KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutus

Riku Tiitta

GRITPRO-KYSELYSOVELLUKSEN JATKOKEHITTÄMINEN PLAN&CARE SPORTS OY:LLE

Opinnäytetyö
Helmikuu 2021

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Helmikuu 2021
Tietojenkäsittely

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä
Riku Tiitta

Nimeke
GritPro-kyselysovelluksen jatkokehittäminen Plan&Care Sports Oy:lle

Toimeksiantaja
Plan&Care Sports Oy

Tämän opinnäytetyön tavoitteena oli suorittaa toimeksiantajan vaatimusten mukainen jatkokehitys GritPro-kyselysovellukseen. Kyselysovellus oli toteutettu web-sovelluksena. Kyselysovelluksen avulla valmentajat voivat lähettää joukkueelle kyselyitä ja luoda niiden vastauksista raportteja. Kyselysovellukseen haluttiin kolme uutta ominaisuutta, jotka olivat kyselyiden luonti, vastaamisen laajentaminen sekä kyselyiden tietojen listaaminen.

Sovellus oli toteutettu käyttäen Laravel- ja Vue.js-sovelluskehyksiä, joten näiden sovelluskehysten avulla myös jatkokehitys suoritettiin. Kehitys suoritettiin full-stack-tyyppisenä kehityksenä. Työssä tarkasteltiin sovelluskehyksiä sekä niiden käytettävyyttä nykyaikaisessa web-sovelluksen kehityksessä.

Opinnäytetyön tuloksena syntyi vaatimusmäärittelyn mukainen jatkokehitys GritPro-kyselysovellukseen. Jatkokehityksen ansiosta käyttäjät voivat luoda uusia kyselyitä järjestelmään, kyselyihin voidaan vastata laajemmin sekä käyttäjät voivat nähdä kyselyiden tiedot halutessaan.

Kieli
suomi

Sivuja 41

Asiasanat
ohjelmointi, ohjelmistokehitys, Vue.js, Laravel

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
February 2021
Degree Programme in Business Information Technology

Tikkarinne 9
80200 JOENSUU
FINLAND
+ 358 13 260 600 (switchboard)

Author
Riku Tiitta

Title
Further development of the GritPro questionnaire application for Plan&Care Sports Oy

Commissioned by
Plan&Care Sports Oy

The goal in this thesis was to develop further GritPro questionnaire web-application based on requirements set by the commissioner. The GritPro web-application allows coaches to send questionnaires to their teams and create reports based on the athletes answers. The commissioner wanted three new features in the application, and these were: creating new questionnaires, expanding the answering to the questionnaires and ability for users to view the questionnaire data.

The application was developed using Laravel and Vue.js web-frameworks, which were also used in the further development of the application. The development was performed as full-stack style development. In this thesis the web-frameworks were also examined and their usefulness in modern web-application development.

Thesis resulted in three new features in the application which allows users to create new questionnaires, expand the answering to the questionnaires and allow listing of questionnaires data when user needs it.

Language
Finnish

Pages 41

Keywords
programming, software development, Vue.js, Laravel

Sisältö

Lyhenteet

Back-end	Myös tunnetaan nimellä server-side, suomeksi palvelinpuoli, jolla viitataan ohjelmiston osaan, joka toimii palvelimella eikä ole käyttäjälle näkyvä (Sagara Technology 2020).
CSS	Cascading Style Sheets, tyylityskieli, jonka avulla muokataan dokumentin näkyvää ulkoasua. Käytetään esimerkiksi HTML-dokumenttien ulkoasun muokkaamiseen. (Christensson 2006a.)
DOM	Document Object Model, kuvaa web-dokumentin rakennetta ja sen sisältöä siten, että ohjelma voi sitä muokata (Mozilla 2021).
Front-end	Myös tunnetaan nimellä client-side, suomeksi käyttäjäpuoli, jolla viitataan ohjelmiston osaan, jonka käyttäjä näkee ja voi olla vuorovaikutuksessa sen kanssa (Sagara Technology 2020).
Full-stack	Ohjelmistokehityksen muoto, jossa kehittäjä kehittää front-end- sekä back-end-osuuksia sovelluksesta.
HTML	Hypertext markup language, merkintäkieli, jonka avulla luodaan dokumentteja, jotka esitetään verkkoselaimessa (Christensson 2015a).
HTTP	Hypertext Transfer Protocol, käytetään siirtämään tietoa pyyntöjen avulla palvelimen ja verkkoselaimen välillä verkkosivustoilla. Protokolla määrittää pyyntöjä kuten GET ja POST. (Christensson 2015b.)
JavaScript	Verkkoselaimissa käytetty ohjelmointikieli, jonka avulla mahdollistetaan vuorovaikutteiset sekä dynaamiset verkkosivut (Christensson 2014).
Modal	Käytetään myös nimeä modal-ikkuna. Verkkosivuilla käytetty teknologia, jolla uusi ikkuna sijoitetaan pääikkunan päälle yleensä jättäen pääikkunan taustalle näkymään.
MV*	Käytetään kun puhutaan yleensä eri Model-View-ohjelmistoarkkitehtuurimalleista.
MVC	Model-View-Controller, suomeksi malli-näkymä-ohjain. Ohjelmistoarkkitehtuurin malli, jota hyödynnetään sovellusten käyttöliittymien luomisessa (Christensson 2018).

MVVM	Model-View-ViewModel, suomeksi malli-näkymä-näkymämalli. Ohjelmistoarkkitehtuurin malli, joka hyödyntää näkymämallia luodakseen helpomman tavan kommunikointia varten (Likness 2014).
PHP	Hypertext Preprocessor, verkkosovellusten kehitykseen käytetty ohjelmointikieli, joka mahdollistaa dynaamisen dokumentin luonnin palvelimen puolella (Christensson 2006b).
SaaS	Software as a Service, verkkosovelluksen tyyppi, yleensä käytetään sovelluksissa, joissa sama sovellusympäristö palvelee useita asiakkaita. Tämän tyyppiset sovellukset yleensä toimivat web-selaimessa. (Christensson 2011.)
SQL	Structured Query Language, tietokantojen tiedon ohjelmointia ja hallitsemista varten käytetty ohjelmointikieli (Christensson 2007).

1 Johdanto

Tässä opinnäytetyössä jatkokehitetään GritPro-nimistä kyselysovellusta. Työ on tehty toimeksiantona Plan&Care Sports Oy -yritykselle. Sovellus on rakennettu käyttäen Vue.js- ja Laravel-sovelluskehyksiä, joiden avulla sovellusta myös jatkokehitettiin. Kyselysovellus on osa isompaa sovelluskokonaisuutta. Plan&Care Sports Oy tarjoaa sovellusta, joka sisältää useita eri työkaluja valmentajien ja urheiluseurojen käyttöön. Sovelluksen avulla voidaan esimerkiksi suunnitella joukkueen harjoitteita ja aikatauluttaa niitä. Yritys toimii kansainvälisesti ja sillä on asiakkaita jo useasta eri maasta. Sovellus toimii SaaS-periaatteella eli palveluna verkon välityksellä. Tämä tarkoittaa, että sama sovellusympäristö palvelee kaikkia asiakkaita. Sovellus toimii verkkoselaimilla useilla eri laitteilla.

GritPro-kyselymoottorin avulla valmentajat pystyvät lähettämään kyselyitä urheilijoille ja sekä saamaan paremman kuvan urheilijoista, joukkueesta ja koko urheiluseuran toiminnasta. Kyselymoottori sisältää valmiiksi urheilupsykologian ammattilaisten luomia kyselyitä. Uusien kyselyiden luominen tällä hetkellä on hankalaa ja vaatii paljon työtä, joten opinnäytetyöni keskittyy tekemään tästä helpompaa. Valmista järjestelmää kyselyiden luomiseksi tulevat käyttämään yrityksen asiakkaat, kuten urheiluseurat ja urheilupsykologian ammattilaiset. Kyselymoottori sisältää muutamia valmiita kyselykomponentteja, joiden avulla voidaan arvioida jotakin esimerkiksi asteikolla 1–7 tai sitten vastata käyttäen tekstiä. Kyselykomponenttien avulla voidaan sitten rakentaa kokonainen kysely. Kyselyn vastausten datasta voidaan sitten luoda monenlaisia raportteja, joiden avulla nähdään kyselyn tulokset.

Käyn myös läpi Vue.js- ja Laravel-sovelluskehysten ominaisuuksia sekä lyhyen historian ja niiden hyödyt, kun puhutaan verkkosovellusten nykyaikaisesta kehityksestä. Vertailen myös lyhyesti Vue.js- ja Laravel-sovelluskehyksiä muutamiin niiden kanssa kilpaileviin sovelluskehyksiin.

2 Web-sovelluskehykset

2.1 Mikä on web-sovelluskehys?

Sovelluskehyksiä on jo olemassa suuri määrä ja niitä kehitetään jatkuvasti lisää. Sovelluskehykset luodaan yleensä helpottamaan ja nopeuttamaan ohjelmistokehitystä. Sovelluskehykset rakennetaan jo ennalta olemassa olevan ohjelmiston taikka ohjelmointikielien päälle, jonka avulla niistä voidaan luoda helppokäyttöisempi kehittäjille. Sovelluskehysten avulla voidaan automatisoida muuten kehittäjälle hankalat tai aikaa vievät ominaisuudet web-sovelluksen kehityksessä. (Ryabtsev 2020.)

Sovelluskehykset yleensä jaetaan kahteen ryhmään: front-end- ja back-end-sovelluskehyksiin. Front-end-sovelluskehyksiä myös kutsutaan nimellä client-side-sovelluskehys, koska se sijaitsee käyttäjän web-selaimessa. Back-end-sovelluskehyksiä kutsutaan sitten server-side-sovelluskehys nimityksellä, koska ne sijaitsevat palvelimella käyttäjän selaimen sijasta. (Ryabtsev 2020.)

2.2 Front-end-sovelluskehykset

Front-end-sovelluskehykset vastaavat käyttäjille näkyvästä sivustosta ja sen käyttöliittymästä ja ne sijaitsevat käyttäjän selaimessa. Front-end-sovelluskehysten avulla voidaan käyttäjälle esittää monenlaista eri tietoa sekä saada käyttäjältä syötetietoa, jota voidaan sitten välittää palvelimelle. (Ryabtsev 2020.)

Esimerkkeinä front-end-sovelluskehyksistä ovat Vue.js, React ja Angular. Front-end-sovelluskehykset yleensä rakennetaan käyttäen HTML-, CSS- ja JavaScript-ohjelmointikieliä (Sagara Technology 2020). Front-end-sovelluskehyksiä hyödynnetään etenkin dynaamisten web-sivustojen luonnissa, koska ne tarjoavat monia eri työkaluja tähän.

2.3 Back-end-sovelluskehykset

Back-end-sovelluskehykset sisältävät paljon enemmän vaihtelua verrattuna front-end-sovelluskehyksiin. Back-end-sovelluskehykset sijaitsevat palvelimella eivätkä juurikaan näy käyttäjälle. Back-end-sovelluskehykset eivät rajoitu tiettyihin ohjelmointikieliin, vaan niitä voidaan kehittää monilla eri ohjelmointikielillä. Back-end-sovelluskehykset sisältävät paljon eri toimintoja, mutta yhtenä niiden päätoimintoja on lähettää käyttäjän selaimessa näytettävä sivusto. Back-end-sovelluskehykset myös vastaavat tietokannoista ja niiden kanssa kommunikoinnista. (Kaluža, Kalanj & Vukelić 2019.)

Esimerkkeinä muutamista ohjelmointikielistä, joita käytetään back-end-sovelluskehyksissä ovat, Ruby, Java, Python, PHP ja Node.js (Sagara Technology 2020). Back-end-sovelluskehyksiä on paljon, mutta muutamana esimerkkinä ovat Laravel, Ruby on Rails ja Django.

3 Vue.js-sovelluskehys

3.1 Perusta

Vue.js on front-end-JavaScript-sovelluskehys, joka on suunniteltu kevyeksi sekä helppokäyttöiseksi. Vue.js suunniteltiin alusta alkaen mahdollisimman joustavaksi. Sen avulla voidaan luoda kokonaisia sivustoja, kun se yhdistetään muiden työkalujen kanssa. Vue.js keskittyy ytimessään käyttäjille näkyvään kerrokseen. (Vue.js 2020a.)

