

Klausurprüfung (Fachtheorie) aus Programmieren und Software Engineering

im Nebentermin 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 align="center">Nebentermin Jänner 2020</p> <p align="center">PROGRAMMIEREN UND SOFTWARE ENGINEERING</p> <p align="center">Aufbaulehrgang für Informatik – Tag (SFKZ 8167) Kolleg für Informatik – Tag (SFKZ 8242)</p>	<p align="center">Seite 2 von 10</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 2022 mit der .NET Core Version 6 zur Verfügung.

Mit der Software DBeaver oder SQLite Studio 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.

Teilaufgabe 1: Erstellen von EF Core Modelklassen

Ein Rechnungssystem für Kleinunternehmen

Im Bereich der KMUs (Klein und Mittelbetriebe) werden oftmals Rechnungen noch händisch mit Hilfe von Microsoft Word erstellt. Sie möchten nach Ihrer Ausbildung ein Startup gründen, welches eine bequemere Rechnungserstellung für Unternehmen anbietet.

Eine klassische Rechnung hat folgendes Aussehen:

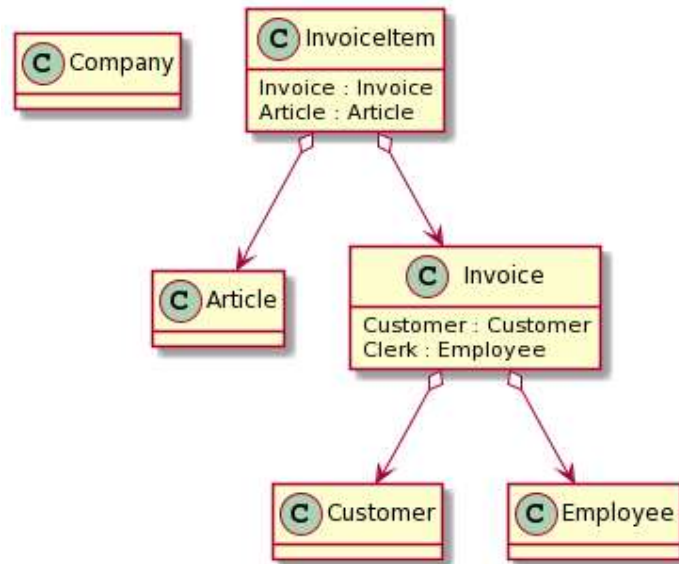
Musterfirma GmbH

Frau
Stefanie Musterkundin
Straße 1
1010 Ort
Rechnung

Firmen E-Mail: office@musterfirma.at
Firmen Telefon: (01) 123 45 67
Bearbeiter/in: Lukas Eifrig
Rechnungsnummer: 123456
Rechnungsdatum: 3. November 2021
Kundennummer: 45678

Position	Art. Nr.	Artikelbezeichnung	Anzahl	Einzelpreis	Gesamtpreis
1	1001	Samsung Galaxy A52s	1	393,90	393,90
2	1002	Samsung Galaxy Tab A7	1	178,00	178,00
3	1003	SanDisk Extreme, microSD	2	70,49	140,98
					712,88
3 % Rabatt					- 21,39
Zu zahlen:					691,49

Als Unterstützung wurde bereits die Umsetzung in ein Klassenmodell vorgenommen. Dieses Modell zeigt allerdings nur die Beziehungen zwischen den Klassen, zusätzliche Properties sind von Ihnen zu implementieren.



Arbeitsauftrag

Sie finden in der Mustersolution ein Projekt *SPG_Fachtheorie.Aufgabe1*. Dort ist in der Klasse *InvoiceContext* bereits ein leerer Datenbankcontext für EF Core vorhanden. Leere Klassendefinitionen wurden im Ordner *Model* bereits erstellt.

Implementieren Sie das obige Klassenmodell, sodass diese Klassen mit EF Core in eine Datenbank persistiert werden können. Beachten Sie dabei folgende Anweisungen.

Am Ende führen Sie den Test *CreateDatabaseTest* aus. Er versucht, eine Datenbank anzulegen. Dieser Test muss erfolgreich durchlaufen werden.


