WIRTSCHAFTSUNIVERSITÄT WIEN
Vienna University of Economics and Business

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Bachelorarbeit

Deutscher Titel der Bachelorarbeit	Registrierkassensystem laut RKSV (BGBI. II Nr. 313/2020)
Englischer Titel der Bachelorarbeit	Point-of-Service-System according to RKSV (BGBI. II Nr. 313/2020)
Verfasser/in **Familienname, Vorname(n)**	Schwarzer Manuel
Matrikelnummer	01426190
Studium	Bachelorstudium Wirtschafts- und Sozialwissenschaften
Beurteiler/in **Titel, Vorname(n), Familienname**	Prof. Dr. Rony G. Flatscher

Hiermit versichere ich, dass

1. ich die vorliegende Bachelorarbeit selbständig und ohne Verwendung unerlaubter Hilfsmittel verfasst habe. Alle Inhalte, die direkt oder indirekt aus fremden Quellen entnommen sind, sind durch entsprechende Quellenangaben gekennzeichnet.

2. die vorliegende Arbeit bisher weder im In- noch im Ausland zur Beurteilung vorgelegt bzw. veröffentlicht worden ist.

3. diese Arbeit mit der beurteilten bzw. in elektronischer Form eingereichten Bachelorarbeit übereinstimmt.

4. (nur bei Gruppenarbeiten): die vorliegende Arbeit gemeinsam mit

   entstanden ist. Die Teilleistungen der einzelnen Personen sind kenntlich gemacht, ebenso wie jene Passagen, die gemeinsam erarbeitet wurden.

24. Juni 2021
Datum

Unterschrift

WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

EFMD EQUIS ACCREDITED

Bachelorarbeit

Registrierkassensystem laut RKSV (BGBI. II Nr. 313/2020)

Manuel Schwarzer

Geburtsdatum: 22.11.1993
Matrikelnummer: 01426190

Fachrichtung: Wirtschaftsinformatik

Studienkennzahl: BaWiSo 12

Betreuer: Rony G. Flatscher

Datum der Einreichung: 23. Juni 2021

Institut für Wirtschaftsinformatik und Gesellschaft, Wirtschaftsuniversität Wien, Welthandelsplatz 1, 1020 Wien, Österreich

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Auflistungsverzeichnis

Abkürzungsverzeichnis

RKSV Registrierkassensicherheitsverordnung

BMF Bundesministerium für Finanzen

BGBI Bundesgesetzblatt

ooRexx Open Object REXX

GUI Graphical User Interface

JSON JavaScript Object Notation

JWS JSON Web Signature

AES Advanced Encryption Standard

BSF Bean Scripting Framework

AES Advanced Encryption Standard

BSF4ooRexx Bean Scripting Framework for ooRexx

JDK Java Platform, Standard Edition Development Kit

JWS JSON Web Signature

ASF Apache Software Foundation

Zusammenfassung

In der Vergangenheit wurde vermehrt daran gearbeitet, einheitliche Rahmenbedingung für Registrierkassensysteme in Österreich zu schaffen. Im April 2017 wurde die Registrierkassensicherheitsverordnung verabschiedet, die diese Lücke im Rechtssystem füllen soll. Diese Verordnung, kurz RKSV (BGBI. II Nr. 313/2020), deckt nicht nur technische Aspekte eines solchen Systems ab, sondern beleuchtet auch die rechtliche Lage. Zudem wird die RKSV ständig angepasst und erweitert. So werden einige technische Aspekte für den Einsatz eines elektronischen Kassensystems vorausgesetzt. Die essenziellsten Voraussetzungen sind zum einen, dass ein Kassensystem diverse Daten mit protokollieren muss, um diese dem zuständigen Bundesministerium für Finanzen übermitteln zu können, zum anderen müssen entsprechende Daten des Systems auf bestimmte Weise aufbereitet und verarbeitet werden. Zudem definiert die RKSV welche zusätzlichen Sicherheitseinrichtungen benötigt werden und wo diese zu erwerben sind. Weiters wird in der Registrierkassensicherheitsverordnung definiert, wie eine elektronische Kasse, beim zuständigen Bundesministerium für Finanzen, registriert werden muss, und welche Daten für dieses Vorhaben benötigt werden. Das zugrunde gelegte System ist so konzipiert worden, dass es für den Einsatz in einem kleinen gastwirtschaftlichen Betrieb eingesetzt werden kann. Für die Verwendung in der Praxis, müssen jedoch kleinere Anpassungen im Rahmen der Zertifizierungspflicht getätigt werden, alle weiteren technischen Voraussetzungen wurden zur Gänze umgesetzt. Für den Einsatz im gastwirtschaftlichen Bereich wurde eigens darauf geachtet, dass die Abwicklung des Kerngeschäfts unterstützt wird. So wurde der Bestellprozess in den Fokus gestellt und das System dahingehend designed und entwickelt, sodass dieser Prozess beschleunigt und erleichtert wird. Zudem wurden weitere essentielle Funktionen integriert, welche nicht direkt in der Registrierkassensicherheitsverordnung vorausgesetzt werden. Diese Funktionalitäten bieten zusätzliche Möglichkeiten den Geschäftsalltag zu erleichtern und werden unterdessen, sowie die vorgeschriebene Rahmenbedingungen, im Laufe dieser Arbeit erläutert ...

1 Einleitung

Seit 1.1.2016 gelten für Betriebe, welche einen bestimmten Betrag in bar einnehmen, verschärfte Regelungen für das Aufzeichnen solcher Transaktionen. Zudem ist es seit 1.4.2017 vorgeschrieben, dass für etwaige Betriebe ein entsprechendes elektronisches Kassensystem (Registrierkasse) zu verwenden ist. Mithilfe der Registrierkasse soll es unmöglich gemacht werden, dass entsprechende Unternehmen Rechnungen oder Belege nachträglich verändern können. Kurz gesagt dient solch ein System zur Vermeidung von Betrug oder Manipulation. Des Weiteren erleichtert eine elektronische und zum Teil maschinelle Verarbeitung von Daten, vermehrt Umsatz- und Transaktionsdaten, die Berechnung der Steuerpflicht für das jeweilige Unternehmen, welches solch eine Registrierkasse in Verwendung hat. Aus oben genannten Gründen ist das Bundesministerium für Finanzen zu dem Entschluss gekommen, dass ab einer gewissen Grenze von Barumsätzen innerhalb eines Geschäftsprozesses, solch ein System zu verwenden ist. Für diesen Zweck wurde eine Verordnung ins Leben gerufen, die die Rahmenbedingungen für eine Registrierkasse absteckt, die sogenannte Registrierkassensicherheitsverordnung, kurz RKSV (BGBI. II Nr. 313/2020). Dies ist mitunter ein Grund für die Entwicklung dieses Registrierkassensystems. Ein weiterer Grund ist, die steigende Nachfrage, sowie die steigenden Kosten für den Erwerb solch eines Systems. Aus bereits genannten Gründen, ist die Intention dieser Arbeit eine kostenfreie Alternative zu schaffen.
Die RKSV ist Grundlage dieser Arbeit und dem dazugehörigen Programm. Das System wurde dahingehend entwickelt, dass es mit kleinen Ergänzungen in der Praxis einsetzbar ist. Zudem wurde die Registrierkasse für einen kleinere gastwirtschaftliche Betriebe entwickelt.
In dieser Arbeit wird ein Überblick über bestimmte Voraussetzungen, sowie Rechte und Pflichten eines solchen Systems gegeben. Weiters folgt eine detaillierte Darstellung der Programminhalte, sowie deren technische Umsetzungen. Da die Bestandteile des Systems in einer Programmiersprache (ooRexx) verfasst wurden, welche als „Open-Source“ definiert ist, wurde bei der Entwicklung dafür Sorge getragen, dass das gesamte Programm und all seine technischen Komponenten ebenfalls als „Open-Source“ deklariert sind. Das hat den großen Vorteil, dass das gesamte Programm frei zugänglich ist, sodass keinerlei Gebühren oder dergleichen anfallen. Zudem ist es möglich etwaige Anpassungen, innerhalb des Source-Codes, vornehmen zu können, ohne dass rechtliche Konsequenzen drohen. Das gesamte Programm, ist dem Anhang beigefügt.

2 Registrierkassensicherheitsverordnung

In diesem Kapitel geht es um die Verordnung, welche die Voraussetzungen für ein etwaiges POS-System (Point-of-Sales-System, zu Deutsch Kassensystem bzw. Registrierkassensystem) vorgibt und dessen Voraussetzung, ab wann ein solches System zur Anwendung kommen muss. Diese Verordnung, BGBI. II Nr. 313/2020, und ihre Vorgänger wurde durch den Gesetzgeber erlassen, um es dem zuständigen Bundesministerium für Finanzen, kurz BMF, zu erleichtern etwaige Geldflüsse besser nachvollziehen zu können. Mehr Details über diese Voraussetzungen ab wann eine Registrierkasse zum Einsatz kommen muss folgen im Kapitel Voraussetzung auf der Seite 13. Weiters definiert die Registrierkassensicherheitsverordnung die generellen Rahmenbedingungen eines entsprechenden Systems. Als weiterer Schritt zum besseren Verständnis, werden die wichtigsten Eckpunkte aus der Verordnung herausgegriffen und detaillierter betrachtet, siehe dazu Kapitel Generelle Rahmenbedingungen eines Registrierkassensystems auf der Seite 14. Zudem wird im Kapitel Potentielle Vertrauensdiensteanbieter auf der Seite 15 auf die Thematik der Vertrauensdiensteanbieter und potentielle Anbieter eingegangen. Als Nächstes folgt eine Erläuterung über die Problematik der Zertifizierungspflicht, siehe Kapitel Zertifizierungspflicht auf der Seite 16. Weiters folgt eine zur Schaustellung der Mechanik, die dafür sorgen soll, dass keinerlei Rechnungsbelege verändert oder gelöscht werden können, siehe Kapitel Lückenlose Belegerstellung auf der Seite 17. Weiters folgt eine Zusammenfassung der Meldepflicht an das Bundesministerium für Finanzen, siehe Kapitel Meldepflicht - Bundesministerium für Finanzen auf der Seite 19 und eine detaillierte Veranschaulichung unterschiedlicher Belegarten, die ebenfalls durch die RKSV definiert sind, siehe Kapitel Belegarten auf der Seite 20. Zu guter Letzt erfolgt eine zur Erläuterung eines maschinenlesbaren Codes, QR-Code, welcher zur maschinellen Verarbeitung zwingender Bestandteil eines Belegs zu sein hat, siehe Kapitel QR-Code auf der Seite 26.

2.1 Voraussetzung

Der Einsatz einer Registrierkasse im Betrieb, ist nicht zwingend notwendig. Damit ein Betrieb zwingend eine Registrierkasse einsetzen muss, müssen einige Voraussetzungen dafür erfüllt sein. Diese Voraussetzungen werden nun näher erläutert. Eine Registrierkassenpflicht besteht erst, wenn:

- der Jahresumsatz je Betrieb €15.000 [1] und
- die Barumsätze dieses Betriebes €7.500 im Jahr überschreiten. [1]

Fällt ein Betrieb nicht in diese Kategorie, ist der Betrieb auch nicht registrierkassenpflichtig und muss andere dokumentarische Leistungen erbringen, die nicht Bestandteil dieser Arbeit sind, da der Fokus auf den Einsatz einer Registrierkasse gelegt wird. Zudem gibt es mehrere Sonderregelungen und Ausnahmen für Betriebe, die zwar per se in diese Pflicht fallen würden, jedoch einige Erleichterungen erwarten dürfen. Solche Ausnahmen und Erleichterungen greifen bei:

- Umsätze im Freien („Kalte-Händeregelung“) bis zu €30.000 im Jahr [1]
- Bestimmte Automaten wie etwa Tischfußball-, Musik- oder Dartautomaten [1]
- Onlineshop-Umsätze, die nicht mit Bargeld erzielt werden [1]
- Bestimmte Kantinen, Buschenschanken und Hütten (Alm-, Berg-, Ski-) [1]

Zudem gibt es noch mehrere und komplexere Ausnahmeregelungen. Für mehr Informationen darüber, wird auf die Wirtschaftskammer-Website verwiesen [2].

Sofern ein Betrieb jedoch registrierkassenpflichtig ist, ist es zwingend notwendig, dass ein RKSV-konformes Kassensystem zur Anwendung kommt. Die rechtlichen, wie technischen Voraussetzungen und Bestimmungen können im Detail in der RKSV selbst nachgelesen werden [3].

2.2 Generelle Rahmenbedingungen eines Registrierkassensystems

Wie bereits erwähnt, gibt der Gesetzgeber einige technische Rahmenbedingungen vor. Diese wurden im vorherigen Kapitel Voraussetzung auf Seite 13 kurz thematisiert. In diesem Kapitel werden die wichtigsten Aspekte näher betrachtet. Die mitunter wichtigsten Bestandteile eines Registrierkassensystems, laut RKSV, lauten wie folgt:

- Datenerfassungsprotokoll
- Drucker oder eine Vorrichtung zur elektronischen Übermittlung von Zahlungsbelegen
- Schnittstelle zu einer Sicherheitseinrichtung mit einer Signatur- bzw. Siegelerstellungseinheit
- Verschlüsselungsalgorithmus AES 256
- Kassenidentifikationsnummer

Ein Datenerfassungsprotokoll dient dazu, wie der Name schon vermuten lässt, um Daten erfassen zu können. Dazu gibt der Gesetzgeber keine genaueren Bestimmungen für die Realisierung vor. Zum einen ist eine Liste, oder ein klassisches Kassenjournal ein solches Protokoll. Zudem ist es ebenfalls möglich Daten innerhalb einer Datenbank erfassen zu können. Der Sinn und Zweck dieses Protokolls ist es, dass jede Transaktion, in der ein Barumsatz generiert wurde, unveränderbar aufgezeichnet wird. Die zweite technische Voraussetzung, laut BGBI. II Nr. 313/2020, ist, dass die Möglichkeit bestehen muss, einen Beleg dem Kunden auszuhändigen zu können. Dies kann entweder auf analoger Basis geschehen, indem der Beleg ausgedruckt und überreicht wird, oder auf digitalem Wege mittels elektronischer Übermittelung per E-Mail. Jedenfalls muss das System garantieren, dass der Kunde nach abgeschlossenem Bezahlvorgang den Rechnungsbeleg einsehen kann. Dies dient einerseits zu Kontrollzwecken auf Seiten des Kunden, ob alle Beträge und Produkte korrekt sind, andererseits muss der Betrieb selbst alle erstellten Belege, in Kopie, aufbewahren [4]. Als dritten wichtigen Bestandteil setzt der Gesetzgeber voraus, dass das Registrierkassensystem mit einer Signatur- bzw. Siegelerstellungseinheit interagieren kann. So eine Einheit dient dafür, dass alle erstellten Belege entsprechend der Vorgaben signiert werden. Mehr dazu in den folgenden Kapiteln: Kapitel Zertifizierungspflicht auf Seite 16, Kapitel Lückenlose Belegerstellung auf Seite 17 und Kapitel Belegarten auf Seite 20. Als vierten Punkt gibt die RKSV (BGBI. II Nr. 313/2020) vor,

dass jedes System einen internen Zählerstand führen muss, der den gesamten Umsatz protokolliert. Sobald eine Bartransaktion, seitens des zahlenden Kunden, getätigt wird, muss der Zählerstand aktualisiert werden. Zu guter Letzt muss einer Registrierkasse eine entsprechende Identifikation zugeordnet werden.

2.3 Potentielle Vertrauensdiensteanbieter

In diesem Kapitel geht es um Vertrauensdiensteanbieter, kurz VDA. Wie im vorherigen Kapitel erwähnt, ist es zwingend notwendig eine Signatur- bzw. Siegelerstellungseinheit in einem POS-System einzubinden bzw. anzubinden. Diese Einheit ist dafür zuständig, dass die Belege signiert werden. In diesem Kapitel werden entsprechende Anbieter und deren Angebot durchleuchtet.

Die Signaturerstellungseinheit dient dazu, um Belege signieren zu können. Diese Signaturen dienen dazu, dass Belege nachträglich, nach Signierung, nicht mehr veränderbar sind, ohne dass dies auffällt. Besagte Einheit muss von einem vertrauenswürdigen Anbieter bezogen werden [3]. Da vertrauenswürdig ein sehr relativer Begriff ist, und zumeist im Auge des Betrachters liegt, nimmt der Gesetzgeber eine Vorauswahl von Anbietern vor. Betriebe, welche so eine Dienstleistung anbieten wollen, müssen sich bei der zuständigen Behörde, in diesem Fall der RTR (Rundfunk und Telekom Regulierungs-GmbH), vorstellen. Danach werden diese Betriebe und deren Service genauestens überprüft. Kommt man zu dem Entschluss, dass ein Betrieb und dessen Service, im Auge des Gesetzgebers, vertrauenswürdig ist, wird dieser Betrieb auf einer eigenen Liste geführt [5].

Die gelisteten Unternehmen haben dafür Sorge zu tragen, dass zu jeder Zeit die angebotenen Dienste vertrauenswürdig und sicher sind. Die angesprochene Liste führt Betriebe unterschiedlicher Branchen, nicht nur für den Betrieb einer Registrierkasse. Lediglich die unterhalb gelisteten Betriebe, in Summe drei, sind dafür zuständig die Thematik für Signatur- bzw. Siegelerstellungseinheit einer Registrierkasse abzuarbeiten. Sprich, sofern ein POS-System in Betrieb genommen werden möchte, so muss davor ein Zertifikat bei einem der drei Betriebe, welche in Tabelle 1 auf der Seite 16 gelistet sind, erworben werden. Mehr dazu in Kapitel Zertifizierungspflicht auf der Seite 16.

VDA	Link
A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH	https://www.a-trust.at/ [6]
e-commerce monitoring GmbH	http://www.e-monitoring.at/ [7]
PrimeSign GmbH	https://www.prime-sign.com/ [8]

Tabelle 1: Vertrauensdiensteanbieter

Wie bereits erwähnt übernehmen diese Unternehmen die Handhabe bezüglich der Erstellung von etwaigen Signaturen für die zu signierenden Belege. In Kapitel Belegarten auf der Seite 20 ist genau beschrieben, wie ein Beleg auszusehen hat und was dieser beinhalten muss. Für Systementwickler, welche ein Registrierkassensystem entwickeln möchten, empfiehlt es sich A-Trust als Anbieter zu wählen, da dieser eine Palette an Dokumenten bereitstellt, die es einem vereinfachen solch ein System RKSV-konform entwickeln zu können. So wurde auch im Zuge der Entwicklung dieses POS-Systems A-Trust als Anbieter gewählt.

2.4 Zertifizierungspflicht

Der Gesetzgeber bzw. die dafür zuständige Institution, das Bundesministerium für Finanzen, haben einen großen Anreiz daran, dass Unternehmen korrekte Belege erstellen. Ein Grund dafür ist, dass auf Grundlage der erstellten Belege in Kombination mit der gesamten Unternehmensbuchhaltung, monetäre Verpflichtungen auferlegt werden. Kurz gesagt, eine korrekte Berechnung der Steuerlast eines Unternehmens, ist zum Vorteil aller beteiligten Institutionen. Da es in der Vergangenheit vermehrt vorkam, dass Betriebe in dieser Hinsicht getrickst haben, werden die Rahmenbedingungen für die Berechnung immer enger gesteckt. Auf Grund dessen, ist es seit 01.04.2017 Pflicht, ein entsprechendes Registrierkassensystem nur in Kombination mit einer Signatur- bzw. Siegelerstellungseinheit zu verwenden, genauere Informationen sind im Kapitel Voraussetzung auf der Seite 13 zu finden. Diese Einheiten kümmern sich um die Signatur etwaiger Belege. So eine Signatur läuft wie folgt ab. Zuerst werden alle relevanten Daten eines Beleges gesammelt. In Kapitel QR-Code auf der Seite 26 (Punkt 1-12) ist eine Liste mit allen relevanten Daten. Die gesammelten Daten werden anschließend an die Signaturerstellungseinheit weitergeleitet. Dies kann in unterschiedlichen Arten erfolgen. Zum einen ist es möglich eine Signatur zu erstellen ohne dass eine aufrechte Internetverbindung bestehen muss, andererseits kann man die Daten auch über das Internet signieren lassen. Beide Varianten wurden für diesen Zweck in das zugrunde gelegte POS-System integriert. Für die Offline-

Signierung ist es notwendig, dass ein Kartenlesegerät plus zugehörige Karte erworben wird. Die Karte beinhaltet mitunter ein einzigartiges Zertifikat, welches dem Betrieb eindeutig zuordenbar ist. Das heißt, jedes Zertifikat für eine Registrierkasse, bzw. besser gesagt für deren Signaturerstellungseinheit ist für jeden Betrieb unterschiedlich, sodass jeder Beleg wiederum nur einem Unternehmen zuordenbar ist. Diese Zertifikat beinhaltet wesentliche Informationen über das Unternehmen, wie die Umsatzsteuernummer, den Firmennamen, bis wann das Zertifikat gültig ist und vieles mehr.
Für die Online-Signierung muss ebenfalls ein Zertifikat erworben werden. Dieses ist jedoch nicht physisch im Besitz des Betriebes, welches das Zertifikat erworben hat, sondern der jeweilige Vertrauensdiensteanbieter verwaltet das entsprechende Zertifikat. Jedoch ist für diesen Zweck eine aufrechte Internetverbindung notwendig, da entsprechende Signierungen beim entsprechenden Anbieter stattfinden und eine Möglichkeit bestehen muss, die fertigen Signierungsdaten empfangen zu können. Mehr über die technische Abfolge unter Kapitel Signierungsart wählen auf der Seite 53. Das Ergebnis einer erfolgreichen Signaturerstellung ist ein sogenannter Hash-Wert. Dieser Wert ist essenziell wichtig, da mithilfe dieses Wertes eine Art Block-Chain der Belege aufgebaut wird. Dieser Sachverhalt wird im nächsten Kapitel genauer erklärt.

Ist es nicht möglich eine Signatur zu erstellen, weil womöglich das Kartenlesegerät defekt ist oder die Internetverbindung abgebrochen ist, so ist es notwendig unsignierte Belege zu kennzeichnen und dem Bundesministerium für Finanzen nachträglich zu melden, bzw. mittels Sammelbeleg, alle nichtsignierten Belege zu erfassen, sobald die Signatureinheit wieder funktionsfähig ist, sodass eine lückenlose Belegerstellung gewährleistet ist. Für die Meldung an das Bundesministerium für Finanzen gibt es eigens dafür vorgesehene Menüpunkte im jeweilige FinanzOnline-Account des Unternehmens, siehe Kapitel Meldepflicht - Bundesministerium für Finanzen auf der Seite 19.

2.5 Lückenlose Belegerstellung

In diesem Kapitel wird der Umstand einer Block-Chain näher erläutert und deren Sinn dahinter. Sofern eine neue Rechnungsperiode startet, muss das POS-System neu initialisiert werden. Für diesen Umstand sieht die RKSV einen sogenannten Startbeleg vor. Genauere Informationen über Belegarten und ihrer Verwendung stehen im Kapitel Belegarten auf der Seite 20. Ein Startbeleg ist essenziell wichtig für das Garantieren einer lückenlosen Belegkette. Da per Definition einer neuen Rechnungsperiode kein Beleg davor erstellt wurde und somit, in dieser Periode, noch nie ein Beleg erstellt wurde, muss sichergestellt werden, dass dieser Beleg als Startpunkt gekennzeichnet

wird. Der Gesetzgeber sieht vor, dass in diesem Fall die Kassenidentifikation herangezogen wird. Sofern der Startbeleg erfolgreich „bezahlt“ wurde, wurde ebenfalls ein entsprechender Signaturwert erzeugt. Dieser Signaturwert wird wiederum auf den nächsten Standardbeleg mit abgebildet. So entsteht eine Kette aus Belegen, wobei immer der jetzige Beleg auf den Beleg davor zeigt. Diese Technik nennt man auch Block-Chain, da ein Beleg auf den zuletzt erstellten Beleg verweist. Zur Illustration und besserem Verständnis siehe Abbildung 1 unterhalb.

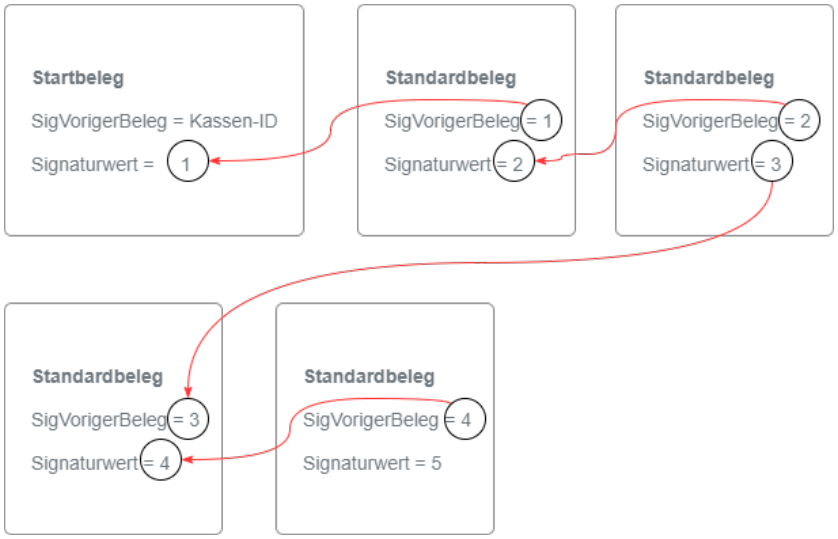

Abbildung 1: Lückenlose Belegerstellung

Mithilfe dieser Technik ist es möglich sofort zu erkennen, ob ein Beleg nachträglich verändert oder gelöscht wurde, weil die entstandene Signaturkette nicht mehr übereinstimmt. Wie in Abbildung 1 ersichtlich, benötigt man für dieses Vorgehen zwei verschiedene Werte. Zur Veranschaulichung wurden diese Werte vereinfacht. Zudem steht das Wort „SigVorigerBeleg“ für „Signaturwert-Voriger-Beleg“, was so viel bedeutet wie, der Signaturwert des davor erstellten Belegs. Wie die Pfeile veranschaulichen, zeigen diese Werte jeweils auf einander, sodass eine Kette entsteht. Die dargestellten Werte sind rein fiktiv und dienen nur zu Demonstrationszwecken. Nähere Informationen bezüglich der technischen Umsetzung stehen im Anhang im Kapitel Sicherheitseinrichtung auf der Seite 118.

2.6 Meldepflicht - Bundesministerium für Finanzen

Das Bundesministerium für Finanzen räumt sich das Recht ein, dass etwaige Point-of-Sales-Systeme gemeldet werden müssen. Somit ist jeder Betrieb verpflichtet, die zu benutzende Registrierkasse zu melden. Dies kann unterschiedlich erfolgen. Unterhalb befindet sich eine Liste mit allen Möglichkeiten.

- Online via FinanzOnline - Datenstromverfahren
- Offline mittels Formular - Dialogverfahren

Die erste Variante ist für all die Systeme gedacht, die zum einen internetfähig sind, zum anderen sogenannte XML-Dateien erstellen können, worin alle möglichen Informationen verpackt und übermittelt werden. XML steht für Extensible Markup Language und ist sowohl vom Menschen, als auch maschinell lesbar [9].
Die zweite Variante, mittels Formular, ist für all jene Systeme gedacht, die zum einen womöglich keine Internetverbindung benötigen oder besitzen, zum anderen für Systeme, welche keine XML-Dateien erstellen können. In diese Kategorie fällt auch das POS-System, auf die diese Bacharbeit basiert. Im Dialogverfahren ist es zuerst notwendig, dass ein FinanzOnline-Konto für den jeweiligen Betrieb existiert. Sofern ein Account besteht, ist es möglich unter den Menüpunkten „Eingabe“ und „Registrierkasse“ die Funktionsauswahl zu erreichen [10].

Nun müssen, Menüpunkt für Menüpunkt, Angaben zur Signatur- bzw. Siegelerstellungseinheit, sowie zu Registrierkasse selbst getätigt werden. Über FinanzOnline ist es ebenfalls möglich, zu melden ob und wann eine Registrierkasse oder die dazugehörige Signatur bzw. Siegelerstellungseinheit ausgefallen ist. Für detailliertere Informationen diesbezüglich wird auf das offizielle Handbuch des Bundesministerium für Finanzen, erreichbar unter folgendem Link:
https://www.bmf.gv.at/dam/jcr:0af97a40-da60-4c81-8e1e-22c3ecca52a4/BMF_Handbuch_Registrierkassen.pdf (Aufruf: 10-06-2021) verwiesen.

2.7 Belegarten

Damit ein Registrierkassensystem RKSV-konform ist und somit vom Bundesministerium für Finanzen genehmigt werden kann, ist es notwendig, dass das System unterschiedliche Belegarten verarbeiten kann. Diese Arten werden im Detail erläutert und mit passenden Abbildungen veranschaulicht. Unterdessen befindet sich unterhalb eine Aufzählung, welche konkreten Arten möglich sind und vom System verarbeitet werden können müssen.

- Startbeleg
- Standardbeleg
- Nullbeleg
- Stornobeleg
- Trainingsbeleg

2.7.1 Startbeleg

Jede Art von Beleg löst eine unterschiedliche Reihe von Funktionen, innerhalb der Registrierkasse, aus. So wurde schon erwähnt, dass ein Startbeleg dazu dient, ein System zu initialisieren. Das heißt, zu Beginn jeder Rechnungsperiode, zumeist am 1. Jänner des Jahres, wird das System auf die neue Rechnungsperiode eingestellt. Das setzt voraus, dass alle bisherigen Belege, welche in einer Datenbank gesammelt gespeichert werden, bereits exportiert und gesichert wurden.

Da noch keine vorherigen Belege, per Definition, bestehen, muss der Beginn der Beleg-Kette gekennzeichnet werden. Dazu wird die Kassenidentifikation herangezogen. Diese Kassen-ID wird kodiert und dient als erste Signatur. Auf diesem Beleg befinden sich weder Produkte, Waren oder andere umsatzfähigen Leistungen. Dieser Beleg dient ausschließlich als Startpunkt der lückenlosen Belegerstellung wie sie in der Abbildung 1 auf Seite 18 fiktiv dargestellt ist. Abbildung 2 stellt solch einen Startbeleg dar.

Fake GmbH

fakestreet 123
0650 123456789
company@chello.at
AT123456789
OrderID: 1
TableNr.: 10
2021-06-11
12:51:11

Item Qty Price Total

Total amount: € 0

MWST Net Tax Gross

Manuel served you!
Thank you for your purchase.
We look forward to seeing you again soon.

Abbildung 2: Startbeleg

Dieser Beleg dient weder zur Berechnung etwaiger Produkte oder Leistungen, noch zu dokumentarischen Zwecken, er wird ausschließlich für den Initialisierungsprozess benötigt.

2.7.2 Standardbeleg

In diese Kategorie von Belegarten fällt der standardisierte Beleg, welcher Kunden ausgehändigt werden muss. Dieser Beleg muss, laut Rechtssprechung, mindestens folgende Bestandteile aufweisen:

1. Bezeichnung des leistenden/liefernden Unternehmens [11]
2. fortlaufende Nummer mit einer oder mehreren Zahlenreihen, die zur Identifizierung des Geschäftsvorfalls einmalig vergeben werden [11]
3. Tag der Belegausstellung [11]
4. Menge und handelsübliche Bezeichnung der Ware oder Dienstleistung [11]
5. Betrag der Barzahlung [11]
6. bei Verwendung von elektronischen Kassen mit Sicherheitseinrichtung: Kassenidentifikationsnummer, Datum und Uhrzeit der Belegausstellung, Betrag der Barzahlung nach Steuersätzen getrennt, maschinenlesbarer Code (OCR-, Bar- oder QR-Code) [11]

Punkt 1 ist gleichgestellt mit dem Namen des Unternehmens, zum Beispiel „Fake GmbH“. Punkt 2 ist die fortlaufende Rechnungsnummer, welche jeden Beleg als einzigartig ausweist. Das Registrierkassensystem muss dafür Sorge tragen, dass eine Rechnungsnummer nicht mehrmals vergeben wird. Punkt 3 ist das Datum an dem die Rechnung/der Beleg erstellt wurde. Danach folgt in Punkt 4 die Auflistung der Waren und der dazugehörigen Beträge. Weiters, in Punkt 5 wird der gesamte Barbetrag ausgewiesen. Da eine Registrierkasse, laut RKSV, auch eine elektronische Kasse mit Sicherheitseinrichtung ist, muss zusätzlich noch die Uhrzeit ersichtlich sein, wann der Beleg erstellt wurde. Zudem muss der Betrag der Barzahlung nach Steuersätzen getrennt aufgelistet werden. Zu guter Letzt, müssen die genannten Informationen plus zusätzliche Daten in einem QR-Code gesammelt ausgewiesen werden. Ein QR-Code ermöglicht es auf möglichst kleinem Raum viele Informationen darstellen zu können. Die Bestandteile eines maschinenlesbaren Codes, in diesem Falle eines QR-Codes, wird im Kapitel QR-Code auf der Seite 26 genauer erläutert. Für eine detailliertere Auflistung wird ebenso auf die RKSV [1] verwiesen.

[1] Link zur Verordnung: [3]

Fake GmbH

fakestreet 123
0650 123456789
company@chello.at
AT123456789
OrderID: 2
TableNr.: 8
2021-06-11
12:53:53

Item	Qty	Price	Total
Test Produkt	2	€ 1.0	€ 2.0

Total amount: € 2.0

MWST	Net	Tax	Gross
20%	1.67	0.33	2.00

Manuel served you!
Thank you for your purchase.
We look forward to seeing you again soon.

Abbildung 3: Standardbeleg

Abbildung 3 stellt einen standardisierten Kundenbeleg dar, worauf alle wesentlichen Punkte abgebildet sind. Zusätzlich sollte erwähnt werden, dass all die gezeigten Informationen auf dem Beleg rein fiktiv sind und nur zur Veranschaulichung des Sachverhalts dienen. Für diesen Zweck wurde eingangs ein Testprodukt erstellt und mit einem Euro beziffert. Weiters wurde das

Produkt mit 20% Mehrwertsteuer versehen. Zusätzlich ist in Abbildung 2 und 3 ersichtlich, dass eine Tischnummer angegeben ist. Das ist deshalb der Fall, weil die vorgestellte Registrierkasse für eine kleine Gaststätte konzipiert wurde, mehr dazu im Kapitel Hauptbestandteile des POS-Systems auf Seite 28.

2.7.3 Nullbeleg

Der sogenannte Nullbeleg unterscheidet sich nur marginal von einem Startbeleg. Die meisten Unterschiede sind nur in ihrer Abwicklung systemintern erkennbar. Diese Belegart dient ausschließlich dazu, um systemintern eine Art Bestandsaufnahme zu tätigen. Das wird meistens am Jahresende, Monatsende bzw. allgemein erstellt, wenn man belegen möchte, wie hoch der Umsatz bis zum Erstellen des Nullbelegs ist. Dieser Beleg ist ebenfalls notwendig, weil das Bundesministerium für Finanzen sich das Recht einräumt, unangekündigt Kontrollen und Bestandsaufnahmen von Registrierkassensystemen durchzuführen. Deshalb muss jederzeit eine Bestandsaufnahme möglich sein. Die relevanten Daten für das Bundesministerium für Finanzen sind im QR-Code kodiert hinterlegt, mehr dazu im Kapitel QR-Code auf der Seite 26.

2.7.4 Stornobeleg

Die nächste essentielle Belegart ist der sogenannte Stornobeleg. Dieser Beleg wird, wie der Name selbst schon vermuten lässt, zur Stornierung zuvor angelegter Belege verwendet. Es gibt durch die kettenartige Verbundenheit der Belege keinerlei Möglichkeit, nachträglich Belege zu ändern oder zu löschen, falls diese fehlerhaft sind. Für diesen Fall wird ein eigener Beleg erstellt der, wie alle Belege, ebenfalls in der Belegkette gespeichert wird. Dieser Beleg erzeugt jedoch keinerlei Umsatz, im Gegenteil er minimiert diesen. Wenn also beim Bezahlvorgang ein Produkt zu viel oder gar das falsche Produkt bezahlt wurde und dieser Vorfall erst nachträglich reklamiert wird, kann dieser fehlerhafte Beleg mittels Stornobeleg korrigiert werden. In Abbildung 3 auf der Seite 23 ist ersichtlich, dass zwei Produkte bezahlt wurden. Sollte jedoch ursprünglich nur ein Produkt dieser Kategorie bezahlt werden, kann mittels Beleg in Abbildung 4 diesem Umstand Rechnung getragen werden und das entsprechende Produkt wird storniert.

Fake GmbH

fakestreet 123
0650 123456789
company@chello.at
AT123456789
OrderID: 3
TableNr.: 8
2021-06-11
13:40:03

Item	Qty	Price	Total
Test Produkt	1	€ 1.0	€ 1.0

Total amount: € 1.0

MWST	Net	Tax	Gross
20%	0.83	0.17	1.00

Manuel served you!
Thank you for your purchase.
We look forward to seeing you again soon.

Abbildung 4: Stornobeleg

Dieser Beleg unterscheidet sich, in seiner Darstellung, nur in der Rechnungsnummer vom Standardbeleg aus Abbildung 3 auf der Seite 23 und in der

Anzahl der Produkte. Da in dem zuvor erwähntem Beispiel ein Produkt zu viel gekauft wurde, wird in Abbildung 4 eben eines davon storniert. Auf den Belegen selbst ist kein Unterschied zu erkennen, jedoch systemintern werden unterschiedliche Methoden ausgelöst und es wird innerhalb des QR-Codes verzeichnet, dass dies eine Stornobuchung war. Im Kapitel Hauptbestandteile des POS-Systems auf Seite 28 wird der Umgang mit den unterschiedlichen Belegarten abermals thematisiert, jedoch mit Bezug auf die Interaktion mit dem Registrierkassensystem selbst.

2.7.5 Trainingsbeleg

Der Gesetzgeber sieht weiters vor, dass ebenfalls Trainingsbuchungen möglich sein müssen. Diese werden, wie alle Belegarten, in die Verkettung integriert, jedoch löst eine Trainingsbuchung bzw. ein Trainingsbeleg keine systeminternen Reaktionen bezüglich des Umsatzzählers aus, da solch eine Buchung bzw. ein Beleg ausschließlich für Trainings- oder Demonstrationszwecken benutzt wird. Der Rechnungsbeleg unterscheidet sich ebenfalls nicht offensichtlich von anderen Belegen, jedoch wird ebenfalls innerhalb des QR-Codes dafür Sorge getragen, dass dies eine Trainingsbuchung ist, mehr dazu ebenfalls im Kapitel Hauptbestandteile des POS-Systems auf der Seite 28.

2.8 QR-Code

Wie im Kapitel Standardbeleg auf der Seite 22 unter Punkt 6 beschrieben, ist es für eine elektronisches Kasse zwingend notwendig, dass Belege mit einem maschinenlesbarer Code versehen werden. Aus diesem Grund wurde eine Funktion in das System integriert, welches ermöglicht etwaige QR-Codes zu generieren. Die technische Implementierung ist im Anhang im Kapitel QR-Code auf der Seite 117 dargestellt. QR-Codes (Quick Response, zu Deutsch schnelle Antwort) dienen dazu, dass alle wichtigen Informationen sowohl schnell, als auch maschinell abrufbar sind. Die Informationen werden in einen zweidimensionalen Code übersetzt und dargestellt [12]. Die Registrierkassensicherheitsverordnung regelt ebenso welche Informationen innerhalb eines QR-Codes ersichtlich sein müssen, in welcher Reihenfolge diese dargestellt werden und definiert ebenfalls die maximale Größe eines Codes. Unterhalb wurde der QR-Code aus Abbildung 3 auf der Seite 23 ausgelesen und dessen Inhalt zur Schau gestellt. Weiters erfolgt eine nähere Betrachtung der einzelnen Punkte.

1. R1-AT1
2. CASHBOX1

3. 2
4. 2021-06-11T12:53:53
5. 2,00
6. 0,00
7. 0,00
8. 0,00
9. 0,00
10. E0bQfNs=
11. 2159b9f6
12. Oopx4Jla0X4=
13. OsKvdJPJ2yd5bDgcEZc9k2FdhcSz52QLSwcxPm3TT8Njq 6nipyILwKpNSIF3hAdIqmTukhQzCUb76SNtek1x0Q==

Punkt 1 wird durch die Signatur- bzw. Siegelerstellungseinheit des jeweiligen Vertrauensdiensteanbieters übermittelt. Jeder der in Kapitel Potentielle Vertrauensdiensteanbieter auf der Seite 15 erwähnten Anbieter hat seine eigene Abkürzung. In diesem Fall ist R1 die Bezeichnung für den verwendeten Registrierkassenalgortihmus und AT1 steht für den ersten Anbieter, nämlich A-Trust. Diese Information wird nicht vom System selbst erzeugt, sondern wird entweder durch ein entsprechendes Kartenlesegerät, inklusive Karte mit Zertifikatsdaten, oder durch die Signierung über das Internet übermittelt. Diese Kürzel dienen dazu, dass auf jedem Beleg ausgewiesen wird, welcher Algorithmus von welchem Anbieter kam zur Signierung zum Einsatz.
Punkt 2 spiegelt die Kassenidentifikation wieder. Bei Initialisierung des Systems wird eine Kassenidentifikation verlangt. In Abbildung 7 auf der Seite 33 ist ersichtlich, dass dies jedmögliche Bezeichnung sein kann. Im Falle dieses Beispiels, lautet die Kassenidentifikation des Systems „CASHBOX1“ (zu deutsch KASSE1). Der gewählte Name der Registrierkasse muss ebenfalls an das Bundesministerium für Finanzen, zum Zwecke der Kontrolle, gemeldet werden und darf innerhalb des Betriebes nur einmal vergeben werden. Jedoch ist es möglich, dass zwei eigenständige Unternehmen mit jeweils einer separaten Kasse ein und die selbe Kassenbezeichnung verwenden.
Punkt 3 ist die ausgewiesene Rechnungsnummer für diesen Beleg. Da zuvor

lediglich die Initialisierung des System statt gefunden hat, durch die Erstellung eines Startbelegs, ist dies der zweite erstellte Beleg des Systems.
Punkt 4 stellt das Datum und die genaue Uhrzeit des jeweiligen Belegs dar, zum Zeitpunkt der Bezahlung.
Die Punkte 5 bis 9 geben die Beträge gesammelt nach Steuersätzen an. Die Reihenfolge dieser Auflistung ist ebenfalls in der Registrierkassensicherheitsverordnung definiert und lautet wie folgt: 20%, 19%, 13%, 0% und 10%. Es ist zudem zu erkennen, dass einzig und alleine ein Produkt mit 20 prozentigem Steuersatz bezahlt wurde und das dieses Produkt einen Verkaufswert von zwei Euro aufweist.
Punkt 10 weist den aktuellen und verschlüsselten Umsatzzähler aus. Dieser dient zur nachträglichen Kontrolle durch das Bundesministerium für Finanzen und muss dementsprechend ausgewiesen werden, jedoch in verschlüsselter Form, damit keine unbefugte Person Einsicht auf sensible Daten nehmen kann. Mehr dazu im Kapitel Generelle Rahmenbedingungen eines Registrierkassensystems auf der Seite 14 und im Anhang im Kapitel Kryptografie auf der Seite 115, wo die technische Implementierung dargestellt wird.
Punkt 11 stellt die Seriennummer des Signaturzertifikates, welches dazu benutzt wurde um diesen Beleg zu signieren, dar.
Punkt 12 ist der Signaturwert des vorhergehenden Belegs, nähere Informationen darüber sind in Kapitel Lückenlose Belegerstellung auf der Seite 17 zu finden.
Und zu guter Letzt, Punkt 13 ist der Signaturwert des Belegs selbst, auch dazu gibt es nähere Information im Kapitel Lückenlose Belegerstellung auf Seite 17.

Zusammenfassend, der QR-Code ist ein Sammelsurium an wichtigen Informationen, der zudem auch noch von Maschinen gelesen und verarbeitet werden kann.

3 Hauptbestandteile des POS-Systems

In den vorhergehenden Kapitel ging es zunehmend um technische oder gesetzliche Voraussetzungen eines POS-Systems. Der Schwerpunkt dieses Kapitels ist die Erläuterung und Darstellung expliziter Programminhalte, des entwickelten Registrierkassensystems. Das POS-System ist im Stande unterschiedliche Rollen eines Benutzers definieren zu können. Angefangen mit der Admin-, weiterführend zur Angestellten, bis hin zur Auszubildenden-Rolle. Diese Rollen wurden speziell gewählt, da unterschiedliche Mitarbeiter und Angestellte unterschiedliche Berechtigungen im Unternehmen haben. Es ist zumeist so, dass neue Mitarbeiter weniger Einblick und damit Zugang zu

geschäftsinternen Informationen benötigen, als dies zum Beispiel bei einem Manager oder einem leitenden Angestellten der Fall ist. Das Kapitel ist in mehrere Unterkapitel gegliedert, was einen besseren Überblick über die Funktionalitäten schafft. Zunächst wird die Administratorrolle näher beleuchtet und zwar die Funktionalitäten, welche ausschließlich nur von dieser Rolle benutzt werden können. Jede weitere untergeordnete Rolle, hat keinen Zugriff auf die Methoden einer höherer kategorisierten Benutzerrolle. Jedoch ist der Administrator in der Lage, jedmögliche Funktion aufrufen zu können, welche die beiden anderen Rollen inne haben. Kurz gesagt, ein Administrator kann alles, ein Angestellter weniger und ein Auszubildender noch weniger.

3.1 Administrator

Wie bereits erwähnt, gibt es die Möglichkeit einer Administratorrolle. Diese Rolle ist die Mächtigste aller bereits genannten Rollen, weil diese Zugriff auf die gesamte Funktionalität des Systems, sowie Einsicht in Unternehmensdaten hat. Aus diesem Grund wird, von Seiten der Entwicklung des System, empfohlen, dass nur ausgewählte und vertrauenswürdige Personen diese Rolle inne haben. Die Unternehmensdaten, welche durch einen Administrator eingesehen werden können, erstrecken sich von weniger sensiblen Daten, wie Anschrift und Telefonnummer des Unternehmens, bis hin zu höchst sensiblen Benutzerdaten von Mitarbeitern und Schlüsseln, mithilfe dessen Verschlüsselungen generiert werden. Aus oben genannten Gründen, sollten nur ausgewählte Personen diese Berechtigung haben. In den nachfolgenden Unterkapitel, wird nun Schritt für Schritt erläutert, welche Methoden und Funktionen ausschließlich von einem Administrator angesteuert werden können.

Das System selbst sieht vor, dass zumindest ein Administrator im System hinterlegt werden muss. Dieser wird unter anderem beim erstmaligen Starten des Programms angegeben. Dazu erscheint, nach dem abgeschlossenen Initialisierungsprozess des Systems ein Fenster, ersichtlich in Abbildung 5 unterhalb. Nähere Informationen bezüglich der Inbetriebnahme des Systems sind im Anhang im Kapitel Konfiguration auf der Seite 72 gegeben.

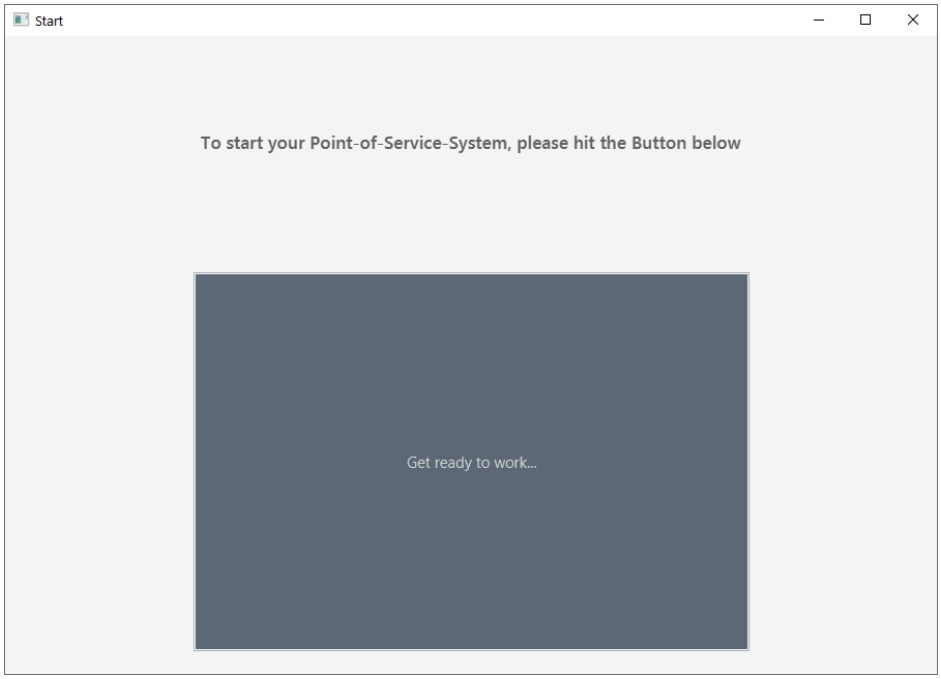

Abbildung 5: Betriebsbereit

Durch das mehrmalige Drücken des Knopfes („Get ready to work... “) in der Mitte der Abbildung 5, wird überprüft ob bereits zumindest ein Administrator vergeben wurde, wenn dies nicht der Fall ist, muss zunächst einer angelegt werden, siehe Kapitel Anlegen des Administrators unterhalb. Ist bereits ein Administrator hinterlegt, so wird überprüft ob relevante Unternehmensdaten angegeben wurden, siehe Kapitel Bekanntgabe der Unternehmensdaten auf der Seite 31. Sofern auch dies erfolgreich abgeschlossen ist, erfolgt der Login in das System, siehe Kapitel Login auf der Seite 33.

Die technische Umsetzung ist im Anhang, in entsprechenden Unterkapiteln, dargestellt.

3.1.1 Anlegen des Administrators

Nachdem das System erfolgreich in Betrieb genommen wurde und weiters der blaue Knopf, ersichtlich in Abbildung 5 auf der Seite 30, gedrückt wurde erscheint eine Eingabemaske. Die Eingabemaske, in Abbildung 6 dargestellt wird, benötigt Daten über den zu erstellenden Administrator, wie zum Beispiel Vor- und Nachname. Weiters benötigt das POS-System, für den späteren Loginprozess, einen Benutzernamen. Der Benutzername muss innerhalb des Systems einzigartig sein, das heißt der Benutzername darf nur einmal verge-

ben werden. Diese Restriktion stellt sicher, dass keinerlei Probleme während des Loginprozesses, verursacht durch mehrmaliges vergeben des selbigen Benutzernamens, auftreten könnten. Zu guter Letzt muss noch ein Passwort gewählt werden. Zu Demonstrationszwecken wurden das Passwort „123“ gewählt. Dieses ist nicht sicher und sollte deshalb niemals gewählt werden, auch wenn sich dadurch der Loginprozess in der Praxis beschleunigt. Dies gilt für alle nachträglich angelegte Benutzer, sowie für die Adiministratorrolle. Jedes Passwort sollte möglichst einen Klein- bzw. Großbuchstaben, sowie Zahlen und Sonderzeichen enthalten.

