# Testdokumentation

## Test der Android-Anwendung

Im Folgenden sind die Tests der Android-Anwendung dokumentiert.

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

**Mitarbeiterlogin:**

|  |  |
| --- | --- |
| Anwendungsfall | Die Bedienung kann sich im System mit ihrem Login-Namen und Login-Passwort authentifizieren. Diese Daten müssen sich zuvor in der Datenbank befinden.  (AW 1b) |
| Verwendete Methode | doInBackground()  onPostExecute() (Klasse: “LoginCheck”)  showToast (String text) |
| **Normalablauf** | Wird „Login“ im Navigation-Drawer ausgewählt, wird ein Bildschirm zum Anmelden im System angezeigt. Um sich anzumelden müssen der Login-Name und das Login-Passwort eingegeben werden.  Stimmen die Daten mit den hinterlegten Daten in der Datenbank überein, dann ist der Mitarbeiter erfolgreich angemeldet.  Der Startbildschirm wird angezeigt. (Bildschirm: Bestellung aufgeben).  (Nach dem ersten Anmelden bleiben die Daten gespeichert). |
| Erwartetes Testergebnis | Die Bedienung hat sich erfolgreich im System authentifiziert und kann auf alle Funktionen zugreifen.  Datenbank-Eintrag der Login-Daten für die Bedienungen:    Eingabe in der Android-Anwendung: |
| Tatsächliches  Testergebnis | Die Bedienung hat sich erfolgreich im System authentifiziert und kann auf alle Funktionen zugreifen. Sie wird auf den Startbildschirm weitergeleitet. |
| **Sonderfall 1:** | Die Bedienung hat falsche Login-Daten eingegeben. Das heißt, das Login-Passwort und/oder der Login-Name sind falsch eingegeben worden. |
| Erwartetes Testergebnis | Die Bedienung kann sich im System nicht authentifizieren und hat somit keinen Zugriff auf die restlichen Funktionen.  Für die Bedienung erscheint eine Fehlermeldung, dass die Login-Daten fehlerhaft sind. So kann sie erneut versuchen, die korrekten Login-Daten anzugeben.  Um den Login-Vorgang abzubrechen, muss im Navigations-Drawer der gewünschte Bildschirm ausgewählt werden.  Mit der „Rückgängig“-Taste kommt man ebenfalls auf den Startbildschirm. (Bildschirm: Bestellung aufgeben).  Für diesen Test bleibt die Datenbank dieselbe wie im Normalfall dargestellt. |
| Tatsächliches  Testergebnis | Der Login-Name (wie hier im Beispiel) oder das Passwort ist falsch.  Folgender Bildschirm wird für die Bedienung sichtbar:    Die Fehlermeldung wird angezeigt. Die Bedienung kann Ihre Angaben korrigieren. |
| **Sonderfall 2:** | Es besteht keine Verbindung zur Datenbank.  Die Bedienung kann sich somit nicht im System authentifizieren. |
| Erwartetes Testergebnis | Die Bedienung kann sich im System nicht authentifizieren und hat somit keinen Zugriff auf die restlichen Funktionen.  Für die Bedienung erscheint eine Fehlermeldung, dass die Verbindung zur Datenbank fehlgeschlagen ist. |
| Tatsächliches  Testergebnis | Folgender Bildschirm wird für die Bedienung sichtbar:    Nachdem die Verbindung zur Datenbank wiederhergestellt worden ist, kann sich die Bedienung erneut anmelden. |
| Test  bestanden | Normalablauf: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

