Tartu Ülikool
Matemaatika-informaatika teaduskond
Arvutiteaduse instituut

# Erinevate programmeerimise raamistike võrdlus lähtuvalt veebi-töölauast PlutoF

Bakalaureusetöö

Autor: Siim Halapuu
Juhendajad: Vambola Leping,
Kessy Abarenkov

Autor: ..... „.....“ juuni 2011

Juhendaja: ..... „.....“ juuni 2011

Juhendaja: ..... „.....“ juuni 2011

Professor: ..... „.....“ juuni 2011

Tartu 2011

# Sisukord

# Sissejuhatus

Paljud veebirakendused ja infosüsteemid sarnanevad üksteisele teatud omaduste poolest [Wuf]. Korduvalt samasuguse arhitektuuri ja funktsionaalsuse programmeerimise vältimiseks on erinevate programmeerimise keelte jaoks välja töötatud veebiraamistikud. Veebiraamistikes on realiseeritud osad, mida kasutatakse enamikes infosüsteemides, nagu kasutajate autentimine, seansside haldamine, andmebaasiga suhtlemine jms. Iga veebiraamistik on loodud erineva eesmärgiga ning sobib mõnede veebirakendute arendamiseks paremini kui teiste.

Antud töös teostatavad hindamised ja võrdlused põhinevad elurikkuse informaatika veebi-töölaua PlutoF[1] funktsionaalsusel ja vajadustel. PlutoF mis on mõeldud loodusteaduslike andmebaaside haldamiseks, sh. andmete veebipõhiseks sisestamiseks, vaatamiseks, täiendamiseks ja analüüsimiseks. Antud töö eesmärgiks on toetada PlutoF edasise arendamise tarbeks sobiva veebiraamistiku ja sellele raamistikule vastava programmeerimise keele valiku tegemist. Töö peab andma ülevaate eelnevalt valitud raamistikest: *Ruby on Rails*, *Python Django*, *PHP Zend*. Antud raamistikud on välja valitud, kuna nad on antud programmeerimiskeelte jaoks tehtud veebiraamistikest populaarseimad. Töö tulemuseks on veebiraamistike võrdlus teatud mõõtepunktide põhjal ning iga veebiraamistiku koondhinne lähtudes veebitöölauast PlutoF.

Veebiraamistike võrdlemisel kasutatakse peamiselt kahte allikat – vastavasisulised artiklid veebist ja autori poolt püstitatud ülesande järgi testrakenduse programmeerimise töö käik. Antud rakenduse programmeerimisel saadud info ja kogemus on üheks peamiseks mõõtetulemuste määrajaks.

1 PlutoF http://plutof.ut.ee/

# 1. Veebi-töölaud PlutoF

Veebi-töölaud PlutoF on elurikkuse keskne infosüsteem, mis on mõeldud loodusteaduslike andmebaaside haldamiseks, sh. andmete veebipõhiseks sisestamiseks, vaatamiseks, täiendamiseks ja analüüsimiseks. PlutoF sobib kasutamiseks biosüstemaatikutele, ökoloogidele, elurikkuse uurijatele, looduskaitsjatele jt. uurimisandmete haldamiseks alates välitöödest kuni molekulaarsete analüüsideni[2]. Hetkel on PlutoF'i platvormi oma andmebaaside koduks valinud Tartu Ülikooli zooloogilised, botaanilised ja mükoloogilised kogud, Eesti Maaülikooli zooloogilised ja mükoloogilised kogud, Tallinna Botaanikaaia kogud, mõned erakogude omanikud, seente molekulaarset määrajat arendav rahvusvaheline konsortsium UNITE jne. 2011. a aprilli alguses oli PlutoF'i veebitöölaual 308 kasutajat umbes 20 riigist ja hõlmati üle 30 eri andmebaasi [Ei].

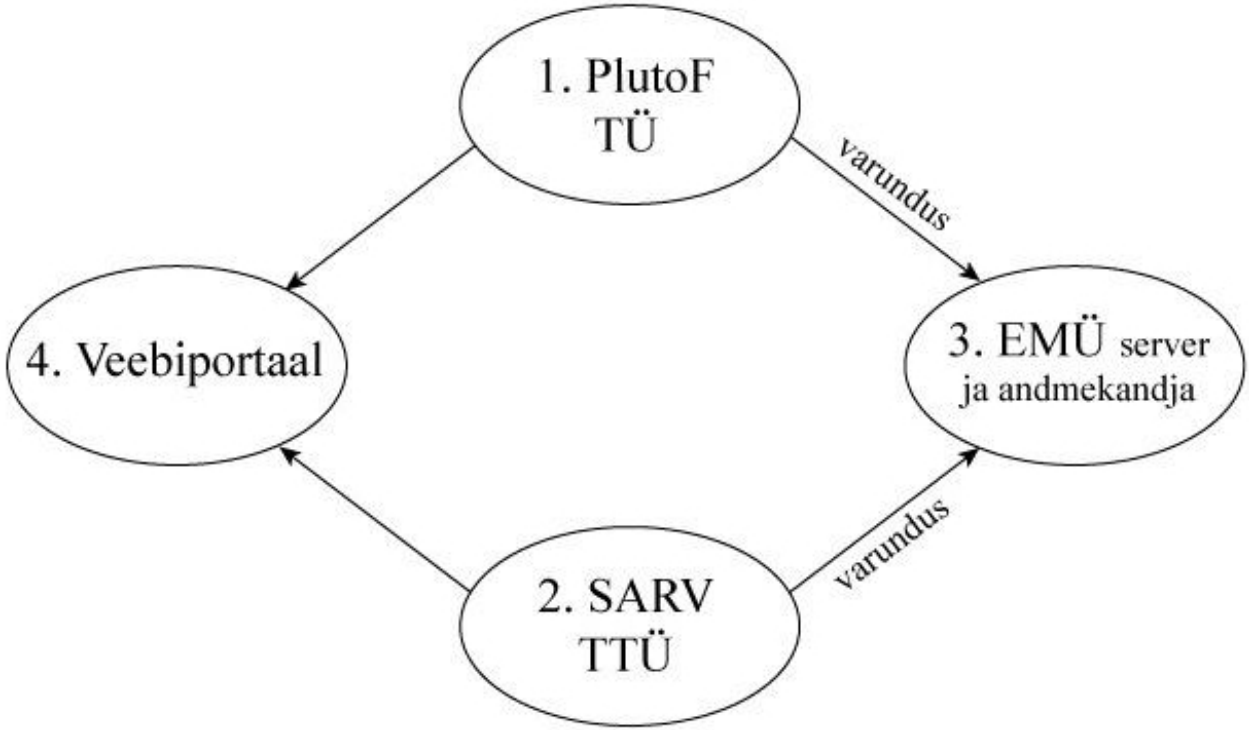
PlutoF pakub õhukesel kliendil (*thin client*) põhinevat pilveandmebaasi (*cloud database*) teenust. PlutoF'i platvormi moodustavad relatsioonilise andmebaasi tarkvara *MySQL*, mis jookseb *Red Hat Linux Apache* veebiserveris, ja veebitöölaud. *MySQL* andmemudel koosneb praegu enam kui 150 tabelist ning seda täiendatakse pidevalt. PlutoF-platvormi arendajate aluseks on olnud idee ühest andmebaasi mudelist, mis võimaldab talletada elurikkusega seotud ökoloogilist, geneetilist, taksonoomilist jm infot. Samas oli arendajate eesmärgiks anda kasutajatele võimalus luua piiramatu arv eri andmebaase, mille andmestik on samaaegselt hallatav ning analüüsitav.

Kasutajale on PlutoF-platvormi kõige silmapaistvam osa elurikkuse andmebaaside haldamiseks mõeldud veebi-töölaud. Viimane on programmeeritud kasutades järgmisi veebitehnoloogiaid: *HTML*, *CSS*, *JavaScript*, *AJAX*, *PHP* ja *Perl*. Veebitöölaua kaudu saab kasutaja samaaegselt hallata oma isiklikku, töörühma või projekti andmebaase, mille liige ta on. PlutoF on testitud kõigi tuntumate veebilehitsejatega ning paljudel erinevatel operatsioonisüsteemidel [Ebi]. Programmeerimisparadigmana on kasutatud protseduurilist programmeerimist.

2 PlutoF 1.0 kirjeldus http://natmuseum.ut.ee/396900

## 2. Miks on vaja antud tööd teha?

PlutoF arendamisega on algusest peale tegelenud peamiselt Kessy Abarenkov, keda on erinevate veebi-töölaua osade programmeerimisel aidanud mitmed inimesed. Veebi-töölaua aktiivne kasutajaskond on pidevalt toonud välja olulisi ja vähem olulisi osasid, mida võiks PlutoF'i juurde teha või täiendada. Lähema aja jooksul on plaanis PlutoF'i sisse viia mitmeid muudatusi sh. süsteemi mitmesse kohta salvestamine ja andmevahetus nende kohtade vahel. PlutoF on plaanis 2011 aasta suvel algava Eesti teaduse taristu teekaardi projekti „Loodusteaduslikud arhiivid ja andmevõrgustik“ raames ühildada TTÜ's loodud maateaduste keskse infosüsteemiga SARV. Kavas on PlutoF ja SARV süsteemide andmemudelid ühele standardsemale kujule viia ning mõlema süsteemi väljundiks luua ühtne veebiportaal. Rajatavat infosüsteemi kirjeldab järgnev *Joonis 1*.




