Janne Anoschkin

TIETOKANTASOVELLUKSEN KEHITTÄMINEN SUUNNITTELUPROSESSIN TEHOSTAMISEEN

Automaatiotekniikan koulutusohjelma

2019

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

TIETOKANTASOVELLUKSEN KEHITTÄMINEN SUUNNITTELUPROSESSIN TEHOSTAMISEEN

Anoschkin, Janne
Satakunnan ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma
Kesäkuu 2019
Sivumäärä: 30

Asiasanat: Tietokanta, SQL, Käyttöliittymä, EAV, Relaatiotietokanta

---

Tämän opinnäytetyön tarkoitus oli suunnitella ja toteuttaa Piir-Group Oy:lle tietokantasovellus sekä web-käyttöliittymä, jotka tehostavat suunnitteluprosessia.

Opinnäytetyössä esittelen sovelluksen kehitysprosessin ja lopputuloksena syntyneen valmiin sovelluksen.

Avainasemaan opinnäytetyön teon aikana nousi kahden erilaisen tietokantarakenteen, relaatio- ja EAV-mallin, vertailu.

DATABASE APPLICATION FOR IMPROVING PLANNING PROCESS

Anoschkin, Janne
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in automation engineering
June 2019
Number of pages: 30
Appendices:

Keywords: Database, SQL, HMI, EAV, Relational database

The purpose of this thesis was to design and implement a database application for Piirgroup Oy that streamlines their designing process.

In the thesis I introduce the designing process of the application and the resulting finished application.

Comparing the two types of database structures, relational- and EAV-model, became a key feature of this thesis during its making.

SISÄLLYS

1 JOHDANTO

Sähkö- ja automaatiosuunnittelu on aikaa vievä prosessi. Yksi tämän prosessin osista on projektia varten tilattavien komponenttien kerääminen yhdeksi tuoteluetteloksi. Näitä komponentteja on yleensä satoja jokaista projektia kohden, joten tuoteluettelon kokoamisen helpottaminen voi säästää huomattavasti aikaa.

PIIR-GROUP Oy on Ulvilassa toimiva vuonna 1991 perustettu automaatiotalo, joka tarjoaa kokonaisvaltaista projektiosaamista ja ratkaisuja teollisuuden sähkö ja automaatiosuunnitteluun. Opinnäytetyön tarve syntyi, koska PIIR-GROUP Oy:llä oli tarve tehostaa suunnitteluprosessiaan.

Tämän opinnäytetyön tarkoituksena oli tehdä tietokantasovellus, jonka avulla suunnittelija kykenisi luomaan projektinsa tuoteluettelon entistä nopeammin. Sovelluksen oli tarkoitus myös avustaa suunnittelijaa tuotteiden valinnassa sekä ehkäistä mahdollisia inhimillisiä virheitä tuotteiden valinnassa.

Tämä opinnäytetyö esittelee PIIR-GROUP Oy:lle tehdyn tietokannan ja käyttöliittymän suunnittelu- ja toteutusprosessin, sekä perustelee prosessin aikana tehtyjä ratkaisuja.

Luvussa 2 esitellään työn tausta. Se esittelee yrityksen toimintatavat ennen tässä opinnäytetyössä tehdyn sovelluksen käyttöönottoa sekä tavoitteet tässä opinnäytetyössä tehtävälle sovellukselle. Luku 3 esittelee tämän opinnäytetyön aikana käytettyjä työkaluja, kuten sovelluksia ja ohjelmointikieliä. Luku 4 esittelee kokonaisuudessaan tässä opinnäytetyössä tehdyn tietokannan ja käyttöliittymän. Se myös selittää, miten tehtyihin ratkaisuihin päädyttiin. Luvussa 5 on pohdintaa työn lopputuloksesta.

2 TYÖN TAUSTA

2.1 Toimeksiantajan toimintatavat

PIIR-GROUP Oy suunnittelee ja valmistaa sähkökeskuksia. Yrityksessä suunnittelija piirtää projektin sähkökuvat sekä luo jokaiselle projektille komponenttiluettelon tilattavista komponenteista. Tämä on aikaa vievä prosessi, sillä suunnittelijan on tähän asti täytynyt hakea jokainen komponentti ja siitä tarvittavat tiedot erikseen ja kopioitava ne luetteloon.

Yrityksellä on ollut käytössään Excel-taulukko, vakiokomponenttiluettelo, jossa on tallennettuna tarvittavat tiedot useimmin käytetyistä komponenteista. Taulukon käyttäminen on kuitenkin ollut hidasta, sillä siinä ei ole kunnollista hakemistoa ja kopioidessa tulee helposti tehtyä virheitä.

2.2 Tavoite

Piir-Group Oy tarvitsi sovellusratkaisun, jonka avulla suunnittelijan olisi helppo valita projektiin tarvitsemansa komponentit ja luoda komponenttiluettelo projektille. Tämä päätettiin toteuttaa tietokantasovelluksena.

Tietokanta ja sen käyttöliittymä tuli suunnitella niin, että se olisi yhteensopiva nykyisen vakiokomponenttiluettelon sekä projektien komponenttiluetteloiden kanssa. Näin kyettiin helposti hyödyntämään vakiokomponenttiluettelossa jo olemassa olevaa dataa myös tietokannassa. Sovelluksesta tuli olla mahdollisimman helppo siirtää sen antama taulukko projektin komponenttiluetteloon.

Avaintavoitteena oli myös se, että sovellus olisi ”älykäs”, eli se pystyisi jollakin määrin avustamaan suunnittelijaa komponenttien valinnassa. Esimerkiksi jotkin komponentit vaativat lisäkomponentteja, jolloin sovellus osaa ehdottaa käyttäjälle tarvittavia komponentteja.

Opinnäytetyö päätettiin rajata koskemaan ainoastaan Piir-Groupin käyttämiä Siemensin logiikkakomponentteja, mutta se tuli rakentaa niin, että tietokannan ja käyttöliittymän laajentaminen kaikkien valmistajien komponentteihin tapahtuisi vaivattomasti.

3 KÄYTETTÄVÄT TYÖKALUT

3.1 IIS

Internet Information Servises (IIS) on Microsoftin kehittämä verkkopalvelinohjelmisto (Microsoft, 2019). Tässä opinnäytetyössä kehitettävä tietokanta tulee lopulta toimimaan IIS palvelimella.

3.2 XAMPP

XAMPP on Apache Friendsin kehittämä ilmainen ohjelmistopaketti, joka sisältää Apache web serverin, MariaDB:n ja PHP:n. Paketin tarkoitus on helpottaa näiden sovellusten asennusta ja käyttöönottoa. (Apache Friends, 2019) XAMPP ympäristöä tullaan käyttämään tämän opinnäytetyön tietokannan kehittämisvaiheessa

3.3 Apache web server

Apache web server on Apache Software Foundationin kehittämä avoimen lähdekoodin palvelinohjelmisto (Apache Friends, 2019). Opinnäytetyön kehityksenvaiheen aikana tietokanta tulee toimimaan Apache web serverillä.

3.4 MySQL

MySQL on avoimen lähdekoodin relaatiotietokantaohjelmisto, jota käyttävät mm. Facebook, Twitter ja YouTube. (Oracle, 2019)

3.5 JavaScript

JavaScript on Netscapen vuonna 1995 kehittämä skriptikieli, jota käytetään web-sivujen muokkaamiseen. Sillä voidaan luoda sivuista interaktiivisempia ja luontevampia esimerkiksi animaatioilla, painikkeilla ja valikoilla. (Mozilla, 2019) Tässä

opinnäytetyössä JavaScriptillä hallitaan web-sivulla olevaa käyttöliittymää, sekä kutsutaan PHP-tiedostoja ja syötetään näiden tiedostojen palauttamia arvoja web-sivulle.

