TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut
Infosüsteemide õppetool

PHP raamistike võrdlus veebiarendusega tegeleva ettevõtte näitel

Bakalaureusetöö

Üliõpilane: Rene Viskar
Üliõpilaskood: 120409IABB
Juhendaja: Enn Õunapuu

Tallinn
2016

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

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

(*kuupäev*) (*allkiri*)

Annotatsioon

Käesolevas töös analüüsitakse erinevate PHP raamistike tugevaid ja nõrki külgi. Töö peamiseks eesmärgiks on ühe veebiarendusega tegeleva ettevõtte jaoks sobiva ja kaasaegse PHP raamistiku leidmine.

Lõputöö on jagatud kolmeks suuremaks osaks: ülevaade ning raamistikule pandavate nõuete valik, raamistike võrdlus ning saadud tulemuste analüüs.

Ülevaates kirjeldatakse töö tausta, analüüsitakse, miks on käesolev töö vajalik ning antakse ülevaade töö struktuurist ja metoodikast ning kirjeldatakse aspekte, mida kasutati raamistike võrdluseks.

Raamistike võrdluse peatükis, nagu nimigi viitab, toimub peamine raamistike võrdlus. Kirjeldatakse valitud raamistikke ning seejärel võrreldakse raamistike eri aspekte seejuures pannes põhirõhu leitud tugevustele või nõrkustele. Ühtlasi hinnatakse raamistiku igat vaatluse all olevat aspekti 10 punkti skaalal.

Tulemustes analüüsitakse eri aspektidele tuginedes raamistike erinevusi ning toimub vastavalt leitud kaaludele punktide summeerimine. Analüüsitakse leitud tulemusi ning antakse hinnang, milline raamistik antud kriteeriumitele kõige paremini sobis. Leitakse, et olgugi, et vaatluse all oleva ettevõtte kasutuses olev Yii raamistik on küllaltki kaasaegne, siis soovitatakse antud töö tulemusi arvesse võttes kaaluda Cake raamistiku valimist.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 3 peatükki, 2 joonist, 3 tabelit.

Abstract

In this thesis different PHP frameworks were assessed for their strengths and weaknesses. The main purpouse of this thesis was to find out the most modern and optimal framwork for a PHP web development company.

The thesis is devided into three main parts: Overview, framework assessment aspect selection, and the result analysis.

In the overview the backround of the subject is described and the overview of the thesis and the methology used in comparing the frameworks is given.

In the framework comparison chapter, as the name implies, the frameworks are assessed using previously found aspects. The different frameworks chosen are briefly described and the strengths or weaknesses of each aspect for each framework is described. In addion, each aspect is rated on a 10 point scale.

In the results chapter, all the different aspects are analized for differences between the frameworks and the calculated endscore of each framework is delivered. The most suitable framework for the found aspects is found and the results are analized. It is found that the framework currently used by the company in question, is indeed quite modern, but still the higher rated Cake framework should be considered by the company.

The thesis is in written in Estonian and contains 40 pages of text, 3 chapters, 2 figures, 3 tables.

Lühendite ja mõistete sõnastik

HTML	***HyperText Markup Language*** Dokumendi struktureerimise keel veebilehtede puhul.
REST	***Representational state transfer*** Standardiseeritud Arhitektuuriline stiil info vahetuseks kliendi ja serveri vahel.
MVC	***Model-View-Controller*** Üldnimetus mudel – vaade – kontroller põhisele lähenemisele rakenduse arendusprotsessis.
SPA	***Single-page Application*** Veebileht, kus sisu muutub dünaamiliselt seejuures lehte vahepeal värskendamata.
ACL	***Access control list*** Andmete kogum kasutaja õiguste suhtes.
RBAC	***Role-Based Access Control*** Rollipõhine kasutajate õiguste süsteem.
LAMP	***Linux, Apache, MySQL, and PHP*** Lühend enamlevinuma veebirakendusi majutava tarkvarapaketi kohta, mis hõlmab Linux operatsioonisüsteemi, Apache serveritarkvara, MySQL andmebaasirakendust ning PHP'd.
CamelCase	***CamelCase*** Programmeerimiskeeles kasutatav stiil, kus muutujad või meetodid nimetatakse kokku kirjutatuna ning suurte algustähtedega.
IDE	***Integrated development environment*** Tarkvara arenduse programm.

CRUD	***Create, Read, Update, Delete*** Akronüüm loomise, lugemise, uuendamise ja kustutamise päringute ühise nimetajana kirjeldamiseks.
Composer	***Composer*** Composer on tööriist sõltuvuste manageerimiseks PHP keeles. Composeri abil on võimalik deklareerida ja hallata projekti sõltuvusi. [1]

Jooniste nimekiri

Tabelite nimekiri

Sisukord

1. Sissejuhatus

Sõltumata keelest või arhitektuurist etendavad raamistikud tänapäeval märkimisväärset rolli nii mastaapsemate kui ka väiksemate süsteemide arendusprotsessides. *World Wide Web* algusajast saadik 20. sajandi viimases kümnes on veebirakenduste arendus läbinud suuri muutuseid. Kui algselt esimesed veebilehed olid pigem staatilist ja lihtsat HTML sisu edasi kandvad, siis aja möödudes on nõuded süsteemidele ning seeläbi ka süsteemide keerukuse tasemed märkimisväärselt tõusnud ning kaasa toonud oluliselt keerukamad lahendused ja probleemid.

On teatav ühisosa enamikel rakendustel, mille funktsionaalne eesmärk on sarnane ja selle pidev taas-arendamine on otseselt võrreldav piltlikult öeldes jalgratta uuesti leiutamisega. Olgu selleks andmebaasiga suhtlus, kasutajale vaadete kuvamise protsess või muu äriloogikat mitte hõlmav element. Sellest tulenevalt on kujunenud välja abivahendid, mille ülesandeks on lihtsustada standartsete protsesside arendamise vajadus – **raamistikud**.

Raamistikud pakuvad juba toimivaid lahendusi probleemidele, millega uut rakendust tehes arendajad muidu silmitsi seisaksid – muuhulgas näiteks andmebaasiga seotud suhtluse ja baastabelite loomine, kasutaja autoriseerimise- ning vaateloogika ja palju muid lisamooduleid, mille nullist arendamine oleks ressursikulukas nii arendajale kui kliendile. Siinkohal tuleb ka arvesse võtta paratamatut vigade marginaali, mis testimata süsteemiga pahatihti kaasas käib. Levinud raamistike arendusprotsess on reeglina pidev ning tagasiside kasutajalt võetakse jooksvalt arvesse ja tänu sellele tehakse ka aktiivselt parandusi süsteemidesse.

1.1 Taust ja probleem

Käesolev töö on innustatud ühe veebi tarkvara arendusega tegeleva ettevõtte probleemidest. Konkreetne ettevõte on juba aastaid oma arendusi teinud Yii raamistikule. On selge, et pika aja vältel sama süsteemiga tegelemine suurendab selle süsteemi meisterlikkuse taset, kuid küsitav on seejuures töövahendi optimaalsus konkreetsele ettevõttele. Esialgses valikuprotsessis oli põhirõhk vaadete kuvamise loogikal, kuid ettevõtte on tänasel päeval läinud peaaegu täismahus REST teenuste peale, kus vaateid serveeritakse kliendi pool, mitte serveris. Raamistike valikuprotsess on aeganõudev töö ning on vähe kasu vaid kahe raamistiku võrdlemisest, samas on ka ülimalt keeruline võrrelda kõiki olemas olevaid lahendusi. Töö peamiseks probleemiks

on ühele veebiarendusega tegelevale ettevõttele kaasaegse sobiva raamistiku leidmine. Kindlasti on käesolev töö abiks kõikidele veebiarendusega tegelevatele üksikisikutele ja ettevõtetele.

1.2 Ülesande püstitus

- Välja töötada kriteeriumid, mille abil raamistike võrrelda
- Leida välja töötatud kriteeriumite abil sobilik valim
- Analüüsida välja valitud raamistikke
- Analüüsida tulemid

1.3 Metoodika

Võrdluses osalevad aspektid annavad kõik kuni 10 punkti, kuid on välja toodud erinevate kaaludega. Kaalud on välja toodud iga aspekti juures eraldi. Maksimaalne võimalik punktisumma on 10, seejuures iga osa punktisumma korrutatakse läbi tema kaaluga ning seejärel liidetakse punktid kokku, mille abil moodustab lõplik punktisumma (Valem 1).

$$S_j = \sum_{i=1}^{7} M_i X_{ij}, \text{ kus} \quad (1)$$

M – koefitsient, i –aspekti number, X – vastuse punktid, j – raamistikud (1…7)

1.3.1 Võrreldavad raamistikud

Töös osalevate raamistike valikul mängib peamiselt rolli autori teadlikkus olemasolevatest raamistikest. Lisaks tehakse ka internetis päringud raamistike populaarsuse kohta ning sellest tulenevalt valitakse 3-5 sobivat kandidaati. Antud töö jaoks raamistike valik toimub kahes etapis. Levinuimatest raamistikest valitakse välja need, mis täidavad kõiki järgnevas alampunktis väljatoodud mitte-funktsionaalseid nõudeid. Valitud raamistike puhul antakse hinnang, kasutades praktilise eesmärgiga kindla ülesehitusega näidisrakenduse loomist, mis on abiks vaatluse all olevate aspektide hindamiseks. Töö lõpuks selgub raamistik, mis on kõiki töös käsitletud aspekte arvesse võttes kõige ökonoomsem arendusteks.

