Tallinna Ülikool

Digitehnoloogia instituut

Broneerimissüsteemi loomine Laraveli raamistiku toel

Bakalaureusetöö

Autor: Meelis Koger

Juhendaja: Jaagup Kippar

Autor: .....”.....” 2018
Juhendaja: .....”.....” 2018
Instituudi direktor: .....”.....” 2018

Tallinn 2018

Autorideklaratsioon

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

.... (kuupäev) (autor)

Sisukord

Sissejuhatus

Loodud bakalureusetöö on rakenduslik, mille eesmärk on luua broneerimissüsteem, kasutades tuntud PHP raamistikku Laravel. Töös uuritakse lähemalt, mis võimalusi ja eeliseid pakub Laraveli raamistik ning leitakse neile kinnitust läbi katsetuste ja broneerimissüsteemi rakenduse loomisel. Töö lugemisel tasub arvestada, et see on üks levinumaid raamistikke, mida saab kasutada ka mitmete teiste veebipõhiste rakenduste või süsteemide loomisel. Põhjusi, miks kasutada just Laraveli raamistikku, on mitmeid ning neid on ka selles töös kirjeldatud.

Käesoleva teema valik on aktuaalne ja põhineb ideel: tutvustada ning anda ülevaade, milline näeb välja broneerimissüsteemi loomise protsess ühe populaarseima PHP veebiraamistikuga – Laravel. Kuna ülikooli õppekavas ei ole ette nähtud Laraveli või mõne muu PHP veebiraamistiku kasutamist, puudub terviklik materjal, kuidas raamistikult põhinevat projekti luua ja üles seadistada. Töö on mõeldud lugemiseks eelkõige neile, kes on vähesemal määral PHP keelega kokku puutunud. Kõik, kes PHP keeles programmeerivad ja soovivad spetsialiseeruda, peaksid suutma käistleda ja luua süsteeme, kasutades mõnda veebiraamistikku.

Töös käigus antakse ülevaade Laravelist ja kirjeldatakse veebiraamistiku poolt pakutavaid levinumaid ja kaasaaegseid võimalusi, mille infole tuginedes saab luua parema ülesehitusega rakendusi. Lisaks kirjeldatakse broneerimissüsteemi kavandamist ja ülesseadmise protsessi veebiraamistikule tuginedes ning võimalusi süsteemi arendamisel, millega saaks süsteemi täiustada. Projekti kood on üleval GitLab keskkonnas, millel on piiratud ligipääs. Koodi saab soovi korral uurida koos töö autoriga. Toimiva testkeskkonna ja tootmiskeskkonnaga saab tutvuda aadressidel https://test.booklux.com/login ja https://app.booklux.com/login, sisestades kasutajatunnuseks “tennis@booklux.com” ja parooliks “admin”.

1 Broneerimissüsteemi tutvustus ja loomise etapid

Ennem kui alustada projekti loomise, andmebaasimudelite loomise ja programmeerimisega, on vajalik välja mõelda kogu protsessi tegevused, toimingud ning lahendused erinevate probleemide kõrvaldamiseks. Selles peatükis kirjeldatakse detailselt süsteemi toimimise protsessi ning mida süsteem peab täpsemalt võimaldama. Samuti käsitletakse punkte, kuidas kavatsetakse tagada süsteemi juures turvalisus, varundamine ja kiirus. Kuna broneerimissüsteem on juba varasemalt loodud, kuid raamistikku kasutamata, oli suur väljakutse planeerida tegevust nii, et süsteemil oleks korrapärane struktuur, arvestades varasemat projekti ülesehitust.

1.1 Süsteemi kirjeldus ja nõuded süsteemile

Töö käigus luuakse broneerimiskeskkond, kus kasutajale tehakse kättesaadavaks automaatselt nii halduskeskkond kui ka kliendivaade. Näiteks iga spordiasutus saab luua oma ettevõttele konto, kus hallata broneeringuid; luua treeninguid; pakkuda broneerimisvõimalust klientidele nende kodulehe ja/või Facebooki lehe kaudu.
Konkureeriva broneerimissüsteemi loomisel on vajalik arvestada juba turul pakutavate süsteemide ja nende võimalustega. Kliendi soovidest ja turu-uuringutest lähtuvalt jõuti järeldusele, et süsteem peab olema lihtne, soodne ning pakkuma kõike käepäraseid meetmeid broneeringute tegemiseks ja haldamiseks. Tegevuse käigus selgus, et ettevõtetel on tihti omad kindlad soovid, mida peaks süsteem just neile võimaldama, mistõttu tuleb süsteemi loomisel tekitada vahendid süsteemi personaliseerimiseks ja ühildamiseks teiste süsteemidega. Omadused/tegevused, mida süsteem täpsemalt kõigile võimaldama, peavad olema järgnevad:

- Broneeringute haldussüsteem
- Broneerimisvorm, mida saab kodulehele ja Facebooki
- Võimalust broneeringuid hallata telefonis, tahvelarvutis ja lauaarvutis
- Android ja iOS äpp broneeringute haldamiseks
- Automaatne emaili ja SMS meeldetuletussüsteem

- Turundusmoodul klientidele teadete saatmiseks
- Kliendihaldussüsteem
- Toodete laoarvestus
- Töögraafikute loomine
- Statistikamoodul
- Kontode haldus
- Personaliseerimisvõimalus

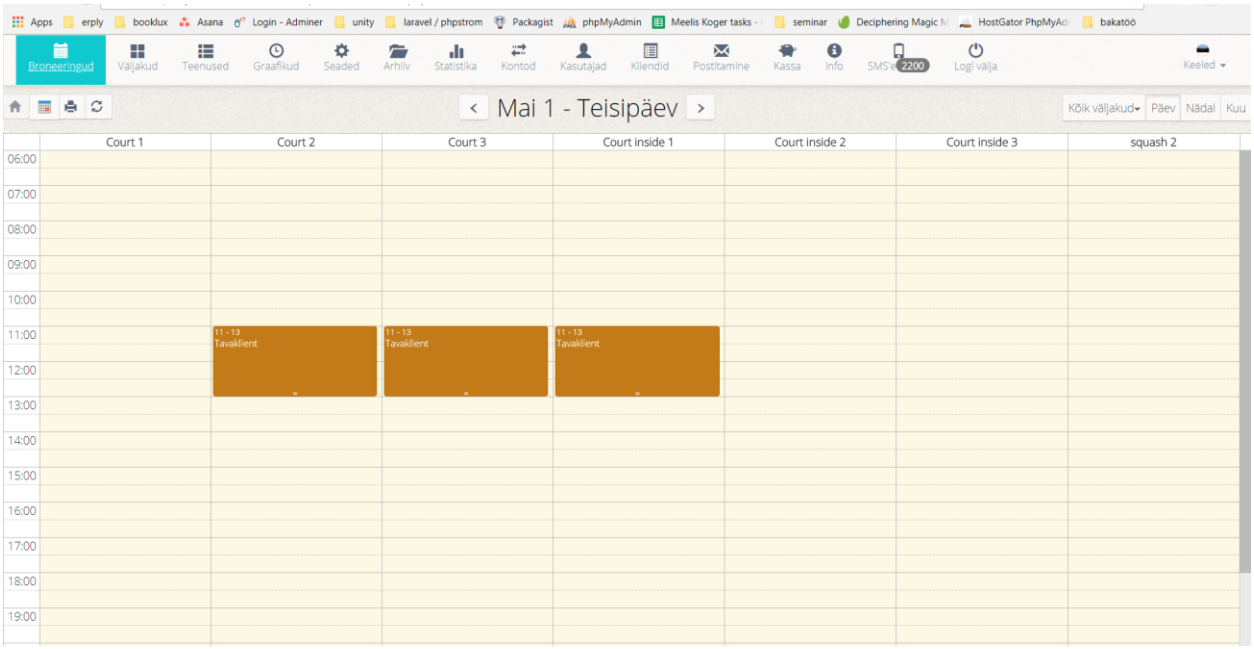
Broneerimissüsteem on tänapäeval eelkõige ilusalongidel, heaolukeskusetel ja teistel teenusepakkujatel igapäeatöö lahutamatu osa, mis aitab vähendada administratiivtegevusi ning on mugav klientidele, eelkõige võimaldades töövälistel aegadel teenuse osutamist broneerida. Lisaks aitab antud keskkond vähendada klientide etteteatamata teenusest loobumisi, kuna süsteem saadab automaatselt meeldetuletusi emaili või SMS teel.

1.2 Halduskeskkond

Enne halduskeskkonna kasutamist, peab ettevõte esmalt kasutajaks registreerima. Kui ettevõte on konto loonud, suunatakse ta automaatselt autenditud kujul halduskeskkonda, kus ettevõte saab etapiti keskkonna üles seadistada. Kui esmased etapid on läbitud, pakub süsteem veel hulga lisavõimalusi, mida saab süsteemi kohandamiseks kasutada. Halduskeskkonna menüüst leiab mitmeid lehti erinevateks otstarveteks (vt pilt 1), mille hulka kuuluvad:

- Broneeringud – koht, kus saab broneeringuid luua ja hallata.
- Väljakud – koht, kus luua ja hallata väljakuid
- Teenused – võimalik luua ja hallata erinevaid broneeringu tüüpe (nt: tava-, püsibroneering)
- Graafikud – koht, kus on saab väljakutele määrata kuupäevapõhist graafikut ja näiteks neid teatud nädalapäevadel lukustada.
- Seaded – koht kus saab määrata detailsed seadistus ning leida infot kodulehele või Facebooki intergreerimiseks

- Arhiiv – koht, kus saab vaadata ja otsida vanu broneeringuid tabeli kujul
- Statistika – koht, kus saab vaadata väljakute ja teenuste tarbivuse suhtes statistikat määratud perioodi vahemikus
- Kontot – leht, kus saab hallata ja luua erinevate õigustega kasutajakontosid
- Kasutajad – leht, kuhu tekivad kodulehe kaudu loodud klientide kasutajakontod. Samuti saab kasutajakontosid ise juurde luua ja hallata.
- Kliendid – koht, kuhu kogunevad automaatselt klientide andmed, kes on halduskeskkonnas või kodulehe kaudu broneeringud loonud. Ning neid eraldi gruppidesse määrata.
- Postitamine – koht, kus saab loodud kliendigruppidele uudiskirju saata ning uudiskirju ise kujundada.
- Ladu – koht, kus saab pidada laoarvestust toodete üle
- Kassa – koht, kus saab broneeringute eest tehtud tehingud näha ning neid hallata.
- Info – koht, kus jooksevad teated uuendustest ning on kättesaadav online klienditugi.