Allgemeine Hinweise

Überlegen Sie sich für jede Klasse einen geeigneten Primärschlüssel. Der einfachste Zugang ist sicher ein auto increment Wert. Primärschlüssel, die nicht Id heißen, müssen entsprechend mit *Key* annotiert werden.

Sehen Sie bei den Navigations immer ein Feld für die Speicherung des Fremdschlüsselwertes vor. Berechnete Werte dürfen natürlich nicht als Felder definiert werden.

Klasse Company

Es soll der Kopf der Rechnung, der wie im Muster ersichtlich aus einigen Firmendaten besteht, generiert werden können. Erstellen Sie dafür die benötigten Felder für Firmenname, Anschrift, E-Mail und Telefonnummer. Diese Klasse hat keine Beziehungen zu den anderen Klassen, da es nur einen einzigen

	<p style="text-align: center;">Nebentermin Jänner 2020</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 10</p>
--	---	-----------------------

Datensatz in der Datenbank, nämlich das Unternehmen welches die Datenbank nutzt, geben wird. Mehrere Unternehmen bilden Sie aus Sicherheitsgründen in separaten Datenbanken ab.

Klasse Employee

Jede Rechnung wird von einer Sachbearbeiter:in (clerk) erstellt. Diese ist eine Mitarbeiter:in und zumindest mit dem Namen als Employee zu speichern.

Klasse Article

Jedes Unternehmen kann Artikel mit einer internen Artikelnummer hinzufügen. Jeder Artikel hat auch einen Preis.

Klasse Customer

Das Unternehmen kann auch seine Kunden erfassen. Bei Kleinunternehmen geschieht dies durch Mitarbeiter:innen im Büro. Eine Selbstregistrierung ist (noch) nicht vorgesehen. Fügen Sie die benötigten Felder hinzu, sodass Name und Anschrift des Kunden auf die Rechnung gedruckt werden kann. Als Kundennummer könnte ein auto increment Wert dienen.

Verwenden Sie für die Anrede (salutation) ein enum mit den Werten *Mr* (männliche Anrede) und *Ms* (weibliche Anrede).

Klasse Invoice

Bei einer Rechnung wird eine neue Instanz dieser Klasse erstellt. Beachten Sie, dass ein Rabatt gegeben werden kann. Dies können Sie als Prozentwert (z. B. 3 für 3 % Rabatt) speichern. Achten Sie auch auf das Rechnungsdatum und die Rechnungsnummer. Für die Rechnungsnummer kann auch ein auto increment Wert verwendet werden.

Klasse InvoiceItem

Stellt die Rechnungsposition dar. Beachten Sie, dass sich die Artikelpreise nach der Rechnungslegung ändern können. Daher ist der Preis, der auf der Rechnung steht, ebenfalls zu speichern.

Bewertung

Jede umgesetzte Klasse wird mit 1 Punkt bewertet, wenn sie alle definierten Spalten aufnehmen kann. Zusätzlich wird ein funktionierender Unittest (CreateDatabaseTest) mit 1 Punkt bewertet. Dadurch können in Summe 7 Punkte bei dieser Aufgabe erreicht werden.

