OPINNÄYTETYÖ
JANNE KOIVURANTA 2013

SÄHKÖINEN TAVAROIDENHALLINTAJÄR-JESTELMÄ

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences
LUC

TIETOTEKNIIKAN KOULUTUSOHJELMA

Rovaniemen
ammattikorkeakoulu
University of Applied Sciences
LUC

ROVANIEMEN AMMATTIKORKEAKOULU

TEKNIIKKA JA LIIKENNE

Tietotekniikan koulutusohjelma

Opinnäytetyö

SÄHKÖINEN TAVAROIDENHALLINTAJÄRJESTELMÄ

Janne Koivuranta

2013

Toimeksiantaja Rovaniemen ammattikorkeakoulun rakennustekniikan laboratorio

Ohjaaja Tauno Tepsa

Hyväksytty _____ 2013 _____

Rovaniemen ammattikorkeakoulu
University of Applied Sciences
LUC

Tekniikka ja liikenne
Tietotekniikan koulutusohjelma

Opinnäytetyön tiivistelmä

Tekijä Janne Koivuranta Vuosi 2013

Toimeksiantaja Rovaniemen ammattikorkeakoulun rakennustekniikan laboratorio
Työn nimi Sähköinen tavaroidenhallintajärjestelmä
Sivu- ja liitemäärä 71 + 2

Tämä työ on tehty toimeksiantona Rovaniemen ammattikorkeakoulun rakennustekniikan laboratorioon. Työn tarkoituksena oli toteuttaa sähköinen tavaroidenhallintajärjestelmä. Järjestelmän tarkoituksena on toimia varastonhallintatyökaluna laboratorion tavaroille, mutta myös kirjanpitojärjestelmänä tavaroiden lainaamisessa ja varaamisessa. Sovellus toimii laboratorion työasemalla, ja sen toteutuksessa on erityisesti kiinnitetty huomiota helppokäyttöisyyteen.

Käyttöliittymän toteuttamisessa on käytetty Microsoftin .NET-arkkitehtuuria hyödyntävää Windows Forms -tekniikkaa ja ohjelmointikielenä C#. Kehitysohjelmistona toimi Microsoft Visual Studio 2010. Tietokanta on toteutettu MySQL-tietokantaan käyttäen tietokannan suunnittelussa apuna Sybase Power Designer -ohjelmaa. Tavaroiden lainaamistoiminnossa käytetään apuna viivakoodinlukijaa.

Opinnäytetyön tuloksena saatiin käyttöönottovalmis sovellus, jota opinnäytetyöprosessin aikataulun takia ei kuitenkaan vielä ehditty ottaa käyttöön. Käyttöliittymän toteutuksessa käytettävät tekniikat toimivat hyvin ja MySQL-palvelin osoittautui hyväksi vaihtoehdoksi helppokäyttöisyytensä ja nopeutensa vuoksi. Lisäksi kaikki vaatimusmäärittelyssä tehdyt vaatimukset toteutuivat.

Järjestelmässä on paljon jatkokehitysmahdollisuuksia, kuten käyttäminen mobiili-laitteilla. Myös oman mobiilisovelluksen toteuttaminen on mahdollista. Käyttöliittymän käytettävyyden ja ulkoasun kannalta monipuolisempi Windows Presentation Foundation (WPF) -tekniikka loisi uusia mahdollisuuksia. WPF olisikin varmasti tekniikka, jolla seuraava versio kannattaisi toteuttaa.

Avainsanat C#, Windows Forms, MySQL, .NET, varastonhallinta

Rovaniemen ammattikorkeakoulu
University of Applied Sciences
LUC

School of Technology
Information Technology Programme

Abstract of Thesis

Author Janne Koivuranta Year 2013

Commissioned by Construction Engineering Laboratory of Rovaniemi University of Applied Sciences

Subject of thesis A Computer-Based Inventory Management System

Number of pages 71 + 2

This thesis was commissioned by the Construction Engineering Laboratory of the Rovaniemi University of Applied Sciences. The purpose of the thesis was to implement a computer-based inventory management system. The system also serves as an accounting system for the loans and bookings of the items. The application was installed on a workstation in the laboratory. Special attention was paid to the ease of use while developing the system.

The user interface was implemented using the Windows Forms technique which uses the Microsoft .NET architecture. The programming language is C#. The Microsoft Visual Studio 2010 was used as the development software. MySQL was used as the database software and the database was designed by using the Sybase Power Designer development software. A barcode reader is used in the loaning process of the goods.

The result of the thesis was a system ready for deployment but due to the schedule of the thesis project the system was not introduced yet. The techniques used in implementing the user interface worked well and the MySQL server proved to be a good choice due to its ease of use and speed. The requirements were met as well.

The system has much room for further development such as using the system through a mobile device. In addition a specific mobile application would be feasible. A more versatile Windows Presentation Foundation or WPF would provide new opportunities from the user interface and from the visual point of view. The WPF is certainly the technique the second version of the system should be implemented with.

Key words C#, Windows Forms, MySQL, .NET, inventory management

SISÄLTÖ

KUVIOLUETTELO 1

ESIMERKKIKOODI-LUETTELO 2

TERMI- JA KÄSITELUETTELO 3

1 JOHDANTO 6

2 KEHITTÄMISESSÄ KÄYTETYT OHJELMISTOT JA TEKNIIKAT 8

2.1 OHJELMOINTIKIELET 8
2.1.1 C# 8
2.1.2 SQL-kyselykieli 9
2.2 OHJELMOINTITEKNIIKAT 10
2.2.1 Ohjelmistot 10
2.2.2 Windows Forms 12
2.3 PROTOILUMALLI 13

3 MÄÄRITTELY 16

3.1 YLEISKUVAUS JA YMPÄRISTÖ 16
3.2 KÄYTTÄJÄTYYPIT 17
3.3 TOIMINNALLISET VAATIMUKSET JA PROSESSIT 18
3.3.1 Lainausten hallinta 18
3.3.2 Palautusten hallinta 21
3.3.3 Tavaroiden hallinta 22
3.3.4 Lainaajien hallinta 24
3.4 ERITYISVAATIMUKSET JA RAJOITUKSET 27
3.5 TEKNINEN MÄÄRITTELY 29
3.5.1 Järjestelmän yleiskuvaus 29
3.5.2 Käyttötapaukset 30
3.5.3 Tietokanta 31
3.5.4 Ohjelmamoduulit ja nimeämiskäytännöt 32

4 TOTEUTUS 35

4.1 PROJEKTIN VAIHEET JA LÄPIVIENTI 35
4.2 KÄYTTÖLIITTYMÄN RAKENNE 36
4.2.1 Ulkoasu ja komponentit 36
4.2.2 Navigointi 37
4.3 KIRJAUTUMINEN JA REKISTERÖITYMINEN 39
4.4 PÄÄSIVU 41
4.4.1 Tavaran lainaaminen 41
4.4.2 Tavarat 44
4.4.3 Lainaajat 47
4.4.4 Lainaukset ja varaukset 48
4.4.5 Kategoriat ja positiot 50
4.4.6 Muistutukset 51
4.5 LISÄÄ- JA MUOKKAA -APUIKKUNAT 52
4.6 TIETOKANTA JA TIETOKANTATAULUT 55
4.7 SYÖTTEIDEN TARKISTAMINEN JA POIKKEUSTENKÄSITTELY 56

5 TESTAUS 59

5.1 TESTAUKSEN TARKOITUS JA KOHTEET 59
5.2 TESTAUKSEN TULOKSET 60

6 TULOKSET JA JOHTOPÄÄTÖKSET 62

LÄHTEET 65

LIITTEET 66

KUVIOLUETTELO

KUVIO 1. WINDOWS FORMS -LOMAKE 13
KUVIO 2. PROTOILUMALLI 14
KUVIO 3. TAVAROIDENHALLINTAJÄRJESTELMÄN YMPÄRISTÖKAAVIO 16
KUVIO 4. LAINAUSPROSESSIN NYKYTILA 19
KUVIO 5. LAINAUSPROSESSIN TAVOITETILA 20
KUVIO 6. PALAUTUSPROSESSIN NYKYTILA 21
KUVIO 7. PALAUTUSPROSESSIN TAVOITETILA 22
KUVIO 8. TAVAROIDENHALLINTAPROSESSIN NYKYTILA 23
KUVIO 9. TAVAROIDENHALLINTAPROSESSIN TAVOITETILA 24
KUVIO 10. LAINAAJIENHALLINTAPROSESSIN NYKYTILA 25
KUVIO 11. LAINAAJIENHALLINTAPROSESSIN TAVOITETILA 26
KUVIO 12. TAVAROIDENHALLINTAJÄRJESTELMÄN PÄÄIKKUNA 29
KUVIO 13. KÄYTTÖTAPAUSKAAVIO 30
KUVIO 14. TAVAROIDENHALLINTAJÄRJESTELMÄN TIETOKANNAN FYYSINEN TIETOMALLI . 31
KUVIO 15. KÄYTTÖLIITTYMÄN MUOKKAA LAINAAJAN TIETOJA -IKKUNA 37
KUVIO 16. GRAAFISEN KÄYTTÖLIITTYMÄN NAVIGOINTIKARTTA 38
KUVIO 17. KIRJAUTUMISSIVU 39
KUVIO 18. PÄÄSIVUN LAINAA TAVARA -VÄLILEHTI 42
KUVIO 19. PÄÄSIVUN TAVARAT-VÄLILEHTI 44
KUVIO 20. PÄÄSIVUN LAINAAJAT-VÄLILEHTI 48
KUVIO 21. PÄÄSIVUN LAINAUKSET/VARAUKSET-VÄLILEHTI 49
KUVIO 22. PÄÄSIVUN KATEGORIAT-VÄLILEHTI 50
KUVIO 23. PÄÄSIVUN POSITIOT-VÄLILEHTI 51
KUVIO 24. PÄÄSIVUN MUISTUTUKSET-VÄLILEHTI 52
KUVIO 25. LISÄÄ TAVARA -IKKUNA 53
KUVIO 26. MUOKKAA TAVARAA -IKKUNA 54

ESIMERKKIKOODI-LUETTELO

ESIMERKKIKOODI 1. C#-SYNTAKSI-ESIMERKKI 8
ESIMERKKIKOODI 2. REGULAR EXPRESSIONS -MERKKIJONOHAHMO 9
ESIMERKKIKOODI 3. SQL:N SELECT-LAUSE 9
ESIMERKKIKOODI 4. WINDOWS FORMS -SOVELLUKSEN ALOITUSPISTE 12
ESIMERKKIKOODI 5. SALASANAN TARKISTAMINEN 40
ESIMERKKIKOODI 6. UUDEN FORM-INSTANSSIN LUOMINEN JA NÄKYVILLE ASETTAMINEN 41
ESIMERKKIKOODI 7. LAINAUKSEN TIETOJEN TALLENTAMINEN TIETOKANTAAN 43
ESIMERKKIKOODI 8. SELECT-LAUSE TAVAROIDENHAKUTOIMINNOSSA 46
ESIMERKKIKOODI 9. HAKUTULOKSIEN TALLENTAMINEN DATAGRIDVIEW-TAULUUN 47
ESIMERKKIKOODI 10. UMPEUTUNEIDEN ERÄPÄIVIEN TARKISTAMINEN 51
ESIMERKKIKOODI 11. CREATE TABLE -LAUSE 56
ESIMERKKIKOODI 12. SYÖTTEIDEN TARKISTAMINEN 57
ESIMERKKIKOODI 13. TRY-CATCH -LOHKO POIKKEUSTENKÄSITTELYSSÄ. 58

TERMI- JA KÄSITELUETTELO

Attribuutti	Tietokantataulun eli kohdetyypin ominaisuus
Auto increment	SQL-relaatiotietokannan tietotyyppi, jota käytetään tietokantataulussa yksilöivänä tunnisteena ja kasvaa automaattisesti seuraavaan vapaana olevaan id-numeroon.
Button	Windows Forms -tekniikan käyttöliittymäkomponentti
C	Alun perin 1970-luvun alussa Unix-käyttöjärjestelmille luotu imperatiivinen ohjelmointikieli, johon monet nykyiset ohjelmointikielet, kuten C++, C# ja Java pohjautuvat (Wikikirjasto 2008).
C++	Laajasti käytetty ja useita eri alustoja tukeva imperatiivinen ohjelmointikieli. Oleellisimpia eroja C-kieleen verrattuna on luokka-ominaisuus.
C#	Olio-pohjainen imperatiivinen ohjelmointikieli
ComboBox	Windows Forms -tekniikan käyttöliittymäkomponentti
Command line	Tietokoneen ja käyttäjän välille toteutettu käyttäjärajapinta, joka ottaa vastaan käyttäjän syöttämiä peräkkäisiä käskyrivejä (Command lines).
DataGridView	Tietotaulukko, ja Windows Forms -tekniikan käyttöliittymäkomponentti
Date	Useissa ohjelmointikielissä käytössä oleva päivämäärä-tietotyyppi.
DateTimePicker	Windows Forms -tekniikan käyttöliittymäkomponentti
DCL	Data Control Language. Käytetään määrittelemään käyttäjän käyttöoikeudet tietokannassa (Oracle FAQ's 2004).
DDL	Data Definition Language. Tietokannan rakenteen käsittelyyn käytetty kieli (Oracle FAQ's 2004).
DDR	Double data rate. Tietoteknisissä laitteissa keskusmuistin tyyppiä kuvaava termi.

DML	Data Manipulation Language. Tietokannan sisältämän tiedon käsittelyyn käytetty kieli. (Oracle FAQ's 2004.)
Foreign key	Tietokantataulun viiteavain
GroupBox	Windows Forms -tekniikan käyttöliittymäkomponentti
GUI	Graphical User Interface. Graafinen käyttöliittymä
Integer	Kokonaisluku-tietotyyppi
Luokka	Olio-ohjelmoinnissa käytetty tietorakenne, joka sisältää samaan asiaan liittyviä tietoja. Luokka voi tiedon lisäksi sisältää omia jäsenfunktioita, joilla tietoa voidaan käsitellä luokan sisällä.
Metodi	C-ohjelmoinnissa askeleittain etenevän toimintosarjan nimitys.
MySQL	Avoimen lähdekoodin relaatiotietokantaohjelmisto
MySQL Connector	MySQL palvelimen C#-ohjelmointikielelle toteutettu rajapinta.
NULL	Tyhjä. Määrittelemätön muuttujan arvo
Offline-tila	Yleisesti käytetty nimitys tilasta, jossa esimerkiksi työaseman verkkoyhteys on katkennut. Vastakohta "Online-tila" tarkoittaa juuri päinvastaista tilaa, jossa työasema on yhteydessä verkkoon.
Olio	Olio-ohjelmoinnissa käytetty nimitys luokan ajonaikaisesta ilmentymästä (eng. object).
Olio-ohjelmointi	Englanniksi object-oriented programming, eli OOP. Esimerkiksi C-kielessä käytettyjen tietorakenteiden sijaan olio-ohjelmoinnissa samaan asiaan liittyvät tiedot kerätään luokiksi. Olio, eli instanssi on luokan ajonaikainen ilmentymä. (Ohjelmointiputka 2011.)
Primary key	Tietokantataulun pääavain
PDM	Physical data model. Kuvaa mm. tietokannan rakennetta ja tietokannan taulujen suhteita.
RadioButton	Windows Forms -tekniikan käyttöliittymäkomponentti
Relaatiotietokanta	Useista toisiinsa viittaavista tietokantatauluista muodostuva tietokanta.

Regular expressions	Regular expressions on muun muassa C#-ohjelmointikielen tukema tekniikka, jota käytetään tunnistamaan merkkijonoista tiettyjä määriteltyjä palasia, ns. merkkijonohahmoja.
Skripti	Joukko ohjeita jonkin halutun toiminnon suorittamiseksi, esimerkiksi tietokannan luominen.
SQL	Structured Query Language, standardoitu kyselykieli relaatiotietokantoihin
Syntaksi	Ohjelmointikielen lauseoppi
TabControl	Windows Forms -tekniikan käyttöliittymäkomponentti
TextBox	Windows Forms -tekniikan käyttöliittymäkomponentti
Viite-eheys	Varmistaa, että tietokannassa tauluun ei voi syöttää tietoa, jota ei taulun äiti- eli parent-taulusta löydy.
Varchar	SQL-relaatiotietokannoissa käytetty merkkijono-tietotyyppi
Windows Forms	Microsoftin kehittämä graafinen ohjelmointirajapinta
WPF	Windows Presentation Foundation. Muodostaa uusimpien Windows-versioiden graafisen rajapinnan.
.Designer.cs	Windows Forms -lomakkeen alustuskoodi-tiedoston tiedostotarkennin
.cs	C#-tiedoston tiedostotarkennin
.exe	Yleisimmin käytetty tiedostotarkennin suoritettavasta tiedostosta.
.resx	Windows Forms -lomakkeen resurssitiedoston tiedostotarkennin
.Net Framework	Microsoftin kehittämä ohjelmistokomponenttikirjasto, jota Microsoftin VisualStudio.NET-ympäristössä kehitetyt ohjelmistot käyttävät (Mannila 2013).

1 JOHDANTO

Tämä opinnäytetyö sai alkunsa toimeksiantona, jonka tavoitteena on toteuttaa sähköinen tavaroidenhallintajärjestelmä. Asiakas on Rovaniemen ammattikorkeakoulun rakennustekniikan laboratorio. Laboratorio sijaitsee Tekniikan ja liikenteen alan kampuksella Rovaniemen Rantavitikalla. Laboratoriota käyttävät niin opiskelijat kuin RAMK:n henkilökuntakin omissa asiakasprojekteissaan.

Rakennustekniikan laboratoriossa säilytetään lukuisia erilaisia työkaluja ja mittavälineitä, joita käytetään mm. oppilastöissä sekä RAMK:n omissa asiakasprojekteissa. Etenkin mittavälineiden kohdalla on tärkeää, että ne kalibroidaan säännöllisesti, ja tämä vaatii tarkkaa kirjanpitoa. Tarkkaa kirjanpitoa vaatii myös tavaroiden varastotietojen hallinta. Tavaroidenhallinta on tähän asti ollut laboratorioinsinöörin vastuulla, ja menetelminään hän on käyttänyt enimmäkseen Microsoft Excel -taulukoita. Tämä on työlästä, koska kullakin tavaralla on useita ominaisuuksia, jotka muuttuvat ajan kuluessa. Ominaisuuksia ovat esimerkiksi nimi, kalibrointipäivämäärä, hankintapäivämäärä sekä tila ja positio. Tavaroiden lainauksista ja lainaajista on myös täytynyt pitää kirjanpitoa, joka on tehty tähän asti paperilla ja kynällä ns. ”muistilappukäytännöllä”.

Opinnäytetyön tavoitteena on toteuttaa mahdollisimman yksinkertainen ja käyttäjäystävällinen sähköinen tavaroidenhallintajärjestelmä, joka helpottaa tavaroiden varaston- ja lainauksienhallintaa. Lainauksiin liittyy aina myös lainaaja, joten myös lainaajista täytyy pitää omaa rekisteriään. Toteutettava järjestelmä kehitetään toimimaan sekä varaston-, lainausten-, varauksien-, että lainaajienhallintajärjestelmänä pitäen sisällään paljon tietoa kustakin ryhmästä. Järjestelmä toteutetaan käytettäväksi rakennustekniikan laboratorion Windows-työasemilla.

Tavaroidenhallintajärjestelmä kehitetään ohjelmistoprojektityönä. Tietolähteinä käytetään pääasiassa tekijän omia tietoja, mutta jonkin verran myös Internet-lähteistä haettuja tietoja. Tässä opinnäytetyöraportissa esitellään projektin vaiheet. Aluksi luodaan teoriapainotteinen katsaus toteutuksessa hyödynnetyistä ohjelmisto- ja ohjelmointikieliratkaisuista. Sen jälkeen kuvataan vaa-

timusmäärittely, tekninen määrittely, toteutus ja testaus. Lopuksi kerätään yhteen tulokset ja pohditaan niiden perusteella johtopäätökset.

2 KEHITTÄMISESSÄ KÄYTETYT OHJELMISTOT JA TEKNIIKAT

2.1 Ohjelmointikielet

2.1.1 C#

C# (lausutaan ”C sharp”) on Microsoftin vuonna 2000 julkaisema .NET-konseptia varten kehitetty olio-pohjainen ohjelmointikieli. C# tukee yleisimpiä metodeja ja tietotyyppejä, jonka ansiosta se on tyyppiturvallinen ja tehokas ohjelmointikieli. C# ei käytä erillisiä otsikointitiedostoja, kuten esimerkiksi C++. Sen sijaan C#-lähdetiedostoon voidaan määritellä niin monta luokkaa, struktuuria, rajanpintaa tai tapahtumaa kuin on tarpeellista. (MSDN 2013.) Edellisen lisäksi C#-kielen muovautuvuus ja nykyaikaisuus olivat syitä C#-kielen valitsemiseen juuri tätä projektia varten. Lopullinen ratkaisu C#-kielen valintaan syntyi, kun sen todettiin olevan hyvin yhteensopiva ohjelmointikieli Windows Forms -tekniikan kanssa. C#-kielen syntaksi on ilmaisultaan erittäin tarkkaa, mutta kuitenkin C-, C++- tai Java -kieltä käyttäneet pystyvät lyhyessä ajassa oppimaan sen.

```
// Tervehdys.cs
using System;
class MainApp
{
     public static void Main()
     {
            Console.WriteLine("Terve C#-maailma!");
     }
}
```

Esimerkkikoodi 1. C#-syntaksi-esimerkki (Anttonen 2006)

Esimerkkikoodissa 1 using-komennolla otetaan käyttöön System-nimiavaruus, jossa sijaitsevan Console-luokan WriteLine-metodille välitetään merkkijono ”Tervehdys maailma!”. Näin saadaan konsolille tulostettua teksti ”Tervehdys maailma”. Tässä tapauksessa sovelluksen luokan nimi on ”MainApp”, jolle on määritelty suojausmääreeksi public eli julkinen. Näin ollen se näkyy myös luokan ulkopuolelle. Static void Main() -metodi on sovelluksen pääohjelma, josta sovellus alkaa. Kooditiedoston nimi Tervehdys.cs on esitetty kommenttina ylimmällä rivillä.

Regular expressions -tekniikka

Regular expressions on muun muassa C#-ohjelmointikielen tukema tekniikka, jota käytetään tunnistamaan merkkijonoista tiettyjä määriteltyjä palasia, ns. merkkijonohahmoja. Merkkijonohahmo määritellään jokseenkin kryptisen oloisella merkkijono-syntaksilla.

```
string haluttuViivakoodiRakenne = @"\b[0-9]{2,19}\b";
```

Esimerkkikoodi 2. Regular Expressions -merkkijonohahmo

Esimerkkikoodi 2 esittää koodia, jossa määritellään Regular expressions -tekniikalla haluttu rakenne etsittävälle merkkijonohahmolle, joka tässä tapauksessa on viivakoodi. Koodissa esitellään merkkijono nimeltä haluttuViivakoodiRakenne, johon sijoitetaan ”@”\b[0-9]{2,19}\b”-merkkijono. Tämä varsin monimutkainen koodinpätkä määrää, että viivakoodi pitää olla pituudeltaan 2–19 merkkiä, jotka saavat olla numeroita väliltä 0–9. Tätä tekniikkaa tarvitaan tässä sovelluksessa muun muassa määrittämään viivakoodin oikea muoto. Kyseisellä tavalla tarkastetaan myös muut käyttäjän antamat syötteet, kuten etunimi, sukunimi, tunniste, ryhmä, sähköpostiosoite, matkapuhelinnumero jne.

2.1.2 SQL-kyselykieli

Tavaroidenhallintajärjestelmässä käytetään SQL-kyselykieltä tukevaa tietokantaratkaisua. SQL (Structured Query Language) on IBM:n kehittämä yksinkertainen ohjelmointikieli, jonka avulla tietokantoja voidaan hallita. SQL-lauseiden avulla tietokantaan voidaan tehdä hakuja, muutoksia, lisäyksiä jne.

```
SELECT merkki, malli, ajetutKm
FROM autot
WHERE ajetutKm BETWEEN 30000 AND 150000
ORDER BY ajetutKm DESC;
```

Esimerkkikoodi 3. SQL:n select-lause

Esimerkkikoodissa 3 käytetään DML-lausetta, jossa select-lauseella haetaan ”autot”-nimisestä tietokantataulusta merkki, malli, ja ajetut kilometrit niiden autojen osalta, joiden ajetut kilometrit ovat välillä 30 000–150 000. Kysely järjestää tulokset ajettujen kilometrien mukaan suurimmasta pienimpään. Tietojen hakemisen lisäksi myös itse tietokannan rakennetta ja käyttöoikeuk-

sia voidaan muuttaa. SQL jaetaan yleensä kahteen osaan: tietokannan rakenteen määrittelyyn (Data Definition Language DDL) ja tietokannan sisällön käsittelyyn (Data Manipulation Language DML) (Oulun seudun ammattiopisto 2004). Data Control Language (DCL) -kieli mahdollistaa käyttäjien käyttöoikeuksien hallinnoimisen.