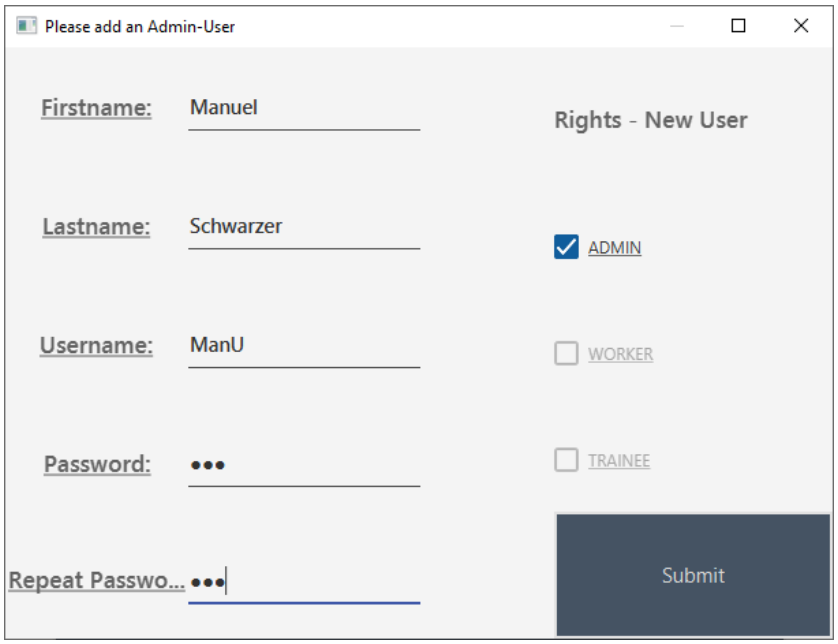

Abbildung 6: Administratorzuweisung

Nachdem alle relevanten Informationen angegeben wurden, müssen diese, durch das Klicken des Submit-Buttons, bestätigt werden. Nachdem dies erfolgt ist, müssen Daten über das Unternehmen selbst angegeben werden. Mehr dazu in Kapitel Bekanntgabe der Unternehmensdaten.

Die technische Umsetzung ist im Anhang im Kapitel Administratorverwaltung auf der Seite 76 dargestellt.

3.1.2 Bekanntgabe der Unternehmensdaten

Abbildung 7 auf der Seite 33 demonstriert, wie die Bekanntgabe der Daten erfolgt. Diese Daten werden zum Teil benötigt, um einen entsprechenden Be-

leg drucken zu können. Wie im Kapitel Belegarten auf der Seite 20 bereits erwähnt wurde, benötigt man zum Erstellen eines Belegs unterschiedliche Daten vom belegerstellenden Unternehmen. Weiters wurde im Kapitel QR-Code auf der Seite 26 kurz erwähnt, dass als Bestandteil eines QR-Codes ein verschlüsselter Umsatzzähler angegeben werden muss. Die Art und Weise wie die Verschlüsselung vonstatten gehen muss wird in der Registrierkassensicherheitsverordnung genauestens definiert. Kurz gesagt, muss der Umsatzzähler mittels AES256-Verschlüsselung verschlüsselt werden. AES256 steht für „Advanced Encryption Standard“ mit einer Schlüssellänge von 256-bit [13]. Mithilfe der AES-Verschlüsselung ist es möglich Daten mit ein und dem selben Schlüssel sowohl ver- als auch entschlüsseln zu können. Die RKSV sieht zudem vor, dass der AES-Schlüssel an das Bundesministerium für Finanzen gemeldet werden muss. Die Übermittlung des Schlüssels ist notwendig, damit das Bundesministerium für Finanzen Zugriff auf die Daten des verschlüsselten Umsatzzählers hat und somit dessen Entstehung kontrollieren kann. Für Unbefugte, Personen ohne Kenntnis des AES-Schlüssels, ist es demnach unmöglich den Umsatzzähler zu entschlüsseln und somit unberechtigter Weise an Unternehmensdaten gelangen zu können. Die Abbildung 7 auf der Seite 33 verlangt unterdessen noch einen weiteren Schlüssel, den sogenannten „JWS“-Schlüssel.
Der „JWSKey“ wird zur späteren Signierung, durch die Offline-Variante, benötigt, siehe Kapitel Signierungsart wählen auf der Seite 53. Da die beiden Schlüssel, AES- wie JWS-Schlüssel, essenziell für das System, aber nicht intuitiv für Jedermann verständlich sind und einem bestimmten Muster folgen müssen, ist ein eigener Button integriert worden, der es ermöglicht sowohl einen AES-Schlüssel, als auch einen JWS-Schlüssel zufällig, vom System selbst, generieren zu lassen. Somit ist gewährleistet, dass die Schlüssel dem richtigen Muster entsprechen und sogleich wird der Vorgang der Schlüsselfindung beschleunigt. Die beiden erzeugten Schlüssel werden in separate Dateien in separaten Verzeichnissen gespeichert, sodass diese nicht verloren gehen können. Weiters werden sie ebenfalls in der zugehörigen Datenbank hinterlegt, für die spätere Verwendung eben dieser.

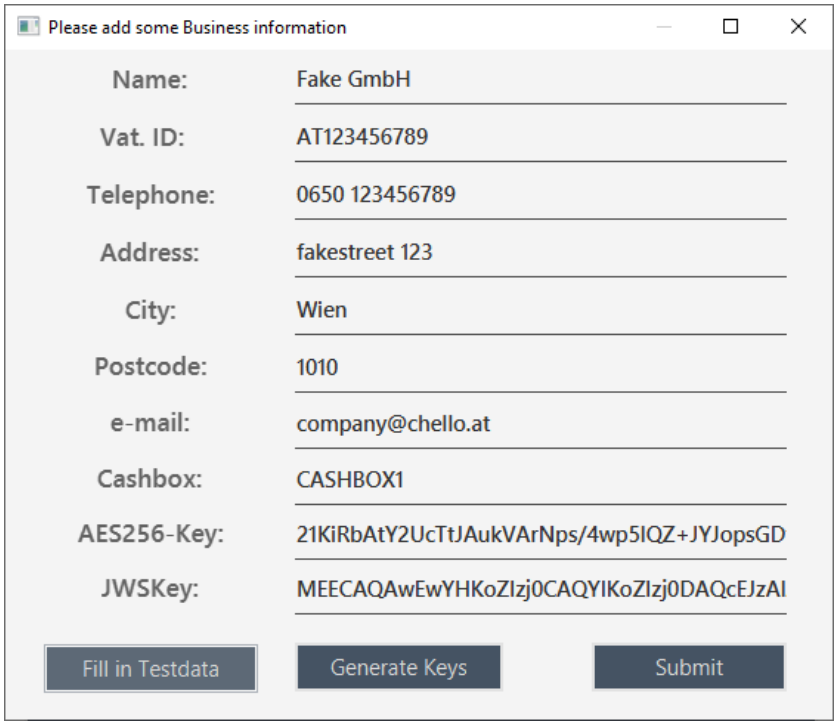

Abbildung 7: Unternehmensdaten

Für Demonstrations- oder Schulungszwecke können Testdaten eingespielt werden, mittels Knopfdruck „Fill in Testdata“ (zu Deutsch Testdaten einfügen). Sinn und Zweck dieses Vorgangs ist es, dass Systemneulinge den Umgang mit dem System erlernen können, ohne viel Zeit für die Verwaltung des Systems aufbringen zu müssen. Sofern das System in der Praxis eingesetzt werden soll, müssen die gezeigten Testdaten durch reale Daten, des benutzenden Betriebs, ersetzt werden. Sind die Daten, laut Benutzer, korrekt, müssen diese wiederum mittels „Submit“-Knopfdruck bestätigt werden. Sofern auch diese Eingabemaske erfolgreich bestätigt wurde, erfolgt nun der erste Login, wie in Abbildung 8 auf der Seite 34 erkennbar ist.

Die technische Umsetzung der Unternehmensdatenverwaltung ist im Anhang im Kapitel Unternehmensdaten auf der Seite 77 dargestellt.

3.1.3 Login

Der erste Login überhaupt findet mit den zuvor angelegten Administratordaten, wie Benutzername und Passwort statt. Im Anhang im Kapitel Passwortverwaltung auf der Seite 114 ist ersichtlich wie Passwörter verarbeitet

und gespeichert werden, sodass diese vor unbefugtem Zugriff geschützt sind. Einzig der Benutzer selbst kennt das konkrete Passwort.

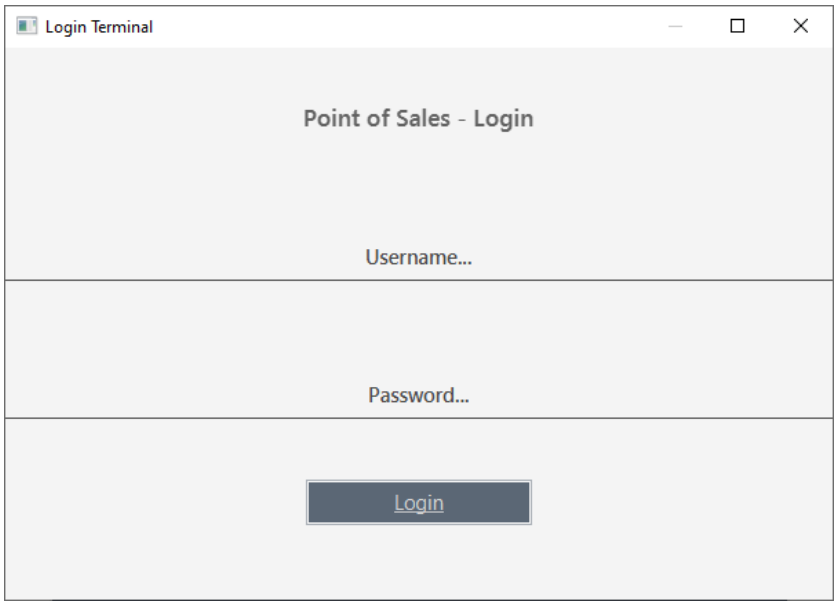

Abbildung 8: Login-Terminal

Sofern die Daten, Benutzername und Passwort eine Benutzers, korrekt eingegeben und vom System überprüft wurden, erscheint das Hauptmenü. Siehe nächstes Kapitel.

Die technische Umsetzung des Logins ist im Anhang im Kapitel Login-verwaltung auf der Seite 86 dargestellt.

3.1.4 Hauptprogramm

Nachdem der Login erfolgreich war, erscheint das Hauptmenü. Abbildung 9 auf der Seite 35 ist der Dreh- und Angelpunkt des gesamten Systems.

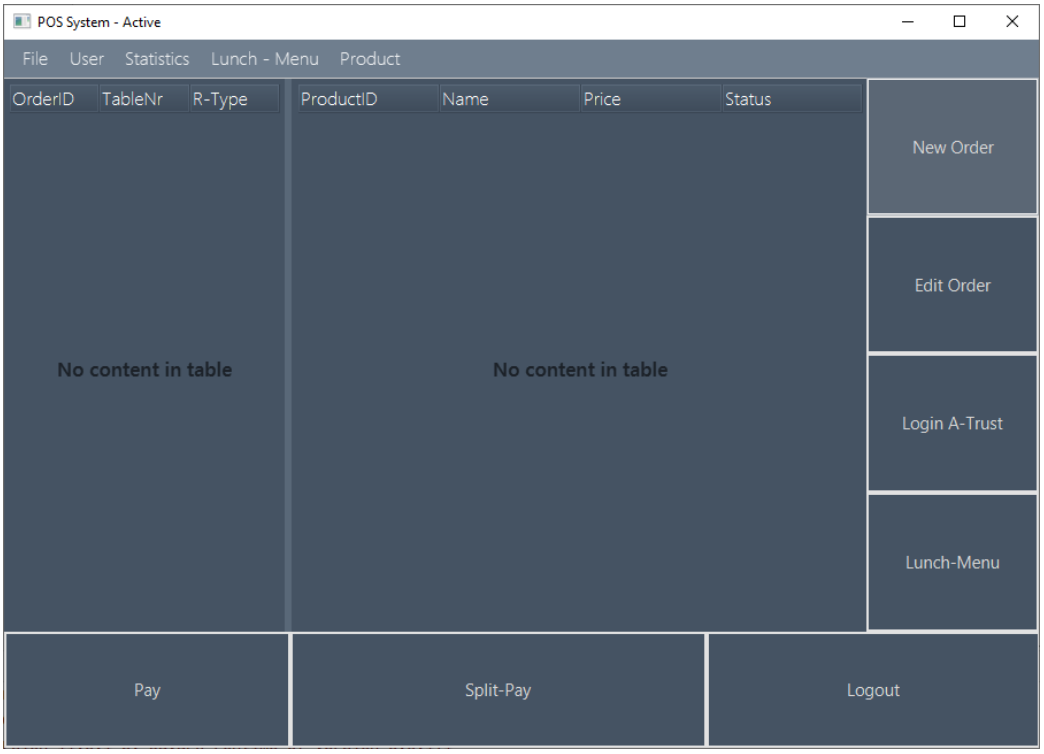

Abbildung 9: Hauptfenster des Registrierkassensystems

Wie in Abbildung 9 ersichtlich, sind alle Menüpunkte freigegeben. Dies ist der Fall, weil ein Benutzer mit Administratorrechten eingeloggt ist. Sofern sich ein anderer Benutzer, dem eine niedrigere Rolle zugewiesen wurde, einloggt, ändert sich die Darstellung des Hauptfensters, in dem Menüpunkte ausgegraut werden, für die die entsprechende Berechtigung fehlt. Wie bereits erwähnt sieht das System vor, dass einzig und allein ein Benutzer mit Administratorstatus dazu berechtigt ist, unternehmensinterne Daten einsehen zu dürfen. Darunter fallen zum Beispiel die eingangs eingegebenen Unternehmensdaten, siehe Abbildung 7 auf der Seite 33. Weiters ist es nur dem Administrator selbst gestattet neue Benutzer anzulegen. Dies hat sicherheitstechnische sowie datenschutzrechtliche Gründe, für nähere Informationen über die Erstellung etwaiger Benutzer siehe Kapitel Benutzerverwaltung.

Die technische Umsetzung ist im Anhang im Kapitel Hauptprogramm auf

der Seite 87 dargestellt.

3.1.5 Benutzerverwaltung

Mithilfe des Menüpunkts „User“ (zu Deutsch Benutzer) und dem Unterpunkt „New User“ (zu Deutsch Neuer Benutzer) kann ein weiterer Benutzer angelegt werden. Abbildung 10 auf der Seite 36 verdeutlicht diesen Umstand.

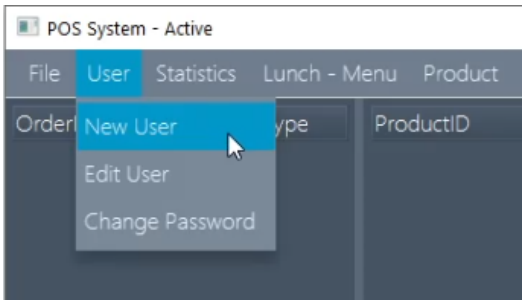

Abbildung 10: Menüpunkt: Neuer Benutzer

Anschließend öffnet sich ein eigenständiges Fenster, mithilfe dessen ein neuer Benutzer angelegt werden kann, siehe Abbildung 11 auf der Seite 37. Diese Eingabemaske ist ident mit der auf dem Kapitel Anlegen des Administrators auf der Seite 30 zuvor, in der eingangs ein Administrator definiert wurde, jedoch mit dem signifikanten Unterschied, dass jetzt weitere Rollen für den Benutzer vergeben werden können. Ein neuer Benutzer kann entweder ein Administrator sein („Admin“), ein Angestellter („Worker“) oder ein Auszubildender („Trainee“).

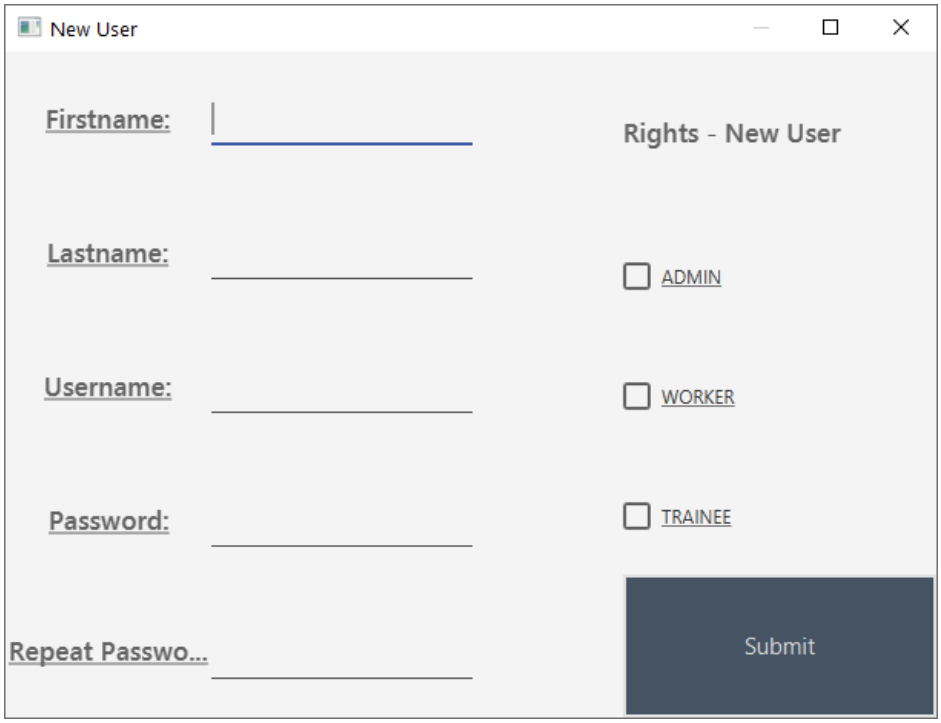

Abbildung 11: Neuer Benutzer anlegen

In Abbildung 11 ist zu erkennen, dass ebenfalls teils sensible Benutzerdaten angegeben werden müssen. Aus bereits genannten Gründen ist dies nur einem Administrator vorbehalten. Um etwaige Daten eines Benutzers ändern zu können, wie zum Beispiel den Nachnamen, so muss der Unterpunkt „Edit User“ (zu Deutsch Benutzer bearbeiten), ausgewählt werden. Siehe Abbildung 12 auf der Seite 37.

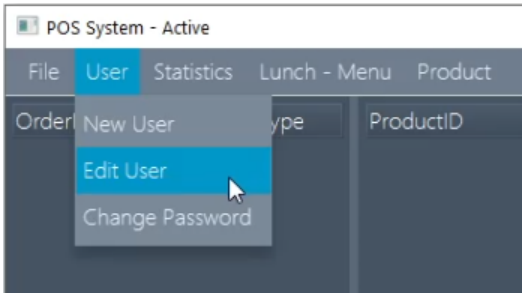

Abbildung 12: Menüpunkt: Benutzer bearbeiten

Es erscheint wiederum ein eigenständiges Fenster, welches alle bisher angelegten Benutzer mit den wichtigsten Daten, überblicksmäßig darstellt. Wie in Abbildung 13 auf der Seite 38 erkennbar ist, wurde bis jetzt nur ein Administrator angelegt.

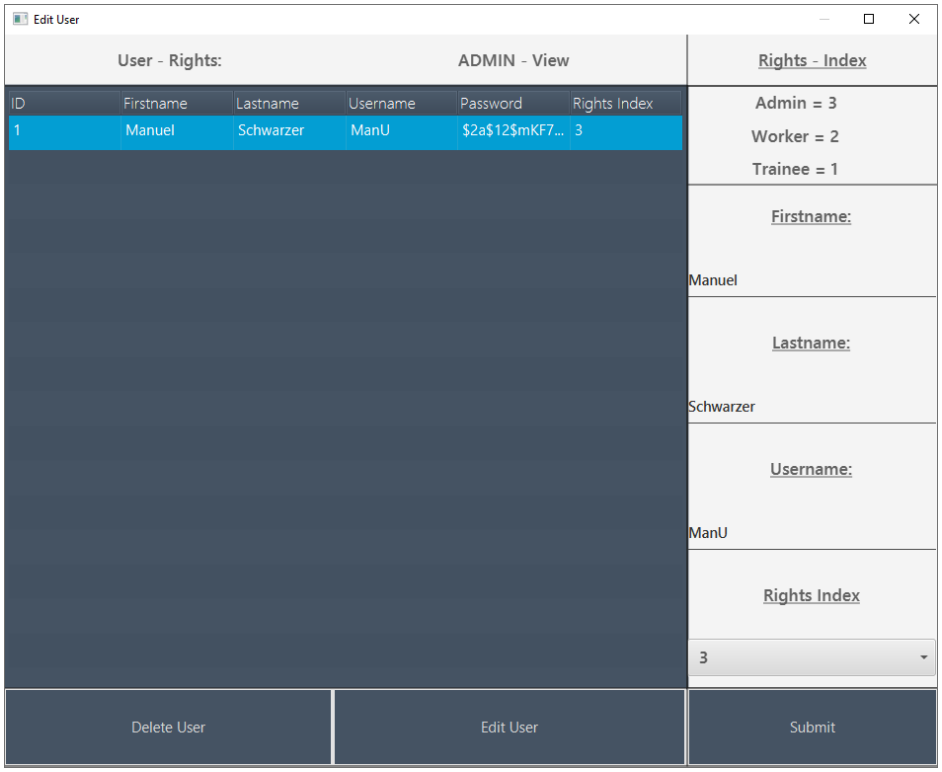

Abbildung 13: Benutzer bearbeiten

Des Weiteren ist in Abbildung 13 erkennbar, dass das Passwort nicht im Klartext dargestellt wird, sondern anhand verschiedener zusammenhangsloser Zeichen. Nähere Informationen über die Passwortverwaltung befinden sich im Anhang im Kapitel Passwortverwaltung auf der Seite 114. Um Daten, wie zum Beispiel Vor- bzw. Nachname, ändern zu können, muss in der Tabelle eine Reihe, besser gesagt ein Benutzer, ausgewählt werden. Nach dem ein Benutzer ausgewählt ist, erkennbar durch den blauen Hintergrund der Reihe, muss der Button „Edit User“ gedrückt werden. Danach können die Daten, welche in der rechten Spalte des Fensters erscheinen, geändert werden und mittels „Submit“ bestätigt werden. In der rechten Spalte kann ebenfalls die Rolle des Benutzers geändert werden, immer mit dem Hintergedanken, dass zumindest ein Administrator bestehen bleiben muss. Zu guter Letzt ist es

möglich, innerhalb dieses Fensters, auch Benutzer zu löschen.

Die technische Umsetzung ist im Anhang im Kapitel Benutzerverwaltung auf der Seite 80 dargestellt.

3.1.6 Umsatzzähler

Die Administratorrolle ist zudem ermächtigt Einsicht in die bestehenden Umsatzzahlen, der Registrierkasse, zu nehmen. Dazu ist unter dem Menüpunkt „Statistics“ ein Unterpunkt eingegliedert siehe Abbildung 14, mithilfe dessen sich ein GUI (Graphical User Interface, zu Deutsch grafische Benutzeroberfläche), mit den entsprechenden Informationen öffnet,. Dies erzeugte Benutzeroberfläche ist in Abbildung 15 auf der Seite 39 dargestellt.

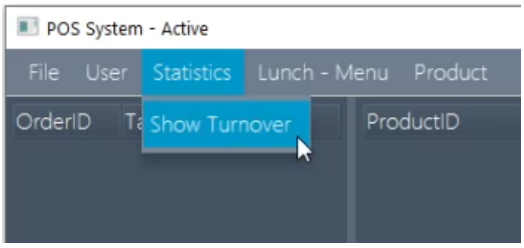

Abbildung 14: Menüpunkt: Umsatzzähler

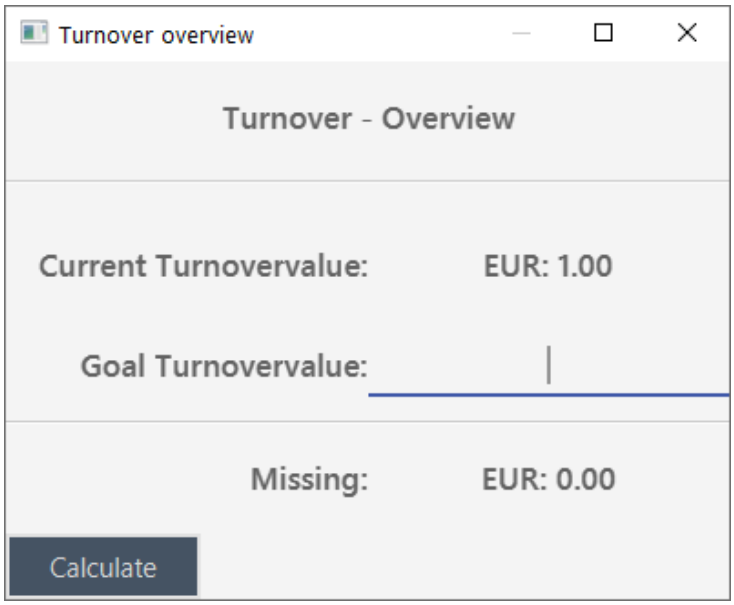

Abbildung 15: Umsatzzähler

Abbildung 15 ermöglicht es zudem, dass eine Vergleichsmarke gesetzt werden kann, woraufhin sich das Fenster entsprechend anpasst. Ist die Vergleichsmarke höher, sprich ist das Umsatzziel noch nicht erreicht, ändert sich das Fenster, wie in Abbildung 16 dargestellt, ist das Ziel erreicht und der Wert darüber, wird das Fenster entsprechend wie in Abbildung 17 dargestellt.

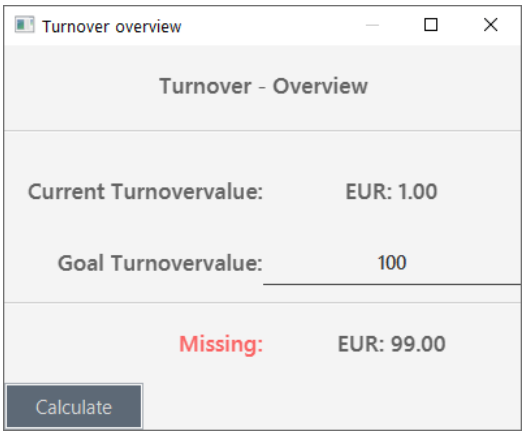

Abbildung 16: Umsatzziel noch nicht erreicht

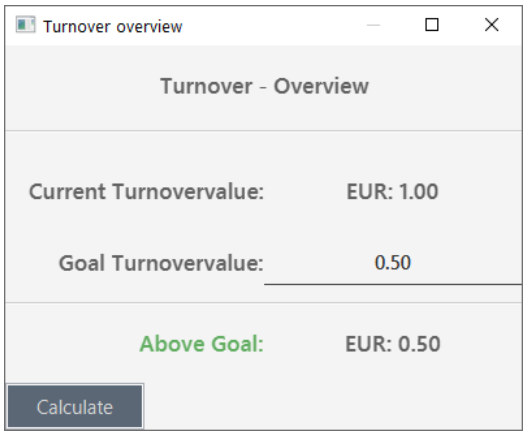

Abbildung 17: Umsatzziel erreicht

Im Kapitel Generelle Rahmenbedingungen eines Registrierkassensystems auf der Seite 14 wird erwähnt, dass es für ein RKSV-konformes Registrierkassensystem zwingend erforderlich ist, einen Umsatzzähler zu führen. Das bedeutet im Grunde nichts weiter, als dass der Barbetrag jedmöglicher Transaktion, welche den Umsatz beeinträchtigt, mit protokolliert werden muss. Somit soll sichergestellt werden, dass sowohl jeder Beleg gespeichert wird, als auch die Barumsätze separat gespeichert werden. Für die technische Erläuterung der Verschlüsselung dieses Zählers ist im Anhang im Kapitel Kryptografie auf der Seite 115 der entsprechende Source-Code abgebildet.

Die technische Umsetzung in Bezug auf die Darstellung und Handhabung bezüglich des Umsatzzählers ist im Anhang im Kapitel Umsatzzähler auf der Seite 84 dargestellt.

3.1.7 Export des Datenerfassungsprotokolls

Der letzte wichtige Menüpunkt, der nur als Administrator ausgeführt werden kann, ist der Export des Datenerfassungsprotokolls. Der Export erfolgt mittels Unterpunkt, ersichtlich in Abbildung 18 auf der Seite 41.

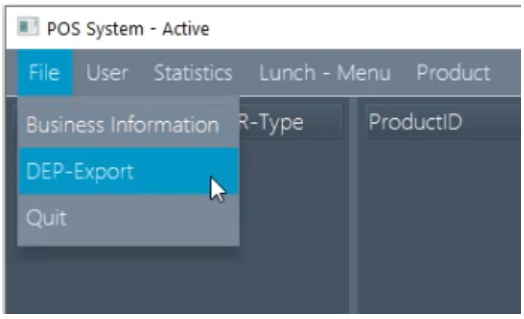

Abbildung 18: Menüpunkt: DEP-Export

Sofern der Export erfolgreich war, erscheint ein Informationsfenster, welches den erfolgreiche Export bestätigt. Ein entsprechender Export ist im Anhang im Kapitel DEP-Export auf der Seite 142 dargestellt. Die Registrierkassensicherheitsverordnung definiert nicht nur die Rahmenbedingungen für ein Registrierkassensystem selbst, sondern definiert auch technische Aspekte, wie das Design der erstellten Export-Datei. Dies wird ebenfalls im Anhang näher erläutert.

Die technische Umsetzung ist im Anhang im Kapitel Hauptprogramm auf der Seite 87 dargestellt. Genauer gesagt in der Auflistung collection.rexx.

3.2 Angestellter

Die bisher gezeigten Funktionen können lediglich von einem Benutzer mit Administratorstatus ausgeführt werden. Die nachfolgenden Funktionen können sowohl von einem Administrator, als auch von einem Benutzer, der die Rolle „Angestellter“ inne hat, ausgeführt werden. Als Angestellter besteht die Möglichkeit sein Passwort ändern zu können. Zudem ist es möglich etwaige verwaltungstechnische Aufgaben, wie die Verwaltung der Mittagsmenü-Karte, die Verwaltung von Produkten oder Bestellungen zu übernehmen. Weiters ist es Benutzern, mit Angestelltenstatus, möglich den Bezahlvorgang einzuleiten. Nähere Information diesbezüglich befinden sich in den darauffolgenden Unterkapiteln.

3.2.1 Passwort ändern

Zunächst einmal muss erwähnt werden, dass Benutzer, welche zuvor von einem Administrator angelegt wurden, ein temporäres Passwort zugeteilt bekommen. Temporär deshalb, weil ein Benutzerpasswort nur dem Benutzer selbst bekannt sein sollte. Aus diesem Grund ist es ratsam, als ersten

Schritt das Passwort zu ändern. Dies erfolgt unter dem Menüpunkt „User“ (zu Deutsch Benutzer) und „Change Password“ (zu Deutsch Passwort ändern), siehe Abbildung 19 unterhalb.

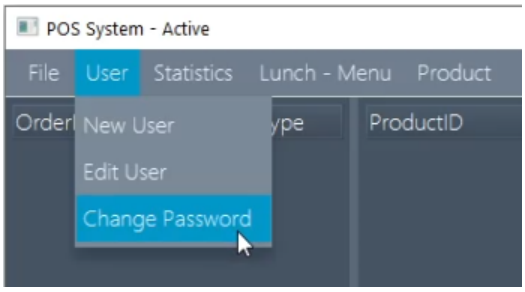

Abbildung 19: Menüpunkt: Passwort ändern

Als nächster Schritt erscheint ein Fenster, innerhalb dessen das Passwort geändert werden kann. Zunächst wird das alte Passwort verlangt. Danach ist es möglich ein neues Passwort zu wählen, siehe Abbildung 20 auf der Seite 42. Das neuerstellte Passwort sollte möglichst sicher sein, sprich zumindest einen Klein- und Großbuchstaben, sowie eine Zahl und ein Sonderzeichen enthalten.

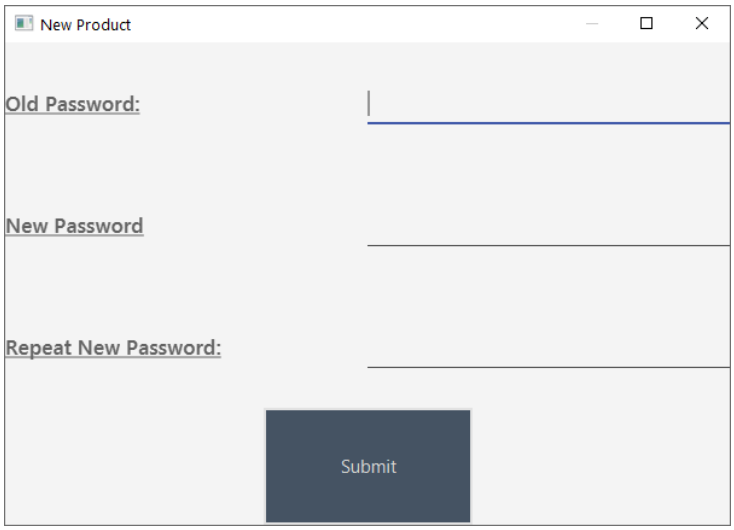

Abbildung 20: Passwort ändern

Nach erfolgreicher Bestätigung ist das Passwort geändert und kann beim neuerlichen Loginprozess bereits verwendet werden.

Die technische Umsetzung ist im Anhang im Kapitel Passwortverwaltung auf der Seite 114 dargestellt.

3.2.2 Mittagsmenü verwalten

Als nächsten Menüpunkt, innerhalb des Hauptfensters, ist „Lunch-Menu“ angegeben. Mithilfe diesem kann eine Menükarte, für das Mittagsgeschäft, erstellt werden. Dazu muss der Unterpunkt, ersichtlich in Abbildung 21 unterhalb, ausgewählt werden.

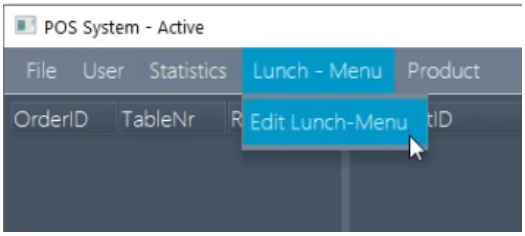

Abbildung 21: Menüpunkt: Mittagsmenü

Daraufhin erscheint ein Fenster, mithilfe dessen man etwaige Produkte auswählen und einer Kategorie zuweisen kann, siehe Abbildung 22 auf der Seite 44.

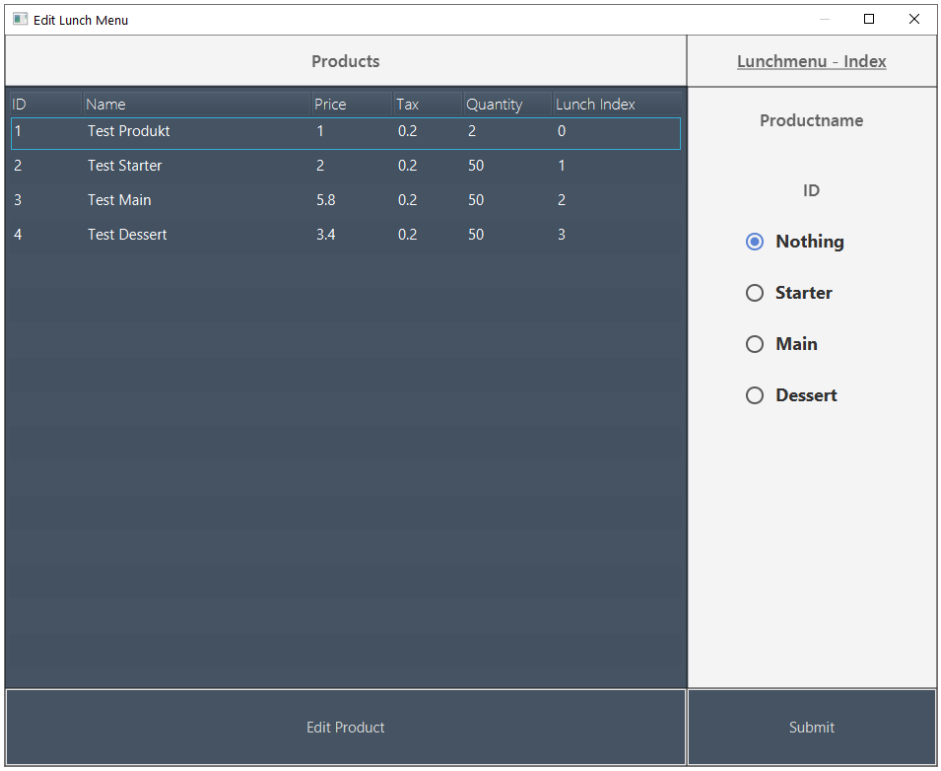

Abbildung 22: Mittagsmenü ändern

Wie in Abbildung 22 ersichtlich, wird der gesamte Lagerbestand an Produkten in tabellarischer Form aufgelistet. Zu Demonstrationszwecken wurden drei weitere Produkte dieser Liste hinzugefügt. Diese Produkte wurden, entsprechend ihres Namens, zugeteilt. Die Kategorie „Starter“ (zu Deutsch Vorspeise), wird hiermit nur dem Produkt zugeteilt, welches als Vorspeise des Menüs deklariert wurde. Die nächste Kategorie ist „Main“, was auf Deutsch so viel bedeutet wie „Hauptgang“. In diese Kategorie fällt ebenfalls nur das Produkt, was zu besagtem Zeitraum als Hauptgang deklariert wurde. Zu guter Letzt kann ein Produkt der Kategorie „Dessert“ (zu Deutsch Nachspeise) zugewiesen werden. Für alle anderen Produkte, die nicht Teil des Mittagsmenüs sind, ist die Kategorie „Nothing“ (zu Deutsch Nichts) vorgesehen.

Eine ebenfalls wichtige Tätigkeit, die durch das Registrierkassensystem unterstützt wird, ist das Anlegen und Verwalten von Produkten. Deshalb gibt es weitere Funktionen des Systems, die darauf bedacht sind Produkte und deren Bestand zu verwalten. Wie bereits in Abbildung 22 auf der Seite 44 veranschaulicht wird, wurden mehrere Produkte zu Demonstrationszwecken erstellt. Wie dies bewerkstelligt werden kann, wird im nachfolgenden Kapitel,

Kapitel Produkte verwalten, näher erläutert.

Die technische Umsetzung des Mittagsmenüs ist im Anhang im Kapitel Mittagsmenüverwaltung auf der Seite 107 dargestellt.

3.2.3 Produkte verwalten

Die im Kapitel Mittagsmenü verwalten auf der Seite 43 gezeigten Produkte wurden zuvor, über einen eigenständigen Menüpunkt und Eingabemaske, in das System integriert. Mittels Menüpunkt, ersichtlich in Abbildung 23 unterhalb, kann auf die Eingabemaske, dargestellt in Abbildung 24 auf der Seite 46, zugegriffen werden.

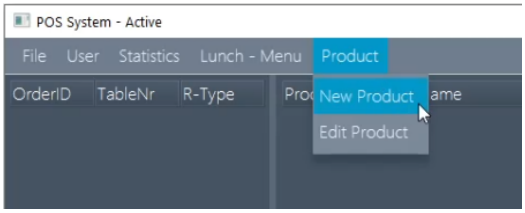

Abbildung 23: Menüpunkt: Neues Produkt

Damit man nun Produkte, wie in Abbildung 24 ersichtlich, anlegen kann, muss dazu die rechte Spalte ausgefüllt werden. In dieser Spalte werden essentielle Informationen über das Produkt abgefragt, wie der Name und der dazugehörige Preis. Des Weiteren wird die vorhandenen Menge abgefragt, damit auch immer der Lagerbestand aktuell gehalten wird. Zudem ist es möglich etwaigen Engpässen beim Bestellvorgang vorzubeugen, da der Lagerbestand von Produkten immer aktuell gehalten wird. Weiters wird ein „Lunch-Index“ (zu Deutsch Mittagessen-Index) verlangt, welcher bereits im Kapitel Mittagsmenü verwalten auf der Seite 43 behandelt wurde. Zu guter Letzt muss dem anzulegenden Produkt noch ein Steuersatz zugewiesen werden. Dieser kann, so wie der Mittagsmenü-Index, mittels Dropdown-Menü aus einer vordefinierten Liste gewählt werden. Mit der Bestätigung der Daten, mittels Knopfdruck des „Submit“-Buttons, wird das Produkt angelegt.

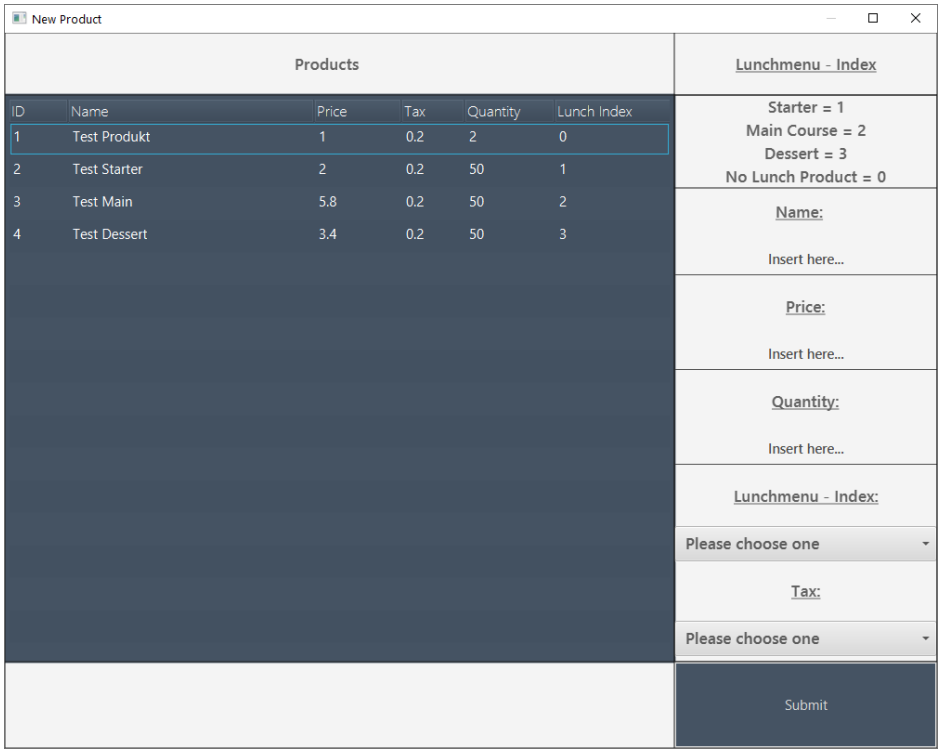

Abbildung 24: Neues Produkt anlegen

Sofern beim anlegen eines Produkts ein Fehler unterlaufen ist, so kann dieser nachträglich ausgebessert werden. Dazu muss der Menüpunkt, dargestellt in Abbildung 25 ausgewählt werden.

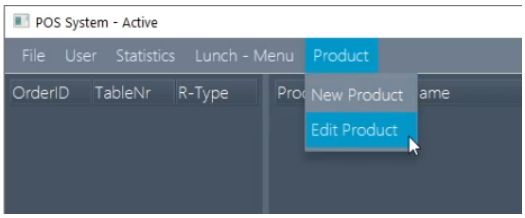

Abbildung 25: Menüpunkt: Produkt bearbeiten

Danach erscheint eine Eingabemaske, Abbildung 26 unterhalb, welches ermöglicht Produkte entweder bearbeiten oder gänzlich löschen zu können. Für den jeweiligen Schritt muss ein Produkt mittels Mausklick ausgewählt werden. Ist ein Produkt ausgewählt, die Reihe des ausgewählten Produktes wird blau hinterlegt, so kann nun gewählt werden, ob dieses Produkt gelöscht

werden soll „Delete Product” (zu Deutsch Produkt löschen), oder ob es lediglich bearbeitet werden soll „Edit Product“ (zu Deutsch Produkt bearbeiten). Sofern das Produkt gelöscht wird, verschwindet es aus der tabellarischen Auflistung, sofern das Produkt nur bearbeitet werden soll, sprich der „Edit Product“- Button ausgewählt wurde, erscheinen die Daten des Produkts in den jeweils dafür vorgesehenen Textfeldern und stehen damit zur Bearbeitung bereit. Der Umstand der Produktbearbeitung ist in Abbildung 27 auf der Seite 48 ersichtlich.

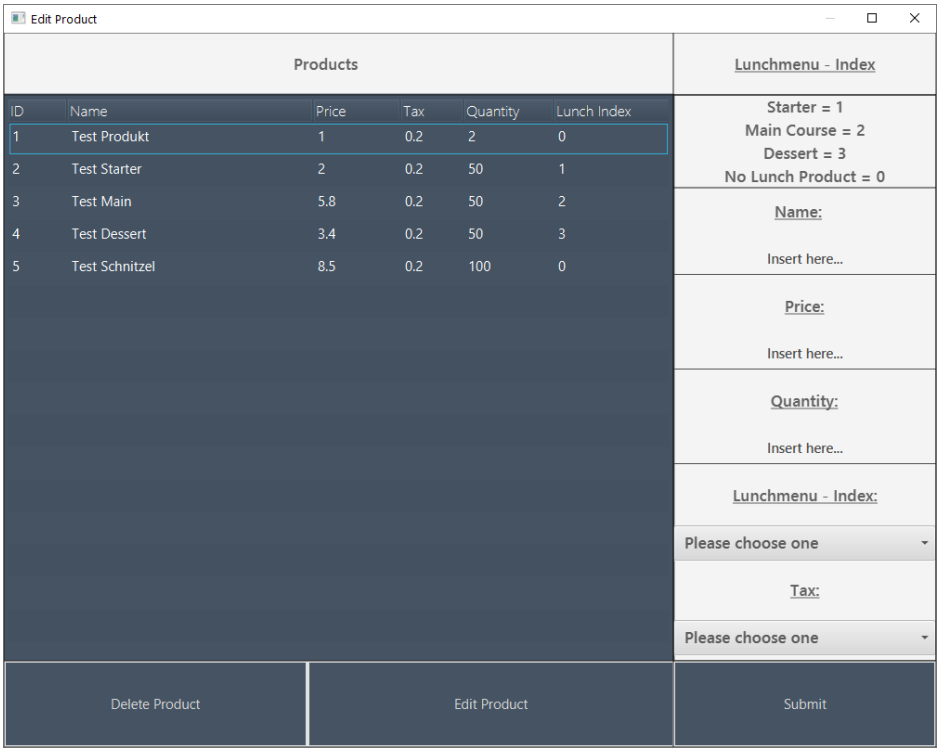

Abbildung 26: Produkt bearbeiten

Abbildung 26 oberhalb zeigt, wie das GUI direkt nach dem Aufruf aussieht. Sofern nun ein Produkt bearbeitet werden soll, muss es zuerst geladen werden, wie oberhalb bereits erläutert wurde. Nachdem ein Produkt erfolgreich geladen wurde, verändert sich das GUI wie in Abbildung 27 unterhalb auf der Seite 48 ersichtlich ist. In der gezeigten Abbildung steht das Produkt „Test Schnitzel“ zur Bearbeitung bereit.

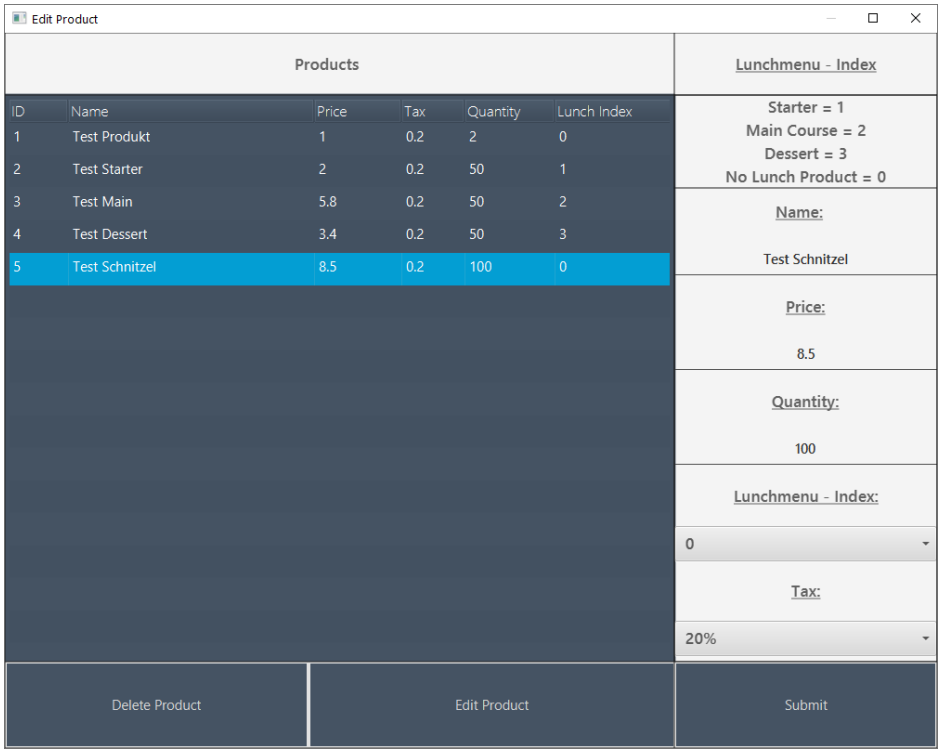

Abbildung 27: Produkt ausgewählt und zur Bearbeitung bereit

Die technische Umsetzung ist im Anhang im Kapitel Produktverwaltung auf der Seite 103 dargestellt.

3.2.4 Bestellungen verwalten

An diesem Punkt ist es erwähnenswert, dass alle notwendigen Schritte für eine erfolgreiche Bestellaufnahme getätigt wurden. Um nun eine Bestellung aufgeben zu können, muss im Hauptfenster „New Order“ (zu Deutsch Neue Bestellung), ersichtlich in Abbildung 28 unterhalb, ausgewählt werden.

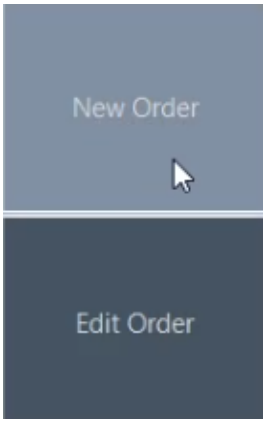

Abbildung 28: Neue Bestellung

Abbildung 29 auf der Seite 49 stellt das Bestellfenster, innerhalb dessen Produkte ausgewählt werden können, dar.

