TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatika instituut
Infosüsteemide õppetool

PostgreSQL-i põhise meta-andmetega juhitavate veebirakenduste kiirprogrammeerimiskeskkonna projekteerimine ja realiseerimine

Magistritöö

Üliõpilane: Rait Raidma
Üliõpilaskood: 143682IAPM
Juhendaja: dotsent Erki Eessaar

Tallinn
2016

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.

....

(kuupäev) *(allkiri)*

Annotatsioon

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 59 leheküljel, 11 peatükki, 20 joonist, 2 tabelit.

Abstract

The thesis is in estonian and contains 59 pages of text, 11 chapters, 20 figures, 2 tables.

Lühendite ja mõistete sõnastik

SQL	*Structured Query Language*, struktureeritud andmebaasikeel andmete käitlemiseks, õiguste jagamiseks ning andmebaasiobjektide haldamiseks
FSF	*Free Software Foundation*, MTÜ, mis propageerib arvuti kasutajate vabadust ja kaitseb vaba tarkvara kasutajate õigusi
OSI	*Open Source Initiative*, Organisatsioon, mis propageerib avatud lähtekoodiga tarkvara
Juurutama	*Deploy*, Tarkvara või riistvara töölepanekuga seotud protsesside - installeerimine, konfigureerimine, käitamine, testimine - läbimine [36]
CRUD	*Create Read Update Delete*, Lühend, mis tähistab andmetega manipuleerimise nelja põhitegevust: loomine, lugemine, muutmine ja kustutamine
Meta-andmed	Andmed andmete kohta
Meta-andmetega juhitav	Süsteemi käitumist ja väljanägemist juhitakse andmetega.
URL	*Uniform Resource Locator*, Internatiaadress. Viit arvutivõrgus olevale ressursile. [36]
Kiirprogrammeerimine	*Rapid Application Development*, Arendussüsteem, mis annab programmeerijatele võimaluse kiiresti programme koostada. Üldiselt on RAD-süsteemides rida graafiliste kasutajaliideste loomiseks mõeldud tööriistu, mis oluliselt lühendab taoliste liideste loomisele kuluvat aega [36]
DOM	*Document Object Model*, dokumendiobjektide mudel. Eeskiri selle kohta, kuidas objekte (tekst, pildid, pealkirjad, lingid jne.) veebilehel esitada. DOM määrab ära, millised atribuudid kuuluvad millise objekti juurde ning kuidas objekte ja atribuute käsitleda [36]

HTML	*HyperText Markup Language*, hüpertekst-märgistuskeel. Enimlevinud kodeerimissüsteem (tekstivorming) veebidokumentide loomiseks. HTML koodid ehk märgendid määravad ära selle, kuidas veebileht arvutiekraanil välja näeb [36]
CSS	*Cascading Style Sheets*, kaskaadlaadistik. Veebilehtede valmistajatele ja kasutajatele mõeldud laadistik. Laadilehed (style sheets) kirjeldavad, kuidas HTML dokumente esitada kuvaril, printeril või kõnesüntesaatorist kostva kõnena [36]
AJAX	*Asynchronous JavaScript And XML*, asünkroonne JavaScript ja XML. Interaktiivsete veebirakenduste loomise meetod, kus andmevahetus brauseri ja veebiserveri vahel toimub ilma, et oleks vaja kogu lehte uuesti laadida [36]

Sisukord

Joonistе loetelu

Tabelite loetelu

1 Sissejuhatus

Järgnevalt on välja toodud, miks ning kuidas antud tööd tehakse.

1.1 Taust ja probleem

Töö idee sai alguse TTÜ-s õpetatavast ainest "Andmebaasid II", mille raames tuleb üliõpilastel ühe õpiväljundina luua andmebaas koos seda kasutava rakendusega, kus rakendus suhtleb andmebaasiga läbi andmebaasiliidese. Antud aines võib kasutada andmebaasisüsteeme PostgreSQL [24] ja Oracle [21]. Juhul, kui andmebaas on loodud Oracle andmebaasisüsteemi abil, siis on üliõpilastel rakenduse loomiseks võimalus kasutada kiirprogrammeerimiskeskkonda Oracle APEX [20]. PostgreSQL andmabaasisüsteemiga loodud andmebaasi korral tuleb rakendus programmeerida kasutades PHP-d [22]. See tähendab, et üliõpilane ei saa keskenduda täielikult andmebaasi täiustamisele vaid peab tegelema ka lisaprogrammeerimisega. Töö tulemusena valmiva süsteemi abil peaks üliõpilastel olema lihtsam luua näidisrakendusi, mis kasutavad andmebaasisüsteemina PostgreSQL-i. Samas leiab autor, et valmiv süsteem on piisavalt võimekas, et leida raakendust teisteski kohtades.

Töö valmis 2016. aasta kevadel Tallinna Tehnikaülikoolis.

1.2 Ülesande püstitus

Töö eesmärgiks on disainida ning realiseerida PostgreSQL põhine kiirprogrammeerimiskeskkond, mille abil saaks luua teisi veebipõhiseid rakendusi, mis hoiavad andmeid samas andmebaasiserveris kui loodav süsteem. Loodavates rakendustes peab andmete pärimine, lisamine, muutmine ning kustutamine käima läbi andmebaasiliidese. Kogu vajalik info rakenduste kuvamiseks ja käitumise juhtimiseks tuleb hoida loodava süsteemi metaandmete andmebaasis.

Loodav süsteem peab toetama PostgreSQL 9.4 ja PHP 5.5.0 ning tuleb välja anda vabavara litsentsi all, et soodustada süsteemi laialdasemat levikut.

1.3 Metoodika

Esiteks tuleb uurida, kas liideste kasutamine andmebaasi poole peal annab süsteemile mingi eelise. Seejärel tuleb selgeks teha, kas ja kuidas saab ühest andmebaasist suhelda teiste andmebaasidega, mis asuvad samas andmabaasiserveris. Ning viimasena, kuidas küsida teistest andmebaasidest infot liideste kohta. Lisaks uurin, milliseid sarnaseid süsteeme on veel olemas ning kuidas need on üles ehitatud.

Töö tulemusena valmib süsteem, mille abil saab luua veebipõhiseid rakendusi, mis kasutavad andmebaasiga suhtlemiseks andmebaasiliideseid. Töö tulemuse valideerimiseks loodakse näidisrakendus, kus realiseeritakse õppejõu poolt ette antud kasutusjuhud, mis sarnanevad üliõpilastöödes esinevatele kasutusjuhtudele.

1.4 Ülevaade tööst

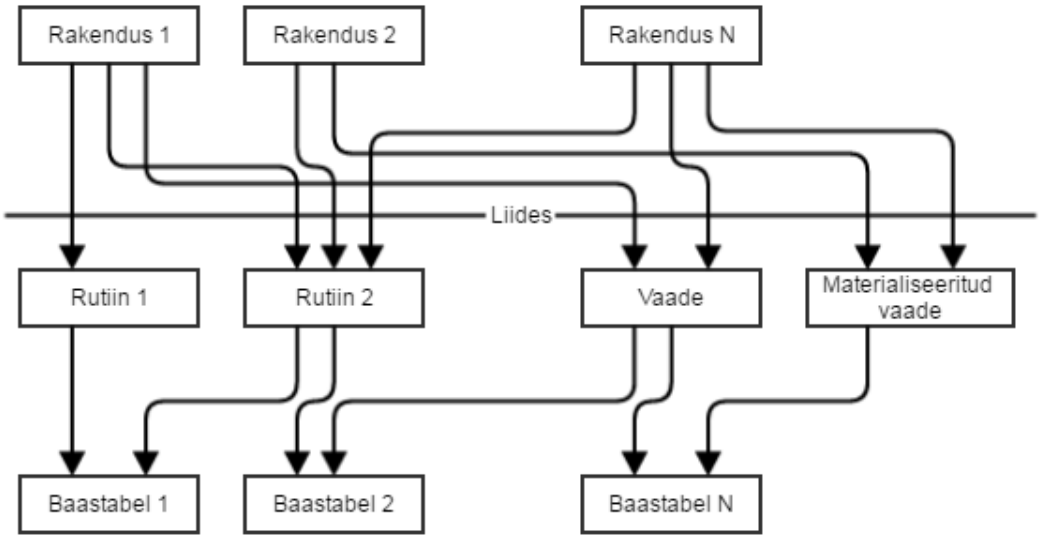
TODO

2 Teoreetiline taust

Järgnevalt viiakse lugeja kurssi vajalike teadmistega, mida on vaja süsteemi loomiseks.

2.1 Andmebaasi avalik liides

Liides on sõltumatute süsteemide vaheline leping, kus on kirjeldatud, millisel viisil saab üks süsteem teisega suhelda. Andmebaasis saab liideseid kirjeldada rutiinidena ja vaadetena (vt Joonis 1).



Joonis 1. Andmebaasi avalik liides

2.1.1 Vaadete kasutamise eelised ja võimalused

- Võimaldavad igale rakendusele saab luua spetsiifilise vaate andmetest, ilma et oleks vaja teha muudatusi andmemudelis.
- Võimaldavad vähendada rakenduse koodi ja andmemudeli vahelist sidusust. See võimaldab teha muudatusi andmemudelis, ilma et olemasolev rakendus katki läheks.
- Neile saab anda rakenduse-spetsiifilised veerunimed, andmetüübid ja pikkused, mis võimaldab otse andmete sidumist rakenduses kasutatavate mudelitega.

- Võimaldavad jõustada andmete turvamist. Erinevatele kasutajagruppidele saab kuvada andmeid erineval kujul, nii et kasutaja näeb üksnes neid andmeid, mida ta on volitatud nägema. PostgreSQL andmebaasisüsteemis tuleks lisaks kasutada turvabarjääri *WITH (security_barrier)* lisatingimust. See takistab peidetud ridade kuvamist ka juhul, kui kasutatakse kuritahtlikult valitud funktsioone ja operaatoreid, et näha varjatud infot [28]
- Võimaldavad pärida andmeid erinevatest tabelitest ja andmebaasidest, peites kasutajate eest päringu tegeliku keerukuse. Vaate koostamiseks vajalik päring on eelnevalt kompilleeritud ja optimiseeritud, et tagada parem jõudlus. Vaated kasutavad päringu täitmisel baastabelitele loodud indekseid.
- Võimaldavad varjata rakenduse eest baastabelites olevaid disaini -ja andmevigasid, andes lisaaega nende parandamiseks.
- Võimaldavad kuvada samu andmeid erineval kujul ühendatuna, kasvõi nt XML-na või JSON-na.
- Läbi vaadete, mis vastavad teatud tingimustele, on võimalik teha andmemuudatusi baastabelites, kui realiseerida INSTEAD OF triggerid.

[6, lk 172-173]

2.1.2 Vaadete kasutamisel tekkida võivad probleemid PostgreSQL andmebaasisüsteemi korral

- Andmebaasisüsteem ei suuda kasutada põhipäringu tingimust (*WHERE* klausel) vaate alampäringus kui vaate alampäringus tehakse ühendi leidmist *UNION* või *UNION ALL* või kui alampäring sisaldab aknafunktsiooni (nt *ROWNUMBER() OVER()* ja *LAG() OVER()*)
- Kui vaate alampäring sisaldab agregaatfunktsiooni (ilma *GROUP BY* klauslita), *ROWNUM* pseudoveergu, aknafunktsiooni *ROWNUMBER() OVER()* või rekursiivset päringut, siis täidetakse vaate põhjal tehtud päring ja vaate alampäring eraldi.
- Vaate turvabarjääri *WITH (security_barrier)* kasutamine seab piirangud vaate tingimusklauslite ning vaate põhjal tehtud päringu tingimusklauslite mestimisele, mille tulemusena loodav täitmisplaan ei pruugi olla optimaalne
- Kui vaate alampäringus viidatakse teistele vaadetele, siis nende vaadete alampäringud täidetakse eraldi, mille tulemusena suureneb päringu täitmiskiirus.

[13, lk 101-102]

2.1.3 Rutiinide kasutamise eelised

- Üle võrgu saadetavate andmete ja SQL koodi hulk hoitakse minimaalsena, mille tulemusel suureneb rakenduse jõudlus.
- Rutiinide kood on andmebaasi serveris eelnevalt kompilleeritud ja optimiseeritud, suurendades rutiini täitmise efektiivsust.
- Andmetöötluse jaoks kasutatakse andmebaasiserveri jõudlust, mitte rakendusserveri ega kliendi masina oma.
- Rutiinis olevat SQL koodi on lihtsam testida ja optimiseerida, kui rakendusse sisse kirjutatud SQL-i.
- Rutiinide käivitusõiguste abil saab piirata ligipääsu teatud rollidele ning suurendada seeläbi turvalisust.
- Rutiinis käivitatavad laused tehakse ühe transaktsiooni jooksul. See aitab vältida osalisi andmemuudatusi, kus üks osa muudatustest läks läbi, teine osa aga mitte.

[6, lk 179, 195]

2.1.4 Rutiinide puudused

- Koodifunktsionaalsus on piiratud.
- Keerulisemaid rutiine ei pruugi olla võimalik teisaldada ühelt andmebaasisüsteemilt teisele.
- Rutiine on keerulisem kapseldada ning selle tulemusena võib üks ärireegel olla jaotatud mitme rutiini vahele, mis teeb ärireegli haldamise keerulisemaks.

[11]

2.1.5 Järeldus andmebaasiliideste kasutamise kohta

Andmebaasiliidesed annavad kasutajale võimaluse viia andmebaasi ja rakenduse vaheline sidusus minimaalseks ning suurendada andmete turvalisust ja terviklikust. Mõningatel

juhtudel võivad aga vaated mõjuda jõudlusele nagatiivselt ning rutiinides võib mõningate tegevuste täitmiseks loodava koodi kirjutamine olla keerulisem kui rakenduse kihis. Autori arvates ei kaalu negatiivsed aspektid üle positiivseid ning seetõttu leiab autor, et andmebaasiliideseid tuleks võimaluse korral kasutada.

