	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 2023/24

Haupttermin September 2024

Aufgabenstellung für die Klausurprüfung

Jahrgang:	6AAIF, 6BAIF, 6CAIF, 6AKIF, 6BKIF
Prüfungsgebiet:	Programmieren und Software Engineering
Prüfungstag:	20. September 2024
Arbeitszeit:	5 Std. (300 Minuten)
Kandidaten/Kandidatinnen:	60
Prüfer/Prüferin:	Mag. Manfred BALLUCH, Martin SCHRUTEK, MSc, Michael SCHLETZ, BEd
Aufgabenblätter:	15 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

_____ RS. _____
Mag. Heidi Steinwender
Abteilungsvorständin

Dr. Gerhard Hager
Direktor

Genehmigt:

Wien, am _____ RS. _____
MinR Mag. Gabriele Winkler Rigler

Klausurprüfung (Fachtheorie) aus Programmieren und Software Engineering

im Haupttermin September 2024

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

Liebe Prüfungskandidatin,
lieber Prüfungskandidat!

Bitte füllen Sie zuerst die nachfolgenden Felder **in Blockschrift** aus, bevor Sie mit der Arbeit beginnen. Trennen Sie dieses Blatt anschließend ab und geben es am Ende ab. Ohne ausgefülltes und abgegebenes Deckblatt kann Ihre Arbeit nicht zugeordnet und gewertet werden!

Maturaaccount (im Startmenü sichtbar):	
Vorname (Blockschrift)	
Zuname (Blockschrift)	
Klasse (Blockschrift)	

Klausurprüfung aus Fachtheorie

Für das 6. Semester des Aufbaulehrganges und des Kollegs am 20. September 2024.

Generelle Hinweise zur Bearbeitung

Die Arbeitszeit für die Bearbeitung der gestellten Aufgaben beträgt **5 Stunden (300 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 2 Stunden für Aufgabe 3. Bei den jeweiligen Aufgaben sehen Sie den Punkteschlüssel. Für eine Einrechnung der Jahresnote sind mindestens 30% der Gesamtpunkte zu erreichen.

Hilfsmittel

In der Datei *P:/SPG_Fachtheorie/SPG_Fachtheorie.sln* befindet sich das Musterprojekt, in dem Sie Ihren Programmcode hineinschreiben. Im Labor steht Visual Studio 2022 mit der .NET Core Version 8 zur Verfügung.

Mit der Software SQLite Studio können Sie sich zur generierten Datenbank verbinden und Werte für Ihre Unittests ablesen. Die Programmdatei befindet sich in *C:/Scratch/SQLiteStudio/SQLiteStudio.exe*.

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

Pfade

Die Solution befindet sich im Ordner *P:/SPG_Fachtheorie*. Sie liegt direkt am Netzlaufwerk, d. h. es ist kein Sichern der Arbeit erforderlich.

Damit das Kompilieren schneller geht, ist der Ausgabepfad der Projekte auf *C:/Scratch/(Projekt)* umgestellt. Das ist zum Auffinden der generierten Datenbank wichtig.

Nullable Reference Types

Das Feature *nullable reference types* wurde in den Projekten aktiviert. Zusätzlich werden Compilerwarnungen als Fehler definiert. Sie können daher das Projekt nicht kompilieren, wenn z. B. ein nullable Warning entsteht. Achten Sie daher bei der Implementierung, dass Sie Nullprüfungen, etc. korrekt durchführen.

SQLite Studio

Zur Betrachtung der Datenbank in *C:/Scratch/Aufgabe1_Test/Debug/net8.0* bzw. *C:/Scratch/Aufgabe2_Test/Debug/net8.0* steht die Software *SQLite Studio* zur Verfügung. Die exe Datei befindet sich

in *C:/Scratch/SPG_Fachtheorie/SQLiteStudio/SQLiteStudio.exe*. Mittels *Database - Add a Database* kann die erzeugte Datenbank (*damages.db* oder *languageweek.db*) geöffnet werden.

Auswählen und Starten der Webapplikation (Visual Studio)

In der Solution gibt es 2 Projekte: *Aufgabe3* (MVC Projekt) und *Aufgabe3RazorPages*. Sie können wählen, ob Sie die Applikation mit MVC oder RazorPages umsetzen wollen. Entfernen Sie das nicht benötigte Projekt aus der Solution und lege das verwendete Projekt als Startup Projekt fest (Rechtsklick auf das Projekt und *Set as Startup Project*).

