HAMK
Hämeen ammatti-
korkeakoulu

Kotikäynnin ajanvarausjärjestelmä

PrestaShop ja VismaPay integraatio

Ammattikorkeakoulututkinnon opinnäytetyö
Tieto- ja viestintätekniikka, insinööri (AMK)
Syksy, 2023
Samu Savikko

HAMK
Hämeen ammatti-
korkeakoulu

Tieto- ja viestintätekniikka, insinööri (AMK) Tiivistelmä
Tekijä Samu Savikko Vuosi 2023
Työn nimi Kotikäynnin ajanvarausjärjestelmä
Ohjaaja Petri Kuittinen

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa ajanvarausjärjestelmä hyvinkääläiselle yritykselle, joka tarjoaa tietotekniikkapalveluita yritys- ja yksityisasiakkaille. Kotikäynnit ovat toimeksiantajan tarjoama palvelu asiakkaillensa, jotka eivät pääse liikkeeseen paikanpäälle tai ongelma ei ole ratkaistavissa liikkeessä. Ajanvarausjärjestelmän tarkoituksena on tarjota tapa kotikäynti asiakkaille varata ja maksaa kotikäynti heille ja toimeksiantajalle sopivalle ajalle.

Opinnäyteyössä perehdyttiin WordPress-julkaisualustaa, VismaPay-maksupäätteeseen, PrestaShop-verkkokauppaa, työn kehitysympäristöön ja työssä käytettyihin verkkotekniikoihin. Lisäksi työssä perehdytään suppeasti relaatiotietokantaan, http-protokollaan ja http-pyynnöissä käytettyihin dataformaatteihin.

Ajanvarasjärjestelmä toteutettiin WordPress-julkaisualustalle käyttäen sen tarjoamia ohjelmistorajapintoja. Toteutuksessa on myös käytetty VismaPay ja PrestaShop PHP-kirjastoja. Ajanvarausjärjestelmässä pääsääntöisesti käytettiin REST-päätepisteitä työn toiminnallisuuden tuottamiseksi.

Projektin lopputuloksena syntyi WordPress-lisäosa, jonka avulla asiakas pystyy varmaan ja maksamaan kotikäyntiaikoja. Toimeksiantaja taas pystyy hallitsemaan asiakkaiden varaamia kotikäyntiaikoja ja kotikäyntiaika varauksien kuitit luodaan tilauksen myötä PrestaSho-verkkokauppaan. Toimeksiantaja pystyy ajanvaruksen PrestaShop-verkkokauppaan lisäämisen jälkeen tulostamaan kotikäynnin kuitin RockPos-kassajärjestelmän kautta.

Avainsanat WordPress, PrestaShop, VismaPay, PHP, JavaScript, CSS ja HTML
Sivut 22 sivua

HAMK
Hämeen ammatti-
korkeakoulu

Information and Communication Technology
Author Samu Savikko
Subject Home Visit Time Reservation System
Supervisor Petri Kuittinen

Abstract
Year 2023

The goal of this thesis was to design and produce a time reservation system for an information technology services company located in Hyvinkää. They offer a home visit service whose purpose is to help people who cannot bring their IT-devices to the shop, or their problem cannot be solved inside the shop. The Time reservation system's purpose is to provide a way for the company's clients to reserve and pay a home visit for a time that is convenient for both parties.

The knowledge base of this thesis consists of information about WordPress web content management system, VismaPay web payment service, PrestaShop online store, development environment and web technologies used to create the product. In addition, relational databases, http protocol and data formats that are briefly introduced.

The time reservation system was implemented using WordPress web content management system and with the application programming interfaces it provides. The time reservation system also uses VismaPay and PrestaShop PHP libraries to make using their services easier. The time reservation system primarily uses REST endpoints to provide its core functionality.

The development work resulted into a WordPress plugin that enables customers to pay and reserve home visit times. The employer can manage reserved home visits and the home visits are then added to the PrestaShop online store as orders. Finally, the employer can then print the home visit receipts using the RockPos point of sale system.

Keywords WordPress, PrestaShop, VismaPay, PHP, JavaScript, CSS and HTML
Pages 22 pages

Sisällys

Kuvat, taulukot ja kaavat

1 Johdanto

Toimeksiantaja tarjoaa asiakkaillensa kotikäyntipalvelua minkä tarkoituksena on auttaa ihmisiä IT asioissa heidän kotonansa, jos he eivät pääse jostain syystä liikkeeseen paikanpäälle tai ongelmaa ei pystytä ratkaisemaan liikkeessä. Toimeksiantajan asiakkaat ovat aikaisemmin varanneet kotikäyntejä soittamalla liikkeeseen tai kysymällä paikan päällä kotikäynneistä. Ajanvarausjärjestelmän avulla toimeksiantaja pystyy automatisoimaan kotikäyntien prosessia ja toimeksiantajan asiakkaat saavat helpon tavan varata ja maksaa kotikäyntejä.

Työ sai alkunsa syksyllä 2022 työharjoittelussa, kun toimeksiantaja pyysi kotikäynnin ajanvarausjärjestelmää verkkosivuilleen. Työn alkuperäisenä tarkoituksena oli vain varata kotikäyntiaikoja, mutta työtä laajennettiin mahdollistamaan verkossa maksamisen VismaPay-verkkomaksupalvelulla ja kuittien lisääminen toimeksiantajan PrestaShop-verkkokauppapalveluun.

Toimeksiantajan nykyiset verkkosivut on kehitetty WordPress-julkaisualustalle, jonka takia ajanvarausjärjestelmä kehitetään WordPress-lisäosana käyttäen lisäosien kehityksessä käytettyjä ohjelmointikieliä ja tekniikoita.

Tämän toiminnallisen opinnäytetyön tavoitteena on suunnitella ja toteuttaa ajanvarausjärjestelmä WordPress-lisäosana, joka mahdollistaa kotikäynnin etukäteen maksamisen ja kuitin luomisen toimeksiantajan verkkokauppaan.

Työn teoriaosuudessa käsitellään WordPress-julkaisualusta, PrestaShop-verkkokauppa, VismaPay-verkkomaksupääte, WordPress-julkaisualustan toimintaan vaadittu kehitysympäristö, WordPress-lisäosan kehitykseen vaadittuja verkkotekniikoita ja WordPress-julkaisualustan tarjoamia rajapintoja.

Työn toteutus- ja suunnitteluosassa käsitellään WordPress-lisäosan kehityksen vaatimuksia, koodaus standardeja ja projektin rakennetta. Toteutus- ja suunnitteluosiossa myös käsitellään WordPress-lisäosaan REST päätepisteiden luomista ja käyttämistä joista projektin toiminnallisuus pääsääntöisesti koostuu. Osiossa myös käsitellään lyhytkoodinluominen ja VismaPay PHP kirjaston käyttäminen ja PrestaShop-verkkopalvelun kanssa kommunikoiminen tilauksien lisäämiseksi PrestaShop-verkkokauppa alustalle.

Tietokannan suunnittelu, käyttöliittymän tyylittely, ylläpitonäkymän luominen, ylläpitoasetuksien luominen, päätepisteiden testaus, PrestaShop-verkkopalvelun integroinnin testaus, VismaPay-verkkomaksupäätteen testaus.

2 Teoria

Tässä osiossa käsitellään WordPress-lisäosan kehityksessä käytettyjä verkkotekniikoita, WordPress-julkaisualustan vaatimaa ympäristöä, toteutuksessa käytettyjä työkaluja, WordPress-julkaisualustaa, WordPress-rajapintoja, VismaPay-verkkomaksupäätettä, PrestaShop-verkkokauppaa ja PrestaShop-verkkopalvelua.

