Erno Kulmala

KYYTI LIKEL –SOVELLUKSEN TEKNINEN TOTEUTUS

Tietojenkäsittelyn koulutusohjelma

2017

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

KYYTI LIKEL –SOVELLUKSEN TEKNINEN TOTEUTUS

Kulmala Erno
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
joulukuu 2017
Sivumäärä: 35

Asiasanat: ionic, laravel, hybridisovellus, joukkoliikenne

Tämän opinnäytetyön tavoitteena oli Kyyti likel –sovelluksen kehittäminen. Kyyti likel –sovellus on Porin Linjoille kehitetty sovellus, jolla voidaan ostaa matkalippuja busseihin sekä nähdä aikataulut ja reaaliaikaisesti bussit kartalla.

Työssä kerrottiin tarkemmin sovelluksessa ja taustajärjestelmässä käytetyistä sovelluskehyksistä, eli Ionicista sekä Laravelista. Tämän lisäksi käydään läpi, että miksi juuri nämä sovelluskehykset päädyttiin ottamaan käyttöön tässä projektissa.

Käytännön osuudessa käytiin läpi sovelluksen ja taustajärjestelmän ominaisuudet ja miten ne keskustelevat keskenään. Siinä kerrottiin myös työn vaiheista ja ongelmista, joita kohdattiin kehityksen edetessä.

Lopputuloksena saatiin iOS- ja Android-alustoille toimiva sovellus, joka on saanut positiivista palautetta. Opinnäytetyössä käytiin läpi kehityksen vaiheita sovelluksen lanseeraukseen asti. Sovellusta on kehitetty myös sen jälkeen ja tullaan jatkossa kehittämään.

BUILDING THE KYYTI LIKEL APPLICATION

Kulmala, Erno
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Technology
December 2017
Number of pages: 35

Keywords: ionic, laravel, hybrid application, public transport

---

The purpose of my thesis was to develop the Kyyti likel application for Pori Linjat, which is the public transport operator for the City of Pori. In the application user can pay bus fares and see bus timetables and buses on the map in real time.

The thesis provided more detailed description of the application frameworks, Ionic and Laravel and reasoning why they were implemented in this project.

The practical section examined the features of the application and application frameworks and how they interact with each other. It also told about the work phases and problems that were encountered as development progressed.

The end product was an application running on iOS and Android platforms. The thesis analyzed the stages of development until the application was launched. The application has received a lot of positive feedback after the release and development work has also been continued to provide new features in the near future.

SISÄLLYS

1 JOHDANTO

Tässä opinnäytetyössä kerron Porin Linjoille tehdyn Kyyti likel –sovelluksen kehityksestä ja siinä käytetyistä teknologioista. Projekti lähti liikkeelle kovalla vauhdilla ja pohjatyö jouduttiin tekemään melko nopealla aikataululla. Tehtyihin valintoihin ollaan kuitenkin hyvinkin tyytyväisiä, sillä sovelluksesta on pidetty paljon ja esiintyneiden ongelmien määrät ovat hyvin vähäisiä käytön laajuuteen verrattuna.

Päätös sovelluksen toteuttamisesta tapahtui Porin Linjojen ja Porilaisen luovan teknologian suunnittelutoimisto Aline Creative Technology Studio Oy:n yhteistyön pohjalta. Sovellus on ensimmäinen laatuaan Suomessa. Suomessa on tarjolla useita sovelluksia, joissa voi seurata joukkoliikenteen kulkuvälineitä ja nähdä aikataulut, mutta niissä ei voi ostaa lippuja. Kyyti likel –sovellukseen on tuotu nämä molemmat toiminnot.

Ionic- ja Laravel-sovelluskehyksiä ollaan avattu melko pintapuolisesti, mutta niistä on kerrottu niitä asioita, joiden vuoksi ne ovat juuri sopivia teknologioita tämänkaltaisen työn tekemiseen.

2 MOBIILISOVELLUKSESSA KÄYTETYT TEKNIIKAT

2.1 Mobiilisovelluksen toteuttamistavat

Sovelluksia on kolmenlaisia: natiivi-, hybridi- ja web-sovelluksia.

Natiivisovellus tarkoittaa sitä, että ohjelma on ohjelmoitu niin, että se käyttää suoraan käyttöjärjestelmän ohjelmointirajapintaa. Esimerkiksi kehitettäessä sovellusta iOS-

käyttöjärjestelmään, tulee ohjelmointikielenä käyttää Swiftiä tai Objective-C:tä, kun taas Androidille tulee käyttää Java-ohjelmointikieltä. Apple ja Google tarjoavat sovelluskehittäjille omia kehitystyökalujaan. iOS-käyttöjärjestelmälle Xcode ja Androidille Android Studio. (Saccomani 2017.)

Web-sovellus on käytännössä tavallinen verkkosivusto. Web-sovelluksella on suora pääsy vain harvaan natiivikomponenttiin. Natiivikomponentilla tarkoitetaan esimerkiksi puhelimen kameraa. (Marketing And Growth Hacking 2017.)

Hybridisovellus taas tarkoittaa natiivisovelluksen ja web-sovelluksen välimaastoa. Hybridisovellus toimii kuten natiivisovellus, mutta itse sovellus pyörii Webview-nimisen komponentin päällä, mikä käytännössä tarkoittaa verkkoselainta. Hybridisovelluksella on kuitenkin pääsy natiivikomponentteihin. Pääsyä suoraan natiivikomponentteihin ei kuitenkaan ole. Natiivikomponenttiin käsiksi pääseminen vaatii käyttöjärjestelmän käyttämällä ohjelmointikielellä ohjelmoidun lisäosan, joka avaa liittymän natiivikomponenttiin. (Saccomani 2017.)

Kaikilla toteuttamistavoilla on omat puolensa. Halvin tapa on kehittää web-sovellus, jolloin sitä voidaan käyttää kaikilla laitteilla verkkoselaimella. Kuten web-sovelluksessa, hybridisovelluksella ohjelmakoodia ei tarvitse kirjoittaa erikseen jokaiselle käyttöjärjestelmälle erikseen, vaan samaa ohjelmakoodia voidaan käyttää kaikilla alustoilla. Hybridisovellus kuitenkin vaatii käytettävältä alustalta, kuten tämän opinnäytetyön projektissa käytetyltä Cordovalta, tukea käyttöjärjestelmälle. Natiivisovelluksissa sama ohjelmakoodi täytyy kirjoittaa erikseen jokaiselle käyttöjärjestelmälle, mikä tarkoittaa huomattavasti suurempaa resurssien tarvetta. (Marketing And Growth Hacking 2017.)

Tässä projektissa on käytetty Ionic-nimistä sovelluskehystä, jolla saadaan tuotettua hybridisovelluksia.

2.2 Ionic

2.2.1 Ionic pähkinänkuoressa

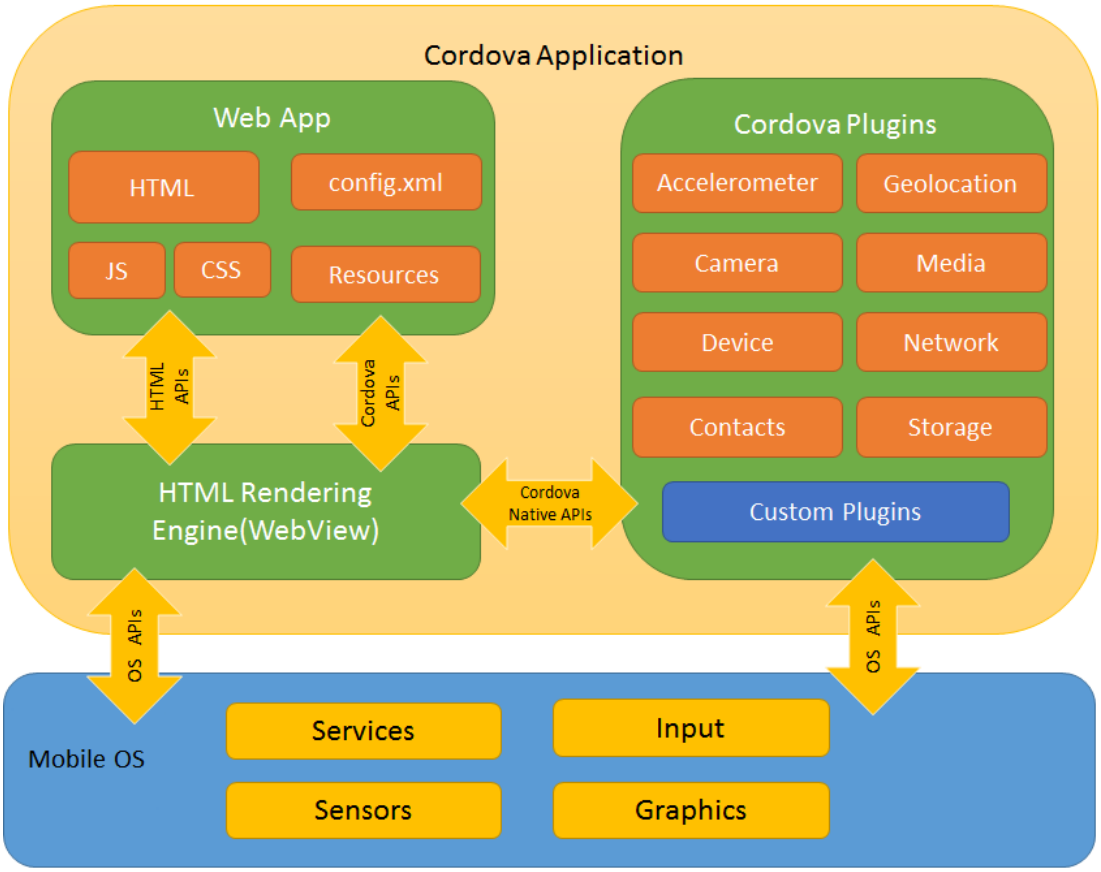
Ionic on avoimena lähdekoodina toteutettu sovelluskehys. Se on rakennettu Cordova-alustan päälle ja käyttää Angular-sovelluskehystä. Ionicilla on takanaan suuri yhteisö. Vuoden 2017 alkuun mennessä Ionicilla oli luotu vuodesta 2014 asti jopa neljä miljoonaa sovellusta. (Lynch 2017.)

