LAPIN AMK
Lapland University of Applied Sciences

SUUNNITELMA TIETOKANTASOVELLUKSESTA

Aikio-Halminen Minna

Opinnäytetyö

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

2021

LAPIN AMK
Lapland University of Applied Sciences

Opinnäytetyön tiivistelmä

Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Tekijä	Minna Aikio-Halminen	Vuosi	2021
Ohjaaja(t)	Juha Orre		
Toimeksiantaja			
Työn nimi	Suunnitelma tietokantasovelluksesta		
Sivu- ja liitesivumäärä	42 + 5		

Kyseessä oli toiminnallinen opinnäytetyö, joka etsi digitaalista apua toimitilojen alivuokrausprosessin hallinnointiin. Tutkittu työtehtävä ei ollut organisaation ydintoimintaa, vaan se hoidettiin muun työn ohessa. Organisaatio oli vastuussa kiinteistön omistajalle alivuokraamistaan tiloista. Tehtävään tarvittiin juuri riittävän rajattu ja riittävän laaja ratkaisu sujuvoittamaan vuokraustoiminnan hoitamista.

Laadullinen tutkimusote aloitti tutkimuksen sekä toteutui muun muassa vaihtoehtojen analysoinnissa. Konstruktiivinen työote keskittyi ongelman yksilöimiseen ja digitaalisen ratkaisun etsimiseen. Viitekehys rakentui puhtaasti työntekijän näkökulmasta.

Tuloksena oli suunnitelma tietokantasovelluksesta, jonka selkeä ulkoasu nopeuttaa sen käyttämistä. Samalla se tekee työtehtävästä selkeämmän hallinnoida. Tekijä on tuntenut tarkalleen tehtävän työvaiheet ja tarpeen, jota varten sovellus on luotu. Työskentelyn myötä ilmaantui uusia tarpeita ja ideoita jatkokehitykseen. Loppujen lopuksi yksinkertaiselta tuntunut työtehtävä sisältää useita eri työvaiheita ja tarvitsee erilaisia toimintoja.

Opinnäytetyössä kehitettiin täsmätyökalu yhden työtehtävän tueksi. On kuitenkin olemassa todella laajasti valmiita työkaluja, sovelluksia ja järjestelmiä, joista voi hyvinkin löytyä apua. Opinnäytetyössä löydettiinkin myös valmis sovellus, jolla ratkaistiin kalenterin tarve. Sovellusten integraatiota tarvitaan nykyään lähes kaikissa hallinnollisissa tehtävissä, mikä on hyvä ottaa huomioon, kun digitarpeisiin haetaan ratkaisuja.

Avainsanat	relaatiotietokanta, tietokantasovellus, käyttöliittymä
Muita tietoja	Opinnäytteessä kehitettiin täsmätyökalu yhden työtehtävän hoitamiseksi

LAPIN AMK
Lapland University of Applied Sciences

Abstract of Thesis

Business Information Technology
Bachelor of Business Administration

Author	Minna Aikio-Halminen	Year	2021
Supervisor	Juha Orre		
Commissioned by			
Subject of thesis	Planning a Database application		
Number of pages	42 + 5		

This is a functional thesis which searched for digital help for managing a subleasing process. The task studied was not one of the organization's core tasks. It was taken care of in addition to other work. The organization was responsible to the owner of the real estate in subleasing property. There was a need for a solution to ease the activity and streamline the rental business.

The qualitative research approach was used at the beginning and also in analyzing different solutions. The constructive approach concentrated on finding a suitable digital solution and reflecting the structure of the application. The frame of reference was built from the employee's point of view.

As a result, an application was developed with a simple looking user interface with a clear layout for speedy use. There is a database at the backend. The application makes the assignment easy and clear to manage. Ideas for further development arose during the research. The assignment might have seemed simple, but after all it included several phases and functions.

There are plenty of ready-to-use applications on the market. On the long run it saves resources if available applications are also searched for and examined. During this research a suitable calendar application was found, which also was taken to use. It is also good to consider the integration of applications, when searching for digital solutions.

Key words	relational database, database application, user interface
Special remarks	The thesis concentrated on designing a precise application to be used in a precise assignment.

SISÄLLYS

KESKEISET KÄSITTEET

css	Cascading Style Sheets. Kieli, jolla määritetään, miten html-elementit näkyvät päätelaitteella. (W3schools 2021a.)
ER-kaavio	Entity-Relationship model on tietokannan rakenteen kuvaustapa. Se kuvaa taulujen välisiä suhteita ja mallia käytetään tietokantasuunnittelussa. (Guru99 2021.)
html	HyperText Markup Language on sivunkuvauskieli, joka säätelee sivun rakennetta ja näyttää sivun elementit oikein selaimessa (Juselius 2004). Tiedoston pääte on html.
Käyttöliittymä (UI)	Käyttöliittymä (userinterface) on käyttäjälle näkyvä ohjelman osa, jolla käyttäjä syöttää ja vastaanottaa tietoa. Graafisessa käyttöliittymässä tietoa siirretään klikkaamalla hiirellä esimerkiksi painikkeita tai valikoita. Merkkipohjainen käyttöliittymä näyttäytyy käyttäjälleen ruudulliselta tekstiä ja merkkejä. Merkkipohjaiseen käyttöliittymään komennot syötetään näppäimistöllä komentoriville. (Helsingin yliopisto.)
MySQL	Relaatiotietokantojen hallintajärjestelmä (w3schools 2021b).
php	Hypertext Prepsosessor. Ohjelmointikieli dynaamisten web-sivujen luomiseksi. (w3schools 2021f.)
Relaatiotietokanta	Toisiinsa yhteydessä olevat nimetyt taulut, joissa on nimetyt sarakkeet. Tieto tallentuu taulun riveille tietueina. (Oracle.com 2021.)
SQL	Structured Query Language, jolla käytetään tietokantoja tehden erilaisia kyselyjä (w3schools 2021g).

1 JOHDANTO

1.1 Opinnäytetyön taustaa

Rovaniemen steinerkasvatus ry on yleishyödyllinen yhdistys, joka ylläpitää perusopetus- sekä päiväkotitoimintaa. Yhdistys alivuokraa toimitilojaan kerryttääkseen varainhankintansa kassaa. Työskentelen yhdistyksen toimistossa, jonka tehtäväkenttä on hyvin monipuolinen. Suurin osa toimiston työajasta kuluu taloushallinnon ja koulusihteerin tehtäviin. Lisäksi tehtäviin kuuluu muun muassa varainhankinnan tehtäviä, kuten toimitilojen alivuokraus. Koska toimistossa on vain yksi työntekijä, työaika on käytettävä tehokkaasti ja työtehtävät saatava sujumaan nopeasti.

Opinnäytetyö kehotetaan tekemään itseä kiinnostavasta aiheesta. Digitalisaatioon keskittyvä koulutus ohjaa etsimään digitaalisia ratkaisuja toiminnan tehostamiseksi. Koulutus sisälsi myös ohjelmointia. Koska opinnäytetyössä on mahdollisuus käyttää oppimaansa tietoa, alivuokraustoiminta tuntui sopivalta lähteä tutkimaan. Alivuokrausprosessi sisältää monta yksittäistä työvaihetta. Kouluyhdistys ei omista kiinteistöä, vaan avaimet ja kulkukortit tilataan kiinteistönomistajalta. Työntekijä toimii välittäjänä; hän luovuttaa avaimen ja kulkukortin alivuokralaiselle. Kulkukortteihin liittyy kulkuoikeuksien koodaus, jonka tekee kiinteistönomistaja. Laskutukseen tarvitaan alivuokrausten oikeat tiedot. Lisäksi tilojen varaus- ja käyttötilanne täytyy olla ajantasaisesti henkilökunnan tiedossa.

Tehtävän eri vaiheissa tarvittavat tiedot hajaantuvat helposti eri sijainteihin ja alivuokrauksien hallinnointi on sekavaa. Yhteydenotto vuokratilan tarpeesta voi tulla eri kanavia pitkin keskeyttäen rutiinitehtävien hoitamisen. Jokainen alivuokrasuhde on erilainen ja vuokrattavia tiloja on useita. Ongelmaan haetaan apua, joka suoraviivaistaisi, helpottaisi ja nopeuttaisi työskentelyä. Tämä vapauttaisi aikaa muihin tehtäviin, mikä vaikuttaisi työnhallintaan ja siten myös työssä jaksamiseen.

Ratkaisua etsitään tavallaan erityistilanteessa. Tehtävää varten ei ole valmiita käytänteitä ja kiinteistönomistajaa tarvitaan kulkukorttien ja avainten hallinnointiin. Yhdistyksen työntekijä luovuttaa vuotuisesti eteenpäin kymmeniä avaimia ja kulkukortteja, joiden käyttöajat vaihtelevat suuresti. Kulkuoikeuksia on pystyttävä

seuraamaan tehokkaasti ja tiedon on täsmättävä kiinteistönomistajan rekisteriin. Laskutus tarvitsee niin ikään oikeat tiedot.

1.2 Opinnäytetyön tarkoitus ja tavoite

Tilojen alivuokrausprosessi sisältää työntekijän näkökulmasta monta eri työvaihetta. On helppo tunnistaa prosessi, yksilöidä yksityiskohdat ja kehitystarpeet. Tilavuokraustoiminta tulisi voida hallinnoida siten, että kaikki relevantti tieto löytyy nopeasti ja ajantasaisesti. Tarkoituksena on etsiä työtehtävään apua digitalisaation keinoin. Tutkimus voi tuoda esiin myös muita ideoita vuokraustoiminnan tehostamiseksi.

Kaisa Turpeenniemen (2020, 17–18) kartoittamat opinnäytetyöhön liittyvät aihealueen valinnan ja rajaamisen etenemisvaiheet auttavat näkemään asian selkeästi. Ongelmanratkaisussa kannattaa keskittyä juuri niihin peruslähtökohtiin, perustoimintoihin, jotka tehdään konkreettisesti. Tilavuokrausten hallinnointi koetaan ongelmalliseksi hoitaa. Tehtävä koostuu useasta vaiheesta vuokrauksen edetessä. Vaiheiden kartoitus antoi pohjan sille, millaista ratkaisua tarvittaisiin ja mitä kannattaisi kokeilla.

Tavoitteena kehitystyössä on etsiä ratkaisu, jossa alivuokraustoiminta voitaisiin mahdollisesti hoitaa yhdessä sijainnissa. Työtehtävä tulisi sujuvammaksi hoitaa, mikä säästäisi työaikaa. Ratkaisu voisi vaikuttaa positiivisesti myös yleensä työnhallintaan. Tutkimusongelma on siis käytännön ongelma: miten tilojen vuokrauskäytäntöä voidaan tehostaa sitä työnään hoitavan työntekijän näkökulmasta?

2 TEORIAN KAUTTA RATKAISUUN

2.1 Teorian toteutuminen käytännössä

Lähtökohta ongelmassa oli alivuokrausprosessi, jota haluttiin sujuvoittaa. Tehtävä tarvitsi kehittämistä, koska se tuntui sekavalta kokonaisuudelta, jonka eri vaiheisiin palattiin, kun alivuokrausprosessi eteni. Kun esimerkiksi laskutusta alettiin tehdä, tietoa voitiin etsiä vuokrasopimuksista, tilakalenterista, avainten luovutusrekisteristä ja sähköpostista. Kokonaisuus ei ollut helposti käsiteltävissä, vaan jokaisen alivuokrauksen tietoja etsittiin useasta eri paikasta. Vuokralaisten varausajat saattoivat muuttua kesken vuokrasuhteen. Korona-aikana tämä tapahtui kaikkien vuokralaisten kohdalla.

Tilavuokraus olisi kokonaisuutena hallittu, kun jokainen yksityiskohta olisi hallittu. Työntekijänä tunnen prosessin ja tunnistan hyvin ne vaiheet, jotka jäävät helposti huomiotta tai kokonaan hoitamatta. Prosessia haluttiin kehittää siten, että jokainen yksityiskohta hoidettaisiin tehokkaasti. Työn kuormittavuus ei lisääntyisi, vaan vaikutukset näkyisivät ajankäytön tehostumisessa ja siten myös työnhallinnassa. Oli tarpeen pohtia, mitä vaihtoehtoja ongelman ratkaisemiseksi voisi löytyä.