Pilt 1: Halduskesskonda demonstreeriv pilt

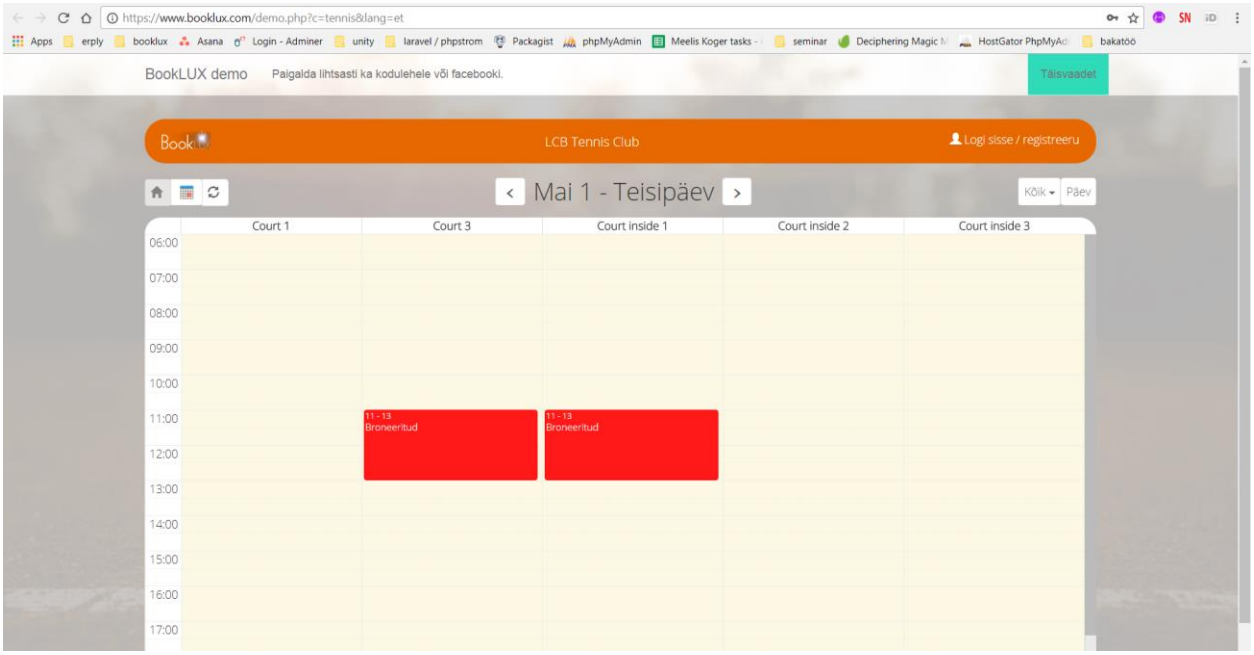
1.3 Kliendi broneerimisvorm

Süsteemi üheks eesmärgiks on pakkuda klientidele broneerimisvõimalust läbi ettevõtte kodulehe. Juhul kui süsteemis on konto loodud ja seadistused tehtud, peab ettevõtte saama lihtsasti lisada broneerimisvormi mistahes kodulehele. Kodulehele ülesseadmise eeltingimus on konto olemasolu, mille tulemusel genereeritakse kontole automaatselt unikaalne veebilink lähtuvalt ettevõtte nimest ning mille alusel on võimalik genereeritud html kood kodulehele integreerida. Kodulehele intergreerimise kood põhineb iframe'i komponendil (vt koodinäide 1).

```
<iframe
      src="https://www.booklux.com/firma_nimi"
      width="100%" height="650px"
      style="border:none">
</iframe>
```

Koodinäide 1: iframe kood, millega saab kodulehel kuvada broneeringuvormi

Juhul kui iframe'iga järgnev kood veebilehele implementeerida, kuvatakse broneerimisvorm ilma taustapildi või värvita, mis jätab mulje, et tegu ei oleks välise süsteemiga (vt pilti 2).



Pilt 2: veebilehe sees kuvatav broneerimisvorm iframe'i abil

1.4 Varundamine

Kogu broneerimissüsteemi puhul kasutatakse ühte serverit – jagatavat pilvepõhist serverit - ainult andmete ja failide varundamiseks. Süsteemi puhul on oluline, et ei saaks tekkida pikaajalist infovahetuse seisakut, mistõttu ei saa süsteemi tugineda vaid ühele VPS serverile. Näiteks on töökoostajal kogemust, kus veebimajutaja kõikides serverites toimus rike, mille puhul ei olnud võimalik saada mitmeid tunde ligipääsu ei süsteemile ega andmetele. Sarnase olukorra vältimiseks otsustati võtta kasutusele teistest sõltumatu veebimajutaja server andmete varundamiseks. Juhul kui ühe veebimajutajale toimub rünne või mõni muu rike, mis peaks häirima severite tööd, saavad kliendid kasutada ajutiselt edasi süsteemi backup serverit, kuniks süsteem taastatakse. Antud lahenduse juures on oluline, et ka andmed (eelkõige broneeringud) andmebaasis reaalajas backup serveris uueneks. Selle lahenduseks tehakse kõiki broneeringute toimingud läbi ühe funktsiooni, kus on defineeritud toimingu tüüp ja saadetav sql päring (vt koodinäide 2). Kõik broneeringute päringud saadetakse funktsiooniga backup serveri veebiaadressile kasutades curl'i, kus vastav päring käivitatakse andmebaasis, et andmed identsed oleksid.

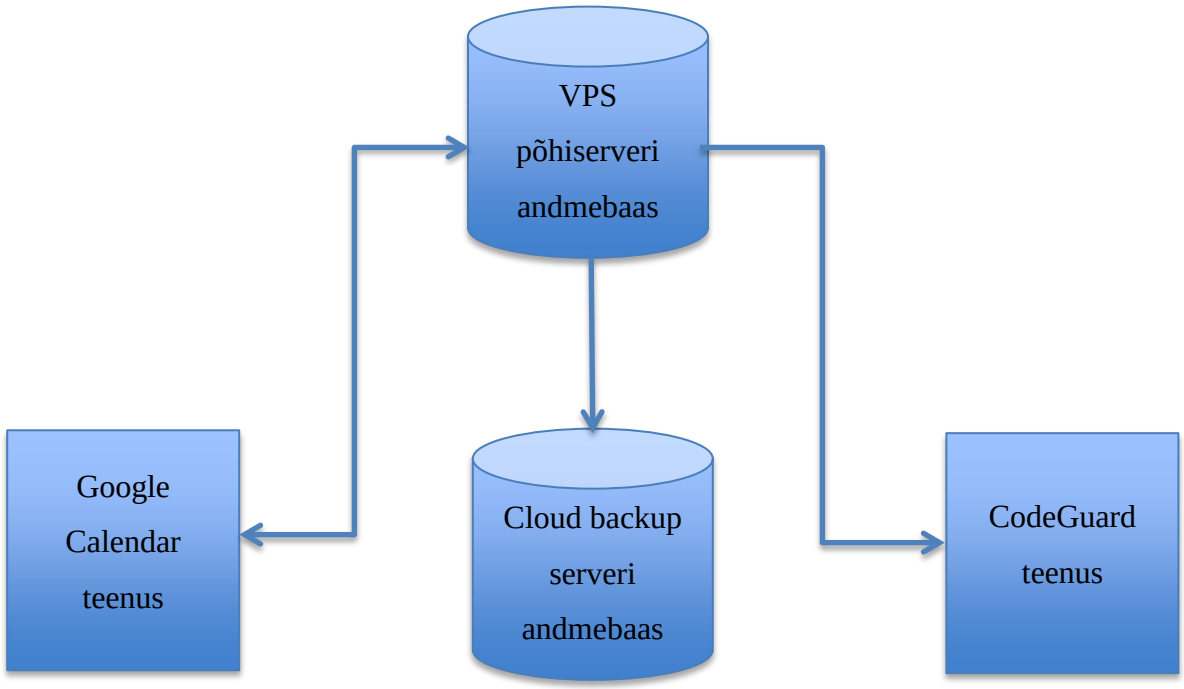
```
public function backupRequest($sql_query, $log_id, $id) {
    $data = [
        "id" => $id,
        "log_id" => $log_id,
        "sql" => base64_encode($sql_query)
    ];
    $runfile = 'http://backup.booklux.net/url';
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $runfile);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query($data));
    curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 0);
    $content = curl_exec ($ch);
    curl_close ($ch);
}
```

Koodinäide 2: curl funktsioon, millega saadetakse paring backup serverisse

Juhul kui andmebaasis peaks toimuma andmete rike, mis võib olla tingitud pahatahtlikust ründest või arendaja enda eksimusest, kus andmeid mingil viisil kustutatakse või valede väärtustega üle kirjutatakse, on oluline teha ka andmete varundust teatud aja tagant, mis annab võimaluse andmeid taastada. Selleks kasutatakse Codeguard teenust.

CodeGuard on teenus, mis pakub automaatset veebisaidi varundamise lahendust. CodeGuard jälgib kasutaja saiti ja hoiab kõikide muudatustega kursis, pakkudes sagedasi varukoopiaid ja taastamisvalikuid seisakute vältimiseks (What Is CodeGuard?, kuupäev puudub). Codeguard teenuse valis töökoostaja seetõttu, kuna ta pakub mugavat lahendust igapäevaselt failide varundamist FTP kaudu ja ka andmebaasi varundamist, kust saab vaid mõne klikiga laadida üles varasema versiooni failidest ja andmetest kaitsmaks võimalikke häkkimiste ja viiruste eest.

Kõik kasutajad, kes soovivad kasutada kahepoolset andmete sünkroniseerimist, saavad lisaks kasutada ka Google'i kalendrit teenust. Kogu andmete varundamise plaani saab näha joonisel (vt joonis 1).



Joonis 1: Varundamise protsess

1.5 Kujundamine

Projektis kasutatatakse kujunduseks ühte tuntuimat *front-end* raamistikku Bootstrap. Selle raamistiku abil saab ehitada kiiresti kaasaaegse veebilahenduse, mis kohandub hästi kõikides seadmetes, näiteks: nutitelefonis, tahvelarvutis, lauaarvutis.

Bootstrap on avatud lähtekoodiga tööriistakomplekt HTML-i, CSS-i ja JS-i arendamiseks. Selle abil saab kiiresti prototüüpida oma ideid või ehitada kogu oma rakendus, kasutades SASSi muutujaid, kohanduvat võrgustikusüsteemi, laialdased eelinstallitud komponente ja jQuery'is ehitatud võimsad pistikprogramme. (Bootstrap, kuupäev puudub)

Bootstrapi puhul on veel palju omadusi, milleks broneerimissüsteemi loomise puhul antud raamistikku eelistati:

- Aitab aega säästa, kasutades juba raamistiku poolt eeldefineerituid klasse, komponente.
- On kohandatav ehk ei pea laadima sisse kõike komponente, mida pole projektis vaja.
- Toimib veatult ja on sarnane kõikides erinevates kaasaaegsetes veebilehitsejates.
- Komponendid ise kohanduvad ja muudavad toimimist vastavalt ekraanisuurusele.
- Põhjalik dokumentatsioon koos erinevate näidetega.
- Raamistik ise on vabavaralise litsentsi all.

1.6 Turvalisus

Turvalisuse tõstmiseks pakub Laraveli raamistik üksjagu võimalusi. Peamised turvalisust käsitletavad aspektid on: ssl sertifikaat, sql injection, csrf token.