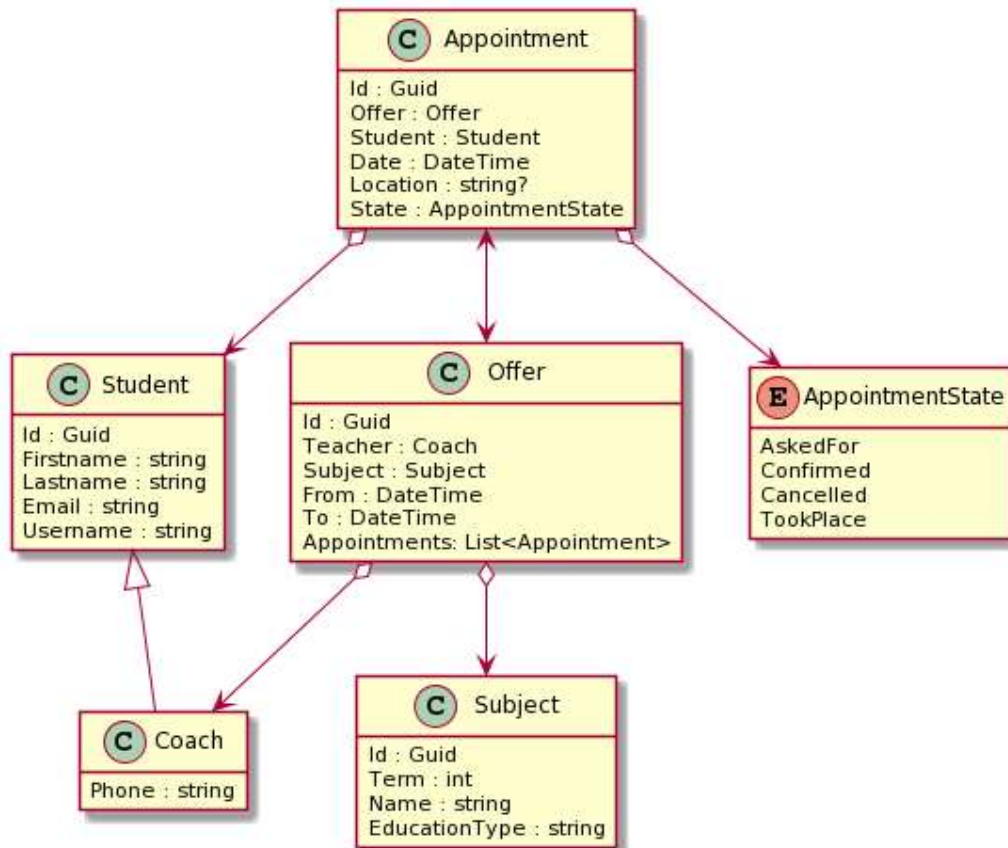
Teilaufgabe 2: Services und Unittests

Eine Nachhilfebörse

Derzeit gibt es bei uns an der Schule eine Nachhilfebörse, in der Schüler:innen oder Studierende Nachhilfe anbieten können (Coaches). Nach einem Login sind die Kontaktdaten sichtbar und Schüler:innen oder Studierende können mit den Coaches per Mail, WhatsApp, etc. in Kontakt treten.

Nun soll dieses System um eine Terminverwaltung ergänzt werden, dass Schüler:innen oder Studierende direkt im System einen Termin vorschlagen können. Dieser Termin kann von den Coaches bestätigt werden.

Eine erste Überlegung lieferte folgendes Klassendiagramm:



Wie erkennbar ist die zentrale Klasse die Klasse *Offer*. Ein Coach kann zu einem Gegenstand seine Dienste anbieten. Das Angebot ist zeitlich begrenzt, dies wird in den Feldern *From* und *To* gespeichert. Der Gegenstand (*Subject*) ist in das Semester (*Term*), den Namen des Gegenstandes und den Ausbildungstyp (*EducationType*) unterteilt. So kann z. B. ein Coach Nachhilfe für das 3. Semester in POS in der Ausbildungsrichtung Abendkolleg anbieten.

Nimmt ein Kunde das Angebot an, wird ein Appointment erstellt. Dort ist der konkrete Zeitpunkt gespeichert, an dem die Nachhilfe statt finden soll. Der Ort (*Location*) wird vom Coach nach der Anmeldung eingegeben. Das Appointment hat 4 Zustände (appointment states):

- *AskedFor* für einen Termin, den ein Kunde angefragt hat.
- *Confirmed*, wenn der Coach den Termin bestätigt hat.
- *Cancelled*, wenn ein Termin abgesagt werden muss.
- *TookPlace*, wenn die Nachhilfe tatsächlich stattgefunden hat.

Arbeitsauftrag

Implementieren der Services

Sie finden in der Mustersolution ein Projekt *SPG_Fachtheorie.Aufgabe2*. Dort ist bereits das Modell bereits mittels EF Core Modelklassen abgebildet. Implementieren Sie in der Klasse *AppointmentService* folgende Methoden:

bool AskForAppointment(Guid offerId, Guid studentId, DateTime date)

Studierende können einen Termin für eine Nachhilfesession vorschlagen.

- Liefert *false*, wenn kein Angebot (offer) zu dieser *offerId* existiert.
- Liefert *false*, wenn das Angebot für diesen Zeitpunkt nicht gültig ist.
- Liefert *true*, wenn die Buchung in die Datenbank eingetragen wurde.

bool ConfirmAppointment(Guid appointmentId)

Setzt den Status auf *Confirmed*.