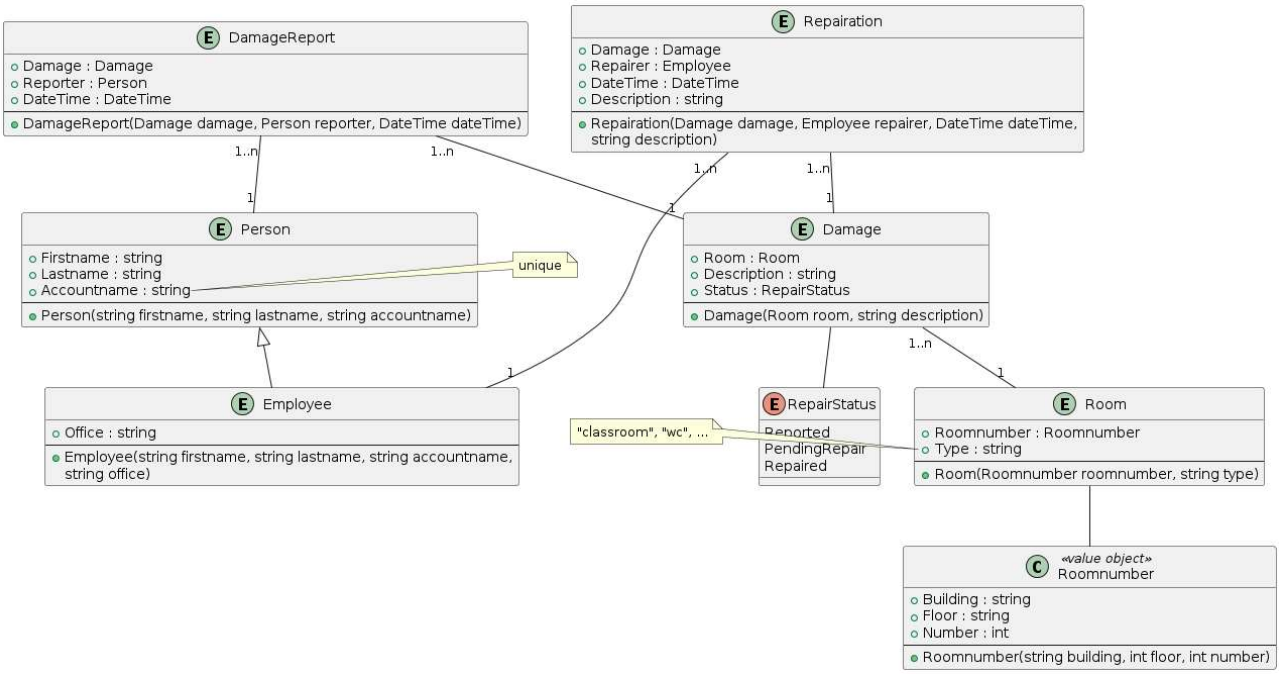
Beim ersten Start erscheint die Frage *Would you like to trust the ASP.NET Core SSL Certificate?* Wähle *Yes* und *Don't ask me again*. Bestätigen Sie den nachfolgenden Dialog zur Zertifikatsinstallation.

Teilaufgabe 1: Erstellen von EF Core Modelklassen

Ein System zum Melden von Schäden

In den letzten Jahren sind Beschädigungen im Haus (leider) immer häufiger ein Thema. Es soll daher ein Meldesystem entwickelt werden, mit dessen Hilfe Studierende, Lehrende und Mitarbeiter/innen der Schule Schäden, die sie entdecken, melden können. Die Applikation soll so aufgebaut werden, dass zuerst ein Raum ausgewählt wird. Danach werden alle Schäden, die schon gemeldet wurden, aufgelistet. Es kann entweder ein neuer Schaden erfasst oder ein bestehender Schaden neu gemeldet werden. So werden Doppelmeldungen vermieden. Deswegen hat ein Schaden im vorgesehenen Modell auch mehrere Meldungen. Die Hausverwaltung hat Einsicht und kann Reparaturen veranlassen. Eine Reparatur besteht manchmal aus mehreren Schritten, da oft Firmen beauftragt werden müssen. Diese Schritte müssen in der Applikation auch dokumentiert werden können.

Modell



Beschreibung der Klassen

Roomnumber (value object)

Die Raumnummern der Schule haben die Form C2.14, etc. Damit die Teile für Abfragen getrennt gespeichert werden, wird ein value object verwendet.

- *Building:* Gebäude ("A", "B", "C" oder "D").
- *Floor:* Stockwerk (1-5), "E", "U" oder "H".
- *Number:* Nummer des Raumes.

Room

Bildet einen Raum in der Schule ab.

- *Roomnumber:* Die gesamte Raumnummer, Referenz auf das *Roomnumber* Objekt.
- *Type:* Frei belegbarer String wie "Stammklasse", "WC", ...

Person

Alle in der Schule beschäftigten Personen wie Studierende, Lehrende und Mitarbeiter/innen.

- *Firstname:* Vorname.
- *Lastname:* Nachname.
- *Accountname:* Accountname im Active Directory der Schule.

Employee (erbt von Person)

- *Office:* Mitarbeiter/innen haben zusätzlich ein Büro. Hier wird die Raumnummer als String gespeichert.

RepairStatus (enum)

Der Status eines gemeldeten Schadens wird als enum gespeichert. Er kann mehrere Zustände haben:

- *Reported*: Der Schaden wurde gemeldet, die Hausverwaltung hat aber noch nicht reagiert.
- *PendingRepair*: Die Hausverwaltung hat den Schaden bestätigt und Reparaturmaßnahmen eingeleitet.
- *Repaired*: Der Schaden wurde behoben.

Damage

Wird ein Schaden gemeldet, wird ein Datensatz in der Tabelle *Damage* angelegt.

- *Room:* In welchem Raum wurde die Beschädigung festgestellt?
- *Description:* Eine kurze Beschreibung (Tisch verschmutzt, Steckdose funktioniert nicht, etc.).
- *Status:* Der Status der Reparatur wie in der enum *RepairStatus* angegeben.

DamageReport

Da ein Schaden mehreren Studierenden auffällt, kann er mehrmals gemeldet werden. So entsteht ein Überblick, wie lange der Schaden schon besteht.

- *Damage:* Verweis auf den Schaden.
- *Reporter:* Die Person, die die Beobachtung macht.
- *DateTime:* Der Zeitpunkt der Beobachtung.

Repairation

Für einen Schaden können mehrere Reparaturschritte notwendig sein. Oft müssen Teile erst bestellt bzw. externe Firmen beauftragt werden. Daher wird in Repairation ein "Log" geführt, mit dessen Hilfe die Hausverwaltung die einzelnen Schritte dokumentieren kann.

- *Damage:* Verweis auf den Schaden.
- *Repairer:* Mitarbeiter/in (Employee), der den Schaden bearbeitet.
- *DateTime:* Datum und Uhrzeit des Eintrages.
- *Description:* Beschreibung (z. B. "Firma X beauftragt", "Schaden behoben", ...)