**Joonis 1 Teekaardi projekti raames rajatava ühtse infosüsteemi skeem**

Seoses algava projektiga on plaanitud kaasata PlutoF'i arendamisesse lisatööjõudu. Selleks, et programmeerimisega saaks jätkata mitmeliikmeline meeskond ning edasine arendamine oleks kiirem on vajalik, et süsteem saaks üle viidud sobivale veebiraamistikule. Järgnevalt toon välja sobiva veebiraamistiku kasutamise peamised eelised [Wuf]:

- Säästab aega – veebiraamistikud sisaldavad tihti valmis tehtud ja testitud komponente

seansside haldamiseks, kasutajate autentimiseks, andmebaasiga suhtlemiseks, sõnede redigeerimiseks jpm. Programmeerija ei pea hakkama uue projekti juures taas kirjutama vahendeid ülaltoodud tegevuste jaoks.

- Organiseeritud rakenduse struktuur – raamistik määrab ära kataloogide struktuuri. Programmeerija ei pea hakkama mõtlema, kuhu mis failid panna, selle eest vastutab valitud raamistik.
- Turvaline kood – tänu pidevale arendamisele ja testimisele kasutavad raamistikud häid turvameetmeid.
- Raamistiku kogukonna tugi – tuntumatel raamistikel on tihtipeale väga lai ja aktiivne kogukond, kust saab väga kiiresti abi tekkinud probleemidele.

2010. aasta novembris valmis Torsten Erikssoni (Rootsi) uurimus [DIr], mille eesmärgiks oli leida Rootsi loodusteaduslike kogude digitaalse infosüsteemi DINA projekti jaoks lähtepunkt – juba olemas olev sarnane süsteem, mida saaks kasutada DINA arendamiseks. Uuringus vaadeldi mitemid sarnaseid infosüsteeme üle maailma ning leiti, et mõistlik oleks hakata kasutama Specify (USA) andmebaasi struktuuri koos selle kasutajaliidesega.
Uuringus oli välja toodud järgnevaid märkusi PlutoF kohta:

- Kuna *JavaScripti* ja *CSS* failid on *root*-kataloogides, ei ole *PHP* failidel järjekorda, aga hea oleks failid paigutada eraldi kataloogidesse vastavalt teemale.
- Kood on paremini kommenteeritud kui Specifyl, ning suurematest failidest saab paremini aru.
- Paljud olemas olevad skriptid kasutavad *PHP* ja *HTML* koodi koos andmebaasi baaskoodiga, mis võib tekitada probleemi juhul kui vahetatakse baaskoodi.
- *Thin-client* lähenemine on eeliseks tarkvara uuenduse puhul, samuti on kood suhteliselt lihtne, vähem hooldusvajadust.
- PlutoF peaks ümberkujundama MVC-struktuuri (*Model-View-Controller*) järgivaks, et tagada lihtsam ülalpidamine tulevikus.

# 3. Veebiraamistikud

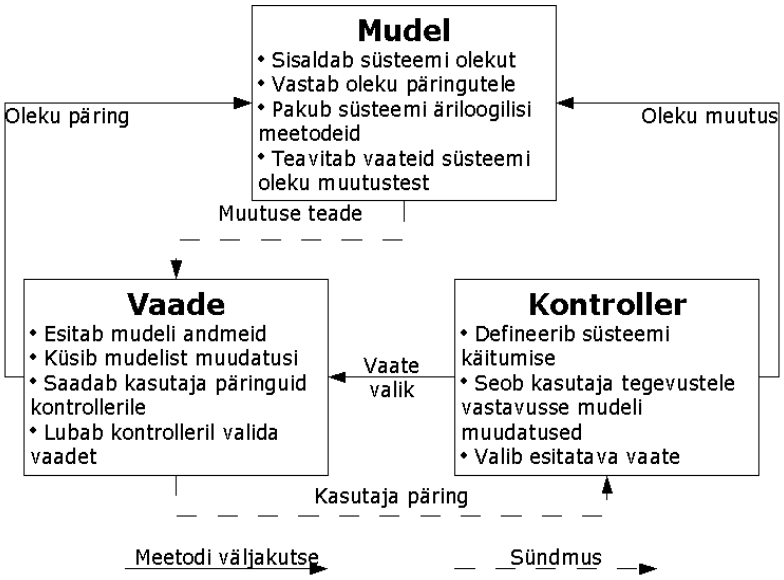
Veebiraamistikud on loodud toetamaks dünaamiliste veebilehtede, veebirakenduste ja -teenuste arendamist. Raamistike peamiseks eesmärgiks on mingite kindlate tegevuste lihtsustamine. Paljud raamistikud sisaldavad teeke andmebaasi ühenduse, kujunduse ja seansside haldamiseks ning tihti toetavad koodi (osade) korduvkasutamist.

Raamistikke kasutatakse eelkõige seetõttu, et väga paljud infosüsteemid sarnanevad üksteisele teatud omaduste poolest ning raamistikesse on sisse ehitatud standardsed lahendused kattuvate osadega [Vr].

Tänapäeva veebiraamistikud jagunevad üldise põhimõtte poolest kaheks[3]:

1. Liim-raamistikud (*glue frameworks*) – annavad arendajale hulga komponente, mida saab, aga ei pea, arenduse käigus kasutada.
2. "Kõik korraga" raamistikud (*full-stack frameworks*) – annavad arendajatele kõik vajaliku veebirakenduste arendamiseks ning sunnivad kogu süsteemi kasutama.

Arhitektuurilistest lahendustest sobib väga paljude projektide korral loogilisteks üksusteks jaotamine. Jaotamise aluseks on iga üksuse erinev funktsionaalsus. Kõige levinum üksusteks jaotamine on arhitektuurimustri MVC ehk *Mudel-Vaade-Kontroller* kasutamine. Antud muster kirjeldab kolmekihilist arhitektuuri:




**Joonis 2 MVC paradigmale vastava raamistiku osade kirjeldus[4]**

3 *MVC frameworks: stack vs glue, and how to pick the right one* http://blog.mixu.net/2010/01/14/mvc-frameworks-stack-vs-glue/

4 Mihkel Nõges. *Infosüsteemide arendamine J2EE platvormil. 2003.* http://lepo.it.da.ut.ee/~mike/mag/Mihkel_Nogese_magistritoo.doc

Enamik tänapäeval levinud veebiraamistikke kasutavad MVC mustrit. Raamistike erinevus tuleb aga sellest, kuidas on mustri erinevad osad realiseeritud. Kuigi fundamentaalsed erinevused raamistike vahel puuduvad, jaotatakse nad siiski *Vaate* (kasutajaliidese) koostamise tööpõhimõtte poolest kaheks[5]:

1. Päringupõhised (*request-based* või *push-based*) – liigutavad andmed kasutajaliidesesse, mis peab olema võimeline neid andmeid vastu võtma. Muudab raskeks *Vaate* koodi (osade) korduvkasutamise.
2. Komponentidest koosnevad (*component-based* või *pull-based*) – üles ehitatud komponentidest, mõeldud uuemate kasutajaliidese tehnoloogiate nagu *AJAX ja Rich Web Application* kasutamiseks.

## 3.1 Võrreldavad veebiraamistikud

Antud töö jaoks on välja valitud kolm raamistikku: *PHP Zend*, *Python Django* ja *Ruby on Rails*. *PHP* raamistik on valitud, kuna sellega on ehitatud praegu toimiv süsteem. Teised kaks raamistikku on valitud, kuna nemad on antud programmeerimise keelte seas enim levinud [HF]. Kõik kolm raamistiku kasutavad MVC arhitektuurimustrit.

## 3.2. Võrdlemise punktid ja hindamise süsteem

Kuna antud töö eesmärgiks on toetada veebiraamistiku valiku tegemist PlutoF jaoks, siis on vajalik määrata mõõtepunktid, mille järgi hakatakse erinevaid veebiraamistikke võrdlema. Iga raamistiku korral antakse hinnangud igale mõõtepunktile eraldi skaalas 0..1. Kokku on maksimaalne punktisumma 6 punkti.

Mõõtepunktide määramisel lähtuti PlutoF veebi-töölaua üldisest funktsionaalsusest ning nõudmisest, et töölaua edasine arendamine saaks toimuda võimalikult kiiresti. Üldiselt sobib antud mõõtepunktide järgi raamistike hindamine kõigi andmebaasist sõltuvate infosüsteemide jaoks sobiva raamistiku valiku tegemiseks.

Iga raamistiku juures vaadeldakse järgmisi punkte:

1. Dokumentatsioon – dokumentatsiooni täielikkus, arusaadavus ja ülesehitus. Mõõtepunkti eesmärk on määrata, kui abistav ja lihtsasti kasutatavalt on dokumentatsioon. Koodinäidete olemas olu dokumentatsioonis.
2. Õppimise kurv – kui palju läheb arendajal konkreetse raamistiku kasutamiseks aega enne,

5 Guy Rutenberg. *Pull vs. Push MVC Architecture*. 2008 http://www.guyrutenberg.com/2008/04/26/pull-vs-push-mvc-architecture/

kui ta suudab sellega produktiivset arendada (keeruka funktsionaalsuse lisamine ning raamistiku talitusest täielik aru saamine).

