

Klausurprüfung (Fachtheorie) aus Programmieren und Software Engineering

im Haupttermin 2020/21

für den Aufbaulehrgang für Informatik – Tag
(SFKZ 8167)

für das Kolleg für Informatik – Tag
(SFKZ 8242)

Liebe Prüfungskandidatin,
liebe Prüfungskandidat!


Bitte füllen Sie zuerst die nachfolgenden Felder in Blockschrift aus, bevor Sie mit der Arbeit beginnen.

Maturaaccount (im Startmenü sichtbar):

Vorname

Zuname

Klasse

	<p style="text-align: center;">Haupttermin September 2021</p> <p style="text-align: center;">PROGRAMMIEREN UND SOFTWARE ENGINEERING</p> <p style="text-align: center;">Aufbaulehrgang für Informatik – Tag (SFKZ 8167) Kolleg für Informatik – Tag (SFKZ 8242)</p>	<p>Seite 2 von 13</p>
----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------

Generelle Hinweise zur Bearbeitung

Die Arbeitszeit für die Bearbeitung der gestellten Aufgaben beträgt 6 Stunden (360 Minuten). Die 3 Teilaufgaben sind unabhängig voneinander zu bearbeiten, Sie können sich die Zeit frei einteilen. Wir empfehlen jedoch eine maximale Bearbeitungszeit von 1.5 Stunden für Aufgabe 1, 1.5 Stunden für Aufgabe 2 und 3 Stunden für Aufgabe 3. Im Beurteilungsblatt können Sie die Anforderungen für die jeweilige Beurteilungsstufe nachlesen.

Hilfsmittel

In der Datei *SPG_Fachtheorie.sln* befindet sich das Musterprojekt, in dem Sie Ihren Programmcode hineinschreiben. Im Labor steht Visual Studio 2017 mit der .NET Core Version 2.1.520 zur Verfügung. Die C# Sprachversion ist 7.3. Daher können keine nullable reference Types oder records verwendet werden.

Mit der Software DBeaver können Sie sich zur generierten Datenbank verbinden und Werte für Ihre Unittests ablesen.

Zusätzlich wird ein implementiertes Projekt aus dem Unterricht ohne Kommentare bereitgestellt, wo Sie die Parameter von benötigten Frameworkmethoden nachsehen können.

Vorbereitung der Arbeitsumgebung

Starten Sie auf *U:\Fachtheorie_POS_NET* die Datei *downloadAssets.cmd*. Es wird DBeaver und das Visual Studio Musterprojekt auf Ihren PC kopiert. Die Dateien befinden sich an folgenden Orten:

- **DBeaver:** C:\Scratch\DBeaver\dbeaver.exe
- **SPG_Fachtheorie.sln:** P:\SPG_Fachtheorie\SPG_Fachtheorie.sln

Beim ersten Start von DBeaver müssen Sie das Fenster mit der Firewallabfrage schließen (nicht bestätigen). Nach dem Start des Musterprojektes führen Sie zuerst mit *Build - Rebuild Solution* einen Buildvorgang aus, damit die Tests sichtbar werden. Der Text Explorer ist unter *Test - Windows - Test Explorer* zu erreichen.

Generalthema

Gerade die aktuelle Situation hat gezeigt, wie wichtig es ist, dass der stationäre Handel auch im Internet präsent ist. Da einige Versuche erfolglos verlaufen sind, möchte der Betreiber eines großen Shoppingcenters direkt im Süden von Wien gemeinsam mit seinen Geschäften eine solche Plattform nun selbst gründen.

Die Idee ist, dass das Shoppingcenter die Logistik und die Software für den Versand der Waren zentral zur Verfügung stellt. Die einzelnen Geschäfte können über die gemeinsame Plattform des Shoppingcenters ihre Waren im Internet anbieten.

Teilaufgabe 1: Erstellen einer Datenbank

Arbeitsauftrag

Der Betreiber des Shoppingcenters benötigt zuerst ein kleines System, mit dessen Hilfe er überhaupt die interessierten Mieter (die Geschäfte) und das Warenangebot erheben kann. In einer Besprechung werden folgende Daten dafür definiert. Erstellen Sie Modelklassen, mit dessen Hilfe Sie mit EF Core eine SQLite Datenbank unter dem Namen *Store.db* erstellen. Die Anforderungen an diese Datenbank ist in den nachfolgenden Punkten und in den Bewertungskriterien definiert.

