Programmier-Projekt „Thesis Datenbank“

Das folgende Programmierprojekt soll bis zum Ende der Vorlesungszeit in Kleingruppen bearbeitet werden. Die dabei zu erstellende Webanwendung ist Grundlage für die abschließende mündliche Prüfung.

Hintergrund & Zielsetzung

Die Lehrstühle der „Data Driven Decisions (D^3) Group“ betreuen Abschlussarbeiten (d.h., Bachelor- und Master- Thesen) in verschiedenen Studiengängen. Dabei durchläuft jedes Thema einer Abschlussarbeit über Monate hinweg einen mehrstufigen Lebenszyklus, angefangen von der anfänglichen Themendefinition bis zur Erstellung des Gutachtens durch den Betreuer. Zur Unterstützung dieses Prozesses soll eine Webanwendung eingesetzt werden, welche wiederum auf einer relationalen Datenbank aufbaut.

Das zentrale Element des Datenbankmodells ist die Klasse/Tabelle „Thesis“, in der alle wesentlichen Informationen zur anfänglichen Themenstellung und dem später zu erstellenden Gutachten enthalten sind. Jede Thesis ist mit einem Betreuer verknüpft, der ein User der Webanwendung ist. Zusätzlich soll die Anwendung zwischen Themen/Thesen bzw. Usern der einzelnen Lehrstühle unterscheiden können, d.h., in einer zusätzlichen Klasse/Tabelle sollen die Lehrstühle der D^3-Gruppe gespeichert werden. Jeder User ist einem Lehrstuhl zugeordnet und damit indirekt auch jedes Thema bzw. jede Thesis.

Die Nutzung der Anwendung umfasst einerseits das Anlegen neuer Thesisthemen, die (sofern noch frei) über eine öffentlich zugreifbare Webseite den Studierenden angezeigt werden. Jede Thesis durchläuft einen Lebenszyklus über mehrere Stati hinweg:

- Frei: Das Thesisthema ist noch nicht vergeben.
- Reserviert: Zu dem Thema fand eine Vorbesprechung statt, die Thesis wurde aber noch nicht offiziell beim Prüfungsamt angemeldet. Aus diesem Status kann das Thema ggfs. auch wieder auf „frei“ zurückwechseln.
- In Bearbeitung: Die Bearbeitungszeit der Thesis läuft.
- Abgegeben: Die Thesis wurde abgegeben, ist aber noch nicht begutachtet worden.
- Begutachtet: Die Thesis wurde abgegeben und das Gutachten inkl. Note wurde erstellt.

Zur Entwicklung eines Prototyps sollen .NET-Technologien eingesetzt werden. Verwenden Sie dazu die in der Lehrveranstaltung vorgestellten Konzepte von C#, ASP.NET MVC, HTML, CSS und JavaScript. Das Projekt ist in mehrere Teile unterteilt, die im Folgenden genauer beschrieben werden.

Teil 1: Models

Legen Sie ein neues ASP.NET-Projekt mit dem MVC-Template an und fügen Sie die Authentifizierungskomponente ASP Identity gleich zu Beginn hinzu. Stellen Sie sicher, dass die Identity-Seiten für Login/Logout per Scaffolding verfügbar gemacht werden sowie eine eigene DbSet- und User-Klasse erstellt werden.

Definieren Sie anschließend in dem Projekt das Datenmodell Ihrer Anwendung, welches folgende Model-Klassen (bzw. Tabellen in der relationalen DB) beinhaltet:

- **Thesis** beinhaltet alle Angaben zur einzelnen Abschlussarbeit, d.h. Titel, Aufgabenstellung, Anmeldedatum, Betreuer, Lehrstuhl, Abgabedatum usw. (Hinweis: Aus der Sicht eines Datenanalysten besteht „Thesis“ eigentlich aus mehreren separaten Entitäten zum Thema, der Thesis, dem Gutachten und dem Studierenden, die hier aber aus Effizienzgründen und zur Vereinfachung der Benutzeroberfläche in einer einzelnen Model-Klasse zusammengefasst werden).

- **Program** enthält Informationen (Name) zu den Studiengängen (u.a. „Bachelor Wirtschaftsinformatik“, „Bachelor Wirtschaftswissenschaften“, „Master Business Management“, „Master Information Systems“). Eine Thesis wird bei Anmeldung einem Studierenden zugeordnet, der wiederum in einem bestimmten Studiengang immatrikuliert ist.
- **Chair** entspricht einem Lehrstuhl, dem ein User zugeordnet ist.
- **AppUser** ist eine Erweiterung der Klasse „IdentityUser“, in der zusätzlich der Vor- und Nachname, die Zugehörigkeit zu einem Lehrstuhl sowie den aktuellen Status. Der Status kann die Werte „active“ und „inactive“ annehmen und dient dazu, zwischen derzeitigen und früheren MitarbeiterInnen eines Lehrstuhls zu unterscheiden.