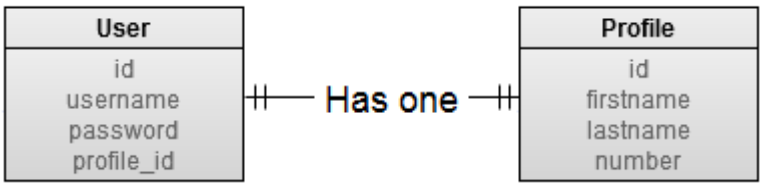
3. Veebist abi kättesaadavus – mõõtepunkti eesmärk on määrata, kui palju on raamistikuga seotud teemasid foorumites ning kui aktiivne ja suur on antud raamistiku kasutav kogukonna.
4. Andmebaasiga suhtlus – mõõtepunkti eesmärk on määrata, kui mugavaks ja lihtsaks on tehtud andmebaasiga suhtlemine ning kuidas toimub andmebaasi vastendus (*mapping*) MVC arhitektuuri *Mudeli* osas. Andmebaasiga suhtlus on antud olukorras tähtis punkt, kuna kogu süsteem on loodud toetama keerukat mitmesaja tabeliga andmebaasi.
5. Arendamise kiirus – mõõtepunkti eesmärgiks on määrata, kui palju on antud raamistikku sisse ehitatud vajalikke tööriistu ning kui mugavaks on tehtud koodi korduvkasutamine.
6. Koodimise lihtsus – vajaminevate funktsioonide olemas olu/üles leidmine. Mõõtepunkti eesmärk on määrata, kui jäik on antud raamistikus programmeerimine ning kui palju on erinevaid piiravaid reegleid.

## 3.3. Testrakendus

Iga võrreldava veebiraamistikuga programmeeritakse samasisuline ja sama funktsionaalsusega testrakendus. Antud rakenduse arendamise eesmärk on tutvuda kõigi raamistikega: nende süntaksi, dokumentatsiooni ja ülesehitusega.

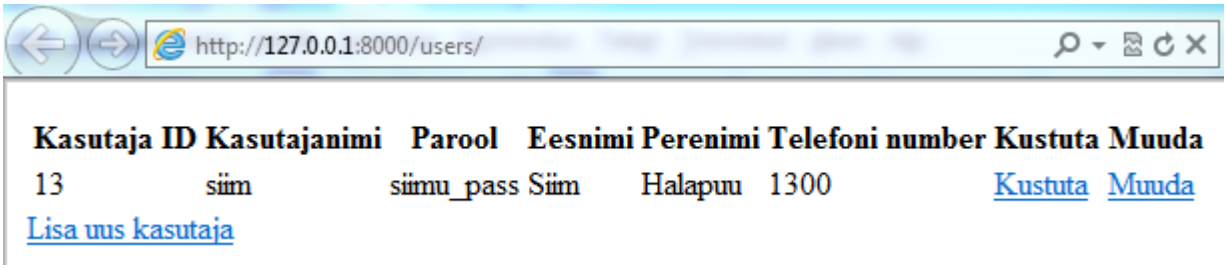
Testrakendus ülesanne on *MySQL* andmebaasi tabelitest sisu kuvamine *HTML* lehele, uute väljade lisamine ning olemasolevate redigeerimine kasutades *HTML* vorme ning tabeliridade kustutamine.

Rakenduse andmemudelit iseloomustab *Joonis 3*.




**Joonis 3 Testrakenduse andmemudel.**

Testrakendus on loodud olema igas programmeerimise keeles sama funktsionaalsuse ja välimusega.




**Joonis 4 Testrakenduse avalehekuva.**

### 3.4. Võrdlemise meetod

1. Dokumentatsioon – raamistiku dokumentatsiooni hindamiseks uuritakse esmalt otsingumootorite abiga (näiteks Google), kuidas hindavad ning on kommenteerinud erinevad raamistikku kasutanud inimesed selle dokumentatsiooni ning häid ja halbu külgi. Lisaks vaadatakse, kuidas on dokumentatsiooni üldine ülesehitus ning kui keeruliselt on erinevaid osasid seletatud. Antud meetod annab kokku 1 punkti.

2. Õppimise kurv – raamistiku õppimise kurvi hindamiseks otsitakse veebist 10 võimalikult erineva taustaga raamistikku kasutanud inimest, kes on kommenteerinud ja hinnanud antud raamistiku õppimise kurvi. Vastavalt raamistiku kasutajate taustale liigitatakse nad kolme rühma – professionaalid, keskmise tasemega ja algajad. Arendajate liigitamine eelnevate oskuste baasil on vajalik, kuna õppimise kurv oleneb väga suures osas eelnevatest teadmistest ja kogemusest teiste veebiraamistikega ja sarnaste tehnoloogiate tundmisega.

   Üheks õppimise kurvi hinde osaks on autori hinnang raamistiku õppimise kurvile lähtudes testrakenduse loomisest. Autori taust on järgnev – ülikooli õpingute raames on autor ühe kursuse jooksul varasemalt kasutanud *PHP Zend* raamistikku. Lisaks sellele on programmeeritud veel *PHP* raamistikuga *Yii*. Autor on kasutanud järgnevaid keeli: *Java*, *Perl*, *MySQL*, *ActionScript 3.0*, *PHP*. Autor liigitab ennast kesktasemega programmeerijaks.

   Kumbki nendest kahest meetodist – veebi uuringust ja autori hinnang – annab 0,5 punkti kogu hindest.

3. Veebist abi kättesaadavus – veebist abi kättesaadavuse hindamiseks võetakse arvesse inimeste hulka, kes on ennast pühendanud antud raamistiku arendamisele (*contributors*). Lisaks leitakse, kui palju on erinevaid küsimusi foorumis Stackoverflow (http://www.stackoverflow.com) ning kui paljud inimesed kuulavad (*follow*) antud

teemasid (*tag*). Teemad on *zend-framework*, *django*, ja *ruby-on-rails*. Teemade kuulajad on isikud, kes on märkinud ennast sellest teemast huvitatuks ning nemad saavad igapäevaselt teavitusi uuematest küsimustest ja küsimuste vastustest antud teemal. Teemade kuulajad on tihtipeale need, kes ka vastavad erinevatele küsimustele – aitavad teisi ja avaldavad oma arvamust. Viimaks tuuakse välja raamistiku Google trendid (http://www.google.com/trends).

Google trendide täpsema tulemuse saamiseks kasutatakse märksõnu, mis on üldisest teemast kitsendatud. Näiteks "django -jazz" – *Django* otsingu tulemuse vastete arv, kuid otsingu tulemusest on maha arvatud kõik tulemused, mis sisaldavad sõna *jazz*, kuna *Django* on lisaks raamistiku nimele ka muusiku nimi.

Kõik neli meetodit – pühendunud inimeste arv (raamistiku arendajad), teemasid foorumis Stackoverflow, teemade kuulajate arv, Google trendid tulemus – annavad igaüks 0,25 punkti.

4. Andmebaasiga suhtlus – andmebaasi suhtluse hindamiseks võetakse arvesse üldist *Mudeli* osa realiseeringut ning kui kerge/keeruline on moodustada keerukaid päringuid. Keerulise päringute tegemise lihtsus on oluline, kuna tihtipeale tuleb andmeid otsida läbi mitme tabeli ning otsingu tulemusi vastavalt olukorrale lisaks töödelda. Antud meetod annab kokku 1 punkti.

5. Arendamise kiirus – arendamise kiirust raamistikuga mõjutavad otseselt ja kaudselt väga paljud tegurid, mida kõiki pole võimalik mõõta. Kuna veebi-töölaud PlutoF on mõeldud peamiselt andmete sisestamiseks, muutmiseks ja kustutamiseks, siis arendamise kiiruse juures on oluline, et andmebaasiga suhtlus oleks võimalikult lihtne ja mugav. Hinnatakse, kui suures osas raamistik toetab erinevate programmi osade (*Mudel*, *Vaade*, *Kontroller*) paigutust kaustades. Võetakse arvesse veebist abi kättesaadavuse hinne ning leitakse, kui suures osas on toetatud koodi korduvkasutust. Viimaks võetakse arvesse testrakenduse arendamise kiirust.

   Kõik viis meetodit – andmebaasiga suhtlus, kaustade struktuuri toetamine, veebist abi kättesaadavus, koodi korduvkasutuse tugi, testrakenduse loomise hinne – annavad igaüks 0,2 punkti.

6. Koodimise lihtsus – koodimise lihtsuse hindamisel võetakse arvesse kõiki eelnevalt läbi tehtud uurimusi ning arvestatakse test rakenduse hinnet koodimise lihtsusele. Lisaks uuritakse, kui paindlik on raamistik – kui mugav on vastavalt vajadusele raamistikku

seadistada.

Varasemalt hinnatud mõõtepunktide kogusumma annab 0,3 punkti, raamistiku paindlikkus annab 0,3 punkti ja testrakendus annab 0,4 punkti.

# 4. Raamistike võrdlus

## 4.1. PHP Zend 1.11.5

- Kõik komponendid on täiesti *PHP 5* ühilduvad ning objektorienteeritud
- *Kasuta-vajadusel* (*Use-at-will*) arhitektuur, kus komponendeid on vähesidusad (*loosely-coupled*) ja sõltuvad üksteisest minimaalselt. Kasutada võib ka raamistiku üksikuid komponente.
- Laiendatav MVC arhitektuur, mis toetab erinevaid välimusi ja *PHP*-põhiseid malle (*template*)
- Erinevate andmebaasisüsteemide tugi (sh MySQL, *MariaDB*, *Oracle*, *IBM DB2*, *Microsoft SQL Server SQLite* jpt)[6]
- Paindlik vahemälu kasutus, mis toetab serveri mällu või failisüsteem andmete kirjutamist.

