TARTU ÜLIKOOL

MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut

Informaatika eriala

Martin Reilent

Automaatne veebilehtede veebipood

Bakalaureusetöö (6 EAP)

Juhendaja: Vambola Leping

TARTU 2014

Automaatne veebilehtede veebipood

Lühikokkuvõte:

Käesoleva bakalaureusetöö tulemusena valmis automaatse veebilehtede veebipoe rakenduse prototüüp, mis teeb lihtsaks tavakasutajatele või võhikutele personaalse kodulehe loomise ja majandamise. Kasutaja saab kerge vaevaga üles seada oma Wordpressi paigalduse ning selle teemasid vahetada. Samuti on olemas võimalus uusi teemasid osta, makstes nende eest pangalingiga. Töö käigus kasutati kasutati prototüübi loomisel erinevaid meetodeid ja tehnoloogiaid, mida antud dokument kirjeldabki.

Võtmesõnad:

Laravel, Bash, PHP, Javascript, Bootstrap, pangalink, veebileht, Wordpress, WP-CLI, Less

Automatic online website store

Abstract:

As a result of this Bachelor thesis, a prototype application for an automatic website webstore was created, which will make it easier for regular people, or people who are unfamiliar with the process of creating a website, to set up and manage their own personal web page. The user can easily install their Wordpress application and also change its themes. There is also a possibility to purchase new themes by paying for them using a bank link. While creating this thesis, different methods and technologies were used to create the prototype, which will be discussed in this document.

Keywords:

Laravel, Bash, PHP, Javascript, Bootstrap, Banklink, webpage, Wordpress, WP-CLI, Less

Sisukord

Sissejuhatus

Tänaseks on internetiavarustes väga populaarseks saanud personaalsed kodulehed või blogid, kuid alati ei oma kõik inimesed vajalikke oskuseid, et oma lehekülg iseseisvalt püsti seada. Muidugi saab alati kasutada tasuta teenuseid nagu blogger.com, kuid see pole see mis päris enda kodulehekülg ning ei paku kasutajale piisavalt võimalusi lehe kohandamiseks. Antud bakalaureusetöö eesmärk ongi disainida rakendus ja luua prototüüp, millega tavakasutajad ja võhikud saavad kergesti oma lehekülje üles seada, paigaldust muuta ning seda kõike ühest kohast ja korraga.

Käesoleva töö käigus valmis prototüüp sellisele rakendusele, mille erinevaid tehnoloogiaid, võimalusi ning loomisprotsessi antud töö kirjeldabki. Töös antakse ka põgus ülevaade olemasolevatest lahendustest ja teenustest selles vallas, mis on potentsiaalsetele kasutajatele kergesti leitavad.

Järgmisena kirjeldatakse rakendust üldiselt, mis annab lugejale esmase üldmulje milline valmis rakendus peaks välja nägema. Kolmandas peatükis kirjeldatakse rakenduse prototüübi serveri poolset osa, kus räägitakse erinevatest tehnikatest ja tehnoloogiatest, mida prototüübi valmimisel kasutati ning neljandas peatükis räägitakse rakenduse kasutajapoolsest osast, mille põhirõhk jääb sellele, mida kasutaja rakendust kasutades näeb.

1 Taust ja olemasolevad lahendused

Praeguse seisuga leidub Eestis palju erinevaid firmasid, kes tegelevad veebilehtede valmistamisega. Kasutades Google otsingumootorit ja otsides märksõnu “veebilehed”, “kodulehed”, “veebilehtede valmistamine” jne leiab mitmete lehekülgede kaupa tulemusi.

Tavakasutaja vaatenurgast ei ole aga need teenused kõige optimaalsemad ja kergesti kasutatavad, kuna enamasti pakutakse nii-öelda rätseptööna loodavaid lahendusi. Rätseptööna tähendab, et vastavalt kliendi soovidele luuakse disain, pannakse paika funktsionaalsus ja tehakse veebileht valmis alustades nullist. Selline tegevus võib aga kliendi (tavakasutaja) jaoks kujuneda äärmiselt aeganõudvaks ja kalliks, sest enamuste projektide korral ei tea kliendid alguses ise ka, mida nad täpselt leheküljelt ootavad ja kuidas see lõpuks nende brauseriaknas välja nägema hakkab. See aga omakorda lisab koormust arendajatele ja muudele projektiga seotud inimestele, kelle töötundide arv pidevalt suureneb.

Siinkohal võiks näitena välja tuua kaks firmat mis Google-s märksõna “veebilehed” tulemusena esimestena silma hakkasid, Interlex Media (http://www.interlexmedia.ee) ja Ultima Design (http://www.ultimadesign.net).

Interlex media kodulehe sõnul luuakse kõik nende lehed rätseptööna ning kasutatakse järgmist loogikat:

1. Kliendi mõistmine
2. Disain
3. Arendus
4. Järeltugi

Kogu protsess peaks alguse saama sellest, et klient võtab nendega ise ühendust, mille peale tehakse kliendile hinnapakkumine.

Ultima Design on sarnane, lehed tehakse rätseptööna samal põhimõttel nagu eespool mainitud firmal. Ultima Design kodulehel on veel lisaks ka välja toodud pakettide hinnad, mis algavad 220.- euroga ja lõppevad 3000.- euroga. Ka Ultima Design puhul peaks lahenduse

väljatöötamine algama sellest, et klient võtab nendega ühendust ning seejärel tehakse kliendile hinnapakkumine ja kui klient sellega nõustub, algab arendus ja disain.

Kui klient telliks veebilehe selliste kanalite kaudu, oleks tal veel kindlasti vaja mõelda ka domeeninime ja serverimajutuse üle. Domeeninime registreerimine ja serverimajutuse tellimine tähendab aga omakorda kulutusi ja nõuab kliendilt aega.

2 Rakenduse kirjeldus

Eelmainitud probleemide lihtsustamiseks otsustas autor luua veebirakenduse, mille abil saab klient lihtsalt arusaadavas keskkonnas endale vajalikud tooted ja teenused valida. Tooted jagunevad järgmiselt: veebirakendus (e-pood, internetilehekülg), disain, logo. Teenusteks on servermajutus, mailiserver ja domeeni majutamine. Rakenduse eesmärk on pakkuda kliendile *out-of-the-box* lahendust arvuti tagant lahkumata. Eelkõige on klientideks väikefirmad ja eraisikud, kes rätsepatööna tellitavat kodulehte ei soovi või seda kalliks peavad.

