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

Studienarbeit

3. Studienjahr

Modul T3100

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

# 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, 09.01.2018 |  |  |  |
| Ort, Datum |  | Unterschrift |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Stuttgart, 09.01.2018 |  |  |  |
| Ort, Datum |  | Unterschrift |  |

Inhaltsverzeichnis

[Ehrenverantwortliche Erklärung II](#_Toc503010055)

[Abbildungsverzeichnis V](#_Toc503010056)

[1 Anforderungen 1](#_Toc503010057)

[1.1 Aufgabenstellung 1](#_Toc503010058)

[2 Pflichtenheft 2](#_Toc503010059)

[2.1 Zielbestimmung 2](#_Toc503010060)

[2.1.1 Musskriterien 2](#_Toc503010061)

[2.1.2 Wunschkriterien 2](#_Toc503010062)

[2.2 Produkteinsatz 3](#_Toc503010063)

[2.2.1 Anwendungsbereiche 3](#_Toc503010064)

[2.2.2 Zielgruppen 3](#_Toc503010065)

[2.2.3 Betriebsbedingungen 3](#_Toc503010066)

[2.3 Produktübersicht 4](#_Toc503010067)

[2.4 Produktfunktionen 5](#_Toc503010068)

[2.4.1 Anwendungsfälle des Kassensystem-Managers 5](#_Toc503010069)

[2.4.2 Anwendungsfälle der Android-Anwendung 14](#_Toc503010070)

[2.5 Produktdaten 19](#_Toc503010071)

[2.6 Produktleistungen 20](#_Toc503010072)

[2.7 Benutzungsoberfläche 21](#_Toc503010073)

[2.7.1 Android-Application 21](#_Toc503010074)

[2.7.2 Lokale Benutzeranwendung 22](#_Toc503010075)

[2.8 Nichtfunktionale Anforderungen 23](#_Toc503010076)

[2.9 Technische Produktumgebung 23](#_Toc503010077)

[2.9.1 Software 23](#_Toc503010078)

[2.9.2 Hardware 23](#_Toc503010079)

[3 Entwurf 24](#_Toc503010080)

[3.1 Klassendesign des Datenbank-Systems 24](#_Toc503010081)

[3.2 Klassendesign der Android-Anwendung 26](#_Toc503010082)

[4 Implementierung 28](#_Toc503010083)

[4.1 Implementierung des Datenbank-Systems 28](#_Toc503010084)

[4.1.1 Klasse DatabaseService\_Interface 28](#_Toc503010085)

[4.1.2 Klasse RestApiController 34](#_Toc503010086)

[4.2 Implementierung der Android-Anwendung 37](#_Toc503010087)

[4.2.1 Klasse TableSelect 37](#_Toc503010088)

[4.2.2 Klasse MainActivity 38](#_Toc503010089)

[5 Test 42](#_Toc503010090)

[5.1 Test des Datenbank-Systems 42](#_Toc503010091)

[5.1.1 Benutzeranwendung Kassensystem-Manager 42](#_Toc503010092)

[5.2 Test der Android-Anwendung 60](#_Toc503010093)

[6 Fazit 75](#_Toc503010094)

[7 Anhang 76](#_Toc503010095)

[7.1 Installationsanweisung 76](#_Toc503010096)

[7.1.1 Datenbank-System 76](#_Toc503010097)

[7.1.2 Android Applikation 78](#_Toc503010098)

[8 Testdokumentation 79](#_Toc503010099)

[8.1 Datenbank-Modul 80](#_Toc503010100)

[8.1.1 DatabaseService-Klasse 80](#_Toc503010101)

[8.1.2 Server-Modul 112](#_Toc503010102)

[8.1.3 Benutzeranwendung 116](#_Toc503010103)

# Abbildungsverzeichnis

[Abbildung 1: Systemübersicht des Projekts 4](#_Toc503009826)

[Abbildung 2: Anwendungsfalldiagramm des Kassensystem-Managers 6](#_Toc503009827)

[Abbildung 3: Anwendungsfalldiagramm der Android-Anwendung 14](#_Toc503009828)

[Abbildung 4: Strukturierung der Datenbank und Abhängigkeiten der Tabellen 19](#_Toc503009829)

[Abbildung 5: Entwurf der Benutzeroberfläche der Android-Applikation 21](#_Toc503009830)

[Abbildung 6: Entwurf der Benutzeroberfläche des Kassensystem-Managers 22](#_Toc503009831)

[Abbildung 7: Software-Package "database" 24](#_Toc503009832)

[Abbildung 8: Das Software-Package "RestApiController" 25](#_Toc503009833)

[Abbildung 9: Software-Package "Kassensystemapplikation" 26](#_Toc503009834)

# Anforderungen

## 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.

Mit der Hilfe einer Manager-Applikation, die eine Benutzeroberfläche zur Datenbankverwaltung darstellt, soll auf dem Computer der Lagerbestand verwaltet und die Wareneingänge, beziehungsweise Warenausgänge gesteuert werden. Ebenfalls sollen in der Manager-Applikation Lagerartikel bearbeitet werden können.

Auf den Handheld-Geräten soll es möglich sein, die derzeit im Lager verfügbaren Artikel, inklusive deren Bestand, einzusehen. Ebenfalls soll es an den Handheld-Geräten möglich sein, kundenbezogen eine Bestellung zusammenzufassen. Nachdem der Bestellvorgang beendet wurde, soll mithilfe eines Bon-Druckers ein Bestellbon/Lieferschein ausgedruckt werden können.

Das gesamte System soll für den Gastronomiebereich ausgelegt werden.

# Pflichtenheft

## Zielbestimmung

### Musskriterien

**Datenbank**

* Speicherung von Daten in vorgegebenen Strukturen
  + Artikel
  + Lagerbestand
  + Bestellungen
  + Tische
* Daten auf Anfrage ausgeben und bearbeiten

**Datenbank-Service**

* Bereitstellen einer Schnittstelle für Zugriffe auf die Datenbank
  + Verändern und Ausgeben der Tabelleninhalte
* Bestellung über einen Bon-Drucker ausdrucken

**Lokale Benutzeranwendung**

* Einsehen der Bestellungen
* Einsehen und Bearbeiten der Artikel, des Lagerbestandes und der Tische

**Server**

* Bedienen von Anfragen von beliebig vielen Clients
* Senden von Statusmeldungen und Datenbankinhalten

**Client**

* Tische, Lagerbestand und Artikel abrufen
* Bestellungen erstellen und an den Server senden
  + Empfangen von Statusmeldungen des Servers nach Absenden einer Bestellung
  + Nicht bezahlte Bestellung eines Tisches abrufen

### Wunschkriterien

* Ergänzung der Datenbank
  + Mitarbeiter
  + Anmeldedaten
* Ergänzung des Clients
  + Login-Dialog für Mitarbeiter
* Ergänzungen des Servers
  + Zugriffsberechtigungen beim Login überprüfen

## Produkteinsatz

### Anwendungsbereiche

* Bestellungsaufnahme und Verarbeitung in der Gastronomie
  + Lagerverwaltung

### Zielgruppen

* Kellner: Bestellungen am Smartphone eingeben
* Köche: Empfangen der Bestellungen am Bon-Drucker
* Gastronom(Chef): Verwaltung des Lagerbestandes und Artikel

### Betriebsbedingungen

* WLAN-Netzwerk, keine Internetanbindung notwendig
* Tägliche Betriebszeit: Dauerbetrieb

## Produktübersicht



Abbildung 1: Systemübersicht des Projekts

## Produktfunktionen

### Anwendungsfälle des Kassensystem-Managers

Im Folgenden werden die möglichen Anwendungsfälle der Benutzeranwendung Kassensystem-Manager dargestellt und spezifiziert.

Zunächst werden im Anwendungsfalldiagramm alle Funktionen die der Benutzer aktiv anwendet, grafisch dargestellt.   
Anschließend werden diese Funktionen in einer tabellarischen Auflistung genauer beschrieben.

#### Anwendungsfalldiagramm



Abbildung 2: Anwendungsfalldiagramm des Kassensystem-Managers

#### Spezifizierung der Anwendungsfälle

|  |  |
| --- | --- |
| **Bezeichnung** | 1 – Bestellungen einsehen |
| Ziel | Der Anwender kann in einer tabellarischen Übersicht alle Bestellungen mit ihren Informationen wie bestellte Artikel, Preis, Datum und Tisch einsehen, die noch nicht bezahlt worden sind. |
| Vorbedingung | Es besteht der Bedarf, alle offenen Bestellungen einzusehen. Das ist beispielsweise der Fall, wenn ein Beleg für die Küche abhandengekommen ist und die Informationen über diese Bestellung benötigt werden. |
| Nachbedingung | Es wurden alle Informationen über die Bestellungen richtig in der Übersicht dargestellt. Es wurden nur offene Bestellungen angezeigt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt und keine Informationen. |
| Akteure | Gastronom, Koch |

|  |  |
| --- | --- |
| **Bezeichnung** | 2 – Bestellung löschen |
| Ziel | Der Anwender kann über einen Rechtsklick auf eine Bestellung einen Menüeintrag zum Löschen einer Bestellung auswählen. Anschließend wird diese Bestellung endgültig aus der Datenbank gelöscht. |
| Vorbedingung | Es besteht der Bedarf eine Bestellung zu löschen. Das ist bspw. der Fall, wenn eine Bestellung fälschlicherweise aufgegeben wurde und storniert werden muss. |
| Nachbedingung | Die Bestellung wird nicht mehr in der Benutzeroberfläche angezeigt und wurde aus der Datenbank gelöscht. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom, Koch, Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 3 – Artikel einsehen |
| Ziel | Der Anwender kann in einer tabellarischen Übersicht alle verfügbaren Artikel mit ihren Informationen wie Bezeichnung, Lagerbestand und Preis einsehen. |
| Vorbedingung | Es besteht der Bedarf, alle Artikel einzusehen, bspw. wenn das Sortiment neu strukturiert werden soll oder wenn neue Wareneingänge hinzugefügt werden sollen. |
| Nachbedingung | Es werden alle Artikel mit ihren Informationen korrekt angezeigt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt und keine Informationen. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 4 – Artikel entfernen |
| Ziel | Ein Artikel soll in der Datenbank als nicht verfügbar markiert und anschließend nicht mehr in der tabellarischen Übersicht angezeigt werden. |
| Vorbedingung | Es besteht der Bedarf einen Artikel zu entfernen, bspw. wenn dieser aus dem Sortiment genommen werden soll oder dieser fälschlicherweise angelegt wurde. |
| Nachbedingung | Der Artikel wurde in der Datenbank als nicht verfügbar markiert und wird nicht mehr in der tabellarischen Übersicht dargestellt. In bisherigen Bestellungen wird der Artikel aber trotzdem weiterhin angezeigt. Auch die Wareneingänge dieses Artikels werden nicht mehr angezeigt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 5 – Artikel hinzufügen |
| Ziel | Es wird ein neuer Artikel mit all seinen Informationen wie Bezeichnung und Verkaufspreis neu erstellt. Außerdem wird gleichzeitig ein neuer Wareneingang für diesen Artikel hinzugefügt. |
| Vorbedingung | Es besteht der Bedarf einen neuen Artikel anzulegen. Dies ist bspw. der Fall, wenn ein neuer Artikel das erste Mal geliefert wird. Dann wird gleichzeitig ein Wareneingang mit der ent-sprechend gelieferten Anzahl angelegt. |
| Nachbedingung | Der Artikel wurde neu in der Datenbank angelegt und wird in der tabellarischen Übersicht angezeigt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 6 – Artikel bearbeiten |
| Ziel | Die Informationen eines Artikels sollen bearbeitet werden. Der bisherigere Artikel wird als nicht verfügbar markiert und der bearbeitete Artikel wird der Datenbank hinzugefügt. |
| Vorbedingung | Es besteht der Bedarf, die Informationen eines Artikels zu bearbeiten. Das ist bspw. der Fall, wenn sich der Preis oder die Bezeichnung eines Artikels ändert. Es muss der ent- sprechende Artikel angeklickt werden und anschließend im entsprechenden Formularfeld die neuen Informationen hinzugefügt werden. |
| Nachbedingung | Der Artikel wurde mit den bearbeiteten Informationen in der Datenbank erstellt und der bisherige Artikel als nicht verfügbar markiert. In der Übersicht wird der Artikel mit einer neuen ID dargestellt, der bisherige Artikel mit den alten Informationen allerdings nicht mehr. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 7 – Tische einsehen |
| Ziel | Es sollen alle verfügbaren Tische in einer tabellarischen Übersicht dargestellt werden. |
| Vorbedingung | Es besteht der Bedarf, die Tische einzusehen. Das ist bspw. der Fall, wenn der Verkaufsraum neu strukturiert werden soll. |
| Nachbedingung | Die Tische wurden korrekt in einer tabellarischen Übersicht angezeigt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 8 – Tisch entfernen |
| Ziel | Einer der Tische soll entfernt werden. |
| Vorbedingung | Es besteht der Bedarf, einen Tisch zu entfernen. Dies ist bspw. der Fall, wenn im Verkaufsraum ein Tisch abgebaut oder umgebaut werden soll. |
| Nachbedingung | Der Tisch wurde in der Datenbank als nicht verfügbar markiert und wird nicht mehr in der tabellarischen Übersicht angezeigt. In vergangenen Bestellungen soll der Tisch weiterhin abrufbar sein. Es muss über einen Rechtsklick auf den entsprechenden Tisch der Menüeintrag zum Löschen ausgewählt werden. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 9 – Tisch hinzufügen |
| Ziel | Es soll ein neuer Tisch zu der Datenbank hinzugefügt werden. |
| Vorbedingung | Es besteht der Bedarf, einen neuen Tisch anzulegen. Das ist bspw. der Fall, wenn ein neuer Tisch im Verkaufsraum aufgestellt werden soll. |
| Nachbedingung | Der Tisch wurde erfolgreich der Datenbank hinzugefügt und wird in der tabellarischen Übersicht angezeigt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 10 – Tisch bearbeiten |
| Ziel | Die Informationen eines Tisches sollen bearbeitet werden. Der Tisch mit den alten Informationen wird als nicht verfügbar markiert und ist weiterhin in vergangenen Bestellungen abrufbar. |
| Vorbedingung | Es besteht der Bedarf, einen Tisch zu bearbeiten. Dies ist bspw. der Fall, wenn ein Tippfehler passiert ist oder die Namensvergebung der Tische in der Gastronomie geändert werden soll. Es muss der entsprechende Tisch angeklickt werden und anschließend im entsprechenden Formularfeld die neuen Informationen hinzugefügt werden. |
| Nachbedingung | Der Tisch wird mit seinen neuen Informationen in der tabellarischen Übersicht dargestellt und in der Datenbank neu hinzugefügt. Der alte Artikel wurde als nicht verfügbar markiert. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 11 – Rechnung für Kunden ausdrucken |
| Ziel | Der Anwender möchte eine schon existierende Bestellung ausdrucken, bspw. weil der Beleg verloren gegangen oder verschmutzt worden ist. |
| Vorbedingung | Es besteht der Bedarf eine Bestellung ein weiteres Mal auszudrucken. Über einen Rechtsklick auf die Bestellung kann ein Menüeintrag ausgewählt werden, über den dann die Bestellung ausgedruckt wird. |
| Nachbedingung | Es wurde über den Bon-Drucker erfolgreich und korrekt eine Bestellung in Form eines Kundenbeleges ausgedruckt. |
| Nachbedingung im Sonderfall | Der Bon wurde nicht ausgedruckt, weil der Drucker nicht installiert oder ausgeschaltet ist. |
| Akteure | Bedienung, Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 12 – Wareneingänge einsehen |
| Ziel | Es sollen alle Wareneingänge von verfügbaren Artikeln in einer tabellarischen Übersicht dargestellt werden. |
| Vorbedingung | Es besteht der Bedarf, die Wareneingänge einzusehen. Dies ist bspw. der Fall, wenn eine bisherige Eingabe kontrolliert werden soll oder eine falsche Eingabe gelöscht werden soll. |
| Nachbedingung | Es werden alle Wareneingänge in einer tabellarischen Übersicht dargestellt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 13 – Wareneingang hinzufügen |
| Ziel | Es soll ein neuer Wareneingang für einen Artikel hinzugefügt werden. |
| Vorbedingung | Es besteht der Bedarf, einen neuen Wareneingang hinzuzufügen. Das ist bspw. der Fall, wenn vom Lieferanten eine neue Lieferung erhalten wurde und diese in das Warensystem eingepflegt werden soll.  Es muss in der Artikelübersicht ein Artikel ausgewählt und dann im entsprechenden Formularfeld die Anzahl eingegeben werden. |
| Nachbedingung | Der neue Wareneingang wurde in der Wareneingangsübersicht dargestellt und der Datenbank hinzugefügt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

