VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Huhtala Toni

Metsola Niklas

WORDPRESS-LIITÄNNÄINEN SOVELLUK-SIEN HINTAKARTOITUKSEEN

Liiketalous
2022

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Toni Huhtala, Niklas Metsola
Opinnäytetyön nimi	WordPress-liitännäinen sovelluksien hintakartoitukseen
Vuosi	2022
Kieli	suomi
Sivumäärä	43
Ohjaaja	Antti Mäkitalo

Tämän opinnäytetyön tarkoituksena oli toteuttaa WordPress-liitännäinen, jonka avulla käyttäjän oli mahdollista saada hinta-arvio sovellusideastaan. Käyttäjältä kysyttiin sovellukseen liittyviä kysymyksiä, joiden avulla hinta laskettiin. Tavoitteena oli kehittää yksinkertainen sovellus, joka laskisi kynnystä lähteä toteuttamaan omaa sovellusideaa.

Sovellus toteutettiin useamman eri ohjelmointikielen avulla. Backend koostui pääasiassa PHP:sta ja Frontend JavaScriptistä sekä HTML:stä. Projekti oli rakennettu Dockerin sisälle, josta myös tietokanta löytyi. Tietokantana toimi MySQL.

Opinnäytetyön aikana saatiin toteutettua toimiva WordPress-liitännäinen, jonka kriittisimmät vaatimukset täyttyivät. Liitännäisen avulla pystyttiin arvioimaan onnistuneesti käyttäjän sovellusideaa. Koska työ piti sisällään paljon uutta asiaa, oli se mielenkiintoinen toteuttaa oppimisen näkökulmasta.

Avainsanat liitännäinen, WordPress, Docker, ohjelmointi

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Toni Huhtala, Niklas Metsola
Title	WordPress plugin for application price mapping
Year	2022
Language	Finnish
Pages	43
Name of Supervisor	Antti Mäkitalo

The goal of this thesis was to create a WordPress plugin that would allow users to get an estimated price for their application data. The user was asked questions about the application which were then used to count the price. The goal was to develop a simple application that would lower the threshold to implement application ideas.

The application was made using multiple programming languages. The Backend was made with PHP and Frontend with JavaScript and HTML. The Project was built inside Docker, where the database also was. MySQL served as a database.

During the making of the thesis a working WordPress plugin was made that met all the set critical requirements. With the help of the plugin, it was possible to successfully evaluate the user's application idea. Although the work contained lots of new things, it was interesting to implement from a learning perspective.

Keywords plugin, WordPress, Docker, programming

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

KÄSITTEET

Kontti on koodin sekä riippuvuuksia pakkaava ohjelmistoyksikkö, joka mahdollistaa sovelluksen nopean ja luotettavan toiminnan ympäristöstä toiseen. (Docker 2022.)

Actor määrittelee käyttäjän roolin tai minkä tahansa muun järjestelmän, joka on vuorovaikutuksessa kohteen kanssa. Actorit voivat olla ihmisiä, muita järjestelmiä tai aika- ja tapahtumalaukaisimia. (Visual Paradigm 2022)

Frontend tarkoittaa verkkosivun tai sovelluksen käyttäjälle näkyvää osaa, jonka avulla käyttäjä on vuorovaikutuksessa niiden kanssa. (ite wiki 2022.)

Backend tarkoittaa verkkosivun tai sovelluksen palvelinpuolta. Palvelinpuoleen kuuluu taustajärjestelmät, tietokannat sekä toimintalogiikka. (ite wiki 2022.)

HTML on verkkosivujen runko. Kuitenkin lähes aina käytetään joitain aputeknologioita parantamaan sivuston interaktiivisuutta. (freecodecamp 2021.)

CSS on tyyliohje, jonka avulla kuvataan HTML- tai XML-kielillä kirjoitettujen tiedostojen ulkoasua. (MDN Web Docs 2022.)

JavaScript on ohjelmistokieli, jonka avulla voidaan toteuttaa monimutkaisia ominaisuuksia verkkosivuilla. (MDN Web Docs 2022.)

jQuery on nopea ja monipuolinen JavaScript-kirjasto. Se tekee HTML-dokumenttien käsittelystä huomattavasti nopeampaa ja helppokäyttöisempää. (jQuery 2022.)

PHP on laajasti käytetty komentosarjakieli. PHP soveltuu erityisesti verkkokehitykseen ja se voidaan liittää osaksi HTML:ää helposti. (PHP 2022.)

1 JOHDANTO

Opinnäytetyö toteutetaan toimeksiantona paikalliselle IT-yritykselle ja se kirjoitetaan päiväkirjan muodossa. Päiväkirjassa käydään läpi viikoittain opinnäytetyön kulkua ja analysoidaan viikkojen tuloksia viikkoanalyysin avulla. Opinnäytetyö toteutetaan parityönä.

Työn tarkoituksena on luoda verkkosivuille työkalu, jonka avulla sivustolla vierailleva asiakas saa hinta-arvion sovellusidealleen. Hinta lasketaan käyttäjän syöttämien vastauksien perusteella. Työkalu on WordPress liitännäinen, jotta se voidaan vaivattomasti ottaa käyttöön verkkosivuille.

Työn toteuttamiseksi tulee luoda ohjelmointiympäristö sekä hankkia erilaisia ohjelmistoja. Lisäksi työ vaatii useamman ohjelmointikielen hallintaa ja loogista päättelykykyä. Työssä hyödynnetään aikaisemmin opittuja ohjelmointikieliä, kuten HTML, CSS, JavaScript, jQuery sekä PHP. Työ vaatii näiden ohjelmointikielien syventämistä sekä uusien ohjelmistojen opettelua. Projektin pääohjelmistoja ovat Docker sekä MySQL Workbench. Docker sisälsi itse projektin sekä myös tietokannan.

Saimme opinnäytetyön aiheen työharjoittelun kautta. Toimeksiantajalla oli useampi eri projektivaihtoehto, joita voisimme lähteä toteuttamaan. Valitsimme tämän projektin, koska se vaikutti mielenkiintoisimmalta ja käytännönläheiseltä työltä.

2 PROJEKTISSA KÄYTETYT TYÖKALUT

2.1 Docker

Docker on eräänlainen alusta, jonka sisällä voidaan käynnistää erilaisia ohjelmistoja. Docker sisältää kontteja, eivätkä ne vaadi omaa käyttöjärjestelmää, toisin kuin virtuaalipalvelimet. Kontit on mahdollista yhdistää toisiinsa Dockerin virtuaaliverkon avulla. Dockerin avulla voidaan pitää sovelluksen tarvitsema ohjelmistoympäristö yhdessä paketissa, joka on myös perinteistä käyttöjärjestelmää kevyempi. (Wallenius Consulting 2022.)

Dockerin suurin hyöty sovelluskehittäjälle tulee siinä, että sovelluksen siirto kehitysvaiheesta testiin sekä tuotantoon on suoraviivaisempaa. Sovellus standardisoi ajoympäristön, joten kehitysympäristö on saman näköinen kuin testi- tai tuotantoympäristö. Dockerin avulla päivitysten ajaminen sekä uusien ominaisuuksien luominen voidaan julkistaa nopeammin verrattuna perinteiseen palvelimiin perustuvassa kehitysrakenteessa. (Wallenius Consulting 2022.)

Docker sisältää eräänlaisia komponentteja, joiden avulla kontit rakennetaan. Komponentit ovat seuraavia: Docker image, Docker File, Engine, Container, Registry. (Wallenius Consulting 2022.)

Docker Image on yksittäinen tiedosto, joka sisältää sovelluksen, ohjelmistot ja käyttöjärjestelmän. Image on yksi iso kokonaisuus mitä voidaan siirtää paikasta toiseen. Tiedosto sisältää kaiken tarvittavan mitä projektin ajaminen vaatii. (Wallenius Consulting 2022.)

Docker File sisältää tiedon siitä, miten Docker Image rakennetaan. Tämä on siis konfiguraatiotiedosto, joka on oleellinen osa Docker-projektia käynnistäessä. (Wallenius Consulting 2022.)

Docker Engine pyörittää palvelimella suoritettavia kontteja. Enginen avulla voidaan pakata sovelluskokonaisuus yhdeksi imageksi, joka voidaan siirtää tarvittaessa, vaikka toiselle käyttäjälle. (Wallenius Consulting 2022.)

Docker Container eli kontti rakentuu imagen kautta. Käynnistetty kontti ei itsessään käytä täysiä resursseja vaan se pyytää käyttöjärjestelmältä resursseja tarvittavan määrän. (Wallenius Consulting 2022.)

Registry on Docker-tiedostojen säilytyspaikka. Näille on olemassa sekä julkisia että yksityisiä varastoja. Tunnetuin julkinen varasto on Docker Hub, jossa on valmiina tuhansia imageja. Yritykset voivat säilyttää myös imageja omassa varastossaan, jolloin yrityksellä on saatavilla juuri itselleen sopivat konfiguraatiot helposti. (Wallenius Consulting 2022.)

Projektissa käytetty tietokanta oli Laravel-ohjelmistokehyksen päälle tehdyn sovelluksen tietokanta, joten kannan käyttämiseksi tuli asentaa myös kyseinen sovellus Docker konttiin. Lokaali kehitysympäristö tuli Laravel Sail komentorivikäyttöliittymän tarjoamana. Laravel Sailin avulla pystytään käyttämään Laravelin oletus Docker-kehityskäyttöympäristöä. Sail tarjoaa hyvän pohjan Laravel-sovelluksien kehittämiseen etenkin, jos ei ole aiempaa kokemusta Dockerista. (Laravel 2022.)

