Fernuniversität Hagen

Bachelor-Thesis

Entwicklung einer iOS-App für Multiple-Choice-Fragen

Studiengang: Wirtschaftsinformatik

Allan Kaufmann
9502998

Lehrgebiet: Parallelität und VLSI
Prof. Dr. Jörg Keller

13. Dezember 2018

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Kapitel 1

Einleitung

Im Folgenden wird aus Gründen der besseren Lesbarkeit ausschließlich die weibliche Form benutzt. Es können dabei aber sowohl männliche als auch weibliche Personen gemeint sein.

1.1 Problemstellung

Mit Multiple-Choice-Aufgaben (MC) lassen sich fast alle Lehrziele, auch solche auf einem hohen Lehrzielniveau (Verständnis, Anwendung) erfassen[7]. Multiple-Choice-Aufgaben werden häufig an Universitäten für Prüfungen verwendet, da die Auswertung dieser Aufgaben effizient möglich ist, die Aufgaben eindeutig zu interpretieren sind und der Aufgabentyp die Rechtssicherheit der Prüfungen erhöht[12].

Damit sich Studierende auf diese Prüfungsform besser vorbereiten können, stellen die Lehrenden häufig Übungsfragen in Multiple-Choice-Form bereit. Diese werden überlicherweise in Papierform ausgegeben oder auf elektronischem Weg per eMail oder in Übungssysteme online bereitgestellt. Weniger verbreitet ist hingegen die Möglichkeit, Übungsfragen direkt auf mobilen Endgeräten zu bearbeiten, da bisher keine App vorhanden ist, welche einerseits den Lehrenden die Erfassung und Auswertung von Übungsfragen erleichtert, andererseits den Kursteilnehmerinnen die Möglichkeitet bietet, diese Fragen direkt zu bearbeiten.

Diese Möglichkeit wurde im Rahmen dieser Bachelor-Thesis näher untersucht und entsprechende Lösungsvorschläge für den praktischen Einsatz erarbeitet.

1.2 Zielsetzung und Erkenntnisinteresse

Ziel dieser Bachelor-Thesis ist die Konzeption und Umsetzung einer Multiple-Choice-App für iOS-basierte Geräte. Diese Multiple-Choice-App soll von Studierenden zur Lernzielkontrolle sowie für Prüfungsvorbereitungen verwendet werden. Desweiteren umfasst diese Arbeit auch die Konzeption und Umsetzung einer Webanwendung, welche es Kursbetreuern ermöglicht, für die Multiple-Choice-App entsprechende Fragen einzugeben und auszuwerten.

Um dieses Ziel zu erreichen, wurden konstruktionsorientierte Methoden zur Erforschung des Themas, sowie zur Umsetzung eines Softwareprototypen, verwendet[8, s.S. 34]. Hierzu gehören Modellierungsmethoden, etwa zur Konzeption eines Datenmodells oder für den Entwurf der Be-

nutzeroberflächen.

Das gesamte Vorgehen der Konzeption und Umsetzung, die gewonnenen Erkenntnisse und die Ergebnisse wurden in dieser Bachelor-Thesis schriftlich fixiert und erläutert. Diese Arbeit enthält die notwendige Dokumentation, sowie in der Anlage den kompletten Quellcode, damit die Ergebnisse in einem Entwicklungssystem nachvollzogen, ggfls. erweitert und verwendet werden könnte.

1.3 Forschungsstand und Theoretische Grundlage

Zum fachlichen Thema Multiple-Choice-Tests existiert eine Vielzahl von Büchern, Artikeln und Dokumenten, welche sich u. A. damit befassen, wie Multiple-Choice-Aufgaben erstellt werden sollten. Nachfolgend drei Beispiele:

- 'Developing and Validating Multiple-choice Test Items' von Thomas M. Haladyna[9]
- 'Erstellen und Bewerten von Multiple-Choice-Aufgaben' von der Universität Hannover[6]
- 'Richtlinien zur Erstellung von einfachen Multiple-Choice-Aufgaben nach Gronlund'[7]

Für die Konzeption und Umsetzung von iOS-basierten Anwendungen kann ebenfalls auf eine umfangreiche Literatur zurückgegriffen werden. Dazu gehören u.A.:

- Xamarin in Action von Jim Bennett[1],
- Swift 2: das umfassende Handbuch von Michael Kofler[2],
- die Dokumentation von Xamarin.iOS[61]

Forschungsschriften, welche das Thema 'mobile Multiple-Choice-Anwendungen' behandeln, sind keine bekannt. Erwähnenswert sind zwei Arbeiten, die thematisch ähnlich sind:

- in der Studienarbeit 'Erstellung einer Webapplikation zur Online-Beantwortung von Multiple-Choice-Klausuren' setzte Höschele eine Webapplikation für die Bearbeitung von Multiple-Choice-Klausuren um. [11]
- in einem wissenschaftlichen Artikel von Chiou und Chen wird die Möglichkeit besprochen, die Wahlen über eine mobile Anwendung durchzuführen. Dies ist nicht direkt mit dem Thema dieser Arbeit vergleichbar, enthält aber ähnliche Anforderungen. [10]

Desweiteren sind im Android Play-Store zwei Apps verfügbar, welche die Erfassung von Multiple-Choice Fragen ermöglichen:

- Multiple Choice Question
- Brainyouu

Bei Prüfung dieser Apps lässt sich jedoch feststellen, dass die meisten Anforderungen aus dieser Problemstellung nicht erfüllt werden, da entweder die Bedienung umständlich wirkt oder eine Interaktion zwischen Kursbetreuerin und Studierende nicht vorgesehen ist.

1.4 Anforderungen

Die Anforderungen an das System wurden in der Problemstellung vorgegeben. Aus den Vorgaben wurden Userstories abgeleitet. Der Vorteil von Userstories ist, das diese in Kurzform fachliche Anforderungen beschreiben, die eine Anwenderin in einer bestimmten Rolle stellt, um das System nutzen zu können.

Die nachfolgende Tabelle zeigt die vereinbarten Anforderungen sowie die Bearbeitungsreihenfolge auf:

ID	Titel	Beschreibung	Akzeptanzkriterien
1	Auswertung spezifischer Fragen	Als Anwenderin möchte ich eine spezifische Frage mit bis zu acht Antwortmöglichkeiten beantworten können. Beispiel: 'Wie hoch ist die Mwst in D.?'	Frage und Antwortmöglichkeit wird angezeigt. Es können mehrere Antworten ausgewählt werden.
2	Auswertung generischer Fragen	Als Anwenderin möchte ich eine generische Frage mit bis zu acht Antwortmöglichkeiten beantworten können. Beispiel: Welche der folgenden Aussagen ist richtig?	Mehrere Antworten können richtig sein.
3	Themengebiet wählen	Als Anwenderin möchte ich wählen, zu welchen Themengebieten Fragen gezogen werden.	Es können mehrere Themengebiete ausgewählt werden. Es werden nur Fragen aus gewählten Themengebieten gezogen. Wenn ein Themengebiet weniger Fragen enthält, dann werden nur diese angezeigt.
4	Fragen anfordern	Als Anwenderin möchte ich, dass ein Satz von zehn Fragen abgerufen wird.	Es werden bis zu zehn Fragen angezeigt. Bereits korrekt beantwortete Fragen erscheinen mit kleinerer Wahrscheinlichkeit. Die Frage muss nicht sofort beantwortet werden, es kann zur nächsten Frage navigiert werden.
5	Datenschutz	Als Anwenderin möchte ich, dass meine Auswertungen vor unberechtigtem Zugriff geschützt sind	Datenschutzvorschriften sind geprüft. Verschlüsselte Datenübertragung kann aktiviert werden.
6	Autorenzugang	Als Kursbetreuerin möchte ich einem bestimmten Themengebiet neue Fragen hinzufügen oder vorhandene Fragen löschen können.	Neue Fragen werden bei Ausführung der Multiple-Choice-App angezeigt. Fragen werden einem Thema zugeordnet. Es können neue Fragen erstellt oder vorhandene Fragen geändert oder gelöscht werden.

7	Auswertung	Als Anwenderin möchte ich, dass nach Bearbeitung angezeigt wird, welche Fragen falsch beantwortet wurden.	Richtig beantwortete Fragen werden angezeigt. Falsch beantworte Fragen werden angezeigt. Richtige Antworten werden angezeigt.
8	Meldung Benutzung der App	Als Kursbetreuerin möchte ich ein Auswertung darüber erhalten, wie oft die App in den letzten 7/30 Tagen verwendet wurde.	Meldung wird nur Kursbetreuerin angezeigt. Meldung zeigt korrekte Anzahl der App-Aufrufe an. Daten werden anonymisiert und nutzerunabhängig gespeichert. Auswertung wird über Autorenzugang geöffnet.
9	Meldung Auswertung von Fragen	Als Kursbetreuerin möchte ich eine Auswertung darüber erhalten, wie häufig die Fragen richtig oder falsch beantwortet wurden.	Meldung wird nur Kursbetreuerin angezeigt. Meldung zeigt an, wie häufig die Fragen richtig und falsch beantwortet wurden. Daten werden anonymisiert und nutzerunabhängig gespeichert. Auswertung wird über Autorenzugang geöffnet.
10	Bilder als Antwortmöglichkeiten	Als Anwenderin möchte ich Fragen bearbeiten können, für die, durch die Kursbetreuerin, Bilder als Antwortmöglichkeiten hinterlegt wurden.	Als Antwortmöglichkeit wird Bild angezeigt. Bild wird komplett angezeigt.
11	Bilder für Antworten hinterlegen	Als Kursbetreuerin möchte ich für eine Frage als Antwortmöglichkeit ein Bild hochladen	Es werden JPG-Dateien akzeptiert. Datei kann mithilfe eines Auswahldialogs gewählt werden.
12	Anzahl Fragen festlegen	Als Anwenderin möchte ich einstellen, wieviele Fragen gezogen werden.	Anzahl Fragen ist einstellbar. Es werden soviele Fragen gezogen, wie eingestellt.
13	Login	Als Kursbetreuerin möchte ich den Zugriff auf die Fragen und Auswertungen mit einem Benutzerkonto schützen.	Zugang zur Webanwendung nur möglich, nach Eingabe der korrekten Username/Kennwort-Kombination.
14	Offline-Implementierung	Als Anwenderin möchte ich die Multiple-Choice-App auch dann benutzen können, wenn der Server nicht erreichbar ist.	Vorhandene Fragen können bearbeitet werden, sofern zuvor bereits eine Verbindung zum Server hergestellt wurde.
15	Themen bearbeiten	Als Kursbetreuerin möchte ich die Liste der möglichen Themen bearbeiten können.	Thema kann angelegt und bearbeitet werden. Zu jedem Thema wird die Anzahl der zugeordneten Fragen angezeigt.