Das Geschäft (Tabelle Store)

Das Shoppingcenter möchte seine Geschäfte mit folgenden Informationen erfassen:

- Die "Hausnummer" im Shoppingcenter (Location, z. B. "Top 4")
- Das Stockwerk
- Den Firmennamen
- Der/die Mieter:in (tenant)
- Status (in Betrieb, vorübergehend geschlossen, aufgelassen)
- Versandangebot (Postversand, eigene Zustellung oder click & collect)

Der/die Mieter:in soll mit Vorname, Zuname und E-Mail Adresse erfasst werden. Es kommt vor, dass mehrere Geschäfte den selben Inhaber haben.

Hinweis: Es kommt vor, dass mehrere Geschäfte an einer Adresse in der Datenbank eingetragen sind. Das tritt dann auf, wenn ein Geschäft aufgelassen wurde und ein Nachmieter das Lokal übernimmt.

Das Warenangebot (Tabelle ProductCategory)

Um einen Eindruck über die Vielfalt zu bekommen, soll jedes Geschäft seine Waren in Kategorien einteilen und diese eintragen. Es sind keine Einzelprodukte oder gar der Preis zu erfassen, das ist Aufgabe des Hauptsystems.

Die Warenkategorie wird natürlich mit dem Namen erfasst (Haushaltsgeräte, Unterhaltungselektronik, ...) Zusätzlich kann jeder Store die Warenkategorie vom Versand ausschließen, so dass diese nicht gelistet wird. Das kommt bei Produkten vor, die nicht versendet werden dürfen wie Lebensmittel, Medikamente, etc.

Geschäfts- und Abholzeiten (Tabellen BusinessHours und PickupHours)

Für die zentrale Logistik ist es wichtig zu wissen, wann die Waren aus den Geschäften abgeholt werden können. Dies sollte natürlich nicht zu den Geschäftszeiten sein, sondern im Anschluss daran. Daher soll jedes Geschäft auch die bevorzugte Abholzeit für das Logistikunternehmen definieren können.

Es ist eine Aufteilung pro Wochentag vorzusehen, d. h. es können folgende Einträge gemacht werden:


- MO 8 - 18 als Geschäftszeit, 18 - 19 Uhr als Abholzeit
- DI 8 - 20 als Geschäftszeit, 7 - 8 Uhr als Abholzeit
- MI 8 - 18 als Geschäftszeit, 18 - 19 Uhr als Abholzeit
- DO 8 - 20 als Geschäftszeit, 7 - 8 Uhr als Abholzeit
- FR 8 - 18 als Geschäftszeit, 18 - 19 Uhr als Abholzeit
- SA 8 - 18 als Geschäftszeit

Abweichende Regelungen an Feiertagen oder besonderen Tagen wie vor Weihnachten sind nicht zu berücksichtigen.

Generieren der Datenbank

Im Projekt *SPG_Fachtheorie.Aufgabe1* finden Sie einen Ordner *Infrastructure* vor, der eine leere Klasse *StoreContext* beinhaltet. Führen Sie hier Ihre notwendigen Ergänzungen durch. Die Modelklassen erstellen Sie im leeren Ordner *Model*.

Starten Sie den Unittest *GenerateDbFromContextTest* im Projekt *SPG_Fachtheorie.Aufgabe1.Test*. Dieser Test versucht, die Datenbank zu erstellen und ist erfolgreich, wenn dies gelingt. Es werden keine weiteren Prüfungen in diesem Unittest durchgeführt, prüfen Sie daher in DBeaver ob die erzeugte Datenbank den definierten Kriterien entspricht.