2.2 Ühendumine teiste andmebaasidega

Kui loodav süsteem installida serverisse, kus on mitu andmebaasi, siis peab kõikide nende andmebaaside põhjal olema võimalik luua rakendusi. Selleks peab loodav süsteem olema võimeline tegema päringuid samas andmebaasiserveris olevatesse andmebaasidesse. PostgreSQL andmebaasisüsteemis pole realiseeritud andmebaaside vahelisi viitasid ning seetõttu ei saa koostada päringuid kujul:

```
select * from other_db_name.schema_name.table_name;
```

Eelnev päring annab tulemuseks veateate:

```
ERROR:  cross-database references are not implemented: "
   other_db_name.schema_name.table_name"
```

Selleks, et ühenduda väliste PostgreSQL andmebaasidega, tuleb kasutada kas dblink või postgres_fdw moodulit.

2.2.1 dblink

Mooduli installeerimine:

```
CREATE EXTENSION IF NOT EXISTS dblink;
```

Andmete küsimiseks välisest andmebaasist tuleb ette anda andmebaasi nimi, kasutaja ja parool ning lause, mida käivitada soovitakse. Päring käivitatakse välises andmebaasis. Päringuks võib olla iga SQL lause, mis tagastab read.[25] Allpool on toodud näide päringu koostamisest dblink mooduli abil.

```
SELECT schema_name, owner_id
FROM dblink(
   'dbname=external_database_name user=
    external_database_user password=
    external_database_user_password',
```

```
  'SELECT upper(nspname), nspowner FROM pg_catalog.
   pg_namespace;'
) AS (
  schema_name varchar,
  owner_id int
);
```

2.2.2 postgres_fdw

Selle mooduli poolt pakutav funktsionaalsus kattub suurel määral *dblink* 2.2.1 mooduli funktsionaalsusega, kuid pakub standardsemat süntaksit päringute koostamiseks ning võib *dblink*-i kohati edestada jõudluse poolest.

postgres_fdw loob ühenduse välise serveriga siis, kui tehakse esimene päring välise tabeli vastu. Seda ühendust hoitakse alles ning kasutatakse järgmiste päringute jaoks sama sessiooni piires. Kui väliselt serverilt küsitakse infot erinevate kasutajatena *user mappings*, siis luuakse iga kasutaja jaoks uus ühendus.

postgres_fdw üritab optimeerida väliseid päringuid, et vähendada küsitavate andmete hulka. Selleks saadetakse koos päringuga *WHERE*-tingimus ning ei laeta alla veerge, mida pole päringu täitmiseks vaja. Selleks et vältida valesid päringutulemusi, ei saadeta *WHERE*-tingimusi, kui kasutatakse midagi peale sisse ehitatud andmetüüpide, operaatorite ja funktsioonide või kui operaatorid ja funktsioonid pole muutumatud (*immutable*). [26]

postgres_fdw võimaldab lisaks andmete küsimisele (*SELECT*) ka andmeid lisada (*INSERT*), muuta (*UPDATE*) ja kustutada (*DELETE*) välisest tabelist. Küll aga ei võimalda antud moodul välja kutsuda välises andmebaasis olevaid funktsioone kujul:

```
SELECT function_from_external_database();
```

2.2.3 Mooduli valik

Kuna loodav süsteem peab suutma välja kutsuda välistes andmebaasides olevaid funktsioone, siis pole tuleb kasutada *dblink* 2.2.1 moodulit.

2.3 Andmebaasiobjektide kirjelduste küsimine

Loodav süsteem peab teistest andmebaasidest küsima infot andmebaasibjektide kohta, et kuvada süsteemi kasutajale info andmebaasiliidestest, mida rakenduse loomisel on võimalik kasutada. Selleks vajaliku info saab küsida süsteemikataloogidest: information_schema ja pg_catalog. Information_schema sisaldab vaateid andmebaasis olevate objektide kohta. Kuna information_schema on defineeritud SQL standardis, siis võib eeldada, et selle formaat ei muutu ning seetõttu tuleks eelistada seda kataloogi, et vältida loodava rakenduse katki minekut järgmiste PostgreSQL versioonide korral. [27] Küll aga ei sisalda information_schema infot PostgreSQL-spetsiifiliste võimaluste kohta. Selleks tuleb pöörduda pg_catalog-i poole. Pg_catalog-st saab lisaks pärida infot samasse andmebaasiserverisse kuuluvate andmebaaside, materialiseeritud vaadete ning kasutajate paroolide kohta. [29] Täpsem loetelu olulisematest süsteemikataloogide vaadetest, mida süsteemi loomisel vaja läheb, on välja toodud Lisa 1-es.

2.4 Eksisteerivate programmide analüüs

Töö käigus uuriti, milliseid sarnaseid süsteeme on veel loodud ning kuidas need on üles ehitatud. Järgnevalt on esitatud ülevaade nendest süsteemidest.

2.4.1 Oracle Application Express (APEX)

Oracle APEX on veebipõhine rakendus loomaks kiirelt ja lihtsalt teisi veebipõhiseid rakendusi. Kogu süsteem on juhitav andmebaasis hoitavate metaandmetega. APEX kasutab tööks Oracle anembaasisüsteemi.

APEX (v 5.0.3.00.03) koosneb neljast põhiosast:

- Application Builder - Võimaldab luua ja hallata uusi rakendusi. Rakendused koosnevad lehtedest. Lehed omakorda sisaldavad regioone. Regioonides võib kuvada raporteid, graafikuid, vorme jpm. Regioonid sisaldavad komponente, mille abil on võimalik kasutajalt infot küsida ning seda esitada. Lisaks on võimalik nähe lehtede statistikat ning hallata seadeid.
- SQL Workshop - Võimaldab näha ja hallata andmebaasiobjekte, jooksutada päringuid, importida/exportida anmebaasis olevaid andmeid, koostada pärngkuid graafilise liidese abil, luua RESTful liideseid jpm.

- Team Development - Tööde- ja vigadehaldus süsteem. Võimaldab arendajatel ülesandeid planeerida ja hallata.
- Packaged Apps - Galerii näidisrakendustest, mida on võimalik kohe kasutamiseks installeerida.

[20]

2.4.2 NuBuilder

NuBuilder on veebipõhine arendusplatvorm loomaks veebipõhiseid rakendusi. Lehtede kirjeldused (sh PHP, JS ja SQL päringud) hoitakse andmebaasis, mis muudab rakenduse varundamise lihtsaks.

NuBuilder on kirjutatud PHP-s ning andmeid hoitakse MySQL andmabaasisüsteemis. Tabelite põhjal on võimalik luua lihtsaid CRUD vorme, kus on võimalik tabelis olevaid andmeid lugeda, lisada, muuta ja kustutada. SQL päringute põhjal on võimalik luua raporteid, mida arendaja saab veebiliidese kaudu disainida. Oma kodulehel väidavad nad, et tegu on *Open Source* tarkvaraga ning lähtekood on avalikult üleval [18], kuid kusagil pole mainitud, millise *Open Source* litsentsi alt on tarkvara välja antud.

Koodi puhul täheldasin mitut puudujääki:

- Failid on kehvasti struktureeritud - php, js, png ja gif failid on kõik koos ühes kaustas.
- PHP ja HTML on kirjutatud läbisegi, mis teeb disaini muutmise keeruliseks.
- Kasutatakse $GLOBALS muutujat - see raskendab arusaamist, kus võidakse muutujale programmi töö ajal väärtusi omistada.
- Funktsioonid on liiga pikad - paljud funktsioonid täidavad korraga liiga palju ülesandeid ja seetõtu on raskendatud nendest arusaamine.

[17]

2.4.3 Xataface

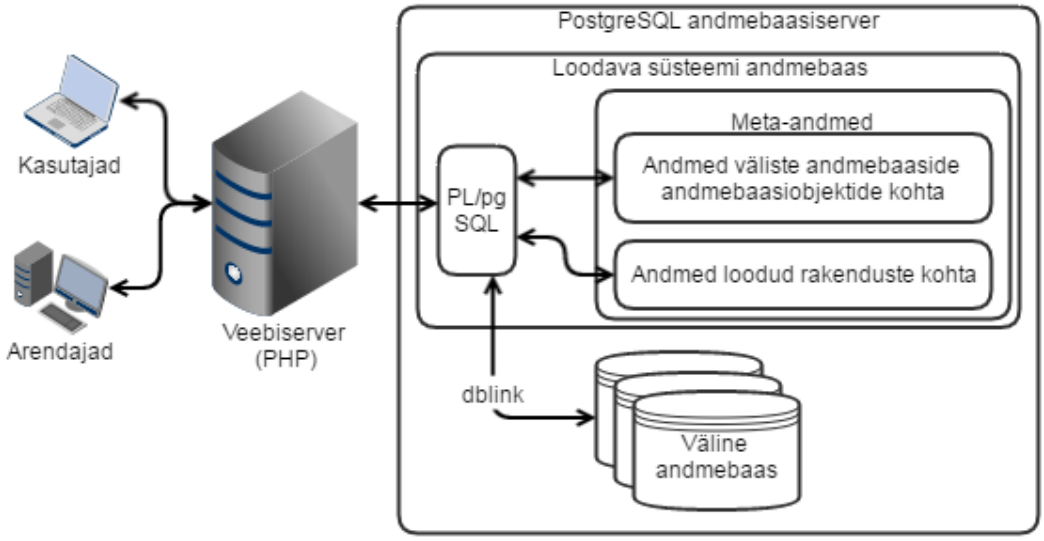
Xataface on programm, millega saab tabelite põhjal genereerida vorme ja kuvasid. Pärast genereerimist tuleb loodud failid serverisse üles laadida. Lehtede konfigureerimine toimub INI failide abil.[37]

Xataface on avatud lähtekoodiga ning antud välja GPL litsentsi all. Programm on kirjutatud PHP-s [22] ning andmebaasina kasutatakse MySQL-i [16].

Kasutatud on palju väliseid teeke. Programmil on üks põhiline arendaja ning igapäevast arendustööd ei toimu. [38]

2.5 Täpsustunud ülesande püstitus

Kasutades PostgreSQL 9.4 andmebaasisüsteemi [24] ja PHP 5.5 skriptimiskeelt [22], tuleb realiseerida süsteem, mille abil saab luua teisi veebipõhiseid rakendusi. Loodavad rakendused peavad andmebaasiga suhtlemiseks kasutama vaadete, materialiseeritud vaatede ja funktsioonide abil loodud liidest 2.1. Loodav süsteem peab info liideste kohta küsima väliste andmebaaside süsteemikataloogidest 2.3 kasutades dblink moodulit 2.2.1ning hoidma seda enda andmebaasis. Süsteemi tõõpõhimõtet kirjeldab joonis 2. Valminud süsteem tuleb teha interneti teel avalikult kättesaadavaks.



Joonis 2. Süsteemi tööpõhimõte

2.6 Litsents

Üheks töö eesmärgiks oli avaldada loodava prototüübi lähtekood avatud tarkvarana. Olemasolevaid litsentse on väga palju. Selleks, et valida välja litsents, mille all avaldada loodav tarkvara, leian esiteks populaarseimad litsentsid ning võdlen neid omavahel. GitHub-i

poolt avaldatud statistika kohaselt on 2016 aasta märtsi seisuga populaarseimad litsentsid: MIT (44,69%), GPLv2 (12,96%), Apache (11,19%) ja GPLv3 (8,88%). [3]

Kõik eelpool nimetatud litsentsid täidavad nii *Free Software* (vt Lisa 2) kui ka *Open Source* (vt Lisa 3) tingimusi. Tabelis 1 on välja toodud litsentside võrdlus.

Tabel 1. Litsentside võrdlus

	Nõutud	Lubatud	Keelatud
MIT	Litsents ja copyright märge	Kaubanduslik kasutamine Jagamine Muutmine Privaatne kasutamine	Võtta vastutusele
Apache License 2.0	Litsents ja copyright märge Teavitus muudatustest	Kaubanduslik kasutamine Jagamine Muutmine Patendi kasutamine Privaatne kasutamine	Võtta vastutusele Kasutada kaubamärki
GNU GPLv3 GNU GPLv2	Lähtekoodi avaldamine Litsents ja copyright märge Sama litsents Teavitus muudatustest	Kaubanduslik kasutamine Jagamine Muutmine Patendi kasutamine Privaatne kasutamine	Võtta vastutusele

[14]

Valitud sai MIT litsents, kuna see seab kasutajatele kõige vähem piiranguid ning arendajale kõige vähem kohustusi.

3 Süsteemi analüüs

3.1 Tegutsejad

- Arendaja.
- Kasutaja.

Arendaja on laiendatud õigustega kasutaja, kellele on lubatud hallata süsteemis loodud rakendusi.

3.2 Terviksüsteemi tükeldus allsüsteemideks

Loodav süsteem jagatakse allsüsteemideks, et oleks lihtsam modulaarselt arendada ning struktureeritult kirjeldada loodavat funktsionaalsust.

3.2.1 Pädevusalad

- Arendaja pädevusala.
- Kasutaja pädevusala.

Arendaja pädevusala kasutab kõiki allsüsteeme.

Kasutaja pädevusala kasutab ainult rakenduse allsüsteemi.

3.2.2 Funktsionaalsed allsüsteemid

- Rakenduse funktsionaalne allsüsteem.
- Rakenduste funktsionaalne allsüsteem.
- Lehtede funktsionaalne allsüsteem.
- Regioonide funktsionaalne allsüsteem.
- Navigatsioonide funktsionaalne allsüsteem.

- Mallide funktsionaalne allsüsteem.

Antud töös ei realiseerita mallide funktsionaalset allsüsteemi, kuna töö maht läheks liiga suureks ning loodav süsteem on kasutatav ka ilma selleta.

3.2.3 Registrid

- Andmebaasiobjektide register.
- Rakenduste register.
- Lehtede register.
- Regioonide register.
- Navigatsioonide register.
- Mallide register.

3.3 Rakenduse funktsionaalne allsüsteem

3.3.1 Eesmärgid

- Võimaldada arendajal ja kasutajal kasutada loodud rakendust.