Kapitel 2

Fachliche und technische Grundlagen

2.1 Rollen und Anwendungsfälle

Studierende und Kursbetreuerin

Das System sieht die beiden Rollen Studierende sowie Kursbetreuerin vor. Die Studierende nutzt die Multiple-Choice-App, um bestimmte Themen, mithilfe von Übungsfragen, zu lernen. Die Kursbetreuerin nutzt den Autorenzugang, um Fragen hinzuzufügen oder zu bearbeiten. Weiterhin wertet die Kursbetreuerin die Nutzung der App sowie die beantworteten Fragen aus.

Die Abbildung 2.1 auf Seite 6 zeigt die Userstories als Anwendungsfälle in einem Anwendungsfalldiagramm. Ein Anwendungsfalldiagramm dokumentiert alle gewünschten Anwendungsfälle eines Softwaresystems in einer überblicksartigen grafischen Darstellung[19, KE 2, s.S.10]. Das Anwendungsfall zeigt deutlich die Geschlossenheit der beiden Teilsysteme, wobei mit Teilsystemen einerseits die Multiple-Choice-App gemeint ist, andererseits der Autorenzugang als Webanwendung. Das Diagramm zeigt, dass die Studierende die App nutzt, um Übungsfragen zu bearbeiten, aber selbst keine Möglichkeit hat, diese Fragen zu verändern. Die Kursbetreuerin bearbeitet die Inhalte der Multiple-Choice-App. Die Kursbetreuerin hat keine direkten Kenntnisse über die Auswertungen der Studierenden. Dennoch erhält die Kursbetreuerin Auswertungsmöglichkeiten, um festzustellen, wie häufig die richtig oder falsch beantwortet wurden.

2.2 Auswahl der verwendeten Technologien der Multiple-Choice-App

Für die Umsetzung der Multiple-Choice-App existieren mehrere mögliche Technologien. In den folgenden Kapiteln werden einige dieser Möglichkeiten erläutert und die Festlegungen auf bestimmte Technologien weitestgehend begründet.

2.2.1 iOS

Bei iOS handelt es sich um ein Betriebssystem, welches für die mobilen Geräte des Unternehmens Apple Inc. entwickelt wurde. Eine erste Version des Betriebssystems wurde im Jahr 2007 für das erste iPhone-Modell veröffentlicht [21]. Im Lauf der Zeit wurden weitere iPhone-Modelle sowie weitere Versionen des Betriebssystems entwickelt. Desweiteren wurde das Betriebssystem

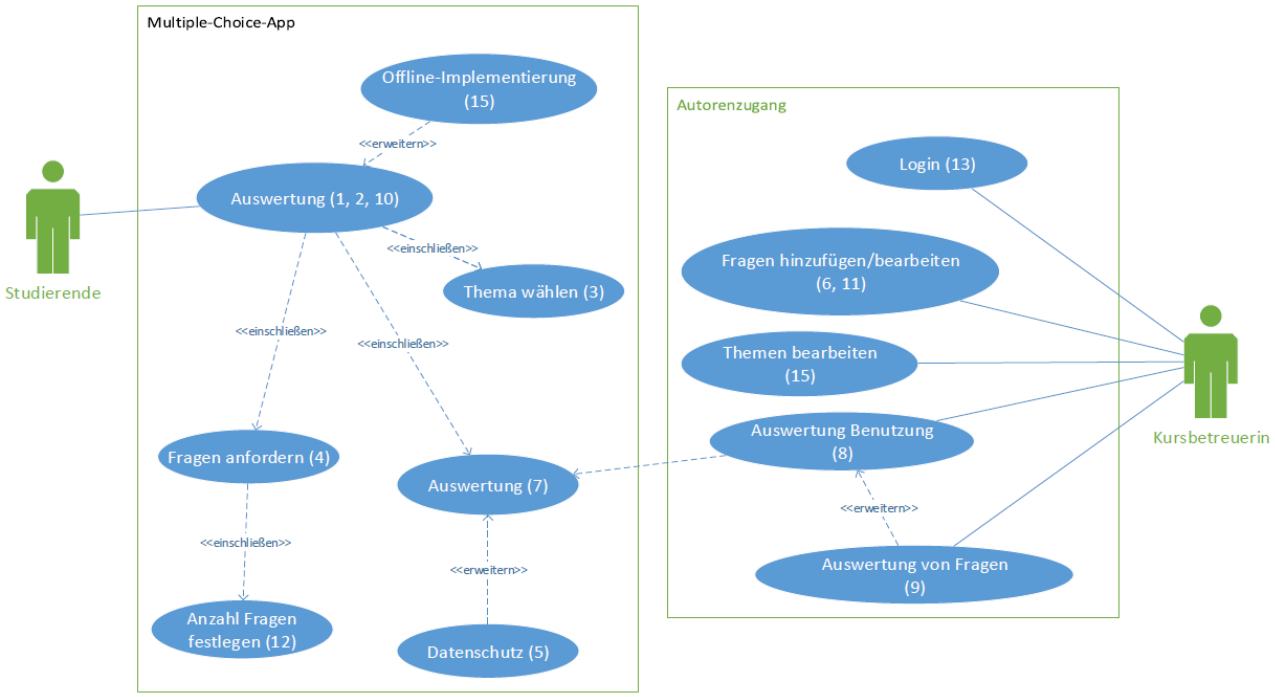


Abbildung 2.1: Anwendungsfalldigramm

auch für andere Apple-Geräte, nämlich dem iPad und dem iPod, eingesetzt. Im Jahr 2010 erhielt die vierte Version dieses Betriebssystems den Namen iOS. Im Jahr 2018 wurde iOS in Version 12 veröffentlicht [22].

Das iOS System basiert auf den gleichen Technolgien, welche auch von MacOS - dem Betriebssystem auf Apple Computern - verwendet wird. Somit handelt es sich bei iOS um ein Betriebssystem, welches auf UNIX aufgebaut wurde[23].

Swift

Das Unternehmen Apple hat für die Entwicklung von Anwendungen für iOS-basierte Geräte die OpenSource-Programmiersprache Swift entwickelt und bereitgestellt [25]. Swift ist, nach Angaben des Herstellers, eine robuste und intiuitive Programmiersprache, die es ermöglicht, schnelle und effiziente Anwendungen zu schreiben. Swift wurde im Juni 2014 vorgestellt [26] und ist aktuell in Version 4.2 veröffentlicht. Bevor Apple die Entwicklung und Verwendung von Swift vorangetrieben hat, waren Objective-C sowie C die vorherrschenden Programmiersprachen zur Entwicklung von Anwendungen im Apple-Universum [24, s. S. 339]. Für die Entwicklung von iOS-Apps stellt Apple die Entwicklungumgebung Xcode bereit. Xcode enthält, nach Angaben des Herstellers, alle notwendigen Funktionen, um Apps für alle Apple-Plattformen umzusetzen [27]. Dazu gehören ein Editor zum Schreiben und Bearbeiten des Quellcodes sowie ein Compiler, um den Quellcode auf verschiedenen Apple-Geräten auszuführen. Mit Xcode wird somit eine App für eine bestimmte iOS-Version entwickelt und diese App kann auf verschiedenen Apple-Geräten emuliert werden. Xcode steht kostenlos im Apple App Store zur Verfügung. Xcode kann jedoch nicht auf einem anderen Betriebssystem betrieben werden. Dies bedeutet vorallem, dass für die Entwicklung von iOS-Apps grundsätzlich ein Apple-Gerät benötigt wird.

Objective-C und C

Xcode

2.2.2 Auswahl der Entwicklungsumgebung und der verwendeten Frameworks

Aus dem vorherigen Abschnitt ist leicht ersichtlich, dass das Unternehmen Apple für die Entwicklung von Apps für iOS die eigene Entwicklungsumgebung Xcode vorgesehen hat. Desweiteren wird die Entwicklung einer iOS-App erschwert, wenn die App auf einem System umgesetzt werden soll, auf dem XCode nicht zur Verfügung steht, da die entwickelte Anwendung ohne Xcode nicht gestartet werden kann. Damit die Entwicklung von Apps auf einer anderen Plattform möglich ist, hat Apple eine Schnittstelle bereitgestellt, die es über eine entfernte Anmeldung an einem Mac-Computer ermöglicht, eine App auszuführen. Daraus ergibt sich einerseits die Möglichkeit, eine App ohne die direkte Verwendung von Xcode umsetzen, andererseits wird es somit möglich, eine App für mehrere Plattformen zu entwickeln, da Xcode nur die eigene Plattform unterstützt.

Nach Benett gibt es vier verschiedene mobile Entwicklungsplattformen [1, s.S. 4]:

- Anbieter-spezifische Apps mithilfe der anbieterspezifischen Entwicklungsumgebungen bspw. von Apple oder Google
- Cordova - plattformübergreifend auf Basis von HTML, JavaScript und CSS
- native Xamarin Anwendungen auf Basis von Xamarin.iOS und Xamarin.Android
- Xamarin.Forms - welches die XAML-Syntax zum Erstellen der Benutzeroberflächen verwendet

Xamarin

Die Abbildung 2.2 vergleicht die vier verschiedenen Plattformen hinsichtlich der verwendeten Programmiersprachen und ordnet diese den Schichten

- Anwendungsschicht,
- Benutzeroberfläche,
- sowie Geschäftslogik

zu.

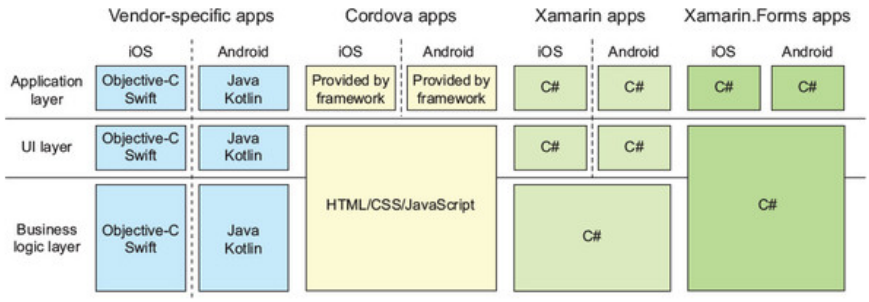


Abbildung 2.2: Vergleich der mobilen Entwicklungsplattformen [1, s.S. 4]

Die Umsetzung der Multiple-Choice-App mit Xcode erfordert die Einarbeitung in die Programmiersprache Swift. Desweiteren wird ein MacBook benötigt, auf dem Xcode verwendet werden kann. Der Vorteil an dieser Variante ist, dass bei der Umsetzung auf eine umfangreiche Literatur zurückgegriffen werden kann, da die Entwicklungsumgebung von Apple aufgrund ihrer Bekanntheit und der kostenlosen Bereitstellung sehr verbreitet ist. Dieses Vorgehen schließt

Xcode

jedoch die Entwicklung dieser App für andere Plattformen aus, da der Swift-Quellcode nicht für andere Plattformen verwendet werden kann.