Põhilised aspektid millele keskendutakse:

- Rakenduse disain - minimalistlik, kuvatakse vaid olulist informatsiooni, lihtne kasutus.
- Kasutusmugavus - seotud disainiga, et kasutajal oleks võimalikult lihtne aru saada sellest, kuidas teenus töötab ilma lisamaterjale lugemata. Suurel osal keerukate veebikeskkondade probleem on selles, et kasutajad ei orienteeru rohke info keskel ja see muudab kasutamise ebamugavaks.
- Kiirus - kasutaja peaks saama 15-30 minutiga oma teenused/tooted valitud ja maksed sooritatud.
- Paindlikkus - teenus on muudetav ja integreeritav konkreetse isiku vajaduste järgi. Kõik on hallatav ühes administreerimisliideses.

Tegemist on täisautomaatse kommertsliku rakendusega, mille valmistoode sisaldab endas kõike: arvete saatmist, seadistamist, kasutajakontosid jne. Lõpprakendus valmib mitmekeelsena, st on mõeldud rahvusvahelisele turule. Programmeerimine toimub põhiliselt PHP, JavaScript, HTML5, AJAX tehnoloogiates. Serveri poole pealt kasutatakse ka UNIX-i serveri süsteeme ja rakendusi ning MySQL-i. Rakenduse peamiseks mootoriks saab olema vabavaral baseeruv ja maailma enimkasutatud sisuhaldustarkvara Wordpress (http://wordpress.org).

Rakendus on mõeldud eelkõige võhikutele, keda ei huvita see kuidas süsteemid töötavad, peaasi, et kõik töötaks ja kiiresti valmis saaks. Sellisel juhul ei ole vaja kliendil pöörduda mingite kolmandate osapoolte poole, nt veebimajutuse, domeeninime otsimisega, vaid saab kõik vajalikud protseduurid ära teha ühest kohast ilma liigsete jõupingutusteta.

Kliendi poolt vaadatuna peaks protsess välja nägema selline:

1. Klient valib endale sobiliku teema, (disaini) mida oma valmislehel kasutada soovib.
2. Valib omale sobiliku logo, mida lehel kasutada tahab.
3. Valib domeeninime, mille saadaval olekut kontrollitakse samas kohapeal.
4. Kui kõik eelnev on edukas, siis suunatakse klient maksmise piirkonda, kus peale makse sooritamist luuakse kliendile konto, millega hiljem haldusliidesesse sisse logida saab.
5. Veebileht installeeritakse serverisse ja kliendile saadetakse andmed, kuidas ta oma uuele lehele sisse logida saab jne. Samuti saadetakse ka arve tehtud tehingu eest.

3 Rakenduse serveri pool

3.1 PHP

PHP ehk "*PHP: Hypertext Preprocessor*" on serveripoolne skrptimiskeel, mida kasutatakse suuresti just veebirakenduste loomisel, kuid ka üldise programmeerimiskeelena. PHP on avatud lähtekoodiga ning seda on võimalik kasutada otse HTML koodi sees. PHP süntaks on saadud C, Perl ja Java programmeerimiskeelte abil ning on väga lihtsasti õpitav [1]. 2013 aasta jaanuari seisuga on PHP kasutusel 244 miljonis veebilehes [2].

On olemas kolm põhilist valdkonda, kus PHP skriptimiskeelt kasutatakse [3]:

- Serveripoolne skriptimiskeel. Kõige traditsioonilisemalt ja enimkasutatavalt kasutatakse php-d serveri poolse skrtiptimskeelena. Selleks on vaja kolme asja: PHP parserit, veebiserverit ja veebilehitsejat. PHP programmi väljundit saab näha otse veebilehitsejast. Seda kõike saab kasutada ka oma lokaalses masinas.
- Käsurea skriptimiskeel. PHP-d saab jooksutada ka ilma serveri või veebibrauserita. Ainuke asi, mida vaja läheb, on PHP parser. Sellist tüüpi kasutus on ideaalne näiteks selliste scriptide jaoks, mis kasutavad cron-i (Cron on programm, mis UNIX tüüpi operatsioonides lubab kasutajal käske või scripte jooksutada automaatselt mingil teatud ajal [4]).
- Graafiliste programmide tegemine. PHP ei ole arvatavasti kõige parem ja levinum keel graafilise kasutajaliidesega programmide tegemisel, kuid kui PHP-d hästi osata ja mõnda tõhusat funktsiooni kasutada, siis saaks ka sellega hakkama. Sellise tegevuse jaoks on välja töötatud PHP-GTK laiend (http://gtk.php.net)

Käesolevas rakenduses kasutatakse PHP-d just tema lihtsuse ja paindlikkuse pärast ning samuti ka seepärast, et PHP-l on väga palju laiendeid ja võimalusi. On kirjutatud palju erinevaid raamistikke ja teeke, mis veebiarendust tunduvalt lihtsustavad.

3.2 Laravel

Laravel on tasuta, avatud lähtekoodiga PHP veebirakenduse raamistik, mis jääb silma just oma elegantse süntaksiga. Laraveli loojad usuvad, et arendamine peaks olema nauditav, et lõpptulemus korralik tuleks. Laravel püüab võimalikult lihtsaks teha selliste levinud veebiprojektide osad nagu autentimine, ümber suunamine ja sessioonid. Laravel on siiani üritanud kombineerida parima sellest, mida teised raamistikud nagu Ruby rn rails, ASP.NET MVC ja Sinatra on pakkunud.

Raamistikku põhilised omadused ja omapärad: [7]

- Sisseehitatud Eloquent ORM, mis kujutab endast süsteemi, millega on kerge andmebaasiga töötada. Igale andmebaasi tabelile vastab rakenduses mudel, mida kasutatakse tabeliga töötamisel [6]
- Migratsioonid, mida kasutatakse andmebaasi skeemide versiooni haldusena.
- Sisse ehitatud "*unit testid*", mida saab käivitada läbi käsurea tööriista.
- PHP klasside automaatne kaasamine, et poleks vajadust neid manuaalselt lisada .
- Marsruudid
- MVC ehk mudel-vaade-kontroller arhitektuur.

Käesolevas rakenduses kasutatakse Laraveli raamistikku põhjana, mille peal ülejäänud funktsionaalsus üles ehitatakse.

Veebiarendajad suhtlevad Laraveliga tavaliselt läbi käsurea tööriista "Artisan", mida kasutatakse Laraveliga loodud tarkvaras väga tihti. Artisani kasutatakse migratsioonide loomisel, andmebaasi täitmisel (database seeding), raamistiku *autoload* failide genereerimisel jms. [12]

Artisani kasutamise süntaks on "php artisan [käsk] [argumendid]". Käsu käivitamiseks peaks praegune töökataloog olema sama, mis rakenduse juurkataloog.

