Entwicklung eines Lagerabrufs- und Verwaltungssystems mit Datenbank und Android-Anwendung zum Einsatz in der Gastronomie

Studienarbeit

3. Studienjahr

Modul T3200

des Studienganges Mechatronik

an der Dualen Hochschule Baden-Württemberg

am Standort Stuttgart

von

Marvin Mai und Daniel Schifano

Bearbeitungszeitraum	12 Wochen
Betreuer der dualen Hochschule	Prof. Dr.-Ing. Johannes Moosheimer

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Ehrenverantwortliche Erklärung

Gemäß § 5 (2) der „Studien- und Prüfungsordnung DHBW Technik“ vom 06. November 2013.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen, als die angegebenen Quellen und Hilfsmittel verwendet.

Stuttgart, 11.05.2018	
Ort, Datum	Unterschrift

Stuttgart, 11.05.2018	
Ort, Datum	Unterschrift

DHBW
Duale Hochschule
Baden-Württemberg
Stuttgart

Inhaltsverzeichnis

Abbildungsverzeichnis 1
1 Anforderungen 1
1.1 Aufgabenstellung 1
2 Pflichtenheft 2
2.1 Neue Funktionen 2
2.2 Anwendungsfälle 4
2.2.1 Anwendungsfalldiagramm Android-Anwendung 4
2.2.2 Anwendungsfalldefinition der Android- Anwendung 4
2.2.3 Anwendungsfalldiagramm Datenbankmanager 8
2.2.4 Anwendungsfalldefinition des Datenbankmanagers 9
3 Entwurf 13
3.1 Beschreibung der Softwarekomponenten 13
3.2 Beschreibung von Programmabläufen und Funktionen 13
3.2.1 Umsetzung der Netzwerkkommunikation 13
3.2.2 Umsetzung des Exception-Handlings 14
3.2.3 Authentifizierung mit Login-Daten 14
3.3 Produktübersicht: Aufbau des Kassensystems und Einordnung der Software-Komponenten 16
3.4 Produktdaten: Entwurf der Datenbankarchitektur (6) 17
3.5 Workflows: Zusammenhänge zwischen Anwendungsfällen und Datenbankinhalten 18
3.5.1 Tische 18
3.5.2 Bestellungen 19
3.5.3 Artikelverwaltung 20
3.5.4 Warenein- und -ausgänge 21
3.5.5 Bedienung 22
3.5.6 Anmeldedaten 23
3.6 Benutzeroberfläche 24

3.6.1 Benutzeroberfläche der Android-Anwendung 24
3.6.2 Benutzeroberfläche des Datenbank-Managers 26
4 Implementierung 27
4.1 Implementierung des Datenbank-Systems 27
4.1.1 Klasse DatabaseService_Interface 27
4.1.2 Klasse RestApiController 39
4.2 Implementierung der Android-Anwendung 46
4.2.1 Klasse TableSelectFragment 46
4.2.2 Klasse MainActivity 47
5 Test 51
5.1 Test des Datenbank-Systems 51
5.1.1 Abrufen von Datenbankinhalten 51
5.1.2 Hinzufügen von Datenbankinhalten 60
5.1.3 Hinzufügen von Login-Daten 67
5.1.4 Bearbeiten von Datenbankinhalten 71
5.1.5 Löschen von Datenbankinhalten 78
5.1.6 Ausdrucken einer Bestellung 84
5.2 Test der Android-Anwendung 86
6 Fazit 106
7 Literaturverzeichnis 107
8 Anhang 108
8.1 Installationsanweisung 108
8.1.1 Datenbank-System 108
8.1.2 Android Applikation 110
9 Testdokumentation 111
9.1 Android-Anwendung 111

Abbildungsverzeichnis

Abbildung 1 Anwendungsfalldiagramm Android Anwendung 4
Abbildung 2 Anwendungsfalldiagramm Datenbankmanager 8
Abbildung 3 Systemübersicht des Projekts 16
Abbildung 4 Entwurf der überarbeiteten Datenbank 17
Abbildung 5 Workflow der Tische 18
Abbildung 6 Workflow der Bestellungen 19
Abbildung 7 Workflow der Artikelverwaltung 20
Abbildung 8 Workflow der Warenein- und -ausgänge 21
Abbildung 9 Workflow der Bedienungen 22
Abbildung 10 Workflow der Anmeldedaten 23
Abbildung 11 GUI Android-Anwendung 24
Abbildung 12 GUI Android-Anwendung Artikel als produziert markieren 25
Abbildung 13 Gui Datenbank-Manager 26

1 Anforderungen

1.1 Aufgabenstellung

Laut Aufgabenstellung soll ein Lagerabrufs- und Verwaltungssystem mit Datenbank und Android-Anwendung entwickelt werden. Das System besteht aus einem zentralen Computer, der über ein Netzwerk mit beliebig vielen Handheld-Geräten verbunden werden kann. Das Netzwerk wird von einem Router via Wireless Local Area Network (WLAN) zur Verfügung gestellt.

Die Grundstruktur des Systems ist bereits aufgebaut und getestet.

Durch eine Optimierung der Datenbank und der Schnittstelle zu den Handheld-Geräten, soll es zu einem Ressourcen-effizienterem Austausch von Daten zwischen Datenbank und Handheld-Geräten kommen. Die Optimierung wird zusätzlich zur Umsetzung verschiedener Erweiterungen benötigt.

Ebenfalls soll das System dahingehen erweitert werden, dass ein Login für verschiedene Mitarbeiter möglich ist. Diese Login Daten müssen auf der Datenbank gegengeprüft werden. Sind diese dort hinterlegt, kann der Mitarbeiter auf das System zugreifen. So wird sichergestellt, dass nur befugtes Personal mit dem System arbeiten kann.

Zusätzliche Erweiterungen des Lagersystems sind im Pflichtenheft vermerkt.

Die Entwicklung der Software basiert auf dem V-Modell, angelehnt an „Using V Models for Testing“ (1).

2 Pflichtenheft

Das Pflichtenheft wurde angelehnt an die Vorlage „Gliederungsschema eines Pflichtenheftes (2. Auflage)“ von der Universität Rostock erstellt. (2)

2.1 Neue Funktionen

Die folgenden Funktionen sollen im Rahmen dieser Studienarbeit erarbeitet und im bisherigen System implementiert werden:

Nr	Bezeichnung	Beschreibung	Betroffene Systeme und deren Änderung (APP: Android-App, DBM: Datenbankmanager, DBS: Database-System)
1	Mitarbeiter-login	• Jeder Service-Mitarbeiter muss sich in der App mit einem Login (Name und Passwort) identifizieren. • Passwort und Name werden im Datenbankmanager definiert. • Die Zugangsdaten werden in der Datenbank abgelegt.	• APP: In einem neuen Login-Fenster soll das Eingeben von den Login-Daten und das Einloggen durch eine Abfrage beim Server möglich sein. Die Verifizierung soll durch den http-Headers bei jeder Anfrage umgesetzt werden. • DBS: Der Server muss so angepasst werden, dass alle Anfragen durch eine Verifizierung im http-Header eindeutig überprüft werden können. • DBM: Ein neuer Tab muss erstellt werden, in dem das Anlegen von Login-Daten möglich ist.
2	Artikel-kategorien	• Alle Artikel sollen Kategorien zugeordnet werden. (Getränke, Hauptspeisen...) • In der App sollen die Artikel in Kategorien geordnet angezeigt werden.	• APP: Es soll beim Befüllen der Artikelanzeige nach Kategorien sortiert werden. Diese sollen logisch angeordnet werden (Vorspeise vor Hauptgang usw.). • DBS: DatabaseService und Netzwerkschnittstelle muss angepasst werden.

			• DBM: Beim Anlegen eines Artikels muss es möglich sein, eine Kategorie auszuwählen.
3	Artikel-kommentare	• Es soll bei einer Bestellung möglich sein, jedem Artikel einen Kommentar hinzuzufügen. Das dient bspw. dem Vermerken von ungewünschten Zutaten. • In der App soll dies durch ein langes Drücken auf den Artikeleintrag passieren. Anschließend geht ein Eingabefenster auf, in dem der Kommentar eingegeben werden kann. Anschließend wird der Artikel mit Kommentar der Bestellung hinzugefügt.	• APP: Implementieren der Funktion, einem bestimmten Artikel einen Kommentar hinzuzufügen. • DBS: Es wird eine neue Tabelle angelegt, in der zu einer Bestellung Artikel mit zugehörigem Kommentar zugeordnet wird. Die Datenbank-Schnittstelle wird nur soweit geändert, dass beim Abholen von Bestellungen diese durch eine Kommentartabelle ergänzt werden. • DBM: Es werden die Kommentare zusätzlich zu den restlichen Informationen angezeigt.
4	Rechnungssplit	• Bei einer offenen Bestellung soll es möglich sein, dass alle Gäste separat Zahlen können. • Dafür soll in einer Übersicht durch die Bedienung die zu Zahlenden Artikel ausgewählt werden können und der Teilbetrag angezeigt werden.	• APP: Die neue Übersicht und Funktion muss implementiert werden und auf das neue Datenbank-Modell angepasst werden. • DBS: Es muss eine neue Tabelle erstellt und angesprochen werden, in der die bestellten Artikel separat abgelegt werden. Diesen wird ein „Bezahlt“-Attribut zugordnet. • DBM: --

2.2 Anwendungsfälle

Die folgenden Anwendungsfälle wurden orientiert an einer Vorlage aus der Vorlesung „Software Engineering“ von Daniel Kulesz erstellt. (3 S. 44 - 46)

2.2.1 Anwendungsfalldiagramm Android-Anwendung

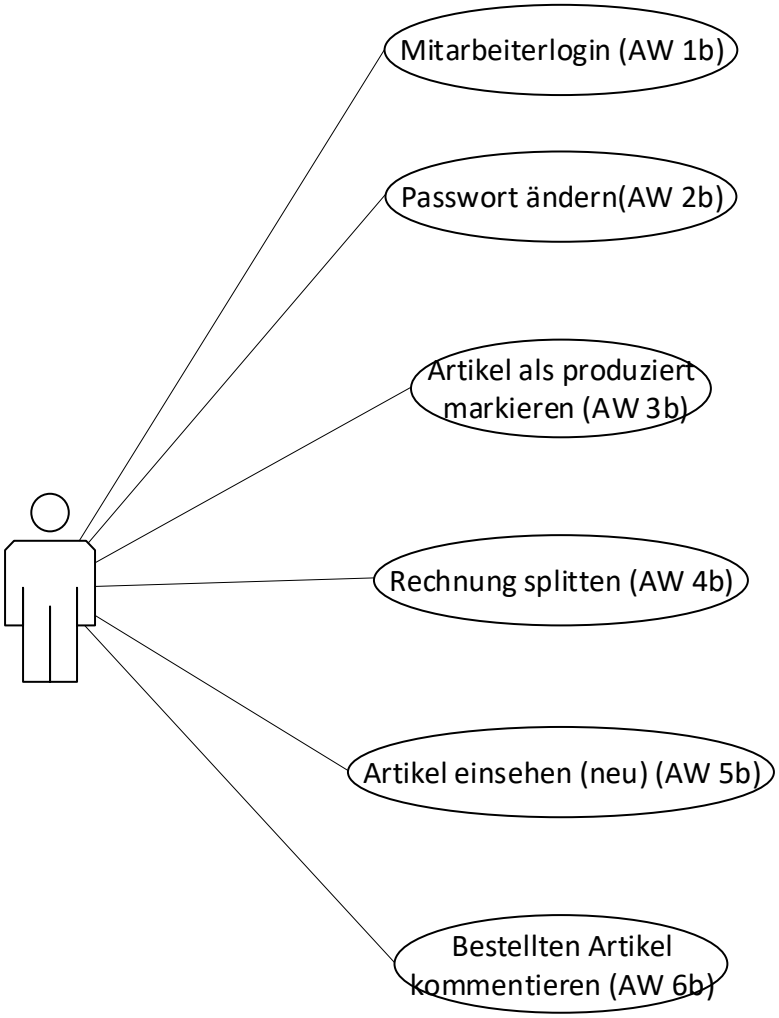


Abbildung 1 Anwendungsfalldiagramm Android Anwendung

2.2.2 Anwendungsfalldefinition der Android- Anwendung

Bezeichnung	AW1b - Mitarbeiterlogin
Priorität	A
Ziel	Der Mitarbeiter loggt sich in der Android-Anwendung mit personifizierten Daten ein. Damit sind alle folgenden Aktionen mit seinem Konto authentifiziert und können ihm zugeordnet werden.
Vorbedingung	Der Mitarbeiter ist nicht eingeloggt und hat Zugriff auf Login-Daten, die in der Datenbank hinterlegt wurden.

Nachbedingung	Die Anmeldung wird mit einer Meldung bestätigt.
Nachbedingung im Sonderfall	1. Login-Name existiert nicht in der Datenbank. 2. Das Passwort ist falsch. 3. Es besteht keine Verbindung zur Datenbank.
Akteure	Bedienung

Bezeichnung	AW2b – Passwort ändern
Priorität	D
Ziel	Der Mitarbeiter ändert in der Android-Anwendung sein Passwort.
Vorbedingung	Der Mitarbeiter ist eingeloggt und hat sein Passwort erneut eingegeben. Das neue Passwort wurde zwei Mal eingegeben.
Nachbedingung	Das Passwort erfüllt Sicherheitskriterien und wurde erfolgreich in der Datenbank geändert.
Nachbedingung im Sonderfall	1. Das Passwort erfüllt die Sicherheitskriterien nicht. 2. Es besteht keine Verbindung zu Datenbank.
Akteure	Bedienung

Bezeichnung	AW3b – Artikel als produziert markieren
Priorität	B
Ziel	Wenn die Bedienung Artikel einer Bestellung von der Küche abholt, markiert sie diese in einer Übersicht in der Android-Anwendung als produziert.
Vorbedingung	Es existiert eine Bestellung mit noch nicht produzierten Artikeln, die in der Küche zubereitet wurden.
Nachbedingung	Die Artikel werden in der Datenbank als produziert markiert.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank aufgebaut werden konnte.
Akteure	Bedienung

Bezeichnung	AW4b – Rechnung splitten
Priorität	C
Ziel	Von einer bereits existierenden Bestellung soll ein Teilbetrag vom Kunden bezahlt werden. Dafür soll von der Bedienung in einer separaten Übersicht die zu zahlenden Artikel ausgewählt werden können. Der zu zahlende Betrag wird angezeigt.
Vorbedingung	Es existiert für den Tisch, an dem der Kunde sitzt, eine Bestellung mit noch nicht bezahlten Artikeln.
Nachbedingung	Der bestellte Artikel wird als bezahlt markiert und erscheint nicht mehr in der Übersicht der zu zahlenden Artikel in der Android-Anwendung.
Nachbedingung im Sonderfall	Es besteht keine Verbindung zur Datenbank und die Artikel wurden in der Datenbank nicht als bezahlt markiert.
Akteure	Bedienung

Bezeichnung	AW5b – Artikel einsehen (neu)
Priorität	F
Ziel	Die Artikel werden in einer Übersicht dargestellt, in der diese nach Kategorien sortiert sind.
Vorbedingung	1. Ein Tisch wurde ausgewählt. 2. In der Datenbank wurden Artikel mit entsprechenden Kategorien hinterlegt.
Nachbedingung	Die Artikel werden richtig und sortiert angezeigt.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank aufgebaut werden konnte.
Akteure	Bedienung

Bezeichnung	AW6b – Bestellten Artikel kommentieren (optional)
Priorität	E
Ziel	Einem Artikel soll ein Kommentar hinzugefügt werden, in dem eine Nachricht für die Küche hinterlegt wird.
Vorbedingung	Es wurde ein Artikel ausgewählt, der einer Bestellung hinzugefügt werden soll.
Nachbedingung	Der Kommentar wurde dem Artikel hinzugefügt und dieser in die Datenbank geschrieben.
Nachbedingung im Sonderfall	--
Akteure	Bedienung

2.2.3 Anwendungsfalldiagramm Datenbankmanager

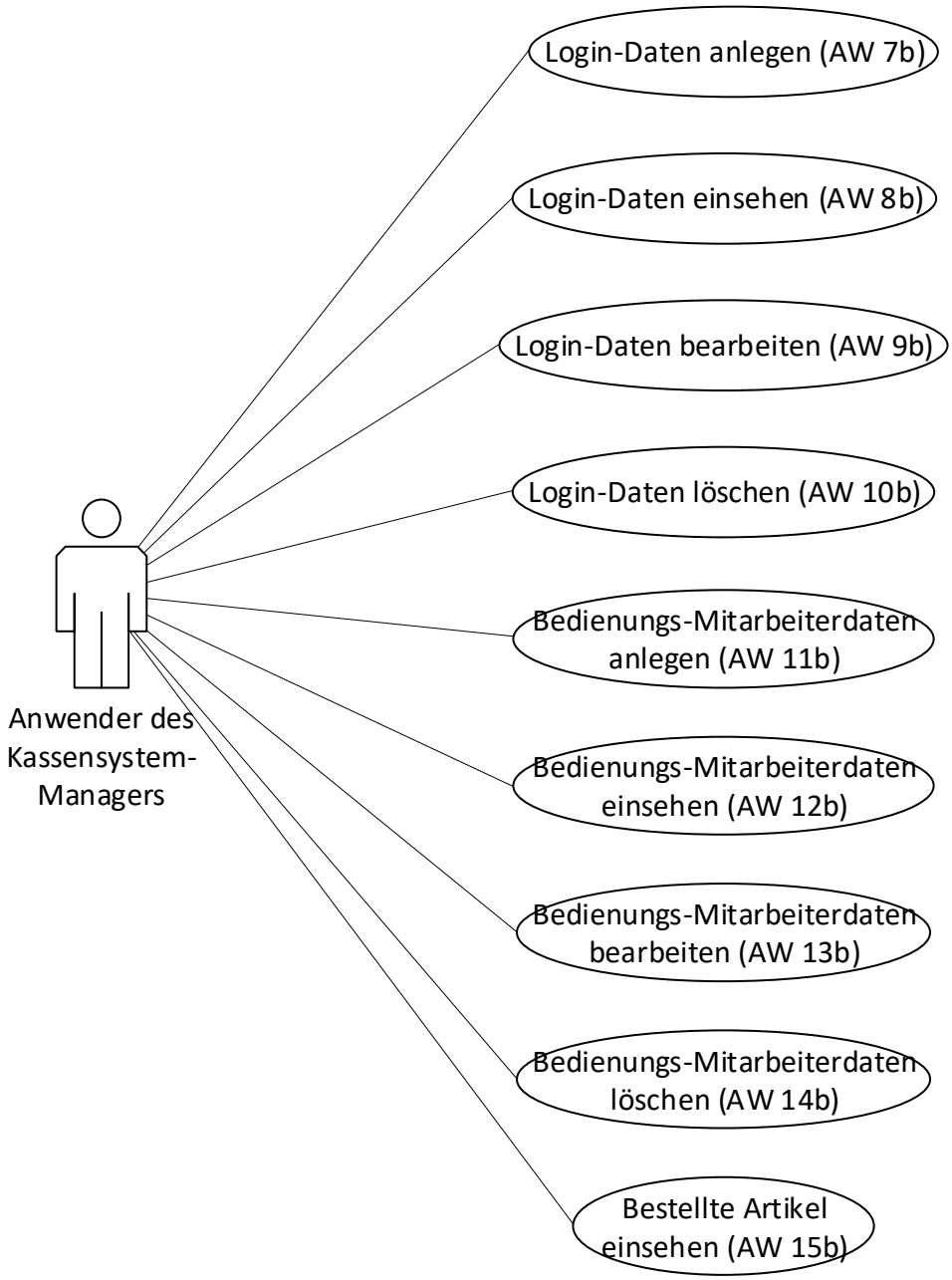


Abbildung 2 Anwendungsfalldiagramm Datenbankmanager

2.2.4 Anwendungsfalldefinition des Datenbankmanagers

Bezeichnung	AW7b - Login-Daten anlegen
Priorität	A
Ziel	Damit sich die Bedienungen im System authentifizieren können, müssen Login-Daten erstellt werden. Die Login-Daten müssen nach dem Anlegen ausgedruckt werden, damit sie der Bedienung übergeben werden können.
Vorbedingung	Eine neue Bedienung wurde eingestellt und benötigt Zugriff auf das System
Nachbedingung	Die Login-Daten wurden erfolgreich in der Datenbank angelegt und ausgedruckt.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank besteht.
Akteure	Gastronom

Bezeichnung	AW8b – Login-Daten einsehen
Priorität	A
Ziel	Die Login-Daten aller Bedienungen können eingesehen werden.
Vorbedingung	Der Gastronom will alle Login-Daten einsehen. Er öffnet den Reiter um die Login-Daten einzusehen
Nachbedingung	Der Gastronom kann alle Login-Daten einsehen.
Nachbedingung im Sonderfall	Es besteht keine Verbindung zur Datenbank. Es wird eine Fehlermeldung angezeigt.
Akteure	Gastronom

Bezeichnung	AW9b – Login-Daten bearbeiten
Priorität	A
Ziel	Die Login-Daten einer Bedienung müssen bearbeitet werden. Der Name ändert sich oder das Passwort soll geändert werden.
Vorbedingung	Es existieren Login-Daten für die entsprechende Bedienung und es besteht der Bedarf diese zu ändern. Die bereits bestehenden Daten wurden erfolgreich in der Anwendung eingegeben.
Nachbedingung	Die Login-Daten wurden erfolgreich geändert und in der Datenbank hinterlegt.
Nachbedingung im Sonderfall	Das geänderte Passwort erfüllt nicht die Sicherheitskriterien. Es besteht keine Verbindung zu Datenbank. Es wird eine Fehlermeldung ausgegeben.
Akteure	Gastronom

Bezeichnung	AW10b – Login-Daten löschen
Priorität	A
Ziel	Die Login-Daten einer Bedienung werden gelöscht, wenn diese nicht mehr benötigt werden.
Vorbedingung	Die Bedienung benötigt die Login-Daten nicht mehr. Beispielsweise aufgrund von einer Kündigung. Der Gastronom löscht die Daten über einen Rechtsklick in der Darstellung.
Nachbedingung	Die Login-Daten wurden erfolgreich gelöscht und die Bedienung kann sich damit nicht mehr im System authentifizieren.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank besteht. Die Login-Daten werden nicht gelöscht
Akteure	Gastronom

Bezeichnung	AW11b – Bedienungs-Mitarbeiterdaten anlegen
Priorität	A
Ziel	Jede Bedienung wird mit ihren persönlichen Daten in der Datenbank hinterlegt. Dazu werden vom Anwender in einem Formular die Daten eingegeben.
Vorbedingung	Es muss ein neuer Mitarbeiter angelegt werden, wenn eine neue Bedienung eingestellt wird.
Nachbedingung	Die Daten der Bedienung wurden in der Datenbank hinterlegt.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank aufgebaut werden konnte.
Akteure	Gastronom

Bezeichnung	AW12b – Bedienungs-Mitarbeiterdaten einsehen
Priorität	A
Ziel	In einer tabellarischen Übersicht sollen die Mitarbeiterdaten aller Bedienungen dargestellt werden.
Vorbedingung	Es besteht eine Verbindung zur Datenbank.
Nachbedingung	Es werden alle vorhandenen Mitarbeiterdaten dargestellt.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank aufgebaut werden konnte.
Akteure	Gastronom

Bezeichnung	AW13b – Bedienungs-Mitarbeiterdaten bearbeiten
Priorität	A
Ziel	Die Daten werden vom Anwender bearbeitet. In einem Formular werden die bisherigen Daten durch neue überschrieben.
Vorbedingung	Es existiert ein Eintrag, dessen Daten bearbeitet werden können.
Nachbedingung	Die Daten wurden erfolgreich bearbeitet und werden aktualisiert in der tabellarischen Ansicht dargestellt.

Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank aufgebaut werden konnte.
Akteure	Gastronom

Bezeichnung	AW14b – Bedienungs-Mitarbeiterdaten löschen
Priorität	A
Ziel	Die Daten eines Mitarbeiters müssen gelöscht werden, weil dieser nicht mehr angestellt ist. Dafür wird vom Anwender mit einem Rechtsklick auf den Eintrag das Kontextmenü „Löschen“ ausgewählt.
Vorbedingung	Einem Mitarbeiter wurde gekündigt.
Nachbedingung	Die Daten wurden gelöscht. In der tabellarischen Übersicht wird der Eintrag nicht mehr angezeigt.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank aufgebaut werden konnte.
Akteure	Gastronom

Bezeichnung	AW15b – Bestellte Artikel einsehen
Priorität	A
Ziel	Die Artikel, die zu einer Bestellung gehören, werden angezeigt. Dafür wird mit Doppelklick oder Rechtsklick auf die Bestellung die Detail-Ansicht der Bestellung in einem neuen Fenster geöffnet.
Vorbedingung	Die tabellarische Übersicht der Bestellungen wurde geöffnet.
Nachbedingung	Die Details mit den bestellten Artikeln wurden in einem neuen Fenster angezeigt.
Nachbedingung im Sonderfall	Es wird eine Fehlermeldung angezeigt, dass keine Verbindung zur Datenbank aufgebaut werden konnte.
Akteure	Gastronom

3 Entwurf

Im Folgenden Abschnitt wird der Entwurf des Datenbanksystems beschrieben. Dabei wird auf einzelne Funktionen des Systems und deren Umsetzung eingegangen. Dies kann außerdem als Informationsquelle für spätere Arbeiten verwendet werden.

3.1 Beschreibung der Softwarekomponenten

Das gesamte System ist aufgeteilt in vier Komponenten:

1. Database-System: Definiert eine Schnittstelle zur MySQL-Datenbank. Beinhaltet außerdem den Rest-Api-Controller der die Schnittstelle zum Netzwerk darstellt.
   ➔ Github-Repository: https://github.com/Kassensystem/DatabaseSystem
2. Android-Application: Android-Anwendung mit Zugriff auf das Datenbank-System über die Netzwerkschnittstelle (Rest-Api-Controller)
   ➔ Github-Repository: https://github.com/Kassensystem/AndroidApplication
3. Manager-Application: Grafisches User Interface (GUI) zur lokalen Verwaltung und Einsicht der Daten. Verwendet das Database-System um auf die MySQL-Datenbank zuzugreifen.
   ➔ Github-Repository: https://github.com/Kassensystem/ManagerApplication
4. MySQL-Datenbank: Wird zu Speicherung der Daten verwendet

Im der folgenden Ausführung werden diese Begriffe verwendet. Es wird teilweise auf den Quellcode in den verlinkten Repositories verwiesen.

3.2 Beschreibung von Programmabläufen und Funktionen

3.2.1 Umsetzung der Netzwerkkommunikation

Die Netzwerkkommunikation zwischen dem Database-System und der Android-Application wurde über eine REST-Api (Representational state transfer – Application Programming Interface) realisiert.

REST ist ein System für die Umsetzung von Webservices, basierend auf HTTP. Sie stellt eine vereinheitlichte Schnittstelle dar, die die Kommunikation im World Wide Web standardisiert.

Die Architektur einer REST-Api orientiert sich an URL's in Kombination mit HTTP-Anfragemethoden wie „GET“, „PUT“ und „POST“.

3.2.2 Umsetzung des Exception-Handlings

Im Database-System können bei der Verarbeitung von Anfragen verschiedene Fehler auftreten: Die Verbindung zur MySQL-Datenbank ist abgebrochen, die Authentifizierung ist aufgrund von fehlenden oder falschen Login-Daten fehlgeschlagen, die übertragenen Daten sind nicht vollständig oder existieren nicht in der Datenbank.

In diesen Fällen wird vom Database-Service eine Exception ausgelöst, in der ein Fehlertext mit Grund hinterlegt wird, der dem Benutzer angezeigt werden soll. Diese Exception wird entweder durch eine Anfrage der Manager-Application oder der Android-Application ausgelöst.

Wenn die Android-Application die Anfrage gestellt hat, wird die Exception vom Rest-Api-Controller abgefangen und eine Antwort an die Android-Application geschickt, in der der Fehlertext übergeben wird. Dieser Fehlertext wird anschließend in der Android-Application wieder extrahiert und dem Benutzer angezeigt.

Wenn die Manager-Application die Anfrage gestellt hat, die die Exception ausgelöst hat, wird der Fehler auch erst in der Manager-Application behandelt. Dem Thread der Anwendung wurde ein „UncaughtExceptionHandler“ zugewiesen (siehe Klasse „KassensystemManagerController, Methode „initialize()“), der alle aufkommenden Exceptions, die nicht im Programmablauf abgefangen werden, behandelt. Dieser Exception-Handler dokumentiert nun alle Fehlerursachen in eine Textdatei „errorlog.txt“. Außerdem werden alle Exceptions, die eine Nachricht für den Anwender enthalten, separat behandelt: Der Fehlertext wird extrahiert und in einem Nachrichtenfenster an den Benutzer ausgegeben.

Diese Methode spart innerhalb der Manager-Application eine Menge redundanten Code in Form von try-catch-Blöcken. So wird das Exception-Handling zentral definiert und kann leicht an neue Anforderungen angepasst werden.

3.2.3 Authentifizierung mit Login-Daten

Im Folgenden wird beschrieben, wie die Authentifizierung im Kassensystem implementiert wurde:

In der Datenbank-Tabelle zur Speicherung von Login-Daten werden ein Login-Name und ein zugehöriges Passwort abgelegt, das in einen 256-bit-Hash-Code verschlüsselt wird. Der Hash-Code wird mit einem Hash-Algorithmus (deutsch: Streuwertfunktion) generiert. Dieser Algorithmus erzeugt aus einem String einen zufälligen Integerwert, der keinen Rückschluss auf den ursprünglichen String erlaubt, allerdings bei selbem

Eingabewert immer den gleichen Wert enthält. Es ist mit heute bekannten Mitteln nicht möglich, aus dem in der Datenbank gespeicherten Hash-Code auf das Passwort zu schließen. Theoretisch erreichbar ist dies nur mit Brute-Force-Attacken, also dem Eingeben von vielen zufälligen Eingabewerten, bis der resultierende Hash-Code mit dem zu entschlüsselnden Hash-Code übereinstimmt. Aufgrund der benötigten Rechenleistung ist dies aber kein realistisches Szenario. (4 S. 13) Ziel der Verschlüsselung ist es, das zu keiner Zeit außer im RAM des Programms das Passwort in Klartext gespeichert wird. Somit ist es nicht möglich, das Passwort abzurufen oder zu rekonstruieren. Da bei der Authentifizierung immer ein String erwartet wird, der anschließend mit dem Hash-Algorithmus verschlüsselt wird, bringt einem potentiellen Angreifer auch das Abgreifen des schon verschlüsselten Hash-Codes nichts. (5)

In der Netzwerkschnittstelle der Rest-API, über die die Android-Anwendung auf die Datenbank zugreift, wird bei jeder Schnittstellenfunktion eine Authentifizierung verlangt, bevor Daten angenommen oder ausgegeben werden. Die Login-Daten werden im http-Header erwartet. In der Android-Anwendung wird über ein Eingabefeld Login-Name und Passwort eingegeben, das Passwort in den Hash-Code verschlüsselt und anschließend bei jeder Anfrage an die Netzwerkschnittstelle im Header mit übergeben.