3.6 PHP

Hypertext Preprocessor (PHP) on ohjelmointikieli, joka soveltuu verkkosivujen ohjelmointiin. Toisin kuin JavaScript, PHP suoritetaan palvelimella ja se ainoastaan palauttaa HTML muotoisen vastauksen (The PHP Group, 2019). Tässä opinnäyetyössä PHP:ta käytetään luomaan SQL kielen kyselyitä tietokannalle ja palauttamaan niiden vastauksia HTML muotoisena JavaScript tiedostolle.

3.7 HTML

Hypertext Markup Language (HTML) on verkkosivujen luontiin tarkoitettu ohjelmointikieli. HTML sivuja on mahdollista selata millä tahansa verkkoselaimella. (W3C, 2019)

3.8 Notepad++

Notepad++ on ilmainen tekstieditori, joka tukee monien ohjelmointikielien syntakseja (Notepad++, 2019). Opinnäytetyön ohjelmointi tullaan tekemään Notepad++:lla.

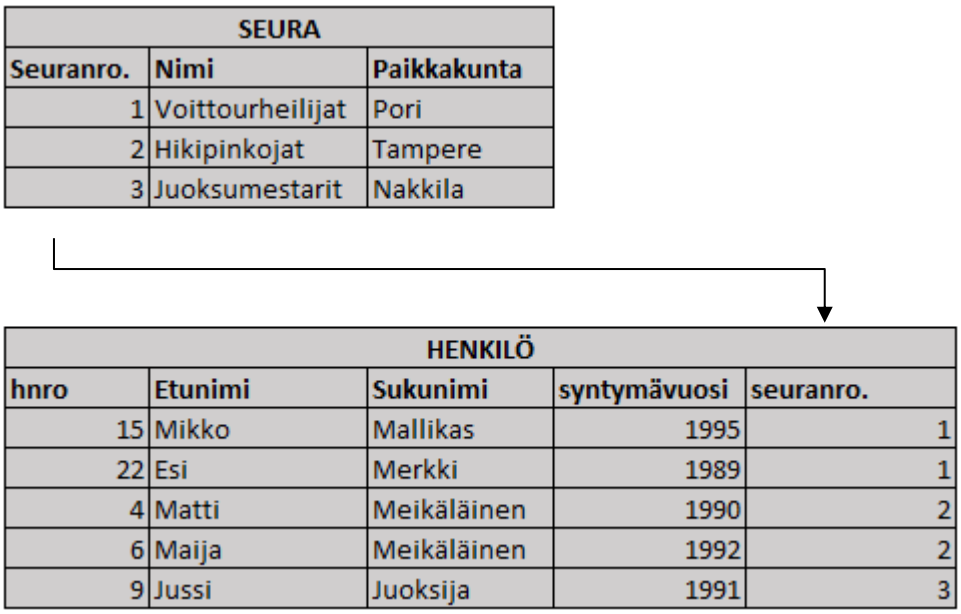
4 SOVELLUKSEN KEHITYSPROSESSI

4.1 Tietokanta

Tietokannalla tarkoitetaan loogisesti yhteenkuuluvien, tallennettujen tietojen joukkoa, jota voidaan helposti käsitellä tietokantakielellä (kuten SQL). (Hovi, 2005)
Kehittäessäni opinnäytetyöni tietokantaa, ryhdyin luomaan sitä relaatiotietokantana, mutta kehitystyön edetessä, päädyin vaihtamaan kannan rakenteen EAV-mallia vastaavaksi. Tässä kappaleessa tulen esittelen tietokannan kehityksen eri vaiheita ja selitän kuinka päädyin tekemiini ratkaisuihin.

4.1.1 Tietokanta yleisesti

Tietokannan peruselementti on taulu. Nämä taulut sisältävät sarakkeita ja rivejä. Jokainen taulun sarakkeista on nimetty yksilöllisesti ja niiden tietojen arvot kuuluvat samaan arvojoukkoon. (Hovi, 2005) Kuvassa 1 on esimerkki kahdesta tietokannan taulusta. SEURA-taulun sarakkeet ovat Seuranro, Nimi ja Paikkakunta.



SEURA		
Seuranro.	**Nimi**	**Paikkakunta**
1	Voittourheilijat	Pori
2	Hikipinkojat	Tampere
3	Juoksumestarit	Nakkila

HENKILÖ				
hnro	**Etunimi**	**Sukunimi**	**syntymävuosi**	**seuranro.**
15	Mikko	Mallikas	1995	1
22	Esi	Merkki	1989	1
4	Matti	Meikäläinen	1990	2
6	Maija	Meikäläinen	1992	2
9	Jussi	Juoksija	1991	3

Kuva 1 Esimerkki kahdesta tietokannan taulusta

Jokaisessa tietokannan taulussa on oltava tunnisteena perusavain. Perusavaimen on oltava yksilöllinen. Tällä tarkoitetaan, että jokaisen sarakkeen arvon on oltava yksilöllinen. (Hovi, 2005) Edellisessä kuvassa HENKILÖ-taulun perusavain on hnro. Jokaiselle taulussa olevalle henkilöllä on yksilöllinen numerotunniste, jota ei voi olla muilla.

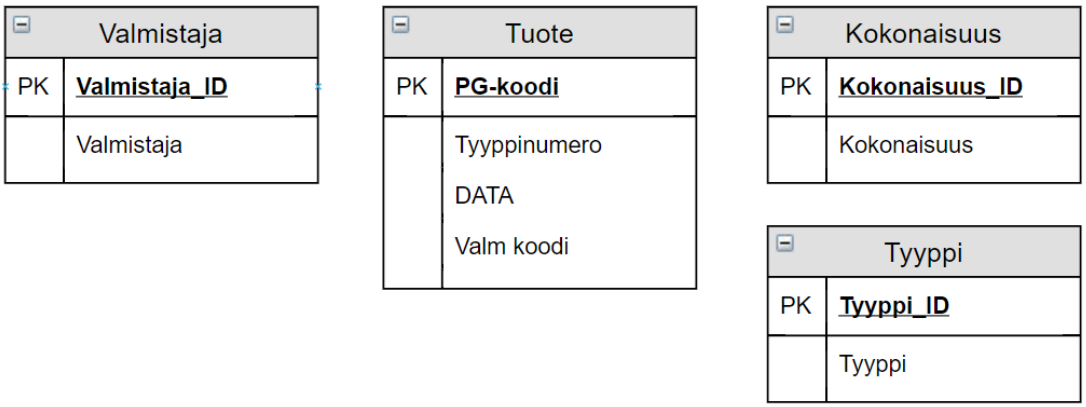
Tietokannan tauluilla voi olla yhteyksiä. Näitä ovat esimerkiksi yksi-yhteen-yhteys, moni-moneen-yhteys ja yksi-moneen-yhteys. Yhteyden esittämiseksi tarvitaan viiteavain. Viiteavaimen sisältävää taulua voidaan kutsua lapsitauluksi ja viitattavaa taulua isätauluksi. (Hovi, 2005) Kuvassa 1 HENKILÖ-taulu sisältää viiteavaimen seuranro. Sillä viitataan SEURA-taulun perusavaimeen. Tämä yhteys on yksi-moneen yhteys. Tässä tapauksessa yhdellä seuralla voi olla monta jäsentä, mutta kukin jäsen voi kuulua vain ja ainoastaan yhteen seuraan.

4.1.2 Käsiteanalyysi

Käsiteanalyysi on usein ensimmäinen vaihe tietokannan suunnittelussa. Sen tavoitteena on kuvata ja havainnollistaa tietokantaan tallennettavia tietoja. Käsiteanalyysin tarkoitus ei ole kuvata tarkasti koko tietokannan toimintaa, vaan sillä pyritään luomaan karkea kuva tietokannan rakenteesta. Käsiteanalyysi on hyvä tehdä yhteistyössä tietokantatuotteen tilaajan kanssa, jolloin heidän toiveensa ja näkemyksensä ovat alusta asti mukana (Hovi, 2005).

