Asiakkuuksien hallintajärjestelmä soveltuvuusselvitys

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tieto- ja viestintätekniikan insinööri (AMK)

Syksy 2022

Topi Jantunen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tieto- ja viestintätekniikan koulutus Tiivistelmä
Tekijä Topi Jantunen Vuosi 2022
Työn nimi Asiakkuuksien hallintajärjestelmä soveltuvuusselvitys
Ohjaaja Kuittinen Petri

Työn tarkoitus oli tehdä CRM-järjestelmä soveltuvuusselvitys Tavastia Software Oy:lle. Tavoitteina oli tehdä järjestelmälle palvelinpuoli, verkkosivujen pohjat, käyttäjien luonti, järjestelmään kirjautuminen, organisaatioiden luominen, mahdollisten asiakkaiden luominen, niiden esittäminen ja datan muokkaaminen web selaimessa. Työ sisältää rakenteen ja pohjan käyttäjähallintajärjestelmälle, jota pystyy käyttämään useampi yritys samaan aikaan ilman pääsyä toistensa tietoihin.

Projektissa otettiin huomioon yrityksen käyttämät teknologiat ja jatkokehityksen soveltuvuuden mahdollisuus. Tavoitteena siis oli saada helposti muokattava pohja, jota yritys pystyy käyttämään ja muuttamaan omien tarpeittensa mukaan. Työ aloitettiin määrittelemällä tarpeet, suunnittelemalla toimiva kokonaisuus ja pilkkomalla työmäärä pienempiin osiin, joiden kanssa oli helpompi aloittaa työskentely.

Lopputuloksena oli toimiva palvelinpuoli CRM-järjestelmään. Data liikkui halutulla tavalla ja tieto esitettiin sille oikeissa osioissa. Sivuston käyttäjäpolku on suunniteltu ja toteutettu, määriteltyjen parametrien kautta. Sovellusta on mahdollista lähteä jatkokehittämään ja viimeistelemään ulkoasua haluttuun muotoon. Projektin esittelyn jälkeen työnohjaaja oli tyytyväinen lopputulokseen.

Avainsanat CRM, Django, websovelluskehitys

Sivut 32

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information and Communication Technology
Author Topi Jantunen
Subject Customer relationship management proof of concept
Supervisors Kuittinen Petri

Abstract
Year 2022

The purpose of the project was to make customer relationship management system proof of concept for Tavastia Software Oy. The goals were to create a back end for the system, basic front end, user creation and logging functions, creating organization, creating, editing, and presenting potential customers on web browser. The work includes structure and basis for user management system that can be used by several customers at the same time without the companies having access to each other's information.

The project considered the technologies used by Tavastia software and the platform used is already familiar to the company. The goal was to have an easily customizable base that the company can use and change according to its own wishes. The work started by planning the project as whole and breaking the workload into smaller parts, which were easier to start working with. The result was a functional back end for the CRM system.

The result was a functional back end CRM system. The data moved as desired, and the information was presented to in the right sections. The site's user path is planned and implemented through defined parameters. It is possible to continue developing the application and finalize the appearance to the desired outlook. After the presentation of the project, the work supervisor was satisfied with the result.

Keywords CRM, Django, web application development

Pages 32

Sisällys

1 Johdanto 1

1.1 Työnantaja 1

1.2 Tarkoitus 1

1.3 CRM 2

1.4 Tavoitteet 3

1.5 Rajaus 3

1.6 Työn kokonaisuus 4

2 Django historia 4

2.1 MVC ja MVT 5

2.2 Sovelluskehykset 7

2.2.1 Django 7

2.2.2 Ruby on Rails 8

2.2.3 Vertailu Django ja Ruby on Rails 8

2.3 React 9

2.3.1 Vue 10

2.3.2 Vertailu React ja Vue 10

2.3.3 JQuery 10

2.4 Yhden sivun sovellukset ja monen sivun sovellukset 11

2.4.1 SPA (Single page application): 11

2.4.2 MPA (Multi page application): 11

2.4.3 SPA ja MPA vertailu 12

2.5 Verkkopalvelimen valinta 13

3 Toteutus 15

3.1 Sovelluksen toteutuksen vaiheet 16

3.2 Kaaviokuva sovelluksesta 17

3.3 Applikaation esittely 19

4 Johtopäätökset 29

4.1 Tavoitteet ja lopputulos 29

4.2 Onnistumiset ja puutteet 29

4.3 Työn jatkokehitys 30

4.4 Mitä opin 30
4.5 Opinnäytetyön ohjaus 31
Lähteet 32

Sanasto

CRM	Käyttäjähallinta järjestelmä
Asiakaspuoli	Kaikki koodi, joka ajetaan selaimessa, ja käyttöliittymä.
Palvelinpuoli	Taustalla toimiva järjestelmä, käyttäen mm. tietokantoja.
Liidi	Mahdollinen asiakas yritykselle.
Verkkokehys	Ohjelmisto, joka nopeuttaa verkkosovellusten ja verkkosivujen kehittämistä ja vähentää matalan tason työtä.
Django	Verkkokehys, joka käyttää python ohjelmointi kieltä.
GitHub	Versionhallinta ohjelma.

1 Johdanto

Opinnäytetyön tavoitteena oli luoda toimiva käyttäjähallinta järjestelmän pohja yritykselle. Ohjelmaan kirjataan yritysten tietoja, jotka ovat mahdollisia asiakkaita ja käyttäjät pystyvät luomaan uusia kirjauksia yrityksistä. Työn alustana toimi Django verkkokehys alusta. Django on pythonilla toimiva verkkosovelluskehitys alusta.

Päädyimme työnantajan kanssa siihen tulokseen, että työni keskittyy ohjelman palvelinpuolen rakentamiseen. Tein työn yritykselle työharjoittelun ohella enkä ollut pientä kouluprojektia lukuun ottamatta aiemmin harrastanut verkkosovelluskehitystä.

1.1 Työnantaja

Tavastia Software Oy on ohjelmistokehitysalan yritys, joka tarjoaa palveluinaan mm. web-kehitystä kuten verkkokaupan tai kotisivujen kehittämistä ja ylläpitoa. Yrityksellä on myös tarjolla mobiiliapplikaatio- ja muita ohjelmistokehityspalveluita asiakkaiden tarpeitten mukaan. Perehdyttäjänä ja ohjaajana toimi Tavastia software Oy:n teknologiajohtaja. Hänellä oli entuudestaan kokemusta Djangon käytöstä ja käännyinkin hänen puoleensa monissa ongelmatilanteissa. Työsuhteen loputtua lopputuotoksen katselmoi myös yrityksen toimitusjohtaja ja hän antoi oman palautteensa lopullisesta tuotteesta.

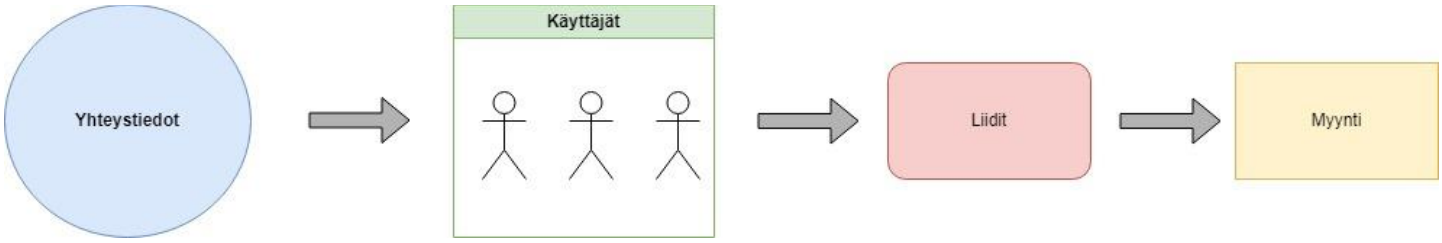
1.2 Tarkoitus

Työni keskittyy siihen, kuinka sovellus toteutettiin, toteutuksen vaiheisiin ja teknologioihin, joita verkkosovelluskehitys sisältää. Vertailen myös lyhyesti työssäni verkkosovelluskehyksiä ja niiden eri toimintamalleja. Dokumentoin myös oman projektini etenemisen ja siihen liittyvät vaiheet. Käyn läpi myös Djangon periaatteita ja verkkosovelluskehysten kanssa työskentelyn hyötyjä.

1.3 CRM

CRM eli asiakkuudenhallinta on yleistermi yritysten menetelmiin hallita ja analysoida asiakasvuorovaikutusta. Tietojärjestelmät ovat iso osa CRM toimintaa, sillä datan säilytys ja sen organisoiminen helpottuu merkittävästi tietotekniikkaa käyttäen. CRM-järjestelmän tarkoituksena on yhdistää kerätyt tiedot yhdeksi isoksi tietokannaksi, josta tarvittava tieto on helppo etsiä. Esimerkkinä tietokannan sisällöstä voi olla nykyiset tai mahdolliset uudet asiakkaat eli liidit (kuva 1). Asiakkuuksien hallinta helpottaa varsinkin johdon mahdollisuutta seurata ja pysyä ajan tasalla tuottavuudesta ja asiakaskontakteista (itewiki, 2022).

Kuva 1. Esimerkki yksinkertaisesta CRM mallista.



Yritykset ovat ryhtyneet suunnittelemaan ja implementoimaan erilaisia CRM toimintamalleja, joilla on saatu kerättyä tutkimustuloksia asiakkuudenhallinnan vaikutuksesta myynninedistämiseen. Esimerkiksi IBM (International Business Machines Corporation) kertoo CRM toimintamallien hyödyistä (kuva 2) kuten asiakassuhteiden kehittymisestä, lisääntyneistä myyntikiintiöistä, parannuksista tuottavuudessa ja vähennyksistä työvoimakustannuksissa. Vuonna 2021 maailmanlaajuisessa Finances Onlinen esittämässä tutkimuksessa CRM todettiin olevan arvokkain taloussuunnittelu- ja sijoitusneuvontayrityksille suunnattu ohjelmisto. (Kumar. s.16–17; Gilbert. 2022).

Kuva 2. CRM hyödyt pääoman tuotossa (Gilbert. 2022)

1.4 Tavoitteet

Ensimmäisessä palaverissa kävimme läpi tavoitteita ja sovimme tekeväni palvelinpuolen toimivaksi käyttäen mahdollisimman paljon Djangon sisäisiä kirjastoja ja pitäen kirjaa sovelluksen toteutuksesta. Kirjanpito tarkoitti kommenttikenttien kirjoittamista ja palavereissa esittäen, kuinka palvelinpuoli toimii. Pidimme työnohjaajan kanssa kommunikointia tärkeänä osana työtä ja pyrimme pitämään viestinnän mahdollisimman selkeänä ja säännöllisenä.

Asiakaspuolen ohjelmointi ei ollut tämän sovelluksen olennainen osa, vaan toimiva palvelinpuolen kokonaisuus, joten sovimme että teen ulkoasusta yksinkertaisen ja nopeasti muokattavan. Pieniä lisäominaisuuksia sain lisättyä applikaatioon käyttämällä, esim. kokoontaitettavaa tekstikenttää ja lataamalla tiedot Excel taulukkona. Käyttöliittymän värimaailmaksi sovimme tumman ja selkeän.

1.5 Rajaus

Projektista jouduimme rajaamaan pois tietoturvallisuuden syvällisemmän tutkimisen, käyttöliittymän valmiin ulkomuodon ja testaus ympäristön rakentamisen. Tietoturva tarkistuksiin käytin vain Djangon tarjoamia työkaluja, enkä käyttänyt aikaa niiden muokkaamiseen. Djangon tietoturva on todella laaja aihe ja pelkästään siitä olisi pystynyt kirjoittamaan opinnäytetyön. Automaattinen testaus on todella hyödyllinen bugien poistamisessa ja varmistamaan että tehdyt muutokset eivät ole vahingoittaneet ohjelmaa.