Yleinen kuva tilanteesta johti miettimään, mitä prosessissa tapahtuu. Prosessi koostuu yksittäisistä tapahtumista, joita ovat esimerkiksi tilojen varaustilanteen tarkistaminen, vuokrasta sopiminen, avaimen luovutus ja palautus. Yksittäiset tapahtumat ovat sovellussuunnittelun taustatietona. Ne vaikuttavat esimerkiksi tietokannan suunnitteluun. Yleiskäsitys ongelmasta johti yksityiskohtien tunnistamiseen ja lopulta ratkaisuun.

Konstruktiivinen ote keskittyi ensin ongelman tunnistamiseen. Ongelmaksi tilavuokrauksien hoitamisessa koettiin tiedon hallinta. Pohdittiin, miten sekava työtehtävä saataisiin toimivaksi. Kaikki tarvittava tieto oli olemassa, kuten esimerkiksi tilatarve päivämäärineen ja kellonaikoineen, sovittu hinta ja laskutuskäytäntö, avaimen numero sekä kulkukortti ja kulkuoikeuden ajankohta. Tieto tuntui kuitenkin olevan hajallaan. Esimerkiksi avaimen palauttamatta jättäminen tai kulkuoikeuden päättyminen saattoi jäädä huomaamatta. Tiedon ja siten tehtävän hallintaan lähdettiin etsimään toimivaa ratkaisua.

Laadullinen tutkimusote toteutui ongelman kartoitusvaiheessa. Mietittiin, mikä on itse ongelma ja miten sitä voitaisiin lähteä ratkaisemaan. Tietokantaa ja sovellusta testattiin ja arvioitiin koko kehityksen ajan. Sovelluksen tulisi auttaa juuri tietyissä toiminnoissa ja nopeuttaa sekä selkeyttää tehtävän hoitamista. Testaus ohjasi suunnittelua, koska silloin nähtiin, mitä toimintoja tarvittiin ja miten niiden tulisi toimia. Testaus toi ilmi esimerkiksi sen, minne ja miten tieto olisi loogista sijoittaa sovelluksessa.

Sovelluksen koekäytöt kuvattiin. Käyttäjäkokemukset antoivat hyviä huomioita käyttöliittymän ulkoasusta ja tiedon sijoittelusta. Ne vaikuttivat tiedon loogiseen sijoitteluun ja siten suoraan sovelluksen toimivuuteen. Sovelluksen tulee olla käyttäjäystävällinen ja vastata ongelmanratkaisun tarpeeseen.

Ongelmanratkaisun tuloksia arvioitiin tarkoituksenmukaisuuden näkökulmasta. Sovellusta ei tulla käyttämään mihinkään muuhun tarpeeseen, joten siihen ei haluttu ylimääräisiä ominaisuuksia. Ratkaisu kokoaa vuokraustiedot siksi aikaa, kun koko prosessi on viety läpi. Sovellus on täsmätyökalu, jota käytetään vain sen aikaa, kunnes prosessi on hoidettu loppuun. Mitään tietoa ei ole tarkoitus säilyttää sovelluksessa pysyvästi.

Tarkoituksenmukaisia kehitysideoita tehtävän hoitamisen käytäntöihin löydettiin matkan varrella. Vuokrasopimus kannattaisi valmistella heti vuokrasuhteen varmistuttua eikä esimerkiksi vasta silloin, kun asiakas on tulossa allekirjoittamaan sopimusta. Tuoreena asiat ovat hyvin muistissa, eikä yksityiskohtia tarvitsisi lähteä myöhemmin muistelemaan. Asiaa sujuvoittaisi, jos vuokrasopimuslomake sijaitsisi sovelluksessa. Se mahdollistaisi sen, että ainakin osa tiedoista voisi siirtyä suoraan vuokrasopimuslomakkeelta tietokantaan.

Integraatio laskutusohjelmaan voisi helpottaa laskutusta. Se voisi olla ajankohtaista, jos vuokrausten määrä kasvaisi merkittävästi. Myös sopimuksellisiin yksityiskohtiin löydettiin ideoita tehostamaan toimintaa. Laskutus voitaisiin sopia tehtäväksi etukäteen, jolloin myös laskutustieto on tuoretta ja paremmin muistissa. Nykyinen käytäntö on laskuttaa tilavuokrat vuokra-ajan päätyttyä. Käytäntö on ollut sopiva korona-aikana, jolloin muutoksia tilavuokrauksiin voi tulla nopealla varoitusajalla. Irtisanomisajassa ja vuokran maksussa on yleisesti joustettu korona-aikana.

2.2 Tutkimusote, -menetelmä ja viitekehys

Hirsjärvi, Remes ja Sajavaara (2004, 71–74) kehottavat tutkimaan aihetta, joka kiinnostaa. He antavat useita neuvoja aiheen pohdintaan, kuten tutkimuksen tulisi olla oman tieteenalan mukainen ja tuoda uutta oppia tekijälleen. Aihe saisi olla merkityksellinen ja tuoda aiheesta jotain uutta esille. Opinnäytteessä on mahdollisuus myös osoittaa oppineensa uutta ja soveltaa sitä käytäntöön.

Tässä tutkimuksessa etsitään ratkaisua arjen työtehtävän suorittamiseen. Ongelma on yksilöity ja on päätetty etsiä digitaalista ratkaisua. Tutkimusote tässä työssä on konstruktiivinen:

1. On löytynyt relevantti ongelma
2. Työnantaja kertoi näkemyksiään tilavuokrauskäytännöistä
3. Ongelmaan ja sen ratkaisumahdollisuuksiin on perehdytty
4. On päätetty innovoida ratkaisumalli ja kehittää konstruktio juuri tähän tarpeeseen
5. Aletaan toteuttaa ja testata ratkaisua
6. Pohditaan ratkaisun soveltumisalaa
7. Tunnistetaan ja analysoidaan teoreettinen kontribuutio (Lukka 2001).

Materiaalia analysoidaan laadullisin menetelmin. Analyysi perustuu loogiseen päättelyyn ja tulkintaan materiaalin sisällöstä. Tarkoituksena on luoda selkeä, tiivis ja yhtenäinen kuva tutkimusongelman tietokokonaisuudesta. (Metsämuuronen 2002, 256.) Opinnäytetyössäni laadullinen tutkimusote toteutuu analysoimalla tutkimusongelmaa, ongelmanratkaisun vaihtoehtoja, käyttäjäkokemuksia, ratkaisun toimivuutta ja tarkoituksenmukaisuutta.

Viitekehys rakentuu työntekijän näkökulmasta: tilavuokrauskäytäntö ja sen työvaiheet selkiytetään. Kun ajantasainen tieto on yhdestä paikasta saatavilla, on tehtävä mahdollista hoitaa nopeasti ja luotettavasti. Ratkaisua haetaan puhtaasti työntekijän tarpeisiin. Vaikka tulevaisuudessa tehtävän jokainen yksityiskohta hoidettaisiin tehokkaasti, työn kuormittavuus ei lisääntyisi. Vaikutukset näkyisivät työnhallinnassa ja siten ajankäytön tehostumisessa.

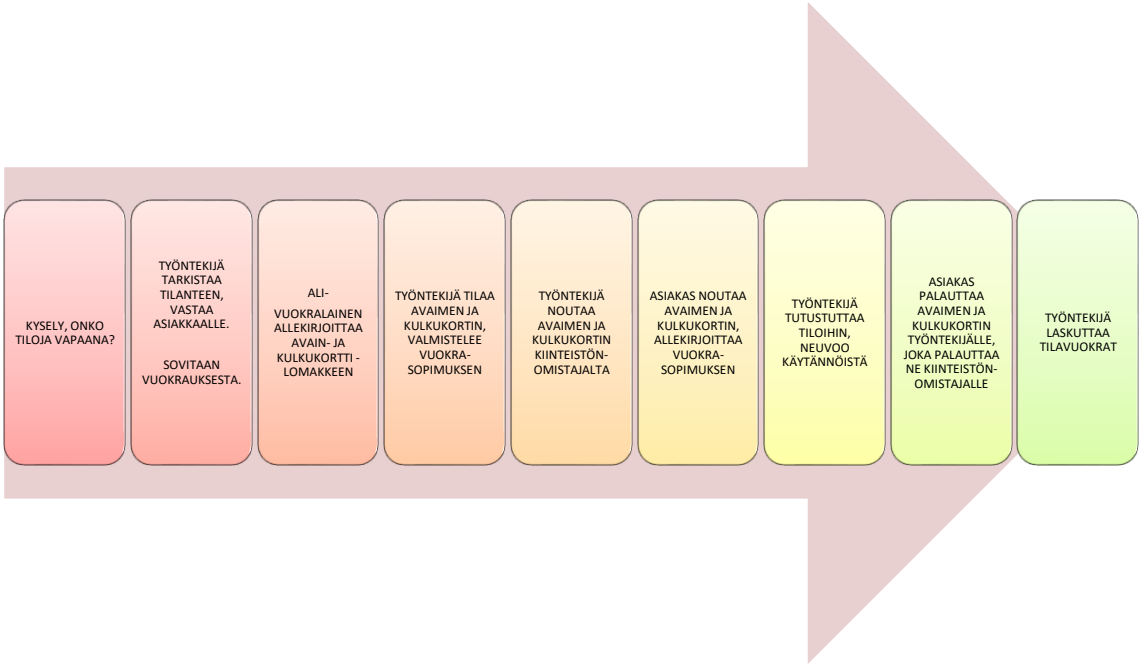
Tutkimus etenee deduktiivisesti: yleisestä yksityiseen. Taustatietojen, lähdeaineiston ja tietoaineiston perusteella päädytään tulokseen. (Hannola 2020.) Tilavuokraustoimintaa on tarkasteltu ensin kokonaisuutena, joka on tuntunut sekavalta ja rönsyilevältä. Tehtävää on lähdetty pilkkomaan osiin, jotta siihen saataisiin rakenne. Osat muodostavat kuvan prosessin kulusta, mikä auttaa näkemään eri vaiheissa tarvittavat työsuoritukset. Tehtävä tulisi voida hoitaa nopeasti ja kokonaisuuden tulisi pysyä selkeänä ja ajantasaisena. Ongelmaan halutaan löytää tehokas ratkaisu.

3 TILOJEN ALIVUOKRAUS KÄYTÄNNÖSSÄ

3.1 Alivuokrausprosessin kulku

Omien toimitilojen alivuokraus on yksi tuote yhdistyksen varainhankinnassa. Prosessissa on monta vaihetta, jotka sitovat työaikaa. Vuokrattavana on useita eri tiloja ja tarpeeseen tulisi vastata nopeasti. Koko prosessi tulisi hoitua sujuvasti ja luotettavasti.

Prosessi käynnistyy, kun asiakas kysyy tilaa vuokralle. Tilanne tarkistetaan ja vastataan asiakkaalle. Jos asiat kohtaavat, sovitaan vuokrauksesta ja asiakas täyttää avain- ja kulkukortin tilauslomakkeen. Työntekijä toimittaa lomakkeen kiinteistönomistajalle. Kun avain ja kulkukortti ovat työntekijällä, alivuokralainen noutaa avaimen ja kulkukortin toimistosta sekä allekirjoittaa vuokrasopimuksen. Avain ja kulkukortti palautetaan, kun vuokraustarve on päättynyt. Vuokraukset laskutetaan vuokrausajan päättymisen jälkeen. (Kuvio 1)



Kuvio 1. Rovaniemen steinerkasvatus ry:n tilojen alivuokrausprosessi

3.2 Esimerkki alivuokrauksesta

Ville Vuokraaja kysyy iltapäivätoiminnan tilaa vuokralle maanantaisin 9.1.–30.4.2021 klo 17.00–19.30. Työntekijä tarkastaa kalenterista, onko tila vapaana maanantaisin kyseiselle ajalle. Jos tila on vapaana, vuokrausprosessi etenee. Ville Vuokraaja täyttää kiinteistönomistajan avaintilauslomakkeen, jolla tilataan avain ja kulkukortti. Työntekijä toimittaa lomakkeen kiinteistön omistajalle. Kiinteistön omistaja koodaa kulkukortin tarvittavalle ajalle. Työntekijä noutaa avaimen ja kulkukortin kiinteistön omistajalta.