|  |  |
| --- | --- |
| **Bezeichnung** | 14 – Wareneingang löschen |
| Ziel | Ein Wareneingang soll aus der Datenbank und der tabellarischen Übersicht gelöscht werden. |
| Vorbedingung | Es besteht der Bedarf, einen Wareneingang zu löschen. Das ist bspw. der Fall, wenn ein Wareneingang fälschlicherweise angelegt wurde oder eine Lieferung nach dem Hinzufügen in das Warensystem beschädigt wurde. |
| Nachbedingung | Der Wareneingang wurde endgültig aus der Datenbank und der tabellarischen Übersicht entfernt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Gastronom |

### Anwendungsfälle der Android-Anwendung

Im Folgenden werden die möglichen Anwendungsfälle der Android-Anwendung dargestellt und spezifiziert.

#### Anwendungsfalldiagramm



Abbildung 3: Anwendungsfalldiagramm der Android-Anwendung

#### Spezifizierung der Anwendungsfälle

|  |  |
| --- | --- |
| **Bezeichnung** | 15 - Tische einsehen |
| Ziel | Der Anwender kann in einem Dropdown-Menü alle verfügbaren Tische einsehen. |
| Vorbedingung | Es besteht der Bedarf, eine Bestellung aufzunehmen. Dieser entsteht immer, wenn ein Kunde eine Bestellung aufgeben möchte. |
| Nachbedingung | Es wurden alle Informationen richtig in dem Dropdown-Menü dargestellt. Es wurden nur verfügbare Tische angezeigt. |
| Nachbedingung im Sonderfall | Es wird eine Fehlermeldung angezeigt. |
| Akteure | Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 16 - Tisch auswählen |
| Ziel | Der Anwender kann aus dem Dropdown Menü den Tisch auswählen, an dem der Kunde bestellen möchte. |
| Vorbedingung | Es besteht der Bedarf, eine Bestellung von einem Kunden aufzunehmen. Ebenfalls befindet sich der Kunde an einem Tisch. |
| Nachbedingung | Der Tisch an dem sich der Kunde befindet wird ausgewählt und der Bestellung hinzugefügt. |
| Nachbedingung im Sonderfall | Der Tisch wird nicht ausgewählt und eine Fehlermeldung wird angezeigt. |
| Akteure | Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 17 - Artikel einsehen |
| Ziel | Der Anwender kann in einer tabellarischen Übersicht alle verfügbaren Artikel mit ihren Informationen Preis und Lagerbestand einsehen. |
| Vorbedingung | Es besteht der Bedarf, eine Bestellung von einem Kunden aufzunehmen. Der Tisch an dem sich der jeweilige Kunde befindet wurde bereits ausgewählt. |
| Nachbedingung | Es wurden alle Informationen richtig in einer tabellarischen Übersicht dargestellt. Es wurden nur verfügbare Artikel angezeigt. |
| Nachbedingung im Sonderfall | Die Artikel werden nicht angezeigt und eine Fehlermeldung wird dargestellt. |
| Akteure | Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 18 - Artikel hinzufügen |
| Ziel | Der Anwender kann einer Bestellung die ein Kunde aufgibt, Artikel hinzufügen. |
| Vorbedingung | Es besteht der Bedarf, eine Bestellung von einem Kunden aufzunehmen. Der Tisch an dem sich der Kunde befindet wurde bereits ausgewählt.  Der Kunde möchte einen oder mehrere Artikel, die in der Speisekarte vorhanden sind bestellen. |
| Nachbedingung | Es wurden alle ausgewählten Artikel der Bestellung hinzugefügt. Inklusive Preis und Anzahl der Artikel die bestellt wurden. |
| Nachbedingung im Sonderfall | Die Artikel werden der Bestellung nicht hinzugefügt und eine Fehlermeldung erscheint. |
| Akteure | Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 19 - Artikel entfernen |
| Ziel | Der Anwender kann bei einer Bestellung die ein Kunde aufgibt, Artikel wieder entfernen. |
| Vorbedingung | Der Kunde entscheidet sich während der Bestellung um und möchte einen anderen Artikel bestellen. Ebenfalls ist es möglich, dass die Bedienung zuerst den falschen Artikel eingibt, ihren Fehler jedoch bemerkt. |
| Nachbedingung | Der zu löschende Artikel wurde erfolgreich aus der vom Kunden aufgegebenen Bestellung entfernt. |
| Nachbedingung im Sonderfall | Die Artikel werden nicht aus der vom Kunden aufgegebenen Bestellung entfernt und eine Fehlermeldung wird angezeigt. |
| Akteure | Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 20 - Aktuellen Gesamtpreis ablesen |
| Ziel | Der Anwender kann bei einer Bestellung die ein Kunde aufgibt den Gesamtpreis ablesen. Dieser setzt sich aus den einzelnen Artikeln, die der Bestellung hinzugefügt wurden, zusammen. |
| Vorbedingung | Es besteht der Bedarf, dass ein Kunde den Gesamtpreis wissen möchte. Dies tritt zum Beispiel ein, wenn der Kunde bezahlen möchte.  Ebenfalls kann die Bedienung dem Kunden während einer Bestellung den bisherigen Preis der Bestellung übermitteln. |
| Nachbedingung | Der Gesamtpreis wurde richtig ermittelt und für den Anwender dargestellt. |
| Nachbedingung im Sonderfall | Eine Fehlermeldung wird dargestellt. |
| Akteure | Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 21 - Bestellung abschicken |
| Ziel | Der Anwender kann die Bestellung die ein Kunde aufgibt an die Küche/Koch weiterleiten. |
| Vorbedingung | Es besteht der Bedarf, dass ein Kunde eine Bestellung aufgeben möchte.  Hat der Kunde seine Bestellung bei einer Bedienung aufgegeben, soll diese Bestellung in der Küche bearbeitet werden, damit sie dem Kunden serviert werden kann. |
| Nachbedingung | Die Bestellung des Kunden wurde erfolgreich an die Küche übermittelt. |
| Nachbedingung im Sonderfall | Die Bestellung wird nicht übermittelt und eine Fehlermeldung wird dargestellt. |
| Akteure | Bedienung |

|  |  |
| --- | --- |
| **Bezeichnung** | 22 - Rechnung für den Kunden anfordern |
| Ziel | Der Kunde bekommt für seine Bestellung einen Beleg ausgedruckt. |
| Vorbedingung | Es besteht der Bedarf, dass ein Kunde seine Bestellung bezahlen möchte/muss. |
| Nachbedingung | Die Artikel die der Kunde bestellt hat, sowie der Gesamtpreis werden auf einen Beleg ausgedruckt. |
| Nachbedingung im Sonderfall | Der Beleg wird nicht ausgedruckt und eine Fehlermeldung wird dargestellt |
| Akteure | Bedienung |

## Produktdaten

Die folgenden Daten müssen von der Datenbank gespeichert und verwaltet werden:



Abbildung 4: Strukturierung der Datenbank und Abhängigkeiten der Tabellen

Je nach Implementierung weiterer Funktionen wird dieses Datenmodell angepasst und erweitert. Diese werden gegebenenfalls separat dokumentiert.

## Produktleistungen

Das Abrufen von Datenbankinhalten am Mobiltelefon sollte mit einer geringen Verzögerung ablaufen. Als maximale Dauer einer Aktualisierung mit großem Umfang werden 5 Sekunden angesetzt.

Das Abschicken einer Bestellung vom Mobiltelefon an die Datenbank bis zur nachfolgenden Bestätigung des Empfangs sollte ebenfalls 5 Sekunden nicht überschreiten.

An der lokalen Benutzeranwendung, die auf dem Datenbank-PC läuft, soll die Dauer der Datenaktualisierung nicht über 3 Sekunden liegen. Die Dauer für Änderungen an Daten bei üblichem Umfang soll nicht länger als 2 Sekunden betragen. Bei Änderung großer Datenbereiche sind 4 Sekunden das Maximum.

## Benutzungsoberfläche

### Android-Application



Abbildung 5: Entwurf der Benutzeroberfläche der Android-Applikation

### Lokale Benutzeranwendung



Abbildung 6: Entwurf der Benutzeroberfläche des Kassensystem-Managers

## Nichtfunktionale Anforderungen

Ordnungsmäßigkeit der Buchführung

* Evtl. Ergänzung einer Erstellung einer PDF des Belegs
* Andere Möglichkeit: doppelt ausdrucken zur Archivierung

Sicherheitsanforderungen, z. B. Passwortschutz, Mitlaufen von Protokollen, sichere Übertragung

* Erweiterung mit Login-Daten: Mitarbeiter können sich am Mobiltelefon einloggen
  + Verwendung von Hashcodes für Passwörter
* Evtl. Verwendung von Https statt Http

## Technische Produktumgebung

### Software

Datenbank-Rechner

* Betriebssystem: Windows 10
* MySQL-Server, MySQL-Workbench zur Verwaltung
* Java Runtime Environment (JRE)

Smartphone

* Betriebssystem: Android, Version > 5

### Hardware

Datenbank-Rechner

* Ausreichend Leistung (CPU, RAM)
* Bon-Drucker
* Schnittstelle zum Bon-Drucker (USB, …)
* Bildschirm, Tastatur, Maus, optional Touchscreen
* Netzwerkkarte

Smartphone

* Ausreichend Speicherplatz

# Entwurf

## Klassendesign des Datenbank-Systems

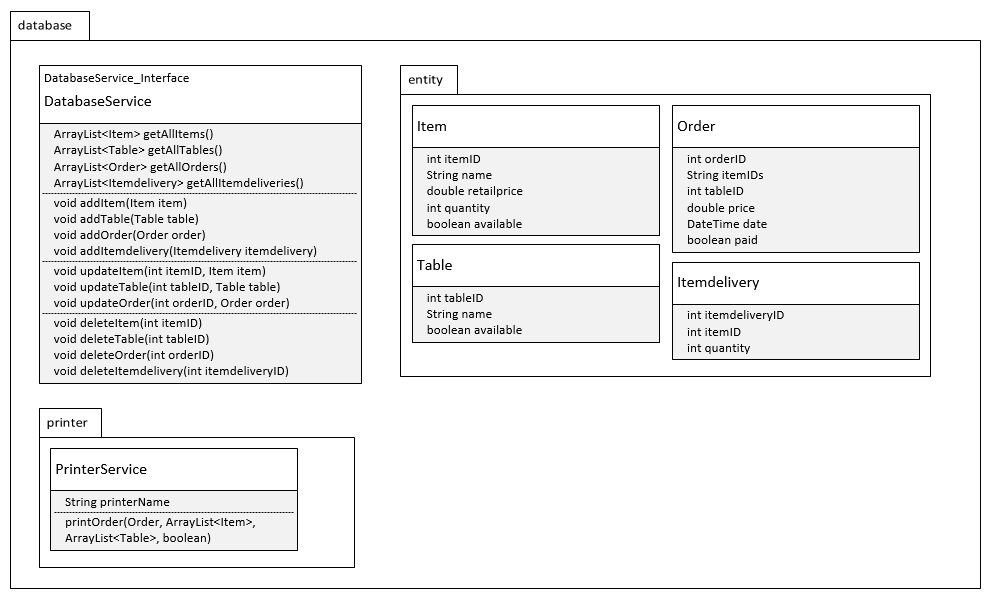


Abbildung 7: Software-Package "database"

Das Datenbank-System ist in zwei Hauptkomponenten aufgeteilt: Der verwaltende Hintergrundprozess und die Benutzeranwendung mit grafischer Oberfläche. Zum Hintergrundprozess gehören mehrere Komponenten: Das Package „database“ (siehe Abbildung 7) und das Package „RestApi“ (siehe Abbildung 8).

Der Database-Service im „database“-Package wird von allen anderen Komponenten verwendet um auf Daten der Datenbank zuzugreifen oder diese zu ändern. Das darin befindliche Package „entity“ beschreibt die Tabellen der Datenbank und dient dazu, neue Datensätze zu erzeugen oder Daten der Datenbank in diesen zu speichern. Das letzte Package „printer“ stellt eine Schnittstelle zur Verfügung, worüber eine Bestellung ausgedruckt werden kann.

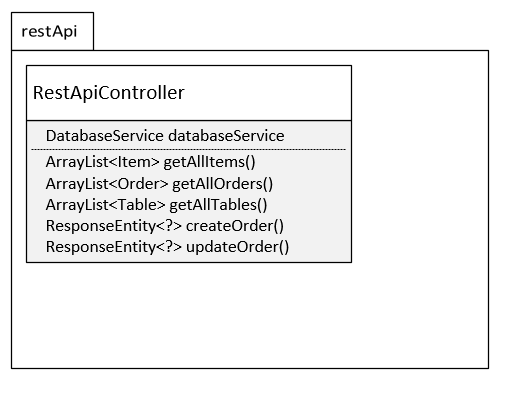


Abbildung 8: Das Software-Package "RestApiController"

Das Package „restApi“ stellt eine Netzwerkschnittstelle zur Verfügung, über die im Netzwerk Anfragen gestellt werden können. Über diese sind Funktionen des Database-Services ansprechbar und somit Daten der Datenbank abruf- und änderbar.