Sisimmältään Sail on docker-compose.yml-tiedosto sekä sail-skripti, mikä tallentuu projektin juureen. Sail siis tarjoaa komentorivikäyttöliittymän, missä on erilaisia menetelmiä vuorovaikutukseen Docker konttien kanssa. Laravel Sailia pystyy käyttämään Windows, Linux sekä macOS -käyttöjärjestelmissä. Windowsilla käytäessä tarvitaan WSL2-yhteensopivuuskerrosta, koska komennot ovat Linux-pohjaisia (Laravel 2022.)

Ensimmäisenä asennuksen vaiheena on php-riippuvuuksien sekä Sailin asennus Dockerin avulla. Tämän jälkeen tulee suorittaa komento: ”cp .env.example .env”,

mikä luo example.env-tiedoston. Kyseisessä tiedostossa säilytetään tietoja prosessin toimintaympäristöstä.

Näiden jälkeen kontti voidaan käynnistää sille luodolla komennolla: ” vendor/bin/sail up -d”, mutta asennus ei ole valmis vaan seuraavaksi tulee siirtyä Sail Shelliin siihen tarkoitetulla komennolla: ” vendor/bin/sail shell”. Ensimmäisenä Sail Shellin sisällä suoritettavana komentona on: ” npm install && npm run prod”. Kyseisellä komennolla luodaan käyttöliittymän resurssit.

Sail Shellin sisällä voidaan suorittaa kaikkia muita komentoja paitsi Sail komentoja. Sail komennot tulee suorittaa Sail Shellin ulkopuolella. (Laravel 2022.)

Projektin tietokantaan on luotu esimerkkitietoja, mitkä saadaan luotua sille tehdyllä komennolla: ” php artisan migrate:fresh --seed”. Vaihtoehtoisesti voidaan myös luoda täysin tyhjä kanta omalla komennollaan: ” php artisan migrate:fresh”. Näitä kahta komentoa voidaan käyttää jatkossakin, jos kannan haluaa siistimmäksi. Projektin aikana käytimmekin näitä komentoja muutamia kertoja, koska testaamisen aikana tuli paljon turhaa sekä huonoa dataa, mitkä hankaloittivat testaamista.

Tämän jälkeen asennus on valmis ja jatkossa Docker-kontti käynnistetään käynnistyskomennolla: ” vendor/bin/sail up -d” ja suljetaan sulkemiskomennolla: ” vendor/bin/sail down”.

2.2 Windows Subsystem for Linux

Windows Subsystem for Linux on yhteensopivuuskerros, jonka avulla voidaan käyttää Linuxille suunnattuja sovelluksia Windows 10 käyttöliittymässä. Microsoft kehitti WSL:n helpottaakseen Linux sovelluksien käyttöä Windowsilla. WSL:n käyttämien vaatii 64-bittisen version Windows 10-käyttöjärjestelmästä, joka on julkaistu 2.8.2016 tai myöhemmin. (Computer Hope 2021.)

WSL versioita on kaksi: alkuperäinen WSL sekä paranneltu versio WSL2. Alkuperäinen versio julkaistiin vuonna 2019. WSL2 paransi tiedostojärjestelmän suorituskykyä sekä lisäsi täyden yhteensopivuuden järjestelmäkutsuille. WSL2 hyötyy aidon Linux-ytimen käytöstä, sillä se käyttää täysin uutta arkkitehtuuria. Uusi arkkitehtuuri muutti Linux-binaarien vuorovaikutusta Windowsin sekä tietokoneen komponenttien kanssa. Tästä huolimatta käyttäjäkokemus on sama kuin edeltäjässä. Kummallakin versioilla pystytään ajamaan yksittäisiä Linux-jakeluja sekä molempien versioiden jakeluja pystytään käyttämään rinnakkain. (Microsoft 2021.)

2.3 WordPress

Wordpress on avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä. WordPressiä käytetään esimerkiksi kotisivujen, blogien sekä verkkokauppojen luomiseen. WordPress on ilmainen alusta, jonka voi kuka tahansa ladata esimerkiksi WordPressin sivuilta. WordPress sisältää useita valmiita ulkoasuteemoja sekä lisäosia. Enemmän teemoja ja lisäosia on saatavilla maksua vastaan. (Zoner 2021.)

WordPressillä on laaja käyttäjäyhteisö, joten yhteisöltä saa apua erilaisiin ongelmatilanteisiin esimerkiksi foorumeilta. Jos WordPressin kanssa kohtaa ongelman, on luultavasti joku muukin kohdannut samat ongelman aikaisemmin, joten laajasta käyttäjäyhteisöstä on hyötyä. (Zoner 2021.)

WordPressillä on hyvin kattava ja monipuolinen ohjausnäkymä. Ohjausnäkymästä voi kontrolloida kaikkea aina sivuston luomisesta valikkojen luontiin. Ohjausnäkymästä voi luoda esimerkiksi valmiin yhteydenottolomakkeen, mutta tarvitset sitä varten jonkin lisäosan. Lisäosia on saatavilla WordPressin sivustolta, joista osa on ilmaisia. Lisäosia on myös mahdollista tehdä itse. (hostingpalvelu 2021.)

Wordpressin tietoturvasta puhutaan laajalti erilaisilla foorumeilla. Yleinen neuvo on, että tietoturva paranee tietoturvalisäosia lisäämällä. Itse Wordpressin ydintä varten ei välttämättä tarvitse asentaa tietoturvaan liittyviä lisäosia. Lisäämällä

lisäosia, lisääntyy tietoturvan tarve. Internetistä löytyy laajoja oppaita Wordpressin tietoturvan lisäämiseksi sekä ylläpitoa varten. (Seravo 2021.)

Wordpress voidaan asentaa omalle palvelimelle, tai voidaan käyttää WordPressin omaa palvelua. WordPressiä voi pitää käynnissä erilaisissa ympäristöissä, ja seuraavaksi käydäänkin läpi, millaisia vaihtoehtoja on olemassa.

Wordpressin palvelunympäristö vaatii PHP-ohjelmointikielestä vähintään version 7.4. Tietokannat vaativat myös tietyn version, riippuen siitä, mitä tietokantaa käyttää. Näiden lisäksi palvelimella tulee olla jokin HTTP-palvelinohjelma. Nykypäivänä on tarjolla useita hosting-yrityksiä, jotka tarjoavat vaatimuksen täyttävät ympäristöt. WordPress voidaan asentaa webhotelliin, WordPressin-premium hostingiin tai omaan virtuaalipalvelimeen. Oikean hosting-vaihtoehdon valitseminen on aina tapauskohtaista. Valinta riippuu siitä, millaista liikennettä sivustolla tapahtuu. Suurin osa sivustoista käyttää webhotellia, joissa liikehdintä on pientä. (WP-opas 2020.)

2.4 MySQL Workbench

MySQL Wordbench on visuaalinen työkalu tietokanta-arkkitehtien, -kehittäjien ja järjestelmänvalvojien käyttöön. Sen ominaisuuksiin kuuluu SQL kehitys, tietojen mallintaminen sekä erilaisia hallintatyökaluja palvelimien konfigurointiin, varmuuskopiointiin ja käyttäjienhallintaan. Siinä on myöskin helppokäyttöinen ratkaisu taulukoiden, objektien ja tietojen siirtämiseen muista tietokantaohjelmistoista. MySQL Workbench toimii Windows, Linux sekä macOS -käyttöjärjestelmissä. (MySQL 2022.)

MySQL Workbenchin avulla pystytään suunnitemaan, mallintamaan, luomaan sekä hallitsemaan tietokantoja visuaalisesti. Datamallintajat pystyvät hyödyntämään MySQL Workbenchin työkaluja monimutkaisten ER-mallien luomiseen, eteenpäin sekä käänteiseen suunnitteluun. Muita ominaisuuksia ovat ratkaisut

vaativaan muutosten hallitsemiseen sekä dokumentointitehtävien suorittamiseen. (MySQL 2022.)

MySQL Workbenchin laajaan tarjontaan kuuluu lukuisia visuaalisia työkaluja, jotka auttavat SQL-kyselyjen luomisessa, suorittamisessa sekä optimoimisessa. SQL-muokkaustyökalun ominaisuuksiin kuuluu automaattinen täydennys, värisyntaksin korostaminen, suoritushistoria sekä SQL-pätkien uudelleenkäyttö. Visuaalisen konsolin kautta pystytään hallitsemaan MySQL-ympäristöjä sekä parantaa tietokantojen näkyvyyttä. (MySQL 2022.)

3 PÄIVÄKIRJARAPORTOINTI

Projektin tavoitteena on luoda yrityksen verkkosivuille WordPress liitännäinen, jonka avulla sivustolla vierailevat asiakkaat saavat sovellusidealleen hinta-arvion. Sivustolle liitettävä liitännäinen sisältää monivaiheisin syöttölomakkeen, ja se sisältää olennaisia kysymyksiä sovelluksen kokonaisuudesta ja toiminnallisuudesta. Tavoitteena on pitää syöttölomake mahdollisimman yksinkertaisena käyttäjäkokemuksen vuoksi. Syöttölomakkeen lopussa tulisi olla mahdollista pyytää yritystä ottamaan yhteyttä antamalla sähköpostiosoite lomakkeen viimeisellä sivulla.

Hinta-arvio muodostuu asiakkaan syöttämien käyttäjien määrästä sekä tapahtumista, mitä sovelluksessa tapahtuu. Syöttölomakkeen avulla kartoitetaan karkeaa kokonaisuutta sovelluksesta, ja siitä syystä tarkkaa hintaa on vaikea arvioida.