*Zend* on *push-* ja *pull-based* MVC arhitektuuriga liim-raamistik. *Zend'i* peetakse üheks populaarsemaks *PHP* veebiraamistikuks [HF]. *Zend* raamistiku peasponsoriks ja -arendajaks on ettevõte Zend Technologies, kelle loojate poolt arendati välja ka PHP keel[7].

*Zend* raamistiku eesmärk on toetada turvaliste, usaldusväärsete ja modernsete web 2.0 rakenduste ja veebiteenuste arendamist. *Zend* toetab paljusid erinevaid *API'sid* (*Application programming interface*) peamistelt tootjatelt nagu Google, Amazon, Yahoo!, Flickr[8].

### 4.1.1. Dokumentatsioon

*Zend* raamistiku dokumentatsioon asub aadressil http://framework.zend.com/manual. Leidub nii põhjalikum *API* dokumentatsioon, mis on mõeldud arendajatele detailse info saamiseks, kui ka üldine õpetus, kirjutatud raamatu formaadis, mis on mõeldud arendajatele reaalsete näidete põhjal *Zend* raamistikust aru saamiseks. Raamistiku dokumentatsioonini jõuab otsingumootorite kasutamisel üpris tihti.

*Zend* raamistiku dokumentatsioon on kirjutatud osaliselt sügavuti, kuid teistes kohtades jällegi liiga pinnapealselt. Dokumentatsioonis on küll hulgaliselt koodinäiteid ja neile ka selgitusi, kuid koodinäited pole tihtipeale sidusad jättes arendajale palju lisatööd ja mõtlemist, kuidas ja kus

6 *Zend_Db_adapter* http://framework.zend.com/manual/en/zend.db.adapter.html
7 *Zend is the PHP Company* http://www.zend.com/en/company/
8 *About Zend Framework* http://framework.zend.com/about/overview

antud meetodit/klassi kasutada. Välja võib veel tuua dokumentatsiooni liiga vähese tükkideks jagamise – peatükid on tihtipeale väga pikad, sisaldavad liiga palju osasid. Dokumentatsiooni tegemisse peaks olema rohkem kaasatud algajaid arendajaid selleks, et nemad lisaksid oma näiteid õppimise protsesi käigus aitamaks uutel algajatel paremini aru saada. Autor hindab dokumentatsiooni hindega 0,6 skaalal 0-1.

### 4.1.2. Õppimise kurv

Veebis läbiviidud uuringu põhjal võib öelda, et *Zend* raamistikku õppimise kurvi peetakse pigem pikaks ja alguses aeglase tõusuga. Antud juhul ei esine erineva taustaga kasutajate seas eriarvamusi. Veebist saadud andmete põhjal on õppimise kurv 0,1 skaalal 0-0,5.

Autori arvates polnud testrakenduse tegemine *Zend* raamistikku kasutades eriti keeruline. Andmebaasiga suhtlus on raamistikus ebapiisav, kuid üldine ülesehitus on üpriski lihtne ja arusaadav. Selleks, et osata kasutada efektiivselt raamistiku enamikke komponente, kulub palju aega, kuna *Zend* raamistik on väga suur ja kompleksne. Testrakenduse tegemisel saadud kogemuse põhjal hindab autor õppimise kurvi 0,4 skaalal 0-0,5, kuna põhifunktsionaalsuse saab realiseerida üpriski lihtsalt.

Kokku on õppimise kurvi hinne *Zend* raamistikul 0,5 punkti 1-st.

### 4.1.3. Veebist abi kättesaadavus

*Zend* raamistiku arendamisega tegeleb 240 inimest (*31.03.2011*)[9]. Need on inimesed, kes teevad uuendusi ja lisavad koodi raamistiku repositooriumisse ning täiendavad raamistiku dokumentatsiooni. Pühendunud inimeste arvu hindeks on 0,2 punkti 0,25'st, kuna arendamisega tegeleb üpriski palju inimesi, kuid vähem kui teistel võrreldavatel veebiraamistikel.

*Zend* raamistiku kohta on foorumis Stackoverflow algatatud kokku 5773 teemat (*31.03.2011*). Viimase 24 tunni jooksul on antud raamistikuga seoses tõstatatud 19 uut küsimust, millest on sel ajavahemikul vastatud juba 14 teemale. Kasutatakse ka foorumit leheküljel *http://www.zfforums.com/*, seetõttu võib kogukonna aktiivsust pidada siiski väga heaks ning hindeks on 0,25 punkti 0,25-st.

Stackoverflow foorumis on teema "*zend-framework*" kuulajaid umbes 1500. Kogukonna aktiivsust näitab see, et postitustele vastatakse peaaegu alati kiiremini kui 20 tundi peale teema tõstatamist. Kuulajate arvu ja vastamise kiiruse punkti hindeks on 0,25 punkti 0,25-st.

9 *Contributors* http://framework.zend.com/community/contributors

Google trendide täpsema tulemuse saamiseks kasutas autor märksõnu "*zend -technologies -studio*" (*Lisa 1*). Antud märksõnad tähendavad, et kuvatakse märksõna "*Zend*" otsingute arvu ning otsingusõnas ei tohi esineda sõnu "*technologies*" ja "*studio*". Jooniselt võib lugeda, et võrreldes teiste raamistikega otsitakse *Zend* raamistikuga seonduvaid teemasid üpris palju, seega on Google trendide hindeks *Zend* raamistikul 0,24 punkti 0,25-st.

Veebist abi kättesaadavuse koondhindeks on 0,94 punkti 1-st, seega võib veebiraamistikku *Zend* pidada tugeva ja aktiivse kogukonnaga raamistikuks.

### 4.1.4. Andmebaasiga suhtlemine

*Zend* raamistiku *Mudeli* osa on minimaalne, kuid sellega saab ära teha kõik vajaliku. Raamistikus on keeruline ja ebamugav moodustada pikki ja keerukaid päringuid andmebaasist[10], valmis tehtud päringute kasutamine on aga mugav ja lihtne.

*Zend* raamistikus pole vaikimisi realiseeritud ORM'i (*Object-Relational Mapping'ut*). Seetõttu on andmemudeli kirjeldamine *Zend's* tülikas ja ebamugav (*Lisa 2*).

Andmebaasist kõigi objektide *Users* tabeli väljade ja nendele vastavate *Profiles* tabeli väljade muutujasse salvestamiseks tuleb *Kontrolleri* osas kirjutada järgmised read:

```
$modelUsers = new Application_Model_Users();
$users = $modelUsers->fetchAllUsers();
```

Andmebaasiga suhtlemise hindeks on 0,3 punkti 1-st, kuna päringute tegemine on tehtud liiga ebamugavaks ning otsene andmemudeli vastandamine puudub.

### 4.1.5. Arendamise kiirus

*Zend* raamistiku andmebaasi suhtluse hindeks on vastavalt eelmisele mõõtepunktile 0,06 punkti 0,2-st. Veebist abi kättesaadavuse hinne on vastavalt eelnevalt tehtud hindamisele 0,19 punkti 0,2-st.

*Zend* raamistik ei suru peale kindlat kaustade struktuuri, kuid kui üks struktuur on ära valitud, jääb siiski arendaja otsustada, kui suures osas ta kasutab MVC arhitektuuri. Raamistikus on mugav arendada, kui on esmalt valitud õige kausta arhitektuur ning arendajad on piisavalt kogenud, et määrata *Kontrollerisse*, *Mudelisse* ja *Vaatesse* koodi osad, mis nendesse kuuluvad. Kui MVC on arendaja jaoks uus arhitektuur, siis oleks mugavam, kui raamistik määraks rohkem

10 *Zend Framework and Stored Procedures* http://blog.acodingfool.com/2009/08/21/zend-framework-and-stored-procedures/

koodi paigutust. Vastupidiselt MVC algajale, on selline lähenemine palju mugavam just kogenud programmeerijatele. Kausta arhitektuuri hindeks on 0,15 punkti 0,2-st, kuna algajal on rohkem vaja mõelda, kuid kogenenumal on mugavam töötada.

*Zend* raamistiku juures on kasutusel DRY (*don't repeat yourself*) põhimõtted. Koodi korduvkasutus on realiseeritud siin nõnda, et kõik *Kontrolleri* meetodid, mida on vaja korduvkasutada, kirjutatakse klassi, mis laiendab *Zend_Controller_Action_Helper_Abstract* klassi. Esitluse kihi puhul kirjutatakse meetodid, mida on vaja korduvalt kasutada, *Zend_View_Helper_Abstract* laiendavasse klassi. Koodi korduvkasutuse hindeks on 0,2 punkti 0,2-st.

Testrakenduse arendamise kiirust hinnates võib öelda, et tänu aktiivsele kogukonnale saab veebist alustamise kohta piisavalt informatsiooni, ning antud tüüpi testrakenduse loomine käib üpriski kiiresti. Testrakenduse arendamise kiiruse hindeks on 0,2 punkti 0,2-st.