Arbeitsauftrag

Erstellung der Modelklassen

Im Projekt *SPG_Fachtheorie.Aufgabe1* befinden sich im Ordner *Model* leere Klassendefinitionen. Bilden Sie jede Klasse gemäß dem UML Diagramm ab, sodass EF Core diese persistieren kann. Beachten Sie folgendes:

- Wählen Sie selbst notwendige Primary keys.
- Definieren Sie Stringfelder mit vernünftigen Maximallängen (z. B. 255 Zeichen für Namen, etc.).
- Roomnumber ist ein *value object*. Stellen Sie durch Ihre Definition sicher, dass kein Mapping dieser Klasse in eine eigene Datenbanktabelle durchgeführt wird.
- Das Feld *Accountname* in *Person* muss unique sein. Stellen Sie dies durch eine geeignete Konfiguration sicher.
- Legen Sie Konstruktoren mit allen erforderlichen Feldern an. Erstellen Sie die für EF Core notwendigen default Konstruktoren als *protected*.
- Wird ein neuer Schaden angelegt, hat dieser den Status *Reported*, der im Konstruktor gesetzt wird.
- Das Feld *Status* in *Damage* ist ein enum Feld. Speichern Sie dieses Feld als String in der Datenbank. Stellen Sie dies durch geeignete Konfiguraton sicher.
- Implementieren Sie die Vererbung korrekt, sodass eine (1) Tabelle *Person* entsteht.
- Legen Sie die erforderlichen DB Sets im Datenbankcontext an.

Verfassen von Tests

Im Projekt *SPG_Fachtheorie.Aufgabe1.Test* ist in *Aufgabe1Test.cs* der Test *CreateDatabaseTest* vorgegeben. Er muss erfolgreich durchlaufen und die Datenbank erzeugen. Sie können die erzeugte Datenbank in *C:/Scratch/Aufgabe1_Test/Debug/net8.0/damages.db* in SQLite Studio öffnen.

Implementieren Sie folgende Tests selbst, indem Sie die minimalen Daten in die (leere) Datenbank schreiben. Leeren Sie immer vor dem *Assert* die nachverfolgten Objekte mittels *db.ChangeTracker.Clear()*.

- Der Test *AddEmployeeSuccessTest* beweist, dass Sie einen Mitarbeiter (Employee) in die Datenbank einfügen können. Prüfen Sie im Assert, ob der angegebene Accountname auch korrekt gespeichert wurde.
- Der Test *AddDamageWithReportSuccessTest* beweist, dass Sie einen Schaden samt Meldung (Report) anlegen können. Legen Sie hierfür einen Schaden (Damage) samt Raum, Employee und Person an und fügen Sie eine Meldung ein. Stellen Sie im Assert sicher, dass der gespeicherte Schaden eine Meldung hat.
- Der Test *AddRepairationSuccessTest* beweist, dass Sie eine Reparatur für einen Schaden anlegen können. Legen Sie dafür einen Schaden samt der notwendigen Daten an und weisen einen Reparatureintrag zu. Prüfen Sie im Assert, ob die Beschreibung der Reparatur in der Datenbank vorhanden ist.

Bewertung (24P, 38.7% der Gesamtpunkte)

Jedes der folgenden Kriterien wird mit 1 Punkt bewertet.

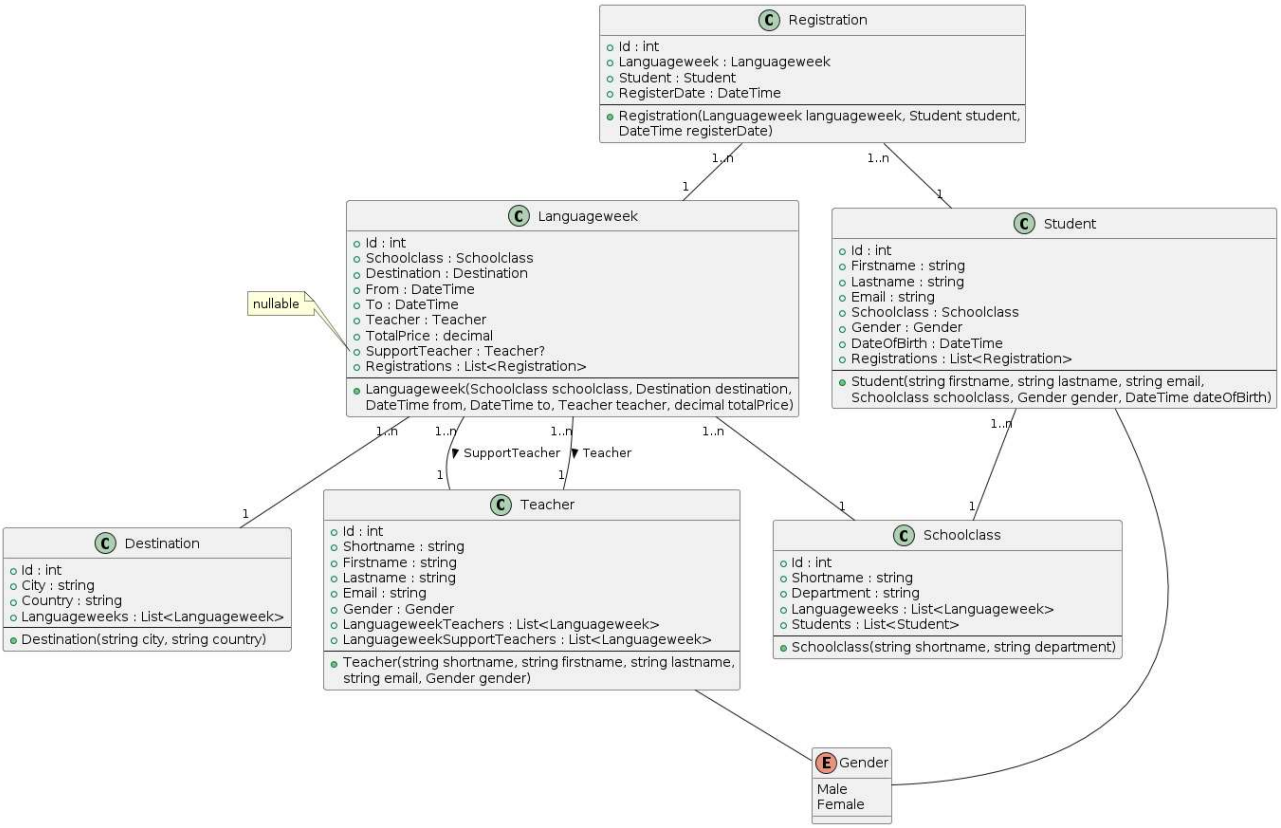
- Die Stringfelder verwenden sinnvolle Längenbegrenzungen.
- Die Klasse *Roomnumber* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.
- Die Klasse *Room* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.