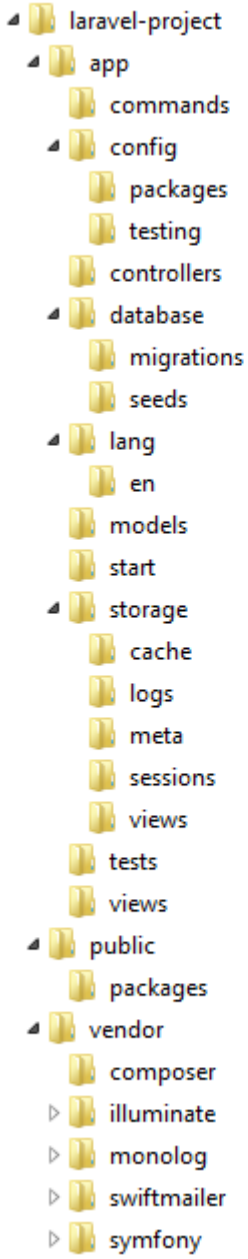
Mõned tähtsamad Artisani käsud:[13]

Käsk	**Kirjeldus**
dump-autoload	Genereerib raamistiku automaatse laadimise failid.
env	Näitab raamistiku praegust keskonda.
list	Kuvab nimekirja võimalikest käskudest.
migrate	Jooksutab andmebaasi migratsioonid.
routes	Kuvab rakenduses kasutusel olevad marsruudid.
tinker	Käivitab PHP käsurea kesta, millega saab käsurealt rakendusega töötada.
migrate:make	Loob uue migratsiooni faili, mille klassinimeks on käsu argument.
migrate:reset	Tühistab kõik andmebaasi migratsioonid.
migrate:rollback	Tühistab viimase andmebaasi migratsiooni.
db:seed	Paljundab andmebaasi admetega.
controller:make	Loob uue kontrolleri faili kausta “app/controllers”, mille klassi- ja failinimeks on käsu argument.

Tabel 1. Tähtsamad tööriista „Artisan“ käsud

Artisan pakub veel võimalust lisada käskude juurde --env argument, mis täpsustab, millises keskkonnas käsku jooksutama peaks. Näiteks käsu “php artisan migrate --env=Staging” korral jooksutatakse käsku “Staging” keskkonnas, mis on eelnevalt defineeritud failis “bootstrap/start.php”

Laraveli struktuur ise on üles ehitatud nii, et raamistiku ülesseadmisega kaasneks võimalikult vähe konfigureerimist, samal ajal kui paljud Java, Pyhon-i ja teised PHP raamistikud vajavad põhjalikku seadistamist. Laraveli puhul piisab vaid kõigest mõnest reast PHP koodist. Selline rakenduse struktuur teeb kergeks arendajatele koodi ülevaatamise, kuna struktuur on sama kõigis Laraveliga arendatud projektides. (Vaata Lisa 3)



Pilt 1. Laraveli projekti kaustade struktuur

Nagu eelnevalt pildilt näha on, siis standardstruktuur koosneb paljudest alamkaustadest, mis alguses võib tunduda natuke koormav, kuid sellise ehitusega tagab Laravel selle, et sinu projekt oleks organiseeritud “Laraveli moodi”, mis muudab koodi lugemise oluliselt paremaks ja lihtsamaks ka teistele arendajatele. [12]

Mõned enimkasutatavate kaustade kirjeldused:

Kaust	**Kirjeldus**
/app	Sisaldab endas kontrollerite, vaadete ja modelite kirjeldusi. Enamus rakenduse koodist asub selles kaustas.
/public	Ainuke kaust mis on nähtav maailmale. Seda kausta tuleks kasutada veebiserveri konfiguratsioonis. Sisaldab endast faili index.php, mis paneb tööle terve rakenduse enda. Samuti hoitakse selles kaustas kasutajale nähtavaid faile nagu CSS, javasctipt, pildid jms.
/vendor	Siin kaustas hoitakse erinevaid kolmanda osapoole faile. Sisaldab endas Laraveli enda lähteteksti ja sõltuvusi ning teisi pakette, mis laiendavad rakenduse funtsionaalsust.
/app/config	Kasutatakse rakenduse käitusaegse konfiguratsiooni hoidmiseks. Hoitakse erinevate keskkondade konfiguratsioone. Näiteks “Staging” keskkonna jaoks oleks andmebaasi sätted failis “/app/config/staging/database.php” ja “local” keskkonna jaoks failis “/app/config/local/database.php”.

/app/database/migrations	Koosneb PHP klasside failidest, mida kasutatakse sümbioosis Artisani käskudega andmebaasi tabelite genereerimisel.
/app/database/seeds	Koosneb PHP klasside failidest mida kasutatakse koos Artisani käsuga “db:seed”, mille käigus käivitatakse selles kaustas paiknevad failid.
/app/models	Sisaldab endas mudelite klasse, mis esindavad andmebaasi informatsiooni ja reegleid selle informatsiooniga töötamiseks. Enamus kordadel väljendab iga mudel rakenduse andmebaasi tabelit.
/app/controllers	Sisaldab kontrollerite klasse, mida kaustatakse mudelitega töötamisel ja vaadete laadimisel.
/app/views	Koosneb mallidest, mida kasutatakse kontrollerite või marsuutide poolt.

Tabel 2. Laraveli raamistiku kasustade kirjeldused.

Terve rakenduse ulatuses on kasutajatele teadete edastamiseks kasutatud Edvinas Kručase poolt loodud paketti nimega “Notifications”. See Laraveli raamistiku laiendus lubab mugavalt kõikjal koodis väljastada kasutajale teate, mis kuvatakse järgmise lehe laadimisega. Teated salvestakse kõik php $_SESSION muutujasse. $_SESSION on PHP massiiv, mis sisaldab endas sessiooni muutujaid, mis jooksva skripti jaoks kasutatavad on.[25] Et määrata vaadetes koht, kus teateid näidata, lisatakse vastavasse kohta lähtekoodis “Notification::showAll()” meetod, mis väljastab lehe laadmisel eelnevalt lisatud teate. Teate lisamiseks kasutatakse meetodeid success(), info(), warning() ja danger(). Näiteks lisatakse õnnestunud sisselogimisel teade “Notification::success('You have logged in successfully')”. Kõik teated tagastatakse Bootstrapi kasutajaliidese raamistikule sobivas kujus CSS-i klassidega “alert alert-<meetodi-nimi>”, mis tagab teadete elegantse disaini. Bootstrapist lähemalt on juttu allpool. [24]