Lomake sisältää erilaisia kysymyksiä ja syöttökenttiä. Syöttökenttien tiedot tulee siirtää lomakkeesta tietokantaan siinä vaiheessa, kun asiakas on täyttänyt lomakkeen ja vastannut jokaiseen kysymykseen. Käytämme toimeksiantajalta saatua tietokantaa, jossa kaikki taulut ovat valmiina. Jokainen sovellusidea siirtyy tietokantaan omana sovelluksenaan, joten tietokannasta ei tarvitse myöhemmässä vaiheessa etsiä tietoja useammasta paikasta.

Toimeksiantaja luovutti projektin alussa syöttölomakkeen mallikuvat. Tavoitteena on johdatella näitä mallikuvia, kuitenkin käyttäjäkokemus huomioiden. Syöttölomaketta suunnitellessa tulee huomioida käyttäjän liikkumista lomakkeen eri sivuilla. Esimerkiksi edellinen- ja seuraava -napit tulee olla samalla kohtaa jokaisella sivulla.

Päiväkirjan tavoitteena on kuvata viikoittain projektin etenemistä. Jokaiselta päivältä on kirjattu muistiinpanoja päivän tapahtumista. Viikot alkavat alkutekstillä, jossa käydään läpi mitä kyseisellä viikolla on tarkoitus tehdä. Jokainen viikko sisältää viikkoanalyysin, jossa pohditaan kulunutta viikkoa.

3.1 Viikko 1

Ensimmäisellä viikolla kävimme läpi projektin kokonaisuutta sekä niitä työkaluja, joita projektissa ensisijaisesti tarvittaisiin. Sovimme muistiinpanojen kirjoittamisesta, työtuntien kirjaamisesta sekä kommunikoinnista projektin aikana. Lopuksi tutustuimme uusien työkalujen dokumentaatioon tarkemmin.

Keskiviikko

Projektin ensimmäinen vaihe oli pitää kokous toimeksiantajan kanssa. Tässä kokouksessa kartoitimme projektin runkoa sekä sen toimintaperiaatetta. Kävimme läpi käytettäviä ohjelmointikieliä sekä työkaluja. Projekti kuulosti tässä vaiheessa osittain hyvinkin haastavalta, sillä toimeksiantajan yrityksessä työskentelevä työharjoittelija oli jo kehittänyt osuutta, joka myöhemmin liitettiin osaksi tätä projektia. Kyseinen osuus oli tehty Laravel-ohjelmistokehyksen päälle, josta meillä ei ollut ollenkaan kokemusta aikaisemmin. Työharjoittelijan osuus sisälsi testitietokannan, jota me tarvitsimme meidän osuudessamme.

Toimeksiantajan kokouksen jälkeen pidimme kahdestaan pienen tapaamisen, jossa kävimme läpi, millaista dokumentaatiota projektin aikana pidetään. Päätimme, että kirjaamme jokaisesta päivästä muistiinpanot helpottaakseen päiväkirjaraportointia. Tämän lisäksi halusimme seurata työtuntejamme, joten kirjasimme työtunteja Exceliin. Mietimme myös vaihtoehtoja, miten projekti vietäisiin läpi. Kyseessä on kuitenkin vain 2 henkilön työ, joten päätimme, että molemmat osallistuvat jokaisen projektin vaiheeseen. Tämä vaihtoehto tuntui parhaalta, sillä projektissa oli paljon uutta asiaa ja täten molemmat pääsivät oppimaan näitä asioita käytännössä.

Torstai

Kokouksen jälkeen aloitimme tutkimaan niitä työkaluja, joita tarvittiin kehitysympäristön pystyttämiseen. Erityisesti Docker nousi fokusoinnin pääaiheeksi, sillä se oli molemmille uutta asiaa. Yllätyimme miten monipuolinen Docker on. Vaikka

emme olleet aloittaneet vielä käytännön projektia, huomasimme heti asioita, mistä Dockerin käyttö tekee helppoa.

Päivän päätteeksi saimme kutsun toimeksiantajan versionhallintaan, mihin loimme omat käyttäjätunnukset. Päivän viimeisenä tehtävänä oli kloonata testitietokanta omille tietokoneillemme sekä tutustua tietokannan rakenteeseen.

Perjantai

Latasimme Docker-nimisen ohjelmiston, jonka avulla projektin eri osat saadaan toimimaan ja keskustelemaan toistensa kanssa. Tutustuimme Dockerin dokumentaatioon ja etsimme ohjeita, joiden avulla saisimme projektin käynnistettyä. Yrityksistä huolimatta emme saaneet projektia käynnistymään, joten päätimme kysyä ohjeita toimeksiantajalta.

Viikkoanalyysi

Ensimmäisen viikon aikana emme päässeet työskentelemään paljoakaan käytännön asioiden kanssa. Tutustuimme pääasiassa Dockeriin sekä itse projektiin. Viikon aikana tuli paljon uutta asiaa, joka toisaalta ehkä hidastikin itse projektin alkua. Viikko päättyi ongelmatilanteeseen, joten jäimme odottamaan ohjeita toimeksiantajalta.

3.2 Viikko 2

Tällä viikolla tehtävänä oli saada projekti toimimaan Dockerissa. Lisäksi selvitimme tietokantayhteyttä Wordpress-sivuilta projektin tietokantaan. Viikon aikana asensimme muun muassa WSL:n, WordPressin sekä MySQL Workbenchin. Viikko oli osittain hyvin teoreettinen ja saimmekin tehdä tutkimustyötä siitä, miten projektissa päästään eteenpäin.

Tiistai

Toimeksiantaja oli vastannut viestiimme ja lupasi laittaa meille ohjeet, joiden avulla pääsisimme projektissa eteenpäin. Latasimme versionhallinnasta projektin uuden version, joka sisälsi ohjeet projektin käynnistämiseen.

Aloitimme lataamalla WSL:n (Windows Subsystem for Linux) -käyttöliittymän Microsoftin sivuilta. Tämä vaadittiin, jotta pystyisimme ajamaan projektin käynnistyskomentoja Linuxin kautta. Seuraavaksi aloitimme syöttämään näitä komentoja Linuxiin käynnistääksemme projektin Dockerissa. Komentoja oli hyvin paljon, mutta ne olivat ohjeissa siinä järjestyksessä, missä ne piti suorittaa.

Torstai

Päivän ensimmäinen tavoite oli asentaa WordPress Dockerin konttiin. Tämän asentaminen ei tuottanut lainkaan ongelmia, sillä netistä oli saatavilla selkeät ohjeet WordPressin asentamiseen. Olimme myös aikaisemmin käyttäneet WordPressiä, mikä helpotti asennusprosessia ja sen käyttämistä.

Seuraavaksi aloimme tutkimaan sitä, miten kahden eri kontin välille saataisiin luotua yhteys. Tarkoitus oli saada omassa kontissa pyörivä WordPress-sivusto lähettämään dataa toisessa kontissa olevaan tietokantaan. Teimme yksinkertaisen testisivuston, jonka tarkoituksena oli auttaa yhteyden luomisessa (Kuva 1). Testisivu sisälsi kolme tekstikenttää ja lähetä-painikkeen. Pidimme testisivun mahdollisimman yksinkertaisena, jotta yhteyden testaaminen olisi selkeämpää. Yrittäessämme lisätä dataa tietokantaan huomasimme, että konttien välinen yhteys ei kuitenkaan toiminut.

```php
$mysqli = new mysqli("172.23.0.2:3306","root","secret","default");

// Check connection
if($mysqli === false){
        die("ERROR: Could not connect. " . $mysqli->connect_error());
}

// Taking all values from the form data(input)
    $customer_name = $_POST['customer_name'];
    $application_name = $_POST['application_name'];
    $description = $_POST['description'];

    if (isset($_POST['sbbutton'])) {
        // Performing insert query execution
        $sql = "INSERT INTO applications (user_id, customer_name, application_name, description)
        VALUES ('1','$customer_name','$application_name','$description')";

        if (($mysqli->query($sql) === true)) {
            echo "Record Added Sucessfully";
        } else {
            echo "Error " . $sql . "<br/>" . $mysqli->error;
        }

    }
    // Close connection
    $mysqli->close();
```

Kuva 1. Testisivun yhteystesti tietokantaan.

Perjantai

Aloitimme päivän tutkimalla tarkemmin tietokantaa ja tauluja, joihin yritimme tietoa siirtää. Kokeilimme myös erilaisia tapoja ja koodeja siirtää dataa tietokantaan. Tätä hetken testailtuamme tulimme kuitenkin siihen lopputulokseen, että testisivulla oleva koodi on oikein kirjoitettu. Päivän päätteeksi etsimme internetistä tarkemmin, miten kaksi Docker-konttia saisi keskustelemaan keskenään, sillä aloimme miettimään, että ongelma johtuisi Dockerista eikä testisivun koodista.

Viikkoanalyysi

Kuluneen viikon aikana vastaan tuli hyvin paljon erilaisia ratkaistavia ongelmia. Pääsimme tutustumaan uusiin työkaluihin, joita projektissa tarvittiin. Asensimme WSL2-käyttöliittymän sekä WordPressin Dockerin avulla liitännäisen tekoa varten.

Tällä viikolla WordPressiä käytettiin vasta testaamiseen eikä itse liitännäisen tekemisiseen. Testasimme tietokantayhteyttä monella eri PHP-koodi-variaatiolla, mutta emme saaneet yhteyttä toimimaan millään niistä.

Olemme kouluaikana tehneet lukuisia erilaisia harjoituksia PHP-kielellä. Kokeilimme myös käyttää muutamia koulussa tekemiämme harjoitustehtäviä apuna. Tiesimme, että ne toimivat ainakin silloin, kun käytimme niitä viimeksi, joten ajattelimme niiden toimivan tässäkin. Nekään eivät tuottaneet toivottua tulosta. Aloimme olemaan lähes varmoja siitä, että vika ei ollut meidän testisivumme koodissa.