## Klassendesign der Android-Anwendung

Abbildung 9: Software-Package "Kassensystemapplikation"

Die Android-Anwendung besteht aus drei verschiedenen Komponenten. Alle drei sind zusammengefasst in dem Package „dhbw.sa.kassensystemapplication“. In der Klasse „MainActivity“ werden die Informationen gespeichert, welche die Applikation von der Datenbank erhält.

Die zweite Komponente umfasst die Benutzeranwendung. Diese Komponente spiegelt sich in dem Package „fragment“ wieder. Dort werden alle Eingaben des Benutzers behandelt. In diesem Package befindet sich ebenfalls die Schnittstelle zu dem Datenbank-System und kann so die benötigten Informationen kommunizieren.   
Die letzte Komponente besteht aus dem Package „entity“ und umfasst ebenfalls die Tabellen der Datenbank (siehe Kapitel 3.1 Entwurf des Datenbank-Systems). Diese werden in der Applikation für eine korrekte Kommunikation benötigt.

# Implementierung

## 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/>

### Klasse DatabaseService\_Interface

dhbw.sa.kassensystem\_rest.database

**Interface DatabaseService\_Interface**

All Known Implementing Classes:

[DatabaseService](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService.html)

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](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#addItem-dhbw.sa.kassensystem_rest.database.entity.Item-)([Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) item)  Fuegt der Datenbank einen neuen Artikel hinzu. |
| void | [addItemdelivery](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#addItemdelivery-dhbw.sa.kassensystem_rest.database.entity.Itemdelivery-)([Itemdelivery](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Itemdelivery.html) itemdelivery)  Fuegt der Datenbank einen neuen Wareneingang hinzu. |
| void | [addOrder](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#addOrder-dhbw.sa.kassensystem_rest.database.entity.Order-)([Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)  Fuegt der Datenbank eine neue Bestellung hinzu. |
| void | [addTable](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#addTable-dhbw.sa.kassensystem_rest.database.entity.Table-)([Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) table)  Fuegt der Datenbank einen neuen Tisch hinzu. |
| void | [connect](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#connect--)()  Stellt eine Verbindung zur MySQL-Datenbank her. |
| void | [deleteItem](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#deleteItem-int-)(int itemID)  Markiert einen Artikel als nicht verfuegbar. |
| void | [deleteItemdelivery](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#deleteItemdelivery-int-)(int itemdeliveryID)  Loescht einen Wareneingang aus der Datenbank. |
| void | [deleteOrder](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#deleteOrder-int-)(int orderID)  Loescht eine Bestellung aus der Datenbank. |
| void | [deleteTable](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#deleteTable-int-)(int tableID)  Markiert einen Tisch als nicht verfuegbar. |
| void | [disconnect](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#disconnect--)()  Beendet eine bestehende Verbindung mit einem MySQL-Server. |
| java.util.ArrayList<[Itemdelivery](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Itemdelivery.html)> | [getAllItemdeliveries](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#getAllItemdeliveries--)()  Fragt die Wareneingaenge der Datenbank ab. |
| java.util.ArrayList<[Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html)> | [getAllItems](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#getAllItems--)()  Fragt die Artikel der Datenbank ab. |
| java.util.ArrayList<[Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html)> | [getAllOrders](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#getAllOrders--)()  Fragt die Bestellungen der Datenbank ab. |
| java.util.ArrayList<[Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html)> | [getAllTables](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#getAllTables--)()  Fragt die Tische der Datenbank ab. |
| [Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) | [getItemById](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#getItemById-int-)(int itemID)  Liefert ein [Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) in Abhaengigkeit von einer ID. |
| [Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) | [getOrderById](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#getOrderById-int-)(int orderID)  Liefert eine Bestellung in Abhaengigkeit von einer ID. |
| [Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) | [getTableById](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#getTableById-int-)(int tableID)  Liefert einen [Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) in Abhaengigkeit von einer ID. |
| void | [printOrderById](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#printOrderById-int-)(int orderID)  Ausdrucken einer Bestellung in Abhaengigkeit von einer ID; |
| void | [updateItem](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#updateItem-int-dhbw.sa.kassensystem_rest.database.entity.Item-)(int itemID, [Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) item)  Aktualisiert die Daten eines Artikels. |
| void | [updateOrder](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#updateOrder-int-dhbw.sa.kassensystem_rest.database.entity.Order-)(int orderID, [Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)  Aktualisiert die Daten einer Bestellung. |
| void | [updateTable](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/DatabaseService_Interface.html#updateTable-int-dhbw.sa.kassensystem_rest.database.entity.Table-)(int tableID, [Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) table)  Aktualisiert die Daten eines Tisches. |

**Method Detail**

**connect**

void 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](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html)> getAllItems()

Fragt die Artikel der Datenbank ab.

Returns:

Artikel der Datenbank.

**getAllTables**

java.util.ArrayList<[Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html)> getAllTables()

Fragt die Tische der Datenbank ab.

Returns:

Tische der Datenbank.

**getAllOrders**

java.util.ArrayList<[Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html)> getAllOrders()

Fragt die Bestellungen der Datenbank ab.

Returns:

Bestellungen der Datenbank.

**getAllItemdeliveries**

java.util.ArrayList<[Itemdelivery](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Itemdelivery.html)> getAllItemdeliveries()

Fragt die Wareneingaenge der Datenbank ab.

Returns:

Wareneingaenge der Datenbank.

**getOrderById**

[Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) getOrderById(int orderID)

Liefert eine Bestellung in Abhängigkeit von einer ID.

Parameters:

orderID - ID der Bestellung.

Returns:

Order mit angegebener ID.

**getItemById**

[Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) getItemById(int itemID)

Liefert ein [Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) in Abhaengigkeit von einer ID.

Parameters:

itemID - ID des Artikels.

Returns:

Item mit angegebener ID.

**getTableById**

[Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) getTableById(int tableID)

Liefert einen [Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) in Abhaengigkeit von einer ID.

Parameters:

tableID - ID des Tisches.

Returns:

Table mit angegebener ID.

**addItem**

void addItem([Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) item)

Fuegt der Datenbank einen neuen Artikel hinzu.

Parameters:

item - neuer Artikel.

**addTable**

void addTable([Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) table)

Fuegt der Datenbank einen neuen Tisch hinzu.

Parameters:

table - neuer Tisch.

**addOrder**

void addOrder([Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)

Fuegt der Datenbank eine neue Bestellung hinzu.

Parameters:

order - neue Bestellung.

**addItemdelivery**

void addItemdelivery([Itemdelivery](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Itemdelivery.html) itemdelivery)

Fuegt der Datenbank einen neuen Wareneingang hinzu.

Parameters:

itemdelivery - neuer Wareneingang.

**updateItem**

void updateItem(int itemID,

[Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html) item)

Aktualisiert die Daten eines Artikels.

Parameters:

itemID - ID des zu aktualisierenden Artikels.

item - neue Artikeldaten.

**updateTable**

void updateTable(int tableID,

[Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html) table)

Aktualisiert die Daten eines Tisches.

Parameters:

tableID - ID des zu aktualisierenden Tisches.

table - neue Tischdaten.

**updateOrder**

void updateOrder(int orderID,

[Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)

Aktualisiert die Daten einer Bestellung.

Parameters:

orderID - ID der zu aktualisierenden Bestellung.

order - neue Bestellungsdaten.

**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.

**printOrderById**

void printOrderById(int orderID)

Ausdrucken einer Bestellung in Abhaengigkeit von einer ID.

Parameters:

orderID - ID der auszudruckenden Order.

### Klasse RestApiController

**Class RestApiController**

java.lang.Object

dhbw.sa.kassensystem\_rest.restApi.controller.RestApiController

@RestController

@ComponentScan(value="dhbw.sa.kassensystem\_database.database")

@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**

|  |
| --- |
| Constructors |
| **Constructor and Description** |
| [RestApiController](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/restApi/controller/RestApiController.html#RestApiController--)() |

**Method Summary**

|  |  |
| --- | --- |
|  | |
| **Modifier and Type** | **Method and Description** |
| org.springframework.http.ResponseEntity<?> | [createOrder](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/restApi/controller/RestApiController.html#createOrder-dhbw.sa.kassensystem_rest.database.entity.Order-)([Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)  Erstellt eine neue Bestellung in der MySQL-Datenbank. |
| java.util.ArrayList<[Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html)> | [getAllItems](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/restApi/controller/RestApiController.html#getAllItems--)()  Durch das Ansprechen des Pfades "... |
| java.util.ArrayList<[Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html)> | [getAllOrders](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/restApi/controller/RestApiController.html#getAllOrders--)()  Durch das Ansprechen des Pfades "... |
| java.util.ArrayList<[Table](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html)> | [getAllTables](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/restApi/controller/RestApiController.html#getAllTables--)()  Durch das Ansprechen des Pfades "... |
| org.springframework.http.ResponseEntity<?> | [updateOrder](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/restApi/controller/RestApiController.html#updateOrder-int-dhbw.sa.kassensystem_rest.database.entity.Order-)(int orderID, [Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)  Updatet eine bereits existierende Bestellung in der Datenbank. |

**Methods inherited from class java.lang.Object**

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

**Constructor Detail**

**RestApiController**

public RestApiController()

**Method Detail**

**getAllItems**

@RequestMapping(value="/items")

public java.util.ArrayList<[Item](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Item.html)> getAllItems()

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](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html)> getAllOrders()

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](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Table.html)> getAllTables()

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

Returns:

Liste aller Tische der Datenbank.

**createOrder**

@RequestMapping(value="/order/",

method=POST)

public org.springframework.http.ResponseEntity<?> createOrder(@RequestBody

[Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)

Erstellt eine neue Bestellung in der MySQL-Datenbank.

Parameters:

order - neu zu erstellende Bestellung.

Returns:

ResponseEntity, das Erstellen entweder bestätigt oder eine Fehlermeldung liefert.

**updateOrder**

@RequestMapping(value="/order/{orderID}",

method=PUT)

public org.springframework.http.ResponseEntity<?> updateOrder(@PathVariable(value="orderID")

int orderID,

@RequestBody

[Order](https://kassensystem.github.io/DatabaseSystem/dhbw/sa/kassensystem_rest/database/entity/Order.html) order)

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.

## 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/>

### Klasse TableSelect

**public class TableSelection**

extends android.support.v4.app.Fragment

In dieser Klasse wird der Startbildschirm der Applikation erstellt. Dieser wird ebenfalls aufgerufen, wenn angefangen wird eine Bestellung aufzugeben.

Author:

Daniel Schifano

Method Detail

* onCreateView
* public android.view.View onCreateView(android.view.LayoutInflater inflater,
* android.view.ViewGroup container, android.os.Bundle savedInstanceState)

Diese Methode wird aufgerufen wenn das Fragment erstellt wird. Dabei wird der Befehl gegeben, dass alle Artikel, Tische und Bestellungen vom Server angefordert werden. Für "Bestellung aufgeben" wird hier der Startbildschirm initialisiert.

Overrides:

onCreateView in class android.support.v4.app.Fragment

Parameters:

inflater - Instantiiert 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.

### 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.

* Nested Class Summary
* Nested classes/interfaces inherited from class android.support.v4.app.SupportActivity

android.support.v4.app.SupportActivity.ExtraData

* Field Summary

|  |  |
| --- | --- |
| Fields | |
| **Modifier and Type** | **Field and Description** |
| static java.util.ArrayList<[Item](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/entity/Item.html)> | [allItems](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#allItems)  Liste die alle Artikel der Datenbank beinhaltet. |
| static java.util.ArrayList<[Order](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/entity/Order.html)> | [allOrders](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#allOrders)  Liste die alle Bestellungen der Datenbank beinhaltet. |
| static java.util.ArrayList<[Table](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/entity/Table.html)> | [allTables](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#allTables)  Liste die alle Tische der Datenbank beinhaltet. |
| static android.content.Context | [context](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#context)  Der Hintergrund für alle weiteren Klassen wird hier gespeichert. |
| static java.lang.String | [ip](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#ip)  Speichert die IP-Adresse des Servers. |
| static java.util.ArrayList <java.lang.Integer> | [orderItemIDs](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#orderItemIDs)  Liste die alle ArtikelIDs einer Bestellung der Datenbank beinhaltet. |
| static int | [selectedOrderID](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#selectedOrderID)  Zwischenspeicher der ausgewählten BestellungsID. |
| static [Table](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/entity/Table.html) | [selectedTable](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#selectedTable)  Der Tisch an dem die Bestellung stattfindet. |
| static java.lang.String | [url](https://nunay.github.io/Kassensytem-AndroidApplikation/JavaDoc/dhbw/sa/kassensystemapplication/MainActivity.html#url)  Speichert die URL des Servers. |
| **Modifier and Type** | **Method and Description** |
| boolean | [loadURL](file:///F:\JavaDoc\dhbw\sa\kassensystemapplication\MainActivity.html#loadURL--)()  In dieser Methode werden die IP-Adresse und die URL geladen. |
| protected void | [onCreate](file:///F:\JavaDoc\dhbw\sa\kassensystemapplication\MainActivity.html#onCreate-android.os.Bundle-) (android.os.Bundle savedInstanceState)  Diese Methode wird aufgerufen wenn die App gestartet wird. |
| boolean | [onNavigationItemSelected](file:///F:\JavaDoc\dhbw\sa\kassensystemapplication\MainActivity.html#onNavigationItemSelected-android.view.MenuItem-)(android.view.MenuItem item)  Mit dieser Funktion werden die verschiedenen Klassen (Fragments) die im Navigation-Drawer auswählbar sind aufgerufen. |
| void | [showToast](file:///F:\JavaDoc\dhbw\sa\kassensystemapplication\MainActivity.html#showToast-java.lang.String-)(java.lang.String handoverText)  Methode, die den übergebenen Text auf dem Smartphone darstellt. |

* Method Detail
* onCreate

protected void onCreate(android.os.Bundle savedInstanceState)

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.

* **onNavigationItemSelected**

public boolean onNavigationItemSelected(android.view.MenuItem item)  
Mit dieser Funktion werden die verschiedenen Klassen (Fragments) die im Navigation-Drawer auswählbar sind aufgerufen.

Specified by:

onNavigationItemSelected in interface android.support.design.widget.NavigationView.OnNavigationItemSelectedListener

Parameters:

item - Das item das in dem Navigation-Drawer ausgewaehlt/angeklickt wurde.

Returns:

true, wenn die Methode ohne Fehler abgearbeitet werden konnte.

* **loadURL**

public boolean loadURL()  
In dieser Methode werden die IP-Adresse und die URL geladen. Hierfür wird in der Klasse AdjustUrl 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.

# Test

## Test des Datenbank-Systems

Im Folgenden sind die Tests der öffentlichen Methoden des Kassensystem-Managers dokumentiert. Die gesamte Testdokumentation ist im Anhang abrufbar.