2.2.2 Angular

Angular on verkkosivustoille tarkoitettu sovelluskehys, joka suoritetaan käyttäjän verkkoselaimen päässä. Angularia lähdettiin kehittämään vuonna 2009 sivuprojektina. Vuonna 2010 toinen kehittäjistä aloitti työt Googlella Google Feedback –projektin parissa. Hän sai vakuutettua pomonsa sen tuomista säästöistä ajassa ja koodissa, ja hän tiiminsä kanssa kirjoitti uudelleen Google Feedbackin koodin käyttäen Angularia. Koodirivien määrä putosi 17000:sta 1500:aan ja ajankäyttö väheni kolmelta ohjelmoijalta puolesta vuodesta kolmeen viikkoon. Hetken päästä tästä sama tiimi julkaisi kirjaston avoimena lähdekoodina ja ensimmäinen versio saatiin ulos toukokuussa 2011. (VanToll 2017.)

2.2.3 Apache Cordova

Apache Cordova on ohjelmistokehys, joka yhdistää web-sovellukset ja laitteiden natiiviominaisuudet. Apache Cordovan ansiosta voi käyttää perinteisiin verkkosivustojen luontiin tarkoitettuja HTML-, CSS- ja JavaScript-tekniikoita ja kääntää ne mobiilisovellukseksi. Suurin hyöty Apache Cordovan käytössä on se, että yhden koodipohjan voi kääntää monelle eri käyttöjärjestelmälle ilman, että tarvitsisi tehdä käyttöjärjestelmäkohtaisia muutoksia sovellukseen. (Apache Cordova 2017.)



Kuva 1. Apache Cordova -sovelluksen arkkitehtuuri.

2.2.4 Kehittäminen Ionicilla

Ionic-sovellusten luominen ja kääntäminen tapahtuvat komentorivipohjaisella työkalulla. Työkalu pohjautuu Node.js-teknologiaan. Node.js:n mukana tulee NPM-pakettienhallintajärjestelmä, jonka avulla saadaan helposti lisättyä kirjastoja Ionic-sovellukseen.

Ionic-sovelluksen luonti tapahtuu komentorivillä suorittamalla ”*ionic start projektin_nimi*”-komento. Komento kysyy vielä, että mille pohjalle projekti halutaan luoda. Vaihtoehtoja ovat muun muassa tyhjä-, välilehdellinen- sekä sivuvalikko-pohjaiset sovellukset. Työkalu luo valinnan mukaisen sovelluksen pohjan, joka on valmis käytettäväksi.

Sovelluksen testaaminen on helppoa, sillä suorittamalla komennon ”*ionic serve*”, Ionic paketoi projektin sovellukseksi, jota voi käyttää selaimella. Tämän lisäksi Ionic myös

käynnistää palvelinsovelluksen, joka tarjoaa sovelluksen tiedostot selaimelle. Kehityksessä suurena apuna on ominaisuus, että kun sovelluksen koodiin tehdään muutoksia palvelinsovelluksen ollessa päällä, Ionic paketoi muutetun koodin ja päivittää selaimeen automaattisesti uusimman version. Mikäli muutos tehdään CSS-tyylitiedostoon, muutokset astuvat voimaan heti paketoinnin jälkeen. Jos muutos tehdään HTML- tai JavaScript-tiedostoihin, sovellus käynnistää itsensä uudelleen paketoinnin jälkeen ladatakseen uusimman version. Suurena miinuksena tässä on natiivikomponenttien puute. Jotkut natiivikomponentit toimivat myös selaimella, mutta se vaatii, että natiivikomponentti on tarjolla myös selainversiona. Useimmant natiivikomponentit ovat tarjolla vain iOS- ja Android-versioina.

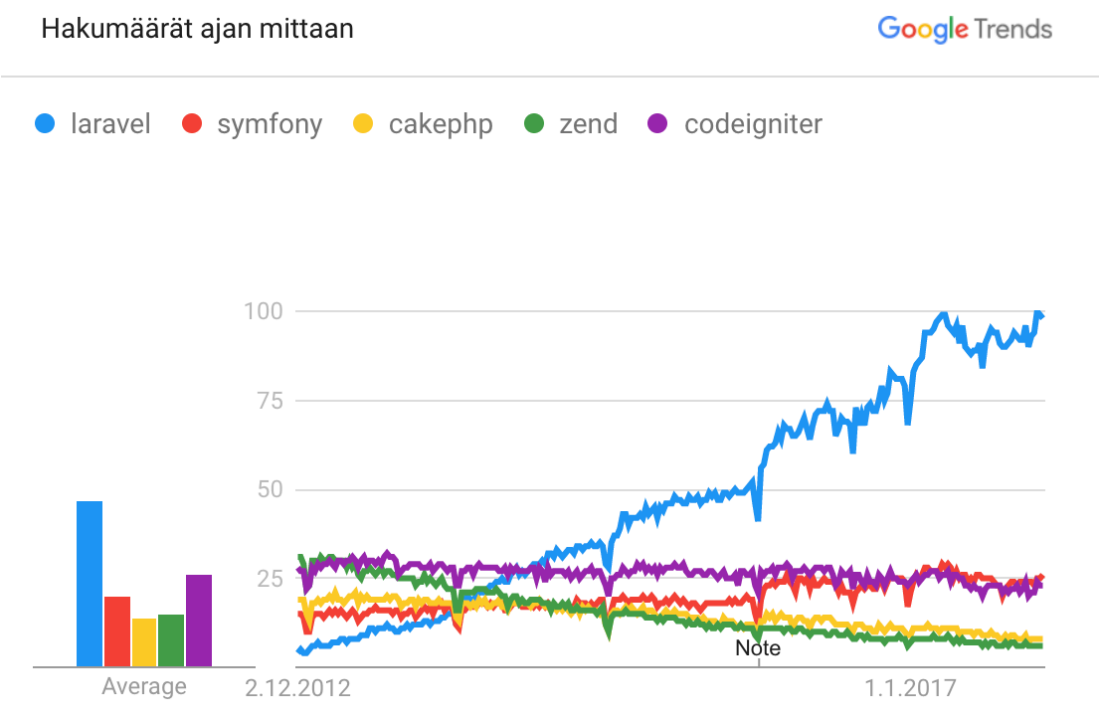
Kirjastojen ja natiivikomponenttien asennus sovellukseen on helppoa. Esimerkiksi suorittamalla komennot ”*ionic cordova plugin add phonegap-plugin-push*” ja ”*npm install --save @ionic-native/push*” saadaan asennettua Push-ilmoitusten natiivikomponentti.

Sovellukseksi paketoiminen on myös helppoa. Komennolla ”*ionic cordova build ios*” saadaan valmis iOS-sovellus. Vaihtamalla ios-tekstin tilalle android, saadaan aikaiseksi valmis Android-sovellus. Lisäämällä komentoon argumentit ”--release” ja ”--prod” saadaan sovelluksesta julkaisukelpoinen versio.