Data Definition Language (DDL) -komentoja

CREATE	luo objekteja tietokantaan
ALTER	muokkaa tietokannan rakennetta
DROP	poistaa objekteja tietokannasta
TRUNCATE	tyhjentää tietokantataulun
COMMENT	lisää kommentin tietokantaan
RENAME	uudelleennimeää objektin tietokannassa

Data Manipulation Language (DML) -komentoja

SELECT	hakee tietoa tietokannasta
INSERT	lisää tietoa tietokantaan
UPDATE	päivittää olemassa olevaa tietoa tietokannassa
DELETE	poistaa tietoa tietokannasta

Data Control Language (DCL) -komentoja

GRANT	myöntää käyttäjälle käyttöoikeuksia tietokannassa
INVOKE	poistaa GRANT-komennolla myönnettyjä oikeuksia (Oracle FAQ's 2004.)

2.2 Ohjelmointitekniikat

2.2.1 Ohjelmistot

Microsoft Visual Studio 2010

Microsoft Visual Studio on Microsoftin vuonna 1997 ensimmäisen kerran julkaisema usean eri ohjelmointikielen yhteinen ohjelmointityökalu. Ohjelma tukee useita eri ohjelmointikieliä, kuten Visual Basic, C++, C# ja J#. Windows-sovelluksien lisäksi Visual Studiolla voi tehdä myös web- ja mobiilisovelluksia. Ohjelman avulla graafisen käyttöliittymän tekeminen on erittäin yk-

sinkertaista, mikä olikin suurin syy ohjelman valitsemiseen käytettäväksi tässä projektissa. Projektissa käytetään Visual Studio 2010 -versiota.

MySQL Server 5.5

Oraclen omistama MySQL on maailman suosituin ilmainen avoimen lähdekoodin relaatiotietokantaohjelmisto (PHPeasystep 2013). MySQL:n etuina ovat mm. nopeus, luotettavuus, helppokäyttöisyys, skaalautuvuus sekä joustavuus. Tämän projektin tietokantapalvelinohjelmistoksi valittiin MySQL sen takia, että tavaroidenhallintajärjestelmä ei nykyisessä laajuudessaan edellytä suurien tietomäärien käsittelyä tietokannoissa. MySQL Server 5.5:een ei sisälly tämän projektin kannalta juurikaan turhia ominaisuuksia. Esimerkiksi Oraclen tai Microsoftin SQL-palvelinohjelmistot ovat suunniteltu ensisijaisesti suurten tietomäärien käsittelyyn. Raskaammat tietokantapalvelimet asettaisivat myös enemmän vaatimuksia laitekokoonpanoihin. MySQL:ssä käytettävä kieli pohjautuu useimpien muiden relaatiotietokantaohjelmistojen tapaan SQL-kieleen (Structured Query Language) ks. kohta 2.1.2.

.NET Framework

.Net Framework on Microsoftin kehittämä ohjelmistokomponenttikirjasto. .Net komponenttikirjasto on monikielinen ympäristö, joka tukee ainakin 17 eri ohjelmointikieltä, joista käytetyimpiä lienevät C#, C++ ja Visual Basic. Tämän tekniikan yksi suurimmista hyödyistä on se, että luokka voidaan kirjoittaa esimerkiksi C++:lla, ja se voidaan periä C#-koodissa. Yksittäisen koodaajan näkökulmasta eri ohjelmointikielien syntaksit eroavat silti toisistaan, ja tämä vaatii aina orientoitumista. (Kero 2000.)

Sybase Power Designer

Tässä opinnäytetyössä käytetään Sybase Power Designer -ohjelman versiota 12.5. Power Designer on monipuolinen mallinnusohjelmisto, jolla voi mallintaa esimerkiksi relaatiotietokantoja sekä UML- ja liiketoimintaprosesseja. Ohjelma tukee myös XML- ja SOAP-internettekniikoita. (Moon Soft 2013.) Tämän projektin kannalta ohjelma tarjoaa yksinkertaisen tavan mallintaa käyttötapauksia ja suunnitella tietokantaa. Tietokannan suunnittelun lisäksi ohjelma osaa luoda CDM-mallista PDM-mallin ja edelleen luoda tästä skriptin. Skriptin avulla tietokanta tauluineen ja suhteineen luodaan tietokantapal-

velimelle. Näin ollen myös tuki MySQL-palvelinta tukevalle skriptille oli painava syy tämän ohjelman valitsemiseen käytettäväksi tässä projektissa.

2.2.2 Windows Forms

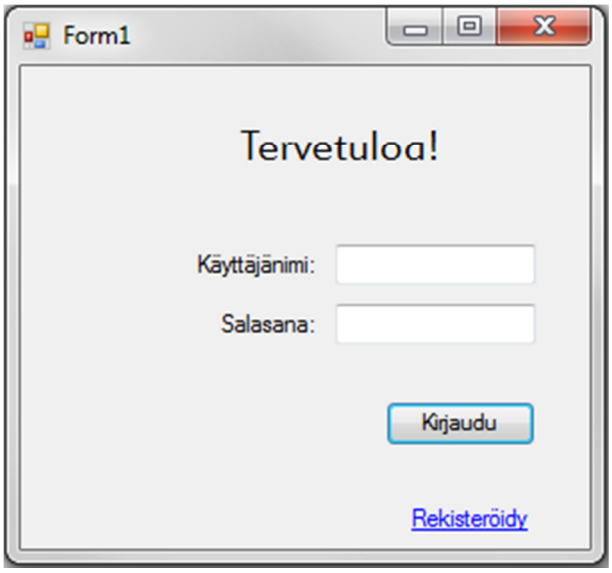
Windows Forms on osa .NET alustaa, joka sisältää GUI-komponentteja, ja niitä käytetään Windows-sovelluksen käyttöliittymän luomiseen (Kero 2000). Windows Forms -sovellukset koostuvat lomakkeista, jotka ovat luokan Forms (System.Windows.Forms) tai siitä periytyvien luokkien instansseja eli olioita/ilmentymiä. Visual Studio luo valmiiksi tiedoston Program.cs, johon Windows Forms -sovelluksen aloituspiste on koodattu. (Kolari 2012.)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace varastonHallinta
{
    static class Program
    {
        /// Ohjelman aloituspiste
        ///

