TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Dmitri Bizjulin 155173IAPB

PGAPEX ARENDUSKESKKONNALE CRUD VEEBIRAKENDUSTE LOOMISE VÕIMEKUSE ANDMINE

Bakalaureusetöö

Juhendaja: Erki Eessaar
PhD

Tallinn 2020

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: Dmitri Bizjulin

12.05.2020

Annotatsioon

Töö eesmärgiks oli anda PostgreSQL andmebaasisüsteemi põhise metaandmetega juhitavate veebirakenduste kiirprogrammeerimise vahendile pgApex võimekus luua selle abil täielikult CRUD funktsionaalsust toetavaid andmebaasirakendusi. CRUD funktsionaalsus tähendab andmete lisamise, lugemise, uuendamise ja kustutamise võimalust.

Töö käigus tehti taustauuring, milles pgApex'iga võrreldi sarnast eesmärgi täitvaid arenduskeskkondi. Samuti uuriti eelmiste selle süsteemi arendamisega tegelenud lõputööde põhjal pgApex'i olemasolevat funktsionaalsust. Seejärel valiti realiseeritav funktsionaalsus ja koostati väljalaske plaan. Lisatava funktsionaaluse hulka kuulusid *näiteks* raport koos nuppudega algatamaks olemi andmete lisamist ja muutmist, alamvormid ning ridade optimistliku lukustamise võimaldamine. Samuti otsustati täiendada vormiväljade häälestamist ning lisada vormidesse uued väljatüübid: kalendrivalik ja liitboks.

Töö tulemusena valmis pgApex'i kolmas versioon, kus on realiseeritud funktsionaalsus, mis võimaldab luua CRUD andmebaasirakendusi. Samuti tehti töö käigus pgApex'is kasutajakogemuse parandamiseks väiksemaid parandusi ja täiendusi.

Tehtud töö valideerimiseks pgApex'i uue versiooni abil loodi rakendus õppeaine „Andmebaasid II" projekti „Golfirajad" andmebaasi põhjal. Lõputöö autor oli selle projekti üks autoritest. Rakendus on kättesaadav aadressilt: http://apex.ttu.ee/pgapex3/public/index.php/app/10/31
(kasutajanimi on *lucile.burgess@frolix.net* ning parool on *laborum*).

Nagu ka eelnevate versioonide puhul on pgApex'i kolmanda versiooni lähtekood avatud (MIT litsents) ning asub järgnevas GitHub'i salves: https://github.com/dmibiz/pgapex3

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 70 leheküljel, 10 peatükki, 52 joonist, 6 tabelit.

Abstract

Giving the Ability to Create CRUD Web Applications With pgApex Development Environment

The aim of the thesis was to design and implement fully the possibility to create CRUD database applications with a PostgreSQL-based rapid development environment for the metadata-driven web applications. The name of the environment is pgApex. CRUD functionality of a database application means that one can create, read, modify, and delete data by using it.

In this thesis the new functionality was developed. Prior to that the author conducted a background research of other similar programming environments as well as investigated the existing functionality of pgApex.

During the background research the author searched for programs that have the similar purpose to pgApex, tried these if possible, and compared their functionality to pgApex (including the pgApex's new functionality that was developed as a result of this thesis). The main conclusions of the analysis is that many investigated enviroments do not offer the same functionality as pgApex, i.e., have shortcomings. For instance, in some of these systems one cannot use user-defined functions in the database or optimistically lock table rows. Moreover, in case of the environments one has to publish files (after each modification) instead of modifying applications' metadata in the database.

After that the release planning was carried out. The thesis supervisor (in the role of *product owner*) proposed features that should be implemented to achieve full CRUD functionality in applications being created with pgApex, which include, *for instance*, new „Report and form" region, the ability to add subforms to a form region, and the ability to implement optimistic locking of rows. In addition, it was decided to improve setting up the appearance of form fields and to implement new field types: calendar and combo box.

As a result of the thesis, the third version of pgApex was developed, which includes the functionality for creating applications with full CRUD abilities. The possibility to implement optimistic locking is based on PostgreSQL built-in feature according to which each base table has a hidden and system-populated column (*xmin*) for the row version identifier. Moreover, some additions and improvements to existing functionality were made in order to improve the user experience.

In order to validate the result, an application was created by using the third version of pgApex. The application is based on the project that was made with author's participation in the „Databases II" course. The application is available at http://apex.ttu.ee/pgapex3/public/index.php/app/10/31
(username is *lucile.burgess@frolix.net* and password is *laborum*).

The third version of pgApex is open source with MIT licence (as the previous versions) and is available on GitHub: https://github.com/dmibiz/pgapex3

The thesis is in Estonian and contains 70 pages of text, 10 chapters, 52 figures, 6 tables.

Lühendite ja mõistete sõnastik

CRUD	Loomine *(Create)*, lugemine *(Read)*, uuendamine *(Update)*, kustutamine *(Delete)*. Põhifunktsioonid andmebaasirakenduses.
CSS	*Cascading Style Sheets*. Keel veebilehtede kujundamiseks. Kasutatakse koos HTML'iga.
DOM	*Document Object Model.* Liides, mis kirjeldab XML või HTML dokumenti puustruktuurina ja selle sõlmed on objektid, mis kirjeldavad dokumendi osasid [1].
HTML	*HyperText Markup Language.* Keel veebilehtede kirjeldamiseks.
IDE	*Integrated development environment*. Tarkvara, mis pakub programmeerijatele tööriistu tarkvara arendamiseks [2].
JSON	*Javascript Object Notation.* Andmete kirjeldamise formaat, mis kasutab Javascript programmeerimiskeele objekti süntaksi.
PDO	*PHP Data Object*. Klass PHP keeles, mille abil saab andmebaasiga suhelda.
SPA	*Single-page application.* Rakendus, kus kogu sisu jaoks kasutatakse ühte HTML lehte ja sisu uuendatakse dünaamiliselt.
SQL	*Structured Query Language.* Relatsioonilisel mudelil põhinev keel SQL- andmebaaside andmete ning nende andmebaaside struktuuri, käitumise ning kasutamise haldamiseks [3].
WSIWYG	*What you see is what you get*. Süsteem, kus teksti ja graafikat kuvatakse täpselt nii, nagu see näeks välja prinditult [4].

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Edukate ja täpsete otsuste aluseks on võimalikult värsked, täpsed ning olukorrale sobiva detailsusastmega andmed. Seega võib öelda, et kõik infosüsteemid (nii looduslikud kui tehislikud) on andmetöötluse süsteemid. Andmeid säilitatakse mingis keskkonnas, milleks tänapäeval on inimmaailmas valdavalt arvutisüsteemid. Selleks, et andmeid mugavalt hoida ja hallata, kasutatakse muuhulgas spetsiaalse tarkvara – andmebaasisüsteemide – abil loodud andmebaase. Sellistes andmebaasides saab andmeid esitada erinevatest abstraktsioonidest lähtuvalt. Neid abstraktsioone tuntakse andmemudelitena. Abstraktsioonid on lihtsustused, mis aitavad kihiliselt ülesehitatud süsteemis kõrgemate kihtide kasutajate eest peita süsteemi madalamate kihtide ehitusplokke, operatsioone ja nende probleeme (näiteks, kuidas on andmed kettale salvestatud). Tänapäeval on laialt levinud andmebaasisüsteemid, mis põhinevad SQL andmebaasikeele aluseks oleval andmemudelil [5]. SQL andmebaasikeel on relatsioonilise andmemudeli üks võimalik realisatsioon [6]. SQL-andmebaaside põhiline ehitusplokk on tabelid. SQL on mahukas, keeruline ja võimalusterohke [7]. Infotehnoloogilistes andmebaaside, sh SQL-andmebaaside, kasutamiseks on vaja programme – andmebaasirakendusi – mille kaudu lõppkasutajad saavad lugeda ja muuta andmebaasides olevaid andmeid. Sellised rakendused peavad ka pakkuma mugavat kasutajaliidest. Tänapäeval luuakse paljud sellised rakendused professionaalsete programmeerijate poolt, kasutades keeli ja vahendeid, mis nõuavad palju oskuseid ning kus ka arendus võtab aega. Kuna andmeid ja andmekogusid on aina rohkem ning aina rohkem on lõppkasutajaid, kellel oleks andmebaasirakendustest palju abi, siis on tarvis leida ja realiseerida viise, kuidas selliseid rakendusi lihtsamalt ja kiiremini luua. Üheks võimaluseks arenduse „massidesse“ viimiseks, on kasutada tarkvara, mida nimetatakse kiirprogrammeerimiskeskkondadeks. Andmebaasirakenduste kiirprogrammeerimise keskkonnad võimaldavad luua andmebaasidel põhinevaid rakendusi ilma programmikoodi kirjutamata, või vähemalt seda vähe tehes. Tegemist on tarkvaraga, mis lähtub ilma kodeerimise vajaduseta arendusplatvormi (*no code development platform*) ideaalist [8]. Sellist tüüpi keskkonna *üheks võimalikuks* tööpõhimõtteks on, et rakenduse ülesehitus, käitumine ja väljanägemine kirjeldatakse andmetena,

salvestatakse andmebaasis ning selles muudatuste tegemiseks peab muutma andmeid andmebaasis. Selline keskkond jälgib avatud-suletud printsiibi [9] põhimõtet, mille kohaselt peab tarkvara olema avatud laiendamiseks, kuid suletud muutmiseks. Sellises keskkonnas loodud andmebaasirakendust saab laiendada (muutes rakenduse kirjeldust), kuid selleks ei ole vaja muuta lähtekoodi.

Üks sellise arenduskeskkonna näide on Oracle APEX [10]. See on kiirprogrammeerimise keskkond, mille abil saab luua rakendusi Oracle andmebaasisüsteemis loodud andmebaaside põhjal. Selle abil loodud rakenduse saab panna kasutama ka teistes andmebaasisüsteemides (näiteks PostgreSQL) loodud andmebaase, kuid selleks on vaja teha palju lisatööd [11].

1.1 Taust ja probleem

pgApex on Oracle APEX'i [10] eeskujul loodud tarkvara, mille abil saab olemasoleva PostgreSQL andmebaasi põhjal luua ja täiendada veebirakendusi kiiresti ja ilma programmeerimisoskusteta. Rakenduse loomiseks läheb vaja andmebaasi avalikku liidest, mis moodustub funktsioonidest ning vaadetest ning nende loomiseks on vaja kirjutada SQL koodi. Kuid rakenduse enda tegemiseks ei ole vaja lähtekoodi kirjutada.

Erinevaid vahendeid, mille abil saab lihtsalt (vähese programmeerimisega või programmeerimiseta) luua andmebaasirakendusi on peale Oracle APEX'it teisigi nagu näiteks: Openxava [12], Fadex [13], Phprad [14], PostgreSQL PHP Generator [15], Radzen [16], 4WS Platform [17]. Sellest võib järeldada, et selliste arenduskeskkondade loomine on aktuaalne ülesanne. Erinevate vahendite sisemine arhitektuur (ülesehitus) on erinev ning mõnede nende vahendite uurimise ja võrdlemisega tegeleb peatükk 3.

pgApex vahendi esimese versiooni lõi magistritöö tulemusena Rait Raidma [18] ning selle täiendamisest Nikolai Kopa bakalaureusetöö [19] tulemusena tekkis teine versioon.

CRUD rakendus võimaldab lisada (C), lugeda (R), uuendada (U) ja kustutada (D) olemite ja nende seoste andmeid. pgApexi enne käesoleva töö algust realiseeritud funktsionaalsus ei võimaldanud loodavates veebirakendustes realiseerida piisavalt hästi C ja U funktsionaalsust. *Näiteks* ei olnud võimalik luua andmete sisestamise ja muutmise vormi, milles saab andmeid sisestada liitbokside, kalendrivalikute, märketuutude ja tekstiakende abil ning kus salvestamise nupu vajutamine käivitaks

vastavalt kas andmete lisamise või muutmise funktsiooni. Ei olnud võimalik olemite aruandest liikuda selle olemi andmete muutmise vormile, milles on automaatselt avatud selle olemi andmed. Vastava funktsionaalsuse alged olid pgApexis realiseeritud, kuid seda oli vaja edasiarendada. Veel üks oluline puudus oli see, et polnud võimalik realiseerida ridade optimistlikku lukustamist, et arvestada andmete samaaegse muutmisega erinevates andmebaasi kasutamise sessioonides ei seanssides. Kui rakendus lukustamist ei realiseeri võivad tagajärjed (andmete kadu, mainekadu, majanduslik kahju) olla tõsised [20] [21] [22].

1.2 Ülesandepüstitus

Eesmärgiks on anda pgApexile täielik CRUD andmebaasirakenduste loomise võimekus. Oodatav tulemus on pgApex'i kolmas versioon, kus on realiseeritud uus funktsionaalsus.

1.3 Metoodika

Eesmärgi saavutamiseks kasutatakse disainiteaduse metoodikat [23] mille kohaselt tuleb kasutajate ja tooteomaniku (antud juhul on tooteomanikuks juhendaja) soovide ning valdkonna teadmiste alusel luua uus tehniline tehis (antud juhul täienenud mudelid ning rakendus) ning seejärel seda valideerida. Lisaks tehistele tekib selle käigus ka uut teadmist kiirprogrammeerimise keskkondade ja nende loomise kohta.

Tarkvara arendatakse iteratiivselt. Arenduse alguses koostatakse iteratsioonide plaan, mida täpsustatakse iga iteratsiooni lõppedes. Väljalaske plaani koostamisel lähtutakse Normani [24] soovitustest. Iga iteratsiooni lõpus näidatakse tulemus tooteomanikule (juhendajale) ette. Arenduse käigus arvestatakse tooteomanikult tulevate nõuetega. Järgitakse paindmetoodikate parimaid praktikaid nagu suletud akna (*closed window*) [25] ning ajakarbi (*timebox*) meetod [26].

Tulemuste valideerimiseks luuakse uus pgApex'i rakendus, kus on realiseeritud kogu CRUD rakenduselt oodatav funktsionaalsus. Selleks kasutatakse autori osalusel „Andmebaasid II" aines loodud (golfiklubi radade arvestuse) andmebaasi [27], millele realiseeritakse radade haldurile ettenähtud funktsionaalsust pakkuv rakendus. Selleks luuakse muuhulgas rakenduse tegemise eelduseks olevad vaated ning funktsioonid.

1.4 Ülevaade tööst

Kõigepealt uuritakse pgApexi tööpõhimõtet ja olemasolevat funktsionaalsust ning kirjeldatakse seda lühidalt peatükis 2. Peatükis 3 uuritakse pgApexi'ga sarnast eesmärki täivaid programme ning võrreldakse nende poolt pakutud funktsionaalsust pgApex'i funktsionaalsusega. Peatükis 4 planeeritakse väljalase (*release planning*), pannakse paika ülesannete prioriteedid ja hinnatakse nende keerukus. Peatükis 5 kirjeldatakse süsteemis tehtavaid muudatusi. Peale seda viiakse läbi süsteemi iteratiivne arendus, kus igas iteratsioonis analüüsitakse, realiseeritase ja testitakse mingit süsteemi osa. Arenduse tulemused esitatakse peatükis 6. Väljalaske lõppedes tutvustatakse statistikat muudatuste kohta andmebaasis ja lähtekoodis (peatükk 0) ja valideeritakse tulemust näiterakenduse loomise abil (peatükk 8). Peatükis 9 esitatakse ideid süsteemi edasiseks arendamiseks. Lõpuks tehakse tehtud tööst kokkuvõte.

2 pgApexi olemasoleva funktsionaalsuse uurimine

Programmi uurimiseks ning arendamiseks tehakse selle teise versiooni git salvest koopia (*fork*), mida kasutatakse ka arendamiseks ja kuhu lõpuks jõuab pgApexi kolmas versioon. See koopia kloonitakse autori arvutisse. Samuti tehakse koopia pgApexi poolt kasutatavast andmebaasist (teise versiooni puhul *pgapex2*) ning hakatakse seda uurimis- ja arendustööks kasutama. Uue andmebaasi nimi on *pgapex3* ja seda hakkab kasutama ka pgApexi uus (kolmas) versioon. Niimoodi on uurimine ja arendus isoleeritud ja saab turvaliselt teha muudatusi ning neid testida. Eelmise versiooni tarkvara ning andmebaasi kasutatakse üliõpilaste poolt „Andmebaasid II“ õppeaines.

Uurimiseks kasutatakse empiirilist meetodit (katsetused, vaatlused) ja samuti eelmise kahe versiooni lõputöö dokumente [18] [19], kus on programmi funktsionaalsus ja tööpõhimõte üksikasjalikult kirjeldatud. Kuna nendes on juba põhjalikud ja ülevaatlikud kirjeldused, siis järgnev on kõigest lühikokkuvõte.

2.1 Kasutatud tehnoloogiad

Selles jaotises antakse lühiülevaade tehnoloogiatest. Käesoleva töö eesmärgiks ei ole tarkvara migreerimine uutele platvormidele (keeled, raamistikud) ja seega kasutatakse ka selles neidsamu vahendeid.

2.1.1 Tagasüsteem

Tagasüsteemi e *back-end'i* e serveri osa arenduseks kasutatakse PHP programmeerimiskeelt ja Slim raamistikku, mis on lihtne, aga samas pakub kogu vajalikku funktsionaalsust serverirakenduse PHP keeles arendamiseks.

2.1.2 Eessüsteem

Eessüsteemi e *front-end* e kasutajaliidese arendamiseks kasutatakse Bootstrap 3 raamistikku, mis võimaldab sellesse sisseehitatud klasside abil luua kasutajaliidest, kirjutades võimalikult vähe HTML ja CSS koodi. Samuti võimaldab see lehel mugavalt positsioneerida DOM objekte. Põhimõtteliselt võimaldab see jagada lehe või mingi

DOM objekti kaheteistkümneks veeruks, mille põhjal saab määrata objekti suurust ning objekti paigutada. Funktsionaalsuse realiseerimiseks kasutatakse dünaamiliste kasutajaliideste tegemist lihtsustavat Javascripti AngularJS raamistikku.

2.1.3 Andmebaas

pgApex kasutab sisemiselt PostgreSQL andmebaasi, kus hoitakse andmeid loodud rakenduste kohta (näiteks struktuur, käitumine, väljanägemine, kuidas toimub kasutajate tuvastamine, milliseid andmebaasiobjekte läheb rakendusel vaja). Samuti on seal andmed ja funktsioonid, mis on vajalikud programmi toimimiseks (näiteks, mallid) [18].

2.1.4 Virtuaalmasin

Selleks, et vältida olukorda, kus programm töötab ühes arvutis ja teises ei tööta, kasutatakse virtuaalmasinate loomiseks ja haldamiseks mõeldud tööriista nimega Vagrant. Selle abil saab luua eraldi keskkonna, kuhu on paigaldadud programmi tööks vajalikud sõltuvused. Selle tulemusena on igaühel, kes tõmbab programmikoodi enda arvutisse, selle käivitamiseks ühesugune keskkond [18].

2.2 Funktsionaalsus

pgApex'it on võimalik kasutada administraatorina e arendajana või lõppkasutajana e kasutajana. Administraatorina saab luua ja muuta veebirakendusi ja kasutajana on võimalik neid mingis rakenduse poolt ettekirjutatud rollis kasutada [19].

Veebirakendus ise koosneb HTML lehtedest ja lehed koosnevad regioonidest, mis määravad elementide paigutuse. Teises versioonis on võimalikud regioonid navigatsiooni-, vormi-, raporti- ja HTML regioonid [19].

Veebirakenduse loomiseks on kohustuslik määrata andmebaas, milles olevaid andmeid hakkab rakendus kasutama. pgApex'il on lisaks oma andmebaas, kus on muuhulgas metaandmed kõikidest samal serveril asuvatest PostgreSQL andmebaasidest, millele on loodud pgApexi abil vähemalt üks rakendus. Neid andmeid esitatakse materialiseeritud vaadetena e hetktõmmistena (*materialized views*), mis on järgmised: *data_type*, *database*, *function*, *parameter*, *schema*, *view* ja *view_column* [19]. Erandina on hetktõmmises *database* andmed kõigi serveris olevate PostgreSQL andmebaaside kohta.

Kõigi rakenduste andmeid hoitakse pgApex’i andmebaasi baastabelites, millest igaüks kuulub mingisse andmebaasi loogilisse alamosasse e registrisse: rakenduste register, lehtede register, regioonide register, navigatsioonide register ja mallide register.

Andmebaasiga suhtlemiseks teeb programmi kasutajaliides POST ja GET päringuid tagasüsteemi kontrollerisse ja kontroller teeb andmebaasi päringuid läbi PDO. Tehtud päringu vastuse saab kasutajaliides JSON formaadis.

3 Taustauuring

Taustauuringu käigus otsitakse programme (arenduskeskkondi, arendusplatvorme), mis täidavad sama eesmärki, nagu pgApex ja uuritakse, kui sarnased need on pgApexiga ning kuidas nende abil saab realiseerida pgApex'is olevat funktsionaalsust, sealhulgas antud töös pgApex'is realiseeritavat uut funktsionaalsust, kui see on võimalik. Kui programm on tasuta või pakub prooviversiooni, siis proovitakse luua rakendust, mis oleks sarnane antud töö tulemuste valideerimiseks loodava rakendusega. Andmebaasina kasutatakse autori osalusel tehtud „Andmebaasid II" aineprojekti „Golfirajad" [27]. Funktsionaalsus, mida üritatakse loodavates rakendustes realiseerida on järgmine.

- Radade loomine ja muutmine.
- Radade nimekiri, kus iga raja juures on nupp konkreetse raja muutmiseks või detailandmete vaatamiseks.
- Alamregioonid, mille abil saab samal lehel lisada, lugeda, muuta ja kustutada rajaga 1:M või 1:0..1 seosetüüpide kaudu seotud olemite andmeid.
- Radade lõpetamine, mida pgApex'is saab realiserida tabelivormiga.
- Ridade optimistlik lukustamine ehk olemi või selle seoste andmete samaaegse muutmisega arvestamine.

Lisaks parema ülevaade andmiseks koostatakse kokkuvõtteks koondtabel (vt Tabel 1), kus iga programmi kohta vastatakse järgmistele küsimustele.

- Milliste andmebaasisüsteemide jaoks saab rakendusi luua?
- Rakenduse loomise keskkond: veebipõhine; töölauarakendus
- Rakenduse paigutamise viis: failid serverisse; andmed andmebaasis.
- Maksumus kasutajale: tasuta, tasuline.
- Kas lähtekood on avalik?
- Kas saab teha rakendust, mis kasutab andmebaasi läbi funktsioonide ja vaadete?
- Kas saab valida, millise olemitüübi puhul milliseid operatsioone (CRUD) läbi viiakse (näiteks teenuseid saab lisada ja lugeda, kuid kaupu saab nii lisada, uuendada kui kustutada)?
- Kas saab luua alamvorme?
- Milliseid operatsioone (CRUD) saab alamvormidel teha?
- Kas andmete muutmise vormil saab kasutada kalendrivalikut?

- Kas andmete muutmise vormil saab kasutada WSIWYG tekstivälja?
- Kas saab luua tabelivormi, mille taga on funktsiooni väljakutsumine?
- Kas realiseerib ridade optimistlikku lukustamist?

Uuritavate keskkondade otsimiseks kasutati Google otsingusõnu: *postgresql apex, postgresql application generator, rapid web application development tools, postgresql rapid application development, oracle apex alternatives.*

Lukustamise kasutamine tagab näiteks selle, et sama andmeelemendi (nt rida) samaaegsed uuendused ei kirjuta üksteist üle nii, et kehtima jääb ainult kõige viimane muudatus ja esimene läheb kaotsi. Öeldakse, et selliseid muudatusi tegevad operatsioonid on konfliktsed. Lukustamiseks on erinevaid strateegiaid – optimistlik ja pessimistlik. Pessimistlik strateegia eeldab, et konfliktseid operatsioone esineb sagedasti ja nendega tuleb ennetavalt tegeleda [28]. Pessimistliku strateegia korral lukustatakse andmeelement (näiteks tabeli rida) selle kasutamisel andmebaasisüsteemi poolt ennetavalt, et ei saaks tekkida selle elemendi samaaegsest muutmisest tulenevaid probleeme. Kui ühes andmebaasi kasutamise sessioonis on element mingi tehingu läbiviimiseks lukustatud, siis ei saa teistes sessioonides samal ajal seda elementi kasutada (lugeda/muuta) või vähemalt ainult muuta (kui andmebaasisüsteem realiseerib multiversioon konkurentsjuhtimist nagu näiteks PostgreSQL) enne kui elemendi lukustamiseni viinud tehing on andmebaasis lõppenud [29]. Pessimistlik strateegia on enamasti kasutusel töölauarakendustes. Optimistliku lukustamise strateegia kasutamine on levinud veebirakendustes ning lähtub eeldusest, et konfliktsed muutused on suhteliselt harvad. Kui konfliktid tekivad (näiteks samaaegselt uuendatakse sama rida), siis tuleb see olukord tuvastada ja konfliktiga tegeleda, et andmebaasi jõuaksid kooskõlalised andmed [29]. Andmeelementi muutev tarkvara peab kontrollima kas peale selle elemendi viimast lugemist on see element muutunud või mitte ja kui on, siis ei saa muudatust teha [29].

Antud töös pakub huvi, kas loodav andmebaasirakendus toetab ridade optimistlikku lukustamist. Selle proovimiseks avatakse samaaegselt sama raja muutmise vorm erinevates veebilehitseja akendes – tarkvara peab raja andmeid lugema. Ühes veebilehitseja aknas muudetakse andmeid ja salvestatakse muudatus. Peale seda on teises aknas olevad andmed juba vananenud. Siis muudetakse raja andmeid teises aknas

ja proovitakse salvestada. Kui optimistlik lukustamine puudub, siis andmete muutmine sellest aknast õnnestub ning lõpuks on raja kohta salvestatud ainult teisest aknast tehtud muudatused. Optimistliku lukustamise kasutamise korral peaks andmete muutmine antud näite puhul teisest aknast ebaõnnestuma, sest muutuse salvestamise hetkel selgub, et neid andmeid on peale lugemist juba kellegi poolt muudetud. Samuti uuritakse seda, kas rakenduse loomisel on võimalus anda korraldus selles optimistliku lukustamise kasutamiseks.

3.1 FADEX

FADEX on vähese koodi kirjutamisega (*low-code)* arenduskeskkond PostgreSQL jaoks [13].

Paraku pakub FADEX ainult tasulist versiooni ning selle kodulehel [13] puudub dokumentatsioon, et selle alusel saaks uurida keskkonna võimalusi ja funktsionaalsust.

3.2 PHPRad

PHPRad on veebirakenduste arenduskeskkond, mis võimaldab luua PHP andmebaasirakendusi erinevates andmebaasisüsteemides loodud andmebaaside põhjal [14]. Rakenduste loomine toimub erinevate komponentide kokkupanemise abil [14]. Katsetamiseks kasutati versiooni 2.7.3.

Programm on tasuline, kuid pakub 30 päevast prooviversiooni. Rakenduse loomiseks ja haldamiseks on vaja arvutisse paigaldada töölauarakendus (Lisa 1 Joonis 24). Rakenduse kasutamiseks on esialgu vaja see publitseerida *e* genereerida, vajutades „Publish“ nuppu (Lisa 1 Joonis 24). Seda on vaja samuti teha rakenduse muutmisel, et muudatused jõustuksid. Loodud rakendus on tavaline veebirakendus, kus tagasüsteemiks on PHP. Seetõttu on selle käivitamiseks vaja veebiserverit. Antud töö raames kasutati uuritavate rakenduste käivitamiseks XAMPPi [30].

Esiteks prooviti realiseerida kõikide radade vaatamise lehte, mis on PHPRad programmi puhul lihtne, kuna uue rakenduse loomisel iga baastabeli (edaspidi tabeli) põhjal genereeritakse kohe leht, kus on olemite nimekiri (Lisa 1 Joonis 25), millelt avanevad detailvaate (Lisa 1 Joonis 26), loomise (Lisa 1 Joonis 27) ja muutmise vormid (Lisa 1 Joonis 28). Vaadete puhul on võimalikud ainult nimekirja ja detailvaate lehed.

Järgmisena prooviti realiseerida alamregiooni, kus saab vaadata konkreetse raja kategooriaid ning neid lisada ja kustutada.

Olulised erinevused pgApex'iga on järgmised.

- Andmete lisamine, muutmine ja kustutamine toimub otseselt tabelite põhjal INSERT, UPDATE ja DELETE lausete abil ning nende operatsioonide teostamiseks pole võimalik kasutada andmebaasis olevaid funktsioone, mis lihtsustab rakenduste loomist, aga mingil määral piirab võimalikku rakenduste funktsionaalsust. Näiteks pole võimalik radade nimekirja lisada nuppu, mis kutsuks välja andmebaasis loodud funktsiooni konkreetse raja lõpetamiseks ehk seisundi muutmiseks.

- Läbi vaadete on võimalik ainult andmeid lugeda. See tähendab, et PHPRad keskkonna arhitektuuri kohaselt pole näiteks võimalik luua sellist vormi, mis oleks eeltäidetud andmetega mingist vaatest, kuna nagu on eelmises punktis mainitud, siis andmete haldamine toimub tabelite põhjal täidetavate SQL lausete abil.

- Andmete lugemiseks on võimalik luua alamregioone (Lisa 1 Joonis 29), kuid pole võimalik luua alamvorme andmete lisamiseks, muutmiseks ja kustutamiseks. Alamvorm tähendab, et see on peavormiga samal lehel ja selles näidatakse peavormis parajasti esitatava olemiga seotud olemite andmeid. Võimalus lisada vormiga lehele teist vormi on olemas, kuid puudub funktsionaalsus selle sidumiseks peavormiga, et saaks näiteks lisada kategooriaid konkreetsele rajale, mille andmed on kuvatud peavormis.

- Ei realiseeri ridade optimistlikku lukustamist.

- Rakenduste haldamiseks peab arendaja paigaldama enda tööarvutisse eraldi töölauarakenduse ning rakenduse loomisel ja iga muudatuse järel tuleb uued/muutunud failid publitseerida. Rakenduse loomise järel tuleb kasutuselevõtuks see panna veebiserverisse nagu tavaline veebirakendus. pgApexis tehakse seda kõike läbi ühe veebipõhise keskkonna ehk pärast rakenduse loomist või selle muutmist saab kohe tulemust näha ja kasutada.

3.3 PostgreSQL PHP Generator

PostgreSQL PHP Generator võimaldab genereerida PHP rakendusi PostgreSQL andmebaaside põhjal ilma koodi kirjutamiseta [15].

Keskkonnast pakutakse tasuta „Lite“ ja tasulist „Professional“ versiooni. Tasuta versioonil on võrreldes tasulise versiooniga palju funktsionaalsuseid puudu [31]. Tasuline version pakub 30-päevast prooviversiooni, seega kasutati proovimiseks seda. Katsetamiseks kasutati versiooni 18.3.

Rakenduse loomine ja muutmine toimub samm-sammult. Esimeseks sammuks on rakenduse poolt kasutatava andmebaasiga ühenduse loomikseks vajaliku info sisestamine (Lisa 2 Joonis 30). Teise sammuna tuleb valida, milliseid andmebaasiobjekte rakendus kasutab (Lisa 2 Joonis 31). Saab kasutada vaateid, tabeleid või luua SQL päringuid Samal lehel saab valida, kas luua valitud objekti põhjal menüüvalik menüüpuu kõige kõrgemale tasemele. Järgmine samm on lehtede omaduste täpsustamine. Sealhulgas saab luua alamlehte, seadistada vormivälju muutmise lehel jne (Lisa 2 Joonis 32). Järgmiseks on vaja valida rakendusele teema ehk väljanägemine (Lisa 2 Joonis 33). Viimase sammuna tuleb valik, kus saab määrata rakenduse lehtede päise ja jaluse väljanägemist, autentimist, andmebaasidraiverit, lokaliseerimist ja loodavate rakenduse failide asukohta (Lisa 2 Joonis 34). Rakenduse genereerimiseks on vaja vajutada “Ready“ nuppu.