1.6 Työn kokonaisuus

Aloittaessani projektin kanssa perehdyin verkkosovelluskehityksen perusteisiin ja käyn tässä opinnäytetyössä läpi vaihe vaiheelta, kuinka työni eteni. Verkkosovelluskehitys on todella nopeasti kehittyvä ja laaja ohjelmistokehityksen ala. Työn selkeä rajaus on tärkeää, ettei aihe kasva liian laajaksi ja vie huomiota pois alkuperäisen työn tavoitteesta. Verkkosovelluskehitys on todella laaja aihe, määrittelyllä ja rajauksella työ pidetään keskittyneenä asetettuihin tavoitteisiin.

2 Django historia

Tutkittuani mahdollisia verkkosovelluskehyksiä, joita oli mahdollista hyödyntää projektini teossa, päädyin Djangoon. Olen käyttänyt Python ohjelmointikieltä ja se antoi minulle eniten pohjaa käyttää verkkokehyksenä Djangoa. Djangossa minua kiehtoi eniten sen selkeä dokumentointi, laaja käyttäjämäärä, CRM-järjestelmään sopivuus ja aloittelijaystävällisyys. Django on myös käytössä Tavastia Softwarella, joten ensimmäisen palaverin jälkeen aloitin projektin Djangolla.

Django kehitettiin 2003 Lawrence Journal-World Lawrencessa, Kansasissa. Djangon tarkoitus oli helpottaa ja nopeuttaa monimutkaisten tietokantapohjaisten verkkosivujen kehitystä. DRY (Don’t Reapeat Yourself) koodi rakenne vähentää työmäärää ja on keskeinen osa Djangon käyttöperiaatteita. Django on enimmäkseen käytössä sivustojen palvelinpuolen osassa, vaikka se soveltuu myös asiakaspuolen kehitykseen. Pythonin ja Djangon päätarkoituksena on pitää koodi helposti luettavana ja luotettavana. Nämä yhteiset ominaisuudet ja tavoitteet huomioon ottaen Python valittiin pohjaksi Djangolle. (Dinder s.10)

2.1 MVC ja MVT

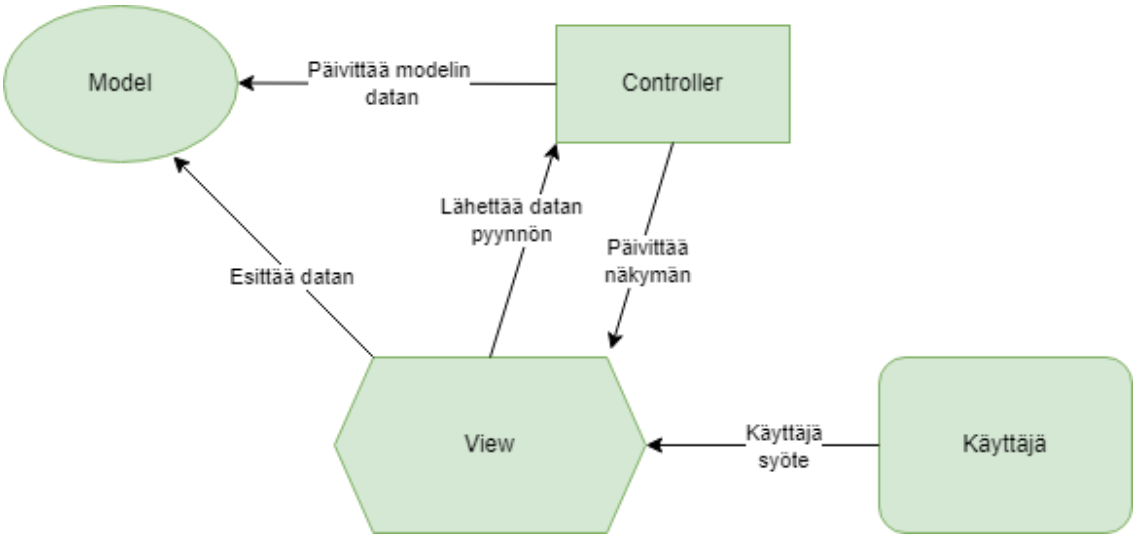
Model View Controller (MVC) on ohjelmiston suunnittelumalli, joka erottaa tietojen esittämisen erottamista komponenteista, jotka käsittelevät tietoa. Kuvassa 3 esitetään MVC sovelluksen dataliikenne. Esimerkki MVC ohjelmistokehyksestä: Ruby on Rails. MVC on jaettu eri komponentteihin:

Model: hallitsee dataa ja sovelluksen logiikkaa ja rajoituksia.

View: esittää käyttäjälle dataa ja tarjoaa siihen työkaluja.

Controller: toimii siltana Model ja View komponenttien välillä, ja renderöi näkymän.

Kuva 3. MVC data liikenne.



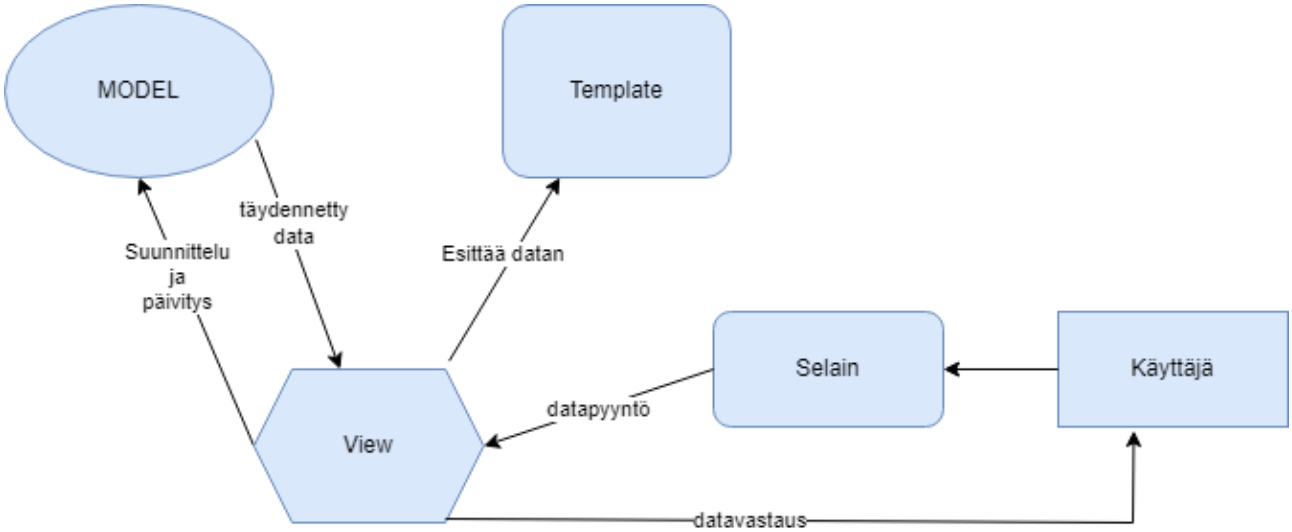
Model View Template (MVT) suunnittelumalli on samantyylinen kuin MVC. Toisin kuin MVC, se käyttää verkkorajapintoja ja Controller osion kehys hoitaa itse. Kuvassa 4 esitetään datan liikenne MVT mallissa. Esimerkki MVT ohjelmistokehyksestä: Django. (Shaji, 28.12.2019) MVT:ssä on myös kolme komponenttia:

Model: Tietojen käyttöliittymä, ja toimii koko sovelluksen loogisen rakenteen pohjana.

View: Renderöi näkymän ja hyväksyy HTTP pyynnön ja palauttaa vastauksen. Toiminta perustuu vuorovaikutukseen Modelin kanssa.

Template: HTML koodipohja, johon data esitetään joko staattisesti tai dynaamisesti.

Kuva 4. MVT datan liikenne.



2.2 Sovelluskehykset

Verkkokehykset ja verkkosovelluskehykset ovat suunniteltu helpottamaan ja nopeuttamaan sovellusten kehittämistä ja rakentamista, sillä niihin on rakennettu sisälle valmiiksi perustoimintoja ja ne tarjoavat selkeät rajapinnat. Kehykset vähentävät kustannuksia ja parantavat ohjelmistojen laatua. (Wikipedia, 2022)

2.2.1 Django

Django on nopea ja korkealaatuinen verkkosovelluskehys, joka on ilmainen ja avoimenlähdekoodin tuote. Suuri suosio ohjelmistokehittäjien keskuudessa takaa, että Django pysyy ajan tasalla ja verkkokehys päivittyy. Ohjelmointi kielenä Django käyttää pythonia, ja tukee mm. PostgreSQL; MariaDB; MySQL; Oracle; SQLite tietokantoja. Django on joustava ja monipuolinen verkkosivustojen rakentamisessa. Useimmiten Djangoa hyödynnetään kuitenkin data painotteisissa tuotteissa, esim. sisällönhallintajärjestelmissä ja wikeissä. Python ohjelmointikielenä sopii myös hyvin datan käsittelyyn. (MDN, 2022).

Djangon toiminta perustuu rajapintoihin ja niiden käyttöön. Rajapinnalla tarkoitetaan pistettä ohjelmassa missä käyttöliittymä ja palvelinpuolen kohtaavat. Rajapinnoissa data muutetaan tietokannasta helposti luettavaan muotoon verkkosivustolle. Django tarjoaa monipuolisia lisäominaisuuksia, joita käyttämällä pystyy välttymään suurelta määrältä työtä. Ohjelmassa on käytössä CSRF token (Cross-Site Request Forgery). Sovellus, joka käyttää CSRF tokenia tarkistaa kaikki HTTP pyynnöt sovelluksen sisällä, antaa niille arvot ja tarkistaa että pyynnöt ovat sallittuja. (synopsys, haettu 2022)

Djangon sisään on rakennettu Cross site scripting (XSS) suojaus. XSS suojaa sivustoja injektio hyökkäyksiltä, joissa hyökkääjä yrittää syöttää skriptejä muiden käyttäjien selaimiin tietokannan kautta. Hyökkäyksen uhri ajaa haitallisen skriptin linkin kautta, joka ajaa selaimen suorittamaan haittaohjelman. (Wikipedia, 2022)

Djangossa on kattavia kirjastoja, joiden avulla on helppo lisätä ominaisuuksia ilman, että joutuu tekemään kaiken alusta alkaen. Django.contrib.auth kirjastossa on käyttäjätunnuksien luomiseen ja siihen liittyvien toimintojen käyttöönottoon sisäänrakennettu pohja, tämä sisältää mm. rekisteröinnin, kirjautumisen ja salasanan

vaihdon. User objekti sisältää valmiiksi tehtyjä ensisijaisia attribuutteja. Esimerkiksi pää attribuutit peruskäyttäjällä ovat, eli default user ovat: username, password, email, first_name ja last_name. Kirjasto sisältää myös käyttäjän ryhmään liittyen modelin groups, jota hyödyntäen voi lisätä uuden käyttäjä automaattisesti ryhmään rekisteröitymisen yhteydessä. (Django software foundation, n.d.)