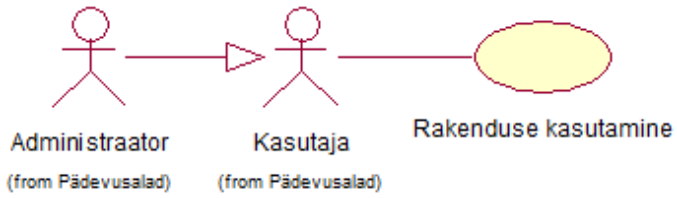
3.3.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem ei teeninda ühtegi registrit.

Allsüsteem kasutab andmebaasiobjektide registrit, rakenduste registrit, lehtede registrit, regioonide registrit, navigatsioonide registrit ja mallide registrit.

3.3.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud rakenduse funktsionaalse allsüsteemi kasutusjuhtude eskiismudel (vt Joonis 3) ja seal esitatud kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis.



Joonis 3. Rakenduse funktsionaalse allsüsteemi kasutusjuhtude eskiismudel

Kasutusjuht: Rakenduse kasutamine

Tegutsejad: Arendaja, Kasutaja

Kirjeldus: Kasutaja saab kasutada loodud rakendust. Kasutajale kuvatakse aktiivse lehekülje nähtavate regioonide sisu. Lehel võidakse kuvada navigatsioone, raporteid, vorme ja HTML teksti. Lehtede vahel saab liikuda klikates navigatsiooniregiooni poolt kuvatavatele linkidele. Kui lehekülg pole valitud, siis suutatakse kasutaja avalehele. Kui rakenduses kuvatav lehekülg nõuab, et kasutaja oleks autenditud, siis kuvatakse kasutajale autentimisvorm, kus küsitakse kasutajanime ja parooli. Kui kasutaja poolt sisestatud kasutajanimi ja parool on korrektsed, siis lubatakse kasutajal näha kaitstud lehekülgi. Sessiooni jooksul peab autentima ainult ühe korra.

3.4 Rakenduste funktsionaalne allsüsteem

3.4.1 Eesmärgid

- Võimaldada arendajal saada ülevaade loodud rakendustest.
- Võimaldada arendajal luua uus rakendus.
- Võimaldada arendajal muuta olemasolevate rakenduste seadeid.
- Võimaldada arendajal kustutada olemasolevaid rakendusi.
- Võimaldada arendajal muuta rakenduse autentimismeetodit.

3.4.2 Allsüsteemi poolt kasutatavad registrid

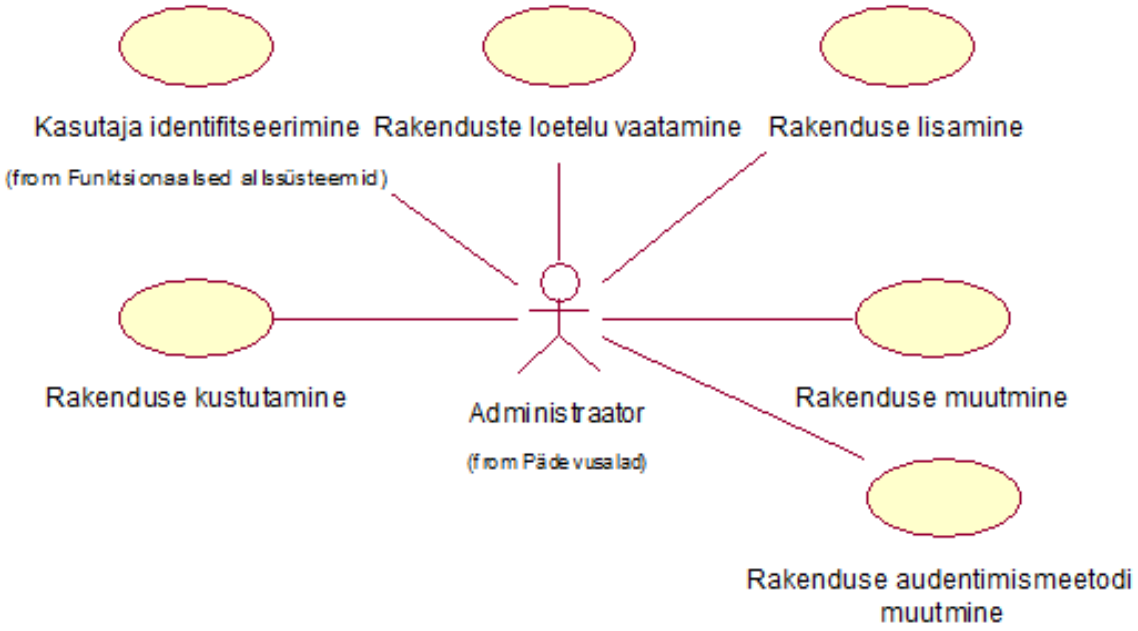
Allsüsteem teenindab rakenduste registrit.

Allsüsteem kasutab andmebaasiobjektide registrit.

Allsüsteem kasutab mallide registrit.

3.4.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud rakenduste funktsionaalse allsüsteemi kasutusjuhtude eskiismudel (vt Joonis 4) ja seal esitatud kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis.



Joonis 4. Rakenduste funktsionaalse allsüsteemi kasutusjuhtude eskiismudel

Kasutusjuht: Kasutaja identifitseerimine

Tegutsejad: Arendaja

Kirjeldus: Arendaja identifitseerib ennast sisestades kasutajanime ja parooli. Kui sellise kasutajanime ja parooliga kasutaja on andmebaasis olemas ning kasutajal on SUPERUSER õigused, siis lubatakse arendajal süsteemi siseneda, vastasel juhul mitte.

Märkus: Kasutusjuht "Kasutaja identifitseerimine" on kasutusel ka järgnevates allsüsteemides: lehtede funktsionaalne allsüsteem, regioonide funktsionaalne allsüsteem, navigatsioonide funktsionaalne allsüsteem.

Kasutusjuht: Rakenduste loetelu vaatamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab saada ülevaadet, mis rakendused on loodud. Süsteem kuvab arendajale loetelu rakendustest, kus on esitatud rakenduse nimi.

Kasutusjuht: Rakenduse lisamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab luua uue rakenduse. Arendaja valib rakendusele nime, aliase, andmebaasi, mille põhjal rakendus luuakse ning sisestab andmebaasi kasutajanime ja parooli, kellena süsteem andmebaasiga suhtleb. Kui sisestatud andmed on korrektsed ning sellise kasutajanime ja parooliga kasutaja eksisteerib, siis luuakse uus rakendus. Vigade korral kuvatakse vastavad veateated.

Kasutusjuht: Rakenduse muutmine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib rakenduse, mida ta soovib muuta. Arendajale kuvatakse rakenduse nimi, alias, andmebaas, mille põhjal rakendus on loodud ning andmebaasi kasutajanimi. Arendaja saab kuvatud andmeid muuta. Salvestamiseks peab ta sisestama ka andmebaasi kasutajale vastava parooli. Kui sisestatud andmed on korrektsed, siis muudatused salvestatakse. Vigade korral kuvatakse vastavad veateated.

Kasutusjuht: Rakenduse kustutamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib rakenduse, mida ta soovib kustutada. Enne kustutamist küsitakse arendajalt kinnitust. Kui arendaja kinnitab kustutamise, siis rakendus ning sellega seotud info kustutatakse.

Kasutusjuht: Rakenduse autentimismeetodi muutmine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib rakenduse, mille autentimismeetodit ta soovib muuta. Arendajale kuvatakse hetkel kasutusel olev autentimismeetod koos autentimisfunktsiooniga ning sisselogmislehe malliga. Arendaja saab eelpool nimetatud andmeid muuta.

3.5 Lehtede funktsionaalne allsüsteem

3.5.1 Eesmärgid

- Võimaldada arendajal saada ülevaade rakendusele kuuluvatest lehtekülgedest.

- Võimaldada arendajal luua uusi lehekülgi.
- Võimaldada arendajal muuta olemasolevate lehekülgede seadeid.
- Võimaldada arendajal kustutada olemasolevaid lehekülgi.

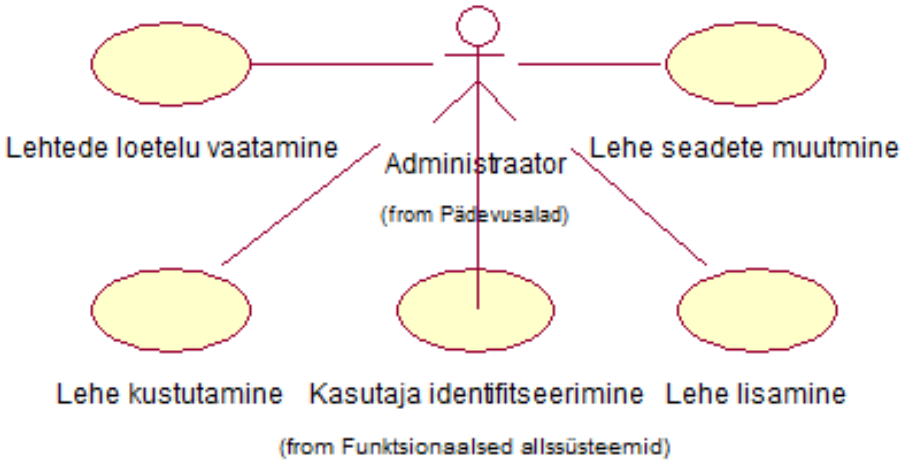
3.5.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem teenindab lehtede registrit.

Allsüsteem kasutab mallide registrit.

3.5.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud lehtede funktsionaalse allsüsteemi kasutusjuhtude eskiismudel (vt Joonis 5) ja seal esitatud kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis.



Joonis 5. Lehtede funktsionaalse allsüsteemi kasutusjuhtude eskiismudel

Kasutusjuht: Lehtede loetelu vaatamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab saada ülevaadet, mis leheküljed on rakenduse alla loodud. Süsteem kuvab arendajale loetelu lehtedest, kus tuuakse välja lehe id, alias, pealkiri ja info selle kohta, kas leht on avaleht ning kas leht nõuab kasutaja autentimist.

Kasutusjuht: Lehe lisamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab luua rakenduse alla uue lehe. Arendaja sisestab lehe pealkirja, aliase, valib lehe malli, mida kasutatakse lehe kuvamisel ja valib kas leht on avaleht ning kas leht nõuab autentimist. Kui andmed on korrektsed, siis luuakse uus leht. Vastasel korral kuvatakse vastavad veateated.

Kasutusjuht: Lehe seadete muutmine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib lehtede loetelust lehe, mida ta soovib muuta. Arendajale kuvatakse lehe pealkiri, alias, mall ja info selle kohta kas tegu on avalehega ning kas leht nõuab kasutaja autentimist. Kuvatavaid andmeid saab muuta. Kui andmed on korraktsed, siis need salvestatakse. Vastasel korral kuvatakse vastavad veateated.

Kasutusjuht: Lehe kustutamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib lehtede loetelust lehe, mida ta soovib kustutada. Enne kustutamist küsitakse arendajalt kinnitust. Kui arendaja kinnitab kustutamise, siis leht ning sellega seotud info kustutatakse.

3.6 Regioonide funktsionaalne allsüsteem

3.6.1 Eesmärgid

- Võimaldada arendajal saada ülevaade lehel olevatest regioonidest.
- Võimaldada arendajal luua navigatsiooni tüüpi regioone.
- Võimaldada arendajal luua HTML tüüpi regioone.
- Võimaldada arendajal luua raporti tüüpi regioone.
- Võimaldada arendajal luua vormi tüüpi regioone.
- Võimaldada arendajal muuta olemasolevaid regioone.
- Võimaldada arendajal kustutada olemasolevaid regioone.

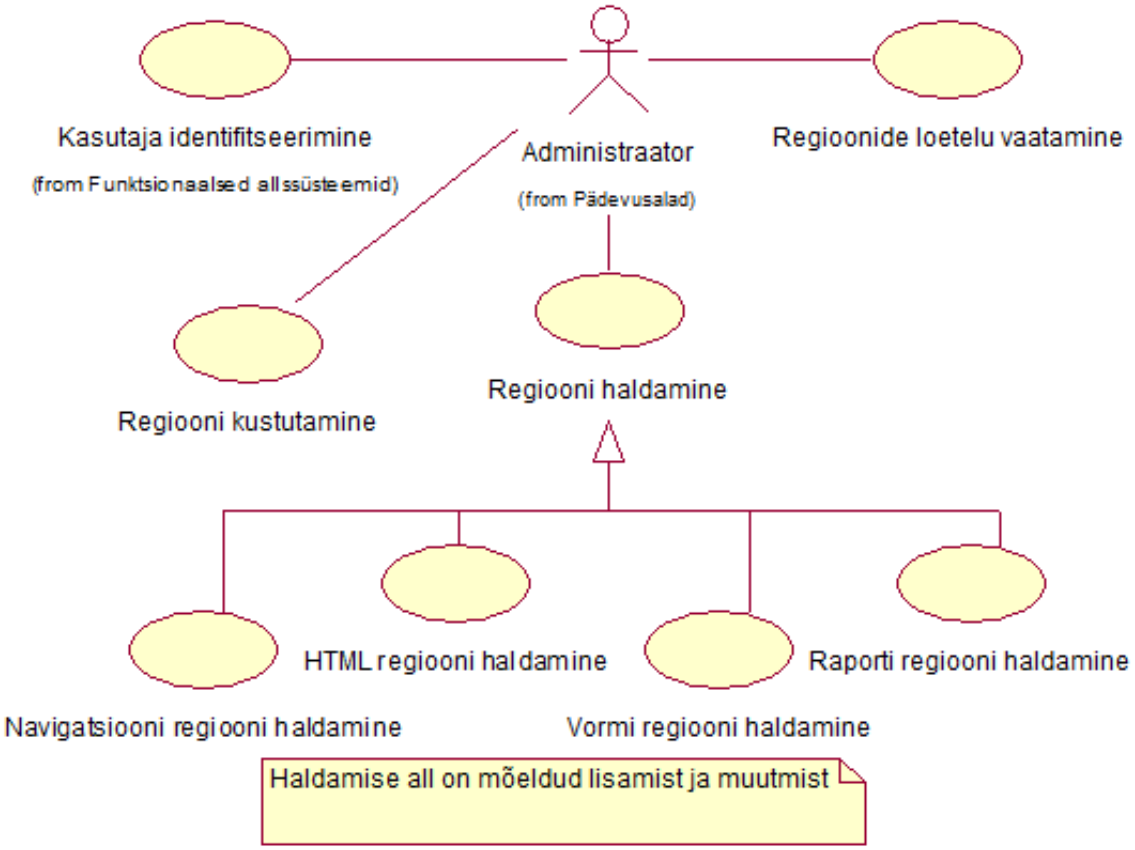
3.6.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem teenindab regioonide registrit.