Arendamise kiiruse koondhindeks on 0,8 punkti 1-st, seega võib öelda, et arendamine toimub *Zend* raamistikuga kiiresti peamiselt, kuna kogukond on aktiivne ja abivalmis ning veebis leidub hulgaliselt erinevaid materjale, mis kiirendavad arendamise protsessi.

### 4.1.6. Koodimise lihtsus

Arvestades kõiki eelnevalt tehtud uurimusi, on nende põhjal koodimise lihtsuse hinne 0,19 punkti 0,3-st.

*Zend* raamistik on loodud olema võimalikult paindlik. Enamus komponente saab kasutada sõltumatult ning väljaspool raamistikku. Kõike on võimalik vastavalt oma soovile ja vajadusele seadistada, kuid kuna raamistik on tehtud nõnda paindlikuks, siis selle arvelt on kaotatud raamistiku lihtsuses[11]. Raamistikul on vähe tehtud vaikimisi seadistusi, mis teeb alustaja jaoks õppimise raskemaks. *Zend* raamistiku paindlikuse hinne on 0,25 punkti 0,3-st.

Autori arvates oli testrakenduse tegemisel koodimine piisavalt lihtne, kui välja arvata andmebaasi päringute koostamise protsess. Testrakenduse koodimise lihtsuse hinne on 0,3 punkti 0,4-st.

Koodimise lihtsuse koondhinne on 0,74 punkti 1-st, seega võib öelda, et üldlevinud programmiosade arendamine on *Zend* rakendusega pigem lihtne.

11 *Zend framework – the cost of flexibility is complexity* http://blog.fedecarg.com/2009/02/22/zend-framework-the-cost-of-flexibility-is-complexity/

### 4.1.7. Plussid ja miinused

Plussid:

- Raamistiku suur populaarsus ning lisamoodulite rohkus.
- Võimalik komponente kasutada väljaspool raamistikku.

Miinused:

- Valmimas on *Zend Framework 2.0*, milles tuuakse sisse hulgaliselt muudatusi. Avalikustatud pole veel projekti lõplike tähtaegasid, kuid *beta* versioon peaks valmima 2011 aasta jooksul. Negatiivne selle juures on, et antud uus versioon pole enam tagasi ühilduv[12]. Kuna pole teada uue raamistiku stabiilse versiooni avalikustamise tähtaega ning PlutoF arendamine uue raamistiku järgi algab 2011 suve alguses, siis võib tekkida probleeme hiljem uuele versioonile üle viimisega, tuleb teha lisatööd.

### 4.1.8. Kokkuvõte

*Zend* raamistiku koondhinne on 3,88 punkti 6-st.

Peamised põhjused, miks *Zend* raamistik antud uurimuse käigus nõnda vähe punkte sai on tema õppimise kurvi liigne sügavus, dokumentatsiooni keerukus ning enim vähendas raamistiku üldpunktisummat see, et pole realiseeritud mugavat *Mudeli* osa ning ORM'i.

Antud uurimuse käigus ei esinenud testrakenduse tegemisel olulisi probleeme, kuid PlutoF-i, väga suuresti andmebaasist sõltuva süsteemi, programmeerimisel *Zend* raamistikuga hakkaks arendamist aeglustama peamiselt *Mudeli* osa ning andmebaasiga suhtlemise puudulikkus.

12 State of Zend Framework 2.0 http://weierophinney.net/matthew/archives/241-State-of-Zend-Framework-2.0.html

## 4.2. Ruby on Rails 3.0.7

- Lihtsustab üldist arendamist kasutades *scaffolding*'t, millega luuakse automaatselt esmased *Mudelid*, *Vaated* ja *Kontrollerid* (iga objekti korral eraldi).
- Arendamiseks pole vaja eraldi serverit, saab kasutada arenduskeskkonnaga kaasas olevat *Rails'i* serverit.
- Sisse on ehitatud *JavaScripti* teekide kasutamine (*Prototype*, *Script.aculo.us*, *jQuery*) *AJAX'i* jaoks.
- Kasutatakse REST (*REpresentational State Transfer*) arhitektuuritüüpi, kus *URL'i* ja *HTTP* meetoditega määratakse ära päringu eesmärk.
- Rõhutatakse kahte põhimõtet – *CoC* (*Convention over Configuration*) ehk rohkem koodi kirjutamist ja vähem konfigureerimist ning *DRY* (*Don't Repeat Yourself*) ehk koodi korduvkasutus[13].

*Ruby on Rails* on *push-based* MVC arhitektuuriga raamistik. Raamistiku õigused kuuluvad selle loojale David Heinemeier Hansson'le. Peamine raamistiku arendaja on ettevõte 37signals.

### 4.2.1. Dokumentatsioon

*Ruby on Rails* raamistiku dokumentatsioon asub aadressil http://api.rubyonrails.org/. Dokumentatsioon on *Rails'i* alustajale väga kasulik – kõik vajalik on piisavalt ära seletatud. Dokumentatsiooni on lisatud ka näited meetodite kasutamisest, kuid seda mitte iga meetodi/klassi juures. *Rails'i* dokumentatsiooni ei saa kasutajad kommenteerida ja oma koodinäidetega varustada.

Lisaks ametlikule dokumentatsioonile leidub ka teisi. Üks hästi kirjutatud mitteametlik dokumentatsioon asub aadressil http://apidock.com/rails. Peale korralikult kirjutatud dokumentatsiooni on seal võimalus raamistiku kasutjatel lisada oma koodinäiteid ja kommentaare kõikidele klassidele, meetoditele ja moodulitele.

*Rails'i* ametlik dokumentatsioon koos ülalnimetatud mitteametlikuga moodustavad arendajale möödapääsmatu abivahendi. Testrakenduse tegemisel osutus eriti kasulikuks just nimetatud mitteametliku dokumentatsiooni kommentaaride osa. Dokumentatsiooni koondhinne on 1 skaalal 0-1.

13 *Principles like CoC and DRY making Ruby on Rails widely used...* http://ruby-on-rails-web-development.blogspot.com/2009/02/principles-like-coc-and-dry-making-ruby.html

### 4.2.2. Õppimise kurv

Veebist läbiviidud uuringu põhjal võib öelda, et *Ruby on Rails*'i õppimise kurvi iseloomustavad sõnad "pikk õppimise kurv, hilisem kiire arendus". Kasutajate arvamused erinesid vastavalt nende kogemusele – algajamad pidasid õppimise kurvi pigem pikaks ning professionaalid pidasid seda keskmise õppimise kurviga raamistikuks. Veebist saadud andmete põhjal on õppimise kurv 0,25 skaalal 0-0,5.

Hindamise meetodis kirjeldatud testrakenduse programmeerimise protsessi kohta võib öelda, et sellise rakenduse tegemine oli *Rails'i* kasutades väga lihtne ja mugav. Suure töö teeb ära *Rails'i* käsurea käsk *generate scaffold*, millega luuakse ühele *Mudelile* vastav *Kontroller* ja *Vaated* andmebaasis olevatele tabeli väljade sisu vaatamiseks, muutmiseks, kustutamiseks ja uue veeru lisamiseks.

*Rails'ga* ei ole keeruline teha rakendusi, mis on ülesehituselt standardsed. Kui aga hakata programmeerima rakendust, mille funktsionaalsus on keeruline/unikaalne, siis selle valmistamiseks on vaja palju õppida. Tasuks ära märkida ka seda, et kuna *Rails* on programmeerimiskeelele *Ruby* kirjutatud veebiraamistik, siis efektiivseks programmeerimiseks tuleb ka *Ruby't* vallata. Testrakenduse loomisel saadud kogemuse põhjal hindab autor õppimise kurvi 0,4 skaalal 0-0,5.

Kokku on õppimise kurvi hinne *Ruby on Rails* raamistikul 0,65 punkti 1-st.

### 4.2.3. Veebist abi kättesaadavus

*Ruby on Rails'i* arendamisega on aastal 2011 (kuni mai kuuni) tegelenud 242 inimest. Läbi aegade on kokku raamistiku arendaimsega seotud olnud 1871 inimest[14]. Pühendunud inimeste hindeks panen 0,25 punkti 0,25'st, kuna antud uurimuse lõikes on see suurim pühendunud inimeste arv, mis ühel veebiraamistikul on.

*Rails* raamistiku kohta on foorumis stackoverflow kokku 38 006 teemat (31.03.2011). Tasuks ära märkida ka, et kuna *Ruby on Rails* versioon 3.0 on eelnevatest mõneti erinev, siis eraldi on 7942 küsimust märgitud ka seotuks just selle versiooniga *Rails* raamistikust. Viimase 24 tunni jooksul on tõstatatud versiooni 3.0-ga seoses 53 uut küsimust, millest 30 küsimust oli juba ka vastatud. Seetõttu võib kogukonna aktiivsust pidada väga heaks ning hindeks on 0,25 punkti 0,25-st.

Stackoverflow foorumis on teema "*Ruby-on-Rails*" kuujalaid umbes 9700. Teema "*Ruby-on-Rails-3*" kuulajaid on aga umbes 1300. Enamik postitustest saavad vastatud 24 tunni jooksul.

14 *Rails Contributors – All time* http://contributors.rubyonrails.org/contributors

Kuulajate arvu ja vastamise kiiruse punkti hindeks on 0,25 punkti 0,25-st.

