1 Beteiligte Personen

Namen	Funktion
Maurin, Jenni	Entwickler
Hodel, Jonas	Entwickler

2 Inhalt

1 Beteiligte Personen 1
2 Inhalt 1
3 Projektbeschreibung 1
3.1 Übersicht 1
3.2 Umgebung 1
4 Entity-Relationship Diagramm 2
5 Relationales Modell 3
6 Physisches Modell 3
7 Reflexion 4

3 Projektbeschreibung

3.1 Übersicht

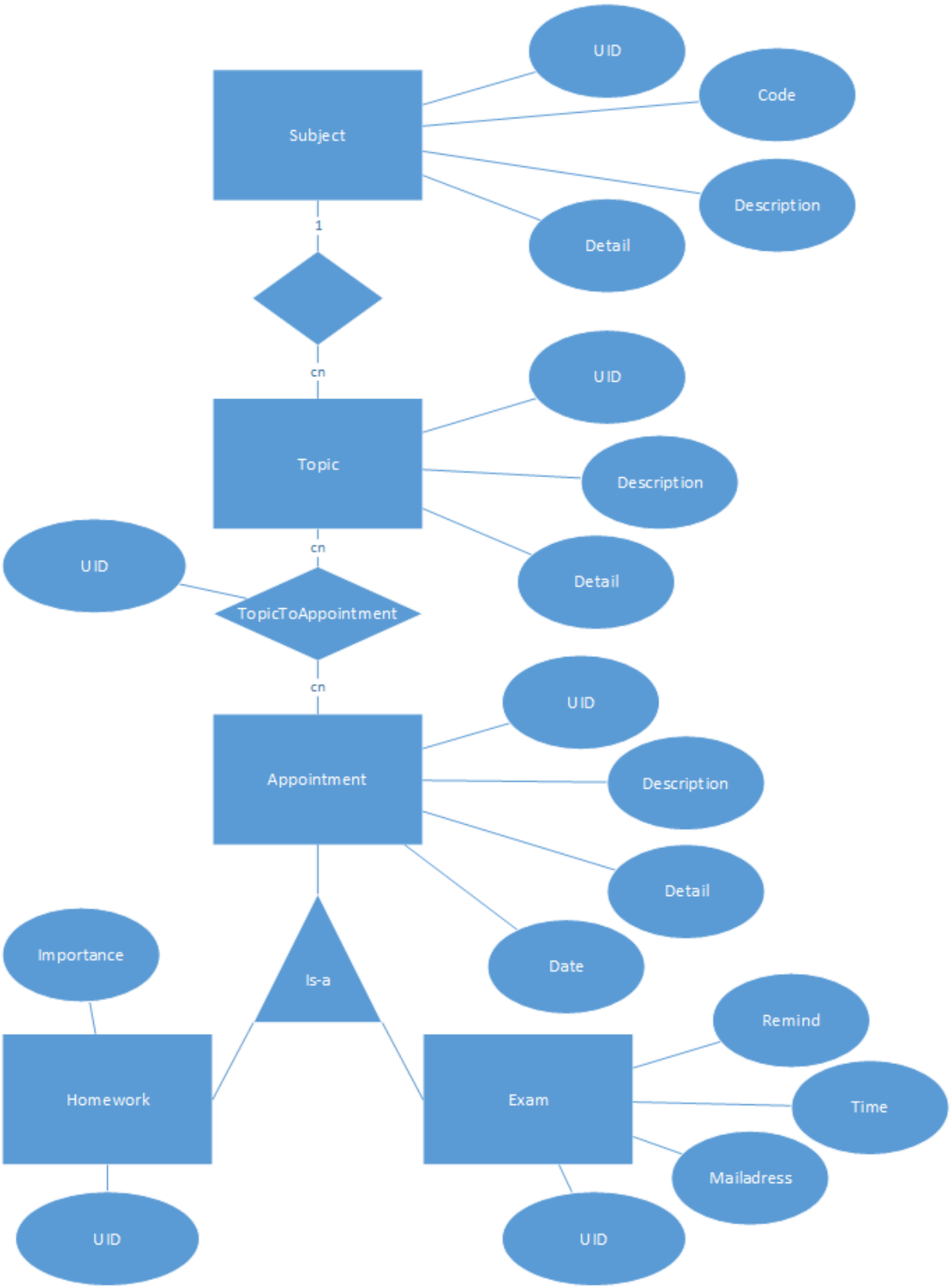
Mit der Hausaufgabenverwaltung „HomeworX“ kann ein Benutzer über einen Webbrowser in erster Linie Termine, wie Hausaufgaben und Prüfungen erfassen. Diese werden als Liste dargestellt und können

Es ist möglich den Terminen Fächer und deren Themen zuzuweisen. Einem Termin können mehrere Themen (z.B. „Kreise“ und „Vierecke“), aber nur ein Fach (z.B. „Mathematik“) zugeordnet werden. Fächer und deren Themen können frei konfiguriert werden.

