Tallinna Ülikool
Informaatika Instituut

Avalduste esitamise infosüsteem Peetri Lasteaed-Põhikooli näitel

Bakalaureusetöö

Autor: Martin Koidu
Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „ 2015
Juhendaja: ..... „ ..... „ 2015
Instituudi direktor: ..... „ ..... „ 2015

Tallinn 2015

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

…….. ……..

(kuupäev) (autor)

TALLINNA ÜLIKOOL

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina **Martin Koidu** (sünnikuupäev: **04.12.1987**)

1. annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "**Avalduste esitamise infosüsteem Peetri Lasteaed-Põhikooli näitel**", mille juhendaja on **Jaagup Kippar**, säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas, _____ 04.05.2015

Sisukord

Sissejuhatus

Infotehnoloogiliste vahendite integreerimisest õppetöösse on räägitud juba mitmeid aastaid. Selle tulemusel on valdav enamik klassides olemas projektorid, dokumendikaamerad ning innovaatilisemates koolides ka tahvelarvutid. Samuti kasutab suurem osa koolidest mingisugust veebikeskkonda, mille kaudu saavad lapsed vaadata, mis järgmiseks päevaks on jäetud kodus teha, lapsevanemad pääsevad ligi oma laste õpitulemustele ning kuhu õpetajad saavad märkida puudujaid ja õppuritele pandud hinnanguid.

Vaatamata sellele tuleb kooli sisseastumiseks, huviringi, pikapäevarühma või eelkooliga liitumiseks laadida kooli kodulehelt alla Microsoft Word'i dokument ning asuda seda täitma. Seejärel tuleb see viia paberkandjal kooli või paremal juhul õnnestub saata digitaalselt allkirjastatuna e-kirja teel kooli. Selline pilt vaatab üldiselt vastu, kui külastada Eesti koolide kodulehekülgi ning tutvuda kuidas nimetatud teenustest osa saada. Loomulikult on mõned üksikud uuendusliku mõtteviisiga koolid leidnud lahendusi nimetatud dokumentide elektrooniliseks esitamiseks, kuid need on pigem erandid.

Käesoleva töö autor on kindel, et võit, mida annaks saavutada infosüsteemi abil, ei seisne ainult mõnes paberipakis, mida oleks aasta jooksul seeläbi võimalik kokku hoida. Eelkõige võimaldaks süsteem töövoogu automatiseerides säästa koolis töötava personali aega ning teisi ressursse.

Teema valik ja aktuaalsus

Autor on töötanud eelnevalt Peetri Lasteaed-Põhikoolis ning samuti ühes Tallinna munitsipaalhuvialakoolis, mistõttu omab head ettekujutust koolis toimuvatest avalduste esitamisega seotud protsessidest. Sellest tulenevalt omab autor kõrgendatud huvi protsesside optimeerimisel, mis on olnud üheks suuremaks mõjutajaks käesoleva bakalaureusetöö teema valikul.

Loodava süsteemi implementeerimine algas autori poolt enne kui teema sai valitud lõputööks. Esimese tarne loodavast süsteemist oli autor juba tellijani (Peetri Lasteaed-Põhikool) toimetanud. Esimese kahe nädala jooksul tuli kaks päringut samas piirkonnas tegutsevast koolist sarnase veebikeskkonna kasutuselevõtu osas. See andis märku üldisemast vajadusest sellise süsteemi järgi, mistõttu on autor veendunud käesoleva töö teema aktuaalsuses.

Sisend ja eesmärk

Käesoleva bakalaureusetöö eesmärgiks on leida tarkvaratehniline lahendus enim kasutatavatele avaldustele ja taotlustele, millega puutuvad kokku nii kooli personal kui ka lapsevanemad. Lisaks püüab loodav süsteem lihtsustada esitatud avalduste haldamist ning toetada avaldustega seotud järelprotsesse, mille all peab autor silmas avalduste alusel laste jaotamist klassidesse, huviringi ja eelkooli rühmadesse. Lõputöö käigus välja töötatud lahendus on loodud tuginedes Peetri Lasteaed-Põhikooli poolt esitatud soovidele.

Töö peamiseks sisendiks on eelnevalt kaardistatud protsessid ning funktsionaalsed nõuded, mis on paika pandud koostöös Peetri Lasteaed-Põhikooli esindajatega läbi mitmete koosolekute, intervjuude, emailivahetuste ja Skype kõnede. Lahenduse väljatöötamisel on autor püüdnud juhinduda väleda (*agile*) arenduse põhimõtetest nii palju kui see on võimalik sellise projekti kallal üksinda töötades.

Struktuur

Töö põhiosa on jaotatud viieks peatükiks.

Esimeses peatükis antakse ülevaade taustast ning avalduste ja taotluste esitamise protsessidest enne veebipõhise süsteemi kasutuselevõttu.

Teises peatükis keskendutakse loodava rakenduse disainile ning planeerimisele. Samuti on kirjeldatud kolmandas peatükis süsteemile seatud nõuded ning piirangud.

Kolmas kirjeldab töö käigus loodud rakenduse implementatsiooni.

Neljas keskendub valminud rakenduse tulemuste analüüsile. Samuti vaadeldakse, millised suuremad probleemid lahenduse loomisel tekkisid.

Viies, ühtlasi ka viimane peatükk, tutvustab peamiseid edasiarenduse võimalusi.

1. Avalduste esitamisega seotud protsessid koolis

Käesolevas peatükis antakse ülevaade Peetri Lasteaed-Põhikoolist ning vaadeldakse nelja peamise kasutusel oleva avalduse tüübiga seonduvaid protsesse. Protsessid on pandud kirja tuginedes mitmetele intervjuudele kooli registripidaja (kooli vastuvõtu taotlused, eelkooli taotlused), sekretäri (kooli vastuvõtu taotlused) ja majandusjuhatajaga (pikapäevarühma taotlused).

1.1. Peetri Lasteaed-Põhikool

Peetri Lasteaed-Põhikool on Harjumaal, Rae vallas, Peetri alevikus asuv üldhariduskool, milles alustati õppetegevust aastal 2009, ning mille õpilaste arv oli toona 111. (Tooming, 2012)

Kooli kõrval tegutseb paralleelselt ka lasteaed, kuid kuna käesoleva töö skoobis ei ole sees lasteaeda puudutavat funktsionaalsust, siis on autor arvestanud analüüsis ning nõuete kaardistamisel ainult kooli soovidega.

Esialgselt oli koolimaja ehitatud 9-klassiliseks, kahe paralleelklassiga (kokku 18 klassikomplekti) põhikooliks, mis tähendab, et kooli mahutavus oli 432 õpilaskohta. Seoses Peetri aleviku rahvastiku kiire kasvuga suurenes ka koolikohtade arv, mis tõi 2010/2011 ja 2011/2012 õppeaastatel kaasa vajaduse avada esimeses kooliastmes kolm klassikomplekti. 2012-2014 aasta arengukavas leiti, et seoses koolikohtade arvu puudusega on vaja teha olemasolevale koolikompleksile juurdeehitus. Lepinguid juurdeehituse tarbeks selle arengukava kinnitamise hetkeks veel sõlmitud ei olnud. (Peetri Lasteaed-Põhikool, 2012)

Juurdeehitus valmis 2013. aasta sügisel, mis tõi juurde 216 uut koolikohta, misjärel sai kooli uueks mahutavuseks 648 kohta. (DELFI, 2013)

Rae vald on Statistikaameti 2014. aastal korraldatud uuringu järgi Eesti kõige noorema rahvastikuga vald. Valla elanike keskmine vanus on 33,2 aastat, mis on ligikaudu 9

aastat vähem kui Eesti elanikkonna keskmine vanus. (Servinski, Hänilane, Kivilaid, Tischler, & Ülle)

Rahvastiku madal vanus tähendab omakorda seda, et Peetri koolis õppivate laste vanemad on võrreldes Eesti keskmisega tunduvalt nooremad. Noorem põlvkond tunneb end uute tehnoloogiatega mugavamalt kui vanemad, mis loob väga hea pinnase uue infosüsteemi kasutuselevõtul.

1.2. Eelkooli taotluste esitamine

Eelkooli taotluste ja gruppidesse jaotamisega tegeleb koolis registripidaja. Oma olemuselt on tegemist kõige lihtsamat tüüpi taotlusega. Avaldusel küsitakse lapse, lapsevanema andmeid ning seda, kas lapsel on õigus osaleda eelkoolis ilma osalustasu maksmata (vt lisa 5). Peale eelkoolis osalejate kinnitamist kooli registripidaja poolt ning vastava raporti esitamist vallale, esitatakse lapsevanematele, kelle laps ei ole osalustasust vabastatud, iga kahe kuu tagant arve.

Eelkooli taotluse esitamise protsess on järgmine:

1. lapsevanem täidab ära kooli kodulehelt saadud taotluse ankeedi (vt lisa 5);
2. lapsevanem edastab täidetud taotluse kas paberil või digitaalselt allkirjastatuna e-maili teel kooli registripidajale;
3. kooli registripidaja kogub kokku ja prindib välja kõik laekunud avaldused;
4. vastava kuupäeva kätte jõudmisel jaotatakse lapsed rühmadesse jälgides järgmisi kriteeriume:
    a. poiste ja tüdrukute osakaal ühes rühmas võimalikult võrdne;
    b. ühe pere lapsed samasse rühma;
    c. lasteaias ühes rühmas käinud lapsed võimalusel ühte rühma;
    d. sama nimega lapsed eraldi rühmadesse;
5. registripidaja saadab valla raamatupidajale nimekirja lapsevanematest, kellele tuleb esitada arve.

Kirjeldatud protsessi puhul tuleb registripidajal tegeleda arvestataval määral andmesisestusega.

1.3. Kooli sisseastumise taotluste esitamine

Seoses Peetri aleviku elanikkonna arvu kiire suurenemisega avatakse õppeaastal 2015/2016 samuti kuus esimest klassi. See tähendab, et ainuüksi esimesse kooliastmesse asub õppima ligikaudu 144 õpilast. Üldiselt läheb avalduste esitamise protsess lahti aprillikuu keskel.

Tavapärane protsess kooli sisseastumise avalduste esitamisel on järgmine:

1. lapsevanem täidab ära kooli kodulehelt saadud eelkooli taotluse ankeedi (vt lisa 7);
2. lapsevanem edastab täidetud taotluse kas paberil või digitaalselt allkirjastatuna e-maili teel kooli registripidajale;
3. kooli registripidaja kogub kokku ja prindib välja kõik laekunud avaldused;
4. Kõikidest avaldustega kaasnevatest dokumentidest tehakse koopia;
5. vastava kuupäeva kätte jõudmisel jaotatakse lapsed klassidesse jälgides järgmisi kriteeriume:
    a. lapse elukoha aadressi (kooli poolt teenindatavates piirkondades elavatel lastel on eesõigus);
    b. poiste ja tüdrukute osakaal ühes klassis võimalikult võrdne;
    c. ühe pere lapsed samasse klassi;
    d. lasteaias ühes rühmas käinud lapsed võimalusel ühte rühma;
    e. sama nimega lapsed eraldi klassidesse;

Kõige kiirem periood kooli vastuvõtu taotluste esitamisel on kaks esimest päeva. Kuna avalduste esitamisega seotud töökoormus on esimestel päevadel väga suur, siis on selleks puhuks sekretär ja kooli registripidaja koopereerunud istudes ühes ruumis koos ning võttes vastu avaldusi. Kuna esimesse kooliastmesse avatakse 6 klassikomplekti ning taotlusi tuleb isegi siis rohkem kui on kohti, siis esimestel päevadel võib kohati olla lausa järjekord sekretäri ukse taga.

Töö on korraldatud nii, et kooli sekretär võtab lapsevanema käest vastu avalduse ning registripidaja teeb samal ajal koopia seonduvatest dokumentidest. Kuigi selline tööde

jaotus on üsna optimaalne, siis taotluste esitamise esimestel päevadel ei ole neil aega teha oma igapäevast tööd.

1.4. Pikapäeva rühma avalduste esitamine

Pikapäevarühmas osalevad ainult esimese klassi lapsed. Peetri Lasteaed-Põhikoolis on 2014/2015 õppeaastal esimesi klasse kuus paralleeli, mis tähendab, et kokku on potentsiaalseid osalejaid pikapäevarühmas ligikaudu 144 õpilast.

Avaldused esitatakse üldiselt paberkandjal kuhu märgitakse muuhulgas, mis päevadel ja kellaaegadel laps pikapäevarühmast osa võtab ning kas ta soovib süüa ka sooja toitu.

Pikapäevarühma avaldustega tegeleb kooli majandusjuhataja, kes iga kuu lõpus esitab Rae valla raamatupidajale sööjate aruande.

1.5. Huviringidega liitumise avalduste esitamine

2013/2014 õppeaastal laekus huviringi astumise taotluste esitamise esimese 10-15 minuti jooksul huvijuhi e-maili aadressile ligikaudu sada avaldust. Lisaks esitati väiksem osa avaldusi ka paberil.

Reeglid huviringidesse vastuvõtul

Selleks, et igal lapsel oleks võimalus osaleda huvitegevuses, on ringidega liitumisele kehtestatud hulk reegleid, mida soovijate vastuvõtul arvestatakse.

Osadesse ringidesse võetakse eelisjärjekorras lapsi vastu eelmise aasta osalejate seast (kitarriring, plokkflööt jne). Teine grupp ringe on jällegi sellised, kuhu võetakse eelmisel aastal mitteosalenud lapsi eesmärgiga anda võimalus teistele soovijatele.

Mõned ringid on sellised, kuhu võetakse vastu kõik, kes soovi avaldavad. Selliste ringide hulka kuuluvad peamiselt erinevad laulukoorid (mudilaskoor, tütarlastekoor, poistekoor jne).

Enamik huviringide töö käib mitmes grupis, mis on üldiselt moodustatud vanusegruppide järgi (näiteks 1.-3. klass, 4.-9. klass jne) olenevalt sellest, kui palju vastavas vanuseastmes õpilasi kokku on.

Huviringi taotluste esitamise ja rühmadesse jaotamise protsess

Enne huviringidega liitumise taotluste esitamise algust pannakse kooli kodulehele üles nimekiri kõikidest avatavatest huviringidest koos tutvustusega, et vanemad ja lapsed saaksid eelnevalt valikuga tutvuda.

Kindla kuupäeva hommikul kell üheksa pannakse üles huviringi taotluse blankett, mille enamus lapsevaemad täidavad ära digitaalselt ning saadavad kooli huvijuhi e-mailile.

Taotluse laekudes prindib huvijuht selle välja, märkides paberile esitamise kellaaja.

Peale valdava enamuse taotluste esitamist alustab huvijuht ringides osalejate jaotamist gruppidesse, mis põhimõtteliselt näeb välja nii, et erinevad ringid ja grupid kirjutatakse suurele tahvlile. Iga grupi alla märgitakse soovijad vastavalt esitatud avaldustele, järjestatuna esitamise kellaaja järgi.

Ringidesse jaotamisel jälgitakse, et iga laps saaks võimalusel osaleda vähemalt ühe huviringi töös. Kui taotlusel on märgitud teise valikuna ring, milles on vähe huvilisi, siis võib üks laps pääseda ka mitmesse ringi.

Selleks, et kõik lapsed saaksid jaotatud soovitud ringidesse, kulus 2013/2014 õppeaastal ligikaudu nädal aega tööd.

Kuna kooli õppima asujate arv 2014/2015 õppeaastal oli võrreldes eelmise aastaga tunduvalt suurem ning ringe samuti rohkem (vt lisa 17), siis oli oodata ringidega seotud toimetamise töömahu suurenemist.

2. Loodava rakenduse planeerimine

Ei ole harv juhus kui klient tuleb tarnija juurde viimasel minutil sooviga arendada midagi uut ja revolutsioonilist võimalikult lühikese ajaga. Sarnane stsenaarium oli ka käesoleva süsteemi planeerimisel.

2.1. Esimene koosolek

Esimene kontakt Peetri kooli esindajatega toimus 28. augustil 2014. Minnes kohtumisele kohale oli teada, et kool soovib uut ja esinduslikku veebilehekülge, ei midagi keerulisemat. Autoril polnud sellest kohtumisest alguse saanud süsteemi suurusest sel hetkel veel aimugi. Kuigi kohalolijate täpne koosseis jäi fikseerimata mäletab autor kooli direktori, õppejuhi, arendusjuhi, infojuhi, registripidaja, lasteaia juhi ja lasteaia õppejuhi kohalolu.

Kaardistades erinevate töötajate soove ja ideid, selgus, et lihtsa kooli kodulehe asemel soovitakse süsteemi, mille kaudu saaksid lapsevanemad liituda huviringidega, registreerida lapsi eelkooli, broneerida ruume, hallata klasside tegevuskavasid.

Kuna kooliaasta oli just algamas ning aega arenduseks niigi küllaltki vähe, siis sai koheselt pandud paika kuupäevad ning paari sõnaga ka funktsionaalsus, mis selleks kuupäevaks valmis pidi olema.

Loodava funktsionaalsuse kuupäevad said paika järgmiselt:

1. pikapäevarühmas osalemise taotlused koos söömisega – 2. september;
2. eelkooli taotluste esitamine – 8. september;
3. huviringi avalduste esitamine – 22. september.

Kuna pikapäevarühma avalduste esitamine pidi saama alguse juba 2. septembril, siis tähendas see seda, et esialgne pikapäevarühma avalduste esitamise funktsionaalsus pidi olema läbinud analüüsi, testimise, arendamise ja kooskõlastamise tellijaga kõigest kuue päevaga.

2.2. Nõuded

Käesolevas peatükis antakse ülevaade peamistest funktsionaalsetest ning mittefunktsionaalsetest nõuetest. Kirjeldatud nõuded ei ole pandud detailselt paika enne süsteemi implementatsiooni algust vaid on arenenud ning täpsustunud pidevalt arendusprotsessi vältel. Osa esimesel koosolekul kõlanud funktsionaalsustest sai tarkvara loomise planeerimisel esialgsest skoobist välja jäetud.

Nõuete analüüsis on autor jaganud nõuded kaheks – funktsionaalsed ja mittefunktsionaalsed nõuded. Funktsionaalsete nõuete eesmärgiks on kirjeldada, mida loodav süsteem peab tegema.

Mittefunktsionaalsete nõuete all peetakse silmas kõiki ülejäänud nõudeid, mis ei ole otseselt seotud mingi konkreetse funktsionaalsusega, kuid omavad väga suurt tähtsust loodava tarkvara kvaliteedi tagamisel. Mittefunktsionaalseid nõudeid võib käsitleda ka kui piirangud. (Stellman, 2009)

2.2.1. Funktsionaalsed nõuded

2.2.1.1. Kasutajakontod

Kõik süsteemis olevad isikud peavad omama personaalset kontot. Isikute alla käivad ka lapsed. Järgnevalt on kirjeldatud kasutajakontodega seotud funktsionaalsused.

Kasutajaks registreerimine

Kasutajaks registreerimisel tuleb rakendada mehhanisme, millega tagada, et ühel isikul oleks süsteemis ainult üks konto.

Kasutajate autentimine

Selleks, et süsteemi kasutada, peavad isikud olema süsteemi sisse logitud. Kasutajad saavad enda autentimiseks ning sisse logimiseks kasutada kolme erinevat viisi:

1. kasutajanime ja parooli kasutades;
2. ID-kaarti kasutades;

3. Mobiil-ID-d kasutades

Kadunud parooli taastamine

Juhul kui kasutajal on kas ununenud või ära kadunud parool, siis peab olema tal võimalus taastada kadunud parool.

Olemasoleva kasutaja info muutmine

Iga kasutaja saab muuta endaga seotud infot. Süsteemis tuleb rakendada meetmeid, mis valideeriks, et kasutaja sisestaks korrektsed andmed.

Kindlasti peab süsteem võimaldama järgmiseid automaatseid toiminguid:

1. isikukoodi korrektsuse kontroll;
2. võimaldama sisestada elukoha aadressi standardsel kujul
   (näiteks: *[Linn/alevik/küla], [vald], [maakond] [tänava nimi] [maja number]-[korteri number]*)

Kasutajate rollid ja õigused

Kõik süsteemi lisandunud kasutajad on automaatselt lapsevanema rollis. See tähendab, et nad saavad teha järgmiseid toiminguid:

1. enda info muutmine;
2. laste lisamine;
3. avalduste esitamine

Täiendav õiguste süsteem peab võimaldama anda kasutajatele õiguseid mingi konkreetse toimingu või siis terve hulga toimingute tegemiseks. Selleks tarbeks tuleks ära defineerida süsteemi kasutajate rollid ning õigused. Roll tähendab sisuliselt kasutajate gruppi, millel on mingi teatav hulk õiguseid erinevate toimingute teostamiseks.

Peale baasrolli saab kasutajatele lisada veel järgmiseid rolle:

1. **administraator**

- omab ligipääsu kõigele ja saab teha kõike;

2. **majandusjuhataja (vt lisa 3)**
    - saab võtta süsteemist välja pikapäevarühma sööjate tabelit;
    - saab teostada otsingut süsteemis olevate laste ning lastevanemate hulgas ning vaadata nende kontaktandmeid, avaldusi ning infot pikapäevarühma söömise kohta;
3. **registripidaja (vt lisa 2)**
    - saab võtta välja eelkooli taotluste ja kooli astumise taotluste raportit;
    - saab teostada otsingut õpilaste ning lapsevanemate hulgas;
    - saab märkida lapse koolist väljaarvatuks;
      väljaarvamisel lõpetatakse automaatselt kõik hetkel kehtivad avaldused
4. **huvijuht**
    - huviringi astumise avalduste kinnitamine/tagasilükkamine;

Kasutajakaart

Kasutajakaart on sisuliselt veebileht, millelt näeb kogu selle kasutaja kohta süsteemis olevat infot. Kasutajakaardil kajastatud info oleneb sellest, kas tegemist on lapse või lapsevanema kontoga. Funktsionaalsus on ette nähtud kasutamiseks kooli töötajatele.

Kasutajate otsing

Tegemist on funktsionaalsusega, mis võimaldab kiirelt teostada otsingut lapsevanemate ja laste kontode seast. Otsida saab peamiste kasutajainfo väljade järgi (nimi, isikukood, email, telefon). Funktsionaalsus on kättesaadav ainult kooli töötajatele.

2.2.1.2. Avaldused

Avalduste funktsionaalsus on loodava süsteemi üks põhifunktsionaalsustest, mis hõlmab endas kõike, mis on seotud avalduste esitamisega kuni avalduse kinnitamiseni välja.

Avalduse esitamise eelduseks on see, et kasutaja on süsteemi lisanud vähemalt ühe lapse.

Süsteemi kaudu saab esitada järgmisi avaldusi:

1. eelkooli avaldus;
2. pikapäevarühma avaldus;
3. huviringis osalemise avaldus;
4. kooli vastuvõtu avaldus

2.2.1.3. Huviringid

Huviringide moodul on tihedalt seotud avalduste mooduliga kuna huviringiga liitumise eelduseks on esitatud ning kinnitatud avaldus.

Huviringid võib jaotada järgmisesse kahte suuremasse gruppi:

1. **automaatse aktsepteerimisega ringid;**
   siia alla kuuluvad grupid, mille puhul ei ole vaja teostada huvijuhi poolt ringiga liitumise avalduse ülevaatamist.
2. **manuaalse aktsepteerimisega ringid;**
   siia alla kuuluvad ringid, mille puhul ei rakendata süsteemis automaatset ringidesse jaotamist. Huviringi juurde peab olema lisatud isik, kes ringide avaldusi kas aktsepteerib või tagasi lükkab.

2.2.1.4. Grupid

Käesoleva bakalaureusetöö raames tehtavat funktsionaalsust arvesse võttes kuuluvad siia alla huviringide grupid, pikapäeva rühmad, klassid (nt. „1A klass“).