Cordova

Cordova ist ein Framework zur Erstellung von Webanwendungen mithilfe von Standardtechnologien wie HTML, JavaScript und CSS [1, s.S. 6]. Das Framework unterstützt die Aufbereitung der Webanwendungen für verschiedene Plattformen wie Android, Windows oder iOS [29]. Dies hat den Vorteil, dass nur eine Webanwendung zu erstellen ist. Diese Webanwendung wird nun innerhalb von plattformspezifischen Apps, mithilfe einer WebView, angezeigt und bedient. Bei einer WebView handelt es sich um eine Art Browser, innerhalb einer App. Das Framework sorgt jedoch zusätzlich dafür, dass einige Besonderheiten der jeweiligen Plattform, bspw. Sensoren oder Dienste, genutzt werden können. Ein weiterer Vorteil ist, dass die Einarbeitung in eine weitere Programmiersprache entfällt, sofern die Entwicklerin Standardtechnologien der Web-Entwicklung beherrscht. Laut Bennet können in Cordova jedoch nur Hardware und Betriebssystem-Dienste genutzt werden, welche in allen Plattformen vorhanden sind. Desweiteren wird das Framework erst einige Zeit nach Veröffentlichung einer neuen Version eines Betriebssystems aktualisiert[1, s.S. 7].

Xamarin.iOS

Mithilfe von Xamarin.iOS und Xamarin.Android können anbieterspezifische Apps mithilfe des .NET-Frameworks entwickelt werden [1, s.S. 6]. Dies bedeutet, dass der Quellcode mit der Entwicklungsumgebung Visual Studio bspw. in der Programmiersprache C# geschrieben werden kann und mithilfe von Xamarin in eine native iOS- oder Android-App überführt wird. Dies ist möglich, da Xamarin eine Auswahl von .NET-Komponenten bereitstellt, welche im Hintergrund die nativen Schnittstellen von iOS oder Android aufrufen [1, s.S. 7]. Laut Anbieter besitzen die, mit Xamarin erstellten, Apps den Zugriff auf das gesamte Funktionsspektrum, das von der zugrunde liegenden Plattform und dem zugrunde liegenden Gerät bereitgestellt wird [5]. Diese Variante kombiniert die Vorteile der beiden vorangegangenen Plattformen, da einerseits eine App entwickelt werden kann, welche anbieterspezifische Dienste nutzen kann, andererseits der größere Teil des Quellcodes für mehrere Plattformen genutzt werden kann. Xamarin steht als Plugin für Visual Studio zur Verfügung und kann entweder auf einem Mac- oder auf einem Windows-Betriebssystem verwendet werden. Damit eine, mit Xamarin-entwickelte, App auf einem Windows-Betriebssystem gestartet werden kann, ist eine Remote-Verbindung zu einem Mac-Computer, mit einer Xcode-Installation, notwendig. Als Nachteil sind die zusätzlichen Lizenzkosten für Visual Studio zu erwähnen. Desweiteren ist die verfügbare Literatur zum Thema Xamarin überschaubar, da diese Entwicklungsplattform, im Vergleich zu Xcode, noch wenig verbreitet ist.

Xamarin.Forms

Mithilfe von Xamarin.Forms ist eine nahezu vollständige Wiederverwendbarkeit des Quellcodes für beide Plattformen möglich. In den zuvor vorgestellten nativen Xamarin-Produkten Xamarin.iOS und Xamarin.Android wird zwar die Geschäftslogik sowie das View-modell für alle Plattformen verwendet, die eigentlichen Benutzeroberflächen werden jedoch speziell für jede Plattform umgesetzt. Bei Verwendung von Xamarin.Forms werden keine anbieterspezifischen Benutzeroberflächen erstellt, sondern die Benutzeroberflächen werden mit einer XML-ähnlichen Syntax namens XAML definiert. Dies erschwert jedoch wiederum die Verwendung von anbieterspezifischen Diensten, ähnlich wie es bei Cordova der Fall ist.

Von den vier vorgestellten Möglichkeiten, stellt sich Xamarin.iOS als die beste Wahl heraus. Da das Lehrgebiet eine Möglichkeit wünscht, die Multiple-Choice-App für mehrere Plattformen zu entwickeln, kommt eine Umsetzung mit Xcode nicht in Frage, da ansonsten die App doppelt entwickelt werden müsste. Cordova bietet sich vorallem an, wenn zusätzlich zu einer iOS-App auch eine Webanwendung für die Studierenden gefordert wäre und keine vollumfängliche iOS-App

benötigt wird. Die Plattformen Xamarin.iOS und Xamarin.Android ermöglichen die plattformübergreifende Entwicklung und bieten aufgrund der Nähe zu Microsoft Visual Studio, eine hohe Unterstützung für die Umsetzung der Anforderungen. Desweiteren lässt sich feststellen, dass die Unterstützung der Apple- und Android-Plattformen hervorragend funktioniert. Xamarin.Forms hingegen erscheint zum gegenwärtigen Zeitpunkt noch etwas zu entfernt von der Zielplattform, da mit Xamarin.Forms noch keine anbieterspezifischen Dienste genutzt werden können. Desweiteren existiert für Xamarin.Forms kein Editor, mit dem Benutzeroberflächen komfortabel erstellt werden könnten.

2.2.3 Microsoft Visual Studio

Für die Umsetzung der Multiple-Choice-App wird die Entwicklungsumgebung Microsoft Visual Studio Enterprise 2017 verwendet.

Die Abbildung A.1 im Anhang auf Seite A zeigt die benötigten Komponenten, damit mit Visual Studio eine iOS-App umgesetzt werden kann. Im Wesentlichen werden die Komponenten '.NET-Desktopentwicklungstools', sowie 'Mobile-Entwicklung mit .NET' benötigt. Für die Entwicklungsumgebung steht auch die Komponente 'Mobile-Entwicklung mit JavaScript' zur Auswahl, welche es ermöglicht, Cordova einzusetzen.

NuGet-Pakete

Für die Entwicklung der Multiple-Choice-App werden für die einzelnen Projekte zusätzliche Erweiterungen in Form von Paketen benötigt. Die einzelnen Pakete werden in Visual Studio mithilfe des NuGet-Paketmanagers ausgewählt und den Projekten hinzugefügt. Die Abbildung A.2 im Anhang auf Seite B zeigt die Pakete an, welche für die Erstellung der Multiple-Choice-App verwendet wurden.

Im Einzelnen sind das:

- MvvmCross 5.2.1 - unterstützt das MVVM-Pattern und verbindet Oberflächen mit View-Modell-Objekten
- PCLStorage 1.0.2 - unterstützt den Datenzugriff auf die Geräte
- sqlite-net-pcl 1.5.231 - ermöglicht die Verwendung von SQLite-Datenbanken
- NUnit 3.10.1 - unterstützt das Testen von Quellcode durch Unittests[63]
- Moq 4.0.0 - unterstützt das Schreiben von Unittest durch sog. Mocks[64]
- NUnit3TestAdapter3 - wird zum Ausführen der Unittests benötigt

Visual Studio mit Mac koppeln

Damit die Multiple-Choice-App auf einem Mac-Gerät gebaut werden kann, muss die Entwicklungsumgebung mit diesem Mac-Gerät gekoppelt werden. Die Abbildung 2.3 skizziert das Zusammenspiel zwischen einer Visual Studio Instanz auf einem Windows PC, der Verbindung zu einem Mac-Computer und der Simulation eines Apple-Endgerätes. Der Quellcode wurde auf dem Windows Computer geschrieben und an den Mac Computer gesendet. Auf dem Mac Computer wird der Quellcode übersetzt und ausgeführt. Das Ergebnis wird wiederum auf dem Windows Computer angezeigt. Die geschriebene App wird somit auf einem emulierten iPhone-Modell am Windows Rechner bedient, aber auf dem Mac Computer ausgeführt.

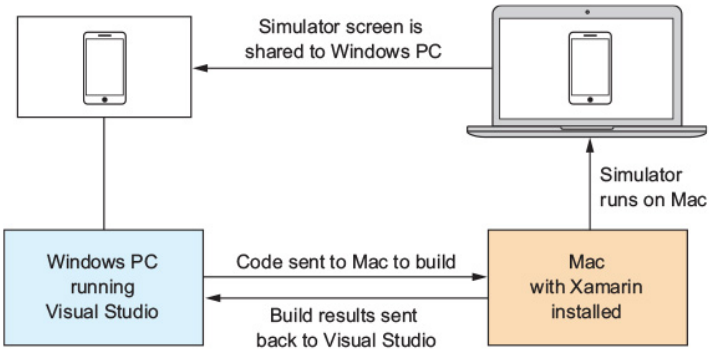


Abbildung 2.3: Verbindung mit Mac[1, s.S. 13]

Damit die Verbindung zwischen dem Windows Computer und dem Mac-Computer möglich ist, muss auf dem Mac-Computer unter Systemeinstellungen im Menü 'Freigaben' die entfernte Anmeldung aktiviert werden. Sofern sich der Mac Computer im gleichen Netzwerk befindet und erreichbar ist, kann nun eine Anmeldung durch den Windows Computer erfolgen. Es ist ebenfalls möglich, eine Verbindung zu einem virtuellen Mac Computer herzustellen, welcher in einem Cloud-Netzwerk bereitgestellt wurde. Für die entfernte Anmeldung muss dann der Mac Computer über eine Internetadresse erreichbar sein, desweiteren muss ein entsprechender Port für die Anmeldung aktiviert werden[14]. Die Abbildung A.3 im Anhang auf Seite C zeigt die zu setzende Einstellung für die entfernte Anmeldung.

Damit der Quellcode übersetzt und mithilfe des Mac Computers ausgeführt werden kann, muss die Entwicklungsumgebung mit dem Mac gekoppelt werden. Hierzu sind die IP-Adresse des Mac Computers, sowie die Zugangsdaten für die entfernte Anmeldung, einzugeben. Die Abbildung A.4 im Anhang auf Seite D zeigt den angezeigten Bildschirm, nachdem Visual Studio mit dem Mac Computer gekoppelt wurde. Sobald die Verbindung zu dem Mac Computer hergestellt wurde, kann die App ausgeführt werden. Desweiteren ist es nun möglich, anbieterspezifische Komponenten, wie bspw. Storyboards, zu öffnen und zu bearbeiten.

2.2.4 MVVM (model-view-view model)

Mit MVVM wird ein Entwurfsmuster bezeichnet, welches für plattformübergreifende Apps weit verbreitet ist [1, s.S. 27]. MVVM wird verwendet, um die Benutzeroberflächen verschiedener Plattformen, sowohl von der Anzeige-Logik, als auch von der Datenverarbeitung zu trennen. Dies ermöglicht eine höhere Wiederverwendbarkeit des Quellcodes, da für die verschiedenen Plattformen nur noch die Oberflächen entwickelt werden müssen, während die Verarbeitung der Daten von verschiedenen Plattformen gleichermaßen verwendet wird.