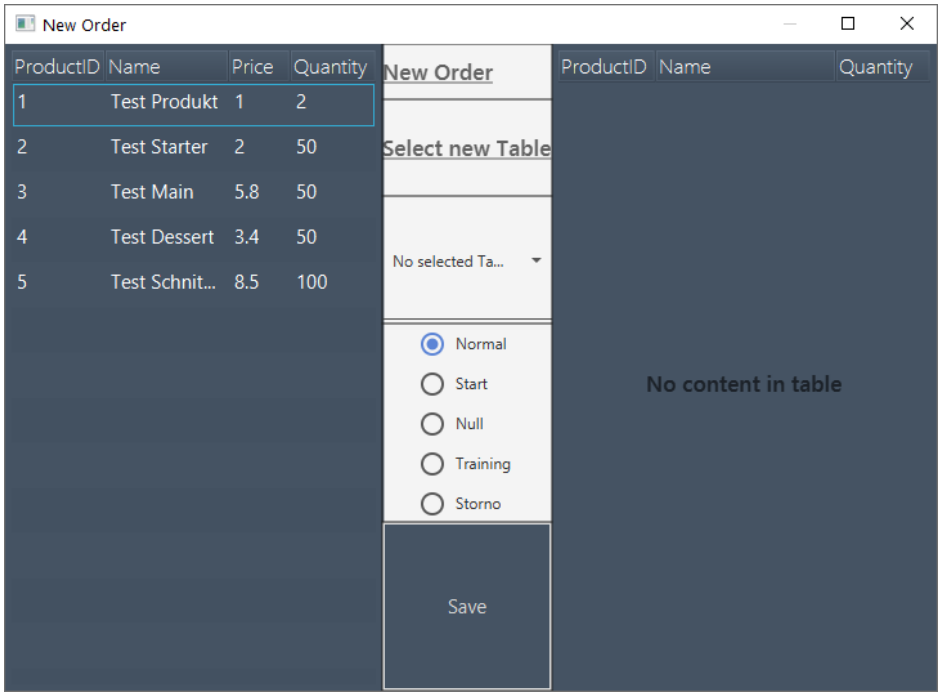

Abbildung 29: Neue Bestellung aufgeben

Die Produkte in Abbildung 29 können, mittels einfachem Klick, der Bestelltabelle, rechte Tabelle, hinzugefügt werden. Sofern das Produkt fälsch-

licherweise hinzugefügt wurde, oder ein Produkt zu oft hinzugefügt wurde, kann durch erneutes Klicken auf das Produkt, innerhalb der rechten Tabelle, einerseits die Menge reduziert werden, andererseits das Produkt ganz aus der Bestellung entfernt werden. Des Weiteren muss jeder Bestellung zwingend ein Tisch zugeordnet werden, andernfalls erscheint eine Fehlermeldung und die Bestellung kann nicht aufgegeben werden. Zu guter Letzt muss noch die Art des Belegs gewählt werden. Genauere Information bezüglich der Belegarten stehen im Kapitel Belegarten auf der Seite 20. Für eine gewöhnliche Bestellung eines Kunden, ist die Kategorie „Normal“ vordefiniert. Sofern dies nicht erwünscht ist, kann durch die Auswahl eines anderer Unterpunkts in der Mitte des Fensters die Belegart geändert werden. Das System ist so konzipiert, dass immer nur zwingend eine Art ausgewählt werden kann. Sprich, sobald eine andere Belegart ausgewählt ist, verschwindet der Punkt bei „Normal“ und wird stattdessen bei der jeweiligen Auswahl angezeigt. Ist die Bestellung nach den Wünschen des Benutzers, kann diese durch „Save“ (zu Deutsch Speichern) im System gespeichert werden. Danach wird die Bestellung im Hauptfenster angezeigt und wartet auf weitere Instruktionen, siehe Abbildung 30.

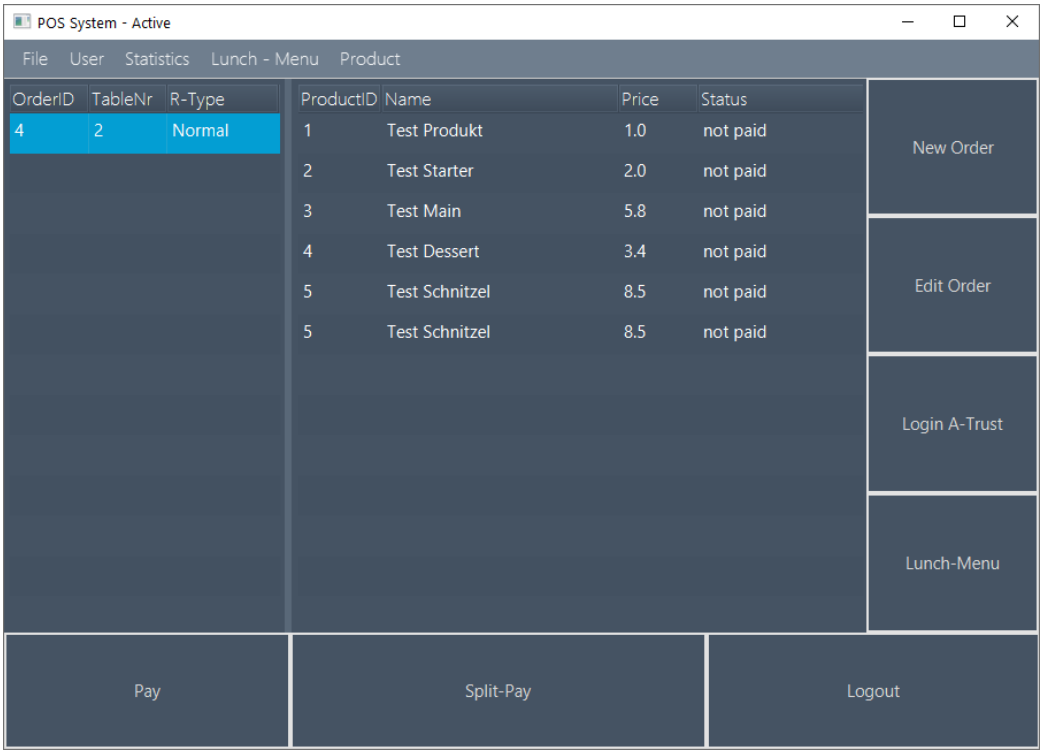

Abbildung 30: Hauptfenster mit Bestellung

In dieser Abbildung sieht man, dass die Bestellung Produkt für Produkt dargestellt wird. Sobald ein Produkt zuvor mehrmals ausgewählt wurde, wird es im Hauptmenü mehrfach dargestellt, Dies ist deshalb der Fall, weil die Möglichkeit besteht, dass eine Bestellung nicht zur Gänze gezahlt werden muss. Es besteht die Möglichkeit einzelne Produkte aus einer Bestellung separat zahlen zu können. Dazu im Kapitel Bezahlprozess auf der Seite 52 mehr. Bevor der Bezahlvorgang genauer erläutert wird, kann es sein, dass eine Bestellung fehlerhaft ist. Ist dies der Fall, so kann die Bestellung vor der Bezahlung geändert werden. Dies erfolgt mittels Knopfdruck des Buttons „Edit Order“, was zu Deutsch „Bestellung bearbeiten“ bedeutet, siehe Abbildung 31 unterhalb.

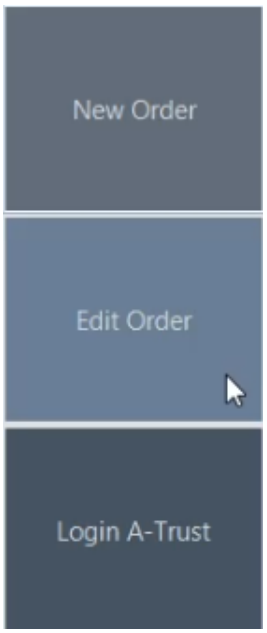

Abbildung 31: Bestellung bearbeiten - Button

Sofern eine Bestellung geändert werden muss, erscheint nach dem Auswählen des Buttons in Abbildung 31 auf der Seite 51, das Bestellfenster abermals, nur diesmal bereits mit vorhandenen Informationen über die Bestellung. Es werden die Produkte, sowie die Bestell- und Tischnummer angegeben, siehe Abbildung 32 auf der Seite 52.

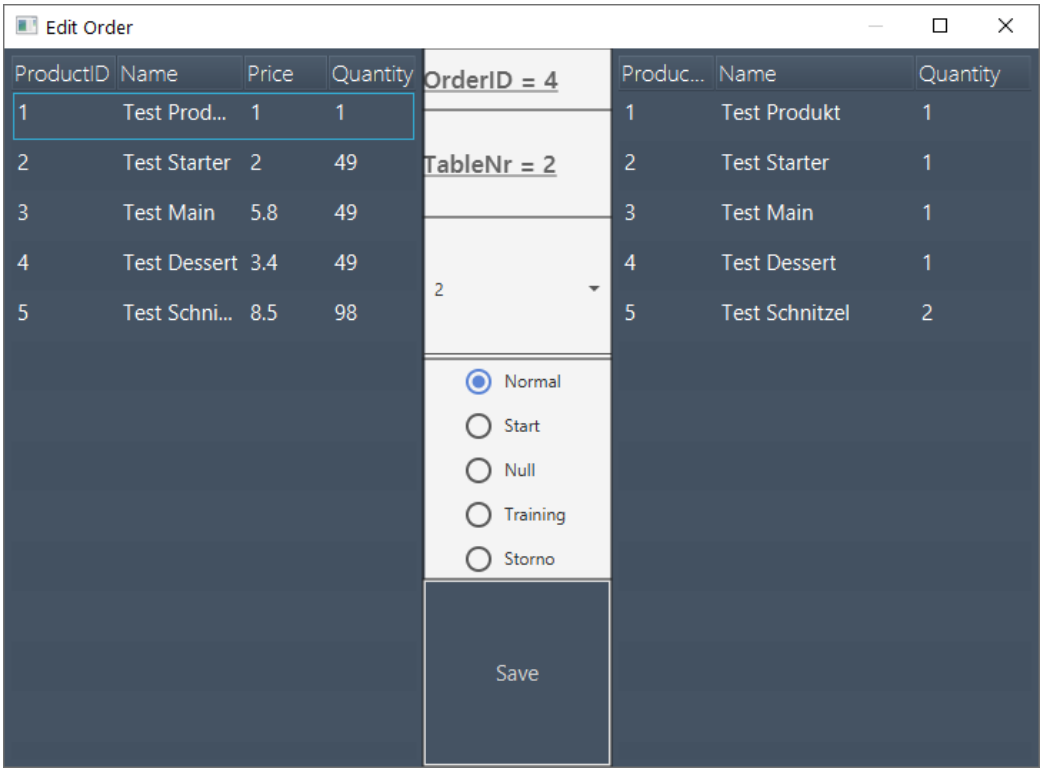

Abbildung 32: Bestellung bearbeiten

Die technische Umsetzung ist im Anhang im Kapitel Bestellungsverwaltung auf der Seite 111 dargestellt.

3.2.5 Bezahlprozess

Sobald eine Bestellung keinerlei Änderungen benötigt und somit fehlerfrei ist, folgt der Bezahlprozess. Die aufgegebene Bestellung kann nun einzeln, als Gesamtrechnung beglichen werden, oder in mehrere Teilbelege unterteilt werden. Dies ist zum Beispiel der Fall, wenn eine Gruppe von Personen eine Bestellung aufgibt, aber die Gesamtrechnung getrennt gezahlt werden soll. Dazu müssen nur Produkte in der rechten Tabelle angeklickt werden. Die ausgewählten Produkte ändern ihren Status von „not paid“ (zu Deutsch nicht bezahlt) auf „pending“, was soviel bedeutet wie „ausstehend“. Jedes Produkt, welches mit dem Status „pending“ versehen wurde, wird Teil der Rechnung, die überbleibenden Produkte bleiben in der Bestellung bestehen. Diese können wiederum als Gesamtes gezahlt werden, oder wieder in mehrere Rechnungsbelege unterteilt werden.

Für den Umstand, dass einzelne Produkte einer Bestellung separat bezahlt werden möchten, gibt es einen eigenen Button innerhalb des Hauptmenüs, welcher in Abbildung 33 unterhalb abgebildet ist.

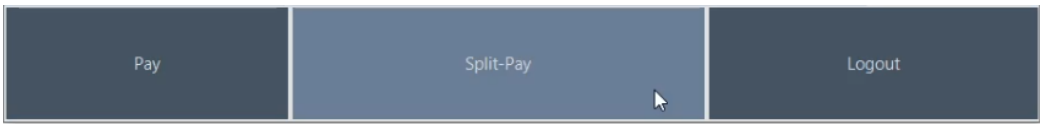

Abbildung 33: Getrennte Bezahlung

Sofern jedoch die gesamte Bestellung oder die verbleibenden Produkte als Gesamtes bezahlt werden möchten, so muss der Button, abgebildet in Abbildung 34, ausgewählt werden.

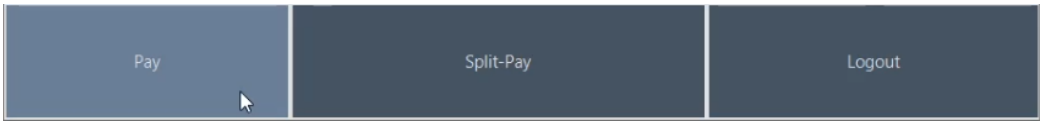

Abbildung 34: Bezahlung der gesamten Bestellung oder deren Restbestand

Die technische Umsetzung ist im Anhang im Kapitel Hauptprogramm auf der Seite 87 dargestellt. Genauer gesagt in Auflistung collection.rexx.

3.2.6 Signierungsart wählen

Als letzter Schritt bevor die jeweilige Bestellung tatsächlich bezahlt und der dazugehörige Beleg ausgedruckt werden kann, muss noch entschieden werden wie der Beleg systemintern signiert werden soll. Dazu stehen, wie eingangs im Kapitel Zertifizierungspflicht auf der Seite 16 erläutert wurde, zweierlei Arten zur Verfügung. Zum einen ist es möglich ohne Internetverbindung, mittels externen Kartenlesegeräts, einen Beleg signieren zu können oder mithilfe des Internets, ohne entsprechendem Lesegerät. Das Registierkassensystem wurde so konzipiert, dass sowohl die „Offline"- Variante als auch die „Online"-Variante eingebunden wurde, jedoch muss erwähnt werden, dass die „Offline"-Variante nur dann einsetzbar ist, wenn ein gültiges Zertifikat eingebunden wird. Da im Rahmen dieser Arbeit und dem Erstellen des POS-System keinerlei Möglichkeit bestand, etwaige Kartenlesegeräte oder dementsprechende Dokumentationen, von den im Kapitel Potentielle Vertrauensdiensteanbieter auf der Seite 15 angesprochenen Vertrauensdiensteanbieter zu bekommen, wurde diese Variante soweit eingebunden, dass sie mit einem bestehenden

Zertifikat für Testzwecke funktioniert. Für den realen Einsatz in der Praxis muss dieses, durch ein entsprechendes Lesegerät, welches mit entsprechender Karte und Zertifikat geliefert wird, ersetzt werden. Das Zertifikat, sowie entsprechende Kartenlesegerät inklusive dazugehöriger Karte, kann bei einem der im Kapitel Potentielle Vertrauensdiensteanbieter auf der Seite 15 aufgelisteten Anbieter erworben werden.

Zur Variante „Online"-Signierung, die Belege über das Internet signieren lassen kann, ist zu erwähnen, dass diese Variante zum Zeitpunkt des Erstellen und des Verfassen dieser Arbeit zur Verfügung stand. Im Laufe der Zeit wurde diese Sparte bei dem Vertrauensdiensteanbieter „A-Trust" jedoch ausgegliedert und steht somit nur noch Partnern dieses Unternehmens zur Verfügung. Diese benötigen einen eigenen Benutzernamen und ein eigenes Passwort, um auf die Infrastruktur von A-Trust zugreifen zu können. A-Trust bietet für Entwickler, solch einer POS-System-Lösung, einen Testzugang an, um diesen Zugang und die Anbindung testen zu können. Alle möglichen Varianten werden nun kurz vorgestellt und erläutert.

Um entsprechende Varianten auswählen zu können, muss zunächst der Button, abgebildet in Abbildung 35 unterhalb, angeklickt werden. Des Weiteren überprüft das System intern, ob eine Variante ausgewählt wurde. Sofern der Bezahlprozess gestartet wird, ohne dass zuvor eine Auswahl der Signierung getroffen wurde, so erscheint eine Fehlermeldung, die darauf hinweist, dass eine Auswahl noch ausständig ist. Sofern die Online-Variante, entweder Live oder Test, ausgewählt wurde, speichert das System den Loginzeitpunkt ab und setzt einen neuerlichen Logintimer auf 60 Minuten. Dieser Timer wird immer wieder zurückgesetzt, sofern ein Beleg innerhalb des Zeitfensters signiert wurde. Sofern der 60-minütige Timer abgelaufen ist, sprich 60 Minuten sind zwischen einer Belegerstellung und der darauffolgenden vergangen, so erfordert die systeminterne Logik eine neuerliche Anmeldung. Lediglich die Offline-Variante hat kein vorgegebenes Zeitfenster. Der angesprochene Timer und dessen Zeitfenster wird von A-Trust vorgegeben, da deren systeminternes Zeitfenster ebenso groß ist.

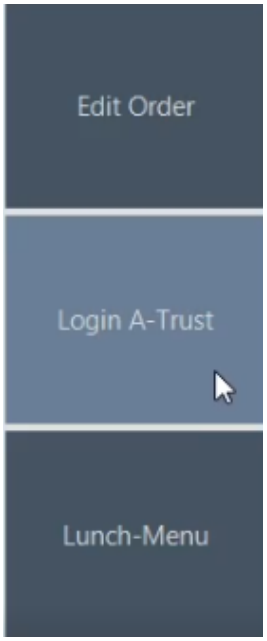

Abbildung 35: Auswahlmöglichkeit Signierung eines Belegs

Sofern der Menüpunkt, ersichtlich in Abbildung 35, betätigt wurde, erscheint ein Fenster, dargestellt in Abbildung 36 auf der Seite 56. Dieses Fenster erscheint ebenso, sobald das Zeitfenster abgelaufen ist und eine neuerliche Bezahlung ansteht.

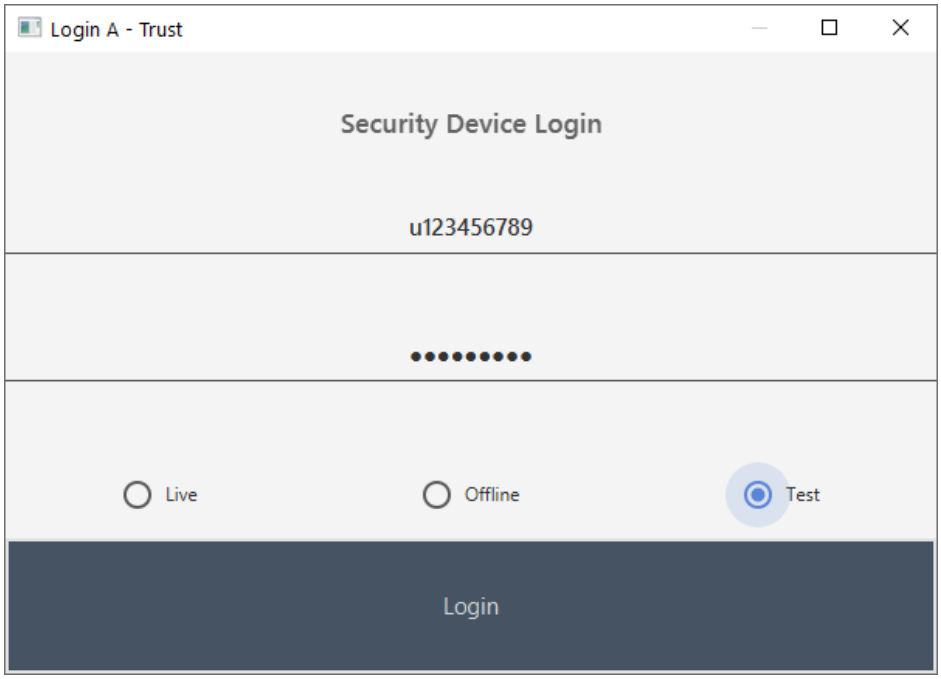

Abbildung 36: Auswahlmöglichkeit Signierung: Test

Wie leicht zu erkennen ist, ist die Variante für Testzwecke vordefiniert. Um diese Variante nutzen zu können, benötigt man einen Benutzernamen und ein Passwort. Diese sind von A-Trust vorgegeben und lauten wie folgt:

- Benutzername/Username: u123456789
- Passwort/Password: 123456789

Diese Daten müssen, wie in Abbildung 36 erkennbar, angegeben werden. Zusätzlich muss der Punkt „Test“ ausgewählt sein. Danach werden Belege, für Testzwecke über das Internet mittels der von A-Trust zur Verfügung gestellten Testinfrastruktur, signiert. Ein signierter Beleg ist in Abbildung 39 auf der Seite 59 dargestellt.

Die nächste Variante, wie ein Beleg signiert werden kann ist, wie bereits erwähnt wurde, die Variante mittels Kartenlesegerät, auch genannt die „Offline“-Variante. Dazu muss lediglich „Offline“ ausgewählt werden, ersichtlich in Abbildung 37 auf der Seite 57. Es werden weder Benutzername, noch Passwort verlangt, da diese Variante mit einem integrierten Testzertifikat arbeitet. Für diese Variante wird der, im Kapitel Bekanntgabe der Unternehmensdaten auf der Seite 31 erwähnte, JWS-Schlüssel verwendet.

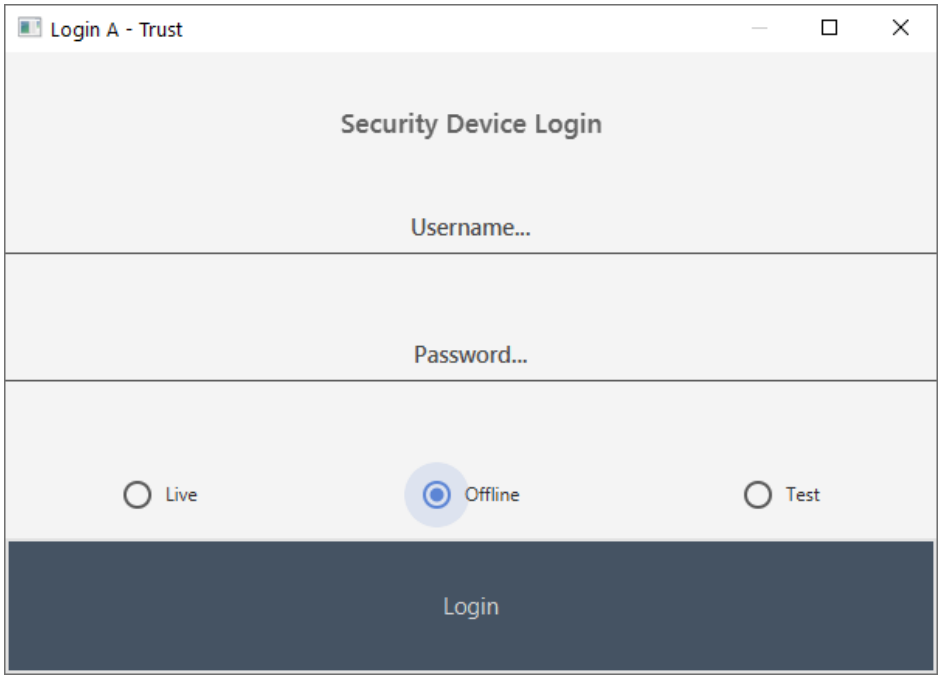

Abbildung 37: Auswahlmöglichkeit Signierung: Offline

Die letzte Variante, ist die „Live“-Signierung. Siehe Abbildung 38 auf der Seite 58. Diese Art von Signierung wird ebenso mittels der Infrastruktur von A-Trust bewerkstelligt. Diese Infrastruktur unterscheidet sich nur in ihrer systeminternen Interaktion, siehe technische Umsetzung im Anhang Kapitel Sicherheitseinrichtung auf der Seite 118. A-Trust bietet sowohl eine Test-, als auch eine offizielle Infrastruktur an. Diese Variante wird, wie erwähnt, über das Internet angesprochen, somit ist eine aufrechte Internetverbindung zwingend notwendig. Leider steht diese Variante nur noch Partnerunternehmen von A-Trust zur Verfügung. Da dies eine studentische Arbeit im Rahmen der Bachelorarbeit ist, ist dieser Zugang leider nicht mehr zugänglich, ist aber für Demonstrationswecken dennoch im System integriert.

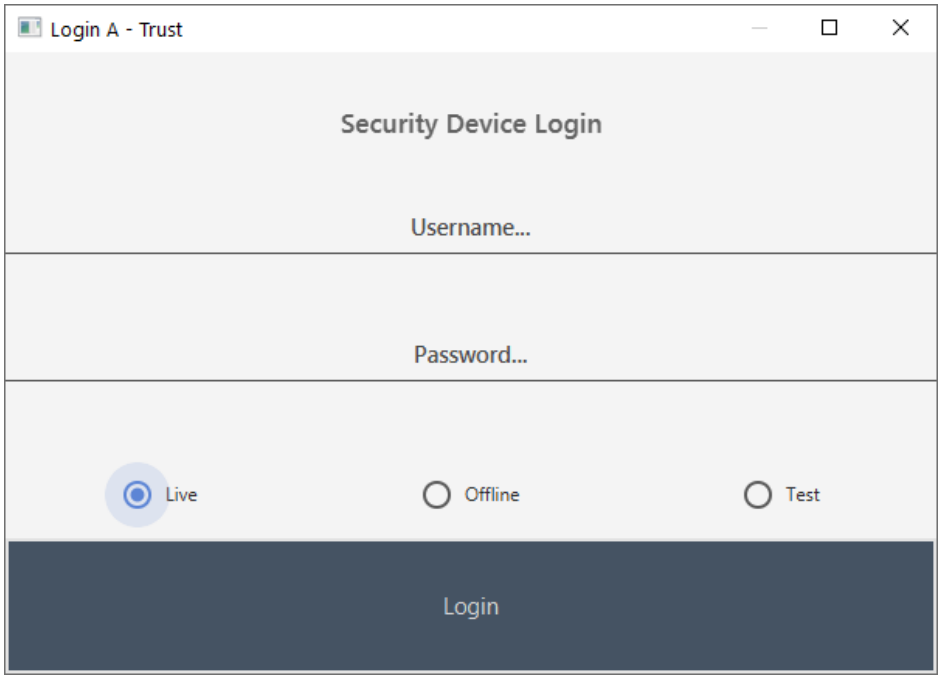

Abbildung 38: Auswahlmöglichkeit Signierung: Live

Um die Live-Variante nutzen zu können, müssen Benutzername und Passwort entsprechender Partnerunternehmen akquiriert werden. Ein Liste solcher Unternehmen stehen unter folgendem Link zur Verfügung: https://www.a-trust.at/de/registrierkasse/rksv-partner/[14]

Die technische Umsetzung ist im Anhang im Kapitel Sicherheitseinrichtung auf der Seite 118 dargestellt.

3.2.7 Resultierender Beleg

Die Bestellung, welche im Kapitel Bestellungen verwalten auf der Seite 48 begonnen wurde, ist nun bezahlt, signiert und ausgedruckt. Auf der Seite 59 in Abbildung 39 ist der resultierende Beleg dargestellt. Sofern der QR-Code gescannt wird, wird ersichtlich, dass alle relevanten Daten aus dem Kapitel QR-Code auf der Seite 26 integriert wurden und somit die RKSV zur Gänze erfüllt wird. Jedoch, wie bereits erwähnt, mittels Testzertifikat. Für den Einsatz in der Praxis, muss dieses ersetzt werden.

Fake GmbH

fakestreet 123
0650 123456789
company@chello.at
AT123456789
OrderID: 4
TableNr.: 2
2021-06-15
10:55:41

Item	Qty	Price	Total
Test Produkt	1	€ 1.0	€ 1.0
Test Starter	1	€ 2.0	€ 2.0
Test Main	1	€ 5.8	€ 5.8
Test Dessert	1	€ 3.4	€ 3.4
Test Schnitzel	2	€ 8.5	€ 17.0

Total amount: € 29.2

MWST	Net	Tax	Gross
20%	24.33	4.87	29.20

Manuel served you!
Thank you for your purchase.
We look forward to seeing you again soon.

Abbildung 39: Standardbeleg einer Bestellung

Abbildung 39 demonstriert, dass alle wichtigen Daten, aus Kapitel Standardbeleg auf der Seite 22, enthalten sind. Der Name und die Anschrift des belegerstellenden Unternehmens sind abgebildet. Weiters ist ersichtlich, dass sowohl die Telefonnummer, als auch die E-Mail-Adresse des Unternehmens dargestellt werden. Zudem wird die Umsatzsteuernummer, sowie die Rechnungsnummer ausgewiesen. Zusätzlich wird die Tischnummer, das Datum und die Uhrzeit zum Zeitpunkt der Belegerstellung ausgewiesen.

3.3 Auszubildender

In diesem Unterkapitel werden all jene Funktionen vorgestellt, die unabhängig von einer speziellen Benutzerrolle, von jeden verwendet werden können. In einem kleinen gastwirtschaftlichen Betriebe ist es oftmals der Fall, dass Personen in Ausbildung am Geschäftsalltag teilhaben und integriert werden. Für diesen Zweck wurde eine eigene Benutzerrolle geschaffen. Sofern ein Benutzer die Rolle eines Auszubildenden inne hat, beschränkt sich dessen Zugang auf das Wesentlichste.

Dies wäre zum einen die Verwaltung des Mittagsmenüs, wie im Kapitel Mittagsmenü verwalten auf der Seite 43 beschrieben wurde. Zum anderen kann ein Auszubildender Bestellungen aufgeben und ändern, jedoch ist solch ein Benutzer nicht berechtigt den Bezahlvorgang einzuleiten. Für diesen Zweck muss ein Angestellter mit entsprechend höherer Berechtigung eingeloggt sein.

3.3.1 Logout

Wie im Kapitel zuvor erwähnt, können unterschiedlich berechtigte Personen auf ein und dem selben System arbeiten. Aus diesem Grund wurde die Möglichkeit geschaffen, sich aus dem laufendem System ausloggen zu können. Das ist dahingehend vorteilhaft, da Benutzer mitunter nicht die gesamte Arbeitszeit vor dem Systembildschirm verbringen, sondern gegebenenfalls diesen verlassen müssen. So könnte, ohne Logout-Funktion, eine unbefugte Person auf das ungesicherte System zugreifen. Sofern sich jedoch ein Benutzer erfolgreich ausgeloggt hat, ist der erneute Zugang zum System nur durch korrekte Angabe von Benutzernamen und Passwort möglich. Mittels Knopfdruck, ersichtlich in Abbildung 40, wird der jeweilige Benutzer aus dem System ausgeloggt. Das Hauptfenster verschwindet und wird durch das Fenster, dargestellt in Abbildung 5 auf der Seite 30, ersetzt.

Abbildung 40: Logout

Nachdem „Get ready to work...“ (zu Deutsch Bereit für die Arbeit...) in Abbildung 5 gedrückt wurde, erscheint neuerlich die Login-Maske, dargestellt in Abbildung 8 auf der Seite 34. Mithilfe dessen kann sich ein Benutzer neu anmelden.

Die technische Umsetzung ist im Anhang im Kapitel Hauptprogramm auf der Seite 87 dargestellt. Genauer gesagt in Auflistung collection.rexx.

3.3.2 Programm beenden

Zu guter Letzt ist es für jedem Benutzer möglich, das Registierkassensystem zu beenden. Siehe Button in Abbildung 41 unterhalb.

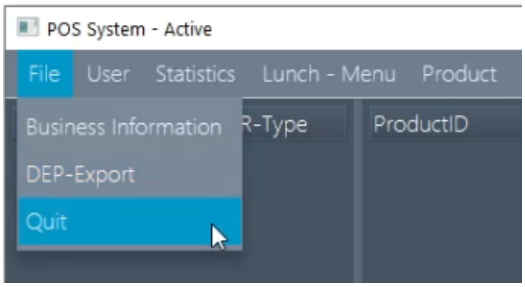

Abbildung 41: System beenden

Nach erfolgreicher Beendigung verbraucht das System keinerlei Ressourcen, wie zum Beispiel Internet oder Strom, sowie generelle Rechnerkapazitäten. Sofern das Programm neu gestartet wird, erscheint wiederum das Startfenster, abgebildet in Abbildung 5 auf der Seite 30, jedoch ohne neuerliches Angeben eines Administrators oder dergleichen. Dies wurde bei erstmaligem Starten des Systems erledigt. Es folgt sofort die Loginmaske, aus dem Kapitel Login auf der Seite 33.

Die technische Umsetzung ist im Anhang im Kapitel Hauptprogramm auf der Seite 87 dargestellt. Genauer gesagt in Auflistung collection.rexx.

3.4 Herausforderung beim Drucken eines Belegs

Die Registrierkassensicherheitsverordnung bestimmt nicht nur Aspekte einer Registrierkasse, sondern gibt vor, dass ein Beleg entweder in gedruckter oder elektronischer Form dem Kunden übergeben werden muss. Das POS-System ist so konzipiert, dass Belege auf einem handelsüblichen Drucker, siehe Abbildung 42 unterhalb, gedruckt werden können. Der Beleg hat eine Größe, welches einem A4-Blatt gleichkommt.

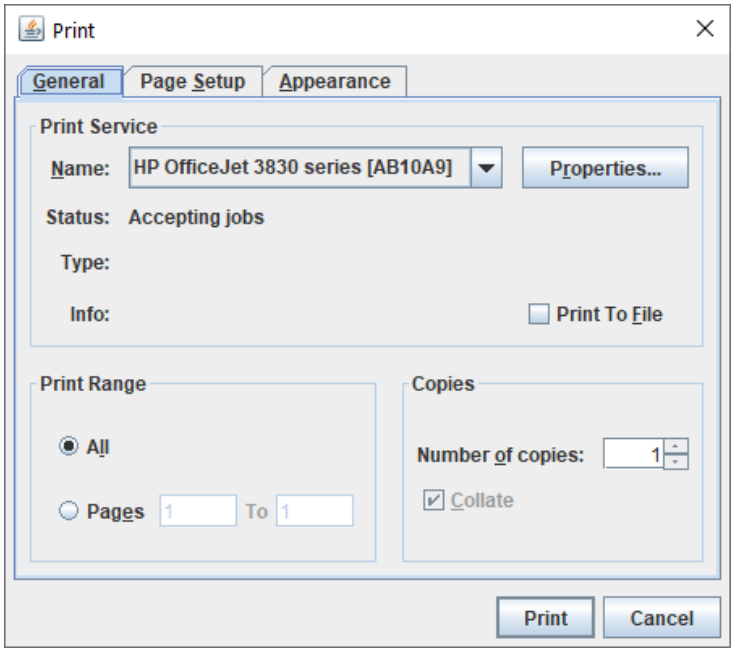

Abbildung 42: Druckereigenschaften

Sofern eigens dafür entwickelte Belegdrucker verwendet werden, muss bedacht werden, dass der zugehörige Programmcode, siehe Anhang im Kapitel Hauptprogramm auf der Seite 87 Auflistung collection.rexx, adaptiert werden muss. Weiters kann es auch notwendig sein, dass etwaige Treiber eines Belegdruckers neuerlich installiert werden müssen.

4 Zusammenfassung und Ausblick

Das zugrunde gelegte System, welches in dieser Arbeit erläutert wurde, erfüllt den Großteil der vorgeschriebenen technischen Voraussetzungen der Registrierkassensicherheitsverordnung und enthält zudem weitere Funktionen,

die den Benutzer, in der Ausübung seiner Arbeit, unterstützen sollen. Leider war es in der vorgegebenen Zeit jedoch nicht möglich alle vorgeschriebene Rahmenbedingungen vollständig zu erfüllen, da entweder wenig bis gar keine Dokumentation oder Hilfestellung, seitens der Vertrauensdiensteanbieter und des Bundesministerium für Finanzen, bereitgestellt wurde. So war es vermehrt unmöglich herauszufinden, wie Sicherheitseinrichtung mit Registrierkasse interagieren, um diese in das System einbinden zu können. Aus diesem Grund wurde nur ein Anbieter von Signatur- und Siegelerstellungseinheiten eingebunden. Dieser Anbieter ist A-Trust, welcher als Einziger entsprechende Dokumentationen bereit gestellt hat. Es wurden drei unterschiedliche Varianten für eine Signaturerstellung eingebunden, jedoch mit Testdaten, welche in der Praxis durch reale, von einem Vertrauensdiensteanbieter bereitgestellte, Daten ersetzt werden müssen, damit das System in der Praxis eingesetzt werden kann. Alle weiteren technischen Voraussetzungen für ein elektronisches Kassensystem wurden zur Gänze umgesetzt. Somit bietet es sich für weitere Entwicklungen des Systems an, dass mitunter weitere Anbieter eingebunden werden, sodass die Registrierkasse universeller einsetzbar wird.

Weiters muss eine Registrierkasse, vom Unternehmen welches diese in Verwendung hat, dem Bundesministerium für Finanzen gemeldet werden. Die Meldung kann auf unterschiedliche Art und Weise vollzogen werden. Zum einen ist es möglich, über das Internet, eine direkte Anbindung an das Bundesministerium für Finanzen zu schaffen, jedoch muss das entsprechende elektronische Kassensystem dafür konzipiert sein. Die entsprechende bereitgestellte technische Dokumentation über die Anbindung zum BMF ist ebenfalls sehr spärlich, weshalb die Meldung des Systems momentan mittels Dialogverfahren vollzogen wird, was im Grunde durch Ausfüllen eines Formulars geschieht. Für weitere Entwicklungen des zugrunde gelegten Registrierkassensystems könnte der Ansatz der automatischen Meldung über das Internet und entsprechende Schnittstellen forciert werden.

Ein generellere Erweiterungsmöglichkeit wäre zudem, dass das elektronische Kassensystem mit unterschiedlichen, im Betrieb eingesetzten, Technologien vernetzt wird, sodass weitere automatische Datenverarbeitungen vonstatten gehen. So bietet sich die Möglichkeit an, dass zum Beispiel ein Warenwirtschaftssystem, was mit der Verwaltung von Waren beauftragt ist, eingebunden wird, sodass zum Beispiel Lagerbestände automatisch aktualisiert werden, sobald eine neue Lieferung antrifft. Dies wird in der aktuellen Version noch händisch adaptiert.

Anhang

Quellenverzeichnis

[1] Wirtschaftskammer Österreich - Registrierkassenpflicht. https://www.wko.at/service/steuern/registrierkassenpflicht-unternehmen.html#:~:text=Betriebe%20sind%20zur%20Verwendung%20einer,zus%C3%A4tzlich%20%C3%BCber%20einen%20Manipulationsschutz%20verf%C3%BCgen. Aufruf: 10-06-2021.

[2] Ausnahmeregelungen für den Einsatz einer Registrierkasse. https://www.wko.at/service/steuern/Registrierkassen--und-Belegerteilungspflicht.html. Aufruf: 10-06-2021.

[3] Registrierkassensicherheitsverordnung. https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20009390. Aufruf: 10-09-2020.

[4] Aufbewahrungspflicht für Unternehmen. https://www.wko.at/service/steuern/Aufbewahrungspflichten.html#:~:text=Grunds%C3%A4tzlich%20sind%20B%C3%BCcher%20und%20Aufzeichnungen,Original%20sieben%20Jahre%20hindurch%20aufzubewahren. Aufruf: 10-06-2021.

[5] Rundfunk und Telekom Regulierungs-GmbH. https://www.rtr.at/TKP/was_wir_tun/vertrauensdienste/anbieter/liste_der_vertrauensdiensteanbieter/Anbieter.de.html. Aufruf: 10-06-2021.

[6] A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH. https://www.a-trust.at/. Aufruf: 10-06-2021.

[7] e-commerce monitoring GmbH. http://www.e-monitoring.at/. Aufruf: 10-06-2021.

[8] PrimeSign GmbH. https://www.prime-sign.com/. Aufruf: 10-06-2021.

[9] Henry S. Thompson and Chris Lilley. XML Media Types. RFC 7303, July 2014. https://datatracker.ietf.org/doc/html/rfc7303. Aufruf: 10-06-2021.

[10] Bundesministerium für Finanzen - Handbuch Registrierkassen. https://www.bmf.gv.at/dam/jcr:0af97a40-da60-4c81-8e1e-22c3ecca52a4/BMF_Handbuch_Registrierkassen.pdf. Aufruf: 10-06-2021.

[11] Unternehmensserviceportal - Kassenbelege. https://www.usp.gv.at/steuern-finanzen/registrierkassen/kassenbelege.html. Aufruf: 10-06-2021.

[12] Funktionsweise eines QR-Codes. https://www.qrcode.com/en/about/. Aufruf: 10-06-2021.

[13] Russ Housley and Jim Schaad. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394, October 2002. https://datatracker.ietf.org/doc/html/rfc3394. Aufruf: 10-06-2021.

[14] Liste von A-Trust-Partnern. https://www.a-trust.at/de/registrierkasse/rksv-partner/. Aufruf: 10-06-2021.

[15] Freie Version Java: Azul JDK-FX. https://www.azul.com/downloads/?version=java-8-lts&package=jdk-fx. Aufruf: 24-06-2021.

[16] Freie Version Java: Liberica Full JDK. https://bell-sw.com/pages/downloads/#/java-8-lts. Aufruf: 24-06-2021.

[17] ooRexx - Versionen. https://sourceforge.net/projects/oorexx/files/oorexx/. Aufruf: 10-06-2021.

[18] BSF4ooRexx - Versionen. https://sourceforge.net/projects/bsf4oorexx/files/. Aufruf: 10-06-2021.

[19] Übersicht: SQLite-JDBC. https://github.com/xerial/sqlite-jdbc. Aufruf: 10-06-2021.

[20] Übersicht: JFoenix. https://github.com/sshahine/JFoenix. Aufruf: 10-06-2021.

[21] Übersicht: Core + Javase. https://github.com/zxing/zxing. Aufruf: 10-06-2021.

[22] Übersicht: JSON-simple. https://code.google.com/archive/p/json-simple/. Aufruf: 10-06-2021.

[23] Übersicht: Nimbus-Jose-Jwt. https://bitbucket.org/connect2id/nimbus-jose-jwt/src/master/. Aufruf: 10-06-2021.

[24] Übersicht: Spring-Security-Crypto. https://spring.io/projects/spring-security. Aufruf: 10-06-2021.

Anhang B Programmdokumentation

B.1 Softwarekomponenten

Um nun das entsprechende Registrierkassensystem einsetzen zu können, müssen einige Komponenten zunächst geladen und installiert werden. Diese Komponenten werden in jeweilige Unterkapitel gegliedert und erläutert. Alle Softwarekomponenten, die der Autor für die Realisierung der vorliegenden Aufgabenstellung erarbeitet und beigefügt hat, stehen unter der Apache Lizenz 2.0.

B.1.1 Java

Auf dem zu benutzendem Betriebssystem muss die Java-Version 8 oder höher installiert sein. Wenn dies noch nicht der Fall ist, muss dies nachgeholt werden. Zudem ist zu erwähnen, dass Oracle, der Konzern welcher Java verwaltet, seine Lizenzbedingungen dahingehend geändert hat, sodass ab Java Version 8 oder später Lizenzgebühren anfallen, sofern Java in einem kommerziellen Bereich eingesetzt wird. Damit das Kassensystem weiterhin ohne Lizenzgebühren benutzt werden kann empfiehlt es sich eine freie Version, ohne anfallende Gebühren zu verwenden. Weiters ist darauf zu achten, dass die geladene Version JavaFX enthält. JavaFX wird für die grafische Darstellung der Benutzeroberflächen benötigt und ist nicht in jeder Version enthalten. Unter folgenden Links können entsprechende freie Java Versionen geladen werden: Azul JDK-FX: https://www.azul.com/downloads/?version=java-8-lts&package=jdk-fx (Aufruf: 24-06-2021), oder Libericas JDK FULL: https://bell-sw.com/pages/downloads/#/java-8-lts (Aufruf: 24-06-2021). Bei der Version von Liberica muss zusätzlich darauf geachtet werden, dass ausschließlich die „FULL“ - Version die entsprechenden Vorgaben erfüllt.

B.1.2 ooRexx

Weiters ist es zwingend erforderlich, dass ooRexx installiert sein muss. ooRexx steht für „Open Object Rexx“ und ist die zugrunde gelegte Programmiersprache. Um nun das Registrierkassensystem entsprechend starten zu können, muss zunächst die Version 5.0 oder höher geladen werden. Die geforderte Version kann unter folgendem Link geladen werden: https://sourceforge.net/projects/oorexx/files/oorexx/ (Aufruf: 10-06-2021). Weiters muss erwähnt werden, dass ebenso auf die Bit-Architektur des Betriebssystems geachtet werden muss. Die erforderliche Version, sollte dieselbe Bit-Architektur wie das Betriebssystem aufweisen, um etwaige Programmfehler ausschließen zu können.

B.1.3 BSF4ooRexx

BSF steht für Bean Scripting Framework und ist ein Projekt der Apache Software Foundation (ASF), das ein Java-Rahmenwerk für die bidirektionale Kommunikation mit Scriptsprachen ermöglicht. BSF4ooRexx ist eine bidirektionale Brücke zwischen ooRexx und Java, mit der man aus ooRexx mit Java-Objekten und umgekehrt, aus Java mit ooRexx-Objekten interagieren kann. Damit wird es unter anderem möglich, von ooRexx aus sämtliche verfügbare Java-Klassen so zu nutzen, wie wenn sie in ooRexx implementiert wären. Eine entsprechende BSF4ooRexx-Version ist unter folgendem Link zugänglich: https://sourceforge.net/projects/bsf4oorexx/files/ (Aufruf: 10-06-2021).

B.1.4 Benötigte Java Klassenbibliotheken

Im Kapitel BSF4ooRexx wurde bereits erwähnt, dass für die sachgerechte Benutzung des Kassensystems einige Komponenten aus einer anderen Programmiersprache benötigt werden. Diese Komponenten sind sogenannte Java Archive (kurz JAR), oder auch Java Klassenbibliotheken genannt. Ein JAR ist nichts weiter als eine Sammlung an Java-Klassen und Methoden, die für einen gemeinsamen Zweck zusammengefasst wurden. Zum Beispiel können mithilfe des Systems Passwörter angelegt werden. Diese sollten im Normalfall nicht im Klartext gespeichert werden. Deshalb wurde, für die Passwortverwaltung, ein eigene Java Klassenbibliothek benutzt, welche alle relevanten Funktionen gesammelt aufweist, um ein etwaiges Passwort sicher verwalten zu können (spring-security). Unterhalb in Tabelle 2 auf der Seite 69 befinden sich alle benötigten Java Klassenbibliotheken und deren dazugehörige Website.

Name	Beschreibung	Verwendung	Link
sqlite-jdbc-3.32.3.2.jar	SQLite JDBC ist eine Bibliothek für den Zugriff und die Erstellung von SQLite-Datenbankdateien in Java [19]	Diese Bibliothek wird im Zuge der Datenspeicherung benötigt, um etwaige Daten, die im Laufe der Benutzung des Kassensystems anfallen, in der eigens dafür kreierten Datenbank zu speichern	https:// github.com/xerial/sqlite-jdbc (Aufruf: 10-06-2021)
jfoenix-9.0.10.jar	JFoenix ist eine Open-Source-Java-Bibliothek, die Google Material Designs mithilfe von Java-Komponenten implementiert [20].	JFoenix wird innerhalb der Registrierkasse für die moderne Darstellung der Fenster und deren Elemente benötigt. Mithilfe dessen ist es möglich moderne Arten von Textfeldern, Knöpfe und dergleichen darstellen zu können.	https:// github.com/sshahine/JFoenix (Aufruf: 10-06-2021)

core-3.4.1.jar **javase-3.4.1.jar**	Core und Javase sind beides Bibliotheken, welche es in Kombination ermöglichen QR-Codes einfach generieren zu können [21].	Mithilfe beider Bibliotheken werden etwaige QR-Codes für die Belegerstellung generiert. Die erzeugten QR-Codes werden systemintern auf dem entsprechenden Beleg abgebildet.	https:// github.com/ zxing/zxing (Aufruf: 10-06-2021)
json-simple-1.1.1.jar	JSON simple ist ein einfaches Java-Toolkit für JSON, mithilfe dessen man JSON-Text codieren oder decodieren kann [22].	Mit dieser Bibliothek ist es möglich, einfach und simpel einen gewünschten Text in JSON-Notation zu erstellen. Dazu werden etwaige Daten der Registrierkasse gesammelt und als JSON-Format ausgegeben. Genauer gesagt wird in diesem Format das Datenerfassungsprotokoll (DEP) exportiert.	https:// code.google.com/ archive/p/json-simple/ (Aufruf: 10-06-2021)
nimbus-jose-jwt-9.8.1.jar	Dies ist eine Java-Bibliothek, welche alle Standard-Signaturalgorithmen versteht (JWS) [23].	Diese Bibliothek kommt während der Offline-Signierung zur Anwendung. Die RKSV schreibt bestimmte Methoden vor, wie eine Signatur abgehandelt werden muss. Dies ist eine Variante davon, welche integriert wurde.	https:// bitbucket.org/ connect2id/ nimbus-jose-jwt/src/master/ (Aufruf: 10-06-2021)
spring-security-crypto-5.4.5	Spring Security ist ein leistungsstarkes Authentifizierungs- und Zugriffskontroll-Framework [24].	Mithilfe dieses Frameworks, bzw. dieser Bibliothek werden etwaige Passwörter codiert, sodass diese nicht im Klartext gespeichert werden. Weiters kommt die Bibliothek zur Überprüfung etwaiger Passwörter, während des Loginprozesses, zum Einsatz.	https:// spring.io/ projects/ spring-security (Aufruf: 10-06-2021)

Tabelle 2: Java Klassenbibliotheken

Sind zunächst alle relevanten Softwarekomponenten geladen, so müssen diese sachgerecht installiert werden. Die sachgerechte Installation ist Bestandteil des nächsten Kapitels, Kapitel Installationsanleitung auf der Seite 70.

B.2 Installationsanleitung

Damit es nun möglich ist, das Registrierkassensystem konfigurieren bzw. starten zu können, müssen die oben bereits genannten Komponenten unterschiedlich behandelt werden. Zudem ist zu erwähnen, dass die folgende Installationsanleitung für das Betriebssystem Windows konzipiert wurde. Es ist jedoch möglich, dass das vorgestellte Kassensystem auch unter Linux und MacOS benutzt werden kann, wenn die entsprechenden Komponenten geladen und installiert wurden.