Grupid võib oma olemuse järgi jaotada kaheks:

1. **vaba toimumisega grupid**
   selliste gruppide alla kuuluvad grupid, millel ei ole määratud kindlat toimumise

aega ega päeva. Näiteks „1A klass“, „eelkool“ jne;

2. **kindlal ajal ning päeval toimuvad grupid**
   grupid, mille liikmed käivad koos kindlatel päevadel ning kindlal kellaajal. Üheks kindlal kellaajal toimuvate gruppide näiteks on mitmesugused huviringid.

2.2.1.5. Aruandlus ja andmete eksport

Süsteemist väljastatavad aruanded on kas CSV või Excel formaadis. Kuna ei ole täpselt teada, milliseid aruandeid täpselt vaja on, siis peaks lahendus võimaldama aruandeid muuta võimalikult vähese vaevaga.

Esialgsed süsteemist väljastatavad aruanded on järgmised:

1. eelkooli taotluse esitanute aruanne;
2. kooli sisseastumise taotluse esitanute aruanne;
3. pikapäeva rühma sööjate aruanne;
    1. käesoleva nädala sööjad;
    2. sööjate kuu koondaruanne;

2.2.2. Mittefunktsionaalsed nõuded

2.2.2.1. Turvalisus (security)

Loodav süsteem peab olema võimalikult turvaline.

- autentimine ID-kaardi ja mobiil-IDga;
- võrguühendus peab käima läbi SSL turvaprotokolli

2.2.2.2. Jõudlus (Performance)

Süsteem peab vastu pidama vähemalt 400 samaaegselt süsteemis toimetava kasutajaga. Andmehulga kasvades ei tohi süsteem muutuda märgatavalt aeglasemaks.

2.2.2.3. Kasutatavus (usability)

Tänapäeval ei saa ükski veebirakendus hakkama ilma, et pööraks tähelepanu kasutatavusele.

Kasutatavuse all mõistetakse üldjuhul süsteemi kasutamise lihtsust ning seda kui kerge on kasutajatel ära õppida selle kasutamine. Kasutatavuse alla kuuluvad õpitavus (*learnability*), efektiivsus (*efficiency*), meeldejäävus (*memorability*), vead (*errors*), rahulolu (*satisfaction*). Tegemist on nii laialdase valdkonnaga, et sellest on saanud lausa omaette teadusharu. (Nielsen, 2012)

Käesoleva töö raames valmiva infosüsteemi eesmärgiks on võetud kasutajale võimalikult lihtsa ja loogilise töövoo võimaldamine. Tuleb vältida kasutaja sattumist segadusse mõne toimingu läbiviimisel.

Infole ligipääs ning toimingud peavad olema esitatud võimalikult loogiliselt nii, et süsteemi kasutamine oleks intuitiivne ning kasutaja ei sattuks segadusse.

Tagasiside aeg, mis kulub kasutaja tehtud klikist kuni selleni, et süsteem sellele vastavalt reageeriks, ei tohi peab olema võimalikult lühike.

2.2.2.4. Varundus (backup)

Loodava rakenduse puhul peab olema minimeeritud tarkvara või riistvara rikkest tulenevate probleemide tõttu tekkiv andmete kadu. Rakenduse andmebaasi varundus peab toimuma automaatselt.

Andmete varunduseks peab olema töötav taastusplaan.

3. Teostus

Loodud süsteemi teostusel on püütud järgida väleda (*agile*) arenduse ühte peamist printsiipi, milleks on "töötav tarkvara esimesest päevast peale". Samuti on kogu projekti vältel olnud tähtsal kohal suhtlus kliendiga ning seeläbi ka kiire tagasiside lisandunud funktsionaalsusele.

Käesolevas peatükis annab autor ülevaate peamistest kasutatud vahenditest, tehnoloogiatest ning süsteemi teostuse tähtsamatest eripäradest. Nende valikul on autor võtnud arvesse eelkõige oma eelnevat kogemust tarkvaraarenduse valdkonnas. Kõik kasutatavad tehnoloogiad on kaasaegsed ning laialdaselt levinud.

3.1. Arenduskeskkond ja vahendid

Kuna kogu loodava süsteemi esimene tarne pidi toimuma juba kaks nädalat peale esialgset kontakti kliendiga, siis oli üheks eesmärgiks arenduskeskkonna ülesseadmisel võimalikult vähene ajakulu.

Peamine arenduseks vajaminev serveritarkvara ning andmebaasimootor oli autori isiklikus arvutis olemas ning eelnevalt seadistatud teiste projektide tarbeks. Vaatamata sellele selgus lähemal uurimisel, et mitmed arenduseks vajalikud moodulid olid siiski puudu. Samuti oleks süsteemi loomiseks kasutusele võetud veebiraamistik nõudnud PHP versiooni uuendamist, mis oleks omakorda loonud uue probleemi kuna mõned varasemalt autori poolt alustatud ning arendatavad projektid töötasid jällegi vanema PHP versiooniga.

3.1.1. Vagrant

Selleks, et eelnevalt kirjeldatud probleeme vältida, otsustas autor võtta kasutusele eraldi virtuaalmasina. Olles tutvunud erinevate võimalustega, sai otsustatud viimase aasta jooksul populaarsust kogunud vabavaralise projekti nimega Vagrant kasuks.

Vagrant on 2010. aasta jaanuaris algust saanud projekt, mille eesmärgiks on lihtsustada terviklike virtuaalsete arenduskeskkondade loomist. Projekti looja on Mitchell Hashimoto, kes arendas kõnealust projekti esimesed kolm aastat põhitöö kõrvalt. Seejärel lõi ta eraldi firma, mis hakkas täies mahus tegelema Vagrant'i arenduse, lisade välja töötamise, koolituse ning klienditoega. (HashiCorp Inc., 2015)

Vagrant kujutab endast tööriista, mis võimaldab lihtsalt luua ning hallata virtuaalmasinaid ilma, et peaks kulutama arvestaval hulgal aega nende konfigureerimise peale kuni selleni, et arendaja saaks reaalselt hakata keskkonnas tegelema arendustööga. (Hashimoto, 2013)

Kuna Vagrant ise ei ole virtualiseerimise platvorm, siis peab täiendavalt olemas olema mõni vastavat teenust pakkuv platvorm (*provider*). Vaikimisi kasutab Vagrant Oracle VirtualBox'i kuna see on vabavaraline ja seetõttu kasutajatele kõige lihtsamini kättesaadav. Lisaks võib soovikorral kasutada VirtualBox'i asemel VMware't või KVM'i. (HashiCorp, 2015)

Erineva konfiguratsiooniga virtuaalmasinaid, mida kutsutakse kastideks (*boxes*) ning mille saab Vagrant'iga lihtsalt kasutusele võtta, on liikvel väga palju erinevaid[1]. Kui on tegemist projektiga, mille raames kasutatakse väga spetsiifilist tarkvara, siis on mõistlik luua selleks puhuks eraldi kast.

Uue arendaja lisandudes projekti ei ole tal vaja teha muud kui lisada kast enda masinasse kasutades käsku `vagrant box add`.

Käesoleva töö raames loodud infosüsteemi arendamisel on autor kasutanud Laravel raamistiku ametlikku kasti "homestead". Kasti lisamine käib kasutades käsku `vagrant box add laravel/homestead`. (Otwell, Laravel Homestead, 2015)

[1] Ametlik kataloog saadaval olevatest virtuaalmasinatest asub järgmisel aadressil: https://atlas.hashicorp.com/boxes/search

Kasti lisamisel tehakse automaatselt ära järgmised toimingid:

- virtuaalmasina loomine olenevalt valitud operatsioonisüsteemist;
- masina füüsiliste parameetrite (RAM, protsessori tuumade arv jne) seadistamine;
- võrguühenduse seadistamine nii, et loodud virtuaalmasin on kättesaadaval kõikidelt teistelt samas võrgus paiknevatelt seadmetelt;
- jagatud kaustade (*shared folders*) seadistamine nii, et arendaja võib töötada failidega peremeesmasinas (*host*) ning muudatused on koheselt saadaval ka serveris

(Hashimoto, 2013)

Kasti paigaldamise järel tuleb seadistada projektid, mille arendamisel masinat kasutatakse. Seadistamiseks tuleb sisestada käsurealt `homestead edit`, mille peale avatakse süsteemi vaikimisi tekstiredaktoris konfiguratsioonifail (vt ekraanipilt 1).

Konfiguratsioonifailis määratakse ära järgmised parameetrid:

- masina sisevõrgu IP-aadress;
- virtuaalmälu ja protsessori tuumade arv;
- SSH kasutamiseks võtmete failid (võtmed peavad olema eelnevalt genereeritud);
- millised on jagatud kaustad (*folders*). Peremeesmasina kaust ning selle ühenduspunkt (*mount point*) virtuaalmasinas;
- masina unikaalne nimi võrgus ja selle juurkataloog (*folders*);
- kasutatav(ad) andmebaas(id) (*databases*);
- keskkonnamuutujad (*variables*)

```yaml
---
ip: "192.168.10.10"
memory: 2048
cpus: 1

authorize: ~/.ssh/id_rsa.pub

keys:
    - ~/.ssh/id_rsa

folders:
    - map: ~/Code
      to: /home/vagrant/Code

sites:
    - map: intra.peetri.dev
      to: /home/vagrant/Code/pintra/public

databases:
    - homestead

variables:
    - key: APP_ENV
      value: local
```

Ekraanipilt 1 – arendusmasina konfiguratsioonifail

Selleks, et masinale nime järgi ligi pääseks, tuleb teha peremeesmasinas paiknevas "hosts" failis vastendus (*mapping*) virtuaalmasina IP-aadressi ja nime vahel.

Peale "hosts" failis muudatuse tegemist on virtuaalmasin valmis ning jääb üle masin käivitada. Masina alglaadimiseks piisab käsust `homestead up`, mille käivitamist ja väljundit näeb ekraanipildil 2.

```
⌘ ~/ homestead up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'laravel/homestead' is up to date...
==> default: Resuming suspended VM...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection refused. Retrying...
==> default: Machine booted and ready!
⌘ ~/
```

Ekraanipilt 2 – arendusmasina alglaadimine ja väljund

Peale masina käivitamist saab siseneda masinasse käsuga `homestead ssh`. (vt ekraanipilt 3)

```
⌘ ~/ homestead ssh
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Apr 28 07:42:14 UTC 2015

  System load:  0.05              Processes:           95
  Usage of /:   5.8% of 39.34GB   Users logged in:     0
  Memory usage: 38%               IP address for eth0: 10.0.2.15
  Swap usage:   0%                IP address for eth1: 192.168.10.10

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

Last login: Tue Apr 28 07:42:15 2015 from 10.0.2.2
vagrant@homestead:~$
```

Ekraanipilt 3 – arendusmasinasse sisenemine

Selleks, et peale töö lõpetamist süsteemist korrektselt välja logida, tuleb kasutada UNIX operatsioonisüsteemides kasutusel olevat käsku `exit`, misjärel võib masina sulgeda käsuga `homestead halt`.

3.1.2. Versioonihaldus

Iga vähegi suurema süsteemi arendusel tuleks kasutada versioonihaldussüsteemi. Versioonihaldussüsteem võimaldab salvestada failidega tehtud muudatused ning hiljem soovi korral taastada ükskõik millisel ajahetkel talletatud mistahes faili sisu.

Versioonihaldus on peaaegu möödapääsmatu, kui projekti kallal töötab rohkem kui üks arendaja, ja kui üritada vältida probleeme, mis tekivad näiteks siis, kui mitu arendajat peavad töötama ühe ja sama failiga. Versioonihaldus võimaldab vaadata ajalugu, kes millises failis millist rida on muutnud. Milline oli rea sisu enne muudatuse tegemist ja

milline pärast. Eelnevalt mainituga ei piirdu versioonihaldussüsteemide [2] funktsionaalsus. (Azad, 2007)

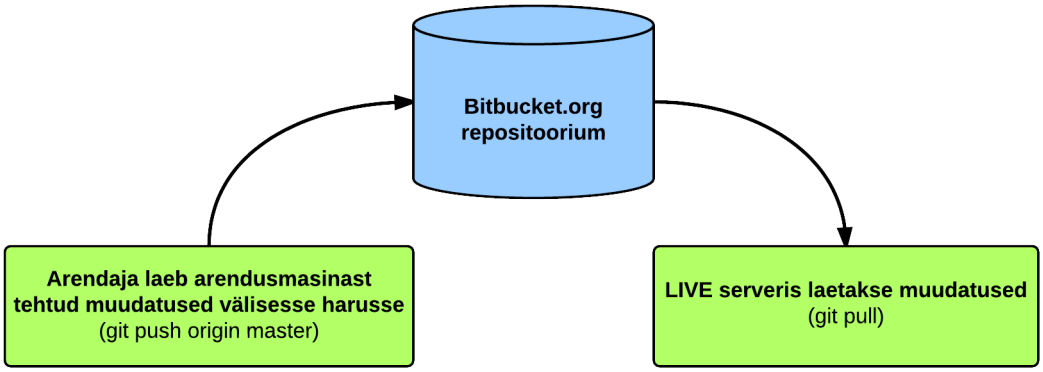
Versioonihaldussüsteeme on olemas mitmeid, kuid tuginedes enda kogemusele on autor käesoleva töö raames loodava rakenduse koodi hoidmiseks valinud vabavaralise lahenduse nimega Git.

Git on hajutatud (*distributed*) versioonihaldussüsteem, mis sai alguse aastal 2005 seoses vajadusega leida Linux kernel'i arendamiseks kasutatud versioonihaldussüsteemi asemele uus. Uut ei leitud kuna Linux'i arendusega seotud kogukond, eesotsas nimetatud operatsioonisüsteemi looja Linus Torvaldsiga, otsustas luua uue ja kaasaegse versioonihaldussüsteemi. Loodava süsteemi võtmeteguriteks said kiirus, mittelineaarse arenduse tugi (võimalus luua paralleelseid harusid), suutlikus hallata suuri projekte, ning hajutatus. Hajutatus tähendab, et projekti lähtekood ei asu mitte ühes keskses serveris, millest arendajad omale soovitud versioonid failidest alla laevad, vaid et kõikides arvutites on olemas terve koodibaas koos kõikide versioonidega projektis eksisteerivatest failidest. Selle lahenduse pluss on see, et keskse serveri riknemine ei too endaga kaasa versioonihaldussüsteemi kadu. (Chacon & Straub, 2014)

Käesolevas töös kasutab autor Giti repositooriumist parema ülevaate saamiseks veebipõhist keskkonda Bitbucket (bitbucket.org). Bitbucket võimaldab hallata repositooriumiga seotud tarkvara vigu (bug tracking) ning hallata ka rakenduse dokumentatsiooni. Tegemist on keskkonnaga, mille suurem potentsiaal avaldub mitme arendajaga projektide puhul. (Git and Mercurial code management for teams, 2015)

Bitbucket'i serveris olevat repositooriumit saab väga edukalt kasutada ka tarkvara arendusest tootmisesse (*live*) viimiseks. Sama lahendust on kasutanud autor ka käesolevas töös (vt joonis 1).

[2] Täiendavat informatsiooni versioonihalduse kohta leiab järgmisest Vikipeedia artiklist: http://en.wikipedia.org/wiki/Revision_control



Joonis 1 – protsess: tarkvara liigutamine arendusest tootmisesse (live)

3.1.3. Grunt

Grunt [3] on tööriist, mille abil saab automatiseerida erinevaid veebiarendusega seonduvaid protsesse. Automatiseerimise aluseks on konfiguratsioonifail `Gruntfile.js`, milles määratakse ära kõik kasutatavad pistikprogrammid (*plugins*) ning protsessid, millega Grunt tegelema peab. (Bocoup, 2014)

Käesoleva töö raames loodava infosüsteemi loomisel kasutatud Grunt konfiguratsioonifaili on näha lisas nr 18.

Kasutatud pistikprogrammid ja nende ülesanded on järgmised:

- **Concat**
  võimaldab faile (üldiselt *.js) liita kokku üheks. Lisas olevas näites liidetakse kõik ./app/assets/js kataloogis olevad js-lõpulised failid kokku üheks app.js failks, mis paigutatakse seejärel rakenduse avalikku kausta;
- **Uglify**
  vähendab failide mahtu eemaldades üleliigsed tühikud, tabid (*whitespace*);
- **Sass**
  kompileerib Sass koodi tavaliseks CSS'iks. Sass on CSS'i laiendamiseks mõeldud metakeel, millest käesoleva töö autor on kirjutanud lähemalt oma seminaritöös;
- **Watch**

[3] Grunt tööriista veebileht on kättesaadaval järgmisel aadressil: http://gruntjs.com/

watch pistikprogramm võimaldab jääda Grunt'il kuulama failidega (vastavalt konfiguratsioonile) toimuvaid muudatusi;

- **Imagemin**

  vähendab pildifailide mahtu vastavalt konfiguratsioonile

Grunt käivitatakse kasutades samanimelist käsku: `grunt`. Käivitamisel jääb programm esialgu ootama muudatusi vastavalt seadistustele (vt ekraanipilt 4) määratud failides. Olenevalt sellest kas tegu on JavaScript'i või Sass failidega käivitatakse kas "concat" ja "uglify" või "sass" pluginad. Juhul kui arendaja on teinud koodi kirjutades vea, siis antakse sellest programmi väljundis ka märku (vt ekraanipilt 4).

```
⌘ ~/Code/pintra/ (master) grunt
Running "watch" task
Waiting...
>> File "app/assets/css/app.scss" changed.
Running "sass:development" (sass) task

Done, without errors.
Completed in 3.087s at Thu Apr 30 2015 13:00:09 GMT+0300 (EEST) - Waiting...
>> File "app/assets/css/app.scss" changed.
Running "sass:development" (sass) task
Error: Invalid CSS after "...    font-weight": expected ";", was ": bold;"
        on line 56 of ./app/assets/css/app.scss
  Use --trace for backtrace.
Warning: Exited with error code 65 Use --force to continue.

Aborted due to warnings.
Completed in 2.390s at Thu Apr 30 2015 13:00:18 GMT+0300 (EEST) - Waiting...
>> File "app/assets/css/app.scss" changed.
Running "sass:development" (sass) task

Done, without errors.
Completed in 2.231s at Thu Apr 30 2015 13:00:24 GMT+0300 (EEST) - Waiting...
```

Ekraanipilt 4 – programmi Grunt käivitamine ja töövoog

Projekti suurenedes ei ole ilmselt mõistlik igal salvestamisel kõiki neid protsesse läbi teha kuna koodibaasi kasvades läheb JavaScript'i failide minimeerimisele kuluv aeg järjest pikemaks. Sellisel juhul tuleks lahendada olukord nii, et minimeerimine toimub näiteks ainult *live* serveris peale koodi repositooriumist alla laadimist (*git pull*).

3.1.4. Composer

Composer on põhimõtteliselt projektipõhine paketihaldur PHP lisateekide paigaldamiseks, uuendamiseks, kustutamiseks ning nende omavaheliste seoste määramiseks. Projektipõhisus tuleneb sellest, et teeke ei paigaldata süsteemselt vaid kaustapõhiselt. (Boggiano & Adermann, 2015)

Sõltuvuste haldamine (*dependency management*) võimaldab defineerida, millistest teekidest (*libraries*) loodav rakendus sõltub. Samuti lahendub Composer'i kasutamisel ära probleem, mis tekib seoses sellega kui rakenduse tarbeks kasutatav teek sõltub omakorde mingist teisest teegist. (Boggiano & Adermann, 2015)

Kõik sõltuvused pannakse paika failis `composer.json`. Peale sõltuvuste defineerimist failis saab puuduvad teegid paigaldada käsuga `composer install`. Kasutatavate pakettide uuendamiseks on käsk `composer update`, mille tööd ning väljundit näeb ekraanipildil nr 5.

```
⌘ ~/Code/pintra/ (master) composer update
Loading composer repositories with package information
Updating dependencies (including require-dev)
  - Removing intervention/image (2.1.4)
  - Installing intervention/image (2.2.0)
    Downloading: 100%

  - Removing patchwork/utf8 (v1.2.1)
  - Installing patchwork/utf8 (v1.2.2)
    Downloading: 100%

  - Removing phpoffice/phpexcel (1.8.0)
  - Installing phpoffice/phpexcel (1.8.1)
    Downloading: 100%

  - Removing bugsnag/bugsnag (v2.5.3)
  - Installing bugsnag/bugsnag (v2.5.4)
    Downloading: 100%

Writing lock file
Generating autoload files
```

Ekraanipilt 5 – projekti lisapakettide uuendamine

3.1.5. Arendamisel kasutatud tehnoloogiad

Järgnevas peatükis antakse ülevaade kasutatud tehnoloogilistest vahenditest. Autor on kirjeldanud peamiseid süsteemi loomisel kasutatud vahendeid ning iga väiksemat projekti, mis loodud süsteemis mingisugust funktsiooni täidab, ei ole kirjeldatud.

Lühidalt kokku võttes on loodava raamistiku serveripoolsed tehnoloogiad PHP skriptimiskeel ja MySQL andmebaas. Kliendipoolsed, tuntud ka kui *front-end* (veebilehitsejas käivitatavad) tehnoloogiad, taanduvad järgmiseks kolmeks tehnoloogiaks: HTML, CSS, JavaScript.

Tehnoloogiate valiku on teinud autor tuginedes peamiselt enda kogemusele, kuid samuti on peetud silmas, et kõik kasutusel olevad vabavaralised projektid oleks suure kasutajaskonnaga. Suure kasutajaskonna olemasolu vähendab riski, et mõni projekt hüljatakse ning seeläbi jäävad tulemata turvalisust parandavad ning funktsionaalsust laiendavad uuendused.

3.1.5.1. Laravel PHP raamistik

Laravel on avatud lähtekoodil põhinev ning MIT litsentsi alusel levitatav kaasaegseid tarkvaraarenduse printsiipe rakendav veebirakenduste loomiseks mõeldud raamistik.

Käesoleva töö kirjutamise hetkeks on Laravel'ist saanud maailmas üks enim kasutatumaid PHP raamistikke. (Skvorc, 2015)

Model-View-Controller arhitektuur

Laravel põhineb laialdaselt kasutatud (*separated presentation*), täpsemalt *Model-View-Controller* (MVC) arhitektuuril, mille eesmärk on hoida lahus äriloogika ning selle presentatsioon.

MVC disainimustrit kasutatavates rakendustes on programmi kood jagatud järgmiselt:

- **mudel** (*model*) – rakenduse domeenis ehk valdkonnas olev reaalse maailma element/objekt. Mudel on täiesti eraldiseisev üksus, mis tähendab, et sama mudelit kasutades saab esitada andmeid üksteisest sõltumatult ning erineval kujul;
- **vaade** (view) - määrab, kuidas kasutaja päringule vastavad andmed esitatakse
- **kontroller** (*controller*) – programmi osa, mis tegeleb kasutajapoolse sisendiga (valideerib andmed, otsustab mida nende andmetega teha jne)

(Fowler, GUI Architectures, 2006)

Eloquent ORM

Laraveliga kaasa tulev Eloquent ORM on *Active Record* disainimustri implementatsioon, mis lihtsustab süsteemi objektide ning andmebaasi vahelist suhtlust. (Otwell, Eloquent ORM, 2015)

Active Record tähendab, et arendaja saab kasutada andmebaasi kirjeid nagu tavalisi süsteemi objekte. Iga andmebaasi tabeli jaoks peab olema mudel (*model*), mille kaudu kirjutada ning lugeda objektide infot andmebaasist. (Fowler, Active Record, 2003)

```
1   // define the user model
2   class User extends Model {
3       protected $table = 'users';
4   }
5
6   // Use Eloquent ORM to create our first user
7   $user = new User;
8   $user->name = 'Robert Test';
9   $user->email = 'user@domain.tld';
10  $user->save();
11
12  // Change the first user's e-mail address and save changes
13  $user = User::find(1);
14  $user->email = 'user_changed_email@domain.tld';
15  $user->save();
```

Koodinäide 1 – Eloquent ORM

Eloquent ORM[4] on oma funktsionaalsuse poolest väga suutlik ning ei piirdu kaugeltki ainult paari meetodiga objekti küsimiseks andmebaasist ning objekti loomiseks või uuendamiseks andmebaasis. Eloquent võimaldab defineerida mudelite omavahelisi seoseid. (Otwell, Query Builder, 2015)

Vaatamata mugavusele ning selgele koodile, mida võimaldab Eloquent ORM, tuleks keerukamate päringute puhul küsida andmeid kasutades kas Laravel'iga kaasa tulnud Query Builder'it[5] või kirjutada puhast SQL'i, mis suhtleb andmebaasiga kõige madalamal tasemel, andes seeläbi ka kõige kiiremini vastused päringutele.