Allsüsteem kasutab mallide registrit, navigatsioonide registrit, andmebaasiobjektide registrit.

3.6.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud regioonide funktsionaalse allsüsteemi kasutusjuhtude eskiismudel (vt Joonis 6) ja seal esitatud kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis.



Joonis 6. Regioonide funktsionaalse allsüsteemi kasutusjuhtude eskiismudel

Kasutusjuht: Regioonide loetelu vaatamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab saada ülevaadet, mis regioonid on valitud lehe alla loodud. Arendajale kuvatakse regioonide loetelu, kus esitatakse regiooni asukoht lehel, regiooni tüüp, järjekorranumber, nimi ning info selle kohta, kas regioon on nähtav või peidetud.

Kasutusjuht: Regiooni kustutamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib regioonide loetelust regiooni, mida ta soovib kustutada. Enne kustutamist küsitakse arendajalt kinnitust. Kui arendaja kinnitab kustutamise, siis regioon ning sellega seotud info kustutatakse.

Kasutusjuht: Regiooni haldamine

Tegutsejad: Arendaja

Kirjeldus: Regiooni lisamisel või muutmisel kuvatakse arendajale vorm, kus küsitakse regiooni nime, järjekorranumbrit, regiooni malli, mida kasutatakse regiooni kuvamisel ning infot selle kohta, kas regioon on nähtav või peidetud. Regiooni muutmise korral on vormi väljad eelnevalt täidetud. Kui esitatud andmed on korrektsed, siis need salvestatakse. Vastasel korral kuvatakse vastavad veateated.

Kasutusjuht: HTML regiooni haldamine

Tegutsejad: Arendaja

Kirjeldus: HTML regiooni lisamisel või muutmisel kuvatakse arendajale vorm, kus lisaks kasutusjuhus "Regiooni haldamine" esitatud andmetele küsitakse arendajalt ka teksti, mida regioonis kuvada.

Kasutusjuht: Navigatsiooni regiooni haldamine

Tegutsejad: Arendaja

Kirjeldus: Navigatsiooni regiooni lisamisel või muutmisel kuvatakse arendajale vorm, kus lisaks kasutusjuhus "Regiooni haldamine" esitatud andmetele küsitakse arendajalt ka navigatsiooni malli, kuvamise tüüpi ja navigatsiooni, mille alusel regioon luuakse ning infot selle kohta, kas regiooni kuvamisel tuleb korrata navigatsioonipunkti malli viimast taset, juhul kui navigatsioonipunkti sügavuse jaoks pole eraldi malli defineeritud.

Kasutusjuht: Raporti regiooni haldamine

Tegutsejad: Arendaja

Kirjeldus: Raporti regiooni lisamisel või muutmisel kuvatakse arendajale vorm, kus lisaks kasutusjuhus "Regiooni haldamine" esitatud andmetele küsitakse arendajalt ka raporti kuvamisel kasutatavat malli, raporti aluseks olevat vaadet, infot selle kohta, kas raporti päist tuleb kuvada või mitte, mitu rida kuvatakse ühel leheküljel, mis URL-parameetriga

antakse edasi hetkel aktiivset lehekülge ning mis veergudest raport koosneb. Pärast vaate valimist saab luua raportile veerge. Raporti veerg võib olla kas valitud vaate veerg või link. Raporti veeru loomisel tuleb sisestada veeru pealkiri, järjekorranumber ning info selle kohta, kas kuvatavas tekstis muudetakse HTML-erimärgid (&, <, >, ", ’) ohutuks või mitte. Lingi korral tuleb lisaks sisestada ka URL ja ning lingi tekst ning võib lisada lisaatribuudid lingi vormindamiseks. Raport peab sisaldama vähemalt ühte veergu.

Kasutusjuht: Vormi regiooni haldamine

Tegutsejad: Arendaja

Kirjeldus: Vormi regiooni lisamisel või muutmisel kuvatakse arendajale vorm, kus lisaks kasutusjuhus “Regiooni haldamine” esitatud andmetele küsitakse arendajalt ka vormi kuvamisel kasutatavat malli, vormi saatmisnupu malli, saatmisnupul kuvatavat teksti, teadet, mida kuvatakse vormi eduka töötlemise korral, teadet, mida kuvatakse, kui vormi töötlemisel tekib viga, URL, kuhu pärast vormi edukat töötlemist edasi suunatakse, funktsiooni, mille alusel vorm luuakse ning info selle kohta, kas vorm tuleb eelnevalt andmetega täita. Pärast funktsiooni valimist kuvatakse funktsiooni parameetrid ning arendaja peab valima, kuidas neid vormis kuvatakse. Arendaja peab sisestama vormi välja nime, kirjelduse, järjekorranumbri, valima välja tüübi ja malli. Lisaks saab ta valida, kas väli on kogustuslik või mitte, nähtav või peidetud, sisestada välja vaikimisi väärtuse ja kasutajat abistava teksti. Juhul kui välja tüübiks on mitme valikuvõimalusega element, siis peab kasutaja valima vaate ja veerud, mille põhjal valikud luuakse. Juhuk kui on valitud vaade, mille põhjal vorm eeltäidetakse, siis peab arendaja ära kirjeldama päringu tingimused, mille alusel leitakse vaatest õige rida ning määrama, millised vormi väljad infoga täidetakse.

3.7 Navigatsioonide funktsionaalne allsüsteem

3.7.1 Eesmärgid

- Võimaldada arendajal saada ülevaade rakendusele kuuluvatest navigatsioonidest.
- Võimaldada arendajal luua uusi navigatsioone.
- Võimaldada arendajal muuta olemasolevate navigatsioone.
- Võimaldada arendajal kustutada olemasolevaid navigatsioone.
- Võimaldada arendajal lisada olemasoleva navigatsiooni alla navigatsioonipunkte.
- Võimaldada arendajal muuta olemasolevaid navigatsioonipunkte.

- Võimaldada arendajal kustutada olemasolevaid navigatsioonipunkte.

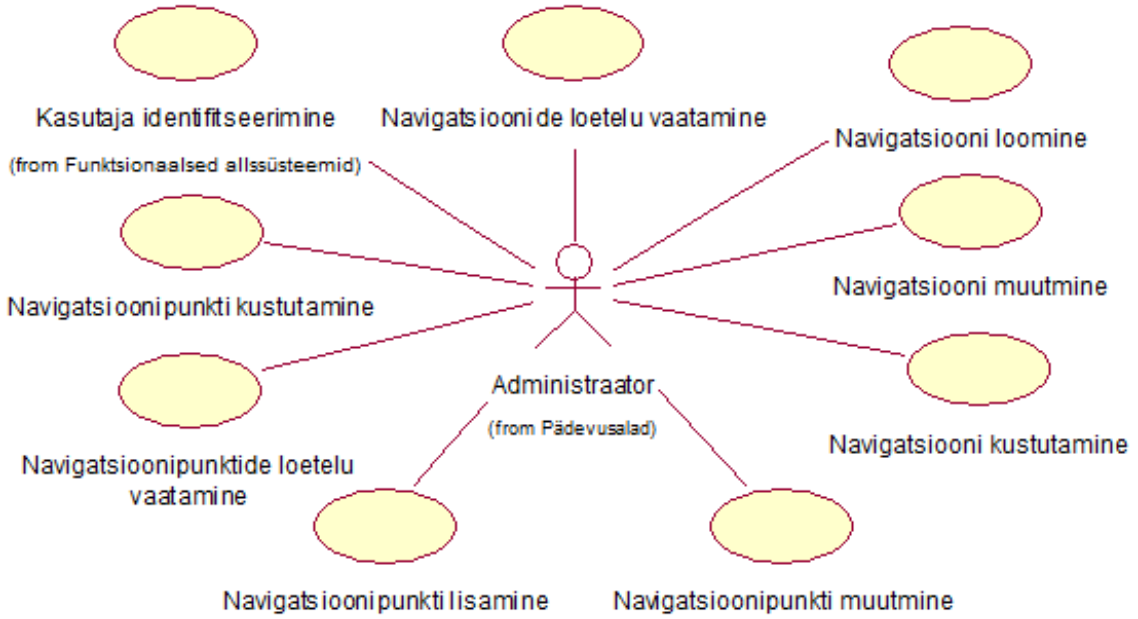
3.7.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem teenindab navigatsioonide registrit.

Allsüsteem kasutab lehtede registrit.

3.7.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud navigatsioonide funktsionaalse allsüsteemi kasutusjuhtude eskiismudel (vt Joonis 7) ja seal esitatud kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis.



Joonis 7. Navigatsioonide funktsionaalse allsüsteemi kasutusjuhtude eskiismudel

Kasutusjuht: Navigatsioonide loetelu vaatamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab saada ülevaadet, mis navigatsioonid on valitud rakenduse alla loodud. Arendajale kuvatakse navigatsioonide nimede loetelu.

Kasutusjuht: Navigatsiooni loomine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab luua uut navigatsiooni. Arendajale kuvatakse vorm, kus küsitakse uue navigatsiooni nime. Kui sisestatud andmed on korrektsed, siis luuakse navigatsioon. Vastasel korral kuvatakse vastvad veateated.

Kasutusjuht: Navigatsiooni muutmine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib navigatsiooni, mida ta soovib muuta. Arendajale kuvatakse navigatsiooni nimi. Kuvatavaid andmeid saab muuta. Kui andmed on korraktsed, siis need salvestatakse. Vastasel korral kuvatakse vastavad veateated.

Kasutusjuht: Navigatsiooni kustutamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib navigatsioonide loetelust navigatsiooni, mida ta soovib kustutada. Enne kustutamist küsitakse arendajalt kinnitust. Kui arendaja kinnitab kustutamise, siis navigatsioon ning sellega seotud info kustutatakse.

Kasutusjuht: Navigatsioonipunktide loetelu vaatamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab saada ülevaadet, millised navigatsioonipunktid on navigatsiooni alla loodud. Arendajale kuvatakse loetelu navigatsioonipunktidest, kus on esitatud navigatsioonipunkti järjekorranumber, nimi ja URL või leht, millele ta viitab. Lisaks on välja toodud, millise navigatsioonipunkti alla ta kuulub.

Kasutusjuht: Navigatsioonipunkti lisamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab navigatsiooni alla lisada navigatsioonipunkti. Arendajale kuvatakse vorm, kus küsitakse navigatsioonipunkti nime, järjekorranumbrit, ülem-navigatsioonipunkti ning URL-i või lehte, millele viidatakse. Kui sisestatud andmed on korrektsed, siis luuakse uus navigatsioonipunkt. Vastasel korral kuvatakse vastavad veateated.

Kasutusjuht: Navigatsioonipunkti muutmine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab muuta olemasolevat navigatsioonipunkti. Arendajale kuvtakse vorm, kus esitatakse olemasoleva navigatsioonipunkti nimi, järjekorranumber, ülem-

navigatsioonipunkt ning URL või leht, millele viidatakse. Arendaja saab muuta eelpool nimetatud andmeid. Kui andmed on korrektsed, siis andmed salvestatakse. Vastasel korral kuvatakse vastavad veateated.

Kasutusjuht: Navigatsioonipunkti kustutamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib navigatsioonipunktide loetelust navigatsioonipunkti, mida ta soovib kustutada. Enne kustutamist küsitakse arendajalt kinnitust. Kui arendaja kinnitab kustutamise, siis navigatsioonipunkt ning sellega seotud info kustutatakse.

3.8 Mittefunktsionaalsed nõuded

Tabelis 2 on välja toodud süsteemile esitatud mittefunktsionaalsed nõuded, mida loodav süsteem peab täitma.