Die Abbildung 2.4 zeigt die einzelnen Schichten einer plattformübergreifenden App und wie diese dem MVVM-Pattern zugeordnet sind. Die Idee dahinter ist, dass die Benutzeroberflächen an das View-Modell gebunden werden und somit weniger Quellcode entwickelt werden muss, um das View-Modell in der Benutzeroberfläche zu implementieren.

Um das MVVM-Pattern besser zu verstehen ist es hilfreich, die einzelnen Schichten näher zu betrachten:

- Modell - diese Schicht enthält die Datenmodelle, sowie die Geschäftslogik, um die Daten bereitzustellen und zu verarbeiten. Im Fall der Multiple-Choice-App werden in dieser Schicht bspw. die Fragen geladen und ausgewertet
- View - diese Schicht enthält lediglich die benötigten Klassen der Benutzeroberfläche. Dies sind bspw. Storyboards für den Startbildschirm oder Tabellen, welche die Fragen anzeigen. Die einzelnen Oberflächenelemente sind an das View-Modell gebunden. In der View-Schicht findet keinerlei Verarbeitung statt, sondern es werden stets die Werte angezeigt, welche das gebundene View-Modell liefert. Sofern der Anwender mit der Benutzeroberfläche interagiert, so wird dies in Form eines Ereignisses an das View-Modell übertragen und das View-Modell liefert dann das zurück, was in der Benutzeroberfläche angezeigt werden soll.
- View-Modell - in dieser Schicht werden die benötigten Daten aus der Modell-Schicht geladen und für die View aufbereitet. Das View-Modell nimmt desweiteren Ereignisse wie bspw. die Betätigung einer Schaltfläche entgegen und führt daraufhin die gewünschte Aktion aus. So werden bspw. nach Betätigung der Start-Schaltflächen, die zuvor gewählten Themen an einen Service der Modell-Schicht gemeldet. Das View-Modell führt daraufhin eine Navigation zu einem anderen Storyboard durch. Innerhalb des View-Modells werden desweiteren die Inhalte so aufbereitet, dass diese in der Benutzeroberfläche angezeigt werden können.

MVVMCross

Für das Binden zwischen den Views und den View-Modellen existieren mehrere Frameworks. Für die Multiple-Choice-App kommt das Framework MVVMCross[31] zum Einsatz. Das Framework MVVMCross unterstützt bei der Kopplung zwischen Views und View-modellen mithilfe weniger Standardmethoden und Annotationen. Dies erspart zusätzlichen Entwicklungsaufwand, im Vergleich zu einer eigenen Implementierung des MVVM-Entwurfmusters.

2.3 Technologieauswahl für Webanwendung

Da für die Umsetzung der Webanwendung verschiedene Technologien in Frage kommen, werden diese in den folgenden Abschnitten näher betrachtet und erläutert.

2.3.1 Client-Serverarchitektur

Client-Serverarchitektur

Aus den Anforderungen zum Autorenzugang geht hervor, dass ein Webserver mit Datenbank zum Einsatz kommt. Dieser Webserver stellt die Themen und Fragen für die Multiple-Choice-App bereit. Zu diesem Webserver gehört ein Client in Form einer Webanwendung, welcher die Erfassung und Bearbeitung dieser Daten ermöglicht.

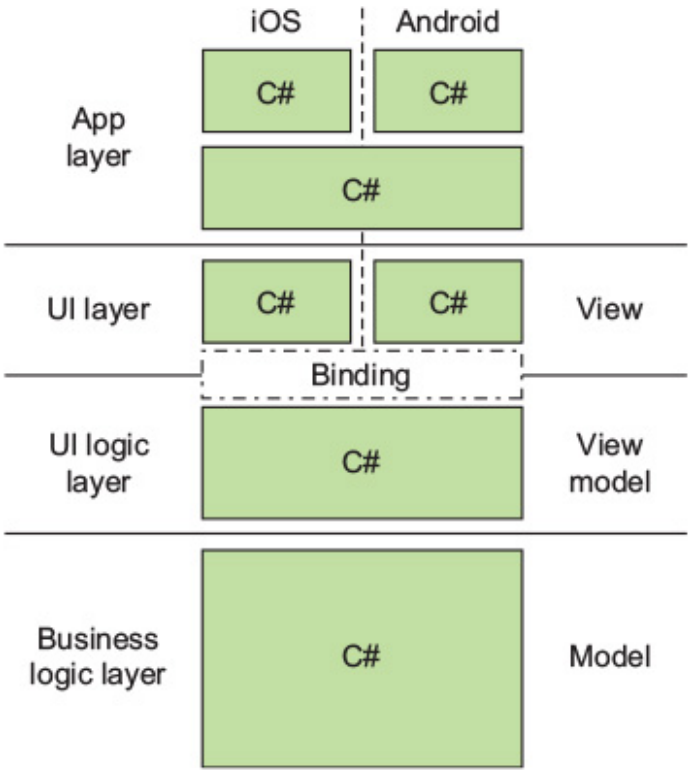


Abbildung 2.4: Model-View-View Model in .NET[1, s.S. 33]

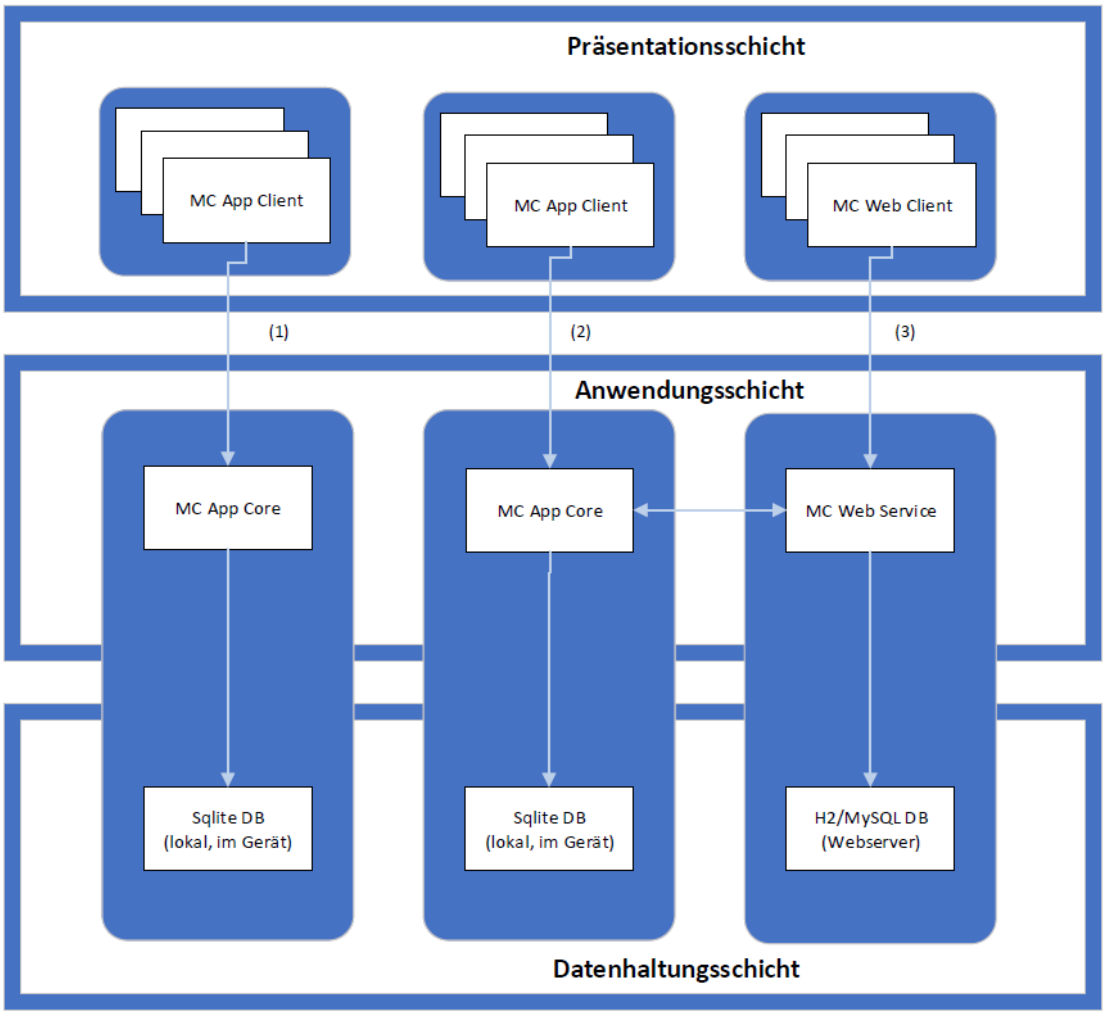


Abbildung 2.5: Client-Serverarchitektur Multiple-Choice-App[20, KE 2 s.S. 68]

Die Abbildung 2.5 zeigt die verwendete Client-Serverarchitektur, welche das Anwendungssystem auf mehrere, miteinander vernetzte Geräte verteilt [20, KE 2 s.S. 68]:

- die Multiple-Choice-App gehört zur Präsentationsschicht und greift als Client auf die Anwendungsschicht zu, um die vorhandenen Fragen und Themen zu laden und anzuzeigen. Die benötigten Informationen werden in der Datenhaltungsschicht aus der Datenbank des Gerätes geladen (1). Dieser Fall tritt ein, wenn der Webserver nicht erreichbar ist, aber zuvor bereits Daten in die Datenbank des Endgerätes geschrieben wurden.
- die Multiple-Choice-App greift auf die Anwendungsschicht zu. Sofern der Webserver erreichbar ist, erfolgt der Aufruf der Webservices, damit Fragen und Themen geladen und in die Datenbank des Endgerätes gespeichert werden. Im Anschluss werden die angefragten Daten aus der lokalen Datenbank gelesen und in der Multiple-Choice-App angezeigt (2).
- die Multiple-Choice Webawendung gehört zur Präsentationsschicht und greift als Webclient auf die Anwendungsschicht zu. In der Anwendungsschicht steht ein Webservice zur

Verfügung. Der Webclient fordert bspw. die Liste der vorhandene Themen an, um diese in der Webanwendung anzuzeigen. Die Daten werden nun in der Multiple-Choice- Webanwendung bearbeitet und beim Speichern der Daten an den Webservice gesendet. Der Webservice greift wiederum auf die Datenhaltungsschicht zu, damit die Änderungen in die Datenbank des Servers geschrieben werden (3). Die Multiple-Choice-App erhält beim nächsten Aufruf die geänderten Daten, durch den Webservice, geliefert und schreibt diese in die Datenbank des Endgerätes.