Kui lisada andmebaasiobjekte ja valida, et nende põhjal luuakse lehed ning kasutada nende omadustena vaikimisi valikuid, siis rakenduse loomisel luuakse iga objekti kohta olemite nimekirja leht (Lisa 2 Joonis 35), detailvaate leht (Lisa 2 Joonis 36), olemi loomise leht (Lisa 2 Joonis 37) ja olemi muutmise leht (Lisa 2 Joonis 38).

Olulised erinevused võrreldes pgApex’iga.

- Nagu PHPRad keskkonnas, siis ka PostgreSQL PHP Generaator võimaldab luua, muuta ja kustutada andmeid ainult INSERT, UPDATE ja DELETE SQL lausete abil. Võrreldes PHPRad keskkonnaga aga on võimalus teha neid operatsioone ka läbi vaadete, kui vaade küsib andmeid ainult ühest tabelist või kui vaatel on INSTEAD OF trigerid või reeglid, tänu millele saab kasutaja vaate kaudu andmeid muuta.

- Sarnaselt PHPRad keskkonnaga pole võimalik andmetega opereerimiseks kasutada andmebaasis olevaid funktsioone. Seega pole võimalik täielikult realiseerida radade lõpetamise funktsionaalsust.

- On võimalik luua alamraporteid, kuid pole võimalik luua alamvorme ning alamraportid asuvad eraldi lehtedel (Lisa 2 Joonis 39). On võimalik realiseerida olemiga 1:M ja 1:0..1 seosetüübi kaudu seotud olemite andmete lisamist, muutmist ja kustutamist, kuid seda tavaliste vormide abil nagu Lisas 2 Joonistel Joonis 37 ja Joonis 39, st mitte alamvormide kaudu.

- Ei realiseeri ridade optimistlikku lukustamist.

- Genereeritavad rakendused on tavalised veebirakendused. Seega neid peab üles seadma veebiserveris ja faile peab uuendama (publitseerima) iga loomise või muutmise operatsiooni järel.

3.4 Radzen

Radzen on töörist ärirakenduste loomiseks [16].

On olemas tasuta *Community* versioon ja tasulised *Professional* ning *Enterprise* versioonid. Proovirakenduse loomiseks kasutati tasuta versiooni. Katsetamiseks kasutati versiooni 2.47.2.

Rakenduste loomiseks ja haldamise jaoks on töölauarakendus. Rakenduse loomisel on vaja valida selle poolt kasutatavat tehnoloogia, mall (kas on tühi või näidisrakendus), teema, nimi ja rakenduse failide asukoht (Lisa 3 Joonis 40). Andmebaasiga sidumiseks on võimalik valida ja seadistada andmeallikaid (Lisa 3 Joonis 41). Antud juhul valiti andmeallikaks PostgreSQL andmebaas ning selle puhul on vaja sisestada allika nimi, andmebaasiserveri aadress, andmebaasi nimi ning selle kasutajanimi ja parool. Peale seda saab genereerida CRUD funktsionaalsusega lehti andmebaasis olevate baastabelite põhjal ja R funktsionaalsusega vaadete põhjal ning neid muuta (Lisa 3 Joonis 42). CRUD lehtede hulka kuuluvad olemite nimekirja vaatamise (Lisa 3 Joonis 43), olemi lisamise (Lisa 3 Joonis 44) ja olemi muutmise (Lisa 3 Joonis 45) lehed. Nimekirja lehel on iga olemi juures nupp selle olemi andmete kustutamiseks. Tuleb esile tuua, et

Radzen võimaldab realiseerida optimistlikku lukustamist. Radzen keskkonna puhul saab andmeallika seadistamisel (Lisa 3 Joonis 42) märkida "Enable optimistic concurrency".

Olulised erinevused võrreldes pgApex'iga.

- Andmete töötlemiseks pole võimalik kasutada andmebaasis olevaid funktsioone. Nagu ka eelneva kahe programmi puhul, siis toimub andmete lisamine, muutmine ja kustutamine SQL lausete abil tabelite või vaadete põhjal.

- Alamraportite loogika on selline, et olemite nimekirjas konkreetse olemi peale vajutades avanevad nimekirja all alamraportid selle olemi kohta (Lisa 3 Joonis 47).

- Ei ole võimalik luua eraldi lehte olemi detailandmete vaatamiseks. Detailvaatega sama eesmärgi täitmiseks saab kasutada olemi muutmise vormi.

- Vormi lehele ei ole võimalik lisada alamvorme. On võimalik realiseerida olemiga 1:M või 1:0..1 seosetüübi kaudu seotud olemite andmete lisamist, muutmist ja kustutamist, kuid tehes seda eraldi vormide abil.

- Loodud rakendus vajab veebiserverit ja failide publitseerimist peale loomist ning iga muudatust.

3.5 4WS Platform

4WS Platform on keskkond, mille abil saab kiiresti luua veebi- ja mobiilirakendusi [17]. Katsetamiseks kasutati versiooni 5.3.2.

Arenduskeskkond on tasuta ja nõuab rakendusete loomiseks töölauarakenduse paigaldamist. Nii arenduskeskkonna, kui ka selle abil loodavad rakendused kasutavad Java't ning seetõttu nõuavad selle arvutisse paigaldamist. Lisas 4 Joonis 47 ja Joonis 48 esitatakse arenduskeskkonna paigaldamisel infot küsivad aknad. Paigaldamise järel genereeritakse kasutaja poolt valitud kausta faile ning arenduskeskkonna käivitamine toimub käsurea abil. Tarkvara ise käivitub veebirakendusena, kus saab luua rakendust ja seda käivitada. Arenduskeskkonna installatsioon on seotud ainult ühte andmebaasiga (nagu on andmebaasi seadistuse lehelt näha Lisas 4 Joonis 47) ning teise andmebaasi jaoks oleks vaja paigaldada uus installatsioon.

Paraku ei õnnestunud autoril kodulehel oleva juhendi abil arenduskeskkonda tööle saada ei nõutud Java 7, kui ka uuema Java versiooni puhul. Süsteem kuvas jätkuvalt veateateid. Seega uuriti võimalikku funktsionaalsust tarkvara kodulehel oleva dokumentatsiooni abil ning erinevused pgApex'ist on esitatud Tabel 1.

3.6 Openxava

Openxava on avatud lähtekoodiga vähese koodi kirjutamisaga (*low-code*) veebirakenduste arendamise keskkond, kus andmete töötlemine toimub peamiselt Java klasside abil [32]. Katsetamiseks kasutati versiooni 6.3.1.

Arenduskeskkond on tasuta ning allalaadimisel on kasutaja käsutuses põhimõtteliselt Java tööruum (*workspace)*, mida on vaja avada mõne IDE abil. Rakenduse loomiseks selles tööruumis on vaja mustri põhjal luua Java projekt, kus on juba eelgenereeritud rakenduse töötamiseks vajalik loogika ning arendaja peab kirjeldama täiendavaid Java klasse [12]. Loodud rakendus on tavaline veebirakendus, kus tagasüsteemiks on Java.

Pärast olemitüüpidele vastavate klasside kirjeldamis genereeris keskkond selle olemite nimekirja (Lisa 5 Joonis 49) ja lisamise (Lisa 5 Joonis 50) vormi. Paraku ei õnnestunud saavutada andmebaasist loetud olemite andmete alusel nimekirja kuvamist vastavale lehele, kuigi andmebaasi ühendamine toimus vastavalt kodulehel olevale juhendile [33]. Alla laetud tööruumis on olemas näidisrakendus, kus on näha, et saab ka luua olemi muutmise lehti (Lisa 5 Joonis 51) ning on olemas ka tabelivormiga sarnane funktsionaalsus olemite nimekirja lehel, kuid ainult olemite kustutamiseks (Lisa 5 Joonis 52).

Tarkvara struktuuri ning dokumentatsiooni uurimise järel on selge, et võrreldes teiste sama eesmärgi täitvate arenduskeskkondadega on rakenduse loomiseks vaja kirjutada suhteliselt suurt hulka Java koodi ning sellest tuleneb ka see, et rakenduse looja peab oskama Java programmeerimiskeelt.

Olulised erinevused võrreldes pgApexiga.

- Rakenduse loomine toimub Java projektina ning rakenduse looja peab selleks oskama Java keeles progremmeerida.

- Suhteliselt suure hulga funktsionaalsust peab realiseerima Java keele abil. Sellest tuleneb, et alamvormide loomise võimalus sõltub rakenduse looja programmeerimisoskustest.

- Nagu kõikide eelnevalt kirjeldatud programmide puhul, siis sisseehitatud vaikimisi funktsionaalsuses pole võimalik kasutada andmetöötluseks andmebaasis olevaid funktsioone.

- Ei realiseeri ridade optimistlikku lukustamist.

- Loodud rakendus on Java tagasüsteemiga veebirakendus, mille peab paigaldama veebiserverisse.

3.7 Uuringu kokkuvõte

Autor leidis kuus pgApex'iga sama eesmärki täitvat arenduskeskkonda, millest nelja sai ta ka oma arvutis katsetada.

Kõikidel uuritud keskkondadel on oma erinev funktsionaalsus, kuid antud uuringus otsiti ja võimalusel katsetati pgApexi kolmanda versiooniga sarnast funktsionaalsust. Uuring näitas, et ühegi uuritud keskkonna abil pole võimalik luua andmebaasirakendust, mis toimiks ainult läbi andmebaasi avalikku liidese, mille moodustavad vaated ja funktsioonid (vaadete ja funktsioonide kasutamise eeliseid ja puuduseid kirjeldatakse Rait Raidmaa magistritöös [18]). Mõnedes uuritud vahendites saab kasutada andmebaasi vaateid (PostgreSQL PHP Generator, Radzen, Openxava), kuid mitte üheski ei saa kasutada funktsioone. Meenutame, et pgApexi abil loodud andmebaasirakenduses peab kogu andmete kasutus toimuma läbi vaadete ja funktsioonide.

Ainult ühes uuritud keskkonnas (Radzen) on sisseehitatud optimistliku lukustamise tugi. Optimistliku lukustamise võimalus reliseeritakse antud lõputöö tulemusel ka pgApexis.

Kõik uuritud keskkonnad nõuavad rakenduste loomiseks ja haldamiseks töölauarakendust ning rakendused luuakse tavaliste veebirakendustena (hulk faile), mis tuleb toimimiseks paigaldada veebiserverisse. pgApex seevastu hoiab andmeid loodud rakenduste kohta enda andmebaasis ning nende haldusliides asub ühes kohas (praegu

see on apex.ttu.ee server). Haldusliidesele pääsevad ligi kõik, kellel on internetiühendus, veebilehitseja ja keskkonna kasutamise õigus. Tänu sellele on rakenduste loomine kiirem ja lihtsam võrreldes teiste keskkondadega, sest pole vaja paigaldada lisatarkvara ning rakenduste käivitamine ja kasutamine on realiseeritud pgApex keskkonna sees.

Üheski uuritud keskkonnas pole võimalik luua alamvorme nii, et need oleksid ühel lehel olemi muutmise vormiga ja seostada olemi unikaalse identifikaatori kaudu põhivormi ja alamvormi.

Uuritud keskkondadel on ka eeliseid võrreldes pgApex keskkonnaga. Näiteks võimaldavad kõik keskkonnad kasutada andmebaase erinevates serverites. pgApex aga võimaldab kasutada ainult neid andmebaase, mis asuvad pgApexiga ühes serveris. Kommentaariks tuleb lisada, et tänu PostgreSQLi väliste tabelite mehhanismile saab luua integratsiooni andmebaasi, mis koondab kokku andmeid erinevates serverites olevatest andmebaasidest (ei pea olema loodud PostgreSQLis) ning sellisele andmebaasile saaks luua pgApex abil andmebaasirakenduse.

Autor leidis ka funktsionaalsusi, mis pgApex keskkonnas puuduvad. Selliste funktsionaalsuse näideteks on lehtede loomine andmebaasi baastabelite põhjal või võimalus vaadata nimekirja lehel olemitega seotud olemite andmeid (PHPRad). Viimane tähendab seda, et olemite nimekirja lehel saab vajutada konkreetsele olemile ja samal lehel selle kõrval avaneb plokk selle olemiga seotud olemite andmetega. PHPRad ja PostgreSQL Generator keskkondadel on ka selline eelis, et need oskavad rakenduste genereerimisel ise tuvastada tabelite vahelisi seoseid. Selle näiteks on tabel *Rada* autori osalusel tehtud „Andmebaasid II“ aineprojektis. Selles tabelis on veerg *raskus_kood*, milles olevate andmete alusel on tabelis *Rada* olevad andmed seotud tabelis *Raskus_aste* olevate andmetega. (rada_kood) on tabelis *Rada* välisvõti. Tabelis *Raskus_aste* on kaks veergu: *raskus_kood* ja *nimetus*. Raja lisamise ning muutmise vormide genereerimisel tuvastatakse see seos ning selle tulemusel on nendes raja raskusastme valimiseks liitboks, kust saab valida raskusastet nimetuse järgi. pgApex keskkonnas tuleb selline funktsionaalsus arendajal käsitsi realiseerida.

Tabel 1 iseloomustatakse/võrreldakse antud uuringus vaadeldud keskkondi küsimuste vastuste alusel. Küsimused on koostatud pgApexi funktsionaalsuse (sealhulgas

kolmandas versioonis realiseeritava funktsionaalsuse) põhjal. Vastused nendele samadele küsimustele pgApexi põhjal on esitatud Tabel 2.

Tabel 1. pgApexiga sarnast eesmärgi täitvate keskkondade võrdluse tabel.

	PHPRad (2.7.3)	**PostgreSQL PHP Generator (18.3)**	**Radzen (2.47.2)**	**4WS Platform (5.3.2)**	**Openxava (6.3.1)**
Milliste andmebaasi-süsteemide jaoks saab rakendusi luua?	Oracle, MS SQL Server, MySQL, PostgreSQL, Mongo DB	PostgreSQL	MS SQL Server, MySQL, PostgreSQL, Oracle	Oracle, PostgreSQL, MS SQL Server, MySQL	MySQL, PostgreSQL, Oracle, MS SQL Server, AS/400, Informix, Db2, Firebird
Rakenduse loomise keskkond: veebipõhine; töölaua-rakendus	Töölaua-rakendus	Töölaua-rakendus	Töölaua-rakendus	Keskkond on veebi-rakendus, mida paigaldatakse ja seadistatakse töölauarakendu use kaudu	Töölaua-rakendus
Rakenduse paigutamise viis: failid serverisse; andmed andmebaasis	Failid serverisse	Failid serverisse	Failid serverisse	Failid serverisse	Failid serverisse

	PHPRad (2.7.3)	**PostgreSQL PHP Generator (18.3)**	**Radzen (2.47.2)**	**4WS Platform (5.3.2)**	**Openxava (6.3.1)**
Maksumus kasutajale: tasuta, tasuline	Tasuline. On olemas 30 päevane proovi-versioon	"Lite edition" on tasuta. "Professional edition" on tasuline ja pakub 30 päevast proovi-versiooni	"Community" versioon on tasuta. "Professional" ja "Enterprise" versioonid on tasulised. "Professional" versioon pakub 15 päevast prooviversiooni	"Community" versioon on tasuta. "Enterprise" versioon on tasuline	Üldjuhul tasuta, aga on olemas tasuline "XavaPro" versioon
Kas lähtekood on avalik?	Ei	Ei	Ei	"Community" versiooni lähtekood on avalik LGPL v2 litsentsiga	Jah. LGPL litsents.
Kas saab teha rakendust, mis kasutab andmebaasi läbi funktsioonide ja vaadete?	Ei	Osaliselt (ainult vaated)	Osaliselt (ainult vaated)	Dokumentat-siooni järgi ei	Osaliselt (ainult vaated)
Kas saab valida, millise olemitüübi puhul milliseid operatsioone (CRUD) läbi viiakse (näiteks	Jah	Jah	Jah	Dokumen-tatsiooni järgi jah	Jah

	PHPRad (2.7.3)	**PostgreSQL PHP Generator (18.3)**	**Radzen (2.47.2)**	**4WS Platform (5.3.2)**	**Openxava (6.3.1)**
teenuseid saab lisada ja lugeda, kuid kaupu saab nii lisada, muuta kui kustutada)?					
Kas saab luua alamvorme?	Ei	Ei	Ei	Dokumen-tatsiooni järgi ei	Ei
Milliseid operatsioone (CRUD) saab alamvormidel teha?	Pole alamvorme	Pole alamvorme	Pole alamvorme	Pole alamvorme	Pole alamvorme
Kas andmete muutmise vormil saab kasutada kalendri-valikut?	Jah	Jah	Jah	Dokumen-tatsiooni järgi jah	Jah
Kas andmete muutmise vormil saab kasutada WSIWYG tekstivälja?	Jah	Jah	Jah	Dokumen-tatsiooni järgi jah	Jah
Kas saab luua tabelivormi, mille taga on	Ei	Ei	Ei	Ei	Ei

	PHPRad (2.7.3)	**PostgreSQL PHP Generator (18.3)**	**Radzen (2.47.2)**	**4WS Platform (5.3.2)**	**Openxava (6.3.1)**
funktsiooni välja-kutsumine?					
Kas realiseerib ridade optimistlikku lukustamist?	Ei	Ei	Jah	Dokumen-tatsioonis ei ole informat-siooni	Ei

Tabel 2. pgApexi omadused.

	pgApex (versioon 3)
Milliste andmebaasisüsteemide jaoks saab rakendusi luua?	PostgreSQL
Rakenduse loomise keskkond: veebipõhine; töölauarakendus	Veebipõhine
Rakenduse paigutamise viis: failid serverisse; andmed andmebaasis	Andmed andmebaasis
Maksumus kasutajale: tasuta, tasuline	Tasuta
Kas lähtekood on avalik?	Jah
Kas saab teha rakendust, mis kasutab andmebaasi läbi funktsioonide ja vaadete?	Jah
Kas saab valida, millise olemitüübi puhul milliseid operatsioone (CRUD) läbi viiakse (näiteks teenuseid saab lisada ja lugeda, kuid kaupu saab nii lisada, muuta kui kustutada)?	Jah
Kas saab luua alamvorme?	Jah
Milliseid operatsioone (CRUD) saab alamvormidel teha?	CRUD
Kas andmete muutmise vormil saab kasutada kalendri-valikut?	Jah

	pgApex (versioon 3)
Kas andmete muutmise vormil saab kasutada WSIWYG tekstivälja?	Jah
Kas saab luua tabelivormi, mille taga on funktsiooni väljakutsumine?	Jah
Kas realiseerib ridade optimistlikku lukustamist?	Jah

4 Väljalaske planeerimine

Et otsustada, mis funktsionaalsust täpsemalt realiseerida ja panna paika ülesannete keerukustasemed ning prioriteedid, viiakse järgnevalt läbi pgApexi kolmanda versiooni väljalaske planeerimine.

4.1 Metoodika

Pärast väljalaske planeerimise kohta erinevatest infoallikatest lugemist [34], [35], [36], [37] leiti, et Nikolai Kopa [19] bakalaureusetöös viidatud meetod [24] on antud juhul kõige parem, sest see selgitab näiteid kasutades ja lihtsalt, kuidas väljalaske planeerimist teha. Seega kasutatakse ka antud töös seda meetodit, kohandades seda ühtlasi vajadusel autori nägemuste ja nõuete järgi.

Antud meetod näeb ette täitmata ülesannete nimekirja e soovilogi (*backlog*) koostamise, kus esitatakse funktsionaalseid nõudeid. Antud töös esitatakse funktsionaalsed nõuded kasutuslugudena (*user stories*). Seejärel määratakse igale nõudele prioriteet ning hinnatakse selle täitmise keerukust. Kui see on tehtud, siis jagatakse nõuded iteratsioonidesse (arenduse pisitsüklitesse) ehk kindlatesse ajavahemikkesse, mille käigus peab ettenähtud hulk nõudeid täidetud saama. Ajakarbi meetod [26] tähendab, et kui mõni nõue jääb täitmata, siis ei pikendata iteratsiooni, vaid see viiakse tagasi täitmata ülesannete nimekirja.

4.1.1 Ülesannete keerukuste hindamine

Kasutuslugude keerukuse hinnangutena kasutatakse Fibonacci jada arve [38]. Tavaliselt kui arendusmeeskonna liikmed hindavad kasutuslugude keerukust, siis kõikidel on erinevad arvamused ja nägemused mingi konkreetse ülesande keerukuse kohta. Fibonacci jada arvud aitavad hajuma kippuvaid hinnanguid kokku tuua. Kui näiteks kahe arendaja arvamused on erinevad, siis nad saavad suure tõenäosusega leppida kokku ühe Fibonacci arvu peale, mis sobiks nende mõlema keerukuse hinnanguga. Tegelikult on see antud lõputöö puhul ebaoluline, kuna tööd teeb üks arendaja (lõputöö autor). Sellest hoolimata on Fibonacci arvudel keerukuse hindamisel võrreldes tavalise monotoonselt kasvava arvujadaga eelis. Keerukuse hindamine on üldjuhul raske ja ebamäärane protsess ning seega arvude valiku vähendamine lihtsustab seda.

4.2 Kasutuslood ja esialgne keerukuste hindamine

Tooteomaniku (kes on antud juhul lõputöö juhendaja) paika pandud funktsionaalsed nõuded ning nende realiseerimise prioriteedid esitatakse Tabel 3. Autor määras oma taustateadmistest ja kogemustest lähtuvalt neile esialgsed keerukuse hinnangud.

Tabel 3. Kasutuslood koos nende hinnatud keerukustega.

ID	Kasutuslugu	Prioriteet	Keerukus
1	Arendajana ma tahan võimalust luua olemi andmete lisamise vormi, mis toetab tekstivälju, liitbokse, märkeruute ja kalendrivalikut, et saaksin andmebaasis registreerida andmeid uute olemite kohta.	1	5
2	Arendajana ma tahan võimalust valida nimekirjavormist olemi, mille andmete muutmise korralduse andmine avab valitud olemi hetkeandmeid sisaldava andmete muutmise vormi, mis toetab tekstivälju, liitbokse, märkeruute ja kalendrivalikut, et saaksin andmebaasis muuta olemite kohta käivaid andmeid.	2	5
3	Arendajana ma tahan võimalust häälestada tekstiväljade, liitbokside, märkeruutude ja kalendrivalikute väljanägemist, et pakkuda kasutajatele parimat kasutuskogemust.	3	8
4	Arendajana tahan ma olemi andmete muutmisel samal lehel lisada, lugeda, muuta ja kustutada selle olemiga 1:M või 1:0..1 seosetüübi kaudu seotud olemite andmeid, et kasutajaliides oleks kompaktsem ja ülevaatlikum.	4	16
5	Arendajana ma tahan võimalust lisada nimekirjavormi otsingukomponendi, et saaksin otsida konkreetseid olemeid.	5	8
6	„Arendajana ma tahan rakendust eksportida ja importida, et teha rakenduse varukoopiat või taastada rakendust varem loodud varukoopia alusel.“ [19]	5	16

4.3 Järgmine keerukuste hindamine ja uus kasutuslugu

Pärast programmi uurimist ja arendamise alustamist selgus, et ühe kasutusloo (ID 4) keerukus on reaalsuses suurem, kui oli alguses hinnatud. Sellepärast hinnati seda uuesti. Samuti lisas tooteomanik (töö juhendaja) veel ühe nõude, mis on vajalik CRUD täieliku funktsionaalsuse realiseerimiseks ning täpsustusid prioriteedid. Tabel 4 on kasutuslugude nimekiri koos uuesti hinnatud keerukusega, prioriteetidega ja uue kasutuslooga (muudatused on tähistatud rasvase fondiga).

Tabel 4. Uuendatud kasutuslood.

ID	Kasutuslugu	Prioriteet	Keerukus
1	Arendajana ma tahan võimalust luua olemi andmete lisamise vormi, mis toetab tekstivälju, liitbokse, märkeruute ja kalendrivalikut, et saaksin andmebaasis registreerida andmeid uute olemite kohta.	1	5
2	Arendajana ma tahan võimalust valida nimekirjavormist olemi, mille andmete muutmise korralduse andmine avab valitud olemi hetkeandmeid sisaldava andmete muutmise vormi, mis toetab tekstivälju, liitbokse, märkeruute ja kalendrivalikut, et saaksin andmebaasis muuta olemite kohta käivaid andmeid.	2	**8**
3	Arendajana ma tahan võimalust häälestada tekstiväljade, liitbokside, märkeruutude ja kalendrivalikute väljanägemist, et pakkuda kasutajatele parimat kasutuskogemust .	3	8
4	Arendajana tahan ma olemi andmete lisamisel või muutmisel samal lehel lisada, lugeda, muuta ja kustutada selle olemiga 1:M või 1:0..1 seosetüübi kaudu seotud olemite andmeid, et kasutajaliides oleks kompaktsem ja ülevaatlikum.	4	16
5	Arendajana ma tahan võimalust lisada nimekirjavormi otsingukomponendi, et saaksin otsida konkreetseid olemeid.	**6**	8

ID	Kasutuslugu	Prioriteet	Keerukus
6	„Arendajana ma tahan rakendust eksportida ja importida, et teha rakenduse varukoopiat või taastada rakendust varem loodud varukoopia alusel.“ [19]	**6**	16
7	**Arendajana ma tahan võimalust tabelivormi funktsioonile valikuliselt anda ette *xmin* (rea versiooni) väärtust, et saaks arvestada ridade samaaegse muutmisega erinevate kasutajate poolt.**	**5**	**5**

4.4 Iteratsioonidesse jaotamine

Iteratsioonidesse jaotamine toimus kasutuslugude prioriteetide põhjal. Teiste sõnadega, kõige prioriteetsemad nõuded tuli täita kõigepealt. Tooteomaniku (juhendaja) kooskõlastamisel otsustati, et kõige madalama prioriteediga ülesandeid (ID 6 – Tabel 4) ei jõua antud lõputöö käigus realiseerida ja selle tulemusena jõudis väljalaske plaani viis ülesannet.

Esialgu otsustatid läbi viia kolm iteratsiooni: kaks on ülesannete realiseerimiseks ning üks jääb varuks. See tähendab, et kui mõni ülesanne esimest või teisest iteratsioonist jääb täitmata, siis see läheb kolmandasse iteratsioonisse. Kui juhendaja poolt tuli üks lisaülesanne (ID 7), siis otsustati see panna kolmandasse (varu) iteratsiooni, et mitte suurendada iteratsioonide arvu. Antud arendustsüklis oli iteratsiooni pikkuseks viis nädalat.

Tabel 5 on lõplik väljalaske plaan koos iteratsioonidega. Sinisega on märgitud esimene-, oranžiga teine- ja kollasega kolmas iteratsioon.

Tabel 5. Lõplik väljalaske plaan.

ID	Kasutuslugu	Prioriteet	Keerukus
1	Arendajana ma tahan võimalust luua olemi andmete lisamise vormi, mis toetab tekstivälju, liitbokse, märkeruute ja kalendrivalikut, et saaksin	1	5

ID	Kasutuslugu	Prioriteet	Keerukus
	andmebaasis registreerida andmeid uute olemite kohta.		
2	Arendajana ma tahan võimalust valida nimekirjavormist olemi, mille andmete muutmise korralduse andmine avab valitud olemi hetkeandmeid sisaldava andmete muutmise vormi, mis toetab tekstivälju, liitbokse, märkeruute ja kalendrivalikut, et saaksin andmebaasis muuta olemite kohta käivaid andmeid.	2	8
3	Arendajana ma tahan võimalust häälestada tekstiväljade, liitbokside, märkeruutude ja kalendrivalikute väljanägemist, et pakkuda kasutajatele parimat kasutuskogemust .	3	8
4	Arendajana tahan ma olemi andmete lisamisel või muutmisel samal lehel lisada, lugeda, muuta ja kustutada selle olemiga 1:M või 1:0..1 seosetüübi kaudu seotud olemite andmeid, et kasutajaliides oleks kompaktsem ja ülevaatlikum.	4	16
7	Arendajana ma tahan võimalust tabelivormi funktsioonile valikuliselt anda ette *xmin* (rea versiooni) väärtust, et saaks arvestada ridade samaaegse muutmisega erinevate kasutajate poolt.	5	5

5 Muudatused mudelitena

Antud töös realiseeritav uus funktsionaalsus ei näe ette süsteemi äriarhitektuuri uute funktsionaalsete allsüsteemide ja registrite lisamist. Seega järgnevalt kirjeldatakse olemasolevate funktsionaalsete allsüsteemide ja registrite osi, mida mõjutas uue funktsionaalsuse lisamine. Kirjeldamine hõlmab osasid strateegilisest analüüsist, detailanalüüsist ja füüsilisest disainist. Samuti ei lisatud töö käigus süsteemi kasutajateks uusi rolle (tegutsejaid), seega neid on endiselt kaks: arendaja ja kasutaja.

Funktsionaalsed allsüsteemid, mida töö käigus muudeti.

- Regioonide funktsionaalne allsüsteem.
- Mallide funktsionaalne allsüsteem.

Registrid, mida muudeti.

- Regioonide register.
- Mallide register.

5.1 Muudatused regioonide funktsionaalse allsüsteemi eskiismudelis

Järgnevalt kirjeldatakse regioonide funktsionaalse allsüsteemi eskiismudeli osi, mida pgApexi kolmandas versioonis muudeti või täiendati.

5.1.1 Muudatused kasutusjuhtude eskiismudelis

pgApexi kolmandas versioonis lisandusid regioonide funktsionaalse allsüsteemi kirjeldusse uued kasutusjuhud ja ühte (Muuda tabelivormi regiooni) täiendati.

Kasutusjuht: Halda vormiga raporti regiooni

Tegutsejad: Arendaja

Kirjeldus: Vormiga raporti regiooni lisamisel või muutmisel on arendaja esialgsed tegevused sarnased raporti ja detailvaate regiooni haldamisele: esialgu peab arendaja valima vaate, millest võetakse raportis kuvatavad andmed. Kui vaade on valitud, siis kuvatakse täitmiseks järgmised väljad: unikaalset identifikaatorit sisaldava veeru nimi

(see väärtus pannakse kaasa url GET parameetriga, kui kasutaja liigub vormilehele, vajutades raportis konkreetse olemi juures nuppu); regiooni nimi; regiooni järjekorranumber lehel; regiooni mall; valik, kas regioon on nähtav või mitte; raporti mall; valik, kas näidata päist või mitte; valik, kas lisada raporti kohale nupp olemi lisamiseks või mitte; kuvatavate ridade arv lehel; url GET parameetri nimi, mille järgi kuvatakse kasutaja valitud leht; raporti veergude haldamise väljad (veergude haldamine on kirjeldatud kasutusjuhus „Halda regiooni veerge“ [19]) ning leht, kuhu kasutaja peab suunama konkreetse olemi juures raportis nupu vajutamine. Kui on valitud, et regioonile tuleb lisada nupp olemi lisamiseks, siis kuvatakse arendajale lisaks kaks välja, kuhu tuleb kirjutada nupu silt ja valida leht, kuhu peab nupu vajutamine kasutaja suunama.

Kasutusjuht: Halda vormi alamregiooni

Tegutsejad: Arendaja

Kirjeldus: Vormiregiooni haldamisel saab arendaja lisada vormile alamvormi või alamtabelivormi, mis on seotud peavormiga selle eeltäitmise väärtuste põhjal. Seega alamregioone saab vormile lisada ainult siis, kui vormil on olemas eeltäitmine. Kui vormile määratakse eeltäitmine, siis aktiveerub nupp alamregioonide lisamiseks, millele vajutades avaneb selle kõrval kontekstimenüü, kust saab valida, millist tüüpi alamregioon lisada. Vormi alamregiooni lisamisel või muutmisel peab arendaja sisestama alamregiooni nime ja järjekorranumbri. Järgnevad tegevused sõltuvad alamregiooni tüübist, mis on vastavalt kirjeldatud kasutusjuhtudes „Halda alamvormi“ ja „Halda alamtabelivormi“.

Kastutusjuht: Halda alamvormi

Tegutsejad: Arendaja

Kirjeldus: Alamvormi haldamisel peab arendaja sisestama vormi kinnitamisnupu sildi; õnnestumise- ja veateate teksti ning valima funktsiooni, mille kinnitamisnupu vajutamine välja kutsub. “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 kogustuslik 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.” [18] Samuti saab iga välja juures valida, millise peavormi eetäitmise veeru väärtusega seda täidetakse. Lisaks saab arendaja määrata vormivälja laiust pikslites või protsendites – välja arvatud märkeruudu ja raadionupu puhul. Kui välja tüüp on tekstväli e *textarea*, siis saab ka määrata välja kõrguse pikslites.