Autentimine

Valdav osa tänapäevastest infosüsteemidest on kasutajapõhised, see tähendab, et süsteem võimaldab lisada uusi kasutajaid ning lubab autentida juba olemasolevaid.

Laravel raamistik teeb autentimisega seotud toimingute programmeerimise arendaja jaoks väga lihtsaks. Toimingud nagu parooli meeldetuletuste saatmine, sisselogimine ja kasutaja andmete meeldejätmine on raamistiku sees ära lahendatud. See tähendab, et nimetatud toimingute tegemiseks tuleb kirjutada minimaalsel hulgal koodi (vt koodinäide 2).

Puhverdamine (*caching*)

Serveri koormuse vähendamiseks tuleb Laravel'iga kaasa unifitseeritud rakendusliides (API), mis andmete puhverdamiseks (*caching*) kasutab *key-value store* lahendusi nagu Memcached[6] ja Redis[7].

Artisan – käsurea tööriist

Laravel'iga tuleb kaasa käsurea tööriist Artisan, mis võimaldab arendajal kasutada suurt hulka käske, mis lihtsustavad ning kiirendavad rakenduse arendamist. (Otwell, Artisan CLI, 2015)

---

[4] Täpsem kirjeldus Eloquent ORM'i võimaluste kohta: http://laravel.com/docs/5.0/eloquent
[5] Lisainfo Query Builder'i kasutamise kohta asub järgmisel aadressil: http://laravel.com/docs/5.0/queries
[6] http://memcached.org/
[7] http://redis.io/

3.1.5.2. MySQL

Maailmas üks enim levinud avatud lähtekoodil põhinevaid relatsioonilisi andmebaasimootoreid, mida arendab Oracle Corporation. Relatsiooniline tähendab seda, et andmebaasis hoitav andmestik on jaotatud loogilisel tasandil tabelitesse. Andmebaas ise aga arvuti kettal erinevatesse failidesse. Andmebaasimootor on väga paindlik ning selle kasutamiseks piisab ka tagasihoidlike parameetritega koduarvutist. Suuremate süsteemide puhul suudab MySQL opereerida ka paigutatuna klastritesse (Oracle Corporation, 2015)

3.1.5.3. Bootstrap

Üks populaarsemaid *front-end* raamistikke, mille esimene versioon tuli välja aastal 2001. Bootstrap võimaldab luua väga lihtsalt täisväärtuslikke kasutajaliideseid, mis on ilma suurema peavaluta kasutatavad lisaks arvuti ekraanidele ka mobiilsetel seadmetel. (Gerchev, 2013)

Bootstrap koosneb kasutajaliidese komponentidest, mille kasutamine lihtsustab ja kiirendab veebirakenduste arendust kuna puudub otsene vajadus muuta CSS-koodi. Standardsed komponendid on kirjutatud kasutades HTML, CSS ja JavaScript tehnoloogiaid.

Mõned peamised komponendid, mis nimetatud raamistikuga kaasa tulevad[8]:

- ikoonid;
- nupud ja rippmenüüd;
- vormide väljad;
- modaalaknad (*modal window*);
- erinevad menüüribad

[8] Täielik nimekiri komponentidest on kättesaadav projekti ametlikult kodulehelt: http://getbootstrap.com/

3.1.5.4. Angular

AngularJS on rikkalike kliendipoolsete rakenduste loomiseks mõeldud raamistik, mille esimene versioon jõudis avalikkuse ette aastal 2012. Rakenduse loojaks oli Google'i töötaja Miško Hevery, kes alustas projekti kallal tööd juba aastal 2009. Praeguseks toetab Google ametlikult AngularJS'i arendust. (Refsnes Data, 2015)

3.2. Andmebaasi struktuur

Loodud rakenduse andmebaasi lõplik struktuur on kujunenud välja implementatsiooni käigus. Rakenduse disainietapis ei olnud eesmärgiks saavutada kohe võimalikult detailne mudel kuna üldjuhul selline asi lihtsalt ei ole võimalik eelkõige kliendi vajaduste muutumise tõttu.

Järgnevalt on toodud ära rakenduse tööks vajalikud tabelid ning nendes paiknevate andmete kirjeldused. Detailne andmemudeli struktuur koos andmeväljade ja tabelite vaheliste seostega on kirjeldatud töö lisas nr 22.

application_attachment_files – avalduste juurde lisatavate dokumentide failid
application_attachments – avalduste juurde lisatavate dokumentide metaandmed
application_comments – avalduse menetlusega seotud märkmed
application_documents – avaldusega seotud dokumentide (PDF ja BDOC) metaandmed
application_document_files – avaldustega seotud dokumentide failid
application_log – avaldusega seotud toimingute logikirjed
application_props – avalduste dünaamilised infoväljad
application_signatures – digitaalselt allkirjastatud avalduste allkirjade metainfo
application_statuses – avalduste süsteemsed staatused ja nende kirjeldused
application_type_notifications – info automaatsete teavituste kohta kooli töötajatele
application_type_requiredattachments – avalduste tüüpide juures nõutud lisadokumendid
application_type_requiredattachments_application_attachments – seosed tabelite `application_type_requiredattachments` ja `application_attachments` vahel

application_types – avalduste tüübid ja nende kirjeldused

application_views – avalduste vaatamiste logi (kooli personali poolt vaadatud avaldused)

applications – esitatud avaldused

assigned_roles – kasutajatele määratud rollid (seosed tabelite `users` ja `roles` vahel)

ehak_codes – EHAK klassifikaatorid

failed_jobs – Laravel'i järjekorras olevate tööde (*jobs*) käivitamise käigus tekkinud vead

grade_user – kasutajate kuuluvus klassi

grades – klassid

group_types – gruppide tüübid (pikk päev, huviring, klass)

group_user – kasutajate kuuluvus gruppidesse (seosed tabelite `groups` ja `users` vahel)

groups – grupid

hobbygroup_categories – huviringide kategooriad

hobbygroups – huviringid

loginlog – sisselogimiste logikirjed

maillog – süsteemist saadetud e-mailide logid

notices – süsteemi töölaual kuvatavad teated

password_reminders – paroolide lähtestamiseks vajaminevad räsikoodid (*hash*)

permission_role – rollile määratud õigused (seosed tabelite `permissions` ja `roles` vahel)

permissions – õigused

postcodes – info Eesti postikoodide kohta

roles – rollid

user_props – kasutajate dünaamilised infoväljad

user_user – lapsevanema ja lapse vahelised seosed

users – kõik süsteemis olevad kasutajad (lapsevanemad, lapsed, kooli töötajad)

Üldiselt ei ole rakenduse struktuuris arenduse jooksul olnud vaja teha suuremaid muudatusi. Funktsionaalsuse täienedes lisandus küll tabeleid juurde, kuid olemasolevaid üldjuhul muuta ei olnud vaja.

Ainuke tehtud muudatus oli seotud dokumentide hoidmisega andmebaasis. Kuna avaldustega seotud dokumendid on ladustatud andmebaasis, siis esialgses andmemudelis oli dokumentide tarbeks kasutusel ainult üks tabel, mis sisaldas nii faile (BLOB kujul) kui ka nende failidega seotud metaandmeid.

Seni kuni lisatavad failid olid väiksed ning tabelis kirjeid vähe, ei olnud probleemi. Peale mõnekümne kooli sisseastumise taotluse esitamist, mille külge olid kasutajad lisanud ka dokumente, muutus baasis failide tabeli vaatamine väga aeglaseks. Kuigi soov oli vaadata failide metaandmeid, laeti ka BLOB väljade sisu. Selle probleemi lahendamiseks sai paigutatud failid eraldi tabelisse `application_attachment_files`. Nimetatud tabelis hoitakse ainult reaalseid andmefaile ning kogu metainfo (suurus, mime tüüp, failinimi jne) sai paigutatud tabelisse `application_attachments`.

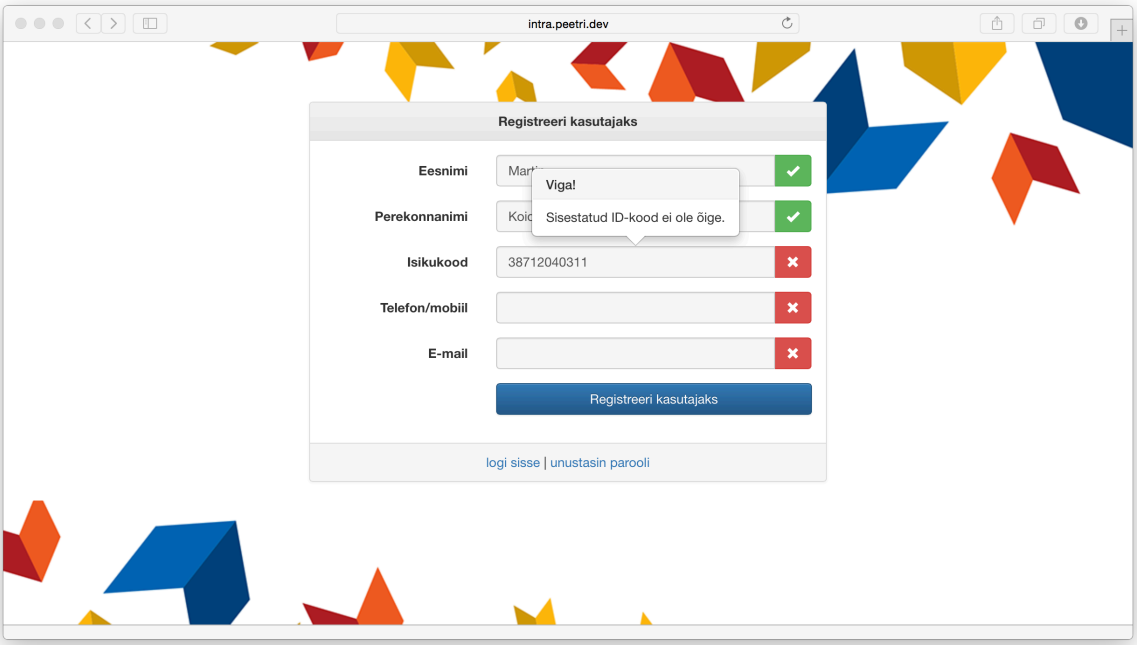
3.3. Funktsionaalsed lahendused

3.3.1. Kasutajakontod

Kasutajate registreerimine

Kasutajakonto registreerimislehel on taotluslikult küsitud võimalikult vähe infot peamise eesmärgiga vähendada vormi täitmiseks kuluvat aega, mis seeläbi peaks parandama kasutajamugavust.

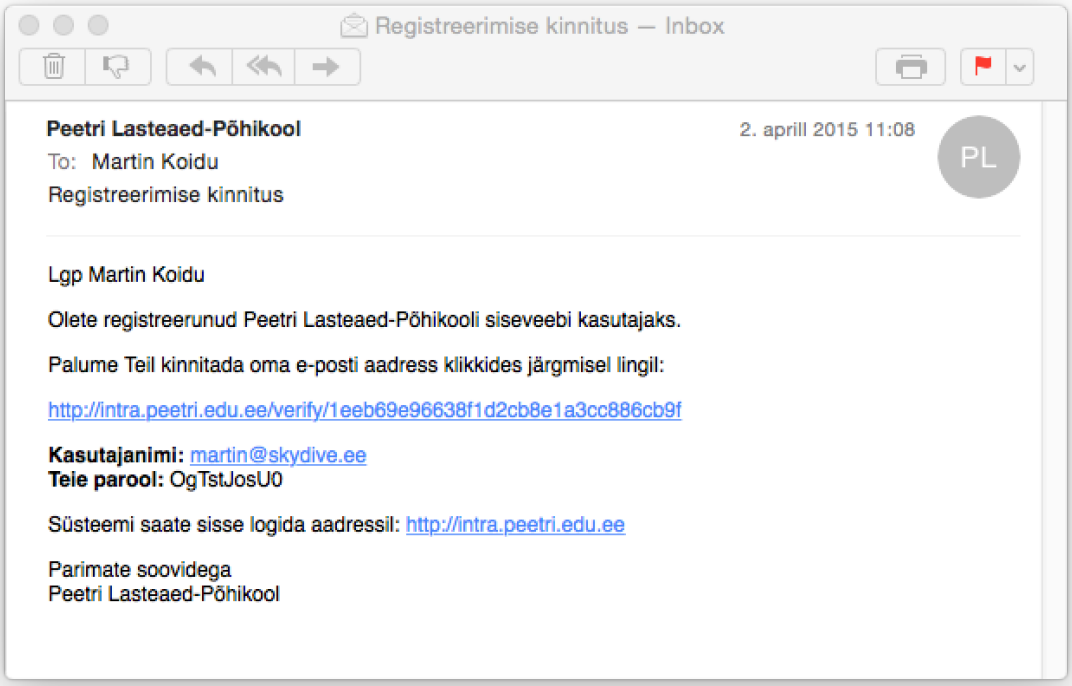
Sisestatud andmed valideeritakse jooksvalt vormi täitmisel kasutades JavaScript'i. Ainukese erandina on isikukood, mille kontrollimine toimub serveripoolel. Kui kasutaja sisestatud väljas on 11 sümbolit, mis on teatavasti Eestis kehtiva isikukoodi pikkuseks, siis saadetakse serverile päring sisestatud koodi valideerimiseks. Esmalt teostatakse koodi valiidsuse kontroll, misjärel kontrollitakse ega sellise isikukoodiga kasutajat juba süsteemis olemas ole. Juhul kui sisestatud number ei ole korrektne või juba eksisteerib, kuvatakse vastav veateade (vt ekraanipilt 6).



Ekraanipilt 6 – Kasutajaks registreerimine

Registreerimisel genereeritakse kasutajale juhuslik 10-tähemärgiline parool (sisaldab nii suuri, väikeseid tähti kui ka numbreid), mis saadetakse koos registreerumise kinnituse ja e-posti aadressi kinnitamise lingiga (vt ekraanipilt 7). E-posti aadressi kinnitamise link aegub 45 minuti jooksul, misjärel tuleb kasutajal saata endale käsitsi uus kinnitusmail.

Automaatse parooli genereerimise kasuks on otsustanud autor turvakaalutlustel. Enamik kasutajaid kasutab üldjuhul küllaltki lihtsaid ja liiga lühikesi paroole. Samuti kasutavad ligikaudu pooled interneti kasutajatest ühte ja sama parooli erinevatel veebilehekülgedel, mis sisuliselt tähendab seda, et kui ühe keskkonna paroolid pääsevad kurjade kavatsustega kasutajate kätte, siis saavad nad sama parooli kasutades sisse ka teistesse selle kasutaja veebikeskkondadesse. (Riley, 2006)



Ekraanipilt 7 – registreerumise kinnitus

E-mailide saatmisel kasutatakse Laravel raamistiku poolt kaasa tulevat `Mail` klassi, millele on autor kirjutanud omalt poolt mähisklassi (*wrapper class*), lihtsustades rakendusest mailide saatmist ja saadetavate mailide üle logi pidamist. Laravel'i `Mail` klass kasutab omakorda Swift Mailer[9] teeki.

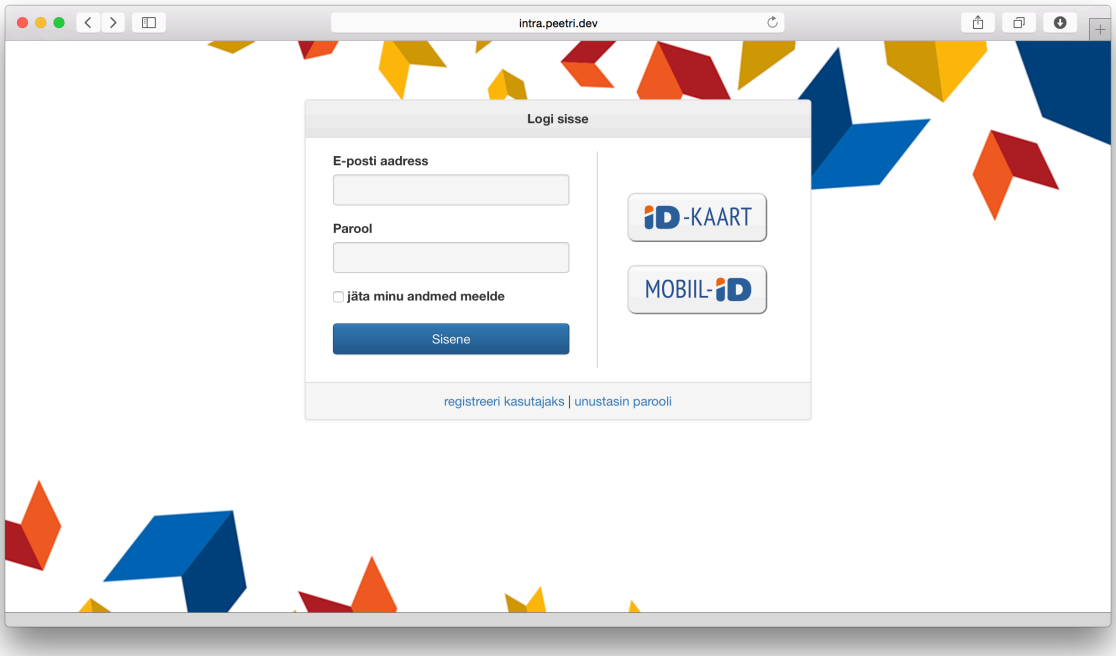
Autentimine

Loodud infosüsteem kasutab kasutajakontodega seotud funktsionaalsuse saavutamiseks standardset Laravel raamistikuga kaasa tulevat `Auth` klassi. `Auth` klass sisaldab meetodeid kasutaja sisse ja välja logimiseks ning autenditud kasutaja kohta info pärimiseks.

Kasutaja süsteemis autentimisel (vt ekraanipilt 8) kontrollitakse esimese asjana, kas sisestatud andmed on täidetud. Väljade täidetuse kontroll on realiseeritud esimese astmena kliendipoolselt kasutades JavaScript'i. Seega ei tohiks juhtuda, et autentimisel

[9] Swift Mailer teegi ametlik lehekülg on saadaval järgmisel aadressil: http://swiftmailer.org/

jõuab rakendus `AuthController` klassis oleva `postLogin()` meetodini (vt koodinäide 2), kuna tühjade väljade korral vorm ei edastata andmeid serveripoolsele skriptile.



Ekraanipilt 8 – Sisselogimise vaade

Eeldusel, et e-maili ja parooli väli on täidetud, üritab meetod `Auth::attempt()` sisendandmeid kasutades leida kasutajate seast vastet. Vaste leidmisel logitakse kasutaja sisse ning väärtusena tagastatakse `true`.

Kasutaja sisselogimisel käivitatakse funktsioon `updateLoginData()`, mis uuendab kasutajate tabelis sisse logimiste arvu ning uuendab ka vastavat ajamärgist. Seejärel lisatakse autentimise õnnestumise kohta kirje logiraamatusse.

Järgmisena kontrollitakse, kas kasutajal on süsteemi sisestatud mõni laps. Juhul kui ei ole, siis suunatakse ta esimese asjana uue lapse lisamise lehele. Muul juhul suunatakse kasutaja kas esilehele (*dashboard*) või enne sisse logimise toimingut päritud aadressile. Juhul kui kasutaja sisestatud andmetele ei vastanud `users` tabelis ühtegi rida, siis lisatakse ebaõnnestunud katse kohta logisse kirje. Kirje lisamise järel kontrollitakse kas kasutaja on sisestanud järjest vale parooli rohkem kui 5 korda. Kui on, siis blokeeritakse

konto ning kasutaja suunatakse lehele tagasi koos veateatega konto ajutisest sulgemisest.

```
1   /**
2    * Handles request from user login form
3    * @return mixed
4    */
5   public function postLogin()
6   {
7       $input = Input::all();
8
9       // In case input is not filled
10      $validator = Validator::make($input, User::$loginRules);
11      if ($validator->fails())
12      {
13          return Redirect::to('/login')->withErrors($validator);
14      }
15
16      $credentials = [
17          'email'     => $input['login'],
18          'password'  => $input['password']
19      ];
20
21      // Set the remember me cookie if user has checked the box
22      $rememberMeChecked = (Input::has('remember')) ? true : false;
23      if (Auth::attempt($credentials, $rememberMeChecked))
24      {
25          Auth::user()->updateLoginData();
26          LoginlogEntry::success($input['login']);
27
28          if (Auth::user()->needsToEnterChildren())
29          {
30              return Redirect::intended('/children/add');
31          }
32
33          // Redirect to the intended page or to /dashboard
34          return Redirect::intended('/dashboard');
35      }
36
37      LoginlogEntry::fail($input['login']);
38
39      // Check if user has tried to log in too many times
40      if(Auth::tooManyAttempts())
41      {
42          return Redirect::to('/login')->with('flash_error', Lang::get('auth.blocked'));
43      }
44
45      // Auth failed! let's go back to the login page
46      return Redirect::to('/login')->with('flash_error', Lang::get('auth.incorrect'))->withInput();
47  }
```

Koodinäide 2 – postLogin() – kasutajanime ja parooliga sisselogimise funktsioon

ID-kaardiga sisse logimisel kontrollitakse serveripoolse muutuja `REDIRECT_SSL_CLIENT_S_DN_CN` väärtuses sisalduvat ID-koodi. Juhul kui see kood eksisteerib andmebaasis, loetakse kasutaja autentimine õnnestunuks ning kasutaja logitakse sisse.

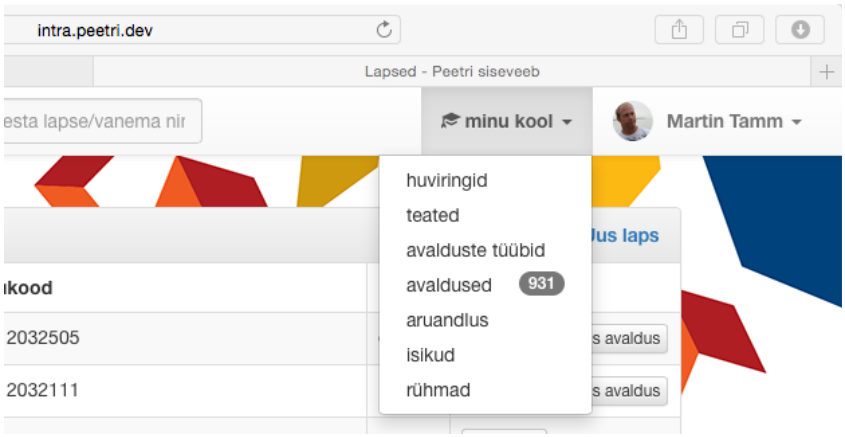
Mobiil-ID on võrreldes ID-kaardiga natukene keerulisem. Esiteks on vaja Mobiil-ID kasutamiseks sõlmida ettevõttega AS Sertifitseerimiskeskus vastava teenuse kasutamiseks leping. Peale lepingu sõlmimist jääb üle ainult vastav funktsionaalsus oma rakenduses tööle panna. Arendajate töö lihtsustamiseks on ettevõte teinud vastava näidisrakenduse ja põhjaliku dokumentatsiooni.