- Die Klasse *Room* besitzt ein korrekt konfiguriertes value object *Address*.
- Die Klasse *Room* wurde korrekt im DbContext registriert.
- Die Klasse *Person* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.
- Die Klasse *Person* wurde korrekt im DbContext registriert.
- Das Feld *Accountname* in *Person* ist als unique definiert.
- Die Klasse *Employee* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.
- Die Klasse *Employee* wurde korrekt im DbContext registriert.
- Die Klasse *Employee* erbt korrekt von der Klasse *Person*.
- Die Klasse *Damage* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.
- Die Klasse *Damage* wurde korrekt im DbContext registriert.
- Das Feld *RepairStatus* in *Damage* wird in der Datenbank als String gespeichert.
- Die Klasse *DamageReport* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.
- Die Klasse *DamageReport* wurde korrekt im DbContext registriert.
- Die Klasse *Repairation* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.
- Die Klasse *Repairation* wurde korrekt im DbContext registriert.
- Der Test *AddEmployeeSuccessTest* ist korrekt aufgebaut.
- Der Test *AddEmployeeSuccessTest* läuft erfolgreich durch.
- Der Test *AddDamageWithReportSuccessTest* ist korrekt aufgebaut.
- Der Test *AddDamageWithReportSuccessTest* läuft erfolgreich durch.
- Der Test *AddRepairationSuccessTest* ist korrekt aufgebaut.
- Der Test *AddRepairationSuccessTest* läuft erfolgreich durch.

Teilaufgabe 2: Services und Unittests

Für die Verwaltung von Sprachwochen an der Schule soll eine Applikation erstellt werden. In den höheren Abteilungen fahren Klassen auf Sprachreise. Der Aufenthalt dauert meist 1 Woche und die Ziele sind meist Städte in Großbritannien, Irland oder Malta. Eine Sprachreise wird zuerst für eine bestimmte Klasse geplant. Danach können sich die Schüler/innen dafür anmelden, da keine Verpflichtung zur Teilnahme besteht. Bei einer Sprachreise gibt es eine/n Leiter/in. Zur Unterstützung fährt ein/e Begleitlehrer/in mit.

Folgendes Klassendiagramm ist als Domain Model bereits vorhanden und kann verwendet werden.



Beschreibung der Klassen

Schoolclass

Die Schulklasse, für die die Sprachreise geplant wird (z. B: 4CHIF).

- *Id:* Auto increment ID
- *Shortname:* Name der Klasse (z. B: "4AHIF")
- *Department:* Abteilung (z. B: "HIF")

Student

Datensatz für den einzelnen Schüler / die Schülerin:

- *Id:* Auto increment ID
- *Firstname:* Vorname
- *Lastname:* Zuname
- *Email:* E-Mail Adresse
- *Schoolclass:* Klasse, die besucht wird (z. B. "4CHIF")
- *Gender:* Geschlecht (enum Feld)
- *DateOfBirth:* Geburtsdatum

Teacher

Datensatz für den einzelnen Lehrer / der Lehrerin:

- *Id:* Auto increment ID
- *Shortname:* Kürzel im Stundenplan (z. B. "SZ")
- *Firstname:* Vorname
- *Lastname:* Zuname
- *Email:* E-Mail Adresse
- *Gender:* Geschlecht (enum Feld)

Languageweek

Wenn eine Sprachreise geplant wird, wird eine Instanz von *Languageweek* angelegt. Sie umfasst die Basisinformationen, damit sich später die Schüler/innen anmelden können.

- *Id:* Auto increment ID
- *Schoolclass:* Klasse, für die die Reise geplant wird (z. B: "4AHIF")
- *Destination:* Wohin geht die Reise? Referenz auf das Destination Objekt.
- *From:* Erster Tag der Sprachreise. Datums/Zeitwert mit 0:00 als Zeitkomponente.
- *To:* Letzter Tag der Reise. Achtung: Es ist ein Datums/Zeitwert mit 0:00 als Zeitkomponente. "2024-06-20" bedeutet, dass der 20. 6. **nicht** mehr inkludiert wird. Die Reise dauert bis inklusive 19.6. (also 20.6. um 0:00).
- *Teacher:* Leiter/in der Veranstaltung. Referenz auf ein *Teacher* Objekt.
- *TotalPrice:* Gesamtpreis in Euro.
- *SupportTeacher:* Begleitlehrer/in. **Achtung: nullable**. Diese Information wird meist später eingetragen, daher bleibt das Feld am Anfang noch leer.

Registration

Bildet die Anmeldung einer Schülerin / eines Schülers zur Sprachreise ab.

- *Id:* Auto increment ID
- *Languageweek:* Verweis auf das zugehörige Languageweek Objekt.
- *Student:* Verweis auf das zugehörige Student Objekt.
- *RegisterDate:* Datum der Anmeldung (Datums/Zeitwert)

Destination

Stadt, die als Ziel der Sprachreise dienen kann.

