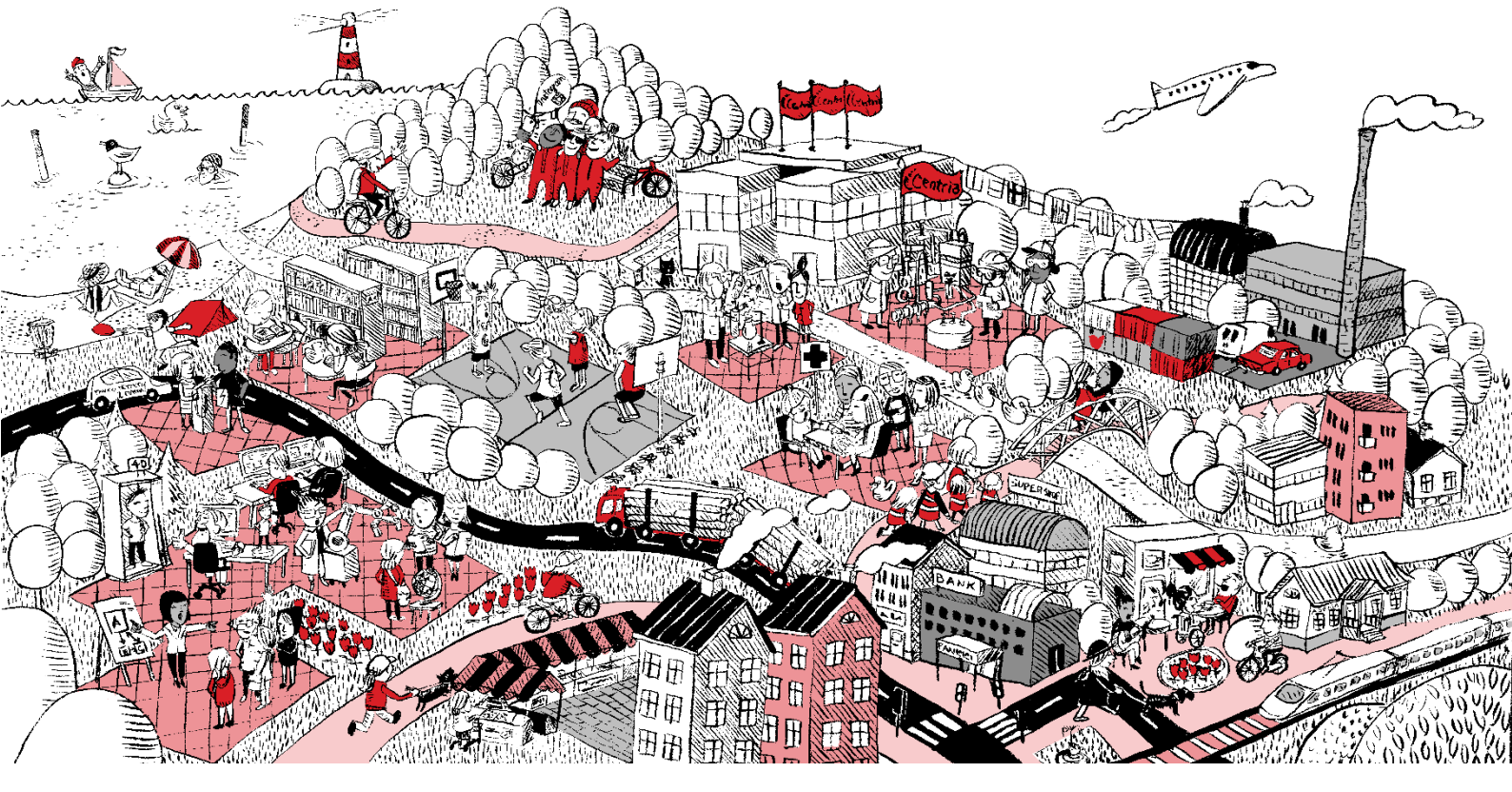
Samu Tihinen

PALVELINSOVELLUKSEN KEHITTÄMINEN MYYNTIPROSESSIN AUTOMATISOINTIIN

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Sähkö- ja automaatiotekniikan koulutus
Toukokuu 2022



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Toukokuu 2022	**Tekijä/tekijät** Samu Tihinen
Koulutus Sähkö- ja automaatiotekniikka		☒ AMK ☐ YAMK
Työn nimi PALVELINSOVELLUKSEN KEHITTÄMINEN MYYNTIPROSESSIN AUTOMATISOINTIIN		
Työn ohjaaja Hannu Leppälä		**Sivumäärä** 19+2
Työelämäohjaaja Tero Kaarlela		

Tämän opinnäytetyön aiheena on palvelinsovelluksen luominen valokuvaamon myynnin automatisointiin. Ohjelmointikielenä käytettiin JavaScriptiä ja koodin muokkaamiseen Visual Studio Code -sovellusta. JavaScriptiä ajetaan paikallisesti työpöydällä NodeJS-ajoympäristön avulla. Sovelluksen tietokantana toimi MySQL. Rajapintojen toimivuuden sekä ominaisuuksien testaamiseen käytettiin sovellusta nimeltä Postman.

Luotu sovellus voidaan asentaa palvelimelle, jolloin sitä voidaan käyttää selaimen kautta. Sovelluksen ominaisuuksiin kuuluu tietokannan hallinta eli tiedon lisääminen, poistaminen, siirtäminen ja muokkaaminen.

Asiasanat
Back-end, Front-end, JavaScript, MySQL, Ohjelmointi, Palvelinsovellus, Tietokanta, Visual Studio Code

ABSTRACT

Centria University of Applied Sciences	**Date** May 2022	**Author** Samu Tihinen
Degree programme Electrical and Automation Engineering		
Name of thesis DEVELOPMENT OF SERVER APPLICATION FOR AUTOMATION OF SALES PROCESS		
Instructor Hannu Leppälä		**Pages** 19+2
Supervisor Tero Kaarlela		

The topic of this thesis is developing a server application for automation of sales process. The code for the application was written on Visual Studio Code using JavaScript as the language. JavaScript code was executed locally using application called NodeJS. The Database and database server were developed using MySQL. All the testing was executed using an application called Postman.

The application can be installed on a server, and it can be used remotely via internet browser. All the information in the database can be read and fully modified with the application.

Key words
Back-end, Database, Front-end, JavaScript, MySQL, Software development, Visual Studio Code, Web application

KÄSITTEIDEN MÄÄRITTELY

Ajoympäristö

Sovelluksen suorittamiseen tarkoitettu alusta.

API

Application Programming Interface. Ohjelmien tai komponenttien välisestä keskustelusta vastaava rajapinta.

Back-end

Sovelluksen tiedonkäsittelypuoli

COCOMO

Constructive Cost Model, ohjelmistotuotannon kustannusmalli.

Front-end

Sovelluksen käyttöliittymä.

Palvelin

Internetin välityksellä palveluja tarjoava tietokone.

JSON

Tiedostomuoto

Tietokanta

Kooste tallennettua tietoa.

Yacc

Koodia kääntävä ohjelma

TIIVISTELMÄ
ABSTRACT
KÄSITTEIDEN MÄÄRITTELY
SISÄLLYS

KUVAT

1 JOHDANTO

Yrityksen kotisivuilla tarkoitetaan yleensä joko yrityksen verkkosivuja tai web-sovellusta. Verkkosivulla ja web-sovelluksella on teknisessä mielessä eroja, mutta yleisesti kummastakin käytetään termejä ”verkkosivu” tai ”kotisivu”. Hyvien kotisivujen voidaan ajatella toimivan yhdenlaisena alustana verkossa. Kotisivujen kautta yritys voi tuoda sekä itsensä että tarjoamansa tuotteet ja palvelut esille. Hyvät kotisivut toimivatkin tehokkaana kanavana yrityksen markkinoinnissa. Kun puhutaan yrityksen imagosta, tarkoitetaan mielikuvaa, jonka yritys antaa itsestään. Kotisivujen vaikutus imagoon voi olla todella merkittävä. Yritys voi kotisivujensa kautta kertoa esimerkiksi ympäristöystävällisistä toimintatavoistaan. Vaikutus imagoon voi olla myös negatiivinen. Heikosti toimivat ja epäsiistin näköiset kotisivut antavat huonon kuvan koko yrityksestä.

