalt rakenduse funktsioonidele koostatud kasutajaliidese prototüüpi ja pärast testimist lõpliku rakenduse vaated ning samm-sammulist rakenduse loomeprotsessi.

Peatükis 2 selgus, milline hakkab olema Parkimise mobiilirakenduse üldine funktsionaalsus. Autor esitles rakenduse kasutusjuhtude- ja tegevusdiagrammi, kirjeldamaks süsteemi tööd ning esialgse kasutajaliidese prototüübi. Samuti sai lugeja aimu «Koridori meetodist» ning uurimismeetodi tähtsustest ja eelistustest tagasiside saamiseks. Autor kirjeldas, mis küsimused olid esitatud potensitaalsetele Parkimise mobiilirakendust kasutavatele klientidele ja mis tagasisided olid saadud.

Peatükis 3 esitles autor Parkimise mobiilirakenduse andmebaasi füüsilist mudelit ning tõi välja arhitektuuri joonise. Tänu läbi viidud uurimismeetodile sai autor saadud tagasiside põhjal potensiaalsetelt klientidelt arendada ja uuendada olemasolevat esialgset prototüüpi ning esitleda lõplikud vaated.

Kõige mahukamas peatükis 4 tõi autor välja vajalikud süsteemi tarkvara osad ning tööriistad. Parkimisrakenduse serveriosa kood on kirjutatud PHP programmeerimise keeles. Mugavuseks Serveriosa kirjutamisel kasutatakse erinevaid tugiraamistikud, PHP puhul osutus see CakePHP. Kliendiosa oli kirjutatud Android programmeerimise keeles. Oli antud ülevaade, kuidas aitavad mobiilirakenduse loomisele kaasa tugiraamistikud ja samuti arendusplatvormid nagu PhpStorm ja AndroidStudio. Andmebaasi valikuks osutus MySQL andmebaas ning serverisüsteemiks valis autor Pilvesüsteemi. Andmebaasi mudel on koostatud vastavalt rakenduse funktsioonidele ning näitab täpselt, millised tabelid on omavahel seotud. Samuti mainis autor, et tänu JSON vormingule toimub andmevahetus kliendirakenduses.

Peatükis 4 esitas autor rakenduse samm-sammulist loomeprotsessi, millises järjekorras ning mida teha tuleks, et Parkimise mobiilirakendust luua. Kokku tuli 9 sammu Parkimise mobiilirakenduse loomiseks.

Autor kirjeldas kogu rakenduse loomist, alustades nullist, kuni kogu süsteemi üles seadmiseni ning sellega kaks põhilist eesmärki, mis olid esitletud töö alguses, saime täidetud.

Kasutatud kirjandus

[1] I. Crook, “When did mobile apps become so popular?,” AppInstitute, 16 March 2018. [Võrgumaterjal]. Available: https://appinstitute.com/mobile-apps-become-popular. [Kasutatud 10 March 2019].

[2] Matteo, “What are the different types of mobile apps?,” duckma, [Võrgumaterjal]. Available: https://duckma.com/en/types-of-mobile-apps/. [Kasutatud 10 March 2019].

[3] Techopedia, “Hallway Usability Testing,” techopedia, [Võrgumaterjal]. Available: https://www.techopedia.com/definition/30678/hallway-usability-testing. [Kasutatud 16 March 2019].

[4] Г. Дударь, “Уроки Андроид программирования,” Гоша Дударь, 12 September 2017. [Võrgumaterjal]. Available: https://www.youtube.com/watch?v=Bx60qVuE-xU. [Kasutatud 3 January 2019].

[5] Andreyex, “Физический сервер против виртуального сервера: Все, что вам нужно знать,” andreyex, 27 December 2016. [Võrgumaterjal]. Available: https://andreyex.ru/stati/fizicheskij-server-protiv-virtualnogo-servera-vse-chto-vam-nuzhno-znat/. [Kasutatud 14 March 2019].

[6] Стэк Ит-Аутсорсинг, “Виртуальный сервер,” Стэк Ит-Аутсорсинг, [Võrgumaterjal]. Available: https://www.stekspb.ru/autsorsing-it-infrastruktury/it-glossary/virtual-server/. [Kasutatud 19 March 2019].