Vue.js-sovelluskehys sai alkunsa vuonna 2013. Kehittäjä Evan You sai idean työskennellessään Google Creative Labsilla, kun yritys tarvitsi nopeasti kehitettävää prototyyppiä ja työkalua tällaisen luomiseksi ei vielä löytynyt markkinoilta. Yrityksen tarkoitukseen työkalun täytyi olla kevyt ja hyvin muokattava nopean käyttöliittymän muokkaamiseen, ja markkinoilla olleet työkalut eivät tähän taipuneet. (Filipova 2016, 10.) Vue.js ensimmäinen GitHub lisäys tapahtui 2013 heinäkuussa (Github 2020).

Vue.js:n ensimmäinen julkaisu tapahtui 2014 huhtikuussa. Evanilla oli markkinointisuunnitelma valmiina projektin julkisuuteen saamiseksi, vaikka markkinoilla oli jo useita vastaavia projekteja. Suunnitelma sisälsi useita eri kanavia, joiden avulla hän sai projektinsa asiakaskunnan tietoon. Kanavia olivat esimerkiksi Hackernews, Reddit ja EchoJS. Julkaisupäivänä Vue.js:n verkkosivu keräsi yli 10 000 uniikkia kävijää. (You 2014.) Vue.js eroaa sen kilpailijoista tänäkin päivänä sillä, että se on yksityisen henkilön ylläpitämä eikä saa tukea suurilta yrityksiltä, kuten sen kilpailijat esimerkiksi React ja Angular. Tässä opinnäytetyössä keskitytään Vue.js:n versioihin 2.0–2.6. Vue.js:n versiosta 2.0 tiedotettiin ensimmäisen kerran 2016 huhtikuussa (Vue.js 2016).

3.2 Kilpailevat sovelluskehykset

3.2.1 React

React on Vue.js:n kaltainen JavaScript-sovelluskehys. React ja Vue.js sisältävät toistensa kanssa hyvin samanlaisia ominaisuuksia. Molemmat käyttävät virtuaalista DOMia, tarjoavat reaktiivisia komponentteja ja keskittyvät pääasiassa näkymään. Molemmat tarjoavat erillisiä kirjastoja reittejä ja muita ominaisuuksia varten. Vue.js on optimoitu päivittämään vain tietyt komponentit, jos instanssissa tapahtuu muutos, kun taas React päivittää kaikki komponentit juurikomponentista lähtien. React myös tarjoaa tähän vaihtoehtoja, joiden avulla voidaan parantaa suorituskykyä. (Vue.js 2020b.)

Vue.js käyttää malleja, joiden avulla voidaan kirjoittaa vapaasti HTML-koodia näkymässä. React käyttää taas JSX-pohjaista menetelmää, jonka takia kaikki näkymät luodaan käyttäen JavaScriptiä. Vue.js:n kanssa voidaan myös käyttää JSX-pohjaista menetelmää tarvittaessa. React käyttää CSS-tyyleihin yleensä JavaScriptin ja CSS:n sekoitusta, jonka takia taas ohjelmoija voi mahdollisesti joutua opettelemaan uuden syntaksin. Vue.js taas vaihtoehtoisesti käyttää web-ohjelmoinnissa tuttuja CSS-tyylejä. Vue.js tukee myös mahdollisuutta käyttää JavaScriptin ja CSS:n sekoitusta tyylien luomiseen. (Vue.js 2020b.)

React ja Vue.js molemmat sisältävät tarvittavat työkalut luomaan laajojakin sovelluksia käyttäen molempien tarjoamia kirjastoja. React on jättänyt näiden kirjastojen ylläpidon käyttäjille, kun taas Vue.js:n kehittäjät itse ylläpitävät tarjoamiaan työkaluja, minkä takia Reactin työkalut on enemmän hajanaisia ja eivätkä ole aina välttämättä ajan tasalla. Pienempien sovellusten luontiin Vue.js tarjoaa helpon tavan päästä alkuun, kun sovelluskehys voidaan lisätä sivustoon vain käyttämällä yhtä linjaa koodia. React vaatii enemmän opettelua sen käyttämän JSX-menetelmän takia ja sen käyttämien järjestelmien takia. (Vue.js 2020b.)

React tarjoaa React Native -sovelluskehyksen avulla mahdollisuuden luoda sovelluksia Android- ja iOS-laitteille käyttäen samaa komponenttimallia kuin verkkosovelluksissa. Vue.js sisältää myös tavan luoda natiivisia sovelluksia Android- ja iOS-laitteille käyttäen Weex-sovelluskehystä, jonka avulla voidaan käyttää Vue.js:stä tuttuja komponenttimalleja mobiilisovellusten luontia varten. (Vue.js 2020b.)

3.2.2 Angular

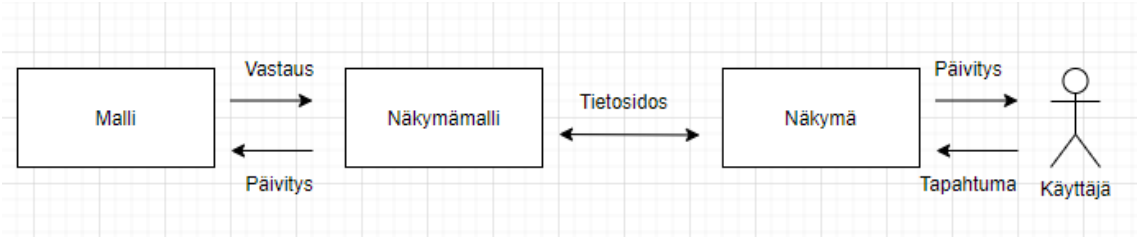
Angular on sovelluskehys, joka aiemmin tunnettiin nimellä AngularJS. Näitä ei tule sekoittaa keskenään, koska Angular on täysin uudelleen kirjoitettu versio AngularJS-sovelluskehyksestä. Angular käyttää ohjelmointikielenä TypeScriptiä, kun taas Vue.js käyttää JavaScriptiä. Vue.js myös tarjoaa mahdollisuuden käyttää TypeScriptiä ohjelmointikielenä, jos kehittäjä niin haluaa, mutta ei yhtä laajasti kuin Angular. Vue.js tarjoaa paljon joustavamman tavan luoda sovelluksia, kun taas Angular sisältää enemmän valmiiksi määritellyn tavan luoda sovelluksia. Angular vaatii käyttäjältä enemmän opiskelua, kun sen avulla aloitetaan rakentamaan sovellusta. (Vue.js 2020b.)

Angularin ohjelmointikielenä käytetään TypeScript-ohjelmointikieltä, jonka takia kehittäjä joutuu mahdollisesti opettelemaan uuden syntaksin. Angularin rajapinta on myös paljon haastavampi oppia laajuutensa takia. Angular on suunniteltu pääasiassa suurten kokonaisuuksien luontia varten, minkä takia sen käytön opettelu onkin haastavampaa. (Vue.js 2020b.)

3.3 Toiminta

Vue.js arkkitehtuuri on luotu käyttäen MVVM-arkkitehtuuria esimerkkinä. Kun Vue.js-instanssi luodaan, sen sisällä asetetaan dataobjekti, joka toimii arkkitehtuurin mallina. Vue.js-instanssi itsessään toimii arkkitehtuurin näkymämallina. Viimeisenä Vue.js:n arkkitehtuurin osana on näkymä, joka sisältää kaiken käyttäjälle näkyvän tiedon ja sitä ohjaa malli näkymämallin kautta. (Filipova 2016, 46.)

MVVM-arkkitehtuuri hyödyntää tietosidosta, joka mahdollistaa näkymän sekä näkymämallin kommunikoinnin keskenään ilman erillistä ohjainta kuten MVC-arkkitehtuurissa (kuvio 1).



Kuvio 1. MVVM-arkkitehtuurin visuaalinen esitys.

Vue.js-instanssi on tärkein osa kehitystä Vue.js:n kanssa, koska se tarjoaa tavan kommunikoida mallin ja näkymän välillä. Instanssin tärkein muuttuja on dataobjekti. Tämän avulla voidaan luoda reaktiivisia elementtejä. Aina kun data-objektissa tapahtuu muutos, Vue.js uudelleen piirtää näkymän käyttäen uusia tietoja data-objektista. (Vue.js 2020a.) Vue.js-instanssi sisältää useita tärkeitä funktioita, joita se kutsuu eri vaiheissa elinkaartaan. Näitä ovat esimerkiksi: created, mounted, updated ja destroyed. Näiden avulla voidaan hallita mitä tapahtuu elementeille tai Vue.js:n data-objektille elinkaaren aikana. (Vue.js 2020c.)

Instanssissa määritellään elementti käyttäen id arvoa, johon Vue.js kiinnittyy sekä voidaan määritellä muuttujia data-objektin sisälle (kuva 1).

```
var app = new Vue({
  el: '#app',
  data: {
    muuttuja: 'Vue.js muuttuja'
  }
})
```

Kuva 1. Esimerkki Vue.js-instanssista.

Vue.js-instanssi kiinnetään johonkin DOM-elementtiin, minkä jälkeen niiden välille muodostuu yhteys. Tämän jälkeen niistä tulee reaktiivisia, ja sitten, jos instanssin muuttujissa tapahtuu muutoksia ne, päivittyvät automaattisesti näkymään. Tämän avulla ohjelmoijan ei tarvitse huolehtia erikseen HTML-koodin päivittämisestä. (Vue.js 2020a.) Vue.js hyödyntää virtuaalista DOMia, joka nopeuttaa näkymän päivittämistä tiedon muuttuessa huomattavasti yhdistettynä reaktiivisuuden kanssa. Tämä myös mahdollistaa manuaalisesti näkymän päivittämisen. Yksi hyöty virtuaalisen DOMin käytössä on myös mahdollisuus luoda näkymä palvelimen puolella käyttäjän selaimen sijasta, mikä nopeuttaa verkkosovelluksen toimitusta käyttäjälle. (Vue.js 2016.) Vue.js käyttää viiksimallia, jonka avulla voidaan Vue.js-instanssista liittää näkymän elementteihin tietoa. Viiksimallilla tarkoitetaan tupla-aaltosulkeita, joiden sisälle haluttu koodi asetetaan. Viiksimalli myös mahdollistaa lyhyen JavaScript-koodin käyttämistä sen sisällä, minkä avulla voidaan muokata elementin arvoa. (Vue.js 2020d.)

Vue.js-näkymässä on määritelty div-elementti arvolla app ja sen sisälle on määritelty h1-elementti, joka saa arvon Vue.js-instanssista viiksimallin avulla (kuva 2).

```
<div id="app">
  <h1> {{ muuttuja }} </h1>
</div>
```

Kuva 2. Esimerkki viiksimallin käytöstä.

Yksi vahvoista Vue.js:n ominaisuuksista ovat komponentit. Niiden avulla voidaan rakentaa suuri kokonaisuus käyttäen useita pieniä paloja. Komponentteja on myös mahdollista käyttää uudelleen useampaan kertaan. Jokainen komponentti sisältää samanlaisen data-objektin kuin itse Vue.js-instanssi ja niille voidaan luoda omia funktioita ja objekteja samalla tavalla kuin Vue.js-instanssille.

Komponentteja voidaan sitten luomisen jälkeen käyttää näkymissä. (Filipova 2016, 53–54.) Komponenteille voidaan antaa tietoja käytön yhteydessä näkymässä. Komponentille luonnin yhteydessä annetaan props-muuttuja, jonka kautta komponentille voidaan syöttää tietoa näkymässä. Tämä mahdollistaa komponentin uudelleen käyttämisen eri tiedoilla. (Vue.js 2020e.)

Direktiivit tarjoavat helpon tavan toimia näkymän sisällä elementtien seassa. Vue.js tarjoaa useita eri direktiivejä jo valmiiksi. Direktiivit on merkitty v-* alkumerkillä. Direktiivien avulla voidaan hallita elementtejä. Esimerkkinä on v-if-direktiivi, jonka avulla voidaan elementille asettaa ehto, jonka täyttyessä elementti näkyy näkymässä käyttäjälle tai jos ehto ei toteudu elementti ei näy näkymässä ollenkaan. Yksi usein käytetty direktiivi on v-model, jonka avulla elementin ja data-objektin välille voidaan luoda kaksisuuntainen linkitys, joka muokkaa muuttujan arvoa, jos se muuttuu näkymässä tai instanssissa. (Vue.js 2020a.)