Lähdimme tutkimaan internetistä, miten ylipäätään kaksi eri konttia voisivat keskustella keskenään. Etsimme tietoa Dockerin omasta dokumentaatiosta sekä muista lähteistä. Vastaan alkoi tulla kommentteja, joiden perusteella aloimme miettimään, että ongelma johtuisi Dockerista. Useissa eri lähteissä oli maininta siitä, että konttien välille on luotava yhteys. Päätimmekin, että lähdemme seuraavalla viikolla tutkimaan tätä yhteyden luomista konttien välille.

3.3 Viikko 3

Viikon aikana oli tarkoitus etsiä ratkaisua sille, miten kahden eri kontin välille voitaisiin luoda yhteys. Olimme tutustuneet aiheeseen jo aiemmin, joten tiesimme että tämä vaihe ei ollut helpoimmasta päästä.

Maanantai

Jatkoimme tiedonhakua siitä, miten kahden eri kontin välillä yhteys voitaisiin luoda. Ennen kuin lähdimme toteuttamaan yhteyttä konttien välille, kokeilimme asentaa WordPressin saman kontin sisään, josta myös aiemmin mainitsemamme tietokanta löytyy. Tutkimme asiaa vielä jonkin aikaa, mutta tajusimme kuitenkin, että tällä tavalla yhteyttä ei saada toimimaan.

Jatkoimme konttien välisen yhteyden muodostamisen tutkimista ja löysimme esimerkin mitä lähdimme kokeilemaan. Esimerkissä luotiin oma verkko Dockerissa ja molemmat kontit yhdistettiin tähän verkkoon. Emme saaneet yhteyttä vielä toimimaan, joten päätimme jatkaa sen tarkastelua ja testausta seuraavalla kerralla.

Tiistai

Jatkoimme aikaisemmin löytämämme esimerkin tutkimista ja soveltamista. Löysimme paremman esimerkin Docker-verkon luomisesta ja lähdimme toteuttamaan sitä. Tällä kertaa saimme verkon luotua sekä yhdistettyä molemmat kontit siihen. Verkon myötä jouduimme myös vaihtamaan PHP-koodin tietokantakyselyyn uuden osoitteen. Osoitteena oli aikaisemmin localhost, mutta Docker-verkon myötä siihen piti vaihtaa tietokannan IP-osoite, joka saatiin verkkoon yhdistämisen yhteydessä. Tässä vaiheessa avasimme Visual Studio Codella WordPressin projektikansion ja loimme Docker-verkon Visual Studio Coden terminaalissa seuraavasti:

Ensimmäiseksi luodaan Docker-verkko komennolla: docker network create – driver bridge my-network. Komennossa “my-network” on verkon nimi, jonka voi päättää itse. Jatkossa verkkoon viitataan sen nimellä.

Verkon luomisen jälkeen molemmat kontit yhdistettiin siihen komennolla: docker network connect my-network container-name (Kuva 2). Komennossa “container-name” on kontin nimi, jonka löytää Dockerista kun kontti on käynnissä. Kyseistä komentoa joudutaan käyttämään aina, kun kontit käynnistetään ja halutaan yhdistää verkkoon. Meidän tapauksessamme tietokanta tuli yhdistää ennen WordPress-sivua, sillä muuten yhteys ei toiminut ja data ei siirtynyt kantaan.

```
PS C:\Users\Nikke\Desktop\reccy_wp> docker network connect my-network reccy-mysql-1
PS C:\Users\Nikke\Desktop\reccy_wp> docker network connect my-network reccy_wp-wordpress-1
```

Kuva 2. Konttien yhdistäminen Docker-verkkoon.

Ongelman ratkettua aloimme testaamaan datan lisäämistä tarkemmin ja huomasimme, että sama data lähetetään tietokantaan joka kerta, kun sivusto

päivitetään. Ongelmaan löytyi nopeasti ratkaisu lyhyen JavaScript-koodin muodossa (Kuva 3).

```
// Stops adding data to database when refreshing the page
if (window.history.replaceState) {
  window.history.replaceState(null, null, window.location.href);
}
```

Kuva 3. JavaScript-koodi datan lisäämisen estämiseksi, kun sivusto päivitetään.

Toinen ongelma, jonka havaitsimme, oli että data lisääntyi tuplana tietokantaan. Tämä korjaantui poistamalla tarpeeton osa koodia, joka oli jäänyt aikaisemmista testailuista. Tarpeeton koodi teki saman asian kuin toimiva SQL INSERT-lause, joten se lisäsi kaiken datan tuplana.

Keskiviikko

Keskiviikkona emme edenneet paljoakaan projektissa, sillä teimme lyhyemmän päivän. Päivä koostui enimmäkseen edellisten päivien aikana tehtyjen asioiden tarkastamisella. Useamman ongelman ratkaisun jälkeen oli tärkeää tarkistaa koodi läpikotaisin ja varmistaa, että ongelmat ovat todellakin ratkaistu. Testasimme tekemäämme koodia ja totesimme, että kaikki toimi kuten pitikin. Päivän lopuksi laitoimme toimeksiantajalle viestin, missä kerroimme mitä olimme saaneet aikaan sekä kysyimme kuinka meidän tulisi jatkaa projektia, jotta olisimme samalla aaltopituudella sen suhteen.

Viikkoanalyysi

Viikon aikana tapahtui paljon edistystä edellisiin viikkoihin verrattuna. Tutkimme Docker-verkon luomista sekä projektien osien laittamista samoihin kontteihin. Docker-verkon luominen onnistui ja saimme kontit yhdistettyä kyseiseen verkkoon. Saimme luotua yhteyden tietokantaan ja pystyimme siirtämään sinne dataa.

Lopuksi oli vielä muutama pienempi ongelma, joihin löytyi kuitenkin ratkaisut nopeasti. Viikon päätteeksi laitoimme viestiä toimeksiantajalle kysyäksemme

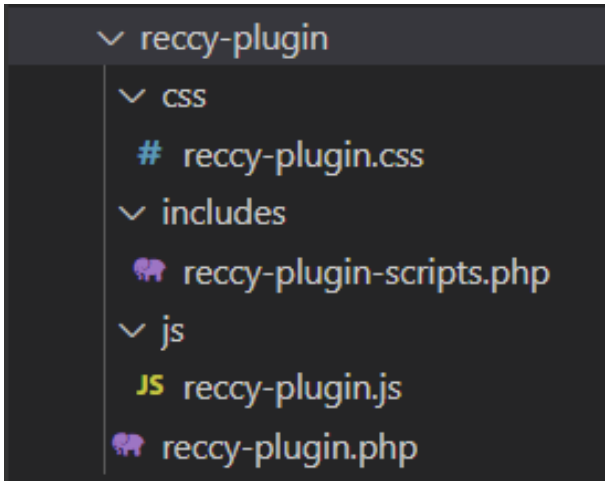
projektin jatkosta. Viikko oli tähän asti edistyksellisin ja saimme paljon aikaan. Opimme uusia asioita niin työkaluista kuin myös koodikielistä. Opimme etenkin Dockerin käyttämistä todella paljon.

3.4 Viikko 4

Tämän viikon tavoitteena oli tutustua lähemmin liitännäisen luomiseen. Loimme kansiorakenteen liitännälle valmiiksi sekä kokeilimme että kaikki toimii niin kuin pitää. Olimme saaneet mallikuvat jo aiemmin, joten lähdimme näiden pohjalta hahmottamaan liitännäistä visuaalisesta näkökulmasta.

Tiistai

Aloitimme viikon tutkimalla, miten WordPress-liitännäinen luodaan. Löysimme hyvän esimerkin, jota lähdimme toteuttamaan. WordPress-sivun plugins-kansioon luotiin uusi kansio liitännäistä varten (Kuva 4). CSS ja JavaScript -kielille luotiin omat tiedostot sekä kansiot, jotta koodi olisi selkeämpää luettavaa. Kansiorakenteen luotuamme lisäsimme ennestään tekemämme testikoodin sinne. Verkkosivulla liitännäistä kutsutaan lyhytkoodin avulla, joka määritettiin liitännäisen koodissa. Lopuksi liitännäinen otettiin käyttöön WordPress-sivun lisäosat -sivulta, jonka jälkeen testikoodimme näkyi verkkosivulla.



Kuva 4. Liitännäisen kansiorakenne.

Saimme projektin alkuvaiheessa liitännäisen mallikuvat, joiden pohjalta lähdimme toteuttamaan itse liitännäistä (Kuva 5). Mallikuvat havainnollistivat sitä, miltä liitännäisen tulisi visuaalisesti näyttää. Sovimme, että suunnittelemme ja toteutamme projektin vaihe vaiheelta, visuaalisuuden sekä toiminnallisuuden taustalla huomioiden.

Keskiviikko

Aluksi täytyi suunnitella, miten liitännäisessä pääsee etenemään sivulta toiselle sekä palaamaan takaisinpäin. Tutkittuamme erilaisia esimerkkejä internetissä ja löysimme hyvän monivaiheisen syöttölomakkeen esimerkin, jota päätimme lähteä soveltamaan omassa työssämme.

Pian huomasimme, että liitännäinen ei näyttänyt siltä miltä piti. Input-kentät sekä painikkeet eivät totelleet kirjoittamaamme CSS-koodia. Joissain tapauksissa myös koko liitännäinen muutti, jolloin sitä ei pystynyt käyttämään ollenkaan. Kokeilimme erilaisia ratkaisuja CSS koodiin, sillä epäilimme ongelman olevan siinä.