Yrityksen kotisivu voi myös toimia sen verkkokauppana. Ennen internetin yleistymistä kuluttajien käytössä mainostaminen tapahtui pääasiassa median ja mainostaulujen kautta. Internetin käytön yleistyttyä myös mainostaminen verkossa alkoi yleistyä. Aluksi verkkomainokset eivät eronneet lehtien mainoksista, mutta nykyään tilanne on toinen. Mainos verkossa voi esimerkiksi olla suora linkki yrityksen verkkokauppaan. Ennen verkkokauppoja asiakkaan piti ostaa tuote kaupasta tai tilata postitse. Tilaaminen tapahtui puhelimitse tai tilauslomakkeella. Kun tuote on mahdollista ostaa verkkokaupasta suoraan kotiin, nopeasti ja vaivattomasti, on kynnys ostamiselle pienempi. Verkkomainonta ja verkkokaupat eivät välttämättä eroa perinteisistä tavoista huomattavasti, mutta kuluttajalle pienetkin erot voivat olla ratkaisevassa asemassa. Verkkokauppojen ansiosta myös pienten yrittäjien on mahdollista saada paljon myyntiä. Verkkokaupassa myytävän tuotteen ei tarvitse kilpailla muiden tuotteiden kanssa näkyvyydestä tai hyllypaikoista marketeissa, sillä tuotteen myynti voidaan tehdä esimerkiksi omasta autotallista. Verkkokauppojen suuri etu perinteisiin kauppoihin verrattuna on saatavuus. Verkkokaupat ovat auki ympäri vuorokauden, vuoden jokaisena päivänä. Verkkokaupassa asiointi on mahdollista hoitaa melkein missä vain, kunhan asiakkaalla on älylaite ja yhteys internettiin.

Kotisivut voivat olla yritykselle oikea valtti kilpailun, asiakaskokemuksen ja myynnin kannalta. Kotisivujen kasvaneen tarpeen takia markkinoilla on paljon yrityksiä, jotka tarjoavat kotisivupalveluita. Olisi helppoa väittää, että kaikkien yritysten tulisi hankkia hyvät kotisivut. Kotisivujen hankkiminen on kuitenkin helpommin sanottu kuin tehty. Kotisivupalveluita käytettäessä ensimmäisenä täytyy ottaa huomioon hinta. Yksinkertaistenkin kotisivujen hankkiminen maksaa. Jos yritys tarvitsee suuren ja

monipuolisen verkkokaupan, voi hinta nousta todella korkeaksi. Kotisivujen hankkimisen jälkeen kuluja tulee kotisivujen vaatimasta ylläpidosta ja päivittämisestä. Teknologian ja tietoturvan kehittyessä kotisivujen tulee pysyä kehityksen mukana. Heikko tai vanhentunut tietoturva yrityksen kotisivuilla on haavoittuvaisuus, joka voi johtaa vakavaan tietovuotoon. Tietovuodon seuraus voi pahimmillaan olla asiakkaiden menetys ja yrityksen ajautuminen konkurssiin. Yrittäjän on mahdollista säästää taloudellisissa kuluissa tekemällä osa kotisivujen työstä itse. Kotisivujen tekemiseen ja ylläpitämiseen vaaditaan kuitenkin sekä työaikaa että ammattitaitoa.

Opinnäytetyön alussa tavoitteena oli luoda toiminnallisuutta sisältävät kotisivut valokuvaamolle. Kotisivujen kautta asiakas voisi tilata omilla kuvillaan varustettuja tuotteita, kuten juomakuppeja tai albumeita. Kuvat voivat olla valokuvaamon ottamia tai asiakkaan kuvaamia. Valokuvaamon työntekijälle voisi olla saatavilla työjono, josta näkyy töiden aikataulut ja käytettävissä oleva kalusto. Kävi kuitenkin nopeasti selväksi, että kokonaisen sivuston tekeminen olisi ollut liian iso työ opinnäytetyöksi. Työtä rajattiin niin, että työhön kuului vain palvelinpuolen toiminnallisuus eli back-end ja tietokanta. Näiden avulla kotisivuille valmistui hyvä pohja, jonka päälle on tulevaisuudessa mahdollista rakentaa myös käyttöliittymä.

2 TYÖN LÄHTÖKOHDAT

Yksi helppo tapa työn tehostamiseen on työvaiheiden automatisointi. Automatisoinnilla työtä saadaan nopeammaksi ja tarkemmaksi. Automaation avulla on myös mahdollista poistaa puuduttavat työvaiheet ja vaikuttaa positiivisesti työntekijöiden motivaatioon. Yrityksen kannattavuuden kannalta pienetkin säästöt voivat olla ratkaisevassa asemassa.

Valokuvaamon nykyisiltä kotisivuilta löytyy yleisiä tietoja yrityksestä sekä tarvittavat yhteystiedot. Yrityksellä on muutama työntekijä, toimitilat ja kotisivut. Valokuvaamon nykyisiltä kotisivuilta löytyy yleisiä tietoja yrityksestä sekä tarvittavat yhteystiedot. Suuri osa valokuvien ja asiakastietojen käsittelystä tapahtui kuitenkin manuaalisesti, joten kotisivujen toiminnallisuuden lisäämiselle on tarvetta. Ensimmäisenä haluttu toiminnallisuus oli myyntiprosessin automatisointi ja myyntiprosessiin liittyvien tietojen käsittely. Toinen haluttu toiminnallisuus oli aikaresurssien työjono ja ajanhallinta. Hyvän työjonon avulla on mahdollista nähdä yhdellä silmäyksellä töiden aikataulut ja käytössä oleva kalusto. Työjono helpottaa työtehtävien aikatauluttamista.

Ennen opinnäytetyön aloittamista pidettiin kokous Tero Kaarlelan kanssa. Kokouksessa käytiin läpi valokuvaamon sivuston kehittämiseen liittyviä tarpeita ja mietittiin, kuinka ne saataisiin toteutettua. Kokouksessa luotiin seuraavanlainen vaatimuslista kotisivujen toiminnallisuuden osalta:

- Tilaustietojen ja tiedostojen tallentaminen järjestelmällisesti eri lähteistä
- Omille tilauksille “type of photoshoot”-valinta
- Automaattinen sähköpostiviestin lähetys ja vastaanotto
- Työjono
- Käyttöliittymä em. ominaisuuksille

Kokouksessa käytiin myös läpi mahdollisia toteutustapoja, kuten erilaisia ohjelmointikieliä, palvelinalustoja ja muuta käytännön toteutukseen liittyvää. Vaatimuslista ja käytetyt työkalut kuitenkin muuttuivat paljon työn aikana.

3 TYÖKALUT

Sovelluskehityksessä työkaluilla tarkoitetaan työssä käytettyjä ohjelmointikieliä ja muita ohjelmia. Ohjelmointikielen valintaan vaikuttaa sovelluksen käyttötarkoitus, sovellusta käyttävä alusta sekä ohjelmoijan ammattitaito. Koodin luomiseen ja muokkaamiseen on markkinoilla monia eri sovelluksia, joista ohjelmoija voi valita mieluisensa. Muita hyödyllisiä ohjelmia sovelluskehityksessä ovat esimerkiksi koodin ajoympäristöt ja rajapintojen testaamiseen tarkoitetut ohjelmat. Palvelinsovellusta luotaessa on myös kannattavaa käyttää ulkoista tai paikallista palvelinta, jotta sovelluksen eri osien toimintaa voidaan testata oikeassa käyttöympäristössä.

3.1 JavaScript

JavaScript on monipuolinen ja laajasti käytetty ohjelmointikieli. Se on HTML-merkintäkielen ja CSS-tyylisivujen kanssa yksi tärkeimmistä nykypäivän työkaluista verkkosivujen tekemiseen. JavaScriptin avulla on mahdollista lisätä verkkosivulle toiminnallisuutta ja sitä käyttää jopa 95 % verkkosivuista. (W3Techs 2021.) JavaScriptiä käytetään pääasiallisesti verkkoympäristöissä, mutta sen käyttö peleissä, verkko-ohjelmoinnissa ja sovellusten luomisessa on yleistynyt. (JavaScript 2021.)

JavaScriptin kehitys käynnistyi Netscapen halusta lisätä komentokieli Navigator-selaimeen. Aluksi tämä yritettiin toteuttaa sulauttamalla joko Java- tai Scheme-ohjelmointikieli selaimeen. Kehitystyön alkuvaiheessa kuitenkin havaittiin, että kokonaan uusi kieli on paras vaihtoehto. JavaScriptin kehitykseen ovat suuresti vaikuttaneet AJAX-sovelluskehitystekniikka, Googlen V8 JavaScript -moottori sekä Node.js-ajoympäristö. (Springborad 2022.)