- Liefert *false*, wenn kein Appointment zu dieser *appointmentId* existiert.
- Liefert *false*, wenn der Status *Confirmed*, *Cancelled* oder *TookPlace* ist.
- Liefert *true*, wenn der Status in der Datenbank auf *Confirmed* gesetzt wurde.

bool CancelAppointment(Guid appointmentId, Guid studentId)

Setzt den Status auf *Confirmed*. *studentId* hat hier 2 Bedeutungen: Die *studentId* kann der der Coach (die Id entspricht also der Id des Coaches) oder ein Kunde sein. Je nach dem verhält sich die Methode:

- Ein Kunde darf nur Termine im Status *AskedFor* absagen.
- Ein Coach darf Termine im Status *AskedFor* oder *Confirmed* absagen.

Ist ein Absagen aufgrund des eingetragenen Status nicht möglich, liefert die Methode *false*. Ansonsten setzt sie den Status auf *Cancelled* und liefert *true*.

Schreiben von Unittests

Ebenfalls in diesem Projekt ist eine Testklasse (*AppointmentServiceTests*) vorgegeben. Implementieren Sie Unittests, die folgende Testfälle abbilden sollen:

Welches Methodenverhalten diese Tests prüfen sollen, ist aus dem Namen des Tests ersichtlich:

- AskForAppointmentSuccessTest
- AskForAppointmentReturnsFalseIfNoOfferExists
- AskForAppointmentReturnsFalseIfOutOfDate
- ConfirmAppointmentSuccessTest
- ConfirmAppointmentReturnsFalseIfStatelInvalid
- CancelAppointmentStudentSuccessTest
- CancelAppointmentCoachSuccessTest
- ConfirmAppointmentStudentReturnsFalseIfStatelInvalid
- ConfirmAppointmentCoachReturnsFalseIfStatelInvalid

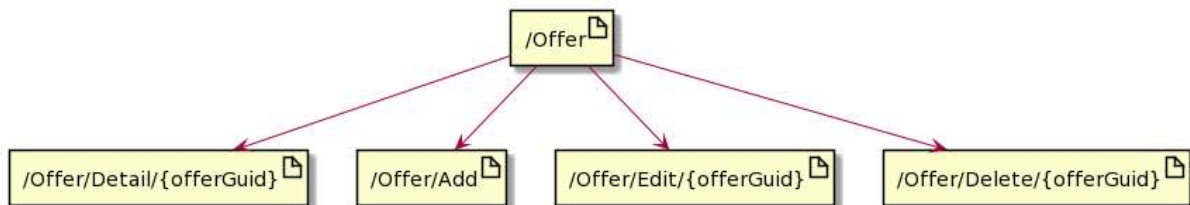
Bewertung

Jedes Feature der Servicemethoden wird mit 1 Punkt bewertet, also 3 Punkte für die Features von AskForAppointment, 3 Punkte für die Features von ConfirmAppointment und 2 Punkte für die Features von CancelAppointment. Zudem gibt es für jeden der oben angeführten Unittests 1 Punkt. Daher sind maximal 17 Punkte zu erreichen.

Teilaufgabe 3: Webapplikation


Themenstellung

Es soll die Nachhilfebörse von Aufgabe 2 als Webapplikation umgesetzt werden. Da das Modell bereits vorgegeben ist, und Ihre Services nicht benötigt werden, ist diese Aufgabe unabhängig von Aufgabe 2 zu lösen.



Arbeitsauftrag

Implementieren Sie die oben abgebildeten Seiten im Form von ASP.NET Core Razor Pages oder ASP.NET Core MVC. Sie finden in der Mustersolution ein bereits erstelltes Projekt *SPG_Fachtheorie.Aufgabe3.Razor* bzw. *SPG_Fachtheorie.Aufgabe3.Mvc*.

	<p style="text-align: center;">Nebentermin Jänner 2020</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 style="text-align: right;">Seite 9 von 10</p>
--	---	--

Für die Authentifizierung steht Ihnen das Service *AuthService* zur Verfügung. Das Property *Username* liefert Ihnen den angemeldeten User (oder null, falls kein User angemeldet ist).