**Login-Passwort ändern:**

|  |  |
| --- | --- |
| Anwendungsfall | Nachdem der Gastronom die Login-Daten für eine neu eingestellte Bedienung angelegt hat, sollte das Passwort geändert werden. So kann sichergestellt werden, dass nur die Bedienung auf die Login-Daten Zugriff hat.  (AW 2b) |
| Verwendete Methode | doInBackground()  onPostExecute() (Klasse: “ChangeLoiginPassword”)  showToast (String text) |
| **Normalablauf** | Damit das Passwort geändert werden kann, muss zunächst im Navigation-Drawer „Login-Passwort bearbeiten“ ausgewählt werden. Danach wird ein Bildschirm angezeigt, in der die Bedienung die Passwort-Änderung vornehmen kann. Dafür muss der Login-Name, das alte Login-Passwort und das neue - gewünschte - Passwort eingegeben werden. (Das neue Passwort muss wiederholt werden, sodass es zu keinem unbewussten Schreibfehler innerhalb des Passworts kommen kann).  Nach erfolgreichem Bearbeiten des Passworts wird der Startbildschirm dargestellt. (Bildschirm: Bestellung aufgeben). |
| Erwartetes Testergebnis | Bei korrekter Eingabe der alten und neuen Login-Daten wird das neue Passwort in der Datenbank hinterlegt. Das alte Passwort wird überschrieben. |
| Tatsächliches  Testergebnis | Die Datenbank wird mit dem neuen Login-Passwort aktualisiert. Die Bedienung kann sich nur noch mit dem neuen Passwort anmelden.  Die Datenbank sieht wie folgt aus: |
| **Sonderfall 1** | Der Login-Name oder das Login-Passwort wurden von der Bedienung falsch eingegeben.  Somit kann das Login-Passwort nicht geändert werden. |
| Erwartetes Testergebnis | Das Passwort in der Datenbank wird nicht aktualisiert.  Der Bedienung wird eine Fehlermeldung angezeigt, dass die von ihr eingegebenen Login-Daten fehlerhaft sind.  Die Eingaben für die Passwörter werden gelöscht. So wird die Bedienung gezwungen, diese nochmals einzugeben.  Anschließend kann die Bedienung erneut versuchen, ihr Login-Passwort zu ändern. |
| Tatsächliches  Testergebnis | Die folgende Fehlermeldung wird der Bedienung angezeigt.    Die Datenbank wurde nicht aktualisiert. |
| **Sonderfall 2** | Die neuen Passwörter stimmen nicht überein. |
| Erwartetes Testergebnis | Das Passwort wird in der Datenbank nicht aktualisiert.  Der Bedienung wird in Rot eine Fehlermeldung angezeigt, dass die Passwörter nicht übereinstimmen. Die Änderung der Passwörter kann nicht erfolgen. |
| Tatsächliches  Testergebnis | Die Datenbank wird nicht aktualisiert:    Der Folgende Bildschirm wird dargestellt: |
| **Sonderfall 3** | Es besteht keine Verbindung zur Datenbank |
| Erwartetes Testergebnis | Es wird beim Betätigen des „PASSWORT ÄNDERN“-Buttons eine Fehlermeldung dargestellt.  Die Datenbank kann nicht aktualisiert werden.  Nach erneuter Verbindung mit der Datenbank kann das Passwort geändert werden. |
| Tatsächliches  Testergebnis | Die folgende Fehlermeldung wird der Bedienung dargestellt.    Die Datenbank wurde nicht aktualisiert. |
| Test  bestanden | Normalablauf: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja  Sonderfall 3: Ja |