2.2.2 Ruby on Rails

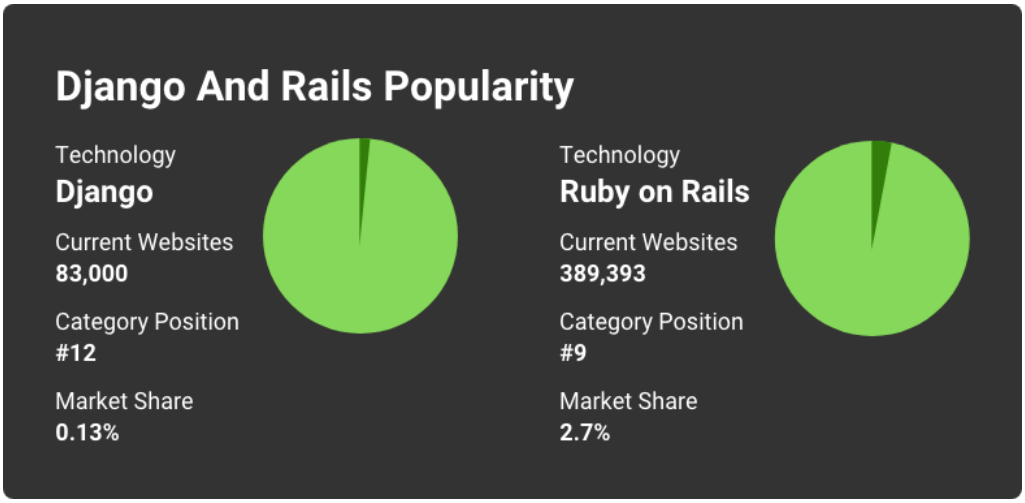
Verkkosovelluskehys, jonka ohjelmointikieli on Ruby. Ruby on Rails on MVC verkkokehys, joka mahdollistaa nopean verkkosovellusten ja sivustojen kehittämisen. Ruby on myös avoimenlähdekoodin tuote, joka käyttää oletuksena SQLite tietokantaa. (Zats, 2022)

2.2.3 Vertailu Django ja Ruby on Rails

Selkein ero Ruby on Rails ja Djangon välillä on ohjelmointikieli. Verkkokehysten nopeuksissa ei ole suuria eroja, mutta Ruby on 0.7 % Djangoa nopeampi. Djangoa hyödynnetään enemmän paljon dataa sisältävissä verkkosivuissa, mutta Ruby on suositumpi verkkokauppa tyyppisissä sivustoissa. Joustavuutensa ansiosta monet verkkokehitykseen keskittyneet sovelluskehittäjät suosivat Rubyä Djangon sijaan. Django on kuitenkin aloittelija ystävällisempi ja tällä hetkellä nousemassa suosituimmaksi käyttäjämäärissä. (Francis, 21.7.2022).

Ruby on Rails on kuitenkin käytössä laajemmin verrattuna Djangoon, koska Ruby on Rails on 10 vuotta vanhempi verkkokehys, ja projekteja on kertynyt enemmän vuosien aikana. StackOverflow:n tekemässä tutkimuksessa Djangoa käytti 14.2 % ja Ruby on Rails 7 % 42,279 vastanneista ohjelmistokehittäjistä. Kaikesta huolimatta Ruby on Rails säilyttää pienen etumatkan suosiossa Djangoon (kuva 5). (Zats, 2022).

Kuva 5. Djangon vs Ruby on Rails. (Zats, 2022)



2.3 React

React on avoimen lähdekoodin Javascript kirjasto, joka laajuudeltaan soveltuu myös verkkosovelluskehys kehitykseen. Suurin osa verkkoapplikaatioista on tänä päivänä toteutettu Reactilla. Esim. Instagram, netfilx, PayPal, ja Facebook. React soveltuu modernien single-page sovellusten ja verkkosivustojen kehittämiseen. (Wikipedia, 2022).

React on nopea ja suosittu SPA (Single Page Application) sovelluskehitysmalli. React soveltuu hyvin puhelinapplikaatioiden kehittämiseen, sen keveyden ja tehokkuuden ansiosta. Suosionsa ansiosta Reactilla on helppo aloittaa työskentely, koska ohjeita on saatavilla helposti. (Charles, 14.7).

Dynaamisuus on myös helppo toteuttaa Reactin avulla, vaikka HTML vaatisi monimutkaisempaa koodausta. React sisältää myös mahdollisuuden uudelleenkäyttää ominaisuuksia niin verkkosovelluksissa kuin puhelinsovelluksissa. Vianetsintään ja hallintaan löytyy kehittäjätyökaluja, jotka helpottavat vikojen löytämistä. (Java assignment help, 2021)

2.3.1 Vue

Vue on JavaScript verkkosovelluskehys asiakaspuolen kehitykseen. Vue on rakennettu HTML, CSS ja JavaScrptin päälle helpottamaan ja nopeuttamaan sovelluskehitystä. Verkkosovelluskehykseen on sisäänrakennettu suurin osa asiakaspuolen ohjelmointiin tarvittavia toimintoja. Vuen vahvuuksina pidetään sen joustavuutta, koodin selkeyttä, komponenttien uusintakäyttö mahdollisuutta, yksikkötestauksen ja laadunvarmistuksen yksinkertaistamista pienempien komponenttien ansiosta (Vue.js, 2022; altexsoft, 2022).

2.3.2 Vertailu React ja Vue

React on yksi maailman suosituimmista teknologioista verkkosovelluksien kehittämiseen. Suuri käyttäjämäärä takaa React kirjaston kehittymisen ja hyvien ohjeiden olemassaolon. React on kevyt kirjasto, joka parantaa ohjelman tehokkuutta ja optimointia. Sovelluksen tehokkuuden tärkeys kasvaa mitä laajemmaksi sovellus kehittyy. Optimointi tosin on hyvin sovelluskohtaista ja ohjelmiston laajetessa React saattaa hidastua.

Vue on aloittelijaystävällisempi, mutta se on vähemmän käytössä verrattuna Reactiin. Vue on Reactia nopeampi ja se pystyy käsittelemään kymmenen kuvaa sekunnissa toisin kuin React vain yhden kuvan sekunnissa. Vaikka Vue tarjoaa nopeamman käyttönopeuden ja tehokkuuden, Reactin käyttöympäristö on laajempi ja monipuolisempi. React sisältää enemmän ja monipuolisempia työkaluja, malleja ja toimintoja, joiden ansiosta suuret verkkoalustat suosivat Reactia. (Fireart, 2022)

2.3.3 JQuery

JQuery on Javascript kirjasto, jota käytetään web applikaatioiden asiakaspuolen ohjelmoinnin kehittämisessä. JQuery on kevyt kirjasto, jonka tarkoitus on tehdä enemmän vähemmällä kirjoituksella. Se on kuitenkin vanhempaa teknologiaa ja monesti se ei nopeuta tai lyhennä koodin määrää. JQuery on käytössä mm. Googlella, Netflixillä ja IBM:llä. Javascriptin kehitys on ollut todella nopeaa ja se on kyseenalaistanut JQueryn tarpeellisuuden. (javaTpoint, 2022).

Ohjelmistokehityksessä täytyy ottaa huomioon halutut tavoitteet ja ohjelmiston käyttötarkoitus. Django, Ruby on Rails, React, Vue ja JQuery ovat erilaisia työkaluja tarkoitettu erilaisiin verkkosivuihin ja sovelluksiin. Valitsemalla oikeat työkalut kehitystyö ja sen tuomat haasteet helpottuvat.

2.4 Yhden sivun sovellukset ja monen sivun sovellukset

Verkkosovellus on applikaatio, joka toimii internetselaimessa. Suurin osa applikaatiosta voidaan jakaa, joko Single Page Applikaatioihin (SPA), tai Multi Page Applikaatioihin (MPA). Verkkosovelluksissa tärkeintä on käyttäjän tarpeisiin vastaaminen, sivuston kokonaisvaltainen toimivuus, käyttäjäystävällisyys ja käyttäjämäärän ylläpitäminen ja kasvattaminen. Tavoitteisiin on mahdollista päästä monilla eri menetelmillä, SPA- ja MPA-sovellukset tarjoavat molemmat omat vahvuudet verkkosovelluksien kehittämiseen.

2.4.1 SPA (Single page application):

SPA tarkoittaa yhdelle sivulle rakennettua verkkosivustoa tai sovellusta. Dynaamisuus ja käyttäjien mielenkiinnon säilyttäminen ns. ”loputtomalla tietovirralla” vetää puoleensa varsinkin sosiaalisen median sovelluksia. Yhden sivuston ohjelmissa vahvuuksia ovat nopeampi käyttönopeus ensi lataamisen jälkeen ja sovelluksen tietojen tallentamisen välimuistiin. JavaScript järjestelmästä riippuvaisuus mahdollistaa ennakkoon ladattujen sivujen selaamisen offline-tilassa. SPA sivustot skaalautuvat hyvin ja niitä pidetään perinteisiä verkkosivuja tehokkaampana ja käyttäjäystävällisempänä. (Simicart 2022; smart SOFTWARE SOLUTIONS. INC.2020; Varaksina, 2022).

2.4.2 MPA (Multi page application):

MPA-sovellukset koostuvat monista eri sivustoista, jotka ovat yhteydessä toisiinsa linkkien avulla. MPA-mallilla tehdyillä sivustoilla on selkeä rakenne, joka helpottaa ohjeiden,

ohjelmisto arkkitehtuurin ja sivuston sisäisen logiikan kehittämisessä. MPA sovellukset toimivat myös ilman JavaScript tukea ja tietoturvan toteutus on SPA-sivustoa helpompaa. Monet vanhemmat sovellukset ja sivustot, joissa ulkonäöllä ei ole suurta merkitystä suosivat MPA-mallia. (Simicart, spa vs mpa; smart SOFTWARE SOLUTIONS. INC. 2020;

Useimmat verkkokaupat ovat rakennettu MPA-mallilla, joka takaa parhaat mahdollisuudet esiintyä Googlen hauissa, tämä lisää näkyvyyttä tuotteille. Googlen orgaaninen ja maksullinen haku tuo yhdessä noin 68 % verkkosivustojen liikenteestä, ja 34.7 % verkkokauppojen liikenteen tuotannosta. Tämä valtava etu SPA-sovelluksiin verrattuna on riittävä syy verkkokaupoille valita MPA-malli. Esimerkiksi Amazon on MPA-sovellus. (Simicart, spa vs mpa; Varaksina, 2022; Chaffey, 26.1.2022)

2.4.3 SPA ja MPA vertailu

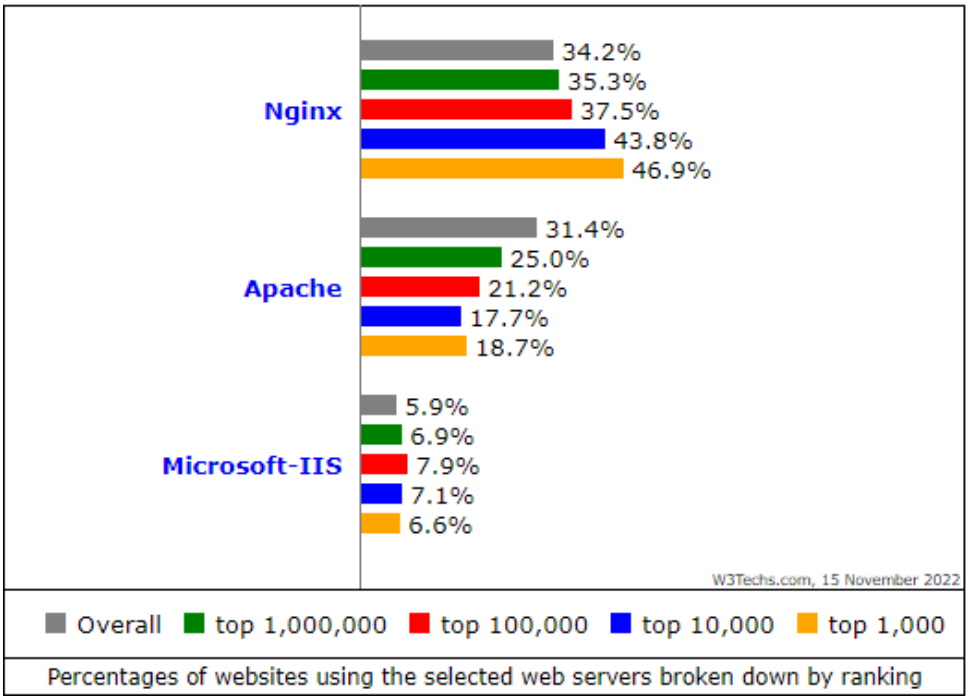
MPA ja SPA sivustojen vertailu on eri lähteiden perusteella ristiriitaista. SPA:ssa sivun avautuessa aikaa kuluu enemmän, koska data on kerätty vain yhdelle sivulle. Sovelluksen käynnistymisen jälkeen tehokkuus on MPA:ta nopeampi. MPA sovelluksissa yleinen käyttönopeus on hitaampi, mutta yksittäisten sivujen lataus nopeampi. Optimointi ja tehokkuus on kuitenkin hyvin sovellus- ja sivustokohtaista. SPA-sivustojen esiintyminen ja näkyvyys Googlen hakumoottorissa vaatii kuitenkin enemmän työtä MPA-mallilla toteutettuihin verrattuna.