Käsiteanalyysin lopputuloksena syntyy käsitemalli. Sen tarkoitus on olla karkea kuvaus tietokannan rakenteesta ja siihen haluttavista asioista. Käsitemallia luodessa ei esimerkiksi tarvitse vielä olla selvillä, mitä tietokantaohjelmistoa tullaan käyttämään. Käsitemallissa tärkeintä on, että sen tekijöillä on yhteinen käsitys sen sisällöstä ja, että jokainen tallennettava tieto löytyy mallista vain ja ainoastaan kerran. (Hovi, 2005)

Tietokannan suunnittelu aloitettiin tapaamisella toimeksiantajan kanssa. Tapaamisen tarkoitus oli yhdessä toimeksiantajan kanssa selvittää mitä tietoa kannan tulisi sisältää, miten sitä olisi paras jäsennellä ja luoda alustava käsitemalli. Kuvassa 2 on esitettynä tapaamisessa luodut alustavat taulut.



Kuva 2.Tietokannan alustavat taulut

Suurella osalla komponentteja on tiettyjä kokonaisuuksia, joihin ne kuuluvat (esimerkiksi Siemensin S7-1500 logiikka). Kokonaisuus tauluun tullaan siis tallentamaan näitä kokonaisuuksia. tietokannassa Jokaisella komponentilla tulee olemaan tuotetyyppi (esimerkiksi Keskusyksikkö tai virtalähde). Nämä tiedot tullaan tallentamaan Tyyppi-tauluun. Valmistaja-tauluun tullaan tallentamaan jokainen valmistaja, jonka komponentteja on tietokannassa.

Tuote-taulun käsitteet ovat PG-koodi, tyyppinumero, DATA ja Valmistajan koodi. PG-koodi on toimeksiantajan oma koodi, jota he käyttävät tilatessaan komponentteja. Tyyppinumero on komponentin tyyppi- tai mallinumero. DATA- käsite sisältää mahdolliset lisätiedot komponentista (esimerkiksi muistikortin koko). Valmistajan koodi on valmistajan käyttämä koodi tuotteesta.

4.1.3 Tarveanalyysi ja normalisointi

Tarveanalyysin tarkoitus on tarkentaa aiemmin luotu käsitemalli yksityiskohtaiselle tasolle. Tarveanalyysiä tehdessä käsitemallia voidaan testata olemassa olevilla tietotarpeilla ja siihen lisätään mahdollisia uusia tarvittavia tietoja tai yhteyksiä. Tarveanalyysin aikana otetaan myös huomioon tulevan sovelluksen käyttöliittymä, näytöt ja raportit. Tarveanalyysin valmistuttua tietokantarakenteen tulisi täyttää kaikki tiedossa olevat tietotarpeet (Hovi, 2005).

Normalisointi on prosessi, jonka aikana tietokannan tietorakenteet optimoidaan paremmiksi. Normalisoinnin tavoitteena on minimoida toistuvat tiedot tauluissa, luoda kannasta helposti päivitettävä ja joustava sekä varmistaa, että tietoja tarvitsee päivittää vain yhteen paikkaan. Normalisointi jaetaan kolmeen eri normaalimuotoon, joista aina seuraava muoto pitää myös sisällään edellisen normaalimuodon (Hovi, 2005).

Ensimmäisessä normaalimuoto saavutetaan, kun tietokanta ei sisällä ainuttakaan toistuvaa ryhmää tai moniarvoista saraketta. Toistuva ryhmä tarkoittaa taulun sisällä olevaa kahta tai useampaa eri saraketta, jotka sisältävät samantyyppistä tietoa (Hovi, 2005). Esimerkiksi, jos jokaisen urheilusuorituksen, jälkeen URHEILIJA tauluun luotaisiin uusi sarake uutta tulosta varten, tauluun syntyisi toistuva ryhmä. Moniarvoisella sarakkeella taas tarkoitetaan yhtä saraketta, johon jokainen urheilusuoritus tallennettaisiin. Niin toistuvat ryhmät, kuten moniarvoiset sarakkeet ovat tietokannan ylläpidon kannalta mahdottomia (Hovi, 2005).

Toisen normaalisäännön mukaan jokaisen taulun sarakkeiden tulee olla funktionaalisesti riippuvaisia taulun perusavaimesta. Käytännössä tämä tarkoittaa sitä, että jokaista taulun perusavaimen arvoa kohti saa taulun muissa sarakkeissa olla korkeintaan yksi arvo. Jos taulu ei toteuta toisen normaalimuodon ehtoja, on sen sarakkeet eroteltava omiksi tauluikseen ja näiden taulujen välille on luotava toimivat yhteydet (Hovi, 2005).

Kolmas normaalimuoto ottaa vielä yhden askeleen pidemmälle toisesta ja vaatii, että jokaisen taulun sarakkeen on oltava riippuvainen vain ja ainoastaan taulun perusavaimesta. (Hovi, 2005)

Tarveanalyysiä ja normalisointia tehdessä nousi esiin suuri ongelma tietokantarakenteen kannalta. Edellisessä kappaleessa esitetyn tuote-taulun DATA-käsite oli saatu suoraan kappaleessa 2 mainitusta vakiokomponenttiluettelosta. Vakiokomponenttiluettelossa tämä datasarake siis sisälsi kaiken datan komponentista, jota suunnittelijan tarvitsi tietää komponentista (jännite, virta, teho jne.). Ihmiselle tällainen esitystapa on helposti luettavissa, mutta relaatiotietokannassa se rikkoisi ensimmäistä normaalimuotoa, sillä tieto ei ole yksilöllistä eikä yksiarvoista.

DATA-käsiteen normalisoinnin ongelmaksi osoittautui erilaisten yksilöllisten käsitteiden suuri määrä, koska jokaisella erilaisella tuotetyypillä olisi myös erityyppistä tietoa, jota kantaan tulisi tallettaa. Tuote-tauluun tulisi luoda jokaista yksilöllistä käsitettä kohden oma sarakkeensa, mutta kuitenkin vain harvalla komponentilla saattaisi olla arvoja kussakin käsiteessä. Tämä johtaisi siihen, että taulussa olisi kymmeniä, ellei jopa satoja, eri sarakkeita, jotka olisivat enimmäkseen tyhjiä. Tuote-tauluun tulisi myös lisätä uusi sarake aina kun tietokantaan tallennettaisiin uusi komponentti, jonka tarvitsemaa tietotyyppiä ei ole vielä olemassa. Kuvassa 3 on esimerkki siitä miltä tuote-taulu voisi näyttää normalisoituna.

PG-koodi	tyyppi	Tyyppinumero	Valm koodi	jännite(V)	virta(A)	teho(W)	mkortin koko(mb)	pituus(mm)	leveys(mm)
111	muistikortti	111	111				2		
222	valaisin	222	222	230	1	230			
333	kotelo	333	333					100	30

Kuva 3. esimerkki tuote-taulusta

Vaikka kuvassa 3 on vain kolme erilaista tuotetyyppiä, niin suuri osa sen tietueista ovat jo tyhjiä, sillä esimerkiksi muistikortilla ei ole muuta tietoa kuin sen koko.
Normalisointiongelma kyettäisiin ratkaisemaan myös hajottamalla tuote-taulu moneksi tauluksi joista jokaisessa olisi vain tuotteita, joilla on samat käsitteet. Tämä kuitenkin tuottaisi suuria ongelmia myöhemmin, kun tietokantaan aloitettaisiin tallentaa tietoja ja luoda kyselyitä, sillä kaikki tuotteet olisivat hajotettuina lukuisiin tauluihin. Ongelmaksi tulisi myös taas se, että jokaista uutta erilaista tuotetta varten olisi luotava uusi taulu tietokantaa.

Kappaleessa 2 tavoitteeksi asetettiin, että tietokantaa tulisi olla mahdollisimman helppo laajentaa, joten kumpikaan edellä esitellyistä vaihtoehdoista ei ollut mahdollisia toteuttaa. Tässä vaiheessa päätin tutkia myös muita vaihtoehtoja kuin relaatiotietokantaratkaisua ja päädyin tutkimaan EAV-tietokantarakennetta.