B.2.1 Java

Zunächst ist es zwingend erforderlich, dass eine entsprechende Java-Version vorab installiert wird. Häufig ist eine Java-Version auf einem entsprechenden Rechner vorinstalliert. Sollte dies nicht der Fall sein, oder die installierte Version entspricht nicht den Vorgaben des vorherigen Kapitels, so muss neuerlich eine entsprechende Version installiert werden. Für diesen Zwecke muss die zunächst geladene Datei (.exe), aus Kapitel Java auf der Seite 67, ausgeführt werden. Nachdem durch einen entsprechenden Doppelklick die Datei ausgeführt wurde, so erscheint ein Installationsmenü. Nun sollten die Instruktionen, welche im Laufe des Installationsprozesses auftreten, ausgeführt werden, sodass am Ende eine Version von Java installiert ist, welche den Anforderungen entspricht.

B.2.2 ooRexx

Ist die geforderte Java-Version installiert, kann nun eine entsprechende Version von ooRexx installiert werden. Dazu muss ebenfalls, die zuvor geladene Datei (.exe), aus Kapitel ooRexx auf der Seite 67, ausgeführt werden. Da ooRexx eine eigenständige Sprache ist, besitzt diese auch eine Schar an Dokumentationen auf die an dieser Stelle, für nähere Informationen, verwiesen wird. Es erscheint ebenfalls nach dem Ausführen der .exe-Datei von ooRexx ein Installationsmanager, dem Folge geleistet werden sollte. Nach Ausführung der unterschiedlichen Instruktionen ist der entsprechende Rechner nun in der Lage sowohl Programme in Rexx verfassen, als auch verstehen zu können. Dies ist essenziell, um die Registrierkasse, welche in Rexx verfasst wurde, starten und benutzen zu können.

B.2.3 BSF4ooRexx

Ein weiterer wichtiger Schritt in Richtung Registrierkassensystem ist, dass das in Kapitel BSF4ooRexx auf der Seite 68 erwähnte, Bean Scripting Fra-

mework for ooRexx installiert wird. Wie bereits erwähnt, ist ooRexx in Kombination mit BSF4ooRexx in der Lage Elemente anderweitiger Programmiersprachen ausführen zu können. In diesem Fall werden Java-Elemente eingegliedert, diese verhalten sich jedoch wie ooRexx-Elemente. In ihrer Ausführung bemerkt man keinen Unterschied, ob eine entsprechende Funktion in Java oder ooRexx geschrieben wurde, so ist es möglich die Vorteile beider Programmiersprachen zu vereinen. Um BSF4ooRexx installieren zu können, muss zunächst der geladene Ordner (ZIP-Archiv) an einer beliebigen Stelle innerhalb des Betriebssystems entpackt werden und dafür Sorge getragen werden, dass bereits Java und ooRexx installiert wurden. Nachdem dies erfolgt ist, erscheint unteranderem ein Unterverzeichnis „install“. Dieses Verzeichnis beinhaltet jeweils eine Installationsdatei aller relevanten Betriebssystemen (Windows, Mac, Linux). Hierbei muss darauf geachtet werden, dass die richtige, dem Betriebssystem entsprechenden, Version ausgeführt wird. Für weitere Informationen wird auf die BSF eigene Dokumentation verwiesen, diese kann im entpackten Unterverzeichnis „information“ und in der mitgelieferten Readme-Datei eingesehen werden.

B.2.4 CLASSPATH

Sind die zuvor erwähnten Installationen erfolgreich abgeschlossen, sowie alle genannten Java Klassenbibliotheken geladen, so ist ooRexx nun in der Lage Java-Elemente wie Bibliotheken, Methoden und dergleichen verwenden zu können. Besagte Elemente sind in den jeweiligen, in Tabelle 2 auf der Seite 69 gelisteten, Java-Archiven gespeichert. Alle dort gelisteten Java-Archive müssen in den CLASSPATH des Betriebssystems integriert werden. Unter Windows besteht die Möglichkeit, dass alle Archive gesammelt in einem Ordner zusammengefasst werden und der gesamte Ordner Teil des CLASSPATHs wird. Für Windows erfolgt nun eine grobe Anleitung wie dies vonstattengeht.

1. (⊞) -Taste drücken und „Systemumgebungsvariablen bearbeiten“ tippen
2. Es erscheint ein Fenster, innerhalb dessen „Umgebungsvariablen“ angeklickt werden muss
3. Es erscheint wiederum ein Fenster „Umgebungsvariablen“
4. Unter Punkt „Systemvariable“ ist nun die Variable CLASSPATH ersichtlich
5. Mittels CLASSPATH editieren gelangt man zur Auflistung aller bereits angelegten Pfade

6. Mittels „Neu“ muss nun ein neuer Pfad zu den Java-Archiven angelegt werden (Entweder Archiv für Archiv, oder den gesamten Ordner als Pfad angeben, ein entsprechendes Beispiel ist unterhalb angeführt)

Beispielpfad-JAR:
C:\Users\manue\Documents\Bachelorarbeit\RegKassa\JAR\spring-security-crypto-5.4.5.jar

Beispielpfad Gesamtordner:
C:\Users\manue\Documents\Bachelorarbeit\RegKassa\JAR*

Nun kann die im nächsten Kapitel anberaumte Konfiguration des Registrierkassensystems selbst vonstattengehen.

B.3 Konfiguration

In diesem Kapitel befinden sich alle relevanten Source-Code-Dateien bezüglich der Konfiguration des Registrierkassensystems. Sofern das System zum ersten Mal gestartet werden soll, müssen zuerst einige Dinge vorkonfiguriert und überprüft werden. Für diesen Fall muss der Anwender die Datei setupComplete.rexx auf Seite 73 ausführen. Diese Datei überprüft ob die richtige Java-Version auf dem Betriebssystem installiert wurde. Des Weiteren, wird eine Datenbank erstellt und konfiguriert, sodass ein sofortiger Start des Systems erfolgen kann. Abbildung 43 zeigt den Zusammenhang zwischen den beiden Dateien, die am Ende das System so weit konfiguriert haben, dass das Registrierkassensystem zum ersten Mal gestartet werden kann. Abbildung 44 auf der Seite 75 zeigt den Zusammenhang unterschiedlicher Dateien im Rahmen des Hauptprogramms.

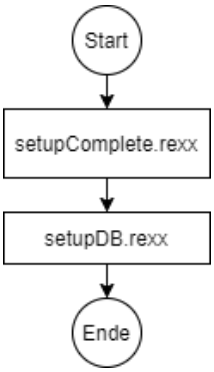

Abbildung 43: Zusammenhang beider Setup-Dateien

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

DriverManager = bsf.import("java.sql.DriverManager")
System = bsf.import("java.lang.System")
JAVA_HOME = value('JAVA_HOME',,'ENVIRONMENT')
Path = value('Path',,'ENVIRONMENT')

IF JAVA_HOME~pos("jdk1.8") > 0 | Path~pos("jdk1.8") > 0 THEN DO
    SAY pp("Java JDK 1.8 found!")
END
ELSE DO
    SAY pp("Java JDK 1.8 not found! Please install the right JDK version, otherwise the POS-
        System might not work")
END

 javaVersion =  System~getProperty("java.version")
IF javaVersion~startsWith("1.8") THEN DO
    SAY pp("Java version 1.8 found!")
END
ELSE DO
    SAY pp("Wrong java version found. Please make sure that the right version is installed
        to ensure the functionality of the POS-System!")
END
------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "setupDB.rexx"
```

Auflistung 1: setupComplete.rexx

```
/*
    This file is used to setup a database for the POS-System
    The database will first be created. After that, tables and tax valus will be added,
    so the POS-System is ready to use.
*/
PARSE SOURCE . . fullPath
CALL directory filespec('L', fullPath)

 .environment~my.app=.directory~new
 .my.app~homeDir = filespec('Location',fullPath)
 directoryExists = 0
 fileName = "databaseData.txt"
 .my.app~DatabaseHandler = .DatabaseHandler~new

 DB_NAME = "POSdatabase"

 Call SysFileTree .my.app~homeDir, "database" , "D"

 DO i = 1 to database.0
     IF database.i~pos("DataBase") <> 0 THEN DO
         directoryExists += 1
     END
     ELSE DO
         directoryExists += 0
     END
 END

 IF directoryExists = 0 THEN DO
     CALL  SysMkDir "DataBase"
 END
  DBpath = .my.app~homeDir || "DataBase"

 CALL directory  DBpath

 path = .my.app~homeDir~changeStr("\", "/")
 DB_URL = "jdbc:sqlite:" || path || "DataBase/"|| DB_NAME || ".db"

IF \SysIsFile(fileName) THEN DO
  databaseData = .stream~new(fileName)
  databaseData~open("write replace")
  databaseData~lineout('[DATABASE Data for later use]')
  databaseData~lineout('DB_NAME = ' || DB_NAME )
  databaseData~lineout('DB_URL = ' || DB_URL )
  databaseData~close
END

  CALL directory .my.app~homeDir
 .my.app~DatabaseHandler~setDBsettings(fileName)
 .my.app~DatabaseHandler~connect
 .my.app~DatabaseHandler~insertTables
```

```
51  .my.app~DatabaseHandler~insertTaxValues
52  .my.app~DatabaseHandler~disconnectDB
53  ------------------------------------------------------
54  ::REQUIRES "BSF.CLS"
55  ::REQUIRES "DatabaseHandler_v2.CLS"
```

Auflistung 2: setupDB.rexx

B.4 Darstellung Hauptprogramm

Dieses Kapitel dient zur Darstellung der programminternen Logik.

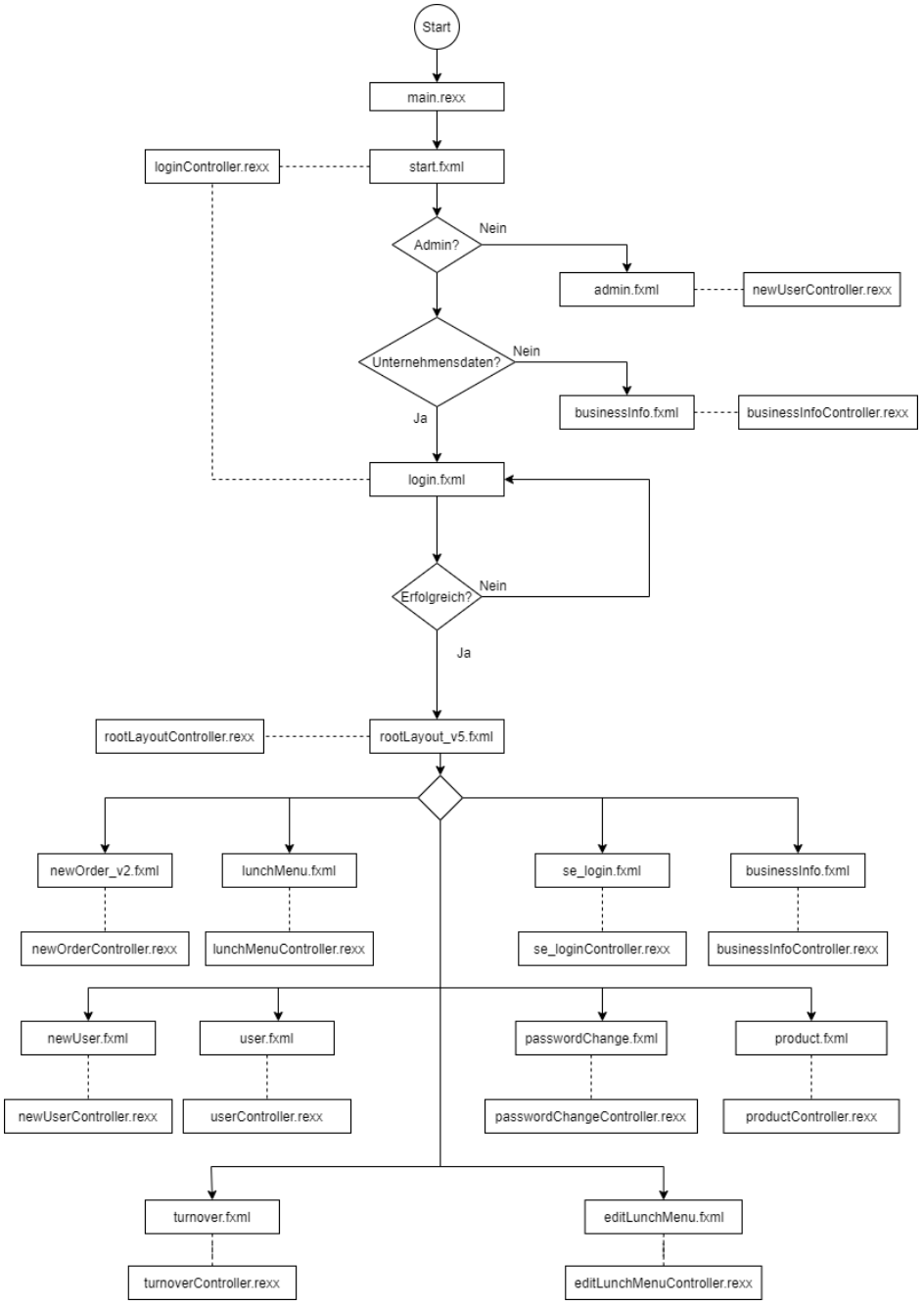

Abbildung 44: Zusammenhang unterschiedlicher Programmabschnitte

Wie in Abbildung 44 ersichtlich erfolgt, nachdem das System erfolgreich in Betrieb genommen wurde (Ausführen der main-Datei), das Startfenster, definiert in Auflistung start.fxml auf der Seite 89. Innerhalb dieses Fenster, wird durch mehrmaliges betätigen des Buttons, eine Abfrage gestartet. Zunächst wird überprüft ob ein Administrator bereits erstellt wurde, wenn dies nicht der Fall ist, erscheint eine Eingabemaske, definiert in Auflistung admin.fxml auf der Seite 76. Durch erneutes betätigen des Knopfes wird zudem überprüft ob bereits Unternehmensdaten systemintern hinterlegt wurden. Wenn diese nicht der Fall ist, erscheint ein Fenster, welches dieses Vorhaben forciert, siehe Kapitel Unternehmensdaten auf der Seite 77. Dieses Kapitel erläutert die benötigten Daten anhand eines Beispiels. Sofern ein Administrator im System hinterlegt ist und auch relevante Daten über das Unternehmen angegeben wurden, erfolgt der Login in das Registrierkassensystem. Sofern der Loginversuch erfolgreich ist, erscheint das Hauptmenü, welches im Kapitel Hauptprogramm auf der Seite 87 näher erläutert wird. Abbildung 44 zeigt unterdessen ebenfalls, dass das Hauptmenü, definiert durch die Auflistung rootLayout_v5.fxml auf der Seite 90 der Dreh- und Angelpunkt des gesamten Systems ist. Vom Hauptmenü aus gelangt man zu allen weiteren Funktionen und wieder zurück.

B.5 Administratorverwaltung

Dieses Kapitel beinhaltet die Auflistung zur Darstellung der Eingabemaske der Administratorzuweisung, sowie die Source-Code-Dateien entsprechender Administratorfunktionen, welche im Kapitel Administrator auf der Seite 29 erläutert wurden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import com.jfoenix.controls.JFXButton?>
4  <?import com.jfoenix.controls.JFXCheckBox?>
5  <?import com.jfoenix.controls.JFXPasswordField?>
6  <?import com.jfoenix.controls.JFXTextField?>
7  <?import javafx.geometry.Insets?>
8  <?import javafx.scene.control.Label?>
9  <?import javafx.scene.layout.BorderPane?>
10 <?import javafx.scene.layout.ColumnConstraints?>
11 <?import javafx.scene.layout.GridPane?>
12 <?import javafx.scene.layout.RowConstraints?>
13 <?import javafx.scene.layout.VBox?>
14 <?language rexx?>
15
16 <BorderPane stylesheets="@DarkTheme.css" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="
       http://javafx.com/fxml/1">
17    <center>
18       <GridPane prefHeight="433.0" prefWidth="402.0">
19          <columnConstraints>
20             <ColumnConstraints hgrow="SOMETIMES" maxWidth="295.0" minWidth="10.0" prefWidth=
                   "233.0" />
21             <ColumnConstraints fillWidth="false" hgrow="SOMETIMES" maxWidth="367.0" minWidth
                   ="10.0" prefWidth="367.0" />
22          </columnConstraints>
23          <rowConstraints>
24             <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
25             <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
26             <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
27             <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
```

```
28              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
29           </rowConstraints>
30           <children>
31              <Label alignment="CENTER" prefHeight="100.0" prefWidth="233.0" text="Firstname:"
                     underline="true" />
32              <Label alignment="CENTER" prefHeight="111.0" prefWidth="233.0" text="Lastname:"
                    underline="true" GridPane.rowIndex="1" />
33              <Label alignment="CENTER" prefHeight="83.0" prefWidth="233.0" text="Username:"
                    underline="true" GridPane.rowIndex="2" />
34              <Label alignment="CENTER" prefHeight="75.0" prefWidth="233.0" text="Password:"
                    underline="true" GridPane.rowIndex="3" />
35              <Label alignment="CENTER" prefHeight="143.0" prefWidth="233.0" text="Repeat
                    Password:" underline="true" GridPane.rowIndex="4" />
36               <JFXTextField fx:id="newUserFNameTxtF" GridPane.columnIndex="1" />
37               <JFXTextField fx:id="newUserLNameTxtF" GridPane.columnIndex="1" GridPane.
                     rowIndex="1" />
38               <JFXTextField fx:id="newUserUNameTxtF" GridPane.columnIndex="1" GridPane.
                     rowIndex="2" />
39               <JFXPasswordField fx:id="newUserPasswordTxtF" GridPane.columnIndex="1" GridPane
                     .rowIndex="3" />
40               <JFXPasswordField fx:id="newUserRepeatPWTxtF" GridPane.columnIndex="1" GridPane
                     .rowIndex="4" />
41               </children>
42        </GridPane>
43     </center>
44     <right>
45        <VBox alignment="BOTTOM_LEFT" prefHeight="433.0" prefWidth="204.0" BorderPane.
              alignment="CENTER">
46           <children>
47              <Label prefHeight="108.0" prefWidth="204.0" text="Rights - New User">
48                 <opaqueInsets>
49                    <Insets />
50                 </opaqueInsets>
51              </Label>
52              <GridPane prefHeight="233.0" prefWidth="204.0">
53                <columnConstraints>
54                  <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
55                </columnConstraints>
56                <rowConstraints>
57                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
58                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
59                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
60                </rowConstraints>
61                 <children>
62                    <JFXCheckBox fx:id="newUcheckBoxAdmin" checkedColor="#105d9c" prefHeight="
                          48.0" prefWidth="195.0" selected="true" text="ADMIN" underline="true"
                          />
63                    <JFXCheckBox fx:id="newUcheckBoxWorker" checkedColor="#105d9c" disable="
                          true" text="WORKER" underline="true" GridPane.rowIndex="1" />
64                    <JFXCheckBox fx:id="newUcheckBoxTrainee" checkedColor="#105d9c" disable="
                          true" text="TRAINEE" underline="true" GridPane.rowIndex="2" />
65                 </children>
66              </GridPane>
67              <JFXButton fx:id="newUserSubmitBtn" onAction="slotDir=arg(arg()); call
                    newUserSubmit slotDir;" prefHeight="93.0" prefWidth="204.0" text="Submit" />
68           </children></VBox>
69     </right>
70      <fx:script source="newUserController.rexx" />
71  </BorderPane>
```

Auflistung 3: admin.fxml

Der dazugehörige Controller ist im Kapitel Benutzerverwaltung auf der Seite 80 dargestellt, siehe Auflistung newUserController.rexx auf der Seite 84. Eine genauere Erläuterung, befindet sich im Hauptteil im Kapitel Anlegen des Administrators auf der Seite 30.

B.5.1 Unternehmensdaten

Dieses Kapitel beinhaltet alle relevanten Dateien, die für die Darstellung der Abfrage von etwaigen Unternehmensdaten benötigt werden. Auflistung

businessInfo.fxml auf Seite 78 definiert das Design und businessInfoController.rexx auf Seite 79 regelt das jeweilige Verwalten bei einem Methodenaufruf.

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

<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXTextField?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?language rexx?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="453.0" prefWidth="563.0" stylesheets="@DarkTheme.css" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <center>
      <GridPane prefHeight="456.0" prefWidth="563.0" BorderPane.alignment="CENTER">
        <columnConstraints>
          <ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" maxWidth="196.0" minWidth="10.0" prefWidth="152.0" />
          <ColumnConstraints hgrow="NEVER" minWidth="10.0" />
        </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 maxHeight="40.0" minHeight="10.0" prefHeight="28.0" vgrow="SOMETIMES" />
            <RowConstraints maxHeight="40.0" minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            <RowConstraints maxHeight="74.0" minHeight="10.0" prefHeight="62.0" vgrow="SOMETIMES" />
        </rowConstraints>
         <children>
            <Label prefHeight="26.0" prefWidth="69.0" text="Vat. ID:" GridPane.halignment="CENTER" GridPane.rowIndex="1" />
            <Label prefHeight="16.0" prefWidth="52.0" text="Name:" />
            <Label text="Telephone:" GridPane.rowIndex="2" />
            <Label text="Address:" GridPane.rowIndex="3" />
            <Label text="City:" GridPane.rowIndex="4" />
            <Label text="Postcode:" GridPane.rowIndex="5" />
            <Label text="e-mail:" GridPane.rowIndex="6" />
            <Label text="Cashbox:" GridPane.rowIndex="7" />
            <Label text="AES256-Key:" GridPane.rowIndex="8" />
            <JFXTextField fx:id="businessName" prefHeight="31.0" prefWidth="333.0" GridPane.columnIndex="1" />
            <JFXTextField fx:id="businessVat" prefWidth="239.0" GridPane.columnIndex="1" GridPane.rowIndex="1" />
            <JFXTextField fx:id="businessTele" prefWidth="239.0" GridPane.columnIndex="1" GridPane.rowIndex="2" />
            <JFXTextField fx:id="businessAddr" prefWidth="239.0" GridPane.columnIndex="1" GridPane.rowIndex="3" />
            <JFXTextField fx:id="businessCity" prefWidth="239.0" GridPane.columnIndex="1" GridPane.rowIndex="4" />
            <JFXTextField fx:id="businessPost" prefWidth="239.0" GridPane.columnIndex="1" GridPane.rowIndex="5" />
            <JFXTextField fx:id="businessEmail" prefWidth="239.0" GridPane.columnIndex="1" GridPane.rowIndex="6" />
            <JFXTextField fx:id="businessCashB" prefWidth="239.0" GridPane.columnIndex="1" GridPane.rowIndex="7" />
            <JFXTextField fx:id="businessAES" prefHeight="31.0" prefWidth="301.0" GridPane.columnIndex="1" GridPane.rowIndex="8" />
            <JFXButton fx:id="businessTestdata" onAction="slotDir=arg(arg()); call fillTestdata slotDir;" prefHeight="30.0" prefWidth="145.0" text="Fill in Testdata" GridPane.rowIndex="10" />
            <Label text="JWSKey:" GridPane.rowIndex="9" />
            <JFXTextField fx:id="businessJWS" GridPane.columnIndex="1" GridPane.rowIndex="9" />
            <GridPane GridPane.columnIndex="1" GridPane.rowIndex="10">
              <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" />
```

```
61                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
62                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
63               </rowConstraints>
64                <children>
65                   <JFXButton fx:id="businessSubmit" onAction="slotDir=arg(arg()); call
                          submit slotDir;" prefHeight="44.0" prefWidth="132.0" text="Submit"
                          GridPane.columnIndex="1" GridPane.halignment="RIGHT" GridPane.rowIndex
                          ="1" />
66                   <JFXButton fx:id="businessGenKey" text="Generate Keys" GridPane.rowIndex="
                          1" />
67                </children>
68             </GridPane>
69          </children>
70       </GridPane>
71    </center>
72     <fx:script source="businessInfoController.rexx" />
73 </BorderPane>
```

Auflistung 4: businessInfo.fxml

```
1  /*@get(businessName businessVat businessTele businessAddr businessCity businessPost
       businessEmail businessCashB businessAES businessJWS)*/
2
3   companyData = .my.app~DatabaseHandler~getCompanyData
4
5   businessName~text = companyData["name"]
6   businessVat~text = companyData["vat_ID"]
7   businessTele~text = companyData["telephone"]
8   businessAddr~text = companyData["address"]
9   businessCity~text = companyData["city"]
10  businessPost~text = companyData["postcode"]
11  businessEmail~text = companyData["e_mail"]
12  businessCashB~text = companyData["cashbox_ID"]
13  businessAES~text = companyData["aes256key"]
14  businessJWS~text = companyData["JWSkey"]
15 ------------------------------------------------------------
16 ::ROUTINE submit PUBLIC
17 /*@get(businessName businessVat businessTele businessAddr businessCity businessPost
       businessEmail businessCashB businessAES businessJWS)*/
18 IF .my.app~DatabaseHandler~checkIFCompanyDataExists THEN DO
19     .my.app~DatabaseHandler~updateCompanyData(businessVat~text,businessName~text,
           businessTele~text,businessAddr~text,businessCity~text,businessPost~text,
           businessEmail~text,businessCashB~text,businessAES~text, businessJWS~text)
20 END
21 ELSE DO
22     .my.app~DatabaseHandler~insertCompanyDetails(businessVat~text,businessName~text,
           businessTele~text,businessAddr~text,businessCity~text,businessPost~text,
           businessEmail~text,businessCashB~text,businessAES~text, businessJWS~text)
23     .my.app~DatabaseHandler~disconnectDB
24 END
25  .my.app~stageHandler~closeWindow
26 ------------------------------------------------------------
27 ::ROUTINE fillTestdata PUBLIC
28 /*@get(businessName businessVat businessTele businessAddr businessCity businessPost
       businessEmail businessCashB businessAES businessJWS)*/
29
30  businessName~text = "Fake GmbH"
31  businessVat~text = "AT123456789"
32  businessTele~text = "0650 123456789"
33  businessAddr~text = "fakestreet 123"
34  businessCity~text = "Wien"
35  businessPost~text = "1010"
36  businessEmail~text = "company@chello.at"
37  businessCashB~text = "CASHBOX1"
38  businessAES~text =  saveToFile("AES")
39  businessJWS~text =  saveToFile("JWS")
40
41 ------------------------------------------------------------
42  ::REQUIRES "collection.rexx"
```

Auflistung 5: businessInfoController.rexx

Eine genauere Erläuterung, befindet sich im Hauptteil im Kapitel Bekanntgabe der Unternehmensdaten auf der Seite 31. Die Speicherung der Daten wird im Anhang in Kapitel Datenbankverwaltung auf der Seite 129 dargestellt.

B.5.2 Benutzerverwaltung

In einem Registrierkassensystem ist es notwendig, dass mehrere Benutzer auf ein und dem selben Programm arbeiten können. Dazu müssen unterschiedliche Benutzer zunächst angelegt und danach verwaltet werden können.

```
<?xml version="1.0" encoding="UTF-8"?>
<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXTextField?>
<?import javafx.scene.control.ChoiceBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.RowConstraints?>
<?language rexx?>
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="742.0" prefWidth="946.0" stylesheets="@DarkTheme.css" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <GridPane gridLinesVisible="true" layoutX="-1.0" layoutY="-1.0" prefHeight="742.0" prefWidth="946.0">
        <columnConstraints>
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="739.0" minWidth="10.0" prefWidth="692.0" />
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="401.0" minWidth="10.0" prefWidth="254.0" />
        </columnConstraints>
        <rowConstraints>
          <RowConstraints fillHeight="false" maxHeight="297.0" minHeight="3.0" prefHeight="3.0" vgrow="SOMETIMES" />
          <RowConstraints maxHeight="610.0" minHeight="10.0" prefHeight="610.0" vgrow="SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
        </rowConstraints>
         <children>
            <TableView fx:id="userTableView" editable="true" prefHeight="200.0" prefWidth="200.0" GridPane.rowIndex="1">
              <columns>
                <TableColumn fx:id="userIDColumn" prefWidth="75.0" text="ID" />
                <TableColumn fx:id="userFNameColumn" prefWidth="101.0" text="Firstname" />
                  <TableColumn fx:id="userLNameColumn" prefWidth="148.0" text="Lastname" />
                  <TableColumn fx:id="userUNameColumn" prefWidth="137.0" text="Username" />
                  <TableColumn fx:id="userPasswordColumn" prefWidth="91.0" text="Password" />
                  <TableColumn fx:id="userRightsColumn" prefWidth="135.0" text="Rights Index" />
              </columns>
            </TableView>
            <GridPane gridLinesVisible="true" prefHeight="442.0" prefWidth="219.0" GridPane.columnIndex="1" GridPane.rowIndex="1">
              <columnConstraints>
                <ColumnConstraints fillWidth="false" hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
              </columnConstraints>
              <rowConstraints>
                <RowConstraints maxHeight="313.0" minHeight="10.0" prefHeight="100.0" vgrow="SOMETIMES" />
                <RowConstraints maxHeight="516.0" minHeight="10.0" prefHeight="510.0" vgrow="SOMETIMES" />
              </rowConstraints>
               <children>
                  <GridPane alignment="CENTER" prefHeight="105.0" prefWidth="254.0">
                    <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>
                     <children>
                        <Label alignment="CENTER" prefHeight="17.0" prefWidth="221.0" text="
```

```
                                Admin = 3" />
57                              <Label alignment="CENTER" prefHeight="17.0" prefWidth="219.0" text="
                                    Worker = 2" GridPane.rowIndex="1" />
58                              <Label alignment="CENTER" prefHeight="17.0" prefWidth="219.0" text="
                                    Trainee = 1" GridPane.rowIndex="2" />
59                           </children>
60                        </GridPane>
61                        <GridPane alignment="CENTER" prefHeight="511.0" prefWidth="252.0" GridPane
                              .rowIndex="1">
62                          <columnConstraints>
63                            <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                                  />
64                          </columnConstraints>
65                          <rowConstraints>
66                              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                    />
67                              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                    />
68                              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                    />
69                              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                    />
70                              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                    />
71                              <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                    />
72                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                  >
73                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                  >
74                          </rowConstraints>
75                           <children>
76                              <Label alignment="CENTER" contentDisplay="CENTER" lineSpacing="1.0"
                                    prefHeight="21.0" prefWidth="269.0" text="Firstname:" underline=
                                    "true" />
77                              <JFXTextField fx:id="userFNameTxtF" promptText="Firstname" GridPane.
                                    rowIndex="1" />
78                              <Label alignment="CENTER" prefHeight="21.0" prefWidth="282.0" text="
                                    Lastname:" underline="true" GridPane.rowIndex="2" />
79                              <JFXTextField fx:id="userLNameTxtF" promptText="Lastname" GridPane.
                                    rowIndex="3" />
80                              <Label alignment="CENTER" prefHeight="21.0" prefWidth="257.0" text="
                                    Username:" underline="true" GridPane.rowIndex="4" />
81                              <JFXTextField fx:id="userUNameTxtF" promptText="Username" GridPane.
                                    rowIndex="5" />
82                              <Label alignment="CENTER" prefHeight="21.0" prefWidth="253.0" text="
                                    Rights Index" underline="true" GridPane.rowIndex="6" />
83                              <ChoiceBox fx:id="rightsChoiceBox" prefHeight="31.0" prefWidth="
                                    259.0" GridPane.rowIndex="7" />
84                           </children>
85                        </GridPane>
86                     </children>
87                  </GridPane>
88                  <Label alignment="CENTER" prefHeight="17.0" prefWidth="324.0" text="Rights -
                        Index" underline="true" GridPane.columnIndex="1" />
89                  <HBox prefHeight="100.0" prefWidth="200.0" GridPane.rowIndex="2">
90                     <children>
91                        <JFXButton fx:id="userDeleteBtn" onAction="slotDir=arg(arg()); call delete
                              slotDir;" prefHeight="79.0" prefWidth="333.0" text="Delete User" />
92                        <JFXButton fx:id="userEditBtn" onAction="slotDir=arg(arg()); call
                              showUserDetails slotDir;" prefHeight="79.0" prefWidth="358.0" text="
                              Edit User" />
93                     </children>
94                  </HBox>
95                  <JFXButton fx:id="userSubmitBtn" onAction="slotDir=arg(arg()); call submit
                        slotDir;" prefHeight="150.0" prefWidth="289.0" text="Submit" GridPane.
                        columnIndex="1" GridPane.rowIndex="2" />
96                  <HBox prefHeight="100.0" prefWidth="200.0">
97                     <children>
98                        <Label alignment="CENTER" contentDisplay="CENTER" prefHeight="53.0"
                              prefWidth="546.0" text="User - Rights: " />
99                        <Label fx:id="userRightsLbl" alignment="CENTER" prefHeight="53.0"
                              prefWidth="558.0" text="Titel" />
100                    </children>
101                 </HBox>
102              </children>
103           </GridPane>
104        </children>
105         <fx:script source="userController.rexx" />
106  </AnchorPane>
```

Auflistung 6: user.fxml

Die Datei, user.fxml auf der Seite 80, definiert das Aussehen der Benutzeroberfläche, wobei userController.rexx auf der Seite 82 die Funktionalität definiert. Dies gilt allerdings nur für bestehende Benutzer. Sofern ein Benutzer neu angelegt werden soll, wird die Eingabemaske, definiert durch newUser.fxml auf der Seite 83, aufgerufen.

```
1  /*@get(userTableView userIDColumn userFNameColumn userLNameColumn userUNameColumn
       userPasswordColumn userRightsColumn userFNameTxtF userLNameTxtF userUNameTxtF
       rightsChoiceBox userDeleteBtn userSubmitBtn userRightsLbl)*/
2  FXCollections = bsf.import("javafx.collections.FXCollections")
3  observRightsArrList = FXCollections~observableArrayList
4  observRightsArrList~add("1")
5  observRightsArrList~add("2")
6  observRightsArrList~add("3")
7  rightsChoiceBox~setItems(observRightsArrList)
8  userRightsLbl~setText("ADMIN - View")
9  usersArrayList = .my.app~DatabaseHandler~getAlluserArrayList
10 signal on syntax
11 userHandler = .UserHandler~new(usersArrayList, userTableView, userIDColumn, userFNameColumn,
       userLNameColumn, userUNameColumn, userPasswordColumn, userRightsColumn)
12 ------------------------------------------------------------
13 ::ROUTINE showUserDetails PUBLIC
14 /*@get( userTableView userFNameTxtF userLNameTxtF userUNameTxtF rightsChoiceBox
       userRightsLbl)*/
15 item = userTableView~getSelectionModel~getSelectedItem
16 user = BsfRexxProxy(item)
17 .my.app~userID = user~userID~get
18 userFNameTxtF~text = user~firstName~get
19 userLNameTxtF~text = user~lastName~get
20 userUNameTxtF~text = user~userName~get
21 rightsChoiceBox~setValue(user~rightsIndex~get)
22 ------------------------------------------------------------
23 ::ROUTINE submit PUBLIC
24 /*@get(userTableView userIDColumn userFNameColumn userLNameColumn userUNameColumn
       userPasswordColumn userRightsColumn userFNameTxtF userLNameTxtF userUNameTxtF
       rightsChoiceBox)*/
25 IF .my.app~DatabaseHandler~getUserIDByUsername(.my.app~userID) THEN DO
26 userUNameTxtF~text rightsChoiceBox~getValue
27 END
28 ElSE DO
29 .my.app~DatabaseHandler~updateUser(.my.app~userID, userFNameTxtF~text, userLNameTxtF~text,
       userUNameTxtF~text,rightsChoiceBox~getValue)
30 usersArrayList = .my.app~DatabaseHandler~getAlluserArrayList
31 userHandler = .UserHandler~new(usersArrayList, userTableView, userIDColumn, userFNameColumn,
       userLNameColumn, userUNameColumn, userPasswordColumn, userRightsColumn)
32 userFNameTxtF~text = ""
33 userLNameTxtF~text = ""
34 userUNameTxtF~text = ""
35 rightsChoiceBox~getSelectionModel~clearSelection
36 END
37 ------------------------------------------------------------
38 ::ROUTINE delete PUBLIC
39 /*@get(userTableView userIDColumn userFNameColumn userLNameColumn userUNameColumn
       userPasswordColumn userRightsColumn userFNameTxtF userLNameTxtF userUNameTxtF
       rightsChoiceBox)*/
40 rightsChoiceBox~getValue
41 item = userTableView~getSelectionModel~getSelectedItem
42 user = BsfRexxProxy(item)
43 IF user~rightsIndex~get = 3 THEN DO
44 .my.app~stageHandler~showErrorWindow("ERROR","Admin can not be deleted","ERROR")
45 END
46 ELSE DO
47 .my.app~DatabaseHandler~deleteUser(user~userID~get)
48 usersArrayList = .my.app~DatabaseHandler~getAlluserArrayList
49 userHandler = .UserHandler~new(usersArrayList, userTableView, userIDColumn, userFNameColumn,
       userLNameColumn, userUNameColumn, userPasswordColumn, userRightsColumn)
50 userFNameTxtF~text = ""
51 userLNameTxtF~text = ""
52 userUNameTxtF~text = ""
53 rightsChoiceBox~getSelectionModel~clearSelection
54 END
55 ------------------------------------------------------------
56 ::REQUIRES "BSF.CLS"
57 ::REQUIRES "DatabaseHandler_v2.CLS"
58 ::REQUIRES "TableH.CLS"
```

Auflistung 7: userController.rexx

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

<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXCheckBox?>
<?import com.jfoenix.controls.JFXPasswordField?>
<?import com.jfoenix.controls.JFXTextField?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Label?>
<?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?>
<?language rexx?>

<BorderPane stylesheets="@DarkTheme.css" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="
    http://javafx.com/fxml/1">
   <center>
      <GridPane prefHeight="433.0" prefWidth="402.0">
         <columnConstraints>
            <ColumnConstraints hgrow="SOMETIMES" maxWidth="295.0" minWidth="10.0" prefWidth=
                "233.0" />
            <ColumnConstraints fillWidth="false" hgrow="SOMETIMES" maxWidth="367.0" minWidth
                ="10.0" prefWidth="367.0" />
         </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>
            <Label alignment="CENTER" prefHeight="100.0" prefWidth="233.0" text="Firstname:"
                 underline="true" />
            <Label alignment="CENTER" prefHeight="111.0" prefWidth="233.0" text="Lastname:"
                underline="true" GridPane.rowIndex="1" />
            <Label alignment="CENTER" prefHeight="83.0" prefWidth="233.0" text="Username:"
                underline="true" GridPane.rowIndex="2" />
            <Label alignment="CENTER" prefHeight="75.0" prefWidth="233.0" text="Password:"
                underline="true" GridPane.rowIndex="3" />
            <Label alignment="CENTER" prefHeight="143.0" prefWidth="233.0" text="Repeat
                Password:" underline="true" GridPane.rowIndex="4" />
             <JFXTextField fx:id="newUserFNameTxtF"  GridPane.columnIndex="1" />
             <JFXTextField fx:id="newUserLNameTxtF" GridPane.columnIndex="1" GridPane.
                 rowIndex="1" />
             <JFXTextField fx:id="newUserUNameTxtF" GridPane.columnIndex="1" GridPane.
                 rowIndex="2" />
             <JFXPasswordField fx:id="newUserPasswordTxtF" GridPane.columnIndex="1" GridPane
                 .rowIndex="3" />
             <JFXPasswordField fx:id="newUserRepeatPWTxtF" GridPane.columnIndex="1" GridPane
                 .rowIndex="4" />
             </children>
      </GridPane>
   </center>
   <right>
      <VBox alignment="BOTTOM_LEFT" prefHeight="433.0" prefWidth="204.0" BorderPane.
          alignment="CENTER">
         <children>
            <Label prefHeight="108.0" prefWidth="204.0" text="Rights - New User">
               <opaqueInsets>
                  <Insets />
               </opaqueInsets>
            </Label>
            <GridPane prefHeight="233.0" prefWidth="204.0">
              <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>
               <children>
                  <JFXCheckBox fx:id="newUcheckBoxAdmin" checkedColor="#105d9c" prefHeight="
                      48.0" prefWidth="195.0" text="ADMIN" underline="true" />
                  <JFXCheckBox fx:id="newUcheckBoxWorker" checkedColor="#105d9c" text="
                      WORKER" underline="true" GridPane.rowIndex="1" />
                  <JFXCheckBox fx:id="newUcheckBoxTrainee" checkedColor="#105d9c" text="
                      TRAINEE" underline="true" GridPane.rowIndex="2" />
               </children>
            </GridPane>
            <JFXButton fx:id="newUserSubmitBtn" onAction="slotDir=arg(arg()); call
```

```
                    newUserSubmit slotDir;" prefHeight="93.0" prefWidth="204.0" text="Submit" />
68          </children></VBox>
69      </right>
70       <fx:script source="newUserController.rexx" />
71  </BorderPane>
```

Auflistung 8: newUser.fxml

Auflistung newUser.fxml definiert das Design des GUIs für neue Benutzer, newUserController.rexx beinhaltet die dazugehörigen Methoden.

```
1
2   ::ROUTINE newUserSubmit PUBLIC
3   use arg slotDir
4   /*@get(newUserPasswordTxtF newUserRepeatPWTxtF newUserFNameTxtF newUserLNameTxtF
        newUserUNameTxtF newUcheckBoxAdmin newUcheckBoxWorker newUcheckBoxTrainee)*/
5
6   IF newUcheckBoxAdmin~isSelected THEN DO
7       rightsInd = 3
8   END
9   IF newUcheckBoxWorker~isSelected THEN DO
10      rightsInd = 2
11  END
12  IF newUcheckBoxTrainee~isSelected THEN DO
13      rightsInd = 1
14  END
15
16
17  IF .my.app~DatabaseHandler~checkIfUserNameExist(newUserUNameTxtF~text) THEN DO
18      .my.app~stageHandler~showErrorWindow("Username already exist", 'The chosen username
            already exist', "ERROR")
19  END
20  ELSE IF newUserPasswordTxtF~text <> newUserRepeatPWTxtF~text THEN DO
21      .my.app~stageHandler~showErrorWindow("Password do not match", '"Password" and "Repeat
            Password" do not match!', "ERROR")
22  END
23  ELSE DO
24      .my.app~DatabaseHandler~insertUser(newUserFNameTxtF~text, newUserLNameTxtF~text,
            newUserUNameTxtF~text, newUserPasswordTxtF~text, rightsInd)
25      .my.app~stageHandler~closeWindow
26  END
```

Auflistung 9: newUserController.rexx

Eine genauere Erläuterung, befindet sich im Hauptteil im Kapitel Benutzerverwaltung auf der Seite 36. Die Abspeicherung der Daten wird im Anhang in Kapitel Datenbankverwaltung auf der Seite 129 dargestellt.

B.5.3 Umsatzzähler

Die Registrierkassensicherheitsverordnung gibt die Rahmenbedingungen für solch ein Registrierkassensystem vor. Nähere Informationen dafür stehen im Hauptteil im Kapitel Generelle Rahmenbedingungen eines Registrierkassensystems auf der Seite 14. Im Grund wird vorgeschrieben, dass jedmögliche Art der Barzahlung, die den Umsatz des Unternehmens beeinflusst, mit protokolliert werden muss. So muss jede Kasse eine Funktion aufweisen, die zu jedem möglichen Zeitpunkt den genauen Umsatzzähler ausweist, bzw. verschlüsselt am Beleg innerhalb eines QR-Codes hinterlegt. Im Kapitel Umsatzzähler auf der Seite 39 stehen nähere Informationen bezüglich der Darstellung des Umsatzzählers. Die grafische Darstellung des Umsatzzählers wird durch die

Auflistung turnover.fxml in Kombination mit der Auflistung turnoverController.rexx unterhalb definiert.

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

<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXTextField?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?language rexx?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
    Infinity" prefHeight="295.0" prefWidth="397.0" stylesheets="@DarkTheme.css" xmlns="http
    ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <center>
      <GridPane alignment="CENTER" prefHeight="268.0" prefWidth="600.0" BorderPane.alignment
          ="CENTER">
        <columnConstraints>
          <ColumnConstraints halignment="RIGHT" hgrow="SOMETIMES" minWidth="10.0" prefWidth=
              "100.0" />
          <ColumnConstraints halignment="CENTER" hgrow="SOMETIMES" minWidth="10.0" prefWidth
              ="100.0" />
        </columnConstraints>
        <rowConstraints>
            <RowConstraints maxHeight="42.0" minHeight="8.0" prefHeight="8.0" vgrow="
                SOMETIMES" />
          <RowConstraints maxHeight="86.0" minHeight="10.0" prefHeight="86.0" vgrow="
                SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
          <RowConstraints maxHeight="39.0" minHeight="10.0" prefHeight="39.0" vgrow="
                SOMETIMES" />
            <RowConstraints maxHeight="70.0" minHeight="10.0" prefHeight="46.0" valignment="
                TOP" vgrow="SOMETIMES" />
            <RowConstraints maxHeight="61.0" minHeight="10.0" prefHeight="38.0" valignment="
                TOP" vgrow="SOMETIMES" />
        </rowConstraints>
         <children>
            <Label text="Current Turnovervalue:" GridPane.rowIndex="1" />
            <Label text="Goal Turnovervalue:" GridPane.rowIndex="2" />
            <Label fx:id="missingTurnLbl" text="Missing:" GridPane.rowIndex="4" />
            <Label fx:id="curTurn" text="EUR: 0.00" GridPane.columnIndex="1" GridPane.
                rowIndex="1" />
            <Label fx:id="missingTurn" text="EUR: 0.00" GridPane.columnIndex="1" GridPane.
                rowIndex="4" />
            <Separator prefWidth="200.0" GridPane.rowIndex="3" />
            <Separator prefWidth="200.0" GridPane.columnIndex="1" GridPane.rowIndex="3" />
            <Separator prefWidth="200.0" />
            <Separator prefWidth="200.0" GridPane.columnIndex="1" />
            <JFXTextField fx:id="goalTurn" alignment="CENTER" GridPane.columnIndex="1"
                GridPane.rowIndex="2" />
            <JFXButton onAction="slotDir=arg(arg()); call calculate slotDir;" prefHeight="
                46.0" prefWidth="107.0" text="Calculate" GridPane.halignment="LEFT" GridPane
                .rowIndex="5" />
         </children>
      </GridPane>
   </center>
   <top>
      <Label alignment="CENTER" prefHeight="62.0" prefWidth="336.0" text="Turnover -
          Overview" BorderPane.alignment="CENTER" />
   </top>
    <fx:script source="turnoverController.rexx" />
</BorderPane>
```

Auflistung 10: turnover.fxml

```
/*@get(curTurn)*/

 TurnoverValue = .my.app~DatabaseHandler~getTurnoverValue(1)

 curTurn~setText(formatToEuro(TurnoverValue))

------------------------------------------------------------
::ROUTINE calculate PUBLIC
/*@get(missingTurnLbl curTurn goalTurn missingTurn)*/
 value = 0
 Color = bsf.loadClass("javafx.scene.paint.Color")
 current = reformatEUR(curTurn~getText)
 goal = goalTurn~text
 goal = format2float(goal)

 value = format2float(goal - current)
IF value > 0 THEN DO
 missingTurnLbl~setText("Missing:")
 missingTurnLbl~setTextFill(Color~RED)
 missingTurn~setText(formatToEuro(value))
END
ELSE DO
 value = ABS(value)
 missingTurnLbl~setText("Above Goal:")
 missingTurnLbl~setTextFill(Color~GREEN)
 missingTurn~setText(formatToEuro(value))
END
------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "collection.rexx"
```

Auflistung 11: turnoverController.rexx

B.6 Loginverwaltung

Für den sachgerechten Umgang mit einem elektronischen Kassensystem ist es wichtig, dass nur Befugte Zugang zu solch einem POS-System haben. Diesbezüglich wurde eine Login-Maske entwickelt, mit zugehöriger Passwortverwaltung, welche genauer im Kapitel Passwortverwaltung auf Seite 114 beschrieben wird. Die Login-Maske wird durch die Datei login.fxml auf Seite 86 definiert. Die zugehörige Logik befindet sich wiederum in der dazugehörigen Controller - Datei, Auflistung loginController.rexx auf Seite 87.

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

<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXPasswordField?>
<?import com.jfoenix.controls.JFXTextField?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?language rexx?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
    Infinity" prefHeight="400.0" prefWidth="600.0" stylesheets="@DarkTheme.css" xmlns="http
    ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <GridPane prefHeight="301.0" prefWidth="600.0">
        <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>
         <children>
```

```
25              <Label alignment="CENTER" text="Point of Sales - Login" textAlignment="CENTER"
                    GridPane.halignment="CENTER" />
26              <JFXTextField fx:id="loginUsernameTxtF" alignment="CENTER" promptText="Username
                    ..." GridPane.rowIndex="1" />
27              <JFXPasswordField fx:id="loginPasswordTxtF" alignment="CENTER" promptText="
                    Password..." GridPane.rowIndex="2" />
28          </children>
29      </GridPane>
30      <JFXButton fx:id="loginBtn" alignment="CENTER" contentDisplay="CENTER" defaultButton="
            true" layoutX="218.0" layoutY="313.0" onAction="slotDir=arg(arg()); call loginPOS
            slotDir;" prefHeight="33.0" prefWidth="164.0" text="Login" textAlignment="CENTER"
            underline="true" />
31  </children>
32   <fx:script source="loginController.rexx" />
33 </AnchorPane>
```

Auflistung 12: login.fxml

```
1  ::ROUTINE startPOS PUBLIC
2   loginSuccessful = .my.app~DatabaseHandler~connect
3
4  IF loginSuccessful = 1 THEN DO
5      CALL firstGUI
6  END
7  ELSE DO
8      .my.app~stageHandler~showErrorWindow("ERROR", "Connection to database can not be
           initialised!", "ERROR")
9  END
10
11 ------------------------------------------------------------
12 ::ROUTINE loginPOS PUBLIC
13  USE ARG slotDir
14  /*@get(loginUsernameTxtF loginPasswordTxtF loginBtn)*/
15  userID = .my.app~DatabaseHandler~getUserIDByUsername(loginUsernameTxtF~text)
16  check = .my.app~DatabaseHandler~checkPassword(userID, loginPasswordTxtF~text)
17
18 IF check THEN DO
19     .my.app~activeUser = .my.app~DatabaseHandler~getUserInfosSetActive(loginUsernameTxtF~
           text)
20     .my.app~stageHandler~closeWindow
21     .my.app~stageHandler~loadScene("POS System - Active", "file:rootLayout_v5.fxml")
22 END
23 ELSE DO
24     .my.app~stageHandler~showErrorWindow("WARNING", "Password or username is wrong!", "
           WARNING")
25 END
26 ------------------------------------------------------------
27 ::REQUIRES "BSF.CLS"
28 ::REQUIRES "DatabaseHandler_v2.CLS"
29 ::REQUIRES "collection.rexx"
```

Auflistung 13: loginController.rexx

Eine genauere Erläuterung des Loginprozesses, befindet sich im Hauptteil im Kapitel Login auf der Seite 33.

