	Höhere technische Bundeslehr- und Versuchsanstalt für Textilindustrie und Datenverarbeitung
	Abteilungen: **Aufbaulehrgang für Informatik – Tag (SFKZ 8167)** **Kolleg für Informatik – Tag (SFKZ 8242)**

Reife- und Diplomprüfung 2021/22

Haupttermin September 2022

Aufgabenstellung für die Klausurprüfung

Jahrgang:	6AAIF, 6BAIF, 6AKIF, 6BKIF
Prüfungsgebiet:	Programmieren und Software Engineering
Prüfungstag:	23. September 2022
Arbeitszeit:	6 Std. (gem. § 6 VO BGBl. II Nr. 8/2022)
Kandidaten/Kandidatinnen:	
Prüfer/Prüferin:	DI Manfred BALLUCH, Martin SCHRUTEK, MSc, Michael SCHLETZ, BEd
Aufgabenblätter:	14 Seiten inkl. Umschlagbogen

Inhaltsübersicht der Einzelaufgaben im Umschlagbogen
(Unterschrift des Prüfers/der Prüferin auf den jeweiligen Aufgabenblättern)

Das versiegelte Kuvert mit der der Aufgabenstellung wurde geöffnet von:

Name: _____ Unterschrift: _____

Datum: _____ Uhrzeit: _____

Zwei Zeugen (Kandidaten/Kandidatinnen)

Name: _____ Unterschrift: _____

Name: _____ Unterschrift: _____

Geprüft: Wien, am

_____
Mag. Heidi Steinwender
Abteilungsvorständin

RS.

_____
Dr. Gerhard Hager
Direktor

Genehmigt:

Wien, am _____

RS.

_____
MinR Mag. Gabriele Winkler Rigler

Klausurprüfung (Fachtheorie) aus Programmieren und Software Engineering

im Haupttermin September 2021/22

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	

Klausurprüfung aus Fachtheorie

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.

Die zu vergebenen Punkte sind bei jeder Aufgabenstellung angeführt. Sie müssen für eine Berücksichtigung der Jahresnote mindestens 30% der Gesamtpunkte erreichen. In Summe sind 46 Punkte zu erreichen.

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.

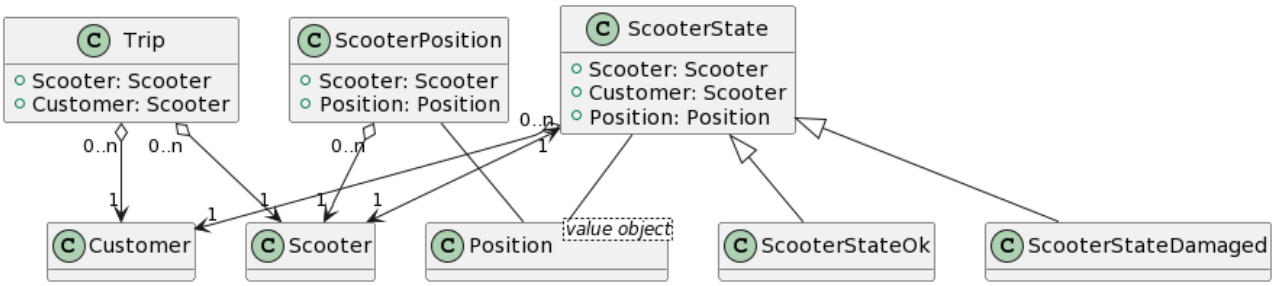
Teilaufgabe 1: Erstellen von EF Core Modelklassen

Ein Verleihsystem für E-Scooter

In letzter Zeit werden E-Scooter gerade im städtischen Bereich immer beliebter. Daher möchten mehrere Anbieter ihre Fahrzeuge auf Verleihbasis anbieten. Ein junges Startup beauftragt Sie mit der Konzeptionierung eines Domain Models, welches folgende Anforderungen abdeckt:

- Jeder Scooter hat eine eindeutige, eingravierte ID, die auch als QR Code ausgelesen werden kann.
- Kunden müssen sich vorab auf einer online Plattform registrieren. Damit die Registrierung gültig ist, muss einmalig ein Ausweisdokument hochgeladen werden. Mitarbeiter:innen prüfen dieses Dokument und schalten den Kunden frei.
- Die Bezahlung erfolgt über einen externen Dienstleister, sie ist nicht Bestandteil des Modelles.
- Die Nutzung der Scooter soll natürlich lückenlos verfolgt werden können. Dies wird durch Übertragung der Position durch integrierte GPS Transmitter.
- Nicht funktionierende Scooter können durch den Kunden mittels der App gemeldet werden. Sie werden aus der Liste verfügbaren Scooter genommen und erst nach der Reparatur werden sie wieder freigegeben.
- Eine klassische Reservierung ist nicht vorgesehen. Die App zeigt einfach die verfügbaren Scooter in der Umgebung an. Der Kunde startet einfach seine Fahrt mittels App und am Ende bestätigt er damit die Rückgabe.

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 *ScooterContext* 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.

Erforderliche Daten für jede Klasse

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 Position (value object) [1 P]

Die übermittelte Position besteht aus Längengrad (decimal), Breitengrad (decimal) und Höhe (decimal). Damit kein JOIN für eine Positionsabfrage gemacht werden muss, ist die Instanz der Position in *ScooterPosition* und *ScooterState* als value object zu definieren.

Klasse Scooter [1 P]

Das Wichtigste ist natürlich die Erfassung der Scooter ID. Diese wird extern vergeben und ist ein 8stelliger Zahlencode. Verwenden Sie daher keine AutoIncrement ID als Primärschlüssel! Neben der Herstellerfirma (string) und der Modellbezeichnung (string) ist die maximal zugelassene Geschwindigkeit in km/h zu speichern.

Klasse Customer [1 P]

Der Kunde registriert sich klassisch mit Vorname (string), Zuname (string), Email Adresse (string) und Handynummer (string). Ein Flag *confirmed* (bool) soll angeben, ob der Kunde anhand eines Ausweisdokumentes verifiziert wurde.

Klasse Trip [1 P]

Startet der Kunde eine Fahrt mit einem Scooter, so wird ein *Trip* angelegt. Dabei ist neben dem Kunden und dem Scooter ein Timestamp mit dem Fahrtbeginn (DateTime) erforderlich. Am Ende der Fahrt wird ein Timestamp mit dem Fahrtende gespeichert. Bei einer gerade stattfindenden Fahrt ist dieser Wert NULL, am Ende wird dieser durch die App auf den Zeitpunkt des Fahrtendes gesetzt.

Klasse ScooterPosition [1 P]

Jeder Scooter übermittelt 1x in der Minute die Position. Dafür wird eine Instanz des definierten value objects *Position* verwendet. Natürlich muss der Timestamp der Meldung (DateTime) erfasst werden.

Klasse ScooterState [3 P]

Die Basisklasse umfasst den Kunden, der den Zustand am Fahrtende bestätigt hat. Ein Timestamp (DateTime) und die Position (Instanz von Position) wird natürlich ebenfalls gespeichert.

Von dieser Klasse erben 2 Subklassen: *ScooterStateOk* hat keine weiteren Felder. *ScooterStateDamaged* hat erweiterte Informationen: Ein Link zu einem Foto, welches der Kunde hochladen kann (string) und eine Textnachricht mit der Beschreibung des Schadens (string)

Unittest CreateDatabaseTest [1 P]

Der vorgegebene Test *CreateDatabaseTest* wird erfolgreich ausgeführt und kann die Datenbank mit den oben angegebenen Klassen erstellen.

In Summe sind für diese Aufgabe 9 Punkte zu erreichen.