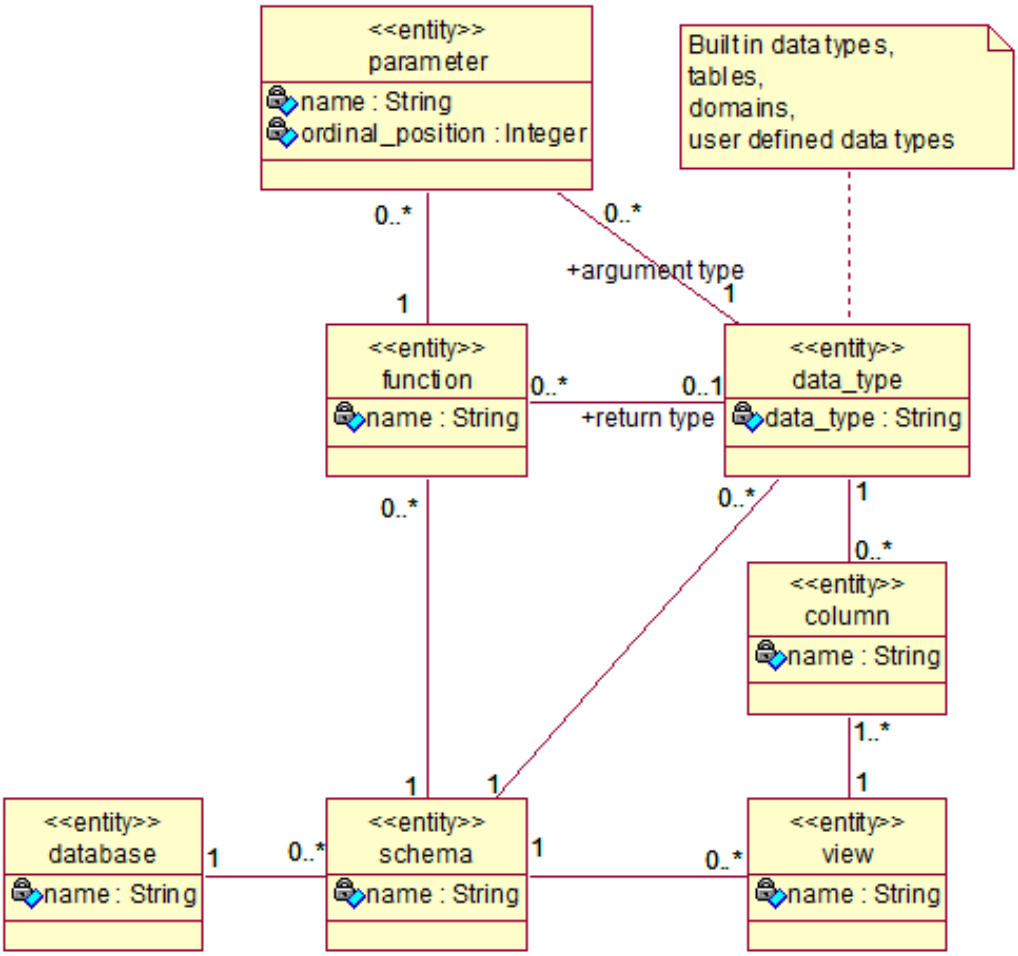
Tabel 2. Mittefunktsionaalsed nõuded

Tüüp	Nõude kirjeldus
Serveri tarkvara	Andmete hoidmiseks peab kasutama andmebaasisüsteemi PostgreSQL 9.4 või uuemat. Rakendus tuleb luua kasutades PHP 5.5.0 või uuemat.
Keel	Süsteemi kasutajaliides peab olema ingliskeelne.
Kasutajaliides	Kasutajaliides peab olema veebipõhine ning arvestama erinevate resolutsioonidega.
Toetatud veebibrauserid	■ Microsoft Internet Explorer 11 või uuem. ■ Mozilla Firefox 43 või uuem. ■ Google Chrome 49 või uuem.
Andmebaasioperatsioonide töökiirus	Andmebaasioperatsioonid peavad süsteemil aega võtma alla 5 sekundi.

4 Andmebaas

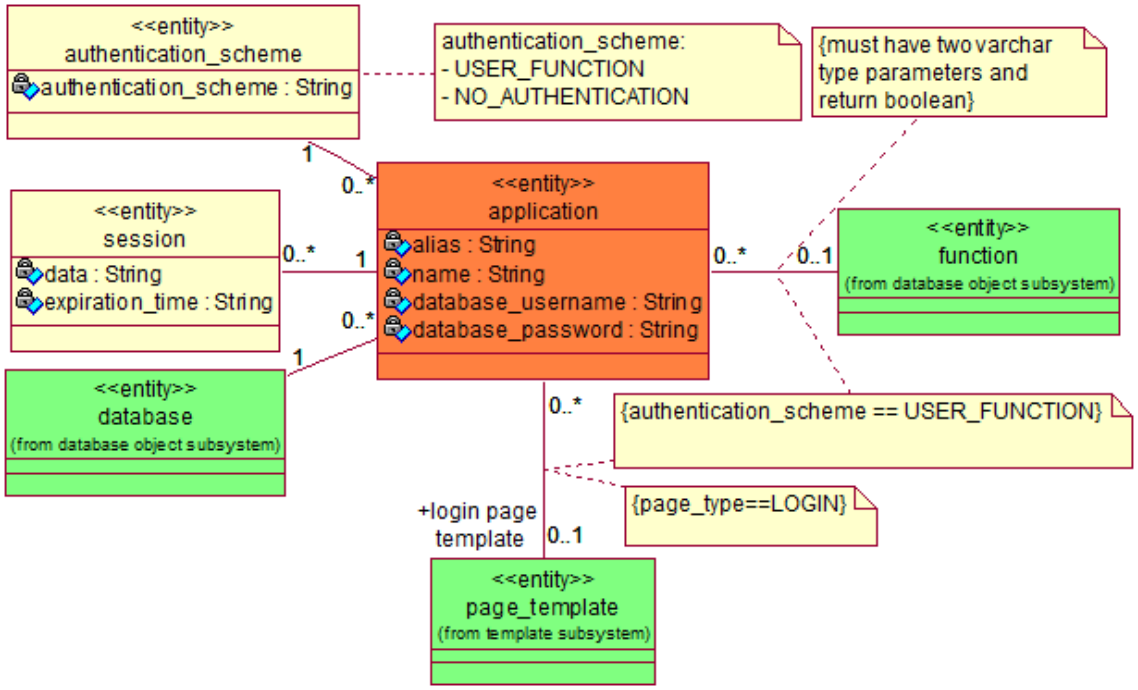
Järgnevalt esitatakse loodava süsteemi andmemudeli olemi-suhte diagrammid registrite kaupa ning kirjeldatakse, mille jaoks vastavaid olemitüüpe ning atribuute vaja läheb. Punasega on tähistatud registri põhiobjekt, kollasega registrisse kuuluvad mitte-põhiobjektid ning rohelisega teistesse registritesse kuuluvad objektid, mida vaadeldav register vajab. Diagrammidel on objektid kirjeldatud inglise keeles, kuna nende põhjal genereeriti tabelid ning lootes, et loodav süsteem leiab laiemat kasutust, siis oleks kasulik, kui süsteemi poolt kasutatavad osad oleks kirjutatud inglise keeles. Olemi-suhte diagrammide põhjal genereeritud tabelid on Lisas X.

4.1 Andmebaasiobjektide register



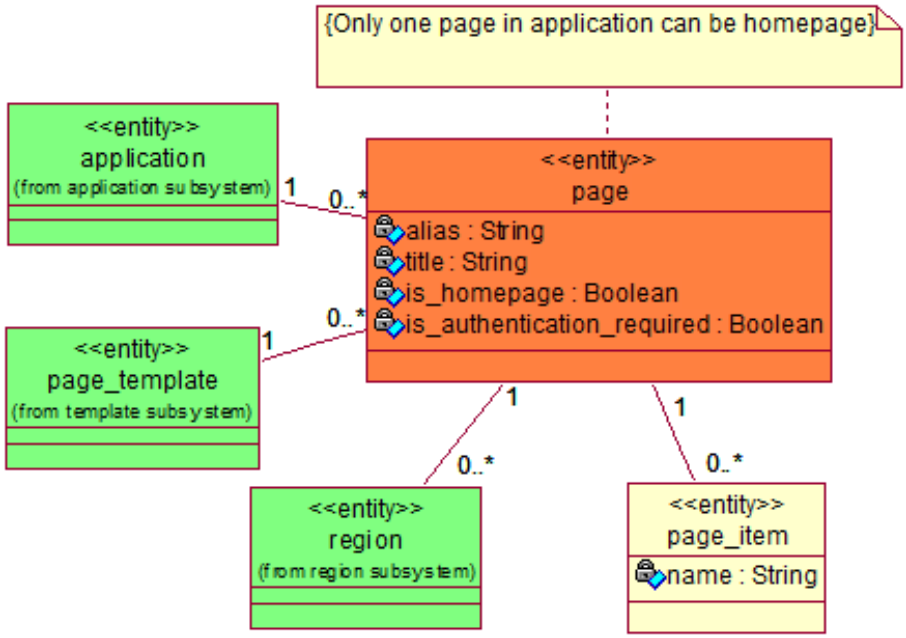
Joonis 8. Andmebaasiobjektide registri olemi-suhte diagramm

4.2 Rakenduste register



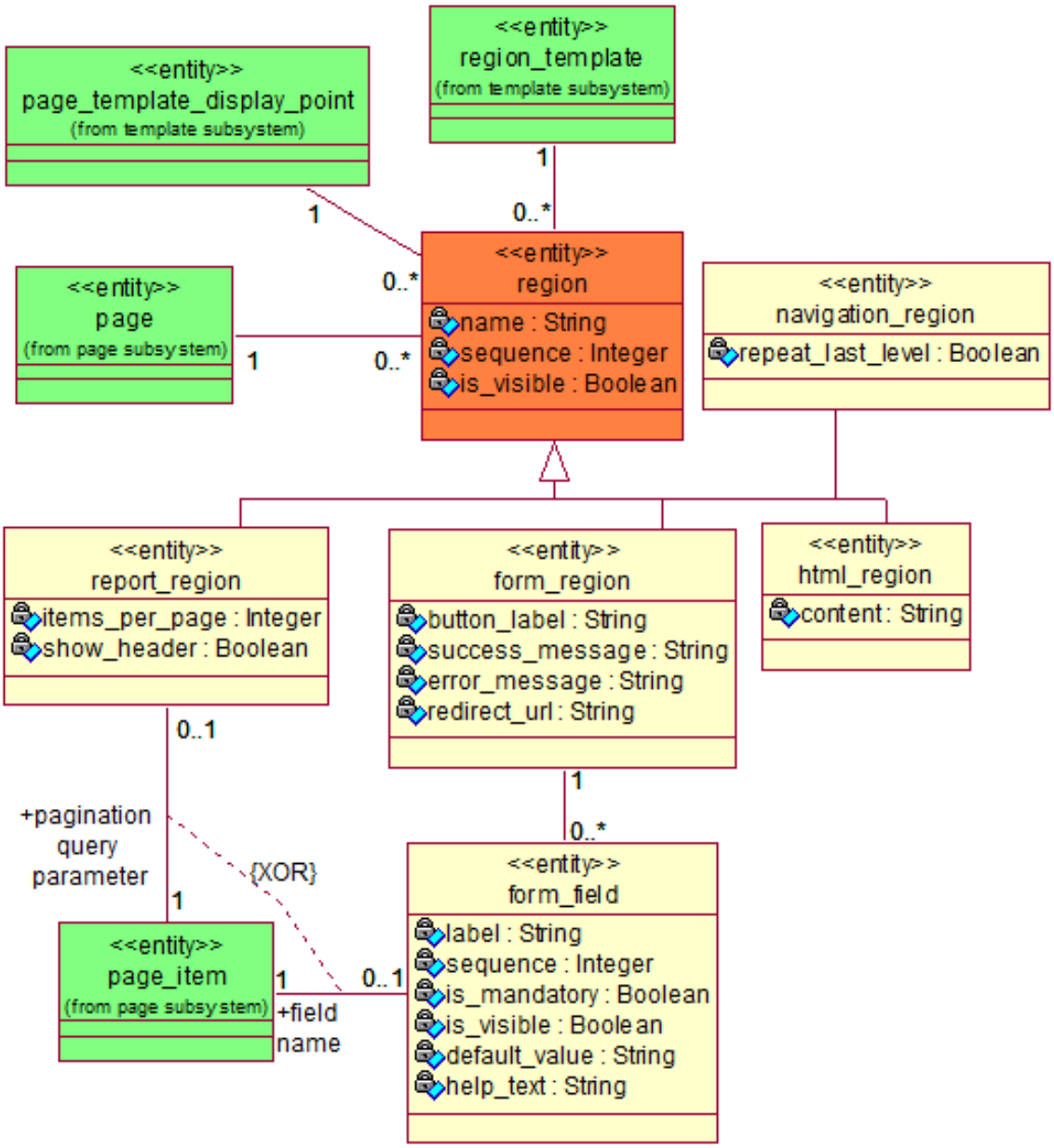
Joonis 9. Rakenduste registri olemi-suhte diagramm

