Tanel Tuuha

Tietokoneohjelmien synkronointi resurssivarausjärjestelmään Metropolia Ammattikorkeakoulussa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

20.6.2023

Tiivistelmä

Tekijä:	Tanel Tuuha
Otsikko:	Tietokoneohjelmien synkronointi resurssivarausjärjestelmään Metropolia Ammattikorkeakoulussa
Sivumäärä:	35 sivua
Aika:	20.6.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Tietojenkäsittely ja tietoliikenne
Ohjaajat:	Osaamisaluepäällikkö, Janne Salonen

Tässä insinöörityössä perehdytään työn toimeksiantajan Metropolia ammattikorkeakoulun käytössä olevan Peppi-opintohallintajärjestelmän resurssi- ja varausjärjestelmän käyttöliittymän ja rajapintojen kautta tietokoneohjelmien inventaarion synkronoimiseksi siihen.

Työn suunnitteluvaiheessa tutustutaan varausjärjestelmään käyttöliittymän kautta ja rajapintoja hyödyntäen. Varattavien tilojen ominaisuuksien tai varusteiden esittämiseen kiinnitetään erityistä huomiota, koska se on sopiva paikka myös tietokoneohjelmille järjestelmässä.

Synkronointia varten selvitetään myös, miten saadaan tieto tietokoneohjelmista ja niiden sijainnista SCCM (System Center Configuration Manager) työkalun avulla.

Työn toteutusvaiheessa hyödynnetään suunnitteluvaiheen selvityksien tuloksia ja piirretään vuokaaviot kolmivaiheisesta synkronointiprosessista. Prosessin perusteella ohjelmoidaan ohjelma python-ohjelmointikielellä ja sitä testataan suunnitelluilla testeillä toimivuuden varmistamiseksi.

Synkronoinnin toimiessa tulee esille erilaisia kehityskohteita ja virheitä järjestelmästä, jotka välitetään eteenpäin varausjärjestelmän kehityskanavia pitkin.

Lopputuloksena Metropolia Ammattikorkeakoululle tehtiin ohjelma, joka synkronoi tietokoneohjelmat tarkoituksenmukaisesti mahdollisimman vähäisellä vaivalla varausjärjestelmään ja niitä voidaan käyttää tilojen hakuun esimerkiksi opetuksen suunnittelussa.

Avainsanat: Peppi, SCCM, Varausjärjestelmä, Rajapinta, Ohjelmistoinventaario, Resurssihallinta

Abstract

Author:	Tanel Tuuha
Title:	Synchronizing computer programs to reservation system in Metropolia University of Applied sciences
Number of Pages:	35 pages
Date:	20 June 2023
Degree:	Bachelor of Engineering
Degree Programme:	Degree Programme in Information and Communication Technology
Professional Major:	Information and Communication Technologies (ICTs)
Supervisors:	Janne Salonen, Head of School

In this engineering work, we familiarize ourselves with the resource and reservation system of the Peppi study management system used by the client, Metropolia University of Applied Sciences, through the user interface and API interfaces to synchronize the inventory of computer programs with it.

In the planning phase of the work, the reservation system is familiarized with the user interface and using the API interfaces. Special attention is paid to presenting of the features or equipment of the rooms to be booked. Because it is also a suitable place for computer programs in the system.

For synchronization, we also find out how to get information about computer programs and their location using the SCCM (System Center Configuration Manager) tool.

In the implementation phase of the work, the results of the planning phase are used, and flowcharts of the three-phase synchronization process are drawn. Based on the process, the synchronizing program is programmed in the Python programming language and evaluated with planned tests to ensure functionality.

When the synchronization was used, various development ideas and errors come up. These were forwarded to the reservation system's development channels.

As a result, a program was created for Metropolia University of Applied Sciences, which synchronizes the computer programs appropriately with as little effort as possible into the reservation system, and they can be used to search for rooms, for example, in teaching planning.

Keywords: Peppi, SCCM, Reservation system, API, Software inventory, Resource management

Sisällys

Lyhenteet

API: Ohjelmointirajapinta eli API (engl. Application Programming Interface) on sopimus, joka mahdollistaa koodin vuorovaikutuksen muiden koodien kanssa. API:t ovat nykyaikaisen ohjelmiston rakennuspalikoita, sillä ne mahdollistavat resurssien ja palveluiden jakamisen eri sovelluksien, organisaatioiden ja laitteiden välillä.

SOAP API: Simple Object Access Protocol API on protokolla, joka mahdollistaa tiedon välittämisen sovellusten välillä verkossa. Se on yleisesti käytetty viestintäprotokolla web-palveluille ja käyttää XML-tietomuotoa.

XML: eXtensible Markup Language on merkintäkieli, jota käytetään tiedon tallentamiseen ja välittämiseen. Se on suunniteltu erityisesti tietojen rakenteen kuvaamiseen ja niiden helppoon käsittelyyn erilaisten järjestelmien välillä.

SQL: Structured Query Language on tietokantakyselyjen kieli, jota käytetään tietokantojen hallintaan, tiedon tallentamiseen, muokkaamiseen ja hakemiseen. Se on standardoitu kieli, jota käytetään laajalti relaatiotietokannoissa.

CSV: CSV-tiedosto (Comma-Separated Values) on yleisesti käytetty tiedostomuoto tietojen tallentamiseen ja jakamiseen taulukkolaskentaohjelmien, tietokantojen ja muiden sovellusten välillä. CSV-tiedosto koostuu tekstimuotoisista tietueista, joissa kentät tai sarakkeet erotellaan yleensä pilkuilla.

1 Johdanto

Tässä insinöörityössä tullaan tarkastelemaan mahdollisuutta tietokoneiden ohjelmistoinventaarion synkronointia varausjärjestelmän tilatietoihin Metropolia Ammattikorkeakoulun toimeksiantona. Työn teoriaosuudessa luodaan pohjaa ymmärrykselle työn taustasta ja historiasta. Tämän jälkeen Python-ohjelmointikielellä rakennetaan ohjelmisto, joka automatisoi ohjelmistoinventaarion synkronoinnin Peppi opintohallintajärjestelmän varaus- ja resurssisuunnittelu osion kanssa. Lopputulosta lähestytään sekä teorian että käytännön esimerkkien kautta, jotka havainnollistavat synkronointiprosessin toteuttamista.

Tämä projekti ei ainoastaan vastaa työn toimeksiantajan tarpeisiin, vaan myös edistää ymmärrystä ohjelmistoinventaarion integroimisesta varausjärjestelmiin. Yhdistämällä taustatiedot ja käytäntö pyritään parantamaan ymmärrystä ja helpottamaan vastaavien toteutusten tekemistä tulevaisuudessa. Samalla tukemalla opetushenkilöstöä ja opiskelijoita Suomen korkeakoulukentällä.

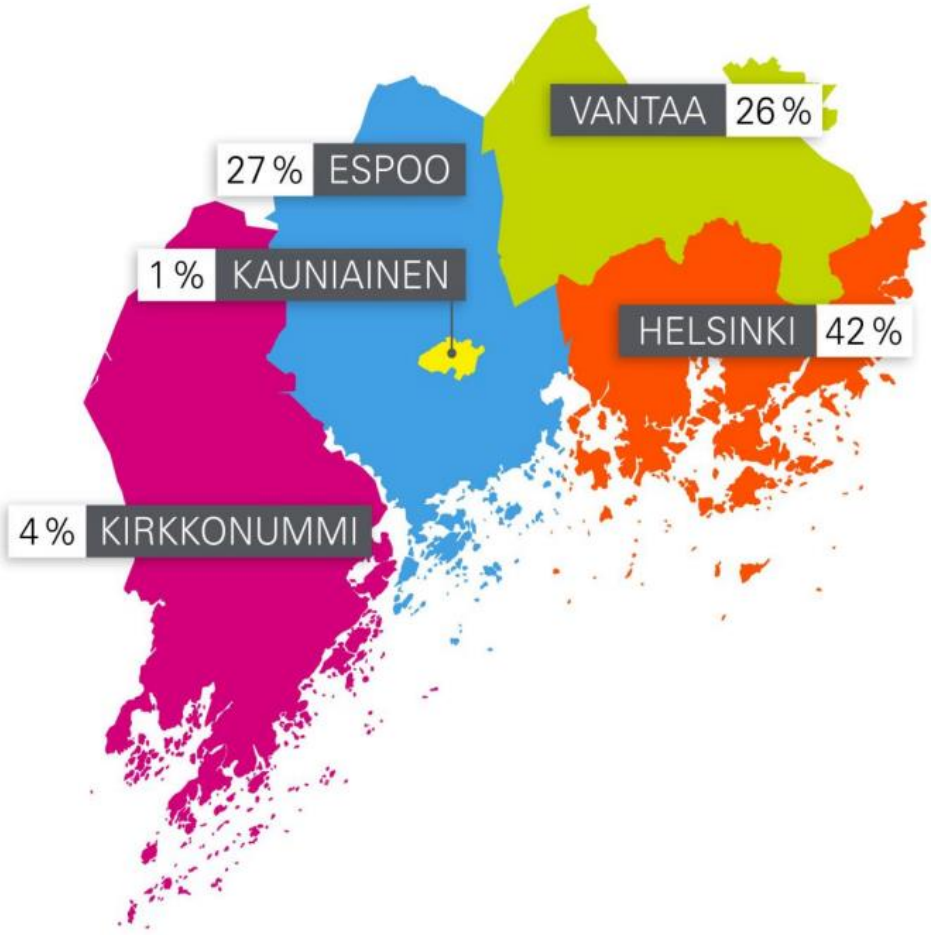
Ohjelmisto mahdollistaa ajantasaisen ja luotettavan ohjelmistoinventaarion ylläpidon, mikä helpottaa varaus- ja resurssihallintaa. Työn avulla voidaan myös havainnollistaa erilaisten järjestelmien integrointia ja tietojen synkronointia. Lisäksi työn tuloksena saatua ohjelmistoa voidaan hyödyntää laajemmin muissakin organisaatioissa, joissa on tarvetta vastaavalle synkronointiratkaisulle. Se tarjoaa myös konkreettisen esimerkin siitä, miten automaattinen synkronointi voidaan toteuttaa käytännössä Python-ohjelmointikielellä.

2 Työn lähtökohdat

Insinöörityön tavoitteena on toteuttaa käytännössä kehitystyö, jossa synkronoidaan tietokoneille asennetut ohjelmat ja niiden fyysiset sijainnit ohjelmistoinventaariosta käytössä olevaan resurssihallinta- ja varausjärjestelmään. Varausjärjestelmän kautta voidaan hakea työn toteuduttua tiloja tilan tietokoneille asennettujen ohjelmien perusteella esimerkiksi opetuksen varauksien suunnitellussa tai, kun tilaa etsitään itsenäiselle työskentelylle.