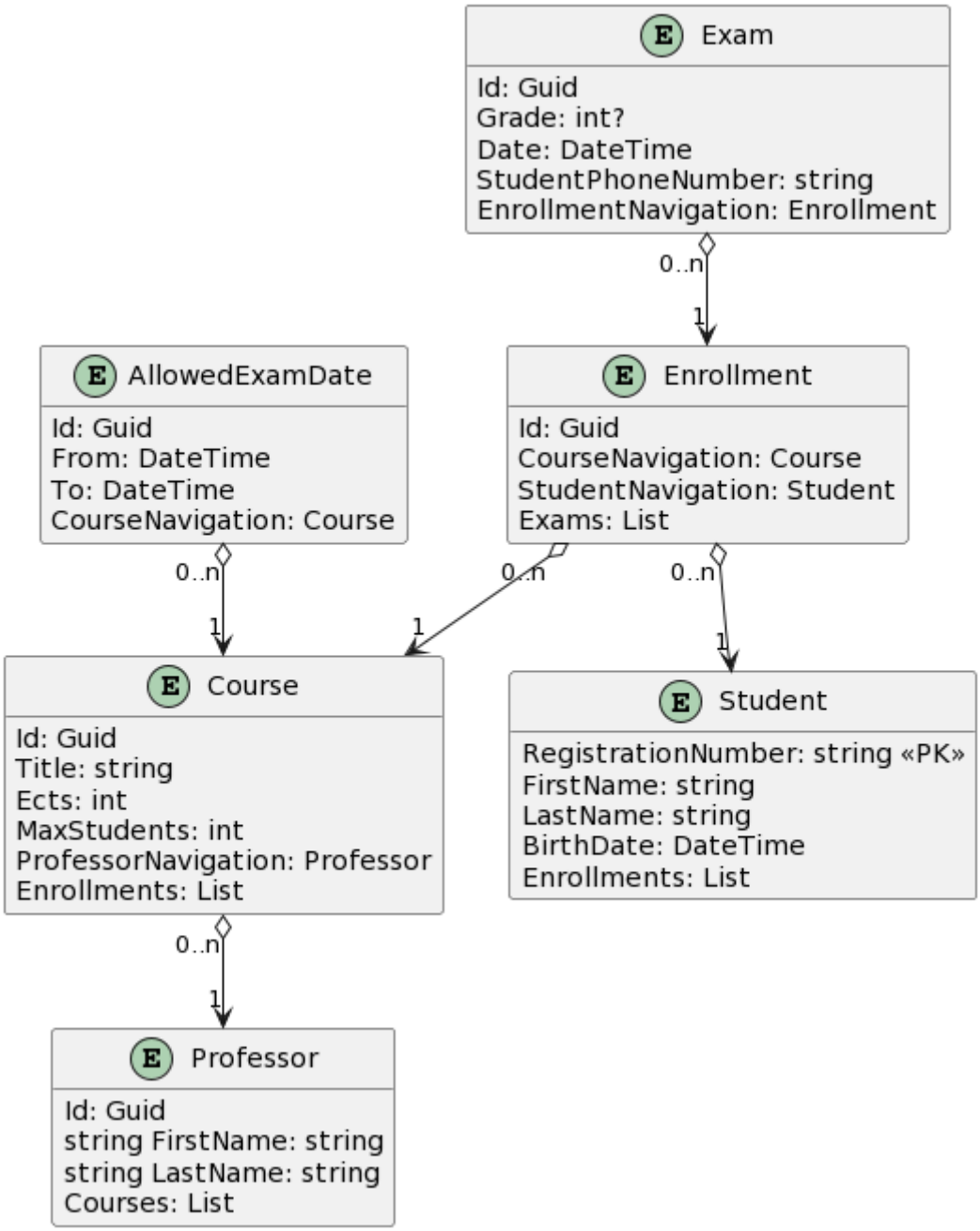
Teilaufgabe 2: Services und Unittests

Eine Universität möchte ihr Verwaltungssystem auf neue Beine stellen. Den Kern des Systems bildet einmal die Verwaltung der Lehrveranstaltungen (*Course*). Für eine Lehrveranstaltung müssen sich Studierende anmelden (*Enrollment*).

Prüfungen sollen auch verwaltet werden. Zu einer Lehrveranstaltung angemeldete Studierende können sich zu einer Prüfung anmelden (*Exam*). Die Leiter:in der Lehrveranstaltung (*Professor*) gibt über *AllowedExams* einen oder mehrere Zeiträume vor, zu denen sich Studierende zu Prüfungen anmelden können. Falls eine Prüfung negativ beurteilt wurde, können sich Studierende nochmals zur Prüfung anmelden.

Das Datenmodell ist für Aufgabe 2 und 3 ident. Dennoch können die beiden Aufgaben völlig unabhängig voneinander bearbeitet werden.