B.7 Hauptprogramm

Anschließend an die Initialisierung und dem erfolgreichen Login erfolgt das Ausführen des Hauptprogramms. Für einen generellen Überblick über die Programmbestandteile siehe Kapitel Darstellung Hauptprogramm auf der Seite 75. Für das Ausführen des Hauptprogramms muss die Datei main.rexx auf Seite 88 ausgeführt werden. Diese Datei startet das Registrierkassensystem und zeigt zunächst die Startseite, definiert in Auflistung start.fxml auf der Seite 89.

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

CALL bsf.import "javafx.fxml.FXMLLoader",                       "fx.FXMLLoader"
CALL bsf.import "javafx.scene.Scene",                           "fx.Scene"
CALL bsf.import "javafx.beans.property.SimpleStringProperty",   "fx.SimpleStringProperty"
CALL bsf.import "javafx.beans.property.SimpleIntegerProperty", "fx.SimpleIntegerProperty"

CALL bsf.import "javafx.collections.FXCollections",             "fx.FXCollections"
CALL bsf.import "javafx.stage.Modality",                        "fx.Modality"
CALL bsf.import "javafx.scene.control.Alert",                   "fx.Alert"
CALL bsf.import "javafx.scene.control.Alert$AlertType",         "fx.Alert.Type"

 .environment~my.app=.directory~new
 .my.app~homeDir = filespec('Location',fullPath)
 stageHandler = .StageHandler~new
 .my.app~stageHandler = stageHandler
 .my.app~DatabaseHandler = .DatabaseHandler~new

 fileName = "databaseData.txt"
 .my.app~DatabaseHandler~setDBsettings(fileName)

 stageHandlerProxy = BsfCreateRexxProxy(stageHandler,,"javafx.application.Application")
 SIGNAL ON SYNTAX
 stageHandlerProxy~launch(stageHandlerProxy~getClass, .nil)

 EXIT 0
------------------------------------------------------------
syntax:
   say "ERROR: Stagehandler failed to load!"
------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "PasswordHandler.cls"
::REQUIRES "DatabaseHandler_v2.cls"
::REQUIRES "AESHandler.cls"
::REQUIRES "QRCodeHandler.CLS"
------------------------------------------------------------
::CLASS StageHandler
::ATTRIBUTE primaryStage
::ATTRIBUTE scene
::ATTRIBUTE windowStage
::METHOD start
 EXPOSE primaryStage scene
 USE ARG primaryStage
 primaryStage~setTitle("Start")
 rootLayoutUrl=.bsf~new("java.net.URL", "file:start.fxml")
 rootLayout   =.fx.FXMLLoader~load(rootLayoutUrl)
 scene=.fx.scene~new(rootLayout)
 primaryStage~setScene(scene)
 primaryStage~show
------------------------------------------------------------
::METHOD loadScene
 EXPOSE primaryStage
 USE ARG title, fileName
 primaryStage~setTitle(title)
 url =.bsf~new("java.net.URL", fileName)
 fxml = .fx.FXMLLoader~load(url)
 primaryStage~getScene~setRoot(fxml)
------------------------------------------------------------
::METHOD newWindow
 EXPOSE primaryStage windowStage
 USE ARG title, fileName
 windowStage = .bsf~new("javafx.stage.Stage")
 windowStage~setTitle(title)
 url =.bsf~new("java.net.URL", fileName)
 fxml = .fx.FXMLLoader~load(url)
 scene = .bsf~new("javafx.scene.Scene", fxml)
 windowStage~setScene(scene)
 windowStage~initModality(bsf.loadClass("javafx.stage.Modality")~WINDOW_MODAL)
 windowStage~initOwner(primaryStage)
 windowStage~show
------------------------------------------------------------
::METHOD closeWindow
 EXPOSE primaryStage windowStage
 window = windowStage~getScene~getWindow
 window~close
------------------------------------------------------------
::METHOD showErrorWindow
 EXPOSE primaryStage windowStage
 USE ARG titel, content, type
IF type == "ERROR" THEN DO
    alert=.fx.alert~new(.fx.Alert.Type~ERROR)
    alert~setHeaderText("An Error has occured!")
```

```
84  END
85  IF type == "WARNING" THEN DO
86      alert=.fx.alert~new(.fx.Alert.Type~WARNING)
87      alert~setHeaderText("Something went wrong!")
88  END
89  IF type == "INFORMATION" THEN DO
90      alert=.fx.alert~new(.fx.Alert.Type~INFORMATION)
91      alert~setHeaderText("Success!")
92  END
93   alert~setTitle(titel)
94   alert~setContentText(content)
95   alert~initOwner(primaryStage)
96   alertWindow = alert~getDialogPane~getScene~getWindow
97   alertWindow~setAlwaysOnTop(.true)
98   alert~showAndWait
99  ------------------------------------------------------------
100 ::METHOD backToRoot PUBLIC
101  EXPOSE primaryStage
102  USE ARG title, fileName
103  primaryStage~setTitle("POS System - Active")
104  url =.bsf~new("java.net.URL", "file:rootLayout_v5.fxml")
105  fxml = .fx.FXMLLoader~load(url)
106  primaryStage~getScene~setRoot(fxml)
```

Auflistung 14: main.rexx

Das Hauptfenster des POS-Systems wird durch die Datei rootLayout_v5.fxml auf Seite 90 definiert. Diese Datei definiert das Design und definiert welche Methode bei welchem Klick eines Buttons ausgeführt werden soll. Die Funktionalität der angesprochenen Methode wird unterdessen in einer eigenen Controller-Datei definiert. Jedes GUI, Graphical User Interface definiert durch eine entsprechende FXML-Datei, besitzt eine Zugehörigkeit zu einem Controller. Dieser kann entweder eine eigenständige Datei sein, oder ein Controller verwaltet mehrere GUIs. Für das bessere Verständnis, welche Eingabemaske bzw. welches Interface zu welchem Controller gehört, siehe Kapitel Darstellung Hauptprogramm auf der Seite 75. Dieses Kapitel erläutert die interne Systemlogik. Eine detailliertere Erläuterung der gesamten Funktionalität des Hauptprogramms, befindet sich im Hauptteil im Kapitel Hauptprogramm auf der Seite 35.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import com.jfoenix.controls.JFXButton?>
4  <?import javafx.scene.control.Label?>
5  <?import javafx.scene.layout.BorderPane?>
6  <?language rexx?>
7
8  <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
       Infinity" prefHeight="616.0" prefWidth="900.0" stylesheets="@DarkTheme.css" xmlns="http
       ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
9     <top>
10       <Label alignment="CENTER" prefHeight="206.0" prefWidth="636.0" text="To start your
             Point-of-Service-System, please hit the Button below" textAlignment="CENTER"
             BorderPane.alignment="CENTER" />
11    </top>
12    <center>
13       <JFXButton prefHeight="366.0" prefWidth="537.0" text="Get ready to work..." onAction="
             slotDir=arg(arg()); call startPOS slotDir;" BorderPane.alignment="CENTER" />
14    </center>
15    <fx:script source="loginController.rexx" />
16 </BorderPane>
```

Auflistung 15: start.fxml

Auflistung start.fxml erzeugt das Startfenster des Registrierkassensystems.

Der dazugehörige Controller ist im Kaptitel Loginverwaltung auf der Seite 86 definiert, siehe Auflistung loginController.rexx auf der Seite 87.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import com.jfoenix.controls.JFXButton?>
4  <?import javafx.scene.control.Menu?>
5  <?import javafx.scene.control.MenuBar?>
6  <?import javafx.scene.control.MenuItem?>
7  <?import javafx.scene.control.SplitPane?>
8  <?import javafx.scene.control.TableColumn?>
9  <?import javafx.scene.control.TableView?>
10 <?import javafx.scene.layout.AnchorPane?>
11 <?import javafx.scene.layout.BorderPane?>
12 <?import javafx.scene.layout.HBox?>
13 <?import javafx.scene.layout.VBox?>
14 <?language rexx?>
15
16 <BorderPane fx:id="rootBorder" prefHeight="848.0" prefWidth="1229.0" stylesheets="@../
       RegKassa/DarkTheme.css" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.
       com/fxml/1">
17    <center>
18       <SplitPane fx:id="rootSplit" dividerPositions="0.32955832389580975" prefHeight="160.0"
             prefWidth="200.0" BorderPane.alignment="CENTER">
19          <items>
20             <AnchorPane fx:id="rsAnchorLeft" minHeight="0.0" minWidth="0.0" prefHeight="160.0"
                   prefWidth="100.0">
21                <children>
22                   <TableView fx:id="rsTableViewLeft" prefHeight="525.0" prefWidth="332.0"
                         AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.
                         rightAnchor="-2.0" AnchorPane.topAnchor="0.0">
23                      <columns>
24                         <TableColumn fx:id="rsTableColumnOID" prefWidth="91.0" text="OrderID
                               " />
25                         <TableColumn fx:id="rsTableColumnTNr" prefWidth="106.0" text="
                               TableNr" />
26                         <TableColumn fx:id="rsTableColumnRType" prefWidth="151.0" text="R-
                               Type" />
27                      </columns>
28                   </TableView>
29                </children>
30             </AnchorPane>
31             <AnchorPane fx:id="rsAnchorRight" minHeight="0.0" minWidth="0.0" prefHeight="160.0
                   " prefWidth="100.0">
32                <children>
33                   <TableView fx:id="rsTableViewRight" prefHeight="596.0" prefWidth="509.0"
                         AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0" AnchorPane.
                         rightAnchor="-2.0" AnchorPane.topAnchor="0.0">
34                      <columns>
35                         <TableColumn fx:id="rsTableColumnPID" prefWidth="93.0" text="
                               ProductID" />
36                         <TableColumn fx:id="rsTableColumnName" prefWidth="235.0" text="Name"
                                />
37                         <TableColumn fx:id="rsTableColumnPrice" prefWidth="98.0" text="Price
                               " />
38                         <TableColumn fx:id="rsTableColumnCont" prefWidth="184.0" text="
                               Status" />
39                      </columns>
40                   </TableView>
41                </children>
42             </AnchorPane>
43          </items>
44       </SplitPane>
45    </center>
46    <right>
47       <VBox fx:id="rootVBox" prefHeight="597.0" prefWidth="150.0" BorderPane.alignment="
             CENTER">
48          <children>
49             <JFXButton fx:id="rvNewOrderBtn" maxWidth="-Infinity" onAction="slotDir=arg(arg
                   ()); call handleNewOrder slotDir;" prefHeight="400.0" prefWidth="400.0" text
                   ="New Order" />
50             <JFXButton fx:id="rvEditOrderBtn" onAction="slotDir=arg(arg()); call
                   handleEditOrder slotDir;" prefHeight="400.0" prefWidth="400.0" text="Edit
                   Order" />
51             <JFXButton fx:id="rvLoginATBtn" onAction="slotDir=arg(arg()); call handleLogin
                   slotDir;" prefHeight="400.0" prefWidth="400.0" text="Login A-Trust" />
52             <JFXButton fx:id="rvLunchMenuBtn" onAction="slotDir=arg(arg()); call
                   handleLunchMenu slotDir;" prefHeight="400.0" prefWidth="400.0" text="Lunch-
                   Menu" />
53          </children>
54       </VBox>
55    </right>
```

```
56      <bottom>
57         <HBox fx:id="rootHBox" prefHeight="101.0" prefWidth="794.0" BorderPane.alignment="
               CENTER">
58            <children>
59               <JFXButton fx:id="rhPayBtn" onAction="slotDir=arg(arg()); call handlePay slotDir
                     ;" prefHeight="101.0" prefWidth="359.0" text="Pay" />
60               <JFXButton fx:id="rhSplitBtn" onAction="slotDir=arg(arg()); call handleSplit
                     slotDir;" prefHeight="101.0" prefWidth="472.0" text="Split-Pay" />
61               <JFXButton fx:id="rhLogoutBtn" onAction="slotDir=arg(arg()); call handleLogout
                     slotDir;" prefHeight="400.0" prefWidth="400.0" text="Logout" />
62            </children>
63         </HBox>
64      </bottom>
65      <top>
66         <MenuBar BorderPane.alignment="CENTER">
67           <menus>
68             <Menu mnemonicParsing="false" text="File">
69               <items>
70                     <MenuItem fx:id="menuBussiness" mnemonicParsing="false" onAction="slotDir=
                           arg(arg()); call handleBusiness slotDir;" text="Business Information"
                           />
71                     <MenuItem fx:id="menuDEPExp" mnemonicParsing="false" onAction="slotDir=arg
                           (arg()); call handleDEPExp slotDir;" text="DEP-Export" />
72                     <MenuItem fx:id="menuQuit" mnemonicParsing="false" onAction="slotDir=arg(
                           arg()); call handleQuit slotDir;" text="Quit" />
73               </items>
74             </Menu>
75             <Menu mnemonicParsing="false" text="User">
76               <items>
77                 <MenuItem fx:id="menuNewUser" mnemonicParsing="false" onAction="slotDir=arg(
                        arg()); call handleNewUser slotDir;" text="New User" />
78                     <MenuItem fx:id="menuEditUser" mnemonicParsing="false" onAction="slotDir=
                           arg(arg()); call handleEditUser slotDir;" text="Edit User" />
79                     <MenuItem fx:id="menuChangePassword" mnemonicParsing="false" onAction="
                           slotDir=arg(arg()); call handleChangePassword slotDir;" text="Change
                           Password" />
80               </items>
81             </Menu>
82               <Menu mnemonicParsing="false" text="Statistics">
83                 <items>
84                   <MenuItem fx:id="menuShowTurnOver" mnemonicParsing="false" onAction="slotDir
                          =arg(arg()); call handleShowTurnOver slotDir;" text="Show Turnover" />
85                 </items>
86               </Menu>
87               <Menu mnemonicParsing="false" text="Lunch - Menu">
88                 <items>
89                   <MenuItem fx:id="menuEditLunch" mnemonicParsing="false" onAction="slotDir=
                          arg(arg()); call handleEditLunch slotDir;" text="Edit Lunch-Menu" />
90                 </items>
91               </Menu>
92             <Menu mnemonicParsing="false" text="Product">
93               <items>
94                 <MenuItem fx:id="menuNewProduct" mnemonicParsing="false" onAction="slotDir=arg
                        (arg()); call handleNewProduct slotDir;" text="New Product" />
95                     <MenuItem fx:id="menuEditProduct" mnemonicParsing="false" onAction="
                           slotDir=arg(arg()); call handleEditProduct slotDir;" text="Edit
                           Product" />
96               </items>
97             </Menu>
98           </menus>
99         </MenuBar>
100     </top>
101      <fx:script source="rootLayoutController.rexx" />
102 </BorderPane>
```

Auflistung 16: rootLayout_v5.fxml

Der dazugehörige Controller zum Hauptfenster wird definiert durch die Datei rootLayoutController.rexx auf Seite 91.

```
1
2  /*@get(rsTableViewLeft rsTableColumnOID rsTableColumnTNr rsTableViewRight rsTableColumnPID
       rsTableColumnName rsTableColumnPrice rsTableColumnCont rsTableColumnRType)*/
3  /*@get(menuBussiness menuDEPExp menuNewUser menuEditUser menuChangePassword menuShowTurnOver
       menuEditLunch menuNewProduct menuEditProduct)*/
4  /*@get(rvNewOrderBtn rvEditOrderBtn rvLoginATBtn rvLunchMenuBtn rhPayBtn rhSplitBtn)*/
5   FXCollections = bsf.import("javafx.collections.FXCollections")
6
7  IF .my.app~activeUser~rightsInd = 2 THEN DO
8      menuBussiness~disable = .true
```

```
    menuDEPExp~disable = .true
    menuNewUser~disable = .true
    menuEditUser~disable = .true
    menuShowTurnOver~disable = .true
END
IF .my.app~activeUser~rightsInd = 1 THEN DO
    menuBussiness~disable = .true
    menuDEPExp~disable = .true
    menuNewUser~disable = .true
    menuEditUser~disable = .true
    menuShowTurnOver~disable = .true
    menuNewProduct~disable = .true
    menuEditProduct~disable = .true
    rvLoginATBtn~disable = .true
    rhPayBtn~disable = .true
    rhSplitBtn~disable = .true
END

 orderDetailsArrayList = FXCollections~observableArrayList
 productsArrayList = FXCollections~observableArrayList
 newOrderDetailsArrayList = FXCollections~observableArrayList
 orderArrayList=.my.app~DatabaseHandler~getAllOpenOrders

 .my.app~orderHandler = .OrderHandler~new(orderArrayList, rsTableViewLeft, rsTableColumnOID,
       rsTableColumnTNr, rsTableColumnRType)
 .my.app~orderDetailsHandler = .OrderDetailsHandler~new(orderDetailsArrayList,
      rsTableViewRight, rsTableColumnPID, rsTableColumnName, rsTableColumnPrice,
      rsTableColumnCont)
-------------------------------------------------------
::ROUTINE handleNewOrder PUBLIC
 .my.app~currentOrderID = .nil
 .my.app~currentTableNR = .nil
 .my.app~stageHandler~newWindow("New Order", "file:newOrder_v2.fxml")
-------------------------------------------------------
::ROUTINE handleEditOrder PUBLIC
/*@get(rsTableViewLeft)*/
 item = rsTableViewLeft~getSelectionModel~getSelectedItem
IF item <> .nil THEN DO
    .my.app~stageHandler~newWindow("Edit Order", "file:newOrder_v2.fxml")
END
ELSE DO
    .my.app~stageHandler~showErrorWindow("WARNING","Please select an item!","WARNING")
END
-------------------------------------------------------
::ROUTINE handleLunchMenu PUBLIC
 .my.app~stageHandler~newWindow("Lunch - Menu","file:lunchMenu.fxml")
-------------------------------------------------------
::ROUTINE handleLogin PUBLIC
 .my.app~stageHandler~newWindow("Login A - Trust","file:se_login.fxml")
-------------------------------------------------------
::ROUTINE handlePay PUBLIC
/*@get(rsTableViewLeft rsTableViewRight)*/
 CALL payProcess rsTableViewLeft, rsTableViewRight, .false
-------------------------------------------------------
::ROUTINE handleSplit PUBLIC
/*@get(rsTableViewLeft rsTableViewRight)*/
 CALL payProcess rsTableViewLeft, rsTableViewRight, .true
-------------------------------------------------------
::ROUTINE handleLogout PUBLIC
 .my.app~DatabaseHandler~disconnectDB
 CALL logOut
-------------------------------------------------------
::ROUTINE handleBusiness PUBLIC
 .my.app~stageHandler~newWindow("Business information", "file:businessInfo.fxml")
-------------------------------------------------------
::ROUTINE handleDEPExp PUBLIC
 CALL exportDEP
-------------------------------------------------------
::ROUTINE handleNewUser PUBLIC
 .my.app~stageHandler~newWindow("New User", "file:newUser.fxml")
-------------------------------------------------------
::ROUTINE handleEditUser PUBLIC
 .my.app~stageHandler~newWindow("Edit User", "file:user.fxml")
-------------------------------------------------------
::ROUTINE handleChangePassword PUBLIC
 .my.app~stageHandler~newWindow("New Product", "file:passwordChange.fxml")
-------------------------------------------------------
::ROUTINE handleNewProduct PUBLIC
 .my.app~prodID = 0
 .my.app~deleteProduct = .false
 .my.app~stageHandler~newWindow("New Product", "file:product.fxml")
-------------------------------------------------------
::ROUTINE handleEditProduct PUBLIC
```

```
 .my.app~EditProduct = .true
 .my.app~deleteProduct = .true
 .my.app~stageHandler~newWindow("Edit Product", "file:product.fxml")
------------------------------------------------------------
::ROUTINE handleShowTurnOver PUBLIC
 .my.app~stageHandler~newWindow("Turnover overview","file:turnover.fxml")
------------------------------------------------------------
::ROUTINE handleEditLunch PUBLIC
 .my.app~stageHandler~newWindow("Edit Lunch Menu", "file:editLunchMenu.fxml")
------------------------------------------------------------
::ROUTINE handleQuit PUBLIC
 bsf.loadClass("javafx.application.Platform")~exit
------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "DatabaseHandler_v2.CLS"
::REQUIRES "TableH.CLS"
::REQUIRES "collection.rexx"
```

Auflistung 17: rootLayoutController.rexx

Die Datei collection.rexx auf Seite 93 dient als Sammelsurium vieler unterschiedlicher Methoden, die im Laufe des Programmablaufs benutzt werden.

```
/*
    This file is some sort of library. The most used ::ROUTINES can be found here.
*/
::ROUTINE formatToEuro PUBLIC
 USE ARG turnoverValue
 turnoverValue = "EUR: " ||  format2float(turnoverValue)
 RETURN turnoverValue
------------------------------------------------------------
::ROUTINE reformatEUR PUBLIC
 USE ARG turnoverValue
 PARSE VAR turnoverValue "EUR: " value
 RETURN value
------------------------------------------------------------
::ROUTINE addNewOrderDetails PUBLIC
 USE ARG newOrderDetails
 .my.app~newOrderDetailsHandler~addItem(newOrderDetails)
------------------------------------------------------------
::ROUTINE findItemNewOrderDetails PUBLIC
 USE ARG newOrderDetails
DO item OVER .my.app~newOrderDetailsHandler~newOrderOrderTableView~getItems
    IF item~PID~get == newOrderDetails~PID~get THEN DO
        RETURN item
    END
END
 RETURN .nil
------------------------------------------------------------
::ROUTINE incrementQuantityUntilQuant PUBLIC
 USE ARG item, quantity
 quantity = quantity -1
 SimpleIntegerProperty = bsf.import("javafx.beans.property.SimpleIntegerProperty")
 sum = item~quantity~getValue
 quantity = box("INTeger", quantity)
 tempSum = box("INTeger", sum)
IF tempSum~toString <= quantity~toString THEN DO
    sum = item~quantity~add(SimpleIntegerProperty~new(1))
    item~quantity~set(sum~getValue)
    RETURN .true
END
ELSE
    RETURN .false
------------------------------------------------------------
::ROUTINE updateTurnoverValue PUBLIC
 USE ARG  updateValue,typeOfReceipt, companyID = 1
 oldTurnoverValue = .my.app~DatabaseHandler~getTurnoverValue(companyID)
IF typeOfReceipt = "Normal" THEN DO
    newValue = oldTurnoverValue + updateValue
    .my.app~DatabaseHandler~updateTurnoverValue(companyID, newValue)
END
IF typeOfReceipt = "Storno" THEN DO
    newValue = oldTurnoverValue - updateValue
    .my.app~DatabaseHandler~updateTurnoverValue(companyID, newValue)
END
------------------------------------------------------------
::ROUTINE encrypteTurnoverTraSto PUBLIC
 USE ARG TraOrSto
 Base64 = bsf.import("java.util.Base64")
 Encoder = Base64~getEncoder
```

```
 turnoverValue = TraOrSto
 turnoverValueString = box("STring", turnoverValue)
 encryptedTurnOverValueInB64 = turnoverValueString~getBytes
 encryptedTurnOverValueInB64= Encoder~encodeToString(encryptedTurnOverValueInB64)
 RETURN encryptedTurnOverValueInB64
------------------------------------------------------------
::ROUTINE print PUBLIC
 USE ARG string
 jEditorPane = .bsf~new("javax.swing.JEditorPane")
 htmlEditorKit = .bsf~new("javax.swing.text.html.HTMLEditorKit")
 jEditorPane~setEditorKit(htmlEditorKit)
 styleSheet = htmlEditorKit~getStyleSheet
 styleSheet~addRule("body {font-size: 7px;}")
 styleSheet~addRule("h1 {font-size: 11px;}")
 styleSheet~addRule("h2 {font-size: 10px;}")
 defaultDocument = htmlEditorKit~createDefaultDocument
 jEditorPane~setDocument(defaultDocument)
 jEditorPane~setText(string)
 RETURN jEditorPane~print
------------------------------------------------------------
::ROUTINE payProcess PUBLIC
 USE ARG  rsTableViewLeft, rsTableViewRight, splitPay
 companyDetails = .my.app~DatabaseHandler~getCompanyData(1)
 AESHandler = .AESHandler~new

IF .my.app~SEHandler = .nil & .my.app~SEHandlerOffline = .nil THEN DO
    .my.app~stageHandler~showErrorWindow("Security Device Failure", "Please login or choose
         offline signing","ERROR")
    .my.app~stageHandler~newWindow("Please login or choose offline","file:se_login.fxml")
END
ELSE IF .my.app~SEHandler <> .nil THEN DO
    seLoginTime = .my.app~DatabaseHandler~getSELoginTime
    seIsOut = .my.app~SEHandler~seIsOut
    IF seIsOut THEN DO
        .my.app~stageHandler~showErrorWindow("Remote signing does not work","Connection to
             security device failed. Check your internet connection or choose offline signing
             ","ERROR")
        .my.app~stageHandler~newWindow("Please login","file:se_login.fxml")
    END
    ELSE IF time < seLoginTime | seLoginTime = .nil  THEN DO
        .my.app~stageHandler~newWindow("Timer run out. Please login again","file:se_login.
             fxml")
    END
    zertSerienNr = .my.app~SEHandler~zertSerienNr_value
END
ELSE IF .my.app~SEHandlerOffline <> .nil THEN DO
    zertSerienNr = .my.app~SEHandlerOffline~getCertNr
END

IF .my.app~offPay = .true | .my.app~remotePay = .true THEN DO
    IF .my.app~offPay = .true THEN DO
        zda = .my.app~SEHandlerOffline~getZDA
    END
    IF .my.app~remotePay = .true THEN DO
        zda = .my.app~SEHandler~getZDAData
     END


    TypeOfReceipt = .my.app~DatabaseHandler~getTypeOfReceipt(.my.app~currentOrderID)

    TableForTax = .table~new
    TableForTax["0.20"] = .array~new
    TableForTax["0.19"] = .array~new
    TableForTax["0.13"] = .array~new
    TableForTax["0.10"] = .array~new
    TableForTax["0.00"] = .array~new

    IF splitPay = .true THEN DO
        .my.app~DatabaseHandler~insertOrder(.my.app~currentTableNR, TypeOfReceipt)
        tempID = .my.app~DatabaseHandler~getLastOrderID

    END
    ELSE DO
        tempID = .my.app~currentOrderID
    END

    productNames = .array~new
    productPrices = .array~new
    productQuantities = .array~new
    DO item over rsTableViewRight~getItems
        IF item~status~get = .my.app~OrderDetailsHandler~statusArray[2] & splitPay  THEN DO
            .my.app~DatabaseHandler~getAllProductsArrayListByID(tempID, .my.app~
                 orderDetailsHandler)
```

```
136             IF productNames~hasItem(item~name~get) THEN DO
137                 productQuantities[productNames~index(item~name~get)] = productQuantities[
                        productNames~index(item~name~get)] + 1
138             END
139             ELSE DO
140                 productNames        ~append(item~name~get)
141                 productPrices       ~append(item~price~get)
142                 productQuantities   ~append(1)
143             END
144             taxRate = .my.app~DatabaseHandler~getTaxRateByProdID(item~PID~get)
145             SELECT
146                 WHEN taxRate = 0.20 THEN TableForTax["0.20"]~append(item~price~get)
147                 WHEN taxRate = 0.19 THEN TableForTax["0.19"]~append(item~price~get)
148                 WHEN taxRate = 0.13 THEN TableForTax["0.13"]~append(item~price~get)
149                 WHEN taxRate = 0.00 THEN TableForTax["0.00"]~append(item~price~get)
150                 WHEN taxRate = 0.10 THEN TableForTax["0.10"]~append(item~price~get)
151             END
152             .my.app~DatabaseHandler~updateContainsStatus(.my.app~OrderDetailsHandler~
                    statusArray[3],item~containsID)
153         END
154
155         IF splitPay <> .true THEN DO    -- FullPay
156             IF productNames~hasItem(item~name~get) THEN DO
157                 productQuantities[productNames~index(item~name~get)] = productQuantities[
                        productNames~index(item~name~get)] + 1
158             END
159             ELSE DO
160                 productNames        ~append(item~name~get)
161                 productPrices       ~append(item~price~get)
162                 productQuantities   ~append(1)
163             END
164             rsTableViewRight~getItems~remove(item)
165             .my.app~DatabaseHandler~updateContainsStatus(.my.app~OrderDetailsHandler~
                    statusArray[3],item~containsID)
166             taxRate = .my.app~DatabaseHandler~getTaxRateByProdID(item~PID~get)
167             SELECT
168                 WHEN taxRate = 0.20 THEN TableForTax["0.20"]~append(item~price~get)
169                 WHEN taxRate = 0.19 THEN TableForTax["0.19"]~append(item~price~get)
170                 WHEN taxRate = 0.13 THEN TableForTax["0.13"]~append(item~price~get)
171                 WHEN taxRate = 0.00 THEN TableForTax["0.00"]~append(item~price~get)
172                 WHEN taxRate = 0.10 THEN TableForTax["0.10"]~append(item~price~get)
173             END
174         END
175     END
176
177     .my.app~orderHandler~orderArrayList=.my.app~DatabaseHandler~getAllOpenOrders
178     .my.app~orderHandler~setItems
179     rsTableViewRight~getItems~clear
180     sumTablePerTaxRate = .table~new
181     formattedSumTablePerTaxRate = .table~new
182     sum = 0
183     DO index OVER TableForTax
184         sumPerTaxRate = sum(TableForTax[index])
185         sumTablePerTaxRate[index] = sumPerTaxRate
186         formattedSumTablePerTaxRate[index] = reformatFloat(sumPerTaxRate)
187         sum += sumPerTaxRate
188     END
189     totalSum = 0
190     DO i = 1 TO productPrices~items
191         totalSum = totalSum + (productPrices[i]*productQuantities[i])
192     END
193
194
195     date = .my.app~DatabaseHandler~getDatePayment(tempID)
196
197     PARSE VAR date year " " time
198     dateWithT = year || "T" || time
199     previousJWScompact = .my.app~databaseHandler~getLastJWSData
200
201     IF previousJWScompact = .nil THEN DO
202         sigVorigerBeleg = AESHandler~hashLastReceipt(companyDetails["cashbox_ID"])
203     END
204     ELSE DO
205         sigVorigerBeleg = AESHandler~hashLastReceipt(previousJWScompact)
206     END
207
208     taxOrderForSignature = .array~of("0.20", "0.10", "0.13", "0.00", "0.19")
209     sigData = .mutableBuffer~new
210     sigData~append("_R1-" || zda || "_" || companyDetails["cashbox_ID"] || "_" || tempID ||
            "_"|| dateWithT )
211
212     IF TypeOfReceipt = "Normal" THEN DO
213         CALL updateTurnoverValue totalSum, TypeOfReceipt
```

```
214         turnoverValue = .my.app~DatabaseHandler~getTurnoverValue(companyDetails["company_ID"
                ])
215         turnoverValue = format((turnoverValue*100),,0)
216         encryptedTurnOverValueInB64 = AESHandler~encryptAES(turnoverValue, companyDetails["
                aes256key"], companyDetails["cashbox_ID"], tempID)
217         DO index OVER taxOrderForSignature
218             sigData~append("_" || formattedSumTablePerTaxRate[index])
219         END
220         sigData~append("_" || encryptedTurnOverValueInB64 || "_" || zertSerienNr || "_" ||
                sigVorigerBeleg)
221     END
222
223     IF TypeOfReceipt = "Null" THEN DO
224         turnoverValue =.my.app~DatabaseHandler~getTurnoverValue(companyDetails["company_ID"
                ])
225         turnoverValue = format((turnoverValue*100),,0)
226         encryptedTurnOverValueInB64 = AESHandler~encryptAES(turnoverValue, companyDetails["
                aes256key"], companyDetails["cashbox_ID"], tempID)
227         DO index OVER taxOrderForSignature
228             sigData~append("_" || "0,00")
229         END
230         sigData~append("_" || encryptedTurnOverValueInB64 || "_" || zertSerienNr || "_" ||
                sigVorigerBeleg)
231     END
232
233     IF TypeOfReceipt = "Start" THEN DO
234         turnoverValue =.my.app~DatabaseHandler~getTurnoverValue(companyDetails["company_ID"
                ])
235         encryptedTurnOverValueInB64 = AESHandler~encryptAES(turnoverValue, companyDetails["
                aes256key"], companyDetails["cashbox_ID"], tempID)
236         DO index OVER taxOrderForSignature
237             sigData~append("_" || "0,00")
238         END
239         sigData~append("_" || encryptedTurnOverValueInB64 || "_" || zertSerienNr || "_" ||
                sigVorigerBeleg)
240     END
241
242     IF TypeOfReceipt = "Training" THEN DO
243         encryptedTurnOverValueInB64 = encrypteTurnoverTraSto("TRA")
244         DO index OVER taxOrderForSignature
245             sigData~append("_" || formattedSumTablePerTaxRate[index])
246         END
247         sigData~append("_" || encryptedTurnOverValueInB64 || "_" || zertSerienNr || "_" ||
                sigVorigerBeleg)
248     END
249
250     IF TypeOfReceipt = "Storno" THEN DO
251         encryptedTurnOverValueInB64 = encrypteTurnoverTraSto("STO")
252         CALL updateTurnoverValue totalSum, TypeOfReceipt
253         DO index OVER taxOrderForSignature
254             sigData~append("_" || formattedSumTablePerTaxRate[index])
255         END
256         sigData~append("_" || encryptedTurnOverValueInB64 || "_" || zertSerienNr || "_" ||
                sigVorigerBeleg)
257     END
258
259     IF .my.app~offPay =.true THEN DO
260         newJWSforReceiptAndDB = .my.app~SEHandlerOffline~signString(sigData)
261         sigZert_value = .my.app~SEHandlerOffline~zertValue
262     END
263     IF .my.app~remotePay =.true THEN DO
264         newJWSforReceiptAndDB = .my.app~SEHandler~getJWS(.my.app~SEHandler~alg_value,
                sigData)
265         sigZert_value =  .my.app~SEHandler~sigZert_value
266     END
267     PARSE VAR newJWSforReceiptAndDB newJWSHeader newJWSPayload newJWSSignature
268     .my.app~DatabaseHandler~updateOrderJWS(tempID,newJWSHeader,newJWSPayload,newJWSSignature
            )
269     qrData = .mutableBuffer~new
270     QRCodeHandler = .QRCodeHandler~new
271     qrData = sigData
272     qrData~append("_" || encodeB64URLtoB64(newJWSSignature))
273     QRCodeHandler~makeQRCode(qrData)
274     CALL saveDEPExport sigZert_value, date, newJWSHeader, newJWSPayload, newJWSSignature
275     .my.app~stageHandler~backToRoot
276     CALL printingReceipt companyDetails, productNames, productPrices, productQuantities,
            totalSum, sumTablePerTaxRate, tempID, .my.app~currentTableNR, date
277 END
278 ------------------------------------------------------------
279 ::CLASS ActiveUser PUBLIC
280 ::ATTRIBUTE userID
281 ::ATTRIBUTE firstname
282 ::ATTRIBUTE lastname
```

```
::ATTRIBUTE rightsInd
::METHOD init
 EXPOSE userID firstname lastname rightsInd
 USE ARG userID, firstname, lastname, rightsInd
------------------------------------------------------------
::CLASS ReceiptDEP PUBLIC
::METHOD init
 EXPOSE certificate jws_header jws_payload jws_signature
 USE ARG certificate, jws_header, jws_payload, jws_signature

 certificate = .fx.SimpleStringProperty~new(certificate)
 jws_header = .fx.SimpleStringProperty~new(jws_header)
 jws_payload = .fx.SimpleStringProperty~new(jws_payload)
 jws_signature = .fx.SimpleStringProperty~new(jws_signature)

::ATTRIBUTE certificate GET
 EXPOSE certificate
 RETURN certificate~get

::ATTRIBUTE certificate SET
 EXPOSE certificate
 USE ARG val
 RETURN certificate~set(val)

::ATTRIBUTE certificateProperty GET
 EXPOSE certificate
 RETURN certificate

::ATTRIBUTE jws_header GET
 EXPOSE jws_header
 RETURN jws_header~get

::ATTRIBUTE jws_header SET
 EXPOSE jws_header
 USE ARG val
 RETURN jws_header~set(val)

::ATTRIBUTE jws_headerProperty GET
 EXPOSE jws_header
 RETURN jws_header

::ATTRIBUTE jws_payload GET
 EXPOSE jws_payload
 RETURN jws_payload~get

::ATTRIBUTE jws_payload SET
 EXPOSE jws_payload
 USE ARG val
 RETURN jws_payload~set(val)

::ATTRIBUTE jws_payloadProperty GET
 EXPOSE jws_payload
 RETURN jws_payload

::ATTRIBUTE jws_signature GET
 EXPOSE jws_signature
 RETURN jws_signature~get

::ATTRIBUTE jws_signature SET
 EXPOSE jws_signature
 USE ARG val
 RETURN jws_signature~set(val)

::ATTRIBUTE jws_signatureProperty GET
 EXPOSE jws_signature
 RETURN jws_signature
------------------------------------------------------------
::ROUTINE sum
 USE ARG array
 sum = 0
 DO i OVER array
   sum += i
 END
 RETURN sum
------------------------------------------------------------
::ROUTINE exportDEP PUBLIC
 receiptArrayList = .my.app~databasehandler~getDEP
 item = receiptArrayList~get(1)
 receipt = BSFRexxProxy(item)
 certificate_one =  receipt~certificate
 say "certificate" certificate

 FXCollections = bsf.import("javafx.collections.FXCollections")
```

```
 FileWriter = bsf.import("java.io.FileWriter")

 value1 =
"MIIEATCCAumgAwIBAgIEOWntwTANBgkqhkiG9w0BAQUFADCBlTELM
AkGA1UEBhMCQVQxSDBGBgNVBAoMP0EtVHJ1c3QgR2VzLiBmLiBTaWN
oZXJoZWl0c3N5c3RlbWUgaW0gZWxla3RyLiBEYXRlbnZlcmtlaHIgR
2liSDEdMBsGA1UECwwUQS1UcnVzdC1UZXN0LVF1YWwtMDIxHTAbBgN
VBAMMFEEtVHJ1c3QtVGVzdC1RdWFsLTAyMB4XDTE0MTEyNDE0NDkxN
loXDTI0MTExODEzNDkxN1owgaExCzAJBgNVBAYTAkFUMUgwRgYDVQQ
KDD9BLVRydXN0IEdlcy4gZi4gU2ljaGVyaGVpdHNzeXN0ZW1lIGltI
GVsZWt0ci4gRGF0ZW52ZXJrZWhyIEdtYkgxIzAhBgNVBAsMGmEtc2l
nbi1QcmVtaXVtLVRlc3QtU2lnLTAyMSMwIQYDVQQDDBphLXNpZ24tU
HJlbWl1bS1UZXN0LVNpZy0wMjCCASIwDQYJKoZIhvcNAQEBBQADggE
PADCCAQoCggEBANwJSfWpRaziThddTTup72CltlXl8oc7HQoK2SWsY
QwZGAd5nJZbwbI4K8VFKlNnK72Zl8UhmQ2FxhzS6u+Q+qEzJOM2xTf
A2NB6A9/KFpTJXUjvCHgRvW16EEF9YpYXxKTSK+QrYCXAC5rL6SuYO
zgA7Qlivq+zLbyXxroux2zVEBIiaBGpZhOHGDFJk6h/4QelIqzd2TI
DCRzvhmLDVmhqX2C1NQb5kZuMgrxxOhG5Cr1F8so1kwyu43JiM+apY
4bZJVQBwi9ATBMz5tfdoLRslQy1BCQ4X+b6u/2856gucU+1e/wa5pB
9Ff0eP+xy+j2DZOXLNd8m/IQvnshjNusCAwEAAaNLMEkwDwYDVR0TA
QH/BAUwAwEB/zARBgNVHQ4ECgQIRgafjkGOFb0wEwYDVR0jBAwwCoA
IQg8xWXA9iecwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb3DQEBBQUAA
4IBAQBq/owq5eGvhxegchLvnMjPnE9gTYIHEvMq8DN6h2J7pTEhKG2
o09LLn/pNHWRjKENU/LqZBIAJ5zebm5XqzB631BYcuu1abyPFfpMdA
L9X4zFuDvg9EGaTir2c81XaBYzVSLN7fxmNLKSmMwUt0JQQyqpe3V9
iyoBE/WcQyKmKaEp7mCzsGFBm6KmJgqD6TPb7X9bWUr3yx6Z5gek71
f3vQi69m1x811azXlxu1i/XFnVpzxkrKRXJWC+wnQRxXmU7YnMzYPO
A7UOpUG6J+7tYi29hY3EpMgyXM/T/BL5MdyzBefbPVzLHng5zVaXNp
O0ENCrlUyi1m3Yd/7QPDdJM"
 value2 =
"MIID4DCCAsigAwIBAgIEOWntvzANBgkqhkiG9w0BAQUFADCBlTELM
AkGA1UEBhMCQVQxSDBGBgNVBAoMP0EtVHJ1c3QgR2VzLiBmLiBTaWN
oZXJoZWl0c3N5c3RlbWUgaW0gZWxla3RyLiBEYXRlbnZlcmtlaHIgR
2liSDEdMBsGA1UECwwUQS1UcnVzdC1UZXN0LVF1YWwtMDIxHTAbBgN
VBAMMFEEtVHJ1c3QtVGVzdC1RdWFsLTAyMB4XDTE0MTEyNDE0NDc0N
loXDTI0MTExODEzNDc0N1owgZUxCzAJBgNVBAYTAkFUMUgwRgYDVQQ
KDD9BLVRydXN0IEdlcy4gZi4gU2ljaGVyaGVpdHNzeXN0ZW1lIGltI
GVsZWt0ci4gRGF0ZW52ZXJrZWhyIEdtYkgxHTAbBgNVBAsMFEEtVHJ
1c3QtVGVzdC1RdWFsLTAyMR0wGwYDVQQDDBRBLVRydXN0LVRlc3QtU
XVhbC0wMjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANM
Bok2fNNtIEcf7Sw47vprkUeti6Y64Rc5rrAjh7cGwo4Jp5LyfvEVdv
9AMNiuOX7ywd1xW99UZWtZ8MzXvWM5M6trLkeBYnCukwc9DqawXcuX
XCYwgTuisFTmYO6GVJNr1iE/LJdSKbu5AVDS3FwXixqyJkjv/xWIwU
4q86oATW8++8wb6Lu+fQlhBbn3Kqpavt6K+lwWSCb+8vIhB47IlKhJ
ZwGqXfGV9l9dDgKYUbZiv3BBa+MRBUTvIcahEKz8hG2E8W4EgCwzIS
MpeStJtRHo/tJnA90KfSBTcz0txrxpHwqFgKwJvgW6nIjY1Sv5MfY5
YJiEWv0d7UUkvlScCAwEAAaM2MDQwDwYDVR0TAQH/BAUwAwEB/zARB
gNVHQ4ECgQIQg8xWXA9iecwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIb
3DQEBBQUAA4IBAQApqSvkQyfbO2yDWewHwo1Zl32uGz41KMP5FYtA3
BIcqh89paHwrW9KfcrybdUIneVz4iSnpyrDrS4LavfP8h/Hl1kRmVZ
RUBsOJRvqc1fiC2B6IJRHrmayb/DbXuyoOsk7Sr8M9xtAD3SzJCRkB
rtjz/U/xQdU9TfV9SQyPN3qI+SR25/LRZDhOKcIFJduVpTYzbnKTIk
l3OUrHXVq5xddxX6XP8bUjT+SqGiDf15H6N5flNBsvolMSo0OoQXFi
DuY33frQSrSbHbA2p/MptwxA8JgGh4lrbgZZxjTvpO1wATBLDc3wGZ
kNuy+tNrrHAmE08B7fiExULHxzfaZEWSF"

 certificationAuthorityArrayList = FXCollections~observableArrayList
 certificationAuthorityArrayList~add(value1)
 certificationAuthorityArrayList~add(value2)

 receiptCompactOne = FXCollections~observableArrayList
 receiptCompactTwo = FXCollections~observableArrayList
 certificate = FXCollections~observableArrayList

DO p OVER receiptArrayList
    IF p~certificate = certificate_one THEN DO
        receiptCompactOne~add(p~jws_header || "." || p~jws_payload || "." || p~jws_signature
            )
    END
    ELSE DO
        certificate_two = p~certificate
        receiptCompactTwo~add(p~jws_header || "." || p~jws_payload || "." || p~jws_signature
            )
    END

END

 receiptGroup = .array~of("Belege-Gruppe")
 header=.array~of("Signaturzertifikat","Zertifizierungsstellen","Belege-Kompakt")

IF receiptCompactTwo~size <> 0 THEN DO
 content1= .array~of(certificate_one, certificationAuthorityArrayList , receiptCompactOne)
 content2= .array~of(certificate_two, certificationAuthorityArrayList , receiptCompactTwo)
```

```
 object1 = Order(header,content1)
 object2 = Order(header,content2)
 group = FXCollections~observableArrayList
 group~add(objet1)
 group~add(objet2)
 groupArr = .array~of(group)
 receiptGroupObject = Order(receiptGroup,groupArr)
END
ELSE DO
 content=.array~of(certificate_one, certificationAuthorityArrayList , receiptCompactOne)
 object = Order(header,content)
 group = FXCollections~observableArrayList
 group~add(object)
 groupArr = .array~of(group)
 receiptGroupObject = Order(receiptGroup,groupArr)
END
 directoryExists = 0

CALL SysFileTree .my.app~homeDir, "file" , "D"  --

DO i = 1 to file.0
    IF file.i~pos("JSON") <> 0 THEN DO
        directoryExists += 1
    END
    ELSE DO
        directoryExists += 0
    END
END

IF directoryExists = 0 THEN DO
    CALL  SysMkDir "JSON"
END
 JSONpath = .my.app~homeDir || "JSON"

 CALL directory  JSONpath

 file = FileWriter~new("DEPExport.json")
 file~write(receiptGroupObject~toJSONString)
 file~flush
 .my.app~stageHandler~showErrorWindow("DEP-Export", "DEP-Export was successful!", "
     INFORMATION")
 CALL directory .my.app~homeDir

 EXIT 0
-------------------------------------------------------------
::ROUTINE Order PUBLIC
 USE ARG header, content
 JSONObject = bsf.import("org.json.simple.JSONObject")
 sortedObject = JSONObject~new

DO i = 1 to header~size
    sortedObject~put(header[i], content[i])
END

 RETURN sortedObject
-------------------------------------------------------------
::ROUTINE saveDEPExport
 USE ARG sigZert_value, date, newJWSHeader, newJWSPayload, newJWSSignature
 .my.app~databasehandler~insertDEPReceipt(sigZert_value, date, newJWSHeader, newJWSPayload,
     newJWSSignature)
-------------------------------------------------------------
::ROUTINE reformatFloat PUBLIC
 USE ARG float
 RETURN translate(format2float(float), ",", ".")
-------------------------------------------------------------
::ROUTINE formatStringToFloat PUBLIC
 USE ARG string
 float = string~changeStr(",", ".")
 SIGNAL ON ANY NAME failedInput
 returnV = format(float,,2)
 SIGNAL OFF ANY
 RETURN returnV
 EXIT 0
 failedInput:
     RETURN -1
-------------------------------------------------------------
::ROUTINE format2float PUBLIC
 USE ARG float
 RETURN format(float,,2)
-------------------------------------------------------------
::ROUTINE checkIFPureText PUBLIC
 USE ARG text
 SIGNAL ON ANY NAME notPure
```

```
 box("STring", text)
 SIGNAL OFF ANY
 RETURN 1
 EXIT 0
  notPure:
      RETURN -1
------------------------------------------------------------
::ROUTINE appendTaxText
 USE ARG prevReceipt, sumTablePerTaxRate
 nl="0d0a"x   -- CRLF characters
DO index OVER sumTablePerTaxRate
    IF index = "0.00" THEN DO
        tax = sumTablePerTaxRate[index]
        net = sumTablePerTaxRate[index]
    END
    ELSE DO
        net = format2float(sumTablePerTaxRate[index]/(index+1))
        tax = format2float(sumTablePerTaxRate[index] - net)
    END
    IF sumTablePerTaxRate[index] > 0 THEN DO
        prevReceipt~~append(nl)~~ append("<tr><td>" || format(index*100,,0) || "%"|| "</td><
            td>" || net || "</td><td>" || tax || "</td><td>" || format(sumTablePerTaxRate[
            index],,2) || "</td></tr>")
    END
END
 RETURN prevReceipt
------------------------------------------------------------
::ROUTINE encodeB64URLtoB64
 USE ARG B64URL
 Base64 = bsf.import("java.util.Base64")
 URLdecoder = Base64~getUrlDecoder
 Encoder = Base64~getEncoder
 normalString = URLdecoder~decode(B64URL)
 RETURN Encoder~encodeToString(normalString)
------------------------------------------------------------
::ROUTINE printingReceipt
 USE ARG companyDetails, productNames, productPrices, productQuantities, totalSum,
      sumTablePerTaxRate, orderID, tableNr, dateFull

 PARSE VAR dateFull date " " time
 IF tableNr = "Start" | tableNr = "Storno" | tableNr = "Null"  THEN DO
    temp = tableNr
 END
 ELSE  DO
    temp = "TableNr.: " || tableNr
 END
 currPrinter=bsf.importClass("javafx.print.Printer")~getDefaultPrinter
 say "Printer selected: " pp(currPrinter~toString)

 mb=.mutableBuffer~new
 nl="0d0a"x   -- CRLF characters
 mb~~append("<!DOCTYPE html>")
 mb ~~append(nl) ~~append("<html>" -
                          "<center id="top">"-
                          "<div class="logo"></div>"-
                          "<div class="info">"-
                          "<h2>"|| companyDetails["name"] || "</h2>"-
                          "</div>")
 mb ~~append(nl) ~~append("<div id="mid">"-
                          "<div class="info">"-
                          "<p>"|| companyDetails["address"] || "<br>"-
                            companyDetails["telephone"]||"<br>" -
                            companyDetails["e_mail"] ||"<br>"-
                            companyDetails["vat_ID"] ||"<br>"-
                            "OrderID: "|| orderID ||"<br>"-
                                     temp ||"<br>"-
                                     date ||"<br>"-
                                     time ||"<br>"-
                          "</p>"-
                          "</div>"-
                          "</div>")
 mb ~~append(nl) ~~append("<table style='width: 150px; height: 100px;'>"-
                          "<tbody>"-
                          "<tr class="tabletitle">"-
                          "<td class="item">"-
                          "<h1>Item</h1>"-
                          "</td>"-
                          "<td class="Qty">"-
                          "<h1>Qty</h1>"-
                          "</td>"-
                          "<td class="Sum">"-
                          "<h1>Price</h1>"-
                          "</td>"-
```

```
                          "<td class="Total">"-
                          "<h1>Total</h1>"-
                          "</td>"-
                          "</tr>")
DO i=1 TO productNames~items
 mb~~append(nl) ~~append("<tr class="Products">"-
                          "<td class="tableitem">"-
                          "<p class="itemtext">"|| productNames[i] || "</p>"-
                          "</td>"-
                          "<td class="tableitem">"-
                          "<p class="itemtext">"|| productQuantities[i] || "</p>"-
                          "</td>"-
                          "<td class="tableitem">"-
                          "<p class="itemtext">"|| "&euro; "|| productPrices[i] || "</p>"-
                          "</td>"-
                          "<td class="tableitem">"-
                          "<p class="itemtext">"|| "&euro; "|| productPrices[i]*
                               productQuantities[i] || "</p>"-
                          "</td>"-
                          "</tr>")