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.

### Benutzeranwendung Kassensystem-Manager

#### Abrufen von Datenbankinhalten

##### 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 nicht bezahlten Bestellungen. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetOrder.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Abrufen der Artikel

|  |  |
| --- | --- |
| Anwendungsfall | 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 | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetItems.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Abrufen der Tische

|  |  |
| --- | --- |
| Anwendungsfall | 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 | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetTables.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Abrufen der Wareneingänge

|  |  |
| --- | --- |
| Anwendungsfall | 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 | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetItemdeliveries.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Hinzufügen von Datenbankinhalten

##### Hinzufügen von neuen Artikeln

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddItemData.png |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddedItem.png* |
| **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.*  *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemName.png*  *2.*  *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemQuantity.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Hinzufügen von neuen Tischen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddTableData.png |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddedTable.png* |
| **Sonderfall** | Es wurde kein Name angegeben. |
| Erwartetes Testergebnis | Fehlermeldung mit der Meldung eines fehlenden Namens. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrTableName.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddItemdeliveryData.png |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddedItemdelivery.png* |
| **Sonderfall** | Es wurde keine Anzahl eingegeben. |
| Erwartetes Testergebnis | Fehlermeldung mit der Meldung einer fehlenden Anzahl. |
| Tatsächliches  Testergebnis | C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemdeliveryQuantity.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Bearbeiten von Datenbankinhalten

##### Bearbeiten von Artikeln

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateItemOld.png  Dieser Artikel soll mit den folgenden Daten aktualisiert werden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateItemNew.png |
| Tatsächliches  Testergebnis | In den Daten ist nun der folgende Artikel zu finden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateItemResult.png |
| **Sonderfall** | Es wird kein Name angegeben. |
| Erwartetes Testergebnis | Eine Fehlermeldung, die einen fehlenden Namen anmerkt. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemName.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Bearbeiten von Tischen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateTableOld.png  Dieser Tisch soll mit den folgenden Daten bearbeitet werden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateTableNew.png |
| Tatsächliches  Testergebnis | In der tabellarischen Übersicht ist nun der folgende Tisch zu finden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateTableResult.png |
| **Sonderfall** | Es wurde kein Name angegeben. |
| Erwartetes Testergebnis | Eine Fehlermeldung über den fehlenden Namen. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrTableName.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Löschen von Datenbankinhalten

##### Löschen von Bestellungen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteOrder.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

##### Löschen von Artikeln

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItem.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

##### Löschen von Tischen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteTable.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

##### 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItemdeliveryOld.png  In der tabellarischen Übersicht der Artikel wird die Anzahl des entsprechenden Artikels reduziert:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItemdeliveryQuantityOld.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. Die Anzahl des Artikels in der Artikel-Übersicht wurde aktualisiert:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItemdeliveryQuantityNew.png |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

#### Ausdrucken einer Bestellung

|  |  |
| --- | --- |
| Anwendungsfall | 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 Bon-Drucker wird der Kundenbeleg ausgedruckt. |
| Erwartetes Testergebnis | Der Bon-Drucker druckt einen Kundenbeleg mit den folgenden Bestellungsdaten:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\printOrder.png |
| Tatsächliches  Testergebnis | Ein Ausdruck wurde ausgegeben:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\img108.jpg |
| **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 |

## 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.

**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:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Tabelle Tische aus der Datenbank.png |
| Tatsächliches  Testergebnis | Das Dropdow-Menü wurde wie folgt dargestellt:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-111103.png |
| **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:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20171208-195008.png |
| **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:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-115102.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

**Auswählen der Tische**

|  |  |
| --- | --- |
| Anwendungsfall | 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“.  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-123422.png |
| 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:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-141703.png |
| **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:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-123915.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

**Artikel Einsehen**

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-141703.png |
| **Sonderfall 1** | Die Verbindung zum Server wird getrennt. |
| Erwartetes Testergebnis | Die Artikel werden weiterhin angezeigt. |
| Tatsächliches  Testergebnis | Der Bildschirm des Smartphones wird folgendermaßen dargestellt:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-141703.png |
| **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:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-151658.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

**Artikel einer Bestellung hinzufügen**

|  |  |
| --- | --- |
| Anwendungsfall | 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 |

**Gesamtpreis darstellen**

|  |  |
| --- | --- |
| Anwendungsfall | Den aktuellen Gesamtpreis einer Bestellung anzeigen. |
| Verwendete Methode | updateSum (boolean isAdd, String itemName) (Klasse: “ItemSelect”) |
| **Normalablauf** | Immer wenn ein Artikel einer Bestellung hinzugefügt oder abgezogen wird, errechnet die Applikation die Gesamtsumme der Bestellung und zeigt diese an. |
| Erwartetes Testergebnis | Die aktuelle Summe wird dargestellt.  Mit folgenden Artikeln aus der Datenbank:  2-mal Pommes  1-mal Rote Wurst  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5.2.5 Gesamtpreis darstellen.png  Erwarteter Gesamtpreis der Bestellung: 8.50 € |
| Tatsächliches  Testergebnis | Der von der Applikation errechnete Gesamtpreis:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-183345.png |
| **Sonderfall** | Keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | -- |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

**Bestellung abschicken**

|  |  |
| --- | --- |
| Anwendungsfall | 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 ausgewählten Artikel, den Gesamtpreis, die Bestellungs-ID, das Datum inklusive Uhrzeit wann die Bestellung erstellt wurde und ob die Bestellung bereits bezahlt wurde.  Mit folgenden Daten wird getestet:   * 2-mal Pommes, 1-mal Rote Wurst * Preis: 8.50 € * Tisch-ID: 1 * Datum und Uhrzeit: 02.01.2018, 18:49 Uhr * Nur bestellt nicht bezahlt |
| Tatsächliches  Testergebnis | Die Datenbank sieht wie folgt aus:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5.2.6 Bestellung abschicken.png |
| **Sonderfall 1** | Wenn an diesem Tisch bereits eine Bestellung besteht, die bis zu diesem Zeitpunkt noch nicht bezahlt wurde. 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 den neu hinzugefügten Artikeln. Der Preis wird ebenfalls aktualisiert. Der Test wird mit der gleichen Bestellung die im Normalfall getestet wurde, gestartet.  Es soll eine zusätzliche Rote Wurst bestellt werden. |
| Tatsächliches  Testergebnis | Die Datenbank sieht wie folgt aus:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5.2.6 Bestellung abschicken sonderfall1.png |
| **Sonderfall 2** | Die Verbindung zum Server wird während eine Bestellung erstellt wird unterbrochen. |
| Erwartetes Testergebnis | Die Bestellung wird nicht gespeichert und eine Fehlermeldung wird angezeigt. |
| Tatsächliches  Testergebnis | Das Smartphone gibt folgende Fehlermeldung aus:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-194633.png |
| Test  bestanden | Normalfall: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

**Rechnung für den Kunden anfordern**

|  |  |
| --- | --- |
| Anwendungsfall | Eine von der Bedienung zusammengestellte Bestellung soll an die Küche geschickt werden und direkt vom Kunden bezahlt werden. |
| Verwendete Methode | doInBackground() (Klasse: “ItemSelect”) |
| **Normalablauf** | Die Bedienung stellt für den Kunden eine Bestellung zusammen. Anschließend klickt sie auf den Button „BEZAHLEN“ und sendet der Küche die Bestellung. Am Bon-Drucker werden zwei Belege ausgedruckt. Einen für die Küche und einen für den Kunden. |
| Erwartetes Testergebnis | In der Datenbank wird eine neue Bestellung erstellt. Diese beinhaltet den ausgewählten Tisch, die ausgewählten Artikel, den Gesamtpreis, die Bestellungs-ID, das Datum inklusive Uhrzeit wann die Bestellung erstellt wurde und ob die Bestellung bereits bezahlt wurde.  Mit folgenden Daten wird getestet:   * 2-mal Pommes, 1-mal Rote Wurst * Preis: 8.50 € * Tisch-ID: 1 * Datum und Uhrzeit: 03.01.2018, 15:45 Uhr * bezahlt |
| Tatsächliches  Testergebnis | Die Datenbank sieht wie folgt aus:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5.2.7 Bezahlung abschicken.png |
| **Sonderfall 1** | Wenn an diesem Tisch bereits eine Bestellung besteht, die bis zu diesem Zeitpunkt noch nicht bezahlt wurde. 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 den neu hinzugefügten Artikeln. Der Preis wird ebenfalls aktualisiert. Der Test wird mit der gleichen Bestellung die im Normalfall getestet wurde, gestartet.  Es soll eine zusätzliche Rote Wurst bestellt werden. |
| Tatsächliches  Testergebnis | Die Datenbank sieht wie folgt aus.  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\5.2.7 Bezahlung abschicken2.png |
| **Sonderfall 2** | Die Verbindung zum Server wird während eine Bestellung erstellt wird unterbrochen. |
| Erwartetes Testergebnis | Die Bestellung wird nicht gespeichert und eine Fehlermeldung wird angezeigt. |
| Tatsächliches  Testergebnis | Das Smartphone gibt folgende Fehlermeldung aus:  C:\Users\Dani\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20180102-194633.png |
| Test  bestanden | Normalfall: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

# Fazit

Zusammenfassend kann gesagt werden, dass die Grundstruktur des Systems entwickelt wurde. Mit dem System kann der normale Tagesablauf in einer Gastronomie bewältigt werden.   
Mit Hilfe des Datenbank Managers kann unter anderem der Bestand, sowie der Wareneingang und Warenausgang überprüft und bearbeitet werden. Die Android-Anwendung hilft dem Personal/den Bedienungen dabei, Bestellungen der Kunden aufzunehmen und abzurechnen.

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.   
Des Weiteren kann das System noch um eine Login-Funktion erweitert werden. Dann kann nicht mehr jeder auf das System zugreifen, sondern lediglich die vom System-Administrator authentifizierte Personen.

Dies sind nur zwei Möglichkeiten die Struktur zu erweitern. Andere Funktionen wären zum Beispiel: In der Android-Applikation kann die Rechnung getrennt werden, die Artikel in der Applikation nach Getränken und Speisen sortiert, der jeweiligen Bestellung ein Kundenwunsch beigefügt werden und weiteres.

# Anhang

## Installationsanweisung

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

### 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

1. 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"

1. 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"

1. 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

1. Download des aktuellsten DatabaseSystems

* <https://github.com/Kassensystem/DatabaseSystem/releases/latest>
* Zum Download auf "Source Code (zip)" klicken
* Entpacken der Dateien

1. Download der aktuellsten ManagerApplication

* <https://github.com/Kassensystem/ManagerApplication/releases/latest>
* Zum Download auf "Source Code (zip)" klicken
* Entpacken der Dateien

1. Anpassen der Firewall

* Um dem Server eine Kommunikation im lokalen Netzwerk zu ermöglichen die folgenden Änderungen durchführen:
  + Windows-Firewall öffnen
  + Firewall komplett deaktivieren

1. Starten des Servers

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

1. 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"

### 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

1. Download der APK auf das Smartphone:

<https://github.com/Nunay/Kassensytem-AndroidApplikation>

1. Heruntergeladene Datei öffnen und installieren

Ordner *Dateien* öffnen 🡪 *app-release.apk* öffnen 🡪 installieren

1. Applikation starten

### 

# 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.

## Datenbank-Modul

### DatabaseService-Klasse

#### Verbindung zur Datenbank

|  |  |
| --- | --- |
| Anwendungsfall | Der DatabaseService verbindet sich mit der MySQL-Datenbank. |
| Verwendete Methode | Void connect() Stellt eine Verbindung zur MySQL-Datenbank her. |
| **Normalablauf** | Wenn die MySQL-Datenbank definitionsgemäß installiert ist und läuft, wird mit dieser eine Verbindung hergestellt. |
| Erwartetes Testergebnis | Ausgabe der erfolgreichen Verbindungsaufnahme in der Konsole. |
| Tatsächliches  Testergebnis | 2017-11-26 13:45:00.113 MYSQL-Info Connecting database...  2017-11-26 13:45:00.883 MYSQL-Info Database connected! |
| **Sonderfall** | Die MySQL-Datenbank wurde nicht definitionsgemäß installiert oder läuft nicht auf dem Computer. |
| Erwartetes Testergebnis | Ausgabe der Fehlermeldung in der Konsole. |
| Tatsächliches  Testergebnis | 2017-11-26 13:47:33.331 MYSQL-Info Connecting database...  2017-11-26 13:47:35.933 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Abrufen von Datenbankinhalten

##### Artikel-Daten

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen von Artikel-Daten aus der Datenbank. |
| Verwendete Methode | java.util.ArrayList<Item> getAllItems() Fragt die Artikel der Datenbank ab. |
| **Normalablauf** | Alle Artikel werden als Item-Objekte aus der MySQL-Datenbank abgefragt und anschließend zu Testzwecken ausgegeben. |
| Erwartetes Testergebnis | Rückgabe einer Liste von Item-Objekten. Validierung über eine Konsolenausgabe der folgenden Datenbankinhalte:   |  |  |  |  | | --- | --- | --- | --- | | **itemID** | **name** | **retailprice** | **available** | | 1 | Pizza | 7 | 1 | | 2 | Pommes | 3.5 | 1 | | 3 | Nudeln | 11 | 1 | | 4 | Lasagne | 7.5 | 1 | | 5 | Kaffee | 2.3 | 1 | | 6 | Cola | 1.5 | 0 | | 9 | Bier | 3.3 | 0 | | 10 | Brot | 5 | 0 | |
| Tatsächliches  Testergebnis | 2017-11-26 14:01:13.697 MYSQL-Info Connecting database...  2017-11-26 14:01:14.654 MYSQL-Info Database connected!  2017-11-26 14:01:14.654 MYSQL-Info Getting Items from MySQL-Database.  -----------------All-Items-Test-----------------  1 Pizza 7.0 4 true  2 Pommes 3.5 7 true  3 Nudeln 11.0 4 true  4 Lasagne 7.5 6 true  5 Kaffee 2.3 45 true  6 Cola 1.5 6 false  9 Bier 3.3 10 false  10 Brot 5.0 10 false |
| **Sonderfall** | Keine Verbindung zur MySQL-Datenbank aufbaubar. |
| Erwartetes Testergebnis | Fehlerausgabe über den Verbindungsfehler in der Konsole. |
| Tatsächliches  Testergebnis | 2017-11-26 14:30:59.934 MYSQL-Info Connecting database...  2017-11-26 14:31:02.756 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen eines Artikels mit einer bestimmten ID. |
| Verwendete Methode | Item getItemById(int itemID) Liefert ein Item in Abhängigkeit von einer ID |
| **Normalablauf** | Artikel mit der ID existiert in der Datenbank und wird als Item-Objekt zurückgeliefert. |
| Erwartetes Testergebnis | Rückgabe des Item-Objekts. Validierung über eine Konsolenausgabe des folgenden Artikels:   |  |  |  |  | | --- | --- | --- | --- | | **itemID** | **name** | **retailprice** | **available** | | 18 | Steak | 15 | 1 | |
| Tatsächliches  Testergebnis | 2017-11-26 15:02:28.831 MYSQL-Info Connecting database...  2017-11-26 15:02:29.549 MYSQL-Info Database connected!  2017-11-26 15:02:29.549 MYSQL-Info Getting Order with ID 18.  -----------------Item-By-ID-Test-----------------  18 Steak 15.0 17 true |
| **Sonderfall** | 1. Keine Verbindung zur MySQL-Datenbank aufbaubar. 2. Die angegebene ID existiert nicht. 3. Es wurde keine ID angegeben. |
| Erwartetes Testergebnis | 1. Fehlerausgabe über den Verbindungsfehler in der Konsole. 2. Fehlermeldung, die eine unbekannte ID meldet. 3. Fehlermeldung, die eine fehlende ID meldet. |
| Tatsächliches  Testergebnis | 2017-11-26 15:36:24.184 MYSQL-Info Connecting database...  2017-11-26 15:36:27.124 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  2.  2017-11-26 15:41:14.551 MYSQL-Info Connecting database...  2017-11-26 15:41:15.418 MYSQL-Info Database connected!  -----------------Item-By-ID-Test-----------------  2017-11-26 15:41:15.418 MYSQL-Info Getting Item with ID 7.  2017-11-26 15:41:15.489 MYSQL-ERROR Item with ID 7 doesn't exist in the database.  Exception in thread "main" java.lang.NullPointerException: Item-ID 7 not found.  3.  2017-11-26 15:42:19.738 MYSQL-Info Connecting database...  2017-11-26 15:42:20.552 MYSQL-Info Database connected!  Exception in thread "main" java.lang.NullPointerException: No Item-ID given.  -----------------Item-By-ID-Test-----------------  2017-11-26 15:42:20.553 MYSQL-ERROR Item-ID may not be null. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Bestellungs-Daten