Google trendide tulemuse saamiseks kasutati märksõnu "*ruby rails*" (*Lisa 1*). Kuna lühem sõna "*rails*" on seotud liialt kõrvaliste teemadega. Jooniselt (*Lisa 1*) võib lugeda, et märksõnadega seotud ülemaailmne liiklus oli väga kõrge ajavahemikul 2005-2008, kuid viimastel aastatel on märksõnadega seotud liiklus vähenenud umbes kaks korda. Arvestades, et PlutoF arendamine raamistikku kasutades hakkaks toimuma tulevikus ja kestaks vähemalt paar aastat (tähtis on võimalikult jätkusuutlik raamistik), siis graafiku põhjal on *Ruby on Rails* raamistiku Google trendide hindeks 0,20 punkti 0,25-st.

Veebist abi kättesaadavuse koondhindeks on 0,95 punkti 1-st, seega võib veebiraamistikku *Ruby on Rails* pidada tugeva ja aktiivse kogukonnaga raamistikuks.

### 4.2.4. Andmebaasiga suhtlemine

*Ruby on Rails'i Mudeli* osa on kogu raamistiku juures üks tähtsamaid. *Mudel* on realiseeritud olema kogu rakenduse keskpunkt[15]. Kogu andmebaasi loogika (tabelite vahelised seosed, valideerimised jms) on siin kirjeldatud.

Keerukamate päringute kirjeldamine on *Rails's* mugav tänu andmebaasi tabelite vaheliste seoste lihtsale kirjeldamisele. *Ruby on Rails* kasutab ORM'i realiseerimiseks *Active Record't* (*Lisa 3*).

Andmebaasist kõigi objektide *Users* tabeli väljade ja nendele vastavate *Profiles* tabeli väljade muutujasse salvestamiseks tuleb *Kontrolleri* osas kirjutada järgmine rida:

```
@users = User.all
```

Andmebaasiga suhtlemise hindeks on *Ruby on Rails'l* 1 punkti 1-st.

### 4.2.5. Arendamise kiirus

*Ruby on Rails* raamistiku andmebaasiga suhtluse hindeks on vastavalt eelmisele mõõtepunktile 0,2 punkti 0,2-st. Veebist abi kättesaadavuse hinne on vastavalt 0,19 punkti 0,2-st.

*Rails'i* rakenduse kaustade struktuuri ja esmaste vajalike failide loomiseks on mõistlik kasutada käsurea käsku "*rails -d mysql test*", millega luuakse rakenduse *test* üldstruktuur, samas määratakse ära ka, et kasutatakse *MySQL* andmebaasi. Loodud kaustastruktuuriga on ka MVC arhitektuuri järgi failid kaustadesse jaotatud. Kausta arhitektuuri hindeks on 0,2 punkti 0,2-st.

15 *10 Ruby on Rails best practices* http://blogs.sitepoint.com/10-ruby-on-rails-best-practices/

*Ruby on Rails* raamistiku üks peamisi eesmärke on DRY põhimõtete järgimine[16]. Seega on koodi korduvkasutuse hindeks 0,2 punkti 0,2-st.

Testrakenduse arendamise kiirust hinnates võib öelda, et suur osa tööst sai tehtud *Rails'i* käsurea käskudega "*generate scaffold*" ja "*generate controller*". Juurde tuli lisada ainult üks *Mudeli* osa (tabel Profiles) ning ühendada programmi siseselt kaks andmebaasi tabelit. Testrakenduse arendamise kiiruse hindeks on 0,2 punkti 0,2-st.

Arendamise kiiruse koondhindeks on 0,99. Võib öelda, et rakenduse arendamine toimub *Ruby on Rails* raamistikku kasutades kiiresti, kuna raamistik suudab ise ära teha väga palju tööd ning raamistikul on aktiivne kogukond.

### 4.2.6. Koodimise lihtsus

Arvestades kõiki eelnevalt tehtud uurimusi, on koodimise lihtsuse hinne 0,28 punkti 0,5-st.

*Ruby on Rails* on loodud olema paindlik, kuid samas näitab raamistik algajale programmeerijale piisavalt ette, kuhu mis osad peavad minema – algajal on *Rails'ga* lihtne hakkama saada. Paindlikuse hinne on 0,3 punkti 0,3-st.

Testrakenduse tegemisel oli koodimine väga mugav ning lihtne, seega hinne on 0,4 punkti 0,4-st.

Koodimise lihtsuse koondhinne on 0,98 punkti 1-st, seega võib öelda, et standardsemate veebirakenduste programmeerimine on *Ruby on Rails'i* kasutades lihtne.

### 4.2.7. Plussid ja miinused

Plussid:

- *Rails'ga* saab mugavalt (vähese programmeerimisega) lahendada lihtsamaid ülesandeid.
- Andmebaasipõhine vormide tegemise süsteem.
- Lihtne *Vaadete* tegemine.
- Vaikimisi sisaldab erinevaid *JavaScript'i* teeke.

Miinused:

- *Ruby on Rails* versioon 3.0, mis avaldati 2010. aasta augustis, ei ühildu mõnes osas varasemate versioonidega[17]. See võib kaasa tuua ebameeldivusi ja probleeme vigade

16 *Ruby on Rails development* http://www.jimcode.org/development-services/ruby-on-rails-development/
17 *Compatibility of Rails 2.3.8 with 3.0.1* http://stackoverflow.com/questions/4990402/compatibility-of-rails-2-3-8-with-3-0-1/4990419#4990419

otsimisel, eriti kui on programmeeritud järgides mõne varasema versiooni õpetust.

- Konfiguratsiooni failide muutmine on keeruline.
- Keerukamate ülesannete lahendamisel on esmane õppimise kurv pikk ja lauge.

### 4.2.8. Kokkuvõte

*Ruby on Rails'i* koondhinne on 5,57 punkti 6-st.

*Rails* on mõeldud tegema arendaja eest võimalikult palju tööd ära ning seda varjatult, et programmerija jaoks toimuks kõik väga lihtsalt [RvD]. Raskeks teeb selline lähenemine arendamise alles siis, kui hakatakse programmeerima keerulisi lahendusi.

Rakendusele esmase funktsionaalsuse lisamine on lihtne. Enamlevinud funktsionaalsuse lisamine pole samuti raske, kuid kui hakata tegema miskit keerulisemat, siis muutub töö *Ruby* spetsiifiliseks ning tuleb väga hästi aru saada raamistiku ülesehitusest. Sellest tuleneb ka see, et lihtsa rakenduse tegemisel on õppimise kurv minimaalne, kuid kui lihtne osa on valmis, siis muutub kurv laugemaks ja pikemaks.

*Rails'ga* on lihtne luua infosüsteeme, mis tegelevad peamiselt andmebaasi andmete salvestamisega ja nende muutmisega.

## 4.3. Python Django 1.3

- Automaatne administreerimise paneeli loomine *Mudeli* baasil.
- Sisseehitatud arenduse server.
- Andmebaasipõhiste rakenduste jaoks sobilike vormide loomine *Mudeli* põhjal.
- Iga *Django* projekt/programm koosneb alamrakendustest, mis kõik omavad erinevat funktsionaalsust ning mida saab sisse ja välja lülitada settings.py failis.
- Mugav interaktiivne käsurea keskkond rakenduse katsetamiseks ja *Django* õppimiseks.
- Järgitakse DRY ehk koodi korduvkasutamise põhimõtteid.

*Python'i* raamistik *Django* on *push-based* MVC arhitektuuriga. *Django* on populaarseim *Pythoni* veebiraamistik.

### 4.3.1. Dokumentatsioon

*Django* dokumentatsioon asub aadressil http://docs.djangoproject.com/en/1.3/. Dokumentatsioon on hästi üles ehitatud – toodud on piisavalt näiteid ning viiteid dokumentatsiooni teiste osade juurde, mis on antud teemaga seotud.

Raamistiku dokumentatsiooni pole võimalik kasutajatel kommenteerida, kuid see ei näi olevat probleem, kuna erinevad osad on küllaltki hästi kommenteeritud ja piisavate näidetega varustatud esialgsete dokumentatsiooni kirjutajate poolt.

*Django* dokumentatsiooni hindaksin 0,9 punktiga skaalal 0-1.

### 4.3.2. Õppimise kurv

Veebis läbi viidud uuringu põhjal võib öelda, et *Pythoni* raamistiku *Django* peetakse kiiresti õpitavaks veebiraamistikuks. Antud juhul leidus kümnest õppimise kurvi kommenteerinud programmeerijast ainult üks programmeerija, kes hindas *Django* õppimise kurvi pikaks[18].

Veebist saadud andmete põhjal on *Django* õppimise kurv 0,45 punkti 0,5-st.

Autori hinnangul oli *Django'ga* testrakenduse programmeerimine pigem lihtne ning vähe aega nõudev protsess. Algaja *Django* kasutaja jaoks on suuresti abi raamistiku ametlikust dokumentatsioonist. Samas leiab väga kiiresti probleemidele lahendusi ja selgitusi

18 *Django learning curve is "Z-shaped"* http://news.ycombinator.com/item?id=459303.

otsingumootorite abil. Testrakenduse baasil hindab autor *Django* õppimise kurvi hindega 0,5.

Kokku on õppimise kurvi hindeks 0,95 punkti 1-st.

### 4.3.3. Veebist abi kättesaadavus

*Django* raamistikku arendavate inimeste ametlikku nimekirja veebis ei leidu. Veebilehel http://ohloh.com on ära toodud 47 inimese nimed, kes on pühendunud raamistiku arendamisele. Pühendunud inimeste hindeks panen 0,2 punkti 0,25-st.