- *Id:* Auto increment ID
- *City:* Name der Stadt (z. B. "Dublin")
- *Country:* Land (z. B. "Irland")

Arbeitsauftrag

Implementierung von Servicemethoden

Im Projekt *SPG_Fachtheorie.Aufgabe2* befindet sich die Klasse *Services/LanguageweekService.cs*. Es sind 2 Methoden zu implementieren:

List<LanguageweekStatisticsDto> ContingentStatistics CalculateStatistics()

Diese Methode soll ermitteln, wie viele Anmeldungen (Anzahl der Registrations) für die einzelnen Sprachwochen eingetragen sind. Dabei wird jede Sprachwoche auf die folgende Klasse *LanguageweekStatisticsDto* abgebildet:

```
,-------------------------.
|LanguageweekStatisticsDto |
|-------------------------|
|Id : int                  |
|From : int                |
|To : int                  |
|DestinationCity : int     |
|TeacherEmail : int        |
|Participants : int        |
|StudentsInClass: int      |
`-------------------------'
```

Die Felder *Id*, *From*, *To* kommen aus dem *Languageweek* Objekt. *DestinationCity* ist der *City* Eintrag der *Destination* Referenz. *TeacherEmail* ist der *Email* Eintrag der *Teacher* Referenz. *Participants* gibt die Anzahl der angemeldeten Schüler/innen an. Die Anmeldungen sind in der Liste *Registrations* der *Languageweek* Klasse zu finden. *StudentsInClass* gibt an, wie viele Schüler/innen die Klasse hat, für die diese Sprachwoche geplant ist. Verwenden Sie das Property *Schoolclass* von *Languageweek* und zählen Sie, wie viele Studierende insgesamt diese Klasse besuchen (DbSet *Students*).

Im Projekt *SPG_Fachtheorie.Aufgabe2.Test* ist der Test *CalculateStatisticsTest* bereits vorgegeben, der die Richtigkeit Ihrer Methode prüft.

void AssignSupportTeacher(int languageweekId, int supportTeacherId)

Diese Methode soll einen Begleitlehrer / eine Begleitlehrerin zuweisen. Dabei sind folgende Randbedingungen zu prüfen:

- Wird die übergebene ID der Sprachwoche (*languageweekId*) nicht gefunden, wird eine *LanguageweekServiceException* mit dem Inhalt *Languageweek not found.* geworfen.
- Wird die übergebene ID des Begleitlehrers (*supportTeacherId*) nicht bei den Lehrern gefunden, wird eine *LanguageweekServiceException* mit dem Inhalt *Teacher not found.* geworfen.
- Ein Begleitlehrer darf nicht schon als Leiter datenbankweit eingetragen werden. Prüfen Sie daher, ob es Sprachwochen gibt, wo das *Teacher* Property diesem Lehrer zugewiesen wurde. Werfen Sie in diesem Fall eine *LanguageweekServiceException* mit dem Text *Teacher is already assigned as leader.*
- Gibt es weibliche Teilnehmerinnen (mindesens ein Eintrag der *Registrations* Liste der *Languageweek* hat den Gender Eintrag *Gender.Female*), so muss entweder der Leiter der Sprachwoche (Property *Teacher*) **oder** der neu einzutragende Begleitlehrer weiblich sein (*Gender.Female*). Falls nicht, werfen Sie eine *LanguageweekServiceException* mit dem Text *Only male teachers are not allowed if there are female participants.*. Wurden alle Bedingungen

erfolgreich geprüft, soll das Property *SupportTeacher* auf den entsprechenden Lehrer gesetzt und in der Datenbank gespeichert werden.

Wichtig: Laden Sie mit *Include* bzw. *ThenInclude* alle Informationen aus der Datenbank, wenn Sie auf Navigations zugreifen!

Testen der Methode AssignSupportTeacher

Schreiben Sie im Projekt *SPG_Fachtheorie.Aufgabe2.Test* in die Klasse *LanguageweekServiceTests* Unittests, die die Korrektheit von *AssignSupportTeacher* prüfen.
Mit *GetEmptyDbContext()* können Sie einen Datenbankcontext zu einer leeren Datenbank erstellen. Befüllen Sie die Datenbank selbst mit minimalen Musterdaten, sodass Sie das Methodenverhalten prüfen können. Verwenden Sie *ChangeTracker.Clear()* des Datenbankcontext, um nach dem Einfügen der Musterdaten und nach Aufrufen der Servicemethode den Changetracker zu leeren.

Mit folgendem Codesnippet können Sie prüfen, ob eine Methode eine bestimmte Exception wirft und eine korrekte Meldung liefert:

```
var ex = Assert.Throws<EventServiceException>(() => MethodToCheck());
Assert.True(ex.Message == "This is the Message");
```

Es sind 3 Tests zu verfassen:

- **AssignSupportTeacherSuccessTest** prüft, ob die Methode einen Begleitlehrer in der Datenbank speichert, wenn alle Bedingungen eingehalten werden.
- **AssignSupportTeacherThrowsExceptionIfTeacherIsAlreadyAssignedTest** prüft, ob die Methode eine *LanguageweekServiceException* mit dem Text *Teacher is already assigned as leader.* wirft, wenn der übergebene Lehrer bereits Leiter einer Sprachwoche ist.
- **AssignSupportTeacherThrowsExceptionIfNoFemaleTeacherTest** prüft, ob die Methode eine *LanguageweekServiceException* mit dem Text *Only male teachers are not allowed if there are female participants.* wirft, wenn bei einer Sprachwoche mit weiblichen Teilnehmerinnen und einem männlichen Leiter auch ein männlicher Begleitlehrer zugewiesen wird.

Bewertung (14P, 22.6% der Gesamtpunkte)

Jedes der folgenden Kriterien wird mit 1 Punkt bewertet.