SSL sertifikaat tagab selle, et veebileht, mida veebilehitseja kuvab, oleks tõepoolest see, kuhu külastaja jõuda soovis ning et ühendus veebilehe ja selle külastaja vahel oleks turvaline. Turvaline ühendus tähendab seda, et saidi ja selle külastaja vahelised päringud on SSL sertifikaadi abil krüpteeritud. Sisuliselt on nii veebilehe kuvamine kui kõik seal tehtavad tegevused päringud. Näiteks järgmisele sisulehele klõpsamine ei ole midagi muud kui veebilehelt info pärimine. Kui ühendus veebilehe ja selle külastaja vahel ei ole krüpteeritud, tehakse kõik päringud avalikult ning kõiksugused pahalased saavad tahtmise korral seda suhtlust pealt kuulata. (Strauss, 2017)

Süsteemi andmebaasi päringute puhul kasutatakse Laraveli päringute loojat. Laraveli päringute looja kasutab PDO-parameetrite sidumist, et kaitsta teie rakendust SQL-i süstimisrünnakute (*sql injection*) vastu. Seega puudub vajadus puhastada stringe, mis kantakse üle sidumistena. (Database: Query Builder, kuupäev puudub)
Väliste päringute kaitseks kasutatakse Laraveli poolt pakutavat csrf kaitset. Iga html vormi ja muu päringu juurde lisatakse varjatud CSRF-märgendi nii, et CSRF-i kaitse vahevara saaks taotlust kinnitada. Laravel muudab teie rakenduse kaitsmise saidivaheliste päringute võltsimisrünnakute (CSRF) eest lihtsaks. Vastastikused päringuplaanid on teatud tüüpi pahatahtlik ekspluateerimine, mille kohaselt volitamata käsud tehakse autentimata kasutaja nimel. Laravel genereerib automaatselt CSRF-i märgendi iga rakenduse hallatava aktiivse kasutaja seansi jaoks. Seda kutset kasutatakse selleks, et kontrollida, kas autentimiseks on kasutaja, kes tegelikult taotlusi esitab. (CSRF Protection, kuupäev puudub)

1.7 Kiiruse tagamine ja koormus (puhverdamine)

Iga suurema süsteemi puhul võib probleemiks tulla varem või hiljem süsteemi kiirus. Esiteks tagab parema kiiruse võimsama serveri valik, milleks kasutatakse veebimajutaja poolt pakutavat kõike võimsamat VPS serverit. Sellel on piisavalt võimekust ja mahtu süsteemi stabiilseks toimimiseks, tuginedes järgnevatele näitajatele:

- Operatiivmälu 6 GB RAM
- 4 tuumaline protsessor
- Kõvaketas 120 GB SSD RAID 10
- Maksimaalne andmeedastusmaht (Bandwidth) 3000 GB

Head kiirust aga ei saa tagada alati vaid võimeka riistvaraga, vaid on oluline minimaliseerida ka serveri koormust, optimeerides koodi, minimaliseerides andmebaasi päringuid ning kasutades puhverdamist.
Vahemälu (häältdatud CASH) on koht, kuhu ajutiselt talletada infot mõnda arvuti keskkonda.

Andmetöötluses talletatakse tihti aktiivsed andmed vahemällu, et lühendada andmetele juurdepääsu aegasid, vähendada latentsust ja parandada sisendit / väljundit (I / O). Kuna peaaegu kogu rakenduste töökoormus sõltub I/O toimingutest, kasutatakse rakenduse jõudluse parandamiseks vahemälu (Rouse, 2017). Antud programmi puhul kasutatakse veebilehitseja põhist puhverdamist, mille puhul salvestatakse süsteemi kõik pildifaild, CSS kujundusfailid, JS skriptifailid kasutaja arvutisse, et süsteemi laadimisel ei peaks veebilehitseja pöörduma iga kord serveri poole samu faile küsima. See info defineeritakse ära .htaccess failis ning neid ei uuendata tihedamini, kui 7 päeva tagant (vt koodinäide 3). Kokkuvõttes vähendab see oluliselt süsteemi kiirust, kuna arvuti suudab kohalikult kettalt veebilehitseja failid kiirem sisse laadida, kui küsida neid üle võrgu väliselt serverilt. Lisaks vähendab see serveri koormust, kuna server ei pea saatma kõike faile iga kord kasutajale, kui ta süsteemi kasutab või veebilehte külastab.

```
<IfModule mod_headers.c>
    # 1 hour for most static assets
       <filesMatch ".(css|jpg|jpeg|png|gif|js|ico)$">
              Header set Cache-Control "max-age=86400, public"
       </filesMatch>
</IfModule>

<IfModule mod_deflate.c>
       AddOutputFilterByType DEFLATE text/html
       AddOutputFilterByType DEFLATE text/css
       AddOutputFilterByType DEFLATE text/javascript
       AddOutputFilterByType DEFLATE text/xml
       AddOutputFilterByType DEFLATE text/plain
       AddOutputFilterByType DEFLATE image/x-icon
       AddOutputFilterByType DEFLATE image/svg+xml
       AddOutputFilterByType DEFLATE application/rss+xml
       AddOutputFilterByType DEFLATE application/javascript
       AddOutputFilterByType DEFLATE application/x-javascript
       AddOutputFilterByType DEFLATE application/xml
       AddOutputFilterByType DEFLATE application/xhtml+xml
       AddOutputFilterByType DEFLATE application/x-font
       AddOutputFilterByType DEFLATE application/x-font-truetype
       AddOutputFilterByType DEFLATE application/x-font-ttf
       AddOutputFilterByType DEFLATE application/x-font-otf
       AddOutputFilterByType DEFLATE application/x-font-opentype
       AddOutputFilterByType DEFLATE application/vnd.ms-fontobject
```

```
        AddOutputFilterByType DEFLATE font/ttf
        AddOutputFilterByType DEFLATE font/otf
        AddOutputFilterByType DEFLATE font/opentype
        # For Olders Browsers Which Can't Handle Compression
        BrowserMatch ^Mozilla/4 gzip-only-text/html
        BrowserMatch ^Mozilla/4\.0[678] no-gzip
        BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
</IfModule>
```

Koodinäide 3: Failide puhverdamise script

1.8 Personaliseerimine

Süsteemi loomisel oli suureks väljakutseks mõelda välja lahendus, kuidas pakkuda ettevõtetele valmistoodet, mida neil on võimalik kohandada ja lisada juurde teenuseid nii, et see ei mõjutaks standardtoodet, selle toimimist ja teisi kliente.

Lahenduseks on siinkohal pluginate süsteem, mille puhul laetakse kliendifail süsteemi sisse juhul, kui see pluginate kaustas eksisteerib. Kliendifaili on võimalik lisada nii personaalset PHP kui Javaskript koodi. Süsteemi javaskripti kood on üles ehitatud funkstionaalse programmeerimise meetmete alusel koos mitmete kuulajatega, mis annab võimaluse meetmeid üle kirjutada ja vastavalt kliendi soovidele süsteemitoimimist muuta (vt koodinäide 4). Kuna hetkel pole süsteemile loodud API’t, ei saame anda veel kliendile võimalust muudatusi ise sisse viia.

```
APP.openBookingForm = function(state){
    $("#saveBookingModal").modal('show');
    callBackFunction("afterOpenBookingForm", state);
}
```

Koodinäide 4: Funktsionaalse programmeerimise javaskripti funktsioon, kus broneerimisvormi avanedes on võimalik defineerida funktsiooni kuulaja kliendi plugina faili

2 Laraveli raamistik

2.1 Tutvustus ja ajalugu

Laraveli on loonud ja arendanud Sir Taylor Otwell, kes proovis anda vanemale PHP-raamistikule CodeIgniter head asendajat. Seda põhjusel, et CodeIgniter ei pakkunud selliseid suurepäraseid omadusi nagu sisseehitatud kliendi autentimist ja nõuetekohase kasutaja autoriseerimist. 2011. aastal 9. juulil avaldas Laravel oma esimese beetaversiooni ja hiljem samal kuul Laravel 1 avalikustati. Lisaks autentimisele pakub Laravel ka sisseehitatud tugiteenust lokaliseerimiseks, vaatetele, sessioonide käsitlusega, konkreetsele kontrollerile päringute suunamisega ja muid hämmastavaid omadusi. (History of Laravel, kuupäev puudub)

Laravel on ekspressiivse ja elegantne süntaksiga veebirakenduste raamistik. Usutakse, et areng peaks olema nauditav, loominguline kogemus, et see oleks tõeliselt täiuslik. Laravel üritab eemaldada probleeme arendustegevustes, lihtsustades enamikes veebiprojektides kasutatavaid ühiseid ülesandeid näiteks autentimist, suunamisi, sessioone ja vahemällu salvestamist. (Laravel Philosophy, kuupäev puudub)

Laraveli eesmärk on muuta arendusprotsessi arendaja jaoks meeldivaks ilma, et oleks vaja rakenduste funktsioone ohverdada. Õnnelikud arendajad loovad parimat koodi. Sel eesmärgil üritati ühendada kõige paremad osad, mida on nähtud teistes veebirakendustes, sealhulgas raamistikud, mida kasutatakse teistes keeltes, näiteks Ruby on Rails, ASP.NET MVC ja Sinatra. (Laravel Philosophy, kuupäev puudub)

Laravel on ligipääsetav ja võimas, pakkudes tõhusaid tööriistu, mis on vajalikud suurte ja jõuliste rakenduste jaoks. Suurepärane inversioon kontrollkonteineritele, ekspressiivse migratsioonisüsteem ja tihedalt integreeritud üksuse testimise tugi annavad teile tööriistad, mida on vaja selleks, et luua mis tahes rakendust, mida on vaja teostada. (Laravel Philosophy, kuupäev puudub)

2.2 Viimased uuendused

Laravel arendab ja täiendab raamistikku pidevalt levinumate probleemide ja vajaduste näol. Töös tuuakse välja mõned silmapaistvamad ja kasulikud uuendused viimase kolme aasta jooksul.

Alates versioonist 5.1, mis avaldati 2015 aasta juunikuus, pakutakse versiooni puhul pikaajalist tuge. Laravel 5.1 saab toe veaparandusteks 2 aastat ja turvaparandusteks 3 aastat. See kõige pikem tugiaken, mida Laravel on välja pakkunud, tagades stabiilsuse ja meelerahu suurematele äriklientidele ja tavaklientidele. Üldjuhul pakutakse 6 kuulist veaparandustuge ja 1 aasta turvalisusetuge. (Release Notes, kuupäev puudub)

2.2.1 Teavitussüsteem

Alates versioonist 5.3 on Laravelil olemas teavitused, mis pakuvad lihtsat ja väljendusrikast API-t teadete saatmiseks mitmesuguste kanalite kaudu, nagu e-post, Slack, SMS ja muud. Laravel toetab teadete saatmist mitmesuguste kohaletoimetamiskanalite kaudu, sealhulgas e-posti, SMS'i (Nexmo kaudu) ja Slack. Teateid võib ka salvestada andmebaasi, et neid saaks kuvada veebiliideses. Tavaliselt peaksid märguanded olema lühikesed, informatiivsed sõnumid, mis teavitavad kasutajaid teie rakenduses esinenud asjadest. Näiteks võib saata oma kasutajatele e-posti ja SMS-kanalite kaudu teatise uuest broneeringust. (Sending Notifications, kuupäev puudub)

SMS-teatiste saatmine Laravel'is töötab Nexmo toel. Enne teatiste saatmist Nexmo-ga peab installima "nexmo/klient" pakkuja komplekti kasutades näiteks composerit ja täiendades konfiguratisioonifaili. (Sending Notifications, kuupäev puudub).

