Toni Lyytikäinen

Verkkokaupasta ostetun lipun luonti, vaihtaminen rannekkeeseen ja liittäminen pelitiliin

Insinööri (AMK)
Tieto- ja viestintätekniikka
Kevät 2019

Tiivistelmä

Tekijä(t): Lyytikäinen Toni

Työn nimi: Verkkokaupasta ostetun lipun luonti, vaihtaminen rannekkeeseen ja liittäminen pelitiliin

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: Laravel, PHP, Shopify, SuperPark

Työ tehtiin SuperPark Oy:n toimeksiannosta. SuperPark on kainuulainen sisäaktiviteettipuistoja rakentava ja ylläpitävä yritys. SuperPark laajensi toimintaansa Aasiaan, ja tästä syystä verkkokauppa päätettiin vaihtaa Shopifyn verkkokauppaan.

Työn tavoitteena oli luoda rajapinta, joka kykenee lähettämään asiakkaalle sähköpostilla tämän verkkokaupasta osama lippu. Tämän lisäksi tavoitteena oli kyetä liittämään lippu RFID-tagilla varustettuun rannekkeeseen, jonka avulla asiakas pääsee puistoon.

Työssä luotiin SuperParkin Laravel-pohjaiselle palvelimelle rajapinnat verkkokaupasta ostetun lipun käsittelyyn, tallentamiseen PostgreSQL-tietokantaan ja lähettämiseen asiakkaalle sähköpostilla. Laravel-palvelimen ohjaimia ohjelmoitiin PHP-koodikielellä. Lisäksi asiakkaalle lähetettävän sivun luomisessa käytettiin Laravelin Blade HTML-mallipohjaa ja JS-skriptikieltä sekä CSS-tyyliskriptikieltä.

Työn seurauksena saatiin toimiva järjestelmä, jolla voidaan luoda lippuja Shopifyn verkkokaupasta ostettujen tuotteiden perusteella. Järjestelmä otettiin käyttöön SuperParkilla ensin Suomen verkkokaupassa syksyllä 2017, ja ulkomaille siirryttyä on tämä järjestelmä ollut osana SuperParkia Hong Kongin puistoa lukuun ottamatta kaikkialla. Aasian vaatimuksiin sopiva verkkokauppajärjestelmä vaatii työtä enemmän, kuin mitä yhden opinnäytetyön aikana ehdittiin tekemään. Tämän työn tuotoksena saatiin hyvä pohja jatkokehitystä varten.

Abstract

Author(s): Lyytikäinen, Toni

Title of the Publication: Ticketing System for SuperPark

Degree Title: e.g. Bachelor of Engineering, Information and communications technology

Keywords: Laravel, PHP, Shopify, SuperPark

This thesis was assigned by SuperPark. SuperPark is a company creating and maintaining indoor activity parks called SuperParks. SuperPark expanded their activity to Asia. A decision was made to transition from SuperPark's old webstore to a Shopify based store.

The goal of this thesis was to create a backend for sending tickets bought from Shopify-based webstore to the customers by email. Additional goal was to connect the ticket with a RFID-tag located in a wristband. This wristband would be then used to access the parks.

A new architecture was created atop of SuperPark´s old Laravel globalserver. This architecture receives the webhook from Shopify when a customer makes an order. It gets the metafields set up to a Shopify product with Metafields Editor. It creates a ticket based on the metafields and saves it to the PostgreSQL database. Then it constructs an email containing a link to the ticket page, where the customer can see the ticket. Tickets can be checked and used in the park by SuperManager, an application created by Super-Park. SuperManager can read and use tickets, read wristbands and create products to wristbands. Super-Manager communicates with the site server, which in turn communicates with the globalserver. The wristbands and products are saved in the postgreSQL database. The laravel controller was programmed with PHP. The ticket page was programmed with Laravel blade template along with JS and CSS languages.

The result of this thesis was a working system that creates tickets based on the product bought from Shopify and sends a link to the created tickets to customers by email. SuperPark started using the created system in the fall of 2017 and this system has been in use by SuperPark in each of its parks excluding Hong Kong. A system fulfilling the requirements set by SuperPark´s Asian partners requires more work than could be done during a single thesis. The result of this thesis was a good base for ongoing develop-ment.

Sisällys

Symboliluettelo

Azure App Service

Azure App Service -palvelu ylläpitää web applikaatioita. Applikaatiot voi ohjelmoida haluamallaan ohjelmointikielellä ja Azure hoitaa kuukausimaksua vastaan palvelimen ylläpidon. (1)

Globaalipalvelin

SuperParkin Laravel-pohjainen palvelin. Palvelin on yhteydessä PostgreSQL tietokantaan, johon on tallennettu tieto kaikkien puistojen käyttäjistä, lipuista ja rannekkeista.

QR-koodi (Quick Response Code)

QR-koodi on kaksiulotteinen viivakoodi, jolla on yksiulotteista viivakoodia suurempi tiedontallennuskapasiteeetti ja korkeampi virheen sietokyky (2).

Ranneke

RFID-tagilla varustettu ranneke. Tagi pitää sisällään sarjanumeron, joka tallennetaan tietokantaan tunnistamista varten. Rannekkeita käytetään SuperParkin puistojen kulunvalvontaan.

REST (Representational state transfer)

RESTit ovat asiakkaan ja palvelimen välisiä tilattomia kutsuja, tarkoittaen, että kutsussa itsessään on mukana kaikki tarvittava tieto. (3)

RFID (Radio Frequency Identification)

RFID on radioaaltoja käyttävä teknologia, jolla voi tunnistaa ihmisiä tai asioita. Yleisin tapa tunnistaa asia RFID:llä on tallentaa sarjanumero mikrosirulle, johon on liitetty antenni tiedon välitystä varten. Tällaista mikrosirua kutsutaan RFID-tagiksi. SuperParkilla RFID-tageja käytetään rannekkeiden tunnistamiseen. (4)

Saittipalvelin

Puistojen lähiverkossa toimivia pienempiä palvelimia. Nämä keskustelevat puiston porttien ja aktiviteettien sekä globaalipalvelimen kanssa.

SuperManager

SuperManager on SuperParkin kehittämä android-sovellus, jolla voidaan hallita asiakkaiden lippuja ja rannekkeita puiston vastaanotossa.

Webhook

Webhookit ovat automatisoituja kutsuja, jotka lähetetään tietyn tapahtuman seurauksena. Työssä käytetään webhookeja verkkokaupan tilausten yhteydessä. Verkkokauppa lähettää webhookin, joka sisältää tilauksen tiedot, globaalipalvelimelle, kun asiakas maksaa tilauksen verkkokaupasta. Tämän seurauksena palvelin tekee asiakkaalle tilausta vastaavan lipun ja lähettää sen asiakkaalle sähköpostilla. (5)

1 Johdanto

SuperPark on kainuulainen sisäaktiviteettipuistoja kehittävä yritys. Sisäaktiviteettipuistojen tarkoituksena on kannustaa omaehtoiseen liikunnalliseen hauskanpitoon eli sparkkaukseen, johon voivat osallistua kaikenikäiset ja -kuntoiset. (6)

SuperPark on laajentamassa toimintaansa Aasian markkinoille. Tämä vaatii nykyisten IT-palveluiden kehittämistä. Tämän seurauksena SuperPark on vaihtamassa verkkokauppansa uuteen Shopify-pohjaiseen verkkokauppaan. Tällä hetkellä SuperParkilla on käytössään Laravel-pohjainen palvelinrajapinta. Puistojen kulunvalvontaan käytetään kertakäyttöisiä rannekkeita. Puistojen asiakkaat voivat luoda pelitilin SuperParkin webapplikaatiossa (osoitteessa app.superpark.fi), johon heidän aktiviteeteissa saamansa tulokset ja videot tallennetaan.

Työn tavoitteena on kehittää palvelinrajapinta, joka voi keskustella uuden Shopify-pohjaisen verkkokaupan kanssa. Rajapinnalla luodaan Shopifyn verkkokaupasta ostetun lipun tietojen mukaan lippu, joka tallennetaan PostgreSQL-tietokantaan ja lähetetään asiakkalle sähköpostilla.

Työn aikana perehdytään Shopifyn rajapintoihin, kuten webhookin lähetykseen ja yksityisiin ohjelmiin, PostgreSQL -tietokantajärjestelmään ja SQL komentokieleen, Laravel palvelinrajapintaan, PHP-ohjelmointikieleen ja HTML:ään.

Työn tilaajana toimi SuperPark Oy. Työn aikana olin osana SuperParkin Digitiimiä. Digitiimin tarkoitus on kehittää ja ylläpitää SuperParkien digitalisointia. Työn ohjaajana SuperParkin puolelta oli työn alkupuolella Tuomas Laatikainen ja myöhemmin Jarkko Piirainen.

2 SuperParkin esittely

SuperPark on kainuulainen sisäaktiviteetteja rakentava ja ylläpitävä yritys. SuperPark pyrkii kannustamaan ihmisiä viihteellisen liikunnan harrastamiseen. Tällä tarkoitetaan sisäiseen motivaatioon perustuvalla ilolla ja leikkimielisyydellä harrastettavaa liikuntaa. SuperParkin tarjoamat puistot ovat kiinteistöjä, joihin on koostettu useita erilaisia liikunnallisia ja pelillisiä aktiviteetteja. Puistot on rakennettu sillä ajatuksella, että lapsi voi yhdessä leikkiä vanhempiensa kanssa. Tämän takia SuperParkin ikäjakauma on laaja.