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen von allen Bestellungen aus der Datenbank. |
| Verwendete Methode | java.util.ArrayList<Order> getAllOrders() Fragt die Bestellungen der Datenbank ab. |
| **Normalablauf** | Alle Bestellungen werden als Order-Objekte aus der MySQL-Datenbank abgefragt und anschließend zu Testzwecken ausgegeben. |
| Erwartetes Testergebnis | Rückgabe einer Liste von Order-Objekten. Validierung über eine Konsolenausgabe der folgenden Datenbankinhalte:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **orderID** | **itemIDs** | **price** | **date** | **tableID** | **paid** | | 1 | 2;1; | 2.3 | 2017-10-19 15:40:25 | 5 | 1 | | 3 | 1;3; | 18 | 2017-10-23 10:44:48 | 6 | 1 | | 4 | 1;2;3;4;5; | 31.3 | 2017-11-05 15:38:42 | 3 | 0 | | 5 | 2;3; | 9.8 | 2017-11-08 23:31:33 | 5 | 1 | | 12 | 4;5; | 9.8 | 2017-11-14 21:32:22 | 5 | 1 | | 17 | 3;5; | 33.99 | 2017-11-12 14:42:34 | 10 | 1 | |
| Tatsächliches  Testergebnis | 2017-11-26 14:27:10.445 MYSQL-Info Connecting database...  2017-11-26 14:27:11.128 MYSQL-Info Database connected!  2017-11-26 14:27:11.128 MYSQL-Info Getting Orders from MySQL-Database.  ----------------All-Orders-Test----------------  1 2;1; 2.3 19.10.2017 15:40:25 5 true  3 1;3; 18.0 23.10.2017 10:44:48 6 true  4 1;2;3;4;5; 31.3 05.11.2017 15:38:42 3 false  5 2;3; 9.8 08.11.2017 23:31:33 5 true  12 4;5; 9.8 14.11.2017 21:32:22 5 true  17 3;5; 33.99 12.11.2017 14:42:34 10 true |
| **Sonderfall** | Keine Verbindung zur MySQL-Datenbank aufbaubar. |
| Erwartetes Testergebnis | Fehlerausgabe über den Verbindungsfehler in der Konsole. |
| Tatsächliches  Testergebnis | 2017-11-26 14:31:39.702 MYSQL-Info Connecting database...  2017-11-26 14:31:42.346 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen einer Order mit einer bestimmten ID. |
| Verwendete Methode | Order getOrderById(int orderID) Liefert eine Bestellung in Abhängigkeit von einer ID. |
| **Normalablauf** | Bestellung mit der ID existiert in der Datenbank und wird als Order-Objekt zurückgeliefert. |
| Erwartetes Testergebnis | Rückgabe des Order-Objekts. Validierung über eine Konsolenausgabe der folgenden Bestellung:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **orderID** | **itemIDs** | **price** | **date** | **tableID** | **paid** | | 47 | 1;5;18;6;6;6;6; | 30.3 | 2017-11-23 15:33:09 | 14 | 0 | |
| Tatsächliches  Testergebnis | 2017-11-26 15:47:07.692 MYSQL-Info Connecting database...  2017-11-26 15:47:08.487 MYSQL-Info Database connected!  -----------------Order-By-ID-Test-----------------  2017-11-26 15:47:08.487 MYSQL-Info Getting Order with ID 47.  2017-11-26 15:47:08.487 MYSQL-Info Getting Orders from MySQL-Database.  47 1;5;18;6;6;6;6; 30.3 23.11.2017 15:33:09 14 false |
| **Sonderfall** | 1. Keine Verbindung zur MySQL-Datenbank aufbaubar. 2. Die angegebene ID existiert nicht. 3. Es wurde keine ID angegeben. |
| Erwartetes Testergebnis | 1. Fehlerausgabe über den Verbindungsfehler in der Konsole. 2. Fehlermeldung, die eine unbekannte ID meldet. 3. Fehlermeldung, die eine fehlende ID meldet. |
| Tatsächliches  Testergebnis | 2017-11-26 15:26:59.049 MYSQL-Info Connecting database...  2017-11-26 15:27:01.781 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  2.  2017-11-26 15:28:04.033 MYSQL-Info Connecting database...  2017-11-26 15:28:04.745 MYSQL-Info Database connected!  -----------------Order-By-ID-Test-----------------  2017-11-26 15:28:04.745 MYSQL-Info Getting Order with ID 7.  Exception in thread "main" java.lang.NullPointerException: Order-ID 7 not found.  2017-11-26 15:28:04.789 MYSQL-ERROR Order with ID 7 doesn't exist in the database.  3.  2017-11-26 15:29:08.224 MYSQL-Info Connecting database...  2017-11-26 15:29:09.066 MYSQL-Info Database connected!  Exception in thread "main" java.lang.NullPointerException: No Order-ID given.  -----------------Order-By-ID-Test-----------------  2017-11-26 15:29:09.067 MYSQL-ERROR Order-ID may not be null. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Tisch-Daten

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen von Tisch-Daten aus der Datenbank. |
| Verwendete Methode | java.util.ArrayList<Table> getAllTables() Fragt die Tische der Datenbank ab |
| **Normalablauf** | Alle Tische werden als Table-Objekte aus der MySQL-Datenbank abgefragt und anschließend zu Testzwecken ausgegeben. |
| Erwartetes Testergebnis | Rückgabe einer Liste von Table-Objekten. Validierung über eine Konsolenausgabe der folgenden Datenbankinhalte:   |  |  |  | | --- | --- | --- | | **tableID** | **name** | **available** | | 1 | A1 | 1 | | 2 | A2 | 1 | | 3 | A3 | 1 | | 4 | B1 | 1 | | 5 | B2 | 1 | | 6 | B3 | 1 | | 7 | C1 | 1 | | 8 | C2 | 1 | |
| Tatsächliches  Testergebnis | 2017-11-26 14:28:33.563 MYSQL-Info Connecting database...  2017-11-26 14:28:34.333 MYSQL-Info Database connected!  2017-11-26 14:28:34.333 MYSQL-Info Getting Tables from MySQL-Database.  -------------All-Tables-Test---------------  1 A1  2 A2  3 A3  4 B1  5 B2  6 B3  7 C1  8 C2 |
| **Sonderfall** | Keine Verbindung zur MySQL-Datenbank aufbaubar. |
| Erwartetes Testergebnis | Fehlerausgabe über den Verbindungsfehler in der Konsole. |
| Tatsächliches  Testergebnis | 2017-11-26 14:30:16.657 MYSQL-Info Connecting database...  2017-11-26 14:30:19.492 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

|  |  |
| --- | --- |
| Usecase | Abfragen eines Tisches mit einer bestimmten ID. |
| Verwendete Methode | Table getTableById(int tableID) Liefert eine Table in Abhängigkeit von einer ID. |
| **Normalablauf** | Bestellung mit der ID existiert in der Datenbank und wird als Order-Objekt zurückgeliefert. |
| Erwartetes Testergebnis | Rückgabe des Table-Objekts. Validierung über eine Konsolenausgabe des folgenden Tisches:   |  |  |  | | --- | --- | --- | | **tableID** | **name** | **available** | | 9 | C3 | 1 | |
| Tatsächliches  Testergebnis | 2017-11-26 15:53:21.071 MYSQL-Info Connecting database...  2017-11-26 15:53:22.259 MYSQL-Info Database connected!  -----------------Table-By-ID-Test-----------------  2017-11-26 15:53:22.259 MYSQL-Info Getting Table with ID 9.  9 C3 true |
| **Sonderfall** | 1. Keine Verbindung zur MySQL-Datenbank aufbaubar. 2. Die angegebene ID existiert nicht. 3. Es wurde keine ID angegeben. |
| Erwartetes Testergebnis | 1. Fehlerausgabe über den Verbindungsfehler in der Konsole. 2. Fehlermeldung, die eine unbekannte ID meldet. 3. Fehlermeldung, die eine fehlende ID meldet. |
| Tatsächliches  Testergebnis | 1.  2017-11-26 16:04:47.801 MYSQL-Info Connecting database...  2017-11-26 16:04:50.394 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  2.  2017-11-26 16:05:57.778 MYSQL-Info Connecting database...  2017-11-26 16:05:58.569 MYSQL-Info Database connected!  -----------------Table-By-ID-Test-----------------  2017-11-26 16:05:58.569 MYSQL-Info Getting Table with ID 16.  2017-11-26 16:05:58.569 MYSQL-Info Getting Tables from MySQL-Database.  2017-11-26 16:05:58.600 MYSQL-ERROR Table with ID 16 doesn't exist in the database.  Exception in thread "main" java.lang.NullPointerException: Table-ID 16 not found.  3.  2017-11-26 16:05:16.382 MYSQL-Info Connecting database...  Exception in thread "main" java.lang.NullPointerException: No Table-ID given.  2017-11-26 16:05:17.155 MYSQL-Info Database connected!  -----------------Table-By-ID-Test-----------------  2017-11-26 16:05:17.155 MYSQL-ERROR Table-ID may not be null. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Wareneingans-Daten

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen von Wareneingangs-Daten aus der Datenbank. |
| Verwendete Methode | java.util.ArrayList<Itemdelivery> getAllItemdeliveries() Fragt die Wareneingaenge der Datenbank ab. |
| **Normalablauf** | Alle Wareneingänge werden als Itemdelivery-Objekte aus der MySQL-Datenbank abgefragt und anschließend ausgegeben. |
| Erwartetes Testergebnis | Rückgabe einer Liste von Itemdelivery-Objekten. Validierung über eine Konsolenausgabe der folgenden Datenbankinhalte:   |  |  |  | | --- | --- | --- | | **itemdeliveryID** | **itemID** | **quantity** | | 1 | 1 | 10 | | 2 | 2 | 10 | | 3 | 3 | 10 | | 4 | 4 | 10 | | 5 | 5 | 10 | | 6 | 6 | 10 | | 7 | 9 | 10 | | 8 | 10 | 10 | |
| Tatsächliches  Testergebnis | 2017-11-26 14:34:40.782 MYSQL-Info Connecting database...  2017-11-26 14:34:41.664 MYSQL-Info Database connected!  2017-11-26 14:34:41.664 MYSQL-Info Getting Itemdeliveries from MySQL-Database.  -------------All-Itemdeliveries-Test---------------  1 1 10  2 2 10  3 3 10  4 4 10  5 5 10  6 6 10  7 9 10  8 10 10 |
| **Sonderfall** | Keine Verbindung zur MySQL-Datenbank aufbaubar. |
| Erwartetes Testergebnis | Fehlerausgabe über den Verbindungsfehler in der Konsole. |
| Tatsächliches  Testergebnis | 2017-11-26 14:52:19.976 MYSQL-Info Connecting database...  2017-11-26 14:52:22.625 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen eines Wareneingangs mit einer bestimmten ID. |
| Verwendete Methode | Itemdelivery getItemdeliveryById(int itemdeliveryID)  Liefert eine Itemdelivery in Abhängigkeit von einer ID. |
| **Normalablauf** | Wareneingang mit der ID existiert in der Datenbank und wird als Itemdelivery-Objekt zurückgeliefert. |
| Erwartetes Testergebnis | Rückgabe des Itemdelivery-Objekts. Validierung über eine Konsolenausgabe des folgenden Wareneingangs:   |  |  |  | | --- | --- | --- | | **itemdeliveryID** | **itemID** | **quantity** | | 23 | 26 | 500 | |
| Tatsächliches  Testergebnis | 2017-11-26 19:15:24.532 MYSQL-Info Connecting database...  2017-11-26 19:15:25.265 MYSQL-Info Database connected!  -----------------Itemdelivery-By-ID-Test-----------------  2017-11-26 19:15:25.265 MYSQL-Info Getting Itemdeliveries from MySQL-Database.  23 26 500 |
| **Sonderfall** | 1. Keine Verbindung zur MySQL-Datenbank aufbaubar. 2. Die angegebene ID existiert nicht. 3. Es wurde keine ID angegeben. |
| Erwartetes Testergebnis | 1. Fehlerausgabe über den Verbindungsfehler in der Konsole. 2. Fehlermeldung, die eine unbekannte ID meldet. 3. Fehlermeldung, die eine fehlende ID meldet. |
| Tatsächliches  Testergebnis | 2017-11-26 19:31:18.553 MYSQL-Info Connecting database...  2017-11-26 19:31:21.384 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  2.  2017-11-26 19:32:30.124 MYSQL-Info Connecting database...  2017-11-26 19:32:30.821 MYSQL-Info Database connected!  -----------------Itemdelivery-By-ID-Test-----------------  2017-11-26 19:32:30.821 MYSQL-Info Getting Itemdeliveries from MySQL-Database.  Exception in thread "main" java.lang.NullPointerException  3.  2017-11-26 19:35:13.716 MYSQL-Info Connecting database...  2017-11-26 19:35:14.612 MYSQL-Info Database connected!  -----------------Itemdelivery-By-ID-Test-----------------  2017-11-26 19:35:14.612 MYSQL-ERROR Itemdelivery-ID may not be null. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Hinzufügen von Datenbankinhalten