Klassenmodell

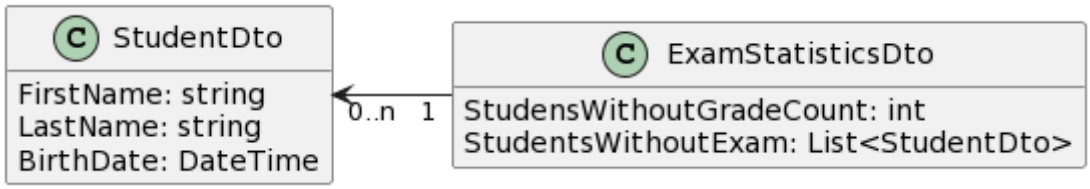


Arbeitsauftrag

Im Projekt *SPG_Fachtheorie.Aufgabe2* ist im Ordner Model eine Service-Klasse *CourseService* vorgegeben. Darin sind folgende 3 Methoden zu implementieren.

ExamStatisticsDto CalcExamStatistics(int courseId)

Diese Methode ermittelt, wie viele Studierende bei der übergebenen Lehrveranstaltung (*courseId*) schon beurteilt wurden. Es ist folgende DTO Klasse zurückzugeben:



Das Property *StudensWithoutGradeCount* gibt an, wie viele angemeldete Studierende noch keine Note für eine angemeldete Prüfung bekommen haben (*Grade* in *Exam* ist null). Studierende, die sich gar nicht für eine Prüfung zu diesem Course angemeldet haben, sollen als Namensliste im Property *StudentsWithoutExam* zurückgegeben werden.

bool SubscribeCourse(string studentRegistrationNumber, int courseId)

Die Methode meldet den übergebenen Studierenden für eine Lehrveranstaltung (Course) an. Dabei sind verschiedene Schritte zu prüfen:

1. Kann der Studierende nicht gefunden werden, soll *false* zurückgegeben werden.
2. Kann der *Course* nicht gefunden werden, soll *false* zurückgegeben werden.
3. Ist der *Student* bereits zu diesem *Course* angemeldet, also hat bereits ein *Enrollment*, soll *false* zurückgegeben werden.
4. Würde die maximale Anzahl an Anmeldungen (*MaxStudents* in *Course*) überschritten werden, soll false zurückgegeben werden.
5. Die Methode liefert *true*, wenn der Datensatz erfolgreich erstellt werden konnte.

Unit Tests

Diese Methode ist durch Unit Tests zu prüfen.

1. Der Test *SubscribeCourse_Invalid_StudentRegistrationNumber* beweist, dass eine nicht vorhandene *studentRegistrationNumber* den Wert *false* liefert.
2. Der Test *SubscribeCourse_Invalid_CourseId* beweist, dass eine nicht vorhandene *courseId* den Wert *false* liefert.
3. Der Test *SubscribeCourse_Student_Already_Enrolled* beweist, dass der Wert *false* zurückgegeben wird, wenn der Studierende bereits zu diesem *Course* angemeldet ist.
4. Der Test *SubscribeCourse_MaxStudent_Exceeded* beweist, dass der Wert *false* zurückgegeben wird, wenn bereits zu viele Studierende zu diesem *Course* angemeldet sind.

5. Der Test *SubscribeCourse_Success* beweist, dass die Methode eine Anmeldung in der Datenbank anlegt. (Es wird *true* geprüft)

bool UnsubscribeCourse(string studentRegistrationNumber, int courseId)

Mit Hilfe dieser Methode sollen sich Studierende zu Lehrveranstaltungen wieder abmelden können. Es soll also der *Enrollment* Datensatz mit den entsprechenden Daten aus der Datenbank gelöscht werden. Um den Datensatz löschen zu können sind folgende Bedingungen zu überprüfen:

1. Ein *Student* kann nur von einem *Course* entfernt werden, wenn er angemeldet ist. Ist er nicht angemeldet, soll *false* zurückgegeben werden.
2. Ein *Student* kann nur von einem *Course* entfernt werden, wenn noch keine Prüfungsanmeldung (*Exam* Datensatz) existiert.
3. Die Methode liefert *true*, wenn der Datensatz erfolgreich gelöscht werden konnte.

Bewertung [17 Punkte]

Implementierung der Methode *CalcExamStatistics()* [4 Punkte]

- *ExamStatisticsDto* wurde vollständig implementiert.
- *StudentDto* wurde vollständig implementiert.
- Die Anzahl der Student:innen, die keine Note haben, wird korrekt befüllt.
- Die Liste jener Student:innen, die zu keiner Prüfung angemeldet sind, wird korrekt befüllt.