Seejärel tuleks luua uus teavitus käsuga "artisan make:notification BookingCreated" ning avada fail "app/Notifications/ BookingCreated.php", kus defineerida ja kohandada sms'i saatmise kood (vt koodinäide 5).

```
public function toNexmo($notifiable){
    return (new NexmoMessage)
                ->content($notifiable->content)
                ->from($notifiable->from_number);
}
```

Koodinäide 5: Sms teavituse kohandamine (Sending Notifications, kuupäev puudub).

Projektis saab teavitust kõikjal rakendada, kus kasutada eelnevalt loodud BookingCreated klassi ning kustudes seda Notification klassi "send" meetodiga välja (vt koodinäide 6).

```
use App\Notifications\BookingCreated;
use Notification;
$data->content = ”Broneeriti uus aeg”;
$data->from_number = ”3725666666”;
$data->phone_number = ”3725555555”;
Notification::send($users, new BookingCreated($data));
```

Koodinäide 6: Sms teavituse kasutamine

2.2.2 Üleslaetud failide salvestamine

Veebirakendustes on üheks kõige tavapärasem kasutus failide salvestamiseks on kasutajate poolt üleslaetavad failid, näiteks profiilipildid, fotod ja dokumendid. Laravel 5.3 muudab üleslaaditud failide salvestamise väga lihtsaks, kasutades "store" meetodit uute üleslaetud failide eksemplari talletamiseks. Lihtsalt tuleb välja kutsuda "store" meetod, kus soovite üleslaaditud faili salvestada. Üleslaadimisel saab lihtsasti kasutada ka Amazoni s3 failide majutamise teenust, kus esmalt tuleb composer'iga ajur installida ja konfigureerimisinfo lisada konfigureerimisfaili "config/filesystems.php". Seejärel saab vähese koodikirjutamisega lisada fail teenusesse üles ning tagastada faili link (vt koodinäide 7). Juhul kui pole oluline määrata üleslaetavale failile kindlat nime ja koodis te seda ei defineeri, genereeritakse failile nimetus automaatselt. (File Uploads, kuupäev puudub)

```
public function update(Request $request)
{
    $path = $request->file('pictures')->store('gallery', 's3');
    return $path;
}
```

Koodinäide 7: Faili üleslaadimise funktsiooni, kasutades Amazon s3 teenust (File Uploads, kuupäev puudub)

2.2.3 *Loop* muutuja

Kui tsüklit luua, on *loop* muutuja tsüklis saadaval. See muutuja võimaldab kasutada mõnda informatsiooni kasulikku bitti, näiteks praeguse tsükli indeksit ja seda, kas see on loendist esimene või viimane iteratsioon (vt koodinäide 8) (The Loop Variable, kuupäev puudub). Näiteks on seda mugav kasutada ka kujundamises, kui soovite jooksvalt luua veebivaatesse andmeridasid, et saaks esimese ja viimase andmekasti veidi ümaramaks või silmapaistvamaks teha.

```
@foreach ($users as $user)
    @if ($loop->first)
        This is the first iteration.
    @endif
    @if ($loop->last)
        This is the last iteration.
    @endif
    <p>This is user {{ $user->id }}</p>
@endforeach
```

Koodinäide 8: Template failis kasutatav kastutajate listi tsüklit käivitatav funktsioon (The Loop Variable, kuupäev puudub)

2.2.4 Autentimise tellingud

Laravel võimaldab hõlpsasti käsitleda serveritpoolset autentimist, kuid Laravel 5.2 pakub mugavat ja välkkiiret viisi luua ka autentimisvaated. Lihtsalt käivitage “make:auth” käsk oma terminalis. See käsk loob kasutaja sisselogimise, registreerimise ja parooli lähtestamise jaoks bootstrapiga ühilduvad vaated. Käsk uuendab marsruutide faili asjakohaste marsruutidega. (Authentication, kuupäev puudub)

Käsitletava süsteemi puhul ei saa hetkel seda lahendust rakendada, kuna kasutusel on korraga kaks andmebaasi korraga, eesmärgiga kasutajat autentida. Küll aga on genereeritud projekti nimetatud käsuga failid, sest tulevikus plaanitakse kaks andmebaasi ühildada või kõikide kasutajate andmed ühe andmebaasi tabelisse tuua. Seejärel on võimalik koheselt antud lahendusele üle minna.

2.2.5 Päringute piiramine

Raamistikuga on kaasas uus kiirusepiiranguga vahevara, mis võimaldab kergesti piirata päringute arvu, mida teatud IP-aadress võib marsruudil määratud arvu piires minutiga teha. Broneerimissüsteemi puhul on oluline võimaliku pahatahtliku häkkimise eest näiteks piirata sisselogimise päringuid. Selleks määratakse marsuutide failis sisselogimispäringutele piirang, kus ühe minuti jooksul ei lubata teha rohkem päringuid kui 60 taotlusega IP-aadressist (vt koodinäide 9). (Stauffer, 2015)

Muul juhul võib eeldada, et päringut ei tee reaalne inimene, vaid keegi üritab kas kontosse sisse häkkida või muul pahatahtlikult moel päringuid teha.

```
Route::post('/login', [
    'middleware' => 'throttle:60,1',
    'AccountController@login'
]);
```

Koodinäide 9: Sisselogimispäringute piiramine broneerimissüsteemis

3 Laravel projekti loomine ja ülesseadmine

Broneerimissüsteemi loomisel arvestati asjaoluga, et süsteem oli varasemalt loodud ühtegi PHP veebiraamistikku kasutamata. Seetõttu ei saanud igas olukorras kasutada ideaalseid meetmeid, mida raamistik pakub, vaid arvestati ka varasemalt loodud projekti seadistuste ja andmebaasistruktuuriga, et süsteem saaks sarnaselt edasi toimida. Süsteemi töövahendite hulka kuuluvad:

- koodi arenduskeskkonnana PhpStorm,
- versioonihaldusena Git Bash rakendus,
- kohaliku testserverina WAMP serveri teenus.

3.1 Arenduskeskkond PhpStorm

Süsteemi töövahendi valiku puhul lähtutakse eelkõige programmikeeltest ja raamistikest. Antud juhul ostus kõige mõistlikumaks ja käepärasemaks arenduskeskkonnaks PhpStorm. Kuigi süsteemi kasutus on tasuline, teeb ta projekti loomise palju lihtsamaks. Ta toetab kõige uuemaid PHP versioone. Tal on rikastatud PHP keele editor, mis jooksvalt kontrollib vigasid, koodi korrektsust ning aitab neid parandada. Lisaks PHP keelele, pakub ta veel veebipõhiste süsteemide või rakenduste loomiseks ka koodieditore Javascriptile, CSS'ile ja HTML'ile. (Features, kuupäev puudub)

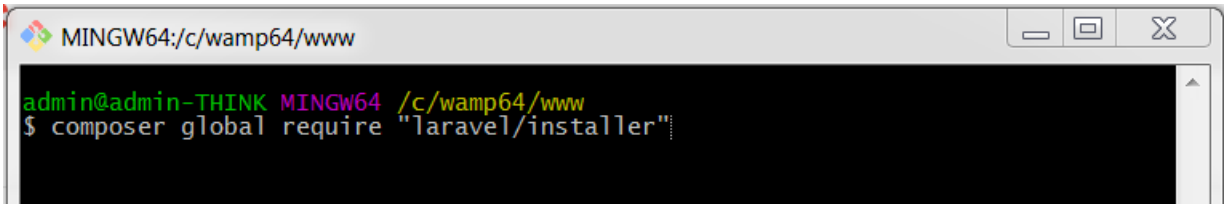
Arvestades kõiki aspekte, mis hõlbustavad süsteemi loomist, on antud keskkonnal lisaks heale kiirusele võrreldes teiste arenduskeskkondadega veel mitmeid eeliseid:

- Silumine - Kuigi on olemas laiendusi, et lisada phpunit ja XDEBUG teistele IDE-dele, muudab PHPStormi GUI seadistamise ja muutmise vaevatuks. Ei pea enam kulutama väärtuslikku aega nende ülesseadmiseks ja koodi silumiseks, sest sageli oleks see iseseisvate silumisteenuste pakkujatega vähem töökorras. Kombineerituma projekti loomiseks võib kuluda 2 või 3 minutit, kuid muude lahendustega võrreldes, pole selle üle muretsemiseks põhjust. (Lynch, 2016)

- Versioonihaldus - Kuigi on kasutusel Git Bash'i mõne GIT-i funktsiooni jaoks, muudab GIT ja SVN integratsioon phpStormis oluliselt lihtsamaks igapäevatoiminguid. Käsud nagu c*ommit, push, pull* ja *revert* on kliki kaugusel, ilma et oleks vaja mõelda täpselt, mis hetkeseis on. (Wiegman, 2014)
- Projektihaldus - projekti mugavatest sätetest kuni lugematute otsinguvõimaluste, hõlpsa navigeerimise ja suurepärase otseteedeni funktsioonide ja muutuvate määratluste leidmiseks, muudab PHP Storm projektis navigeerimise ülilihtsaks. (Lynch, 2016)
- Koodistandardid - PhpStorm kui tõeline arenduskeskkond järgib standardeid. Ühe klahvivajutuse abil suudab ta kõik vead parandada ühe faili või isegi täieliku projekti ulatuses. See on omadus, mis teeb arendamise kergemaks ja aitab hoida koodi järjepidevana ja vigadeta. (Wiegman, 2014)

3.2 Laravel algse projekti genereerimine

Laraveli kasutamiseks tuli esmalt installida arvutisse composer, mis on Laraveli raamistiku puhul nõutud. Kui Wamp server on installitud ja üles seatud, kasutatakse esmalt Git Bash rakendust Laravel'i projekti ülesseadmiseks. Selleks suundutakse Wamp serveri "www" kausta ning parema klahviga käivitatakse Git Bash rakenduse, et olla koheselt õiges kaustas, kuhu luua uus Laraveli projekt. Sinna kirjutatakse esimeseks käsk, mis laeb alla Laravel'i paigaldaja (vt pilt 3).