Erstellen Sie vor diesem Hintergrund Model-Klassen die im Einzelnen mit den folgenden Datenfeldern ausgestattet sind. Fügen Sie darüber hinaus weitere Felder hinzu, um die oben erwähnten Verknüpfungen zwischen Klassen/Tabellen abzubilden.

Entität	Feld	Datentyp	Bemerkungen
Thesis	Id	Int	
	Title	String	
	Description	String	Mehrzeiliger Text
	Bachelor	Bool	Thema ist für Bearbeitung als Bachelor Thesis geeignet
	Master	Bool	Thema ist für Bearbeitung als Master Thesis geeignet
	Status	Enumeration	Siehe Erklärung im Text
	StudentName	String	
	StudentEMail	String	
	StudentId	String	Matrikelnummer
	Registration	DateTime	Anmeldedatum
	Filing	DateTime	Abgabedatum
	Summary	String	Mehrzeiliger Text
	Strengths	String	Mehrzeiliger Text
	Weaknesses	String	Mehrzeiliger Text
	Evaluation	String	Mehrzeiliger Text
	ContentVal	Int	Bewertung 1-5
	LayoutVal	Int	Bewertung 1-5
	StructureVal	Int	Bewertung 1-5
	StyleVal	Int	Bewertung 1-5
	LiteratureVal	Int	Bewertung 1-5
	DifficultyVal	Int	Bewertung 1-5
	NoveltyVal	Int	Bewertung 1-5
	RichnessVal	Int	Bewertung 1-5

	ContentWt	Int	Gewichtung 0-100, Defaultwert 30
	LayoutWt	Int	Gewichtung 0-100, Defaultwert 15
	StructureWt	Int	Gewichtung 0-100, Defaultwert 10
	StyleWt	Int	Gewichtung 0-100, Defaultwert 10
	LiteratureWt	Int	Gewichtung 0-100, Defaultwert 10
	DifficultyWt	Int	Gewichtung 0-100, Defaultwert 5
	NoveltyWt	Int	Gewichtung 0-100, Defaultwert 10
	RichnessWt	Int	Gewichtung 0-100, Defaultwert 10
	Grade	Int	Note 1,0 – 5,0
	LastModified	DateTime	Zeitstempel der letzten Änderung am Datensatz
Supervisor	FirstName	String	
	LastName	String	
	Active	Bool	
Program	Id	Int	
	Name	Enumeration	Siehe Beispiele im Text

Geben Sie für alle Felder auch über Annotationen eine deutsche Beschriftung an, so wie sie später auf dem Bildschirm erscheinen soll. Felder wie „Title“, „Description“, „LastModified“ etc. können als [Required] gekennzeichnet werden. Achten Sie auch darauf, der Enumeration für das Feld „Status“ für die einzelnen möglichen Werte jeweils eine deutsche Beschriftung mitzugeben, da die Werte in der Status-Enumeration in Ihrem Code wie folgt Englisch benannt werden sollen:

- Available
- Allocated
- Filed
- Submitted
- Graded

Die Felder „Bachelor“ und „Master“ dienen zur Kennzeichnung, ob ein bestimmtes Thema zur Vergabe als Bachelor- und/oder Master-Thesis geeignet ist. Eines der beiden Felder muss mindestens „true“ sein, d.h., erstellen Sie eine Validierungsmethode für die Model-Klasse, die dies prüft und ggfs. dem ModelState einen Fehler hinzufügt, der dann in der Webseite angezeigt wird. Bei dieser Gelegenheit sollte die Validierung auch prüfen, ob die Summe aller Gewichtungsfaktoren („ContentWt“ usw.) genau 100% ergibt. Außerdem sollte die Validierung prüfen, ob eine Thesis mit dem Status „graded“ eine Note enthält.

Passen Sie die Startdatei „program.cs“ und die Appsettings so an, dass eine relationale DB verfügbar ist sowie ASP Identity mit Usern und Rollen. Identity und Ihre eigenen Models sollen in derselben DB enthalten sein, d.h. Sie nutzen stets den für Identity erstellten Datenbankkontext.

Fügen Sie der DbContext-Klasse in OnModelCreating einige Zeilen hinzu, um mindestens einen Admin-User zu erstellen, der auch die Rolle „Administrator“ erhält. Erstellen Sie eine Datenbank und legen darin mit den Befehlen "add- migration" und "update-database" das Datenmodell an. Prüfen Sie, ob Ihre Testdaten auch tatsächlich in den Tabellen gespeichert wurden.