Tässä työssä JavaScriptiä päätettiin käyttää monestakin syystä. JavaScript on erittäin monipuolinen ja nykyaikainen kieli, ilman suurempia yhteensopivuusongelmia. JavaScriptillä on myös mahdollista luoda tarvittava käyttöliittymä tulevaisuudessa.

3.2 Node.js

Node.js on vuonna 2009 julkaistu avoimen lähdekoodin JavaScript-ajoympäristö. Node.js mahdollistaa JavaScript-koodin ajamisen muuallakin kuin verkkosivulla. Node.js:än ansiosta nykyään ei tarvita

omia ohjelmia verkkosivuille ja palvelimille, vaan kaikki ohjelmointi voidaan tehdä yhdellä ohjelmointikielellä ja ajaa palvelimella. Node.js yksinkertaistaa ja nopeuttaa verkkosivujen toimintaa ja helpottaa ohjelmoijan työtä. (Node.js 2021.)

3.3 Express.js

Express.js, tai yksinkertaisemmin Express, on Node.js:lle kehitetty avoimen lähdekoodin verkkosovellus runko, joka toimittaa yhdessä paketissa kevyen ja joustavan pohjan verkkosovelluksien kehittämiselle. Expressin ensimmäisen version julkaisi TJ Holowaychuk vuonna 2010. (Github 2021.) Express on laajalti käytetty (Expressjs 2021.) ja hyvin tunnettu verkkosovelluksien runkona. Tämän ansiosta Expressin käyttäminen on helppoa ja yleisimpien ongelmien ratkaiseminen helppoa.

3.4 Visual Studio Code

Lähdekoodin muokkaustyökaluna käytettiin ohjelmaa Visual Studio Code. Se on Microsoftin luoma avoimen lähdekoodin tekstieditori, joka tukee useita alustoja ja ohjelmointikieliä. Visual Studio Code sisältää monia koodin luomista ja muokkausta helpottavia työkaluja kuten koodin täydennyksen, automaattisen virheiden korjauksen ja syntaksin korostuksen. Nämä valmiit ominaisuudet yhdistettynä ohjelman vapaaseen muokattavuuteen ja laajaan valikoimaan lisäosia tekevät Visual Studio Codesta erittäin käyttäjäystävällisen. (Visual Studio Code 2021.)

3.5 Postman

Postman on ohjelmistorajapintojen testaamiseen käytetty ohjelma graafisella käyttöliittymällä. Sen avulla oli helppo testata opinnäytetyön sovelluksen rajapinnat tietokannan hallintaa varten. Postman on ilmainen, helppokäyttöinen ja monipuolinen työkalu rajapintojen toiminnallisuuden testaamiseen. Monipuolisten ja kattavien testien ansiosta mahdolliset virheet koodin tai tietokannan toiminnassa löydetään helposti. (Postman 2021.)

4 TIETOKANTA

Opinnäytetyössä luotiin sovelluksen rinnalle tietokanta, joka vastaa tallennettujen tietojen säilyttämisestä ja saatavuudesta. Yksinkertaisimmillaan tietokanta voi olla tekstitiedosto, johon kirjataan esimerkiksi asiakkaiden yhteystiedot. Kun tallennettavan tiedon määrä kasvaa, muuttuu manuaalisesti päivitettävän tietokannan ylläpitäminen työlääksi. Joustavan tietokannan luomiseen on moderneja työkaluja, joilla tietokantaan saadaan automatiikkaa ja monia eri toimintoja tulevaisuuden varalle.

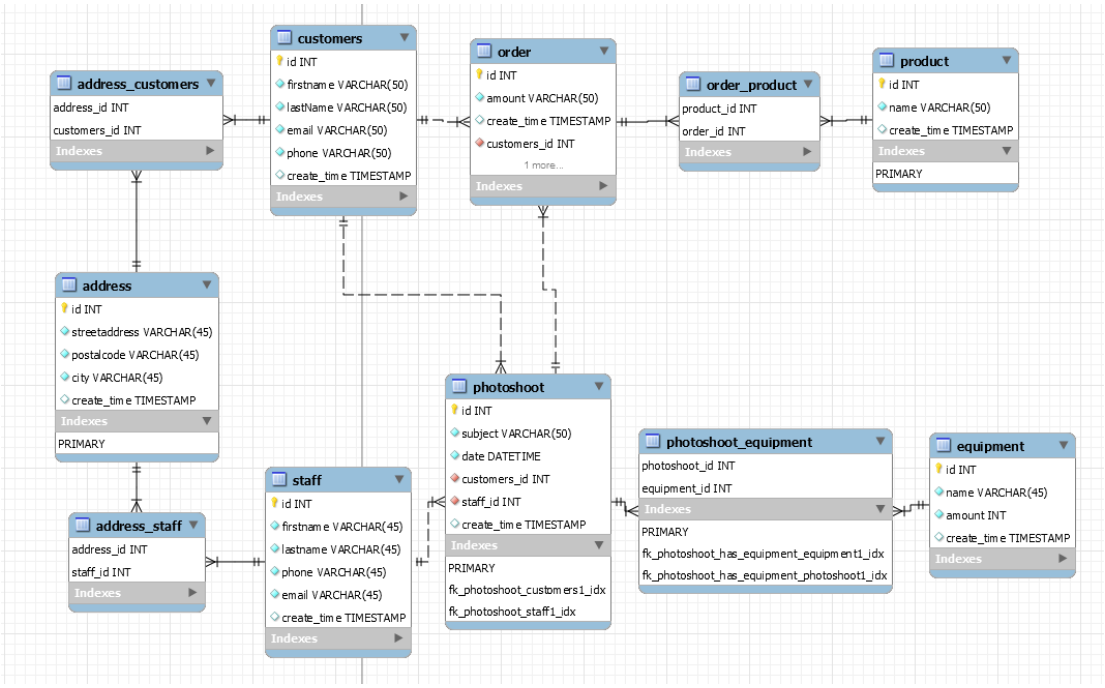
4.1 MySQL

MySQL on Ruotsalaisen yrityksen, MySQL AB:n vuonna 1995 julkaisema avoimen lähdekoodin relaatiotietokantaohjelmisto. Ohjelmiston kehitys alkoi jo vuonna 1994 Ruotsalaisen David Axmarkin ja Suomenruotsalaisen Michael Wideniuksen toimesta. MySQL on kirjoitettu käyttäen C- ja C++-ohjelmointikieliä ja sen SQL jäsentäjänä toimii Yacc. MySQL on yhteensopiva useiden alustojen kuten Linuxin, MacOS:n ja Microsoft Windowsin kanssa. Aluksi MySQL pohjautui toiseen tietokannanhallintaohjelmaan nimeltään Mini SQL. Mini SQL tuotti kuitenkin ongelmia hitautensa ja joustamattomuutensa vuoksi. David Axmark ja Michael Widenius päättivät tehdä uuden SQL-rajapinnan mutta pitivät käytössä Mini SQL:n ohjelmistorajapinnan. Tämän ansiosta monet ohjelmistokehittäjät pystyivät käyttämään MySQL-ohjelmistoa Mini SQL-ohjelmiston sijaan. Tällä oli suuri vaikutus MySQL-ohjelmiston suosioon, sillä MySQL oli kokonaan ilmainen, jopa kaupallisessa käytössä, toisin kuin Mini SQL. (Oreilly 2022.)

Asiakastietojen käsittelyyn tietokannaksi valittiin MySQL, koska se on ilmainen, monipuolinen ja laajalti käytetty. Relaatiotietokannassa tiedot on varastoitu omiin taulukoihin ja jokaiselle riville annetaan oma uniikki tunniste. Tunnisteiden avulla varmistetaan, etteivät kahden saman nimisen ihmisen tiedot mene sekaisin. Taulukoiden välille rakennetaan relaatiot, joilla määritellään tietojen keskinäinen sidonnaisuus. Nämä relaatiot ovat yleensä yksi yhteen (merkataan 1:1) tai yksi moneen (merkataan 1:N). Tietokantaa ja tietokantapalvelinta hallitaan MySQL Workbench-ohjelmalla. Sen avulla voidaan tietokannan taulukot, rivit, tunnisteet ja relaatiot luoda visuaalisena puuna, jonka ohjelma kääntää koodiksi ja ajaa palvelimelle. Visuaalinen lähestymistapa tietokannan luomiseen on todella yksinkertainen ja intuitiivinen. Kokonaisuuden näkeminen kerralla helpottaa ongelmien havaitsemista ja vähentää ulkoa muistamisen määrää. (MySQL 2022.)