3.3 Produktübersicht: Aufbau des Kassensystems und Einordnung der Software-Komponenten

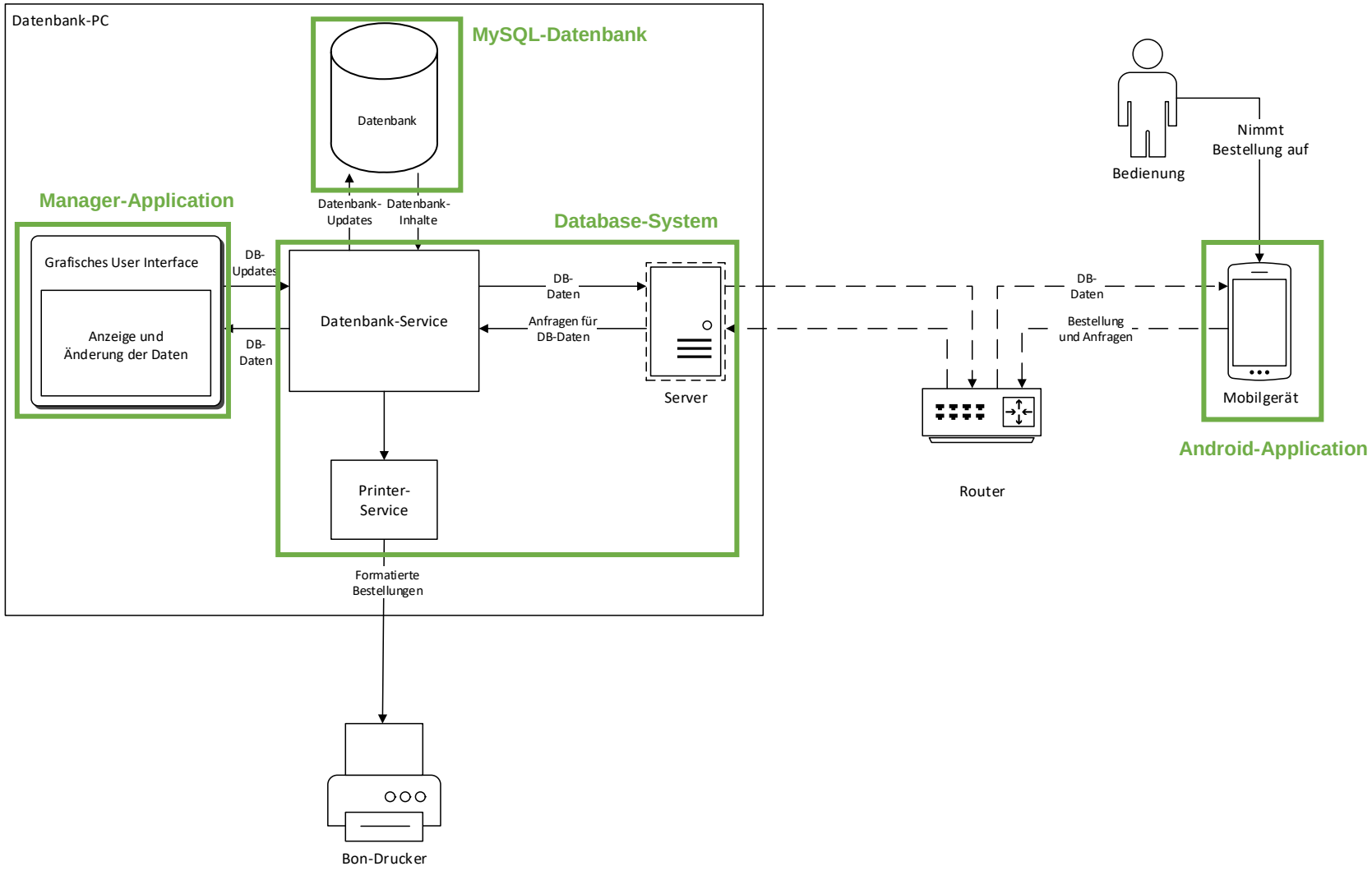


Abbildung 3 Systemübersicht des Projekts

3.4 Produktdaten: Entwurf der Datenbankarchitektur (6)

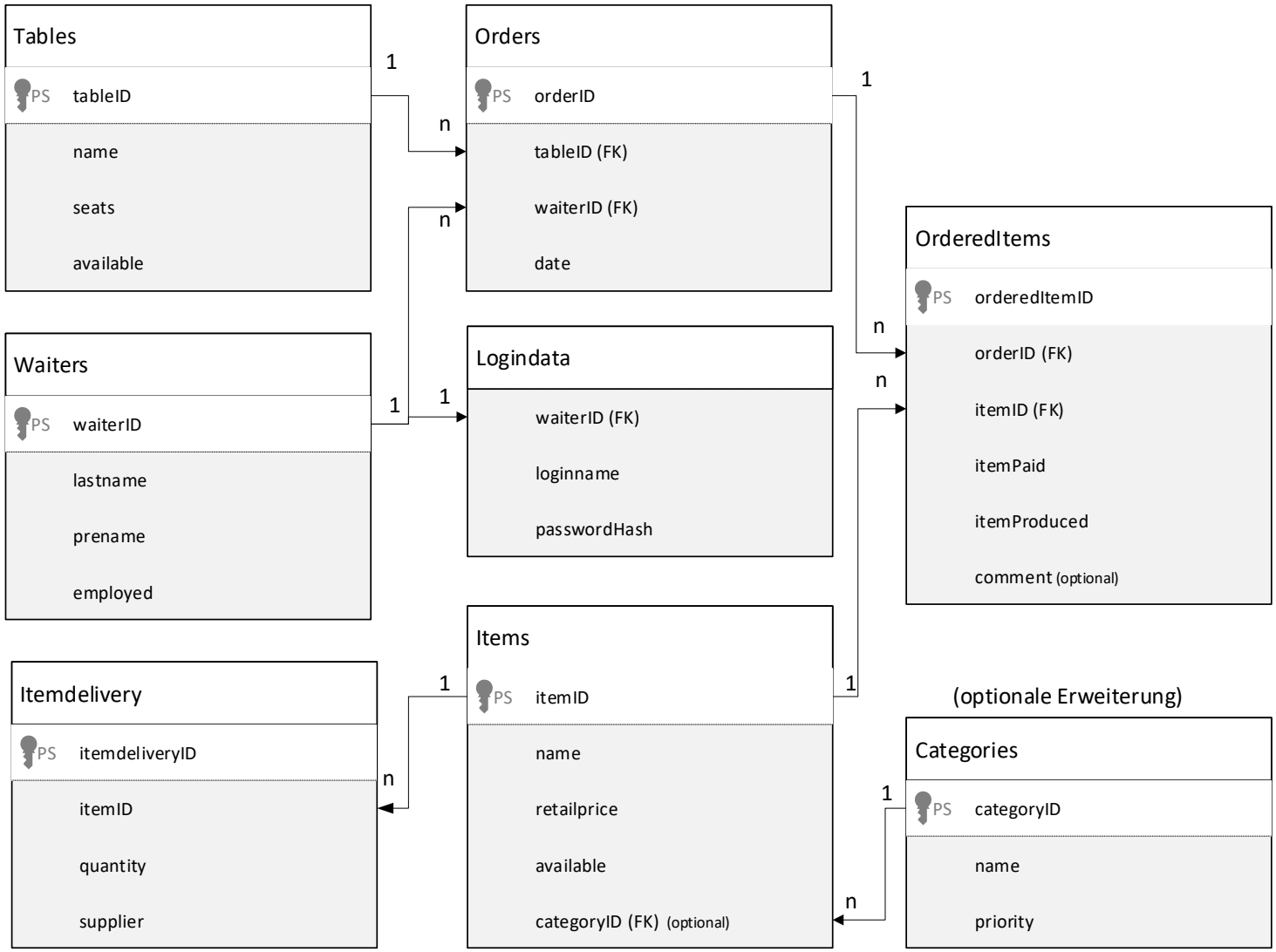


Abbildung 4 Entwurf der überarbeiteten Datenbank

3.5 Workflows: Zusammenhänge zwischen Anwendungsfällen und Datenbankinhalten

Im Folgenden wird in Form von Workflows der Kontext zwischen den Anwendungsfällen und der Datenbankstruktur hergestellt. Dafür werden Geschäfts- und Prozessabläufe in Flussdiagrammen dargestellt und deren Auswirkung auf die Datenbankinhalte definiert.

3.5.1 Tische

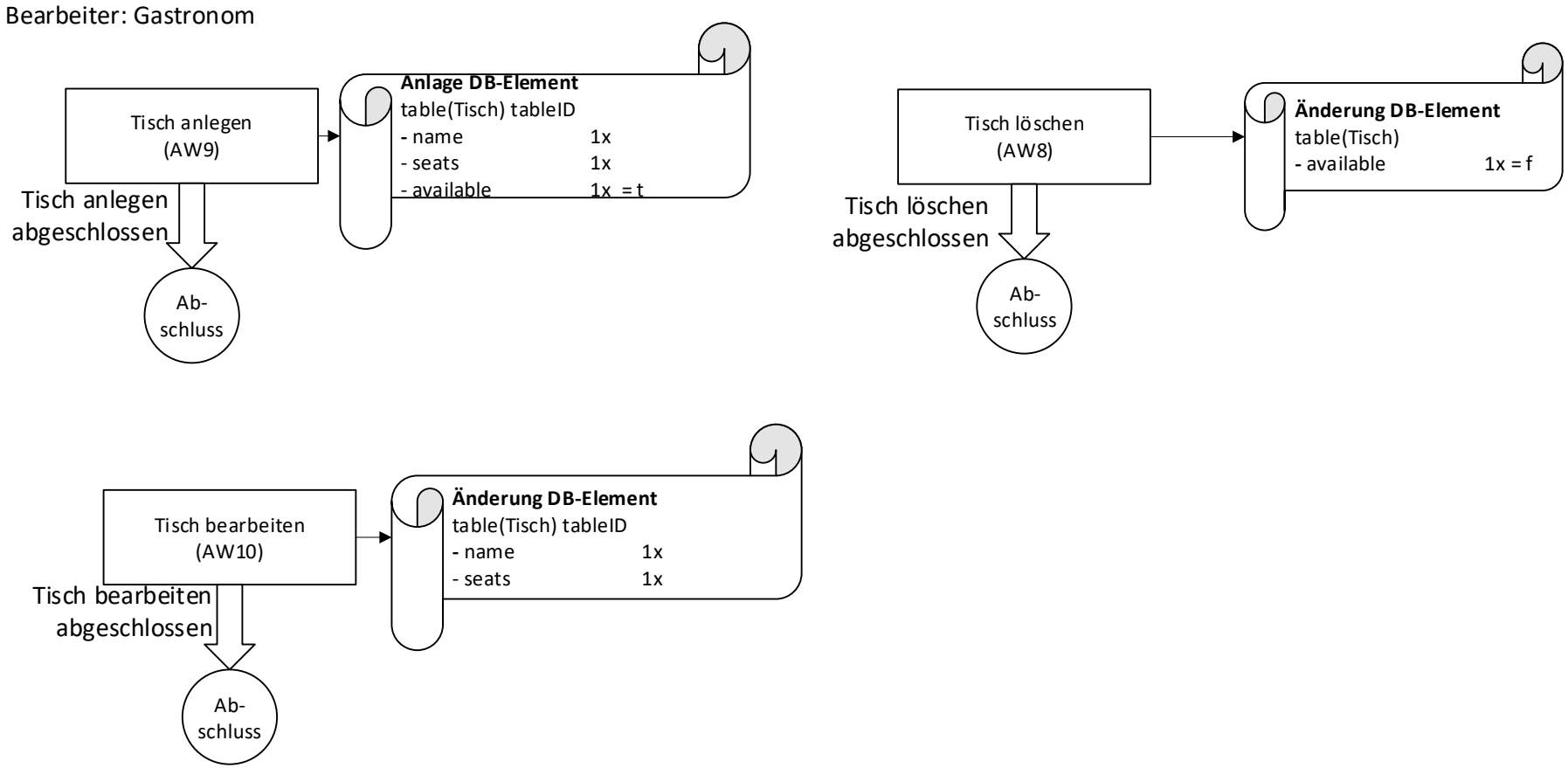
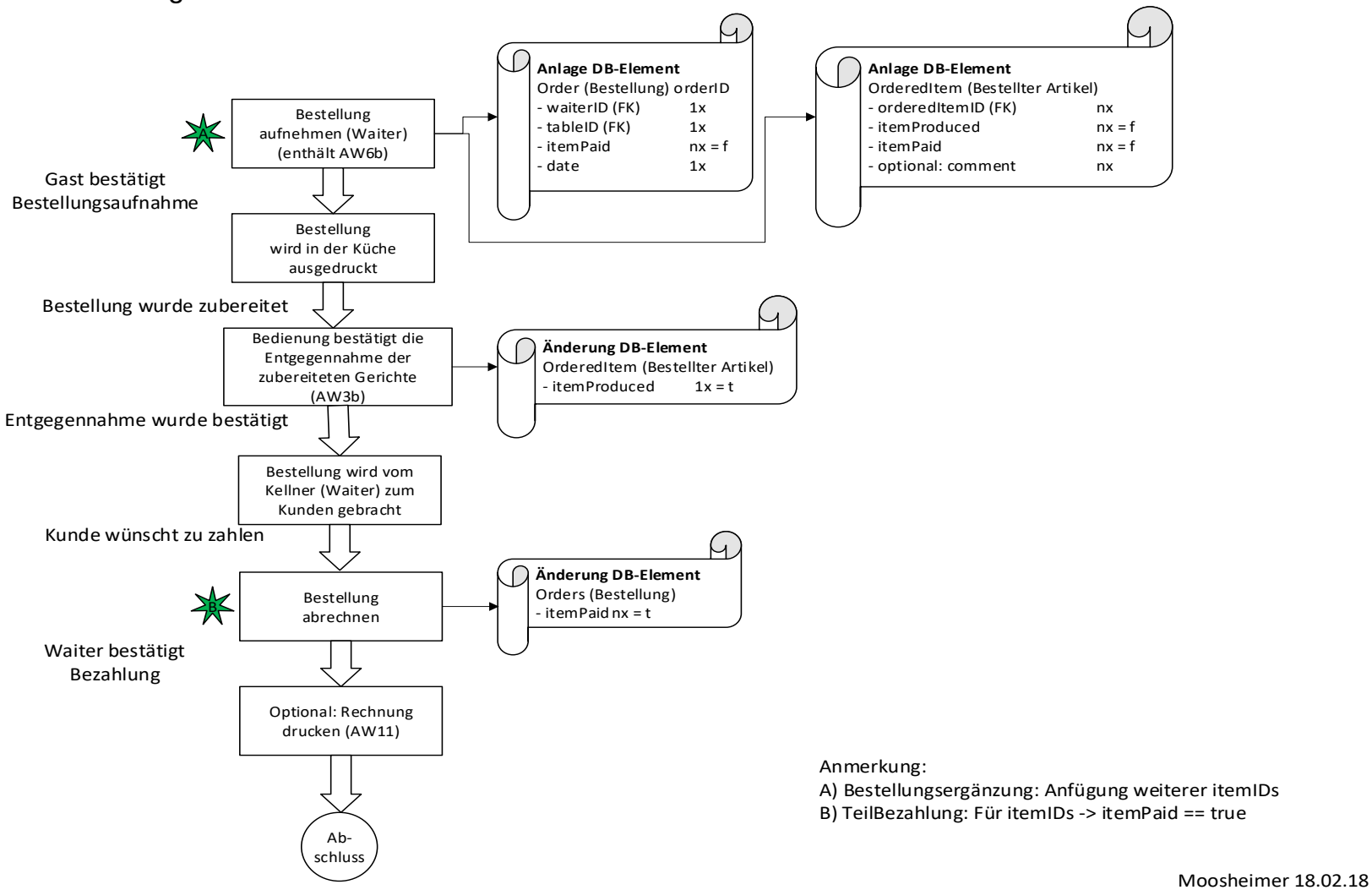


Abbildung 5 Workflow der Tische

3.5.2 Bestellungen



Anmerkung:
A) Bestellungsergänzung: Anfügung weiterer itemIDs
B) TeilBezahlung: Für itemIDs -> itemPaid == true