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Pääsivu());
        }
    }
}
```

Esimerkkikoodi 4. Windows Forms -sovelluksen aloituspiste (Kolari 2012)

Esimerkkikoodi 4:ssä on esitetty Windows Forms -sovelluksen aloituspiste. System.Windows.Forms.Application-luokan EnableVisualStyles- ja SetCompatibleTextRenderingDefault -metodikutsujen jälkeen kutsutaan Application-luokan metodia Run, jolle välitetään parametrina esittely uudesta Pääsivu-tyypistä. Run-metodi käynnistää näin siis Forms-sovelluksen.

Kyseinen tekniikka valittiin tavaroidenhallintajärjestelmän kehittämiseen sen takia, että sillä saadaan toteutettua Windows-työpöytäohjelmia. Siitä myös löytyy runsaasti valmiita käyttöliittymäkomponentteja, joita tavaroidenhallintajärjestelmän käyttöliittymässä tarvitaan. Lisäksi ”formeissa”, eli ikkunoissa käytetty ulkoasu on entuudestaan tuttu Windows-käyttöjärjestelmistä.

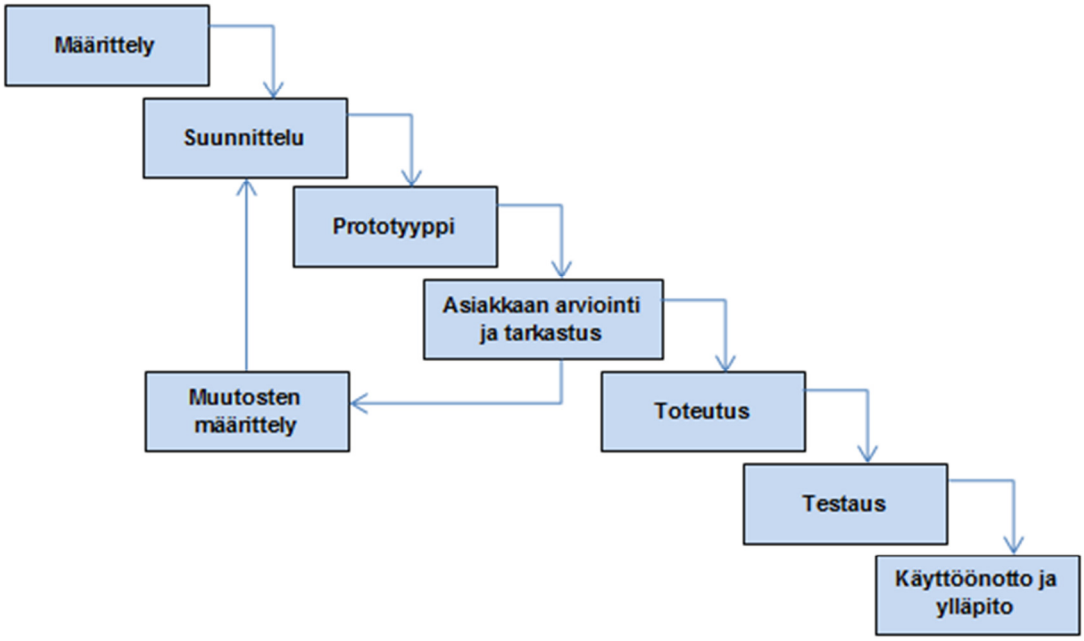


Kuvio 1. Windows Forms -lomake

Kuvio 1 on esimerkki hyvin yksinkertaisesta Windows Forms -lomakkeesta. Siinä on kolme Label-, kaksi TextBox-, yksi Button- ja yksi LinkLabel -GUI-komponenttia. TextBox on tekstilaatikko, johon käyttäjä voi kirjoittaa tekstiä. Tässä tapauksessa Kirjaudu-nappi laukaisisi kirjautumisyrityksen käyttäjänimi- ja salasana -kenttien syötteiden perusteella. Windows Forms -tekniikka soveltuu hyvin tämän projektin toteuttamistekniikaksi sen selkeän ja yksinkertaisen ulkoasun takia. Lisäksi yksinkertainen ohjelmoitavuus on painava kriteeri tämän tekniikan valinnassa juuri tähän projektiin.

2.3 Protoilumalli

Protoilumalli on ohjelmistotuotannon projektin läpiviemiseen kehitetty ns. ketterä menetelmä. Kuten kaikki ketterät menetelmät, protoilumallilla pyritään saamaan projektin läpivientiin dynaamisuutta ja selkeyttä. Ohjelmiston kehittämisprosessissa voi ilmetä erilaisia vastoinkäymisiä ja viivästyksiä, joita ei aina voi ennalta arvata. Tästä syystä ketterät menetelmät ovat hyvä tapa kehittää ohjelmistoja, koska niissä yleensä tavalla tai toisella on mahdollisuus palata prosessissa montakin askelta taaksepäin ja pitää silti prosessi kontrollissa. Protoilumalli sopii tähän projektiin hyvin, koska kehitettävä järjestelmä ei ole laajuudeltaan kovin iso. Näinpä siitä saa kohtuullisella työmäärällä toteutettua prototyyppejä. Kuten muissakin ketterissä menetelmissä, myös protoilumallissa ovat omat askeleet, joiden mukaan projekti etenee.



Kuvio 2. Protoilumalli

Kuvio 2 esittää protoilumallin ja sen askeleet. Kuten valtaosassa projekteja, se alkaa määrittelyvaiheella, jossa kerätään asiakkaan vaatimukset kehitettävälle ohjelmalle. Tätä vaihetta seuraa suunnittelu, jossa asiakkaalla ei ole kovin suurta roolia. Suunnittelussa pureudutaan syvemmin siihen, miten ohjelma toteutetaan, eli päätetään käytettävät tekniikat ja ohjelmointiympäristöt.

Seuraavassa vaiheessa tehdään ohjelmasta ensimmäinen prototyyppi. Prototyyppi auttaa kehittäjää ja asiakasta pääsemään yhteisymmärrykseen asiakkaan ohjelmalle asettamista käytännön vaatimuksista. Prototyyppi toimii tässä tapauksessa siis ikään kuin täydentäjänä määrittelyvaiheelle. Asiakas tarkastaa ja arvioi prototyyppiä yhdessä kehittäjän kanssa. Useimmiten tässä vaiheessa löytyy muutos- ja/tai parannusehdotuksia. Sen jälkeen muutokset määritellään ja siirrytään takaisin suunnitteluvaiheeseen, jonka jälkeen muutokset toteutetaan uudessa prototyypissä. Jos prototyypissä on asiakkaan mielestä vielä korjattavaa, tehdään sama prosessi uudestaan. Jos prototyyppi on asiakkaan mielestä toteuttamiskelpoinen, siirrytään toteutusvaiheeseen, jossa itse ohjelma koodataan ja toteutetaan käytännössä.

Toteutusvaihetta seuraa testaus, jossa mahdolliset toiminnalliset virheet ja epäkohdat pyritään saamaan selville mahdollisimman yksityiskohtaisella ja kattavalla testauksella. Testauksen ja mahdollisten korjausten jälkeen ohjelma otetaan käyttöön. Tätä seuraa ylläpitovaihe, jossa käytössä olevalle oh-

jelmalle tarjotaan ylläpidollista tukea. Ylläpitovaihe kestää asiakkaan kanssa sovitun ajan, tai sitä ei välttämättä ole ollenkaan.

3 MÄÄRITTELY

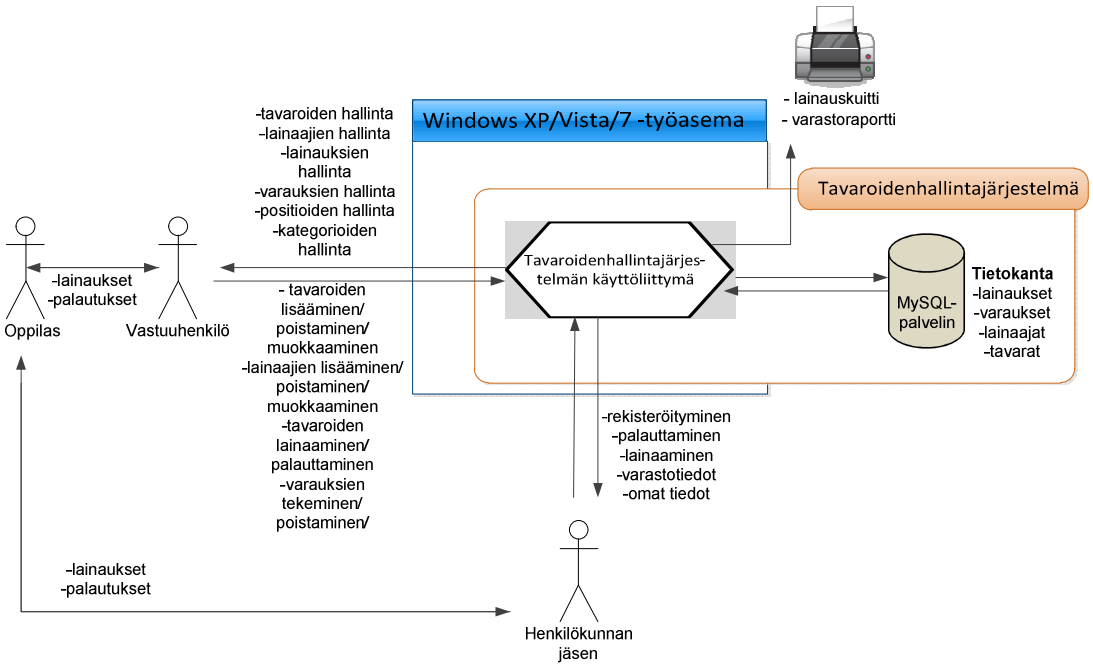
3.1 Yleiskuvaus ja ympäristö

Tavaroidenhallintajärjestelmän päätarkoitus varastonhallintatyökaluna toimimisen lisäksi on toimia lainausautomaattina, joka lainaustilanteessa merkitsee tavaran lainatuksi ja tallentaa tiedon rekisteriin. Järjestelmä toimii myös lainaajarekisterinä, ja siitä on helposti saatavissa selville lainaajan yhteystiedot, mikäli esimerkiksi lainatun tavaran eräpäivä on mennyt umpeen. Eräpäivän mennessä umpeen järjestelmä antaa ilmoituksen sovelluksen käyttöliittymässä, kun käyttäjä kirjautuu sisään sovellukseen.

Kukin tavara on yksilöity omalla viivakoodillaan. Viivakoodinlukija esitellään kohdassa 3.5.1. Lainaajat yksilöidään omalla tunnisteellaan, joka oppilaiden tapauksessa on opiskelijanumero ja muussa tapauksessa käyttäjän tietyin ehdoin itse keksimä tunniste.

Ympäristö

Järjestelmän käyttöympäristöön kuuluu useita osia, kuten eri käyttäjätyypit, käyttöliittymää pyörittävä työasema, tietokanta jne.



Kuvio 3. Tavaroidenhallintajärjestelmän ympäristökaavio

Kuviossa 3 on esitetty tavaroidenhallintajärjestelmän ympäristön päärajapinnat ja toimijat. Oppilas voi käyttää järjestelmää ainoastaan joko vastuuhenki-

lön tai henkilökunnan jäsenen välityksellä suorittaen tavaroiden lainaamisen ja palauttamisen. Vastuuhenkilö pääsee hallitsemaan tavaroidenhallintajärjestelmässä olevaa tietoa suoraan työasemalla toimivan käyttöliittymän kautta. Myös henkilökunnan jäsen voi käyttää järjestelmää käyttöliittymän kautta, mutta hänellä on siihen vain rajoitetut oikeudet. Hänellä on oikeus oman rekisteröitymisen ja tavaroiden lainaamisen sekä palauttamisen lisäksi tarkastella tavaroiden varasto-, lainaus-, sekä varaustietoja. Lisäksi hän pystyy muokkaamaan omia tietojaan, jotka on tallennettu järjestelmään.

Järjestelmästä on mahdollisuus tulostaa lainauskuitti silloin, kun tavara lainataan. Lisäksi tavaroista, lainaajista, lainauksista ja varauksista voidaan tulostaa raportteja. Tietojen tallennuspaikkana toimii MySQL-tietokanta, johon työasema on yhteydessä Internet- tai lähiverkkoyhteyden kautta.

3.2 Käyttäjätyypit

Järjestelmällä on kaksi varsinaista konkreettista käyttäjätyyppiä; vastuuhenkilö ja henkilökunnan jäsen. Toisaalta järjestelmän käyttäjäksi voidaan lukea myös oppilas, sillä sekin kuuluu järjestelmään tallennettavien lainaajatyyppien joukkoon. Oppilas ei kuitenkaan lainausprosessin aikana itse käytä järjestelmää muuta kuin kirjastosta tutulla tavalla joko vastuuhenkilön tai henkilökunnan jäsenen toimiessa ikään kuin kirjastovirkailijana.

Vastuuhenkilö

Vastuuhenkilö vastaa tavaroiden lisäämisestä järjestelmään ja hän myös pitää tavaroiden tiedot ajan tasalla. Lisäksi hän toimii ensisijaisena välikätenä lainaustilanteessa ikään kuin ”kirjastonhoitaja”. Vastuuhenkilö pystyy järjestelmässä lisäämään, muokkaamaan ja poistamaan niin tavaroita, lainaajia, kuin myös varauksia ja lainauksiakin. Lainauksien poistaminen tapahtuu yleensä automaattisesti silloin kun tavara palautetaan, mutta vastuuhenkilöllä täytyy olla keino poistaa lainaus myös silloin, jos esimerkiksi tavaran viivakoodi vaurioituu eikä sitä voi enää lukea. Näiden lisäksi vastuuhenkilöllä on mahdollisuus lisätä, poistaa ja muokata olemassa olevia tavarakategorioita ja tavaroiden sijainti-positioita. Positio voi olla esimerkiksi jokin kaappi, jossa säilytetään tietyntyyppisiä tavaroita, kuten lämpökameroita.

Henkilökunnan jäsen

Henkilökunnan jäsen -tili on luotu järjestelmään siltä varalta, mikäli lainauksia tarvitsee tehdä silloin, kun itse vastuuhenkilö ei ole paikalla. Henkilökunnan jäsenellä on oikeus ainoastaan palauttaa ja lainata tavaroita sekä selata tavaroiden tietoja ja lainaus-/varaustilannetta. Henkilökunnan jäsen on siis kuka tahansa Rovaniemen ammattikorkeakoulun henkilökunnasta tai tilojen käyttöön oikeutetuista henkilöistä paitsi oppilaista. Henkilöllä tulee kuitenkin olla käyttäjätunnus koulun verkkoon päästäkseen käyttämään työasemaa, jossa sovellus on asennettuna.

Oppilas

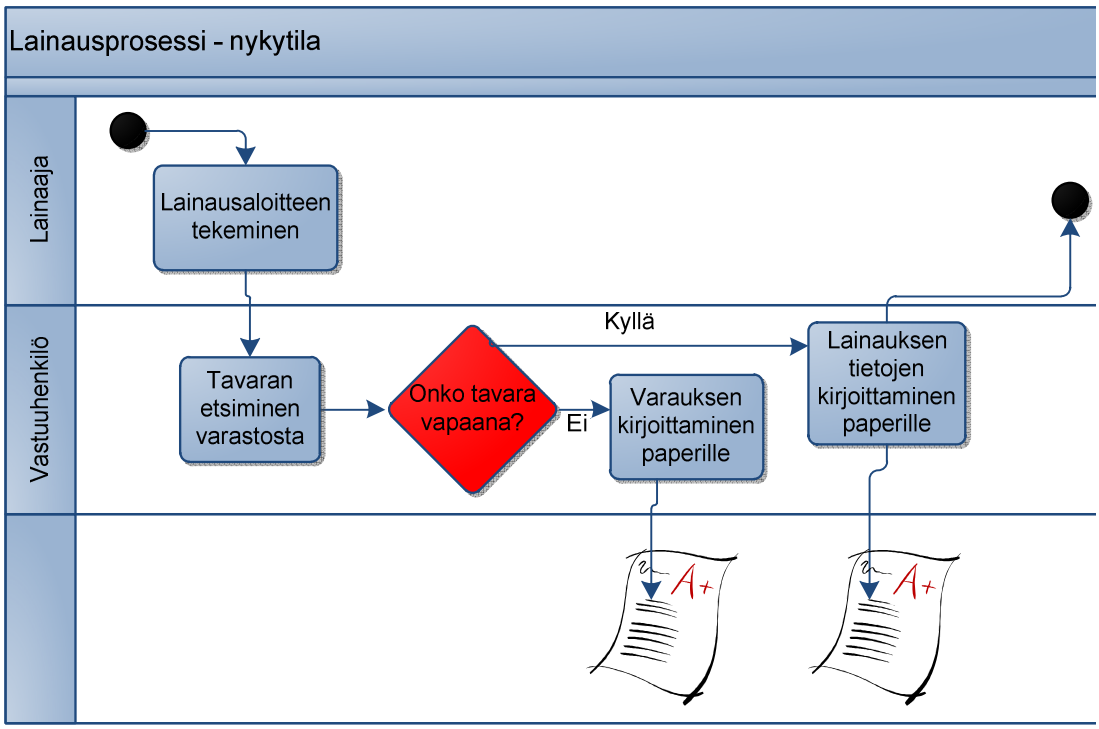
Oppilas ei ole järjestelmän ”perinteisen” kaltainen käyttäjä. Oppilas kuitenkin luetellaan käyttäjätyypiksi, sillä vaatimus on, että oppilaatkin pystyvät lainaamaan tavaroita laboratorion tiloista. Oppilas, kuten kuka tahansa muukin tavaroita lainaava henkilö, saa lainauksestaan mukaan tulostettavan lainauskuitin. Kuitti sisältää oleellisia tietoja lainauksesta, kuten lainauspäivämäärän, eräpäivän ja lainatun tavaran nimen.

3.3 Toiminnalliset vaatimukset ja prosessit

3.3.1 Lainausten hallinta

Nykytila

Tällä hetkellä lainausprosessi on vastuuhenkilölle erittäin työläs ja epätarkka. Myöskään ajan tasalla olevaa varastotilannetta ei käytännössä ole koskaan tiedossa, pois lukien inventaariopäivä, jolloin kaikki tavarat on kartoitettu ja kirjattu Excel-taulukoihin.

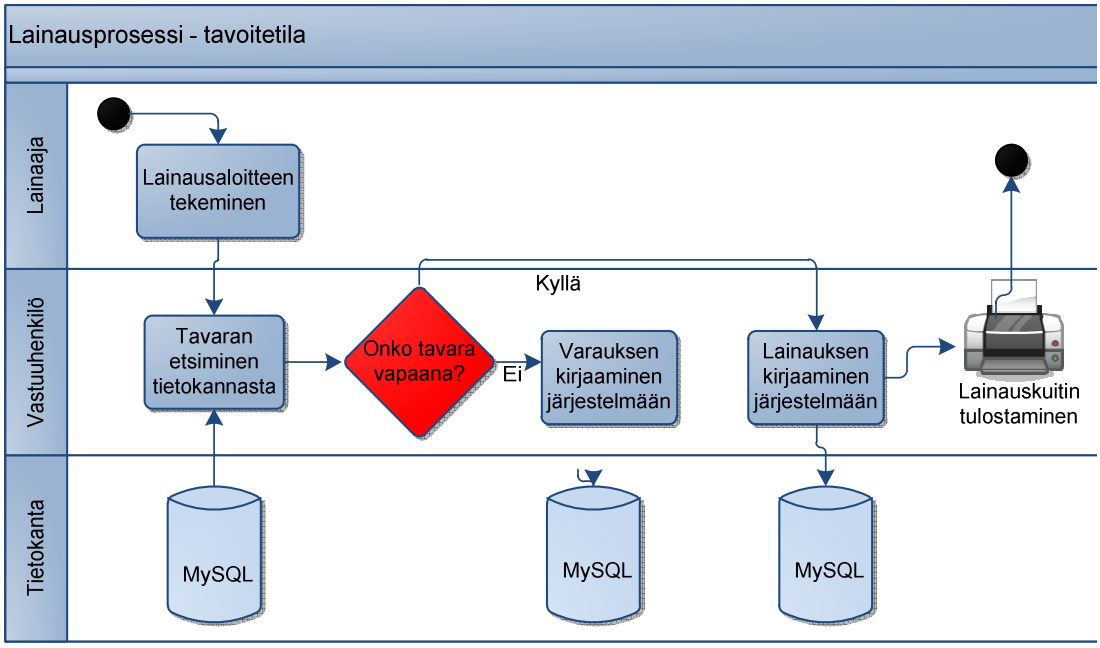


Kuvio 4. Lainausprosessin nykytila

Lainausprosessi on tähän mennessä toiminut kuvion 4 esittämällä tavalla. Lainaajan halutessa lainata jokin tavara, on vastuuhenkilön täytynyt ensin etsiä ja ottaa selville, onko tavara varastossa ja vapaana. Sen lisäksi jos tavara on jokin mittalaite, on täytynyt selvittää Excel-taulukoista, onko laite kalibroitu ajallaan. Tämän jälkeen on voitu tehdä lainaus, jonka vastuuhenkilö on kirjoittanut paperille kirjaten ylös lainatun tavaran, lainauspäivämäärän, sekä lainaajan nimen ja yhteystiedot. Kun tavara on palautettu, on vastuuhenkilö ainoastaan ottanut tavaran vastaan ja tuhonnut lainaustiedot sisältävän paperin.

Tavoitetila

Järjestelmän tavoitetilassa pyritään kirjanpidon tarkentamisen ja eheyden parantamisen lisäksi pitämään huoli siitä, että lainaukset tulee varmasti kirjattua muistiin. Näin ollen vältetään tilanteet, jolloin kukaan ei tiedä, missä tavara on, tai kenelle se on lainattu.



Kuvio 5. Lainausprosessin tavoitetila

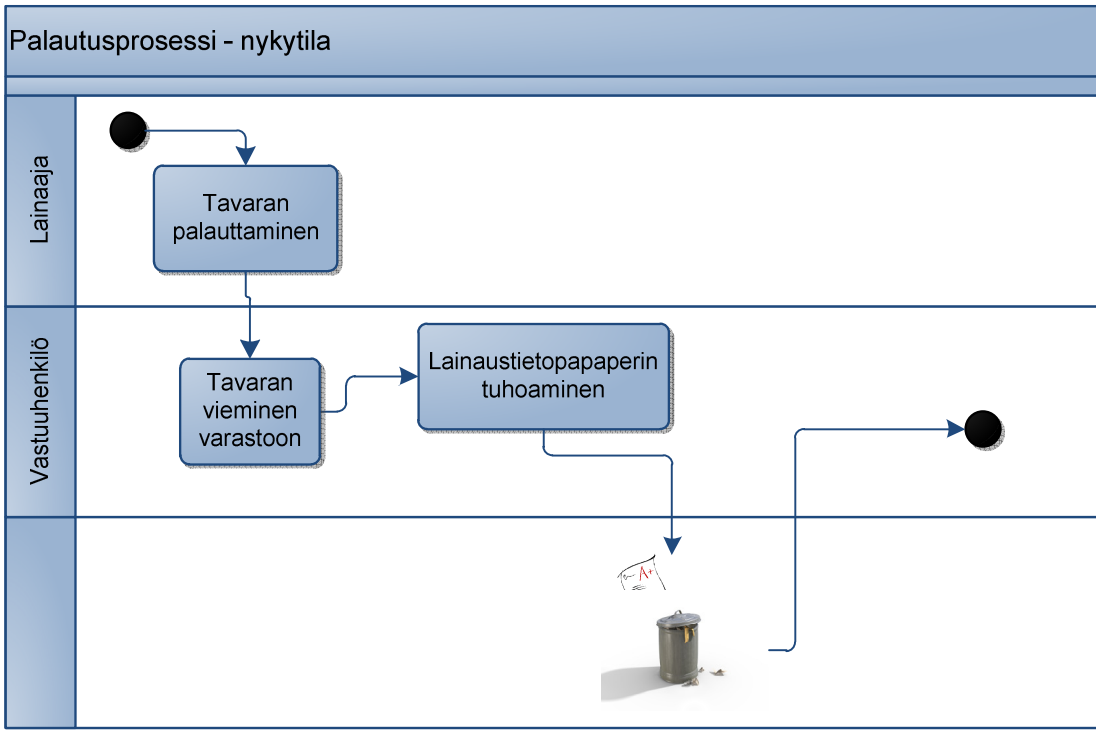
Tavoite on saada lainausprosessi toimimaan kuvion 5 esittämällä tavalla. Vaikka tavoitetila ensisilmäyksellä näyttää kaaviossa monimutkaisemmalta nykytilaan verrattuna, on se kuitenkin nopeampi ja ennen kaikkea täsmällisempi tapa kuin nykyinen. Lisäksi tavoitetilaan yhdistyy myös varastonhallinta. Tavoitetilassa kolmanneksi toimijaksi toteutetaan MySQL-tietokanta, joka ylläpitää kaikkia tietoja järjestelmään liittyen. Lainaajan halutessa lainata tavaroita vastuuhenkilö tai henkilökunnan jäsen etsii ne tietokannasta haluamillaan ehdoilla. Hän näkee listauksesta mitkä ovat vapaina, ja missä tilassa ja missä positiossa se on. Mikäli se on vapaana, vastuuhenkilö lukee viivakoodinlukijalla tavaran viivakoodin sekä syöttää lainaajan tunnisteen ja halutun eräpäivän sovellukseen. Sen jälkeen hyväksytään lainaus, ja sovellus luo lainauskuitin. Vastuuhenkilö tulostaa kuitin joko paikallisella tulostimella tai verkkotulostimella ja ojentaa sen lainaajalle. Näin ollen järjestelmä on koko ajan tietoinen siitä, mitä tavaroita on lainassa ja kenellä, sekä mikä on lainauksen eräpäivä.

Lainausta tehdessään vastuuhenkilö näkee tavaralistalta myös, mikäli tavara on lainassa, tai se on varattu jollekin muulle lainaajalle. Tällöin on mahdollisuus tehdä varaus.

3.3.2 Palautusten hallinta

Nykytila

Palautusten hallinnan osalta kirjanpitoa ei ole ollut käytännössä lainkaan. Tämä on lisännyt aikaa, joka kuluu tavaran etsimiseen ja uudelleen lainaamiseen.

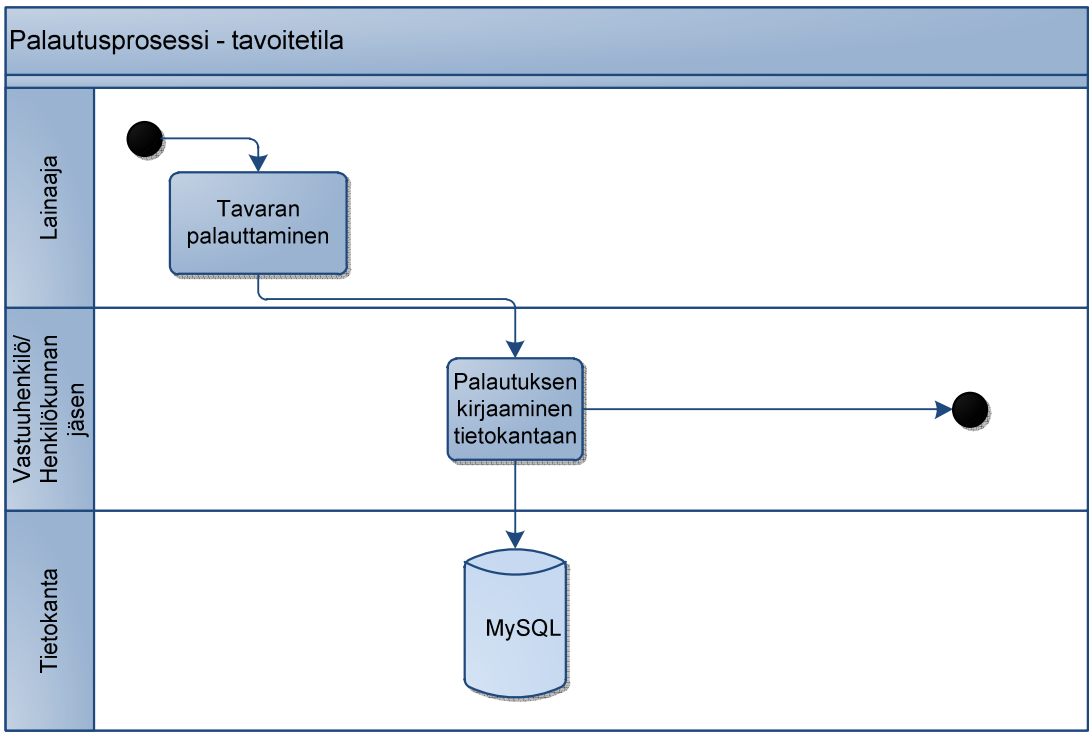


Kuvio 6. Palautusprosessin nykytila

Kuten kuviosta 6 voidaan päätellä, myös tavaran palauttaminen on suoritettu erittäin alkeellisella tavalla. Tietoa tavaran palautuksesta ei ole voitu rekisteröidä mihinkään, vaan se on viety takaisin varastoon ja lainaustietopaperi tuhottu. Näin ollen kenelläkään ei ole ollut ajantasaista tietoa siitä, onko tavara kyseisellä hetkellä lainassa, varattuna vai vapaana.

Tavoitetila

Palautusprosessin tavoitetilassa tieto tavaran palauttamisesta kirjautuu automaattisesti tietokantaan, kun tavara palautetaan.



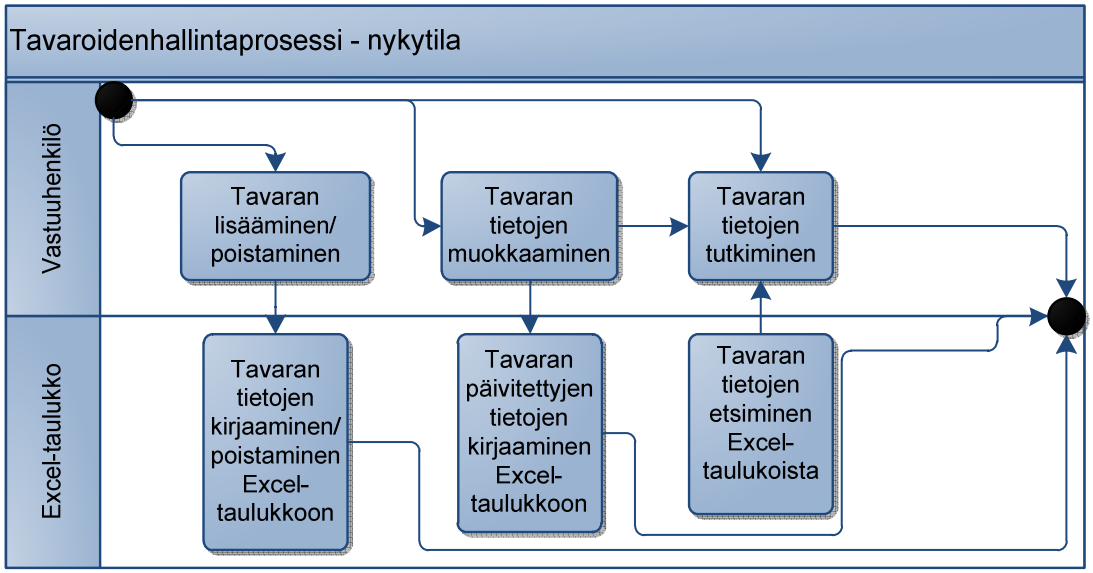
Kuvio 7. Palautusprosessin tavoitetila

Palautusprosessin tavoitetila tavaroidenhallintajärjestelmässä on kuvion 7 kaltainen. Palautustilanteessa tavaran voivat järjestelmän avulla palauttaa joko sekä vastuuhenkilö tai henkilökunnan jäsen. Ennen kaikkea kehitysaskel on se, että järjestelmä päivittää varastotilanteen ja merkitsee tavaran palautetuksi tietokantaan. Tästä eteenpäin järjestelmästä voidaan tutkia, onko tavaralla varauksia, ja jos on, niin kuka on varaaja. Varaajasta puolestaan löytyvät tarvittaessa varaajan yhteystiedot, ja näin häntä voidaan tiedottaa siitä, että tavara on lainattavissa. Vastuuhenkilön kirjauduttua sovellukseen järjestelmä antaa hälytyksen sovelluksen käyttöliittymässä, jos jonkin lainauksen eräpäivä on mennyt umpeen.

3.3.3 Tavaroiden hallinta

Nykytila

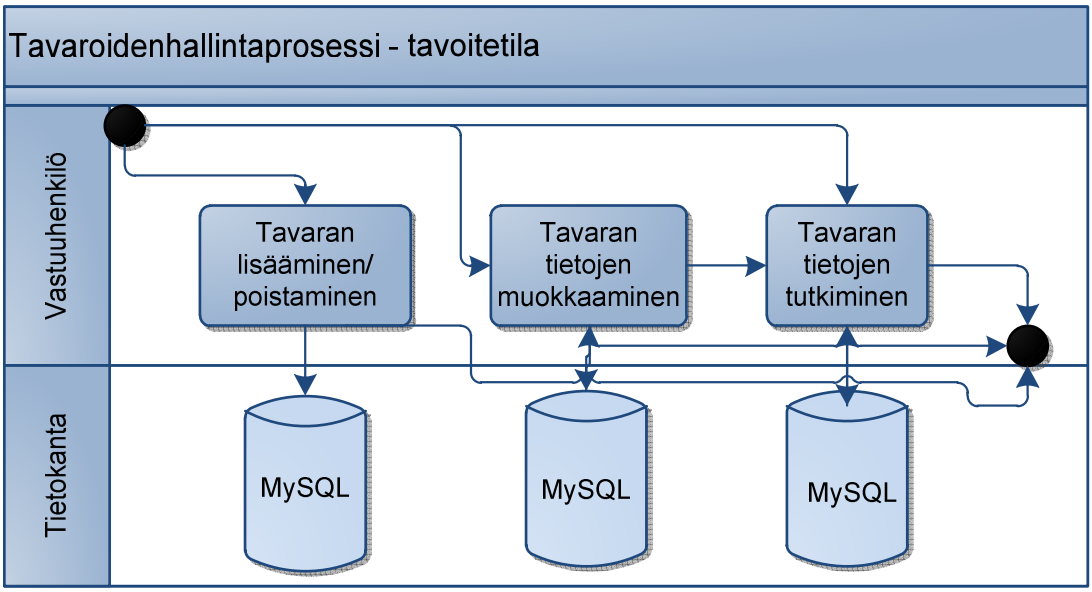
Tähän mennessä tavaroita on hallittu Excel-taulukoiden avulla, joihin on kirjattu oleellisimpia tietoja tavaroista, ja tämä on vienyt paljon aikaa.



Kuvio 8. Tavaroidenhallintaprosessin nykytila

Kuvio 8 osoittaa, että tähän mennessä tavaroiden tiedot ovat olleet ainoastaan Excel-taulukoissa. Tavaroista pidettäviä tietoja ovat viivakoodi, nimi, hankintapäivämäärä, kalibroimispäivämäärä, seuraava kalibroimispäivämäärä sekä inventaariopäivämäärä. Lisäksi tiedossa on ollut positio, joka viittaa esimerkiksi laboratorion tiloissa oleviin eri kaappeihin. On myös huomion arvoista, että tavaroilla ei välttämättä aina ole positiota. Joitakin tavaroita saattavaa olla myös lattialla, mutta silti ne on saatettu määritellä tiettyyn positioon. Tämä vaikeuttaa varastonhallintaa ja seuraavaksi esitellään ratkaisu siihen.

Tavoitetila



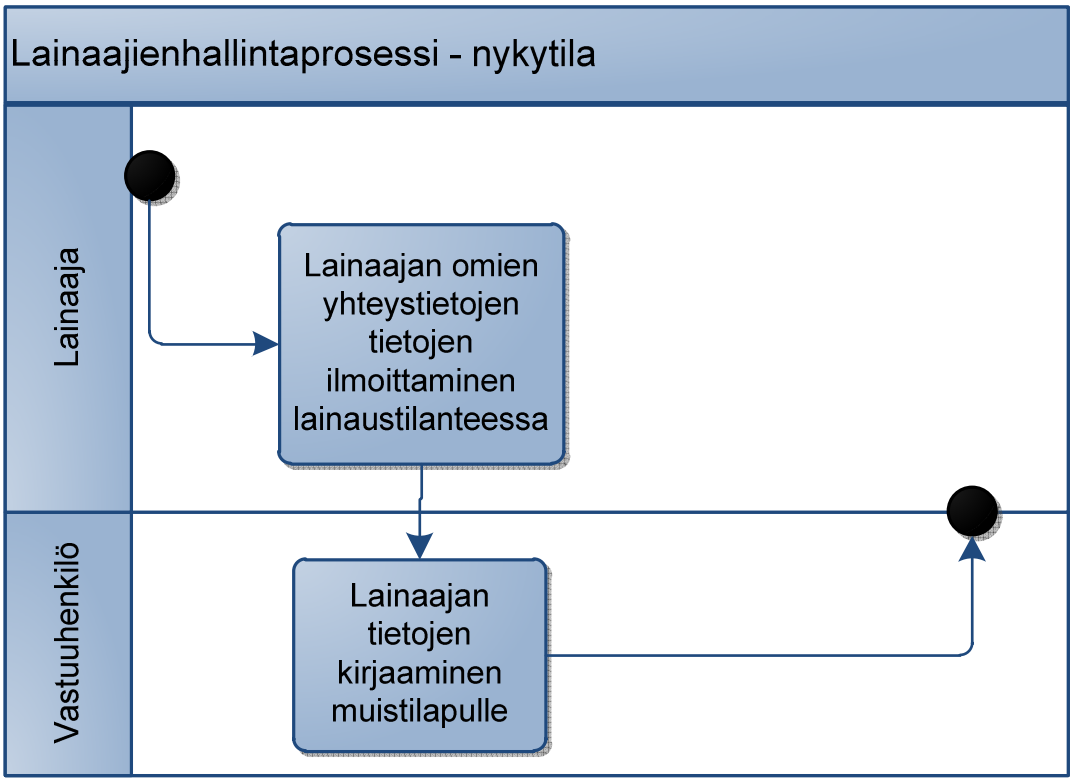
Kuvio 9. Tavaroidenhallintaprosessin tavoitetila

Kuvio 9 esittää pääpiirteittäin tavaroidenhallintaprosessin tavoitetilan. Tavaroiden tiedot tallennetaan järjestelmän käyttöliittymän avulla MySQL-tietokantaan, josta tiedot saadaan helposti haettua. Tavoitetilassa tavaroista tallennetaan aiempaa tarkempi määritelmä siitä, missä tilassa ja missä positiossa tavara on. Tämän lisäksi tavarat jaetaan omiin kategorioihin. Esimerkiksi lämpökamera kuuluu käsimittari-kategoriaan. Positio voi olla esimerkiksi ”kone1”, eli konelaboratorion kaappi numero 1. Tilana voi olla joko automaatiolaboratorio tai konelaboratorio. Tällä tavalla saadaan aiempaa tarkempi tieto siitä, missä tavara milloinkin on. Jako kategorioihin auttaa varastonhallintaa sen tuoman uuden luokitteluperusteen avulla. Kategorioita ja positioita voidaan myös lisätä, muokata ja poistaa tarpeen vaatiessa.

3.3.4 Lainaajien hallinta

Nykytila

Nykytilassa lainaajista ei ole jäänyt mitään jälkiä kirjanpitoon sen jälkeen, kun tavara on palautettu.

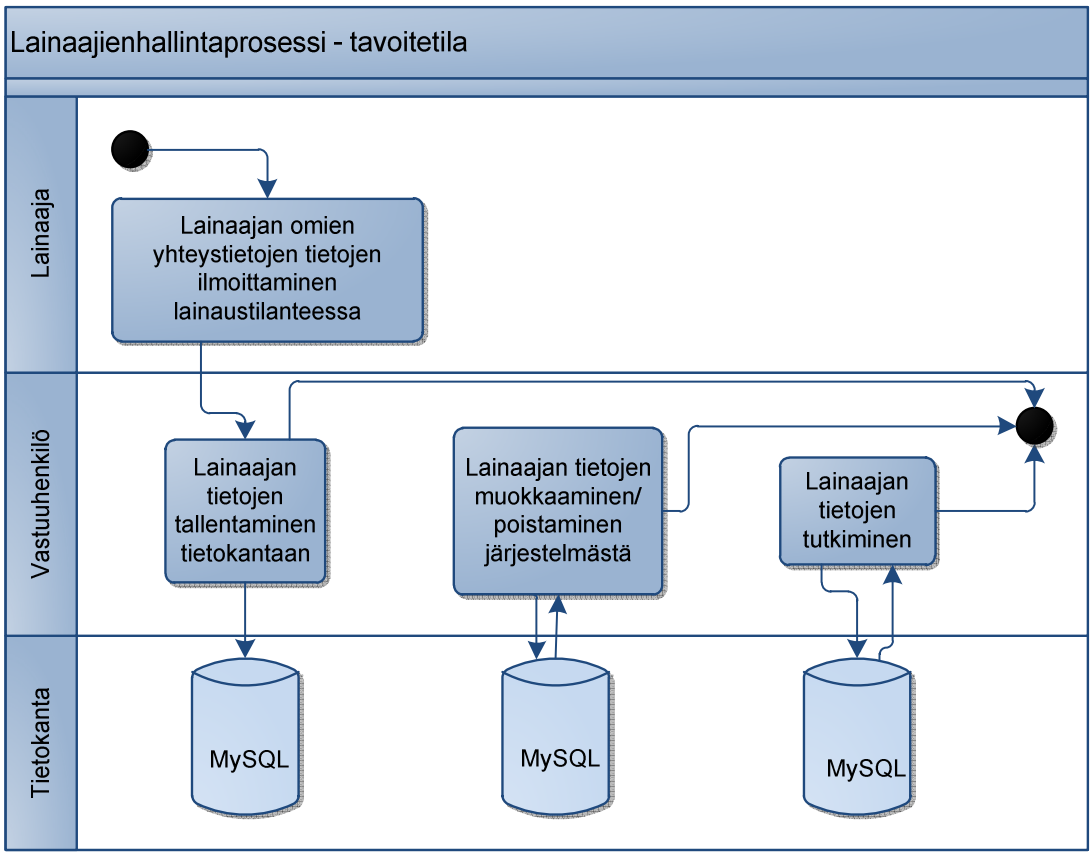


Kuvio 10. Lainaajienhallintaprosessin nykytila

Kuten kuvio 10 osoittaa, käytännössä minkäänlaista lainaajienhallintaa ei tähän mennessä ole ollut olemassa. Näin ollen myöskään ei ole kirjanpitoa lainaushistoriasta, mikä olisi tärkeää monessa tapauksessa.

Tavoitetila

Tavoitetilassa jokaisen lainaajan täytyy rekisteröityä lainaajaksi järjestelmään, jotta lainaaminen onnistuu. Tämä tarkoittaa lainaajan tietojen tallentamista järjestelmään, mikä taas on suuri apu siinä vaiheessa, jos on tarve saada tietää tietyn tavaran lainaushistoria.



Kuvio 11. Lainaajienhallintaprosessin tavoitetila

Kuvion 11 esittämässä tavoitetilassa lainaajienhallinta on toteutettu siten, että vastuuhenkilöllä on mahdollisuus rekisteröidä uusia lainaajia järjestelmään. Lainattaessa tavaroita lainaajan tietoihin päivittyy automaattisesti myös lainaushistoria. Vian ilmentyessä tavaraan ja epäiltäessä, että se on tapahtunut edellisen lainassaolon yhteydessä, voidaan järjestelmästä helposti etsiä tavaran edellisen lainaajan yhteystiedot.

Tallennettavat tiedot

Kaikilta lainaajilta rekisteröidään tunniste, etu- ja sukunimi sekä puhelinnumero ja/tai sähköpostiosoite. Jos lainaaja on opiskelija, tallennetaan järjestelmään opiskelijanumero, joka toimii opiskelijan ainutkertaisena tunnisteena järjestelmässä. Aina kun opiskelija lainaa tavaran, syötetään sovellukseen hänen opiskelijanumeronsa, ja näin järjestelmä osaa rekisteröidä lainauksen oikealle lainaajalle. Opiskelijalle ei tallenneta salasanaa, koska hän ei missään vaiheessa käytä järjestelmän käyttöliittymää.

Jos lainaaja on henkilökunnan jäsen, toimii hänen ainutkertaisena tunnistimenaan jokin tämän itse keksimänsä tunniste. Tunniste täytyy olla 6–12 merkkiä pitkä, ja siinä saa olla isojen ja pienien kirjainten lisäksi myös numeroita. Rekisteröityessä järjestelmä antaa virheilmoituksen, mikäli syötetty tunniste on jo rekisteröity järjestelmään. Henkilökunnan jäseneltä vaaditaan myös salasana, jossa tulee olla 6–10 merkkiä, ja se saa sisältää isojen ja pienien kirjainten lisäksi myös numeroita.

3.4 Erityisvaatimukset ja rajoitukset

Kehitettäessä sovellusta monitasoisille käyttäjille, tulee erityisesti käytettävyyteen liittyvät asiat ottaa huomioon. Toteutettavan järjestelmän tulee olla helppokäyttöinen niin, että perustason tietoteknisillä taidoilla kuka tahansa sovelluksen käyttäjä ymmärtää sovelluksen antamasta informaatiosta sen, mitä tulee tehdä päästäkseen haluttuun lopputulokseen. Prosessit eivät myöskään saa olla liikaa aikaa vieviä, sillä se vähentää käyttömukavuutta merkittävästi. Käyttöliittymäkomponenttien tulee olla loogisesti sijoiteltuna käyttöliittymään, ja turhien tietokenttien käyttöä tulee välttää. Sovelluksen logiikka ja käyttöliittymä eivät saa olla raskaita ja hitaita, vaan niiden tulee toimia mahdollisimman sulavasti ja tehokkaasti. Sovelluksen tulee myös olla helposti ja nopeasti asennettavissa työasemalle.