3.2 Umgebung

Das Webprojekt wurde mit ASP.NET MVC umgesetzt. Das Persistieren der Daten wurde mit der Datenbank MSSQL und dem Entity Framework mit Repository Pattern realisiert. Als Versionskontrolle und Ablage des Quellcodes wurde GitHub verwendet.

4 Entity-Relationship Diagramm



5 Relationales Modell

Tabelle	Attribute
Subject	UID, Code, Description, Detail
Appointment	UID, Description, Detail, Date, SubjectUID
Homework	UID, Importance
Exam	UID, Remind, Time, Mailadress
Topic	UID, Description, Detail, SubjectUID
TopicToAppointment	UID, TopicUID, AppointmentUID

6 Physisches Modell

Relation	Attributname	Datentyp	Eingabe erford.	Verweis auf Tabelle	Bemerkungen
Subject	UID	uniqueidentifier	Ja		Autowert
	Code	String(10)	Ja		
	Description	String(100)	Ja		
	Detail	Text	Nein		
	Kandidatenschlüssel: Die Kombination von «Code» und «Description» ist eindeutig				
Appointment	UID	uniqueidentifier	Ja		Autowert
	Description	String(100)	Ja		
	Detail	Text	Nein		
	Date	Date	Ja		
	SubjectUID	Integer	Ja	Subject	
	Kandidatenschlüssel: Die Kombination von «Description» und «Date» ist eindeutig				
Homework	UID	uniqueidentifier	Ja	Appointment	
	Importance	Tinyint	Nein		Zwischen 1 und 5

Relation	Attributname	Datentyp	Eingabe erford.	Verweis auf Tabelle	Bemerkung
Exam	UID	uniqueidentifier	Ja	Appointment	
	Remind	Bit	Ja		Boolean
	Time	Datetime	Nein		Auf eine Minute genau.
	Mailadress	String(50)	Nein		
Topic	UID	uniqueidentifier	Ja		Autowert
	Description	String(100)	Ja		
	Detail	Text	Nein		
	SubjectUID	uniqueidentifier	Ja	Subject	
	Kandidatenschlüssel: «Description» ist eindeutig				
TopicTo Appointment	UID	uniqueidentifier	Ja		Autowert
	TopicUID	uniqueidentifier	Ja	Topic	
	AppointmentUID	uniqueidentifier	Ja	Appointment	

7 Reflexion

Mit dem erarbeiten Projekt sind wir nicht vollständig zufrieden. Der Clientseitige teil, sprich HTML(cshtml), JS und CSS(Bootstrap) ist uns gut gelungen. Beim JS Code hatten wir Probleme, dass wir diese nicht in ein externes File auslagern konnten. Dies hat mit dem CSHTML Code, welcher auch im Script vorhanden ist zu tun. Zu Beginn wollten wir ES6 benutzen, doch dies konnten wir nicht in CSHTML einbauen. Somit mussten wir auf JS mit JQuery ausweichen. Der Bootstrap Teil ist uns gut gelungen, auch wenn nicht alle Controls responsive sind. Dies war aber auch nicht das Ziel des Projekts. Serverseitig sind wir mit der Validierung nicht zufrieden. Die Validierung wurde im Metadaten File implementiert, doch wird dem User keine Meldung angezeigt, wenn bei der Validierung ein Problem aufgetreten ist. Dies ist sehr unschön. Die Werte werden mit der ModelState.AddModelError Funktion hinzugefügt. Jedoch ist kein ValidationMessageFor für die Keys vorhanden. Somit werden diese auch nicht angezeigt. Dies müsste noch implementiert werden. Was aber serverseitig gut gelungen ist, ist das Umsetzten der UnitOfWork. Wir haben ein gutes GenericRepository entwickelt. Somit mussten wir nicht mehr viele individuelle Anpassungen an den einzelnen Repositories vornehmen.