Kasutusjuht: Halda alamtabelivormi

Tegutsejad: Arendaja

Kirjeldus: Alamtabelivormi haldamisel peab arendaja esialgu valima vaate, mille põhjal täidetakse vorm andmetega ja samuti tegema valiku, kas vormil on viide teisele lehele või mitte. Kui vaade on valitud, siis saab määrata vormiga seotud veerud ja lisada või muuta vormi nuppe ning veerge. Kui on valitud, et vormil on viide teisele lehele, siis peab arendaja valima unikaalset identifikaatorit sisaldava veeru nime (kui kasutaja navigeerib uuele lehele kasutades viidet konkreetse olemi juures alamtabelivormis, siis sellest loetud väärtus pannakse kaasa url GET parameetriga) ja lehe, millele viidatakse. Seotud veerud määratakse nii, et kuvatakse nimekiri peavormi eeltäitmisveergudest, mis on valitud peavormi eeltäitmise päringu tingimuseks ning iga veeru kohta peab valima, mis alamtabelivormi veerg vastab sellele veerule. Näiteks saab teha nii, et alamtabelivormis kuvataks andmeid konkreetse olemi kohta, mille andmetega on eeltäidetud peavorm. “Nuppude lisamise või muutmisel valib arendaja nupu malli, funktsiooni, sisestab nupu kuvamise järjekorra, nupu teksti ning funktsiooni täitmise õnnestumise ning ebaõnnestumise teated.” [19] Kui funktsioon on valitud, siis kuvatakse selle parameetrite nimekiri ja arendaja peab valima igale parameetrile veeru, millest loetud väärtus antakse funktsiooni väljakutsumisel selle parameetri väärtuseks e argumendiks. Veergude haldamine on kirjeldatud kasutusjuhus „Halda regiooni veerge“. [19]

Lisaks muudeti kasutusjuhtu “Halda tabelivormi regiooni” [19]. Muudatus seisneb selles, et nüüd saab arendaja valida, millises veerus on *xmin* väärtus ehk rea versioon ning nuppude lisamisel või muutmisel saab valida, kas seda väärtust kasutatakse funktsiooni väljakutsumisel teise argumendina või mitte. Sellest sõltub, kas süsteemse

muutuja APP_USER väärtust antakse funktsioonile ette teise või kolmanda argumendina.

5.2 Muudatused kontseptuaalses andmemudelis

Järgnevalt esitletakse muudatused kontseptuaalses andmemudelis. Kirjeldus sisaldab uuendatud olemi-suhte diagramme ja nendel kujutatud olemitüüpide ja atribuutide definitsioone.

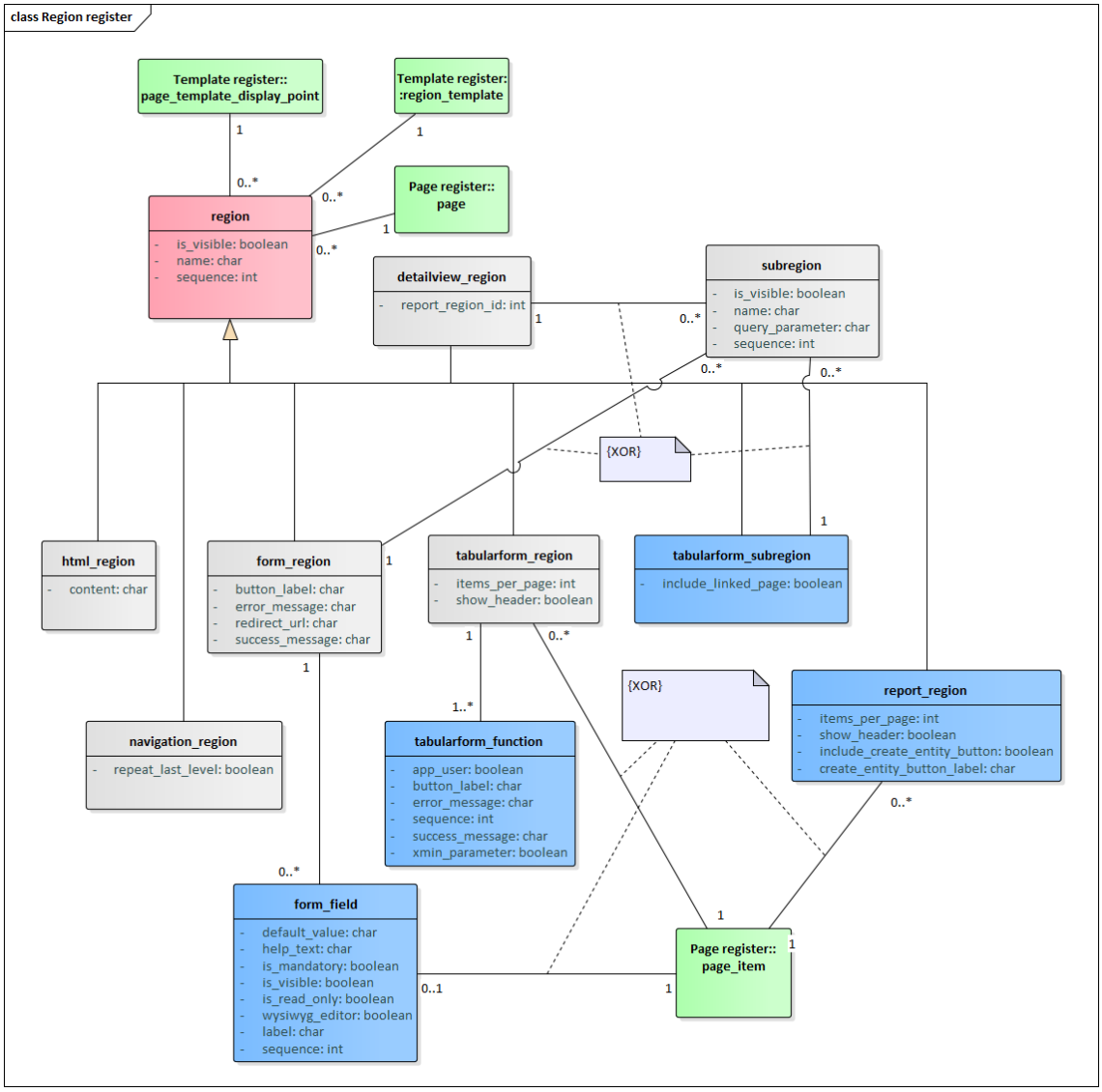
Värvide tähendus olemi-suhte diagrammides on järgmine.

- Punasega on märgitud registri põhiolemitüübid.
- Rohelisega on märgitud olemitüübid, mis ei kuulu diagrammiga kirjeldatavasse registrisse, kuid on seotud mõne selle registri olemitüübiga.
- Sinisega on märgitud registrisse kuuluvad olemitüübid, mis lisandusid pgApexi kolmanda versiooni arenduse tulemusena või muudetud olemasolevad olemitüübid.
- Halliga on märgitud registrisse kuuluvad olemitüübid, mis eksisteerisid juba eelmistes pgApex'i versioonides ja mida ei muudetud pgApexi kolmanda versiooni arenduse käigus.

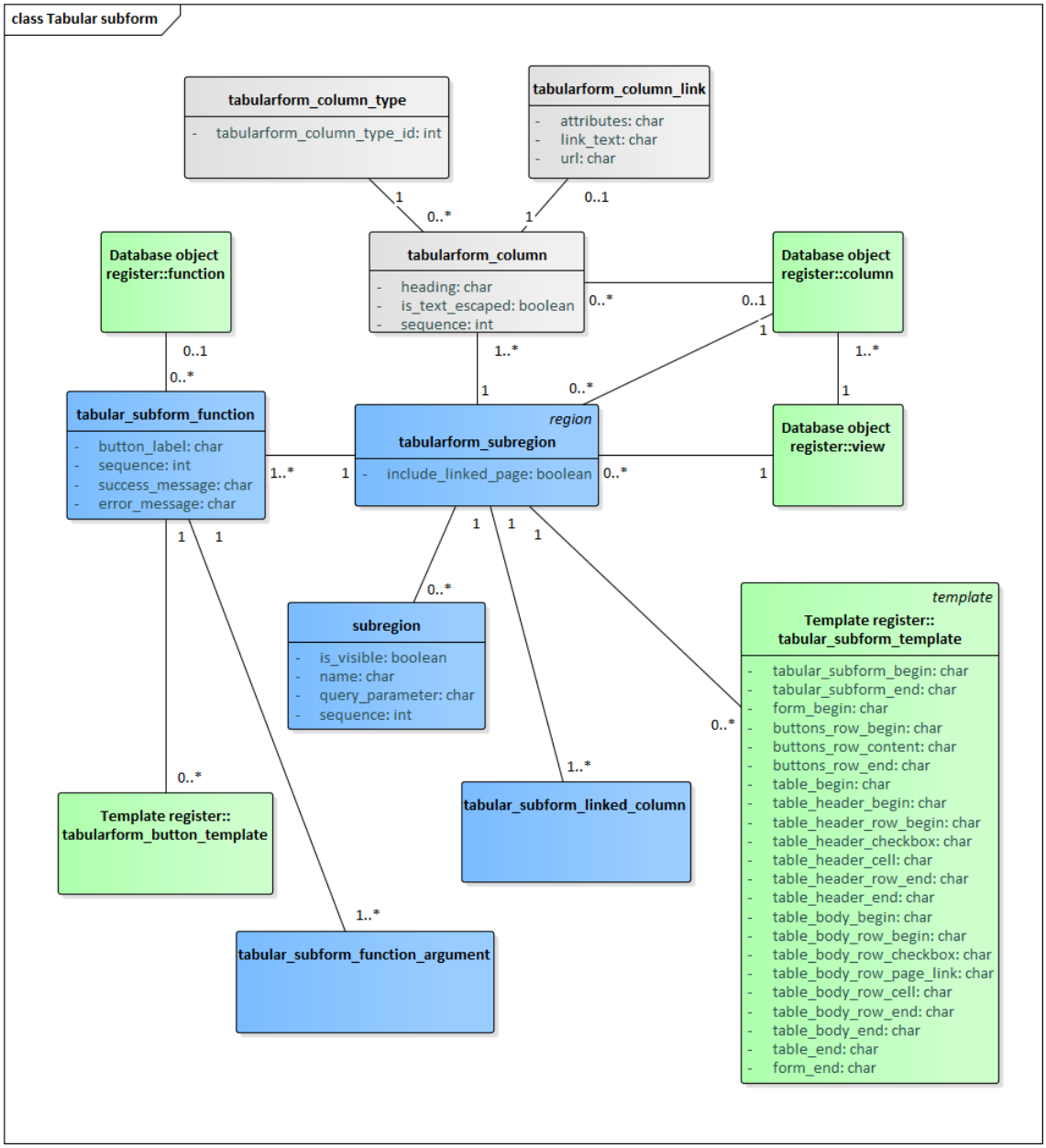
Joonis 1, Joonis 2 ja Joonis 3 esitatakse regioonide- ning Joonis 4 mallide registri olemi-suhte diagrammid. Diagrammid ei kirjelda registreid kogu ulatuses, vaid kajastavad põhiliselt pgApexi kolmandas versioonis tehtud muudatusi.

Tabel 6 esitatakse pgApex'i kolmandas versioonis lisandunud olemitüüpide sõnalisi kirjeldusi ja nende kuuluvust registritesse.

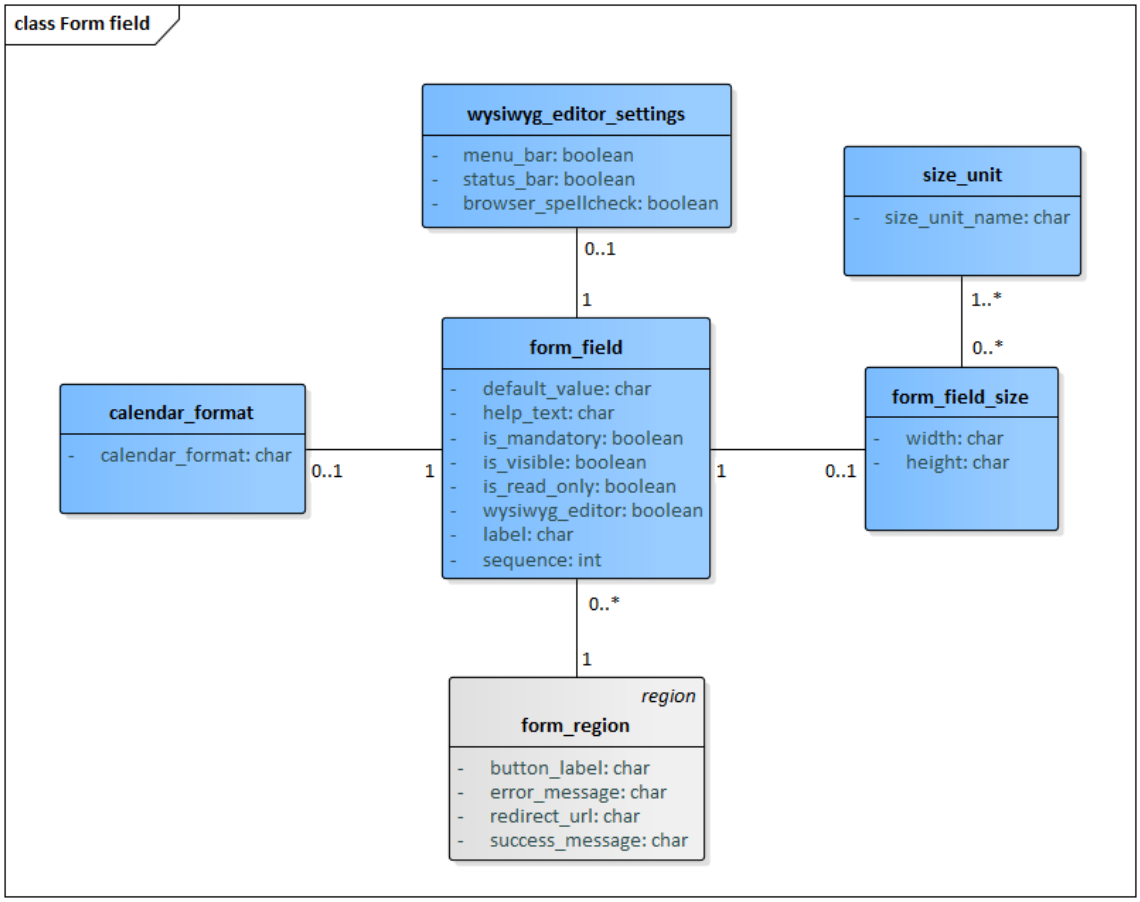
Tabel 7 esitatakse eelmainutud olemite atribuutide sõnalisi kirjeldusi. Lisaks nendele kirjeldatakse ka olemasolevatele olemitele lisandunud atribuute. Nad on märgitud sinisega.



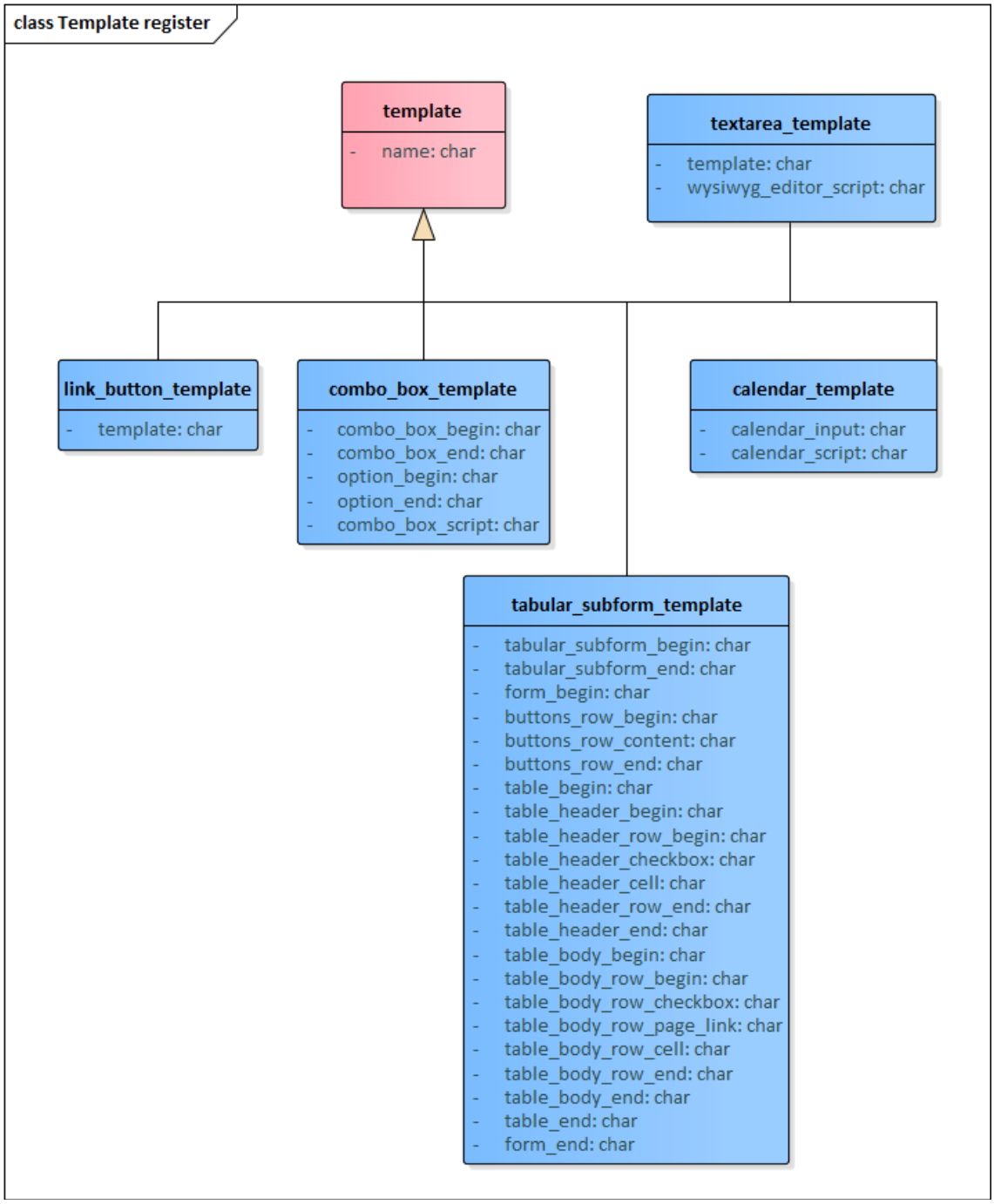
Joonis 1.Regioonide registri olemi-suhte diagramm.



Joonis 2. Regioonide registri olemi-suhte diagramm. Alamtabelivorm.



Joonis 3. Regioonide registri olemi-suhte diagramm. Vormiväli.



Joonis 4. Mallide registri olemi-suhte diagramm.

Tabel 6. Olemitüüpide kirjeldused.

Olemitüübi nimi	**Kuuluvus registrisse**	**Definitsioon**
tabularform_subregion	Regioonide register	Alamtabelivormi regioon.
tabular_subform_function	Regioonide	Alamtabelivormi funktsioon.

Olemitüübi nimi	Kuuluvus registrisse	Definitsioon
	register	
tabular_subform_function_argument	Regioonide register	Alamtabelivormi funktsiooni argument. Seos alamtabelivormi funktsiooni parameetri ja vaate veeru vahel.
tabular_subform_linked_column	Regioonide register	Seos alamtabelivormi ja peavormi vahel vaate veergude abil.
calendar_format	Regioonide register	Kalendri tüübi vormivälja formaat.
form_field_size	Regioonide register	Konkreetse vormivälja suurus (kõrgus ja laius) ning nende ühikud.
size_unit	Regioonide register	Elemendi (nt vormivälja) suuruse ühik.
wysiwyg_editor_settings	Regioonide register	WYSIWYG tekstiredaktori seadistused konkreetse vormivälja kohta.
link_button_template	Mallide register	Link-nupu mall. Link-nupp on nupu väljanägemisega HTML "<a>" märgend.
combo_box_template	Mallide register	Liitboksi mall.
calender_template	Mallide register	Kalendri mall.
tabular_subform_template	Mallide register	Alamtabelivormi mall.

Tabel 7. Atribuutide kirjeldused.

Olemitüübi nimi	**Atribuudi nimi**	**Atribuudi definitsioon**
tabularform_subregion	include_linked_page	Kas alamtabelivormis on link olemi andmetega teisele lehele (peamiselt andmete muutmise lehele) või mitte?
tabular_subform_function	button_label	Alamtabelivormi nupu tekst.
tabular_subform_function	sequence	Alamtabelivormi nupu järjekorranumber.
tabular_subform_function	success_message	Alamtabelivormi funktsiooni käivitamise õnnestumisel kuvatav tekst.
tabular_subform_function	error_message	Alamtabelivormi funktsiooni käivitamise ebaõnnestumisel kuvatav tekst.
calendar_format	calendar_format	Kalendri kuupäeva ja aja formaat tekstivormis. Näiteks, “%Y-%m-%d %H:%i:%s”, kus on järjest „-“ märgiga ühendatud aasta, kuu, päev ning „:“ märgiga järjest ühendatud tunnid, minutid ja sekundid. Kuupäev ja aeg on eraldatud tühikuga.
form_field_size	width	Vormivälja laius.
form_field_size	height	Vormivälja kõrgus.

Olemitüübi nimi	**Atribuudi nimi**	**Atribuudi definitsioon**
size_unit	size_unit_name	Suuruse ühiku nimetus.
wysiwyg_editor_settings	menu_bar	Kas näidata kasutajale tekstiredaktori ülaservas menüüriba või mitte.
wysiwyg_editor_settings	status_bar	Kas näidata kasutajale tekstiredaktori allservas asuvat staatuse riba või mitte.
wysiwyg_editor_settings	browser_spellcheck	Kas kasutada veebilehitseja poolset grammatika kontrolli või mitte.
link_button_template	template	Link-nuppu kirjeldav HTML märgend.
combo_box_template	combo_box_begin	Liitboksi algus ehk "<select>" HTML märgend.
combo_box_template	combo_box_end	Liitboksi lõpp ehk "</select>" HTML märgend.
combo_box_template	option_begin	Liitboksi valiku algus ehk "<option>" HTML märgend.
combo_box_template	option_end	Liitboksi valiku lõpp ehk "</option>" HTML märgend.
combo_box_template	combo_box_script	Liitboksi töötamiseks vajalik JavaScript keele skript, mis on kirjutatud "<script></script>" HTML märgendite paari vahele.

Olemitüübi nimi	**Atribuudi nimi**	**Atribuudi definitsioon**
calendar_template	calendar_input	Teksti tüübi vormivälja "<input>" HTML märgend, mis muudetakse skripti abil kalendriks.
calendar_template	calendar_script	Kalendrivaliku töötamiseks vajalik JavaScript keele skript, mis on kirjutatud "<script></script" HTML märgendite paari vahele.
tabular_subform_template	tabular_subform_begin	Alamtabelivormi algus ehk " " HTML märgend.
tabular_subform_template	tabular_subform_end	Alamtabelivormi lõpp ehk " " HTML märgend.
tabular_subform_template	form_begin	Alamtabelivormi vormi osa algus ehk "<form>" HTML märgend.
tabular_subform_template	buttons_row_begin	Alamtabelivormi nuppude osa algus ehk " " HTML märgend.
tabular_subform_template	buttons_row_content	Alamtabelivormi nupud. Nende mallide HTML märgendid võetakse "tabularform_button_ template" olemitüübile vastavast andmestruktuurist vastavalt arendaja vaates valitud nupu mallile.

Olemitüübi nimi	**Atribuudi nimi**	**Atribuudi definitsioon**
tabular_subform_template	buttons_row_end	Alamtabelivormi nuppude osa lõpp ehk " " HTML märgend.
tabular_subform_template	table_begin	Alamtabelivormi tabeli osa algus ehk "<table>" HTML märgend koos selle ümbrise algusega " " HTML märgendi kujul.
tabular_subform_template	table_header_begin	Alamtabelivormi tabeli päise algus ehk "<thead>" HTML märgend.
tabular_subform_template	table_header_row_begin	Alamtabelivormi tabeli päise rea algus ehk "<tr>" HTML märgend.
tabular_subform_template	table_header_checkbox	Alemtabelivormi tabeli päise rea märkeruudu osa ehk "<th></th>" HTML märgend, mille sees on märkeruudu tüübi "<input/>" märgend.
tabular_subform_template	table_header_cell	"<th></th>" HTML märgend, mis sisaldab arendaja vaates määratud alamtabelivormi veeru nime.
tabular_subform_template	table_header_row_end	Alamtabelivormi tabeli päise rea lõpp ehk "</tr>" HTML märgend.

Olemitüübi nimi	**Atribuudi nimi**	**Atribuudi definitsioon**
tabular_subform_template	table_header_end	Alamtabelivormi tabeli päise lõpp ehk "</thead>" HTML märgend.
tabular_subform_template	table_body_begin	Alamtabelivormi kehandi algus ehk "<tbody>" HTML märgend.
tabular_subform_template	table_body_row_begin	Alamtabelivormi kehandi rea algus ehk "<tr>" HTML märgend.
tabular_subform_template	table_body_row_ checkbox	Alemtabelivormi tabeli kehandi rea märkeruudu osa ehk "<th></th>" HTML märgend, mille sees on märkeruudu tüübi "<input/>" märgend.
tabular_subform_template	table_body_row_page_ link	Alamtabelivormi kehandi rea teisele lehele viitava lingi osa, mis koosneb "<th></th>" HTML märgistest, mille sees on lingi sisaldav "<a>" märgis koos lingi ikooni sisaldava "" märgisega.
tabular_subform_template	table_body_row_cell	"<th></th>" HTML märgend, mis sisaldab arendaja vaates määratud alamtabelivormi veeru sisu, mis on kas väärtus vastavast vaate veerust või

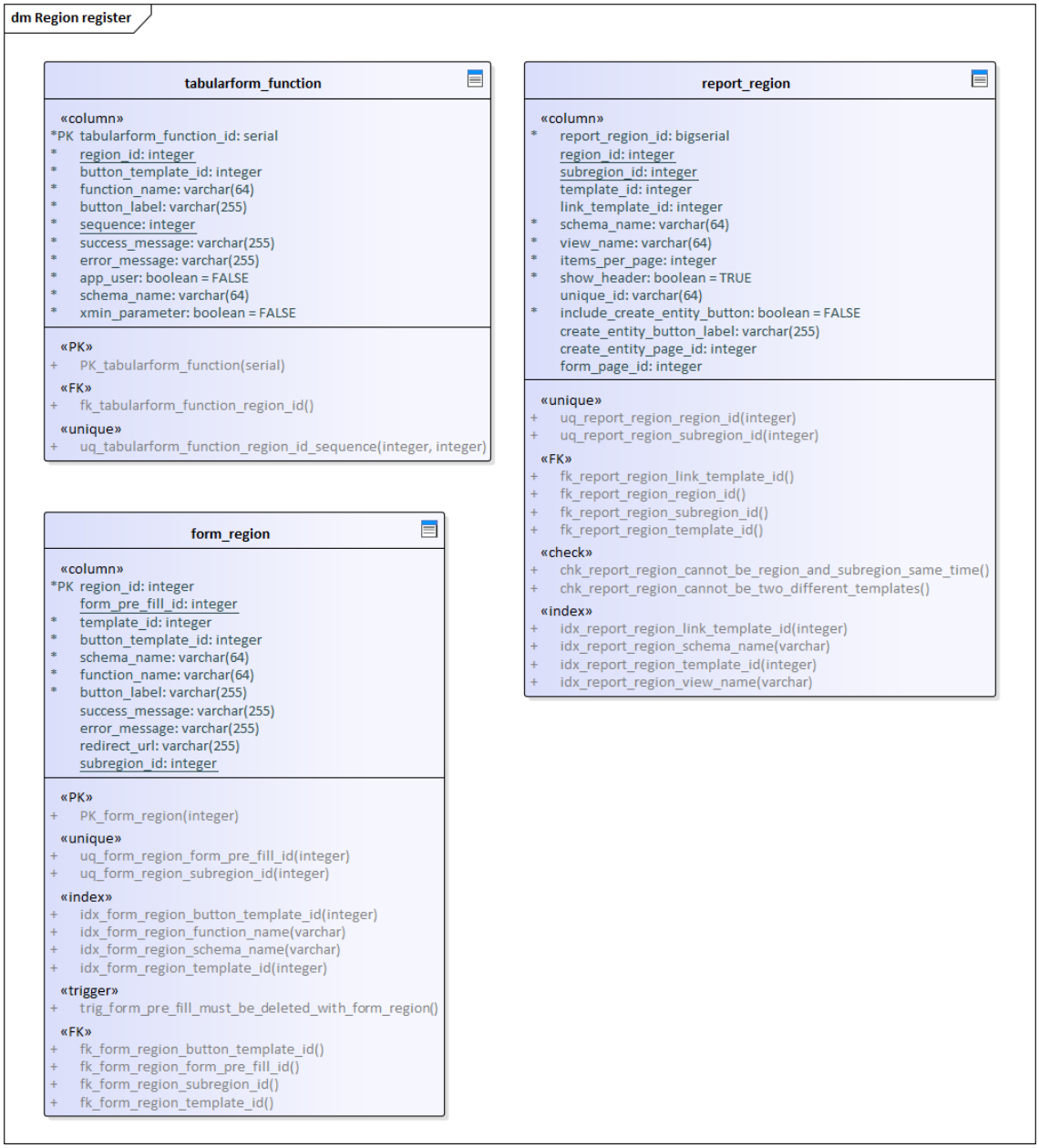
Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon
		link.
tabular_subform_template	table_body_row_end	Alamtabelivormi tabeli kehandi rea lõpp ehk "</tr>" HTML märgend.
tabular_subform_template	table_body_end	Alamtabelivormi tabeli kehandi lõpp ehk "</tbody>" HTML märgend.
tabular_subform_template	table_end	Alamtabelivormi tabeli osa lõpp ehk "<table>" HTML märgend.
tabular_subform_template	form_end	Alamtabelivormi vormi osa lõpp ehk "</form>" HTML märgend.
form_field	is_read_only	Kas vormivälja väärtus on ainult lugemiseks (ei saa muuta) või mitte?
form_field	wysiwyg_editor	Kas suurema tekstivälja e *textarea* tüübi vormivälja puhul kasutada WYSIWYG tekstiredaktorit või mitte?
tabularform_function	xmin_parameter	Kas anda tabelivormi funktsioonile teise parameetri argumendina ette *xmin* veeru väärtus (rea versioon) või mitte?

Olemitüübi nimi	Atribuudi nimi	Atribuudi definitsioon
report_region	include_create_entity_ button	Kas lisada raportisse nupp olemi lisamiseks või mitte?
report_region	create_entity_button_ label	Raportis oleva olemi lisamise nupu tekst.
textarea_template	wysiwyg_editor_script	WYSIWYG tekstiredaktori töötamiseks vajalik JavaScript keele skript, mis on kirjutatud "<script></script>" HTML märgendite paari vahele.

