Projektin loppuraportti

Vuoto

versio 1.0

Versiohistoria

|  |  |  |  |
| --- | --- | --- | --- |
| Versio | Päiväys | Tekijät | Selite (Alkuperäinen, muutokset, korjaukset...) |
| 1.0 | 21.5 | R19 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sisällysluettelo

1. JOHDANTO 4

2. PROJEKTIN ORGANISOINTI 5

2.1 Projektin kuvaus ja tavoite 5

2.2 Projektiryhmä 5

2.3 Projektin sidosryhmät 6

3. PROJEKTIN TOTEUTUS 7

3.1 Projektin vaiheet 7

3.2 Työmäärät 7

3.3 Käytetyt työkalut ja menetelmät 9

3.4 Toteutuksen onnistumiset 10

3.5 Toteutuksen ongelmat 10

3.6 Knowledge transfer (eli mitä me tästä opimme) 11

4. PROJEKTIN LOPPUTUOTE 15

4.1 Mitä saavutettiin? 15

4.2 Miten voidaan jatkaa? 15

4.3 Arviot projektin onnistumisesta 15

5. LOPPUSANAT 16

# JOHDANTO

Tämä on Ohjelmistotuotanto-kurssin R19 ryhmän loppuraportti liittyen VuokraToimistot Oy:lle toimitettavaan ohjelmaan. Loppuraportissa käsitellään sitä, että kuinka, miten ja milloin ohjelma laadittiin VuokraToimistot Oy:n tarjouspyynnön pohjalta. Lisäksi pyrimme tuomaan esiin projektin esiintuomia haasteita ja siirtää tästä opittua informaatiota eteenpäin.

VuokraToimistot Oy toimitti projektiryhmä R19:ta tarjouspyynnön ohjelmaa varten, joka mahdollistaisi heidän nostamaan heidän myyntinsä ja asiakaspalvelun insaktiin fokukseen ja näin pyrkimään parantamaan asiakaspalvelua ja myyntiä. Ohjelma mahdollistaa muutosten tekemisen asiakkaiden varauksiin ja tiloihin, joita toimipisteet vuokraavat eteenpäin palveluineen ja laitteineen.

Ohjelma toimii javaa käyttäen ja se saa tietonsa tietokannasta, joka on aluksi projektiryhmän määrittämässä pilvessä. Tämä mahdollistaa kaikkien toimipisteiden keskinäisen työskentelyn ja ylemmän johtoportaan valvonnan koskien myyntiä. Projektiryhmä on arvioinut tietokannan, sekä ohjelman itsessään kestävän varmasti vaaditun kahdeksan toimipisteen yhtenäisen käytön.

Projektiryhmällä on apunaan ollut tarjottu Microsoft Azure DevOps, suositeltu versionhallinta. git sekä tietokanta Azuren pilvessä. Lisäksi ryhmäläiset ovat käyttäneet parhaakseen näkemiä kehitysympäristöjä (5 Netbeans, 1 Intteli J).



# PROJEKTIN ORGANISOINTI

## Projektin kuvaus ja tavoite

Projektin liittyy VuokraToimistot Oy:n lähettämään tarjouspyyntöön ohjelmasta, jonka avulla kyseinen yritys voi uusine toimipisteineen hallita vuokrattavia tiloja, näiden laitteita, palveluita, asiakkaita sekä laskuttaa asiakkaita varauksien mukaan.

Ohjelma tuotettiin käyttäen javafx: llää, Scenebuilderia, Java 8.1 ja ohjelmassa käsiteltävät tiedot tallennettiin mySQL:lla laadittuun tietokantaan, joka alustavasti sijaitsee projektiryhmän päättämässä pilvessä. Lisäksi projektiryhmän käytössä oli Microsoft DevOps ja versionhallinta Git.

Tavoitteet olivat asiakkaan tarjouspyynnön perusteella seuraavat:

* toimipisteiden hallinta – mahdollisuus lisätä, muokata ja hallita yrityksen toimitiloja yleisesti
* palveluiden hallinta – mahdollisuus hallita, luoda uusia ja manageroida niiden käyttöä toimistotilojen mahdollisuuksien mukaisesti
* toimistotilavarausten hallinta – mahdollisuus hallita toimipisteiden alla hallussa olevia vuokratiloja
* asiakashallintajärjestelmä – hallita asiakkaita ja heidän varauksiansa
* laskujen hallinta ja seuranta – toiminto toimipisteille, joka mahdollistaa laskun laatimisen asiakkaan varauksen perusteella, joko sähköisesti tai postitse
* tilojen palveluiden ja laitteiden hallinta – luoda asiakkaalle mahdollisuus hallita varauksen perusteella vuokratilojen palveluita ja laiteta sekä toimipisteelle hallita niitä

Ohjelma luotiin kolmessa vaiheessa (Määrittely ja suunnittelu, Toteutus, Testaus) vesiputousmallin mukaisesti aikavälillä 29.3.2021 – 20.5.2021.