1.3.2 Võrdluses osalevad aspektid

Kõik valitud raamistikud peavad toetama järgnevaid tööriistu või omadusi, mida käesolevas töös eraldi ei hinnata. Mõne järgnevalt nimetatud funktsionaalsuse puudujääk välistab selle raamistiku vaatluse alla võtmise.

1. Mudelite tugi – MVC põhise raamistiku üheks alustalaks on mudelid, mis vahendavad andmebaasis olevate andmete töötlust ja manipulatsiooni.

2. Kontrollerite funktsionaalsus - Kontrollerid vastutavad info vahendamise eest kliendilt klient-server tüüpi rakenduses.

3. Sisse ehitatud vaadete loomise struktuur.

4. Vormide valideerimine – Sissetulevate andmete valideerimise ja korrektsuse eest vastutavad vormid, mille abil tagatakse rakenduses töötlemiseks kasutatavate andmetüübide ja kuju õigsus.

5. Modulaarsus – Raamistik peab olema laiendatav moodulite abil ehk võimaldama lisada eraldiseisvaid taaskasutatavaid rakenduse osasid, millel on kindel spetsiifiline funktsioon.

6. *Composer* tugi – *Composer* on kõrgel tasemel PHP rakenduste arendusel vajalik tööriist, mis lisab süsteemile modulaarsust ja edasiarendatavust.

7. Arvestatav kasutajaskond – Lisaks keelelistele eripäradele on oluline, et raamistik ka piisava aktiivse kasutajaskonnaga oleks. Selle aspekti hindamisel võetakse arvesse kogukonna aktiivsust *Stack Overflow* keskkonnas ning raamistikul peaks olema vähemalt 2000 seotud küsimust.

1.3.2.1 Seadistamine ja generaatorid

Seadistamine on projekti loomisel peamiselt ühekordne protsess, kuid samas võib kuluda ebaoptimaalne aeg teatud funktsionaalsuste konfigureerimisele. Seega on oluline, et raamistiku puhul seadistamisele kuluv ajakulu oleks minimaalne. Lisaks hinnatakse antud punktis ka olemasolevate koodigeneraatorite kasutatavust ja kasulikkust.

Selle punkti kaal antud skoobis on hinnatud 10% peale lõplikust.

1.3.2.2 Dokumentatsioon

Korralik dokumentatsioon kiirendab programmeerimisele kuluvat aega ning lihtsustab arendusprotsessi olulisel määral. Piisavalt dokumenteeritud raamistiku korral on võimalik arendajal hõlpsamini implementeerida raamistikku enda rakendusse. Korraliku dokumentatsiooni puudumine seevastu vähendab raamistiku väärtust, kuna arendajad ei ole täiest funktsionaalsusest tihtipeale teadlikud ja seetõttu ei oska potentsiaali ära kasutada.

Dokumentatsiooni osakaaluks antud töö raames on 15%

1.3.2.3 Arendamine ja õppimiskõver

Olgugi, et raamistikud on sama arhitektuuri peal on siiski teataval määral erinevusi nende ülesehitustes ning see toob kaasa ka erinevused ajas, mil arendajal on võimalik raamistikuga efektiivselt tööd teha. Siinkohal kuulub hindamise alla ka koodi stuktuurne ülesehitus ning koodi kaasaegsus. Samuti arendusrakendusega kokkusobivus kuulub siin hindamise alla.

Selle punkti osakaaluks antud töö raames on 10%

1.3.2.4 Liidestuse tugi

Käesoleva töö kirjutamise hetkel on üha enam veebi rakendusi muutumas ühe-lehe-aplikatsioonideks (edaspidi SPA). SPA puhul vastutab vaadete eest *JavaScript* kliendi poolel, mitte ei serveerita vaateid serverist nagu traditsiooniliste veebilehekülgede puhul. Seetõttu on selliste rakenduste puhul oluline PHP raamistiku tugi näiteks REST teenustele. Ühtlasi väärib ka võrdlemist rakenduse teenuste dokumentatsiooni genereerimise võimekus, mille abil on automatiseeritult võimalik luua teenuste tehniline kirjeldus.

Liidestuse osakaaluks antud töö raames on 20%

1.3.2.5 Lokalisatsioon

Mitmes eri keeles sisuga manipuleerimine on suur aspekt, mida PHP raamistikult nõutakse. Lokalisatsiooni tugi moodustab suure osa hinnangust, mis raamistik lõppkokkuvõttes saab, kuna eraldiseisvana keele toe lisamine on võrdlemisi ressursikulukas ettevõtmine arendajale.

Lokalisatsiooni toe osakaaluks antud töö raames on 10%

1.3.2.6 Testitavus

Testitavus PHP raamistiku valimisel on oluline aspekt. Testitavuse all mõistame süsteemi testkriteeriumite loomist ja tema komponente ning testide käivitamist, et veenduda

kriteeriumitele vastavuses ja leida isoleeritult vigu [2]. Funktsionaalsete (*Functional testing)* ja ühik-testide (*Unit testing)* tugi on vajalik veebirakenduse töökindluse garanteerimiseks. PHP raamistikes levinult kasutatava *ActiveRecord* tüüpi andmemudelite tõttu on keeruline ühik-testida mudeleid hõlmavaid meetodeid ilma kirjeid andmebaasis manipuleerimata.

Testitavuse osakaaluks antud töö raames on 20%

1.3.2.7 Turvalisus

Turvalisuse all mõistetakse nii süsteemi võimekust takistada disainitud funktsionaalsuse välist kasutust kui ka andmete turvalisust kolmandate osapoolte eest [2]. Süsteemi turvalisuse all hinnatakse kuivõrd on turvalisusele rõhku pandud antud raamistiku disainil, näiteks ACL või RBAC tugi.

Turvalisuse kaal antud skoobis on hinnatud 15% peale lõplikust.

1.3.3 Rakendus

Objektiivsemaks võrdluseks luuakse valitud raamistikega ka väiksemahuline rakendus, mille abil eri aspektide hinnangu protsess kujuneb. *JavaScript* baasil olev vaate kiht, mida kasutatakse PHP raamistiku liideste välja kutsumiseks, eeldab rakenduselt järgnevas tabelis välja toodud liideseid (Tabel 1).

Olgu öeldud, et käesoleva töö väärtus ei seisne konkreetses rakenduses, vaid rakenduse loomine on abiks analüüsimaks raamistike erinevusi ning võimekusi ka praktilistest vaatenurkadest.

Tabel 1. Näidisrakenduse teenuste kirjeldus

Päring	Sisend	Kirjeldus
GET /topic/topic	-	Teemade nimekirja päring. Tagastab kõik teemad süsteemis.
POST /topic/topic	`{ title: "Pealkiri" description: "Kirjeldus" }`	Uue teema loomise päring. Tagastatakse sama teema.
GET /topic/topic/:id	-	Teema vaatamise päring koos postitusega. Süsteem peab teema külge siduma kõik postitused.
POST /topic/:id/post	`{ text: "Testkommentaar" }`	Teadlikult ebaoptimaalne päring postituse tegemiseks teemasse. Sisendiks tekst, süsteemilt eelduseks kasutaja viite lisamine postitusele.
PUT /topic/post /:id	`{ text: "Testkommentaar" }`	Postituse muutmise päring. Analoogne uuendamise päringule
DELETE /topic/post /:id	-	Postituse kustutamine. Süsteem vastab ainult HTTP koodidega.

2. Raamistike võrdlus

Raamistike valikul võrdlusesse kasutati eelkõige Google otsingut järgnevate märksõnadega: *„Most popular PHP framework 2015“*, *„Top PHP frameworks“*. Saadud tulemusi analüüsides jõuti järgnevate raamistikeni: Laravel, Symfony, Nette, CodeIgniter, Yii, Zend, CakePHP, Phalcon [3] [4]. Nimetatud valimist täpsema võrdlusprotsessi valimiks kujunesid Laravel, Symfony, Yii, CakePHP ning Zend. Teise etapi valimist jäid välja Nette, Phalcon ja CodeIgniter põhjustel, mis on välja toodud punktis käesoleva peatüki punktis 2.2.

2.1 Valitud raamistikud

2.1.1 Yii

Käesolevas töös käsitleti viimast stabiilset versiooni Yii'st, milleks oli 2.0.7.

2.1.1.1 Dokumentatsioon

Yii on väga põhjalik raamistik, mis on koodile vastavalt ka põhjalikult dokumenteeritud. Kuna Yii omab väga palju tööriistu erinevate raskusastmetega probleemide kiireks ja korrektseks lahendamiseks, siis dokumentatsiooni maht on võrdlemisi suur. Sellegipoolest on kogu raamistiku funktsionaalsus koos näidetega eeskujulikult kirja pandud. Lisaks täiendab dokumentatsiooni aktiivne ja võrdlemisi suur kogukond. Yii põhjalik dokumentatsioon ei vea alt võimast raamistikku, mida Yii kahtlemata on.