##### Artikel hinzufügen

|  |  |
| --- | --- |
| Anwendungsfall | Hinzufügen eines neuen Artikels |
| Verwendete Methode | void addItem(Item item) Fügt der Datenbank einen neuen Artikel hinzu. |
| **Normalablauf** | Ein neuer Eintrag in der Datenbanktabelle Item wird erzeugt. Für die angegebene Anzahl des Artikels wird außerdem ein neuer Wareneingang angelegt. |
| Erwartetes Testergebnis | Erfolgreiches Erstellen eines neuen Item- und WareneingangsEintrags in der Datenbank. Validierung über einen Abgleich der Datenbank nach einem neu erzeugten Item mit folgenden Daten:  String name = **"Pizza"**; **double** price = 5.99; **int** quantity = 15; **boolean** avaiable = **true**; |
| Tatsächliches  Testergebnis | Konsolenausgabe*:*  2017-11-27 13:01:57.813 MYSQL-Info Connecting database...  2017-11-27 13:01:58.610 MYSQL-Info Database connected!  -----------------Add-Item-Test-----------------  0 Pizza 5.99 15 true  2017-11-27 13:01:58.610 MYSQL-Info Adding Item to MySQL-Database.  2017-11-27 13:01:58.641 MYSQL-Info Adding Itemdelivery to MySQL-Database.  Neue Datenbankeinträge:  *Artikel-Tabelle:*   |  |  |  |  | | --- | --- | --- | --- | | **itemID** | **name** | **retailprice** | **available** | | 37 | Pizza | 5.99 | 1 |   *Wareneingangs-Tabelle:*   |  |  |  | | --- | --- | --- | | **itemdeliveryID** | **itemID** | **quantity** | | 47 | 37 | 15 | |
| **Sonderfall** | 1. Es besteht keine Verbindung zur MySQL-Datenbank. 2. Eines der Attribute wurde nicht gesetzt. 3. Es wurde dem neuen Artikel eine ID vergeben. |
| Erwartetes Testergebnis | 1. Die Fehlerhafte Verbindungsaufnahme wird in einer Konsolenausgabe gemeldet.  2. Das fehlende Attribut wird als fehlend gemeldet und kein Artikel hinzugefügt.  3. Es wird rückgemeldet, dass die ID nicht vom Anwender, sondern von dem Datenbank-Server gesetzt wird. |
| Tatsächliches  Testergebnis | *1.*  2017-11-27 13:20:20.457 MYSQL-Info Connecting database...  2017-11-27 13:20:23.051 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-27 13:26:07.176 MYSQL-Info Connecting database...  2017-11-27 13:26:07.895 MYSQL-Info Database connected!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Der Artikel ist unvollständig! Folgende Attribute fehlen: Name Available  -----------------Add-Item-Test-----------------  0 5.99 15 false  2017-11-27 13:26:07.895 MYSQL-Info Adding Item to MySQL-Database.  2017-11-27 13:26:07.895 MYSQL-ERROR Name is missing.  2017-11-27 13:26:07.895 MYSQL-ERROR New Item cannot be set unavailable.  2017-11-27 13:26:07.895 MYSQL-ERROR Item was not added to the Database!  *3.*  2017-11-27 13:29:38.309 MYSQL-Info Connecting database...  2017-11-27 13:29:39.231 MYSQL-Info Database connected!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Es darf keine ID übergeben werden. Die ID wird vom Datenbank-Server gewählt!  -----------------Add-Item-Test-----------------  4 5.99 20 false  2017-11-27 13:29:39.231 MYSQL-Info Adding Item to MySQL-Database.  2017-11-27 13:29:39.231 MYSQL-ERROR ID may not be set by the user.  2017-11-27 13:29:39.231 MYSQL-ERROR Item was not added to the Database! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

**Bestellung hinzufügen**

|  |  |
| --- | --- |
| Anwendungsfall | Hinzufügen einer neuen Bestellung. |
| Verwendete Methode | void addOrder(Order order) Fügt der Datenbank eine neue Bestellung hinzu. |
| **Normalablauf** | Es wird eine neue gültige Bestellung der Datenbank-Tabelle Order hinzugefügt. |
| Erwartetes Testergebnis | Erfolgreiches Erstellen eines neuen Order-Eintrags in der Datenbank. Validierung über einen Abgleich der Datenbank nach einer neu erzeugten Order mit folgenden Daten:  String itemIDs = **"12;13;15"**; **int** tableID = 8; **double** price = 3.3; **boolean** paid = **true**;  Außerdem wird die Bestellung für die Küche audgedruckt, wenn paid = false gesetzt wird. Wenn paid = true gesetzt, wird einmal ein Küchenbeleg und zusätzlich ein Kundenbeleg ausgedruckt. |
| Tatsächliches  Testergebnis | Konsolenausgabe*:*  2017-11-27 15:52:30.328 MYSQL-Info Connecting database...  2017-11-27 15:52:31.063 MYSQL-Info Database connected!  -----------------Add-Order-Test-----------------  0 12;13;15 3.3 27.11.2017 15:52:31 8 true  Neuer Datenbankeintrag:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **orderID** | **itemIDs** | **price** | **date** | **tableID** | **paid** | | 58 | 12;13;15 | 3.3 | 2017-11-27 15:52:31 | 8 | 1 |   Beide Belege wurden ausgedruckt.  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\addOrderTest.jpgC:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\addOrderTest.jpg |
| **Sonderfall** | 1. Verbindungsproblem mit dem MySQL-Server 2. Eines der Attribute wurde nicht gesetzt. 3. Eine der Item-IDs existiert nicht in der Datenbank. 4. Die Table-ID existiert nicht in der Datenbank. 5. Es wurde eine Order-ID gesetzt. |
| Erwartetes Testergebnis | 1. Das Verbindungsproblem wird gemeldet.  2. Die fehlenden Attribute (Artikel und Tisch) werden in einer Meldung zurückgegeben.  3. Die nichtexistierende Item-ID wird gemeldet.  4. Die nichtexistierende Table-ID wird gemeldet.  5. Es wird rückgemeldet, dass die ID nicht vom Anwender, sondern von dem Datenbank-Server gesetzt wird. |
| Tatsächliches  Testergebnis | *1.*  2017-11-27 15:56:38.190 MYSQL-Info Connecting database...  2017-11-27 15:56:40.769 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-28 13:27:44.886 MYSQL-Info Connecting database...  2017-11-28 13:27:45.624 MYSQL-Info Database connected!  -----------------Add-Order-Test-----------------  0 0.0 28.11.2017 13:27:45 0 true  2017-11-28 13:27:45.625 MYSQL-Info Adding Order to MySQL-Database.  2017-11-28 13:27:45.715 MYSQL-ERROR Item-IDs missing.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Die Bestellung ist unvollständig! Die folgenden Parameter fehlen: Artikel Tisch  2017-11-28 13:27:45.715 MYSQL-ERROR Table-ID missing.  2017-11-28 13:27:45.715 MYSQL-ERROR Order was not added to the database!  *3.*  2017-11-28 13:39:13.824 MYSQL-Info Connecting database...  2017-11-28 13:39:14.574 MYSQL-Info Database connected!  -----------------Add-Order-Test-----------------  0 15;7;13 0.0 28.11.2017 13:39:14 8 true  2017-11-28 13:39:14.575 MYSQL-Info Adding Order to MySQL-Database.  2017-11-28 13:39:14.677 MYSQL-ERROR One or multiple Item-IDs do not exist in the database!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Eine oder mehrere angegebene Artikel-IDs existieren nicht in der Datenbank.  2017-11-28 13:39:14.677 MYSQL-ERROR Order was not added to the Database!  *4.*  2017-11-28 13:45:11.977 MYSQL-Info Connecting database...  2017-11-28 13:45:12.718 MYSQL-Info Database connected!  -----------------Add-Order-Test-----------------  0 15;13 0.0 28.11.2017 13:45:12 20 true  2017-11-28 13:45:12.719 MYSQL-Info Adding Order to MySQL-Database.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Die angegebene Table-ID existiert nicht in der Datenbank!  2017-11-28 13:45:12.817 MYSQL-ERROR The Table-ID does not exist in the database!  2017-11-28 13:45:12.818 MYSQL-ERROR Order was not added to the database!  *5.*  2017-11-28 13:42:13.595 MYSQL-Info Connecting database...  2017-11-28 13:42:14.348 MYSQL-Info Database connected!  -----------------Add-Order-Test-----------------  4 15;13 0.0 28.11.2017 13:42:14 8 true  2017-11-28 13:42:14.350 MYSQL-Info Adding Order to MySQL-Database.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Es darf keine ID übergeben werden. Die ID wird vom Datenbank-Server gewählt!  2017-11-28 13:42:14.429 MYSQL-ERROR ID may not be set by the user. 2017-11-28 13:42:14.429 MYSQL-ERROR Order was not added to the Database! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Tisch hinzufügen

|  |  |
| --- | --- |
| Anwendungsfall | Hinzufügen eines neuen Tisches zur Datenbank. |
| Verwendete Methode | void addTable(Table table) Fügt der Datenbank einen neuen Tisch hinzu. |
| **Normalablauf** | Es wird ein neuer gültiger Tisch der Datenbank-Tabelle Table hinzugefügt. |
| Erwartetes Testergebnis | Erfolgreiches Erstellen eines neuen Tisch-Eintrags in der Datenbank. Validierung über einen Abgleich der Datenbank nach einem neu erzeugten Tisch mit folgenden Daten:  String name = **"Tisch5"**; **boolean** available = **true**; |
| Tatsächliches  Testergebnis | *Konsolenausgabe:*  2017-11-28 14:19:37.356 MYSQL-Info Connecting database...  2017-11-28 14:19:38.197 MYSQL-Info Database connected!  -----------------Add-Table-Test-----------------  0 Tisch5 true  2017-11-28 14:19:38.198 MYSQL-Info Adding Table to MySQL-Database.  *Datenbankeintrag:*   |  |  |  | | --- | --- | --- | | **tableID** | **name** | **available** | | 17 | Tisch5 | 1 | |
| **Sonderfall** | 1. Verbindungsproblem mit dem MySQL-Server 2. Eines der Attribute wurde nicht gesetzt. 3. Es wurde eine Table-ID gesetzt. |
| Erwartetes Testergebnis | 1. Das Verbindungsproblem wird gemeldet.  2. Das fehlende Attribut (Name) wird in einer Meldung zurückgegeben.  3. Es wird rückgemeldet, dass die ID nicht vom Anwender, sondern von dem Datenbank-Server gesetzt wird. |
| Tatsächliches  Testergebnis | 1.  2017-11-28 14:23:37.119 MYSQL-Info Connecting database...  2017-11-28 14:23:39.710 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  2.  2017-11-28 14:25:30.090 MYSQL-Info Connecting database...  2017-11-28 14:25:31.071 MYSQL-Info Database connected!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Der Tisch ist unvollständig! Die folgenden Parameter fehlen: Tisch  -----------------Add-Table-Test-----------------  0 true  2017-11-28 14:25:31.072 MYSQL-Info Adding Table to MySQL-Database.  2017-11-28 14:25:31.072 MYSQL-ERROR Table missing.  2017-11-28 14:25:31.072 MYSQL-ERROR Table was not added to the database!  3.  2017-11-28 14:27:04.905 MYSQL-Info Connecting database...  2017-11-28 14:27:05.617 MYSQL-Info Database connected!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Es darf keine ID übergeben werden. Die ID wird vom Datenbank-Server gewählt!  -----------------Add-Table-Test-----------------  5 Tisch10 true  2017-11-28 14:27:05.618 MYSQL-Info Adding Table to MySQL-Database.  2017-11-28 14:27:05.618 MYSQL-ERROR ID may not be set by the user.  2017-11-28 14:27:05.618 MYSQL-ERROR Table was not added to the Database! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Wareneingang hinzufügen

|  |  |
| --- | --- |
| Anwendungsfall | Hinzufügen eines neuen Wareneingangs zu Datenbank. |
| Verwendete Methode | void addItemdelivery(Itemdelivery itemdelivery) Fügt der Datenbank einen neuen Wareneingang hinzu |
| **Normalablauf** | Es wird ein neuer gültiger Wareneingang der Datenbank-Tabelle Itemdelivery hinzugefügt. |
| Erwartetes Testergebnis | Erfolgreiches Erstellen eines neuen Wareneingags-Eintrags in der Datenbank. Validierung über einen Abgleich der Datenbank nach einem neu erzeugten Wareneingang mit folgenden Daten:  **int** itemID = 5; **int** quantity = 20; |
| Tatsächliches  Testergebnis | *Konsolenausgabe:*  2017-11-28 14:34:26.274 MYSQL-Info Connecting database...  2017-11-28 14:34:26.995 MYSQL-Info Database connected!  -----------------Add-Itemdelivery-Test-----------------  0 5 20  2017-11-28 14:34:26.996 MYSQL-Info Adding Itemdelivery to MySQL-Database.  *Datenbankeintrag:*   |  |  |  | | --- | --- | --- | | **itemdeliveryID** | **itemID** | **quantity** | | 48 | 5 | 20 | |
| **Sonderfall** | 1. Verbindungsproblem mit dem MySQL-Server 2. Eines der Attribute wurde nicht gesetzt. 3. Die gesetzte Item-ID existiert nicht in der Datenbank. 4. Es wurde eine Table-ID gesetzt. |
| Erwartetes Testergebnis | 1. Das Verbindungsproblem wird gemeldet.  2. Das fehlende Attribut (Item-ID, Anzahl) wird in einer Meldung zurückgegeben.  3. Es wird rückgemeldet, dass die Item-ID in der Datenbank nicht existiert.  4. Es wird rückgemeldet, dass die ID nicht vom Anwender, sondern von dem Datenbank-Server gesetzt wird. |
| Tatsächliches  Testergebnis | 1.  2017-11-28 14:38:43.712 MYSQL-Info Connecting database...  2017-11-28 14:38:46.334 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  2.  2017-11-28 14:39:31.260 MYSQL-Info Connecting database...  2017-11-28 14:39:32.024 MYSQL-Info Database connected!  -----------------Add-Itemdelivery-Test-----------------  0 0 0  2017-11-28 14:39:32.025 MYSQL-Info Adding Itemdelivery to MySQL-Database.  2017-11-28 14:39:32.135 MYSQL-ERROR Item-ID missing  2017-11-28 14:39:32.135 MYSQL-ERROR Quantity missing.  2017-11-28 14:39:32.135 MYSQL-ERROR Itemdelivery was not added to the database!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Der Wareneingang ist unvollständig! Die folgenden Parameter fehlen: Artikel Anzahl  3.  2017-11-28 14:41:52.971 MYSQL-Info Connecting database...  2017-11-28 14:41:54.023 MYSQL-Info Database connected!  -----------------Add-Itemdelivery-Test-----------------  0 7 50  2017-11-28 14:41:54.024 MYSQL-Info Adding Itemdelivery to MySQL-Database.  2017-11-28 14:41:54.109 MYSQL-ERROR The Item-IDs does not exist in the Database!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Die angegebene Artikel-ID existiert nicht in der Datenbank.  2017-11-28 14:41:54.109 MYSQL-ERROR Itemdelivery was not added to the Database!  4.  2017-11-28 14:43:46.465 MYSQL-Info Connecting database...  2017-11-28 14:43:47.428 MYSQL-Info Database connected!  -----------------Add-Itemdelivery-Test-----------------  5 5 50  2017-11-28 14:43:47.430 MYSQL-Info Adding Itemdelivery to MySQL-Database.  2017-11-28 14:43:47.506 MYSQL-ERROR ID may not be set by the user.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Es darf keine ID übergeben werden. Die ID wird vom Datenbank-Server gewählt!  2017-11-28 14:43:47.506 MYSQL-ERROR Itemdelivery was not added to the Database! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Aktualisieren von Datenbankinhalten