## Projektiryhmä

**Tuukka Gavrilov** Projektipäällikkö, tietokanta vastuumies

**Sauli Karvinen** Käyttöliittymä Suunnittelija ja koodaaja

**Marko Munsterhjelm** Käyttöliittymä Suunnittelija ja koodaaja

**Aleksi Rytkönen** Asiantuntija ja testaaja

**Ville Ahonen** Asiantuntija vaatimusmäärittelijä

**Toni Cantarella** Toimintamäärittelijä

Lisäksi kukin projektiryhmän jäsenistä osallistui ohjelmakoodin ja tietokannan laatimiseen sekä niiden ongelmien ja kehityksen ympärillä pyörineeseen keskusteluun.

## Projektin sidosryhmät

Projekti oli osa ohjelmistotuotanto-kurssia, joka järjestettiin Karelia AMK:n opetuksen mukaisesti. Kurssin opettajat pitivät ryhmälle koulutuksia liittyen ohjelmakoodiin, tietokantaan ja erilaisiin ongelma tilanteisiin liittyen.

Osana projektia oli myös ohjelman esittely, joka laadittiin englanniksi kurssin English Communication Skills lisävaatimusten mukaisesti.

# PROJEKTIN TOTEUTUS

## Projektin vaiheet

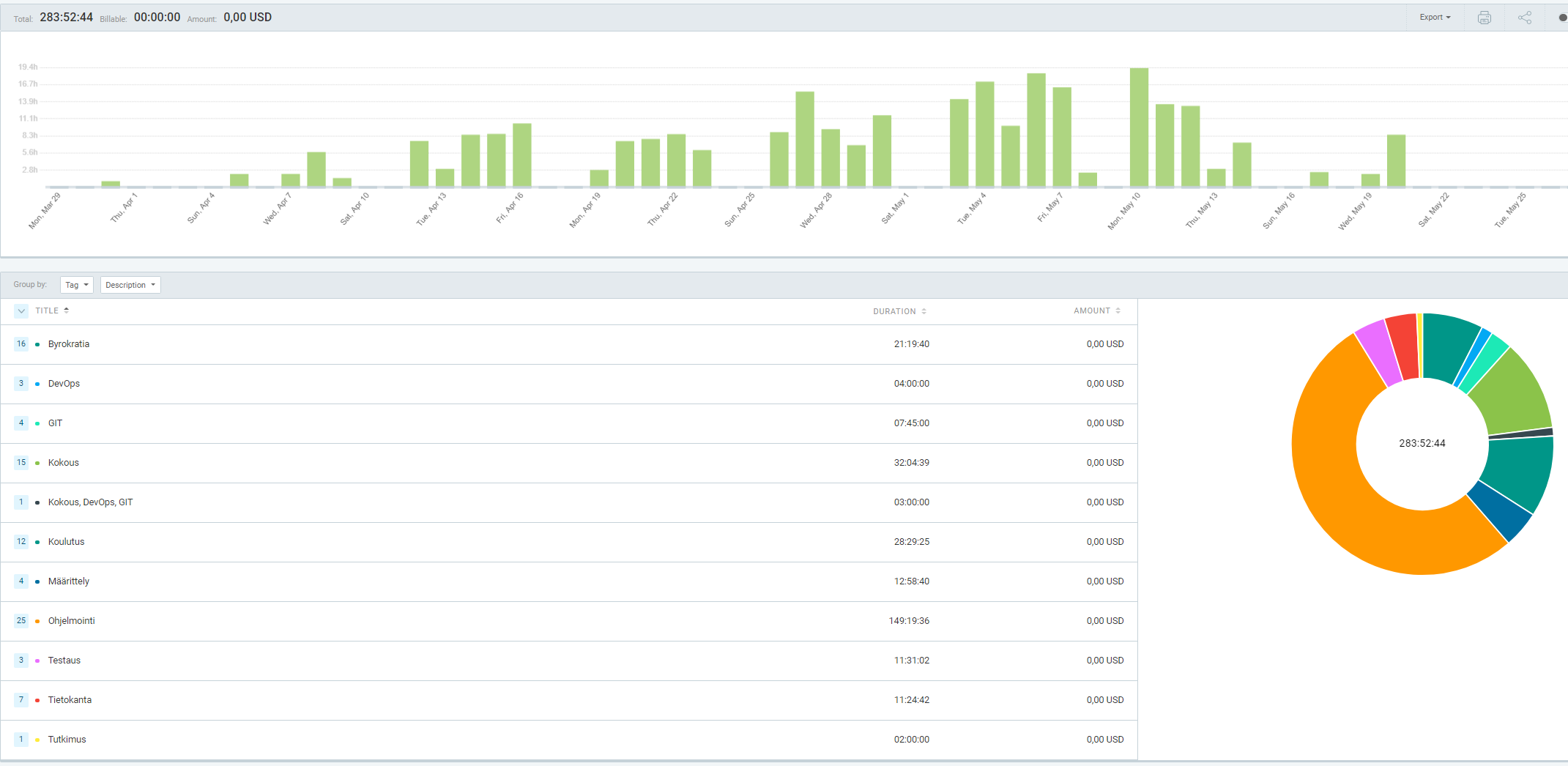
Ohjelman laatiminen suorititettiin vesiputousmallin mukaisesti jakamalla työaika kolmeen osaan: Määrittely ja suunnittelu, toteutus sekä testausvaihe päivien 29.3.2021 – 20.5.2021 välillä.