Tutkimme asiaa lisää ja tulimme siihen johtopäätökseen, että WordPress-sivun oletusteeman CSS-koodi sotki liitännäisen omaa CSS-koodia. Ongelma korjaantui

sillä, että oletusteeman otti pois käytöstä. Testasimme myös toista teemaa ja siinä liitännäinen näytti siltä miltä pitikin.

Kuva 5. Esimerkki mallikuvista.

Viikkoanalyysi

Tähän viikkoon mahtui paljon erilaista tekemistä. Pääsimme aloittamaan todellisen liitännäisen tekoa, kun olimme saaneet testiympäristöt toimimaan aikaisempien viikkojen aikana. Saimme luotua kansiorakenteen liitännäiselle sekä testattua sen toimivuutta aikaisemman testikoodin avulla. Teimme myös monivaiheisen syöttölomakkeen toiminnallisuutta, mikä oli molemmille täysin uutta asiaa.

Ongelmitta viikko ei kuitenkaan mennyt, sillä liitännäisen ulkoasun kanssa oli ongelmia. Onneksi ongelman aiheuttaja löytyi kohtuullisen nopeasti ja asia saatiin kuntoon. Mitään ratkaistavaa ei siis seuraavalle viikolle jäänyt.

3.5 Viikko 5

Edessä oli backend-ohjelmointiin keskittyvä viikko. Tarkoitus oli luoda backend-koodia liitännäistä varten. Tiesimme etukäteen, että viikko tulisi olemaan edellisiä viikkoja haastavampi. Emme halunneet keskittyä tällä viikolla liitännäisen käyttäjäkokemukseen tai ulkoasuun, vaan saada backend-koodi toimimaan.

Tiistai

Päivän tavoitteena oli saada actorit siirtymään liitännäisestä tietokantaan. Actorit oli saatava siirtymään ensimmäisenä tietokantaan, sillä niiden avulla muodostetaan myöhemmin liitännäisessä user-storyja. Actorit on siis sovelluksen käyttäjiä ja user-storyt ovat käyttäjien tapahtumia.

Aiempiin testeihin verrattuna actorien kantaan lisääminen erosi sillä tavalla, että samaan tauluun menee mahdollisesti useampi rivi tietoa, kun aikaisemmin meni vain yksi rivi. Lisäämisen sai toimimaan siten, että insert-lauseen arvo piti lisätä niin monta kohtaa kuin tekstikenttiä on (Kuva 6).

```
if (isset($_POST['sbbutton'])) {
    // Performing insert query execution
    $sql = "INSERT INTO actors (application_id, title_text) VALUES
    ('1','$actor1'), ('1','$actor2'), ('1','$actor3'), ('1','$actor4'), ('1','$actor5')";

    if (($mysqli->query($sql) === true)) {
        echo "Record Added Sucessfully";
    } else {
        echo "Error " . $sql . "<br/>" . $mysqli->error;
    }
}
```

Kuva 6. Actorien insert-lause.

Keskiviikko

Saatuamme actorien lisäämisen toimimaan edellisellä kerralla päätimme koittaa, miten saisimme useampaan tietokannan tauluun lisäämisen toimimaan. Kokeilimmme sitä testi insert-lauseen sekä actor insert-lauseen avulla. Lisäsimme alkuperäisen testi insert-lauseen actor insert-lauseen kokonaisuuteen ja saimme sen toimimaan (Kuva 7).

```php
if (isset($_POST['sbbutton'])) {
    // Performing insert query execution
    $sql = "INSERT INTO actors (application_id, title_text) VALUES
    ('1','$actor1'), ('1','$actor2'), ('1','$actor3'), ('1','$actor4'), ('1','$actor5')";

    $sql2 = "INSERT INTO applications (user_id, customer_name, application_name, description) VALUES
    ('1','$customer_name','$application_name','$description')";

    if (($mysqli->query($sql) === true) && ($mysqli->query($sql2) === true)) {
        echo "Record Added Sucessfully";
    } else {
        echo "Error " . $sql . "<br/>" . $mysqli->error;
        echo "Error " . $sql2 . "<br/>" . $mysqli->error;
    }
}
```

Kuva 7. Multi table insert.

Päivän päätteeksi päätimme vielä tutkia sitä, miten tekstikentästä pystyisi lisätä arvoja samanaikaisesti kahteen eri tekstikenttään sekä tietokantaan. Kyseinen osuus tulisi syöttölomakkeen kahteen seuraavaan osaan, sillä ne ovat melkein identtiset. Ainoa ero niissä oli se, että mihin tietokannan tauluun arvot menevät. Etsimme erilaisia esimerkkejä internetistä, mutta samankaltaisia esimerkkejä oli hyvin harvassa. Emme myöskään saaneet toimimaan liitännäistä niillä ohjeilla mitä löysimme.

Torstai

Jatkoimme eilisen ongelman tutkimista ja testasimme erilaisia tapoja tilanteen ratkaisemiseksi. Löysimme useita eri malleja, jotka auttoivat eteenpäin ongelman kanssa. Näiden mallien avulla pääsimme projektissa eteenpäin. Kävimme läpi esimerkkejä ja yhdistelimme koodeja, ja tällä tavoin saimme toimivan kokonaisuuden.

Teimme JavaScript-koodin, jolla arvot lisätään tekstikenttään sekä myöhemmän osan pudotusvalikkoon (Kuva 8). Ensimmäisillä riveillä määritetään nimet tekstikentälle sekä pudotusvalikolle. Seuraavassa vaiheessa otetaan käyttäjän syöttämä tekstikentän arvo ja määritellään sille nimi, jotta sitä voidaan helposti kutsua myöhemmin koodissa. Kolmannessa vaiheessa varataan muuttujalle paikka tekstikenttään sekä pudotusvalikkoon. Viimeisessä kohdassa muuttujan arvo määritellään tekstikenttään sekä pudotusvalikkoon käyttäjän syöttämän arvon mukaan.

```
//Insert new item to DomainEventsList listbox & dropdown
function insertItemIntoDomainEventsListBox() {
  var x = document.getElementById("domainEventsList1");
  var t = document.getElementById("domainEventsList2");

  var item = document.getElementById("domainItem2").value;

  var option1 = document.createElement("option");
  var option2 = document.createElement("option");

  option1.text = item;
  option2.text = item;
}
```

Kuva 8. Tekstikenttiin lisääminen.

Seuraavassa JavaScript-koodissa käydään tekstikenttä läpi ja valitaan sieltä kaikki arvot, jotta ne voidaan lisätä tietokantaan (Kuva 9). Tämä koodi tarvitaan, sillä insert-lause ei pysty lisäämään yksin kaikkia tekstikentän arvoja.

```
//Get all values from DomainList
function getDomainEventsList1(){
  var myList = document.getElementById("domainEventsList1");
  for(i=0;i<myList.length;i++){
    myList[i].selected = true;
  }
    return true;
}
```

Kuva 9. Arvojen valinta tekstikentästä.

Viimeiseksi arvot lisätään kantaan insert-lauseella (Kuva 10). Tässä kohtaa jouduimme käyttämään foreach-silmukkaa, koska arvoja oli useampi samasta tekstikentästä. Tämä erosi aikaisemmista insert-lauseista ja vaati vähän enemmän tutkimista ja testaamista, jotta saimme sen toimimaan haluamallamme tavalla.

```
// Inserting domain_events values to Database
if(isset($_POST['sbbutton'])){
    foreach ($_POST['domainEventsList1'] as &$domain) {
        $sql4 = "INSERT INTO domain_events (title_text, application_id) VALUES
        ('$domain','1')";

        mysqli_query($mysqli, $sql4);
    }
}
```

Kuva 10. Domain Events insert-lause.

Viikkoanalyysi

Tällä viikolla pääsimme pääasiassa tekemään backend-ohjelmointia. Useista löydetyistä esimerkeistä huolimatta viikko ei kuitenkaan ollut helppo. Jouduimme yhdistelemään uusia sekä vanhoja asioita, jotta saimme ratkaistua ongelmatilanteita. Missään vaiheessa ei ollut suoraa vastausta toteutuksesta vaan kaikki piti itse rakentaa pienistä osista liitännäisen tarpeen mukaiseksi.

Viikon aikana opimme paljon uusia asioita niin SQL kuin JavaScript -kielistä. Projekti eteni selkeästi eteenpäin, josta oli hyvä jatkaa seuraavalla viikolla.

3.6 Viikko 6

Tämän viikon työtehtäviin kuului toiminnallisuuksien viimeistelyä sekä ulkoasun työstämistä. Emme olleet aikaisemmin tehneet samankaltaista ulkoasua, joten viikko toi mukanaan paljon uutta asiaa. Ulkoasun tekemisen takia pääsimme käyttämään CSS-koodikieltä, joten pääsimme soveltamaan vanhoja oppeja sekä oppimaan uusia asioita.

Maanantai

Liitännäisen toiminnallisuudesta puuttui enää viimeinen osa, missä aikaisemmilla sivuilla syötetyt tiedot yhdistetään käyttäjätarinoiksi. Sivussa oleviin pudotusvalikkoihin oli tehty jo aikaisemmissa vaiheissa JavaScript-koodi, joten ainoa puuttuva osa oli niiden tietojen yhdistäminen ja lisääminen tekstikenttään.

Aluksi testasimme yhdistää tietoja samalla kertaa, kun ne lisätään tekstikenttään. Tämä ei kuitenkaan toiminut ja mietimme miten muuten asian pystyisi hoitamaan. Ratkaisu olikin helpompi kuin aluksi luulimme, sillä toimivassa ratkaisussa tiedot yhdistetään ennen tekstikenttään lisäämistä. Täten tiedot pystyttiin lisäämään yhtenä pakettina, kuten muissakin liitännäisen osissa.

