Probeklausur

Allgemeines

Die Arbeitszeit für die Bearbeitung der gestellten Aufgaben beträgt 5 Unterrichtseinheiten (250Minuten). Die 3 Teilaufgaben sind unabhängig voneinander zu bearbeiten, Sie können sich die Zeit frei einteilen. Wir empfehlen jedoch eine maximale Bearbeitungszeit von 60 Minuten für Aufgabe 1; 70 Minuten für Aufgabe 2 und 120 Minuten für Aufgabe 3.

Hilfsmittel

In der Datei *SPG_Probefachtheorie.sln* befindet sich das Musterprojekt, in dem Sie Ihren Programmcode hineinschreiben. Arbeiten sie bitte auf eigener Hardware.

Mit der Software DBeaver oder SQLite-Studio können Sie sich zur generierten Datenbank verbinden und Werte für Ihre Unit Tests ablesen.

Thema

Es ist ein Webbasierendes Reservierungs- und Abholsystem für eine Bücherei zu entwickeln. Leider haben die Büchereien die das System verwenden, keine Möglichkeit die Bücher zu versenden, also müssen Abhol- und Rückgabezeiten vereinbart werden. Auf Grund der aktuellen Corona-Situation darf sich immer nur ein Kunde in einem definierten Zeitfenster in der Bibliothek aufhalten. Ein solches Zeitfenster dauert genau 10 Min., danach darf der nächste Kunde in die Bibliothek.

Aufgabe 1: Erstellen einer Datenbank

Arbeitsauftrag

Es ist eine Datenbank nach dem Code First Szenario zu erstellen. Es sind dabei alle geforderten Datenbanktabellen als Entitäten abzubilden, der zugehörige *DBContext* zu erstellen und daraus eine Datenbank zu generieren.

Library

Da das System Mandantenfähig sein soll, müssen die einzelnen Libraries abgelegt werden könne, welche durch das System verwaltet werden sollen. Notwendige Felder sind:

- string Name
- string Suffix
- string Phrase
- string Bs
- int Rating

Category

Jede *Library* verfügt über mehre *Categories*, in welche die Bücher unterteilt werden können. Notwendige Felder sind:

- string Name
- Guid

Book

Das eigentliche Buch, das entliehen werden kann.

Notwendige Felder sind:

- string Name
- string Abstract
- string Ean13
- Guid

BorrowCharge und ChargeType

Jedes *Book* verfügt über mehrere *BorrowCharges* (Entlehnpreise, z.B. Normalpreis, Rabattpreis, Studentenpreis, …) Diese Gebühren werden über den *ChargeType* geregelt. D.h. jeder *BorrowCharge* hat einen konkreten *ChargeType*.

Notwendige Felder sind:

- [Datentyp selbst wählen] Amount
- string Currency
- Guid

Jeder *BorrowCharges* hat, wie gesagt, genau einen *ChargeType*. Der *ChargeType* verfügt neben der *Id* noch über einen *Name*.

Um die Abrechnung (*BorrowCharge * Angefangene Entlehnmonate*) müssen sie sich aber nicht kümmern. Es geht nur um die Terminvergabe!

Selbstverständlich muss jede Entität über eine ID verfügen!

Book, Category und *BorrowCharge* müssen darüber hinaus über eine *GUID* verfügen!

Generieren der Datenbank

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

Starten Sie den Unittest GenerateDbFromContextTest im Projekt *SPG_ProbeFachtheorie.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 SQLite-Studio oder DBeaver ob die erzeugte Datenbank den definierten Kriterien entspricht.

Bewertungskriterien

1. Es existiert eine Tabelle *Libraries* mit den Spalten, die die definierten Daten aufnehmen können.
2. Es existiert eine Tabelle *Categories* mit den Spalten, die die definierten Daten aufnehmen können.
3. Es existiert eine Tabelle *Books* mit den Spalten, die die definierten Daten aufnehmen können.
4. Es existiert eine Tabelle *BorrowCharges* mit den Spalten, die die definierten Daten aufnehmen können.
5. Es existiert eine Tabelle *ChargeTypes* mit den Spalten, die die definierten Daten aufnehmen können.
6. Die Tabelle *Categories* verweist korrekt auf die Tabelle *Libraries* über einen Fremdschlüssel.
7. Die Tabelle *Books* verweist korrekt auf die Tabelle *Categories* über einen Fremdschlüssel.
8. Die Tabelle *BorrowCharges* verweist korrekt auf die Tabelle *Books* über einen Fremdschlüssel.
9. Die Tabelle *ChargeTypes* verweist korrekt auf die Tabelle *BorrowCharges* über einen Fremdschlüssel.

Aufgabe 2: Services und Unit Tests

Arbeitsauftrag

In der vorgefertigten Solution sind bereits Modelklassen und ein *DBContext* vorgegeben. Sie können das verwenden um ihre Services zu implementieren.