Järjestelmän käyttämiselle on määritelty myös laadullisia vaatimuksia. Nämä vaatimukset ovat aikakehyksiä, joiden puitteissa tietyt toiminnot on saatava tehtyä:

- Käyttäjän on kyettävä lainaamaan tavara 30 sekunnissa, jos tavara ja lainaaja on jo tallennettu järjestelmään ja tavara on vapaana varastossa.
- Käyttäjän on kyettävä lainaamaan tavara 120 sekunnissa, jos vain tavara on tallennettu järjestelmään.
- Käyttäjän on kyettävä lainaamaan tavara 90 sekunnissa, jos vain lainaaja on tallennettu järjestelmään.
- Käyttäjän on kyettävä lainaamaan tavara 180 sekunnissa, jos tavaraa ja lainaajaa ei ole vielä tallennettu järjestelmään.

- Käyttäjän on kyettävä palauttamaan tavara 15 sekunnissa.
- Sovelluksen avaaminen ja sisäänkirjautuminen ei saa olla monimutkaista. Avaamiseen saa kulua enintään 5 sekuntia ja sisäänkirjautumiseen 5 sekuntia. Sovelluksen tulee siis olla käytettävissä viimeistään 10 sekunnin päästä ohjelman exe-tiedoston avaamisesta olettaen, että käyttäjällä on perustaidot Windows-ympäristöistä.

Järjestelmävaatimukset

Tavaroidenhallintajärjestelmän käyttöliittymäsovellus ei saa viedä työaseman kovalevytilaa yli 30 megatavua. Lisäksi on määritelty, että 1Gb DDR-tyyppistä keskusmuistia pitää riittää käyttöliittymäsovelluksen pyörittämiseen työasemalla. Työasemalla täytyy myös olla laajakaistatasoinen Internet- tai lähiverkkoyhteys.

Rajoitukset

Määrittelyvaiheessa toimeksiantajan toimesta nousi esiin toive toteuttaa tavaroidenhallintajärjestelmän käyttöliittymä myös mobiili-laitteelle/-laitteille. Kuitenkin projektin ollessa laajahko jo nykyisessä muodossaan, päätettiin toteutus tehdä ainoastaan Windows XP-, Windows Vista- sekä Windows 7 -työasemille.

Edellisen lisäksi toiveena oli, että järjestelmää pystyisi käyttämään myös offline-tilassa, mutta niin ikään sekin ominaisuus jätettiin mahdolliseen jatkokehitykseen. Em. ominaisuuksien lisäksi myös käyttöliittymän skaalautuvuus työaseman näytön koon muuttuessa jätetään tässä versiossa toteuttamatta. Rajoituksena voitaneen mainita myös, että toteutettavassa versiossa edes vastuukäyttäjällä ei ole mahdollisuutta muokata tietokantatauluja tai tavaroiden tietokenttien nimiä käyttöliittymän kautta, vaan niiden muuttamiseksi vaaditaan oikeudet MySQL-palvelimelle.

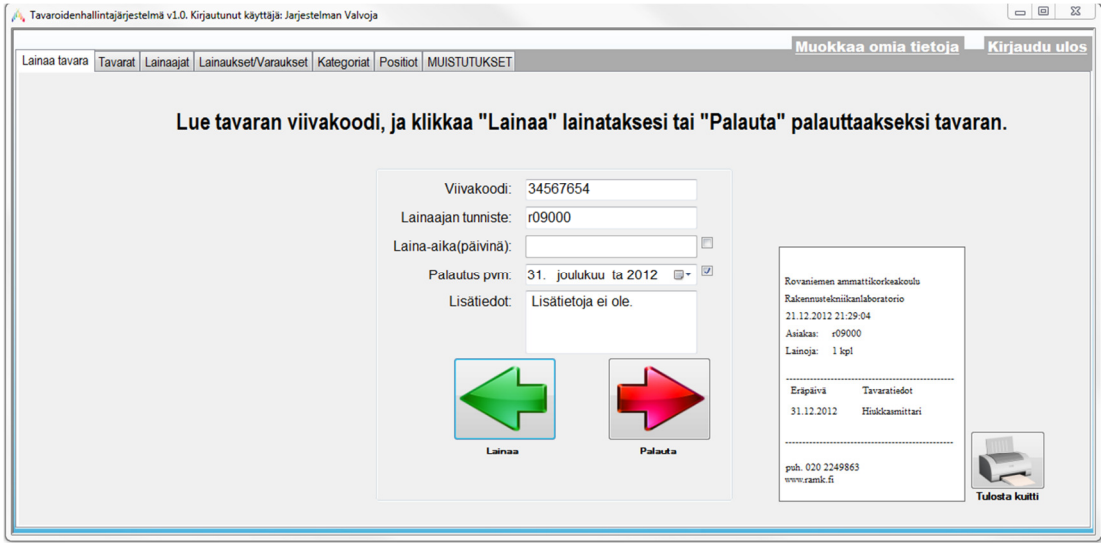
Sovelluksen virheviestien tulee olla mahdollisimman selkeitä ja informatiivisia. Niitä tulee käyttäjän antamista vääränlaisista syötteistä tai MySQL-tietokantayhteyden ongelmista. Virheviestejä tulee myös esimerkiksi yrittäes-

sä poistaa tietokannasta jotain objektia, jolla on viite toiseen tietokantatauluun.

3.5 Tekninen määrittely

3.5.1 Järjestelmän yleiskuvaus

Tavaroidenhallintajärjestelmän käyttöliittymä koostuu yhdestä pääikkunasta, joka on jaettu seitsemään ulkoasultaan samanlaiseen välilehteen.



Kuvio 12. Tavaroidenhallintajärjestelmän pääikkuna

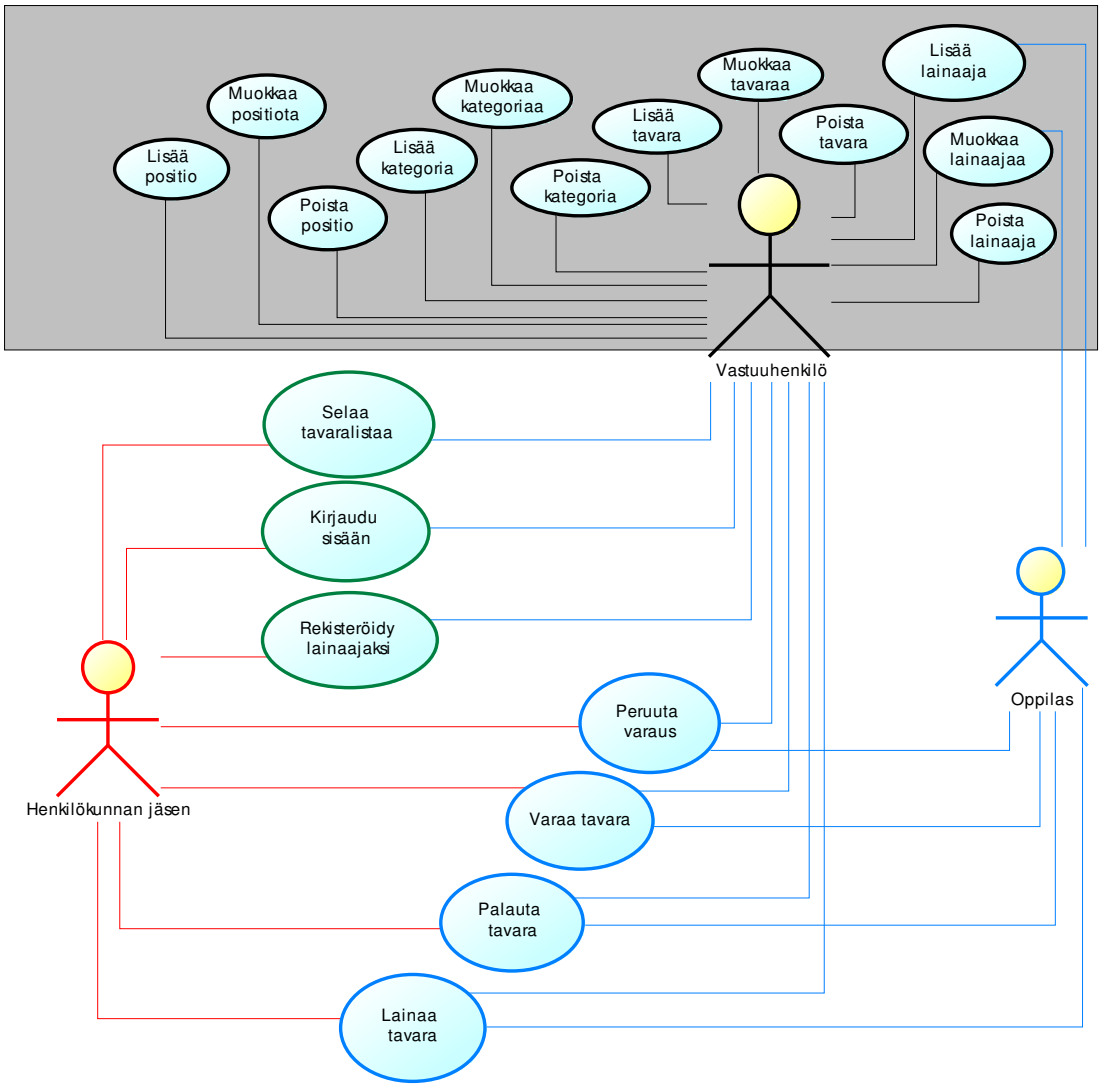
Kuvio 12 esittää Pääikkunan välilehtineen, jotka ovat seuraavat: Lainaa tavara, Tavarat, Lainaajat, Lainaukset/Varaukset, Kategoriat, Positiot ja MUISTUTUKSET. Lisäksi on kirjautumisikkuna, rekisteröitymisikkuna sekä lisäys- ja muokkausikkuna tavaroille, lainaajille, kategorioille sekä positioille. Sisäänkirjautuneen käyttäjän on mahdollista muokata myös omia tietojaan Muokkaa omia tietoja -linkin kautta avautuvassa omassa ikkunassa. Em. ikkunat avautuvat aina pääikkunan päälle. Tällöin pääikkuna asetetaan pois käytöstä eli ”disabloidaan”, ja otetaan jälleen käyttöön kun lisäikkuna suljetaan.

Viivakoodinlukija

Tallennettaessa tavaroiden tietoja ja rekisteröitäessä lainauksia, viivakoodin lukija helpottaa viivakoodin lukemista ja syöttämistä. Tässä projektissa käytetään USB-porttiin liitettävää Manhattan SD313B SmartPro CCD -

viivakoodinlukijaa, joka tukee kaikkia yleisimpiä viivakoodiformaatteja ja soveltuu näin ollen hyvin käytettäväksi tässä tavaroidenhallintajärjestelmässä.

3.5.2 Käyttötapaukset



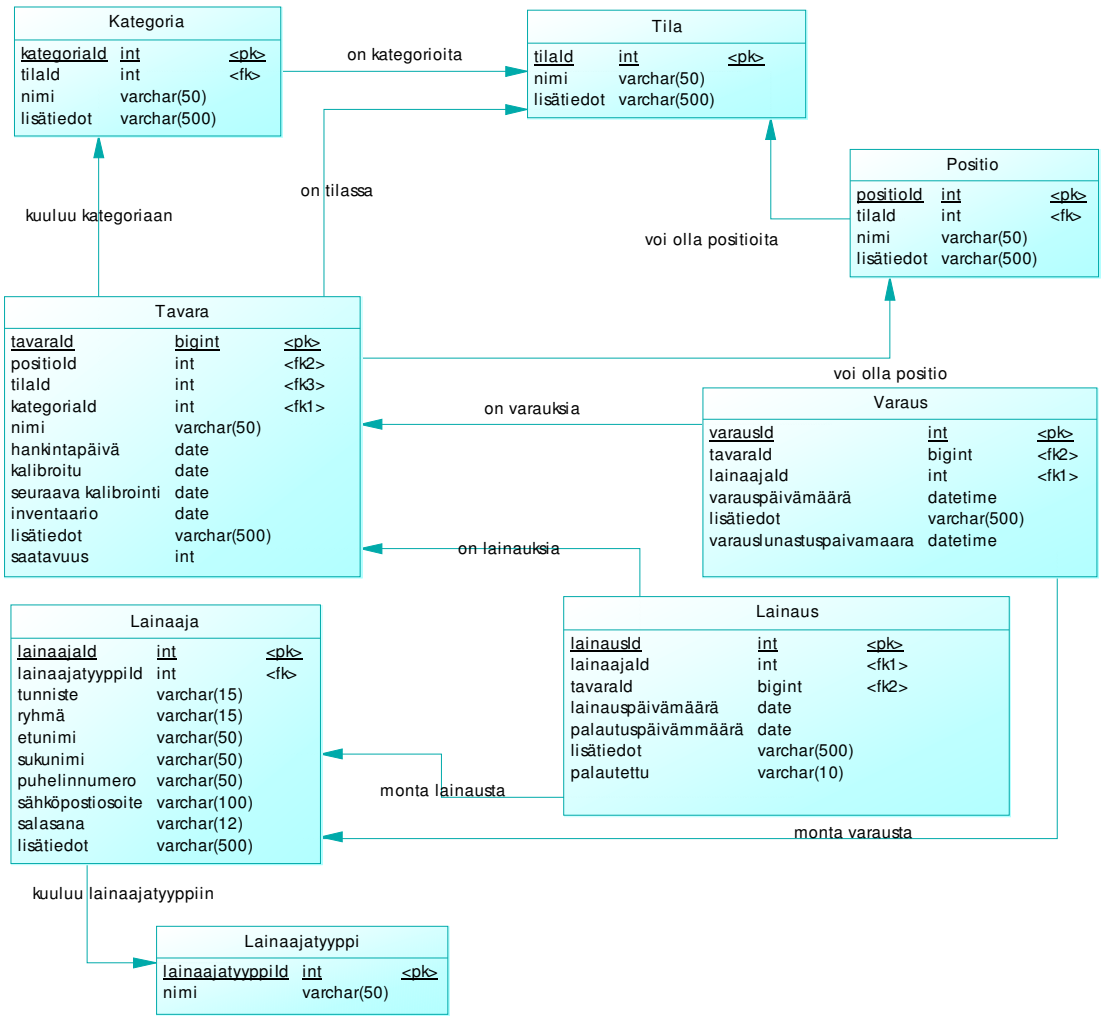
Kuvio 13. Käyttötapauskaavio

Kuvio 13 esittää Tavaroidenhallintajärjestelmän korkean tason käyttötapauskaavion. Oppilaan käyttötapauksissa on aina mukana joko Vastuuhenkilö tai Henkilökunnan jäsen. Vastuuhenkilö on siis ainoa, joka voi tehdä muutoksia tavaroihin, positioihin ja kategorioihin. Näiden hallinnoimisessa vastuuhenkilö pystyy lisäämään, poistamaan ja muokkaamaan niitä. Vastuuhenkilö on myös ainoa, jolla on oikeus lisätä ja muokata lainaajien tietoja sekä poistaa lainaajia järjestelmästä. Henkilökunnan jäsen voi vastuuhenkilön tapaan toimia oppilaan kanssa tavaran lainaamisessa, palauttamisessa, varaamisessa ja varauksen peruuttamisessa. Henkilökunnan jäsen pystyy myös rekisteröitymisen lisäksi selaamaan tavaralistaa. Kuten kaaviokin osoittaa, oppilas ei

pysty itse kirjautumaan ja käyttämään järjestelmää, vaan vaatii joko henkilökunnan jäsenen tai vastuuhenkilön avukseen.

3.5.3 Tietokanta

Tavaroidenhallintajärjestelmän tietokanta sisältää 8 tietokantataulua kuvion 14 osoittamalla tavalla.



Kuvio 14. Tavaroidenhallintajärjestelmän tietokannan fyysinen tietomalli

Tietokannan tulee olla mahdollisimman dynaaminen, jotta monimutkaistenkin tietokantakyselyiden tekeminen olisi mahdollisimman yksinkertaista ja loogista. Dynaamisuudella saadaan myös tarkennettua tavaroiden ja lainauksien kirjanpitoa. Jokaisella tavaralla on aina kategoria ja tila, joihin se kuuluu. Tavaralla voi olla varauksia ja lainauksia samoin kuin lainaajalla. Lainaaja kuuluu aina johonkin lainaajatyyppiin. Tässä tapauksessa toteuttamisen helpottamiseksi myös jokainen kategoria pakotettiin kuulumaan johonkin tilaan. Lisäksi tavaralla voi olla positio ja positio voi kuulua johonkin tilaan.

Taulujen pää- ja viiteavaimien ansiosta varmistetaan se, että tietokannasta ei voi poistaa esimerkiksi tilaa, mikäli siihen on liitetty kuuluvaksi kategorioita. Myöskään kategoriaa ei voida poistaa, jos siihen kuuluu tavaroita. Sama pätee myös lainaajatyyppiin, jota ei voi poistaa, mikäli siihen kuuluu jokin lainaaja. Taulujen pääavaimet ovat tietotyypiltään kokonaislukuja. Ne on asetettu auto increment -tyyppisiksi, mikä tarkoittaa automaattisesti kasvavaa numeroa, eli rivin yksilöivää avaimen arvoa. Tavara-taulussa pääavain ei ole auto increment -tyyppinen, ja viivakoodin pituuden takia kokonaisluku on paljon suurempi kuin muilla tauluilla.

3.5.4 Ohjelmamoduulit ja nimeämiskäytännöt

Tavaroidenhallintajärjestelmän käyttöliittymä ja toimintalogiikka jaetaan omiin loogisiin moduuleihinsa. Kukin moduuli vastaa tässä tapauksessa yhtä Windows Forms -lomaketiedostoa, ”ikkunaa”, eli käytännössä C#-kielellä toteutettua luokkaa, jonka tiedostotarkennin on ”.cs”. Jokaisella näillä on rinnallaan designer-tiedosto, jossa toteutetaan graafisen käyttöliittymän komponenttien lisääminen ja niiden tyyliominaisuudet sekä esitellään tapahtumankäsittelijät. Visual Studio luo designer-tiedostot automaattisesti sen mukaan, mihin käyttäjä GUI-komponentit tyylimäärityksineen sijoittaa. Käytännössä jokainen moduuli on yksi graafinen käyttöliittymäikkuna. Järjestelmän moduulit ovat seuraavat:

- KirjautuminenForm.cs
- RekisteröityminenForm.cs
- PääsivuForm.cs
- LisääTavaraForm.cs
- MuokkaaTavaraaForm.cs
- LisääLainaajaForm.cs
- MuokkaaOmiaTietojaForm.cs
- LisääKategoriaForm.cs
- MuokkaaKategoriaaForm.cs
- LisääPositioForm.cs
- MuokkaaPositiotaForm.cs
- VarmennusForm.cs.

KirjautuminenForm.cs sisältää käyttäjän käyttäjänimen ja salasanan tarkistamisen sekä sisäänkirjautumis-toiminnon. RekisteröityminenForm.cs sisältää uuden käyttäjän tietojen syötteiden tarkistamisen ja lisäämisen tietokantaan.

PääsivuForm.cs sisältää toiminnallisuudet tavaroiden lainaamiseen, varaamiseen ja hallitsemiseen. Lisäksi lainaajien, positioiden ja kategorioiden hallinta toteutetaan tässä tiedostossa. Muistutukset-toiminto toteutetaan tässä tiedostossa niin, että kun lainauksen eräpäivä on mennyt umpeen, asetetaan Muistutus-välilehti oletusnäkymäksi, kun käyttäjä kirjautuu sisään.

LisäätavaraForm.cs sisältää tavaran tavaranlisäystoiminnon, jossa ensin tarkistetaan syötteet ja sen jälkeen lisätään tavaran tiedot tietokantaan. MuokkaaTavaraaForm.cs on kuten LisääTavaraForm, mutta sen avautuessa tietokentissä näkyy tavaran senhetkiset tiedot.

LisääLainaajaForm.cs sisältää toteutuksen lainaajien lisäämiselle. Tässäkin ensin tarkistetaan syötteet, jonka jälkeen lainaajan tiedot lisätään tietokantaan. MuokkaaOmiaTietojaForm.cs sisältää kaksi eri toiminnallisuutta. Kun sisäänkirjautunut käyttäjä painaa pääsivun ”Muokkaa omia tietoja” -linkkiä, niin käyttäjälle näytetään hänen omat tietonsa. Kun taas käyttäjä valitsee lainaajalistalta jonkin lainaajan muokattavaksi, lomake näyttää valitun lainaajan tiedot ja antaa mahdollisuuden muokata niitä. Myös muokattujen tietojen tallentaminen toteutetaan tässä tiedostossa.

LisääKategoriaForm.cs sisältää toteutuksen kategorian lisäämiselle ja tässäkin, kuten LisääPositioForm.cs -tiedostossa, syötteet tarkistetaan ennen tietokantaan tallentamista. MuokkaaKategoriaaForm.cs ja MuokkaaPositiotaForm.cs ovat lähes identtisiä, joista toinen käsittelee kategorioita ja toinen positioita.

VarmennusForm.cs sisältää ainoastaan toiminnallisuuden, jossa käyttäjältä kysytään järjestelmänvalvojan salasana ja tarkistetaan oliko se oikein. Sen jälkeen palautetaan parametrina totuusarvo siitä, oliko salasana oikein vai ei. Tätä tarvitaan vain silloin, kun halutaan rekisteröidä uusi vastuuhenkilötyyppinen käyttäjä.

Nimeämiskäytännöt

Sovelluksen käyttöliittymään kuuluu paljon nimettäviä kohteita, kuten kooditiedostot, käyttöliittymäkomponentit ja metodien nimet. Kooditiedostot nimetään niin, että jokaisen ohjelmamoduulin nimen viimeinen sana on ”Form”. Nimen alkuosa muodostetaan tiedostojen toiminnallisuuksien mukaan, kuten esimerkiksi ”KirjautuminenForm.cs”. Tiedoston nimessä jokainen sana kirjoitetaan yhteen isolla alkukirjaimella. Kyseisellä käytännöllä helpotetaan tiedostojen erottamista toisistaan.

Kuten kooditiedostot, myös metodit nimetään niiden toiminnallisuuksien mukaan. Nimet alkavat aina pienellä alkukirjaimella. Seuraavat sanat kirjoitetaan yhteen isolla alkukirjaimella, esimerkiksi ”muodostaTavaraSelect”.

Käyttöliittymäkomponentit nimetään niin, että nimen alkuosasta on pääteltävissä, mihin ikkunaan tai mille välilehdelle komponentti kuuluu. Esimerkiksi ”lainausViivakoodiTxtBox” kuuluu Lainaa tavara -välilehdelle, johon syötetään tavaran viivakoodi. Nimet alkavat aina pienellä alkukirjaimella ja seuraavat sanat erotellaan toisistaan isoilla alkukirjaimilla. Nimen loppuosa erottelee komponenttityypit toisistaan. Esimerkiksi ”TxtBox” = TextBox, ”Lbl” = Label ja ”Btn” = Button.

4 TOTEUTUS

4.1 Projektin vaiheet ja läpivienti

Projekti vietiin läpi omana osanaan tätä opinnäytetyötä. Projektissa käytettiin ketteriin menetelmiin kuuluvaa protoilumallia. Protoilumalli sopi tähän projektiin hyvin, sillä yksinkertaisen prototyypin käyttöliittymästä ja toiminnallisuudesta sai aikaan suhteellisen pienellä työmäärällä. Protoilumalli lienee parhaita keinoja yhteisymmärryksen aikaansaamiseksi asiakkaan ja kehittäjän välillä esimerkiksi ulkoasun ja käyttöliittymän logiikan osalta. Kuten jokaisessa projektissa, aluksi laadittiin projektisuunnitelma, johon tehtiin suunnitelma ja aikataulutus tulevasta projektista. Suunnitelman ohessa laadittiin myös riskianalyysi, jossa analysoitiin projektissa mahdollisesti ilmeneviä viivästyksiä ja ongelmatilanteita sekä muita riskejä. Lisäksi tehtiin työmääräarvio projektiin käytettävistä työtunneista.

Esitutkimusvaihe

Projektin alussa tehtiin esitutkimusta, jossa kartoitettiin perusasioita, kuten järjestelmän käyttötarkoitus, ympäristö sekä perusvaatimukset. Näin saatiin kokonaisvaltainen kuva siitä, millaista järjestelmää aletaan kehittää. Esitutkimusvaihe kesti vain noin kaksi viikkoa, jonka aikana asiakkaan kanssa pidettiin palavereita. Samalla kirjattiin ylös asiakkaan järjestelmään kohdistuvat vaatimukset ja toiveet.

Määrittely- ja suunnitteluvaihe

Määrittelyvaihe kesti noin kuukauden. Kyseisessä vaiheessa alettiin jo tehdä ensimmäistä prototyyppiä niiden tietojen perusteella mitä oltiin esitutkimus- ja määrittelyvaiheessa ehditty keräämään. Määrittelyvaiheessa suunniteltiin jo alustavasti tietokannan rakennetta ja siitäkin tehtiin prototyyppi. Prototyypissä oli aluksi vain 4 taulua: lainaaja, tavara, lainaus ja varaus. Pian kävi kuitenkin selväksi, ettei lainaajan ja tavaran kaikkia tietoja ole järkevää sisällyttää samaan tauluun. Näinpä pian päätettiin tehdä tavaralle kategoria-, tila- ja positio-taulut. Lainaajalle puolestaan tehtiin lainaajatyyppi-taulu. Määrittelyvaiheessa laadittiin vaatimusmäärittelydokumentti, jossa määriteltiin järjestelmän vaatimukset, mutta ei vielä otettu kantaa teknisiin yksityiskohtiin. Suunnitteluvaiheessa sen sijaan laadittiin tekninen määrittelydokumentti. Siinä

määriteltiin muun muassa järjestelmän osat ja graafisen käyttöliittymän osat sekä navigointikartta. Tekninen määrittely alusti toteutusvaihetta, jossa sekä vaatimus- että teknisessä määrittelyssä laaditut dokumentit toimivat pohjina.

Toteutusvaihe

Kun toimiva ja mielestämme valmis prototyyppi oli saatu tehtyä, alettiin toteuttaa lopullista versiota. Toteutusvaihe joka kesti noin kaksi kuukautta, erosi prototyyppivaiheesta siten, että siinä kaikki yritettiin tehdä alusta asti mahdollisimman lopulliseen muotoon. Graafisen käyttöliittymän komponentit sijoiteltiin mahdollisimman loogisille paikoille. Värejä sekä kuvia lisättiin tavoitteena saada käyttöliittymästä helppokäyttöisempi ja silmää miellyttävämpi. Toteutusvaiheessakin tuli vielä esille asioita, joita piti miettiä yhdessä asiakkaan kanssa. Esimerkiksi tavaran hakusuodattimia tarkennettiin ja sovittiin varauksen voimassaoloaika.

Toteutusvaiheen kenties vaikein vaihe oli, kun koodattiin lainaus- ja varaustoimintoja. Etenkin varauslistatoiminnon ja varauksen eräpäivän määrittely osoittautui haasteelliseksi, ja niitä jouduttiin korjaamaan useaan otteeseen. Nämä operaatiot vaativat paljon tietokantahakuja ja tarkistuksia.

Testausvaihe

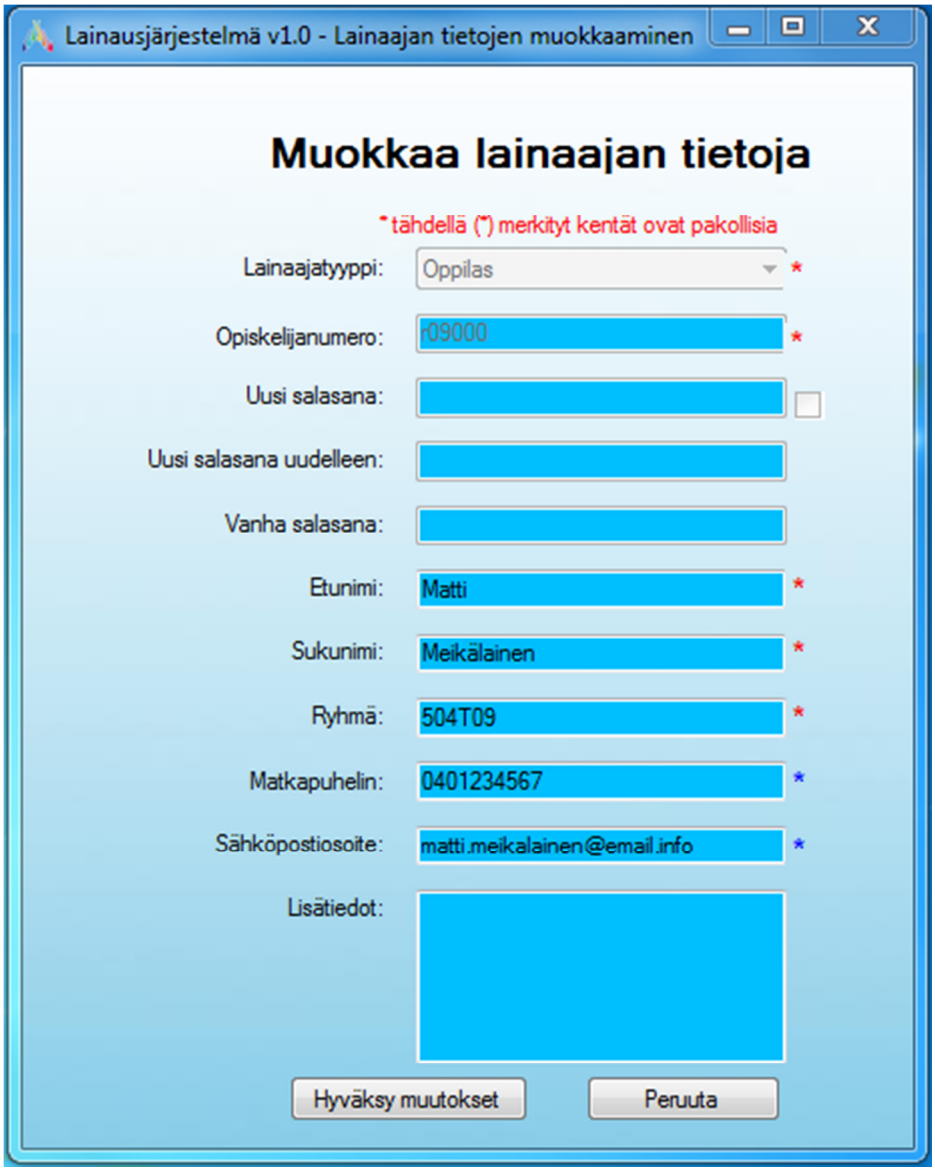
Kun järjestelmästä oli toteutettu toimiva versio, aloitettiin testausvaihe, joka kesti noin 2 viikkoa. Testauksessa järjestelmästä yritettiin löytää niin paljon virheitä ja epäkohtia, kuin mahdollista. Testausvaiheesta lisää luvussa 5.

4.2 Käyttöliittymän rakenne

4.2.1 Ulkoasu ja komponentit

Graafisen käyttöliittymän ulkoasu toteutettiin Windows Forms -tekniikan formeilla, eli ikkunoilla. Ikkunoihin lisättiin tarvittavia komponentteja, joista yleisimmät olivat TextBox; ”tekstikenttä”, Button, ”painike”, Label, ”otsikko”, ComboBox, ”valintalaatikko” ja DataGridView, ”tietotaulukko”. Tyylimäärityksiin käytettiin yksinkertaisia Windows Forms -tekniikan tarjoamia värejä ja muotoja. Pääikkunan vallitseva taustaväri on transparent, joka vastaa käytännössä vaaleanharmaata. Tällä saatiin ulkoasu mahdollisimman neutraaliksi, mutta kuitenkin sellaiseksi, että komponentit erottuvat hyvin toisistaan.

Painikkeissa on käytetty Internetistä haettuja yleiseen käyttöön tarkoitettuja kuvakkeita. Lisäikkunat poikkeavat hieman pääikkunan värimaailmasta.

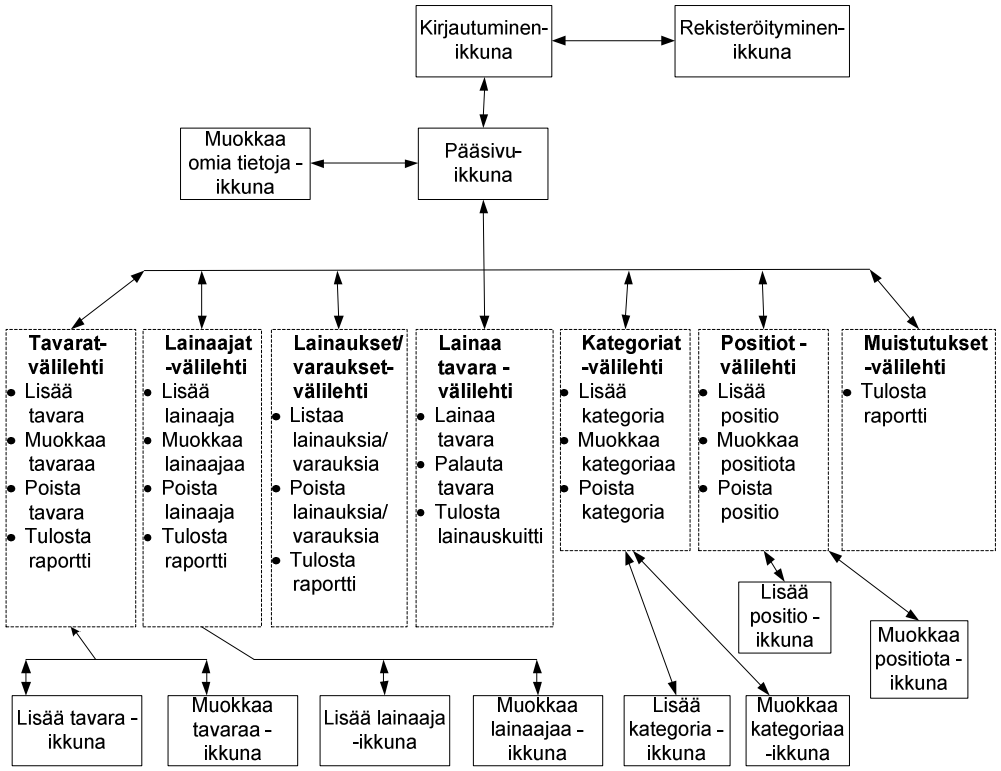


Kuvio 15. Käyttöliittymän Muokkaa lainaajan tietoja -ikkuna

Jokaisen ikkunan yläpalkissa on Rovaniemen ammattikorkeakoulun logo muokattuna icon- eli ikoni-tyyppiseksi. Kaikkien lisäikkunoiden, kuten kuviossa 15, Lainaajan tietojen muokkaaminen -ikkunan taustalla on käytetty itse tehtyä ”sinisestä valkoiseen” -väriliukua ulkoasun elävöittämiseksi.

4.2.2 Navigointi

Kuten kuvio 16 osoittaa, järjestelmän graafisessa käyttöliittymässä on 19 eri näkymää, joista välilehti-näkymät ovat osa Pääsivu-ikkunaa.



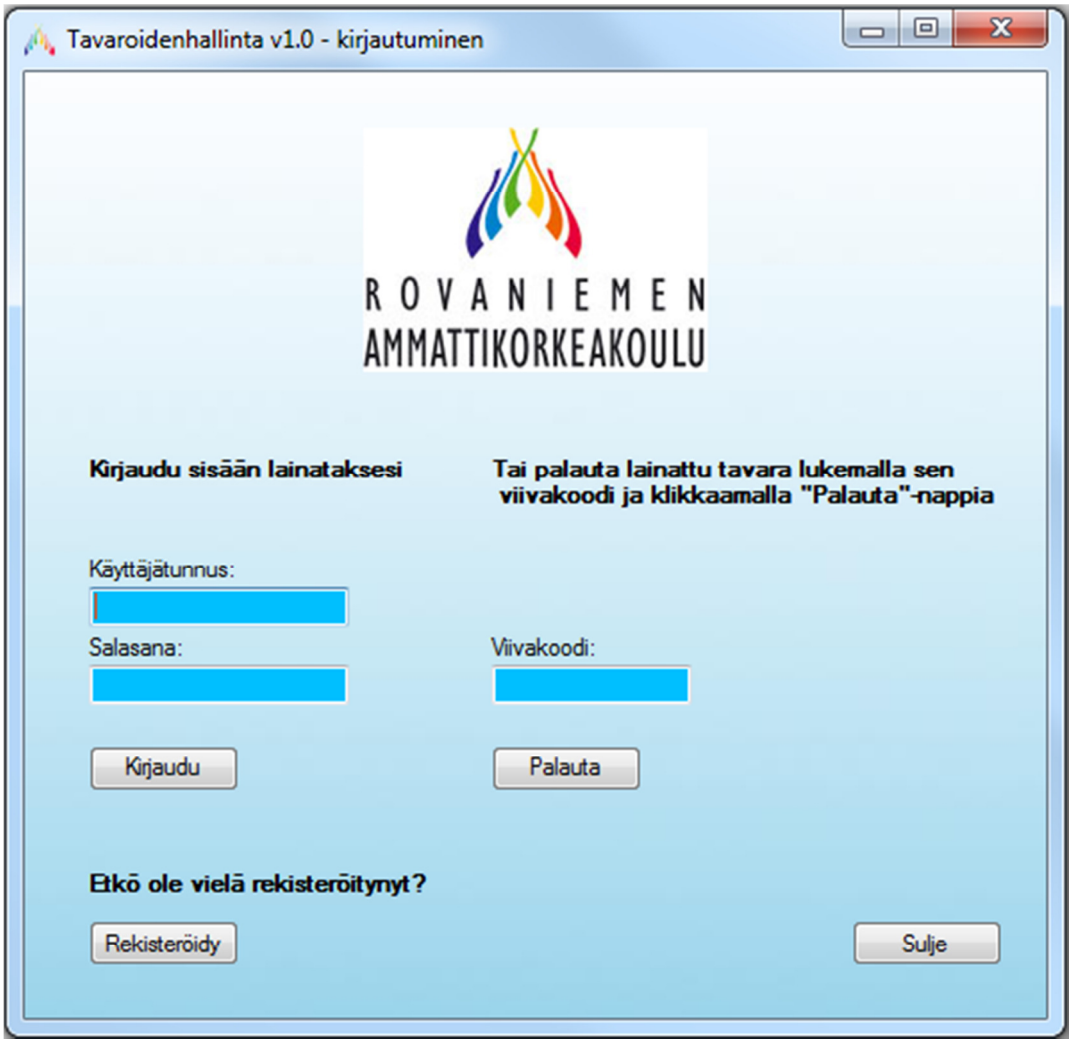
Kuvio 16. Graafisen käyttöliittymän navigointikartta

Muokkaa lainaajaa -ikkuna on toteutettu samassa ikkunassa Muokkaa omia tietoja -ikkunan kanssa. Kokonaisuudessaan vain vastuuhenkilö pystyy navigoimaan kaikkiin näkymiin. Henkilökunnan jäsen ei pääse käsiksi Kategoriat-, Positiot- eikä Lainaajat-välilehtiin. Muistutukset-välilehti on näkyvissä vain silloin, jos jonkin lainauksen eräpäivä on mennyt umpeen.

Käyttöliittymän navigointi on suurimmaksi osaksi toteutettu napeilla ja niiden taakse toteutetuilla tapahtumankuuntelijoilla. Kooditasolla kuuntelijat luovat tarvittaessa uuden ikkunan, eli Form-olion, eli instassin, ja asettavat sen näkyville. Samalla ikkuna, josta navigoidaan pois joko suljetaan, piilotetaan tai asetetaan disabled-tilaan, jolloin se on näkyvissä, mutta poissa käytöstä.

Pääikkuna sisältää paljon toiminnallisuuksia, ja niinpä se päätettiin toteuttaa käyttämällä välilehtiä. Windows Forms -tekniikassa välilehtiä ja niiltä siirtymisiä ei tarvitse erikseen koodata, vaan tavallinen hiirellä navigoiminen on valmiiksi toteutettu. Koodissa on kuitenkin mahdollista lisätä, poistaa ja muokata välilehtiä ja asettaa niitä näkyville, jne.

4.3 Kirjautuminen ja rekisteröityminen



Kuvio 17. Kirjautumissivu

Kuvion 17 esittämä kirjautumisikkuna on ensimmäinen ikkuna, joka avautuu, kun ohjelma käynnistetään exe-tiedostosta. Kirjautumisikkunassa voi joko kirjautua sisään järjestelmään, palauttaa tavaran tai rekisteröityä, mikäli kyseessä on uusi käyttäjä. Rekisteröidy-painikkeesta avautuu rekisteröitymisikkuna, jossa uuden käyttäjän tiedot syötetään. Tavaran voi palauttaa lukemalla viivakoodin omaan kenttäänsä viivakoodinlukijalla tai kirjoittaa sen käsin, mikäli viivakoodi on esimerkiksi vioittunut. Palauta-painiketta painamalla palautus rekisteröityy järjestelmään, tai jos viivakoodi on virheellinen, käyttäjälle annetaan virheilmoitus. Palautustoiminnon toiminnallisuus on koodattu PääsivuForm.cs-tiedostossa. Palautettaessa kirjautumissivun kautta koodissa täytyy luoda uusi instanssi pääsivusta, ja käyttää sen julkista metodia palau-

taTavara. Sisäänkirjautumis-toiminto on koodattu KirjautuminenForm.cs-tiedostoon.