Moosheimer 18.02.18

Abbildung 6 Workflow der Bestellungen

3.5.3 Artikelverwaltung

Bearbeiter: Gastronom

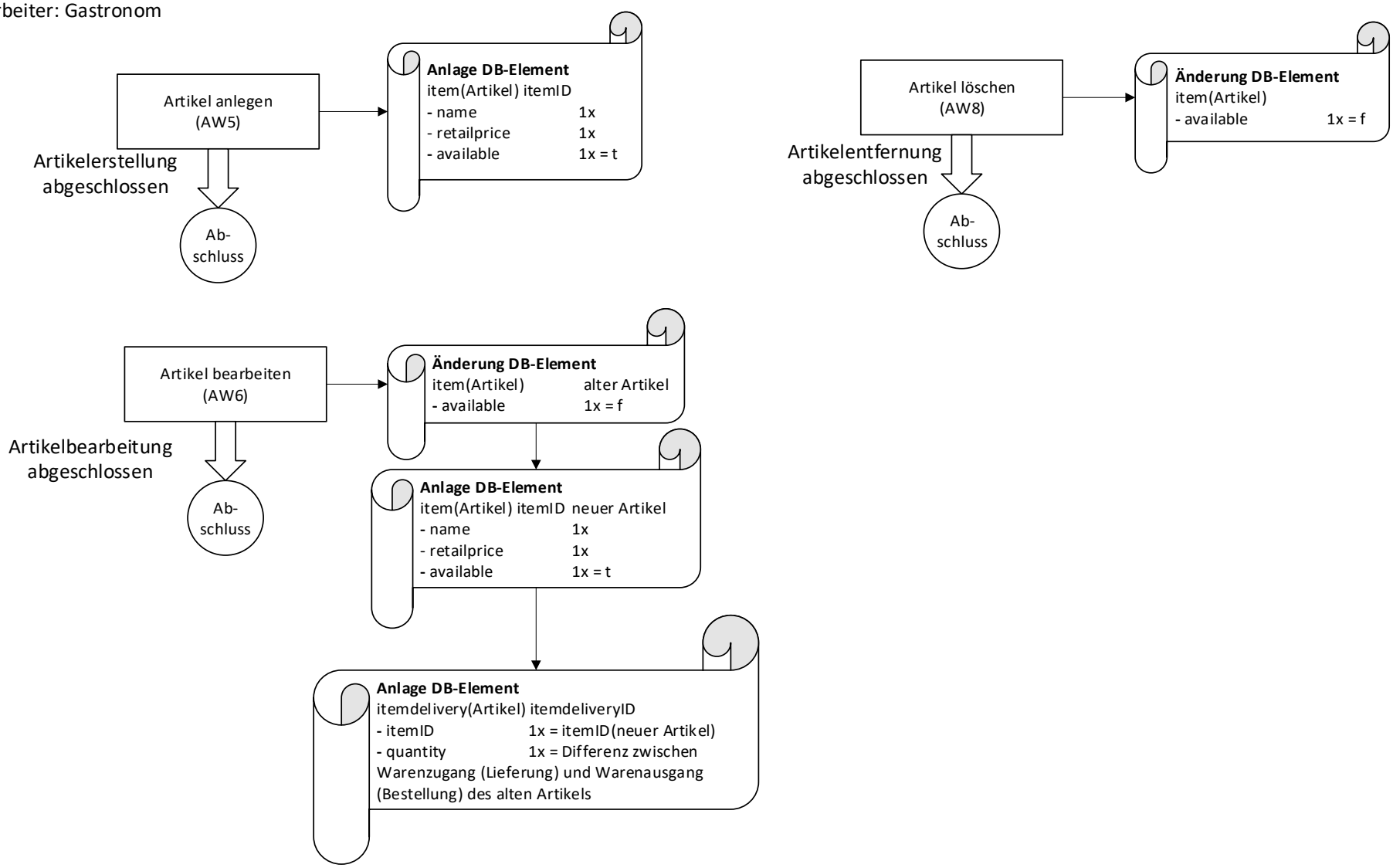


Abbildung 7 Workflow der Artikelverwaltung

3.5.4 Warenein- und -ausgänge

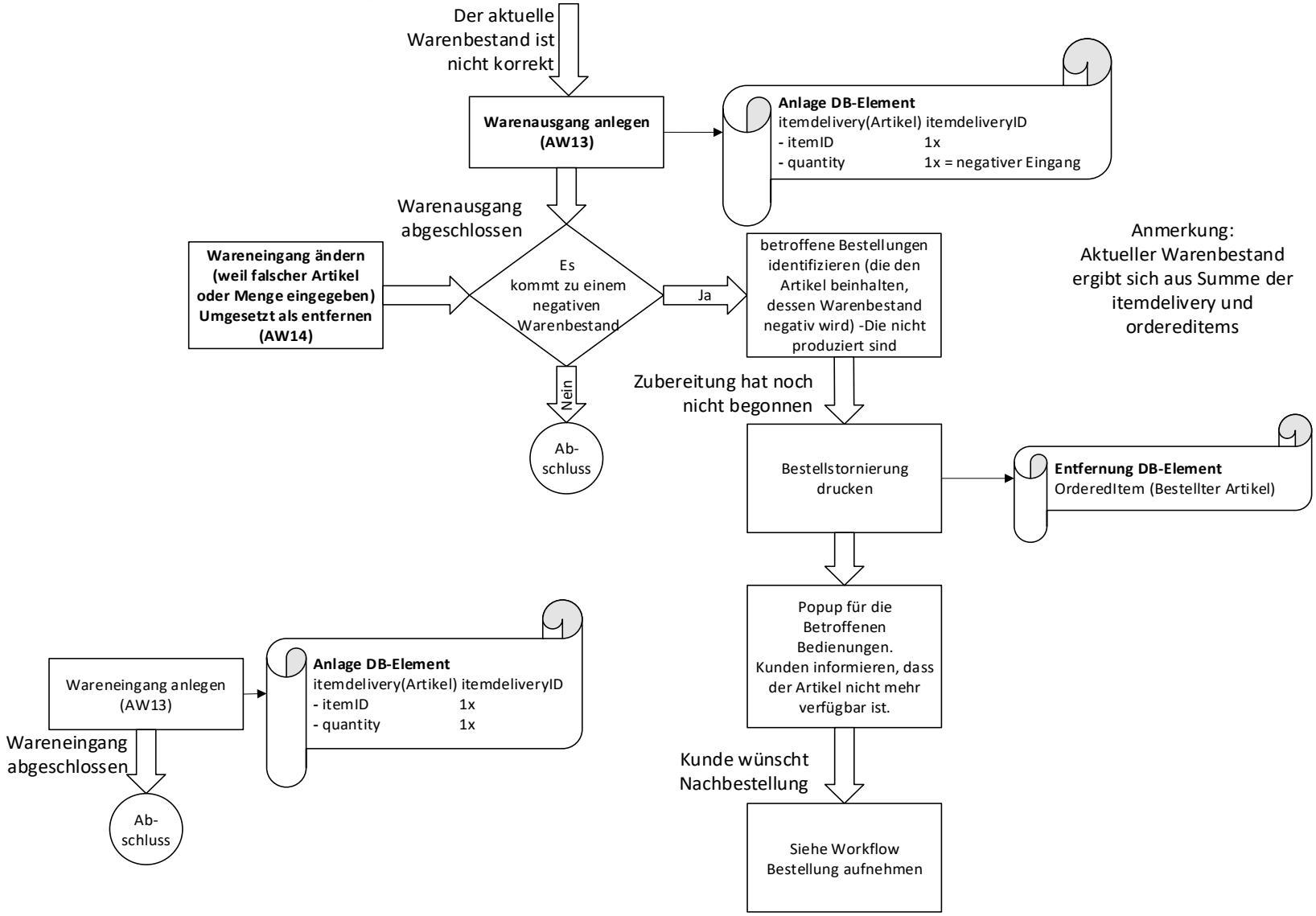


Abbildung 8 Workflow der Warenein- und -ausgänge

3.5.5 Bedienung

Bearbeiter: Gastrom

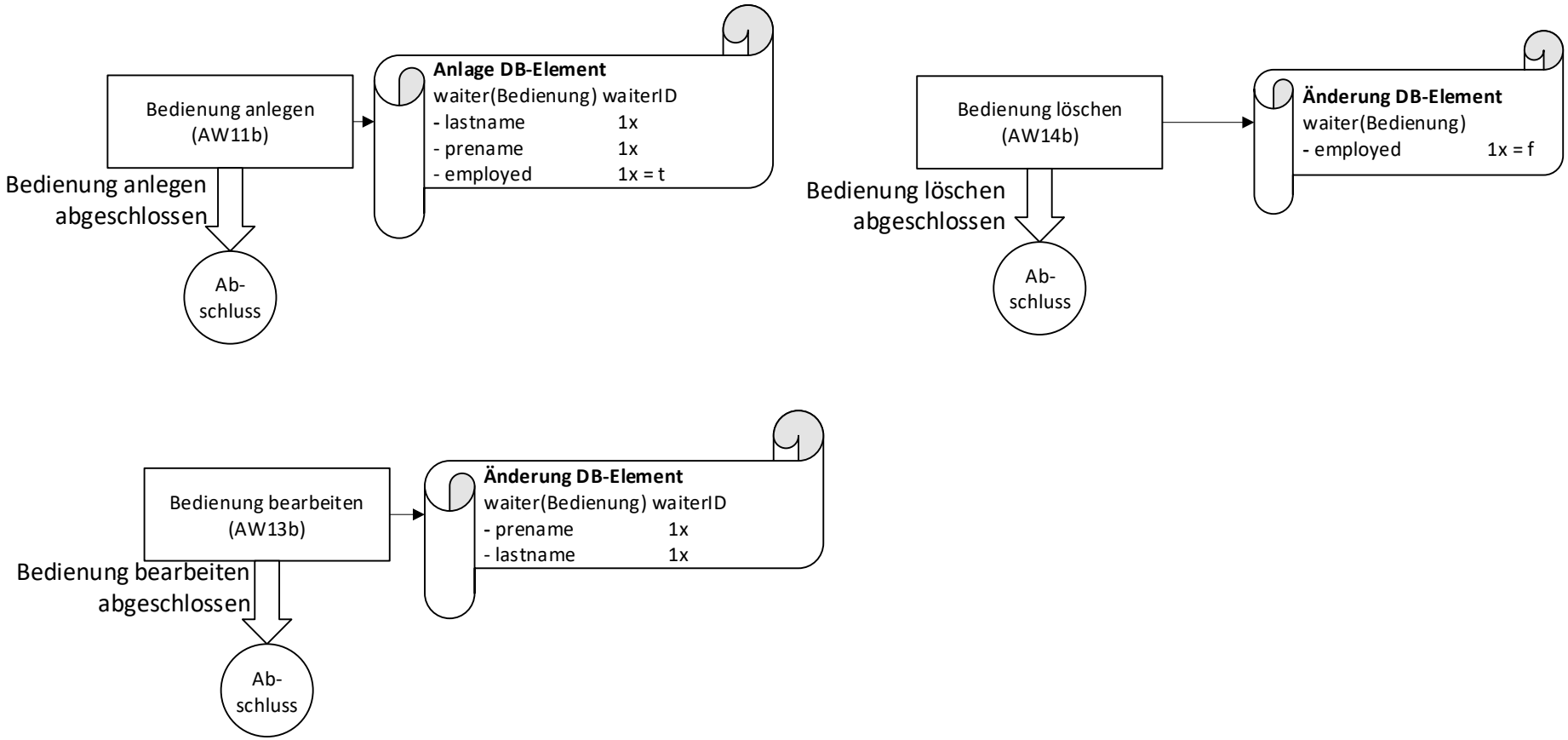


Abbildung 9 Workflow der Bedienungen

3.5.6 Anmeldedaten

Bearbeiter: Gastronom

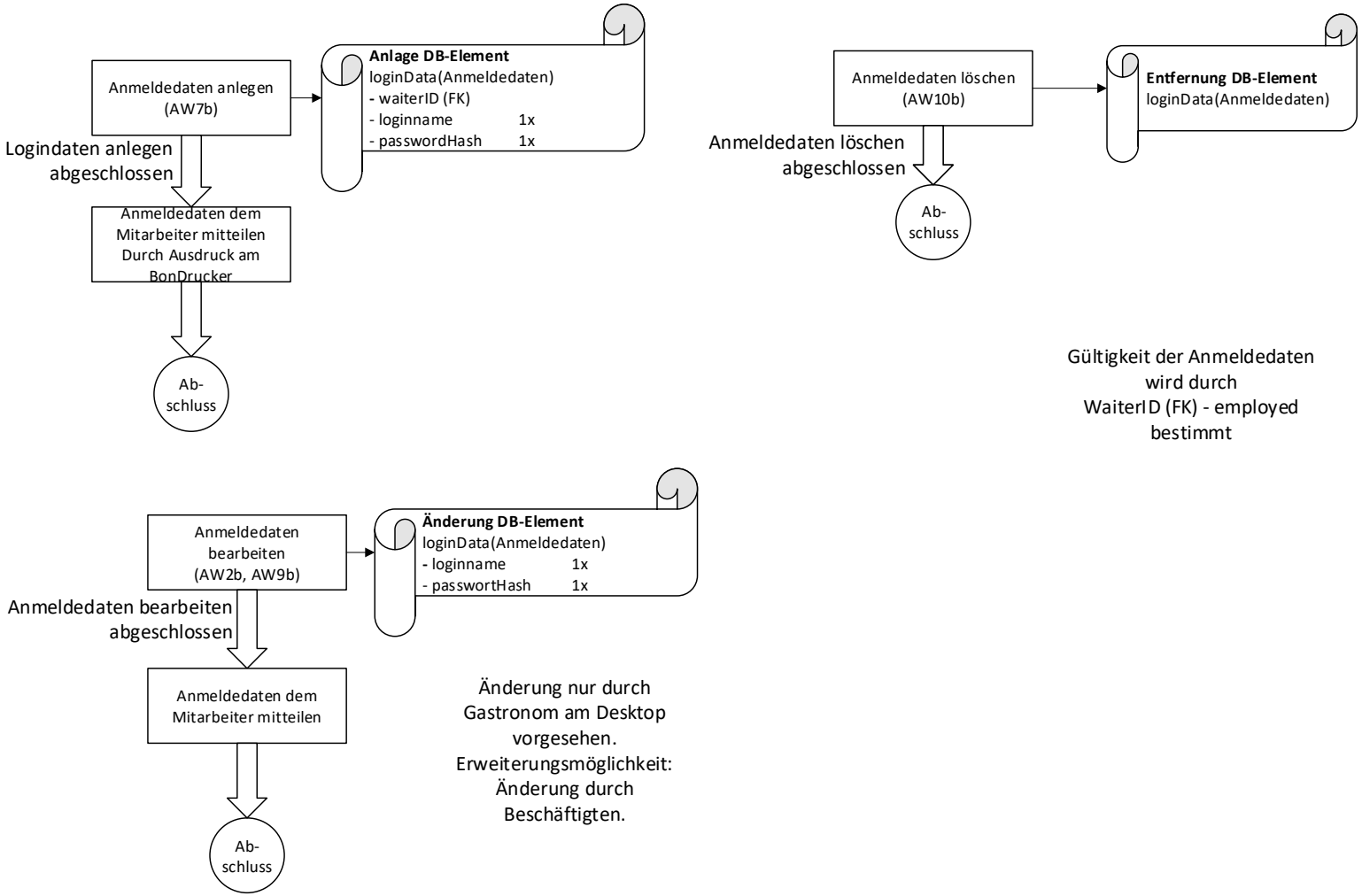


Abbildung 10 Workflow der Anmeldedaten

3.6 Benutzeroberfläche

3.6.1 Benutzeroberfläche der Android-Anwendung

3.6.1.1 Bestellvorgang

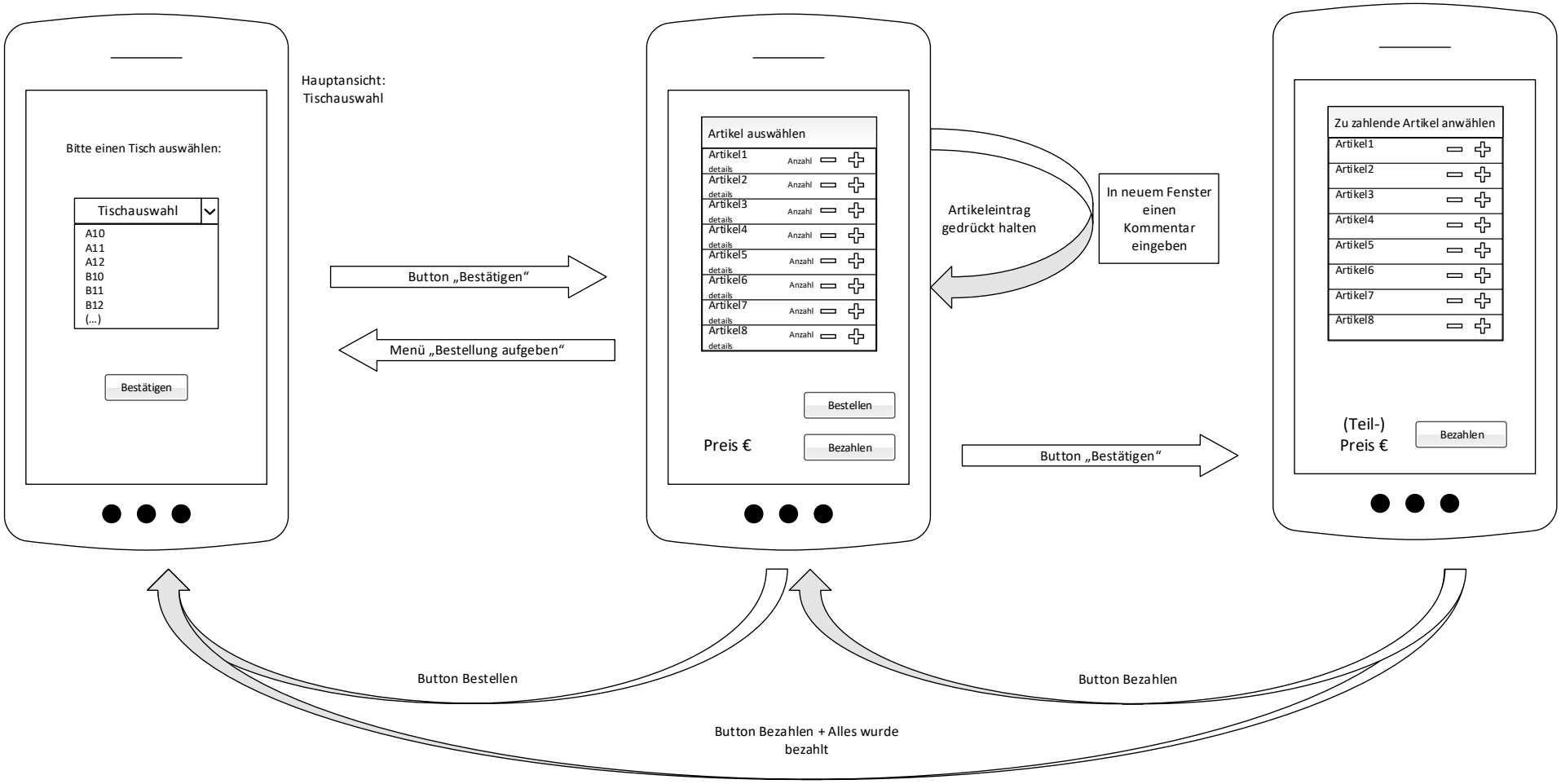


Abbildung 11 GUI Android-Anwendung

3.6.1.2 Bestellte Artikel als produziert markieren (AW3b)

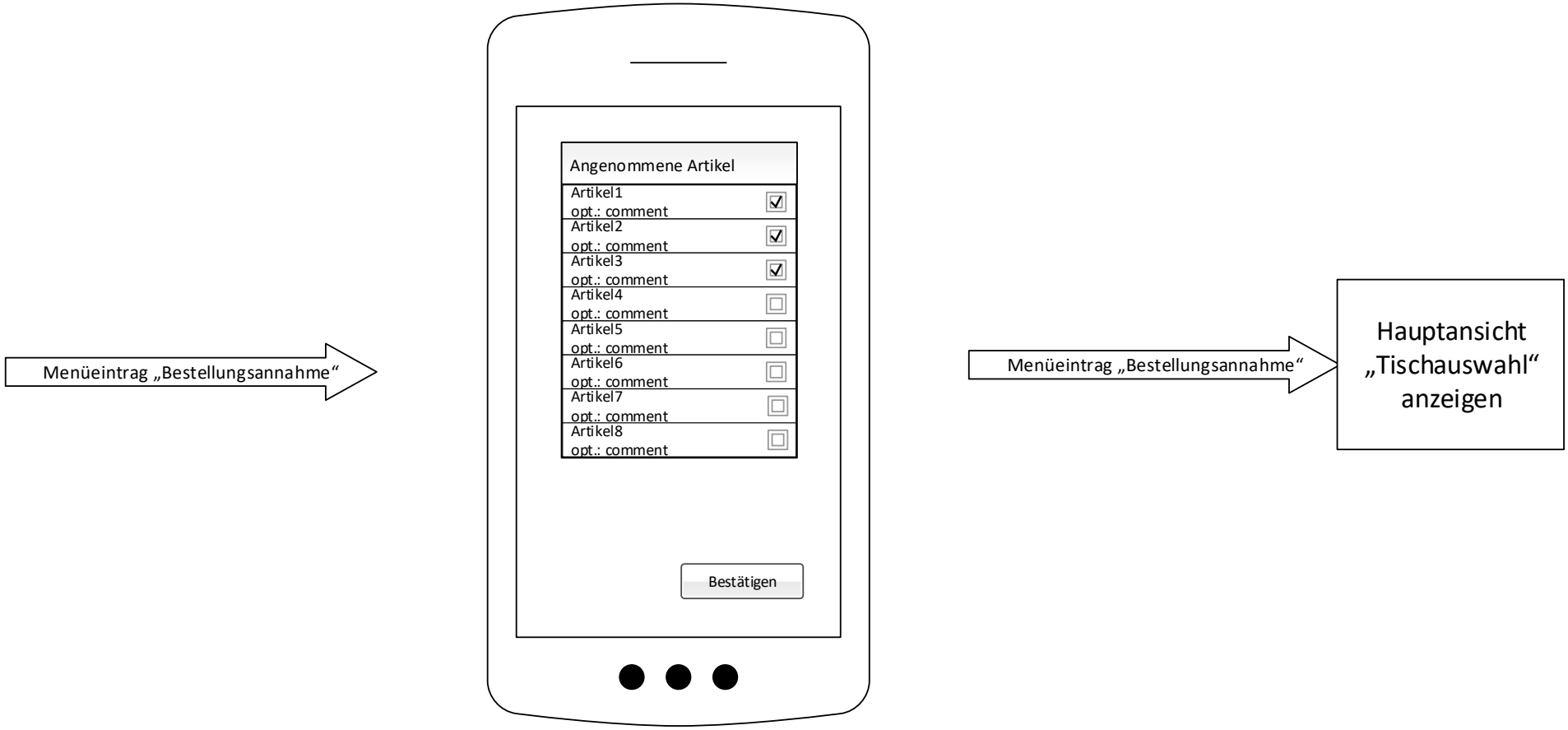


Abbildung 12 GUI Android-Anwendung Artikel als produziert markieren

3.6.2 Benutzeroberfläche des Datenbank-Managers

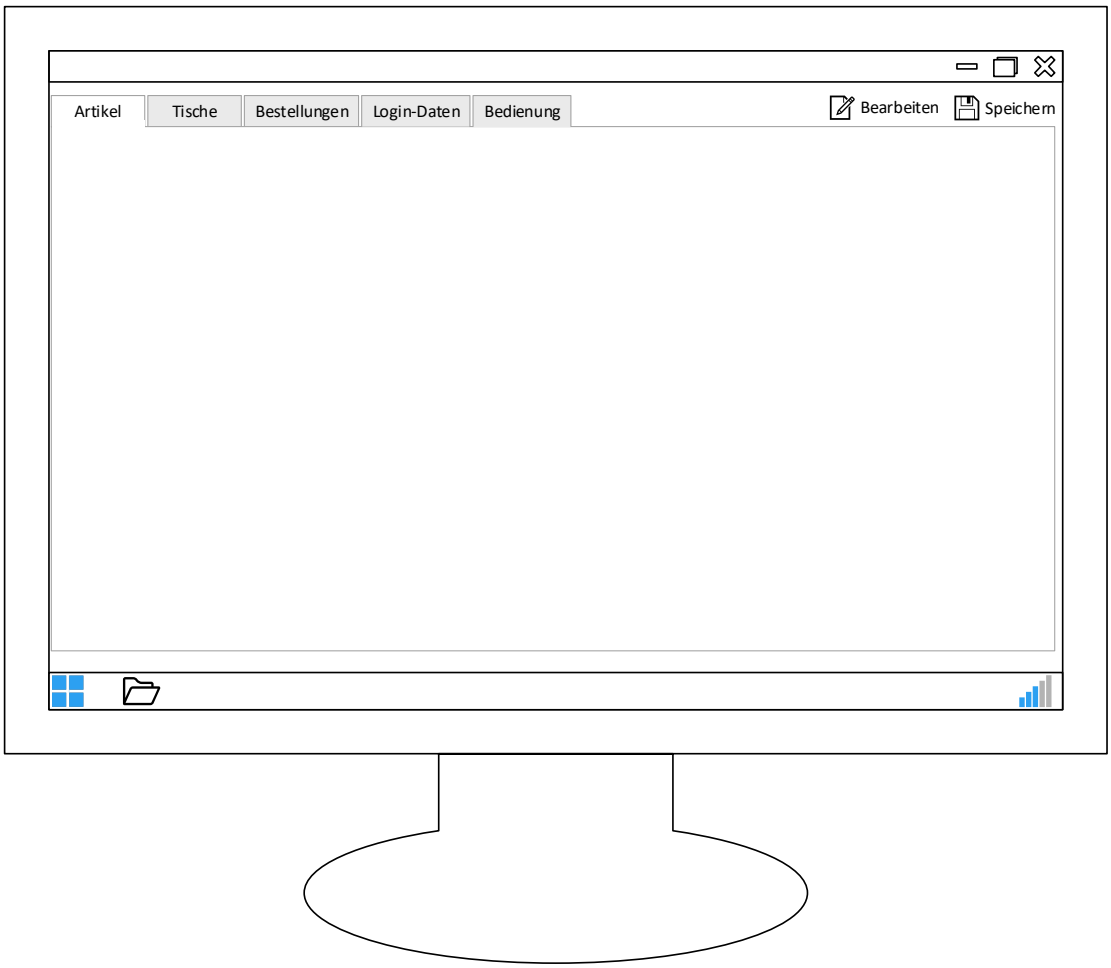


Abbildung 13 Gui Datenbank-Manager

4 Implementierung

4.1 Implementierung des Datenbank-Systems

Im Folgenden sind ausgewählte Klassen des Datenbank-System exemplarisch dokumentiert. Unter dem folgenden Link ist die gesamte Dokumentation abrufbar: https://kassensystem.github.io/DatabaseSystem/

4.1.1 Klasse DatabaseService_Interface

All Known Implementing Classes: DatabaseService

public interface DatabaseService_Interface

Der DatabaseService stellt die Schnittstelle zu einer MySQL-Datenbank dar.

Author:

Marvin Mai

Method Summary

Modifier and Type	Method and Description
void	addItem(Item item) Fuegt der Datenbank einen neuen Artikel hinzu.
void	addItemdelivery(Itemdelivery itemdelivery) Fügt der Datenbank einen neuen Wareneingang hinzu.
void	addLogindata(Logindata logindata) Fügt der Datenbank einen neuen Login-Daten-Eintrag hinzu.
int	addOrder(Order order) Fuegt der Datenbank eine neue Bestellung hinzu.
void	addOrderedItem(OrderedItem orderedItem) Fügt der Datenbank ein neues OrderedItem hinzu.
void	addTable(Table table) Fuegt der Datenbank einen neuen Tisch hinzu.
void	addWaiter(Waiter waiter) Fügt der MySQL-Datenbank eine neue Bedienung hinzu.
static java.sql.Connection	connect() Stellt eine Verbindung zur MySQL-Datenbank her.
void	deleteItem(int itemID) Markiert einen Artikel als nicht verfuegbar.
void	deleteItemdelivery(int itemdeliveryID)

	Löscht einen Wareneingang aus der Datenbank.
void	deleteLogindata(int waiterID) Löscht einen Login-Daten-Satz aus der Datenbank.
void	deleteOrder(int orderID) Loescht eine Bestellung aus der Datenbank.
void	deleteOrderedItem(int orderedItemID) Löscht ein OrderedItem aus der Datenbank.
void	deleteTable(int tableID) Markiert einen Tisch als nicht verfuegbar.
void	deleteWaiter(int waiterID) Setzt den Eintrag der Bedienung auf unemployed, also employed = false.
void	disconnect() Beendet eine bestehende Verbindung mit einem MySQL-Server.
java.util.ArrayList<Item>	getAllAvailableItems() Fragt alle verfügbaren Artikel der Datenbank ab.
java.util.ArrayList<Table>	getAllAvailableTables() Liefert alle als verfügbar markierten Tische der Datenbank.
java.util.ArrayList<Itemdelivery>	getAllItemdeliveries() Fragt die Wareneingaenge der Datenbank ab.
java.util.ArrayList<Item>	getAllItems() Fragt die Artikel der Datenbank ab.
java.util.ArrayList<Logindata>	getAllLogindata() Liefert alle Login-Daten, die sich in der Datenbank befinden.
java.util.ArrayList<OrderedItem>	getAllOrderedItems() Ermittelt alle bestellten Artikel () der Datenbank.
java.util.ArrayList<Order>	getAllOrders() Fragt die Bestellungen der Datenbank ab.
java.util.ArrayList<Table>	getAllTables() Fragt die Tische der Datenbank ab.
java.util.ArrayList<Waiter>	getAllWaiters() Ermittelt alle Bedienungen, die sich in der MySQl-Datenbank befinden.
Item	getItemById(int itemID) Liefert ein Item in Abhängigkeit von einer ID.
Itemdelivery	getItemdeliveryById(int itemdeliveryID)

	Ermittelt anhand einer gegebenen Itemdelivery-ID den zugehörigen Wareneingang ().
Order	getOrderById(int orderID) Liefert eine Bestellung in Abhängigkeit von einer ID.
OrderedItem	getOrderedItemById(int orderedItemID) Bestellten Artikel anhand einer ID ermitteln.
java.util.ArrayList<OrderedItem>	getOrderedItemsByItemId(int itemID) Ermittelt alle bestellten Artikel, die zu einem vorgegebenen Artikel gehören.
java.util.ArrayList<OrderedItem>	getOrderedItemsByOrderId(int orderID) Ermittelt alle bestellten Artikel, die zu einer Bestellung gehören.
float	getOrderPrice(int orderID) Berechnet für eine gegebene Order-ID den Preis.
Table	getTableById(int tableID) Liefert eine Table in Abhängigkeit von einer ID.
Waiter	getWaiterByID(int waiterID) Ermittelt eine Bedienung, die zu einer vorgegebenen ID gehört.
void	printDataConflict(java.util.ArrayList<OrderedItem> orderedItems) Druckt einen gelösten Datenkonflikt aus.
void	printLogindata(java.lang.String loginname, java.lang.String password, Waiter waiter) Druckt einen Login-Daten-Satz aus.
void	printOrder(int orderID, java.util.ArrayList<OrderedItem> orderedItems) Druckt eine Order für die Küche aus mit den neu hinzugefügten orderedItems.
void	printOrderById(int orderID) Ausdrucken einer Bestellung in Abhängigkeit von einer ID.
void	printReceipt(int orderID) Ausdrucken eines Kundenbeleges in Abhängigkeit von einer ID.
void	updateItem(int itemID, Item item) Aktualisiert die Daten eines Artikels.
void	updateLogindata(Logindata logindata) Aktualisiert einen Login-Daten-Satz in der Datenbank mit den Daten eines neuen Login-Daten-Satzes.
void	updateOrder(int orderID, Order order) Aktualisiert die Daten einer Bestellung.
void	updateOrderedItem(int orderedItemID, OrderedItem orderedItem) Aktualisiert die Daten eines OrderedItem,

void	updateTable(int tableID, Table table) Aktualisiert die Daten eines Tisches.
void	updateWaiter(int waiterID, Waiter waiter) Aktualisiert eine Bedienung in der Datenbank mit den Daten eines neuen Bedienungs-Datensatzes.

Method Detail

- connect

  static java.sql.Connection connect()

  Stellt eine Verbindung zur MySQL-Datenbank her.

  Throws: java.lang.IllegalStateException - wenn die Datenbank nicht erreichbar ist.

- disconnect

  void disconnect()

  Beendet eine bestehende Verbindung mit einem MySQL-Server.

- getAllItems

  java.util.ArrayList<Item> getAllItems()

  Fragt die Artikel der Datenbank ab.

  Returns: Artikel der Datenbank.

- getItemById

  `Item` getItemById(int itemID)

  Liefert ein Item in Abhängigkeit von einer ID.

  Parameters: itemID - ID des Artikels.
  Returns: Item mit angegebener ID.

- getAllAvailableItems

  java.util.ArrayList<`Item`> getAllAvailableItems()

Fragt alle verfügbaren Artikel der Datenbank ab.

- getAllTables

  java.util.ArrayList<`Table`> getAllTables()

  Fragt die Tische der Datenbank ab.

  Returns: Tische der Datenbank.

- getTableById

  `Table` getTableById(int tableID)

  Liefert eine Table in Abhängigkeit von einer ID.

  Parameters: tableID - ID des Tisches.
  Returns: Table mit angegebener ID.

- getAllAvailableTables

  java.util.ArrayList<`Table`> getAllAvailableTables()

  Liefert alle als verfügbar markierten Tische der Datenbank.

  Returns: Eine Liste aller verfügbaren Tische

- getAllOrders

  java.util.ArrayList<`Order`> getAllOrders()

  Fragt die Bestellungen der Datenbank ab.

  Returns: Bestellungen der Datenbank.

- getOrderById

  `Order` getOrderById(int orderID)

  Liefert eine Bestellung in Abhängigkeit von einer ID.

  Parameters: orderID - ID der Bestellung.
  Returns: Order mit angegebener ID.

- getOrderPrice

float getOrderPrice(int orderID)

Berechnet für eine gegebene Order-ID den Preis. Dafür wird für jeden bestellten Artikel () der Preis des zugehörigen Artikels () aufsummiert.

Parameters: orderID -
Returns: Preis des Artikels.

- getAllItemdeliveries

  java.util.ArrayList<`Itemdelivery`> getAllItemdeliveries()

  Fragt die Wareneingaenge der Datenbank ab.

  Returns: Wareneingaenge der Datenbank.

- getItemdeliveryById

  `Itemdelivery` getItemdeliveryById(int itemdeliveryID)

  Ermittelt anhand einer gegebenen Itemdelivery-ID den zugehörigen Wareneingang ().

  Parameters: itemdeliveryID -
  Returns: Wareneingang mit der ID itemdeliveryID.

- getAllOrderedItems

  java.util.ArrayList<`OrderedItem`> getAllOrderedItems()

  Ermittelt alle bestellten Artikel () der Datenbank.

  Returns: Alle bestellten Artikel aus der Datenbank.

- getOrderedItemById

  `OrderedItem` getOrderedItemById(int orderedItemID)

  Bestellten Artikel anhand einer ID ermitteln.

  Parameters: orderedItemID - ID des bestellten Artikels.
  Returns: Bestellten Artikel aus der Datenbank.

- getOrderedItemsByOrderId

  java.util.ArrayList<`OrderedItem`> getOrderedItemsByOrderId(int orderID)

Ermittelt alle bestellten Artikel, die zu einer Bestellung gehören.

Parameters: orderID - ID der Bestellung.
Returns: Eine Liste mit den bestellten Artikeln der Bestellung.

- getOrderedItemsByItemId

  java.util.ArrayList<`OrderedItem`> getOrderedItemsByItemId(int itemID)

  Ermittelt alle bestellten Artikel, die zu einem vorgegebenen Artikel gehören.

  Parameters: itemID - Die ID des Artikels.
  Returns: Eine Liste aller bestellten Artikel, die den Artikel mit der itemID beinhalten.

- getAllWaiters

  java.util.ArrayList<`Waiter`> getAllWaiters()

  Ermittelt alle Bedienungen, die sich in der MySQl-Datenbank befinden.

  Returns: Eine Liste mit allen Bedienungen.

- getWaiterByID

  `Waiter` getWaiterByID(int waiterID)

  Ermittelt eine Bedienung, die zu einer vorgegebenen ID gehört.

  Parameters: waiterID - ID der zu ermittelnden Bedienung.
  Returns: Die Bedienung mit der waiterID.

- getAllLogindata

  java.util.ArrayList<`Logindata`> getAllLogindata()

  Liefert alle Login-Daten, die sich in der Datenbank befinden.

  Returns: Eine Liste mit allen Login-Daten.

- addItem

  void addItem(`Item` item)

  Fuegt der Datenbank einen neuen Artikel hinzu.

  Parameters: item - neuer Artikel.

- addTable

  void addTable(`Table` table)

  Fuegt der Datenbank einen neuen Tisch hinzu.

  Parameters: table - neuer Tisch.

- addOrder

  int addOrder(`Order` order)

  Fuegt der Datenbank eine neue Bestellung hinzu.

  Parameters: order - neue Bestellung.

- addItemdelivery

  void addItemdelivery(`Itemdelivery` itemdelivery)

  Fügt der Datenbank einen neuen Wareneingang hinzu.

  Parameters: itemdelivery - neuer Wareneingang.

- addOrderedItem

  void addOrderedItem(`OrderedItem` orderedItem)

  Fügt der Datenbank ein neues OrderedItem hinzu.

  Parameters: orderedItem - Das hinzuzufügende OrderedItem.

- addWaiter

  void addWaiter(`Waiter` waiter)

  Fügt der MySQL-Datenbank eine neue Bedienung hinzu.

  Parameters: waiter - Die hinzuzufügende Bedienung.

- addLogindata

  void addLogindata(`Logindata` logindata)

  Fügt der Datenbank einen neuen Login-Daten-Eintrag hinzu.

  Parameters: logindata - Der hinzuzufügende Login-Daten-Satz

- void updateItem(int itemID, `Item` item)

  Aktualisiert die Daten eines Artikels.

  Parameters:
  itemID - ID des zu aktualisierenden Artikels
  item - neue Artikeldaten.

- void updateTable(int tableID, `Table` table)

  Aktualisiert die Daten eines Tisches.

  Parameters:
  tableID - ID des zu aktualisierenden Tisches.
  table - neue Tischdaten.

- void updateOrder(int orderID, `Order` order)

  Aktualisiert die Daten einer Bestellung.

  Parameters:
  orderID - ID der zu aktualisierenden Bestellung.
  order - neue Bestellungsdaten.

- void updateOrderedItem(int orderedItemID, `OrderedItem` orderedItem)

  Aktualisiert die Daten eines OrderedItem,

  Parameters:
  orderedItemID - ID des zu aktualisierenden OrderdItems.
  orderedItem - Neue OrderedItem Daten.

- void updateWaiter(int waiterID, `Waiter` waiter)

  Aktualisiert eine Bedienung in der Datenbank mit den Daten eines neuen Bedienungs-Datensatzes.

  Parameters:
  waiterID - ID der zu aktualisierenden Bedienung.
  waiter - Die neuen Daten, mit denen die Daten mit der waiterID aktualisiert werden sollen.

- void updateLogindata(`Logindata` logindata)

Aktualisiert einen Login-Daten-Satz in der Datenbank mit den Daten eines neuen Login-Daten-Satzes.

Parameters:
logindata - Die neuen Login-Daten, mit denen die alten Daten ersetzt werden sollen.

- deleteItem

  void deleteItem(int itemID)

  Markiert einen Artikel als nicht verfuegbar. Daten werden nicht geloescht.

  Parameters:
  itemID - ID des als nicht verfuegbar zu markierenden Artikels,

- deleteTable

  void deleteTable(int tableID)

  Markiert einen Tisch als nicht verfuegbar. Daten werden nicht geloescht.

  Parameters:
  tableID - ID des als nicht verfuegbar zu markierenden Tisches,

- deleteOrder

  void deleteOrder(int orderID)

  Loescht eine Bestellung aus der Datenbank.

  Parameters:
  orderID - ID der zu loeschenden Bestellungen.

- deleteItemdelivery

  void deleteItemdelivery(int itemdeliveryID)

  Löscht einen Wareneingang aus der Datenbank.

  Parameters:
  itemdeliveryID - ID des zu loeschenden Wareneingangs.

- deleteOrderedItem

  void deleteOrderedItem(int orderedItemID)

Löscht ein OrderedItem aus der Datenbank.

Parameters:
orderedItemID - Die ID des zu löschen OrderedItems.

- deleteWaiter

  void deleteWaiter(int waiterID)

  Setzt den Eintrag der Bedienung auf unemployed, also employed = false.

  Parameters:
  waiterID - Die ID der zu kündigenden Bedienung.

- deleteLogindata

  void deleteLogindata(int waiterID)

  Löscht einen Login-Daten-Satz aus der Datenbank.

  Parameters:
  waiterID - Die ID der Bedienung, dessen Login-Daten-Satz aus der Datenbank gelöscht werden soll.

- printOrderById

  void printOrderById(int orderID)

  Ausdrucken einer Bestellung in Abhängigkeit von einer ID. Hierbei handelt es sich um einen Kundenbeleg.

  Parameters:
  orderID - ID der auszudruckenden Order.

- printReceipt

  void printReceipt(int orderID)

  Ausdrucken eines Kundenbeleges in Abhängigkeit von einer ID.

  Parameters:
  orderID - ID der auszudruckenden Order.

- void printOrder(int orderID,
  java.util.ArrayList<`OrderedItem`> orderedItems)

Druckt eine Order für die Küche aus mit den neu hinzugefügten orderedItems.

Parameters:
orderID - ID der auszudruckenden Order.
orderedItems - Die neu hinzugefügten Artikel, die in der Küche zubereitet werden sollen.

- printLogindata

  void printLogindata(java.lang.String loginname,
  java.lang.String password,
  `Waiter` waiter)

  Druckt einen Login-Daten-Satz aus.

  Parameters:
  loginname - Der Login-Name.
  password - Das Passwort in Klartext!
  waiter - Die zum Login-Daten-Satz gehörende Bedienung.

- printDataConflict

  void printDataConflict(java.util.ArrayList<`OrderedItem`> orderedItems)

  Druckt einen gelösten Datenkonflikt aus. Ein Datenkonflikt tritt im GUI auf, wenn der Warenbestand eines Artikels negativ wird. Dann erscheint ein Dialog, in dem der Anwender den Datenkonflikt lösen muss. Wenn dies abgeschlossen wurde, wird ein Ausdruck getätigt, auf dem die zu informierenden Tische mit den betroffenen und nicht mehr verfügbaren Artikeln vermerkt sind. So können alle Kunden informiert werden.

  Parameters:
  orderedItems - Alle vom Datenkonflikt betroffenen bestellten Artikel.

4.1.2 Klasse RestApiController

dhbw.sa.kassensystem_rest.restApi.controller
java.lang.Object
dhbw.sa.kassensystem_rest.restApi.controller.RestApiController

```
@RestController
@ComponentScan(value="dhbw.sa.kassensystem_database.data-
base")
 @RequestMapping(value="/api")
public class RestApiController
extends java.lang.Object
```

Der RestApiController stellt einen Server dar, über den Funktionen des DatabaseServices angesprochen werden können. Diese sind über das Netzwerk verfügbar. Dabei müssen die entsprechenden Pfade beachtet werden. Der Root-Pfad ist ".../api".