4 Laravel-sovelluskehys

4.1 Perusta

Laravel on back-end-sovelluskehys, joka on luotu käyttäen PHP-ohjelmointikieltä ja se on kehitetty helpottamaan ja nopeuttamaan verkkosovellusten kehitystä. Laravel sisältää suuren kirjon työkaluja ohjelmistokehityksen avuksi. Laravel keskittyy ytimessään kehittäjiin ja heidän tarpeisiinsa. Laravelin avulla on helppo luoda uusi projekti, vaikka olisi vasta aloittelija, ja sitä oppii käyttämään nopeasti. (Stauffer 2019, 5–6.) Laravelin avulla voidaan helposti yhdistää myös front-end-sovelluskehys kuten React tai Vue.js projektiin ja se sisältää näille sovelluskehyksille pohjan jo valmiiksi. Laravel myös tukee Bootstrap-kirjastoa. (Laravel 2020a.)

Laravel sai alkunsa kuten moni muukin sovelluskehys, kun kehittäjien usein käytettyjen toimintojen toteuttaminen projekteissa oli liian hankalaa tai aikaa vievää käyttäen pelkästään PHP 5.3-ohjelmointikielen perusteita. Taylor Otwell näki nämä hankaluudet ja päätti tehdä jotain niiden ratkaisemiksesi ja näin syntyi Laravel. Laravelin ensimmäinen versio julkaistiin kesäkuussa vuonna 2011.

Laravel tuki jo ensimmäisestä versiosta lähtien sisäänrakennettua järjestelmää käyttäjien rekisteröimiseen ja sisäänkirjautumiseen. Laravelin ensimmäinen versio ei vielä kuitenkaan seurannut täysin MVC-arkkitehtuuria. (Surguy 2013.)

Laravelin toinen versio julkaistiin marraskuussa 2011. Tässä versiossa esiteltiin tuki ohjaimille, minkä vuoksi Laravel viimein täytti kriteerit MVC-arkkitehtuurin sovelluskehyksestä. Laravel version kolme ansiosta se alkoi jo saada kiinni muita markkinoilla olevia PHP-sovelluskehyksiä ja sen suosio kasvoi. (Surguy 2013.) Tässä opinnäytetyössä käsitellään Laravel-versioita 5.8-8.0.

4.2 Kilpailevat sovelluskehykset

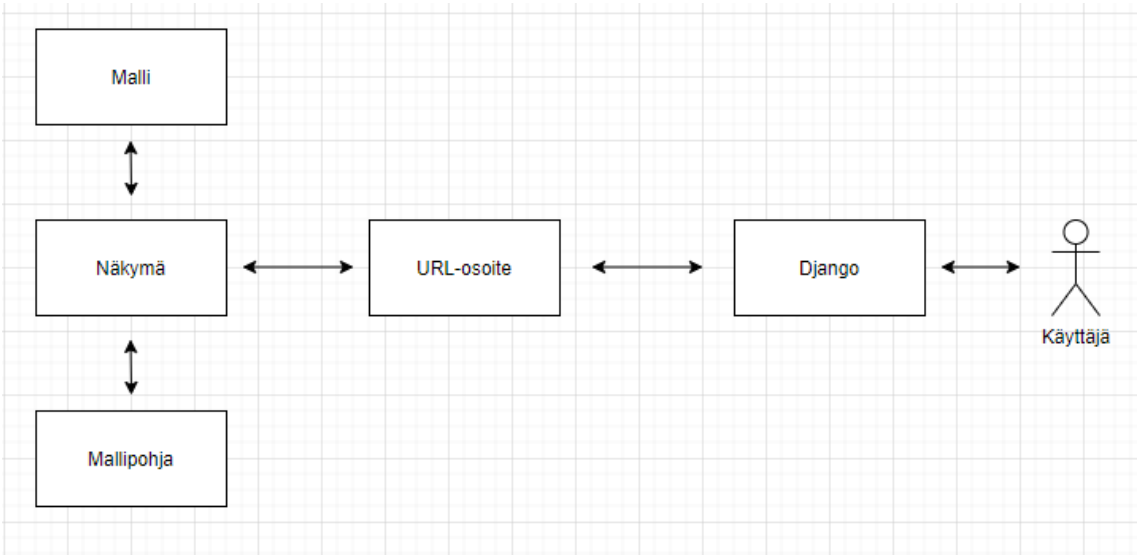
Laravel kilpailee useiden eri back-end-sovelluskehysten kanssa, mutta valitsin muutaman suosituimman kilpailijan vertailuun. Valitsin vertailuun sovelluskehykset Ruby on Rails ja Django.

Ruby on Rails on sovelluskehys, joka tunnetaan myös pelkästään Rails-nimellä. Rails käyttää Ruby-ohjelmointikieltä ja seuraa MVC-arkkitehtuuria. Rails soveltuu hyvin pienten sovellusten luomiseen, kun taas Laravelin avulla voidaan rakentaa laajojakin sovelluksia ilman ongelmia. Rails tarjoaa suuremman yhteisön sen ympärillä sekä hyvän dokumentaation, kun taas Laravelin yhteisö on ailahtelevampi, mutta tarjoaa myös hyvän dokumentaation. Rails on suorituskyvyltään hitaampi ja tämä rajoittaa sen käyttöä suuremmissa sovelluksissa. Rails keskittyy enemmän yhteisön luomiin työkaluihin, kun taas Laravel sisältää useita eri työkaluja jo valmiiksi. (Educba 2020.)

Django on sovelluskehys, joka on luotu käyttäen Python-ohjelmointikieltä. Django seuraa MVT-arkkitehtuuria, kun Laravel taas käyttää MVC-arkkitehtuuria. (Nader 2020.) MVT-arkkitehtuuri on lyhenny sanoista Model-View-Template, suomennettuna malli-näkymä-mallipohja. MVT- ja MVC-arkkitehtuureilla on samoja ominaisuuksia, kuten molemmissa malli on samanlainen eli se esittää jotain tietokantataulua. MVT-arkkitehtuurissa näkymä vastaa siitä mitä tietoa näytetään. Mallipohjien avulla muodostetaan rakenne, jonka avulla voidaan esittää tietoa mallin perusteella. MVT-toimii suoraviivaisemmin, mikä tekee siitä hyvän vaihtoehdon myös pienille sovelluksille. MVT-

arkkitehtuuri ei tarvitse erillistä ohjainta tai itse kirjoitettua menetelmää tiedonhakua varten, koska arkkitehtuuri hoitaa sen itse. (GeekInsta 2019.)

Djangon MVT-arkkitehtuurin toteutuksessa Django hoitaa itse tiedonvaihdon käyttäjän ja halutun URL-osoitteen välillä. Sitten tämän avulla näkymä rakennettaan ja toimitetaan käyttäjälle. (Kuvio 2.)



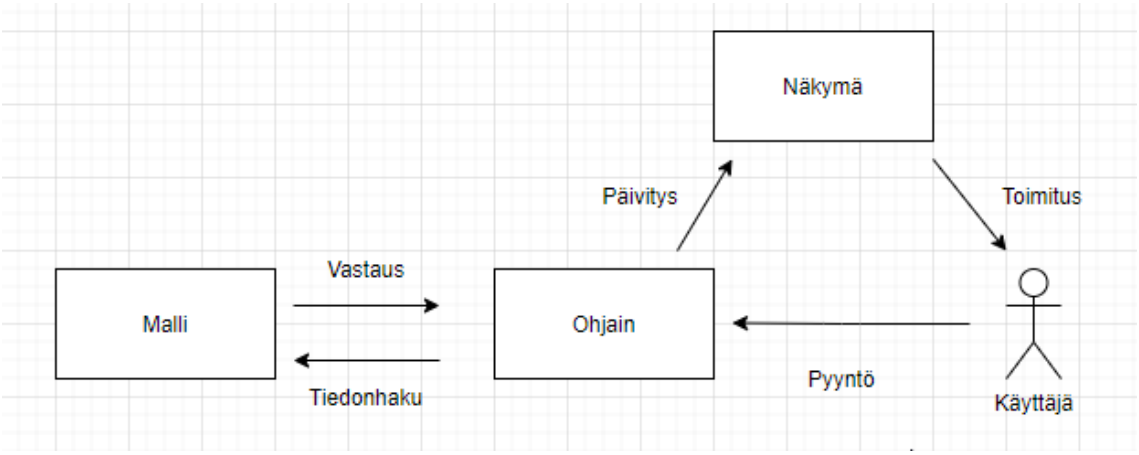
Kuvio 2. Django MVT-arkkitehtuurin visuaalinen esitys.

Uuden kehittäjän on helpompi päästä alkuun käyttäen Django-sovelluskehystä, kun taas Laravel vaatii enemmän opettelua. Django voittaa Laravelin nopeudessa sen käyttämän Python-ohjelmointikielen ansiosta. Django on panostanut paljon tietoturvaan, kun taas Laravelissa on otettu huomioon tietoturva ja se sisältää perustan turvalliselle sovellukselle, mutta ei pärjää Djangolle tässä osiossa. Laravel tarjoaa helposti saatavilla olevan rajapinnan ja se on hyvin dokumentoitu, kun taas Django vaatii erillistä kirjastoa rajapinnan käyttöjä varten. Django ei myöskään tarjoa front-end-sovelluskehyksille tukea vaan kehittäjä joutuu itse tämän luomaan, kun taas Laravel tukee front-end-sovelluskehyksiä kuten React ja Vue.js suoraan asennuksessa. (Nader 2020.)

4.3 Toiminta

Laravel on rakennettu noudattaen MVC-arkkitehtuuria. MVC-arkkitehtuuri sisältää kolme konseptia, jotka ovat malli, näkymä ja ohjain. Malli esittää jotakin yksittäistä tietokantataulua. Näkymä taas esittää käyttäjälle näkyvää verkkosivuston mallia. Ohjain hallitsee verkkoliikennettä käyttäjän ja palvelimen välillä molempiin suuntiin, kuten tietokannan lukemista tai kirjoittamista ja käyttäjälle vastausten lähettämistä. (Stauffer 2019, 23–24.)

MVC-arkkitehtuurissa ohjain hoitaa kommunikoinnin käyttäjän kanssa ja vastaa päivitetyn näkymän toimituksesta käyttäjälle (kuvio 3).



Kuvio 3. MVC-arkkitehtuurin visuaalinen esitys.

Reititys on tärkeää missä tahansa verkkosovelluksessa, koska tämän avulla voidaan hallita tapahtuvaa liikennettä käyttäjän ja sovelluksen välillä. Reitityksen avulla voidaan ohjata HTTP-pyyntöjä ja tämän avulla palauttaa käyttäjälle tietoa. Laravel tarjoaa laajan kirjon työkaluja reitityksien luomisen helpottamiseksi. Kun reitti luodaan, niin voidaan hallita mitä käyttäjälle tapahtuu, kun hän lähettää HTTP-pyynnön selaimestaan sovelluksen reitille. Reitin luonnin yhteydessä on myös mahdollista määrittää reitin hyväksymät HTTP-pyynnöt kuten GET, POST tai muut HTTP-pyynnöt. (Stauffer 2019, 26–28.)

Laravel tarjoaa Blade-nimisen mallinnusmoottorin. Moottorin avulla voidaan helposti yhdistää Laravel sekä PHP- ja HTML-ohjelmakoodi, jonka avulla saadaan nopeasti käyttäjälle näkyviä sivustoja. Tupla-aaltosulkeiden avulla voidaan määrittää sijainti PHP-dokumenttiin, johon halutaan sisällyttää Laravelin avulla tietoa. Blade-moottori tarjoaa myös direktiivejä, joiden avulla voidaan käyttää PHP-ohjelmakoodia HTML-koodien seassa. Direktiivit merkitään etumerkillä @. Tämä nopeuttaa ohjelmointia verrattuna siihen, jos käyttäisi suoraan PHP:n

vastaavia menetelmiä. Näkymiä voidaan myös laajentaa käyttämällä direktiivejä, joiden avulla voidaan luoda pohja näkymälle. Näkymän pohjaa voidaan sitten käyttää uudestaan, kun luodaan uusia näkymiä ja laajentaa sitten niitä käyttäen yhtä pohjaa. Laravel tarjoaa myös mahdollisuuden luoda omia direktiivejä, joiden avulla voidaan lyhentää usein toistettavia koodipätkiä käyttäen yhtä direktiiviä. (Stauffer 2019, 63–70.)