Tulemus: 10p

2.1.1.2 Seadistamine ja generaatorid

Yii baasrakenduse installeerimine on võrdlemisi lihtne. Mõne Composer käsuga on rakendus püsti ja initsialiseeritud. Yii vajab päris suurt hulka erinevaid komponente, mistõttu on rakenduse esialgne seadistamine konkurentidest selgelt aeglasem protsess – kuid seejuures mitte keerukam. Yii omab ühena vähestest ka eraldi veebibrauseris avatavat koodi-genereerimist hõlbustavat liidest „*Gii*“, mis on suureks abivahendiks arendajale. Andmebaasistruktuuri olemasolul luuakse mudelid, kontrollerid ja vaated ning soovi korral ka täisväärtuslik CRUD haldus. Võttes arvesse Yii mudelite komplekset ülesehitust, siis generaatori kasutus oleks ka eelistatud viis mudelite loomiseks.

Tulemus: 7p

2.1.1.3 Arendamine

Yii on arendajasõbralik raamistik. Kood on kaasaegne ja lihtsasti loetav. Klassid ja meetodid on selgelt ja korrektselt dokumenteeritud ja IDE tugi on täielik. Yiiga kaasneb hulgaliselt väärtuslikke ja võimsaid abiklasse, mis annavad raamistikule ohtralt lisaväärtusi. Kõikvõimalikud lisaklassid ja abimeetodid raskendavad algaja arendaja seisukohalt mõningal määral konkreetse raamistiku õppimisprotsessi, kuna intuitiivselt on keeruline jõuda olemasolevate lahenduse kasutusteni. Vaid üheks näiteks võib Yii puhul tuua võimsa massiividega töötamise abilise „*ArrayHelper*, mis transformeerib keerukamad massiivide töötlused väga lihtsalt loetavaks koodiks. Tööriist on olemas ja väga hästi dokumenteeritud, kuid arendaja, kellel vähe kokkupuuteid Yiiga ei pruugi seda leida. Selge on see, et tegu ei ole negatiivse, kuivõrd õppimiskõverat tõstva aspektiga. Eelnevalt väljatoodu teeb Yii'st väga võimsa tööriista arendajale eeldusel, et arendaja suudab täisväärtuslikult kasutada kogu raamistiku võimsust.

Tulemus: 9p

2.1.1.4 Liidestavus

Kõikidest konkurentidest vajab Yii enim konfigureerimist, et korrektselt JSON kujul andmeid vastu võtta ja väljundisse kuvada. Sellegipoolest on tugi olemas vastavate abiklasside näol, kuid on selge, et Yii puhul pole liideste põhisele arendusele primaarset rolli pandud. Yii peamiseks funktsionaalsuseks on ikkagi täisväärtuslik MVC süsteem, ehk teisisõnu Yii väärtus on autori hinnangul oluliselt suurem, kui panna ta ettenähtud vaadete eest ise vastutama.

Tulemus: 5p

2.1.1.5 Turvalisus ja rollid

Võrreldavatest raamistikest on Yii vaieldamatult kõige võimsama olemasoleva süsteemiga rolliloogika seisukohalt. Yii omab võimast RBAC komponenti, mis on juba seotud pakutava kontrollerite funktsionaalsusega. Teisisõnu on Yii puhul võrdlemisi lihtsa vaevaga kontrolleri tasandil kasutaja õigusi piirata.

Ka autoriseerimisloogika on Yii puhul piisava võimekusega, et suuta tagada süsteemis enam kui piisava turvalisuse. Autoriseerimise baaskomponent lubab eriviisilisi autoriseerimise meetodeid registreerida samade kasutajate puhul, mis on kasulik näiteks Eesti E-ID autoriseeriserimiste puhul.

Tulemus: 10p

2.1.1.6 Testitavus

Testitavus on Yii puhul väga põhjalikult kaetud. Yii kasutab funktsionaalseks testimiseks Codeception ja ühiktestimiseks PHPUnit abi, mis on täiesti piisav. Mõningaseks takistuseks Yii puhul automaat-testide arendusel on ActiveRecord tüüpi andmemudeli kasutamine, mistõttu saab testimisel kasutada vaid korrektseid andmebaasis olevaid andmeid. See võib osutuda aeganõudvaks protsessiks korrektsete relatsioonide fikstuuride loomisel.

Tulemus: 8p

2.1.1.7 Lokalisatsioon

Yii omab loogilist tõlgete funktsionaalsuse struktuuri, kus tõlkeid on võimalik laadida failisüsteemist, andmebaasist või mõlemast korraga. Lisaks on kõik tõlked grupeeritavad, mis teeb tõlgete halduse märkimisväärselt lihtsamaks. Mitmekeelse süsteemi loomine Yii puhul on täiesti kaetud, kuid mõned probleemid siiski jäid silma. Näiteks võib probleeme tekkida kasutajale kuvatava veateate näol, kui mõnes keeles tõlge puudu on, kuna sellisel juhul ei valita automaatselt olemasolevat keelt, mida konkurendid suudavad teha.

Tulemus: 9p

2.1.2 Symfony

Käesolevas töös käsitleti viimast stabiilset versiooni Symfonist, milleks oli 3.0.4.

2.1.2.1 Dokumentatsioon

Vaatluse all olevatest süsteemidest oli üks põhjalikuima dokumentatsiooniga raamistik. Tegu on erakordselt suurt ja aktiivset kogukonda omava raamistikuga. Lisaks sellele on põhjalik dokumentatsioon saadaval ka kodulehel ning väljastatud on nii ametlikke kui ka mitteametlikke arendust soosivaid raamatuid, millest mõningad on ka tasuta kättesaadavad veebiväljaandena.

Tulemus: 10p

2.1.2.2 Seadistamine ja generaatorid

Symfony install on kerge sõltumata operatsioonisüsteemist. Baas-rakenduse ülesseadmine Linux serverisse võtab küll võrdlemisi kaua aega, aga valitud raamistikest on tegu selgelt ka kõige massiivsema rakendusega, võttes baasrakendusena kettal ruumi 55MB, mis on kordades rohkem kui kõikidel konkurentidel.

Symfony pakub hulgalistelt esmaklassilisi generaatoreid, mis arendaja elu olulisel määral lihtsustavad. Lihtsa vaevaga on näiteks võimalik luua uusi mudeleid nii koodi kui andmebaasi, kasutades selleks vaid käsurida. Sellised generaatorid hoiavad hulgaliselt aega kokku ja näiteks andmebaasi väljade loomine ning sealt ka käsitsi rakenduse mudeli formuleerimine on niivõrd aeganõudev protsess, et Symfony generaatorid suudavad selleks kuluvat aega kümneid kordi vähendada.

Seevastu on generaatoritel ka negatiivseid aspekte. Nimelt ei ümbritseta SQL lauseid transaktsiooniga, mistõttu ühe päringu ebaõnnestumine ei pööra tagasi teisi päringuid ning tuleb käsitsi baasi korrigeerida. Positiivne seejuures on, et võimalik on generaatorist küsida ka toores SQL päring, mida käsitsi oluliselt turvalisem käivitada on ja vajadusel saab transaktsiooni sinna juurde lisada.

Tulemus: 9p

2.1.2.3 Arendamine

Rakenduse arendamisel nõuab Symfony seevastu üsna spetsiifilist ja konkreetset lähenemist. Täna kompleksele ja PHP mõistes võrdlemisi ebastandartsele lähenemisele jääb arendaja dokumentatsiooni lugemata tihemini hätta kui konkureerivate raamistike puhul.

Doctrine mudeli puhul põhifunktsionaalsuse (loomine, uuendamine, kustutamine) rakendamine on märkimisväärselt ajakulukam protsess eelkõige õppimiskõvera seisukohalt. Ühtlasi on kood ka üsna kohmakas, kuna lihtsate operatsioonide tegemine on mitme rea võrra pikem kui konkurentidel. Nagu ka joonisel (Joonis 1) demonstreeritud, siis näiteks Yii2 on võimalik mudel kustutada ühe reaga, kuid Symphony2 puhul kulub selleks oluliselt rohkem koodi. Olgu siin öeldud, et tegu on hinnanguga puhtalt raamistiku baasfunktsionaalsust ning dokumentatsiooni põhjal kirjeldatud juhendit arvestades. On selge, et ise vastav abimeetod valmistada ei ole niivõrd suur vaev üksikjuhtumi puhul.

```
// Delete in Symfony
$topic = $this->getDoctrine()
    ->getRepository('AppBundle:Topic')
    ->find($id);

$em = $this->getDoctrine()->getManager();
$em->remove($topic);
$em->flush();

// Delete in Yii
Topic::deleteAll(['id' => $id]);
```

Joonis 1. Erinevus koodi külluses

Ühe näitena osutub problemaatiliseks Doctrine puhul andmebaasist andmete laadimine, kui *NULL* välja üritatakse serialiseerida massiviks. Server annab vea, kuna NULL kirjega ei õnnestu tühja massiivi tagastada, kuigi elementaarne oleks, et see olukord oleks lahendatud. Antud olukord tekkis baaskasutaja loomisel ilma rolle lisamata. Siinkohal näeb autor lahenduseks käsitsi andmebaasis tekkinud olukorda parandada, kui selline olukord tekkinud on või hoopiski raamistiku koodi parandada oma projektis. See on jällegi lisa ajakulu, mis võiks olla raamistiku poolt lahendatud.