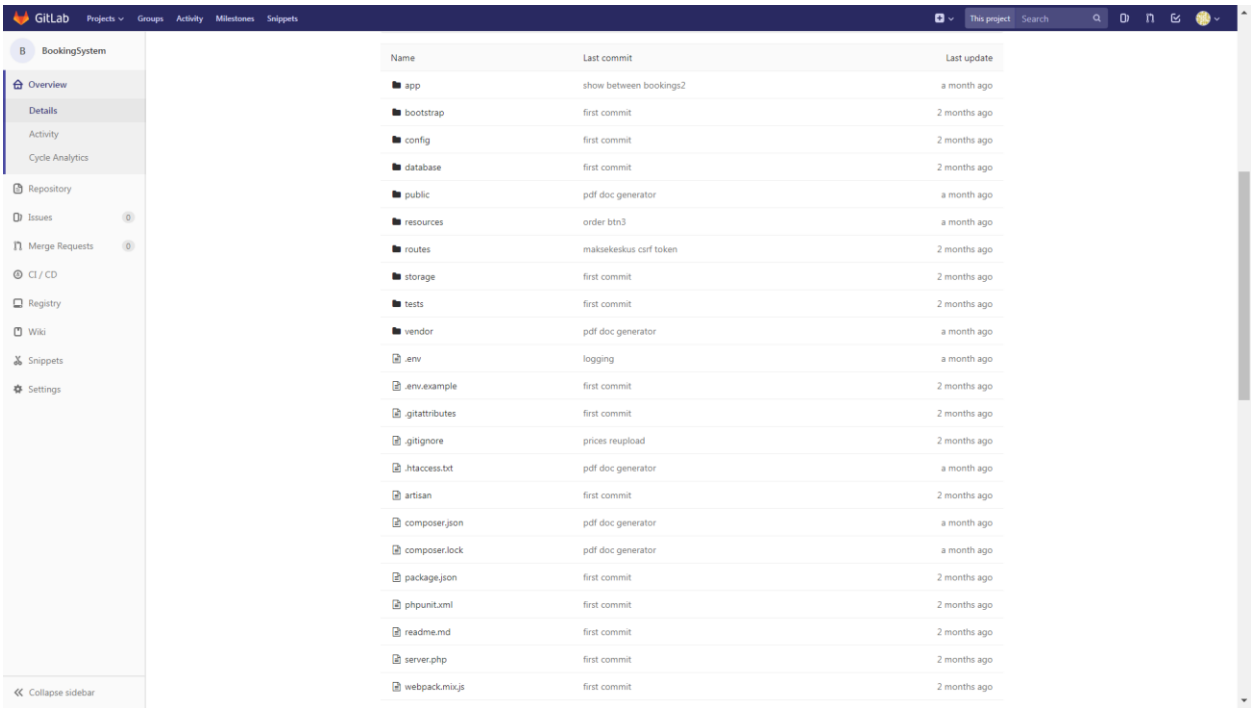


Pilt 3: Git Bash rakendus

Seejärel saab kasutada lihtsat käsku uue Laravel projekti loomiseks, mis genereerib projektikausta soovitud nimega ja installib sinna projekti tarbeks kõik vajalikud failid. Selleks kasutatakse käsku "laravel new booklux_laravel", mis loob booklux_laravel nimelise projekti kausta (vt pilt 4).

Pilt 4: Uue laraveli projekti loomise käsk

Järgnevalt genereeritakse kausta kõik projekti jaoks vajalikud põhifailid ja failipakid, mida saab vastavalt süsteemile kohendama ja konfigureerima hakata (vt pilt 5).



Pilt 5: Laraveli genereeritud projektifailid ja failipakid

3.3 Serveri konfigureerimine

Antud projekti puhul kasutatakse peale backup serveri veel kahte serverit: kohalikku serverit ja parema jõudluse tagamiseks virtuaal privaatserverit (VPS majutust). Kohalikuks serverina kasutatakse WAMP serveri teenust.

WampServer on Windowsi veebiarendusplatvorm, mis võimaldab luua dünaamilisi veebirakendusi koos Apache2, PHP ja MySQL-ga. WampServer installib automaatselt kõik, mis

on vajalik veebirakenduste arendamiseks intuitiivselt. Oma serverit on võimalik häälestada ilma, et sellega seotud faile puudutaks. WampServer on saadaval tasuta (GPML-litsentsi all) nii 32- kui ka 64-bitiste operatsioonisüsteemide versioonide puhul (Wampserver, kuupäev puudub).
Virtuaalne privaatserver (VPS) on virtuaalne server, mida kasutajad tunnevad kui privaatserverit, kuigi see on installitud mitut operatsioonisüsteemi kasutavasse füüsilisse arvutisse. (Virtual Private Server, kuupäev puudub)
Laraveli projekti tarbeks on mõlema serveri puhul vajalik uuendada apache konfiguratsioonifaili, mis määraks projekti põhikaustaks "public" kausta, mis jääb veebis ainsaks avalikult kättesaadavaks kaustaks. Wamp serveri puhul muudetakse faili httpd-vhosts.conf faili (vt koodinäidet 10).
VPS serveris luuakse kaks erinevat alamdomeeni, millest üks jääb testkeskkonnasks (test.booklux.com) ja teine tootmise (*live*) keskkonnaks (app.booklux.com).

```
# Virtual Hosts
<VirtualHost *:80>
	ServerName localhost
	DocumentRoot c:/wamp64/www/booklux_laravel/public
	<Directory  "c:/wamp64/www/booklux_laravel/public/">
		Options +Indexes +Includes +FollowSymLinks +MultiViews
		AllowOverride All
		Require local
	</Directory>
</VirtualHost>
```

Koodinäide 10: public kausta määramine projekti juurkaustaks

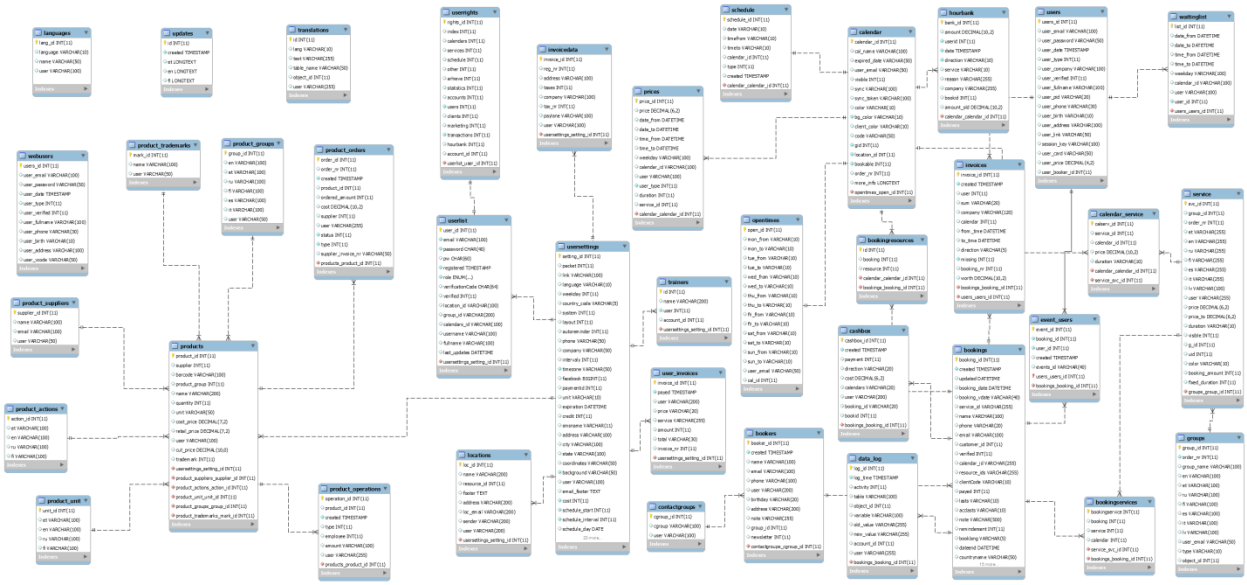
3.4 Andmebaasistruktuuri loomine ja kavandamine

Projekti puhul on andmebaasi teenuseks kasutusele võetud MySql, kuna selle teenusega on töö kirjutaja kõige enam kokku puutunud ja ka rohkem kogemusi. Ennem projekti konfigureerimisfailide kallale asumist luuakse ja kavandatakse andmebaasimudel, et andmebaas koos tabelitega üles seada. Käesoleva süsteemi puhul on vajalik luua 4 andmebaasi ning iga ühe alla 38 andmetabelit, mis on ER mudelis kujutatud (vt pilti 6). Kaks andmebaasi on testimiskeskkonna tarbeks ja ülejäänud kaks tootmiskeskkonna tarbeks. Kahest andmebaasist

kuulub üks spordivaldkonnale ja teine teiste teenusepakkujatele eelkõige ilusalongidele. Algselt arvati, et mõistlik on kaks andmebaasi eraldi hoida: eeldati, et nende valdkondade puhul jääb andmebaasistruktuur väga erinevaks ja sel viisil oleks tagatud parem töökindlus (kus ühe valdkonna arendused ja uuendused ei saaks teist mõjutada). Nüüdseks plaanitakse süsteem nii luua, et andmebaasid omavahel taas ühildada, kuid jättes võimaluse valdkonniti eraldi arendusi luua.

Andmebaasi mudeli kavandamisel tuli võtta arvesse eelkõige järgnevaid punkte:

1. Ettevõtteid peab saama lisada mitu.
2. Ühel ettevõttel saab seotud olla mitmeid väljakut, asukohti, teenuseid, teenusegruppe, tooteid, töötajaid, kliendikontosid.
3. Ühel ettevõtte kontoga peab olema võimalik sisuda teisi erinevate õigustega kasutajaid.
4. Ettevõttel saab olla mitmeid broneeringuid, kus ühe broneeringuga saab olla seotud mitu teenust ja väljakut korraga.
5. Ettevõtte broneeringute toimingutest peab säilima logi.