2.3.2 Webservices

Aus den Anforderungen geht hervor, dass eine Schnittstelle zum Austausch der Daten benötigt wird. Einerseits benötigt die Webanwendung eine Schnittstelle zur Anwendungsschicht, damit Informationen, wie Fragen oder Themen, bearbeitet werden können. Andererseits benötigt die Multiple-Choice-App eine Schnittstelle, um die Daten vom Webserver zu laden, sowie um Daten für die Auswertungen an den Webserver zu senden.

Eine Möglichkeit zum Datenaustausch zwischen den Systemen ist die Verwendung entfernter Methodenaufrufe. Laut Mönch steht für Java-Anwendungen der Remote-Method-Invocation-Mechanismus (RMI) zur Verfügung[20, KE 4, s.S. 22]. Mit RMI könnte bspw. ein Java-Webclient eine entfernte Methode auf einem Server aufrufen. RMI käme daher als Schnittstelle zwischen der Multiple-Choice-Webanwendung und der Anwendungs- bzw. Datenhaltungsschicht auf dem Webserver in Frage. RMI

Bei der Multiple-Choice-App handelt es sich jedoch um eine .NET-Anwendung. Für .NET-Anwendungen kann, laut Mönch, das .NET-Remoting-Rahmenwerk verwendet werden, damit eine Kommunikation zwischen Objekten möglich ist, welche sich in verschiedenen Prozessen befinden[20, KE 4, s.S. 29]. Die Multiple-Choice-App könnte also mithilfe des .NET-Remoting-Rahmenwerks auf die Anwendungs- bzw. Datenhaltungsschicht der Multiple-Choice-Webanwendung zugreifen. .NET-Remoting-Rahmenwerk

Die Festlegung auf RMI oder auf das .NET-Remoting-Rahmenwerk schränkt die Technologiewahl etwas ein, da nun Java oder .NET, als Technologie für alle Systemkomponenten verwendet werden müsste. Damit die benötigte Schnittstelle von allen gängigen Plattformen verwendet werden kann, ist es sinnvoll, anstelle von entfernten Methodenaufrufen, Webservices zu verwenden. Die Definition von Webservices lautet nach Mönch: „Webservices sind selbstbeschreibende, plattformunabhangige, gekapselte Softwarekomponenten. Sie bieten eine Netzwerkschnittstelle zur Veroffentlichung, Lokalisierung und zum entfernten Aufruf ihrer Funktionalitat an. Webservices konnen lose durch Nachrichtenaustausch miteinander gekoppelt werden [20, KE 4, s.S.41]“. Webservices

Für Webservices wurde ursprünglich das SOAP-Protokoll verwendet, welches zum Nachrichtenaustausch im XML-Format diente. Da das SOAP-Protokoll einen gewissen Overhead brachte und die Kommunikation zu rechenintesiv war [35], werden Webservices immer häufiger im REST-Architekturstil (Representational State Transfer) bereitgestellt. SOAP

Bei REST-Services handelt es sich um zustandslose Dienste, deren URL eine Resource repräsentieren, welche mithilfe der vier CRUD-Operatoren aufgerufen und bearbeitet werden[1, S.S.228]. Bei dem Begriff CRUD handelt es sich um ein Akronym, welches die vier Basis Typen für die Datenspeicherung beschreibt: Create, Read, Update und Delete [36, s.S. 381]. Die Abbildung 2.6 verdeutlicht die Funktionsweise von REST-Services. REST-Service CRUD

In dem Beispiel aus der Abbildung erfolgt der Zugriff auf eine Ressource über die URL '/messages'. Auf der linken Seiten wird die Ressource 'Nachricht' mit der ID 1 angefragt. Hierfür stellt der Client einen HTTP GET-Request an die URL '/messages/1'. Desweiteren wurde in den Header ein ApiKey für die Authentifzierung eingetragen. Der REST-Service verarbeitet diesen Request und liefert daraufhin einen Response zurück. Der Respone enthält, im Body-Bereich, die angefragte Nachricht im JSON-Format. GET-Request

JSON ist ein Akronym für 'JavaScript Object Notation' und ist ein einfaches Verfahren, um Daten in eine Zeichenkette zu transformieren. Dabei erfolgt die Schreiben und der Zugriff auf die Informationen mithilfe von Zugriffsschlüssel/Werte-Paaren. Bei den Werten handelt es sich entweder um eine String-Zeichenkette, um eine Zahl oder um weitere Zugriffsschlüssel/Werte-Paare [1, s.S.231]. JSON

Das JSON-Format wird inbesondere im Zusammenhang mit REST-Services und in Javascript-basierten Webanwendungen verwendet. Die Abbildung 2.6 zeigt, innerhalb des HTTP GET-Response, im Body-Bereich ein vereinfachtes Beispiel für ein JSON-Dokument. Das JSON-Dokument enthält die beiden Schlüssel 'id' mit dem Wert 1 sowie 'message' mit dem Inhalt 'Hi!'.

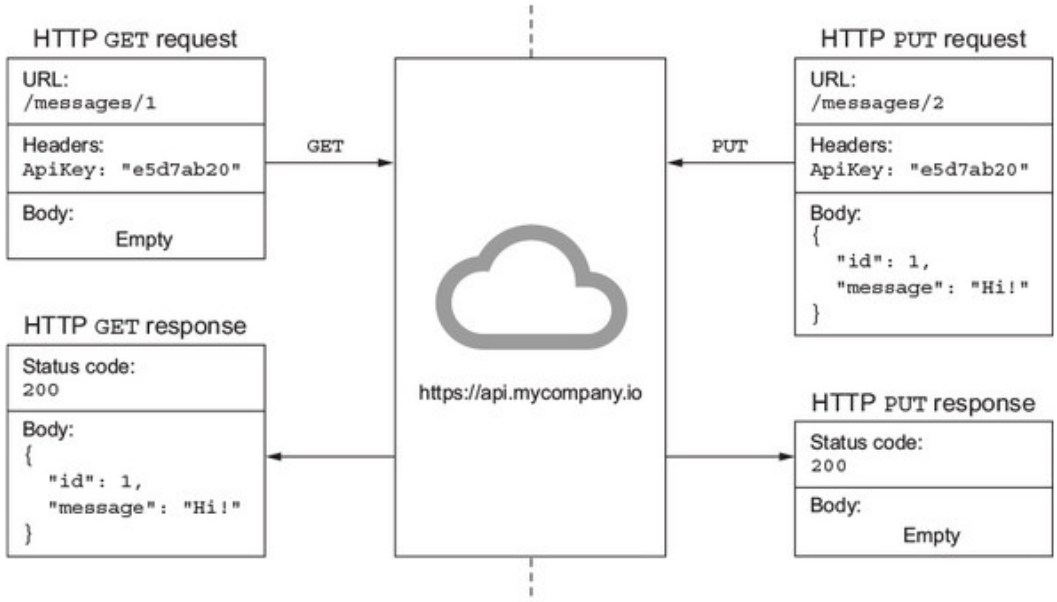


Abbildung 2.6: Beispiel für REST API[1, s.S. 228]

Die Abbildung 2.6 zeigt auf der rechten Seite ein Beispiel für einen PUT-Request. In diesem Fall wird eine Nachricht an den REST-Service gesendet. Die Nachricht wird im Body-Bereich des PUT-Requests hinterlegt. Der REST-Service liefert daraufhin einen Response mit dem http-Status 200. Dieser http-Status sagt aus, dass der Request erfolgreich verarbeitet wurde[37]. PUT-Request

2.3.3 Auswahl Entwicklungsplattform

Im Abschnitt 2.2.2 auf Seite 7 wurde gezeigt, welche Entwicklungsplattformen zur Umsetzung der Multiple-Choice-App in Frage kommen und aufgrund einiger Vorteile wurde Visual Studio

als Entwicklungsumgebung ausgewählt. Es liegt nun nahe, mit Visual Studio auch die Webanwendung umzusetzen. Dies hätte den Vorteil, dass nur eine Entwicklungsumgebung einzurichten ist und mit C# die Einarbeitung in nur einer Programmiersprache notwendig wäre.

Laut Theis wird in Visual Studio beim Start einer Webanwendung automatisch ein interner Webserver aufgerufen [3, s. S. 415]. Der Anbieter sieht also vor, dass als Webserver die Dienste-Plattform 'Internet Information Services (IIS)' verwendet wird [34]. Es wäre folglich möglich, mithilfe von Visual Studio, die für den Autorenzugang benötigten Dienste und Webseiten umsetzen und Daten für die Multiple-Choice-App bereitzustellen.

Da die Bearbeitungszeit für die Umsetzung der Multiple-Choice-Webanwendung jedoch sehr eingeschränkt ist, wurden verschiedene Frameworks untersucht, welche einen Teil der benötigten Dienste und Webseiten mit überschaubarem Aufwand bereitstellen. Einige dieser Frameworks sind für Java-Anwendungen erhältlich und stehen kostenfrei für den Produktivbetrieb zur Verfügung.

Im einzelnen handelt es sich um das Spring Framework, welches u. A. verwendet wurde, um die Projektstruktur zu erstellen sowie um die Daten als Webservice bereitzuhalten. Desweiteren wurde JHipster[47] verwendet, um die die fachlichen Objekte sowie das Grundgerüst der Webanwendung zu erstellen.

Spring

JHipster

Die Syntax von C# unterscheidet sich im Vergleich zu Java nur unwesentlich. Desweiteren ist die Kommunikation zwischen der Multiple-Choice-App und der Multiple-Choice-Webawendung über REST-Webservices problemlos möglich. Der zusätzliche Aufwand zur Einrichtung einer zweiten Entwicklungsumgebung ist somit überschaubar und sollte in einem entsprechenden Verhältnis zum Nutzen stehen. Aus diesen Gründen wurde die Multiple-Choice-Webanwendung auf Basis von Java erstellt.

2.3.4 Spring

Spring ist ein open source Framework und wurde von Rod Johnson erstellt und erstmals 2002 in seinem Buch 'Expert One-on-One J2EE Design and Development' beschrieben [39, s.S.4]. Spring wurde erstellt, um die Entwicklung von javabasierten Unternehmensanwendungen zu vereinfachen [39, s.S.xxi]. Spring veränderte somit die Komplexität von Java Anwendungen durch vier Schlüsselstrategien[39, s.S.4]:

Spring Framework

- leichtgewichtige und minimalistische Entwicklung mithilfe von POJOs (einfachen Java-Objekte)
- lose Kopplung mithilfe von Dependency Injection und Schnittstellen-Orientierung
- deklarative Programmierung durch Aspekte und übliche Konventionen
- Reduzierung von häufig verwendetem Code (boilerplate code) mithilfe von Vorlagen

Mit der leichtgewichtigen Entwicklung soll erreicht werden, dass das Schreiben von Java-Klassen auf das nötigste reduziert wird. Als das Springframework geschrieben wurde, da enthielten Java-Klassen in Unternehmensanwendungen (J2EE) größere Mengen von Code, obwohl häufig nur einzelne Methoden oder Variablen benötigt wurden. Die Entwicklung von Unternehmensanwendungen wurde daher mthilfe des Spring-Frameworks vereinfacht. Der JavaEE-Standard wurde im Laufe der Zeit ebenso weiterentwickelt, sodass inzwischen eine leichtgewichtigte Entwicklung von Unternehmensanwendungen auch mit JavaEE möglich ist[40].