Tulemus: 7p

2.1.2.4 Kasutaja autoriseerimine ja turvalisus

Symfony kasutajate ja autoriseerimise haldus käib läbi FOSUserBundle mooduli, mida ametlikult pakutakse [5]. Paraku tekib selliste moodulite kasutamisel baasrakenduse puhul probleem, kus versioonid ei ühtlustu. Testrakenduseks oli viimane väidetavalt stabiilne versioon, milleks oli töö kirjutamise hetkel 3.0.4. Selle versiooniga tekkis aga mitmete moodulite ühtlustamisel probleeme, ning seda ka konkreetse FOSUserBundle'ga, mis kasutas klasse, mida enam ei toetatud baasrakenduse poolt.

Turvalisuse seisukohalt on FOSUserBundle abil võimaldada süsteemi piisavalt turvaline käitumine ja välistada paljud elementaarsed turvaprobleemid. Arvestades, et tegu on ametliku mooduliga, siis siinkohal etteheiteid ei ole. Peamiseks probleemiks selle mooduli juures on tema tugev seotus brauseri küpsistega ning see seab teatavad piirangud sobiva audentimisviisi valikul, näiteks päringu päisesse autoriseerimisvõtme lisamise näol, mis vajab märkimisväärset lisaarendust.

Tulemus: 7p

2.1.2.5 Liidestavus

Symphony liidestamise puhul tulevad välja mõningad Doctrine head omadused. Eesotsas neist on võimekus automaatselt mudel oma relatsioonidega alammudelite näol välja küsida. See tähendab, et vaikimisi on võimalik mudeleid küsida rekursiivselt koos kõikide mudelite alamrelatsioonidega. Osad konkurendid sellist funktsionaalsust ei paku ning nende puhul teise astme relatsioonide pärimine on väga raskendatud. Yii2 puhul on konkreetne omadus arendamises, kuid hetkel võimalus teise astme relatsioone mudeliga tagastada baasrakenduse puhul puudub. Teenus-tüüpi rakenduse arendamist raskendavad järgnevad ajakulukad probleemid:

- Objekti JSON kujule viimine vajab eraldi komponenti
- JSON kujul objekti ei oska Symphony vastu võtta, mistõttu vajab rakendus lisakonfiguratsiooni teatud javaScripti raamistikega toimimiseks (nt. AngularJs)

Tulemus: 6p

2.1.2.6 Testitavus

Symfony testitavus on eeskujulik. Ühiktestimisel kasutab ka Symfony, sarnaselt kõikidele teistele raamistikele, PHPUnit abi. Funktsionaalsed testide rakendamiseks pole abiraamistikke - nagu näiteks Codeception – kasutusse võetud vaid kasutatud Symfony enda poolt lahendatud tehnoloogiat. Tänu Doctrine tüüpi andmebaasi vahekihile on Symfony puhul võimalik andmebaasist sõltumatute testide loomine.

Tulemus: 10p

2.1.2.7 Lokalisatsioon

Symfony jätkab oma eeskujulikku rolli ka keelte toe osas. Tõlked on lihtsasti formuleeritavad ja arendajale mugavad kasutada. Võimalik on nii andmebaasi kui ka failisüsteemne lähenemine. Positiivne on automaatse keele valik kasutajale veateate andmise asemel, kui mõni tekst on rakenduse administraatoril tõlkimata ununenud.

Tulemus: 10p

2.1.3 Laravel

Käesoleva töö kirjutamise hetkel oli viimane stabiilne versioon Laravel 5.2, mida ka kasutati.

2.1.3.1 Seadistamine ja generaatorid

Laraveli seadistamine on tehtud väga lihtsaks. Linux serveri olemasolul piisab käsurealt kahe käsu käivitamisest, ning andmebaasi konfiguratsiooni seadistamisest, et rakendus püsti saada. Selleks kulub vaid mõnikümmend sekundit. Kui muud raamistikud nõuavad juba olemasolevat LAMP serverit, siis Laraveli on kättesaadav juba täisväärtusliku virtuaalse serveriga (*Homestead*), mis teeb projektide haldamise arendaja jaoks väga kergeks. See teeb Laraveli autori hinnangul erakordselt sobilikuks vähese kogemusega arendaja jaoks. Puudust siinkohal tunneb CRUD generaatorist, mis tuleks otsida kolmanda osapoole lahendusena.

Tulemus: 9p

2.1.3.2 Arendamine

Baasmeetodid on loogilised ning intuitiivselt lähenedes tihtipeale õnnestub dokumentatsiooni lugemata vajalikud operatsioonid ära teha. Paraku Laraveli puhul esines konkurentidest mõnevõrra rohkem olukorda, kus IDE kuvas hoiatusi, kuna paljud meetodid kutsutakse välja tööriista jaoks maagiliselt, mis tähendab, et IDE peab teatavat osa korrektsest arendusest standarditele mittevastavaks. See vähendab koodi subjektiivset ilu ja pisut raskendab korrektset arenduskäiku, kuna raskem on eristada viga korrektsest koodist. Olgu märgitud, et sellele vaatamata on võimalik näiteks PhpStormi puhul lisada kolmanda osapoole teek, mille abil oluliselt paraneb Laraveli IDE tugi.

Tulemus: 8p

2.1.3.3 Liidestavus

Laraveliga on hämmastavalt lihtne teha REST printsiipe järgivat rakendust, kuna automaatselt suudab Laravel mudelid JSON kujule viia. Sellest tulenevalt on Laravel hea raamistik lihtsamaid liidestusi vajavate rakenduste jaoks. Samuti on Laravel koheselt valmis vastu võtma JSON kujul infot ning seda töötlema kui vormi parameetreid. Analoogselt Symfonyga on ka Laraveli puhul kolmanda ja enama alamrelatsiooni pärimine lihtne funktsionaalsus, mis ei vaja keerukat lähenemist. Olgu öeldud, et laraveli puhul osutus oodatust oluliselt keerukamaks vastuse vormindamisel atribuutide *CamelCase* vormistamine. Üllatav on see aspekt seetõttu, et Laravel oma olemuselt peaks olema ja üldiselt ka on väga hea liidestatavusega raamistik.

Tulemus: 9p

2.1.3.4 Dokumentatsioon

Laravel on üldiselt hästi dokumenteeritud raamistik. Juhendid on põhjalikud ja katavad põhifunktsionaalsuse ära. Siiski paljud dokumendid kordavad üksteist esmapilgul sarnase sisuga, kuid täpsemalt uurides selgub vahe rakenduse versioonides. Nimetatud aspekt tekitab pisut segadust õige dokumentatsiooni leidmisel. Väga suur väärtus on Laraveli kogukonnal StackOverflow keskkonnas, kus konkreetse raamistiku eksperte on väga palju ja see on kahtlemata abiks probleemidele vastuste saamisel.

Tulemus: 8p

2.1.3.5 Turvalisus ja rollid

Laraveli baasrakenduses rolliloogikat implementeeritud ei ole. On primitiivsed autoriseerimise abiklassid, mis küll katavad esmase vajaduse ära, kuid rollipõhise süsteemi jaoks tuleks läheneda mõne mooduli abiga. Laravelile on võimalik kolmanda osapoole lahendusena lisada kasulikke turvalisust lisavaid mooduleid, eesotsas väga võimas tööriist nimega „RBAC for Laravel 5.1“ [6].

Tulemus: 4p

2.1.3.6 Testitavus

Laravel on lihtsasti testitav raamistik. Ühiktestimisel on abiks PHPUnit ning funktsionaalseks rakenduse testimiseks on loodud mitmed testimist lihtsustavad abiklassid. Võimalik on nii mudelite, kui ka sündmuste imiteerimine. Raamistik toetab rakenduse testimist täielikult ning seetõttu ka maksimaalsed punktid.

Tulemus: 10p

2.1.3.7 Lokalisatsioon

Laraveli puhul on tõlke tugi küll olemas Symfony poolt pakutava kompondendi näol, kuid baasfunktsionaalsusest on puudu tõlgete andmebaasi salvestamise tugi. Baaskomponent võimaldab ühesõnaga tõlgete laadimist ainult failisüsteemist, mis teeb tõlgete haldamise mõnevõrra keerukamaks. Andmebaasi lahenduse saavutamiseks oleks vaja teha ise täiendav arendus või leida mõni kolmanda osapoole komponent abiks. Positiivne aspekt on, et näiteks tugi tõlkimata teksti käsitlemiseks on olemas.

Tulemus: 7p

2.1.4 CakePHP

Käesoleva töö kirjutamise hetkel oli viimane stabiilne versioon CakePHP 3.2.6, mida ka kasutati.

2.1.4.1 Seadistamine ja generaatorid

CakePHP (edaspidi Cake) hämmastab oma lihtsusega. Seadistamine on imelihtne, piisab Composeri käsu käivitamisest ja on projekt arenduseks valmis. Olemasoleva andmebaasi struktuuri olemasolul saab mõne käsuga luua omale täisväärtusliku CRUD rakenduse koos mudelite, kontrollerite ja vaadetega. Lisaks luuakse paralleelselt ka migratsioonide funktsionaalsus ning automaattestid, millega elementaarsed tegevused ka kohe testidega kaetud said. Lisaks ka korralik pagineerimise moodul ja ka mõistlik kujundus sealjuures. Paigaldamise ja seadistamise seisukohalt on kahtlemata Cake kõige mugavam raamistik valitutest, millega tegutseda.