Pilt 6: ER andmebaasi mudel

Broneerimissüsteem loodi Laravelile arvestades küll varasemat andmebaasistruktuuri, kuid toodi ka kolm täiendust:

1. Loodi väljakut ja broneeringut seostuv tabel, et ühte broneeringut saaks üle mitme väljaku siduda ja kuvada.
2. Luuakse eraldi tabel kliendi poolt tehtavate objektide tõlgete jaoks (vt koodinäide 11). Objektideks võivad olla näiteks väljakud, paketid, paketigrupid, tooted, millele saavad kliendid ise lisada tõlkeid ükskõik millises keeles.
3. Broneeringutele lisatakse andmebaasi tabelisse juurde atribuudi parameeter. Atribuuti parameetri alla talletatakse täidavat infot broneeringute kohta, juhul kui ettevõte soovib klientidelt lisainfo küsida.

```
CREATE TABLE `translations` (
  `id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `lang` varchar(10) DEFAULT NULL,
  `text` varchar(255) DEFAULT NULL,
  `table_name` varchar(50) DEFAULT NULL,
  `object_id` int(11) DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Koodinäide 11: Sql skript tõlgete tabeli loomiseks

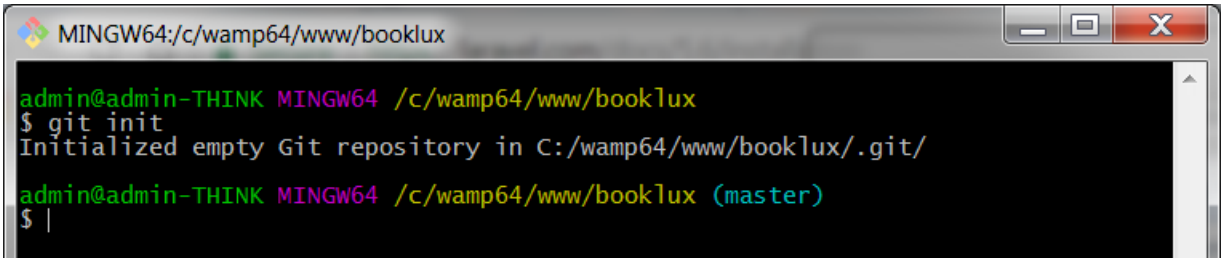
3.5 Git'i installimine ja konfigureerimine

Pärast andmebaasi loomist ja ennem koodi kirjutamist on mõistlik projektis lisaks kasutada mõnda versioonihaldustarkvara. Versioonikontrollisüsteem (VCS) võimaldab jälgida kogu failide ajalugu. Ta toetab erinevate versioonide loomist failikogudele. Iga versioon lööb failide hetkeseisu teatud ajahetkel ja VCS võimaldab nende versioonide vahel vahetada. Need versioonid on salvestatud kindlas kohas, mida tavaliselt kutsutakse hoidlateks. (Vogel, 2018)

Versiooni halduseks kasutatakse antud näite puhul Git tarkvara. Git on tasuta avatud lähtekoodiga versioonikontrolli süsteemi tööriist, mis on loodud nii, et see hõlmaks kõike, alates väikestest kuni väga suurte projektideni, tagades kiiruse ja tõhususe. See loodi Linus Torvaldsi poolt 2005. aastal Linux Kerneli arendamiseks. Gitil on funktsionaalsus, jõudlus, turvalisus ja paindlikkus, mida enamik meeskondi ja individuaalseid arendajaid vajavad. (Ahmed, 2016).

Versioonikontrolli tarkvara võimaldab saada projekti versioone, mis näitavad aja jooksul koodeksi tehtud muudatusi ning võimaldavad vajaduse korral tagasilükkamist ja nende muudatuste tühistamist (Bruce, 2012). Git annab hea võimaluse ka toote edasiseks arendamiseks, et oleks võimalik testserveris kõik tehtud uuendused ja muudatused korraga live keskkonda sisse viia.

Git kasutamiseks projektis peame esmalt ta projekti initsialiseerima käsuga „git init“ (vt pilt 7).



Pilt 7: Git initsialiseerise käsk

Töös kasutatakse projekti demonstreerimiseks Gitlab keskkonda. Selleks, et saaks kogu projekti Gitlab keskkonda üles, tuleb esmalt luua Gitlabi konto ja uus tühi projekt. Selle käigus genereeritakse url, mis tuleb käesoleva projekti git seadetes uuendada ning mille tulemusel saab vaid mõne käsuga kogu projekt Gitlab keskkonda üles laadida (vt koodinäidet 12).

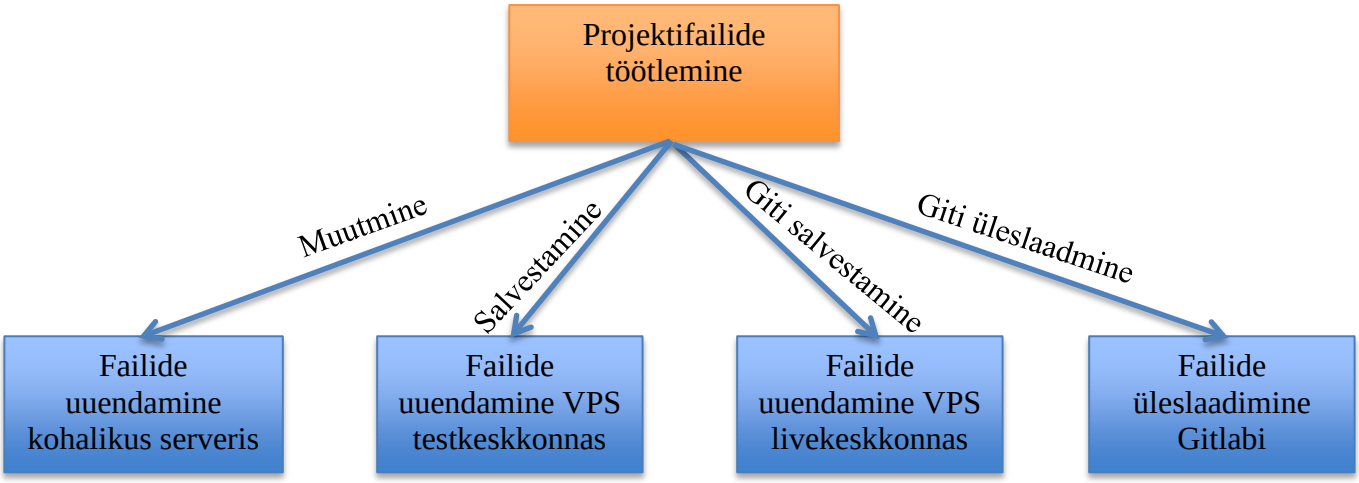
```
git config --global user.name "Meelis Koger"
git config --global user.email "info@booklux.com"
git remote rename origin old-origin
git remote add origin https://gitlab.com/booklux/BookingSystem.git
git push -u origin --all
```

Koodinäide 12: Gitile omistatakse Gitlabi keskkonna url

3.6 Failitöötlusprotsessi kujundamine

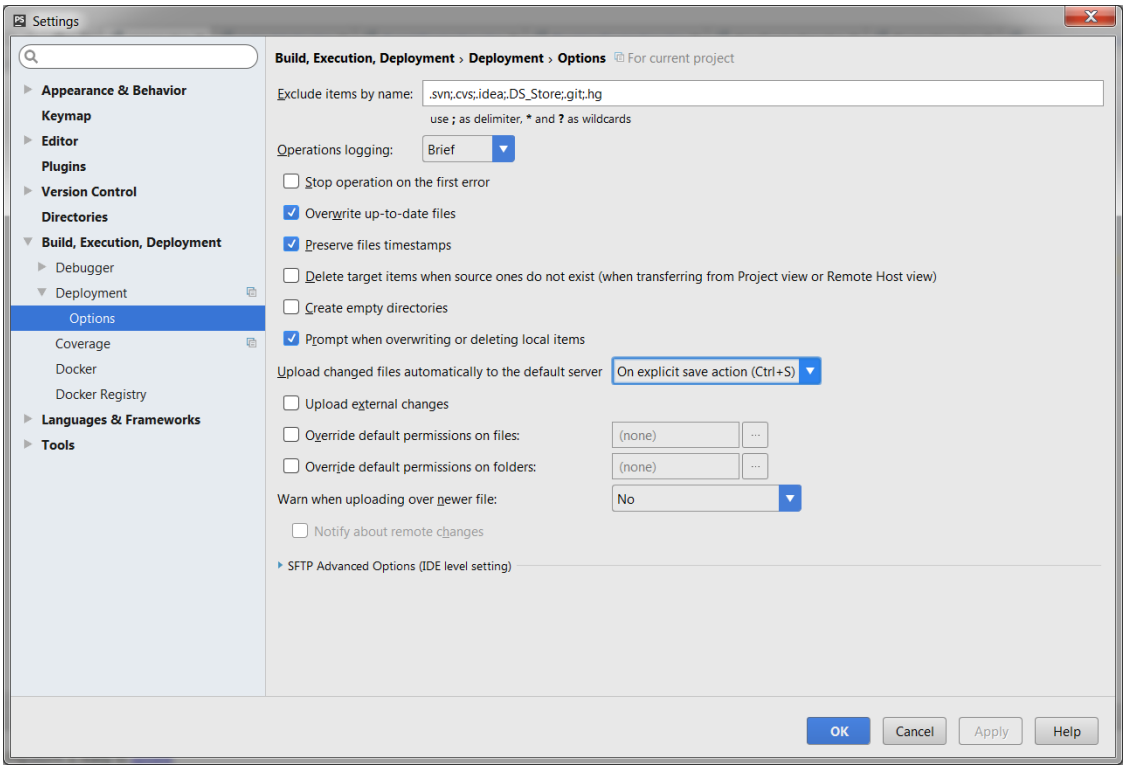
Kui projektis kasutusel versioonihaldustarkvara Git, pakub PhpStorm sellele mitmeid lisavõimalusi. Erinevate versioonide haldamist, ühe või mitme arendaja poolt loodud arenduste ühendamist ning andmevahetust mitmete serverite vahel saab lahendada erinevalt. Kogu

arendusprotsessi planeerimisel tuleks arvesse võtta nii tiimi suurust, arendusprotsessi meetmeid, projekti suurust ja kasutuses olevate serverite poolt pakutavaid võimalusi. Antud projekti arendamisega tegeleb ainult üks arendaja, süsteem nõuab pidevat täiendamist ja uuendamist (sh. kliendiarendusi) ning on oluline, et arendused saaks hästi kiirelt ja lihtsalt sisse viia, millest lähtudes kavandame sobiliku arendusmudeli (vt joonis 2).



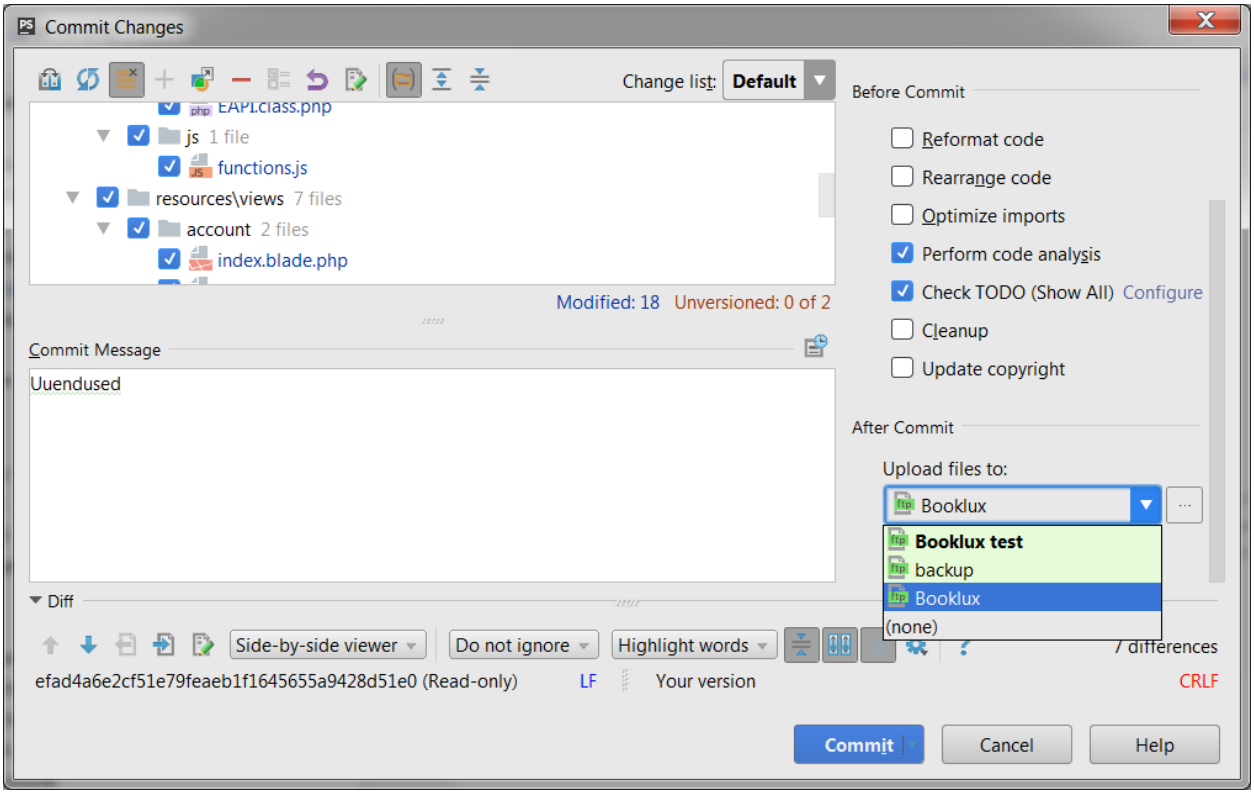
Joonis 2: Andmete liigutamise protsess

PhpStromi keskkonnas uuendatakse faile automaatselt, mille tulemusel on võimalik kohalikus serveris näha koheselt muutuse tulemust. Arenduskeskkonnas saame ka määrata, et iga (ctrl + s) salvestamise käsuga laetakse uuendus koheselt vaikeserverisse FTP vahendusel üles (vt pilt 8). Käesoleva projekti puhul määratakse vaikeserveriks VPS testkeskkond, et kliendil oleks jooksvalt võimalik salvestatud muudatusi testkeskkonnas testida.



Pilt 8: Salvestuskäsuga uuendatakse failid vaikeserveris

Teiseks saab PhpStromi keskkonda Git'i kasutamise puhul rakendada meetme, mis uue versooni salvestamisel Git'i versioonihaldusesse käsuga "commit", uuendaks kõik muudetud failid ka valitud serveris ehk VPS põhiserveris FTP vahendusel (vt pilt 9).



Pilt 9: Git commit käsuga, laetakse uuendused valitud serverisse

3.7 Projektifailide konfigureerimine

Kui andmebaas on loodud ja eelnev seadistus tehtud, saab asuda projektifailide konfigureerimise kallale, kus defineeritakse ära andmebaasi info, kasutatavate teenuste info ja kõik teised konfiguratsiooniväärtused. Laraveli põhilised konfiguratsioonifailid on lisatud konfiguratsioonikataloogi ehk “config” nimelisse kataloogi.

Esmalt defineeritakse keskkonnamuutujad, mida saab defineerida .env nimelises failis ning mille muutujaid saab kõikjal projektis kasutada. Tavapäraselt defineeritakse seal ära näiteks andmebaasi muutujad. Keskkonna muutujad on need, mis pakuvad veebirakendusele veebiteenuste loendit. Kõik keskkonnamuutujad deklareeritakse .env-failis, mis sisaldab konfiguratsiooni initsialiseerimiseks vajalikke parameetreid (vt koodinäide 13).

(Laravel – Configuration, kuupäev puudub)