Työntekijä kirjaa asiakkaan sekä avain- että kulkukorttitiedot tiedot Exceliin. Työntekijä valmistelee vuokrasopimuksen, jonka hän printtaa allekirjoitettavaksi perinteisesti kynällä. Työntekijä ilmoittaa Vuokraajalle, että hän voi tulla noutamaan avaimen ja kulkukortin. Kun Ville Vuokraaja tulee noutamaan avaimet, he allekirjoittavat vuokrasopimuksen. Avain ja kulkukortti siirtyvät vuokraajalle. Avaimen ja kulkukortin luovutuspäivämäärä talletetaan Exceliin. Vuokrasopimus siirretään kansioon.

Jos vuokraukseen liittyy jotain erikseen sovittua, se löytyy esimerkiksi tulostetusta sähköpostiviestistä vuokrasopimuskansiosta. Eri alivuokralaisten kanssa voidaan sopia erilaisista asioista. Ville Vuokraaja haluaa säilyttää kaiutintaan eurytmiasalin hyllyllä. Se voi olla mahdollista, jos kaiutin ei haittaa muuta toimintaa. Myös laskutuskäytännöistä voidaan sopia joustavasti. Ville Vuokraajalla on muitakin vuokraustarpeita kevätlukukauden aikana. Vuokrat voidaan sopia laskutettavaksi koko kevätlukukauden päätteeksi, kaikki kevään vuokraukset samalla laskulla.

Kun vuokra-aika päättyy, Ville Vuokraaja palauttaa avaimen ja kulkukortin. Työntekijä merkkaa avaimen ja kulkukortin palautetuksi Exceliin. Työntekijä palauttaa avaimen ja kulkukortin kiinteistön omistajalle. Työntekijä toimittaa laskun Vuokraajalle. Vuokraaja maksaa laskun. Jos Vuokraaja vuokraisi tilaa uudelleen piankin, hän ei ehkä palauttaisi avainta eikä kulkukorttia. Tällöin työntekijän pitäisi huolehtia, että kortti koodataan uudelleen uutta tarvetta varten. Vuokrasopimus säilytetään jonkin aikaa. Laskutustilannetta seurataan laskutusjärjestelmässä.

4 RATKAISUJEN ETSINTÄ

4.1 Lähtökohdat ja taustaa ratkaisulle

Rovaniemen steinerkoulun toiminnanjohtaja Niina Tuisku tuntee organisaation alivuokraustoiminnan käytännöt. Hänen ajatuksiaan otetaan huomioon ratkaisun etsinnässä. Alivuokrauksiin liittyvät tiedot tulisi löytää selkeästi ja keskitetysti yhdestä paikasta. Avainkäytäntö on jäykkä, sitä olisi hyvä sujuvoittaa. Henkilökunnan olisi hyvä nähdä ajantasaisesti tilojen varaustilanne. Tulevaisuuteen yltävät visiot koskivat ovien koodilukkoja ja julkista tilojen varauskalenteria. (Tuisku 2020.)

Lähtökohta kehitystyössä on saada tilavuokraukset hoidettua sujuvasti ja tehokkaasti muun työn ohessa. Alivuokraustiedot tulee löytää yhdestä paikasta: asiakastiedot, avain- ja kulkukorttitiedot sekä vuokrasopimuksen tiedot. Tiloja on useita ja vuokrasuhteet erilaisia. Hinnat vaihtelevat käytön mukaan. Myös työntekijöiden avain- ja kulkukorttien luovuttamis- ja palauttamistietoja seurataan. Organisaatio ei omista kiinteistöä eikä kulunvalvontajärjestelmää. Ongelmia voi tulla, jos ei huomaa palauttamatta jäänyttä avainta tai alivuokralaisen kulkuoikeus on umpeutunut, vaikka kulkuoikeutta vielä tarvittaisiin.

Jos tilavuokrauksien hallintaan etsitään digitaalista ratkaisua, kannattaa ensin etsiä valmiita tuotteita. Google Kalenteri on valmis sovellus, johon voi merkata kellonaikojen mukaan tapahtumia. Kalenterinäkymässä on mahdollista seurata vaikka useita eri kalentereita yhtäaikaa. Kalenterinäkymää voi vaihdella esimerkiksi viikko- tai kuukausinäkymäksi. Sovellus on pilvipalvelu, jonka kalentereita on helppo jakaa. Kalenterin käyttöönottaja voi luoda useita eri kalentereita ja rajata niiden käyttäjät ja/tai katselijat. (Google.com 2020.) Googlen tämäkin palvelu toimii luotettavasti ja on helppo käyttää. Palvelu on rajatusti vain kalenteri-palvelu. Palvelu ei sisällä esimerkiksi asiakasrekisteriä.

4.2 Esimerkkejä tilavuokrausten hallintasovelluksista

Rapal Oy tarjoaa Optimaze-ohjelmistoa niin sisä- kuin ulosvuokrattujen tilojen ja kaiken niihin liittyvän tiedon hallinnointiin. Ohjelmistossa myös vuokralaiset pääsevät asiakastililleen muun muassa tarkastelemaan sopimuksensa tietoja. Ohjelmistoa tarjotaan esimerkiksi kunta-alalle. (Rapal.com 2020.) Ohjelmisto on laaja ja sisältää osioita pohjakuvista standardin mukaiseen vuokrasopimusraportointiin. Optimaze on kiinteistöalan yrityksille tarkoitettu maksullinen ja laaja palvelu.

Itech-Bros Oy tarjoaa Vuokratili-sovellusta ilmaiseksi tai edullisesti tarpeesta riippuen. Vuokratili-sovellus kokoaa vuokraustiedot ajantasaisesti yhteen paikkaan. Käyttöliittymän mainostetaan olevan selkeä ja helppokäyttöinen. Tietoja talletetaan vain kolmessa ikkunassa. Sovellus hoitaa toiminnanohjauksen, kirjanpidon, kannattavuuslaskennan ja kassavirtaseurannan. Se myös tulostaa valmiin veroraportin. (Vuokratili.fi 2021.) Ilmaisena palveluna se sisältää riittävät ja rajatut ominaisuudet vuokraustoimintaan.

Itech-Bros Oy:n Vuokratili herätti mielenkiinnon. Se vaikuttaa näppärältä ja helppokäyttöiseltä. Se on ominaisuuksiltaan rajattu ja selkeä. Rovaniemen steinerkasvatus ry ei yleishyödyllisenä yhteisönä maksa veroa varainhankintansa tuloksesta. Alivuokraustoiminnasta ei siten tehdä erikseen veroilmoitusta. Yhdistyksen vuokraustoiminta on myös hyvin pienimuotoista, eikä se ole organisaation ydintoimintaa. Rajattunakin sovelluksena siinä on kuitenkin paljon toimintoja, joita ei tällä hetkellä tarvita.

Rapal Oy:n Optimize on kiinteistöalan yrityksille tarkoitettu ja siten aivan liian laaja ohjelmisto yhdistyksen varainhankinnan vuokraustoiminnan tarpeisiin. Yhdistyksellä ei ole halua maksaa pienen alivuokraustoiminnan hallinnoinnista. Valmiiden sovellusten etsimisen jälkeen alkoi tuntua siltä, että valmisratkaisut eivät ratkaise ongelmaa yksinkertaisesti mutta tehokkaasti.

5 RATKAISU SOVELLUKSESTA

Yhdistyksen tulee hallinnoida tilavuokrauksensa hyvin; luotettavasti ja mielellään nopeasti. Työtehtävää ei hoideta päivittäin, vaan tarvittaessa. Siksi tehtävästä on hyvä saada nopeasti ajantasainen käsitys. Tarvittava tieto tulee löytyä keskitetysti, jotta prosessi saadaan käyntiin ja vietyä läpi sujuvasti. Keveä ja selkeä sovellus nopeuttaisi ja tehostaisi parhaiten oheistoimintana pyöritettävää tilavuokrausta.

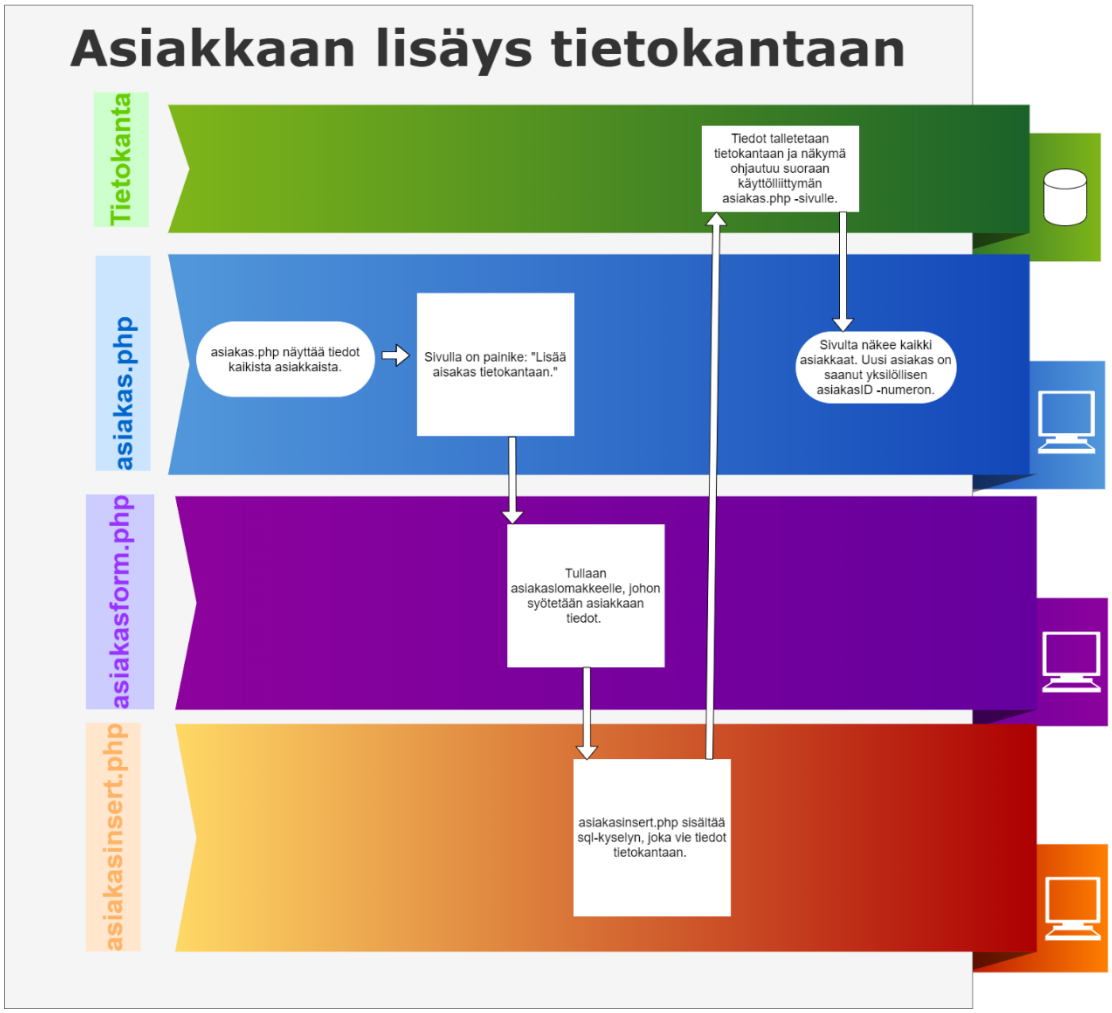
Räätälöity sovellus työntekijän omalla työasemalla olisi nopea käyttää. Se voisi olla tietokantapohjainen sovellus, jonka kautta olisi mahdollista hoitaa jokainen alivuokrausprosessin vaihe. Sovelluksen keskeinen tehtävä olisi keskittää tieto yhteen paikkaan. Tämä antaisi mahdollisuuden hoitaa alivuokraustoiminta tehokkaasti. Avainten ja kulkukorttien luovutusta olisi mahdollista seurata helposti. Vuokraustiedot laskutusta varten löytyisi yhdestä paikasta. Kalenterin tulisi olla helppo ja nopea käyttää. Kalenterista näkisi eri tilojen varaustilanteen nopeasti. Myös henkilökunnan tulisi tietää ajantasaisesti eri tilojen varaustilanne.