Kasutajate rollid ja õigused

Kasutajate ja rollide funktsionaalsus on implementeeritud kasutades Laravel'i lisapaketti nimega Entrust[10].

Nimetatud pakett võimaldab defineerida rollid ja õigused, seostada õigused rollidega ning rollid omakorda ühe konkreetse kasutajaga. Samuti on võimalik lisaks rollile anda kasutajale juurde ka õiguseid eraldiseisvalt, mis tähendab, et põhimõtteliselt võib anda ainult lapsevanema rollis olevale kasutajale õiguseid hallata esitatud huviringide taotlusi.

Juhul kui kasutajale on lisatud juurde mõni kooli töötajatele omistatavatest rollidest (huvijuht, registripidaja, majandusjuhataja), siis lisandub menüüribale valik "minu kool" (vt ekraanipilt 9)



Ekraanipilt 9 – menüü lisavalik "minu kool"

[10] Entrust projekti ametlik lehekülg on kättesaadav järgmiselt aadressilt: https://github.com/Zizaco/entrust

Eelneval ekraanipildil on näha administraatorina sisseloginud kasutaja valikud. Teistel rollidel on menüüpunkte mõnevõrra vähem.

Käesoleva töö kirjutamise hetkel ei ole rollide lisamine ja muutmine tehtav kasutajaliidesest ning vastavad seosed tuleb käsitsi luua andmebaasis. Rollide ja õiguste haldamise funktsionaalsus implementeeritakse edasiste arenduste käigus.

Kasutajakaart

Projekti alguses sai üksikute vaadete jaoks kasutades veebipõhist tööriista Moqups[11] loodud mõned prototüübid.

Kasutajakaardi prototüüp on näha lisas nr 20 Vaate teostus näeb välja täpselt selline nagu prototüübil. Küll aga on toimunud vahepeal muudatus (nähtav kõikidelt käesolevas töös olevatelt ekraanipiltidelt) üldises süsteemi menüüpaigutuses, mis esialgu asus tulbana vasakul (vt lisa 20).

Hiljem sai sellisel kujul prototüüpimisest loobutud kuna samasuguse vaate loomine Bootstrap raamistikku kasutades ei olnud märgatavalt aeganõudvam võrreldes Moqups'i peal tehtuga. Viimasel juhul tuli vaade ikkagi lõpuks HTML'is realiseerida nii, et kokkuvõtteks läks aega ehk rohkemgi kui kohesel koodi kirjutamisel.

Kasutajate otsing

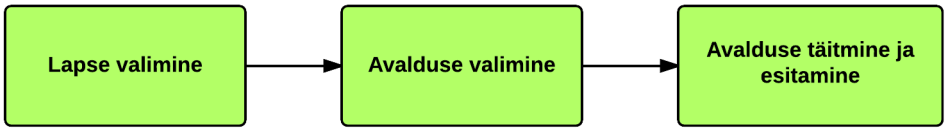
Kasutajate otsingule pääsevad ligi ainult kooli personali hulka kuuluvad kasutajad, kes leiavad vastava sisestuskasti menüüriba keskelt (vt lisa 12).

Andmete sisestamisel ja "enter" klahvi vajutamisel tehakse AJAX päring serveripoolsele skriptile, mis tulemuse korral tagastab HTML-kujul tabeli. Vastuseks saadav HTML paigutatakse Bootstrap raamistiku modaalaknasse (*modal window*) ning seejärel kuvatakse kasutajale (vt lisa 14). Juhul kui andmeid ei leitud, siis akent ei näidata.

[11] Moqups tarkvara ametlik veebileht asub järgmisel aadressil: https://moqups.com/

3.3.2. Avaldused

Kõikide avalduste esitamise protsess koosneb üldjuhul kolmest etapist. Huviringi taotluse puhul on peale avalduse (tüübi) valimist veel ka huviringi enda valik.



Joonis 2 – avalduse esitamise protsess

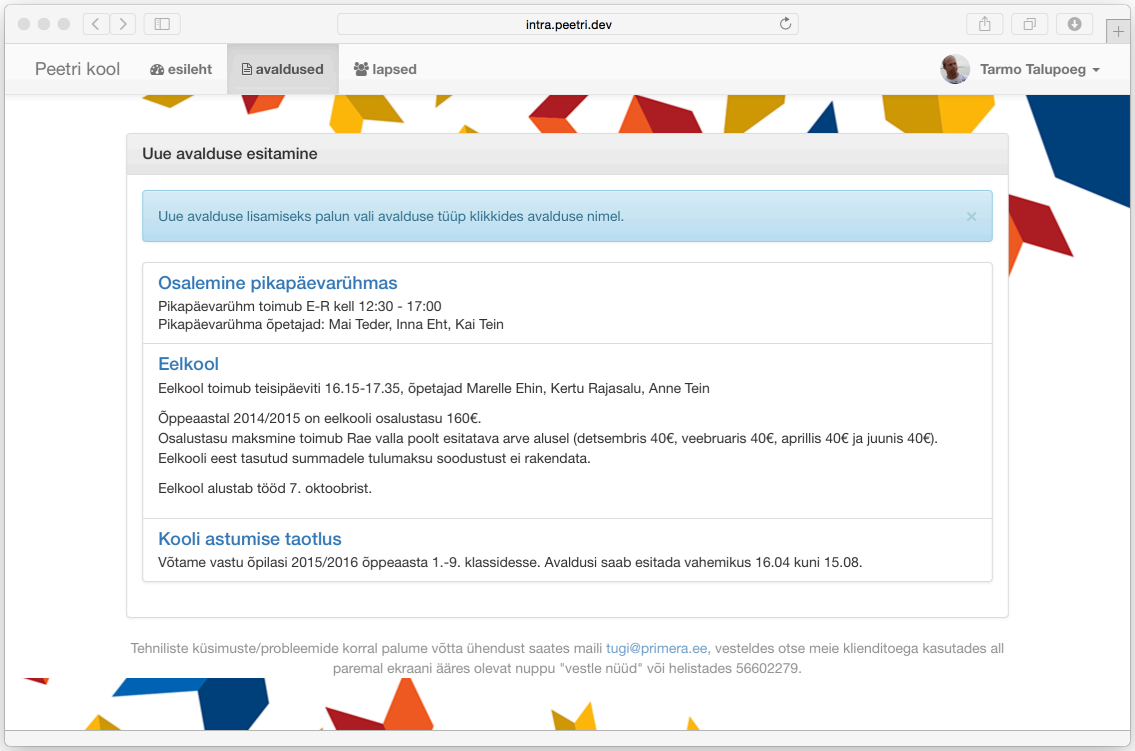
Lapse valimine

Lapse valimise etapp kehtib ainult kasutajatele, kellel on süsteemis lisatud rohkem kui üks laps. Ühe lapse puhul jääb esimene etapp vahele ning kasutajale kuvatakse koheselt loetelu avalduste tüüpidest (vt ekraanipilt 10).

Infosüsteemi arenduse varajases faasis toimus lapse info sisestamine otse avalduselt. Kuna uute avalduse tüüpide lisandumisel tulid juurde taotlused, mis olid mõeldud ainult kindlas klassiastmes õppivatele lastele, siis otsustas autor teha lapse valikust eraldi etapi, mis võimaldab avalduse valimisel kuvada nimekirjas ainult valitud õppuri nimel esitatavad taotlused.

Avalduse/taotluse valimine

Avalduste valiku kuvamisel (vt ekraanipilt 10) arvestatakse valitud lapse klassi. See tähendab, et kui on valitud laps, kes käib näiteks kooli 5. klassis, siis ei kuvata nimekirjas kooli astumise taotlust.



Ekraanipilt 10 – avalduse (tüübi) valimine

Avalduse täitmine ja esitamine

Avalduse vormis on info jaotatud järgmistesse loogilistesse plokkidesse:

- **lapse/õpilase andmed**

  üldjuhul tulevad süsteemist eeltäidetult;
- **avalduse esitaja/lapsevanema andmed**

  tulevad süsteemist eeltäidetult ning üldiselt on muudetavad;
- **avalduse spetsiifikast tulenevad andmed**

  kõik on kasutaja poolt täidetavad väljad;

Kooli astumise taotluse puhul on täiendavalt lisatud veel kaks plokki, et info oleks arusaadavamalt esitatud (vt lisa 8).

Töö lisas nr 24 on näha detailne protsess sellest, kuidas käib läbi loodud rakenduse kooli sisse astumise taotluse esitamine.

Kuna väljad on olenevalt avalduse tüübist erinevad, siis tuli välja mõelda lahendus, kuidas talletada seda infot andmebaasis. Kõige primitiivsem lahendus oleks olnud teha eraldi tabelid erinevate avalduse tüüpide jaoks. Sellisel juhul oleks aga tulnud tulevikus uute avalduse tüüpide lisandudes hakata andmebaasi uusi tabeleid juurde tegema. Kui peaks juhtuma, et mõnda avaldusel olevat välja pole enam üldse vaja või kui avaldusele tahetakse lisada juurde uusi andmevälju, siis tuleks hakata vastava tabeli väljades muudatusi tegema.

Teine lahendus oleks olnud tekitada `applications` tabelisse üks väli, millel hoida infot *serialized* kujul.

Serialized kuju tähendab sisuliselt mingisuguse andmestruktuuri teisendamist salvestatavale kujule (vt koodinäide 3) ilma struktuuri ja andmetüüpi kaotamata. (The PHP Group, 2015)

```
1    $storableData = [
2       'application_field_1' => 'Application field 1 value',
3       'application_field_2' => 'Application field 1 value'
4    ];
5
6    $serializedData = serialize($storableData);
7
```

Koodinäide 3 – andmestruktuuri teisendamine salvestatavale kujule

Eelneval joonisel oleva koodi käivitamisel saab muutuja `$serializedData` väärtuseks järgmine tähemärkide jada:

```
a:2:{s:19:"application_field_1";s:25:"Application field 1
value";s:19:"application_field_2";s:25:"Application field 2 value";}
```

See tähemärkide jada salvestataks ühele andmebaasi väljale. Hiljem andmebaasist selle väärtuse küsimisel saab taastada `unserialize()` funktsiooni kasutades esialgset andmestruktuuri, milleks on koodinäites 3 olev `$storableData` massiiv (*array*).

Serialized kuju kasutamise plussiks on kindlasti see, et kuna kogu andmestik on paigutatud ühele tabeli reale, siis väljade muutumisel on vaja käivitada ainult üks päring andmebaasi, mis praktikas tähendaks võitu süsteemi toimimise kiiruses. Miinuseks võib pidada andmete inimsilmale küllaltki loetamatut kuju. Lisaks sellele on oht muuta andmestik loetamatuks, kui peaks olema tarvis käsitsi baasis muuta mõne avalduse spetsiifilise andmevälja väärtust.

Loodava infosüsteemi puhul otsustas autor kasutada kolmandat varianti, mis on kombinatsioon kahest eelnevalt kirjeldatud viisist sama probleemi lahendamiseks.

Dünaamiliste väljade jaoks on tehtud andmebaasi eraldi tabel, mille nimi moodustub põhiobjekti nimest ning sufiksist "_props". Avalduste puhul on põhiobjekti nimi "application" seega selle objekti dünaamiliselt lisatavad väljad ja nende väärtused on paigutatud tabelisse `application_props` (vt lisa 22).

Sellisel viisil on andmed salvestatud baasi ka inimese jaoks kergesti loetaval kujul ning juhul kui peaks olema tarvis muuta andmeid otse baasist, siis võimalus, et käsitsi muutmisel hiljem mingi viga tekib, on väike. Lahenduse miinuseks on aga see, et andmebaasi koormus on mõnevõrra suurem kuna andmete muutmiseks tuleb teha rohkem päringuid.

Programmi koodis on lahendatud kirjeldatu selliselt, et kui andmeobjektile (*data object*) omaduste (*properties*) kirjeldamisel kasutatakse prefiksiga "prop_" andmevälja, siis andmete salvestamisel teab süsteem, et need väljad tuleb paigutada eraldi vastavasse tabelisse.

```
1  $application = new Application();
2  $application->start_date = '2015-04-10';
3  $application->user_id = 1212;
4  $application->subject_user_id = 1214;
5  $application->prop_payer_name = 'Tarmo Tamm';
6  $application->prop_payer_email = 'tarmo@tamm.ee';
7  $application->save();
```

Koodinäide 4 – andmeobjekti salvestamine

Eelnevas näites salvestatakse väljad `start_date`, `user_id`, `subject_user_id` tabelisse `applications` (vt lisa 22). Väljad `prop_payer_name` ja `prop_payer_email` salvestatakse tabelisse `application_props`, kusjuures eesliide "prop_" salvestamisele ei kuulu (vt joonis 3).

```
-------------------------------------------------------
| application_id | name               | value         |
-------------------------------------------------------
| 1216           | payer_name         | Tarmo Tamm    |
-------------------------------------------------------
| 1216           | payer_email        | tarmo@tamm.ee |
-------------------------------------------------------
```

Joonis 3 – avalduste dünaamilised andmeväljad salvestatuna baasi

Selleks, et peamisi andmeobjektidega seotud meetodite funktsionaalsust laiendada "_prop" tabelite kasutamiseks, on autor loodud rakenduses laiendanud klassi `Eloquent` uue klassiga `Model`. Loodud klassis on üle kirjutatud peamiselt andmeobjektide salvestamiseks ja baasist küsimiseks pruugitavad meetodid. Andmeobjektide salvestamiseks koos lisandväljadega pruugitav meetod `save()` on näha käesoleva töö lisas nr 19.

Esitatud avaldused

Esitatud avaldusi näevad ja kinnitavad ainult kooli töötajad. Esitatud avalduste vaadet näeb lisas nr 12.

Esitatud avalduste nimekirja realiseerimisel on kasutatud AngularJS'i, ning täiendavaid mooduleid AngularStrap[12] ja ngTable[13]. ngTable võimaldab luua kiirelt ja mugavalt kasutajapoolselt (browser) kohandatavaid tabeleid. Näiteks on ngTable moodulit kasutades võimalik ilma lehekülge laadimata peita/kuvada tabeli veerge. Samuti on lahendatud nimetatud mooduliga lehekülgedele jaotamine.

[12] http://mgcrea.github.io/angular-strap/
[13] http://bazalt-cms.com/ng-table/

Esitatud avaldustega toimetamiseks tuleb märgistada üks või mitu avaldust, misjärel ilmub lehekülje alla toimingute valikuriba. Olenevalt parajasti valitud avalduste tüübist kuvatakse ribal erinevad toimingud.

3.3.3. Huviringid

Selleks, et oleks võimalus üldse huviringi avaldusi esitada peab olema huviringiga seotud vähemalt üks grupp. Seega on huviringide funktsionaalsus on väga tihedalt seotud gruppidega.

Üldistatult asjale lähenedes võiks vaadelda huviringe kui mõttelisi gruppe, mis koondavad enda alla reaalsed inimeste kogumid (tegelikud süsteemis olevad grupid).

Huviringi andmeobjektis (vt lisa 22) sisaldub järgmine info, mis on oluline laste gruppidesse jaotamise automatiseerimiseks:

- `requirements` – klassiastmed;
- `automatic_approval` – kas ring on automaatse aktsepteerimisega või mitte;
- `open_for_all` – kas huviring on piiratud arvu osalejatega või mitte;
- `disabled` – huviringi olek (aktiivne/mitteaktiivne)

Lisaks on tähtis väli huviringi alguskuupäev (`date_start`), mis on vajalik sööjate aruande jaoks.

2014/2015 õppeaastal on manuaalse aktsepteerimisega ringe 4-5, mis tähendab, et kõik ülejäänud lapsed jaotab süsteem automaatselt vastavatesse ringidesse.

Käesoleva töö kirjutamise hetkel puudub süsteemis võimalus dünaamiliselt hallata huviringide andmeobjekte. Haldamise funktsionaalsuse lisamine on võetud plaani järgmiste arenduste käigus.

3.3.4. Grupid

Gruppide funktsionaalsuse teostamisel on autor püüdnud luua võimalikult universaalse lahenduse, mis suudaks hakkama saada kõikvõimalike variatsioonidega gruppide olemuses. Teisisõnu, lahendus on püütud teha võimalikult universaalseks nii, et ühes tabelis `groups` (vt lisa 22) oleks võimalik hoida nii gruppe, mis oma olemuselt on lihtsalt isikute kogum kui ka gruppe, mis on lisaks seotud kooli protsessidest tuleneva loogikaga. Viimaste hulka kuuluvad käesoleva töö skoobis olevad huviringide grupid.

Gruppide andmeobjekt on "group" (grupp), mille ladustamisel kasutatakse samanimelist tabelit andmebaasis (vt lisa 22)

Lisaks muudele infole, omavad huviringide gruppidesse jaotamisel väga olulist tähtsust järgmised andmeväljad:

- `ref_id` – vastava huviringi identifikaator;
- `size` – liikmete arv grupis;
- `requirements` – klassiastmed, milles õppivad lapsed võivad gruppi kuuluda;

Lisaks kasutatakse `weekdays` välja sööjate aruande koostamisel.

3.3.5. Aruandlus

Aruanded on realiseeritud kasutades projekti Laravel Excel[14], mis kasutab omakorda PHPExcel teeki. Laravel Excel võimaldab kasutada Laravel raamistikuga kaasa tulevat vaadete (*views*) lahendust, mis teeb xlsx failide kujundamise ja väljastamise väga lihtsaks.

Raportite väljastamisel käivitatakse klassis `ExportController` paiknev meetod `getApplications()` (vt koodinäide 5). Parameetritena antakse kaasa avalduse tüüp, millise staatusega avaldusi raportisse soovitakse ja viimasena formaat (csv või xlsx).

[14] https://github.com/Maatwebsite/Laravel-Excel

Seejärel luuakse Exceli instants ning uus tööleht (*sheet*), mille sees võetakse raportisse minevad avaldused, mis antakse edasi vaatele (*view*).

```
1   /**
2    * Generate applications report
3    * @param $applicationTypeTag
4    * @param $applicationStatus
5    * @param $format
6    */
7    public function getApplications($applicationTypeTag, $applicationStatus, $format)
8    {
9        $applicationType = ApplicationType::getByTag($applicationTypeTag);
10       $reportFilename = Helper::stringToURL($applicationType->name).'-eksport-'.date('dmY');
11
12       // Create excel output
13       Excel::create($reportFilename,
14           function($excel) use ($applicationType, $applicationStatus)
15       {
16           // Add sheet
17           $excel->sheet($applicationType->name,
18               function($sheet) use ($applicationType, $applicationStatus)
19           {
20               // Get report data
21               $applications = $applicationType->getApplicationsByStatus($applicationStatus);
22
23               // Generate view
24               $sheet->loadView('export.'.$applicationType->tag, ['applications' => $applications]);
25           });
26       })->export($format);
27   }
```

Koodinäide 5 – Esitatud avalduste raporti väljastamise meetod

Vaate fail (vt koodinäide 6) on kirjutatud kasutades tavapärast HTML markeeringut. Andmestik paigutatakse HTML elemendi `<table>` sisse. Lahtrite sisu ning tabelit ennast saab kujundada kasutades HTML4 standardi aegadel kasutusel olnud kujunduse määramise atribuute (näit. `align`, `valign`, `width`) või CSS'i.

Kuna vaate faili sisu ei kuvata kusagil veebilehitsejas (*browser*), vaid sõelutakse (*parse*) eelnevalt Laravel Excel'i poolt läbi, siis ei pea üldiselt keskenduma reeglitele vastava HTML kirjutamisele.

```
1 <html>
2     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
3     {{ HTML::style('/assets/css/excel.css') }}
4     <table>
5         <thead>
6             <tr>
7                 <th>Lapse nimi</th>
8                 <th>Isikukood</th>
9                 <th>Lasteaed</th>
10                <th>Kodune keel</th>
11                <th>Lapsevanema (maksja) nimi</th>
12                <th>E-post</th>
13                <th>Telefon</th>
14                <th>maksevabastus</th>
15            </tr>
16        </thead>
17        @foreach ($applications as $application)
18        <tr>
19            <td align="left">{{ $application->subjectUser->getName() }}</td>
20            <td align="left">{{ $application->subjectUser->idcode }}</td>
21            <td align="left">{{ $application->subjectUser->prop_kindergarden }}</td>
22            <td align="left">{{ $application->subjectUser->prop_home_lang }}</td>
23            <td align="left">{{ $application->prop_payer_name }}</td>
24            <td align="left">{{ $application->prop_payer_email }}</td>
25            <td align="left">{{ $application->prop_payer_phone }}</td>
26            <td align="left">@if($application->prop_preschool_no_payment) jah @endif</td>
27        </tr>
28        @endforeach
29    </table>
30 </html>
```

Koodinäide 6 – raporti vaate (view) fail

Käesoleva töö lisas nr 16 on näha sööjate nädala väljavõtet. Nimetatud raporti väljastamise meetod on sarnane koodinäites 5 olevaga. Peamine erinevus seisneb raportisse andmete võtmise loogikas.

3.4. Mittefunktsionaalsed lahendused

3.4.1. Turvalisus

Kogu loodud infosüsteemi liiklus veebiserveri ja kasutaja veebilehitseja vahel käib läbi krüpteeritud (SSL) ühenduse.

Kasutaja autentimisel kasutatakse lisaks tavapärasele, e-maili aadressi ja parooliga sisselogimisele, mobiil-ID ja ID-kaardiga isikutuvastamise võimalusi.

3.4.2. Jõudlus

Autor on valinud rakenduse majutamiseks virtuaalses privaatserveris asuva Ubuntu 10.04 Linux'i. Serveri füüsiliseks asukohaks on London.

Kõnealuse operatsioonisüsteemi kasuks otsustas autor eelkõige tänu sellele, et selle Linux'i distributsiooni kasutamisega omab ta kõige suuremat kokkupuudet.

Täiendavalt kasutab rakendus oma töös veebiserverit Apache2, PHP 5.6, MySql versiooni 5.6 ning puhverdamiseks tarkvara Redis't.

Jõudluse parandamiseks kasutatakse andmebaasipäringute puhverdamist (*caching*) rakendades selleks Laravel'iga kaasa tulnud *Caching API* võimalusi.

3.4.3. Kasutatavus

Kogu rakenduse juures on käesoleva töö autor loonud kasutajaliidesed tuginedes oma teadmistele kasutatavuse valdkonnas.

Täiendavalt on viidud läbi testid kasutajatega, kes langevad kokku lapsevanema rollis oleva persoonaga (kirjeldatud täpsemalt lisas nr 1).

Vastavalt testimise tulemustele ning lapsevanematelt saadud tagasisidele on tehtud korrektuure avalduste esitamisega seotud kasutajaliideses.

3.4.4. Varundus (backupid)

Loodud infosüsteemi andmebaasi varundus toimub igal varahommik kasutades "Cron"[15] tööriista ja vastavat skripti. Skript teostab esmalt andmebaasist koopia kasutades "mysqldump" tööriista ning seejärel saadab koopia edasi teise serverisse maandamaks riski, kui rakenduse LIVE masinaga peaks füüsiliselt midagi juhtuma.

Täiendavalt teeb serveri teenusepakkuja iga nädal koopia kogu masinast.

3.5. Testimine

Loodava rakenduse testimine on toimunud paralleelselt arendusega. Lisaks arendaja esialgsetele testidele on kirjutatud tähtsama funktsionaalsuse jaoks funktsionaalsed testid (tuntud ka kui *Acceptance* testid*).*

Acceptance testid vastutavad selle eest, et kasutajalood (*user stories*) oleks täidetud. Teisisõnu, kas kasutaja seisukohalt teeb programm seda, mida vaja. (Bordo, 2010)

Testide loomisel on kasutatud Codeception raamistikku. Kuigi kõnealune raamistik võimaldab kasutada lisaks ka funktsionaalseid (*functional tests*) ning ühikteste (*unit tests*), ei ole käesoleva töö raames neid tehtud.