Ensimmäinen SuperPark avattiin Vuokatissa vuonna 2012. Tämän jälkeen on rakennettu 13 puistoa Suomeen ja tähän mennessä puistoja on rakennettu myös ulkomaille Hongkongiin, Uppsalaan, Suzhouhun, Singaporeen ja Kuala Lumpuriin. (6)

2.1 Nykytilanne

SuperPark on aiemmin ulkoistanut lähes kaikki IT-palvelunsa. Nykyinen verkkokauppa on CoreGolta tilattu. Aiemmin käytössä ollut IDControlin kehittämä kulunvalvonta on korvattu Satavisionilta tilatulla Experience-platformilla, joka mahdollistaa kulunvalvonnan, pelitilien luomisen ja rannekkeen liittämisen pelitiliin. Kulunvalvontaa käytetään asiakasvirran hallintaan, sekä varmistetaan, ettei puiston maksimikapasiteetti pääse ylittymään ja paloturvallisuusnormit täyttyvät. Kaikissa Suomen puistoissa on ollut kassajärjestelmänä Suomen Kassajärjestelmät. Nykyinen verkkokauppapalvelu ei tue ulkomaille laajenemista, joten tähän on tarvittu uusi globaali verkkokauppajärjestelmä. SuperParkin IT-kehityksestä vastaava Digijohtaja Tuomas Laatikainen on löytänyt mahdolliseksi ratkaisuksi Shopifyn.

SuperParkilla on käytössään Laravel-pohjainen palvelinrajapinta. Käytössä on yksi Azuren pilvipalvelimella sijaitseva globaalipalvelin, joka hallitsee tietokantaa. Lisäksi osilla puistoista on käytössään saittipalvelin, joka toimii puiston lähiverkossa. Saittipalvelimet keskustelevat globaalipalvelimen, puiston aktiviteettien, sekä porttipalvelimen kanssa. Asiakas voi ostaa verkkokaupasta lippuja. Nämä liput tallennetaan CoreGon tietokantaan.

2.2 Asiakkaan kokemus lipun ostosta puistokäyntiin

Asiakkaat voivat ostaa CoreGon verkkokaupasta lippuja, joita vastaan hän saa puiston kassalta rannekkeen puistoon pääsyä varten. Asiakkaat voivat luoda palvelimelle pelitilin sekä liittää käytössään olevia rannekkeita pelitililleen. Tälle pelitilille tulevat puistoista saadut pisteet ja muistot näkyviin. Puistossa lippu luetaan CoreGon järjestelmällä, ja asiakas saa rannekkeen, joka tallennetaan saittipalvelimen tietokantaan. Tallennuksen yhteydessä saittipalvelin lähettää globaalipalvelimelle tiedon uudesta rannekkeesta. Tämän seurauksena globaalipalvelin tallentaa omaan tietokantaansa kyseisen rannekkeen.

Asiakas voi lukea saamansa rannekkeen puiston portilla. Luettuaan rannekkeen portti kysyy porttipalvelimelta, voidaanko kyseinen ranneke päästää puistoon. Porttipalvelin lähettää saittipalvelimelle kutsun, joka sisältää rannekkeen ID:n ja portin ID:n. Tällä kutsulla saittipalvelin tarkastaa tietokannasta, onko kyseinen ranneke aktivoitu ja voimassa oleva. Mikäli ranneketta ei ole aktivoitu, suoritetaan aktivointi tässä vaiheessa. Aktivoimalla ranneke asetetaan sille voimassaoloaika rannekkeen keston mukaan. Seuraavaksi saittipalvelin lähettää Globaalille palvelimelle viestin rannekkeen aktivoinnista, jolloin aktivointi tallennetaan myös globaaliin tietokantaan. Tämän jälkeen saittipalvelin antaa porttipalvelimelle hyväksyvän vastauksen ja asiakas pääsee puistoon.

Puiston sisällä asiakas voi liikkua ja leikkiä mielensä mukaan. Joissain aktiviteeteissa on ”Loginpoint”, jonka tarkoituksena on aktivoida kyseinen aktiviteetti. Loginpointeissa on NFC-lukija, jota voidaan käyttää rannekkeella. Pelin aktivoiduttua laite varataan pelin ajaksi, jolloin muut pelaajat eivät pääse aktivoimaan peliä. Pelin jälkeen saittipalvelimelle lähetetään aktiviteetin pisteet, jotka saittipalvelin tallentaa tietokantaan sekä lähettää globaalipalvelimelle. Puistokäynnin jälkeen asiakas voi rekisteröityä tai kirjautua app.superpark.fi-sivustolle. Täällä hän voi liittää käyttämänsä rannekkeen omalle pelitililleen. Rannekkeen liittämisen yhteydessä kaikki puistossa kerrytetyt pisteet liitetään asiakkaan pelitilille.

2.3 Tavoitteellinen lopputilanne

Asiakas voi ostaa uudesta verkkokaupasta lipun, verkkokauppa lähettää SuperParkin Laravel pohjaiselle palvelimelle webhookin, ja tätä webhookia hyväksikäyttäen palvelin luo tietokantaan lipun. Palvelin lähettää linkin lippusivuun sähköpostilla asiakkaalle. Asiakas tulee lipun kanssa

puistoon, jossa lipun QR-koodi luetaan ja asiakas saa rannekkeen. Tällä rannekkeella asiakas voi kulkea porteista ja aktivoida puistojen pelejä ja aktiviteetteja. Aktiviteettien pisteet siirtyvät asiakkaan vuoron jälkeen automaattisesti hänen pelitiliinsä ja hän voi kirjautumalla palvelimen applikaatioon käydä tarkastelemassa näitä pisteitä ja lähettää ja jakaa kuvia sekä muistoja sosiaalisessa mediassa tai ystävien kesken.

3 Kolmansien osapuolten ohjelmistot

Tässä kappaleessa esitellään työssä käytettävät kolmannen osapuolten kehittämät ohjelmistot. Kolmannen osapuolen ohjelmistolla tarkoitetaan jonkun muun kuin SuperParkin aiemmin kehittämää ohjelmistoa. Tämän työn osalta merkittävimpiä ohjelmistoja on Laravel-palvelinympäristö, PHP-ohjelmointikieli, PostgreSQL-tietokantaohjelmisto, JavaScript Object Notation -tiedonvälitysformaatti sekä Shopify-verkkokauppapalvelu.

3.1 Laravel

Laravel on Taylor Otwellin kehittämä php verkkosivujen kehitysrajapinta. Rajapinnan ensimmäinen versio julkaistiin vuonna 2011. Laravelilla on kolmannen osapuolten paketeille tuki, jolla voi lisätä ominaisuuksia palvelimelle. Pakettien asennus onnistuu Composer- ja NPM-pakettienhallintasovellusten avulla. Paketit asennetaan oletuksena vendor-kansioon. Laravelilla on Vagrant-pohjainen Ubuntu-virtuaalikehitysympäristö, Homestead- sekä PHP-ohjelmointikaava, Blade. (7)

3.2 Composer

Composer on PHP:lle kehitetty riippuvuuksienhallintatyökalu. Tällä voidaan hallita projektien tarvitsemia kolmannen osapuolen kirjastoja. Composer vaatii toimiakseen vähintään PHP version 5.3.2. Composer on saatavilla Windowsille, Linuxille ja macOS:lle. (8) Laravel käyttää Composeria hallitsemaan projektin riippuvuuksia (7).

3.3 Artisan

Artisan on Laravelin komentokehotetyökalu. Laravelin mukana tulee useita komentoja projektin rakennuksessa. Nämä komennot ovat nähtävillä komennolla php artisan list. Lisäksi artisan komentoja voidaan luoda käyttäjälähtöisesti. (9).

3.4 Node

Node.js on tapahtuma-ajoinen javascript ajo-ohjelma. Node on skaalautuva, sillä sen ajo suoritetaan ainoastaan tapahtumien tullessa. Mikäli tapahtumia ei ole, Node siirtyy lepotilaan. Mitään ajon estävää kutsusilmukkaa ei ole. (10)

3.5 Node Packet Manager (npm)

npm (Node packet manager) on maailman suurin ohjelmistorekisteri. Se koostuu kolmesta osasta: verkkosivusta, komentorivi-käyttöliittymästä (CLI) ja itse rekisteristä. (11) Npm:n CLI:tä käytetään projektissa javascript-pakettienhallintaan, se huolehtii Noden moduulien säilömisestä ja riippuvuuksien hallinnoinnista.

3.6 Shopify

Shopify on kanadalainen verkkokauppapalvelu. Shopifyllä voi rakentaa verkkokaupan omien tarpeiden mukaisesti. Shopifyn avainominaisuuksiin kuuluu app store, josta voi löytää käyttäjien tekemiä ilmaisia tai maksullisia ohjelmia, joiden avulla voi laajentaa verkkokaupan ominaisuuksia omien tarpeiden mukaisesti. (12) Tässä projektissa käytettäviä applikaatioita on esimerkiksi ”metafields editor”, joka mahdollistaa myytävien tuotteiden metatietojen muokkauksen palvelimen tarpeiden mukaisesti.