SPA-sovelluksia pidetään käyttäjäystävällisempinä. Tällä hetkellä näyttää siltä, että suurin osa tulevaisuuden verkkokehityksestä tullaan tekemään SPA-mallilla. Twitter on tehty osittain SPA-mallilla ja hyödyntää sovelluksessaan SPA- ja MPA-mallia, joka tekee sovelluksesta ns. hybridi mallisen, tämä takaa Google hakumoottorin tarjoaman käyttäjäliikenteen hyödyntämisen myös SPA sovelluksessa. SPA ja MPA sovellukset molemmat skaalautuvat hyvin, mutta siitä kumpi paremmin on eriäviä mielipiteitä. Molemmilla verkkokehitys malleilla on omat hyvät ja huonot puolensa riippuen halutusta lopputuloksesta. (Tran, 2022; Churylov. 2022)

2.5 Verkkopalvelimen valinta

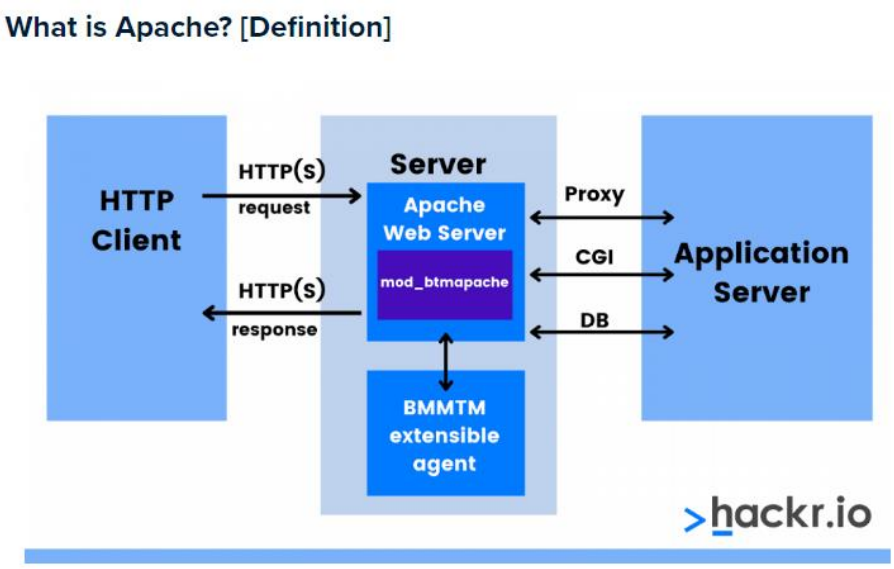
Tavastia Software Oy käyttää Apache alustaa ja en itse ollut vastuussa palvelimen valitsemisessa. Valitettavasti ajan puutteen takia en päässyt työskentelemään web-palvelimen kanssa, työssä vertailen vain kahta suosittua verkkopalvelinta. Taulukossa (kuva 6) näkyy suosittujen verkkopalvelimien tilastoja.

Kuva 6. Apache ja Nginx tilastoja (W3Techs, n.d.).



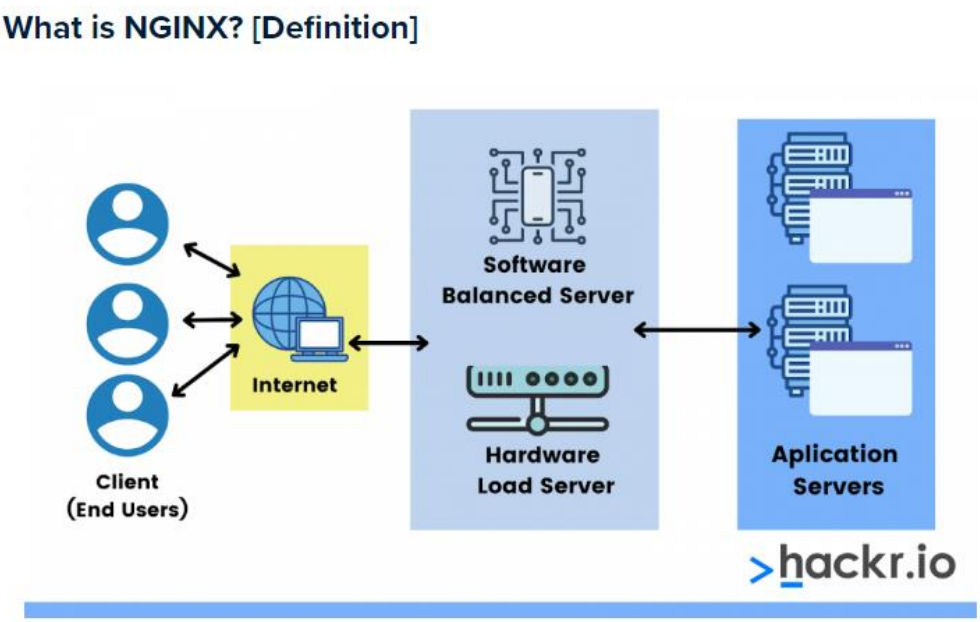
Apache: http serveri, joka on avoimen lähdekoodin verkkopalvelin ohjelmisto. Apache on suunniteltu tarjoamaan turvallinen http standardien mukainen verkkopalvelin (kuva 7). Apache on useasti käytössä pienemmillä verkkosivustoilla. Apache on ollut käytössä pidempään ja sille on kasvanut vahva käyttäjäyhteisö, jossa kehitetään uutta sisältöä ohjelmistolle.

Kuva 7. Apache periaatekaavio (Krishnan, 2022).



Nginx: Verkkopalvelin staattisten tiedostojen käsittelemiseen (kuva 8). Nginx on myös avoimen lähdekoodin verkkopalvelin, mutta ei pärjää Apachen laajoille toiminnoille ja koko ajan parantuvalle lähdekoodille. Tehokkuudessaan Nginx on moninkertainen Apacheen verrattuna, jonka takia sivustot joitten käyttäjämäärät ovat suuria suosivat Nginxiä. (Krishnan, 2022)

Kuva 8. Nginx periaatekaavio (Krishnan, 2022.).



3 Toteutus

Django käyttää oletus tietokantana SQLite. SQLite on Djangoon sisään asennettu, joten se oli yksinkertaisin ratkaisu sovelluksen aloittamisen kannalta. Django tukee myös muita tietokantoja. Kolme eniten käytettyä tietokantaa Djangon kanssa ovat SQLite, MySQL ja PostgreSQL. Yhteisö ja virallinen Django-dokumentaatio ilmoittaa PostgreSQL olevan Django Web Appsin ensisijainen tietokanta. (McCullum, 2021)

Projektin toteutusmallina käytin Scrum-menetelmää. Scrum tarkoittaa, että kehitystyö jaetaan useampaan osaan. Toimin ainoana kehittäjänä projektissa, joten minulla oli enemmän vapauksia toiminnan suhteen ja kehityssyklini oli lyhyempiä sen takia. Aloitin syklini toteutuksen suunnittelulla ja palaverilla työnohjaajani kanssa, jonka jälkeen siirryin sprinttiosioon, jossa toteutin haluamani muutokset ja lisäykset. (Wikipedia, 2022)

Työympäristönä toimi Visual Studio -sovellus. Tämä koodausympäristö on minulle tutuin alusta. Virtuaaliympäristönä projektille käytin python virtual environment (venv). Virtuaaliympäristö tarkoittaa eristettyjä asennushakemistoja, joiden avulla lokalisoidaan projektit ilman, että niitä pitää asentaa koko järjestelmän laajuisesti. (freeCodeCamp, 2022)

3.1 Sovelluksen toteutuksen vaiheet

Ensimmäiset kaksi viikkoa kului Djangon omaan oppimateriaaliin tutustumiseen ja alustan perustoimintojen harjoittelemiseen. Viikoittaiset palaverit olivat avain työn sujuvassa etenemisessä ja toimiva kommunikaatio työnohjaajan kanssa paransi huomattavasti projektin laatua. Syklin ensimmäisellä viikolla kävimme läpi työnantajani kanssa tavoitteet tuleville kahdelle viikolle ja suunnitelman halutuille toiminnoille. Palaverin jälkeen opiskelin tarvittavaa tietoa suunnitelman toteuttamiseen. Suurin osa ajasta kului Djangon dokumentaatioon tutustumiseen, pythonin harjoitteluun ja asioiden testaukseen. Syklin toisella viikolla aloin toteuttamaan toimintoja projektiin. Hyvän suunnitelman ja perehtymisen jälkeen pystyin nopeasti tekemään halutun toiminnon järjestelmään. Ongelma tilanteissa otin yhteyttä työnohjaajaan, ja pyrimme yhdessä etsimään ongelmaan ratkaisua.