2.1 Toimeksiantajana Metropolia Ammattikorkeakoulu

Metropolia Ammattikorkeakoulu on yksi Suomen suurimmista kansainvälisesti auditoiduista ammattikorkeakouluista, joka tarjoaa monipuolista korkeakoulutusta eri aloilta. Metropolia on profiloitunut erityisesti tekniikan, liiketalouden, terveydenhuollon ja sosiaalialan koulutuksen tarjoajana. Metropolia perustettiin vuonna 2008 yhdistämällä kolme aiempaa ammattikorkeakoulua Helsingissä, Espoossa ja Vantaalla, joissa sijaitsee myös nykyiset Metropolian kampukset. Metropolian omistajuus on kuitenkin hajautetumpaa kuten Kuva 1 nähdään. Se on jaettu seuraavin osuuksin Helsinki 42 %, Espoo 27 %, Vantaa 26 %, Kirkkonummi 4 % ja Kauniainen 1 %. Metropolia kattaakin näin monella tapaa koko Helsingin Metropolia-alueen. (Metropolia Ammattikorkeakoulu 2023)



Kuva 1 Metropolian omistajakunnat (Metropolia 2023)

Metropolian kampukset eli oppilaitokset tai koulurakennukset sijaitsevat pääkaupunkiseudulla ja ne tarjoavat modernit työ-, opetus- ja oppimistilat henkilökunnalle ja opiskelijoille. Tilojen saatavuus ja tehokas käyttö varmistetaan Peppi-opintohallintajärjestelmän resurssi- ja varausjärjestelmällä tai ressulla lyhyesti. Peppi-opintohallintajärjestelmä tunnetaan Metropoliassa myös nimellä OMA ja OMA-intranet. Järjestelmä saattaa yhteen erilaiset opetuksessa tarvittavat resurssit, kuten opetushenkilökunnan, toteutuksen eli kurssin tiedot, tilat ja välineet sekä itse opiskelijat. Järjestelmä on tärkeä ja sitä kehitetäänkin aktiivisesti korkeakoulujen kesken Peppi-konsortiossa.

2.2 Kehittämiskohteen kuvaus ja kehittämistavoitteet

Työn lähtökohtina on Peppi-opintohallintojärjestelmän resurssien suunnittelu- ja varausominaisuus, jonka kautta hallitaan tilojen tietoja ja tehdään varauksia Metropolian tiloihin. Järjestelmän kautta varaavilla on usein tarve löytää tila, jossa on tietokoneelle asennettu tietty ohjelma tai tietyn ohjelman tietty versio. Tieto ohjelmista tietyillä tietokoneilla on olemassa sekä tieto tietokoneiden sijainnista on olemassa. Tietoa ei ole vain vielä onnistuttu yhdistämään varausjärjestelmään.

Ohjelmistoinventaarion synkronointia varausjärjestelmään on mietitty Metropolia Ammattikorkeakoulussa jo pitkään. Välillä aihe on ollut enemmän pinnalla ja välillä jätetty hautumaan erilaisien syitten takia. Esteenä on ollut muun muassa suuri ohjelmien määrä, niiden kohdistaminen tiloihin, niiden suodatus sekä näkyminen loppukäyttäjälle. Hallintakin on nähty käytännössä mahdottomuutena olemassa olevilla keinoilla kuten käsin kirjaamisella tai CSV-tiedostona siirtämisenä. Johtuen suuresta ohjelmien ja varausjärjestelmässä olevien tilojen määrästä sekä tarpeesta pitää tiedot ajan tasalla eli tiheästä päivitysvälistä. (Järkkä 2023)

3 Työn suunnittelu

Työn suunnittelu alkoi tutustumalla huolellisesti varausjärjestelmään ja ohjelmistoinventaarioon, joiden välillä synkronointi on tarkoitus toteuttaa. Varausjärjestelmän avulla saadaan yleiskuva varustuksista, jotka merkitään varustenipuilla tilojen tietoihin. Tämä on ensiarvoisen tärkeää, jotta voidaan suunnitella tapa näyttää tietokoneohjelmat käyttäjille varausjärjestelmässä järkevästi.

Ohjelmistoinventaario tarjoaa tärkeää tietoa olemassa olevista ohjelmista ja niihin liittyvistä tiedoista. Tutustumalla ohjelmistoinventaarioon saadaan selville, mitä ohjelmistoja on jo käytössä ja missä tilassa. Lisäksi selvitetään missä muodossa tieto on. Näin voidaan suunnitella synkronointiohjelmiston toimivuus siten, että oikeat ohjelmistot näkyvät oikeiden tilojen tiedoissa.

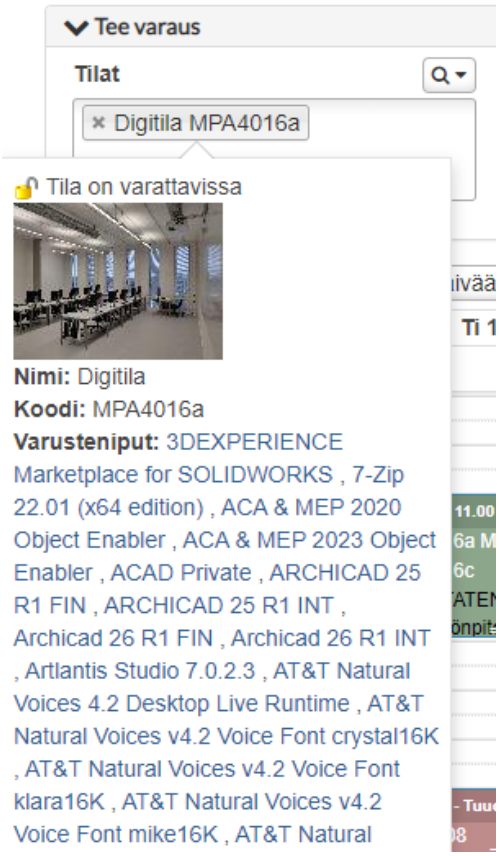
3.1 Varausjärjestelmä

Varausjärjestelmän varustenippuominaisuuteen tutustuminen avasi uusia mahdollisuuksia. Varusteniput toimivat kätevänä tapana yhdistää useita varusteita yhdeksi kokonaisuudeksi, mikä helpottaa niiden hallintaa ja tiloihin liittämistä. Jokainen varustenippu näkyy omana yksikkönään, jossa luetellaan varusteryhmät ja varusteet niiden sisällä. Oletuksena varusteniput ovat kiinni Kuva 2 mukaisesti. Varustenipun käyttäjä saa auki varustenippua hiirellä painamalla, jolloin se aukeaa ja näyttää varusteet Kuva 3 mukaisesti. Tämä antaa käyttäjälle kattavan kuvan siitä, mitä varusteita tilassa on saatavilla.

Kuva 2 Varustenipun näkyminen tilan tiedoissa

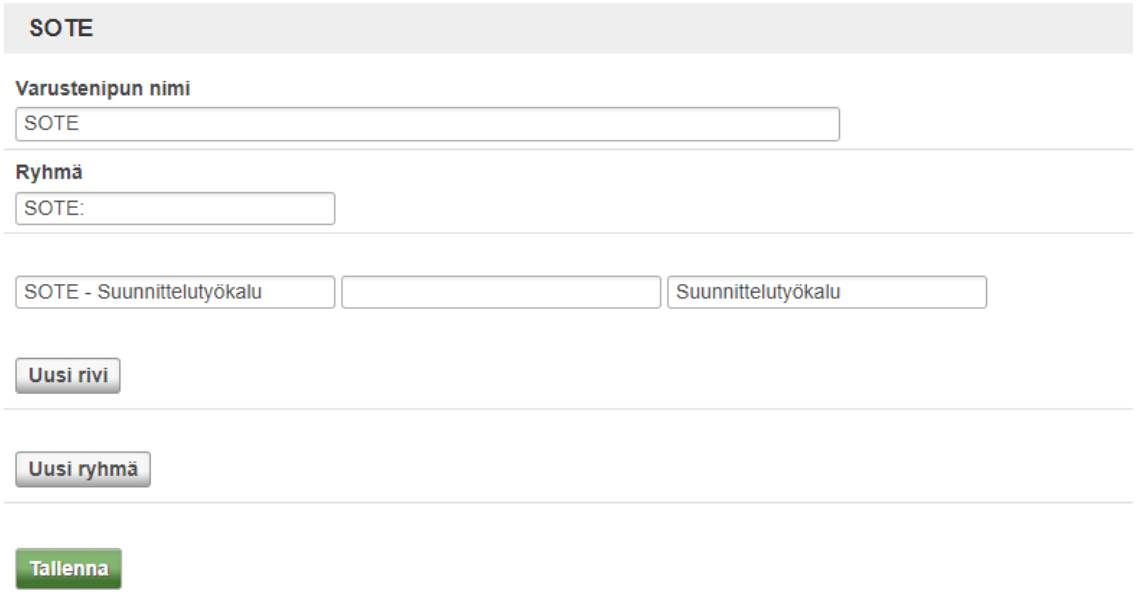
Kuva 3 varustenipun näkyminen tilan tiedoissa auki

Ohjelmat tulisi lisätä varausjärjestelmään varustenippujen sisälle eikä yksittäisinä nippuina. On tärkeää huomioida, että näkymä tukkeutuisi liiallisella tiedolla kuten Kuva 4 on kuvitettu. Tästä syystä järjestelmä tarjoaa mahdollisuuden selkeään ja jäsenneltyyn näkymään varustenipuista. Käyttäjä pystyy näkemään yhdellä silmäyksellä mitä etsii ja halutessaan selaamaan tilassa olevia ohjelmia. Näin käyttäjä voi tehokkaasti valita juuri tarvitsemansa ohjelmat ilman tarpeetonta selailua.



Kuva 4 Kuvitettu kuva tilan tiedoista, jos kaikki ohjelmat olisivat erillisiä varustenippuja

Varustenipun rakenne järjestelmässä on yksinkertainen ja helppokäyttöinen. Se koostuu ensinnäkin nimestä, joka kuvaa varustenippua tiiviisti ja ytimekkäästi. Tämä auttaa käyttäjää tunnistamaan varustenipun sisällön heti. Varustenippu sisältää Ryhmän, joka taas sisältää luettelon varusteista ja niiden tiedoista kuten Kuva 5 on esitetty.



Kuva 5 Kuvakaappaus varustenipun rakenteesta

Useampi varustenippu ja tila voidaan linkittää toisiinsa riippuen tarpeesta kuvan 6 tapaan. Näin voidaan jakaa ohjelmistoinventaario tarvittaessa yhdelle tai useammalle tilalle.

Kuva 6 Kuvakaappaus varustenipun ja tilojen liitoksista

Koska ohjelmistoinventaarion synkronointi on tarkoitus toteuttaa automaattisesti eikä manuaalisesti, tutustuttiin resurssi- ja varausjärjestelmän tekniseen dokumentaatioon. Dokumentaatiossa kuvailtiin ohjelmiston toimintaa ja API-

rajapintoja. Rajapinnat olivat soap-tyyppisiä ja kutsut XML-muodossa esimerkkikoodin 1 tapaan. Lisäksi rajapintoja testattiin SoapUI-ohjelmalla staging- eli testiympäristössä. SoapUI mahdollisti rajapintojen testauksen ja tarjosi työkaluja testitapausten luomiseen ja testien suorittamiseen. Näin varmistettiin, että ohjelman käyttämät rajapinnat toimivat odotetulla tavalla ja kommunikoivat oikein resurssi- ja varausjärjestelmän kanssa.

