WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

Bachelor Thesis

Titel of Bachelor Thesis (english)	Point of Sale Software: An Open Source Approach Based on the Austrian Cash Register Security Regulation
Titel of Bachelor Thesis (german)	Point of Sale Software: Ein Open Source Ansatz in Anlehnung an die österreichische Registrierkassensicherheitsverordnung
Author (last name, first name):	Langer Daniel
Student ID number:	01451548
Degree program:	Bachelor of Science (WU), BSc (WU)
Examiner (degree, first name, last name):	Prof. Dr. Rony G. Flatscher

I hereby declare that:

1. I have written this Bachelor thesis myself, independently and without the aid of unfair or unauthor-ized resources. Whenever content has been taken directly or indirectly from other sources, this has been indicated and the source referenced.
2. This Bachelor Thesis has not been previously presented as an examination paper in this or any other form in Austria or abroad.
3. This Bachelor Thesis is identical with the thesis assessed by the examiner.
4. (only applicable if the thesis was written by more than one author): this Bachelor thesis was written together with

   The individual contributions of each writer as well as the co-written passages have been indicated.

07/04/2019
Date

Daniel Langer
Unterschrift

POINT OF SALE SOFTWARE: AN OPEN SOURCE APPROACH BASED ON THE AUSTRIAN CASH REGISTER SECURITY REGULATION

April 7, 2019

Author: Daniel Langer

Student ID: 01451548
Supervisor: Prof. Dr. Rony G. Flatscher
Vienna University of Economics and Business

Abstract

Since 2017 most companies have to comply with the Austrian Cash Register Security Regulation. This forces them to pay monthly fees for software that abides by the regulations. This practical work provides an open source alternative. The free, open source programming language Open Object Rexx (ooRexx) is used with BSF4ooRexx for effortless software development that can be understood quickly without prior knowledge.

This paper gives an introduction to the ooRexx language so even programming beginners can follow along.

With BSF4ooRexx the developed software is also making use of Java while still writing with the simple syntax of ooRexx. This also allows the use of JavaFX which is used to create the graphical user interface.

The database used in this work is MySQL. It is shown how to make use of it in ooRexx. Besides the practical use of the database the paper is also describing how to model databases so that they are free of redundancies and anomalies.

Besides the core features of a *Point of Sale* software like handling orders and products this work also describes additional features like password hashing, encryption and receipt printing.

Contents

List of Figures

Listings

Abbreviations

BSF4ooRexx Bean Scripting Framework for ooRexx

BSF4Rexx Bean Scripting Framework for Rexx

CSS Cascading Style Sheets

ECDSA Elliptic Curve Digital Signature Algorithm

GLP GNU General Public License

GUI graphical user interface

HTML Hypertext Markup Language

JVM Java Virtual Machine

JWS JSON Web Signature

ooRexx Open Object Rexx

PBKDF2 Password-Based Key Derivation Function 2

POS Point of Sale

RKSV Cash Register Security Obligation (Registrierkassensicherheitsverordnung)

SHA-256 Secure Hash Algorithm 256

1 Introduction

On April 1st, 2017 the Cash Register Security Obligation (Registrierkassensicherheitsverordnung) (RKSV) came into effect. The regulation provides different requirements that cash registers in Austria have to fulfill [WKO16]. *Point of Sale (POS)* software needs to meet these requirements to conform with the law. As it is not an easy task to develop a software that meets all the requirements companies have to use expensive third party software. Therefore there is a desire for a free to use, open source alternative. The goal of this work is to develop an open source software freely accessible to everyone with a public source code that can be used, modified or expanded by everyone.

POS is an abbreviation for *Point of Sale* and describes the time and place where a retail transaction is completed [Wik19a]. When talking about POS software it can either stand for *Point of Service* software or *Point of Sale* software. Both terms are used for very similar things: While *Point of Sale* software only handles the sale process, *Point of Service* software includes additional features like table reservation and invoice printing [Wik19b, Ora19]. The developed software will mainly implement a *Point of Sale* system but also include some features of a *Point of Service* system like invoice printing.

Figure 1 shows an example setup. POS software is mostly used on touch screen devices. This should be kept in mind when designing the graphical user interface. The second part of the setup besides the touch screen device is a receipt printer.

Figure 1: "Point of Sale" - Example Setup

Source: Travelarz, Wikipedia (CC BY-SA 3.0) [Com19]

2 Cash Register Security Regulation

The BMF (Austrian Ministry of Finance / Bundesministerium für Finanzen) describes a cash register as "any electronic recording system used for the daily tally of proceeds and documentation of individual cash takings" [BMF19]. Cash registers are required for businesses with a net annual turnover of €15,000 per operation and cash transactions exceeding €7,500 net per year [BMF19]. These cash registers have to comply with the Cash Register Security Regulation. The main purpose of the regulation is to prevent fraud and manipulation. It provides different security measures to reach this goal. These are described in section 6.3.

3 Installation Guide

This section will focus on installing all the necessary tools to run the *Point of Sale* software. As Windows is the most used operating system it is used to show the installation process. However the installation process for Linux and MacOS should behave very similarly.

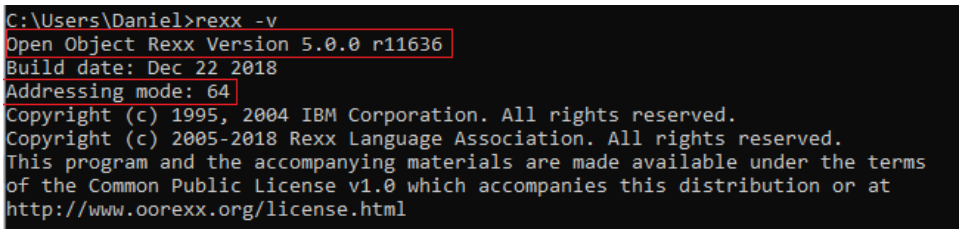
3.1 ooRexx

ooRexx is the programming language that is used to develop the *Point of Sale* software. Stable releases can be found at http://www.oorexx.org/download.html. The latest stable release at the time of writing this (January 2019) is release 4.2.0. Newer, beta releases as well as older releases are available under https://sourceforge.net/projects/oorexx/files/oorexx/. Open Object Rexx Version 5.0.0 beta is used to develop this program. Earlier versions are not recommended to run this software as they might not be compatible.
To download Open Object Rexx head over to Sourceforge.net (link mentioned above) and click on the *5.0.0beta* folder. This will display all the different version for the selected release. Select the version that fits your operating system and the bit version of your PC (32 bit or 64 bit). An instruction on how to detect which system is currently running is included in the appendix (Appendix 11.1). After downloading the installation file, run it and follow the steps shown on screen. The different settings during the installation process can be left at default. With the following steps the installation can be verified:

- Open the command prompt:
  - Windows Key + "R" Key
  - Type "cmd" and press enter
- Type "rexx -v" and press enter

If the installation was successful, it should yield the following output (Figure 2):

Figure 2: "rexx -v" Output



The output shows the currently installed version of Rexx. In this case the 64bit version of Open Object Rexx 5.0.0 is installed.

3.2 Java

With BSF4ooRexx it is possible to use Java while still writing with the easy syntax of Rexx. Before BSF4ooRexx can be installed Java SE Development Kit (SDK) needs to be installed on the system. This kit includes the Java runtime environment (JRE) and additionally tools for development like compilers and debuggers. As the *Point of Sale* software is going to use Java SDK 8 it is recommended to install the same version so there won't be any compatibility issues. This version can be found at https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html. Similarly to Rexx, this will display multiple versions for different operating systems and 32 bit (x86) or 64 bit (x64) systems. It is important that the bit version matches the bit version chosen for ooRexx. Before downloading is is necessary to read and accept the license agreement. After the download is completed, run the setup file and follow the installation process. After the installation is complete the *bin* directory (located at *C:\Program Files\Java\jdk1.8.0_191\bin* or *C:\Program Files (x86)\jdk1.8.0_191\bin*) needs to be included in the *Path* environment variable. (see Appendix 11.2)

OpenJDK

Besides the JDK provided by Oracle it is also possible to use the open source alternative *OpenJDK*. It is freely available at https://adoptopenjdk.net/. Click on the version (OpenJDK 8 recommended) and JVM of your choice and then on the *Latest release* button to downloaded the JDK. Clicking on *Help me Choose* shows the difference between the two available JVMs *HotSpot* and *OpenJ9*.

3.3 BSF4ooRexx

BSF4ooRexx "makes all of Java directly available to ooRexx and vice versa" [Sou19]. Before installing it is important that ooRexx and Java are already installed on the system. BSF4ooRexx is available at https://sourceforge.net/projects/bsf4oorexx/. Download and unpack the ZIP file. The directory ./bsf4oorexx/install contains three directories for the three main operating systems. Open the directory corresponding to your operating system and run the file named "install". If OpenOffice is not installed on the system the installation might pause and display the following error message (Figure 3):

Figure 3: Open Office Error Message

```
setupAllAndRun.rex: error while setting up OpenOffice support: res=[-14].
Maybe AOO/OOo/LO is not installed?
2019-01-18T13:49:18.974000: Hit enter to continue ...
```

As Open Office won't be used, the error message can be ignored. Press enter to continue the installation process. Once the installation is complete it will display a message "Installation of 'BSF4ooRexx' was succesful!". Hit enter again to close the window.

3.4 MySQL

MySQL Community 8 and Connector J 8.0.13,
GNU General Public License
MySQL has a freely available community version under GNU General Public License (GLP) license. It can be found at https://dev.mysql.com/downloads/mysql/. The bottom of the page contains all installers for different operating systems. The easiest way to set MySQL up on Windows is to use the MySQL Installer. It is available at https://dev.mysql.com/downloads/windows/installer/8.0.html. On the bottom of the page you can choose either the offline or web installer. After downloading the installer run the *.msi* file (for Windows) to start the installation process. Follow the steps in the installation process while keeping the default options. Only the following windows require changes:

- *Check Requirements*: If Visual Studio is shown as a requirement it can be ignored as it is not needed for the POS software. Any other requirement should be satisfied as they might be needed to function properly.

- *Accounts and Roles*: Choose a root password and enter it twice. This is used to connect to the database. Press *next* to continue.

- *Connect To Server*: Enter the password you chose in the *Accounts and Roles* window and hit *check*. If the entered password is correct the status will change to *Connection succeeded*.

The last step is to include *mysql-connector-java* in the *CLASSPATH* variable. This can be done either manually or (only for Windows and Linux) by adding the JAR to the /bin/ folder of the POS software. The JAR is located at *INSTALLATION_PATH/MySQL/Connector J 8.0*. In most cases the full path for windows and version 8.0.13 is *C:\Program Files (x86)\MySQL\Connector J 8.0\mysql-connector-java-8.0.13.jar*

3.5 Java Archives (JAR)

All JARs listed below are required for the program to work properly. To include JARs they need to be downloaded and added to the *CLASSPATH* environment variable (see Appendix 11.2). For Windows and Linux systems it is also possible to add all JARs to the /bin/ folder of the POS software. The JARs will then be included automatically in the *CLASSPATH* for the runtime of the program.
In total 9 different JARs are used:

- MySQL:
  - mysql-connector-java-8.0.14.jar: Is used to access the MySQL database from ooRexx (Section 3.4)
- ZXING:
  - core-3.3.3.jar: Is the core image decoding library (Section 3.5.1)
  - javase-3.3.3.jar: Java specific extension for the core library (Section 3.5.1)
- Google Material Design
  - jfoenix-8.0.8.jar: Implements the Google Material Design (Section 3.5.2)
- JSON Web Signature:
  - jose4j-0.6.5.jar: Implements the JSON Web Signature (Section 3.5.3)
  - slf4j-api-1.7.25.jar: Simple Logging Facade for Java API - Required for Jose4J (Section 3.5.3)
  - slf4j-simple-1.7.25.jar: Includes the StaticLoggerBinder Class that is used in the SLF4J API (Section 3.5.3)

3.5.1 QR-Code: zxing

Zxing: Version 3.3.3, Apache License Version 2.0
Zxing is used to create QR-Codes for the receipts. The two JARs needed can be found at https://github.com/zxing/zxing/wiki/Getting-Started-Developing. The first paragraph *Just Need a JAR?* includes a link to the Maven release repository. In the repository you can find the two JARs that are needed: *core-3.3.3.jar* and *javase-3.3.3.jar*. First select the needed JAR (core or javase) then the version and lastly find the correct JAR file. After downloading both JARs they need to be included in the *CLASSPATH* variable.

3.5.2 JFoenix - Google Material Design

JFoenix: Version 8.0.8, Apache License Version 2.0
JFoenix is used to improve the graphical user interface. It is used to create multiple menus as shown in Section 6.1.1. The library is available at https://github.com/jfoenixadmin/JFoenix. The JAR for Java Version 8 can be found in the first section under *JFoenix for Java 8 - download jar (8.x.x)*. If Java 9 is used it is important to get JFoenix for Java 9. After downloading, include the JAR in the *CLASSPATH* variable.

3.5.3 JSON Web Signature: Jose4j and SLF4J

Jose4J: Version 0.6.5, Apache License 2.0
SLF4J: Version 1.7.25, MIT license
Jose4j is an open-source implementation for JWS (JSON Web Signature) [Cam19]. JWS is used for signing data which later is necessary for the receipt creation. All versions are available at https://bitbucket.org/b_c/jose4j/downloads/. Version 0.6.5 is used for this work. To work properly Jose4j also requires SLF4J. The ZIP file containing all required JARs is available at https://www.slf4j.org/download.html under stable releases. Only two of the many included JARs are required. These are: *slf4j-api-1.7.25.jar* and *slf4j-simple-1.7.25.jar*.

3.6 Scene Builder

Used version: 8.5.0
Scene Builder is used to create the graphical user interface (GUI). This program is optional and only needed if there is a desire to change the existing GUI or create new ones. The builder for Java 8 can be found at https://gluonhq.com/products/scene-builder/ in the section *Download Scene Builder for Java 8*.

4 Programming Languages and Concepts

This section describes the programming languages, query language and framework used in this application. This includes ooRexx, BSF4ooRexx, Java and MySQL.

4.1 ooRexx

The source code of the developed POS software is written in the ooRexx programming language. The syntax is described in great detail so it is easier to follow along with the rest of the work.

4.1.1 History

Rexx was developed in 1979 at IBM by Mike F. Cowlishaw as a successor to EXEC with the goal to create a human-centric language [Fla17]. The name is an acronym for "**RE**structured e**X**tended e**X**ecutor" [IBM14b]. The languages provides easy-to-use instructions and flexible syntax and format [IBM14a] which makes it a perfect language to develop programs easily and quickly even for beginners. In April 2005 the Rexx Language Associatation released the open source version Open Object Rexx (ooRexx) [Fla17].

4.1.2 Case Sensitivity

The Rexx interpreter translates all commands to upper case and is therefore not case sensitive (Listing 1).

Listing 1: Case Sensitivity

```
say "Hello"
SAY "Hello"
sAy "Hello"
/* All commands results in the same output */
/* Output: Hello */
```

This is different to many other programming languages where functions written in different cases won't work.

4.1.3 Printing to Console

In Rexx the command *SAY* is used to print to the standard output device (Listing 2). It automatically appends a newline character to the output.

Listing 2: Hello World

```
SAY "Hello World"
SAY "How are you?"
/* Output:
Hello World
How are you?
*/
```

4.1.4 Variables

Variables make it possible to store, change and retrieve values and reference them with a discretionary name called an identifier [Fla17]. ooRexx is a dynamically typed programming language and therefore it is not necessary to declare a type for a variable. This makes it possible to write code faster and more flexible.

Listing 3: Variables

```
x = 2
y = "Hello"
```

In the example above x is the identifier for the literal *2* and y the identifier for the string literal *Hello* (Listing 3).
Variables can also be used to store the result of an expression, e.g. the sum of two numbers (Listing 4).

Listing 4: Simple Arithmetics

```
x = 3
y = 5
z = x + y
SAY z       /* Output: 8 */
```

Unlike other programming languages two different variables with the same name but different upper/lowercase letters are the same (Listing 5).

Listing 5: Case Sensitivity for Variables

```
age = 20
Age = 25
SAY age    /* Output: 25 */
```

If - Statements

If - statements allow the execution of code if a specific defined condition is true.

Listing 6: If Statements

```
PARSE PULL age
IF age > 30 THEN DO
    SAY "You are over 30 years old"
END
ELSE DO
    SAY "You are 30 years old or younger"
END
```

The code example above takes the input of the user and stores it in the variable *age* (Listing 6). If the first expression (age bigger than 30) evaluates to true then the first code block will be executed otherwise it continues with the *ELSE* code block.
Additional conditions can be checked with the so called *ELSE IF* statement (Listing 7).

Listing 7: If - Else If - Else

```
PARSE PULL age
IF age > 30 THEN DO
    SAY "You are over 30 years old"
END
ELSE IF age < 30 THEN DO
    SAY "You are under 30 years old"
END
ELSE DO
    SAY "You are 30 years old"
END
```

In this case the block with the condition that first evaluates to true will be executed. If none of the conditions is true then the code block of the else statement will be executed.

When checking multiple conditions it is often preferred to make use of the *SELECT* statement (Listing 8). It checks multiple conditions and only if none of them evaluate to true the code block after *OTHERWISE* will be executed (similarly to *ELSE*).

Listing 8: Select

```
PARSE PULL age
SELECT
        WHEN age > 30 THEN SAY "You are over 30 years old"
        WHEN age < 30 THEN SAY "You are under 30 years old"
        OTHERWISE SAY "You are 30 years old"
END
```

Loops

When creating programs it is often necessary to repeat a block of code multiple times. This would be possible without loops but makes the code very hard to read. Besides iterating a certain number of times loops can also be used to iterate over table elements or collection classes like arrays. Both code examples below count to 5, one without loop (Listing 9) and one with loop (Listing 10, TO DO loop example).

Listing 9: Printing Multiple Times without Loops

```
SAY 1
SAY 2
SAY 3
SAY 4
SAY 5
```

The *DO TO* loop is used to repeat a block of code a certain number of times. It uses a control variable (in this case i) that is incremented by one at the end of each iteration.

Listing 10: "DO TO" Loop

```
DO i=1 TO 5
    SAY i
END
```

In the example of counting to 5 the difference between using a loop and not using a loop might not seem too big. But just considering the changes that are necessary to count to 100 might make the advantage of loops clearer. In the example without loops we have to add 95 more lines of code to get the desired result. In the loop example a single number needs to be changed. The code structure does not change and the changes do not interfere with the readability.

It is also possible to loop a certain number of times without using the control variable.

Listing 11: "DO" Loop

```
DO 3
    SAY "Hello"
END
/* Output: Hello
Hello
Hello */
```

The *DO WHILE* loop repeats a block of code as long as the defined condition is true.

Listing 12: "DO WHILE" Loop

```
i = 2
DO WHILE i <= 512
    SAY i
    i *= 2
END
```

To easily loop over collection classes like arrays the DO OVER loop is used. It loops over all the collected objects and stores them temporarily in a variable. For example to square all elements of an array the following code could be used:

Listing 13: "DO OVER" Loop

```
numbers = .array~of(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

DO num OVER numbers
        SAY num**2
END
```

4.1.5 Routines

Routines are defined to make a block of code reusable. This improves the structure and readability of the code. Routines are either used as procedures (no return value) or functions (with return value). Routines are treated as if they were proper programs and therefore have their

own scope. If the keyword *PUBLIC* is given routines are also available for other calling programs [Fla17]. In this case the calling program needs to use the *::REQUIRES* directive to make the routine accessible (see Section 4.1.6).

Listing 14: Routine

```
numbers = .array~of(1, 2, 3, 4, 5, 6)
SAY sum(numbers)

::ROUTINE sum
    USE ARG a
    sum = 0
    DO val OVER a
        sum += val
    END
    RETURN sum
```

4.1.6 Requires

The *::REQUIRES* directive allows naming a Rexx program which is then called before executing other directives. This makes all public routines and public classes available to the calling program [Fla17]. In the example below one script (file1.rex) calls the routine *helloWorld* that is defined in another script (file2.rex).

file1.rex

```
CALL helloWorld

::REQUIRES "file2.rex"

/* Output: Hello World called from file2 */
```

file2.rex

```
::ROUTINE helloWorld PUBLIC
        SAY "Hello World called from file2"
```

4.1.7 Object Orientation

The goal of object oriented programming is to simulate objects in the real world as closely as possible [ooR09b]. To create these objects in code we define classes. A class can be seen as a template for objects. They define the set of acceptable values and allowable operations [Fla17]. The class **User** for example could have the following set of attributes:

- Username
- Password
- Email
- Date of registration

But a **User** is also able to perform different actions, which are defined by the allowable operations, such as:

- Login
- Change user information
- Delete account
- Turn on / off newsletter

In ooRexx they can be implemented with the *::CLASS directive* [Fla17]. The *::METHOD directive* is used either alone to define operations or with the keyword ATTRIBUTE to define attributes (acceptable values).
In the example below (Listing 17) a very simple **User** class is defined. This class has one attribute **username** and one method **init**. The INIT method is a special method called constructor. It is automatically executed when an instance of the class is created. The first line creates a new user named "Max" with the keyword **new**. The next line prints the username of the newly created user.

Listing 17: Class Directive

```
user1 = .User~new("Max")
SAY user1~username

::CLASS User
::METHOD username ATTRIBUTE
::METHOD init
    /* Expose instance variable to init method */
        EXPOSE username
        USE ARG username=.nil
        SAY "New user created."
```

The class defined above is very simplified and barely has any utility. To add some functionality the class is extended by one method and one attribute in the code example below (Listing 18). A new user is now created with a username and a password. Additionally, after a user is created it is possible to check if a given password matches the user password.

Listing 18: Advanced Class

```
user1 = .User~new("Daniel", "password123")
user1~checkPassword("abcd54321")
user1~checkPassword("password123")

/* Output:
Incorrect password!
Please try again!
Password correct!
*/

::CLASS User
::METHOD username ATTRIBUTE
::METHOD password ATTRIBUTE
::METHOD init
        EXPOSE username password
        USE ARG username=.nil, password=.nil
```

```
:: METHOD checkPassword
        EXPOSE password
        USE ARG passwordToCheck
        IF passwordToCheck == password THEN DO
                SAY "Password correct!"
        END
        ELSE DO
                SAY "Incorrect password!"
                SAY "Please try again!"
        END
```

4.1.8 Exception Handling

In ooRexx catching exceptions is possible either with

- *CALL on/off category [NAME label]* or
- *SIGNAL on/off category [NAME label].*

As this work mainly uses the latter it is used as an example and explained in further detail. To catch exceptions in a block of code you start with the **ON** keyword. It will then intercept either all exceptions (**ANY** keyword) or specific exceptions which are defined in the ooRexx docs [ooR09c]. The exception handling can be turned off with the **OFF** keyword.
The example below (Listing 19) intercepts syntax errors. The first *SAY* statement has no error and will therefore print the sum of 3 and 4. The second *SAY* statement will raise an exception because the addition of a number and a string does not work. It therefore will jump to the internal routine defined at the bottom of the file. Rexx will store the line that raised an exception in a variable named **SIGL**. The second *SAY* statement will therefore output *Error on line 3.*

Listing 19: Exception Handling

```
SIGNAL ON SYNTAX
SAY 3 + 4
SAY 3 + "X" /* Raises exception */
SIGNAL OFF SYNTAX
EXIT 0

SYNTAX:
        SAY "Error on line" SIGL
        EXIT -1
```

4.2 BSF4ooRexx

The following paragraph will introduce BSF4ooRexx as presented in "CAMOUFLAGING JAVA AS OBJECT REXX" at "The 2004 International Rexx Symposium", Böblingen, Germany by Rony G. Flatscher [Fla04].
The open source "Bean Scripting Framework" by IBM makes it possible to invoke programs from Java, which are written in a different programming language. Bean Scripting Framework for Rexx (BSF4Rexx) which was firstly introduced at the "2001 International Rexx symposium", not only makes it possible to invoke Rexx from Java but also to use Java as a function library for Rexx. This allows to use the full power of Java with the easy syntax of Rexx. Later, when version 4.0 of ooRexx was released, Bean Scripting Framework for ooRexx (BSF4ooRexx) was created which focused solely on Open Object Rexx. In code the full Java functionality can be made available to Rexx with *::REQUIRES BSF.CLS*. In the example below (Listing 20) the Java class **JOptionPane** is imported and stored in a variable named JPane. Next the *showInputDialog* class method is called which displays a simple GUI to get an input from the user (Figure 4). The user input is then stored in the variable *age* and printed to the command line.

Listing 20: Use Java in Rexx

```
JPane = bsf.import("javax.swing.JOptionPane")
age = JPane~showInputDialog("Please enter your age")

SAY "You are" age "years old."

::REQUIRES BSF.CLS
```

Figure 4: BSF4ooRexx Example: JOptionPane

Creating a Simple Program with Graphical User Interface (JavaFX)

This section uses *Anatomy of a GUI (graphical user interface)*, Flatscher R.G. (2018), "The 2018 International Rexx Symposium" [Fla18] as a reference. For a detailed description and more examples the original reference is recommended.

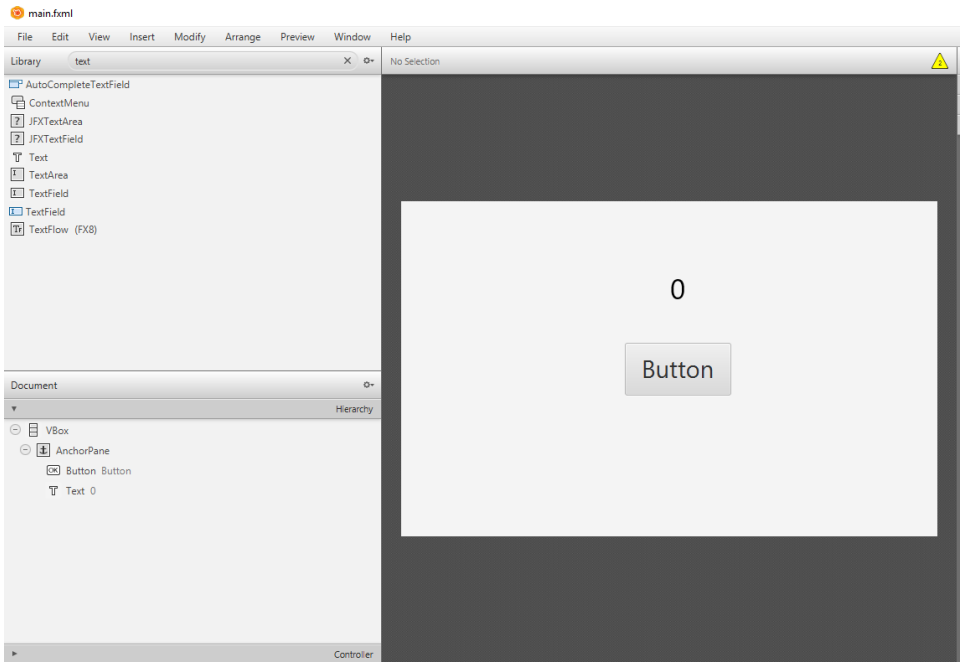
To describe how JavaFX, ooRexx and Scene Builder work together a simple GUI is created that counts the number of times a button is pressed. The three following files work together to create a GUI:

- FXML file: defines the graphical user interface
- Controller file: mainly handles the interactivity between the GUI and user (e.g. what happens when a button is clicked)
- Main file including a class that extends the Java abstract class *Application*: Runs the program using the JavaFX package [Fla19]

Building the GUI (main.fxml)

The GUI is created using Scene Builder (see 3.6) and the Basic Application template. The two necessary elements, a button and a text element, can be added via drag-and-drop from the library panel on the left to the GUI (see figure 5). After adding the elements the GUI can be saved as a fxml file.

Figure 5: Button Click Counter: Scene Builder



The created file can then be opened with a text editor to add ooRexx support, button on action routines and a controller source. Figure 6 displays the neccessary additions that need to be made.

Firstly, the tag *<?language rexx?>* is included to add the Rexx language support.

Secondly, the tag *<fx:script source="controller.rex" />* is added to include *controller.rex* as a script source. The controller file is created in the next step and includes the routine that is executed when the button is pressed.

Thirdly, the *onAction* property is added to the button. The first part *slotDir=arg(arg());* fetches the last supplied argument. The second part *call buttonClicked slotDir* will call the public routine *buttonClicked*, which is defined in *controller.rex*, and supply *slotDir* as an argument [Fla18].
Finally, an ID is defined for the text element so it can be accessed from code.

Figure 6: Button Click Counter: FXML File

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>

<VBox prefHeight="400.0" prefWidth="640.0" xmlns="http://javafx.com/javafx/8.0.171
" xmlns:fx="http://javafx.com/fxml/1">
  <fx:script source="controller.rex" />
  <children>
    <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0
    " VBox.vgrow="ALWAYS">
        <children>
           <Button layoutX="267.0" layoutY="169.0" mnemonicParsing="false"
           onAction="slotDir=arg(arg()); call buttonClicked slotDir" text="Button
           ">
              <font>
                 <Font size="29.0" />
              </font>
           </Button>
           <Text fx:id="counterText" layoutX="321.0" layoutY="117.0" strokeType="
           OUTSIDE" strokeWidth="0.0" text="0">
              <font>
                 <Font size="34.0" />
              </font>
           </Text>
        </children>
    </AnchorPane>
  </children>
</VBox>
```

Creating the controller script (controller.rex)

The controller script defines the public routine that is executed on each button click. The text element with fx:id "counterText" defined in fxml can be accessed with the Rexx script annotation */*@get(counterText)*/*. It is also possible to make multiple elements accessible by defining multiple elements inside parenthesis separated by spaces (e.g. */*@get(counterText homeButton)*/*). After making it accessible it can be used like any other variable. In this example the variable is incremented by one.

controller.rex

```
:: ROUTINE buttonClicked PUBLIC
  USE ARG slotDir
  /*@get(counterText)*/
  counterText~text += 1
```

Main script and application class (main.rex)

This script will be executed to run the program. It defines a class (in this example named **StageHandler**) that extends the **Application** class from the JavaFX library. The class implements the *start* method which loads the previously created *main.fxml* file. The first few lines of the script create an instance of the class and use the *BsfCreateRexxProxy* routine to extend the **Application** class. Lastly, it invokes the *launch* method, which will create the GUI thread and invoke the *start* method [Fla18].

main.rex

```
/* Creating an instance of StageHandler */
handler = .StageHandler~new
/* Inherit from javafx.application.Application */
app = BsfCreateRexxProxy(handler,,"javafx.application.Application")
/* Launch app: Invokes start method */
app~launch(app~getClass, .nil)

:: CLASS StageHandler
:: METHOD start
```

```
USE ARG stage
stage~setTitle("Button click counter")
url = .bsf~new("java.net.URL", "file:main.fxml")
FXMLLoader = bsf.import("javafx.fxml.FXMLLoader")
fxml = FXMLLoader~load(url)
scene = .bsf~new("javafx.scene.Scene", fxml)
stage~setScene(scene)
stage~show

::REQUIRES BSF.CLS
```

4.3 Java

Java is an object oriented, strictly typed programming language currently developed by Oracle [Fla19]. It is rated as the most popular programming language by the TIOBE index [TIO19].

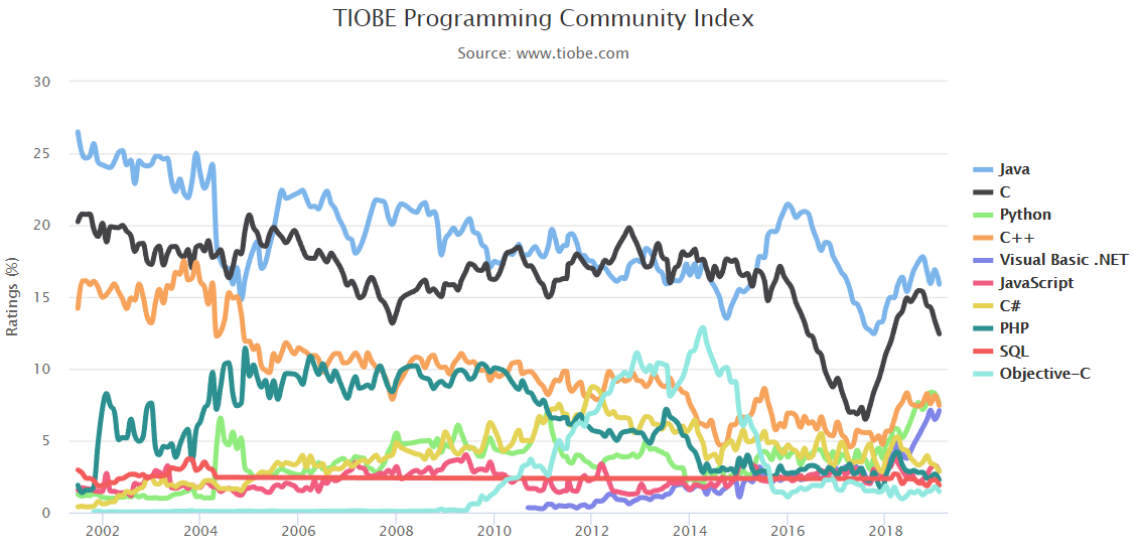


Figure 7: TIOBE Index, *Source: https://www.tiobe.com/tiobe-index/*

Java code is executed on the Java Virtual Machine (JVM). The JVM provides two important functionalities [Tys18]: It

- allows Java programs to run on any operating system and
- manages and optimizes program memory

4.4 MySQL

MySQL is a database management system currently developed by Oracle. SQL ("Structured Query Language") is the most used and well known standardized language to access databases [MyS19]. With Java it is easily possible to use SQL and because of BSF4ooRexx also with ooRexx. The full documentation of the Java *SQL package* is available in the Oracle docs [Ora18]. The following example (Listing 23) describes how to connect to the database and write statements via ooRexx.

Listing 23: Use SQL in ooRexx

```
DriverManager = bsf.import("java.sql.DriverManager")
properties = .bsf~new("java.util.Properties")
/* Set properties */
properties~put("user", "root")
properties~put("password", "password")
properties~put("serverTimezone", "CET")
properties~put("useSSL", "false")
path = "jdbc:mysql://localhost/pos_database"
SIGNAL ON SYNTAX
/* Raises exception on unsuccessful connection */
conn = DriverManager~getConnection(path,properties)
SIGNAL OFF SYNTAX

query = "SELECT * FROM product"
prepStatement = conn~prepareStatement(query)
queryResults = prepStatement~executeQuery