Shopify app storesta löytyvien julkisten ohjelmien lisäksi sinne voi tehdä kauppakohtaisia yksityisiä ohjelmia.

3.7 Ubuntu LTS

Ubuntu on Linux-kernelin päälle rakennettu käyttöjärjestelmä. Ubuntun Long term support -julkaisu tapahtuu joka toinen vuosi huhtikuussa, ja niillä on viiden vuoden tuki huoltopäivityksille. (13) Projektissa käytetty Ubuntun versio 16.04 on palvelinten käyttöön erikoistettu käyttöjärjestelmä, jolla on pitkän ajan tuki luvattu vuodelle 2021 saakka.

3.8 Vagrant

Vagrant on ubuntu-pohjainen paikallinen virtuaalikehitysympäristö, jota käytetään projektin kehityksessä virtuaalikoneen pystytyksessä. Yhdellä ”vagrantfile”-tiedostolla voidaan varmistaa, että samanlainen kehitysympäristö on käytössä kaikilla projektin jäsenillä. (14)

3.9 PHP 7.2

PHP on HTML kehitykseen erikoistunut ohjelmointikieli. PHP-koodin sulauttaminen HTML kieleen onnistuu PHP:n avaus- ja sulkemismerkinnöillä ”<?php” ja ”?>”. Yleisin PHP-koodin käyttötarkoitus on palvelinohjelmoinnissa. Toimiakseen tämä tarvitsee PHP-kääntäjän, web-palvelimen sekä web-selaimen. Projektissa käytetään PHP-versiota 7.2 (15)

3.10 JavaScript Object Notation (JSON)

JavaScript Object Notation on standardoitu tiedonvälitysformaatti ECMA-404, ISO/IEC 21778. JSON koostuu kahdesta rakenteesta. Ensimmäinen on olio, eli järjestelemätön sarja nimi ja arvopareista. Toinen on taulukko, eli järjestetty lista arvoja. (16)

3.11 PostgreSQL 9.5

PostgreSQL on avoimen lähdekoodin tietokantajärjestelmä, joka käyttää SQL-kieltä tietojen tallentamiseen (17).

3.12 Metafields editor

Metafields editor on Webify Technologyn kehittämä julkinen applikaatio Shopifylle. Sen avulla voidaan muokata Shopifyn tuotteiden metadataa (18). Tätä käytetään projektissa antamaan Shopifyn verkkokaupasta ostettaville tuotteille palvelimen tarvitsemaa tuotekohtaista tietoa.

3.13 Supervisor

Supervisor on prosessinhallintatyökalu UNIX-pohjaisille käyttöjärjestelmille. Supervisor aloittaa hallittavat ohjelmat itsensä aliohjelmina, ja tällä tavalla sillä on tarkka tieto ohjelmien tilasta. (19) Tätä käytetään suurien ja kauan kestävien prosessien (töiden) hallintaan.

3.14 Laravel-työjono

Laravel mahdollistaa aikaa vievien tehtävien siirtämisen automatisoituun työjonoon. Työt käsitellään niille määritellyn työntekijä-aliohjelman avulla. Työntekijän suorittaminen voidaan siirtää tausta-ajoksi erillisen prosessinhallintaohjelman, kuten Supervisor, avulla. (20)

4 Toteutussuunnitelma

Työ toteutetaan vaiheittain. Ensin tehdään verkkokaupalle rajapinta, jolla asiakkaan ostamasta tuotteesta tehdään tietokantaan merkintä. Tämän jälkeen tehdään rajapinta pdf-tiedoston luomiseen. Lopuksi tehdään sähköpostin lähetykseen asiakkaalle rajapinta, johon liitetään pdf-tiedosto.

4.1 Shopifyn tuotteen rakenne

Verkkokaupassa myytävällä tuotteella voi olla useita vaihtoehtoja eli variantteja. Esimerkiksi päivälippuja voidaan myydä useaan puistoon. Sen sijaan, että jokaisen puiston päivälippu olisi oma erillinen tuotteensa, voidaan nämä asettaa yhden päivälipputuotteen eri variantiksi. Yhdellä tuotteella voi olla kolmea eri optiota ja sata eri varianttia. (21)

4.2 Verkkokaupasta ostetun lipun tallentaminen kantaan

Jotta lippu saadaan lähetettyä asiakkaalle mahdollisimman nopeasti tilauksen jälkeen, tulee palvelimen saada tieto ostosta mahdollisimman pian. Tätä varten verkkokaupassa on mahdollisuus lähettää webhookeja määriteltyjen tapahtumien seurauksena. Esimerkiksi kun asiakas maksaa tilauksen, Shopify lähettää määritettyyn osoitteeseen json-taulukon, jossa on tiedot tilauksesta. Tätä json-taulukkoa hyväksikäyttäen voidaan luoda tilaukseen sopivat liput tietokantaan.

4.3 Webhookin rakenne

Webhookit ovat automatisoituja HTTP-kutsuja. Kutsut lähetetään kolmella ylätunnisteella: X-Shopify-Topic-ylätunniste kertoo tapahtuman, jonka seurauksena webhook on lähetetty. X-Shopify-Hmac-Sha256-ylätunniste luodaan applikaation jaetulla salausavaimella ja webhookin sisältämällä asiatiedolla. Tätä ylätunnistetta käytetään todentamaan kutsun oikeellisuus. X-Shopify-Shop-Domain kertoo verkkokaupan, josta kutsu on lähetetty. (5)

4.4 Metadatan asetus

Shopifyn kolmannen osapuolen valmistamalla lisäosalla ”metafields editor” voidaan lisätä tuotteille metadataa. Sitä hyväksikäyttäen voidaan määritellä tulostettavien lippujen määrä, lipputyyppi, puisto, johon lippu käy, lipulla saatavan rannekkeen voimassaoloaika minuutteina ja sähköpostiin tai luotavaan lippuun tulevat lisähuomiot. Jokaisen tuotteen metadata voidaan hakea Shopifystä kutsulla. *{Shopify_api_url} / products / {product_id} / variants / {variant_id} / metafields.json.* jossa Shopify_api_url on verkkokaupan osoite, joka sisältää käyttäjätunnuksen ja salasanan. Product_id on ostetun tuotteen ID ja variant_id on ostetun tuotteen vaihtoehdon ID.

4.5 Lipun luonti

Lippu luodaan PostgreSQL-tietokantaan. Shopifyn webhookista saadaan tilauksen ID, tilattu tuote ja asiakkaan tiedot. Metadatat hakemalla saadaan tuotekohtaiset tiedot, joilla palvelin osaa tehdä oikeanlaisen lipun. Lippukantaan tallennettavat sarakkeet ovat:

Tuotteen nimi ja tyyppi, asiakkaan nimi ja sähköposti, lipun uniikki ja arvaamattomissa oleva koodi, onko lippu käytetty, missä ja millä laitteella lippu on käytetty, tuotteelle käyvän puiston id, verkkokaupan antamat tilauksen ja tilaustuotteen id:t ja lippuun tulevat erityishuomiot.

4.6 Lippusivun luominen

Haluttiin, että lipun ostaja voi jakaa ostamansa lipun helposti toisen henkilön kanssa. Tämä onnistuu käyttämällä selaimella avattavaa lippulinkkiä. Lipun luonnin jälkeen palvelin lähettää asiakkaalle sähköpostin, joka sisältää linkin palvelimella olevaan ”lippusivuun”, johon generoidaan lipun tiedot. Lippusivun osoitteessa on mukana lipun uniikki ja arvaamattomissa oleva koodi. Tällä saadaan haettua sivun avautuessa lipun tiedot, jolloin jokaiselle lipulle ei tarvita erillistä sivua.

4.7 Sähköpostipohjan luominen

Asiakkaalle lähetettävälle sähköpostille luodaan pohja palvelimelle. Pohja luodaan blade tiedostona. Pohjalle voidaan lähettää lipun tiedot taulukkona, jolloin sähköposti voidaan yksilöllistää asiakkaan tilauksen mukaan.

4.8 Lipun lähettäminen asiakkaalle sähköpostilla.

Laravelissa on rajapinta sähköpostin lähettämiseen, Swiftmail. Tämän sekä sähköpostipalvelimen, kuten sendgrid, avulla voidaan lähettää sähköposti haluttuun osoitteeseen. Laravel suosittelee käyttämään API-pohjaisia ajureita sähköpostien lähettämiseen SMTP-palvelimen sijaan ajureiden nopeuden ja yksinkertaisuuden vuoksi. Nämä ajurit vaativat toimiakseen Guzzle HTTP-kirjaston. (22)

4.9 Lipun vaihtaminen rannekkeeseen

Tämän opinnäytetyön rinnalla SuperPark on kehittänyt applikaation puistojen käyttöön nimeltään SuperManager. Tätä applikaatiota käytetään puistojen kassalla lippujen ja rannekkeiden hallintaan. Lippujen vaihto rannekkeeseen tehdään jatkossa SuperManager-ohjelmalla. Opinnäytetyössäni teen tähän ohjelmaan rajapinnat globaalipalvelimen päähän. Globaalipalvelimen tulee pystyä hakemaan lipun tiedot QR-koodin perusteella sekä liittämään tiedot rannekkeeseen. SuperManager lähettää rannekkeen RFID-koodin, jolloin palvelin hakee tiedot kyseisellä rannekkeella olevista tuotteista ja lisää lipun tiedot uudeksi tuotteeksi. Mikäli ranneketta ei löydy ennestään, se luodaan tässä vaiheessa.