Vaiheiden työosat:

* **Määrittely ja suunnittelu 29.3–18.4** – Tuona aikana Sauli ja Marko laativat Scenebuileria käyttäen kaksi erilaista käyttöliittymä hahmotelmaa, jonka valinnan pohjalta he veivät suunnitelman pidemmälle ja loivat käyttöliittymän. Näiden tietojen pohjalta laadittiin vaatimusmäärittelyt sekä projektikortti. Samalla lähdettiin työstämään toimintamäärittelyä. Lisäksi tietokannan hahmotelma laadittiin
* **Toteutusvaihe 19.4–9.5** – Ohjelmaa ja tietokantaa lähdettiin kehittämään pidemmälle, samalla työstäen toimintamäärittelydokumenttia, jonka pohjalta ohjelman lopullinen toiminta määriteltiin
* **Testausvaihe 9.5–21.5** – Testaussuunnitelman mukaisesti laadittu testauskattavuus on arvioitu olevan 90 % ja sisältäen yksikkö ja järjestelmätestauksia. Testauksen suorittivat Aleksi ja Ville. Testauksen suorittavilla henkilöillä on pääsy ohjelman koodiin. Ohjelmaan itseensä tehtiin pieniä muutoksia testauksien pohjalta. Testaus suoritettiin mustalaatikko menetelmällä

## Työmäärät

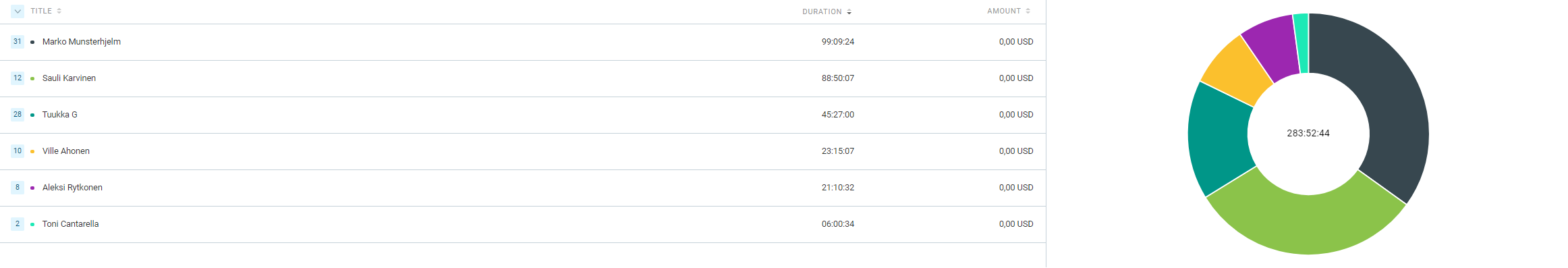
Työmäärien pääasiallinen seuranta toteutettiin käyttämällä suositeltua clokify-nimistä internetpalvelua. Kyseiseen ohjelmaan luotiin erillinen tunnus johon kaikilla oli pääsy ja jotta managerointi olisi helpompaa, tällä lisäksi pystyttiin varmistamaan kaikkien pääsy kyseiseen projektinalueelle.



[Koko projektin merkityt työt ja ajat]

* **Byrokratia –** 7,15 % kattaa alleen kaiken mahdollisiin palautettaviin dokumentteihin liittyen
* **DevOps –** 1,41 % devopsin alustaminen, tehtävien luonti ja ylläpito
* **GIT –** 2,73 % pääasiallinen ongelmien selvittely
* **Kokous –** 11,30 % maanantaiset ja perjantaiset kokoukset
* **Koulutus -** 10,04 % oppi- ja harjoitustunnit lisäksi ns. ”koodauspäivät”
* **Määrittely –** 4,57 % Tästä aika liittyy aika paljon byrokratiaan, ohjelman määrittely ja yhteistensääntöjen asettaminen
* **Ohjelmointi –** 52,60 % Ohjelmakoodin luonti
* **Testaus –** 4,06 % Ohjelman testaus mustalaatikko periaatteen mukaisesti
* **Tietokanta –** 4,02 % Tietokannan suunnittelu, luonti ja korjaus
* **Tutkimus –** 0,70 % Jos jäsenellä on tullut uusi asia vastaan, hän on merkannut käytetyn ajan mitä hän on sen opiskeluun käyttänyt

Huomioidaan tulevaisuuden varalle, että clokifyhin tulisi merkata tarkemmin työn vaiheet (suunnittelu,määrittely,testaus). Se selkeyttäisi huomattavasti kuhunkin työvaiheeseen käytettyä aikaa.



[Projektin töiden jakautuminen henkilöittäin]

* **Tuukka Gavrilov** – 45:26:00 h Byrokratia, SQL
* **Sauli Karvinen** – 88:50:07 h Ohjelmakoodi
* **Marko Munsterhjelm** – 99:09:24 h Ohjelmakoodi
* **Aleksi Rytkönen** – 21:10:32 h Testaus, vaatimusmäärittely
* **Ville Ahonen** – 23:15:07 h Testaus, vaatimusmäärittely
* **Toni Cantarella** – 06:00:34 h Toimintamäärittely

Haastattelujen perusteella suoritettu näkymättömän työn määrä arvioidaan olevan noin 5-15 % nykyisten tuntien päälle. Syitä tällaiseen ovat olleet unohdukset ja mahdolliset merkkaus ongelmat. Näistä syistä Kurssin lopullinen ja toteutunut tuntimäärä on noin **320 tuntia**, joka on noin 50 % arvioitua vähemmän.