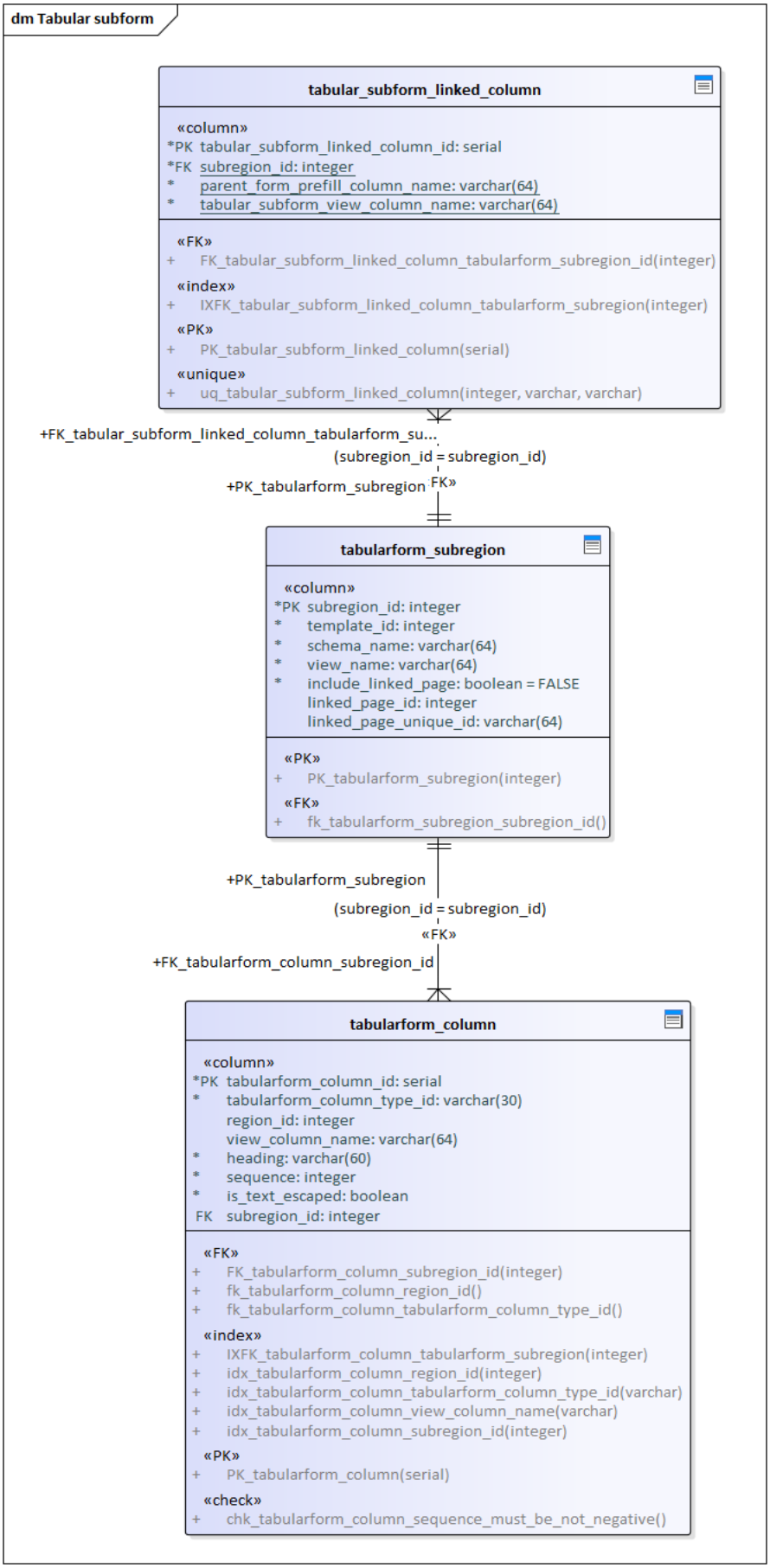
5.3 Muudatused füüsilises disainis

Järgnevalt esitatakse muudatused andmebaasi füüsilises disainis. Füüsilise disaini kirjeldus sisaldab andmebaasi disaini diagramme, mis kujutavad uusi ja muudetud andmebaasi tabeleid. Diagrammidel kajastatakse ainult neid tabeleid, mis loodi või mida muudeti pgApexi kolmanda versiooni arendamise käigus. Statistika andmebaasis tehtud muudatuste kohta esitatakse jaotises 7.1.

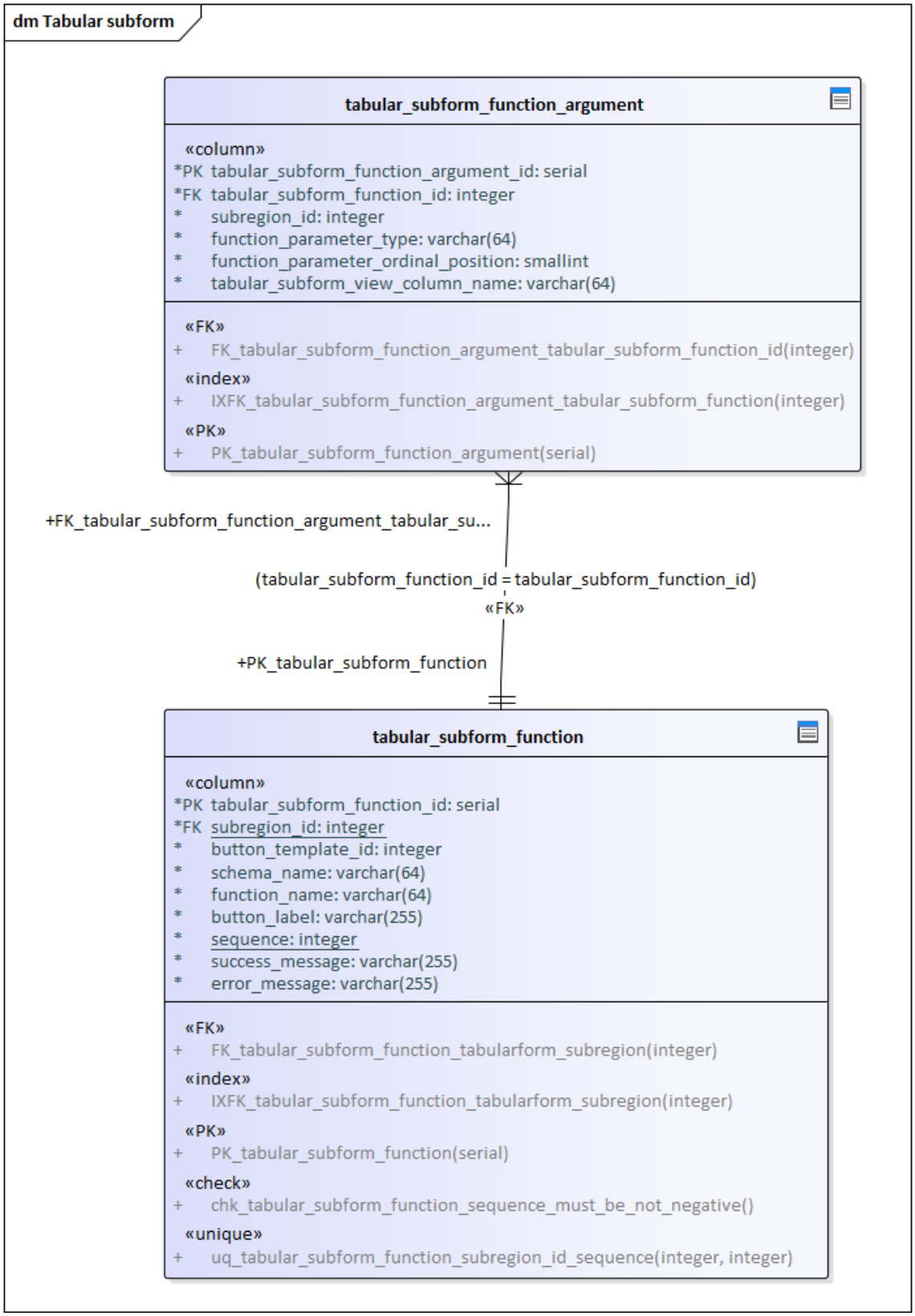
Joonis 5, Joonis 6, Joonis 7, Joonis 8 ja Joonis 9 on esitatud regioonide- ning Joonis 10 mallide registri füüsilise disaini andmebaasidiagrammid.



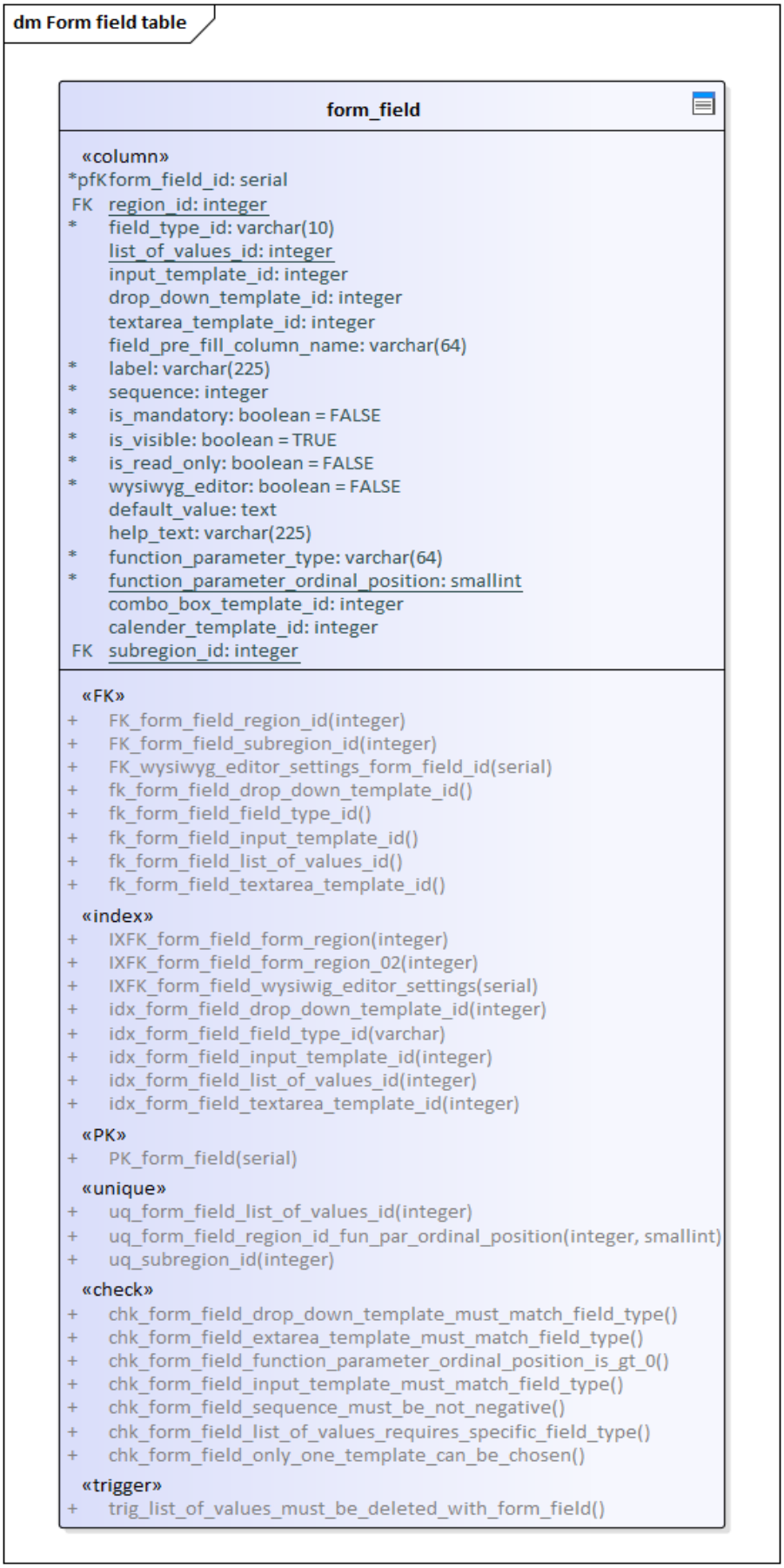
Joonis 5. Regioonide registri füüsilise disaini andmebaasi diagramm. „tabularform_function“, „report_region“, ja „form_region“ tabelid.



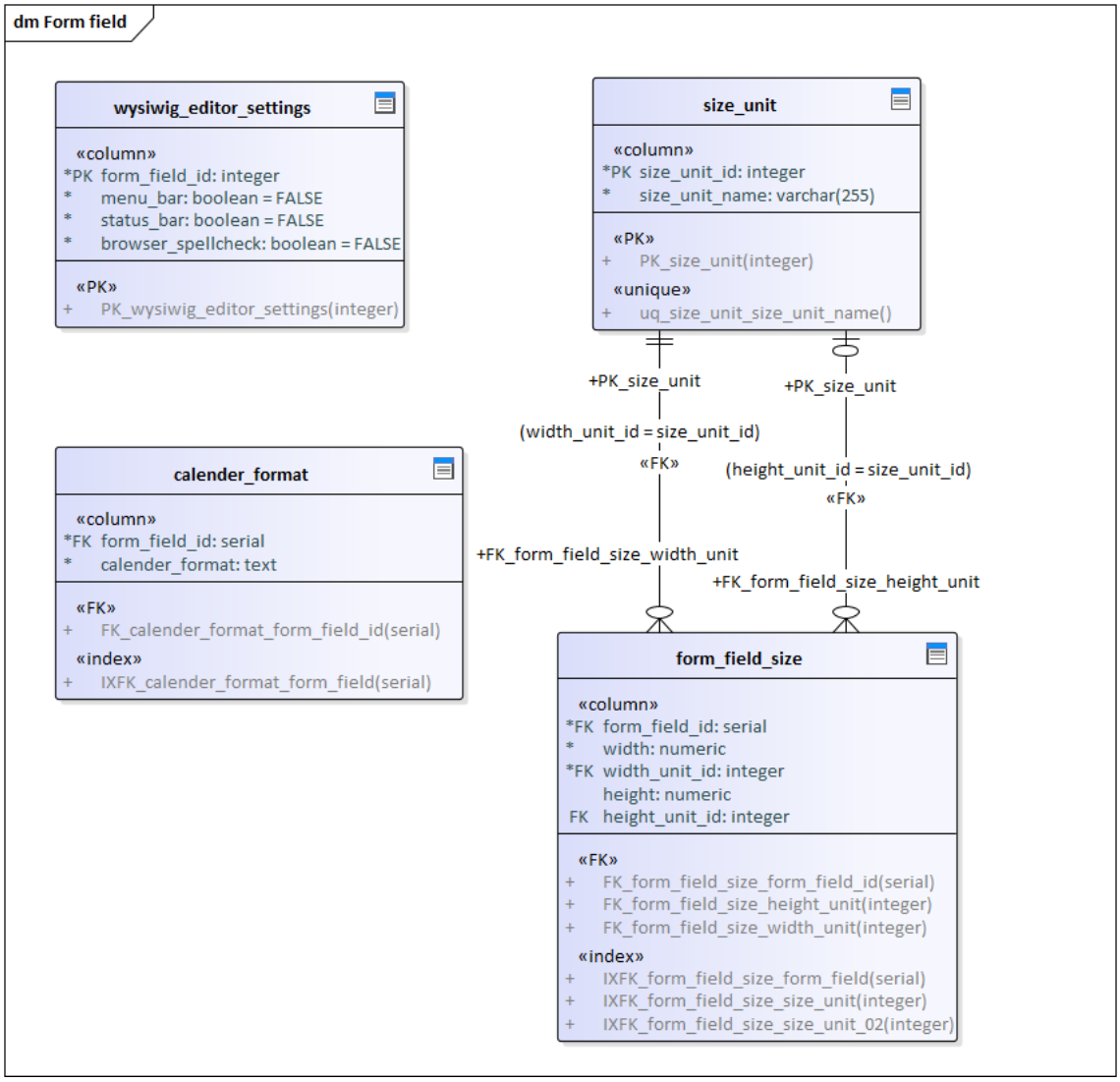
Joonis 6. Regioonide registri andmebaasidiagramm. Alamtabelivorm.



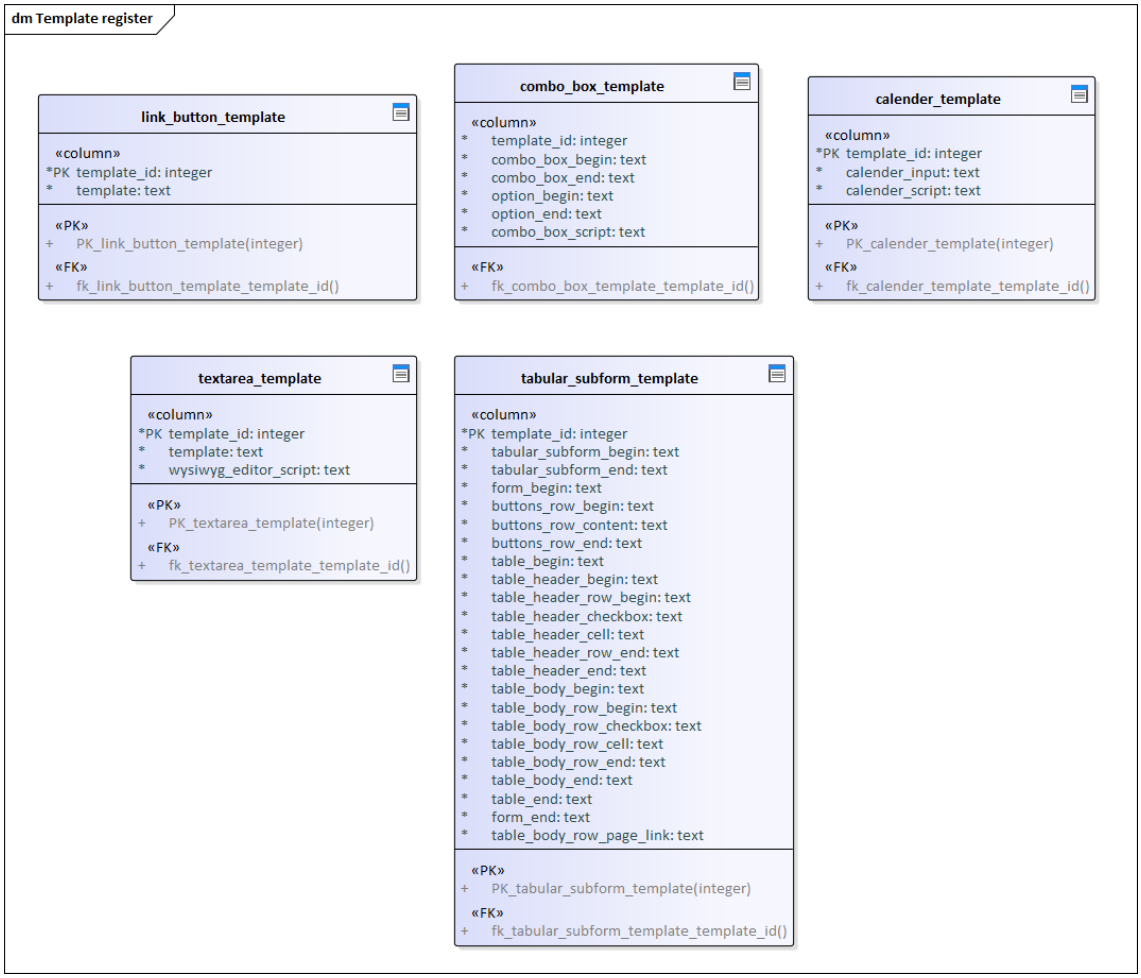
Joonis 7. Regioonide registri andmebaasidiagramm. Alamtabelivormi funktsioon.



Joonis 8. Regioonide registri andmebaasidiagramm. Vormiväli



Joonis 9. Regioonide registri füüsilise disaini andmebaasi diagramm. Vormiväljaga seotud tabelid.



Joonis 10. Mallide registri füüsilise disaini andmebaasi diagramm.

6 Arendus

Järgnevalt kirjeldatakse arenduse tulemusi tehnilisemas vaates. Kirjutatakse, mil viisil realiseeriti pgApex'i kolmanda versiooni uue funktsionaalsuse osad ning tehti muudatusi ja parandusi olemasolevas funktsionaalsuses.

6.1 Andmete lisamine ja muutmine

Alates esimesest versioonist on pgApex'is võimalus luua vormiregioone, mis on seotud andmebaasis loodud funktsioonidega. See tähendab, et vorm toimib nagu liides mingi andmebaasi funktsiooni kasutamiseks ehk vormi kinnitamise nupu vajutamisel antakse selle väljade väärtused funktsioonile argumentidena ette ja käivitatakse see funktsioon.

Vormiregiooni olemasolev funktsionaalsus võimaldas realiseerida olemi andmete lisamist, aga kasutajaliideses oleks see nagu eraldiseisev regioon. Seega oli see vaja integreerida mõne olemasoleva või uue regiooni tüübiga ning tooteomanikuga (juhendajaga) arutamise tulemusena otsustati, et raporti regiooni tüüpi tehakse võimalus lisada nupp, millele vajutamisel avaneb mingi eelnevalt loodud leht. Antud juhul oleks see leht, kus on olemi andmete lisamise vorm. Raporti haldamise vormi arendaja vaatesse lisati märkeruut, millel klõpsamine määrab, kas raportil näidatakse seda nuppu või mitte. Kui märkeruut täidetakse, siis ilmub väli, kus saab valida olemi andmete lisamise lehe, mille avab nupu vajutamine.

Olemi andmete muutmise võimaldamiseks lisati vormiregioonile võimalus selle eeltäitmiseks. Näiteks kui klõpsatakse radade raportis konkreetse raja muutmise nupule, siis peab avanema uus leht, milles olevasse vormi on laaditud selle raja andmed. Selleks on vaja valida vaade, millest need andmed tulevad ja iga vormivälja kohta tuleb määrata veerg, milles olevat väärtust see väli kuvab. Samuti on vaja valida veerg mille alusel moodustatud WHERE klausli abil leitakse vaatest rida, mida selles vormis on vaja kuvada. Eelneva eesmärk seisneb selles, et vormi avamisel poleks väljad tühjad, vaid oleksid täidetud konkreetse olemi andmetega, mis on võetud vaatest. Seega sobib selline funktsionaalsus olemi andmete muutmiseks, aga see on samuti vaja integreerida mingi regiooni tüübiga. Pärast tooteomanikuga (juhendajaga) arutamist tekkis vastav kasutuslugu (ID 2) (vt Tabel 5), mille järgi tuli realiseerida uut tüüpi raporti regioon – „raport ja vorm", kus iga esitatava olemi juures on nupp, millele vajutamisel avaneb

vastav olemi muutmise vorm. Selle põhimõte on sarnane “raport ja detailvaade” regiooniga. Tuleb valida vaade raporti jaoks, kirjeldada raporti veerge, valida olemi unikaalset identifikaatorit sisaldav veerg ja lõpuks valida eelnevalt loodud leht, kus on andmete muutmise vorm. Raportis konkreetse olemi juures nupu vajutamisel suunatakse kasutaja arendaja vaates määratud lehele. Lingis on unikaalne identifikaator GET parameetrina. See on vajalik andmete lugemiseks, et vormi eeltäita (et avaneks leht, milles olevasse vormi on selle olemi andmed laaditud). Näiteks kui vaate *koik_rajad* põhjal loodud raportis algatatakse raja, mille kood on 2, muutmine, siis avaneb leht, millel olevasse vormi laaditakse Joonis 11 oleva päringuga leitud andmed.

```
SELECT * FROM koik_rajad WHERE raja_kood=2;
```

Joonis 11. SQL päring raja koodiga 2 andmete pärimiseks *koik_rajad* vaatest.

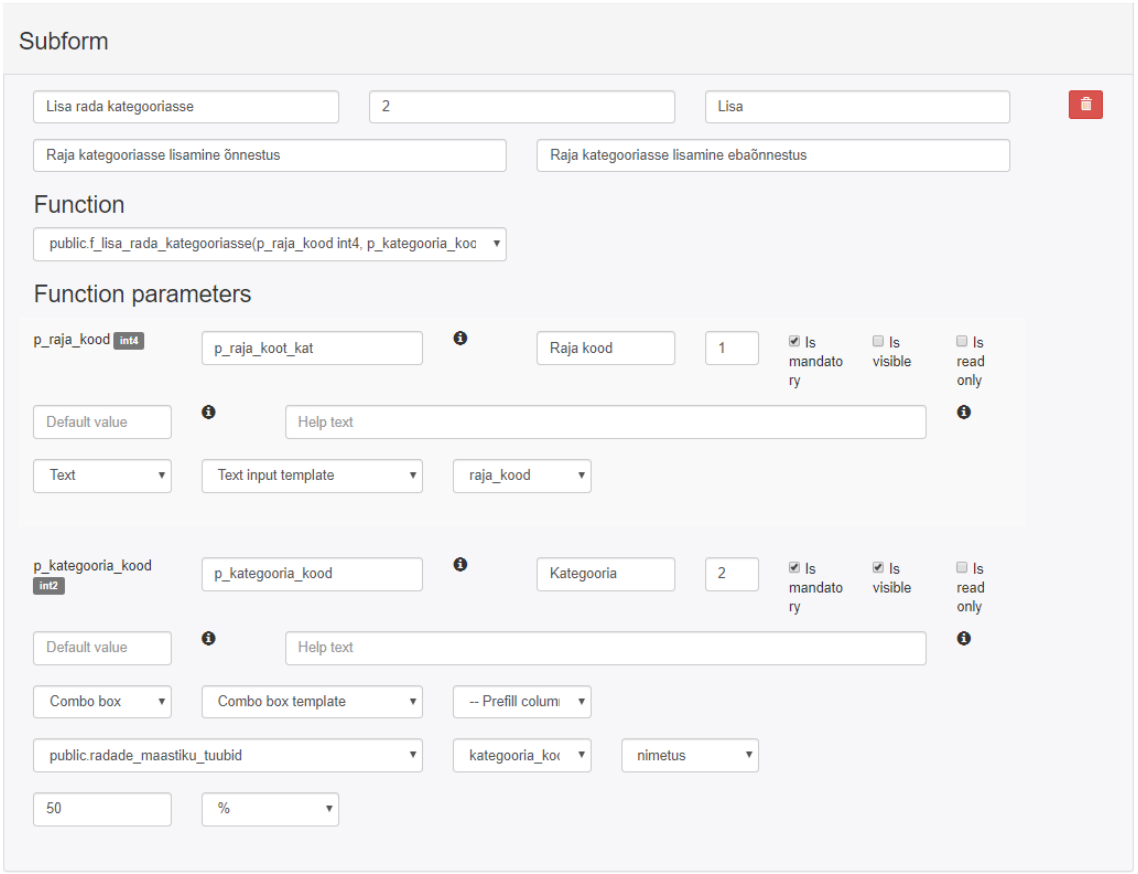
6.2 Alamvormid

Et saaks realiseerida olemiga 1:M või 1:0..1 seosetüübi kaudu seotud olemite andmete lisamist, lugemist, muutmist ja kustutamist, on vaja tekitada arendajale võimalus lisada vormiregioonile alamregioone (alamvorme) nagu see on tehtud detailvaate regioonis, kus on võimalik lisada alamraporteid.

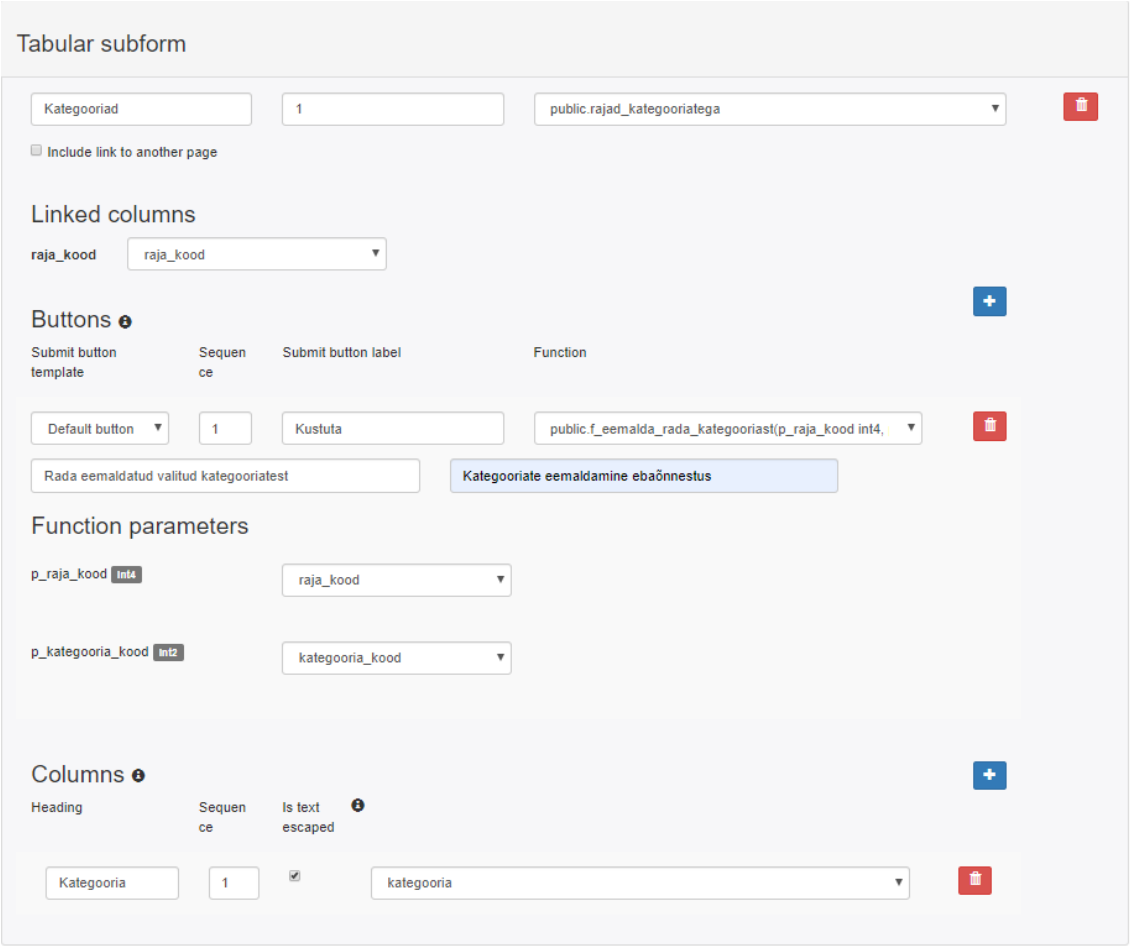
Esialgse plaanina oli kavas realiseerida üks alamvormi tüüp, kus samas alamregioonis oleks võimalik andmeid lisada, lugeda, muuta ja kustutada. Hiljem seda lahendust analüüsides aga jõuti järeldusele, et selline alamregioon oleks liiga suur ja seda oleks raske hallata. Samuti poleks võimalik realiseerida ainult valikulisi operatsioone (näiteks ainult andmete lisamist ja lugemist).

Otsustati, et andmete lisamist saab realiseerida tavalise vormiga. Seega realiseeriti alamvorm, mis on sama, mis vormiregioon, aga asub alamregioonis ja veerg, mille alsusel siduda põhivorm alamvormiga (et alamvormis näidatakse ainult põhivormis oleva olemiga seoutd andmeid) valitakse peavormi eeltäitmise vaatest. Niimoodi saab alamvormi seostada peavormiga näiteks vaates oleva olemi unikaalse identifikaatori põhjal. Näiteks peavormis on raja andmed ning selle alamvormis on selle konkreetse rajaga seotud kategooria omamised. Joonis 12 on alamtabelivormi haldusvorm arendaja vaates.

Andmete lugemist ja kustutamist realiseerib hästi tabelivorm, mis lisandus pgApexisse Nikolai Kopa [19] töö tulemusena. Seega nende operatsioonide jaoks loodi alamtabelivorm, aga see on olemasolevast tabelivormist erinev. Nimelt, tavalises tabelivormis on võimalik anda funktsioonile ette maksimaalselt kaks (kolmandas versioonis kolm) kindlat argumendi – olemi unikaalne identifikaator ja sisselogitud kasutaja kasutajanimi (APP_USER süsteemse muutuja väärtus [19]). Kolmandas versioonis lisandus võimalus anda argumendina ette *xmin* veeru väärtus (rea versioon). Alamtabelivormi lahendus on universaalsem: arendajale kuvatakse funktsiooni parameetrite nimekiri ja iga parameetri juures saab ta valida, millise veeru väärtus antakse funktsioonile ette selle parameetri väärtusena. Seega on argumentide arv piiratud ainult funktsiooni parameetrite arvuga nagu vormiregioonil. Joonis 13 on alamtabelivormi haldusvorm arendaja vaates.



Joonis 12. pgApex'i arendaja vaate. Alamvormi haldamine.



Joonis 13. pgApex'i arendaja vaate. Alamtabelivormi haldamine.

6.3 Uued vormiväljade tüübid ja nende seadistamine

Antud töö plaanitud realiseeritava funktsionaalsuse hulka kuulus ka uute vormivälja tüüpide lisamine vormiregioonile.