END
 mb~~append(nl) ~~append("</tbody>"-
                          "</table>"-
                          "</div>")
 mb~~append(nl) ~~append("<div>==========================================</div>")
 mb~~append(nl) ~~append("<h2> Total amount: " || "&euro; "||  totalSum || "</h2>") --<div
      ><p>Total amount: 14.8</p> </div>
 mb~~append(nl) ~~append("<div>==========================================</div>")
 mb~~append(nl) ~~append("<tr>" -
                          "<th>MWST</th>"-
                           "<th>Net</th>"-
                          "<th>Tax</th>"-
                          "<th>Gross</th>"-
                          "</tr>")
 CALL appendTaxText mb, sumTablePerTaxRate
 filePath = "file:" || .my.app~homeDir || "/QR/qr.png"
 mb~~append(nl) ~~append("</table><br>"-
                          '<img src="' || filePath || '" width="100" height="100"><br>' -
                          .my.app~ActiveUser~firstname || " served you!" || -
                          "<br>" -
                          "Thank you for your purchase. <br> We look forward to seeing you
                               again soon. <br>"-
                          "</span></body></html>"-
                          "</div>"-
                          "</div>")
 mb ~~append(nl) ~~append("</body>")
 mb ~~append(nl) ~~append("</html>")

 done = print(mb~string)
IF done THEN DO
    .my.app~stageHandler~showErrorWindow("Printing job successful!" , "Printing job
         successful!", "INFORMATION" )
END
ELSE DO
    .my.app~stageHandler~showErrorWindow("Printing job NOT successful!" , "Printing job
         FAILED", "ERROR" )
END

------------------------------------------------------------
::ROUTINE firstGUI PUBLIC
 checkIFPOSAdminExists = .my.app~DatabaseHandler~checkIFPOSAdminExists
 checkIFCompanyDataExists = .my.app~DatabaseHandler~checkIFCompanyDataExists
IF \checkIFPOSAdminExists THEN DO
    .my.app~stageHandler~newWindow("Please add an Admin-User", "file:admin.fxml")
END
ELSE IF \checkIFCompanyDataExists THEN DO
    .my.app~stageHandler~newWindow("Please add some Business information", "
         file:businessInfo.fxml")
END
ELSE DO
    .my.app~stageHandler~newWindow("Login Terminal", "file:login.fxml")

END
------------------------------------------------------------
::ROUTINE logOut PUBLIC
 .my.app~stageHandler~loadScene("Start", "file:start.fxml")
------------------------------------------------------------
::ROUTINE generateRandomJWSKeyPrivate PUBLIC
 KeyPairGenerator = bsf.loadClass("java.security.KeyPairGenerator")
 SecureRandom = bsf.import("java.security.SecureRandom")
 Base64 = bsf.import("java.util.Base64")
 Encoder = Base64~getEncoder
```

```
 keyGenEC = KeyPairGenerator~getInstance("EC")
 randomValue = SecureRandom~getInstance("SHA1PRNG")
 keyGenEC~initialize(256, randomValue)
 pair = keyGenEC~generateKeyPair
 private = pair~getPrivate~getEncoded
 RETURN Encoder~encodeToString(private)
------------------------------------------------------------
::ROUTINE generateRandomAESKey PUBLIC
 KeyGenerator = bsf.import("javax.crypto.KeyGenerator")
 Base64 = bsf.import("java.util.Base64")
 keyGenAES = KeyGenerator~getInstance("AES")
 keyGenAES~bsf.invoke("init", 256)
 secretKey = keyGenAES~generateKey
 encodedSecKey = secretKey~getEncoded
 Encoder = Base64~getEncoder
 base64Key = Encoder~encodeToString(encodedSecKey)
 RETURN base64Key
------------------------------------------------------------
::ROUTINE saveToFile PUBLIC
 USE ARG directory
 directoryExists = 0
 CALL SysFileTree .my.app~homeDir, "file" , "D"

DO i = 1 to file.0
    IF file.i~pos(directory) <> 0 THEN DO
        directoryExists += 1
    END
    ELSE DO
        directoryExists += 0
    END
END

IF directoryExists = 0 THEN DO
    CALL  SysMkDir directory
END
 path = .my.app~homeDir || directory

CALL directory  path

 fileName = directory || ".txt"
 IF directory = "AES" THEN DO
    randomSecreteKey = generateRandomAESKey()
 END
 IF directory = "JWS" THEN DO
    randomSecreteKey = generateRandomJWSKeyPrivate()
 END
 ELSE DO
    say "Directory not required"
 END

IF \SysIsFile(fileName) THEN DO
    Data = .stream~new(fileName)
    Data~open("write replace")
    Data~lineout('[' || directory || 'key data for later use]')
    Data~lineout(directory ||"Key: " || randomSecreteKey )
    Data~close
END

CALL directory .my.app~homeDir

say pp("A file named " || fileName ||" has been generated and stored. This file contains an"
     || directory||"-Key," -
"which can be used in the POS-System")
say pp("The key is stored at this path") pp(path)

RETURN randomSecreteKey
------------------------------------------------------------
::ROUTINE mkDir PUBLIC
 USE ARG directory
 directoryExists = 0
 CALL SysFileTree .my.app~homeDir, "file" , "D"

DO i = 1 to file.0
    IF file.i~pos(directory) <> 0 THEN DO
        directoryExists += 1
    END
    ELSE DO
        directoryExists += 0
    END
END

IF directoryExists = 0 THEN DO
    CALL  SysMkDir directory
```

```
767 END
768
769 ------------------------------------------------------
770 ::REQUIRES "BSF.CLS"
771 ::REQUIRES "DatabaseHandler_v2.CLS"
772 ::REQUIRES "TableH.CLS"
773 ::REQUIRES "AESHandler.CLS"
774 ::REQUIRES "SEHandlerRemote.CLS"
775 ::REQUIRES "SEHandlerOffline.CLS"
776 ::REQUIRES "QRCodeHandler.CLS"
```

Auflistung 18: collection.rexx

B.7.1 Produktverwaltung

Für ein POS-System ist es relevant, etwaige Produkte anlegen zu können. Aus diesem Grund wurde eigens ein GUI dafür konzipiert. Auflistung product.fxml auf der Seite 103 beinhaltet den dazugehörigen Source-Code für die GUI-Darstellung und productController.rexx auf der Seite 105 die dazugehörigen Methoden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import com.jfoenix.controls.JFXButton?>
4  <?import com.jfoenix.controls.JFXTextField?>
5  <?import javafx.scene.control.ChoiceBox?>
6  <?import javafx.scene.control.Label?>
7  <?import javafx.scene.control.TableColumn?>
8  <?import javafx.scene.control.TableView?>
9  <?import javafx.scene.layout.BorderPane?>
10 <?import javafx.scene.layout.ColumnConstraints?>
11 <?import javafx.scene.layout.GridPane?>
12 <?import javafx.scene.layout.HBox?>
13 <?import javafx.scene.layout.RowConstraints?>
14 <?language rexx?>
15
16 <BorderPane prefHeight="770.0" prefWidth="1004.0" stylesheets="@DarkTheme.css" xmlns="http
       ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
17    <center>
18       <GridPane gridLinesVisible="true" prefHeight="742.0" prefWidth="946.0">
19         <columnConstraints>
20           <ColumnConstraints hgrow="SOMETIMES" maxWidth="739.0" minWidth="10.0" prefWidth="
                 692.0" />
21           <ColumnConstraints hgrow="SOMETIMES" maxWidth="401.0" minWidth="10.0" prefWidth="
                 254.0" />
22         </columnConstraints>
23         <rowConstraints>
24           <RowConstraints fillHeight="false" maxHeight="297.0" minHeight="3.0" prefHeight="
                 3.0" vgrow="SOMETIMES" />
25           <RowConstraints maxHeight="610.0" minHeight="10.0" prefHeight="610.0" vgrow="
                 SOMETIMES" />
26           <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
27         </rowConstraints>
28          <children>
29            <Label alignment="CENTER" contentDisplay="CENTER" prefHeight="53.0" prefWidth="
                  693.0" text="Products" />
30            <TableView fx:id="prodTableView" prefHeight="200.0" prefWidth="200.0" GridPane.
                  rowIndex="1">
31              <columns>
32                <TableColumn fx:id="prodIDColumn" prefWidth="64.0" text="ID" />
33                <TableColumn fx:id="prodNameColumn" prefWidth="265.0" text="Name" />
34                  <TableColumn fx:id="prodPriceColumn" prefWidth="94.0" text="Price" />
35                  <TableColumn fx:id="prodTaxColumn" prefWidth="68.0" text="Tax" />
36                  <TableColumn fx:id="prodQuantColumn" prefWidth="97.0" text="Quantity" />
37                  <TableColumn fx:id="prodLunchColumn" prefWidth="128.0" text="Lunch Index"
                        />
38              </columns>
39            </TableView>
40            <GridPane gridLinesVisible="true" prefHeight="442.0" prefWidth="219.0" GridPane.
                  columnIndex="1" GridPane.rowIndex="1">
41              <columnConstraints>
42                <ColumnConstraints fillWidth="false" hgrow="SOMETIMES" minWidth="10.0"
                      prefWidth="100.0" />
```

```
43                  </columnConstraints>
44                  <rowConstraints>
45                    <RowConstraints maxHeight="313.0" minHeight="10.0" prefHeight="100.0" vgrow=
                           "SOMETIMES" />
46                    <RowConstraints maxHeight="516.0" minHeight="10.0" prefHeight="510.0" vgrow=
                           "SOMETIMES" />
47                  </rowConstraints>
48                   <children>
49                      <GridPane alignment="CENTER" prefHeight="99.0" prefWidth="287.0">
50                        <columnConstraints>
51                          <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                                 />
52                        </columnConstraints>
53                        <rowConstraints>
54                          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                 >
55                          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                 >
56                          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                 >
57                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
58                        </rowConstraints>
59                         <children>
60                            <Label alignment="CENTER" prefHeight="26.0" prefWidth="287.0" text="
                                   Starter = 1" />
61                            <Label alignment="CENTER" prefHeight="26.0" prefWidth="301.0" text="
                                   Main Course = 2" GridPane.rowIndex="1" />
62                            <Label alignment="CENTER" prefHeight="26.0" prefWidth="298.0" text="
                                   Dessert = 3" GridPane.rowIndex="2" />
63                            <Label alignment="CENTER" prefHeight="26.0" prefWidth="306.0" text="
                                   No Lunch Product = 0" GridPane.rowIndex="3" />
64                         </children>
65                      </GridPane>
66                      <GridPane alignment="CENTER" prefHeight="510.0" prefWidth="285.0" GridPane
                             .rowIndex="1">
67                        <columnConstraints>
68                          <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0"
                                 />
69                        </columnConstraints>
70                        <rowConstraints>
71                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
72                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
73                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
74                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
75                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
76                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
77                            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES"
                                   />
78                          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                 >
79                          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                 >
80                          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" /
                                 >
81                        </rowConstraints>
82                         <children>
83                            <Label alignment="CENTER" contentDisplay="CENTER" lineSpacing="1.0"
                                   prefHeight="21.0" prefWidth="269.0" text="Name:" underline="true
                                   " />
84                            <JFXTextField fx:id="prodNameTxtF" alignment="CENTER" promptText="
                                   Insert here..." GridPane.rowIndex="1" />
85                            <Label alignment="CENTER" prefHeight="21.0" prefWidth="282.0" text="
                                   Price:" underline="true" GridPane.rowIndex="2" />
86                            <JFXTextField fx:id="prodPriceTxtF" alignment="CENTER" promptText="
                                   Insert here..." GridPane.rowIndex="3" />
87                            <Label alignment="CENTER" prefHeight="26.0" prefWidth="287.0" text="
                                   Quantity:" underline="true" GridPane.halignment="LEFT" GridPane.
                                   rowIndex="4" />
88                            <JFXTextField fx:id="prodQuantTxtF" alignment="CENTER" promptText="
                                   Insert here..." GridPane.rowIndex="5" />
89                            <Label alignment="CENTER" prefHeight="26.0" prefWidth="297.0" text="
                                   Lunchmenu - Index:" underline="true" GridPane.rowIndex="6" />
90                            <ChoiceBox fx:id="lunchChoiceBox" prefHeight="38.0" prefWidth="287.0
                                   " GridPane.rowIndex="7" />
91                            <Label alignment="CENTER" prefHeight="26.0" prefWidth="301.0" text="
                                   Tax:" underline="true" GridPane.rowIndex="8" />
```

```
92                          <ChoiceBox fx:id="taxChoiceBox" prefHeight="38.0" prefWidth="297.0"
                                GridPane.rowIndex="9" />
93                      </children>
94                   </GridPane>
95                </children>
96             </GridPane>
97             <Label alignment="CENTER" prefHeight="17.0" prefWidth="324.0" text="Lunchmenu -
                   Index" underline="true" GridPane.columnIndex="1" />
98             <HBox prefHeight="100.0" prefWidth="200.0" GridPane.rowIndex="2">
99                <children>
100                  <JFXButton fx:id="prodDeleteBtn" onAction="slotDir=arg(arg()); call delete
                           slotDir;" prefHeight="110.0" prefWidth="329.0" text="Delete Product"
                           />
101                  <JFXButton fx:id="prodEditBtn" onAction="slotDir=arg(arg()); call
                           showProductDetails slotDir;" prefHeight="93.0" prefWidth="397.0" text=
                           "Edit Product" />
102               </children>
103            </HBox>
104            <JFXButton fx:id="prodSubmitBtn" onAction="slotDir=arg(arg()); call submit
                   slotDir;" prefHeight="150.0" prefWidth="289.0" text="Submit" GridPane.
                   columnIndex="1" GridPane.rowIndex="2" />
105         </children>
106      </GridPane>
107   </center>
108    <fx:script source="productController.rexx" />
109 </BorderPane>
```

Auflistung 19: product.fxml

```
1  /*
2      This controller manages the tab Products within the menu bar
3  */
4
5  /*@get(lunchChoiceBox taxChoiceBox prodTableView prodIDColumn prodNameColumn prodPriceColumn
       prodTaxColumn prodQuantColumn prodLunchColumn prodEditBtn prodDeleteBtn)*/
6
7   FXCollections = bsf.import("javafx.collections.FXCollections")
8   observTaxArrList = FXCollections~observableArrayList
9   observTaxArrList~add("Please choose one")
10  observTaxArrList~add("20%")
11  observTaxArrList~add("19%")
12  observTaxArrList~add("13%")
13  observTaxArrList~add("10%")
14  observTaxArrList~add("0%")
15  taxChoiceBox~setItems(observTaxArrList)
16  taxChoiceBox~setValue("Please choose one")
17
18  observLunchIndArrList = FXCollections~observableArrayList
19  observLunchIndArrList~add("Please choose one")
20  observLunchIndArrList~add("0")
21  observLunchIndArrList~add("1")
22  observLunchIndArrList~add("2")
23  observLunchIndArrList~add("3")
24  lunchChoiceBox~setItems(observLunchIndArrList)
25  lunchChoiceBox~setValue("Please choose one")
26
27  productsArrayList = .my.app~DatabaseHandler~getAllProductsArrayList
28
29  SIGNAL ON SYNTAX
30  prodHandler = .ProductHandler~new(productsArrayList, prodTableView, prodIDColumn,
       prodNameColumn, prodPriceColumn, prodTaxColumn, prodQuantColumn, prodLunchColumn)
31
32 IF .my.app~EditProduct <> .true THEN DO
33     prodEditBtn~visible = .false
34 END
35 IF .my.app~deleteProduct <> .true THEN DO
36     prodDeleteBtn~visible = .false
37 END
38 ------------------------------------------------------
39 ::ROUTINE showProductDetails PUBLIC
40  /*@get( prodTableView prodNameTxtF prodPriceTxtF prodQuantTxtF lunchChoiceBox taxChoiceBox
       )*/
41
42  item = prodTableView~getSelectionModel~getSelectedItem
43  product = BsfRexxProxy(item)
44  .my.app~prodID = product~id~get
45  prodNameTxtF~text = product~name~get
46  prodPriceTxtF~text = product~price~get
47  prodQuantTxtF~text = product~quantity~get
48  lunchChoiceBox~setValue(product~lunchInd~get)
49
```

```
50  taxString = trunc(product~tax~get*100) -- String has to look like '20%' to will be shown in
        taxChoiceBox
51  taxString = taxString || "%"
52  taxChoiceBox~setValue(taxString)
53  ------------------------------------------------------------
54  ::ROUTINE submit PUBLIC
55   /*@get( prodTableView prodNameTxtF prodPriceTxtF prodQuantTxtF lunchChoiceBox taxChoiceBox
        prodIDColumn prodNameColumn prodPriceColumn prodTaxColumn prodQuantColumn
       prodLunchColumn)*/
56  IF .my.app~DatabaseHandler~productAlreadyExists(.my.app~prodID) THEN DO
57      IF lunchChoiceBox~getValue = "Please choose one" THEN DO
58          .my.app~stageHandler~showErrorWindow("Lunch Index missing" , "Please choose a lunch
                index", "WARNING" )
59      END
60      ELSE IF taxChoiceBox~getValue = "Please choose one" THEN DO
61          .my.app~stageHandler~showErrorWindow("Tax missing" , "Please choose a tax value", "
                WARNING" )
62      END
63      ELSE DO
64          IF formatStringToFloat(prodPriceTxtF~text) = -1 | formatStringToFloat(prodQuantTxtF~
                text) = -1 THEN DO
65              .my.app~stageHandler~showErrorWindow("Not a number" , "Something went wrong
                    during the process. Please check if all inputs are correct", "ERROR" )
66          END
67          ELSE DO
68              .my.app~DatabaseHandler~updateProduct(.my.app~prodID, prodNameTxtF~text,
                    formatStringToFloat(prodPriceTxtF~text), prodQuantTxtF~text,lunchChoiceBox~
                    getValue ,taxChoiceBox~getValue)
69              productsArrayList = .my.app~DatabaseHandler~getAllProductsArrayList
70              prodHandler = .ProductHandler~new(productsArrayList, prodTableView, prodIDColumn
                    , prodNameColumn, prodPriceColumn, prodTaxColumn, prodQuantColumn,
                    prodLunchColumn)
71              prodNameTxtF~text = ""
72              prodPriceTxtF~text = ""
73              prodQuantTxtF~text = ""
74              lunchChoiceBox~getSelectionModel~clearSelection
75              taxChoiceBox~getSelectionModel~clearSelection
76           END
77      END
78  END
79  ELSE DO
80      IF lunchChoiceBox~getValue = "Please choose one" THEN DO
81          .my.app~stageHandler~showErrorWindow("Lunch Index missing" , "Please choose a lunch
                index", "WARNING" )
82      END
83      ELSE IF taxChoiceBox~getValue = "Please choose one" THEN DO
84          .my.app~stageHandler~showErrorWindow("Tax missing" , "Please choose a tax value", "
                WARNING" )
85      END
86      ELSE DO
87          IF formatStringToFloat(prodPriceTxtF~text) = -1 | formatStringToFloat(prodQuantTxtF~
                text) = -1 THEN DO
88              .my.app~stageHandler~showErrorWindow("Not a number" , "Something went wrong
                    during the process. Please check if all inputs are correct", "ERROR" )
89          END
90          ELSE DO
91              .my.app~DatabaseHandler~insertProduct(prodNameTxtF~text, formatStringToFloat(
                    prodPriceTxtF~text), prodQuantTxtF~text,lunchChoiceBox~getValue ,
                    taxChoiceBox~getValue)
92              productsArrayList = .my.app~DatabaseHandler~getAllProductsArrayList
93              prodHandler = .ProductHandler~new(productsArrayList, prodTableView, prodIDColumn
                    , prodNameColumn, prodPriceColumn, prodTaxColumn, prodQuantColumn,
                    prodLunchColumn)
94              prodNameTxtF~text = ""
95              prodPriceTxtF~text = ""
96              prodQuantTxtF~text = ""
97              lunchChoiceBox~getSelectionModel~clearSelection
98              taxChoiceBox~getSelectionModel~clearSelection
99           END
100     END
101 END
102 ------------------------------------------------------------
103 ::ROUTINE delete PUBLIC
104  /*@get( prodTableView prodNameTxtF prodPriceTxtF prodQuantTxtF lunchChoiceBox taxChoiceBox
        prodIDColumn prodNameColumn prodPriceColumn prodTaxColumn prodQuantColumn
       prodLunchColumn)*/
105
106 IF prodTableView~getSelectionModel~getSelectedItem = .nil THEN DO
107     .my.app~stageHandler~showErrorWindow("No Product selected" , "Please choose a product,
            which should be deleted", "ERROR" )
108 END
109 ELSE DO
110     item = prodTableView~getSelectionModel~getSelectedItem
```

```
111
112     product = BsfRexxProxy(item)
113
114     .my.app~DatabaseHandler~deleteProduct(product~id~get)
115     productsArrayList = .my.app~DatabaseHandler~getAllProductsArrayList
116     prodHandler = .ProductHandler~new(productsArrayList, prodTableView, prodIDColumn,
            prodNameColumn, prodPriceColumn, prodTaxColumn, prodQuantColumn, prodLunchColumn)
117     prodNameTxtF~text = ""
118     prodPriceTxtF~text = ""
119     prodQuantTxtF~text = ""
120     lunchChoiceBox~getSelectionModel~clearSelection
121     taxChoiceBox~getSelectionModel~clearSelection
122 END
123 ------------------------------------------------------------
124 ::REQUIRES "BSF.CLS"
125 ::REQUIRES "DatabaseHandler_v2.CLS"
126 ::REQUIRES "TableH.CLS"
127 ::REQUIRES "collection.rexx"
```

Auflistung 20: productController.rexx

Eine genauere Erläuterung, befindet sich im Hauptteil im Kapitel Produkte verwalten auf der Seite 45. Die Abspeicherung der Daten wird im Anhang in Kapitel Datenbankverwaltung auf der Seite 129 dargestellt.

B.7.2 Mittagsmenüverwaltung

Als kleine Erleichterung für Angestellte im Gastronomiebereich, wurde eigens ein GUI konzipiert, welches auf einem Blick die gesamte Mittagsmenü-Karte aufzeigt. Das Design wird durch die Datei lunchMenu.fxml auf der Seite 107 definiert. Der dazugehörige Controller für dieses GUI wird durch die Datei lunchMenuController.rexx auf der Seite 108 definiert.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import com.jfoenix.controls.JFXButton?>
4  <?import javafx.scene.control.Label?>
5  <?import javafx.scene.layout.AnchorPane?>
6  <?import javafx.scene.layout.ColumnConstraints?>
7  <?import javafx.scene.layout.GridPane?>
8  <?import javafx.scene.layout.RowConstraints?>
9  <?import javafx.scene.text.Font?>
10 <?language rexx?>
11
12 <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
       Infinity" prefHeight="400.0" prefWidth="600.0" stylesheets="@DarkTheme.css" xmlns="http
       ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
13     <!--<fx:script source="luchMenuController.rexx" />-->
14    <children>
15       <GridPane alignment="CENTER" layoutX="93.0" layoutY="6.0" prefHeight="292.0" prefWidth
             ="507.0">
16         <columnConstraints>
17           <ColumnConstraints hgrow="SOMETIMES" maxWidth="249.0" minWidth="10.0" prefWidth="
                 170.0" />
18           <ColumnConstraints hgrow="SOMETIMES" maxWidth="337.0" minWidth="10.0" prefWidth="
                 337.0" />
19         </columnConstraints>
20         <rowConstraints>
21           <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
22           <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
23           <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
24         </rowConstraints>
25          <children>
26            <Label alignment="CENTER" prefHeight="46.0" prefWidth="228.0" text="Starter:"
                   underline="true">
27              <font>
28                <Font name="System Bold" size="36.0" />
29              </font>
30            </Label>
31            <Label alignment="CENTER" prefHeight="46.0" prefWidth="228.0" text="Main Course"
                    underline="true" GridPane.rowIndex="1">
```

```
32                <font>
33                   <Font name="System Bold" size="36.0" />
34                </font>
35             </Label>
36             <Label alignment="CENTER" prefHeight="46.0" prefWidth="228.0" text="Dessert:"
                   underline="true" GridPane.rowIndex="2">
37                <font>
38                   <Font name="System Bold" size="36.0" />
39                </font>
40             </Label>
41             <Label fx:id="lunchStarterLbl" prefHeight="46.0" prefWidth="228.0" styleClass="
                   label-lunch" GridPane.columnIndex="1">
42                <font>
43                   <Font name="System Bold" size="36.0" />
44                </font>
45             </Label>
46             <Label fx:id="lunchMainLbl" prefHeight="46.0" prefWidth="228.0" styleClass="
                   label-lunch" GridPane.columnIndex="1" GridPane.rowIndex="1">
47                <font>
48                   <Font name="System Bold" size="36.0" />
49                </font>
50             </Label>
51             <Label fx:id="lunchDessertLbl" prefHeight="46.0" prefWidth="228.0" styleClass="
                   label-lunch" GridPane.columnIndex="1" GridPane.rowIndex="2">
52                <font>
53                   <Font name="System Bold" size="36.0" />
54                </font>
55             </Label>
56          </children>
57       </GridPane>
58       <JFXButton fx:id="lunchBackBtn" layoutX="261.0" layoutY="319.0" onAction="slotDir=arg(
             arg()); call back slotDir;" text="Back" />
59    </children>
60     <fx:script source="lunchMenuController.rexx" />
61 </AnchorPane>
```

Auflistung 21: lunchMenu.fxml

```
1  /*@get(lunchStarterLbl lunchMainLbl lunchDessertLbl lunchBackBtn)*/
2
3
4  lunchStarterLbl~setText( .my.app~DatabaseHandler~getLunchMenuStarter)
5  lunchMainLbl~setText( .my.app~DatabaseHandler~getLunchMenuMain)
6  lunchDessertLbl~setText( .my.app~DatabaseHandler~getLunchMenuDessert)
7  ------------------------------------------------------------
8  ::ROUTINE back PUBLIC
9  use arg slotDir
10 .my.app~stageHandler~closeWindow
11 ------------------------------------------------------------
12 ::REQUIRES "BSF.CLS"
13 ::REQUIRES "DatabaseHandler_v2.CLS"
```

Auflistung 22: lunchMenuController.rexx

Sofern jedoch das Mittagsmenü zusammengestellt werde muss, kommen andere Benutzeroberflächen zu tragen. Auflistung editLunchMenu.fxml definiert das Fenster, welches erzeugt wird sofern das Mittagsmenü angepasst werden soll. Die dazugehörige Logik befindet sich in der Controller-Datei, siehe Auflistung editLunchMenuController.rexx auf der Seite 110.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import com.jfoenix.controls.JFXButton?>
4  <?import com.jfoenix.controls.JFXRadioButton?>
5  <?import javafx.scene.control.Label?>
6  <?import javafx.scene.control.TableColumn?>
7  <?import javafx.scene.control.TableView?>
8  <?import javafx.scene.control.ToggleGroup?>
9  <?import javafx.scene.layout.BorderPane?>
10 <?import javafx.scene.layout.ColumnConstraints?>
11 <?import javafx.scene.layout.GridPane?>
12 <?import javafx.scene.layout.HBox?>
13 <?import javafx.scene.layout.RowConstraints?>
14 <?import javafx.scene.layout.VBox?>
```

```
<?import javafx.scene.text.Font?>
<?language rexx?>

<BorderPane stylesheets="@DarkTheme.css" xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="
    http://javafx.com/fxml/1">
   <center>
      <GridPane gridLinesVisible="true" prefHeight="742.0" prefWidth="946.0">
        <columnConstraints>
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="739.0" minWidth="10.0" prefWidth="
              692.0" />
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="401.0" minWidth="10.0" prefWidth="
              254.0" />
        </columnConstraints>
        <rowConstraints>
          <RowConstraints fillHeight="false" maxHeight="297.0" minHeight="3.0" prefHeight="
              3.0" vgrow="SOMETIMES" />
          <RowConstraints maxHeight="610.0" minHeight="10.0" prefHeight="610.0" vgrow="
              SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
        </rowConstraints>
         <children>
            <Label alignment="CENTER" contentDisplay="CENTER" prefHeight="53.0" prefWidth="
                693.0" text="Products" />
            <TableView fx:id="prodTableView" prefHeight="200.0" prefWidth="200.0" GridPane.
                rowIndex="1">
              <columns>
                <TableColumn fx:id="prodIDColumn" prefWidth="75.0" text="ID" />
                <TableColumn fx:id="prodNameColumn" prefWidth="232.0" text="Name" />
                  <TableColumn fx:id="prodPriceColumn" prefWidth="83.0" text="Price" />
                  <TableColumn fx:id="prodTaxColumn" prefWidth="71.0" text="Tax" />
                  <TableColumn fx:id="prodQuantColumn" prefWidth="91.0" text="Quantity" />
                  <TableColumn fx:id="prodLunchColumn" prefWidth="135.0" text="Lunch Index"
                      />
              </columns>
            </TableView>
            <Label alignment="CENTER" prefHeight="17.0" prefWidth="324.0" text="Lunchmenu -
                Index" underline="true" GridPane.columnIndex="1" />
            <HBox prefHeight="100.0" prefWidth="200.0" GridPane.rowIndex="2">
               <children>
                  <JFXButton fx:id="prodEditBtn" onAction="slotDir=arg(arg()); call
                      showProductDetails slotDir;" prefHeight="79.0" prefWidth="697.0" text=
                      "Edit Product" />
               </children>
            </HBox>
            <JFXButton fx:id="prodSubmitBtn" onAction="slotDir=arg(arg()); call submit
                slotDir;" prefHeight="150.0" prefWidth="289.0" text="Submit" GridPane.
                columnIndex="1" GridPane.rowIndex="2" />
            <VBox alignment="TOP_CENTER" prefHeight="200.0" prefWidth="100.0" spacing="25.0"
                 GridPane.columnIndex="1" GridPane.rowIndex="1">
               <children>
                  <Label fx:id="prodNameLbl" alignment="CENTER" prefHeight="67.0" prefWidth=
                      "254.0" text="Productname" />
                  <Label fx:id="prodIDLbl" lineSpacing="30.0" text="ID" />
                  <JFXRadioButton fx:id="nothingRadio" lineSpacing="1.0" prefHeight="27.0"
                       prefWidth="134.0" selected="true" style="-jfx-selected-color: #5c85d6;
                       " styleClass="jfx-radio-button" text="Nothing">
                     <toggleGroup>
                        <ToggleGroup fx:id="luchChoice" />
                     </toggleGroup>
                     <font>
                        <Font name="System Bold" size="18.0" />
                     </font>
                  </JFXRadioButton>
                  <JFXRadioButton fx:id="starterRadio" prefHeight="27.0" prefWidth="134.0"
                       style="-jfx-selected-color: #5c85d6;" styleClass="jfx-radio-button"
                       text="Starter" toggleGroup="$luchChoice">
                     <font>
                        <Font name="System Bold" size="18.0" />
                     </font>
                  </JFXRadioButton>
                  <JFXRadioButton fx:id="mainRadio" prefWidth="134.0" style="-jfx-selected-
                       color: #5c85d6;" styleClass="jfx-radio-button" text="Main" toggleGroup
                       ="$luchChoice">
                     <font>
                        <Font name="System Bold" size="18.0" />
                     </font>
                  </JFXRadioButton>
                  <JFXRadioButton fx:id="dessertRadio" prefWidth="134.0" style="-jfx-
                       selected-color: #5c85d6;" styleClass="jfx-radio-button" text="Dessert"
                        toggleGroup="$luchChoice">
                     <font>
                        <Font name="System Bold" size="18.0" />
                     </font>
```

```
                    </JFXRadioButton>
                </children>
            </VBox>
        </children>
    </GridPane>
  </center>
   <fx:script source="editLunchMenuController.rexx" />
</BorderPane>
```

Auflistung 23: editLunchMenu.fxml

```
/*@get(prodTableView prodIDColumn prodNameColumn prodPriceColumn prodTaxColumn
    prodQuantColumn prodLunchColumn  prodIDLbl)*/
/*@get(luchChoice starterRadio mainRadio dessertRadio)*/

 productsArrayList = .my.app~DatabaseHandler~getAllProductsArrayList
 prodHandler = .ProductHandler~new(productsArrayList, prodTableView, prodIDColumn,
      prodNameColumn, prodPriceColumn, prodTaxColumn, prodQuantColumn, prodLunchColumn)
------------------------------------------------------------
::ROUTINE showProductDetails PUBLIC
/*@get( prodTableView prodNameLbl prodIDLbl starterRadio mainRadio dessertRadio nothingRadio
    )*/

 item = prodTableView~getSelectionModel~getSelectedItem
 product = BsfRexxProxy(item)
 prodID = product~id~get
 prodNameLbl~setText(product~name~get)
 prodIDLbl~setText(prodID)

IF product~lunchInd~get  == 0 THEN DO
    nothingRadio~setSelected(.true)
END
IF product~lunchInd~get  == 1 THEN DO
    starterRadio~setSelected(.true)
END
IF product~lunchInd~get  == 2 THEN DO
    mainRadio~setSelected(.true)
END
IF product~lunchInd~get  == 3 THEN DO
    dessertRadio~setSelected(.true)
END
------------------------------------------------------------
::ROUTINE submit PUBLIC
/*@get(prodTableView prodIDColumn prodNameColumn prodPriceColumn prodTaxColumn
    prodQuantColumn prodLunchColumn)*/
/*@get( prodTableView luchChoice prodNameLbl prodIDLbl starterRadio mainRadio dessertRadio
    nothingRadio)*/
 selected = luchChoice~getSelectedToggle~toString
 PARSE VAR selected "'" selected "'"

IF selected == "Nothing" THEN DO
    lunchInd = 0
END
IF selected == "Starter" THEN DO
    lunchInd = 1
END
IF selected == "Main" THEN DO
    lunchInd = 2
END
IF selected == "Dessert" THEN DO
    lunchInd = 3
END
 prodID = prodIDLbl~getText

 .my.app~DatabaseHandler~updateProductLunchInd(prodID,lunchInd)

 productsArrayList = .my.app~DatabaseHandler~getAllProductsArrayList
 prodHandler = .ProductHandler~new(productsArrayList, prodTableView, prodIDColumn,
     prodNameColumn, prodPriceColumn, prodTaxColumn, prodQuantColumn, prodLunchColumn)
 prodIDLbl~setText("ID")
 prodNameLbl~text = "Productname"
 nothingRadio~setSelected(.true)
------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "DatabaseHandler_v2.CLS"
::REQUIRES "TableH.CLS"
```

Auflistung 24: editLunchMenuController.rexx

Eine genauere Erläuterung, befindet sich im Hauptteil im Kapitel Mittagsmenü verwalten auf der Seite 43. Die Abspeicherung der Daten wird im Anhang in Kapitel Datenbankverwaltung auf der Seite 129 dargestellt.

B.7.3 Bestellungsverwaltung

Im Sinne einer effizienten Registrierkasse, ist es essenziell, dass eine Bestellung von Produkten aufgegeben werden kann. Für diesen Fall wurde ein weiteres eigenständiges GUI designed und eingebunden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import com.jfoenix.controls.JFXButton?>
4  <?import com.jfoenix.controls.JFXComboBox?>
5  <?import com.jfoenix.controls.JFXRadioButton?>
6  <?import javafx.scene.control.Label?>
7  <?import javafx.scene.control.TableColumn?>
8  <?import javafx.scene.control.TableView?>
9  <?import javafx.scene.control.ToggleGroup?>
10 <?import javafx.scene.layout.BorderPane?>
11 <?import javafx.scene.layout.ColumnConstraints?>
12 <?import javafx.scene.layout.GridPane?>
13 <?import javafx.scene.layout.RowConstraints?>
14 <?language rexx?>
15
16 <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
       Infinity" prefHeight="502.0" prefWidth="724.0" stylesheets="@DarkTheme.css" xmlns="http
       ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
17    <left>
18       <TableView fx:id="newOrderProductTableView" prefHeight="502.0" prefWidth="293.0"
             BorderPane.alignment="CENTER">
19         <columns>
20           <TableColumn fx:id="newOrderProductID" prefWidth="75.0" text="ProductID" />
21           <TableColumn fx:id="newOrderProductTName" prefWidth="74.0" text="Name" />
22             <TableColumn fx:id="newOrderProductPrice" prefWidth="58.0" text="Price" />
23             <TableColumn fx:id="newOrderProductQuantity" prefWidth="81.0" text="Quantity" />
24         </columns>
25       </TableView>
26    </left>
27    <right>
28       <TableView fx:id="newOrderOrderTableView" prefHeight="502.0" prefWidth="299.0"
             BorderPane.alignment="CENTER">
29         <columns>
30           <TableColumn fx:id="newOrderOrderProdID" prefWidth="75.0" text="ProductID" />
31           <TableColumn fx:id="newOrderOrderProdName" prefWidth="147.0" text="Name" />
32             <TableColumn fx:id="newOrderOrderProdQuant" prefWidth="89.0" text="Quantity" />
33         </columns>
34       </TableView>
35    </right>
36    <center>
37       <GridPane gridLinesVisible="true" BorderPane.alignment="CENTER">
38         <columnConstraints>
39           <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
40         </columnConstraints>
41         <rowConstraints>
42             <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
43           <RowConstraints maxHeight="95.0" minHeight="10.0" prefHeight="62.0" vgrow="
                 SOMETIMES" />
44           <RowConstraints maxHeight="138.0" minHeight="10.0" prefHeight="86.0" vgrow="
                 SOMETIMES" />
45             <RowConstraints maxHeight="154.0" minHeight="10.0" prefHeight="154.0" vgrow="
                   SOMETIMES" />
46           <RowConstraints maxHeight="166.0" minHeight="10.0" prefHeight="118.0" vgrow="
                 SOMETIMES" />
47         </rowConstraints>
48          <children>
49            <Label fx:id="newOrderOrderIDLbl" prefHeight="45.0" prefWidth="144.0" text="
                   OrderID:" underline="true" />
50            <Label fx:id="newOrderTableNRLbl" prefHeight="21.0" prefWidth="149.0" text="
                   TableNr.:" underline="true" GridPane.rowIndex="1" />
51            <JFXButton fx:id="newOrderSaveBtn" onAction="slotDir=arg(arg()); call save
                   slotDir;" prefHeight="181.0" prefWidth="133.0" text="Save" GridPane.rowIndex
                   ="4" />
52            <JFXComboBox fx:id="newOrderComboBox" prefHeight="88.0" prefWidth="132.0"
                   promptText="Select Table" GridPane.rowIndex="2" />
```

```
53              <GridPane GridPane.rowIndex="3">
54                <columnConstraints>
55                  <ColumnConstraints fillWidth="false" hgrow="SOMETIMES" minWidth="10.0"
                        prefWidth="100.0" />
56                </columnConstraints>
57                <rowConstraints>
58                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
59                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
60                  <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
61                    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
62                    <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
63                </rowConstraints>
64                 <children>
65                    <JFXRadioButton fx:id="receiptNormal" prefHeight="17.0" prefWidth="73.0"
                          selected="true" style="-jfx-selected-color: #5c85d6;" text="Normal"
                          GridPane.halignment="CENTER">
66                        <toggleGroup>
67                          <ToggleGroup fx:id="receiptTypeGroup" />
68                        </toggleGroup>
69                    </JFXRadioButton>
70                    <JFXRadioButton fx:id="receiptStart" prefHeight="17.0" prefWidth="73.0"
                          text="Start" style="-jfx-selected-color: #5c85d6;" toggleGroup="
                          $receiptTypeGroup" GridPane.halignment="CENTER" GridPane.rowIndex="1"
                          />
71                    <JFXRadioButton fx:id="receiptNull" prefHeight="17.0" prefWidth="73.0"
                          text="Null"  style="-jfx-selected-color: #5c85d6;" toggleGroup="
                          $receiptTypeGroup" GridPane.halignment="CENTER" GridPane.rowIndex="2"
                          />
72                    <JFXRadioButton fx:id="receiptTraining" prefWidth="73.0" text="Training"
                          style="-jfx-selected-color: #5c85d6;" toggleGroup="$receiptTypeGroup"
                          GridPane.halignment="CENTER" GridPane.rowIndex="3" />
73                    <JFXRadioButton fx:id="receiptStorno" prefWidth="73.0" text="Storno" style
                          ="-jfx-selected-color: #5c85d6;" toggleGroup="$receiptTypeGroup"
                          GridPane.halignment="CENTER" GridPane.rowIndex="4" />
74
75                 </children>
76              </GridPane>
77            </children>
78        </GridPane>
79    </center>
80     <fx:script source="newOrderController.rexx" />
81  </BorderPane>
```

Auflistung 25: newOrder_v2.fxml

Auflistung newOrder_v2.fxml oberhalb auf der Seite 111 definiert das Aussehen des GUIs und Auflistung newOrderController.rexx unterhalb auf der Seite 112 definiert die Funktionalität.

```
1  /*@get(newOrderProductTableView newOrderProductID newOrderProductTName newOrderProductPrice
       newOrderProductQuantity newOrderComboBox newOrderOrderIDLbl newOrderTableNRLbl)*/
2  /*@get(newOrderOrderTableView newOrderOrderProdID newOrderOrderProdName
       newOrderOrderProdQuant)*/
3  /*@get(receiptStart receiptNull receiptTraining receiptStorno receiptNormal)*/
4
5  FXCollections = bsf.import("javafx.collections.FXCollections")
6  productsArrayList = .my.app~DatabaseHandler~getAllProductsArrayList
7  someOrderExist  = .my.app~DatabaseHandler~checkOrders
8
9  IF \someOrderExist THEN DO
10     receiptStart~setSelected(.true)
11     receiptNull~setDisable(.true)
12     receiptTraining~setDisable(.true)
13     receiptStorno~setDisable(.true)
14     receiptNormal~setDisable(.true)
15 END
16
17 observTableArrList = FXCollections~observableArrayList
18 IF .my.app~currentTableNR == .nil THEN DO
19     tableNr = .array~of( "No selected Table","1", "2", "3", "4", "5", "6", "7", "8", "9", "
           10")
20     DO p OVER tableNr
21         observTableArrList~add(p)
22     END
23     newOrderComboBox~setItems(observTableArrList)
24     newOrderComboBox~setValue("No selected Table")
25 END
26 ELSE DO
27     observTableArrList~add(.my.app~currentTableNR)
```

```
28      newOrderComboBox~setItems(observTableArrList)
29      newOrderComboBox~setValue(.my.app~currentTableNR);
30  END
31
32  IF .my.app~currentOrderID == .nil THEN DO
33      newOrderOrderIDLbl~setText("New Order")
34      newOrderTableNRLbl~setText("Select new Table")
35      newOrderDetailsArrayList = FXCollections~observableArrayList
36      .my.app~newOrderDetailsHandler = .newOrderDetailsHandler~new(newOrderDetailsArrayList,
            newOrderOrderTableView, newOrderOrderProdID, newOrderOrderProdName,
            newOrderOrderProdQuant)
37  END
38  ELSE DO
39      newOrderOrderIDLbl~setText("OrderID = " || .my.app~currentOrderID)
40      newOrderTableNRLbl~setText("TableNr = " || .my.app~currentTableNR)
41      newOrderDetailsArrayList = .my.app~DatabaseHandler~getAllProductsForNewOrderByID(.my.app
            ~currentOrderID) -- List by orderID will be loaded
42      .my.app~newOrderDetailsHandler = .newOrderDetailsHandler~new(newOrderDetailsArrayList,
            newOrderOrderTableView, newOrderOrderProdID, newOrderOrderProdName,
            newOrderOrderProdQuant)
43  END
44
45  newOrderHandler = .newOrderHandler~new(productsArrayList, newOrderProductTableView,
        newOrderProductID, newOrderProductTName, newOrderProductPrice, newOrderProductQuantity)
46  ------------------------------------------------------------
47  ::ROUTINE save PUBLIC
48  /*@get(newOrderOrderTableView newOrderOrderProdID newOrderOrderProdName
        newOrderOrderProdQuant newOrderComboBox)*/
49  /*@get(receiptTypeGroup)*/
50   receiptChoice = receiptTypeGroup~getToggles~indexOf(receiptTypeGroup~getSelectedToggle)
51          SELECT
52              WHEN receiptChoice = 0 THEN receiptType = "Normal"
53              WHEN receiptChoice = 1 THEN receiptType = "Start"
54              WHEN receiptChoice = 2 THEN receiptType = "Null"
55              WHEN receiptChoice = 3 THEN receiptType = "Training"
56              WHEN receiptChoice = 4 THEN receiptType = "Storno"
57          END
58
59  IF .my.app~currentOrderID = .nil THEN DO
60      IF newOrderComboBox~getValue <>  "No selected Table" THEN DO
61          .my.app~DatabaseHandler~insertOrder(newOrderComboBox~getValue, receiptType)
62          orderID = .my.app~DatabaseHandler~getLastOrderID
63          DO item OVER newOrderOrderTableView~getItems
64              DO i = 1 FOR item~quantity~get
65                  .my.app~DatabaseHandler~insertContains("not paid", item~PID~get, orderID)
66              END
67          END
68                .my.app~stageHandler~backToRoot
69                .my.app~stageHandler~closeWindow
70          END
71          ELSE DO
72              .my.app~stageHandler~showErrorWindow("ERROR","Please choose a table number","
                    ERROR")
73          END
74  END
75  ELSE DO
76      .my.app~DatabaseHandler~removeOrdersFromDB(.my.app~currentOrderID)
77      DO item OVER newOrderOrderTableView~getItems
78          DO i = 1 FOR item~quantity~get
79               .my.app~DatabaseHandler~insertContains("not paid", item~PID~get, .my.app~
                     currentOrderID)
80          END
81      END
82      .my.app~stageHandler~backToRoot
83      .my.app~stageHandler~closeWindow
84  END
85
86  ------------------------------------------------------------
87  ::REQUIRES "BSF.CLS"
88  ::REQUIRES "DatabaseHandler_v2.CLS"
89  ::REQUIRES "TableH.CLS"
```

Auflistung 26: newOrderController.rexx

Eine genauere Erläuterung, befindet sich im Hauptteil im Kapitel Bestellungen verwalten auf der Seite 48. Die Abspeicherung der Daten wird im Anhang im Kapitel Datenbankverwaltung auf der Seite 129 dargestellt.

B.8 Passwortverwaltung

Wie im Anhang im Kapitel Loginverwaltung auf der Seite 86 erwähnt wurde, ist es essenziell, dass etwaige Passwörter sicher abgespeichert und verwaltet werden. Für diesen Zweck wurde eigens eine eigene Rexx-Klasse erstellt mit mehreren Methoden, die für die Passwortverwaltung sinnvoll sind. Ersichtlich sind diese Methoden in der Auflistung PasswordHandler.cls auf der Seite 114.

```
/*
 Copyright (c) 2006 Damien Miller <djm@mindrot.org>

 Permission to use, copy, modify, and distribute this software for any
 purpose with or without fee is hereby granted, provided that the above
 copyright notice and this permission notice appear in all copies.

 THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
 WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
 MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
 ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
 WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
 ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
 OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
*/
::CLASS PasswordHandler PUBLIC
::ATTRIBUTE BCrypt
::ATTRIBUTE String
::METHOD init
 EXPOSE BCrypt String
 BCrypt = bsf.import("org.springframework.security.crypto.bcrypt.BCrypt")
------------------------------------------------------------
::METHOD hashPassword PUBLIC
 EXPOSE BCrypt hashedPWD
 USE ARG plainPW

 salt = BCrypt~gensalt(12)
 hashedPWD =  BCrypt~hashpw(plainPW, salt)
 RETURN hashedPWD
------------------------------------------------------------
::METHOD checkPass
 EXPOSE BCrypt
 USE ARG plainPW, hashedPWD

 password_verified = BCrypt~checkpw(plainPW, hashedPWD)
 RETURN password_verified
------------------------------------------------------------
::REQUIRES "BSF.CLS"
```

Auflistung 27: PasswordHandler.cls

Sofern ein Benutzer nachträglich sein Passwort ändern möchte, erscheint ein separates GUI, mithilfe dessen das Passwort geändert werden kann. Das Aussehen wird durch den Source-Code in der Auflistung passwordChange.fxml auf Seite 114 definiert. Die Funktionalität wird in der Datei passwordChangeController.rexx auf der Seite 115 definiert.

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

<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXPasswordField?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?language rexx?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
    Infinity" prefHeight="400.0" prefWidth="600.0" stylesheets="@DarkTheme.css" xmlns="http
    ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <center>
      <GridPane prefHeight="318.0" prefWidth="600.0" BorderPane.alignment="CENTER">
```

```
        <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>
         <children>
            <Label prefHeight="57.0" prefWidth="300.0" text="Old Password:" underline="true" />
            <Label prefHeight="17.0" prefWidth="311.0" text="New Password" underline="true" GridPane.rowIndex="1" />
            <Label text="Repeat New Password:" underline="true" GridPane.rowIndex="2" />
            <JFXPasswordField fx:id="oldPasswordTxtF" GridPane.columnIndex="1" />
            <JFXPasswordField fx:id="newPasswordTxtF" GridPane.columnIndex="1" GridPane.rowIndex="1" />
            <JFXPasswordField fx:id="repeatPasswordTxtF" GridPane.columnIndex="1" GridPane.rowIndex="2" />
         </children>
      </GridPane>
   </center>
   <bottom>
      <JFXButton fx:id="passwordSubmitBtn"  prefHeight="97.0" prefWidth="173.0" onAction="slotDir=arg(arg()); call submitChange slotDir;" text="Submit" BorderPane.alignment="CENTER" />
   </bottom>
    <fx:script source="passwordChangeController.rexx" />
</BorderPane>
```

Auflistung 28: passwordChange.fxml

```
/*
    This controller handles the Password Change tab within the menu bar
*/

::ROUTINE submitChange PUBLIC
 /*@get(oldPasswordTxtF newPasswordTxtF repeatPasswordTxtF)*/

IF newPasswordTxtF~text == repeatPasswordTxtF~text THEN DO
    check =.my.app~DatabaseHandler~checkPassword(.my.app~userID, oldPasswordTxtF~text)
    IF check THEN DO
        .my.app~DatabaseHandler~updateUserPassword(.my.app~userID, newPasswordTxtF~text)
    END
    ELSE DO
        .my.app~stageHandler~showErrorWindow("Password change failed", "Old password is wrong!", "ERROR")
    END
END
ELSE DO
    .my.app~stageHandler~showErrorWindow("Password change failed", '"New Password" and "Repeat New Password" do not match!', "ERROR")
END
------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "DatabaseHandler_v2.CLS"
```

Auflistung 29: passwordChangeController.rexx

B.9 Kryptografie

Ein weiterer wichtiger Aspekt, damit ein System RKSV-konform ist, ist der Umstand, dass der Stand des integrierten Umsatzzählers verschlüsselt am selbigen Beleg hinterlegt werden muss. Die Verschlüsselung erfolgt mittels AES256 und wird innerhalb des QR-Code übergeben. Für diesen Fall wurde eine eigene Rexx-Klasse erstellt, die für die Verschlüsselung herangezogen wird. Die entsprechende Datei AESHandler.cls ist auf der Seite 116 dargestellt.

```
/*
    This Class manages the encryption for the turnover value, which has to be included within the QR-Code.
    For testing purpose, a ::ROUTINE which decrypts the value has been included.
*/
::CLASS AESHandler PUBLIC
::ATTRIBUTE Cipher
::ATTRIBUTE SecretKeySpec
::ATTRIBUTE KeyGenerator
::ATTRIBUTE StandardCharsets
::ATTRIBUTE Base64
::ATTRIBUTE IvParameterSpec
::ATTRIBUTE IV
::ATTRIBUTE MessageDigest
::METHOD init
 EXPOSE Cipher SecretKeySpec KeyGenerator StandardCharsets Base64 IvParameterSpec MessageDigest
 Cipher = bsf.import("javax.crypto.Cipher")
 SecretKeySpec = bsf.import("javax.crypto.spec.SecretKeySpec")
 KeyGenerator = bsf.import("javax.crypto.KeyGenerator")
 Base64 = bsf.import("java.util.Base64")
 StandardCharsets = bsf.import("java.nio.charset.StandardCharsets")
 IvParameterSpec = bsf.import("javax.crypto.spec.IvParameterSpec")
 MessageDigest = bsf.loadClass("java.security.MessageDigest")
------------------------------------------------------------
::METHOD encryptAES PUBLIC
 EXPOSE Cipher  SecretKeySpec Base64 IvParameterSpec IV
 USE ARG turnover, secretKey, cashID, receiptID, turnOverCounterLengthInBytes = 5
 ByteBuffer = bsf.loadClass("java.nio.ByteBuffer")
 Decoder = Base64~getDecoder
 Encoder = Base64~getEncoder
 decodedSecKey = Decoder~decode(secretKey)

 IV = self~getIV(cashID,receiptID) -- IV = byteArray

 byteBufferIV = ByteBuffer~allocate(16)
 byteBufferIV~put(IV)
 IV = byteBufferIV~array

 turnover = box("Long",turnover)
 byteBufferData = ByteBuffer~allocate(16)
 byteBufferData~putLong(turnover)
 data = byteBufferData~array

 turnOverCounterByteRep = self~get2ComplementRepForLong(turnover, turnOverCounterLengthInBytes)
 bsf.loadClass("java.lang.System")~arraycopy(turnOverCounterByteRep, 0, data, 0, turnOverCounterByteRep~size)

 IVspec = IvParameterSpec~new(IV)
 aesKey = SecretKeySpec~new(decodedSecKey, "AES")
 cipher = Cipher~getInstance("AES/CTR/NoPadding")
 cipher~bsf.invoke("init", Cipher~ENCRYPT_MODE, aesKey, IVspec)
 encryptedTurnOverValueComplete = cipher~doFinal(data)
 encryptedTurnOverValue = bsf.CreateJavaArray("byte.class", turnOverCounterLengthInBytes)

 bsf.loadClass("java.lang.System")~arraycopy(encryptedTurnOverValueComplete, 0, encryptedTurnOverValue, 0, turnOverCounterLengthInBytes)
 encryptedB64 = Encoder~encodeToString(encryptedTurnOverValue)    -- to get base64-String and no byte array
 RETURN encryptedB64
------------------------------------------------------------
::METHOD getIV PUBLIC
 EXPOSE StandardCharsets MessageDigest
 USE ARG cashboxID, receiptID

 utf8cashboxID = StandardCharsets~UTF_8~encode(cashboxID)
 utf8receiptID = StandardCharsets~UTF_8~encode(receiptID)

 utf8IV_chain = utf8cashboxID || utf8receiptID

 digest = MessageDigest~getInstance("SHA-256") -- returns 32 byte
 encodedHash = digest~digest(BsfRawBytes(utf8IV_chain))
 encodedHashExt = bsf.createJavaArray("byte.class", 16)
 bsf.loadClass("java.lang.System")~arraycopy(encodedHash, 0, encodedHashExt, 0, 16) -- extracting first 16 byte of 32 as IV
 RETURN encodedHashExt
------------------------------------------------------------
::METHOD decryptAES PUBLIC
 EXPOSE Cipher  SecretKeySpec Base64 IvParameterSpec IV
 USE ARG encryptedTurnOverValueInB64, secretKey, cashID, receiptID, turnOverCounterLengthInBytes = 5
 ByteBuffer = bsf.loadClass("java.nio.ByteBuffer")
```

```
76
77   Decoder = Base64~getDecoder
78   Encoder = Base64~getEncoder
79   secretKey = Decoder~decode(secretKey)
80   decodedTurnOverValue = Decoder~decode(encryptedTurnOverValueInB64)
81
82   byteBufferIV = ByteBuffer~allocate(16)
83   byteBufferIV~put(IV)
84   IV = byteBufferIV~array
85
86   IV = IvParameterSpec~new(IV)                              --use same IV to decode AES
87   aesKey = SecretKeySpec~new(secretKey, "AES")
88   cipher = Cipher~getInstance("AES/CTR/NoPadding")
89   cipher~bsf.invoke("init", Cipher~DECRYPT_MODE, aesKey, IV)
90   turnover = cipher~doFinal(decodedTurnOverValue)
91   textB64 = Encoder~encodeToString(turnover)    -- to get base64-String and no byte array
92   RETURN self~getLong(turnover)
93  ------------------------------------------------------------
94  ::METHOD hashLastReceipt PUBLIC
95   EXPOSE MessageDigest StandardCharsets Base64
96   USE ARG text, N=8
97   Encoder = Base64~getEncoder
98
99   javaString = box("STring", text)
100  digest = MessageDigest~getInstance("SHA-256")
101  hash = digest~digest(javaString~getBytes(StandardCharsets~UTF_8))
102  conDigest = bsf.CreateJavaArray("byte.class", N)
103  bsf.loadClass("java.lang.System")~arraycopy(hash, 0, conDigest, 0, N)
104  RETURN Encoder~encodeToString(conDigest)
105 ------------------------------------------------------------
106 ::METHOD get2ComplementRepForLong PUBLIC
107  USE ARG value, numberOfBytesFor2ComplementRepresentation
108  ByteBuffer = bsf.loadClass("java.nio.ByteBuffer")
109  byteBufferArray = ByteBuffer~allocate(8)
110
111  byteBufferArray~putLong(value)
112  longRep = byteBufferArray~array
113
114 IF numberOfBytesFor2ComplementRepresentation = 8 THEN DO
115     RETURN longRep
116 END
117  byteRep = bsf.CreateJavaArray("byte.class", numberOfBytesFor2ComplementRepresentation)
118
119  bsf.loadClass("java.lang.System")~arraycopy(longRep, 8-
         numberOfBytesFor2ComplementRepresentation, byteRep,0,
         numberOfBytesFor2ComplementRepresentation)
120  RETURN byteRep
121 ------------------------------------------------------------
122 ::METHOD getLong PUBLIC
123  USE ARG bytes
124
125  BigInteger = bsf.import("java.math.BigInteger")~new(bytes)
126  RETURN BigInteger~longValue
127 ------------------------------------------------------------
128 ::METHOD generateRandomAESKey PUBLIC
129  EXPOSE KeyGenerator Base64
130
131  keyGenAES = KeyGenerator~getInstance("AES")
132  keyGenAES~bsf.invoke("init", 256)
133  secretKey = keyGenAES~generateKey
134  encodedSecKey = secretKey~getEncoded
135  Encoder = Base64~getEncoder
136  base64Key = Encoder~encodeToString(encodedSecKey)
137  RETURN base64Key
138  ------------------------------------------------------------
139 ::REQUIRES "BSF.CLS"
```

Auflistung 30: AESHandler.cls

B.10 QR-Code

Ein weiterer essenzieller Punkt der Registrierkassensicherheitsverordnung ist, dass entsprechende Daten und Informationen mittels QR-Code auf dem Beleg abgebildet werden. Welche Daten und Informationen dafür benötigt werden,

steht im Kapitel QR-Code auf der Seite 26 im Hauptteil dieser Arbeit. Mit Hilfe der Datei QRCodeHandler.CLS auf der Seite 118 werden die gesammelten Informationen in einen entsprechenden QR-Code übertragen.

```
/*
    This Class creates and stores QR-Code-Images for the receipt, created by the POS-System
*/

::CLASS QRCodeHandler PUBLIC
::METHOD BarcodeFormat ATTRIBUTE
::METHOD BitMatrix ATTRIBUTE
::METHOD QRCodeWriter ATTRIBUTE
::METHOD MatrixToImageWriter ATTRIBUTE
::METHOD init
 EXPOSE BarcodeFormat BitMatrix QRCodeWriter MatrixToImageWriter

 BarcodeFormat = bsf.loadClass("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 makeQRCode PUBLIC
 EXPOSE BarcodeFormat BitMatrix QRCodeWriter MatrixToImageWriter
 USE ARG text
 text = text~makeString
 CALL mkDir "QR"
 path = "QR/qr.png"
 path = .bsf~new("java.io.File", path)
 QRCodeWriter = QRCodeWriter~new
 bitMatrix = QRCodeWriter~encode(text, BarcodeFormat~QR_CODE, 300, 300)
 MatrixToImageWriter~writeToFile(bitMatrix, "PNG", path)
 RETURN .true
-------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "collection.rexx"
```

Auflistung 31: QRCodeHandler.CLS

B.11 Sicherheitseinrichtung

Im Kapitel Zertifizierungspflicht auf der Seite 16 wird dem Umstand Sorge getragen, dass innerhalb einer RKSV-konformen Registrierkasse eine sogenannte Signatur- bzw. Siegelerstellungseinheit benutzt werden muss. Diese Einheit, auch manchmal als Sicherheitseinrichtung bezeichnet, dient dafür, dass etwaige Belege signiert werden und somit nicht mehr nachträglich geändert werden können. Für diesen Zweck gibt es mehrere Methoden einen Beleg signieren zu können. Einerseits kann man einen Beleg ohne Zuhilfenahme des Internets signieren lassen, welches im nächsten Kapitel näher erläutert wird, andererseits kann man Belege auch mithilfe des Internets signieren lassen. Eine genauere Erläuterung über die Signierung und deren Varianten, befindet sich im Hauptteil im Kapitel Signierungsart wählen auf der Seite 53. Die Abspeicherung der Daten wird im Anhang in Kapitel Datenbankverwaltung auf der Seite 129 dargestellt.

B.11.1 Offline-Signierung

Unterhalb in der Auflistung SEHandlerOffline.cls auf der Seite 119 ist der Source-Code ersichtlich, welcher für die Signierung ohne Internet benötigt

wird.

```
/*
    This Class file manages the offline-signing process. Required data will be extracted of
        a certificate.
*/

::CLASS SEHandlerOffline PUBLIC
::ATTRIBUTE Base64
::ATTRIBUTE ECDSASigner
::ATTRIBUTE Decoder
::ATTRIBUTE KeyPairGenerator
::ATTRIBUTE SecureRandom
::ATTRIBUTE zertValue
::ATTRIBUTE Certificate
::METHOD init
 EXPOSE Base64  ECDSASigner Decoder KeyPairGenerator SecureRandom Certificate zertValue
 Base64 = bsf.import("java.util.Base64")
 ECDSASigner = bsf.import("com.nimbusds.jose.crypto.ECDSASigner")
 JWSSigner = bsf.loadClass("com.nimbusds.jose.JWSSigner")
 Decoder = Base64~getDecoder
 Certificate = "-----BEGIN CERTIFICATE-----
MIIEvTCCA6WgAwIBAgIEIVm59jANBgkqhkiG9w0BAQsFADCBoTELMAk
GA1UEBgwCQVQxSDBGBgNVBAoMP0EtVHJ1c3QgR2VzLiBmLiBTaWNoZX
JoZW10c3N5c3RlbWUgaW0gZWxla3RyLiBEYXRlbnZlcmtlaHIgR21iS
DEjMCEGA1UECwwaYS1zaWduLVByZW1pdW0tVGVzdC1TaWctMDIxIzAh
BgNVBAMMGmEtc2lnbi1QcmVtaXVtLVRlc3QtU2lnLTAyMB4XDTE2MDM
xMDE0NDczOVoXDTIwMDUxMDE0NDczOVowYzELMAkGA1UEBgwCQVQxGT
AXBgNVBAMMEFVJRDogQVRVMTIzNDU2NzgxFDASBgNVBAQMC0FUVTEyM
zQ1Njc4MQwwCgYDVQQqDANVSUQxFTATBgNVBAUMDDUyOTkxMjU1MjI4
NjBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABDmCCjepLRPT+MvR4CH
qycC25UNfFaFcyqCL8lwEVsUVk10bmQn290u20foEaa075geQvJokWb
TwTQqLgORYVVejggIDMIIB/zCBhAYIKwYBBQUHAQEEeDB2MEYGCCsGA
QUFBzAChjpodHRwOi8vd3d3LmEtdHJ1c3QuYXQvY2VydHMvYS1zaWd
uLVByZW1pdW0tVGVzdC1TaWctMDIuY3J0MCwGCCsGAQUFBzABhiBod
HRwOi8vb2NzcC10ZXN0LmEtdHJ1c3QuYXQvb2NzcDAOBgNVHQ8BAf8
EBAMCBsAwJwYIKwYBBQUHAQMBAf8EGDAWMAgGBgQAjkYBATAKBggrB
gEFBQcLATCBrgYDVR0fBIGmMIGjMIGgoIGdoIGahoGXbGRhcDovL2x
kYXAtdGVzdC5hLXRydXN0LmF0L291PWEtc2lnbi1QcmVtaXVtLVRlc
3QtU2lnLTAyICChTSEEtMjU2KSxvPUEtVHJ1c3QsYz1BVD9jZXJ0aWZ
pY2F0ZXJldm9jYXRpb25saXN0P2Jhc2U/b2JqZWN0Y2xhc3M9ZWlkQ
2VydGlmaWNhdGlvbkF1dGhvcml0eTAJBgNVHRMEAjAAMFkGA1UdIA
RSMFAwCAYGBACLMAEBMEQGBiooABEBCzA6MDgGCCsGAQUFBwIBFix
odHRwOi8vd3d3LmEtdHJ1c3QuYXQvZG9jcy9jcC9hLXNpZ24tUHJl
bW1lbTATBgNVHSMEDDAKgAhGBp+OQY4VvTARBgNVHQ4ECgQITEStz
GF8AcEwDQYJKoZIhvcNAQELBQADggEBADKLJSnaG7sMsX0USrzLHR
XuWPKCrOXbLfXmnmlAN2+2Pzd/wAXXL1G2dpRpyspoC4QxLVQrUQE
+WQae94lXdI7yFwVk8Y5bR+K/bY/EfNSPiwSyxXtYOnMrH+s9szH5
nCG4aRGxys5hrbxfyS8GHXqv8V6cUlddCYkwXL4XXBpSxh9UofVOA
1+mtKEtESg9ifcayV6TwBOHmzg7GZe0IY32VIpjaJQA8LQU1YDARx
+yfuxtc4KBk3o5ZaVmZ4xyX/elUTLQX8aMnT7sIbhA3+FFvaD7OiY
p+9kZL9LO9bzfS/yv0xmPjFLVQ817DwTHM1ZFGk5d6WRGBIm7fjq8
QuM=-----END CERTIFICATE-----"
 PARSE VAR Certificate "-----BEGIN CERTIFICATE-----" zertValue "-----END CERTIFICATE-----"
------------------------------------------------------------
::METHOD signString PUBLIC
 EXPOSE ECDSASigner
 USE ARG string
 key = .my.app~DatabaseHandler~getJWSSigningKey
 payl = box("STring", string~string)

 JWSObject = bsf.import("com.nimbusds.jose.JWSObject")
 JWSAlgorithm = bsf.import("com.nimbusds.jose.JWSAlgorithm")
 JWSHeader = bsf.import("com.nimbusds.jose.JWSHeader")
 Payload = bsf.import("com.nimbusds.jose.Payload")

 JWSAlgorithmParse = JWSAlgorithm~parse("ES256")
 payload = Payload~new(payl)
 JWSHeaderNew = JWSHeader~new(JWSAlgorithmParse)
 jwsSigner = ECDSASigner~new(self~getPrivateKey(key))
 jwsObject = JWSObject~new(JWSHeaderNew,payload)
 jwsObject~sign(jwsSigner)
 compactJWS = jwsObject~serialize
 PARSE VAR compactJWS jws_header "." jws_payload "." jws_signature
 RETURN jws_header jws_payload jws_signature
------------------------------------------------------------
::METHOD getPrivateKey PUBLIC
 EXPOSE Decoder
 USE ARG keyString
 PKCS8EncodedKeySpec = bsf.import("java.security.spec.PKCS8EncodedKeySpec")
 KeyFactory = bsf.loadClass("java.security.KeyFactory")~getInstance("EC")
 privateBytes = Decoder~decode(keyString)
 private = KeyFactory~generatePrivate(PKCS8EncodedKeySpec~new(privateBytes))
```

```
 RETURN private
------------------------------------------------------------
::METHOD getCertNr PUBLIC
 EXPOSE Certificate
 X509CertUtils = bsf.import("com.nimbusds.jose.util.X509CertUtils")~NEW
 CertificateParsed = X509CertUtils~parse(Certificate)
 CertificateString = CertificateParsed~toString
 PARSE VAR CertificateString "SerialNumber: [" certSerialNumber "]"
 certSerialNumber = certSerialNumber~strip()
 RETURN certSerialNumber
------------------------------------------------------------
::METHOD getZDA PUBLIC
 EXPOSE Certificate
 X509CertUtils = bsf.import("com.nimbusds.jose.util.X509CertUtils")~NEW
 CertificateParsed = X509CertUtils~parse(Certificate)
 CertificateString = CertificateParsed~toString
 PARSE VAR CertificateString  "C=" c +4 " "
 PARSE VAR c "C=" zda
 zda = zda || "1"
 RETURN ZDA
------------------------------------------------------------
::REQUIRES "BSF.CLS"
```

Auflistung 32: SEHandlerOffline.cls

B.11.2 Online-Signierung

Für die Signierung via Internet werden zusätzlich noch ein Benutzername und ein Passwort benötigt und abgefragt. Die Abfrage wird durch ein eigenes GUI getätigt, welches unterhalb in Auflistung se_login.fxml auf der Seite 120 dargestellt ist.

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

<?import com.jfoenix.controls.JFXButton?>
<?import com.jfoenix.controls.JFXPasswordField?>
<?import com.jfoenix.controls.JFXRadioButton?>
<?import com.jfoenix.controls.JFXTextField?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?language rexx?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
    Infinity" prefHeight="400.0" prefWidth="600.0" stylesheets="@DarkTheme.css" xmlns="http
    ://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <top>
      <Label alignment="CENTER" prefHeight="91.0" prefWidth="290.0" text="Security Device
          Login" BorderPane.alignment="CENTER" />
   </top>
   <center>
      <GridPane prefHeight="269.0" prefWidth="524.0" BorderPane.alignment="CENTER">
        <columnConstraints>
          <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
        </columnConstraints>
        <rowConstraints>
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
          <RowConstraints maxHeight="121.0" minHeight="10.0" prefHeight="121.0" vgrow="
              SOMETIMES" />
            <RowConstraints maxHeight="72.0" minHeight="10.0" prefHeight="46.0" vgrow="
                SOMETIMES" />
            <RowConstraints maxHeight="87.0" minHeight="10.0" prefHeight="80.0" vgrow="
                SOMETIMES" />
        </rowConstraints>
         <children>
             <JFXTextField fx:id="seLoginUsername" alignment="CENTER" promptText="Username
                 ..." />
             <JFXPasswordField fx:id="seLoginPassword" alignment="CENTER" promptText="
                 Password..." GridPane.rowIndex="1" />
             <JFXButton fx:id="seLoginButton" onAction="slotDir=arg(arg()); call seLogin
                 slotDir;" prefHeight="150.0" prefWidth="600.0" text="Login" GridPane.
                 rowIndex="3" />
```

```
            <GridPane prefHeight="33.0" prefWidth="600.0" GridPane.rowIndex="2">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />
                  <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>
               <children>
                  <JFXRadioButton fx:id="seLoginLive" style="-jfx-selected-color: #5c85d6;"
                       text="Live" GridPane.halignment="CENTER">
                     <toggleGroup>
                        <ToggleGroup fx:id="seLoginGroup" />
                     </toggleGroup></JFXRadioButton>
                  <JFXRadioButton fx:id="seLoginTest" selected="true" style="-jfx-selected-
                       color: #5c85d6;" text="Test" toggleGroup="$seLoginGroup" GridPane.
                       columnIndex="2" GridPane.halignment="CENTER" />
                  <JFXRadioButton fx:id="seLoginOffline" text="Offline" style="-jfx-selected
                       -color: #5c85d6;" toggleGroup="$seLoginGroup" GridPane.columnIndex="1"
                        GridPane.halignment="CENTER" />
               </children>
            </GridPane>
         </children>
      </GridPane>
   </center>
    <fx:script source="se_loginController.rexx" />
</BorderPane>
```

Auflistung 33: se_login.fxml

In Auflistung se_loginController.rexx auf der Seite 121 wird der dazugehörige Controller ausgewiesen.

```
/*
    This controller manages the login process between A-Trust's remote signing device or if
         needed switches to offline
    signing
*/

::ROUTINE seLogin PUBLIC
 /*@get(seLoginUsername seLoginPassword seLoginButton seLoginLive seLoginTest seLoginOffline
      )*/
 datetime = .DateTime~new
 PARSE VAR datetime "T" time
IF seLoginOffline~isSelected THEN DO
    .my.app~SEHandlerOffline = .SEHandlerOffline~new
    .my.app~offPay =.true
    .my.app~remotePay = .false
END
ELSE DO

 username = seLoginUsername~text
 password = seLoginPassword~text

    IF seLoginTest~isSelected THEN DO
        loginTest = .true
    END
    ELSE DO
        loginTest = .false
    END

    .my.app~SEHandler = .SEHandler~new
    .my.app~SEHandler~login(username, password, loginTest)

    IF .my.app~SEHandler~seIsOut THEN DO
        .my.app~stageHandler~showErrorWindow("Security Device does not work","Connection to
             security device failed. Check your internet connection","ERROR")
    END
    ELSE DO
        .my.app~SEHandler~updateSessionDetailsDB
    END
    .my.app~DatabaseHandler~updateSELogin(time)
END
 seLoginUsername~text = ""
 seLoginPassword~text = ""
 .my.app~stageHandler~closeWindow

 EXIT 0

```

```
------------------------------------------------------------
::REQUIRES "BSF.CLS"
::REQUIRES "SEHandlerRemote.CLS"
::REQUIRES "SEHandlerOffline.CLS"
```

Auflistung 34: se_loginController.rexx

In Auflistung SEHandlerRemote.CLS auf der Seite 122 ist die dazugehörige Rexx-Klasse dargestellt.

```
/*
    The SEHandler-Class will be used to interact with the A-Trust RK ONLINE security device.
*/
::CLASS SEHandler public
::ATTRIBUTE username
::ATTRIBUTE password
::ATTRIBUTE sigZert_value
::ATTRIBUTE zertstellen_value
::ATTRIBUTE zertnummer_value
::ATTRIBUTE zertSerienNr_value
::ATTRIBUTE alg_value
::ATTRIBUTE seIsOut
::ATTRIBUTE testEnv
::METHOD init
------------------------------------------------------------
::METHOD login PUBLIC
 EXPOSE username password sigZert_value zertstellen_value zertnummer_value
      zertSerienNr_value alg_value testEnv
 USE ARG username, password, testEnv

 username = username
 password = password
 testEnv = testEnv

IF testEnv = .true THEN DO
    urlStr = "https://hs-abnahme.a-trust.at/asignrkonline/v2/" || username || "/Certificate"
             -- TEST
END
ELSE DO
    urlStr = "https://www.a-trust.at/asignrkonline/v2/" || username || "/Certificate"    --
         LIVE
END
 url = .bsf~new("java.net.URL", urlStr)

 SIGNAL ON ANY
 conn = url~openConnection
 conn~setRequestMethod("GET")
 rd = .bsf~new("java.io.BufferedReader", .bsf~new("java.io.InputStreamReader", conn~
      getInputStream))
 mb = .mutableBuffer~new
 line = rd~readLine
DO WHILE line <> .nil
     mb~append(line)
     line = rd~readLine
END
 responseStr = mb~string   -- create a single string
 rd~close
 conn~disconnect
 PARSE VAR responseStr '{"' sigZert_name '":"' sigZert_value '","' zertstellen_name '":["'
      zertstellen_value '"],"' zertnummer_name '":"' zertnummer_value '","' zertSerienNr_name
       '":"' zertSerienNr_value '","' alg_name '":"' alg_value '"}'
 .my.app~SEHandler~seIsOut = .false
 .my.app~remotePay = .true
 .my.app~offPay =.false
 EXIT 0
------------------------------------------------------------
ANY:
    .my.app~SEHandler~seIsOut = .true
------------------------------------------------------------
::METHOD getCertificateHex PUBLIC
 EXPOSE zertSerienNr_value
 return zertSerienNr_value
------------------------------------------------------------
::METHOD updateSessionDetailsDB PUBLIC
 EXPOSE username password
 USE ARG testEnv = .true

IF testEnv = .true THEN DO
    urlStr = "https://hs-abnahme.a-trust.at/asignrkonline/v2/Session/" || username
END
```

```
65 ELSE DO
66     urlStr = "https://www.a-trust.at/asignrkonline/v2/Session/" || username
67 END
68  url = .bsf~new("java.net.URL", urlStr)
69  SIGNAL ON ANY
70  conn = url~openConnection
71  conn~setRequestMethod("PUT")
72  conn~setDoOutput(.true)
73  conn~setDoInput(.true)
74  conn~setUseCaches(.false)
75  conn~setAllowUserInteraction(.false)
76  conn~setRequestProperty("Content-Type", "application/json")
77
78  request = '{"password":"'|| password ||'"}'
79  out = conn~getOutputStream
80  writer = .bsf~new("java.io.OutputStreamWriter", out , "UTF-8")
81  writer~write(request)
82  writer~close
83  out~close
84  rd = .bsf~new("java.io.BufferedReader", .bsf~new("java.io.InputStreamReader", conn~
        getInputStream))
85  mb = .mutableBuffer~new
86  line = rd~readLine
87 DO WHILE line <> .nil
88    mb~append(line)
89    line = rd~readLine
90 END
91  responseStr = mb~string   -- create a single string
92  rd~close
93  conn~disconnect
94  PARSE VAR responseStr '{"' . '":"' sessionID '","' . '":"' sessionKey '"}'
95
96  .my.app~DatabaseHandler~updateSessionData(1,sessionID,sessionKey)
97
98  EXIT 0
99 ------------------------------------------------------------
100 ANY:
101   .my.app~SEHandler~seIsOut = .true
102 ------------------------------------------------------------
103 ::METHOD getZDAData PUBLIC
104  EXPOSE username
105  USE ARG testEnv = .true
106
107 IF testEnv = .true THEN DO
108     urlStr = "https://hs-abnahme.a-trust.at/asignrkonline/v2/" || username || "/ZDA"
109 END
110 ELSE DO
111     urlStr = "https://www.a-trust.at/asignrkonline/v2/" || username || "/ZDA"
112 END
113  url = .bsf~new("java.net.URL", urlStr)
114
115  SIGNAL ON ANY
116  conn = url~openConnection
117  conn~setRequestMethod("GET")
118  rd = .bsf~new("java.io.BufferedReader", .bsf~new("java.io.InputStreamReader", conn~
        getInputStream))
119  mb = .mutableBuffer~new
120  line = rd~readLine
121
122 DO WHILE line <> .nil
123    mb~append(line)
124    line = rd~readLine
125 END
126  responseStr = mb~string   -- create a single string
127  rd~close
128  conn~disconnect
129
130  PARSE VAR responseStr '{"' . '":"' zda_value '"}'
131  RETURN zda_value
132  EXIT 0
133 ------------------------------------------------------------
134 ANY:
135    .my.app~SEHandler~seIsOut = .true
136 ------------------------------------------------------------
137 ::METHOD getJWS PUBLIC
138  USE ARG headerAlgValue, Payload, testEnv = .true
139
140  Header = '{"alg":"'|| headerAlgValue ||'"}'
141  Header = Header~encodeBase64
142
143  Payload = Payload~makeString
144  Payload = Payload~encodeBase64
145
```

```
146  to_be_signed = Header || '.' || Payload
147
148  sessionIDplusKey = .my.app~DatabaseHandler~getSessionData("1")
149  PARSE VAR sessionIDplusKey sessionID sessionKey
150
151 IF testEnv = .true THEN DO
152     urlStr = "https://hs-abnahme.a-trust.at/asignrkonline/v2/Session/" || sessionID || "/
            Sign/JWS"
153 END
154 ELSE DO
155     urlStr = "https://www.a-trust.at/asignrkonline/v2/Session/" || sessionID || "/Sign/JWS"
156 END
157
158  url = .bsf~new("java.net.URL", urlStr)
159  SIGNAL ON ANY
160  conn = url~openConnection
161
162  conn~setRequestMethod("POST")
163  conn~setDoOutput(.true)
164  conn~setDoInput(.true)
165  conn~setUseCaches(.false)
166  conn~setAllowUserInteraction(.false)
167  conn~setRequestProperty("Content-Type", "application/json")
168
169  request = '{"sessionkey":"' || sessionKey || '", "' 'jws_payload":"' || payload '"}'
170  out = conn~getOutputStream
171  writer = .bsf~new("java.io.OutputStreamWriter", out , "UTF-8")
172  writer~write(request)
173  writer~close
174  out~close
175
176  rd = .bsf~new("java.io.BufferedReader", .bsf~new("java.io.InputStreamReader", conn~
         getInputStream))
177  mb = .mutableBuffer~new
178  line = rd~readLine
179 DO WHILE line <> .nil
180    mb~append(line)
181    line = rd~readLine
182 END
183  responseStr = mb~string   -- create a single string
184  rd~close
185  conn~disconnect
186
187  PARSE VAR responseStr '{"' . '":"' jws_header '.' jws_payload '.' jws_signature '"}'
188  RETURN jws_header jws_payload jws_signature
189  EXIT 0
190 ------------------------------------------------------------
191 ANY:
192     .my.app~SEHandler~seIsOut = .true
193 ------------------------------------------------------------
194 ::REQUIRES "BSF.CLS"
195 ::REQUIRES "DatabaseHandler_v2.CLS"
196 ::REQUIRES "collection.rexx"
```

Auflistung 35: SEHandlerRemote.CLS

B.12 Tabellenverwaltung

Im Kapitel Hauptbestandteile des POS-Systems auf der Seite 28 wird die Funktionalität des gesamten System erläutert. Für das Design mittels Tabellen werden einige spezielle Klassen und Methoden benötigt. Diese werden gesammelt in der Auflistung TableH.cls auf der Seite 124 dargestellt.

```
1  /*
2      This Class file manages the interaction between tables and items.
3  */
4
5  ::CLASS PropertyValueFactory PUBLIC
6  ::METHOD init
7   EXPOSE  propName
8   USE STRICT ARG propName
9  ------------------------------------------------------------
10 ::METHOD call
11  EXPOSE propName
```

```
12  | USE ARG o
13  | RETURN BsfRexxProxy(o~getValue)~send(propName)
14  | ----------------------------------------------------------
15  | ::CLASS Order  PUBLIC
16  | ::ATTRIBUTE orderID
17  | ::ATTRIBUTE tableNR
18  | ::ATTRIBUTE type
19  | ::METHOD init
20  | EXPOSE orderID tableNR type
21  | USE ARG ID, NR, type
22  | SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")
23  |
24  | orderID = SimpleStringProperty~new(ID)
25  | tableNR = SimpleStringProperty~new(NR)
26  | type = SimpleStringProperty~new(type)
27  | ----------------------------------------------------------
28  | ::CLASS OrderHandler PUBLIC
29  | ::ATTRIBUTE orderArrayList
30  | ::ATTRIBUTE rsTableViewLeft
31  | ::ATTRIBUTE rsTableColumnOID
32  | ::ATTRIBUTE rsTableColumnTNr
33  | ::ATTRIBUTE rsTableColumnRType
34  | ::METHOD init
35  | EXPOSE orderArrayList rsTableViewLeft rsTableColumnOID rsTableColumnTNr rsTableColumnRType
36  | USE ARG orderArrayList, rsTableViewLeft, rsTableColumnOID, rsTableColumnTNr,
      |     rsTableColumnRType
37  | rp=BSFCreateRexxProxy(self, , "javafx.event.EventHandler")
38  | rsTableViewLeft~setOnMouseClicked(rp)
39  | rsTableColumnOID~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("orderID"
      |     ), ,"javafx.util.Callback"))
40  | rsTableColumnTNr ~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("tableNR
      |     "),  ,"javafx.util.Callback"))
41  | rsTableColumnRType ~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("type"
      |     ),  ,"javafx.util.Callback"))
42  | rsTableViewLeft~setColumnResizePolicy(rsTableViewLeft~CONSTRAINED_RESIZE_POLICY)
43  | self ~ setItems
44  | ----------------------------------------------------------
45  | ::METHOD setItems
46  | EXPOSE rsTableViewLeft orderArrayList
47  | rsTableViewLeft~setItems(orderArrayList)
48  | ----------------------------------------------------------
49  | ::METHOD handle
50  | USE ARG event
51  |
52  | table = event~getSource
53  | IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
54  |    item = table~getSelectionModel~getSelectedItem
55  |    order = BsfRexxProxy(item)
56  |    FXCollections = bsf.import("javafx.collections.FXCollections")
57  |   .my.app~orderDetailsHandler~orderDetailsArrayList = FXCollections~observableArrayList
58  |   .my.app~DatabaseHandler~getAllProductsArrayListByID(order~orderID~get,.my.app~
      |       orderDetailsHandler)
59  |
60  |   .my.app~currentOrderID = order~orderID~get
61  |   .my.app~currentTableNR = order~tableNR~get
62  | END
63  | ----------------------------------------------------------
64  | ::CLASS OrderDetails PUBLIC
65  | ::ATTRIBUTE containsID
66  | ::ATTRIBUTE PID
67  | ::ATTRIBUTE Name
68  | ::ATTRIBUTE Price
69  | ::ATTRIBUTE Status
70  | ::METHOD INIT
71  | EXPOSE  containsID PID Name Price Status
72  | USE ARG containsID, PID, Name, Price, Status
73  |
74  | SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")
75  | SimpleIntegerProperty = bsf.import("javafx.beans.property.SimpleIntegerProperty")
76  | SimpleFloatProperty = bsf.import("javafx.beans.property.SimpleFloatProperty")
77  |
78  | PID = SimpleIntegerProperty~new(PID)
79  | Name = SimpleStringProperty~new(Name)
80  | Price = SimpleFloatProperty~new(Price)
81  | Status = SimpleStringProperty~new(Status)
82  | ----------------------------------------------------------
83  | ::CLASS OrderDetailsHandler PUBLIC
84  | ::ATTRIBUTE orderDetailsArrayList
85  | ::ATTRIBUTE rsTableViewRight
86  | ::ATTRIBUTE rsTableColumnPID
87  | ::ATTRIBUTE rsTableColumnName
88  | ::ATTRIBUTE rsTableColumnPrice
89  | ::ATTRIBUTE rsTableColumnCont
```

```
::ATTRIBUTE statusArray
::METHOD INIT
 EXPOSE orderDetailsArrayList rsTableViewRight rsTableColumnPID rsTableColumnName
      rsTableColumnPrice rsTableColumnCont statusArray
 USE ARG orderDetailsArrayList, rsTableViewRight, rsTableColumnPID, rsTableColumnName,
      rsTableColumnPrice, rsTableColumnCont
 statusArray = .array~new
 statusArray[1] = "not paid"
 statusArray[2] = "pending"
 statusArray[3] = "paid"

 rp = BSFCreateRexxProxy(self,, "javafx.event.EventHandler")
 rsTableViewRight~setOnMouseClicked(rp)

 rsTableColumnPID~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("PID"), ,
      "javafx.util.Callback"))
 rsTableColumnName~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("Name"),
       ,"javafx.util.Callback"))
 rsTableColumnPrice~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("Price"
      ), ,"javafx.util.Callback"))
 rsTableColumnCont~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("Status"
      ), ,"javafx.util.Callback"))
 rsTableViewRight~setColumnResizePolicy(rsTableViewRight~CONSTRAINED_RESIZE_POLICY)
 self~setItems
-------------------------------------------------------
::METHOD setItems
 EXPOSE rsTableViewRight orderDetailsArrayList
 rsTableViewRight~setItems(orderDetailsArrayList)
-------------------------------------------------------
::METHOD addItem PUBLIC
 EXPOSE rsTableViewRight orderDetailsArrayList
 USE ARG item
 orderDetailsArrayList~add(item)
 self~setItems
-------------------------------------------------------
::METHOD handle
 EXPOSE statusArray
 USE ARG event

 table = event~getSource
IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
    item = table~getSelectionModel~getSelectedItem
    orderDetails = BsfRexxProxy(item)
    IF orderDetails~Status~get = statusArray[1] THEN DO
        orderDetails~Status~set(statusArray[2])
        .my.app~DatabaseHandler~updateContainsStatus(statusArray[2], orderDetails~containsID
            )
    END
    ELSE IF orderDetails~Status~get = statusArray[2] THEN DO
         orderDetails~Status~set(statusArray[1])
        .my.app~DatabaseHandler~updateContainsStatus(statusArray[1], orderDetails~containsID
            )
    END
END
-------------------------------------------------------
::CLASS newOrderHandler PUBLIC
::ATTRIBUTE newOrderArrayList
::ATTRIBUTE newOrderTableView
::ATTRIBUTE newOrderProdID
::ATTRIBUTE newOrderName
::ATTRIBUTE newOrderPrice
::ATTRIBUTE newOrderQuantity
::METHOD init
 EXPOSE newOrderArrayList newOrderTableView newOrderProdID newOrderName newOrderPrice
      newOrderQuantity
 USE ARG newOrderArrayList, newOrderTableView, newOrderProdID, newOrderName, newOrderPrice,
      newOrderQuantity
 rp=BSFCreateRexxProxy(self, "javafx.beans.value.ChangeListener", "javafx.event.EventHandler
      ")
 newOrderTableView~setOnMouseClicked(rp)
 newOrderProdID~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("id"), ,"
      javafx.util.Callback"))
 newOrderName ~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("name"),  ,"
      javafx.util.Callback"))
 newOrderPrice ~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("price"),
      ,"javafx.util.Callback"))
 newOrderQuantity ~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("
      quantity"),  ,"javafx.util.Callback"))
 newOrderTableView~setColumnResizePolicy(newOrderTableView~CONSTRAINED_RESIZE_POLICY)
 self ~ setItems
-------------------------------------------------------
::METHOD setItems
 EXPOSE newOrderTableView newOrderArrayList
```

```
158  newOrderTableView~setItems(newOrderArrayList)
159 ------------------------------------------------------------
160 ::METHOD handle
161  USE ARG event
162
163  table = event~getSource
164 IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
165     item = table~getSelectionModel~getSelectedItem
166     product = BsfRexxProxy(item)
167     newOrderDetails = .newOrderDetails~new(product~id~get, product~name~get, 1)
168     item = findItemNewOrderDetails(newOrderDetails)
169     IF item = .nil THEN DO
170         IF product~quantity~get >= 1 THEN DO
171             call addNewOrderDetails newOrderDetails  -- if the product isnt already in Order
                      , add it
172             .my.app~DatabaseHandler~removeProductQuantity(product~id~get)
173         END
174         ELSE DO
175             .my.app~stageHandler~showErrorWindow("Availability Error","Not enough quantity
                      available!","ERROR")
176         END
177     END
178     ELSE DO
179         enough = incrementQuantityUntilQuant(item, product~quantity~get) -- if the product (
                item) is already there, increment its quantity
180         IF enough == .false THEN DO
181             .my.app~stageHandler~showErrorWindow("Availability Error","Not enough quantity
                      available!","ERROR")
182         END
183         ELSE DO
184             .my.app~DatabaseHandler~removeProductQuantity(product~id~get) --FEHLERQUELLE
                      DENK ICH, bin mir ned sicher
185         END
186     END
187 END
188 ------------------------------------------------------------
189 ::CLASS newOrderDetails PUBLIC
190 ::ATTRIBUTE PID
191 ::ATTRIBUTE Name
192 ::ATTRIBUTE Quantity
193 ::METHOD init
194  EXPOSE  PID Name Quantity
195  USE ARG PID, Name, Quantity
196
197  SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")
198  SimpleIntegerProperty = bsf.import("javafx.beans.property.SimpleIntegerProperty")
199
200  PID = SimpleIntegerProperty~new(PID)
201  Name = SimpleStringProperty~new(Name)
202  Quantity = SimpleIntegerProperty~new(Quantity)
203 ------------------------------------------------------------
204 ::CLASS newOrderDetailsHandler PUBLIC
205 ::ATTRIBUTE newOrderDetailsArrayList
206 ::ATTRIBUTE newOrderOrderTableView
207 ::ATTRIBUTE newOrderOrderProdID
208 ::ATTRIBUTE newOrderOrderProdName
209 ::ATTRIBUTE newOrderOrderProdQuant
210 ::METHOD init
211  EXPOSE newOrderDetailsArrayList newOrderOrderTableView newOrderOrderProdID
        newOrderOrderProdName newOrderOrderProdQuant
212  USE ARG newOrderDetailsArrayList, newOrderOrderTableView, newOrderOrderProdID,
        newOrderOrderProdName, newOrderOrderProdQuant
213  rp=BSFCreateRexxProxy(self, , "javafx.event.EventHandler")
214  newOrderOrderTableView~setOnMouseClicked(rp)
215  newOrderOrderProdID~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("PID")
        , ,"javafx.util.Callback"))
216  newOrderOrderProdName ~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("
        Name"),  ,"javafx.util.Callback"))
217  newOrderOrderProdQuant ~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("
        Quantity"),  ,"javafx.util.Callback"))
218  newOrderOrderTableView~setColumnResizePolicy(newOrderOrderTableView~
        CONSTRAINED_RESIZE_POLICY)
219  self~setItems
220 ------------------------------------------------------------
221 ::METHOD setItems
222  EXPOSE newOrderOrderTableView newOrderDetailsArrayList
223  newOrderOrderTableView~setItems(newOrderDetailsArrayList)
224 ------------------------------------------------------------
225 ::METHOD addItem PUBLIC
226  EXPOSE newOrderOrderTableView newOrderDetailsArrayList
227  USE ARG item
228  newOrderDetailsArrayList~add(item) --bridge for tableView
229  self~setItems
```

```
230 ||------------------------------------------------------
231 ||::METHOD handle
232 || USE ARG event
233 ||
234 || table = event~getSource
235 ||IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
236 ||    item = table~getSelectionModel~getSelectedItem
237 ||    product = BsfRexxProxy(item)
238 ||    quantity = product~quantity~get
239 ||    IF quantity > 1 THEN DO
240 ||        product~quantity~set(quantity-1)
241 ||    END
242 ||    ELSE DO
243 ||        table~getItems~remove(item)
244 ||    END
245 ||    .my.app~DatabaseHandler~addProductQuantity(product~PID~get)
246 || END
247 ||------------------------------------------------------
248 ||::CLASS Product PUBLIC
249 ||::ATTRIBUTE id
250 ||::ATTRIBUTE name
251 ||::ATTRIBUTE price
252 ||::ATTRIBUTE tax
253 ||::ATTRIBUTE quantity
254 ||::ATTRIBUTE lunchInd
255 ||::METHOD init
256 || EXPOSE id name price tax quantity lunchInd
257 || USE ARG id, name, price, tax, quantity, lunchInd
258 ||
259 || SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")
260 || SimpleIntegerProperty = bsf.import("javafx.beans.property.SimpleIntegerProperty")
261 ||
262 || id = SimpleStringProperty~new(id)
263 || name = SimpleStringProperty~new(name)
264 || price = SimpleStringProperty~new(price)
265 || tax = SimpleStringProperty~new(tax)
266 || quantity = SimpleStringProperty~new(quantity)
267 || lunchInd = SimpleStringProperty~new(lunchInd)
268 ||------------------------------------------------------
269 ||::CLASS ProductHandler PUBLIC
270 ||::ATTRIBUTE productArrayList
271 ||::ATTRIBUTE prodTableView
272 ||::ATTRIBUTE prodIDColumn
273 ||::ATTRIBUTE prodNameColumn
274 ||::ATTRIBUTE prodPriceColumn
275 ||::ATTRIBUTE prodTaxColumn
276 ||::ATTRIBUTE prodQuantColumn
277 ||::ATTRIBUTE prodLunchColumn
278 ||::METHOD init
279 || EXPOSE productArrayList prodTableView prodIDColumn prodNameColumn prodPriceColumn
          prodTaxColumn prodQuantColumn prodLunchColumn
280 || USE ARG productArrayList, prodTableView, prodIDColumn, prodNameColumn, prodPriceColumn,
          prodTaxColumn, prodQuantColumn, prodLunchColumn
281 || rp = BSFCreateRexxProxy(self,, "javafx.event.EventHandler")
282 || prodTableView~setOnMouseClicked(rp)
283 || prodIDColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("id"), ,"
          javafx.util.Callback"))
284 || prodNameColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("name"), ,"
          javafx.util.Callback"))
285 || prodPriceColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("price"),
          ,"javafx.util.Callback"))
286 || prodQuantColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("quantity"
          ), ,"javafx.util.Callback"))
287 || prodTaxColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("tax"), ,"
          javafx.util.Callback"))
288 || prodLunchColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("lunchInd"
          ), ,"javafx.util.Callback"))
289 || prodTableView~setColumnResizePolicy(prodTableView~CONSTRAINED_RESIZE_POLICY)
290 || self~setItems
291 ||------------------------------------------------------
292 ||::METHOD setItems
293 || EXPOSE prodTableView productArrayList
294 || prodTableView~setItems(productArrayList)
295 ||------------------------------------------------------
296 ||::METHOD handle
297 || USE ARG event
298 || table = event~getSource
299 ||IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
300 ||    product = table~getSelectionModel~getSelectedItem
301 ||    product = BsfRexxProxy(product)
302 ||END
303 ||------------------------------------------------------
304 ||::CLASS User  PUBLIC                    -- for database adding
```

```
305 ::ATTRIBUTE userID
306 ::ATTRIBUTE firstName
307 ::ATTRIBUTE lastName
308 ::ATTRIBUTE userName
309 ::ATTRIBUTE password
310 ::ATTRIBUTE rightsIndex
311 ::METHOD init
312  EXPOSE userID firstName lastName userName password rightsIndex
313  USE ARG userID, fName, lName, uName, password, rightInd
314  SimpleStringProperty = bsf.import("javafx.beans.property.SimpleStringProperty")
315
316  userID = SimpleStringProperty~new(userID)
317  firstName = SimpleStringProperty~new(fName)
318  lastName = SimpleStringProperty~new(lName)
319  userName = SimpleStringProperty~new(uName)
320  password = SimpleStringProperty~new(password)
321  rightsIndex = SimpleStringProperty~new(rightInd)
322 ------------------------------------------------------------
323 ::CLASS UserHandler PUBLIC
324 ::ATTRIBUTE usersArrayList
325 ::ATTRIBUTE userTableView
326 ::ATTRIBUTE userIDColumn
327 ::ATTRIBUTE userFNameColumn
328 ::ATTRIBUTE userLNameColumn
329 ::ATTRIBUTE userUNameColumn
330 ::ATTRIBUTE userPasswordColumn
331 ::ATTRIBUTE userRightsColumn
332 ::METHOD init
333  EXPOSE usersArrayList userTableView userIDColumn userFNameColumn userLNameColumn
       userUNameColumn userPasswordColumn userRightsColumn
334  USE ARG usersArrayList, userTableView, userIDColumn, userFNameColumn, userLNameColumn,
       userUNameColumn, userPasswordColumn, userRightsColumn
335
336  rp = BSFCreateRexxProxy(self,, "javafx.event.EventHandler")
337  userTableView~setOnMouseClicked(rp)
338  userIDColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("userID"), ,"
       javafx.util.Callback"))
339  userFNameColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("firstName
       "), ,"javafx.util.Callback"))
340  userLNameColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("lastName"
       ), ,"javafx.util.Callback"))
341  userUNameColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("userName"
       ), ,"javafx.util.Callback"))
342  userPasswordColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("
       password"), ,"javafx.util.Callback"))
343  userRightsColumn~setCellValueFactory(BsfCreateRexxProxy(.PropertyValueFactory~new("
       rightsIndex"), ,"javafx.util.Callback"))
344  userTableView~setColumnResizePolicy(userTableView~CONSTRAINED_RESIZE_POLICY)
345  self~setItems
346 ------------------------------------------------------------
347 ::METHOD setItems
348  EXPOSE userTableView usersArrayList
349  userTableView~setItems(usersArrayList)
350 ------------------------------------------------------------
351 ::METHOD handle
352  USE ARG event
353
354  table = event~getSource
355 IF table~getSelectionModel~getSelectedItem <> .nil THEN DO
356     product = table~getSelectionModel~getSelectedItem
357     product = BsfRexxProxy(product)
358 END
359 ------------------------------------------------------------
360 ::REQUIRES "DatabaseHandler_v2.CLS"
361 ::REQUIRES "BSF.CLS"
362 ::REQUIRES "collection.rexx"
```

Auflistung 36: TableH.cls

B.13 Datenbankverwaltung

Wie bereits erwähnt, wird für die Sammlung von Daten eine entsprechende Datenbank erstellt. Mit Hilfe der gelisteten Methoden in Auflistung DatabaseHandler_v2.cls auf der Seite 130, ist es möglich auf die Datenbank und

ihre Datensätze zugreifen zu können.