Acceptance teste jooksutatakse läbi veebilehitseja. Kuna tegemist on automaatsete testidega, mille puhul graafilise liidese olemasolu lehitsejal ei oma erilist tähtsust, siis on autor valinud testide jooksutamiseks niinimetatud "*headless*" veebilehitseja PhantomJS'i, mis kasutab WebKit mootorit. (Hidayat, 2015)

PhantomJS sai paigaldatud peremeesmasinasse, kuid testid ise käivitatakse otse arendusmasinast (vt ekraanipilt 12). Enne testide käivitamist tuleb seadistada testide kogum (*test suite)*. Vastav konfiguratsioonifail on näha ekraanipildil 11.

Konfiguratsioonifailis määratakse, millised moodulid on kasutusel ning mis on nende moodulite seadistused. Kuna käesolevas peatükis on näitena toodud üks kõige lihtsam

[15] UNIX süsteemides kasutatav tööriist, mis võimaldab käivitada programme kindlate ajavahemike tagant

kasutajalugu (*user story*), siis on näitena toodud konfiguratsioonifailist eemaldatud kõik ebavajalik (nt andmebaasi konfiguratsioon).

```
1   # Codeception Test Suite Configuration
2
3   class_name: AcceptanceTester
4   modules:
5       enabled:
6           - WebDriver
7           - AcceptanceHelper
8
9       config:
10          WebDriver:
11              url: 'http://intra.peetri.dev'
12              host: '192.168.1.129'
13              browser: phantomjs
```

Ekraanipilt 11 – acceptance testi konfiguratsioonifail

Codeception on *acceptance* testi kirjutamise teinud inimesele võimalikult loetavaks (vt koodinäide 7). Seega võiks lihtsamate testide kirjutamisega hakkama saada ka igapäevaselt programmeerimisega mitte kokku puutuvad inimesed.

```
1   /**
2    * Try to login as regular user
3    * @param \AcceptanceTester|\FunctionalTester $I
4    */
5   public function tryToLogIn(AcceptanceTester $I)
6   {
7       $I->am('parent of student in Peetri school');
8       $I->wantTo('Log in to the intranet');
9
10      $I->amOnPage('/login');
11
12      $I->fillField('E-posti aadress', 'acceptance@tester.ee');
13      $I->fillField('Parool', 'test');
14
15      $I->click('Sisene');
16
17      $I->see('Tere, ');
17  }
```

Koodinäide 7 – süsteemi sisse logimise automaattest

Esimeste ridadega (read 7-8) defineeritakse, mis rollis kasutaja on ja millist kasutajalugu antud test üritab läbi teha. Testi tegemine algab olles lehel `/login`. Rakenduse täielik aadress võetakse konfiguratsioonifailist (vt ekraanipilt 11).

Järgnevalt täidetakse sisselogimiseks nõutud e-posti ja parooli väli ning vajutatakse nupul "Sisene". Test loetakse läbituks, kui järgmisel lehel on kusagil kirjutatud fraas "Tere, " (vt lisa 4).

```
vagrant@homestead:~/Code/pintra/app$ codecept run acceptance --debug
Codeception PHP Testing Framework v2.0.13
Powered by PHPUnit 4.6.6 by Sebastian Bergmann and contributors.

Acceptance Tests (1) ------------------------------------------------------------
Modules: WebDriver, AcceptanceHelper
--------------------------------------------------------------------------------
Log in to the intranet (LoginCest::tryToLogIn)
Scenario:
* As a parent of student in Peetri school
* I am on page "/login"
* I fill field "E-posti aadress","acceptance@tester.ee"
* I fill field "Parool","test"
* I click "Sisene"
* I see "Tere, "
 PASSED

--------------------------------------------------------------------------------

Time: 11.81 seconds, Memory: 11.75Mb

OK (1 test, 1 assertion)
```

Ekraanipilt 12 – automaattesti käivitamine ja väljund

Testid jooksutatakse läbi enne uute arenduste *live*'i viimist virtuaalmasinas paiknevas testkeskkonnas. *Live*'i läheb uus arendus alles peale kõikide testide edukat läbimist.

Süsteemi edasise arenduse juures on arenduse kvaliteedi tagamiseks plaanis võtta kasutusele lisaks ka ühiktestid (*unit tests*).

4. Evalvatsioon

Käesoleva bakalaureusetöö eesmärgiks oli luua veebipõhine infosüsteem, mis automatiseeriks avalduste/taotluste esitamist, töövoogu kuni avalduse kinnitamise/tagasilükkamiseni ning olenevalt avalduse olemusest tulenevaid täiendavaid protsesse.

Töö andis ka ülevaate alates Peetri Lasteaed-Põhikoolis toimuvatest avalduste ja taotluste esitamisega seotud protsessidest nimetatud süsteemi loomise etappidest ning kasutatud tehnoloogiatest kuni reaalsete autori poolt välja töötatud lahendusteni.

Avalduste esitamise ning haldamise töö on muutunud tunduvalt lihtsamaks ning kiiremaks võimaldades kooli personalil kasutada aega ratsionaalsemalt ning kasulikumalt.

Kõige suurem ajaline võit on tõenäoliselt saavutatud huviringide gruppide automaatse moodustamise pealt. Käesoleva töö raames loodud süsteem võimaldab kokku hoida ligikaudu 2 nädalat ühe inimese tööd, mis oleks muidu kulunud sama tegevuse peale.

Kindlasti ei ole vähem tähtis kooli avalduste esitamine ning pikapäevarühma sööjate aruande genereerimine.

Eelnevale tuginedes võib lugeda eesmärgiks olnu saavutatuks. Arvestades kasutajatelt saadud tagasisidet ning kasutamisega seotud probleemide vähesusele, võib väita, et rakenduse kasutamine on üldiselt mugav ning arusaadav. Lapsevanematelt saadud tagasiside on tulnud kas e-kirja või otsesel vestlusel LiveChat'i[16] kaudu. Kooli töötajate tagasisidet on saadud lisaks veel läbi pidevate koosolekute ja Skype vestluste.

Tabelis 1 on näha kokkuvõtet infosüsteemi *live*'i minekust kuni käesoleva töö esitamise päevani esitatud avaldustest, kasutajate registreerimisest ning lisatud lastest.

[16] LiveChat tarkvara veebilehekülg asub järgmisel aadressil: http://www.livechatinc.com

Tabel 1– kuupäevaks 04. aprill 2015 tehtud toimingud

Toiming	Ühikud
kasutajate registreerumine/lisamine	1385
kellest lapsevanemad (sh kooli töötajad)	676
kellest lapsed	709
esitatud avaldusi kokku	937
millest eelkooli avaldusi	100
millest pikapäevarühmas osalemise avaldusi	128
millest huviringis osalemise avaldusi	562
millest kooli õppima asumise taotlusi	147

4.1.Tekkinud probleemid

Projekti arenduse käigus tulid välja mõned probleemid, mille peamiseks põhjuseks loeb autor väga lühikest perioodi, mis jäi antud funktsionaalsuse implementeerimiseks.

Serveri jõudlus huviringitaotluste esitamisel

Esialgne virtuaalne privaatserver oli küllaltki väikse jõudlusega, mistõttu oli planeeritud serveri vahetamine esimesel võimalusel, kui arendusest aega üle jääb. Kuna huviringi avalduste aeg jõudis kätte enne serveri vahetust, mis tõi kaasa üle 300 samaaegse kasutaja süsteemis, siis ei pidanud server sellele koormusele vastu.

Serveri üleviimiseks võimsama riistvara peale tuli lükata huviringi avalduste esitamise alguspäeva edasi kahe päeva võrra, mis tähendas ka suure hulga pahaste kasutajate ees vabandamist.

Teine katse läks edukalt ning esimese 3-4 minutiga esitati läbi süsteemi ligikaudu 300 avaldust huviringis osalemiseks.

Pikapäevarühma taotlused ja söömine

Rakenduses on liidetud kokku pikapäevarühma ning pikapäeva söögiavaldused.

Reaalsuses oleks tulnud need kaks avaldust süsteemis lüüa lahku. See, et avaldused on koos, ei anna lapsevanematele piisavalt head ülevaadet söömistest, mis tõi kaasa järgmise situatsiooni tekkimise:

- lapsevanemal oli tehtud pikapäevarühma taotlus, millele oli märkinud, et soovib süüa kaks korda nädalas, teisipäeval ning neljapäeval;
- sama lapse kohta oli esitatud ka huviringi taotlus, kuhu oli samuti märkinud, et soovib osaleda pikapäevarühma söömises. Kui nüüd lapsevanem soovis lapse täielikult eemaldada pikapäevarühma söömisest, siis tegi ta seda muutes ainult pikapäevarühma avaldust. Huviringi avalduselt söömise soovi ta ei eemaldanud, mille tulemusena tekkisid vead sööjate aruande väljavõttes. Väljavõtte alusel esitas valla raamatupidaja lapsevanematele arved kuigi lapsevanem enda teada võttis lapse söömisest maha.

Süsteemis on avaldused, mille puhul on võimalik osaleda pikapäevarühma söömises. Iga söögikorra eest läheb lapsevanemale kuu lõpus koondarve. Avalduse sisestamisel läheb arve saajaks vaikimisi selle esitaja andmed. Pikapäevatoidu koondraportisse võetakse maksja andmed esimese kehtiva avalduse pealt. See tähendab, et näiteks juhul kui lapsel on 3 kehtivat avaldust, millel on märgitud ka soov saada pikapäevatoitu ning millest esimese esitas ema, teised kaks aga isa, siis arve saadetakse söömiste eest ainult emale.

Tuli välja, et mitte kõik lapsevanemad ei ole enda lapse osalemise kohta esitanud avaldust läbi selleks loodud infosüsteemi. Osad lapsevanemad on läinud otse rühmajuhendaja jutule, kes on nad otse võtnud rühma. Selle tulemusena käis pikapäevatoitu söömas hulk lapsi, kes tegelikult seal ei oleks pidanud käima. Tõenäoliselt oleks selle probleemi vältimiseks tulnud tegeleda laialdasema teavitustööga kooli poolel.

5. Võimalused edasiseks arenduseks

Käesoleva töö raames valminud süsteem on peamiselt küll keskendunud avalduste ja taotluste esitamisega seotud funktsionaalsusele, kuid tõenäoliselt läheb süsteemi skoop edaspidi laiemaks. Järgnevas peatükis antakse lühiülevaade planeeritud edasistest arendustest, mis on osaliselt juba kokku lepitud ka tellijaga.

Kalendrimoodul

Kalendrimoodul peab võimaldama sisestada üritusi (sisestab ainult huvijuht). Sisestatud ürituste jaoks tuleb teha API ots, mille kaudu saaks kätte avalikuna märgitud üritused ning mida oleks võimalik kuvada peatselt loodaval uuel avalikul veebilehel. Samuti peab saama märkida läbi kalendri päevi, millal ei toimu pikapäevatoidu söömist.

Koolivormi tellimine

Hetkel käib koolivormi tellimine avalikult kodulehelt. Kuna käesoleva töö raames loodud ning hetkel peamiselt avalduste/taotluste esitamiseks mõeldud infosüsteemis on olemas andmed nii laste kui ka lapsevanemate kohta, siis ilmselt tuleb koolivormi tellimise lahendus samuti loodud süsteemi ringi teha. Kuna koolivormi tellimine on mõeldud puhtalt koolis õppivate laste vanematele (kellel süsteemis konto nagunii juba olemas on), siis selle liigutamine käesoleva töö raames loodud süsteemi oleks igati loogiline käik.

Täiendused huviringide moodulis

Seetõttu, et huviringides osalejad muutuvad pidevalt, on tekkinud vajadus luua huviringide jaoks järjekorrasüsteem. Ehk kui huviringi rühmades on maksimum arv osalejaid, siis seda huviringi mitte ei peideta esitatavate huviringide nimekirjast (nagu praeguse loodud süsteemi puhul), vaid lapsevanem saab lisada end järjekorda. Koha vabanemisel teavitatakse sellest lapsevanemat, kes seejärel saab esitada vastava avalduse või loobuda kohast. Loobumisel teavitatakse sellest järjekorras järgmist lapsevanemat.

Täiendused sööjate aruandes

Sööjate aruannet tuleks täiendada nii, et kogu info oleks korrektne (st likvideerida peatükis 4.1 kirjeldatud puudused). Kui puudused on likvideeritud, siis peaks aruanne saadetama valla raamatupidaja e-mailile iga kuu viimasel päeval automaatselt.

Sidumine väliste infosüsteemidega

Käesoleva töö raames ei ole tehtud ühtegi liidestamist väliste infosüsteemidega. Küll aga on planeeritud luua andmete eksportimise võimalus riikliku registri EHIS (Eesti Hariduse Infosüsteem) tarbeks.

Kokkuvõte

Bakalaureusetöö tulemusena anti ülevaade avalduste esitamisega seotud protsessidest, mis kulutavad kooli personali arvestatavat ressurssi ning loodi rakendus, mis lihtsustab ning automatiseerib kooli personali poolt kulutatavat aega, mis kulub lastevanemate poolt esitatavate avalduste ja taotlustega tegelemiseks. Samuti lihtsustab ja kiirendab loodud rakendus märgatavalt huviringi rühmadesse jaotamist ning uute õpilaste klassidesse jaotamist.

Käesoleva töö raames valminud avalduste esitamise infosüsteemi loomisel sai üha selgemaks, et sellisest keskkonnast tuntakse puudust. Üheks indikaatoriks oli süsteemi loomise ajal kahest erinevast koolist tulnud päring ligilähedase funktsionaalsusega süsteemi osas. Nähes, et sellisest süsteemist tunneb puudust mitu kooli, sai lisaks viidud läbi lühike uuring 70 Harjumaal, Pärnumaal ja Tartumaal asuva kooli seas.

Tulemused annavad veelkord kinnitust, et koolidel on sellise süsteemi vastu huvi olemas. Kokku oli vastanuid umbes 33 kooli, millest avalduste esitamise võimalus läbi veebikeskkonna eksisteeris vähem kui kümnel. Enamikul neist oli kasutusel Google Forms[17] tarkvara, mis sisuliselt lihtsalt kogub andmed ühte kohta kokku. Maja siseselt arendatud või kusagilt tellitud süsteem, mille kaudu erinevaid avaldusi vastu võetakse, oli olemas ainult paaril üksikul koolil.

Kuna enamikes Eesti koolides käib avalduste esitamine ikkagi paberil, mis tähendab lisaks paberi kulule suurel määral ka inimressursse, siis oleks väga tõenäoline pakkuda sarnast avalduste esitamise lahendust ka teistes koolides.

Kuivõrd avalduste esitamise protsess ning avaldustel täidetavad väljad ei ole kindlasti kõikides koolides ühesugused (st, et puudub riiklik standard), tuleks uute avalduse tüüpide lisamine teha dünaamiliseks. See tähendab, et avaldused ning nendel küsitavad infoväljad peaksid olema süsteemihaldusest dünaamiliselt hallatavad. Selle saavutamiseks tuleks teha ringi avalduste mooduli süsteemiarhitektuur.

[17] Google Forms tarkvara detailsem kirjeldus asub järgmisel aadressil: https://www.google.com/forms/about/

Küll aga hakkab järjekordse uue veebikeskkonna lisandumisega tekkima järgmine probleem – neid süsteeme, mida kool oma töös kasutab on väga palju erinevaid ning enamasti puudub nende vahel automaatne andmevahetus, mis tähendab kooli personalile lisatööd.

Loodud süsteemi ja soovi korral ka koodipoolega saab lähemalt tutvuda võttes ühendust käesoleva bakalaureusetöö autoriga.

Summary: Application Submission Information System: the example of Peetri Primary School

Although the use of ICT in education and in schools is quite well developed in most of the schools in Estonia, people are still applying for school the old-fashioned way by printing out paper, filling it using a pen and then bringing it physically to the institution.

School employees are putting a lot of work into scanning through the received applications. In some extreme cases people are spending even weeks while handling the submitted data.

The aim of this B.A thesis was to find a solution and to develop a web-based information system for automating processes related to submission of different types of applications. The system is tailor-made for the administration of Peetri Primary School, but can be fitted to others as well.

The thesis consists of five chapters. The first chapter gives an overview of the processes currently being carried out in Peetri school.

The second one gives an overview of the information system planning process. Which where the functional requirements and which were the constraints, also known as the non-functional requirements.

Third chapter concentrates on the implementation by first describing what tools and technologies were used. Secondly an overview of actual solutions which the author found out to the problems related the application submission progresses.

Fourth chapter evaluates the outcome and analysis implementation process.

Fifth and the last chapter describes further development opportunities.

The system developed during this thesis is not ready yet and the author will definitely keep on working on it because he believes people in schools should spend their time on giving better education than putting a lot of time into paperwork.

Kasutatud kirjandus

Hashimoto, M. (2013). *Vagrant: Up and Running.* (C. N. Mike Loukides, Toim.) Sebastopol, CA, Ameerika Ühendriigid: O'Reilly Media, Inc.

Peetri Lasteaed-Põhikool. (31. jaanuar 2012. a.). *Dokumentatsioon.* Kasutamise kuupäev: 6. jaanuar 2015. a., allikas Peetri Lasteaed-Põhikool: http://www.peetri.edu.ee/abifailid/pdf/Arengukava.pdf

Gerchev, I. (11. detsember 2013. a.). *The 5 Most Popular Frontend Frameworks of 2014 Compared*. Kasutamise kuupäev: 4. veebruar 2015. a., allikas Sitepoint: http://www.sitepoint.com/5-most-popular-frontend-frameworks-compared/

DELFI. (31. august 2013. a.). *Peetri kool muutus poole suuremaks*. Kasutamise kuupäev: 2. veebruar 2015. a., allikas Delfi: http://www.delfi.ee/news/paevauudised/eesti/peetri-kool-muutus-poole-suuremaks?id=66666775

Servinski, M., Hänilane, B., Kivilaid, M., Tischler, G., & Ülle, V. (kuupäev puudub). *Eesti piirkondlik areng. 2014. Regional Development in Estonia.* Allikas: Eesti Statistika: http://www.stat.ee/publication-download-pdf?publication_id=36388

Azad, K. (27. september 2007. a.). *A Visual Guide to Version Control*. Kasutamise kuupäev: 15. märts 2015. a., allikas Better Explained: http://betterexplained.com/articles/a-visual-guide-to-version-control/

Chacon, S., & Straub, B. (2014). *Pro Git* (II tr.). New York City: Apress.

Oracle Corporation. (2015). *What is MySQL?* Kasutamise kuupäev: 23. märts 2015. a., allikas MySQL: http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html

Otwell, T. (15. jaanuar 2015. a.). *Laravel Homestead*. Kasutamise kuupäev: 5. aprill 2015. a., allikas Laravel: http://laravel.com/docs/5.0/homestead

Riley, S. (veebruar 2006. a.). *Password Security: What Users Know and What They Actually Do*. Kasutamise kuupäev: 24. aprill 2015. a., allikas Usability News: http://psychology.wichita.edu/surl/usabilitynews/81/Passwords.asp

The PHP Group. (29. aprill 2015. a.). *serialize*. Kasutamise kuupäev: 29. aprill 2015. a., allikas PHP.net: http://php.net/manual/en/function.serialize.php

Bocoup. (12. mai 2014. a.). *GRUNT: The JavaScript Task Runner*. Kasutamise kuupäev: 28. aprill 2015. a., allikas Grunt: http://gruntjs.com/

Nielsen, J. (4. jaanuar 2012. a.). *Usability 101: Introduction to Usability*. Kasutamise kuupäev: 30. aprill 2015. a., allikas Nielsen Norman Group: http://www.nngroup.com/articles/usability-101-introduction-to-usability/

Hidayat, A. (30. aprill 2015. a.). *Headless Testing*. Kasutamise kuupäev: 30. aprill 2015. a., allikas PhantomJS: http://phantomjs.org/headless-testing.html

HashiCorp. (3. mai 2015. a.). *Default Provider*. Kasutamise kuupäev: 3. mai 2015. a., allikas Vagrant Documentation: https://docs.vagrantup.com/v2/providers/default.html

Boggiano , J., & Adermann, N. (2. mai 2015. a.). *Getting Started*. Kasutamise kuupäev: 2. mai 2015. a., allikas Composer: https://getcomposer.org/doc/00-intro.md

Stellman, A. (3. oktoober 2009. a.). *Using nonfunctional requirements to build better software*. Kasutamise kuupäev: 29. aprill 2015. a., allikas Building Better Software: http://www.stellman-greene.com/2009/10/03/using-nonfunctional-requirements/

Otwell, T. (15. jaanuar 2015. a.). *Query Builder*. Kasutamise kuupäev: 3. mai 2015. a., allikas Laravel: http://laravel.com/docs/5.0/eloquent

Otwell, T. (15. jaanuar 2015. a.). *Eloquent ORM*. Kasutamise kuupäev: 3. mai 2015. a., allikas Laravel: http://laravel.com/docs/5.0/eloquent

Fowler, M. (jaanuar 2003. a.). *Active Record*. Kasutamise kuupäev: 2. mai 2015. a., allikas Catalog of Patterns of Enterprise Application Architecture: http://www.martinfowler.com/eaaCatalog/activeRecord.html

Skvorc, B. (28. märts 2015. a.). *Best PHP Framework for 2015 – SitePoint Survey Results*. Kasutamise kuupäev: 1. mai 2015. a., allikas SitePoint: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

Otwell, T. (15. jaanuar 2015. a.). *Artisan CLI*. Kasutamise kuupäev: 1. mai 2015. a., allikas Laravel: http://laravel.com/docs/5.0/artisan

Fowler, M. (18. juuli 2006. a.). *GUI Architectures*. Kasutamise kuupäev: 16. veebruar 2015. a., allikas http://martinfowler.com/eaaDev/uiArchs.html

Refsnes Data. (1. aprill 2015. a.). *AngujarJS Tutorial*. Kasutamise kuupäev: 29. aprill 2015. a., allikas W3Schools.com: http://www.w3schools.com/angular/

Git and Mercurial code management for teams. (1. märts 2015. a.). Kasutamise kuupäev: 3. märts 2015. a., allikas Bitbucket: https://bitbucket.org/#features

Bordo, V. (20. jaanuar 2010. a.). *Overview of User Acceptance Testing (UAT) for Business Analysts (BAs)*. Kasutamise kuupäev: 2. mai 2015. a., allikas DevelopMentor: https://www.develop.com/useracceptancetests

HashiCorp Inc. (01. mai 2015. a.). *About Vagrant*. Kasutamise kuupäev: 01. mai 2015. a., allikas Vagrant: https://www.vagrantup.com/about.html

Tooming, U. (19. september 2012. a.). *Peetri lasteaed–põhikooli hakatakse laiendama*. Kasutamise kuupäev: 05. jaanuar 2015. a., allikas Postimees: http://tallinncity.postimees.ee/978102/peetri-lasteaed-pohikooli-hakatakse-laiendama

Võõrkeelsete lühendite loetelu

CSS – *Cascading StyleSheets*, astmelised stiililehed, mida kasutatakse veebirakenduste disainimisel.

PHP – *Hypertext Preprocessor*, serveripoolne skriptimiskeel dünaamiliste veebilehekülgede loomiseks.

SASS - *Syntactically Awesome Stylesheets*, metakeel, mille eesmärgiks on laiendada CSS'i võimalusi.

CSV – *Comma-Separated Values*, universaalne failivorming, milles andmed on üksteisest eraldatud semikoolonite või tabelduskla hvi abil.