4.1.4 EAV-malli ja vertailu relaatiotietokantaan

EAV on lyhenne sanoista entity, attribute ja value. EAV-mallin tietokantoja käytetään laajasti potilastietorekistereissä ja Online-katalogeissa. Mallin nimi juontuu sen

tavasta käyttää pääasiallisesti kolmea taulua tallentamaan kaikki tietokantaan syötettävä tieto. Nämä taulut nimetään siis usein nimillä ENTITY, ATTRIBUTE ja ENTITY_ATTRIBUTE_VALUE (Prakash M. Nadkarni, 2019).

ENTITY-taulu on tietokannan taulu, johon tallennetaan tiedot käsiteltävästä asiasta. Se voi esimerkiksi olla asiakas, tuote tai yritys. Yleensä taulun perusavain on jokin yksilöivä tunnistenumero, kuten henkilötunnus tai yksinkertaisesti juokseva ID numerointi. Taulu voi myös sisältää sellaista tietoa, joka on kaikille taulun tiedoille yhtenevää, kuten etu- ja sukunimi jos kyseessä on henkilö (Prakash M. Nadkarni, 2019). Kuvassa 4 on esimerkki ENTITY-taulusta, joka sisältää erilaisia tuotteita.

ENTITY			
Tuote_ID	Nimi	valmistaja	Paino(g)
1	Pallo	Tehdas Oy	200
2	Käsipaino	Paja Oy	5000
3	kynä	Tehdas Oy	30

Kuva 4 Esimerkki ENTITY-taulusta

ATTRIBUTE-taulussa on kaksi saraketta, perusavain, joka on juokseva ID-numero sekä jokaista ID:tä vastaava attribuutti. Se sisältää kaikki sellaiset ENTITY-taulussa olevien asioiden ominaisuudet, jotka eivät ole kaikille yhteneviä. (Prakash M. Nadkarni, 2019) Esimerkiksi tuotteiden tapauksessa näitä voi olla Ympärysmitta, halkaisija, leveys, lukumäärä pakkausta kohden tai Pakkauksen paino. Kuvassa 5 on esimerkki ATTRIBUTE-taulusta.

ATTRIBUTE	
ID	attribuutti
1	Ympärysmitta(cm)
2	halkaisija(cm)
3	leveys(cm)
4	lkm/pakkaus
5	pakkaus paino(g)

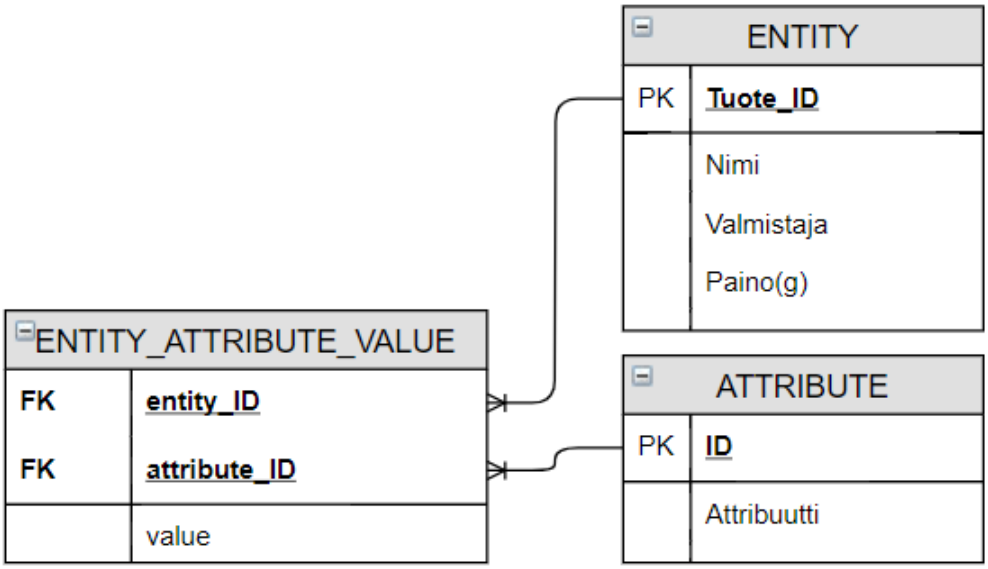
Kuva 5 Esimerkki ATTRIBUTE-taulusta

Tuotteiden ominaisuuksien arvot tallennetaan lopulta ENTITY_ATTRIBUTE_VALUE-tauluun. Tämä taulu sisältää kolme saraketta, jotka ovat entity_ID, attribute_ID ja value. (Prakash M. Nadkarni, 2019)

ENTITY_ATTRIBUTE_VALUE		
entity_ID	attribute_ID	value
1	1	50
2	3	35
1	2	16
3	4	5
1	4	2
2	5	5050

Kuva 6 Esimerkki ENTITY_ATTRIBUTE_VALUE-taulusta

Kuvassa 6 on esimerkki ENTITY_ATTRIBUTE_VALUE-taulusta. Entity_ID-sarake toimii viiteavaimena ENTITY-tauluun. Siihen tallennetaan ENTITY-taulusta halutun tuotteen perusavaimen arvo, kuten esimerkiksi rivillä 3 oleva entity_ID on 1. ENTITY-taulussa Tuote_ID:tä 1 vastaava tuote on pallo. Attribute_ID toimii viiteavaimena ATTRIBUTE-tauluun. Siihen taas tallennetaan ATTRIBUTE-taulusta halutun attribuutin perusavaimen arvo. Rivin 3 tapauksessa tämä on 2, jolloin sitä vastaava attribuutti on halkaisija (cm). Viimeinen sarake, value, on lopulta tuotteen ominaisuuden arvo. Rivillä kolme tämä on 16. Näin ollen ENTITY_ATTRIBUTE_VALUE-taulun kolmannelle riville on tallennettu tieto, että pallo-nimisen tuotteen halkaisija on 16 cm. Samoin esimerkiksi riviltä 5 käy ilmi, että saman tuotteen lukumäärä pakkausta kohden on 2 (Prakash M. Nadkarni, 2019). Kuvassa 7 on vielä havainnollistettu taulujen väliset yhteydet tietokantarakenteessa.



Kuva 7 EAV-mallin taulujen yhteydet

Kappaleessa 4.1.3 esittelin, kuinka perinteinen relaatiotietokanta ei soveltuisi ominaisuuksiltaan sovellukseni tarpeisiin. ongelmana oli, että TUOTE-taulu tulisi sisältämään hallitsemattoman määrän sarakkeita, joiden suurin osa tietueista tuli olemaan kuitenkin tyhjiä. Toinen vaihtoehto olisi ollut pilkkoa TUOTE-taulu osiin, joka olisi vaikeuttanut kyselyiden luomista. EAV-malli ratkaisee nämä ongelmat, sillä huolimatta tuotteiden tai niiden ominaisuuksien määrästä, taulujen- ja niissä olevien sarakkeiden määrä tulee säilymään muuttumattomana. Ainoastaan rivien määrä kussakin taulussa tulee muuttumaan, mikä on hyvän tietokannan tarkoitus.