```
APP_ENV = local
APP_DEBUG = true
APP_KEY = base64:ZPt2wmKE/X4eEhrzJU6XX4R93rCwYG8E2f8QUA7kGK8 =
APP_URL = http://localhost
DB_CONNECTION = mysql
DB_HOST = 127.0.0.1
DB_PORT = 3306
DB_DATABASE = homestead
DB_USERNAME = homestead
DB_PASSWORD = secret
MAIL_DRIVER = smtp
MAIL_HOST = mailtrap.ioMAIL_PORT = 2525
MAIL_USERNAME = null
MAIL_PASSWORD = null
MAIL_ENCRYPTION = null
```

Koodinäide 13: Põhilised keskkonna muutujate konfigureerimise paramteerid (Laravel – Configuration, kuupäev puudub)

Failis .env deklareeritud keskkonnamuutujaid saab kasutada env() tugifunktsioonidega, mis kutsuvad vastavat parameetrit. Need muutujad on samuti loetletud $ _ENV globaalsesse muutujateni, kui rakendus saab kasutaja lõppkasutajalt päringu. (Laravel – Configuration, kuupäev puudub)

3.8 Süsteemilehtede loomine

Iga halduskeskkonna veebilehe jaoks on eraldi kontroller ja template fail. Iga halduskeskkonna päring suunatakse esmalt halduskeskkonna konto kontollerile (vt koodinäide 14). Selles kontrolleris tehakse esmane autentimine, päringu kontroll ning õiguste olemasolul küsitakse välja põhiline kliendiinfo.

```
Route::match(array('GET', 'POST'),'/account/{view}', 'AccountController@view');
```

Koodinäide 14: Kontokontrolleri käivitamine vastavalt urlile

Vastavalt urli viimase parameetri väärtusele avab ta samanimelisele template faili, kuhu ta kaasab lisaks samanimelise kontrolleri poolt päritava info juhul, kui sellise nimega kontoller

eksisteerib ning kasutajal on õigust seda näha (vt koodinäide 15). Näiteks kui url on https://app.booklux.com/account/schedule, avab konto kontroller "schedule.blade.php" nimelise template faili ning lisaks kaasab ScheduleController'I faili seest päritava info, mis on just sellel lehel tarvis. Sama loogika alusel kuvatakse ka kliendivaate info, kus vastavalt uril viimase parameetri alusel kuvatakse kindla ettevõtte kliendivaateinfo.

```
public function getViewData($view){
       $viewClassName= "\\App\Http\Controllers\Account\\".ucfirst($view)."Controller";
       if(class_exists($viewClassName)){
           $classData = new $viewClassName($this->data, $_GET, $_POST);
           $this->data = $classData->data();
       }
       return view('account.'.$view, ["view"], $this->data);
}
```

Koodinäide 15: Meetod, mis vastavalt uril nimetusele käivitab kontrolleri faili ja avab template faili

Igas kontrolleris kasutatakse __construct metaprogrammeerimise funktsiooni, mis initsialiseerib uue kontrolleri objekti loomisel esmalt vanema kontrolleri ning kontrollib esmalt, kas kontrollerile on saated mõni *POST* või *GET* päring meetodiga "checkRequest" (vt koodinäide 16).

```
public function __construct($data = []) {
        parent::__construct();
        $this->data = $data;
        $this->checkRequests();
}
```

Koodinäide 16: __construct metaprogrammerimise meetod, mis kasutatakse iga halduskeskkonna kontolleri puhul.

Süsteemi loomisel arvestati ka erinevate valdkondade ja nende erisusega. Kuna süsteemi puhul soovitakse, et statistikaleht oleks spordivaldkondadel erinev võrreldes tavapärase statistikaga, mis on sobilik ilusalongidele, saab luua "custom" kausta alla eraldi template faili näiteks sport_statistics.blade.php. Kontroller esmalt kontollib, kuhu kategooriasse sisselogitud kasutaja kuulub ning seejärel vaatab, kas vastava kategooria nimeline vaatefail "custom" kasutas eksisteerib (vt koodinäide 17).

```
if(View::exists('custom.'.$_COOKIE["db"]."_".$this->data["view"])){
    $this->data["view"] = 'custom.'.$_COOKIE["db"]."_".$this->data["view"];
}
```

Koodinäide 17: Funktsiooniosa mis kontrollib kategooriapõhise vaatefaili olemasolu

3.9 Android ja iOS rakendused

Põhjusel, et haldussüsteem on ehitatud erinevates seademetes kohanduvalt, kasutades Bootstrap raamistikku, saame luua veebivaatepõhised Android jai iOS rakendused. Veebivaatepõhise rakenduse loomine puhul kuvatakse täisekraanil veebilehitsejat, mida saab implementeerida WebView objektina ning määrata juurde veebiaadress, mida ta veebilehitseja sees avab (vt koodinäide 18).

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        WebView myWebView = (WebView) findViewById(R.id.liveview);
        myWebView.setWebViewClient(new MyWebViewClient());
        myWebView.setWebChromeClient(new WebChromeClient());
        WebSettings webSettings = myWebView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        webSettings.setDomStorageEnabled(true);
        String language = Locale.getDefault().getLanguage();
        language = language == "" ? "et" : language;
        myWebView.loadUrl("https://app.booklux.com/account/login?applang="+language);
    }
}
```

Koodinäide 18: Failide puhverdamise skript

Rakenduse teiseks eesmärgiks saata kliendile teateid, juhul kui süsteemi tekib uus broneering või broneeringu tühistamine. Ennem kui me saame kasutajale teateid saata, on vaja omavahel ära siduda kindla telefoni broneerimissüsteemi kasutajakontoga, teadmaks, millisele telefonile

teateid saata. Selle tarbeks on soovituslik kasutada javascripti silda android rakenduse ja veebirakenduse vahel. Kui klient avab rakenduse ja logib süsteemikontosse sisse, käivitatakse veebirakenduses funktsioon, millele antakse kaasa telefoni identifitseerimiseks kasuatav Firebase ID (vt koodinäide 19). Firebase ise on mobiili- ja veebirakenduste arendusplatvorm, mis pakub arendajatele hulgaliselt tööriistu ja teenuseid, et aidata neil arendada kvaliteetseid rakendusi. (Rajat, 2017)

```
private class MyWebViewClient extends WebViewClient {
        @Override
        public void onPageFinished (final WebView view, String url){
            view.loadUrl("javascript:notificationId('"+tokenId.toString()+"')");
        }
}
```

Koodinäide 19: Failide puhverdamise skript

4 Võimalused süsteemi arendamisel

Laravel pakub hulga võimalusi, mida ei saa antud töö raames kasutada või nõuab süsteemi puhul veel eelnevat arendust. Kuna antud näitel on varem loodud projekt üritatud tuua Laravelile üle, ei ole võimalik koheselt kasutusele võtta parimaid meetmeid, sõltuvalt varasemast andmebaasistruktuurist ja ajaressursi puudusest. Projektile läheneti agiilse arendusmeetmega, muutes samm-sammu haaval süsteemi paremaks. Eesmärk oli saada võimalikult kiiresti süsteem Laravelil raamistikul tööle, mistõttu üritati leida kõige optimaalsemaid lahendusi.

4.1 Eloquent ORM mudel

Hetkel kasutatakse süsteemis andmebaasi päringuteks ainult “Query Builderit” mudelite asemel. Varasemas projektis ei kasutatud mudeleid, mistõttu aja kokkuhoiu suhtes otsutas töökirjutaja mudelite loomise hilisemaks arenduseks jätta.