4.2 Rakenne

Tässä opinnäytetyössä luotu relaatiotietokanta muodostuu kahdesta osasta: taulukoista, joissa säilytettävä tieto sijaitsee, sekä taulukoiden välisistä relaatioista. (KUVA 1.) Taulukko on yksinkertainen lista esimerkiksi asiakkaiden nimistä. Tietokannan taulukoita tehdessä on kuitenkin mahdollista, ja suotavaa, asettaa tunnisteen ja päivämäärän luominen automaattiseksi. Tämä tarkoittaa sitä, että aina kun taulukkoon lisätään tietoa, liitetään siihen aikaleima ja tietueelle luodaan automaattisesti uniikki tunniste. Aikaleima ja uniikki tunniste mahdollistavat tiedon säilyvyyden ja käytettävyyden. Toinen osa tietokantaa ovat taulukoiden väliset relaatiot eli suhteet. Yksi yhteen relaatio voi esimerkiksi olla henkilön nimi ja asiakasnumero. Yhdellä henkilöllä on yksi asiakasnumero ja yksi asiakasnumero vastaa yhtä asiakasta. Puolestaan yksi moneen relaatio voi olla esimerkiksi tilaukset ja asiakkaat. Yhdellä tilauksella on vain yksi asiakas, mutta yhdellä asiakkaalla voi olla monta tilausta. Joissakin tilanteissa kahden taulun välinen suhde voi olla monta moneen. Esimerkiksi tilaukset ja tuotteet. Yhdellä tilauksella voi olla monta tuotetta, ja yhdellä tuotteella voi olla monta tilausta. Tämä relaatio on kuitenkin sellaisenaan käyttökelvoton, sillä siitä on mahdotonta tietää, mikä tuote kuuluu millekin tilaukselle. Näissä tilanteissa voidaan ongelma kuitenkin ehkäistä tekemällä yksi lisätaulu tilausten ja tuotteiden väliin, joka vastaa oikeiden tilausten ja tuotteiden yhdistämisestä. Tärkeä osa tietokannan tiedon järjestelyä on primary key eli taulukon tiedon tunniste. Tunniste voi olla periaatteessa mikä tahansa taulukkoon tallennettu tieto, mutta yleisimmin se on automaattisesti luotu juokseva numero. Tunnisteelle on tärkeää, että se on uniikki. Yksinkertaistettuna esimerkkinä vodaan sanoa, että tietokannassa on monen asiakkaan tiedot, joiden etunimi on ”Matti”. Oikeaa etunimeä ei kuitenkaan voitaisi yhdistää oikeaan henkilöön, ellei sillä olisi mitään, joka erottaa sen toisista saman nimisistä henkilöistä. Kun etunimi tallennetaan tietokantaan uniikin tunnisteen kanssa, voidaan tunnisteen avulla löytää oikea henkilö. Kun tietokannassa on useita tauluja, paljon relaatioita ja asiakastietoja monen vuoden ajalta, ovat tunnisteet elintärkeitä tiedon käytettävyyden kannalta.



KUVA 1. Rakennetun tietokannan visuaalinen esitys.

4.3 Tietokannan asentaminen ja käyttäminen

MySQL-tietokannan asentaminen Windows-palvelimelle onnistuu helpoiten lataamalla ja asentamalla MySQL Installer -sovellus. Sovelluksen käyttäminen on yksinkertaista ja intuitiivista. Sovellus käynnistetään ja seurataan sen antamia ohjeita asennusprosessin loppuun asti. Asennettaessa käyttäjä voi valita palvelimelle haluamansa asetukset ja polun. Asetuksilla voi vaikuttaa muun muassa palvelimen käyttäjätasoihin, kirjautumiseen, asennustyyliin, yhteyksiin ja todennuksiin. Ongelmatilanteissa voi yrittää uudelleenasennusta tai etsiä apua MySQL-kotisivulta ja foorumilta. (MySQL 2022.)

Tietokannan käyttäminen on hieman asentamista monimutkaisempaa, mutta ei kuitenkaan vaikeaa. Tietokantapalvelin ja sovellus käynnistetään Windows-komentorivin kautta. Sovellus luo automaattisesti yhteyden tietokantaan. Kun yhteys on muodostettu, voi tietokantaa lukea tai muokata, riippuen käyttäjälle annetuista oikeuksista. (Phoenixnap 2022.) Tietokantaa on tietenkin mahdollista käyttää myös tässä opinnäytetyössä luodun sovelluksen avulla. Sovelluksen kautta tietoja on helppo lukea ja muokata, kunhan palvelimeen saadaan yhteys internetin kautta. Sovellukselle voidaan lähettää tietoja muodossa JSON, ja tietojen perusteella ohjelma tekee halutulle tiedolle halutun muutoksen. Tietojen lähettäminen täytyy tällä hetkellä tehdä kolmannen osapuolen ohjelmalla, kuten Postmanilla, sillä käyttöliittymää ei sovelluksessa vielä ole. Tietokannan rakennetta ei voi sovelluksen avulla hallita,

mutta se onnistuu helposti MySQL workbench -sovelluksella, jonka avulla tietokanta alun perin luotiin. Jos taulukoihin tekee muutoksia, täytyy muistaa päivittää myös sovellus vastaamaan uutta tilannetta. Muutoin tietojen tallentaminen oikeisiin kohtiin ei onnistu.

4.4 Ylläpito

Tietokannan ylläpitäminen on todella tärkeä osa, tietokannan toiminnan varmistamiseksi. Ilman ylläpitoa tietokannan tarvitsema levytila kasvaa ja sen toiminta hidastuu. Ylläpito voidaan jakaa neljään osaan: hakemiston eheytys, loki tiedostojen ylläpito, tiedostojen eheytys ja yhtenäisyyden tarkistaminen. Listaan voidaan myös lisätä varmuuskopiointi. Varmuuskopiointi ei suoranaisesti ole tietokantaa ylläpitävää toimintaa, mutta sen vaikutus on saman kaltainen. (Officetools 2021.)

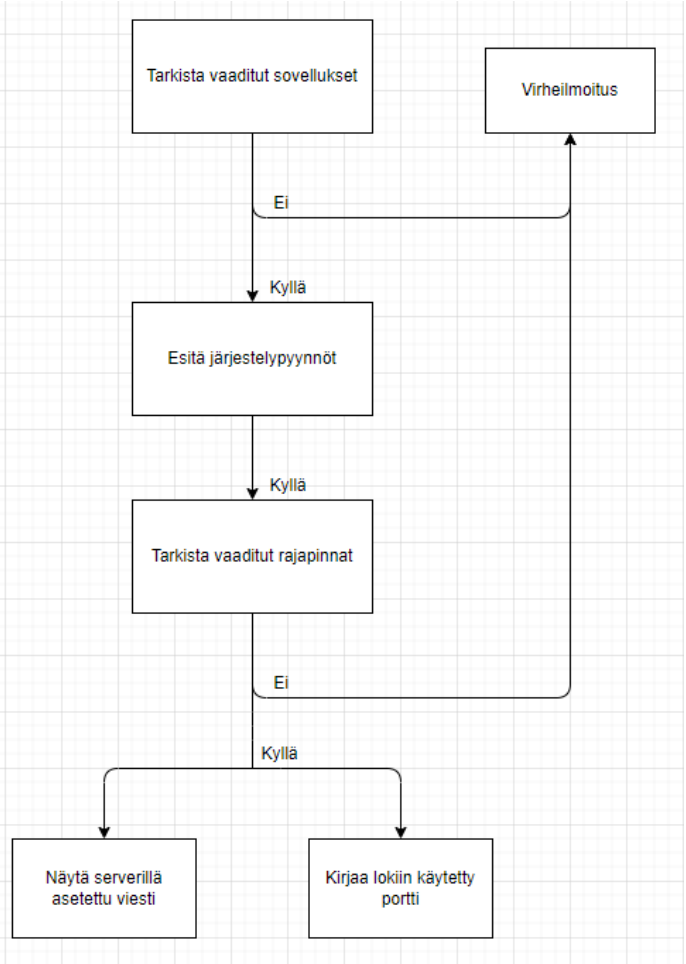
Tietojen hakeminen tietokannasta vaikuttaa suoraan sen hakemistoon. Levylle tallennettu uusi tieto tallentuu sinne, mihin se sillä hetkellä mahtuu. Tästä seuraa vääjäämättä tiedon pirstaloituminen. Hakemiston eheytyksenä on tarkoitus siirtää toisiinsa liittyvä tieto yhtenäiseen paikkaan. Kun kaikki tarvittava tieto sijaitsee yhdessä paikassa, on sen hakeminen ja käyttäminen tehokasta. (Officetools 2021.)