4.10 Rannekkeen liittäminen pelitiliin

Asiakas voi luoda itselleen pelitilin osoitteessa app.superpark.fi. Jokainen ranneke on liitetty pelitiliin. Palvelin luo väliaikaisen pelitilin, mikäli ranneketta luodessa ei anneta pelitiliä. Asiakkaat voivat liittää tämän rannekkeen omaan pelitiliinsä kirjautumalla omalle pelitililleen. Työssä

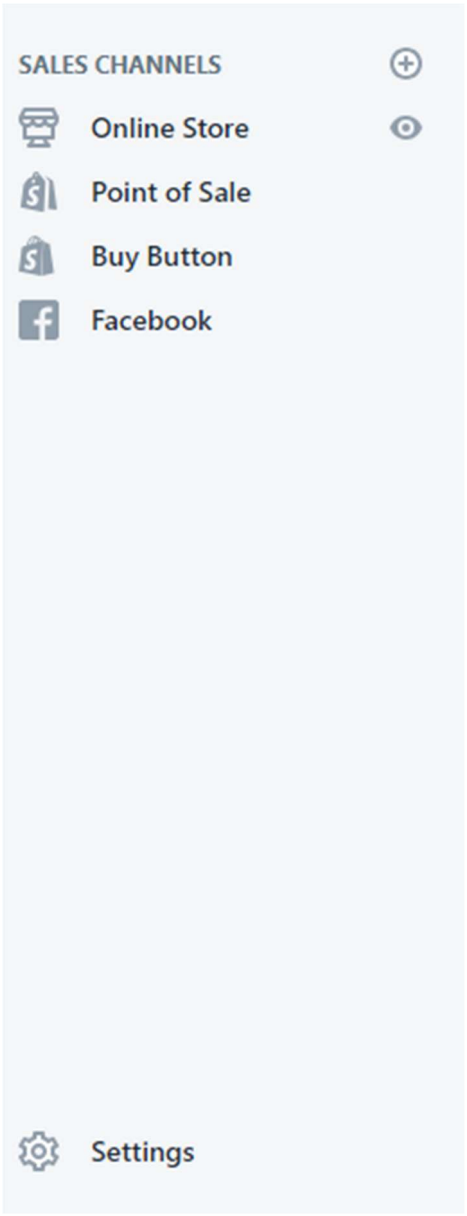
tullaan luomaan SuperManagerille rajapinta, joka mahdollistaa rannekkeen liittämisen pelitiliin ranneketta luodessa.

5 Toteutus

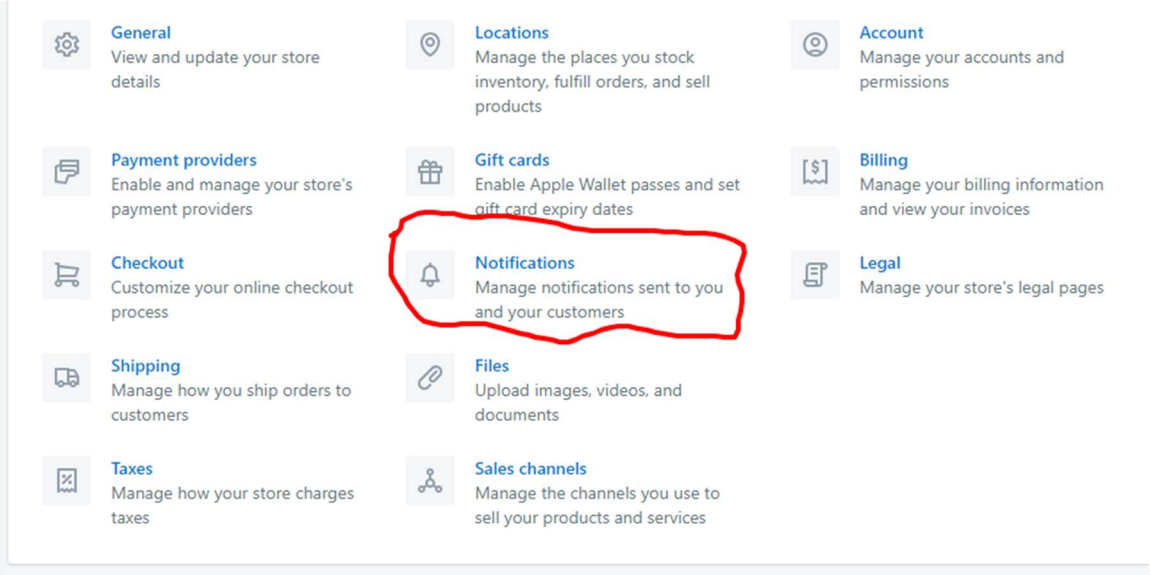
Tässä kappaleessa käydään läpi työn toteutukseen liittyvät vaiheet, kuten kehitysympäristön pystytys, verkkokaupan pystytys ja palvelinrajapintojen luominen.

5.1 Shopifyn webhookin luonti

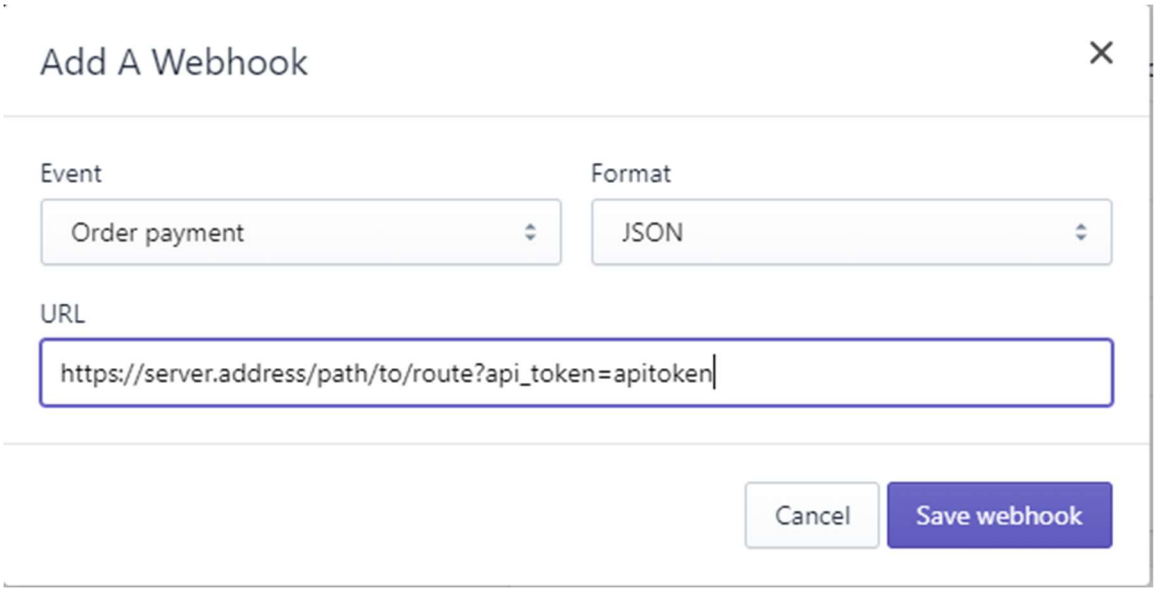
Shopifyn käyttämät webhookit asetetaan Shopifyn admin paneelista. Valitsemalla sivun vasemmasta alalaidasta ”Settings” päästään asetukset-sivulle. Esiteltynä kuvassa 1. Webhookien asetukset löytyvät ”Notifications”-valikon alta, joka on esitelty kuvassa 2. Webhookin luontilomake aukeaa painamalla ”Create Webhook” -painiketta, esiteltynä kuvassa 3. Lomakkeen ”Event”-kenttään valitaan vetovalikosta ”Order payment”, format-kenttään valitaan ”JSON” ja URL-kenttään kirjoitetaan osoite, johon webhook lähetetään. Tämän jälkeen webhook voidaan tallentaa painamalla ”Save webhook” -nappia. Webhookin luontilomake on esitelty kuvassa 4.



Kuva 1. Shopify admin paneelin settings-valikko.



Kuva 2. Notifications valikko.



Kuva 3. Webhookin luonti.