[7] Zone, “Pilveserver VPS,” zone, [Võrgumaterjal]. Available: https://www.zone.ee/et/pilveserver/vps/. [Kasutatud 5 February 2019].

[8] Veebimajutus, “Domeeni registreerimine,” veebimajutus, [Võrgumaterjal]. Available: https://www.veebimajutus.ee/domeenid/. [Kasutatud 5 February 2019].

[9] DigitalOcean, “Compute,” digitalocean, [Võrgumaterjal]. Available: https://www.digitalocean.com/pricing/. [Kasutatud 5 February 2019].

[10] php, “Что такое PHP?,” php, [Võrgumaterjal]. Available: https://www.php.net/manual/ru/intro-whatis.php. [Kasutatud 22 March 2019].

[11] wikipedia, “CakePHP,” wikipedia, 9 May 2018. [Võrgumaterjal]. Available: https://ru.wikipedia.org/wiki/CakePHP. [Kasutatud 19 March 2019].

[12] CakePHP, “Первое знакомство с CakePHP,” CakePHP, 3 May 2019. [Võrgumaterjal]. Available: https://book.cakephp.org/4.0/ru/intro.html. [Kasutatud 22 March 2019].

[13] CakePHP, “Цикл обработки запросов CakePHP,” CakePHP, 3 May 2019. [Võrgumaterjal]. Available: https://book.cakephp.org/4.0/ru/intro.html. [Kasutatud 22 March 2019].

[14] Jetbrains, “PhpStorm,” jetbrains, [Võrgumaterjal]. Available: https://www.jetbrains.com/phpstorm/. [Kasutatud 5 March 2019].

[15] Developers, “About the platform,” developers.android, [Võrgumaterjal]. Available: https://developer.android.com/guide/platform. [Kasutatud 18 March 2019].

[16] Developer, “Android Studio,” developer.android, [Võrgumaterjal]. Available: https://developer.android.com/studio. [Kasutatud 18 March 2019].

[17] И. Константин, “Доля ОС Android достигла 87,5% рынка мобильных ОС,” mobilereview, 3 November 2016. [Võrgumaterjal]. Available: https://mobile-review.com/news/dolya-os-android-dostigla-875-rynka-mobilnyx-os. [Kasutatud 26 March 2019].

[18] Метод Лаб, “СУБД MySQL,” Метод Лаб, [Võrgumaterjal]. Available: https://www.methodlab.ru/technology/mysql.shtml. [Kasutatud 28 March 2019].

[19] Wikipedia, “JSON,” wikipedia, 12 April 2019. [Võrgumaterjal]. Available: https://ru.wikipedia.org/wiki/JSON. [Kasutatud 19 April 2019].

Lisa 1 – Kasutusjuhtude diagrammi dokumentatsioon

Tabel 4. Parkimise mobiilirakenduse kasutusjuhtude diagrammi dokumentatsioon Joonisele 1

Kasutusjuht: Kasutajakonto registreerimine **Tegutsejad:** Klient **Eeltingimused:** Klient on rakenduse käivitanud. **Kirjeldus:** Kasutajakonto registreerimisel klient peab täitma vajalikku registreerimisvormi kus ta edastab parkimisfirmale oma kontaktandmed.
Kasutusjuht: Vabade kohtade vaatamine **Tegutsejad:** Klient **Eeltingimused:** Klient on kasutajakonto loonud. **Kirjeldus:** Pärast kasutajakonto loomist kliendile antakse võimalus vaadata parklas olevaid vabade kohtade arvu ja teostada broneerimist.
Kasutusjuht: Parkimiskoha broneerimine **Tegutsejad:** Klient **Eeltingimused:** Pärast kasutajakonto loomist kliendile antakse võimalus broneerida parkimskohta soovitud aja perioodiks. **Kirjeldus:** Parkimiskoha broneerimisel klient valib soovitud saabumise ja lahkumise kuupäeva ning kellaja (soovi korral võib jätta parklast lahkumise kuupäeva ja aja määramatuks).
Kasutusjuht: Maksmise teostamine **Tegutsejad:** Klient **Eeltingimused:** Klient on parkimisajaperioodi valinud. **Kirjeldus:** Klient teostab maksmise kliendirakenduses «Finish» nupu vajutamisel.
Kasutusjuht: Parkla hõivatuse jälgimine **Tegutsejad:** Parkla Omanik **Eeltingimused:** Parkla **Kirjeldus:** Parkla omanik võib jälgida parkla hõivatust soovitud ajaperioodil (nädala kaupa).
Kasutusjuht: Registreeritud kasutajakontode vaatamine **Tegutsejad:** Parkla Omanik **Eeltingimused:** Kasutajad on kasutajakonto loonud. **Kirjeldus:** Parkla omanik näeb kõiki registreerinud kasutajaid ja samuti näeb nende andmeid (nimi, perekonnanimi, email, telefoninumber, autonumbrid).