- Die Methode *CalculateStatistics* ermittelt die Anzahl der Teilnehmer/innen pro Sprachwoche korrekt.
- Die Methode *CalculateStatistics* ermittelt die Anzahl der Schüler/innen der Klasse pro Sprachwoche korrekt.
- Die Methode *CalculateStatistics* liest die anderen Properties der DTO Klasse korrekt aus.
- Die Methode *CalculateStatistics* verwendet LINQ und keine imperativen Konstrukte wie Schleifen, ...
- Die Methode *AssignSupportTeacher* prüft korrekt, ob die übergebene Languageweek und Teacher ID vorhanden ist.
- Die Methode *AssignSupportTeacher* prüft korrekt, ob der übergebene Begleitlehrer nicht schon wo als Leiter eingetragen ist.
- Die Methode *AssignSupportTeacher* prüft korrekt, ob bei vorhandenen weiblichen Teilnehmerinnen auch Lehrer oder Begleitlehrer weiblich sind.
- Die Methode *AssignSupportTeacher* weist den Begleitlehrer korrekt zu und speichert den Datensatz.
- Der Unittest *AssignSupportTeacherSuccessTest* hat den korrekten Aufbau (arrange, act, assert).
- Der Unittest *AssignSupportTeacherSuccessTest* läuft erfolgreich durch.
- Der Unittest *AssignSupportTeacherThrowsExceptionIfTeacherIsAlreadyAssignedTest* hat den korrekten Aufbau (arrange, act, assert).
- Der Unittest *AssignSupportTeacherThrowsExceptionIfTeacherIsAlreadyAssignedTest* läuft erfolgreich durch.
- Der Unittest *AssignSupportTeacherThrowsExceptionIfNoFemaleTeacherTest* hat den korrekten Aufbau (arrange, act, assert).
- Der Unittest *AssignSupportTeacherThrowsExceptionIfNoFemaleTeacherTest* läuft erfolgreich durch.

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.

Arbeitsauftrag

Implementieren Sie die folgenden Seiten im Projekt *SPG_Fachtheorie.Aufgabe3*, falls Sie mit MVC arbeiten möchten. Verwenden Sie *SPG_Fachtheorie.Aufgabe3.RazorPages*, falls Sie mit Razor Pages arbeiten möchten. Löschen Sie das nicht benötigte Projekt aus der Solution und legen Ihr gewünschtes Webprojekt als Startprojekt fest.

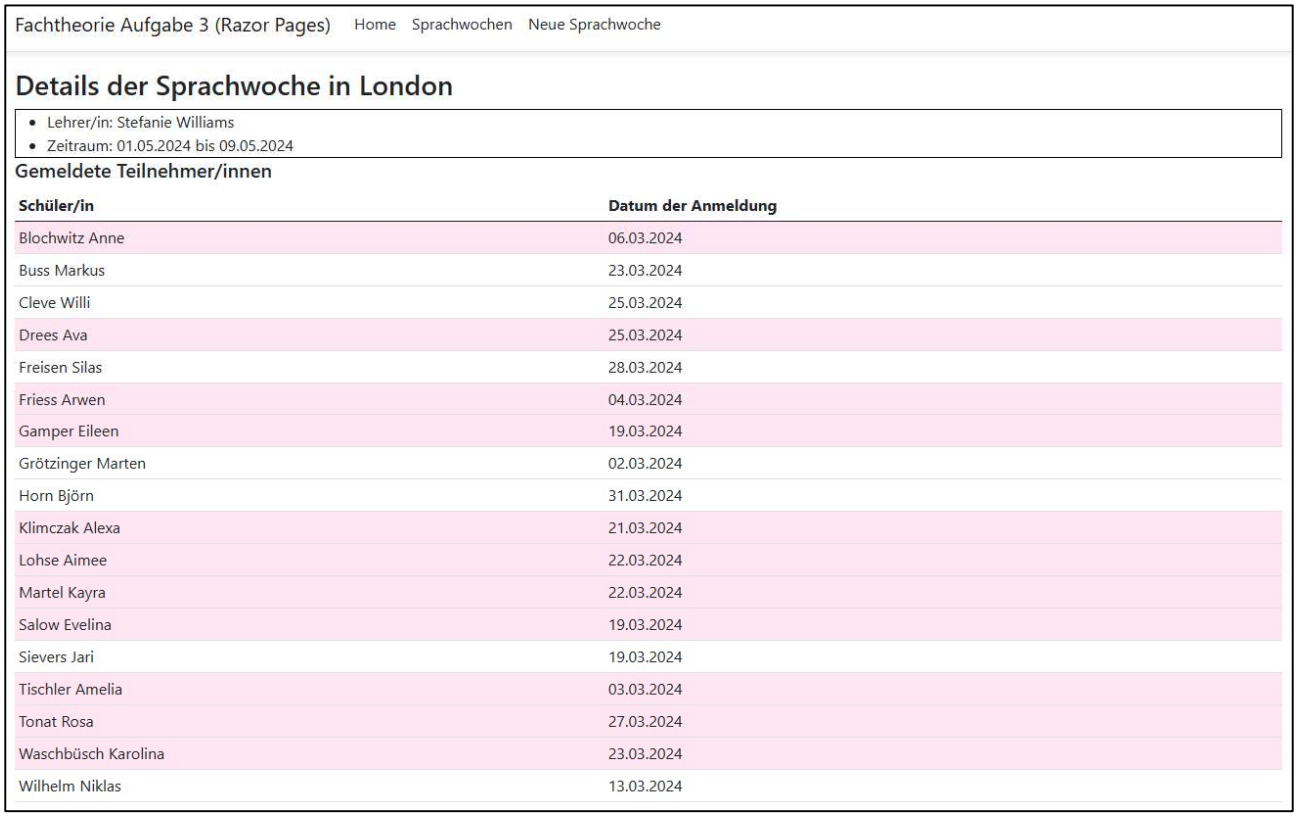
Seite /Languageweeks/Index

Der Menüpunkt *Sprachwochen* in der Navigation verweist auf diese Seite. Diese Seite soll alle eingetragenen Sprachwochen ausgeben. Ordnen Sie die Sprachwochen nach dem Von Datum (Property *From*). Pro Sprachwoche soll das Ziel (*City* und *Country* aus *Destination*), das von und bis Datum (*From* und *To*) sowie den/die Leiter/in (Properties *Firstname*, *Lastname* und *Email* aus *Teacher*) ausgegeben werden. Formatieren Sie Datumswerte mit dem Formatstring *dd.MM.yyyy*. Für jede Sprachwoche soll ein Link angeboten werden, der auf die Detailseite der Sprachwoche verweist (siehe nächster Punkt). Verwenden Sie als Übergabeparameter die auto increment ID (Property *Id*).