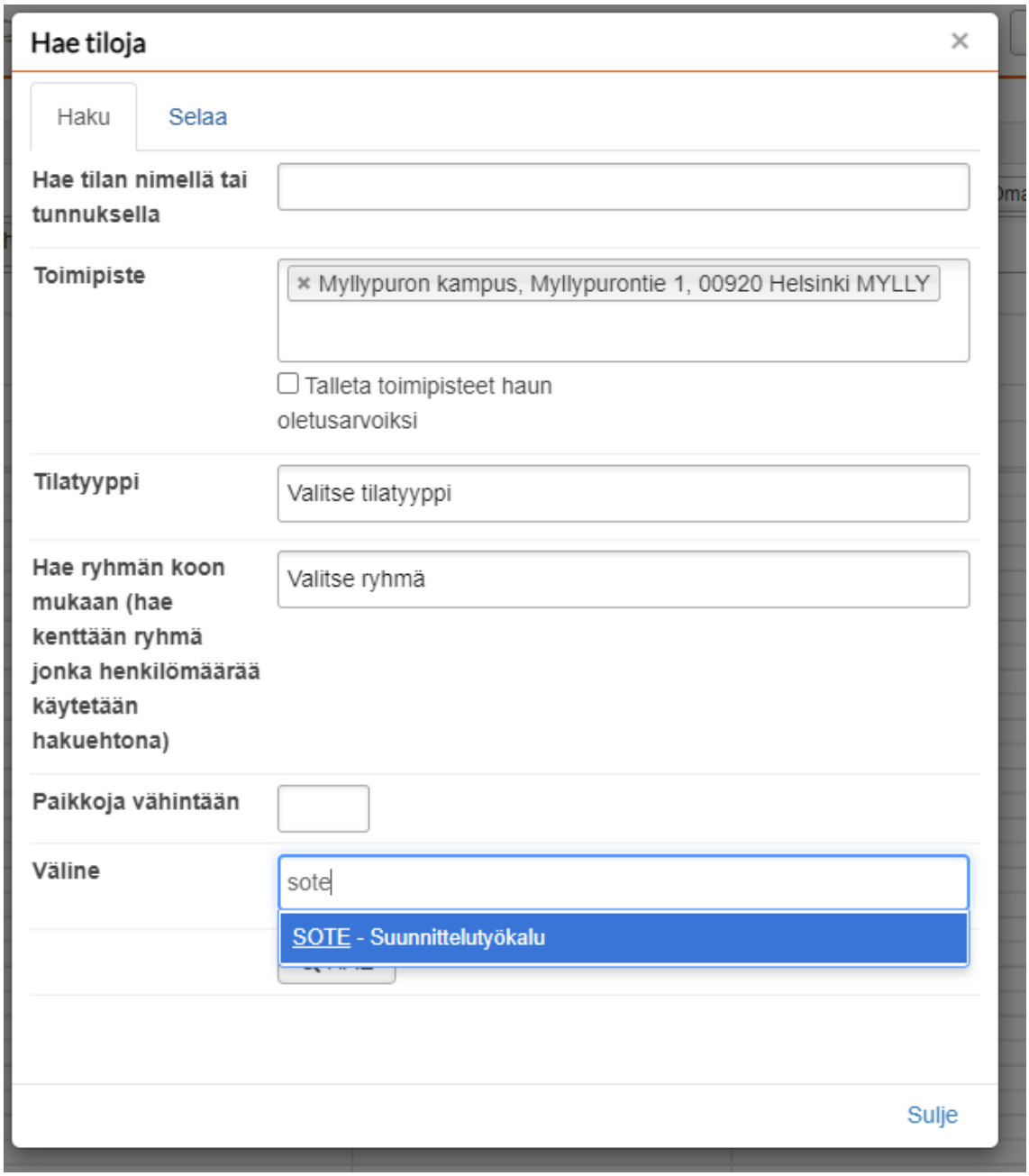
```
<soapenv:Envelope>
<soapenv:Header/>
<soapenv:Body>
<end:addAccessoryBundle>
<arg0>
     <impl:id>?</impl:id>
     <impl:title>
          <valueEn>Bundle</valueEn>
          <valueFi>Nippu</valueFi>
          <valueSv>Knippa</valueSv>
     </impl:title>
</arg0>
</end:addAccessoryBundle>
</soapenv:Body>
</soapenv:Envelope>
```

Esimerkkikoodi 1.

Varausjärjestelmän tilojen hakemiseen käytetty hakutoiminto on suuressa roolissa, koska kun haetaan tiloja ohjelmalla ei haluta etsiä suuresta joukosta ohjelmia ja tiloja sopivaa vaihtoehtoa. Hakutoiminto ehdottaa haluttua varustetta varustenipuista, kun käyttäjä alkaa kirjoittamaan varusteen nimeä hakukenttään. Tämä ennakoiva toiminta auttaa käyttäjää löytämään haluamansa ohjelman ilman tarvetta kirjoittaa koko ohjelman nimeä. Tämä ominaisuus nopeuttaa ja helpottaa käyttäjän työskentelyä, kun hän voi löytää tarvitsemansa varusteen nopeasti ja vaivattomasti.

Lisäksi, kun hakutoiminto suorittaa haun, järjestelmä suodattaa mahdolliset kaksoiskappaleet. Tämä tarkoittaa, että jos etsittävä varuste löytyy useammasta varustenipusta, hakutuloksissa näytetään varuste vain kerran Kuva 7 mukaisesti.

Kuitenkin hakutoiminto ottaa huomioon kaikki varusteniput, joissa ohjelma esiintyy, jotta käyttäjä saa kattavan listan tiloista, joissa ohjelma on saatavilla.



Kuva 7 Tilojen hakeminen varustenipuilla

Yhteenvetona voidaan todeta, että varausjärjestelmä on valmis tukemaan tilojen hakua tietokoneohjelmien perusteella. Hakutoiminto, joka ennakoivasti ehdottaa ohjelmia käyttäjän kirjoittaessa ohjelman nimeä hakukenttään, mahdollistaa

nopean ja vaivattoman ohjelmien löytämisen tuhansien ohjelmien joukosta. Lisäksi järjestelmä suodattaa hakutuloksista kaksoiskappaleet ja näyttää ohjelman vain kerran, mutta käyttää kaikkia varustenippuja, joissa ohjelma esiintyy. Tämä parantaa käyttäjäkokemusta ja helpottaa ohjelmistojen hallintaa. Varausjärjestelmä mahdollistaa tehokkaan synkronoinnin tietokoneohjelmien ja tilojen välillä, jolloin käyttäjät voivat hyödyntää olemassa olevia resursseja optimaalisesti.

3.2 Ohjelmistoinventaarion kuvaus

Ohjelmistoinventaario RoomProgList tietokantataulu koostetaan SCCM (System Center Configuration Manager) -työkalun tietokannoista yhdistämällä kolme tietokantataulua: Add_Remove_Programs_64_DATA, Add_Remove_Programs_32_DATA ja v_r_system. Yhdistämällä nämä kolme tietokantataulua saadaan aikaiseksi kokonaisvaltainen lista kaikista asennetuista tietokoneohjelmista ja tietokoneiden sijainneista. (Järkkä 2023)

SCCM (System Center Configuration Manager) on Microsoftin kehittämä työkalu, joka tarjoaa keskitetyn hallinnan ja automaation IT-infrastruktuurin asennuksille, päivityksille, ja hallinnalle. SCCM:llä organisaatiot voivat tehokkaasti hallita laitteita, sovelluksia ja käyttöjärjestelmiä, ja varmistaa niiden ajantasaisuuden ja turvallisuuden. (Rubenstein 2020)

Ensimmäinen alla kuvattu tietokantataulu Taulukko 1, Add_Remove_Programs_64_DATA, sisältää tiedot 64-bittisistä asennetuista ohjelmistoista. Tästä taulusta saadaan tietoa ohjelmistojen nimistä, versioista, julkaisijoista ja muista asiaankuuluvista tiedoista. Tämä taulu tarjoaa yksityiskohtaista tietoa 64-bittisistä ohjelmistoista, jotka on asennettu tietokoneisiin SCCM:n avulla. (Järkkä 2023)

Add_Remove_Programs_64_DATA	
Sarake	**Esimerkki**
Displayname00	3DEXPERIENCE Marketplace

MachineID	123
Timekey	2023-05-29 00.50.54.000

Taulukko 1 kuvaus taulusta Add_Remove_Programs_64_DATA

Toinen tietokantataulu Taulukko 2, Add_Remove_Programs_32_DATA, vastaa 32-bittisten asennettujen ohjelmistojen tiedoista. Tässä taulussa on samanlaisia tietoja kuin edellisessäkin, mutta se keskittyy erityisesti 32-bittisiin ohjelmistoihin. Näin varmistetaan, että inventaariotietokanta sisältää täydellisen kuvan sekä 64- että 32-bittisistä asennetuista ohjelmistoista. (Järkkä 2023)

Add_Remove_Programs_32_DATA	
Sarake	**Esimerkki**
Displayname00	3DEXPERIENCE Marketplace
MachineID	123
Timekey	2023-05-29 00.50.54.000

Taulukko 2 kuvaus taulusta Add_Remove_Programs_32_DATA

Kolmas tietokantataulu Taulukko 3, v_r_system, sisältää yleiset tiedot tietokoneista, kuten niiden nimet ja sijainnit. Tämä taulu toimii yhdistävänä tekijänä Add_Remove_Programs_64_DATA- ja Add_Remove_Programs_32_DATA-taulujen kanssa. Näiden taulujen yhdistäminen v_r_system-tauluun mahdollistaa ohjelmistojen liittämisen tiettyihin tietokoneisiin ja antaa kattavan kuvan ohjelmistoinventaariosta järjestelmässä. (Järkkä 2023)

v_r_system	
Sarake	**Esimerkki**
ResourceID	123
Name0	AR-P23

Taulukko 3 kuvaus taulusta v_r_system

Kun nämä kolme tietokantataulua yhdistetään, saadaan luotua RoomProgList-tietokanta eli ohjelmistoinventaario, joka on kuvattu Taulukko 4. Tässä tietokannassa on kaikki tarvittavat tiedot asennetuista ohjelmistoista, niiden versioista, julkaisijoista sekä niiden liittämisestä tiettyihin tietokoneisiin. Tämä inventaario-tietokanta mahdollistaa tarkat haut järjestelmän asennetuista ohjelmistoista ja auttaa tehokkaassa ohjelmistojen hallinnassa ja ylläpidossa. (Järkkä 2023)

RoomProgList		
Sarake	**Tietomuoto**	**Esimerkki**
sessionline	int(1,1)	3716835
ItRoomName	varchar(16)	AR-P23
RoomName	varchar(16)	ARP23
ProgName	varchar(128)	3DEXPERIENCE Marketplace
ProgBase	varchar(4)	64b
Rundate	datetime	2023-05-29 00.50.54.000