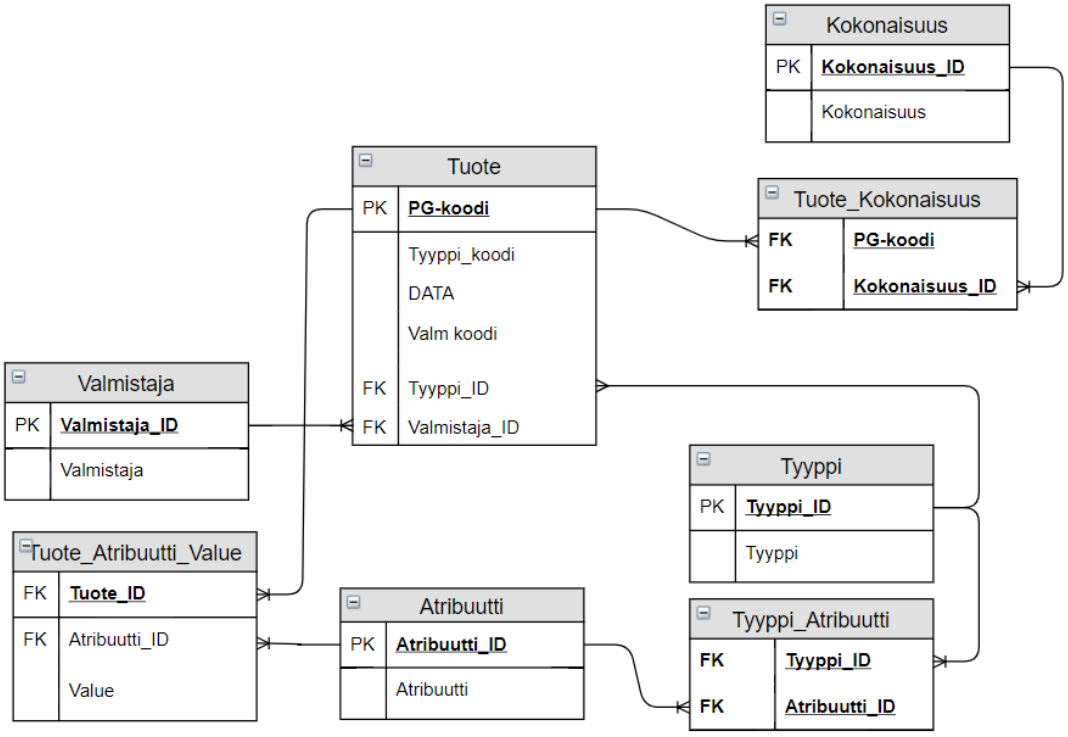
EAV-mallilla on kuitenkin haittapuolensa verrattuna relaatiomalliin. Sitä käyttämällä on huomattavasti vaikeampi valvoa, että kantaan syötetty tieto on oikeanlaista. Tämän vuoksi on ehdottoman tärkeä, että syötettäessä tietoja kantaan, niiden oikeellisuus on tarkistettu jo aiemmin. Myös SQL-kielen kyselyt ovat EAV-mallia käytettäessä huomattavasti monimutkaisempia (Prakash M. Nadkarni, 2019).

4.1.5 Sovelluksen lopullinen tietokantarakenne

Edellisessä luvussa 4.1.4. esitellyn EAV-mallin mukaisen rakenteen avulla sovelluksen tietokannan taulujen määrä pysyy vähäisenä ja helposti hallittavana. Sitä

käyttämällä tietokantaan voidaan tallentaa lukematon määrä erilaisia tuotteita, joilla on erilaisia ominaisuuksia ilman, että taulujen määrä kasvaa. Myöskään yhteenkään tauluun ei tule missään vaiheessa tarvetta lisätä uusia sarakkeita.

Käytin myös tietokannassa perinteisiä relaatiomallin rakenteita EAV-mallin tukena. Kuvassa 4. on esiteltynä tietokannan lopullinen rakenne yhteyksineen. Siitä on helposti havaittavissa EAV-mallin kolme perustaulua TUOTE, ATTRIBUUTTI ja TUOTE_ATTRIBUUTTI_VALUE.



Kuva 8Tietokannan lopullinen rakenne

Tuote-taulu on pysynyt lähes muuttumattomana siitä, miten se esiteltiin kappaleessa 4.1.1. Siihen on lisätty kaksi viiteavainta, jotka yhdistävät sen Valmistaja- ja Tyyppi-tauluihin. DATA-käsite sisältää yhä kaiken sellaisen suunnittelijalle hyödyllisen tiedon tuotteesta, jota ei olisi mielekästä erotella omaksi attribuutikseen, mutta jonka suunnittelijan on tuotteita selatessaan nähtävä. DATA-käsite ei siis ole yksilöllistä tietoa, vaan pikemmin paikka tallentaa muistiinpanoja tuotteesta.

Valmistaja-, Tyyppi- sekä Kokonaisuus-taulut ovat täysin samanlaisia kuin kappaleessa 4.1.1 esiteltiin. Kukin sisältää juoksevan numeroinnin, joka tallennetaan ID-käsitteisiin, sekä niitä vastaavat tiedot Valmistajasta, Tyypistä sekä Kokonaisuudesta.

Tuote_Kokonaisuus- taulu oli lisättävä tietokantaa, jotta Tuote- ja Kokonaisuus-taulujen välille saatiin moni moneen yhteys. Yhdellä tuotteella saattaa siis olla useita kokonaisuuksia, joita haettaessa sen tulisi näkyä. Myös luonnollisesti yhteen kokonaisuuteen kuuluu useita tuotteita. Tyyppi- ja Valmistaja-tauluilla on yksi-moneen-yhteys Tuote-taulun kanssa, sillä valmistajalla voi olla useita tuotteita ja moni tuote voi kuulua tiettyyn tyyppiin, mutta yhdellä tuotteella kumpiakin voi olla vain yksi.

Attribuutti-taulu sisältää kaksi käsitettä. Attribuutti_ID on juokseva numerointi attribuuteista ja Attribuutti on mahdollinen ominaisuus joka tuotteella voi olla. Näitä ominaisuuksia ovat siis esimerkiksi pituus, leveys, virta tai teho. Attribuutti- taululla on yksi moneen yhteys Tyyppi_Attribuutti- taulun kanssa. Tämä taulu toimii samalla tavalla kuten aiemmin mainittu Tuote_Kokonaisuus- taulu. Se luo moni moneen yhteyden Tyyppi- ja Attribuutti-taulujen välille.

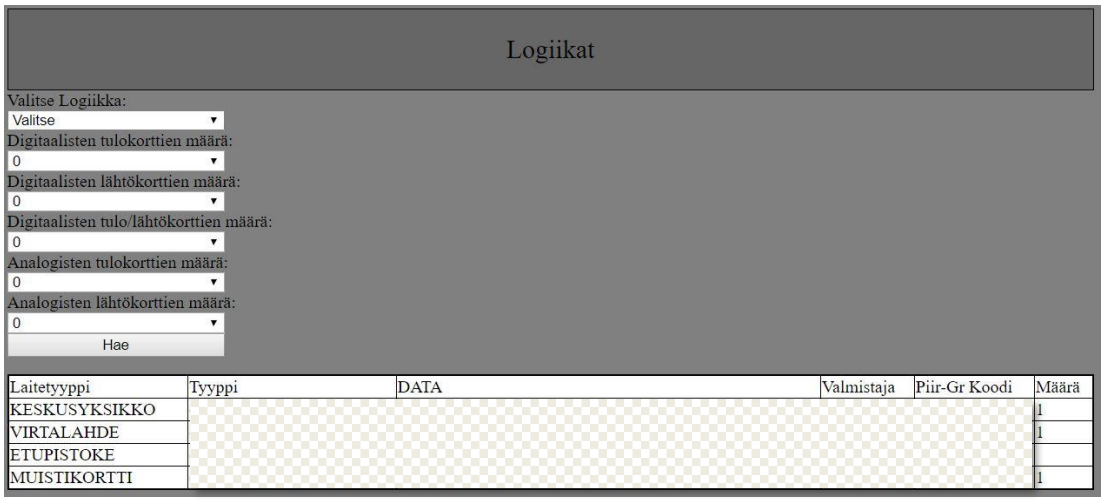
Tuote_Attribuutti_Value-taulu on se tietokannan taulu, johon tallennetaan itse tuotteen ominaisuuden arvo. Sen Tuote_ID-käsitteeseen tallennetaan tieto siitä, mikä tuote on kyseessä. Sillä on moni-yhteen-yhteys Tuote-taulun kanssa, sillä yhdellä tuotteella voi olla useita arvoja tässä taulussa. Attribuutti_ID-käsitteeseen tallennetaan tieto siitä, mikä kyseisen tuotteen ominaisuus on kyseessä. Sillä on yksi-moneen-yhteys Attribuutti-taulun kanssa, sillä useilla tuotteilla on samoja ominaisuuksia. Value- käsitteeseen tallennetaan siis itse tuotteen ominaisuuden arvo.