	<p style="text-align: center;">Haupttermin September 2021</p> <p style="text-align: center;">PROGRAMMIEREN UND SOFTWARE ENGINEERING</p> <p style="text-align: center;">Aufbaulehrgang für Informatik – Tag (SFKZ 8167) Kolleg für Informatik – Tag (SFKZ 8242)</p>	<p>Seite 5 von 13</p>
----------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------

Bewertungskriterien

Zur Bewertung wird die von Ihnen erstellte SQLite Datenbank herangezogen und nach folgenden Gesichtspunkten analysiert

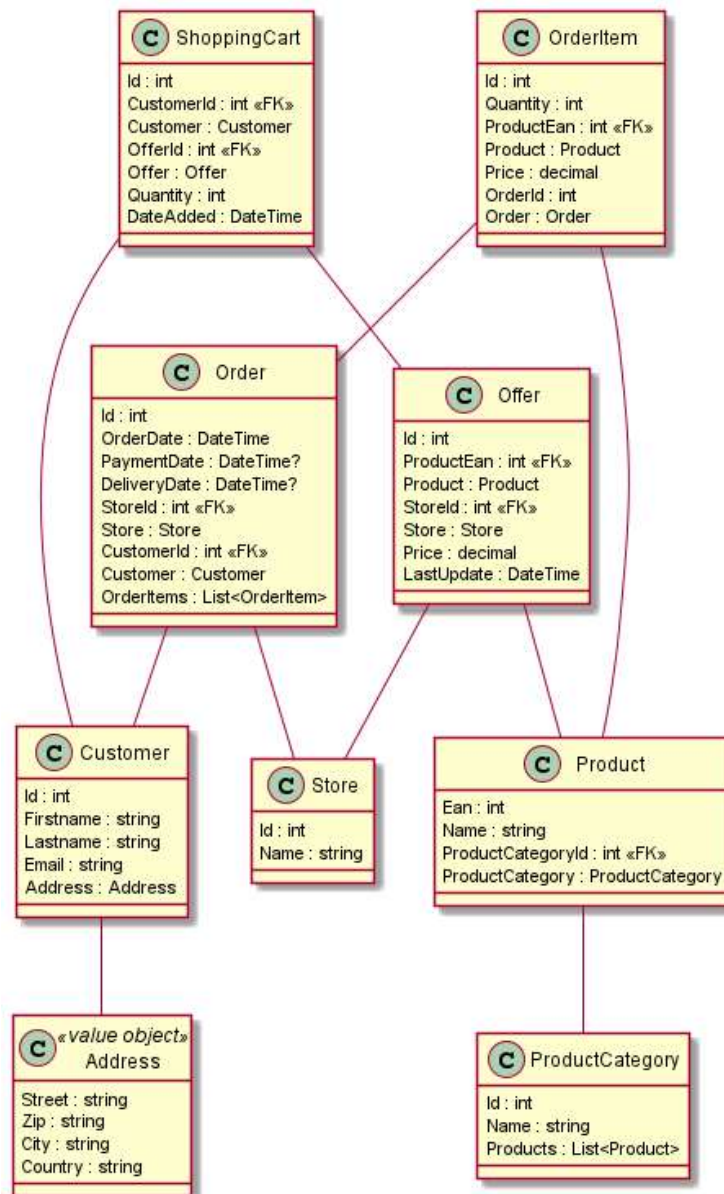
1. Es existiert eine Tabelle *Store* mit den Spalten, die die definierten Daten aufnehmen können.
2. Es existiert eine Tabelle *Tenant* mit den Spalten, die die definierten Daten aufnehmen kann.
3. Es existiert eine Tabelle *ProductCategory* mit den Spalten, die die eingegebenen Warenkategorien pro Geschäft speichern kann.
4. Es existiert eine Tabelle *BusinessHours*, die die Geschäftszeit pro Geschäft und Wochentag speichern kann.
5. Es existiert eine Tabelle *PickupHours*, die die Abholzeit pro Geschäft und Wochentagspeichern kann.
6. Die Tabelle *Store* verweist korrekt auf die Tabelle *Tenant* über einen Fremdschlüssel.
7. Die Tabelle *BusinessHours* verweist korrekt auf die Tabelle *Store* über einen Fremdschlüssel.
8. Die Tabelle *PickupHours* verweist korrekt auf die Tabelle *Store* über einen Fremdschlüssel.
9. Die Tabelle *Store* speichert den Status und das Versandangebot als Enumeration.



Teilaufgabe 2: Services und Unittests

Arbeitsauftrag: Erstellen eines OrderService

Fast alle Mieter wollten an der zentralen Verkaufsplattform teilnehmen. Nun geht es um die Abwicklung der eingehenden Bestellungen. Als Basis sind im Projekt *SPG_Fachtheorie.Aufgabe2* im Ordner *Model* schon Modelklassen vorgegeben.