Taulukko 4 Ohjelmistoinventaarion SQL-tietokantataulun kuvaus

4 Työn toteutus

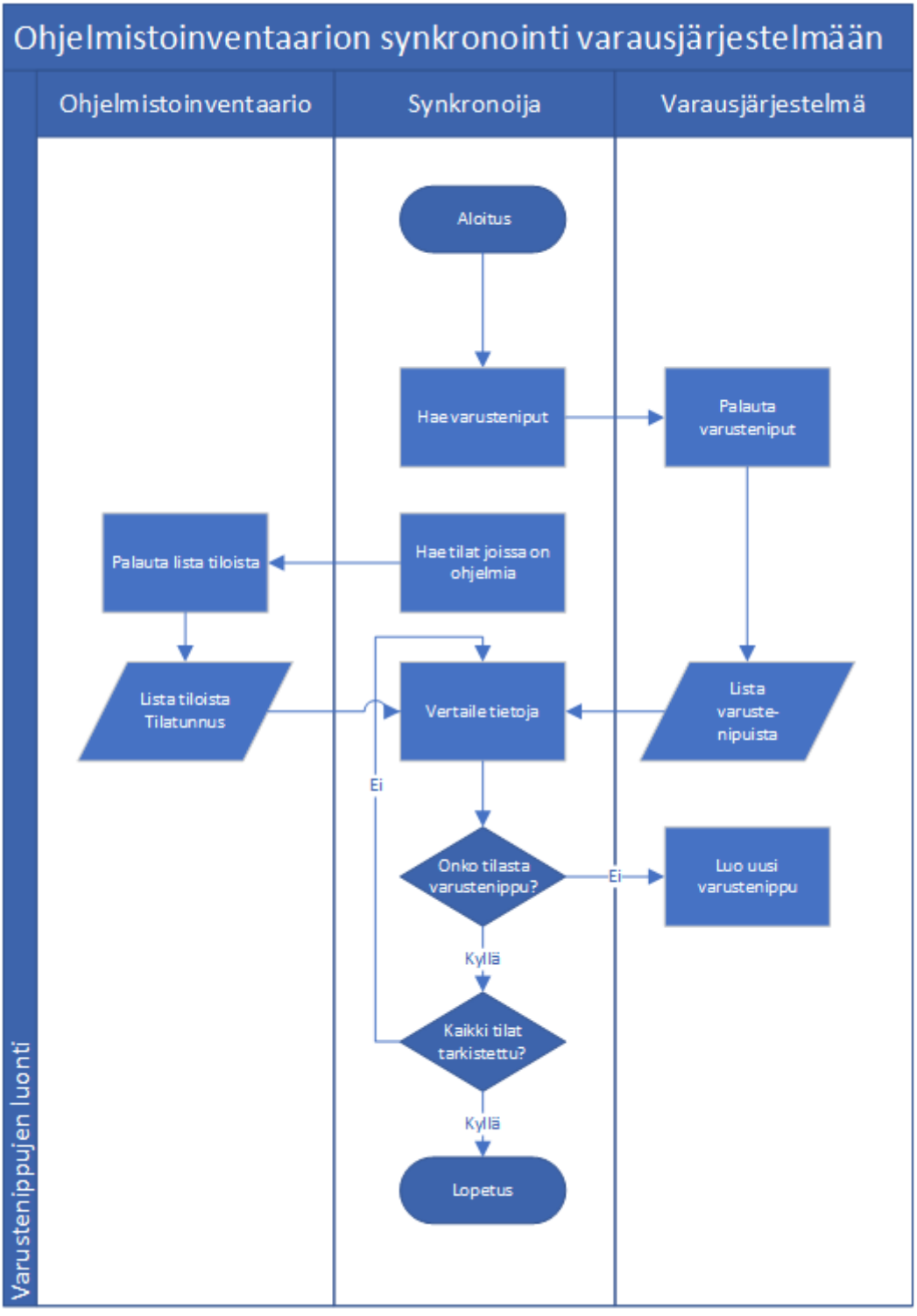
Kehitystyön toteutus voidaan jakaa kolmeen vaiheeseen: suunnittelu, ohjelmointi ja testaus. Suunnitteluvaiheessa määritellään tarkemmin kehitettävän ohjelmiston tai järjestelmän vaatimukset ja tavoitteet. Tämä sisältää tarpeiden ymmärtämisen ja tavoitteiden määrittämisen.

Ohjelmointivaiheessa toteutetaan suunnitteluvaiheessa määritellyt toiminnot ja rakenteet Python-koodikielllä. Apuna käytetään myös erilaisia valmiita liitännäisiä kaikkien yleisien toimintojen aikaan saamiseksi. Testausvaiheessa varmistetaan, että kehitetty ohjelmisto toimii oikein ja täyttää asetetut tavoitteet. Testaus voi sisältää virheiden havaitsemista, vianmääritystä ja korjausten tekemistä.

4.1 Suunnittelu

Työn toteutus jaettiin kolmeen loogiseen vaiheeseen: varustenippujen luonti, varustenippujen sisällön päivittäminen ja varustenippujen yhdistäminen tiloihin. Jokainen vaihe oli korvaamaton osa synkronointia ja vaati oman erityisen huomionsa. Vaiheiden suunnittelu ja hahmottaminen tehtiin käyttämällä Microsoft Vision vuokaavio- ja kaavio-ohjelmaa, jonka avulla luotiin helposti ymmärrettäviä vuokaavioita.

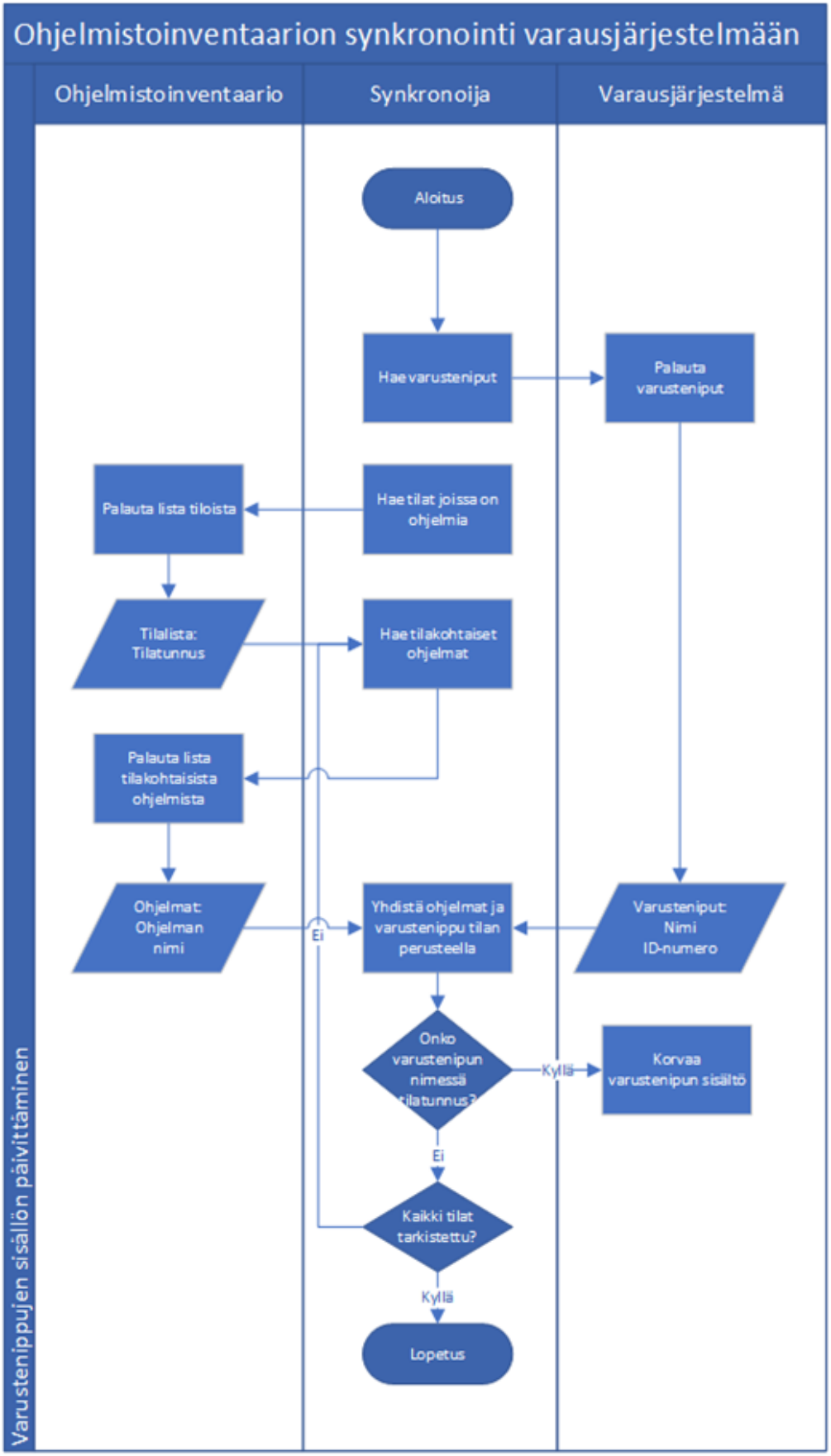
Ensimmäisessä vaiheessa, joka on kuvattu Kaava 1 varustenippujen luonti, keskityttiin uusien varustenippujen luomiseen järjestelmään. Tämän vaiheen tarkoitus on luoda uusia varustenippuja tiloittain Pepin resurssihallinta- ja varausjärjestelmään ohjelmistoinventaarion perusteella, jos niitä ei vielä ole olemassa. Esimerkiksi synkronointiohjelma tulisi määrittelemään varustenipun nimen ”AR304 Ohjelmat” ja ryhmäksi ”Ohjelmat”. Tämä vaihe vaatii tietoa ohjelmistoinventaariosta ja varausjärjestelmästä rajapinnan yli päättelyä varten. Päättelyn perusteella taas kirjoitetaan rajapinnan yli varausjärjestelmään.