Tilavuokrauksiin liittyvät eri työvaiheet määrittävät sovelluksessa tarvittavat toiminnot. Suunnittelun edetessä nähdään tarvittavat ominaisuudet ja miten niitä olisi järkevä toteuttaa. Kun tehtävään alkaa perehtyä, huomaa, että tehtävä ei ole aivan yksinkertainen. Toimintoja on useita ja niiden tulee toimia tarvelähtöisesti.

Sovellus voi muodostaa henkilörekisterin, jos vuokraaja on yksityishenkilö. Vuokraaja voi olla pieni yhdistys, jolloin laskut pyydetään usein lähettämään yksityishenkilöiden nimellä heidän henkilökohtaiseen osoitteeseensa. Tällöin tulee huomioida henkilötietoihin ja tietoturvaan liittyvät kysymykset. Tietoja, ja siten henkilörekisteriä, ei säilytetä kauaa. Tiedot poistetaan, kun vuokraussuhde on päättynyt ja kaikki siihen liittyvät toimet on tehty.

Ratkaisuksi suunniteltiin sovellus, jonka taustalla toimii tietokanta. Tiedot talletetaan tietokantaan eri nimisiin tauluihin. Sovellusta käytetään selainpohjaisella käyttöliittymällä. Käyttöliittymän rakenne luodaan html-kielellä. Ulkonäköä varten luodaan css-tyylitiedosto. Käyttöliittymän kautta kuljetetaan tietoa tietokannan taulujen ja web-sivujen välillä. Tieto välittyy ohjelmointikieli php:n avulla.

Esimerkkinä tiedonkulusta tarkkaillaan uuden asiakkaan lisäämistä tietokantaan. Käynnistetään sovellus selaimessa. Asiakas-sivulla on painike, joka ohjaa lisäämään uuden asiakkaan tietokantaan. Painike vie asiakaslomakkeelle, johon syötetään asiakkaan tiedot. Kun tiedot talletetaan, tallennuspainike vie tiedostoon asiakasinsert.php. Tiedosto sisältää sql-kyselyn, joka vie php:n avulla asiakkaan tiedot tietokantaan. Tallennuksen jälkeen käyttäjä ohjataan suoraan takaisin asiakas-sivulle, josta näkee kaikki sen hetkiset asiakkaat. Juuri lisätty asiakas näkyy sivulla ja hän on saanut yksilöllisen asiakastunnisteen asiakasID:n. AsiakasID yksilöi asiakkaan ja sen avulla tietyn asiakkaan tietoja voidaan käsitellä prosessin aikana. (Kuvio 2.)



Kuvio 2. Asiakkaan lisäys tietokantaan

6 TOTEUTUKSESSA TARVITTAVAT TYÖKALUT

Tietokantasovelluksen tekemisessä tarvitaan useita työkaluja (taulukko 1). Useimmat työkaluista ovat omalle tietokoneelle ladattavia sovelluksia ja ohjelmointiympäristöjä. Työskentely aloitettiin tietokannan suunnittelusta, jonka ensimmäiset hahmottelut tehtiin kynällä paperille. Kun rakenne alkoi hahmottua riittävästi, siirryttiin kaaviosovellukseen. Tässä käytettiin apuna Draw.io -sovellusta, joka tarjoaa valmiita muotoja ja kuvioita erilaisten kaavioiden piirtämiseen. Sovellus tarjoaa mm. valmiit taulupohjat tietokannan suunnittelua varten sekä nuolet tietokannan relaatioiden ilmaisemiseen.

Myös käyttöliittymän ulkonäkö kannattaa suunnitella ensin piirtäen paperille tai grafiikkaohjelmalla. Käyttöliittymää hahmoteltiin karkeasti paperille. Sen ulkoasu ja siten käyttöliittymä haluttiin mahdollisimman selkeäksi ja yksinkertaiseksi. Sivun yläosassa ensimmäisenä on otsikko. Vaakatasossa oleva selkeä valikko asetetaan myös sivun yläosaan. Sivun alaosassa sijaitsee footer. Kukin sisältö aukeaa index.php:n sisältöosaan sivun keskelle. Kun sovelluksen rakenne, ulkonäkö ja tarvittavat toiminnot olivat selvillä, aloitettiin käyttöliittymän rakenteen koodaus html:llä. Tilavuokrasovelluksen sivut koodattiin koodieditorilla Brackets.

Tietokanta luotiin MySQL-ohjelmistolla paikallisella Apache-palvelimella. Tietokantaa testattiin SQL-kyselyillä. PHP välittää tiedonsiirron sovelluksen ja tietokannan välillä.

Taulukko 1. Työskentelyssä käytetyt työkalut

Brackets	koodieditori
Draw.io	monipuolinen kaaviosovellus vuokaavioiden toteuttamiseen
Xampp	serveri, joka sisältää Apache -web palvelimen sekä MySQL-tietokantapalvelimen.
Apache	paikallinen web-palvelin, sisältää php-ohjelmointiympäristön
MySQL	ohjelmisto relaatiotietokantojen luomiseen, ylläpitämiseen ja käyttämiseen
SQL	kyselykieli, joka välittää tietoa tietokannasta tai tietokantaan
PHP	palvelinohjelmointikieli

7 TIETOKANNAN SUUNNITTELU JA KÄSITEANALYYSI

7.1 Tietokanta

Tietokanta on tiedon säilyttämiseen käytettävä työkalu. Kun tietoa kertyy paljon, tietokannassa tieto pysyy järjestyksessä ja sitä on helpompi ymmärtää ja käyttää. Tietokantoihin talletetaan esimerkiksi henkilötietoja tai tietoa erilaisista tuotteista. Tietokannan toteutus aloitetaan kartoittamalla tarve: miksi ja millaista tietoa varten tietokanta tarvitaan? (Microsoft.com 2020.)

Relaatiotietokannassa tieto talletetaan tauluihin. Taulut yhdistetään toisiinsa loogisesti erilaisilla yhteyksillä (relaatioilla) sen mukaan, miten tietoa eri tauluista poimitaan. Yhteyksien luonnissa otetaan huomioon, onko tieto pakollinen vai ehdollinen. Puhutaan tiedon ”isä-lapsisuhteista”. Esimerkiksi asiakasnumero on pakollinen isä-tieto, jolla voi olla usea ehdollinen lapsi-tieto (esimerkiksi vuokrasuhteissa: sali, käsityöluokka, kotitalousluokka). Kun relaatiot on suunniteltu toimivasti, tietokannasta on mahdollista etsiä sujuvasti tietoa eri tauluista tai manipuloida tietoja. (Hovi, Huotari & Lahdenmäki 2005, 8–9.)

Tietokantaan talletetaan vain perustietoa. Esimerkiksi veron määrä voidaan laskea perustiedoista: hinta ja veroprosentti. Veron määrää on siis turha tallentaa tietokantaan. (Koskenniemi 2020.) Käsiteltävän tiedon laadun ymmärtäminen ja prosessin huomioiminen auttavat suunnittelemaan tietokantaa. Tiedontarpeen pohjalta suunnitellaan tietokannassa tarvittavat taulut ja niiden suhteet. Taulut ja niiden relaatiot muodostavat tietokannan rakenteen.

7.2 Viite-eheyssäännöt

Viite-eheyssäännöillä pyritään pitämään tietokanta ”siistinä”. Viite-eheyssäännön määrittelyllä pyritään varmistamaan se, että tietokantaan ei jää ”seisomaan” tietoa, joka jäisi tulevaisuudessa käyttämättä ja olisi talletettuna tietokannassa turhaan. Viite-eheyssäännöt sujuvoittavat myös ohjelmointia. (Hovi, Huotari & Lahdenmäki 2005, 22.)

Viite-eheyssäännöt ovat:

- R (restrict), jolloin isä-tietoa ei voi poistaa, jos lapsi-tietoa löytyy. Määritys estää yksinäisen tiedon jäämisen tietokantaan.
- C (cascade), jolloin isä-tiedon poistaminen poistaa myös kaikki lapsi-tiedot.
- N (se null), jolloin isä-tiedo poistaminen muuttaa lapsi-tietojen viiteavaimen null-arvoksi.
- D (set default), jolloin viiteavain palautuu määritettyyn oletusarvoon. (Koskenniemi 2020.)

7.3 Käsitemalli ja normalisointi

Tietokannan suunnittelun alkaessa käsiteanalyysi etenee yleiseltä tasolta käsitemallin luomiseen. Kun tiedetään, minkälaista tietoa tarvitaan, luodaan tietokannasta käsitemalli. Käsitemallissa tietokanta kuvataan graafisesti. Suunnitellaan taulut ja mitä tietoa kukin taulu tulee säilyttämään. Taulut yhdistetään toisiinsa loogisesti ja tarpeen mukaan ER-kaaviossa. ER-kaavio kannattaa suunnitella mahdollisimman selkeäksi, jotta kokonaisuudesta saataisiin selkeä kuva. (Koskenniemi 2020.)

Relaatiotietokannassa tieto (data) talletetaan erillisiin tauluihin. Tauluista ja niiden tietueista laaditaan DataDictionary tietokannan luonnin ja sen kehittämisen tueksi. DataDictionaryssa nimetään taulut ja niiden käsitteet (tietueet). Jokaisessa taulussa on oltava perusavain, tiedon yksilöivä uniikki käsite. Muita käsitteitä voivat olla esimerkiksi katuosoite, postinumero tai puhelinnumero. DataDictionary selittää käsitteitä tarkemmin ja ohjaa tietokannan luomisessa. Se kertoo mm. mikä käsite on perusavain-kenttä ja mikä käsite on tietokannassa pakollista tietoa. Siinä määritellään myös se, paljonko kullekin tiedolle varataan tilaa tietokannassa. (Meador 2018.)

Tiedot järjestetään tauluihin siten, ettei synny päällekkäistä tietoa. Yksi tieto on tauluissa vain kerran. Esimerkiksi ”tila”-taulussa jokainen tila on nimetty vain ker-

ran. Tiloja ei nimetä muihin taulukoihin. Tätä kutsutaan normalisoinniksi. (Microsoft.com 2020.) Tietokannan normalisointi takaa sen, että tietokannasta haettava tieto tulee juuri oikeasta paikasta (oikeasta taulusta) ja on siten oikeaa, haluttua dataa.

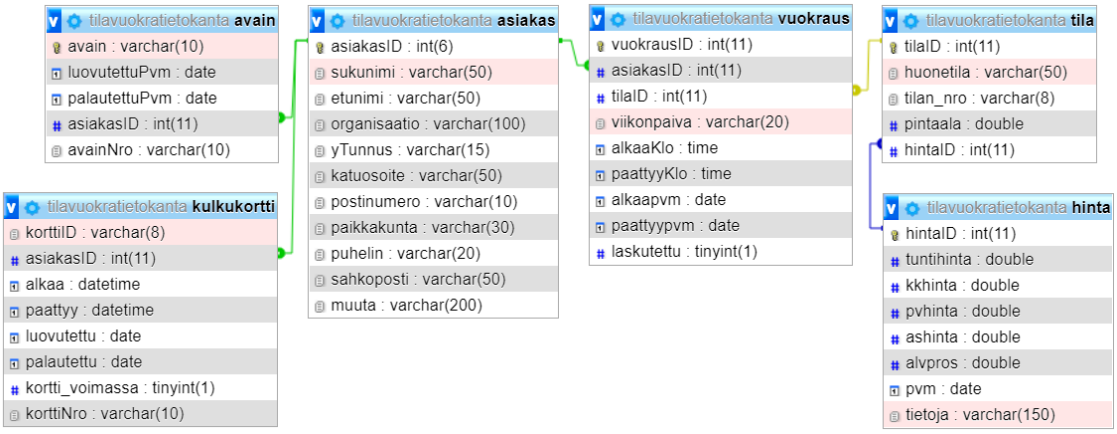
8 SOVELLUKSEN TIETOKANTA

8.1 Tietokannan rakenne ja toteutus

Sovelluskehitys aloitettiin tietokannan suunnittelusta. Prosessin pohjalta alettiin miettiä, millaista tietoa sovellus tulee käsittelemään. Mitä tietoa prosessissa tarvitaan? Miten tieto tulee liikkumaan? Mitä tietoa etsitään prosessin eri vaiheissa? Mistä tietoa etsitään? Mikä tieto liittyy mihinkin toimintoon?