4.3 Lehtede register

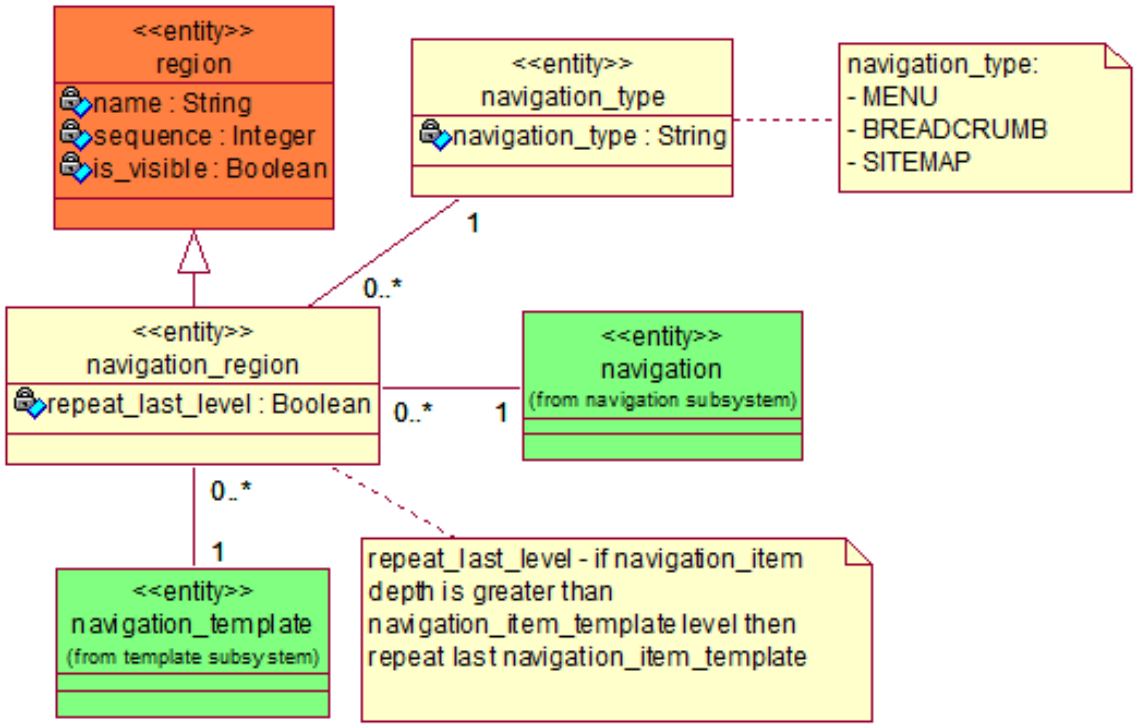


Joonis 10. Lehtede registri olemi-suhte diagramm

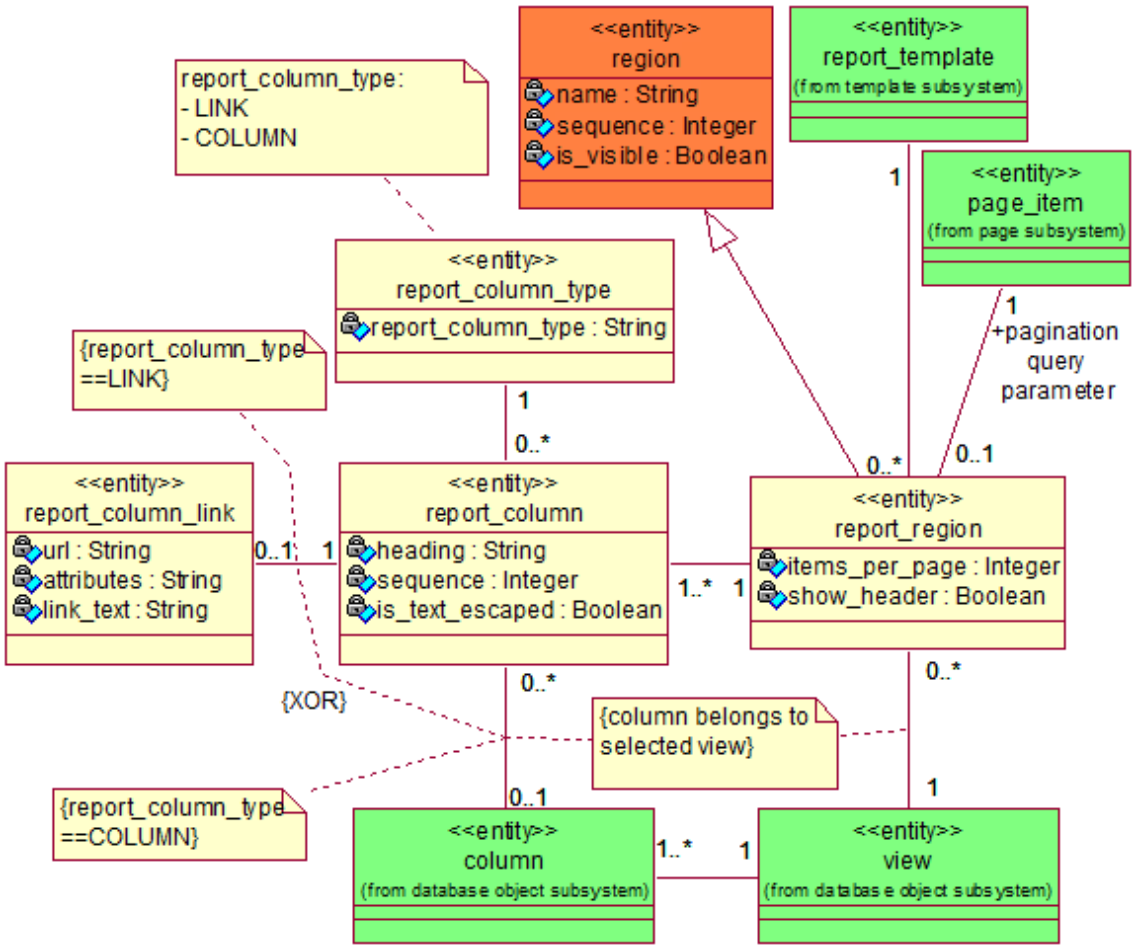
4.4 Regioonide register



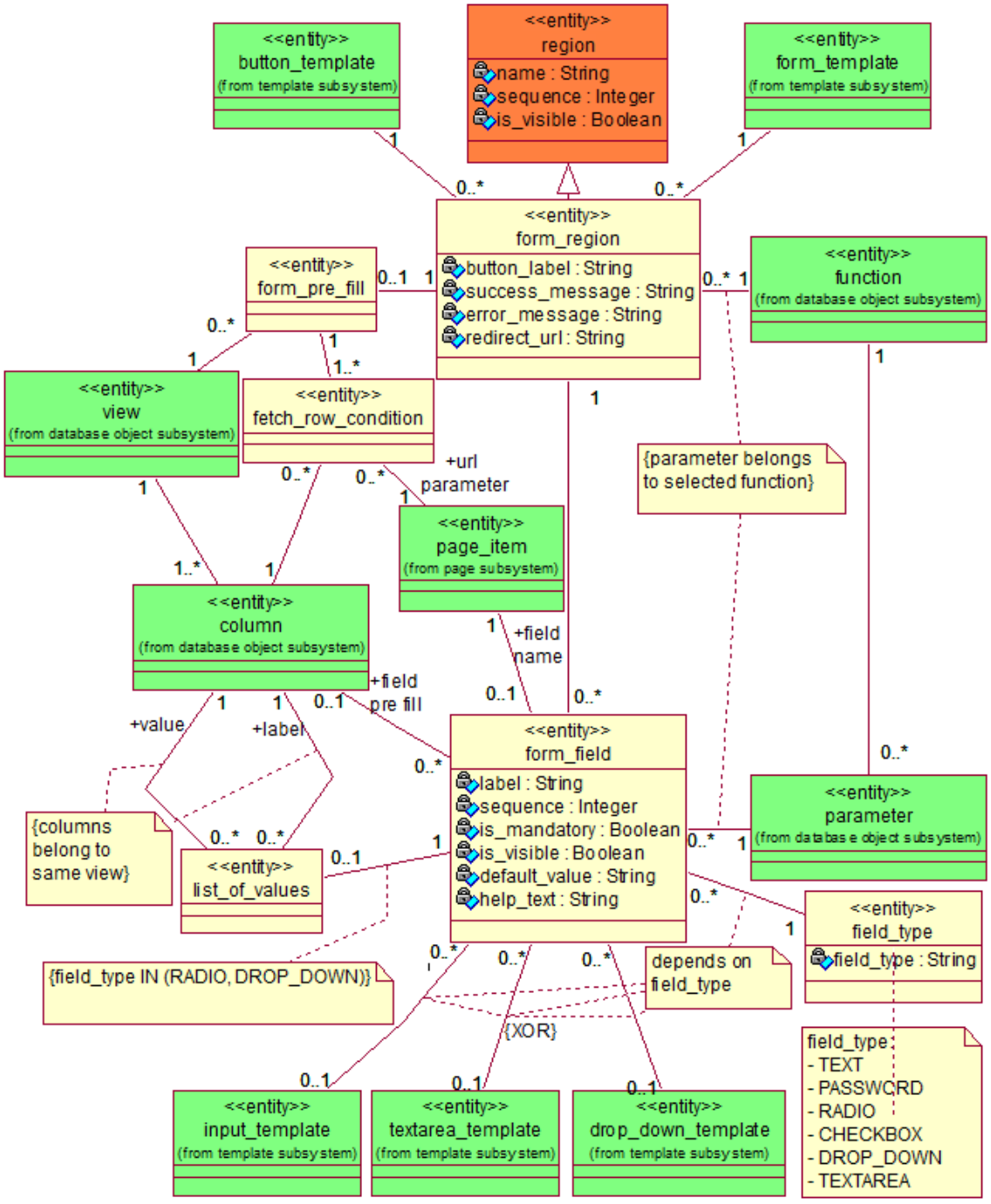
Joonis 11. Regioonide registri olemi-suhte diagramm osa 1



Joonis 12. Regioonide registri olemi-suhte diagramm osa 2



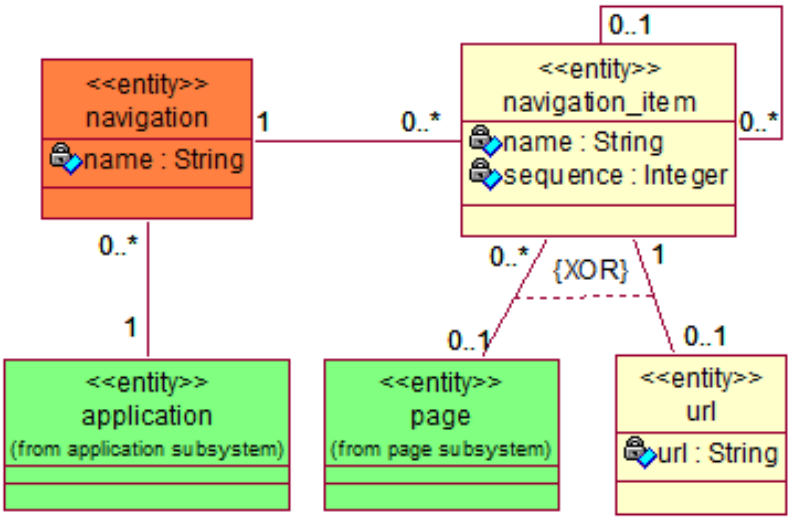
Joonis 13. Regioonide registri olemi-suhte diagramm osa 3



Joonis 14. Regioonide registri olemi-suhte diagramm osa 4

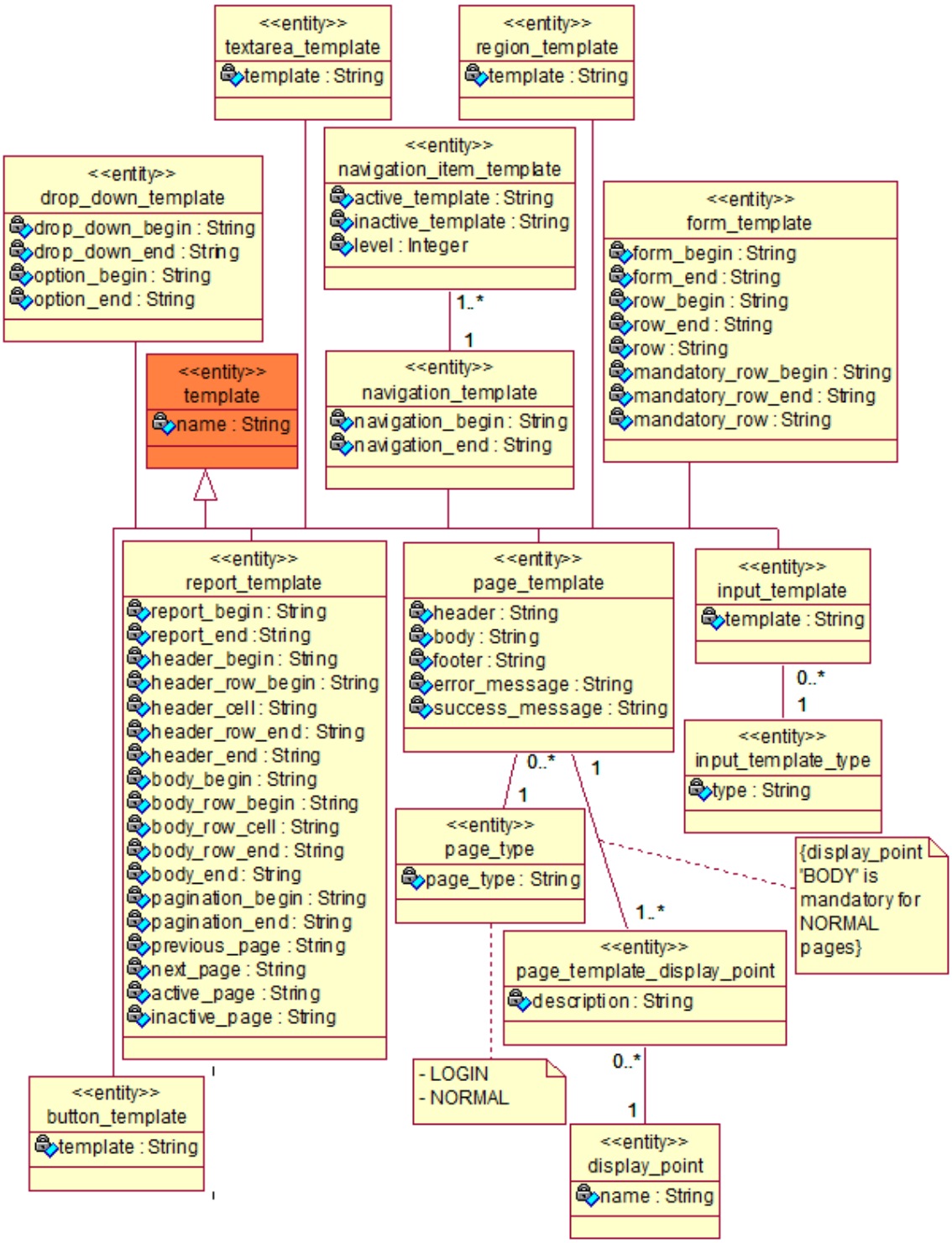
4.5 Navigatsioonide register

Navigatsioonide register



Joonis 15. Navigatsioonide registri olemi-suhte diagramm

4.6 Mallide register



Joonis 16. Mallide registri olemi-suhte diagramm

5 Arendusvahendid ja -protsess

5.1 Vagrant

Vagrant on käsureaprogramm, millega saab hallata virtuaalmasina elutsüklit. Vagrant isoleerib programmilised sõltuvused ja nende konfiguratsioonid ühtsesse eraldiseisvasse keskkonda. Keskkonna konfigureerimiseks saab kasutada käsurea käsklusi, *Ansible*-t [2], *Puppet*-it [30], *Chef* -i [7], *Docker*-it [8] ja *Salt*-i [31]. Tänu Vagrantile saavad kõik luua endale täpselt ühesuguse keskkonna, kus programme jooksutada, vähendades võimalust, et ühes arvutis programm jookseb, teises aga mitte. [35]

5.2 AngularJS

Angular on raamistik loomaks dünaamilisi veebirakendusi. See võimaldab laiendada HTML süntaksit, et panna elemendid käituma vastavalt arendaja soovile. Angular kasutab kahe suunalist andmesidumist (*data binding*). See tähendab et muudatused javascripti koodis kajastuvad automaatselt HTML-is ning vastupidi. Tänu sellele peab arendaja vähem tegelema DOM-i manipuleerimisega. [1].