```
string MySQL = "SELECT salasana, lainaajaid, etunimi, sukunimi from
LAINAAJA "
               + "WHERE tunniste = '" + tunnisteTxt-
Box.Text.ToString() + "'";

using (MySqlCommand cmdSelect = new MySqlCommand(MySQL, connection))
{
     mySqlLukija = cmdSelect.ExecuteReader();
     while (mySqlLukija.Read())
     {
          oikeaSalasana = mySqlLukija["salasana"].ToString();
          kayttajaId = mySqlLukija["lainaajaid"].ToString();

          if (tiivistettySalasana == oikeaSalasana)
          {
               kayttaja_ += mySqlLukija["etunimi"].ToString();
               kayttaja_ += " ";
               kayttaja_ += mySqlLukija["sukunimi"].ToString();
               tunnuksetOk = true;
          }
          else
          {
               tunnuksetOk = false;
          }
     }
     mySqlLukija.Close();
}
```

Esimerkkikoodi 5. Salasanan tarkistaminen

Esimerkkikoodi 5 on koodiesimerkki kirjautumissivun eli KirjautuminenForm.cs-tiedoston tarkistaTunnukset-metodista. Metodi on aluksi hakenut tietokannasta käyttäjänimen perusteella sille kuuluvan salasanan MD5-tiivisteen, ”tiivistettySalasana”. Seuraavaksi metodi on luonut käyttäjän syöttämästä salasanasta oman MD5-tiivisteen. Kuviossa ensimmäisellä rivillä on esitelty merkkijono MySQL, johon on tallennettu DML-komentotyypin select-lause, joka hakee lainaajan tiedot tietokannasta käyttäjän syöttämän tunnisteen perusteella. Seuraavaksi MySqlDataReader-tyyppiselle mySqlLukija-nimiselle instanssille annetaan kyselyn aloituskäsky mySqlLukija = cmdSelect.ExecuteReader();. While-silmukassa haun tulokset tallennetaan paikallisiin muuttujiin. Mikäli luotu tiiviste ”tiivistettySalasana”, täsmää haetun tiivisteen ”oikeaSalasana” kanssa, haetaan myös käyttäjän etu- ja sukunimi talteen. Samalla totuusarvotyyppinen jäsenmuuttuja ”tunnuksetOk” saa arvon true. Jos käyttäjän tunnistetta ei löydy tietokannasta, salasana-tiivistettä ei palaudu lainkaan. Tällöin tiivisteetkään eivät voi täsmätä, joten ”tunnuksetOk”

saa arvon false. Tällöin käyttäjä saa virheilmoituksen, joka kehottaa tarkistamaan käyttäjätunnuksen ja salasanan.

Esimerkkikoodi 6 sisältää esimerkkikoodin kirjautumisikkunan Kirjaudu-painikkeen tapahtumankuuntelija-metodista.

```
private void kirjauduBtn_Click(object sender, EventArgs e)
{
    tarkistaTunnukset();
    if (tunnuksetOk)
    {
       PääsivuForm ps = new PääsivuForm(this, vastuuHenkilo, kaytta-
       ja_, kayttajaId);
       ps.Show();
       this.Hide();
    }
}
```

Esimerkkikoodi 6. Uuden Form-instanssin luominen ja näkyville asettaminen

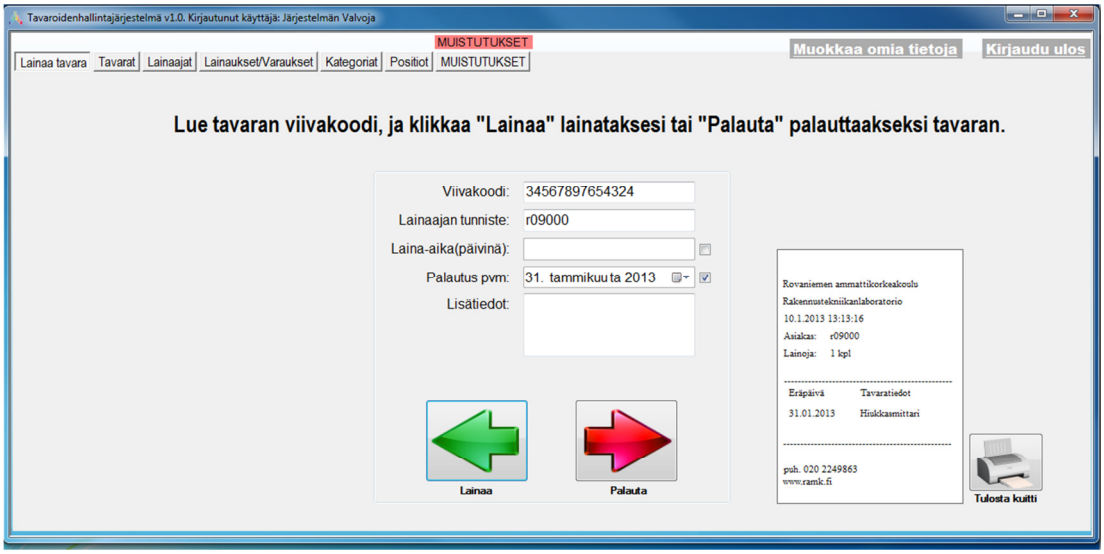
Mikäli tarkistaTunnukset()-metodikutsun jälkeen ”tunnuksetOk”-muuttujan arvo on true, tunnukset ovat oikein. Tällöin täyttyy myös if-lauseen ehto, ja näin siirrytään kohtaan, jossa luodaan PääsivuForm-tyyppinen olio ”ps”. Formin rakentajametodille välitetään parametreina nykyinen koko form-instanssi avainsanalla this. Koko form välitetään sen takia, koska järjestelmästä ulos kirjautuessaan käyttäjä palautetaan kirjautumissivulle. Tällöin siitä ei silloin enää tarvitse luoda uutta instanssia. Pääsivun rakentajametodille välitetään lisäksi totuusarvo-tyyppinen muuttuja ”vastuuHenkilo”, joka ilmaisee sen, onko kirjautuva käyttäjä vastuuhenkilö vai ei. Kayttaja_-muuttujassa välitetään merkkijonotyyppinen tieto, joka sisältää käyttäjän etu- ja sukunimen. KayttajaId on kokonaislukumuuttuja, int, jossa välittyy käyttäjän id-numero käyttäjän tunnistamiseksi mahdollisissa myöhemmissä tilanteissa. Tämän jälkeen uusi pääikkuna asetetaan näkyville ps-instanssin Show-metodilla. Samalla nykyinen kirjautumisikkuna piilotetaan komennolla this.Hide().

4.4 Pääsivu

4.4.1 Tavaran lainaaminen

Pääsivun avautuessa onnistuneen sisäänkirjautumisen jälkeen ensimmäisenä käyttäjälle tulee näkyviin Lainaa tavara -välilehti. Tällä on helpotettu tavaran lainaamisprosessia, koska suurimmassa osassa tapauksista käyttäjä kirjautuu järjestelmään lainatakseen tavaroita. Välilehti pyrittiin toteuttamaan

mahdollisimman yksinkertaiseksi ja informatiiviseksi. Lehdellä ei ole mitään muuta informaatiota kuin lainaamiseen tarvittavat käyttöliittymäkomponentit ja ohjeteksti.



Kuvio 18. Pääsivun Lainaa tavara -välilehti

Kuviossa 18 on kuvankaappaus Lainaa tavara -välilehdeltä. Tekstikenttien yläpuolella on Label-tyyppinen otsake, joka kertoo lyhyesti, miten tavaran lainaaminen ja palauttaminen onnistuu. Tavaran viivakoodi luetaan ”Viivakoodi”-tekstikenttään viivakoodinlukijalla. Tämän jälkeen syötetään lainaajan tunniste sekä valitaan valintaruuduista tapa, jolla halutaan ilmaista lainauksen voimassaoloaika. Seuraavaksi syötetään joko haluttu määrä päiviä, tai valitaan DateTimePicker-valitsimesta haluttu eräpäivä. Lainaukselle voi kirjoittaa lisätietoja Lisätiedot-kenttään, ja tämän jälkeen klikataan Lainaa-painiketta. Tämän jälkeen, kun syötteet on tarkistettu ja lainaus on rekisteröity, tulostuu lainauskuitti näkymän oikeassa reunassa olevan Panel-elementin sisälle. Kuitissa näkyy oleellisimmat tiedot lainauksesta, kuten eräpäivä ja lainatun tavaran nimi. Kuitti voidaan tulostaa painamalla Tulosta kuitti -painiketta. Jos kyseessä olisi tavaran palauttaminen, pelkkä viivakoodin lukeminen ja Palauta-painikkeen painaminen riittäisi tavaran palauttamiseksi.

Tilanteessa, jossa tavara olisi merkitty lainatuksi jo jollekin toiselle, järjestelmä ilmoittaisi siitä ja kysyisi, halutaanko tavara varata. Ilmoitus annettaisiin myös, jos lainaajalla olisi jo varaus tai jos sen olisi varannut joku toinen. Järjestelmässä on varauslista, johon varauksia voidaan tallentaa. Siitä hetkestä jolloin tavara palautetaan, asetetaan listalle ensimmäisenä lisätyn varauksen

voimassaoloajaksi seuraavat 7 päivää. Jos varaaja ei lainaa tavaraa 7 päivän kuluessa, varaus poistetaan.

Olettaen, että tavara on vapaa ja lainaaja on rekisteröity järjestelmään sekä lainausta koskevat syötteet on hyväksytty, luodaan kaksi tietokantaan tehtävää DML-komentoa. Toisessa lainaus-tauluun lisätään insert-lauseella uusi rivi, johon tallennetaan lainausta koskevat tiedot. Toisessa lauseessa tavara-tauluun tallennetaan tavaraa koskeva saatavuus-tieto, joka tässä tapauksessa asetetaan arvoon 1, joka osoittaa tavaran lainatuksi.