2.3 Laravel

2.3.1 Laravel pähkinänkuoressa

Laravel on avoimen lähdekoodin ohjelmistokehys, joka on toteutettu PHP:lla. Se on suosituin PHP:lla toteutettu ohjelmistokehys tällä hetkellä. (Zen Of Coding 2017). Laravelin tehokkuus perustuu myös uusimman PHP-version käyttämiseen. Laravelin heinäkuussa 2017 julkaistu 5.5-versio vaatii toimiakseen vähintään PHP:n 7-versiota (Barnes 2017). Monet ohjelmistokehykset ja sisällönhallintajärjestelmät mahdollistavat vanhempien PHP-versioiden tuen, minkä vuoksi ne eivät voi ottaa käyttöön PHP:n uusimpia ominaisuuksia.



Kuva 2. Google-haut PHP-ohjelmistokehyksistä viimeiseltä viideltä vuodelta.

Laravelista on helppo tykätä. Web-sovelluksen tekeminen on helppoa ja nopeaa. Selkeä dokumentointi ja erilaisten oppaiden suuri määrä auttavat kehittäjiä ymmärtämään Laravelia ja miten sitä kannattaa käyttää web-sovelluksen kehityksessä. Itselleni eteen ei ole vielä kirjoitushetkellä tullut yhtään ongelmaa, johon ei olisi saanut vastausta nopealla Googlettamisella.

2.3.1.1 Toimintamalli

Ohjelmistokehyksien perusperiaate on ”älä keksi pyörää uudelleen”. Käytännössä tämä tarkoittaa sitä, että tietyt toiminnallisuudet tarjotaan valmiiksi, joita tullaan tarvitsemaan lähes jokaisessa web-sovelluksessa. Yksi tällainen toiminnallisuus on esimerkiksi käyttäjän todentaminen. Laravel tarjoaa lähes valmiina käyttäjän rekisteröitymisen sekä kirjautumisen. Tämä on ainoastaan yhden komentorivikomennon takana, mikä luo automaattisesti näkymät rekisteröitymiselle ja kirjautumiselle sekä hoitaa niiden logiikan.

Sen lisäksi, että ohjelmistokehys tarjoaa valmiita toiminnallisuuksia, Laravel tarjoaa myös turvallisuusominaisuuksia. Se esimerkiksi tarjoaa salasanan tallennukseen turvallisen ratkaisun, CSRF- sekä XSS-suojaukset, SQL-injektioiden välttäviä rajapintoja tietokantaan sekä turvalliset evästeet. (Tutorials Point 2017.)

2.3.2 Artisan

Artisan on komentorivipohjainen ohjelma, joka tulee Laravelin mukana. Sillä on mahdollista suorittaa monenlaisia komentoja, kuten tietokantamigraatioiden luominen ja suorittaminen. (Laravel 2017.)

Tietokantamigraation luominen tapahtuu komennolla ”php artisan make:migration MigraationNimi”. Komento luo uuden PHP-luokan (kuva 3), jota voidaan täydentää ennen migraation suorittamista. PHP-luokassa voidaan määritellä tietokantaan tehtäviä muutoksia, kuten taulujen luomista tai sarakkeiden lisäämistä ja poistamista. Migraatio suoritetaan komennolla ”php artisan migrate”.

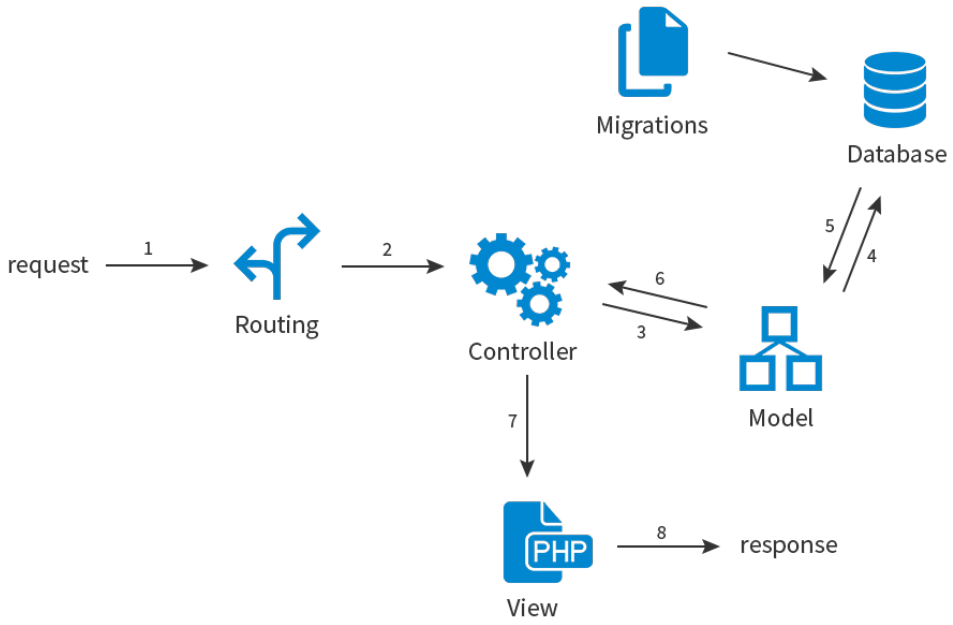
```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class MigraationNimi extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        //
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        //
    }
}
```

Kuva 3. Artisanin luoma PHP-luokka.

Artisaniin on mahdollista luoda omia komentoja. Tässä projektissa niitä on muun muassa 10-korttien vanhentumisen tarkistus ja ilmoitusten lähettäminen käyttäjille.

2.3.3 MVC-malli Laravelissa

Laravel toimii MVC-periaatteella. MVC, eli **M**odel (suom. Malli), **V**iew (suom. Näkymä) ja **C**ontroller (suom. Kontrolleri). Pyyntö ohjataan reitittimelle, joka ohjaa pyynnön eteenpäin oikealle kontrollerille. Kontrolleriin luodaan toimintalogiikka, jolla voidaan hakea tietoa tietokannasta mallin avulla. Kontrolleri ei välttämättä käytä yhtään mallia toimintalogiikassa, mikäli ei ole tarvetta. Kontrolleri palauttaa näkymän, jolle se on antanut tarvittavat tiedot näkymän rakentamiseen. Kuvassa 4 on esitetty MVC-mallin toimintaperiaate Laravelissa.



Kuva 4. MVC-malli kuvattuna. (Lekovic n.d.)

Laravelissa määritetään reititykset, joiden avulla tietty osoite osataan ohjata oikealle kontrollerille. Reititystietoihin on mahdollista määrittää suoraan toimintalogiikka, jolloin kontrolleria ei tarvita. Osoitteelle voidaan määritellä kyselyn tyyppi, esimerkiksi GET- tai POST-pyyntö. Tämän lisäksi voidaan muun muassa määrittää, että vaaditaanko osoitteessa käyttäjän todennusta, tiettyjä parametreja tai esimerkiksi tiettyä käyttöoikeutta.

Kontrollerit ovat PHP-luokkia, joissa sijaitsee toimintalogiikka. Käytännössä reitityksessä on määritelty, että mille kontrolleriluokalle ja sen metodille pyyntö ohjataan. Metodi antaa vastauksen, joka ohjataan käyttäjälle. Vastaus voi olla esimerkiksi näkymä, tekstiä tai vaikkapa JSON-muotoista tietoa.

Lähes jokaisella tietokannan taululla on oma mallinsa. Poikkeuksena ovat monen suhde moneen –suhteet, joille ei tarvitse luoda omaa mallia. Tällainen tapaus on esimerkiksi käyttäjälle määritettävä käyttäjärooli, missä monella käyttäjällä voi olla monta käyttäjäroolia. Mallissa on muun muassa määritelty, että missä tietokannan taulussa tieto sijaitsee, suhteet sekä kenttien tietotyyppejä.

Näkymiin on luotu valmiiksi pohja, joka täydennetään kontrollerista saatavalla tiedolla. Laravelissa näkymät käyttävät Blade-mallia (eng. Blade template). Blade-malli eroaa muista suosituista mallinnusmoottoreista niin, että se sallii PHP:n käyttämisen näkymissä. Blade-mallit tallennetaan välimuistiin puhtaana PHP:na, missä sitä säilytetään siihen asti, että näkymään tehdään muutoksia. Kuvassa 5 on esimerkki näkymästä, joka on toteutettu Blade-mallia käyttäen. (Laravel 2017.)

```
1   @extends('layouts.site')
2
3   @section('content')
4   <h1 class="text-center">FAQ</h1>
5   <div id="faqs">
6       @foreach($faqs AS $faq)
7           <div class="qa">
8               <h2>{{ $faq->question }}</h2>
9               <div class="answer">
10                  {{ $faq->answer }}
11              </div>
12          </div>
13      @endforeach
14  </div>
15  @endsection
```

Kuva 5. FAQ-sivun näkymä.