Välja tüübid, mis olid juba pgApexis olemas.

- Tekst
- Parool
- Raadionupud
- Märkeruut
- Rippmenüü e *dropdown*
- Suurem tekstiväli e *textarea*

pgApexi kolmandas versioonis lisandusid järgmised väljatüübid.

- Kalender
- Liitboks

6.3.1 Kalender

Kalendrivalik on väga vajalik vormiväli, kuna andmebaasides on tihti vaja hoida ajalisi (temporaalseid) andmeid. Eelmistes pgApexi versioonides tuli kasutajal ajalisi väärtusi sisestada tavatekstina, mis ei ole väga mugav ning kasutaja ei pruugi alati teada, millises formaadis tuleb neid sisestada.

Kalendrivaliku välja realiseerimiseks uuriti võimalust kasutada nii HTML keele sisse ehitatud *date* tüübi vormivälja, kui ka väliseid lahendusi [39] [40] [41] [42]. Lõpuks valiti kalendri väljatüübi jaoks avatud lähtekoodiga lahendus nimega AnyTime [39]. Valiku põhjenduseks on see, et antud lahendusel on kõige paindlikumad häälestamisvõimalused ning seda oli lihtne pgApex'iga ja PostgreSQL andmebaasisüsteemi kuupäeva andmetüübiga integreerida. Nagu ka pgApex'il, on AnyTime lahendusel MIT litsents ja seega need sobivad kokku ning selle lahenduse integreerimine pgApex'isse on lubatud.

6.3.2 Liitboks

Liitboksi põhimõte on sama, mis rippmenüül: vormivälja täitmiseks saab kasutaja valida nimekirjast mingi eeldefineeritud väärtuse. Erinevus on selles, et liitboksi on sisse ehitatud otsing e filtreering. Teiste sõnadega saab kasutaja sisestada mingi teksti käsitsi ning talle kuvatakse nimekirjast need väärtused, mis sisaldavad sisestatud teksti. See on kasulik olukordades, kus eeldefineeritud väärtusi on palju ja nende kerimine võtaks palju aega.

Liitboksiks kasutatakse ning kohandati pgApexi jaoks JQuery UI lahendust [43].

6.3.3 Vormiväljade seadistamine

Eelmistes pgApexi versioonides olid kõik vormiregiooni väljad ühesuguse laiusega, mis oli 100% regiooni laiusest. Nii laiad väljad pole alati vajalikud. Näiteks, kui on mingi väli, kuhu peab sisestama numbrilise väärtuse, siis tihti piisab väikesest laiusest. Muudatuse tulemusena lisati kõikidele vormiregiooni (sh alamregiooni) väljadele peale

märkeruudu ja raadionupu võimalus määrata laiust kõige levinumates veebidisainis ühikutes: protsendites ja pikslites. Lisaks suurema tekstivälja e *textarea* puhul lisati võimalus määrata kõrgust pikslites ja ridades.

Samuti realiseeriti võimalus teha vormiväli ainult lugemiseks. See tähendab, et väljas sisaldavat väärtust ei saa muuta. Sellise välja kasutuse näidiseks on olukord, kus andmebaasi tahetakse kirjutada mingi kindel, arendaja poolt vormi loomisel ettemääratud väärtus (näiteks sisseloginud kasutaja kasutajanimi), mida lõppkasutaja ei saa muuta, vaid saab ainult teatavaks võtta.

Kalendrivaliku puhul oli lisatud võimalus määrata kalendriformaati. Seda saab määrata tekstina, aga pidades kinni reeglitest, mis on kirjeldatud AnyTime [39] veebilehel ja mis on ka lisatud kalendriformaadiga välja juurde abitekstina.

Suurema tekstivälja e *textarea* puhul lisati võimalus kasutada seda tüüpi väljas WYSIWYG tekstiredaktorit. Esialgu uuriti, millist lahendust võiks kasutada [44] [45] [46] [47] ning lõpuks võeti kasutusele TinyMCE [44] lahendus, kuna seda on lihtne integreerida *textarea* tüübi vormiväljaga ning samuti pakub see kõige rohkem häälestamisvõimalusi. Sellel lahendusel on *GNU LESSER GENERAL PUBLIC* litsents [48] ning see ütleb: “5\. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.” [48] Seega on TinyMCE integreerimine pgApex keskkonda lubatud, st see ei muuda pgApexi levitamise tingimusi. TinyMCE häälestatavatest omadustest on pgApexi arendusvaate kaudu muudetavaks tehtud järgmised omadused.

- Valik, kas näidata kasutajale kasutajale redaktori ülaservas olevat menüüriba või mitte.
- Valik, kas näidata kasutajale redaktori alaservas asuvat staatuse riba või mitte.
- Valik, kas kasutada veebilehitseja poolset grammatika kontrolli või mitte.

Ülejäänud võimalikud omadused on kirjeldatud TinyMCE dokumentatsioonis [49].

6.4 Andmete samaaegne muutmine

Antud töö üks eesmärk oli tekitada võimalus luua rakendusi, kus on arvestatud andmete samaaegne muutmisega ja sellest tulenevate probleemide vältimiseks on realiseeritud ridade optimistlik lukustamine. See on oluline osa CRUD rakenduste funktsionaalsusest. Selleks tehti pgApexi funktsionaalsusesse mõned täiendused. Eeldatakse, et rea versiooni määramiseks kasutatakse andmebaasisüsteemi poolt värskendatavat *xmin* väärtust, mis on igas baastabelis olevas peidetud *xmin* veerus. *xmin* väärtuse kasutamise võimalus PostgreSQL andmebaasi kasutamisel optimistliku lukustamise realiseerimiseks on arendajatele juba varasemast tuttav [50], st see pole käesoleva töö leiutis.

6.4.1 Vormiregioonide kinnitamise loogikat muutmine

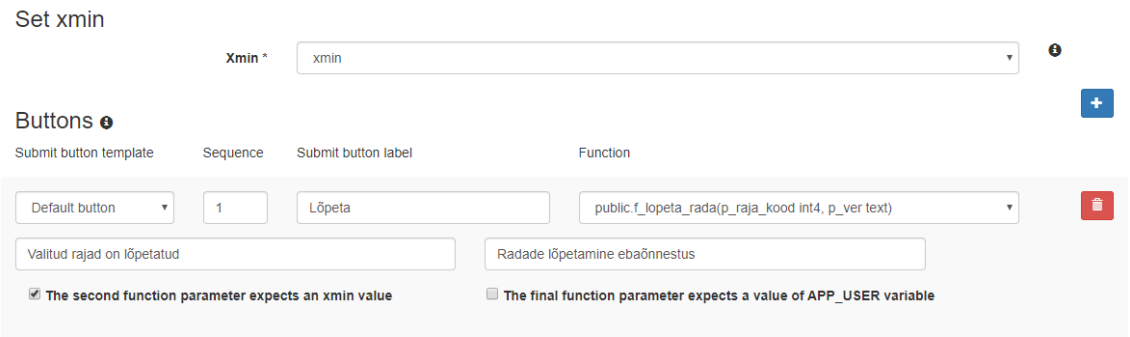
Vormiregioonile saab määrata teksti, mida kuvatakse kasutajale, kui funktsiooni käivitamine õnnestus ning samuti veateade teksti ebaõnnestumise korral. Eelmistes pgApex'i versioonides oli veateade lahendatud nii, et veateadet kuvatakse kasutajale ainult siis, kui funktsiooni käivitamisel tekib mingi erand e *exception.* Optimistlikku lukustamist realiseerivates andmebaasi funktsioonides kasutatakse rida uuendava lause tingimuses rea versiooni. Kui rida on peale viimast lugemist muutunud (sellel on teistsugune versiooni identifikaator kui funktsioonile etteantud versiooni identifikaator), siis ei muuda lause ühtegi rida ja erandit ei teki.

Kuna funktsioon ei tekita erandit, siis kasutajale kuvati õnnestumise teksti isegi siis, kui andmete muutmine ebaõnnestus, sest peale viimast andmete lugemist oli muudetavast reast tekkinud uus versioon – keegi oli juba vahepeal seda rida muutnud. Probleemi saanuks lahendada pgApex'i funktsionaalsust muutmata: olemi andmete uuendamise funktsiooni tuleks täiendada nii, et kui andmete uuendamise SQL lause ei mõjuta andmebaasis ühtegi rida, siis funktsioon tekitab erandi. Selline lahendus aga eeldab, et funktsioon tuleb kirjutada PL/pgSQL keeles. Paljudel juhtudel piisaks ja oleks ka töökiiruse mõttes parem kasutada SQL funktsioone. Seega täiendati pgApex'i funktsionaalsust nii, et kui andmebaasi funktsioon ei tagasta väärtust või tagastab FALSE, siis kasutajale kuvatakse samuti veateade. Muudatus hõlmas nii tavalisi kui ka tabelivorme.

6.4.2 Võimalus anda ette tabelivormi funktsioonile xmin parameetrit

Kuna tabelivormiga on samuti võimalik andmeid muuta (näiteks muuta olemi seisundit), siis võimalus anda seal funktsioonile ette ainult olemi unikaalne identifikaator ja sisselogitud kasutaja kasutajanimi pole optimistliku lukustamise realiseerimiseks piisav. Optimistliku lukustamise jaoks peaks saama funktsioonile parameetri väärtusena ette anda ka rea versiooni identifikaatori (*xmin* väärtuse). Selleks täiendati tabelivormi funktsionaalsust. pgApexi kolmandas versioonis saab valida, millises vaate veerus on *xmin* väärtused ning tabelivormi kirjeldamisel saab määrata, kas selle taga olevale funktsioonile antakse ette *xmin* väärtus teise parameetri väärtusena või mitte. See eeldab, et valitud andmebaasi funktsiooni teise parameetri oodatav väärtus on *xmin* (oluline on just parameetrite järjekord, mitte nimi).

Joonis 14 esitab tabelivormi haldamislehe osa, kus saab määrata *xmin* väärtusele vastavat vaateveeru ja valida, et seda antakse ette funktsioonile teise parameetri väärtusena.



Joonis 14. pgApex'i arendaja vaates tabelivormi haldamine – xmin parameetri osa.

6.5 Muudatused ja parandused olemasolevas funktsionaalsuses

Arenduse käigus jäid autorile ja tooteomanikule (juhendajale) silma mõned probleemid olemasolevas pgApexi funktsionaalsuses, mida oleks hea parandada, tehes selleks funktsionaalsuses muudatusi või täiendusi. Enamus nendest on arendaja kasutajaliidese probleemid.

Nimekiri tehtud parandustest, muudatustest ja täiendustest on järgmine.

- Arendaja vaates olid vormiregioonil funktsiooni parameetrid suvalises järjekorras. Paranduse tulemusena on need sorteeritud järjekorranumbri järgi.

- Arendaja vaates kuvati kõikides vaadete valimise kohtades neid suvalises järjekorras. Paranduse tulemusena on need sorteeritud kõigepealt skeemi nime ja siis vaate nime järgi.

- Arendaja vaates kuvati kõikides vaate veergude valimise kohtades neid suvalises järjekorras. Paranduse tulemusena on need sorteeritud järjekorranumbri järgi tabelis.

- Arendaja vaates olid vormiregiooni vormiväljade seadistamise vormil mõned väljad kasutajaliideses liiga kitsad ja seega polnud nendesse sisestatud tekst nähtav. Paranduse tulemusena muudeti vormiväljade seadistamise vormide kujundust. Joonis 15 on esitatud vormivälja haldamisvorm teises- ja Joonis 16 kolmandas pgApex'i versioonis.

- Arendaja vaates peab vormiregioonil vormiväljade seadistamise vormil olema välja nimi unikaalne ühe lehe piires ja mitte sisaldama tühikuid ning numbreid. Paraku seda ei valideeritud ja sellepärast polnud arusaadav, et selline tingimus eksisteerib. Paranduse tulemusena lisati puuduv valideerimine ja vormivälja nime juurde lisati abitekst.

- Alates pgApexi esimesest versioonist on vormiregioonile võimalik lisada märkeruudu tüüpi element. Paraku selgus, et see et tööta nii nagu peab. Nimelt jõuab vormis olevate andmete salvestamisel märkeruudule vastav väärtus tagasüsteemini nagu tühi string, kuid peaks jõudma tõeväärtustüüpi väärtus. Paranduse tulemusena jõuab tagasüsteemile TRUE või FALSE väärtus vastavalt sellele, kas märkeruudul on kasutajaliideses märgistatud või mitte.

- Arendaja vaates ei saanud vormiregiooni vormivälja seadistuse vormis maha võtta eeltäitmisveergu, kui see oli valitud. Paranduse tulemusena loodi võimalus see maha võtta.

- Vormiregiooni vormiväljale saab määrata abiteksti. See tekst ei olnud väga nähtavas kohas. Nimelt kuvati seda hüpiktekstina ehk kasutaja pidi panema hiirekursori vastava vormivälja sildi juurde, et seda näha. Paranduse tulemusena kuvatakse abiteksti nüüd ka vormivälja all väiksema halli abitekstina, kui selle pikkus on lühem, kui 100 märki. Vastasel juhul kuvatakse endiselt ainult

hüpikaknas. Selle tingimuse põhjuseks on see, et liiga pikk abitekst ei näe ekraanil hea välja.

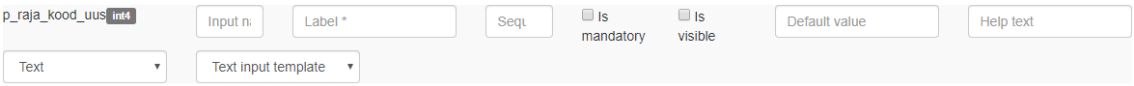
- Vormiregiooni vormivälja abitekstis lubati kasutada HTML märgiseid, et saaks näiteks välja tuua mingit abitekstis olevat tähtsamat informatsiooni „“ HTML märgiste paari abil.

- Vormiregiooni raadionupu, rippmenüü ja liitboksi tüüpi vormiväljade puhul puudus võimalus teha valik, kus väärtus puudus ja seetõttu vormi kinnitamisel pidi sellist tüüpi väljadel alati olema väärtus, mis mõnikord takistas rakenduste kasutamist. Näiteks oletame, et igal töötajal on null või üks amet (st amet võib ka puududa). Kui isikule oli korra amet määratud ei saanud enam registreerida fakti, et isikul amet puudub. Paranduse tulemusena loodi võimalus teha eelnimetatud vormiväljadel valik, millele vastab NULL. See kas NULL on rakenduse kasutajale pakutavas valikus sõltub sellest, kas vastav vormiväli on arendaja vaates märgitud kohustuslikuks või või mitte – kui on kohustuslik, siis NULL’i ei ole valikus kui ei ole kohustuslik, siis NULL on valikus.

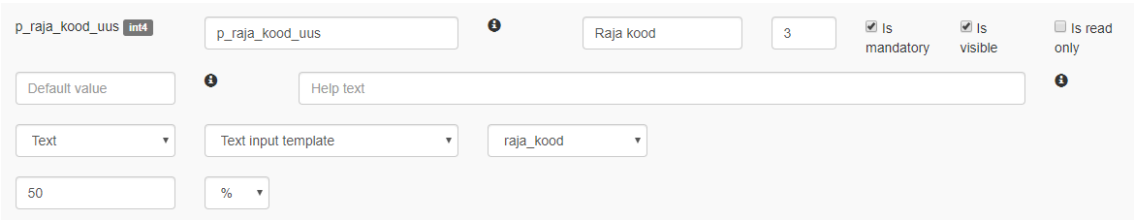
- Võib tekkida olukord, kus vormiregiooni poolt kasutatavale andmebaasi funktsioonile lisati või sealt kustutati mingeid parameetreid. Arendaja kasutajaliides ei arvestanud sellega. See tähendab, et arendajale kuvati vana parameetrite hulka, kuna funktsiooni parameetrite nimekiri ja nendele vastavate vormiväljade seadistused salvestatakse pgApexi enda andmebaasi ja hiljem loetakse sealt kasutajaliideses kuvamiseks informatsiooni nende kohta. Selleks, et näha uut parameetrite hulka, pidi arendaja valima vormiregioonis mingi teise funktsiooni ja siis panema tagasi õige funktsiooni. See oli arendajale halb kasutajakogemus, sest niimoodi tehes tühistati kõikide vormiväljade seadistused ning neid pidi uuesti kirjeldama. Paranduse tulemusena lisati loogika, mille järgi arendajale kuvatakse ka uued funktsiooni parameetrid ning kustutatuid enam ei näidata.

- Arendaja vaates konkreetse rakenduse ja lehekülge elementide haldamisel (näiteks regiooni loomisel või muutmisel) polnud arusaadav, millise rakendusega või leheküljega on tegemist. Tulemuseks lisati rakenduse ja lehekülje nimes konkreetse rakenduse või lehekülge elementide

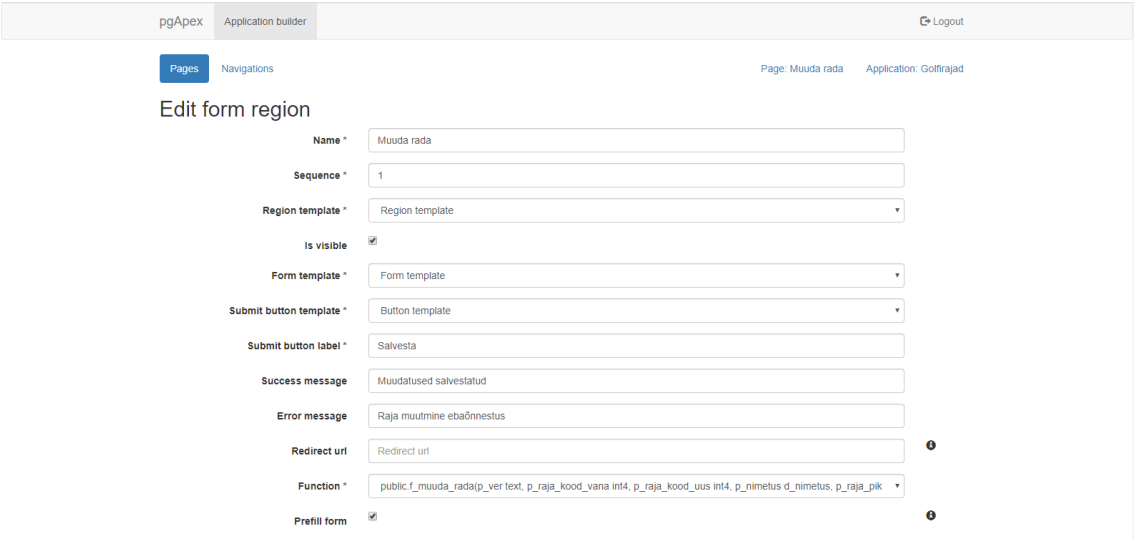
haldamisvaadetesse. Näidiseks on vormiregiooni halduslehe osa Joonis 17, kus on üleval paremas nurgas lehekülje ja rakenduse nimi.



Joonis 15. pgApex'i haldusliides. Vormiregiooni vormivälja haldusvorm teises versioonis.



Joonis 16. pgApex'i arendaja vaate. Vormiregiooni vormivälja haldusvorm kolmandas versioonis.



Joonis 17. pgApex'i arendaja vaate. Vormiregiooni halduslehe, kus on kuvatud lehekülje ja rakenduse nimi.

7 Muudatuste statistika

Järgnevalt esitatakse statistiline ülevaade programmi kolmanda versioonis tehtud muudatustest ja täiendustest pgApexi andmebaasis ja lähtekoodis, millest tekkis kolmas versioon.

7.1 Muudatused andmebaasis

Andmebaasi muudatused saadi kätte pgAdmin [51] tarkvara 4.20 versiooni lisandunud „Schema diff“ [52] tööriista abil. See võimaldab võrrelda kahe andmebaasi skeeme. Selle tööriista abil võrreldi skeemi *pgapex* pgApexi teise versiooni poolt kasutatavas andmebaasis *pgapex2* ning kolmanda versiooni poolt kasutatavas andmebaasis *pgapex3*. Paraku ei saanud selle vahendi abil täpset ülevaadet funktsioonide muudatustest, kuna kõikide funktsioonide puhul oli muudetud õiguseid ja seetõttu näitas töörist, et muudeti kõiki funktsioone, mis pole tegelikult tõsi. Seega andmebaasifunktsioonide muudatused saadi kätte GitHub keskkonda sisseehitatud kahe salve võrdlemise funktsionaalsuse abil. Tabelite veergude arvu kättesaamiseks käivitati *pgapex2* ja *pgapex3* andmebaasides Joonistel Joonis 18 ja Joonis 19 olevad SQL päringud.

```
SELECT COUNT(*) AS number_of_base_table_columns FROM
information_schema.columns INNER JOIN information_schema.tables USING
(table_schema, table_name) WHERE table_schema = 'pgapex' AND
table_type='BASE TABLE';
```

Joonis 18. SQL lause baastabelite veergude arvu kättesaamiseks.

```
SELECT Count(*) AS number_of_materialized_view_columns
FROM pg_class INNER JOIN pg_namespace ON
pg_class.relnamespace=pg_namespace.oid
INNER JOIN pg_attribute ON pg_class.oid=pg_attribute.attrelid
WHERE relkind = 'm' AND attnum>=1 AND
nspname='pgapex';
```

Joonis 19. SQL lause hetktõmmiste veergude arvu kättesaamiseks.

pgApex ei kasuta sisemiselt vaateid ning väliseid tabeleid ja päringud arvestavad sellega.

pgApexi teises versioonis baastabelites on kokku 295 veergu ja hetktõmmistes 25 veergu – kokku 320 veergu.

pgApexi kolmanda versioonis baastabelites on kokku 390 veergu ja hetktõmmistes 26 veergu – kokku 416 veergu.

Seega lisandus tabelitesse kokku 96 veergu.

Käesoleva töö käigus ei kustutatud ühtegi tabelit, funktsiooni, hetktõmmist, andmetüüpi ega veergu.

Võrdluse tulemus:

- Uued baastabelid: 11
- Muudetud baastabelid: 10
- Muudetud hetktõmmised: 2
- Uued andmebaasifunktsioonid: 33
- Muudetud andmebaasifunktsioonid: 30
- Uued andmetüübid: 1
- Uued veerud: 96

7.2 Muudatused lähtekoodis

Lähtekoodi muudatuste kätte saamiseks kasutati GitHubi keskkonda sisseehitatud salvede võrdlemise funktsionaalsust. Võrreldi teise ja kolmanda versiooni salvesid. Tulemus on järgmine:

- Muudetud failide arv: 81
- Lisatud koodiridade arv: 13796
- Kustutatud koodiridade arv: 481

8 Tulemuste valideerimine

Tulemuste valideerimiseks loodi pgApexi abil rakendus autori osalusel tehtud „Andmebaasid II“ aineprojekti [27] põhjal. Loodud rakenduses on kasutusel kogu pgApexi kolmandas versioonis realiseeritud funktsionaalsus. Et funktsionaalsust saaks täielikult kasutada, siis „Andmebaasid II“ aineprojekti täiendati vajalike vaadetega ja funktsioonidega. Järgnevalt tutvustatakse ainult nende funktsionaalsuste kasutuse tulemust, mis lisandusid pgApexi kolmandasse versiooni.

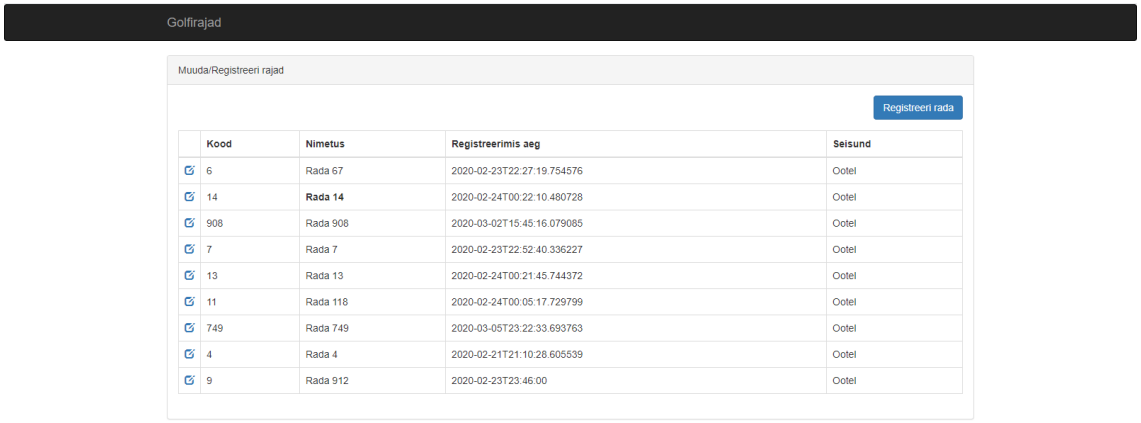
Joonis 20 esitab pgApexi kolmandas versioonis tekkinud „Raport ja vorm“ regiooni, kus on radade nimekiri ning mille juures on nupp, mis avab vormi raja lisamiseks (Joonis 21). Samuti on iga raja juures nupp, mis avab selle raja mutumiseks mõeldud vormi (Joonis 22). Raja muutmise ja lisamise vormil on näha uusi väljatüüpe: kalendrivalik raja avamise aja juures (Joonis 23 on seda näha avatud kujul) ning liitboks raskusastme valikuks. Samuti on näha suuremad tekstiväljad, mille puhul on sisse lülitatud WYSIWYG tekstiredaktor raja kirjelduse juures. Raja lisamise vormil on registreerija e-posti aadressi väli mõeldud ainult lugemiseks (täidetakse automaatselt) ehk väljas olevat väärtust ei saa muuta. Raja pikkuse juures on näha ka uuendatud välja abiteksti kuvamist. Raja nimetuse puhul on näha seda, et saab kasutada HTML märgendeid. Antud juhul on kasutatud „” märgendite paari. Raja muutmise vormil (Joonis 22) on neli alamvormi.

- *Kategooriad*. See on alamtabelivorm rajaga seotud kategooriate vaatamiseks ning kustutamiseks.

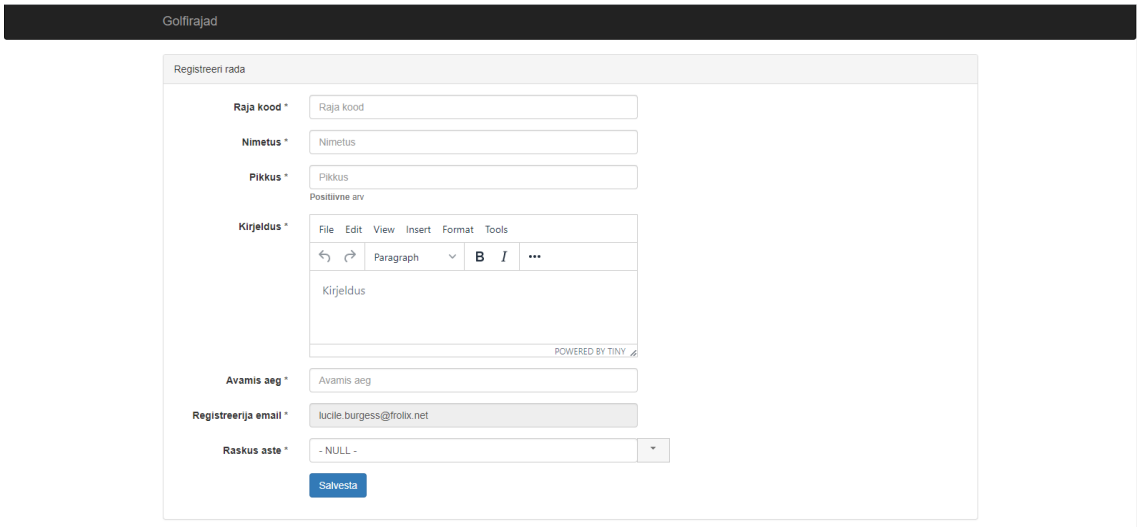
- *Lisa rada kategooriasse*. See on tavaline alamvorm raja kategooriasse lisamiseks. Pärast selle vormi kinnitamist on lisatud kategooriat näha kategooriate nimekirjas *Kategooriad* alamtabelivormis.

- *Reserveeringud*. See on alamtabelivorm raja reserveeringude nimekirja vaatamiseks ning kustutamiseks. Erinevus *Kategooriad* alamtabelivormist seisneb selles, et see alamtabelivorm on seadistatud nii, et iga reserveeringu juures on nupp, millele vajutamine viib selle reserveeringu muutmise vormile. Põhimõte on sama, nagu *Report and form* tüüpi regioonil.

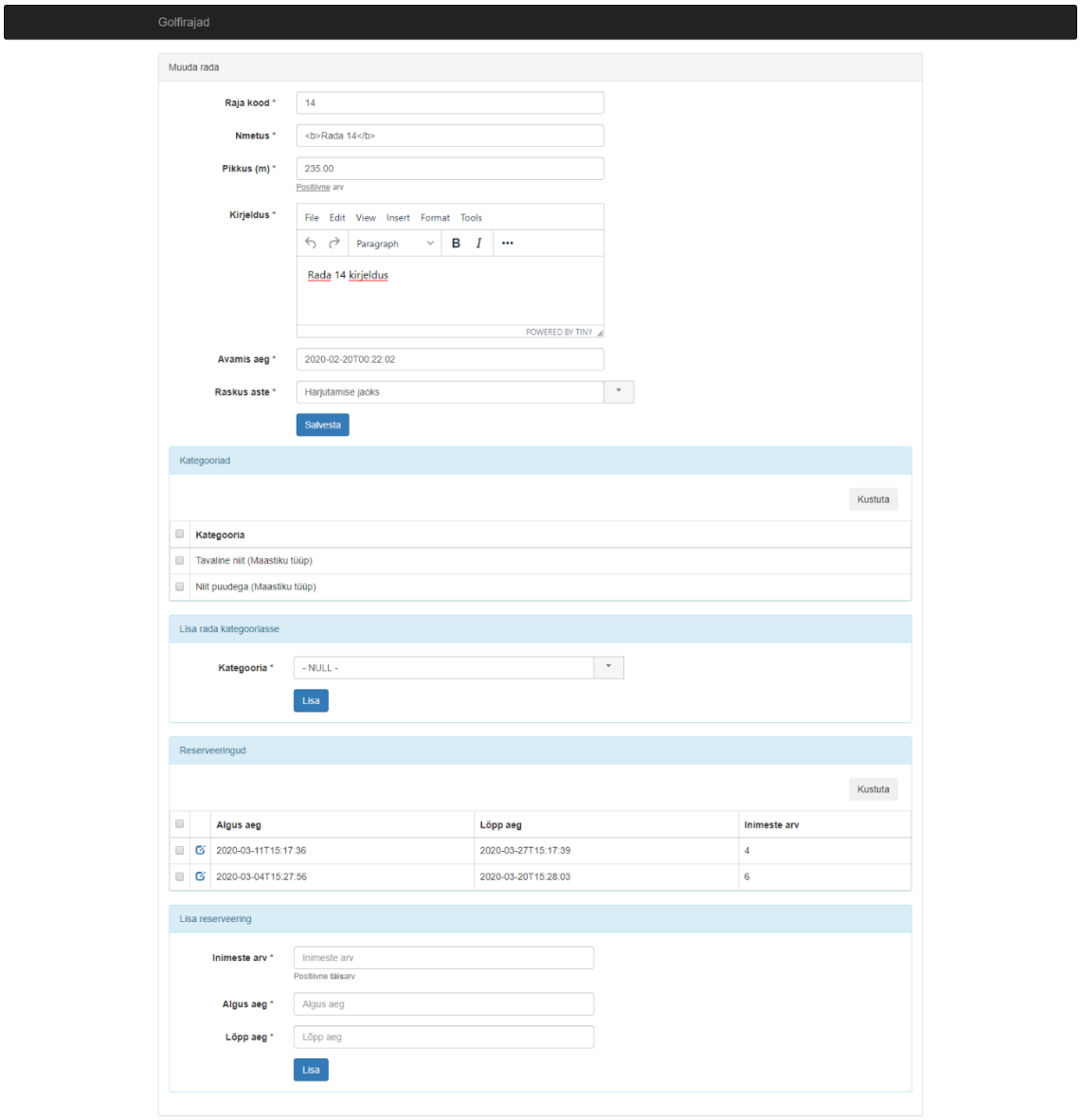
- *Lisa reserveering*. See on tavaline alamvorm, mille abil saab rajale lisada reserveeringut. Sarnaselt *Lisa rada kategooriasse* vormile, pärast andmete salvestamist kui reserveeringu lisamine õnnestub, siis lisatud reserveeringut on näha reserveeringute nimekirjas *Reserveeringud* alamtabelivormis.