Tulemus: 10p

2.1.4.2 Arendamine ja generaatorid

Näidisrakendus üllatuslikult vajas arendajalt vaid suunamise konfigureerimist ja muu tehti piisavalt hästi arendajale ette ära. Sarnaselt Laraveliga on Cake väga intuitiivne ja arendaja-sõbralik raamistik. Lähtekood on eeskujulikult dokumenteeritud ja muutujate signatuurid defineeritud. CakePHP hiilgab eriti oma võimega olla väga lihtsasti konfigureeritav. Generaatorid tekitavad automaatselt korrektset MVC süsteemi järgivad mudelid, kontrollerid ja vaated ning arendajal jääb üle vaid delikaatse info eemaldamine. Cake hea omadus on kontrollerite sõltumatus soovitavast vaatest ning samu kontrollereid saab kasutada nii teenuste kui ka HTML vaadete kuvamiseks, mida ühegi teise raamistiku puhul autor ei täheldanud. Eelised Cake puhul Laraveli ees on täisväärtuslik IDE tugi, mille tulemusel on kood puhtam ja korrektsem.

CakePHP puhul on märkimisväärne relatsiooniliste mudelite *ContainableBehaiviour* [7], mille abil on relatsioonide haldamine päringutes ülimalt lihtne ja mille sarnasest käitumisest konkurentide puhul vajaka jääb. Peamine kasu sellisest käitumisest mudelite puhul on võime hallata tema alamrelatsioone päringutes kus kõiki relatsioone vaja ei pruugi minna. See aitab

kokku hoida andmemahtu mida vahetatakse serveri ja kliendi vahel ning mida süsteem töötlema peab ning vähendab ka päringuid andmebaasi pihta.

Cake on kulutanud väga suurt ressurssi ja vaeva ning arendanud omaenda ORM mudeli, mis on autori hinnangul erakordselt hea ja millega on arendajal kerge tööd teha.

Tulemus: 10p

2.1.4.3 Liidestavus

Cake päringute suunamise (ingl.k. *routing*) konfiguratsioon on väga loogiliselt ja mugavalt üles ehitatud ning on arendaja jaoks kergesti konfigureeritav. Lisaks on toetatud baaskontrolleri tasemel nii XML kui JSON andmekihid, mis on loodud käituma kui vaadete alternatiivid ning on seetõttu väga lihtsasti implementeeritavad üheaegselt nii teenus-tüüpi rakenduses kui ka serveri poolt vaate formuleerimisel.

Tulemus: 10p

2.1.4.4 Dokumentatsioon

Cake dokumentatsioon on põhjalik ja katab elementaarsed elemendid viisakalt ära, kuid siiski jätab elementaarsete teadmiste nõude arendajalt alles. Tänu sellele on hoitud dokumentatsiooni mahtu kokku jättes vajalikud eelteadmised kõrvale, kuid mille õppimiseks on heal juhul jäetud korrektne viide. Lisaks on Cake on suurepärase kasutajaskonnaga raamistik omades suurt kasutajaskond StackOverflow keskkonnas.

Tulemus: 9p

2.1.4.5 Turvalisus ja rollid

Käesoleva töö kirjutamise hetkel CakePHP 3.0 ei omanud veel täielikku ACL tuge. Vastav komponent on eelmise versiooni baasrakendusest välja tõstetud ja kasutatav, kuid alles arendusjärgus ja stabiilse versioonita. Sellegipoolest omab Cake 3.0 piisavat autoriseerimise loogikat koos elementaarsete turvalisuse aspektidega, mille hulgas ka krüpteerimise abiklassid esmase vajaduse täitmiseks.

Tulemus: 6p

2.1.4.6 Testitavus

Cake on testitavuselt Laraveliga väga sarnane. Ühiktestide kasutamine käib läbi PHPUnit raamistiku ning funktsionaalne testimine läbi Cake'i enda abiklasside. Lisaks on saadaval ka täisväärtuslik Codeception moodul, mis on integreeritud Cake mudelisse. Üldises plaanis on testitavus ka Cake puhul eeskujulikul tasemel.

Tulemus: 10p

2.1.4.7 Lokalisatsioon

Cake raamistikku on implementeeritud väga võimas tõlgete tööriist. Tõlkeid võib laadida nii failidest kui ka baasist, kusjuures baasfunktsionaalsus lubab tõlkeid hoida nii koos ühes tabelis, kui ka mudelispetsiifilist struktuuri kasutades. Tänu sellele, et tegu on raamistiku meeskonna enda arendatud funktsionaalsusega, siis on lokalisatsioon põimitud mugavalt ORM mudeliga koostöös olema. Ühtlasi on võimalik kasutada olemasolevaid abivahendeid ka näiteks kuupäevade ja rahaühikute lokalisatsioonipõhiseks kuvamiseks.

Tulemus: 10p

2.1.5 Zend

Käesoleva töö kirjutamise hetkel oli viimane stabiilne versioon Zend 2.5.1.

2.1.5.1 Dokumentatsioon

Zend dokumentatsioon on käepäraselt jaotatud ning võimalik on valida endale sobiva versiooni järgi juhendid ja dokumendid. Siiski käesolev töö kasutas uusimat versiooni raamistikust ning selle versiooni kohta dokumentatsiooni valida ei saanud. Üldiselt dokumentatsioon on temaatiliselt põhjalik, kuid sisult ebapiisav ja see kajastub ka kasutajate kommentaarides. Dokumenteeritud on Zend, nagu ka kogu lähtekood, aegunud PHP metoodikat kasutades ja tänapäeval vähendab see autori hinnangul loetavust.

Tulemus: 4p

2.1.5.2 Seadistamine

Zend seadistamine on algajale üsna keeruline. Juhenditest on välja jäetud mingil määral kriitilist informatsiooni, et rakendus toimima saada. Kui teiste raamistike puhul polnud autoril probleeme peale lühikest aega oma kontrollerite pihta päringute tegemine, siis Zendi puhul toob

kõige primitiivsema päringu seadistamine välja korraliku laviini muresid rakenduse seadistamisel. Olles juhendit samm-sammult jälginud, ei garanteeri see rakenduse töötamist ning sama kajastub ka hulgalistes kommentaarides. Sellest tulenevalt võib kuluda ootamatult palju aega enne arenduse algust. Kindlasti siinkohal on ka määravaks autori kogenematus vanema PHP süntaksi osas, kuid teisest küljest on üha keerulisem leida uusi kasutajaid kui arenduseks on vajalikud ebaaktuaalsed teadmised spetsialistil.

Tulemus: 3p

2.1.5.3 Arendamine

Zend puhul paistab silma läbivalt vanemat tüüpi PHP keele kasutamine. Üheks näiteks on massiivide puhul „*array()*“ kasutamine kantsulgude „*[ ]*“ asemel. Ühest küljest on see mõistetav, et vanemate versioonide tugi alles jäetakse, kuid teisest küljest tuleks koodi kindlasti ka kaasajastada, et arengut mitte peatada. PHP.net andmetel on tugi 5.3 versioonile lõpetatud käesolevast hetkest ligi 2 aastat tagasi, seega ei ole vana versiooni toetamine turvakaalutlustel enam õigustatud [8].

Zend raamistik ei kirjuta ette, millist mudeli struktuuri kasutama peaks, kuna arendajate sõnul on tegu rakenduse äriloogikat mõjutava aspektiga [9]. Ühest küljest on see täiesti mõistetav ja loogiline, kuid teisest küljest tähendab see suurel määral lisakulutusi mudeli arhitektuuri valikul. Lisaks puuduvad raamistikul sellest tulenevalt tugistruktuurid mudelitega töötamiseks ning ka andmebaasiga ühendus tuleb endal arendada.

Tulemus: 5p

2.1.5.4 Liidestavus

Zend toetab erinevaid liidestuse meetodeid, kuid nende seadistamine on võib osutuda üsna keerukaks ja aeganõudvaks. Juhendeid ja baasrakenduse koodi jõuab järeldusele, et subjektiivse hinnangu kohaselt ei ole liidestatavus Zendi tugevate külgede hulka kuuluv omadus. Lisaks on ebavajalikult keerukas liideste asukohtade defineerimine. Zendi puhul on mõistlik kasutada mõnda kolmanda osapoole lahendust, kus baasfunktsionaalsust on parandatud ning mis teevad rakenduse liidestuse konfigureerimise kergemaks.

Tulemus: 5p

2.1.5.5 Turvalisus ja rollid

Zendiga kaasneb korralik sisse ehitatud RBAC funktsionaalsus, mille abil on võimalik kontrollerite tasandil rollipõhine kasutaja õiguste piiramine. Kasutaja autoriseerimise loogika ja funktsionaalsus on Zendi puhul eeskujulikult lahendatud ja kõik vajalik on olemas rakenduse primitiivse turvalisuse tagamiseks.