2.1 HTML

HTML on merkintäkieli minkä tarkoituksena on kertoa selaimelle verkkosivujen sisällöstä käyttämällä elementtejä ja tekstiä. Jokaisella elementillä on aloitus- ja lopetustagit minkä väliin voidaan sijoittaa muita elementtejä tai tekstiä. Elementin aloitustagiin voi lisätä attribuutteja millä täytyy olla nimi ja arvo. Attribuuttien tarkoituksena on kertoa tarkemmin elementistä ja sen toiminnallisuudesta. HTML-dokumenttiin on myös mahdollistaa upottaa muuta sisältöä kuten kuvia, videoita, äänitiedostoja ja skriptejä käyttämällä niille tarkoitettuja elementtejä. Selaimet muodostavat HTML-dokumentista DOM-puun, joka on muistissa oleva dataversio HTML-dokumentista ja tätä versiota voi muokata skripteillä kuten JavaScript. (HTML Standard, 2023)

2.2 JavaScript

JavaScript on verkkoympäristössä käytetty skriptauskieli minkä standardi on ECMAScript. ECMAScript on olio-ohjelmointikieli minkä tarkoituksena on manipuloida olemassa olevaa ympäristöä. Tämä tarkoittaa sitä, että ECMAScript ei toimi itsenään vaan se tarvitsee isäntäympäristön mikä tarjoaa sille tarvittavat objektit ja funktiot ympäristön manipuloimiseen. ECMAScript alun perin tarkoitettiin verkkoympäristön skriptauskieleksi, mutta sitä voi nykyään käyttää muissakin ympäristöissä kuten Node.js. (ECMAScript 2024 Language Specification, 2023)

FETCH-ohjelmistorajapinta minkä avulla JavaScript pystyy lähettämään http pyyntöjä ja vastaanottamaan http-vastauksia. FETCH-ohjelmistorajapinta tarjoaa julkisen funktion fetch

mikä on yksinkertainen ja looginen lähettää pyyntöjä asynkronisesti verkossa. (MDN Web docs, 2023-a)

2.3 Dataformaatit

JSON on tekstipohjainen dataformaatti mikä esittää strukturoitua dataa JavaScript objekti syntaksin mukaisesti ja sitä käytetään datan siirtämiseen Web-applikaatioissa. Vaikka JSON seuraakin JavaScript objektien syntaksia voidaan sitä käyttää muissakin ohjelmointikielissä kuten PHP. (MDN Web Docs, 2023-b)

XML on merkintäkieli mikä muistuttaa HTML-merkintäkieltä, mutta se ei käytä etukäteen määriteltyjä tageja. Käyttäjät pystyvät määrittelemään omat taginsa XML-merkintäkielessä tarpeidensa mukaan ja vaikka dataa siirrettäisiin toisesta järjestelmästä toiseen, pystytään dataa käsittelemään samalla tavalla jokaisessa järjestelmässä sen standardoidun syntaksin ansiosta. XML-dataa käytetään Web-applikaatioiden välisessä kommunikoinnissa samalla tavalla kuin JSON-dataa. (MDN Web Docs, 2023-c)

2.4 CSS

CSS on tyylittelykieli minkä tarkoituksena on kertoa miltä HTML elementit näyttävät verkkosivuilla vierailevalle käyttäjälle. CSS-merkintäkielessä luodaan ryhmiä sääntöjä mitkä kertovat elementeille miltä niiden täytyisi näyttää. CSS-sääntöryhmät alkavat valinnalla missä valitaan elementti sen tyypin tai attribuuttien perusteella, jonka jälkeen elementille voi määritellä sääntöjä kertomaan miltä sen tulee näyttää. Säännöissä määritellään elementin ominaisuus mitä halutaan muokata, jonka jälkeen ominaisuudelle määritellään arvo minkä täytyy olla sen sallimassa muodossa. (MDN Web Docs, 2023-d)

2.5 PHP

PHP on avaimen lähdekoodin yleiskäyttöön tarkoitettu skriptauskieli mikä soveltuu myös verkkosivujen kehitykseen, sillä PHP-koodia voi upottaa HTML koodin sekaan käyttämälle PHP aloitus- ja lopetustageja. PHP on palvelimella suoritettava ohjelmointikieli minkä tulos lähetään käyttöliittymässä olevalle käyttäjälle HTML-koodina. Tämä tarkoittaa sitä, että sivuilla olevalla käyttäjä näkee vain koodin tuloksen eikä itse koodia. (PHP, n.d.)

2.6 Kehitysympäristö

XAMPP on avoimen lähdekoodin Apache-verkkopalvelinjakelu, joka sisältää MariaDB-relaatiotietokannan hallintajärjestelmän, PHP- ja PERL-ohjelmointikielet. XAMPP-paketin tarkoituksen on olla helppo asentaa, käyttää ja se on saatavilla Windows ja Linux käyttöjärjestelmille. (Apache Friends, 2023)

Apache on avoimen lähdekoodin http-verkkopalvelin sovellus mikä julkaistiin vuonna 1995. Apache on yksi maailman käytetyimmistä verkkopalvelin sovelluksista ja sen tarkoituksen on tarjota turvallinen, tehokas, laajennettava verkkopalvelinsovellus, joka pysyy ajan tasalla nykyisten http-standardien kanssa. (Apache HTTP Server Project, 2023)

MariaDB on avoimen lähdekoodin relaatiotietokannan hallintajärjestelmä mikä perustuu MySQL relaatiotietokannan hallintajärjestelmään. MariaDB sisältää kaikki yleisimmät tietokantamoottorit kuten InnoDB mitä se käyttää oletus tietokantamoottorinaan. (MariaDB.org, n.d.)

Relaatiotietokanta on tietokantamalli mikä koostuu tauluista, jotka sisältävät rivejä eli tietueita ja sarakkeita eli attribuutteja. Relaatiotietokantamallissa voidaan tietueita yhdistää keskenään toisiinsa liittyvien attribuuttien avulla. (MariaDB.org, 2020)

Esimerkki relaatiotietokannasta voisi olla, vaikka kirjatietokanta missä on kaksi taulua kirjailijat ja kirjat. Kirjailijalla on attribuutit nimi, syntymäaika ja tunniste. Kirjalla on taas attribuutit nimi, kuvaus, julkaisupäivä, tunniste ja kirjailijan tunniste. Nyt näiden kahden taulun tietueet voidaan yhdistää niiden keskenään jakavan kirjailijan tunnisteen avulla, jolloin lopputulokseksi saadaan yksi taulu minkä tietueissa olisi kirjailijan tiedot ja kirjan tiedot.

Visual Studio Code on Microsoftin kehittämä ilmainen koodieditori mitä voi laajentaa ja muokata lisäosilla ja teemoilla. Visual Studio Code tarjoaa kehittäjille monia erilaisia operaatioita kuten virheidenjäljitys, tehtävien suorittaminen ja versionhallinta. Visual Studio Code myös käyttää Intellisense-teknologiiaa minkä tarkoituksena on korostaa syntaksia ja täydentää muuttujatyyppien, funktioiden määrityksien ja lisättyjen moduulien mukaan. (Visual Studio Code, n.d.)

2.7 WordPress

WordPress on vuonna 2003 alkunsa saanut julkaisualusta, jonka tavoitteena on olla helppo, suorituskykyinen ja turvallinen tapa toteuttaa verkkosivuja. WordPress on yksi suosituimmista verkkosivujen toteutustavoista maailmassa ja noin 43 % maailman verkkosivuista on toteutettu sitä käyttäen. WordPress on toteutettu käyttäen PHP ohjelmointikieltä ja MySQL-tietokantaa ja alustalle on mahdollista laajentaa lisäosilla ja teemoilla. (Wordpress, n.d.)