Teil 2: Controller & Views

Nachdem die Modelklassen und die Datenbank stehen, müssen Sie in einem zweiten Schritt eine Benutzeroberfläche erstellen, die den Lebenszyklus einer Thesis von der anfänglichen Themendefinition bis zur Begutachtung und abschließenden Archivierung abbildet. Zu diesem Zweck müssen diverse Controller bzw. Views erzeugt werden. Gehen Sie dabei wie folgt vor:

- Beginnen Sie zunächst mit einer per Scaffolding erzeugten Oberfläche zur Pflege der Lehrstuhl-Tabelle, die nur für Administratoren zugreifbar ist. Die Detail-Seite können Sie entfernen und den entsprechenden Link aus der Index-Seite auch.
- Anschließend braucht es eine Oberfläche, über die neue Nutzer angelegt werden können. Hierbei soll nicht auf die DB-Tabelle mit den Nutzerdaten direkt zugegriffen werden, sondern indirekt über den UserManager, der von Identity bereitgestellt wird. Gehen Sie dazu so vor, dass Sie einen Controller erstellen, der in seinem Konstruktor den UserManager entgegennimmt und intern speichert. Erstellen Sie Views für die Liste aller User, das Anlegen eines neuen Users und das Bearbeiten eines vorhandenen Users. Controller und Views kommunizieren über ein ViewModel, in dem die ID eines Users und die zu bearbeitenden Felder enthalten sind. Das ViewModel wird mit Hilfe des UserManagers mit Daten befüllt bzw. umgekehrt werden Daten aus dem ViewModel über den UserManager in die DB übertragen.
- Erstellen Sie anschließend auch für das Model „Thesis“ mit Hilfe von Scaffolding einen Controller mit Views, der nur für eingeloggte User zugreifbar ist. Die Index-Seite soll nur die wichtigsten Spalten und auch nicht alle Thesen auf einmal anzeigen, sondern nur die ersten 10 (nach Datum sortiert, so dass die neuesten zuerst angezeigt werden). Weitere 10 Thesen können per Button nachgeladen werden usw. Sorgen Sie außerdem dafür, dass die Liste der Themen/Thesen nach Lehrstuhl, dem Namen des Bearbeiters bzw. einem Wort im Titel gefiltert werden kann. In der Index-View braucht es also eine Select-Liste sowie zwei Eingabefelder, in denen Bestandteile des Namens bzw. des Titels eingegeben werden können und einen Filter-Button, der die Liste der Thesen erneut vom Server lädt. Wichtig: Diese Filterwerte müssen auch für das Nachladen von Listeneinträgen gelten.
- Gestalten Sie die Detail-View einer Thesis als ausdruckbare Ansicht des Gutachtens. Darin ist keine Themenbeschreibung enthalten und auch keine Eignung für Bachelor/Master, sondern nur die für das Gutachten relevanten Informationen. Wichtig: Die View soll nicht auf die Standard-Layout-Seite zurückgreifen, sondern mit purem HTML so formatiert sein, dass man die Seite gut ausdrucken kann.
- Erstellen Sie darüber hinaus eine einzelne View, in der alle derzeit freien oder reservierten Thesisthemen nach Lehrstühlen sortiert angezeigt werden. Diese View soll auch für nicht eingeloggte BenutzerInnen zugreifbar sein. Die View kann entweder zusätzlich über den zuvor beschriebenen Thesis-Controller bereitgestellt werden oder Sie erstellen für diese einzelne View einen eigenen Controller, der nur eine einzelne Action-Methode für diese View umfasst. Gestalten Sie die Oberfläche so, dass zunächst nur nach Lehrstuhl gruppiert eine Liste aller Titel, dem jeweiligen Betreuer (mit Mailadresse als Link dahinter) und dem aktuellen Status angezeigt wird. Jedes Thesisthema soll aufklappbar sein, so dass die Themenbeschreibung angezeigt wird.

Gestalten Sie bis auf die Gutachten-View alle Views der Anwendung in einigermaßen ansprechender Form mit Hilfe von Bootstrap 5. Gestalten Sie entsprechend auch die Login/Logout-Seiten von Identity ein wenig um.