Im Ordner *Services* befindet sich eine leere Klasse *OrderService*. Implementieren Sie folgende Methoden und stellen Sie die Funktionalität über entsprechende Unittests in der Klasse *OrderServiceTests* im Testprojekt *SPG_Fachtheorie.Aufgabe2.Test* sicher. Im Punkt *Bewertung* ist aufgeführt, welche Tests Sie implementieren müssen.

Im Projekt *SPG_Fachtheorie.Aufgabe2.Test* finden Sie zudem in der Klasse *DbContextTests* eine Methode *GetDbContext*. Verwenden Sie diese Methode, um eine initialisierte Datenbank zum Testen Ihrer Services zu bekommen.

Die initialisierte Datenbank wird im Ordner *SPG_Fachtheorie.Aufgabe2.Test\bin\Debug\netcoreapp2.1* unter dem Namen *Store.db* abgelegt. Sie können DBeaver verwenden, um auf diese Datenbank zuzugreifen. **Wichtig: Trennen Sie vor einem erneuten Testdurchlauf in DBeaver die Datenbank, sonst kann der Test diese nicht neu anlegen.**

bool AddToShoppingCart(int customerId, int offerId, int quantity)

Fügt das übergebene Angebot mit der übergebenen Anzahl in den Warenkorb des übergebenen Kunden ein. Dabei wird *DateAdded* auf den aktuellen Zeitstempel (*DateTime.UtcNow*) gesetzt. Befindet sich bereits ein solches Angebot im Warenkorb, wird die Menge zum bestehenden Eintrag dazugezählt und der Zeitstempel wird aktualisiert.

Liefert *false*, falls das Angebot oder der Kunde nicht gefunden wurde. Liefert *true* im Erfolgsfall. Die Methode darf keine Exceptions werfen.

void RemoveFromShoppingCart(int customerId, int offerId)

Löscht das übergebene Angebot aus dem Warenkorb des übergebenen Kunden. Die Methode darf keine Exceptions werfen.

bool Checkout(int customerId)

Erstellt pro unterschiedlichem Store im Warenkorb eine neue Bestellung in der Order Tabelle und legt das aktuelle Datum (*DateTime.UtcNow*) als Bestelldatum fest. Alle Produkte des Warenkorbes (*ShoppingCart*) sind als *OrderItem* zur Bestellung beim richtigen Store hinzuzufügen.

Liefert *false*, falls der Kunde nicht gefunden wurde oder kein Eintrag im Warenkorb vorhanden ist, sonst *true*.

Bewertungskriterien

1. *AddToShoppingCartNewOfferSuccessTest* beweist, dass die Methode *AddToShoppingCart* erfolgreich ein Angebot, welches zuvor nicht im Warenkorb war, mit der übergebenen Anzahl in den Warenkorb legt.

2. *AddToShoppingCartExistingOfferSuccessTest* beweist, dass die Methode *AddToShoppingCart* erfolgreich die übergebene Anzahl zu einem Angebot, welches schon im Warenkorb liegt, hinzufügt.
3. *AddToShoppingCartInvalidCustomerTest* beweist, dass die Methode *AddToShoppingCart* bei einem nicht vorhandenen Kunden *false* liefert.
4. *AddToShoppingCartInvalidOfferTest* beweist, dass die Methode *AddToShoppingCart* bei einem nicht vorhandenen Angebot *false* liefert.
5. *RemoveFromShoppingCartSuccessTest* beweist, dass die Methode *RemoveFromShoppingCart* erfolgreich ein Angebot aus dem Warenkorb entfernt.
6. *CheckoutSuccessTest* beweist, dass die Methode *Checkout* mit den Produkten aus dem Warenkorb des übergebenen Kunden eine neue Bestellung erstellt und *true* liefert.
7. *CheckoutInvalidCustomerTest* beweist, dass die Methode *Checkout* bei einem nicht vorhandenen Kunden *false* liefert.
8. *CheckoutEmptyShoppingCartTest* beweist, dass die Methode *Checkout* bei einem nicht leeren Warenkorb *false* liefert.

Teilaufgabe 3: Webapplikation

Arbeitsauftrag

Die interne Bearbeitung der eingehenden Bestellungen (Versand, Pflege der Produkte, ...) soll allen teilnehmenden Stores eine Webapplikation zur Verfügung gestellt werden. Beim Start wird der Store, der am System angemeldet ist, ausgewählt. Dies ist bereits im Musterprojekt implementiert. Die Anforderungen an diese Applikation ist in den nachfolgenden Punkten und in den Bewertungskriterien definiert.