5.3 Bootstrap

Bootstrap on mobiilisõbralik kasutajaliidese raamistik, mille abil saab luua dünaamilist veebidisaini (*responsive web design*), mis arvestab kasutaja ekraani suurusega ning kohandab end jooksvalt vastavalt sellele. Bootstrap-is on realiseeritud mitmed komponendid, mis kiirendavad kasutajaliidese loomist. Bootstrap kasutab HTML-i, javascripti js CSS-i. [4]

5.4 Bower

Bower on paketihaldussüsteem (*package manager*), mis on mõeldud veebis kasutatavate failide, nagu näiteks HTML, CSS, javascript, fondid ja pildid, haldamiseks. Bower-i kasutamiseks peab masinasse olema installitud node, npm ja git. Paketide haldus toimub

bower.json failis, kus kirjeldatakse ära soovitud paketid ning nende versioonid. [5]

5.5 TravisCI

TravisCI on pideva integratsiooni (*continuous integration*) keskkond, mille abil saab luua virtuaalse keskkonna koodi kompilleerimiseks, testimiseks ja juurutamiseks. Keskkonna seadistamine toimub faili .travis.yml abil, kus määratakse ära virtuaalkeskkonna operatsioonisüsteem, teegid, mis tuleb installida ning käivitatavad käsud.

5.6 Arendusprotsess

Esimese asjana sai paika pandud esmased nõuded, mida süsteem peaks võimaldama teha. Kuna süsteemi esimene potentsiaalne kasutuskoht oleks TTÜ-s õpetatavas aines "Andmebaasid II", siis konsulteeriti antud õppeaine õppejõuga ning kaardistati enim levinud kasutusjuhud, mida antud aine raames tuleb üliõpilastel realiseerida. Kui nõuded olid paigas, siis loodi nende põhjal andmemudel ning kasutajaliidese prototüüp, mis võeti hiljem loodavas süsteemis kasutusele. Prototüüp kasutas andmete kuvamiseks võltsandmeid (*mock data*). See andis hea ettekujutuse loodava süsteemi võimekusest ning aitas juhtida tähelepanu aspektidele, millele ilma prototüübi abita ei oleks kohe tuldud.

Selleks et loodavat süsteemi oleks ka teistel arendajatel lihtsam kasutusele võtta ning täiustada sai arenduskeskkonna loomiseks kasutusele võetud Vagrant [34], mille abil loodi virtuaalmasin koos kõigi arenduseks vajalike teekidega. Virtuaalmasina konfigureerimiseks kasutati bash-i skripti.

Dünaamilise kasutajaliidese loomiseks kasutati AngularJS 1.4 [1]. Lihtsustamaks kasutajaliidese ühtset väljanägemist erinevates brauserites ning eri suurustes ekraanidega, kasutati Bootstrap 3 [4]. Kasutajaliidese poolt kasutatavaid sõltuvusi hallati [5]-i abil. Koodi kvaliteedi kontrollimiseks ja säilitamiseks kirjutati testid, mida jooksutati Karma [12] abil.

Andmebaasi loomisel lähtuti ideest, et kogu suhtlus andmebaasiga peab käima läbi andmebaasiliidese 2.1. Andmete salvestamiseks ning küsimiseks tuleb kasutajal välja kutsuda vastav andmebaasifunktsioon. Kasutamaks ära PostgreSQL-i võimalust väljastada JSON tüüpi andmeid luuakse kasutajaliidese jaoks vajalik vastus juba andmebaasis. Tänu sellele pole andmetega manipuleerimine süsteemis laiali jaotatud vaid toimub üksnes andmebaasi poolel.

Andmebaasi ja kasutajaliidese vaheline suhtlus toimub läbi PHP-s [22] kirjutatud rakenduse. Kuna antud rakenduse kiht on üpriski õhuke, siis sai selle loomiseks valitud ka lihtsakoeline raamistik Slim Framework 3 [32]. PHP-s kirjutatud koodi testimiseks kasutati PHPUnit-it [23] ning Mockery-t [15]. Koodi sõltuvusi hallati Composer-i abil [**?**].

Koodi hoidmiseks kasutatakse GitHub-i [9]. Iga kord, kui koodihoidlasse midagi üles laetakse luuakse TravisCI-s [33] virtuaalkeskkond, kuhu tõmmatakse GitHub-st loodava süsteemi kood ning testide eduka jooksutamise korral juurutatakse serverisse. Tänu sellele on serveris alati näha süsteemi viimane töötav version.

6 Kasutajaliidese disain

Kasutajaliides koosneb neljast põhiosast, milleks on moodulid, vaated, teenused ning tõlkefailid. Iga allsüsteemi haldamiseks ning kasutajate autentimiseks loodi moodul. Moodulis määrati ära, mis lehe korral mingit kontrollerit jooksutatakse ning millist vaadet kasutatakse. Vaadete osas on ära kirjeldatud erinevate lehtede väljanägemine ning teenuste osas suhtlus. Kuigi hetkel ei ole arendajatel võimalus kasutajaliidese keelt muuta, tuleb kogu kuvatav tekst tõlkefailidest. Seetõttu on lihtne tõlkida süsteemi ka teistesse keeltesse.

Lehele minnes kontrollitakse, mis URL-le kasutaja tuli ja käivitatakse vastav kontroller ning laetakse sisse vaade, mida antud kontroller kasutab. Kontrolleri käivitamisel antakse talle ette teenuste objektid, mida ta oma tööks vajab. Selleks kasutatakse sõltuvuste süstimise (*dependency injection*) disainimustrit. See tähendab, et kontrolleri loomisel vaadatakse, milliseid teenuseid ta vajab ja luuakse vajalikud instantsid ning antakse kontrollerile kaasa. Seetõttu ei pea kontroller ise tegelema vajalike teenuste initsialiseerimisega.

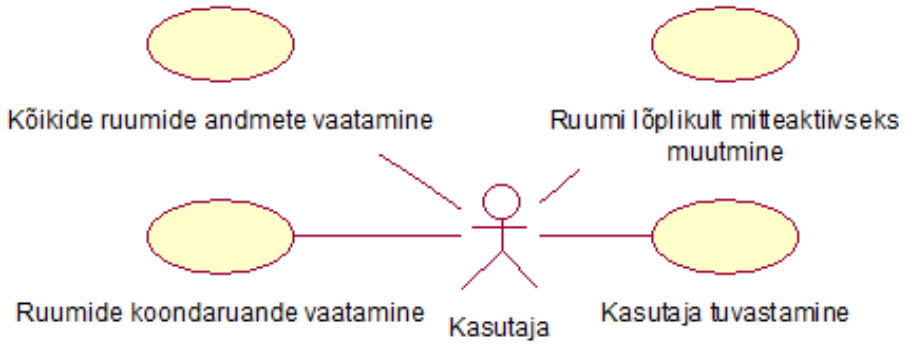
Igal kontrolleril on objekti tüüpi $scope muutuja, mille abil toimub andmete vahetamine vaatega. Kui $scope objektis andmed muutuvad, siis kuvatakse vastavad muudatused koheselt ka vaates ning vastupidi.

Teenuste abil toimub suhtlus serveriga AJAX-i abil. Tänu sellele ei pea andmete küsimisel ega andmetega manipuleerimisel kogu lehte uuesti laadima, vaid vajaliku info edastamine toimub rakenduse taustal. Andmete küsimine toimub GET-meetodi abil ning andmetega manipuleerimine POST-meetodi abil.

7 Rakenduse disain

8 Näidisrakendus

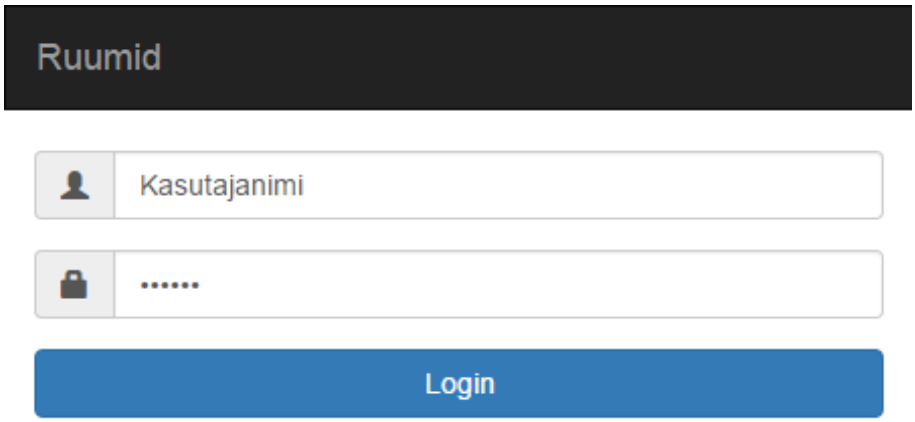
Valideerimaks, kas loodud süsteem vastab nõuetele, loon näiterakenduse ja realiseerin neli kasutusjuhtu (vt Joonis 17), mis sarnanevad üliõpilastöödes esinevatele kasutusjuhtudele. Kasutaja tuvastamine põhineb funktsioonil functions.f_is_boss, mille esimese parameetri oodatav väärtus on kasutajanimi ja teise parameetri oodatav väärtus on paool. Kõikide ruumide anmete vaatamine põhineb vaatel public.overview_of_rooms. Ruumide koondaruande vaatamine põhineb vaatel public.number_of_rooms_by_state. Ruumi mitteaktiivseks muutmine põhineb funktsioonil functions.f_permanently_inactivate_a_room, mille oodatavaks argumendiks on ruumi kood, mis tuleb valida vaatest public.active_temporariliy_inactive_rooms.



Joonis 17. Näidisrakenduse kasutusjuhtude eskiismudel

8.1 Kasutaja tuvastamine

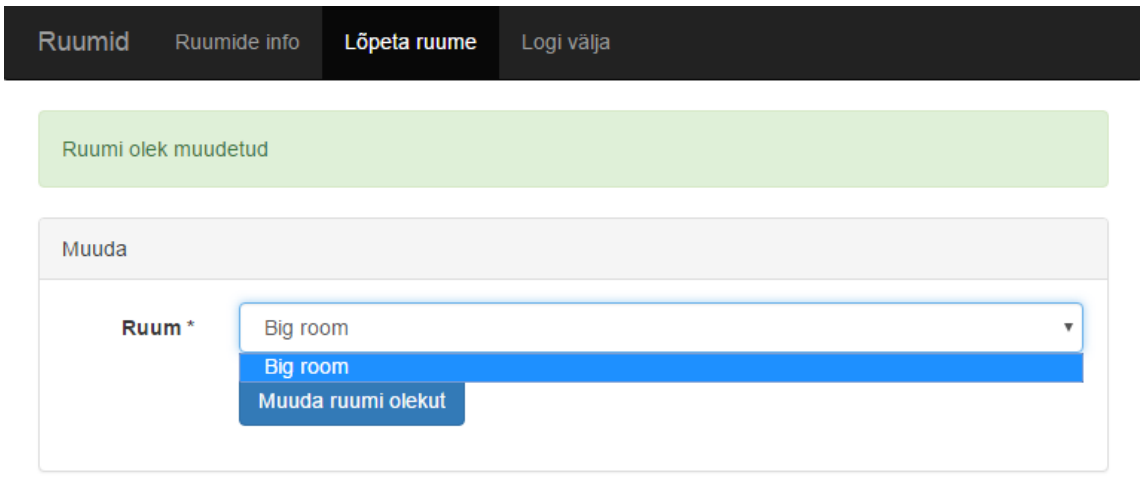
Kui kasutaja läheb lehele, mille nägemiseks peab ta olema autenditud, siis kuvatakse talle sisselogimise vorm, kuhu tuleb sisestada kasutajanimi ja parool (vt Joonis 18). Kui kasutajanimi ja parool on õiged, siis logitakse kasutaja sisse ning kasutaja näeb edaspidi autentimist nõudvate lehtede sisu.