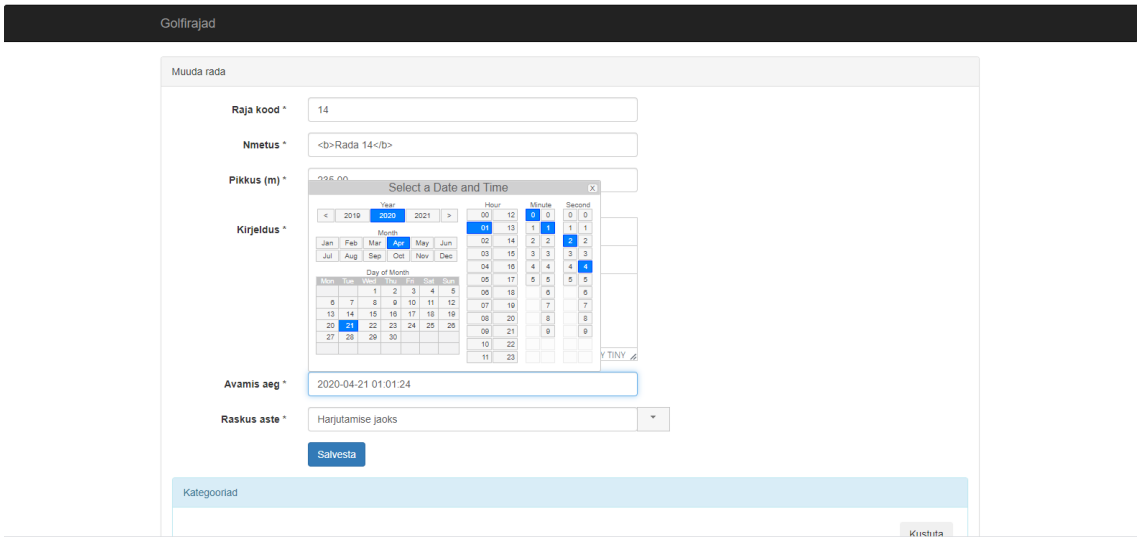
Joonis 20. pgApex3. Raport koos nuppudega olemite lisamiseks ja muutmiseks.



Joonis 21. pgApex3. Olemi lisamise vorm.



Joonis 22. pgApex3. Olemi muutmise vorm.



Joonis 23. pgApex3. Avatud kalendrivalik olemi muutmise vormil.

9 Arendusvaade

Järgnevalt kirjeldatakse ideed selle kohta, mida võiks realiseerida pgApexis uute funktsionaalsustena ja mida oleks vaja muuta või täiendada olemasolevas funktsionaalsuses lisaks Rait Raidma [18] ja Nikolai Kopa [19] lõputöödes pakutud ideedele.

9.1 Ideed uueks funktsionaalsuseks

- Võimalus kasutada rakendusele väljanägemise muutmiseks erinevaid teemasid.
- Võimalus luua regioone, mis kuvatakse hüpikaknas (*pop-up)*. Näiteks on samal lehel loodud raport ja (peidetud) detailvaate regioon. Kui olemite nimekirjas vajutatakse konkreetse olemi juures nupule, siis detailvaadet võiks kuvada mitte eraldi lehel, vaid hüpikaknas.
- Võimalus määrata kõikides kohtades, kus kutsutakse välja andmebaasifunktsioone (vormiregioonid), milliste funktsiooni tagastatavate väärtuste puhul kuvatakse kasutajale õnnestumis- ja veateadet. Praegu on nii, et veateadet kuvatakse vaid juhul, kui funktsiooni täitmisel tekib erand (*exception*), funktsioon tagastab tõeväärtuse FALSE või ei tagasta väärtust. Muul juhul kuvatakse õnnestumise teade.
- Võimalus seadistada vormiregiooni väljade paigutust. Praegu kuvatakse kõik väljad üksteise järel ülevalt alla. Arendajatele võiks anda võimaluse paigutada vormivälju näiteks ühele reale üksteise kõrvale. Sama põhimõtet saaks rakendada ka detailvaate puhul.
- Võimalus kopeerida elemente (näiteks lehekülge ja regioone). Võib tekkida olukord, kus on vaja luua kaks väga sarnase ülesehitusega regiooni (näiteks olemi lisamise vorm ja olemi muutmise vorm) ning kopeerimise võimalus kiirendaks rakenduse loomist oluliselt.
- Võimalus kopeerida rakendust, et võtta uue rakenduse loomisel aluseks olemasolev rakendus.

- Võimalus arendaja vaates salvestada poolikut või vigaste andmetega regiooni (hetkel pole see võimalik). Selline vajadus võib tekkida näiteks olukorras, kui arendaja avastas regiooni loomisel, et mõni õnnestumiseks vajalik eeltingimus on täitmata (näiteks pole funktsioonis vajalikku parameetrit või vaates vajaliku veergu). Praegu, kui arendaja hakkab regiooni looma või muutma, siis ta peab kindlasti selle salvestama, sest muidu lähevad kõik muudatused kaotsi. Näiteks vormiregioonis võib olla palju alamvorme ja vormi taga oleval funktsioonil võib olla palju parameetreid ning seetõttu on selle uuesti nullist loomine tülikas.

- Võimalus luua võimalikult automaatselt (arendaja peab võimalikult vähe midagi sisestama) täielikku CRUD funktsionaalsusega raportit, kus oleks olemite nimekiri ja iga olemi juures oleksid nupud selle olemi detailandmete vaatamiseks, uuendamiseks ja kustutamiseks ning oleks võimalik seadistada, milliseid operatsioone saab raportis olemitega läbi viia.

9.2 Ideed olemasoleva funktsionaalsuse muutmiseks

- Arendaja vaates saab mõnedele kasutajaliidese elementidele (veerud, vormiväljad, alamregioonid) määrata järjekorranumbri. Praegu sisestatakse seda käsitsi vastavasse vormi välja. Järjekorranumbrite määramise võiks teha visuaalseks, et elemendite järjekorra muutmine oleks kiirem ja mugavam. See tähendab, et kasutajaliidese elementide järjekord sõltuks otseselt nende paigutuse järjekorrast arendaja vaates. Selleks võiks teha võimaluse muuta järjekorda kas elemente pukseerides (*drag and drop*) või iga elemendi juures oleva nupu abil. Teise variandi puhul võiksid iga elemendi juures olla näiteks noolekujulised nupud, mille abil saaks elementi üles- või allapoole liigutada.

- Muuta alamvormide värskendamine dünaamiliseks. Näiteks on olukord, kui lehel on alamtabelivorm olemite nimekirjaga ja alamvorm, mille abil saab sellesse nimekirja lisada olemeid. Praegu, kui alamvormi andmed salvestatakse, siis värskendatakse tervet lehte. See on tavaline HTML vormi käitumine. Võiks teha nii, et alamvormis uuendatakse andmeid jooksvalt ilma terve lehe värskendamiseta ehk kasutada SPA põhimõtteid.

- Praegu, kui arendaja vaates regiooni salvestamisel tekib mingi andmebaasi viga (näiteks mingi andmebaasi kitsenduse tingimus pole täidetud), siis ei väljasta tarkvara mingit veateadet, vaid arendaja suunatakse tagasi regioonide nimekirja lehele nagu oleks andmed edukalt salvestatud, aga andmed pole tegelikult salvestatud. Sellisel juhul võiks programm teada anda, millisel põhjusel andmete salvestamine ebaõnnestus, kuvades kasutajaliideses konkreetseid veateateid. Kui programmis pole loogikavigu, siis tavalisel kasutajal ei tohiks sellist olukorda juhtuda, aga see täiendus lihtsustaks oluliselt silumist nende jaoks, kes hakkavad tulevikus pgApex'it edasi arendama.

- Praegu saab märkeruudu tüüpi vormiväljale määrata vaikimisi väärtust vabatekstina, kuid tavaliselt kasutakse märkeruutu tõeväärtuste (TRUE või FALSE) salvestamiseks ning esitamiseks. Märkeruudu tüüpi välja puhul võiks muuta vaikimisi väärtuse määramise välja vabatekstilisest väljast rippmenüüks, kus saab valida, kas vaikimisi väärtus on TRUE, FALSE, või see puudub.

- Praegu saab rakenduse loomiseks kasutada ainult pgApexiga samas serveris olevaid andmebaase (andmebaas tuleb ära määrata rakenduste omadustes). Võiks teha nii, et saaks kasutada ka teistes serverides olevaid andmebaase. Siiski tuleb märkida, et väliste tabelite abil on juba praegu võimalik luua rakendust ka teistes serverites olevate andmebaaside ja isegi mitte-PostgreSQL andmebaaside põhjal.

- Realiseerida ridade optimistlik lukustamine ka pgApex arenduskeskkonnas.

- Anda võimalust kasutada funktsioonide ülelaadimist. Ülelaadimine tähendab, et skeemis on mitu funktsiooni ühe nimega, aga erinevate parameetritega. Praegu, kui rakenduse poolt kasutatava andmebaasi skeemis on mitu samanimelist funktsiooni ja ühte nendest kasutatakse vormiregiooni funktsioonina, siis funktsiooni parameetrite nimekiri läheb arendaja vaates katki.

10 Kokkuvõte

Töö eesmärgiks oli anda pgApex keskkonnale võimekus luua andmebaasirakendusi, kus on täielikult realiseeritud CRUD funktsionaalsus. CRUD akronüüm viitab operatsioonidele andmetega (loo, loe, uuenda, kustuta), mida selline rakendus peab võimaldama läbi viia.

Töö raames uuriti pgApexi olemasolevat, esimeses ja teises versioonis realiseeritud, funktsionaalsust. Samuti viidi läbi taustauuring, mille tulemusena leiti kuus pgApexiga sarnast eesmärki täitvat keskkonda ning viite neist uuriti lähemalt. Uuringu käigus proovis autor nende keskkondade abil luua andmebaasirakendust, kus oleks samasugune funktsionaalsus nagu on realiseeritud käesoleva töö tulemuste valideerimiseks pgApex keskkonna abil loodud rakenduses. Uuringu tulemusena selgus, et paljude keskkondade funktsionaalsus erineb pgApex'ist. Näiteks, pole neis võimalik kasutada andmebaasis loodud funktsioone ning keskkonnad nõuavad failide veebiserverisse paigaldamist (publitseerimist). Samuti ei võimalda mitmed uuritud keskkonnad luua andmebaasirakendusi, kus toimuks ridade optimistlik lukustamine.

Peale seda planeeriti väljalase. Tooteomanik (lõputöö juhendaja) nimetas täieliku CRUD funktsionaalsuse realiseerimiseks vajalikke täiendusi pgApex keskkonnas ning määras nende prioriteedid. Autor hindas nende täienduste keerukust ning jagas need iteratsioonidesse.

Järgmiseks analüüsiti süsteemi. Mudelite abil kirjeldati uusi ja muudetud pgApex'i süsteemi osasid.

Töö tulemusena täiendati pgApexit funktsionaalsustega, mis on vajalikud täieliku CRUD funktsionaalsuse realiseerimiseks selle abil loodavates rakendustes. Realiseeriti uus „Raport ja vorm" tüüpi regioon, mis sisaldab raportit (aruannet), kus iga olemi andmete juures on nupp, mis avaneb vormi selle olemi andmete muutmiseks. Samuti on võimalik regiooni lisada nupp, mis avab vormi uue olemi andmete lisamiseks. Vormiregioonile oli tekitatud võimalus lisada kahte tüübi alamvorme: tavaline alamvorm ja alamtabelivorm. Tavaline alamvorm on sarnane tavalise vormiregiooniga, kuid asub alamregioonis. Alamtabelivormi abil saab kuvada olemite nimekirja ning rakendada nende suhtes arendaja vaates määratud funktsioone. Näiteks saab selle abil

olemeid (täpsemalt nende andmeid) kustutada. Mõlemat tüüpi alamvormid on seostatavad peavormi eeltäitmisega ja seega on nende abil võimalik realiseerida peavormis oleva olemiga 1:M või 1:0..1 seosetüübi kaudu seotud olemite andmetega CRUD operatsioonide tegemist. Näiteks peavormis näidatakse raja andmeid ning alamvormis saab lisada ja kustutada selle raja kategooria omamisi.

Loodi võimalus realiseerida rakendustes ridade optimistlikku lukustamist. Selleks muudeti vormi kinnitamise loogikat vormiga regioonides ning tabelivormi puhul loodi võimalus anda selle vormiga seotud funktsioonile argumendina ette *xmin* väärtus, mis näitab tabelis oleva rea versiooni. Seega kasutatakse optimistliku lukustamise jaoks PostgreSQLi sisseehitatud funktsionaalsust, mille kohaselt on igas baastabelis peidetud veerg, milles on süsteemi poolt automaatselt uuendatav rea versiooni identifikaator.

Realiseeriti kaks uut vormivälja tüüpi: kalendrivalik ja liitboks. Samuti lisati vormiväljadele seadistatavaid omadusi: välja laius, kas väli on mõeldud ainult lugemiseks, kalendri formaat kalendrivaliku puhul ning kõrgus ja WYSIWYG tekstiredaktori kasutamine suurema tekstivälja puhul. Lisaks leiti ja realiseeriti töö käigus võimalusi olemasoleva funktsionaalsuse ja kasutajaliidese parandamiseks ning täiendamiseks, mis oluliselt parandasid kasutajakogemust.

Tulemuste valideerimiseks loodi pgApexi abil rakendus lõputöö autori osalusel tehtud „Andmebaasid II“ aineprojekti „Golfirajad“ põhjal. Selles rakenduses on realiseeritud uut funktsionaalsust, mille tegi võimalikuks pgApexi uus versioon. Rakendus on kättesaadav aadressil: http://apex.ttu.ee/pgapex3/public/index.php/app/10/31

Rakendusele sisselogimiseks vajalik kasutajanimi on *lucile.burgess@frolix.net* ning parool on *laborum*

pgApexi kolmanda versiooni lähtekood on avalik, MIT litsentsiga kaitstud ning asub järgmises GitHub’i salves: https://github.com/dmibiz/pgapex3

Võimalus luua täieliku CRUD funktsionaalsusega rakendusi on pgApexi arengus suur samm, kuid täiendus- ja parandamisvõimalusi on veel palju. Kuna pgApex on loodud Oracle APEX keskkonna eeskujul PostgreSQLi jaoks, siis ideaaliks oleks luua sellega samal tasemel keskkond.

Kasutatud kirjandus

[1] Document Object Model - Wikipedia [WWW] https://en.wikipedia.org/wiki/Document_Object_Model (13.10.2020)

[2] Integrated development environment - Wikipedia [WWW] https://en.wikipedia.org/wiki/Integrated_development_environment (19.04.2020)

[3] SQL - Wikipedia [WWW] https://en.wikipedia.org/wiki/SQL (13.10.2020)

[4] Definition of WYSIWYG [WWW] https://www.thefreedictionary.com/WSIWYG (20.04.2020)

[5] DB-Engines ranking [WWW] https://db-engines.com/en/ranking (13.10.2019)

[6] Andmebaasid I. *Teema 2. Hierarhilise-, võrk- ja relatsioonilise andmemudeli põhimõisteid.* TalTech Tarkvarateaduse instituut.

[7] Modern SQL [WWW] https://modern-sql.com/ (13.10.2019)

[8] No-code development platform - Wikipedia [WWW] https://en.wikipedia.org/wiki/No-code_development_platform (01.10.2019)

[9] Open-Closed Principle [WWW] https://deviq.com/open-closed-principle/ (13.10.2019)

[10] Spendolini, B., Geller, A. *Oracle Application Express: Build Powerful Data-Centric Web Apps with APEX*. Oracle Press. 2017.

[11] Savenko, D. Read-Write APEX application fully based on alien data or is it mandatory to use exactly Oracle Database? [WWW] https://dsavenko.me/read-write-apex-application-fully-based-on-alien-data/ (06.10.2019)

[12] Openxava. Getting Started [WWW] https://www.openxava.org/OpenXavaDoc/docs/getting-started_en.html (18.04.2020)

[13] FADEX. Design beautiful web applications with your PostgreSQL database [WWW] https://sqldev.tech/getfadex (20.10.2019)

[14] The Advance Rapid Application Development GUI That Delivers Complete Web Applications [WWW] https://phprad.com/ (24.03.2020)

[15] PostgreSQL PHP Generator [WWW]

https://www.sqlmaestro.com/products/postgresql/phpgenerator/ (24.03.2020)

[16] Radzen. Create business web applications fast & easy [WWW] https://www.radzen.com/ (24.03.2020)

[17] 4WS Platform [WWW] http://4wsplatform.org/ (24.03.2020)

[18] R. Raidma. PostgreSQL andmebaasisüsteemi põhine metaandmetega juhitavate veebirakenduste kiirprogrammeerimise keskkond. Magistritöö. Tallinna Tehnikaülikool [WWW] https://digi.lib.ttu.ee/i/?6787 (06.10.2019)

[19] Kopa, N. *PostgreSQL andmebaassüsteemi põhise metaandmetega juhitavate veebirakenduste kiirprogrammeerimise keskkonna edasiarendus.* Bakalaureusetöö. Tallinna Tehnikaülikooli Tarkvarateaduse instituut, 2019. [WWW] https://digikogu.taltech.ee/et/Item/5675f9b7-ff11-4678-8551-7a5040ff4c2c (16.05.2020)

[20] How I stole roughly 100 BTC from an exchange and how I could have stolen more! [WWW] https://www.reddit.com/r/Bitcoin/comments/1wtbiu/how_i_stole_roughly_100_btc_from_an_exchange_and (27.03.2020)

[21] Warszawski, T, Bailis, P. *ACIDRain: Concurrency-Related Attacks on Database-Backed Web Applications*. Stanford InfoLab.

[22] Bailis, P., Fekete, A., Franklin, M., Ghodsi, A., Hellerstein, J., Stoica, I. *Feral Concurrency Control: An Empirical Investigation of Modern Application Integrity*. UC Berkley and University of Sidney.

[23] Design science (methodology) - Wikipedia [WWW] https://en.wikipedia.org/wiki/Design_science_(methodology) (11.10.2019)

[24] Norman, T. Agile Release Planning 101 [WWW] http://tommynorman.blogspot.com/2012/09/agile-release-planning-101.html (21. 10.2019)

[25] Agile Methods: The Good, the Hype and the Ugly [WWW] https://www.slideshare.net/tgrandison/agile-methods-the-good-the-hype-and-the-ugly-acm-webinar18-february-2015-bertrand-meye (21.10.2019)

[26] What is a timebox? [WWW] https://www.agilealliance.org/glossary/timebox/ (11.10.2019)

[27] Fridlund, A., Bizjulin, B., Zilkin, V. *Golfiklubi radade arvestus.* Projekt ainetes "Andmebaasid I" ja "Andmebaasid II". Tallinna Tehnikaülikool.

[28] TIL-9: Optimistic vs. Pessimistic Locking [WWW]

https://medium.com/@recepinancc/til-9-optimistic-vs-pessimistic-locking-79a349b76dc8 (07.05.2020)

[29] Optimistic and pessimistic locking with SQL [WWW] https://convincedcoder.com/2018/09/01/Optimistic-pessimistic-locking-sql/ (07.05.2020)

[30] XAMPP Apache + MariaDB + PHP + Perl [WWW] https://www.apachefriends.org/index.html (16.04.2020)

[31] PostgreSQL PHP Generator feature matrix [WWW] https://www.sqlmaestro.com/products/postgresql/phpgenerator/feature_matrix/ (10.04.2020)

[32] Openxava. Open Source Low-Code Platform for Rapid Development of Enterprise Web Applications [WWW] https://www.openxava.org/ (18.04.2020)

[33] Openxava. Configuration for PostgreSQL [WWW] https://www.openxava.org/OpenXavaDoc/docs/postgres_en.html (18.04.2020)

[34] Agile Release Planning: Let's Break It Down! [WWW] https://www.mpug.com/articles/agile-release-planning-lets-break-it-down/

[35] Release Planning in an Agile Project [WWW] https://www.dummies.com/careers/project-management/release-planning-agile-project/ (21.10.2019)

[36] Adaptive Planning - Release and Iteration Plan [WWW] https://www.atlassian.com/agile/agile-at-scale/long-term-agile-planning (21.10.2019)

[37] An agile guide to the planning processes [WWW] https://www.pmi.org/learning/library/agile-guide-planning-agile-approach-6837 (21.10.2019)

[38] What is Story Point in Agile? How to Estimate a User Story? [WWW] https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile (22.10.2019)

[39] Any+Time DatePicker/TimePicker AJAX Calendar Widget [WWW] https://www.ama3.com/anytime/ (10.12.2019)

[40] Jquery UI. Datepicker [WWW] https://jqueryui.com/datepicker/ (10.12.2019)

[41] Date range picker [WWW] https://www.daterangepicker.com/ (10.12.2019)

[42] pickadate.js [WWW] https://amsul.ca/pickadate.js/ (10.12.2019)

[43] JQuery UI. Autocomplete [WWW] https://jqueryui.com/autocomplete/#combobox (10.12.2019)

[44] TinyMCE [WWW] https://www.tiny.cloud/ (10.04.2020)

[45] Quill [WWW] https://quilljs.com/ (10.04.2020)

[46] Editor.js [WWW] https://editorjs.io/ (10.04.2020)

[47] CKEditor [WWW] https://ckeditor.com/ (10.04.2020)

[48] TinyMCE docs. License [WWW] https://www.tiny.cloud/docs-3x/extras/TinyMCE3x@License/ (10.04.2020)

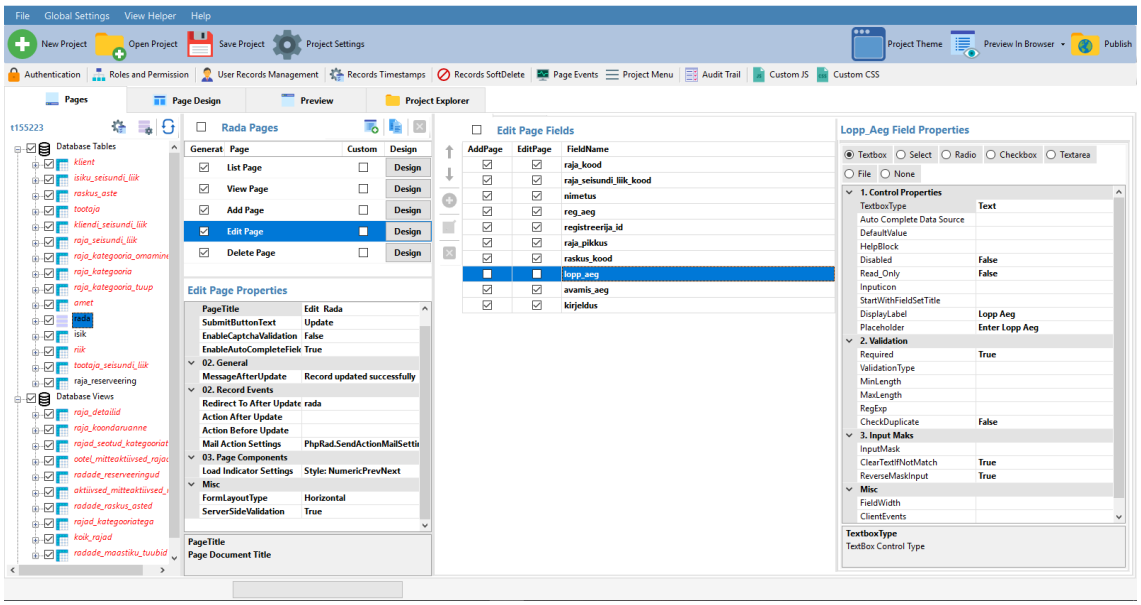
[49] TinyMCE docs [WWW] https://www.tiny.cloud/docs/ (10.04.2020)

[50] Concurrency Tokens. Npgsql Documentation [WWW] https://www.npgsql.org/efcore/modeling/concurrency.html (17.05.2020)

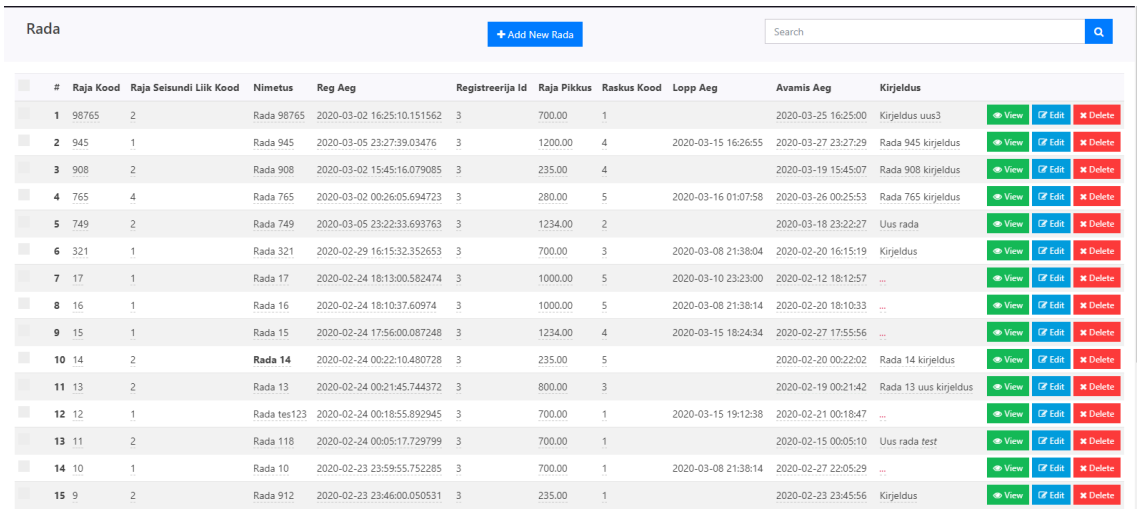
[51] pgAdmin [WWW] https://www.pgadmin.org/ (21.04.2020)

[52] pgAdmin. Schema Diff [WWW] https://www.pgadmin.org/docs/pgadmin4/3.x/schema_diff.html (21.04.2020)

Lisa 1 – PHPRad keskkonna ekraanipildid

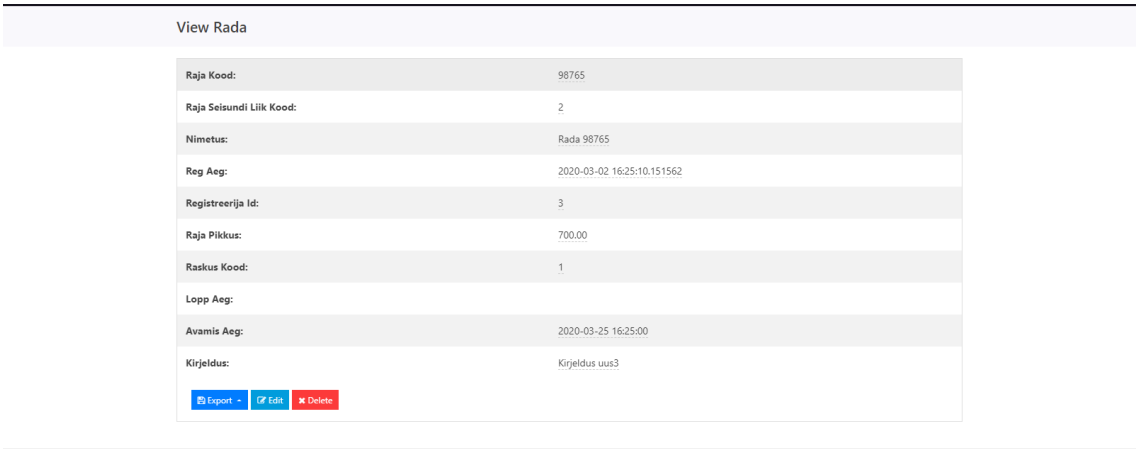


Joonis 24. PHPRad rakenduse haldamise töölaua rakendus.

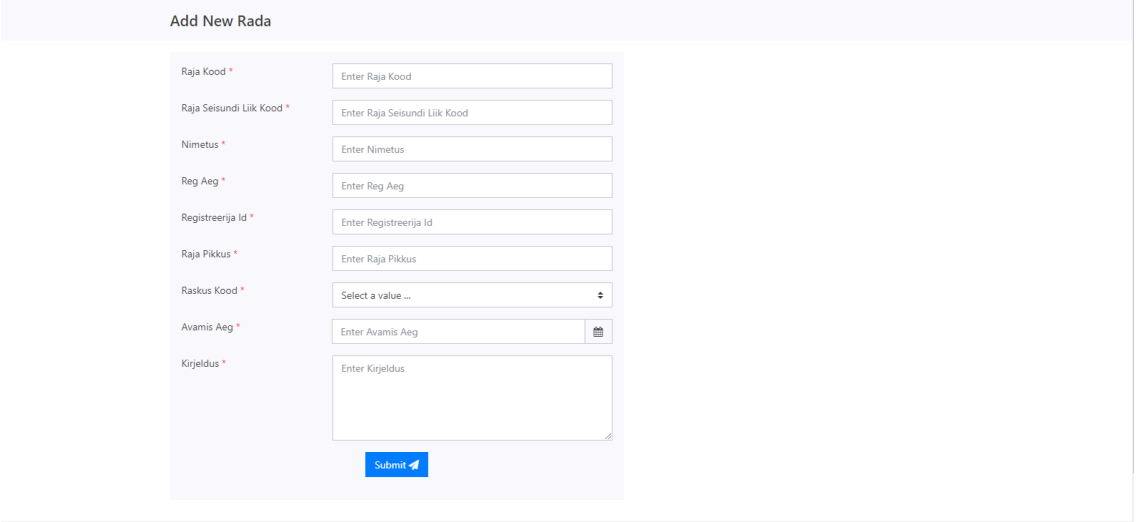


#	Raja Kood	Raja Seisundi Liik Kood	Nimetus	Reg Aeg	Registreerija Id	Raja Pikkus	Raskus Kood	Lopp Aeg	Avamis Aeg	Kirjeldus	
1	98765	2	Rada 98765	2020-03-02 16:25:10.151562	3	700.00	1		2020-03-25 16:25:00	Kirjeldus uus3	View Edit Delete
2	945	1	Rada 945	2020-03-05 23:27:39.03476	3	1200.00	4	2020-03-15 16:26:55	2020-03-27 23:27:29	Rada 945 kirjeldus	View Edit Delete
3	908	2	Rada 908	2020-03-02 15:45:16.079085	3	235.00	4		2020-03-19 15:45:07	Rada 908 kirjeldus	View Edit Delete
4	765	4	Rada 765	2020-03-02 00:26:05.694723	3	280.00	5	2020-03-16 01:07:58	2020-03-26 00:25:53	Rada 765 kirjeldus	View Edit Delete
5	749	2	Rada 749	2020-03-05 23:22:33.693763	3	1234.00	2		2020-03-18 23:22:27	Uus rada	View Edit Delete
6	321	1	Rada 321	2020-02-29 16:15:32.352653	3	700.00	3	2020-03-08 21:38:04	2020-02-20 16:15:19	Kirjeldus	View Edit Delete
7	17	1	Rada 17	2020-02-24 18:13:00.582474	3	1000.00	5	2020-03-10 23:23:00	2020-02-12 18:12:57	...	View Edit Delete
8	16	1	Rada 16	2020-02-24 18:10:37.60974	3	1000.00	5	2020-03-08 21:38:14	2020-02-20 18:10:33	...	View Edit Delete
9	15	1	Rada 15	2020-02-24 17:56:00.087248	3	1234.00	4	2020-03-15 18:24:34	2020-02-27 17:55:56	...	View Edit Delete
10	14	2	**Rada 14**	2020-02-24 00:22:10.480728	3	235.00	5		2020-02-20 00:22:02	Rada 14 kirjeldus	View Edit Delete
11	13	2	Rada 13	2020-02-24 00:21:45.744372	3	800.00	3		2020-02-19 00:21:42	Rada 13 uus kirjeldus	View Edit Delete
12	12	1	Rada tes123	2020-02-24 00:18:55.892945	3	700.00	1	2020-03-15 19:12:38	2020-02-21 00:18:47	...	View Edit Delete
13	11	2	Rada 118	2020-02-24 00:05:17.729799	3	700.00	1		2020-02-15 00:05:10	Uus rada *test*	View Edit Delete
14	10	1	Rada 10	2020-02-23 23:59:55.752285	3	700.00	1	2020-03-08 21:38:14	2020-02-27 22:05:29	...	View Edit Delete
15	9	2	Rada 912	2020-02-23 23:46:00.050531	3	235.00	1		2020-02-23 23:45:56	Kirjeldus	View Edit Delete

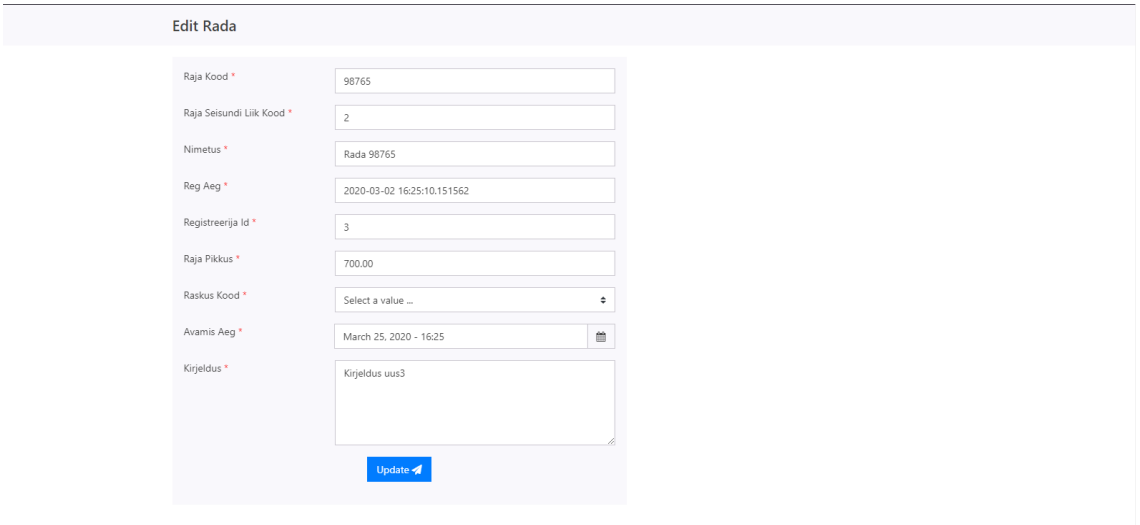
Joonis 25. PHPRad rakenduse loomisel genereeritud olemite nimekirja leht.