4.2 Käyttöliittymä

Luvussa 2.2 tavoitteeksi asetettiin, että käyttöliittymän tulee olla yhteensopiva yrityksen projekteissaan käyttämien komponenttiluetteloiden kanssa. Käyttöliittymää on suunniteltu ja muotoiltu pitäen mielessä tämä tavoite, sekä myös käyttöliittymän laajennettavuus.

4.2.1 Käyttöliittymän kehitys

Kuten tietokantarakenteen suunnittelu, myös käyttöliittymän luominen tapahtui tiiviissä yhteistyössä toimeksiantajan kanssa. Tietokannan valmistumisen jälkeen ryhdyin luomaan käyttöliittymän ensimmäistä versiota. Tässä vaiheessa oli tavoitteena luoda yksinkertainen versio käyttöliittymästä, jonka avulla on mahdollista testata tietokannan toimintaa käytännössä, sekä luoda runko lopullista käyttöliittymää varten. Kuvassa 9 on esimerkki käyttöliittymän ensimmäisestä versiosta.



Kuva 9 Käyttöliittymän ensimmäinen versio

Ensimmäinen versio sisälsi siis mahdollisuuden valita tietyn logiikkatyypin, valinnan kunkin komponentin lukumäärälle, sekä taulukon, jossa esitetään kyselyn palauttamat vastaukset (kuvassa 9 vastaukset peitetty). Tässä vaiheessa kommunikaatio käyttöliittymän ja tietokannan välillä toimi moitteettomasti ja käyttöliittymän taulukko esitti halutut tiedot tavalla, josta ne olisivat kopioitavissa komponenttiluetteloon. Kuitenkin itse käyttöliittymän käytettävyys oli vielä tässä vaiheessa huono. Pidimme toimeksiantajan kanssa palaverin, jossa esittelin käyttöliittymän alustavan version ja jonka aikana suunnittelimme tarkemmin, mitä ominaisuuksia sen tulisi sisältää ja minkälainen sen ulkoasu tulisi olla.

Palaverin pääkehityskohdaksi nousi vastaustaulukon kankeus. Taulukkoon tuli esimerkiksi aina vain ensimmäinen sopiva komponentti, joka tietokannasta löytyi. Myöskään komponenttien lukumäärää ei kyennyt vaihtamaan enää haun jälkeen tai

komponentteja ei voinut poistaa taulukosta tekemättä uutta hakua. Myös tietojen kopiointi taulukosta oli vielä mahdotonta.

Käyttöliittymän toimintaperiaate päätettiin muuttaa täysin. Sen sijaan, että yhdellä haulla se toisi kokonaisen logiikkakokoonpanon, sen avulla haettaisiin yhtä komponenttityyppiä kerralla erilaisten hakuvalikoiden avulla. Nämä tietokannalta vastauksena saadut komponentit tuotaisiin yhteen taulukkoon, josta suunnittelija voisi valita sopivan komponentin ja siirtää sen toiseen taulukkoon. Tällöin suunnittelijan olisi mahdollista rakentaa komponenteista juuri sellainen kokonaisuus, kun on kulloinkin tarpeen. Tällöin myös koko sovelluksen laajentaminen muihinkin kuin logiikkakomponentteihin olisi huomattavasti helpompaa, sillä käyttöliittymästä tulisi paljon yleiskäyttöisempi.

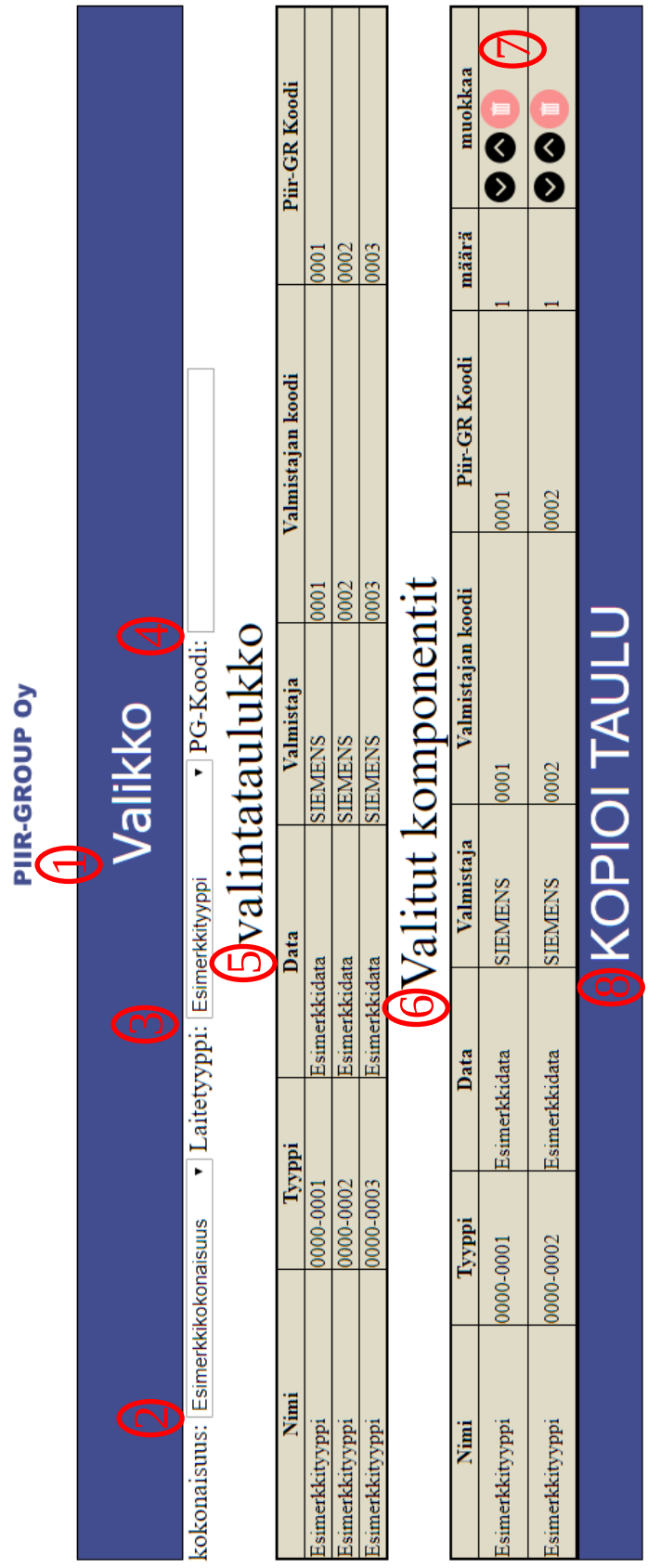
Lopullisen käyttöliittymän kehityksen aikana esittelin sen eri vaiheita toimeksiantajalle. Samalla esittelin omia kehitysideoitani ja ajatuksiani, joita mielestäni tuli sovelluksessa olla. Tiiviin yhteistyön ja toimeksiantajan palautteen avulla kykenimme räätälöimään sovelluksen parhaiten sopivaksi toimeksiantajan tarpeisiin.

4.2.2 Lopullinen käyttöliittymä

Kuvassa 10 on esiteltynä sovelluksen lopullinen käyttöliittymä. Sen taulukoissa olevat esimerkkitiedot kuvastavat oikealla haulla saatuja tietoja. Seuraavaksi esittelen käyttöliittymän osat ja niiden toiminnan.