Implementierung der Methode *SubscribeCourse()* [5 Punkte]

- Bedingung 1 ist korrekt implementiert (Student kann nicht gefunden werden).
- Bedingung 2 ist korrekt implementiert (Course kann nicht gefunden werden).
- Bedingung 3 ist korrekt implementiert (Student ist bereits angemeldet).
- Bedingung 4 ist korrekt implementiert (maximale Anmeldungen werden berücksichtigt).
- Bedingung 5 ist korrekt implementiert (Erfolgreicher Durchlauf).

Unit Tests zu *SubscribeCourse()* [5 Punkte]

- Unit Test *SubscribeCourse_Invalid_StudentRegistrationNumber* beweist Bedingung 1.
- Unit Test *SubscribeCourse_Invalid_CourseId* beweist Bedingung 2.
- Unit Test *SubscribeCourse_Student_Already_Enrolled* beweist Bedingung 3.
- Unit Test *SubscribeCourse_MaxStudent_Exceeded* beweist Bedingung 4.
- Unit Test *SubscribeCourse_Success* beweist Bedingung 5.

Methode *UnsubscribeCourse()* [3 Punkte]

- Bedingung 1 ist korrekt implementiert (Student ist nicht angemeldet).
- Bedingung 2 ist korrekt implementiert (Prüfungsanmeldung darf nicht existieren).
- Bedingung 3 ist korrekt implementiert (Erfolgreicher Durchlauf).

Teilaufgabe 3: Webapplikation

Das Datenmodell aus Aufgabe 2 soll nun herangezogen werden um eine *Server Side Rendered Web Application* zu erstellen. Die Ausgaben der nachfolgenden Layouts können abweichen, müssen aber alle geforderten Features anbieten. Da es sich um Muster handelt, weichen die konkreten Daten in der zur Verfügung gestellten Musterdatenbank ab.

Das zu implementierende Front-End verfügt über Authentication. Diese ist vollständig implementiert. An dieser Stelle ist nichts zu erledigen. Lediglich der *AuthService* ist in den entsprechenden Controllern oder Pages über Dependency Injection bereitzustellen und zu verwenden.

Die Klasse *AuthService* stellt das *read only property RegistrationNumber* zur Verfügung. Dieses liefert die Matrikelnummer (*RegistrationNumber*) des angemeldeten Studierenden. Damit Sie dieses Service in Ihren Controllern oder Pages nutzen können, müssen Sie die Klasse *AuthService* über Dependency Injection in den Konstruktor einfügen.

Arbeitsauftrag

Für das Universitäts-Datenmodell aus Aufgabe 2 soll nun ein Web-Portal implementiert werden. Für folgende Routen sind *Controller* und *Views* (MVC) ODER Razor-Pages vollständig zu implementieren. Informationen über die Bewertung der einzelnen Features entnehmen Sie aus dem Punkt *Bewertung*.

Seiten (Routen)

/Course/Index

Diese Route stellt eine Liste aller *Course* Datensätze (Lehrveranstaltungen) als Tabelle dar. Die Ausgabe soll nach Titel sortiert erfolgen und kann so aussehen:

Liste der Lehrveranstaltungen

Titel	ECTS	Professor
Algorithmen und Datenstrukturen	4	Schlau, Herbert
Datenmodellierung	4	Weise, Stefan
Einführung in das Programmieren	6	Eifrig, Johannes

/Enrollment/Index

Diese Seite soll die angemeldeten Lehrveranstaltungn des angemeldeten Studierenden ausgeben. Hinweis: Verwenden Sie das Property *RegistrationNumber* aus dem Service *AuthService*, um die angemeldeten Matrikelnummer (RegistrationNumber) auszulesen.

Es sollen die Lehrveranstaltungen in Tabellenform ausgegeben werden. Da es zu einer Lehrveranstaltung (*Course*) auch Prüfungen geben kann, sind die Noten auch auszugeben. Falls eine Prüfung noch nicht beurteilt wurde (*Grade* ist null) erfolgt keine Ausgabe der Note.

Klickt der angemeldete Studierende auf die Note, wird die Detailseite der Prüfung angezeigt (nächster Punkt). Der Link *Zur Prüfung anmelden* führt auf die Anmeldeseite zur Prüfung.