ORM – *Object-Relational Mapping*, objekt-relatsiooniline kaardistamine

SQL – *Structured Query Language*, päringukeel andmebaasis paikneva andmestikuga toimetamiseks

SSL – *Secure Sockets Layer*, infoturbe protokoll

SSH – *Secure Shell*, turvaline võrguprotokoll krüpteeritud sessioonide pidamiseks teiste (remote) masinatega

UAT – *User Acceptance Testing*, kindla tegevuse või funktsiooni toimivuse testimine

API – *Application Programming Interface*, reeglistik olemasoleva valmisprogrammiga suhtlemiseks, mis võimaldab programmeerijatel kirjutada lisaprogramme esialgse funktsionaalsuse täiendamiseks

LISAD

Lisa 1. Persoona 1.

Vanus: 25

Haridus:

Amet: assistent

Mirjam

Mirjam on noor lapsevanem, kelle esimene laps õpib kooli teises klassis ning noorem laps on just minemas esimesse klassi.

Mirjamile meeldib paberivaba asjaajamine ning soovib, et kõik kooliga seotud paberil esitatavad avaldused oleks tehtavad ka elektrooniliselt.

Eesmärgid: kuna kooli astumisega seoses on väga palju toimetamist paberavalduste ning sinna juurde kuuluvate dokumentidega, siis soovib Mirjam, et avaldusi oleks võimalik täita mugavalt otse Interneti teel.

Foto 1. Autor Naom Chen [https://www.flickr.com/photos/israelphotogallery/14555927522]

Lisa 2: Persoona 2.

Vanus: 28

Haridus: kasvatusteaduste bakalaureus

Amet: registripidaja

Stella

Stella on kooli registripidaja, kelle peamiseks ülesandeks on hallata lastevanemate ning kooli õpilaste registreid.

Samuti tegeleb Stella kooli astuvate õpilaste koordineerimisega ning üldise õpilaste liikumisega (kooli õppima tulijad ning koolist lahkujad). Lisaks on Stella ülesanded lastevanemate ja lastega seotud info haldamine ja registrite korrastamine.

Eesmärgid: kuna kooli astumisega seoses on väga palju aeganõudvat toimetamist paberavalduste ning sinna juurde kuuluvate dokumentidega, siis soovib Stella, et avalduste esitamiseks oleks mingisugune veebikeskkond, mis lihtsustaks sellega seonduvaid toiminguid.

Foto 2 – Autor Big D2112 [https://www.flickr.com/photos/bigd2112/5025686582]

Lisa 3: Persoona 3.

Vanus: 41

Haridus: organisatsioonikäitumise bakalaureus

Amet: majandusjuhataja

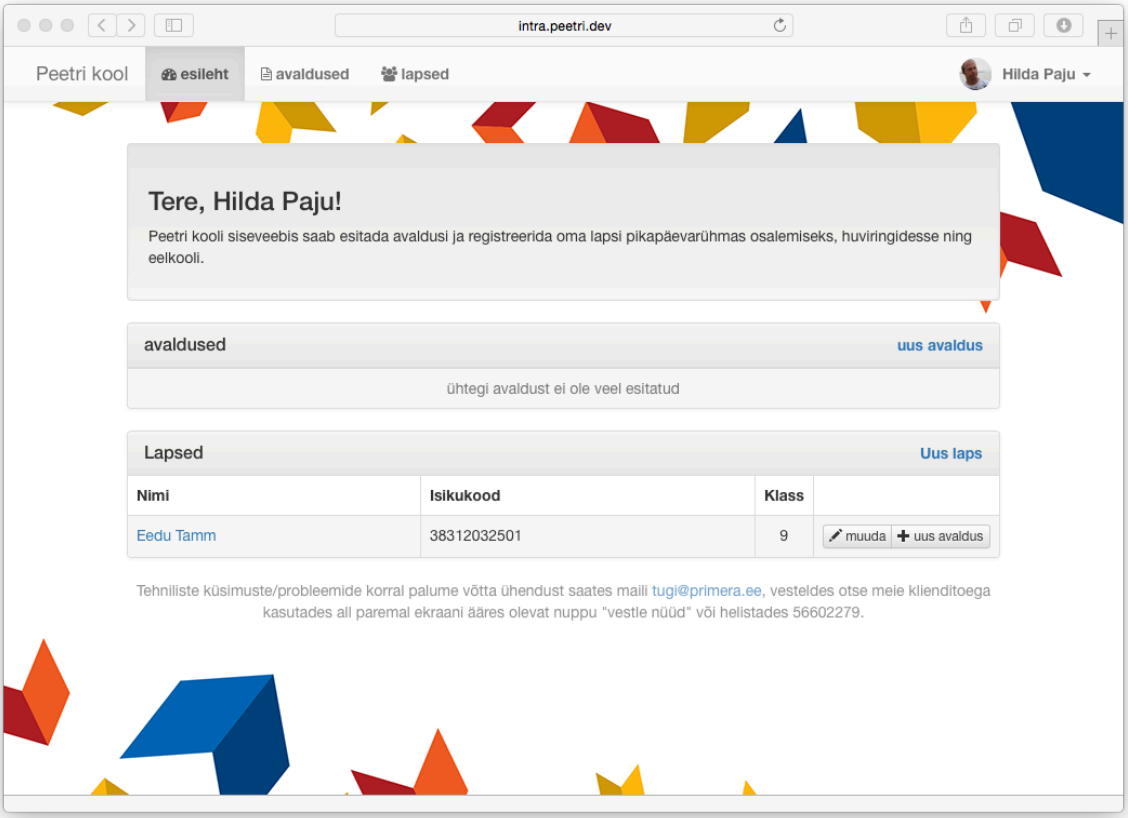
Merike

Merike on kooli majandusjuhataja, kelle ülesannete hulka kuulub kooli sööjate aruannete koostamine ning edastamine valla raamatupidajale.

Eesmärgid: kuna pikapäevarühma avalduste ja söömise koondaruande koostamine on küllaltki aeganõudev töö, siis sooviks Merike, et koolis oleks selleks tarbeks kasutusel mingisugune süsteem, seda tööd lihtsustaks.

Foto 3 – Autor Mike McCormick [https://www.flickr.com/photos/mike_mccormick/2716870664]

Lisa 4: avalduste esitamise infosüsteemi esileht (tavakasutaja)



Lisa 5: Eelkooli taotlus (esialgne)

EELKOOLI TAOTLUS

Palun registreerida minu laps ……..... eelkooli
ees- ja perekonnanimi

Lapse andmed:

Isikukood ……..…

Kodune keel ……...……

Elukoht …….....……

Laps käib/ei käi ..... lasteaias.
lasteaia nimi

Õppeaastal 2014/2015 on osalustasu 160 eurot.
Osalustasu maksmine toimub Rae valla poolt esitatava arve alusel (detsembris 40 €, veebruaris 40 €, aprillis 40€ ja juunis 40 €).
Eelkooli eest tasutud summadele tulumaksu soodustust ei rakendata.

Minu lapsel on õigus osaleda eelkoolis osalustasu maksmata, kuna talle ei ole tagatud munitsipaallasteaias lasteaiakohta enne kooli minekut. ..…….
jah/ei

Lapsevanema andmed:

Vanema (maksja) ees- ja perekonnanimi ..….....…….....

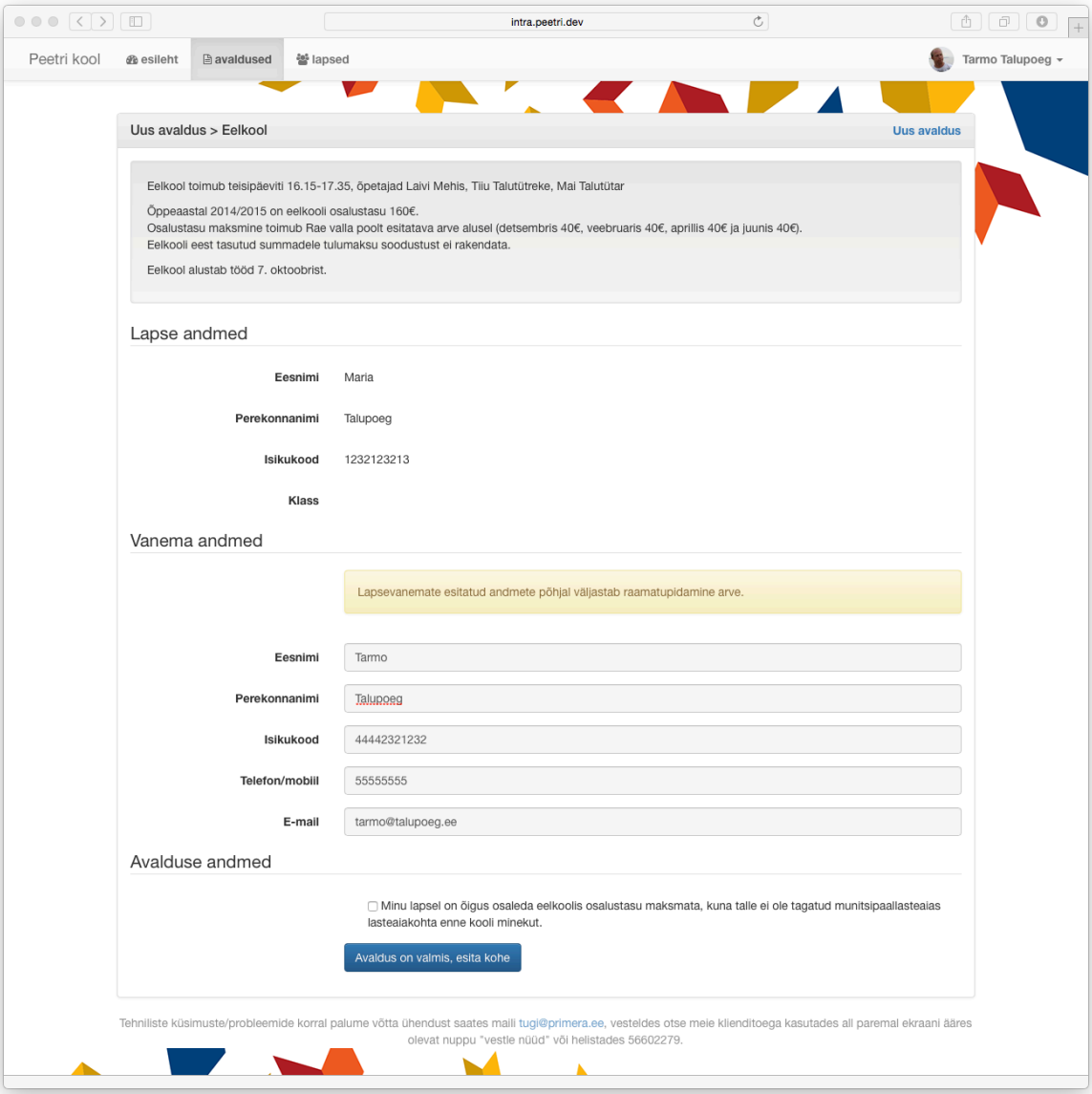
Isikukood.....

E-post …….....……..

Kontakttelefon …….....……

….....….. .....….....
(allkiri) (kuupäev)

Lisa 6: Eelkooli taotlus (elektrooniline)



Lisa 7. Kooli astumise taotlus (esialgne)

Peetri Kooli direktor

TAOTLUS

Palun võtta minu laps.....
(ees- ja perekonnanimi trükitähtedega)

Peetri Põhikooli ..... klassi alates ..... õppeaastast.

Lapse andmed:

Isikukood |__|__|__|__|__|__|__|__|__|__|__|

Kodune aadress ..... Indeks .....

Taotluse esitaja andmed:

Ees- ja perekonnanimi .....

Isikukood |__|__|__|__|__|__|__|__|__|__|__|

Telefon .....

E-post .....

Millisest lasteasutusest/ koolist Teie laps meie kooli tuleb? .....

.....

Meie perest õpib Peetri koolis õde/vend .....

.....

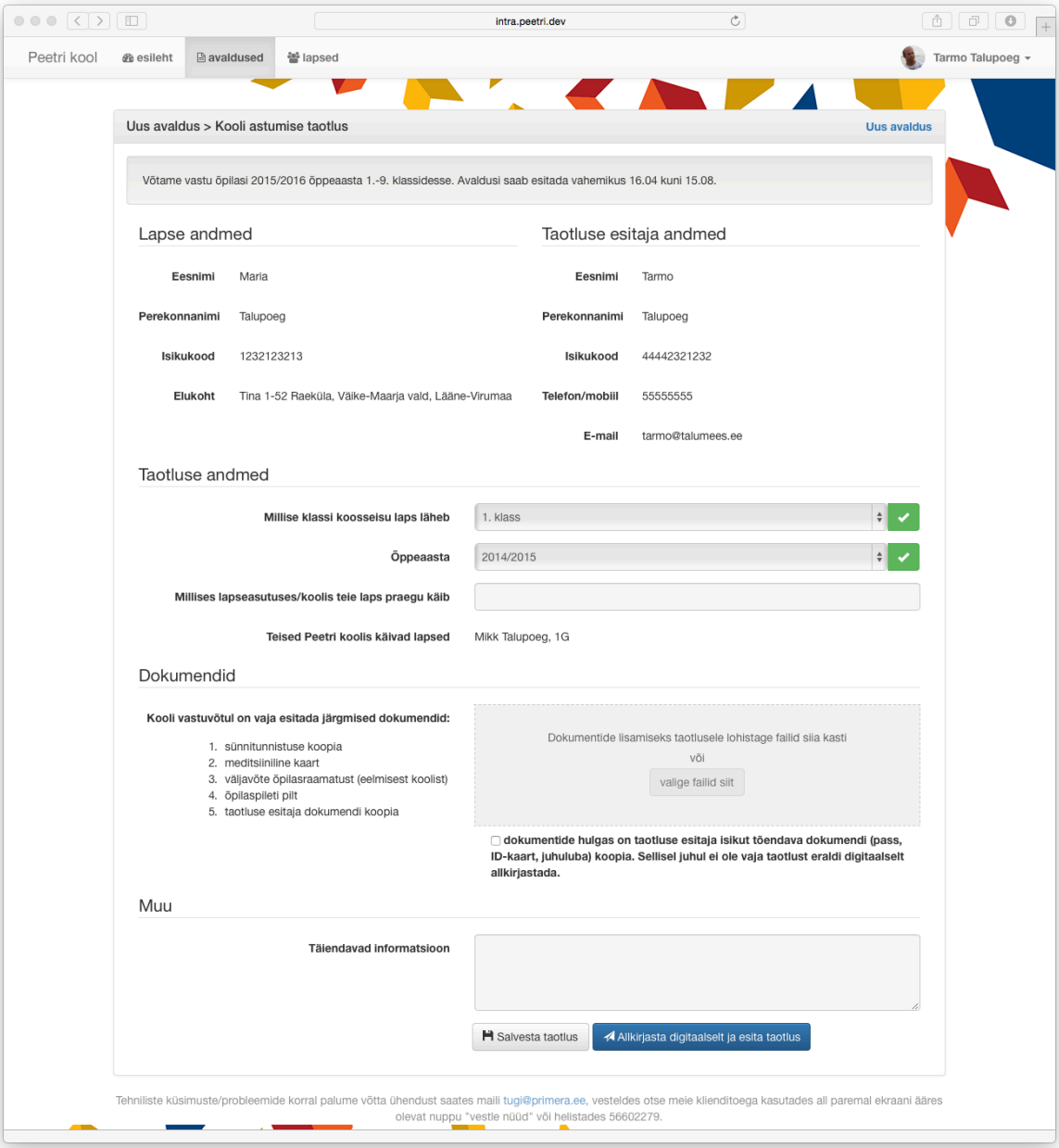
Kooli vastuvõtul on vaja esitada järgmised dokumendid:

- ☐ Sünnitunnistuse koopia
- ☐ Meditsiiniline kaart
- ☐ Väljavõte õpilasraamatust (eelmisest koolist)
- ☐ Õpilaspileti pilt

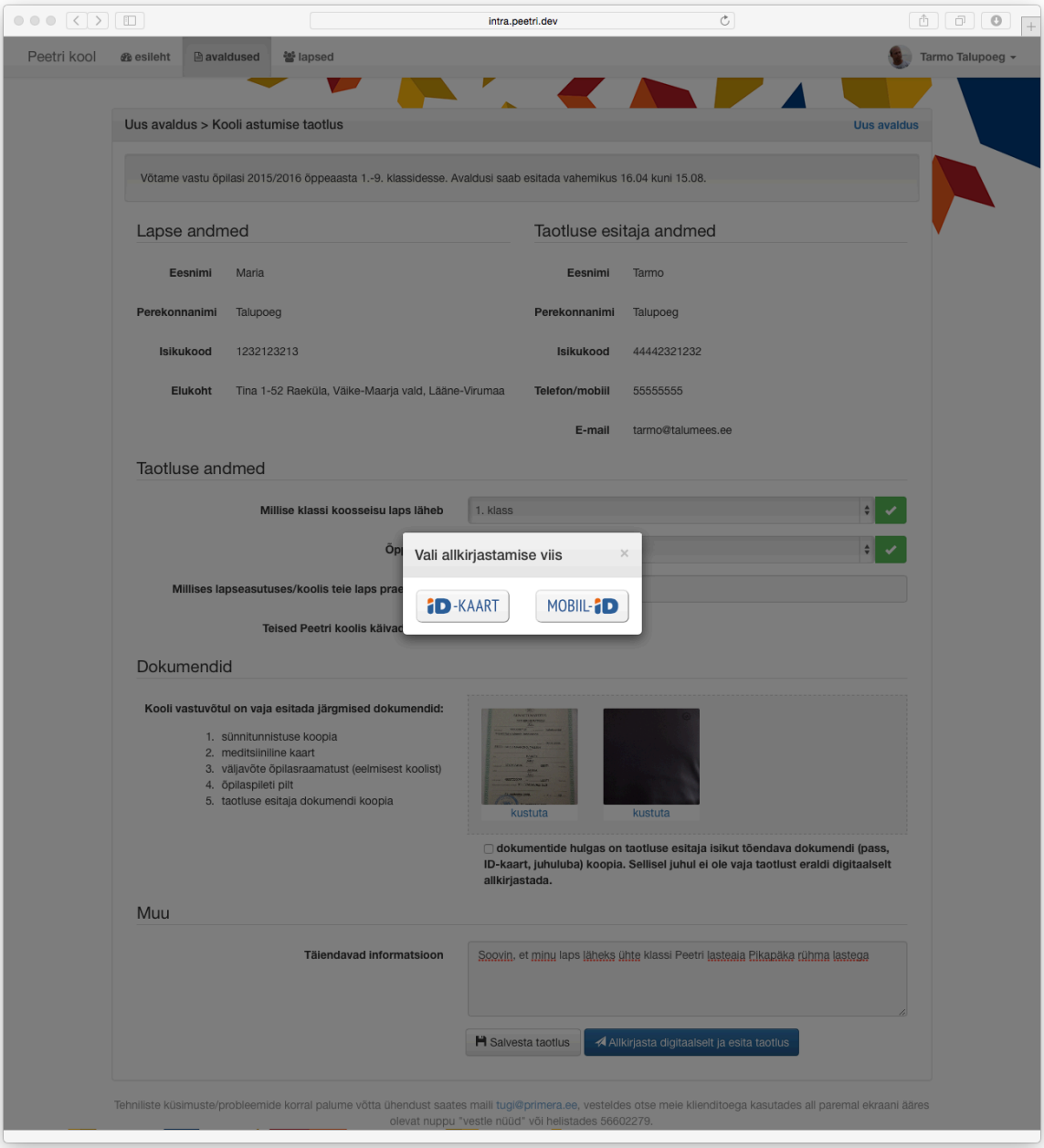
..... |__|__| |__|__| |__|__|__|__|

Taotluse esitaja allkiri Kuupäev (pp/kk/aaaa)

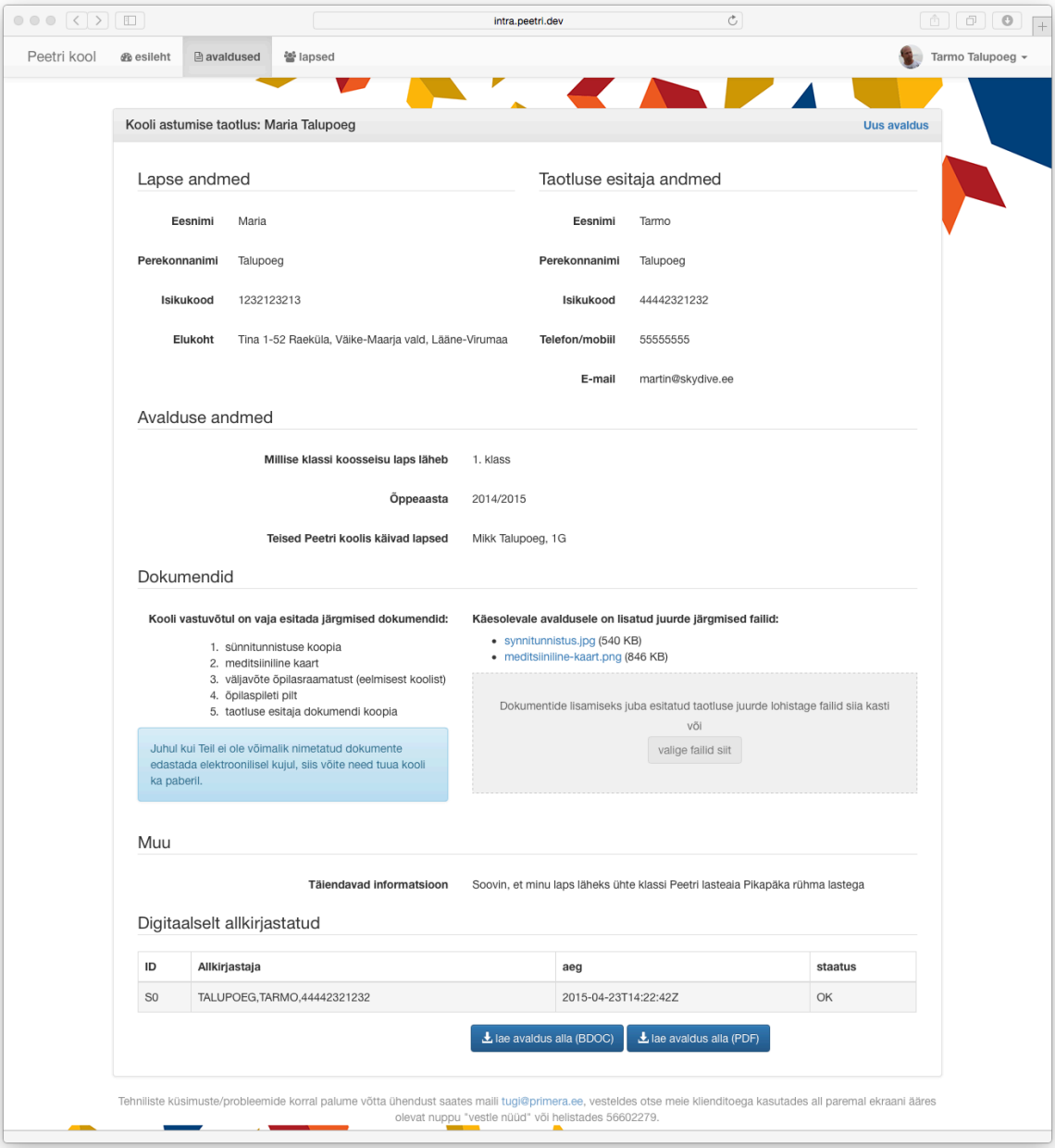
Lisa 8: Kooli astumise taotlus (elektrooniline)