WordPress-julkaisualustan mukana tulee vain sen vaatima perustoiminnallisuus, mitä voidaan muokata tai laajentaa lisäosilla ilman koodaamistaitoa lataamalla yhteisönluomia lisäosia alustan tarjoamasta lisäosa hakemistosta, joka löytyy alustan ylläpitonäkymästä.

Käyttäjien on mahdollista myös itse luoda lisäosia, jos hakemistosta ei löydy tarpeisiinsa tarvittua lisäosaa käyttäen PHP-ohjelmointikieltä. Lisäosat voivat sisältää muutakin kuin PHP ohjelmointikieltä kuten kuvia, CSS ja JavaScript sisältöä. Lisäosat käyttävät WordPress API-rajapintaa sen toiminnallisuuden muokkaamiseen ja laajentamiseen. (WordPress Developer Resources, 2023. -a)

Teemat ovat tapa WordPressin käyttäjille muokata sivuston ulkonäköä ja sen vaihtaminen muuttaa mitä sivuilla vieraileva käyttäjä näkee käyttöliittymässä. Käyttäjien on myös mahdollista ladata yhteisönluomia teemoja alustan tarjoamasta lisäosahakemistosta, joka löytyy alustan ylläpitonäkymästä samalla tavalla kuin lisäosatkin.

Käyttäjät voivat luoda omia teemojaan tai laajentaa olemassa olevia teemoja niille tarkoitetuilla WordPress API-rajapinnoilla. WordPress-lisäosia voi ajatella kokoelmina malleja, CSS-tiedostoja ja muuta sisältöä kuten kuvia. Mallien tarkoituksena on kertoa sivustolle mitä ja miten tietoa esitetään sivulla oleville käyttäjille ja CSS taas kertoo miltä sivun HTML elementtien tulee näyttää. Vaikka teemoissa olisi mahdollista luoda samankaltaista toiminnallisuutta kuin lisäosissa sillä teeman vaihdossa kaikki teemaan ominaisuudet katoavat. (Wordpress Developer Resources, 2023-b)

API eli ohjelmointirajapinta on rajapinta mikä toimii määritetyillä säännöillä mitkä mahdollistavat kahden eri sovelluksen kommunikoida keskenään. WordPress API on WordPress alustan tarjoama rajapinta mikä on kokoelma pienempiä rajapintoja, joilla jokaisella on oma käyttötarkoituksensa. WordPress Apin tarkoituksena on tarjota työkaluja lisäosien ja teemojen kehittämiseen.

2.7.1 Plugin API

Plugin API eli lisäosarajapinta tarjoaa lisäosille ja teemoille funktioita, joiden tarkoituksena on kytkeä lisäosien ja teemojen funktioita itsensä, toistensa ja WordPress alustan kanssa, näitä funktioita kutsutaan koukuiksi. Koukkuja on kahta eri tyyppiä toiminnot ja filtterit.

Toimintojen tarkoituksena on olla funktioita, joita suoritetaan tietyssä kohtaa lisäosan, teeman tai WordPress alustan suoritusta. Koukkujen luonnissa käyttäjän tulee luoda funktio mikä suoritetaan, kun koukkua kutsutaan ja luotu funktio lisätään olemassa olevaan koukkuun tai uuteen koukkuun käyttämällä sille tarkoitettua funktiota. Koukkuja voidaan kutsua taas sille tarkoitetulla funktiolla ja kun funktiot kutsutaan se suorittaa kaikki siihen kytketyt funktiot.

Filtterit eroavat toiminnoista siten, että niiden läpi kuljetaan arvoja, joita filtterit muokkaavat ja palauttavat muokkauksen jälkeen. Filttereiden luominen on samankaltaista kuin toimintojen luominen sillä ne tarvitsevat myös luodun funktiot mitä kutsua koukkua suorittaessa.

2.7.2 Shortcode API

Shortcode API eli lyhytkoodiohjelmistorajapinta tarjoaa tavan lisätä sisältöä WordPress alustalla oleviin julkaisuihin ja sivuihin. Käyttäjät pystyvät lisäämään lyhytkoodeja WordPress sivun muokkausnäkymästä lisäämällä lyhytkoodikomponentin. Lyhytkoodikomponenttiin tulee kirjoittaa lyhytkoodin nimi hakasulkeiden sisälle. Hakasulkeiden sisälle voidaan myös syöttää attribuutteja, joiden avulla voidaan muokata lyhytkoodin toiminnallisuutta, jos lyhytkoodiin on määritelty niille toiminnallisuutta. Lyhytkoodeja pystytään myös asettamaan sisäkkäin, jos lyhytkoodi on kehitetty tukemaan tätä ominaisuutta. (WordPress Codex, n.d.)

2.7.3 HTTP API

HTTP on hypertekstin siirtoprotokolla ja se on koko internetin perusta missä asiakkaan lähettävät http-pyyntöjä, ja palvelimet vastaavat asiakkaiden http-pyyntöihin http-vastaukilla mitkä voivat olla esimerkiksi html-dokumentteja, JSON- tai XML-dataa. PHP-skriptauskieli tarjoaa monia eri tapoja tehdä http-pyyntöjä, mutta WordPress http-ohjelmistorajapinnan tarkoituksena helpottaa näiden käyttämistä ja tukea mahdollisimman monia näistä eri tavoista. (WordPress Developer Resources, 2023-c)

HTTP-protokollassa on monia eri metodeja mitkä kuvaavat tietyntyyppisiä toimintoja, joita käytetään http-pyyntöjen lähetyksessä kertomaan mitä toimintoja halutaan suorittaa. WordPress HTTP API tarjoaa toimintoja kolmelle yleisimmille metodeille GET, POST ja HEAD. (WordPress Developer Resources, 2023-c)

GET-metodia käytetään tiedonhakemiseen ja se on ylivoimaisesti yksi yleisimmin käytetyistä metodeista. Aina kun halutaan selata jotain sisältöverkossa kuten www-sivuja haetaan niiden sisältä GET-metodia käyttäen. POST-metodia taas käytetään datan lähettämiseen palvelimelle, kun halutaan sen toimivan tietyllä tavalla. Yleisin käyttötarkoitus POST-metodille on lomakkeiden datan lähettäminen palvelimelle, jotta lomakkeen data voidaan tallentaa esimerkiksi tietokantaan. HEAD-metodi ei ole yhtä käytetty kuin muut metodit ja se toimii samalla tavalla kuin GET-metodi, mutta datan hakemisen sijaan sillä haetaan tietoa vastaanotettavasta datasta. Selaimet käyttävät HEAD-metodia esimerkiksi tarkistamaan onko käyttäjän aikaisemmin vierailuilla sivuilla tapahtunut muutoksia, jonka seurauksena selain voi päätellä tarvitseeko se pyytä uudestaan sivun sisältä vai voiko se käyttää tallennettua välimuistissa olevaa versiota sivusta. Tämän avulla voidaan säästää kaistanleveyttä sillä joissain tapauksissa GET-metodien käyttäminen voi olla raskasta joidenkin resurssien sisältävän suuria määriä dataa. (WordPress Developer Resources, 2023-c)

2.7.4 REST API

WordPress REST-ohjelmistorajapinnan tarkoituksena on tarjota rajapinta, jonka avulla pystytään lähettämään ja vastaanottamaan JSON-dataa. REST-ohjelmistorajapinnan avulla pystytään parantamaan käyttäjäkokemusta WordPress-alustan sisällä tai siirtämään WordPress sisältöä toiselle alustalle. WordPress REST-ohjelmistorajapinta käyttää REST päätepisteitä mitkä esittävät WordPress-alustalla olevia objekteja kuten sivuja, julkaisuja ja muita sivustolla olevia datatyyppejä. Näihin päätepisteisiin pystytään lähettämään http-pyyntöjä minkä avulla pystytään tekemään esimerkiksi CRUD-operaatioita mitkä ovat luo, lue, päivitä ja poista. WordPress REST-rajapinta käyttää http-pyyntöjä ja -vastauksia kommunikaatioon, ja viestit ovat JSON-dataobjekteja. (WordPress Developer Resources, 2023-d)