Sivun yläosaan on sijoitettu navigointi, sekä hakuominaisuudet. Valikko-painiketta (1) painamalla käyttäjä palaa takaisin toimeksiantajan kotisivulle, josta oli linkki tähän sovellukseen. Kokonaisuusvalinta (2) ja Laitetyyppivalinta (3) ovat molemmat pudotusvalikoita, jotka esittävät kappaleessa 5.1.1 esiteltyjen kokonaisuus- ja tyyppitaulujen tiedot. Ne toimivat yhteistyössä niin, että jos toisesta on valittuna jokin arvo, toinen valikko esittää vain sellaisia arvoja, joilta tuotetaulusta löytyy yhteinen tuote. Kun esimerkiksi kokonaisuudeksi valitsee S7-1200 logiikan, joka ei käytä muistikortteja, ei tyyppivalikko esitä mahdollisuutta valita muistikorttia. Tällä tavoin helpotetaan käyttäjän hakutyötä vähentämällä valikoiden vaihtoehtojen määrä vain tarpeellisiin ja

estetään käyttäjää tekemästä hakua, joka ei tuota yhtään tulosta. PG-koodi (4) on tekstikenttä, jonka avulla käyttäjä voi hakea tiettyä tuotetta kannasta. Se päivittää hakutuloksia jokaisen merkin syöttämisen jälkeen, jolloin usein riittää vain osan koodista syöttäminen ja hakutuloksia on enää muutama jäljellä.



Kuva 10 Kuvankaappaus käyttöliittymästä selaimessa

Hakuominaisuuksien alapuolella sijaitsee valintataulukko (5). Sovellus hakee tietokannasta kaikki hakuehtoja vastaavat tuotteet ja syöttää ne tähän taulukkoon. taulukon koko riippuu tuotteiden määrästä. Käyttäjälle esitellään tuotteesta kuvassa näkyvät tiedot. Kun jotakin taulukossa olevaa tuotetta klikkaa, sovellus siirtää siitä kopion valitut komponentit-tauluun (6). Tähän tauluun käyttäjä kerää kaikki tarvitsemansa tuotteet. Sen sisältö ei muutu hakuehtoja vaihtamalla, vaan ainoastaan käyttäjän päätöksellä poistaa tai lisätä tuote.

Muokkauspainikkeilla (7) käyttäjä voi lisätä tai vähentää tietyn tuotteen lukumäärää valitut komponentit-taulussa (6), sekä poistaa tuotteen kokonaan taulusta. Kukin painike vaikuttaa ainoastaan sitä vastaavaan riviin. Sivun alalaidassa sijaitsee aina kopioi taulu-painike. Sitä painamalla käyttäjä kopioi leikepöydälle koko valitut komponentit-taulun (6) sisällön (pois lukien otsikot ja muokkauspainikkeet). Kopiointi tapahtuu niin, että tiedot säilyttävät taulukkomaisen rakenteensa. Tällöin tietojen liittäminen esimerkiksi taulukkolaskentaohjelmaan on yksinkertaista.

Kuten luvussa 2.2 mainittiin, jotkin komponentit ovat sellaisia, että ne tarvitsevat aina toisen lisäkomponentin toimiakseen. Esimerkiksi Siemensin kosketusnäytöt tarvitsevat aina muistikortin. Tällaista komponenttia valitessa sovellus tekee uuden haun tietokantaan ja luo ponnahdusikkunan sivulle, josta käyttäjän on helppo valita lisäkomponentti.

4.3 Sovelluksen toiminta

Sovelluksessa käytetään tietokantaohjelmistona MySQL tietokantaa. MySQL ohjelmisto toimii Microsoftin IIS palvelimella. Nämä ohjelmistot valikoituivat käyttöön, sillä toimeksiantajalle oli jo olemassa olevia sovelluksia, jotka toimivat näissä ympäristöissä. Toimeksiantajalla on palvelintietokone, jolle IIS on asennettuna. Kaikilla yrityksen tietokoneilla, jotka ovat kytkettynä yrityksen lähiverkkoon on mahdollisuus käyttää tietokantaa.

4.3.1 Esimerkki sovelluksen toiminnasta

SQL-kielen kyselyt tietokantaan suoritetaan PHP-tiedostoilla. Nämä tiedostot sisältävät pääsääntöisesti kolme osiota: kutsun tiedostolle, joka avaa yhteyden tietokantaan, SQL-kyselyn sekä osion, joka palauttaa kyselyn tuloksen. Kuvassa 11 on esimerkki yhdestä PHP-tiedostosta.

```php
<?php

$kokonaisuus = $_REQUEST["kokonaisuus"] ?? '';
//Kysely laitetyyppivalinnan pudotusvalikolle
include "connect.php"; //avataan yhteys tietokantaan
$nimiQuery = mysqli_query($conn, "
SELECT DISTINCT tuotetyyppi.tuotetyyppi
FROM tuotetyyppi
JOIN tuote
    ON tuotetyyppi.id = tuote.tuotetyyppi_id
WHERE tuote.pg_koodi IN (   SELECT  tuote_kokonaisuus.tuote_id
                            FROM tuote_kokonaisuus
                                JOIN kokonaisuus
                                    ON kokonaisuus.id = tuote_kokonaisuus.kokonaisuus_id
                            WHERE kokonaisuus.kokonaisuus LIKE '%".$kokonaisuus."%')
                            ORDER BY tuotetyyppi");

        //palauttaa tulokset html <select> tagin sisälle
        //ensimmäinen arvo tyhjä
        echo '<option value="">Valitse</option>';
        while($result = mysqli_fetch_array($nimiQuery, MYSQLI_ASSOC)){
                echo '<option value="'.$result['tuotetyyppi'].'"> '.$result['tuotetyyppi'].'</option>';
        }
?>
```

Kuva 11 Esimerkki laitetyyppivalinnan pudotusvalikon PHP-tiedostosta

PHP-tiedostoon tuodaan käyttöliittymästä muuttujia, jotka syötetään itse SQL-kyselyyn. Kuvassa 11 laitetyyppivalinnan kyselyyn vaikuttaa esimerkiksi kokonaisuusmuuttuja. Tässä tapauksessa käyttäjä on saattanut valita tietyn kokonaisuuden, jonka laitteita hän haluaa tarkastella. Tällöin kysely hakee tietokannasta vain niitä laitetyyppejä, joita kyseinen kokonaisuus sisältää. Jos käyttäjä on jättänyt kokonaisuuden valitsematta, kysely palauttaa kaikki sen sisältämät laitetyypit.

PHP-tiedoston tietokannalta saama vastaus ei ole sellaisessa muodossa, että sen voi sellaisenaan syöttää käyttöliittymään vaan se on muokattava. Tämä suoritetaan PHP-tiedoston sisällä. Kuvan 11 tapauksessa vastaus halutaan syöttää pudotusvalikkoon, joten se muokataan HTML-koodiksi ja palautetaan sen jälkeen. Vastauksen muokkaaminen HTML-muotoon välittömästi helpottaa sen käsittelyä myöhemmin.

Käyttöliittymää ohjaava JavaScript-tiedosto sisältää funktioita, joilla kutsutaan näitä PHP-tiedostoja. Käyttäjän tehdessä valintoja käyttöliittymässä JavaScript tiedosto suoritta tarpeelliset kutsut ja palauttaa vastaukset käyttöliittymälle.

```
function tyyppivalinta() {

        var kokonaisuus =  document.getElementById("kokonaisuus").value;
        var xmlhttp = new XMLHttpRequest();

        xmlhttp.open("GET", "Queries/TyyppivalikkoQuery.php?kokonaisuus=" + kokonaisuus, true); //kutsutaan php-tiedosto

        xmlhttp.send();

        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("tyyppi").innerHTML = this.responseText //vastaus syötetään <select> tagiin
            }
        };
}
```

Kuva 12 Esimerkki laitetyyppivalinnan pudotusvalikon funktiosta JavaScript-tiedostossa