```
InsertCommand.CommandText =
"INSERT INTO LAINAUS (lainaajaid, tavaraid, lainauspaivamaara, "
+ "PALAUTUSPAIVAMAARA, lisatiedot, palautettu) VALUES (" +
LainaajaId + ", " + TavaraId + ", NOW(), " + erapaiva + ", '" +
lainausLisaTiedotTxtBox.Text.ToString() + "', 'ei')";
InsertCommand.ExecuteNonQuery();

UpdateCommand.CommandText = "UPDATE TAVARA set SAATAVUUS = 1 "
+ "WHERE tavaraid = " + TavaraId;
UpdateCommand.ExecuteNonQuery();
```

Esimerkkikoodi 7. Lainauksen tietojen tallentaminen tietokantaan

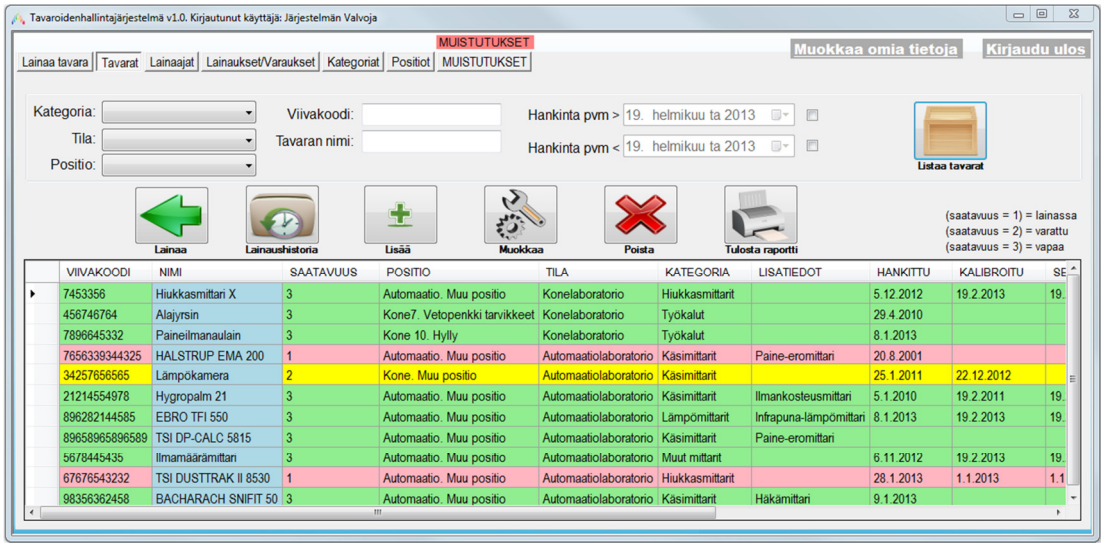
Esimerkkikoodi 7 kuvaa lainaustapahtuman koodia lainaaTavara-metodin sisällä. MySqlCommand-tyyppinen ”insertCommand”-olio saa tekstikseen Insert-lauseen, jossa lainaus-tauluun lisätään lainauksen tiedot. Tauluun ei lisätä koodissa lainkaan lainauksen id-numeroa, koska se on lainaustaulussa määritelty auto increment -tyyppiseksi, joka kasvaa automaattisesti, kun rivi lisätään. Tallennettavat arvot on kerätty käyttäjän syötteistä. Lisäksi tietoihin lisätään lainauspäivämäärä, joka saadaan suoraan sql-palvelimelta koodilla ”NOW()”. Saatavuus-kentän arvoksi tallennetaan ”ei” ilmaisemaan, että lainausta ei ole palautettu. Lainaus voitaisiin toisaalta myös poistaa, kun tavara palautetaan. Tässä tapauksessa on kuitenkin määritelty, että tavaralla ja lainaajalla pitää olla lainaushistoria, joten lainaus pitää jäädä tauluun palautuksen jälkeenkin. Palautettu-ominaisuus erottelee palautetut lainaukset palauttamattomista. Komento ajetaan tietokantaan komennolla InsertCommand.ExecuteNonQuery().

Update-lauseessa päivitetään tavara-taulun saatavuus-kenttä lainattavan tavaran osalta. Arvo 1 ilmaisee, että tavara on lainassa. Tämä tieto on oleellinen silloin, kun käyttäjä tarkastelee tavaran tietoja. Arvo toimii myös indi-

kaattorina operaatioille, joissa tietokantahauissa otetaan selvää tavaran senhetkisestä saatavuudesta.

4.4.2 Tavarat

Pääsivun Tavarat-välilehti on koko graafisen käyttöliittymän toiminnoiltaan monipuolisin näkymä. Välilehdellä voidaan lisätä, poistaa ja muokata tavaroita. Välilehti näkyy sekä vastuuhenkilö- että henkilökunnan jäsen -käyttäjille, mutta vain Vastuuhenkilö pystyy lisäämään uusia tavaroita sekä muokkaamaan ja poistamaan niitä.



Kuvio 19. Pääsivun Tavarat-välilehti

Kuvion 19 esittämästä Tavarat-välilehti -näkymästä päästään lisäämään, muokkaamaan ja poistamaan tavaroita. Lisäksi päästään tarkastelemaan lainaushistoriaa valitsemalla tavara listalta ja painamalla Lainaushistoria-painiketta. Tämän jälkeen näkymä siirtyisi Lainaukset/Varaukset-välilehdelle. Jos painettaisiin Lainaa-painiketta, näkymä siirtyisi Lainaa tavara -välilehdelle. Listatuista tavaroista voidaan myös tulostaa raportti, jossa näkyy tavaralista sellaisenaan kuin käyttöliittymässäkin. Kooditasolta tarkasteltuna monimutkaisin ominaisuus on tavaroiden hakutoiminto. Yllä on kolme pudotusvalikkoa, joista voi asettaa hakukriteereiksi joko kategorian, tilan, tai position, tai jokaisen näistä. Lisäksi tavaroita voidaan hakea viivakoodin, nimen ja hankintapäivämäärän perusteella. Tässä järjestelmässä tietokannan hakulauseen rakentamiseksi on käytetty if-lohkoja.

```
public void muodostaTavaraSelect()
{
   string hakuLause = "SELECT TAVARA.TAVARAID AS VIIVAKOODI, TAVA
   RA.NIMI AS NIMI, TAVARA.SAATAVUUS AS SAATAVUUS, POSITIO.NIMI AS
   POSITIO, TILA.NIMI AS TILA, KATEGORIA.NIMI AS KATEGORIA,
   TAVARA.LISATIEDOT AS LISATIEDOT, TAVARA.HANKINTAPAIVA AS HAN KIT-
   TU, TAVARA.KALIBROITU AS KALIBROITU, TAVARA.SEURAAVA_KALIBROINTI
   AS 'SEUR.KALIB.', TAVARA.INVENTAARIO AS INVENTAARIO,
   POSITIO.LISATIEDOT AS 'POS.LISATIED', KATEGORIA.LISATIEDOT AS
   'KAT.LISATIED', TILA.LISATIEDOT AS 'TILA.LISATIED.'
   FROM   TAVARA INNER JOIN " + "KATEGORIA ON TAVARA.KATEGORIAID  =
   KATEGORIA.KATEGORIAID INNER JOIN
   POSITIO ON TAVARA.POSITIOID = POSITIO.POSITIOID INNER JOIN
   TILA ON TAVARA.TILAID = TILA.TILAID";

   if (alkuHankintaPvmChb.Checked == true || loppuHankin-
               taPvmChb.Checked == true ||
               !string.IsNullOrEmpty(tViivakoodiTxtBox.Text) ||
              !string.IsNullOrEmpty(kategoriaCb.Text) ||
              !string.IsNullOrEmpty(tilaCb.Text) ||
              !string.IsNullOrEmpty(positioCb.Text) ||
              !string.IsNullOrEmpty(tViivakoodiTxtBox.Text) ||
              !string.IsNullOrEmpty(tTavaranNimiTxtBox.Text))
   {
     if (!string.IsNullOrEmpty(kategoriaCb.Text))//PELKKÄ KATEGORIA
     VALITTU
     {
        hakuLause += " AND        (KATEGORIA.NIMI = '" + katego-
        riaCb.SelectedItem.ToString() + "')";
     }
     if (!string.IsNullOrEmpty(tilaCb.Text))
     {
        hakuLause += " AND        (TILA.NIMI = '" + ti-
        laCb.SelectedItem.ToString() + "')";
     }
     if (!string.IsNullOrEmpty(positioCb.Text))
     {
        hakuLause += " AND        (POSITIO.NIMI = '" + posi-
        tioCb.SelectedItem.ToString() + "')";
     }
     if (!string.IsNullOrEmpty(tViivakoodiTxtBox.Text))
     {
        hakuLause += " AND        (TAVARA.tavaraid LIKE '%" +
        tViivakoodiTxtBox.Text.ToString() + "%')";
     }
     if (!string.IsNullOrEmpty(tTavaranNimiTxtBox.Text))
     {
        hakuLause += " AND        (TAVARA.NIMI LIKE '%" + tTavaran-
        NimiTxtBox.Text.ToString() + "%')";
     }
     if (alkuHankintaPvmChb.Checked)
     {
        hakuLause += " AND        (TAVARA.HANKINTAPAIVA >= '" + al-
        kuHankintaPvmDtp.Value.ToString("yyyy-MM-dd") + "')";
        if (!loppuHankintaPvmChb.Checked)
        {
           hakuLause += ";";
        }
     }
     if (loppuHankintaPvmChb.Checked)
     {
        hakuLause += " AND        (TAVARA.HANKINTAPAIVA <= '" + lop-
        puHankintaPvmDtp.Value.ToString("yyyy-MM-dd") + "');";
```

```
      }
    }
    haeTavarat(hakuLause);
    tavaraDg.ClearSelection();
}
```

Esimerkkikoodi 8. Select-lause tavaroidenhakutoiminnossa

Kun käyttäjä painaa Tavarat-välilehden Listaa tavarat -painiketta, painikkeen tapahtumankuuntelija-metodissa kutsutaan muodostaTavaraSelect-nimistä metodia, jonka sisällöstä Esimerkkikoodi 8 -koodiesimerkin koodi on kaapattu. Alussa luodaan merkkijono-tyyppinen muuttuja nimeltä ”hakuLause”, johon muodostettava tietokannan select-lause tallennetaan. Seuraavaksi muodostetaan kova-koodaamalla lauseen alku, jossa määritellään haettavat ominaisuudet kustakin taulusta. Haku kohdistuu tässä tapauksessa tauluihin Tavara, Tila, Positio ja Kategoria. ”Inner join”-komennolla yhdistetään taulujen yksilöivät avaimet, jotta vältetään kaksinkertaiset hakutulokset. Tämän jälkeen if-lohkoissa tarkistetaan, mitä hakukenttiä käyttäjä on valinnut. Tämän mukaan lisätään hakuLause-merkkijonoon ehtoja WHERE = -osan perään. LIKE-sanalla toteutetaan se, että haku etsii tietokannasta hakuosumia, vaikka hakukenttään olisi kirjoitettu pelkästään yksi kirjain. Tällöin kysely palauttaa kaikki hakuosumat kentistä, joissa annettu kirjain/kirjaimet esiintyy. Jos hakuehtoja ei ole valittu, muodostuu select-lause pelkästään esimerkkikoodin 8 alussa olevasta osasta. Silloin listataan lauseeseen kova-koodatut ominaisuudet kaikista tavaroista. Lopulta kun lause on valmis, se välitetään haeTavarat-metodille, joka laukaisee hakukomennon. Hakutulokset tallennetaan DataGridView-tauluun, jonka nimi on ”tavaraDg”.

```
private void haeTavarat(string hakuLause)
{
  string mySql = hakuLause;

  using (connection)
  {
    try
    {
      if (connection.State != System.Data.ConnectionState.Open)
      {
        connection.Open();
        }
        using (MySqlCommand cmdSel = new MySqlCommand(mySql, connec-
        tion))
        {
          DataTable dt = new DataTable();
          MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);
          dt.Clear();
          dt.Dispose();
```

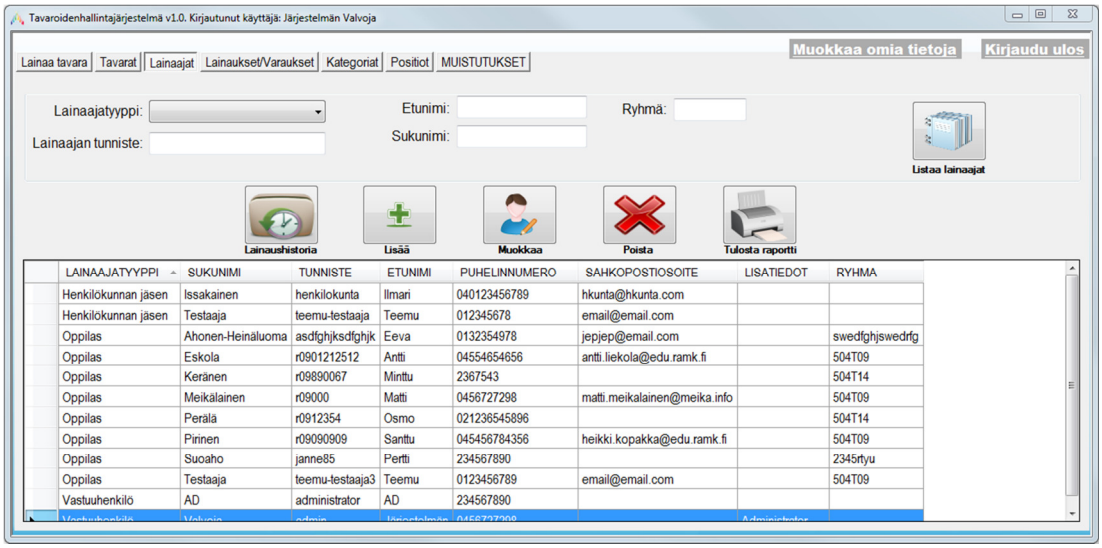
```
            da.Fill(dt);
            tavaraDg.DataSource = dt;
          }
          connection.Close();
          tavaraDg.ClearSelection();
          tavaraId = "";
          }
        catch (Exception e)
        {
          MessageBox.Show(e.ToString());
          connection.Close();
        }
        varitaTavaraDg();
    }
}
```

Esimerkkikoodi 9. Hakutuloksien tallentaminen DataGridView-tauluun

Esimerkkikoodi 9:ssa haeTavarat-metodi vastaanottaa parametrina esimerkkikoodi 8:ssa muodostetun hakulauseen ja ajaa sen tietokantaan sekä tulostaa hakutulokset listalle. Hakulause välitetään ”da”-nimiselle MySqlDataAdapter:lle, joka ajaa haun tietokantaan, ja täyttää ”dt”-nimisen DataTable-instanssin haetuilla hakutuloksilla. Seuraavaksi DataGridView-tyyppisen tietotaulun, ”tavaraDg”:n tietolähteeksi (DataSource) asetetaan hakutuloksilla täytetty tietotaulu, ”dt”. Tämän jälkeen hakutulokset näkyvät listalla, mutta ennen sitä kutsutaan funktiota varitaTavaraDg, jossa rivien taustavärit muutetaan tavaran saatavuustilanteen mukaan helpottamaan listan luettavuutta.

4.4.3 Lainaajat

Lainaajat-välilehti on ulkoasultaan samanlainen kuin Tavarat-välilehti. Tavaroiden sijaan tällä välilehdellä voidaan lisäämisen, poistamisen ja muokkaamisen lisäksi hakea ja listata järjestelmään rekisteröityjä lainaajia. Tilanteessa, jossa oppilas haluaa lainata tavaran, hänet täytyy rekisteröidä tämän välilehden kautta Lisää-painikkeella. Henkilökunnan jäsen -käyttäjä ei kuitenkaan pysty näkemään tätä välilehteä, joten ainoa, joka voi rekisteröidä oppilaita, on vastuuhenkilö. Hakutoiminnossa hakukriteereinä voi käyttää lainaajatyyppiä, lainaajan tunnistetta sekä etu- tai -sukunimeä ja/tai ryhmää.

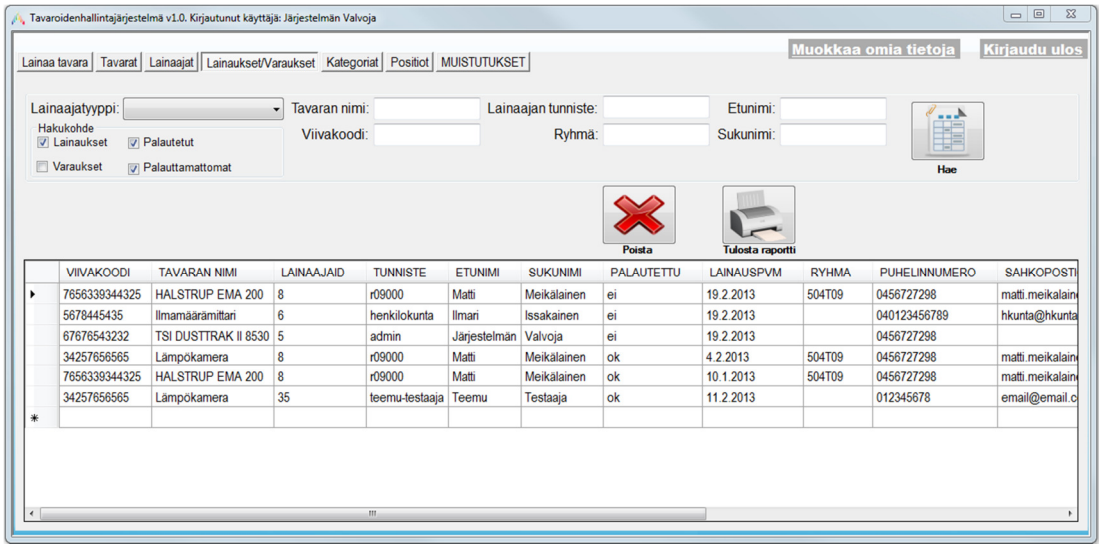


Kuvio 20. Pääsivun Lainaajat-välilehti

Kuvio 20 on kuvankaappaus Lainaajat-välilehdestä, joka on siis muutamaa poikkeusta lukuun ottamatta Tavarat-välilehden kaltainen. Tässä välilehdessä käyttäjä voi selata lainaajien tietoja. Välilehden ehkäpä tärkeimpänä ominaisuutena on uuden lainaajan lisääminen järjestelmään. Lainaajan lainaushistoria saadaan näkyviin valitsemalla lainaaja listalta ja painamalla Lainaushistoria-painiketta. Tällöin käyttöliittymä siirtyy automaattisesti Lainaukset/Varaukset-välilehdelle, ja listaa lainaajan lainaushistorian siellä olevaan taulukkoon. Tulosta raportti -painikkeella voi muiden välilehtien tapaan tulostaa listan sisällön. Tavarat-välilehdestä poiketen lainaajien hakutoiminnossa haetaan tietoa vain lainaaja- ja lainaajatyyppi-tauluista. Tämä yksinkertaistaa huomattavasti hakulauseen rakentamista kooditasolla. Hakutulokset voi myöhemmin järjestää listalla klikkaamalla sarakkeen otsikkoa. Tulokset järjestetään joko aakkosjärjestyksessä tai käännetyssä aakkosjärjestyksessä. Sama toiminto pätee myös muiden välilehtien taulukkoihin.

4.4.4 Lainaukset ja varaukset

Kuten Tavarat- ja Lainaajat-välilehdet, myös Lainaukset/varaukset-välilehti on hyvin samankaltainen. Tällä välilehdellä voidaan listata palautetut ja/tai palauttamattomat lainaukset sekä varaukset lainaajatyypin, tavaran nimen, viivakoodin, lainaajan tunnisteen, ryhmän ja/tai etu- ja sukunimen mukaan.



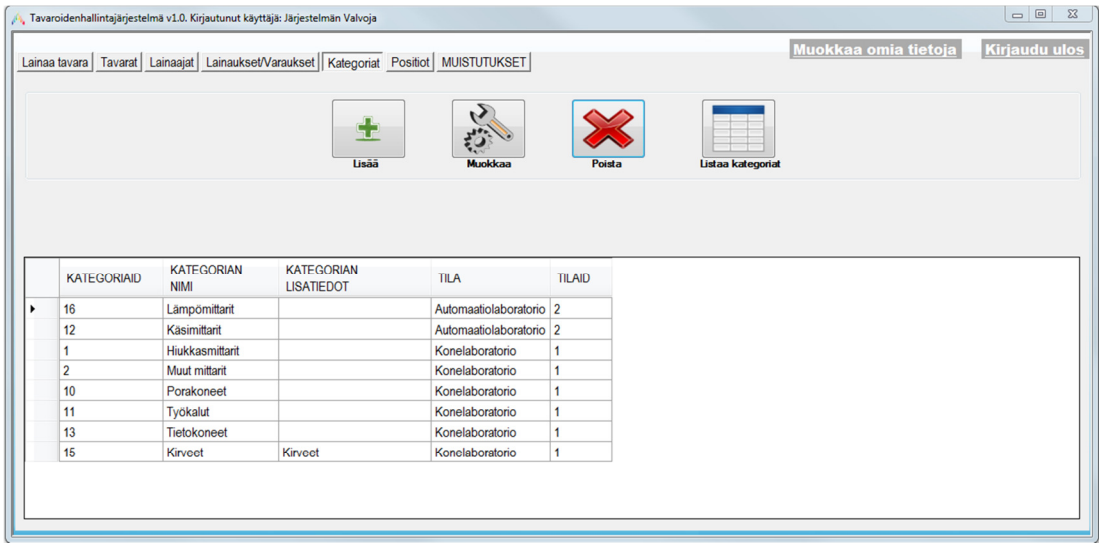
Kuvio 21. Pääsivun Lainaukset/Varaukset-välilehti

Koska lainaukset ja varaukset muodostuu aina lainaus- tai varaustapahtuman yhteydessä, niitä ei tarvitse käyttäjän toimesta erikseen luoda. Tällä välilehdellä, kuten kuvio 21 osoittaa, on ainoastaan poista-painike lainauksien ja varausten hallitsemiseksi. Lainausta ei normaalisti pitäisi joutua poistamaan, koska se poistuu automaattisesti, kun tavara palautetaan. Kuitenkin, jos jostain syystä on tarve poistaa lainaus muuten kuin palauta-toiminnon kautta, on sillekin toteutettu oma toiminnallisuus. Myös varauksen voi poistaa tarvittaessa.

Kooditasolla varauksen poistamistoiminnossa täytyy ottaa huomioon monta asiaa. Olettaen, että tavara on palautettuna varastossa, mutta sillä on varaus, täytyy ensin poistaa varaus varaus-taulusta. Samalla pitää päivittää myös tavara taulun saatavuus-ominaisuus sen mukaan, onko tavaralla muita varauksia. Jos poistettava varaus on varauslistan ensimmäisellä sijalla, tulee seuraavalle, eli ensimmäiselle sijalle siirtyvälle varaukselle asettaa eräpäivä. Tämä vaatii useita tarkistuksia tietokannasta. PääsivuForm.cs -tiedostoon on toteutettu tätä varten totuusarvon palauttavat onkoTavarallaVarauksia- sekä onkoTavaraLainassa -metodit. Varauslistaa ei ole toteutettu perinteisenä listana, vaan ”varauslistaan” ensimmäisenä varauksen tehnyt henkilö saadaan selville hakemalla varaus-taulusta tavaran viivakoodin ja varaajan id-numeron perusteella ”pienin” varauspäivämäärä aggregaatti-funktiolla MIN(). (Liite 1.)

4.4.5 Kategoriat ja positiot

Koska rakennustekniikan laboratoriossa on monentyyppisiä laitteita monessa eri paikassa, on tärkeää että myös kategorioita ja positioita pystytään lisäämään ja niiden ominaisuuksia muokkaamaan. Näin saadaan parannettua järjestelmän dynaamisuutta ja sovellettavuutta. Kategoriat-välilehti näkyy ainoastaan Vastuuhenkilölle, jolla on oikeus lisätä, poistaa ja muokata kategorioita.

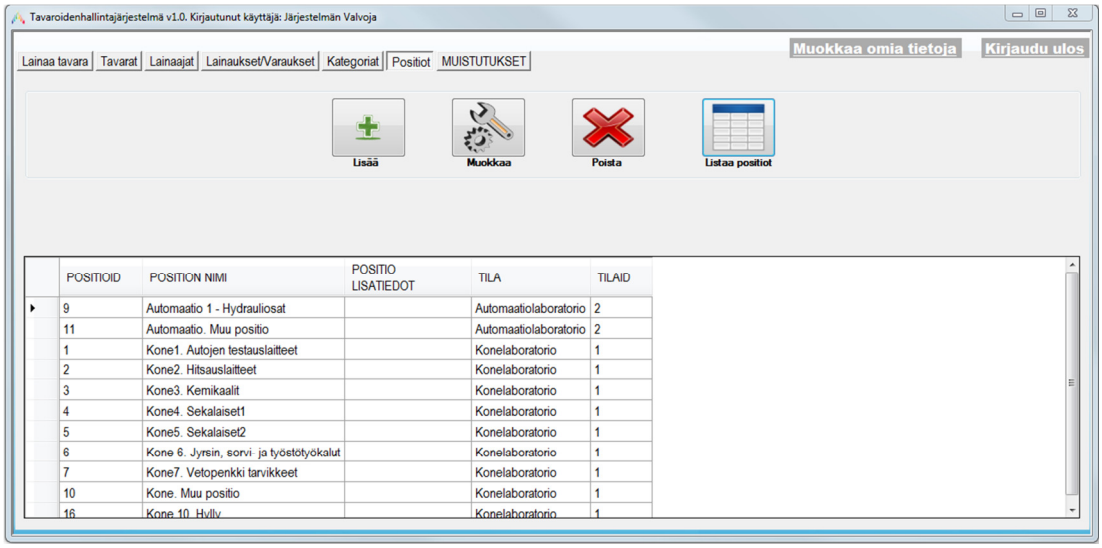


Kuvio 22. Pääsivun Kategoriat-välilehti

Kuten kuviosta 22 näkyy, myös Kategoriat-välilehti noudattaa samaa ulkoasua ja asettelua kuin muutkin Pääsivun välilehdet. Hakutoimintoa ei tarvita, koska kategorioita on melko vähän, ja niillä on vähän ominaisuuksia. Näinpä välilehdellä on ainoastaan Lisää-, Muokkaa-, Poista-, ja Listaa kategoriat -painikkeet. Listaan ei myöskään tule näkyviin muuta kuin kategoriataulusta haetut tiedot. Kategorioista ei ole tarpeen tulostaa raportteja, joten sillekään ei ole omaa painiketta.

Positiot-välilehti

Positiot-välilehti on lähes identtinen Kategoriat-välilehden kanssa. Positiot-välilehti Kategoria-välilehden tapaan näkyy ainoastaan vastuuhenkilölle, jolla on oikeus lisätä, poistaa, ja muokata positioita. Välilehdet päätettiin toteuttaa erillisiksi käytettävyyden parantamiseksi.



Kuvio 23. Pääsivun Positiot-välilehti

Kuviosta 23 nähdään, että Positiot-välilehti ei eroa juurikaan Kategoriat-välilehdestä muutoin kuin siten, että siinä näkyy ainoastaan positiot ja niiden omat tiedot. Edelleen Lisää-, Muokkaa-, Poista-, ja Listaa positiot -painikkeet löytyvät tutuilta paikoiltaan.

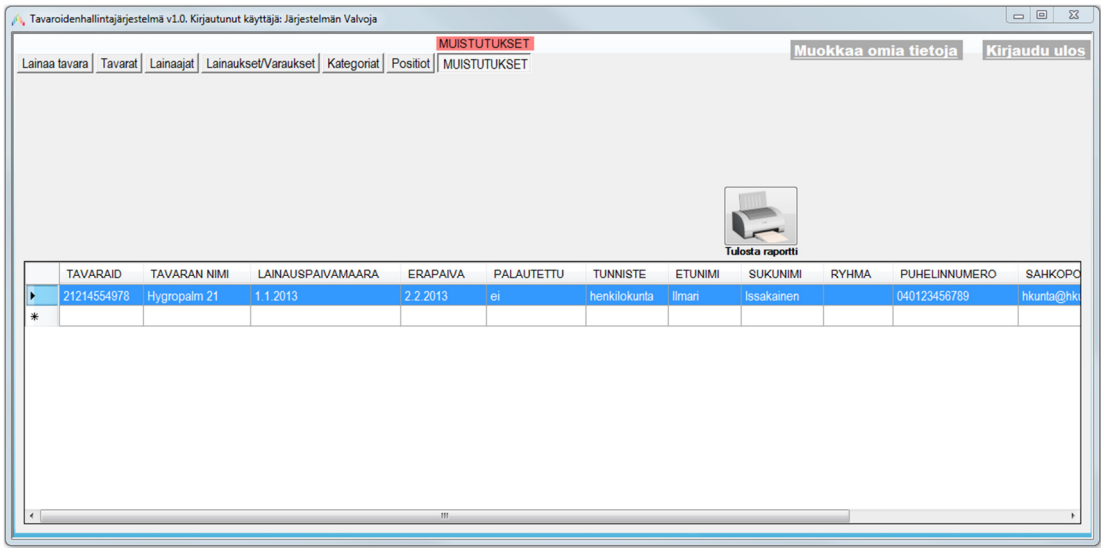
4.4.6 Muistutukset

Seitsemäs välilehti Pääsivulla on Muistutukset. Tällä välilehdellä listalle tulostaa automaattisesti niiden lainauksien tiedot, joiden eräpäivä on mennyt umpeen. Umpeutunut eräpäivä saadaan selville tietokantakyselyä käyttämällä.