Laravel tarjoaa helpon tavan kommunikoida tietokantojen kanssa sekä tarjoaa siihen useita eri työkaluja. Laravelin avulla voidaan luoda erilliset tietokantayhteydet tietokannan lukemista, kirjoittamista tai muita SQL-toimintoja varten. Laravel myös mahdollistaa useamman eri tietokannan käyttämistä samaan aikaan. (Laravel 2020b.) Laravel sisältää myös oman toiminnon suorittaa tietokantakyselyitä, jossa on sisäänrakennettu turvallisuusjärjestelmä, joka estää kyselyitä muokkaamasta tietokantaa ilman lupaa (Laravel 2020c). Tietokantojen luontia varten ryhmäprojektissa Laravel myös tarjoaa helpon tavan käyttäen migraatioita, joiden suoritettaessa Laravel luo automaattisesti tietokannan tiedot ja rivit. Tämä helpottaa projektin käyttöönottoa uusilla kehittäjillä (Laravel 2020d). Laravel myös sisältää kyvyn luoda tietokantaan satunnaista tietoa kehittäjän toimesta, mikä mahdollistaa tietokannan testauksen erilaisilla tiedoilla (Laravel 2020e).

5 Muut teknologiat

5.1 Docker

Docker on alusta, jonka avulla voidaan rakentaa eristettyjä sovelluksia. Dockerin avulla luodaan kontteja, joiden sisällä sovellus suoritetaan. Docker mahdollistaa suurtenkin sovellusten suorittamisen konttien sisällä. Kontin luomisen jälkeen se on kevyt suorittaa sekä helposti jaettavissa muille tai suorittaa myös vaikka pilvipalvelussa. Docker paketoi halutun sovelluksen ja sen kaikki tarvitsemat osat kontin sisälle ja luo siitä suoritettavan. Kontit eivät kuitenkaan suorita kokonaista käyttöjärjestelmää sisällään, mikä tekee niistä kevyen ja nopean vaihtoehdon virtuaalikoneille. Kontit ovat itsenäisiä ja tämän vuoksi useita eri

kontteja voidaan ajaa samanaikaisesti rinnakkain. Jokainen kontti sisältää oman tiedostojärjestelmän. (Docker Docs 2020.)

Docker tarjoaa kattavan kehitysympäristön Docker Desktop -nimisen sovelluksen avulla. Tämän sovelluksen avulla kontteja voidaan luoda ja suorittaa helposti. Docker tukee myös Windows-, Linux- ja macOS-käyttöjärjestelmiä. Docker Desktop -sovelluksen avulla kontteja voidaan käynnistää ja pysäyttää helposti käyttöliittymän kautta. (Docker Docs 2020.)

5.2 Visual Studio Code

Visual Studio Code on sovellus tekstin muokkaamista varten. Sovellus on kuitenkin pääasiassa suunnattu ohjelmakoodin muokkaamista varten. Se on hyvin kevyt sovellus ja tukee useita eri käyttöjärjestelmiä kuten Windowsia, macOSia ja Linuxia. Visual Studio Code sisältää tuen useille eri ohjelmointikielille suoraan kuten JavaScriptille, TypeScriptille ja Node.js:lle. Näiden lisäksi sovellukseen voidaan asentaa lisäosia muille ohjelmointikielille. (Visual Studio Code 2020a.)

Sovellus tarjoaa ominaisuuksia kuten IntelliSense, jonka avulla se tarjoaa käyttäjälle avustusta ohjelmakoodin kirjoittamiseen, kuten automaattisia ehdotuksia muuttujista tai funktioista (Visual Studio Code 2020a). Visual Studio Code myös sisältää sisäänrakennetun versionhallinnan käyttäen eri versionhallinnan työkaluja, mutta sisäänrakennettuna on Git-tuki. Muita versionhallinnan työkaluja voidaan asentaa lisäosina (Visual Studio Code 2020b).

5.3 HeidiSQL

HeidiSQL on avoimen lähdekoodin sovellus, jonka avulla voidaan tulkita tietokantoja. HeidiSQL tukee useita eri tietokantamuotoja kuten MariaDB, MySQL ja Microsoft SQL. Sovelluksen avulla voidaan yhdistää useampaan tietokantaan helposti. Sovellusta voidaan käyttää joko komentoriviltä tai sitten käyttöliittymän kautta. Sovelluksen avulla voidaan lukea tietokantoja sekä myös muokata niitä. Sovelluksen avulla voidaan myös tallentaa tietokanta sellaisella muodossa, että

se voidaan helposti luoda sitten uudestaan eri tietokoneella ja se pysyy täysin samana. (HeidiSQL 2020.)

HeidiSQL tarjoaa monia ominaisuuksia tietokannan muokkausta varten. Sen avulla monia tauluja voidaan muokata samanaikaisesti. Sen avulla tauluja myös voidaan optimoida automaattisesti sekä tehdä niistä luettavampia. Tietokannan kyselyitä varten HeidiSQL tarjoaa korostusta SQL-tekstiä varten, kuten tunnistaa automaattisesti, jos kyselyssä viitataan tietokannan tauluun. (HeidiSQL 2020.)

6 Sovelluksen kehitys

6.1 Sovelluskokonaisuuden toiminta

Kyseessä on sovelluskokonaisuus, joka on luotu valmentajille helpottamaan urheilijoiden valmentamista joukkue- sekä yksilölajien parissa. Sovelluksen avulla voidaan suunnitella harjoitteita, seurata ja kuunnella urheilijoita ja saada heiltä palautetta sekä kommunikoida urheilijoiden ja valmentajien kanssa. Sovelluksen avulla voidaan luoda monipuolista tietoa urheilijoiden tilasta sekä heidän antamastaan palautteesta. Sovellus sisältää toiminnon, jonka avulla joukkueelle voidaan lähettää kyselyitä ja heidän vastauksistaan voidaan luoda diagrammeja. Joukkueen urheilijat myös näkevät sovelluksesta heidän harjoitteensa. Sovellus on toteutettu verkkopohjaisesti eli se toimii selaimessa ja mobiililaitteissa.

Kyseessä oli GritPro-sovelluksen jatkokehitys eli lähdettiin jatkokehittämään olemassa olevaa sovellusta. GritPro on vain yksi osa tätä sovelluskokonaisuutta ja tämä sovelluksen osa vastaa vain urheilijoille tarkoitetuista kyselyistä. Jatkokehityksessä jouduttiin myös toimimaan sovelluksen muiden osien kanssa yhteensopivuuden takia, mutta tämä jäi kuitenkin vähäiseksi. Valitut teknologia olivat ennalta määrättyjä projektia varten, koska kyseessä oli jatkokehitys ja sovellus on alun perin rakennettu käyttäen teknologioita, joita käytetään jatkokehityksessä. Tämän vuoksi myös jatkokehityksessä huomioitiin muu sovellus eli ohjelmakoodi pidettiin samanlaisena kuin se on muissa sovelluksen osissa sekä

kaikki käyttäjälle näkyvät osiot tehtiin vastaamaan muun sovelluksen ulkoasua. Kyseessä oli myös käyttäjille käyttöön tuleva sovellus eli pidettiin mielessä käytettävyys ja ymmärrettävyys.

6.2 Vaatimusmäärittely

Kyselysovellukseen haluttiin kolme uutta toimintoa, joiden oli tarkoitus antaa käyttäjälle vapaammat kädet hallita kyselyitä. Järjestelmässä oli valmiita kyselyitä, jotka oli luotu manuaalisesti järjestelmään yrityksen työntekijöiden toimesta käyttäen urheilupsykologien luomia kyselyitä. Kyselyitä varten haluttiin toiminto, jonka avulla käyttäjät voivat itse luoda uusia kyselyitä suoraan sovelluksesta. Kyselyiden kysymyksiin vastataan käyttäen erilaisia komponentteja, jotka mahdollistavat tarkempien vastausten saamisen riippuen kysymyksestä. Esimerkkejä komponenteista on vapaamuotoinen tekstivastaus ja sitten arvio numeerisella asteikolla. Seuraavana haluttuna toimintona oli uudet komponentit vastausten laajentamista varten. Kyselyitä voidaan lähettää joukkueelle valmentajan toimesta ja kysely valitaan pudotusvalikosta. Tähän haluttiin lisätä näkyviin valitun kyselyn sisältämät tiedot, kuten kysymykset ja niiden aputekstit.

Kyselyiden luomista varten haluttiin sille oma alasivu. Uuden alasivun ulkoasun tuli vastata sovelluksen yleistä ulkoasua. Kyselyä varten piti pystyä antamaan kyselyn nimi sekä kuvaus. Kyselyyn piti olla mahdollista lisätä niin monta kysymystä kuin käyttäjä haluaisi. Jokaista kysymystä varten tuli olla mahdollista antaa kysymyksen kuvaus sekä aputeksti. Tämän lisäksi jokaiseen kysymykseen piti olla mahdollista valita komponentti, jonka avulla siihen vastataan, ja komponentille tuli olla mahdollista syöttää tarvittavat lisätiedot kuten asteikon minimi- ja maksimiarvot. Lopuksi käyttäjän piti pystyä painamaan nappia, joka sitten luo ja tallentaa kyselyn tietokantaan ja tämän jälkeen sen täytyi olla heti lähetettävissä joukkueelle. Uusien kyselyiden vastausten perusteella piti olla mahdollista luoda raportti, joka visualisoi joukkueen vastauksia. Kyselyn luonnissa tietokantaan tuli ottaa huomioon käyttäjän valittu kieli sovelluksessa, että kysely luodaan vain tällä kielellä. Sovellukseen tuli jäädä myös mahdollisuus lisätä tulevaisuudessa kyselyyn käännökset muille kielille. Piti myös ottaa huomioon, että kun käyttäjä luo kyselyn, tämän kyselyn tuli vain näkyä vain hänelle ja hänen joukkueellensa.

Vastausten laajentamista varten haluttiin lisätä kolme uutta komponenttia, joiden avulla kyselyn kysymyksiin voidaan vastata. Ensimmäisenä uutena komponenttina oli kyllä- ja ei-vaihtoehdot, jonka avulla käyttäjä voi valita vastauksena jommankumman. Toisena uutena komponenttina oli pudotusvalikko, jonka avulla käyttäjä voi valita annetuista vaihtoehdoista yhden. Kolmantena uutena komponenttina oli muunneltu versio toisesta uudesta komponentista, jossa yhden valinnan sijaan käyttäjä voi valita valikosta monta eri vaihtoehtoa. Kyselyiden lisäämiseen liittyen uusien komponenttien vuoksi täytyi muokata olemassa olevaa raportointia kyselyiden vastauksista sitten, että se huomioi uudet komponentit raportin luonnissa.

Kyselyiden tiedot tuli saada näkyviin modal-ikkunaan, josta valitaan lähetettävä kysely. Valitusta kyselystä tuli hakea ja näyttää kaikki sen sisältämät kysymykset sekä niiden aputekstit, jotta käyttäjä pystyy tarkistamaan kyselyn sisällön ennen sen lähettämistä joukkueelle.

6.3 Perustan luonti

Kehitys tehtiin täysin paikallisessa ympäristössä. Docker Desktop -sovelluksen avulla saatiin sovelluskokonaisuus toimimaan paikallisesti käyttäen sovelluksen konttia. Dockerin avulla käytettiin paikallisena WWW-palvelimena nginx-ohjelmistoa. Dockerin avulla myös pystytettiin paikallinen MySQL-tietokanta, jota hyödynnettiin kehityksen ajan. Dockerin avulla saatiin myös pääsy Linuxin komentoriviin, jonka avulla voitiin suorittaa tarvittavia Laravel-komentoja. Komentorivillä piti suorittaa komento php artisan migrate:fresh –seed, jonka avulla Laravel loi tietokantaan kaikki tarvittavat taulut ja täytti ne testitiedoilla, jonka avulla kehitys saatiin alkuun. Kyselyiden luomista varten täytyi lisätä uusi ohjain, jota voidaan sitten hyödyntää HTTP-pyyntöjen saapuessa. Ohjaimet luodaan käyttäen komentoriviä komennolla php artisan make:controller.

Ohjaimelle piti määrittää reitit, joita varten se kuuntelee HTTP-pyyntöjä ja sitten suorittaa halutun tapahtuman, kun pyyntö saapuu tietylle reitille. Sovelluksessa reitit on määritelty erillisiin tiedostoihin sen mukaan mihin sovelluksen osioon

reitti kuuluu. Kyselyiden muokkaamista varten täytyi lisätä POST- ja GET-reitit, jotka ovat yhteydessä aiemmin luomaamme ohjaimeen.