##### Artikel aktualisieren

|  |  |
| --- | --- |
| Anwendungsfall | Bearbeiten der Daten eines Artikels. |
| Verwendete Methode | void updateItem(int itemID, Item item) Aktualisiert die Daten eines Artikels |
| **Normalablauf** | Der richtige Artikel wird mit den Daten des neuen Items aktualisiert. |
| Erwartetes Testergebnis | Der Artikel mit der ID 16 wird aktualisiert.   |  |  |  |  | | --- | --- | --- | --- | | **itemID** | **name** | **retailprice** | **available** | | 16 | Kuchen | 2.99 | 1 |   Die folgenden Daten werden zum Aktualisieren übergeben:  **int** ID = 16; String name = **"Kirschkuchen"**; **double** retailprice = 3.49; **boolean** available = **false**; |
| Tatsächliches  Testergebnis | Konsolenausgabe:  2017-11-28 16:09:28.476 MYSQL-Info Connecting database...  2017-11-28 16:09:29.222 MYSQL-Info Database connected!  -----------------Update-Item-Test-----------------  16 Kirschkuchen 3.49 0 false  2017-11-28 16:09:29.223 MYSQL-Info Updating Item with ID 16.  Datenbankeintrag:   |  |  |  |  | | --- | --- | --- | --- | | **itemID** | **name** | **retailprice** | **available** | | 16 | Kirschkuchen | 3.49 | 0 | |
| **Sonderfall** | 1. Es besteht keine Verbindung zur Datenbank. 2. Die zu aktualisierende ID existiert nicht in der Datenbank. 3. Die übergebenden Daten sind nicht vollständig. 4. Es wurde keine zu aktualisierende ID übergeben. |
| Erwartetes Testergebnis | 1. Verbindungsfehlermeldung wird ausgegeben. 2. Eine nichtexistierende ID 14 wird gemeldet. 3. Die fehlenden Attribute werden gemeldet. 4. Eine fehlende ID wird gemeldet. |
| Tatsächliches  Testergebnis | *1.*  2017-11-28 16:13:03.562 MYSQL-Info Connecting database...  2017-11-28 16:13:06.420 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-28 16:13:59.112 MYSQL-Info Connecting database...  2017-11-28 16:13:59.924 MYSQL-Info Database connected!  -----------------Update-Item-Test-----------------  14 Kirschkuchen 3.49 0 false  2017-11-28 16:13:59.926 MYSQL-Info Updating Item with ID 14.  2017-11-28 16:14:00.021 MYSQL-ERROR Item with ID 14 does not exist in the database!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Artikel mit der ID 14 existiert nicht in der Datenbank!  *3.*  2017-11-28 16:16:08.972 MYSQL-Info Connecting database...  2017-11-28 16:16:09.796 MYSQL-Info Database connected!  -----------------Update-Item-Test-----------------  16 0.0 0 false  2017-11-28 16:16:09.798 MYSQL-Info Updating Item with ID 16.  2017-11-28 16:16:09.889 MYSQL-ERROR Name is missing.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Der Artikel ist unvollständig! Folgende Attribute fehlen: Name  2017-11-28 16:16:09.889 MYSQL-ERROR Item was not added to the Database!  *4.*  2017-11-28 16:17:43.228 MYSQL-Info Connecting database...  2017-11-28 16:17:44.036 MYSQL-Info Database connected!  -----------------Update-Item-Test-----------------  0 Kirschkuchen 3.49 0 false  2017-11-28 16:17:44.037 MYSQL-Info Updating Item with ID 0.  2017-11-28 16:17:44.119 MYSQL-ERROR Item-ID may not be null.  Exception in thread "main" java.lang.NullPointerException: No Item-ID given. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Tisch aktualisieren

|  |  |
| --- | --- |
| Anwendungsfall | Bearbeiten der Daten eines Tisches. |
| Verwendete Methode | void updateTable(int tableID, Table table) Aktualisiert die Daten eines Tisches. |
| **Normalablauf** | Der richtige Tisch wird mit den Daten des neuen Table aktualisiert. |
| Erwartetes Testergebnis | Der Tisch mit der ID 17 wird aktualisiert:   |  |  |  | | --- | --- | --- | | **tableID** | **name** | **available** | | 17 | Tisch5 | 1 |   Die folgenden Daten werden zum Aktualisieren übergeben:  **int** ID = 17; String name = **"Tisch10"**; **boolean** available = **false**; |
| Tatsächliches  Testergebnis | Konsolenausgabe*:*  2017-11-28 16:22:02.501 MYSQL-Info Connecting database...  2017-11-28 16:22:03.488 MYSQL-Info Database connected!  -----------------Update-Table-Test-----------------  17 Tisch10 false  2017-11-28 16:22:03.489 MYSQL-Info Updating Table with ID 17.  2017-11-28 16:22:03.489 MYSQL-Info Getting Tables from MySQL-Database.  Datenbankeintrag:   |  |  |  | | --- | --- | --- | | **tableID** | **name** | **available** | | 17 | Tisch10 | 0 | |
| **Sonderfall** | 1. Es besteht keine Verbindung zur Datenbank. 2. Die zu aktualisierende ID existiert nicht in der Datenbank. 3. Die übergebenden Daten sind nicht vollständig. |
| Erwartetes Testergebnis | 1. Verbindungsfehlermeldung wird ausgegeben. 2. Eine nichtexistierende ID 16 wird gemeldet. 3. Die fehlenden Attribute werden gemeldet. |
| Tatsächliches  Testergebnis | *1.*  2017-11-28 16:24:24.418 MYSQL-Info Connecting database...  2017-11-28 16:24:27.133 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-28 16:25:22.870 MYSQL-Info Connecting database...  2017-11-28 16:25:23.770 MYSQL-Info Database connected!  -----------------Update-Table-Test-----------------  16 Tisch10 false  2017-11-28 16:25:23.771 MYSQL-Info Updating Table with ID 16.  2017-11-28 16:25:23.803 MYSQL-ERROR Table with ID 16 does not exist in the database!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Tisch mit der ID 16 existiert nicht in der Datenbank!  *3.*  2017-11-28 16:26:41.177 MYSQL-Info Connecting database...  2017-11-28 16:26:41.912 MYSQL-Info Database connected!  -----------------Update-Table-Test-----------------  17 false  2017-11-28 16:26:41.913 MYSQL-Info Updating Table with ID 17.  2017-11-28 16:26:41.940 MYSQL-ERROR Table missing.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Der Tisch ist unvollständig! Die folgenden Parameter fehlen: Tisch  2017-11-28 16:26:41.940 MYSQL-ERROR Table was not added to the database! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Bestellung aktualisieren

|  |  |
| --- | --- |
| Anwendungsfall | Bearbeiten der Daten einer Bestellung. |
| Verwendete Methode | void updateOrder(int orderID, Order order) Aktualisiert die Daten einer Bestellung. |
| **Normalablauf** | Die richtige Bestellung wird mit den Daten der neuen Order aktualisiert. |
| Erwartetes Testergebnis | Die Bestellung mit der ID 45 wird aktualisiert:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **orderID** | **itemIDs** | **price** | **date** | **tableID** | **paid** | | 45 | 16;5;4; | 12.79 | 2017-11-22 22:18:14 | 1 | 0 |   Die folgenden Daten werden zum Aktualisieren übergeben:  **int** ID = 45; String itemIDs = **"16;5;4;25"**; **double** price = 13.29; **int** tableID = 5; **boolean** paid = **true**;  Wenn paid = false gesetzt ist, wird nur ein Küchenbeleg mit den hinzugefügten Artikeln ausgedruckt. Wenn paid = true ist, wird zusätzlich ein vollständiger Kundenbeleg ausgedruckt. |
| Tatsächliches  Testergebnis | Konsolenausgabe:  2017-11-28 16:47:08.875 MYSQL-Info Connecting database...  2017-11-28 16:47:09.744 MYSQL-Info Database connected!  -----------------Update-Order-Test-----------------  45 16;5;4;25 13.29 28.11.2017 16:47:09 5 true  2017-11-28 16:47:09.746 MYSQL-Info Updating Order with ID 45.  Datenbankeintrag:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **orderID** | **itemIDs** | **price** | **date** | **tableID** | **paid** | | 45 | 16;5;4;25 | 13.29 | 2017-11-28 16:47:10 | 5 | 1 |   Die Belege wurden beide korrekt ausgedruckt:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateOrderTest.jpgC:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateOrderTest.jpg |
| **Sonderfall** | 1. Es besteht keine Verbindung zur Datenbank. 2. Die zu aktualisierende ID existiert nicht in der Datenbank. 3. Die übergebenden Daten sind nicht vollständig. 4. Eine der Item-IDs existiert nicht in der Datenbank. 5. Die Table-ID existiert nicht in der Datenbank. |
| Erwartetes Testergebnis | 1. Verbindungsfehlermeldung wird ausgegeben. 2. Eine nichtexistierende ID 2 wird gemeldet. 3. Die fehlenden Attribute werden gemeldet. 4. Es wird eine Fehlermeldung über eine nichtexistierende Item-ID ausgegeben. 5. Es wird eine Fehlermeldung über eine nichtexistierende Table-ID ausgegeben. |
| Tatsächliches  Testergebnis | *1.*  2017-11-28 16:52:06.673 MYSQL-Info Connecting database...  2017-11-28 16:52:09.581 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-28 16:57:23.970 MYSQL-Info Connecting database...  2017-11-28 16:57:24.707 MYSQL-Info Database connected!  -----------------Update-Order-Test-----------------  2 16;5;4;25 13.29 28.11.2017 16:57:24 5 true  2017-11-28 16:57:24.709 MYSQL-Info Updating Order with ID 2.  2017-11-28 16:57:24.799 MYSQL-Info Getting Tables from MySQL-Database.  2017-11-28 16:57:24.804 MYSQL-ERROR Order with ID 2 does not exist in the database!  dhbw.sa.kassensystem\_rest.exceptions.DataException: Bestellung mit der ID 2 existiert nicht in der Datenbank!  *3.*  2017-11-28 16:58:50.037 MYSQL-Info Connecting database...  2017-11-28 16:58:50.795 MYSQL-Info Database connected!  -----------------Update-Order-Test-----------------  45 13.29 28.11.2017 16:58:50 0 true  2017-11-28 16:58:50.797 MYSQL-Info Updating Order with ID 45.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Die Bestellung ist unvollständig! Die folgenden Parameter fehlen: Artikel Tisch  2017-11-28 16:58:50.878 MYSQL-ERROR Item-IDs missing.  2017-11-28 16:58:50.878 MYSQL-ERROR Table-ID missing.  2017-11-28 16:58:50.878 MYSQL-ERROR Order was not added to the database!  *4.*  2017-11-28 17:00:18.817 MYSQL-Info Connecting database...  2017-11-28 17:00:19.541 MYSQL-Info Database connected!  -----------------Update-Order-Test-----------------  45 16;5;4;25;7 13.29 28.11.2017 17:00:19 5 true  2017-11-28 17:00:19.543 MYSQL-Info Updating Order with ID 45.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Eine oder mehrere angegebene Artikel-IDs existieren nicht in der Datenbank.  2017-11-28 17:00:19.645 MYSQL-ERROR One or multiple Item-IDs do not exist in the database!  2017-11-28 17:00:19.645 MYSQL-ERROR Order was not added to the Database!  *5.*  2017-11-28 17:01:49.394 MYSQL-Info Connecting database...  2017-11-28 17:01:50.265 MYSQL-Info Database connected!  -----------------Update-Order-Test-----------------  45 16;5;4;25 13.29 28.11.2017 17:01:50 18 true  2017-11-28 17:01:50.267 MYSQL-Info Updating Order with ID 45.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Die angegebene Table-ID existiert nicht in der Datenbank!  2017-11-28 17:01:50.384 MYSQL-ERROR The Table-ID does not exist in the database!  2017-11-28 17:01:50.384 MYSQL-ERROR Order was not added to the database! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Löschen von Datenbankinhalten

Das Löschen der Datenbankinhalte Artikel und Tisch funktioniert über ein Aktualisieren des Datenbankeintrags mit Setzten der Verfügbarkeit auf falsch. Dies wurde mit dem Testen des Aktualisierens schon abgedeckt. Daher muss nur das Löschen von Bestellungen und Wareneingängen getestet werden.