2.7.5 Settings API

Settings API on WordPress-alustalla oleva tapa puoliautomaattisesti hallita asetuslomakkeita antamalla käyttäjille tavan luoda asetussivuja ja osioita, joiden sisällä on asetuskenttiä. Settings-ohjelmistorajanpinnan käyttämisen ansiosta lisäosan asetuksia pystytään helposti muokkaamaan WordPress-ylläpitonäkymästä. (WordPress Developer Resources, 2023-e)

2.7.6 Options API

Options API on WordPressin tarjoama tapa tallentaa asetuksia ja niihin liittävää tietoa WordPress-asetukset tietokantatauluun. Ohjelmistorajapinta mahdollistaa tietokantatauluun asetuksien luomisen, lukemisen, muokkaamisen ja poistamisen. Tietokantatauluun tallennetut asetuksen pystyvät olemaan kahta eri tyyppiä yksittäisiä arvoja tai tietotaulukkoja mitkä sisältävät avaimia ja arvoja. (WordPress Developer Resources, 2023-f)

2.7.7 WordPress-ohjelmointikäytännöt

WordPress-lisäosan kehityksessä tulee ottaa huomioon muuttujien, funktioiden ja luokkien nimeäminen, ettei muiden lisäosien kanssa tapahdu nimien päällekkäisyyksiä. Nimien päällekkäisyyksiä pystyy välttämään käyttämällä etuliitteitä, nimiavaruuksia tai tarkistamalla PHP-skriptauskielen funktioilla, joiden tarkoituksena on tarkistaa funktion olemassa oleminen. Näitä kaikkia tapoja välttää nimien päällekkäisyyksiä voi käyttää yhdessä tai vain yhtä niistä. (WordPress Developer Resources, 2023-g)

Tiedostorakenteessa suositellaan projektin päätiedoston sisältävän lisäosan nimen mukaan nimetyn PHP-tiedoston, vaihtoehtoisen poisto PHP-tiedoston ja loput tiedoston tulee sijoittaa alikansioihin. Kansioissa on myös hyvä olla indeksi PHP-tiedosto minkä takia pystytään välttää hakemiston sisällön paljastamista verkkoselaimella. Jokaisen tiedoston alussa myös suositellaan myös tarkistamaan ABSPATH niminen globaali muuttuja määritelty, jotta tiedostoihin ei päästä käsiksi muualta kuin lisäosan päätiedostosta. (WordPress Developer Resources, 2023-g)

2.8 VismaPay

VismaPay on Visman omistama verkkomaksupalvelu ja se tarjoaa verkkomaksu ohjelmistorajapinnan minkä avulla pystyy verkkokauppias helposti lisäämään tavan maksaa

tuotteista verkkosivuille. VismaPay myös tarjoaa turvalliset ja responsiiviset maksulomakkeet verkkokauppiaille. (Visma, n.d.)

2.9 PrestaShop

PrestaShop on avoimen lähdekoodin verkkokauppapalvelu, jonka hallinta on helppoa sen sisällönhallintajärjestelmällä. PrestaShop-verkkokauppaa on mahdollista laajentaa moduuleilla ja teemoilla. PrestaShopin ollessa avoimen lähdekoodin verkkokauppa on sitä mahdollista muokata HTML-merkintäkielellä, CSS-tyylittelykielellä ja PHP-skriptauskielellä verkkokaupan tarpeiden mukaan. (PrestaShop, n.d.)

PrestaShop mahdollistaa kauppiaiden antaa oikeuden PrestaShopin ulkoisille sovelluksille hallita kaupan tietokantaa sen verkkopalvelun kautta, joka on toteutettu käyttäen REST-ohjelmistorajapintaa. PrestaShop REST-ohjelmistorajapinnassa on mahdollista käyttää JSON- tai XML-dataa http-pyynnöissä ja http-vastauksissa. (PrestaShop Developer Documentation, 2023)

2.10 RockPos

RockPos on PrestaShop-verkkokaupalle oleva kassajärjestelmä, jonka avulla pystyy synkronoimaan paikalliset ostoksen verkkokaupan kanssa. RockPos-kassajärjestelmään on mahdollista lisätä monia eri maksutapoja ja järjestelmästä on mahdollista tulostaa personaalisoidun kuitin. (RockPos, n.d.)

3 Ajanvarausjärjestelmän suunnittelu ja toteutus

Ajanvarausjärjestelmän tarkoituksena on helpottaa toimeksiantajan toimintaa kotikäynteihin liittyen tarjoamalla toimeksiantajan asiakkaille tavan varata kotikäyntiaikoja heidän verkkosivuillaan. Toimeksiantajan nykyisten verkkosivujen ollessa toteutettu WordPress-julkaisualustalle toteutetaan ajanvarausjärjestelmä WordPress-lisäosana. WordPress-lisäosassa on mahdollista käyttää seuraavia verkkoteknologioita: PHP, JavaScript, MariaDB, CSS ja HTML. WordPress-lisäosassa ohjeistetaan, että lisäosan toteutuksessa käytetään WordPress-julkaisualustan varmistamia kirjastoja ja resursseja, jonka takia on työhön valikoitunut teoriaosassa kuvatut teknologiat ja työkalut.

Projektin kehitysympäristönä käytetään XAMPP-sovellusta sen tarjotessa WordPress-julkaisualustan vaatimat palvelut: Apache ja MariaDB. XAMPP-sovelluksen kaltaisia sovelluksia on monia, joita voisi käyttää XAMPP-sovelluksen sijaan, mutta XAMPP valikoitui kehitysympäristöksi sen ollessa ennestään tuttu.

Projektin rakenne perustuu WordPress-lisäosan koodauskäytännön kehottamaan rakenteeseen, jonka mukaan lisäosan tiedostorakenne koostuu lisäosan päätiedostosta, vaihtoehtoisesta lisäosan poisto tiedostosta ja alikansioista, jotka voivat sisältää ohjelmistokoodia tai muuta lisäosan vaatimia resursseja. WordPress-lisäosan koodauskäytännön seuraaminen auttaa muita kehittäjiä ymmärtämään paremmin projektin sisältöä, jos tulevaisuudessa sitä halutaan laajentaa. Projektissa myös pyritään seuraamaan muita WordPress-lisäosan koodauskäytäntöjä, jotta koodi saadaan pysymään selkeänä muille kehittäjille, jos toimeksiantaja haluaa itse myöhemmin muokata koodin toimintaa.

Ajanvarausjärjestelmä koostuu seuraavista komponenteista: lyhytkoodista, REST-päätepisteistä, ylläpitonäkymästä, ylläpitoasetuksista, VismaPay PHP-kirjastosta ja PrestaShop WebService-kirjastosta. Ajanvarauslomake on toteutettuna lyhytkoodina toimeksiantajan toiveesta, koska lyhytkoodin on mahdollista sijoittaa mille tahansa WordPress-sivulle.

Projektissa käytetään REST-päätepisteitä, jotta ajanvarauslomakkeesta saadaan mukavampi ja sujuvampi käyttää sillä voidaan REST-päätepisteisä hakea vapaita aikoja asiakkaille heidän valitsemansa päivän käyttäen JavaScriptin FETCH-ohjelmistorajapintaa. FETCH-ohjelmistorajapinnan avulla saadaan lomakkeen saatavilla olevat päivittymään reaaliajassa käyttäjän vaihtaessa ajanvarauksen päivää. REST-päätepisteiden sijaan olisi projektissa voinut käyttää AJAX-ohjelmistorajapintaa minkä toiminnallisuus on samankaltainen kuin REST-ohjelmistorajanpinnan, mutta se ei kuitenkaan tarjoa kaikkia hyödyllisiä ominaisuuksia mitä REST-ohjelmistorajapinta tarjoaa.