/* Get column count */
colCount = queryResults~getMetaData~getColumnCount
IF queryResults~next THEN DO
        SAY "======================"
        DO i=1 TO colCount
                say queryResults~getString(i)
        END
END
EXIT 0
```

SYNTAX:
SAY "Error connecting to database."
EXIT −1

:: REQUIRES BSF.CLS

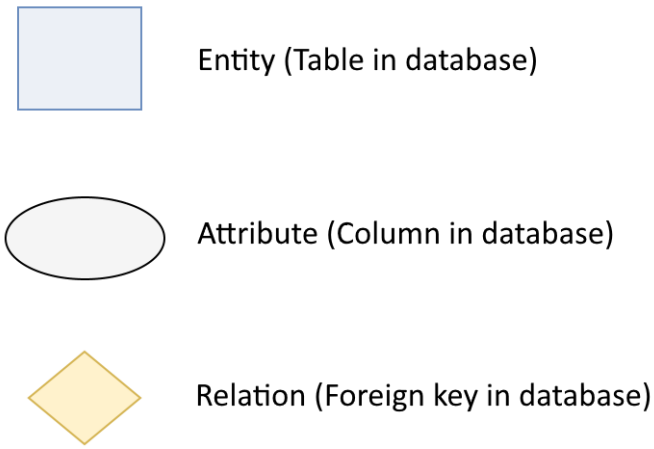
5 Relational Database

The database built with MySQL is a relational database. This means that it consists of multiple elements (tables) which are set in relation to each other. This relationship is often modeled with an Entity-Relationship (ER) model [1].

5.1 Entity-Relationship Model

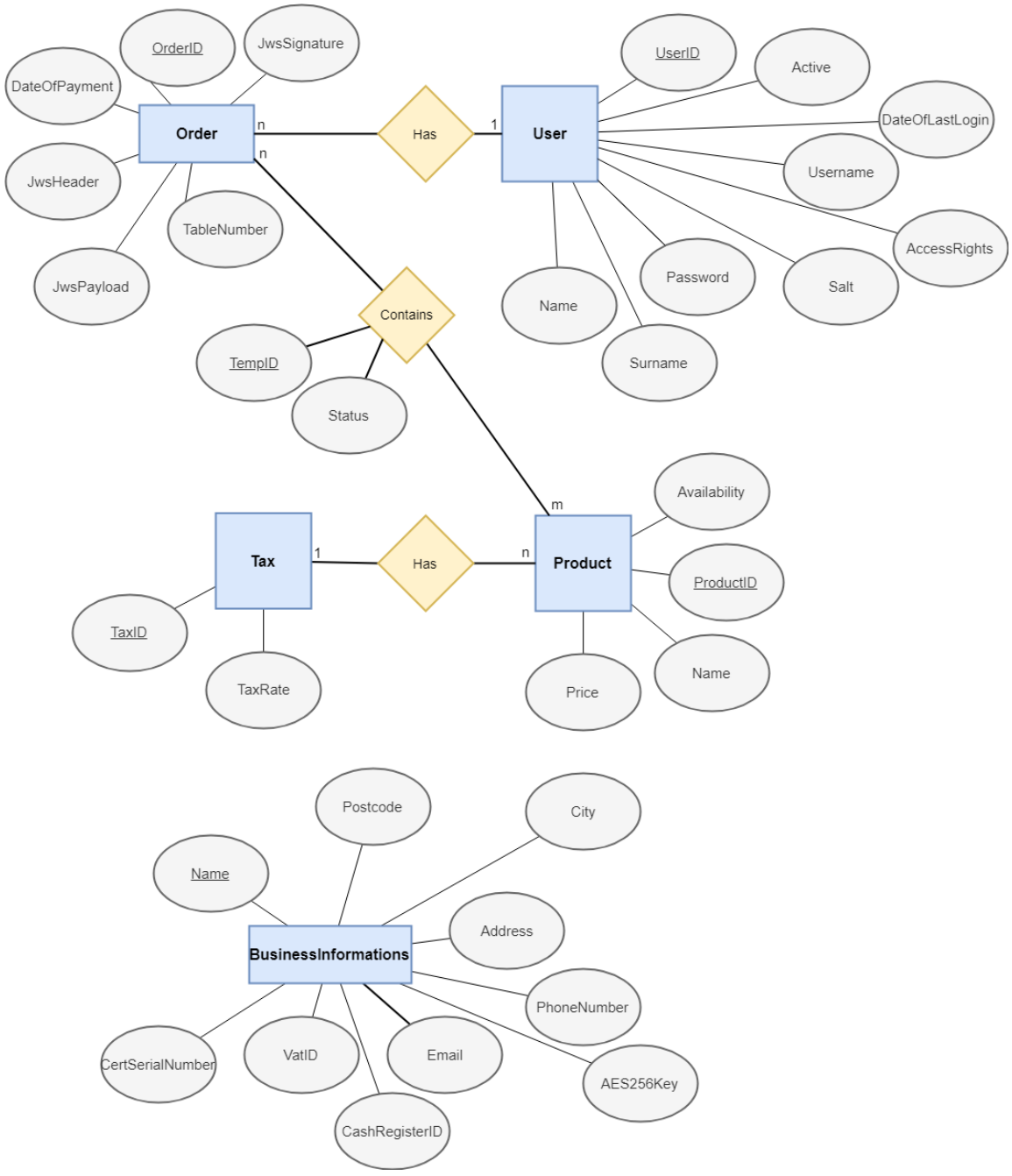
As an ER model structure matches the structure of a database it is often used as a draft for databases. It consists of entries, attributes and relations (Figure 8). These three elements are very easily transferable to tables, columns and foreign keys. Figure 9 shows the ER model that was created as a draft for the database of the developed POS software.

Figure 8: Entity Relationship Elements



[1] ER model introduction: Chaitanya Singh, `https://beginnersbook.com/2015/04/e-r-model-in-dbms/` (Visited 05.03.2019)

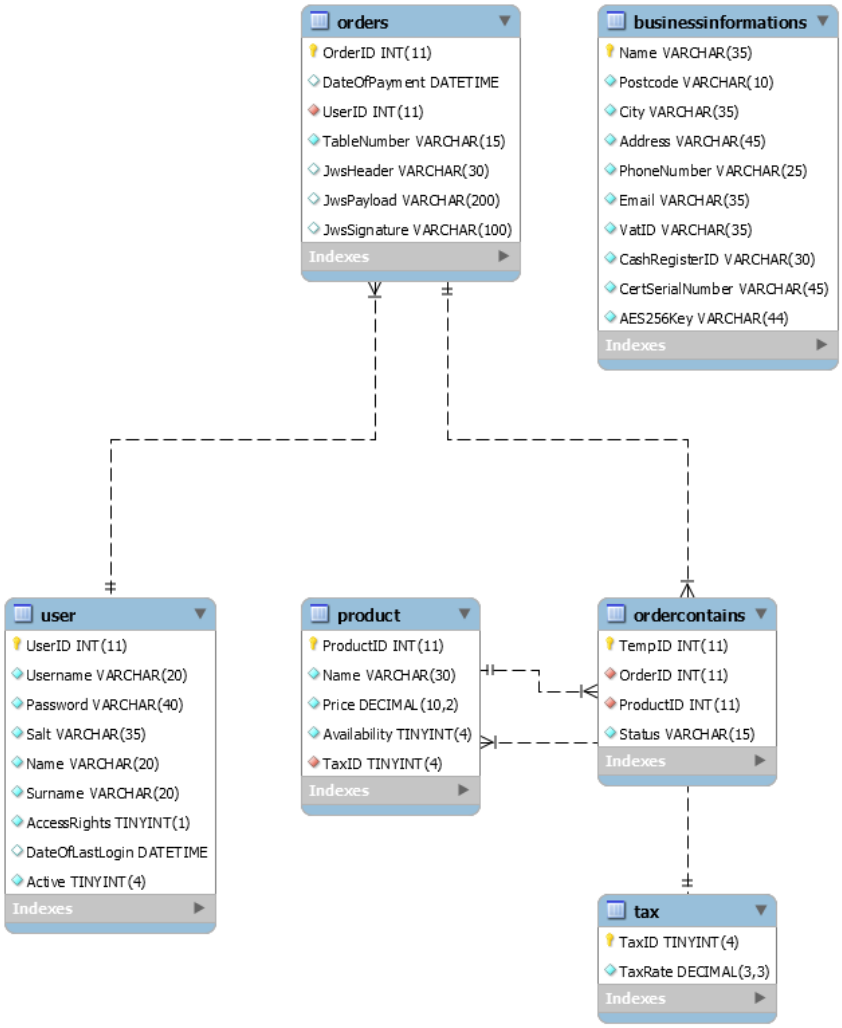
Figure 9: Entity Relationship Model



Source: Created with `www.draw.io`

After creating the database structure in MySQL it can also be exported as a diagram (Figure 10).

Figure 10: MySQL Model



5.2 Primary Key

Primary keys are used to uniquely identify a tuple (row) in a database [IBM19]. Most tables use a single column (attribute) as a primary key but it is also possible to use a combination of columns as a primary key.

5.3 Foreign Key

Foreign keys are used to create a relation between two tables. The foreign key of a table **Y** always corresponds to the primary key of another table **X** that has a relation with table **Y**.

5.4 Database Normalization

Database normalization is required to avoid anomalies. Anomalies are the result of redundancies which should be avoided to prevent inconsistencies and an unnecessary waste of memory [Wei15]. There are three different types of anomalies that can occur in a badly designed database:

- Deletion anomaly: Occurs when the deletion of database entries leads to the unintended loss of other data.
- Update anomaly: Occurs when an update of a database entry leads to inconsistencies with other entries.
- Insertion anomaly: Occurs when a tuple cannot be inserted because of missing attributes.

Bad example

Figure 11: Badly Designed Database

student	course	room
123456 John Doe	1212 Economics	AB.001
123456 John Doe	5124 Programming	B.003
100200 Jane Roe	1212 Economics	AB.001
355400 Olivia Smith	3333 Biology	C.211

Figure 11 shows an example for a badly designed database. The database stores which student is enrolled in which course and the room

where the course takes place.
A **deletion anomaly** would occur when the student *Olivia Smith* is removed from the database. This would also remove the information about course *3333 Biology* which might be unwanted.
An **update anomaly** would occur if the room of course *1212 Economics* is changed. In this case the information has to be changed in the first and third row or the database will be inconsistent.
An **insertion anomaly** would occur if a new course is added to the database. As there are no students enrolled in the course yet the student attribute would be empty and a null value has to be placed instead.

Improvements

Figure 12: Improved Database

studentID	name	surname
123456	John	Doe
100200	Jane	Roe
355400	Olivia	Smith

courseID	name	room
1212	Economics	AB.001
5124	Programming	B.003
3333	Biology	C.211

courseID	studentID
1212	123456
5124	123456
1212	100200
3333	355400

After normalizing the database (full process described in *Part 5: Relational Database Theory - Normalization, J. Sun* [Sun02]) the structure changed as displayed in figure 12. The database is now split into three relations (tables). The student ID is now split from the student name and used as primary key for the students table. Similarly, the course ID is split from the course name and used as primary key for the courses table. The third table contains all the course enrollments and uses the foreign keys course ID and student ID. These improvements remove all redundancies and prevent anomalies.

6 Program Overview

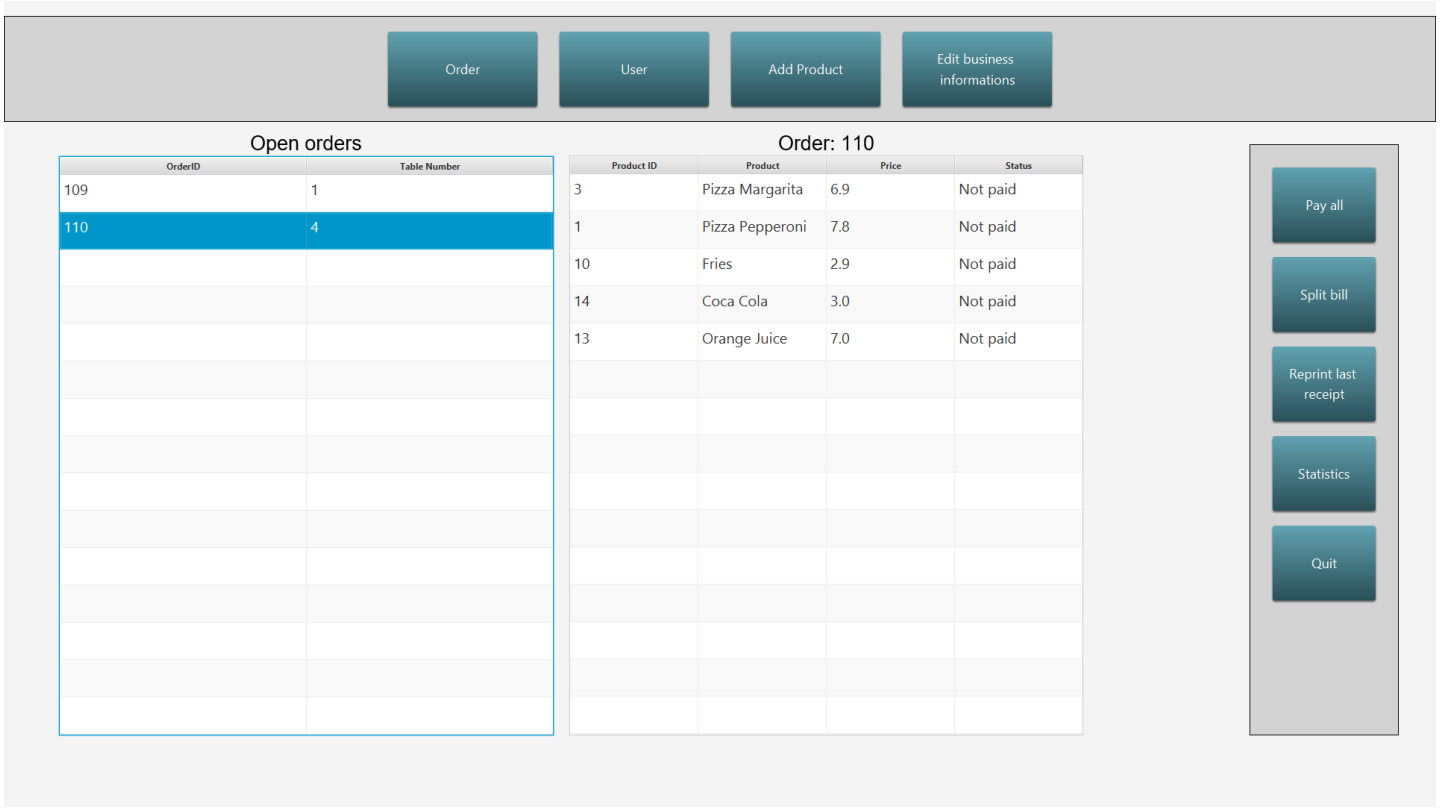
The program structure follows an object oriented approach. The following classes are defined:

- ActiveUser: Stores information about the currently logged in user
- AES256: Enables encryption and decryption of data with AES256 (Advanced Encryption Standard [BMF16]). It is used to encrypt the *sales counter* as required by the RKSV.
- DatabaseHandler: Handles access to the database. This includes all the necessary functionality like connecting, storing, updating and retrieving.
- JWSHandler: Implements JSON Web Signature (JWS) signature creation as required by the RKSV.
- PasswordHasher: Generates new random salts and hashes password with Password-Based Key Derivation Function 2 (PBKDF2) [IET11].
- QRWriter: Creates new QR Codes and stores them as PNG files.
- StageHandler: Starts the program. Loads the different FXML files.
- TableHandler classes: Multiple classes that make it possible to fill data into JavaFX table views.

6.1 Graphical User Interface (GUI)

The main goal of the GUI is a simple and fast access for touch screen users. The main page of the user interface includes two tables (Figure 13): The first table displays all open orders and the second table shows the order details of the currently marked order. The top and right side contain a menu bar with buttons.

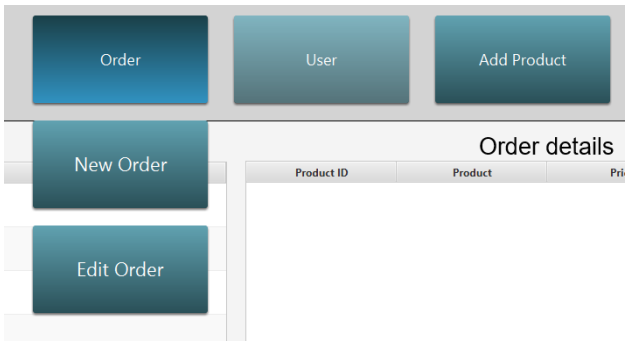
Figure 13: GUI: Main Window



6.1.1 JFoenix - Button Menu

The buttons *Order* and *User* open a small menu with additional buttons (Figure 14). This makes the main window clearer as it is not overfilled with buttons. These hidden menus are created with JFXNodesList and JFXButtons of the JFoenix library.

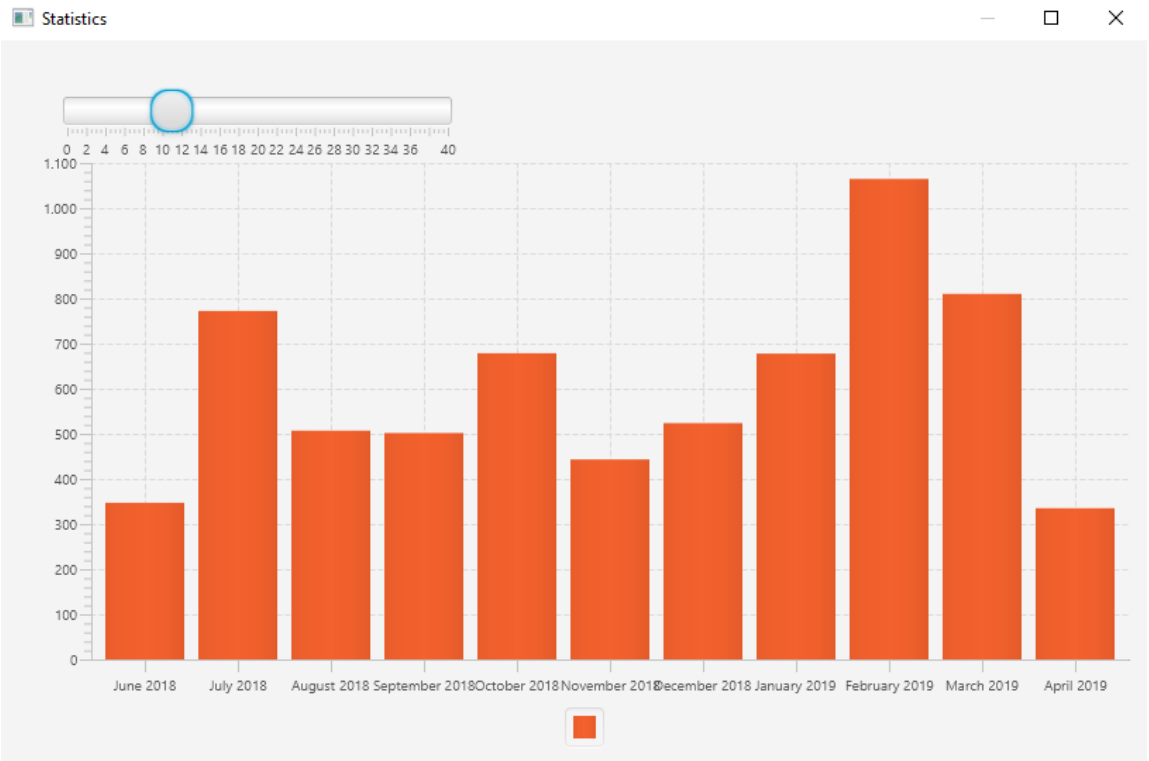
Figure 14: GUI: Hidden Buttons



6.1.2 Statistics

When clicking the *Statistics* button in the main page the sales history will be displayed in a bar chart (Figure 15). The amount of month displayed in the bar chart can be changed with a slider. The bar chart creation is possible with the FXML *StackedBarChart*. The data will be retrieved from the database with the *getMonthlyTurnover* method of the **DatabaseHandler** class. The data is then added to a *XYChartSeries* object which can then be added to the bar chart (Listing 24). To automatically change the displayed bar chart when the slider is moved a change listener is implemented. This is done with the *SliderHandler* class.

Figure 15: GUI: Bar Chart



Listing 24: Creating JavaFX Bar Chart

```
/*@get(barChart slider)*/
barChart~animated = .false
.local~monthlyTurnover = .my.app~dbh~getMonthlyTurnover
maxMonths = .my.app~dbh~getDistinctMonths

sliderHandler = .SliderHandler~new(slider, barChart, maxMonths)

::CLASS SliderHandler
::METHOD slider ATTRIBUTE
::METHOD barChart ATTRIBUTE
::METHOD INIT
  EXPOSE slider barChart XYChart XYChartData XYChartSeries
  USE ARG slider, barChart, maxMonths
   listener  = BsfCreateRexxProxy(self,,"javafx.beans.value.ChangeListener")
   slider~valueProperty~addListener(listener)
  XYChart = bsf.loadClass("javafx.scene.chart.XYChart")
  XYChartData = bsf.import("javafx.scene.chart.XYChart$Data")
```

```
  XYChartSeries = bsf.import("javafx.scene.chart.XYChart$Series")
  IF maxMonths > 12 THEN monthToDisplay = 12
  ELSE monthToDisplay = maxMonths
  slider~setMax(maxMonths)
  slider~setValue(monthToDisplay)

::METHOD changed
  EXPOSE slider barChart XYChart XYChartData XYChartSeries
  monthToDisplay = format(slider~getValue,,0)
  maxMonths = slider~getMax
  dataSeries1 = XYChartSeries~new
  DO i=maxMonths-monthToDisplay+1 TO maxMonths
    date = .monthlyTurnover['dates'][i]
    rev = box("float", .monthlyTurnover['revenues'][i])
    dataSeries1~getData~add(XYChartData~new(date, rev))
  END
  barChart~getData~clear
  barChart~getData~add(dataSeries1)

::REQUIRES "BSF.CLS"
::REQUIRES "DatabaseHandler.CLS"
```

6.2 Security

To secure the user passwords only the password hash values are stored. The creation of hashes is done with hash algorithms. These algorithms have different properties that make them very useful for securing passwords. They:

- turn an input (plain text) of any length to an output (hash) with fixed length
- are one-way functions: It is basically impossible to recreate the plain text with a given hash
- return completely different hashes if the plain text is changed slightly

Hashing algorithms used for password hashing should also have the property of collision resistance. This means that it should practically

be impossible to create the same hash with different plain texts. The example below (Listing 25) displays the properties of a hash algorithm. A routine is defined that takes plain text as an input and returns the hash of the Message Digest Algorithm 5 (MD5). MD5 is just used as an example and should not be used for password hashing as it is possible to generate the same hash with different plain texts [Sel11]. The comment in the code shows the result of hashing the strings "Hello" and "Helloo". Although the two strings differ only by one character (one "o") there are no similarities in the hashes. Furthermore, both hashes have the same length although the inputs have different lengths.

Listing 25: MD5 Algorithm

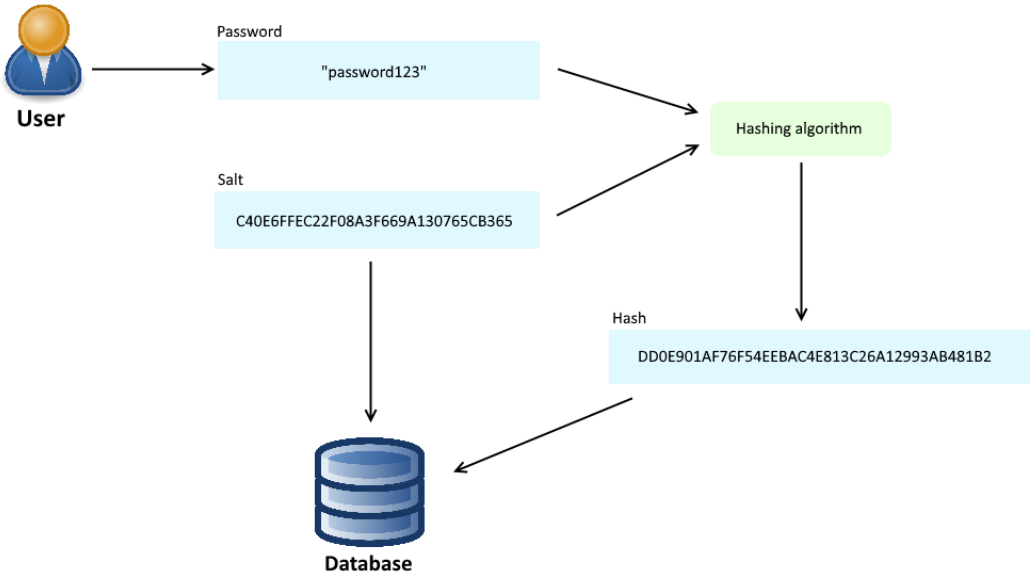
```
SAY md5Hash("Hello")
SAY md5Hash("Helloo")

/* Output:
8B1A9953C4611296A827ABF8C47804D7
55BD746519349F3F372EF409061B0FFF
*/

::ROUTINE md5Hash
        USE ARG plaintext
        MessageDigest = bsf.import("java.security.MessageDigest")
        DC = bsf.import("javax.xml.bind.DatatypeConverter")
        /* String to byte array */
        textBytes = bsfRawBytes(plaintext)
        /* Get MD5 Instance */
        md = MessageDigest~getInstance("MD5")
        /* Hash the plain text */
        hash = md~digest(textBytes)
        /* Convert byte array to hex string */
        return DC~printHexBinary(hash)

::REQUIRES BSF.CLS
```

For further account protection the POS software is also making use of salted password hashing. Salt is random data [Wik19d] that is unique to each user and is appended to the password before hashing it. This increases the protection against rainbow table and dictionary attacks [Ari18].

Account creation / login procedure

The password hashing process including the random salt generation is implemented with the class **PasswordHasher** (Reference [Mil19]). When a new account is created by a user (Figure 16) the application will first generate a new salt. This is done with the method *generateRandomSalt* (Listing 26).

Figure 16: Storing Password Hash



Listing 26: Generating Random Salt

```
:: METHOD generateRandomSalt PUBLIC
  random = .bsf~new("java.security.SecureRandom")
  salt   = bsf.createArray("byte.class", 16)
  random~nextBytes(salt)
  RETURN salt
```

The generated salt and the entered password will then be used as arguments for the *generatePasswordHash* method (Listing 27).

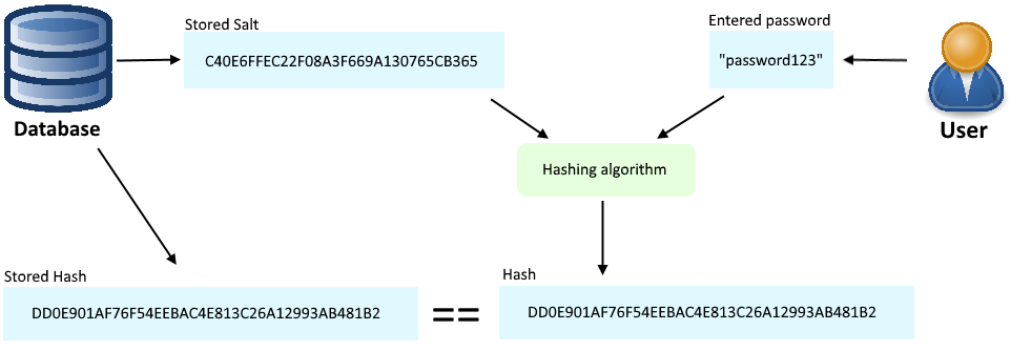
Listing 27: Hashing Password Routine

```
:: METHOD generatePasswordHash PUBLIC
  USE ARG password, salt

  SecretKeyFactory = bsf.import("javax.crypto.SecretKeyFactory")
  PBEKeySpec = bsf.import("javax.crypto.spec.PBEKeySpec")

  iterationCount = 100000
  keyLength = 160

  pwCharArr = self~stringToJavaCharArray(password)
  spec = PBEKeySpec~new(pwCharArr, salt, iterationCount, keyLength)
  factory = SecretKeyFactory~getInstance("PBKDF2WithHmacSHA1")
  hash = factory~generateSecret(spec)~getEncoded
  hashString = self~byteArrayToHexString(hash)
  RETURN hashString
```

The generated hash as well as the generated salt will then be stored in the MySQL database.

When a user logs in (Figure 17) the stored salt will be retrieved from the database and used in combination with the entered password to create a new hash. The newly created hash will then be compared to the stored hash. If the hashes match the entered password is correct and the user is able to login. Otherwise the entered password is wrong and the login fails.

Figure 17: Checking Password



The *passwordCorrect* method of the **DatabaseHandler** class is defined to handle this process (Listing 28).

Listing 28: Checking Password

```
::METHOD passwordCorrect PUBLIC
  EXPOSE conn
  USE ARG username, password
  query = "SELECT Password, Salt FROM user WHERE Username = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, username)
  queryResults = prepStatement~executeQuery
  IF \queryResults~next THEN DO
        RETURN .false
  END

  saltHex = queryResults~getString("Salt")
  passwordHashStored = queryResults~getString("Password")

  pwh = .PasswordHasher~new
  salt  = pwh~hexStringToByteArray(saltHex)
  passwordHash = pwh~generatePasswordHash(password, salt)

  RETURN passwordHash == passwordHashStored
```

6.3 Receipt

This section describes all the elements that have to be included in a receipt to comply with the RKSV. At the same time it shows how to implement the required elements in code. Next it describes how the plain text is used to create a QR-Code and lastly how the receipt is printed.

6.3.1 QR-Code

The QR-Code (Quick Response Code) is a machine-readable, two-dimensional barcode [Wik19c]. All receipts have to include a QR-Code. The RKSV § 10 (2) defines nine different elements that need to be included in the QR-Code on a receipt [WKO15]:

1) Cash register algorithm identifier
2) Till identification number
3) Sequential number of the cash transactions
4) Date and time of receipt issuing
5) Amount of cash payments separated according to tax rates
6) Encrypted status of the *sales counter* (AES 256 encrypted)
7) Serial number of the signature certificate
8) Signature value of the previous cash transaction of the data collection protocol (chain value)
9) Signature value of the respective cash transaction

The plain text of the QR code also has a defined format. The following example describes this format (Reference [BMF16]). This text is normally concatenated and is only split apart to show the different elements.
*1) _R1-AT9_*
*2) CASHREGISTER1_*

*3) 11_*
*4) 2019-02-25T23:56:38_*
*5) 2,80_7,30_0,00_0,00_0,00_*
6) TBy8E0ks0iIlxuDhLdC3pw==_
*7) 584b4e14_*
8) y1kwxzTX5fI=_
9) 9vwYiHgQlAbHWwuJVRAU8YghfPrDxeQBlZqEHys57T
YdcsAIoS3DOms725fED9C/8DIRpxH7rR0U7xwuhEtpEw==

The **first** element *R1-AT9* is the cash register algorithm identifier. R1 stands for the hashing algorithm ES256 (JSON Web Algorithm standard). Currently R1 is the only available algorithm identifier. When different algorithms are available they will be abbreviated with the identifiers R2, R3 and so on. The second part of the first element is an identifier for the Certification Service Provider (ZDA). AT1 for example stands for A-Trust and AT2 for GLOBAL TRUST.

The **second** element *CASHREGISTER1* is the till identification number (also referred to as cash register ID in the database). If a business has multiple cash registers they each have to have a unique ID.

The **third** element *11* is the sequential number of cash transactions (also referred to as order ID in the database). This number makes it possible to uniquely identify each transaction. In the MySQL database this ID is used as primary key for the orders table. It is also set to AUTO_INCREMENT so all orders automatically get sequential numbers assigned as primary key.

The **fourth** element *2019-02-25T23:56:38* is the date and time of receipt issuing separated by "T". The format is YYYY-MM-DD"T"hh:mm:ss. When a receipt is printed the *insertDateOfPayment* method is called (Listing 29). It will insert the current date and time with the MySQL *NOW()* function. After updating the date of payment a SELECT statement is used to retrieve the date and time from the database.

This date will then be returned by this method so it can be reformatted and included in the receipt.

Listing 29: Inserting the Date of Payment

```
::METHOD insertDateOfPayment PUBLIC
  EXPOSE conn
  USE ARG orderID
  query="UPDATE orders SET DateOfPayment = now() WHERE OrderID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, orderID)
  prepStatement~executeUpdate

  query = "SELECT DateOfPayment FROM orders WHERE OrderID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, orderID)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    date = queryResults~getString("DateOfPayment")
  END
  RETURN date
```

The **fifth** element *2,80_7,30_0,00_0,00_0,00* is the amount of cash payments separated according to the tax rates (20%, 10%, 13%, 0%, 19%). The format specifies that the different tax rate have to be separated by the underscore character ("_").

The **sixth** element *TBy8E0ks0iIlxuDhLdC3pw==* is the status of the *sales counter* encrypted with AES-256. As AES is a symmetric key algorithm it uses the same key for encryption and decryption. The encryption will be implemented with the **AES256** class. Two important methods of the class are shown in the code snippet below (Listing 30).

Listing 30: AES256 Class

```
/**
  * AES256 Class provides AES256 message encryption and decryption
  * Additionally, it is able to create new AES256 keys and
  * checksums to verify their integrity
*/
```

```
:: CLASS AES256 PUBLIC
:: METHOD KeyGenerator ATTRIBUTE
:: METHOD Base64 ATTRIBUTE
:: METHOD SecretKeySpec ATTRIBUTE
:: METHOD Cipher ATTRIBUTE
:: METHOD INIT
  EXPOSE KeyGenerator Base64 SecretKeySpec Cipher
  KeyGenerator = bsf.import("javax.crypto.KeyGenerator")
  Base64 = bsf.import("java.util.Base64")
  SecretKeySpec = bsf.import("javax.crypto.spec.SecretKeySpec")
  Cipher = bsf.import("javax.crypto.Cipher")

:: METHOD generateAES256Key PUBLIC
  EXPOSE KeyGenerator Base64

  keyGen = KeyGenerator~getInstance("AES")
  keyGen~bsf.invoke("init", 256)
  secretKey = keyGen~generateKey
  encodedSecKey = secretKey~getEncoded
  Encoder = Base64~Encoder
  base64Key = Encoder~encodeToString(encodedSecKey)
  RETURN base64Key

:: METHOD encrypt PUBLIC
  EXPOSE Base64 SecretKeySpec Cipher
  USE ARG message, secretKey

  Decoder = Base64~Decoder
  encodedSecKey = Decoder~decode(secretKey)
  secKeySpec = SecretKeySpec~new(encodedSecKey, "AES")
  c = Cipher~getInstance("AES")
  c~bsf.invoke("init", Cipher~ENCRYPT_MODE, secKeySpec)
  encrypted = c~doFinal(BsfRawBytes(message))
  Encoder = Base64~Encoder
  encoded = Encoder~encodeToString(encrypted)
  RETURN encoded
```

When a receipt is printed an instance of the **AES256** class is created which loads all the necessary Java classes. Additionally, the *getTurnover* method is called to retrieve the status of the *sales counter*. The *sales counter* is then converted from € (euro) to ¢ (eurocent) (Listing 31, Line 4) and used as an argument for the *encrypt* method (Listing 31).

Listing 31: Encrypting Sales Counter

```
/* Encrypt turnover */
aes = .AES256~new
turnover = .my.app~dbh~getTurnover
turnover = format(turnover*100,,0)
SIGNAL ON ANY NAME encryptionError
turnoverEncoded = aes~encrypt(turnover, receipt["AES256Key"])
SIGNAL OFF ANY
EXIT 0

encryptionError:
  .my.app~stageHandler~showDialog("ERROR", "Encryption error", -
  "Invalid AES key. Use generateKeys.rex script to generate a new key.")
```

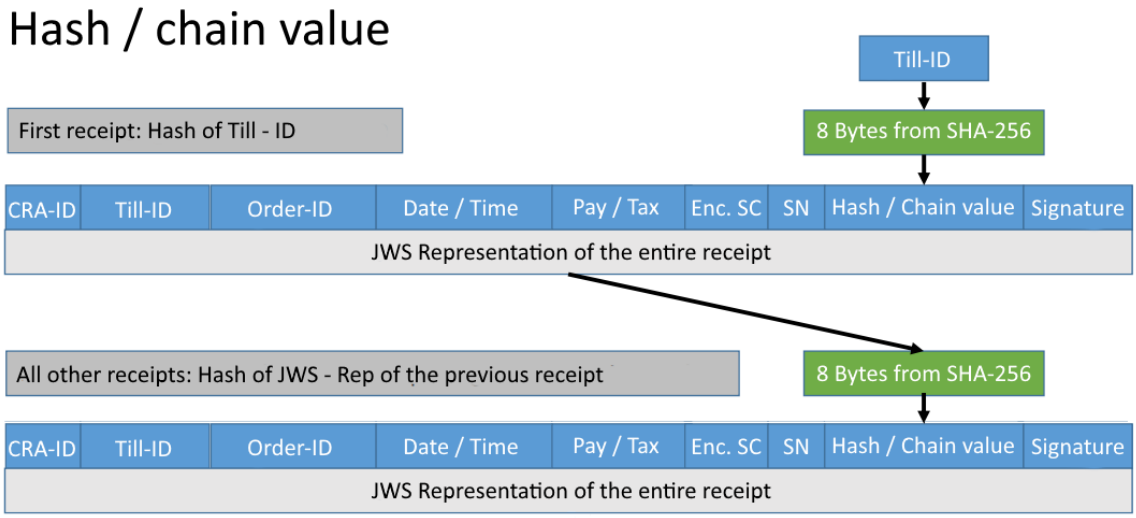
The **seventh** element *584b4e14* is the serial number of the signature certificate in hexadecimal representation. As the serial number is entered in decimal representation it first needs to be converted to hexadecimal before it is added to the QR code. This can easily be done with the ooRexx built in function *D2X()*. As the default numeric precision in ooRexx is set to 9 and the serial number has more than 9 digits it is necessary to increase the numeric precision first. This is done with *NUMERIC DIGITS 20* (Listing 32). After converting the serial number to hexadecimal the numeric precision is set back to the default 9 with *NUMERIC DIGITS*.

Listing 32: Increase Numeric Precision