On myös huomattavaa että kaikki jäsenet eivät liittyneet projektiin täysin samalla viivalla ja lähtökohdat ohjelmakoodiin osallistumisessa ovat olleet rajalliset. Tästä enemmän kohdassa 3.3.

## Käytetyt työkalut ja menetelmät

Projektiryhmä käyttivät kaikki Windows pohjaisia tietokoneita ja yhtä lukuun ottamatta netbeans kehitysympäristöä ohjelmakoodin tekemiseen. Ryhmän pääasiallinen keskustelumuoto oli discord palvelun kautta tapahtuva viestittely ja viikoittaiset kokoukset, joiden lisänä olivat toisinaan Markon ja Saulin johdolla vedetyt koodikoulut.

DevOpsin käyttö oli kaikille projektinjäsenille tuttua jo aiemmilta kursseilta ja tällä kurssilla sen käyttöön päästiin syventymään syvemmin. Uutena asiana kaikille tuli childtaskien ja issuieden käyttö, joka selvästi parantaa työn seurantaa. DevOpsin käytössä ei ryhmän kanssa huomattu muuta vikaan kuin erinäköisten asetuksien ja valikoiden olevan monen alavalikon takana piilossa. Esimerkki tästä on, kun yritimme vaihtaa projektivaiheen toteutukseen, mutta kaikki issuet ja taskit menivät määrittelyn puolelle. Ongelma oli, että vaikka ajoitukset oli tehty niitä ei ollut aktivoitu oikealla tavalla.

Työajan seurantaan käytimme clokifyta, josta projektipäällikön piti usein muistuttaa. Lisäksi projektiryhmä tarkkaili töiden määrää ja tehtäviä DevOpsin avulla.

Tietokantana käytimme mySQL-kieltä ja loimme tietokannan koulun tarjoamaan pilveen, johon projektijäsenillä oli pääsy sekä ohjelman itsensä kautta, myös heidiSQL ohjelmaa käyttäen. Tietokannan alkuperäinen suunnitelma luotiin käyttäen ERDplus sivustoa, jonka pohjalta luotiin relaatiokaavio ja sitä kautta mySQL lause, joka syötettiin pilveen. Tämän jälkeen tietokantaan tehtiin pieniä muutoksia kyselyillä ja tietokannan varmuuskopiota säilytettiin projektipäällikön koneella.

HeidiSQLn käyttö tietokantaa muokatessa ei ollut ongelmaton. Kokeilimme monesti muokata tietokantaa käyttöliittymän avulla, mutta kyseine tapa sai ohjelman monesti solmuun ja kaatumaan. Lisäksi vaikutti toisinaan, että pilvi oli ruuhkautunut, kun käskyt tietokantaan eivät menneet läpi.

Versionhallinnassa käytettiin Gittiä, jonka käyttö oli ryhmämme kohdalla painajaista alusta asti. Onnistuimme taiteilemaan määrittelyvaiheen yli, kunnes huomasimme seinän tulleen vastaan. Ongelmamme oli alusta asti se, ettemme olleet luoneet. gitignore tiedosta, ja tästä kohdassa ongelmat lisäävät.

Projektiryhmässämme on kaksi erittäin osaavaa koodaria, jotka suunnittelivat ja pääasiassa toteuttivatkin koko ohjelman. Heillä oli parhaat eväät ja kokemus luoda koodia, mutta se hieman kääntyikin itseään vastaan, sillä kukaan ryhmästä ei ollut aikaisemmin nähnyt sellaista tapaa koodata. Monella ryhmän jäsenellä on aukkoja varsinkin olio-ohjelmoinnin ja FXMLn kanssa, joten koodaaminen koettiin osittain jopa ylitsepääsemättömäksi ja koodin luonti kasaantui vain sen aloittaneiden harteille. Tätä kuitenkin pyrittiin paikkamaan niin, että lahjakkaiden koodareiden harteille kasattaisiin byrokraattisia toimia mahdollisimman vähän antaen heidän lahjansa kukkia mahdollisimman hyvin.

## Toteutuksen onnistumiset

Ryhmän jäsenet kokivat työn ollut erittäin opettavainen ja syventänyt osaamista laaja-alaisesti. Ongelmista, tehtävistä ja omista näkemyksistä keskusteltiin aktiivisesti ryhmän kesken discordissa, niin kokouksien kuin niiden ulkopuolella. Discord mahdollisti reaaliaikaisen keskusteluyhteyden jäsenten välillä ja varsinkin tietokannan ja ohjelmakoodin välisten ongelmiin päästiin pureutumaan nopeasti ilman että jonkin osapuolen olisi tarvinnut odottaa kauaa.

Kiveen lyödyt kaksi kokous kertaa viikossa koettiin hyväksi käytännöksi. Maanantain palaverin tarkoitus oli kerrata tulevan viikon asiat ja mahdollisesti pureutua ongelmiin, kun taas Perjantain palaverissä käytiin läpi tehdyt työt ja mahdolliset ongelmat. Tämä helpotti raportointia ja toi ryhmän jäsenille ilmi miten ja kuka oli töitä tehnyt sillä esimerkiksi koodin työstäminen on melko näkymätöntä työtä.