Author:

Marvin Mai

Constructor Summary

RestApiController()

Method Summary

Modifier and Type	Method and Description
org.springframework.http.ResponseEntity<java.lang.Integer>	createOrder(Order order, java.lang.String loginname, java.lang.String passwordhash) Erstellt eine neue Bestellung in der MySQL-Datenbank.
org.springframework.http.ResponseEntity<?>	createOrderedItems(java.util.ArrayList<OrderedItem> orderedItems, java.lang.String loginname, java.lang.String passwordhash)

java.util.ArrayList<Item>	getAllItems(java.lang.String loginname, java.lang.String passwordhash) Durch das ansprechen des Pfades ".../items“
java.util.ArrayList<OrderedItem>	getAllOrderedItems(java.lang.String loginname, java.lang.String passwordhash)
java.util.ArrayList<Order>	getAllOrders(java.lang.String loginname, java.lang.String passwordhash) Durch das ansprechen des Pfades ".../orders“
java.util.ArrayList<Table>	getAllTables(java.lang.String loginname, java.lang.String passwordhash) Durch das ansprechen des Pfades ".../tables“
java.util.ArrayList<OrderedItem>	getAllUnproducedOrderedItems(java.lang.String loginname, java.lang.String passwordhash)
java.util.ArrayList<OrderedItem>	getOrderedItemsByOrderId(int orderId, java.lang.String loginname, java.lang.String passwordhash)
java.lang.String	handleIndexNotFoundException(MySQLServerConnectionException e, javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse resp)
boolean	login(java.lang.String loginname, java.lang.String passwordhash)

org.springframework.http.ResponseEntity<?>	printReceipe(int orderID, java.lang.String loginname, java.lang.String passwordhash)
boolean	updateLogindata(java.lang.String newPassword, java.lang.String loginname, java.lang.String passwordhash)
org.springframework.http.ResponseEntity<?>	updateOrder(int orderID, Order order, java.lang.String loginname, java.lang.String passwordhash) Updatet eine bereits existierende Bestellung in der Datenbank.
org.springframework.http.ResponseEntity<?>	updateOrderedItems(java.util.ArrayList<OrderedItem> orderedItems, java.lang.String loginname, java.lang.String passwordhash)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

public RestApiController()

Method Detail

- getAllItems

```
@RequestMapping(value="/items")
public java.util.ArrayList<Item> getAllItems(
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

Durch das ansprechen des Pfades ".../api/items" können die Artikel der Datenbank abgefragt werden.

Returns:

Liste aller Artikel der Datenbank.

- getAllOrders

```
@RequestMapping(value="/orders")
public java.util.ArrayList<Order> getAllOrders(
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

Durch das ansprechen des Pfades ".../api/orders" können die Bestellungen der Datenbank abgefragt werden.

Returns:

Liste aller Bestellungen der Datenbank.

- getAllTables

```
@RequestMapping(value="/tables")
public java.util.ArrayList<Table> getAllTables(
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

Durch das ansprechen des Pfades ".../api/tables" können die Tische der Datenbank abgefragt werden.

Returns:

Liste aller Tische der Datenbank.

- getAllOrderedItems

```
@RequestMapping(value="/orderedItems")
public java.util.ArrayList<OrderedItem> getAllOrderedItems(
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

- getAllUnproducedOrderedItems

  @RequestMapping(value**="/unproducedOrderedItems**")
  public java.util.ArrayList<OrderedItem> getAllUnproducedOrderedItems(
  @RequestHeader(value="loginname")
  java.lang.String loginname,
  @RequestHeader(value="passwordhash")
  java.lang.String passwordhash)

- getOrderedItemsByOrderId

  @RequestMapping(value**="/orderedItems/{orderID}")**
  public java.util.ArrayList<OrderedItem> getOrderedItemsByOrderId(
  @PathVariable(value="orderID")
  int orderId,
  @RequestHeader(value="loginname")
  java.lang.String loginname,
  @RequestHeader(value="passwordhash")
  java.lang.String passwordhash)

- createOrder

  @RequestMapping(value**="/order"**, method=**POST**)
  public org.springframework.http.ResponseEntity<java.lang.Integer> cre-
  ateOrder(
  @RequestBody
  Order order,
  @RequestHeader(value="loginname")
  java.lang.String loginname,
  @RequestHeader(value="passwordhash")
  java.lang.String passwordhash)

  Erstellt eine neue Bestellung in der MySQL-Datenbank.

  Parameters:

  order - neu zu erstellende Bestellung.

  Returns:

  ResponseEntity, das Erstellen entweder bestätigt oder eine Fehlermeldung liefert.

- createOrderedItems

  @RequestMapping(value**="/orderedItem"**, method=**POST**)
  public org.springframework.http.ResponseEntity<?> createOrderedItems(

```
@RequestBody
java.util.ArrayList<OrderedItem> orderedItems,
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

- printReceipe

```
@RequestMapping(value="/printOrder/{orderID}", method=POST)
public org.springframework.http.ResponseEntity<?> printReceipe
(@PathVariable(value="orderID")
int orderID,
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

- updateOrder

```
@RequestMapping(value="/order/{orderID}", method=PUT)
public org.springframework.http.ResponseEntity<?> updateOrder(
@PathVariable(value="orderID")
int orderID,
@RequestBody
Order order,
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

Updatet eine bereits existierende Bestellung in der Datenbank.

Parameters:

orderID - Zu aktualisierende Bestellung.

order - Bestellung, deren Daten anstelle der existierenden Bestellung gespeichert werden sollen.

Returns:

ResponseEntity, das Updaten entweder bestätigt oder eine Fehlermeldung liefert.

- updateOrderedItems

```
@RequestMapping(value="/orderedItem", method=PUT)
public org.springframework.http.ResponseEntity<?> updateOrderedItems(
@RequestBody
java.util.ArrayList<OrderedItem> orderedItems,
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

- updateLogindata

```
@RequestMapping(value="/changeLoginPassword", method=PUT)
public boolean updateLogindata(@RequestBody
java.lang.String newPassword,
@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

- login

```
@RequestMapping(value="/login", method=GET)
public boolean login(@RequestHeader(value="loginname")
java.lang.String loginname,
@RequestHeader(value="passwordhash")
java.lang.String passwordhash)
```

- handleIndexNotFoundException

```
@ExceptionHandler(value=MySQLServerConnectionException.class)
@ResponseStatus(value=NOT_FOUND)
@ResponseBody
public java.lang.String handleIndexNotFoundException(MySQLServerCon-
nectionException e,
javax.servlet.http.HttpServletRequest request,
javax.servlet.http.HttpServletResponse resp)
```

4.2 Implementierung der Android-Anwendung

Im Folgenden sind zwei ausgewählte Klassen der Android-Anwendung exemplarisch dokumentiert. Diese exemplarische Dokumentation ist stark gekürzt. Die gesamte Dokumentation ist unter folgendem Link abrufbar: https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/

4.2.1 Klasse TableSelectFragment

public class AnnotationFragment

extends android.support.v4.app.Fragment

In dieser Klasse wird der Kommentare-Hinzufügen-Bildschirm der Applikation erstellt.

Author:
Daniel Schifano

- Constructor Detail:

public AnnotationFragment(Item item)

Der Konstruktor, der zum aufrufen dieser Klasse benötigt wird. Damit wird der neue Bildschirm initialisiert und kann auf dem Smartphone angezeigt werden. Dabei wird der Artikel übergeben, welcher einen Kommentar erhalten soll.

Parameters:

`item` - Der Artikel, welchem ein oder mehrere Kommentare hinzugefügt werden sollen.

public AnnotationFragment()

Der Konstruktor, der zum aufrufen dieser Klasse benötigt wird. Er benötigt keine Übergabe-Parameter. Damit wird der neue Bildschirm initialisiert und kann auf dem Smartphone angezeigt werden.

- Method Detail

- onCreateView

  Diese Methode wird aufgerufen, wenn das Fragment erstellt wird. Dabei werden alle Nodes initialisiert.

  Parameters:

  `inflater` - Instantisiert ein XML-Layout in ein passendes View Objekt

  `container` - Erlaubt den Zugriff auf container Eigenschaften

  `savedInstanceState` - Gibt an in welchem Abschnitt des Lebenszyklus die App sich befindet. Ob sie z.B. geschlossen wurde oder gestartet wurde.

  Returns:

  View die dargestellt werden soll.

4.2.2 Klasse MainActivity

public class MainActivity

extends android.support.v7.app.AppCompatActivity

implements android.support.design.widget.NavigationView.OnNavigationItemSelectedListener

Diese Klasse dient als Container (Hintergrund) für alle anderen Klassen. Zusätzlich werden in dieser Klasse alle Informationen die von der Datenbank empfangen werden, gespeichert.

- Verwendete Variablen
  - allTables

    ```
    public static java.util.ArrayList<Table> allTables
    ```

    Liste die alle Tische der Datenbank beinhaltet.

  - allItems

    ```
    public static java.util.ArrayList<Item> allItems
    ```

    Liste die alle Artikel der Datenbank beinhaltet.

- allOrders

  ```
  public static java.util.ArrayList<Order> allOrders
  ```

  Liste die alle Bestellungen der Datenbank beinhaltet.

- ip

  ```
  public static java.lang.String ip
  ```

  Speichert die IP-Adresse des Servers.

- url

  ```
  public static java.lang.String url
  ```

  Speichert die URL des Servers.

- context

  ```
  public static android.content.Context context
  ```

  Der Hintergrund für alle weiteren Klassen wird hier gespeichert

- Method Detail
  - onCreate

    ```
    protected void onCreate(android.os.Bundle savedInstanceS-
    tate)
    ```

    Diese Methode wird aufgerufen wenn die App gestartet wird. Dabei wird das Layout(Hintergrund) für alle weiteren Klassen initialisiert.

    Overrides:

    `onCreate` in class `android.support.v7.app.AppCompatActivity`

    Parameters:

    `savedInstanceState` - Gibt an in welchem Abschnitt des Lebenszyklus die App sich befindet. Ob sie z.B. geschlossen wurde oder gestartet wurde.

  - showToast

    ```
    public void showToast(java.lang.String handoverText)
    ```

Methode, die den übergebenen Text auf dem Smartphone darstellt.

Parameters:

`handoverText` - Der Text welcher dargestellt werden soll.

- onOptionsItemSelected

```
public boolean onOptionsItemSelected(android.view.Menu-
Item item)
```

Mithilfe dieser Methode wird der Button initialisiert.

Overrides:

`onOptionsItemSelected` in class `android.app.Activity`

Parameters:

`item` - Der Button des Navigation Drawer.

Returns:

true, wenn die Methode richtig abgearbeitet werden kann.

- onNavigationItemSelected

```
public boolean onNavigationItemSelected(an-
droid.view.MenuItem item)
```

Mit dieser Funktion werden die verschiedenen Klassen (Fragments) die im Navigation-Drawer auswählbar sind aufgerufen.

- loadSavedSettings

```
public boolean loadSavedSettings()
```

In dieser Methode werden die IP-Adresse und die URL geladen. Hierfür wird in der Klasse UrlAdjustorFragment die IP-Adresse und die URL über den Lebenszyklus der Applikation gespeichert

Returns:

true, wenn bereits ein URL gespeichert wurde. False, wenn noch kein URL gespeichert wurde

- onBackPressed

```
public void onBackPressed()
```

Mit dieser Methode wird das Verhalten der Anwendung beschrieben, wenn auf dem Smartphone die Rückgängig-Taste gedrückt wird. Wenn diese Taste gedrückt wird, wird automatisch der Startbildschirm der Anwendung (Bestellung-aufgeben-Bildschirm) dargestellt.

Overrides:

`onBackPressed` in class `android.support.v4.app.FragmentActivity`

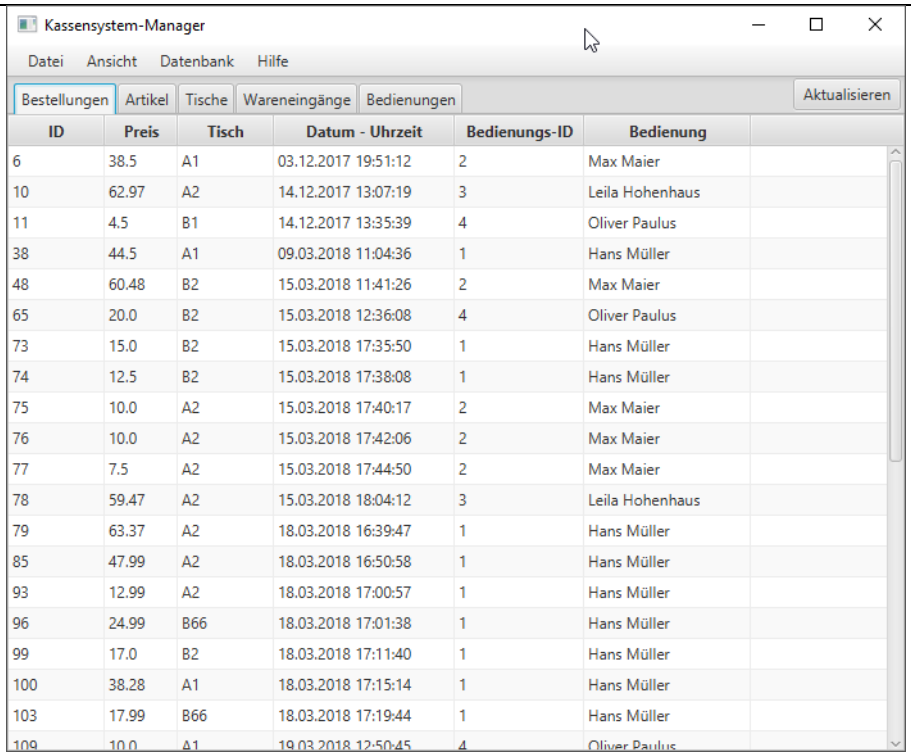
5 Test

5.1 Test des Datenbank-Systems

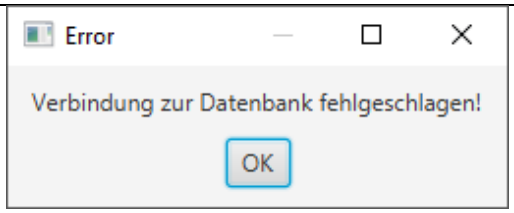