Liitännäisen eri osien valmistumisen jälkeen aloimme tutkimaan, miten saisimme tehtyä multi-step formin, jotta liitännäisen eri osat saataisiin erillisille sivuilleen. Löysimme aika nopeasti hyvältä vaikuttavan esimerkin, jota lähdimme soveltamaan omaan työhömme. Multi-step formin toiminnallisuus tehtiin käyttäen JavaScriptiä. Saimme sovellettua sen ja kaikki eri osat olivat omilla sivuillaan, mutta huomasimme, että seuraava ja takaisin painikkeet eivät toimineet, jolloin emme saaneet mentyä liitännäistä läpi. Loppupäivän ajan tutkimme, mistä ongelma saattaisi johtua, mutta emme löytäneet ratkaisua ja päätimme jatkaa tutkimista seuraavalla kerralla.

Tiistai

Päivä alkoi multi-step formin ongelman selvittämisellä. Tutkimme asiaa jonkin aikaa sekä testasimme muokata koodia, mutta emme saaneet sitä toimimaan. Multi-step formin tekeminen oli meille täysin uusi asia, joten emme halunneet hukata liikaa aikaa tutkimalla asiaa turhan pitkään, joten päätimmekin, että etsimme uuden esimerkin, mitä voisimme yrittää soveltaa. Hiukan tutkittuamme asiaa, löysimme uuden tavan tehdä multi-step form, joka vaikutti lupaavalta projektimme kannalta. Sovelsimme sitä hieman niin, että se oli sopiva jo tekemäämme koodiin. Testasimme uutta koodia ja tällä kertaa se toimi ja pääsimme liikkumaan eteen sekä taaksepäin liitännäisen sisällä. Ensimmäisen version ongelman aiheuttaja ei selvinnyt meille ollenkaan, joka jäi harmittamaan, mutta tärkeämpää oli saada projektiimme toimiva ratkaisu ilman turhaa ajan hukkaamista.

Keskiviikko

Testatessamme multi-step formin toiminnallisuutta huomasimme, että kaikki tiedot eivät menneet kantaan. Se oli outoa, koska tietokantaan lisääminen oli toiminut täydellisesti ennen multi-step formin tekemistä. Päättelimmekin, että ongelma johtuisi siitä. Huomasimme, että jokaisella liitännäisen osalla oli omat form-tagit. Kantaan lisääminen toimi siten, että lisäyspainike havaitsi lisättävät tiedot form-tagien avulla ja se pystyi havaitsemaan tietoja vain yksien form-tagien sisältä. Otimme kaikista eri osista tagit pois ja pistimme liitännäisen ympärille yhdet omat, jonka jälkeen kantaan lisääminen onnistui jälleen. Olimme siis oikeassa, että ongelma johtui multi-step formin koodiin soveltamisesta.

Selvitettyämme ongelman huomasimme, että actorit eivät siltikään menneet kantaan. Aloimme tutkimaan asiaa, mutta emme löytäneet mitään normaalista poikkeavaa. Yritimme selvittää asiaa loppupäivän ajan siinä onnistumatta.

Torstai

Päivän aluksi jatkoimme edellisellä kerralla ilmenneen ongelman selvittämistä. Huomasimme, että actorien name-luokka oli vaihdettu id-luokkaan. Tämä teki sen, että actorit menivät normaalisti pudotusvalikkoon, mutta ei tietokantaan. Vaihdoimme id-luokan takaisin name-luokkaan, jonka jälkeen kantaan lisääminen onnistui jälleen. Huomasimme kuitenkin, että tarvitsemme myös id-luokkaa, jotta actorit lisääntyvät myös pudotusvalikkoon, joten lisäsimme senkin, jonka jälkeen molemmat lisäämiset toimivat normaalisti.

Ongelmien selvittämisten jälkeen pääsimme vihdoin tekemään ulkoasua alkuperäisten ohjeiden mukaisesti. Käytimme työssä CSS flexbox-verkkoasettelumallia, joka helpotti liitännäisen skaalattavuuden tekemistä. Tästä huolimatta ulkoasun tekemisessä oli hyvin paljon työtä, emmekä saaneet sitä päivän aikana kokonaan tehtyä.

Perjantai

Jatkoimme ulkoasun tekemistä. Teimme sitä osa kerrallaan ja kokeilimme päästä mahdollisimman lähelle annettuja ohjeita. Muutama osa tuotti hieman vaikeuksia, että kaiken sai juuri oikeille kohdille. Helpottaviakin tekijöitä oli, sillä ensimmäisellä sekä viimeisellä sivuilla oli enimmäkseen vain tekstiä sekä nappulat, joten sommittelu oli hyvin helppoa. Myös kaksi osaa olivat melkein identtiset. Ainoana erottavan tekijänä oli eri teksti ja syötettävät arvot. Päivän aikana saimme ulkoasun jo hyvin lähelle valmista, mutta vielä jäi muutama kohta, mitä pitäisi parantaa.

Viikkoanalyysi

Viikon aikana tapahtui paljon edistystä niin toiminnallisuuden kuin ulkoasunkin kannalta. Ongelmiakin viikkoon mahtui aika paljon, mutta ne saatiin kuitenkin selvitettyä. Työ alkoi jo näyttää melkein valmiilta muutamia ominaisuuksia sekä ulkoasuun tarvittavia muokkauksia lukuun ottaen.

Pääsimme käyttämään CSS-koodikieltä, jota emme olleet vielä projektissa juuri ollenkaan käyttäneet. Kokemusta kuitenkin oli jo koulussa tehdyistä projekteista, joten kaikkea ei tarvinnut oppia kokonaan alusta alkaen vaan enemminkin palautella vanhoja opittuja asioita takaisin mieleen. CSS-flexboxin käyttäminen oli kuitenkin molemmille melko uutta ja sen oppiminen vei jonkin verran aikaa, mutta se myös helpotti ulkoasun tekemistä.

3.7 Viikko 7

Viimeisen viikon tavoitteina oli lisätä viimeiset ominaisuudet sekä korjata virheitä ja siistiä koodia. Liitännäisestä puuttui muutamia ominaisuuksia, jotka paransivat käyttäjäkokemusta, joten ne oli hyvä lisätä. Tähänkin viikkoon mahtui vielä paljon ongelmanratkaisukykyä vaativia tilanteita.

Maanantai

Viikko alkoi ongelman selvittelyllä, jossa sovelluksen eri vaiheissa oli mahdollista lisätä tyhjiä rivejä erilaisiin tekstilaatikoihin sekä tietokantaan. Teimme jokaiselle tekstikentälle tarkistuksen, joka käy läpi jokaisen kohdan mihin käyttäjä pystyy syöttämään arvoja (Kuva 11). Tällä tavoin pystyimme välttämään tyhjien rivien ilmestymistä sovellukseen sekä itse tietokantaan.

```
//Check if user has written something to the input field
if (item == null || item == "") {
  alert("Lisää arvo kenttään.");
    return false;
}else{
  x.add(option1);
  t.add(option2);
}
```

Kuva 11. Tyhjien rivien tarkistus.

Sovelluksesta puuttui vielä hinnan laskeminen, joka saatiin myös tehtyä päivän aikana. Hinnan laskeminen toimi siten, että pudotusvalikkojen arvojen summat kerrottiin yksittäisen arvon hinnalla. Tämän jälkeen kaikkien pudotusvalikkojen summat yhdistettiin, josta muodostui lopullinen hinta.

Tiistai

Maanantaina tehdyistä tyhjien rivien tarkistuksista jäi vielä puuttumaan actor-kohdan tarkistus. Sen tekeminen on huomattavasti monimutkaisempaa, koska se sisältää useamman tekstikentän. Sovimme, että palaamme tähän ongelmaan vielä myöhemmin tällä viikolla.

Huomasimme, että seuraava ja takaisin -painikkeet olivat eri kohdissa jokaisella sivulla, joten päätimme korjata sen käyttökokemuksen mukavuuden kannalta. Painike laitettiin niille tehdyn flexbox-item -luokan sisään, jonka avulla ne saatiin aseteltua helposti samaan kohtaan jokaisella sivulla.

Torstai

Jatkoimme tiistaina kesken jäänyttä actor-ongelman tutkimista ja testaamista. Tutkittuamme asiaa pidemmän aikaa, löysimme ratkaisun, joka paransi insert-lausetta, mutta ei kuitenkaan vielä ratkaissut tyhjien rivien lisäämistä. Parannellussa insert-lauseessa käytettiin foreach-silmukkaa, joka mahdollisti sen, että jokaista actor-tekstikenttää ei tarvinnut lisätä erikseen. Tutkimme tyhjien rivien lisäämistä vielä lisää ja lopulta saimme keksittyä ratkaisun, millä kantaan ei enää mennyt tyhjiä rivejä (Kuva 12).

```php
// Actor insert
if (isset($_POST['sbbutton'])) {
    $actors=$_POST["actor"];
    foreach ($actors as $actor) {
      // Check if actor input fields are empty
      if (empty($actor)) {
          echo "Variable is empty.<br>";
          return false;
      }else{
        // Inserting actors values to Database
        $sql1 = "INSERT INTO actors (application_id, title_text) VALUES ('1','$actor')";
        $result=mysqli_query($mysqli, $sql1);
      }
    }
}
```

Kuva 12. Paranneltu insert-lause.