Dependency Injection

Mit Dependency Injection wurde erreicht, dass innerhalb von Klassen keine weiteren Klassen instanziert werden müssen. In der Abbildung 2.7 wird gezeigt, dass die Klasse Foo ein Exemplar von Bar, sowie ein Exemplar von Baz, injiziert. Bevor Dependency Injection verwendet wurde, mussten innerhalb der Klasse Bar die beiden Exemplare instanziert werden. Dies erschwerte die Modularisierung einer Anwendung, da in der aufrufenden Klasse ggfls. Details zur Erzeugung dieser Objekte enthalten sein mussten. Durch Dependency Injection muss die Klasse Foo zwar die benötigten Klassen Bar und Baz kennen, aber die Erzeugung wird an anderer Stelle durchgeführt. Die verwendeten Klassen Bar und Baz können ggfls. unter Zuhilfenahme von Schnittstellen ausgetauscht werden, ohne dass eine Änderung an der aufrufenden Klasse notwendig ist.

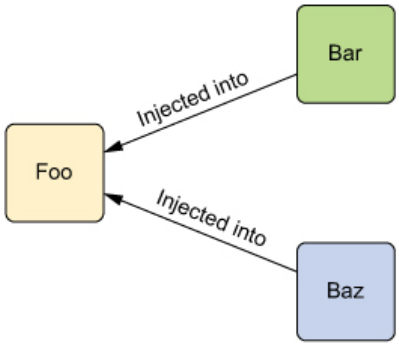


Abbildung 2.7: Dependency Injection[39, s.S. 6]

Aspektorientierte Programmierung

Mithifle von Aspekten wird erreicht, dass in Klassen nur noch fachliche Probleme behandelt werden, während zusätzliche implizite Anforderungen außerhalb der fachlichen Klasse verarbeitet werden. Typische Beispiele hierfür sind die Verwaltung von Datenbanktransaktionen, das Schreiben von Protokollen oder die Datensicherheit. Die Abbildung 2.8 zeigt hierzu fünf fachliche Services für ein Kurs-Buchungssystem. Für die Nutzung aller Servicemethoden werden Datenbanktransaktionen benötigt. Das Erstellen und Schließen dieser Transaktionen wurde jedoch nicht innerhalb dieser Servicemethoden implementiert, sondern wurde mithilfe von Aspekten um die einzelnen Methoden herumgebaut. So wird also beispielsweise bei Buchung eines Kurses eine Datenbanktransaktion geöffnet und nach der Buchung wieder geschlossen.

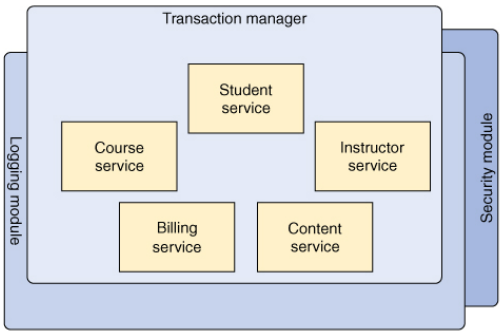


Abbildung 2.8: Aspektorientierte Programmierung[39, s.S. 12]

Bolierplate Code

Die Reduzierung von häufig verwendetem Code, durch Vorlagen, wird am Beispiel einer Datenbankabfrage deutlich. Für Datenbankabfragen wurde häufig ähnlicher Code geschrieben, bspw. um aus einer Datenbankressource eine Verbindung zu erstellen oder um eine Ergebnisliste auszulesen und deren Einträge in Fachobjekte zu schreiben. Das Springframework verwendet hierfür Vorlagen, um das Schreiben von häufig verwendetem Code zu reduzieren.

Spring MVC

Das Spring-Framework besteht aus einer größeren Anzahl von Komponenten, um Unternehmensanwendungen zu schreiben. Zu diesem Komponenten gehört bspw. Spring MVC, welches die Umsetzung von Webanwendungen erleichtert. Eine andere Komponente ist Spring Security, welche zum Einsatz kommt, um den Zugriff auf Daten nur authentifizierten Anwendern zu gestatten. Schließlich bietet das Spring-Framework mit Spring Boot eine Komponente an, welche es ermöglicht, mit wenig Aufwand ein vonkonfiguriertes Projekt für die Multiple-Choice-Webanwendung aufzusetzen.

Spring Security

Spring Boot

Spring Boot besteht aus vier Hauptmerkmalen [39, s.S. 540]:

- Spring Boot starters - die verschiedenen 'Boot Starter' gruppieren mehrere Abhängigkeiten aus üblicherweise benötigten Bibliotheken und fassen diese zu einzelnen Paketen zusammen. Somit lassen sich Projekte mit einer oder mit wenigen Abhängigkeiten erstellen und mithilfe eines Buildsystemes bauen. Zu diesen Buildsystemen gehören Maven[41] sowie Gradle[43].
- Autoconfiguration - Spring Boot prüft, anhand der geladenen Bibliotheken im Klassenpfad, welche Standardkonfigurationen aus Spring benötigt werden und erzeugt diese[39, s.S.546].
- Kommandozeilen Schnittstelle CLI - die Spring Boot CLI ist eine Kommendozeilen-Anwendung. Diese dient zur Ausführung von Befehlen in der Skriptsprache Groovy[44]. Dies ermöglicht die Bereitstellung einer Spring-basierten Anwendung mit wenigen Befehlszeilen.
- Actuator - der Spring Boot Actuator stellt einige nützliche Funktionen für Spring Boot Anwendungen bereit. Dazu gehören einige Services zur Wartung der Anwendung. Mit dem Spring Boot Actuator kann bspw. die Erreichbarkeit der Anwendung geprüft werden oder eine Übersicht der vorhandenen Threads angezeigt werden[39, s.S.565]

Spring Initializr

Um den Einstieg in die Entwicklung einer Spring Boot Anwendung zu erleichtern, hat der Hersteller die Webseite Spring Initializr[38] zur Verfügung gestellt. Auf dieser Seite werden die Parameter der Anwendung sowie die benötigten Abhängigkeiten eingegeben, um eine Spring Boot Anwendung zu erzeugen. Die Abbildung 2.9 zeigt die möglichen Parameter für die Multiple-Choice-Webanwendung.

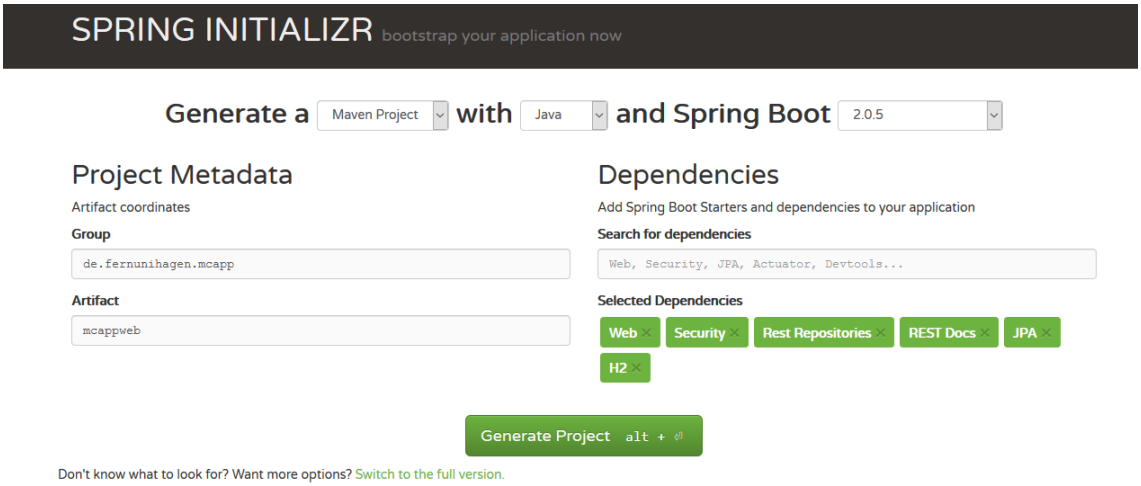


Abbildung 2.9: Spring Boot Projekt' konfigurieren[38]

Zunächst wurde der Name der Paketstruktur als 'de.fernunihagen.mcapp' sowie der Name des Projektartefakts als 'mcappweb' festgelegt. Als Buildsystem wurde Maven ausgewählt.

Bei der Auswahl der Abhängigkeiten sind die Anforderungen, an die Anwendung, zu berücksichtigen. Aus der Anforderungen der Multiple-Choice–Webanwendung geht hervor, dass das Web-Paket benötigt wird. Das Web-Paket beinhaltet einen Webserver sowie Spring MVC. Desweiteren wird das Security-Paket für die Benutzeranmeldung benötigt. Für die Kommunikation zwischen Client und Webanwendung wird REST benötigt. Zuletzt wird für die Datenverarbeitung JPA[45] sowie für das Entwicklungssystem eine H2-Datenbank [46] benötigt.

Nach Betätigung der Schaltfläche 'Generate Project' stehen die Quellen des Projektes als ZIP-Archiv zum Herunterladen bereit. Das erstellte Projekt wird nun einmal mit dem Buildsystem Maven[41] gebaut. Nach dem erfolgreichen Buildprozess lässt sich das Projekt, bspw. über eine Startkonfiguration in der Entwicklungsumgebung, ausführen.

Während der Konzeption der Multiple-Choice-Webanwendung wurde bereits nach kurzer Zeit ein REST-Service zum Laden der Themen erstellt. Der nächste logische Schritt wäre die Umsetzung des Webclients gewesen. Bei der Auswahl des entsprechenden Frameworks wurde auch das Framework JHipster geprüft, welches ebenfalls Spring Boot verwendet, aber die Entwicklung der Multiple-Choice-Webanwendung noch weiter vereinfachen könnte.

2.3.5 JHipster

JHipster ist eine Entwicklungsplattform für die Erstellung, Entwicklung und die Bereitstellung von Spring Boot und Angular[51] Webanwendungen[47]. Dabei nutzt JHipster die Vorteile, die bereits Spring Boot verwendet. Darüberhinaus ist es jedoch möglich, die benötigten Fachobjekte, die hierzu passenden REST-Services sowie ein Grundgerüst der Benutzeroberflächen zu generieren.