Im Folgenden werden die Tests der Anwendungsfälle der Benutzeranwendung dokumentiert. Diese sind angelehnt an das Skript aus der Vorlesung „Software Engineering“. (7 S. 2 - 17)

5.1.1 Abrufen von Datenbankinhalten

5.1.1.1 Abrufen der Bestellungen

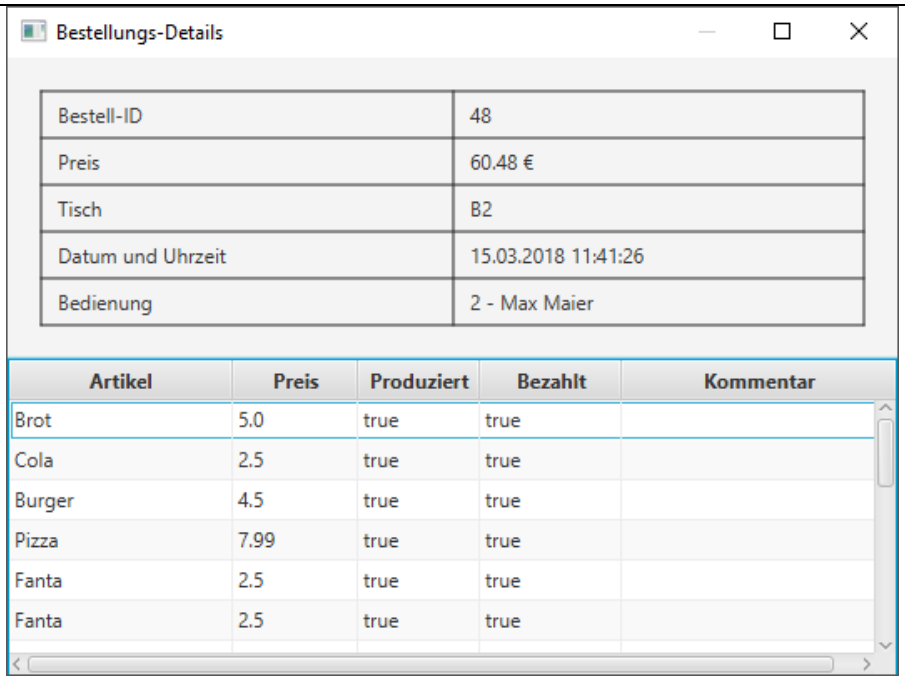
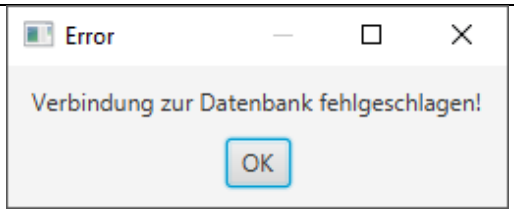
Anwendungsfall	Einsehen aller Bestellungen in dem Kassensystem-Manager (AW 1)
Verwendete Methode	*void refreshOrderData()*
Normalablauf	Nach dem Öffnen der Anwendung werden alle nicht bezahlten Bestellungen in einem Tab dargestellt.
Erwartetes Testergebnis	Eine tabellarische Darstellung aller Bestellungen.
Tatsächliches Testergebnis	
Sonderfall	Es besteht keine Verbindung zum MySQL-Server.

ID	Preis	Tisch	Datum - Uhrzeit	Bedienungs-ID	Bedienung
6	38.5	A1	03.12.2017 19:51:12	2	Max Maier
10	62.97	A2	14.12.2017 13:07:19	3	Leila Hohenhaus
11	4.5	B1	14.12.2017 13:35:39	4	Oliver Paulus
38	44.5	A1	09.03.2018 11:04:36	1	Hans Müller
48	60.48	B2	15.03.2018 11:41:26	2	Max Maier
65	20.0	B2	15.03.2018 12:36:08	4	Oliver Paulus
73	15.0	B2	15.03.2018 17:35:50	1	Hans Müller
74	12.5	B2	15.03.2018 17:38:08	1	Hans Müller
75	10.0	A2	15.03.2018 17:40:17	2	Max Maier
76	10.0	A2	15.03.2018 17:42:06	2	Max Maier
77	7.5	A2	15.03.2018 17:44:50	2	Max Maier
78	59.47	A2	15.03.2018 18:04:12	3	Leila Hohenhaus
79	63.37	A2	18.03.2018 16:39:47	1	Hans Müller
85	47.99	A2	18.03.2018 16:50:58	1	Hans Müller
93	12.99	A2	18.03.2018 17:00:57	1	Hans Müller
96	24.99	B66	18.03.2018 17:01:38	1	Hans Müller
99	17.0	B2	18.03.2018 17:11:40	1	Hans Müller
100	38.28	A1	18.03.2018 17:15:14	1	Hans Müller
103	17.99	B66	18.03.2018 17:19:44	1	Hans Müller
109	10.0	A1	19.03.2018 12:50:45	4	Oliver Paulus

Erwartetes Testergebnis	Es wird eine Fehlermeldung angezeigt.
Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

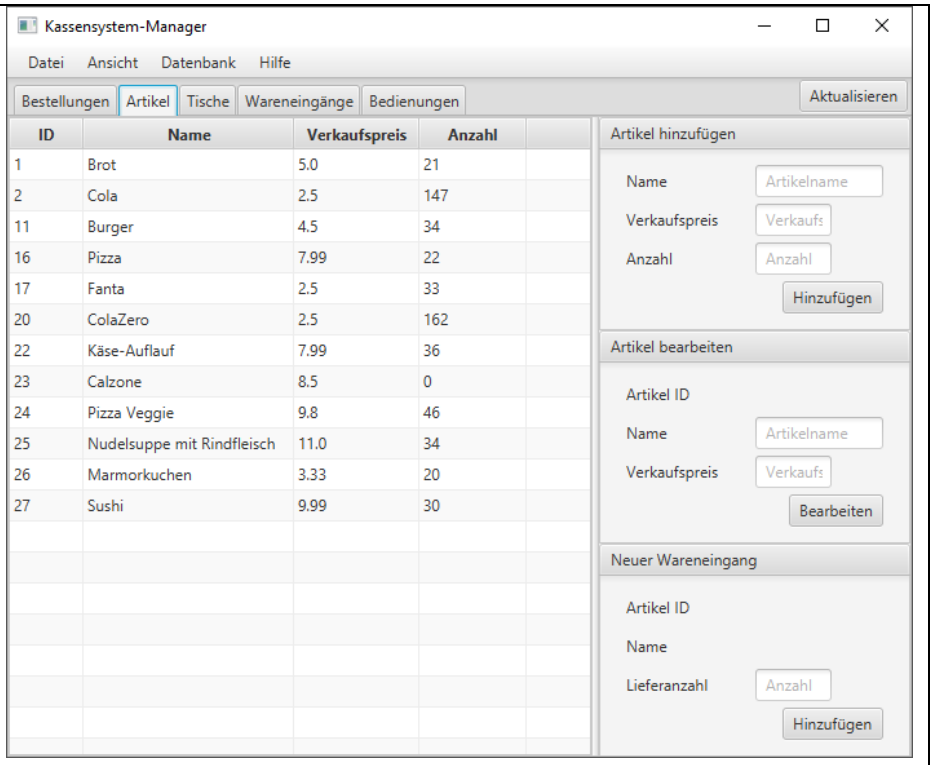
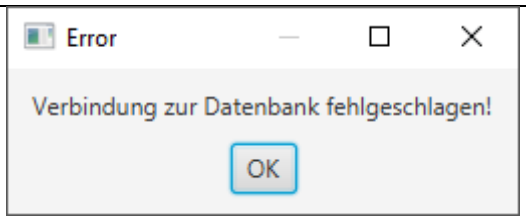
5.1.1.2 Abrufen von Bestelldetails

Anwendungs-fall	AW15b – Bestellte Artikel einsehen
Verwendete Methode	*void showOrderDetails(ActionEvent actionEvent)*
Normalablauf	Mit einem Doppelklick oder einem Rechtsklick und der Auswahl des Menüeintrags „Details“ auf eine Bestellung wird ein Fenster geöffnet, in dem alle Informationen der Bestellung und eine tabellarische Auflistung der bestellten Artikel angezeigt werden.
Erwartetes Testergebnis	Ein Fenster mit den Details der Bestellung.

Tatsächliches Testergebnis	
Sonderfall	Es besteht keine Verbindung zum MySQL-Server.
Erwartetes Testergebnis	Es wird eine Fehlermeldung angezeigt.
Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

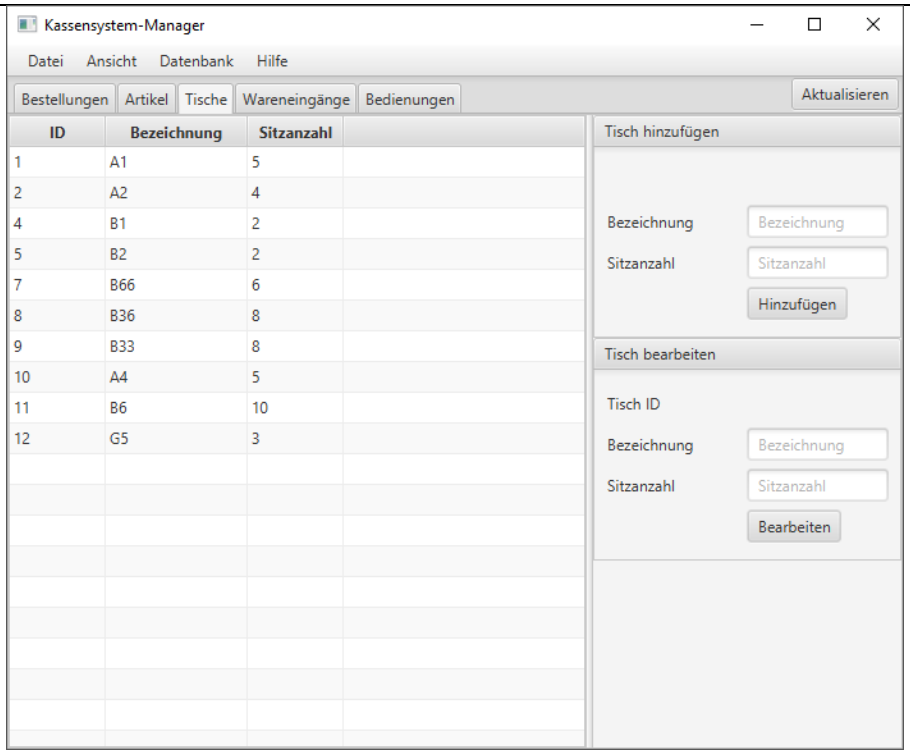
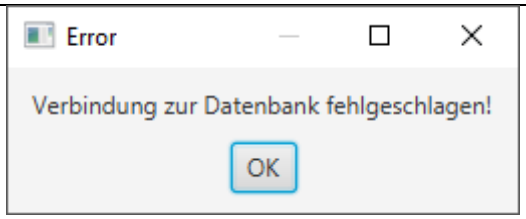
5.1.1.3 Abrufen der Artikel

Anwendungs-fall	Einsehen aller Artikel im Kassensystem-Manager (AW 3)
Verwendete Methode	*void refreshItemData()*
Normalablauf	Nach dem Öffnen der Anwendung werden alle verfügbaren Artikel in einem Tab dargestellt.
Erwartetes Testergebnis	Eine tabellarische Darstellung aller Artikel.

Tatsächliches Testergebnis	
Sonderfall	Es besteht keine Verbindung zum MySQL-Server.
Erwartetes Testergebnis	Es wird eine Fehlermeldung angezeigt.
Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

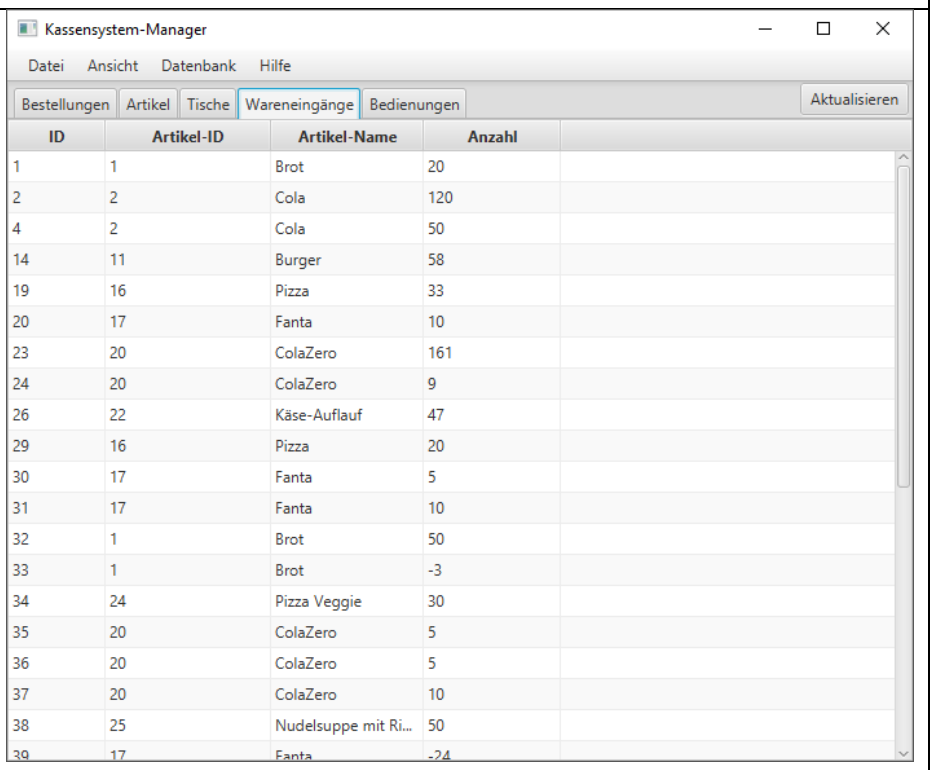
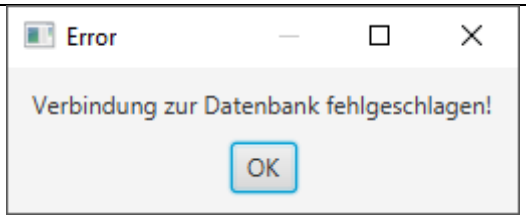
5.1.1.4 Abrufen der Tische

Anwendungs-fall	Einsehen aller Tische im Kassensystem-Manager (AW 7)
Verwendete Methode	*void refreshTableData()*
Normalablauf	Nach dem Öffnen der Anwendung werden alle verfügbaren Tische in einem Tab dargestellt.

Erwartetes Testergebnis	Eine tabellarische Darstellung aller Tische.
Tatsächliches Testergebnis	
Sonderfall	Es besteht keine Verbindung zum MySQL-Server.
Erwartetes Testergebnis	Es wird eine Fehlermeldung angezeigt.
Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.1.5 Abrufen der Wareneingänge

Anwendungs-fall	Einsehen aller Wareneingänge in dem Kassensystem-Manager (AW 12)
Verwendete Methode	*void refreshItemdeliveryData()*

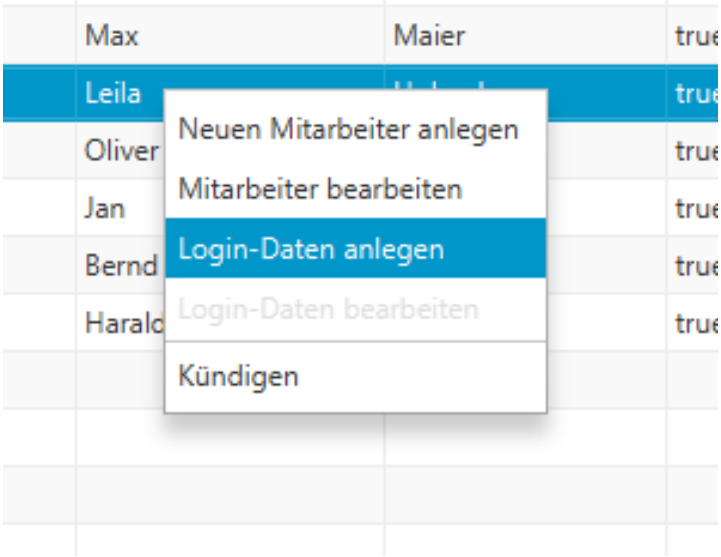
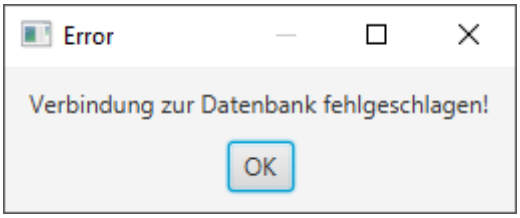
Normalablauf	Nach dem Öffnen der Anwendung werden alle Wareneingänge in einem Tab dargestellt.
Erwartetes Testergebnis	Eine tabellarische Darstellung aller Wareneingänge.
Tatsächliches Testergebnis	
Sonderfall	Es besteht keine Verbindung zum MySQL-Server.
Erwartetes Testergebnis	Es wird eine Fehlermeldung angezeigt.
Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

ID	Artikel-ID	Artikel-Name	Anzahl
1	1	Brot	20
2	2	Cola	120
4	2	Cola	50
14	11	Burger	58
19	16	Pizza	33
20	17	Fanta	10
23	20	ColaZero	161
24	20	ColaZero	9
26	22	Käse-Auflauf	47
29	16	Pizza	20
30	17	Fanta	5
31	17	Fanta	10
32	1	Brot	50
33	1	Brot	-3
34	24	Pizza Veggie	30
35	20	ColaZero	5
36	20	ColaZero	5
37	20	ColaZero	10
38	25	Nudelsuppe mit Ri...	50
39	17	Fanta	-24

5.1.1.6 Login-Daten abrufen

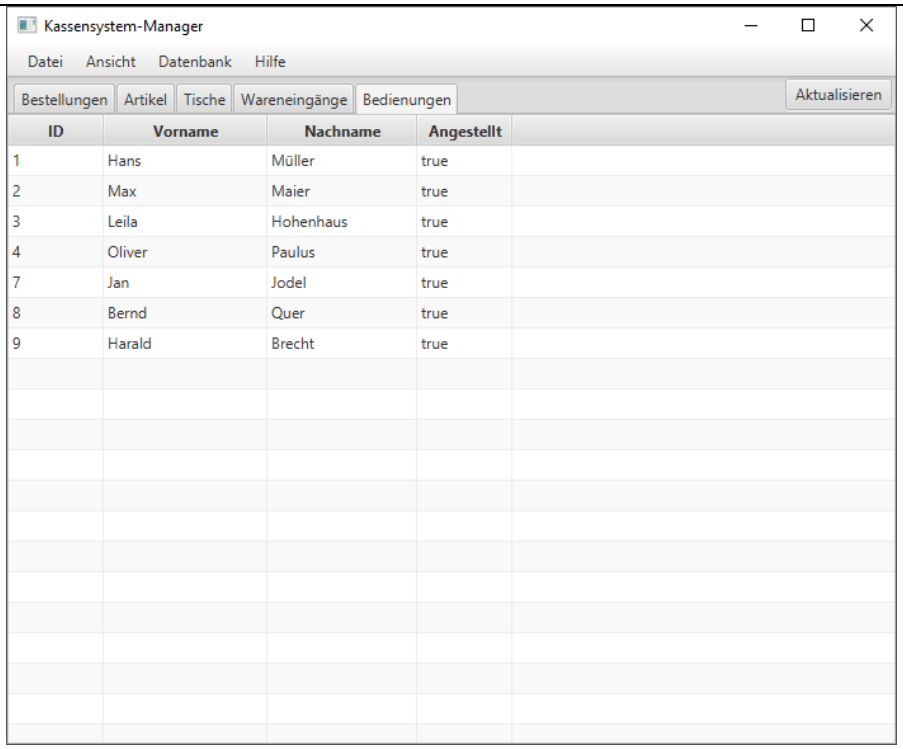
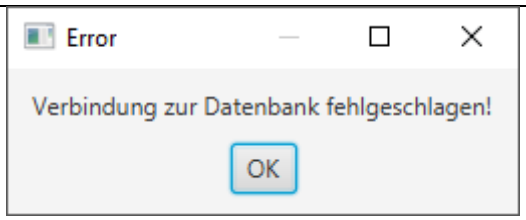
Anwendungs-fall	AW8b – Login-Daten einsehen

Verwendete Methode	*void showLogindataDialog(WaiterModel waiterModel, boolean update)*
Normalablauf	Über einen Rechtsklick auf eine Bedienung in der Bedienungs-Tabelle kann der Login-Daten-Satz eingesehen werden, wenn bereits einer angelegt wurde. Wenn noch keiner angelegt wurde, ist es nicht möglich, Daten einzusehen. Stattdessen kann nur einer neuer Login-Daten-Satz erstellt werden. Außerdem kann nur der Login-Name und nicht das Passwort eingesehen werden, da dieses nicht in Klartext vorliegt.
Erwartetes Testergebnis	Die Anzeige eines Fensters, in dem der Login-Name einer Bedienung eingesehen werden kann.
Tatsächliches Testergebnis	
Sonderfall	1. Es existiert kein Login-Daten-Satz. 2. Es besteht keine Verbindung zur Datenbank.
Erwartetes Testergebnis	1. Es ist nicht möglich, die Login-Daten einzusehen. Stattdessen kann nur ein neuer Login-Daten-Satz erstellt werden. 2. Es wird eine Fehlermeldung angezeigt.
Tatsächliches Testergebnis	1.

	 *2.* 
Test bestanden	Normalablauf: Ja Sonderfall: Ja

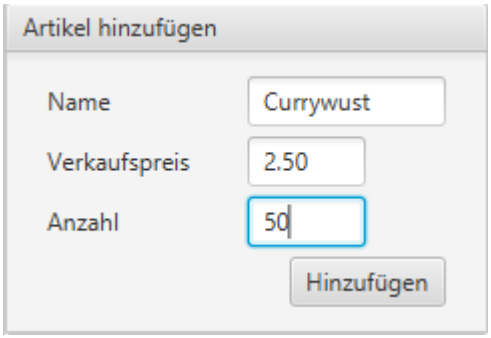
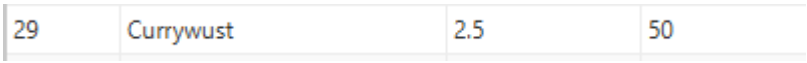
5.1.1.7 Mitarbeiter-Daten abrufen

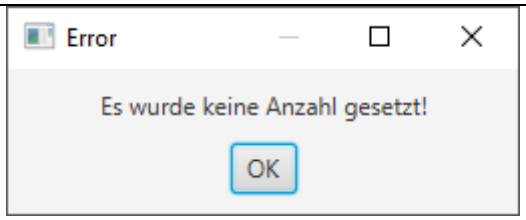
Anwendungs-fall	AW12b – Bedienungs-Mitarbeiterdaten einsehen
Verwendete Methode	*void refreshWaiterData()*
Normalablauf	Es wird eine tabellarische Übersicht aller Mitarbeiter angezeigt.
Erwartetes Testergebnis	Eine tabellarische Übersicht aller Mitarbeiter.

Tatsächliches Testergebnis	
Sonderfall	Es besteht keine Verbindung zum MySQL-Server.
Erwartetes Testergebnis	Es wird eine Fehlermeldung angezeigt.
Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

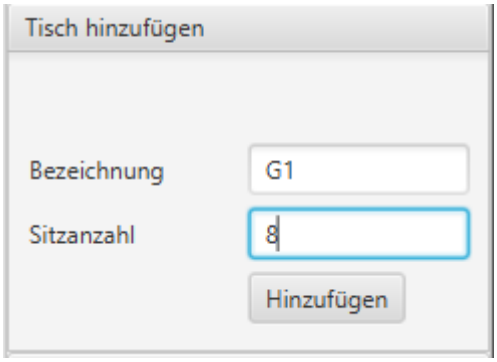
5.1.2 Hinzufügen von Datenbankinhalten

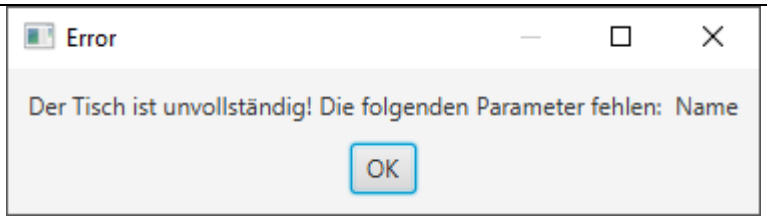
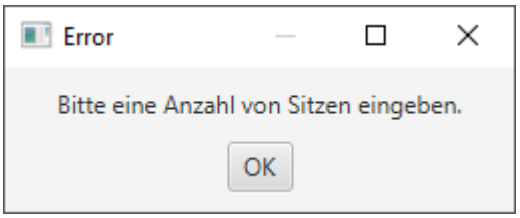
5.1.2.1 Hinzufügen von neuen Artikeln

Anwendungs-fall	Hinzufügen eines neuen Artikels, der neu in das Sortiment/ die Speisekarte aufgenommen wurde (AW 5)
Verwendete Methode	*void addItem(ActionEvent actionEvent)*
Normalablauf	In die Felder werden die Daten des neuen Artikels eingegeben. Es wird der Datenbank ein neuer Eintrag hinzugefügt und anschließend in der tabellarischen Übersicht angezeigt.
Erwartetes Testergebnis	Ein neuer Artikel mit den folgenden Daten: 
Tatsächliches Testergebnis	
Sonderfall	1. Es wurde kein Name angegeben. 2. Es wurde keine Anzahl angegeben.
Erwartetes Testergebnis	1. Anzeige einer Fehlermeldung mit einem fehlenden Namen. 2. Anzeige einer Fehlermeldung mit einer fehlenden Anzahl.
Tatsächliches Testergebnis	*1.* *2.*

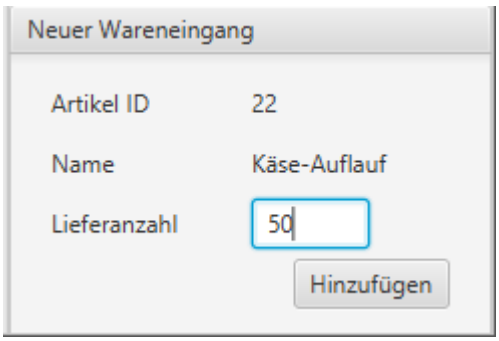
	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.2.2 Hinzufügen von neuen Tischen

Anwendungs-fall	Hinzufügen eines neuen Tisches, der neu im Geschäftsbereich eingerichtet wird (AW 9)
Verwendete Methode	*void addTable(ActionEvent actionEvent)*
Normalablauf	Es werden in das Feld die Daten des neuen Tisches eingegeben. Es wird der Datenbank ein neuer Eintrag hinzugefügt und anschließend in der tabellarischen Übersicht angezeigt.
Erwartetes Testergebnis	Ein Tisch mit den folgenden Daten: 
Tatsächliches Testergebnis	
Sonderfall	1. Es wurde kein Name angegeben. 2. Es wurde keine Sitzanzahl eingegeben.
Erwartetes Testergebnis	1. Fehlermeldung mit der Meldung eines fehlenden Namens. 2. Fehlermeldung mit der Meldung der fehlenden Sitzanzahl.
Tatsächliches	1.

Testergebnis	 2. 
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.2.3 Hinzufügen von neuen Wareneingängen

Anwendungsfall	Hinzufügen eines neuen Wareneingangs. Das wird während des Befüllens des Lagers gemacht (AW 13)
Verwendete Methode	*void addItemdelivery(ActionEvent actionEvent)*
Normalablauf	Ein Artikel wird angeklickt und anschließend die neue Anzahl eingegeben. Es wird der Datenbank ein neuer Eintrag hinzugefügt und anschließend in der tabellarischen Übersicht angezeigt.
Erwartetes Testergebnis	Ein neuer Wareneingang mit den folgenden Daten: 
Tatsächliches Testergebnis	88 | 22 | Käse-Auflauf | 50
Sonderfall	1. Es wurde keine Anzahl eingegeben. 2. Der Warenbestand des Artikels, für den ein Wareneingang erstellt wird, wird durch den neuen Wareneingang negativ.
Erwartetes Testergebnis	1. Fehlermeldung mit der Meldung einer fehlenden Anzahl. 2. Folgende Möglichkeiten bestehen: a) Es existieren mindestens so viele noch nicht zubereitete bestellte Artikel für diesen Artikel wie der Warenbestand negativ ist. In dem Fall wird ein Dialog geöffnet, in dem solange bestellte Artikel entfernt werden müssen, bis der Datenkonflikt behoben wurde. Anschließend wir ein Beleg ausgedruckt, auf dem alle gelöschten bestellten Artikel mit zugehörigem Tisch aufgelistet werden. Der folgende Artikel erhält einen Warenausgang von 37:

11	Burger	4.5	34

Neuer Warenbestand nach Bestätigen: -3

Noch nicht produzierte Burger: 5

Artikel	Preis	Produziert	Bezahlt	Kommentar
Burger	4.5	false	true	
Burger	4.5	false	true	
Burger	4.5	false	true	
Burger	4.5	false	true	
Burger	4.5	false	true	

→Anzeige des Dialoges zum beheben des Datenkonfliktes

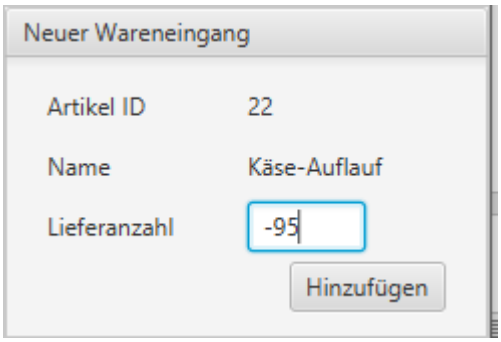
b) Es existieren weniger noch nicht zubereitete bestellte Artikel als der Warenbestand negativ ist.

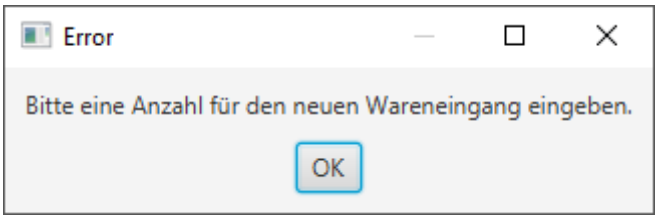
In diesem Fall muss es sich um einen Fehler handeln, da nicht mehr Artikel aus dem Warenbestand entfernt werden können, als überhaupt noch im Warenbestand existieren. Demnach muss entweder eine falsche Eingabe getätigt worden sein oder es wurde in der Vergangenheit ein Wareneingang vergessen, sodass der aktuelle Warenbestand nicht stimmt.

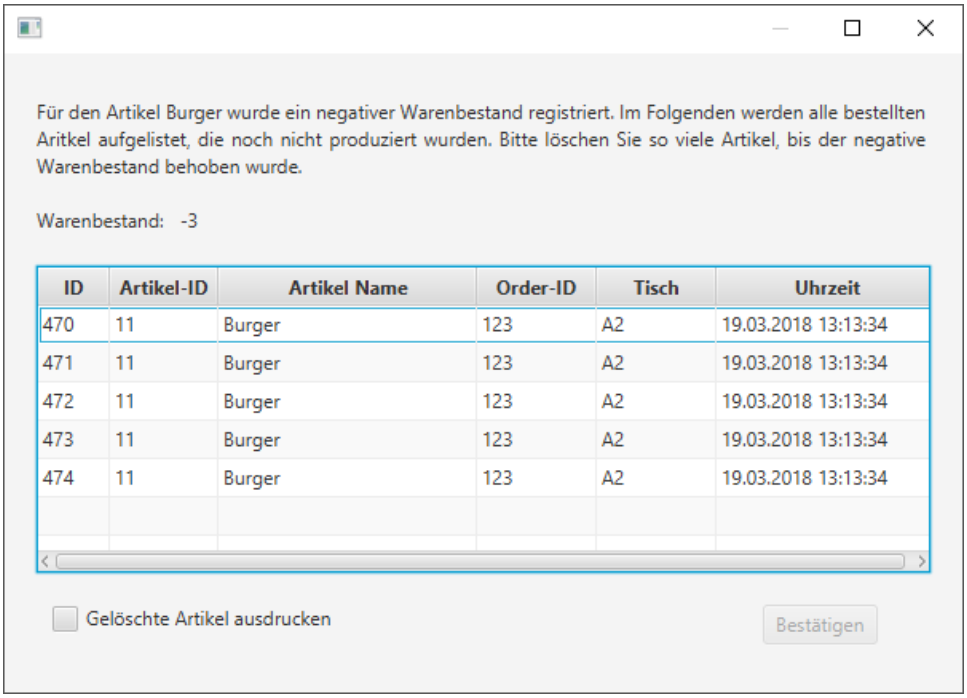
Diese Information wird in Dialogen ausgegeben.

Der folgende Artikel erhält einen Warenausgang von 95:

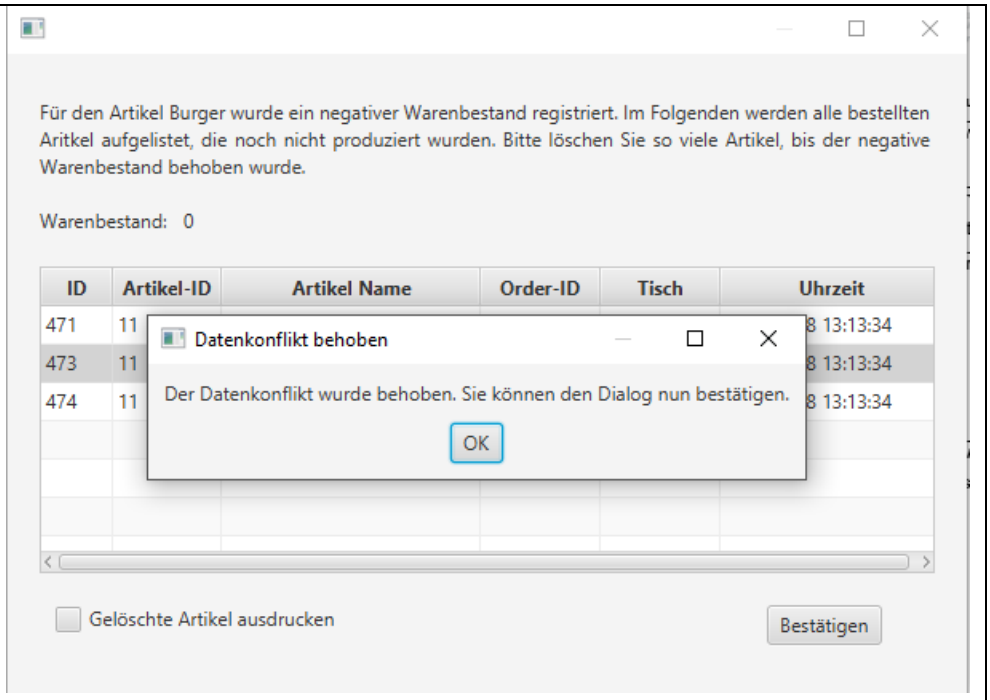
22	Käse-Auflauf	7.99	86



<table>
<tr>
<td></td>
<td>Neuer Warenbestand nach Bestätigen: -9
Noch nicht produzierte Burger: 2
<table>
<tr><th>Artikel</th><th>Preis</th><th>Produziert</th><th>Bezahlt</th><th>Kommentar</th></tr>
<tr><td>Calzone</td><td>8.5</td><td>true</td><td>true</td><td></td></tr>
<tr><td>Käse-Auflauf</td><td>7.99</td><td>true</td><td>true</td><td></td></tr>
<tr><td>Käse-Auflauf</td><td>7.99</td><td>false</td><td>true</td><td></td></tr>
<tr><td>Käse-Auflauf</td><td>7.99</td><td>false</td><td>true</td><td></td></tr>
<tr><td>Calzone</td><td>8.5</td><td>true</td><td>true</td><td></td></tr>
</table>
→Anzeige eines Dialoges, der erklärt, dass ein Fehler vorliegen muss.</td>
</tr>
<tr>
<td>Tatsächliches Testergebnis</td>
<td>1.
 

2.
a)
 

Über Rechtsklick -> „Löschen“ bestellte Artikel entfernen.
Sobald genügen gelöscht (Warenbestand = 0) wurden:</td>
</tr>
</table>



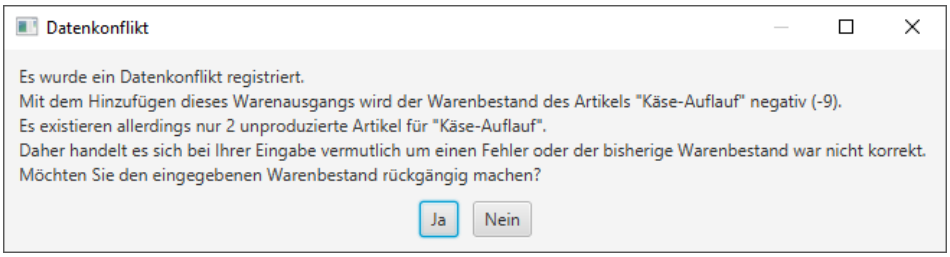
„Gelöschte Artikel ausdrucken“ anwählen und „Bestätigen“ drücken:

```
Die folgenden Tische informieren:
A2      Burger
A2      Burger
A2      Burger
```

Artikel „Burger“ hat nun einen Warenbestand von 0:

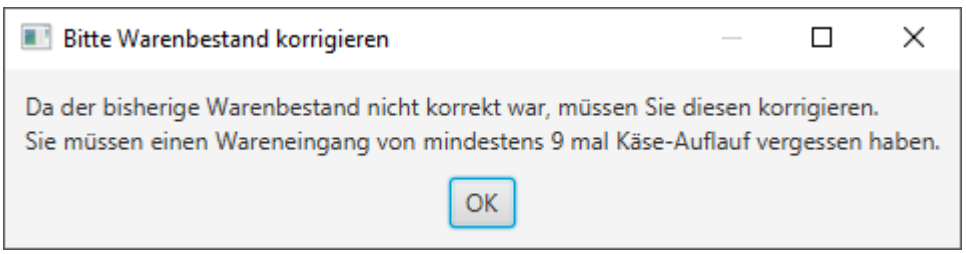
11	Burger	4.5	0

b)



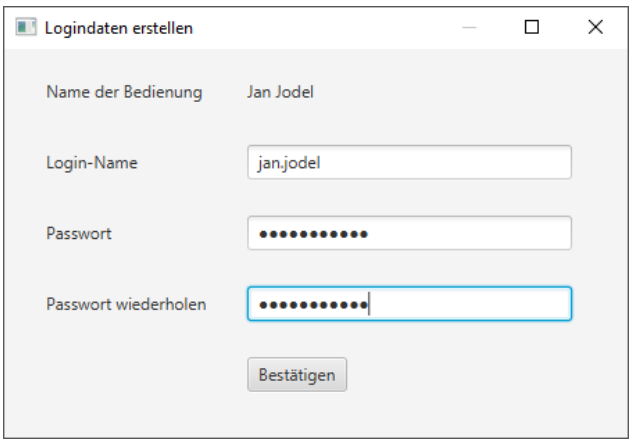
Klick auf „Ja“: Eingegebener Warenausgang wird nicht übernommen.

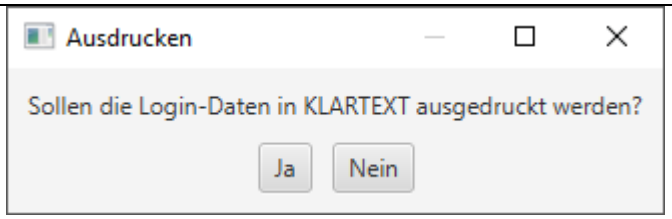
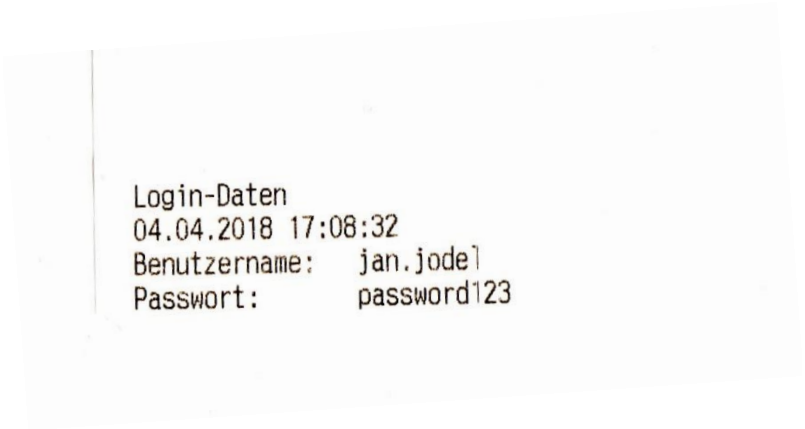
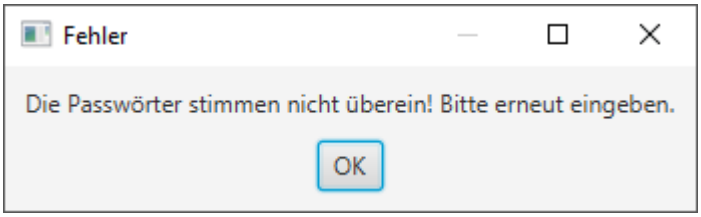
Klick auf „Nein“:

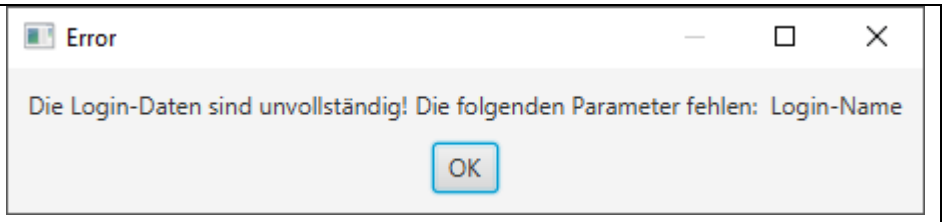
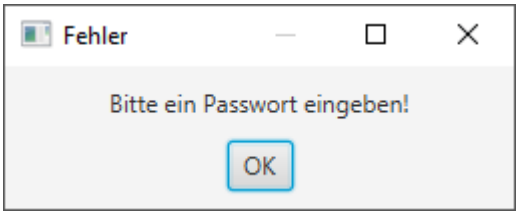
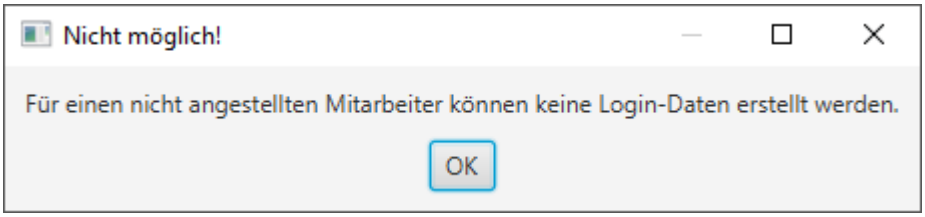


	Eingegebener Warenausgang wird trotzdem übernommen und es entsteht ein negativer Warenbestand. 22 | Käse-Auflauf | 7,99 | -9
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.3 Hinzufügen von Login-Daten

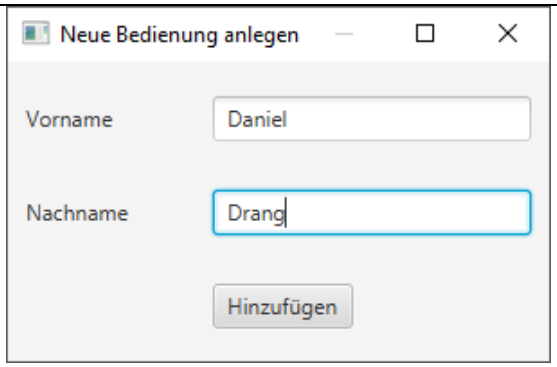
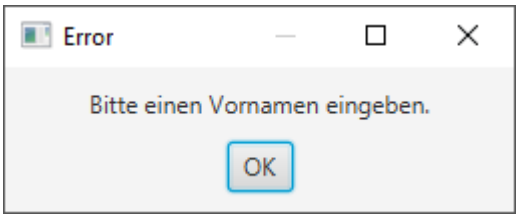
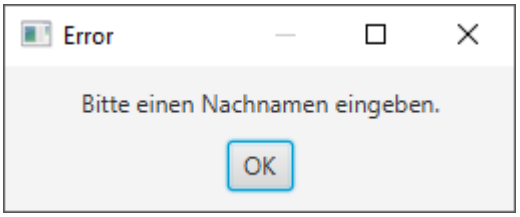
Anwendungs-fall	AW7b - Login-Daten anlegen
Verwendete Methode	*void createLogindata(ActionEvent actionEvent)*
Normalablauf	Mit einem Rechtsklick auf eine Bedienung können neue Login-Daten angelegt werden, solange noch kein Login-Daten-Satz für den Mitarbeiter existiert. Anschließend erscheint ein Dialog, ob der Benutzername und das Passwort in Klartext ausgedruckt werden soll. Wenn gewünscht, werden anschließend mit dem Bon-Drucker ausgedruckt (Passwort ein Klartext).
Erwartetes Testergebnis	Einen neuen Login-Daten-Satz, der ausgedruckt wird, nachdem dies im Dialog bestätigt wurde und der unter Login-Daten bearbeiten eingesehen werden kann. Folgende Login-Daten werden eingegeben:  Passwort: „password123“

Tatsächliches Testergebnis	 Ja:  Nein: Kein Ausdruck
Sonderfall	1. Die Passwörter stimmen nicht überein. 2. Der Benutzername wurde leer gelassen. 3. Das Passwort wurde leer gelassen. 4. Der Mitarbeiter ist nicht mehr angestellt.
Erwartetes Testergebnis	1. Fehlermeldung, dass das Passwort neu eingegeben werden soll. Löschen beider Passwort-Felder. 2. Fehlermeldung, dass ein Benutzername eingegeben werden soll. 3. Fehlermeldung, dass ein Passwort eingegeben werden muss. 4. Fehlermeldung, dass für nicht angestellte Mitarbeiter keine Login-Daten erstellt werden können.
Tatsächliches Testergebnis	1.  2.

	 3.  4. 
Test bestanden	Normalablauf: Ja Sonderfall: Ja

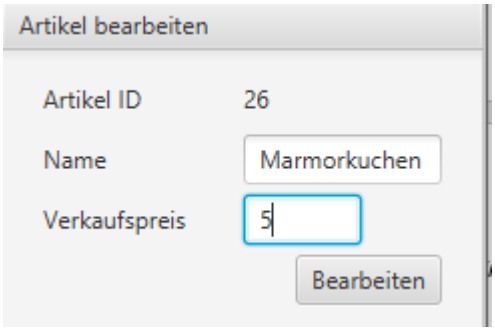
5.1.3.1 Hinzufügen von Mitarbeiter-Daten

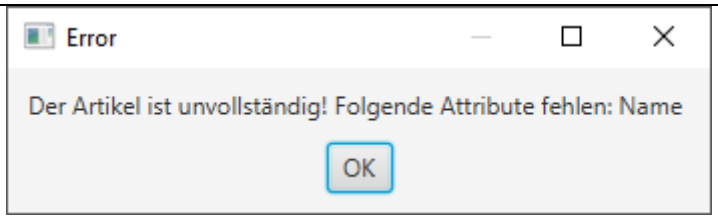
Anwendungs-fall	AW11b – Bedienungs-Mitarbeiterdaten anlegen
Verwendete Methode	*void createNewWaiter(ActionEvent actionEvent)*
Normalablauf	Mit einem Rechtklick in der Mitarbeiter-Tabelle kann der Menüeintrag „Neuen Mitarbeiter anlegen“ ausgewählt werden. Anschließend öffnet sich in Dialog, in dem die Daten des Mitarbeiters eingegeben werden können.
Erwartetes Testergebnis	Einen neuen Mitarbeitereintrag mit den folgenden Daten:

	
Tatsächliches Testergebnis	10 | Daniel | Drang | true
Sonderfall	1. Es wurde kein Vorname eingegeben. 2. Es wurde kein Nachname eingegeben.
Erwartetes Testergebnis	1. Fehlermeldung, dass ein Vorname eingegeben werden soll. 2. Fehlermeldung, dass ein Nachname eingegeben werden soll.
Tatsächliches Testergebnis	*1.*  *2.* 
Test bestanden	Normalablauf: Ja Sonderfall: Ja

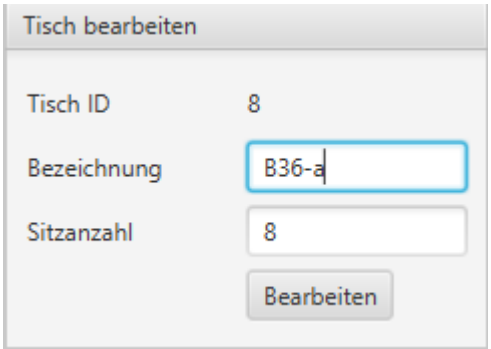
5.1.4 Bearbeiten von Datenbankinhalten

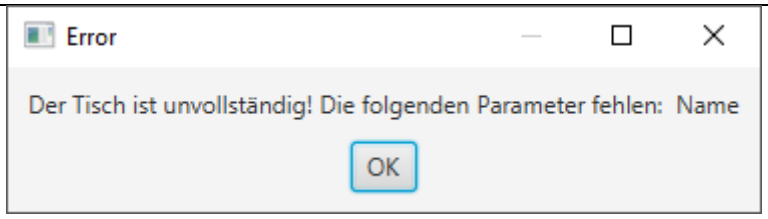
5.1.4.1 Bearbeiten von Artikeln

Anwendungs-fall	Ändern der Daten eines Artikels, wie bspw. Preisänderung (AW 6)
Verwendete Methode	*public void editItem(ActionEvent actionEvent)*
Normalablauf	Ein Artikel wird angeklickt. Im entsprechenden Feld zum Bearbeiten des Artikels erscheinen die aktuellen Daten. Diese können bearbeitet werden. Wenn der „Bearbeiten“-Button gedrückt wird, wird der bisherige Artikel als nicht verfügbar markiert und ein neuer Datenbankeintrag mit den bearbeiteten Daten erzeugt. In der tabellarischen Übersicht wird der bearbeitete Artikel angezeigt.
Erwartetes Testergebnis	Der folgende Artikel soll aktualisiert werden: 26 | Marmorkuchen | 3.33 | 20 Dieser Artikel soll mit den folgenden Daten aktualisiert werden: 
Tatsächliches Testergebnis	In den Daten ist nun der folgende Artikel zu finden: 30 | Marmorkuchen | 5.0 | 20
Sonderfall	Es wird kein Name übergeben.
Erwartetes Testergebnis	Eine Fehlermeldung, die einen fehlenden Namen anmerkt.

Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

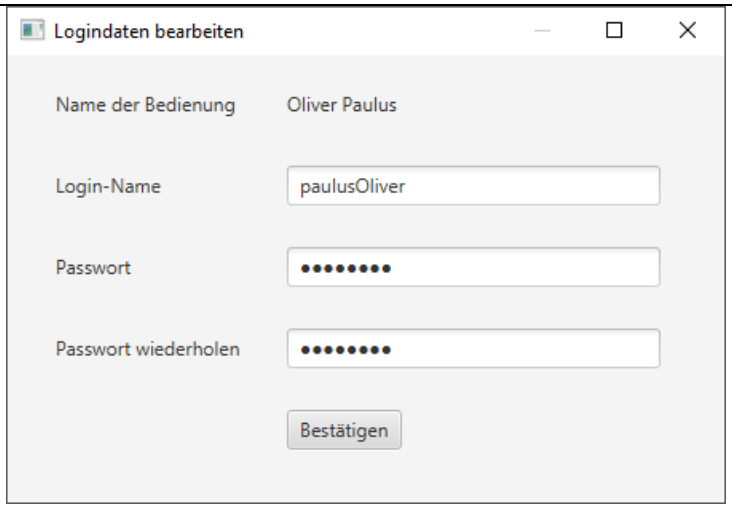
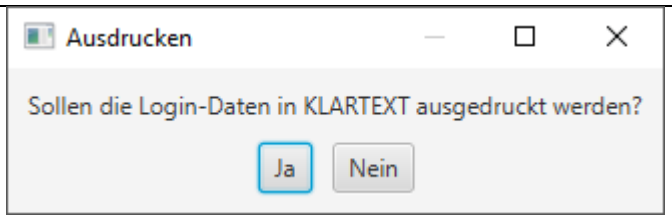
5.1.4.2 Bearbeiten von Tischen

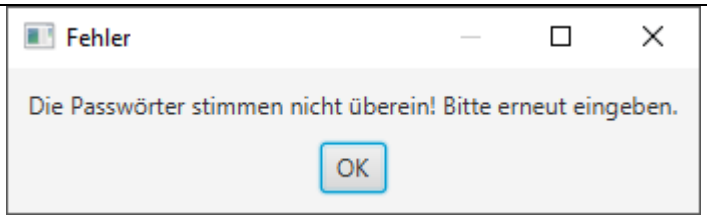
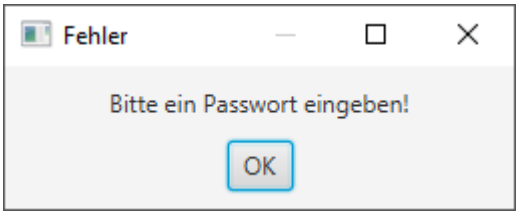
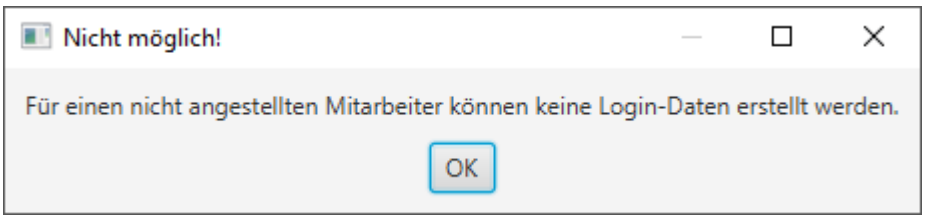
Anwendungs-fall	Ändern der Bezeichnung eines Tisches (AW 10)
Verwendete Methode	*public void editTable(ActionEvent actionEvent)*
Normalablauf	Ein Tisch wird angeklickt. Im entsprechenden Feld zum Bearbeiten des Tisches erscheinen die aktuellen Daten. Diese können bearbeitet werden. Wenn der „Bearbeiten“-Button gedrückt wird, wird der bisherige Tisch als nicht verfügbar markiert und ein neuer Datenbankeintrag mit den bearbeiteten Daten erzeugt. In der tabellarischen Übersicht wird der bearbeitete Tisch angezeigt.
Erwartetes Testergebnis	Der folgende Tisch soll aktualisiert werden: 8 | B36 | 8 Dieser Tisch soll mit den folgenden Daten bearbeitet werden: 
Tatsächliches Testergebnis	In der tabellarischen Übersicht ist nun der folgende Tisch zu finden:

	14 B36-a 8
Sonderfall	Es wurde kein Name angegeben.
Erwartetes Testergebnis	Eine Fehlermeldung über den fehlenden Namen.
Tatsächliches Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.4.3 Bearbeiten von Login-Daten

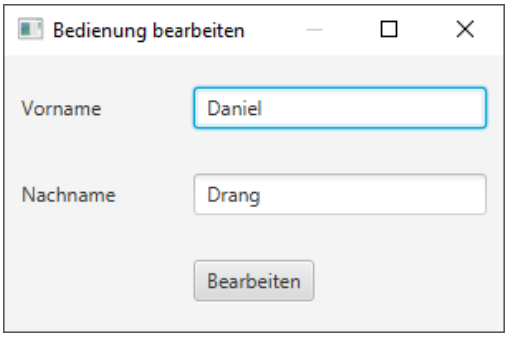
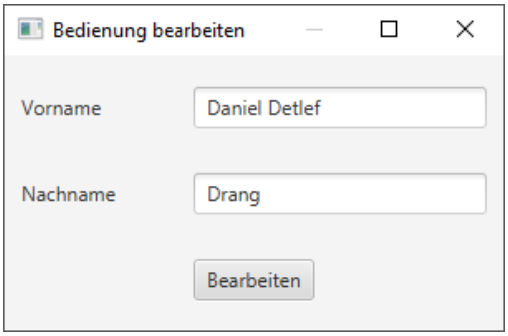
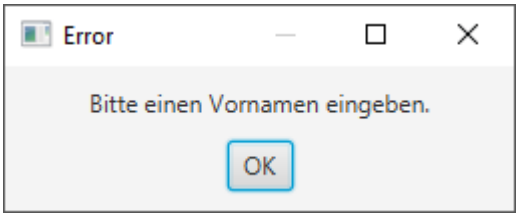
Anwendungs-fall	AW9b – Login-Daten bearbeiten
Verwendete Methode	*void editLogindata(ActionEvent actionEvent)*
Normalablauf	Mit einem Rechtsklick auf einen Mitarbeiter in der Mitarbeiter-Tabelle und auswählen des Menüeintrags „Login-Daten bearbeiten“ wird ein Dialog geöffnet. In diesem können die Login-Daten bearbeitet werden. Eine Eingabe des alten Passwortes ist hier nicht notwendig.
Erwartetes Testergebnis	Ein Ausdruck der folgenden Login-Daten:

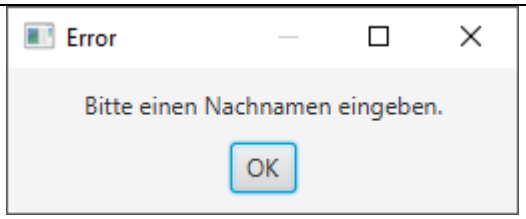
	 Passwort: „wordpass“
Tatsächliches Testergebnis	 Klick auf „Ja“: Login-Daten 04.04.2018 18:16:45 Benutzername: paulusOliver Passwort: wordpass Klick auf „Nein“: Kein Ausdruck
Sonderfall	5. Die Passwörter stimmen nicht überein. 6. Der Benutzername wurde leer gelassen. 7. Das Passwort wurde leer gelassen. 8. Der Mitarbeiter ist nicht mehr angestellt.
Erwartetes Testergebnis	5. Fehlermeldung, dass das Passwort neu eingegeben werden soll. Löschen beider Passwort-Felder. 6. Fehlermeldung, dass ein Benutzername eingegeben werden soll. 7. Fehlermeldung, dass ein Passwort eingegeben werden muss. 8. Fehlermeldung, dass für nicht angestellte Mitarbeiter keine Login-Daten erstellt werden können.
Tatsächliches	1.

Testergebnis	 2. 3.  4. 
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.4.4 Bearbeiten von Mitarbeiter-Daten

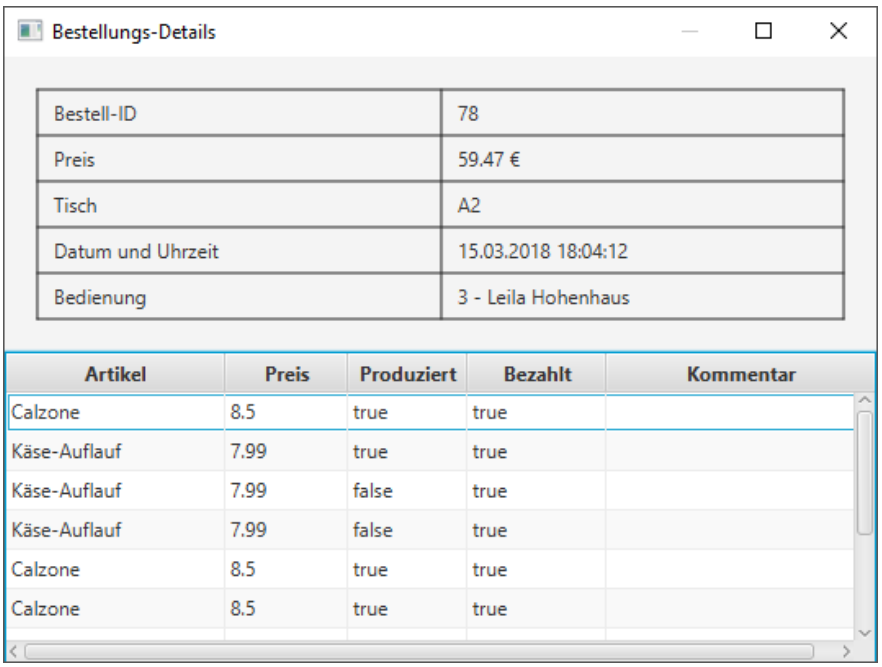
Anwendungs-fall	AW13b – Bedienungs-Mitarbeiterdaten bearbeiten
Verwendete Methode	*void editWaiter(ActionEvent actionEvent)*
Normalablauf	Es öffnet sich ein Dialog, in dem die Daten eines Mitarbeiters bearbeitet werden können. In der Datenbank haben sich die Daten geändert.

Erwartetes Testergebnis	Folgender Mitarbeiter wird bearbeitet:  Folgende Daten werden eingegeben: 
Tatsächliches Testergebnis	Der folgende Eintrag befindet sich nun in der Mitarbeiter-Tabelle: 10 | Daniel Detlef | Drang | true
Sonderfall	3. Es wurde kein Vorname eingegeben. 4. Es wurde kein Nachname eingegeben.
Erwartetes Testergebnis	3. Fehlermeldung, dass ein Vorname eingegeben werden soll. 4. Fehlermeldung, dass ein Nachname eingegeben werden soll.
Tatsächliches Testergebnis	*1.*  *2.*

	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.5 Löschen von Datenbankinhalten

5.1.5.1 Löschen von Bestellungen

Anwendungs-fall	Löschen einer fehlerhaften oder überschüssigen Bestellung (AW 2)
Verwendete Methode	*public void deleteOrder(ActionEvent actionEvent)*
Normalablauf	Eine Bestellung wird über einen Rechtsklick ausgewählt und der Menüeintrag zum Löschen ausgewählt. Der Eintrag der Bestellung verschwindet aus der tabellarischen Übersicht und ist nicht mehr in der Datenbank zu finden.
Erwartetes Testergebnis	Der Eintrag der Bestellung wird aus der tabellarischen Übersicht entfernt. Die folgende Bestellung soll gelöscht werden: 
Tatsächliches Testergebnis	Der Eintrag existiert nicht mehr in der Anwendung.
Sonderfall	keiner
Erwartetes Testergebnis	--
Tatsächliches Testergebnis	--

Test bestanden	Normalablauf: Ja Sonderfall: --

5.1.5.2 Löschen von Artikeln

Anwendungs-fall	Löschen eines Artikels der aus dem Sortiment genommen wurde (AW 4)
Verwendete Methode	*public void deleteItem(ActionEvent actionEvent)*
Normalablauf	Ein Artikel wird über einen Rechtsklick ausgewählt und der Menüeintrag zum Löschen ausgewählt. Der Eintrag der Bestellung verschwindet aus der tabellarischen Übersicht und in der Datenbank wird der Artikel als nicht verfügbar markiert.
Erwartetes Testergebnis	Der folgende Artikel soll gelöscht werden: 20 ColaZero 2.5 165
Tatsächliches Testergebnis	Der Eintrag existiert nicht mehr in der Anwendung.
Sonderfall	keiner
Erwartetes Testergebnis	--
Tatsächliches Testergebnis	--
Test bestanden	Normalablauf: Ja Sonderfall: --

5.1.5.3 Löschen von Tischen

Anwendungs-fall	Löschen eines Tisches, der von der Verkaufsfläche entfernt wurde (AW 8)
Verwendete Methode	*public void deleteTable(ActionEvent actionEvent)*

Normalablauf	Ein Tisch wird über einen Rechtsklick ausgewählt und der Menüeintrag zum Löschen ausgewählt. Der Eintrag des Tisches verschwindet aus der tabellarischen Übersicht und in der Datenbank wird der Tisch als nicht verfügbar markiert.
Erwartetes Testergebnis	Der folgende Tisch soll gelöscht werden: 14 | B36-a | 8
Tatsächliches Testergebnis	Der Eintrag existiert nicht mehr in der Anwendung.
Sonderfall	keiner
Erwartetes Testergebnis	--
Tatsächliches Testergebnis	--
Test bestanden	Normalablauf: Ja Sonderfall: --

5.1.5.4 Löschen von Wareneingängen

Anwendungsfall	Ein Wareneingang soll gelöscht werden, der bspw. fälschlicherweise angelegt wurde. (AW 14)
Verwendete Methode	public void deleteItemdelivery(ActionEvent actionEvent)
Normalablauf	Ein Wareneingang wird über einen Rechtsklick ausgewählt und der Menüeintrag zum Löschen ausgewählt. Der Eintrag des Wareneingangs verschwindet aus der tabellarischen Übersicht und ist nicht mehr in der Datenbank zu finden.
Erwartetes Testergebnis	Der folgende Wareneingang soll gelöscht werden: 89 | 30 | Marmorkuchen | 20 In der tabellarischen Übersicht der Artikel wird die Anzahl des entsprechenden Artikels reduziert:

	30 Marmorkuchen 5.0 43
Tatsächliches Testergebnis	Der Eintrag existiert nicht mehr in der Anwendung. Die Anzahl des Artikels in der Artikel-Übersicht wurde aktualisiert: 30 Marmorkuchen 5.0 23
Sonderfall	keiner
Erwartetes Testergebnis	--
Tatsächliches Testergebnis	--
Test bestanden	Normalablauf: Ja Sonderfall: --

5.1.5.5 Löschen von Login-Daten

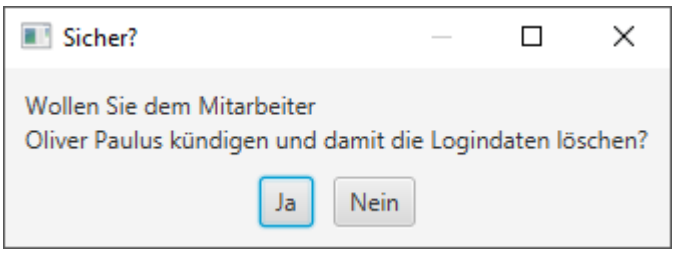
AW10b – Login-Daten löschen kann nur über das Kündigen von Mitarbeitern ausgeführt werden und wird somit in AW114b implizit mit getestet.

5.1.5.6 Löschen von Mitarbeiter-Daten

Anwendungs-fall	AW14b – Bedienungs-Mitarbeiterdaten löschen
Verwendete Methode	*void unemployWaiter(ActionEvent actionEvent)*
Normalablauf	Über Rechtsklick → „Kündigen“ wird ein Bestätigungs-Dialog aufgerufen. Wird dieser positiv bestätigt wird anschließend das „Angestellt“-Attribut auf negativ gesetzt.
Erwartetes Testergebnis	Dem folgenden Mitarbeiter soll gekündigt werden: 4 Oliver Paulus true

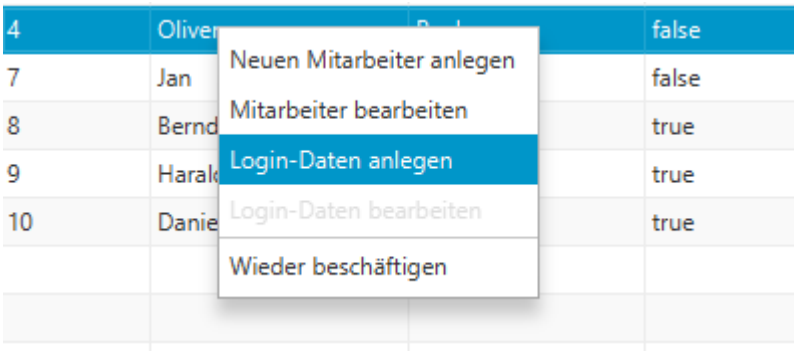
<table>
<tr><td>Tatsächliches Testergebnis</td><td>

Klick auf „Kündigen“:

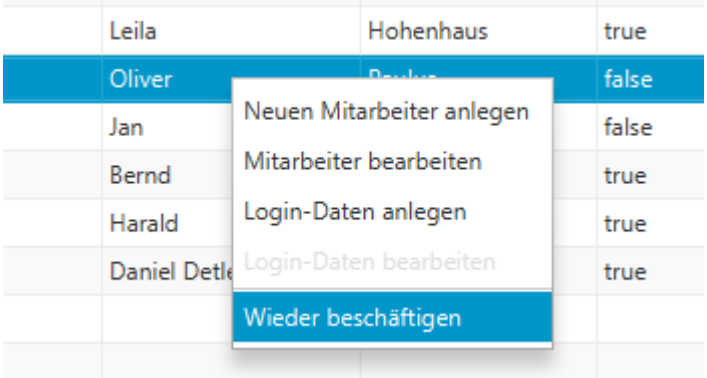
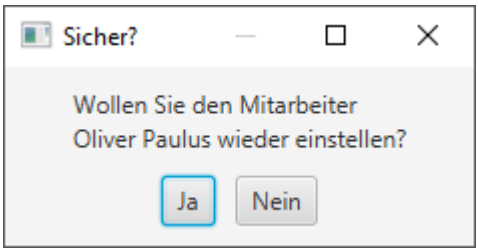
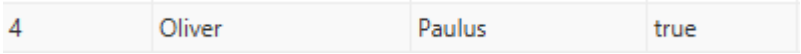


Klick auf „Ja“:

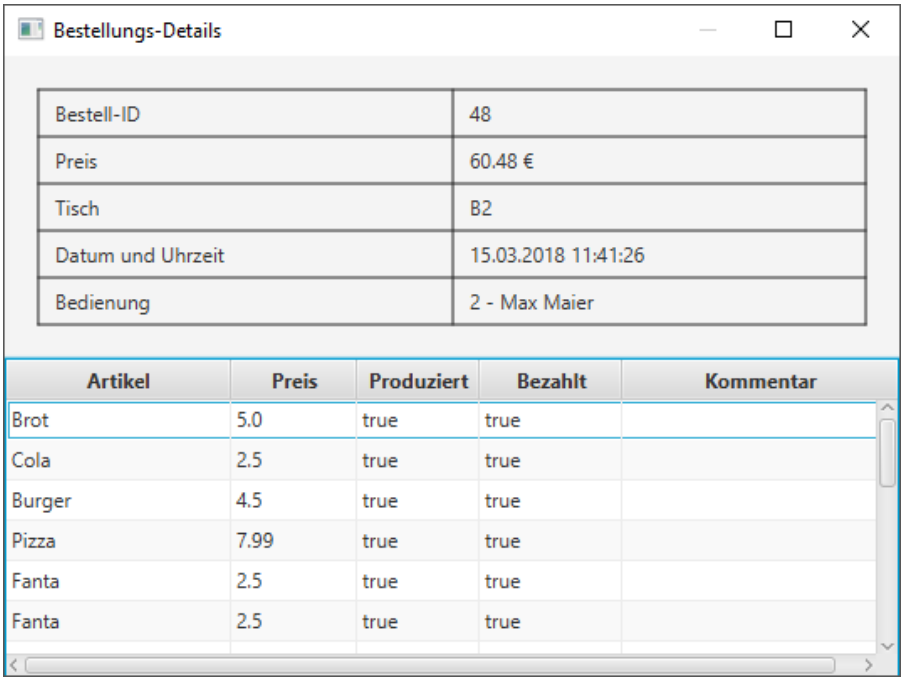
Login-Daten wurden gelöscht:



Klick auf „Nein“: Daten werden nicht geändert.
</td></tr>
<tr><td>Sonderfall</td><td>1. Der Mitarbeiter ist nicht angestellt.</td></tr>
<tr><td>Erwartetes Testergebnis</td><td>1. Es ist möglich diesen wieder anzustellen. Es wird ein Dialog angezeigt, in dem um Bestätigung gebeten wird.</td></tr>
</table>

Tatsächliches Testergebnis	 *Klick auf „Wieder beschäftigen“:*  *Klick auf „Ja“:*  *Klick auf „Nein“: Daten werden nicht geändert.*
Test bestanden	Normalablauf: Ja Sonderfall: Ja

5.1.6 Ausdrucken einer Bestellung

Anwendungs-fall	Nachträgliches Ausdrucken eines Belegs, nachdem der ursprüngliche Beleg verloren gegangen oder zerstört bzw. verschmutzt wurde (AW 12)
Verwendete Methode	*public void printOrder(ActionEvent actionEvent)*
Normalablauf	Über einen Rechtsklick wird eine Bestellung ausgewählt und der Menüeintrag zum Ausdrucken angeklickt. Über den Bondrucker wird der Kundenbeleg ausgedruckt.
Erwartetes Testergebnis	Der Bondrucker druckt einen Kundenbeleg mit den folgenden Bestellungsdaten: 
Tatsächliches Testergebnis	Ein Ausdruck wurde ausgegeben:

Bestellungs-Details

Bestell-ID	48
Preis	60.48 €
Tisch	B2
Datum und Uhrzeit	15.03.2018 11:41:26
Bedienung	2 - Max Maier

Artikel	Preis	Produziert	Bezahlt	Kommentar
Brot	5.0	true	true	
Cola	2.5	true	true	
Burger	4.5	true	true	
Pizza	7.99	true	true	
Fanta	2.5	true	true	
Fanta	2.5	true	true	

	 ----------Kundenbeleg---------- Restaurante Gaumenfreude Gourmetstraße 11 49082 Leckerschmeckerhausen +49 541 123 456 Ihre Bestellung: Brot 5,00 EUR Cola 2,50 EUR Burger 4,50 EUR Pizza 7,99 EUR Fanta 2,50 EUR Fanta 2,50 EUR Fanta 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR ColaZero 2,50 EUR Käse-Auflauf 7,99 EUR --- Summe 60,48 EUR inkl. MWST 19% 12,04 EUR Sie saßen an Tisch B2. Vielen Dank für Ihren Besuch! 15.03.2018 11:41:26
Sonderfall	Der Drucker ist nicht angeschlossen oder abgeschaltet.
Erwartetes Testergebnis	Der Beleg wird ausgedruckt, sobald der Drucker erreichbar ist.
Tatsächliches Testergebnis	Nach dem Abschalten und wieder Anschalten des Drucker wird der Beleg wie erwartet ausgedruckt.
Test bestanden	Normalablauf: Ja Sonderfall: Ja

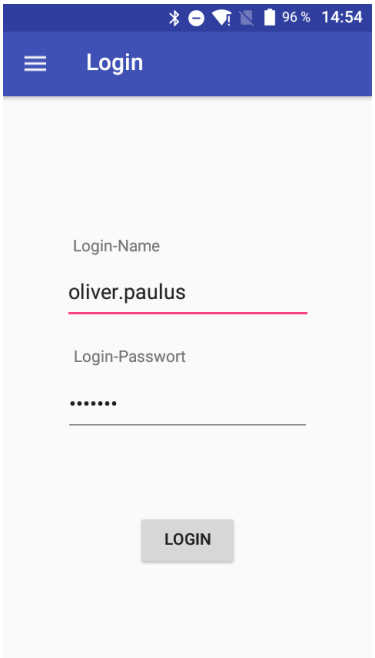
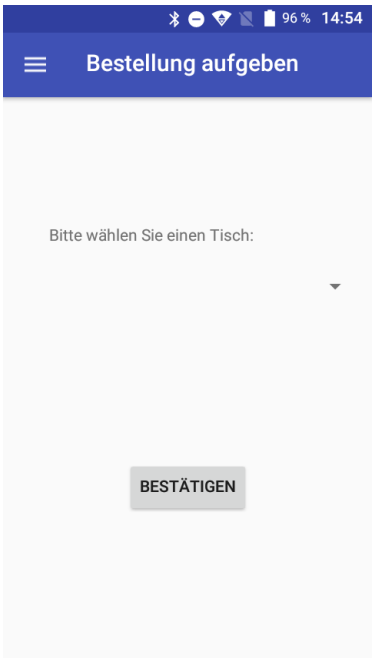
5.2 Test der Android-Anwendung

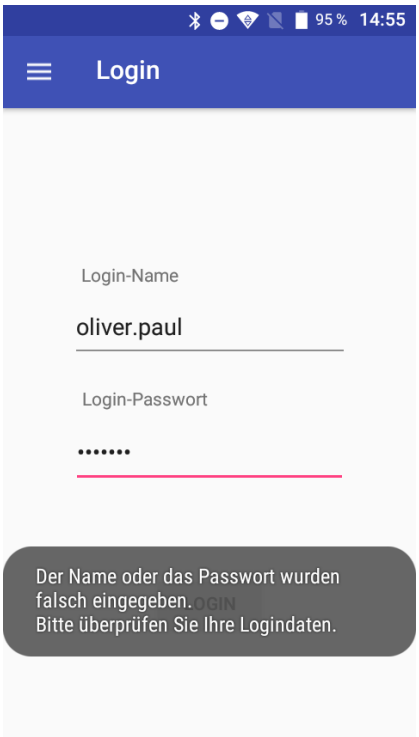
Im Folgenden sind die Tests der Android-Anwendung dokumentiert.

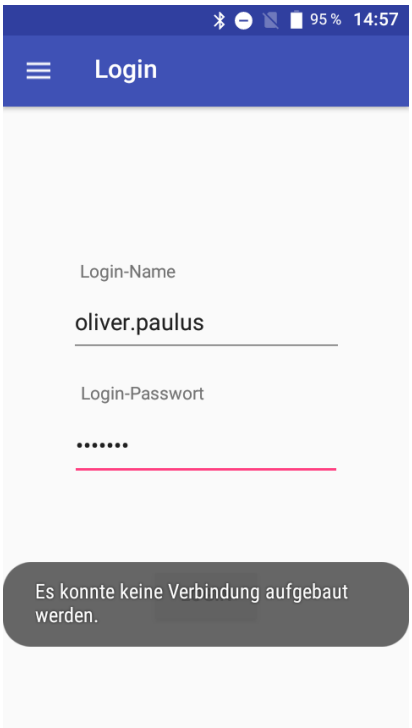
Es wird jeweils der Anwendungsfall und die getestete Methode beschrieben, außerdem Normalablauf und Sonderfälle, bspw. einem Laufzeitfehler wie Verbindungsprobleme oder falsche Eingaben. Anschließend wird das zu erwartenden und das tatsächliche Testergebnis für den Normalablauf und den Sonderfall dokumentiert. In einigen Fällen wurden die Testergebnisse gekürzt.

Mitarbeiterlogin:

<table>
<tr><td>Anwendungs-fall</td><td>Die Bedienung kann sich im System mit ihrem Login-Namen und Login-Passwort authentifizieren. Diese Daten müssen sich zuvor in der Datenbank befinden.
(AW 1b)</td></tr>
<tr><td>Verwendete Methode</td><td>doInBackground()
onPostExecute() (Klasse: “LoginCheck”)
showToast (String text)</td></tr>
<tr><td>Normalablauf</td><td>Wird „Login“ im Navigation-Drawer ausgewählt, wird ein Bildschirm zum Anmelden im System angezeigt. Um sich anzumelden müssen der Login-Name und das Login-Passwort eingegeben werden.
Stimmen die Daten mit den hinterlegten Daten in der Datenbank überein, dann ist der Mitarbeiter erfolgreich angemeldet.
Der Startbildschirm wird angezeigt. (Bildschirm: Bestellung aufgeben).
(Nach dem ersten Anmelden bleiben die Daten gespeichert).</td></tr>
<tr><td>Erwartetes Testergebnis</td><td>Die Bedienung hat sich erfolgreich im System authentifiziert und kann auf alle Funktionen zugreifen.
Datenbank-Eintrag der Login-Daten für die Bedienungen:
<table>
<tr><th>waiterID</th><th>loginname</th><th>passwordhash</th></tr>
<tr><td>1</td><td>hans.müller</td><td>3329</td></tr>
<tr><td>2</td><td>marie.maier</td><td>3003444</td></tr>
<tr><td>4</td><td>oliver.paulus</td><td>3556498</td></tr>
<tr><td>7</td><td>ian.iodel</td><td>99043158</td></tr>
<tr><td>8</td><td>bernd.auer</td><td>-1217363389</td></tr>
<tr><td>9</td><td>harald.brecht</td><td>3556498</td></tr>
</table></td></tr>
</table>

	Eingabe in der Android-Anwendung: 
Tatsächliches Testergebnis	Die Bedienung hat sich erfolgreich im System authentifiziert und kann auf alle Funktionen zugreifen. Sie wird auf den Startbildschirm weitergeleitet. 
Sonderfall 1:	Die Bedienung hat falsche Login-Daten eingegeben. Das heißt, das Login-Passwort und/oder der Login-Name sind falsch eingegeben worden.

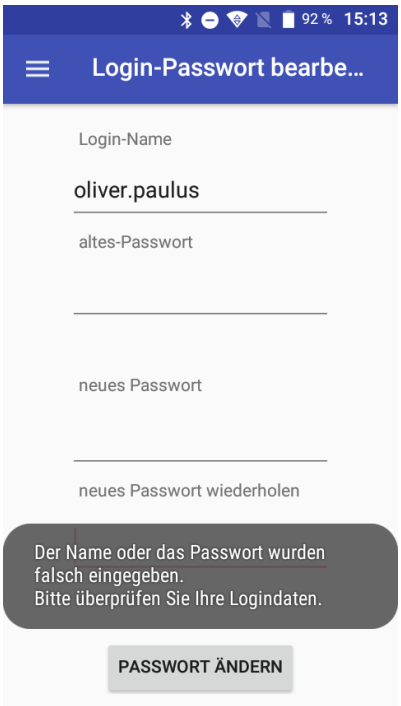
Erwartetes Testergebnis	Die Bedienung kann sich im System nicht authentifizieren und hat somit keinen Zugriff auf die restlichen Funktionen. Für die Bedienung erscheint eine Fehlermeldung, dass die Login-Daten fehlerhaft sind. So kann sie erneut versuchen, die korrekten Login-Daten anzugeben. Um den Login-Vorgang abzubrechen, muss im Navigations-Drawer der gewünschte Bildschirm ausgewählt werden. Mit der „Rückgängig“-Taste kommt man ebenfalls auf den Startbildschirm. (Bildschirm: Bestellung aufgeben). Für diesen Test bleibt die Datenbank dieselbe wie im Normalfall dargestellt.
Tatsächliches Testergebnis	Der Login-Name (wie hier im Beispiel) oder das Passwort ist falsch. Folgender Bildschirm wird für die Bedienung sichtbar:  Die Fehlermeldung wird angezeigt. Die Bedienung kann Ihre Angaben korrigieren.
Sonderfall 2:	Es besteht keine Verbindung zur Datenbank. Die Bedienung kann sich somit nicht im System authentifizieren.

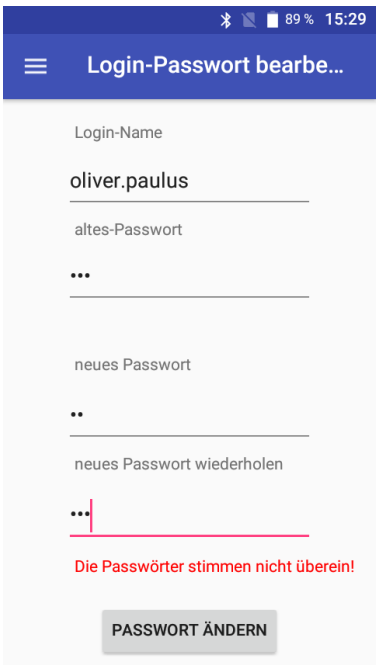
Erwartetes Testergebnis	Die Bedienung kann sich im System nicht authentifizieren und hat somit keinen Zugriff auf die restlichen Funktionen. Für die Bedienung erscheint eine Fehlermeldung, dass die Verbindung zur Datenbank fehlgeschlagen ist.
Tatsächliches Testergebnis	Folgender Bildschirm wird für die Bedienung sichtbar:  Nachdem die Verbindung zur Datenbank wiederhergestellt worden ist, kann sich die Bedienung erneut anmelden.