Der folgende Screenshot zeigt die gewünschte Ausgabe. Es handelt sich um Echtdaten, d. h. bei Ihnen soll die Ausgabe auch so aussehen.

Seite /Languageweeks/Details/(id)

Diese Seite soll alle Teilnehmer/innen der angegebenen Sprachwoche darstellen. Die ID wird als Routingparameter übergeben. Geben Sie in der Überschrift die Stadt aus (*City* in *Destination*). Geben Sie als Header die Informationen zur Sprachwoche (Leiter/in, von und bis) aus. Geben Sie alle Teilnehmer/innen mit Name (Nach- und Vorname) sowie das Datum der Anmeldung aus. Sortieren Sie die Liste nach Nachname, dann nach Vorname. Formatieren Sie Datumswerte mit dem Formatstring *dd.MM.yyyy*. Weibliche Studierende sollen farblich unterlegt werden.



Seite der Sprachwoche vom 01.05.2024 - 09.05.2024, http://localhost:5000/Languageweeks/Details/1

Seite /Languageweeks/Add

Der Menüpunkt *Neue Sprachwoche* in der Navigation verweist auf diese Seite. Diese Seite soll das Eintragen einer neuen Sprachwoche ermöglichen. Ein Dropdownfeld listet alle im System gespeicherten Klassen sortiert nach *Shortname* auf. Ein Dropdownfeld listet alle im System gespeicherten Ziele (Destination) sortiert nach *City* auf. Ein Dropdownfeld listet alle im System gespeicherten Lehrer/innen sortiert nach *Lastname* und dann nach *Firstname* auf. Im Feld soll ein String bestehend aus Nachname und Vorname angezeigt werden. Für von und bis sollen Eingabefelder vom Typ *date* zur Verfügung stehen. Für *TotalPrice* soll ein Eingabefeld vom Typ *number* zur Verfügung stehen. Stellen Sie durch Validierung sicher, dass

- das von Datum kleiner als das bis Datum ist.
- der Preis zwischen 1 und 2000 Euro liegt.

Nach dem Klick auf *Speichern* soll eine neue Sprachwoche mit den entsprechenden Daten erstellt und in der Datenbank gespeichert werden. Geben Sie Validierungsfehler auf der Seite aus. Konnte die Sprachwoche gespeichert werden, soll mit *RedirectToPage* auf die Seite */Languageweeks/Index* verwiesen werden.

Bewertung (24P, 38.7% der Gesamtpunkte)

- Die Seite */Languageweeks/Index* besitzt eine korrekte Dependency Injection der Datenbank.
- Die Seite */Languageweeks/Index* fragt die benötigten Informationen aus der Datenbank korrekt ab.
- Die Seite */Languageweeks/Index* zeigt die geforderten Daten der Sprachwoche an.
- Die Seite */Languageweeks/Index* sortiert die Sprachwochen korrekt.
- Die Seite */Languageweeks/Index* formatiert die Datumswerte korrekt.
- Die Seite */Languageweeks/Index* verweist korrekt auf die Detailseite der Sprachwoche.
- Die Seite */Languageweeks/Details* besitzt eine korrekte Dependency Injection der Datenbank.
- Die Seite */Languageweeks/Details* besitzt einen Routingparameter für die Sprachwochen ID.
- Die Seite */Languageweeks/Details* fragt die benötigten Informationen aus der Datenbank korrekt ab.
- Die Seite */Languageweeks/Details* zeigt die geforderten Daten der Sprachwoche an.
- Die Seite */Languageweeks/Details* zeigt die geforderten Daten der Teilnehmer/innen an.
- Die Seite */Languageweeks/Details* sortiert die Teilnehmer/innen korrekt.
- Die Seite */Languageweeks/Details* hebt die weiblichen Teilnehmerinnen hervor.
- Die Seite */Languageweeks/Details* formatiert die Datumswerte korrekt.
- Die Seite */Languageweeks/Add* besitzt eine korrekte Dependency Injection der Datenbank.
- Die Seite */Languageweeks/Add* fragt die benötigten Informationen aus der Datenbank korrekt ab.
- Die Seite */Languageweeks/Add* zeigt ein Dropdownfeld mit allen Klassen korrekt an.
- Die Seite */Languageweeks/Add* zeigt ein Dropdownfeld mit allen Zielen korrekt an.
- Die Seite */Languageweeks/Add* zeigt ein Dropdownfeld mit allen Lehrer/innen korrekt an.
- Die Seite */Languageweeks/Add* zeigt die Eingabefelder für von, bis und Preis korrekt an.
- Die Seite */Languageweeks/Add* validiert die Eingaben.
- Die Seite */Languageweeks/Add* zeigt bei einer ungültigen Eingabe die Seite wieder an.
- Die Seite */Languageweeks/Add* speichert die Sprachwoche korrekt in der Datenbank.
- Die Seite */Languageweeks/Add* verweist nach dem Speichern auf die Indexseite.