```
NUMERIC DIGITS 20
hexSerialNr = D2X(receipt["CertSerialNumber"])
NUMERIC DIGITS
```

The **eighth** element *y1kwxzTX5fI=* is the hash of the JWS representation of the previous receipt. The used hashing algorithm is Secure Hash Algorithm 256 (SHA-256). This process is described in figure 18. If there are no previous receipts than the till identification is hashed instead. Additionally, it is specified that only the first 8 bytes of the hash are used. This process is implemented in the *hashLastReceipt* method (Listing 33).

Figure 18: Creating Hash / Chain Value



Source: Posch R. (2016). Elektronische Signatur: Sicherheitsanforderungen und Nachweise [Pos16]

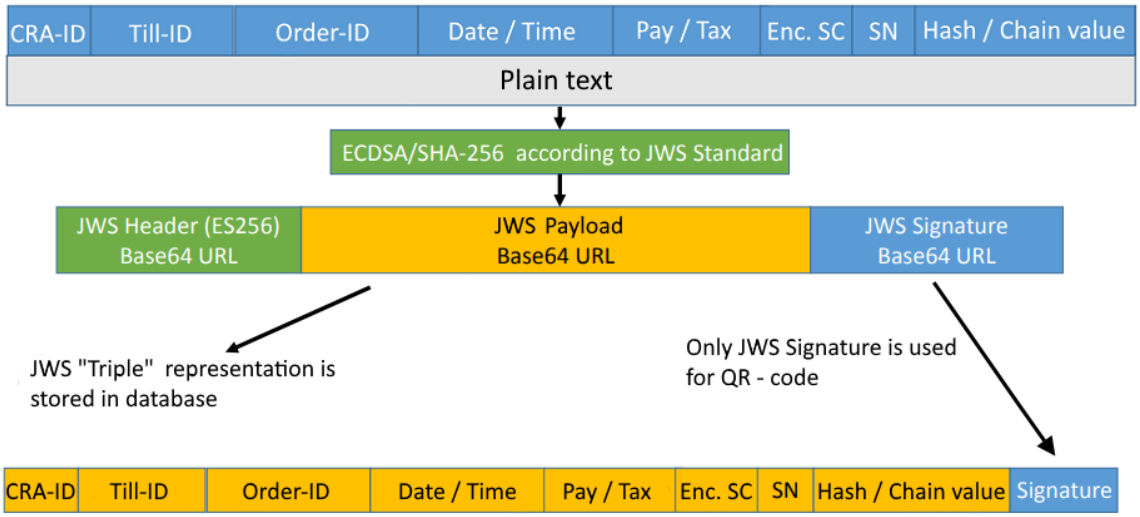
Listing 33: Hashing Receipt

```
::METHOD hashLastReceipt PUBLIC
  EXPOSE MessageDigest StandardCharsets System
  USE ARG str, N=8
  jStr = box("STring", str)
```

```
digest = MessageDigest~getInstance("SHA-256")
hash = digest~digest(jStr~getBytes(StandardCharsets~UTF_8))

/* Copy N Bytes */
conDigest = bsf.CreateJavaArray("byte.class", N)
System~arraycopy(hash, 0, conDigest, 0, N)

RETURN self~base64Encode(conDigest)
```

The **ninth** and last element *9vwYiHgQlAbHWwuJVRAU8YghfPrDxeQB lZqEHys57TYdcsAIoS3DOms725fED9C/8DIRpxH7rR0U7xwuhEtpEw==* is the last part of the JWS triple (Figure 19). The JWS triple is created using the *encryptECDSA* method (Listing 34).

Figure 19: Creating JWS "Triple"



Source: Posch R. (2016). Elektronische Signatur: Sicherheitsanforderungen und Nachweise [Pos16]

Listing 34: Elliptic Curve Digital Signature Algorithm (ECDSA) Encryption

```
::METHOD encryptECDSA PUBLIC
  USE ARG payload, privateKey

  JsonWebSignature = bsf.import("org.jose4j.jws.JsonWebSignature")
  jws = JsonWebSignature~new
  jws~setKey(privateKey)
  jws~setPayload(payload)
  jws~setAlgorithmHeaderValue("ES256")
  jws~sign
  RETURN jws~getCompactSerialization
```

It takes the first eight elements of the QR plain text and the private key as arguments and returns the JWS triple. After parsing the triple the signature (last part of triple) is added to the QR plain text (Listing 35). The triple is then stored in the database.

Listing 35: Parsing JWS Triple

```
jws = jwsHandler~encryptECDSA(qrText~string, privateKey)
PARSE VAR jws header "." payload "." signature .
signatureB64 = jwsHandler~recode(signature)
qrBuffer = qrText~append("_" || signatureB64)
qrString = qrBuffer~string
.my.app~dbh~storeJWS(id, header, payload, signature)
```

The actual creation of the QR-Code is abstracted in a class which loads all the necessary Java classes when a new instance is created (Listing 36). It has one method *write* that has two parameters, the path for the QR-Code and the plain text.

Listing 36: QR-Code Writer Class (Reference: [Sin17])

```
::CLASS QRWriter PUBLIC
::METHOD BarcodeFormat ATTRIBUTE
::METHOD BitMatrix ATTRIBUTE
::METHOD QRCodeWriter ATTRIBUTE
::METHOD MTIWriter ATTRIBUTE
```

```
::METHOD INIT
  EXPOSE BarcodeFormat BitMatrix QRCodeWriter MTIWriter

  BarcodeFormat = bsf.import("com.google.zxing.BarcodeFormat")
  BitMatrix = bsf.import("com.google.zxing.common.BitMatrix")
  QRCodeWriter = bsf.import("com.google.zxing.qrcode.QRCodeWriter")
  MTIWriter = bsf.import("com.google.zxing.client.j2se.MatrixToImageWriter")

::METHOD write
  EXPOSE BarcodeFormat BitMatrix QRCodeWriter MTIWriter
  USE ARG path, qrString

  path = .bsf~new("java.io.File", path)

  writer = QRCodeWriter~new
  bitMatrix = writer~encode(qrString, BarcodeFormat~QR_CODE, 300, 300)
  MTIWriter~writeToFile(bitMatrix, "PNG", path)

::REQUIRES BSF.CLS
```

6.3.2 Printing

To print the receipts the application uses the Java Swing classes **JEditorPane** and **HTMLEditorKit** (Listing 37). With these classes it is possible to print an Hypertext Markup Language (HTML) document. Additionally, Cascading Style Sheets (CSS) can be used to change the style of the document. The printing is done by a routine called *print* that expects an HTML string as an argument. It returns true if the printing was successful or false if there was an error or the printing job was canceled by the user.

Listing 37: Printing HTML Document

```
::ROUTINE print
  USE ARG string

  jEditorPane = .bsf~new("javax.swing.JEditorPane")
  kit = .bsf~new("javax.swing.text.html.HTMLEditorKit")
```

```
/* Add css to HTML */
jEditorPane~setEditorKit(kit)

/* Define stylesheet */
styleSheet = kit~getStyleSheet
styleSheet~addRule("body {font-size: 7px;}")
styleSheet~addRule("h1 {font-size: 13px;}")
styleSheet~addRule("h2 {font-size: 10px;}")

doc = kit~createDefaultDocument
jEditorPane~setDocument(doc)
jEditorPane~setText(string)
RETURN jEditorPane~print
```

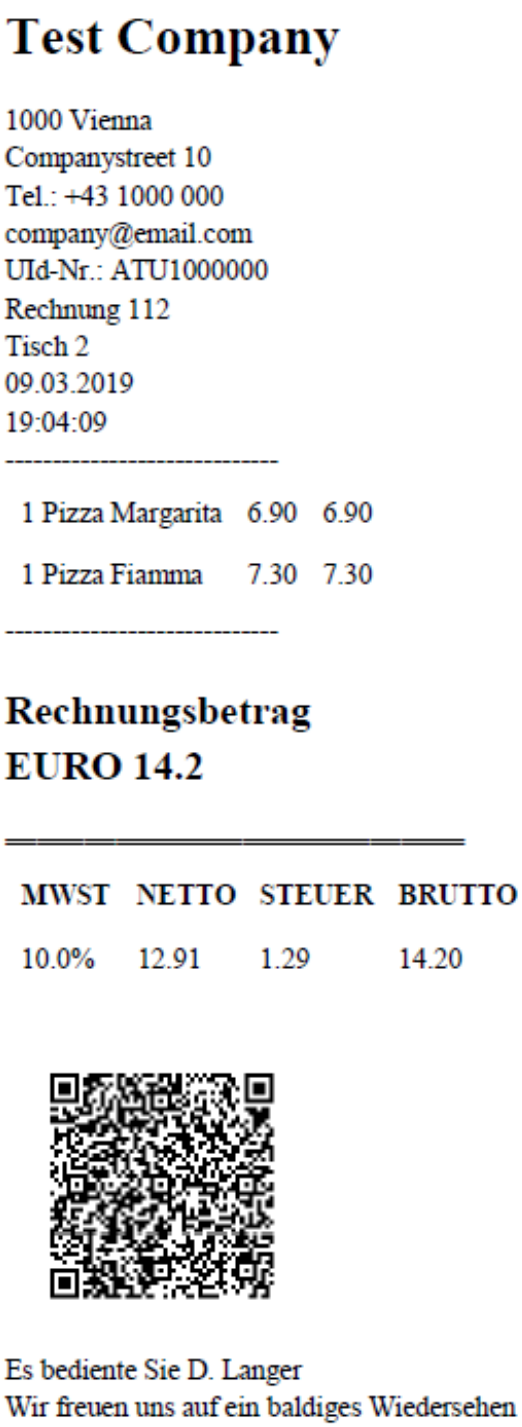
As the HTML string needs to change dynamically depending on the order it is defined in code and not as a separate file. The HTML string is a concatenation of multiple HTML tags. To create this string more efficiently the ooRexx class **MutableBuffer** is used (Listing 38). In the ooRexx documentation it is explained that *MutableBuffers*, unlike String objects, can be altered without requiring a new object allocation [ooR09a]. Figure 20 shows an example of a printed receipt.

Listing 38: Creating HTML String Using Mutable Buffer

```
text = .mutableBuffer~new
text~append("<!DOCTYPE html>" -
      "<html>" -
        "<body>" -
          "<h1>" || receipt['Name'] || "</h1>" -
            "<span>" || receipt['Postcode'] receipt['City'] || "<br>" -
             || receipt["Address"] || "<br>" -
              "Tel.: " || receipt["PhoneNumber"] || "<br>" -
              receipt["Email"] || "<br>" -
              "UId-Nr.: " || receipt["VatID"] || "<br>" -
              "Rechnung" receipt['OrderID'] || "<br>" -
              || "Tisch" receipt['TableNumber'] || "<br>" -
              || date || "<br>" -
```

```
                || time || "<br>" -
            "--------------------------------<br>" -
            "<table>")

DO item OVER receipt['Items']
  text~append("<tr>")
  DO i OVER item
    text~append("<td>" || i || "</td>")
  END
  text~append("</tr>")
END
text~append("</table>")
text~append("--------------------------------<br>" -
      "<h2>Rechnungsbetrag <br>EURO" receipt['Sum'] || " </h2>" -
      "==================================<br>" -
      "<table>" -
      "<tr>" -
        "<th>MWST</th>"-
        "<th>NETTO</th>"-
        "<th>STEUER</th>"-
        "<th>BRUTTO</th>"-
      "</tr>")

text = appendSums(text, receipt['SumTable'])

filePath = "file:" || directory() || "\res\qr.png"

text~append("</table><br>"-
      '<img src="' || filePath || '" width="100" height="100"><br>' -
      "Es bediente Sie " || left(receipt['UserName'], 1) || "." -
      receipt['UserSurname'] || "<br>" -
      "Wir freuen uns auf ein baldiges Wiedersehen <br>"-
      "</span></body></html>")
```

Figure 20: Receipt Example



Test Company

1000 Vienna
Companystreet 10
Tel.: +43 1000 000
company@email.com
UId-Nr.: ATU1000000
Rechnung 112
Tisch 2
09.03.2019
19:04:09

---

1 Pizza Margarita	6.90	6.90
1 Pizza Fiamma	7.30	7.30

Rechnungsbetrag
EURO 14.2

==========

MWST	NETTO	STEUER	BRUTTO
10.0%	12.91	1.29	14.20

Es bediente Sie D. Langer
Wir freuen uns auf ein baldiges Wiedersehen

7 Running the Program

This section describes how to run the program for the first time. This assumes that all the necessary tools are already installed correctly and the MySQL server is running.

7.1 Setup

After installing all the necessary tools described in Section 3 and including all JARs either in the *CLASSPATH* or in the /bin/ folder of the program the first step to run the program is to execute the Rexx script setup.rex (Figure 21). The script reads the database settings from the configuration file *conf.ini* (Listing 39). If the file does not exist before running the setup script it will be created with the default settings. The configuration file also includes the MySQL username and password which are necessary to connect to the database. This is set to **root** and **password** by default. This can be changed either in the *conf.ini* file manually or by running the setup.rex script with two arguments, first the *username* and second the *password*. Once the *setup.rex* script is executed it will check if all necessary JARs are available, create the configuration file if it does not exist and create the database with all necessary tables.

Listing 39: Configuration File: "conf.ini"

```
[DATABASE SETTINGS]
DB_NAME = pos_database
DB_USER = root
DB_PASSWORD = password
DB_URL = jdbc:mysql://localhost/
DB_TIMEZONE = CET
DB_USE_SLL = false
```

After the file is created it will be used in other scripts when connecting to the database. This is why it is important that all the settings are correct or the connection to the database will fail.

Figure 21: Running "setup.rex" with username and password

```
D:\POSRexx>rexx setup.rex root password
[+] SUCCESS: mysql-connector-java found.
[+] SUCCESS: core-3 found.
[+] SUCCESS: javase-3 found.
[+] SUCCESS: slf4j-api found.
[+] SUCCESS: slf4j-simple found.
[+] SUCCESS: jfoenix found.
[+] SUCCESS: jose4j found.
[+] SUCCESS: Java JDK 1.8 found.
[+] SUCCESS: Java version 1.8 installed correctly.
[+] SUCCESS: conf.ini created.
[+] Connecting to database...
[+] SUCCESS: Connected to database.
[+] Checking if schema exists...
[+] Schema pos_database does not exist.
[+] Creating schema...
[+] SUCCESS: Created schema pos_database
[+] Creating table tax...
[+] SUCCESS: Table tax created.
[+] Creating table user...
[+] SUCCESS: Table user created.
[+] Creating table product...
[+] SUCCESS: Table product created.
[+] Creating table orders...
[+] SUCCESS: Table orders created.
[+] Creating table orderContains...
[+] SUCCESS: Table orderContains created.
[+] Creating table businessInformation...
[+] SUCCESS: Table businessInformation created.
================================================
[+] Setup completed. 0 warning(s) found.
Press enter to quit.
```

7.2 Filling Database with Random Data (Optional)

For testing purposes the database can be automatically filled with data. The script fillDatabase.rex inserts different products and a certain amount of random orders (Figure 22). The amount of random orders will either correspond to the first command line argument or default to 500 when no argument is given.

Figure 22: Running "fillDatabase.rex"

```
D:\POSRexx>rexx fillDatabase.rex 5
Connection Successful.
Adding new user...
Getting inserted ID...
Adding products...
Adding random order number 1.
Adding random order number 2.
Adding random order number 3.
Adding random order number 4.
Adding random order number 5.
Press enter to quit.
```

7.3 Generating AES256 Key + Key Pair

The Rexx script generateKeys.rex generates a key pair (used for JWS signature) and a new AES256 key with the corresponding hash value and writes it in a txt file. When first running the main program the user is asked to enter these generated keys.
The AES key is also necessary when officially registering the cash register at https://finanzonline.bmf.gv.at/. If the program is only run for testing purposes it is also possible to use the *Fill with default* button to fill the company information with default values.

7.4 Running the Main Program

The main program is executed with the *main.rex* script. When running the program for the first time the user is asked to add the business information and to create a new user account. The first created user account is automatically a new root user with full access. Once the user logs into his newly created account the setup is completed and the program can be used to its full extent. The *root* user is then able to create new accounts with different access rights.

8 Summary and Outlook

This paper presented the development of a *Point of Sale* software in ooRexx. The simple syntax makes it easy to learn the language fast and create programs quickly. It also provides full object orientation which allows to write better structured code. After learning ooRexx for just one semester without prior knowledge it was possible to create this POS software.
Additionally, the work made use of BSF4ooRexx which allowed the access to the full Java functionality. This was especially useful for the creation of the graphical user interface with JavaFX. It also provided access to MySQL which was used to store all the relevant data.

This work also gave a quick overview of designing a database without redundancies and anomalies. Before the database was implemented an ER model was created to give an overview of the required database structure. This was helpful to find and correct problems in the structure early on because models are much easier to modify than an already implemented database.

As the full source code of the developed POS software is publicly available all users are free to further improve and develop the software or use it as a reference for their own software. While the coding itself was made relatively easy by ooRexx the fulfilling of requirements imposed by the RKSV turned out to be more complex.
This work implemented many of the RKSV requirements but the handling of training and cancellation bookings is still missing which is required to fully comply with the RKSV.
Possible further improvements could be made to support different screen resolutions. The graphical user interface developed in this work focuses on 1920x1080 displays and might not display properly on different resolutions.

9 References

[Ari18] Dan Arias. Adding Salt to Hashing: A Better Way To Store Passwords. https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/, 2018. [Visited 20.02.2019].

[BMF16] BMF. Festlegung des BMF zu Detailfragen der Registrierkassensicherheitsverordnung (RKSV). https://www.bmf.gv.at/egovernment/projekte/registrierkassen/20160905-Detailfragen-RKSV-V1.2.pdf?67rvn3, 2016. [German, Visited 22.02.2019].

[BMF19] BMF. Cash Register and Receipt Issuing Obligation. https://english.bmf.gv.at/taxation/Cash_Register_and_Receipt_Issuing_Obligation.html, 2019. [Visited 11.02.2019].

[Cam19] Brian Campbell. Jose4J JAR Download. https://bitbucket.org/b_c/jose4j/wiki/Home, 2019. [Visited 16.02.2019].

[Com19] Creative Commons. CC BY-SA 3.0. https://creativecommons.org/licenses/by-sa/3.0/, 2019.

[Fla04] Rony G. Flatscher. "CAMOUFLAGING JAVA AS OBJECT REXX" in "The 2004 International Rexx Symposium, Böblingen, Germany, May 3rd - May 6th. 2004.

[Fla17] Rony G. Flatscher. Business programming 1 course slides. http://wi.wu.ac.at:8002/rgf/wu/lehre/autowin/material/foils/, 2017.

[Fla18] Rony G. Flatscher. Anatomy of a GUI (Graphical User Interface), "The 2018 International Rexx Symposium", Aruba, Dutch West Indies, March 25th 29th. 2018.

[Fla19] Rony G. Flatscher. Business programming 2 course slides. http://wi.wu.ac.at:8002/rgf/wu/lehre/autojava/material/foils/, 2019.

[IBM14a] IBM. Features of rexx. https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ikjb300/ikjb30032.htm, 2014. [Visited 12.02.2019].

[IBM14b] IBM. Rexx programming language. https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ikjb300/xrexx.htm, 2014. [Visited 12.02.2019].

[IBM19] IBM. Primary and Foreign Keys. https://www.ibm.com/support/knowledgecenter/en/SS9UM9_9.1.1/com.ibm.datatools.dimensional.ui.doc/topics/c_dm_primary-foreignkeys.html, 2019. [Visited 18.02.2019].

[IET11] IETF. RFC 6070, Password-Based Key Derivation Function 2. https://www.ietf.org/rfc/rfc6070.txt, 2011. [Visited 28.02.2019].

[Mil19] Sam Millington. Hashing a password in java. https://www.baeldung.com/java-password-hashing, 2019. [Visited 20.02.2019].

[MyS19] MySQL. MySQL documentation. https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, 2019. [Visited 10.02.2019].

[ooR09a] ooRexx. ooRexx Reference. http://www.oorexx.org/docs/rexxref/book1.htm, 2009. [Visited 12.02.2019].

[ooR09b] ooRexx. ooRexx Reference, Object Oriented Programming. http://www.oorexx.org/docs/rexxref/c188.htm, 2009. [Visited 12.02.2019].

[ooR09c] ooRexx. ooRexx Reference, Signal. http://www.oorexx.org/docs/rexxref/x4407.htm, 2009. [Visited 12.02.2019].

[Ora18] Oracle. Java Documentation, SQL Package. https://docs.oracle.com/javase/8/docs/api/index.html?java/sql/package-summary.html, 2018. [Visited 11.02.2019].

[Ora19] Oracle. "Restaurant Point of Sale" heißt jetzt "Restaurant Point of Service". https://de.wikipedia.org/wiki/Point-of-Service-System, 2019. [German, Visited 21.01.2019].

[Pos16] Reinhard Posch. Elektronische Signatur: Sicherheitsanforderungen und Nachweise. https://wien.gerichts-sv.at/fileadmin/user_upload/Aktuelles/Elektronische_Signatur_Sicherheitsanforderungen_und_Nachweise.pdf, 2016. [German, Visited 28.02.2019].

[Sel11] Peter Selinger. MD5 Collision Demo. https://www.mathstat.dal.ca/~selinger/md5collision/, 2011. [Visited 17.02.2019].

[Sin17] Rajeev Kumar Singh. Generate QR Code in Java using Zxing. https://www.callicoder.com/generate-qr-code-in-java-using-zxing/, 2017. [Visited 18.02.2019].

[Sou19] Sourceforge. BSF4ooRexx Download. https://sourceforge.net/projects/bsf4oorexx/, 2019. [Visited 10.02.2019].

[Sun02] Junping Sun. Part 5: Relational Database Theory - Normalization. http://scis.nova.edu/~jps/teaching/phdiss/diss02s/diss750/notes/diss02-5.pdf, 2002.

[TIO19] TIOBE. "TIOBE index". https://www.tiobe.com/tiobe-index/, 2019. [Visited 11.02.2019].

[Tys18] Matthew Tyson. "What is the JVM? Introducing the Java virtual machine". https://www.javaworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html, 2018. [Visited 14.02.2019].

[Wei15] Albert Weichselbraun. DBMS Optimierung und Evaluierung des Tabellendesigns. https://weichselbraun.net/dbs/p/slides/04-Normalisierung.pdf, 2015. [German, Visited 28.02.2019].

[Wik19a] Wikipedia. Point of Sale. https://en.wikipedia.org/wiki/Point_of_sale, 2019. [Visited 21.01.2019].

[Wik19b] Wikipedia. Point of Service. https://de.wikipedia.org/wiki/Point-of-Service-System, 2019. [German, Visited 21.01.2019].

[Wik19c] Wikipedia. QR-Code. https://en.wikipedia.org/wiki/QR_code, 2019. [Visited 16.02.2019].

[Wik19d] Wikipedia. Salt (cryptography). https://en.wikipedia.org/wiki/Salt_(cryptography), 2019. [Visited 20.02.2019].

[WKO15] WKO. Federal law gazette for the republic of austria, cash register security regulation. https://www.wko.at/branchen/information-consulting/unternehmensberatung-buchhaltung-informationstechnologie/it-dienstleistung/BGBLA-2015-II-410-EN.PDF, 2015. [Visited 16.02.2019].

[WKO16] WKO. Registrierkassensicherheitsverordnung (rksv) veröffentlicht. https://www.wko.at/branchen/gewerbe-handwerk/Registrierkassensicherheitsverordnung-(RKSV)-veroeffentli.html, 2016. [German, Visited 21.01.2019].

10 License: Apache License Version 2.0

The source code of this work is licensed under Apache License Version 2.0. The full license can be found online at `https://www.apache.org/licenses/LICENSE-2.0`.

Listing 40: License

```
Copyright 2019 Daniel Langer

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
    or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

11 Appendix

The appendix explains how to find the right bit version and how to edit the *Path* and *CLASSPATH* environment variable. It also contains the full source code and a flow diagram to make the working of the main program clearer.

11.1 32 or 64 Bit - Based PC (Windows)

There are multiple ways to find out in Windows if your operating system is running on 64 bit or 32 bit. One possible way is to open any folder and find "This PC" shown in the image below (Figure 23). Right click the text and select properties. This opens a window that displays if the computer is running with 64 bit or 32 bit (Figure 24).

Figure 23: Properties

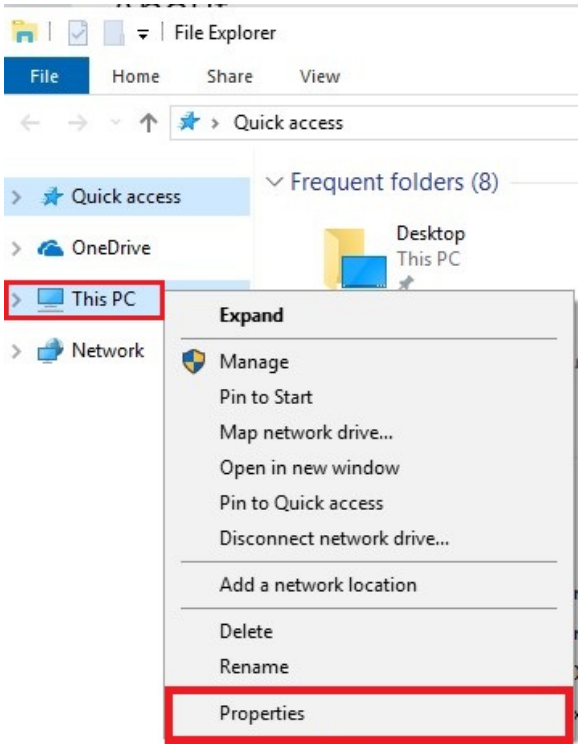
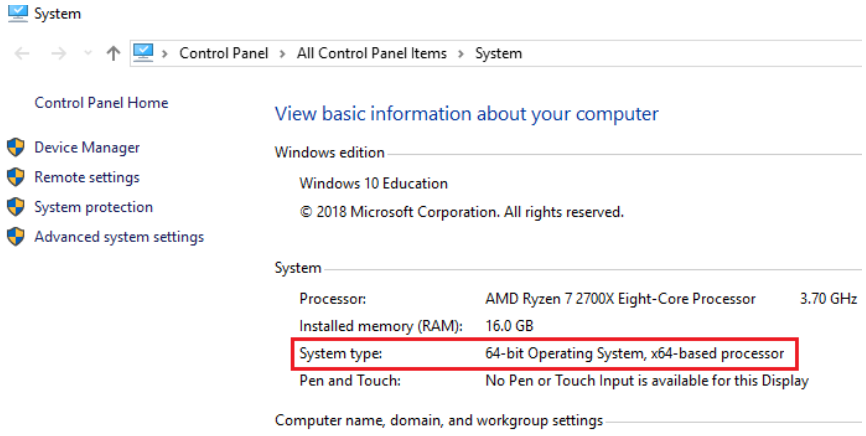


Figure 24: System Information



11.2 Adding Environment Variables

In the same way as in section 11.1 open the system window. Click on *Advanced System settings* to open the system properties (Figure 25). After that click on *Environment Variables* (Figure 26).

Path (for Java JDK) (Windows)

The *User variables* section should include the *Path* variable. To modify the existing *Path* variable press *Edit...* or press *New...* if it does not exist yet. To add the Java JDK it is necessary to include the full path to the *bin* folder that is located inside the jdk folder.

Classpath (for JARs) (Windows)

In the section *System variables* you should find a variable named *CLASSPATH*. If it does not exist it can be added with the button *New...*. Press edit to modify the existing *CLASSPATH* variable (Figure 27). When adding new JARs it is important to add the full path including the file name extension ".jar" (Figure 28).

Figure 25: Adding Environment Variables

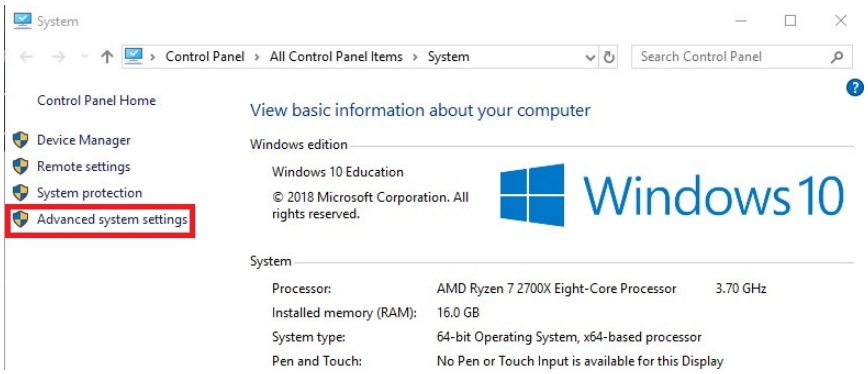


Figure 26: System Properties

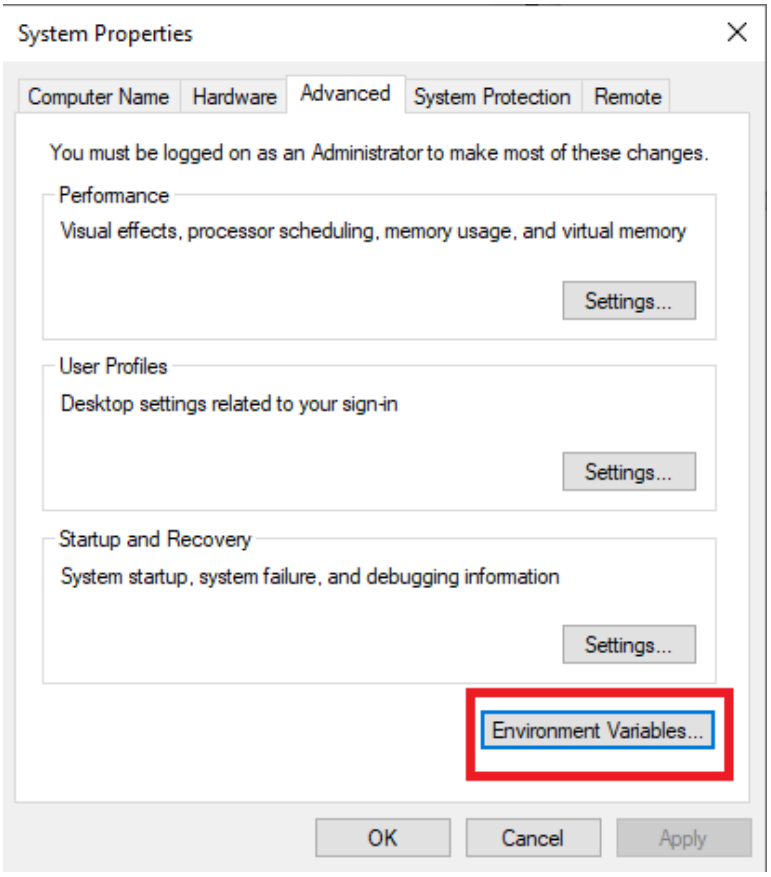


Figure 27: Editing Classpath

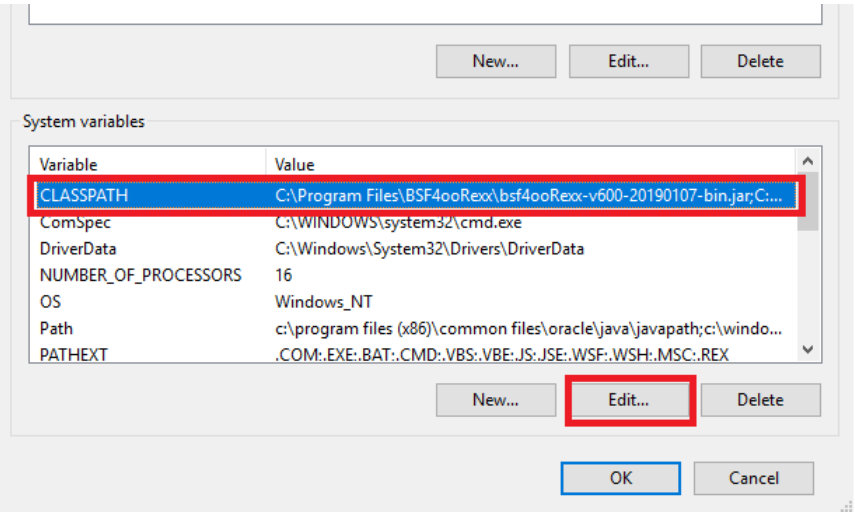
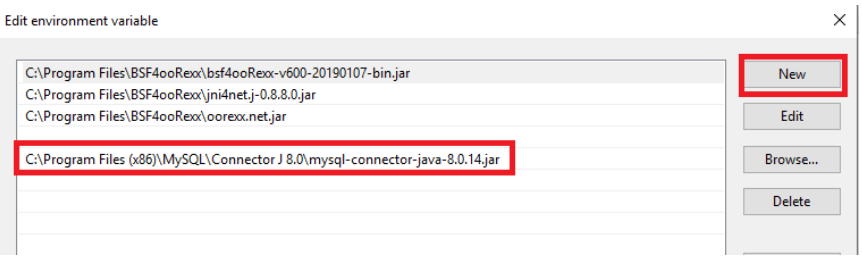


Figure 28: Adding New Path to Classpath



Path & Classpath (Linux)

To permanently add something to the *JAVA_HOME* or *CLASSPATH* environment variable in Linux the *".bashrc"* file needs to be edited. To open and edit this file run the following command: *nano ~/.bashrc* (Figure 29). This will open the *".bashrc"* file. Add the following line *export CLASSPATH="${CLASSPATH}:* and the full path to each JAR separated by "**:**" (Figure 30). The full statement could look like the following (for readability the elements are separated in newlines):
export CLASSPATH="${CLASSPATH}:
/home/daniel/Desktop/POSRexx/bin/mysql-connector-java-8.0.14.jar:
/home/daniel/Desktop/POSRexx/bin/core-3.3.3.jar:
/home/daniel/Desktop/POSRexx/bin/javase-3.3.3.jar:
/home/daniel/Desktop/POSRexx/bin/jfoenix-8.0.8.jar:

/home/daniel/Desktop/POSRexx/bin/jose4j-0.6.5.jar:
/home/daniel/Desktop/POSRexx/bin/slf4j-api-1.7.25.jar:
/home/daniel/Desktop/POSRexx/bin/slf4j-simple-1.7.25.jar"

In the same file it is also possible to add the *JAVA_HOME* variable by adding *export JAVA_HOME=* and the full path to the java directory.