Um mit JHipster ein Projekt zu erstellen, muss JHipster zunächst heruntergeladen und installiert

werden. Nach Eingabe des Befehls 'jhipster' in der Kommandozeile des Betriebssystems, ist die Erstellung eines Projektes mithilfe eines Eingabedialogs möglich. Die Abbildung A.5 im Anhang auf Seite E zeigt die Parameter, welche für die Erstellung der Multiple-Choice-Webanwendung verwendet wurden.

Um einige wichtige Parameter zu nennen:

- type of application: die Multiple-Choice-Webanwendung ist eine einfache Webanwendung, ohne Microservice-Architektur[53]. Aus diesem Grund wurde eine monolitische Anwendung ausgewählt
- production database: für den produktiven Betrieb wurde MySQL ausgewählt
- development database: für das Entwicklungssystem wurde eine H2[46]-Datenbank ausgewählt
- build system: für den Buildprozess wurde Maven ausgewählt
- client framework: als Framework für den Webclient wurde Angular 6 gewählt
- internationalization: für die Multiple-Choice-Webanwendung wurde die Sprache 'Deutsch' als native Sprache eingestellt. Zusätzlich wurde die Sprache 'Englisch' ausgewählt

Nach Eingabe der Parameter wurden die Quelldateien des Projektes generiert. Die Verarbeitung endete mit der Ausgabe 'JHipster execution is complete!'. Die Quelldateien enthalten die Javaklasse 'McappWebApp', welche eine Mainmethode für den Start der Webanwendung enthält. Die Webanwendung enthält einen Webserver. Dieser Webserver ist bereits vorkonfiguriert und wird mit dem Start der Mainmethode ausgeführt. Das Projekt ist vor der ersten Ausführung mit dem Buildsystem Maven[41] zu bauen, damit die Quelldateien für die Ausführung auf dem Server übersetzt werden.

2.3.6 Maven Buildprozess

Maven ist ein Standardverfahren, um Software-Komponenten zu erstellen. Mit Maven wird klar definiert, aus welchen anderen Komponenten eine Software besteht. Maven erleichtert desweiteren die projektübergreifende Verwendung dieser Komponenten [42].

Damit ein Maven-Build auf dem Entwicklungssystem möglich ist, sollte zunächst das Programm maven installiert und in den Pfad aufgenommen werden[41]. Sofern die Einrichtung von Maven auf dem Entwicklungssystem nicht gewünscht ist, so kann alternativ der Befehl 'mvnw' aus dem Projektordner aufgerufen werden. Mit 'mvnw' ist es möglich einen Maven-Build ohne explizite Maven-Installation durchzuführen.

Der Befehl, um den Buildprozess zu starten, lautet: 'mvn clean install -DskipTests'. Dies zeigt die Abbildung A.6 im Anhang auf Seite F.

Sofern Maven korrekt eingerichtet wurde, sollte der Buildprozess erfolgreich durchlaufen. Dies zeigt das Ergebnis in Abbildung 2.10. Nach dem erfolgreichen Build sollte die Anwendung ausführbar sein.

Desweiteren enthält der 'target'-Ordner ein Buildartefakt namens 'mcapp-web-0.0.1-SNAPSHOT.war'. Dieses Buildartefakt wird üblicherweise verwendet, wenn die Anwendung im Produktivbetrieb

Buildartefakt

auf einem Webserver bereitgestellt werden soll. Für die Ausführung im Entwicklungssystem ist dies jedoch nicht notwendig, da der Webserver bereits im Spring Boot Start enthalten ist. Näheres wird im Abschnitt 2.3.8 erläutert.

```
[INFO] --- maven-war-plugin:3.2.2:war (default-war) @ mcapp-web ---
[INFO] Packaging webapp
[INFO] Assembling webapp [mcapp-web] in [C:\mcapp\mcapp\MCAPP_WEB\target\mcapp-web-0.0.1-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp webResources [C:\mcapp\mcapp\MCAPP_WEB\src/main/webapp] to [C:\mcapp\mcapp\MCAPP_WEB\target\mcapp-web-0.0.1-SNAPSHOT]
[INFO] Copying webapp resources [C:\mcapp\mcapp\MCAPP_WEB\target\www]
[INFO] Webapp assembled in [2364 msecs]
[INFO] Building war: C:\mcapp\mcapp\MCAPP_WEB\target\mcapp-web-0.0.1-SNAPSHOT.war
[INFO]
[INFO] --- spring-boot-maven-plugin:2.0.5.RELEASE:repackage (default) @ mcapp-web ---
[INFO]
[INFO] --- maven-resources-plugin:3.1.0:copy-resources (docker-resources) @ mcapp-web ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 47 resources
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ mcapp-web ---
[INFO] Installing C:\mcapp\mcapp\MCAPP_WEB\target\mcapp-web-0.0.1-SNAPSHOT.war to C:\Users\Allan\.m2\repository\de\fernunihagen\mcapp\mcappweb\mca
[INFO] Installing C:\mcapp\mcapp\MCAPP_WEB\pom.xml to C:\Users\Allan\.m2\repository\de\fernunihagen\mcapp\mcappweb\mcapp-web\0.0.1-SNAPSHOT\mcapp-
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:23 min
[INFO] Finished at: 2018-11-25T11:54:02+01:00
[INFO] ------------------------------------------------------------------------
```

Abbildung 2.10: Maven Build Ergebnis

2.3.7 Angular

Angular ist ein populäres JavaScript-Framework, das von Google entwickelt wird [52, s.S. XIII]. Mit JavaScript wird ein Teil der Aufgaben einer Webanwendung im Client bzw. im Browser ausgeführt. Angular erleichtert hierbei die Erstellung von Webanwendungen, welche auf JavaScript basieren. Desweiteren unterstützt Angular die Erstellung von Webanwendungen im Architekturstil von Single Page Applications, dies sind Anwendungen, die aus einer einzigen Seite bestehen[52, s.S. XIII].

JavaScript

Single Page Applications

Obwohl es sich bei Angular um ein JavaScript-Framework handelt, wird üblicherweise TypeScript verwendet. Zu dieser Thematik schrieb Steyer wie folgt: 'Angular wurde mit der Sprache TypeScript geschrieben. Dabei handelt es sich um eine Obermenge von JavaScript, die ein Compiler in handelsübliches JavaScript übersetzt. Deswegen ist TypeScript auch für eigene Angular-Projekte die erste Wahl, wenngleich Angular auch JavaScript ... unterstützt' [52, s.S. 13].

TypeScript

Für die Umsetzung des Autorenzugangs wurde Angular verwendet. In den nachfolgenden Abschnitte ab Seite 60 wurden daher einige Beispiele in TypeScript aufgeführt.

2.3.8 IntelliJ

Als Entwicklungsumgebung für die Multiple-Choice-Webanwendung wurde IntelliJ 2018.2 Ultimate Edition verwendet. Die Webanwendung kann grundsätzlich auch mit anderen Java-Entwicklungsumgebungen weiterentwickelt werden.

Die Abbildung 2.11 zeigt die Startkonfiguration der Multiple-Choice-Webanwendung. Diese muss nicht explizit angelegt werden, sondern die Entwicklungsumgebung IntelliJ bereitet diese bereits vor, da Spring Boot sowie die Main-Methode in der Klasse McappWebApp vorhanden ist.

Um die Multiple-Choice-Webanwendung zu starten, ist sicherzustellen, dass die Startkonfiguration 'McappWebApp' ausgewählt ist und anschließend ist der 'Run'-Button zu betätigen.

Undertow

Bei der Ausführung der Multiple-Choice-Webanwendung wird, über Spring Boot, ein Webserver gestartet. Dieser Webserver ist bereits durch Spring vorkonfiguriert und wird nicht explizit administriert. Bei dem verwendeten Webserver handelt es sich um einen Undertow-Webserver[62].

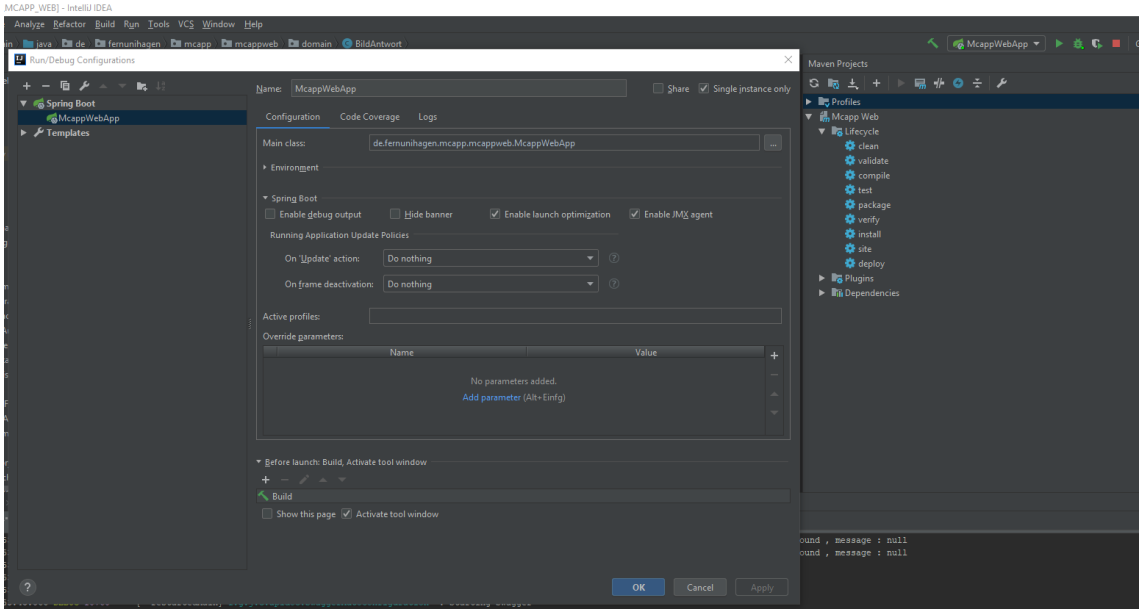


Abbildung 2.11: Startkonfiguration Multiple-Choice-Webanwendung

Nach dem erfolgreichen Start wird in der Konsole angezeigt, dass die Applikation 'mcapp_web' gestartet wurde. Desweiteren wird die URL angezeigt, unter welcher die Webanwendung erreichtbar ist. Dies zeigt die Abbildung 2.12.