Kaava 1 Varustenippujen luontiprosessi

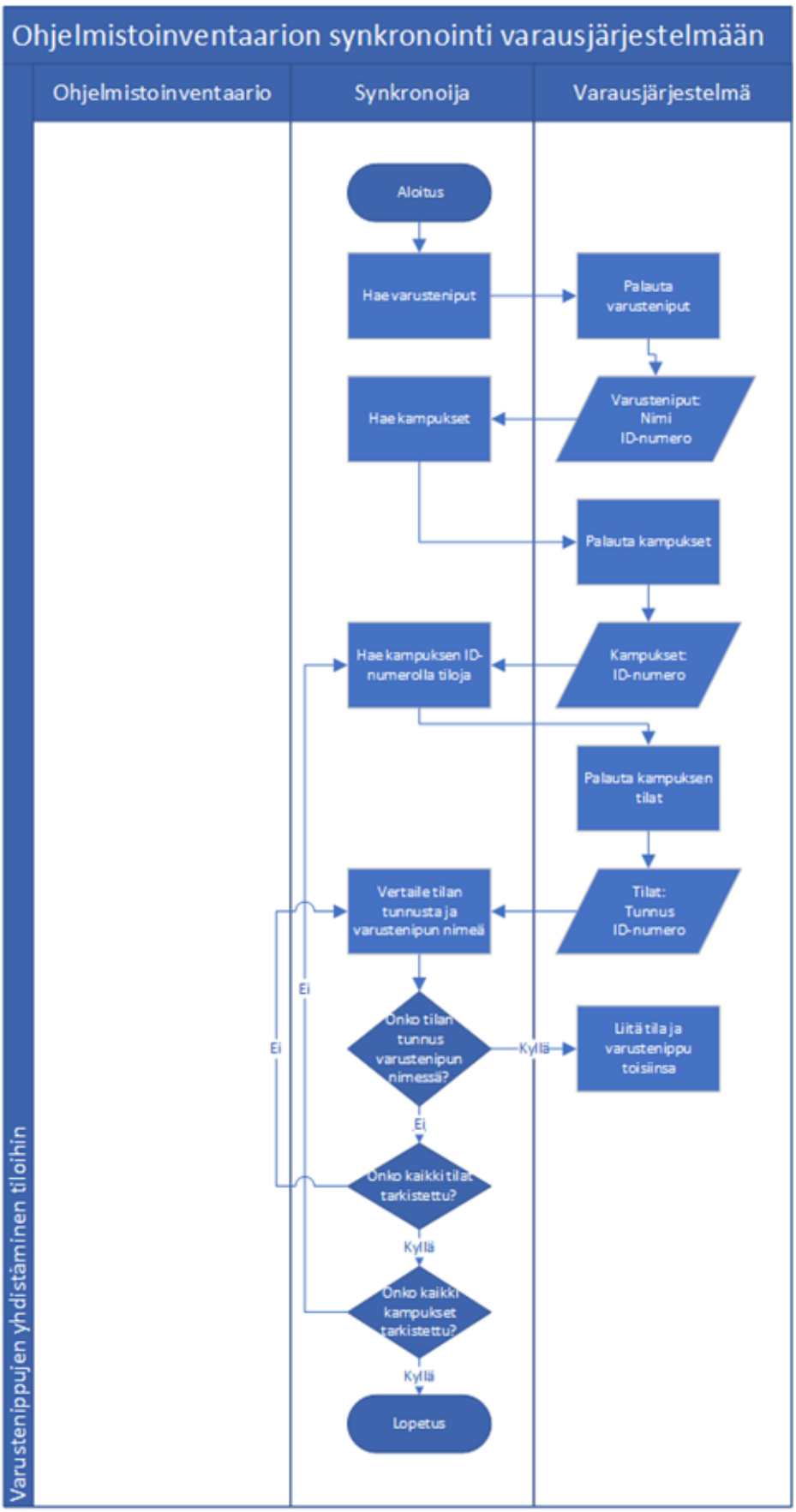
Kaava 2 kuvattu vaihe keskittyy varustenippujen sisällön päivittämiseen. Tässä vaiheessa tarvitaan ohjelmistoinventaariosta listaa tiloista, joissa on ohjelmia,

listaa ohjelmista, joita on tiloissa ja varustenippujen tietoja rajapinnan yli varausjärjestelmästä. Vaiheessa verrataan ohjelmistoinventaariosta saatuja tiloja ja varustenippuja. Jonka jälkeen tilaan merkityt ohjelmat synkronoidaan varustenippuun rajapinnan yli, jos tila ja varusteniput täsmäävät. Vanha sisältö varustenipusta pyyhitään pois varmistaen.



Kaava 2 Varustenippujen sisällön täyttäminen

Kaava 3 esitetty kolmas vaihe varustenippujen yhdistäminen tiloihin tarvitsee toimiakseen rajapintoja varausjärjestelmään. Rajapintojen yli verrataan varustenippujen nimiä ja tilojen tunnuksia. Mikäli varustenipun nimi täsmää tilan tilatunnukseen yhdistetään resurssit toisiinsa.



Kaava 3 Varustenippujen tiloihin yhdistämisprosessi

Kaikkien näiden vaiheiden suunnittelu ja hahmottaminen tehtiin Microsoft Vision vuokaaviotyökalulla. Vuokaaviot tarjosivat visuaalisen tavan hahmottaa ja selittää ohjelmiston eri vaiheet sekä niiden väliset yhteydet ja riippuvuudet. Ne auttoivat ymmärtämään järjestelmän toimintaa ja tekemään tarvittavat muutokset suunnitelmiin ennen varsinaista ohjelmointia.

4.2 Ohjelmointi

Synkronointiohjelma toteutettiin ohjelmoimalla Python-ohjelmointikielellä IntelliJ IDEA kehitysympäristössä kaikki kolme suunniteltua vaihetta omina tiedostoinaan ja neljäntenä tiedostona yhteenvetotiedosto, joka käynnistää muut tiedostot järjestyksessä. Ensimmäisenä ohjelmointiin varustenippujen luonti ohjelmistoinventaarion tilaluettelon perusteella. Sen jälkeen Varustenippujen sisällön täyttö ohjelmistoinventaarion ohjelmilla tilanumeron perusteella ja kolmantena varustenippujen yhdistäminen tiloihin.

Kaikkien vaiheiden ohjelmissa ladataan ensimmäisenä liitännäiset ja ohjelmat esimerkkikoodin 2 mukaan. Liitännäisiä olivat pyodbc, requests, dotenv, xml.dom.minidom ja os. Pyodbc on Pythonin tietokantaohjaimen liitännäinen, joka mahdollistaa yhteyden luomisen tietokantoihin. Requests joka tarjoaa yksinkertaisen tavan tehdä pyyntöjä palvelusta toiseen ja käsitellä vastauksia eli tehdä esimerkiksi rajapintakutsuja. Dotenv ympäristömuuttujien lataamista varten. Xml.dom.minidom XML-tiedostojen käsittelyyn ja os tiedostojen hallintaan. Os liitännäisellä esimerkiksi käynnistetään tiedostot järjestyksessä. Esimerkkikoodissa on myös risuaidalla merkittyjä koodissa olevia kommentteja, jotka auttavat selventämään koodin tarkoitusta ja toimintaa ihmiselle. Ne voivat muun muassa auttaa muita ohjelmoijia ymmärtämään koodia helpommin ja nopeammin.

```
#Ladataan liitännäiset
import pyodbc
import requests
import dotenv
import xml.dom.minidom
import os

#Ladataan ympäristömuuttujat
load_dotenv()
```

Esimerkkikoodi 2. Liitännäisien lataaminen

Esimerkkikoodissa 3 haetaan Pepin soap-rajapintaa hyödyntäen kaikki varusteniput muuttujaan ”varusteniput”. Soap-kutsu on XML-kielimuodossa eli se on rakennettu kerroksiin tageittain <tagi></tagi>. Muut tarvittavat soap-kutsut noudattavat samoja periaatteita.

```
url = os.getenv("PepinReservationAdminRajapinta")

soapKutsuVarustenippujenHakemiseen = '''
<soapenv:Envelope>
    <soapenv:Header/>
    <soapenv:Body>
        <end:getAllAccessoryBundles/>
    </soapenv:Body>
</soapenv:Envelope>
   '''

headers = {
   'Content-Type': 'text/xml; charset=utf-8',
   'username': os.getenv("PeppiUsername")
}

varusteniput = requests.request("POST", url, headers=headers, data=
soapKutsuVarustenippujenHakemiseen).text
```

Esimerkkikoodi 3. Soap APIn käyttäminen

Esimerkkikoodissa 4 ohjelma tekee SQL-kyselyn ohjelmistoinventaarioon tilanumeroiden hakemiseen. Samantyyppisellä kysellyllä haetaan myös tiloissa olevat ohjelmat.

```
driver = os.getenv('ohjelmistoinventaarionAjuri')
server = os.getenv('ohjelmistoinventaarionPalvelin')
database = os.getenv("ohjelmistoinventaarionTietokanta")
username = os.getenv("ohjelmistoinventaarionKayttaja")
password = os.getenv("ohjelmistoinventaarionSalasana")


cnxn = pyodbc.connect('DRIVER={'+driver+'};SERVER='+server+';DATA-
BASE='+database+';UID='+username+';PWD='+ password)
cursor = cnxn.cursor()

cursor.execute("select distinct [RoomName] FROM [PC_DB].[dbo].[RoomP-
rogList];")
```

Esimerkkikoodi 4. SQL-kyselyn tekeminen

Esimerkkikoodissa 5 näytetään search_word yksinkertainen metodi, jota käytetään koodissa varustenippujen nimien esimerkiksi AR304 Ohjelmat vertailuun tilanumeroon AR304. Käytännössä ohjelma vertailee kahta merkkijonoa keskenään ja palauttaa totuusarvon tosi, jos tilanumero puuttuu vertailukohteesta eli varustenipun nimestä.

```
def search_word(word, string):
   for i in range(len(string)-len(word)+1):
      if string[i:i+len(word)] == word:
         return False
   return True
```

Esimerkkikoodi 5. search_word metodi

Soveltamalla esimerkkikoodien mukaisia koodeja voidaan muodostaa suunnitteluvaiheessa suunnitellut osiot. Esimerkkikoodien käyttö suunnitteluvaiheessa mahdollistaa suunniteltujen osioiden luomisen helposti ja tehokkaasti. Koodien soveltaminen säästää aikaa ja vaivaa, kun suunnitelmat voidaan toteuttaa suoraan käyttäen valmiita koodipohjia.

4.3 Testaus

Ohjelmistontestaaminen on prosessi, jossa ohjelmistosovelluksissa ja -tuotteissa tarkistetaan vikoja ja virheitä niiden toiminnan varmistamiseksi sekä varmistetaan ohjelmiston luotettavuus. Varhaisessa vaiheessa testaamisen aloittaminen mahdollistaa virheiden nopean havaitsemisen ja korjaamisen ennen kuin ne vaikuttavat koko järjestelmään. Näiden ongelmien korjaaminen myöhemmissä kehitysvaiheissa voi olla paljon kalliimpaa ja aikaa vievää kuin varhaisessa vaiheessa havaittujen virheiden korjaaminen. (Draganjac 2022)

Testaaminen tässä kehitystyössä aloitettiin määrittelemällä testitapauksia Taulukko 5, jotka kattaisivat eri toiminnallisuuksia ja skenaarioita, joita haluttiin testata. Testin nimisarakkeessa on testi tiivistettynä otsikkotasolle, kuvaussarakkeessa selitettiin mitä kyseinen testi vaati ja hyväksytty lopputulos sarakkeessa millaisia tuloksia odotettiin. Tämä testausprosessi jatkui, kunnes kaikki testitapaukset oli suoritettu ja ohjelmisto toimi halutulla tavalla kaikkien testien osalta. Ohjelmiston koodia päivitettiin tarpeen mukaan, ja testitaulukkoa käytettiin jatkuvasti päivittämään ja seuraamaan testauksen etenemistä.