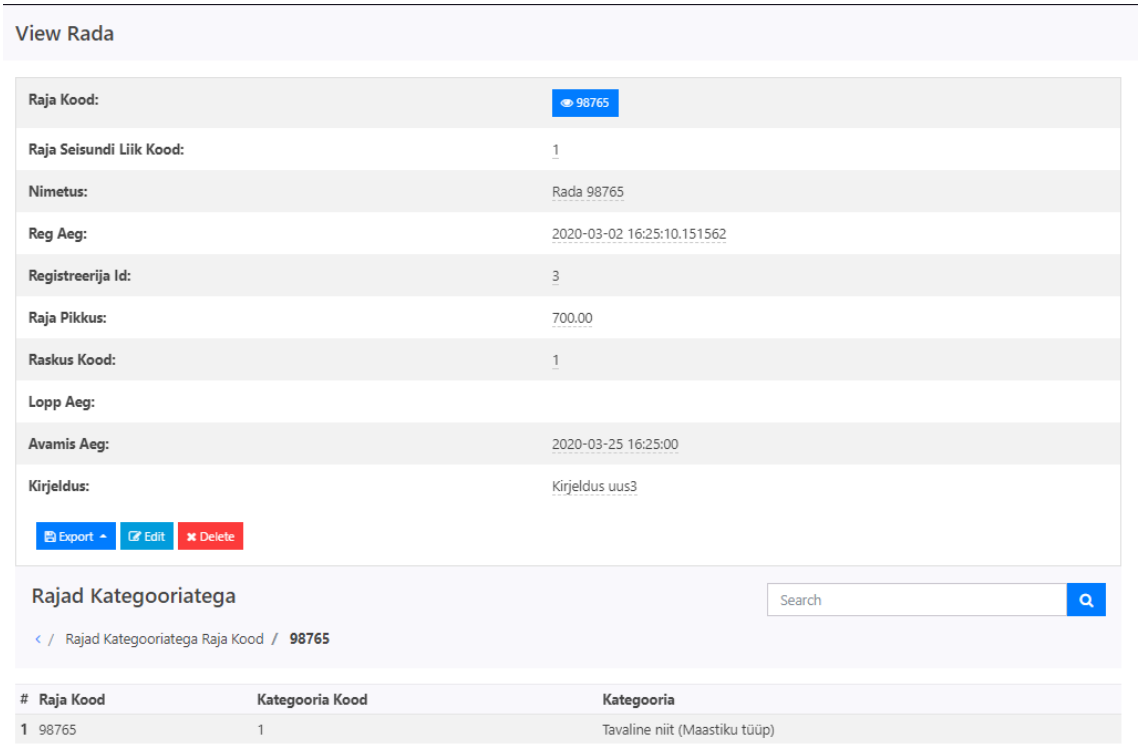
Joonis 26. PHPRad rakenduse loomisel genereeritud olemi detailvaate leht.



Joonis 27. PHPRad rakenduse loomisel genereeritud olemi lisamise leht.

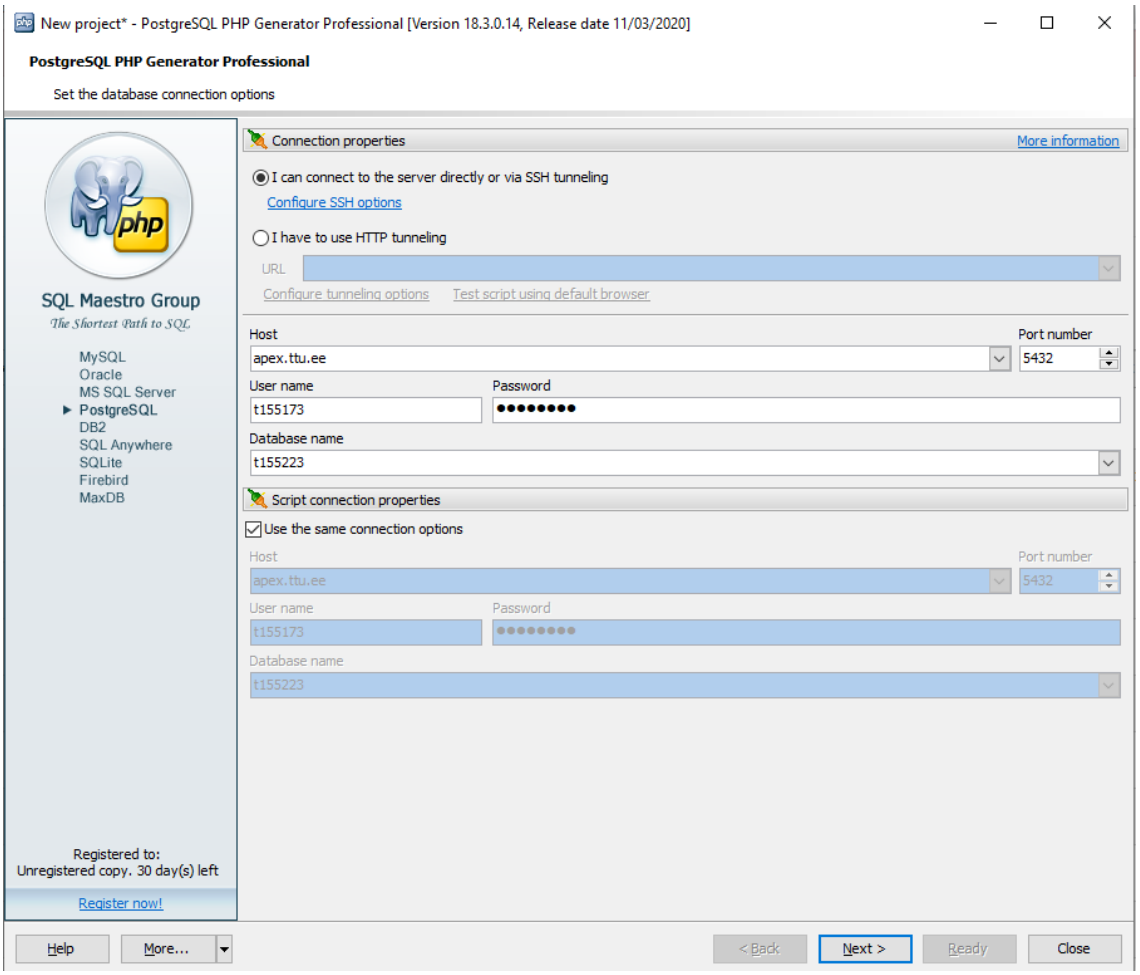


Joonis 28. PHPRad rakenduse loomisel genereeritud olemi muutmise leht.

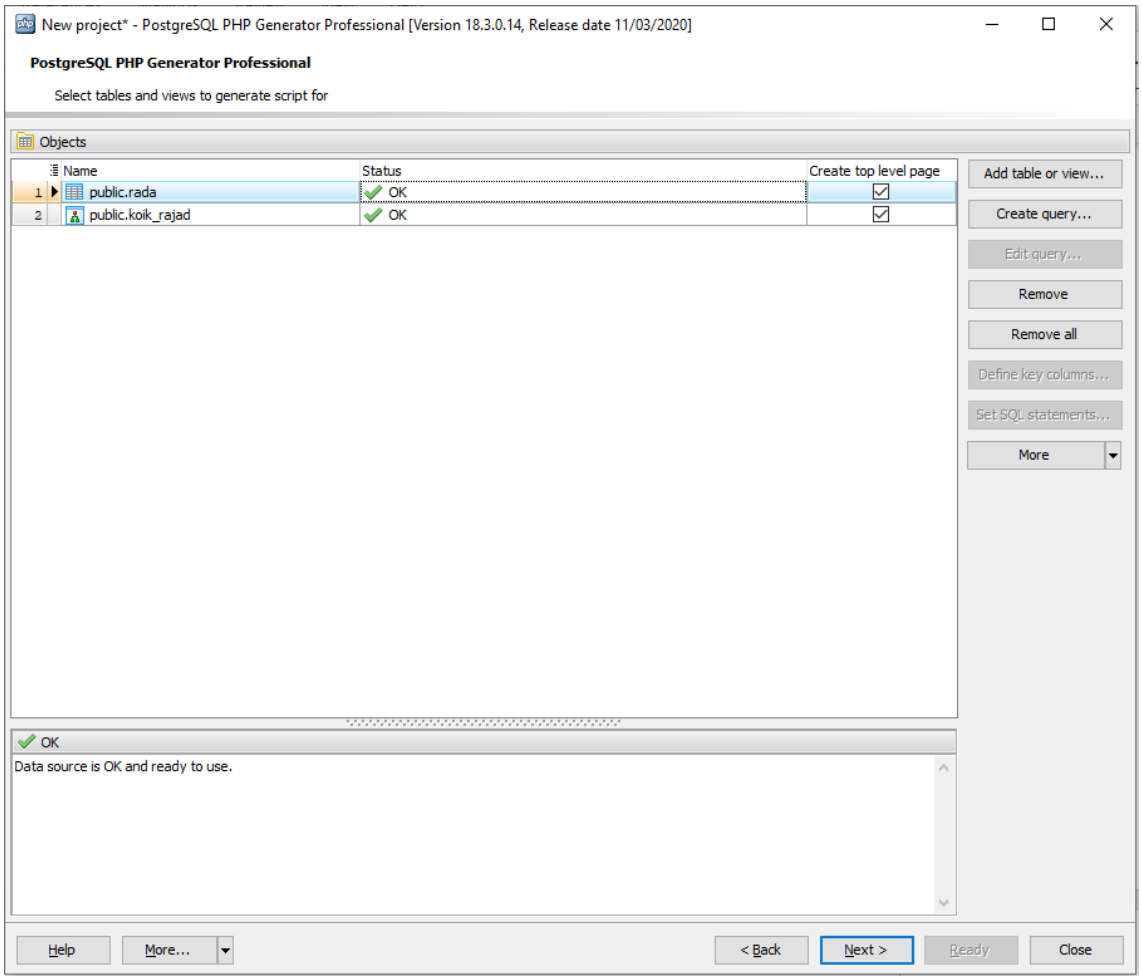


Joonis 29. PHPRad abil loodud rakendusel raja detailandmete leht koos alamlehega selle kategooriate vaatamiseks.

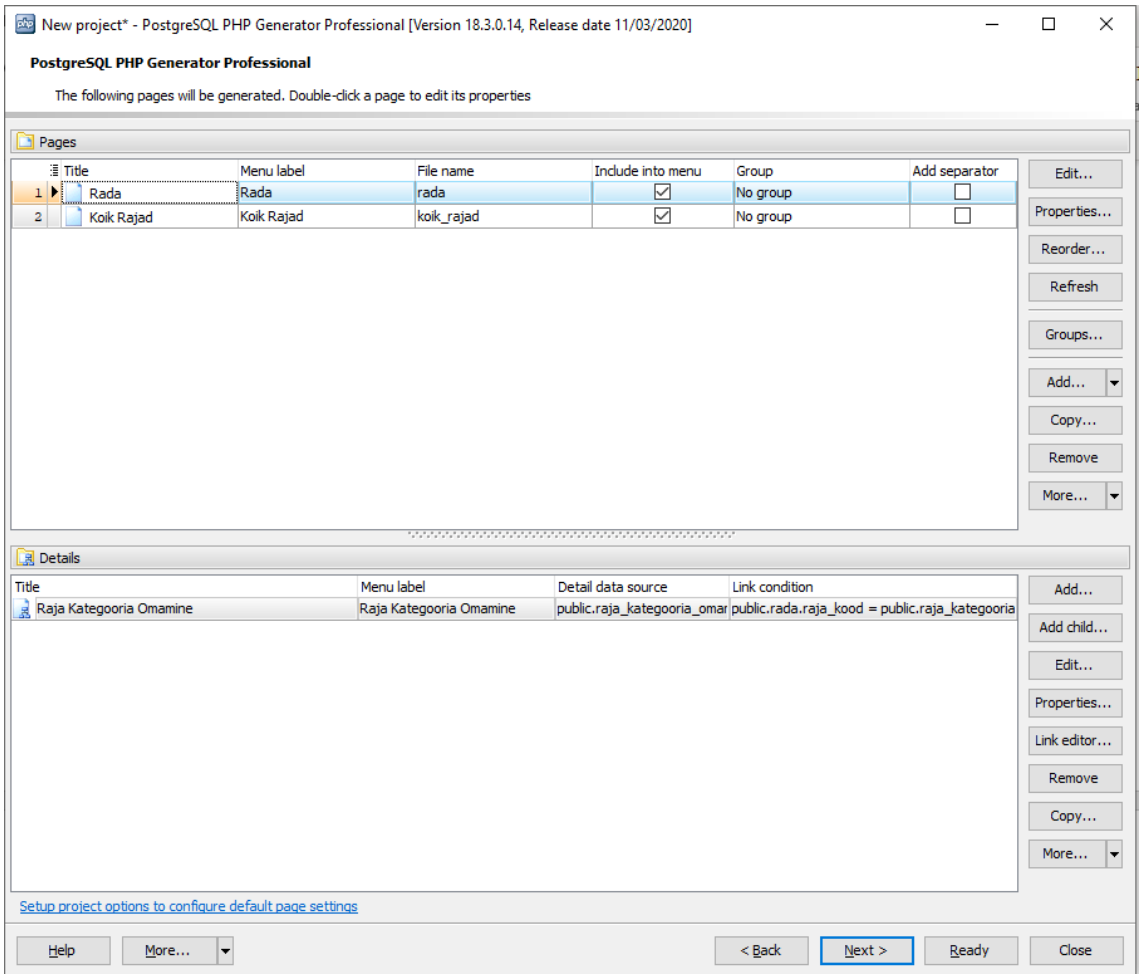
Lisa 2 – PostgreSQL PHP Generator keskkonna ekraanipildid



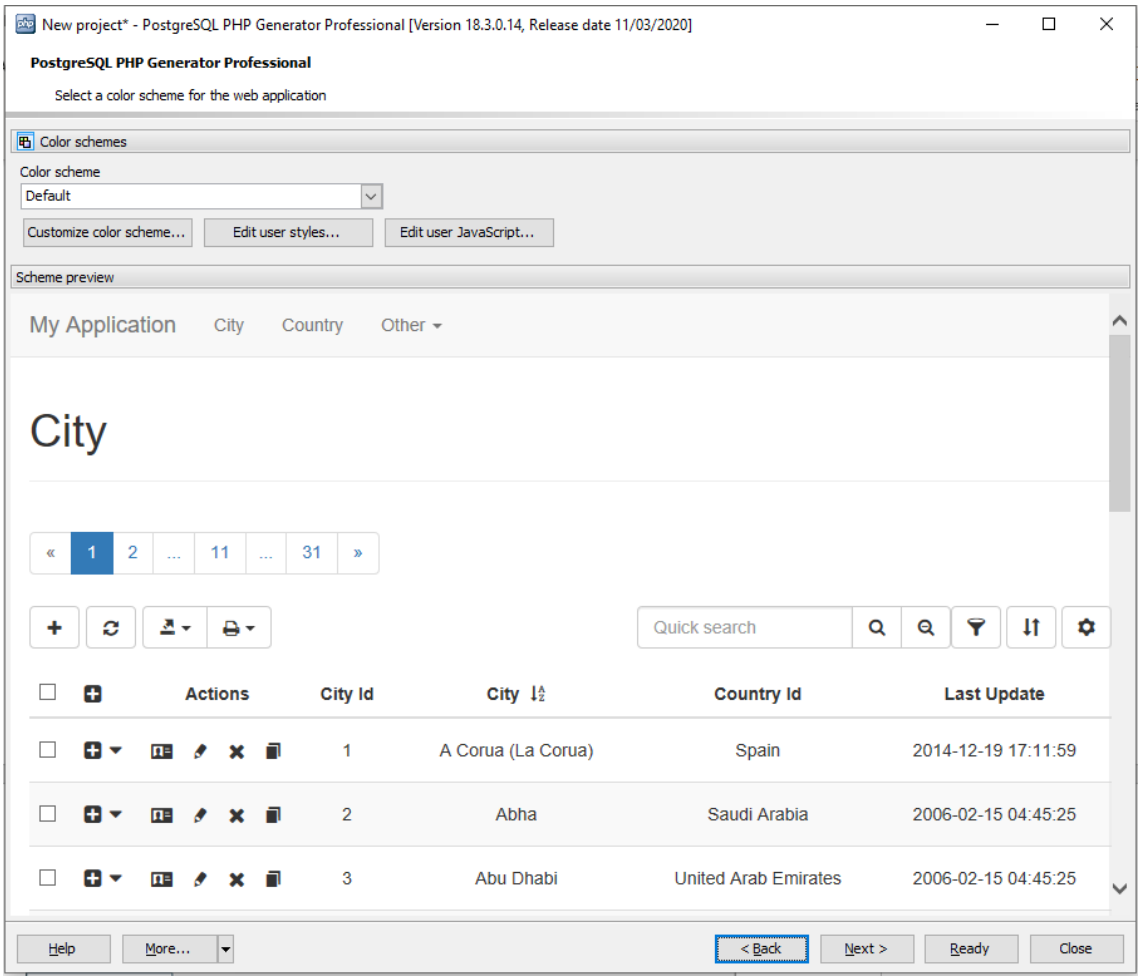
Joonis 30. PostgreSQL PHP Genarator. Andmebaasi seadistused.



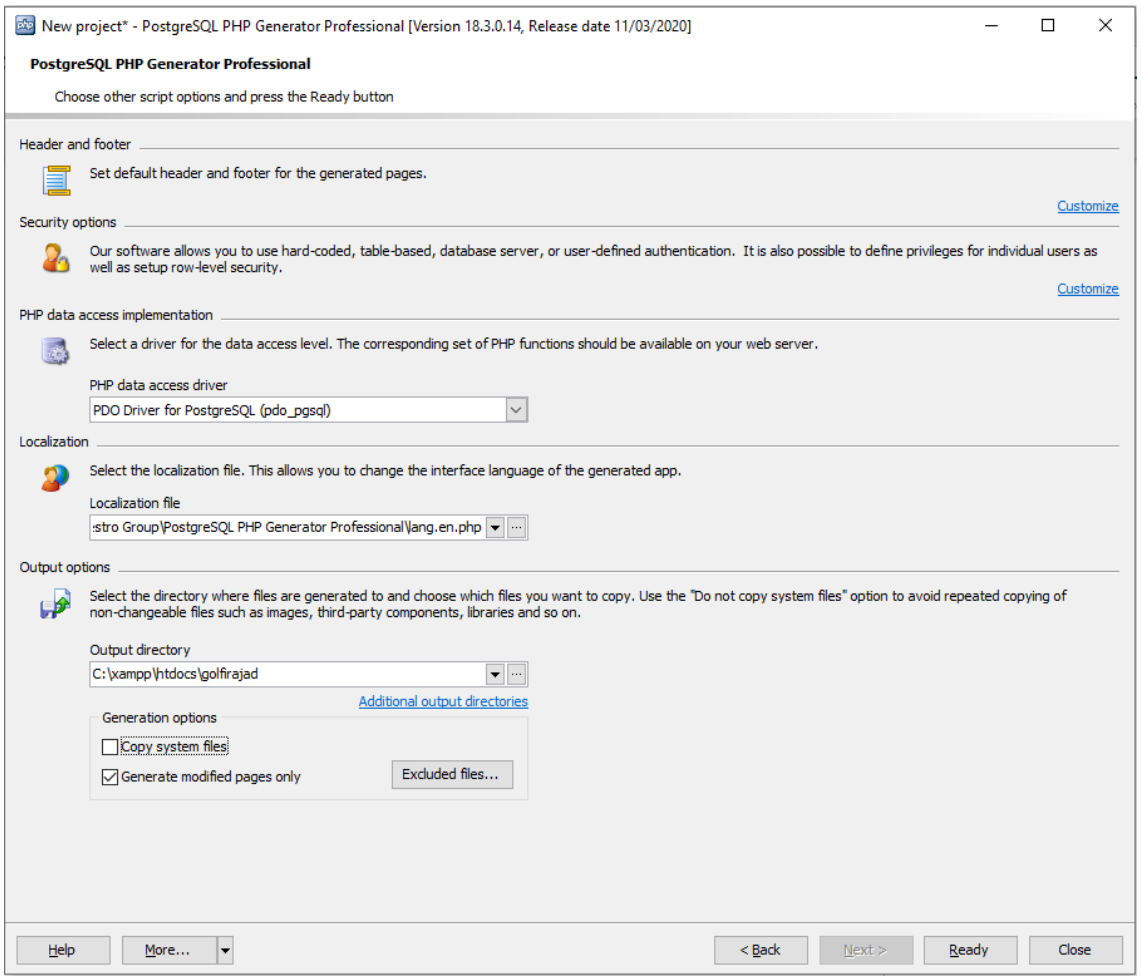
Joonis 31. PostgreSQL PHP Generator. Andmebaasiobjektide lisamine.



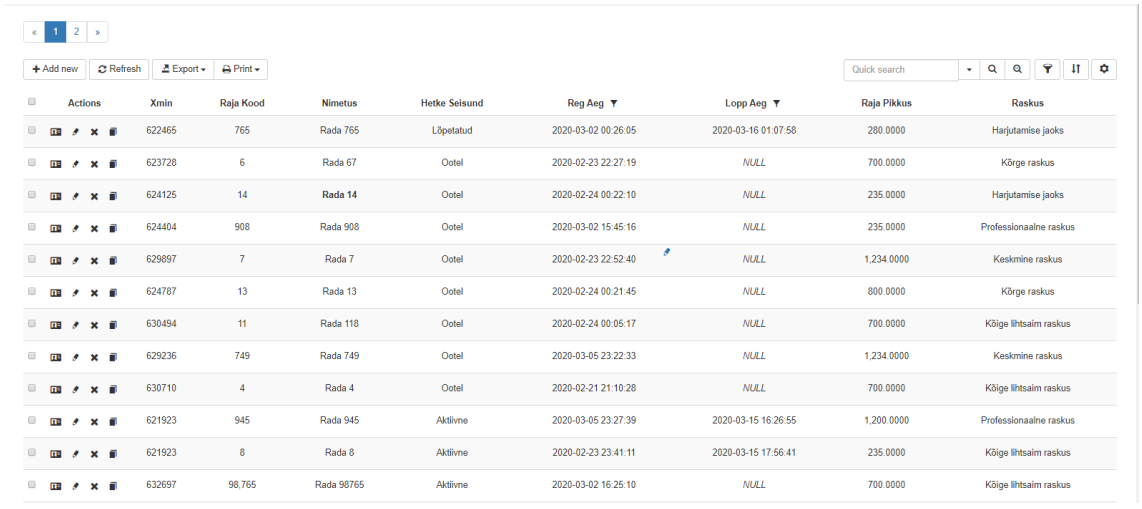
Joonis 32. PostgreSQL PHP Generator. Lehtede seadistamine.



Joonis 33. PostgreSQL PHP Genarator. Teema seadistamine.

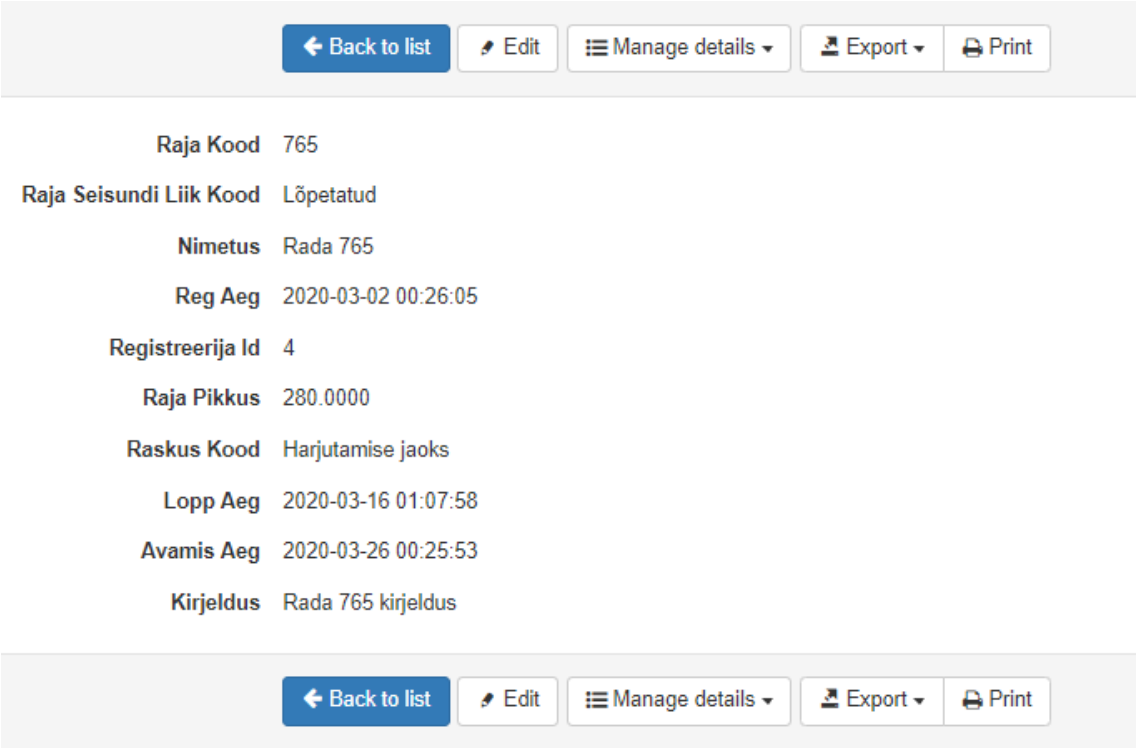


Joonis 34. PostgreSQL PHP Genarator. Viimane samm rakenduse seadistamisel.

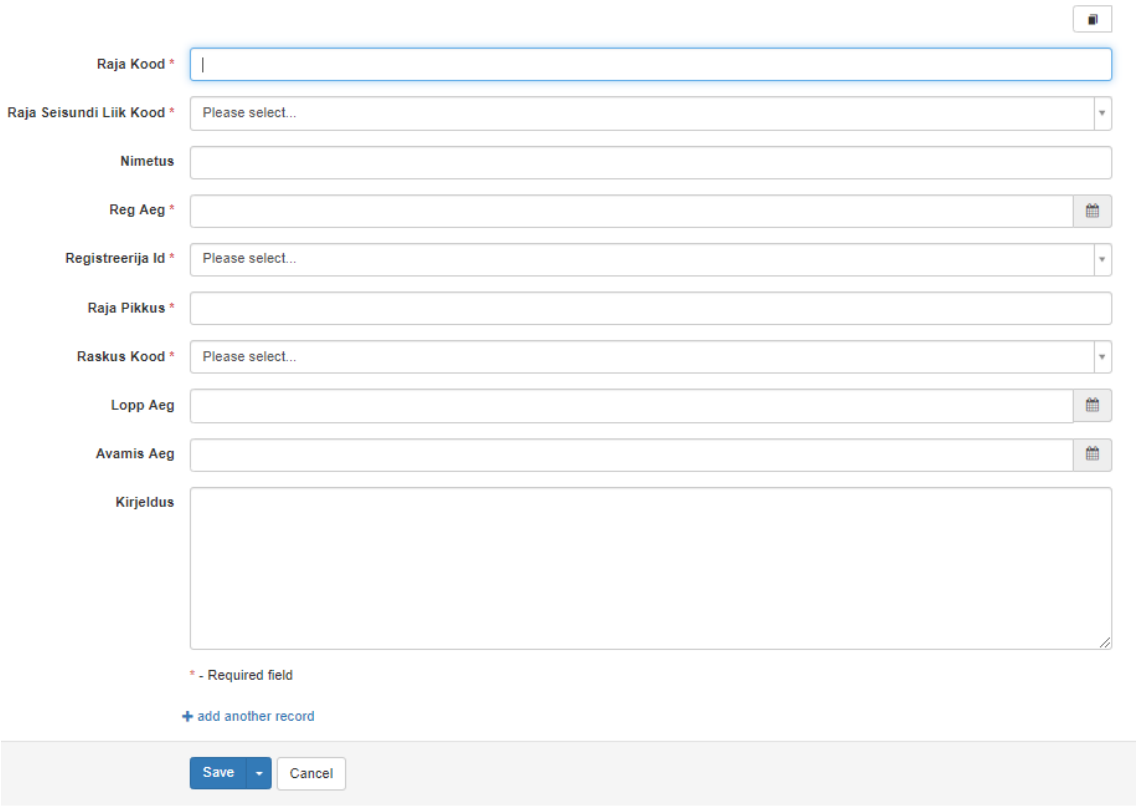


Actions	Xmin	Raja Kood	Nimetus	Hetke Seisund	Reg Aeg	Lopp Aeg	Raja Pikkus	Raskus
	622465	765	Rada 765	Lõpetatud	2020-03-02 00:26:05	2020-03-16 01:07:58	280.0000	Harjutamise jaoks
	623728	6	Rada 67	Ootel	2020-02-23 22:27:19	NULL	700.0000	Kõrge raskus
	624125	14	Rada 14	Ootel	2020-02-24 00:22:10	NULL	235.0000	Harjutamise jaoks
	624404	908	Rada 908	Ootel	2020-03-02 15:45:16	NULL	235.0000	Professionaalne raskus
	629897	7	Rada 7	Ootel	2020-02-23 22:52:40	NULL	1,234.0000	Keskmine raskus
	624787	13	Rada 13	Ootel	2020-02-24 00:21:45	NULL	800.0000	Kõrge raskus
	630494	11	Rada 118	Ootel	2020-02-24 00:05:17	NULL	700.0000	Kõige lihtsaim raskus
	629236	749	Rada 749	Ootel	2020-03-05 23:22:33	NULL	1,234.0000	Keskmine raskus
	630710	4	Rada 4	Ootel	2020-02-21 21:10:28	NULL	700.0000	Kõige lihtsaim raskus
	621923	945	Rada 945	Aktiivne	2020-03-05 23:27:39	2020-03-15 16:26:55	1,200.0000	Professionaalne raskus
	621923	8	Rada 8	Aktiivne	2020-02-23 23:41:11	2020-03-15 17:56:41	235.0000	Kõige lihtsaim raskus
	632697	98,765	Rada 98765	Aktiivne	2020-03-02 16:25:10	NULL	700.0000	Kõige lihtsaim raskus

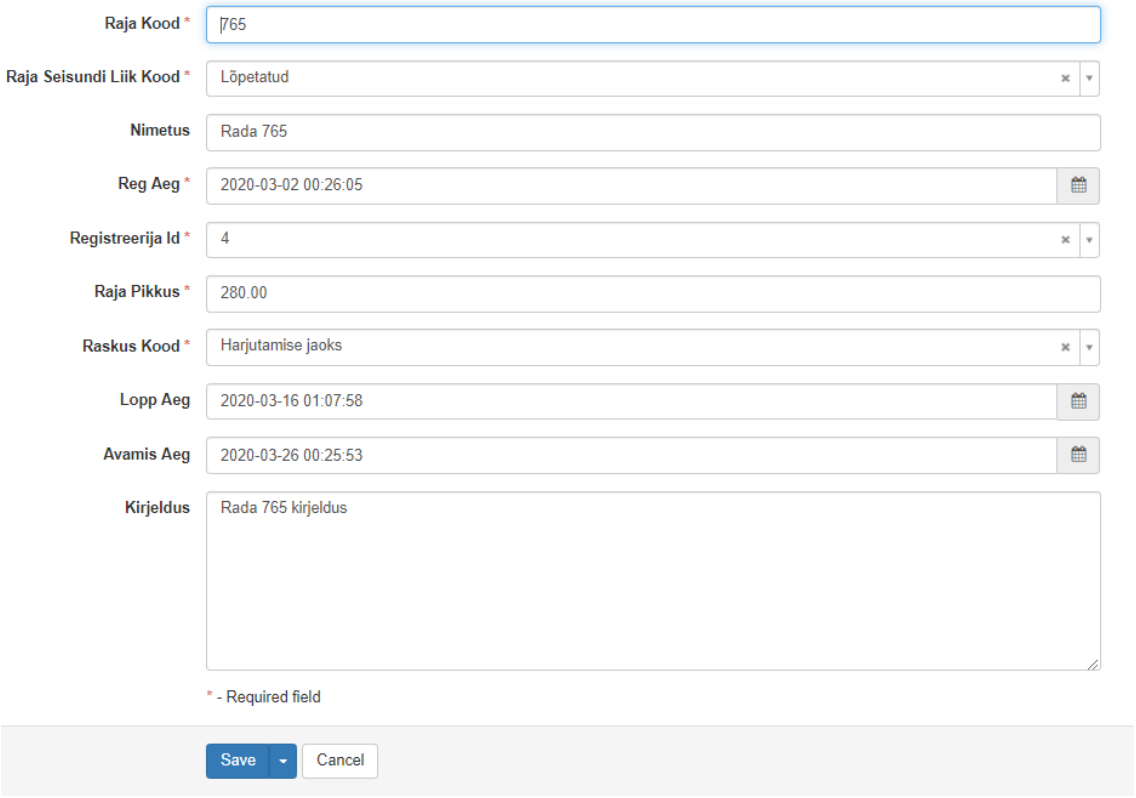
Joonis 35. PostgreSQL PHP Generator. Olemite nimekirja leht.