Foorumis Stackoverflow on *Django* raamistiku kohta tõstatatud kokku 17828 teemat. Viimase 24 tunni jooksul on tõstatatud 42 uut teemat, millest vastuse on saanud 32 teemat. Teema "Django" kuulajaid on kokku umbes 6500 registreerinud kasutajat. Stackoverflow foorumiga seotud punktid on 0,5 punkti 0,5-st.

Google trendide täpsema tulemuse saamiseks kasutati märksõnu "*django -jazz*"(*Lisa 1*). Sõna "*jazz*" on otsingutulemusest välja jäetud, kuna raamistikule pandi nimi kuulsa džässmuusiku järgi. Jooniselt (*Lisa 1*) võib lugeda, et *Django* on töös nimetatud raamistikest enim otsingutes esinev. Seega antud kriteeriumi hindeks on 0,25 punkti 0,25-st.

Veebist abi kättesaadavuse koondhindeks on 0,95 punkti 1-st.

### 4.3.4. Andmebaasiga suhtlemine

*Django* raamistik kasutab täielikult *Object-Relational Mapping'ut* (ORM) – kõik andmebaasi tabelid vastandatakse vaikimisi eraldi klassidega. Välisvõtmete ja erinevat tüüpi seostega (üks-mitmele, üks-ühele jne.) on võimalik *Mudeli* siseselt siduda andmebaasi tabeleid. Päringute tegemine on võimalik väga mugavalt korraldada *Mudelis* ilma *SQL'i* kasutamata (*Lisa 4*), kuid vajadusel on ka see võimalik.

Andmebaasist kõigi objektide Users tabeli väljade ja nendele vastavate Profiles tabeli väljade salvestamiseks muutujasse võtmiseks tuleb *Kontrolleri* osas kirjutada järgmine rida:

```
users = Users.objects.all()
```

Andmebaasiga suhtluse hindeks on 1 punkt 1-st.

### 4.3.5. Arendamise kiirus

*Python Django* andmebaasi suhtluse hindeks on vastavalt eelmisele mõõtepunktile 0,2 punkti 0,2-st. Veebist abi kättesaadavuse hinne on vastavalt eelnevalt tehtud hindamisele 0,19 punkti

0,2-st.

*Django* raamistiku üldine kaustade paigutuse struktuur ei ole rakenduse loomisel paika pandud. *Django* rõhutab küll MVC kasutamist, kuid kaustastruktuur tuleks arendajal endal korralikult seadistada. Kausta struktuuri hindeks on 0,15 punkti 0,2-st.

*Django* järgib koodi korduvkasutuse (*DRY*) põhimõtteid[19], seega on antud punkti hinne 0,2 punkti 0,2-st.

Testrakenduse kirjutamisel kulus esmalt aega *Python'i* süntaksist aru saamisel, kuid kui *Python'ga* on varem kokkupuudet olnud ning MVC ja ORM põhimõtted on tuttavad, siis on testrakenduse laadse programmi arendamine lihtne. Antud mõõtepunkti hinne on 0,2 punkti 0,2-st.

Arendamise kiiruse koondhinne on 0,94 punkti 1-st.

### 4.3.6. Koodimise lihtsus

Arvestades kõiki eelnevalt tehtud uurimusi, on koodimise lihtsuse hinne 0,28 punkti 0,3-st.

*Django'le* on loodud hulgaliselt erinevaid mooduleid, mida saab projektides lihtsasti kasutada. Lisaks sellele võimaldatakse kasutada erinevaid andmebaasisüsteeme (relatsioonilised kui ka mitterelatsioonilised). Võimaldatud on ka erinevate MVC osade väljavahetamine teiste samalaadsete süsteemidega (*Vaadete* mallisüsteemi väljavahetamine jms)[20]. Vaikimisi seadete muutmine on küll mõningal määral raske, kuid soovi korral võimalik. Seega võib öelda, et *Django* on paindlik veebiraamistik. Kuna aga algajal on *Django'ga* vaikesätteid muutmata lihtne programmeerida, siis on raamistiku paindlikuse hinne 0,3 punkti 0,3-st.

Testrakenduse tegemisel toimus *Python'i* süntaksi õppimine kiiresti ning üldine rakenduse funktsionaalsuse lisamine probleemivabalt. Kuna *Django* on täiesti tagasiühilduv, siis kõik veebist leitud soovitused ja koodinäited töötasid probleemideta. Testrakenduse koodimise lihtsuse hinne on 0.4 punkti 0.4-st.

Koodimise lihtsuse hinne on 0,98 punkti 1-st.

### 4.3.7. Plussid ja miinused

Plussid:

19 *Django pilosophies* http://docs.djangoproject.com/en/dev/misc/design-philosophies/.
20 *Django is not flexible* http://agiliq.com/blog/2010/10/django-is-not-flexible/

- Lihtne *Vaadete* kirjutamine.
- Mugav andmebaasipõhine vormide tegemise süsteem.
- Täielikult tagasiühilduv varasemate versioonidega.

Märkimisväärseid negatiivseid omadusi *Django* juures ei leitud.

### 4.3.8. Kokkuvõte

*Python Django* koondhinne on 5,72 punkti 6-st.

*Django* on loodud tegema programmeerija eest võimalikult palju tööd ära, kuid samas rõhutatakse/näidatakse arendajale, mil määral raamistik tema eest tööd teeb [RvD].

*Django* raamistikus rakenduse loomine on lihtne. Arendamisel tuleb küll paljusid asju seadistada, kuid selle võrra on arendajal rohkem võimalik teha valikuid vastavalt loodavale rakendusele. Lisaks on seadistamine tehtud arendaja jaoks kergesti õpitavaks.

Üldiselt on *Django* mugav raamistik suurte ja keeruliste rakenduste tegemiseks[21].

21 *Is Python good enough for big applicaitons* http://stackoverflow.com/questions/3700413/is-python-good-enough-for-big-applications.

## 4.4. Võrdlustulemused

*Tabelis 1* on toodud töös kasutatud veebiraamistike hindamispunktide tulemused ning ka koondhinded.

| | **Dokumentatsioon** | **Õppimise kurv** | **Veebist abi kättesaadavus** | **Andmebaasiga suhtlemine** | **Arendamise kiirus** | **Koodimise lihtsus** | **KOKKU** |
|---|---|---|---|---|---|---|---|
| ***PHP Zend*** | 0,6 | 0,5 | 0,94 | 0,3 | 0,8 | 0,74 | **3,88** |
| ***Ruby on Rails*** | 1 | 0,65 | 0,95 | 1 | 0,99 | 0,98 | **5,57** |
| ***Python Django*** | 0,9 | 0,95 | 0,95 | 1 | 0,94 | 0,98 | **5,72** |

*Tabel 1: Võrdlustulemused*

Võrdlustulemuste tabelist selgub, et antud võrdlusmeetodeid kasutades sai kõige rohkem punkte *Python'i* raamistik *Django* – 5,72. *Ruby on Rails* sai 0,15 punkti vähem ning kõige vähem punkte sai *PHP* raamistik *Zend* – 3,88.

*Zend'i* peamisteks puudujääkideks antud võrdlusmeetodi juures olid tema pikk õppimise kurv ning väga puudulik andmebaasiga suhtlemise osa – *Zend'l* puudub *Mudeli* realisatsioon ning ORM. Seetõttu pole võimalik andmebaasi tabeleid lihtsalt programmi klassidele vastandada ning programmisisene andmemudeli kirjeldamine on keeruline ning tülikas. Seoses kehva andmebaasiga suhtlemise realisatsiooniga sai ka raamistikus tehtud testrakendus üldiselt madalamaid hindeid, kui teistel raamistikel.

*Ruby on Rails* ja *Python Django* vaheline punktide erinevus ei ole antud võrdlemise meetodi juures märkimisväärne. Peamine punktivahe tuli just õppimise kurvi hindamise punktist. *Ruby on Rails'i* õppimise kurvi peamiseks eeliseks sai see, et suur osa tööd tehakse programmeerija eest ära ilma, et programmeerija peaks palju koodi kirjutama. Djagno puhul ei tee raamistik programmeerija selja taga n.-ö. tööd ära. Selle võrra muutub programmi kood pikemaks, aga ka ennast selgitavamaks ning muudatuste tegemine ja keerulisema funktsionaalsuse lisamine lihtsamaks.

Võrdlustulemused on antud autorist sõltuvad, teiste hindaja tulemused antud mõõtepunktidele võivad erineda.

# Kokkuvõte

PlutoF'i veebi-töölaud on pidevalt uuenev ja täienev, millele juures on oluline, et täienduste ja uute osade lisamine oleks võimalikult lihtne ja kiire protsess. Pidades silmas PlutoF'i veebi-töölaua vajadusi sai antud töö käigus hinnatud ja hiljem võrreldud kolme veebiprogrammeerimise raamistikku – *Python Django*, *Ruby on Rails* ja *PHP Zend*.

Raamistike hindamisel kasutati peamiselt kahte allikat - vastavasisulised artiklid veebist ja autori poolt püstitatud ülesande järgi testrakenduse programmeerimine. Testrakendus oli lihtne *MySQL* andmebaasist kahe välisvõtmega seotud tabeli veergude kuvamise, lisamise, muutmise ja kustutamise veebisüsteem.