2.3.4 Autentikointi

Laravelissa autentikointi on tehty valmiiksi, mutta vaatii käyttöönottamiseksi kaksi komentorivikomentoa. Komennoilla ”php artisan make:auth” ja ”php artisan migrate” saadaan luotua kontrolleri sekä näkymät kirjautumista, rekisteröintiä sekä salasanan palautusta varten. (Laravel 2017.)

Laravel tarjoaa Passport-lisäosan kautta autentikoinnin API:n (suom. Ohjelmointirajapinta) kautta. Tämän avulla käyttäjän autentikointi saadaan tuotua sovellukseen. Passport on rakennettu Leaguen OAuth 2.0 Serverin päälle. Passport palauttaa kirjautumisessa turvatunnisteen (eng. Token). Käytännössä sovellus sisällyttää tämän turvatunnisteen jokaiseen pyyntöön, jotta käyttäjä voidaan tunnistaa jokaisessa pyynnössä. (Laravel 2017.)

2.3.5 Eloquent ORM

Laraveliin sisällytetty Eloquent ORM (Object-relational mapping) tarjoaa Active Record-toteutuksen. Active Record –toteutuksessa objektit sisältävät datan sekä toimintalogiikan, jolla käsitellään objektin sisältämää dataa. Active Record tarjoaa mallille käyttölogiikan, jolla käyttäjä voi lisätä ja muuttaa dataa. (Rails Guides 2017.)

Eloquent tarjoaa helpon rajapinnan tiedon hakemiseen tietokannasta. Malliin ei tarvitse itse kirjoittaa tietokantakyselyä, vaan Eloquent hoitaa sen puolen. Jos esimerkiksi on olemassa luokka Koulu ja halutaan saada kaikki koulut, voidaan se hakea yksinkertaisesti koodilla *Koulu::all()*. Tämä hakee taulukkoon kaikki koulut, jotka ovat tietokannassa. Haettavia tietoja voidaan myös helposti rajata ja järjestää. Kuvissa 6 ja 7 on esitetty esimerkkejä Eloquentin käytöstä. (Laravel 2017.)

```
Esimerkkitietokannan rakenne:

Taulu   Kentät
Koulu   id, nimi
Luokka  id, nimi, koulu_id

<?php
foreach(Koulu::all() AS $koulu) {
    print sprintf("Koulun %s luokat:\n", $koulu->nimi);

    foreach($koulu->luokat AS $luokka) {
        print sprintf("%s\n", $luokka->nimi);
    }

    print "\n";
}
```

Kuva 6. Eloquentin käytöstä esimerkki, jossa tulostetaan kaikki tietokannassa olevien luokkien nimi.

```
Esimerkkitietokannan rakenne:

Taulu   Kentät
Koulu   id, nimi, kunta_id
Luokka  id, nimi, oppilasmaara, koulu_id
Kunta   id, nimi

<?php
$luokat = Luokka::with('Koulu')->whereHas('Koulu', function($query) {
    $query->with('Kunta')->whereHas('Kunta', function($query2) {
        $query2->where('nimi', 'Pori');
    });
})->orderBy('oppilasmaara', 'DESC')->get()->all();

foreach($luokat AS $luokka) {
    print sprintf("Koulun %s, luokka: %s, oppilasmäärä: %d\n",
        $luokka->koulu->nimi,
        $luokka->nimi,
        $luokka->oppilasmaara);
}
```

Kuva 7. Eloquentin käytöstä esimerkki, jossa haetaan Porilaisten koulujen luokat oppilasmäärän mukaan järjestettynä.

3 MOBIILISOVELLUS

3.1 Teknologian valinta

Mobiilisovellusta varten tehtiin paljon selvitystyötä, että minkälainen pohja sovellukselle valittaisiin. Ionic astui kuvioon melko aikaisessa vaiheessa ja sitä ryhdyttiin tutkimaan lähemmin. Aiempi kokemus kehittää sovellus pelkällä Cordovalla oli jo sen verran hyvä, että tästä alkoi muodostua vahva ykkösvalinta.

Ionic 2 oli beta-vaiheessa, kun projektia lähdettiin pistämään liikkeelle ja valitsemaan käytettäviä teknologioita. Ionic 1 käytti Angularin ensimmäistä versiota, joista allekirjoittanut oli kuullut hyvin paljon enemmän negatiivista kuin positiivista kommenttia. Sen lisäksi oli annettu ymmärtää, että Angularin kakkosversio, johon Ionic 2 pohjautuu, olisi täysin uudenlainen eikä ollut yhteensopiva aiemman versionsa kanssa, joten päätimme lähteä rakentamaan sovellusta Ionic 2:lla. Tämä tuntui hieman riskiltä, koska Ionic 2:n julkaisuaikataulusta ei ollut minkäänlaista tietoa. Ionicin kehitystiimi

ilmoitti, että julkaisuversio saadaan ulos, kun se on valmis ja ilmenneet ongelmat on saatu korjattua. Ionic 2:n julkaisuaikatauluun vaikutti vahvasti Angular 2:n julkaisu, sillä Ionicia ei julkaistu ennen Angular 2:n julkaisua.

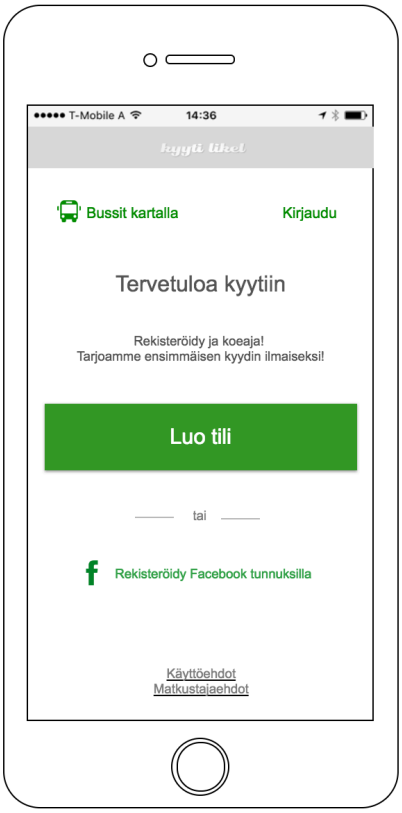
Keskeneräisessä teknologiassa ongelmana oli myös se, että Ioniciin tehtiin suuriakin muutoksia vielä beta-vaiheessa versioiden välillä. Tähän lisättynä melko vaatimaton dokumentaatio pidensi ongelmien ratkaisemisaikoja, mutta päivä päivältä tämä helpottui. Release candidate –version ilmestyttyä muutokset vähenivät huomattavasti ja täten uudet oppaat ja verkkokeskustelut pysyivät huomattavasti pidempään ajantasaisena. Näistä huolimatta jouduin useasti tarkastelemaan Ionicin lähdekoodia ongelman ratkaisemiseksi.

Itse Angular 2:n opettelu oli suhteellisen helppoa ja pääsin sen toimintatapaan nopeasti sisälle. Beta-vaiheessa ongelmia tosiaan oli jonkin verran, eikä ongelmiin tahtonut löytyä millään vastausta. Julkaisuversion julkistuksen jälkeen ongelmiin alkoi löytyä eri keskustelufoorumeilta ratkaisuja hyvinkin kattavasti.

Ionicin versio 2 julkaistiin 25.1.2017, hieman yli 3 kuukautta ennen lanseerausta. Tässä kohtaa voitiin huokaista helpotuksesta ja jatkaa matkaa helpottuneena.