Laraveliga kaasnev Eloquent ORM pakub andmebaasiga töötamiseks lihtsat ActiveRecordi rakendust. Igal andmebaasi tabelil on vastav "mudel", mida kasutatakse selle tabeliga suhtlemiseks. Mudelid võimaldavad kasutajal oma tabelites andmeid küsida ja tabelisse uued andmed lisada. (Eloquent: Getting Started, kuupäev puudub).

Query Builderit on mõistlik siiski kasutada suuremahuliste andmete pärimiseks, kuna ta on kiirem. Samuti keerukate andmebaasipäringute tarbeks, sest mudelid tihtipeale nõuavad mitut päringut, ega ei saa ühe päringuga mitmest tabelist korraga omavahel seostatavat infot võtta. (Geladaris, 2014)

4.2 Autentimine

Laraveli autentimisvõimalused on oma põhiosas "valvurid" ja "pakkujad". Valvurid määratlevad, kuidas kasutajad iga päringu jaoks autentida. Näiteks Laravel laeb seansi valvuriga, mis säilitab oleku, kasutades seansi mälu ja küpsiseid.

Pakkujad määratlevad, kuidas kasutajad püsivast salvestusest infot saavad. Laravel võimaldab kasutajate leidmiseks kasutada Eloquenti mudelit ja andmebaasi päringute loojat. Siiski säilib õigus määratleda täiendavaid pakkujaid, kui see on kasutaja rakendusele vajalik. Autentimisfunktsioonifail asub aadressil "config/auth.php", mis sisaldab mitmeid hästi dokumenteeritud võimalusi autentimisteenuste käitumise tõhustamiseks. (Authentication, kuupäev puudub)

Laravel muudab rakendamise autentimise väga lihtsaks. Tegelikult on peaaegu kõik konfigureeritud. Laravel pakub kiiret viisi kõikide autentimiseks vajalike liinide ja vaadete genereerimiseks, kasutades ühte lihtsat käsku: "php artisan make:auth". (Authentication, kuupäev puudub)

Projekti puhul on seda käsku kasutatud, kuid süsteemi struktuuri tõttu pole Laraveli poolt pakutavat autentimissüsteemi rakendanud. Üks käsk genereerib kõik vajalikud funktsioonid ja vaadet, mille kaudu saab sisse logida, registreerida, parooli meeldetuletust küsida. Üks pealmisi põhjuseid, miks seda kasutada, on selle turvalisus ning teeb mugavaks kõikidele süsteemilehtedele autentimiskihti juurde lisada, kus see on vajalik.

4.3 API loomine

Üheks mudelite kasutamise eeliseks on lihtsama vaevaga API loomine. API on vajalik ja perspektiivne lähenemine personaalsete lahenduste loomisel või mõnede teiste süsteemidega ühildamisel.

API-liidesed (rakendusprogrammide liidesed) pakuvad võimalust arvuti tarkvara komponentide ühendamiseks. Laialdased API-d võimaldavad organisatsioonidel avada oma taustandmeid ja funktsioone uute rakenduste teenuste taaskasutamiseks. (API strategy 101, kuupäev puudub)

API koostamisel võib tekkida vajadus ümberkujundamise kiht, mis asetseb sinu mudelite ja JSONi vastuste vahel, mis on rakenduse kasutajatele tegelikult tagastatud. Laraveli ressursiklassid võimaldavad ekspressiivselt ja lihtsalt muuta oma mudelid ja mudelikollektsioonid JSONiks. (Eloquent: API Resources, kuupäev puudub)

API loomisel tuleb lisaks arvestada kliendi autentimisega. Laravel pakub ka ise API autentimise lahendust Laravel Passport'i, kasutades täieliku OAuth2-serveri rakendust kasutaja Laraveli rakenduse jaoks mõne minuti jooksul (API Authentication: passport, kuupäev puudub). OAuth on tuntud kui turvaline, kolmanda osapoole kasutajaagent, delegeeritud volitus. (Roger, 2017) Broneerimisssüsteemi API loomise puhul ei ole aga vajalik ega mõistlik kasutada kolmanda osapoole kasutajaagenti, kuna see teeb API'ga ühildamise palju tulikamaks, vaid plaanitakse kasutada API võtme genereerimise, kasutajatunnuse ja parooli kaudu, mida kliendid audentimiseks kasutada saavad.

5 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli uurida ja leida mitmeid erinevaid mooduseid, kuidas võiks kavandada ja arendada suuremamahulist rakendust Laraveli raamistiku toel ning loodi kogu broneerimissüsteemi halduskeskkond ja kodulehele integreeritav kliendivaade. Süsteemi planeerimisel tuleb mõelda mitmetele aspektidele seal hulgas töövahendite valimisele, varundamisele, turvalisuse tagamisele, kogu arendusprotsessile, kiiruse ja stabiilsuse tagamisele ning kõigele, mis on antud süsteemi puhul oluline.

Töö käigus selgus, et broneerimissüsteemi ülekolimine ja rakendamine Laraveli raamistikule oli mõistlik ja perspektiivne ettevõtmine. Laravel on üks populaaarseimaid PHP veebiraamistikke, mis teeb arendamise oluliselt mugavaks, pakkudes mitmed võimalusi ja liideseid süsteemi kiiremaks ülesseadmiseks ja tulevasteks arendusteks. Vaid vähese vaevaga saab luua kogu failide üleslaadmise ja talletamise protsessi, turvalise autentimissüsteemi ning teavitussüsteemi mitmesuguste kanalite tarbeks. Mugavaks on tehtud ka Eloquent ORM mudelite ja API loomine, mida tulevikus plaanitakse süsteemile rakendada.

Laraveli projekti loomine käib lihtsasti, kus mõne käsuga saab genereerida kõik projektiks vajaminevad failid ning neid vastavalt projektile konfigureerima hakata. Kogu protsessi käigus teadvustas töö autor, kui oluline on suuremal projektil planeerida ja analüüsida algselt korralikult läbi kogu andmebaasistruktuur. Hilisem andmebaasimudeli muutmine, töötlemine ja ühendamine on väga keeruline ja riskantne tegevus eelkõige juhul, kui andmemahud on suured. Kuna broneerimissüsteemi kasutab juba paarkümmend rahulolevat klienti, võib töö tulemusega rahule jääda.

6 Summary

The aim of this bachelor thesis was to explore and find various ways in which a large-scale application could be organized and developed with the support of the Laravel framework. Furthermore, entire booking management system and also client view for booking system which can be integrated on the homepage were created. When planning a system, there should be considered a number of aspects including selection of tools, backup system, security, the entire development process, speed and stability and everything that is important for this application.

In the process we revealed that the transfer and implementation of the reservation system to the Laravel framework was a reasonable and perspective undertaking. Laravel is one of the most popular PHP web frameworks, which makes the development considerably comfortable, providing several options and tools for faster system setup and future development. With just a little effort, you can create a complete file uploading and storaging process, a secure authentication system, and a notification system for various channels. The creation of Eloquent ORM models and APIs that are planned to be implemented in the future is also convenient.

Creating a Laravel project is quite easy, with few commands, you can generate all the files needed for the project and start configuring them according to the project. Throughout the process, the author acknowledged that it is important for the larger project to initially plan and analyze the entire database structure thoroughly. Changing, processing, and merging a database model is a very tricky and risky activity, especially if data volumes are large. As the reservation system has already been used by a dozen satisfied customers, the author is satisfied with the result of the work.

7 Kasutatud kirjandus

Ahmed, R. (2016, 16.november). *What Is Git ? – Explore A Distributed Version Control Tool.* Loetud aadressil https://www.edureka.co/blog/what-is-git/

API Authentication (Passport). (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.6/passport

API Strategy 101:What is an API? (kuupäev puudub). Loetud aadressil http://www.apiacademy.co/resources/api-strategy-lesson-101-what-is-an-api/

Authentication. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.3/authentication

Bootstrap. (kuupäev puudub). Loetud aadressil https://getbootstrap.com/

Bruce, J. (2012, 3.oktoober). *What Is Git & Why You Should Use Version Control If You're a Developer.* Loetud aadressil https://www.makeuseof.com/tag/git-version-control-youre-developer/

CSRF Protection. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.6/csrf

Database: Query Builder. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.6/queries

Eloquent: API Resources. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.6/eloquent-resources

Eloquent: Getting Started. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.6/eloquent

Features. (kuupäev puudub). Loetud aadressil https://www.jetbrains.com/phpstorm/features/

File Uploads. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.3/filesystem#file-uploads

Geladaris, G. (2014). *Eloquent vs. Raw SQL? Which is really better?* Loetud aadressil https://laravel.io/forum/04-23-2014-eloquent-vs-raw-sql-which-is-really-better

History of Laravel. (kuupäev puudub). Loetud aadressil https://www.w3schools.in/laravel-tutorial/history/

Laravel Configuration. (kuupäev puudub). Loetud aadressil https://www.tutorialspoint.com/laravel/laravel_configuration.htm

Laravel Philosophy. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/4.2/introduction

Lynch, C. (2016, 21.märts). *5 reasons why PHP Storm is my go-to IDE*. Loetud aadressil http://blog.helastel.com/5-reasons-why-php-storm-is-my-go-to-ide

Rajat, S. (2015, aprill). *Introduction to Firebase.* Loetud aadressil https://hackernoon.com/introduction-to-firebase-218a23186cd7

Release Notes. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.3/releases

Roger, A. (2017, 16.august). *What is OAuth? How the open authorization framework works*. Loetud aadressil https://www.csoonline.com/article/3216404/authentication/what-is-oauth-how-the-open-authorization-framework-works.html

Rouse, M. (2015, aprill). C*ache (computing).* Loetud aadressil https://searchstorage.techtarget.com/definition/cache

Sending Notifications. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.3/notifications#sending-notifications

Stauffer, M. (2015, 19.detsember). *API rate limiting in Laravel 5.?* Loetud aadressil https://mattstauffer.com/blog/api-rate-limiting-in-laravel-5-2/

Strauss, V. (2017, 26.juuli). *Mis on SSL sertifikaat ning miks seda vaja on?* Loetud aadressil https://www.voog.com/blogi/mis-on-ssl-sertifikaat-ning-miks-seda-vaja-on

The Loop Variable. (kuupäev puudub). Loetud aadressil https://laravel.com/docs/5.3/blade#the-loop-variable

Virtual Private Server (VPS) . (kuupäev puudub). Loetud aadressil https://www.techopedia.com/definition/4800/virtual-private-server-vps

Vogel, L. (2018). *What is a version control system?* Loetud aadressil http://www.vogella.com/tutorials/Git/article.html#versioncontrolssystems

Wampserver. (kuupäev puudub). Loetud aadressil https://sourceforge.net/projects/wampserver/

Wiegman, C. (2014, 13.mai). *Sublime Text to PhpStorm – Why I Switched.* Loetud aadressil https://www.chriswiegman.com/2014/05/sublime-text-phpstorm-switched/

What Is CodeGuard? (kuupäev puudub). Loetud aadressil http://support.hostgator.com/articles/offers-bonuses/what-is-codeguard