Samuti on terve rakenduse ulatuses kasutusel Laraveli laiendus “LaravelShoppingcart”, mis on loodud Rob Gloudemansi poolt. Igale kasutajale loodakse õnnestunud sisselogimise korral oma nn “ostukäru” eksemplar, mis on kasutajanimega samanimeline ning seda jagatakse kõikide rakenduse kontrollerite ja vaadete vahel.
Mõned ostukäru meetodid, mida kasutatakse: [26]

Meetod	Kirjeldus
Cart::instance(kasutajanimi)->add(toote id, toote nimi, toote arv, toote hind)	Lisab ostukorvi toote.
Cart::instance(kasutajanimi)->remove(rea id)	Eemaldab ostukorvist toote, kus rea id on ostukorvi enda sisemine id, mille saab leida cart::search() meetodiga.
Cart::instance(kasutajanimi)->content()	Tagastab ostukorvi massiiv.
Cart::instance(kasutajanimi)->destroy()	Kustutab kõik tooted ostukorvist.
Cart::instance(kasutajanimi)->search(massiv)	Otsib etteantud massiviga ostukorvist tooteid, leides tagastab ostukorvi rea id.

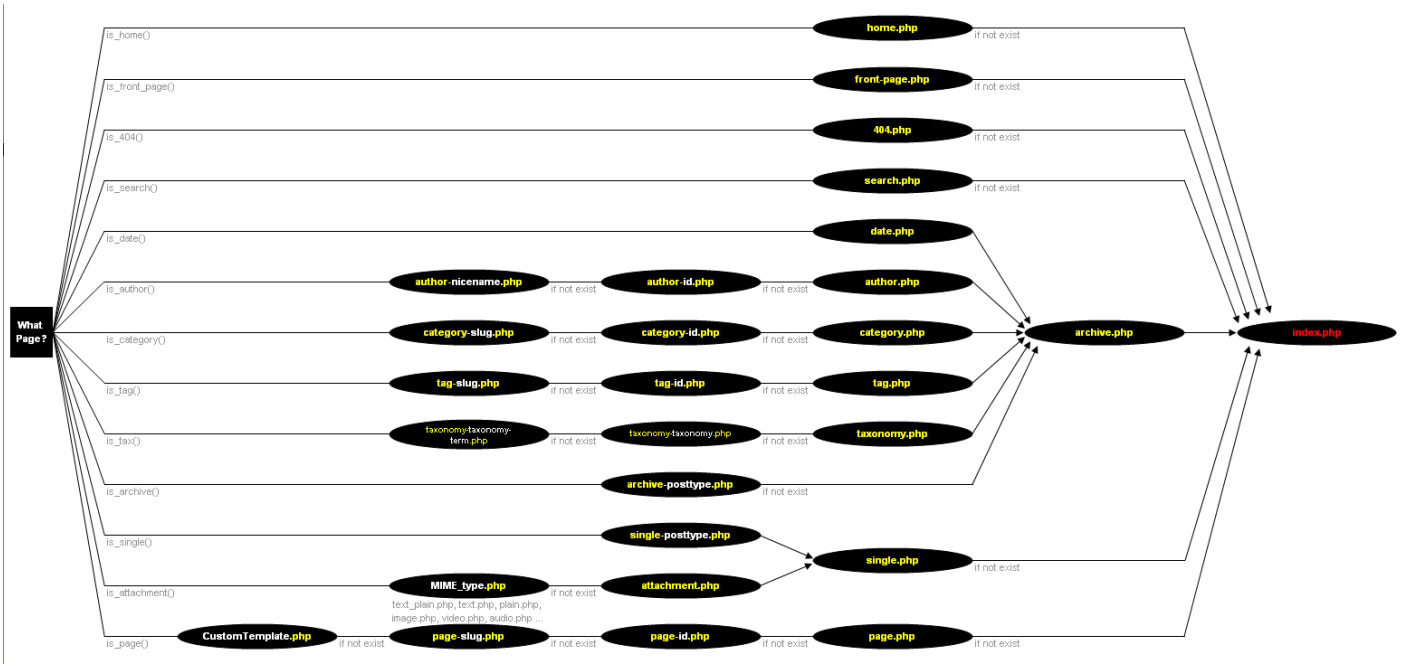
Tabel 3. Laraveli laienduse „LaravelShoppingcart“ meetodid ja nende kirjeldused.

3.3 Wordpress

Wordpress on tasuta ja avatud lähtekoodiga blogimise ja sisuhalduse tarkvara, mis baseerub PHP ja MySql-i peal. Wordpress on kõige populaarsem sisuhalduse tarkvara 05.05.2014 seisuga. [8]

Antud rakenduses kasutatakse klientide veebilehtede põhjana Wordpressi, mille konfiguratsioon genereeritakse Laraveli kaudu. Lõpptoode kujutab endast valmis Wordpressi lehekülge, millesse saab klient sisse logida ja oma artikleid või lehti lisada ja muuta.

Käesolevas rakenduses kasutatakse Wordpressi peamiselt just teemade pärast. Teemad (ing "Themes") on Wordpressiga loodud veebilehel väga tähtsal kohal, millega saab lihtsalt muuta lehekülje väljanägemist ilma sisu muutmiseta. [10] Seega on kasutajatel (klientidel) täielik kontroll oma veebilehe väljanägemise üle.



Pilt 2. Wordpressi teemafailide hierarhia.

3.4 WP-CLI

Käesolevas rakenduses on tähtsal kohal käsurea tööriist WP-CLI (Command line interface for WordPress), mida kasutatakse uue Wordpressi saidi konfiguratsiooni faili loomisel, andmebaasi loomisel ja eemaldamisel, saidi enda installeerimisel ning teemade vahetamisel ja lisamisel [14]. Samuti kasutatakse WP-CLI tööriista, et Wordpressi paigalduses kindlaks teha millised teemad aktiivsed ning installeeritud on.

WP-CLI käsud, mida rakenduses kasutatakse:

Käsk	**Kirjeldus**
wp core config	Genereerib wp-config.php faili. Parameetrid: --dbname - andmebaasi nimi --dbuser - andmebaasi kasutajanimi --dbpass - andmebaasi kasutaja parool --dbhost - andmebaasi
wp db create	Loob andmebaasi nagu kirjeldatud wp-config.php failis, mis genereeritakse wp core config käsuga.
wp db drop	Kustutab andmebaasi nagu kirjeldatud wp-config.php failis. Parameetrid: --yes - Käsk käivitatakse ilma kasutaja kinnituseta.
wp core install	Loob andmebaasi WordPressi tabelid. Parameetrid: --url - Uue saidi aadress --title - Uue saidi esialgne nimi, mida kasutajal on võimalus hiljem ise muuta. --admin_user - administraatori kasutajanimi

	--admin_password - administraatori parool --admin_email - administraatori email
wp theme activate <teema>	Aktiveerib teema. Parameetrid: <teema> - Teema kausta nimi. (wp-content/themes kaustast).
wp theme install <teema_url>	Installeerib teema. Parameetrid: <teema_url> - Absoluutne rada teema arhiivifailini. --activate - Olemasolul aktiveeritakse tema kohe peale installeerimist.