```
string mySql = "select count(*) from LAINAUS where (PALAUTUSPAIVA-
MAARA < DATE_SUB(NOW(), INTERVAL 1 DAY)) AND (palautettu = 'ei')";
```

Esimerkkikoodi 10. Umpeutuneiden eräpäivien tarkistaminen

Esimerkkikoodissa 10 on select-lause, jolla päätellään, onko yhdenkään lainauksen eräpäivä mennyt umpeen. ”mySql”-nimiseen merkkijonoon tallennetaan hakulause, joka hakee lukumäärän lainaus-taulusta. Lauseen ehtoina on, että palautuspäivämäärä-attribuutin arvo on pienempi kuin nykypäivämäärästä vähennetty yksi päivä, ja palautettu-attribuutin arvo on ”ei”. Käytännössä tämä ehto tarkoittaa sitä, että valitaan ne lainaukset, joiden eräpäivä on ollut edellisenä päivänä. ”palautettu = ’ei”, määrittää sen, että tavaraa ei ole vielä palautettu. Muistutukset-välilehti tulee oletuksena näkyviin käyttäjän kirjautuessa järjestelmään silloin, kun myöhästyneitä lainauksia on löytynyt.



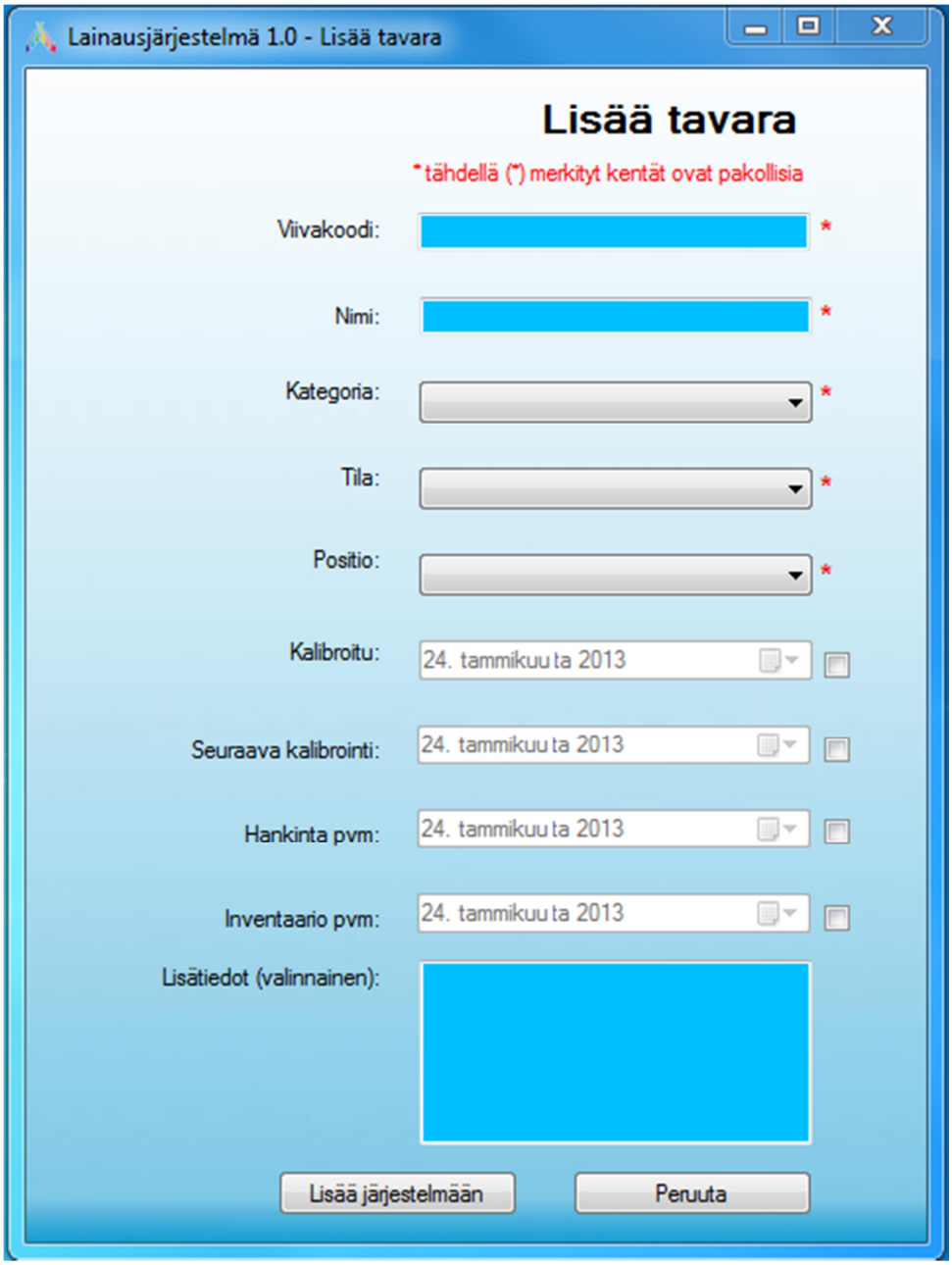
Kuvio 24. Pääsivun Muistutukset-välilehti

Sen lisäksi, että välilehti on oletuksena näkyvillä käyttäjän kirjautuessa sisään, ilmestyy myös punaisella taustalla oleva teksti ”MUISTUTUKSET” välilehden nimen yläpuolelle kuvion 24 esittämällä tavalla. Tämän tarkoitus on ainoastaan toimia muistutuksena silloin, kun käyttäjä navigoi pois välilehdeltä. Muutoin välilehti on ulkoasultaan edellisten välilehtien kaltainen. Muistutuksien ja niiden tietojen tulostaminen Tulosta raportti -painikkeella on ainoa välilehdellä tehtävissä oleva operaatio.

4.5 Lisää- ja Muokkaa-apuikkunat

Lisää-ikkuna

Lisää-ikkuna avautuu aina, kun Tavarat-, Lainaajat-, Kategoriat-, tai Positiot -välilehdiltä painetaan Lisää-painiketta. Uuden tavaran tiedot tulee syöttää ikkunassa oleviin kenttiin. Punaisella tähdellä merkityt kentät ovat pakollisia.



Kuvio 25. Lisää tavara -ikkuna

Kuvio 25 esittää Lisää tavara -ikkunaa, joka avautuu tässä tapauksessa, kun Tavarat-välilehdellä painetaan Lisää-painiketta. Kategoria, tila ja positio valitaan combobox-valikosta, joiden tiedot on haettu tietokannasta. Valintaruudut ruksaamalla saadaan aktivoitua niiden kohdalla oleva DateTimePicker-valikko eli päivämäärävalitsin. Jos valitsimia ei ole aktivoitu, tietokantaan ei tallenneta sen ominaisuuden arvoksi mitään. Kun käyttäjä on syöttänyt uuden tavaran tiedot, hän painaa Lisää järjestelmään -painiketta. Jos kaikki tarvittavat syötteet on annettu oikeassa muodossa, näytetään ilmoitus onnistuneesta lisäämisestä, ja ikkuna sulkeutuu.

Muokkaa-ikkuna

Muokkausikkuna puolestaan avautuu, kun Tavarat-, Lainaajat-, Kategoriat-, tai Positiot -välilehdellä painetaan Muokkaa-painiketta. Ikkunan avautuessa kenttiin haetaan tietokannasta muokattavan kohteen senhetkiset tiedot.

Kuvio 26. Muokkaa tavaraa -ikkuna

Kuvion 26 tapauksessa Muokkaa tavaraa -ikkunassa kentät on täytetty tietokannasta haetuilla tiedoilla. Jos tietokannasta löytyy tieto esimerkiksi kalibrointipäivämäärästä, asettaa sovellus automaattisesti päivämäärävalitsimeen oikean arvon, ja sen kohdalla oleva valintaruutu ruksataan. Jos tietoa ei löydy, valitsinta ei aktivoida eikä valintaruutua ruksata. Kun käyttäjä on muo-

kannut tiedot, hän painaa Hyväksy muokkaus -painiketta. Jos kaikki tarvittavat syötteet on annettu oikeassa muodossa, näytetään ilmoitus onnistuneesta muokkaamisesta, ja ikkuna sulkeutuu.

4.6 Tietokanta ja tietokantataulut

Tavaroidenhallintajärjestelmän tietokanta toteutettiin MySql Server 5.5 -palvelinohjelmistoa käyttävälle palvelimelle. Tietokannan suunnittelussa käytettiin Sybase Power Designer -ohjelmaa, jolla tehtiin aluksi CDM-malli, ja sen jälkeen PDM-malli. PDM-mallin jälkeen ohjelmalla generoitiin skripti tietokannan luomista varten. Skripti ajettiin palvelimelle MySql-palvelimen command line -ikkunan kautta. Jotta Windows Forms -sovelluksesta saatiin yhteys MySql-tietokantaan, täytyi sovellusta varten hakea Internetistä ns. MySql Connector, joka on C#-kielelle valmiiksi toteutettu MySql-rajapinta. Käytännössä tämä ei vaatinut kuin yhden dll-tiedoston lisäämisen sovelluksen resurssitiedostoihin.

Taulukko 1. Lainaaja-taulu

Attribuutti	**Tyyppi**	**Selite**
LAINAAJAAJAID	int not null, auto increment	lainaajan id-numero pääavain
LAINAAJAATYYPPIID	int not null	viiteavain lainaajatyyppi-tauluun
TUNNISTE	varchar(15) not null	lainaajan tunniste
RYHMA	varchar(15)	oppilaan ryhmä
ETUNIMI	varchar(30) not null	lainaajan etunimi
SUKUNIMI	varchar(30) not null	lainaajan sukunimi
PUHELINNUMERO	varchar(50)	lainaajan puhelinnumero
SAHKOPOSTIOSOITE	varchar(50)	lainaajan sähköpostiosoite
SALASANA	varchar(50) not null	salasanan MD5-tiiviste
LISATIEDOT	varchar(500)	lainaajan lisätiedot

Lainaajataulu koostuu taulukko 1:n osoittamalla tavalla kymmenestä attribuutista. LAINAAJAID on taulun pääavain ja yksilöivä kenttä. Kentän ominaisuudeksi on määritelty auto increment eli automaattisesti kasvava numero. LAINAAJATYYPPIID-attribuutti on viiteavain lainaajatyyppi-tauluun. Kuten kaksi edellistä, myös TUNNISTE-attribuutti on määritelty not null:ksi, eli tauluun ei voi lisätä riviä, jos näillä attribuuteilla ei ole arvoa. Myöskään attribuutit ETUNIMI, SUKUNIMI ja SALASANA eivät voi olla tyhjiä. Taulukon 1 Tyyppi-

sarakkeessa on ilmoitettu attribuutin tietotyyppi. Varchar, eli merkkijonon tapauksessa sulkeissa on myös pituus. Selite-sarakkeessa selitetään attribuutin tarkoitus, ja pääavaimen tapauksessa myös pääavain. Myös muiden tietokantataulujen tiedot on jaoteltu omiin taulukoihinsa. (Liite 2.)

Tietokantataulujen luomiseen Power Designer -ohjelmalla luotu skripti sisältää käytännössä vain peräkkäisiä SQL-lauseita.

```
create table LAINAAJA
(
   LAINAAJAID                     int not null AUTO_INCREMENT,
   LAINAAJATYYPPIID               int not null,
   TUNNISTE                       varchar(15) not null,
   …
   primary key (LAINAAJAID)
)
```

Esimerkkikoodi 11. Create table -lause

Esimerkkikoodi 11:ssä luodaan LAINAAJA-taulu create-lauseella. Aaltosulkeiden sisällä määritellään aluksi attribuutin nimi, ja sen perässä attribuutin tyyppi sekä not null, mikäli sen ei tule saada tyhjää arvoa. Pääavaimen perään laitetaan tarvittaessa automaattisesti kasvavan luvun tunnus auto increment. Lopuksi määritellään pääavain, eli primary key, joka tässä tapauksessa on LAINAAJAID.

4.7 Syötteiden tarkistaminen ja poikkeustenkäsittely

Järjestelmää käytettäessä poikkeustilanteita voi tulla useammastakin syystä. Tätä varten tarvitaan poikkeustenkäsittelyrutiini. Sen lisäksi, että virheet käsitellään niin, ettei ohjelma kaadu, tulee myös käyttäjää informoida lyhyesti ja informatiivisesti. Poikkeustenkäsittelyn yksi oleellisimmista tarkoituksista tässä järjestelmässä on tarkastaa käyttäjän antamat syötteet. Näitä ovat mm. kirjautumisessa tarkastettavat tunnukset ja tavaran tai vaikkapa lainaajan lisäämisessä annettavat tiedot.

```
private void lisaaBtnClick(object sender, EventArgs e)
{
  if (!tarkistaViivakoodi(viivakoodiTxtBox.Text.ToString()))
  {
    MessageBox.Show("Viivakoodissa saa olla ainoastaan numeroita,"
                    + " ja sen pituus pitää olla 2-19.",
                    "Huomautus");
  }
  else if (!tarkistaNimi(tavaraNimiTxtBox.Text.ToString()))
  {
    MessageBox.Show("Tavaran nimi täytyy olla 2-50 merkkiä pitkä, "
           + " ja se saa sisältää numeroita 0-9 ja ,._- -merkkejä.",
           "Huomautus");
  }
  else if (string.IsNullOrEmpty(kategoriaCb.Text))
  {
    MessageBox.Show("Valitse kategoria!", "Huomautus");
  }
  else if (string.IsNullOrEmpty(tilaCb.Text))
  {
    MessageBox.Show("Valitse tila!", "Huomautus");
  }
  else if (string.IsNullOrEmpty(positioCb.Text))
  {
    MessageBox.Show("Valitse positio!", "Huomautus");
  }
  else
  {
    lisaaTavara();
  }
}
```

Esimerkkikoodi 12. Syötteiden tarkistaminen

Esimerkkikoodissa 12 on esimerkkikoodi syötteiden käsittelystä tavaran lisäämisen tapauksessa. Koodi on Lisää tavara -ikkunan Lisää-painikkeen tapahtumankäsittelijä-metodi. Alussa tarkistetaan viivakoodin oikeellisuus käyttämällä totuusarvo-tyyppistä tarkistaViivakoodi-metodia, jolle välitetään parametrina viivakoodikentän teksti merkkijono-tietotyypiksi muutettuna. Metodi palauttaa joko arvon true tai false sen mukaan, onko viivakoodi läpäissyt tarkistuksen. Tarkistuksessa käytetään Regular expressions -tekniikkaa, josta lisää kohdassa 2.1.1. Jos viivakoodi ei läpäise tarkistusta, annetaan käyttäjälle virheviesti pienessä MessageBox-ikkunassa. Jos taas tarkistus läpäistään, siirrytään else if -haaraan, jossa tarkistetaan samalla tavalla tavaran nimen oikeellisuus käyttämällä samanlaista metodia nimeltä tarkistaNimi. Sen mukaan, läpäiseekö käyttäjän syöttämä nimi tarkistuksen, annetaan joko virheilmoitus tai siirrytään seuraavaan else if -haaraan. Seuraavassa haarassa tarkistetaan onko kategoria-combobox -valikosta valittu teksti tyhjä merkkijono tai null. Tämä tarkistus tehdään myös tila- ja positio -comboboxeille. Jos mikään if- tai else if -haarojen ehdoista ei toteudu, siirrytään else-haaraan,

jossa kutsutaan void-tyyppistä metodia lisaaTavara. LisaaTavara-metodi sisältää toiminnallisuuden tavaran tietojen keräämisen ja muodostamisen insert-lauseeksi, joka ajetaan tietokantaan.

Myös try-catch -lohkoja käytetään hyväksi poikkeustenkäsittelyssä. Erityisesti tietokantoihin tehtävät kyselyt tehdään aina try-lohkossa, jonka sisällä olevat mahdollisesti tapahtuvat poikkeukset/virheet catch-lohko käsittelee. Esimerkiksi, jos tietokantaan ei saada yhteyttä, tapahtuu virhe, joka käsitellään catch-lohkon sisältämällä logiikalla.