Im Namespace Services ist eine Klasse mit dem Namen *PickupTimeService* vorgegeben. Der Service soll die Möglichkeit zur Verfügung stellen, Abholzeiten für einen bestehenden Sammelkorb zu verwalten. Die Abholzeiten werden dabei so organisiert, dass ein Kunde ein Zeitfenster von 10 Minuten zur Verfügung hat um den Sammelkorb abzuholen.

(z.B.: Kunde-1: 01.01.2022 08:10, Kunde-7: 01.01.2022 08:30, Kunde-3: 01.01.2022 09:50, …) Jeder Abholzeitpunkt darf also nur einmal existieren.

Implementieren Sie dazu folgende Methoden:

AddAppiontment

Die Methode erstellt einen neuen Abholdatensatz in der Datenbank.

Dafür gelten folgende Regeln:

- Es darf nur ein Abholtermin pro 10 Minuten Zeitfenster existieren (08:00, 08:10, 08:20, …)
- Der Abholtermin muss in der Zukunft liegen (+ 1 Tag)

Weiters:

- Die Methode erhält als Parameter ein *Data Transfer Object (DTO)*.

DeleteAppiontment

Löscht einen existierenden Abholdatensatz aus der Datenbank. Die Methode liefert *true*, bei Erfolg, *false* bei Misserfolg. Es darf keine Exception geworfen werden.

Dafür gelten folgende Regeln:

- Der Termin darf nur gelöscht werden, wenn er in der Zukunft liegt.

Bewertungskriterien

1. *AddAppointmentSuccessTest* beweist, dass die Methode *AddAppiontment* erfolgreich einen Abholtermin in der Datenbank anlegt.
2. *AddAppointmentNotUniqueValidationErrorTest* beweist, dass eine ValidationException geworfen wird, wenn ein Abholtermin bereits existiert.
3. *AddAppointmentNotInFutureValidationErrorTest* beweist, dass seine ValidationException geworfen wird, wenn ein Abholtermin nicht mind. einen Tag in der Zukunft liegt.
4. *DeleteAppointmentSuccessTest* beweist, dass die Methode *DeleteAppointment* erfolgreich einen Datensatz in der Datenbank löscht
5. *DeleteAppointmentNotInFutureValidationErrorTest*, beweist, dass die Methode *false* liefert, wenn ein Termin in der Vergangenheit gelöscht wird.

Aufgabe 3: Webapplikation

Arbeitsauftrag

Die Applikation soll über ein Web Frontend verfügen. Die Logik dafür kann, aus Gründen der Zeitersparnis, direkt in den Controllern platziert werden. Es müssen keine weiteren Services angelegt werden. (Selbstverständlich dürfen sie das aber tun)

Es ist dafür eine Razor Pages Applikation vorbereitet, die sie erweitern sollen. Tipp: Aktivieren Sie das gewünschte Projekt als Startprojekt in Visual Studio.

Das Datenmodell sowie die Datenbank werden von Aufgabe 2 verwendet. Der Datenbankkontext ist bereits registriert und kann über die LibraryContext Klasse mittels Dependency Injection genutzt werden. Des Weiteren sind eine vollständige Authentifizierung und Autorisierung vorhanden. Das Service *HttpCookieAuthService.cs* stellt die Methode *UserId()* zur Verfügung, welche den angemeldeten User zur Verfügung stellt.

Folgende Pages sollen zur Verfügung gestellt werden

User

Folgende Route */User/Index* zeigt eine Auflistung aller User in der Datenbank an. Es sollen *Username, Firstname, Lastname, EMail* und die *Role* angezeigt werden.

Authorisierung:

Ist der User mit der Rolle *Guest* angemeldet, soll nur er selbst in der Liste angezeigt werden (Also nur ein Datensatz). Ist der User mit der Rolle *BackOfficeEmployee* angemeldet, sollen alle User aus der DB angezeigt werden.

Generell muss man angemeldet sein um die Liste überhaupt sehen zu können. *[Authorize()]*

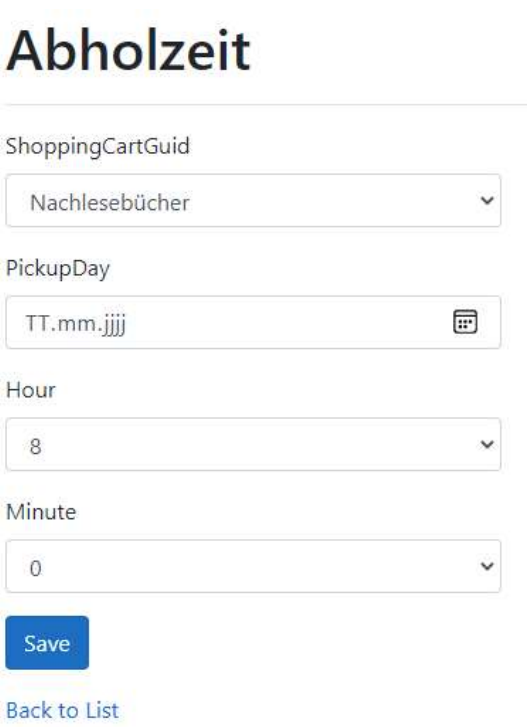
Die Route */User/Index/Details/{GUID}* zeigt eine Liste der Abholzeiten des ausgewählten Users an. Es soll das vollständige Datum und Zeit angezeigt werden, das Format ist frei wählbar. Zusätzlich soll auch der Name des zugehörigen Sammelkorbes angezeigt werden. Dieser enthält einen Link zur Route */ShoppingCarts/Details/{GUID}*. (Details siehe unten)