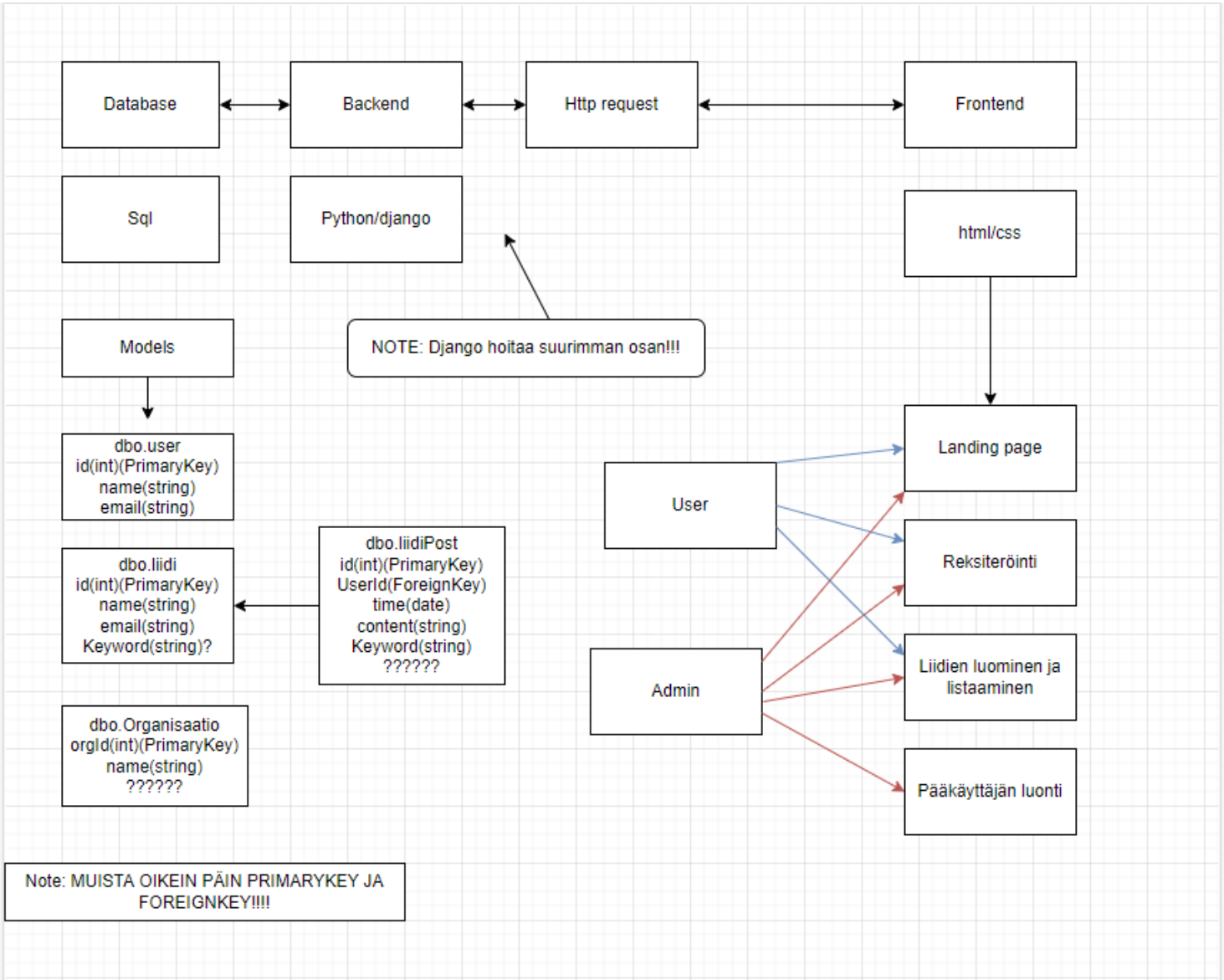
Nopeasti saavutetut tavoitteet olivat mahdollisia selkeiden Djangon ohjeiden ansiosta. Ylimääräisellä ajalla jatkoin ohjelman kehittämistä ja palasin aikaisemmin tehtyjen toimintojen pariin siistimään ja miettimään rakenteita uudelleen. Eniten aikaa työssä vei palvelinpuolen rakentaminen ja suunnittelu, jonka jälkeen tekeminen siirtyi näkymien luomiseen ja datan esittämiseen. Opiskeltuani kuinka Djangon views mallit toimivat, yksinkertaisten näkymien rakentaminen eteni nopeasti. Haastavimmat asiat olivat lomakkeiden (Forms) kanssa työskentely ja niiden oikeaoppinen täyttäminen.

Saatuani palvelinpuolen valmiiksi ja tiedon liikkumaan ohjelman sisällä haluamallani tavalla, siirryin rajapintojen kanssa työskentelyyn. Djangon dokumentaatiot tarjoavat hyvät ohjeet mallineisiin ja niiden käyttöön. Ulkoasu ei ollut prioriteetti, vaan että data oli haluamallani sivustolla oikein suodatettuna. Tietokannan datan tuominen rajapintojen kautta sivustoille esitettäväksi oikeassa muodossa aiheutti ongelmia. Hakiessani ohjeita ongelmien ratkaisuun Internetistä, tieto oli usein vanhentunutta. Tämä kuvastaa kuinka nopeasti verkkosovelluskehitys etenee ja kuinka tieto vanhenee nopeasti. Viimeiset viikot työn parissa käytin ulkoasun siistimiseen ja bootstrapin opiskeluun. Sovimme työnohjaajan kanssa, että tavoite ulkoasulle on yksinkertaisuus ja helppo muokattavuus.

3.2 Kaaviokuva sovelluksesta

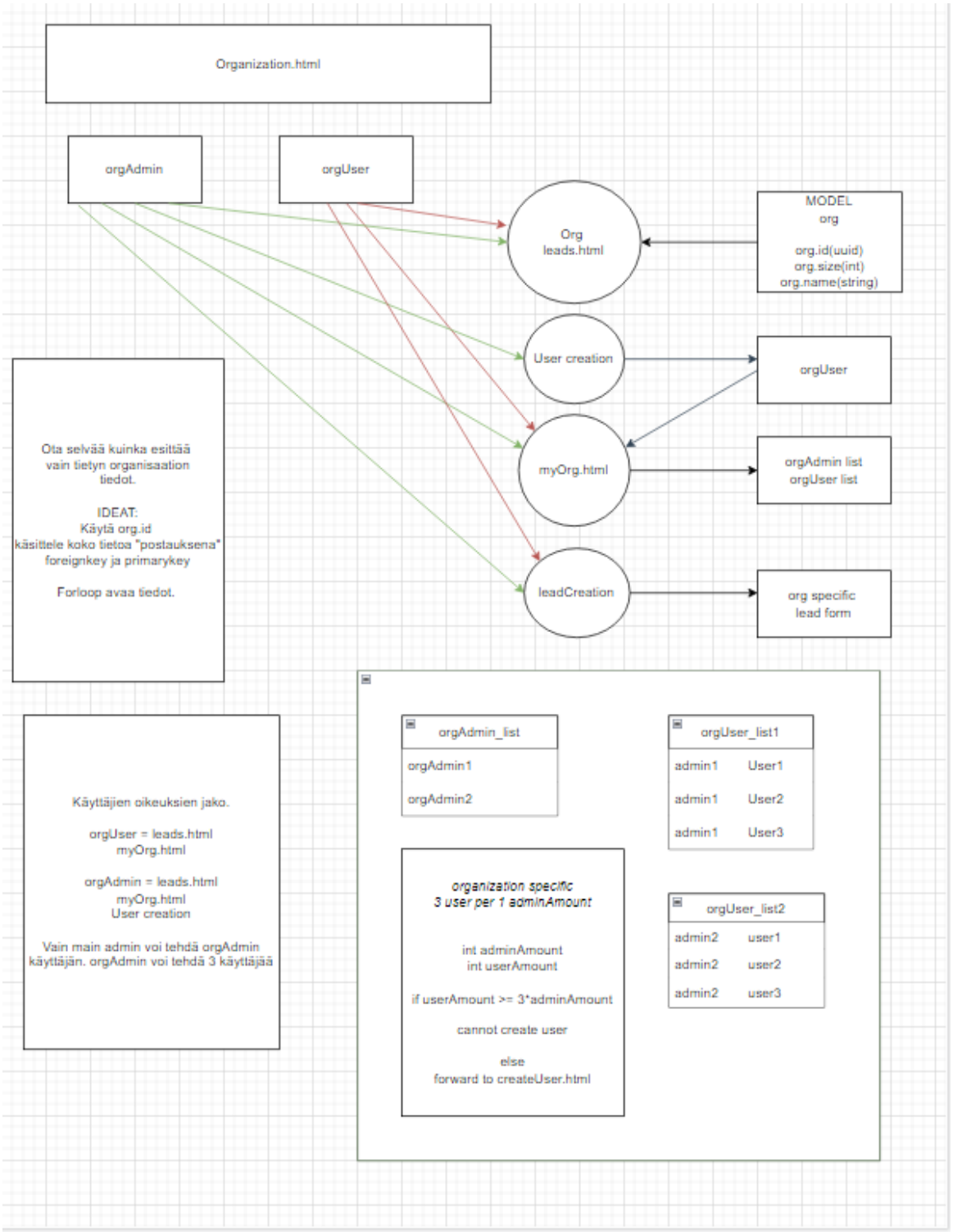
Alkuperäinen suunnitelma työstä ja sen tietorakenteesta esitetään kaaviomuodossa (kuva 9). Siitä näkyy kuinka Django helpottaa ja nopeuttaa tiedon liikkumista palvelinpuolen ja käyttöliittymän välillä. Kuvassa selviää myös, kuinka primary key ja foreign key eroavat toisistaan. Esim. Primary key:n tarkoitus on olla uniikki ja varmistaa että data tietyssä sarakkeessa on yksilöllistä. Foreign key on sarake tai ryhmä sarakkeita, jotka yhdistävät dataa taulukoiden välillä. (geeksforgeeks, 2022)

Kuva 9. CRM-järjestelmän kaaviokuva.



Organisaation kaaviosuunnitelma (kuva 10) sisäisistä suhteista ja kuinka käyttäjäoikeudet voisivat toimia ja rajoittaa käyttäjämäärää. Tämä suunnitelma jäi osittain vajaaksi ja jatkokehityksessä syventyisin organisoimaan ohjelman kuvan mukaisesti.

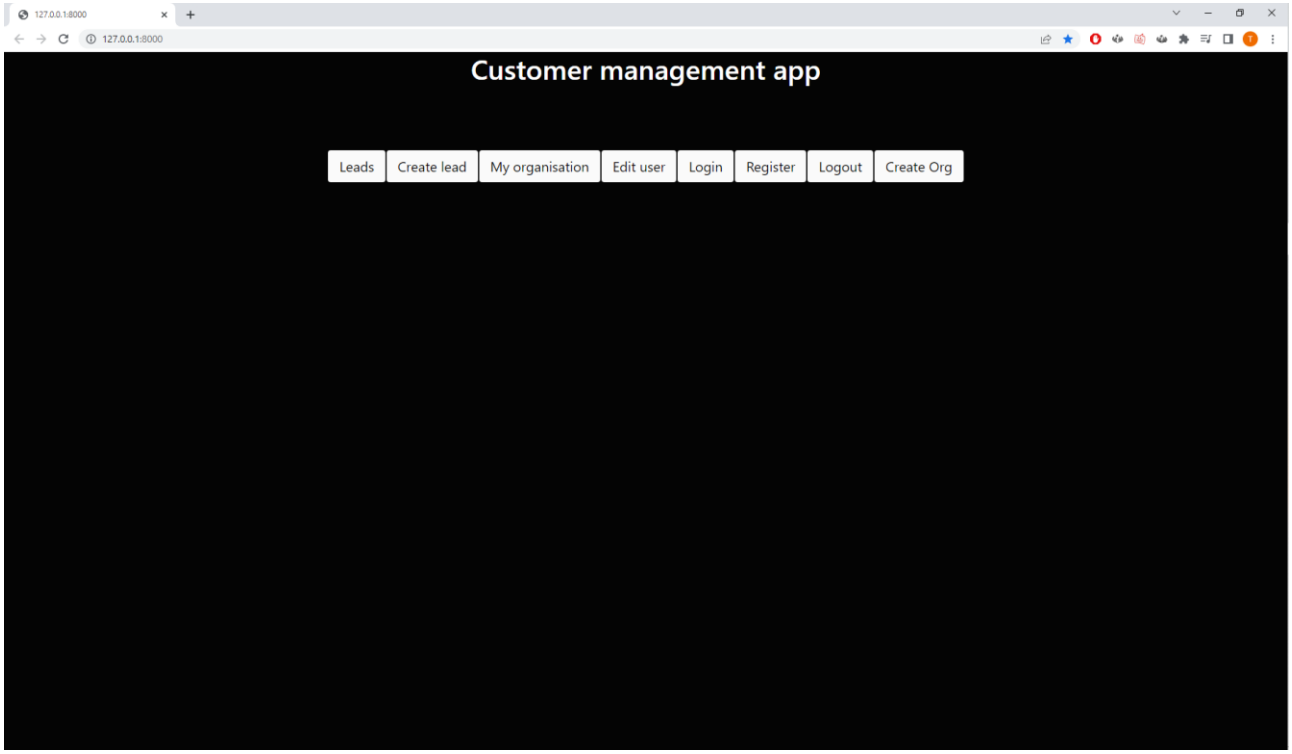
Kuva 10. Organisaation kaaviosuunnitelma.