Tulemus: 10p

2.1.5.6 Testitavus

Testitavus Zendi puhul piirdub PHPUnit kasutamisega, mis tähendab, et põhirõhk on pandud ühiktestidele. Täiendavaid testimisvõimalusi baasrakendus olulisel määral ei paku, kuid lisatud on primitiivne abimetoodika kontrollerite funktsionaalse testimise näol. Sellise kasutajaskonnaga raamistikult eeldaks pisut suuremat funktsionaalsete testide võimekust, kuid selleks tuleb lahendust otsida raamistikust väljastpoolt.

Tulemus: 7p

2.1.5.7 Lokalisatsioon

Arendaja kasutuses on piisavalt võimekas lokalisatsiooni tööriist, mille abil on mitmekeelse rakenduse loomine võrdlemisi lihtne. Analoogselt Laravelile on puudu andmebaasis tõlgete hoidmise tugi, mis teeb tõlgete halduse arenduse keerukamaks.

Tulemus : 7p

2.2 Välja jäänud raamistikud

2.2.1 CodeIgniter

Vaatamata sellele, et tuginedes erinevatele allikatele [3] [4] on *CodeIgniter* raamistiku näol tegu ühe populaarseima raamistikuga, on käesoleva töö kirjutamise ajal ta teistega võrreldes autori hinnangul paar sammu konkurentidest maas. Peamisteks põhjusteks, miks *CodeIgniter* raamistik skoobist välja jäi on:

- Puudub mugavalt kasutatav andmebaasiga suhtluse vahekiht mudelitele.
- Puudub *Composer* tugi.
- Lähtekoodis PHP klasside kirjeldus puudulik ja kood on aegunud.

2.2.2 Nette ja Phalcon

Nii Nette kui ka Phalconi populaarsus on käesoleva töö kirjutamise ajal selgelt kasvamas, kuid käesoleva töö kirjutamise hetkel on tegu veel vähese kasutajakonnaga raamistikega. Nette puhul oli *Stack Overflow* portaalis vaatlusaluse perioodi kohta alla 50 seotud teema. Kindlasti ei ole Nette kogukond nii väike, kuna tegu on võrdlemisi populaarse raamistikuga osade allikate põhjal [3]. Täpsemal uurimisel selgub, et Nette populaarsus on suures osas inglise keelt mitte kõnelevas keskkonnas, mistõttu ka vähene inglise keelne kasutajaskond.

Phalconi puhul oli *Stack Overflow* portaalis küsimuste arv mõnevõrra suurem, ligikaudu 1000 teema ringis, kuid subjektiivse hinnangu kohaselt pidas autor ka seda ebapiisavaks. Mõlema raamistiku puhul on tegu esmapilgul väga tugevavate konkurentidega, millel on suur potentsiaal tulevikus populaarsust koguda ja mõne aasta pärast kogukonna suurenedes võib nende juurde naaseda.

3. Tulemused

Tabelis (Tabel 2) on välja toodud raamistike punktid iga arvesse võetud aspekti osas ning käesolevas peatükis analüüsitakse saadud tulemused.

Tabel 2. Raamistike punktisummade kokkuvõte

	Yii	CakePHP	Symfony	Zend	Laravel
Seadistamine	7	10	9	3	9
Dokumentatsioon	10	9	10	4	8
Arendamine	9	10	7	5	8
Liidestavus	5	10	6	5	10
Testitavus	8	10	10	7	10
Turvalisus	10	6	7	10	4
Lokalisatsioon	9	10	10	7	7

3.1 Seadistamine ja generaatorid

Selle alampunkti juures hinnati kui komplitseeritud on raamistiku esialgne seadistamine ning mil määral on võimeline raamistik abi pakkuma arendajale erinevate generaatorite näol. Raamistiku lõpphindest moodustab saadud punktide kogus 10% ning kalkuleerituna lõpptulemiks on võimalik maksimaalselt lisada siit ühe punkti lõppskoori.

Selles aspektis antud valimi hulgas võis kõige vähem rahule jääda Zend raamistikuga, mis teenis vaid 3 punkti, mis punktiarvestusse annab 0,3p. Zendi seadistamine nõuab arendajalt väga suurt ressurssi ja enim aega dokumentatsioonis vajaliku informatsiooni otsimisele.

Vaatamata Yii väga heale abilisele *Gii*, mille näol on tegu eraldi automaatselt genereeritud keskkonnaga, kus läbi kasutajaliidese mugavalt arendaja aega säästa saab, jäi ta siiski teistele konkurentidele alla, eelkõige tema konfigureerimise keerulisuse tõttu. Yii kalkuleeritud tulemus selles kategoorias 0.7 punkti.

Symfony seadistamine on ligilähedane täiuslikkusele ning eeskujulikud 0,9 punkti sellest kategooriast.

Võrdväärselt esmaklassiliselt lihtsa konfigureerimisega raamistikeks osutusid Laravel ja Cake, millede puhul rakenduse seadistamine arendajalt minimaalset pingutust nõuab.

3.2 Dokumentatsioon

Selle peatüki all anti hinnang raamistiku kasutusjuhenditele, dokumenteeritusele ning arvestati ka koguduse aktiivsust. Tegu on üsna kaaluka aspektiga, kuna arendusprotsess hõlmab ka aktiivselt dokumentatsioonist abi otsimist, mis peaks olema vähe aega nõudev toimetus. Seetõttu on selle aspekti kaaluks 15%, ning summaarselt maksimaalsed võimalikud punktid on 1.5p.

Ka siin oli Zendi puhul märgata oluliselt madalamat kvaliteeti kui konkurentidel. Zendi dokumentatsioon oli vaadeltavate aspektide puhul kohati selgelt ebapiisav ning sama kajastus ka kasutajate kommentaarides. Summaarselt tulemuseks 0,6 punkti.

Kõigi teiste raamistike dokumentatsioonide kirjeldustega võib autori hinnangul rahule jääda. Mõnevõrra kehvema tulemuse sai Laraveli dokumentatsioon, kuid mitte ebapiisavusest vaid pigem andmehulga küllasusest, mis tegi õige informatsiooni leidmise pisut keerukamaks. Laraveli arvutuslikuks tulemuseks kujunes 1,2p.

Maksimaalsest punktisummast pisut jäi puudu Cake raamistikul, eelkõige seetõttu, et dokumentatsioon eeldab lugejalt teatavaid baasteadmisi ning seetõttu arvutuslik tulemus 1,35 punkti.

Kõige põhjalikumalt dokumenteeritud raamistikeks on Yii ja Symfony saades maksimum, 1.5p selles kategoorias. Lisaks omavad mõlemad raamistikud raamistikku õppivale arendajale tehtud käsiraamatuid.

3.3 Arendamine

Selle peatüki all hinnati koodi ülesehitust, kaasaegsust, üldist õppimiskõverat ning ka arenduskeskkonnaga sobivust. Aspekt on võrdlemisi madala kaaluga, kuna omab väiksemat rolli arenduse pikas perspektiivis. Siiski on tegu vajaliku aspektiga konkreetses võrdlussüsteemis. Summaarsesse punktitulemisse on siit võimalik lisada 1p.

Zendi puhul jäi eelkõige silma laialdane aegunud koodikasutus. Samuti on autori hinnangul konkreetse mudelistruktuuri puudumine tagasilöögiks raamistiku konkurentsivõimele. Seega kalkuleeritud punktitulemuseks pool võimalikust, ehk 0.5 punkti.

Symfony tugevaks küljeks on selgelt tema komponentide modulaarsus, mis eri projekte arendades võimaldab läbivalt samasisulisi komponente konkurentides kõige lihtsamini taaskasutada. Symfony on autori hinnangul kõige järsema õppimiskõveraga. Kohati üsna keerukas ülesehitus valmistab arendajale antud valimist enim probleeme. Samuti vajab Symfony tihtipeale konkurentidega võrreldes märkimisväärselt rohkem koodi arendajalt. Sellele vaatamata on kood kaasaegne, meetodid korralikult defineeritud ning võrdlemisi tugev kalkuleeritud punktisumma 0.7 punkti.

Laraveli olulisemaks puudujäägiks on IDE toe vähesem ära kasutamine konkurentidega võrreldes, mis vähendab arenduskiirust, kui kasutusel on professionaalsed tööriistad nagu näiteks PhpStorm. Sellegipoolest on Laravel arendajasõbralik raamistik ning kalkuleerituna teenib 0.8 punkti.

Yii selgeks tugevusteks on hulgalised abiklassid, mis arendaja elu lihtsamaks teevad. Seevastu raskendavaks asjaoluks on üsna pikad, kohati tuhanderealiseks kujunevad klassid, mis arendajal koodi toimimise funktsionaalsuse mõistmist raskendavad. Yii tulemusele lisatakse selles aspektis 0.9 punkti.

Autori subjektiivse hinnangu kohaselt on töö Cake raamistikuga vaadeldavate aspektide osas puudusteta. Antud raamistik nõuab arendusprotsessis arendajalt minimaalset energiat. Sellest tulenevalt ka maksimaalne tulemus sellest kategooriast.