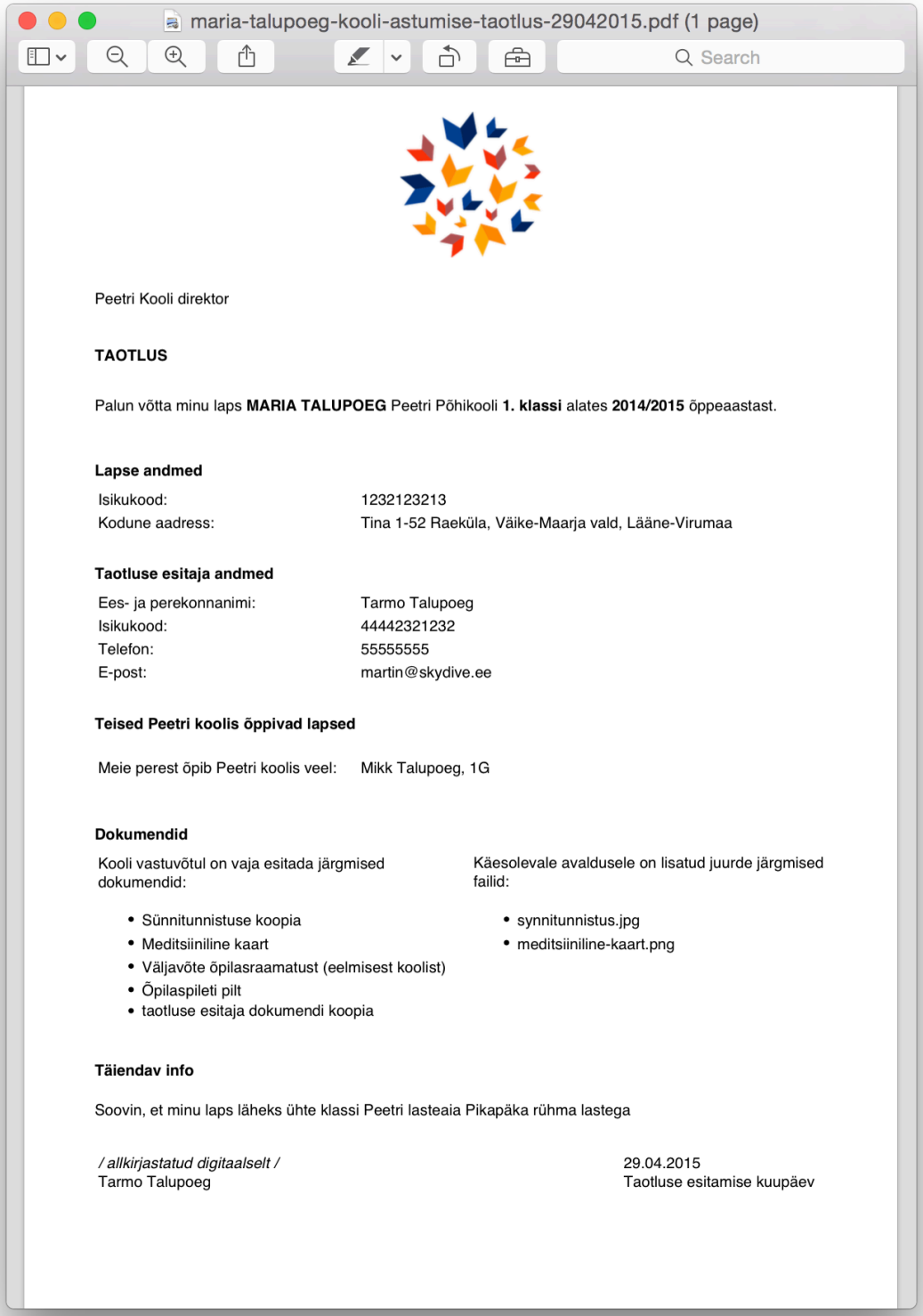
Lisa 9: Kooli astumise taotluse (täidetud) digitaalne allkirjastamine



Lisa 10: Esitatud avalduse vaade (kooli astumise taotlus)



Lisa 11: Esitatud taotluse trükis (PDF) arvutisse laetuna



maria-talupoeg-kooli-astumise-taotlus-29042015.pdf (1 page)

Search

Peetri Kooli direktor

TAOTLUS

Palun võtta minu laps **MARIA TALUPOEG** Peetri Põhikooli **1. klassi** alates **2014/2015** õppeaastast.

Lapse andmed

Isikukood:	1232123213
Kodune aadress:	Tina 1-52 Raeküla, Väike-Maarja vald, Lääne-Virumaa

Taotluse esitaja andmed

Ees- ja perekonnanimi:	Tarmo Talupoeg
Isikukood:	44442321232
Telefon:	55555555
E-post:	martin@skydive.ee

Teised Peetri koolis õppivad lapsed

Meie perest õpib Peetri koolis veel:	Mikk Talupoeg, 1G

Dokumendid

Kooli vastuvõtul on vaja esitada järgmised dokumendid:

- Sünnitunnistuse koopia
- Meditsiiniline kaart
- Väljavõte õpilasraamatust (eelmisest koolist)
- Õpilaspileti pilt
- taotluse esitaja dokumendi koopia

Käesolevale avaldusele on lisatud juurde järgmised failid:

- synnitunnistus.jpg
- meditsiiniline-kaart.png

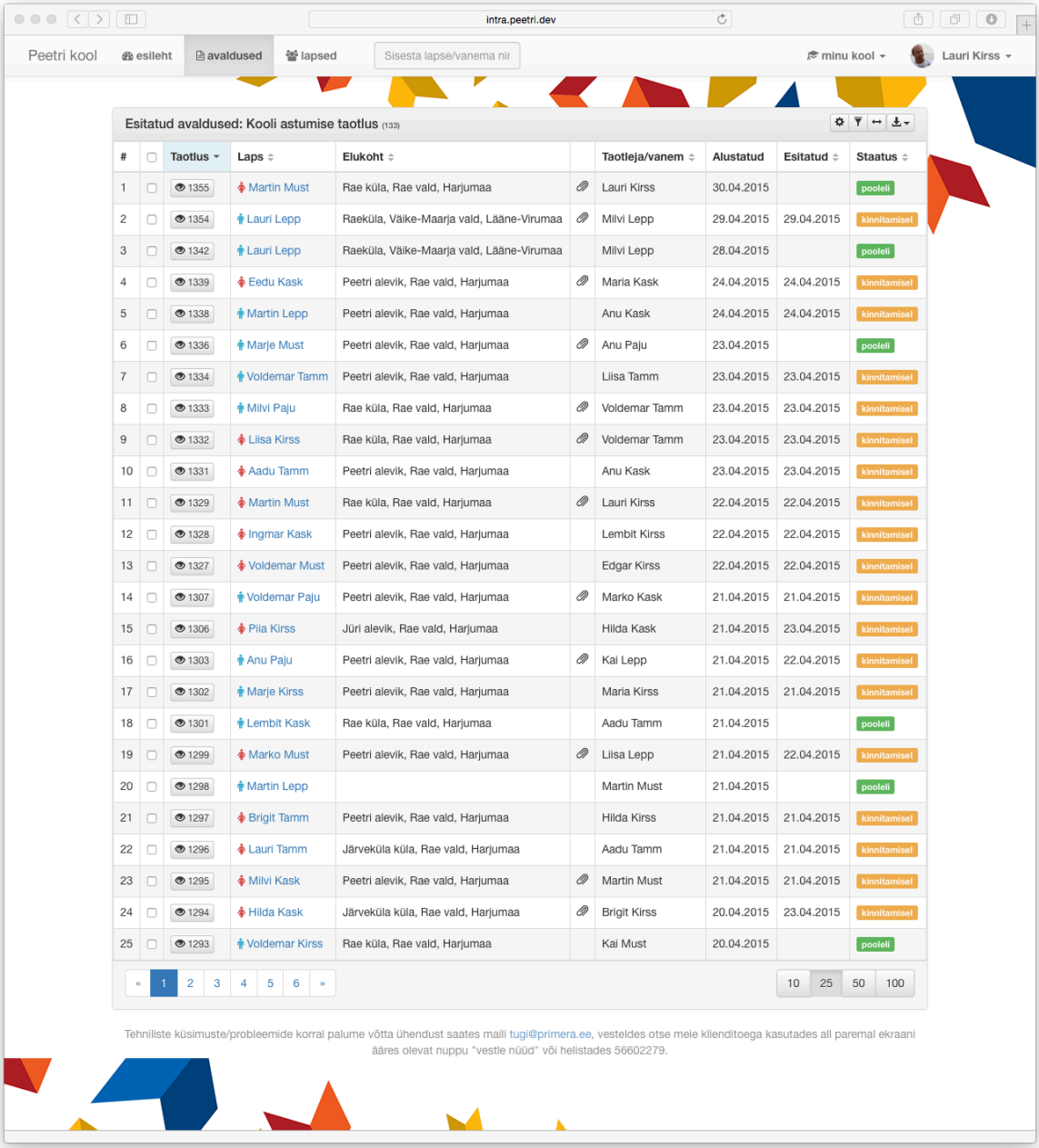
Täiendav info

Soovin, et minu laps läheks ühte klassi Peetri lasteaia Pikapäka rühma lastega

/ allkirjastatud digitaalselt /
Tarmo Talupoeg

29.04.2015
Taotluse esitamise kuupäev

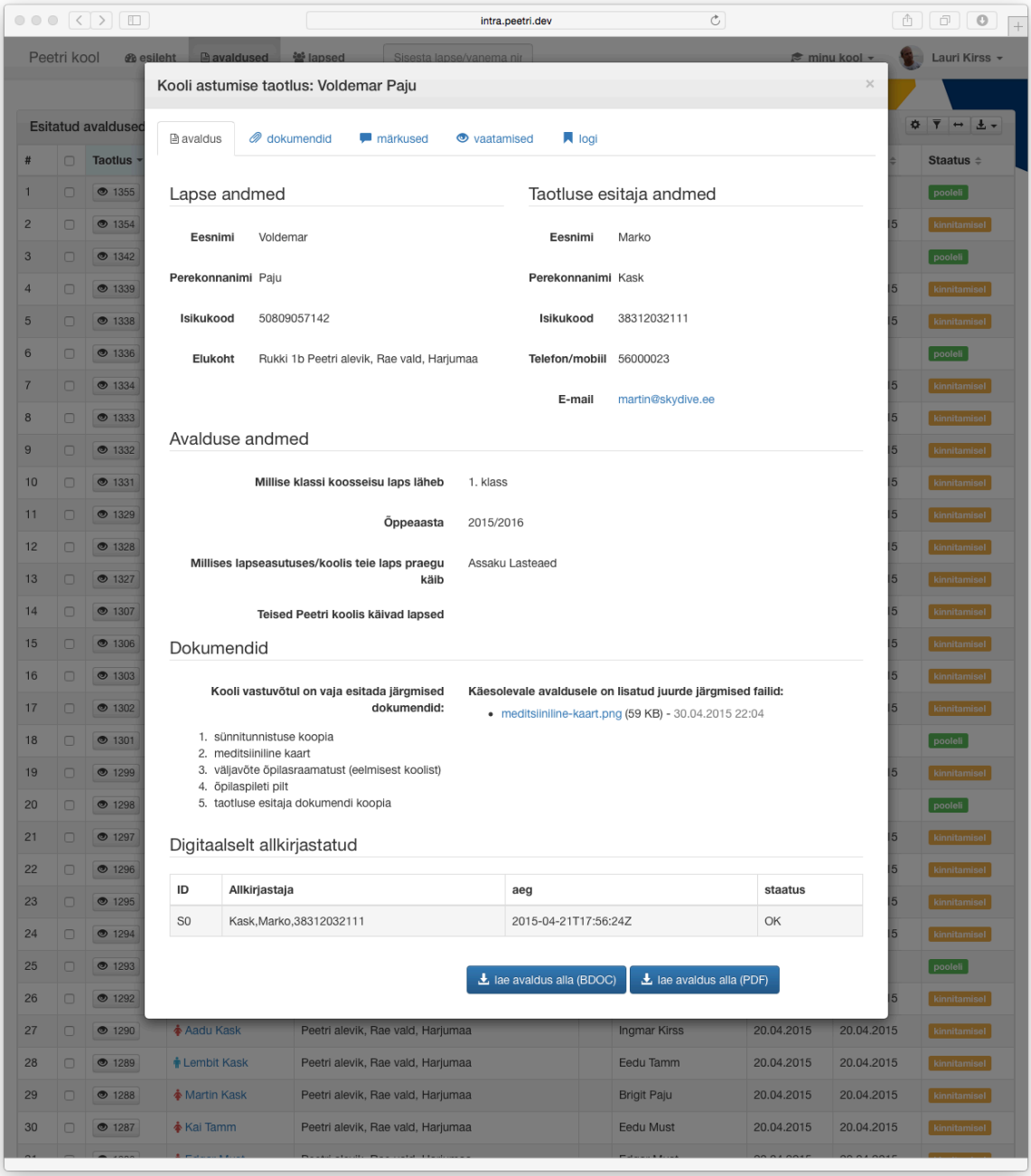
Lisa 12: Esitatud avaldused (kooli astumise taotlus)



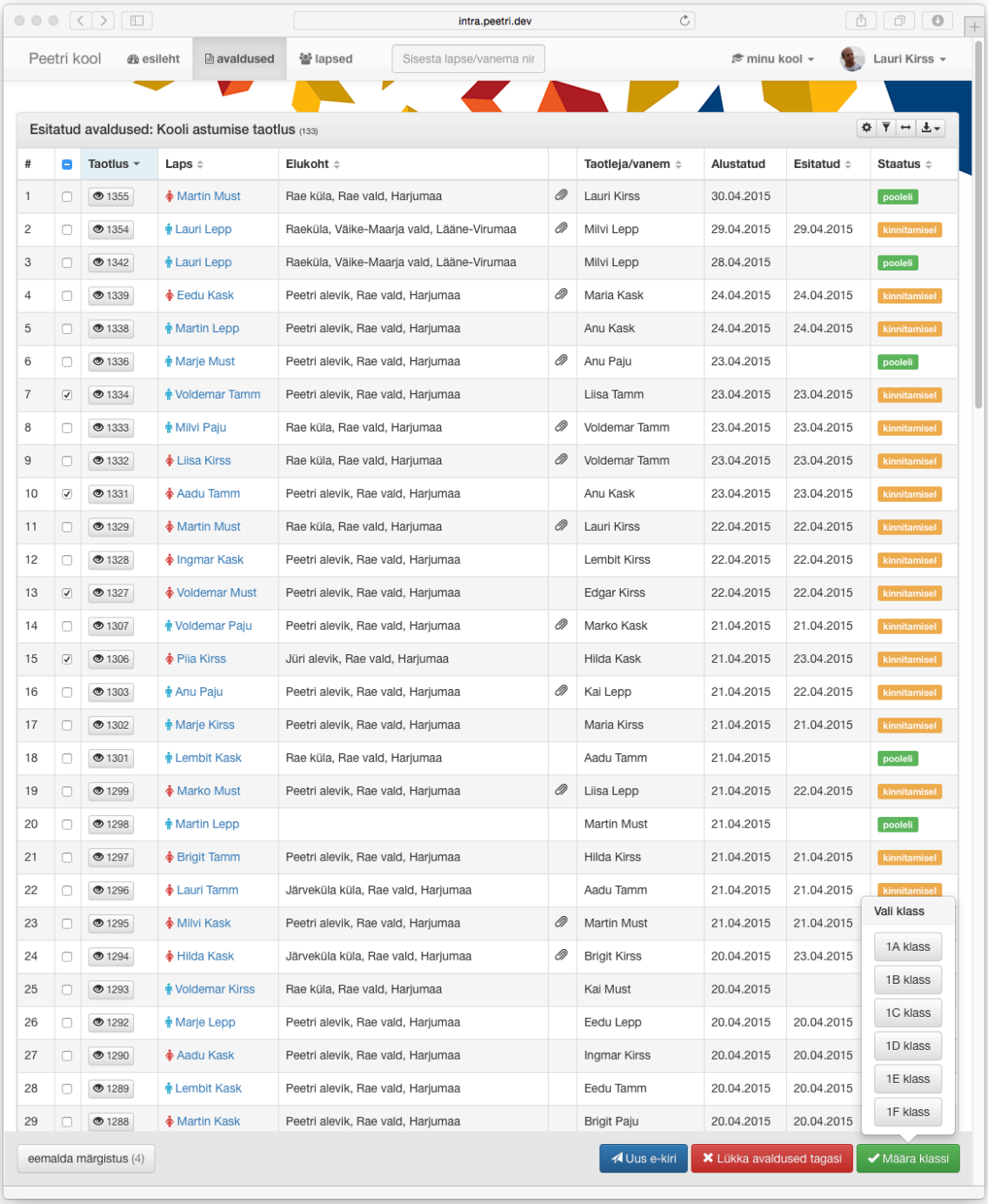
#	Taotlus	Laps	Elukoht	Taotleja/vanem	Alustatud	Esitatud	Staatus
1	1355	Martin Must	Rae küla, Rae vald, Harjumaa	Lauri Kirss	30.04.2015		pooleli
2	1354	Lauri Lepp	Raeküla, Väike-Maarja vald, Lääne-Virumaa	Milvi Lepp	29.04.2015	29.04.2015	kinnitamisel
3	1342	Lauri Lepp	Raeküla, Väike-Maarja vald, Lääne-Virumaa	Milvi Lepp	28.04.2015		pooleli
4	1339	Eedu Kask	Peetri alevik, Rae vald, Harjumaa	Maria Kask	24.04.2015	24.04.2015	kinnitamisel
5	1338	Martin Lepp	Peetri alevik, Rae vald, Harjumaa	Anu Kask	24.04.2015	24.04.2015	kinnitamisel
6	1336	Marje Must	Peetri alevik, Rae vald, Harjumaa	Anu Paju	23.04.2015		pooleli
7	1334	Voldemar Tamm	Peetri alevik, Rae vald, Harjumaa	Liisa Tamm	23.04.2015	23.04.2015	kinnitamisel
8	1333	Milvi Paju	Rae küla, Rae vald, Harjumaa	Voldemar Tamm	23.04.2015	23.04.2015	kinnitamisel
9	1332	Liisa Kirss	Rae küla, Rae vald, Harjumaa	Voldemar Tamm	23.04.2015	23.04.2015	kinnitamisel
10	1331	Aadu Tamm	Peetri alevik, Rae vald, Harjumaa	Anu Kask	23.04.2015	23.04.2015	kinnitamisel
11	1329	Martin Must	Rae küla, Rae vald, Harjumaa	Lauri Kirss	22.04.2015	22.04.2015	kinnitamisel
12	1328	Ingmar Kask	Peetri alevik, Rae vald, Harjumaa	Lembit Kirss	22.04.2015	22.04.2015	kinnitamisel
13	1327	Voldemar Must	Peetri alevik, Rae vald, Harjumaa	Edgar Kirss	22.04.2015	22.04.2015	kinnitamisel
14	1307	Voldemar Paju	Peetri alevik, Rae vald, Harjumaa	Marko Kask	21.04.2015	21.04.2015	kinnitamisel
15	1306	Piia Kirss	Jüri alevik, Rae vald, Harjumaa	Hilda Kask	21.04.2015	23.04.2015	kinnitamisel
16	1303	Anu Paju	Peetri alevik, Rae vald, Harjumaa	Kai Lepp	21.04.2015	22.04.2015	kinnitamisel
17	1302	Marje Kirss	Peetri alevik, Rae vald, Harjumaa	Maria Kirss	21.04.2015	21.04.2015	kinnitamisel
18	1301	Lembit Kask	Rae küla, Rae vald, Harjumaa	Aadu Tamm	21.04.2015		pooleli
19	1299	Marko Must	Peetri alevik, Rae vald, Harjumaa	Liisa Lepp	21.04.2015	22.04.2015	kinnitamisel
20	1298	Martin Lepp		Martin Must	21.04.2015		pooleli
21	1297	Brigit Tamm	Peetri alevik, Rae vald, Harjumaa	Hilda Kirss	21.04.2015	21.04.2015	kinnitamisel
22	1296	Lauri Tamm	Järveküla küla, Rae vald, Harjumaa	Aadu Tamm	21.04.2015	21.04.2015	kinnitamisel
23	1295	Milvi Kask	Peetri alevik, Rae vald, Harjumaa	Martin Must	21.04.2015	21.04.2015	kinnitamisel
24	1294	Hilda Kask	Järveküla küla, Rae vald, Harjumaa	Brigit Kirss	20.04.2015	23.04.2015	kinnitamisel
25	1293	Voldemar Kirss	Rae küla, Rae vald, Harjumaa	Kai Must	20.04.2015		pooleli

« 1 2 3 4 5 6 » 10 25 50 100

Lisa 13: Üksiku avalduse kaart



Lisa 14: Avalduste alusel klassidesse lisamine



Lisa 15: Pikapäeva rühma taotlus (esialgne)

PIKAPÄEVARÜHMA REGISTREERIMISLEHT

Palun registreerida minu laps .....

pikapäevarühma alates ..... 2014 a.

Laps õpib Peetri Lasteaed-Põhikooli ..... klassis.

Olen huvitatud lapse osalemisest pikapäevarühmas järgnevatel päevadel ja ajavahemikel

Nädalapäev	**Pikapäevarühmas viibimise aeg**	**Soovin toitlustamist**
E☐	....	.....
T☐	....	.....
K☐	....	.....
N☐	....	.....
R☐	....	.....

LAPSEVANEMA ANDMED:

Nimi .....

Isikukood .....

Kontakttelefon .....

E-post .....

....