3.3 Applikaation esittely

Etusivun (kuva 11) tarkoitus on antaa käyttäjälle linkit haluamille jatkosivuille ja toiminnoille.

Kuva 11. Etusivu



Leads/models.py tiedostossa alustan tarvitsemani modelit. Modelin osat valittiin yrityksen tarpeitten perusteella.

```
from pyexpat import model
from django.db import models
import uuid
from django.contrib.auth.models import User, Group


class Lead(models.Model):

    group = models.ForeignKey(Group, null=True, on_delete=models.CASCADE)
    user = models.ForeignKey(User, null=True, on_delete=models.CASCADE)
    lead_id = models.AutoField(
        primary_key = True,
        editable = False)
    lead_name = models.CharField(max_length=200)
    lead_email = models.CharField(max_length=200)
```

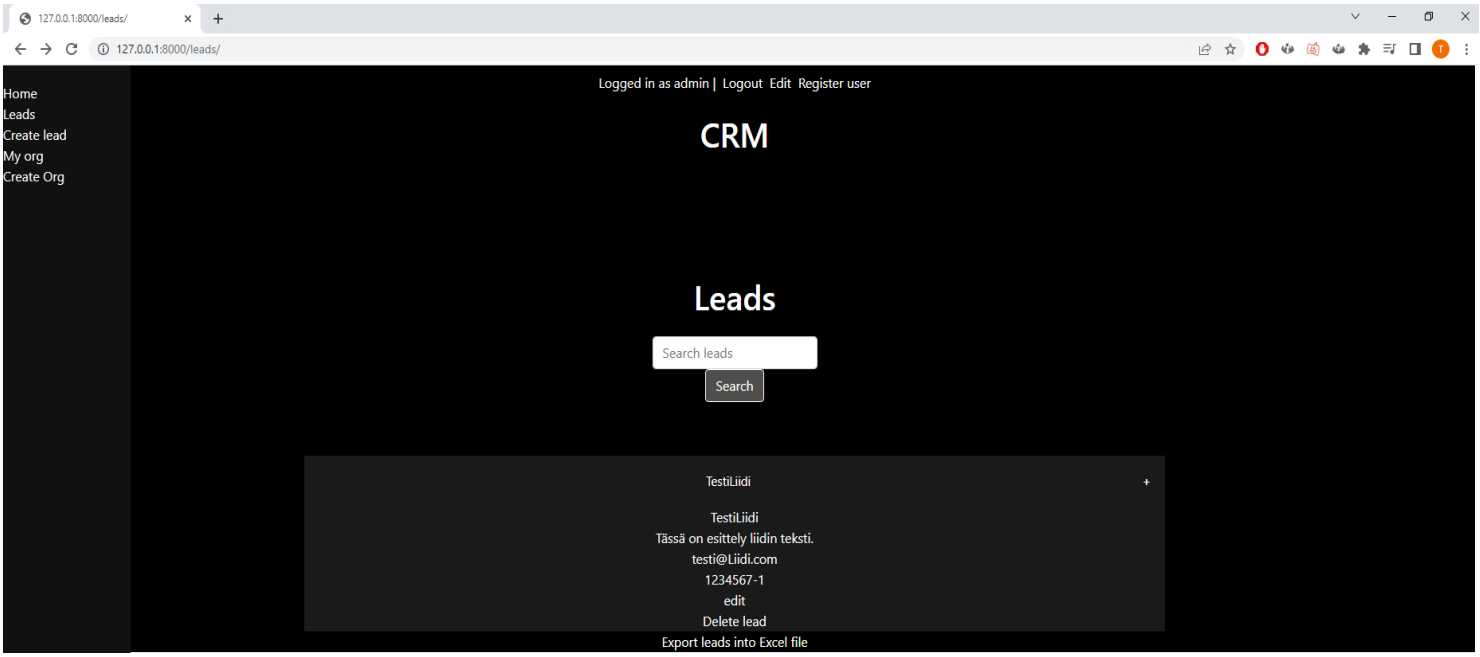
```
    pub_date = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    lead_text = models.CharField(max_length=200)
    lead_pNumber = models.CharField(max_length=200)
    y_tunnus = models.CharField(max_length=200)

    def __str__(self):
        return (self.lead_name)
```

leads/forms.py luodaan lomake käyttäen model.py tietoja. Lomakkeeseen lisätään modelit, jotka halutaan olevan käyttäjän täyttämiä.

```
from django.forms import ModelForm
from django import forms
from .models import Lead


class LeadList(ModelForm):

    class Meta:
        model = Lead
        fields = ('lead_name', 'lead_email', 'lead_text', 'lead_pNumber',
"y_tunnus")

        widgets = {
            'lead_name': forms.TextInput(attrs={'class':'form-control'}),
            'lead_email': forms.TextInput(attrs={'class':'form-control'}),
            'lead_text': forms.Textarea(attrs={'class':'form-select'}),
            'lead_pNumber': forms.TextInput(attrs={'class':'form-select'}),
            'y_tunnus': forms.TextInput(attrs={'class':'form-select'}),

        }
```

Seuraavaksi luon näkymän views.py tiedostoon. Views ottaa verkkopyyntöjä ja palauttaa verkkovastauksia. Tänne kirjoitetaan funktiot, jotka määrittävät mitä dataa jokaiselle sivulle lähetetään. Liidien näkymä verkkosivulla (kuva 12) esittää halutut tiedot sovelluksen käyttäjälle.

Kuva 12. Leads.html näkymä



Esimerkki LeadView funktiosta views.py tiedostossa:

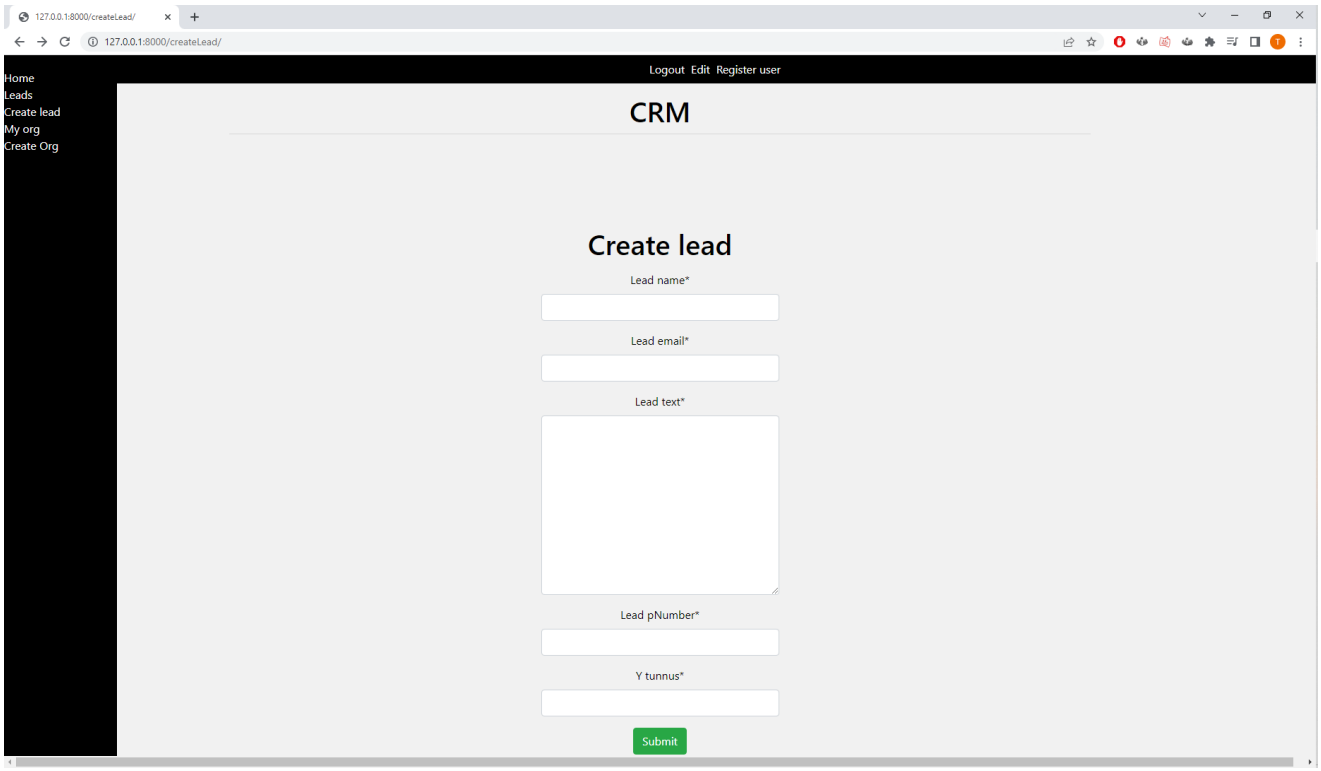
```
class LeadView(LoginRequiredMixin, generic.ListView):

    template_name = 'leads/leads.html'
    context_object_name = 'lead_list'           # nimi millä haetaan
    login_url = '/login/'
    redirect_field_name = '/'


    def get_queryset(self):
        l = self.request.user.groups.values_list('id',flat = True) #kirjautuneen
käyttäjän ryhmät l muuttujaan
        l_as_list = list(l)    #muuttujasta tehdään lista
        groupId = l_as_list[0]  #otetaan listan ensimmäinen jäsen muuttujaan

        return(Lead.objects.filter(group_id=groupId)) #filtteröi palautuksen ryhmän
id perusteella
```

Esimerkki liidien luonti funktiosta views.py tiedostossa:

```
@login_required
def CreateLead(request):
        l = request.user.groups.all()[0] #kirjautuneen käyttäjän ryhmät l
muuttujaan

        current_user = request.user
        current_group = l

        if request.method == "POST":
            form = LeadList(request.POST)

            if form.is_valid():
                form.instance.user = current_user   #liitetään form instanceen
tämän hetkinen käyttäjä
                form.instance.group = current_group #liitetään form instanceen
tämän hetkinen ryhmä
                form.save()

                messages.success(request, "created lead successfully." )

            return redirect("/createLead")
        else:
            form = LeadList()

        return render(request, "leads/createLead.html", {"form":form})
```

CreateLead sivustolla (kuva 13) on tekstikenttä, jonne kirjataan liidien tiedot.

Kuva 13. Liidien luonti näkymä



Seuraavaksi multi page applikaatiossa pitää yhdistää näkymä mallinneihin. Se tapahtuu url.py tiedostossa. Esimerkki urls.py tiedostosta:

```
from django.urls import path

from . import views
from leads import views as v
from django.urls import re_path as url
from django.contrib.staticfiles.urls import staticfiles_urlpatterns


app_name = 'leads'


urlpatterns = [

url(r'^$', views.index, name='index'),
path('leads/', views.LeadView.as_view(), name='Leads'),
path('createLead/', v.CreateLead, name='createLead'),
path('<int:pk>/edit/', v.edit_lead, name ='edit_lead' ),
path('deleteLead/<lead_id>', v.lead_delete, name ='lead_delete' ),
```

```
path('search/', v.search_leads, name = 'search'),
path('export-to-csv/', v.export_to_csv, name='export-to-csv'),

]
urlpatterns += staticfiles_urlpatterns()
```

Lopuksi luodaan mallinne kansioon leads.html sivusto, jossa tuodaan näkymä verkkosivulle ja tehdään ulkoasu määrittelyt.