Route-luokan avulla määritellään HTTP-pyynnön tyyppi ja sen hyväksymä reitti ja sitten ohjaimesta valitaan tietty funktio, joka suoritetaan pyynnön saapuessa kyseiseen reittiin. Määriteltyn index-funktion on tarkoitus palauttaa käyttäjälle näkymä kyselyiden luomista varten. Funktiota create käytetään myöhemmin itse kyselyn luomiseen tietokantaan. (Kuva 3.)

```
Route::get('/elite/gritpro/questionnaire/editor/', [GritProQuestionnaireEditorController::class, 'index']);
Route::post('/elite/gritpro/questionnaire/editor/create', [GritProQuestionnaireEditorController::class, 'create']);
```

Kuva 3. Reittien luonti HTTP-pyyntöjä varten.

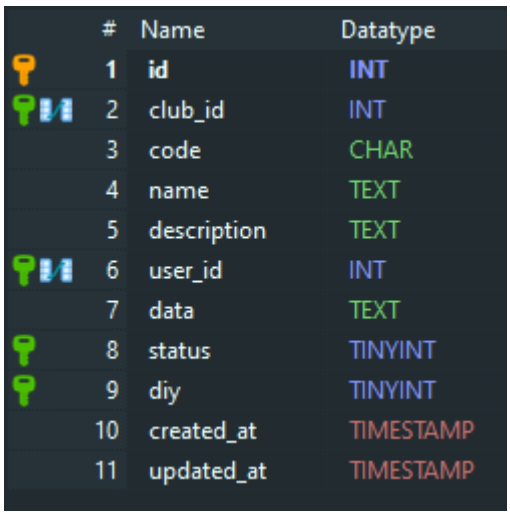
Täytyi luoda uusi näkymä käyttäen Laravel Blade-mallinnusmoottoria ja tähän käytettiin valmista pohjaa toisesta näkymästä, että saadaan näkymä näyttämään samalta kuin muut näkymät. Tämä näkymä palautetaan käyttäjälle HTTP-pyynnön saapuessa index-funktion avulla. Piti myös luoda uusi Vue.js-komponentti, jota käytettiin itse kyselyiden luontia varten. Vue.js-komponenttia voi käyttää suoraan näkymässä, kun se on rekisteröity Vue.js:n kautta sovellukseen. Vue.js-komponenttia varten käytettiin toista valmista komponenttia pohjana, että niiden ohjelmakoodin ulkoasu pysyy samanlaisena. Uuden ohjaimen sisälle luotiin funktio nimeltä index. Tämä funktio käyttää Laravelin view-funktiota palauttaakseen näkymän käyttäjälle, jos hän vierailee määritellyssä reitissä.

6.4 Kyselyiden rakenne

Kyselyiden luontia varten piti tutkia, kuinka kyselyt ovat tallennettu järjestelmään ja missä muodossa. Tätä varten tutkittiin sovelluksen MySQL-tietokantaa HeiqiSQL-sovelluksen avulla. Tietokannassa GritPro-taulut on merkitty elite_gp-alkumerkinnällä. Kyselyt on sijoitettu tauluun elite_gp_templates ja tähän tauluun keskityttiin kehityksessä.

Taulun rakenteesta nähdään, kuinka kyselyt sijaitsevat tietokannassa. Tärkeimmät kentät kehitystä varten ovat code, name, description ja data. Kentällä code määritetään, mitä raportointimenetelmää käytetään, kun kyselyn vastauksista

luodaan raportti. Kenttä name pitää sisällään kyselyn nimen ja kenttä description pitää sisällään kyselyn kuvauksen tai aputekstin. Viimeisenä tärkeänä kenttänä on data, joka sisältää kyselyn sisältämät kyselyt sekä niihin liittyvät tiedot kuten millä komponentilla tiettyyn kysymykseen vastataan. Muita kenttiä kuten user_id ja club_id käytetään kyselyn lähetyksen yhteydessä valitakseen mille joukkueelle kysely lähetetään. (Kuva 4.)

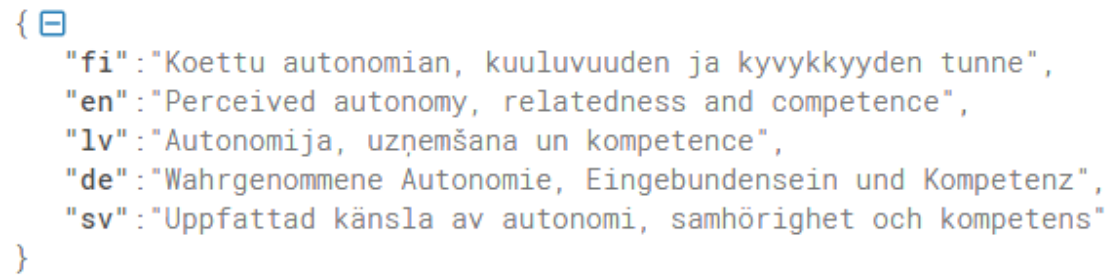


#	Name	Datatype
1	id	INT
2	club_id	INT
3	code	CHAR
4	name	TEXT
5	description	TEXT
6	user_id	INT
7	data	TEXT
8	status	TINYINT
9	diy	TINYINT
10	created_at	TIMESTAMP
11	updated_at	TIMESTAMP

Kuva 4. Kyselyiden MySQL-taulun rakenne tietokannassa.

Koska sovellus tukee useita eri kieliä, tarvitsevat myös kyselyt mahdollisesti useampia kieliä, joten name-, description- ja data-kentät on toteutettu JSON-muodossa. Kentät name ja description ovat objekteja, jotka mahdollistavat kyselyn nimen ja kuvauksen kääntämisen usealle eri kielelle.

Jokainen kieli merkitään lyhenteellä, jota seuraa sitten käännös haluttua kieltä varten. Tämä mahdollistaa kääntämisen niin monelle kielelle kuin joukkue tarvitsee. (Kuva 5.)



```
{
   "fi":"Koettu autonomian, kuuluvuuden ja kyvykkyyden tunne",
   "en":"Perceived autonomy, relatedness and competence",
   "lv":"Autonomija, uzņemšana un kompetence",
   "de":"Wahrgenommene Autonomie, Eingebundensein und Kompetenz",
   "sv":"Uppfattad känsla av autonomi, samhörighet och kompetens"
}
```

Kuva 5. MySQL-taulun name-kenttä JSON-muodossa.

Taulun data-kenttä on myös JSON-muodossa, mutta se on monimutkaisempi, koska se tarvitsee myös taulukkoja objektien lisäksi. Kysymykset on eroteltu ryhmiin, joita varten on oma taulukko. Jokaisella ryhmällä on name ja description, jotka ovat tavallisia merkkijonoja. Jokaisella ryhmällä on myös items-taulukko, joka sisältää ryhmän kysymykset. Jokaisella kysymyksellä on oma description ja help, jotka ovat samanlaisia objekteja kuin kyselyn name, joka mahdollistaa niiden kääntämisen usealla eri kielelle. Kysymyksellä on myös component-objekti, jolla voidaan määrittää millä, komponentilla käyttäjä vastaa kysymykseen ja antaa komponentille asetukset tämän kautta. (Kuva 6.)


```
{
   "groups":[
      {
         "name":"Esimerkki ryhmä",
         "type":"group",
         "items":[
            {
               "description":{
                  "fi":"Esimerkki kysymys"
               },
               "help":{
                  "fi":"Kysymyksen aputeksti"
               },
               "component":{
                  "type":"opentext",
                  "settings":[

                  ]
               }
            },
            {
               "description":{
                  "fi":"Toinen esimerkki kysymys"
               },
               "help":{
                  "fi":"Aputeksti toista kysymystä varten"
               },
               "component":{
                  "type":"scale",
                  "settings":{
                     "min":1,
                     "max":"10",
                     "invert":0
                  }
               }
            }
         ]
      }
   ]
}
```


Kuva 6. MySQL-taulun data kenttä JSON-muodossa.

Tämän tutkimuksen avulla saatiin kaikki tarvittava tieto, jota käyttäjältä täytyy kerätä luomassamme Vue.js-komponentissa, että kysely voidaan luoda onnistuneesti tietokantaan.

6.5 Kyselyiden luonti järjestelmään

Kyselyiden luontia varten täytyi luoda käyttäjäystävällinen näkymä, jota käyttäjät osaavat käyttää. Kyselyihin täytyi olla mahdollisuus lisätä kysymyksiä rajattomasti. Tätä varten luotiin jo aiemmin Vue.js-komponentti. Käyttäjäystävällisyyttä varten kysymyksiä täytyi myös olla mahdollisuus poistaa ja muokata, jos käyttäjä tekee virheen ennen kyselyn luontia.

Vue.js-komponenttiin täytyi sijoittaa sen data-objektiin tarvittavat muuttujat, joita hyödynnetään näkymässä. Sovelluksen standardien mukaan käytetään dataa funktiona objektin sijaan. Ensimmäisiä muuttujina luotiin merkkijonot title ja description, joihin käyttäjä voi syöttää halutun kyselyn nimen ja sen kuvauksen. Itse komponentin näkymään eli sen template-osioon, joka tulee käyttäjälle näkymään, täytyi sijoittaa tekstikentät, joihin käyttäjä voi syöttää tiedot. Yleistä ulkoasua varten näkymään haettiin toiselta sivulta ulkoasu, joka jakaa sivun kahteen osioon, joissa oikealle sijoitetaan näkymään kyselyyn lisätyt kysymykset ja vasemmalle kyselyn nimi ja sen kuvaus sekä napit kysymysten lisäämistä ja kyselyn luontia varten.

Tekstikenttiä varten hyödynnettiin Vue.js-direktiiviä v-model, jonka avulla saatiin luotua kaksisuuntainen linkitys tekstikenttään syötetyn tiedon ja sille määritellyn muuttujan välille (kuva 7).

```
<div class="md-form">
  <label class="active" for="title">Nimi</label>
  <input id="title" v-model="title" type="text" class="form-control">
</div>
```

Kuva 7. Vue.js v-model direktiivin käyttö kaksisuuntaisen linkityksen luontia varten.

Kysymysten lisäämistä varten kyselyyn valittiin parhaaksi vaihtoehdoksi modal-ikkuna, joka avautuu käyttäjän painaessa nappia kysymysten lisäämistä varten. Komponenttiin voidaan määritellä funktioita sen methods-objektin sisälle. Modalin näyttämistä ja piilottamista varten täytyi luoda uudet funktiot. Modalin sisälle sijoitettiin tekstikentät uuden kysymyksen nimelle ja kuvaukselle tai aputekstille. Lisättiin myös pudotusvalikko, josta voidaan valita kysymykselle komponentti, jolla siihen vastataan. Komponentille tarvittiin myös pääsy käyttäjän syötteisiin kysymyksen osalta. Kysymykselle lisättiin omat muuttujat data-funktioon, joihin

sitten käytettiin v-model-direktiiviä luodakseen linkki. Tämän avulla komponentin muuttujat saavat käyttäjän antamat arvot tekstikentissä sekä pudotusvalikossa. Tässä vaiheessa luotiin myös uusi taulukko nimeltä questions kysymyksiä varten. Se mahdollistaa kysymysten tallentamisen komponentin sisällä. Kysymyksiä varten valitut komponentit vastaamista varten tarvitsevat myös mahdollisesti asetuksia valitun komponentin perusteella. Tätä varten luotiin myös uusi objekti, joka sisälsi jokaiselle komponentille omat asetukset. Käyttäjälle piti luoda myös mahdollisuus muokata valitun komponentin asetuksia vastausta varten, joten tähän hyödynnettiin v-if-direktiiviä, jonka avulla näkymään saatiin luotua jokaiselle komponentille omat asetukset.

Voidaan luoda vaihtoehtoinen elementti, joka näkyy vain, jos direktiivin sisällä annettu ehto täyttyy. Elementin sisällä olevat elementit seuraavat samaa näkyvyyttä. Tässä tapauksessa käytetään kysymyksen tyyppiä data-funktiosta. (Kuva 8.)

```
<div v-if="question_type === 'scale'">
</div>
<div v-else-if="question_type === 'selectlist'">
</div>
```

Kuva 8. Vue.js v-if-direktiivin käyttö elementtien vaihtoehtoista näkyvyyttä varten.