Suunnittelussa eteenpäin auttoi käsitys erilaisista vuokrauksista. Yhdellä vuokralaisella voi olla usea alivuokrasuhde. Vuokrattavana on usea erilainen tila. Alivuokralaisen kanssa voidaan sopia yksilöllisesti vuokraukseen liittyvistä asioista. Myös vuokralaiset ovat yksilöllisiä. Vuokralainen voi olla yritys tai yksityishenkilö. Pienten yhdistysten laskutus voi kulkea yksityishenkilöiden kautta. Vuokralainen tarvitsee avaimen ja mahdollisesti myös kulkukortin. Tarpeen ratkaisee se, mihin vuorokauden aikaan tila on vuokralaisen käytössä.

Kun tietokantaan talletettavan ja sen kautta käsiteltävän tiedon laatu oli kartoitettu, suunniteltiin relaatiotietokannan taulut. Taulut otsikoitiin ja kuhunkin tauluun luotiin kentät tarvittavalle datalle. Tarvitaan esimerkiksi taulu ”asiakas”, jonka alle kerätään tarvittavat tiedot asiakkaasta. Näitä ovat esimerkiksi etunimi, sukunimi, organisaatio, puhelinnumero. Alla oleva ER-kaavio luotiin Draw.io-sovelluksella. Kuvasta 1 nähdään, mitä tietoja alivuokrattavien tilojen hallinnoinnissa tarvitaan.



Kuva 1. Tilavuokrasovelluksen tietokanta

Asiakkaat vuokraavat jonkin tilan tietylle päivälle tiettynä kellonaikana. Ajanjaksot vaihtelevat ja samalle päivälle voi olla usea vuokralainen. Tarvitaan taulu ”tila” tiloja varten. Koska vuokraukset laskutetaan yksitellen, tarvitaan tieto kustakin vuokrauksesta erikseen. Taulu ”vuokraus” kokoaa vuokratiedot kustakin vuokrauksesta. Erilaisille hinnoille tarvitaan taulu ”hinta”. Asiakkaille luovutetaan avain ja/tai kulkukortti. Taulujen primarykey-kentät ovat kutakin rivitietoa yksilöivät autoincrement-kentät, joita käytetään tiedon välittämisessä tietokantaan tai tietokannasta. Avain- ja kulkukortti-taulussa on myös toinen numerokenttä (avainNro ja korttiNro). Ne ovat kiinteistön omistajan numeroimia avaimia ja kulkukortteja. Kentästä näkyy mikä avain tai kulkukortti käyttäjälle on luovutettu.

Tietokannasta tehtiin DataDictionary, jossa on dokumentoituna jokaisen taulun tiedot. Käsitteet mietittiin tarkoin ja tietokanta suunniteltiin selkeäksi. Taulujen ominaisuudet luotiin käyttötarvetta arvioiden. DataDictionarysta näkee selkeästi eri tiedon datatyypin, sille varattavan tilan, mahdollisen oletusarvon, onko tieto pakollinen ja onko tieto yksilöllinen. Suunnittelussa oli hyvä pohtia mm. sitä, mikä tieto kannattaa olla pakollinen tieto. Pakollista tietoa ei haluttu liikaa, jotta sovelluksen käyttö ja tiedon tallentaminen ei vaikeutuisi. DataDictionary on hyvä työkalu mahdolliseen tietokannan kehittämiseen tulevaisuudessa. (Liite 1)

8.2 Tietokannan testaus

Testattaessa tietokannan toimivuutta erilaisilla SQL-kyselyillä, huomattiin, että kulkukortti- ja vuokraustaulussa oli samannimiset kentät. Kenttien nimet päivitettiin erilaisiksi, jolloin kyselyiden toimivuuteen voi luottaa. Suoritettiin siis normalisointia. Muutosten jälkeen tietokannan toimivuutta testattiin uudelleen. Tauluihin vietiin tietoa kyselyillä sekä haettiin tietoa useasta taulusta yhtäaikaa yhdellä kyselylauseella (Kuvat 2 ja 3).

```
INSERT INTO asiakas (asiakasID, sukunimi, etunimi, organisaatio)
VALUES ('NULL', ’Laulaja’, ’Laura’, 'Laulu ry');
```

Kuva 2. SQL-kysely, joka vie asiakastietoa tietokantaan

Tietokantaa testattiin myös hakemalla tietoa useasta eri taulusta. Kuvan 3 SQL-kysely tulostaa tiedon, kenen kulkuoikeus on päättynyt ja mihin saakka huonetila

on hänelle vuokrattu. Tällainen kysely voi olla hyvä vaihtoehto seurata kulkuoikeuksien päättymistä. Jos tilatarve jatkuu kulkuoikeuden päättymisen jälkeen, kulkuoikeutta pitää koodata lisää.

```
SELECT etunimi, sukunimi, huonetila, paattyypvm, korttiID, paattyy
FROM asiakas
INNER JOIN vuokraus ON asiakas.asiakasID=vuokraus.asiakasID
INNER JOIN tila ON vuokraus.tilaID=tila.tilaID
INNER JOIN kulkukortti ON asiakas.asiakasID=kulkukortti.asiakasID
WHERE paattyy < CURRENT DATE;
```

Kuva 3. Kysely hakee tietoa kolmesta taulusta.

Kyselyitä tehtiin, jotta nähtiin, että tietokanta toimii oikein. Sovellukseen tulee asiakaslomake, jolla syötetään uuden asiakkaan tiedot tietokantaan. Sovellus tulostaa asiakkaat erilliseen taulukkoon, joka noutaa tiedot tietokannasta. Taulukossa on mahdollista päivittää tietoja tietokantaan. Jos esimerkiksi asiakkaan osoite muuttuu, sen voi kirjoittaa suoraan taulukkoon. Taulukon painike vie tiedot tietokantaan.

Asiakas-taulukosta voi myös poistaa asiakkaan ja kaikki asiakasta koskevat vuokraustiedot. On tärkeää, että sql-lauseet toimivat luotettavasti. Tietokannassa käytetään viite-eheyssääntöä cascade, jolloin on mahdollista poistaa kaikki asiakasta koskevat tiedot tietokannasta sen jälkeen, kun tietoja ei enää tarvita. Cascade-määrittelyn käyttö on perusteltua, koska tarvittaessa vanha tieto löytyy muualtakin. Vuokraustiedot löytyvät vuokrasopimuksesta sekä usein sähköpostista. Avain- ja kulkukorttitiedot löytyvät allekirjoitetuista avainlomakkeista. Myyntireskontra seuraa maksutilannetta. Sovelluksen tarkoitus on olla työtehtävää helpottava ja selkeyttävä työkalu. Kun asiakas on palauttanut avaimen ja kulkukortin ja vuokralasku on maksettu, tietoja ei välttämättä enää tarvitse säilyttää. Asiakkaan tiedot poistetaan tietokannasta, kun vuokrausprosessi on hoidettu loppuun.

9 SOVELLUKSEN KÄYTTÖLIITTYMÄ

9.1 Käyttöliittymä, toiminnallisuus ja käytettävyys

Käyttöliittymä on se osa sovellusta, joka näkyy käyttäjälle ja millä käyttäjä käyttää sovellusta. Käyttöliittymä välittää tiedon tietokannan ja käyttäjän välillä. Merkkipohjainen käyttöliittymä on pelkkää tekstiä. Käyttäjä kirjoittaa komennot komentoriville. Graafinen käyttöliittymä sisältää kuvakkeita ja painikkeita. Käyttäjä käyttää sovellusta niitä klikkailemalla. (Helsingin yliopisto 2021.) Tilavuokrasovellukseen suunniteltiin graafinen käyttöliittymä.

Käyttöliittymän toteutus aloitettiin kartoittamalla tarve: mitä tehtäviä sovellus tulee hoitamaan? Sovellus kannattaa suunnitella modulaariseksi. Se tarkoittaa sovelluksen jakamista eri komponentteihin, jolloin selkeän rakenteensa ansioista käytettäväksi voidaan valita vain haluttu osa järjestelmästä. Jos esimerkiksi vain kalenterivaraukset haluttaisiin näyttää henkilökunnalle, se olisi helpompi toteuttaa.

Responsiivisuus tarkoittaa sitä, että nettisivu muotoutuu käyttäjän päätelaitteella käyttäjäystävälliseen muotoon. Sivun tulee näkyä sekä toimia niin puhelimella kuin isommalla ruudulla moitteettomasti. Nykyään ihmiset käyttävät sivustoja ja erilaisia verkkopalveluita enimmäkseen mobiililaitteilla. (eLuotsi.fi 2020.) Kehiteltävän sovelluksen ei tarvitsisi olla responsiivinen, koska sitä tullaan käyttämään kannettavalla tietokoneella. Jos sovellus kuitenkin tulevaisuudessa sisältäisi moduuleita laajempaan käyttöön, sen olisi hyvä olla jo valmiiksi responsiivinen. HTML5 ymmärtää skaalata sivun eri näytöille, kun metatiedot sisältävät viewport-määrittelyt (Kuva 4). (w3schools 2020f.)