REST-päätepisteitä voidaan myös käyttää muualla projektissa, kuten ylläpitonäkymä tai hakea niistä tietoa ulkoisella sovelluksella. REST-päätepisteissä tulee kuitenkin olla tarkka niiden käyttöoikeuksista, ettei käyttäjät ilman käyttöoikeuksia olevat käyttäjät pääse käsiksi tietoihin mihin heillä ei ole oikeuksia. Tämän toteuttaminen on kuitenkin onneksi yksinkertaista, sillä WordPress REST-ohjelmistorajapinnassa on tätä varten määriteltyjä funktioita. REST-päätepisteissä on hyödynnetty WP_REST_Controller-luokkaa, jonka tarkoituksena on helpottaa CRUD-toimintoja käyttävien päätepisteiden toteuttamista. Kaikkia

päätepisteitä ei ole kuitenkaan toteutettu laajentamalla WP_REST_Controller-luokkaa sillä kaikki päätepisteet eivät vaadi kaikkia CRUD-toimintoja.

Toteutusosiossa olevissa ohjelmistokoodeissa käytetään esimerkkejä oikean projektin koodin sijaan. Esimerkit ovat yksinkertaistettuja versioita oikeasta ohjelmasta, jotta työssä oleva koodi saadaan pidettyä lyhyenä.

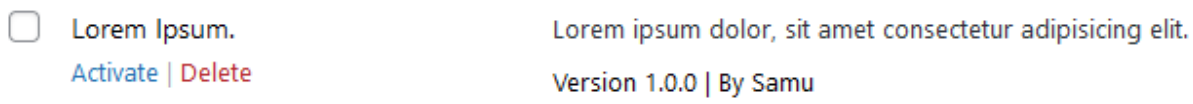
3.1 WordPress-lisäosan aktivointi, deaktivointi ja poistaminen

WordPress-lisäosa on vähintään PHP-tiedosto mikä sisältää tiedoston alussa kommentin mihin on määritelty vähintään lisäosan nimi. Lisäosan-tiedoston tulee sijaita WordPress pää tiedoston sisällä wp-content/plugins kansiossa. WordPress kuitenkin suosittelee jokaisen lisäosan olevan omassa kansiossaan, jotta lisäosan tiedostojen nimet, alikansion ja muut sen resurssit eivät mene päällekkäin ja projektin rakenne on selkeämpi.

WordPress etsii lisäosan päätiedoston sen alkuun lisätyn kommentin avulla, jonka takia on sen lisääminen pakollista lisäosan aktivointia varten. Kommenttiin on mahdollista määrittää muitakin parametreja kuin aikaisemmin mainittu lisäosan nimi ja kommentin parametrit näytetään WordPress lisäosa näkymässä (Kuva 1).

```
/**
 * Plugin name: Lorem Ipsum.
 * Description: Lorem ipsum dolor, sit amet consectetur adipisicing elit.
 * Author: Samu
 * Version: 1.0.0
 */
```

Kuva 1. Kommentin parametrin lisäosa näkymässä.



Lisäosan aktivointiin, deaktivointiin ja poistamiseen on määritelty omat funktiot, joiden avulla pystytään toimintojen aikana suorittaa haluttua toiminnallisuutta. Lisäosan aktivoinnissa luodaan lisäosan omia tietokantatauluja ja asetuksien ja lisäosan deaktivoinnissa poistetaan väliaikaisia tiedostoja kuten välimuistiin tallennettuja arvoja. Lisäosan poistamisessa taas poistetaan lopullisesti lisäosan luomia resursseja. Lisäosan poistamisessa voi

vaihtoehtoisesti luoda lisäosan pääkansioon PHP-tiedoston nimeltä uninstall.php ja WordPress osaa automaattisesti käyttää tiedostoa lisäosan poistamisessa.

Ajanvarausjärjestelmässä lisäosan aktivoinnissa luodaan asetukset minkä avulla pystytään hallitsemaan järjestelmän aukioloaikaa, sulkeutumisaikaa, aikavyöhykettä, ajanvarauksen pituutta. Aktivoinnissa myös luodaan lisäosan toimintaa vaaditut tietokantataulut. Ajanvarausjärjestelmän lisäosan deaktivoinnissa ei tehdä mitään ainakaan sen nykyisessä versiossa sillä se ei tallenneta mitään väliaikaista tietoa. Lisäosan poistamisessa vaihtoehtoisesti poistetaan lisäosan luomat tiedot sillä kotikäyntiaikoja ei välttämättä haluta poistaa, vaikka lisäosa itsessään halutaan poistaa WordPress-alustalta. Lisäosan poiston yhteydessä on mahdollista poistaa sen luomat tietokantataulut, jos käyttäjä näin haluaa tehdä. Poistamista varten on lisäosaan määritelty asetus, joka on totuusarvomuuttuja ja sen tarkoituksena on kertoa, poistetaanko tietokantataulut.

3.2 Lisäosan pääluokka

Lisäosan pääluokan tehtävänä on yhdistää kaikki lisäosan komponentit keskenään. Lisäosan pääluokassa on yksi julkinen funktio, jonka tarkoituksena on ladata lisäosan vaatimat resurssit, lisästä ylläpitäjä- ja julkisetkoukut.

Lisäosan resurssit ladataan ja aktivoidaan konditionaalisesti kuten WordPress-lisäosan ohjelmointikäytännöissä määritellään. Konditionaalinen lataus kuitenkin lisäosassa on minimaalista, sillä ladataan lisäosassa resurssit käyttäjän nykyisen näkymän mukaan. Tämä tarkoittaa ajanvarausjärjestelmän kohdalla sitä, että käyttäjän ollessa ylläpitonäkymässä ladataan ylläpitonäkymään vaaditut resurssit ja käyttäjän ollessa ajanvarauslomakkeessa ladataan sen toimintaan vaaditut resurssit. Tämä on mahdollista WordPress-alustalla olevalla is_admin funktiolla mikä palauttaa totuusarvon, joka kertoo käyttäjän olevan ylläpitonäkymässä tai ei.

Lisäosan ylläpitokoukuissa lisätään ajanvarausjärjestelmän aikojen hallintasivu ja järjestelmän asetukset sivu. Lisäosan julkisissa koukuissa lisätään ajanvarauslomake lyhytkoodi.

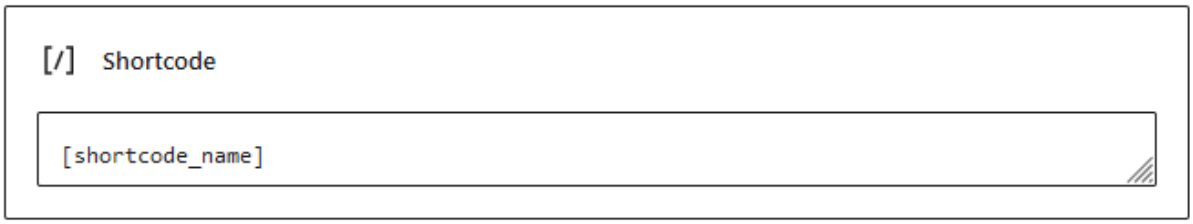
```
if (is_admin()) {
   register_admin_hooks();
} else {
   register_public_hooks();
}
```

3.3 Ajanvarauslomake

WordPress-alustalle on mahdollista lisätä lyhytkoodeja sen tarjoaman lyhytkoodiohjelmistorajapinnan avulla. Ohjelmistorajapinta sisältää funktiot lyhytkoodien suorittamiseen ja lisäämiseen. Lyhytkoodin lisäysfunktiossa (Ohjelmistokoodi 3) vaaditaan kaksi parametria: lyhytkoodin nimi ja funktio. Lyhytkoodin nimeä voidaan käyttää lyhytkoodin suorittamiseen koodissa tai sivunmuokkausnäkymässä (Kuva 2) lisäämällä lyhytkoodi elementin sivustolle.

```
add_shortcode("shortcode_name", "shortcode_function");
do_shortcode("shortcode_name");
```

Kuva 2. Lyhytkoodin lisääminen sivunmuokkausnäkymässä.