Figure 29: Opening "bashrc"

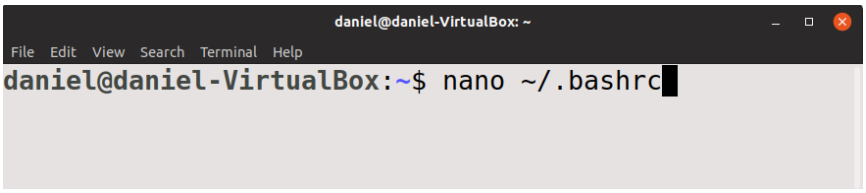
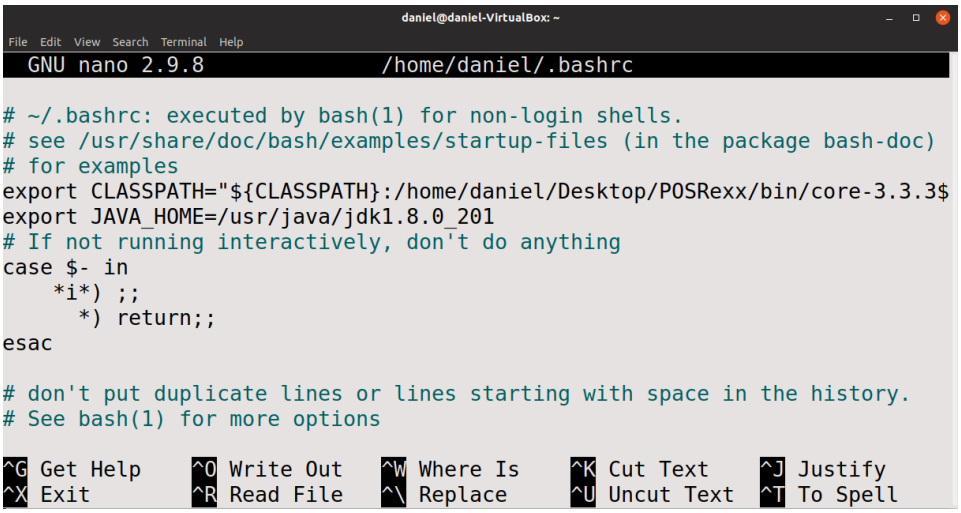
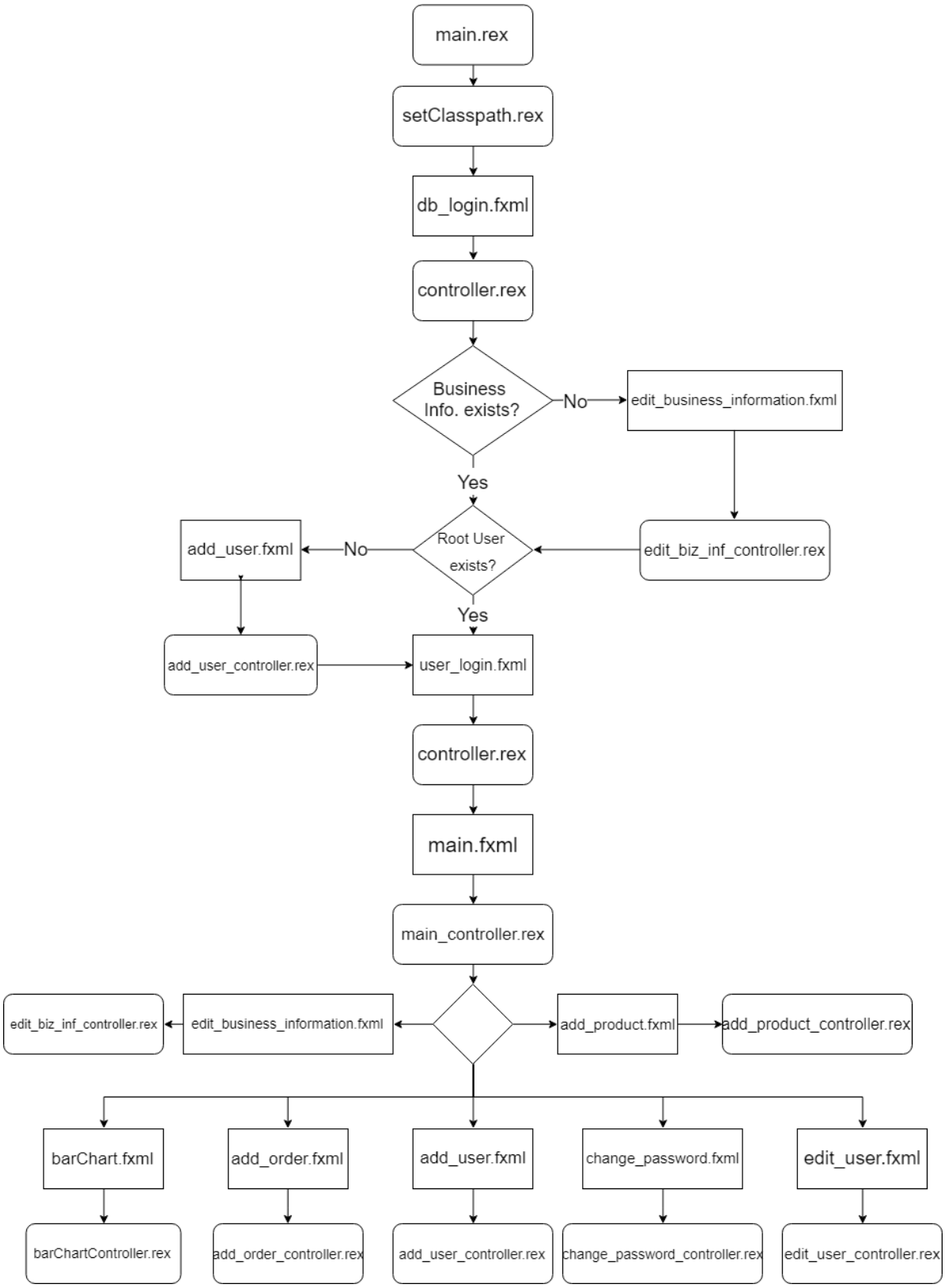


Figure 30: Editing "bashrc"



11.3 Flow Diagram

Figure 31: Flow diagram



Source: Created with `www.draw.io`

11.4 Source Code

11.4.1 Souce Code: Setup

The setup checks all requirements and sets up the database. This includes the creation of the schema as well as the necessary tables. The setup is started with *setup.rex* (Listing 41). As described in section 7.1 the script can be run without command line arguments or with the *username* and *password* that are used for the database. If no arguments are given it will read the configuration settings from *conf.ini*. If *conf.ini* does not exist it will create it with the default values *root* and *password*.
Besides *setup.rex* the setup also uses *setClasspath.rex* (Listing 42) to include all the necessary JARs and *DatabaseHandler.CLS* (Listing 43) to manage the database access. *DatabaseHandler.CLS* itself requires *ActiveUser.CLS* (Listing 44) to store information about the currently logged in user, *PasswordHasher.CLS* (Listing 45) to hash user passwords and *TableHandler.CLS* (Listing 46) to transfer data from the database to tables and the other way around.

Listing 41: Source Code: setup.rex

```
PARSE SOURCE . . fullPath
CALL directory filespec('L', fullPath)

warningCount = 0
fileName = "conf.ini"

classpath = value('CLASSPATH',, 'ENVIRONMENT')
javaHome = value('JAVA_HOME',,'ENVIRONMENT')
path = value('Path',,'ENVIRONMENT')

classpathRequirements = .array~of("mysql-connector-java", "core-3", "javase-3", "slf4j-api", "slf4j-simple"
    , "jfoenix", "jose4j")

DO req OVER classpathRequirements
  IF checkPath(classpath, req) THEN SAY "[+] SUCCESS:" req "found."
  ELSE DO
    SAY "[-] WARNING:" req "not found. CLASSPATH set corretly?"
    warningCount +=1
  END
END



IF checkPath(javaHome, "jdk1.8") | checkPath(path, "jdk1.8") THEN SAY "[+] SUCCESS: Java JDK 1.8
    found."
ELSE IF checkPath(javaHome, "jdk") | checkPath(path, "jdk") THEN DO
  SAY "[+] SUCCESS. Java JDK found."
  SAY "[-] WARNING: Java JDK not version 8. Program might not work properly."
  warningCount +=1
```

```
END
ELSE DO
  SAY "[−] WARNING: JDK not found. JAVA_HOME / Path set correctly?"
END

DriverManager = bsf.import("java.sql.DriverManager")
System = bsf.import("java.lang.System")
IF System~getProperty("java.version")~startsWith("1.8") THEN SAY "[+] SUCCESS: Java version 1.8
    installed correctly."
ELSE DO
  SAY "[−] WARNING: Wrong java version found. This program was made for java 8. Errors might occur."
  warningCount += 1
END

PARSE ARG arg1 arg2 .

editFile = .false
IF arg1 = "" | arg2 = "" THEN DO
  DB_USER = "root"
  DB_PASSWORD = "password"
END
ELSE DO
  editFile = .true
  DB_USER = arg1
  DB_PASSWORD = arg2
END
/* Read settings - Add properties */
IF \SysIsFile(fileName) | editFile THEN DO
  settings = .stream~new(fileName)
  settings~open("both replace")
  settings~lineout('[DATABASE SETTINGS]')
  settings~lineout('DB_NAME = pos_database')
  settings~lineout('DB_USER = ' || DB_USER)
  settings~lineout('DB_PASSWORD = ' || DB_PASSWORD)
  settings~lineout('DB_URL = jdbc:mysql://localhost/')
  settings~lineout('DB_TIMEZONE = CET')
  settings~lineout('DB_USE_SLL = false')
  settings~close
  SAY "[+] SUCCESS: " || fileName || " created."
END

dbh = .DatabaseHandler~new
dbh~initSettings(fileName, .false)

/* establish database connection */
SAY "[+] Connecting to database..."
```

```
success = dbh~connect
IF \success THEN CALL connectionError

DB_NAME = dbh~DB_NAME

SAY "[+] SUCCESS: Connected to database."
SAY "[+] Checking if schema exists..."
IF dbh~databaseExists(DB_NAME) THEN DO
  SAY "[+] SUCCESS: Schema " || DB_NAME || " exists."
END
ELSE DO
  SAY "[+] Schema " || DB_NAME || " does not exist."
  SAY "[+] Creating schema..."
  stmt = dbh~conn~createStatement
  SIGNAL ON SYNTAX NAME updateError
  r = stmt~executeUpdate("CREATE DATABASE" DB_NAME)
  SIGNAL OFF SYNTAX
  SAY "[+] SUCCESS: Created schema" DB_NAME
END

dbh~conn~setCatalog(DB_NAME)

tableNames = .array~of("tax", "user", "product", "orders", "orderContains", "businessInformation")
statements = .array~of("CREATE TABLE tax(TaxID TINYINT NOT NULL AUTO_INCREMENT, TaxRate
    DECIMAL(3, 3) NOT NULL, PRIMARY KEY (TaxID)) CHARACTER SET = utf8mb4", -
"CREATE TABLE user(UserID INT NOT NULL AUTO_INCREMENT, Username VARCHAR(20) NOT
    NULL, Password VARCHAR(40) NOT NULL, Salt VARCHAR(35) NOT NULL, Name VARCHAR(20)
    NOT NULL, Surname VARCHAR(20) NOT NULL, AccessRights TINYINT(1) NOT NULL,
    DateOfLastLogin DATETIME NULL, Active TINYINT NOT NULL DEFAULT 1, UNIQUE(Username),
    PRIMARY KEY(UserID)) CHARACTER SET = utf8mb4", -
"CREATE TABLE product(ProductID INT NOT NULL AUTO_INCREMENT, Name VARCHAR(30) NOT
    NULL, Price DECIMAL(10, 2) NOT NULL, Availability TINYINT NOT NULL DEFAULT 1, TaxID
    TINYINT NOT NULL, PRIMARY KEY (ProductID), FOREIGN KEY(TaxID) REFERENCES tax(TaxID
    )) CHARACTER SET = utf8mb4", -
"CREATE TABLE orders(OrderID INT NOT NULL AUTO_INCREMENT, DateOfPayment DATETIME
    NULL, UserID INT NOT NULL, TableNumber VARCHAR(15) NOT NULL, JwsHeader VARCHAR(30)
    NULL, JwsPayload VARCHAR(200) NULL, JwsSignature VARCHAR(100) NULL, PRIMARY KEY (
    OrderID), FOREIGN KEY(UserID) REFERENCES user(userID)) CHARACTER SET = utf8mb4", -
"CREATE TABLE orderContains(TempID INT NOT NULL AUTO_INCREMENT, OrderID INT NOT NULL,
    ProductID INT NOT NULL, Status VARCHAR(15) NOT NULL, PRIMARY KEY (TempID), FOREIGN
    KEY (OrderID) REFERENCES orders(OrderID), FOREIGN KEY (ProductID) REFERENCES product(
    ProductID)) CHARACTER SET = utf8mb4", -
"CREATE TABLE businessInformation(Name VARCHAR(35) NOT NULL, Postcode VARCHAR(10) NOT
    NULL, City VARCHAR(35) NOT NULL, Address VARCHAR(45) NOT NULL, PhoneNumber VARCHAR
    (25) NOT NULL, Email VARCHAR(35) NOT NULL, VatID VARCHAR(35) NOT NULL, CashRegisterID
    VARCHAR(30) NOT NULL, CertSerialNumber VARCHAR(45) NOT NULL, AES256Key VARCHAR(44)
    NOT NULL, PrivateKey VARCHAR(100) NOT NULL, PRIMARY KEY (Name)) CHARACTER SET =
```

```
    utf8mb4")

DO i=1 to tableNames~size
  IF \dbh~tableExists(tableNames[i], DB_NAME) THEN DO
    SAY "[+] Creating table" tableNames[i] || " ..."
    stmt = dbh~conn~createStatement
    SIGNAL ON SYNTAX NAME createTableError
    stmt~execute(statements[i])
    SIGNAL OFF SYNTAX
    SAY "[+] SUCCESS: Table" tableNames[i] "created."
  END
  ELSE SAY "[+]" tableNames[i] "table exists already."
END

taxInsert = "INSERT INTO tax(taxRate) VALUES (0.2), (0.19), (0.13), (0.1), (0)"
stmt = dbh~conn~createStatement
SIGNAL ON SYNTAX NAME createTableError
stmt~execute(taxInsert)
SIGNAL OFF SYNTAX

SAY "================================================="
SAY "[+] Setup completed." warningCount "warning(s) found."
SAY "Press enter to quit."
PARSE PULL .
EXIT 0

connectionError:
  SAY "[-] ERROR: Failed connecting to database."
  SAY "[-] ERROR: Verify" fileName
  CALL quit

updateError:
  SAY "[-] ERROR: Failed updating database."
  CALL quit

createTableError:
  SAY "[-] ERROR: Failed creating table."
  CALL quit

quit:
  SAY "[-] ERROR: Quitting program."
  SAY "================================================="
  SAY "[-] Setup unsuccessful."
  SAY "Press enter to quit."
  PARSE PULL .
  EXIT -1
```

```
::ROUTINE checkPath
  USE ARG path, fileName
  IF POS(fileName, path) <> 0 THEN RETURN .true
  ELSE RETURN .false

::REQUIRES "setClasspath.rex"
::REQUIRES "BSF.CLS"
::REQUIRES "DatabaseHandler.CLS"
```

Listing 42: Source Code: setClasspath.rex

```
PARSE SOURCE . . fullPath
homeDir = filespec('Location',fullPath)
osVer = SysVersion()
os = lower(left(osVer, 1))

IF os <> 'w' & os <> 'l' THEN DO
  SAY "[-] WARNING: For operating systems other than Windows and Linux the classpath has to be set
      manually."
  RETURN
END

SELECT
  WHEN os == 'w' THEN binDir = homeDir || "bin\"
  WHEN os == 'l' THEN binDir = homeDir || "bin/"
END

CALL SysFileTree binDir, "file", "O"

DO i=1 TO file.0
  IF POS(".jar", lower(file.i)) > 0 THEN DO
    SELECT
      WHEN os == 'w' THEN "set CLASSPATH=%CLASSPATH%;" || file.i
      WHEN os == 'l' THEN "export CLASSPATH=$CLASSPATH:" || file.i
      OTHERWISE
    END
  END
END
```

Listing 43: Source Code: DatabaseHandler.CLS

```
/**
  * DatabaseHandler Class stores, updates and retrieves  data from the database
*/

::CLASS DatabaseHandler PUBLIC
::METHOD conn ATTRIBUTE
```

```
::METHOD DB_URL ATTRIBUTE
::METHOD DB_NAME ATTRIBUTE
::METHOD DriverManager ATTRIBUTE
::METHOD FXCollections ATTRIBUTE
::METHOD properties ATTRIBUTE
::METHOD init
  EXPOSE DriverManager FXCollections settings
  DriverManager = bsf.import("java.sql.DriverManager")
  FXCollections = bsf.import("javafx.collections.FXCollections")

::METHOD initSettings PUBLIC
  EXPOSE properties DB_URL DB_NAME
  USE ARG fileName, withSchema=.true

  properties = .bsf~new("java.util.Properties")
  settings = .stream~new(fileName)~~open
  DO line OVER settings
    PARSE VAR line key "=" .
    PARSE VAR line "=" setting .
    key = key~strip
    setting = setting~strip
    SELECT
      WHEN key = "DB_USER" THEN properties~put("user", setting)
      WHEN key = "DB_PASSWORD" THEN properties~put("password", setting)
      WHEN key = "DB_TIMEZONE" THEN properties~put("serverTimezone", setting)
      WHEN key = "DB_USE_SLL" THEN properties~put("useSSL", setting)
      WHEN key = "DB_URL" THEN DB_URL = setting
      WHEN key = "DB_NAME" THEN DB_NAME = setting
      OTHERWISE
    END
  END
  settings~close
  IF withSchema THEN DO
    DB_URL = DB_URL || DB_NAME
  END

::METHOD connect PUBLIC
  EXPOSE DriverManager DB_URL properties conn

  SIGNAL ON SYNTAX NAME connectionError
  conn = DriverManager~getConnection(DB_URL, properties)
  SIGNAL OFF SYNTAX

  RETURN .true

connectionError: /* Connection failed - show error text */
  RETURN .false
```

```
::METHOD rootUserExists PUBLIC
  EXPOSE conn
  statement = conn~createStatement
  queryResults = statement~executeQuery("SELECT * FROM user WHERE accessRights=4")
  IF queryResults~next THEN DO -- Query returned something = Root User Exists
    RETURN .true
  END
  RETURN .false -- No root user found

::METHOD bizInfExists PUBLIC
  EXPOSE conn
  statement = conn~createStatement
  queryResults = statement~executeQuery("SELECT Name FROM businessInformation")
  IF queryResults~next THEN DO -- Entry for business informations exists
    RETURN .true
  END
  RETURN .false -- No information found

::METHOD removeBizInf PUBLIC
  EXPOSE conn
  query = "DELETE FROM businessInformation"
  prepStatement = conn~prepareStatement(query)
  prepStatement~execute

::METHOD addBusinessInformation PUBLIC
  EXPOSE conn
  USE ARG name, postcode, city, address, number, email, vatID, cashRegID, certSerialNr, aes256, priv
  query = "INSERT INTO businessInformation VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, name)
  prepStatement~setString(2, postcode)
  prepStatement~setString(3, city)
  prepStatement~setString(4, address)
  prepStatement~setString(5, number)
  prepStatement~setString(6, email)
  prepStatement~setString(7, vatID)
  prepStatement~setString(8, cashRegID)
  prepStatement~setString(9, certSerialNr)
  prepStatement~setString(10, aes256)
  prepStatement~setString(11, priv)
  prepStatement~execute

::METHOD loadUserInformation PUBLIC
  EXPOSE conn
  USE ARG username
```

```
  query = "SELECT UserID, Name, Surname, AccessRights FROM user WHERE Username = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, username)
  queryResults = prepStatement~executeQuery

  IF queryResults~next THEN DO
    userID = queryResults~getString("UserID")
    name = queryResults~getString("Name")
    surname = queryResults~getString("Surname")
    accessRights = queryResults~getString("AccessRights")
  END

  RETURN .ActiveUser~new(userID, username, name, surname, accessRights)

::METHOD passwordCorrect PUBLIC
  EXPOSE conn
  USE ARG username, password
  query = "SELECT Password, Salt FROM user WHERE Username = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, username)
  queryResults = prepStatement~executeQuery
  IF \queryResults~next THEN DO
    RETURN .false
  END

  saltHex = queryResults~getString("Salt")
  passwordHashStored = queryResults~getString("Password")
  pwh = .PasswordHasher~new
  salt  = pwh~hexStringToByteArray(saltHex)
  passwordHash = pwh~generatePasswordHash(password, salt)

  RETURN passwordHash == passwordHashStored

::METHOD usernameExists PUBLIC
  EXPOSE conn
  USE ARG username
  query = "SELECT UserID FROM user WHERE Username = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, username)
  queryResults = prepStatement~executeQuery
  IF \queryResults~next THEN DO
    RETURN .false
  END
  RETURN .true

::METHOD tableExists PUBLIC
  EXPOSE conn
```

```
  USE ARG tableName, schema
  stmt = conn~createStatement
  meta = conn~getMetaData
  tables = meta~getTables(schema, .nil, tableName, .nil)
  IF tables~next THEN DO
    RETURN .true
  END
  RETURN .false

::METHOD databaseExists PUBLIC
  EXPOSE conn
  USE ARG DB_NAME
  queryResults = conn~getMetaData~getCatalogs
  DO WHILE queryResults~next
    dbnameResult = queryResults~getString(1)
    IF dbnameResult = DB_NAME THEN DO
      RETURN .true
    END
  END
  RETURN .false

::METHOD addUserToDB PUBLIC
  EXPOSE conn
  USE ARG username, password, name, surname, accessRights
  pwh = .PasswordHasher~new

  salt = pwh~generateRandomSalt
  /* Convert to hex string for storage in database */
  saltHexString = pwh~byteArrayToHexString(salt)
  /* hash password */
  passwordHash = pwh~generatePasswordHash(password, salt)
  query = "INSERT INTO user(Username, Password, Salt, Name, Surname, AccessRights) VALUES (?, ?, ?, ?,
      ?, ?)"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, username)
  prepStatement~setString(2, passwordHash)
  prepStatement~setString(3, saltHexString)
  prepStatement~setString(4, name)
  prepStatement~setString(5, surname)
  prepStatement~setInt(6, accessRights)
  prepStatement~execute

::METHOD changePassword PUBLIC
  EXPOSE conn
  USE ARG username, newPassword

  pwh = .PasswordHasher~new
```

```
  salt  = pwh~generateRandomSalt
  saltHexString = pwh~byteArrayToHexString(salt)
  passwordHash = pwh~generatePasswordHash(newPassword, salt)

  query = "UPDATE user SET Password = ?, Salt = ? WHERE username = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, passwordHash)
  prepStatement~setString(2, saltHexString)
  prepStatement~setString(3, username)
  prepStatement~executeUpdate

::METHOD toggleAccountActiveState PUBLIC
  EXPOSE conn
  USE ARG id
  query = "UPDATE user SET Active = !Active WHERE UserID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, id)
  prepStatement~executeUpdate

::METHOD accountActive PUBLIC
  EXPOSE conn
  USE ARG username
  query = "SELECT Active FROM user WHERE username = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, username)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    active  = queryResults~getString("Active")
  END
  RETURN active


::METHOD addProductToDB PUBLIC
  EXPOSE conn
  USE ARG name, price, taxRate
  taxID = self~getTaxID(taxRate)
  query = "INSERT INTO product(Name, Price, TaxID) VALUES (?, ?, ?)"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, name)
  prepStatement~setString(2, price)
  prepStatement~setString(3, taxID)
  prepStatement~execute

::METHOD getTaxRate PUBLIC
  EXPOSE conn
  USE ARG taxID
  query = "SELECT TaxRate FROM tax WHERE TaxID = ?"
```

```
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, taxID)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    RETURN queryResults~getString("TaxRate")
  END


/* Input: taxRate in format: i.e. "20%" */
/* Returns: TaxID */
::METHOD getTaxID PUBLIC
  EXPOSE conn
  USE ARG taxRate
  taxRate = translate(taxRate, "", "%")
  taxRate = taxRate/100
  query = "SELECT TaxID FROM tax WHERE TaxRate = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, taxRate)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    RETURN queryResults~getString("TaxID")
  END

::METHOD getTaxByProductID PUBLIC
  EXPOSE conn
  USE ARG prodID
  query = "SELECT TaxID FROM product WHERE ProductID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, prodID)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    taxID = queryResults~getString("TaxID")
  END
  RETURN self~getTaxRate(taxID)

::METHOD insertLastLoginDate PUBLIC
  EXPOSE conn
  USE ARG id
  query = "UPDATE user SET DateOfLastLogin = now() WHERE UserID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, id)
  prepStatement~execute


::METHOD getOpenOrderArrayList PUBLIC
  /* If DateOfPayment is null the order is still open */
  EXPOSE conn FXCollections
```

```
  query = "SELECT OrderID, TableNumber FROM orders WHERE DateOfPayment IS NULL"
  statement = conn~createStatement
  queryResults = statement~executeQuery(query)

  orderArrayList = FXCollections~observableArrayList
  DO WHILE queryResults~next
    id = queryResults~getString("OrderID")
    tableNumber = queryResults~getString("TableNumber")
    orderArrayList~add(.Order~new(id, tableNumber))
  END
  RETURN orderArrayList

/* Runs a MySQL query and returns products in an observableArrayList */
::METHOD getProductArrayList PUBLIC
  EXPOSE conn FXCollections
  query = "SELECT * FROM product WHERE Availability = 1"
  statement = conn~createStatement
  queryResults = statement~executeQuery(query)

  productArrayList = FXCollections~observableArrayList
  DO WHILE queryResults~next
    id = queryResults~getString("ProductID")
    name = queryResults~getString("Name")
    price = queryResults~getString("Price")
    taxID = queryResults~getString("TaxID")
    taxRate = self~getTaxRate(taxID)
    productArrayList~add(.Product~new(id, name, price, taxRate))
  END
  RETURN productArrayList

::METHOD getUserArrayList PUBLIC
  EXPOSE conn FXCollections
  query = "SELECT UserID, Username, Name, Surname, AccessRights, DateOfLastLogin, Active FROM user"
  statement = conn~createStatement
  queryResults = statement~executeQuery(query)
  userArrayList = FXCollections~observableArrayList
  DO WHILE queryResults~next
    id = queryResults~getString("UserID")
    username = queryResults~getString("Username")
    name = queryResults~getString("Name")
    surname = queryResults~getString("Surname")
    accessRight = queryResults~getString("AccessRights")
    dateOfLastLogin = queryResults~getString("DateOfLastLogin")
    active = queryResults~getString("Active")
    IF active THEN active = "Active"
    ELSE active = "Deactivated"
```

```
    userArrayList~add(.User~new(id, username, name, surname, accessRight, dateOfLastLogin, active))
  END
  RETURN userArrayList

::METHOD addOrderToDB PUBLIC
  EXPOSE conn
  USE ARG tableNumber
  query = "INSERT INTO orders(UserID, TableNumber) VALUES (?, ?)"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, .my.app~activeUser~id)
  prepStatement~setString(2, tableNumber)
  prepStatement~execute

  -- Get ID of last inserted order
  query = "SELECT LAST_INSERT_ID()"
  statement = conn~createStatement
  queryResults = statement~executeQuery(query)
  queryResults~next
  lastInsertedID = queryResults~getString("LAST_INSERT_ID()")
  RETURN lastInsertedID

::METHOD addOrderDetailsToDB PUBLIC
  EXPOSE conn
  USE ARG orderID, productID, status
  query = "INSERT INTO orderContains(OrderID, ProductID, Status) VALUES (?, ?, ?)"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, orderID)
  prepStatement~setString(2, productID)
  prepStatement~setString(3, status)
  prepStatement~execute

::METHOD setOrderDetailsById PUBLIC
  EXPOSE conn FXCollections
  USE ARG id, orderDetailsHandler, paymentTable
  orderDetailsHandler~orderArrayList = FXCollections~observableArrayList

  query = "SELECT product.ProductID, product.Name, product.Price, orderContains.TempID, orderContains.
      Status FROM orderContains INNER JOIN product ON product.ProductID = orderContains.ProductID
      WHERE OrderID = ?"

  IF \paymentTable THEN DO
    query = query "GROUP BY ProductID"
  END

  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, id)
  queryResults = prepStatement~executeQuery
```

```
  DO WHILE queryResults~next
    productID = queryResults~getString("ProductID")
    productName = queryResults~getString("Name")
    productPrice = queryResults~getString("Price")
    tempID = queryResults~getString("TempID")
    status = queryResults~getString("Status")
    If paymentTable THEN DO
      orderDetailsHandler~addItem(.OrderDetails~new(productID, productName, productPrice, 1, status,
          tempID))
    END
    ELSE DO
      countQuery = "SELECT COUNT(ProductID) AS Count FROM orderContains WHERE OrderID = ?
          AND ProductID = ?"
      prepStatement = conn~prepareStatement(countQuery)
      prepStatement~setString(1, id)
      prepStatement~setString(2, productID)
      queryResultCount = prepStatement~executeQuery
      IF queryResultCount~next THEN DO
        quantity = queryResultCount~getString("Count")
      END
      orderDetailsHandler~addItem(.OrderDetails~new(productID, productName, productPrice, quantity,
          status))
    END
  END

::METHOD removeOrderDetailsFromDB PUBLIC
  EXPOSE conn
  USE ARG id, status=.nil
  IF status = .nil THEN DO
    query = "DELETE FROM orderContains WHERE OrderID = ?"
    prepStatement = conn~prepareStatement(query)
    prepStatement~setString(1, id)
  END
  ELSE DO
    query = "DELETE FROM orderContains WHERE OrderID = ? AND Status = ?"
    prepStatement = conn~prepareStatement(query)
    prepStatement~setString(1, id)
    prepStatement~setString(2, status)
  END
  prepStatement~execute

::METHOD setStatusInDB PUBLIC
  EXPOSE conn
  USE ARG status, tempID
  query = "UPDATE orderContains SET Status = ? WHERE TempID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, status)
```

```
  prepStatement~setString(2, tempID)
  prepStatement~executeUpdate

::METHOD insertDateOfPayment PUBLIC
  EXPOSE conn
  USE ARG orderID
  query = "UPDATE orders SET DateOfPayment = NOW() WHERE OrderID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, orderID)
  prepStatement~executeUpdate

  query = "SELECT DateOfPayment FROM orders WHERE OrderID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, orderID)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    date = queryResults~getString("DateOfPayment")
  END
  RETURN date

::METHOD orderEmpty PUBLIC
  EXPOSE conn
  USE ARG orderID
  query = "SELECT * FROM orderContains WHERE orderID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, orderID)
  queryResults = prepStatement~executeQuery
  IF \queryResults~next THEN DO
    RETURN .true
  END
  RETURN .false

::METHOD removeOrderFromDB PUBLIC
  EXPOSE conn
  USE ARG orderID
  query = "DELETE FROM orders WHERE OrderID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, orderID)
  prepStatement~execute

::METHOD makeProductUnavailable PUBLIC
  EXPOSE conn
  USE ARG productID
  query = "UPDATE product SET Availability = 0 WHERE ProductID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, productID)
  prepStatement~executeUpdate
```

```
::METHOD updateAccessRight PUBLIC
  EXPOSE conn
  USE ARG username, accessRight
  query = "UPDATE user SET accessRights = ? WHERE Username = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, accessRight)
  prepStatement~setString(2, username)
  prepStatement~executeUpdate


::METHOD getBusinessInformation PUBLIC
  EXPOSE conn
  bizInf = .table~new
  query = "SELECT * FROM businessInformation"
  prepStatement = conn~prepareStatement(query)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    bizInf["Name"] = queryResults~getString("Name")
    bizInf["Postcode"] = queryResults~getString("Postcode")
    bizInf["City"] = queryResults~getString("City")
    bizInf["Address"] = queryResults~getString("Address")
    bizInf["PhoneNumber"] = queryResults~getString("PhoneNumber")
    bizInf["Email"] = queryResults~getString("Email")
    bizInf["VatID"] = queryResults~getString("VatID")
    bizInf["CashRegisterID"] = queryResults~getString("CashRegisterID")
    bizInf["CertSerialNumber"] = queryResults~getString("CertSerialNumber")
    bizInf["AES256Key"] = queryResults~getString("AES256Key")
    bizInf["PrivateKey"] = queryResults~getString("PrivateKey")
  END
  RETURN bizInf

::METHOD getTurnover PUBLIC
  EXPOSE conn
  query = 'SELECT sum(product.Price) AS sum FROM product INNER JOIN orderContains ON
      orderContains.ProductID = product.ProductID WHERE orderContains.Status = "Paid"'
  prepStatement = conn~prepareStatement(query)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    turnover = queryResults~getString("sum")
  END
  RETURN turnover

::METHOD getDistinctMonths PUBLIC
  EXPOSE conn
  query = "SELECT COUNT(DISTINCT MONTH(DateOfPayment), YEAR(DateOfPayment)) as count
      FROM orders"
```

```
  prepStatement = conn~prepareStatement(query)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN count = queryResults~getString("count")
  ELSE count = 0
  RETURN count

::METHOD countActiveRootUsers PUBLIC
  EXPOSE conn
  query = "SELECT count(Username) AS count FROM user WHERE AccessRights = 4 AND Active = 1"
  prepStatement = conn~prepareStatement(query)
  queryResults = prepStatement~executeQuery
  IF queryResults~next THEN DO
    count = queryResults~getString("count")
  END
  RETURN count

::METHOD getLastOrderJWS PUBLIC
  EXPOSE conn
  query = "SELECT JwsHeader, JwsPayload, JwsSignature FROM orders ORDER BY DateOfPayment desc
      LIMIT 2"
  prepStatement = conn~prepareStatement(query)
  queryResults = prepStatement~executeQuery

  DO WHILE queryResults~next
     header = queryResults~getString("JwsHeader")
    payload = queryResults~getString("JwsPayload")
    signature = queryResults~getString("JwsSignature")
  END

  IF header = .nil THEN RETURN .array~of(.nil, .nil)
  RETURN .array~of(payload, signature)



::METHOD storeJWS PUBLIC
  EXPOSE conn
  USE ARG id, header, payload, signature
  query = "UPDATE orders SET JwsHeader = ?, JwsPayload = ?, JwsSignature = ? WHERE OrderID = ?"
  prepStatement = conn~prepareStatement(query)
  prepStatement~setString(1, header)
  prepStatement~setString(2, payload)
  prepStatement~setString(3, signature)
  prepStatement~setString(4, id)
  prepStatement~executeUpdate

::METHOD getMonthlyTurnover PUBLIC
    EXPOSE conn
    query = 'SELECT MONTH(DateOfPayment), MONTHNAME(DateOfPayment) AS month, YEAR(
```

```
        DateOfPayment) AS year, sum(product.Price) AS sum FROM product INNER JOIN orderContains
        ON orderContains.ProductID = product.ProductID INNER JOIN orders ON orders.OrderID =
        orderContains.OrderID WHERE orderContains.Status = "Paid" GROUP BY YEAR(orders.
        DateOfPayment), MONTHNAME(orders.DateOfPayment), MONTH(orders.DateOfPayment) ORDER
        BY YEAR(orders.DateOfPayment), MONTH(orders.DateOfPayment)'
    prepStatement = conn~prepareStatement(query)
    queryResults = prepStatement~executeQuery
    dates = .array~new
    revenues = .array~new
    DO WHILE queryResults~next
        dates~append(queryResults~getString("month") queryResults~getString("year"))
        revenues~append(queryResults~getString("sum"))
    END
    monthlyTurnover = .table~new
    monthlyTurnover['dates'] = dates
    monthlyTurnover['revenues'] = revenues
    RETURN monthlyTurnover



::REQUIRES "BSF.CLS"
::REQUIRES "ActiveUser.CLS"
::REQUIRES "PasswordHasher.CLS"
::REQUIRES "TableHandler.CLS"
```

Listing 44: Source Code: ActiveUser.CLS

```
/**
  * ActiveUser Class keeps track of the currently logged in user
*/

::CLASS ActiveUser PUBLIC
::METHOD id ATTRIBUTE
::METHOD username ATTRIBUTE
::METHOD name ATTRIBUTE
::METHOD surname ATTRIBUTE
::METHOD accessRights ATTRIBUTE
::METHOD INIT
  EXPOSE id username name surname accessRights
  USE ARG id, username, name, surname, accessRights
```