Antud hindamismeetodit kasutades ja võrreldavate raamistike üldist tausta uurides selgus, et PlutoF edasiseks arendamiseks sobib kõige paremini *Python'i* veebiraamistik *Django*. Raamistikul on olemas kõik vajalik, et PlutoF arendamine toimuks kiiresti ja mugavalt.

*Django* raamistiku dokumentatsioon on kirjutatud arusaadavalt ja piisavalt kommenteeritult – dokumentatsioonist on abi nii algajal kui ka professionaalil. *Django* õppimise kurv on võrreldavatest raamistikest kõige lühem ning raamistikul on väga aktiivne kogukond – veebist abi leidmine ei ole keeruline. *Django* kasutab *Object-Relational Mapping'ut*, mis lihtsustab andmebaasiga suhtlust ning keeruliste päringute tegemist. Samuti on *Django'ga* rakenduste arendamine kiire ning lihtne.

Võrdluse tulemustest järeldus, et kõige ebasobivam raamistik PlutoF'i arendamiseks on *PHP Zend*. Silmas tuleks pidada, et lähtutud on veebi-töölaua PlutoF funktsionaalsusest. Kõik vaadeldavad raamistikud on kasutatavate programmeerimise keelte seas populaarseimad. Kõigi arendamisel on lähtutud rangetest nõuetest raamistikule ning iga raamistik on välja töötatud professionaalide poolt. Kõik raamistikud võimaldavad arendada veebi-töölaua PlutoF, kuid lähtudes töös kasutatud hindamise meetodist on sobivaim raamistik *Python Django*.

# Summary


**Bachelor's thesis**

**Siim Halapuu**

**The comparisson of different web application frameworks according to web based workbench PlutoF**

PlutoF is a fast developing web based workbench, which needs that adding new functionality would be a fast and simple process. In this paper the author evaluates and compares three web application frameworks - *Pyton Django*, *Ruby on Rails* and *PHP Zend* - keeping in mind the needs of the web based workbench.

For comparing the frameworks there were used mainly two sources - web articles and a test application which was written for every framework. In the test application a person could see data from two *MySQL* database tables (which were connected via foreign key). Person could also modify or delete existing data and add new if needed.

Using the evaluation method and researching the background of the frameworks, it came out that the most suitable web application framework for developing PlutoF is *Python Django*. The framework has everything necessary for enabling fast and simple development of PlutoF.

*Django's* documentation is well-written and sufficiently commented for beginner and also professional framework user. *Django* has a smaller learning curve than the other two frameworks and also it has a very active community willing to help out every beginner and professional. *Djagno* uses Object-Relational Mapping, which eases the database communication and simplifies making complex queries. All in all, development of a web application is simple and fast using *Python's* web framework *Django*. According to previous, *Django* got the most points in the evaluation process compared to *PHP Zend* and *Ruby on Rails*.

According to the overall comparison of the frameworks, the most unsuitable framework for developing PlutoF is *PHP Zend*. It should be kept in mind that the comparison and evaluation was based on PlutoF's functionality. All of the mentioned frameworks are the most popular among their programming languages. All of them have been developed by professionals according to strict requirements to programming frameworks. It is possible to develop PlutoF

with all of the frameworks but according to the evaluation points used in this paper, *Python Django* is the most suitable.

# Kasutatud kirjandus


[Wuf] Tengku Zahasman, *Why you should use a web application framework.* 2007. http://web2.0entrepreneur.com/7/why-you-should-use-a-web-application-framework.html – viimati vaadatud 25. mai 2011.

[Ei] U. Kõljalg, K. Abarenkov. *Elurikkuse informaatika*. 2011

[Ebi] Abarenkov, K,, Tedersoo, L,, Nilsson, RH, Vellak, K, Saar, I, Veldre, V, Parmasto, E, Prous, M, Aan, A, Ots, M, Kurina, O, Ostonen, I, Jõgeva, J, Halapuu, S, Põldmaa, K, Toots, M, Truu, J, Larsson, K.-H, Kõljalg, U. 2010. PlutoF – a web based workbench for ecological and taxonomic research, with an online implementation for fungal ITS sequences. Evolutionary Bioinformatics 6: 189–196.

[DIr] Torsten Eriksson. *Gemstamt system för samlingsdatabaser Rapport.* 2010.

[Vr] Raivo Laanemets, *Veebirakenduste raamistikud: Struts, Spring ja JSF.* 2005. http://courses.cs.ut.ee/2005/tvt/uploads/Main/RaivoLaanemets_raamistikud.pdf – viimati vaadatud 25. mai 2011.

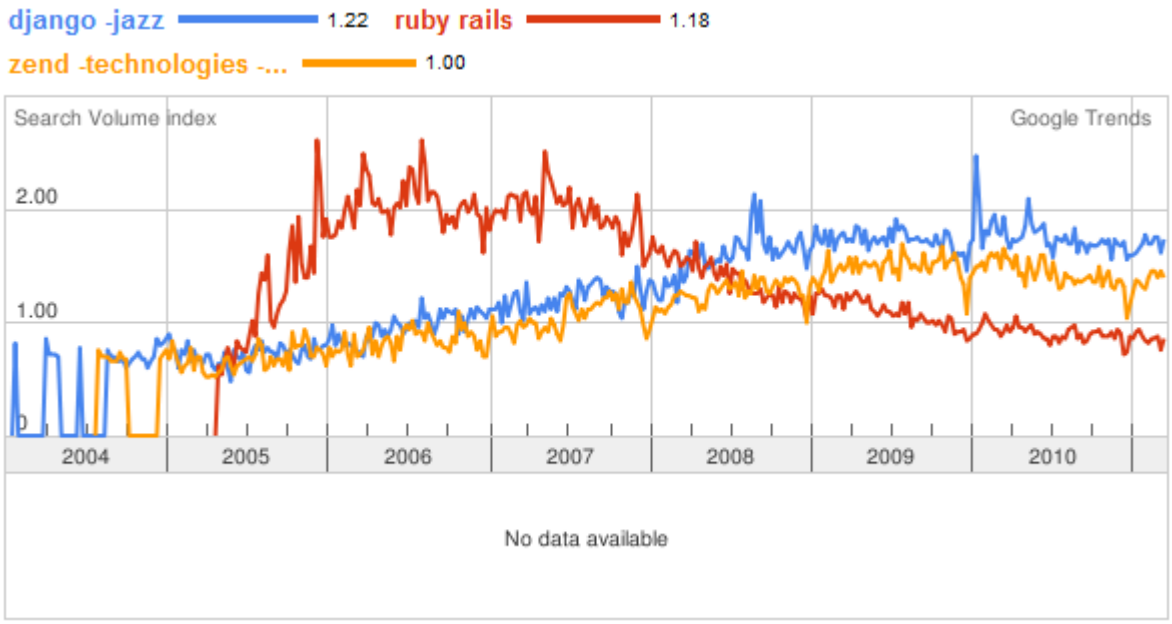
[HF] HotFrameworks. http://hotframeworks.com/rankings – viimati vaadatud 25. mai 2011.

[RG] Getting Started with Rails http://guides.rubyonrails.org/getting_started.html – viimati vaadatud 25. mai 2011.

[RvD] Rails vs Django: A Developer's Comparison. 2010. http://www.ctctlabs.com/index.php/blog/detail/rails_vs_django/ - viimati vaadatud 25. mai 2011.

# Lisad

## *Lisa 1.*




Google trends graafik, mis kujutab keskmist ülemaailmset liiklust antud raamistikel läbi aastate.

## *Lisa 2.*

Zend raamistikuga "Users" tabeli ühendamine "Profiles" tabeliga ühe kindla rea pärimiseks.

```
public function fetchUser($id) {
      $select = $this->select();
      $select->setIntegrityCheck(false);
            $select->from('users', array(
                              'username',
                              'password'))
                        ->join('profiles', 'users.profile_id = profiles.id',
array(
                              'firstname' => 'profiles.firstname',
                              'lastname' => 'profiles.lastname',
                              'number' => 'profiles.number'))
                  ->where('users.id = ?', $id);
      return $this->fetchRow($select);
}
```

## *Lisa 3.*

Ruby on Rails raamistikus realiseeritud andmebaasi tabeli Users vastanduv klass.

```
class User < ActiveRecord::Base
      validates :username,  :presence => true # välja username
valideerimine
      validates :password, :presence => true, # välja parool valideerimine
                           :length => { :minimum => 6 } # jätkub
      has_one :profile, :class_name => "Profile", :foreign_key =>
:profile_id, :dependent => :destroy # Andmebaasi tabeliga "Profiles"
sidumine välisvõtme kaudu. Kustutamisel kustutatakse ka vastav Profiles
tabeli veerg.
      accepts_nested_attributes_for :profile, :reject_if => lambda { |a|
a[:firstname].blank? }, :allow_destroy => true
end # Ühes vormis kasutamiseks lisatakse nüüd ka profiili väljad. Lisaks
ka valideerimine.
```

## *Lisa 4*

Python'i Django raamistikus realiseeritud andmebaasi tabeli Users vastanduv klass.

```
class Users(models.Model):
      id = models.AutoField(primary_key=True)
      username = models.CharField(unique=True, max_length=255)
      password = models.CharField(max_length=255)
      profile = models.OneToOneField(Profiles)
      class Meta:
            db_table = u'users'
      def __unicode__(self):
            return self.username
```