Numeroskaala-komponenttia varten tarvittiin tekstikentät minimi- ja maksimiarvoille sekä valintaruutu, jos kysymyksen vastausten raportointi halutaan käänteiseksi. Näihin kaikkiin hyödynnettiin v-model-direktiiviä sekä aiemmin luomaamme objektia komponenttien asetuksia varten data-funktiosta. Viimeiseksi modal-ikkunan lisätään nappi, jota painamalla uusi kysymys lisätään aiemmin luomaamme taulukkoon kysymyksiä varten. Nappia varten täytyy luoda uusi funktio nimeltä saveQuestion methods-objektiin, joka avulla kysymys voidaan tallentaa.

Käytettävyyden takia täytyi myös lisätä mahdollisuus käyttäjälle muokata kysymyksiä jälkeenpäin ennen kyselyn luomista. Esimerkiksi jos käyttäjä tekee virheen, niin hän voi korjata sen. Tätä varten lisättiin uusi muuttuja, jota voidaan hyödyntää saveQuestion-funktiossa myöhemmin. Tässä vaiheessa täytyi myös saada listaus kysymyksiä oikeaan lohkoon sivustoa, että käyttäjä näkee, mitkä

kysymykset hän on jo luonut ja tarvittaessa voi muokata niitä taikka poistaa. Muokkausta ja poistamista varten täytyy myös luoda omat funktiot methods-objektiin.

Vue.js:n avulla voidaan taulukko käydä läpi näkymässä, joka mahdollistaa kysymysten listauksen näkymään sekä jokaiselle kysymykselle voidaan luoda muokkaus- ja poista-elementit, jotka hyödyntävät Vue.js click-tapahtumaa (kuva 9).

```
<div class="col-md-6 mt-2">
  <h5 class="h5-responsive card-title text-left mb-3 py-1">
    <strong>Kysymykset</strong>
  </h5>
  <div v-for="(question, index) in questions" :key="question.id">
    <strong> Kysymys {{ index + 1 }} </strong>
    <br><small> {{ question.description }} </small>
    <div class="text-right mt-2">
      <a @click="editQuestion(question.id)"> Muokkaa </a>
      <a @click="deleteQuestion(question.id)"> Poista </a>
    </div>
  </div>
</div>
```

Kuva 9. Vue.js v-for-direktiivin hyödyntäminen taulukon näyttämiseen näkymässä.

Kyselyn luontia varten aiemmin luotiin nappi ja tähän nappiin täytyi luoda uusi funktio nimeltä createQuestionnare. Tämä funktio vastaa HTTP-pyynnön lähettämisestä Laravelin ohjaimelle, jonka avulla kysely tallennetaan tietokantaan. Sovellus hyödyntää Axios-kirjastoa HTTP-pyyntöjä varten. Axios-kirjaston avulla voidaan lähettää POST HTTP-pyyntö aiemmin luotuun create-reittiin. POST-pyynnölle annettiin parametriksi luotu reitti ja objekti, joka sisältää kyselyn nimen, kuvauksen ja sen sisältämät kysymykset. Luomaamme ohjaimeen täytyi luoda uusi funktio nimeltä create, jota reitti kutsuu saadessaan HTTP-pyynnön käyttäjän painaessa kyselyn luontia varten tehtyä nappia.

Axios-kirjaston post-funktiolle voidaan antaa parametrit käyttäen objektia, jotka se lähettää POST-pyynnön mukana (kuva 10).

```
createQuestionnaire() {
  axios.post('/elite/gritpro/questionnaire/editor/create/', {
    name: this.title,
    description: this.description,
    items: this.questions,
  }).then(() => {
    this.notyf.success('Kysely luotu onnistuneesti!');
    window.location.href = '/elite/gritpro';
  });
},
```

Kuva 10. Funktio HTTP-pyynnön lähettämiselle Laravelin ohjaimelle.

Ohjaimen create-funktion sisällä Laravel tarjoaa menetelmän purkaa HTTP-pyynnön sisältämä tieto. Funktiossa on parametri, jonka luokka on Request. Tämän avulla parametristä voidaan hakea tiedot objektin kenttien nimien mukaisesti Request-luokan input-funktion avulla. Kyselyn luontia varten täytyi hakea kirjautuneen käyttäjän käyttämä kieli, koska kysely luodaan tällä kielellä järjestelmään. Laravelin avulla voidaan käyttää session-funktiota, jonka avulla voidaan hakea sovelluksen tallentama kieli. Tietojen purkamisen jälkeen käyttäen input-funktiota saatiin nimi ja kuvaus merkkijonoina ja kysymykset taulukkona. Kyselyjen nimi ja kuvaus täytyy tallentaa JSON-muodossa siten, että kyseisen kielen lyhenteellä merkitään haluttu nimi ja kuvaus tietokantaan. Tämä onnistuu luomalla PHP-taulukot käyttäen käyttäjän kieltä sekä saatuja tietoja.

Tiedot voidaan purkaa POST-pyynnöstä ohjaimessa helposti käyttäen Laravelin Request-luokan funktioita (kuva 11).

```
$locale = session()->get('locale') ?? 'fi';

$name = $request->input('name');
$description = $request->input('description');
$items = $request->input('items');

$names = [$locale => $name];
$descriptions = [$locale => $description];
```

Kuva 11. Tietojen hankita HTTP-pyynnöstä ohjaimen sisällä.

Kyselyt on luotu käyttäen ryhmiä, mutta toimeksiannossa haluttiin kyselyt vain käyttäen yhtä ryhmää, joten tätä varten täytyi myös luoda uusi taulukko, joka sisälsi vain yhden ryhmän. Tälle yhdelle ryhmälle sijoitettiin kaikki kyselyn sisältämät kysymykset. Kysymysten taulukko piti käydä läpi sekä luoda jokaisesta

kysymyksestä oma uusi taulukko käyttäen käyttäjän valittua kieltä sekä kysymyksen asetuksia, sitten tämä luotu taulukko lisättiin ryhmään.

Ryhmän luomisen jälkeen se täytyi tallentaa tietokantaan. Tähän hyödynnettiin Templates-mallia, joka on Laravelin käyttämä kerros kommunikointia varten elite_gp_templates MySQL-taulun kanssa.

Mallissa näkyvät $fillable-kentät, jotka vastaavat MySQL-taulun kenttiä, niin niille voidaan asettaa arvot ohjaimen kautta. Kentät name ja description on merkitty $translatable-taulukossa, jonka avulla ne kääntyvät JSON-muotoon kyselyn luonnin yhteydessä Laravelin toimesta. (Kuva 12.)

```php
/**
 * @var array
 */
protected $fillable = ['club_id', 'name', 'user_id', 'data', 'status', 'description', 'code'];

/**
 * @var array
 */
protected $translatable = ['name', 'description'];
```

Kuva 12. Templates-malli elite_gp_templates MySQL-taulua varten.

Ennen kyselyn luontia täytyi hakea käyttäjän joukkue sekä hänen oman tilinsä tunnistenumero, jotka tarvitaan kyselyn luontia varten. Funktiota json_encode käytettiin kääntämään aiemmin luomamme taulukko JSON-muotoon, että se voidaan tallentaa tietokantaan. Templates-mallin avulla saatiin luotua kysely tietokantaan. (Kuva 13.)

```php
$team = Team::find(session()->get('active_team')->id);

$template = Templates::create([
    'club_id' => $team->club->id,
    'name' => $names,
    'description' => $descriptions,
    'user_id' => session()->get('active_account')->id,
    'data' => json_encode($groups),
    'status' => 2,
    'code' => $request->input('code', 'gen')
]);
```

Kuva 13. Kyselyn luonti tietokantaan käyttäen Templates-mallia.

Templates-malli täyttää muut kentät, jotka eivät ole $fillable-taulukossa mallin sisällä automaattisesti tietokantaan luonnin yhteydessä.

6.6 Kyselyiden vastausten laajentaminen

Kyselyiden sisältämiin kysymyksiin pystyi vastaamaan vain käyttäen asteikkoa tai tekstivastausta. Vaatimusmäärittelyn mukaisesti täytyi toteuttaa uudet komponentit, joiden avulla käyttäjä voi vastata kysymyksiin. Asteikko sekä tekstivastaus olivat molemmat toteutettu erillisinä Vue.js-komponentteina, joten tämän mukaisesti uudet komponentit myös luodaan erikseen. Uudet komponentit, jotka luotiin, olivat kyllä- tai ei-komponentti, pudotusvalikko-komponentti sekä muunnos pudotusvalikko-komponentista, joka mahdollisti käyttäjän valitsemaan monta eri vaihtoehtoa listalta yhden sijaan. Kehityksen edetessä kuitenkin pudotusvalikko-komponentti vaihdettiin vaihtoehtolistaan, koska se sopii paremmin kysymyksiin, joissa on vain muutama vaihtoehto valintaa varten.

Kysymykset näytetään erillisessä modal-ikkunassa, kun käyttäjä valitsee kyselyyn vastaamisen, jos sellainen hänelle on lähetetty. Tätä varten oli jo toteutus, joka näytti kyselyn kysymykset järjestyksessä sekä kysymykseen valitun komponentin käyttäjälle. Kysymysten vastaukset tallennetaan tietokantaan heti kysymykseen vastaamisen jälkeen. Tämän takia täytyi vain rekisteröidä uudet komponentit kyselyiden vastaamiseen tarkoitetussa Vue.js-komponentissa ja näyttää ne käyttäjälle, jos kysymyksessä oli tarkoitettu komponentti käytössä.

Ensimmäisenä komponenttina luotiin kyllä- tai ei-komponentti vastaamista varten. Tähän hyödynnettiin olemassa olevan tekstivastauksen komponentin pohjaa. Komponentti ei tarvinnut muita muutoksia, kun vain tekstikentän sijalle asetettiin kyllä- ja ei-napit. joita painamalla käyttäjän vastaus tallentuu tietokantaan.

Seuraavaksi luotiin vaihtoehtolista-komponentti. Tähän komponenttiin käytettiin myös samaa tekstivastauksen pohjaa, mutta komponentti tarvitsi myös erillisiä tietoja sen luonnin yhteydessä, koska komponentille täytyi toimittaa vaihtoehdot listaamista varten sekä myös onko kysymyksessä sallittu monta valintaa. Kyselyiden vastaamisen tarkoitettu komponentti hakee tiedot tietokannasta sekä tallentaa ne muuttujiksi komponenttiin. Tämän avulla voidaan hakea tallennetut

vaihtoehdot valintaa varten kysymyksen asetuksista, jotka käyttäjä asetti kyselyn luomista varten tarkoitetussa komponentissa.

Asetuksia voidaan syöttää komponentin rekisteröinnin yhteydessä, jonka avulla ne voidaan hakea rekisteröidyn komponentin props-objektista (kuva 14).

```
<select-list v-else-if="getComponent(groups.ordering, index).type === 'selectlist'" class="animated fadeIn"
             :question="template[groups.ordering].items[index].description[lang]"
             :help="template[groups.ordering].items[index].help[lang]"
             :group="group"
             :groups="groups"
             :options="getComponent(groups.ordering, index).settings.options"
             :multiselect="getComponent(groups.ordering, index).settings.allow_multiple" />
```

Kuva 14. Vaihtoehtolista-komponentin näyttäminen käyttäjälle sekä asetusten antaminen sen yhteydessä.

Komponentti saa asetukset luonnin yhteydessä ja ne listattiin käyttäjälle hyödyntäen HTML-syötekenttiä, valintaruutua ja valintanappia riippuen onko usean vaihtoehdon valinta sallittua (kuva 15).

```
<div v-for="option in options" :key="option.id">
  <div class="custom-control custom-switch my-1 pl-3 c-pointer">
    <div v-if="multiselect === 0">
      <input :id="'toggle-'+option.id" v-model="values" :value="option" type="radio" name="option"
             class="custom-control-input" :true-value="1" :false-value="0">
      <label class="custom-control-label" :for="'toggle-'+option.id">
        <small>{{ option.name }}</small>
      </label>
    </div>
    <div v-else-if="multiselect === 1">
      <input :id="'toggle-'+option.id" v-model="values" :value="option" type="checkbox"
             class="custom-control-input" :true-value="1" :false-value="0">
      <label class="custom-control-label" :for="'toggle-'+option.id">
        <small>{{ option.name }}</small>
      </label>
    </div>
  </div>
</div>
```

Kuva 15. Vaihtoehtolistan näyttäminen käyttäjälle komponentissa.

Vaihtoehtolista-komponentissa ei luotu vielä vastauksen tallentamista tietokantaan. Kyselyiden vastausten raportoinnissa ei ollut suoranaista tukea tälle, joka jää tulevaisuuden jatkokehitykseen.