(kuupäev) (allkiri)

Lisa 16: nädala sööjate aruanne

#	Õpilase nimi	Klass	E	T	K	N	R	Kokku
1	Kask Aadu	1E	jah	jah		jah		3
2	Kask Aadu	2C			jah			1
3	Kask Anu	1B	jah					1
4	Kask Brigit	2A			jah			1
5	Kask Edgar	5C			jah			1
6	Kask Eedu	4B				jah		1
7	Kask Hilda	1D		jah			jah	2
8	Kask Ingmar	3A		jah	jah			2
9	Kask Ingmar	5A	jah		jah	jah		3
10	Kask Kai	3A		jah	jah			2
11	Kask Kai	2A			jah			1
12	Kask Lauri	1D	jah		jah			2
13	Kask Lauri	1B		jah		jah	jah	3
14	Kask Lembit	1C	jah		jah	jah	jah	4
15	Kask Liisa	1B					jah	1
16	Kask Marje	3E			jah			1
17	Kask Marje	2C			jah	jah		2
18	Kask Marko	4C			jah			1
19	Kask Piia	3D		jah				1
20	Kask Voldemar	1A	jah		jah			2
21	Kask Voldemar	2D	jah					1
22	Kask Voldemar	1C					jah	1
23	Kirss Aadu	1G	jah		jah	jah		3
24	Kirss Aadu	5A	jah	jah				2
25	Kirss Aadu	6A	jah		jah			2
26	Kirss Anu	1E	jah	jah		jah		3
27	Kirss Anu	3D				jah		1
28	Kirss Brigit	4B		jah		jah		2
29	Kirss Edgar	1B	jah	jah	jah	jah	jah	5
30	Kirss Edgar	1G	jah	jah	jah	jah		4
31	Kirss Edgar	3A			jah			1
32	Kirss Edgar	1E	jah	jah	jah	jah	jah	5
33	Kirss Eedu	1G	jah	jah	jah	jah	jah	5
34	Kirss Hilda	3C		jah	jah			2
35	Kirss Hilda	6B	jah	jah				2
36	Kirss Hilda	1A	jah	jah	jah	jah	jah	5
37	Kirss Hilda	1A	jah		jah			2
38	Kirss Ingmar	3E			jah			1
39	Kirss Kai	1D	jah	jah	jah	jah	jah	5
40	Kirss Lauri	1D			jah	jah		2
41	Kirss Lauri	2G	jah			jah		2
42	Kirss Lembit	4B			jah			1
43	Kirss Lembit	4A	jah		jah			2
44	Kirss Lembit	2B				jah		1
45	Kirss Lembit	5A	jah	jah				2
46	Kirss Liisa	1D					jah	1
47	Kirss Marje	5A	jah		jah			2
48	Kirss Marje	3C				jah		1
49	Kirss Marko	5B				jah		1
50	Kirss Martin	3A	jah					1
51	Kirss Martin	3A		jah		jah		2
52	Kirss Milvi	2A			jah			1
53	Kirss Piia	2B			jah	jah		2

Lisa 17: 2014/2015 õppeaastal avatavad huviringid

1.	1.-2. klassi lauluring	Toimub kolmapäeviti 6. tund, osalevad Peetri kooli neiud .
2.	3.-5. klassi mudilaskoor	Toimub neljapäeviti 6. Ja 7. tund, osalevad tüdrukud
3.	Plokkflööt	Toimub kolmapäeviti 7. tund, osalevad õpilased eelkõige 4a klassist, rühma suurus 12 õpilast.
4.	Poistekoor	Toimub esmaspäeviti ja kolmapäeviti 6. tund, osalevad 1.-5. klassi poisid.
5.	Muusika- ja pilliring	Toimub esmaspäeviti 7. tund ja teisipäeviti 6 tund, osalevad 1.-5. klassi õpilased, rühma suurus 12 õpilast mõlemal päeval.
6.	Kitarr	Toimub teisipäeviti ja kolmapäeviti 8. tund, rühma suurus on 8 õpilast.
7.	Trumm	Toimub kolmapäeviti 6., 7. ja 8. tund, rühma suurus on 12 õpilast
8.	Kunst	Toimub kolmapäeviti 7. tund 3.-4. klassidele ja neljapäeviti 6. tund 1. klassidele, 7. tund 2. klassidele, 8 tund 5.-9. klassidele. Rühmas on 16 õpilast.
9.	Scrapbooking + muud meisterdused	Toimub teisipäeviti 7. ja 8. tund, osalevad 2.-9. klassi õpilased, rühma suurus on 16 õpilast.
10.	Puutöö	Toimub esmaspäeviti 7. tund 2.-4. klassid, 8 tund 5.-9. klassid. Rühmas on 12 õpilast.
11.	Draama	Toimub esmaspäeviti ja kolmapäeviti 6. tund, osalevad on 3. ja 4. klassi õpilased, rühmas on 16 õpilast.
12.	1. klassi näitering	Toimub esmaspäeviti ja kolmapäeviti 5. tund, mõlemas rühmas on 16 õpilast
13.	5. klassi näitering	Toimub kolmapäeviti 6. ja 7 tund, rühmas on 16 õpilast.
14.	6.-8. klassi näitering	Toimub kolmapäeviti 8. ja 9. tund, rühmas on 16 õpilast.
15.	Estraaditants	Toimub reedeti 7. ja 8. tund, osalevad 4.-5. klassi õpilased, rühmas on 16 õpilast.
16.	Rahvatants	Toimub teisipäeviti 6. tund ja kolmapäeviti 7. tund, osalevad 2.-3. klassi õpilased, rühmas on 16 õpilast
17.	Sulgpall	Toimub kolmapäeviti 7. tund, osalevad

		4.-5 klassi õpilased, rühmas on 16 õpilast.
18.	Grossing	Toimub esmaspäeviti 8. tund 1. klassidele, 9. tund 2. klassidele ja neljapäeviti 7. tund 1.-2. klassidele, 8. tund 3.-4. klassidele, 9. tund 5.-9. klassidele. Rühma suurus on 12 õpilast.
19.	Jooga	Toimub neljapäeviti 6. tund, osalevad 1.-2. klassi õpilased, rühmas on 16 õpilast.
20.	Matkaring	Toimub teisipäeviti 7. tund, osalevad 4.-6. klassi õpilased. Rühma suurus on 14 õpilast.
21.	Hispaania keel	Toimub esmaspäeviti ja teisipäeviti 8. tund, osalevad õpilased alates 5. klassist, rühma suurus on 16 õpilast.
22.	Hiina keel	Toimub esmaspäeviti ja kolmapäeviti 7. tund 4.-5. klassidele, 8. tund 6.-9. klassidele. Rühma suurus on 16 õpilast.
23.	Vene keele ja kultuuri ring 1.-4. klassile	Toimub esmaspäeviti 6. tund, rühma suurus on 16 õpilast.
24.	Vene keele ja kultuuri ring 5.-9. klassile	Toimub teisipäeviti 8. tund, rühma suurus on 16 õpilast.
25.	Kokaklubi	Toimub teisipäeviti 6., 7. ja 8. tund, osalevad 3.-4. klassi õpilased, rühma suurus on 12 õpilast.
26.	Animatsioon	Toimub teisipäeviti 7. tund, osalevad 3.-6. klassi õpilased, rühma suuru son 16 õpilast.
27.	Robootika	Toimub neljapäeviti 7. ja 8 tund, osalevad 3.-4. klassi õpilased, rühma suurus on 16 last.
28.	Mõttemängud	Toimub teisipäeviti 5. tund 2. klassidele ja reedeti 5. tund 1. klassidele, rühma suurus on 16 last.
29.	Avastusring	Toimub kolmapäeviti 6. ja 7 tund, osalevad 2. klassi õpilased, rühma suurus on 16 õpilast.
30.	3D graafika	Toimub esmaspäeviti 8. ja 9. tund, osalevad 6.-9. klassi õpilased, rühma suurus on 16 õpilast.
31.	Ajalugu	Toimub kolmapäeviti 8. tund, osalevad 5.-9. klassi õpilased, rühma suurus on 16 õpilast.

Lisa 18: Grunt konfiguratsioonifail

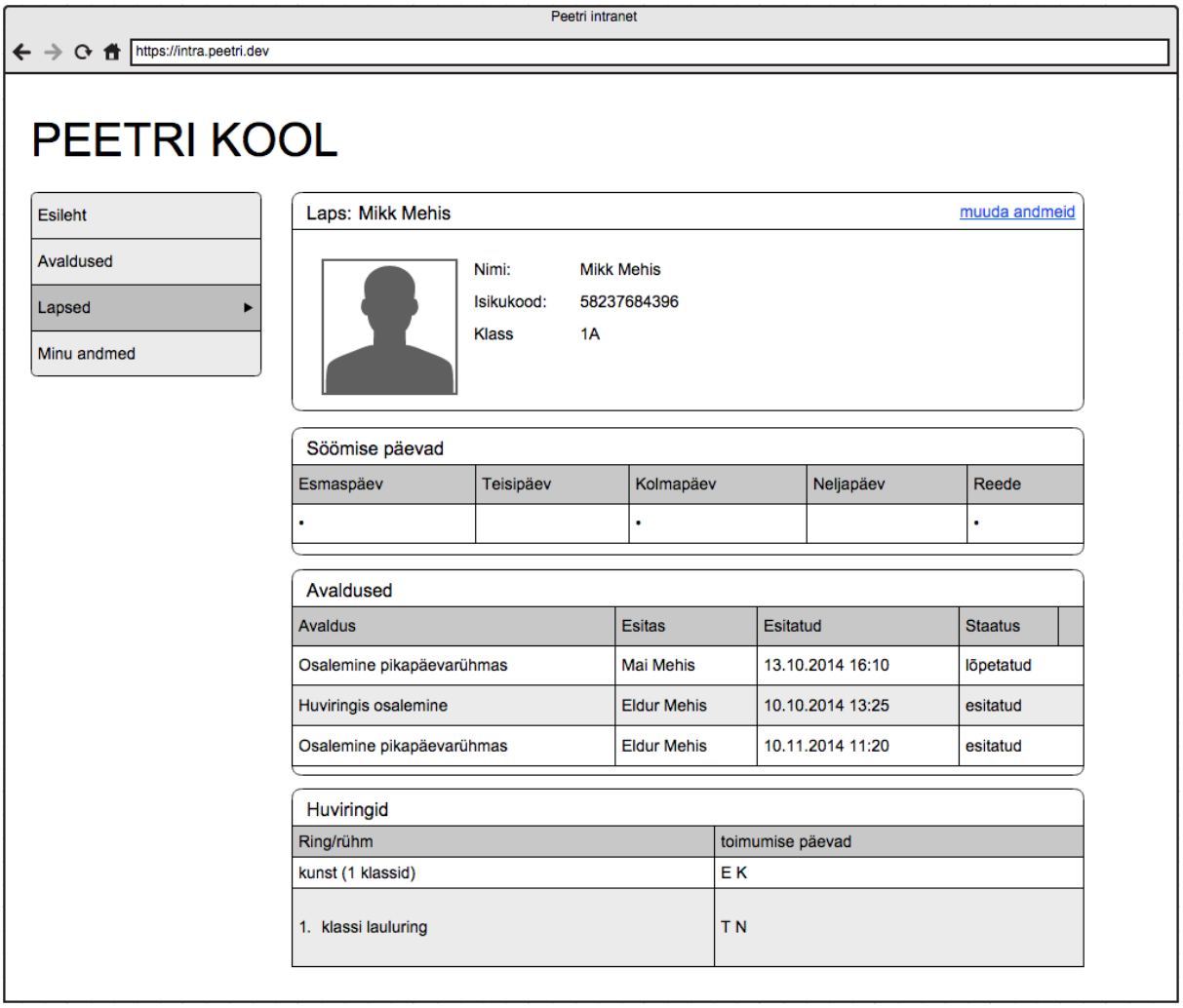
```
1   module.exports = function(grunt) {
2       grunt.initConfig({
3           concat: {
4               js: {
5                   src: [
6                       './app/assets/js/*.js'
7                   ],
8                   dest: './public/assets/js/app.js'
9               }
10          },
11          uglify: {
12              js: {
13                  files: {
14                      ./public/assets/js/app.js': ./public/assets/js/app.js',
15                      ./public/assets/js/*.js': ./app/assets/js/controllers/*.js'
16                  }
17              }
18          },
19          sass: {
20              development: {
21                  files: {
22                      ./public/assets/css/app.css":./app/assets/css/app.scss",
23                      ./public/assets/css/excel.css":./app/assets/css/excel.scss"
24                  }
25              }
26          },
27          watch: {
28              js: {
29                  files: [
30                      ./app/assets/js/*/*.js',
31                      ./app/assets/js/*.js'
32                  ],
33                  tasks: ['concat:js', 'uglify:js']
34              },
35              sass: {
36                  files: ['./app/assets/css/*.scss'],
37                  tasks: ['sass']
38              }
39          },
40          imagemin: {
41              png: {
42                  options: {
43                      optimizationLevel: 7
44                  },
45                  files: [
46                      {
47                          expand: true,
48                          cwd: './app/assets/img/',
49                          src: ['**/*.png'],
50                          dest: './public/assets/img/',
51                          ext: '.png'
52                      }
52                  ]
53              }
54          }
55      });
56
57      grunt.loadNpmTasks('grunt-contrib-concat');
58      grunt.loadNpmTasks('grunt-contrib-sass');
59      grunt.loadNpmTasks('grunt-contrib-uglify');
60      grunt.loadNpmTasks('grunt-contrib-watch');
61      grunt.loadNpmTasks('grunt-contrib-imagemin');
62
63      grunt.registerTask('default', ['watch']);
64  }
```

Lisa 19: Model::save() meetod

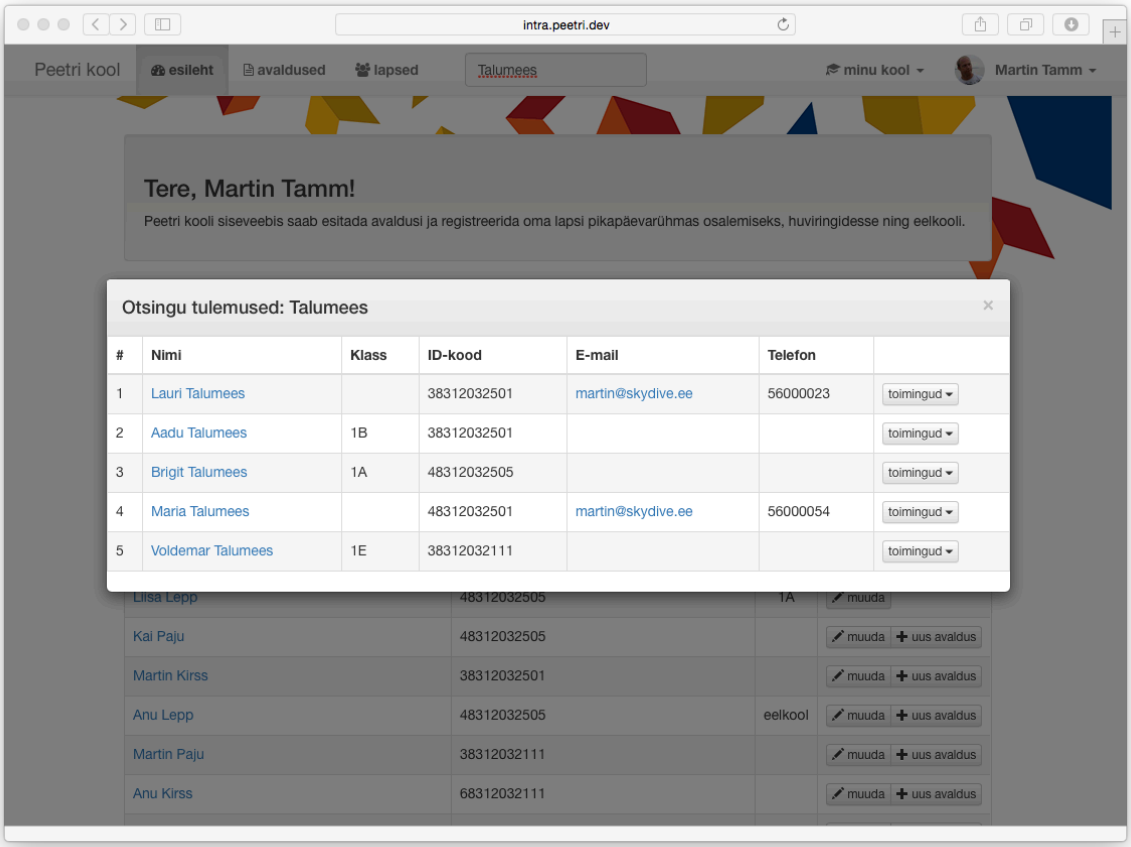
```
1   public function save(array $options = [])
2   {
3       $props = [];
4       if ($this->hasProps)
5       {
6           // Get props
5           foreach($this->attributes as $field => $value)
6           {
7                if (starts_with($field, 'prop_'))
8                {
9                    $value = trim($value);
10                   if ($value) $props[substr($field, 5)] = $value;
12                   unset($this->attributes[$field]);
13               }
14           }
15
16          $callerClass = strtolower(get_called_class());
17          $propTableName = ($this->propTableBaseName ? $this->propTableBaseName.'_props' :
$callerClass.'_props');
18          $propTableIdName = ($this->propTableBaseName ? $this->propTableBaseName.'_id' :
$callerClass.'_id');
19
20          // Save main object data in order to get ID for prop entries
21           $saved = parent::save($options);
22          if ($saved && !empty($props))
23          {
24              $savedProps = [];
25              foreach ($this->props->toArray() as $savedProp)
26              {
27                  $savedProps[$savedProp['name']] = $savedProp['value'];
28              }
29
30              // Store props
31             foreach ($props as $field => $propVal)
32             {
33                 // Skip existing and same values
34                 if (array_key_exists($field, $savedProps) && $savedProps[$field] == $propVal)
35                     continue;
36
37                 // if does not exist
38                 $createData = [];
39                 if (!array_key_exists($field, $savedProps))
40                 {
41                     $createData[] = [
42                         $propTableIdName    => $this->id,
43                         'name'              => $field,
44                         'value'             => $propVal
45                     ];
46                 }
47
```

```
48                  // if has changed
49                  if (array_key_exists($field, $savedProps) && $savedProps[$field] != $propVal)
50                  {
51                      DB::table($propTableName)->where([$propTableIdName => $this->id, 'name' =>
$field])->update(['value' => $propVal]);
52                  }
53
54                  // Do database updates
55                  if ($createData) DB::table($propTableName)->insert($createData);
56              }
57          }
58      }
59      else
68      {
61          $saved = parent::save($options);
62      }
63
64      return $saved;
65  }
```

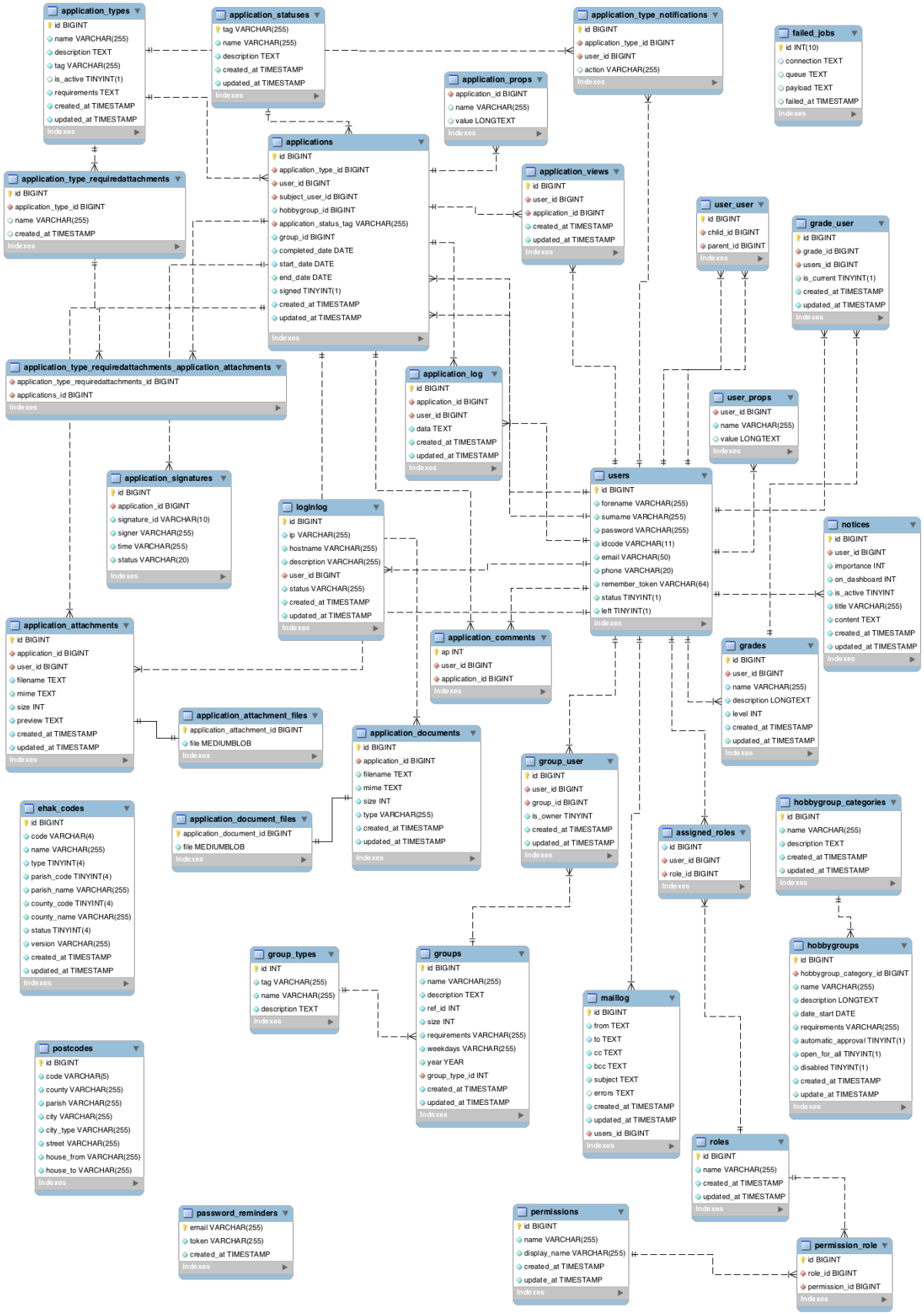
Lisa 20: kasutajakaart (prototüüp)



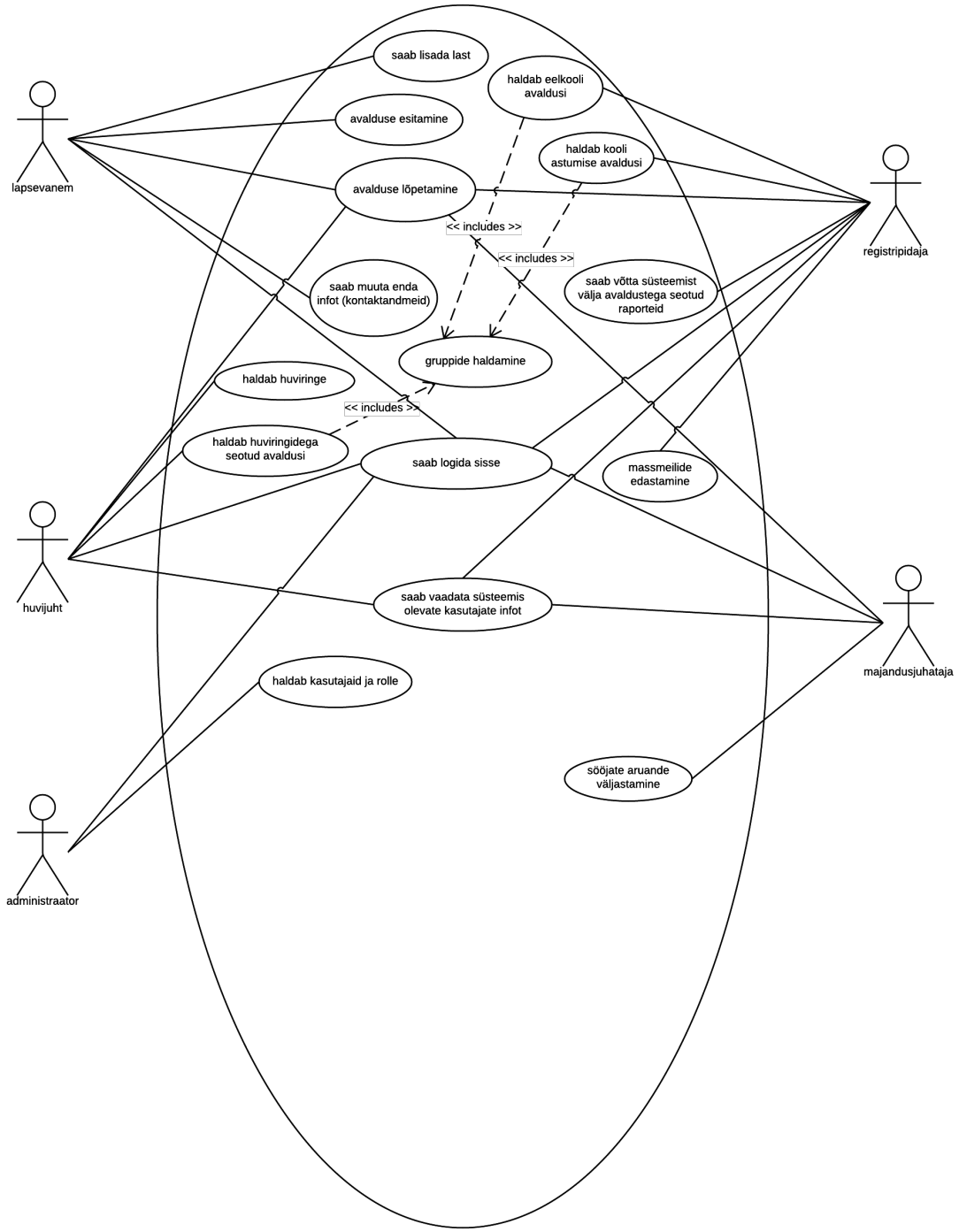
Lisa 21: kasutajate otsing



Lisa 22: Rakenduse andmebaasi struktuur



Lisa 23: Kasutajad ja tegevused



Lisa 24: Kooli sisseastumise taotluse protsess