5.2 Yksityisen ohjelman luonti

Globaalipalvelin tarvitsee pääsyn Shopifyn tilauksen, tuotteiden ja toteutusten (fulfillments) tietoihin voidakseen hakea tilauksen tuotekohtaiset metatiedot ja merkatakseen tilaus toteutetuksi. Tätä varten luodaan yksityinen ohjelma. Yksityisen ohjelman luonti tapahtuu Shopifyn admin-paneelista. Valitaan vasemmasta laidasta ”Apps”-välilehti, kuvassa 5. Apps-välilehdellä voi hallita kolmannen osapuolen valmistamia applikaatioita. Tältä sivulta pääsee

myös Shopifyn app storeen, josta voi etsiä ja löytää tarpeisiinsa sopivia ohjelmia. Lisäksi tältä sivulta pääsee yksityisten ohjelmien hallintaan painamalla sivun alalaidasta linkkiä ”Manage private apps”. Esiteltynä kuvassa 6. Aukeavalta sivulta painamalla sivun oikeasta ylälaidasta painiketta ”Create a new private app” avautuu sivu, jolla voi luoda yksityisiä ohjelmia. Tämä painike on esitelty kuvassa 7.

Avautuvalle sivulle syötetään ohjelman tiedot. App details-osioon tulee applikaation nimi ja kehittäjän sähköpostiosoite. Api permissions-osioon tulevat oikeudet, joita applikaatiolla on. Tässä tapauksessa tarvittavia oikeuksia ovat read_products, write_products, read_orders ja write_orders. Ohjelman tallentamisen jälkeen sivu näyttää ohjelman tietojen lisäksi ohjelman api-avaimen, salasanan ja jaetun salausavaimen. Näiden tietojen tallentaminen tietokantaan esitellään seuraavassa kappaleessa.

5.3 Shopifyn tietojen tallentaminen tietokantaan

Jotta tilauksen webhook voidaan yhdistää oikeaan verkkokauppaan, tarvitaan verkkokaupan tiedot palvelimelle. Tätä varten luodaan tietokantaan uusi shopifies-taulukko. Taulukon sarakkeina ovat shop domain, app secret, api_url, locale ja location_id. Shop domain on verkkokaupan osoite, tämä löytyy webhookien *HTTP_X_SHOPIFY_SHOP_DOMAIN*-otsikosta. App secret on verkkokaupan uniikki salausavain, jolla saadaan varmistettua webhookin oikeellisuus. Api_url on verkkokaupan osoite, joka pitää sisällään käytetyn yksityisen ohjelman api_keyn ja salasanan. Locale on verkkokaupan oletuskieli. Tästä verkkokaupasta luodut liput lähetetään tällä kielellä asiakkaille. Location_id on kaupan sijainnin ID. Mikäli kaupalla on useampi sijainti, täytyy verkkokaupan kutsuihin määrittää, mistä sijainnista ostetut liput luodaan.

5.4 Reitin luominen palvelimelle

Palvelimen routes-kansiossa on määriteltynä reitit, joita kutsumalla voidaan palvelimen ulkopuolelta kutsua palvelimensisäisiä kutsuja. Jotta shopifyn lähettämät webhookit pääsevät palvelimelle, luodaan uusi reitti muokkaamalla routes-kansiossa olevaa api.php-tiedostoa.

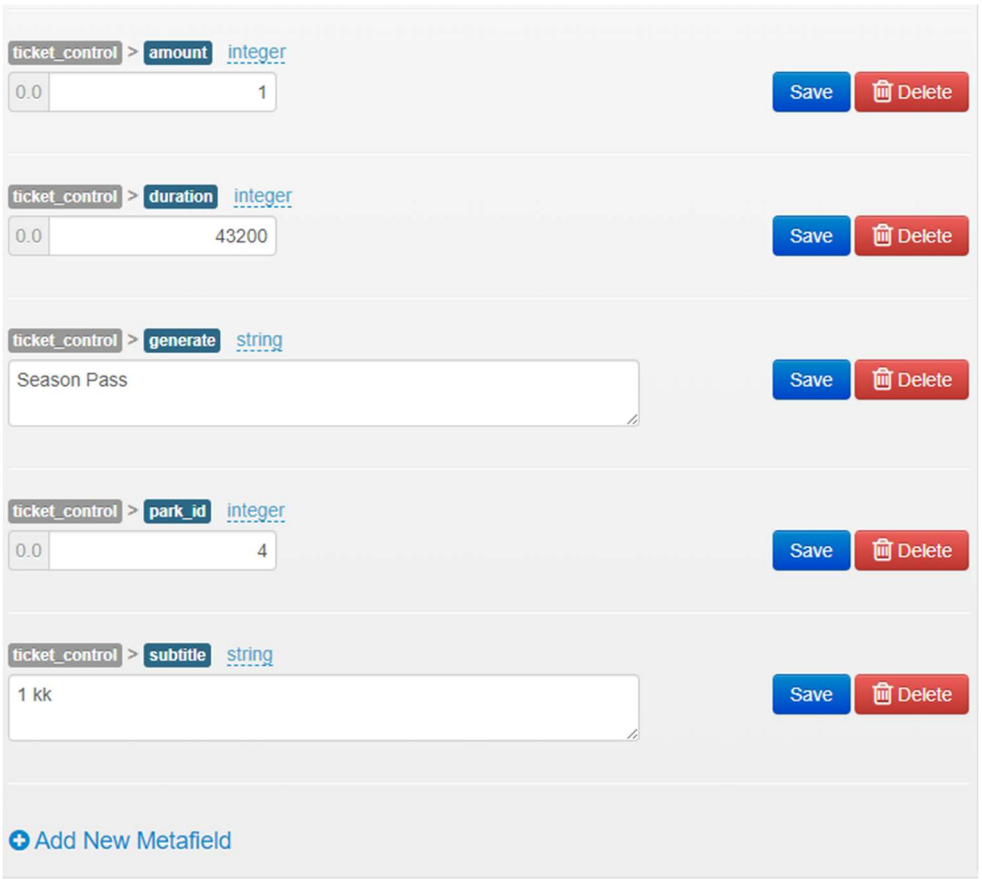
Seuraavan reittiryhmän sisään lisätään rivi:

Route::post('shopify_ticket', 'Api\ShopifyOrderController@add');

Tämän jälkeen ajetaan artisan komento *php artisan route:cache*, joka tallentaa käytettävät reitit palvelimen välimuistiin. Tämän jälkeen verkkokaupan lähettämät webhookit pääsevät palvelimelle asti.

5.5 Metatietojen asettaminen verkkokauppaan

Koska yhdessä verkkokaupassa myydään usean puiston lippuja, tulee yrityksen saada enemmän tietoa myydystä tuotteesta kuin se, mitä Shopify valmiiksi antaa. Päätettiin hankkia lisätieto metatiedon avulla. Metatiedot ovat käyttäjän asettamaa tietoa, joka on liitetty tiettyyn resurssiin. Tässä tapauksessa metadata on palvelimen tarvitsemaa lisätietoa lipusta, joka on liitetty Shopifystä myytävän tuotteen vaihtoehtoon eli varianttiin. Asennettiin Shopifyn verkkokauppaan kolmannen osapuolen applikaatio: ”Metafields editor”. Tämän avulla pystyttiin lisäämään tuotteille ja niiden varianteille metatietoja. Asetettiin metatiedot päivälippujen varianteille kuvan 4 mukaan.



Kuva 4. Metatietojen asetus

5.6 Verkkokaupasta ostetun lipun tallentaminen tietokantaan

Asiakkaan ostaessa lipun verkkokaupasta aiemmin asetettu webhook lähettää palvelimelle kutsun, jossa on mukana tilauksen tiedot. Näitä tietoja hyväksikäyttäen voimme luoda tietokantaan merkinnän ostetusta lipusta. Tässä kappaleessa käydään läpi globaalipalvelimen toiminnan webhookin saamisen jälkeen lipun tallentamiseen kantaan.

5.7 Webhookin oikeellisuuden tarkastaminen

Shopify on kehittänyt varmennusmenetelmän, jotta varmistetaan, että webhookia ei ole ulkopuolelta muokattu tai ettei webhookin kaltaisia haitallisia kutsuja lähetetä lipunluontirajapintaan. Webhookin oikeellisuus tarkastetaan ottamalla talteen Shopifyn kutsussa lähetetty hmac_header ja vertaamalla sitä palvelimella laskettua hmac arvoa vastaan. Hmac lasketaan käyttäen php:n valmista kutsua *hash_hmac* käyttäen kutsun *php://input* arvoa ja webhookin luonnin yhteydessä saatua salausavainta ja muuttamalla saatu tulos *base64* muotoon. Mikäli laskettu ja kutsun hmac arvot täsmäävät, voidaan olla varmoja, että kutsu on aito. Webhookin oikeellisuuden tarkastus on havainnollistettu kuvissa 5 ja 6.

```
13  function verify_Webhook($data, $hmac_header, $shopify)
14  {
15      if(!hash_equals($hmac_header, $calculated_hmac = base64_encode(hash_hmac('sha256',$data,$shopify->app_secret,true))))
16      {
17          throw new \Exception('hmac do not match!');
18      }
19      return hash_equals($hmac_header, $calculated_hmac);
20  }
```

Kuva 5, Webhookin oikeellisuuden tarkastus, 1/2

```php
if($shopify = Shopify::where('shop_domain', $_SERVER['HTTP_X_SHOPIFY_SHOP_DOMAIN'])->first())
{
    // Verify the webhook
    try
    {
        $verified = $this->verify_Webhook(file_get_contents('php://input'), $_SERVER['HTTP_X_SHOPIFY_HMAC_SHA256'],$shopify);
    }
    catch (\Exception $e)
    {
        Log::error($e->getMessage(), $this->logContext);
        return response()->json(['message' => $e->getMessage()])->setStatusCode(400);
    }
}
else
{
    Log::error('Could not verify webhook! ', $this->logContext);
    return response()->json(['invalid webhook'])->setStatusCode(200);
}
```