Tietokannassa on lokitiedosto, johon tallentuu automaattisesti kaikki tietokannassa tapahtuva toiminta. Lokitiedot ovat todella hyvä keino palauttaa tietokanta aikaisempaan tilaan, vaikka tietokanta korruptoituisi kokonaan. Vaikka palauttaminen onnistuu myös varmuuskopiolla, voi lokitiedon käyttäminen olla parempi vaihtoehto. Kyseessä voi olla tilanne, jossa varmuuskopio tallennetaan kerran päivässä, mutta lokitiedot tallentuvat automaattisesti jokaisen muutoksen jälkeen. Tämän ansiosta ei menetetä käytännössä yhtään tietoja tai työtä. Tietokannan käyttämisen aikana lokitiedostot kuitenkin kasvavat. Mitä enemmän tiedostoja on ja mitä suurempia ne ovat, sitä enemmän täytyy tietokannan käydä läpi tietoja, kun muutoksia tehdään. Vaikka lokitiedot normaalisti tehostavat tietokannan toimintaa huomattavasti, voivat ne myös haitata sitä. Tämän takia on tärkeää karsia pois turhat tiedot ja tiivistää lokitietoja mahdollisuuksien mukaan. (Officetools 2021.)

MySQL-tietokannan toiminnallisuuden ylläpitäminen on helppoa. Kaikkiin aikaisemmin esiteltyihin toimintoihin on olemassa valmis komento. (Fromdual 2022.) Tietokannan ylläpitämisen vastuu kuuluu tietokannan omistajalle ja tämän opinnäytetyön tapauksessa tietokannan omistaja on työn tilannut asiakas. Markkinoilla on useita konesalipalveluita, joille palvelimen ja tietokannan ylläpito on mahdollista ulkoistaa. Ulkoistamisella tietokannan omistaja voi vähentää omaa vastuutaan ja työkuormaansa.

5 SOVELLUS

Sovelluksessa on viisi pääosaa: server, package, package-lock, node_modules ja app. Kun sovellus käynnistetään, Server-osa (KUVA 2.) varmistaa, että kaikki tarvittavat osat löytyvät ja ovat käytettävissä. Ensin tarkistetaan Express-sovelluksen olemassaolo. Tämän jälkeen esitetään järjestelypyynnöt, jotka varmistavat, että tulevat pyynnöt ovat oikean kaltaisia. Seuraavana tarkistetaan tarvittavien rajapintojen olemassaolo. Lopuksi palvelimelle lähetetään ennalta asetettu viesti, ja lokiin kirjataan käytettävä portti. Jos jokin vaadituista osista puuttuu, tulee käyttäjälle näkyviin virheilmoitus.

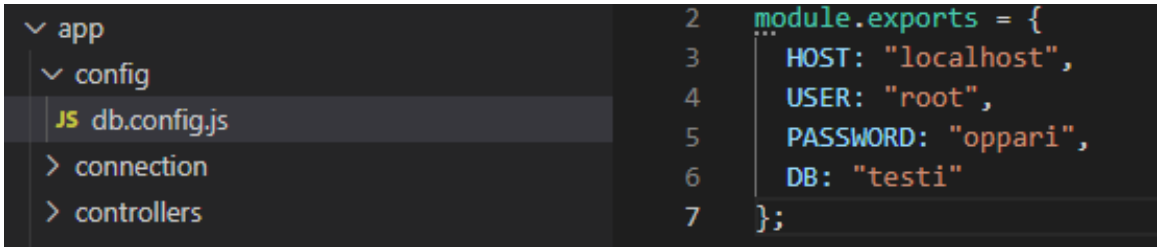


KUVA 2. Server-osan lohkodiagrammi

Package-osa määrittelee sovellukseen liittyvää informaatiota. Tällä informaatiolla tarkoitetaan esimerkiksi ominaisuuksia, sovelluksen versiota, tekijää ja lisenssiä. Package-lock puolestaan määrittelee projektin riippuvaisuuksia. Vaikka kaikki riippuvaisuudet on mahdollista sisällyttää package-osaan, on osien erottelu järkevää, jos riippuvaisuuksia on paljon. Node_modules pitää sisällään kaikki ulkoiset

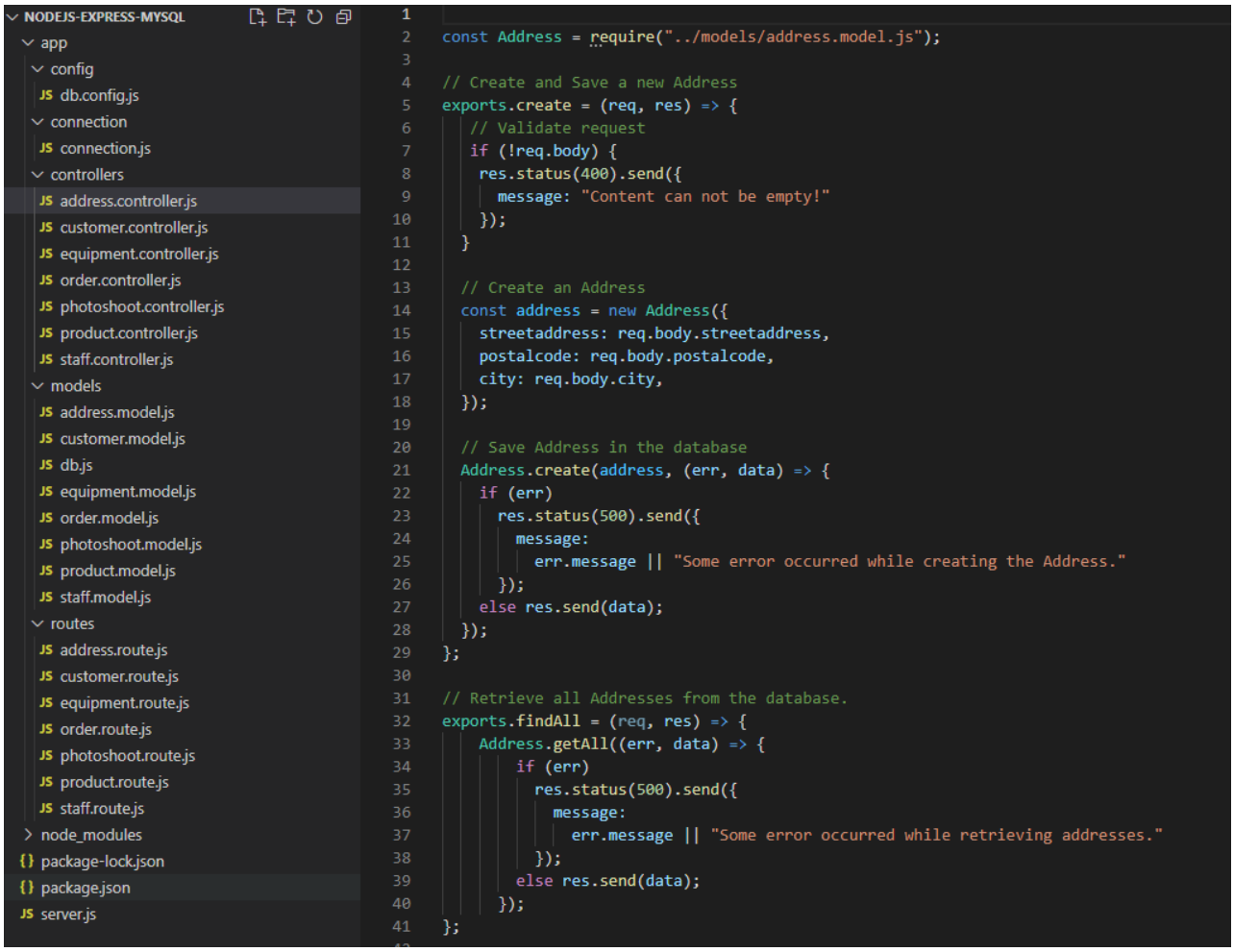
osat ja sovellukset, jotka on aiemmin määritelty. Node_modules -osa luodaan automaattisesti, kun sovellus asennetaan.