6.7 Kyselyiden tietojen listaaminen

Kyselyistä ei ollut mahdollista nähdä, mitä kysymyksiä ne sisältävät ennen lähettämistä. Käyttäjälle näkyi vain kyselyn nimi eikä muita tietoja. Tämän vuoksi

täytyi listata valitun kyselyn kysymykset, jonka avulla käyttäjä voi tutkia kyselyn sisältöä ennen sen lähettämistä urheilijoille.

Kyselyiden lähettämistä varten on oma modal-ikkuna. Tämän toteutus poikkeaa aiemmasta kehitysestä siinä, että se on toteutettu käyttäen Laravel Blade-mallinnusmoottoria Vue.js:n sijaan. Tätä varten on luotu oma Laravel-näkymä, joka lähetetään käyttäjälle ohjaimen toimesta. Näkymän toimituksen yhteydessä ohjain hakee kyselyt tietokannasta ja toimittaa ne näkymän yhteydessä, jonka avulla saadaan kyselyt listattua pudotusvalikkoon. Kyselyitä varten on oma pudotusvalikko, joka listaa kaikki kyselyt, joita käyttäjällä on oikeudet lähettää urheilijoille.

Kyseessä on Laravel-näkymä, joka hyödyntää Blade-mallinnusmoottoria sen kääntämiseksi. Näkymissä hyödynnetään sisennettyä PHP-ohjelmakoodia käyttäen Laravelin omia direktiivejä.

Näkymän yhteydessä toimitetaan PHP-muuttuja $templates, joka sisältää kaikki järjestelmän kyselyt joihin käyttäjällä on pääsy. Kyselyt listataan käymällä läpi $templates-muuttuja hyödyntäen Blade-moottorin direktiiviä @foreach. (Kuva 16.)

```
<select id="template_id" name="template_id" class="form-control" style="width: 100%;">
    @foreach($templates as $template)
        <option value="{{ $template->id }}">{{ $template->name }}</option>
    @endforeach
</select>
```

Kuva 16. Kyselyn valitsemiseen luotu pudotusvalikko modal-ikkunassa.

Samalla tavalla voidaan listata kyselyn sisältämät kysymykset. Ongelmana oli, että Laravel-näkymä ei tarjoa dynaamisesti näkyviä elementtejä muuttujien mukaisesti, kuten Vue.js tarjoaa hyödyntäen v-if-direktiiviä. Blade-moottorin avulla voidaan käyttää @if-direktiiviä, mutta tämä sopii vain, jos käytetään PHP-muuttujia. Tätä ei voida hyödyntää pudotusvalikosta valitun vaihtoehdon kanssa. Tähän täytyi hyödyntää JavaScript-ohjelmakoodia.

Jokaisen kyselyn tiedot luodaan omaan elementtiin, joka ei ole näkyvissä käyttäjälle hyödyntäen @foreach-direktiiviä. Kyselyiden rakenteen vuoksi jouduttiin hyödyntämään monitasoista @foreach-silmukkaa. (Kuva 17.)

```
@foreach($templates as $template)
    @php $counter = 1 @endphp
    <div id="template_data{{ $template->id }}" style="display:none">
        @foreach($template->getGroupDescriptions() as $description)
            @foreach($description as $desc)
                <h4> Kysymys {{ $counter }} </h4>
                <h6> {{ $desc }} </h6>
                @php $counter = $counter + 1 @endphp
            @endforeach
        @endforeach
    </div>
@endforeach
```

Kuva 17. Kaikkien kyselyiden sisältämien kysymysten listaus.

Blade-moottori tarjoaa mahdollisuuden sisällyttää JavaScript-ohjelmakoodia näkymään, joka on dynaamisesti luotu PHP:n avulla. Tähän käytetään direktiiviä @push. JavaScriptin avulla voidaan näyttää tai piilottaa elementti sen mukaan, mikä kysely pudotusvalikosta on valittu.

Modal-ikkunan avautuessa pudotusvalikkoon rekisteröitiin tapahtuma, joka näyttää ensimmäisen kyselyn tiedot. Käytettiin ensimmäistä kyselyä, koska se on valittuna automaattisesti modal-ikkunan avautuessa. Sitten pudotusvalikkoon rekisteröitiin tapahtuma, kun sen valinta vaihtuu, niin otetaan pois näkyvistä kaikkien kyselyiden tiedot ja näytetään sitten uuden valitun kyselyn tiedot. (Kuva 18.)

```
@push('js')
$('#template_id').change(function () {
    @foreach($templates as $template)
        $('#template_data{{ $template->id }}').hide();
    @endforeach

    let val = $('#template_id').val();
    $('#template_data' + val).show();
});

$('#template_id').ready(function () {
    $('#template_data{{ $templates[0]->id }}').show();
});
@endpush
```

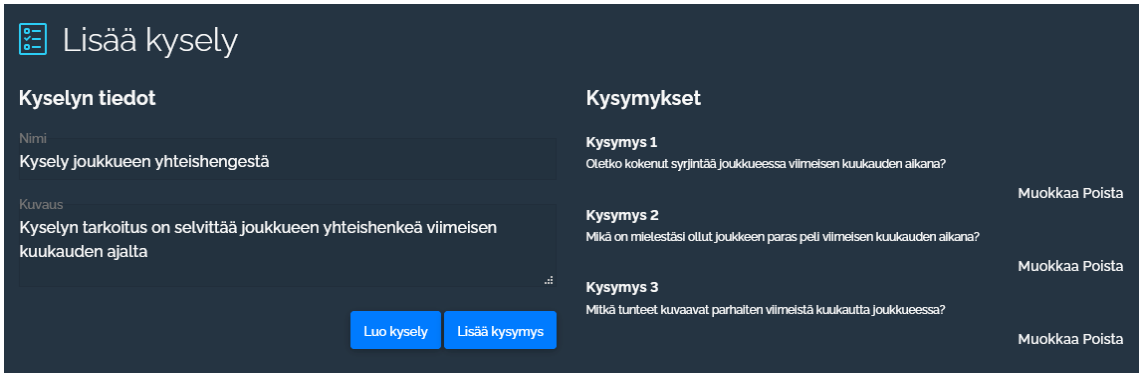
Kuva 18. JavaScript-koodin hyödyntäminen näkymässä elementtien näkyvyyden manipulointia varten.

Tämän tuloksena kyselyiden tiedot saatiin näkymään modal-ikkunassa käyttäjälle ennen kyselyiden lähettämistä. Kun käyttäjä valitsee uuden kyselyn pudotusvalikosta, modal-ikkuna näyttää vain valitun kyselyn sisältämät kysymykset.

7 Tulokset

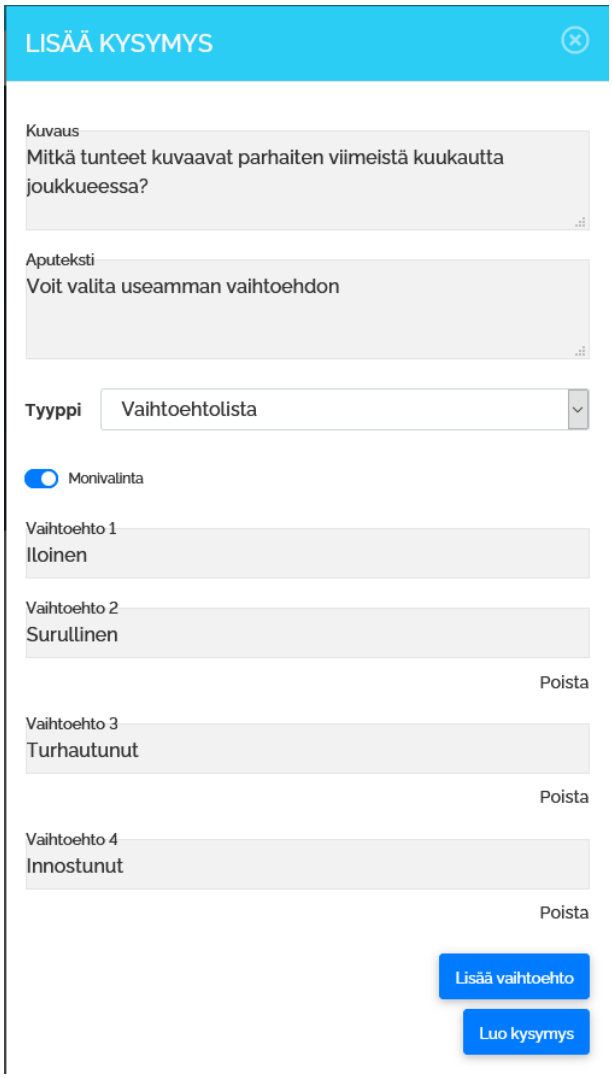
Opinnäytetyön tuloksena sovellukseen luotiin uudet ominaisuudet, jotka vastasivat toimeksiantajan vaatimusmäärittelyä. Käyttäjä voi luoda omia kyselyitä järjestelmään helposti käyttäen luotua käyttöliittymää, jossa on huomioitu käyttäjäystävällisyys. Kyselyihin voi lisätä kysymyksiä rajattomasti sekä kysymyksiä voi poistaa ja muokata jälkeenpäin, jos käyttäjä niin haluaa. Kysymyksille voi määrittää nimen sekä aputekstin vastaamista varten. Kysymyksille voi myös määrittää sille tarkoitetun komponentin vastaamista varten sekä komponentille voidaan antaa omat asetukset jokaista kysymystä kohden. Kyselyt tallennetaan tietokantaan käyttäjän joukkueen ja oman tunnistenumeron kanssa, mikä mahdollistaa kyselyiden näkymisen vain hänen joukkueellensa. Kyselyt tallennetaan käyttäjän omalla valitulla kielellä tietokantaan.

Kyselyiden luomista varten luotiin käyttäjäystävällinen näkymä, joka näyttää kyselyyn jo lisätyt kysymykset ja sen tiedot (kuva 19).



Kuva 19. Perusnäkymä kyselyn luontia varten.

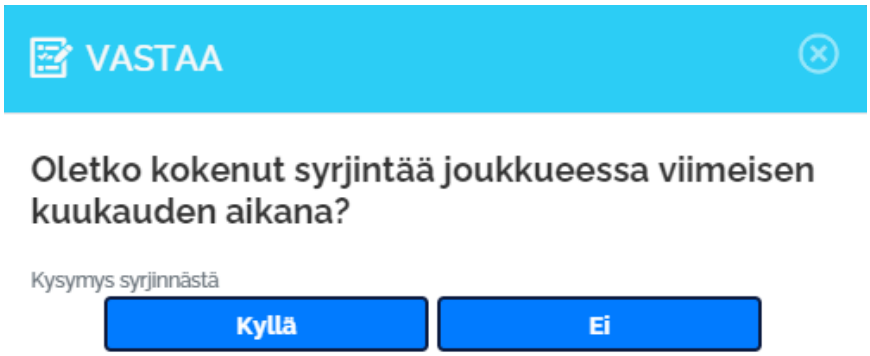
Lisää kysymys-nappia painamalla tuodaan esiin modal-ikkuna, jolla määritetään kysymyksen tiedot (kuva 20).



Kuva 20. Näkymä kysymyksen luontia varten.

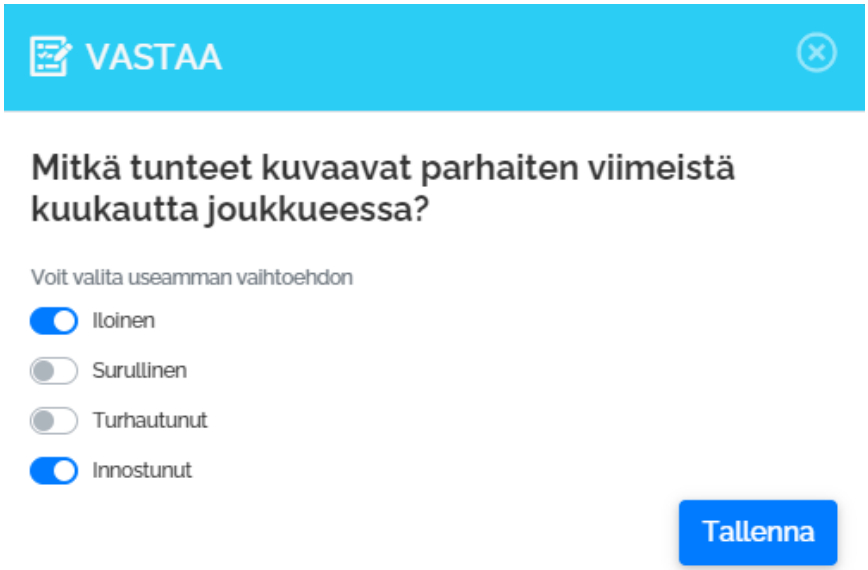
Kyselyiden kysymyksiin voi vastata laajemmin opinnäytetyön toteutuksen vuoksi, mikä mahdollistaa valmentajia luomaan parempia kyselyitä. Olemassa olevien asteikon ja tekstikentän lisäksi toteutuksen toimesta kysymyksiin voi määrittää vastaukseksi kyllä tai ei sekä vaihtoehtolistan, johon voi määrittää mahdollisuuden valita monta vaihtoehtoa kysymyksen asetuksista.