Test bestanden	Normalablauf: Ja Sonderfall 1: Ja Sonderfall 2: Ja

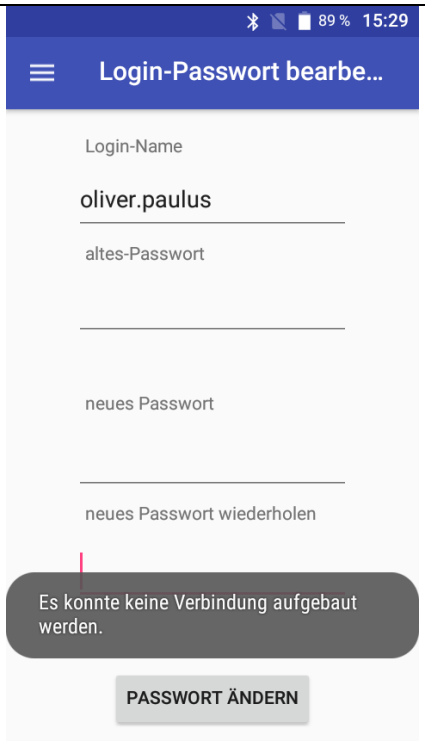
Login-Passwort ändern:

Anwendungs-fall	Nachdem der Gastronom die Login-Daten für eine neu eingestellte Bedienung angelegt hat, sollte das Passwort geändert werden. So kann sichergestellt werden, dass nur die Bedienung auf die Login-Daten Zugriff hat. (AW 2b)
Verwendete Methode	*doInBackground()* *onPostExecute()* (Klasse: *"ChangeLoiginPassword"*) showToast (String text)
Normalablauf	Damit das Passwort geändert werden kann, muss zunächst im Navigation-Drawer „Login-Passwort bearbeiten" ausgewählt werden. Danach wird ein Bildschirm angezeigt, in der die Bedienung die Passwort-Änderung vornehmen kann. Dafür muss der Login-Name, das alte Login-Passwort und das neue - gewünschte - Passwort eingegeben werden. (Das neue Passwort muss wiederholt werden, sodass es zu keinem unbewussten Schreibfehler innerhalb des Passworts kommen kann). Nach erfolgreichem Bearbeiten des Passworts wird der Startbildschirm dargestellt. (Bildschirm: Bestellung aufgeben).
Erwartetes Testergebnis	Bei korrekter Eingabe der alten und neuen Login-Daten wird das neue Passwort in der Datenbank hinterlegt. Das alte Passwort wird überschrieben.
Tatsächliches Testergebnis	Die Datenbank wird mit dem neuen Login-Passwort aktualisiert. Die Bedienung kann sich nur noch mit dem neuen Passwort anmelden. Die Datenbank sieht wie folgt aus:

waiterID	loginname	passwordhash
1	hans.müller	3329
2	marie.maier	3003444
4	oliver.paulus	110251487
7	jan.jodel	99043158
8	bernd.auer	-1217363389
9	harald.brecht	3556498

Sonderfall 1	Der Login-Name oder das Login-Passwort wurden von der Bedienung falsch eingegeben. Somit kann das Login-Passwort nicht geändert werden.
Erwartetes Testergebnis	Das Passwort in der Datenbank wird nicht aktualisiert. Der Bedienung wird eine Fehlermeldung angezeigt, dass die von ihr eingegebenen Login-Daten fehlerhaft sind. Die Eingaben für die Passwörter werden gelöscht. So wird die Bedienung gezwungen, diese nochmals einzugeben. Anschließend kann die Bedienung erneut versuchen, ihr Login-Passwort zu ändern.
Tatsächliches Testergebnis	Die folgende Fehlermeldung wird der Bedienung angezeigt.  Die Datenbank wurde nicht aktualisiert. waiterID | loginname | passwordhash 1 | hans.müller | 3329 2 | marie.maier | 3003444 4 | oliver.paulus | 3556498 7 | jan.jodel | 99043158 8 | bernd.auer | -1217363389 9 | harald.brecht | 3556498
Sonderfall 2	Die neuen Passwörter stimmen nicht überein.

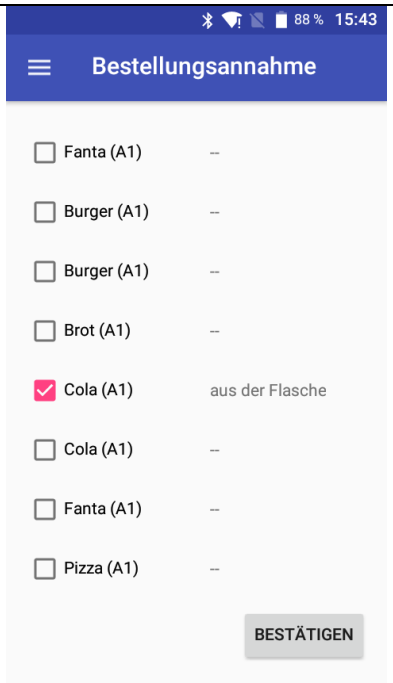
Erwartetes Testergebnis	Das Passwort wird in der Datenbank nicht aktualisiert. Der Bedienung wird in Rot eine Fehlermeldung angezeigt, dass die Passwörter nicht übereinstimmen. Die Änderung der Passwörter kann nicht erfolgen.
Tatsächliches Testergebnis	Die Datenbank wird nicht aktualisiert: <table><tr><th>waiterID</th><th>loginname</th><th>passwordhash</th></tr><tr><td>1</td><td>hans.müller</td><td>3329</td></tr><tr><td>2</td><td>marie.maier</td><td>3003444</td></tr><tr><td>4</td><td>oliver.paulus</td><td>3556498</td></tr><tr><td>7</td><td>jan.jodel</td><td>99043158</td></tr><tr><td>8</td><td>bernd.auer</td><td>-1217363389</td></tr><tr><td>9</td><td>harald.brecht</td><td>3556498</td></tr></table> Der Folgende Bildschirm wird dargestellt: 
Sonderfall 3	Es besteht keine Verbindung zur Datenbank
Erwartetes Testergebnis	Es wird beim Betätigen des „PASSWORT ÄNDERN“-Buttons eine Fehlermeldung dargestellt. Die Datenbank kann nicht aktualisiert werden. Nach erneuter Verbindung mit der Datenbank kann das Passwort geändert werden.
Tatsächliches Testergebnis	Die folgende Fehlermeldung wird der Bedienung dargestellt.

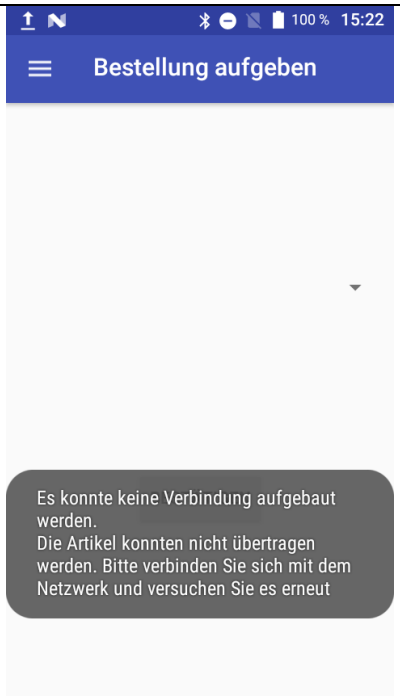
<table>
<tr>
<td></td>
<td>


Die Datenbank wurde nicht aktualisiert.
<table>
<tr><th>waiterID</th><th>loginname</th><th>passwordhash</th></tr>
<tr><td>1</td><td>hans.müller</td><td>3329</td></tr>
<tr><td>2</td><td>marie.maier</td><td>3003444</td></tr>
<tr><td>4</td><td>oliver.paulus</td><td>3556498</td></tr>
<tr><td>7</td><td>jan.jodel</td><td>99043158</td></tr>
<tr><td>8</td><td>bernd.auer</td><td>-1217363389</td></tr>
<tr><td>9</td><td>harald.brecht</td><td>3556498</td></tr>
</table>
</td>
</tr>
<tr>
<td>Test bestanden</td>
<td>Normalablauf: Ja
Sonderfall 1: Ja
Sonderfall 2: Ja
Sonderfall 3: Ja</td>
</tr>
</table>

Artikel als produziert markieren:

Anwendungs-fall	In der Datenbank soll vermerkt werden, dass ein Artikel von der Bedienung abgeholt wurde. Anschließend soll der Artikel zum jeweiligen Kunden gebracht werden. Dadurch kann im Falle eines Fehlers identifiziert werden, welcher Artikel bereits an einen Kunden geliefert wurde. (AW 3b)
Verwendete Methode	*doInBackground()* *onPostExecute()* (Klasse: *"UpdateOrder"*) *showToast (String text)*
Normalablauf	Damit der entsprechende Artikel „markiert" werden kann muss im Navigation-Drawer „Bestellungsannahme" ausgewählt werden. In dem dann angezeigten Bildschirm werden alle nicht produzierten Artikel aufgelistet. Dabei wird der Artikelname, -kommentar und der dazugehörige Tisch angezeigt. Der von der Bedienung angenommene Artikel kann in dieser Darstellung ausgewählt werden. Mit anschließendem Klicken auf den „BESTÄTIGEN"-Button werden die ausgewählten Artikel in der Datenbank als produziert hinterlegt.
Erwartetes Testergebnis	Die Artikel werden in der Datenbank als „produziert" markiert. Die Datenbank sieht wie folgt aus: orderedItemID, orderID, itemID, itemPaid, itemProduced, comment 194, 78, 2, 0, 0, NULL 195, 78, 2, 0, 0, NULL 196, 78, 2, 0, 0, NULL 197, 90, 1, 1, 0, NULL 198, 90, 1, 1, 0, NULL 199, 92, 17, 1, 0, NULL 200, 94, 11, 0, 0, NULL 201, 94, 11, 0, 0, NULL 202, 94, 1, 0, 0, NULL 203, 94, 2, 0, 0, aus der Flasche 204, 94, 2, 0, 0, NULL 205, 94, 17, 0, 0, NULL 206, 94, 16, 0, 0, NULL Im Test wird der Artikel mit der „orderedItemID" *203 markiert.* Im Beispiel der Artikel Cola an Tisch *A1* mit dem Kommentar *aus der Flasche*:

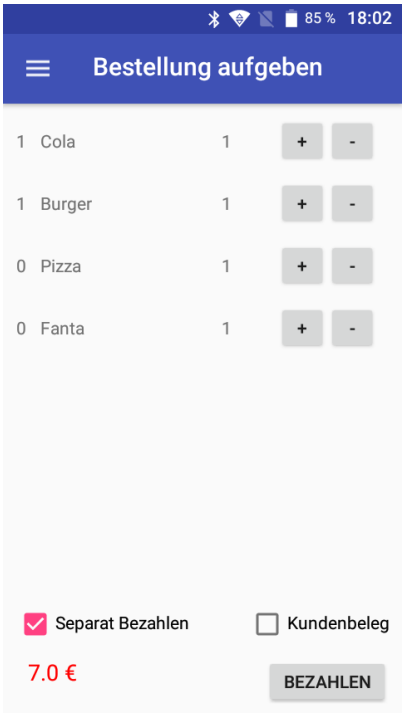
<table>
<tr><td></td><td> 
</td></tr>
<tr><td>Tatsächliches Testergebnis</td><td>Der Artikel mit der „orderedItemID“ 203 wurde als produziert hinterlegt.
Die Datenbank sieht wie folgt aus:
<table>
<tr><th>orderedItemID</th><th>orderID</th><th>itemID</th><th>itemPaid</th><th>itemProduced</th><th>comment</th></tr>
<tr><td>194</td><td>78</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>195</td><td>78</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>196</td><td>78</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>197</td><td>90</td><td>1</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>198</td><td>90</td><td>1</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>199</td><td>92</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>200</td><td>94</td><td>11</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>201</td><td>94</td><td>11</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>202</td><td>94</td><td>1</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>203</td><td>94</td><td>2</td><td>0</td><td>1</td><td>aus der Flasche</td></tr>
<tr><td>204</td><td>94</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>205</td><td>94</td><td>17</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>206</td><td>94</td><td>16</td><td>0</td><td>0</td><td>NULL</td></tr>
</table>
Die Daten wurden erfolgreich in der Datenbank eingetragen.</td></tr>
<tr><td>Sonderfall</td><td>Es besteht keine Verbindung zur Datenbank. Die Daten können somit nicht aktualisiert werden.</td></tr>
<tr><td>Erwartetes Testergebnis</td><td>Der Bedienung wird eine Fehlermeldung dargestellt, dass keine Verbindung zur Datenbank existiert.
Anschließend wird der Startbildschirm der Android-Anwendung dargestellt. Die Bedienung muss nach erfolgreichem Wiederverbinden mit dem Netzwerk die Artikel nochmals eintragen.</td></tr>
<tr><td>Tatsächliches</td><td>Die Bedienung erhält folgende Fehlermeldung:</td></tr>
</table>

Testergebnis	
Test bestanden	Normalablauf: Ja Sonderfall: Ja

Rechnung splitten:

Anwendungs-fall	Möchte eine Gruppe von Kunden ihre Bestellung getrennt bezahlen, so kann der Preis einfach dargestellt werden. Die Bedienung kann für jeden Kunden eingeben welche Artikel bezahlt werden sollen. Anschließend kann sie an dem Bildschirm ablesen, welchen Betrag der jeweilige Kunde zu zahlen hat. Dies kann die Bedienung für jeden Kunden separat machen. (AW 4b)
Verwendete Methode	*doInBackground()* *onPostExecute()* (Klasse: *"UpdateOrder"*) *showToast* (String text)
Normalablauf	Eine Bedienung kommt an einen Tisch, an denen verschiedene Personen bezahlen möchten. Sie geht in die Bestellung und klickt dann auf den Button „BEZAHLEN". Danach wird ihr ein neuer Bildschirm angezeigt. Sie kann nun auswählen, ob die Kunden gemeinsam bezahlen oder ob jeder Kunde für sich bezahlen möchte. Wenn jeder Kunde einzeln zahlen möchte, markiert sie die Checkbox „Separat Bezahlen". Nun können die Kunden die Artikel getrennt zahlen.
Erwartetes Testergebnis	Der Preis wird richtig berechnet und dargestellt. Die Artikel die ausgewählt wurden, sind in der Datenbank als bezahlt hinterlegt. Die Datenbankinhalte vor dem Test sehen wie folgt aus:

orderedItemID	orderID	itemID	itemPaid	itemProduced	comment
200	94	11	1	0	NULL
201	94	11	1	0	NULL
202	94	1	1	0	NULL
203	94	2	1	1	aus der Flasche
204	94	2	1	0	NULL
205	94	17	1	0	NULL
206	94	16	1	0	NULL
207	95	2	0	0	NULL
208	95	2	0	0	NULL
209	95	11	0	0	NULL
210	95	11	0	0	NULL
211	95	16	0	0	NULL
212	95	17	0	0	NULL

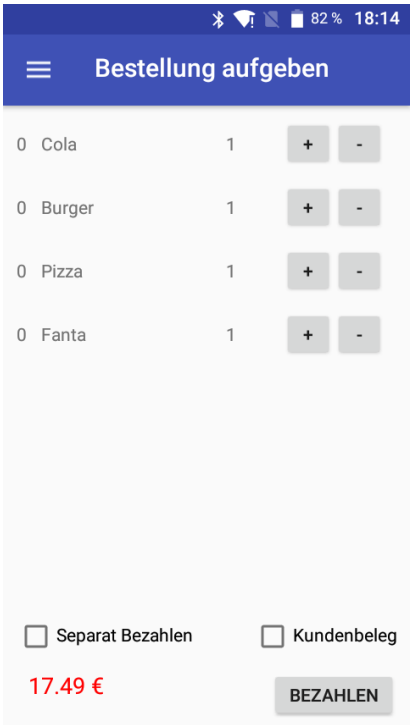
<table>
<tr><td></td><td>Die separat bezahlten Artikel:
 

Eine Cola und ein Burger werden für 7.0 € bezahlt. Der Rest der Rechnung bleibt offen.</td></tr>
<tr><td>Tatsächliches Testergebnis</td><td>Die Datenbank wird aktualisiert. Die Artikel die bezahlt wurden werden auf „1“ gesetzt.
Der Preis wird in der Anwendung richtig berechnet und dargestellt.
Die Datenbankinhalte nach dem Test:
<table>
<tr><th>orderedItemID</th><th>orderID</th><th>itemID</th><th>itemPaid</th><th>itemProduced</th><th>comment</th></tr>
<tr><td>200</td><td>94</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>201</td><td>94</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>202</td><td>94</td><td>1</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>203</td><td>94</td><td>2</td><td>1</td><td>1</td><td>aus der Flasche</td></tr>
<tr><td>204</td><td>94</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>205</td><td>94</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>206</td><td>94</td><td>16</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>207</td><td>95</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>208</td><td>95</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>209</td><td>95</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>210</td><td>95</td><td>11</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>211</td><td>95</td><td>16</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>212</td><td>95</td><td>17</td><td>0</td><td>0</td><td>NULL</td></tr>
</table>
Die Artikel mit der „orderedItemID“ 207 und 209 wurden bezahlt.</td></tr>
<tr><td>Sonderfall</td><td>Die Rechnung wird auf einmal bezahlt. Die Kunden möchten ihre Rechnung nicht separat bezahlen. Der Vorgang verhält sich</td></tr>
</table>

<table>
<tr><td></td><td>gleich wie im Normalfall. Jedoch darf die Checkbox „Separat Bezahlen“ nicht aktiviert werden.</td></tr>
<tr><td>Erwartetes Testergebnis</td><td>Alle Artikel werden in der Datenbank als bezahlt hinterlegt. Der Preis der Bestellung wird der Bedienung richtig dargestellt.

orderedItemID	orderID	itemID	itemPaid	itemProduced	comment
200	94	11	1	0	NULL
201	94	11	1	0	NULL
202	94	1	1	0	NULL
203	94	2	1	1	aus der Flasche
204	94	2	1	0	NULL
205	94	17	1	0	NULL
206	94	16	1	0	NULL
207	95	2	0	0	NULL
208	95	2	0	0	NULL
209	95	11	0	0	NULL
210	95	11	0	0	NULL
211	95	16	0	0	NULL
212	95	17	0	0	NULL

Der Rest der Bestellung aus dem Normalfall wird nun zusammen bezahlt. In der Android-Anwendung sieht das wie folgt aus:

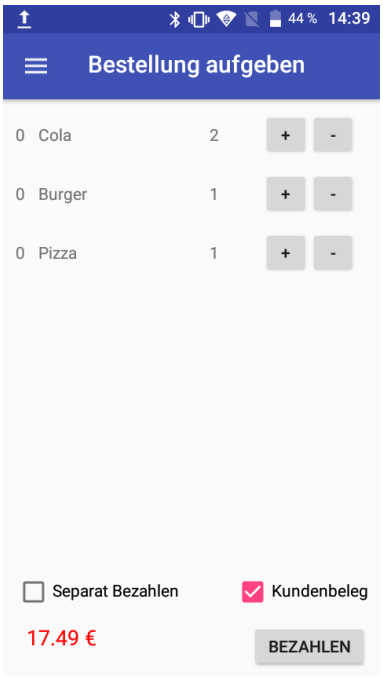

</td></tr>
<tr><td>Tatsächliches Testergebnis</td><td>Die Datenbank wird aktualisiert. Die Artikel die bezahlt wurden werden auf „1“ gesetzt.
Der Preis wird in der Anwendung richtig berechnet und dargestellt.
Die Datenbank wird wie folgt dargestellt:</td></tr>
</table>

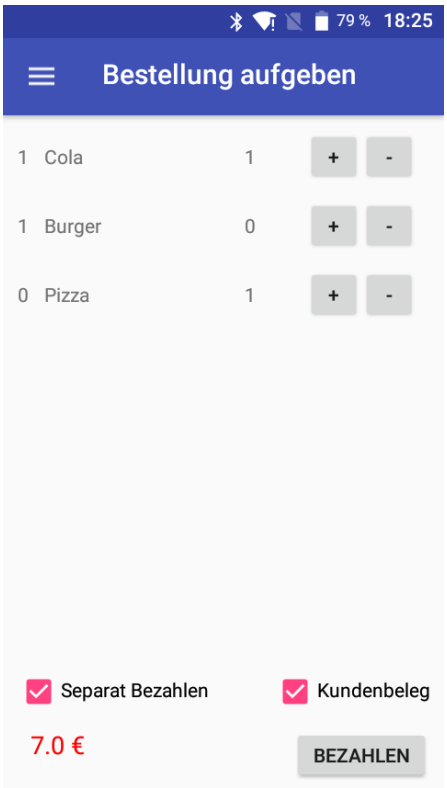
orderedItemID	orderID	itemID	itemPaid	itemProduced	comment
200	94	11	1	0	NULL
201	94	11	1	0	NULL
202	94	1	1	0	NULL
203	94	2	1	1	aus der Flasche
204	94	2	1	0	NULL
205	94	17	1	0	NULL
206	94	16	1	0	NULL
207	95	2	1	0	NULL
208	95	2	1	0	NULL
209	95	11	1	0	NULL
210	95	11	1	0	NULL
211	95	16	1	0	NULL
212	95	17	1	0	NULL

Alle Artikel mit „OrderID“ *95* wurden bezahlt.

Test bestanden	Normalablauf: Ja Sonderfall: Ja

Kundenbeleg anfordern

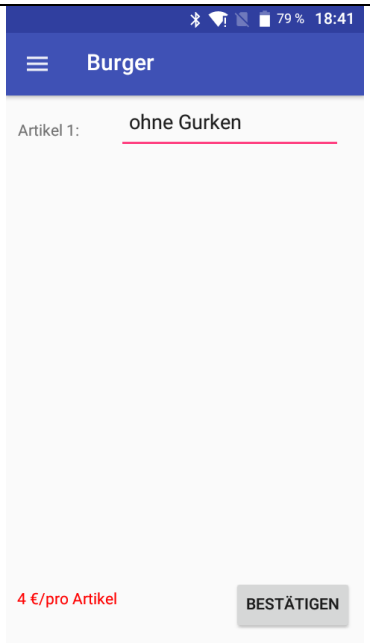
Anwendungs-fall	Falls ein Kunde einen Beleg für den Besuch in der Gastronomie möchte, kann ein Kundenbeleg ausgedruckt werden. (Mit Tisch, Anschrift der Gastronomie, den bestellten Artikeln und dem Preis).
Verwendete Methode	*doInBackground()* *onPostExecute()* (Klasse: *"PrinterSalesCheck"*) *showToast* (String text)
Normalablauf	Eine Bedienung kommt an einen Tisch, an dem Kunden bezahlen möchten. Sie geht in die Bestellung und klickt dann auf den Button „BEZAHLEN". Danach wird ihr ein neuer Bildschirm angezeigt. Sie kann nun auswählen, ob die Kunden einen Beleg möchten oder nicht. Wenn der Kunde einen möchte, muss die Checkbox „Kundenbeleg" angewählt werden. Nach anschließendem Klicken auf den Button „Bezahlen", wird der Beleg ausgedruckt.
Erwartetes Testergebnis	Der Kundenbeleg wird mit Anschrift der Gastronomie, bestellten Artikeln, Preis und Tisch ausgedruckt. Im Test sieht die Bestellung wie folgt aus: 

Tatsächliches Testergebnis	Der Beleg sieht wie folgt aus: ----------Kundenbeleg---------- Restaurante Gaumenfreude Gourmetstraße 11 12345 Leckerschmeckerhausen +49 541 466 655 Ihre Bestellung: Cola 2,50 EUR Cola 2,50 EUR Burger 4,50 EUR Pizza 7,99 EUR _____ Summe 17,49 EUR inkl. MWST 19% 3,48 EUR Sie saßen an Tisch A1. Vielen Dank für Ihren Besuch! 20.03.2018 18:20:44
Sonderfall	Die Kunden möchten getrennt bezahlen und möchten einen Kundenbeleg. Das Vorgehen ist dasselbe wie im Normalfall beschrieben.
Erwartetes Testergebnis	Der gesamte Beleg einer Bestellung wird ausgedruckt. Dieser kann dann für jeden Kunden einzeln ausgedruckt werden. Im Test sieht die Bestellung wie folgt aus: 

Tatsächliches Testergebnis	Der Beleg sieht wie folgt aus: ----------Kundenbeleg---------- Restaurante Gaumenfreude Gourmetstraße 11 12345 Leckerschmeckerhausen +49 541 466 655 Ihre Bestellung: Cola 2,50 EUR Cola 2,50 EUR Burger 4,50 EUR Pizza 7,99 EUR Summe 17,49 EUR inkl. MWST 19% 3,48 EUR Sie saßen an Tisch A1. Vielen Dank für Ihren Besuch! 20.03.2018 18:20:44
Test bestanden	Normalablauf: Ja Sonderfall: Ja

Bestellten Artikel kommentieren:

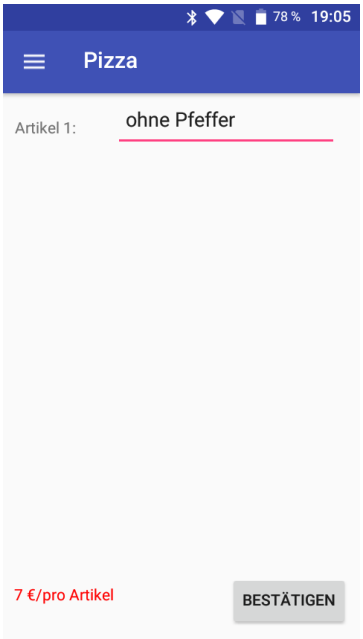
Anwendungsfall	Der Kunde möchte bei einem Artikel eine bestimmte Art der Zubereitung oder hat einen sonstigen Wunsch, den der Koch beachten soll. Dann kann dem bestellten Artikel eine Information mitgegeben werden. Diese Information wird dann ebenfalls in der Datenbank abgelegt.
Verwendete Methode	*doInBackground()* *onPostExecute()* (Klasse: *"UpdateOrder"*) *showToast* (String text)
Normalablauf	Ein Kunde äußert einen Wunsch zu einem Artikel. Dieser Wunsch kann dem Artikel angefügt werden. Dazu muss die Bedienung im „Bestellung aufgeben" Verfahren auf den Artikel lange drücken. Es öffnet sich ein neues Fenster, in dem die Bedienung den geäußerten Wunsch anmerken kann. Anschließend kann die Bestellung normal zu Ende geführt werden.
Erwartetes Testergebnis	Der Kommentar wird gespeichert und in der Datenbank hinterlegt. Der Küchenbeleg beinhaltet den Kommentar. Der Bestellung wird folgender Kommentar angeheftet:

<table>
<tr><td></td><td> 
</td></tr>
<tr><td>Tatsächliches Testergebnis</td><td>Die Datenbank sieht nach der Bestellung wie folgt aus:

<table>
<tr><th>orderedItemID</th><th>orderID</th><th>itemID</th><th>itemPaid</th><th>itemProduced</th><th>comment</th></tr>
<tr><td>208</td><td>95</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>209</td><td>95</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>210</td><td>95</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>211</td><td>95</td><td>16</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>212</td><td>95</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>213</td><td>96</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>214</td><td>96</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>215</td><td>96</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>216</td><td>96</td><td>16</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>217</td><td>98</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>218</td><td>98</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>219</td><td>98</td><td>11</td><td>0</td><td>0</td><td>ohne Gurken</td></tr>
<tr><td>220</td><td>98</td><td>16</td><td>0</td><td>0</td><td>NULL</td></tr>
</table>

Der Küchenbeleg sieht wie folgt aus:

</td></tr>
<tr><td>Sonderfall</td><td>Dem Artikel soll ein Wunsch hinzugefügt werden, nachdem dieser bereits bestellt wurde.</td></tr>
<tr><td>Erwartetes Testergebnis</td><td>In der Datenbank wird der Kommentar nicht mehr gespeichert und kann somit auch nicht berücksichtigt werden.</td></tr>
</table>

<table>
<tr><td></td><td>Der Kundenwunsch für den Artikel:
 
</td></tr>
<tr><td>Tatsächliches Testergebnis</td><td>Die Datenbank sieht wie folgt aus:
<table>
<tr><th>orderedItemID</th><th>orderID</th><th>itemID</th><th>itemPaid</th><th>itemProduced</th><th>comment</th></tr>
<tr><td>208</td><td>95</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>209</td><td>95</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>210</td><td>95</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>211</td><td>95</td><td>16</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>212</td><td>95</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>213</td><td>96</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>214</td><td>96</td><td>2</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>215</td><td>96</td><td>11</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>216</td><td>96</td><td>16</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>217</td><td>98</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>218</td><td>98</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>219</td><td>98</td><td>11</td><td>0</td><td>0</td><td>ohne Gurken</td></tr>