App-osan alla on suurin osa sovelluksen toiminnallisuutta. App pitää sisällään viisi osaa. Osat config ja connection vastaavat yhteyksistä sovelluksen eri osien ja palvelimen välillä. Yhteyden muodostamisen mahdollistamiseksi palvelimelle on määritelty nimi ja tietokannalle on määritelty nimi, käyttäjänimi sekä salasana. Määritellyt tiedot syötetään sovellukselle config- ja connection-osioihin. (KUVA 3.) Aina kun tietokannalle tai palvelimelle määriteltyjä tietoja päivitetään, tulee sovelluksenkin tiedot päivittää. Jos tiedot eivät vastaa toisiaan, on yhteyden muodostaminen mahdotonta. Sovellukselle syötettyjä tietoja vaihtamalla on myös mahdollista yhdistää toiseen tietokantaan ja palvelimeen.



KUVA 3. Sovellukselle syötetyt tiedot yhteyden muodostamiseksi.

App-kansion alla olevat osat controllers, models ja routes ovat sovelluksen rajapintoja. Rajapintojen avulla eri sovellukset keskustelevat keskenään. Osien välinen keskustelu mahdollistaa sovelluksen toimimisen. Jokaiselle tietokannan taulukolle, jonka tietoja halutaan lukea ja muokata, täytyy luoda rajapinnat. (KUVA 5.) Jos taulukoiden nimiä muutetaan tai taulukoiden rivejä muokataan, täytyy rajapinnatkin päivittää vastaamaan uutta tilannetta.



KUVA 4. Esimerkki rajapinnoista.

5.1 Toiminnot

Sovelluksessa on neljä (KUVA 5.) pääasiallista toimintoa tietokannan tietojen lukemiseen ja muokkaamiseen. Create-toiminnolla on mahdollista luoda uusi rivi tietoja haluttuun taulukkoon. Retrieve-toiminnolla voidaan tietokannasta hakea tietoa halutusta taulukosta. Retrieven avulla on mahdollista hakea kaikki taulukon tiedot kerralla tai yksittäinen rivi tunnisteen perusteella. Tietokannassa olevia tietoja voidaan päivittää Update-toiminnolla. Tähän tarvitaan päivitettävän rivin tunniste. Viimeisenä toimintona on tietojen poistaminen. On mahdollista poistaa kaikki taulukon tiedot kerralla tai yksittäinen rivi tunnisteen perusteella. Tietojen poistaminen taulukoista ei kuitenkaan ole suositeltavaa, ellei se ole jostain syystä pakollista. Tietokannan tietoja muokkaavat toiminnot tarvitsevat aina muokattavan tiedon tunnisteen. Tämä riippuvaisuus on hyvä lisäesimerkki siitä, kuinka tärkeä osa tietokantaa tunnisteet ovat.

```
1
2   module.exports = app => {
3       const customers = require("../controllers/customer.controller.js");
4
5       // Create a new Customer
6       app.post("/customers", customers.create);
7
8       // Retrieve all Customers
9       app.get("/customers", customers.findAll);
10
11      // Retrieve a single Customer with customerId
12      app.get("/customers/:customerId", customers.findOne);
13
14      // Update a Customer with customerId
15      app.put("/customers/:customerId", customers.update);
16
17      // Delete a Customer with customerId
18      app.delete("/customers/:customerId", customers.delete);
19
20      // Delete all Customers
21      app.delete("/customers", customers.deleteAll);
22    };
```

KUVA 5. Esimerkki toiminnoista Customers-taulukolle.

5.2 Testaaminen

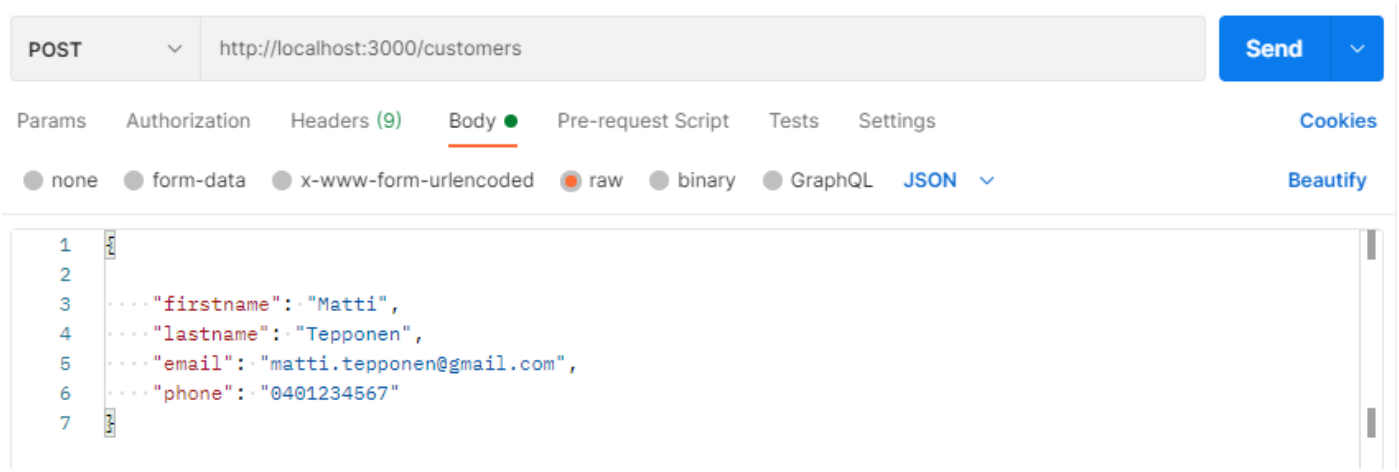
Sovelluksen testaamisessa oli kaksi tärkeää osaa. Aluksi täytyi varmistaa, että tietokantaan yhdistäminen onnistuu. Tämä oli testattavissa helposti. Sovellus ilmoittaa yhteyden muodostamisesta tai mahdollisesta virheestä. (KUVA 6.) Kun käyttäjänimi, salasana ja portti oli määritetty oikein, saatiin palvelimeen yhteys.

```
Server is running on port 3000.
Successfully connected to the database.
```

KUVA 6. Sovelluksen palauttama viesti käynnistämisen jälkeen.

Toinen tärkeä osa oli toiminnallisuuden ja rajapintojen testaaminen. Koska sovelluksessa ei vielä ole käyttöliittymää, testaamiseen käytettiin kolmannen osapuolen ohjelmaa Postman. Aluksi tietokantaan

haluttiin lähettää tietoja. (KUVA 7.) Poluksi asetettiin muokattavan taulukon polku palvelimella ja lähetyksen muodoksi POST. Tiedoston muotona oli JSON, sillä sovellus oli rakennettu käsittelemään juuri kyseistä tiedostomuotoa. Halutut tiedot asetettiin otsikoiden taakse ja tiedot lähetettiin. Jos tietojen lähettäminen onnistuu, tulee vastauksena varmistusviesti vastaanotetuista tiedoista. Lähetyksen epäonnistuessa tulee virheilmoitus.



KUVA 7. Tietojen lähettäminen tietokantaan.

Lähetettyjä tietoja päästään tarkastelemaan selaimen kautta. Työn teon aikana osoite oli http://localhost:3000/(haluttu taulukko), mutta tämä tietenkin vaihtuu virallisen palvelimen osoitteeseen myöhemmin. Kun selaimella luetaan customers-taulukkoa, nähdään kaikki sinne lähetetyt tiedot. (KUVA 8.) Tiedot ovat yhtenä jonona ilman mitään kunnollista jäsentelyä, mutta se ei haittaa tässä vaiheessa. Tiedon jäsentelyn hoitaa tulevaisuudessa käyttöliittymä. Kokeilun vuoksi taulukkoon on tallennettu samat tiedot kolmesti, mutta se ei tuottanut ongelmia automaattisesti muodostuvan uniikin tunnisteen ansiosta.