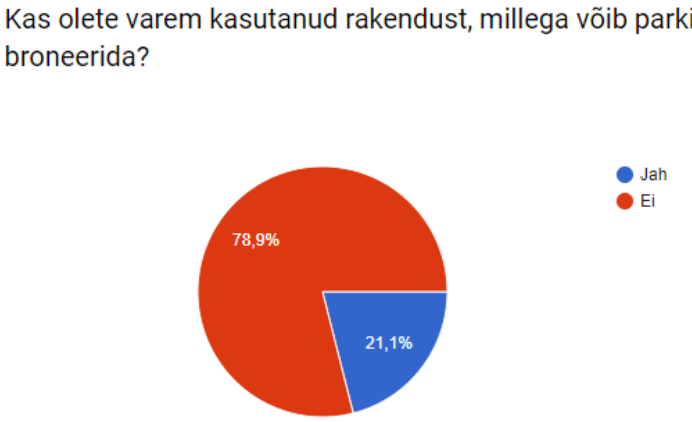
Lisa 2 – Tegevusdiagrammi dokumentatsioon

Tabel 5. Parkimise mobiilirakenduse tegevusdiagrammi dokumentatsioon Joonisele 2

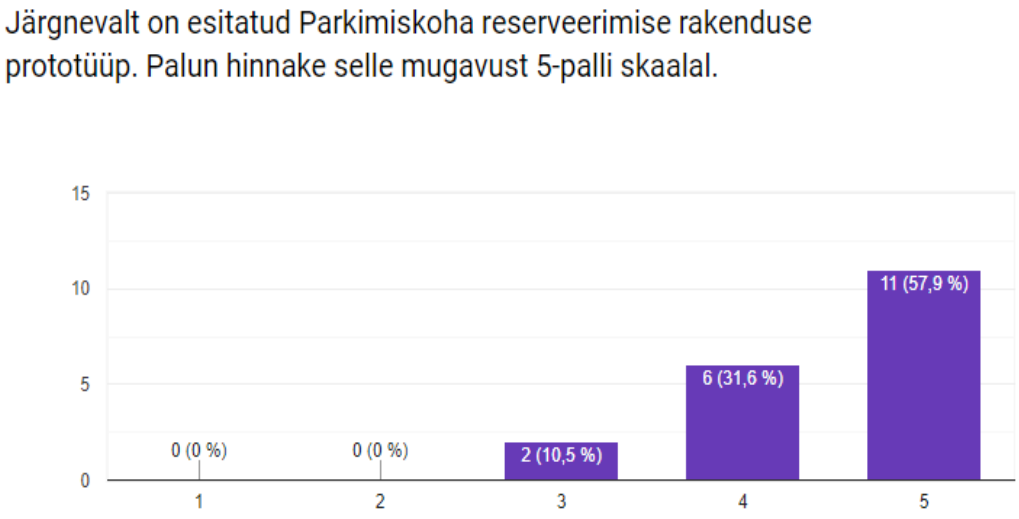
Tegevus	**Tegevuse kirjeldus**
Soovib parkimiskohta broneerida	Klient soovib rakenduse kaudu parkimiskohta ette broneerida.
O1	Rakendusse sisse logides kliendil on võimalus siseneda olemasoleva kasutajakontoga või kui kliendil pole kontot, alustada registreerimisega.
Loob kasutajakonto	Kliendil ei ole kasutajakontot ja ta alustab kasutajakonto registreerimist.
Siseneb olemasoleva kasutajakontoga	Kui kliendil on kasutajakonto juba olemas, saab ta kohe alustada parkimiskoha broneerimisega.
O2	Kasutajakonto loomiseks klient peab nõustuma Parkimisfirma tingimustega isikliku andmete edastamiseks, sealhulgas kasutajakonto registreerimisel esitada kontaktandmed: nimi, perenimi, email, telefoninumber, autonumbrid. Tingimustega nõustumisel klient jätkab registreerimisvormi täitmisega. Tingimustega mitte nõustumisel süsteem lükkab tagasi parkimiskoha broneerimist.
Täidab registreerimisvormi	Parkimiskoha broneerimiseks kliendile tuleb registreerimisvormis esitada kõik vajalikud kontaktandmed.
O3	Registreerimisel klient peab täitma kõik vajalikud lahtrid, ühe lahtri täitmata jätmine tähendab registreerimise ebaõnnestumist. Registreerimisvormi õigesti täitmisel süsteem salvestab andmed ja loob kasutajakonto.
Täidab ülejäänud lahtrid	Klient täidab nõutud lahtri ja jätkab registreerimisega.
Kuvab vabade kohtade arvu	Eduka registreerimise korral, süsteem kuvab kasutajale parklas olevad vabade kohtade arvu.
Vaatab vabade kohtade arvu	Broneerimise teostamiseks klient peab veenduma, et parklas tema soovitud ajaperioodil on vabu kohti.
Valib saabumise, lahkumise kuupäeva ja aja	Parkimiskoha broneerimisel klient valib saabumise, lahkumise kuupäeva ja kellaaja (soovi korral klient võib jätta lahkumise kuupäeva ja kellaaja määramatuks).