Sie können entscheiden, ob Sie die Aufgabe als ASP.NET Core MVC oder ASP.NET Core Razor Pages Applikation umsetzen. Die jeweiligen Projekte werden in *SPG_Fachtheorie.Aufgabe3Mvc* bzw. *SPG_Fachtheorie.Aufgabe3RazorPages* bereitgestellt. Tipp: Aktivieren Sie das gewünschte Projekt als Startprojekt in Visual Studio.

Das Datenmodell sowie die Datenbank wird von Aufgabe 2 verwendet. Der Datenbankkontext ist bereits registriert und kann über die *StoreContext* Klasse mittels Dependency Injection genutzt werden. Des Weiteren ist die Klasse *AuthService* registriert. Sie bietet mit dem Property *CurrentStore* die Möglichkeit, auf die Id (int) des aktuell angemeldeten Stores zuzugreifen.

Anzeigen der Produktkategorien und deren Produkte

In der Tabelle *ProductCategory* befinden sich alle eingetragenen Kategorien. Erstellen Sie eine Seite */Category/Index*, die folgende Informationen für jede Kategorie anzeigt: Name, Anzahl der eingetragenen Produkte. Die Darstellung ist frei, es soll aber für jede Kategorie ein Link zur entsprechenden Detailseite dieser Kategorie vorgesehen werden.

Diese Detailseite der Kategorie soll unter */Category/Details/{CategoryId}* erreichbar sein. Sie zeigt eine Auflistung aller Produkte, die sich in dieser Kategorie befinden, an. Falls ein Angebot des angemeldeten

Stores zu diesem Produkt existiert, wird der Preis angezeigt. Ansonsten wird die Ausgabe "(kein Angebot)" angezeigt. Das kann mittels einer HTML Tabelle geschehen. Es sollen EAN Nummer, Name und (wenn vorhanden) der Preis des Produktes in diesem Store angezeigt werden. Zu jedem Produkt soll ein Link auf die Detailseite des Produktes (siehe nächster Punkt) angeboten werden.

Produkte ansehen und bearbeiten

Unter */Product/Details/{ProductId}* soll eine Detailansicht des übergebenen Produktes mit den in der Datenbank eingetragenen Informationen (Ean Code, Name, Kategorienname) erfolgen. Es ist nur das Produkt anzuzeigen, nicht das spezifische Angebot des Stores.

Auf dieser Seite sollen folgende Links angeboten werden:

- Link auf die Editierseite des Produktes unter *Edit/{ProductId}*.
- Link auf die Löschseite des Produktes unter *Delete/{ProductId}*.
- Link auf eine Seite zur Erstellung neuer Produkte unter *Add*.
- Link auf eine Seite zur Erstellung oder Bearbeitung eines Angebotes unter */Offer/Upsert/{ProductId}*.

Auf der Editierseite soll der Name des Produktes und die Kategorie bearbeitet werden können. Die Kategorie soll mittels Dropdown Liste neu zugeordnet werden können. Die Löschseite soll eine Nachfrage anbieten, ob das Produkt gelöscht werden soll. Achtung: Wenn Angebote zum Produkt existieren, ist natürlich kein löschen möglich. Geben Sie in diesem Fall statt der Nachfrage eine entsprechende Meldung im Browser aus. Bei der Neuerfassung soll die EAN Nummer und der Name eingegeben werden können. Die Kategorie soll über ein Dropdown Menü ausgewählt werden.

Angebote erstellen und bearbeiten

Das Angebot ist nichts anderes als eine storespezifische Zuordnung eines Preises zu einem Produkt. Unter `/Offer/Upsert/{ProductId}` sollen noch einmal die Details des Produktes (EAN, Name, Kategorienname) angezeigt werden. Ein Eingabefeld für den Preis ist mit dem vorhandenen Angebot vorinitialisiert. Existiert noch kein Angebot, ist das Feld natürlich leer. Preise zwischen 0 und 999999 Euro sind gültig, Kommazahlen sind erlaubt. Bei falschen Eingaben ist eine Validierungsmeldung auszugeben. Das Feld `LatUpdate` wird bei einer Änderung bzw. Neuerstellung auf das aktuelle Systemdatum (`DateTime.UtcNow`) gesetzt.

