

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Informaatika instituut

IDU70LT

Rait Raidma 143682

POSTGRESQL ANDMEBAASISÜSTEEMI
PÕHINE METAANDMETEGA JUHITAVATE
VEEBIRAKENDUSTE
KIIRPROGRAMMEERIMISE KESKKOND

Magistritöö

Juhendaja: Erki Eessaar
Doktor
dotsent

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.

Autor: Rait Raidma

09.05.2016

Annotatsioon

Käesoleva magistritöö eesmärgiks oli disainida ning realiseerida PostgreSQL andmebaasisüsteemi põhine ja veebipõhine kiirprogrammeerimise keskkond, mis võimaldaks luua veebirakendusi. Loodavad rakendused peavad suhtlema andmebaasiga läbi andmebaasiliidese ning rakenduste väljanägemist ning käitumist peab olema võimalik juhtida andmebaasis hoitavate metaandmetega.

Töö käigus uuriti, kuidas saab PostgreSQL andmebaasisüsteem suhelda mitme samas serveris asuva PostgreSQL andmebaasiga ning kust saab infot vajalike andmebaasiobjektide kohta. Seejärel kirjeldati loodava süsteemi funktsionaalsus, disainiti selle andmebaas, kasutajaliides ning rakendus.

Töö tulemusena valmis süsteemi kirjeldav dokumentatsioon ning sellele vastav süsteem, mille kasutatavuse valideerimiseks realiseeriti ühe rakenduse näidis-kasutusjuhud. Süsteem võimaldab hallata rakendusi, lehti ning neis olevaid regioone, mida antud töö käigus realiseeriti neli erinevat tüüpi: HTML-tekst, navigatsioon, raport ja vorm.

Näidisrakendusele pääseb ligi kasutajanimega *kask@ttu.ee* ja parooliga *test* aadressilt <http://apex.ttu.ee/pgapex/public/index.php/app/ruumid>.

Töö tulemus avaldati MIT litsentsiga ning on avalikult kättesaadav aadressilt <https://github.com/raitraidma/pgapex>.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 72 leheküljel, 10 peatükki, 38 joonist, 14 tabelit.

Abstract

A PostgreSQL-based Rapid Development Environment for the Metadata-driven Web Applications

The goal of this master thesis is to design and implement a PostgreSQL-based and web-based rapid development environment for the metadata-driven web applications that store data in the same database server. These web applications must communicate with a database through database interface. Applications' look and behaviour must be possible to manipulate by using metadata that is kept in the database.

The work investigates how it is possible to connect with multiple PostgreSQL databases in PostgreSQL database management system and where to look for the information about necessary database objects. After that the system functionality, database, user interface, and application design are described.

The outcomes of this thesis are documentation that describes the system and implementation of the system. For the creation of the system PHP, PostgreSQL, AngularJS, Bootstrap, and Slim Framework were used. The system uses PostgreSQL's *dblink* extension to communicate to external databases. Information about database objects are asked from *information_schema* and *pg_catalog*.

The system allows developers to create applications based on external databases and for each application select the authentication method from the set of two methods (no authentication and authentication based on the database function). Application may contain pages that consist of regions. Four region types were implemented in the system. HTML region allows us to add HTML-formatted text. Navigation region displays menus that can be used to navigate between application's pages or refer to external ones. Report region displays a table based on a view which rows can be divided between pages. Form region allows us to ask information from a user and send it to external database by calling out a function from external database.

Created system is the result of the first development iteration. Only the most important functionality was implemented to ensure the system's primary usability to create real applications. To validate the results, a sample application was made to show what the system can do.

Sample application can be accessed with username *kask@ttu.ee* and password *test* from <http://apex.ttu.ee/pgapex/public/index.php/app/ruumid>.

The result of current thesis is available under MIT licecne from
<https://github.com/raitraidma/pgapex>.

The thesis is in Estonian and contains 72 pages of text, 10 chapters, 38 figures, 14 tables.

Lühendite ja mõistete sõnastik

ACID	<i>Atomicity, Consistency, Isolation, Durability</i> , atomaarsus, konsistentsus, isoleeritus, püsivus. “Andmebaasihaldurite puhul tehingutöötluse põhikarakteristikud. Atomaarsus tähendab seda, et andmebaasihaldur tagab kas antud tehingu kõigi tegumite või mitte ühegi toimumise. Konsistentsus tähendab, et pärast tehingu lõppu peab andmebaas olema legaalses olekus. Isoleeritus tähendab, et tehing toimub varjatult teiste operatsioonide eest. Püsivus tähendab seda, et kui kasutajat on juba kord teavitatud tehingu edukast toimumisest, siis tehing jääb kehtima ja seda ei saa olematuks muuta” [46]
AJAX	<i>Asynchronous JavaScript And XML</i> , asünkroonne JavaScript ja XML. “Interaktiivsete veebirakenduste loomise meetod, kus andmevahetus brauseri ja veebiserveri vahel toimub ilma, et oleks vaja kogu lehte uuesti laadida” [46]
CRUD	<i>Create Read Update Delete</i> , lühend, mis tähistab andmetega manipuleerimise nelja põhitegevust: loomine, lugemine, muutmine ja kustutamine
CSRF	<i>Cross-Site Request Forgery</i> , rünnak, mille puhul käivitatakse veebirakenduses mingi toiming teisest domeenist tulnud päringu korral
CSS	<i>Cascading Style Sheets</i> , kaskaadlaadistik. “Veebilehtede valmistajatele ja kasutajatele mõeldud laadistik. Laadilehed (style sheets) kirjeldavad, kuidas HTML dokumente esitada kuvaril, printeril või kõnesüntesaatorist kostva kõnena” [46]
DOM	<i>Document Object Model</i> , dokumendiobjektide mudel. “Eeskiiri 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” [46]
FSF	<i>Free Software Foundation</i> , MTÜ, mis propageerib arvuti kasutajate vabadust ja kaitseb vaba tarkvara kasutajate õigusi

HTML	<i>HyperText Markup Language</i> , hüpertekst-märgistuskeel. “Enim-levinud kodeerimissüsteem (tekstivorming) veebidokumendi loomiseks. HTML koodid ehk märgendid määravad ära selle, kuidas veebileht arvutiekraanil välja näeb” [46]
Juurutama	<i>Deploy</i> , tarkvara või riistvara töölepanekuga seotud protsesside - installeerimine, konfigureerimine, käitamine, testimine - läbimine [46]
Kiirprogrammeerimine	<i>Rapid Application Development</i> , “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” [46]
Metaandmed	“Andmed andmeelementide kohta, sealhulgas nende andme- kirjeldused, ning andmed andmete omanduse, pöördusteede, pääsuõiguste ja muutuvuse kohta” [18]
Metaandmetega juhitud	Süsteemi käitumist ja väljanägemist juhatakse andmetega
OSI	<i>Open Source Initiative</i> , organisatsioon, mis propageerib avatud lähtekoodiga tarkvara
SQL süstimine	<i>SQL Injection</i> , rünnak, mille puhul kasutaja muudab käivitavat SQL lauset
SQL	<i>Structured Query Language</i> , struktureeritud andmebaasikeel andmete käitlemiseks, õiguste jagamiseks ning andmebaasi- objektide haldamiseks
URL	<i>Uniform Resource Locator</i> , internatiaadress. Viit arvutivõrgus olevale ressursile. [46]

Sisukord

1	Sissejuhatus	14
1.1	Taust ja probleem	14
1.2	Ülesande püstitus	15
1.3	Metoodika	15
1.4	Ülevaade tööst	16
2	Teoreetiline taust	17
2.1	Andmebaasi avalik liides	17
2.1.1	Vaadete kasutamise eelised ja võimalused	18
2.1.2	Vaadete kasutamisel tekkida võivad probleemid PostgreSQL and- mebaasisüsteemi näitel	19
2.1.3	Andmebaasiserveris talletatud rutiinide kasutamise eelised	19
2.1.4	Andmebaasiserveris talletatud rutiinide puudused	20
2.1.5	Järeldus andmebaasiliideste kasutamise kohta	20
2.2	Ühendumine teiste andmebaasidega	20
2.2.1	dblink	21
2.2.2	postgres_fdw	22
2.2.3	Mooduli valik	22
2.3	Andmebaasiobjektide kirjelduste küsimine	22
2.4	Eksisteerivate programmide analüüs	23
2.4.1	Oracle Application Express (APEX)	23
2.4.2	NuBuilder	24
2.4.3	Xataface	24
2.4.4	Andmetega juhitud arendussüsteem PostgreSQL andmebaasiraken- duste genereerimiseks	25
2.5	Täpsustunud ülesande püstitus	25
2.6	Litsents	26
3	Süsteemi analüüs	27
3.1	Tegutsejad	27
3.2	Terviksüsteemi tükeldus allsüsteemideks	27
3.2.1	Pädevusalad	27
3.2.2	Funktsionaalsed allsüsteemid	27
3.2.3	Registrid	28
3.3	Rakenduste funktsionaalne allsüsteem	28
3.3.1	Eesmärgid	28
3.3.2	Allsüsteemi poolt kasutatavad registrid	29
3.3.3	Allsüsteemi kasutusjuhtude eskiismudel	29

3.4	Lehtede funktsionaalne allsüsteem	31
3.4.1	Eesmärgid	31
3.4.2	Allsüsteemi poolt kasutatavad registrid	31
3.4.3	Allsüsteemi kasutusjuhtude eskiismudel	32
3.5	Regioonide funktsionaalne allsüsteem	33
3.5.1	Eesmärgid	33
3.5.2	Allsüsteemi poolt kasutatavad registrid	33
3.5.3	Allsüsteemi kasutusjuhtude eskiismudel	33
3.6	Navigatsioonide funktsionaalne allsüsteem	35
3.6.1	Eesmärgid	35
3.6.2	Allsüsteemi poolt kasutatavad registrid	36
3.6.3	Allsüsteemi kasutusjuhtude eskiismudel	36
3.7	Mittefunktsionaalsed nõuded	38
4	Andmebaas	39
4.1	Rakenduse genereerimine	39
4.2	Andmebaasikirjeldus	40
4.2.1	Andmebaasiobjektide register	40
4.2.2	Rakenduste register	41
4.2.3	Lehtede register	43
4.2.4	Regioonide register	44
4.2.5	Navigatsioonide register	48
4.2.6	Mallide register	49
5	Kasutatavad tehnoloogiad ja arendusprotsess	54
5.1	Vagrant	54
5.2	Bower	54
5.3	AngularJS	54
5.4	Bootstrap	55
5.5	TravisCI	55
5.6	PHP	55
5.7	Composer	55
5.8	Slim Framework 3	56
5.9	Postgresql	56
5.10	Arendusprotsess	56
6	Kasutajaliides	58
7	Rakenduse disain	59
7.1	Rakenduse ülesehitus	59

7.1.1	Kontrollerid	59
7.1.2	Vahevara	60
7.1.3	Mudelid	60
7.1.4	Teenused	61
7.1.5	Marsruuter	61
8	Näidisrakendus	62
8.1	Kasutaja tuvastamine	62
8.2	Ruumi lõplikult mitteaktiivseks muutmine	63
8.3	Ruumide koondaruande ja kõikide ruumide vaatamine	63
9	Arendusvaade	65
10	Kokkuvõte	67
	Kasutatud kirjandus	69
	Lisa 1 - PostgreSQL andmabaasisüsteemi süsteemikataloogid	73
	Lisa 2 - Free Software	75
	Lisa 3 - Open Source	76
	Lisa 4 - Populaarsemate litsentside võrdlus	77
	Lisa 5 - Kasutusjuhtude diagrammid	78
	Lisa 6 - Andmebaasi diagrammid	81
	Lisa 7 - Kasutajaliidese ekraanipildid	85
	Lisa 8 - Metaandmete küsimine PostgreSQL andmebaasisüsteemis	88

Jooniste loetelu

1	Andmebaasi avalik liides.	17
2	SQL: Andmete küsimine välisest andmebaasist (hüpoteetiline süntaks). . .	21
3	SQL viga andmete küsimisel välisest andmebaasist.	21
4	SQL: dblink mooduli installeerimine.	21
5	SQL: SQL lause käivitamine välises andmebaasis dblink mooduli abil. . .	21
6	SQL: funktsiooni väljakutse.	22
7	Süsteemi tööpõhimõte.	26
8	Andmebaasiobjektide registri olemi-suhte diagramm.	40
9	Rakenduste registri olemi-suhte diagramm.	42
10	Lehtede registri olemi-suhte diagramm.	43
11	Regioonide registri olemi-suhte diagramm osa 1.	44
12	Regioonide registri olemi-suhte diagramm osa 2.	45
13	Regioonide registri olemi-suhte diagramm osa 3.	45
14	Regioonide registri olemi-suhte diagramm osa 4.	46
15	Navigatsioonide registri olemi-suhte diagramm.	48
16	Mallide registri olemi-suhte diagramm.	50
17	PHP: Andmebaasisga ühendamine.	60
18	PHP: Kasutaja õiguste kontroll.	61
19	Näidisrakenduse kasutusjuhtude diagramm.	62
20	Näidisrakendus: Kasutaja tuvastamine.	63
21	Näidisrakendus: Ruumi lõplikult mitteaktiivseks muutmine.	63
22	Näidisrakendus: Ruumide koondaruande ja kõikide ruumide vaatamine. .	64
23	Rakenduste funktsionaalse allsüsteemi kasutusjuhtude diagramm.	78
24	Lehtede funktsionaalse allsüsteemi kasutusjuhtude diagramm.	79
25	Regioonide funktsionaalse allsüsteemi kasutusjuhtude diagramm.	79
26	Navigatsioonide funktsionaalse allsüsteemi kasutusjuhtude diagramm. . .	80
27	Andmebaasiobjektide registri andmebaasi diagramm.	81
28	Rakenduste registri andmebaasi diagramm.	82
29	Lehtede registri andmebaasi diagramm.	82
30	Regioonide registri andmebaasi diagramm.	83
31	Navigatsioonide registri andmebaasi diagramm.	83
32	Mallide registri andmebaasi diagramm.	84
33	Loodud süsteem: Autentimine.	85
34	Loodud süsteem: Rakenduse loomine.	85
35	Loodud süsteem: Raporti regiooni muutmine.	86
36	Loodud süsteem: Vormi regiooni muutmine.	87

37	SQL: Andmebaasis vastavas skeemis olevate vaadete ja materialiseeritud vaadete küsimine.	88
38	SQL: Andmebaasis vastavas skeemis oleva funktsiooni parameetrite info küsimine.	88

Tabelite loetelu

1	Mittefunktsionaalsed nõuded.	38
2	Andmebaasiobjektide registri olemitüüpide definitsioonid.	41
3	Andmebaasiobjektide registri olemitüüpide atribuutide definitsioonid. . .	41
4	Rakenduste registri olemitüüpide definitsioonid.	42
5	Rakenduste registri olemitüüpide atribuutide definitsioonid.	42
6	Lehtede registri olemitüüpide definitsioonid.	43
7	Lehtede registri olemitüüpide atribuutide definitsioonid.	43
8	Regioonide registri olemitüüpide definitsioonid.	46
9	Regioonide registri olemitüüpide atribuutide definitsioonid.	47
10	Navigatsioonide registri olemitüüpide definitsioonid.	49
11	Navigatsioonide registri olemitüüpide atribuutide definitsioonid.	49
12	Mallide registri olemitüüpide definitsioonid.	51
13	Mallide registri olemitüüpide atribuutide definitsioonid.	51
14	Populaarsemate litsentside võrdlus.	77

1 Sissejuhatus

Tänapäeval on arvutisüsteemi toega infosüsteemid väga laialdaselt kasutusel. Tihti aga puudub võimekus kiirelt reageerida muutuvatele info haldamise ning kuvamise vajadustele, kuna pole piisavalt vastavate oskustega inimesi. Seetõttu oleks palju kasu süsteemist, mis ei nõua kasutajalt programmeerimisoskust mitmes erinevas keeles, vaid võimaldaks infosüsteemi info haldamiseks mõeldud veebirakendusi kiiresti ja mugavalt graafilise kasutajaliidese abil täiendada.

1.1 Taust ja probleem

Töö idee sai alguse TTÜ-s õpetatavast ainek "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 [32] ja Oracle [26]. Juhul, kui andmebaas on loodud Oracle andmebaasisüsteemi abil, siis on üliõpilastel rakenduse loomiseks võimalus kasutada metaandmetega juhivat veebipõhist kiirprogrammeerimise keskkonda Oracle APEX [25]. PostgreSQL andmebaasisüsteemiga loodud andmebaasi korral tuleb rakendus programmeerida kasutades näiteks PHP-d [30]. 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 veebipõhiseid andmebaasirakendusi, mis kasutavad andmebaasisüsteemina PostgreSQL-i. Samas leiab autor, et valmiv süsteem on piisavalt võimekas, et leida rakendust mujalgi.

PostgreSQL on avatud lähtekoodiga ja tasuta pakutav andmebaasisüsteem. 2016. aasta maikuu seisuga on see andmebaasisüsteemide populaarsuse indeksis [9] viiendal kohal. On hea võimalus, et sellel põhineval arendussüsteemil võiks olla lai kasutajaskond. Oracle samalaadse Oracle APEX arenduskeskkonna abil arendajate kogukond ulatub 300000 inimeseni [27] ning seda kasutatakse mitmesuguse äritarkvara loomiseks.

2008. aastal kaitsti magistritöö [19], mille käigus juba uuriti sellise süsteemi PostgreSQL-i põhjal realiseerimise võimalust ning realiseeriti esialgne prototüüp. Selle töö tulemused näitasid, et taolisel põhimõttel loodud süsteemi loomine PostgreSQL-i põhjal on põhimõtteliselt võimalik. Paraku oli loodud prototüüp suhteliselt algeline, selle lähtekood läks kaduma ning mingit edasist arendust sellega ei toimunud. Seega tuleb pidada õigustatuks selle süsteemi idee uuesti elule äratamist ja süsteemi täiesti uuesti loomist.

Autor tutvus 2008. aasta lõputöö dokumendiga, kuid nii käesoleva süsteemi kavandid kui ka realisatsioon on täiesti uus tulemus.

Töö valmis 2016. aasta kevadel Tallinna Tehnikaülikoolis.

1.2 Ülesande püstitus

Töö eesmärgiks on disainida ning realiseerida PostgreSQL andmebaasisüsteemi põhine, veebipõhine ja metaandmetega juhitud kiirprogrammeerimise keskkond, mille abil saaks luua teisi veebipõhiseid rakendusi. Need rakendused kasutaksid arendussüsteemiga samas serveris olevaid PostgreSQL andmebaase. Loodavates rakendustes peab andmete pärimine, lisamine, muutmine ning kustutamine käima läbi avaliku andmebaasiliidese e virtuaalse andmete kihi. 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 vaba tarkvara litsentsi all, et soodustada süsteemi laialdasemat levikut ning edasist arendamist.

1.3 Metoodika

Esiteks tuleb uurida, kas liideste kasutamine andmebaasi poole peal annab süsteemile mingi eelise. Seejärel tuleb selgeks teha, kas ja kuidas saab PostgreSQL andmebaasisüsteem ja selle kaudu tema teenuseid kasutav rakendus suhelda mitme samas serveris asuva PostgreSQL andmebaasiga. Samuti on vaja selgitada, kuidas küsida teistest andmebaasidest infot liideseid moodustavate andmebaasiobjektide kohta. Lisaks uuritakse, milliseid sarnaseid süsteeme on veel olemas ning kuidas need on üles ehitatud. Viimane võimaldab leida nõudeid uuele süsteemile.