Kuva 6, Webhookin oikeellisuuden tarkastus, 2/2

5.8 Metatiedon hakeminen Shopifystä

Shopifyn webhook ei lähetä tilauksen tiedon mukana tuotteen metatietoja vaan nämä pitää hakea erikseen. Tilauksen mukana tulee tilatun tuotteen ja sen variantin ID:t. Näiden avulla voidaan hakea Shopifystä tuotteen metatiedot. Lähetetään cURL-kutsu Shopifyyn kuvan 7 mukaisesti. Kuvassa oleva *$this->shopify['api_url']* on privaatti applikaation osoite, joka sisältää käyttäjätunnuksen ja salasanan, *$product_id* on webhookista saatu tuotteen ID, ja *$variant_id* on saman tuotteen variantin ID. Tällä kutsulla saadaan paluuviestinä kyseisen tuotteen kyseisen variantin metatiedot JSON taulukkona.

```php
$url = $this->shopify['api_url'].'/products/'.$product_id.'/variants/'.$variant_id.'/metafields.json';
$curl = curl_init();
curl_setopt_array($curl, array(
    CURLOPT_RETURNTRANSFER => 1,
    CURLOPT_URL => $url,
    CURLOPT_USERAGENT => 'Codular Sample cURL Request'
));
$result = curl_exec($curl);
$json_response = curl_getinfo($curl, CURLINFO_HTTP_CODE);
$status = curl_getinfo($curl, CURLINFO_HTTP_CODE);
if($status != 200)
{
    throw new \Exception("Error: call to URL $url failed with status $status, response $json_response, curl_error " .
    curl_error($curl) . ", curl_errno " . curl_errno($curl));
}
curl_close($curl);
$response = json_decode($result, true);
```

Kuva 7. Metatiedon hakeminen Shopifystä.

5.9 Lipun luominen saaduista tiedoista

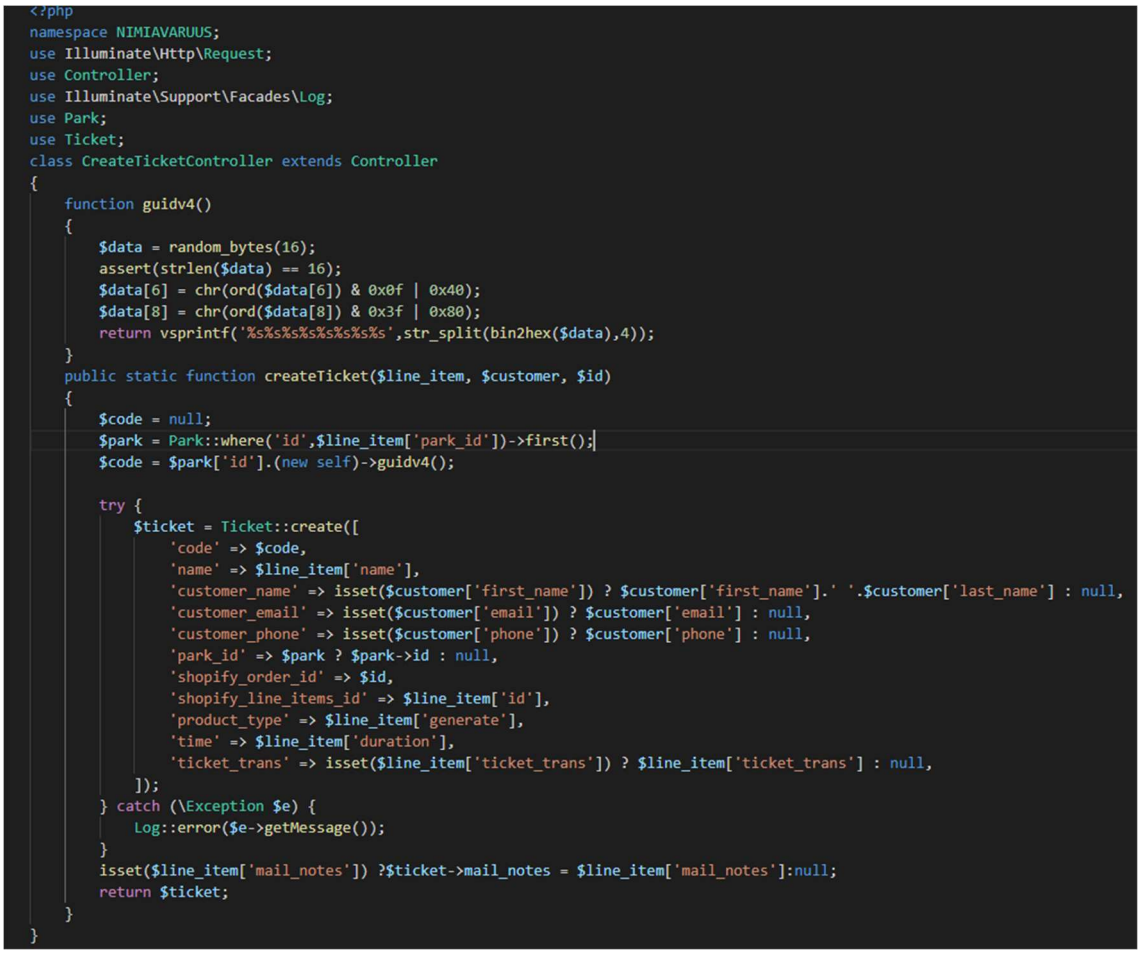
Tietokantojen hallintaan Laravelilla on käytössä Eloquent ORM. Tällä saadaan jokaiselle taulukolle tehtyä sitä vastaava malli. Nämä mallit helpottavat taulukosta etsimistä ja uusien rivien lisäämistä. Tehdään lipuille oma malli komennolla *php artisan make:model Ticket*. Tämä kutsu luo automaattisesti tarvittavat riippuvuudet mallille.

Jotta taulukkoon voidaan asettaa useamman sarakkeen arvo yhdellä kutsulla taulukon riviä luodessa, tulee määrittää sarakkeet, jotka voi täyttää tällä tavoin. Nämä voidaan määrittää model luokan *protected $fillable* -nimisellä taulukolla. Täytettävien sarakkeiden määrittäminen on esiteltynä kuvassa 8.

```
protected $fillable = [
    'code',
    'used',
    'used_in',
    'device_gid',
    'customer_name',
    'email_model',
    'customer_email',
    'ticket_trans',
    'customer_phone',
    'name',
    'park_id',
    'shopify_order_id',
    'shopify_line_items_id',
    'filename',
    'product_type',
    'time',
    'subtitle',
    'expires_at',
    'season',
    'usage_time_start',
    'usage_time_end',
    'pdftitle',
    'ticketText',
    'usable_on_any_park',
    'date',
    'variant_id',
    'ticket_type'
];
```

Kuva 8. Fillable-taulukon määrittäminen

Käytetään luotua Ticket-mallia lipun luomiseen tietokantaan kuvan 9 mukaan.



```php
<?php
namespace NIMIAVARUUS;
use Illuminate\Http\Request;
use Controller;
use Illuminate\Support\Facades\Log;
use Park;
use Ticket;
class CreateTicketController extends Controller
{
    function guidv4()
    {
        $data = random_bytes(16);
        assert(strlen($data) == 16);
        $data[6] = chr(ord($data[6]) & 0x0f | 0x40);
        $data[8] = chr(ord($data[8]) & 0x3f | 0x80);
        return vsprintf('%s%s%s%s%s%s%s%s',str_split(bin2hex($data),4));
    }
    public static function createTicket($line_item, $customer, $id)
    {
        $code = null;
        $park = Park::where('id',$line_item['park_id'])->first();
        $code = $park['id'].(new self)->guidv4();

        try {
            $ticket = Ticket::create([
                'code' => $code,
                'name' => $line_item['name'],
                'customer_name' => isset($customer['first_name']) ? $customer['first_name'].' '.$customer['last_name'] : null,
                'customer_email' => isset($customer['email']) ? $customer['email'] : null,
                'customer_phone' => isset($customer['phone']) ? $customer['phone'] : null,
                'park_id' => $park ? $park->id : null,
                'shopify_order_id' => $id,
                'shopify_line_items_id' => $line_item['id'],
                'product_type' => $line_item['generate'],
                'time' => $line_item['duration'],
                'ticket_trans' => isset($line_item['ticket_trans']) ? $line_item['ticket_trans'] : null,
            ]);
        } catch (\Exception $e) {
            Log::error($e->getMessage());
        }
        isset($line_item['mail_notes']) ?$ticket->mail_notes = $line_item['mail_notes']:null;
        return $ticket;
    }
}
```

Kuva 9. Lipun tallentaminen tietokantaan.