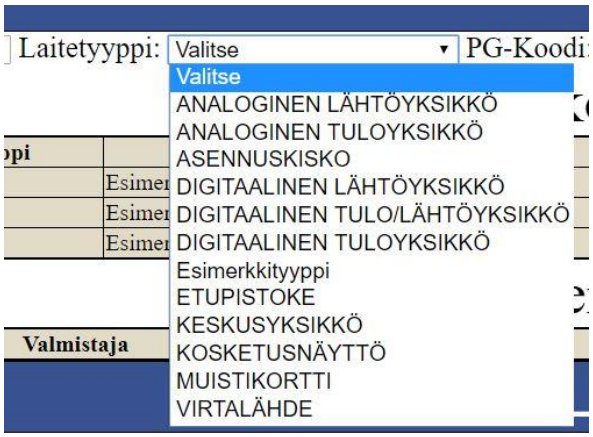
Kuvassa 12 on esimerkki tyyppivalinnan pudotusvalikon funktiosta JavaScript tiedoston sisällä. Kuten PHP-tiedostot, myös nämä funktiot ovat pääsääntöisesti kolmeosaisia. Ensin funktio hakee käyttöliittymästä tarvittavat muuttujat. Tämän jälkeen se suorittaa kutsun PHP-tiedostolle ja syöttää sille muuttujien arvot. Viimeiseksi se vastaanottaa PHP-tiedostolta saamansa vastauksen, ja syöttää sen käyttöliittymään.

```
Laitetyyppi:
<!--laitetyyppivalinnan pudotusvalikko-->
<!--täyttyy tyyppivalinta-funktion arvoilla -->
<!--arvon muuttuessa päivittää Valintataulukon-->
<select name= "tyyppi" id="tyyppi" onfocus="tyyppivalinta()" onchange="valintataulukko_update();"></select>
```

Kuva 13 Esimerkki laitetyyppivalinnan pudotusvalikon elementistä HTML-tiedostossa

Käyttöliittymänä toimii HTML-sivu. Sivun latautuessa se sisältää ainoastaan tyhjiä elementtejä, joita Javascript tiedosto sitten täyttää. Kuvassa 13 on laitetyyppivalinnan HTML-elementti. Kun Javascript suorittaa tyyppivalinta-funktion, palauttaa se PHP-tiedostolta saamansa HTML-muotoisen vastauksen tämän elementin sisälle.

Kuva 14 on kaappaus käyttöliittymästä, jossa sovellus on suorittanut kyselyn tietokantaan ja palauttanut sieltä saamansa vastauksen laitetyyppivalinnan pudotusvalikkoon.



Kuva 14 Laitetyyppivalinnan pudotusvalikko HTML-sivulla

Edellä esitelty laitetyyppivalinnan pudotusvalikko on esimerkki kaikkien sovelluksen komponenttien toiminnasta. Jokaisella on oma PHP-tiedostonsa, joka sisältää tarvittavan SQL-kyselyn, sekä oma funktionsa Javascript-tiedostossa, joka syöttää saamansa vastauksen HTML-elementtiin.

5 POHDINTA

Opinnäytetyön tavoitteena oli luoda PIIR-GROUP Oy:lle tietokantasovellus, joka tehostaisi suunnitteluprosessia. Sovelluksen oli tarkoitus olla eräänlainen hakukone, jonka avulla suunnittelija voisi vaivattomasti valita projektinsa osaluetteloon tarvittavat komponentit.

Sovelluksen kehityksen alkuvaiheessa ei kenelläkään vielä ollut tarkkaa kuvaa siitä, minkälainen lopullisesta sovelluksesta tulisi, vaan kehitystyö tapahtui vahvassa yhteistyössä toimeksiantajan kanssa. Esittelin tekemiäni ratkaisuja ja niiden perusteluja säännöllisesti toimeksiantajalle. Sain toimeksiantajalta jokaisella kerralla palautetta, siitä mitä ratkaisuja he mielestään pitivät hyvinä ja mihin he haluaisivat muutoksia tai lisäyksiä. Tällaisesta yhteistyöstä hyvä esimerkki on opinnäytetyössä tehty ratkaisu käyttää EAV-mallia relaatiomallin sijaan.

Opinnäytetyön lopputuloksena syntyi tietokanta ja käyttöliittymä, sekä taustalla toimiva koodi, joka ohjaa molempien toimintaa. Tietokannan päätavoitteet olivat, että se olisi yhteensopiva olemassa olevien komponenttiluetteloiden kanssa, ja että sen laajentaminen olisi mahdollisimman yksinkertaista. Mielestäni molemmat tavoitteet täyttyivät kokonaisuudessaan. Tietokantaan on mahdollista tallentaa tuotteista samat tiedot, jotka vanha vakiokomponenttiluettelo sisälsi, sekä EAV-malli mahdollistaa kannan äärettömän laajentamisen.

Käyttöliittymästä tuli mielestäni myös tarkoituksenmukainen. Sen tavoitteena oli olla mahdollisimman helppokäyttöinen ja myös yhteensopiva käytössä olevien osaluetteloiden kanssa. En luonut käyttöliittymälle käyttöohjetta, sillä koin sen olevan riittävän intuitiivinen ja yksinkertainen, ettei sellaiselle ollut tarvetta. Toimeksiantajalta saamani palautteen mukaan sovelluksen käyttö on ollut helppoa ja ohjeita ei ole tarvinnut.

Tulevaisuudessa on tarkoituksena toimeksiantajan kanssa yhteistyössä syöttää tietokantaan kaikki olemassa olevien luetteloiden komponentit, sekä ylläpitää sitä päivittämällä uusia komponentteja ja poistamalla käytöstä poistuneita. Käyttöliittymään on tarkoitus luoda erilaisia sivuja, joiden avulla on mahdollista tehdä eri tarkoituksiin

tarvittavia kokonaisuuksia. Esimerkiksi sivu, johon syöttämällä moottorin arvot, sovellus ehdottaisi listan tarvittavista suojalaitteista.

Toimeksiantaja on sittemmin suunnitellut siirtyvänsä käyttämään tietokantapohjaista suunnitteluohjelmistoa. Tällainen ohjelmisto sisältäisi hyvin samanlaisen toiminnallisuuden kuin opinnäytetyössä tekemäni sovellus, mutta se olisi huomattavasti vahvemmin integroitu koko suunnitteluprosessiin.

Sovellusta tehdessäni ymmärrykseni relaatiotietokantojen toiminnasta kasvoi. Varsinkin siitä mihin sovellustarkoituksiin relaatiotietokanta on soveltuva ja missä, kuten tässä opinnäytetyössä, sillä ei saada haluttua tulosta. Tutustuin vasta tätä opinnäytetyötä tehdessäni ensimmäistä kertaa EAV-kantaan. Vaikka EAV-mallin periaate onkin yksinkertainen, sen toteuttaminen käytännössä vaati syventymistä ja huomattavan määrän kokeilua toimiakseen. Tämän opinnäytetyön tehtyäni tiedän EAV-kannan edut tietyissä tapauksissa verrattuna relaatiotietokantaan, mutta myös sen, että suurimmassa osassa sovelluksia relaatiomalli on huomattavasti parempi vaihtoehto.

LÄHTEET

Apache Friends. (1. 17 2019). *Apache Friends*. Noudettu osoitteesta https://www.apachefriends.org

Hovi, H. &. (2005). *Tietokantojen suunnittelu & indeksointi.* Porvoo: WS Bookwell.

Microsoft. (15. 5 2019). *IIS.net*. Noudettu osoitteesta https://www.iis.net/

Mozilla. (1. 17 2019). *MDN web docs*. Noudettu osoitteesta https://developer.mozilla.org/en-US/

Notepad++. (12. 4 2019). *Notepad++*. Noudettu osoitteesta https://notepad-plus-plus.org/

Oracle. (23. 1 2019). *Oracle*. Noudettu osoitteesta https://www.oracle.com/mysql/

Prakash M. Nadkarni, M. L. (24. 5 2019). *NCBI*. Noudettu osoitteesta Organization of Heterogeneous Scientific Data Using the EAV/CR Representation: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC61391/

The PHP Group. (15. 5 2019). *What is PHP*. Noudettu osoitteesta https://secure.php.net/manual/en/intro-whatis.php

W3C. (15. 4 2019). *W3C-verkkosivut*. Noudettu osoitteesta https://www.w3.org/