<tr><td>220</td><td>98</td><td>16</td><td>0</td><td>0</td><td>NULL</td></tr>
</table>
Die Datenbank wurde nicht aktualisiert.</td></tr>
<tr><td>Test bestanden</td><td>Normalablauf: Ja
Sonderfall: Ja</td></tr>
</table>

6 Fazit

Zusammenfassend kann gesagt werden, dass das Kassensystem nun in einer Gastronomie eingesetzt werden kann. Mit dem System kann der normale Tagesablauf in einer Gastronomie bewältigt werden.

Nun können auch anspruchsvollere Tätigkeiten von dem Kassensystem übernommen werden. Dazu gehört zum Beispiel die Authentifizierung des geschulten Personals, oder auch das Hinzufügen von Kommentaren zu einer Bestellung. Durch die Optimierungen ist es nun möglich, kundenspezifischer zu reagieren. Dadurch können alltägliche Aufgaben leichter bearbeitet werden.

Das System weist jedoch noch Potential auf, die Anwendungen benutzerfreundlicher zu gestalten. Durch Implementierung verschiedener zusätzlicher Funktionen kann das System den Arbeitsalltag im Gastronomie-Bereich noch vereinfachen.

Hierbei sind verschiedene Funktionen mit unterschiedlichem Nutzen für den Anwender zu nennen. Beispielsweise kann das System dahingehend erweitert werden, dass die gesammelten Daten, unter anderem von den verschiedenen Bestellungen, grafisch (in Form eines Diagramms) dargestellt werden können. Dadurch kann der Nutzer sich zu jedem Zeitpunkt einen Überblick darüber verschaffen, welche Artikel, zu welcher Uhrzeit, gut beziehungsweise weniger gut, verkauft werden.

Dies ist nur ein Beispiel zur Erweiterung des Kassensystems. Es könnten noch Zahlreiche weitere Funktionen implementiert werden.

7 Literaturverzeichnis

1. **Firesmith, Donald.** Using V Models for Testing. [Online] 11. November 2013. [Zitat vom: 6. Mai 2018.] https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html.

2. **Universität Rostock - Institut für Informatik - LS Softwaretechnik.** Gliederungsschema eines Pflichtenheftes (2. Auflage). [Online] [Zitat vom: 6. Mai 2018.] http://swt.informatik.uni-rostock.de/swt_lehre/swt_lehrangebot/swt_vorlesungen/swt_vl_swt/swt_pflichtenheft/schema_2/?L=1%20AND%201%3D1--.

3. **Kulesz, Daniel.** *Softwareengineering - T3 - Spezifikation.* 2017.

4. **Dobraunig, Christoph , Eichlseder, Maria und Mendel, Florian.** Security Evaluation of SHA-224, SHA-512/224, and SHA-512/256. [Online] Februar 2015. [Zitat vom: 25. März 2018.] http://www.cryptrec.go.jp/estimation/techrep_id2401.pdf.

5. **Schneier, Bruce.** Cryptanalysis of MD5 and SHA: Time for a New Standard. [Online] [Zitat vom: 6. Mai 2018.] https://www.schneier.com/essays/archives/2004/08/cryptanalysis_of_md5.html.

6. **Konzeptueller Datenbankentwurf. [Online] 2005. [Zitat vom: 25. März 2018.] http://dbis.informatik.uni-freiburg.de/content/DBBuch/Folien/kapitel04.pdf.**

7. Kulesz, Daniel. *Software Engineering - T6 - Sofware Test.* [PDF] 2017.

8 Anhang

8.1 Installationsanweisung

Im Folgenden wird beschrieben, wie man das Datenbank-System auf einem Computer installiert und die Android Applikation auf einem Android fähigem Smartphone installiert.

8.1.1 Datenbank-System

Die folgenden Schritte müssen durchgeführt werden, um das Datenbank-System auf einem Computer zu installieren und betreiben.

1. Wenn nicht vorhanden, JRE (Java Runtime Environment) installieren.

2. Installation des MySQL Community Servers
   Download des MySQL-installers von:
   https://dev.mysql.com/downloads/windows/installer/5.5.html
   Folgende Komponenten installieren:
   - MySQL-Server
   - MySQL-Workbench
   - MySQL-Notifier

3. Imortieren der Datenbank Strukturen
   - MySQL-Workbench öffnen: Management -> Data Import/Restore
   - "Import from Dump Project Folder":
     Die Datei "Datenbank Import/Dump20171128.sql" auswählen
   - Im Drop-Down-Menü "Dump Structure Only" auswählen
   - Imporieren mit "Start Import"

4. Anlegen eines neuen Users für den Database-Service
   - MySQL-Workbench: Management -> Users and Privileges
   - Den foglenden User anlegen:
     - Login Name: DatabaseService
     - Password: password
     - Im Tab "Administrative Roles": "DBManager" auswählen
   - Anwenden mit "Apply"
5. Installieren des Druckertreibers

- Download des Treibers: https://download.epson-biz.com/modules/pos/index.php?page=single_soft&cid=5131&pcat=3&scat=31
- APD_507_T88V.exe im Ordner "Druckertreiber APD_507_T88V_EWM" starten
- Installationsanweisungen folgen

6. Download des aktuellsten DatabaseSystems
   - https://github.com/Kassensystem/DatabaseSystem/releases/latest
   - Zum Download auf "Source Code (zip)" klicken
   - Entpacken der Dateien

7. Download der aktuellsten ManagerApplication
   - https://github.com/Kassensystem/ManagerApplication/releases/latest
   - Zum Download auf "Source Code (zip)" klicken
   - Entpacken der Dateien

8. Anpassen der Firewall
   - Um dem Server eine Kommunikation im lokalen Netzwerk zu ermöglichen die folgenden Änderungen durchführen:
     - Windows-Firewall öffnen
     - Firewall komplett deaktivieren

9. Starten des Servers
   - In entpackten Dateien des DatbaseSystems:
     - Datei "start.bat" starten
     - Das Kommandozeilenfenster geöffnet lassen
     - Zum Beenden des Servers das Fenster schließen.

10. Starten der ManagerApplication
    - In entpackten Dateien der ManagerApplication:
      - Datei "start.bat" starten
      - Oder “kassensystem_manager.exe” im Pfad: "\out\artifacts\kassensystem_managerApplication\bundles\kassensystem_managerApplication"

8.1.2 Android Applikation

Die folgenden Schritte müssen durchgeführt werden, um die Applikation auf einem Smartphone zu installieren und zu betreiben.

1. Voraussetzung: Betriebssystem: Android, Version > 5

2. Installation von Apps aus unbekannten Quellen zulassen:
   Einstellungen öffnen → Unterpunkt *Sicherheit* auswählen → *Unbekannte Herkunft* muss aktivieren

3. Download der APK auf das Smartphone:
   https://github.com/Nunay/Kassensytem-AndroidApplikation

4. Heruntergeladene Datei öffnen und installieren
   Ordner *Dateien* öffnen → *app-release.apk* öffnen → installieren

5. Applikation starten

9 Testdokumentation

Im Folgenden sind die Tests der öffentlichen Methoden der Software-Module dokumentiert. Es wird jeweils der Anwendungsfall und die getestete Methode beschrieben, außerdem Normalablauf und Sonderfälle, bspw. einem Laufzeitfehler wie Verbindungsprobleme oder falsche Eingaben. Anschließend wird das zu erwartenden und das tatsächliche Testergebnis für den Normalablauf und Sonderfall dokumentiert. In einigen Fällen wurden die Testergebnisse gekürzt.

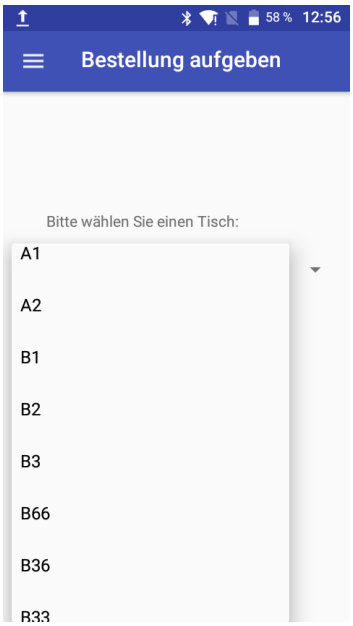
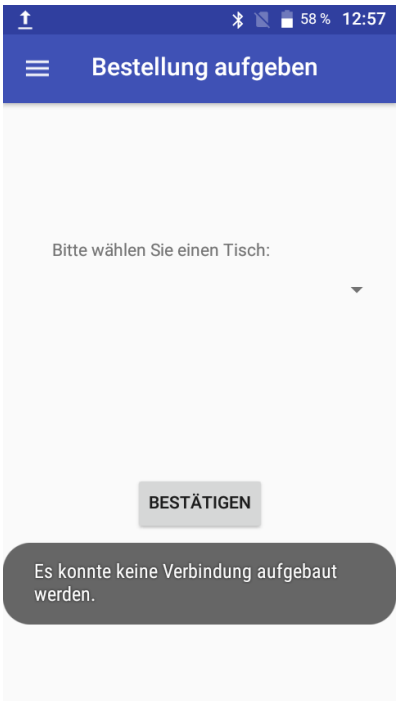
9.1 Android-Anwendung

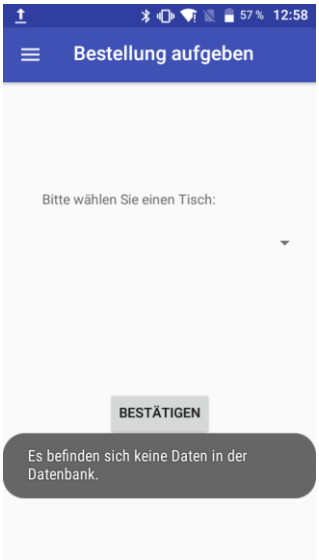
Die folgenden Tests müssen aufgrund der Optimierung die an dem System vorgenommen wurden noch einmal durchlaufen werden.

Die Tests sind dieselben wie in der vorangegangenen Version. Sie wurden jedoch am neuen System durchgeführt.

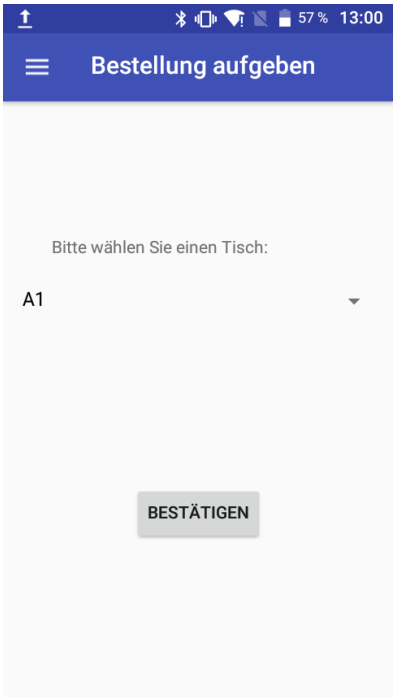
Einsehen der Tische

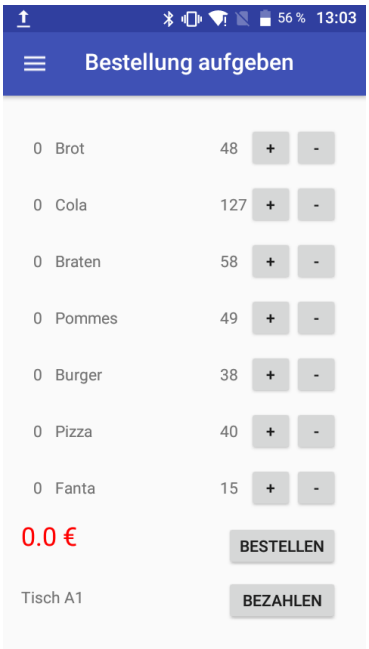
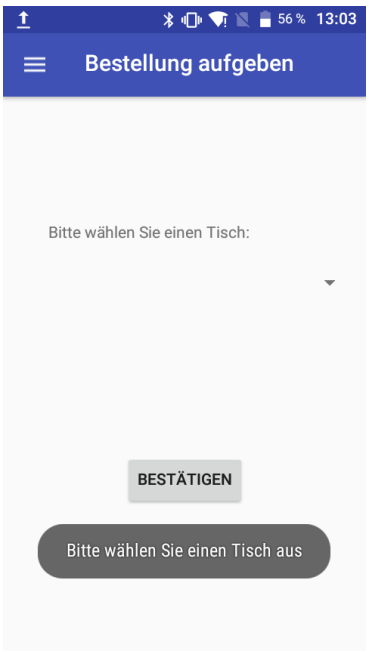
Anwendungsfall	Einsehen aller verfügbaren Tische in der Android-Anwendung (AW 15)
Verwendete Methode	doInBackground() onPostExecute() (Klasse: “GetAllTables”)
Normalablauf	Wird „Bestellung aufgeben“ im Navigation-Drawer ausgewählt, sollen in einem Dropdown-Menü alle verfügbaren Tische der Datenbank dargestellt werden.
Erwartetes Testergebnis	In dem Dropdown-Menü werden die verfügbaren Tische der Datenbank angezeigt. Folgende Tische, die unter dem Register „available“ mit einer 1 markiert sind, werden angezeigt: tableID, name, seats, available 1, A1, 5, 1 2, A2, 4, 1 3, A3, 10, 0 4, B1, 2, 1 5, B2, 2, 1 6, B3, 4, 1 7, B66, 6, 1 8, B36, 8, 1 9, B33, 8, 1

Tatsächliches Testergebnis	Das Dropdow-Menü wurde wie folgt dargestellt: 
Sonderfall 1	Es kann keine Verbindung zur Datenbank aufgebaut werden, beziehungsweise der Server wurde nicht gestartet.
Erwartetes Testergebnis	Das Dropdown-Menü bleibt leer und es wird eine Fehlermeldung angezeigt, dass eine Verbindung zum Server nicht möglich ist.
Tatsächliches Testergebnis	Das Smartphone gibt folgende Fehlermeldung aus: 
Sonderfall 2	Es kann eine Verbindung aufgebaut werden. In der Datenbank befinden sich jedoch keine Tische.

Erwartetes Testergebnis	Das Dropdown-Menü bleibt leer und es wird eine „Fehlermeldung“ angezeigt, dass keine Tische verfügbar sind.
Tatsächliches Testergebnis	Das Smartphone gibt folgende Fehlermeldung aus: 
Test bestanden	Normalablauf: Ja Sonderfall 1: Ja Sonderfall 2: Ja

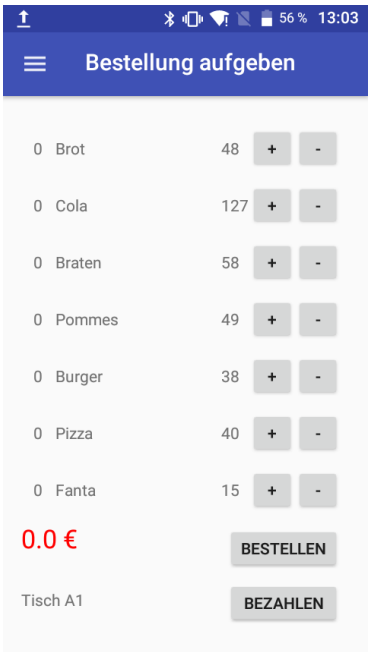
Auswählen der Tische

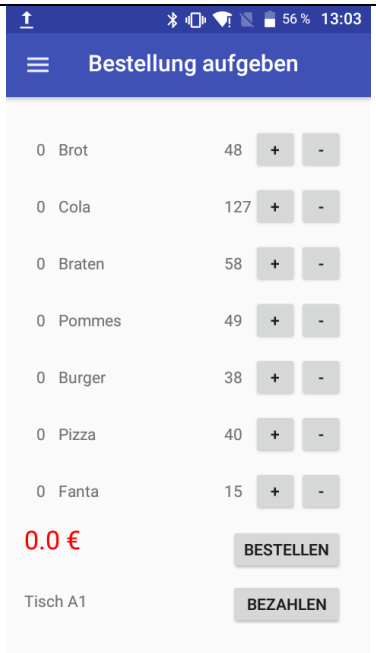
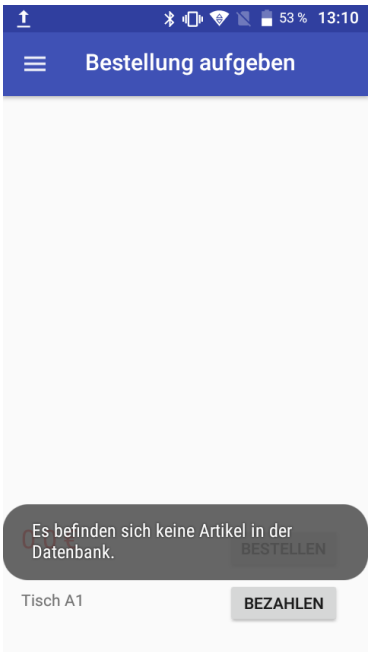
Anwendungs-fall	Auswählen der in dem Dropdown-Menü angezeigten Tische (AW 16)
Verwendete Methode	*onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)* (Klasse: “GetAllTables”)
Normalablauf	Der Anwender wählt aus dem Dropdown-Menü einen angezeigten Tisch aus und bestätigt diesen mit dem Button „BESTÄTIGEN“. 
Erwartetes Testergebnis	Der Tisch wird gespeichert, damit er der Bestellung hinzugefügt werden kann und auf dem Kundenbeleg dargestellt werden kann. Das nächste Fragment wird dargestellt. In diesem Fragment werden alle verfügbaren Artikel angezeigt.

Tatsächliches Testergebnis	Der Tisch wird gespeichert. Der Bildschirm des Smartphones wird folgendermaßen dargestellt: 
Sonderfall	Es wurde kein Tisch ausgewählt und der „BESTÄTIGEN“-Button wird geklickt.
Erwartetes Testergebnis	Es wird eine Fehlermeldung angezeigt, dass ein Tisch ausgewählt werden soll.
Tatsächliches Testergebnis	Das Smartphone gibt folgende Fehlermeldung aus: 

Test bestanden	Normalablauf: Ja Sonderfall: Ja

Artikel Einsehen

Anwendungs-fall	Einsehen der Artikel, die in der Datenbank als verfügbar angelegt sind (AW 17)
Verwendete Methode	*onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)* (Klasse: “ItemSelect”)
Normalablauf	Alle verfügbaren Artikel der Datenbank werden dargestellt.
Erwartetes Testergebnis	Alle Artikel werden dargestellt, inklusive Preis und Menge.
Tatsächliches Testergebnis	Der Bildschirm des Smartphones wird folgendermaßen dargestellt: 
Sonderfall 1	Die Verbindung zum Server wird getrennt.
Erwartetes Testergebnis	Die Artikel werden weiterhin angezeigt.
Tatsächliches Testergebnis	Der Bildschirm des Smartphones wird folgendermaßen dargestellt:

	
Sonderfall 2	Die Datenbank enthält keine verfügbaren Artikel die auf dem Smartphone angezeigt werden.
Erwartetes Testergebnis	Es werden keine Artikel angezeigt. Eine „Fehlermeldung“ gibt an, dass keine Artikel in der Datenbank verfügbar sind.
Tatsächliches Testergebnis	Das Smartphone gibt folgende Fehlermeldung aus: 
Test bestanden	Normalablauf: Ja Sonderfall 1: Ja Sonderfall 2: Ja

Artikel einer Bestellung hinzufügen

Anwendungs-fall	Der Bestellung einen Artikel hinzufügen und löschen (AW 18 und AW 19)
Verwendete Methode	*onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)* (Klasse: “ItemSelect”)
Normalablauf	Allen verfügbaren Artikeln werden ein „Plus“ und ein „Minus“ Button hinzugefügt. Wenn der „Plus“ Button geklickt wird, wird der Bestellung der ausgewählte Artikel einmal hinzugefügt. Wird der „Minus“ Button geklickt, wird der ausgewählte Artikel einmal von der Bestellung entfernt.
Erwartetes Testergebnis	Alle Artikel werden hinzugefügt oder abgezogen.
Tatsächliches Testergebnis	Die ausgewählten Artikel werden der Bestellung hinzugefügt beziehungsweise abgezogen.
Sonderfall 1	Es soll ein Artikel hinzugefügt werden, der die Menge „0“ besitzt. Dieser Artikel befindet sich nicht mehr im Lager.
Erwartetes Testergebnis	Der Artikel lässt sich der Bestellung nicht mehr hinzufügen.
Tatsächliches Testergebnis	Der Artikel lässt sich der Bestellung nicht mehr hinzufügen.
Sonderfall 2	Von der Bestellung soll durch Klicken auf den „Minus“ Button ein Artikel entfernt werden, der in der Bestellung nicht mehr vorhanden ist.
Erwartetes Testergebnis	In der Bestellung wird der Artikel weiterhin mit „0“ angegeben und ist in der Bestellung somit nicht vorhanden.
Tatsächliches Testergebnis	In der Bestellung wird der Artikel weiterhin mit „0“ angegeben und ist in der Bestellung somit nicht vorhanden.
Sonderfall 3	Von der Bestellung soll durch Klicken auf den „Minus“ Button ein Artikel entfernt werden, der im Vorfeld schon bestellt wurde.
Erwartetes Testergebnis	Die Anzahl der Artikel die bereits bestellt wurden, wird nicht unterschritten.

Tatsächliches Testergebnis	Die Anzahl der Artikel die bereits bestellt wurden, wird nicht unterschritten.
Test bestanden	Normalablauf: Ja Sonderfall 1: Ja Sonderfall 2: Ja Sonderfall 3: Ja

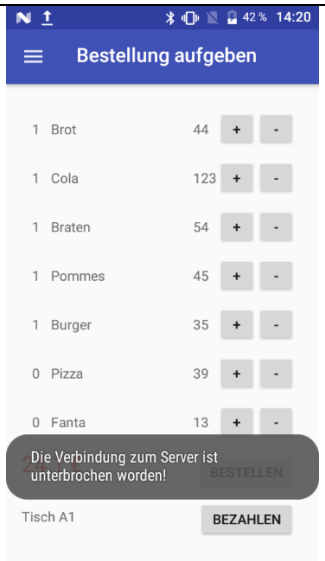
Bestellung abschicken

Anwendungs-fall	Eine von der Bedienung zusammengestellte Bestellung soll an die Küche geschickt werden.
Verwendete Methode	*doInBackground()* (Klasse: “ItemSelect”)
Normalablauf	Die Bedienung stellt für den Kunden eine Bestellung zusammen. Anschließend klickt sie auf den Button „BESTELLEN“ und sendet der Küche die Bestellung.
Erwartetes Testergebnis	In der Datenbank wird eine neue Bestellung erstellt. Diese beinhaltet den ausgewählten Tisch, die Bestellungs-ID, das Datum inklusive Uhrzeit wann die Bestellung erstellt wurde und die Bedienungs-ID. Also welche Bedienung die Bestellung aufgenommen hat. Mit folgenden Daten wird getestet: • Tisch-ID: 1 • Datum und Uhrzeit: 31.03.2018, 18:49 Uhr • Bedienungs-ID: 4 • Bestellungs-ID: 104
Tatsächliches Testergebnis	Die Datenbank sieht wie folgt aus:

orderID	date	tableID	waiterID
92	2018-03-20 12:15:51	1	4
93	2018-03-20 12:19:03	1	4
94	2018-03-20 12:19:26	1	4
95	2018-03-20 18:02:31	1	4
96	2018-03-20 18:20:44	1	4
97	2018-03-20 18:40:49	1	4
98	2018-03-20 18:41:03	1	4
99	2018-03-21 13:03:20	1	4
100	2018-03-21 13:10:20	1	4
101	2018-03-21 13:10:31	1	4
102	2018-03-21 13:12:39	1	4
103	2018-03-21 13:13:49	1	4
104	2018-03-21 13:16:24	1	4

Sonderfall 1	Wenn an diesem Tisch bereits eine Bestellung besteht, bei der zu diesem Zeitpunkt noch nicht alle bestellten Artikel bezahlt wurden. Die Bestellung wird um die neuen Artikel erweitert.
Erwartetes Testergebnis	Es wird in der Datenbank die Bestellung aktualisiert. Mit dem neuen Datum/Uhrzeit und den bereits bestehenden Artikeln und

<table>
<tr><td></td><td>den neu hinzugefügten Artikeln. Der Preis wird ebenfalls aktualisiert. Der Test wird mit der gleichen Bestellung die im Normalfall getestet wurde, gestartet.
Es sollen zwei weitere Artikel bestellt werden.
Die Datenbank sieht vor der Bestellung folgendermaßen aus:
<table>
<tr><th>orderedItemID</th><th>orderID</th><th>itemID</th><th>itemPaid</th><th>itemProduced</th><th>comment</th></tr>
<tr><td>223</td><td>102</td><td>2</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>224</td><td>102</td><td>3</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>225</td><td>102</td><td>4</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>226</td><td>103</td><td>1</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>227</td><td>103</td><td>2</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>228</td><td>103</td><td>3</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>229</td><td>103</td><td>4</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>230</td><td>103</td><td>16</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>231</td><td>103</td><td>11</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>232</td><td>104</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>233</td><td>104</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>234</td><td>105</td><td>1</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>235</td><td>105</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
</table></td></tr>
<tr><td>Tatsächliches Testergebnis</td><td>Die Datenbank sieht nach dem Test wie folgt aus:
<table>
<tr><th>orderedItemID</th><th>orderID</th><th>itemID</th><th>itemPaid</th><th>itemProduced</th><th>comment</th></tr>
<tr><td>225</td><td>102</td><td>4</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>226</td><td>103</td><td>1</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>227</td><td>103</td><td>2</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>228</td><td>103</td><td>3</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>229</td><td>103</td><td>4</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>230</td><td>103</td><td>16</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>231</td><td>103</td><td>11</td><td>1</td><td>1</td><td>NULL</td></tr>
<tr><td>232</td><td>104</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>233</td><td>104</td><td>17</td><td>1</td><td>0</td><td>NULL</td></tr>
<tr><td>234</td><td>105</td><td>1</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>235</td><td>105</td><td>2</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>236</td><td>105</td><td>3</td><td>0</td><td>0</td><td>NULL</td></tr>
<tr><td>237</td><td>105</td><td>4</td><td>0</td><td>0</td><td>NULL</td></tr>
</table></td></tr>
<tr><td>Sonderfall 2</td><td>Die Verbindung zum Server wird während eine Bestellung erstellt wird unterbrochen.</td></tr>
<tr><td>Erwartetes Testergebnis</td><td>Die Bestellung wird nicht gespeichert und eine Fehlermeldung wird angezeigt.</td></tr>
<tr><td>Tatsächliches Testergebnis</td><td>Das Smartphone gibt folgende Fehlermeldung aus:</td></tr>
</table>

	 Nachdem die Verbindung wiederhergestellt wurde, kann die Bestellung abgeschickt werden.
Test bestanden	Normalfall: Ja Sonderfall 1: Ja Sonderfall 2: Ja

Die Tests zu „Bestellung abschicken“ verliefen genauso, wenn in der Artikelübersicht auf den „BEZAHLEN“ Button und nicht auf den „BESTELLEN“ Button geklickt wurde. Der Unterschied besteht darin, dass nach klicken auf den „BEZAHLEN“ Button ein anderer Bildschirm dargestellt wird. (Bildschirm: Siehe „Rechnung splitten“).