5.10 Lippusivun luonti

Luodaan lippusivu, joka näyttää lipun tiedot asiakkaalle halutulla tavalla. Lippusivu luodaan käyttämällä apuna laravel blade -HTML-mallia ja käyttäjä ohjataan sivulle TicketController-luokan kautta. TicketController-luokkaan luodaan kutsu *”showTicket”*, joka ottaa syötteeksi lipun koodin. Kutsu hakee lipun tiedot tietokannasta koodin perusteella ja lähettää tiedot lippusivulle. Lippusivulla näytetään tiedot käyttäjälle. SuperParkilla on Azuren web-service nimeltä qr.superpark.fi, joka kääntää syötetyn tekstin QR-koodiksi. Lipun koodista tehdään QR-koodi, joka näytetään kuvana lippusivulla. QR-koodin luonti on esitelty kuvassa 10. Lopullinen lippusivu on esiteltynä liitteessä 1.

```
<div style="text-align: center; top: 6vw; position: relative">
    <img src="{!!base64_encode(@file_get_contents(url('http://qr.superpark.fi/'.$ticketCode)))!!}"
    style="position: relative" width="40%" height="auto">
</div>
```

Kuva 10. QR-koodin luonti.

5.11 Lipun lähettäminen asiakkaalle

Lippu lähetetään asiakkaalle sähköpostilla. Tätä varten luodaan sähköpostisivu, johon liitetään linkki lippusivuun. Tämä sähköpostisivu tehdään käyttämällä laravelin blade -html-mallia sekä *php artisan make:mail* -komentoa. Tämä komento luo mailable-luokan palvelimen app/Mail-kansioon.

Sähköpostin lähetys voidaan tehdä kutsumalla Mail-luokkaa. Queue-funktion kutsuminen laittaa lähetyksen Laravelin työjonoon, josta työntekijät suorittavat sähköpostin valmistelun ja lähettämisen. Lippusähköpostin kutsuminen esiteltynä kuvassa 4.

```
if($this->tickets){
    Log::info('Sending tickets to customer ',$this->logContext);
    $email = Mail::to($customer['email'])->queue(new ticketSent($this->tickets, $this->shopify['locale']));
}
```

Kuva 11. Sähköpostin lähetyksen kutsuminen

5.12 Ranneketaulukon jakaminen rannekkeisiin ja tuotteisiin

Rannekkeet ovat olleet tähän asti kertakäyttöisiä, jolloin ei ole ollut tarvetta pitää ranneketta ja rannekkeella olevia tuotteita erillään. Tarve tälle on kuitenkin nyt tullut kausikorttien kautta. Kausikortit ovat tuotteita, jotka ovat voimassa tietyn kauden (1 kk, 3 kk, 6 kk, 12 kk) ajan. Tänä aikana asiakas pääsee puistoon puiston aukioloajan puitteissa. Kausikortteina käytetään silikonista tehtyä ranneketta. Tätä ranneketta näyttämällä puiston vastaanotossa asiakas saa kertakäyttöisen rannekkeen puistoon täksi päiväksi.

Luodaan ”Rannekkeet”- ja ”Tuotteet” -taulukot. Rannekkeet taulukko pitää sisällään tiedon rannekkeista sekä rannekkeeseen liitetystä käyttäjästä. Lisäksi rannekkeella on tieto, missä se on viimeksi luettu sekä merkintä, onko se poistettu käytöstä. Tuotteet taulukko taas pitää sisällään

merkinnät rannekkeesta, jolle tuote on määritelty, tuotteen voimassaoloajan, puiston, johon ranneke käy, tuotteen nimen sekä voimassaoloajan keston minuutteina.

5.13 Lipun vaihtaminen rannekkeeseen puistossa

Lipun QR-koodi luetaan puistossa SuperManager -ohjelmalla, jolloin SuperManager saa lipun koodin. Tällä haetaan palvelimelta lipun tiedot.

Mikäli lippu on voimassa, voidaan se merkata käytetyksi. Tämän jälkeen voidaan luoda asiakkaalle ranneke, jolla pääsee puistoin porteista rannekkeen voimassaoloajan ajan.

5.14 Rannekkeen liittäminen pelitiliin

Rannekkeen voi liittää käyttäjän pelitiliin joko ranneketta luotaessa antamalla käyttäjän tietokanta ID tai käyttäjän puhelinnumero. Vaihtoehtoisesti asiakas voi liittää rannekkeen omalle pelitililleen kirjautumalla pelitilille osoitteessa app.superpark.fi.

```
$user = User::where('id', $request->user_gid)->first();
if($wristband && $user)
{
    $wristband->user_id = $user->id;
    $wristband->save();
}
```

Kuva 12. Rannekkeen liittäminen pelitiliin käyttäjän tietokanta-ID:llä.

```php
$user = User::where('telephone', $request->telephone)->first();
if(!$user)
{
    $namePrefix = 'Sparker';
    try {
        $ai = Ai::create(['requester' => 'user']);
        if (!$ai) {
            throw new \Exception('Could not create new ai record on ticket_bought');
        }
        $displayName =  $namePrefix . $ai->id;
    }catch (\Exception $e) {
        Log::error($e->getMessage());
        $displayName = 'Sparker';
    }

    $user = User::create([
        'firstname' => $displayName,
        'telephone' => $request->telephone,
        'uses_application' => 0,
        'temporary' => 0,
    ]);
    if (!$user) {
        throw new \Exception('Could not create the temporary user');
    }
}
```

Kuva 13. Rannekkeen liittäminen pelitiliin käyttäjän puhelinnumerolla.

6 Tuote ja tuotteen tarkastelu

Projektin tuloksena saatiin toimeksiantajalle toimiva lipunmyyntijärjestelmä. Ostaessaan verkkokaupasta lipun asiakas saa automaattisesti sähköpostiinsa yksilöidyn lipputuotteen, jonka vaihtaminen rannekkeeseen onnistuu lukemalla lipussa oleva QR-koodi puiston vastaanotossa.

Palvelimen rajapintojen ohjelmointi ja testaus tapahtui työkoneelle asennetulla virtuaalipalvelimella. Laravelilla on valmis virtuaalikehitysympäristö Homestead (23). Homestead asennettiin Oraclen VirtualBox-alustalle ja sen pystyttämiseen käytettiin apuna Vagrantia. Verkkokaupan testausta varten pystytettiin testiverkkokauppa Shopifyyn. Tässä testikaupassa on samoja tuotteita ja niihin on asetettu samat metatiedot kuin oikeassa verkkokaupassa olevilla tuotteilla, tällä tavoin testikauppaa voidaan käyttää tilausten testaukseen. Tilauksen testaamista varten virtuaalitestiympäristö jaettiin julkiseen verkkoon ngrokin avulla. Ngrokilla jaettu osoite asetettiin testishopifyn webhookin osoitteeksi, jolloin verkkokaupasta tehdyt tilaukset lähetetään virtuaalipalvelimelle. Shopifyn admin paneelista pystyy tekemään tilauksia. Tällä tavalla saatiin testattua tilausten tekeminen. Virtuaalipalvelimelle on asetettu oma tietokanta, johon luodut liput tallennetaan. Virtuaalipalvelimelle asetettiin sähköpostipalvelimeksi mailtrap. Mailtrapilla lähetetyt sähköpostit eivät mene koskaan perille, vaan jäävät mailtrapin omaan saapuneet kansioon. Tällä tavoin voidaan turvallisesti testata sähköpostin lähetys. Sähköpostin mukana tullut linkki lippusivuun ohjaa virtuaalipalvelimen lippusivuun, joka pystyy hakemaan lipun tiedot omasta tietokannastaan.

SuperManagerille kehitettyjen rajapintojen, kuten lipun tarkistus, merkkaus käytetyksi ja liittäminen rannekkeeseen, testattiin Insomnia-nimisen sovelluksen kanssa. Insomnia on REST-kutsujen lähettämiseen kehitetty sovellus. Tällä voidaan lähettää kutsuja palvelimelle, kutsuille voidaan asettaa otsikoita, tunnistautumiskeinoja sekä kutsun runko. Insomnia näyttää palvelimelta saadun vastauksen, ja tällä tavalla voidaan testata, että lähetetyt kutsut toimivat halutulla tavalla.

Kehityksen aikana havaittuja haasteita on tullut monesta suunnasta. Shopify pyrkii estämään mahdollisia DDOS-hyökkäyksiä palvelimelleen käyttämällä ns. Leaky bucket-algoritmia. Tämä algoritmi laskee samasta osoitteesta tulleet kutsut yhteen ja vähentää tästä tietyn määrän joka sekunti. Yhteenlaskettu kutsujen määrä ei saa ylittää kutsujen ylärajaa tai kutsu hylätään. (24) Tämän seurauksena lipunluontirajapinnan käyttämä tuotteen metatietojen haku Shopifystä

saattoi epäonnistua, mikäli yhdessä tilauksessa on useita lippuja. Tämä korjattiin hidastamalla metatietojen hakua php:n sleep-komennolla.

Työn seuraava vaihe on kausikorttituotteiden lisääminen ja niiden käsittely. Lisäksi Aasian kävijämäärät ovat niin suuria, että liput tulee myydä tietyille aikajaksoille koko päivän lippujen sijaan. Liput tulee myös voida tilata tietylle päivälle, jotta voidaan varmistaa asiakkaiden pääsy puistoon haluamanaan ajankohtana.