Süsteemi kavandamiseks ning töös realiseeritava süsteemi alamosa leidmiseks kasutatakse süsteemi tükeldamist allsüsteemideks. Allsüsteemideks jagamine võimaldab ka süsteemi mudeleid paremini struktureerida ja esitada. Süsteemi kavandamine toimub mitmevaateliselt (andmevaade, funktsionaalne vaade) ja mudelipõhiselt. Kontseptuaalseid andmudeleid kasutati tehnilise lahenduse e tabelite kirjelduste genereerimiseks.

Töö tulemusena valmib veebipõhine süsteem, mille abil saab luua veebirakendusi, mis kasutavad andmebaasiga suhtlemiseks andmebaasiliideseid. Töö tulemuse valideerimiseks loodakse näiterakendus, kus realiseeritakse õppejõu poolt ette antud kasutusjuhud, mis

sarnanevad üliõpilastöodes esinevatele kasutusjuhtudele.

1.4 Ülevaade tööst

Töö alguses antakse ülevaade, mis on andmebaasiliides, kuidas toimub suhtlus väliste andmebaasidega ning kust saab infot andmebaasiobjektide kohta. Lisaks uuritakse milliseid sarnaseid süsteeme on veel olemas. Seejärel tuuakse välja nõudmised, mida süsteem täitma peab ning kirjeldatakse süsteemi ülesehitust. Lõpuks valideeritakse töö tulemust näiterakenduse realiseerimisega ning antakse ülevaade funktsionaalsustest, mida järgnevate iteratsioonide korral teostada võiks.

2.1.1 Vaadete kasutamise eelised ja võimalused

- Võimaldavad igale rakendusele luua spetsiifilise vaate andmetest, ilma et oleks vaja teha muudatusi andmestruktuurides, mille põhjalt vaateid serveeritakse.
- Võimaldavad vähendada rakenduse koodi ja andmemudeli vahelist sidusust. See võimaldab teha muudatusi andmemudelis, ilma, et olemasolev rakendus katki läheks.
- Neis saab kasutada rakenduse-spetsiifilisi veerunimesid, andmetüüpe ja pikkuseid, mis võimaldab andmete paremat sidumist rakenduses kasutatavate mudelitega.
- Võimaldavad ühe täiendava meetme andmete turvalisuse tagamiseks. Kasutajatele või rollidele saab anda õiguse küsida andmeid vaadetest, kuid mitte anda neile õigust lugeda või muuta andmeid baastabelites. Erinevatele kasutajatele 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. [37]
- Võimaldavad pärida andmeid erinevatest tabelitest ja andmebaasidest, peites kasutajate eest päringu tegeliku keerukuse. Vaate koostamiseks vajalik päring on eelnevalt koostatud ja optimeeritud. Kui kasutaja teeb vaate, millel on alampäring q1, põhjal päringu q2, siis andmebaasisüsteem koostab q1 ja q2 põhjal uue päringu. Selle täitmiseks kasutatakse vaate aluseks olnud tabelitele loodud indekseid.
- Võimaldavad varjata rakenduse eest baastabelites olevaid disaini -ja andmevigasid, andes lisaäga nende parandamiseks.
- Võimaldavad kuvada samu andmeid erineval kujul ühendatuna, kasvõi nt XML või JSON dokumentidena.
- Läbi vaadete, mis vastavad teatud tingimustele, on võimalik teha andmemuudatusi baastabelites. Lisaks sellele saavad arendajaid vaateid ise andmemuudatusi võimaldavaks ümber programmeerida, kasutades selleks *INSTEAD OF* trigereid.

[6, lk 172-173]

2.1.2 Vaadete kasutamisel tekkida võivad probleemid PostgreSQL andmebaasisüsteemi näitel

Kõik järgnevad punktid osutavad sellele, et tehes vaate põhjal päringu võib andmebaasisüsteem mõnikord koostada täitmisplaani, mille alusel lause täitmine võtab rohkem aega võrreldes sellega, kui päring oleks tehtud otse baastabelite põhjal.

- 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.
- Kui vaate alampäringus viidatakse teistele vaadetele, siis nende vaadete alampäringud täidetakse eraldi.

[16, lk 101-102]

2.1.3 Andmebaasiserveris talletatud rutiinide kasutamise eelised

- Üle võrgu saadetavate andmete ja SQL koodi hulk hoitakse minimaalsena, mille tulemusel suureneb rakenduse jõudlus.
- Rutiinide kood on andmebaasiserveris eelnevalt koostatud ja optimeeritud, suurendades rutiini täitmise efektiivsust.
- Andmetöötluse jaoks kasutatakse andmebaasiserveri jõudlust, mitte rakendusserveri ega kliendi masina oma.
- Rutiinis olevat SQL koodi on lihtsam testida ja optimeerida, kui rakendusse sisse kirjutatud SQL-i.
- Rutiinide käivitusõiguste abil saab piirata andmetele ligipääsu teatud rollidele ning suurendada seeläbi turvalisust.

- Rutiinis käivitavad laused tehakse ühe transaktsiooni jooksul. See aitab vältida osalisi andmemuudatusi, kus üks osa muudatustest läheb läbi, teine osa aga mitte.

[6, lk 179, 195]

2.1.4 Andmebaasiserveris talletatud rutiinide puudused

- Koodifunktsionaalsus on piiratud. Rutiinides kasutatavad tsüklid pole nii kiired kui rakenduses. Eriti aeglased ning protsessori-nõudlikud on kursorid.
- Keerulisemaid rutiine ei pruugi olla võimalik teisaldada ühelt andmebaasisüsteemilt teisele, sest erinevates andmebaasisüsteemides on rutiinide kirjutamiseks kasutusel erinevad keeled.
- Rutiin on keerulisem grupeerida ning selle tulemusena võib üks ärireegel olla jaotatud mitme rutiini vahele, mis teeb ärireegli haldamise keerulisemaks.

[13]

2.1.5 Järeldus andmebaasiliideste kasutamise kohta

Andmebaasiliidesed annavad kasutajale võimaluse vähendada andmebaasi ja rakenduse vahelist sidusust ning suurendada andmete turvalisust ja terviklikust. Mõningatel juhtudel võivad aga vaated ja rutiinid mõjuda operatsioonide jõudlusele negatiivselt ning rutiinides võib mõningate tegevuste täitmiseks loodava koodi kirjutamine olla keerulisem kui rakenduskihis. Küll aga ei kaalu autori arvates negatiivsed aspektid üle positiivseid, ning seetõttu leiab autor, et andmebaasiliideseid tuleks võimaluse korral siiski kasutada.

2.2 Ühendumine teiste andmebaasidega

Kui loodav süsteem installeerida 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 teistesse andmebaasidesse (lisaks metaandmete andmebaasile, mida süsteem tööks vajab). PostgreSQL andmebaasisüsteemis ei saa kasutada "tavalisi andmemuudatuse lauseid, (vt Joonis 2) mis viitaksid korraga mitmele andmebaasile.

```
SELECT * FROM other_db_name . schema_name . table_name ;
```

Joonis 2. SQL: Andmete küsimine välisest andmebaasist (hüpoteetiline süntaks).

Eelnev päring annab tulemuseks veateate (vt Joonis 3).

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

Joonis 3. SQL viga andmete küsimisel välisest andmebaasist.

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

2.2.1 dblink

Mooduli kasutamiseks tuleb see esmalt installeerida (vt Joonis 4).

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

Joonis 4. SQL: dblink mooduli installeerimine.

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 (vt Joonis 5). [34]

```
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  
) ;
```

Joonis 5. SQL: SQL lause käivitamine välises andmebaasis dblink mooduli abil.

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 välises andmebaasis oleva tabeli vastu. Seda ühendust hoitakse alles ning kasutatakse järgmiste päringute jaoks sama sessiooni piires. Kui väliselt serverilt küsitakse infot erinevate kasutajatenä, siis luuakse iga kasutaja jaoks uus ühendus.

Postgres_fdw üritab optimeerida väliseid päringuid, et vähendada küsitavate andmete hulka. Selleks saadetakse koos päringuga *WHERE*-klausel ning ei laeta alla veerge, mida pole päringu täitmiseks vaja. Selleks, et vältida valesid päringutulemusi, ei saadeta *WHERE*-klauslit, kui kasutatakse midagi peale sisse ehitatud andmetüüpide, operaatorite ja funktsioonide või kui operaatorid ja funktsioonid pole muutumatud (*immutable*). [35]

Postgres_fdw võimaldab lisaks andmete küsimisele (*SELECT*) ka andmeid lisada (*INSERT*), muuta (*UPDATE*) ja kustutada (*DELETE*). Küll aga ei võimalda antud moodul välja kutsuda välises andmebaasis olevaid funktsioone (vt Joonis 6).

```
SELECT function_from_external_database ();
```

Joonis 6. SQL: funktsiooni väljakutse.

2.2.3 Mooduli valik

Kuna loodav süsteem peab suutma välja kutsuda samas serveris asuvates välistes PostgreSQL andmebaasides olevaid funktsioone, siis tuleb kasutada *dblink* (2.2.1) moodulit.

2.3 Andmebaasiobjektide kirjelduste küsimine

Loodav süsteem peab teistest andmebaasidest küsima infot andmebaasiobjektide 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, mis esitavad andmeid andmebaasis olevate andmebaasiobjektide kohta. Kuna *information_schema* on defineeritud SQL standardis, siis võib eeldada, et selle formaat ei muutu ning seetõttu tuleks eelistada seda kata-

loogi, et vältida loodava süsteemi katkiminekut järgmiste PostgreSQL versioonide korral. [36] 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. [38] Täpsem loetelu olulisematest süsteemikataloogide vaadetest, mida süsteemi loomisel vaja läheb, on välja toodud Lisas 1.

2.4 Eksisteerivate programmide analüüs

Töö käigus uuriti, milliseid sarnaseid süsteeme on veel loodud ning kuidas need on üles ehitatud. Järgnevalt on esitatud lühiülevaade 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 ning kasutab tööks Oracle andmebaasisüsteemi.

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 näha lehtede statistikat ning hallata seadeid.
- **SQL Workshop** - Võimaldab näha ja hallata andmebaasiobjekte, käivitada päringuid, importida/exportida andmebaasis olevaid andmeid, koostada päringuid graafilise liidese abil, luua RESTful liideseid jpm.
- **Team Development** - Tööde- ja vigadehaldussüsteem. Võimaldab arendajatel ülesandeid planeerida ja hallata.
- **Packaged Apps** - Galerii näidisrakendustest, mida on võimalik kohe kasutamiseks installeerida.

[25] Käesoleva süsteemi idee on tekkinud just Oracle APEX arenduskeskkonna põhjal. Soov on midagi sarnast realiseerida ka PostgreSQL-i jaoks. Käesolevas töös realiseeritav

funktsionaalsus on alamosa Oracle APEX-i Application Builderi poolt pakutavast funktsionaalsusest.

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 andmebaasis. 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. Rakenduse genereerimine toimub PHP-pole peal. [22] Oma kodulehel väidavad nad, et tegu on avatud lähtekoodiga tarkvaraga ning lähtekood on avalikult üleväl [23], kuid kusagil pole mainitud, millise 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 seetõttu on raskendatud nendest arusaamine.

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.[48]

Xataface on avatud lähtekoodiga ning antud välja GPL litsentsi all. Programm on kirjutatud PHP-s [30] ning andmebaasisüsteemina kasutatakse MySQL-i [21].

Kasutatud on palju väliseid teke. Programmil on üks põhiline arendaja ning igapäevast arendustööd ei toimu. [49]

2.4.4 Andmetega juhitud arendussüsteem PostgreSQL andmebaasirakenduste genereerimiseks

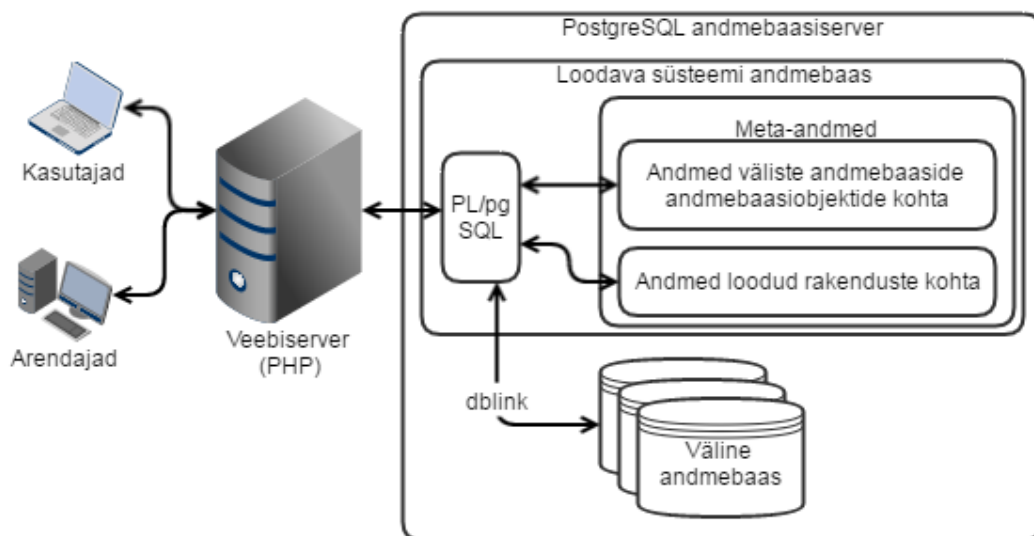
Tegu on Mati Metsise magistritöö [19] tulemusega. Paraku läks selle lähtekood kaduma ning edasist arendust sellega ei toimunud. Küll aga on olemas dokument, mille põhjal on võimalik saada ülevaade, mida tema poolt loodud prototüüp võimaldas teha.

Prototüübis realiseeriti alamosa dokumendis kirjeldatud funktsionaalsusest. Loodud süsteem võimaldas luua rakendusi ja lisada neisse lehti, mis omakorda sisaldasid regioone. Raporti tüüpi regioon genereeriti SQL päringu põhjal. Lisaks oli võimalik luua vorme. Valikkasti ja raadionupu väärtuste määramine käis SQL-päringu abil. Iga rakenduse jaoks loodi andmebaasi uus skeem. Andmete muutmise võimalust antud töös ei realiseeritud.

Dokumendi põhjal jäi mulje, et prototüüp ei genereerinud loodud rakendust andmebaasi poolele, vaid seda tehti PHP-abil rakenduskihis.

2.5 Täpsustunud ülesande püstitus

Kasutades PostgreSQL 9.4 andmebaasisüsteemi [32] ja PHP 5.5 skriptimiskeelt [30], tuleb realiseerida veebipõhine arendussüsteem, mille abil saab luua teisi veebirakendusi. Tarkvara versioonid, mille põhjal süsteemi arendati, valiti lähtuvalt süsteemi esimese kasutaja käsutuses olevast keskkonnast. Süsteem on vaja luua nii, et see töötaks ka hilisemate versioonide põhjal. Loodavad rakendused peavad andmebaasiga suhtlemiseks kasutama vaadete, materialiseeritud vaadete ja funktsioonide abil loodud andmebaasi avalikku liidest (2.1). Loodav süsteem peab infot liidest moodustavate andmebaasiobjektide kohta küsima samas serveris asuvate väliste PostgreSQL andmebaaside süsteemikataloogidest (2.3) kasutades *dblink* moodulit (2.2.1) ning hoidma seda enda andmebaasis. Kasutamaks võimalikult palju ära andmebaasi võimekust, tuleb rakendused genereerida valmis andmebaasi poole peal. Süsteemi tööpõhimõtet kirjeldab Joonis 7. Valminud süsteemi lähtekood tuleb avaldada.



Joonis 7. Süsteemi tööpõhimõte.

2.6 Litsents

Üheks töö eesmärgiks oli avaldada loodava prototüübi lähtekood vaba tarkvara litsentsi all. Olemasolevaid litsentse on väga palju. Sobiva litsentsi valimiseks, leian esiteks populaarseimad litsentsid ning võrdlen neid omavahel. GitHub-i poolt avaldatud statistika kohaselt on 2016. aasta veebruari 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. Lisas 4 tabelis 14 on välja toodud nende litsentside võrdlus. Loodav süsteem avaldatakse MIT litsentsi all, kuna see seab kasutajatele kõige vähem piiranguid ning arendajale kõige vähem kohustusi.

3 Süsteemi analüüs

Järgnevalt leitakse süsteemi tegutsejad ja allsüsteemid ning kirjeldatakse neis realiseeritavad kasutusjuhud.

3.1 Tegutsejad

- Arendaja.
- Kasutaja.

Arendaja on laiendatud õigustega kasutaja, kellele on lubatud hallata süsteemis loodud rakendusi. Kasutaja on arendaja poolt loodud rakenduste lõppkasutaja.

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

- 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. Sellele ei vasta eraldi funktsionaalset allsüsteemi. Sellesse registrisse laaditakse andmeid väliste andmebaaside süsteemikataloogidest. See funktsionaalsus kuulub rakenduste funktsionaalse allsüsteemi alla.
- Rakenduste register.
- Lehtede register.
- Regioonide register.
- Navigatsioonide register.
- Mallide register.

3.3 Rakenduste funktsionaalne allsüsteem

Järgnevalt on esitatud rakenduste funktsionaalse allsüsteemi eesmärgid, kasutatavad registrid ja kasutusjuhtude eskiismudel.

3.3.1 Eesmärgid

- Võimaldada arendajal saada ülevaade loodud rakendustest.
- Võimaldada värskendada infot andmebaasiobjektide kohta, mille põhjal rakendust luuakse.
- Võimaldada arendajal luua uus rakendus.
- Võimaldada arendajal muuta olemasolevate rakenduste seadeid.

- Võimaldada arendajal kustutada olemasolevaid rakendusi.
- Võimaldada arendajal muuta rakenduse autentimismeetodit.
- Võimaldada arendajal ja kasutajal kasutada loodud rakendust.

3.3.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem teenindab rakenduste registrit ja andmebaasiobjektide registrit.

Allsüsteem kasutab lehtede registrit, regioonide registrit, navigatsioonide registrit ja mal-
lide registrit.

3.3.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud rakenduste funktsionaalse allsüsteemi kasutusjuhtude tekstikirjel-
dused kõrgtaseme formaadis. Kasutusjuhtude mudel on eeskätt tekstiline mudel. Kasutus-
juhtude diagramm on sisuliselt selle sisukorraks. Tulenevalt sellest ja tööle seatud mahu-
piirangutest pidasin paremaks esitada kasutusjuhtude diagrammid siin ja edaspidi lisades.
Selle allsüsteemi kasutusjuhtude diagramm on Lisas 5 Joonisel 23.

Kasutusjuht: Kasutaja identifitseerimine

Tegutsejad: Arendaja

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

Märkus: Kasutusjuht “Kasutaja identifitseerimine” on kasutusel ka järgnevates allsüs-
teemides: lehtede funktsionaalne allsüsteem, regioonide funktsionaalne allsüsteem, na-
vigatsioonide funktsionaalne allsüsteem.

Kasutusjuht: Andmebaasiobjektide informatsiooni värskendamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab uuendada infot välistes andmebaasides hoitavate andmebaasi-
objektide kohta. Arendajale kuvatakse nupp, millele vajutades uuendatakse infot andme-
baasiobjektide kohta ainult nendest andmebaasidest mida loodud rakendused kasutavad.

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. Arendajad näevad kõiki serveris selle vahendi abil loodud rakendusi ja saavad neid kõiki hallata.

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. Arendajad saavad rakenduse luua mistahes samas serveris loodud PostgreSQL andmebaasi põhjal, sh ka metaandmete andmebaasi enese põhjal. Kui arendaja soovib andmebaasi põhjal luua uue rakenduse, siis võiks ta andmebaasis luua piiratud õigustega kasutaja, millel on need, ainult need ja täpselt need õigused, mida rakendus vajab andmebaasiga suhtlemiseks. Selle kasutaja kasutajanime ja parooli tuleks kasutada uue rakenduse loomisel.