Angemeldete Lehrveranstaltungen

Titel	ECTS	Professor	Noten
Algorithmen und Datenstrukturen	4	Schlau, Herbert	1 (3.2.2022)
Datenmodellierung	4	Weise, Stefan	5 (3.2.2022) 3 (19.4.2022)
Einführung in das Programmieren 1	6	Eifrig, Johannes	5 (4.2.2022)
Einführung in das Programmieren 2	6	Eifrig, Johannes	

Klicken Sie auf die Note, um das Zeugnis abzurufen. Um sich zur Prüfung anzumelden, klicken Sie auf diesen Link:
Zur Prüfung anmelden

/Exam/Details/(id)

Zu jeder Prüfung soll ein Zeugnis abgerufen werden können. Es hat folgenden Aufbau:

Zeugnis für die Lehrveranstaltung Algorithmen und Datenstrukturen

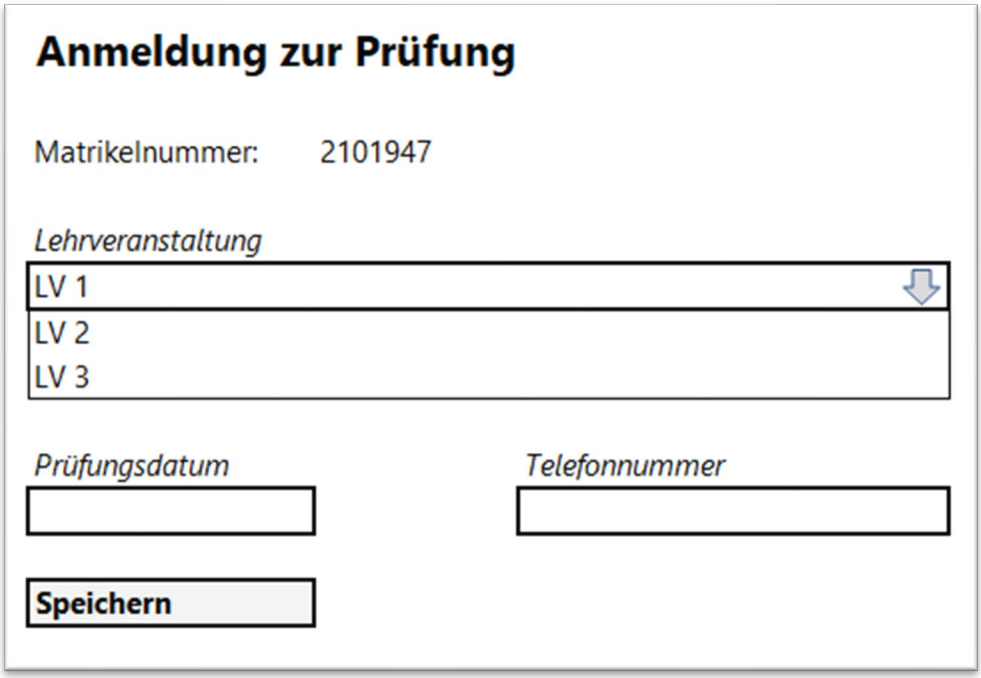
Name der LV:	Algorithmen und Datenstrukturen
Professor:	Schlau, Herbert
Datum der Prüfung:	03.02.2022
ECTS Punkte:	4
Note:	1

Wird die angegebene Exam ID nicht gefunden, so wird 404 not found geliefert. Falls die ID zwar existiert, die Prüfung aber nicht benotet wurde (*Grade* ist null), wird ebenfalls 404 geliefert.

/Exam/Create

Klickt der Studierende auf der Enrollment Seite auf *Zur Prüfung anmelden*, so wird diese Anmeldeseite geöffnet. Auf dieser Seite kann die Lehrveranstaltung als Dropdownliste ausgewählt werden. Es werden alle Lehrveranstaltungen, zu der der Studierende angemeldet ist, angeboten.

Das gewünschte Prüfungsdatum und die Telefonnummer für kurzfristige Informationen werden erfasst und beim Klick auf *Speichern* wird die Anmeldung gespeichert. Zu jeder Lehrveranstaltung (Course) gibt es in der Datenbank Datensätze vom Typ *AllowedExamDate*. Falls ein eingegebenes Datum nicht in einem dieser Zeiträume ist, ist eine entsprechende Fehlermeldung nach dem Klick auf *Speichern* auszugeben. Falls keine Telefonnummer angegeben wurde, ist auch eine entsprechende Fehlermeldung auszugeben.