Lisäksi tulevaisuudessa haluttavia kehityksiä järjestelmään on mahdollisuus lisätä useampia eri tuotteita Shopifystä myytäviin lipputuotteisiin. Tästä esimerkkinä lipputuote, jonka mukana saa trampoliinisukat puiston kassalta.

SuperParkilla on tavoitteena avata yhteensä 100 puistoa vuoteen 2023 mennessä. Tämän tavoitteen saavuttamiseksi uusien puistojen verkkokauppa tulee olla helposti pystytettävissä. Tätä varten tarvitaan automatisoituja järjestelmiä esimerkiksi verkkokaupan tuotteiden luomiseen.

Testien tekeminen manuaalisesti, kuten tässä työssä on tehty, on hidasta ja siitä jää helposti pois vaiheita tai koodin muokkaaminen saattaa aiheuttaa vaikutuksia testattujen kohteiden ulkopuolella. Tätä varten kannattaa tehdä palvelimelle automaattiset testaukset. Laravel tukee jo automaattisten testien tekemistä, jolla voidaan tehdä ns. Feature testejä, kuten lipun luominen on, mutta sillä voi myös tehdä ns. Browser-testejä, kuten lippusivun testaus. Näiden testien luominen tulee helpottamaan jatkokehitystä, kun testit voidaan ajaa automaattisesti uutta versiota luodessa. (25)

7 Yhteenveto

SuperPark laajentaa markkinoitaan Aasiaan. Tämän takia verkkokauppa on uudistettava mahdollistaakseen lisääntyneet verkosta ostettavien lippujen määrät. Lisäksi kassat on muutettava ulkomailla sopiviksi kassoiksi. Tätä varten otettiin käyttöön Shopify. Opinnäytetyön tarkoituksena oli rakentaa olemassa olevalle palvelimelle rajapinnat Shopifyn verkkokaupasta ostettujen lippujen käsittelyyn.

Työ aloitettiin tutkimalla Shopifyn tilauksista ja tuotteista tallentamaa dataa ja kuinka tätä voitaisiin hyödyntää lipunmyynnissä. Shopifyssä on mahdollista lähettää webhookeja tiettyjen tapahtumien takia. Tätä käytetään lipunmyynnin yhteydessä tilauksen lähettämisessä globaalipalvelimelle. Shopifyn verkkokauppaan asennettiin uusi kolmannen osapuolen tekemä applikaatio, Metafields editor. Tämän avulla voitiin lisätä tuotteille metatietoja. Metatiedoilla voitiin tuotteille asettaa lipun luontiin tarvittavia tietoja.

Palvelin tarkastaa saadun webhookin oikeellisuuden vertaamalla webhookin mukana tullutta HMAC headeria palvelimen päässä laskettuun HMAC-arvoon. Mikäli nämä täsmäävät, palvelin lisää työjonoon uuden työn, joka luo liput tilauksen perusteella.

Työ hakee Shopifystä tilatun tuotteen metatiedot ja liittää nämä osaksi tilauksen Line-Items taulukkoa. Seuraavaksi työ tarkastelee metatietoja ja etsii sieltä ”generate” kenttää, jonka avulla tiedetään, luodaanko tilauksesta lippua. Mikäli tämä löytyy, työ kutsuu palvelimen createTicket controllerin kutsua, jolla lippu tallennetaan tietokantaan. Tämä kutsu palauttaa työlle tallennetun lipun tiedot. Tämän jälkeen työ luo sähköpostin lähetystyön, jolla lähetetään sähköposti asiakkaalle tilauksesta. Tämä sähköposti pitää sisällään linkin lippusivuun. Lippusivu on reitti palvelimen osoitteeseen, joka näyttää lipun sen koodin perusteella. Lippusivu on muotoa *app.superpark.fi/ticket/{code},* jossa code on lipun uniikki ja arvaamattomissa oleva koodi. Tähän osoitteeseen mentäessä palvelin lähettää ticketControllerille kutsun, jossa lipun koodin perusteella haetaan lipun tiedot tietokannasta ja lähetetään Laravelin front endiin. Nämä tiedot asetetaan osaksi lippusivun HTML-koodia, jonka avulla asiakas saa näkyviin lipun tiedot.

Asiakas voi sähköpostiinsa saadun lipun kanssa tulla puiston kassalle, jossa lipussa oleva QR-koodi luetaan SuperManager-ohjelmalla. Samalla ohjelmalla liitetään lippu rannekkeeseen, tai siinä tapauksessa, että asiakas tulee puistoon ilman lippua, voi hän ostaa kassalta lipun ja puiston henkilökunta luo rannekkeen SuperManager-ohjelmalla. SuperManager lukee luotavasta

rannekkeesta RFID-tagin ja lähettää sen sekä lipun palvelimelle. Palvelin lisää tietokantaan merkinnän uudesta rannekkeesta ja liittää sille lippua vastaavan tuotteen. Tällä tavoin asiakas saa toimivan rannekkeen ja pystyy kulkemaan porteista rannekkeelle merkatulla aikavälillä.

Porteista kulkiessaan asiakas näyttää ranneketta portin lukijalle. Portti lähettää tästä viestin saittipalvelimelle, joka sisältää rannekkeen RFID tagin. Palvelin etsii tietokannasta kyseisellä tagilla varustettua ranneketta ja tälle rannekkeelle asetettua tuotetta. Mikäli tuote löytyy ja se on tällä hetkellä voimassa oleva tuote, lähettää palvelin portille OK-kutsun ja portti päästää asiakkaan puistoon.

Lähteet

(1) App Service overview - Azure. Available at: https://docs.microsoft.com/en-us/azure/app-service/overview. Accessed 22.05., 2019.

(2) QR Code Basics. Available at: https://www.qr-code-generator.com/qr-code-marketing/qr-codes-basics/. Accessed 24.4., 2019.

(3) Abeysinghe S. RESTful PHP Web Services. Olton: Packt Publishing; 2008.

(4) What is RFID. Available at: https://www.rfidjournal.com/faq/show?49. Accessed -04-17, 2019.

(5) Using webhooks. Available at: https://help.shopify.com/api/getting-started/webhooks#verify-webhook. Accessed 16.6., 2018.

(6) Sparkkaus. Available at: https://superpark.fi/sparkkaus/. Accessed May 29, 2019.

(7) Otwell Taylor. Laravel documentation. Available at: https://laravel.com/docs/5.3. Accessed 26.3., 2018.

(8) Composer Documentation. Available at: https://getcomposer.org/doc/00-intro.md. Accessed 16.1., 2019.

(9) Artisan Console. Available at: https://laravel.com/docs/5.7/artisan. Accessed 13.2., 2019.

(10) Node Documentation. Available at: https://nodejs.org/en/about/. Accessed 16.1., 2019.

(11) About npm. Available at: https://docs.npmjs.com/about-npm/. Accessed 13.2., 2019.

(12) Larkin M. Shopify Application Development. Olton Birmingham: Packt Publishing; 2014.

(13) Ubuntu release cycle. Available at: https://www.ubuntu.com/about/release-cycle. Accessed 13.2., 2019.

(14) Vagrant documentation. Available at: https://www.vagrantup.com/intro/index.html. Accessed 13.2., 2019.

(15) PHP: What is PHP. Available at: http://fi2.php.net/manual/en/intro-whatis.php. Accessed 23.01., 2019.

(16) JSON. Available at: https://www.json.org.

(17) PostgreSQL: About. Available at: https://www.postgresql.org/about/. Accessed 13.2., 2019.

(18) Metafields editor by Webify Technology. Available at: https://apps.shopify.com/metafields-editor. Accessed 13.2., 2019.

(19) Supervisor website. Available at: http://supervisord.org/index.html. Accessed 13.2., 2019.

(20) Queues - Laravel - The PHP Framework For Web Artisans. Available at: https://laravel.com/docs/5.8/queues. Accessed May 29, 2019.

(21) Adding variants - Shopify Help Center. Available at: https://help.shopify.com/en/manual/products/variants/add-variants. Accessed 16.2., 2019.

(22) Mail - Laravel - The PHP Framework For Web Artisans. Available at: https://laravel.com/docs/5.8/mail. Accessed May 29, 2019.

(23) Laravel Homestead. Available at: https://laravel.com/docs/5.7/homestead. Accessed 23.1., 2019.

(24) REST Admin API rate limits. Available at: https://help.shopify.com/en/api/reference/rest-admin-api-rate-limits. Accessed -05-24, 2019.

(25) Testing: Getting Started - Laravel - The PHP Framework For Web Artisans. Available at: https://laravel.com/docs/5.8/testing. Accessed May 29, 2019.

Liitteet


KAUSIKORTTI
SUPERPARK VANTAA
Lippu on ostettu: 2019-02-25
Lippu on voimassa 2020-02-25 asti. Lippu vaihdetaan rannekkeeseen, jolla puisto on käytössäsi koko päivän. Voit välillä poistua puistosta ja tulla saman päivän aikana takaisin.
SuperPark Vantaa | Tammiston kauppatie 13, 01510 Vantaa
Lipunmyynti ja info: +358 44 488 6211 | www.superpark.fi