Bewertungskriterien

1. Die Adresse `/Category/Index` zeigt eine Übersicht aller in der Datenbank eingetragenen Produktkategorien an.
2. Die Adresse `/Category/Index` zeigt die Anzahl der Produkte in dieser Kategorie korrekt an.
3. Die Adresse `/Category/Details/{CategoryId}` zeigt eine Übersicht der in der Datenbank eingetragenen Angebote des Stores der jeweiligen Kategorie an.
4. Die Adresse `/Category/Details/{CategoryId}` zeigt den Preis des Angebotes zu einem Produkt - wenn vorhanden - korrekt an.
5. Die Adresse `/Product/Details/{ProductId}` zeigt Detailinformationen zum übergebenen Produkt an.
6. Die Adresse `/Product/Add` ermöglicht es, ein neues Produkt zur Datenbank hinzuzufügen.
7. Die Adresse `/Product/Add` bietet eine Dropdown Liste an, um die Kategorie zuzuordnen.
8. Die Adresse `/Product/Add` meldet zurück, wenn ein Produkt mit der eingegebenen EAN Nummer schon existiert.
9. Die Adresse `/Product/Edit/{ProductId}` ermöglicht es, ein Produkt in der Datenbank zu editieren.
10. Die Adresse `/Product/Edit/{ProductId}` bietet eine Dropdown Liste an, um die Kategorie zuzuordnen.
11. Die Adresse `/Product/Delete/{ProductId}` ermöglicht es, das übergebene Produkt aus der Datenbank zu entfernen.
12. Die Adresse `/Product/Delete/{ProductId}` meldet zurück, wenn zu diesem Produkt bereits Angebote existieren und ein Löschen nicht möglich ist.
13. Die Adresse `/Offer/Upsert/{ProductId}` ermöglicht es, einen Preis für ein vorhandenes Produkt zu definieren. Ist ein solcher Preis schon eingetragen, wird er aktualisiert. Existiert noch kein Angebot, wird es erstellt.
14. Die Adresse `/Offer/Upsert/{ProductId}` prüft den Preis auf Gültigkeit. Preise zwischen 0 und 999999 Euro sind gültig, Kommazahlen sind erlaubt.

Beurteilungsblatt (vom Prüfer auszufüllen)

Für jede erfüllte Teilaufgabe gibt es 1 Punkt. Beim Teilthema 3 werden für teilweise korrekte Lösungen auch 0.5 Punkte vergeben. In Summe sind also 31 Punkte zu erreichen.

Für eine Berücksichtigung der Jahresnote müssen mindestens 30 % der Gesamtpunkte erreicht werden. Für eine positive Beurteilung der Klausur müssen mindestens 50 % der Gesamtpunkte erreicht werden.

Beurteilungsstufen:

- 31 – 27.5 Punkte: Sehr gut
- 27 – 23.5 Punkte: Gut
- 23 – 19.5 Punkte: Befriedigend
- 19 – 15.5 Punkte: Genügend

Teilaufgabe 1: Erstellen einer Datenbank

1. Es existiert eine Tabelle *Store* mit den Spalten, die die definierten Daten aufnehmen können.
2. Es existiert eine Tabelle *Tenant* mit den Spalten, die die definierten Daten aufnehmen kann.
3. Es existiert eine Tabelle *ProductCategory* mit den Spalten, die die eingegebenen Warenkategorien pro Geschäft speichern kann.
4. Es existiert eine Tabelle *BusinessHours*, die die Geschäftszeit pro Geschäft und Wochentag speichern kann.
5. Es existiert eine Tabelle *PickupHours*, die die Abholzeit pro Geschäft und Wochentagspeichern kann.
6. Die Tabelle *Store* verweist korrekt auf die Tabelle *Tenant* über einen Fremdschlüssel.
7. Die Tabelle *BusinessHours* verweist korrekt auf die Tabelle *Store* über einen Fremdschlüssel.
8. Die Tabelle *PickupHours* verweist korrekt auf die Tabelle *Store* über einen Fremdschlüssel.
9. Die Tabelle *Store* speichert den Status und das Versandangebot als Enumeration.