Tabel 4. Rakenduses kasutavate WP-CLI käsud ja nende kirjeldused.

Kõigi mainitud käskudega koos kasutatakse parameetrit “--path”, millele antakse väärtuseks saidi juurkataloogi absoluutne rada, seda sellepärast, et töökataloog, millest WP-CLI käske käivitatakse, ei ole sama, mis WordPressi saitide juurkataloog.

3.5 Bash script

Rakenduses kasutatakse WordPressi saitide installeerimisel WP-CLI käske, kuid need on omakorda kirjeldatud kolmes UNIX Bash skriptis - installeerimine, eemaldamine ja teemavahetus. Bash on Unixi käsukest, mis avaldati 1989 aastal ning loodi asenduseks Bourne'i käsukestale. Bash on GNU, Linuxi ja Mac OS X-i vaikekäsukest. Bash on käsuprotsessor, mis tavaliselt töötab tekstiaknas ja mis lubab kasutajal sisestada erinevaid käske. Lisaks sellele suudab Bash käske lugeda ka failist, mida kutsutakse skriptiks. [16]

Kõigi kolme skripti korral salvestatakse iga käsu väljund logifaili, mis asub kaustas “/logs”. Vastavalt käivitatud skriptile on salvestatava faili nimeks “<skriptinimi>-LOG-<päev><kuu><aasta>_<tund><minut><sekund>.log”, näidates nimes aega, millal skript käivitati.

3.6 Pangalink

Rakenduse prototüübis on kasutatud SEB Panga pangalinki. Rakenduse lõppversioonis saab olema toetatud ka teiste Eesti pankade nagu Nordea, Swedbank jms pangalingid. SEB pangalink on installeeritud eelkõige sellepärast, et SEB Pangas jookseb lisaks päris pangakeskkonnale ka testbaas, millega on hea rakendust testida ilma selle eest raha tasumata.

Pangalingi tööle saamiseks on vaja panga poolt saadud avalikku võtit ja enda privaatvõtit. Panga avaliku võtmega kontrollitakse panga poolt saadud vastuseid ja oma privaatvõtmega allkirjastatakse kõik andmed, mis panka saadetakse. Sarnaselt peab pangal olemas olema kaupmehe enda avalik võti, et kaupmehelt tulevaid päringuid kontrollida. [29] Nii pangast tulevate kui panka saadetavate andmete korral tuleb kokku panna kõigi väljade põhjal (Vaata Lisa 1 ja Lisa 2) kontrollsumma, mida kasutatakse andmete allkirjastamisel/valideerimisel.

Kontrollsumma (VK_MAC) leitakse järgmiselt:[28][29]

1. Iga päringu välja jaoks vaadatakse selle välja väärtuse pikkust.
2. Iga välja pikkuse jaoks koostatakse kolmekohaline number. Näiteks kui VK_SND_ID = “martintest”, siis selle välja pikkus väljendatakse kui 010.
3. Välja pikkusele liidetakse järgi tema väärtus. Antud näite korral siis “010martintest”.
4. Kõikide väljade väärtused ja pikkused liidetakse kokku, saades ühe pika sõne.
5. Saadud sõne allkirjastatakse/kontrollitakse vastavalt teenusele kaupmehe privaatse võtmega/panga avaliku võtmega

Hetkel kasutab rakenduse pangalink vaid teenuseid koodidega 1001 ja 1101, mis tähendavad vastavalt “Teenindaja saadab Panka Kliendi sooviavalduse Tehingu tegemiseks” ja “Vastus Tehingu aktsepteerimise kohta”, kui pangalt tuleb vastus mõne muu koodiga, siis antakse kasutajale veateade.[28]

4 Rakenduse kasutajapool

4.1 Bootstrap

Bootstrap on kasutajaliidese raamistik, mis kaasab endas tasuta töövahendeid veebilehtede ja veebirakenduste loomiseks. Bootstrap sisaldab HTML ja CSS baasil loodud tüpograafiat, forme, nuppe, navigatsioonielemente ja palju teisi erinevaid komponente. Lisaks sellele on võimalik kasutada ka javascripti laiendeid, et loodud rakendus veelgi dünaamilisemaks teha. Bootstrap on GitHub-is 1. kohal olev projekt, millel on üle 66,000 tähe ja 24,000 harutamise [17] ning on kasutusel isegi NASA poolt [18].

Raamistik on ühilduv kõigi suuremate brauserite viimaste versioonidega, nagu näiteks Mozilla Firefox, Safari, Opera jms, kuid ei ole kasutatav vanemate brauseritega, nagu Internet Explorer 8. Alates 2.0 versioonist toetab raamistik ka nö “tundlikku” disaini, mis tähendab, et veebilehtede kujundus kohaneb dünaamiliselt vastavalt sellele, millist seadet parasjagu veebilehe vaatamiseks kasutatakse. Antud bakalaureusetöö rakenduses kasutatakse samuti tundlikku disaini.

Bootstrap kasutab lehekülje kompositsiooni loomisel võrgustiku süsteemi, mis koosneb ridadest ja veergudest, millele lisatakse sisu. Järgnev tabel demonstreerib, kuidas võrgustiku süsteem erinevate seadmetega tööle hakkab. [19]

	Ekstra väikesed seadmed nagu Telefonid (laius < 768px)	**Väikesed seadmed nagu Tahvelarvutid (laius >= 768px)**	**Keskmised seadmed nagu Arvutid (laius >= 992px)**	**Suured seadmed nagu Arvutid (laius >= 1200px)**
Võrgustiku käitumine	Alati horisontaalne.	Alguses varisenud, horisontaalne murdepunktides.	Alguses varisenud, horisontaalne murdepunktides.	Alguses varisenud, horisontaalne murdepunktides.
Konteineri laius	Puudub	750px	970px	1170px

	(automaatne)			
Css klassi eesliide	.col-xs-	.col-sm-	.col-md-	.col-lg-
Veergude arv	12	12	12	12
Veeru laius	Automaatne	60px	78px	95px
Veergude vahe	30px	30px	30px	30px

Tabel 5. Tabel näitamaks, kuidas Bootstrapi võrgustiku süsteem töötab erinevate seadmetega.