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" shrink-to-fit="no">
```

Kuva 4. HTML5:lle määritetty viewport

Responsiivisuus, joka huomioi eri tyylit erikokoisille näytöille, määritetään CSS:n @media-tekniikalla. Tekniikalla määritetään pysäytyspiste, jonka jälkeen sivu vaihtaa tyyliä. Määrittely mahdollistaa nimenomaan Mobile First-suunnittelun,

mikä tarkoittaa sitä, että sovellus suunnitellaan lähtökohtaisesti ensin mobiililaitteilla käytettäväksi. Sovellus muuttaa tyyliä, kun sitä käytetään isommilla näytöillä. Tyylien responsiivisuus toteutettiin tallentamalla mediamäärittely css-tyylitiedostoon (Kuva 5). (w3schools 2021e.)

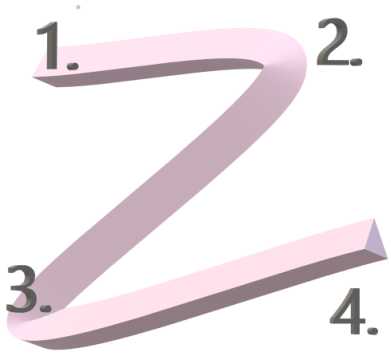
```
@media(min-width:468px)
```

Kuva 5. Tyylitiedostoon talletettu mediamäärittely.

Suunnittelussa huomioidaan käytettävyys ja selkeys. Hahmottelussa keskityttiin toimintojen sijaintiin. Esimerkiksi navigaatiopalkki haluttiin sivun yläosaan, koska sinä sen koettiin toteuttavan nimenomaan selkeyttä ja Z-mallia. Z-mallipohjaan perustuvassa sovelluksessa käyttäjän on helppo suorittaa eri toimintoja, koska ne on sijoitettu käyttäjäkokemusten perusteella tutuille paikoille (Banc & Cao 2014, 37–38).

Bankin ja Caon (2014, 37–38) tutkimuksen mukaan ihmissilmä skannaa sivua tähän kuviossa 3 esitetyn mallin mukaisesti:

- 1. piste ylhäällä vasemmalla on paras kohta organisaation logolle
- 2. piste ylhäällä oikealla, tässä olisi hyvä kohta jollekin toiminnolle, esimerkiksi kirjaudu-painikkeelle
- Sivun keskiosaan latautuu keskeinen sisältö
- 3. piste alhaalla vasemmalla voisi ohjata viimeiseen toimintoon viimeiseen z-pisteeseen
- 4. piste on luonteva sijainti viimeiselle toimintoon kutsulle, esim. lähetä lomake -painikkeelle. (Kuvio 3)



Kuvio 3. Z-malli sivun (tai sovelluksen) suunnittelussa

Käyttöliittymä päätettiin rakentaa alusta saakka itse, jotta se sisältäisi vain tarvittavat ominaisuudet. Sovelluksen kehitystä on mahdollista tulevaisuudessa tarpeen tullen jatkaa. Käyttöliittymän rakenne kirjoitettiin html:llä. Koska kyseessä on tietokantasovellus, tiedostot talletettiin php-päätteellä. Sovelluksen sisältö näytetään Index.php-sivulla. Rakenne koodattiin index.php-sivulle, jonka sisältöosassa esitetään muilta sivuilta tuleva tieto.

Käyttöliittymän rakenne toteutettiin grids-tekniikalla. Sivu jaettiin 12:een columniin, mikä toimii laskennallisesti parhaiten eri päätelaitteissa. Koko sivu rakentuu luokan ”container” sisään, joka sisältää kaikki 12 columnia. Sivu jaettiin kuuteen gridsriviin, joista kolmas on automaattinen rivi. Keskeisin sisältö latautuu automaattiselle riville, jolloin se ottaa automaattisesti kaiken tarvitsemansa tilan. Eri sivujen sisältö käyttää eri verran tilaa.

Muotoilut ja tyylit talletettiin erillisenä tyylitiedostona css-kansioon, tiedoston päätteenä on css. Tiedosto määrittää tyylit luokkina (class) eri kohteille. Oma luokka tyylitiedostossa määritettiin mm. headerille, otsikolle, navigaatiopalkille ja footerille. Jokainen luokka sisältää grids-määrittelyn, jonka mukaan se asemoituu sivulle.

9.2 Tiedon siirtäminen käyttöliittymän ja tietokannan välillä

PHP on ohjelmointikieli, jota käytetään palvelinympäristössä interaktiivisten web-sivujen tekemiseen (w3schools 2021d). Tilavuokrasovelluksen käyttöliittymä saa yhteyden tietokantaan php:n avulla. PHP tuo tietokantaan siirrettyä tietoa käyttöliittymän sivuille. Myös navigaatiopalkin koodaamisessa on käytetty php-kieltä.

Sovelluksessa on eri lomakkeita, joille syötetty tieto siirtyy PHP-skriptin avulla tietokantaan. Tieto syötetään käyttöliittymän html-lomakkeelle, josta tieto siirtyy tietokantaan html:n submit-metodilla. Esimerkiksi asiakastietolomakkeelle on koodattu html-painike, joka siirtää tiedon asiakasinsert.php -tiedostoon. Asiakasinsert.php sisältää sql-koodin, joka vie tiedot tietokantaan php:n välittämänä. (Liite 2)

Tiedonsiirto käyttöliittymän ja tietokannan välillä tapahtuu PHP-ohjelmointikielen POST-metodilla. Sensitiivinen tieto kannattaa siirtää aina POST-metodilla, koska silloin tieto siirtyy näkymättömänä muille sulautettuna html-komentoihin ja siten tietoturvallisesti. Siirrettävän tiedon määrää ei myöskään ole POST-metodissa rajoitettu. (w3schools 2021c.) Tietokantayhteys katkaistaan automaattisesti aina scriptin jälkeen. Tämä on otettu huomioon jokaisessa scriptissä, joka vie tietoa tietokantaan. (Liite 3)

Avain-, asiakas- ja vuokraus-sivuilla on taulukot, jotka kokoavat ajantasaiset tiedot avaimien ja kulkukorttien luovutuksista, asiakastiedoista sekä vuokraustiedoista. Tiedot tulostuvat tietokannasta käyttöliittymän sivuille tulostamalla rivimuuttujia niin kauan, kuin pyydettyjä tietoja tietokannasta tulee. Tiedot tulostuvat kyseisten sivujen taulukoihin. (Liite 4)

Tietojen päivitys tietokantaan toteutetaan taulukoilla, joihin pääsee Avain- ja Asiakas- ja Vuokraukset-sivujen taulukoitten Update-painikkeella. Painike johdattaa toiselle sivulle taulukkoon, joka sisältää olemassa olevat tiedot input-kentissä. Input-kenttiin voi suoraan syöttää uuden tiedon. Päivitystaulukon Päivitä-painike vie uuden tiedon tietokantaan.

10 RATKAISU KALENTERISTA

Yhdistys otti käyttöön G Suite for Education -ympäristön koronakeväänä 2020. Ympäristössä on kalenterisovellus, joka on koko työyhteisön käytettävissä. Tilavuokrauksessa tarvittavat kalenterit päätettiin sijoittaa G Suite -ympäristöön, jossa varaustilanne on helposti koko henkilökunnan nähtävissä. Samalla toteutuu tehokas tiedonjako ja vältetään päällekkäisvaraukset.

Googlen kalenterisovellus on joustava. Kalenterinäkymässä voi näkyä yhtä aikaa usea tai vain yksi tila. Näkymää voi tarkkailla viikko- tai kuukausinäkymänä ja tapahtumia on helppo lisätä ja muokata. Käyttömukavuus tulee helppokäyttöisyydestä ja selkeydestä. Kalenterit ja tapahtumat voivat olla eri värisiä, mikä helpottaa tiedon lukemista. Googlen kalenterisovelluksessa voi ottaa käyttöön riittävän paljon kalentereita eri tiloille. (Google.com 2020.)

Erilaiset käyttöajat on helppo syöttää Google-kalenterisovellukseen. Yhdistyksen asiakkaat voivat varata tiloja tunneittain, jolloin samalle päivälle voi olla monta käyttäjää. Asiakkaat voivat vuokrata tiloja myös säännöllisesti viikoittain tiettyyn aikaan tai satunnaisesti, esimerkiksi viikonloppuisin. Koko henkilökunnalla on oikeus myös tehdä varauksia jokaisen tilan kalenteriin, joten ajantasaiset tilojen käyttötiedot ovat kaikkien löydettävissä yhdessä sijainnissa. Kun kalenterisovelluksesta löytyy asiakkaalle sopiva vapaa tila, käännytään tilavuokrasovelluksen puoleen.

11 SUUNNITELTU TILAVUOKRASOVELLUS

11.1 Sovelluksen toimintaperiaate

Sovellus on suunniteltu hallinnoimaan alivuokrauksiin liittyvää tietoa. Käyttöliittymän asiakaslomakkeelle syötetään asiakkaan tiedot sekä luovutetun avaimen ja kulkukortin tiedot. Lomake vie tiedot tietokantaan. Asiakastiedot tulostuvat käyttöliittymän Asiakkaat-sivulle. Sivun taulukosta voi päivittää asiakkaan tietoja, jos esimerkiksi asiakkaan osoite muuttuu.

Avain- ja kulkukorttitiedot koettiin ehkä tärkeimmäksi seurantakohteeksi ongelman määritysvaiheen alussa. Tietoja haluttiin seurata selkeästi ja helposti. Ne tulostuvat käyttöliittymässä omalle Avaimet-sivulle omaan taulukkoonsa. Sivulta näkee helposti esimerkiksi sen, milloin kulkukortin kulkuoikeus on päättynyt tai onko avain palautunut. Sivun taulukon kautta voi päivittää avaimiin ja kulkukortteihin liittyvää tietoa tietokantaan, esimerkiksi avaimen palautuspäivän.

Sovelluksessa on vuokrasopimuslomake, jonka voi täyttää ja tulostaa allekirjoitettavaksi. Vuokrasopimuslomakkeen voisi koodata siten, että kun siihen syötetään asiakasnumero, asiakkaan tiedot siirtyvät lomakkeelle automaattisesti tietokannasta. Uudella asiakkaalla ei ole asiakasnumeroa, vaan tietokanta luo tällöin asiakasnumeron automaattisesti. Loput vuokraukseen liittyvät tiedot syötetään lomakkeen kenttiin. Vuokrauslomakkeelta vuokraustiedot siirtyvät tietokantaan ja tulostuvat tietokannasta Vuokraukset-sivulle. Kun vuokralainen noutaa avaimen ja kulkukortin, hän samalla allekirjoittaa tulostetun vuokrauslomakkeen.

Kun vuokra on maksettu sekä avain ja kulkukortti palautettu, vuokraukseen liittyviä tietoja ei enää tarvita. Kaikki vuokraukseen liittyvät tiedot voidaan poistaa tietokannasta Vuokraukset-sivun taulusta. Kaikki asiakkaan tiedot voidaan poistaa Asiakas-sivun taulukosta. Sovellus on säilyttänyt tiedot yhdessä paikassa ja auttanut alivuokrasuhteen seurannassa koko prosessin ajan.

11.2 Sovelluksen esittely

Käyttöliittymän etusivu esittelee käyttötarkoituksen ja eri sivujen sisällön. Sovelluksen asettelua voi halutessaan helposti muuttaa, koska rakenne on koodattu

index.php-sivulle, jolle muut sivut tulostuvat. Tämän yhden sivun käsittelyllä on mahdollista muuttaa käyttöliittymän ulkoasua. Sovelluksen etusivu esittelee sovelluksen (Kuva 6).

Kuva 6. Sovelluksen etusivu

Tilat-sivu kokoaa faktatietoa tiloista. Se nopeuttaa ja helpottaa tiedon antamista asiakkaalle eikä työntekijän tarvitse muistaa ulkoa tilatietoja. Eri tiloja voi koskea mm. erilaiset käyttöohjeet, joten tieto on hyvä olla helposti saatavilla. Kaikkien tilojen ulosvuokrausta rajaa yhdistyksen oma käyttö. (Kuva 7)

Kuva 7. Sovelluksen Tilat-sivu

Eri tilojen kalenterit haluttiin löytyvän sovelluksesta samalla periaatteella kuin muukin tieto. Navigaatiopalkista löytyy painike Kalenterit -sivulle. Kalenterit-sivu

ohjaa ainoastaan kirjautumaan yhdistyksen Google-ympäristöön, jossa eri tilojen kalenterit ovat koko henkilöstön käytettävissä.

Avaimet-sivulla on taulukko, joka sisältää tiedot käyttäjille luovutetuista avaimista ja kulkukorteista. Taulusta näkee myös kulkukortin voimassaoloajan, jolloin tietoa on helppo seurata. Jos kulkuoikeus on päättynyt, mutta kulkukorttia ei ole palautettu, päivämäärä näkyy hälyttävän punaisena. Kaikki sovelluksen taulukot saavat hissit skaalautuessaan pienempään näyttöön, mikä tekee tekevät taulukoista lukukelpoiset myös pienemmillä päätelaitteilla (Kuva 8).

Kuva 8. Avain-sivun taulukko pienemällä näytöllä

Asiakkaat-sivulla on taulukko, joka tulostaa tiedot kaikista asiakkaista. Taulukko sisältää sarakkeen ”Muuta”, johon voi kirjata esimerkiksi asiakkaan kanssa yksilöllisesti sovittuja asioita. Vain asiakastaulukosta voi poistaa kaikki asiakkaaseen liittyvät tiedot tietokannasta. (Kuva 9)

Kuva 9. Sovelluksen Asiakas-sivu

Vuokraukset-sivulla näkyy taulukossa kaikki vuokraustiedot. Sivulta saa yleiskuvan vuokrauksista ja tiedot laskutusta varten. Sivulta voi tarkistaa, kuka vuokralainen on käyttänyt tiettyä tilaa tiettynä aikana. Taulukosta voi päivittää vuokraustietoja, jos niihin tulee muutos. Sivulta voi poistaa yksittäisen vuokrauksen tiedot, kun vuokraustietoa ei enää tarvita. Vuokraukset sivu myös ohjaa uuden vuokrasopimuksen tallentamiseen.

Avain-, Asiakas- ja Vuokraukset-sivujen taulukoista pääsee päivittämään taulukoitten tietoja. Taulukoitten Update-painike vie taulukkoon, joka sisältää inputkentät. Näihin kenttiin voi kirjoittaa päivitettävän tiedon. Esimerkiksi avaimen ja kulkukortin palautuspäivämäärä päivitetään tätä kautta. Päivitä-painike vie päivitetyt tiedot tietokantaan. (kuva 10)

Kuva 10. Taulukko, johon voi päivittää asiakastietoja.

Sovellus on lähtökohtaisesti vain yhden työntekijän henkilökohtainen työkalu. Sitä ei ole tarkoitus ottaa laajempaan käyttöön esimerkiksi verkkoasemalle. Kirjautuminen sovellukseen olisi kuitenkin hyvä olla olemassa.

11.3 Sovelluksen testaus ja säätö

Tiedon välitys tietokannan ja sovelluksen välillä tapahtuu php-kielellä. Yhteys tietokantaan luotiin php-kielellä. Yhteysmuuttuja ottaa yhteyden tietokantaan ”mysqli_connect”-komennolla. Jos yhteys onnistuu, sovellus tulostaa ”Tietokantayhteys on olemassa”. Muuten sovellus tulostaa ”Tietokantayhteyttä ei ole”. Positiivinen kuittaus kommentoitiin sen jälkeen, kun yhteys oli toimiva.

Kun yhteys tietokantaan on toimiva, php-koodi ohjaa valitsemaan tietokannan ”mysqli_select_db”-komennolla. Jos tietokannan valinta ei onnistunut, sovellus tulosti ”Tietokantaa ei valittu”. Yhteys saatiin nopeasti toimimaan ja tietokanta säilytettiin samassa sijainnissa kehitystyön ajan. Yhteys toimi siten hyvin koko ohjelmoinnin ajan.

Käyttöliittymän sivut tulostuvat index.php-sivun sisältöosaan. Se, miten tieto asettui sivulle, ohjasi korjaamaan koodia. Esimerkiksi gridien määrittelyä muutettiin tyylitiedostossa tässä vaiheessa 12 columniin.

Sovelluksen skaalautuvuutta testattiin, kun käyttöliittymässä toimi muutama sivu. Responsiivisuus-määrittelyt html:ssä ja css-tyylitiedostossa tekivät käyttöliittymästä skaalautuvan selaimessa. Navigaatiopalkki ja footeri skaalautuivat. Navigaatiopalkin otsikot asettuivat allekkain. Samoin sivun sisältöosassa sijaitsevat tekstit asettuvat allekkain. Asiakas- ja avaintaulusivut käyttäytyivät eri lailla kuin muut sivut eivätkä taulukot skaalautuneet. Taulukkosivujen skaalautuvuus pienempään näyttöön vaati taulukolle oman div-määrittelyn html:ssä (Kuva 11). Tämä ratkaisu asetti taulukoihin hissit, jolloin niiden lukeminen pienemmässä näytössä on miellyttävämpää.