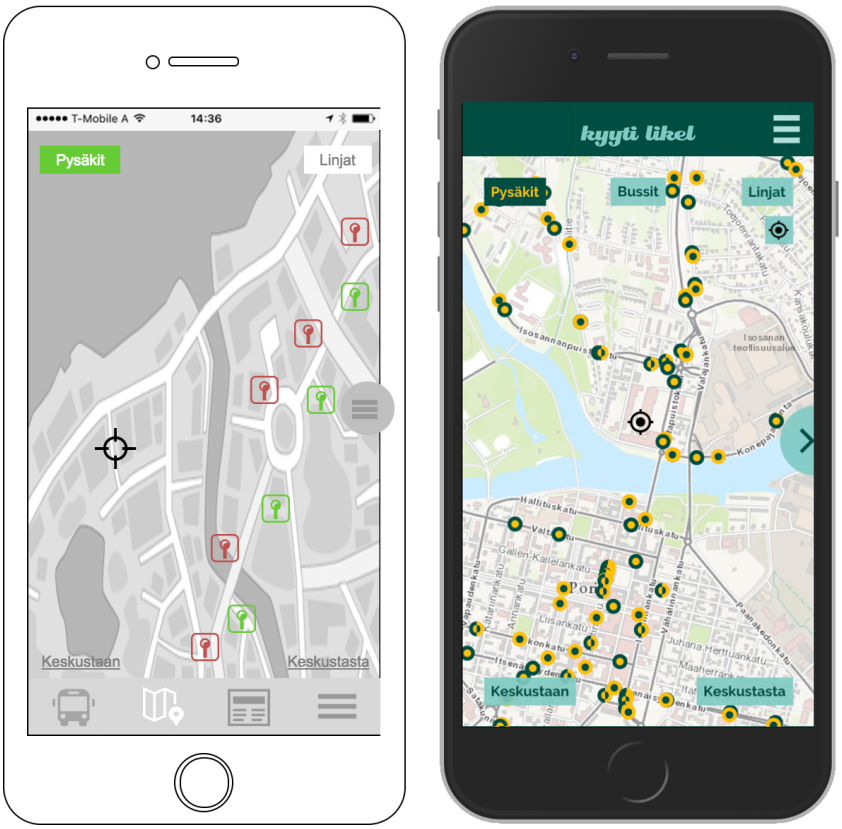
3.2 Kehittämisen ensiaskeleet

Ennen kuin projekti tuli minun pöydälleni, oli sovelluksen suunnitteluun käytetty huomattavan paljon aikaa. Kun sovelluksen konseptointi oli saatu valmiiksi, se toimitettiin minulle ja tehtäväkseni jäi toteuttaa konseptin mukainen sovellus.



Kuva 8. Sovelluksen aloitusnäkymä konseptissa.

Konseptivaiheessa sovelluksen visuaalisella puolella ei merkitystä, vaan siinä pyritään tuomaan esiin kaikki toiminteet, jotka tulevat sovellukseen. Sovelluksen lopullinen ulkoasu saattaa olla hyvinkin erilainen konseptiin verrattuna, joten aluksi keskityttiin vain tekniseen puoleen.



Kuva 9. Konseptin ja kirjoitushetkellä olevan version ero.

Tässä vaiheessa ei ollut käytössä vielä palvelinyhteyttä, joten kaikki data luotiin valmiiksi sovellukseen sisään.

3.3 Sovelluksen ominaisuudet

3.3.1 Sovelluksen käyttäminen

Jotta sovellusta voisi käyttää matkustamiseen, on käyttäjän rekisteröidyttävä palvelun käyttäjäksi. Bussit kartalla –ominaisuutta voi käyttää ilman rekisteröitymistä palveluun.

Sovellus vaatii verkkoyhteyden toimiakseen. Jatkuvaa verkkoyhteyttä ei kuitenkaan tarvita.

3.3.2 Rekisteröityminen käyttäjäksi ja kirjautuminen

Käyttäjän tunnistaminen perustuu puhelinnumeroon, joten perinteinen käyttäjänimen tai sähköpostiosoitteen perusteella tunnistamisen sijasta sovellukseen rekisteröidytään puhelinnumerolla.

Puhelinnumero vahvistetaan lähettämällä tekstiviesti käyttäjän ilmoittamaan puhelinnumeroon. Käyttäjä syöttää tekstiviestissä olevan koodin sovelluksen sitä pyytäessä. Vahvistuksen jälkeen käyttäjä voi matkustaa.

Sovellukseen voi rekisteröityä myös Facebook-tunnuksella. Facebook-rekisteröitymisen etuna on se, ettei käyttäjän tarvitse muistaa ja syöttää salasanaansa kirjautumisensa yhteydessä, vaan kirjautumiseen voidaan käyttää Facebook-tunnusta. Myös rekisteröitymisen yhteydessä Facebookista noudetaan käyttäjän nimi, sähköposti sekä syntymäaika, mikäli käyttäjä valitsee Facebook-rekisteröitymisen.

Sovelluksessa voi kirjoitushetkellä maksaa ainoastaan luottokortilla, jonka vuoksi rekisteröityminen edellyttää luottokorttia.

Porin Linjat tarjoaa palveluun rekisteröityneelle käyttäjälle ilmaisen matkan, minkä vuoksi käyttäjä täytyy todentaa oikeaksi.

3.3.3 10-kortti

Porin Linjat tarjoaa 10-kortteja, jolla on nimensä mukaisesti mahdollista matkustaa 10 kertaa Porin Linjojen bussilla. 10-kortilla voi myös matkustaa useampi henkilö samalla kertaa. Muissa lipputyypeissä täytyy ottaa huomioon matkan pituus sekä mahdollisesti myös kellonaika, sillä nämä vaikuttavat lipun hintaan. 10-kortissa näitä hintavariaatioita ei ole, joten tämä oli sopiva lipputyyppi sovelluksen ensimmäistä versiota ajatellen.

10-kortilla on matkustusaikaa kaksi tuntia, jonka aikana voi käyttää yhden vaihdon. Käytännössä tämä tarkoittaa sitä, että käyttäjällä on viisi minuuttia aikaa nousta bussiin matkan aloittamisen jälkeen ja kahden tunnin aikana käyttäjän on mahdollista käyttää yksi vaihto.

Kuva 10. Kuvasarja 10-kortin matkan aloittamisesta sekä vaihdosta.

Matkan vahvistus (kuva 10, neljäs kuva) lisättiin myöhemmässä vaiheessa palautteen perusteella. Moni oli tahtomattaan aloittanut matkan, minkä vuoksi lisäsimme vielä yhden varmistuksen, että vahvistetaanko matka aloitus. Tämän jälkeen tästä ei ole tullut palautetta.

3.3.4 Perheprofiili

Käyttäjä voi luoda perheprofiilin esimerkiksi lapselleen. Perheprofiili eroaa tavallisesta profiilista niin, että perheprofiililla ei voi syöttää luottokorttitietoja, vaan 10-korttien maksu veloitetaan isäntäkäyttäjään liitetyltä luottokortilta. Perheprofiilille voidaan antaa oikeus ostaa 10-kortti suoraan sovelluksesta. Mikäli oikeutta ei anneta, niin perheprofiilin käyttäjä voi pyytää sovelluksessa uutta 10-korttia, mikä lähettää ilmoituksen isäntäkäyttäjän puhelimeen ja isäntäkäyttäjän puhelin avaa ilmoitusta painettaessa sovellukseen näkymän, jossa voi ostaa 10-kortin perheprofiilille.

Isäntäkäyttäjä voi nähdä perheprofiilin matkustus- ja matkahistorian myös omasta puhelimestaan.

Kuva 11. Perheprofiilinäkymä.

3.3.5 Matkojen vanhentuminen

10-kortti on voimassa enintään 90 päivää ensileimauksen jälkeen. Ensileimaus on se hetki, jolloin 10-kortin ensimmäinen matka käytetään. Jos käyttäjä ostaa esimerkiksi kaksi 10-korttia, niin jälkimmäisen 10-kortin ensimmäinen käyttökerta on vasta silloin, kun kyseisestä kortista käytetään ensimmäinen matka.

Taustajärjestelmästä tulee muistutus 10-kortin vanhentumisesta Push- tai tekstiviestinä.

3.3.6 Muita ominaisuuksia

Sovelluksella voi lukea sovellukseen liittyvät tiedotteet, usein kysytyt kysymykset, matkustus- ja käyttöehdot, yhteystiedot sekä lähettää palautetta. Näitä ominaisuuksia voi käyttää myös ilman kirjautumista.