Võrgustiku klassid rakendatakse seadmetele, mille ekraani laius on suurem või võrdne üleval pool kirjeldatud murdepunkti suurustele ja ignoreeritakse võrgustiku klasse, mis on mõeldud väiksemate seadmete jaoks. Näiteks kui lisada veerule klass ".col-md-", siis mõjutab see keskmiste suurustega seadmete vormingut ning samas ka suurte seadmete vormingut, kui klass ".col-lg-" ei ole määratud.

Käesolevas rakenduses kasutatakse lisaks võrgustiku süsteemile veel navigatsiooni, tüpograafia, teadete, formide, nuppude, modali (dialoogiaken, millega kasutaja peab enne vanemaknasse tagasipöördumist suhtlema) ja tabelite komponente Bootstrapi raamistikus.

Rakenduse prototüübi nö "navigatsiooniriba" on loodud koostöös javascriptiga. Kui kasutatava seadme ekraanilaius on väiksem kui 768 pikslit, siis kuvatakse navigatsiooniriba elementide asemele nupp, millele vajutades avaneb alam-menüü, kus asetsevad kõik navigatsiooniriba elemendid. Seda selleks, et kasutajal oleks mugav telefonide ja muude seadetega, mille ekraanilaius on väike, rakenduses lihtsasti liigelda.

Modali diloogiaken kasutab samuti javascripti laiendust, mida rakendus kasutab uue kasutaja registratsiooni vormi kuvamiseks [22]

4.2 Less

Less on CSS-i eeltöötlus, mis tähendab, et see laiendab CSS keelt. Less lisab uusi funktsioone, mis lubavad kasutada muutujaid, mixin-e (saab kasutada eelnevalt defineeritud klassi ja id selektoreid teiste sees[21]), funktsioone ja teisi erinevaid tehnikaid, mis muudavad CSS-i rohkem hooldatavaks, muudetavaks ja laiendatavaks. [20]

Käesolevas rakenduses kaustatakse Less-i Bootstrapi vaikevormingu muutmiseks ja genereerimiseks, et kõiki muudatusi ei peaks eraldi .css failiga üle defineerima.

Näiteks saab Bootstrapis lessiga muuta võrgustiku muutujaid:

`@grid-columns:10;`	Muudab võrgustiku veergu arvu 10-ks. (vaikeväärtus 12).
`@grid-gutter-width:20px;`	Muudab võrgustiku veergude vahe 20px suuruseks (vaikeväärtus 30px).

Tabel 6. Näide sellest, kuidas lessiga muuta võrgustiku muutujaid.

4.2 Javascript, Ajax ja jQuery

Javascript on programmeerimiskeel, mis jookseb veebibrauseris. Seda arendasid samad inimesed, kes tegid Netscape brauseri ning Javascript oli sinna sisse ehitatud alates teisest versioonist. Javascriptil ei ole mingit erilist seost Java programmeerimiskeelega ning kuigi mõned nüansid võivad klappida, siis tegelikult on need kaks keelt täiesti erinevad. Algselt teati teda üldse nimega “Livescript”, kuid kui Sun Microsystems Java välja lasi, nimetati keel ümber Javascriptiks. [27]

Javascript jookseb brauseris ja ei vaja mingit serveripoolset tarkvara toimimiseks. Kõik muutused nagu piltide liikumised, teksti teisendused jms on võimalikud läbi Javascripti. Samuti kasutatakse Javascripti html vormide algkontrollina, et andmeid ei peaks mitu korda serverile saatma. [27]

Antud rakenduses kasutatakse Javascripti jQuery teeki. jQuery on kiire, väiksemahuline ja funktsioonirohke Javascripti teek, mis teeb HTML dokumentide manipuleerimise, sündmuste, animatsioonide ja Ajaxi toimingute tegemise palju lihtsamaks, kasutatades selleks lihtsalt rakendusliidest, mis on ühilduv enamuste brauseritega. Oma mitmekülgsusega on jQuery muutnud miljonite inimeste Javascripti kirjutamise viisi. [30]

Prototüübis on kasutatud koostöös Bootstrapi raamistiku ja jQuery teekiga Modali dialoogiakna võimalusi, kus kasutaja saab endale konto registreerida, samuti on Javascripti kasutatud nn "tundliku" disaini loomiseks navigatsiooniribal. Javascripti kasutatakse ka Ajaxi päringute tegemisel.

Ajax, ehk "asünkroone Javascript ja XML" on veebiarenduse tehnika, millega saab luua asünkroonseid veebirakendusi. Ajaxiga saab veebirakenduse andmeid saata ja andmeid vastu võtta asünkroonselt (tagaplaanil) ilma lehe uuesti laadimiseta (Vaata Lisa 4). Antud rakenduses on Ajaxi tehnoloogiat kasutatud uue Wordpressi paigalduse lisamisel, kus kasutajale näidatakse laadimise animatsiooni. Samuti on Ajaxi võimalusi kasutatud kasutaja registreerimise vormil validatsiooni teadete kuvamiseks, et lehekülge peale igat registreerimiskatset uuesti laadima ei peaks.

Kokkuvõte

Käesolevas töös kirjeldati, kuidas töötab ja milliseid tehnikaid ning tehnoloogiaid kasutati Automaatse veebilehtede veebipoe prototüübi loomisel. Loodud tarkvara võimaldab kasutajal peale registreerimist luua veebipoe serverisse oma Wordpressi paigaldus. Samuti saavad kasutajad vahetada ja lisada oma Wordpressi installatsiooni uusi teemasid ning vajadusel saab ka kõik ära kustuda ja otsast alustada. Samuti saab kasutaja näha profiilis oma andmeid.

Kuna prototüübi edasiarendus saab olema kommertstoode, siis on lisatud prototüüpi ka uute teemade ostmise võimalus, mis hõlmab endas ostukorvi süsteemi, kasutaja makse ajaloo kuvamist ning pangalinki. Maksed sooritatakse SEB panga testbaasis ning reaalset raha ei vajata.

Prototüübi loomisel kasutati erinevaid Javascripti, PHP, Bash skripti ja Linuxi tööriistu. Rakenduse põhi on üles ehitatud Laraveli PHP raamistiku abil, mis ei ole praegu veel väga levinud vahend veebirakenduste loomisel, kuid kogub arendajate seas kiiresti kuulsust oma paindlikkuse ja elegantse disaini põhimõtetega. Prototüübi serveri poole ülesehitusel lähtuti mudel-vaade-kontroller arhitektuurist. Rakenduse kasutajapoole kujundamisel ja ülesehitusel kasutati väga tuntud Twitteri Bootstrapi raamistikku, mis ühildub ideaalselt ka Laraveliga, mille komponendid kasutavad info näitamiseks Bootstrapi vormingut.