```
<div style=”overflow-x:auto;”>Taulukko tähän</div>
```

Kuva 11. Koodi responsiiviselle taulukolle

Sovelluksen mobiililaitesoveltuvuus voidaan testata esimerkiksi googlen Mobile-friendly työkalulla (Google.com 2021). Sovellusta tullaan käyttämään ainoastaan laptopilla, mutta se on suunniteltu responsiiviseksi, mikä voi olla hyvä tulevaisuudessa.

Sisällöllisesti tuli ottaa huomioon, miten tieto olisi loogisesti löydettävissä. Avain- ja asiakastiedot viedään tietokantaan asiakaslomakkeella. Lomake tulee sisältämään siis asiakastietojen lisäksi avain- ja kulkukorttitiedot. Vuokraustiedot viedään tietokantaan vuokrasopimuslomakkeella, koska yhdellä asiakkaalla voi olla usea eri vuokrauskohde samanaikaisesti. Vuokrasopimuslomakkeelle syötetään asiakasnumero, jonka perusteella asiakkaan tiedot siirtyvät lomakkeelle. Vuokralomake antaa vuokraukselle oman tunnisteen, jonka perusteella vuokrauksen tiedot voidaan poistaa tietokannasta.

Tiedot tulostuvat tietokannasta eri sivuille: asiakastiedot asiakkaat-sivulle, avain- ja kulkukorttitiedot avaimet-sivulle ja vuokratiedot vuokraukset-sivulle. Tietojen tulostuminen omille sivuilleen tuo selkeyttä sovellukseen ja tiedon etsintään. Tietojen päivitys tulisi voida tehdä nopeasti. Asiakas-, avain- ja vuokraustiedot tulostuvat taulukoihin, joista tietoja voidaan päivittää tietokantaan. Kaikki asiakkaaseen liittyvät vuokraustiedot voidaan poistaa vain asiakkaat-taulusta. Vuokraussivulta voidaan poistaa yksittäisen tietyn vuokrauksen tiedot tietokannasta.

Alussa oli ajateltu siten, että kun vuokrauksesta sovitaan, asiakastietolomakkeen kautta syötetään vuokraustiedotkin tietokantaan. Selkeämpi ratkaisu oli kuitenkin se, että sovellukseen sijoitetaan vuokrasopimuslomake. Kun vuokrauksen tiedot syötetään sopimuslomakkeelle, lomake vie vuokraustiedot tietokantaan ja samalla luodaan oma tunniste jokaiselle vuokraukselle. Tällöin Vuokraukset-sivun taulukosta voidaan poistaa kunkin loppuun hoidetun vuokrauksen tiedot. Käyttöliittymässä näkyisi siten yhden kerran kunkin asiakkaan asiakastiedot, mutta jokainen vuokraus omana tietonaan.

11.4 Sovelluksen jatkokehitys, ideat parempaan toimivuuteen

Sovellus ei tullut valmiiksi opinnäytetyöskentelyn aikana. Käyttäjän mielestä sovelluksesta tulee hyvin selkeä ja se nopeuttaa alivuokrauksissa tarvittavan tiedon hallinnointia. Valmistuessaan se organisoi alivuokraustoiminnon sujuvammaksi hoitaa, koska tieto on yhdessä paikassa helposti löydettävissä ja sovelluksen painikkeet tekevät toimintoja suoraan.

Työn edetessä huomattiin asioita, jotka kannattaisi huomioida käyttömukavuuden ja käyttökelpoisuuden vuoksi. Jokainen tarvittava toiminto tulisi voida tehdä käyttöliittymässä, jotta käyttäjän ei tarvitsisi käydä esimerkiksi tietokannassa päivittämässä tietoa. Esimerkiksi myös Tilat-sivun tiedot tulisi voida päivittää Tilat-sivulta suoraan, koska myös niiden tiedot voivat muuttua.

Kulkukortin kulkuoikeuden päättyminen tulisi ”pompsahtaa” käyttäjän tietoon. Tarvittaessa asiaa voisi hoitaa eteenpäin tai kuitata hälytyksen pois. Avaimen palauttamatta jättäminen tulisi myös hälyttää sovelluksen käyttäjää.

Vuokrasopimuslomakkeelle kannattaa merkata kulkukortin voimassaoloaika. Silloin se olisi alivuokralaisen tiedossa ja helposti tarkastettavissa. Tällöin olisi helpompi välttää tilanne, jossa vuokratarve jatkuu, mutta kulkukortin voimassaolo on päättynyt.

Sovelluksen ulkonäkö voidaan muokata selkeämmäksi käyttäjälle. Jos taustakuva on liian levoton, se voi häiritä käyttäjää. Javascriptillä voi toimintoihin saada lisää käyttökelpoisuutta, kuten esimerkiksi navigaatiopalkin skaalautumisen pienemmäksi.

12 POHDINTA

Tutkimus lähti etsimään apua alivuokrausten hallinnointiin ja tilavarauksista tiedottamiseen henkilökunnalle. Tehtävä tuntui koostuvan monista erilaisista toiminnoista vuokrauksesta sopimisen ja avaintilausten kautta laskutukseen. Tieto tuntui pirstaleiselta. Täsmäsovelluksen kehittämiseen päädyttiin, koska tieto haluttiin yhteen sijaintiin. Sovellusta ei käytetä muuhun tarkoitukseen. Alivuokrausprosessin hallinnoinnin selkeytyminen vaikuttaa suoraan myös työnhallintaan. Tehtävän sujuvampi hoituminen vapauttaa resursseja toimiston muihin tehtäviin.

Alivuokrausprosessin tarkka pilkkominen auttoi suunnittelemaan selkeän ja tarpeeseen kohdistetun sovelluksen. Sovelluksen käyttöönotto voidaan aloittaa, vaikka ohjelmointi vielä jatkuukin. Sovellukseen voidaan jo syöttää tiedot asiakkaista, välitetyistä avaimista ja kulkokorteista sekä vuokrausten tiedot. Näitä ydintietoja on helppo alkaa seurata käyttöliittymän sivuilta, mikä houkuttaakin sovelluksen käytön aloittamiseen.

Tilakalenterit on tehty yhdistyksen Google G Suite -ympäristön kalenterisovellukseen, joten tiedonvälitys tilojen varaustilanteesta henkilökunnan kesken toimii erinomaisesti. Valmis kalenterisovellus oli perusteltua ja nopea ottaa käyttöön.

Laskutustiedot voidaan syöttää esimerkiksi Asiakastaulun Muuta-kenttään, niin silloin myös hintatiedot sijaitsevat sovelluksessa. Laskutustiedot kirjataan tulevaisuudessa käyttöliittymän vuokrauslomakkeelle, mikä onkin seuraava moduuli, joka sovellukseen toteutetaan. Tällä hetkellä integraatiota ei tarvita, eikä sitä tämän koulutuksen perusteella osaisi toteuttaakaan.

Koko opinnäytetyöskentelyn ajan pohdin, onko perusteltua koodata sovellus vain yhtä työtehtävää varten. Opintojen kertauksen näkökulmasta opinnäytteen aihe tuntui mainiolta. Lisäksi aihe kiinnosti ja sovellus hyödyttäisi valmistuessaan omaa työskentelyäni. Internetissä on paljon selkeitä web-pohjaisia sovelluksia, joita on helppo ottaa käyttöön. Muitakin täsmäsovelluksia on siis olemassa, eikä niiden tarvitsekaan olla laajoja ja monimutkaisia.

Sovelluskehityksessä on tärkeää ymmärtää mahdollisimman tarkoin tarve ja tehtävä, jota varten sovellus suunnitellaan. Olisi ihanteellista, jos koodaajalla olisi

monipuolista työkokemusta. Ainakin tässä kehitystyössä prosessin tarkka tunteminen hyödytti valtavasti suunnittelussa. Tilanne oli erikoinen; välikätenä toiminen kulkukorttien ja avainten luovuttamisessa. On toimittava luotettavasti kiinteistönomistajan suuntaan ja talon sisäinen tieto täytyy kulkea ajantasaisesti.

Nykyään työpaikoilla käytetään useita sovelluksia ja järjestelmiä eri työtehtävissä. Isoista järjestelmistä voidaan ottaa käyttöön vain tarvittavat ominaisuudet. Silti järjestelmien muut moduulit ja käyttöominaisuudet voivat jäädä näkyville ja siten ehkä sekoittaa työskentelyä. Tässä opinnäytetyössä sovellukselta toivottiin vain sellaisia ominaisuuksia, joita tehtävän hoitamisessa tarvitaan. Tarve oli helppo rajata ja integraatiota muihin järjestelmiin ei tarvittu. Kun sovellus valmistuu, se riittää pitkään alivuokrauksien hallinnointiin, vaikka toiminta kasvaisi.

Sovellusten tulisi olla käyttäjäystävällisiä ja ohjata käyttäjää siten, että sovelluksen käyttö on helppoa. Erillisiä yksityiskohtaisia tietoja siitä, miten sovellusta käytetään, ei nykyään enää saisi käyttäjältä vaatia. Kehitetty sovellus on selkeä ja sitä on helppo alkaa käyttää. Olisi mielenkiintoista tutkia, nopeuttaako sovellus todellisesti tehtävän hoitamista ja vaikuttaako oleellisesti työnhallintaan. Olisiko sillä yhtään vaikutusta työssä jaksamiseen?

Kehitysideoita tuli paljon mieleen sitä mukaa, kun työskentely edistyi. Sovelluksen yksityiskohtia voi vielä tulla hiottavaksi. Tiedon kokoaminen ja päivittäminen ei ole niin yksinkertaista, miltä se alussa tuntui olevan. Sovelluksen koodaaminen ei myöskään ole yksinkertaista. Se vaatii hyvin paljon aikaa ja monipuolista tietoa.

Koko maailma ottaa tällä hetkellä digiloikkia. Vaikka työtehtävien hoitaminen tehostuu, työtehtävien monipuolistuminen on todennäköistä. Jokaisen uuden tehtävän taustalla on paljon taustatietoa. Digitalisaatio vaatii siten paljon uuden oppimista ja ajan hermolla pysymistä. Vaikka yksittäisten tehtävien hoitaminen tehostuisi, miten hallitaan kaikki uusi tieto kaikkien uusien työtehtävien taustalla ja siten työtehtävät. Työntekijä kun ei kuitenkaan ole kone ja siten ohjelmoitavissa.

LÄHTEET

Bank, C & Cao, J. 2014. Web UI design best practices. Gdansk Puola. UXPin. E-kirja. Viitattu 27.11.2020 https://s3.amazonaws.com/uxpin/uxpin_web_ui_design_best_practices.pdf.

eLuotsi.fi 2020. Mitä responsiivinen tarkoittaa? Viitattu 26.11.2020 https://www.eluotsi.fi/responsiivisuus/.

Google.com 2020. Kalenterin luominen huonetta tai jaettua tilaa varten. Viitattu 25.11.2020 https://support.google.com/calendar/answer/44105?hl=fi.

Google.com 2021. Onko verkkosivusi mobiililaitteille sopiva? Viitattu 25.4.2021 https://search.google.com/test/mobile-friendly.

Guru99 2021. ER Diagram: Entity Relationship Diagram Model | DBMS Example. Viitattu 28.11.2021 https://www.guru99.com/er-diagram-tutorial-dbms.html.

Hannola, H. 2020. Tutkimus- ja kehittämistoimintojen perusteet. Viitattu 28.10.2020 https://moodle.eoppimispalvelut.fi/pluginfile.php/651768/mod_resource/content/2/Tutkimus-%20ja%20kehitt%C3%A4mistoiminnan%20perusteet%20I%20-%202020.pdf

Helsingin yliopisto 2021. Opiskelijan digitaidot. Viitattu 15.5.2021 https://blogs.helsinki.fi/opiskelijan-digitaidot/1-tietokoneen-kayton-perusteet/1-1-tietokoneen-toimintaperiaate/kayttojarjestelma-ja-kayttoliittyma/.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2004. Tutki ja kirjoita. Helsinki Tammi.

Hovi, A., Huotari, J. & Lahdenmäki T. 2005. Tietokantojen suunnittelu & indeksointi. 1.painos Jyväskylä Docendo. E-kirja. Viitattu 24.11.2021 https://www.ellibslibrary.com/fi/book/951-846-777-3, Lapin korkeakoulukonsernin LUC kirjaston kokoelma.

Juselius, U. 2004. HyperText Markup Language. Ulrika Juselius. Viitattu 7.11.2020 http://www.phpoint.fi/ulrikaj/www/html.htm.

Koskenniemi, Y. 2020. Tiedonhallinta. Suunnittelun vaiheita. Opetusmateriaali 2020.

Lukka, K. 2001. Konstruktiivinen tutkimusote. Metodix.com. Menetelmäartikkelit. Viitattu 29.10.2020 https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/.

Meador D. 2018. What is Data Dictionary. Viitattu 28.11.2021 https://www.tutorialspoint.com/What-is-Data-Dictionary.

Metsämuuronen, J. 2002. Tutkimuksen tekemisen perusteet ihmistieteissä 2 – opiskelijalaitos. Helsinki: International Methelp Oy.

Microsoft.com 2020. Perustiedot tietokannasta. Viitattu 27.11.2020 https://support.microsoft.com/fi-fi/office/perustiedot-tietokannasta-a849ac16-07c7-4a31-9948-3c8c94a7c204.

Oracle.com 2021. What is a Relational DAtabase (RDBMS)? Viitattu 18.11.2021 https://www.oracle.com/database/what-is-a-relational-database/.

Rapal.com 2020. Tarjoamme työkalut vuokrauskenhallintaan. Viitattu 25.11.2020 https://www.rapal.com/fi/vuokrauksenhallinta#sisaan-ja-ulosvuokraus.

Tuisku, N. 2020. Rovaniemen Seudun Steinerkouluyhdistys ry. Toiminnanjohtajan haastattelu 26.11.2020.

Turpeenniemi, K. 2020. Opinnäytetyö. Viitattu 28.10.2020 https://moodle.eoppimispalvelut.fi/pluginfile.php/651770/mod_resource/content/2/Tiede%20ja%20tutkimus%20-%202020.pdf

Vuokratili.fi 2021. Vuokratili – vuokranantajan sovellus. Viitattu 15.5.2021 https://www.vuokratili.fi/.

W3schools 2021a. CSS Tutorial. Online web tutorials.Viitattu 12.6.2021 https://www.w3schools.com/css/default.asp.

W3schools 2021b. MySQL Tutorial. Online web tutorials. Viitattu 12.6.2021 https://www.w3schools.com/mysql/default.asp.

W3schools 2021c. PHP Form Handling. Online web tutorials. Viitattu 12.6.2021 https://www.w3schools.com/php/php_forms.asp.

W3schools 2021d. PHP Introduction. Online web tutorials. Viitattu 12.6.2021 https://www.w3schools.com/php/php_intro.asp.

W3schools 2021e. Responsive Web Design – Media Queries. Online web tutorials. Viitattu 12.6.2021 https://www.w3schools.com/css/css_rwd_mediaqueries.asp.

W3schools 2020f. Responsive Web Design – The Viewport. Online web tutorials. Viitattu 14.12.2020 https://www.w3schools.com/css/css_rwd_viewport.asp.

W3schools 2021g. SQL Tutorial. Online web tutorials. Viitattu 12.6.2021 https://www.w3schools.com/sql/.

LIITTEET

Liite 1. Data Dictionary Tilavuokratietokanta

Liite 2. Koodiesimerkki html-lomakkeesta

Liite 3. Koodiesimerkki lomakkeen tiedon viemisestä php-scriptillä tietokantaan

Liite 4. Koodiesimerkki tulostaa tietokannan tiedot taulukkoon

Liite 1. Data Dictionary Tilavuokratietokanta 1(2)

Tilavuokratietokanta | Copyright (C) CRaG Sys

Printed On: 12.6.2021 16:09

Vuokratilojen edelleen vuokraamista varten suunnitellun sovelluksen tietokanta. Tulee vain toimistotyöntekijän käyttöön.

Asiakas

Data Member Name	Description	Type	Additional Type Information	Defult Value	Mandatory?
asiakasID	The unique identifier of the customer	Integer	6 digits	0	Yes
sukunimi	clients lastname	varchar(50)	text	null	Yes
etunimi	clients firstname	varchar(50)	text	null	Yes
organisaatio	clients organization	varchar(100)	text	null	No
yTunnus	business ID of the clients organization	varchar(15)	text	null	No
katuosoite	street address	varchar(50)	text	null	Yes
postinumero	postal code	varchar(10)	digits	null	Yes
paikkakunta	locality	varchar(30)	text	null	Yes
puhelin	phone number	varchar(20)	digits	null	Yes
sahkoposti	email	varchar(50)	text	null	Yes
verkkolaskuosoite	e-invoice address	varchar(50)	digits	null	No
muuta	else	varchar(200)	text	null	No

Entiteetti: Avain

Data Member Name	Description	Type	Additional Type Information	Defult Value	Mandatory?
avain	Avaintaulun yksilöivä autoincrement-kenttä	varchar(10)	10 digits	0	Yes
luovutettuPvm	pvm, jolloin avain on luovutettu asiakkaalle	date	text	null	No
palautettuPvm	pvm, jolloin asiakas on palauttanut avaimen	date	text	null	No
asiakasID	Asiakastaulun asiakkaaseen viittaava tunniste	int(6)	text	null	No
avainNro	Asiakkaalle luovutetun avaimen nro (kiinteistön omistajan numeroima)	varchar(10)	text	null	Yes

Hinta

Data Member Name	Description	Type	Additional Type Information	Defult Value	Mandatory?
hintaID	Hintataulun yksilöivä autoincrement-kenttä	int(11)	11 digits	0	Yes
tuntihinta	tuntikohtainen tilan vuokran hinta	double	digits	null	No
kkhinta	kuukausikohtainen tilan vuokran hinta	double	digits	null	No
pvhinta	päiväkohtainen tilan vuokran hinta	double	digits	null	No
ashinta	Asiakkaan kanssa sovittu hinta	double	digits	null	No
alvpros	vuokran alv-prosentti	double	digits	null	No
pvm	Päivämäärä, jolloin hinta on sovittu	date		null	No
tietoja	Muuta tietoa	varchar(150)	text	null	No

Kulkukortti

Data Member Name	Description	Type	Additional Type Information	Defult Value	Mandatory?
korttiID	Korttitaulun yksilöivä autoincrement-kenttä	Integer	6 digits	0	Yes
asiakasID	Asiakastaulun asiakkaaseen viittaava tunniste	varchar(50)	text	null	Yes
alkaa	Päivämäärä ja kellonaika, jolloin kortin voimassaolo alkaa	varchar(50)	text	null	Yes
paattyy	Päivämäärä ja kellonaika, jolloin kortin voimassaolo päättyy	varchar(100)	text	null	No
luovutettu	Päivämäärä, jolloin kulkukortti on luovutettu vuokralaiselle	varchar(15)	text	null	No
palautettu	Päivämäärä, jolloin kulkukortti on palautettu	varchar(50)	text	null	No
kortti_voimassa	Päivämäärä, johon saakka kulkukortti on voimassa.	varchar(10)	digits	ei1=kortti voimassa 0=kulkuoikeus vanhentunut	No

Liite 1. Data Dictionary Tilavuokratietokanta 2(2)

Tila						
Data Member Name	**Description**	**Type**	**Additional Type Information**	**Defult Value**	**Mandatory?**	**Unique?**
tilaID	Vuokrattavan tilan yksilöivä autoincrement-kenttä	Integer	11 digits	0	Yes	Yes
huonetila	Vuokrattavan tilan nimi	varchar(50)	text	null	Yes	No
tilan_nro	Vuokrattavan tilan nro kiinteistön pohjapiiroksessa	varchar(8)	text	null	No	No
pintaala	Vuokrattavan tilan pinta-ala	varchar(8)	text	null	No	No
hintaID	Vuokrattavan tilan hinta	int(11)	11 digits	null	No	No

Vuokraus						
Data Member Name	**Description**	**Type**	**Additional Type Information**	**Defult Value**	**Mandatory?**	**Unique?**
vuokrausID	Kunkin vuokrauksen yksilöivä autoincrement-kenttä	Integer(11)	11 digits	0	Yes	Yes
asiakasID	Asiakkaan yksilöivä asiakasnro asiakas-taulusta	Integer(11)	11 digits	null	Yes	Yes
tilaID	Tilan yksilöivä tilan nro tila-taulusta	Integer(11)	11 digits	null	Yes	Yes
viikonpaiva	sanallisesti viikkonpäivä, jolloin tila on varattu	varchar(20)	text	null	No	No
alkaaKlo	Kellonaika, jolloin varaus alkaa	time		null	No	No
paattyyKlo	Kellonaika, jolloin varaus päättyy	time		null	No	No
alkaapvm	Kellonaika, jolloin varaus alkaa	date		null	No	No
Paattyypvm	Kellonaika, jolloin varaus alkaa	date		null	No	No

Liite 2. Koodiesimerkki html-lomakkeesta