3.4 Liidestavus

Antud töö raames pandi liidestavuse hinnangule võrdlemisi suurt rõhku ning see moodustab ka viiendiku lõpptulemusest. Liidestavuse all hinnati peamiselt REST põhist liidestuse võimekust. Peamiselt vaadeldi raamistiku tuge liideste metoodika ja funktsionaalsuse seisukohast. Maksimaalseks võimalikuks kalkuleeritud tulemuseks selles kategoorias kujunes 2.0 punkti.

Ühtlaselt ebamugav on nii Yii kui ka Zendi liidestatavus. Mõlema raamistiku puhul on see nõrgaks küljeks ning nõuab arendajalt väga suuremahulist ja tülikat konfigureerimist korrektse tööseisundi saavutamiseks. Samuti ei ole kõige paremini lahendatud mõlemal juhul liideste

adresseerimise loogika, mis on Zendil veelgi ebaloogilisem kui Yiil. Punktisummana väljendades tulemuseks 1 punkt kogutulemusse juurde.

Symfonil on liidestamine samuti keerukas, kuid siiski lihtsam kui eelnevalt nimetatud raamistikel. Paraku vajab ka Symfony siiski liigset seadistamist, et korrektselt liidestust tekitada, sealjuures näiteks JSON kujul andmete töötluse saavutamine. Symfony teenib siit kategooriast 1.2 punkti lisaks.

Nii Laravel kui Cake särasid selles kategoorias. On selge, et need raamistikud on arendatud selliseid funktsionaalsuseid silmas pidades ja liidestuses on kerged etteheited vaid Laravelile. Nimelt on Laraveli puhul ressursikulu silmas pidades keeruline vormindada väljundandmeid, kui selleks vajadus olema peaks. Sellest tulenevalt 1.8 punkti Laravelile ning maksimaalsed 2 punkti Cake raamistikule.

3.5 Testitavus

Käesolevas punktis anti hinnang raamistiku toest koodi testitavusele. Testitud kood annab arendajale ja süsteemi haldajale teatava töökindluse garantii ning seetõttu on antud aspekt üsna oluliselt hinnatud – selle kategooria eest võib saada juurde kuni kaks punkti. Üldiselt olid valitud raamistikud valdavalt piisava testitavusega.

Zend toetub testimisel vaid ühiktestidele, mistõttu ei ole kogu funktsionaalsus täielikult testitav baasrakendusele toetudes. Zendi kalkuleeritud tulemuseks siit 1.4 punkti.

Yii toetab oma testides vaid kolmanda osapoole lahendustele ning on üldiselt piisavalt kaetud. Paraku seab Yii puhul piirangud tema andmemudel, mida on erakordselt keeruline ilma korrektsete reaalsete andmeteta testides imiteerida. Yii tulemuseks 1.6 punkti.

Ülejäänud raamistikud on võrdselt eeskujuliku testimisvõimekusega ning teenivad siit kategooriast maksimumpunktid.

3.6 Turvalisus ja rollid

Käesolevas töös hinnati teoreetilisel tasemel ka turvalisuse aspekte, seda peamiselt arvesse võttes rollipõhist kasutajate grupeerimise ja õiguste jagamise loogikat. Kogutulemusest moodustab see kategooria 15%, mis teeb maksimaalseks võimalikuks tulemuseks 1.5 punkti.

Laraveli ja Cake baasrakendused siinkohal väga suurt funktsionaalsust ei paku. Rollipõhist lähenemist saab rakendada vaid kolmanda osapoolte poolt kirjutatud rakenduste abil. Arvutuslikuks tulemuseks Laravelile 0.6 punkti. Cake pakub mõnevõrra läbimõeldumat funktsionaalsust ning lisaks ka krüpteerimiseks vajalikud abiklassid, mistõttu Cake saab hindeks 0.9 punkti.

Symfony pakub üldiselt piisavalt tugevat kasutaja turvalisusega seotud komponenti, kuid probleeme tekib rakenduse versioonidega ühtlustamisel. Symfony punktidele lisab see aspekt 1.05 punkti.

Eeskujuliku turvalisuse ja rolliloogika baasfunktsionaalusega hiilgavad nii Yii kui Zend, kus antud funktsionaalsus on üsna põhjalikult läbi käidud ning piisavad, et anda arendajale võimalus süsteemi turvaliseks disainiks. Mõlemale raamistikule omistatakse siit maksimaalsed 1.5 punkti.

3.7 Lokalisatsioon

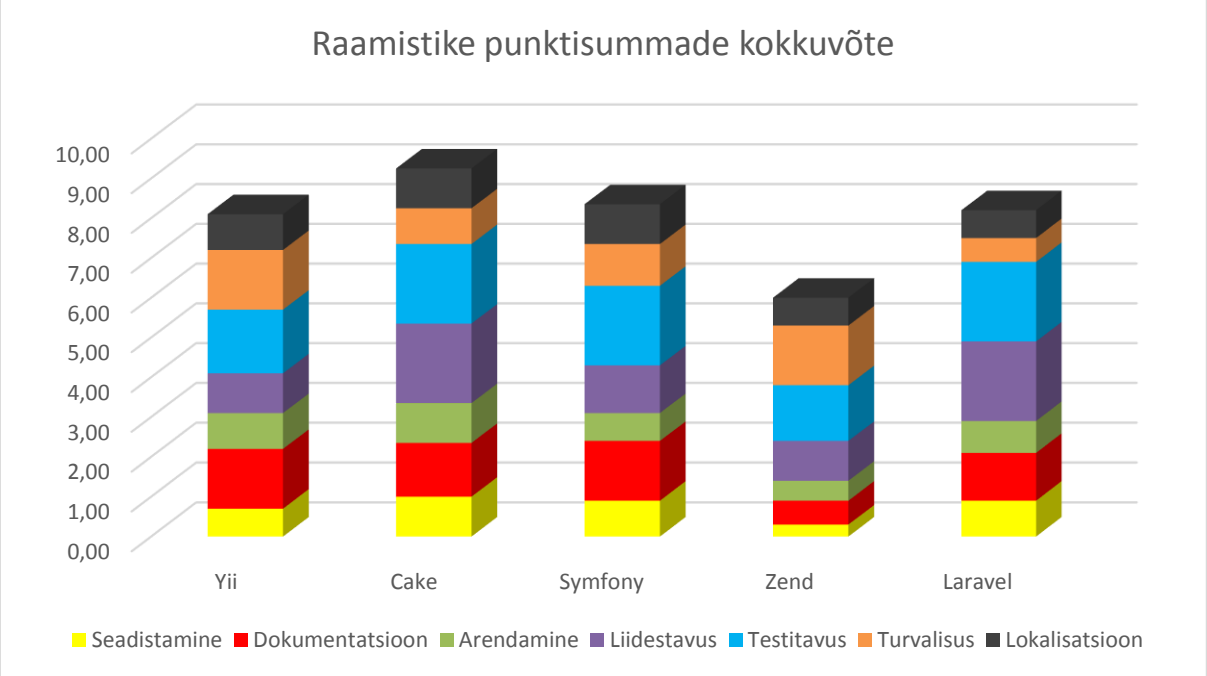
Lokalisatsiooni aspekti puhul hinnati rakenduse võimekust mitmekeelse sisuga manipuleerimiseks. Oluliseks aspektiks on ka raamistiku tugi tekitada tõlgetele halduskeskkonda, kus administreeriv kasutaja saaks läbi kasutajaliidese tõlkeid hõlpsasti muuta. Viimast näiteks võimaldasid baasfunktsinaalsusena Yii, Symfony ja Cake. Maksimaalselt võimalik siit kategooriast tulemusele lisada ühe punkti.

Vaid failisüsteemse lahenduse tõttu kõige madalamad punktid said Zend ja Laravel. Muus osas mõlema raamistiku puhul on tõlgete kasutamine ja rakendusse integreerimine mugav ja aktsepteeritavalt lahendatud, kuid halduskeskkonna tekitamiseks tuleks mõningaid muudatusi arendaja poolt sisse viia. Mõlemad raamistikud siit aspektist lisavad oma kalkuleeritud punktisummale 0.7 punkti.

Teised raamistikud, Yii, Symfony ja Cake, omasid täielikku tõlgete tuge, selle möönudsega, et Yii puhul puudub automaatne tõlke puudumise käsitlemine näiteks vaikimisi tõlke valimiseks. Sellele vaatamata on see viga võrdlemisi lihtsasti parandatav arendaja poolt. Yiile lisatakse siit 0.9 punkti, Symfony ja Cake saavad maksimaalse 1 punkti.

3.8 Analüüs

Järgnevalt on analüüsitud tulemused esitatud nii tabelite kui ka graafiku kujul. Tabelis (Tabel 3) on tulemused välja toodud jaotises 1.3 kirjeldatud valemit kasutades: Valem 1, lk 11.



Joonis 2. Raamistike punktisummade kokkuvõte.

Tabel 3. Kalkuleeritud punktisummad.

	Yii	Cake	Symfony	Zend	Laravel
Seadistamine	0,70	1,00	0,90	0,30	0,90
Dokumentatsioon	1,50	1,35	1,50	0,60	1,20
Arendamine	0,90	1,00	0,70	0,50	0,80
Liidestavus	1,00	2,00	1,20	1,00	2,00
Testitavus	1,60	2,00	2,00	1,40	2,00
Turvalisus	1,50	0,90	1,05	1,50	0,60
Lokalisatsioon	0,90	1,00	1,00	0,70	0,70
	8,10	**9,25**	**8,35**	**6,00**	**8,20**

Kõige madalama punktisummani, saades kokku 6 punkti kümnest, jõuab käesolevas töös Zend raamistik. Enim kaotas Zend konkurentidega võrreldes punkte dokumentatsiooni, arendamise ja seadistamise aspektides.