Rakenduse prototüübi edasiarenduses hakkab rakendus tööle ka PLESK kontrollpaneeliga (http://www.parallels.com/products/plesk/) ning tekivad juurde võimalused logo valimiseks ning domeeninime registreerimiseks, samuti teostatakse ka kliendile arvete saatmine ja lisatakse maksevõimalusi.

Prototüübi leiab aadressilt: http://martin1337.tk/os/public/.

Automatic online website store

Bachelor thesis

Martin Reilent

Summary

Nowadays having a personal website or a blog is very popular on the Internet, however, not many people possess the skills it takes to create one on their own. It was discussed in this paper which techniques and technologies were used in creating an automatic online website store prototype and how they worked. The created software allows the user to create their own wordpress installation to the applications server after they register. The users can also change and add new themes to their wordpress installation and when needed, they can also erase their work and start over. The user's data is also accessible in their profile.

Since after further development, the product will be used for commercial purposes, the prototype includes an option to buy new themes, which also encompasses a shopping cart system, displaying the user's payment history and a bank link. The payments are done in an SEB bank testing environment and no real money is needed.

The prototype was created with the aid of Javascript, PHP, Bash script and linux tools. The base of the prototype is built with the help from Laravel's PHP framework, which as of now is not a very popular means to create web applications, but is gaining reputation amongst developers for its flexibility and elegant design. When designing the application's server, the model-view-controller architecture was taken into account. When designing and creating the application's front end, a very popular framework, namely Twitter's Bootstrap was used, that also works perfectly with Laravel, the components of which use Bootstrap's format to display information.

After developing the application's prototype, the application will run with the PLESK control panel (http://www.parallels.com/products/plesk/) and there will be new possibilities to choose a logo and to register a domain name; the possibility to send a client their bill and payment options will be added as well.

Prototype can be found at: http://martin1337.tk/os/public/.

Kasutatud kirjandus

[1] - http://www.php.net/manual/en/preface.php

[2] - http://www.php.net/usage.php

[3] - http://www.php.net/manual/en/intro-whatcando.php

[4] - http://www.unixgeeks.org/security/newbie/unix/cron-1.html

[5] - http://laravel.com/docs/introduction

[6] - http://laravel.com/docs/eloquent

[7] - https://tutsplus.com/tutorial/why-laravel-is-taking-the-php-community-by-storm/

[8] - http://trends.builtwith.com/cms

[10] - https://codex.wordpress.org/Using_Themes

[12] - http://laravelbook.com/laravel-architecture/

[13] - käsk “php artisan list”

[14] - http://wp-cli.org

[15] - http://wp-cli.org/commands

[16] - http://en.wikipedia.org/wiki/Bash_(Unix_shell)

[17] - http://getbootstrap.com

[18] - http://spotthestation.nasa.gov

[19] - http://getbootstrap.com/css/

[20] - http://lesscss.org/

[21] - http://lesscss.org/features

[22] - http://getbootstrap.com/javascript/

[24] - https://github.com/edvinaskrucas/notification

[25] - http://www.php.net/manual/en/reserved.variables.session.php

[26] - https://github.com/Crinsane/LaravelShoppingcart

[27] - http://www.webdevelopersnotes.com/basics/languages_on_the_internet.php3

[28] - http://www.seb.ee/ariklient/igapaevapangandus/maksete-kogumine/maksete-kogumine-internetis/pangalingi-tehniline

[29] - https://www.zone.ee/blogi/2006/12/12/pangalink/

[30] - http://jquery.com

[31] - http://en.wikipedia.org/wiki/Ajax_(programming)