Vollst. erfüllt	Tw. Erfüllt	Nicht erfüllt

Teilaufgabe 2: Services und Unittests

1. *AddToShoppingCartNewOfferSuccessTest* beweist, dass die Methode *AddToShoppingCart* erfolgreich ein Angebot, welches zuvor nicht im Warenkorb war, mit der übergebenen Anzahl in den Warenkorb legt.
2. *AddToShoppingCartExistingOfferSuccessTest* beweist, dass die Methode *AddToShoppingCart* erfolgreich die übergebene Anzahl zu einem Angebot, welches schon im Warenkorb liegt, hinzufügt.
3. *AddToShoppingCartInvalidCustomerTest* beweist, dass die Methode *AddToShoppingCart* bei einem nicht vorhandenen Kunden *false* liefert.
4. *AddToShoppingCartInvalidOfferTest* beweist, dass die Methode *AddToShoppingCart* bei einem nicht vorhandenen Angebot *false* liefert.
5. *RemoveFromShoppingCartSuccessTest* beweist, dass die Methode *RemoveFromShoppingCart* erfolgreich ein Angebot aus dem Warenkorb entfernt.

Vollst. erfüllt	Tw. Erfüllt	Nicht erfüllt

6. CheckoutSuccessTest beweist, dass die Methode *Checkout* mit den Produkten aus dem Warenkorb des übergebenen Kunden eine neue Bestellung erstellt und *true* liefert.
7. CheckoutInvalidCustomerTest beweist, dass die Methode *Checkout* bei einem nicht vorhandenen Kunden *false* liefert.
8. CheckoutEmptyShoppingCartTest beweist, dass die Methode *Checkout* bei einem nicht leeren Warenkorb *false* liefert.

Teilaufgabe 3: Webapplikation

1. Die Adresse */Category/Index* zeigt eine Übersicht aller in der Datenbank eingetragenen Produktkategorien an.
2. Die Adresse */Category/Index* zeigt die Anzahl der Produkte in dieser Kategorie korrekt an.
3. Die Adresse */Category/Details/{CategoryId}* zeigt eine Übersicht der in der Datenbank eingetragenen Angebote des Stores der jeweiligen Kategorie an.
4. Die Adresse */Category/Details/{CategoryId}* zeigt den Preis des Angebotes zu einem Produkt - wenn vorhanden - korrekt an.
5. Die Adresse */Product/Details/{ProductId}* zeigt Detailinformationen zum übergebenen Produkt an.
6. Die Adresse */Product/Add* ermöglicht es, ein neues Produkt zur Datenbank hinzuzufügen.
7. Die Adresse */Product/Add* bietet eine Dropdown Liste an, um die Kategorie zuzuordnen.
8. Die Adresse */Product/Add* meldet zurück, wenn ein Produkt mit der eingegebenen EAN Nummer schon existiert.
9. Die Adresse */Product/Edit/{ProductId}* ermöglicht es, ein Produkt in der Datenbank zu editieren.
10. Die Adresse */Product/Edit/{ProductId}* bietet eine Dropdown Liste an, um die Kategorie zuzuordnen.
11. Die Adresse */Product/Delete/{ProductId}* ermöglicht es, das übergebene Produkt aus der Datenbank zu entfernen.
12. Die Adresse */Product/Delete/{ProductId}* meldet zurück, wenn zu diesem Produkt bereits Angebote existieren und ein Löschen nicht möglich ist.
13. Die Adresse */Offer/Upsert/{ProductId}* ermöglicht es, einen Preis für ein vorhandenes Produkt zu definieren. Ist ein solcher Preis schon eingetragen, wird er aktualisiert. Existiert noch kein Angebot, wird es erstellt.
14. Die Adresse */Offer/Upsert/{ProductId}* prüft den Preis auf Gültigkeit. Preise zwischen 0 und 999999 Euro sind gültig, Kommazahlen sind erlaubt.

Vollst. erfüllt	Tw. Erfüllt	Nicht erfüllt

Punktesumme Teilaufgabe 1
Punktesumme Teilaufgabe 2
Punktesumme Teilaufgabe 3
Summe

	von 9
	von 8
	von 14
	von 31

Beurteilung im Wintersemester
Beurteilung im Sommersemester
Resultierende Jahresnote
Beurteilung der Klausur
Gesamtbeurteilung