Komponenttia kyllä tai ei voidaan hyödyntää kysymyksiin, jotka eivät vaadi laajempaa vastausta (kuva 21).



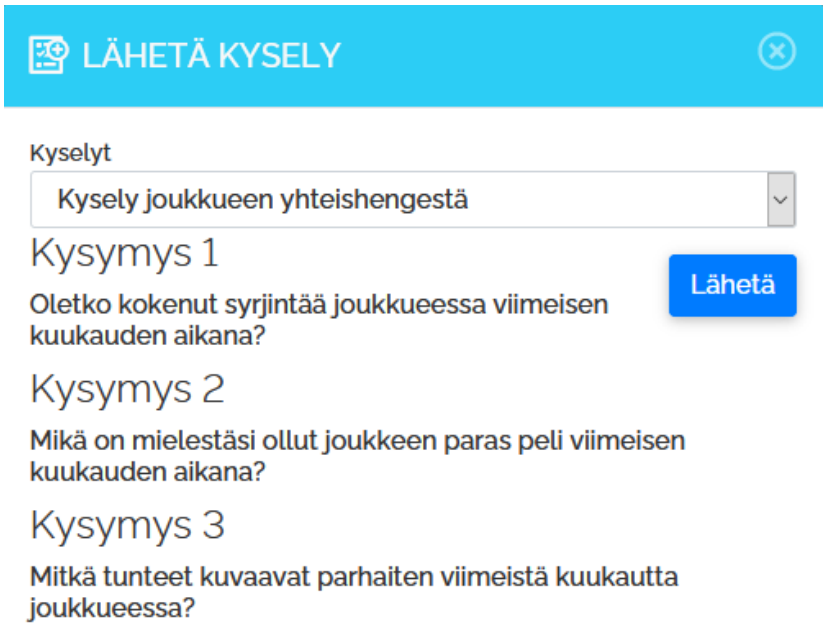
Kuva 21. Komponentti kyllä tai ei kysymykseen vastaamista varten.

Komponentti vaihtoehtolistaa varten hyväksyy monivalinnan vain, jos asetuksista tämä on asetettu päälle kysymyksen luonnin yhteydessä (kuva 22).



Kuva 22. Vaihtoehtolista-komponentti kysymykseen vastaamista varten.

Kyselyiden sisältämät kysymykset ovat näkyvissä kyselyn lähettäjälle, mikä mahdollistaa valmentajan tutustumisen kyselyyn ennen sen lähettämistä (kuva 23).



Kuva 23. Kyselyn sisältäminen kysymysten listaus.

Aiemmin kyselyistä ei ollut mahdollista nähdä muita tietoja kuin sen nimi. Tämän vuoksi muut valmentajat voivat myös käyttää muiden valmentajien luomia kyselyitä, koska he voivat katsoa kyselyn sisällön ja päättää, sopisiko tämä kysely juuri heidän joukkueensa käyttöön.

Opinnäytetyön tuloksena myös selvisi Vue.js:n tuoma hyöty nykyaikaisessa web-kehityksessä. Vue.js-sovelluskehys sopii erinomaisesti dynaamisten web-sivujen luontiin sen tarjoaman reaktiivisuuden vuoksi. Tämä mahdollistaa web-sivun dynaamisen näkymän käyttäjän syötteiden perusteella ilman suurempaa työtä kehittäjältä. Vue.js:n käyttämä viiksimalli, joka mahdollistaa ohjelmakoodin tai muuttujien suoran käytön HTML-ohjelmakoodin seassa, on mainio ominaisuus, kun toimitaan dynaamisten web-sivustojen kanssa. Vue.js myös tarjoaa erinomaisia direktiivejä, joiden avulla web-sivusto saadaan vielä dynaamisemmin toimimaan, kuten esimerkiksi vaihtoehtoinen elementtien piirtäminen. MVVM-arkkitehtuurin tuoma tietosidos näkymän ja näkymämallin välille nopeuttaa kehitystä, koska ei tarvitse erikseen kommunikoida ohjaimen kanssa, kuten MVC-arkkitehtuurimallissa. Vue.js:n avulla voidaan kehittää suuriakin web-sovelluksia, kunhan se yhdistetään vain jonkin toisen sovelluskehyksen tai työkalun kanssa kuten Laravelin.

8 Pohdinta

Vaatimusmäärittely muuttui hieman kehityksen aikana, mikä aiheutti omia ongelmia kehityksessä. Full-stack-kehitystyö toi haasteita opinnäytetyössä, koska täytyi opetella front-end- ja back-end-osioita samanaikaisesti. Käytössä oli myös sovelluskehykset, josta oli vain pintapuolisesti tietoa ennen kehitystä. Tämä toi myös haasteita front-end- ja back-end-osioiden kehityksessä, mutta tietoperustaa laatiessa niistä syntyi hyvä kuva sekä niiden tärkeimmistä toiminnoista ja ominaisuuksista.

Opinnäytetyössäni sovelluskehykset tarjosivat kattavat työkalut kehitykseen sekä sovelluskehysten laaja dokumentointi nopeutti kehitystä. Opinnäytetyö toteutettiin full-stack-kehityksenä, mutta suurin osa kehityksestä kuitenkin keskittyi front-end-kehitykseen Vue.js-sovelluskehyksen avulla. Vue.js-sovelluskehyksen pariin oli helppo päästä eikä mennyt kauan aikaa, kunnes oppi sen tärkeät ominaisuudet jatkokehityksen aikaansaamiseksi. Laravel- ja Vue.js-sovelluskehykset kuitenkin molemmat ovat laajoja kokonaisuuksia, niin tämän opinnäytetyön perustella sain vain pintapuolisen käsityksen niiden sisältämäistä ominaisuuksista. Opinnäytetyö antoi hyvän kuvan web-sovellusten nykyaikaisesta kehitysestä sekä suuren sovelluskokonaisuuden pienen osan kehityksestä.

Sovellukseen jäi useita jatkokehityksen mahdollisuuksia. Kyseessä on kansainvälinen sovellus sekä joissakin joukkueissa on pelaajia useista eri maista. Tätä varten olemassa oleviin kyselyihin voisi luoda mahdollisuuden lisätä käännökset eri kielille valmentajien toimesta. Kyselyiden vastausten perusteella voidaan luoda raportteja, joista valmentajat näkevät visuaalisesti pelaajien vastaukset, mutta raportointi ei ota huomioon tämän opinnäytetyön jatkokehityksessä luotuja komponentteja vastauksia varten, joten tämä jäi myös tulevaisuuden jatkokehitykseen. Myös käyttöliittymä luotiin tietokoneen web-selain mielessä ja testattiin web-selaimen avulla, mutta sovellusta käytetään myös mobiililaitteilla paljon, joten tätä varten myös käyttöliittymästä voisi tehdä mobiiliystävällisemmän tulevaisuudessa.

Lähteet

Christensson, P. 2006a. CSS Definition. https://techterms.com/definition/css. 25.1.2021.

Christensson, P. 2006b. PHP Definition. https://techterms.com/definition/php. 25.1.2021.

Christensson, P. 2007. SQL Definition. https://techterms.com/definition/sql. 25.1.2021.

Christensson, P. 2011. SaaS Definition. https://techterms.com/definition/saas. 18.2.2021.

Christensson, P. 2014. JavaScript Definition. https://techterms.com/definition/javascript. 25.1.2021.

Christensson, P. 2015a. HTML Definition. https://techterms.com/definition/html. 25.1.2021.

Christensson, P. 2015b. HTTP Definition. https://techterms.com/definition/http. 25.1.2021.

Christensson, P. 2018. MVC Definition. https://techterms.com/definition/mvc. 29.1.2021.

Docker Docs. 2020. Docker overview. https://docs.docker.com/get-started/overview/. 8.12.2020.

Educba. 2020. laravel vs Ruby on Rails. https://www.educba.com/laravel-vs-ruby-on-rails/. 17.11.2020.

Filipova, O. 2016. Learning Vue.js 2. Birmingham, UK: Packt Publishing Ltd. Google-kirjat. https://books.google.fi/books?id=nszcDgAAQBAJ&printsec=frontcover. 15.10.2020.

GeekInsta. 2019. Difference between MVC and MVT architecture. https://www.geekinsta.com/difference-between-mvc-and-mvt/. 21.1.2021.

Github. 2020. initial setup. https://github.com/vuejs/vue/commit/83fac017f96f34c92c3578796a7ddb443d4e1f17. 10.10.2020.

HeidiSQL. 2020. HeidiSQL. https://www.heidisql.com/. 8.12.2020.

Kaluža, M., Kalanj, M. & Vukelić, B. 2019. A COMPARISON OF BACK-END FRAMEWORKS FOR WEB APPLICATION DEVELOPMENT. https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=321176&lang=en. 25.10.2020.

Laravel. 2020a. JavaScript & CSS Scaffolding. https://laravel.com/docs/7.x/frontend. 27.10.2020.

Laravel. 2020b. Database: Getting Started. https://laravel.com/docs/8.x/database. 13.10.2020.

Laravel. 2020c. Database: Query Builder. https://laravel.com/docs/8.x/queries. 26.10.2020.

Laravel. 2020d. Database: Migrations. https://laravel.com/docs/8.x/migrations. 27.10.2020.

Laravel. 2020e. Database: Seeding. https://laravel.com/docs/8.x/seeding. 26.10.2020.

Likness, J. 2014. Model-View-ViewModel (MVVM) Explained. Blog. 24.4.2014. https://www.wintellect.com/model-view-viewmodel-mvvm-explained/. 29.1.2021.

Mozilla. 2021. Introduction to the DOM. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. 29.1.2021.

Nader, Y. 2020. Django Vs Laravel: Which framework to choose?. Programming Blog. 1.11.2020. https://hackr.io/blog/django-vs-laravel. 17.11.2020.

Ryabtsev. A. 2020. Web Frameworks: How To Get Started. Django Stars Blog. 2.11.2020. https://djangostars.com/blog/what-is-a-web-framework/. 2.11.2020.

Sagara Technology. 2020. The Fundamentals of Front End and Back End Development. Blog. 2.9.2020. https://www.sagaratechnology.com/blog/the-fundamentals-of-front-end-and-back-end-development/. 24.10.2020

Stauffer, M. 2019. Laravel: Up & Running: A Framework for Building Modern PHP Apps. Massachusetts, USA: O'Reilly Media, Inc. Google-kirjat. https://books.google.fi/books?id=HcqPDwAAQBAJ&printsec=frontcover. 5.10.2020.

Surguy, M. 2013. History of Laravel PHP framework, Eloquence emerging. Blog. 27.7.2013. https://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging/. 10.10.2020.

Visual Studio Code. 2020a. Getting Started. https://code.visualstudio.com/docs/. 8.12.2020.

Visual Studio Code. 2020b. Using Version Control in VS Code. https://code.visualstudio.com/docs/editor/versioncontrol/. 8.12.2020.

Vue.js. 2016. Announcing Vue.js 2.0. Vue.js. 27.4.2016. https://vuejs.org/2016/04/27/announcing-2.0/. 8.10.2020.

Vue.js. 2020a. Introduction. https://vuejs.org/v2/guide/. 19.10.2020.

Vue.js. 2020b. Comparison with Other Frameworks. https://vuejs.org/v2/guide/comparison. 17.11.2020.

Vue.js. 2020c. The Vue Instance. https://vuejs.org/v2/guide/instance. 20.10.2020.

Vue.js. 2020d. Template Syntax. https://vuejs.org/v2/guide/syntax. 20.10.2020.

Vue.js. 2020e. Component Basics. https://vuejs.org/v2/guide/components. 19.10.2020.

You, E. 2014. First week of launching Vue.js. Evan You. 11.2.2014. https://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/. 5.10.2020.