Teil 3: Funktionalität zur Thesisverwaltung

Um die Thesisdaten in den verschiedenen Lebenszyklen zu verwalten, ist eine Modifikation der per Scaffolding erzeugten Thesis-View Edit notwendig. Gruppieren Sie die Inhalte der Thesis-Klasse nach Bereichen:

- Thema: Titel, Beschreibung (mehrzeilig!), Betreuer (angezeigt als Vor-/Nachname + Lehrstuhl), Bachelor, Master
- Bearbeitung: Status, Student-Name, Matrikelnummer, Student-Email, Studiengang, Anmeldedatum, Abgabedatum.
- Gutachten: Zusammenfassung (mehrzeilig!), Stärken (mehrzeilig!), Schwächen (mehrzeilig!), Evaluation (mehrzeilig!), einzelne Bewertungskriterien mit Bepunktung von 1 bis 5 und prozentualer Gewichtung, Summe aller Gewichte.
- Note: Berechnetes Feld mit Notenvorschlag, Note als Auswahlfeld (1.0, 1.3, 1.7, …, 4.0, 5.0).

Die einzelnen quantitativen Bewertungen sollen auf dem Bildschirm strukturiert angezeigt werden, d.h., die Bewertungsmöglichkeiten von 1 bis 5 als Radio Buttons nebeneinander und daneben wiederum der Gewichtungsfaktor. Unter der Spalte mit den Gewichtungsfaktoren soll das (nur lesbare) Feld mit der Summe aller Gewichtungen stehen. Das Layout soll also etwa so aussehen (bessere Vorschläge sind auch willkommen):

Kriterium	**Bewertung**	**Gewichtung**
Inhalt	1 o o o o o 5	30 %
Gestaltung	1 o o o o o 5	15 %
Aufbau	1 o o o o o 5	10 %
Sprache	1 o o o o o 5	10 %
Literatur	1 o o o o o 5	10 %
Schwierigkeitsgrad	1 o o o o o 5	5 %
Neuigkeitsgrad	1 o o o o o 5	10 %
Themenerfassung	1 o o o o o 5	10 %
		100 %

Für die Ermittlung eines Notenvorschlags soll eine Funktion eingebaut werden, die auf Basis der Einzelbewertungen einen Notenvorschlag berechnet. Die Einzelbewertungen bewegen sich wie gesagt jeweils auf einer Skala von 1 bis 5. Zusätzlich ist jede Teilbewertung mit einer Gewichtung zwischen 0 und 100 versehen. Bauen Sie in die View einen zusätzlichen Button und ein Readonly-Textfeld für das Ergebnis ein. Wird der Button geklickt, berechnet eine Javascript-Funktion auf dem Client einen Notenvorschlag als gewichteten Durchschnitt der Einzelbewertungen. Falls keine Berechnung möglich ist, wird eine Fehlermeldung angezeigt (über die Javascript-Anweisung „alert“). Der Notenvorschlag kann auch krumme Werte ergeben (z.B. „1,87“), soll aber auf zwei Nachkommastellen gerundet werden.

Wichtig: Das Feld mit der tatsächlichen Note ist unabhängig von dem Notenvorschlag, d.h. es soll nicht der Vorschlag auf die möglichen Notenwerte gemappt und dort eingetragen werden.

Teil 4: Auswertung

Für den Admin sollten zwei Auswertungsseiten angelegt werden, die direkt von der Startseite der Anwendung aus über die Menüleiste erreichbar sind:

- Anzahl abgegebener Thesen pro Kalenderjahr und Lehrstuhl über alle Jahre als Liniendiagramm.
- Anzahl abgegebener Thesen nach Studiengang über alle Jahre als Tortendiagramm.
- Anzahl in der DB eingetragener Thesisthemen nach Betreuer als horizontales Balkendiagramm (nach Anzahl absteigend sortiert).

Erstellen Sie zu diesem Zweck einen Controller mit zwei Action-Methoden und zwei Views (je eine Methode und eine View pro Auswertung). Die Action-Methoden fragen die jeweils benötigten Daten aus der DB ab und übergeben diese an die View. In der View werden die Daten direkt in den HTML-Code integriert, d.h., es müssen nicht dynamisch Daten vom Server nachgeladen werden oder Ähnliches. Es müssen auch keine Datenverarbeitungen selbst programmiert werden, da die Daten in der benötigten Form direkt mit Hilfe der Funktionalität des Entity Framework ermittelt werden können. In beiden Auswertungen müssen die Rohdaten aus der DB nach ein bzw. zwei Kriterien gruppiert sowie aufsummiert werden und können dann direkt in die View „abgekippt“ werden.

Verwenden Sie für die grafische Darstellung der Daten in der Webseite ein Framework wie Google Charts (https://developers.google.com/chart/interactive/docs/quick_start).

Prof. Thiesse und Marco Röder wünschen Ihnen viel Erfolg mit der Programmieraufgabe!