O4	Süsteem küsib kasutajalt broneerimise kinnitust ja nõusolekut broneerimist kinnitada, kasutaja positiivsel vastusel fikseerib süsteem parkimiskoha broneeringu, mitte nõustumise korral süsteem lükkab broneerimise tagasi.
Teostab maksmise	Klient maksab parkimise eest parkimise ajaperioodi lõpetamisel.
O5	Parkimiskoha eest maksmisel süsteem kontrollib maksmise edukust, eduka tasumise korral fikseerib süsteem maksmise sooritust, mitte eduka tasumise korral klient suunatakse maksmise teostamise kordamisele.

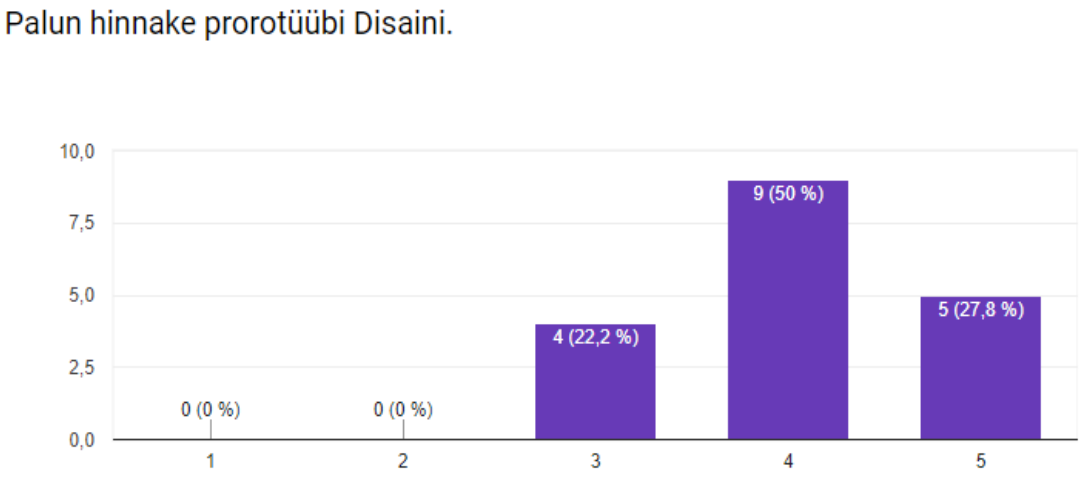
Lisa 3 – Küsitluse vastused



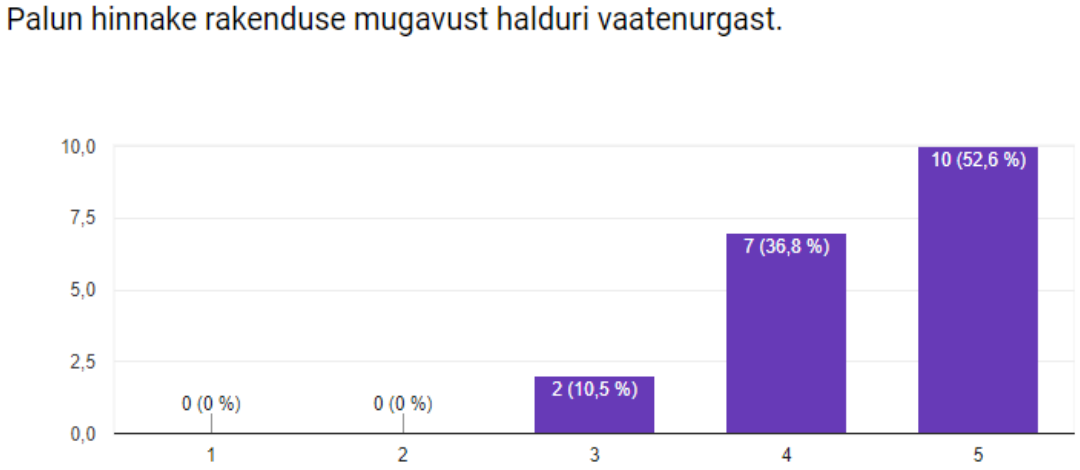
Joonis 14. Klientide parkimisrakenduse kasutamiskogemus.



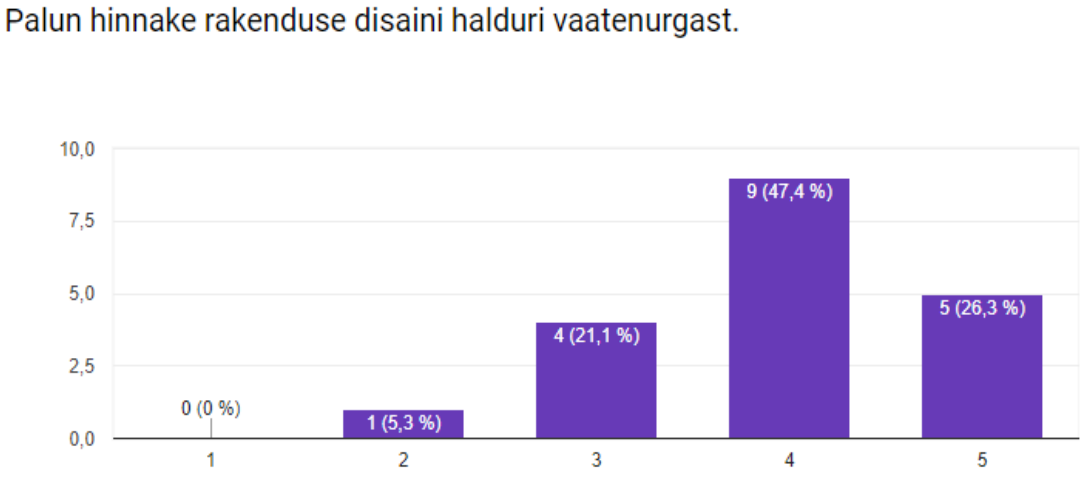
Joonis 15. Parkimisrakenduse mugavuse hinnang 5-palli skaalal.



Joonis 16. Prototüübi disaini hinnang.



Joonis 17. Rakenduse mugavuse hinnang halduri vaatenurgast.



Joonis 18. Rakenduse disaini hinnang halduri vaatenurgast.