```
string mySql =
           "select count(*) from LAINAUS where (tavaraid = '"
            + tavaraId + "') AND (palautettu = 'ei')";
try
{
  if (connection.State != System.Data.ConnectionState.Open)
  {
    connection.Open();
  }
  MySqlCommand selectCountCmd = new MySqlCommand(mySql, connection);
  tunnusCount =int.Parse(selectCountCmd.ExecuteScalar().ToString());
  connection.Close();
}
catch (Exception e)
{
  MessageBox.Show("Ongelmia tarkistettaessa onko lainaajalla laina-
uksia " + e.ToString(), "Virhe");
  connection.Close();
}
```

Esimerkkikoodi 13. Try-catch -lohko poikkeustenkäsittelyssä

Esimerkkikoodi 13 esittää koodia onkoTavaraLainassa-funktion sisällä. Tässä try-lohkon sisään on laitettu kaikki sellaiset toiminnot, joissa todennäköisesti voi tapahtua virhe. Alussa muodostetaan tietokantayhteys, ja sitten luodaan MySqlCommand-tyyppinen olio, jolle annetaan parametreina select-lause ja tietokantayhteys. Sen jälkeen tunnusCount-muuttujaan asetetaan tietokantahaun tulos, ja tuloksen kokonaislukutyyppi varmistetaan käyttämällä int-tyypin Parse-metodia. Parse-metodi muuttaa string-tyyppisen hakutuloksen kokonaisluku- eli int-tyyppiseksi. Lopuksi tietokantayhteys suljetaan. Jos näissä operaatioissa tapahtuu virhe, ajo siirtyy catch-lohkoon. Lohkossa asetetaan käyttäjälle näkyviin ikkuna, joka ilmoittaa käyttäjälle virheestä, sekä näyttää tarkemman kuvauksen muuttamalla Exception-tyyppisen e-nimisen instanssin merkkijonoksi ja näyttää sen käyttäjälle.

5 TESTAUS

5.1 Testauksen tarkoitus ja kohteet

Toteutusvaiheen lopussa järjestelmä testattiin. Testauksen tarkoituksena oli löytää järjestelmästä toteutusvaiheessa syntyneet mahdolliset virheet ja epäkohdat. Testaamisesta tehtiin aluksi testaussuunnitelma. Suunnitelmassa määriteltiin, millä alkuvaatimuksilla, kriteereillä ja syötteillä testataan sekä mitkä ovat läpäisyvaatimukset. Läpäisyvaatimuksena oli, että testitapauksen tuloksen pitää olla se, mikä on määritelty odotetuksi tulokseksi. Testausympäristönä käytettiin tavallista Windows 7 -työasemaa. Tietokantapalvelimena testauksessa toimi Internetistä vuokrattu virtuaalipalvelin, jossa on asennettuna MySql-palvelinohjelmisto. Testaajana toimi tämän opinnäytetyön tekijä.

Testattavat kohteet

Testauksen kohteena olivat järjestelmälle vaatimuksiksi määritetyt toiminnallisuudet. Sisäänkirjautumisessa testattiin kirjautuminen väärällä tunnuksella ja salasanalla sekä tyhjillä syötteillä. Rekisteröitymisessä sekä tavaran, lainaajan, kategorian ja position lisäämistoiminnoissa testattiin vääränlaisten syötteiden antaminen ja niistä ilmoittaminen käyttäjälle. Tietojen muokkaustapauksessa testattiin syötteiden lisäksi myös duplikaattien tarkistaminen tietokannasta. Kahdella lainaajalla ei saanut olla samaa tunnistetta eikä kahdella kategorialla tai positiolla samaa nimeä. Poistamistoiminnoissa testattiin, pystyykö esimerkiksi tavaran poistamaan, mikäli se on lainattu tai varattu. Positioiden ja kategorioiden tapauksessa testattiin, pystyykö niitä poistamaan, mikäli niihin kuuluu tavaroita. Uloskirjautumisen testaamisessa varmistettiin, että järjestelmä toimii oikealla tavalla.

Testauksessa testattiin myös laadullisten tavoitteiden toteutumista. Tavaran lainaamisessa sai kulua enintään 30 sekuntia. Jos lainaajaa ei ollut vielä rekisteröity järjestelmään, lainaamiseen sai kulua enintään 120 sekuntia. Tämä aika sisältää lainaajan rekisteröimisen ja tavaran lainaamisen. Jos ainoastaan tavaraa ei ollut rekisteröity järjestelmään, aikaa sai kulua enintään 90 sekuntia. 90 sekuntia sisältää tavaran rekisteröimisen ja sen lainaamisen. Jos taas lainaajaa eikä tavaraa ollut rekisteröity järjestelmään, aikaa lainaamiseen sai kulua enintään 180 sekuntia. Tässä tapauksessa sekä lainaaja

että tavara piti rekisteröidä järjestelmään ja sen jälkeen lainata tavara. Tavaran palauttaminen sai kestää enintään 30 sekuntia ja sisäänkirjautuminen piti hoitua 10 sekunnissa. Laadullisten vaatimusten testitapauksissa käytettiin testaajaan toimesta mahdollisimman vähän aikaa syöttäen tiedot järjestelmään mahdollisimman nopeasti. Testitapauksia oli yhteensä 82.

5.2 Testauksen tulokset

Testauksen yhteydessä sisäänkirjautumisessa huomattiin, että käyttäjän kirjauduttua ulos uudelleen kirjautumiseen riitti ainoastaan salasanan oikeinkirjoittaminen. Tunniste-kenttä sai olla tyhjä. Virhe johtui siitä, että käyttäjän tunniste-muuttujaa ei tyhjennetty kirjautumisen jälkeen. Tämä oli helposti korjattavissa ja testitapaukset menivät läpi korjauksen jälkeen. Rekisteröitymisessä huomattiin, että virheellinen puhelinnumero ja sähköpostiosoite pystyttiin tallentamaan järjestelmään. Jos toinen oli väärässä muodossa, järjestelmä antoi virheilmoituksen. Kyseinen virhe johtui puutteellisesti toteutetusta syötteiden käsittelystä ja se korjattiin. Lisäys-, muokkaus- ja poistotoiminnoista ei löytynyt virheitä tai epäkohtia.

Laadullisten ominaisuuksien osalta tulokset osoittivat, että asetetut aikarajoitukset alittuivat pääsääntöisesti selkeästi. Tavaran lainaamisen kului 19 sekuntia, joka on 11 sekuntia vähemmän kuin määritelty maksimiaika. Tapaus, jossa lainaajaa ei ollut rekisteröity järjestelmään, aikaa kului vain 45 sekuntia, joka on 75 sekuntia vähemmän kuin suurin sallittu aika. Jos pelkästään tavaraa ei ollut rekisteröity järjestelmään, lainaamiseen kului 34 sekuntia, joka on 56 sekuntia alle enimmäisajan. Tapauksessa, jossa sekä lainaaja että tavara piti rekisteröidä järjestelmään tavaran lainaamiseksi, aikaa kului 50 sekuntia, eli 130 sekuntia alle enimmäisajan. Tavaran palautustoiminnossa kului testin mukaan aikaa 8 sekuntia, joka on 22 sekuntia alle sallitun maksimin. Sisäänkirjautuminen hoitui 6 sekunnissa jääden 4 sekuntia enimmäisajan alle.

Testauksen johtopäätökset

Testauksen tuloksista voidaan päätellä, että järjestelmän toteuttaminen on onnistunut hyvin. Järjestelmän käyttäjälle antamat ilmoitukset ja virheviestit opastavat käyttäjää antamaan syötteet oikeassa muodossa.

Testauksessa ilmennyt tiettyjen syötteiden virheellinen käsittely ei testien tulosten mukaan olisi aiheuttanut vakavia ongelmia järjestelmän toimivuuteen. Ainoana vakavana ongelmana oli sisäänkirjautumisen yhteydessä havaittu virhe. Käyttöönoton jälkeen tämä olisi voinut johtaa siihen, että asiaton käyttäjä olisi voinut kirjautua järjestelmään esimerkiksi vastuuhenkilön tunnuksella. Vaikka rakennustekniikan laboratorio ei yleisesti ottaen ole kovin tietoturvaton paikka, olisi se silti nykyajan sovelluksessa erittäin vakava virhe.

6 TULOKSET JA JOHTOPÄÄTÖKSET

Työn tavoitteena oli suunnitella ja toteuttaa sähköinen tavaroidenhallintajärjestelmä helpottamaan Rovaniemen ammattikorkeakoulun rakennustekniikan laboratorion käyttäjiä. Järjestelmän tuli kattaa lainausten, varausten ja lainaajienhallinnan lisäksi tavaroiden varastonhallinta. Lopputuloksena syntynyt järjestelmä vastaa pääosin näihin vaatimuksiin. Toimeksiantajan alkuperäisessä tilauksessa toivottiin, että lainaajien tunnistaminen hoidettaisiin henkilökortinlukijalla. Kortinlukijan toiminnallisuuden toteuttava ohjelmisto kuitenkin puuttui, ja siksi siitä luovuttiin jo projektin alkuvaiheessa. Henkilökortinlukijan ohjelmiston toteuttaminen olisi ollut liian laaja tehtävä tämän opinnäytetyön osalta. Lopulta päätimme toteuttaa tunnistamisen itse keksittävillä tunnisteilla. Lainaajien tunnistaminen omilla henkilökohtaisilla tunnisteilla lienee riittävän yksinkertainen tapa myös käytettävyyden kannalta.

Projektiin valitut työvälineet ja työskentelymenetelmät toimivat moitteettomasti koko projektin ajan. Myös kanssakäyminen toimeksiantajan kanssa toimi sujuvasti. Projektin aikataulua ei tarvinnut muuttaa, vaan jokainen vaihe valmistui aikataulussa.

Työmäärän arviointi on ohjelmistoalalla yleensä melko vaikeaa. Projektin alussa tehty arvio työtuntien määrästä oli 300. Toteutuneet työtunnit projektin kaikki vaiheet laskettuna mukaan oli 210. Kun otetaan huomioon se, että tämä oli ensimmäinen asiakasprojektini, ei arvion epätarkkuus tullut yllätyksenä.

Ongelmakohdat ja hankaluudet

Vaatimusmäärittelyprosessissa olisi voitu käyttää enemmän aikaa hyvien kysymyksien miettimiseen toimeksiantajalle. On selvää, että toimeksiantaja ei ole täysin perehtynyt ohjelmistokehitykseen, eikä näin ollen osaa ajatella prosessia täysin ohjelmistokehityksen näkökulmasta. Tässä tapauksessa olisi ollut hyödyllistä käyttää enemmän aikaa myös toimeksiantajan kanssa yhdessä otettaessa selvää järjestelmän vaatimuksista. Toteutusvaiheessa huomattiin, että tietokannansuunnittelu oli haasteellista johtuen määrittelyvaiheen puutteellisesta selvitystyöstä.

Käyttöliittymän ja järjestelmän logiikan toteuttamisessa eniten aikaa veivät varausten ja lainausten hallintatoiminto. Nämä vaativat monimutkaisia tarkistuksia tietokannasta. Myös hakutoimintojen toteuttaminen monilla erilaisilla hakulauseilla vei aikaa. Työssä valittu else- ja if-lohkojen käyttäminen hakulauseiden rakentamisessa toimi tämän järjestelmän osalta tarpeeksi hyvin. Hakutoiminnot saatiin toteutettua riittävän dynaamisiksi ja nopeiksi.

Opinnäytetyön aikataulusta johtuvista syistä järjestelmää ei ehditty ottaa käyttöön tämän työn puitteissa. Nähtäväksi jää, otetaanko järjestelmä käyttöön tulevaisuudessa.

Jatkokehitysideoita

Jatkokehitystä silmällä pitäen eniten mahdollisuuksia löytyy käyttöliittymästä. Nyt toteutettu järjestelmä on sidottu työasemaan ja toimivaan tietokantayhteyteen. Olisi erittäin hyödyllistä pystyä käyttämään järjestelmää myös langattomasti. Mobiilisovellus tai vaihtoehtoisesti web-selainkäyttöliittymä mahdollistaisi järjestelmän langattoman käytön. Tällöin langattomassa verkossa oleva kannettava tietokone, puhelin tai tabletti voisi toimia päätelaiteena. Näiden toteuttamisessa tietokantaan ei tarvitsisi tehdä merkittävämpiä rakenteellisia muutoksia. Kuten nykyisessä järjestelmässäkin, ainoa suurempi haaste on saada järjestelmä toimimaan myös offline-tilassa. Järjestelmä tulisi toteuttaa siten, että tietokanta tallentuisi väliaikaisesti päätelaitteeseen, jolloin kantaan voitaisiin tehdä kyselyjä ja muutoksia. Kun laite siirtyisi online-tilaan, tulisi laitteen automaattisesti päivittää muutokset tietokantaan. Tämä on toteutettavissa, mutta ei niin yksinkertaisesti kuin nykyinen versio.

Käytettävyyden parantamisen näkökulmasta WPF-tekniikka olisi oikea valinta järjestelmän seuraavaa versiota kehitettäessä. WPF-tekniikka on uudempi ja nykyaikaisempi sekä tarjoaa käyttöliittymään visuaalisesti monipuolisempia ratkaisuja. Käytettävyyttä parantaisi myös henkilökortinlukija, jonka avulla lainaajat voitaisiin tunnistaa.

Koko järjestelmän dynaamisuuden kannalta mahdollisuus tietokannan kokonaisvaltaiseen muokkaamiseen olisi tervetullut. Tässä projektissa tehdyssä käyttöliittymässä tiloja ja lainaajatyyppejä ei käyttäjä pysty muokkaamaan. Myöskään tietokantataulujen attribuutteja ei pysty lisäämään, muokkaamaan

tai poistamaan. Kyseisten muutosten tekeminen edellyttäisi käyttäjältä tietokantaosaamista, mutta toimintona se tekisi järjestelmästä paljon dynaamisemman.

Johtopäätökset

Testauksen tuloksien mukaan järjestelmä on käyttökelpoinen ja valmis käyttöönotettavaksi. Järjestelmän käyttäjien tueksi tehtiin myös käyttöohje, joka kattaa suurimman osan järjestelmän käyttötapauksista. Ennen käyttöönottoa täytyy kuitenkin tehdä päätös siitä, mille palvelimelle tietokanta luodaan. Tässä työssä tehtyjen testien perusteella internetverkon kautta toimiva palvelin osoittautui täysin riittäväksi, eikä häiritsevää latenssia juurikaan syntynyt. Jos verkkoyhteyksien toimivuus halutaan varmistaa, paras ratkaisu lienee asentaa tietokantapalvelin ammattikorkeakoulun omaan verkkoon.

Kaiken kaikkiaan projekti onnistui erittäin hyvin. Työn aihe oli haastava, monimuotoinen ja opettava. Tietokantatuntemus ja C#-ohjelmointiosaamiseni kehittyivät työn aikana merkittävästi. Myös ohjelmistokehityksen ymmärtäminen lisääntyi, ja projekti antoi runsaasti käytännön kokemusta. Onnistuminen ensimmäisessä asiakasprojektissa antoi runsaasti luottamusta omiin kykyihini. Motivaationikin lisääntyi entisestään tulevaa työelämää silmällä pitäen. Tietotekniikan alalla korostuu itsensä kehittämisen tärkeys ja se konkretisoitui tässäkin opinnäytetyössä.

LÄHTEET

Anttonen, M. 2006. C#-kielen perusteita. Osoitteessa http://www2.amk.fi/mater/tietotekniikka/c_hash/. 1.10.2012.

Kero, P. 2000. Microsoft .NET. Helsingin yliopiston Tietojenkäsittelytieteen laitos 2000. Osoitteessa http://www.cs.helsinki.fi/u/laine/otv/kero/kero.html. 4.1.2013.

Kolari, M. 2012. Windows Forms -sovellukset. Osoitteessa http://mikakolari.fi/csharp-dotnet/winforms/lomakkeet/. 4.12.2012.

Mannila, K. 2013. Mikäs on mikäs? Osoitteessa http://www.peda.net/veraja/jklammattiopisto/tekninen/opettajat/leinonen/kotisivu/leinonen/mannila/kysymyksia. 6.2.2013.

Moon Soft 2013. Sybase PowerDesginer 15.2. Osoitteessa http://www.moonsoft.fi/products/000276.aspx. 4.1.2013.

MSDN 2013. Introduction to the C# Language and the .NET Framework. Osoitteessa http://msdn.microsoft.com/library/z1zx9t92(VS.100).aspx. 1.10.2012.

Ohjelmointiputka 2011. C++-ohjelmointi: Osa 9 - Luokkien perusteet. Osoitteessa http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=cpp_ohj_09. 6.2.2013.

Oracle FAQ's 2004. What are the difference between DDL, DML and DCL commands? Osoitteessa http://www.orafaq.com/faq/what_are_the_difference_between_ddl_dml_and_dcl_commands. 4.12.2012.

Oulun seudun ammattiopisto 2004. MySQL-tietokannan käyttö ja SQL-kielen perusteet. Osoitteessa http://www.okol.org/verkkokurssit/datanomi/tietojarjestelmien_kehittaminen/tiedonhallintajarjestelmat/sqlkieli.htm. 4.12.2012.

PHPeasystep 2013. Introduction to MySQL database. Osoitteessa http://www.phpeasystep.com/mysql/1.html. 4.12.2012.

Wikikirjasto 2008. C/Historia. Osoitteessa http://fi.wikibooks.org/wiki/C/Historia. 6.2.2013.

LIITTEET

Esimerkkikoodi varauksen peruuttamistoiminnosta Liite 1
Tietokantataulujen kuvaustaulukot Liite 2

Esimerkkikoodi varauksen peruuttamistoiminnosta Liite 1

```
private void poistaVarausListalta()
{ //poistetaan ensin varaus varaus-taulusta
  string mySql1 = "DELETE from VARAUS WHERE(TAVARAID = '"
  + tavaraId + "') AND (LAINAAJAID = " + lainaajaId.ToString()
  + ")";
  int saatavuus = 0;
  try
  {
    using (connection)
    {
      if (connection.State != System.Data.ConnectionState.Open)
      {
        connection.Open();
      }
      MySqlCommand command = connection.CreateCommand();
      command.CommandText = mySql1;
      //ajetaan komento
      command.ExecuteNonQuery();
      connection.Close();
      MessageBox.Show("Varaus poistettu onnistuneesti tunnukselta: "
      + lainaajanTunnisteTxtBox.Text.ToString(), "Ilmoitus");
    }
    if (onkoTavaraLainassa()) //tarkastetaan onko tavara lainassa
    {
      saatavuus = 1; //jos on lainassa saatavuus on 1
    }
    if (onkoTavarallaVarauksia() && !onkoTavaraLainassa())
    //tarkastetaan onko tavaralla varauksia ja onko se palautettu
    {
      saatavuus = 2; /*jos on varauksia ja tavara on varastossa,
      saatavuus on 2*/
    }
    else
    {
      saatavuus = 3; //muussa tapauksessa saatavuus on 3, eli vapaa
    }
    /*muodostetaan päivityslause tavaran saatavuuden päivittämisek-
    si*/
    string mySql2 = "update TAVARA set saatavuus = " + saatavuus + "
    where tavaraid = " + tavaraId;
    using (connection)
    {
      if (connection.State != System.Data.ConnectionState.Open)
      {
        connection.Open();
      }
      MySqlCommand command = connection.CreateCommand();
      command.CommandText = mySql2;
      command.ExecuteNonQuery();
      connection.Close();
      MessageBox.Show("Saatavuustilanne päivitetty onnistuneesti
      tavaralle " + tavaraId, "Ilmoitus");
    }
    /*jos tavaralla on varauksia, aletaan asettamaan päälimmäiseksi
     tulleen varauksen eräpäivää*/
    if (onkoTavarallaVarauksia())
    {
      try
      {
        string varausId = "";
        string sql =   "SELECT        VARAUSID "
```

```
                        + "FROM            VARAUS "
                        + "WHERE        (VARAUSPAIVAMAARA = "
                        + "(SELECT   MIN(VARAUSPAIVAMAARA) AS Expr1 "
                        + "FROM   VARAUS "
                        + "WHERE   (TAVARAID = " + tavaraId + ")))";
           /*etsitään varaus-taulusta varauksen id-numero, joka on
varauslistan vanhin, eli jonka varauspaivamaara on ”pienin”, ja jol-
la on viittaus kyseessä olevaan tavaraan*/
        MySqlCommand cmdSel1 = new MySqlCommand(sql, connection);
        MySqlDataReader myReader = null;
        if (connection.State != System.Data.ConnectionState.Open)
        { connection.Open(); }
        myReader = cmdSel1.ExecuteReader();
        while (myReader.Read())
        {
          try
          {//tallennetaan varauksen id-numero muuttujaan
            varausId = myReader["varausid"].ToString();
          }
          catch (MySqlException e) { MessageBox.Show("Virhe: " +
           e.ToString(), "Virhe"); }
        }
        myReader.Close();
        /*muodostetaan lause jolla päivitetään uuden päällimmäisenä
        olevan varauksen lunastuspäivämääräksi 7 päivää nykypäivä-
        määrästä eteenpäin*/
        sql = "update VARAUS set varauslunastuspaivamaara =
        DATE_ADD(NOW(),INTERVAL 7 DAY) where varausid = " +
        varausId;
        MySqlCommand cmdUpdate = new MySqlCommand(sql, connection);
        cmdUpdate.ExecuteNonQuery();
        MessageBox.Show("Varauksen eräpäivän päivitys onnistui.",
        "Huomautus");
      }
      catch (Exception e)
      {
        MessageBox.Show("Varauksen eräpäivän päivityksessä tapahtui
        virhe " + e.ToString(), "Virhe");
      }
    }
  }
  catch (MySqlException me)
  {
    MessageBox.Show("Virhe poistettaessa varausta: " +
     me.ToString(), "Virhe");
  }
  //paivitetaan saatavuustilanne 3:ksi, jos arvo on ollut 2, eli
  pelkästään varattuna.

  muodostaLainVarSelect(); //päivitetään tietotaulukko
}
```

Tietokantataulujen kuvaustaulukot Liite 2

Taulukko 1. Lainaaja-taulu

Attribuutti	Tyyppi	Selite
LAINAAJAAJAID	int not null, auto increment	lainaajan id-numero pääavain
LAINAAJAATYYPPIID	int not null	viiteavain lainaajatyyppi-tauluun
TUNNISTE	varchar(15) not null	lainaajan tunniste
RYHMA	varchar(15)	oppilaan ryhmä
ETUNIMI	varchar(30) not null	lainaajan etunimi
SUKUNIMI	varchar(30) not null	lainaajan sukunimi
PUHELINNUMERO	varchar(50)	lainaajan puhelinnumero
SAHKOPOSTIOSOITE	varchar(50)	lainaajan sähköpostiosoite
SALASANA	varchar(50) not null	salasanan MD5-tiiviste
LISATIEDOT	varchar(500)	lainaajan lisätiedot

Taulukko 2. Tavara-taulu

Attribuutti	Tyyppi	Selite
TAVARAID	bigint not null, auto increment	tavaran id-numero (viivakoodi) pääavain
POSITIOID	int not null	viiteavain positio-tauluun
TILAID	int not null	viiteavain tila-tauluun
KATEGORIAID	int not null	viiteavain kategoria-tauluun
NIMI	varchar(50) not null	tavaran nimi
HANKINTAPAIVA	date	tavaran hankintapäivämäärä
KALIBROITU	date	tavaran edellinen kalibrointipäivämäärä
SEURAAVA_KALIBROINTI	date	tavaran seuraava kalibrointipäivämäärä
INVENTAARIO	date	edellinen inventaariopäivämäärä
SAATAVUUS	varchar(500)	tavaran saatavuus (vapaa/varattu/lainassa)
LISATIEDOT	int not null	tavaran lisätiedot

Taulukko 3. Lainaus-taulu

Attribuutti	Tyyppi	Selite
LAINAUSID	int not null, auto increment	lainauksen id-numero pääavain
LAINAAJAID	int not null	viiteavain lainaaja-tauluun
TAVARAID	bigint not null	viiteavain tavara-tauluun
LAINAUSPAIVAMAARA	date	lainauspäivämäärä
PALAUTUSPAIVAMAARA	date	lainauksen eräpäivä
LISATIEDOT	varchar(500)	lainauksen lisätiedot
PALAUTETTU	varchar(10)	lainaus palautettu vai ei (ok/ei)

Taulukko 4. Varaus-taulu

Attribuutti	Tyyppi	Selite
VARAUSID	int not null, auto increment	varauksen id-numero pääavain
LAINAAJAID	int not null	viiteavain lainaaja-tauluun
TAVARAID	bigint not null	viiteavain tavara-tauluun
VARAUSPAIVAMAARA	datetime	varauspäivämäärä
VARAUSLUNASTUSPAIVAMAARA	datetime	varauksen päättymispäivämäärä
LISATIEDOT	varchar(500)	varauksen lisätiedot

Taulukko 5. Lainaajatyyppi-taulu

Attribuutti	Tyyppi	Selite
LAINAAJATYYPPID	int not null, auto increment	lainaajatyypin id-numero pääavain
NIMI	varchar(50) not null	lainaajatyypin nimi

Taulukko 6. Tila-taulu

Attribuutti	Tyyppi	Selite
TILAID	int not null, auto increment	tilan id-numero pääavain
NIMI	varchar(50) not null	tilan nimi
LISATIEDOT	varchar(500)	tilan lisätiedot

Taulukko 7. Kategoria-taulu

Attribuutti	Tyyppi	Selite
KATEGORIAID	int not null, auto increment	kategorian id-numero pääavain
TILAID	int not null	viiteavain tila-tauluun
NIMI	varchar(50) not null	kategorian nimi
LISATIEDOT	varchar(500)	kategorian lisätiedot

Taulukko 8. Positio-taulu

Attribuutti	Tyyppi	Selite
POSITIOID	int not null, auto increment	position id-numero pääavain
TILAID	int not null	viiteavain tila-tauluun
NIMI	varchar(50) not null	position nimi
LISATIEDOT	varchar(500)	position lisätiedot