Liitännäisen puolella tyhjiä rivejä lisääntyi vielä actorin pudotusvalikkoon, joka piti ratkaista seuraavaksi. Sitä ei pystynyt ratkaisemaan samalla tavalla kuin muita kohtia, koska actoreille oli varattu useampia tekstikenttiä. Aloimme miettimään, jos lisäisimme arvot taulukkoon ja tarkastaisimme, onko siinä tyhjiä arvoja. Emme saaneet ratkaisua tehtyä pelkällä JavaScriptillä, joten jouduimme käyttämään myös jQueryä (Kuva 13). Tämä oli ainut kerta projektissa, kun käytimme sitä. Yhdessä JavaScriptin sekä jQueryn avulla saimme ratkaistua ongelman.

```
//Adds actor inputfield values to an array
var arr_act = $('input[name^=actor]').map(function(idx, elem) {
  return $(elem).val();
}).get();
```

Kuva 13. jQuery taulukko.

Tämänkin jälkeen oli vielä yksi ongelma selvitettävänä. Käyttäjän mennessä takaisin muokkaamaan actoreita, koodi lisäsi pudotusvalikon arvot uudelleen. Yritimme muokata aiemman ongelman koodia, mutta se ei auttanut.

Actor-sivussa ei ollut lisää -painiketta, joten jouduimme laittamaan lisäyksen sivun seuraava -painikkeeseen, josta ongelma johtui. Jokainen kerta, kun käyttäjä painaa painiketta, lähtevät tiedot uudelleen pudotusvalikkoon. Keksimme, että laitamme pienen funktion seuraavan sivun takaisin -painikkeeseen, joka tyhjentäisi pudotusvalikon arvot, minkä jälkeen ongelma ratkesi.

Perjantai

Tänään lähdimme tarkastelemaan tuottamaamme koodia sekä korjaamaan virheitä. Poistimme ylimääräisiä välejä koodista sekä etsimme mahdollisia bugeja. Kävimme koodin rivi riviltä läpi ja pyrimme kommentoimaan sitä mahdollisimman tarkasti kuvataksemme mitä missäkin vaiheessa tarkoitetaan. Kommentointi on hyvin merkittävä osuus ohjelmointia, sillä hyvin kommentoitu koodi auttaa myöhemmin, jos koodiin tulee tehdä muutoksia tai päivityksiä jostain syystä. Tämä oli hyvin mieluisaa puuhaa, sillä tiesimme että olemme loppusuoralla ja ylimääräinen kertaus koodin kulusta opetti meitä entistä enemmän.

Lähetimme koko projektin toimeksiantajallemme, jonka tehtävänä oli tarkistaa ja arvioida liitännäisen kokonaisuutta. Tämän jälkeen jäimme odottamaan mahdollisia korjausehdotuksia ja aloimme suunnittelemaan opinnäytetyön kirjallista osuutta.

Viikkoanalyysi

Viikon aikana ratkaisimme useamman erilaisen ongelman, jotka olivat konkreettisia virheitä liitännäisen toiminnan kannalta. Osa virheistä oli ratkaistavissa samalla tavalla, mutta esimerkiksi actorien lisäämisen kanssa tuli miettiä erilainen ratkaisu. Aikaisemmalla viikolla mietittyä insert-koodia muokkasimme myös erilaiseksi. Tämä oli hyvä muistutus siitä, että asiat voi tehdä monella eri tapaa ohjelmoinnin parissa.

Kuluneella viikolla pääsimme syventymään lähes kaikkiin projektissa käytettyihin ohjelmointikieliin. Vaikka meillä ei ollut aikaisemmin kokemusta jQuerysta, löysimme ratkaisun silti melko helposti.

Viimeisenä päivänä kävimme koodia läpi, joka oli hyvää kertausta projektista. Vaikka teimme kommentointia jo koodia kirjoittaessamme, löysimme silti osioita mitä pystyimme tarkentaa entistä paremmin kommentoinnilla.

4 YHTEENVETO

4.1 Toni Huhtala

Opinnäytetyön tavoitteena oli luoda toimiva sovellus verkkosivuille sovelluksien hintakartoitukseen. Sovellus oli tarkoitettu WordPress -sivustojen lisäliitännäiseksi. Sovelluksen tuli olla mahdollisimman yksinkertainen ja samalla myös monipuolinen, jotta sovellusidean vastaanottaja saisi mahdollisimman hyvän kuvan asiakkaan mahdollisesta sovellusideasta.

Opinnäytetyö osaltani onnistui erittäin hyvin. Saavutin kaikki itselle asettamani tavoitteet. Vaikka työ tehtiin parityönä, pääsin työskentelemään sovelluksen sekä teorian parissa. Tämä työ opetti minulle hyvin paljon uutta alasta. Pääsin tutustumaan esimerkiksi erilaisiin ohjelmiin, joita käytetään päivittäisessä työssä. Kirjallista raporttia tehdessäni opin yhdistelemään erilaisia lähteitä tekstiini sopivaksi.

Projektissa käytimme useampaa ohjelmointikieltä sekä erilaisia ohjelmia. Vaikka ohjelmointikielet olivat hyvin tuttuja, joutui niiden osaamista tässä projektissa syventämään. Pääsimme tekemään erilaisia tietokantalauseita PHP:llä, jotka lisäsivät useaan tauluun useasta syöttökentästä tietoa kerralla. Front-endin puolella pääsimme syventämään css-taitojamme visuaalisen suunnittelun merkeissä.

JavaScriptiä en ollut aiemmin paljoakaan käyttänyt. Tässä projektissa pääsimme käyttämään JavaScriptiä useammankin kerran, joka oli hyvää oppia tulevaa varten. Jouduimme etsimään netistä hieman useammin ratkaisuja, kun JavaScriptin parissa työskentelimme, mutta hakukoneen käytön optimointi vian etsintään harjaantui siinä samalla.

Ohjelmistoympäristön pystyttäminen ja Dockerin käyttö vaati eniten aikaa. Docker oli täysin uusi työkalu meille molemmille, joten jouduimme käyttämään sen dokumentaatioon huomattavasti tunteja. Tiesimme kuitenkin, että Dockeria käytetään myös työelämässä, joten tutustuimme mielellään sen käyttöön tarkemmin.

Projektin aikana pääsimme tutustumaan myös isomman valmiin tietokannan rakenteeseen. Olimme koulussa tietysti jo tehneet aiemmin tietokantaratkaisuja sekä tietokantoja, mutta tässä työssä tietokanta oli vielä hieman isompi. Oli mielenkiintoista nähdä, miten työelämässä luotu oikea tietokanta on rakennettu, ja tästä saikin hyviä vinkkejä tuleviin projekteihin.

Mielestäni onnistuimme kokonaisuudessaan hyvin ja olenkin tyytyväinen työn lopputulokseen. Erityisen hyvin onnistuimme visuaalisen puolen suunnittelussa. Olimme saaneet projektin aikana MVP-kuvat, jotka antoivat kokonaiskuvan siitä miltä liitännäisen tulisi näyttää. Saimme tehtyä lähes identtisen liitännäisen visuaalisesta näkökulmasta tarkasteltuna.

Kaiken kaikkiaan tämä projekti opetti minulle ohjelmointikieliä sekä myös ohjelmia, joita ohjelmoinnissa voidaan käyttää. Mielestäni olen oppinut hahmottamaan paremmin sitä, miten projekti tulisi suunnitella ennen sen aloitusta. Lisäksi opin miten eri ohjelmia voidaan hyödyntää erilaisissa projekteissa. Esimerkiksi tietokannan ja Dockerin yhdistäminen oli helppoa sen jälkeen, kun Dockerin sai toimimaan. Samaa tietokantaa voi helposti käyttää useammassa eri projektissa, jos se vain pyörii Dockerissa jo valmiiksi.

Parannettavaa olisi varmasti voinut löytyä versionhallinnan käytöstä. Olisimme voineet hyödyntää enemmän versionhallintaa, josta projekti olisi ollut aina helppo saada päivittäin käsille. Tällöin ei olisi tarvinnut huolehtia projektin eri versioista ja olisimme voineet palata tarpeen tullen aiempaan pisteeseen, jos sovellus ei toiminutkaan. Toisaalta työskentelytapamme oli sellainen, että emme jakaneet tiettyjä osia erikseen vaan kehitimme sovelluksen alusta loppuun yhdessä vaihe vaiheelta.

Vaikka projektimme laajuus keskittyi vain pieneen osuuteen, olisimme varmasti voineet dokumentoida testausvaiheen. Testasimme syöttölomaketta ja etsimme virheitä, joiden avulla saatiinkin poistettua useampi virhe. Vaikka testaus oli hyvin

suppeaa projektin koon takia, olisimme saaneet varmasti hyvää oppia testauksien dokumentoinnista.

4.2 Niklas Metsola

Työn alusta asti oli paljon uutta opittavaa sekä vanhojen oppien syventämistä. Aluksi työympäristön pystyttäminen toi heti uusia opeteltavia ohjelmistoja sekä teknologioita. Uusien ohjelmistojen käyttäminen toi myös mukanaan ongelmatilanteita, mitkä pitkittivät prosessia, mutta nekin saatiin selvitettyä tutkimalla asioita tarkemmin, sekä kysymällä apua toimeksiantajalta. Opinnäytetyön aikana käytettiin myös monia koodikieliä, joita olin jo aikaisemmin käyttänyt ja pääsinkin syventämään vanhoja oppeja.

Opinnäytetyön aikana tuli tutuksi monia sovelluksia, etenkin Docker. Docker oli minulle täysin tuntematon sovellus, joten projektin alku meni täysin sen opiskeluun sekä testaamiseen. Alkuun Dockeria oli vaikea sisäistää ja sen kanssa menikin pitkään ennen kuin saimme projektin ylös. Projektin edetessä sovellus kuitenkin tuli tutuksi ja sen hyödyllisyyttä oppi arvostamaan. Uskoisin, että tulenkin jatkossa käyttämään Dockeria erilaisiin projekteihin.