Kasutusjuht: Rakenduse muutmine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib rakenduse, mida ta soovib muuta. Talle 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 temalt 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. Talle kuvatakse hetkel kasutusel olev autentimismeetod koos autentimisfunktsiooniga ning siselogmislehe malliga. Arendaja saab eelpool nimetatud andmeid muuta. Kui sisestatud andmed on korrektsed, siis muudatused salvestatakse. Vigade korral kuvatakse vastavad veateated. Süsteem peaks välja pakkuma vähemalt kaks autentimismeetodit - autentimine puudub ning autentimine kasutades serveris loodud funktsiooni, millele tuleb ette anda kasutajanimi ja parool ning mis tagastab tõeväärtuse.

Kasutusjuht: Rakenduse kasutamine

Tegutsejad: Arendaja, Kasutaja

Kirjeldus: Kasutaja saab kasutada loodud rakendust. Talle kuvatakse aktiivse lehe nähtavate regioonide sisu. Lehel võidakse kuvada navigatsioone, raporteid, vorme ja HTML-teksti. Lehtede vahel saab liikuda klikates navigatsiooniregiooni poolt kuvatavatele linkidele. Kui leht pole valitud, siis suutatakse kasutaja avalehele. Kui rakenduses kuvatav leht nõuab, et kasutaja oleks autenditud, siis kuvatakse kasutajale autentimisvorm, kus küsitakse kasutajanime ja parooli, mille korrektse sisestamise korral lubatakse kasutajal näha kaitstud lehti. Sessiooni jooksul peab autentima ainult ühe korra.

3.4 Lehtede funktsionaalne allsüsteem

Järgnevalt on esitatud lehtede funktsionaalse allsüsteemi eesmärgid, kasutatavad registrid ja kasutusjuhtude eskiismudel.

3.4.1 Eesmärgid

- Võimaldada arendajal saada ülevaade rakendusele kuuluvatest lehtedest.
- Võimaldada arendajal luua uusi lehti.
- Võimaldada arendajal muuta olemasolevate lehtede seadeid.
- Võimaldada arendajal kustutada olemasolevaid lehti.

3.4.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem teenindab lehtede registrit.

Allsüsteem kasutab mallide registrit.

3.4.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud lehtede funktsionaalse allsüsteemi kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis. Ruumipuuduse tõttu on kasutusjuhtude eskiismudel esitatud Lisas 5 Joonisel 24.

Kasutusjuht: Lehtede loetelu vaatamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab saada ülevaadet, mis lehed on rakenduse alla loodud. Süsteem kuvab talle 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. Ta 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. Talle kuvatakse lehe pealkiri, alias, mall ja info selle kohta kas tegu on avalehega ning kas leht nõuab kasutaja autentimist. Kuvatud andmeid saab muuta. Kui andmed on korrektsed, 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 temalt kinnitust. Kui arendaja kinnitab kustutamise, siis leht ning sellega seotud info kustutatakse.

3.5 Regioonide funktsionaalne allsüsteem

Järgnevalt on esitatud regioonide funktsionaalse allsüsteemi eesmärgid, kasutatavad registrid ja kasutusjuhtude eskiismudel.

3.5.1 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.

3.5.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem teenindab regioonide registrit.

Allsüsteem kasutab mallide registrit, navigatsioonide registrit, andmebaasiobjektide registrit.

3.5.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud regioonide funktsionaalse allsüsteemi kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis. Ruumipuuduse tõttu on kasutusjuhtude eskiismudel esitatud Lisas 5 Joonisel 25.

Kasutusjuht: Regioonide loetelu vaatamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja tahab saada ülevaadet, mis regioonid on valitud lehe alla loodud. Talle 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 temalt 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 lingi tekst ning võib lisada lisaatribuute 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. Selleks peab ta sisestama vormi välja nime, kirjelduse, järjekorranumbri, valima välja tüübi ja malli. Lisaks saab ta valida, kas väli on kogustulik või mitte, nähtav või peidetud, sisestada välja vaikimisi väärtuse ja kasutajat abistava teksti. Juhul kui välja tüübiks on element, mille abil on võimalik kuvada mitut valikuvõimalust, siis peab arendaja valima vaate ja veerud, mille põhjal valikud luuakse. 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.

3.6 Navigatsioonide funktsionaalne allsüsteem

Järgnevalt on esitatud navigatsioonide funktsionaalse allsüsteemi eesmärgid, kasutatavad registrid ja kasutusjuhtude eskiismudel.

3.6.1 Eesmärgid

- Võimaldada arendajal saada ülevaade rakendusele kuuluvatest navigatsioonidest.
- Võimaldada arendajal luua uusi navigatsioone.
- Võimaldada arendajal muuta olemasolevaid navigatsioone.

- Võimaldada arendajal kustutada olemasolevaid navigatsioonide.
- Võimaldada arendajal lisada olemasoleva navigatsiooni alla navigatsioonipunkte.
- Võimaldada arendajal muuta olemasolevaid navigatsioonipunkte.
- Võimaldada arendajal kustutada olemasolevaid navigatsioonipunkte.

3.6.2 Allsüsteemi poolt kasutatavad registrid

Allsüsteem teenindab navigatsioonide registrit.

Allsüsteem kasutab lehtede registrit.

3.6.3 Allsüsteemi kasutusjuhtude eskiismudel

Järgnevalt on esitatud navigatsioonide funktsionaalse allsüsteemi kasutusjuhtude tekstikirjeldused kõrgtaseme formaadis. Ruumipuuduse tõttu on kasutusjuhtude eskiismudel esitatud Lisas 5 Joonisel 26.

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 nimi on korrektne, 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, mida ta saab muuta. Kui nimi on korrektne, siis see salvestatakse. Vastasel korral kuvatakse vastavad veateated.

Kasutusjuht: Navigatsiooni kustutamine

Tegutsejad: Arendaja

Kirjeldus: Arendaja valib navigatsioonide loetelust navigatsiooni, mida ta soovib kustutada. Enne kustutamist küsitakse temalt kinnitust. Kui arendaja kinnitab kustutamise, siis navigatsioon ning sellega seotud info kustutatakse. Navigatsiooni ei saa kustutada, kui see on mõne navigatsiooniregiooni poolt kasutusel.

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 kuvatakse 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.

3.7 Mittefunktsionaalsed nõuded

Tabelis 1 on välja toodud süsteemile esitatud mittefunktsionaalsed nõuded, mida loodav süsteem peab täitma.

Tabel 1. 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	<ul style="list-style-type: none">■ 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

Kogu rakenduse genereerimine ning selleks vajaliku info hoidmine toimub andmebaasi poole peal. Järgnevalt on kirjeldatud, kuidas rakenduse genereerimine toimub ning milliseid andmeid ja kuidas selleks talletatakse.

4.1 Rakenduse genereerimine

Rakenduse genereerimiseks tuleb funktsioonile `pgapex.f_app_query_page` ette anda rakenduse asukoht, rakenduse alias või id, lehe alias või id, päringu meetod (GET või POST), päringu päis (*header*), GET-parameetrid ja POST-parameetrid.

Esmalt luuakse ajutised tabelid, kuhu transaktsiooni tööaja jooksul kantakse päringuspetsiifilised seaded (nt rakenduse id ja lehe id), tagastatava päise info, süsteemi poolt genereeritud teated ning regioonide info. Võimaldamaks sama andmebaasi sessiooni raames genereerida mitu rakenduse lehte antakse ajutistesse tabelitesse andmete lisamisel kaasa ka transaktsiooni id `txid_current()`.

Seejärel kontrollitakse, kas rakendus ja leht on olemas ning avatakse sessioon. Sessiooni avamisel kontrollitakse kas kasutaja poolt saadetud päised sisaldavad küpsise infot, kus hoitakse sessiooni id-d. Kui sessiooni id on olemas, siis uuendatakse olemasoleva sessiooni aegumistähtaega. Vastasel korral genereeritakse uus sessiooni id, mis salvestatakse andmebaasi ning luuakse küpsis, mis saadetakse kasutajale tagasi päise parameetri *set-cookie* abil.

Kui rakendusele saadeti POST päring, siis kontrollitakse, kas kasutaja tahab autentida või salvestada andmeid vormi regiooni abil. Mõlemal juhul kutsutakse välja välises andmebaasis olev funktsioon. Autentimise korral oodatakse funktsioonilt tõeväärtus (*boolean*) tüüpi väärtust mille põhjal otsustatakse, kas kasutajal on ligipääs lubatud või mitte. Kui autentimisfunktsioon tagastab *TRUE*, siis lisatakse sessiooni märges, et kasutaja on autentitud.

Nii GET kui ka POST-meetodi korral kontrollitakse, kas kasutajal on ligipääs antud lehele. Kui ei ole, siis tagastatakse rakenduse seadetes määratud autentimisvorm. Kui kasutajal on õigus lehel viibida, siis käiakse läbi lehega seotud nähtavad regioonid ning genereeritakse meta-info põhjal kasutajale tagastatav leht.

Funktsioon `pgapex.f_app_query_page` tagastab JSON-objekti, mis sisaldab kahte välja

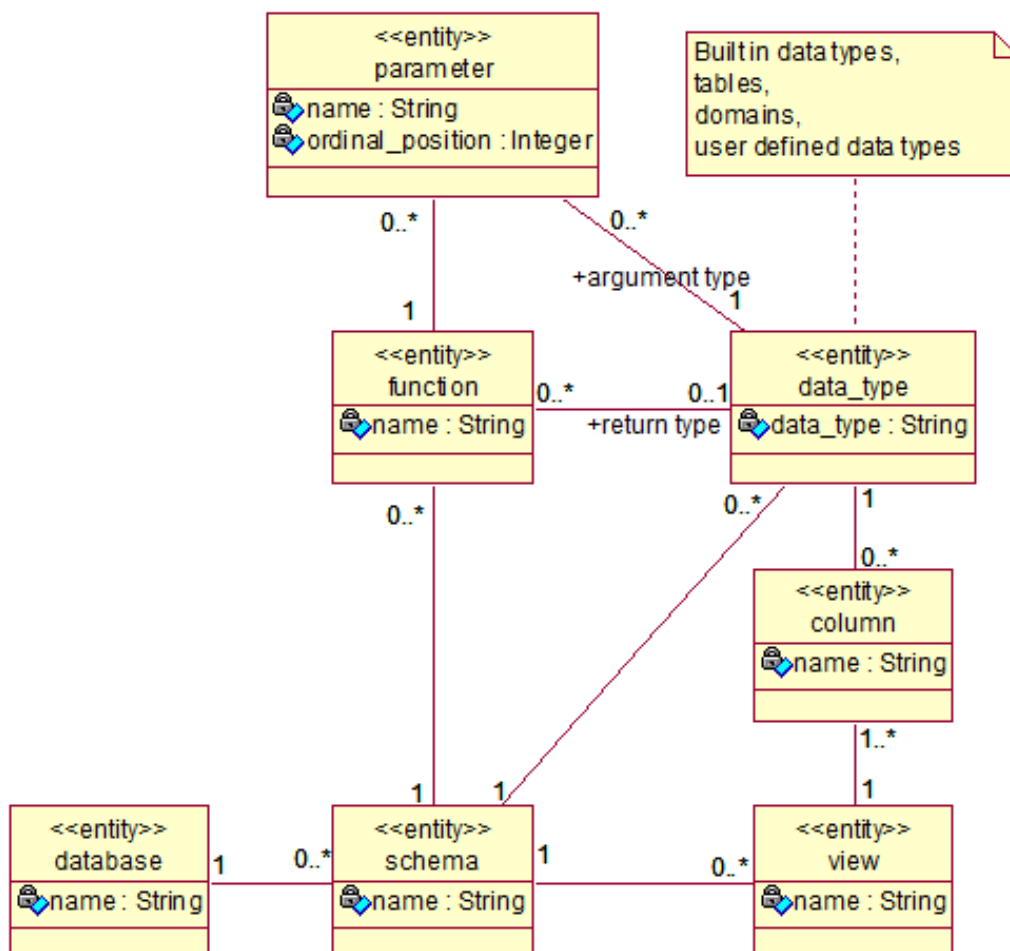
header ja *body*, mis sisaldavad vastavalt kasutajale saadetavat päist ning kuvatavat sisu.

4.2 Andmebaasikirjeldus

Järgnevalt esitatakse loodava süsteemi olemi-suhte diagrammid ja neis esitatud olemitüüpide ning nende atribuutide definitsioonid registrite kaupa. Olemi-suhte diagrammide põhjal genereeritud tabelid on Lisas 6. Kuna süsteemi lähtekood on avaldatud ning andmebaasi tabelite kirjeldus luuakse olemi-suhte diagrammide põhjal, siis kasutatakse olemi-suhte diagrammides inglise keelt.

4.2.1 Andmebaasiobjektide register

Joonisel 8 on esitatud andmebaasiobjektide registri olemi-suhte diagramm. Tabelis 2 on esitatud registrisse kuuluvate olemitüüpide definitsioonid ning tabelis 3 nende atribuutide definitsioonid.



Joonis 8. Andmebaasiobjektide registri olemi-suhte diagramm.

Tabel 2. Andmebaasiobjektide registri olemitüüpide definitsioonid.

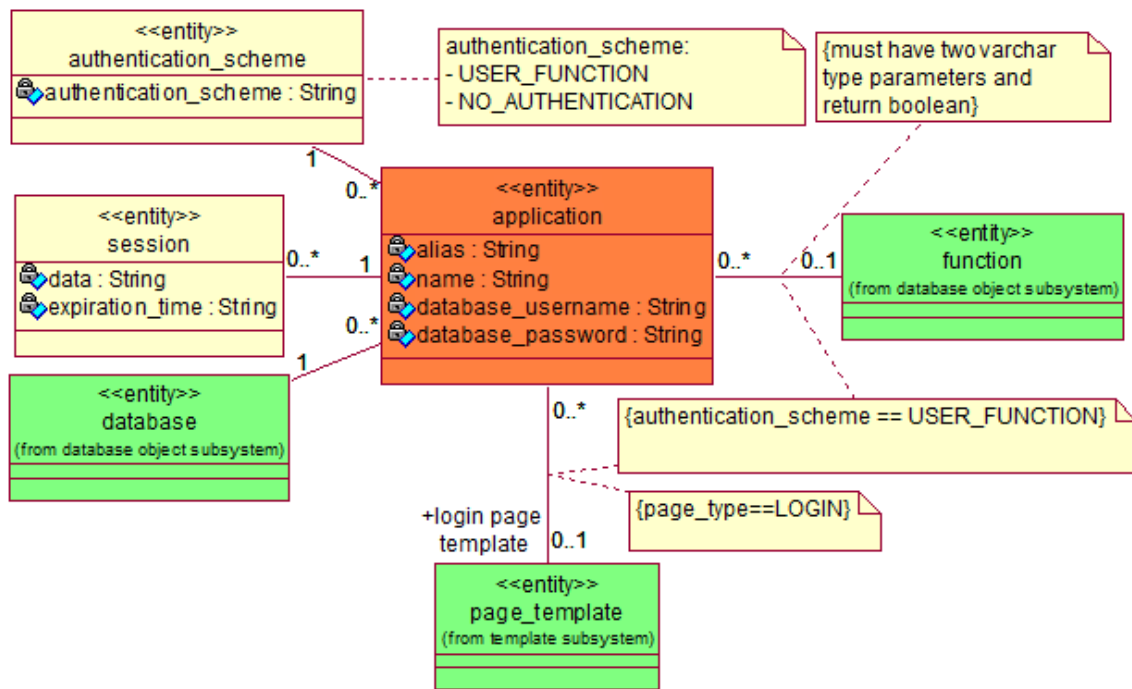
Olemitüübi nimi	Definitsioon
column	Välises andmebaasis olevas vaates esinev veerg
data_type	Välises andmebaasis kasutusel olev andmetüüp
database	Väline andmebaas on samas serveris asuv PostgreSQL andmebaas, mille kasutamiseks soovitakse rakendust luua. Selleks andmebaasiks võib ka olla metaandmete andmebaas ise
function	Välises andmebaasis olev funktsioon
parameter	Välises andmebaasis oleva funktsiooni parameeter
schema	Välises andmebaasis olev skeem
view	Välises andmebaasis olev vaade

Tabel 3. Andmebaasiobjektide registri olemitüüpide atribuutide definitsioonid.

Olemitüübi nimi	Atribuudi nimi	Definitsioon
column	name	Veeru nimi
data_type	data_type	Andmetüübi nimi. Selleks võib olla nii andmebaasi sisse ehitatud andmetüüp, domeen või kasuta poolt defineeritud tüüp
database	name	Samas andmebaasiserveris oleva andmebaasi nimi
function	name	Funktsiooni nimi
parameter	name	Parameetri nimi, kui see on olemas
parameter	ordinal_position	Parameetri esinemisjärjekord funktsioonis
schema	name	Skeemi nimi
view	name	Vaate nimi

4.2.2 Rakenduste register

Joonisel 9 on esitatud rakenduste registri olemitüüpide diagramm. Tabelis 4 on esitatud registrisse kuuluvate olemitüüpide definitsioonid ning tabelis 5 nende atribuutide definitsioonid.



Joonis 9. Rakenduste registri olemitüüpide diagramm.

Tabel 4. Rakenduste registri olemitüüpide definitsioonid.

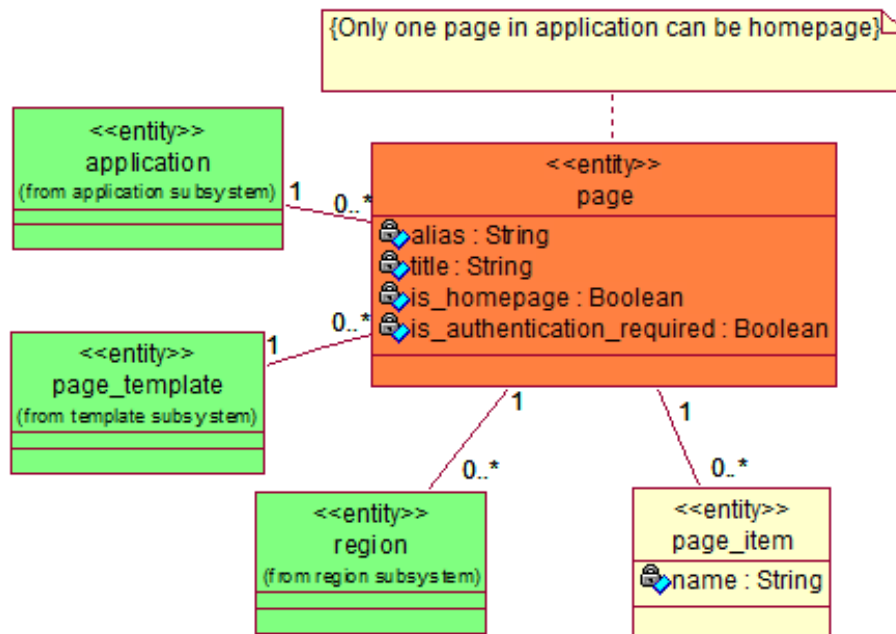
Olemitüübi nimi	Definitsioon
application	Andmebaasi põhjal loodud rakendus
authentication_scheme	Autentimismeetod, mida kasutatakse autentimist nõudvatel lehtedel
session	Rakenduse kasutaja sessiooniinfo

Tabel 5. Rakenduste registri olemitüüpide atribuutide definitsioonid.