Haupttermin September 2024

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 62 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:
62 – 55 Punkte: Sehr gut, 54 – 47 Punkte: Gut, 46 – 39 Punkte: Befriedigend, 38 – 32 Punkte: Genügend

Aufgabe 1 (jew. 1 Punkt, 24 in Summe)	Erf.	Nicht erf.
Die Stringfelder verwenden sinnvolle Längenbegrenzungen.		
Die Klasse *Roomnumber* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.		
Die Klasse *Room* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.		
Die Klasse *Room* besitzt ein korrekt konfiguriertes value object *Address*.		
Die Klasse *Room* wurde korrekt im DbContext registriert.		
Die Klasse *Person* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.		
Die Klasse *Person* wurde korrekt im DbContext registriert.		
Das Feld *Accountname* in *Person* ist als unique definiert.		
Die Klasse *Employee* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.		
Die Klasse *Employee* wurde korrekt im DbContext registriert.		
Die Klasse *Employee* erbt korrekt von der Klasse *Person*.		
Die Klasse *Damage* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.		
Die Klasse *Damage* wurde korrekt im DbContext registriert.		
Das Feld *RepairStatus* in *Damage* wird in der Datenbank als String gespeichert.		
Die Klasse *DamageReport* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.		
Die Klasse *DamageReport* wurde korrekt im DbContext registriert.		
Die Klasse *Repairation* beinhaltet die im UML Diagramm abgebildeten Felder und korrekte public bzw. protected Konstruktoren.		
Die Klasse *Repairation* wurde korrekt im DbContext registriert.		
Der Test *AddEmployeeSuccessTest* ist korrekt aufgebaut.		
Der Test *AddEmployeeSuccessTest* läuft erfolgreich durch.		
Der Test *AddDamageWithReportSuccessTest* ist korrekt aufgebaut.		
Der Test *AddDamageWithReportSuccessTest* läuft erfolgreich durch.		
Der Test *AddRepairationSuccessTest* ist korrekt aufgebaut.		
Der Test *AddRepairationSuccessTest* läuft erfolgreich durch.		

Aufgabe 2 (jew. 1 Punkt, 14 in Summe)	Erf.	Nicht erf.
Die Methode *CalculateStatistics* ermittelt die Anzahl der Teilnehmer/innen pro Sprachwoche korrekt.		
Die Methode *CalculateStatistics* ermittelt die Anzahl der Schüler/innen der Klasse pro Sprachwoche korrekt.		
Die Methode *CalculateStatistics* liest die anderen Properties der DTO Klasse korrekt aus.		
Die Methode *CalculateStatistics* verwendet LINQ und keine imperativen Konstrukte wie Schleifen, …		

Die Methode *AssignSupportTeacher* prüft korrekt, ob die übergebene Languageweek und Teacher ID vorhanden ist.		
Die Methode *AssignSupportTeacher* prüft korrekt, ob der übergebene Begleitlehrer nicht schon wo als Leiter eingetragen ist.		
Die Methode *AssignSupportTeacher* prüft korrekt, ob bei vorhandenen weiblichen Teilnehmerinnen auch Lehrer oder Begleitlehrer weiblich sind.		
Die Methode *AssignSupportTeacher* weist den Begleitlehrer korrekt zu und speichert den Datensatz.		
Der Unittest *AssignSupportTeacherSuccessTest* hat den korrekten Aufbau (arrange, act, assert).		
Der Unittest *AssignSupportTeacherSuccessTest* läuft erfolgreich durch.		
Der Unittest *AssignSupportTeacherThrowsExceptionIfTeacherIsAlreadyAssignedTest* hat den korrekten Aufbau (arrange, act, assert).		
Der Unittest *AssignSupportTeacherThrowsExceptionIfTeacherIsAlreadyAssignedTest* läuft erfolgreich durch.		
Der Unittest *AssignSupportTeacherThrowsExceptionIfNoFemaleTeacherTest* hat den korrekten Aufbau (arrange, act, assert).		
Der Unittest *AssignSupportTeacherThrowsExceptionIfNoFemaleTeacherTest* läuft erfolgreich durch.		

Aufgabe 3 (jew. 1 Punkt, 24 in Summe)	Erf.	Nicht erf.
Die Seite */Languageweeks/Index* besitzt eine korrekte Dependency Injection der Datenbank.		
Die Seite */Languageweeks/Index* fragt die benötigten Informationen aus der Datenbank korrekt ab.		
Die Seite */Languageweeks/Index* zeigt die geforderten Daten der Sprachwoche an.		
Die Seite */Languageweeks/Index* sortiert die Sprachwochen korrekt.		
Die Seite */Languageweeks/Index* formatiert die Datumswerte korrekt.		
Die Seite */Languageweeks/Index* verweist korrekt auf die Detailseite der Sprachwoche.		
Die Seite */Languageweeks/Details* besitzt eine korrekte Dependency Injection der Datenbank.		
Die Seite */Languageweeks/Details* besitzt einen Routingparameter für die Sprachwochen ID.		
Die Seite */Languageweeks/Details* fragt die benötigten Informationen aus der Datenbank korrekt ab.		
Die Seite */Languageweeks/Details* zeigt die geforderten Daten der Sprachwoche an.		
Die Seite */Languageweeks/Details* zeigt die geforderten Daten der Teilnehmer/innen an.		
Die Seite */Languageweeks/Details* sortiert die Teilnehmer/innen korrekt.		
Die Seite */Languageweeks/Details* hebt die weiblichen Teilnehmerinnen hervor.		
Die Seite */Languageweeks/Details* formatiert die Datumswerte korrekt.		
Die Seite */Languageweeks/Add* besitzt eine korrekte Dependency Injection der Datenbank.		
Die Seite */Languageweeks/Add* fragt die benötigten Informationen aus der Datenbank korrekt ab.		
Die Seite */Languageweeks/Add* zeigt ein Dropdownfeld mit allen Klassen korrekt an.		
Die Seite */Languageweeks/Add* zeigt ein Dropdownfeld mit allen Zielen korrekt an.		
Die Seite */Languageweeks/Add* zeigt ein Dropdownfeld mit allen Lehrer/innen korrekt an.		
Die Seite */Languageweeks/Add* zeigt die Eingabefelder für von, bis und Preis korrekt an.		
Die Seite */Languageweeks/Add* validiert die Eingaben.		
Die Seite */Languageweeks/Add* zeigt bei einer ungültigen Eingabe die Seite wieder an.		
Die Seite */Languageweeks/Add* speichert die Sprachwoche korrekt in der Datenbank.		
Die Seite */Languageweeks/Add* verweist nach dem Speichern auf die Indexseite.		