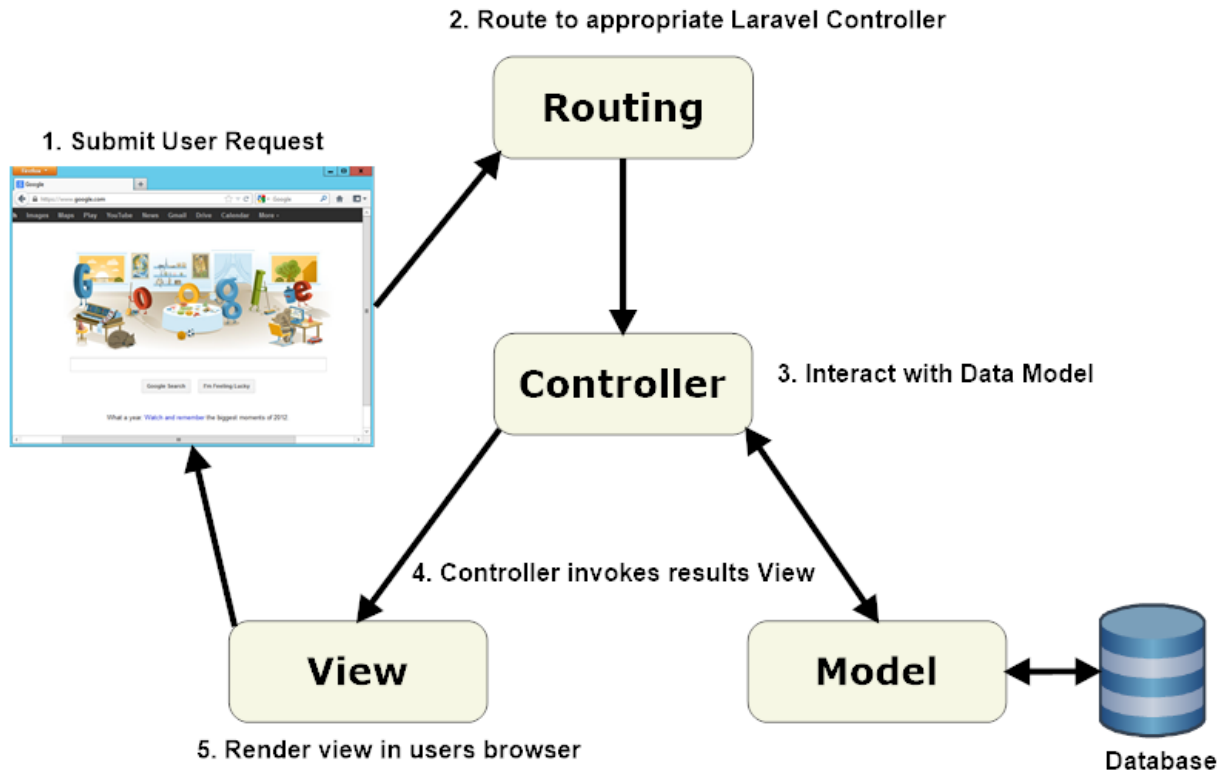
Lisad

Jrk.	Välja nimi	Pikkus	Kirjeldus
1	VK_SERVICE	4	Teenuse number (1001)
2	VK_VERSION	3	Kasutatav krüptoalgoritm (008)
3	VK_SND_ID	15	Päringu koostaja ID (Kaupluse ID)
4	VK_STAMP	20	Päringu ID
5	VK_AMOUNT	12	Maksmisele kuuluv summa
6	VK_CURR	3	Valuuta lühend: EUR
7	VK_ACC	34	Saaja konto number
8	VK_NAME	70	Saaja nimi
9	VK_REF	35	Maksekorralduse viitenumber
10	VK_MSG	95	Maksekorralduse seletus
-	VK_MAC	700	Kontrollkood e. allkiri
-	VK_RETURN	255	URL, kuhu vastatakse edukal tehingu sooritamisel
-	VK_LANG	3	Soovitav suhtluskeel

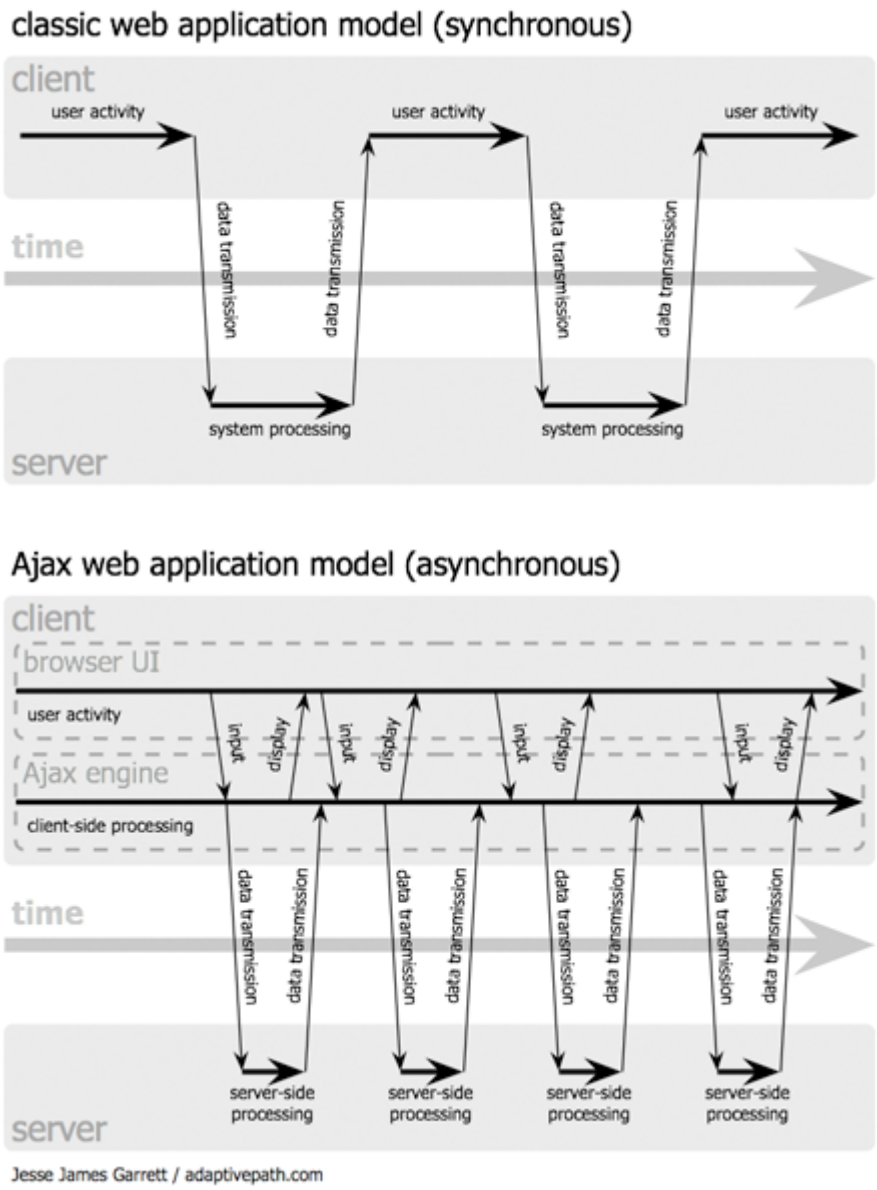
Lisa 1. Panka saadetavad väljad, teenus koodiga 1001

Jrk.	Välja nimi	Kirjeldus
1	VK_SERVICE	Teenuse number (1101).
2	VK_VERSION	Kasutatav krüptoalgoritm 008.
3	VK_SND_ID	Päringu koostaja ID (Panga ID).
4	VK_REC_ID	Päringu vastuvõtja ID (Kaupluse ID).
5	VK_STAMP	Päringu ID.
6	VK_T_NO	Maksekorralduse number.
7	VK_AMOUNT	Makstud summa.
8	VK_CURR	Valuuta lühend: EEK (Alates 01.01.2011 EUR).
9	VK_REC_ACC	Saaja kontonumber.
10	VK_REC_NAME	Saaja nimi.
11	VK_SND_ACC	Maksja kontonumber.
12	VK_SND_NAME	Maksja nimi.
13	VK_REF	Maksekorralduse viitenumber.
14	VK_MSG	Maksekorralduse selgitus.
15	VK_T_DATE	Maksekorralduse kuupäev.
-	VK_CHARSET	Sõnumi kodeering. Mittekohustuslik parameeter. Lubatud ISO-8859-1(vaikeväärtus) või UTF-8.
-	VK_MAC	Kontrollkood e. allkiri.
-	VK_LANG	Soovitav suhtluskeel.
-	VK_AUTO	Näitab seda, kas pakett oli saadetud automaatselt (`Y`) või mitte (`N`).

Lisa 2. Pangast saadetud väljad, teenuskoodiga 1101



Lisa 3. Laraveli komponentide vaade



Lisa 4. Ajaxiga ja ilma Ajaxita veebirakenduse mudel

Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina **Martin Reilent** (sünnikuupäev: 28.01.1992)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   Automaatne veebilehtede veebipood,
   mille juhendaja on **Vambola Leping**,
   1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **14.05.2014**