Hier sollen die Zeiten administriert werden können (Hinzufügen, Löschen) Verwenden sie dafür den zuvor erstellten Service. (Sollte ihnen die Implementierung des Services nicht gelungen sein, können sie den notwendigen Code auch direkt im Controller platzieren).

Alle Routen hier sollen nur erreichbar sein, wenn man angemeldet ist!

AddPickupTime

Die Route */PickupTimes/Create* zeigt ein Formular an, in dem eine neue Abholzeit eingetragen werden kann. Die Darstellung ist frei wählbar. Folgender Vorschlag kann umgesetzt werden:



Eingegeben werden muss: Um welchen Sammelkorb es sich handelt und den genauen Terminslot. Dieses Formular besteht aus Dropdown-Feldern. Zumindest die Uhrzeit und den Sammelkorb darf man nicht frei eingeben, da die Terminslots ja definiert sind. (Es gelten dieselben Regeln wie in Kapitel 2 beschrieben)

RemovePickupTime

Die GET-Route */PickupTimes/Delete/*{GUID} zeigt die Details des zu löschenden Datensatzes in einem neuen Fenster an. Dieses verfügt auch über einen Button „wirklich löschen?“. Dieser löst die POST-Route aus und löscht den Termin endgültig aus der Datenbank. (Es gelten dieselben Regeln wie in Kapitel 2 beschrieben)

Alle Routen hier sollen nur dann erreichbar sein, wenn man angemeldet ist!

ShoppingCart

Die Route */ShoppingCarts/Details/{GUID}* zeigt den Inhalt des gewählten Sammelkorbes in Form einer Liste an. Anzuzeigen ist dabei, der Name des Buches, die EAN-Nummer und der Preis

Alle Routen hier sollen nur erreichbar sein, wenn man angemeldet ist!

Bewertungskriterien

1. Die Route */Users/Index* zeigt eine Übersicht aller in der Datenbank eingetragenen User an, wenn man als *BackOfficeEmploee* angemeldet ist. (Anzuzeigende Daten siehe oben)
2. Die Route */Users/Index* zeigt nur den angemeldeten User an, wenn man als *Guest* angemeldet ist. (Anzuzeigende Daten, wie 1.)
3. Die Route */Users/Index/Details/{GUID}* zeigt die Abholzeiten des gewählten Users an.
4. Alle Routen in *Users* sind nur erreichbar, wenn man angemeldet ist.
5. Die Route */ShoppingCarts/Details/{GUID}* zeigt den Sammelkorb der gewählten Abholzeit an. (Anzuzeigende Daten siehe oben)
6. Alle Routen in *ShoppingCarts* sind nur erreichbar, wenn man angemeldet ist.
7. Die Route */PickupTimes/Create* zeigt ein Formular an zur Eingabe einer neuen Abholzeit.
8. Die neue Abholzeit wird in der Datenbank gespeichert.
9. Die GET-Route */PickupTimes/Delete*/{GUID} zeigt die Details des zu löschenden Datensatz nocheinmal an
10. Die GET-Route */PickupTimes/Delete*/{GUID} verfügt über einen Button „wirklich löschen?“
11. Die POST-Route */PickupTimes/Delete*/{GUID} löscht den Datensatz endgültig aus der Datenbank.
12. Alle Routen in *PickupTimes* sind nur erreichbar, wenn man angemeldet ist.

Beurteilungsblatt:

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 28 Punkte zu erreichen. Für eine positive Beurteilung der Klausur müssen mindestens 50 % der Gesamtpunkte erreicht werden.

Beurteilungsstufen

28 – 24.5 Punkte: Sehr gut
24 - 21 Punkte: Gut
20.5 – 17.5 Punkte: Befriedigend
17 – 14 Punkte: Genügend

Abgabe:

Bitte erstellen sie zur Abgabe ein ZIP-File und ergänzen sie den Dateinamen um *_[Nachname]_[Vorname]* und geben sie das Ergebnis auf MS Teams ab. Eine Abgabe (unter Aufgaben) ist eingerichtet.

SPG_Probefachtheorie_[Nachname]_[Vorname].zip

Viel Glück!