Joonis 18. Kasutaja tuvastamine

8.2 Ruumi lõplikult mitteaktiivseks muutmine

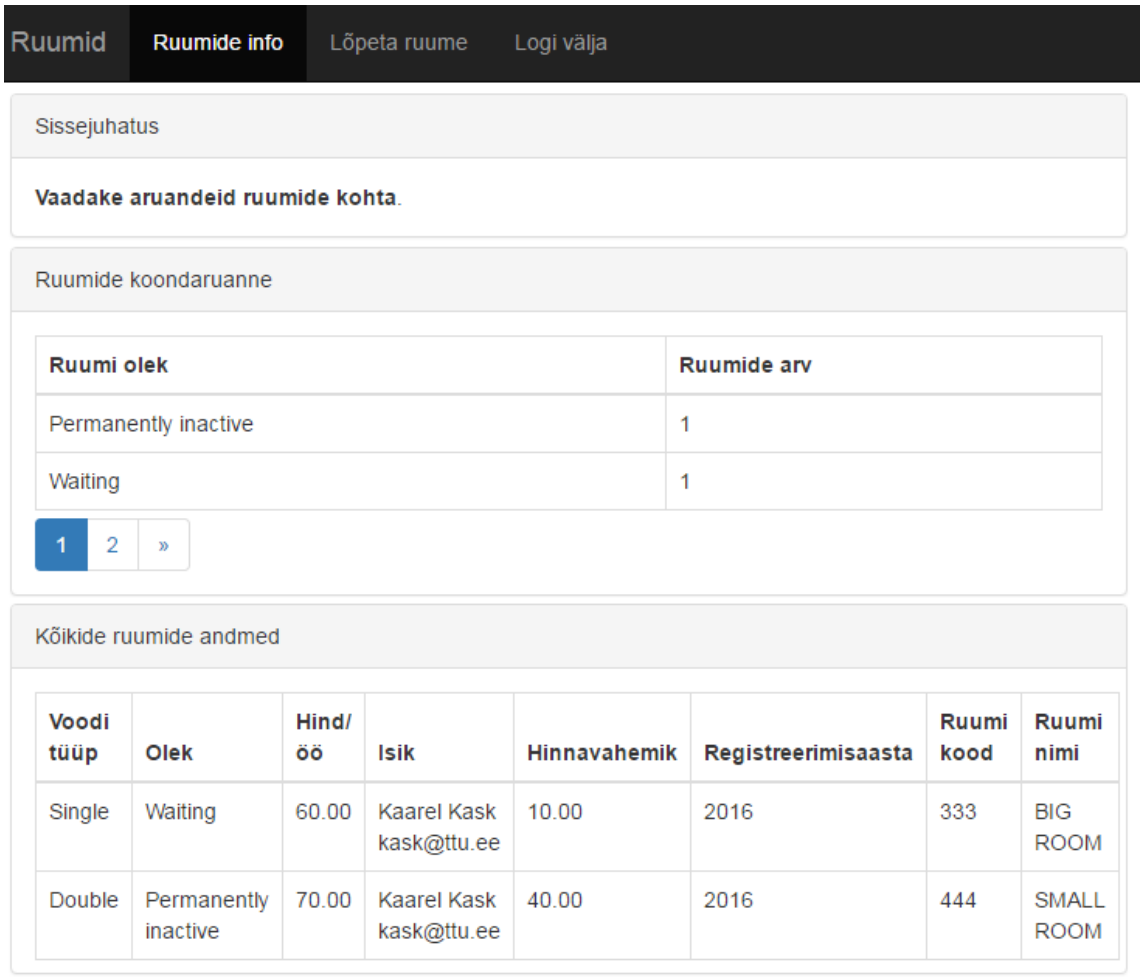
Kasutajale kuvatakse vorm, kus kust ta saab valida millist ruumi ta muuta soovib (vt Joonis 19). Ruumide loetelu saadakse vaate public.active_temporariliy_inactive_rooms põhjal. Pärast vormi saatmist kutsutakse välja funktsioon functions.f_permanently_inactivate_a_room. Kui funktsiooni lõpetab oma töö ilma vigadeta, siis tagastatakse kasutajale teade, et vormi saatmine õnnestus.



Joonis 19. Ruumi lõplikult mitteaktiivseks muutmine

8.3 Ruumide koondaruande ja kõikide ruumide vaatamine

Kasutajale kuvatakse ühel lehel nii ruumide koondaruanne kui ka kõikide ruumide info, kusjuures raportite read on võimalik jaotada lehekülgedele (vt Joonis 20)



Joonis 20. Ruumide koondaruande ja kõikide ruumide vaatamine

9 Arendusvaade

10 Kokkuvõte

Kokkuvõte

11 Summary

Kokkuvõte

Kasutatud kirjandus

[1] AngularJS. [WWW] https://angularjs.org/. (20.02.2016).

[2] Ansible is Simple IT Automation. [WWW] https://www.ansible.com. (07.03.2016).

[3] Ben Balter. Open source license usage on GitHub.com. [WWW] https://github.com/blog/1964-open-source-license-usage-on-github-com, 2015. (20.02.2016).

[4] Bootstrap. [WWW] http://getbootstrap.com//. (20.02.2016).

[5] Bower. [WWW] http://bower.io/. (06.05.2016).

[6] Larry Burns. *Building the Agile Database - How to Build a Successful Application Using Agile Without Sacrificing Data Management*. Technics Publications, LLC, 1 edition, 2011.

[7] Chef - Code Can | Chef. [WWW] https://www.chef.io/. (07.03.2016).

[8] Docker - Build, Ship, and Run Any App, Anywhere. [WWW] https://www.docker.com/. (07.03.2016).

[9] GitHub. [WWW] https://github.com/. (21.04.2016).

[10] What is free software? [WWW] http://www.gnu.org/philosophy/free-sw.html. (20.02.2016).

[11] P Hambrick. Advantages and Drawbacks of Using Stored Procedures for Processing Data. [WWW] http://www.seguetech.com/blog/06/04/Advantage-drawbacks-stored-procedures-processing-data. (07.03.2016).

[12] Karma - Spectacular Test Runner for Javascript. [WWW] https://karma-runner.github.io/0.13/index.html/. (21.04.2016).

[13] Darja Kašnikova. Vaadete mõju päringute täitmisplaanide koostamisele kahe andmebaasisüsteemi näitel. Master's thesis, Tallinna Tehnikaülikool, 2015. [WWW] http://digi.lib.ttu.ee/i/?3676. (06.03.2016).

[14] Licenses. [WWW] http://choosealicense.com/licenses/. (20.02.2016).

[15] Mockery. [WWW] http://docs.mockery.io/en/latest/. (21.04.2016).

[16] MySQL. [WWW] https://www.mysql.com/. (08.03.2016).

[17] nuBuilder. [WWW] https://www.nubuilder.net. (29.02.2016).

[18] GitHub: nuSoftware/nuBuilderPro: Web Application Builder. [WWW] https://github.com/nuSoftware/nuBuilderPro. (29.02.2016).

[19] The Open Source Definition. [WWW] https://opensource.org/osd-annotated. (20.02.2016).

[20] Oracle Application Express. [WWW] https://apex.oracle.com/en/. (20.02.2016).

[21] Oracle Database. [WWW] https://www.oracle.com/database/index.html. (20.02.2016).

[22] PHP: Hypertext Preprocessor. [WWW] http://php.net/. (20.02.2016).

[23] PHPUnit - The PHP Testing Framework. [WWW] https://phpunit.de/. (21.04.2016).

[24] PostgreSQL. [WWW] http://www.postgresql.org/. (20.02.2016).

[25] PostgreSQL: Documentation: 9.4: dblink. [WWW] http://www.postgresql.org/docs/9.4/static/contrib-dblink-function.html. (22.04.2016).

[26] PostgreSQL: Documentation: 9.4: postgres_fdw. [WWW] http://www.postgresql.org/docs/9.4/static/postgres-fdw.html. (22.04.2016).

[27] PostgreSQL: Documentation: 9.4: The Information Schema. [WWW] http://www.postgresql.org/docs/9.4/static/information-schema.html. (20.02.2016).

[28] PostgreSQL: Documentation: 9.4: Rules and Privileges. [WWW] http://www.postgresql.org/docs/9.4/static/rules-privileges.html. (21.04.2016).

[29] PostgreSQL: Documentation: 9.4: System Catalogs. [WWW] http://www.postgresql.org/docs/9.4/static/catalogs.html. (20.02.2016).

[30] Puppet Labs: IT Automation Software for System Administrators. [WWW] https://puppetlabs.com/. (07.03.2016).

[31] SaltStack automation for CloudOps, ITOps & DevOps at scale. [WWW] https://saltstack.com/. (07.03.2016).

[32] Slim Framework. [WWW] http://www.slimframework.com/. (21.04.2016).

[33] Travis CI - Test and Deploy Your Code with Confidence. [WWW] https://travis-ci.org/. (21.04.2016).

[34] Vagrant. [WWW] https://www.vagrantup.com/. (06.03.2016).

[35] Vagrant - Why Vagrant. [WWW] https://www.vagrantup.com/docs/why-vagrant/. (06.03.2016).

[36] Vallaste - e-Teatmik: IT ja sidetehnika seletav sõnaraamat. [WWW] http://vallaste.ee/. (06.03.2016).

[37] Xataface | The fastest way to build a front-end for your MySQL Database. [WWW] http://xataface.com/. (08.03.2016).

[38] shannah/xataface: Framework for building data-driven web applications in PHP and MySQL. [WWW] https://github.com/shannah/xataface. (08.03.2016).

Lisa 1 - PostgreSQL andmabaasisüsteemi süsteemikataloogid

11.1 information_schema

schemata — Sisaldab kõiki skeeme, millele kasutajal on ligipääs.

views — Sisaldab kõiki vaateid, mis asuvad antud andmebaasis. Näidatakse ainult selliseid vaateid, millele kasutajal on ligipääs. Paraku ei saa sealt aga infot materialiseeritud vaadete kohta.

columns — Sisaldab infot andmebaasis olevate tabelite ja vaadete veergude kohta. Näidatakse ainult neid veerge, millele kasutajal on ligipääs. Kui tagastatav tüüp on massiiv, siis saab selle kohta infot information_schema.element_types vaatest. Kui tagastatav tüüp on USER-DEFINED, siis saab selle kohta infot udt_name veerust. Kui veerg on loodud domeeni põhjal, siis saab domeeni nime domain_name veerust

routines — Sisaldab infot andmebaasis olevate funktsioonide kohta, millele kasutajal on ligipääs. data_type veerg sisaldab infot tagastatava tüübi kohta. Kui tagastatav tüüp on massiiv, siis saab selle kohta infot information_schema.element_types vaatest. Kui tagastatav tüüp on USER-DEFINED, siis saab selle kohta infot type_udt_name veerust.

parameters — Sisaldab infot andmabaasis olevate funktsioonide parameetrite kohta. Parameetreid näidatakse ainult nende funktsioone kohta, millele kasutajal on ligipääs.

element_types — Sisaldab infot massiivi tüüpide kohta.

[27]

11.2 pg_catalog

pg_database — Säilitab infot olemas olevate andmebaaside kohta. Erinevalt enamikest süsteemi kataloogidest on pg_database jagatud kõikide klastrisse kuuluvate andmebaaside vahel.

pg_namespace Säilitab infot nimeruumide kohta. Sealt on võimalik kätte saada andmebaasis olevad skeemid.

pg_shadow Sisaldab infot kasutajate kohta, kellel on sisselogimisõigus. See tabel sisaldab paroole kujul 'md5' || md5(parool||kasutajanimi).

pg_class Sisaldab infot kõige kohta, millel on veerud, või on mõnes muus mõttes tabeliga sarnane. Sealt saab infot vaadete ja materialiseeritud vaadete kohta. Selle tabeli pealt on tehtud ka vaates pg_views ja pg_matviews, millest on samuti võimalik küsida infot vastavalt vaadete ja materialiseeritud vaadete kohta. Lisaks ei pea kasutajatel olema reaalne ligipääs antud objektidele, et näha infot nende objektide kohta.

pg_attribute Sisaldab infot veergude kohta.

pg_type Sisaldab infot andmetüüpide kohta. Siin tabelis on esindatud nii põhiandmetüübid, kasutaja loodud tüübid, domeenid ja komposiitandmetüübid, mis luuakse iga andmebaasis oleva tabeli jaoks.

pg_proc Sisaldab infot funktsioonide kohta.

[29]

Lisa 2 - Free Software

Free Software (Vaba tarkvara) tähendab, et kasutajatel on vabadus tarkvara jooksutada, kopeerida, levitada, uurida, muuta ja täiustada. Seega *Free Software* rõhub kasutaja vabadusele, mitte tarkvara hinnale.

Tarkvara on *Free Software*, kui selle kasutajate jaoks on täidetud neli olulist kriteeriumit:

- Vabadus 0: jooksutada programmi oma suva järgi, ükskõik mis eesmärgil
- Vabadus 1: uurida, kuidas programm töötab ja seda muuta (eeldab ligipääsu lähtekoodile)
- Vabadus 2: levitada antud tarkvara
- Vabadus 3: levitada antud tarkvara muudetud kujul (eeldab ligipääsu lähtekoodile)

Vabadus levitada (vabadused 2 ja 3) tähendab vabadust jagada antud tarkvara muudetud või muutmata kujul kas tasu eest või tasuta - selleks ei pea kelleltki luba küsima. Küll aga peab jagatav koopia sisaldama nii lähtekoodi kui ka käivitatavat programmi (kui programmeerimiskeel toetab seda võimalust)

Free Software ei tähenda, et tegu ei võiks olla kommertstarkvaraga. *Free Software* võib omandada tasuta või raha eest. Vaatamata sellele, kuidas koopia antud tarkvarast omandati, jääb omandajale vabadus antud tarkvara jagada, muuta ja müüa. [10]

Lisa 3 - Open Source

Open Source (Avatud lähtekood) ei tähanda ainult ligipääsu lähtekoodile. Tarkvara levitamisel peab lähtuma järgmistest reeglitest:

1. Vaba jagamine - Litsents ei tohi piirata ühtegi osapoolt tarkvara müümast või jagamast.
2. Lähtekood - Tarkvara peab sisaldama lähtekoodi ning lähtekoodi ja kompileeritud koodi jagamine peab olema lubatud. Kui tarkvara ei jagata koos lähtekoodiga, peab lähtekood olema mujalt mõistliku vaevaga kättesaadav.
3. Tuletatud tarkvara - Litsents peab lubama muudatusi ja tuletatud tarkvara ning peab lubama nende jagamist samadel litsentsitingimustel.
4. Autori lähtekoodi terviklikkus - Litsents võib keelata muudetud lähtekoodi jagamist üksnes siis, kui on lubatud jagada paikefaile (*patch file*), et muuta programmi lähtekoodi selle loomise mingis järgus (*build time*). Litsents peab selgelt lubama muudetud lähtekoodiga tarkvara jagamist. Litsents võib nõuda, et tuletatud tarkvara kannaksid teist nime või versiooninumbrit, kui originaaltarkvara.
5. Isikute või gruppide diskrimineerimiskeeld - Litsents ei tohi diskrimineerida ühtegi isikut või isikute gruppi.
6. Tegevusvaldkonna diskrimineerimiskeeld - Litsents ei tohi piirata ühtegi konktreetset tegevusvaldkonda.
7. Litsentsi jagamine - Programmile sätestatud õigused kehtivad kõigile, kellele programm on jagatud, ilma, et osapooled vajaksid täiendavat litsentsi.
8. Litsents ei tohi olla tootespetsiifiline - Programile sätestatud õigused ei tohi sõltuda sellest, kas programm kuulub mõne teise programmi koosseisu.
9. Litsents ei tohi piirata teisi tarkvarasid - Litsents ei tohi panna piiranguid teistele tarkvaradele, mida jagatakse koos antud tarkvaraga.
10. Litsents peab olema tehnoloogiliselt neutraalne - Ükski klausel ei tohi viidata konkreetsele tehnoloogiale, stiilile või liidesele.

[19]