Ajanvarauslomake on toteutettu lyhytkoodina käyttäen malleja, joiden tarkoituksena vähentää koodia lyhytkoodin funktion sisällä. Malli on PHP-tiedosto mikä sisältää HTML-merkintäkielellä luotua sisältöä, jonka tarkoituksena on kuvata lyhytkoodin sisältöä. Mallin ollessa PHP-tiedosto on sen sisälle mahdollista upottaa PHP-koodia HTML-elementtien sekaan, jotta voidaan sivustolle generoida dynaamisesti elementtejä. WordPress-alusta tarjoaa myös tavan lisätä nykyisen WordPress-sivun head elementtiin JavaScript-tiedostoja ja CSS-tiedostoja.

JavaScript koodia on mahdollista lisätä WordPress-alustan verkkosivulle käyttämällä sille tarkoitettua funktiota. Funktion parametreissa vaaditaan skriptille uniikki nimi ja URL-osoite. Vaihtoehtoisesti voi funktiossa lisättävälle skriptille määritellä riippuvuuksia, versio ja onko lisätyn skriptin sijainti footer-elementissä.

```
wp_enqueue_script("script_name", $url, $deps, $ver, $in_footer);
```

CSS-tyylittelykieltä on mahdollista lisätä samalla tavalla kuin JavaScript-koodia. CSS-tyylittelykielen ja JavaScript-koodin lisäämisfunktiossa on kuitenkin erona niiden viimeinen

parametri. CSS-funktiossa viimeiseen parametriin määritellään head tai footer sijainnin sijaan missä media tilassa resurssia käytetään.

```
wp_enqueue_style("style_name", $url, $deps, $ver, $media);
```

Projektissa oleva JavaScript-koodi käyttää REST-päätepisteitä ja tämän takia täytyy JavaScript-koodille kertoa päätepisteiden osoitteet. Tämä on mahdollista käyttäen WordPress funktiota, jonka avulla pysytään lisäämään olemassa olevaan JavaScript-koodiin PHP-taulukoita JavaScript-objekteina. Funktio vaatii parametreina olemassa olevan JavaScript-koodin nimen, JavaScript objektin nimen ja PHP-taulukon.

```
wp_localize_script("script_name", "JS_Object", $data);
```

Koodin lisäämässä JavaScript-objektissa on määriteltynä ajanvarausjärjestelmän REST-nimiavaruuden osoite ja nonce, jonka tarkoituksena on suojella WordPress verkkosivuja CSRF-hyökkäyksiltä. JavaScript koodissa haetaan saatavilla olevat ajat niille luodusta REST-päätepisteestä ja lomake lähetään toiseen REST-päätepisteeseen. Pyyntöjen lähettäminen tapahtuu käyttäen FETCH-rajapinnassa olevaa funktiota. Funktiossa määritellään parametreissa URL-osoite ja http-pyyntö objekti.

```
await fetch(url, request)
```

Saatavilla olevien aikojen tapauksessa lähetään REST-päätepisteeseen GET-pyyntö, minkä URL-osoitteeseen lisätään haluttu päivämäärä. Päätepiste vastaa JavaScript-ohjelmalle palauttamalla listan vapaana olevista ajoista päivälle tai virheenä. Vastauksen jälkeen JavaScript-ohjelma näyttää ajanvarauslomakkeessa vapaana olevat ajat tai vastauksen ollessa virhe näytetään lomakkeessa, ettei vapaita aikoja päivälle löydetty.

Vapaana olevien aikojen hakemisfunktio kutsutaan, kun käyttäjä tekee muutoksia lomakkeessa olevaan HTML-elementtiin. Tämä on mahdollista lisäämällä tapahtumankuuntelijoita, joiden tarkoituksena on kutsua niihin lisätty funktio, kun jokin verkkosivuilla määritelty tapahtuma tapahtuu.

```
addEventListener("change", availableTimes)
```

JavaScript-ohjelma on vastuussa lomakkeen lähettämisestä ajanvarausjärjestelmän REST-päätepisteeseen missä lomake myös tarkistetaan. REST-päätepisteestä tulee onnistunut- tai

virhevastaus minkä perusteella ohjelma näyttää käyttäjälle virheet tai uudelleenohjaa käyttäjän VismaPay-verkkomaksupääte sivuille.

3.4 REST-päätepisteet

Ajanvarausjärjestelmässä on kolme REST-päätepistettä mitkä ovat: ajanvaraukset, saatavilla olevat ajat ja maksu. Ajanvarauksetpäätepiste laajentaa WP_REST_Controller-luokan, joka on abstrakti luokka, joka tarjoaa pohjan CRUD-toimintoja käyttävälle päätepisteelle. Aika- ja maksupäätepisteet eivät laajenna WP_REST_Controller-luokaa sillä niiden toiminnallisuus ei vaadi kaikkia CRUD-toimintoja.

REST-päätepisteiden lisääminen WordPress-alustalla tapahtuu rest_api_init toimintakoukun suorituksessa. Toimintakoukku kutsutaan aina, kun WordPress vastaanottaa http-pyynnön, jonka ansiosta ladataan REST-päätepisteet vain silloin kuin niitä tarvitaan. REST-päätepisteiden lisääminen tapahtuu register_rest_route funktiolla mihin määritelellään päätepisteen nimiavaruus, polku, aseukset ja JSON-skeema.

```
register_rest_route("namespace", "/route", array(
    array(
        "methods" => "GET",
        "callback" => "callback_function",
        "permission_callback" => "permission_callback_function"
    ),
    "schema" => "get_public_item_schema"
));
```

3.4.1 Ajat REST-päätepiste

Ajat-päätepisteen tarkoituksena on hakea ajanvarauslomakkeelle saatavilla olevat kotikäynti ajat. Päätepiste olettaa http-pyynnön URL-osoitteessa olevan päivän miltä halutaan tietävän saatavilla olevat ajat. URL-osoitteessa olevaa päivää käytetään tietokannasta varattujen päivien hakemiseen, tarkistamaan onko päivä menneisyydessä vai tulevaisuudessa ja onko päivä vapaapäivä. Vapaapäivät haetaan ulkoisesta REST-ohjelmistorajapinnasta, joka tarjoaa suomen vapaapäivät. Päivän ollessa kelvollinen selvitetään kaikki mahdolliset ajat ajanvarausjärjestelmän asetuksien ja kotikäyntivarauksien mukaan. Päivien selvityksen jälkeen palautetaan saatavilla olevat päivät listana. Jos jossain kohtaa ohjelmaa tapahtuu virhe tai kaikki mahdolliset ajat ovat varattu palautetaan http-virhe, joka kertoo pyynnön lähettäjälle, ettei saatavilla olevia aikoja löydetty.

3.4.2 Ajanvaraukset REST-päätepiste

Ajanvaraukset-päätepiste laajentaa WP_REST_Controller-luokan sen toiminnan vaatiessa kaikki CRUD-toiminnot. Päätepisteessä on siis mahdollista luoda, lukea, muokata ja poistaa ajanvarauksia. Päätepiste on piilotettu ja vaatii ylläpitäjän käyttöoikeudet sillä ajanvaraukset sisältävät arkaluontoista tietoa mitä ei haluta antaa ulkopuolisten käsiksi. Päätepisteen ollessa piilotettu sitä ei ole mahdollista löytää WordPress REST-päätepisteistä eikä sen polkuja tai JSON-skeemaa paljasteta.