Testin nimi	**Testin kuvaus**	**Hyväksytty lopputulos**
Varustenipun luonti A	Synkronoidaan varusteniput.	Varausjärjestelmän varustenippunäkymään ilmestyy uudet varusteniput.

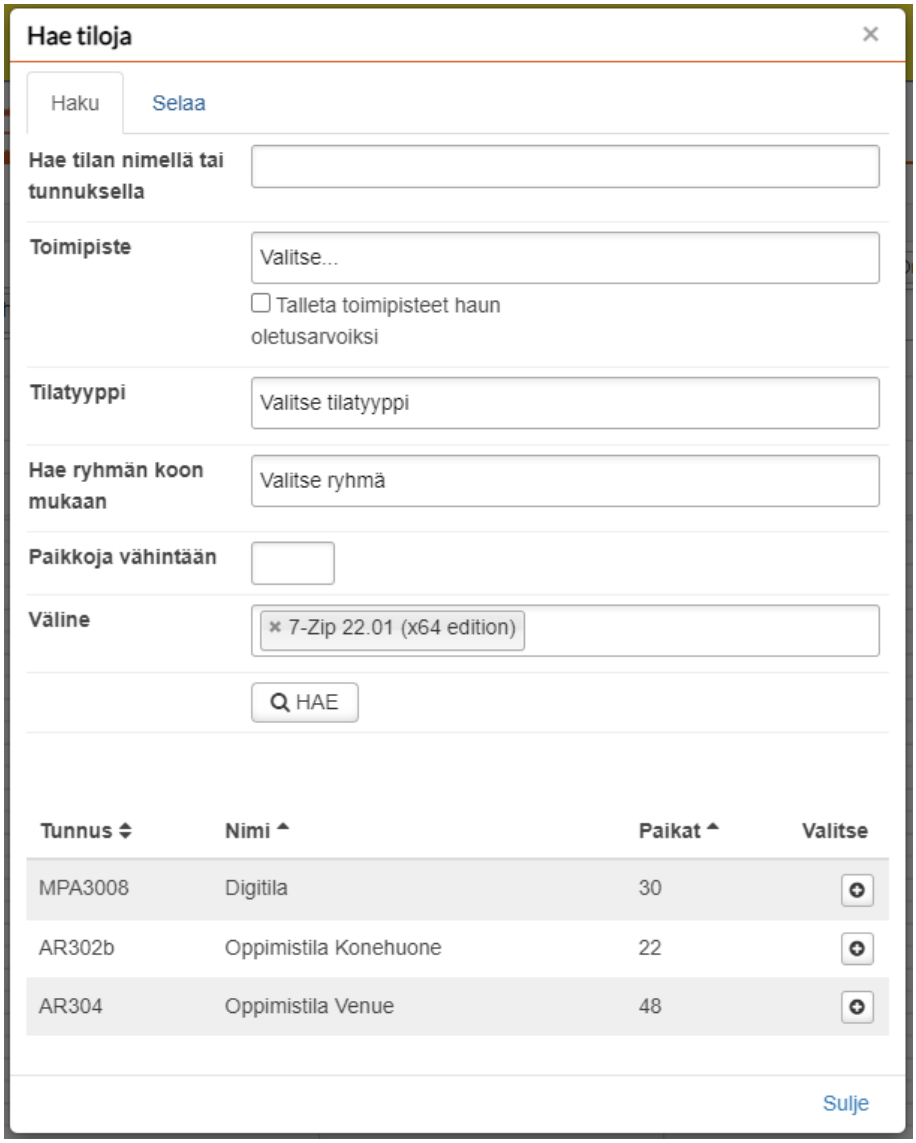
Kaksoiskappaleiden tarkistus	Synkronoidaan tiloille varusteniput toistamiseen.	Varausjärjestelmän varustenippunäkymään ei ilmestyy uudet varusteniput.
Varustenipun luonti B	Poistetaan yksi tai useampi varustenippu ja synkronoidaan tiloille varusteniput toistamiseen.	Varausjärjestelmän varustenippunäkymään ilmestyy puuttuvien tiloille uudet varusteniput.
Varustenipun täyttö	Päivitetään tyhjien varustenippujen sisältö.	Varustenipun sisälle tulee ohjelmat.
Kaksoiskappaleiden tarkistus	Päivitetään tyhjien varustenippujen sisältö toistamiseen.	Varustenipun sisälle ei tule lisää ohjelmia.
Varustenipun päivitys	Poistetaan muutama ohjelma varustenipusta. Päivitetään varustenippujen sisältö toistamiseen.	Varustenipun sisälle tulee puuttuvat ohjelmat.
Varustenipun liitos tilaan	Liitetään varusteniput tiloihin, jos varustenipussa ja tilassa on sama tilanumero.	Liitokset onnistuvat ja tilat näkyvät varustenipuissa.

Kaksoiskappaleiden tarkistus	Liitetään varusteniput tiloihin uudelleen, jos varustenipussa ja tilassa on sama tilanumero.	Tilat eivät näy kahteen kertaan varustenipuissa.
Tilan haku välineellä Pepissä	Haetaan kalenteriin näkyväksi tiloja tietyllä välineellä.	Ohjelmalla varustetut tilat tulevat näkyviin.
Tilan haku välineellä varaukseen	Haetaan varaukseen tiloja tietyllä välineellä.	Ohjelmalla varustetut tilat tulevat näkyviin.
Tilan haku välineellä gantt-näkymässä	Haetaan tila näkyväksi gantt-kalenteriin. Tunnetaan myös nimellä vapaan tilan haku.	Ohjelmalla varustetut tilat tulevat näkyviin.

Taulukko 5 Testitaulukko

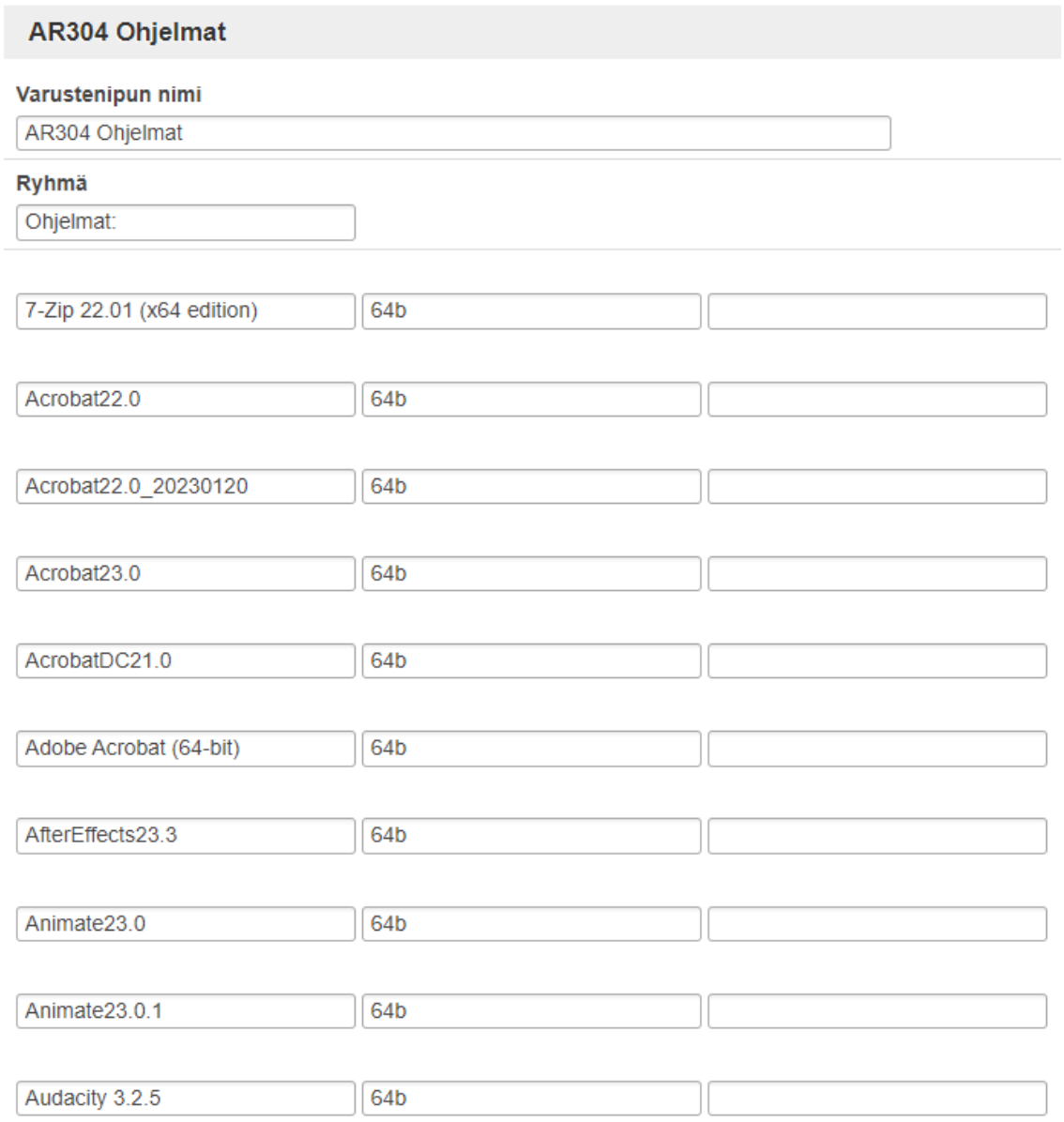
5 Kehittämiskohteen tulokset tai löydökset

Tietokoneohjelmien eli ohjelmistoinventaarion synkronointi opintohallintajärjestelmän Pepin resurssi- ja varausjärjestelmään onnistui tarkoituksellisesti. Tämä mahdollisti tilojen hakemisen ja suodatuksen järjestelmässä ajantasaisien asennettujen ohjelmien perusteella Kuva 8 mukaisesti jo järjestelmässä olevilla välinehaku ominaisuuksilla. Nyt käyttäjät voivat etsiä vaivattomasti tiloja tietokoneohjelmien perusteella varausjärjestelmästä ja löytää helposti heidän tarpeitaan vastaavia tiloja.