Olemitüübi nimi	Atribuudi nimi	Definitsioon
application	alias	Tähemärkidest ning numbritest koosnev sõne rakenduse tuvastamiseks. Võimaldab luua URL aadressi, kus rakendusele viidatakse aliase abil
application	name	Rakenduse nimi
application	database_username	Kasutaja, kellest süsteem suhtleb välise andmebaasiga
application	database_password	Parool, millega sisenetakse välisesse andmebaasi
authentication_scheme	authentication_scheme	Autentimismeetod. See võib kas puududa või nõuda funktsiooni, mis teostab õiguste kontrolli
session	data	Rakenduse kasutaja sessioonis hoitavad andmed
session	expiration_time	Rakenduse kasutaja sessiooni aegumise aeg

4.2.3 Lehtede register

Joonisel 10 on esitatud lehtede registri olemitüüpide diagramm. Tabelis 6 on esitatud registrisse kuuluvate olemitüüpide definitsioonid ning tabelis 7 nende atribuutide definitsioonid.



Joonis 10. Lehtede registri olemitüüpide diagramm.

Tabel 6. Lehtede registri olemitüüpide definitsioonid.

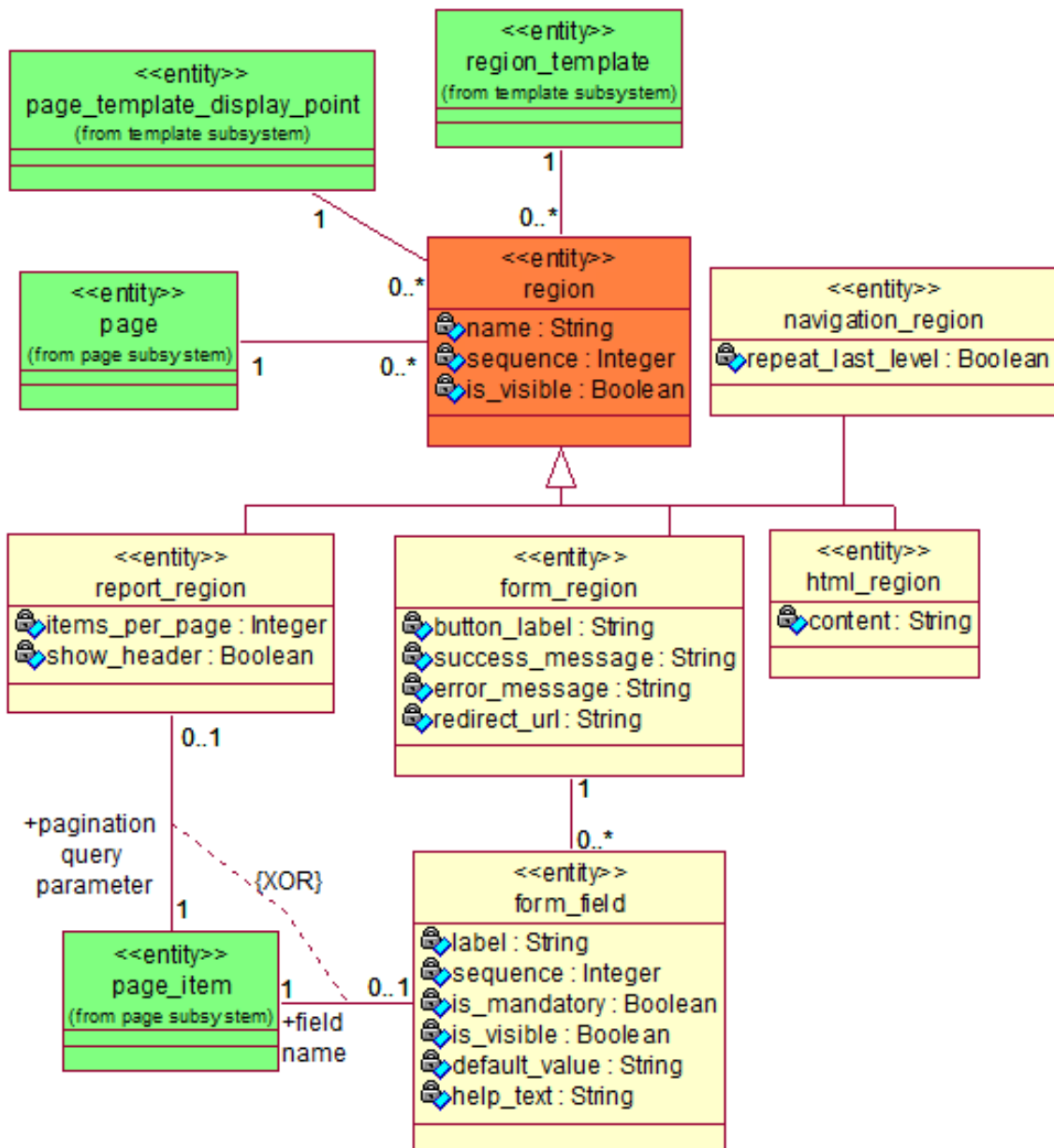
Olemitüübi nimi	Definitsioon
page	Rakenduses kuvatav leht
page_item	Lehel kasutatavad vormielemendid ning URL-parameetrid

Tabel 7. Lehtede registri olemitüüpide atribuutide definitsioonid.

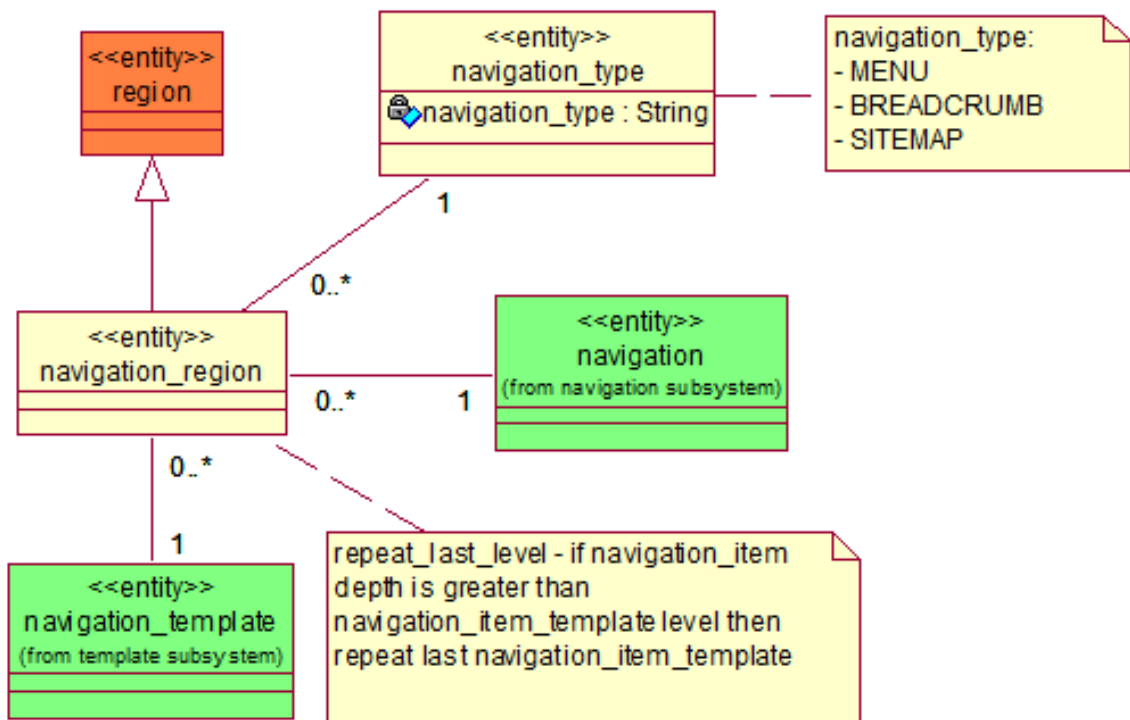
Olemitüübi nimi	Atribuudi nimi	Definitsioon
page	alias	Tähemärkidest ning numbritest koosnev sõne rakenduses lehe tuvastamiseks. Võimaldab luua URL aadressi, kus lehele viidatakse aliase abil
page	title	Lehel kuvatav pealkiri
page	is_homepage	Kas tegu on avalehega. Ühel rakendusel saab olla ainult üks avaleht. Kui rakendus sisaldab lehti, siis peab üks leht olema avaleht
page	is_authentication_required	Kas lehele pääsemiseks peab kasutaja olema autenditud
page_item	name	Vormielemendi või URL-parameetri nimi

4.2.4 Regioonide register

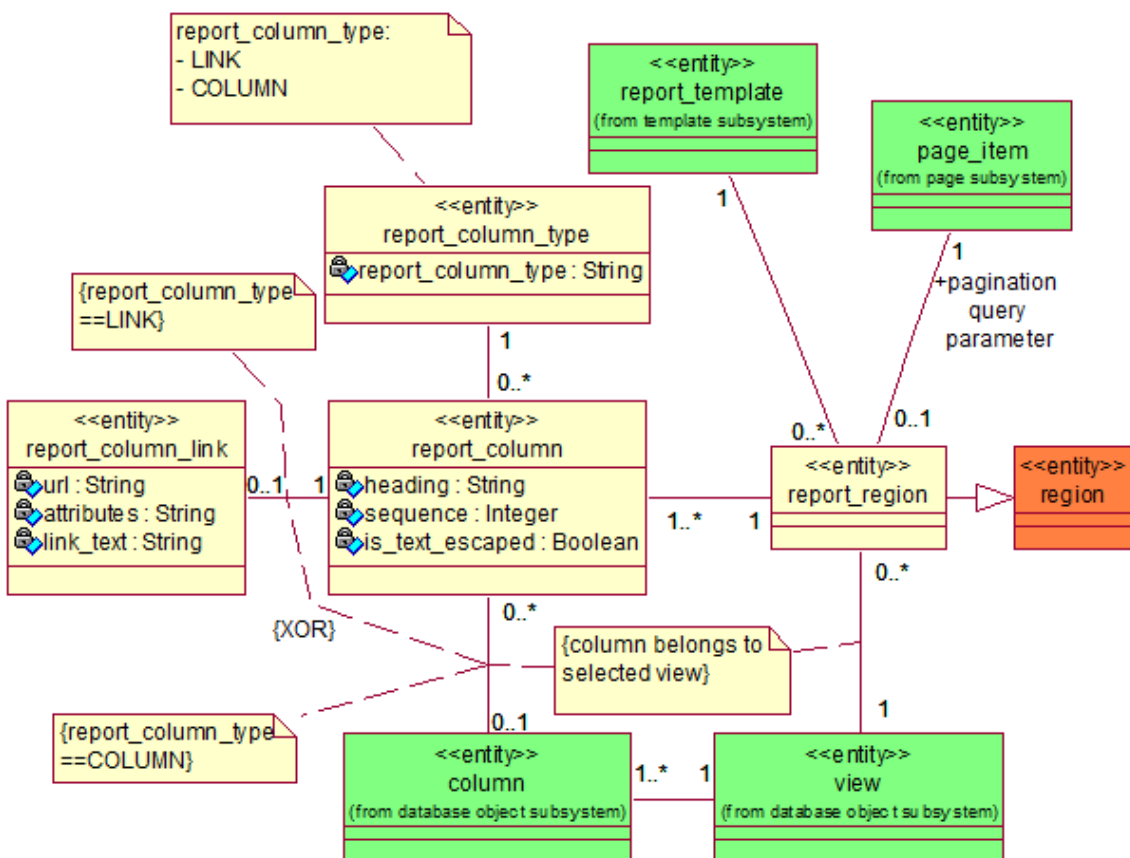
Joonistel 11, 12, 13 ja 14 on esitatud regioonide registri olemi-suhte diagramm. Tabelis 8 on esitatud registrisse kuuluvate olemitüüpide definitsioonid ning tabelis 9 nende atribuutide definitsioonid.



Joonis 11. Regioonide registri olemi-suhte diagramm osa 1.



Joonis 12. Regionide registri olemi-suhte diagramm osa 2.



Joonis 13. Regionide registri olemi-suhte diagramm osa 3.

Olemitüübi nimi	Definitsioon
list_of_values	Kui vormi elemendi tüübiks on RADIO või DROP_DOWN, siis määratakse siin ära, millise vaate väljade põhjal valikud luuakse
navigation_region	Navigatsiooni tüüpi regioon
navigation_type	Navigatsiooni tüüp
region	Lehel kuvatav allosa
report_column	Raportis kuvatav veerg
report_column_link	Raportis veerus kuvatav link, kui veeru tüübiks on LINK
report_column_type	Raporti veeru tüüp
report_region	Raporti tüüpi regioon

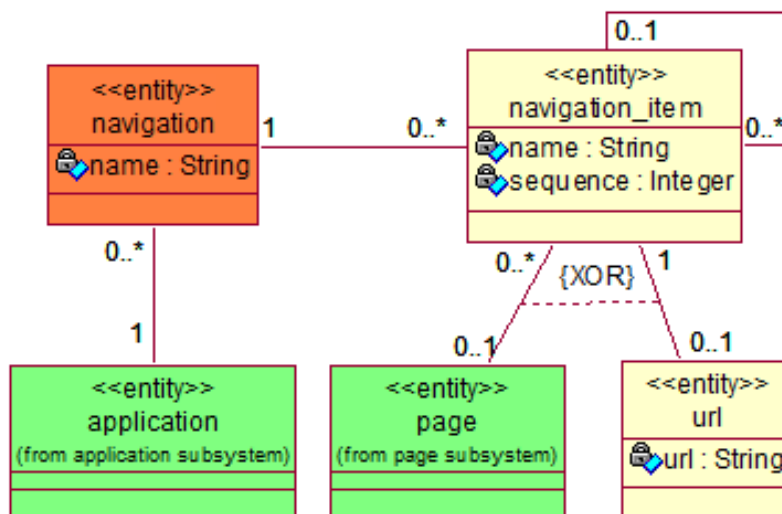
Tabel 9. Regioonide registri olemitüüpide atribuutide definitsioonid.

Olemitüübi nimi	Atribuudi nimi	Definitsioon
field_type	field_type	Välja tüübi nimi. Võimalikd väärtused on TEXT, PASSWORD, RADIO, CHECKBOX, DROP_DOWN, TEXTAREA
form_field	label	Vormi välja nimi
form_field	sequence	Vormi välja kuvamise järjekorranumber
form_field	is_mandatory	Kas välja täitmine on kohustuslik
form_field	is_visible	Kas väli on kasutajale nähtav
form_field	default_value	Välja vaikimisi väärtus juhul kui väli on tühi. Võib sisaldada samu muutujaid kui <i>form_region.redirect_url</i>
form_field	help_text	Abistav tekst, mis selgitab, milliseid andmeid kasutajalt oodatakse
form_region	button_label	Saatmisnupul kuvatav tekst
form_region	success_message	Vormi saatmisel käivitatava välise andmebaasi funktsiooni eduka lõpetamise korral kuvatav tekst
form_region	error_message	Vormi saatmisel käivitatava välise andmebaasi funktsiooni ebaeduka lõpetamise korral kuvatav tekst. Lõpetamine loetakse ebaedukaks, kui funktsioon heidab veateate
form_region	redirect_url	Aadress, kuhu kasutaja suunatakse vormi eduka töötlemise korral. Võib sisaldada muutujaid &APPLICATION_ROOT& - rakenduse faili asukoht, &APPLICATION_ID& - rakenduse id, &PAGE_ID& - lehe id, &USERNAME& - sisseloginud kasutaja kasutajanimi
html_region	content	Lehel kuvatav sisu. Võib sisaldada HTML-i
navigation_region	repeat_last_level	Kui navigatsiooni sügavamatele elementidele pole loodud vastavat malli, siis kasutatakse viimast navigatsioonipunkti malli, et kuvada need navigatsioonipunktid. Vastasel juhul jäävad need kuvamata
navigation_type	navigation_type	Navigatsiooni tüüp. Selle põhjal otsustatakse, kuidas tuleb navigatsioon valmis renderdada. Võimalikd väärtused on MENU, BREADCRUMB, SITEMAP

Olemitüübi nimi	Atribuudi nimi	Definitsioon
region	name	Regiooni nimi. Võidakse kuvada lehel
region	sequence	Regiooni kuvamise järjekord
region	is_visible	Kas regioon on lehel nähtav
report_column	heading	Raporti veeri päise pealkiri
report_column	sequence	Raporti veeru kuvamise järjekord
report_column	is_text_escaped	Kas veerus kuvatavas tekstis muudetakse HTML erimärgid ohutuks
report_column_link	url	Aadress, millele link viitab. Võttesõna %VEERU_NIMI% asendatakse vastavas veerus oleva väärtusega. Võib sisaldada samu muutujaid kui <i>form_region.redirect_url</i>
report_column_link	attributes	Lisaatribuudid, mida on võimalik lingile lisada
report_column_link	link_text	Lingil kuvatav tekst. Võib sisaldada samu võttesõnu ja muutujaid kui <i>report_column_link.url</i>
report_column_type	report_column_type	Raporti veeru tüüp. Võimalikud väärtused on COLUMN, LINK. COLUMN-i korral kuvatakse vaatest saadava välja sisu, LINK-i korral aga link.
report_region	items_per_page	Mitu rida raportist kuvatakse ühel leheküljel. Ülejäänud ridade nägemiseks luuakse lingid
report_region	show_header	Kas raportil kuvatakse veerude päiste pealkirjad

4.2.5 Navigatsioonide register

Joonisel 15 on esitatud navigatsioonide registri olemi-suhte diagramm. Tabelis 10 on esitatud registrisse kuuluvate olemitüüpide definitsioonid ning tabelis 11 nende atribuutide definitsioonid.



Joonis 15. Navigatsioonide registri olemi-suhte diagramm.

Tabel 10. Navigatsioonide registri olemitüüpide definitsioonid.

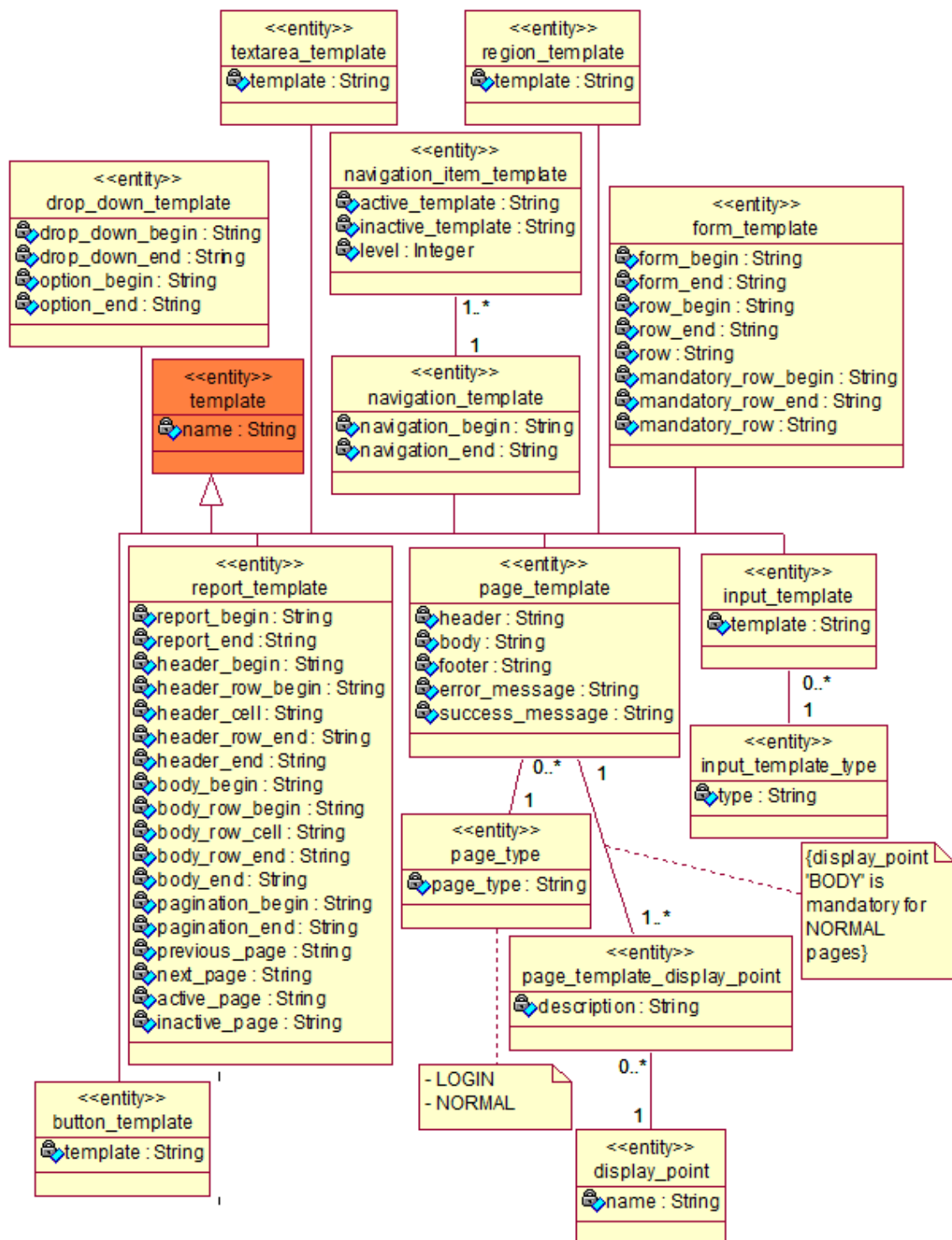
Olemitüübi nimi	Definitsioon
navigation	Navigatsioon grupeerimaks erinevaid navigatsioonihierarhiaid
navigation_item	Navigatsioonis kuvatav element. Võib sisaldada alamelemente
url	Aadress, kuhu navigatsioonis kuvatav element suunab

Tabel 11. Navigatsioonide registri olemitüüpide atribuutide definitsioonid.