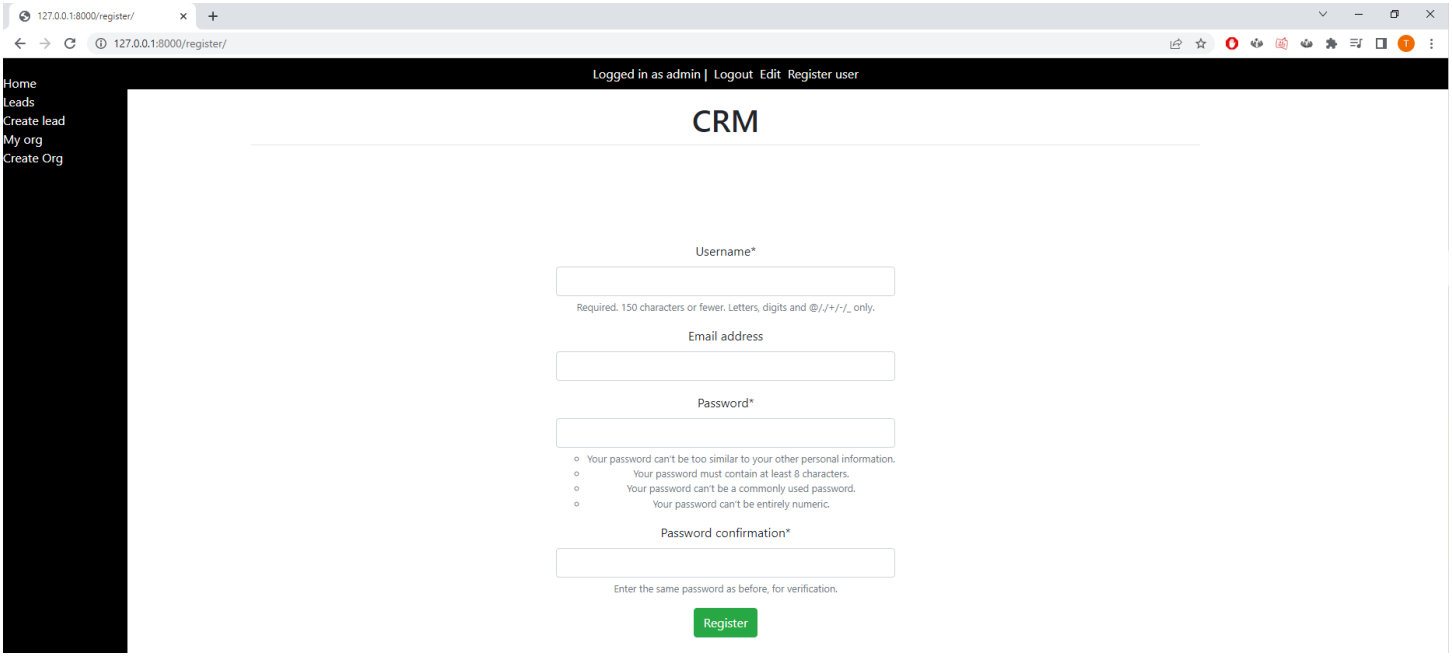
```
{% extends 'leads/base.html' %}{% block content %}
{% load static i18n %}
{% load static %}
{% get_current_language as language_code %}
<link rel="stylesheet" type="text/css" href="{% static '/css/style.css' %}" />

<head>
  <div class="title">
    <h1>Leads</h1>
  </div>
<div class="searchBar">
      <form action="{% url 'leads:search' %}" method="POST">
        {% csrf_token %}
        <input class="form-control me-2" type="search" placeholder="Search
leads" aria-label="Search" name="searched">
        <button class="btn btn-outline-secondary" type="submit">Search</button>
    </form>
  </div>
</div>
</head>


<body>
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script
>


      <div class="container">

        {% if lead_list %}

          {% for lead in lead_list %}
          <div class ="wrapper">
          <button type="button" class="collapsible"
onclick="openInfo(event)">{{ lead.lead_name }}</button>

          <div class="infoList">
                  <li>{{ lead.lead_name }}</li>
                  <li maxlength="10" size="10">{{ lead.lead_text }}</li>
                  <li>{{ lead.lead_email }}</li>
```

```
                <li>{{ lead.y_tunnus  }}</li>
                <li><a href="{% url 'leads:edit_lead' lead.pk
%}">edit</a></li>
                <li><a href="{% url 'leads:lead_delete'
lead.pk%}"  onclick="confirmation(event)" method="POST">Delete
lead</a></li>

          </div>
        </div>
          {% endfor %}

          {% else %}
            <p>No leads available.</p>
          {% endif %}


        </div>

<div class = "export_csv">
  <a href="{% url 'leads:export-to-csv'  %}"> Export leads into Excel file </a>


</div>
<script>
//[0] = button child element
//[1] = infoList child element
//Muista käyttää console.log("tänne mitä haluat tietää")!!!!!.
//Siirrä omaan kansioon

function openInfo(mouseClick) //leads ja näyttää collapsible tiedot

{
  var infoListElement = mouseClick.target.parentElement.children[1];
      if (infoListElement.style.maxHeight)
      {
        infoListElement.style.maxHeight = null;
      }
      else
      {
        infoListElement.style.maxHeight = infoListElement.scrollHeight + "px";
      }

}
</script>
<script>
  function confirmation(e) {

    let text;
    if(!confirm('Are you sure you want to delete this?')){
          //prevent sending the request when user clicked 'Cancel'
          e.preventDefault();
```

```
      }

    else {
      text = "You canceled!";
    }
    document.getElementById("demo").innerHTML = text;
  }
  </script>

</body>
</html>
{% endblock content %}
```

Rekisteröinti tapahtuu sille luodussa sivustossa (Kuva 14). Tekstikenttiin täytetään uuden käyttäjän tiedot ja sivusto tarkistaa tekstikenttien tietojen sopivuuden, ja salasanan riittävän vahvuuden.

Kuva 14. Uuden käyttäjän rekisteröinti.

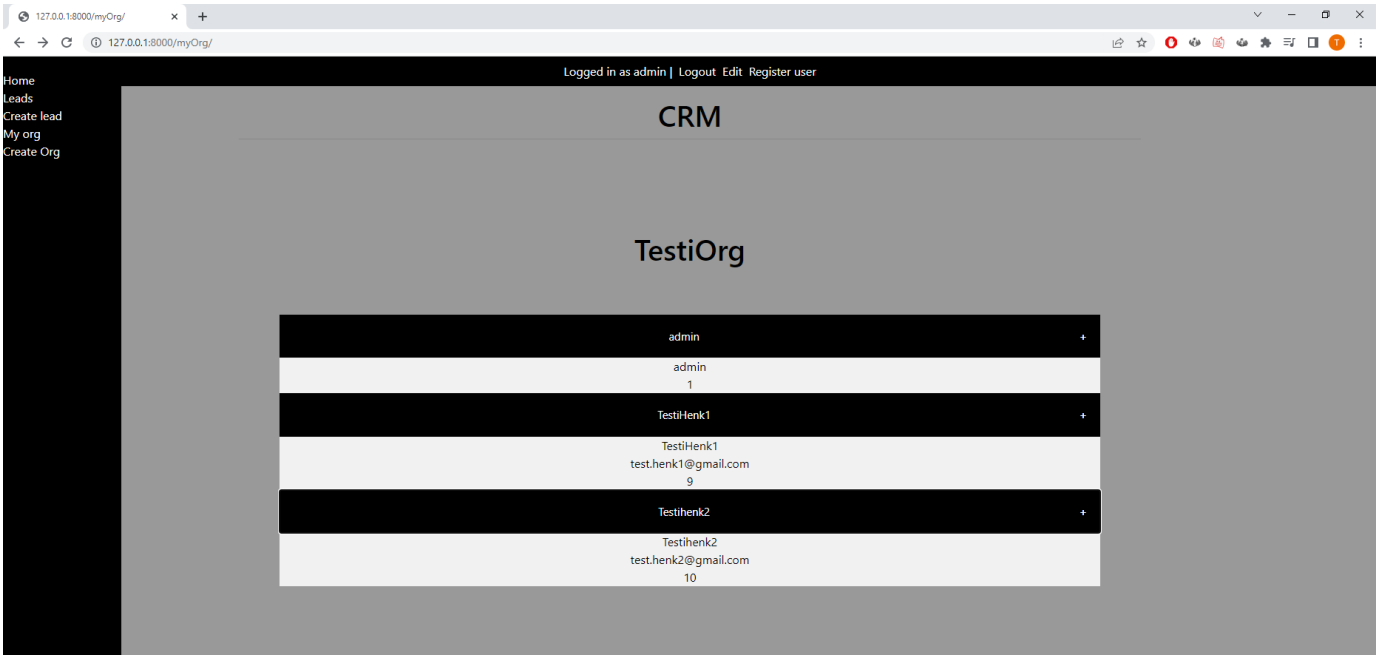


Rekisteröinti funktiossa käyttäjä liitetään rekisteröivän käyttäjän ryhmään automaattisesti.

```
@login_required
def register(request):

        l = request.user.groups.values_list('id',flat = True) #kirjautuneen
käyttäjän ryhmät l muuttujaan
        l_as_list = list(l)    #muuttujasta tehdään lista
        groupId = l_as_list[0]  #otetaan listan ensimmäinen jäsen muutujaan

        #rekisteröinti functio. palauttaa etusivulle rekisteröinnin jälkeen.
        if request.method == "POST":

            registerForm = RegisterForm(request.POST)   #haetaan RegisterForm

            if registerForm.is_valid():

                registerForm.save() #käyttäjä luodaan
                messages.success(request, "Registration successful." )


                user_name =
                User.objects.get(username=registerForm['username'].value())
                #etsitään käyttäjän nimi
                user_name.groups.add(groupId)   #lisätään käyttäjälle ryhmä
                käyttäen ID:tä (int value)

                return redirect("/")


            messages.error(request, "Unsuccessful registration. Invalid
            information.")

        else:
            registerForm = RegisterForm()
        return render(request, "registration/register.html", {"form":registerForm})
```

Käyttäjät näkevät oman ryhmän jäsenet ja heidän tietonsa. (kuva 15)

Kuva 15. Organisaation käyttäjien näkymä.




```
@login_required
def showOrg(request):

    form = RegisterForm(request.POST) #käytettävä form on RegisterForm.

    l = request.user.groups.all()[0] #kirjautuneen käyttäjän ryhmät l muuttujaan

    current_group = l
    form.instance.group = current_group      #Register formista haetaan ryhmä
    groupFiltered = User.objects.filter(groups=current_group) #Filtteröidään
näkyvät ryhmät vain käyttäjän omaksi

    return
render(request,'registration/myOrg.html',{'groupFiltered':groupFiltered})#palauteta
an myOrg.html sivulle filtteröity ryhmä näkymä
```

4 Johtopäätökset

Kokonaisuudessaan olen tyytyväinen projektiini ja valitsemaani aiheeseen. Ymmärrykseni alaan vahvistui ja tavoitteeni taitojeni kehittämisestä toteutui. Koin työympäristöni olevan ammattimaista, vaikka mahdollisuutta toimistotyöskentelyyn ei ollut. Rajaus työssäni onnistui hyvin eikä aihe levinnyt liian suureksi. Verkkosovelluskehitykseen löytyy paljon hyödyllisiä työkaluja ja niitä käyttämällä sovelluskehitys helpottuu huomattavasti. Työskentely on helppo aloittaa uusilla alustoilla niiden kattavien ohjeistuksien ja aktiivisten yhteisöjen ansiosta.