Listing 45: Source code: PasswordHasher.CLS

```
/**
  * JWSHandler Class creates PBKDF2 hashes and generates new random salts
*/

::CLASS PasswordHasher PUBLIC
```

```
::METHOD DatatypeConverter ATTRIBUTE
::METHOD INIT
  EXPOSE DatatypeConverter
  DatatypeConverter = bsf.import("javax.xml.bind.DatatypeConverter")

::METHOD generatePasswordHash PUBLIC
  USE ARG password, salt

  SecretKeyFactory = bsf.import("javax.crypto.SecretKeyFactory")
  PBEKeySpec = bsf.import("javax.crypto.spec.PBEKeySpec")

  iterationCount = 100000
  keyLength = 160

  passwordCharArr = self~stringToJavaCharArray(password)
  spec = PBEKeySpec~new(passwordCharArr, salt, iterationCount, keyLength)
  factory = SecretKeyFactory~getInstance("PBKDF2WithHmacSHA1")
  hash = factory~generateSecret(spec)~getEncoded
  hashString = self~byteArrayToHexString(hash)
  RETURN hashString

::METHOD generateRandomSalt PUBLIC
  random = .bsf~new("java.security.SecureRandom")
  salt = bsf.createArray("byte.class", 16)
  random~nextBytes(salt)
  RETURN salt

::METHOD byteArrayToHexString PUBLIC
  USE ARG byteArr
  RETURN C2X(BsfRawBytes(byteArr))

::METHOD hexStringToByteArray PUBLIC
  USE ARG hexString
  RETURN BsfRawBytes(X2C(hexString))

::METHOD stringToJavaCharArray PUBLIC
  USE ARG inputString
  len = length(inputString)
  byteArr = bsf.createJavaArray("char.class", len)
  DO i=1 TO len
    byteArr[i] = inputString[i]
  END
  RETURN byteArr

::REQUIRES "BSF.CLS"
```

Listing 46: Source Code: TableHandler.CLS

```
/* Class PropertyValueFactory taken out of sample fxml_99 by Prof. Rony G. Flatscher under Apache Version
    2.0 license */
/* Handler Classes are using PersonOverviewController Class out of sample fxml_99 by Prof. Rony. G. Flatscher
     as a reference under Apache Version 2.0 license */

 /*
     ---------------------- Apache Version 2.0 license ----------------------
        Copyright 2016-2017 Rony G. Flatscher

        Licensed under the Apache License, Version 2.0 (the "License");
        you may not use this file except in compliance with the License.
        You may obtain a copy of the License at

            http://www.apache.org/licenses/LICENSE-2.0

        Unless required by applicable law or agreed to in writing, software
        distributed under the License is distributed on an "AS IS" BASIS,
        WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
        See the License for the specific language governing permissions and
        limitations under the License.
     ------------------------------------------------------------------------
*/



/* ============================================================== */

/* implements "R javafx.util.Callback<P,R>(P o) for PropertyValueFactory */

/* This class allows instances that remember the message to be sent to person instances to
   return the property of the attribute that should be shown in the table cell.
*/

::class PropertyValueFactory PUBLIC
::method init
  expose propName --handler -- name of property getter method
  use strict arg propName -- , handler

::method call
  expose propName -- handler
  use arg o          -- an observable value for the ooRexx person object boxed in a Java RexxProxy object
  return BsfRexxProxy(o~getValue)~send(propName)

::CLASS Order PUBLIC
::METHOD id ATTRIBUTE
::METHOD tableNumber ATTRIBUTE
```

```
::METHOD INIT
  EXPOSE id tableNumber
  USE ARG idArg, tableNumberArg
  SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")
  id = SimpleStringProperty~new(idArg)
  tableNumber = SimpleStringProperty~new(tableNumberArg)

::CLASS OrderHandler PUBLIC
::METHOD INIT
  EXPOSE orderArrayList table idColumn tableColumn
  USE ARG orderArrayList, table, idColumn, tableColumn

  listener = BsfCreateRexxProxy(self,,"javafx.event.EventHandler")
  table~setOnMouseClicked(listener)
  --table~getSelectionModel~selectedItemProperty~addListener(listener)
  table~setColumnResizePolicy(table~CONSTRAINED_RESIZE_POLICY)

  idColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("id"), ,"javafx.util.Callback
      "))
  tableColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("tableNumber"), ,"
      javafx.util.Callback"))
  self~setItems

::METHOD setItems
  EXPOSE table orderArrayList
  table~setItems(orderArrayList)

::METHOD handle
  USE ARG event
  table = event~getSource
  IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
    order = table~getSelectionModel~getSelectedItem
    /* Convert Rexx proxy back to order instance */
    order = BsfRexxProxy(order)
    FXCollections = bsf.import("javafx.collections.FXCollections")
    .my.app~orderDetailsHandler~orderArrayList = FXCollections~observableArrayList

    .my.app~dbh~setOrderDetailsById(order~id~get, .my.app~orderDetailsHandler, .true)
    .my.app~orderText~text = "Order:" order~id~get
    .my.app~currentOrderID = order~id~get
    .my.app~tableNumber = order~tableNumber~get
  END

::METHOD orderArrayList ATTRIBUTE
::METHOD table ATTRIBUTE
::METHOD idColumn ATTRIBUTE
::METHOD tableColumn ATTRIBUTE
```

```
::CLASS OrderDetails PUBLIC
::METHOD tempID ATTRIBUTE
::METHOD productID ATTRIBUTE
::METHOD productName ATTRIBUTE
::METHOD productPrice ATTRIBUTE
::METHOD quantity ATTRIBUTE
::METHOD status ATTRIBUTE
::METHOD INIT
  EXPOSE productID productName productPrice quantity status tempID
  USE ARG productIDArg, productNameArg, productPriceArg, quantityArg, statusArg, tempID=1

  SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")
  SimpleIntegerProperty = bsf.import("javafx.beans.property.SimpleIntegerProperty")
  SimpleFloatProperty = bsf.import("javafx.beans.property.SimpleFloatProperty")

  productID = SimpleStringProperty~new(productIDArg)
  productName = SimpleStringProperty~new(productNameArg)
  productPrice = SimpleFloatProperty~new(productPriceArg)
  quantity = SimpleIntegerProperty~new(quantityArg)
  status = SimpleStringProperty~new(statusArg)

::CLASS OrderDetailsHandler PUBLIC
::METHOD INIT
  EXPOSE orderArrayList table idColumn nameCol priceCol quantityCol statusCol statusMessage
      paymentTable
  USE ARG orderArrayList, table, idColumn, nameCol, priceCol, quantityCol, statusCol, paymentTable=.true

  statusMessage = .array~new
  statusMessage[1] = "Not paid"
  statusMessage[2] = "Pending"
  statusMessage[3] = "Paid"

  listener  = BsfCreateRexxProxy(self,,"javafx.event.EventHandler")
  table~setOnMouseClicked(listener)
  --table~getSelectionModel~selectedItemProperty~addListener(listener)
  table~setColumnResizePolicy(table~CONSTRAINED_RESIZE_POLICY)
  idColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("productID"), ,"javafx.util.
      Callback"))
  nameCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("productName"), ,"javafx.
      util.Callback"))
  priceCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("productPrice"), ,"javafx.util
      .Callback"))
  IF paymentTable THEN DO
      statusCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("status"), ,"javafx.util.
          Callback"))
    END
```

```
    ELSE DO
      quantityCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("quantity"), ,"javafx.
          util.Callback"))
    END

  self~setItems

::METHOD orderArrayList ATTRIBUTE
::METHOD table ATTRIBUTE
::METHOD idColumn ATTRIBUTE
::METHOD nameCol ATTRIBUTE
::METHOD priceCol ATTRIBUTE
::METHOD quantityCol ATTRIBUTE
::METHOD paymentTable ATTRIBUTE
::METHOD statusMessage ATTRIBUTE
::METHOD setItems
  EXPOSE table orderArrayList
  table~setItems(orderArrayList)
::METHOD addItem
  EXPOSE table orderArrayList
  USE ARG item
  orderArrayList~add(item)
  self~setItems

::METHOD handle
  EXPOSE statusMessage paymentTable
  USE ARG event
  table = event~getSource
  order = table~getSelectionModel~getSelectedItem

  IF order <> .nil THEN DO
    -- Convert Rexx proxy back to order instance
    orderInstance = BsfRexxProxy(order)
    tempID = orderInstance~tempID
    productID = orderInstance~productID~get
    productName = orderInstance~productName~get
    productPrice = orderInstance~productPrice~get
    quantity = orderInstance~quantity~get
    status = orderInstance~status~get
    IF paymentTable THEN DO
      IF status = statusMessage[2] THEN DO
        orderInstance~status~set(statusMessage[1])
        .my.app~dbh~setStatusInDB(statusMessage[1], tempID) -- Update changes to DB
      END
      ELSE IF status = statusMessage[1] THEN DO
        orderInstance~status~set(statusMessage[2])
        .my.app~dbh~setStatusInDB(statusMessage[2], tempID)
```

```
      END
    END
    ELSE DO -- Not paymentTable
      IF quantity > 1 THEN DO
        orderInstance~quantity~set(quantity-1)
      END
      ELSE DO
        table~getItems~remove(order)
      END
    END
  END

::CLASS Product PUBLIC
::METHOD id ATTRIBUTE
::METHOD name ATTRIBUTE
::METHOD price ATTRIBUTE
::METHOD tax ATTRIBUTE
::METHOD INIT
  EXPOSE id name price tax
  USE ARG idArg, nameArg, priceArg, taxArg

  SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")

  id = SimpleStringProperty~new(idArg)
  name = SimpleStringProperty~new(nameArg)
  price = SimpleStringProperty~new(priceArg)
  tax = SimpleStringProperty~new(taxArg)



::CLASS ProductHandler PUBLIC
::METHOD INIT
  EXPOSE productArrayList table idColumn nameColumn priceColumn taxColumn
  USE ARG productArrayList, table, idColumn, nameColumn, priceColumn, taxColumn

  listener = BsfCreateRexxProxy(self,,"javafx.event.EventHandler")
  table~setOnMouseClicked(listener)
  --table~getSelectionModel~selectedItemProperty~addListener(listener)

  idColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("id"), ,"javafx.util.Callback
      "))
  nameColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("name"), ,"javafx.util.
      Callback"))
  priceColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("price"), ,"javafx.util.
      Callback"))
  taxColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("tax"), ,"javafx.util.
      Callback"))
  table~setColumnResizePolicy(table~CONSTRAINED_RESIZE_POLICY)
```

```
  self~setItems

::METHOD productArrayList ATTRIBUTE
::METHOD table ATTRIBUTE
::METHOD idColumn ATTRIBUTE
::METHOD nameColumn ATTRIBUTE
::METHOD priceColumn ATTRIBUTE
::METHOD taxColumn ATTRIBUTE
::METHOD setItems
  EXPOSE table productArrayList
  table~setItems(productArrayList)

::METHOD handle
  USE ARG event
  table = event~getSource
  IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
    product = table~getSelectionModel~getSelectedItem
    /* Convert Rexx proxy back to product instance */
    product = BsfRexxProxy(product)
    orderDetails = .OrderDetails~new(product~id~get, product~name~get, product~price~get, 1, "Not paid"
        )
    item = findProductInOrder(orderDetails)
    IF item = .nil THEN DO
      CALL addOrderDetails orderDetails
    END
    ELSE DO
      CALL incrementQuantity item
    END
  END


::CLASS User PUBLIC
::METHOD id ATTRIBUTE
::METHOD username ATTRIBUTE
::METHOD name ATTRIBUTE
::METHOD surname ATTRIBUTE
::METHOD accessRight ATTRIBUTE
::METHOD dateOfLastLogin ATTRIBUTE
::METHOD active ATTRIBUTE
::METHOD INIT
  EXPOSE id username name surname accessRight dateOfLastLogin active
  USE ARG idArg, usernameArg, nameArg, surnameArg, accessRightArg, dateOfLastLoginArg, activeArg

  SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")

  id = SimpleStringProperty~new(idArg)
```

```
  username = SimpleStringProperty~new(usernameArg)
  name = SimpleStringProperty~new(nameArg)
  surname = SimpleStringProperty~new(surnameArg)
  accessRight = SimpleStringProperty~new(accessRightArg)
  dateOfLastLogin = SimpleStringProperty~new(dateOfLastLoginArg)
  active = SimpleStringProperty~new(activeArg)


::CLASS UserHandler PUBLIC
::METHOD INIT
  EXPOSE userArrayList table idCol usernameCol nameCol surnameCol accessRightCol dateOfLastLoginCol
      activeCol
  USE ARG userArrayList, table, idCol, usernameCol, nameCol, surnameCol, accessRightCol,
      dateOfLastLoginCol, activeCol

  idCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("id"), ,"javafx.util.Callback"))
  usernameCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("username"), ,"javafx.
      util.Callback"))
  nameCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("name"), ,"javafx.util.
      Callback"))
  surnameCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("surname"), ,"javafx.util.
      Callback"))
  accessRightCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("accessRight"), ,"
      javafx.util.Callback"))
  dateOfLastLoginCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("
      dateOfLastLogin"), ,"javafx.util.Callback"))
  activeCol~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("active"), ,"javafx.util.
      Callback"))

  table~setColumnResizePolicy(table~CONSTRAINED_RESIZE_POLICY)

  self~setItems

::METHOD userArrayList ATTRIBUTE
::METHOD table ATTRIBUTE
::METHOD idCol ATTRIBUTE
::METHOD usernameCol ATTRIBUTE
::METHOD nameCol ATTRIBUTE
::METHOD surnameCol ATTRIBUTE
::METHOD accessRightCol ATTRIBUTE
::METHOD dateOfLastLoginCol ATTRIBUTE
::METHOD activeCol ATTRIBUTE
::METHOD setItems
  EXPOSE table userArrayList
  table~setItems(userArrayList)
::METHOD updateArrayList
  EXPOSE userArrayList
```

```
  USE ARG userArrayList
  self~setItems

::REQUIRES "BSF.CLS"
::REQUIRES "functions.rex"
```

11.4.2 Souce Code: Filling the Database

For testing purposes the database can be filled with random orders. This is done with the *fillDatabase.rex* (Listing 47) script. It also makes use of *setClasspath.rex* (Listing 42) and *DatabaseHandler.CLS* (Listing 43) as shown in section 11.4.1. The amount of orders to generate can be specified as the first command line argument. If no command line argument is given it will default to 500 orders.

Listing 47: Source Code: fillDatabase.rex

```
PARSE SOURCE . . fullPath
CALL directory filespec('L', fullPath)
conn = getConnection()
IF conn = .false THEN EXIT -1
st = conn~createStatement

testUser = 'INSERT INTO user(Username, Password, Salt, Name, Surname, AccessRights, Active) VALUES ("
    Filler", "Test", "Test", "Test", "Test", 1, 0)'
products = 'INSERT INTO product(Name, Price, TaxID) VALUES' -
  '("Pizza Pepperoni", 7.8, 4),' -
  '("Pizza Cheese", 7.0, 4),' -
  '("Pizza Margarita", 6.9, 4),' -
  '("Pizza Exotica", 8.0, 4),' -
  '("Pizza Veggie", 7.3, 4),' -
  '("Pizza Fiamma", 7.3, 4),' -
  '("Pizza Salami", 7.0, 4),' -
  '("Garlic Bread", 2.0, 4),' -
  '("Cheese Fries", 3.2, 4),' -
  '("Fries", 2.9, 4),' -
  '("Greek Salad", 4.9, 4),' -
  '("Pineapple Juice", 2.8, 1),' -
  '("Orange Juice", 7.0, 1),' -
  '("Coca Cola", 3.0, 1),' -
  '("Apple Juice", 2.5, 1)'

selectID = 'SELECT UserID FROM user WHERE Username = "Filler"'
rs = st~executeQuery(selectID)
IF rs~next THEN DO
  id = rs~getString('UserID')
END
```

```
ELSE DO
  SAY "Adding new user..."
  st~executeUpdate(testUser)
  SAY "Getting inserted ID..."
  rs = st~executeQuery("SELECT LAST_INSERT_ID()")
  IF rs~next THEN id = rs~getString("LAST_INSERT_ID()")

  SAY "Adding products..."
  st~executeUpdate(products)
END

getFirstProduct = 'SELECT ProductID from product WHERE Name = "Pizza Pepperoni"'
rs = st~executeQuery(getFirstProduct)
IF rs~next THEN firstProductID = rs~getString('ProductID')

ordersToGenerate = 500
PARSE ARG arg1 .
if arg1 <> "" & DATATYPE(arg1) == "NUM" THEN ordersToGenerate = arg1
date = DATE('E')
PARSE VAR date "/" currentMonth "/" currentYear .
currentYear += 2000
DO i=1 TO ordersToGenerate
  SAY "Adding random order number" i || "."
  table = random(1, 15)
  day = random(1, 28)
  CALL newRandom
  DO WHILE year == currentYear & month > currentMonth
    CALL newRandom
  END

  insert = "INSERT INTO orders(DateOfPayment, UserID, TableNumber) VALUES (DATE '" || year || "-" ||
       month || "-" || day || "', " || id || ", " || table || ")"
  st~executeUpdate(insert)
  rs = st~executeQuery("SELECT LAST_INSERT_ID()")
  IF rs~next THEN orderID = rs~getString("LAST_INSERT_ID()")
  DO j=1 TO random(1, 15)
    prodID = random(firstProductID, firstProductID+14)
    insert = "INSERT INTO orderContains(OrderID, ProductID, Status) VALUES ("orderID "," prodID ", '
         Paid')"
    st~executeUpdate(insert)
  END
END
SAY "Press enter to quit."
PARSE PULL .
EXIT 0

newRandom:
```

```
  month = random(1, 12)
  year = random(currentYear-3, currentYear)
  RETURN

::ROUTINE getConnection
  dbh = .DatabaseHandler~new
  dbh~initSettings("conf.ini")
  success = dbh~connect
  IF success THEN SAY "Connection Successful."
  ELSE DO
    SAY "[-] ERROR: Unable to connect to database."
    SAY "Press enter to quit."
    PARSE PULL .
    RETURN .false
  END
  RETURN dbh~conn

::REQUIRES "setClasspath.rex"
::REQUIRES "DatabaseHandler.CLS"
```

11.4.3 Souce Code: Generating Keys

generateKeys.rex (Listing 48) generates the necessary keys and write them in a *.txt* file. These are used to encrypt elements of the receipt. It makes use of *AES256.CLS (Listing 49)* for generating an AES256 key and *JWSHandler.CLS* (Listing 50) for generating a key pair for ECDSA encryption.

Listing 48: Source Code: generateKeys.rex

```
PARSE SOURCE . . fullPath
CALL directory filespec('L', fullPath)

fileName = "key" || time("S") || ".txt"
aes = .AES256~new
jwsHandler = .JWSHandler~new
Base64 = bsf.import("java.util.Base64")
Encoder = Base64~getEncoder

SAY "Creating file" fileName || "."
 file  = .stream~new(fileName)
 file ~open("both replace")

SAY "Generating AES key..."
key = aes~generateAES256Key
SAY "Writing AES key..."
 file ~lineout("AES256 Key:" key)
SAY "Calculating hash..."
checkSum = aes~calcCheckSumFromKey(key)
```

```
SAY "Writing hash..."
 file ~lineout("Checksum:" checkSum)
SAY "Checking hash..."
IF aes~checkValSum(key, checkSum) THEN DO
   msg = "Checksum successfully calculated."
   SAY "[+] Successfully generated AES256 key."
END
ELSE DO
   msg = "Error calculating checksum."
   SAY "[-] ERROR: Generating AES256 key unsuccessful."
END
 file ~lineout(msg)
 file ~lineout("==========================")

SAY "Generating key pair..."
keyPair = jwsHandler~generateKeyPair
priv  = keyPair~getPrivate~getEncoded
pub = keyPair~getPublic~getEncoded
SAY "Writing key pair..."
 file ~lineout("Private Key:" Encoder~encodeToString(priv))
 file ~lineout("Public Key:" Encoder~encodeToString(pub))
SAY "[+] Success generated key pair."
 file ~close
SAY "Press enter to quit."
PARSE PULL .

:: REQUIRES "AES256.CLS"
:: REQUIRES "JWSHandler.CLS"
```

Listing 49: Source Code: AES256.CLS

```
/**
   * AES256 Class provides AES256 Message encryption and decryption
   * Additionally, it is able to create new AES256 keys and also checksums to verify their integrity
*/

:: CLASS AES256 PUBLIC
:: METHOD KeyGenerator ATTRIBUTE
:: METHOD Base64 ATTRIBUTE
:: METHOD SecretKeySpec ATTRIBUTE
:: METHOD Cipher ATTRIBUTE
:: METHOD INIT
   EXPOSE KeyGenerator Base64 SecretKeySpec Cipher
   KeyGenerator = bsf.import("javax.crypto.KeyGenerator")
   Base64 = bsf.import("java.util.Base64")
   SecretKeySpec = bsf.import("javax.crypto.spec.SecretKeySpec")
   Cipher = bsf.import("javax.crypto.Cipher")
```

```
::METHOD generateAES256Key PUBLIC
  EXPOSE KeyGenerator Base64

  keyGen = KeyGenerator~getInstance("AES")
  keyGen~bsf.invoke("init", 256)
  secretKey = keyGen~generateKey
  encodedSecKey = secretKey~getEncoded
  Encoder = Base64~Encoder
  base64Key = Encoder~encodeToString(encodedSecKey)
  RETURN base64Key

::METHOD encrypt PUBLIC
  EXPOSE Base64 SecretKeySpec Cipher
  USE ARG message, secretKey

  Decoder = Base64~Decoder
  encodedSecKey = Decoder~decode(secretKey)
  secKeySpec = SecretKeySpec~new(encodedSecKey, "AES")
  c = Cipher~getInstance("AES")
  c~bsf.invoke("init", Cipher~ENCRYPT_MODE, secKeySpec)
  encrypted = c~doFinal(BsfRawBytes(message))
  Encoder = Base64~Encoder
  encoded = Encoder~encodeToString(encrypted)
  RETURN encoded

::METHOD decrypt PUBLIC
  EXPOSE Base64 SecretKeySpec Cipher
  USE ARG encryptedMessage, secretKey

  Decoder = Base64~Decoder
  encodedSecKey = Decoder~decode(secretKey)
  encryptedMessage = Decoder~decode(encryptedMessage)
  secKeySpec = SecretKeySpec~new(encodedSecKey, "AES")
  c = Cipher~getInstance("AES")
  c~bsf.invoke("init", Cipher~DECRYPT_MODE, secKeySpec)
  message = c~doFinal(encryptedMessage)
  RETURN BsfRawBytes(message)

::METHOD calcCheckSumFromKey PUBLIC
  EXPOSE Base64
  USE ARG base64AESKey, N=3
  md = bsf.loadClass("java.security.MessageDigest")~getInstance("SHA-256")

  sha256hash = md~digest(BsfRawBytes(base64AESKey))
  sha256hashNbytes = bsf.createJavaArray("byte.class", N)
  bsf.loadClass("java.lang.System")~arraycopy(sha256hash, 0, sha256hashNbytes, 0, N)
```

```
  Encoder = Base64~Encoder
  base64sha256hashNbytes = Encoder~encodeToString(sha256hashNbytes)
  valSumCalc = translate(base64sha256hashNbytes, "", "=")
  return valSumCalc

::METHOD checkValSum PUBLIC
  USE ARG base64AESKey, userCheckSum, N=3
  calculatedCheckSum = self~calcCheckSumFromKey(base64AESKey, N)
  return calculatedCheckSum == userCheckSum

::REQUIRES "BSF.CLS"
```

Listing 50: Source Code: JWSHandler.CLS

```
/**
  * JWSHandler Class creates JWS Signature that are used for the QR Codes on receipts
*/

::CLASS JWSHandler PUBLIC
::METHOD MessageDigest ATTRIBUTE
::METHOD StandardCharsets ATTRIBUTE
::METHOD System ATTRIBUTE
::METHOD Encoder ATTRIBUTE
::METHOD Decoder ATTRIBUTE
::METHOD UrlDecoder ATTRIBUTE
::METHOD KeyPairGenerator ATTRIBUTE
::METHOD SecureRandom ATTRIBUTE
::METHOD Signature ATTRIBUTE
::METHOD INIT
  EXPOSE MessageDigest StandardCharsets System Encoder Decoder KeyPairGenerator SecureRandom
      UrlDecoder
  StandardCharsets = bsf.import("java.nio.charset.StandardCharsets")
  System = bsf.import("java.lang.System")
  Base64 = bsf.import("java.util.Base64")
  Encoder = Base64~Encoder
  Decoder = Base64~Decoder
  UrlDecoder = Base64~getUrlDecoder
  KeyPairGenerator = bsf.loadClass("java.security.KeyPairGenerator")
  SecureRandom = bsf.import("java.security.SecureRandom")
  MessageDigest = bsf.loadClass("java.security.MessageDigest")

::METHOD hashLastReceipt PUBLIC
  EXPOSE MessageDigest StandardCharsets System
  USE ARG str, N=8
  jStr = box("STring", str)

  digest = MessageDigest~getInstance("SHA-256")
```

```
  hash = digest~digest(jStr~getBytes(StandardCharsets~UTF_8))

  /* Copy N Bytes */
  conDigest = bsf.CreateJavaArray("byte.class", N)
  System~arraycopy(hash, 0, conDigest, 0, N)

  RETURN self~base64Encode(conDigest)

::METHOD base64Encode PUBLIC
  EXPOSE Encoder
  USE ARG str
  RETURN Encoder~encodeToString(str)

::METHOD encryptECDSA PUBLIC
  USE ARG payload, privateKey

  JsonWebSignature = bsf.import("org.jose4j.jws.JsonWebSignature")
  jws = JsonWebSignature~new
  jws~setKey(privateKey)
  jws~setPayload(payload)
  jws~setAlgorithmHeaderValue("ES256")
  jws~sign
  RETURN jws~getCompactSerialization

::METHOD privateKeyFromString PUBLIC
  EXPOSE Decoder
  USE ARG keyString
  PKCS8EncodedKeySpec = bsf.import("java.security.spec.PKCS8EncodedKeySpec")
  kf = bsf.loadClass("java. security .KeyFactory")~getInstance("EC")
  privateBytes = Decoder~decode(keyString)

  RETURN kf~generatePrivate(PKCS8EncodedKeySpec~new(privateBytes))

::METHOD generateKeyPair
  EXPOSE KeyPairGenerator SecureRandom
  keyGen = KeyPairGenerator~getInstance("EC")
  random = SecureRandom~getInstance("SHA1PRNG")
  keyGen~initialize(256, random)
  RETURN keyGen~generateKeyPair

::METHOD recode PUBLIC
  EXPOSE Encoder UrlDecoder
  USE ARG base64URL
  decoded = UrlDecoder~decode(base64URL)
  RETURN Encoder~encodeToString(decoded)

::METHOD restoreQRString PUBLIC
```

```
  EXPOSE UrlDecoder
  USE ARG payload, signature
  IF payload = .nil THEN DO
    RETURN .my.app~dbh~getBusinessInformation()["CashRegisterID"]
  END

  decodedPayloadBytes = UrlDecoder~decode(payload)
  decodedPayload = BsfRawBytes(decodedPayloadBytes)
  sigBase64 = self~recode(signature)

  RETURN decodedPayload || "_" || sigBase64

::REQUIRES "BSF.CLS"
```

11.4.4 Souce Code: Main Program

main.rex (Listing 51) is executed to start the main program. As described in section 11.4.1 it also uses *setClasspath.rex* (Listing 42) and *DatabaseHandler.CLS* (Listing 43). Additionally, it uses *functions.rex* (Listing 52) which provides additional helper routines. *functions.rex* also requires, besides the already mentioned classes, *QRWriter.CLS* (Listing 53) which is used to create QR-Codes. After loading all the relevant variables the script will open the first FXML file (= loads first scene) named *db_login.fxml* (Listing 55, Figure 32). This scene uses *controller.rex* (Listing 54) as controller. This combination of one FXML file and one controller file is the same for the rest of the source code, where the FXML file is used to create the graphical user interface and the controller handles the user interaction with it.

Listing 51: Source Code: main.rex

```
PARSE SOURCE . . fullPath
CALL directory filespec('L', fullPath)

.environment~setEntry("my.app", .directory~new)
.my.app~homeDir = filespec('Location',fullPath)
.my.app~dbh = .DatabaseHandler~new
stageHandler = .StageHandler~new
.my.app~stageHandler = stageHandler
stageHandlerProxy = BsfCreateRexxProxy(stageHandler,,"javafx.application.Application")
stageHandlerProxy~launch(stageHandlerProxy~getClass, .nil)
EXIT 0

::CLASS StageHandler
::METHOD stage ATTRIBUTE
::METHOD scene ATTRIBUTE
::METHOD windowStage ATTRIBUTE
::METHOD FXMLLoader
::METHOD init
```

```
  EXPOSE FXMLLoader
  FXMLLoader = bsf.import("javafx.fxml.FXMLLoader")
::METHOD start
  EXPOSE stage scene FXMLLoader
  USE ARG stage
  /* GraphicsEnvironment to get Screen resolution */
  GraphicsEnvironment = bsf.loadClass("java.awt.GraphicsEnvironment")
  gd = GraphicsEnvironment~getLocalGraphicsEnvironment~getDefaultScreenDevice
  stage~setTitle("Database login")
  stage~getIcons~add(.bsf~new("javafx.scene.image.Image", "file:res/icon.png"))
  url=.bsf~new("java.net.URL", "file:db_login.fxml")
  fxml = FXMLLoader~load(url)
  scene = .bsf~new("javafx.scene.Scene", fxml, gd~getDisplayMode~getWidth, gd~getDisplayMode~getHeight
      )
  stage~setScene(scene)
  scene~getStylesheets~add("file:stylesheet.css")

  stage~setFullScreen(.true)
  stage~show

::METHOD loadScene
  EXPOSE stage FXMLLoader
  USE ARG title, fileName
  stage~setTitle( title )
  url =.bsf~new("java.net.URL", fileName)
  fxml = FXMLLoader~load(url)
  stage~getScene~setRoot(fxml)

::METHOD newWindow
  EXPOSE stage windowStage FXMLLoader
  USE ARG title, fileName
  windowStage = .bsf~new("javafx.stage.Stage")
  windowStage~setTitle(title)
  url =.bsf~new("java.net.URL", fileName)
  fxml = FXMLLoader~load(url)
  scene = .bsf~new("javafx.scene.Scene", fxml)
  scene~getStylesheets~add("stylesheet.css")
  windowStage~setScene(scene)
  windowStage~initModality(bsf.loadClass("javafx.stage.Modality")~WINDOW_MODAL)
  windowStage~initOwner(stage)
  windowStage~show

::METHOD loadMainScene PUBLIC
  self~loadScene("Main", "file:main.fxml")

::METHOD showDialog
  EXPOSE stage
```

```
  USE ARG type, title, content
  Alert = bsf.import("javafx.scene.control.Alert")
  AlertType = bsf.import("javafx.scene.control.Alert$AlertType")
  IF type~upper = "WARNING" THEN DO
    alert = Alert~new(AlertType~WARNING)
  END
  ELSE IF type~upper = "ERROR" THEN DO
    alert = Alert~new(AlertType~ERROR)
  END
  ELSE DO
    alert = Alert~new(AlertType~INFORMATION)
  END
  alert~setTitle( title )
  alert~setHeaderText(.nil)
  alert~setContentText(content)
  alert~initOwner(stage)
  alertStage = alert~getDialogPane~getScene~getWindow
  alertStage~setAlwaysOnTop(.true)
  alert~showAndWait



::REQUIRES "setClasspath.rex"
::REQUIRES "DatabaseHandler.CLS"
::REQUIRES "BSF.CLS"
::REQUIRES "functions.rex"
```