Olemitüübi nimi	Atribuudi nimi	Definitsioon
navigation	name	Navigatsiooni kirjeldav nimi
navigation_item	name	Navigatsioonielemendi kuvamiseks kuvatav tekst
navigation_item	sequence	Navigatsioonielemendi kuvamisjärjekord oma hierarhia tasemel
url	url	Aadress, kuhu navigatsioonis kuvatav element suunab

4.2.6 Mallide register

Joonisel 16 on esitatud mallide registri olemi-suhte diagramm. Tabelis 12 on esitatud registrisse kuuluvate olemitüüpide definitsioonid ning tabelis 13 nende atribuutide definitsioonid.



Joonis 16. Mallide registri olemi-suhte diagramm.

Tabel 12. Mallide registri olemitüüpide definitsioonid.

Olemitüübi nimi	Definitsioon
button_template	Nupu mall
display_point	Asukohad, kuhu on võimalik lisada regioone
drop_down_template	HTML select-elementi mall
form_template	Vormi mall
input_template	HTML input-elementi mall
input_template_type	Võimalikud elementi kuvamisviisid
navigation_item_template	Navigatsioonipunkti mall
navigation_template	navigatsiooni mall
page_template	Lehe mall
page_template_display_point	Lehel kasutatavad asukohad, kuhu on võimalik lisada regioone
page_type	Lehe malli tüüp
region_template	Regiooni mall
report_template	Raporti mall
template	Üldine malli kirjeldus
textarea_template	HTML textarea-elementi mall

Tabel 13. Mallide registri olemitüüpide atribuutide definitsioonid.

Olemitüübi nimi	Atribuudi nimi	Definitsioon
button_template	template	Nupu mall. Võib sisaldada võtmesõnu #NAME# ja #LABEL#, mis asendatakse vastavalt vormi elementi nimega ning kasutajale kuvatava tekstga
display_point	name	Asukoha nimi
drop_down_template	drop_down_begin	HTML select-elementi algus. Võib sisaldada võtmesõnu #ROW_LABEL# ja #NAME#, mis asendatakse vastavalt vormi välja nimega ning vormi elementi nimega
drop_down_template	drop_down_end	HTML select-elementi lõpp
drop_down_template	option_begin	HTML option-elementi algus. Võib sisaldada võtmesõnu #VALUE# ja #SELECTED#, mis asendatakse vastavalt valiku väärtusega ning millega tähistatakse aktiivset valikut
drop_down_template	option_end	HTML option-elementi lõpp
form_template	form_begin	Vormi algus. Võib sisaldada võtmesõna #SUBMIT_BUTTON#, mis asendatakse saatmisnupuga
form_template	form_end	Vormi lõpp. Võib sisaldada võtmesõna #SUBMIT_BUTTON#, mis asendatakse saatmisnupuga
form_template	row_begin	Vormi rea algus
form_template	row_end	Vormi rea lõpp

Olemitüübi nimi	Atribuudi nimi	Definitsioon
form_template	row	Vormi rida. Võib sisaldada võtmesõnu #FORM_ELEMENT#, #HELP_TEXT# ja #LABEL#, mis asendatakse vastavalt vormi elemendi, abistava teksti ja välja nimega
form_template	mandatory_row_begin	Vormi rea algus, mis sisaldab kohustuslikku elementi
form_template	mandatory_row_end	Vormi rea lõpp, mis sisaldab kohustuslikku elementi
form_template	mandatory_row	Vormi rida, mis sisaldab kohustuslikku elementi. Võib sisaldada samu võtmesõnu kui <i>form_template.row</i>
input_template	template	HTML input-elemendi kujundus
input_template_type	type	Võimalikud elemendi kuvamisviisid: TEXT, PASSWORD, CHECKBOX, RADIO
navigation_item_template	active_template	Aktiivse navigatsioonielemendi mall
navigation_item_template	inactive_template	Mitteaktiivse navigatsioonielemendi mall
navigation_item_template	level	Navigatsioonielemendi sügavus, millal antud malli rakendatakse
navigation_template	navigation_begin	Navigatsiooni algus
navigation_template	navigation_end	Navigatsiooni lõpp
page_template	header	Lehe päis. Võib sisaldada võtmesõnu #TITLE#, #APPLICATION_NAME#, #ERROR_MESSAGE#, #SUCCESS_MESSAGE#, #LOGOUT_LINK#, mis asendatakse vastavalt lehe pealkirjaga, rakenduse nimega, veateatega, teatega, väljalogimise URL-aadressiga
page_template	body	Lehe sisuosa. Võib sisaldada samu võtmesõnu kui <i>page_template.header</i> ning lisaks #BODY#, mis asendatakse regiooniga. Võib sisaldada veel teisigi positsiooni määravaid võtmesõnu, mis regioonidega asendatakse
page_template	footer	Lehe jalus. Võib sisaldada samu võtmesõnu kui <i>page_template.header</i>
page_template	error_message	Lehel kuvatav veateade. Võib sisaldada võtmesõna #MESSAGE#, mis asendatakse vastava teatega
page_template	success_message	Lehel kuvatav teade. Võib sisaldada võtmesõna #MESSAGE#, mis asendatakse vastava teatega
page_template_display_point	description	Lehel oleva asukoha kirjeldus
page_type	page_type	Lehe malli tüüp. Võimalikud väärtused: LOGIN, NORMAL. NORMAL tüüpi leht peab sisaldama #BODY# võtmesõna
region_template	template	Regiooni mall

Olemitüübi nimi	Atribuudi nimi	Definitsioon
report_template	report_begin	Raporti algus. Võib sisaldada võtmesõna #PAGINATION#, mis asendatakse lehtede jaotamise linkidega
report_template	report_end	Raporti lõpp. Võib sisaldada samu võtmesõnu kui <i>report_template.report_begin</i>
report_template	header_begin	Raporti päise algus. Võib sisaldada samu võtmesõnu kui <i>report_template.report_begin</i>
report_template	header_row_begin	Raporti päise rea algus
report_template	header_cell	Raporti päise kast, kus kuvatakse veeru pealkiri
report_template	header_row_end	Raporti päise rea lõpp
report_template	header_end	Raporti päise lõpp. Võib sisaldada samu võtmesõnu kui <i>report_template.report_begin</i>
report_template	body_begin	Raporti sisuosa algus. Võib sisaldada samu võtmesõnu kui <i>report_template.report_begin</i>
report_template	body_row_begin	Raporti sisuosa rea algus
report_template	body_row_cell	Raporti sisuosa kast, kus kuvatakse infot
report_template	body_row_end	Raporti sisuosa rea lõpp
report_template	body_end	Raporti sisuosa lõpp. Võib sisaldada samu võtmesõnu kui <i>report_template.report_begin</i>
report_template	pagination_begin	Lehtede jaotamise algus
report_template	pagination_end	Lehtede jaotamise lõpp
report_template	previous_page	Link eelmisele lehele. Võib sisaldada võtmesõna #LINK#, mis asendatakse viitega eelmisele lehele
report_template	next_page	Link järgmisele lehele. Võib sisaldada võtmesõna #LINK#, mis asendatakse viitega järgmisele lehele
report_template	active_page	Aktiivse lehe link. Võib sisaldada võtmesõnu #LINK# ja #NUMBER#, mis asendatakse vastavalt lehe aadressiga ning lehe numbriga
report_template	inactive_page	Mitteaktiivse lehe link. Võib sisaldada samu võtmesõnu kui <i>report_template.active_page</i>
template	name	Malli nimi
textarea_template	template	HTML textarea-elementi mall

5 Kasutatavad tehnoloogiad ja arendusprotsess

Järgnevalt antakse ülevaade, miks ja milliseid programme ning raamistikke süsteemi loomisel kasutati. Seejärel antakse ülevaade süsteemi arendamise protsessist.

5.1 Vagrant

Vagrant on käsureaprogramm, millega saab hallata virtuaalmasina elutsükli. Vagrant isoleerib programmilised sõltuvused ja nende konfiguratsioonid ühtsesse eraldiseisvasse keskkonda. Keskkonna konfigureerimiseks saab kasutada käsurea käsklusi, *Ansible*-t [2], *Puppet*-it [40], *Chef*-i [7], *Docker*-it [10] ja *Salt*-i [41]. Tänu Vagrantile saavad kõik luua endale täpselt ühesuguse keskkonna, kus programme käivitada, vähendades võimalust, et ühes arvutis programm töötab, teises aga mitte. [45]

5.2 Bower

Bower on paketi haldussüsteem (*package manager*), mis on mõeldud veebis kasutatavate failide - 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. Tänu sellele ei pea arendaja tegelema koodus kasutatavate pakettide haldamisega. [5]

5.3 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].

5.4 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 ja CSS-i. [4]

5.5 TravisCI

TravisCI on pideva integratsiooni (*continuous integration*) arendusvahend, mille abil saab luua virtuaalse keskkonna koodi kompilleerimiseks, testimiseks ja juurutamiseks. Keskkonna seadistamine toimub faili `.travis.yml` abil, kus määratakse ära virtuaalkeskkonna operatsioonisüsteem, installitavad teegid ning käivitavad käsud. [43]

5.6 PHP

PHP (*PHP: Hypertext Preprocessor*) on avatud lähtekoodiga skriptimiskeel, mis on peamiselt mõeldud veebiprogrammeerimiseks. [47] PHP koodi protsessitakse PHP interpreteri abil. Üldjuhul kasutatakse interpreteerimiseks *Zend Engine*-t, kuid PHP koodi on võimalik interpreteerida ka *HHVM*-i [14] abil. PHP toetab erinevaid operatsioonisüsteeme, sealhulgas Windows-i erinevaid versioone ja Linuxi erinevaid distributsioone.

5.7 Composer

Composer on PHP-s kirjutatud sõltuvuste haldamise süsteem. Sõltuvused kirjeldatakse `composer.json` failis ning Composer ise tegeleb nende allalaadimisega ning uuendamisega. [8]

5.8 Slim Framework 3

Slim [42] on PHP mikro-raamistik. See tähendab, et temas on realiseeritud põhifunktsionaalsused ning paljud lisad on välja jäetud. Kuna loodud süsteemi PHP-rakenduse poolne osa on üpriski õhuke ning lihtne, siis valitigi antud raamistik.

Slim koosneb järgmistest põhiosadest:

- Marsruuter (*router*) analüüsib kasutaja poolt tehtud päringut ning võrdleb seda defineeritud marsruutidega. Kui marsruutide seast leitakse sobiv vaste, siis initsialiseeritakse vastav kontrolleri ning päring edastatakse kontrolleri.
- Kontrolleri (*controller*) antakse ette päringu (*request*) ja vastuse (*response*) objektid. Päringu objekt sisaldab kasutaja poolt saadetud infot ning vastuse objekti peab kontrolleri lisama kasutajale kuvatava vastuse.
- Vahevara (*middleware*) abil on võimalik manipuleerida päringu ja vastuse objekte enne ning pärast kontrolleri jõudmist.

5.9 Postgresql

PostgreSQL on avatud lähtekoodiga objekt-relatsiooniline andmebaasisüsteem, mis vastab täielikult *ACID* nõuetele. See toetab muuhulgas välisvõtmete deklareerimist, tabelite ühendamise operatsioone, vaateid, trigereid ja andmebaasiserveris talletatud funktsioone. PostgreSQL toetab erinevaid operatsioonisüsteeme, sealhulgas Windows-i erinevaid versioone ja Linuxi erinevaid distributsioone. [33]

5.10 Arendusprotsess

Sellise mahuka ja potentsiaalselt väga suure funktsionaalsuse hulgaga süsteemi puhul on selge, et arendus peab toimuma iteratiivselt ja inkrementaalselt ning magistritöö tulemusena pole reaalne tervet süsteemi valmis saada.

Alustuseks sai paika pandud esmased nõuded, mida süsteem peaks võimaldama teha. Kuna süsteemi esimene võimalik kasutuskoht oleks TTÜ-s õpetatavas aines “Andmebaasid II”, siis konsulteeriti antud õppeaine õppejõuga ning kaardistati enamlevinud kasutusjuhud, mida antud aine raames tuleb üliõpilastel andmebaasirakenduste ehitamisel realisee-

rida.

Neid silmas pidades loodi süsteemi esialgne tükeldus allsüsteemideks, mida koos juhendajaga üle vaadates pandi paika prioriteedid ja määrati kindlaks allsüsteemid, millele antud töös keskenduda. Need allsüsteemid on toodud välja ka töö peatükis 3.

Allsüsteemide tükeldust silmas pidades loodi süsteemi kasutatavaid metaandmeid kirjeldav kontseptuaalne andmemudel ning kasutajaliidese prototüüp, mis võeti hiljem loodavas süsteemis kasutusele. Prototüüp kasutas andmete kuvamiseks võltsandmeid. See andis hea ettekujutuse loodava süsteemi võimekusest ning aitas juhtida tähelepanu aspektidele, millele ilma prototüübi abita ei oleks kohe tuldud. Kontseptuaalsele andmemudelile teisendusreegleid rakendades jõuti metaandmete andmebaasi tabelite kirjeldusteni.

Selleks, et loodavat süsteemi oleks ka teistel arendajatel lihtsam kasutusele võtta ning täiustada, sai arenduskeskkonna loomiseks kasutatud Vagrant-i [44], 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 veebilehitsejates ning eri suurustes ekraanidega, kasutati Bootstrap 3 [4]. Kasutajaliidese poolt kasutatavaid sõltuvusi hallati Bower-i [5] abil. Koodi kvaliteedi kontrollimiseks ja säilitamiseks kirjutati testid, mille käivitamiseks kasutati Karma-t [15].

Arendussüsteemi 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 [32] 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 [30] kirjutatud rakenduse. Kuna antud rakenduse kiht on üpriski õhuke, siis sai selle loomiseks valitud ka lihtsakoeline raamistik Slim Framework 3 [42]. PHP-s kirjutatud koodi testimiseks kasutati PHPUnit-it [31] ning Mockery-t [20]. Koodi sõltuvusi hallati Composer-i abil [8].

Koodi hoidmiseks kasutatakse GitHub-i [11]. Iga kord, kui koodihoidlasse midagi üles laetakse luuakse TravisCI-s [43] virtuaalkeskkond, kuhu tõmmatakse GitHub-st loodava süsteemi kood ning testide eduka läbimise korral juurutatakse serverisse. Tänu sellele on serveris alati näha süsteemi viimane töötav versioon.

6 Kasutajaliides

Kasutajaliides koosneb neljast põhiosast, milleks on moodulid, vaated, teenused ning tõlkefailid. Iga allsüsteemi haldamiseks ning kasutajate autentimiseks on loodud eraldi moodul. Moodulis on ära määratud, mis lehe korral mingi kontrolleri käivitatakse 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-ile kasutaja tuli ja käivitatakse vastav kontrolleri ning laetakse sisse vaade, mida antud kontrolleri kasutab. Kontrolleri käivitamisel antakse talle ette tööks vajalikud teenuste objektid. Selleks kasutatakse sõltuvuste süstamise disainimustrit. See tähendab, et kontrolleri loomisel vaadatakse, milliseid teenuseid ta vajab ja luuakse vajalikud eksemplarid ning antakse need kontrolleri kaasa. Seetõttu ei pea kontrolleri ise tegelema vajalike teenuste initsialiseerimisega.

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

Teenuste abil suheldakse rakenduse serveriga AJAX-i abil. Tänu sellele ei pea andmete küsimisel ega saatmisel kogu lehte uuesti laadima, vaid vajaliku info edastamine toimub rakenduse taustal. Andmete küsimine toimub GET-meetodi abil ning andmete lisamine, muutmine ja kustutamine POST-meetodi abil.

Olulisemad vaated kasutajaliidese on välja toodud Lisas 7 Joonistel 33, 34, 35, 36.

7 Rakenduse disain

Rakendus on kirjutatud PHP programmeerimiskeeles. Lihtsustamaks ning kiirendamaks rakenduse loomist võeti kasutusele PHP raamistik Slim Framework 3 [42].

7.1 Rakenduse ülesehitus

Loodud rakendus koosneb järgnevatest põhiosadest: kontrolleriid (*controller*), vahevara (*middleware*), mudelid (*model*), teenused (*service*) ja marsruuter (*router*).

7.1.1 Kontrolleriid

Kontrolleriit initsialiseeritakse Slim raamistiku poolt ning sellele antakse ette päringu ja vastuse objektid ning olemasolu korral ka URL-parameetrid. Kontrolleris saadetakse vajaduse korral kasutaja poolt tulnud andmed edasi valideerimisteenusele, mis kontrollib sisendandete korrektsust. Kui andmetest vigu ei leitud, siis saadetakse need edasi mudelisse, mis saadab need omakorda edasi andmebaasi. Mudeli poolt tagastatavad andmed edastatakse kasutajale.

Loodud süsteem sisaldab järgmisi kontrolleriite:

- **AppController** - Kasutatakse loodud rakendusega suhtlemiseks. Annab mudelile ette PHP-faili asukoha, rakenduse ja lehe id või aliase, mille vastu päring tehti, päringu meetod (GET või POST), päised, GET-parameetrid ja POST-parameetrid. Mudelilt oodatakse tagasi JSON-objekti, mis peab sisaldama (*header*) ja (*body*) välju. *Header*-is olevad võti-väärtus paarid lisatakse HTTP(S) vastuse päisesse ning *body*-s olev sisu kuvatakse kasutajale.
- **ApplicationController** - Haldab rakenduste kohta info küsimise ning rakenduste loomise, muutmise ja kustutamise päringuid.
- **AuthController** - Tegeleb arendajate autentimisega ja väljalogimisega.
- **DatabaseController** - Haldab päringuid, mis küsivad infot andmebaasiobjektide kohta.
- **NavigationController** - Haldab navigatsiooni ja navigatsioonipunktidega seotud info küsimise, lisamise, muutmise ja kustutamise päringuid.

- PageController - Haldab lehtedega seotud info küsimise, lisamise, muutmise ja kustutamise päringuid.
- RegionController - Haldab regioonidega seotud info küsimise, lisamise, muutmise ja kustutamise päringuid.
- TemplateController - Haldab kõiki päringuid, mis küsivad infot kõikvõimalike mallide kohta.