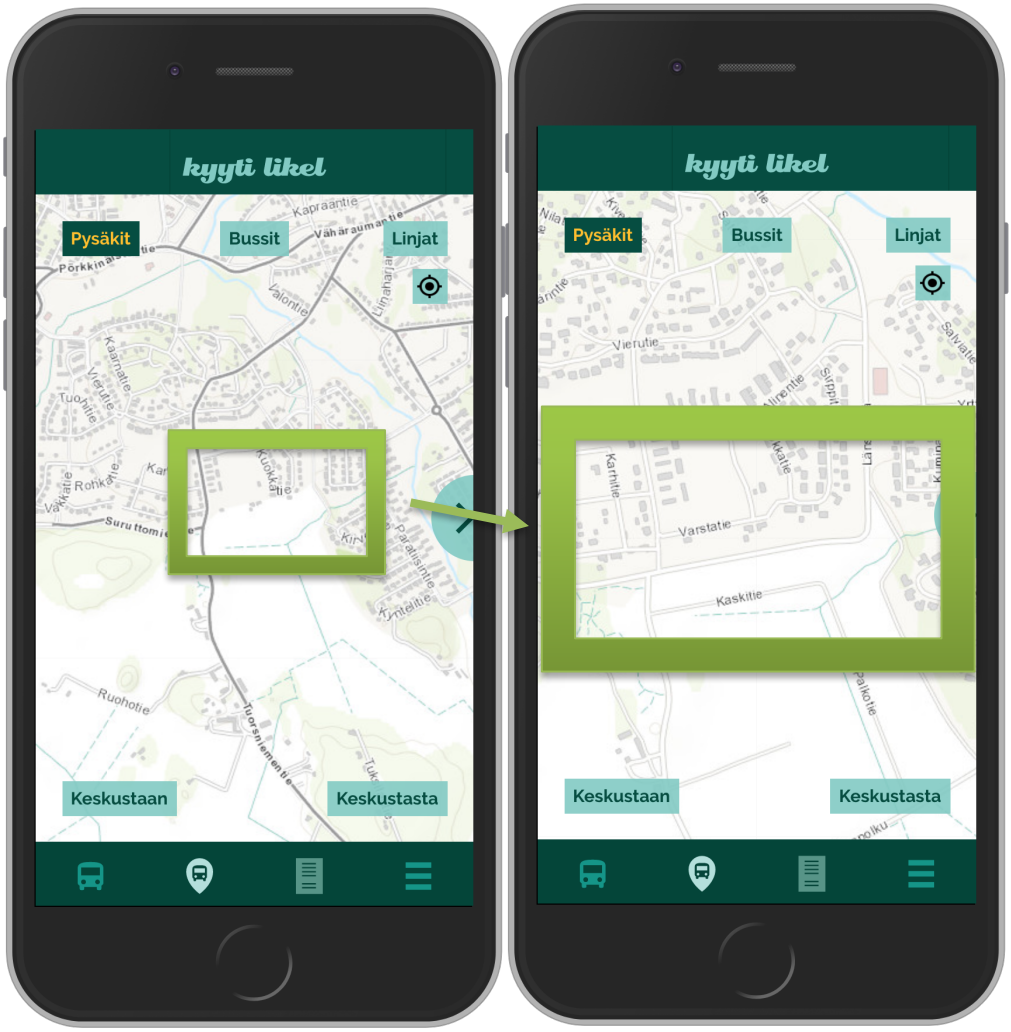
Kirjautuneet käyttäjät voivat vaihtaa salasanansa, linkittää tai poistaa linkityksen Facebook-tilin käyttämisestä kirjautumisessa sekä nähdä matkustus- ja maksuhistoriansa.

3.4 Bussit kartalla

3.4.1 Karttapalvelu

Porin Kaupunki tarjoaa avointa dataa. Avoimen datan jakelukanavaksi on muodostunut Esrin ArcGIS-paikkatietojärjestelmä. Esri tarjoaa myös karttapalvelua, jota sovellus käyttää.

Ongelmaksi Esrin kartoissa on muodostunut se, että karttakuvat ovat vanhoja. Sama karttapohja saattaa näyttää erilaista näkymää riippuen tarkennuksesta, kuten kuvassa 12 on esimerkkinä.



Kuva 12. Esimerkki kartassa olevista eroavaisuuksista riippuen tarkennuksesta. Vihreät laatikot esittävät samaa kohtaa kartalla, mutta eri tarkennustasoilta. Vasemmasta kuvasta puuttuu uudemmat tiet, jotka löytyvät oikeanpuolisesta tarkemmasta kuvasta.

Tällä hetkellä käytössä oleva kartta on ajantasaisin valittavissa olevista karttapohjista. Käyttöön olisi otettu visuaalisesti selkeämpi Streets-niminen karttapohja, mutta se osoittautui vielä vanhemmaksi kartaksi kuin nykyinen karttapohja. Nykyisessä karttapohjassa näkyy muun muassa sellaisia Puuvillatehtaan rakennuksia, jotka purettiin 2014 avatun kauppakeskuksen rakennustöiden yhteydessä.

3.4.2 Tekninen toteutus sovelluksessa

Kartan näyttämiseen ja hallintaan otimme käyttöön Leaflet-kirjaston, sillä kyseinen kirjasto tarjoaa tarvittavat ominaisuudet interaktiiviset kartan luomiseen.

Esri on itse luonut Esri Leaflet –nimisen kirjaston, joka yhdistää Leaflet-kirjaston ja Esrin kartan sekä tarjoaa selkeät rajapinnat ArcGIS-palvelun käyttämiseen. Tämä kirjasto on käytössä sovelluksessa. Tämän rajapinnan käyttö on sovelluksessa kuitenkin melko vähäistä, sillä sovellus käyttää suoraan ArcGIS-palvelua ainoastaan reittiviivojen piirtämiseen. Muut tiedot tulevat sovelluksen taustajärjestelmästä.

3.4.3 Karttanäkymä

Karttanäkymässä, joka on myös Bussit kartalla –osion aloitusnäkymä, näkyvät kaikki pysäkit, jotka ovat jonkin linjan reitillä. Pysäkit on merkitty myös sen mukaan, mihin suuntaan niissä bussit ovat menossa. Pois keskustasta menevä pysäkki tarkoittaa sitä, että pysäkillä pysähtyvät ainoastaan bussit, joiden reitillä kyseinen pysäkki on pois keskustasta menevään suuntaan. Keskustaan menevä pysäkki tarkoittaa päinvastaista. Pysäkiltä voi kulkea bussi sekä keskustaan että pois keskustasta.

Pysäkkejä on järjestelmässä kirjoitushetkellä (13.11.2017) 2020, joista 1367 on merkitty jonkun linjan reitille. Tämä tuo omat haasteensa sovellukseen, sillä 1367 pysäkin näyttäminen liikuteltavassa kartassa on melko raskasta. Pysäkkien määrästä huolimatta sovelluksen käyttö tulisi olla mahdollisimman jouheaa. Pienellä optimoinnilla saatiin parannettua suorituskykyä käytettävälle tasolle.

3.4.4 Bussit

Kartalle saa näkymään kaikki bussit. Bussin ikonissa näkyy ajettava linja. Bussia voi myös painaa, jolloin avautuu laatikko, josta näkee bussin ajaman linjan. Laatikossa on myös painike, josta voi rajata näytettäväksi ainoastaan kyseisen linjan bussit sekä reitin. Bussin sijainti päivittyy kerran kolmessa sekunnissa.

Sijaintitiedon hakemiseen on käytössä kaksi eri vaihtoehtoa. Ensisijainen tapa on avata taustajärjestelmään Websocket-yhteys. Websocket-yhteys on koko ajan auki taustajärjestelmään, joten datapakettien koko on hyvin pieni. Mikäli taustajärjestelmä hakee

bussien sijaintiedot välimuistista, on sijaintipyynnön ja vastauksen vastaanottamisessa jopa alle 20 millisekunnin viive.

Toissijainen tapa on perinteisempi https-pyyntö. Toissijainen tapa aktivoituu, mikäli ensisijainen tapa ei onnistu tai siinä ilmenee häiriöitä.

3.4.5 Reitit ja aikataulut

Käytössä olevat linjat saa auki Linjat-painikkeen takaa. Näkymään avautuu listaus linjoista, joita klikkaamalla voi valita kyseisen linjan. Klikkauksen jälkeen listausnäkymä poistuu ja kartalta häviää kaikki muut pysäkit, jotka eivät ole kyseisellä linjalla. Kartalle piirtyy myös reittiviiva.

Aikataulut saa näkyville klikkaamalla ensin mitä pysäkkiä. Klikkauksesta avautuu laatikko näytön alareunaan, josta näkee edellisen lähteneen bussin sekä seuraavaksi lähtevän bussin aikatauluineen. Laatikossa näkyy myös kyseisellä pysäkillä pysähtyvät linjat sekä painike, josta voi avata näkymän, jossa näkyvät kaikki kyseisen pysäkin aikataulut. Avautuvassa näkymässä näkyy pysäkin linjojen aikataulut. Aikatauluja voi tarkastella linja- ja päiväkohtaisesti. Mikäli bussi on seuraavan viiden minuutin kuluttua saapumassa pysäkille, on suoraan kyseisestä näkymästä mahdollisuus siirtyä kymppikortin käyttämisen näkymään, jossa kyseinen linja on valittuna.