Seite /Offer

Zeigt eine Liste aller Nachhilfeangebote (Offers) des angemeldeten Coaches an. Ist kein Coach angemeldet, so wird auf die Loginseite verwiesen. Stellen Sie dies durch eine entsprechende *Authorize* Annotation sicher.

Es sollen die Felder *Term* (Semester), *Name* (Name des Gegenstandes) *EducationType* (Ausbildungsrichtung) aus Subject sowie *From* und *To* aus Offer in Tabellenform angezeigt werden.

Folgende Spalten sollen zusätzlich ausgegeben werden:

- Anzahl der Appointments (der Status des Appointments muss nicht berücksichtigt werden).
- Link zur Detailseite (*/Offer/Detail/{offerGuid}*) des jeweiligen Angebotes.
- Link zur Edit Seite (*/Offer/Edit/{offerGuid}*) des jeweiligen Angebotes.
- Link zur Delete Seite (*/Offer/Delete/{offerGuid}*) des jeweiligen Angebotes.

Am Ende der Seite soll ein Link zur Add Seite (*/Offer/Add*) erscheinen.


Seite /Offer/Detail/{offerGuid}

Zeigt alle Appointments dieses Angebotes an. Dabei sind folgende Felder in Tabellenform auszugeben: Vor- und Zuname des Studierenden, Datum, Ort und Status. Abgesagte Termine sollen grau hinterlegt werden. Am Ende der Seite soll ein Link zurück zur Seite */Offer* führen.

Seite /Offer/Add

Fügt ein neues Offer hinzu. Es sollen folgende Punkte berücksichtigt werden:

- Der Gegenstand soll mittels Dropdown Menü (select Element in HTML) gewählt werden. Im Dropdown Menü soll natürlich nicht die GUID des Gegenstandes sondern als String der Form "Semester - Name - EducationType" ausgegeben wrden.
- Für die Eingabefelder der Werte *From* und *To* sind Datumsfelder (*input type="date"*) in HTML zu verwenden.
- DateTo darf nicht in der Vergangenheit liegen.
- Bei Fehlerhaften Eingaben sind natürlich die Daten wieder anzuzeigen und es wird eine entsprechende Fehlermeldung ausgegeben.

	<p style="text-align: center;">Nebentermin Jänner 2020</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 10 von 10</p>
--	---	------------------------

Seite /Offer/Edit/{offerGuid}

Über diese Seite kann ein bestehendes Angebot (Offer) verlängert werden. Dadurch ist nur das Feld *To* editierbar. Stellen Sie also ein Eingabefeld für das neue Datum bereit. Überprüfen Sie, ob das neue Datum größer als das bestehende Datum ist. Es sollen nur Verlängerungen möglich sein.

Am Beginn der Seite sollen nochmals die Daten des Angebotes (Gegenstand, Semester, Ausbildungstyp, Von und Bis) ausgegeben werden.

Am Ende der Seite soll ein Link zurück zur Seite */Offer* führen.

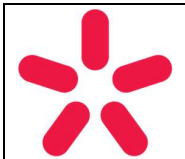
Seite /Offer/Delete/{offerGuid}

Löscht ein Offer, wenn es keine Appointments dazu gibt. Gibt es eingetragene Appointments zu diesem Termin, so ist auf der Seite die Meldung "Löschen nicht möglich, denn es gibt eingetragene Termine" auszugeben.

Nach dem Klick auf Löschen soll auf die Seite */Offer* zurückgesprungen werden.

Bewertung

Jedes der geforderten Features der einzelnen Seiten wird mit 1 Punkt bewertet. Dadurch sind maximal 20 Punkte zu erreichen.




Beurteilungsblatt (vom Prüfer auszufüllen)

Für jede erfüllte Teilaufgabe gibt es 1 Punkt. In Summe sind also 44 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.