Kuva 8 Kuvakaappaus tilojen hakemisesta 7-Zip ohjelmalla

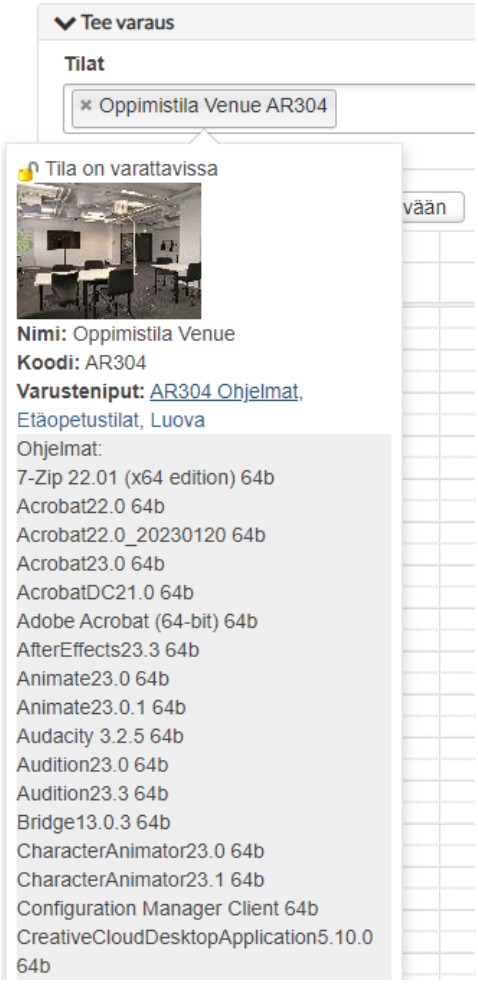
Myös varustenippuja tarkastellessa hallintaosiossa tietokoneohjelmat näkyvät ryhmiteltynä selkeästi ja siististi kuten Kuva 9 voidaan nähdä. Lisäksi ohjelmat on eritelty bittiversioiden mukaan, jolloin käyttäjät voivat valita tarvittavan version käyttöjärjestelmänsä mukaan. Bittiversion lisäksi voidaan hyödyntää myös tyhjäksi jäänyttä lisätietokenttää tulvaisuudessa. Kenttään voidaan kirjata esimerkiksi lyhyt kuvaus tietokoneohjelmasta tai poistoaika.



Kuva 9 Synkronoitu varustenippu

Tietokoneohjelmat näkyvät suunnitellusti varustenipuissa Kuva 10 mukaan mikä tarkoittaa, että ohjelmat vievät vähemmän tilaa näytöllä, mikä auttaa säästämään arvokasta näyttötilaa erityisesti silloin, kun käytössä on rajoitetusti näyttötilaa. Kun käyttäjä haluaa tarkastella tai avata ohjelmalistan, hän voi napsauttaa hiirellä varustenippua. Tällöin varustenippu aukeaa ja ohjelmat näytetään käyttäjälle Kuva 11 tavoin. Tämä mahdollistaa ohjelmien etsimisen ja selaamisen tarpeen mukaan.

Kuva 10 Ohjelmat-varustenippu kiinni



Kuva 11 Ohjelmat-varustenippu auki

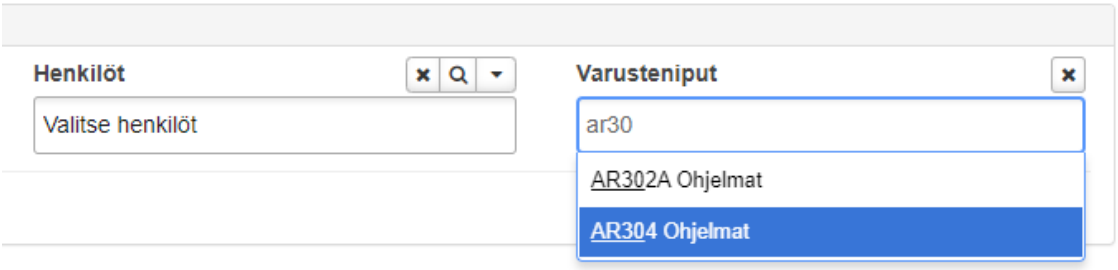
Kaiken kaikkiaan tietokoneohjelmien synkronointi varausjärjestelmään onnistui erinomaisesti ja tämä parantaa käyttäjien kokemusta tilojen hakemisessa ja tilatietojen tarkastelussa. Tämä kehitys tehostaa varausprosessia, säästää aikaa ja tarjoaa käyttäjille paremman käyttäjäkokemuksen.

Käyttämisen aikana ilmeni kuitenkin ominaisuuksia, joita voisi vielä kehittää ja niistä annettiin palautetta kehitysideoina ja korjausehdotuksina eteenpäin Peppi-järjestelmän kehitykseen. Kehityskohteita olivat muun muassa tilahaun välinekohdan yhdenmukaistaminen tai harmonisointi eri kalenterinäkymien välillä.

Haun kehittäminen monipuolisemmaksi ja haettavan välineen ennakoinnin parantaminen.

5.1 Haun harmonisointi eri näkymissä

Kun käytetään samanlaisia hakuominaisuuksia, kuten hakukenttiä, suodattimia ja hakunappeja, eri osissa järjestelmää, käyttäjät voivat intuitiivisesti ymmärtää niiden toiminnan ja käyttötavan. Varausjärjestelmän gantt-näkymässä kuitenkin välinehaun tilalla on varustenippuhaku, joka on esillä kuvassa 12 . Varustenippuhaku hakee ylätasolta varustenippujen nimiä eikä niiden sisällä olevia välineitä kuten muiden kalenterien haku. Eli gantt-näkymässä ei saa tiloja suodatettua tietokoneohjelmilla muiden kalenterinäkymien tapaan.

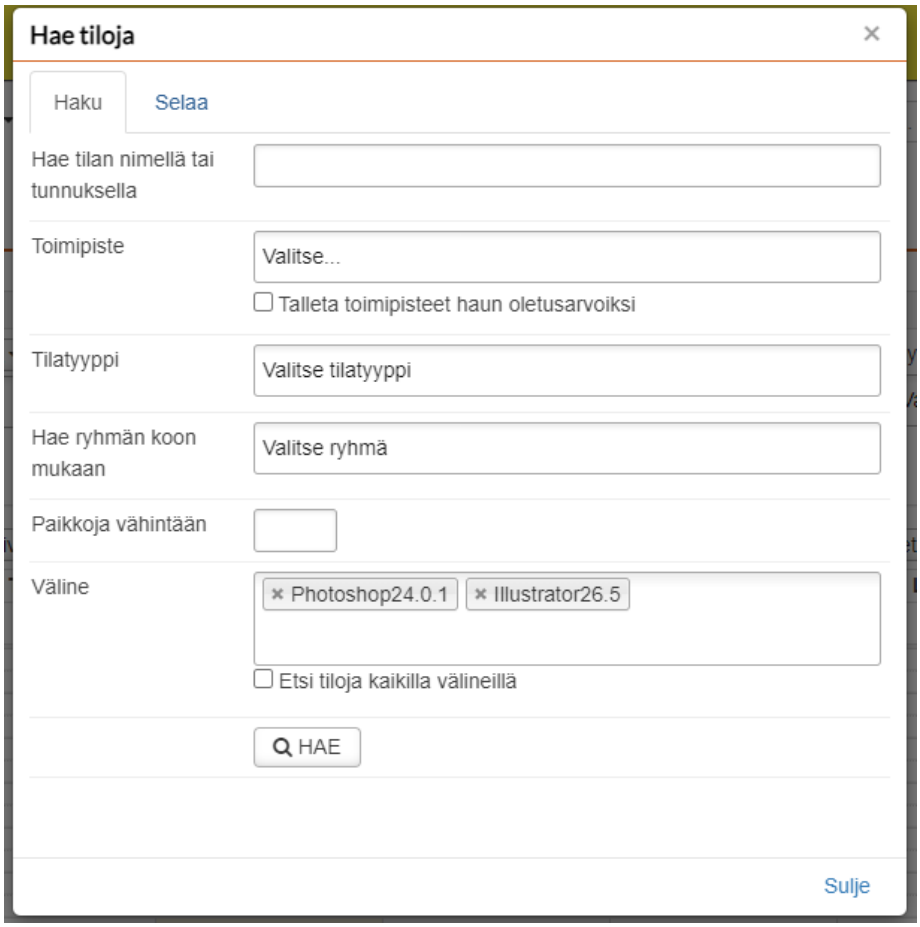


Kuva 12 Kuvakaappaus varausjärjestelmän gantt-näkymän hakukentästä

Kokonaisvaltainen harmonia hakuominaisuuksien välillä helpottaa käyttäjien vuorovaikutusta järjestelmän kanssa ja vähentää mahdollisia sekaannuksia. Kun käyttäjät huomaavat samankaltaisia hakuominaisuuksia eri osissa järjestelmää, he omaksuvat sen nopeammin ja he voivat hyödyntää tätä tietoa sivuston eri osien välillä.

5.2 Haun kehittäminen

Tiloja hakiessa ohjelmien perusteella nousi esille myös kehitystarve haun kehittämisestä, joka vietiin eteenpäin järjestelmän kehityksestä vastaavalle konsortion teemaryhmälle. Tilojen hakulomakkeen välinehaun olisi hyödyllistä sisällyttää myös "AND"-tyyppinen haku "OR"-tyyppisen haun lisäksi. "AND"-haku tarkoittaa sitä, että hakutulosten tulee sisältää kaikki haettavat ohjelmat ja välineet samanaikaisesti, kun taas "OR"-haku palauttaa tuloksia, jotka sisältävät vähintään yhden haetuista ohjelmista. Ominaisuus voisi olla käyttöön otettava optio hakukentän alapuolella Kuva 13 kuvamuokkauksen tapaan.



Kuva 13 Kuvitettu kuvamuokkaus tilojen hakulomakkeesta

Kun käytetään tilojen hakulomaketta, voi olla tilanteita, joissa käyttäjä haluaa rajata hakutuloksia entisestään yhdistämällä useita hakuehtoja. Esimerkiksi, jos

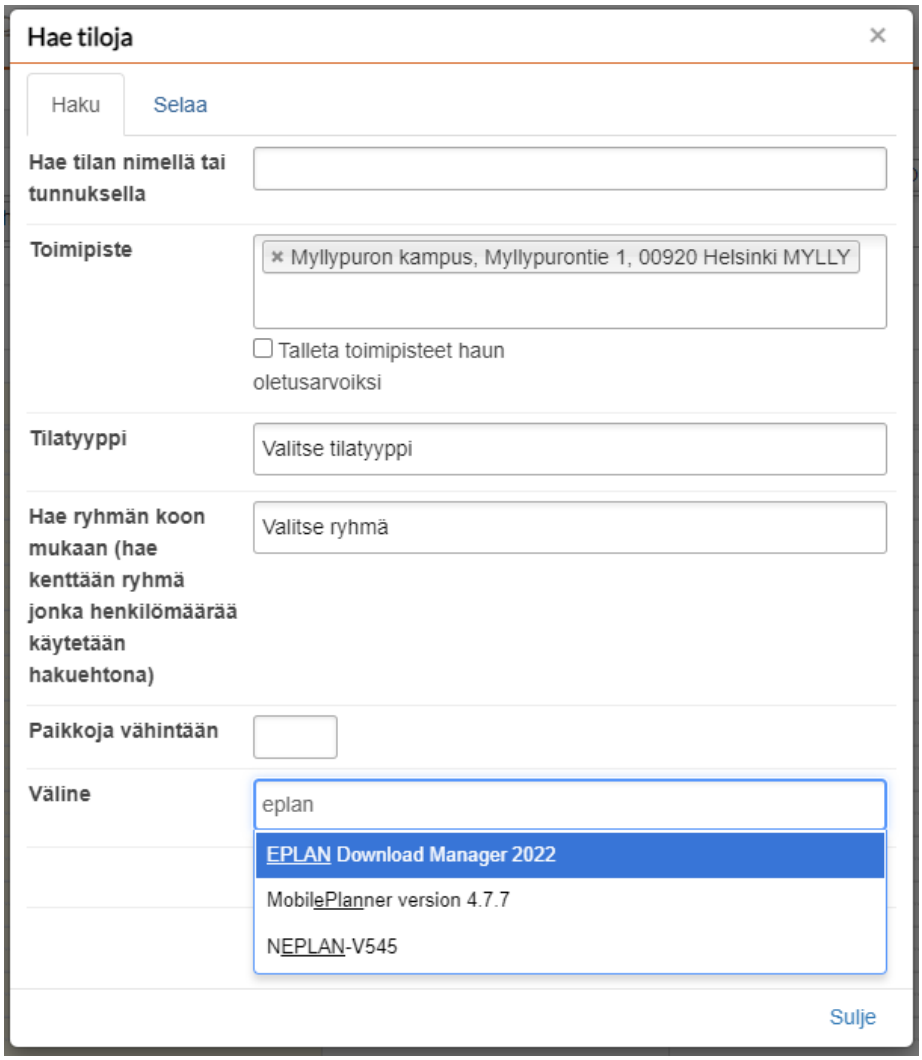
henkilö etsii tilaa kuvamuokkausohjelmalla, mutta haluaa samalla varmistaa, että tiloissa on myös tarjolla tarvittava ohjelma kuvan viemiseen verkkoon, "and"-haku mahdollistaisi näiden kahden ehdot täyttävien tilojen löytämisen. Hakutulokset rajautuisivat vain niihin tiloihin, joissa molemmat ehdot täyttyvät samanaikaisesti.