Bussien sijaintitiedon lisäksi busseista tulee tieto, että ovatko ne aikataulussa minuutin tarkkuudella. Mikäli bussi on etuajassa tai myöhässä, muuttuu tulevilla pysäkeillä aikatauluissa kyseisen vuoron aikataulu. Käytännössä tieto aikataulupoikkeamasta saadaan, kun bussissa oleva GPS-laite lähettää tiedon sijainnistaan kolmannen osapuolen palveluun, johon on syötetty myös reitin pysäkkien koordinaatit sekä aikataulut. Palvelu näkee, kun bussi on tarpeeksi lähellä reitin seuraavaa pysäkkiä ja vertaa sen hetkistä aikaa aikataulussa olevaan tietoon ja näin osaa laskea aikataulupoikkeaman.

3.5 Ongelmat

Marraskuussa 2017 myyntiin tullut iPhone X loi pienen ongelman. iPhone X:n näyttö poikkeaa muista puhelimista suuresti. Näyttö ei ole kulmikas, vaan reunat ovat pyöristetyt ja yläreunassa näyttö kiertää kuuloke- ja kamera-kohdat. Syyskuussa 2017 julkaistussa iOS 11 –versio sisälsi tuen iPhone X:n näyttöön, mutta samalla se aiheutti muille iPhoneille ongelman, jossa sovelluksen yläreuna ei näkynyt oikein. (jcesarmobile 2017.)

Kuva 13. iPhone X

Tämän lisäksi myös Android-puhelimien laaja skaala on aiheuttanut paljon päänvaivaa. Sen lisäksi, että puhelimia on eri kokoisia erilaisilla resoluutioilla, myös käyttöjärjestelmän versioita on laaja skaala käytössä. Vanhin tilastoissa näkyvä Androidin käyttöjärjestelmäversio on 4.4.4 ja uusin 8.0.0. Version 4.4.4 viimeisin päivitys on kolmen vuoden takaa. Sovelluksen lanseerauksen jälkeen ilmeni, että sovellus ei toiminut ollenkaan 4.4.4-versiolla, mutta pienellä korjauksella sovellus saatiin toimintakuntoon.

4 TAUSTAJÄRJESTELMÄ

4.1 Sovelluskehyksen valinta

Sovellukselle lähdettiin hakemaan mahdollisimman yksinkertaista, mutta tehokasta sovelluskehystä. Laravel oli tuttu aiemmista projekteistani ja oli osoittautunut hyvinkin tehokkaaksi. Pieni tutkimus mahdollisista sovelluskehyksistä antoi varmuuden siitä, että Laravel on hyvä valinta tämänkaltaisen taustajärjestelmän pohjaksi.

4.2 Maksupalvelu

Maksupalveluksi valitsimme Stripe-nimisen palvelun. Stripe on liitetty taustajärjestelmään niin, että rekisteröityessä Stripeen luodaan uusi asiakas, jolle liitetään rekisteröitymisessä syötetty luottokortti. Luottokortin tiedot toimitetaan suoraan Stripeen, eivätkä niitä tallenneta missään vaiheessa taustajärjestelmään. Stripe palauttaa asiakkaasta ja luottokortista yksilölliset tunnisteet, joiden avulla voidaan laskuttaa asiakasta.

Tunnisteiden avulla luottokorttitiedot voidaan säilyttää palvelussa, joka panostaa erityisen paljon resursseja turvallisuuteen. Palvelun tuottamalla tunnisteella voidaan suorittaa ostoja ainoastaan kyseisen sovelluksen tuotteille, joten mahdolliset vahingot saadaan minimoitua.

4.3 Facebook

Facebook tarjoaa sovelluskehittäjän näkökulmasta helpon rajapinnan käyttäjän tunnistamiseen. Kirjautumiseen voidaan käyttää Facebook-sovellusta tai selainta, minkä sovellus avaa automaattisesti. Kirjautumistietojen syöttämisen jälkeen Facebook pyytää lupaa tietojen luovuttamisesta sovellukselle, jonka hyväksymisen jälkeen palataan takaisin itse sovellukseen.

Facebook-kirjautuminen on toteutettu sovellukseen niin, että kun Facebook palauttaa sovellukselle kirjautumisen jälkeen yksilöidyn tunnisteen, sovellus lähettää tunnisteen eteenpäin taustajärjestelmälle. Rekisteröityessä taustajärjestelmä hakee Facebookista tunnisteen avulla käyttäjän id:n, joka tallennetaan tietokantaan. Kirjautuessa taustajärjestelmä vertaa Facebookista noudettua käyttäjän id:tä tallennettuun tietoon.

Olen käyttänyt Facebookin rajapintoja tiedon hakemiseen useassa projektissa. Tämä oli kuitenkin ensimmäinen projekti, jossa jokaisen käyttäjän täytyi pystyä kirjautumaan sisään. Tämä vaati pientä selvitystä Facebookille sovelluksesta ja mihin tietoja käytetään. Facebookille täytyi myös toimittaa sovelluksen kehitysversio, jotta he pääsivät näkemään, että rajapintoja käytettiin heidän sääntöjen mukaan.

4.4 Push-ilmoitukset

Push-ilmoitukset Android-puhelimiin lähetetään Firebase-palvelun kautta. iOS-puhelimiin ilmoitukset lähetetään Applen Apple Push Notification services (APNs) -palvelun kautta.

Push-ilmoituksien kanssa oli pitkään hyvin paljon ongelmia. Kun sovelluksen Android-versioon saatiin ilmoitukset perille, niin iOS-versiossa ne lakkasivat toimimasta. Ja kun iOS-versiossa saatiin toimimaan, niin Androidissa lakkasi toimimasta. Useiden oppaiden ja dokumentaatioiden jälkeen ilmoitukset saatiin toimimaan. Android-puhelimiin ilmoitukset saadaan toimitettua suoraan Firebasesta http-pyynnöllä, mutta iOS-puhelimiin lähetykseen rakennettiin Node.js-pohjainen skripti, joka ottaa yhteyden Applen palveluun. Kyseinen skripti otettiin käyttöön sen vuoksi, että siihen löytyi valmis ja helppokäyttöinen kirjasto, vaikkakin käytössä on eri koodikieli.

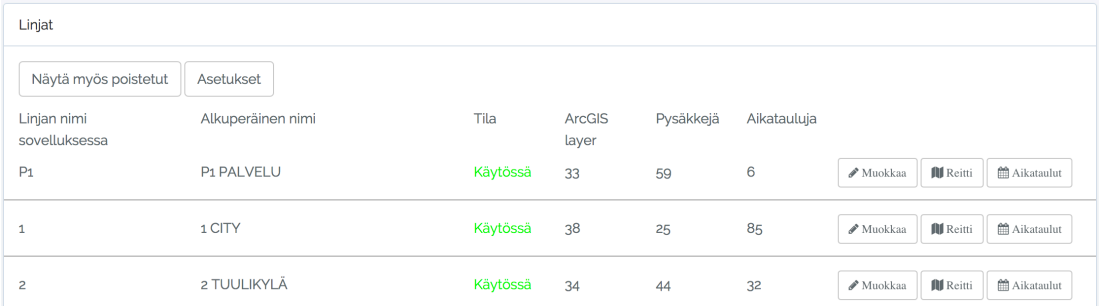
Push-ilmoituksia käytetään sovelluksessa kahdessa käyttötarkoituksessa. Ensimmäinen on, kun perheprofiilin käyttäjä pyytää lisää matkoja 10-korttiin. Toinen on 10-kortin matkojen lähestyvän vanhenemisen tai vanhenemisen ilmoittaminen. Mikäli 10-kortin vanhenemiseen on jäljellä viikko, eikä ilmoituksen toimittaminen onnistu, lähetetään tekstiviesti käyttäjän puhelimeen.

4.5 Bussit kartalla

4.5.1 Aineisto

Porin Linjat tuottavat aineiston yhdessä Porin kaupungin kanssa. Porin kaupungin palvelimelle kerätään ainesto kokoon, josta Kyyti likel –palvelun palvelin noutaa aineiston kerran vuorokaudessa. Bussien sijaintitieto noudetaan tarvittaessa korkeintaan kerran kolmessa sekunnissa.

Kyyti likel –palvelun hallinnassa määritellään käytössä olevat linjat. Linjoille määritellään myös ArcGIS-palvelusta haettavan reittiviivan id-tunniste, joka toimitetaan sovellukseen. Sen lisäksi hallinnassa lisätään ainestoon vielä pysäkkien suunta reitillä. Tästä koostuu myös tieto, että kulkeeko pysäkillä bussi keskustaan ja pois keskustasta, vaiko ainoastaan jompaankumpaan suuntaan.