On selge, et Zend antud kriteeriumeid silmas pidades ei ole kõige sobilikum kandidaat veebiarendusega tegeleva ettevõtte tööriistaks soovitamisel. Kindlasti ei ole Zend näol tegu nõrga raamistikuga omades mitmeid tugevusi, kuid käesolevas töös käsitletud aspekte arvesse võttes jäi ta konkurentidele selgelt alla eelkõige kaasaegsuse aspektis.

Tulemustest selgub, et üsna võrdse lõpptulemusega on kolm raamistikku: Yii, Symfony ja Laravel said võrdlemisi sarnased tulemused, seejuures omades tugevad ja nõrki külgi üpriski erinevates aspektides. Yii kolmest marginaalselt nõrgeim saades 8.1 punkti. Vaid kümnendiku punktiga edestab Yii'd Laravel, saades 8.2 punkti, kes omakorda kaotab vaid 0,15 punkti Symfony raamistikule. Laraveli punktisumma kannatab enim tänu nõrgale tulemusele turvalisuse aspektis, Yii ja Symfony mõlemad jäid nõrgaks liidestuse aspektis.

Tulemustest lähtuvalt ei oleks töös käsitletud aspekte arvesse võttes ettevõttel, kes juba kasutab Yii raamisikku tarkvara arenduseks, mõistlik vahetada seda välja Symfony või Cake vastu, sest autori hinnangul uue raamistiku õppimiskõver ei kaalu üle marginaalset eelist funktsionaalsuses.

Kokkuvõttes saavutas kõrgeima punktisumma, saades 9.25 punkti maksimaalselt 10st võimalikust, Cake raamistik, millel silmnähtavalt nõrki aspekte võrdluse all olevates aspektides ei täheldatud.

Autori hinnangul on ettevõttel põhjust kaaluda Cake PHP raamistiku peale migreerimist, võttes arvesse käesolevas töös käsitletud aspekte. Siinkohal tasub ka märkimist, et Yii edestas Cake raamistikku turvalisuse ja dokumentatsiooni aspektist, mida ettevõte peaks kindlasti oma otsustamise juures arvestama.

4. Kokkuvõte

Töö põhiliseks eesmärgiks oli välja selgitada, kas veebiarendusega tegelev ettevõte kasutab optimaalset ja kaasaegset tööriista PHP raamistiku näol oma loodavate süsteemide arendusprotsessis. Selle välja selgitamiseks oli vaja kaardistada kriteeriumid, mille abil võrdlusprotsessi üles ehitada ning seejärel välja selgitada sobivad kandidaadit töö valimiks. Peale sobivate kandidaatide leidmist analüüsiti raamistikke valitud kriteeriumite alusel.

Käesolevas töös raamistike võrdlemise hinnatavateks kriteeriumiteks osutusid seadistamine ja generaatorid, dokumentatsioon, arendamine ja õppimiskõver, liidestuse tugi, lokalisatsioon, testitavus ja turvalisus. Välistati vähese kasutajaskonnaga ning sellised raamistikud, kus puudus korrektne MVC struktuur ja moduleerimise võimekus. Raamistike valikul oli väga suureks kaaluks tema populaarsus ning valikul mängis mõningat rolli ka autori enda teadlikkus eri raamistikest. Lisaks kasutati otsinguportaali abi leidmaks raamistikke, mis sobiksid valimisse.

Levinuimateks käesolevasse töösse sobivateks raamistikeks osutusid Yii, Symfony, Laravel, CakePHP, Zend, Nette ja Phalcon, milledest viimased kaks, Nette ja Phalcon, välistati peamiselt vähese kasutajaskonna tõttu. Teistega tehti ka praktiline analüüs primitiivse rakenduse arendamise näol, eelkõige eesmärgiga anda hinnang raamistiku eri aspektidele.

Antud tulemustest selgub, et valitud kriteeriumeid arvesse võttes on Cake PHP selge ülekaaluga saades kokku 9.25 punkti võimalikust 10st sobilikuim valik. Ühtlasi selgub, et vaatluse all oleva veebiarendusega tegeleva ettevõtte tööriistaks olev Yii ei pruugi olla optimaalseim töövahend. Olgugi, et käesolevas töös käsitleti nii teoreetilist kui ka praktilist analüüsi raamistike võrdluseks, seisneb autori hinnangul käesoleva töö väärtus eelkõige eri aspektide teoreetilisel võrdlusel. Pealtnäha sarnast uuringut on tehtud ka varem, kuid näiteks Zurkiewicz ja Milosz oma raamistike võrdluses, hinnates pisut teisi aspekte, muuhulgas ka jõudlust, jõudsid järeldusele, et Yii ja Zend on parimad raamistikud [10]. Käesolevas töös välja selgitatud tulemused peaksid eelkõige olema abiks ettevõtte raamistiku valikul ning autor soovitab siiski ettevõttel sooritada prooviprojekt lõpliku otsuse tegemisel ja lähtuma valikul siiski vajaduspõhiselt.

Autor näeb mitmeid võimalikke käesoleva töö edasiarenduse võimalusi. Paraku ei mahtunud käesoleva töö skoopi ei praktiline turvalisuse aspekti võrdlus ega ka erinevate raamistike jõudluse võrdlus – mõlemad toodud aspektid väärivad mahult eraldiseisvat uurimist, mistõttu ka käesoleva töö skoobist välja jäeti.

Käesoleva töö eesmärgiks oli kaasaegse PHP raamistiku leidmine veebiarendusega tegeleva ettevõtte jaoks ning eesmärk ka saavutati. Selgus, et ettevõttes kasutusel olev Yii raamistik on iseenesest küllalt kaasaegne tööriist, kujunes käesolevast tööst välja teoreetiliselt sobilikum raamistik Cake PHP näol.

Summary

The main goal of this thesis was to find out, whether a certain web development company is using a modern and optimal PHP framework for developing. Firstly was to map the criteria to be used when assessing the different frameworks. Secondly the caandidates for the frameworks were selected, and finally all the results were analizsed using the mentioned criteria.

The main aspects for assessing the frameworks were found as follows: configuration and scafolding, documentation, ease of development and learning curve, web services, localization, testability and security. Popularity of the framework played a significant role in the selection of the framework, and some affect had authors knowlegde about existing frameworks. A search engine was also used to find suitable frameworks.

The most suitable and popular frameworks found in this thesis were Yii, Symfony, Laravel, CakePHP, Zend, Nette and Phalcon, from whitch the last two, Nette ja Phalcon, were not further investigated due to their lack of active users. With the other frameworks some practical work was made to help assess the aspects named earlier.

Having analized the results, Cake PHP, with a total score of 9.25/10, was found to be the most suitable framework to be used in web development for the selected criteria. Also turned out that Yii, the framework used by the company in subject, was not the most optimal tool to be used. The results in this thesis can be used in order to find a suitable framework for developing a web application but the author strongly advises the developers to test the framework on a sample project first, to make sure it fits all the needs.

The main goal of the thesis „The comparison of PHP frameworks following the example of a web development company“ was to find the most suitable modern PHP framework for a certain web development company and the goal was achieved. The author suggests Cake PHP to be the most optimal tool for said company.

Kasutatud kirjandus

[1] „Introduction - Composer,“ Composer, [Võrgumaterjal]. Available: https://getcomposer.org/doc/00-intro.md. [Kasutatud 14 4 2016].

[2] C. Lee, Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in, Ohio: The Ohio State University, 2012.

[3] „SitePoint,“ 2015. [Võrgumaterjal]. Available: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/. [Kasutatud 20 märts 2016].

[4] „linkedIn,“ [Võrgumaterjal]. Available: https://www.linkedin.com/pulse/7-best-php-frameworks-2015-winspire-web-solution. [Kasutatud 2 aprill 2016].

[5] „Getting Started With FOSUserBundle,“ Symfony, [Võrgumaterjal]. Available: http://symfony.com/doc/current/bundles/FOSUserBundle/index.html. [Kasutatud 27 3 2016].

[6] „Github,“ [Võrgumaterjal]. Available: https://github.com/DynamicCodeNinja/RBAC. [Kasutatud 1 4 2016].

[7] CakePHP, „Containable,“ [Võrgumaterjal]. Available: http://book.cakephp.org/2.0/en/core-libraries/behaviors/containable.html. [Kasutatud 4 11 2016].

[8] The PHP Group, „PHP.net,“ [Võrgumaterjal]. Available: http://php.net/eol.php. [Kasutatud 11 4 2016].

[9] „Database and models,“ Zend Framework, [Võrgumaterjal]. Available: http://framework.zend.com/manual/current/en/user-guide/database-and-models.html. [Kasutatud 11 4 2016].

[10] M. M. Adrian Zurkiewicz, SELECTING A PHP FRAMEWORK FOR A WEB APPLICATION PROJECT — THE METHOD AND CASE STUDY, Lublin, 2015.