Kõik kontrollid peale ApplicationController'i jagastavad vastused JSON-na.

7.1.2 Vahevara

Vahevara abil kontrollitakse, kas päringud vastavad vajalikele tingimustele. Loodud süsteemis kasutatakse järgmisi vahevarasid:

- ApiMiddleware - Kontrollib, kas API vastu tehtavad päringud sisaldavad *X-Requested-With* päist väärtusega *XMLHttpRequest*. Selle abil tõstetakse süsteemi turvalisust vähendades *CSRF* rünnaku võimalust, kuna seda päist ei saa lisada AJAX-päringutele, mis küsivad infot teisest domeenist.
- AuthMiddleware - Kontrollib, kas arendaja on sisse loginud ning omab õigust vastavaid päringuid teostada.

7.1.3 Mudelid

Mudelite abil toimub suhtlus andmebaasiga. Selleks kasutatakse PHP laiendust PDO. PDO on liides, mille abil pääseb ligi andmebaasidele. Iga andmebaasisüsteemi jaoks tuleb kasutada vastavat juhtprogrammi (*driver*). Andmebaasiga ühendamiseks tuleb ette anda juhtprogrammi nimi, andmebaasiserveri asukoht ja port, mida andmebaasisüsteem kuulab ning kasutajanimi ja parool (vt Joonis 17). [28]

```
new PDO( 'pgsql:host=localhost;port=5432;dbname=pgapex', 'username', 'password' );
```

Joonis 17. PHP: Andmebaasiga ühendamine.

Vältimaks SQL süstimist (*SQL injection*) kasutatakse päringute loomiseks *prepare* meetodit ning väärtused antakse edasi *bindValue* meetodi abil. Joonisel 18 on näidatud, kuidas teostatakse kasutaja õiguste kontroll.

```
$statement = $connection->prepare('SELECT pgapex.  
    f_is_superuser(:username, :password)');  
$statement->bindValue(':username', $username);  
$statement->bindValue(':password', $password);  
$statement->execute();  
return $statement->fetchColumn() === true;
```

Joonis 18. PHP: Kasutaja õiguste kontroll.

7.1.4 Teenused

Teenustesse on talletatud loogika, mida võivad kasutada mitmed süsteemi osad. Loodud rakenduses on kasutusel järgmised teenused:

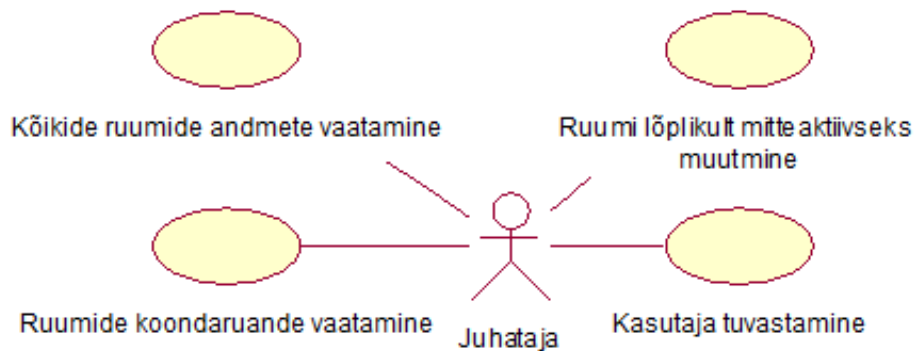
- Authentication - Tegeleb kasutaja autentimisega ning kontrollib kas kasutajal on lubatud ligipääs süsteemi osadele.
- Database - Tegeleb andmebaasiühenduse loomise, haldamise ning sulgemisega.
- Session - Haldab sessioonide loomist ja kustutamist. Läbi selle teenuse saab lisada ning küsida sessioonis hoitavaid andmeid.
- Validator teenused - Kontrollivad kasutaja poolt saadetud andmete korrektsust. Kõik seda tüüpi teenused realiseerivad *validate* meetodi, kuhu antakse ette päringu-objekt, mille põhjal kontrolli teostatakse.

7.1.5 Marsruuter

Marsruuteris on ära kirjeldatud, millise päringu korral mingi kontrolleri käivitatakse. Lisaks on seal ära defineeritud, millistele päringutele poogitakse külge vahevara.

8 Näidisrakendus

Valideerimaks, kas loodud süsteem vastab nõuetele, loodi näiterakendus ja realiseeriti neli kasutusjuhtu (vt Joonis 19), mis sarnanevad üliõpilaste töödes esinevatele kasutusjuhtudele. [29] Rakendus realiseerib hotelli infosüsteemi ruumide arvestuse allsüsteemi funktsionaalsuse juhataja töökoha ulatuses. Kasutaja tuvastamine põhineb funktsioonil *functions.f_is_boss*, mille esimese parameetri oodatav väärtus on kasutajanimi ja teise parameetri oodatav väärtus on paool. Kõikide ruumide andmete 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_temporariily_inactive_rooms*.



Joonis 19. Näidisrakenduse kasutusjuhtude diagramm.

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 20). Kui kasutajanimi ja parool on õiged, siis logitakse kasutaja sisse ning kasutaja näeb edaspidi autentimist nõudvate lehtede sisu.

The image shows a login form with a dark header bar containing the text 'Ruumid'. Below the header, there are two input fields. The first field has a user icon on the left and the placeholder text 'Kasutajanimi'. The second field has a lock icon on the left and a series of dots representing a password. Below these fields is a blue button with the text 'Login'.

Joonis 20. Näidisrakendus: Kasutaja tuvastamine.

8.2 Ruumi lõplikult mitteaktiivseks muutmine

Kasutajale kuvatakse vorm, kus kust ta saab valida millist ruumi ta muuta soovib (vt Joonis 21). Ruumide loetelu saadakse vaate *public.active_temporarily_inactive_rooms* põhjal. Pärast vormi saatmist kutsutakse välja funktsioon *functions.f_permanently_inactivate_a_room*. Kui funktsioon lõpetab oma töö ilma vigadeta, siis tagastatakse kasutajale teade, et vormi töötlemine õnnestus.

The image shows a web interface with a dark navigation bar at the top containing links: 'Ruumid', 'Ruumide info', 'Lõpeta ruume', and 'Logi välja'. Below the navigation bar is a green message box that says 'Ruumi olek muudetud'. Underneath is a form titled 'Muuda'. The form contains a label 'Ruum *' followed by a dropdown menu. The dropdown menu is open, showing 'Big room' as the selected option. Below the dropdown menu is a blue button with the text 'Muuda ruumi olekut'.

Joonis 21. Näidisrakendus: 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 mitmele leheküljele (vt Joonis 22)

Ruumid
Ruumide info
Lõpeta ruume
Logi välja

Sissejuhatus

Vaadake aruandeid ruumide kohta.

Ruumide koondaruanne

Ruumi olek	Ruumide arv
Permanently inactive	1
Waiting	1

1
2
»

Kõikide ruumide andmed

Voodi tüüp	Olek	Hind/öö	Isik	Hinnavahe	Registreerimisaasta	Ruumi kood	Ruumi nimi
Single	Waiting	60.00	Kaarel Kask kask@ttu.ee	10.00	2016	333	BIG ROOM
Double	Permanently inactive	70.00	Kaarel Kask kask@ttu.ee	40.00	2016	444	SMALL ROOM

Joonis 22. Näidisrakendus: Ruumide koondaruande ja kõikide ruumide vaatamine.

9 Arendusvaade

Loodud süsteemi funktsionaalsus on autori hinnangul piisav, et seda saaks reaalse tarkvara loomiseks kasutada, kuid edasiste iteratsioonide käigus võiks süsteemi kindlasti täiustada. Kuna loodud süsteem võimaldab luua rakenduse ka metaandmete andmebaasile, siis saab süsteemi ennast kasutada süsteemi laienduste loomiseks. Järgnevalt on esitatud loetelu funktsionaalsustest, mida võiks süsteemile lisada.

- Loodud rakenduse kirjeldust peaks olema võimalik eksportida ning importida. Sellisel juhul oleks võimalik rakendust varundada ning taastada ja kasutada versiooni-haldustarkvara rakenduse versioonide haldamiseks. Kuna loodud süsteem kasutab ka ise andmebaasiliidest ning kõik rakendustega seotud tegevused tehakse läbi andmebaasifunktsioonide, siis saaks neid samu funktsioone kasutada rakenduste kirjeldamiseks. Sellisel juhul meenutaks loodud kirjeldus tavaprogrammeerimises tavalisi funktsioonide väljakutseid, mis oleks semantiliselt arusaadavam kui tavaline andmetõmmis (*data dump*).
- Arendajal peaks olema võimalus näha vealogisid, sessiooniandmeid, vormidega saadetavaid andmeid ning muud abistavat infot, mis aitaksid probleemide korral rakendust siluda.
- Loodud süsteem on hetkel ingliskeelne, et soodustada selle laiemat levikut. Arendajal võiks aga olla võimalus muuta kasutajaliideses kasutatavat keelt. Preagune tekst tuleb juba tõlkefailist ning seetõttu tuleks ainult realiseerida keelevaliku korral uute keelefailide sisselaadimine.
- Hetkel saavad kõik rakendused kasutada üksnes eeldefineeritud malle. Iga rakenduse jaoks võiks aga saada luua spetsiifilisi malle, mis vastaksid täpselt rakenduse vajadustele.
- Andmabaasiobjektid välistes andmebaasides võivad muutuda ning kaduda. Süsteem peaks olema võimeline kontrollima, millised andmabaasiobjektid on muutunud niivõrd, et ei võimalda rakendusel enam sihipäraselt töötada ning teavitada sellest arendajat.
- Praeguses süsteemis sai realiseeritud neli erinevat regioonitüüpi: navigatsioon, HTML-tekst, raport ja vorm. Alati ei pruugi nendest aga piisata. Seetõttu tuleks uurida, millisel kujul oleks veel vaja infot kuvada ning realiseerida vastavad regioonid. Näiteks võiks olla võimalik kujutada infot graafikutena või punktidenägemisena maailmakaardil.

- Arendajal oleks mugav rakendust arendada, kui ta saaks võimalikult palju tegevusi teha ühes ja samas keskkonnas. Seetõttu võiks süsteem võimaldada hallata välises andmebaasis olevaid andmebaasiobjekte ning andmeid.
- Kui rakenduse kasutajad muudavad samal ajal samu andmeid, siis esimesena salvestatud andmed salvestatakse viimase poolt üle. Süsteem võiks sellisel juhul teavitada järgmist kasutajat, kes üritab salvestada, et andmed on vahepeal muutunud.
- Süsteem võiks võimaldada luua REST-liideseid, mille abil saaksid nii loodavad rakendused, kui ka välised süsteemid välistest andmebaasidest infot küsida.
- Realiseerida rakenduste loojatele väljapakkumiseks kasutajate autentimise meetod, mille korral igale rakenduse lõppkasutajale peab vastama andmebaasi kasutaja. See tähendab, et rakendusse logimisel tuleb sisestada andmebaasi kasutaja kasutajanimi ja parool ning lõppkasutaja saab teha andmebaasis (ja seega ka süsteemis) tegevusi andmebaasi kasutajale määratud õiguste piires.
- Tulevikus peaks olema võimalik muuta rakenduse seisundit, mis omakorda määrab selle, kas see on kasutajatele kättesaadav või mitte. Praegu puudub võimalus rakenduse hetkeseisundi muutmiseks ja selle kaudu rakenduse lõppkasutajate eest peitmiseks. Hetkel on ainus võimalus rakendus kustutada. Arvestades ka sellega, et üksikut rakendust ei saa hetkel varundada ja taastada on see ebasoovitav variant.

10 Kokkuvõte

Antud töö eesmärgiks oli luua PostgreSQL andmebaasisüsteemi põhine ja veebipõhine kiirprogrammeerimiskeskond, mis võimaldaks luua veebirakendusi. Loodavad rakendused peavad suhtlema andmebaasiga läbi avaliku andmebaasiliidese ning rakenduste väljanägemist ning käitumist peab olema võimalik juhtida metaandmetega.

Töö käigus selgitati, kuidas seda eesmärki saavutada. Selleks oli vaja uurida, kuidas toimub andmete pärimine välisest andmebaasist ning kust saab infot andmebaasiobjektide kohta. Seejärel pandi paika esmased nõuded süsteemile, kirjeldati kasutusjuhud kõrgformaadis ning loodi olemi-suhte diagrammid, mis kirjeldasid süsteemi poolt vajatavaid metaandmeid.

Töö tulemusena valmis veebipõhine süsteem, mille loomiseks kasutati PHP-d, PostgreSQL-i, AngularJS-i, Bootstrapi ning Slim Framework-i. Süsteem võimaldab arendajatel luua veebirakendusi ning valida rakenduse autentimismeetodite vahel. Rakendusse saab luua lehti, mis omakorda sisaldavad regioone, mida antud töös realiseeriti neli erinevat tüüpi. HTML-regioon võimaldab lisada lehele HTML-vormindusega teksti. Navigatsiooni regiooni abil saab kuvada lehel navigeerimiseks vajaliku menüüd. Raportite regioon kuvab kasutajale vaate põhjal tabeli, mida on võimalik jagada mitmele leheküljele. Vormi regioon võimaldab küsida kasutajalt infot ning saata seda välisesse andmebaasi kutsudes välja välises andmebaasis oleva funktsiooni.

Loodud süsteem on arenduse esimese iteratsiooni tulemus ning seetõttu realiseeriti vaid oluliseimad funktsionaalsused, et tagada süsteemi esmane kasutatavus reaalse tarkvara loomiseks. Seetõttu esitati töös nägemus funktsionaalsustest, mida järgnevate iteratsioonide korral võiks süsteemile juurde lisada. Süsteemi lähtekoodi avaldamine võiks aidata kaasa sellele, et tulevikus selle arendamine jätkub.

Kõige olulisem põhimõtteline erinevus loodud arendussüsteemi ja Oracle APEX-i vahel seisneb selles, et Oracle APEX võimaldab luua rakenduse nii otse baastabelitele kui ka andmebaasi avalikule liidesele. Samas loodav süsteem nõuab, et rakendus kasutaks andmebaasi avaliku liidese elemente (vaated, materialiseeritud vaated, funktsioonid). Otse baastabelitele selle abil rakendusi luua ei saa ning tarkvara loomisel ei pakuta valikutesse baastabeleid.

NuBuilder-ist, Xataface-st ja Mati Metsise magistritööst erineb loodud süsteem enim selle poolest, et loodud süsteemi puhul genereeritakse rakendused valmis andmebaasis, mitte rakenduskihis, kasutades võimalikult palju ära andmebaasi võimekust. Näiteks on töö tu-

lemusena valminud süsteemis allsüsteemide teenindamiseks loodud 103 andmebaasis tal-
letatud funktsiooni. Lisaks ei nõua võrdluses välja toodud süsteemid andmebaasi avaliku
liidese kasutamist.

Mati Metsise magistritöö tulemusena valminud süsteemis loodi iga rakenduse jaoks samas
andmebaasis uus skeem. Antud töö tulemusena valminud süsteem võimaldab aga luua ra-
kendusi samas andmebaasiserveris olevate andmebaaside põhjal ning kasutada funktsioo-
ne, vaateid ja materialiseeritud vaateid kõikidest skeemidest peale *information_schema* ja
pg_catalog-i. Lisaks võimaldavad antud töö tulemusena valminud süsteemi abil loodud
rakendused teostada andmebaasides andmemuudatusi. Samuti realiseeriti suurem hulk
erinevaid regiooni tüüpe.

Loodud tulemuse valideerimiseks reliseeriti õppejõu poolt ette antud rakenduse kasutus-
juhud, mis sarnanevad üliõpilastöodes esinevatele kasutusjuhtudele. Sellega sai süsteem
edukalt hakkama ning seetõttu võib väita, et töö eesmärk saavutati. Näidisrakendus asub
aadressil <http://apex.ttu.ee/pgapex/public/index.php/app/ruumid>
(kasutajanimi: *kask@ttu.ee*, parool: *test*). Arenduskeskkond asub aadressil <http://apex.ttu.ee/pgapex>.

Töö tulemus on avaldatud MIT litsentsi all ning on avalikult kättesaadav aadressilt
<https://github.com/raitraidma/pgapex>.

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 I Chef. [WWW] <https://www.chef.io/>. (07.03.2016).
- [8] Composer - Dependency Manager for PHP. [WWW] <https://getcomposer.org>. (06.05.2016).
- [9] DB-Engines Ranking - popularity ranking of database management systems. [WWW] <http://db-engines.com/en/ranking>. (09.05.2016).
- [10] Docker - Build, Ship, and Run Any App, Anywhere. [WWW] <https://www.docker.com/>. (07.03.2016).
- [11] GitHub. [WWW] <https://github.com/>. (21.04.2016).
- [12] What is free software? [WWW] <http://www.gnu.org/philosophy/free-sw.html>. (20.02.2016).
- [13] 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).
- [14] HHVM. [WWW] <http://hhvm.com/>. (07.03.2016).
- [15] Karma - Spectacular Test Runner for Javascript. [WWW] <https://karma-runner.github.io/0.13/index.html/>. (21.04.2016).

- [16] 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).
- [17] Licenses. [WWW] <http://choosealicense.com/licenses/>. (20.02.2016).
- [18] metaAndmed. [WWW] <http://eki.ee/dict/its/index.cgi?Q=metaandmed&F=M&C06=et&C10=1>. (07.05.2016).
- [19] Mati Metsis. Andmetega juhitud arendussüsteem PostgreSQL andmebaasirakenduste genereerimiseks. Master's thesis, Tallinna Tehnikaülikool, 2008.
- [20] Mockery. [WWW] <http://docs.mockery.io/en/latest/>. (21.04.2016).
- [21] MySQL. [WWW] <https://www.mysql.com/>. (08.03.2016).
- [22] nuBuilder. [WWW] <https://www.nubuilder.net>. (29.02.2016).
- [23] GitHub: nuSoftware/nuBuilderPro: Web Application Builder. [WWW] <https://github.com/nuSoftware/nuBuilderPro>. (29.02.2016).
- [24] The Open Source Definition. [WWW] <https://opensource.org/osd-annotated>. (20.02.2016).
- [25] Oracle Application Express. [WWW] <https://apex.oracle.com/en/>. (20.02.2016).
- [26] Oracle Database. [WWW] <https://www.oracle.com/database/index.html>. (20.02.2016).
- [27] Oracle's 2.5-Year Effort to Re-engineer APEX Bears Fruit – ADTmag. [WWW] <https://adtmag.com/blogs/watersworks/2015/05/oracle-apex-update.aspx>. (09.05.2016).
- [28] PHP Documentation - PDO. [WWW] <http://php.net/manual/en/intro.pdo.php>. (06.05.2016).
- [29] Pgapex - näidisrakendus. Kasutajanimi: kask@ttu.ee. Parool: test. [WWW] <http://apex.ttu.ee/pgapex/public/index.php/app/ruumid>. (09.05.2016).
- [30] PHP: Hypertext Preprocessor. [WWW] <http://php.net/>. (20.02.2016).