4.1 Tavoitteet ja lopputulos

Suunnitelman alkuperäisiin tavoitteisiin päästiin ja tuloksena pohja asiakkuuksienhallintajärjestelmälle. Aika riitti muutaman ylimääräisen toiminnon lisäämiseen, kuten Leads sivuston lataamisen Excel taulukkona omalle tietokoneelle ja JavaScript skriptiin, joka avaa liidin tiedon samalle sivulle käyttäen kokoontaitettavaa tekstikenttä toimintoa. Työnohjaaja ja minä olimme molemmat tyytyväisiä lopputulokseen, ja loppupalaverissa kävimme läpi mitkä asiat projektissa olivat onnistuneet ja mitkä olisivat voineet mennä paremmin.

4.2 Onnistumiset ja puutteet

Olen tyytyväinen lopputulokseen, käyttöliittymä voisi olla siistimpi ja bootstrappiä olisi voinut hyödyntää paremmin. Ulkoasuun ja sen osien nimeämiseen olisi voinut käyttää selkeämpää tapaa, joka tulevaisuudessa helpottaisi muutoksien tekemistä. Testaus ympäristö jäi puutteelliseksi ja se on todella tärkeä ja hyödyllinen toiminto kaikissa ohjelmointi projekteissa. Mikäli aloittaisin uudestaan työn tekemisen, käyttäisin aikaa sen luomiseksi ja ymmärtämiseksi. Uskon että pidemmällä aikavälillä testaus olisi maksanut itsensä takaisin menetetyssä ajassa. Githubin käyttö olisi voinut olla siistimpää ja olisin voinut käyttää hyödyksi kaikkia sen toimintoja ja tehdä muutoksien kuvauksista selkeämpiä. Toimiessa yksin ohjelmistonkehittäjänä kuvittelee muistavansa kaikki toiminnot ja muutokset ilman tarkkoja selityksiä. Github on hyödyllinen työkalu, jota en käyttänyt kaikella sen potentiaalilla.

4.3 Työn jatkokehitys

Seuraava askel työn kehittämisessä on tietoturvallisuuden testaus ja sen parantaminen. Ulkoasun viimeistely ja mielestäni sovellukseen voisi lisätä tiimi ryhmä toiminnon, jossa liidin voisi liittää tiettyyn tiimiin, joka koostuu organisaation sisäisistä jäsenistä. Ohjelmaan ei ole rakennettu testauksia. Mikäli jatkokehitystä tapahtuu, testaukset olisivat tärkeitä varsinkin koodin määrän kasvaessa.

Tällä hetkellä käyttäjä tasoja on vain kaksi, admin (superuser) ja käyttäjä. Jatkokehityksessä voisi tehdä kolmannen tason, joka toimii näiden välissä, orgAdmin. Uudella käyttäjätasolla olisi oikeudet päästä luomaan organisaatioita ja tekemään niihin käyttäjiä. Tämä mahdollistaisi yhden henkilön hallitsemaan useamman yrityksen asiakkuuksia. Järjestelmän kaupallistamisen kannalta jokaiselle yritykselle olisi hyvä asettaa käyttäjäraja. Ylläpitäjä asettaisi yrittäjälle käyttäjämäärän perustuen maksuun Esim. jokaista orgAdmin käyttäjää kohtaan mahdollista luoda 3 perustason käyttäjää.

4.4 Mitä opin

Opin kuinka Django toimii ja opin ymmärtämään sovelluskehityksen perusteita. Projektien suunnittelu ja tavoitteiden neuvotteleminen työnohjaajan kanssa tietyn aikarajan puitteissa oli myös minulle uusi kokemus. Etätyöskentelystä opin kuinka jakaa oma aika järkevästi ja saada töitä tehtyä kotoa käsin. Huonona puolena etätyöskentelyssä oli vähäinen vuorovaikutus kollegoitteni kanssa, ja ryhmä työskentelyn puute. Itsenäisyys oli iso osa tätä projektia ja vastuu oli täysin työn etenemisestä itselläni. Projektin alkuvaiheessa suurin osa ajasta kului ohjeiden lukemisessa ja Djangon omaan oppimateriaalin perehtymisessä. Tämä vahvisti perusosaamista työympäristöstä ja omaa itsevarmuutta Djangon kanssa työskentelystä.

4.5 Opinnäytetyön ohjaus

Suoritin opinnäytetyön teknisen osion kesän aikana ja aiheeni oli hyvin rajattu. Tavastia software Oy:n ohjaaja toimi ammattimaisesti ja tarjosi hyvää ohjausta teknisessä osiossa. Sähköposteihini vastattiin ja sain yhteyden ohjaajiini, niin koulun kuin Tavastia softwaren puolesta. Minua ohjattiin oikeisiin oppaisiin ja neuvottiin kuinka lähteä tutkimaan tietoa ja ratkomaan ongelmia.

Opinnäytetyön ohjaaja tarjosi hyvän alun opinnäytetyön kirjoittamisen aloittamiseen ja tarjosi neuvoja tiedonhakuun ja dokumentaation rakenteeseen. Opinnäytetyön ohjauspalaverissa sain kommentteja työn hyvistä puolista ja sen puutteista. Dokumentaatio ohjeisiin minun olisi pitänyt perehtyä enemmän ja esimerkiksi lähteitten kirjaamiseen olisi ollut hyvä saada enemmän henkilökohtaista opastusta, vaikka ohjeet olivat saatavilla HAMKin verkkosivuilla. Kokonaisuudessaan olen tyytyväinen opinnäytetyön ohjaukseen ja saamaani tukeen työn eri vaiheissa.

Lähteet

Altexsoft (23.11.2022) *The Good and the Bad of Vue.js Framework Programming*. https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/

Chaffey. D. (26.1.2022*) Search engine marketing statistics 2022* https://www.smartinsights.com/search-engine-marketing/search-engine-statistics/

Charle. L. (14.7.) *Django Vs React JS: What is the difference and which is the best?* https://medium.com/ palvelinpuoleners-club/django-vs-react-js-what-is-the-difference-and-which-is-the-best-5d9f7369a885

Churylov. (n.d.). *SPA vs MPA: The definitive guide for decision makers*. Haettu 20.11.2022 https://www.mindk.com/blog/single-page-applications-the-definitive-guide/

Cross-site scripting (10.11.2022) Wikipedia-artikkeli. https://fi.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=20998953

Dinder. M (27.6.2022) *Becoming and Enterprise Django Developer*

Django Software Foundation (n.d.) *Using the Django authentication system version. 4.1.* https://docs.djangoproject.com/en/4.1/topics/auth/default/

FinancesOnline (27.11.2022). *ROI benefits experienced by business using CRM systems.* [kuva] Haettu osoitteesta https://financesonline.com/crm-software-statistics/

Fireart. (7.2022). *Vue vs. React in 2022.* https://fireart.studio/blog/vue-vs-react-in-2022/

Francis. P. (21.7.2022) *Django vs Ruby on Rails Comparison: Which framework is better in 2022?* https://uvik.net/blog/django-vs-ruby-on-rails/

freeCodeCamp, (n.d.) *Python Virtual Environment Explained with Examples, 1.2.2022.* https://www.freecodecamp.org/news/python-virtual-environments-explained-with-examples/

GeeksforGeeks (n.d.*) The Model—View—Controller(MVC) Pattern [Kuva]. Difference Between MVC, MVP and MVVM Architecture Pattern in Android*. Haettu 20.11.2022 osoitteesta https://www.geeksforgeeks.org/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android/

Gilbert. N. (5.11.2022) *75 Basic CRM Software Statistics: 2022 Data Analysis & Market Share.* https://financesonline.com/crm-software-statistics/

Itewiki, (n.d.) *asiakkuudenhallinta CRM.* Haettu 15.11.2022 https://www.itewiki.fi/opas/asiakkuudenhallinta-crm

Java assignment help (22.12.2021) *Django Vs React: Which One Is Best for Web Development*? https://www.javaassignmenthelp.com/blog/django-vs-react/

javaTpoint (n.d.) *what is jQuery*. Haettu 15.10.2022 https://www.javatpoint.com/what-is-jquery

Krishnan. A. (19.9.2022) *What is NGINX? [kuva]. NGINX vs Apache: Head to Head Comparison*. Haettu osoitteesta https://hackr.io/blog/nginx-vs-apache

Krishnan. A. (19.9.2022). *What is Apache? [kuva]. NGINX vs Apache: Head to Head Comparison*. Haettu osoitteesta https://hackr.io/blog/nginx-vs-apache

Krishnan. A. (19.9.2022). *NGINX vs Apache: Head to Head Comparison*. https://hackr.io/blog/nginx-vs-apache

Kumar. V. *(26.7.2012) Statistical Methods in Customer Relationship Management*

McCullum. N. (1.1.2021) *The Best Database for Django Web Apps* https://www.nickmccullum.com/best-database-django-web-apps/

MDN, (9.9.2022) *Django introduction* https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

Ohjelmistokehys (23.10.2022) Wikipedia-artikkeli. https://fi.wikipedia.org/w/index.php?title=Ohjelmistokehys&oldid=20923959

Pedamkar. P. (n.d.*) Django Architecture [Kuva]. Django Architecture*. Haettu 20.11.2022 osoitteesta https://www.educba.com/django-architecture/

React JavaScript library (17.11.2022) Wikipedia-artikkeli. https://en.wikipedia.org/w/index.php?title=React_(JavaScript_library)&oldid=1122468159

Scrum (27.10.2022). Wikipedia-artikkeli https://fi.wikipedia.org/w/index.php?title=Scrum&oldid=20945696

Shaji. M. (28.12.2019*) Difference between MVC and MVT architecture*. https://www.geekinsta.com/difference-between-mvc-and-mvt/

Smart SOFTWARE SOLUTIONS. INC. (13.2.2020) *Single Page Applications SWOT Analysis* https://www.smartsoftwareinc.com/articles/spa-swot

synopsys (n.d.) *How does Cross-site request forgery work?* Haettu 17.11.2022. https://www.synopsys.com/glossary/what-is-csrf.html

Tran. T. (19.9.2022*) SPA VS MPA: Pros, Cons & How To Make Final Choice.* https://www.simicart.com/blog/spa-vs-mpa/

Varaksina. S. (10.10.2022) *Single-Page Applications vs Multi-Page Applications: The Battle of the Web Apps* https://themindstudios.com/blog/spa-vs-mpa/

Vue.js (n.d.) *What is Vue?* Haettu 23.11.2022 https://vuejs.org/guide/introduction.html

W3Techs (n.d.) *Usage broken down by ranking [Kuva]. Comparison of the usage statistics of Nginx vs. Apache vs. Microsoft-IIS for websites*. Haettu 17.11.2022 osoitteesta https://w3techs.com/technologies/comparison/ws-apache,ws-microsoftiis,ws-nginx

Zats. I. (1.2022) *Django and Rails Popularity [Kuva]. Rails vs Django comparison*. Haettu osoitteesta https://keyua.org/blog/rails-vs-django-comparison/

Zats. I. (1.2022). *Rails vs Django comparison*. https://keyua.org/blog/rails-vs-django-comparison