Hyväksi havaitittiin ns ”koodauspäivä”, jona ryhmän jäsenet kokoontuivat kuuntelemaan ja oppimaan kun pääkoodari selosti koodaustyön kulkua ja keinoja miten koodia luodaan. Tämä päivä suoritettiin toisin hieman myöhään, ja totesimme paremmaksi tavaksi sijoittaa koodauspäivä sille, päivälle kun ensimmäiset oikeat koodirivit kirjoitetaan.

Kaikilla ryhmän jäsenillä oli admin-tason oikeudet devopsiin, joka mahdollisti nopeiden muutosten tekemisen, sekä mahdollisti tasapuolisen oppimisen kyseisestä työkalusta. Uusina ja syventävinä asioina ryhmän jäsenille tuli erilaisten child-taskien ja issueiden laatiminen, jotka tekevät ohjelman käytöstä hieman selkeämpää.

Projektin alussa luotiin clokifyhin uusi tili, jonka tarkoitus oli mahdollistaa kaikille ryhmän jäsenille pääsy tuntitietoihin ja muokata niitä tarvittaessa. Tämä mahdollisti myös helpomman tuntiraporttien luomisen.

Kaikinpuolin ryhmän kaikki jäsenet olivat erittäin tyytyväisiä ryhmätyön kulkuun ja lopputulokseen.

## Toteutuksen ongelmat

Kaikkien ryhmäläisten mielestä suurin ongelma oli versionhallinta gitin kanssa. Nojaten edeltäviin kokemuksiin gitin kanssa emme osanneet kuvitellakaan minkälaisia ongelmia. gitignore tiedoston pois jättäminen voisi tulla tuodessaan. Ongelmat gitin kanssa alkoivat heti alkumetreillä, mutta suurin osa ongelmista korjautui aina kun poisti omalta koneeltaan vanhentuneen version ja kloonasi uuden omalle koneelleen. Ongelmaksi huomattiin netbeansin tapa tehdä ajonaikaisia tiedostoja, jotka eivät toimi toisten koneilla.

Ongelmat kuitenkin alkoivat käymään ylitsepääsemättömiksi, kun tuotantovaihe saapui ja koodaajia alkoi olla useampia. Yritimme kokouksessa löytää ratkaisua ja yksi mahdollinen sellainen oli -rm komennoilla lopettaa oman gitin ongelma kansioiden selaaminen. Tämä oli viimeinen niitti ryhmälle ja päädyimme osin opettajien avustuksella ydinoptioon, joka tarkoitti repon poistamista ja uudelleen luontia. gitignore tiedoston kanssa, sillä jostain syystä kyseistä tiedostoa ei pysty kesken projektin enää luomaan. Kyseisen tiedoston luonti ei ollut sekään mutkatonta, sillä alunpitäen gittiä korjanneen jäsenen tietokone oli osittain kykenemätön luomaan kyseisen tiedoston, sillä luodessaan sen se teki siitä .txt päätteisen tehden siitä näin vääränlaisen. Ongelma korjattiin luomalla kyseinen tiedosto toisen koneella komentoriviä käyttäen, tällöin on mahdollista pakottaa tiedosto tietynlaiseksi. Huomioitavaa on, että jos koneella on Visual studio niin Windows tunnistaa kyseisen tiedoston gittiin liittyväksi.

Seuraavaksi suurimmat ongelmat liittyivät taulujen relaatioihin. Pääkoodareiden turhautuminen oli käsinkosketeltavaa, kun heidän alkaessa laatia ohjelmakoodia, tulivat he jatkuvasti vastaan ongelmaan, jossa heidän on ensimmäiseksi luotava toiminto toiselle taululle, jotta he voisivat käyttää nykyistä taulua. Asiaa ei auttanut HeidiSQLn toimimattomuus käyttöliittymän kautta tehtyihin muutoksiin, joten kaikki muutokset jouduttiin tekemään kyselyillä.

Mietimme ryhmässä, että olisiko tietokannan luominen ollut mahdollista täysin itsenäisillä tauluilla ja yhdistellä niitä tarvittaessa vain LEFT JOIN syntakseilla? Ohjelma ja tietokanta olivat tässä vaiheessa kuitenkin niin pitkällä, että päätimme jatkaa vanhaan malliin ja hyväksyä relaatioiden kanssa painimisen.

## Knowledge transfer (eli mitä me tästä opimme)

Projekti oli erittäin opettavainen ryhmän jäsenille kaikkiaan. Uutta asiaa tuli ja erinäköiset ongelmat loivat lujaa osaamista tulevaisuutta varten. Ryhmän sisällä onkin suurten ongelmien ratkaisujen myötä (kuten git-ongelmat) jaettu linkkejä devops työpaikoista joissa toivotaan kyseisen työkalun hallintaa.

Yhteiset koodaushetket ovat syventäneet ryhmän kaikkien jäsenten osaamista koodaamisen suhteen. Pääkoodarit Marko ja Sauli opettivat täysin uudellaisen tavan ryhmän jäsenille luoda koodia ja varsinkin olio-ohjelmoinnin saralla. On valitettavaa, että muiden ryhmäläisten apu koodissa jäi pieneksi, mutta uskomme että jokainen on koodamisessa tämän kurssin jälkeen parempi ja ainakin tietoinen siitä, kuinka laatia koodia yhdessä muiden kanssa.