[{"id":1,"firstname":"Matti","lastName":"Tepponen","email":"matti.tepponen@gmail.com",
"phone":"0401234567","create_time":"2021-03-18T17:23:06.000Z"},
{"id":2,"firstname":"Matti","lastName":"Tepponen","email":"matti.tepponen@gmail.com","
phone":"0401234567","create_time":"2021-09-23T16:16:36.000Z"},
{"id":3,"firstname":"Kaija","lastName":"Koo","email":"testi.testi@gmail.com","phone":"
0401234567","create_time":"2021-09-24T10:58:09.000Z"},
{"id":4,"firstname":"Matti","lastName":"Tepponen","email":"matti.tepponen@gmail.com","
phone":"0401234567","create_time":"2022-03-03T23:30:03.000Z"}]

KUVA 8. Tiedot customer-taulukossa

6 JATKOKEHITYS

Laitteiden, palveluiden ja sovelluksien kehittäminen on jatkuvaa. Tehokkuutta ja toiminnallisuutta voi aina parantaa. Ohjelmointikielet kehittyvät ja rikolliset löytävät aikaisemmin tuntemattomia tietoturva-aukkoja jatkuvasti. Tässä opinnäytetyössä luotu palvelinsovellus on hyvä pohja sovellukselle, mutta jatkokehitystä tarvitaan vielä ennen sovelluksen kaupallista käyttöä.

6.1 Front-End

Front-endillä tarkoitetaan yleisesti ohjelman tai sovelluksen käyttöliittymää, jolla sovellusta ohjataan. Yksinkertaistetusti voidaan ajatella, että jos ohjelma olisi auto, olisi back-end sen moottori ja front-end sen ratti. Tässä työssä ohjelmaan ei kuitenkaan tehty front-endiä, sillä työ määrä olisi paisunut liian laajaksi. Tästä syystä toimintojen käyttäminen on vaivalloista ja rajoitettua. Jatkokehityksessä front-end on tärkein työvaihe.

Yleisimmät front-end ohjelmointikielet web-sovelluksille ovat HTML, CSS ja JavaScript. (Flatironschool 2022.) Tässä tapauksessa luonnollisin vaihtoehto on JavaScript, koska back-end luotiin sitä käyttäen. Kun koko sovellus on luotu yhdellä ohjelmointikielellä, ei tarvita rajapintoja tai muita välikäsiä front- ja back-endin välille. Tämä yksinkertaistaa sovellusta ja pienentää virheiden mahdollisuuksia. On myös suositeltavaa sisällyttää front-end samaan sovellukseen ja ylläpitää sitä samalla palvelimella. Tämä helpottaa sovelluksen toimintaa, kun kahden erillisen sovelluksen ei tarvitse keskustella keskenään. Ylläpito on myös helpompaa, kun kaikki sijaitsee yhdessä paikassa. Tietenkin täytyy muistaa, että muiden ohjelmointikielien ja toteutustapojen käyttäminen on täysin mahdollista. Toteutustavan valitseminen on ohjelmoijan päätettävissä.

Front-end tulee olemaan asiakkaalle ja työntekijälle näkyvä osa. Sen kautta asiakkaan pitää päästä käsiksi omiin kuviinsa ja tekemään niistä tilauksia. Tämä voidaan hoitaa joko niin, että asiakas käyttää hänelle luotua koodia kuvauksesta, tai käyttäjätunnusta. Käyttäjätunnuksen avulla olisi myös helppoa seurata omia tilauksia ja tehdä niihin tarvittaessa muutoksia. Työntekijällä tietenkin toiminnallisuuksia olisi enemmän. Työntekijöiden tulee päästä käsiksi kaikkiin kuviin ja asiakastietoihin, joita tietokantaan on tallennettu. Nähtävissä tulee olla myös työjono ja käytettävissä oleva kalusto.

6.2 Tietoturva

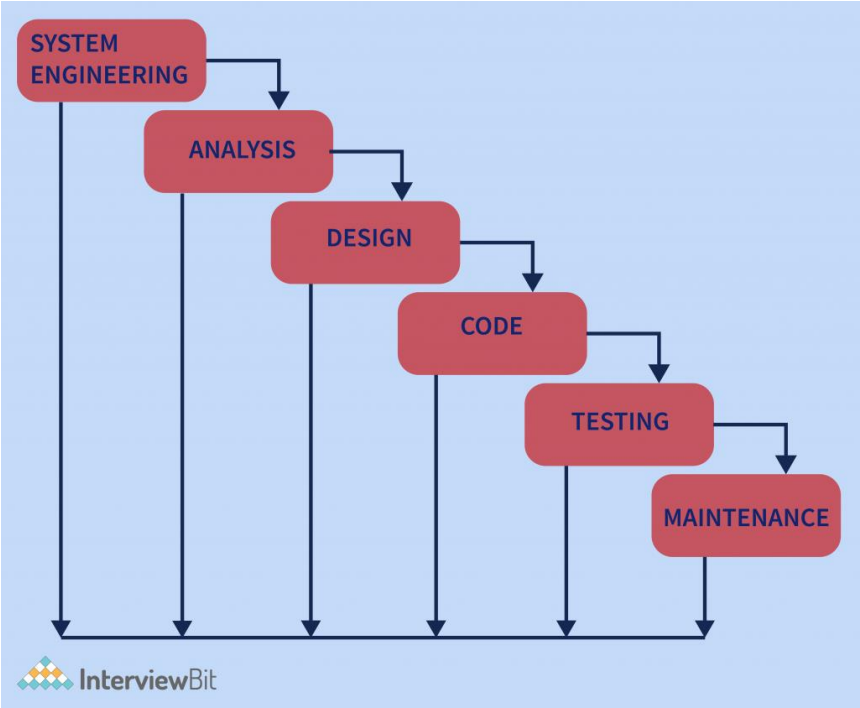
Ensimmäiset virukset ovat olleet olemassa kauemmin kuin internet. Ensimmäinen viruksen tavoin toimiva ohjelma tehtiin vuonna 1971. Virus oli itsestään kopioituva ohjelma nimeltään Creeper. Creeper luotiin osana tietoturvatestiä, jossa haluttiin nähdä, onko itsensä kopioiva ohjelma mahdollista tehdä. (Kaspersky 2022.) Toisin kuin Creeper, nykypäivän virukset luodaan tarkoituksella haitallisiksi. Niillä halutaan yleensä joko varastaa uhrin henkilökohtaisia tietoja, tai käyttää uhrin tietokonetta osana hyökkäystä. Tästä syystä verkkosivujen ja sovellusten tietoturvallisuuden merkitys on suurempi kuin koskaan.

Tässä työssä ohjelmaan ei ole otettu huomioon tietoturvaa, eikä sovellusta tule käyttää, ennen kuin tietoturva on asianmukaisesti huomioitu. Kun sovelluksen tietokantaan kerätään asiakastietoja, siitä tulee henkilörekisterilain alainen asiakasrekisteri (Finlex 2021). Henkilörekisterilaki määrää käytännöt tietojen keräämisestä, säilyttämisestä, käyttämisestä sekä poistamisesta. Tämän työn tapauksessa rekisterissä ei ole pelkästään henkilötietoja, vaan myös ihmisten valokuvia. Tästä syystä hyvän tietoturvan ja tietokannan oikeaoppisen hallinnan merkitys on entistä suurempi. Ensimmäinen ja helpoin tapa tietoturvan saavuttamiseksi on salasanasuojattujen käyttäjätilien lisääminen. Kun asiakkaille, työntekijöille, tietokannan ylläpitäjälle ja tietokannan omistajalle on eri tasoiset käyttäjätilit, on helppo varmistaa, etteivät käyttäjät vahingossa saa käsiinsä tietoa, joihin heillä ei ole oikeuksia.

Nykyään palvelunestohyökkäyksen voi ostaa internetistä helposti ja halvalla. (Securelist 2022.) Hyökkäyksiä käytetään yleensä palvelujen lamauttamiseen ja tietoturvavastaavien harhauttamiseen. Tämän takia hyökkäyksiltä suojautuminen on tärkeä osa tietoturvaa ja palvelujen toiminnan varmistamista. Tämä onnistuu helpoiten hankkimalla ulkoinen palvelunestohyökkäyksien estämiseen erikoistunut palvelu omalle verkkosivulle. Jos palvelimen ylläpitäminen on ostettu ulkoiselta palvelun tarjoajalta, voi palveluun kuulua tietoturvan ylläpito. Tietoturva on erittäin laaja ja koko ajan muuttuva ala, jossa uusia haavoittuvaisuuksia löytyy ja paikataan jatkuvasti. Tämän takia tietoturvan hoitamiseen kannattaa palkata alan ammattilainen.