3.4.3 Maksu REST-päätepiste

Maksu-päätepiste on päätepiste mihin ajanvarauslomake lähettää http-pyynnön. Päätepisteen käyttämiseen vaaditaan, että pyynnön mukana lähetään ajanvarauksen tiedot ja nonce. Maksu-päätepiste kommunikoi VismaPay ja PrestaShop palveluiden kanssa käyttäen palveluille kehitettyjä PHP-kirjastoja.

Päätepisteessä hyödynnetään ajanvaraukset-päätepistettä luomiseen, lukemiseen ja muokkaamiseen. Tämä on mahdollista WordPress-alustan sisäisillä http-pyynnöillä. Pyyntöjen käyttämisessä täytyy kuitenkin olla varovainen sillä ne eivät tarkista käyttöoikeuksia. Ajanvaraukset-päätepisteen ansiosta voidaan tarkistaa ajanvarauksen kelvollisuus sen JSON-skeeman ansiosta ja maksupäätepisteen vaatimaa toiminnallisuutta ei tarvitse kehittää uudelleen.

Ajanvarauksen luomisen jälkeen päätepisteessä lähetään VismaPay-maksupäätteeseen http-pyyntö. Pyynnön tarkoituksena on hakea maksutoken, jonka avulla voidaan generoida maksuosoite. Pyyntö lähetään käyttäen VismaPay PHP-kirjastoa mihin on valmiiksi määritelty cURL-kirjastoa käyttäen VismaPay-päätepisteiden pyynnöt. Virheelliset pyynnön aiheuttavat VismaPay-poikkeuksen, joka täytyy käsitellä try-catch-rakenteella. Maksun luomisessa VismaPay-palveluun vaaditaan maksun tiedot, asiakkaan tiedot, tuotteen tiedot, paluu URL-osoite ja ilmoitus URL-osoite.

```
{
    "version": "w3.1",
    "api_key": "api_key",
    "order_number": "order_number",
    "amount": 1000,
    "currency": "EUR",
    "payment_method": {},
```

```
    "authcode": "authcode",
    "customer": {},
    "products": []
}
```

VismaPay-palvelu vastaa palauttamalla JSON-dataobjektin mikä sisältää tulosnumeron, tokenin, maksutavan tai virheet. Tulosnumerot kertovat oliko maksun luominen onnistunut vai virheellinen.

```
{
    "result": 0,
    "token": "token",
    "type": "e-payment"
}
```

Tokenilla generoidaan maksuosoite, johon käyttäjä uudelleenohjataan, jos maksun luominen onnistui VismaPay-palveluun. VismaPay-maksusivulla asiakas voi valita haluamansa maksutavat. Maksun suorituksen jälkeen VismaPay-palvelu uudelleen ohjaa asiakkaan paluuosoitteeseen mikä määritellään http-pyynnössä. Samaan osoitteeseen palataan maksun ollessa onnistunut, keskeytetty tai virheellinen. VismaPay-palvelu lisää osoitteeseen parametrit: paluukoodi, tilausnumero, autentikointikoodi ja riippuen maksun tilasta parametrina voi olla maksun ratkaisukoodi tai tapahtumakoodi. URL-parametrien ansiosta voidaan tarkistaa maksuntila ja onko maksu oikea syöttämällä URL-parametrit VismaPay PHP-kirjaston tarjoamaan tarkistus funktioon.

Ajanvarausjärjestelmän luomassa VismaPay-maksussa määritellään paluuosoitteeksi maksu REST-päätepiste, joka tarkistaa maksun ja päivittää ajanvarauksen maksun tilan tietokantaan. Maksutilan päivittämisen jälkeen käyttäjä uudelleenohjataan takaisin WordPress-verkkosivuille.

Maksupäätepisteessä luodaan VismaPay-palvelun onnistuneen maksun palautuksen yhteydessä tilaus PrestaShop-verkkokauppaan käyttäen PrestaShop-verkkopalvelu PHP-kirjastoa. PrestaShop-verkkopalvelu PHP-kirjasto sisältää CRUD-funktiot, joiden avulla voidaan hallita verkkokaupan resursseja.

PrestaShop-verkkopalvelu PHP-kirjastossa CRUD-funktiot vastaan ottavat yhden parametrin, joka on taulukko mihin määritellään pyyntöön liittyviä parametreja. Esimerkiksi get-funktion avulla voidaan hakea resurssien XML-skeemoja mitä käytetään post-, edit-funktioiden lähettämisessä, että XML-datan struktuuri saadaan pidettyä verkkopalvelun olettamassa

muodossa. PrestaShop-verkkopalvelu PHP-kirjaston CRUD-funktiot palauttavat SimpleXMLElement-luokan tai aiheuttavat poikkeuksen, joka käsitellään try-catch-rakenteella. SimpleXMLElement-luokkaan on mahdollista määritellä dynaamisesti arvoja samalla tavalla kuin stdClass-luokkaan.

```
try {
    $xml = $web_service->get(array(
        "resource" => "carts",
        "schema" => "blank"
    ));

    $xml->cart->id_customer = 0;
    ...

    $xml = $web_service->post(array(
        "resource" => "carts",
        "postXML" => $xml->asXML()
    ));

} catch (PrestaShopWebserviceException $e) {
    ...
}
```

Kuittien luomista varten Ajanvarausjärjestelmä luo PrestaShop-verkkokauppaan seuraavat resurssit: ostoskärryn, tilauksen, asiakasviestiketjun ja asiakasviestin. Ostoskärry on vaadittu resurssi tilauksen luomista varten. Ostoskärryresurssiin lisätään sen luomisen yhteydessä PrestaShop-verkkokaupassa olevat kotikäyntituote, jonka tunniste on tallennettuna lisäosan asetuksiin. Tilauksen luomiseen vaaditaan myös asiakas-, osoite-, valuutta-, kieli- ja toimitusresursseja, jotka ovat määriteltävissä ajanvarausjärjestelmän asetuksissa. Syynä ratkaisuun on tunnisteiden ollessa vaadittuja muiden resurssien luomiseen ja niiden pysyvän samana jokaisessa luodussa tilauksessa. RockPos-kassajärjestelmässä käytetään samankaltaista menetelmää paikallisten tilauksien luomisessa. Tämä tarkoittaa sitä, että PrestaShop-verkkokauppaan on luotava asiakas ja myyjä mitä ajanvarausjärjestelmä käyttää ja kaikki ajanvaraukset lisätään luodulle asiakkaalle ja myyjälle. Ajanvarausjärjestelmän tilauksen luomisessa kuitenkin lisätään tilaukseen asiakasviesti missä kerrotaan ajanvarauslomakkeeseen täydennetyt tiedot. Tämä on tärkeätä, sillä RockPos-kassajärjestelmä tulostaa näkyvillä olevat asiakasviestit kuittiin. Tämä mahdollistaa ajanvarauksen tietojen olevan näkyvillä tulostetussa kuitissa.

4 Pohdinta

Työn tavoitteena oli toteuttaa ajanvarausjärjestelmä, jolla toimeksiantajan asiakkaan pystyvät varaamaan ja maksamaan kotikäyntiaikoja. Ajanvarausjärjestelmä tuo hyötyä toimeksiantajalle ja heidän asiakkailleen helpottamalla kotikäynnin ajanvaraus prosessia. Ajanvarausjärjestelmän ansiosta ajanvarauksien käsittely on yksinkertaisempaa sillä ohjelma luo aikaisemmin manuaalisesti luodun kuitin automaattisesti PrestaShop-verkkokauppaan.