**Artikel als produziert markieren:**

|  |  |
| --- | --- |
| Anwendungsfall | In der Datenbank soll vermerkt werden, dass ein Artikel von der Bedienung abgeholt wurde. Anschließend soll der Artikel zum jeweiligen Kunden gebracht werden.  Dadurch kann im Falle eines Fehlers identifiziert werden, welcher Artikel bereits an einen Kunden geliefert wurde.  (AW 3b) |
| Verwendete Methode | doInBackground()  onPostExecute() (Klasse: “UpdateOrder”)  showToast (String text) |
| **Normalablauf** | Damit der entsprechende Artikel „markiert“ werden kann muss im Navigation-Drawer „Bestellungsannahme“ ausgewählt werden. In dem dann angezeigten Bildschirm werden alle nicht produzierten Artikel aufgelistet. Dabei wird der Artikelname, -kommentar und der dazugehörige Tisch angezeigt.  Der von der Bedienung angenommene Artikel kann in dieser Darstellung ausgewählt werden.  Mit anschließendem Klicken auf den „BESTÄTIGEN“-Button werden die ausgewählten Artikel in der Datenbank als produziert hinterlegt. |
| Erwartetes Testergebnis | Die Artikel werden in der Datenbank als „produziert“ markiert.  Die Datenbank sieht wie folgt aus:    Im Test wird der Artikel mit der „orderedItemID“ *203 markiert.*  Im Beispiel der Artikel Cola an Tisch *A1* mit dem Kommentar *aus der Flasche*: |
| Tatsächliches  Testergebnis | Der Artikel mit der „orderedItemID“ *203* wurde als produziert hinterlegt.  Die Datenbank sieht wie folgt aus:    Die Daten wurden erfolgreich in der Datenbank eingetragen. |
| **Sonderfall** | Es besteht keine Verbindung zur Datenbank. Die Daten können somit nicht aktualisiert werden. |
| Erwartetes Testergebnis | Der Bedienung wird eine Fehlermeldung dargestellt, dass keine Verbindung zur Datenbank existiert.  Anschließend wird der Startbildschirm der Android-Anwendung dargestellt. Die Bedienung muss nach erfolgreichem Wiederverbinden mit dem Netzwerk die Artikel nochmals eintragen. |
| Tatsächliches  Testergebnis | Die Bedienung erhält folgende Fehlermeldung: |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

**Rechnung splitten:**

|  |  |
| --- | --- |
| Anwendungsfall | Möchte eine Gruppe von Kunden ihre Bestellung getrennt bezahlen, so kann der Preis einfach dargestellt werden. Die Bedienung kann für jeden Kunden eingeben welche Artikel bezahlt werden sollen. Anschließend kann sie an dem Bildschirm ablesen, welchen Betrag der jeweilige Kunde zu zahlen hat. Dies kann die Bedienung für jeden Kunden separat machen.  (AW 4b) |
| Verwendete Methode | doInBackground()  onPostExecute() (Klasse: “UpdateOrder”)  showToast (String text) |
| **Normalablauf** | Eine Bedienung kommt an einen Tisch, an denen verschiedene Personen bezahlen möchten.  Sie geht in die Bestellung und klickt dann auf den Button „BEZAHLEN“. Danach wird ihr ein neuer Bildschirm angezeigt. Sie kann nun auswählen, ob die Kunden gemeinsam bezahlen oder ob jeder Kunde für sich bezahlen möchte.  Wenn jeder Kunde einzeln zahlen möchte, markiert sie die Checkbox „Separat Bezahlen“. Nun können die Kunden die Artikel getrennt zahlen. |
| Erwartetes Testergebnis | Der Preis wird richtig berechnet und dargestellt.  Die Artikel die ausgewählt wurden, sind in der Datenbank als bezahlt hinterlegt.  Die Datenbankinhalte vor dem Test sehen wie folgt aus:    Die separat bezahlten Artikel:    Eine Cola und ein Burger werden für 7.0 € bezahlt. Der Rest der Rechnung bleibt offen. |
| Tatsächliches  Testergebnis | Die Datenbank wird aktualisiert. Die Artikel die bezahlt wurden werden auf „1“ gesetzt.  Der Preis wird in der Anwendung richtig berechnet und dargestellt.  Die Datenbankinhalte nach dem Test:    Die Artikel mit der „orderedItemID“ *207* und *209* wurden bezahlt. |
| **Sonderfall** | Die Rechnung wird auf einmal bezahlt. Die Kunden möchten ihre Rechnung nicht separat bezahlen. Der Vorgang verhält sich gleich wie im Normalfall. Jedoch darf die Checkbox „Separat Bezahlen“ nicht aktiviert werden. |
| Erwartetes Testergebnis | Alle Artikel werden in der Datenbank als bezahlt hinterlegt. Der Preis der Bestellung wird der Bedienung richtig dargestellt.    Der Rest der Bestellung aus dem Normalfall wird nun zusammen bezahlt. In der Android-Anwendung sieht das wie folgt aus: |
| Tatsächliches  Testergebnis | Die Datenbank wird aktualisiert. Die Artikel die bezahlt wurden werden auf „1“ gesetzt.  Der Preis wird in der Anwendung richtig berechnet und dargestellt.  Die Datenbank wird wie folgt dargestellt:    Alle Artikel mit „OrderID“ *95* wurden bezahlt. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