Toisaalta "OR"-haku on hyödyllinen silloin, kun käyttäjä haluaa laajentaa hakuaan ja löytää tiloja, jotka täyttävät vähintään yhden annetun hakuehdon. Esimerkiksi, jos henkilö etsii tiloja, joissa voi olla mikä tahansa versio jostain ohjelmasta, "OR"-haku palauttaisi tilat, jotka täyttävät minkä tahansa ehdotetuista vaihtoehdoista.

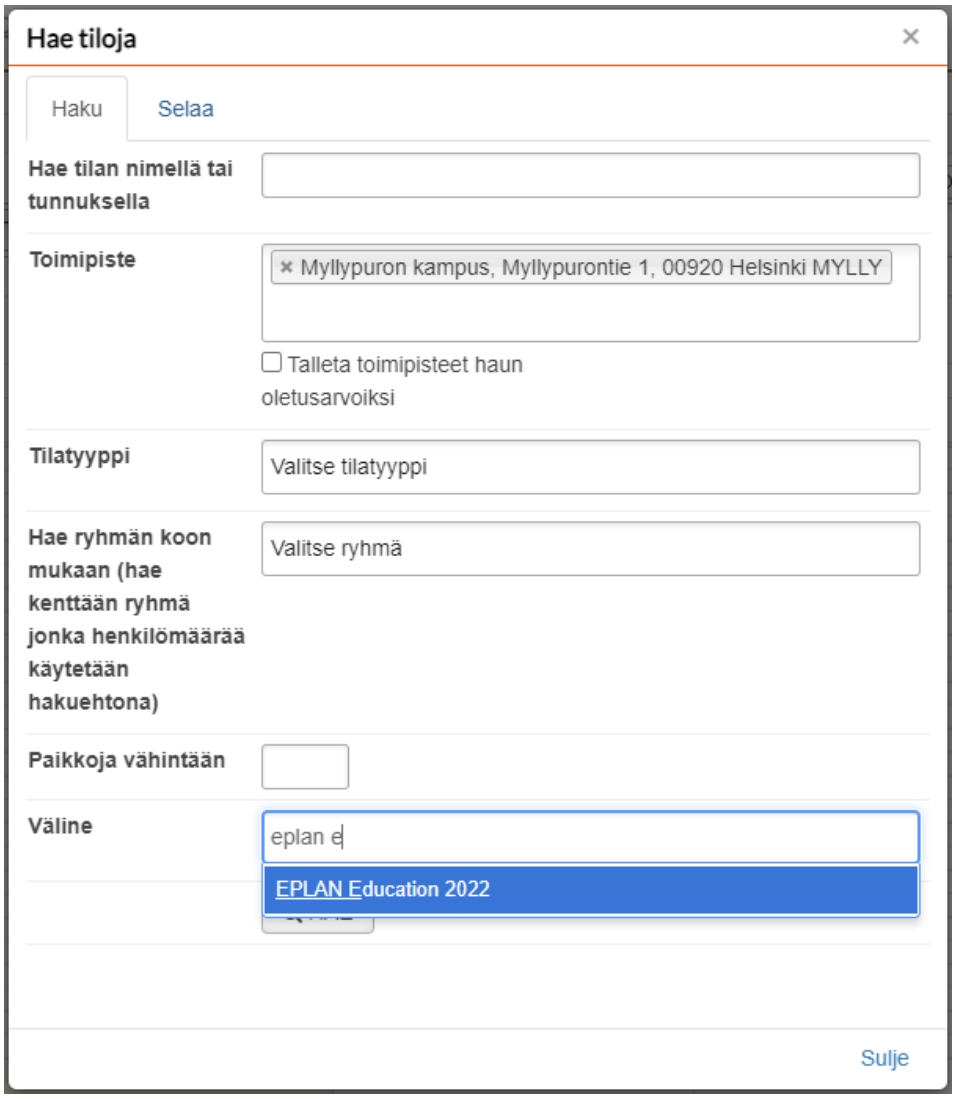
Sisällyttämällä sekä "AND"- että "OR"-tyyppiset haut tilojen hakulomakkeessa, käyttäjille tarjotaan joustavuutta ja tarkkuutta heidän hakuehtojensa mukaisesti. Tämä auttaa parantamaan hakukokemusta ja varmistaa, että käyttäjät löytävät juuri heidän tarpeisiinsa sopivat tilat.

5.3 Haun epäsäännöllisyys

Käytössä huomattiin epäsäännöllisyyksiä, kun tilojen haku ominaisuuteen liittyen tuli palautetta Metropolian henkilökunnalta. Hakua käyttäessä esimerkiksi hakusanalla "eplan", haku ei palauttanut odotettua tulosta. Ennakoiva haku on yleensä tehokas tapa löytää relevanttia tietoa, kun ei tiedetä tarkkaan mitä haetaan. Tässä tapauksessa se ei kuitenkaan toiminut odotetulla tavalla. Hakusanalla "eplan" olisi pitänyt tulla kaikki ohjelmistot, joissa sana esiintyy. Hakutulokset eivät kuitenkaan sisältäneet tätä ohjelmaa tai siihen liittyvää tietoa kuten Kuva 14 nähdään. Kun hakusanaan lisäsi loppuun muutaman kirjaimen lisää "eplan e", palautta ennakoivahaku halutun tuloksen Kuva 15 mukaisesti.



Kuva 14 Kuvakaappaus tilahaun ennakoinnista hakusanalla eplan



Kuva 15 Kuvakaappaus tilahaun ennakoinnista hakusanalla eplan e

Tämä esimerkki osoittaa, että ennakoiva haku ei aina tuota odotettuja tuloksia, vaikka käytettyjä hakusanoja voisi pitää suhteellisen tarkkoina. Erilaiset tekijät voivat vaikuttaa hakutulosten sisältöön, kuten hakukoneen indeksointisäännöt ja käytetyt hakusanat. Esimerkin perusteella tehtiin virheenkorjauspyyntö tuotekehittäjälle, koska on tärkeää käyttäjän kokemuksen kannalta, että ennakoiva haku palauttaa odotetun tuloksen aina mahdollisimman nopeasti ja tarkasti mahdollisuuksien mukaan. Muuten tämä saattaa johtaa väärinymmärrykseen. Esimerkiksi käyttäjä saattaa luulla, että ohjelmaa ei ole missään tilassa asennettuna.

6 Yhteenveto

Kehitystyön aikana luotiin Metropolia Ammattikorkeakoulun toimeksiantajalle python-ohjelmointikielellä ohjelma, joka synkronoi tietokoneohjelmien tiedot ohjelmistoinventaariosta Peppi-opintohallintojärjestelmän resurssi- ja varausjärjestelmään. Työ toteutettiin, jotta olemassa olevaa tietoa ohjelmista voidaan hyödyntää tehokkaammin suoraan käyttäjien toimesta.

Varausjärjestelmässä tietokoneohjelmien tiedot näytetään varustenippujen muodossa, mikä mahdollistaa selkeän ryhmittelyn eikä pitkä lista ohjelmista tuki käyttäjän ruutua. Käyttäjät voivat hyödyntää näitä varustenippuja hakuehtoina, kun he etsivät opetukselle tai itsenäiselle opiskelulle sopivaa tilaa. Tämä mahdollistaa nopean ja kohdennetun tilanhaun, jossa otetaan huomioon ohjelmistovaatimukset. Voidaankin todeta tietokoneohjelmien haun olevan kysyntä eikä tarjontaperusteista.

Kehitystyön aikana paljastui myös useita epäkohtia, kuten epäsäännöllisyyksiä hakukentän ennakoinnissa ja eroavia toiminnallisuuksia eri näkymissä. Näiden havaintojen perusteella tehtiin kehitysehdotuksia tilanhaun parantamiseksi. Kehitysehdotuksen tavoitteena oli korjata havaitut epäsäännöllisyydet ja parantaa hakutoimintojen toimivuutta ja käyttäjäystävällisyyttä.

Kehitystyön tulokset tarjosivat toimeksiantajalle toimivan ohjelman, joka synkronoi tietokoneohjelmien tiedot varausjärjestelmään. Lisäksi ne paljastivat kehitettäviä konkreettisia parannusehdotuksia tilanhaun optimoimiseksi. Tämä kokonaisuus auttoi Metropolia Ammattikorkeakoulua parantamaan resurssien hallintaa ja tarjoamaan käyttäjilleen parempia työkaluja tilojen varaamiseen ja käyttämiseen.

Lähteet

Ben, Rubenstein. 2020. Senior Manager. TechTarget. Microsoft System Center Configuration Manager (SCCM). https://www.techtarget.com/searchwindowsserver/definition/Microsoft-System-Center-Configuration-Manager-2012
Luettu: 12.6.2023

Heimo, Järkkä. 2023. Järjestelmäsuunnittelija, Metropolia Ammattikorkeakoulu, Helsinki. Keskustelu 9.6.2023.

Katarina Draganjac 16.3.2022 The Importance of Software Testing. Collective mind development. https://collectivemind.dev/blog/the-importance-of-software-testing Luettu 13.6.2023

Metropolia Ammattikorkeakoulu 2023. Metropolia Ammattikorkeakoulu. Kun haluat ratkaisijaksi. https://www.metropolia.fi/fi/metropoliasta Luettu 15.5.2023

Metropolia 2023. Metropolia Ammattikorkeakoulu. Organisaatio ja strategia https://www.metropolia.fi/fi/metropoliasta/organisaatio-ja-strategia
Luettu 14.6.2023