Bewertung [20 Punkte]

Route /Course/Index [3 Punkte]

- Es wird die Liste der Lehrveranstaltungen in der Datenbank ausgegeben.
- Der Vor- und Zuname des Professors wird wie im Layout angegeben ausgegeben.
- Die Darstellung ist nach dem Titel der Lehrveranstaltung sortiert.

Route /Enrollment/Index [5 Punkte]

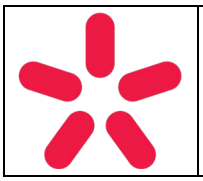
- Das AuthService wird über Dependency Injection korrekt genutzt.
- Es wird die Liste der angemeldeten Lehrveranstaltungen in der Datenbank ausgegeben.
- Die Noten erscheinen wie im Layout angegeben.
- Jede Note besitzt einen Link auf die Entsprechende Detailseite des Exams.
- Der Link *Zur Prüfung anmelden* verweist auf die Create Seite eines Exams.

Route /Exam/Details/(id) [4 Punkte]

- Die Zeugnisdaten werden wie im Layout vorgegeben angezeigt.
- Wird die id nicht gefunden (ist also keine gültige Exam-ID), wird 404 not found geliefert.
- Existiert die Prüfung zwar, hat jedoch noch keine eingetragene Note, wird ebenfalls
- 404 not found geliefert.

Route /Exam/Create [8 Punkte]

- Das AuthService wird über Dependency Injection korrekt genutzt.
- Die Dropdownliste wird mit allen angemeldeten Lehrveranstaltungen befüllt.
- Das Prüfungsdatum wird korrekt als Eingabefeld für Datumswerte dargestellt.
- Die Telefonnummer wird korrekt als Eingabefeld für Texte dargestellt.
- Bei der Eingabe eines ungültigen Datums (das Datum ist nicht im Bereich der Werte in *AllowedExamDates*) wird nach Klicken auf *Speichern* eine entsprechende Fehlermeldung auf dieser Seite dargestellt.
- Bei fehlender Telefonnummer wird nach Klicken auf *Speichern* eine entsprechende Fehlermeldung auf dieser Seite dargestellt.
- Der Button *Speichern* speichert den Datensatz korrekt in der Datenbank.
- Der Button *Speichern* verweist nach dem Speichern auf die Route */Enrollment/Index* zurück.



Haupttermin September 2022

PROGRAMMIEREN UND SOFTWARE ENGINEERING

Aufbaulehrgang für Informatik – Tag (SFKZ 8167)
Kolleg für Informatik – Tag (SFKZ 8242)

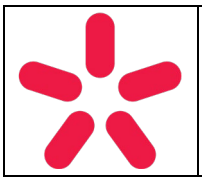
Bewertungsblatt (vom Prüfer auszufüllen)

Für jede erfüllte Teilaufgabe gibt es 1 Punkt. In Summe sind also 46 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:
46 – 41 Punkte: Sehr gut, 40 – 35 Punkte: Gut, 34 – 29 Punkte: Befriedigend, 28 – 24 Punkte: Genügend

Aufgabe 1 (jew. 1 Punkt, 9 in Summe)	Erf.	Nicht erf.
Die Klasse Position bildet die geforderten Informationen korrekt ab.		
Die Klasse Scooter bildet die geforderten Informationen korrekt ab.		
Die Klasse Customer bildet die geforderten Informationen korrekt ab.		
Die Klasse Trip bildet die geforderten Informationen korrekt ab.		
Die Klasse ScooterPosition bildet die geforderten Informationen korrekt ab.		
Die Klasse ScooterState bildet die geforderten Informationen korrekt ab.		
Die Klasse ScooterStateOk bildet die geforderten Informationen korrekt ab.		
Die Klasse ScooterStateDamaged bildet die geforderten Informationen korrekt ab.		
Der Test CreateDatabaseTest erstellt die Datenbank.		