Vaikkei koodaaminen kaikkien leipä tällä kurssilla olutkaan, ovat kaikki ryhmän jäsenet saaneet koodin toimimaan omalla koneella ja oppineet kuinka koodata yhteistä koodia omalla kehitysympäristöllä (Kehitysympäristö kurssin koodi oli niinkin yksinkertaista eikä mennyt syvemmälle IDEjen käyttöön Gitin kanssa).

DevOpsin käyttö tällä kurssilla pidettiin ns. ”avoimena”, jotta kaikilla on mahdollisuus luoda ja merkata taskeja ja issueita. Näitä käsiteltiin yhdessä monissa viikottaisissa palavereissä.

* **Tuukka Gavrilov** - Projektipäällikkönä olen pyrkinyt olemaan nopea vastaamaan muiden kysymyksiin, sekä yrittänyt selvittää niitä mahdollisimman tehokkasti. Olen oppinut ohjelmistotuotannosta paljon ja erityisesti suunnitteluvaihe on todistanut itsensä tärkeyden tällä kurssilla. Katsellessa taaksepäin projektia ja tehtyjä tunteja koen olevani hieman epäonnistunut työn organisoijana, sillä paljon tästä projektista (erityisesti ohjelmakoodista) kaatui kahden henkilön niskaan. Olisinko voinut organisoida työtä paremmin? Johtaminen olisi varmasti ollut helpompaa jos omat koodaus taitoni olisivat samalla tasolla pääkodareiden kanssa. Tietokanta-vastaavana opin paljon tietokannan käsittelystä tällaisen ohjelman luonnissa. Tuntui vaikka miten olisimme suunnittelut tietokannan niin sen käyttäminen oli erittäin vaikeaa, aluksi nimeämiskäytäntö oli hukassa ja lopuksi suuret ongelmat relaatioiden kanssa.
* **Sauli Karvinen** - Projekti on ollut todella opettavainen. Hyvän suunnittelun tärkeys ja/tai sen puute on korostunut todella vahvasti projektia tehdessä. GIT ympäristönä on projektin edetessä tullut enemmän ja enemmän tutuksi, mikä on mahtava juttu. En ole aiemmin elämässäni juurikaan tehnyt projektiluontoisia töitä, joten pitkäjänteiseen projektityöskentelyyn ja yhdessä tekemiseen liittyen on oppinut paljon asioita tätä tehdessä. On tullut huomattua, että toimintojen toteuttamista saa priorisoida melko vahvasti sen mukaan, kuinka paljon aikaa on jäljellä ja kuinka paljon aikaa on valmis käyttämään projektiin aikaa. Kuten yllä on mainittu, kaikkia haluamiaan toimintoja ei välttämättä ehdi toteuttaa, vaan prioriteetiksi muodostuu se, että ohjelma toteuttaa karkeasti ne toiminnot mitä asiakas on tilannut. Esimerkiksi käyttäjän virheiden käsittelyssä on varmasti puutteita, joka varmasti tulee esiin testausvaiheessa ja testien tuloksia odotankin mielenkiinnolla. Ohjelman ulkoasua olisi myös kiva viilata mukavamman näköiseksi jo senkin takia, että se osoittautui todella hauskaksi ja koukuttavaksi hommaksi ja sitä kiinnostaisi opetella lisää. Tähän ei kuitenkaan ylimääräistä aikaa ole tässä projektissa juuri jäänyt.
* **Marko Munsterhjelm** - Ryhmätyö on voimaa. Kun ongelmia ratkotaan yhdessä, ei ole sellaista ongelmaa jota ei pystyttäisi ratkaisemaan. Kun useampi henkilö miettii asioita, saadaan useampia näkökulmia asioihin. Näin omia ajatuksia voi virkistää ja kehittää. Ratkaisuja voi olla useampikin, ryhmässä löytynee paras ratkaisu, kun on useampi miettimässä. Hyvän ryhmätyön perusta on siinä, että jokainen osallistuu. Tässä ryhmässä oli ilo työskennellä. Ohjelmistotuotanto on erittäin monimutkainen ja sekava polku. Se selkiytynee ajan ja kokemuksen mukana. Vaikka omasta mielestä projektin alussa mietti kaikkia mahdollisia kohtia ohjelmiston suunnittelussa, niin edelleen ohjelmiston edetessä törmättiin uusiin ja odottamattomiin asioihin. Edellisissä kursseissa opitut asiat eivät olleet ihan niin hyvässä muistissa kuin itse luuli, monia asioita piti kerrata uudelleen. Hyvin suunniteltu tosiaan helpottaa koodaamisvaiheessa, silti voisi edelleen paremmin suunnitella. Kokemuksen karttuessa helpottunee. Koodaaminen vaatii harjoittelua, vielä kun olisi aikaa siihen.
* **Aleksi Rytkönen** - Opin että ryhmätyöskentely voi olla joustavaa ja samalla aikaa hyvin stressaavaa jos huomaat jääväsi työtuntimäärissä muista jälkeen. Myös jos projekti on organisoitu ja johdettu järkevästi, työskentely hoituu hyvin sujuvasti.
* **Ville Ahonen** - Opin tämän projektin aikana heräämään maanantaisin kello 8.55 iloisin mielin maanantaipalaveriin, taistelemaan Git -hirviöitä vastaan ja lukemaan kaverin kirjoittamaa koodia hieman paremmin. Azure DevOps ja Git tuli taas hieman tutummaksi, vaikka niihinkin opeteltavaa vielä jäi syvemmältä. Ryhmätyöskentely toimi täydellisesti, kaikki oli mukana ja aktiivisena heti alusta asti ja homma toimi kuin rasvattu meetwursti ruisleivällä.
* **Toni Cantarella** – Oma roolini kyseisessä ryhmätyössä jäi hiukan pieneksi myöhäisen liittymiseni takia, mutta olen kuitenkin oppinut joitain tärkeitä asioita ryhmätyöhön liittyen varsinkin ohjelmistotuotannon saralla. Pääasiallisena seikkana ryhmätyöskentelyssä on ilmennyt se, että vaikka kuinka tahtoisi jakaa ja rajata selkeitä tehtäviä eri henkilöille, niin ainakin tällaisessa projektissa on hyvä jokaisen olla joko mukana tai ainakin hyvin perillä jokaisen osa-alueen tapahtumista. Suunnittelu vaikuttaa testaukseen, testaussuunnitelmalla on vaikutuksia ohjelmoinnin toteutukseen, tietokannan toteutuksella on vaikutusta ohjelmistosuunnitteluun ja niin edespäin. Olisi hienoa, jos jokainen tehtävä voitaisiin selkeästi irrottaa toisistaan, mutta se ei näytä ainakaan tässä vaiheessa kovinkaan mahdolliselta tai ainakaan suositeltavalta. Toisin sanoen, tiimin yhtenäisyys ja sisäinen kommunikaatio on äärimmäisen tärkeää. Toisekseen olen huomannut, kuinka tärkeää on huomioda myös toisten mahdollisia puutteita osaamisessa, koska yhden epäselvyyden tiedustelu voi avata oven muille tärkeille kysymyksille, joiden ratkominen on valtava taakan pudotus myöhemmin. Siksipä "tyhmien" kysymysten kysyminen alkuvaiheessa on osaamisen kasvattamisen lisäksi tulevien ongelmien ennaltaehkäisyä.