```
::CLASS DatabaseHandler PUBLIC
::ATTRIBUTE conn
::ATTRIBUTE DB_URL
::ATTRIBUTE DriverManager
::ATTRIBUTE PasswordHandler
::ATTRIBUTE FXCollections
::METHOD init
 EXPOSE DriverManager PasswordHandler  FXCollections
 DriverManager = bsf.import("java.sql.DriverManager")
 FXCollections = bsf.import("javafx.collections.FXCollections")
 PasswordHandler = .PasswordHandler~new
------------------------------------------------------------
::METHOD connect PUBLIC
 EXPOSE DriverManager DB_URL conn
 SIGNAL ON ANY
 conn = DriverManager~getConnection(DB_URL)
 RETURN .true
 EXIT 0

 ANY:
    Say "ERROR: Failed to connect to database."
    EXIT -1
------------------------------------------------------------
::METHOD disconnectDB PUBLIC
 EXPOSE DriverManager DB_URL conn
 conn~close()
------------------------------------------------------------
::METHOD setDBsettings PUBLIC
 EXPOSE DriverManager DB_URL conn DB_NAME
 USE ARG fileName
 DBpath = .my.app~homeDir || "DataBase"
 CALL directory  DBpath

 databaseData = .stream~new(fileName)~~open
DO line over databaseData
    PARSE VAR line "DB_NAME = " DB_NAME
    PARSE VAR line "DB_URL = " DB_URL
END
 databaseData~close
 CALL directory .my.app~homeDir
------------------------------------------------------------
::METHOD insertTables PUBLIC
 EXPOSE DriverManager DB_URL conn

 tableDesc = .array~of("DEPReceipt", "companyDetails", "user", "tax", "product", "order", "
     contains")
 tableStatements = .array~of('CREATE TABLE "DEPReceipt" ("id" INTEGER PRIMARY KEY
     AUTOINCREMENT, "certificate" TEXT,"Date_paid" TEXT,' -
 '"jws_header" TEXT, "jws_payload" TEXT, "jws_signature" TEXT);', -
 'CREATE TABLE "companyDetails" ("company_ID" INTEGER PRIMARY KEY AUTOINCREMENT, "vat_ID"
     TEXT, "name" TEXT, "telephone" TEXT, "address" TEXT, "city" TEXT,'-
 '"postcode" INTEGER, "e_mail" TEXT, "cashbox_ID" TEXT,"aes256key" TEXT, "JWSkey" TEXT,'-
 '"se_sessionid" TEXT, "se_sessionkey" TEXT, "se_lastLogin" TEXT, "turnoverCounter" INTEGER)
     ;', -
 'CREATE TABLE "user" ("user_ID"    INTEGER PRIMARY KEY AUTOINCREMENT, "first_name" TEXT, "
     last_name" TEXT, "user_name" TEXT, "password" TEXT, "rightsIndex" INTEGER);',-
 'CREATE TABLE "tax" ("tax_ID"  INTEGER PRIMARY KEY AUTOINCREMENT, "taxRate" NUMERIC);',-
 'CREATE TABLE "product" ("prod_ID" INTEGER PRIMARY KEY AUTOINCREMENT, "name"   TEXT, "price
     " NUMERIC,'-
 '"quantity"    INTEGER, "property_ind" INTEGER, "tax_ID" INTEGER, FOREIGN KEY("tax_ID")
     REFERENCES tax("tax_ID"));',-
 'CREATE TABLE "order" ("order_ID"  INTEGER PRIMARY KEY AUTOINCREMENT, "Date_paid" TEXT, "
     table_nr" INTEGER , "receipt_type" TEXT,'-
 '"jws_header" TEXT, "jws_payload" TEXT, "jws_signature" TEXT, "user_ID" INTEGER,'-
 'FOREIGN KEY("user_ID") REFERENCES user("user_ID"));',-
 'CREATE TABLE "contains" ("contains_ID"    INTEGER PRIMARY KEY AUTOINCREMENT, "status_order
     " TEXT,'-
 '"prod_ID" INTEGER, "order_ID" INTEGER,'-
 'FOREIGN KEY("order_ID") REFERENCES "order"("order_ID"), FOREIGN KEY("prod_ID") REFERENCES
     product("prod_ID"));'   -
                          )
DO i = 1 TO tableDesc~size
    tableExist = self~checkIFtableExists(tableDesc[i])
    IF \tableExist THEN DO
        prepStatement = conn~createStatement
        prepStatement~execute(tableStatements[i])
    END
    ELSE DO
        SAY pp("Table already exists! --> " tableDesc[i])
    END
END
```

```
72  SAY pp("Database setup was successfull...")
73  ------------------------------------------------------------
74  ::METHOD insertTaxValues PUBLIC
75   EXPOSE DriverManager conn
76   tax = .array~of("20%", "10%", "13%", "0%", "19%")
77   taxValueArray = .array~of("INSERT INTO  tax (taxRate) VALUES (0.20);","INSERT INTO  tax (
         taxRate) VALUES (0.10);","INSERT INTO  tax (taxRate) VALUES (0.13);",-
78                             "INSERT INTO  tax (taxRate) VALUES (0.00);","INSERT INTO  tax (
                                    taxRate) VALUES (0.19);")
79
80  DO i=1 to tax~size
81      prepStatement = conn~createStatement
82      prepStatement~execute(taxValueArray[i])
83  END
84  ------------------------------------------------------------
85  ::METHOD checkIFtableExists PUBLIC
86   EXPOSE DriverManager conn
87   USE ARG tableName
88   query = "PRAGMA table_info(["|| tableName || "]);"
89   prepStatement = conn~prepareStatement(query)
90   queryResults = prepStatement~executeQuery
91   IF queryResults~next THEN DO
92      RETURN .true
93   END
94   ELSE DO
95      RETURN .false
96   END
97  ------------------------------------------------------------
98  ::METHOD checkIFPOSAdminExists PUBLIC
99   EXPOSE DriverManager DB_URL conn
100  query = "SELECT * FROM 'user' WHERE rightsIndex = 3;"
101  prepStatement = conn~prepareStatement(query)
102  queryResults = prepStatement~executeQuery
103
104 IF queryResults~next THEN DO
105     RETURN .true
106 END
107 ELSE DO
108     RETURN .false
109 END
110 ------------------------------------------------------------
111 ::METHOD getJWSSigningKey PUBLIC
112  EXPOSE DriverManager DB_URL conn
113  query = "SELECT JWSkey FROM 'companyDetails' WHERE company_ID = 1;"
114  prepStatement = conn~prepareStatement(query)
115  queryResults = prepStatement~executeQuery
116
117 IF queryResults~next THEN DO
118     key = queryResults~getString("JWSkey")
119     RETURN key
120 END
121 ------------------------------------------------------------
122 ::METHOD checkIFCompanyDataExists PUBLIC
123  EXPOSE DriverManager DB_URL conn
124  query = "SELECT * FROM companyDetails WHERE company_ID = 1;"
125  prepStatement = conn~prepareStatement(query)
126  queryResults =prepStatement~executeQuery
127 IF queryResults~next THEN DO
128    RETURN .true
129 END
130 ELSE DO
131     RETURN .false
132 END
133 ------------------------------------------------------------
134 ::METHOD insertUser PUBLIC
135  EXPOSE DriverManager DB_URL conn PasswordHandler
136  USE ARG firstName, lastName, userName, password, rightsIndex
137
138  password = PasswordHandler~hashPassword(password)
139
140  query = "INSERT INTO 'user' (first_name, last_name, user_name, password, rightsIndex)
         VALUES (?,?,?,?,?);"
141  prepStatement = conn~prepareStatement(query)
142  prepStatement~setString(1, firstName)
143  prepStatement~setString(2, lastName)
144  prepStatement~setString(3, userName)
145  prepStatement~setString(4, password)
146  prepStatement~setString(5, rightsIndex)
147  prepStatement~execute
148 ------------------------------------------------------------
149 ::METHOD updateUserPassword PUBLIC
150  EXPOSE conn PasswordHandler
151  USE ARG userID, password
```

```
152
153  password = PasswordHandler~hashPassword(password)
154
155  query = "UPDATE 'user' set password = ? WHERE user_ID = ?;"
156  prepStatement = conn~prepareStatement(query)
157  prepStatement~setString(1, password)
158  prepStatement~setString(2, userID)
159  prepStatement~execute
160 ------------------------------------------------------------
161 ::METHOD checkPassword PUBLIC
162  EXPOSE conn PasswordHandler
163  USE ARG userID, password
164 IF userID = 0 THEN DO   -- user doent exist
165     return 0
166 END
167 ELSE DO
168     query = "SELECT password FROM 'user' WHERE user_ID = ?;"
169     prepStatement = conn~prepareStatement(query)
170     prepStatement~setString(1, userID)
171     queryResults =prepStatement~executeQuery
172     DO  queryResults~next
173         hashedPasswordDB = queryResults~getString("password")
174     END
175     check = PasswordHandler~checkPass(password, hashedPasswordDB)
176     RETURN check
177 END
178 ------------------------------------------------------------
179 ::METHOD checkIfUserNameExist PUBLIC
180  EXPOSE conn
181  USE ARG username
182  query = "SELECT * FROM 'user' WHERE user_name = ?;"
183  prepStatement = conn~prepareStatement(query)
184  prepStatement~setString(1, username)
185  queryResults = prepStatement~executeQuery
186
187 IF queryResults~next THEN DO
188     RETURN 1
189 END
190  RETURN 0
191 ------------------------------------------------------------
192 ::METHOD getUserIDByUsername PUBLIC
193  EXPOSE conn
194  USE ARG username
195
196  query = "SELECT user_ID FROM 'user' WHERE user_name = ?;"
197  prepStatement = conn~prepareStatement(query)
198  prepStatement~setString(1, username)
199  queryResults = prepStatement~executeQuery
200
201 IF queryResults~next THEN DO
202     userID = queryResults~getString("user_ID")
203     RETURN userID
204 END
205 ELSE DO
206     RETURN 0
207 END
208 ------------------------------------------------------------
209 ::METHOD getUserInfosSetActive PUBLIC
210  EXPOSE conn
211  USE ARG username
212
213  query = "SELECT user.user_ID, user.first_name, user.last_name, user.rightsIndex FROM 'user'
        WHERE user_name = ?;"
214  prepStatement = conn~prepareStatement(query)
215  prepStatement~setString(1, username)
216  queryResults = prepStatement~executeQuery
217
218 IF queryResults~next THEN DO
219     userID = queryResults~getString("user_ID")
220     firstname = queryResults~getString("first_name")
221     lastname = queryResults~getString("last_name")
222     rightsInd = queryResults~getString("rightsIndex")
223 END
224  RETURN .ActiveUser~new(userID, firstname, lastname, rightsInd)
225 ------------------------------------------------------------
226 ::METHOD deleteUser PUBLIC
227  EXPOSE conn
228  USE ARG userID
229
230  query = "DELETE FROM 'user' WHERE user_ID = ?;"
231  prepStatement = conn~prepareStatement(query)
232  prepStatement~setString(1, userID)
233  prepStatement~execute
```

```
234 ------------------------------------------------------------
235 ::METHOD getAlluserArrayList PUBLIC
236  EXPOSE conn FXCollections
237  usersArrayList = FXCollections~observableArrayList
238  query = "SELECT * FROM user;"
239
240  prepStatement = conn~prepareStatement(query)
241  queryResults =prepStatement~executeQuery
242
243 DO WHILE queryResults~next
244     user_ID = queryResults~getString("user_ID")
245     first_name = queryResults~getString("first_name")
246     last_name = queryResults~getString("last_name")
247     user_name = queryResults~getString("user_name")
248     password = queryResults~getString("password")
249     rightsIndex = queryResults~getString("rightsIndex")
250     usersArrayList~add(.User~new(user_ID, first_name, last_name,user_name, password,
            rightsIndex))
251 END
252  RETURN usersArrayList
253 ------------------------------------------------------------
254 ::METHOD updateUser PUBLIC
255  EXPOSE conn
256  USE ARG user_ID, first_name, last_name, user_name, rightsIndex
257
258  query = "UPDATE 'user' SET first_name = ?, last_name = ?, user_name = ?, rightsIndex = ?
        WHERE user_ID = ?;"
259  prepStatement = conn~prepareStatement(query)
260  prepStatement~setString(1, first_name)
261  prepStatement~setString(2, last_name)
262  prepStatement~setString(3, user_name)
263  prepStatement~setString(4, rightsIndex)
264  prepStatement~setString(5, user_ID)
265  prepStatement~execute
266 ------------------------------------------------------------
267 ::METHOD updateSELogin PUBLIC
268  EXPOSE conn
269  USE ARG time, company_ID = 1
270
271  query = "UPDATE companyDetails SET se_lastLogin = ? WHERE company_ID = ?;"
272  prepStatement = conn~prepareStatement(query)
273  prepStatement~setString(1, time)
274  prepStatement~setString(2, company_ID)
275  prepStatement~execute
276 ------------------------------------------------------------
277 ::METHOD getSELoginTime PUBLIC
278  EXPOSE conn
279  USE ARG company_ID = 1
280  query = "SELECT se_lastLogin FROM companyDetails WHERE company_ID = ?;"
281  prepStatement = conn~prepareStatement(query)
282  prepStatement~setString(1, company_ID)
283  queryResults = prepStatement~executeQuery
284 IF queryResults~next THEN DO
285     time = queryResults~getString("se_lastLogin")
286 END
287  RETURN time
288 ------------------------------------------------------------
289 ::METHOD insertOrder PUBLIC
290  EXPOSE conn
291  USE ARG table_nr, receipt_type
292  say "table_nr" table_nr
293  say "receipt_type" receipt_type
294  query = 'INSERT INTO "order" (table_nr, receipt_type, user_ID) VALUES (?,?,?);'
295  prepStatement = conn~prepareStatement(query)
296  prepStatement~setString(1, table_nr)
297  prepStatement~setString(2, receipt_type)
298  prepStatement~setString(3, .my.app~ActiveUser~userID)
299  prepStatement~execute
300 ------------------------------------------------------------
301 ::METHOD getAllOpenOrders PUBLIC
302  EXPOSE conn FXCollections
303  USE ARG table_nr, receipt_type, user_ID
304  openOrderArrayList = FXCollections~observableArrayList
305  query = "SELECT * FROM 'order' WHERE Date_paid IS NULL;"
306  prepStatement = conn~prepareStatement(query)
307  queryResults =prepStatement~executeQuery
308 DO WHILE queryResults~next
309     user_ID = queryResults~getString("order_ID")
310     table_nr = queryResults~getString("table_nr")
311     type = queryResults~getString("receipt_type")
312     openOrderArrayList~add(.Order~new(user_ID, table_nr, type))
313 END
314  RETURN openOrderArrayList
```

```
315 ------------------------------------------------------------
316 ::METHOD checkOrders PUBLIC
317  EXPOSE conn FXCollections
318  query = "SELECT * FROM 'order';"
319  prepStatement = conn~prepareStatement(query)
320  queryResults =prepStatement~executeQuery
321 IF queryResults~next THEN DO
322     RETURN .true
323 END
324 ELSE DO
325     RETURN .false
326 END
327 ------------------------------------------------------------
328 ::METHOD updateOrderJWS PUBLIC
329  EXPOSE  conn
330  USE ARG order_ID ,jws_header, jws_payload, jws_signature
331
332  query = "UPDATE 'order'  SET jws_header = ?, jws_payload = ?, jws_signature = ? WHERE
         order_ID = ?;"
333  prepStatement = conn~prepareStatement(query)
334  prepStatement~setString(1, jws_header)
335  prepStatement~setString(2, jws_payload)
336  prepStatement~setString(3, jws_signature)
337  prepStatement~setString(4, order_ID)
338  prepStatement~execute
339 ------------------------------------------------------------
340 ::METHOD getDatePayment PUBLIC
341  EXPOSE  conn
342  USE ARG order_ID
343  PARSE VAR jwsdetails jws_header jws_payload jws_signature
344
345  query = "UPDATE 'order'  SET Date_paid = datetime('now','localtime') WHERE order_ID = ?;"
346  prepStatement = conn~prepareStatement(query)
347  prepStatement~setString(1, order_ID)
348  prepStatement~execute
349
350  query = "SELECT Date_paid  FROM 'order' WHERE order_ID = ?;"
351  prepStatement = conn~prepareStatement(query)
352  prepStatement~setString(1, order_ID)
353  queryResults = prepStatement~executeQuery
354
355 IF queryResults~next THEN DO
356     datePaid = queryResults~getString("Date_paid")
357 END
358   RETURN datePaid
359 ------------------------------------------------------------
360 ::METHOD getTaxID PUBLIC
361  EXPOSE conn
362  USE ARG taxRate
363
364  query = "SELECT tax_ID FROM tax WHERE taxRate = ?;"
365  prepStatement = conn~prepareStatement(query)
366  prepStatement~setString(1, taxRate)
367  queryResults = prepStatement~executeQuery
368
369 IF queryResults~next THEN DO
370     tax_ID = queryResults~getString("tax_ID")
371 END
372  RETURN tax_ID
373 ------------------------------------------------------------
374 ::METHOD getTaxRateByProdID PUBLIC
375  EXPOSE conn
376  USE ARG ProdID
377
378  query = "SELECT taxRate FROM product INNER JOIN tax ON product.tax_ID = tax.tax_ID WHERE
         prod_ID = ?;"
379  prepStatement = conn~prepareStatement(query)
380  prepStatement~setString(1, ProdID)
381  queryResults = prepStatement~executeQuery
382
383 IF queryResults~next THEN DO
384   taxRate = queryResults~getString("taxRate")
385 END
386  RETURN taxRate
387 ------------------------------------------------------------
388 ::METHOD insertProduct PUBLIC
389  EXPOSE conn
390  USE ARG name, price, quantity, property_ind, taxPercent
391  PARSE VAR taxPercent taxRate "%"
392  taxRate = taxRate/100
393  tax_ID = self~getTaxID(taxRate)
394
395  query = "INSERT INTO product (name, price, quantity, property_ind, tax_ID) VALUES (?, ?, ?,
```

```
      ?, ?);"
 prepStatement = conn~prepareStatement(query)
 prepStatement~setString(1, name)
 prepStatement~setString(2, price)
 prepStatement~setString(3, quantity)
 prepStatement~setString(4, property_ind)
 prepStatement~setString(5, tax_ID)
 prepStatement~execute
-------------------------------------------------------------
::METHOD updateProduct PUBLIC
 EXPOSE conn
 USE ARG prod_ID, name, price, quantity, property_ind, taxPercent
 PARSE VAR taxPercent taxRate "%"
 taxRate = taxRate/100
 tax_ID = self~getTaxID(taxRate)

 query = "UPDATE product SET name = ?, price  = ? , quantity = ? , property_ind = ?, tax_ID
      = ? WHERE prod_ID = ?;"
 prepStatement = conn~prepareStatement(query)
 prepStatement~setString(1, name)
 prepStatement~setString(2, price)
 prepStatement~setString(3, quantity)
 prepStatement~setString(4, property_ind)
 prepStatement~setString(5, tax_ID)
 prepStatement~setString(6, prod_ID)
 prepStatement~execute
-------------------------------------------------------------
::METHOD updateProductLunchInd PUBLIC
 EXPOSE conn
 USE ARG prod_ID, property_ind

 query = "UPDATE product SET property_ind = ? WHERE prod_ID = ?;"
 prepStatement = conn~prepareStatement(query)
 prepStatement~setString(1, property_ind)
 prepStatement~setString(2, prod_ID)
 prepStatement~execute
-------------------------------------------------------------
::METHOD addProductQuantity PUBLIC
 EXPOSE conn
 USE ARG prod_ID
 query = "UPDATE product SET quantity = quantity + 1 WHERE prod_ID = ?;"
 prepStatement = conn~prepareStatement(query)
 prepStatement~setString(1, prod_ID)
 prepStatement~execute
-------------------------------------------------------------
::METHOD removeProductQuantity PUBLIC
 EXPOSE conn
 USE ARG prod_ID
 query = "UPDATE product SET quantity = quantity -1 WHERE prod_ID = ?;"
 prepStatement = conn~prepareStatement(query)
 prepStatement~setString(1, prod_ID)
 prepStatement~execute
-------------------------------------------------------------
::METHOD deleteProduct PUBLIC
 EXPOSE conn
 USE ARG prod_ID

 query = "DELETE FROM product WHERE prod_ID = ?;"
 prepStatement = conn~prepareStatement(query)
 prepStatement~setString(1, prod_ID)
 prepStatement~execute
-------------------------------------------------------------
::METHOD getAllProductsArrayListByID PUBLIC
 EXPOSE conn FXCollections
 USE ARG orderID, orderDetailsHandler
 orderDetailsHandler~orderDetailsArrayList = FXCollections~observableArrayList

 query = "SELECT product.prod_ID, product.name, product.price, contains.contains_ID,
      contains.status_order FROM contains INNER JOIN product ON product.prod_ID = contains.
      prod_ID WHERE order_ID = ?;"
 prepStatement = conn~prepareStatement(query)
 prepStatement~setString(1, orderID)
 queryResults =prepStatement~executeQuery

DO WHILE queryResults~next
    prod_ID = queryResults~getString("prod_ID")
    name = queryResults~getString("name")
    price = queryResults~getString("price")
    contains_ID = queryResults~getString("contains_ID")
    status_order = queryResults~getString("status_order")
    IF status_order <> "paid" THEN DO
        orderDetailsHandler~addItem(.OrderDetails~new(contains_ID,prod_ID, name, price,
            status_order))
```

```
474     END
475 END
476 ------------------------------------------------------------
477 ::METHOD getAllProductsForNewOrderByID PUBLIC
478  EXPOSE conn FXCollections
479  USE ARG order_ID, prod_ID = .nil
480  newOrderDetailsArrayList = FXCollections~observableArrayList
481
482  query = "SELECT product.prod_ID, product.name, product.quantity FROM contains INNER JOIN
          product ON product.prod_ID = contains.prod_ID WHERE order_ID = ? GROUP BY contains.
          prod_ID;"
483  prepStatement = conn~prepareStatement(query)
484  prepStatement~setString(1, order_ID)
485  queryResults =prepStatement~executeQuery
486
487 DO WHILE queryResults~next
488     prod_ID = queryResults~getString("prod_ID")
489     name = queryResults~getString("name")
490     quantity = self~countProdID(order_ID, prod_ID)
491     newOrderDetailsArrayList~add(.newOrderDetails~new(prod_ID, name, quantity))
492 END
493  RETURN newOrderDetailsArrayList
494 ------------------------------------------------------------
495 ::METHOD countProdID PUBLIC
496  EXPOSE conn
497  USE ARG order_ID, prod_ID = .nil
498
499  query = "SELECT COUNT(prod_ID) AS sum FROM contains WHERE order_ID = ? AND prod_ID = ?;"
500  prepStatement = conn~prepareStatement(query)
501  prepStatement~setString(1, order_ID)
502  prepStatement~setString(2, prod_ID)
503  queryResults = prepStatement~executeQuery
504 IF queryResults~next THEN DO
505     quantity = queryResults~getString("sum")
506 END
507 IF prod_ID = .nil THEN
508     RETURN 1
509 ELSE
510     RETURN quantity
511 ------------------------------------------------------------
512 ::METHOD getTypeOfReceipt PUBLIC
513  EXPOSE conn
514  USE ARG order_ID
515
516  query = "SELECT receipt_type FROM 'order' WHERE order_ID = ?;"
517  prepStatement = conn~prepareStatement(query)
518  prepStatement~setString(1, order_ID)
519  queryResults =prepStatement~executeQuery
520 IF queryResults~next THEN DO
521     receipt_type = queryResults~getString("receipt_type")
522 END
523  RETURN receipt_type
524 ------------------------------------------------------------
525 ::METHOD getAllProductsArrayList PUBLIC
526  EXPOSE conn FXCollections
527  productsArrayList = FXCollections~observableArrayList
528
529  query = "SELECT product.prod_ID, product.name, product.price, product.quantity, product.
          property_ind, tax.taxRate FROM product INNER JOIN tax ON product.tax_ID = tax.tax_ID;"
530  prepStatement = conn~prepareStatement(query)
531  queryResults =prepStatement~executeQuery
532 DO WHILE queryResults~next
533     prod_ID = queryResults~getString("prod_ID")
534     name = queryResults~getString("name")
535     price = queryResults~getString("price")
536     quantity = queryResults~getString("quantity")
537     lunchInd = queryResults~getString("property_ind")
538     tax = queryResults~getString("taxRate")
539     productsArrayList~add(.Product~new(prod_ID, name, price,tax, quantity, lunchInd))
540 END
541  RETURN productsArrayList
542 ------------------------------------------------------------
543 ::METHOD productAlreadyExists PUBLIC
544  EXPOSE conn
545  USE ARG prodID
546
547  query = "SELECT * FROM product WHERE prod_ID = ?;"
548  prepStatement = conn~prepareStatement(query)
549  prepStatement~setString(1, prodID)
550  queryResults =prepStatement~executeQuery
551 IF queryResults~next THEN DO
552     RETURN .true
553 END
```

```
554 ELSE DO
555     RETURN .false
556 END
557 ------------------------------------------------------------
558 ::METHOD getLunchMenuStarter PUBLIC
559  EXPOSE conn
560  query = "SELECT name FROM product WHERE property_ind = 1;"
561
562  prepStatement = conn~prepareStatement(query)
563  queryResults =prepStatement~executeQuery
564 IF  queryResults~next THEN DO
565     name = queryResults~getString("name")
566 END
567  RETURN name
568 ------------------------------------------------------------
569 ::METHOD getLunchMenuMain PUBLIC
570  EXPOSE conn
571
572  query = "SELECT name FROM product WHERE property_ind = 2;"
573  prepStatement = conn~prepareStatement(query)
574  queryResults =prepStatement~executeQuery
575 IF  queryResults~next THEN DO
576     name = queryResults~getString("name")
577 END
578  RETURN name
579 ------------------------------------------------------------
580 ::METHOD getLunchMenuDessert PUBLIC
581  EXPOSE conn
582
583  query = "SELECT name FROM product WHERE property_ind = 3;"
584  prepStatement = conn~prepareStatement(query)
585  queryResults =prepStatement~executeQuery
586 IF  queryResults~next THEN DO
587     name = queryResults~getString("name")
588 END
589  RETURN name
590 ------------------------------------------------------------
591 ::METHOD insertCompanyDetails PUBLIC
592  EXPOSE conn
593  USE ARG vat_ID, name, telephone, address, city, postcode, e_mail, cashbox_ID, aes256key,
         JWSkey
594
595  query = "INSERT INTO companyDetails (vat_ID, name, telephone, address, city, postcode,
         e_mail, cashbox_ID, aes256key, turnoverCounter, JWSkey) VALUES (?, ?, ?, ?, ?,
         ?,?,?,?,0,?);"
596  prepStatement = conn~prepareStatement(query)
597  prepStatement~setString(1, vat_ID)
598  prepStatement~setString(2, name)
599  prepStatement~setString(3, telephone)
600  prepStatement~setString(4, address)
601  prepStatement~setString(5, city)
602  prepStatement~setString(6, postcode)
603  prepStatement~setString(7, e_mail)
604  prepStatement~setString(8, cashbox_ID)
605  prepStatement~setString(9, aes256key)
606  prepStatement~setString(10, JWSkey)
607  prepStatement~execute
608 ------------------------------------------------------------
609 ::METHOD insertContains PUBLIC
610  EXPOSE conn
611  USE ARG status_order, prod_ID, order_ID
612  query = "INSERT INTO  contains (status_order, prod_ID, order_ID) VALUES (?, ?, ?);"
613  prepStatement = conn~prepareStatement(query)
614  prepStatement~setString(1, status_order)
615  prepStatement~setString(2, prod_ID)
616  prepStatement~setString(3, order_ID)
617  prepStatement~execute
618 ------------------------------------------------------------
619 ::METHOD getLastOrderID PUBLIC
620  EXPOSE conn
621  query = "SELECT seq FROM sqlite_sequence WHERE name = 'order';"
622  prepStatement = conn~prepareStatement(query)
623  queryResults =prepStatement~executeQuery
624 IF queryResults~next THEN DO
625     id = queryResults~getString("seq")
626 END
627  RETURN id
628 ------------------------------------------------------------
629 ::METHOD updateContainsStatus PUBLIC
630  EXPOSE conn
631  USE ARG status_order, contains_ID
632
633  query = "UPDATE contains SET status_order = ? WHERE contains_ID = ?;"
```

```
634  prepStatement = conn~prepareStatement(query)
635  prepStatement~setString(1, status_order)
636  prepStatement~setString(2, contains_ID)
637  prepStatement~execute
638 ------------------------------------------------------------
639 ::METHOD removeOrdersFromDB PUBLIC
640  EXPOSE conn
641  USE ARG order_ID
642  query = "DELETE FROM contains WHERE order_ID = ?;"
643  prepStatement = conn~prepareStatement(query)
644  prepStatement~setString(1, order_ID)
645  prepStatement~execute
646 ------------------------------------------------------------
647 ::METHOD updateSessionData PUBLIC
648  EXPOSE conn
649  USE ARG companyID, sessionID, sessionKey
650
651  query = "UPDATE companyDetails SET se_sessionid = ?, se_sessionkey = ? WHERE company_ID =
         ?;"
652  prepStatement = conn~prepareStatement(query)
653  prepStatement~setString(1, sessionID)
654  prepStatement~setString(2, sessionKey)
655  prepStatement~setString(3, companyID)
656  prepStatement~execute
657 ------------------------------------------------------------
658 ::METHOD updateTurnoverValue PUBLIC
659  EXPOSE conn
660  USE ARG companyID, value
661
662  query = "UPDATE companyDetails SET turnoverCounter = ? where company_ID = ?;"
663  prepStatement = conn~prepareStatement(query)
664  prepStatement~setString(1, value)
665  prepStatement~setString(2, companyID)
666  prepStatement~execute
667 ------------------------------------------------------------
668 ::METHOD getSessionData PUBLIC
669  EXPOSE conn
670  USE ARG companyID
671
672  query = "SELECT se_sessionid, se_sessionkey FROM companyDetails WHERE company_ID = ?;"
673  prepStatement = conn~prepareStatement(query)
674  prepStatement~setString(1, companyID)
675  queryResults =prepStatement~executeQuery
676 IF queryResults~next THEN DO
677     sessionID = queryResults~getString("se_sessionid")
678     sessionKey = queryResults~getString("se_sessionkey")
679 END
680  RETURN sessionID sessionKey
681 ------------------------------------------------------------
682 ::METHOD getCompanyData PUBLIC
683  EXPOSE conn
684  USE ARG companyID = 1
685  company = .table~new
686  query = "SELECT company_ID, vat_ID, name, telephone, address, e_mail, city, postcode,
         aes256key, cashbox_ID, JWSkey FROM companyDetails WHERE company_ID =  ?;"
687  prepStatement = conn~prepareStatement(query)
688  prepStatement~setString(1, companyID)
689  queryResults =prepStatement~executeQuery
690 IF queryResults~next THEN DO
691     company["company_ID"] = queryResults~getString("company_ID")
692     company["vat_ID"] = queryResults~getString("vat_ID")
693     company["name"] = queryResults~getString("name")
694     company["telephone"] = queryResults~getString("telephone")
695     company["address"] = queryResults~getString("address")
696     company["e_mail"] = queryResults~getString("e_mail")
697     company["city"] = queryResults~getString("city")
698     company["postcode"] = queryResults~getString("postcode")
699     company["aes256key"] = queryResults~getString("aes256key")
700     company["cashbox_ID"] = queryResults~getString("cashbox_ID")
701     company["JWSkey"] = queryResults~getString("JWSkey")
702 END
703   RETURN company
704 ------------------------------------------------------------
705 ::METHOD updateCompanyData PUBLIC
706  EXPOSE conn
707  USE ARG vat_ID, name, telephone, address, city, postcode, e_mail, cashbox_ID, aes256key,
         company_ID = 1
708
709  query = "UPDATE companyDetails SET vat_ID = ?, 'name' = ?, telephone = ?, 'address' = ?,
         city = ?, postcode = ?, e_mail = ?, cashbox_ID = ?, aes256key = ? where company_ID = ?;
         "
710  prepStatement = conn~prepareStatement(query)
711  prepStatement~setString(1, vat_ID)
```

```
712  prepStatement~setString(2, name)
713  prepStatement~setString(3, telephone)
714  prepStatement~setString(4, address)
715  prepStatement~setString(5, city)
716  prepStatement~setString(6, postcode)
717  prepStatement~setString(7, e_mail)
718  prepStatement~setString(8, cashbox_ID)
719  prepStatement~setString(9, aes256key)
720  prepStatement~setString(10, company_ID)
721  prepStatement~execute
722 ------------------------------------------------------------
723 ::METHOD getTurnoverValue PUBLIC
724  EXPOSE conn
725  USE ARG companyID
726
727  query = "SELECT turnoverCounter FROM companyDetails WHERE company_ID = ?;"
728  prepStatement = conn~prepareStatement(query)
729  prepStatement~setString(1, companyID)
730  queryResults =prepStatement~executeQuery
731 IF  queryResults~next THEN DO
732     turnover = queryResults~getString("turnoverCounter")
733 END
734  RETURN turnover
735 ------------------------------------------------------------
736 ::METHOD getLastJWSData PUBLIC
737  EXPOSE conn
738
739  query = "SELECT jws_header, jws_payload, jws_signature FROM 'order' ORDER BY Date_paid desc
         LIMIT 2;"
740  prepStatement = conn~prepareStatement(query)
741  queryResults = prepStatement~executeQuery
742 IF  queryResults~next THEN DO
743     jws_header = queryResults~getString("jws_header")
744     jws_payload = queryResults~getString("jws_payload")
745     jws_signature = queryResults~getString("jws_signature")
746 END
747 IF jws_header = .nil THEN DO
748     RETURN .nil
749 END
750  jws_compact = jws_header || "." || jws_payload || "." || jws_signature
751  RETURN jws_compact
752 ------------------------------------------------------------
753 ::METHOD insertDEPReceipt PUBLIC
754  EXPOSE conn
755  USE ARG certificate,datetime,  jws_header, jws_payload, jws_signature
756  query = "INSERT INTO  DEPReceipt (certificate, Date_paid, jws_header, jws_payload,
       jws_signature) VALUES (?, ?, ?, ?, ?);"
757  prepStatement = conn~prepareStatement(query)
758  prepStatement~setString(1, certificate)
759  prepStatement~setString(2, datetime)
760  prepStatement~setString(3, jws_header)
761  prepStatement~setString(4, jws_payload)
762  prepStatement~setString(5, jws_signature)
763  prepStatement~execute
764 ------------------------------------------------------------
765 ::METHOD getDEP PUBLIC
766  EXPOSE conn FXCollections
767  receiptArrayList = FXCollections~observableArrayList
768
769  query = "SELECT * FROM 'DEPReceipt' ORDER BY Date_paid asc ;"
770  prepStatement = conn~prepareStatement(query)
771  queryResults = prepStatement~executeQuery
772 DO WHILE  queryResults~next
773     certificate = queryResults~getString("certificate")
774     jws_header = queryResults~getString("jws_header")
775     jws_payload = queryResults~getString("jws_payload")
776     jws_signature = queryResults~getString("jws_signature")
777     receiptArrayList~add(.receiptDEP~new(certificate,jws_header, jws_payload, jws_signature
          ))
778 END
779  RETURN receiptArrayList
780 ------------------------------------------------------------
781 ::REQUIRES "BSF.CLS"
782 ::REQUIRES "PasswordHandler.CLS"
783 ::REQUIRES "TableH.CLS"
784 ::REQUIRES "collection.rexx"
```

Auflistung 37: DatabaseHandler_v2.cls

B.14 Cascading Style Sheet

Die Auflistung DarkTheme.css auf der Seite 140 dient dazu, das Design eines GUIs anpassen zu können, ohne dass dies neuerlich entwickelt werden muss. Dazu wird die entsprechenden Komponenten innerhalb der CSS-Datei angegeben, benannt und definiert.

```
1
2  .background {
3      -fx-background-color: #455363;
4  }
5
6  .label {
7      -fx-font-size: 13pt;
8      -fx-font-family: "Segoe UI Semibold";
9      -fx-text-fill: black;
10     -fx-opacity: 0.6;
11 }
12
13 .label-bright {
14     -fx-font-size: 11pt;
15     -fx-font-family: "Segoe UI Semibold";
16     -fx-text-fill: white;
17     -fx-opacity: 1;
18 }
19
20 .label-header {
21     -fx-font-size: 15pt;
22     -fx-font-family: "Segoe UI Light";
23     -fx-text-fill: white;
24     -fx-opacity: 1;
25 }
26
27 .table-view {
28     -fx-base: #455363;
29     -fx-control-inner-background: #455363;
30     -fx-background-color: #455363;
31     -fx-table-cell-border-color: transparent;
32     -fx-table-header-border-color: transparent;
33     -fx-padding: 5;
34 }
35
36 .table-view .column-header-background {
37     -fx-background-color: transparent;
38 }
39
40 .table-view .column-header, .table-view .filler {
41     -fx-size: 35;
42     -fx-border-width: 0 0 1 0;
43     -fx-background-color: transparent;
44     -fx-border-color:
45         transparent
46         transparent
47         derive(-fx-base, 80%)
48         transparent;
49     -fx-border-insets: 0 10 1 0;
50 }
51
52 .table-view .column-header .label {
53     -fx-font-size: 12pt;
54     -fx-font-family: "Segoe UI Light";
55     -fx-text-fill: white;
56     -fx-alignment: center-left;
57     -fx-opacity: 1;
58 }
59 .table-row-cell {
60     -fx-cell-size: 35px;
61     -fx-font-size: 15px;
62 }
63
64 .jfx-radio-button {
65     -fx-selected-color: #455363;
66 }
67
68 .table-view:focused .table-row-cell:filled:focused:selected {
69     -fx-background-color: -fx-focus-color;
70 }
71
72 .split-pane:horizontal > .split-pane-divider {
```

```
73      -fx-border-color: transparent #455363 transparent #455363;
74      -fx-background-color: transparent, derive(#455363,20%);
75  }
76
77  .split-pane {
78      -fx-padding: 1 0 0 0;
79  }
80
81  .menu-bar {
82      -fx-background-color: derive(#455363,40%);
83  }
84
85  .context-menu {
86      -fx-background-color: derive(#455363,50%);
87  }
88
89  .menu-bar .label {
90      -fx-font-size: 12pt;
91      -fx-font-family: "Segoe UI Light";
92      -fx-text-fill: white;
93      -fx-opacity: 0.9;
94  }
95
96  .text-field {
97      -fx-font-size: 11pt;
98      -fx-font-family: "Segoe UI Semibold";
99  }
100
101 .button {
102     -fx-padding: 5 22 5 22;
103     -fx-border-color: #e2e2e2;
104     -fx-border-width: 2;
105     -fx-background-radius: 0;
106     -fx-background-color: #455363;
107     -fx-font-family: "Segoe UI", Helvetica, Arial, sans-serif;
108     -fx-font-size: 11pt;
109     -fx-text-fill: #d8d8d8;
110     -fx-background-insets: 0 0 0 0, 0, 1, 2;
111 }
112
113 .button:hover {
114     -fx-background-color: #697e96;
115 }
116
117 .button:pressed, .button:default:hover:pressed {
118   -fx-background-color: white;
119   -fx-text-fill: #455363;
120 }
121
122 .button:focused {
123     -fx-border-color: white, white;
124     -fx-border-width: 1, 1;
125     -fx-border-style: solid, segments(1, 1);
126     -fx-border-radius: 0, 0;
127     -fx-border-insets: 1 1 1 1, 0;
128 }
129
130 .button:disabled, .button:default:disabled {
131     -fx-opacity: 0.4;
132     -fx-background-color: #455363;
133     -fx-text-fill: white;
134 }
135
136 .button:default {
137     -fx-background-color: -fx-focus-color;
138     -fx-text-fill: #ffffff;
139 }
140
141 .button:default:hover {
142     -fx-background-color: derive(-fx-focus-color,30%);
143 }
144
145 .label-lunch {
146
147     -fx-font-size: 15pt;
148     -fx-font-family: "Roboto Medium";
149     -fx-text-fill: black;
150     -fx-opacity: 0.6;
151
152 }
```

Auflistung 38: DarkTheme.css

B.15 DEP-Export

Zeigt einen Beispiel-Export, der für das Bundesministerium für Finanzen angefertigt werden muss.

```
{ "Belege-Gruppe":[
    {"Signaturzertifikat":
"MIIEvTCCA6WgAwIBAgIEIVm59jANBgkqhkiG9w0BAQsFA
DCBoTELMAkGA1UEBgwCQVQxSDBGBgNVBAoMP0EtVHJ1c3Q
gR2VzLiBmLiBTaWNoZXJoZWl0c3N5c3RlbWUgaW0gZWxla
3RyLiBEYXRlbnZlcmtlaHIgR21iSDEjMCEGA1UECwwaYS1
zaWduLVByZW1pdW0tVGVzdC1TaWctMDIxIzAhBgNVBAMMG
mEtc2lnbi1QcmVtaXVtLVRlc3QtU2lnLTAyMB4XDTE2MDM
xMDE0NDczOVoXDTIwMDUxMDE0NDczOVowYzELMAkGA1UEB
gwCQVQxGTAXBgNVBAMMEFVJRDogQVRVMTIzNDU2NzgxFDA
SBgNVBAQMC0FUVTEyMzQ1Njc4MQwwCgYDVQQqDANVSUQxF
TATBgNVBAUMDDUyOTkxMjU1MjI4NjBZMBMGByqGSM49AgE
GCCqGSM49AwEHA0IABDmCCjepLRPT+MvR4CHqycC25UNfF
aFcyqCL8lwEVsUVk10bmQn290u20foEaa075geQvJokWbT
wTQqLgORYVVejggIDMIIB\/zCBhAYIKwYBBQUHAQEEeDB2
MEYGCCsGAQUFBzAChjpodHRwOi8vd3d3LmEtdHJ1c3QuYX
QvY2VydHMvYS1zaWduLVByZW1pdW0tVGVzdC1TaWctMDIu
Y3J0MCwGCCsGAQUFBzABhiBodHRwOi8vb2NzcC10ZXN0Lm
EtdHJ1c3QuYXQvb2NzcDAOBgNVHQ8BAf8EBAMCBsAwJwYI
KwYBBQUHAQMBAf8EGDAWMAgGBgQAjkYBATAKBggrBgEFBQ
cLATCBrgYDVR0fBIGmMIGjMIGgoIGdoIGahoGXbGRhcDov
L2xkYXAtdGVzdC5hLXRydXN0LmF0L291PWEtc2lnbi1Qcm
VtaXVtLVRlc3QtU2lnLTAyIChTSEEtMjU2KSxvPUEtVHJ1
c3QsYz1BVD9jZXJ0aWZpY2F0ZXJldm9jYXRpb25saXN0P2
Jhc2U\/b2JqZWN0Y2xhc3M9ZWlkQ2VydGlmaWNhdGlvbkF
1dGhvcml0eTAJBgNVHRMEAjAAMFkGA1UdIARSMFAwCAYGB
ACLMAEBMEQGBiooABEBCzA6MDgGCCsGAQUFBwIBFixodHR
wOi8vd3d3LmEtdHJ1c3QuYXQvZG9jcy9jcC9hLXNpZ24tU
HJlbWl1bTATBgNVHSMEDDAKgAhGBp+OQY4VvTARBgNVHQ4
ECgQITEStzGF8AcEwDQYJKoZIhvcNAQELBQADggEBADKLJ
SnaG7sMsX0USrzLHRXuWPKCrOXbLfXmnmlAN2+2Pzd\/wA
XXL1G2dpRpyspoC4QxLVQrUQE+WQae94lXdI7yFwVk8Y5b
R+K\/bY\/EfNSPiwSyxXtYOnMrH+s9szH5nCG4aRGxys5h
rbxfyS8GHXqv8V6cUlddCYkwXL4XXBpSxh9UofVOA1+mtK
EtESg9ifcayV6TwBOHmzg7GZe0IY32VIpjaJQA8LQU1YDA
Rx+yfuxtc4KBk3o5ZaVmZ4xyX\/elUTLQX8aMnT7sIbhA3
+FFvaD70iYp+9kZL9LO9bzfS\/yv0xmPjFLVQ817DwTHM1
```

```
ZFGk5d6WRGBIm7fjq8QuM=",
      "Zertifizierungsstellen":[
"MIIEATCCAumgAwIBAgIEOWntwTANBgkqhkiG9w0BAQUFA
DCBlTELMAkGA1UEBhMCQVQxSDBGBgNVBAoMP0EtVHJ1c3Q
gR2VzLiBmLiBTaWNoZXJoZWl0c3N5c3RlbWUgaW0gZWxla
3RyLiBEYXRlbnZlcmtlaHIgR21iSDEdMBsGA1UECwwUQS1
UcnVzdC1UZXN0LVF1YWwtMDIxHTAbBgNVBAMMFEEtVHJ1c
3QtVGVzdC1RdWFsLTAyMB4XDTE0MTEyNDE0NDkxN1oXDTI
0MTExODEzNDkxN1owgaExCzAJBgNVBAYTAkFUMUgwRgYDV
QQKDD9BLVRydXN0IEdlcy4gZi4gU2ljaGVyaGVpdHNzeXN
0ZW1lIGltIGVsZWt0ci4gRGF0ZW52ZXJrZWhyIEdtYkgxI
zAhBgNVBAsMGmEtc2lnbi1QcmVtaXVtLVRlc3QtU2lnLTA
yMSMwIQYDVQQDDBphLXNpZ24tUHJlbWl1bS1UZXN0LVNpZ
y0wMjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggE
BANwJSfWpRaziThddTTup72CltlXl8oc7HQoK2SWsYQwZG
Ad5nJZbwbI4K8VFKlNnK72Zl8UhmQ2FxhzS6u+Q+qEzJOM
2xTfA2NB6A9\/KFpTJXUjvCHgRvW16EEF9YpYXxKTSK+Qr
YCXAC5rL6SuYOzgA7Q1ivq+zLbyXxroux2zVEBIiaBGpZh
OHGDFJk6h\/4QelIqzd2TIDCRzvhmLDVmhqX2C1NQb5kZu
MgrxxOhG5Cr1F8solkwyu43JiM+apY4bZJVQBwi9ATBMz5
tfdoLRslQy1BCQ4X+b6u\/2856gucU+1e\/wa5pB9Ff0eP
+xy+j2DZOXLNd8m\/IQvnshjNusCAwEAAaNLMEkwDwYDVR
0TAQH\/BAUwAwEB\/zARBgNVHQ4ECgQIRgafjkGOFb0wEw
YDVR0jBAwwCoAIQg8xWXA9iecwDgYDVR0PAQH\/BAQDAgE
GMA0GCSqGSIb3DQEBBQUAA4IBAQBq\/owq5eGvhxegchLv
nMjPnE9gTYIHEvMq8DN6h2J7pTEhKG2o09LLn\/pNHWRjK
ENU\/LqZBIAJ5zebm5XqzB631BYcuu1abyPFfpMdAL9X4z
FuDvg9EGaTir2c81XaBYzVSLN7fxmNLKSmMwUt0JQQyqpe
3V9iyoBE\/WcQyKmKaEp7mCZsGFBm6KmJgqD6TPb7X9bWU
r3yx6Z5gek71f3vQi69m1x811azXlxu1i\/XFnVpzxkrKR
XJWC+wnQRxXmU7YnMzYPOA7UOpUG6J+7tYi29hY3EpMgyX
M\/T\/BL5MdyzBefbPVzLHng5zVaXNpO0ENCrlUyi1m3Yd
\/7QPDdJM",
"MIID4DCCAsigAwIBAgIEOWntvzANBgkqhkiG9w0BAQUFA
DCBlTELMAkGA1UEBhMCQVQxSDBGBgNVBAoMP0EtVHJ1c3Q
gR2VzLiBmLiBTaWNoZXJoZWl0c3N5c3RlbWUgaW0gZWxla
3RyLiBEYXRlbnZlcmtlaHIgR21iSDEdMBsGA1UECwwUQS1
UcnVzdC1UZXN0LVF1YWwtMDIxHTAbBgNVBAMMFEEtVHJ1c
3QtVGVzdC1RdWFsLTAyMB4XDTE0MTEyNDE0NDc0N1oXDTI
0MTExODEzNDc0N1owgZUxCzAJBgNVBAYTAkFUMUgwRgYDV
QQKDD9BLVRydXN0IEdlcy4gZi4gU2ljaGVyaGVpdHNzeXN
```

```
0ZW1lIGltIGVsZWt0ci4gRGF0ZW52ZXJrZWhyIEdtYkgxH
TAbBgNVBAsMFEEtVHJ1c3QtVGVzdC1RdWFsLTAyMR0wGwY
DVQQDDBRBLVRydXN0LVRlc3QtUXVhbC0wMjCCASIwDQ
YJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANMBok2fNNtI
Ecf7Sw47vprkUeti6Y64Rc5rrAjh7cGwo4Jp5LyfvEVdv9
AMNiuOX7ywd1xW99UZWtZ8MzXvWM5M6trLkeBYnCukwc9D
qawXcuXXCYwgTuisFTmYO6GVJNr1iE\/LJdSKbu5AVDS3F
wXixqyJkjv\/xWIwU4q86oATW8++8wb6Lu+fQlhBbn3Kqp
avt6K+lwWSCb+8vIhB47IlKhJZwGqXfGV9l9dDgKYUbZiv
3BBa+MRBUTvIcahEKz8hG2E8W4EgCwzISMpeStJtRHo\/t
JnA90KfSBTcz0txrxpHwqFgKwJvgW6nIjY1Sv5MfY5YJiE
Wv0d7UUkvlScCAwEAAaM2MDQwDwYDVR0TAQH\/BAUwAwEB
\/zARBgNVHQ4ECgQIQg8xWXA9iecwDgYDVR0PAQH\/BAQD
AgEGMA0GCSqGSIb3DQEBBQUAA4IBAQApqSvkQyfbO2yDWe
wHwo1Zl32uGz41KMP5FYtA3BIcqh89paHwrW9KfcrybdUI
neVz4iSnpyrDrS4LavfP8h\/Hl1kRmVZRUBsOJRvqc1fiC
2B6IJRHrmayb\/DbXuyoOsk7Sr8M9xtAD3SzJCRkBrtjz\
/U\/xQdU9TfV9SQyPN3qI+SR25\/LRZDhOKcIFJduVpTYz
bnKTIkl3OUrHXVq5xddxX6XP8bUjT+SqGiDf15H6N5flNB
svolMSo0OoQXFiDuY33frQSrSbHbA2p\/MptwxA8JgGh4l
rbgZZxjTvpO1wATBLDc3wGZkNuy+tNrrHAmE08B7fiExUL
HxzfaZEWSF"],
      "Belege-Kompakt":[
        "eyJhbGciOiJFUzI1NiJ9.X1IxLUFUMV9DQVNI
Qk9YMV8xXzIwMjEtMDQtMzBUMTI6NDY6NDFfMCwwMF8wLD
AwXzAsMDBfMCwwMF8wLDAwX0s3aGg2K1E9XzIxNTliOWY2
X09vcHg0SmxhMFg0PQ.SOV8F7VKV-ssllIL6_yYSJ7lPkS
EQGr_wZtrzg5W7GSSZDSVeni9_SzSczUX3ZSFogWdeYzB6
_2cG4LxNcHfLw",
        "eyJhbGciOiJFUzI1NiJ9.X1IxLUFUMV9DQVNI
Qk9YMV8yXzIwMjEtMDQtMzBUMTI6NDg6NTFfMSwwMF8wLD
AwXzIsNTBfMCwwMF8yLDAwXzkzREladDA9XzIxNTliOWY2
X09vcHg0SmxhMFg0PQ.NEm68G3-zoTCcCvaTLutJTkUXuD
0ELdH7BjTVvMk4PV246j-_J3MLuJs5k5Mojd93ROg2smXT
pibE6s9efp8Uw"
      ]
    }
  ]
}
```

Auflistung 39: DEPExport.json