Listing 52: Source Code: functions.rex

```
/* All functions and routines */

::ROUTINE startPayment PUBLIC
  USE ARG orderDetailsTable, orderTable, splitPayment = .true
  pendingcount = countPending(orderDetailsTable)
  IF pendingCount = 2 THEN splitPayment = .false -- All remaining items are getting paid next

  IF orderDetailsTable~getItems~isEmpty THEN DO
    .my.app~stageHandler~showDialog("WARNING", "WARNING", "Please choose an order")
    EXIT
  END
  ELSE IF pendingCount = 0 & splitPayment THEN DO
    .my.app~stageHandler~showDialog("WARNING", "WARNING", "Please select products")
    EXIT
  END

  /* Get business informations for receipt */
  receipt = .my.app~dbh~getBusinessInformation
```

```
NUMERIC DIGITS 20
hexSerialNr = D2X(receipt["CertSerialNumber"])
NUMERIC DIGITS


jwsHandler = .JWSHandler~new
SIGNAL ON ANY NAME privateKeyError
privateKey = jwsHandler~privateKeyFromString(receipt['PrivateKey'])
jwsHandler~encryptECDSA("Test", privateKey)
SIGNAL OFF ANY

turnoverT = .table~new
turnoverT["20"] = .array~new -- Products with 20% tax
turnoverT["10"] = .array~new  ------------ 19% tax
turnoverT["13"] = .array~new ------------ 13% tax
turnoverT["0"] = .array~new  ------------ 10% tax
turnoverT["19"] = .array~new  ------------ 0% tax

products = .array~new
productCount = .array~new
prices = .array~new
IF splitPayment THEN DO
  id = .my.app~dbh~addOrderToDB(.my.app~tableNumber)
END
ELSE id = .my.app~currentOrderID

DO item OVER orderDetailsTable~getItems
  IF item~status~get = .my.app~orderDetailsHandler~statusMessage[2] & splitPayment | \splitPayment
      THEN DO
    IF splitPayment THEN DO
      .my.app~dbh~addOrderDetailsToDB(id, item~productID~get, .my.app~orderDetailsHandler~
          statusMessage[3])
      .my.app~dbh~removeOrderDetailsFromDB(.my.app~currentOrderID, .my.app~orderDetailsHandler~
          statusMessage[2])
      .my.app~dbh~setOrderDetailsById(.my.app~currentOrderID, .my.app~orderDetailsHandler, .true)
    END
    ELSE DO
      .my.app~dbh~setStatusInDB(.my.app~orderDetailsHandler~statusMessage[3], item~tempID)
    END
    tax = .my.app~dbh~getTaxByProductID(item~productID~get)
    price = item~productPrice~get

    /* Count product frequency */
    name = item~productName~get
    IF products~hasItem(name) THEN DO
      productCount[products~index(name)] += 1
```

```
    END
    ELSE DO
      products~append(name)
      productCount~append(1)
      prices~append(price)
    END

    /* Sort products by tax brackets */
    SELECT
      WHEN tax = 0.2 THEN turnoverT["20"]~append(price)
      WHEN tax = 0.19 THEN turnoverT["19"]~append(price)
      WHEN tax = 0.13 THEN turnoverT["13"]~append(price)
      WHEN tax = 0.1 THEN turnoverT["10"]~append(price)
      WHEN tax = 0 THEN turnoverT["0"]~append(price)
      OTHERWISE -- PASS
    END
    orderDetailsTable~getItems~remove(item)
  END
END
IF .my.app~dbh~orderEmpty(.my.app~currentOrderID) & splitPayment THEN DO
  .my.app~dbh~removeOrderFromDB(.my.app~currentOrderID)
  .my.app~orderHandler~orderArrayList = getOpenOrderArrayList()
  .my.app~orderHandler~setItems
END
IF splitPayment THEN DO
  IF .my.app~dbh~orderEmpty(.my.app~currentOrderID) THEN DO
    .my.app~dbh~removeOrderFromDB(.my.app~currentOrderID)
    .my.app~orderHandler~orderArrayList = getOpenOrderArrayList()
    .my.app~orderHandler~setItems
    orderDetailsTable~getItems~clear
  END
END
ELSE DO
  orderDetailsTable~getItems~clear
  orderTable~getItems~remove(orderTable~getSelectionModel~getSelectedItem)
END
date = .my.app~dbh~insertDateOfPayment(id)

  /* Create itemStringArray for later use on receipt */
itemStringArray = .array~new
DO i=1 TO products~items
  itemStringArray~append(.array~of(productCount[i] products[i], format(prices[i],,2), format(prices[i]*
      productCount[i],,2)))
END

receipt['Items'] = itemStringArray
datetime = CHANGESTR(" ", date, "T") -- Reformat date
```

```
sumTable = .table~new
formattedSumTable = .table~new

sum = 0
DO key OVER turnoverT
  singleSum = sum(turnoverT[key])
  sumTable[key] = singleSum
  formattedSumTable[key] = reformatFloat(singleSum)
  sum += singleSum
END

receipt['OrderID'] = id
receipt['Datetime'] = datetime
receipt['TableNumber'] = .my.app~tableNumber
receipt['Sum'] = sum
receipt['SumTable'] = sumTable
receipt['UserName'] = .my.app~activeUser~name
receipt['UserSurname'] = .my.app~activeUser~surname




oldJWS = .my.app~dbh~getLastOrderJWS
payload = oldJWS[1]
signature = oldJWS[2]
oldQRString = jwsHandler~restoreQRString(payload, signature)
hashLastReceipt = jwsHandler~hashLastReceipt(oldQRString)

taxOrder = .array~of("20", "10", "13", "0", "19")

/* Encrypt turnover */
aes = .AES256~new
turnover = .my.app~dbh~getTurnover
turnover = format(turnover*100,,0)
SIGNAL ON ANY NAME encryptionError
turnoverEncoded = aes~encrypt(turnover, receipt["AES256Key"])
SIGNAL OFF ANY

/* Concatenate QR - String */
qrText = .mutableBuffer~new
qrText~append("_R1-AT_" || receipt["CashRegisterID"] || "_" || id || "_" || datetime)
DO i OVER taxOrder
  qrText~append("_" || formattedSumTable[i])
END
qrText~append("_" || turnoverEncoded || "_" || hexSerialNr || "_" || hashLastReceipt)
```

```
  jws = jwsHandler~encryptECDSA(qrText~string, privateKey)

  PARSE VAR jws header "." payload "." signature .


  signatureB64 = jwsHandler~recode(signature)

  qrBuffer = qrText~append("_" || signatureB64)
  qrString = qrBuffer~string

  .my.app~dbh~storeJWS(id, header, payload, signature)

  /* Create QR - Code */
  CALL createQRCode qrString

  .my.app~lastReceipt = receipt
  CALL printReceipt receipt
  EXIT

encryptionError:
  .my.app~stageHandler~showDialog("ERROR", "Encryption error", "Invalid AES key. Use generateKeys.rex
      script to generate a new key.")
  EXIT -1

privateKeyError:
  .my.app~stageHandler~showDialog("ERROR", "Encryption error", "Invalid private key. Please verify the
      entered private key.")
  EXIT -1

::ROUTINE createQRCode
  USE ARG text, path="res/qr.png"
  qrWriter = .QRWriter~new
  qrWriter~write(path, text)


::ROUTINE appendSums PUBLIC
  USE ARG text, sumTable
  DO key OVER sumTable
    percentage = key || ".0%"
    value = sumTable[key]
    IF key = "0" THEN DO
      tax = format2f(value)
      sum = format2f(value)
    END
    ELSE DO
      tax = format2f(value/(key/100+1))
      sum = format2f(value-(value/(key/100+1)))
```

```
    END
    IF value > 0 THEN DO
      text~append("<tr><td>" || percentage || "</td><td>" || tax || "</td><td>" || sum || "</td><td>"
          || format(sumTable[key],,2) || "</td></tr>")
    END
  END
  RETURN text

::ROUTINE format2f
  USE ARG float
  RETURN format(float,,2)

/* E.g. Input: 20.014010
    Output: 20,01 */
::ROUTINE reformatFloat
  USE ARG float
  RETURN translate(format2f(float), ",", ".")

::ROUTINE printReceipt PUBLIC
  USE ARG receipt

    datetime = receipt['Datetime']
  /* Split date and time */
  PARSE VAR datetime date "T" time .
  /* Reformat date from YYYY-mm-dd TO dd.mm.YYYY */
  PARSE VAR date year "-" month "-" day .
  date = day || "." || month || "." || year

  /* Create mutableBuffer to store HTML string for receipt printing */
  text = .mutableBuffer~new
  text~append("<!DOCTYPE html>" -
        "<html>" -
          "<body>" -
            "<h1>" || receipt['Name'] || "</h1>" -
              "<span>" || receipt['Postcode'] receipt['City'] || "<br>" -
               || receipt["Address"] || "<br>" -
                "Tel.: " || receipt["PhoneNumber"] || "<br>" -
                receipt["Email"] || "<br>" -
                "UId-Nr.: " || receipt["VatID"] || "<br>" -
                "Rechnung" receipt['OrderID'] || "<br>" -
                || "Tisch" receipt['TableNumber'] || "<br>" -
                || date || "<br>" -
                || time || "<br>" -
              "----------<br>" -
              "<table>")

  DO item OVER receipt['Items']
```

```
    text~append("<tr>")
    DO i OVER item
      text~append("<td>" || i || "</td>")
    END
    text~append("</tr>")
  END
  text~append("</table>")
  text~append("------------------------------<br>" -
        "<h2>Rechnungsbetrag <br>EURO" receipt['Sum'] || " </h2>" -
        "==============================<br>" -
        "<table>" -
        "<tr>" -
          "<th>MWST</th>"-
          "<th>NETTO</th>"-
          "<th>STEUER</th>"-
          "<th>BRUTTO</th>"-
        "</tr>")

  text = appendSums(text, receipt['SumTable'])

  filePath = "file:" || .my.app~homeDir || "/res/qr.png"

  text~append("</table><br>"-
        '<img src="' || filePath || '" width="100" height="100"><br>' -
        "Es bediente Sie " || left(receipt['UserName'], 1) || "." -
        receipt['UserSurname'] || "<br>" -
        "Wir freuen uns auf ein baldiges Wiedersehen <br>"-
        "</span></body></html>")

  /* Minimize Program so print window is visible*/
  .my.app~stageHandler~stage~getScene~getWindow~setIconified(.true)
  done = print(text~string)
  .my.app~stageHandler~stage~getScene~getWindow~setIconified(.false)
  IF done THEN DO
    SAY "Printing completed."
  END
  ELSE DO
    .my.app~stageHandler~showDialog("ERROR", "ERROR", "Error printing receipt")
  END

::ROUTINE print
  USE ARG string

  jEditorPane = .bsf~new("javax.swing.JEditorPane")
  kit = .bsf~new("javax.swing.text.html.HTMLEditorKit")

  /* Add css to HTML */
```

```
  jEditorPane~setEditorKit(kit)

  /* Define stylesheet */
  styleSheet = kit~getStyleSheet
  styleSheet~addRule("body {font-size: 7px;}")
  styleSheet~addRule("h1 {font-size: 13px;}")
  styleSheet~addRule("h2 {font-size: 10px;}")

  doc = kit~createDefaultDocument
  jEditorPane~setDocument(doc)
  jEditorPane~setText(string)
  RETURN jEditorPane~print


/* Return sum of array */
::ROUTINE sum
  USE ARG arr
  sum = 0
  DO i OVER arr
    sum += i
  END
  RETURN sum

::ROUTINE itemSelected PUBLIC
  USE ARG table
  item = table~getSelectionModel~getSelectedItem
  IF item <> .nil THEN RETURN .true
  ELSE RETURN .false

/* Count pending payments
Returns 2: all payments of selected order are pending
    1: atleast one pending payment but not all
    0: no pending payment
*/
::ROUTINE countPending PUBLIC -- Check if products for next payment are selected
  USE ARG orderDetailsTable
  pendingCount = 0
  itemCount = 0
  DO item OVER orderDetailsTable~getItems
      IF item~status~get = .my.app~orderDetailsHandler~statusMessage[2] THEN DO
        pendingCount += 1
      END
      itemCount +=1
  END
  IF pendingCount = 0 THEN RETURN 0
  ELSE IF pendingCount = itemCount THEN RETURN 2
  ELSE RETURN 1
```

```
::ROUTINE incrementQuantity PUBLIC
  USE ARG item
  SimpleIntegerProperty = bsf.import("javafx.beans.property.SimpleIntegerProperty")
  sum = item~quantity~add(SimpleIntegerProperty~new(1))
  item~quantity~set(sum~getValue)

::ROUTINE findProductInOrder PUBLIC
  USE ARG orderDetails
  DO item OVER .my.app~orderDetailsHandler~table~getItems
    IF item~productID~get == orderDetails~productID~get THEN DO
      RETURN item
    END
  END
  RETURN .nil

::ROUTINE addOrderDetails PUBLIC
  USE ARG orderDetails
  .my.app~orderDetailsHandler~addItem(orderDetails)

::ROUTINE loadFirstFXML PUBLIC
  /* Check if root user exits, if not open FXML for creating a root user */
  IF \.my.app~dbh~bizInfExists() THEN DO
    .my.app~stageHandler~newWindow("Add business informations", "file:edit_business_information.fxml")
  END
  ELSE IF \.my.app~dbh~rootUserExists THEN DO -- No root user found, Load root user account
      creation
    .my.app~addRoot = .true -- Next user added is going to have root access
    .my.app~stageHandler~loadScene("Add Root User", "file:add_user.fxml")
  END
  ELSE DO --Root user exists, Load user login
    .my.app~stageHandler~loadScene("User Login", "file:user_login.fxml")
  END

::ROUTINE quitApp PUBLIC
  bsf.loadClass("javafx.application.Platform")~exit

::REQUIRES "BSF.CLS"
::REQUIRES "TableHandler.CLS"
::REQUIRES "PasswordHasher.CLS"
::REQUIRES "AES256.CLS"
::REQUIRES "QRWriter.CLS"
::REQUIRES "JWSHandler.CLS"
```

Listing 53: Source Code: QRWriter.CLS

```
/**
```

```
    * JWSHandler Class creates 300x300 QR Codes
*/

::CLASS QRWriter PUBLIC
::METHOD BarcodeFormat ATTRIBUTE
::METHOD BitMatrix ATTRIBUTE
::METHOD QRCodeWriter ATTRIBUTE
::METHOD MatrixToImageWriter ATTRIBUTE
::METHOD INIT
  EXPOSE BarcodeFormat BitMatrix QRCodeWriter MatrixToImageWriter

  BarcodeFormat = bsf.import("com.google.zxing.BarcodeFormat")
  BitMatrix = bsf.import("com.google.zxing.common.BitMatrix")
  QRCodeWriter = bsf.import("com.google.zxing.qrcode.QRCodeWriter")
  MatrixToImageWriter = bsf.import("com.google.zxing.client.j2se.MatrixToImageWriter")

::METHOD write
  EXPOSE BarcodeFormat BitMatrix QRCodeWriter MatrixToImageWriter
  USE ARG path, qrString

  path = .bsf~new("java.io.File", path)

  writer = QRCodeWriter~new
  bitMatrix = writer~encode(qrString, BarcodeFormat~QR_CODE, 300, 300)
  MatrixToImageWriter~writeToFile(bitMatrix, "PNG", path)



::REQUIRES "BSF.CLS"
```

Listing 54: Source Code: controller.rex

```
::ROUTINE submitAcessRightChanges PUBLIC
  /*@get(accessRightGroup)*/
  IF accessRightGroup~getSelectedToggle = .nil THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Please choose access right")
  END
  ELSE DO
    accessRight = accessRightGroup~getToggles~indexOf(accessRightGroup~getSelectedToggle)+1
    SIGNAL ON ANY NAME updateError
    .my.app~dbh~updateAccessRight(.my.app~selectedUsername, accessRight)
    SIGNAL OFF ANY
    .my.app~stageHandler~windowStage~close
    userArrayList = .my.app~dbh~getUserArrayList
    .my.app~userHandler~updateArrayList(userArrayList)
  END
  EXIT 0
```

```
updateError:
  .my.app~stageHandler~showDialog("ERROR", "ERROR", "Database update error")
  RETURN

::ROUTINE returnToMain PUBLIC
  .my.app~stageHandler~loadMainScene


::ROUTINE submitDatabaseLogin PUBLIC
  USE ARG slotDir
  /* Start connecting to mysql */
  .my.app~dbh~initSettings("conf.ini")
  success = .my.app~dbh~connect
  IF success THEN DO
    CALL loadFirstFXML
  END
  ELSE DO
    .my.app~stageHandler~showDialog("ERROR", "Connection error", "Connecting to database failed." "0A"
        X "Try:" "0A"X "Run setup.rex" "0A"X "Verify configurations in conf.ini" "0A"X "Make sure
        JConnector jar is added to classpath")
  END
  EXIT 0


::ROUTINE submitUserLogin PUBLIC
  USE ARG slotDir
  /*@get(username password successText)*/
  IF .my.app~dbh~passwordCorrect(username~text, password~text) THEN DO
    IF \.my.app~dbh~accountActive(username~text) THEN DO
      .my.app~stageHandler~showDialog("error", "ERROR", "Account deactivated. Please contact root user."
          )
      EXIT 0
    END
    successText~opacity = 1

    CALL SYSSLEEP 0.4
    .my.app~activeUser = .my.app~dbh~loadUserInformation(username~text)
    .my.app~dbh~insertLastLoginDate(.my.app~activeUser~id)
    SIGNAL ON ANY
    .my.app~stageHandler~loadMainScene
    SIGNAL OFF ANY
  END
  ELSE DO
    .my.app~stageHandler~showDialog("error", "ERROR", "Wrong username or password")
  END
  EXIT 0
```

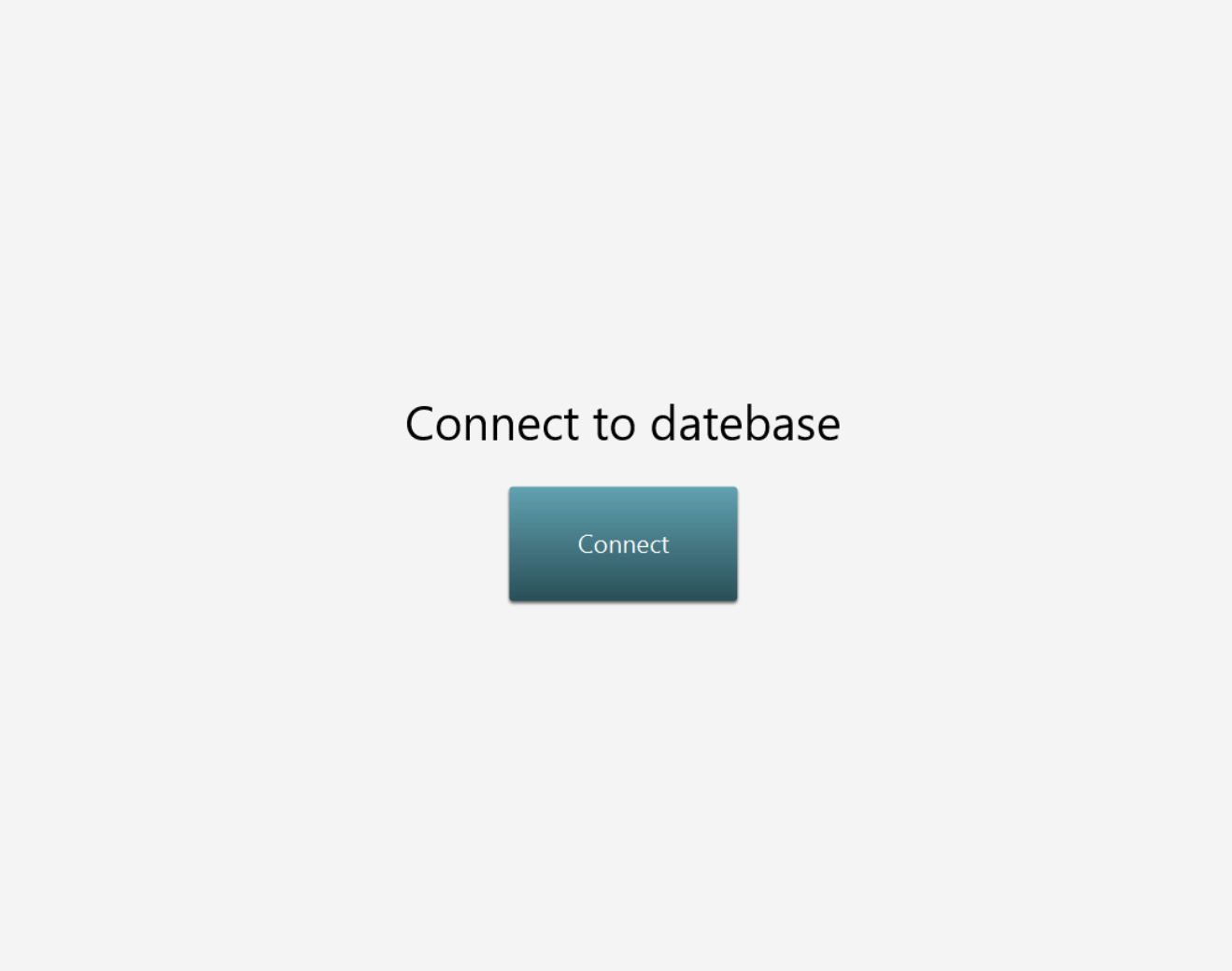
```
ANY:
  SAY "[-] ERROR!"
  SAY "[-] Make sure CLASSPATH for jfoenix-8.0.7.jar is set."
  .my.app~stageHandler~showDialog("error", "ERROR", "Error loading main page. Make sure CLASSPATH
      for jfoenix-8.0.7.jar is set.")
  SAY "[-] Quitting."
  CALL quitApp

::REQUIRES "BSF.CLS"
::REQUIRES "functions.rex"
::REQUIRES "TableHandler.CLS"
::REQUIRES "DatabaseHandler.CLS"
```

Listing 55: Source Code: db_login.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.Group?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.StackPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0" xmlns="http://
    javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1">
    <fx:script source="controller.rex" />
    <children>
        <StackPane AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor
            ="0.0" AnchorPane.topAnchor="0.0">
            <children>
                <Group StackPane.alignment="CENTER">
                    <children>
                        <VBox alignment="CENTER" spacing="30.0">
                            <children>
                                <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Connect to
                                    datebase" textAlignment="CENTER">
                                    <font>
                                        <Font size="42.0" />
                                    </font>
                                </Text>
                                <Button fx:id="submitButton" contentDisplay="CENTER" mnemonicParsing
                                    ="false" onAction="slotDir=arg(arg()); call submitDatabaseLogin slotDir;
                                    " text="Connect">
                                    <font>
                                        <Font size="36.0" />
```

```
                        </font>
                    </Button>
                    <Text fx:id="errorText" fill="red" opacity="0.0" strokeType="OUTSIDE"
                        strokeWidth="0.0" text="Failed to establish connection" textAlignment="
                        CENTER" wrappingWidth="170.45703125" />
                    <Text fx:id="successText" fill="green" opacity="0.0" strokeType="
                        OUTSIDE" strokeWidth="0.0" text="Connection succesful."
                        textAlignment="CENTER" wrappingWidth="170.45703125" />
                </children>
            </VBox>
        </children>
    </Group>
</children>
</StackPane>
</children>
</AnchorPane>
```

Figure 32: Screenshot: db_login.fxml



After connecting to the database the user is asked to add business information and to create a new user account. This is only necessary once. The business information are added with *edit_business_information.fxml* (Listing 57, Figure 33) and *edit_biz_inf_controller.rex* (Listing 56). The new user account is created with *add_user.fxml* (Listing 59, Figure 34) and *add_user_controller.rex* (Listing 58). After creating and adding the relevent information or if they exist already the user will be asked to login. This is done with *user_login.fxml* (Listing 60, Figure 35) and *controller.rex* (Listing 54)

Listing 56: Source Code: edit_biz_inf_controller.rex

```
/*@get(defaultButton nameTF postcodeTF cityTF addressTF numberTF emailTF vatIdTF cashRegTF
    certSerialTF aes256TF privKeyTF)*/
IF .my.app~dbh~bizInfExists THEN DO
  bizInf = .my.app~dbh~getBusinessInformation
  nameTF~text = bizInf['Name']
  postcodeTF~text = bizInf['Postcode']
  cityTF~text = bizInf['City']
  addressTF~text = bizInf['Address']
  numberTF~text = bizInf['PhoneNumber']
  emailTF~text = bizInf['Email']
  vatIdTF~text = bizInf['VatID']
  cashRegTF~text = bizInf['CashRegisterID']
  certSerialTF~text = bizInf['CertSerialNumber']
  aes256TF~text = bizInf['AES256Key']
  privKeyTF~text = bizInf['PrivateKey']
  defaultButton~disable = .true
  defaultButton~opacity = 0
END

::ROUTINE fillDefault PUBLIC
  /*@get(nameTF postcodeTF cityTF addressTF numberTF emailTF vatIdTF cashRegTF certSerialTF
      aes256TF privKeyTF)*/
  nameTF~text = "Test Company"
  postcodeTF~text = "1000"
  cityTF~text = "Vienna"
  addressTF~text = "Companystreet 10"
  numberTF~text = "+43 1000 000"
  emailTF~text = "company@email.com"
  vatIdTF~text = "ATU1000000"
  cashRegTF~text = "REGISTRIERKASSE1"
  certSerialTF~text = "21483750523945"
  aes256TF~text = "bNAnv0+/MtGMBfYJGhLv70wWtg431Df1iMX923D7tt0="
  privKeyTF~text = "MEECAQAwEwYHKoZIzj0CAQYIKoZIzj0DAQcEJzAlAgEBBCBcRhqsftzHbOkX/
      DmxI1gUUBi/YwVOo2Cg/rp5rBv//w=="


::ROUTINE submitChange PUBLIC
```

```
  USE ARG slotDir
  /*@get(nameTF postcodeTF cityTF addressTF numberTF emailTF vatIdTF cashRegTF certSerialTF
      aes256TF privKeyTF)*/
  name = nameTF~text
  postcode = postcodeTF~text
  city  = cityTF~text
  address = addressTF~text
  number = numberTF~text
  email = emailTF~text
  vatID = vatIdTF~text
  cashReg = cashRegTF~text
  certSerial  = certSerialTF~text
  aes256 = aes256TF~text
  priv  = privKeyTF~text

  IF name~strip = "" | postcode~strip = "" | city~strip = "" | address~strip = "" | number~strip = "" |
      email~strip = "" | vatID~strip = "" | cashReg~strip = "" | certSerial~strip = "" | aes256~strip = "" |
      priv~strip = "" THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Please fill in all inputs")
    EXIT 0
  END
  IF DATATYPE(certSerial~strip) <> "NUM" THEN CALL badValue
  ELSE IF certSerial~strip~length > 18 THEN CALL sizeError

  IF .my.app~dbh~bizInfExists THEN .my.app~dbh~removeBizInf -- Remove old business informations
  SIGNAL ON ANY NAME insertError
  .my.app~dbh~addBusinessInformation(name, postcode, city, address, number, email, vatID, cashReg,
      certSerial, aes256, priv)
  SIGNAL OFF ANY
  .my.app~stageHandler~windowStage~close
  IF \.my.app~dbh~rootUserExists THEN DO -- No root user found, Load root user account creation
      .my.app~addRoot = .true -- Next user added is going to have root access
      .my.app~stageHandler~loadScene("Add Root User", "file:add_user.fxml")
    END
  EXIT

insertError:
  .my.app~stageHandler~showDialog("ERROR", "Form error", "Please verify inputs.")
  EXIT

badValue:
  .my.app~stageHandler~showDialog("ERROR", "Form error", "The certifcation serial number should be in
      decimal format.")
  EXIT
sizeError:
  .my.app~stageHandler~showDialog("ERROR", "Form error", "The certifcation serial number is too long.")
  EXIT
```

```
:: REQUIRES "functions.rex"
:: REQUIRES "BSF.CLS"
```

Listing 57: Source Code: edit_business_information.fxml

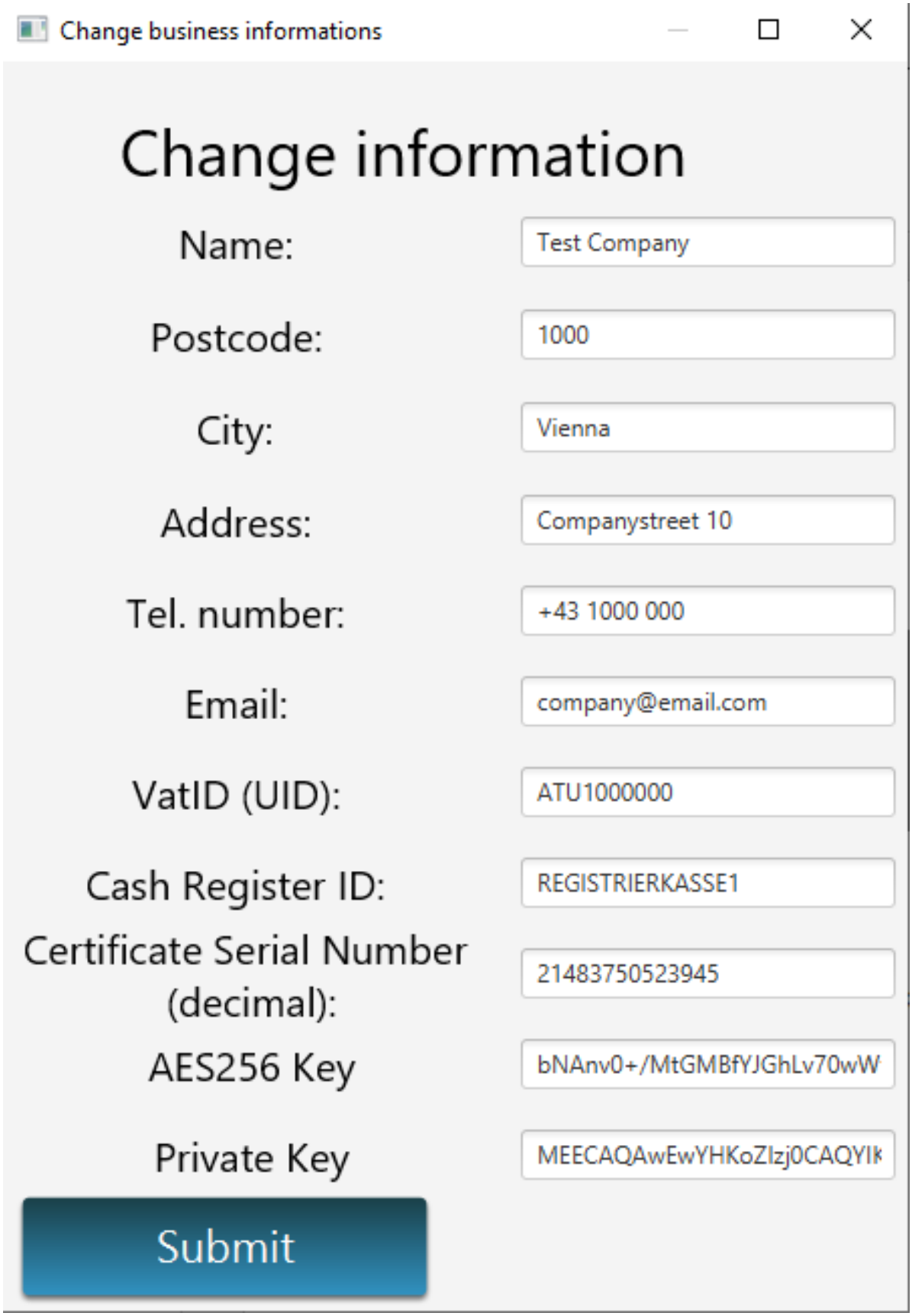
```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<VBox prefHeight="620.0" prefWidth="450.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://
    javafx.com/fxml/1">
    <children>
        <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="620.0" prefWidth="-1.0" VBox.
            vgrow="ALWAYS">
            <children>
                <GridPane layoutX="11.0" layoutY="20.0" prefHeight="590.0" prefWidth="433.0">
                    <columnConstraints>
                        <ColumnConstraints hgrow="SOMETIMES" maxWidth="308.0" minWidth="10.0"
                            prefWidth="224.0" />
                        <ColumnConstraints hgrow="SOMETIMES" maxWidth="197.0" minWidth="10.0"
                            prefWidth="163.0" />
                    </columnConstraints>
                    <rowConstraints>
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                    </rowConstraints>
                    <children>
                        <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Change information"
```

```
        textAlignment="CENTER" wrappingWidth="378.13671875" GridPane.
        columnSpan="2">
        <font>
            <Font size="32.0" />
        </font>
    </Text>
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Name:" textAlignment="
        CENTER" wrappingWidth="210.78515625" GridPane.rowIndex="1">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Postcode:" textAlignment=
        "CENTER" wrappingWidth="210.78515625" GridPane.rowIndex="2">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="City:" textAlignment="
        CENTER" wrappingWidth="210.78515625" GridPane.rowIndex="3">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call submitChange
        slotDir;" prefHeight="38.0" prefWidth="198.0" text="Submit" GridPane.
        columnSpan="2" GridPane.rowIndex="12">
        <font>
            <Font size="20.0" />
        </font>
    </Button>
    <TextField fx:id="nameTF" promptText="e.g. Testcompany" GridPane.columnIndex
        ="1" GridPane.rowIndex="1" />
    <TextField fx:id="postcodeTF" promptText="e.g. 1010" GridPane.columnIndex="1"
        GridPane.rowIndex="2" />
    <TextField fx:id="cityTF" promptText="e.g. Wien" GridPane.columnIndex="1"
        GridPane.rowIndex="3" />
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Tel. number:"
        textAlignment="CENTER" wrappingWidth="210.78515625" GridPane.rowIndex=
        "5">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <TextField fx:id="addressTF" promptText="e.g. Companystreet 13" GridPane.
        columnIndex="1" GridPane.rowIndex="4" />
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Address:" textAlignment="
```

```
        CENTER" wrappingWidth="210.78515625" GridPane.rowIndex="4">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <TextField fx:id="numberTF" promptText="e.g. +43 1234 20 20" GridPane.
        columnIndex="1" GridPane.rowIndex="5" />
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Email:" textAlignment="
        CENTER" wrappingWidth="210.78515625" GridPane.rowIndex="6">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <TextField fx:id="emailTF" promptText="e.g. name@company.com" GridPane.
        columnIndex="1" GridPane.rowIndex="6" />
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="VatID (UID):"
        textAlignment="CENTER" wrappingWidth="210.78515625" GridPane.rowIndex=
        "7">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <TextField fx:id="vatIdTF" promptText="e.g. ATU1231313" GridPane.columnIndex=
        "1" GridPane.rowIndex="7" />
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Cash Register ID:"
        textAlignment="CENTER" wrappingWidth="210.78515625" GridPane.rowIndex=
        "8">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Certificate Serial Number (
        decimal):" textAlignment="CENTER" wrappingWidth="226.78515625" GridPane.
        rowIndex="9">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <TextField fx:id="cashRegTF" promptText="e.g. REGISTRIERKASSE1" GridPane.
        columnIndex="1" GridPane.rowIndex="8" />
    <TextField fx:id="certSerialTF" promptText="e.g. 21483750523945" GridPane.
        columnIndex="1" GridPane.rowIndex="9" />
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="AES256 Key"
        textAlignment="CENTER" wrappingWidth="226.78515625" GridPane.rowIndex=
        "10">
        <font>
            <Font size="20.0" />
```

```
                        </font>
                    </Text>
                    <TextField fx:id="aes256TF" prefWidth="205.0" promptText="e.g. y5h+od/
                        ulMm1K3GqIsX4OIJ8uCdFT0u5ULFI2X2i7q8=" GridPane.columnIndex="1"
                        GridPane.rowIndex="10" />
                    <Button fx:id="defaultButton" mnemonicParsing="false" onAction="slotDir=arg(arg
                        ()); call fillDefault slotDir;" prefHeight="38.0" prefWidth="198.0" text="Fill with
                         default" GridPane.columnIndex="1" GridPane.rowIndex="12">
                        <font>
                            <Font size="20.0" />
                        </font>
                    </Button>
                    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Private Key"
                        textAlignment="CENTER" wrappingWidth="226.78515625" GridPane.rowIndex=
                        "11">
                        <font>
                            <Font size="20.0" />
                        </font>
                    </Text>
                    <TextField fx:id="privKeyTF" prefWidth="205.0" promptText="e.g.
                        MEECAQAwEwYHKoZIzj0CAQYIKoZIzj0DAQcEJzAlAgEBBCBcRhqsftzHbOkX
                        /DmxI1gUUBi/YwVOo2Cg/rp5rBv//w==" GridPane.columnIndex="1" GridPane
                        .rowIndex="11" />
                </children>
            </GridPane>
        </children>
        <fx:script source=" edit_biz_inf_controller .rex" />
    </AnchorPane>
  </children>
</VBox>
```

Figure 33: Screenshot: edit_business_information.fxml



Listing 58: Source Code: add_user_controller.rex

```
/*@get(accessRightGroup rootRadioButton backToMainButton)*/
-- Disable access right choice on adding root user
-- First user created has to be root user
IF .my.app~addRoot THEN DO
  backToMainButton~visible = .false
  rootRadioButton~setSelected(.true)
  DO node OVER accessRightGroup~getToggles
    node~disable = .true
  END
END
IF .my.app~accessRight <> .nil THEN DO
  IF .my.app~accessRights < 4 THEN DO -- User who is not root cannot create a new root user
    rootAccessNode = accessRightGroup~getToggles~get(3)
    rootAccessNode~disable = .true
  END
END

::ROUTINE addUser PUBLIC
  USE ARG slotDir
  /*@get(addUserButton username password password2 name surname successText accessRightGroup
      backToMainButton)*/

  IF password~text <> password2~text THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Password does not match")
  END
  ELSE IF length(username~text) > 20 THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Username too long (max 20 chars)")
  END
  ELSE IF length(name~text) > 20 THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Name too long (max 20 chars)")
  END
  ELSE IF length(surname~Text) > 20 THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Surname too long (max 20 chars)")
  END
  ELSE IF username~text~strip = "" | password~text~strip = "" | name~text~strip = "" | surname~Text~
      strip = "" THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Please fill in all inputs")
  END
  ELSE IF accessRightGroup~getSelectedToggle = .nil THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Please choose access right")
  END
  ELSE IF .my.app~dbh~usernameExists(username~text) THEN DO
    .my.app~stageHandler~showDialog("ERROR", "Form error", "Username already exists")
  END
  ELSE DO
```

```
    accessRight = accessRightGroup~getToggles~indexOf(accessRightGroup~getSelectedToggle)+1
    SIGNAL ON ANY NAME databaseError
    .my.app~dbh~addUserToDB(username~text, password~text, name~text, surname~Text, accessRight)
    SIGNAL OFF ANY
    addUserButton~disable = .true
    successText~visible = .true
    CALL SYSSLEEP 0.5
    IF \backToMainButton~visible THEN DO -- backToMainButton is not visible = currently adding first
        root user
      .my.app~stageHandler~loadScene("User Login", "file:user_login.fxml")
    END
    ELSE DO
      .my.app~stageHandler~loadMainScene
      .my.app~remove(addRoot)
    END
  END
  EXIT 0

databaseError:
  .my.app~stageHandler~showDialog("ERROR", "MySQL Insert Error", "MySQL database insert error")
  RETURN



::REQUIRES "functions.rex"
::REQUIRES "controller.rex"
```

Listing 59: Source Code: add_user.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.Group?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.StackPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<BorderPane prefHeight="200.0" prefWidth="200.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="
```