Beurteilungstufen:

44 – 39 Punkte: Sehr gut, 38 – 34 Punkte: Gut, 33 – 28 Punkte: Befriedigend, 27 – 23 Punkte: Genügend

Teilaufgabe 1: Erstellen einer Datenbank (jew. 1 Punkt)	Erfüllt	Nicht erf.
Es existiert eine Tabelle Company mit den Spalten, die die definierten Daten aufnehmen können.		
Es existiert eine Tabelle InvoiceItem mit den Spalten, die die definierten Daten aufnehmen können.		
Es existiert eine Tabelle Article mit den Spalten, die die definierten Daten aufnehmen können.		
Es existiert eine Tabelle Invoice mit den Spalten, die die definierten Daten aufnehmen können.		
Es existiert eine Tabelle Customer mit den Spalten, die die definierten Daten aufnehmen können.		
Es existiert eine Tabelle Employee mit den Spalten, die die definierten Daten aufnehmen können.		
CreateDatabaseTest kann die Datenbank erzeugen.		
Teilaufgabe 2: Services und Unittests (jew. 1 Punkt)	Erfüllt	Nicht erf.
AskForAppointment liefert false, wenn kein Angebot (offer) zu dieser offerId existiert.		
AskForAppointment liefert false, wenn das Angebot für diesen Zeitpunkt nicht gültig ist.		
AskForAppointment liefert true, wenn die Buchung in die Datenbank eingetragen wurde.		
ConfirmAppointment liefert false, wenn kein Appointment zu dieser appointmentId existiert.		
ConfirmAppointment liefert false, wenn der Status Confirmed, Cancelled oder TookPlace ist.		
ConfirmAppointment liefert true, wenn der Status in der Datenbank auf Confirmed gesetzt wurde.		
CancelAppointment gestattet Kunden nur das Absagen im Status AskedFor.		
CancelAppointment gestattet Coaches nur das Absagen im Status AskedFor oder Confirmed.		
AskForAppointmentSuccessTest		
AskForAppointmentReturnsFalseIfNoOfferExists		
AskForAppointmentReturnsFalseIfOutOfDate		
ConfirmAppointmentSuccessTest		
ConfirmAppointmentReturnsFalseIfStatusInvalid		
CancelAppointmentStudentSuccessTest		
CancelAppointmentCoachSuccessTest		
ConfirmAppointmentStudentReturnsFalseIfStatusInvalid		
ConfirmAppointmentCoachReturnsFalseIfStatusInvalid		

	<p style="text-align: center;">Nebentermin Jänner 2020</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>
--	---

Teilaufgabe 3: Webapplikation (jew. 1 Punkt)	Erfüllt	Nicht erf.
<i>Seite /Offer</i>		
Anzahl der Appointments (der Status des Appointments muss nicht berücksichtigt werden).		
Link zur Detailseite (/Offer/Detail/{offerGuid}) des jeweiligen Angebotes.		
Link zur Edit Seite (/Offer/Edit/{offerGuid}) des jeweiligen Angebotes.		
Link zur Delete Seite (/Offer/Delete/{offerGuid}) des jeweiligen Angebotes.		
<i>Seite /Offer/Detail/{offerGuid}</i>		
Die Seite gibt die verlangten Informationen aus.		
Abgesagte Termine werden grau hinterlegt.		
Ein Link zurück zur Seite /offer ist vorhanden.		
<i>Seite /Offer/Add</i>		
Der Gegenstand kann mittels Dropdown Menü gewählt werden.		
Für From und To werden Datumsfelder verwendet.		
DateTo wird so geprüft, dass es nicht in der Vergangenheit liegt.		
Bei Fehlerhaften Eingaben wird auf die Seite zurückverwiesen.		
Bei erfolgreichen Eingaben werden die Daten in der Datenbank gespeichert.		
<i>Seite /Offer/Edit/{offerGuid}</i>		
Am Beginn der Seite werden die Daten des Angebotes angezeigt.		
Für DateTo wird ein Datumsfeld verwendet.		
DateTo wird so überprüft, dass es höher als der alte Wert sein muss.		
Ein Link führt zur Seite /offer zurück.		
Das editierte Angebot wird der Datenbank gespeichert.		
<i>Seite /Offer/Delete/{offerGuid}</i>		
Die Seite lehnt das Löschen ab, wenn es Anmeldungen für das Angebot gibt.		
Die Seite löscht - wenn möglich - das Angebot aus der Datenbank.		
Ein Link führt zur Seite /offer zurück.		