- [31] PHPUnit - The PHP Testing Framework. [WWW] <https://phpunit.de/>. (21.04.2016).
- [32] PostgreSQL. [WWW] <http://www.postgresql.org/>. (20.02.2016).
- [33] PostgreSQL: About. [WWW] <http://www.postgresql.org/about/>. (07.03.2016).
- [34] PostgreSQL: Documentation: 9.4: dblink. [WWW] <http://www.postgresql.org/docs/9.4/static/contrib-dblink-function.html>. (22.04.2016).
- [35] PostgreSQL: Documentation: 9.4: postgres_fdw. [WWW] http://www.postgresql.org/docs/9.4/static/postgres_fdw.html. (22.04.2016).
- [36] PostgreSQL: Documentation: 9.4: The Information Schema. [WWW] <http://www.postgresql.org/docs/9.4/static/information-schema.html>. (20.02.2016).
- [37] PostgreSQL: Documentation: 9.4: Rules and Privileges. [WWW] <http://www.postgresql.org/docs/9.4/static/rules-privileges.html>. (21.04.2016).
- [38] PostgreSQL: Documentation: 9.4: System Catalogs. [WWW] <http://www.postgresql.org/docs/9.4/static/catalogs.html>. (20.02.2016).
- [39] Private Interface. [WWW] <http://c2.com/cgi/wiki?PrivateInterface>. (09.05.2016).
- [40] Puppet Labs: IT Automation Software for System Administrators. [WWW] <https://puppetlabs.com/>. (07.03.2016).
- [41] SaltStack automation for CloudOps, ITops & DevOps at scale. [WWW] <https://saltstack.com/>. (07.03.2016).
- [42] Slim Framework. [WWW] <http://www.slimframework.com/>. (21.04.2016).
- [43] Travis CI - Test and Deploy Your Code with Confidence. [WWW] <https://travis-ci.org/>. (21.04.2016).
- [44] Vagrant. [WWW] <https://www.vagrantup.com/>. (06.03.2016).
- [45] Vagrant - Why Vagrant. [WWW] <https://www.vagrantup.com/docs/why-vagrant/>. (06.03.2016).

- [46] Vallaste - e-Teatmik: IT ja sidetehnika seletav sõnaraamat. [WWW] <http://vallaste.ee/>. (06.03.2016).
- [47] What is PHP? [WWW] <http://php.net/manual/en/intro-what-is.php>. (07.03.2016).
- [48] Xataface | The fastest way to build a front-end for your MySQL Database. [WWW] <http://xataface.com/>. (08.03.2016).
- [49] 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

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.

[36]

pg_catalog

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

<code>pg_namespace</code>	Säilitab infot nimeruumide kohta. Sealt on võimalik kätte saada andmebaasis olevad skeemid.
<code>pg_shadow</code>	Sisaldab infot kasutajate kohta, kellel on sisselogimisõigus. See tabel sisaldab paroole kujul 'md5' md5(parool kasutajanimi).
<code>pg_class</code>	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 <code>pg_views</code> ja <code>pg_matviews</code> , 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.
<code>pg_attribute</code>	Sisaldab infot veergude kohta.
<code>pg_type</code>	Sisaldab infot andmetüüpide kohta. Siin tabelis on esindatud nii põhiandmetüübid, kasutaja loodud tüübid, domeenid ja komposiitandmetüübid, mis luuakse iga andmebaasis oleva tabeli jaoks.
<code>pg_proc</code>	Sisaldab infot funktsioonide kohta.

[38]

Lisa 2 - Free Software

Free Software (Vaba tarkvara) tähendab, et kasutajatel on vabadus tarkvara käivitada, 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: käivitada 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 tasuta eest või tasuta - selleks ei pea kelleltki luba küsima. Küll aga peab jagatav koopia sisaldama nii lähtekoodi kui ka käivitavat 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. [12]

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 konkreetset 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 - Programmile 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.

[24]

Lisa 4 - Populaarsemate litsentside võrdlus

Tabel 14. Populaarsemate litsentside võrdlus.

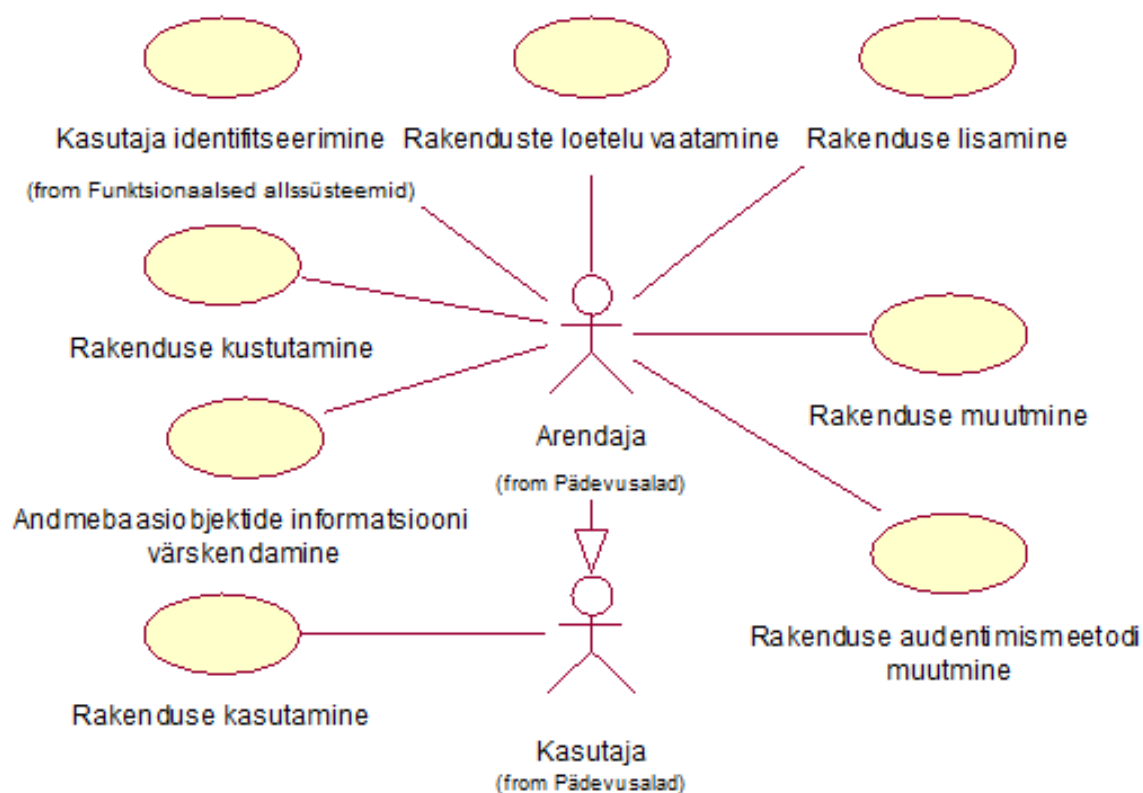
	Nõutud	Lubatud	Keelatud
MIT	Litsents ja copyright märke	Kaubanduslik kasutamine Jagamine Muutmine Privaatne kasutamine	Võtta vastutusele
Apache License 2.0	Litsents ja copyright märke Teavitus muudatustest	Kaubanduslik kasutamine Jagamine Muutmine Patendi kasutamine Privaatne kasutamine	Võtta vastutusele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

[17]

Lisa 5 - Kasutusjuhtude diagrammid

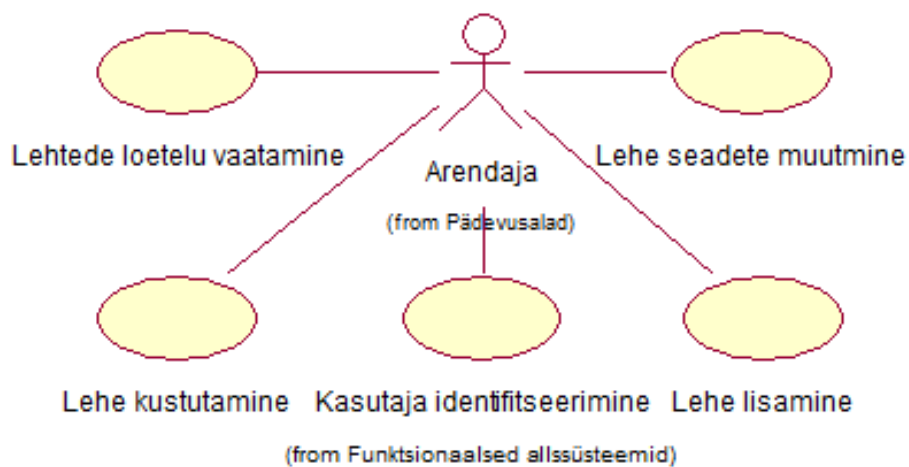
Alljärgnevalt on esitatud kasutusjuhtude diagrammid allsüsteemide kaupa.

Rakenduste funktsionaalne allsüsteem



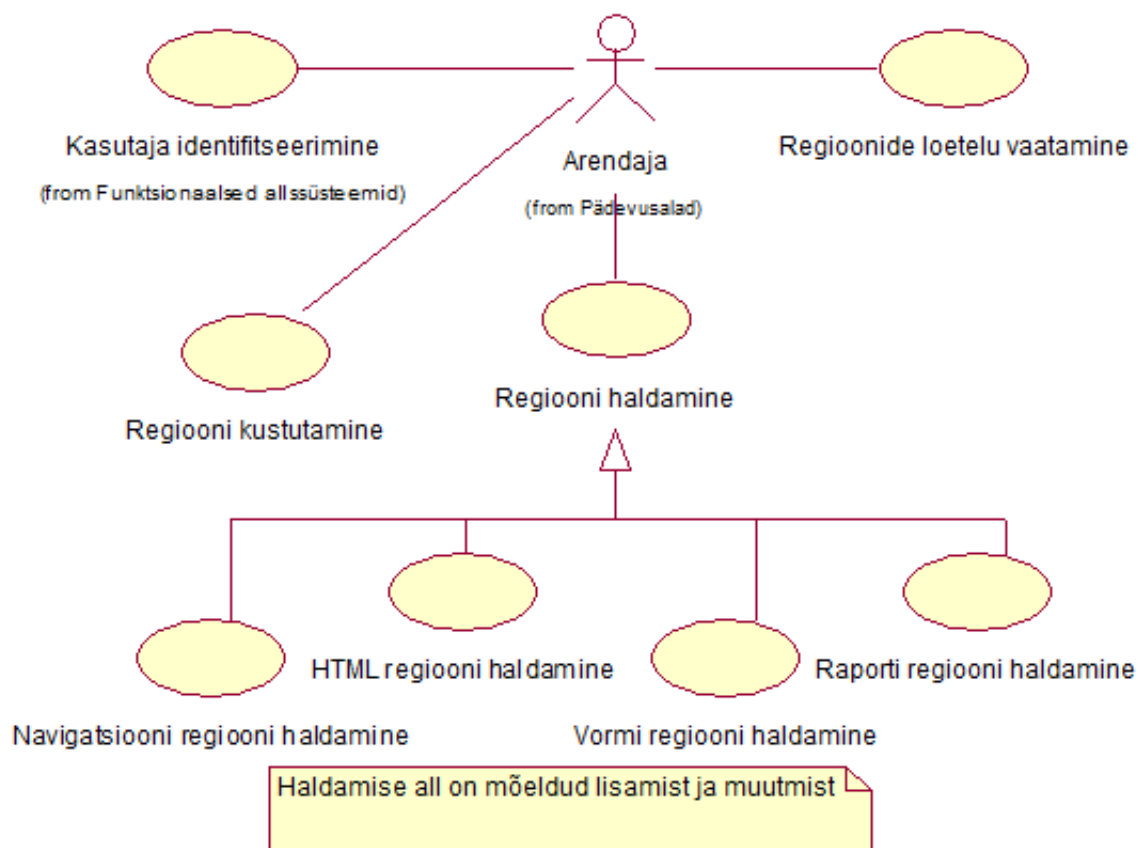
Joonis 23. Rakenduste funktsionaalse allsüsteemi kasutusjuhtude diagramm.

Lehtede funktsionaalne allsüsteem



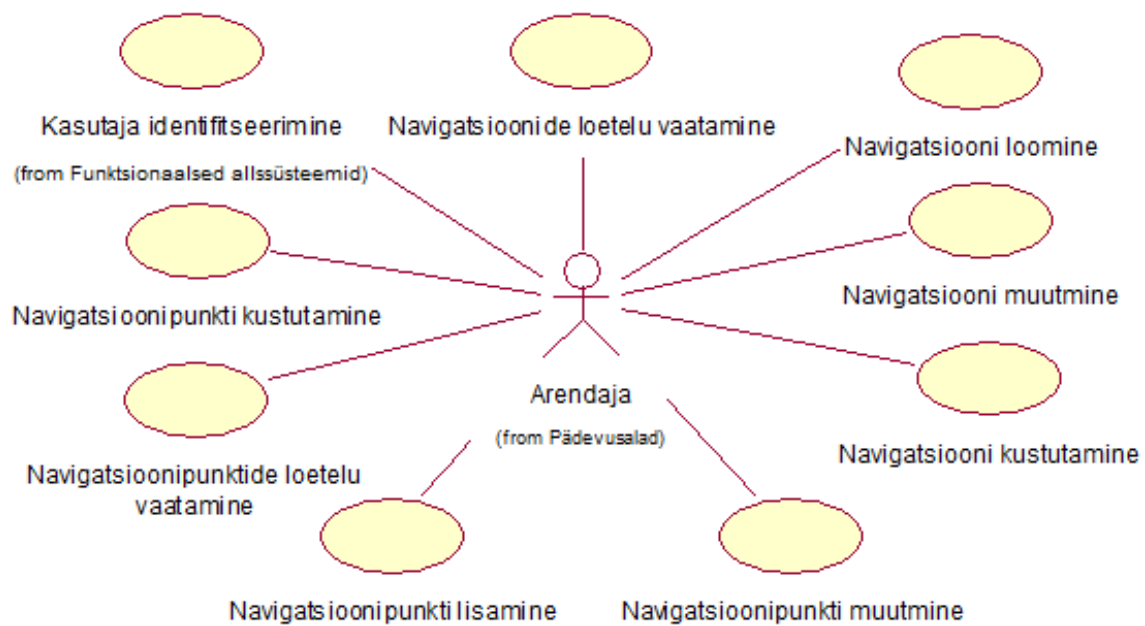
Joonis 24. Lehtede funktsionaalse allsüsteemi kasutusjuhtude diagramm.

Regioonide funktsionaalne allsüsteem



Joonis 25. Regioonide funktsionaalse allsüsteemi kasutusjuhtude diagramm.

Navigatsioonide funktsionaalne allsüsteem



Joonis 26. Navigatsioonide funktsionaalse allsüsteemi kasutusjuhtude diagramm.

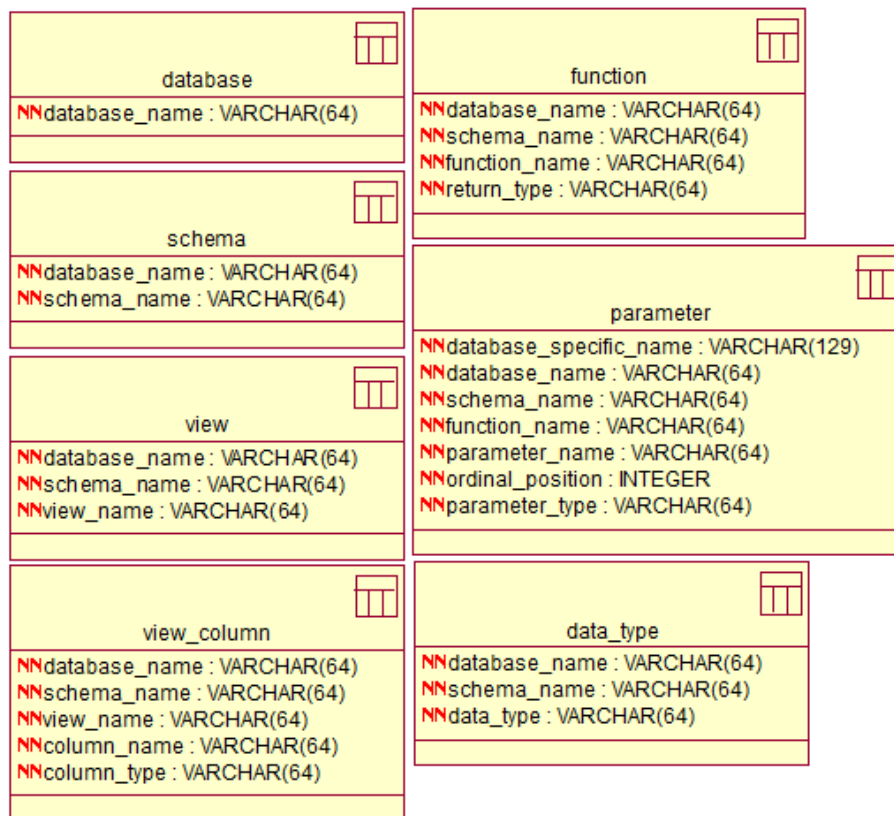
Lisa 6 - Andmebaasi diagrammid

Järgnevalt on välja toodud andmebaasidiagrammid registrite kaupa.

Andmebaasiobjektide register

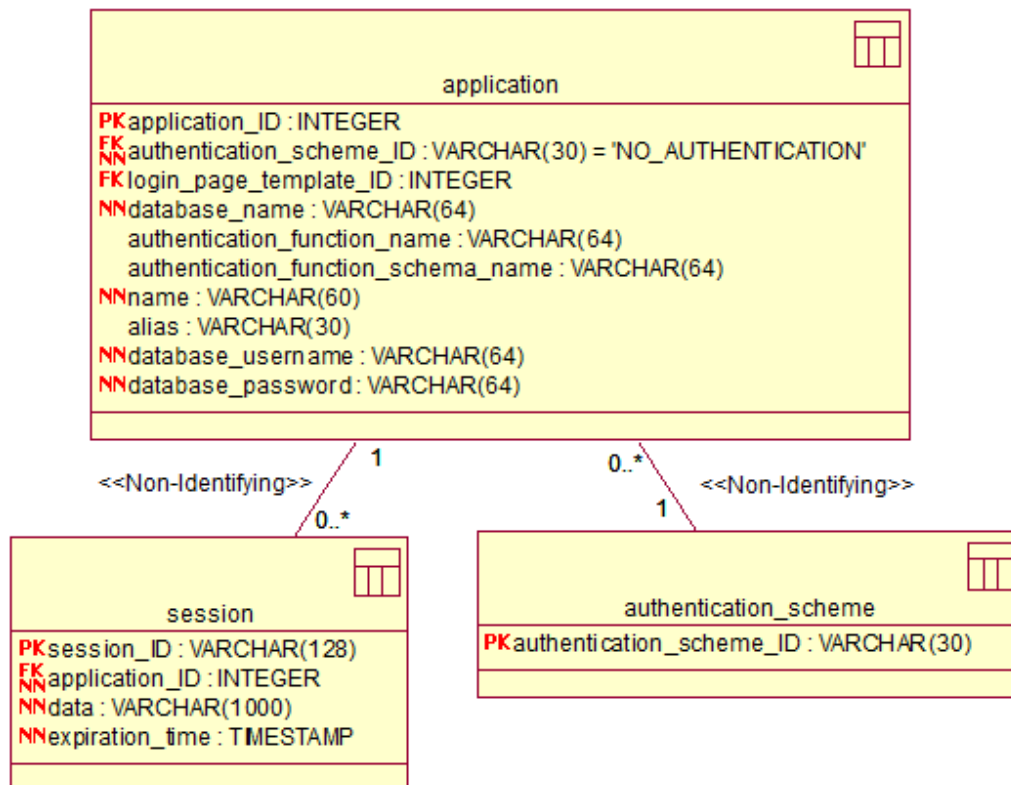
Andmebaasiobjekte kirjeldavad tabelid realiseeriti materialiseeritud vaadetenä. Kuna vaadetes olevaid veerge ei saa kasutada välisvõtmekitsendustes, siis sisaldavad alltoodud vaated rohkem infot kui olemi-suhte diagrammidel näha. Sedasi tagatakse iga rea unikaalsus.

Arenduskeskkonna kasutajale on loodud nupp, millele vajutades tema loodava rakenduse andmebaasi kirjeldust värskendatakse.