Opinnäytetyön aikana tehty liitännäinen tuli WordPressillä tehdyille verkkosivuille, joten WordPress tuli myös hyvin tutuksi projektin aikana. Vaikka WordPressin omia ominaisuuksia ei hirveästi käytettykään koen, että oli sen käyttämisestä hyötyä tulevaisuutta ajatellen. Ongelmiakin oli sen kanssa, esimerkiksi valmiiksi luodut teemat sekoittivat CSS-koodia ja sen tutkimiseen menikin aikaa mikä ei ollut kovin miellyttävää. Tästä syystä aion tulevaisuudessa tehdä WordPress-sivut alusta asti ilman valmiita teemoja, jolloin kyseisiä ongelmia ei pitäisi tulla.

Projektin tietokannan pystytyksessä tarvittu WSL2 oli myös täysin uusi sovellus minulle. Olin aikaisemmin käyttänyt Linuxia, mutta en ollut käyttänyt Linux peräisiä komentoja Windowsilla. Sovellusta oli mielestäni erittäin helppo käyttää, koska se on hyvin samanlainen kuin Windowsin oma komentorivi, mitä olin käyttänyt

aikaisemmin. Linux-pohjaisten komentojen käyttäminen palautti vanhoja opeteltua asioita mieleen, mikä oli mielestäni hyvä asia, koska ohjelmoinnissa tarvitsee aika ajoin Linuxia.

Liitännäisen ulkoasun pohja luotiin käyttämällä HTML-kieltä. Minulla oli jo aikaisemmin paljon kokemusta kyseisen kielen käyttämisestä, joten projektin aikana en oppinut siitä juurikaan mitään uutta, mutta mielestäni sitä oli mukava käyttää sekä palauttaa mieleen vanhoja opittuja asioita.

CSS-kieltä olin myös käyttänyt jo aikaisemmin, mutta opin siitä hyvin paljon projektin aikana. Liitännäisen skaalattavuuden luomiseen käytimme FlexBoxia, mistä olin kuullut aikaisemmin, mutta en ollut käyttänyt sitä. Projektin aikana se tuli hyvinkin tutuksi, helpottaen ulkoasun tekemistä, sekä vähensi koodin määrää verrattuna siihen, että kaiken olisi tehnyt ilman sitä. Sen avulla kaiken ulkoasun eri osat sai omiin selvästi pienempiin osiin, mikä oli koodin selkeyden kannalta tärkeää.

JavaScript oli itselleni mielenkiintoisin koodikieli projektin aikana. Olin aiemmin käyttänyt sitä vain vähän, joten opin projektin aika paljon uusia ja mielenkiintoisia asioita. Projektin ulkoasun toiminnallisuus tehtiin melkein kokonaan JavaScriptillä ja mielestäni tärkeimpänä oppina JavaScriptin kannalta oli monivaiheisen syöttölomakkeen luominen. Teimme projektin aikana kaksi hieman eri tavoilla tehtyä syöttölomaketta, koska ensimmäinen ei toiminut kunnolla. Emme saaneet selville syytä, joten teimme sen eri tavalla. Ongelman jääminen mysteeriksi jäi kuitenkin mietityttämään, mutta onneksi saimme sen kuitenkin tehtyä toisella tavalla. Projektin aikana JavaScript alkoi kiinnostaa minua paljon ja ajattelenkin tulevaisuudessa syventää oppimiani asioita sekä opiskella JavaScript-kirjastoja, kuten esimerkiksi ReactJS.

Ainoana minulle täysin uutena koodikieliin liittyvistä asioista oli JavaScript-kirjasto jQuery. Olin kyllä kuullut siitä aikaisemmin, mutta en ollut opiskellut tai käyttänyt sitä yhtään. Projektissa käytimme sitä vain yhteen pieneen kohtaan, mitä emme saaneet pelkällä JavaScriptillä ratkaistua. Huomasin, että hyvin vähällä määrällä

koodia sai todella enemmän aikaan kuin normaalilla JavaScriptillä. Vaikka käytimme sitä hyvin vähän, sain kuitenkin hyvän vaikutelman siitä ja aionkin jatkossa opiskella myös sitä.

PHP oli HTML:n ohella minulle kaikista tutuin kieli mitä käytimme projektissa. Suurin osa sillä tehdyistä koodinosista olikin vanhojen asioiden mieleen palauttamista, mutta opin myös muutamia uusia asioita, sillä osa tietokantakyselyistä oli hieman monimutkaisempia, mitä olin aikaisemmin tehnyt. Niistäkin onneksi selvittiin pienellä opiskelulla sekä loogisella päättelykyvyllä. Uskon, että nämä uudet opit tulevat tarpeeseen, koska aion jatkossakin käyttää PHP-kieltä omissa projekteissa sekä mahdollisesti työelämässäkin.

Parannettavaa työssä olisi mielestäni ollut suunnittelussa, sillä välillä oli hetkiä, jolloin jouduimme tekemään asioita uusiksi tai teimme niitä huonossa järjestyksessä. Esimerkiksi teimme yhdessä kohtaa syöttölomaketta, kun olisi kannattanut tehdä tietokantakyselyt ensin valmiiksi. Ajoittain myös asioita ajateltiin liian monimutkaisesti, etenkin ongelmatilanteissa, jolloin ratkaisu olisi ollut paljon yksinkertaisempi, mitä aluksi ajattelimme. Ongelmatilanteet eivät kuitenkaan olleet täysin huono asia, sillä niistä pystyi ottamaan opikseen ja jo työn aikana pystyi huomaamaan, että opit menivät perille ja työn tekeminen helpottui.

Opinnäytetyön tekeminen parityönä oli minusta hyvä asia, koska se helpotti työskentelyä vaativissa tilanteissa, kun oli kaksi ihmistä etsimässä ratkaisuja sekä jakamassa vastuuta. Koin, että työnjakomme oli hyvä tämänkaltaiseen projektiin.

Henkilökohtaisina tavoitteinani minulla oli saada tehtyä toimiva liitännäinen, joka oli mahdollisimman paljon ohjeiden mukaan tehty sekä oppia projektiin kuuluvia ohjelmia ja koodikieliä. Mielestäni saimme tehtyä liitännäisen mikä vastasi annettuja ohjeita sekä opin paljon uuta kaikista käyttämistämme teknologioista, joten pääsin asettamiini tavoitteisiin.

Opinnäytetyö oli pitkä prosessi, johon kuului hyvin paljon työskentelyä sekä todella paljon uusien asioiden tutkimista ja opettelemista. Työn alussa kaikki tuntui kohtalaisen selvältä, mutta mitä pitemmälle prosessi eteni, sitä monimutkaisemmalta se tuntui. Monien vaiheiden jälkeen saimme kuitenkin työn valmiiksi, ja siitä sai paljon tärkeitä oppeja tulevaisuuden työelämää varten.

LÄHTEET

Computer Hope. WSL. Viitattu 16.6.2022. https://www.computerhope.com/jargon/w/wsl.htm

Docker. What is a Container? Viitattu 25.07.2022. https://www.docker.com/resources/what-container/

Freecodecamp. What is HTML. Viitattu 25.07.2022. https://www.freecodecamp.org/news/what-is-html-definition-and-meaning/

Hostingpalvelu. Wordpressin käyttöönotto. Viitattu 23.09.2022

https://www.hostingpalvelu.fi/ohjeet/wordpress/wordpressin-kayttoonotto/

ite wiki. Backend-kehittäjä. Viitattu 25.07.2022. https://www.itewiki.fi/opas/backend-kehittaja/

ite wiki. Frontend-kehittäjä. Viitattu 25.07.2022. https://www.itewiki.fi/opas/frontend-kehittaja/

Jquery. What is jQuery? Viitattu 25.07.2022. https://jquery.com/

Laravel. Laravel Sail. Viitattu 22.09.2022. https://laravel.com/docs/9.x/sail

MDN Web Docs. CSS: Cascading Style Sheets. Viitattu 25.07.2022. https://developer.mozilla.org/en-US/docs/Web/CSS

MDN Web Docs. What is JavaScript? Viitattu 25.07.2022. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

Microsoft. What is the Windows Subsystem for Linux? Viitattu 16.6.2022. https://docs.microsoft.com/en-us/windows/wsl/abo

MySQL. MySQL Workbench. Viitattu 25.07.2022. https://www.mysql.com/products/workbench/

Php. What is PHP? Viitattu 25.07.2022. https://www.php.net/manual/en/intro-whatis.php

Seravo. Wordpress-tietoturvan perusteet. Viitattu 04.10.2022. https://seravo.com/fi/wordpress-tietoturvan-perusteet/?gclid=Cj0KCQjwkOqZBhDNARIsAACsbfKQoJD-Ubk3sUSqKkl3QuTuDkVnEVirNB89QWG04mCcHlekKwTDa80aAlNvEALw_wcB

Visual Paradigm. Types of Actor in a Use Case Model. Viitattu 25.07.2022. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/types-of-actor-in-use-case-model/

Wallenius Consulting. Mikä on Docker ja mitä hyötyä siitä on? Viitattu 16.6.2022. https://niklaswallenius.fi/mika-on-docker/

WP-opas. Mitä eri WordPress-hosting vaihtoehtoja on? Viitattu 04.10.2022 https://wpopas.fi/mita-eri-wordpress-hosting-vaihtoehtoja-on/https://wpopas.fi/mita-eri-wordpress-hosting-vaihtoehtoja-on/

Zoner. Mikä on Wordpress? Viitattu 25.07.2022. https://www.zoner.fi/wordpress/mika-on-wordpress/