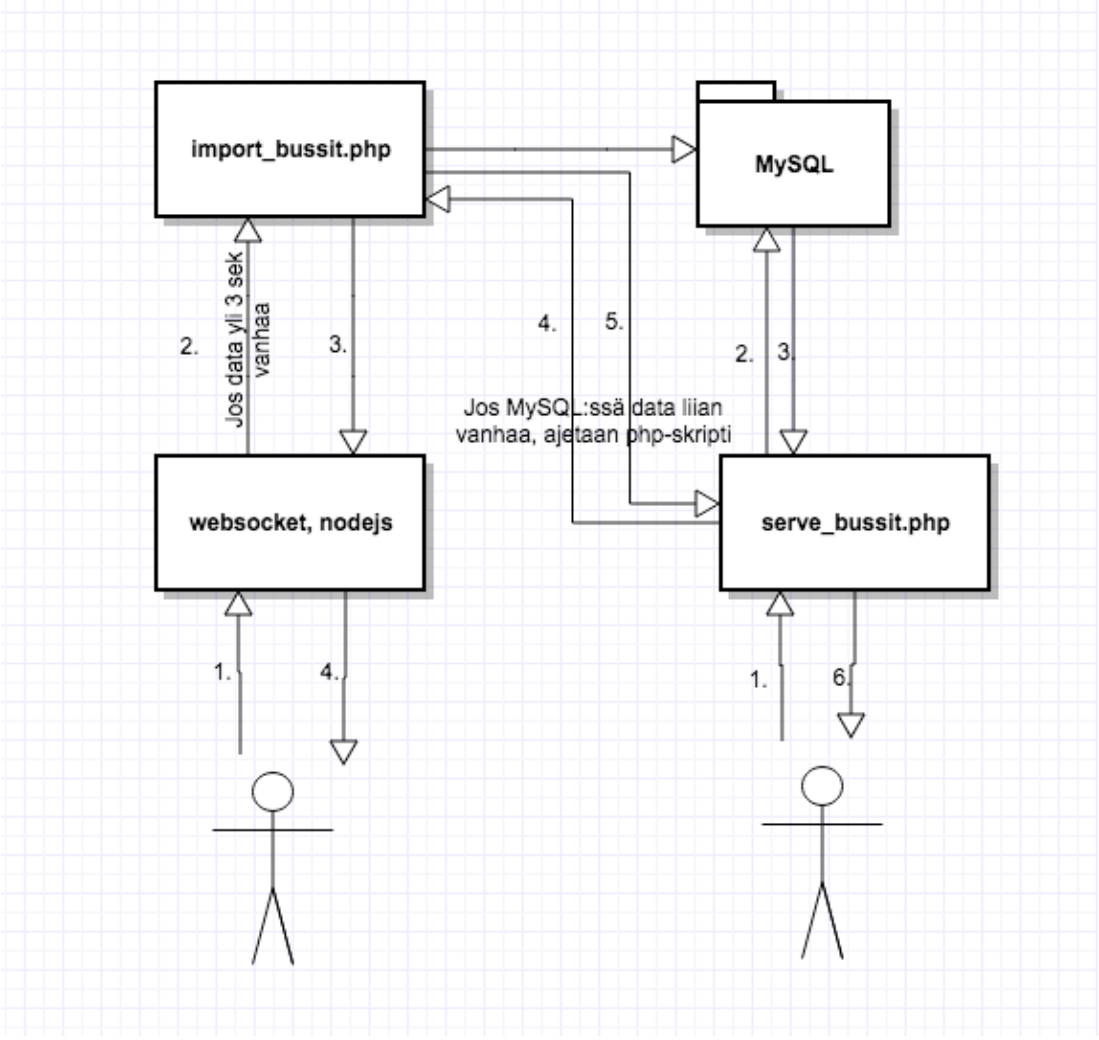


Kuva 14. Linjojen hallintasivu.

Kuva 15. Pysäkkien suunnan määritys reitillä.

4.5.2 Rajapinta sovellusta varten

Taustajärjestelmän rajapinta on toteutettu PHP:lla. Se on toteutettu mahdollisimman kevyeksi ja yksinkertaiseksi. Ensimmäisellä kerralla, kun sovelluksella haetaan tietoa rajapinnasta, niin tiedot koostetaan ja koostettu tieto tallennetaan MySQL-tietokantaan. Tämän jälkeen, kun tietoa haetaan uudelleen, koostettu paketti haetaan tietokannasta ja palautetaan käyttäjälle. Koostettu paketti pysyy tietokannassa siihen asti, kunnes tieto haetaan uudelleen kaupungin palvelimelta tai linjoihin tehdään muutoksia hallintapaneelissa.



Kuva 16. Suunnitelma rajapinnan toteutuksesta.

5 YHTEENVETO

Aluksi projektissa kuljettiin melko hämärällä tiellä kohti päämäärää, kun sovelluksen pohjaksi päätettiin ottaa teknologia, joka ei ollut vielä valmis eikä sen käyttöön ollut kovin suurta asiantuntemusta. Tämä päätös osoittautui lopulta kuitenkin hyväksi ja tulokseen ollaan oltu tyytyväisiä. Työn edetessä asiantuntemusta on karttunut hyvinkin paljon ja teknologian ymmärrys on saatu nostettua riittävälle tasolle. Sovelluksen koodia on tarkasteltu jatkuvasti ja päivitetty vastaamaan uusimman Ionic-version vaatimuksia.

Taustajärjestelmän kanssa tekeminen on ollut tasaista ja varmaa. Tuntemus Laravelista auttoi kehittämään taustajärjestelmän niin, ettei siihen ole tarvinnut tehdä juurikaan muutoksia. Päivitykset on saatu tehtyä kivuttomasti eikä katkoja ole tullut missään vaiheessa.

Tässä työssä on tullut hyvin esille oma tapa oppia, joka omalla kohdalla on itse tekemällä ja ottamalla muista oppia. Uuden teknologian opettelu alkaa katsomalla, että miten muut ovat tehneet asioita ja soveltamalla niitä omaan sovellukseen. Alkuun saattaa tulla paljon hyvinkin sekaista koodia, mutta jonkin ajan kuluttua on opittu uusi tapa, jolla selkeyttää sekainen koodi.

Lopputulokseen täytyy olla tyytyväinen. Hyvästä lopputuloksesta tietää, kun puhelin ei ole juurikaan soinut. Tyytyväisyys saa kuitenkin väistyä, kun kirjoitushetkellä (6.12.2017) ollaan jo tekemässä seuraavaa suurta päivitystä, jossa sovellukseen tuodaan huomattavan paljon uusia toiminnallisuuksia.

LÄHTEET

Saccomani, P. 2017. Native, Web or Hybrid Apps? What's the difference. Viitattu 2.12.2017. https://www.mobiloud.com/blog/native-web-or-hybrid-apps/

Marketing And Growth Hacking. 2017. Viitattu 2.12.2017. https://blog.markgrowth.com/native-vs-web-vs-hybrid-apps-whats-the-difference-1df4c5e4bc50

Lynch, M. 2017. Ionic's 2016 - By the Numbers. Viitattu 20.11.2017. https://medium.com/ionic-and-the-mobile-web/ionics-2016-by-the-numbers-3a8e2c65177b

VanToll, T. 2017. What is Angular? Viitattu 15.11.2017. https://developer.telerik.com/topics/web-development/what-is-angular/

Zen Of Coding. 2017. THE STATE OF PHP MVC FRAMEWORKS IN 2017 (LARAVEL, SYMFONY, CODEIGNITER, CAKEPHP, ZEND). Viitattu 28.11.2017. http://www.zenofcoding.com/2017/02/27/the-state-of-php-mvc-frameworks-in-2017-laravel-symfony-codeigniter-cakephp-zend/

Apache Cordova. 2017. Viitattu 20.11.2017. https://cordova.apache.org/docs/en/latest/guide/overview/)

Barnes, E. 2017. Laravel 5.5 will require PHP 7.0+. Viitattu 22.11.2017. https://laravel-news.com/laravel-5-5-php-7-0

Tutorials Point. 2017. Viitattu 18.11.2017. https://www.tutorialspoint.com/laravel/laravel_overview.htm

Laravel. 2017. Viitattu 15.11.2017. https://laravel.com/docs/5.5/

Rails Guides. 2017. Viitattu 28.11.2017. http://edgeguides.rubyonrails.org/active_record_basics.html

Lekovic, S. n.d. Architecture of Laravel 5.2 Applications. Viitattu 15.11.2017. http://www.sinisalekovic.com/blog/architecture-of-laravel-5-2-applications

jsecarmobile. 2017. CB-13311: (iOS) Statusbar does not overlay correctly on iPhone X. Viitattu 15.11.2017. https://github.com/apache/cordova-plugin-statusbar/pull/87