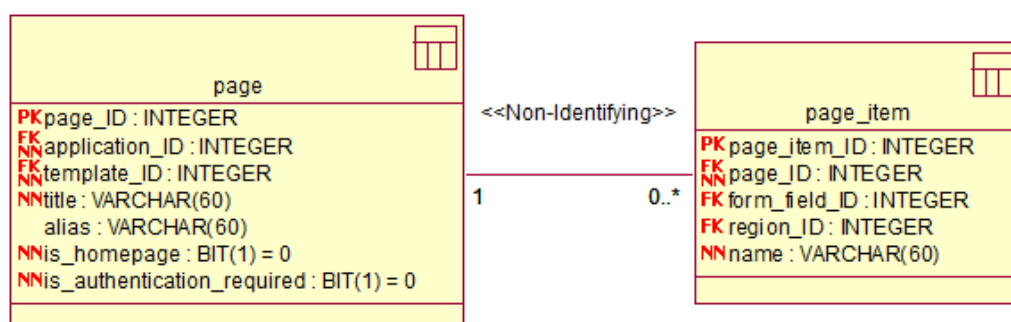
Joonis 27. Andmebaasiobjektide registri andmebaasi diagramm.

Rakenduste register



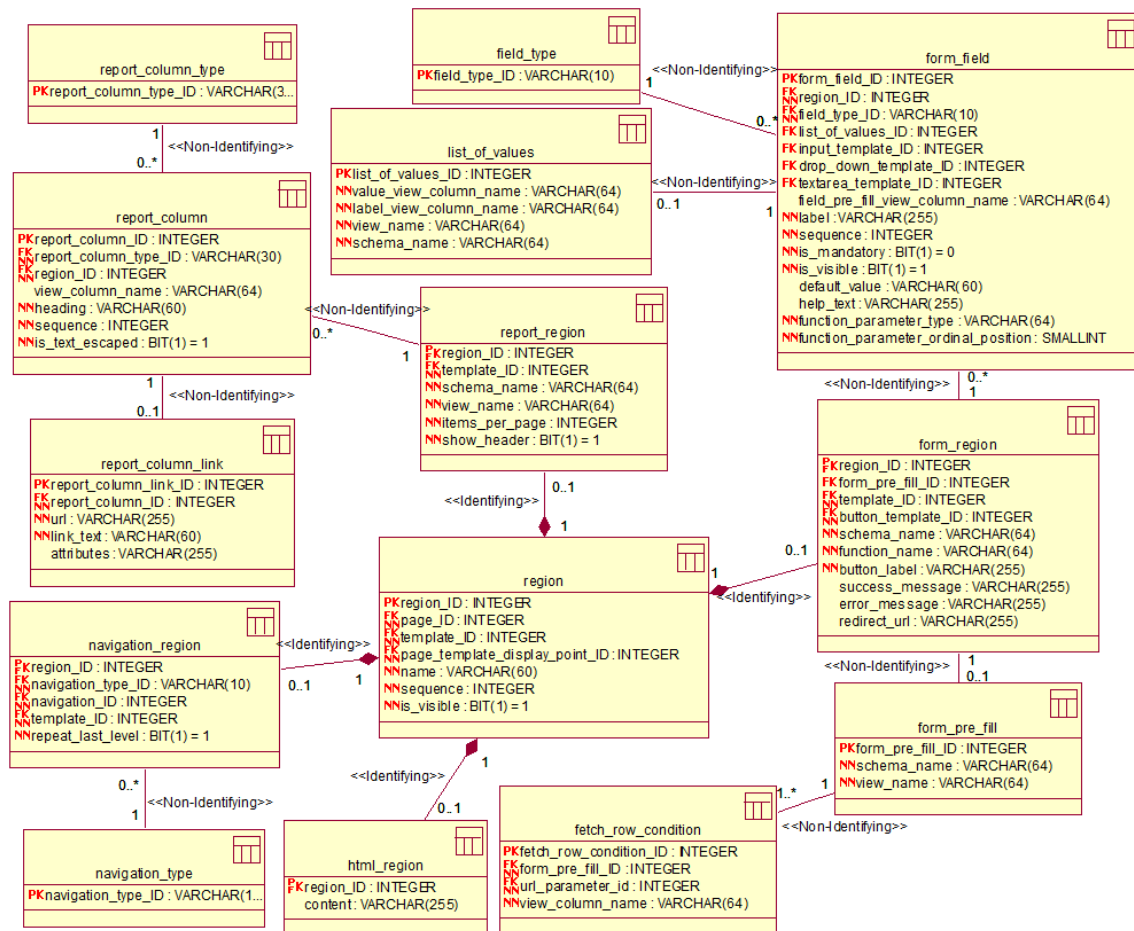
Joonis 28. Rakenduste registri andmebaasi diagramm.

Lehtede register



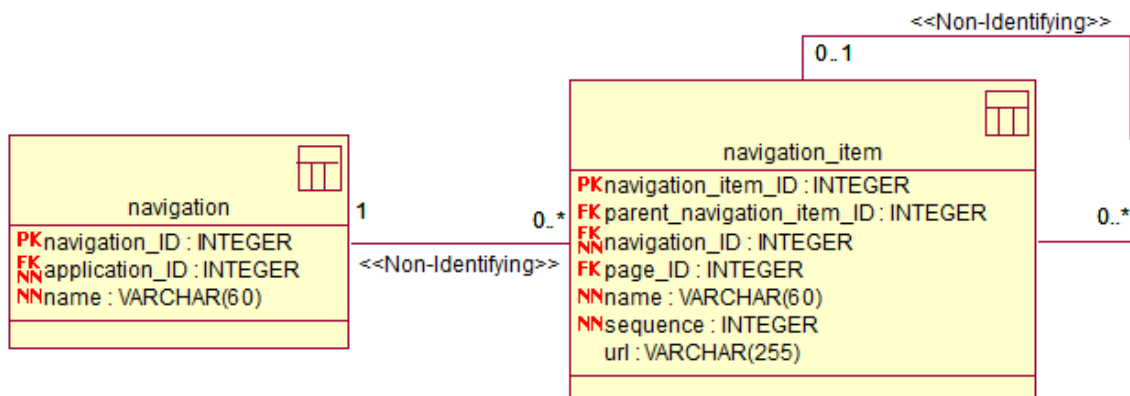
Joonis 29. Lehtede registri andmebaasi diagramm.

Regionide register



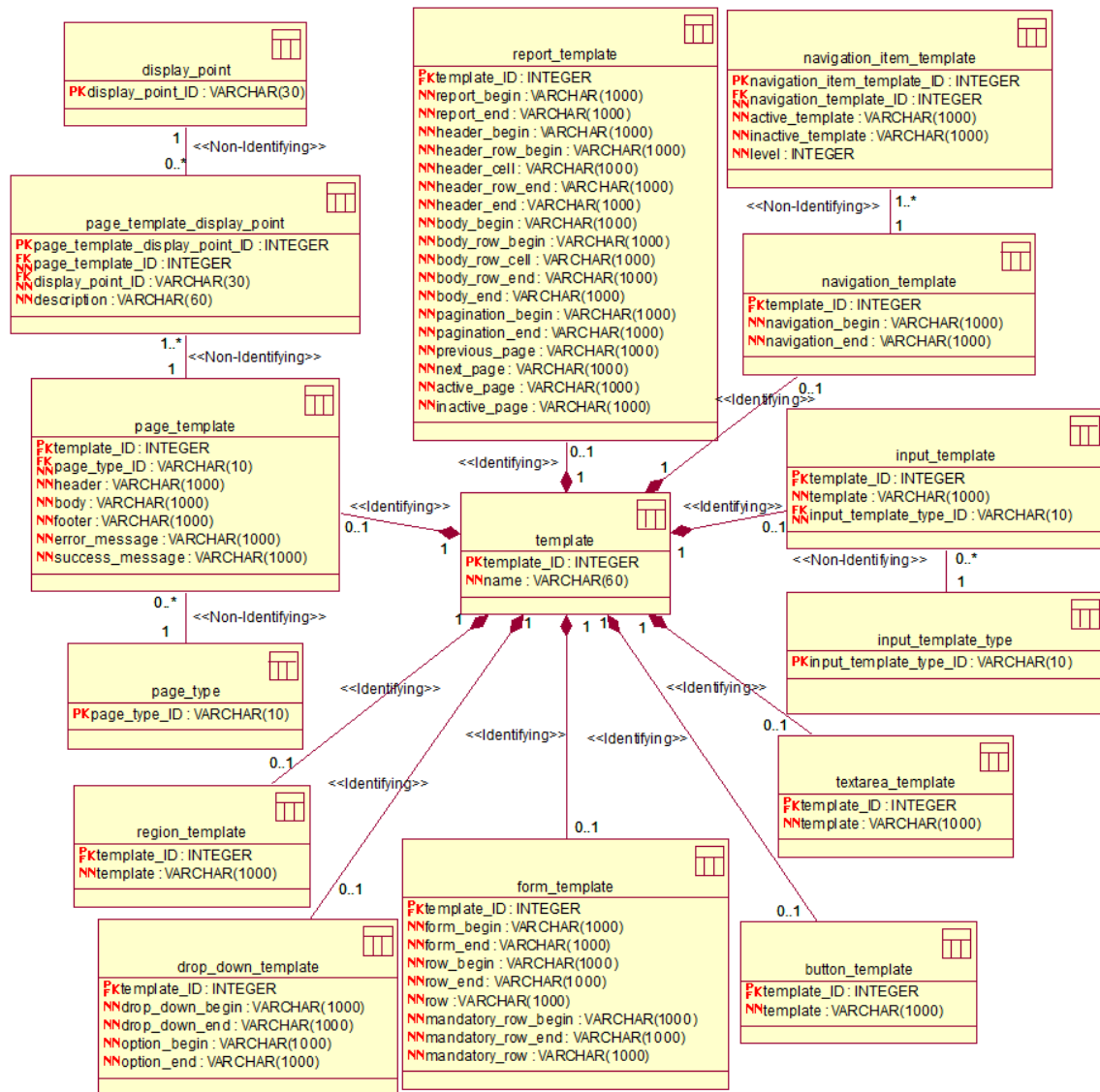
Joonis 30. Regionide registri andmebaasi diagramm.

Navigatsioonide register



Joonis 31. Navigatsioonide registri andmebaasi diagramm.

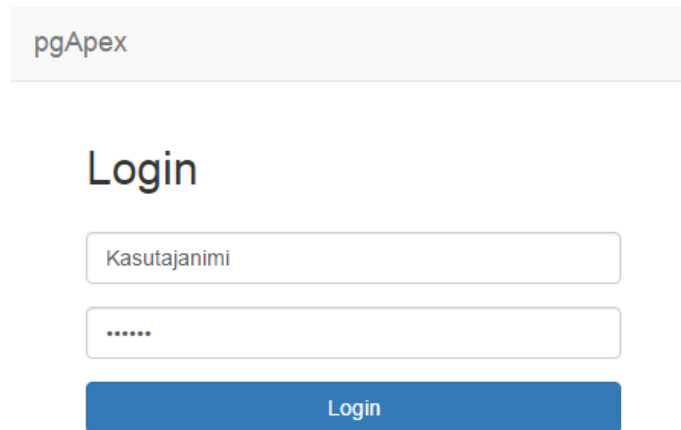
Mallide register



Joonis 32. Mallide registri andmebaasi diagramm.

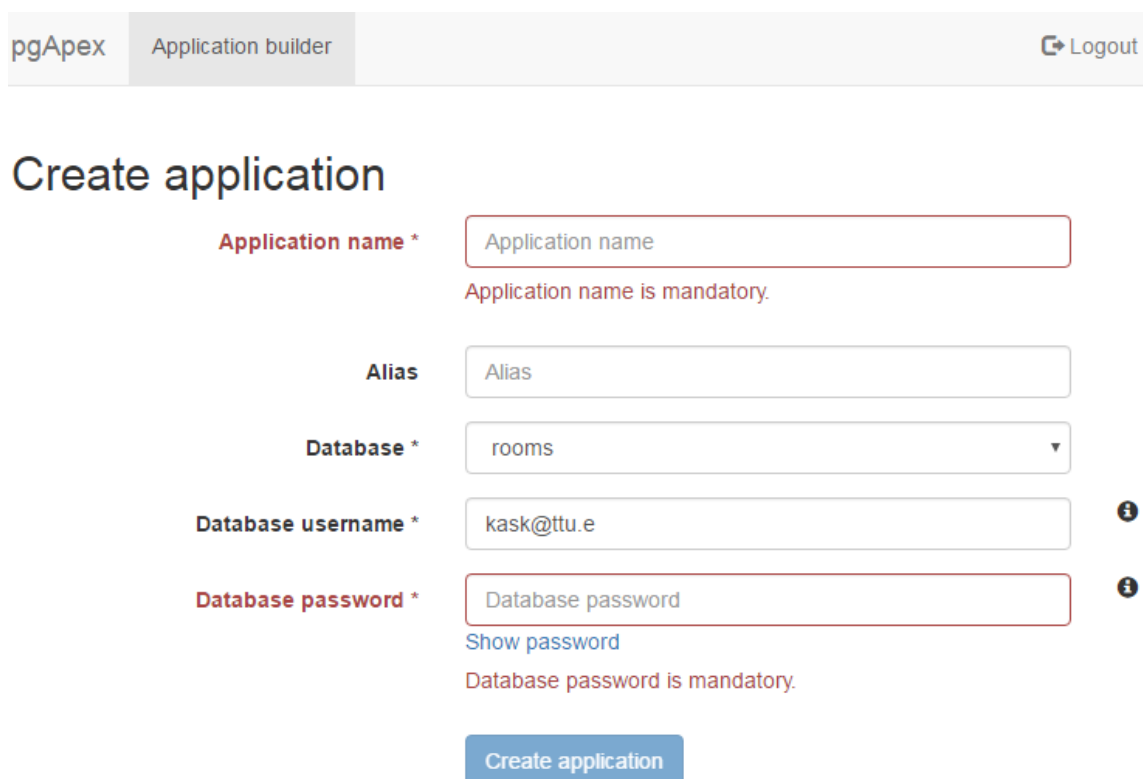
Lisa 7 - Ekraanipildid loodud arenduskeskkonnast

Järgnevalt on välja toodud väike osa süsteemi kuvadest, et anda lugejale aimdus sellest, milline süsteem välja näeb. Selleks, et ekraanipildid lehtedele ära mahuksid, on mõningatel pildidel vormielementide laiusi ning nende vahelisi kaugusi vähendatud.



The screenshot shows the login interface of the pgApex application. At the top, the 'pgApex' logo is displayed in a light gray box. Below the logo, the word 'Login' is centered in a large, dark font. Underneath, there are two input fields: the first is labeled 'Kasutajanimi' (Username) and the second is a password field with masked characters (dots). A blue 'Login' button is positioned below the password field.

Joonis 33. Loodud süsteem: Autentimine.



The screenshot displays the 'Create application' form within the pgApex application builder. The top navigation bar includes the 'pgApex' logo, the text 'Application builder', and a 'Logout' link with an external icon. The main heading 'Create application' is prominently displayed. The form contains several fields: 'Application name' (with a red border and a mandatory error message), 'Alias', 'Database' (a dropdown menu currently showing 'rooms'), 'Database username' (containing 'kask@ttu.e'), and 'Database password' (with a red border, a 'Show password' link, and a mandatory error message). Information icons are visible next to the username and password fields. A blue 'Create application' button is located at the bottom of the form.

Joonis 34. Loodud süsteem: Rakenduse loomine.

Selleks et raportis loodud link suunaks näiteks samas rakenduses lehele, mille alias on “muuda”, ning annaks URL-parameetrile *p_room_code_old* kaasa vaate veerus *room_code* oleva väärtuse, tuleb URL-aadressi välja lisada järgnev tekst:
&APPLICATION_ROOT&/app/&APPLICATION_ID&/muuda?p_room_code_old=%room_code%.

Edit report region

Name *

Muudetavate ruumide nimekirj

Sequence *

1

Region template *

Region template

Is visible

☒

Report template *

Report template

View *

public.modifiable_rooms

Show header

☒

Items per page *

20

Pagination query parameter *

p

Columns

Heading

Sequence

Is text escaped

Column

Link

Mine muutma	1	<input checked="" type="checkbox"/>	&APPLICATION_ID	Muuda	class="btn"	
Ruumi kood	2	<input checked="" type="checkbox"/>	room_code			
Ruumi nimi	3	<input checked="" type="checkbox"/>	room_name			

Edit report region

Joonis 35. Loodud süsteem: Raporti regiooni muutmine.

Edit form region

Name *	<input type="text" value="Muuda ruumi"/>
Sequence *	<input type="text" value="1"/>
Region template *	<input type="text" value="Region template"/>
Is visible	<input checked="" type="checkbox"/>
Form template *	<input type="text" value="Form template"/>
Button template *	<input type="text" value="Button template"/>
Button label *	<input type="text" value="Muuda"/>
Success message	<input type="text" value="Muutmine õnnestus"/>
Error message	<input type="text" value="Error message"/>
Redirect url	<input type="text" value="Redirect url"/>
Function *	<input type="text" value="functions.f_change_a_room(p_room_code_old int4, p_ro"/>
Pre fill form	<input checked="" type="checkbox"/>



Pre fill form First returned value is used

At least one condition must be added.

View *	<input type="text" value="public.modifiable_rooms"/>
room_name	<input type="text"/>
room_code	<input type="text" value="p_room_code_old"/>

Function parameters

p_room_code_old <small>int4</small>	p_roon	Ruumi vana kood	5	<input checked="" type="checkbox"/> Is mandatory	<input type="checkbox"/> Is visible	Default value	Help text
<input type="text" value="Text"/>	<input type="text" value="Text input template"/>					<input type="text" value="room_code"/>	
p_room_code_new <small>int4</small>	p_roon	Ruumi uus kood	6	<input checked="" type="checkbox"/> Is mandatory	<input checked="" type="checkbox"/> Is visible	Default value	Help text
<input type="text" value="Text"/>	<input type="text" value="Text input template"/>					-- Pre fill col	

Edit form region

Joonis 36. Loodud süsteem: Vormi regiooni muutmine.

Lisa 8 - Andmebaasiobjektide info küsimine PostgreSQL andmebaasisüsteemis

Järgnevalt on esitatud mõned päringud, millega küsitakse infot andmebaasis olevate andmebaasiobjektide kohta.

Kuna *information_schema* ei sisalda infot materialiseeritud vaadete kohta, siis küsitakse seda infot *pg_catalog*-st (vt Joonis 37).

```
SELECT n.nspname AS view_schema, c.relname AS view_name
FROM pg_catalog.pg_class c
LEFT JOIN pg_catalog.pg_namespace n ON n.oid = c.
    relnamespace
WHERE c.relkind IN ('v', 'm')
    AND n.nspname NOT IN ('information_schema', 'pg_catalog')
    AND n.nspname NOT LIKE 'pg_toast%'
    AND n.nspname NOT LIKE 'pg_temp%'
```

Joonis 37. SQL: Andmebaasis vastavas skeemis olevate vaadete ja materialiseeritud vaadete küsimine.

Kuna ühes skeemis võib olla mitu sama nimega funktsiooni, siis nende unikaalsuse tagamiseks on võetud kasutusele väli *specific_name* (vt Joonis 38).

```
SELECT r.specific_name, r.routine_schema, r.routine_name, p.
    parameter_name, p.ordinal_position, p.udt_name
FROM information_schema.routines r
JOIN information_schema.parameters p ON r.specific_name = p.
    specific_name
WHERE r.routine_type = 'FUNCTION'
    AND r.routine_schema NOT IN ('pg_catalog', '
    information_schema')
```

Joonis 38. SQL: Andmebaasis vastavas skeemis oleva funktsiooni parameetrite info küsimine.