```
<?php
echo '<div class="form">';
echo '<form action="asiakasinsert.php" method="post">
        <ul>Sukunimi: <input type="text" name="sukunimi"></ul>
        <ul> Etunimi: <input type="text" name="etunimi"> </ul>
        <ul> Organisaatio: <input type="text" name="organisaatio"></ul>
        <ul> Y-tunnus: <input type="text" name="ytunnus"> </ul>
        <ul> Katusoite: <input type="text" name="katuosoite"></ul>
        <ul> Postinumero: <input type="text" name="postinumero"> </ul>
        <ul> Paikkakunta: <input type="text" name="paikkakunta"> </ul>
        <ul> Puhelin: <input type="text" name="puhelin"></ul>
         <ul>  Sähköposti: <input type="text" name="sahkoposti"> </ul>
        <ul> <input type="submit" value="Lisää asiakas tietokantaan"> </ul>
        </form>';
echo '</div>';
?>
```

Liite 3. Koodiesimerkki lomakkeen tiedon viemisestä php-scriptillä tietokantaan

```
$sukunimi=$_POST['sukunimi'];
$etunimi=$_POST['etunimi'];
$organisaatio=$_POST['organisaatio'];

$sql="INSERT INTO asiakas(sukunimi, etunimi, organisaatio) VALUES ('$suku-
nimi', '$etunimi', '$organisaatio')";

if($con->query($sql)===TRUE){
   header ('location:../index.php?sivu=asiakas.php';
}   else{
    echo "Error:" .$query. "<br>" .$con->error;
}
$con->close();
```

Liite 4. Koodiesimerkki tulostaa tietokannan tiedot taulukkoon

```
while($row = mysqli_fetch_array($tulos)){
    echo "<tr>";
    echo "<td>".$row['asiakasID']."</td>";
    echo "<td>".$row['sukunimi']."</td>";
    echo "<td>".$row['etunimi']."</td>";
    echo "<td>".$row['organisaatio']."</td>";
    echo "<td> <a href=update.php>Update</a></td>";
    echo "<td><a href=delete.php?id=".$row['asiakasID'].">Del</a></td>";
    echo "</tr>";
```