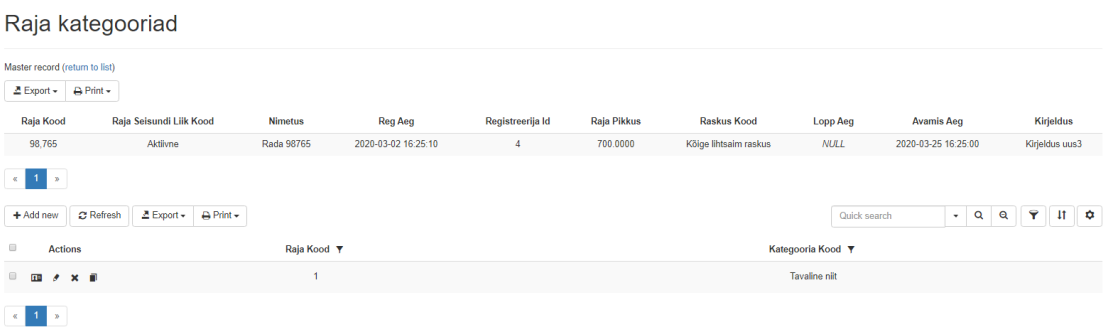
Joonis 36. PostgreSQL PHP Genarator. Olemi detailvaate leht.



Joonis 37. PostgreSQL PHP Generator. Olemi lisamise leht.



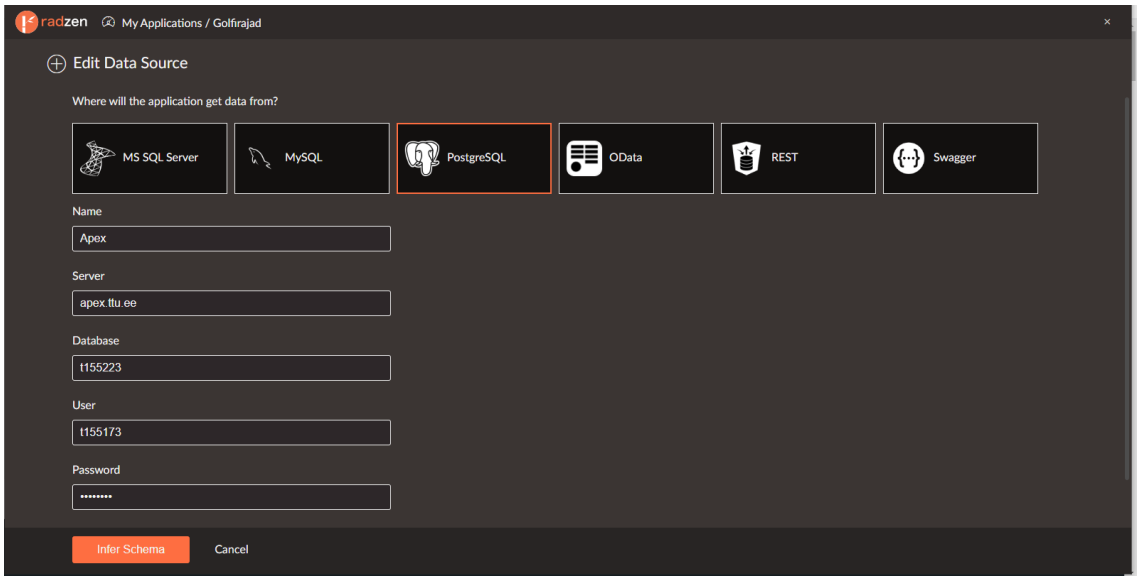
Joonis 38. PostgreSQL PHP Generator. Olemi muutmise leht.



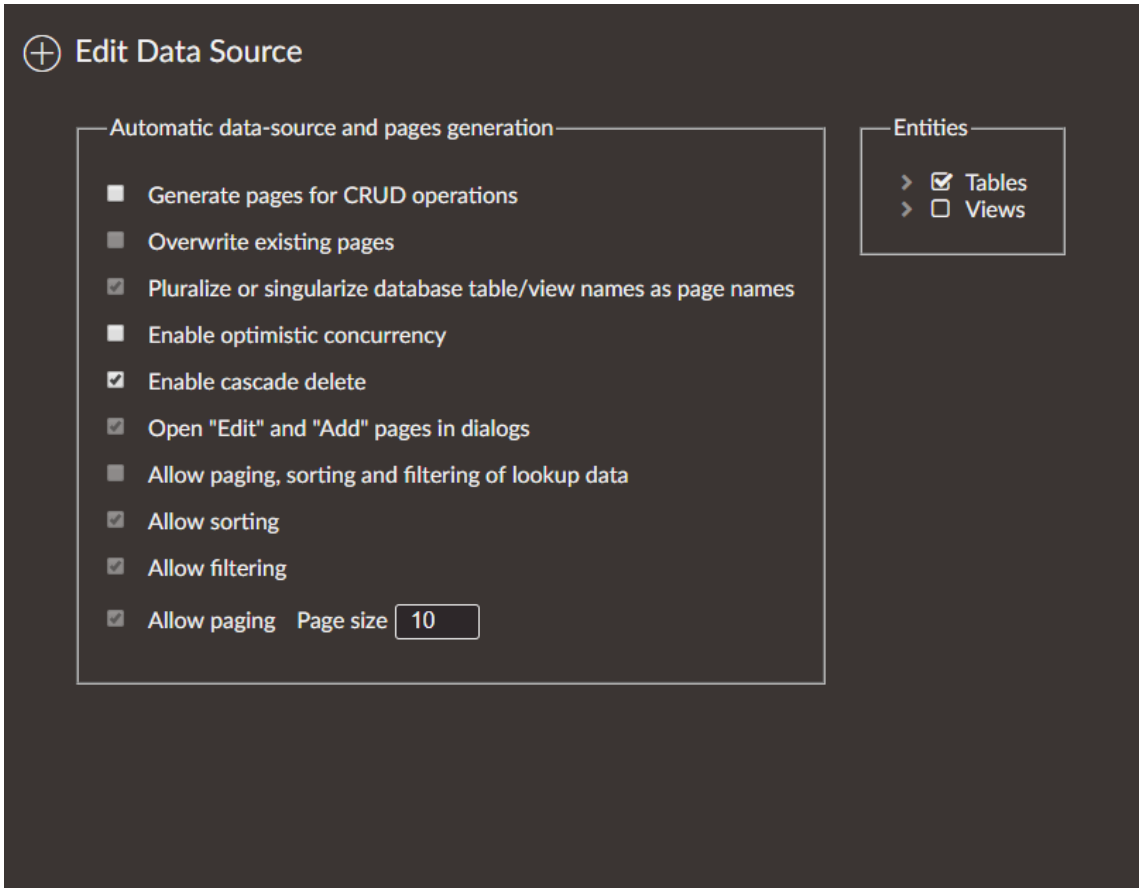
Joonis 39. PostgreSQL Generator. Alamraport.

Lisa 3 – Radzen keskkonna ekraanipildid

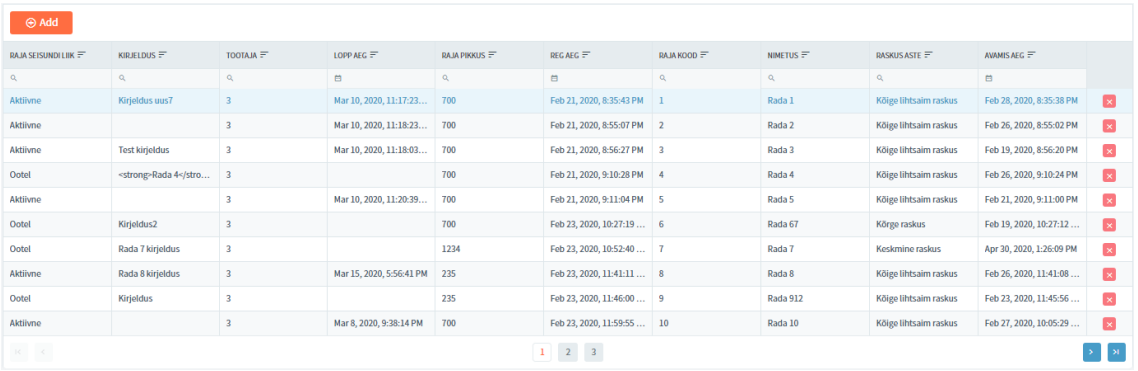
Joonis 40. Radzen. Rakenduse loomise leht.



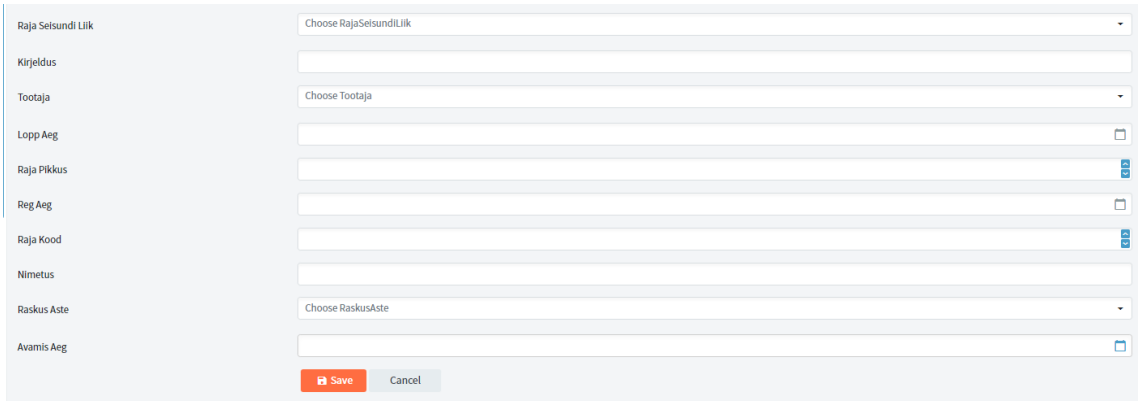
Joonis 41. Radzen. Andmeallika seadistamine.



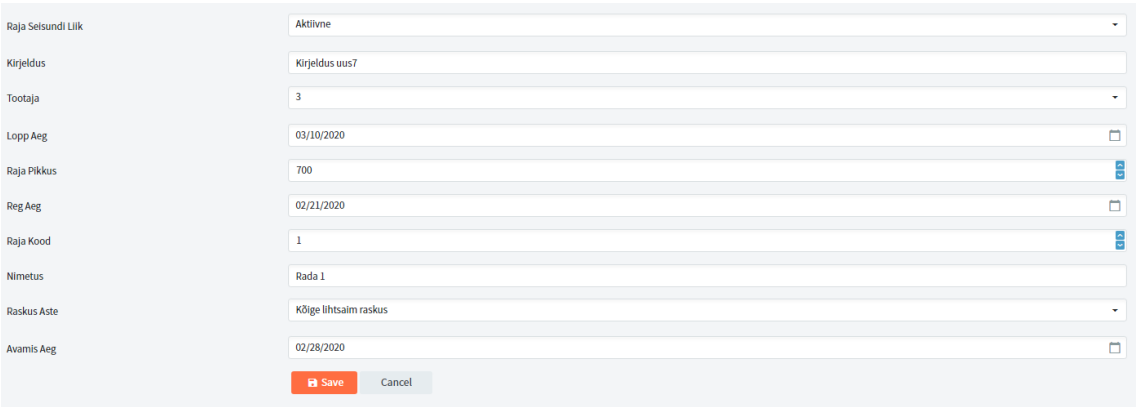
Joonis 42. Radzen. Andmeallika seadistamine. Osa 2.



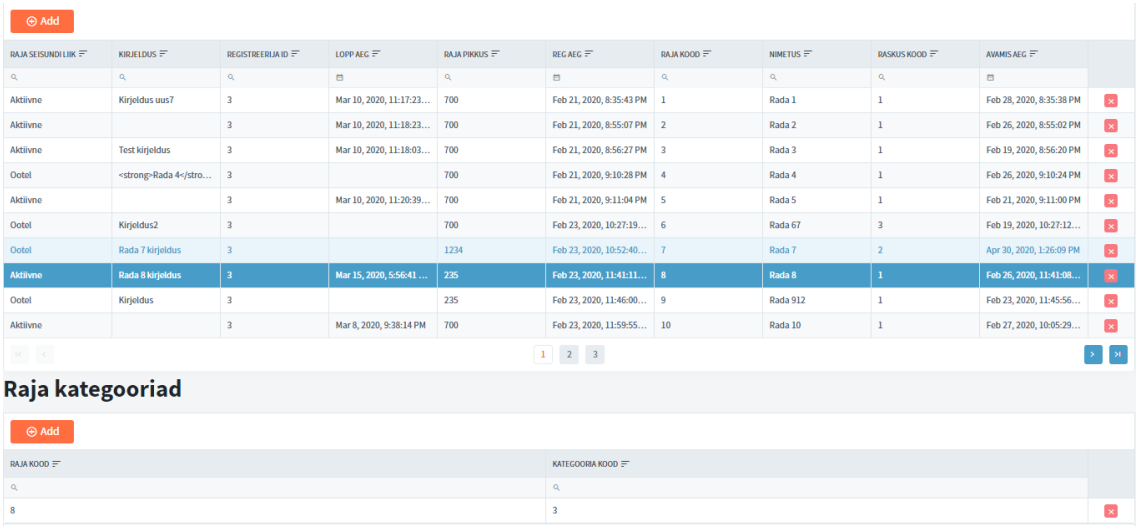
Joonis 43. Radzen. Olemite nimekirja leht.



Joonis 44. Radzen. Olemi lisamise leht.

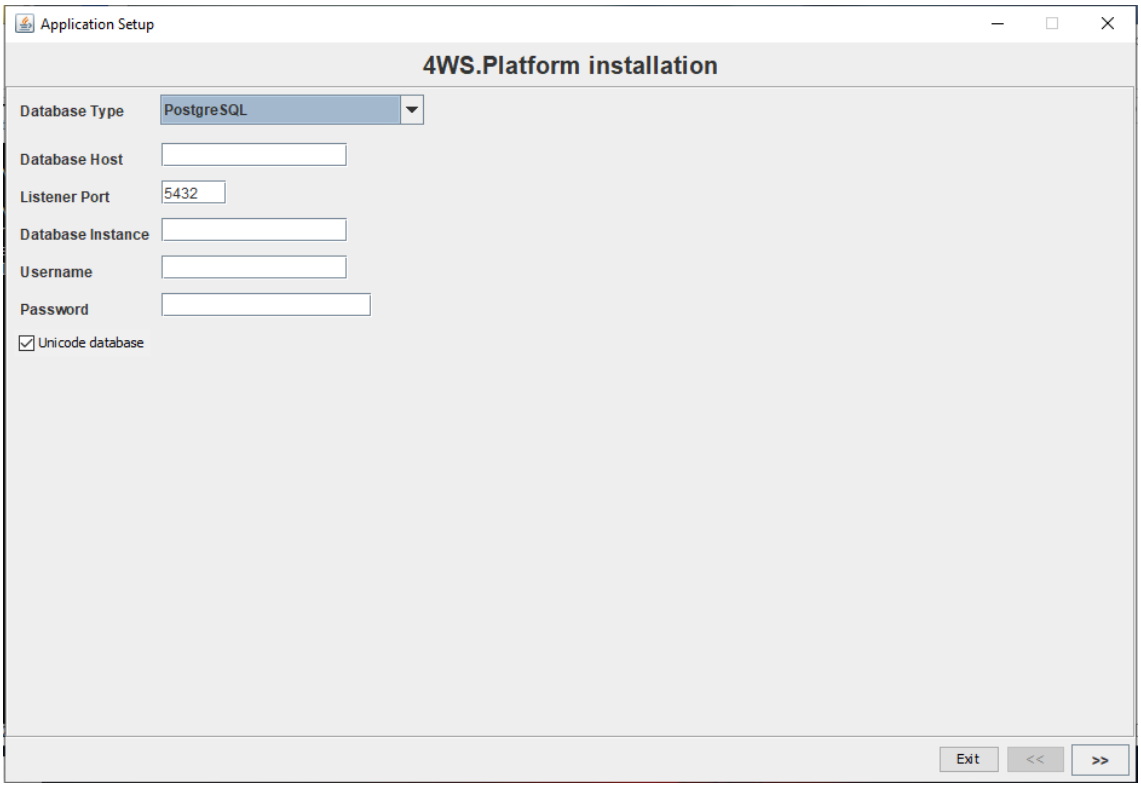


Joonis 45. Radzen. Olemi muutmise leht.

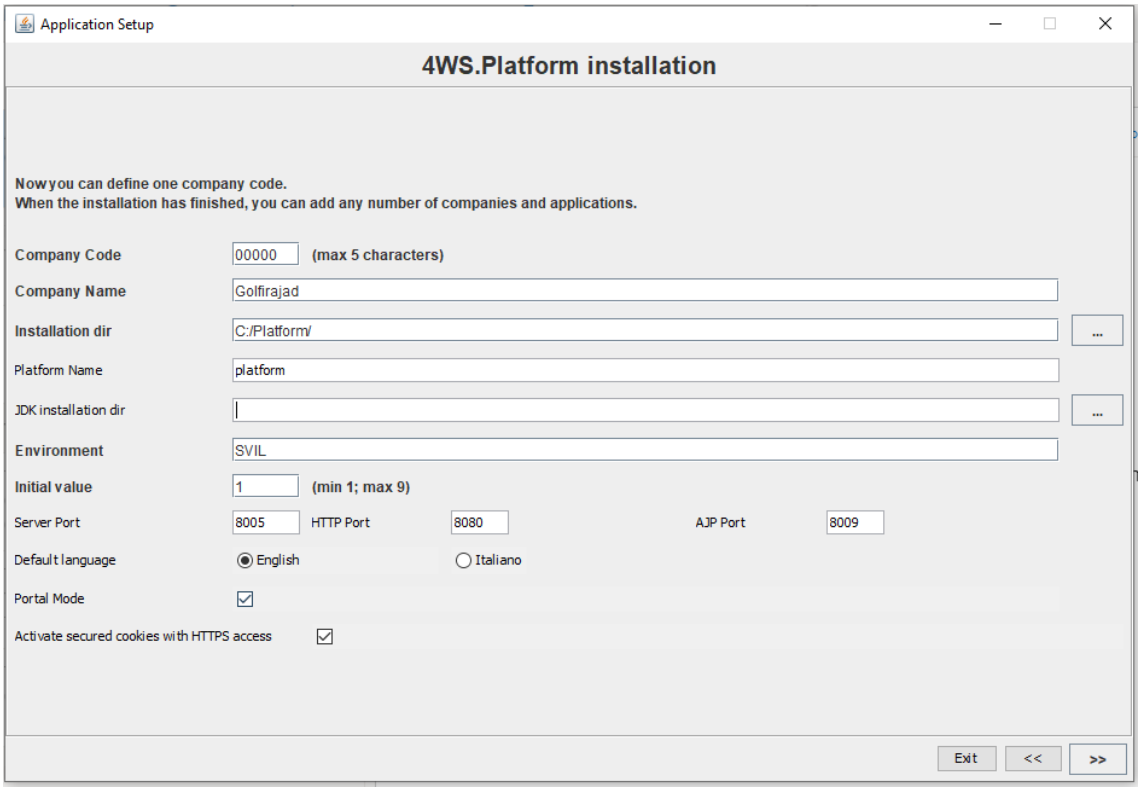


Joonis 46. Radzen. Olemite nimekiri koos alamraportiga.

Lisa 4 – 4WS Platform ekraanipildid

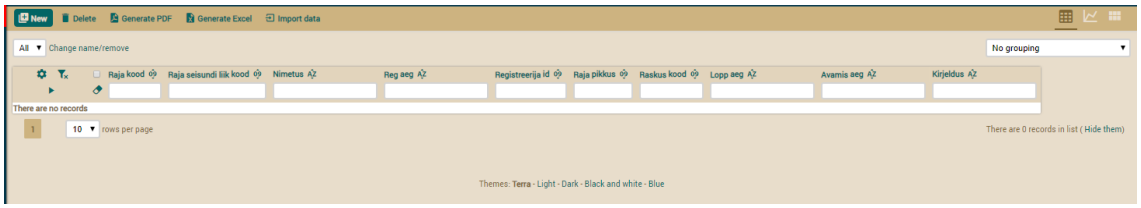


Joonis 47. 4WS Platvorm. Andmebaasi seadistused keskkonna paigaldamisel.

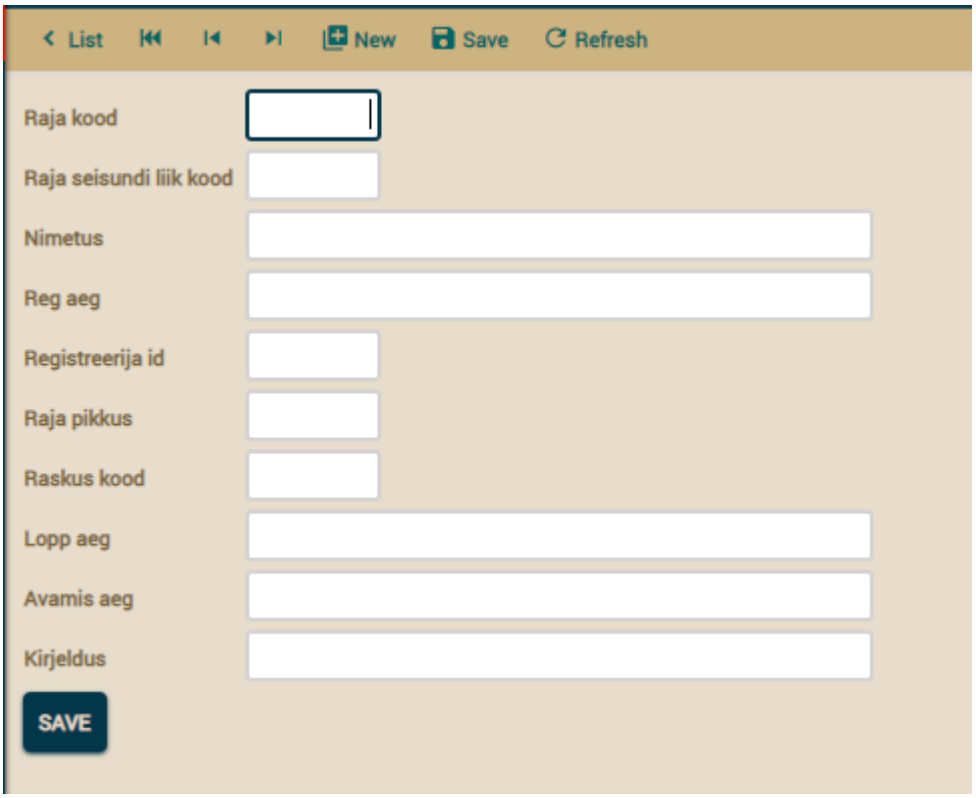


Joonis 48. 4WS Platform. Üldised keskkonna seadistused paigaldamisel.

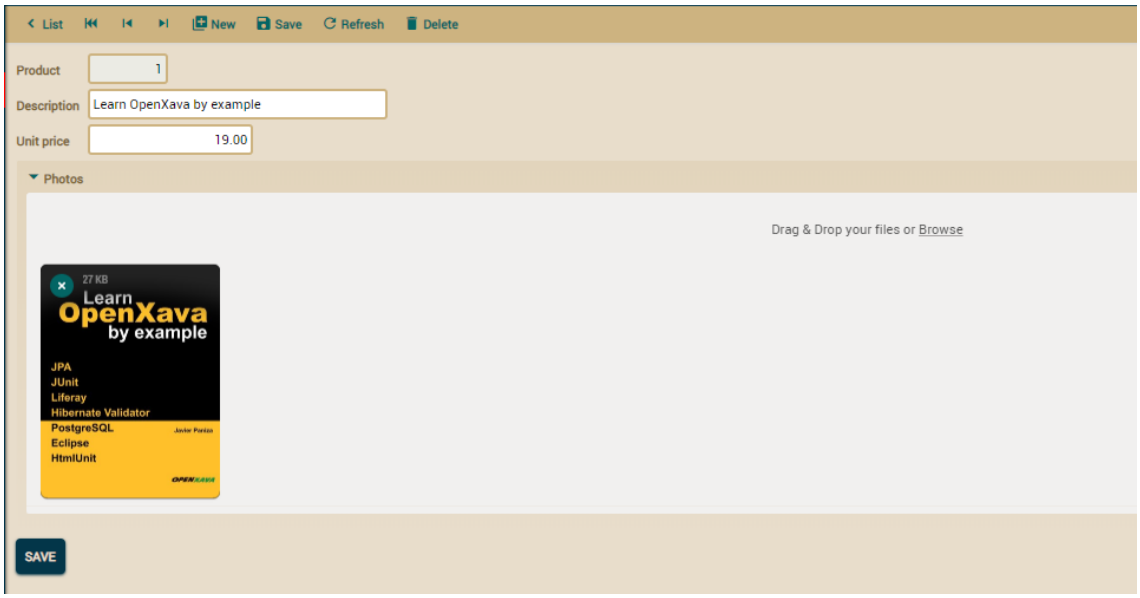
Lisa 5 – Openxava keskkonna ekraanipildid



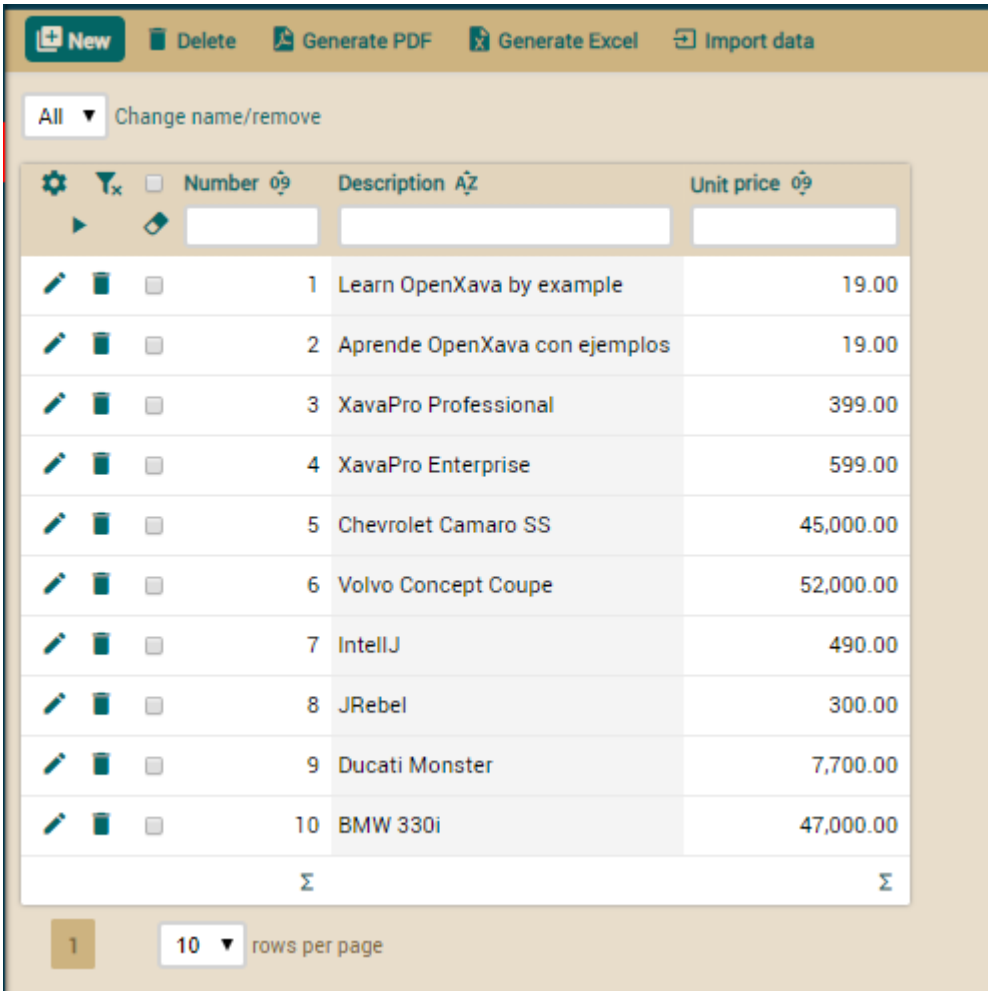
Joonis 49. Openxava. Olemite nimekirja leht.



Joonis 50. Openxava. Olemi lisamise leht.



Joonis 51. Openxava. Näidisrakenduse olemi muutmise leht.



Joonis 52. Openxava. Näidisrakenduse olemi nimekirja leht.