**Kundenbeleg anfordern**

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

|  |  |
| --- | --- |
| Tatsächliches  Testergebnis | Der Beleg sieht wie folgt aus: |
| **Sonderfall** | Die Kunden möchten getrennt bezahlen und möchten einen Kundenbeleg.  Das Vorgehen ist dasselbe wie im Normalfall beschrieben. |
| Erwartetes Testergebnis | Der gesamte Beleg einer Bestellung wird ausgedruckt. Dieser kann dann für jeden Kunden einzeln ausgedruckt werden.  Im Test sieht die Bestellung wie folgt aus: |

|  |  |
| --- | --- |
| Tatsächliches  Testergebnis | Der Beleg sieht wie folgt aus: |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

**Bestellten Artikel kommentieren:**

|  |  |
| --- | --- |
| Anwendungsfall | Der Kunde möchte bei einem Artikel eine bestimmte Art der Zubereitung oder hat einen sonstigen Wunsch, den der Koch beachten soll.  Dann kann dem bestellten Artikel eine Information mitgegeben werden. Diese Information wird dann ebenfalls in der Datenbank abgelegt. |
| Verwendete Methode | doInBackground()  onPostExecute() (Klasse: “UpdateOrder”)  showToast (String text) |
| **Normalablauf** | Ein Kunde äußert einen Wunsch zu einem Artikel. Dieser Wunsch kann dem Artikel angefügt werden. Dazu muss die Bedienung im „Bestellung aufgeben“ Verfahren auf den Artikel lange drücken. Es öffnet sich ein neues Fenster, in dem die Bedienung den geäußerten Wunsch anmerken kann.  Anschließend kann die Bestellung normal zu Ende geführt werden. |
| Erwartetes Testergebnis | Der Kommentar wird gespeichert und in der Datenbank hinterlegt. Der Küchenbeleg beinhaltet den Kommentar.  Der Bestellung wird folgender Kommentar angeheftet: |
| Tatsächliches  Testergebnis | Die Datenbank sieht nach der Bestellung wie folgt aus:    Der Küchenbeleg sieht wie folgt aus: |
| **Sonderfall** | Dem Artikel soll ein Wunsch hinzugefügt werden, nachdem dieser bereits bestellt wurde. |
| Erwartetes Testergebnis | In der Datenbank wird der Kommentar nicht mehr gespeichert und kann somit auch nicht berücksichtigt werden.  Der Kundenwunsch für den Artikel: |
| Tatsächliches  Testergebnis | Die Datenbank sieht wie folgt aus:    Die Datenbank wurde nicht aktualisiert. |
| Test  bestanden | Normalablauf: Ja  Sonderfall: Ja |

Die folgenden Tests müssen aufgrund der Optimierung die an dem System vorgenommen wurden noch einmal durchlaufen werden.   
Die Tests sind dieselben wie in der vorangegangenen Version. Sie wurden jedoch am neuen System durchgeführt.