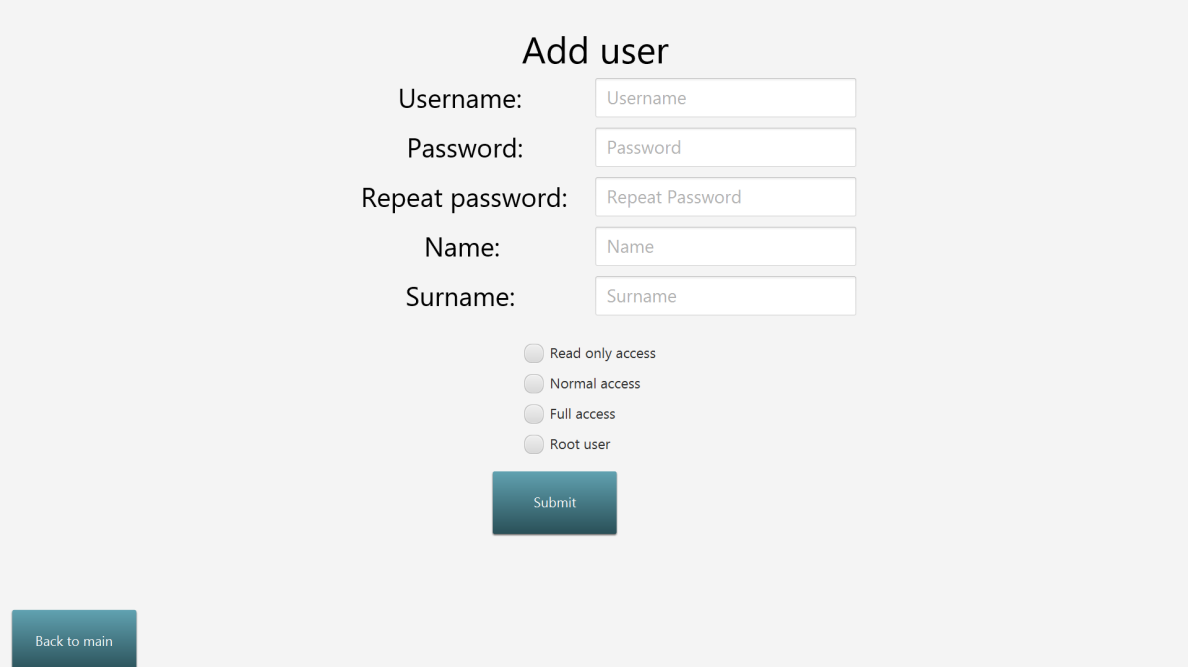
```
http://javafx.com/fxml/1">
<bottom>
    <Button fx:id="backToMainButton" mnemonicParsing="false" onAction="slotDir=arg(arg()); call
        returnToMain slotDir;" text="Back to main" BorderPane.alignment="CENTER_LEFT">
        <font>
            <Font size="31.0" />
        </font>
        <BorderPane.margin>
            <Insets bottom="20.0" left="20.0" />
        </BorderPane.margin>
    </Button>
</bottom>
<center>
    <VBox prefHeight="1080.0" prefWidth="1920.0" BorderPane.alignment="CENTER">
        <children>
            <AnchorPane maxHeight="-1.0" maxWidth="-1.0">
                <children>
                    <Text layoutX="843.0" layoutY="104.0" strokeType="OUTSIDE" strokeWidth="0.0
                        " text="Add user" textAlignment="CENTER">
                        <font>
                            <Font size="60.0" />
                        </font>
                    </Text>
                    <Button fx:id="addUserButton" layoutX="796.0" layoutY="763.0" mnemonicParsing
                        ="false" onAction="slotDir=arg(arg()); call addUser slotDir;" prefHeight="57.0"
                        prefWidth="331.0" text="Submit" textAlignment="CENTER">
                        <font>
                            <Font size="40.0" />
                        </font>
                    </Button>
                    <Text fx:id="successText" fill="GREEN" layoutX="1166.0" layoutY="979.0"
                        strokeType="OUTSIDE" strokeWidth="0.0" text="Success." textAlignment="
                        CENTER" visible="false">
                        <font>
                            <Font size="30.0" />
                        </font>
                    </Text>
                    <GridPane alignment="CENTER" layoutX="541.0" layoutY="144.0" prefWidth="
                        842.0" vgap="50.0">
                        <columnConstraints>
                            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="
                                100.0" />
                            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="
                                100.0" />
                        </columnConstraints>
                        <rowConstraints>
                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
```

```
            " />
        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
            " />
        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
            " />
        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
            " />
        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
            " />
    </rowConstraints>
    <children>
        <TextField fx:id="username" prefHeight="63.0" prefWidth="0.0" promptText
            ="Username" GridPane.columnIndex="1">
            <font>
                <Font size="29.0" />
            </font>
        </TextField>
        <PasswordField fx:id="password" promptText="Password" GridPane.
            columnIndex="1" GridPane.rowIndex="1">
            <font>
                <Font size="29.0" />
            </font>
        </PasswordField>
        <PasswordField fx:id="password2" promptText="Repeat Password" GridPane
            .columnIndex="1" GridPane.rowIndex="2">
            <font>
                <Font size="29.0" />
            </font>
        </PasswordField>
        <TextField fx:id="name" promptText="Name" GridPane.columnIndex="1"
            GridPane.rowIndex="3">
            <font>
                <Font size="29.0" />
            </font>
        </TextField>
        <TextField fx:id="surname" promptText="Surname" GridPane.columnIndex=
            "1" GridPane.rowIndex="4">
            <font>
                <Font size="29.0" />
            </font>
        </TextField>
        <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Username: "
            textAlignment="CENTER" wrappingWidth="416.1796875">
            <font>
                <Font size="43.0" />
            </font>
        </Text>
```

```
            <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Password:"
                textAlignment="CENTER" wrappingWidth="420.1796875" GridPane.
                rowIndex="1">
                <font>
                    <Font size="43.0" />
                </font>
            </Text>
            <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Name: "
                textAlignment="CENTER" wrappingWidth="422.8463134765625"
                GridPane.rowIndex="3">
                <font>
                    <Font size="43.0" />
                </font>
            </Text>
            <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Surname: "
                textAlignment="CENTER" wrappingWidth="416.1796875" GridPane.
                rowIndex="4">
                <font>
                    <Font size="43.0" />
                </font>
            </Text>
            <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Repeat password:"
                textAlignment="CENTER" wrappingWidth="417.5130615234375"
                GridPane.rowIndex="2">
                <font>
                    <Font size="43.0" />
                </font>
            </Text>
        </children>
    </GridPane>
    <GridPane layoutX="847.0" layoutY="547.0" prefHeight="245.0" prefWidth="229.0"
        >
        <columnConstraints>
            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="
                100.0" />
        </columnConstraints>
        <rowConstraints>
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                " />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                " />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                " />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                " />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                " />
```

```
                        </rowConstraints>
                        <children>
                            <RadioButton mnemonicParsing="false" text="Read only access">
                                <font>
                                    <Font size="23.0" />
                                </font>
                                <toggleGroup>
                                    <ToggleGroup fx:id="accessRightGroup" />
                                </toggleGroup>
                            </RadioButton>
                            <RadioButton mnemonicParsing="false" text="Normal access" toggleGroup=
                                "$accessRightGroup" GridPane.rowIndex="1">
                                <font>
                                    <Font size="23.0" />
                                </font>
                            </RadioButton>
                            <RadioButton mnemonicParsing="false" text="Full access" toggleGroup="
                                $accessRightGroup" GridPane.rowIndex="2">
                                <font>
                                    <Font size="23.0" />
                                </font>
                            </RadioButton>
                            <RadioButton fx:id="rootRadioButton" mnemonicParsing="false" text="
                                Root user" toggleGroup="$accessRightGroup" GridPane.rowIndex="3">
                                <font>
                                    <Font size="23.0" />
                                </font>
                            </RadioButton>
                        </children>
                    </GridPane>
                    <StackPane prefHeight="150.0" prefWidth="200.0">
                        <children>
                            <Group />
                        </children>
                    </StackPane>
                </children>
            </AnchorPane>
        </children>
    </VBox>
  </center>
  <fx:script source="add_user_controller.rex" />
</BorderPane>
```

Figure 34: Screenshot: add_user.fxml

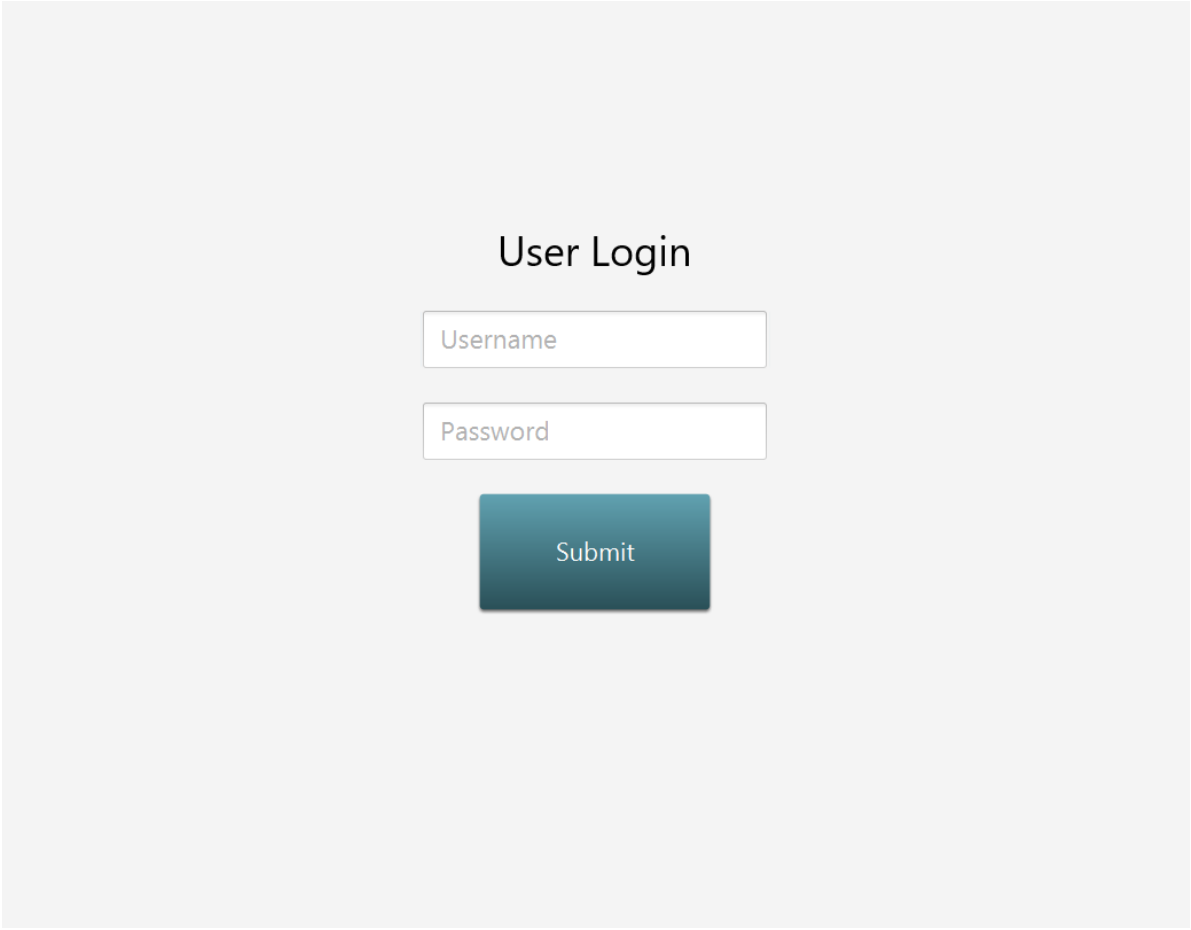


Listing 60: Source Code: user_login.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.Group?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.StackPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0" xmlns="http://
    javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <StackPane AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor
            ="0.0" AnchorPane.topAnchor="0.0">
            <children>
                <Group StackPane.alignment="CENTER">
                    <children>
                        <VBox alignment="CENTER" spacing="30.0">
                            <children>
                                <Text strokeType="OUTSIDE" strokeWidth="0.0" text="User Login"
                                    textAlignment="CENTER">
                                    <font>
                                        <Font size="36.0" />
                                    </font>
```

```
                        </Text>
                        <TextField fx:id="username" prefWidth="300.0" promptText="Username"
                            styleClass="textInput"/>
                        <PasswordField fx:id="password" prefWidth="300.0" promptText="Password
                            " styleClass="textInput"/>
                        <Button fx:id="submitButton" mnemonicParsing="false" onAction="slotDir=
                            arg(arg()); call submitUserLogin slotDir;" text="Submit">
                            <font>
                                <Font size="36.0" />
                            </font>
                        </Button>
                        <Text fx:id="successText" fill="green" opacity="0.0" strokeType="
                            OUTSIDE" strokeWidth="0.0" text="Connection succesful."
                            textAlignment="CENTER" wrappingWidth="212.45703125">
                            <font>
                                <Font size="20.0" />
                            </font>
                        </Text>
                    </children>
                </VBox>
            </children>
        </Group>
    </children>
</StackPane>
</children>
<fx:script source="controller.rex" />
</AnchorPane>
```

Figure 35: Screenshot: user_login.fxml



After logging in the user lands on the main scene *main.fxml* (Listing 62, Figure 36). This scene uses *main_controller.rex* (Listing 61) as controller. When adding or editing orders the scene *add_order.fxml* (Listing 64, Figure 37) with controller *add_order_controller.rex* (Listing 63) is loaded. When adding products the scene *add_product.fxml* (Listing 66, Figure 38) with controller *add_product_controller.rex* (Listing 65) is loaded. Identically, the scene *barChart.fxml* (Listing 68, Figure 39) with controller *barChartController.rex* (Listing 67), scene *change_password.fxml* (Listing 70, Figure 40) with controller *change_password_controller.rex* (Listing 69), scene *edit_user.fxml* (Listing 72, Figure 41) with controller *edit_user_controller.rex* (Listing 71) and scene *change_access_right.fxml* (Listing 73, Figure 42) with controller *controller.rex* (Listing 54) are loaded when the corresponding button is pressed.

Listing 61: Source Code: main_controller.rex

```
/*@get(anchorPane orderTable orderIDCol tableNumberCol orderDetailsTable orderProdIDCol
    orderProdNameCol orderProdPriceCol orderProdQuantityCol orderStatusCol orderText)*/
/*@get(showStatsButton addProductButton addUserButton newOrderButton editOrderButton splitPayButton
    fullPayButton editUserButton editBizInfButton)*/
FXCollections = bsf.import("javafx.collections.FXCollections")

.my.app~orderText = orderText
```

```
orderArrayList = .my.app~dbh~getOpenOrderArrayList()
.my.app~orderHandler = .OrderHandler~new(orderArrayList, orderTable, orderIDCol, tableNumberCol)
orderDetailsArrayList = FXCollections~observableArrayList
.my.app~orderDetailsHandler = .OrderDetailsHandler~new(orderDetailsArrayList, orderDetailsTable,
    orderProdIDCol, orderProdNameCol, orderProdPriceCol, orderProdQuantityCol, orderStatusCol)

accessRight = .my.app~activeUser~accessRights

tierTwoButtons = .array~of(addProductButton, newOrderButton, editOrderButton, splitPayButton,
    fullPayButton)
tierThreeButtons = .array~of(addUserButton, showStatsButton)
tierFourButtons = .array~of(editUserButton, editBizInfButton)

IF accessRight < 4 THEN DO
  DO button OVER tierFourButtons
    button~disable = .true
  END
  IF accessRight < 3 THEN DO
    DO button OVER tierThreeButtons
    button~disable = .true
    END
    IF accessRight < 2 THEN DO
      DO button OVER tierTwoButtons
        button~disable = .true
      END
    END
  END
END

::ROUTINE openStats PUBLIC
  IF .my.app~dbh~getDistinctMonths = 0 THEN DO
    .my.app~stageHandler~showDialog("ERROR", "No data error", "No data found. Check again after
        completing orders")
    RETURN
  END
  .my.app~stageHandler~newWindow("Statistics", "file:barChart.fxml")

::ROUTINE startFullPayment PUBLIC
  /*@get(orderDetailsTable orderTable)*/
  CALL startPayment orderDetailsTable, orderTable, .false

::ROUTINE startSplitPayment PUBLIC
  /*@get(orderDetailsTable orderTable)*/
  CALL startPayment orderDetailsTable, orderTable

::ROUTINE openPasswordChangeWindow PUBLIC
  .my.app~resetPassword = .false
```

```
  .my.app~stageHandler~newWindow("Password change", "file:change_password.fxml")

::ROUTINE openBizInfWindow PUBLIC
  .my.app~stageHandler~newWindow("Change business informations", "file:edit_business_information.fxml")

::ROUTINE openUserForm PUBLIC
  .my.app~addRoot = .false
  .my.app~stageHandler~loadScene("Add User", "file:add_user.fxml")

::ROUTINE editOrder PUBLIC
  IF .my.app~currentOrderID = .nil THEN DO
    .my.app~stageHandler~showDialog("WARNING", "WARNING", "Please choose an order")
  END
  ELSE DO
    .my.app~stageHandler~loadScene("Add Order", "file:add_order.fxml")
  END

::ROUTINE reprintLastReceipt PUBLIC
  rec = .my.app~lastReceipt
  IF rec = .nil THEN .my.app~stageHandler~showDialog("ERROR", "ERROR", "No receipt printed in
      current session")
  ELSE DO
    CALL printReceipt rec
  END
::ROUTINE openProductForm PUBLIC
  .my.app~stageHandler~loadScene("Add Product", "file:add_product.fxml")


::ROUTINE newOrder PUBLIC
  .my.app~currentOrderID = .nil
  .my.app~stageHandler~loadScene("Add Order", "file:add_order.fxml")

::ROUTINE editUserScene PUBLIC
  .my.app~stageHandler~loadScene("Edit users", "file:edit_user.fxml")



::REQUIRES "BSF.CLS"
::REQUIRES "functions.rex"
::REQUIRES "TableHandler.CLS"
```

Listing 62: Source Code: main.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXNodesList?>
<?import java.lang.String?>
```

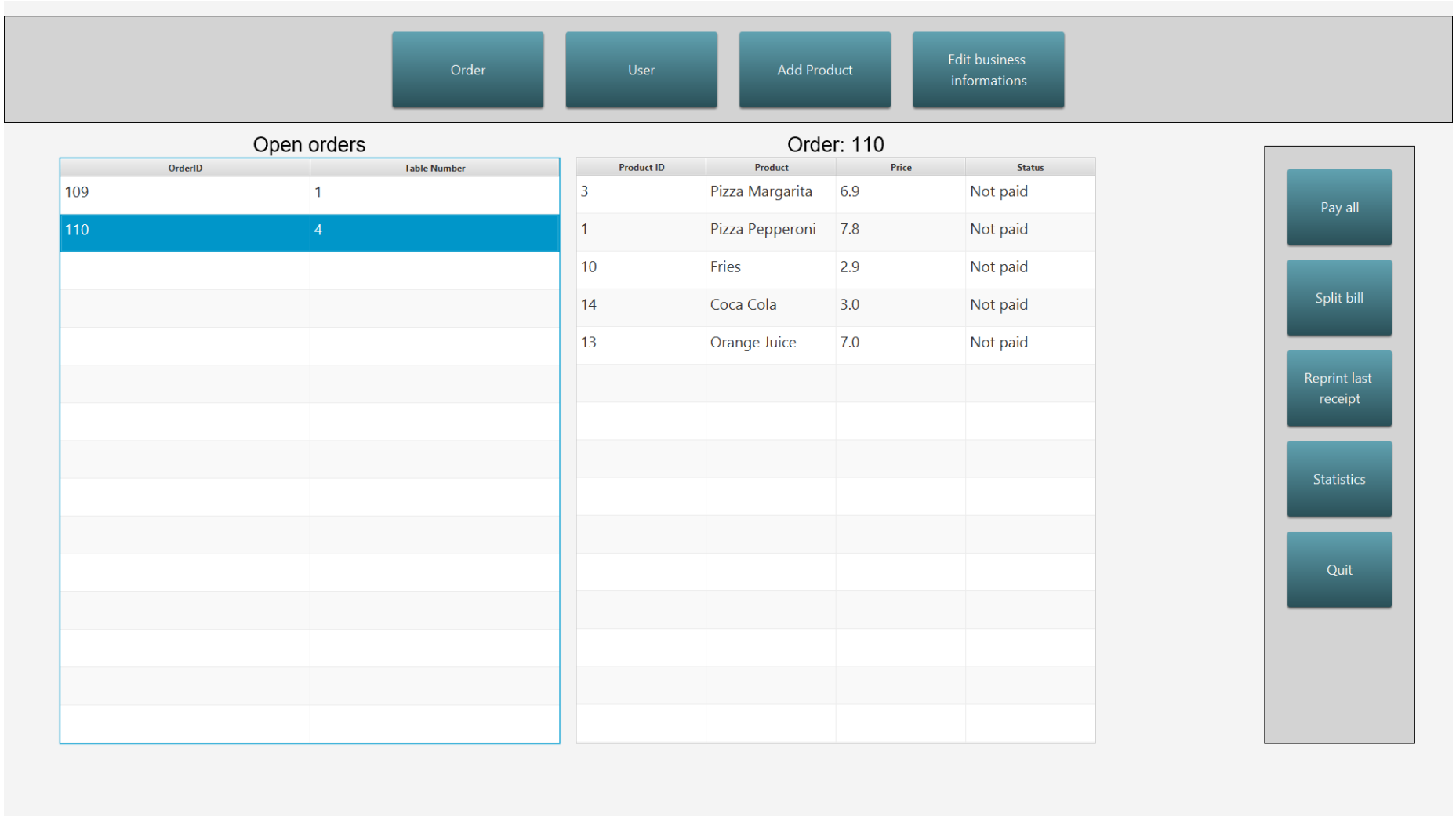
```
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<BorderPane fx:id="borderPane" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/
    fxml/1">
    <center>
        <AnchorPane fx:id="anchorPane" BorderPane.alignment="CENTER">
            <children>
                <GridPane hgap="30.0" layoutX="44.0" layoutY="72.0" prefHeight="822.0" prefWidth="
                    1447.0" vgap="30.0" AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="50.0"
                    AnchorPane.rightAnchor="20.0" AnchorPane.topAnchor="20.0">
                    <columnConstraints>
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                            />
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                            />
                    </columnConstraints>
                    <rowConstraints>
                        <RowConstraints maxHeight="-Infinity" minHeight="10.0" prefHeight="30.0" vgrow
                            ="SOMETIMES" />
                        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
                    </rowConstraints>
                    <children>
                        <Text fx:id="orderText1" strokeType="OUTSIDE" strokeWidth="0.0" text="Open
                            orders" textAlignment="CENTER" wrappingWidth="174.13671875" GridPane.
                            halignment="CENTER">
                            <font>
                                <Font name="Arial" size="27.0" />
                            </font>
                        </Text>
                        <TableView fx:id="orderTable" prefHeight="776.0" prefWidth="663.0" GridPane.
                            rowIndex="1">
                            <columns>
                                <TableColumn fx:id="orderIDCol" prefWidth="75.0" text="OrderID" />
                                <TableColumn fx:id="tableNumberCol" prefWidth="75.0" text="Table
                                    Number" />
                            </columns>
```

```
                    </TableView>
                    <TableView fx:id="orderDetailsTable" editable="true" prefHeight="777.0" prefWidth
                        ="689.0" GridPane.columnIndex="1" GridPane.rowIndex="1">
                        <columns>
                            <TableColumn fx:id="orderProdIDCol" prefWidth="75.0" text="Product ID"
                                />
                            <TableColumn fx:id="orderProdNameCol" prefWidth="75.0" text="Product"
                                />
                            <TableColumn fx:id="orderProdPriceCol" prefWidth="75.0" text="Price" />
                            <TableColumn fx:id="orderStatusCol" prefWidth="75.0" text="Status" />
                        </columns>
                    </TableView>
                    <Text fx:id="orderText" strokeType="OUTSIDE" strokeWidth="0.0" text="Order
                        details" textAlignment="CENTER" wrappingWidth="174.13671875" GridPane.
                        columnIndex="1" GridPane.halignment="CENTER">
                        <font>
                            <Font name="Arial" size="27.0" />
                        </font>
                    </Text>
                </children>
            </GridPane>
        </children>
    </AnchorPane>
</center>
<right>
    <VBox minWidth="100.0" prefWidth="200.0" spacing="20.0" styleClass="buttonBox">
        <children>
            <JFXButton fx:id="splitPayButton" buttonType="RAISED" mnemonicParsing="false"
                onAction="slotDir=arg(arg()); call startSplitPayment slotDir;" prefHeight="100.0"
                prefWidth="150.0" styleClass="menuButton" text="Split-Payment" wrapText="true">
                <font>
                    <Font size="16.0" />
                </font>
            </JFXButton>
            <JFXButton fx:id="fullPayButton" buttonType="RAISED" mnemonicParsing="false"
                onAction="slotDir=arg(arg()); call startFullPayment slotDir;" prefHeight="100.0"
                prefWidth="200.0" styleClass="menuButton" text="Full-Payment" wrapText="true">
                <font>
                    <Font size="16.0" />
                </font>
            </JFXButton>
            <JFXButton buttonType="RAISED" mnemonicParsing="false" onAction="slotDir=arg(arg())
                ; call reprintLastReceipt slotDir;" prefHeight="100.0" prefWidth="200.0" styleClass="
                menuButton" text="Reprint last receipt" textAlignment="CENTER" wrapText="true">
                <font>
                    <Font size="16.0" />
                </font>
```

```
            </JFXButton>
            <JFXButton fx:id="showStatsButton" buttonType="RAISED" mnemonicParsing="false"
                onAction="slotDir=arg(arg()); call openStats slotDir;" prefHeight="100.0" prefWidth="
                200.0" styleClass="menuButton" text="Statistics" textAlignment="CENTER" wrapText
                ="true">
                <font>
                    <Font size="16.0" />
                </font>
            </JFXButton>
            <JFXButton buttonType="RAISED" lineSpacing="1.0" mnemonicParsing="false" onAction=
                "slotDir=arg(arg()); call quitApp slotDir;" prefHeight="100.0" prefWidth="175.0"
                styleClass="menuButton" text="Quit" textAlignment="CENTER" wrapText="true">
                <font>
                    <Font size="16.0" />
                </font>
            </JFXButton>
        </children>
        <BorderPane.margin>
            <Insets left="20.0" right="50.0" top="30.0" />
        </BorderPane.margin>
        <padding>
            <Insets bottom="30.0" left="30.0" right="30.0" top="30.0" />
        </padding>
    </VBox>
</right>
<bottom>
    <HBox prefHeight="100.0" prefWidth="200.0" BorderPane.alignment="CENTER" />
</bottom>
<top>
    <HBox alignment="CENTER" prefHeight="100.0" prefWidth="200.0" spacing="30.0" styleClass="
        buttonBox" BorderPane.alignment="CENTER">
        <children>
            <JFXNodesList fx:id="nodeList" prefHeight="40.0" prefWidth="42.0" spacing="20.0">
                <children>
                    <JFXButton buttonType="RAISED" text="Order">
                        <styleClass>
                            <String fx:value="animated-option-button" />
                            <String fx:value="menuButton" />
                        </styleClass>
                    </JFXButton>
                    <JFXButton fx:id="newOrderButton" buttonType="RAISED" onAction="slotDir=
                        arg(arg()); call newOrder slotDir;" styleClass="animated-option-button" text="
                        New Order" />
                    <JFXButton fx:id="editOrderButton" buttonType="RAISED" onAction="slotDir=
                        arg(arg()); call editOrder slotDir;" styleClass="animated-option-button" text="
                        Edit Order" />
                </children>
```

```
            </JFXNodesList>
            <JFXNodesList fx:id="nodeList" prefHeight="40.0" prefWidth="42.0" spacing="20.0">
                <children>
                    <JFXButton buttonType="RAISED" text="User">
                        <styleClass>
                            <String fx:value="animated-option-button" />
                            <String fx:value="menuButton" />
                        </styleClass>
                    </JFXButton>
                    <JFXButton fx:id="addUserButton" buttonType="RAISED" onAction="slotDir=arg(
                        arg()); call openUserForm slotDir;" styleClass="animated-option-button" text="
                        New User" />
                    <JFXButton fx:id="changePasswordButton" buttonType="RAISED" onAction="
                        slotDir=arg(arg()); call openPasswordChangeWindow slotDir;" styleClass="
                        animated-option-button" text="Change Password" />
                    <JFXButton fx:id="editUserButton" buttonType="RAISED" onAction="slotDir=arg
                        (arg()); call editUserScene slotDir;" styleClass="animated-option-button" text="
                        Edit Users" />
                </children>
            </JFXNodesList>
            <JFXButton fx:id="addProductButton" buttonType="RAISED" mnemonicParsing="false"
                onAction="slotDir=arg(arg()); call openProductForm slotDir;" styleClass="menuButton"
                text="Add Product" textAlignment="CENTER" />
            <JFXButton fx:id="editBizInfButton" buttonType="RAISED" lineSpacing="1.0"
                mnemonicParsing="false" onAction="slotDir=arg(arg()); call openBizInfWindow slotDir;"
                styleClass="menuButton" text="Edit business informations" textAlignment="CENTER"
                wrapText="true" />
        </children>
        <BorderPane.margin>
            <Insets top="20.0" />
        </BorderPane.margin>
        <opaqueInsets>
            <Insets />
        </opaqueInsets>
        <padding>
            <Insets bottom="20.0" left="20.0" right="20.0" top="20.0" />
        </padding>
    </HBox>
</top>
<fx:script source="main_controller.rex" />
</BorderPane>
```

Figure 36: Screenshot: main.fxml



Listing 63: Source Code: add_order_controller.rex

```
/*@get(tableSpinner productTable productIDCol productNameCol productPriceCol productTaxCol
    orderDetailsTable orderProdIDCol orderProdNameCol orderProdPriceCol orderProdQuantityCol
    orderStatusCol)*/
FXCollections = bsf.import("javafx.collections.FXCollections")
IntegerSpinnerValueFactory = bsf.import("javafx.scene.control.
    SpinnerValueFactory$IntegerSpinnerValueFactory")

valueFactory = IntegerSpinnerValueFactory~new(0, 50, 0, 1)
tableSpinner~setValueFactory(valueFactory)
.local~tableSpinner = tableSpinner

orderDetailsArrayList = FXCollections~observableArrayList
productArrayList = .my.app~dbh~getProductArrayList()
productHandler = .ProductHandler~new(productArrayList, productTable, productIDCol, productNameCol,
    productPriceCol, productTaxCol)

.my.app~orderDetailsHandler = .OrderDetailsHandler~new(orderDetailsArrayList, orderDetailsTable,
    orderProdIDCol, orderProdNameCol, orderProdPriceCol, orderProdQuantityCol, orderStatusCol, .false)
IF .my.app~currentOrderID <> .nil THEN DO -- Current Order not nil --> Editing existing order
  .my.app~dbh~setOrderDetailsById(.my.app~currentOrderID, .my.app~orderDetailsHandler,.false)
  .tableSpinner~getValueFactory~setValue(box("Int", .my.app~tableNumber))
END



::ROUTINE saveOrder PUBLIC
  /*@get(orderDetailsTable)*/
  IF orderDetailsTable~getItems~isEmpty THEN DO
    .my.app~stageHandler~showDialog("WARNING", "WARNING", "Order is empty")
  END
  ELSE IF .tableSpinner~getValue = 0 THEN DO
    .my.app~stageHandler~showDialog("WARNING", "WARNING", "Please choose a table")
  END
  ELSE DO
    -- Add new order and return generated ID
    IF .my.app~currentOrderID <> .nil THEN DO -- Order already exists
      orderID = .my.app~currentOrderID
      .my.app~dbh~removeOrderDetailsFromDB(orderID) -- Remove old entries --> Then add new entries
    END
    ELSE do -- Order does not exist --> Create new order
      orderID = .my.app~dbh~addOrderToDB(.tableSpinner~getValue) -- Order does not exist --> Add
          new order and return generated id
    END
    DO item OVER orderDetailsTable~getItems
      DO i=1 FOR item~quantity~get
        .my.app~dbh~addOrderDetailsToDB(orderID, item~productID~get, item~status~get)
```

```
    END
  END
  .my.app~currentOrderID = .nil
  .my.app~stageHandler~loadMainScene
END

::REQUIRES "BSF.CLS"
::REQUIRES "functions.rex"
::REQUIRES "TableHandler.CLS"
::REQUIRES "controller.rex"
```

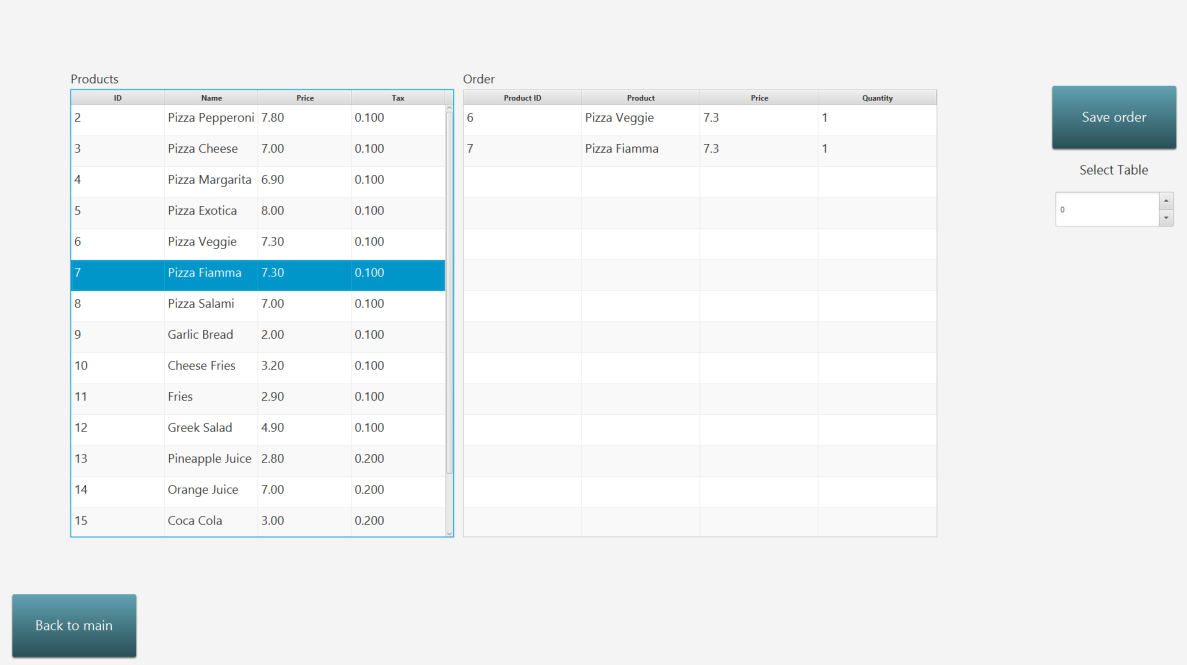
Listing 64: Source Code: add_order.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Spinner?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?language rexx?>
<BorderPane xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1">
    <center>
        <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0">
            <children>
                <TableView fx:id="productTable" editable="true" layoutX="14.0" layoutY="46.0" prefHeight
                    ="722.0" prefWidth="619.0">
                    <columns>
                        <TableColumn fx:id="productIDCol" prefWidth="96.0" text="ID" />
                        <TableColumn fx:id="productNameCol" prefWidth="92.0" text="Name" />
                        <TableColumn fx:id="productPriceCol" prefWidth="83.0" text="Price" />
                        <TableColumn fx:id="productTaxCol" prefWidth="97.0" text="Tax" />
                    </columns>
                </TableView>
                <TableView fx:id="orderDetailsTable" editable="true" layoutX="648.0" layoutY="46.0"
                    prefHeight="722.0" prefWidth="766.0">
                    <columns>
                        <TableColumn fx:id="orderProdIDCol" prefWidth="75.0" text="Product ID" />
                        <TableColumn fx:id="orderProdNameCol" prefWidth="75.0" text="Product" />
                        <TableColumn fx:id="orderProdPriceCol" prefWidth="75.0" text="Price" />
                        <TableColumn fx:id="orderProdQuantityCol" prefWidth="75.0" text="Quantity" />
                    </columns>
```