Lisäksi alkuperäinen tarkoitus oli toteuttaa ohjelma tietokantahakujen puolesta niin, että tietokantayhteys avataan vain siksi ajaksi kun joku kysely suoritetaan ja suljetaan heti sen jälkeen. Tämän vuoksi jokaisessa tietokanta-metodissa on ”näppärästi” toiminnallisuus tähän suoraan metodin sisällä. Myöhemmin ohjelmassa huomasi, että joitakin asioita pitää hakea ehkä loopissa, tai sitten muuten vain suorittaa putkeen useampi tietokantahaku. Ohjelma hidastuu tällöin melko paljon, kun jokainen kysely avaa ja sulkee yhteyden ja olisikin varmaan viisaampaa avata tietokantayhteys siellä missä tietokannan metodeja kutsutaan ja sulkea yhteys sen jälkeen, kun kaikki haut on tehty.

Ohjelmassa suoritetaan monessa näkymässä aina uusi tietokantahaku, kun hakutuloksia rajataan (esim. toimitilan mukaan). Ohjelmassa voisi ehkä hyödyntää enemmän ohjelman sisällä jonkinlaisia lista-/hajautustaulu ratkaisuja haettujen tietojen säilömiseen jolloin ohjelma nopeutuisi jos näkymässä muutetaan rajauksia useita kertoja.

Ryhmä on huomannut, että toimintojen toteuttamista saa priorisoida melko vahvasti sen mukaan, kuinka paljon aikaa on jäljellä ja kuinka paljon aikaa on valmis käyttämään projektiin aikaa. Kuten yllä on mainittu, kaikkia haluamiaan toimintoja ei välttämättä ehdi toteuttaa, vaan prioriteetiksi muodostuu se, että ohjelma toteuttaa karkeasti ne toiminnot mitä asiakas on tilannut. Esimerkiksi käyttäjän virheiden käsittelyssä on varmasti puutteita, joka varmasti tulee esiin testausvaiheessa ja testien tuloksia odotankin mielenkiinnolla. Ohjelman ulkoasua olisi myös kiva viilata mukavamman näköiseksi jo senkin takia, että se osoittautui todella hauskaksi ja koukuttavaksi hommaksi ja sitä kiinnostaisi opetella lisää. Tähän ei kuitenkaan ylimääräistä aikaa ole tässä projektissa juuri jäänyt.