Ennen työn aloittamista toimeksiantajalla ei ollut ajanvarausjärjestelmää asiakkaiden kotikäyntien varaamiseen ja maksamiseen, jonka takia työn toteuttaminen aloitettiin. Alun perin toteutuksen tarkoituksena oli olla kotikäyntiajan pyyntölomake mikä kehittyi mahdollistamaan ajanvarauksen maksamisen ja kuitin luomiseen toimeksiantajan järjestelmään.

Suunniteltu ajanvarausjärjestelmä toteutettiin WordPress-julkaisualustalle ja sen tarjoamia ohjelmistorajapintoja käyttäen. Ajanvarausjärjestelmässä on käytetty kahta ulkopuolista PHP-kirjastoa yksinkertaistamaan VismaPay-verkkomaksupäätteen ja PrestaShop-verkkopalvelun käyttämistä. Työn käyttöliittymien rakentamisessa on käytetty ainoastaan HTML-merkintäkieltä, CSS-tyylittelykieltä ja JavaScript-skriptauskieltä eikä niille olevia kirjastoja käyttöliittymien ollessa yksinkertaisia. Työn käyttöliittymien kehityksessä olisi kuitenkin ollut mahdollista käyttää kirjastoja, mutta ne olisivat lisänneet turhaa monimutkaisuutta työhön.

Ajanvarausjärjestelmässä käytetyt teknologiat valkoituivat toimeksiantajan pyynnöstä ja PHP-kirjastot valikoituivat projektiin niiden ollessa VismaPay-verkkomaksupäätteen ja PrestaShop-verkkopalveluiden suosittelemia kirjastoja heidän palveluiden käyttämiseen. Projektissa alun perin käytettiin WordPress AJAX-ohjelmistorajapintaa, mutta se vaihtui nopeasti REST-ohjelmistorajapintaa AJAX-ohjelmistorajapinnan käyttämisen muuttuessa sekavammaksi ja REST-ohjelmistorajapinnan ollessa selkeämpi käyttää ja täyttäen paremmin projektin vaatimukset.

Ajanvarausjärjestelmä koostuu pääsääntöisesti WordPress REST-päätepisteistä, joiden avulla järjestelmässä on mahdollista hakea saatavilla olevia aikoja, hallita ajanvarauksia CRUD-toiminnoilla ja maksaa ajanvarauksia. REST-ohjelmistorajapinnan käyttäminen on nykypäivänä yleistä, sillä sen avulla järjestelmän ulkoiset sovelluksen pystyvät kommunikoimaan järjestelmässä olevien REST-päätepisteiden kanssa. REST-päätepisteiden ansiosta projektin käyttöliittymä pystytään ulkoistamaan WordPress-julkaisualustasta, jos haluttaisiin käyttöliittymä rakentaa esimerkiksi käyttäen Svelte- tai

React-ohjelmistokehystä. Nykyisessä toteutuksessa kuitenkin jouduttaisiin tekemään muutamia muutoksia sen ulkoistamiseen, mutta ei se olisi mahdotonta työn nykyisellä rakenteella. REST-päätepisteiden ansiosta projektiin ei tarvinnut kehittää erillisiä luokkia käyttäjän ja ylläpitäjän vaatimiin toiminnallisuuksiin sillä molemmat pystyvät tekemään pyyntöjä samoihin päätepisteisiin, jos heidän käyttöoikeutensa sen sallivat.

Työssä haasteeksi osoittautui PrestaShop-verkkopalvelun käyttäminen, sillä PrestaShop-verkkopalvelun resurssien dokumentaatio on minimaalinen ja tästä syystä resurssien vaatimien kenttien selvittämiksi jouduin tutkimaan PrestaShop-verkkokaupan tietokantatauluja saadakseni paremman käsityksen resurssien luomiseen. Itse verkkopalvelun käyttäminen on hyvin dokumentoituna ja kaikki tarvittava tieto sen käyttämisestä löytyy dokumentaatiosta. Työnhalutun toiminnallisuuden suorittamiseen oli PrestaShop-tietokannan ja -käyttöliittymän tutkiminen tarpeellista, sillä kaikkia ominaisuuksia ei ole dokumentoituna.

Ongelmaksi myös PrestaShop-verkkopalvelussa koitui kehitysympäristössä käytetyn PrestaShop-version virheelliset XML-skeemat, jotka aiheuttivat verkkopalvelun kaatumisen skeemojen hakemisessa. Tämän ei pitäisi olla ongelma uudemmassa PrestaShop-versiossa.

Opinnäytetyöprosessin aikana opin enemmän WordPress-lisäosien kehittämisestä, vaikka olen aikaisemminkin kehittänyt WordPress-lisäosia, mutta en samankaltaisessa skaalassa kuin ajanvarausjärjestelmä. Jouduin opinnäytetyöprosessissa perehtymään VismaPay-verkkomaksupäätteeseen ja PrestaShop-verkkokauppaan mihin minulla ei ole aikaisempaa kokemusta.

Lähteet

Apache Friends. (2023). XAMPP Installers and Downloads for Apache Friends. https://www.apachefriends.org/

Apache HTTP Server Project. (2023). Welcome! The Apache HTTP Server Project. https://httpd.apache.org/

ECMAScript 2024 Language Specification. (2023). ECMAScript 2024 Language Specification. Overview. https://tc39.es/ecma262/#sec-overview

HTML Standard. (2023). HTML Standard. A quick introduction to HTML. https://html.spec.whatwg.org/multipage/introduction.html#a-quick-introduction-to-html

MariaDB.org. (2020). Introduction to Relational Databases. MariaDB Knowledge Base. https://mariadb.com/kb/en/introduction-to-relational-databases/

MariaDB.org. (n.d.). Documentation. https://mariadb.org/documentation/

MDN Web docs. (2023-a). Using the Fetch API. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

MDN Web Docs. (2023-b). Working with JSON. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON

MDN Web Docs. (2023-c). XML introduction. https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction

MDN Web Docs. (2023-d). What is CSS?. https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS

PHP. (n.d.). What is PHP? https://www.php.net/manual/en/intro-whatis.php

PrestaShop Developer Documentation. (2023). Getting Started. https://devdocs.prestashop-project.org/8/webservice/getting-started/

PrestaShop. (n.d.). Ecommerce platform. https://prestashop.com/ecommerce-platform/

RockPos. (n.d.). Point of sale system for PrestaShop. https://rockpos.com/

Visma. (n.d.). Verkkomaksupalvelu Visma Pay. https://www.visma.fi/vismapay/

Visual Studio Code. (n.d.). Visual Studio Code. https://code.visualstudio.com/

WordPress Codex. (n.d.). Shortcode API. https://codex.wordpress.org/Shortcode_API

WordPress Developer Resources. (2023. -a). What is a Plugin? https://developer.wordpress.org/plugins/intro/what-is-a-plugin/

Wordpress Developer Resources. (2023-b). What is a Theme? https://developer.wordpress.org/themes/getting-started/what-is-a-theme/

WordPress Developer Resources. (2023-c). HTTP API. https://developer.wordpress.org/plugins/http-api/

WordPress Developer Resources. (2023-d). REST API Handbook. https://developer.wordpress.org/rest-api/

WordPress Developer Resources. (2023-e). Settings API. https://developer.wordpress.org/plugins/settings/settings-api/

WordPress Developer Resources. (2023-f). Options API. https://developer.wordpress.org/plugins/settings/options-api/

WordPress Developer Resources. (2023-g). Best Practices. https://developer.wordpress.org/plugins/plugin-basics/best-practices/

Wordpress. (n.d.). About. https://wordpress.org/about/