Aufgabe 2 (jew. 1 Punkt, 17 in Summe)	Erf.	Nicht erf.
Implementierung der Methode *CalcExamStatistics()* [4 Punkte]		
ExamStatisticsDto wurde vollständig implementiert.		
StudentDto wurde vollständig implementiert.		
Die Anzahl der Student:innen, die keine Note haben, wird korrekt befüllt.		
Die Liste jener Student:innen, die zu keiner Prüfung angemeldet sind, wird korrekt befüllt.		
Implementierung der Methode *SubscribeCourse()* [5 Punkte]		
Bedingung 1 ist korrekt implementiert (Student kann nicht gefunden werden).		
Bedingung 2 ist korrekt implementiert (Course kann nicht gefunden werden).		
Bedingung 3 ist korrekt implementiert (Student ist bereits angemeldet).		
Bedingung 4 ist korrekt implementiert (maximale Anmeldungen werden berücksichtigt).		
Bedingung 5 ist korrekt implementiert (Erfolgreicher Durchlauf).		
Unit Tests zu *SubscribeCourse()* [5 Punkte]		
Unit Test *SubscribeCourse_Invalid_StudentRegistrationNumber* beweist Bedingung 1.		
Unit Test *SubscribeCourse_Invalid_CourseId* beweist Bedingung 2.		
Unit Test *SubscribeCourse_Student_Already_Enrolled* beweist Bedingung 3.		
Unit Test *SubscribeCourse_MaxStudent_Exceeded* beweist Bedingung 4.		
Unit Test *SubscribeCourse_Success* beweist Bedingung 5.		
Methode *UnsubscribeCourse()* [3 Punkte]		
Bedingung 1 ist korrekt implementiert (Student ist nicht angemeldet).		
Bedingung 2 ist korrekt implementiert (Prüfungsanmeldung darf nicht existieren).		
Bedingung 3 ist korrekt implementiert (Erfolgreicher Durchlauf).		



Haupttermin September 2022

PROGRAMMIEREN UND SOFTWARE ENGINEERING

Aufbaulehrgang für Informatik – Tag (SFKZ 8167)
Kolleg für Informatik – Tag (SFKZ 8242)

Aufgabe 3 (jew. 1 Punkt, 20 in Summe)	Erf.	Nicht erf.
Route /Course/Index [3 Punkte]		
Es wird die Liste der Lehrveranstaltungen in der Datenbank ausgegeben.		
Der Vor- und Zuname des Professors wird wie im Layout angegeben ausgegeben.		
Die Darstellung ist nach dem Titel der Lehrveranstaltung sortiert.		
Route /Enrollment/Index [5 Punkte]		
Das AuthService wird über Dependency Injection korrekt genutzt.		
Es wird die Liste der angemeldeten Lehrveranstaltungen in der Datenbank ausgegeben.		
Die Noten erscheinen wie im Layout angegeben.		
Jede Note besitzt einen Link auf die Entsprechende Detailseite des Exams.		
Der Link *Zur Prüfung anmelden* verweist auf die Create Seite eines Exams.		
Route /Exam/Details/(id) [4 Punkte]		
Die Zeugnisdaten werden wie im Layout vorgegeben angezeigt.		
Wird die id nicht gefunden (ist also keine gültige Exam-ID), wird 404 not found geliefert.		
Existiert die Prüfung zwar, hat jedoch noch keine eingetragene Note, wird ebenfalls		
404 not found geliefert.		
Route /Exam/Create [8 Punkte]		
Das AuthService wird über Dependency Injection korrekt genutzt.		
Die Dropdownliste wird mit allen angemeldeten Lehrveranstaltungen befüllt.		
Das Prüfungsdatum wird korrekt als Eingabefeld für Datumswerte dargestellt.		
Die Telefonnummer wird korrekt als Eingabefeld für Texte dargestellt.		
Bei der Eingabe eines ungültigen Datums (das Datum ist nicht im Bereich der Werte in *AllowedExamDates*) wird nach Klicken auf *Speichern* eine entsprechende Fehlermeldung auf dieser Seite dargestellt.		
Bei fehlender Telefonnummer wird nach Klicken auf *Speichern* eine entsprechende Fehlermeldung auf dieser Seite dargestellt.		
Der Button *Speichern* speichert den Datensatz korrekt in der Datenbank.		
Der Button *Speichern* verweist nach dem Speichern auf die Route */Enrollment/Index* zurück.		