```
            </TableView>
            <Label layoutX="14.0" layoutY="14.0" text="Products">
                <font>
                    <Font size="20.0" />
                </font>
            </Label>
            <Label layoutX="648.0" layoutY="14.0" text="Order">
                <font>
                    <Font size="20.0" />
                </font>
            </Label>
        </children>
    </AnchorPane>
</center>
<bottom>
    <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call returnToMain slotDir;"
        prefHeight="50.0" prefWidth="118.0" text="Back to main" BorderPane.alignment="TOP_LEFT"
        >
        <BorderPane.margin>
            <Insets bottom="20.0" left="20.0" />
        </BorderPane.margin>
    </Button>
</bottom>
<right>
    <VBox alignment="TOP_CENTER" spacing="20.0" BorderPane.alignment="TOP_RIGHT">
        <children>
            <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call saveOrder slotDir;"
                prefHeight="84.0" prefWidth="188.0" text="Save order">
                <font>
                    <Font size="19.0" />
                </font>
            </Button>
            <Label prefHeight="27.0" prefWidth="112.0" text="Select Table">
                <font>
                    <Font size="21.0" />
                </font>
            </Label>
            <Spinner fx:id="tableSpinner" prefHeight="56.0" prefWidth="191.0" />
        </children>
        <opaqueInsets>
            <Insets />
        </opaqueInsets>
        <BorderPane.margin>
            <Insets right="20.0" top="40.0" />
        </BorderPane.margin>
    </VBox>
</right>
```

```
    <fx:script source="add_order_controller.rex" />
    <left>
        <VBox prefHeight="200.0" prefWidth="100.0" BorderPane.alignment="CENTER" />
    </left>
    <top>
        <HBox prefHeight="100.0" prefWidth="200.0" BorderPane.alignment="CENTER" />
    </top>
</BorderPane>
```

Figure 37: Screenshot: add_order.fxml



Listing 65: Source Code: add_product_controller.rex

```
/*@get(taxChoiceBox productTable productIDCol productNameCol productPriceCol productTaxCol)*/
FXCollections = bsf.import("javafx.collections.FXCollections")
observArrList = FXCollections~observableArrayList
observArrList~add("20%")
observArrList~add("19%")
observArrList~add("13%")
observArrList~add("10%")
observArrList~add("0%")
taxChoiceBox~setItems(observArrList)

productArrayList = .my.app~dbh~getProductArrayList
productHandler = .ProductHandler~new(productArrayList, productTable, productIDCol, productNameCol,
    productPriceCol, productTaxCol)


::ROUTINE returnToMain PUBLIC
```

```
  .my.app~remove(prodID)
  .my.app~stageHandler~loadMainScene

::ROUTINE editProduct PUBLIC
  /*@get(productTable productName productPrice taxChoiceBox)*/

  item = productTable~getSelectionModel~getSelectedItem
  IF item == .nil THEN DO
    .my.app~stageHandler~showDialog("WARNING", "WARNING", "Please select a product")
  END
  ELSE DO
    product = BsfRexxProxy(item)
    .my.app~prodID = product~id~get
    productName~text = product~name~get
    productPrice~text = product~price~get
    taxString = trunc(product~tax~get*100) || "%" -- taxID to percentage
    taxChoiceBox~setValue(taxString)
  END


::ROUTINE deleteProduct PUBLIC
  /*@get(productTable productIDCol productNameCol productPriceCol productTaxCol)*/
  item = productTable~getSelectionModel~getSelectedItem
  IF item == .nil THEN DO
    .my.app~stageHandler~showDialog("WARNING", "Warning", "Please select a product")
  END
  ELSE DO
    product = BsfRexxProxy(item)
    SIGNAL ON ANY NAME updateError
    .my.app~dbh~makeProductUnavailable(product~id~get)
    SIGNAL OFF ANY
    productArrayList = .my.app~dbh~getProductArrayList
    productHandler = .ProductHandler~new(productArrayList, productTable, productIDCol, productNameCol,
        productPriceCol, productTaxCol)
  END
  EXIT 0

  updateError:
  .my.app~stageHandler~showDialog("ERROR", "MySQL Update Error", "MySQL database update error")
  RETURN

::ROUTINE submitProduct PUBLIC
  USE ARG slotDir
  /*@get(productTable productName productPrice taxChoiceBox productIDCol productNameCol productPriceCol
      productTaxCol)*/
  choice = taxChoiceBox~getSelectionModel~getSelectedItem
  price = translate(productPrice~text, ".", ",")
```

```
  IF productName~text = "" | price = "" | choice = .nil | datatype(price) <> "NUM" THEN DO
    .my.app~stageHandler~showDialog("error", "ERROR", "Verify input fields")
  END
  ELSE DO
    IF prodID = .nil THEN DO - New Product not edit
      SIGNAL ON ANY NAME insertError
      .my.app~dbh~addProductToDB(productName~text, price, choice)
      SIGNAL OFF ANY
    END
    ELSE DO
      SIGNAL ON ANY NAME updateError
      .my.app~dbh~makeProductUnavailable(.my.app~prodID)
      SIGNAL OFF ANY

      SIGNAL ON ANY NAME insertError
      .my.app~dbh~addProductToDB(productName~text, price, choice)
      SIGNAL OFF ANY
    END
    productArrayList = .my.app~dbh~getProductArrayList
    productHandler = .ProductHandler~new(productArrayList, productTable, productIDCol, productNameCol,
          productPriceCol, productTaxCol)
    productName~text = ""
    productPrice~text = ""
    taxChoiceBox~getSelectionModel~clearSelection
  END

  EXIT 0

insertError:
  .my.app~stageHandler~showDialog("ERROR", "MySQL Insert Error", "MySQL database insert error")
  RETURN

updateError:
  .my.app~stageHandler~showDialog("ERROR", "MySQL Update Error", "MySQL database update error")
  RETURN


::REQUIRES "BSF.CLS"
::REQUIRES "controller.rex"
```

Listing 66: Source Code: add_product.fxml

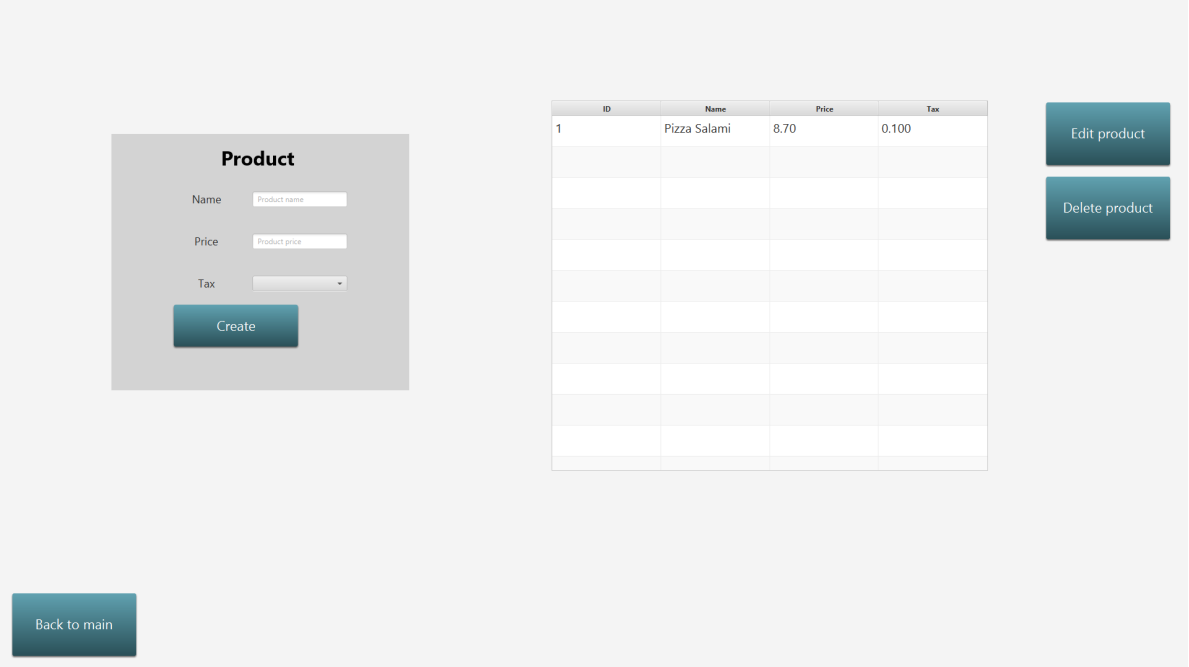
```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ChoiceBox?>
<?import javafx.scene.control.Label?>
```

```
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<BorderPane xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1">
    <center>
        <VBox prefHeight="1080.0" prefWidth="1920.0">
            <children>
                <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0"
                    VBox.vgrow="ALWAYS">
                    <children>
                        <GridPane alignment="CENTER" hgap="10.0" layoutX="180.0" layoutY="218.0"
                            prefHeight="414.0" prefWidth="481.0" styleClass="grid" vgap="10.0">
                            <columnConstraints>
                                <ColumnConstraints hgrow="SOMETIMES" maxWidth="126.0" minWidth=
                                    "10.0" prefWidth="108.0" />
                                <ColumnConstraints hgrow="SOMETIMES" maxWidth="153.0" minWidth=
                                    "10.0" prefWidth="153.0" />
                            </columnConstraints>
                            <rowConstraints>
                                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                                    " />
                                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                                    " />
                                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                                    " />
                                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                                    " />
                                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                                    " />
                                <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES
                                    " />
                            </rowConstraints>
                            <children>
                                <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Product"
                                    textAlignment="CENTER" wrappingWidth="270.13671875">
                                    <font>
                                        <Font name="System Bold" size="32.0" />
                                    </font>
```

```
            </Text>
            <Label alignment="CENTER" prefHeight="27.0" prefWidth="105.0" text="
                Name" textAlignment="JUSTIFY" GridPane.rowIndex="1">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>
            <Label alignment="CENTER" prefHeight="27.0" prefWidth="105.0" text="
                Price" textAlignment="JUSTIFY" GridPane.rowIndex="2">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>
            <Label alignment="CENTER" prefHeight="27.0" prefWidth="105.0" text="
                Tax" textAlignment="JUSTIFY" GridPane.rowIndex="3">
                <font>
                    <Font size="18.0" />
                </font>
            </Label>
            <ChoiceBox fx:id="taxChoiceBox" prefHeight="25.0" prefWidth="153.0"
                GridPane.columnIndex="1" GridPane.rowIndex="3" />
            <TextField fx:id="productName" promptText="Product name" GridPane.
                columnIndex="1" GridPane.rowIndex="1" />
            <TextField fx:id="productPrice" prefHeight="25.0" prefWidth="130.0"
                promptText="Product price" GridPane.columnIndex="1" GridPane.
                rowIndex="2" />
            <Button fx:id="addProductButton" mnemonicParsing="false" onAction="
                slotDir=arg(arg()); call submitProduct slotDir;" prefHeight="50.0"
                prefWidth="336.0" text="Create" GridPane.columnSpan="2" GridPane.
                rowIndex="4">
                <font>
                    <Font size="23.0" />
                </font>
            </Button>
        </children>
    </GridPane>
    <TableView fx:id="productTable" editable="true" layoutX="891.0" layoutY="164.0"
        prefHeight="598.0" prefWidth="705.0">
        <columns>
            <TableColumn fx:id="productIDCol" prefWidth="96.0" text="ID" />
            <TableColumn fx:id="productNameCol" prefWidth="92.0" text="Name" />
            <TableColumn fx:id="productPriceCol" prefWidth="83.0" text="Price" />
            <TableColumn fx:id="productTaxCol" prefWidth="97.0" text="Tax" />
        </columns>
    </TableView>
    <VBox layoutX="1690.0" layoutY="167.0" spacing="20.0">
        <children>
```

```
                        <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call
                            editProduct slotDir;" prefHeight="50.0" prefWidth="150.0" text="Edit
                            product">
                            <font>
                                <Font size="15.0" />
                            </font>
                        </Button>
                        <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call
                            deleteProduct slotDir;" prefHeight="50.0" prefWidth="150.0" text="
                            Delete product">
                            <font>
                                <Font size="15.0" />
                            </font>
                        </Button>
                    </children>
                </VBox>
            </children>
            <fx:script source="add_product_controller.rex" />
        </AnchorPane>
    </children>
</VBox>
</center>
<bottom>
    <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call returnToMain slotDir;"
        prefHeight="69.0" prefWidth="169.0" text="Back to main" BorderPane.alignment="
        CENTER_LEFT">
        <font>
            <Font size="22.0" />
        </font>
        <BorderPane.margin>
            <Insets bottom="20.0" left="20.0" />
        </BorderPane.margin>
    </Button>
</bottom>
</BorderPane>
```

Figure 38: Screenshot: add_product.fxml



Listing 67: Source Code: barChartController.rex

```
/*@get(barChart slider)*/
XYChart = bsf.loadClass("javafx.scene.chart.XYChart")
XYChartData = bsf.import("javafx.scene.chart.XYChart$Data")
XYChartSeries = bsf.import("javafx.scene.chart.XYChart$Series")
barChart~animated = .false
.local~monthlyTurnover = .my.app~dbh~getMonthlyTurnover
maxMonths = .my.app~dbh~getDistinctMonths

sliderHandler = .SliderHandler~new(slider, barChart, maxMonths)

::CLASS SliderHandler
::METHOD slider ATTRIBUTE
::METHOD barChart ATTRIBUTE
::METHOD INIT
        EXPOSE slider barChart XYChart XYChartData XYChartSeries
        USE ARG slider, barChart, maxMonths
        listener = BsfCreateRexxProxy(self,,"javafx.beans.value.ChangeListener")
        slider~valueProperty~addListener(listener)
        XYChart = bsf.loadClass("javafx.scene.chart.XYChart")
        XYChartData = bsf.import("javafx.scene.chart.XYChart$Data")
        XYChartSeries = bsf.import("javafx.scene.chart.XYChart$Series")
        IF maxMonths > 12 THEN monthToDisplay = 12
        ELSE monthToDisplay = maxMonths
        slider~setMax(maxMonths)
        slider~setValue(monthToDisplay)
```

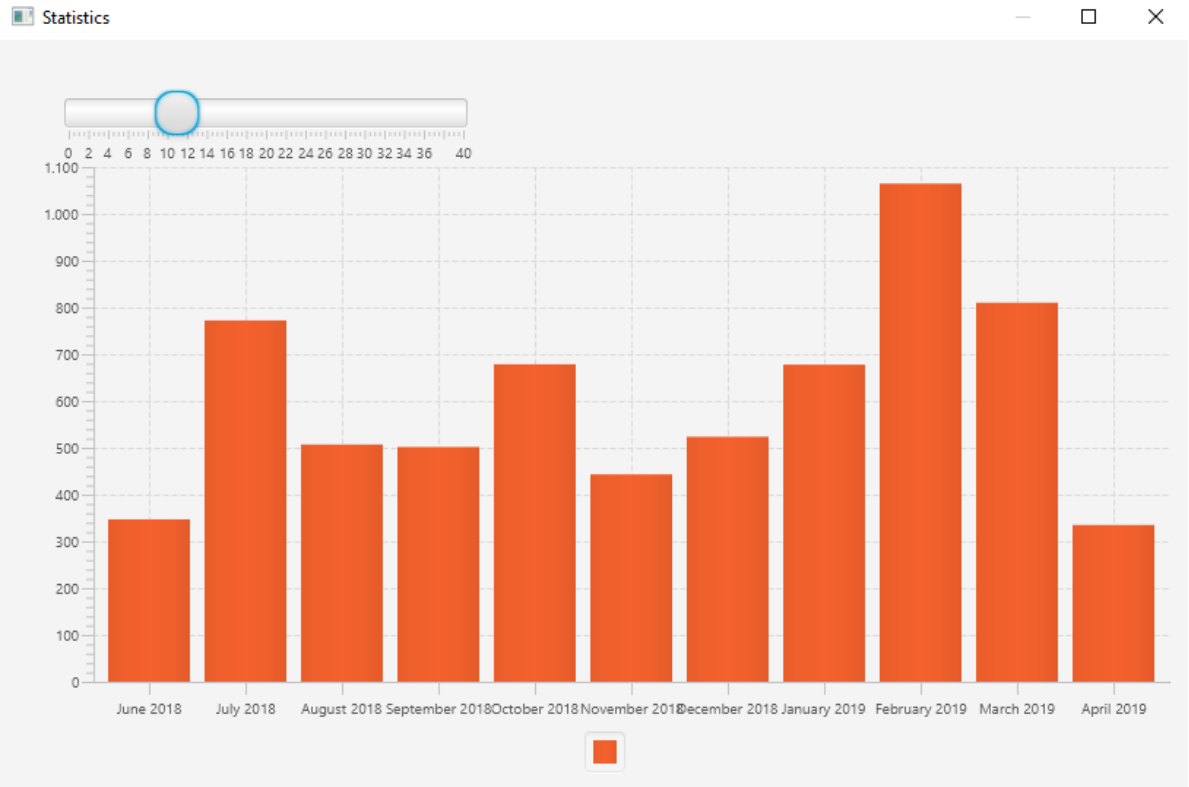
```
::METHOD changed
        EXPOSE slider barChart XYChart XYChartData XYChartSeries
        monthToDisplay = format(slider~getValue,,0)
        maxMonths = slider~getMax
        dataSeries1 = XYChartSeries~new
        DO i=maxMonths-monthToDisplay+1 TO maxMonths
          date = .monthlyTurnover['dates'][i]
          rev = box("float", .monthlyTurnover['revenues'][i])
          dataSeries1~getData~add(XYChartData~new(date, rev))
        END
        barChart~getData~clear
        barChart~getData~add(dataSeries1)

::REQUIRES "BSF.CLS"
::REQUIRES "DatabaseHandler.CLS"
```

Listing 68: Source Code: barChart.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.chart.CategoryAxis?>
<?import javafx.scene.chart.NumberAxis?>
<?import javafx.scene.chart.StackedBarChart?>
<?import javafx.scene.control.Slider?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?language rexx?>

<VBox xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0" VBox.
            vgrow="ALWAYS">
            <children>
                <StackedBarChart fx:id="barChart" layoutX="14.0" layoutY="72.0" prefHeight="434.0"
                    prefWidth="800.0">
                    <xAxis>
                        <CategoryAxis side="BOTTOM" />
                    </xAxis>
                    <yAxis>
                        <NumberAxis side="LEFT" />
                    </yAxis>
                </StackedBarChart>
                <Slider id="updateButtonSmall" fx:id="slider" blockIncrement="1.0" layoutX="32.0" layoutY="
                    35.0" majorTickUnit="2.0" prefWidth="300.0" showTickLabels="true" showTickMarks="true
                    " snapToTicks="true" />
            </children>
        </AnchorPane>
```

```
    </children>
    <fx:script source="barChartController.rex" />
</VBox>
```

Figure 39: Screenshot: barChart.fxml



Listing 69: Source Code: change_password_controller.rex

```
/*@get(oldPassword)*/

IF .my.app~resetPassword THEN oldPassword~disable=.true

::ROUTINE submitPasswordChange PUBLIC
  /*@get(oldPassword newPassword newPasswordRepeat)*/
  IF newPassword~text <> newPasswordRepeat~text THEN DO
    .my.app~stageHandler~showDialog("ERROR", "ERROR", "New password does not match")
  END
  ELSE IF newPassword~text = oldPassword~text THEN DO
    .my.app~stageHandler~showDialog("ERROR", "ERROR", "New password matches old password")
  END
  ELSE IF \oldPassword~disable & \.my.app~dbh~passwordCorrect(.my.app~activeUser~username,
      oldPassword~text) THEN DO
    .my.app~stageHandler~showDialog("ERROR", "ERROR", "Incorrect password")
  END
  ELSE DO
    SIGNAL ON ANY NAME databaseError
```

```
    IF .my.app~resetPassword THEN DO -- Changing password of a different user
      .my.app~dbh~changePassword(.my.app~selectedUsername, newPassword~text)
    END
    ELSE DO -- Changing own password
      .my.app~dbh~changePassword(.my.app~activeUser~username, newPassword~text)
    END
    SIGNAL OFF ANY
    .my.app~stageHandler~showDialog("INFO", "Success message", "Password successfully changed.")
    .my.app~stageHandler~windowStage~close
  END

  EXIT 0

databaseError:
  .my.app~stageHandler~showDialog("ERROR", "Database error", "Database error on password change")
  RETURN

::REQUIRES "controller.rex"
::REQUIRES "functions.rex"
::REQUIRES "BSF.CLS"
```

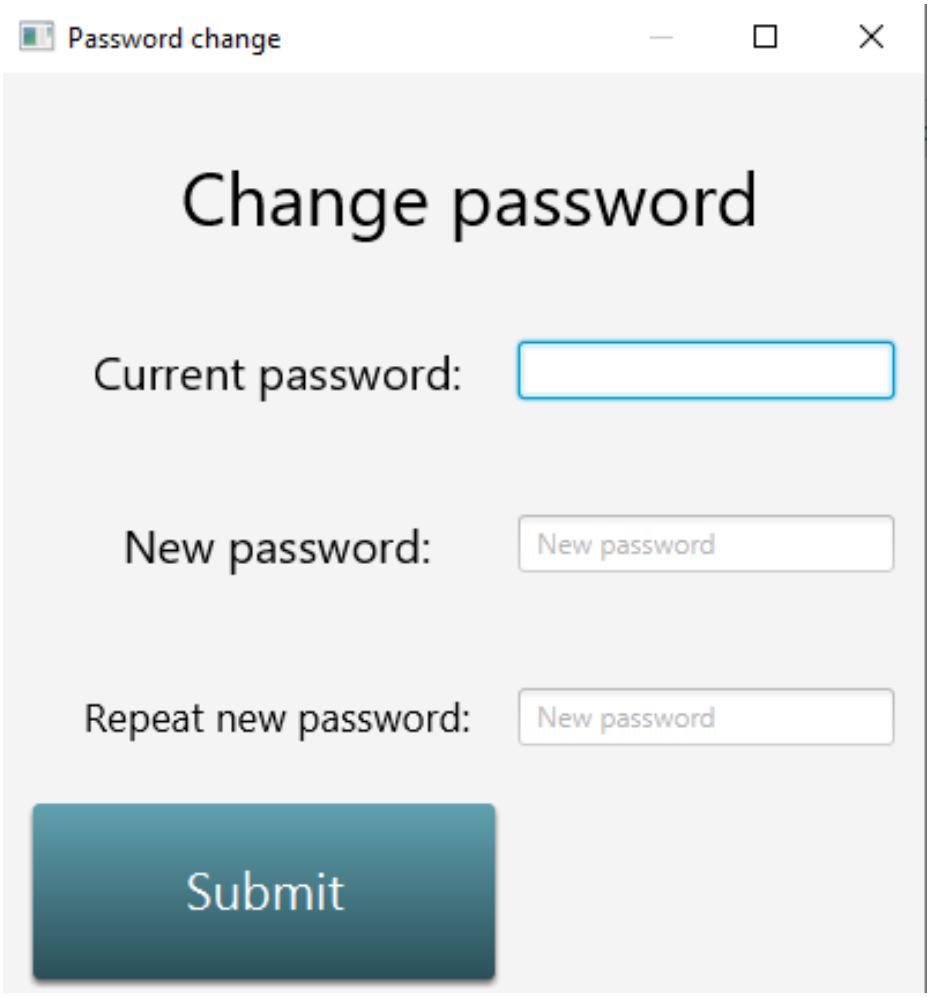
Listing 70: Source Code: change_password.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<VBox prefHeight="400.0" prefWidth="400.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://
    javafx.com/fxml/1">
    <children>
        <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0" VBox.
            vgrow="ALWAYS">
            <children>
                <GridPane layoutX="14.0" layoutY="14.0" prefHeight="377.0" prefWidth="373.0">
                    <columnConstraints>
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                            />
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                            />
                    </columnConstraints>
```

```
<rowConstraints>
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
</rowConstraints>
<children>
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Change password"
        textAlignment="CENTER" wrappingWidth="378.13671875" GridPane.
        columnSpan="2">
        <font>
            <Font size="32.0" />
        </font>
    </Text>
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Current password:"
        textAlignment="CENTER" wrappingWidth="210.78515625" GridPane.rowIndex=
        "1">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <PasswordField fx:id="oldPassword" promptText="Password" GridPane.columnIndex
        ="1" GridPane.rowIndex="1" />
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="New password:"
        textAlignment="CENTER" wrappingWidth="210.78515625" GridPane.rowIndex=
        "2">
        <font>
            <Font size="20.0" />
        </font>
    </Text>
    <Text strokeType="OUTSIDE" strokeWidth="0.0" text="Repeat new password:"
        textAlignment="CENTER" wrappingWidth="210.78515625" GridPane.rowIndex=
        "3">
        <font>
            <Font size="17.0" />
        </font>
    </Text>
    <PasswordField fx:id="newPassword" promptText="New password" GridPane.
        columnIndex="1" GridPane.rowIndex="2" />
    <PasswordField fx:id="newPasswordRepeat" promptText="New password" GridPane.
        columnIndex="1" GridPane.rowIndex="3" />
    <Button mnemonicParsing="false" prefHeight="44.0" prefWidth="198.0" text="
        Submit" GridPane.columnSpan="2" GridPane.rowIndex="4" onAction="slotDir=
        arg(arg()); call submitPasswordChange slotDir;">
        <font>
            <Font size="20.0" />
```

```
                        </font>
                    </Button>
                </children>
            </GridPane>
        </children>
        <fx:script source="change_password_controller.rex" />
    </AnchorPane>
  </children>
</VBox>
```

Figure 40: Screenshot: change_password.fxml



Listing 71: Source Code: edit_user_controller.rex

```
/*@get(userTable idCol usernameCol nameCol surnameCol accessRightCol dateOfLastLoginCol activeCol)*/

userArrayList = .my.app~dbh~getUserArrayList
.my.app~userHandler = .UserHandler~new(userArrayList, userTable, idCol, usernameCol, nameCol,
    surnameCol, accessRightCol, dateOfLastLoginCol, activeCol)
```

```
::ROUTINE editAccessRight PUBLIC
  /*@get(userTable)*/
  item = userTable~getSelectionModel~getSelectedItem
  IF \itemSelected(userTable) THEN .my.app~stageHandler~showDialog("WARNING", "WARNING", "No
      entry selected")
  ELSE DO
    user = BsfRexxProxy(item)
    .my.app~selectedUsername = user~username~get
    accessRight = user~accessRight~get
    active = user~active~get
    IF accessRight = 4 & .my.app~dbh~countActiveRootUsers() <= 1 & active = "Active" THEN DO
      .my.app~stageHandler~showDialog("ERROR", "ERROR", "Cannot remove/reduce rights of last active
          root user.")
    END
    ELSE DO
      .my.app~stageHandler~newWindow("Access right change", "file:change_access_right.fxml")
    END
  END

::ROUTINE overwritePassword PUBLIC
  /*@get(userTable)*/
  item = userTable~getSelectionModel~getSelectedItem
  IF \itemSelected(userTable) THEN .my.app~stageHandler~showDialog("WARNING", "WARNING", "No
      entry selected")
  ELSE DO
    user = BsfRexxProxy(item)
    .my.app~selectedUsername = user~username~get
    .my.app~resetPassword = .true
    .my.app~stageHandler~newWindow("Password change", "file:change_password.fxml")
  END

::ROUTINE changeActiveState PUBLIC
  /*@get(userTable)*/
  item = userTable~getSelectionModel~getSelectedItem
  IF \itemSelected(userTable) THEN .my.app~stageHandler~showDialog("WARNING", "WARNING", "No
      entry selected")
  ELSE DO
    user = BsfRexxProxy(item)
    id = user~id~get
    accessRight = user~accessRight~get
    active = user~active~get
    SIGNAL ON ANY NAME updateError
    IF accessRight = 4 & .my.app~dbh~countActiveRootUsers <= 1 & active = "Active" THEN DO
      .my.app~stageHandler~showDialog("error", "Error", "Cannot disable last active root user")
    END
    ELSE DO
      .my.app~dbh~toggleAccountActiveState(id)
```

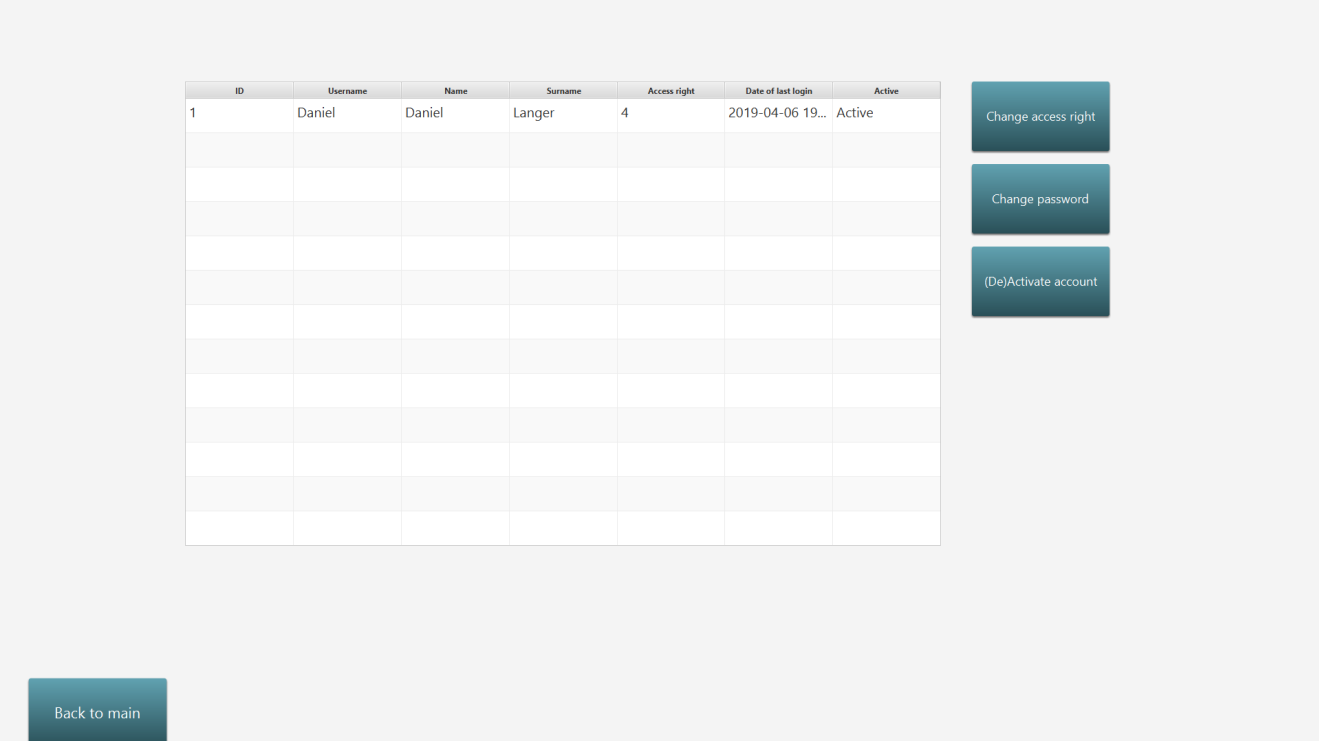
```
    END
    SIGNAL OFF ANY
    userArrayList = .my.app~dbh~getUserArrayList
    .my.app~userHandler~updateArrayList(userArrayList)
  END
  EXIT 0

updateError:
  .my.app~stageHandler~showDialog("error", "Datebase error", "Database error")
  RETURN

::REQUIRES "BSF.CLS"
::REQUIRES "controller.rex"
::REQUIRES "functions.rex"
```

Listing 72: Source Code: edit_user.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?language rexx?>
<VBox prefHeight="1080.0" prefWidth="1920.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http
    ://javafx.com/fxml/1">
    <children>
        <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0" VBox.
            vgrow="ALWAYS">
            <children>
                <Button layoutX="42.0" layoutY="988.0" mnemonicParsing="false" onAction="slotDir=arg(
                    arg()); call returnToMain slotDir;" prefHeight="50.0" prefWidth="118.0" text="Back to
                    main">
                    <font>
                        <Font size="15.0" />
                    </font>
                </Button>
                <TableView fx:id="userTable" layoutX="269.0" layoutY="120.0" prefHeight="675.0"
                    prefWidth="1099.0">
                    <columns>
                        <TableColumn fx:id="idCol" prefWidth="75.0" text="ID" />
                        <TableColumn fx:id="usernameCol" prefWidth="75.0" text="Username" />
                        <TableColumn fx:id="nameCol" prefWidth="75.0" text="Name" />
                        <TableColumn fx:id="surnameCol" prefWidth="75.0" text="Surname" />
                        <TableColumn fx:id="accessRightCol" prefWidth="75.0" text="Access right" />
                        <TableColumn fx:id="dateOfLastLoginCol" prefWidth="75.0" text="Date of last
```

```
                        login" />
                    <TableColumn fx:id="activeCol" prefWidth="75.0" text="Active" />
                </columns>
            </TableView>
            <VBox layoutX="1413.0" layoutY="120.0" spacing="20.0">
                <children>
                    <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call editAccessRight
                         slotDir;" prefHeight="73.0" prefWidth="250.0" styleClass="menuButton" text="
                        Change access right">
                        <font>
                            <Font size="18.0" />
                        </font>
                    </Button>
                    <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call
                        overwritePassword slotDir;" prefHeight="73.0" prefWidth="250.0" styleClass="
                        menuButton" text="Change password">
                        <font>
                            <Font size="18.0" />
                        </font>
                    </Button>
                    <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call
                        changeActiveState slotDir;" prefHeight="73.0" prefWidth="250.0" styleClass="
                        menuButton" text="(De)Activate account">
                        <font>
                            <Font size="18.0" />
                        </font>
                    </Button>
                </children>
            </VBox>
        </children>
        <fx:script  source=" edit_user_controller .rex" />
    </AnchorPane>
  </children>
</VBox>
```

Figure 41: Screenshot: edit_user.fxml



Listing 73: Source Code: change_access_right.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
<?language rexx?>
<VBox prefHeight="400.0" prefWidth="400.0" xmlns="http://javafx.com/javafx/8.0.171" xmlns:fx="http://
    javafx.com/fxml/1">
    <children>
        <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="-1.0" prefWidth="-1.0" VBox.
            vgrow="ALWAYS">
            <children>
                <GridPane layoutX="86.0" layoutY="104.0" prefHeight="245.0" prefWidth="229.0">
                    <columnConstraints>
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                            />
```

```
        </columnConstraints>
        <rowConstraints>
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
        </rowConstraints>
        <children>
            <RadioButton mnemonicParsing="false" text="Read only access">
                <font>
                    <Font size="23.0" />
                </font>
                <toggleGroup>
                    <ToggleGroup fx:id="accessRightGroup" />
                </toggleGroup>
            </RadioButton>
            <RadioButton mnemonicParsing="false" text="Normal access" toggleGroup="
                $accessRightGroup" GridPane.rowIndex="1">
                <font>
                    <Font size="23.0" />
                </font>
            </RadioButton>
            <RadioButton mnemonicParsing="false" text="Full access" toggleGroup="
                $accessRightGroup" GridPane.rowIndex="2">
                <font>
                    <Font size="23.0" />
                </font>
            </RadioButton>
            <RadioButton fx:id="rootRadioButton" mnemonicParsing="false" text="Root user"
                toggleGroup="$accessRightGroup" GridPane.rowIndex="3">
                <font>
                    <Font size="23.0" />
                </font>
            </RadioButton>
            <Button mnemonicParsing="false" onAction="slotDir=arg(arg()); call
                submitAcessRightChanges slotDir;" prefHeight="0.0" prefWidth="235.0" text="
                Submit" textAlignment="CENTER" GridPane.rowIndex="4">
                <font>
                    <Font size="19.0" />
                </font>
            </Button>
        </children>
    </GridPane>
    <Text layoutX="71.0" layoutY="59.0" strokeType="OUTSIDE" strokeWidth="0.0" text="
        Select access right" textAlignment="CENTER">
        <font>
```

```
                    <Font size="32.0" />
                </font>
            </Text>
        </children>
        <fx:script source="controller.rex" />
    </AnchorPane>
  </children>
</VBox>
```

Figure 42: Screenshot: change_access_right.fxml