##### Bestellung löschen

|  |  |
| --- | --- |
| Anwendungsfall | Löschen einer Bestellung |
| Verwendete Methode | void deleteOrder(int orderID) Loescht eine Bestellung aus der Datenbank |
| **Normalablauf** | Die Bestellung wird aus der Datenbank gelöscht. |
| Erwartetes Testergebnis | Die folgende Bestellung existiert nicht mehr in der Datenbank:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **orderID** | **itemIDs** | **price** | **date** | **tableID** | **paid** | | 52 | 13;16;18; | 18.49 | 2017-11-23 15:50:20 | 1 | 0 | |
| Tatsächliches  Testergebnis | *Konsolenausgabe:*  2017-11-28 17:33:35.528 MYSQL-Info Connecting database...  2017-11-28 17:33:36.341 MYSQL-Info Database connected!  -----------------Delete-Order-Test-----------------  2017-11-28 17:33:36.342 MYSQL-Info Deleting Order with ID 52.  *Datenbankeintrag:*  *Wurde gelöscht.* |
| **Sonderfall** | 1. Es besteht keine Verbindung zu Datenbank. 2. Es existiert keine Bestellung mit der angegebenen ID. |
| Erwartetes Testergebnis | 1. Ein Verbindungsfehler wird gemeldet 2. Es wird gemeldet, dass die ID nicht in der Datenbank existiert. |
| Tatsächliches  Testergebnis | *1.*  2017-11-28 17:35:02.744 MYSQL-Info Connecting database...  2017-11-28 17:35:05.441 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-28 17:35:37.840 MYSQL-Info Connecting database...  2017-11-28 17:35:38.855 MYSQL-Info Database connected!  -----------------Delete-Order-Test-----------------  2017-11-28 17:35:38.855 MYSQL-Info Deleting Order with ID 59.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Bestellung mit der ID 59 existiert nicht in der Datenbank! Es konnte nichts gelöscht werden  2017-11-28 17:35:38.906 MYSQL-ERROR Order with ID 59 does not exist in the database! Nothing was deleted. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Wareneingang löschen

|  |  |
| --- | --- |
| Anwendungsfall | Löschen eines Wareneingangs |
| Verwendete Methode | void deleteItemdelivery(int itemdeliveryID) Löscht einen Wareneingang aus der Datenbank |
| **Normalablauf** | Der Wareneingang wird aus der Datenbank gelöscht. |
| Erwartetes Testergebnis | Der folgende Wareneingang existiert nicht mehr in der Datenbank:   |  |  |  | | --- | --- | --- | | **itemdeliveryID** | **itemID** | **quantity** | | 23 | 26 | 500 | |
| Tatsächliches  Testergebnis | *Konsolenausgabe:*  2017-11-28 17:40:31.660 MYSQL-Info Connecting database...  2017-11-28 17:40:32.503 MYSQL-Info Database connected!  -----------------Delete-Itemdelivery-Test-----------------  2017-11-28 17:40:32.503 MYSQL-Info Deleting Itemdelivery with ID 23.  *Datenbankeintrag:*  *Wurde gelöscht.* |
| **Sonderfall** | 1. Es besteht keine Verbindung zu Datenbank. 2. Es existiert keine Bestellung mit der angegebenen ID. |
| Erwartetes Testergebnis | 1. Ein Verbindungsfehler wird gemeldet 2. Es wird gemeldet, dass die ID nicht in der Datenbank existiert. |
| Tatsächliches  Testergebnis | *1.*  2017-11-28 17:41:09.252 MYSQL-Info Connecting database...  2017-11-28 17:41:11.894 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-28 17:39:15.810 MYSQL-Info Connecting database...  2017-11-28 17:39:16.613 MYSQL-Info Database connected!  -----------------Delete-Itemdelivery-Test-----------------  2017-11-28 17:39:16.613 MYSQL-Info Deleting Itemdelivery with ID 59.  2017-11-28 17:39:16.647 MYSQL-ERROR Itemdelivery with ID 59 does not exist in the database! Nothing was deleted.  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Wareneingang mit der ID 59 existiert nicht in der Datenbank! Es konnte nichts gelöscht werden. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Drucken einer Bestellung

|  |  |
| --- | --- |
| Anwendungsfall | Drucken einer Bestellung im Kassensystem-Manager. |
| Verwendete Methode | void printOrderById(int orderID) Ausdrucken einer Bestellung in Abhängigkeit von einer ID. |
| **Normalablauf** | Die Bestellung mit der angegebenen ID wird ausgedruckt. |
| Erwartetes Testergebnis | Ein Ausdruck der Bestellung mit den folgenden Daten: |
| Tatsächliches  Testergebnis | Die Bestellung wurde ausgedruckt:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\printOrderByIdTest.jpg |
| **Sonderfall** | 1. Es besteht keine Verbindung zur Datenbank. 2. Es existiert in der Datenbank keine Bestellung mit der angegebenen ID. 3. Der Drucker ist nicht erreichbar. |
| Erwartetes Testergebnis | 1. Der Verbindungsfehler wird gemeldet. 2. Es wird gemeldet, dass die ID nicht in der Datenbank existiert. 3. Die Bestellung wird mit dem nächsten Anschalten des Druckers ausgedruckt. |
| Tatsächliches  Testergebnis | *1.*  2017-11-28 17:41:09.252 MYSQL-Info Connecting database...  2017-11-28 17:41:11.894 MYSQL-ERROR Verbindung zur Datenbank fehlgeschlagen!  java.lang.ExceptionInInitializerError  Caused by: java.lang.IllegalStateException: Verbindung zur Datenbank fehlgeschlagen!  *2.*  2017-11-28 17:57:54.594 MYSQL-Info Connecting database...  2017-11-28 17:57:55.445 MYSQL-Info Database connected!  Exception in thread "main" dhbw.sa.kassensystem\_rest.exceptions.DataException: Bestellung mit der ID 60 existiert nicht in der Datenbank! Es kann nichts gedruckt werden.  2017-11-28 17:57:55.533 MYSQL-ERROR Order with ID 60 does not exist in the database!  *3.*  Nach einem erneuten Anschalten des Druckers wird die Bestellung ausgedruckt:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\printOrderByIdTest.jpg |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

### Server-Modul

#### Abrufen von Datenbankinhalten

Im Folgenden werden die Abrufsfunktionen des Rest-Api-Controllers getestet. Dafür wird die entsprechende URL in einen Browser angesprochen und die Rückmeldung überprüft. Für diesen Test muss auf demselben Rechner der Controller laufen und mit der URL „localhost:8080“ erreichbar sein.

##### Artikel-Daten

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen von Artikel-Daten aus der Datenbank. |
| Verwendete Methode | java.util.ArrayList<Item> getAllItems() Durch das Ansprechen des Pfades ".../api/items" können die Artikel der Datenbank abgefragt werden. |
| **Normalablauf** | Alle Artikel werden als Item-Objekte aus der MySQL-Datenbank abgefragt und über Http als JSON-Datei übertragen |
| Erwartetes Testergebnis | Rückgabe eines JSON-Textes mit folgenden Inhalten:   |  |  |  |  | | --- | --- | --- | --- | | **itemID** | **name** | **retailprice** | **available** | | 1 | Pizza | 7 | 1 | | 2 | Pommes | 3.5 | 1 | | 3 | Nudeln | 11 | 1 | | 4 | Lasagne | 7.5 | 1 | | 5 | Kaffee | 2.3 | 1 | | 6 | Cola | 1.5 | 0 | | 9 | Bier | 3.3 | 0 | | 10 | Brot | 5 | 0 | |
| Tatsächliches  Testergebnis | [{"**itemID**":1,"name":"Pizza","retailprice":7.0,"quantity":4,"available":true},  {"**itemID**":2,"name":"Pommes","retailprice":3.5,"quantity":7,"available":true},  {"**itemID**":3,"name":"Nudeln","retailprice":11.0,"quantity":4,"available":true},  {"**itemID**":4,"name":"Lasagne","retailprice":7.5,"quantity":6,"available":true},  {"**itemID**":5,"name":"Kaffee","retailprice":2.3,"quantity":66,"available":true},  …] |
| **Sonderfall** | Keine Verbindung zur MySQL-Datenbank aufbaubar. |
| Erwartetes Testergebnis | Http-Nachricht über den Fehler. |
| Tatsächliches  Testergebnis | Whitelabel Error Page  This application has no explicit mapping for /error, so you are seeing this as a fallback.  Tue Nov 28 19:06:04 CET 2017  There was an unexpected error (type=Not Found, status=404).  Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Bestellungs-Daten

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen von allen Bestellungen aus der Datenbank. |
| Verwendete Methode | java.util.ArrayList<Order> getAllOrders() Durch das ansprechen des Pfades ".../api/orders" können die Bestellungen der Datenbank abgefragt werden. |
| **Normalablauf** | Alle Bestellungen werden als Order-Objekte aus der MySQL-Datenbank abgefragt und über Http als JSON-Datei übertragen. |
| Erwartetes Testergebnis | Rückgabe eines JSON-Textes mit folgenden Inhalten:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **orderID** | **itemIDs** | **price** | **date** | **tableID** | **paid** | | 1 | 2;1; | 2.3 | 2017-10-19 15:40:25 | 5 | 1 | | 3 | 1;3; | 18 | 2017-10-23 10:44:48 | 6 | 1 | | 4 | 1;2;3;4;5; | 31.3 | 2017-11-05 15:38:42 | 3 | 0 | | 5 | 2;3; | 9.8 | 2017-11-08 23:31:33 | 5 | 1 | | 12 | 4;5; | 9.8 | 2017-11-14 21:32:22 | 5 | 1 | | 17 | 3;5; | 33.99 | 2017-11-12 14:42:34 | 10 | 1 | |
| Tatsächliches  Testergebnis | [{"**orderID**":1,"itemIDs":"2;1;","tableID":5,"price":2.3,"date":  1508420425000,"paid":true,"table":5,"items":"2;1;"},  {"**orderID**":3,"itemIDs":"1;3;","tableID":6,"price":18.0,"date":  1508748288000,"paid":true,"table":6,"items":"1;3;"},  …] |
| **Sonderfall** | Keine Verbindung zur MySQL-Datenbank aufbaubar. |
| Erwartetes Testergebnis | Http-Nachricht über den Fehler. |
| Tatsächliches  Testergebnis | Whitelabel Error Page  This application has no explicit mapping for /error, so you are seeing this as a fallback.  Tue Nov 28 19:24:36 CET 2017  There was an unexpected error (type=Not Found, status=404).  Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Tisch-Daten

|  |  |
| --- | --- |
| Anwendungsfall | Abfragen von Tisch-Daten aus der Datenbank. |
| Verwendete Methode | java.util.ArrayList<Table> getAllTables() Durch das Ansprechen des Pfades "…/api/tables" können die Tische der Datenbank abgefragt werden. |
| **Normalablauf** | Alle Tische werden als Table-Objekte aus der MySQL-Datenbank abgefragt und über Http als JSON-Datei übertragen. |
| Erwartetes Testergebnis | Rückgabe eines JSON-Textes mit folgenden Inhalten:   |  |  |  | | --- | --- | --- | | **tableID** | **name** | **available** | | 1 | A1 | 1 | | 2 | A2 | 1 | | 3 | A3 | 1 | | 4 | B1 | 1 | | 5 | B2 | 1 | | 6 | B3 | 1 | | 7 | C1 | 1 | | 8 | C2 | 1 | |
| Tatsächliches  Testergebnis | [{"tableID":1,"name":"A1","available":true},{"tableID":2,"name":"A2","available":true},{"tableID":3,"name":"A3","available":true},{"tableID":4,"name":"B1","available":true},{"tableID":5,"name":"B2","available":true}, …] |
| **Sonderfall** | Keine Verbindung zur MySQL-Datenbank aufbaubar. |
| Erwartetes Testergebnis | Http-Nachricht über den Fehler. |
| Tatsächliches  Testergebnis | Whitelabel Error Page  This application has no explicit mapping for /error, so you are seeing this as a fallback.  Tue Nov 28 19:27:43 CET 2017  There was an unexpected error (type=Not Found, status=404).  Verbindung zur Datenbank fehlgeschlagen! |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Aktualisieren und Hinzufügen einer Bestellung

Das Hinzufügen und Aktualisieren einer Bestellung kann nicht ohne weiteres ohne die Android-App getestet werden. Der Test dieser Funktionen wird mit den Tests der Android-App abgedeckt.

### Benutzeranwendung

#### Abrufen von Datenbankinhalten

##### 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 nicht bezahlten Bestellungen. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetOrder.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Abrufen der Artikel

|  |  |
| --- | --- |
| Anwendungsfall | 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 | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetItems.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Abrufen der Tische

|  |  |
| --- | --- |
| Anwendungsfall | 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 | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetTables.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Abrufen der Wareneingänge

|  |  |
| --- | --- |
| Anwendungsfall | 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 | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiGetItemdeliveries.png* |
| **Sonderfall** | Es besteht keine Verbindung zum MySQL-Server. |
| Erwartetes Testergebnis | Es wird eine Fehlermeldung angezeigt. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiConnectionError.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Hinzufügen von Datenbankinhalten

##### Hinzufügen von neuen Artikeln

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddItemData.png |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddedItem.png* |
| **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.*  *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemName.png*  *2.*  *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemQuantity.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Hinzufügen von neuen Tischen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddTableData.png |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddedTable.png* |
| **Sonderfall** | Es wurde kein Name angegeben. |
| Erwartetes Testergebnis | Fehlermeldung mit der Meldung eines fehlenden Namens. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrTableName.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddItemdeliveryData.png |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiAddedItemdelivery.png* |
| **Sonderfall** | Es wurde keine Anzahl eingegeben. |
| Erwartetes Testergebnis | Fehlermeldung mit der Meldung einer fehlenden Anzahl. |
| Tatsächliches  Testergebnis | C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemdeliveryQuantity.png |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Bearbeiten von Datenbankinhalten

##### Bearbeiten von Artikeln

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateItemOld.png  Dieser Artikel soll mit den folgenden Daten aktualisiert werden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateItemNew.png |
| Tatsächliches  Testergebnis | In den Daten ist nun der folgende Artikel zu finden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateItemResult.png |
| **Sonderfall** | Es wird kein Name übergeben. |
| Erwartetes Testergebnis | Eine Fehlermeldung, die einen fehlenden Namen anmerkt. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrItemName.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

##### Bearbeiten von Tischen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateTableOld.png  Dieser Tisch soll mit den folgenden Daten bearbeitet werden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateTableNew.png |
| Tatsächliches  Testergebnis | In der tabellarischen Übersicht ist nun der folgende Tisch zu finden:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\updateTableResult.png |
| **Sonderfall** | Es wurde kein Name angegeben. |
| Erwartetes Testergebnis | Eine Fehlermeldung über den fehlenden Namen. |
| Tatsächliches  Testergebnis | *C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\guiErrTableName.png* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

#### Löschen von Datenbankinhalten

##### Löschen von Bestellungen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteOrder.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

##### Löschen von Artikeln

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItem.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

##### Löschen von Tischen

|  |  |
| --- | --- |
| Anwendungsfall | 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteTable.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

##### 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:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItemdeliveryOld.png  In der tabellarischen Übersicht der Artikel wird die Anzahl des entsprechenden Artikels reduziert:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItemdeliveryQuantityOld.png |
| Tatsächliches  Testergebnis | Der Eintrag existiert nicht mehr in der Anwendung. Die Anzahl des Artikels in der Artikel-Übersicht wurde aktualisiert:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deleteItemdeliveryQuantityNew.png |
| **Sonderfall** | keiner |
| Erwartetes Testergebnis | -- |
| Tatsächliches  Testergebnis | *--* |
| Test  bestanden | Normalablauf: Ja  Sonderfall: -- |

#### Ausdrucken einer Bestellung

|  |  |
| --- | --- |
| Anwendungsfall | 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 Bon-Drucker wird der Kundenbeleg ausgedruckt. |
| Erwartetes Testergebnis | Der Bon-Drucker druckt einen Kundenbeleg mit den folgenden Bestellungsdaten:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\printOrder.png |
| Tatsächliches  Testergebnis | Ein Ausdruck wurde ausgegeben:  C:\Users\marvi\AppData\Local\Microsoft\Windows\INetCache\Content.Word\img108.jpg |
| **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 |