6.3 COCOMO-malli

Ohjelmistojen kehittämisessä ja ylläpitämisessä tärkeä osa ovat myös kustannukset. Ohjelmistotuotannon kustannuksien laskemiseksi voidaan käyttää esimerkiksi COCOMO-mallia (KUVA 9). Alkuperäisen COCOMO-mallin esitteli vuonna 1981 Barry Boehm, josta hän julkaisi päivitetyn version COCOMO II vuonna 1991. Malli tarjoaa useita eri kaavoja kustannustekijöiden arviointiin ja onkin nykyään yksi käytetyimpiä kustannusmalleja maailmalla. COCOMO mahdollistaa myös ohjelmistoprojektin keston arvioinnin ja kaikki sen algoritmit ovat vapaasti saatavilla. (Archive 2022)



KUVA 9. COCOMO II malli.

6.4 Sovelluksen kaupallinen käyttö

Työn alkuvaiheilla yhtenä ideana oli myös valmiin sovelluksen kaupallistaminen. Sovellusta myytäessä täytyy muistaa ottaa huomioon lakitekniset asiat ja tietoturvaan kuuluvat seikat. Tällä hetkellä sovellus on mahdollista myydä puhtaana koodina ja sen mukana voidaan myydä eri määrä oikeuksia. Jos oikeuksia myydään rajallisesti, ostaja saa käyttää sovellusta tai koodia mutta ei saa muokata sitä. Koodin mukana voidaan myös myydä kaikki oikeudet. Kaikki oikeudet ostettuaan asiakas voi vapaasti jatkokehittää koodia ja käyttää sitä omiin kaupallisiin tarkoituksiinsa. Kun front-end ja tietoturva ovat kunnossa, on sovellusta mahdollista myydä myös palveluna. Palvelun ylläpitäminen vaatii työtä, mutta

se voi myös olla hyvä jatkuva tulonlähde. Kun oikeuksia tai lähdekoodia myydään eteenpäin, on aina mahdollista, että ostaja väärinkäyttää ostamiaan asioita. Käyttöoikeuksien avulla voi ostaja päästä käsiksi esimerkiksi toisen yrityksen asiakastietoihin. Hyvin suunnitellussa ja toteutetussa sovelluksessa edellä esitetyn kaltaisen tietovuodon ei tietenkään pitäisi olla mahdollista, mutta sovelluksissa on aina vikoja. Jos lähdekoodi myydään eteenpäin ja se joutuu vääriin käsiin, vaarantuu koko sovellus. Asiantuntijan on mahdollista löytää lähdekoodista kaikki mahdolliset aukot sovelluksen tietoturvassa. Tästä voi seurata suuria tietovuotoja ja sovelluksen väärinkäyttöä.

7 YHTEENVETO

Tämän työn aikana luotiin toimiva tietokanta ja palvelimen back-end valokuvaamon web-sovellukselle. Työssä käytettiin apuna aikaisempaa opiskelun aikana kerättyä tietotaitoa ohjelmoinnista sekä internetistä löytyneitä ohjeita. Alussa opinnäytetyöhön kuului back-end, tietokanta, front-end sekä alustavaa tietoturvaa. Opinnäytetyön aikana kävi kuitenkin ilmi, että aika ei riittänyt kaikkien osien tekemiseen ja työn määrää piti rajata. Liian optimistinen asenne työmäärään johtui omasta kokemattomuudestani ohjelmistokehittämisen saralla. Tämän työn aikana opin ymmärtämään ohjelmistokehittämisen prosessia kokonaisuutena paljon paremmin. Taitoni ohjelmoijana myös kehittyivät huomattavasti ja opin paljon alaan liittyvästä tiedonhausta. Internetistä löytyneet esimerkit olivat hyviä, mutta eivät suoraan käytettävissä. Kaikki piti muokata tähän työhön sopivaksi ja toimivaksi kokonaisuudeksi. Työn aikana koodissa oli paljon virheitä, jotka piti korjata. Onneksi tämän kaltaisten sovellusten tekeminen JavaScriptillä on todella yleistä, joten apua ongelmiin löytyi helposti. Iso osa opinnäytetyötä oli myös työn aikataulutus, tai tässä tapauksessa sen puute. Alkuperäinen aikataulu ei pitänyt ja työn valmistuminen viivästyi paljon. Aikataulun venymiseen oli paljon hyviä syitä, mutta oma laiskuuteni oli yksi iso tekijä. Viivästyksien jälkeen sovimme ohjaavan opettajani kanssa työn etenemiseen tarkat päivämäärät ja pidimme työn etenemisen seuranta palavereita noin joka toinen viikko. Näillä keinoilla työn tekeminen saatiin organisoitua hyvin. Tämän ansiosta opin sen, että liian ympäripyöreät aikataulut eivät sovi minulle. Tulevaisuudessa osaan ottaa huomioon oman työskentelytapani.

LÄHTEET

Archive 2022. *Software Engineering Economics*, Saatavissa: https://archive.org/details/softwareengineer0000boeh/mode/2up. Viitattu 11.5.2022.

Expressjs 2021. *Companies using Express in prduction.* Saatavissa: https://expressjs.com/en/resources/companies-using-express.html. Viitattu 16.03.2022.

Finlex 2021. *Henkilörekisterilaki 471.* Saatavissa: https://www.finlex.fi/fi/laki/alkup/1987/19870471. Viitattu 16.03.2022.

Flatironschool 2022. *Front End vs. Back End Development* Saatavissa: https://flatironschool.com/blog/front-end-vs-back-end-development/#:~:text=Key%20takeaway%20%E2%86%92%20HTML%2C%20CSS,three%20languages%20and%20JavaScript%20frameworks. Viitattu 07.04.2022.

Fromdual 2022. *Typical automated MySQL Maintenance Jobs*. Saatavissa: https://fromdual.com/typical-automated-mysql-maintenance-jobs. Viitattu 07.04.2022.

Github 2021. *Express History*. Saatavissa: https://github.com/expressjs/express/blob/master/History.md#4140--2016-06-16. Viitattu 18.03.2022.

JavaScript 2021. *JavaScript.* Saatavissa: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Viitattu 15.12.2021.

Kaspersky 2022. *A Brief History of Computer Viruses* Saatavissa: https://www.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds#:~:text=As%20noted%20by%20Discovery%2C%20the,by%20Bob%20Thomas%20of%20BBN. Viitattu 20.03.2022.

MySQL 2022. *MySQL kotisivut.* Saatavissa: https://www.mysql.com/. Viitattu 07.04.2022.

Node.js 2021. *Node.js kotisivut*. Saatavissa: https://nodejs.org/en/. Viitattu 15.12.2021.

Officetools 2021. *Database Maintenance Explained*. Saatavissa: https://www.officetools.com/knowledgebase/database-maintenance-explained/. Viitattu 15.12.2021.

Oreilly 2022. *MySQL History*. Saatavissa: https://www.oreilly.com/library/view/understanding-mysql-internals/0596009577/ch01.html. Viitattu 07.04.2022.

Phoenixnap 2022. *How to Connect to MySQL from Windows Command Line.* Saatavissa: https://phoenixnap.com/kb/connect-to-mysql-windows-command-line#:~:text=Enter%20mysql.exe%20%2Du-root%20%2D,connect%20to%20the%20MySQL%20server. Viitattu 09.05.2022.

Postman 2021. *Postman kotisivut.* Saatavissa: https://www.postman.com/. Viitattu 15.12.2021.

Securelist 2022. *The Cost of Launching a DDoS Attack.* Saatavissa: https://securelist.com/the-cost-of-launching-a-ddos-attack/77784/. Viitattu 09.05.2022.

Springboard 2022. *The History of JavaScript.* Saatavissa: https://www.springboard.com/blog/data-science/history-of-javascript/. Viitattu 3.6.2022

Visual Studio Code 2021. *Visual Studio Code kotisivut.* Saatavissa: https://code.visualstudio.com/. Viitattu 15.12.2021.

W3techs 2021. *Usage statistics of JavaScript as client-side programming language on websites,* Saatavissa: https://w3techs.com/technologies/details/cp-javascript/. Viitattu 18.10.2021.