# PROJEKTIN LOPPUTUOTE

## Mitä saavutettiin?

* **Toimipisteiden hallinta 100 % -** Toimipisteitä voi hallita täydellisesti
* **Palveluiden ja laitteiden hallinta 100 % -** Laitteita ja palveluita voi lisätä toimitiloihin täydellisesti ja testauksessa havaittu ongelma on korjattu
* **Toimitilavarausten hallinta 100 % -** Toimii, Ongelmat koskevat poikkeustenhallintaa kuten aloituspäivän valitseminen menneisyydestä ja aloituspäivän laitto myöhemmäksi kuin lopetuspäivä.
* **Asiakashallintajärjestelmä 100 % -** Toimii, ei ongelmia
* **Laskujen hallinta, laskutus ja seuranta ominaisuus 90 %** -Muuten toimii mutta laskujen muokkaus ei toimi SQL syntaksin vuoksi

Ohjelma toimii erittäin hyvin ja se on erittäin pitkälle kehittynyt. Tietokanta, josta tiedot haetaan toimii ohjelmaa käyttäessä, virhe syötteitä on otettu kattavasti huomioon ja ohjelma on ryhmän ylpeyden aihe. Saulin ja Markon rautainen työnjälki on luonut esimerkin ryhmäläisille kuinka ja miten näin isojen kokonaisuuksien ohjelmakoodia tulee tehdä.

Vaatimuksiin kuulunut dokumentointi on ollut aikaa vievää mutta se on ollut kattavaa ja pääsäännöllisesti hyvää.

## Miten voidaan jatkaa?

Kun ohjelmassamme käytetään useita erilaisia näkymiä, saattaa jossakin näkymässä jäädä paljon asioita piiloon, joita olisi käyttäjän mukavuuden takia hyvä olla näkyvissä ja/tai valittavissa. Jälkeenpäin joidenkin rakenteellisten/koodillisten ratkaisujen takia moni tämmöinen ”olisi kiva olla olemassa” -toiminnallisuus on siirtynyt osastoon ”jos aikaa jää niin toteutetaan”.

Palavereissa tuli monesti esille, kuinka asiakasrajapinnan lisääminen ohjelmaan toisi sille huomattavan paljon lisäarvoa ja mahdollistaisi yrityksen paremman resurssien kohdentamisen myyntiin ja muihin ydintoimintoihin.

## Arviot projektin onnistumisesta

Ohjelma on erittäin kehittynyt ja pitkälle ajateltu, sekä sen toiminallisuus on erittäin pitkälle mietitty. Kiitos tästä on tietysti pääkoodareillemme Saulille ja Markolle. Dokumentointi on ollut kattavaa ja sitä on ollut riittävästi. Ryhmätyön teko on sujunut ilman pahempia ongelmia (poisluettuna git) ja ryhmä kokee ansaitsevansa

arvosanan: **5**

# LOPPUSANAT

Käyttöliittymän puolesta on todella konkreettisesti huomannut, että sellaiset asiat jotka omasta mielestä ovat hyvin selkeitä, eivät välttämättä sitä ole käyttäjän mielestä. Tätä asiaa on hyvin tukenut meneillään oleva Ihminen ja vuorovaikutteinen teknologia -kurssi. Käyttöliittymä tuli suunniteltua ensimmäisenä projektia aloittaessa ja onkin projektin edetessä ollut silmiä avaavaa huomata joissakin kohdissa jälkeenpäin, kuinka epäselvä jokin toiminnallisuus voi käyttäjälle (ja jopa itselleen) olla mikä suunnitellessa vaikutti todella selkeältä ratkaisulta. Käyttäjän ei tulisi joutua arvailemaan kuinka jokin asia tehdään vaan ohjelmassa voisi olla enemmän selkeitä ohjeita tähän.

On myös huomattu selkeästi, että suunnitteluvaiheessa ei välttämättä nähdä kaikilta osin kokonaiskuvaa tietyissä asioissa ja kuinka joidenkin suunnitteluvirheiden korjaaminen myöhemmin toteutusvaiheessa osoittautuu todella aikaa vieväksi ja aikataulun puitteissa jopa mahdottomaksi korjata. Suunnitellessa jokin asia/toiminnallisuus tuntuu mielessä todella selkeältä, mutta kun sitä alkaa toteuttamaan niin huomaa, että asia ei olekaan niin yksinkertainen kuin miltä se ensin vaikutti ja yksi huomioimaton asia suunnittelussa voi vaikeuttaa usean eri ohjelman osan toteuttamista myöhemmin.

Lopussa kiitos seisoo, ihminen yleensä katsoo tekemiään asioita tulevaisuudesta erittäin kriittisesti ja toteaa: ”että tuon olisin tehnyt toisin”, ”ehkä näin olisi ollut parempi” tai” miksi tein näin?”. Kaikesta itsekriittisyydestä ja negatiivisen ajattelumme seasta emme toisinaan kykene erottamaan kyseistä tapahtumaa oppimiseksi.

Ohjelmistotuotanto eroaa paljon, muista tämän kouluvuoden aikana tehdyistä ryhmätöistä sen hyvin toisistaan erilaisten osien tuomisesta yhteen. Milloin on kyettävä luomaan toimivaa tietokantaa ja muokkaamaan sitä ohjelman toiminnan vaatimalla tavalla ja toisinpäin. Voimme lisäksi pyrkiä pienentämään tällaisia ongelmia hyvällä suunnittelulla, mutta suunnitteluakaan ei voi tehdä niin hyvin etteikö ongelmia ilmenisi, joten ohjelman onnistumisen sala on aktiivisessa ryhmässä, joka pystyy vastaamaan nopeasti ongelmiin ja toimimaan on avain onnistuneeseen projektiin, mikä tämä projekti todellakin oli.