**Einsehen der Tische**

|  |  |
| --- | --- |
| Anwendungsfall | Einsehen aller verfügbaren Tische in der Android-Anwendung (AW 15) |
| Verwendete Methode | doInBackground()  onPostExecute() (Klasse: “GetAllTables”) |
| **Normalablauf** | Wird „Bestellung aufgeben“ im Navigation-Drawer ausgewählt, sollen in einem Dropdown-Menü alle verfügbaren Tische der Datenbank dargestellt werden. |
| Erwartetes Testergebnis | In dem Dropdown-Menü werden die verfügbaren Tische der Datenbank angezeigt. Folgende Tische, die unter dem Register „available“ mit einer 1 markiert sind, werden angezeigt: |
| Tatsächliches  Testergebnis | Das Dropdow-Menü wurde wie folgt dargestellt: |
| **Sonderfall 1** | Es kann keine Verbindung zur Datenbank aufgebaut werden, beziehungsweise der Server wurde nicht gestartet. |
| Erwartetes Testergebnis | Das Dropdown-Menü bleibt leer und es wird eine Fehlermeldung angezeigt, dass eine Verbindung zum Server nicht möglich ist. |
| Tatsächliches  Testergebnis | Das Smartphone gibt folgende Fehlermeldung aus: |
| **Sonderfall 2** | Es kann eine Verbindung aufgebaut werden. In der Datenbank befinden sich jedoch keine Tische. |
| Erwartetes Testergebnis | Das Dropdown-Menü bleibt leer und es wird eine „Fehlermeldung“ angezeigt, dass keine Tische verfügbar sind. |
| Tatsächliches  Testergebnis | Das Smartphone gibt folgende Fehlermeldung aus: |
| Test  bestanden | Normalablauf: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

**Auswählen der Tische**

|  |  |
| --- | --- |
| 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“. |
| Erwartetes Testergebnis | Der Tisch wird gespeichert, damit er der Bestellung hinzugefügt werden kann und auf dem Kundenbeleg dargestellt werden kann. Das nächste Fragment wird dargestellt. In diesem Fragment werden alle verfügbaren Artikel angezeigt. |

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

**Artikel Einsehen**

|  |  |
| --- | --- |
| 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: |
| **Sonderfall 1** | Die Verbindung zum Server wird getrennt. |
| Erwartetes Testergebnis | Die Artikel werden weiterhin angezeigt. |
| Tatsächliches  Testergebnis | Der Bildschirm des Smartphones wird folgendermaßen dargestellt: |
| **Sonderfall 2** | Die Datenbank enthält keine verfügbaren Artikel die auf dem Smartphone angezeigt werden. |
| Erwartetes Testergebnis | Es werden keine Artikel angezeigt. Eine „Fehlermeldung“ gibt an, dass keine Artikel in der Datenbank verfügbar sind. |
| Tatsächliches  Testergebnis | Das Smartphone gibt folgende Fehlermeldung aus: |
| Test  bestanden | Normalablauf: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

**Artikel einer Bestellung hinzufügen**

|  |  |
| --- | --- |
| 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 |

**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 Bestellungs-ID, das Datum inklusive Uhrzeit wann die Bestellung erstellt wurde und die Bedienungs-ID. Also welche Bedienung die Bestellung aufgenommen hat.  Mit folgenden Daten wird getestet:   * Tisch-ID: 1 * Datum und Uhrzeit: 31.03.2018, 18:49 Uhr * Bedienungs-ID: 4 * Bestellungs-ID: 104 |
| Tatsächliches  Testergebnis | Die Datenbank sieht wie folgt aus: |
| **Sonderfall 1** | Wenn an diesem Tisch bereits eine Bestellung besteht, bei der zu diesem Zeitpunkt noch nicht alle bestellten Artikel bezahlt wurden. Die Bestellung wird um die neuen Artikel erweitert. |
| Erwartetes Testergebnis | Es wird in der Datenbank die Bestellung aktualisiert. Mit dem neuen Datum/Uhrzeit und den bereits bestehenden Artikeln und den neu hinzugefügten Artikeln. Der Preis wird ebenfalls aktualisiert. Der Test wird mit der gleichen Bestellung die im Normalfall getestet wurde, gestartet.  Es sollen zwei weitere Artikel bestellt werden.  Die Datenbank sieht vor der Bestellung folgendermaßen aus: |
| Tatsächliches  Testergebnis | Die Datenbank sieht nach dem Test wie folgt aus: |
| **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:    Nachdem die Verbindung wiederhergestellt wurde, kann die Bestellung abgeschickt werden. |
| Test  bestanden | Normalfall: Ja  Sonderfall 1: Ja  Sonderfall 2: Ja |

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