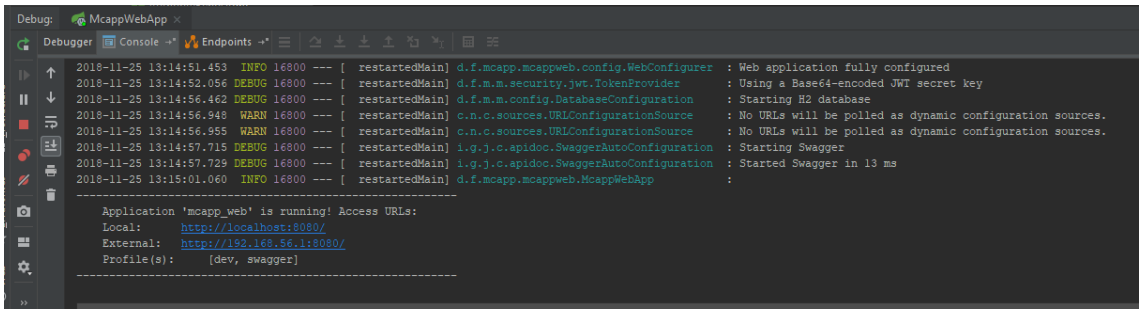


Abbildung 2.12: Start: Multiple-Choice-Webanwendung

2.4 Datenbanken

Für das System werden einerseits Daten direkt auf dem Endgeräten gespeichert, andererseits wird eine Datenbank auf dem Server für den Autorenzugang benötigt.

Auf dem Endgerät werden die Fragen der Kursbetreuerin abgelegt. Desweiteren werden Auswertungen gespeichert, damit u. A. falsch beantwortete Fragen zu einem späteren Zeitpunkt erneut gezogen werden.

SQLite

Für die Datenspeicherung auf dem mobilen Endgerät wird die Datenbank SQLite verwendet. Bei SQLlite handelt es such um eine kleine, schnelle und dateienbasierte Datenbank. SQLite ist sowohl in iOS als auch Android eingebettet [1, s. S. 214].

ORM

Damit eine Anwendung direkt mit der Datenbank interagieren kann kommt ORM (object-relational mapping) zum Einsatz. Mit ORM werden bspw. Datenbanktabellen direkt aus Klassen erstellt und abgefragt. Für die Multiple-Choice-App wird die ORM 'SQLite-Net-Pcl' verwendet [17].

Datenbankmodell App

Für die Multiple-Choice-App wird ein Datenbankmodell benötigt, welches den Offlinebetrieb ermöglicht. Das Datenbankmodell enthält somit Tabellen für Themen, Fragen sowie Antworten. Darüberhinaus enthält das Datenbankmodell die Tabellen Quiz sowie Quiz_Frage, welche die benötigten Daten für die Auswertungen enthalten.

Die Abbildung 2.13 zeigt das verwendete Datenbankmodell.

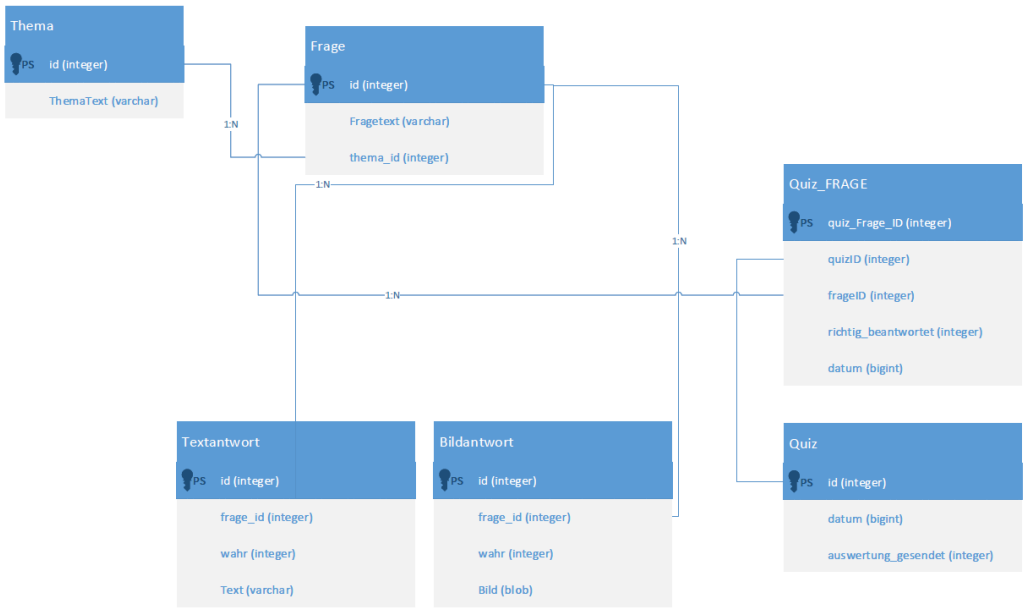


Abbildung 2.13: Datenbankmodell Multiple-Choice-App

Im Einzelnen werden folgende Datenbanktabellen verwendet:

- Thema - entspricht der Fachklasse Thema. Das Thema wird durch einen Thementext beschrieben.
- Frage - entspricht der Fachklasse Frage und enthält die eigentliche Fragestellung in Textform. Die Tabelle enthält weiterhin den Fremdschlüssel thema_id, welches einem Frage-Datensatz das Thema zuordnet.

- Textantwort - beschreibt eine Antwortmöglichkeit in Textform zu einer Frage. Die Antwortmöglichkeit wird als Zeichenkette in die Spalte 'text' gespeichert. Jede Textantwort gehört zu einer Frage. Sofern es sich um eine wahre Antwort handelt, so ist das Datenfeld 'wahr' mit 1 belegt.
- Bildantwort - beschreibt eine Antwortmöglichkeit in Bildform zu einer Frage. Die Antwortmöglichkeit wird als byte-Array in die Spalte 'bild' gespeichert. Jede Bildantwort gehört zu einer Frage. Sofern es sich um eine wahre Antwort handelt, so ist das Datenfeld 'wahr' mit 1 belegt.
- Quiz - jede Ausführung eines Multiple-Choice-Tests wird als 'Quiz' bezeichnet. Ein Datensatz in der Tabelle Quiz entspricht somit einer Ausführung. Zu dem Datensatz gehört das Datum der Ausführung. Desweiteren stellt die Eigenschaft 'auswertung_gesendet' sicher, dass jeder Datensatz nur einmal an den Server gesendet wird.
- Quiz_Frage - zu jedem Quiz gehört ein Satz von Fragen. Ein Datensatz entspricht somit einer beantworteten Frage aus einem Multiple-Choice-Test. Sofern die Frage korrekt beantwortet wurde, so ist der Wert der Spalte 'richtig_beantwortet' mit 1 belegt.

Datenbank für Webanwendung

Auf dem Webserver sind die Daten gespeichert, welche zuvor durch die Kursbetreuerin angelegt wurden. Desweiteren werden von der Anwenderin Auswertungen gesendet und in diese Datenbank geschrieben.

Die Abbildung 2.14 zeigt das Datenbankmodell der Webanwendung. Das Datenbankmodell wurde mit der Webanwendung JDL-Studio[48] erstellt. Dieses Datenbankmodell diente auch als Grundlage, um mit JHipster die benötigten Entitäten-Klassen und Oberflächen zu generieren.

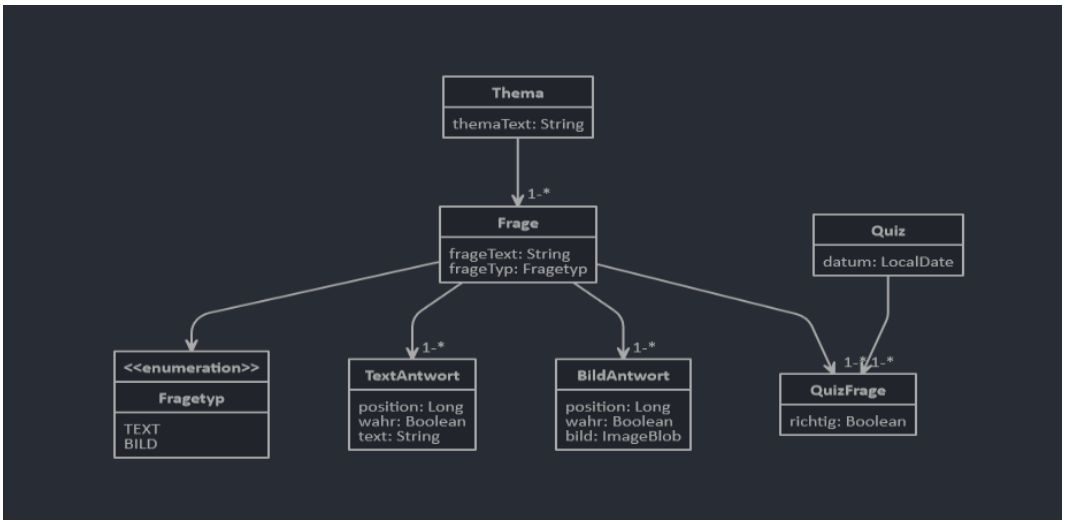


Abbildung 2.14: Datenbankmodell Multiple-Choice-Webanwendung

Das Datenbankmodell der Webanwendung entspricht im Wesentlichen dem Datenbankmodell des mobilen Endgerätes. Die Tabellen enthalten den Primärschlüssel mit dem Namen 'id' sowie die Fremdschlüssel für die Zugriff auf die Tabellen, die in Beziehung stehen. Die Fremdschlüssel werden aufgrund der Notationsform in JDL-Studio jedoch nicht angezeigt.

Die Primärschlüssel der Tabelle Quiz und Quiz_Frage unterscheiden sich wertmäßig von den Primärschlüsseln des mobilen Endgerätes, um Schlüsselverletzungen im Mehrbenutzerbetrieb zu vermeiden.

Desweiteren enthält die Tabelle 'Frage' das Datenfeld 'Fragetyp', welche einen Frage-Datensatz eindeutig als Text- oder Bildfrage deklariert. Dieses Merkmal dient der Erfassung der Antwortmöglichkeiten in der Webanwendung, wird jedoch auf dem Endgerät nicht benötigt. Fragetyp

Wie bereits im Abschnitt 2.3.5 auf Seite 19 erläutert, kommt für die Webanwendung im Entwicklungssystem eine H2 Datenbank zum Einsatz. Die Datenbank wird beim Start der Webanwendung angelegt und enthält zunächst einige Systemtabellen. Dazu gehören bspw. Standardbenutzerkonten für die Kursbetreuerin.

Die fachlichen Tabellen aus der Abbildung 2.14 werden beim ersten Start der Anwendung angelegt. Dies ist möglich, da zuvor mit der Webanwendung 'JDL-Studio' das Datenbankmodell erstellt und mit dem Befehl 'jhipster import-jdl' importiert wurde [49]. Mit diesem Befehl wurden die benötigten Entitäten und Benutzeroberflächen generiert. JDL-Studio

Die Datenbank der Webanwendung kann mit einem Benutzerkonto der Rolle 'Administrator' eingesehen werden. Um die Datenbank zu öffnen, ist innerhalb des Menüs 'Administration' die Schaltfläche 'Datenbank' zu betätigen. Dies zeigt die Abbildung 2.15.

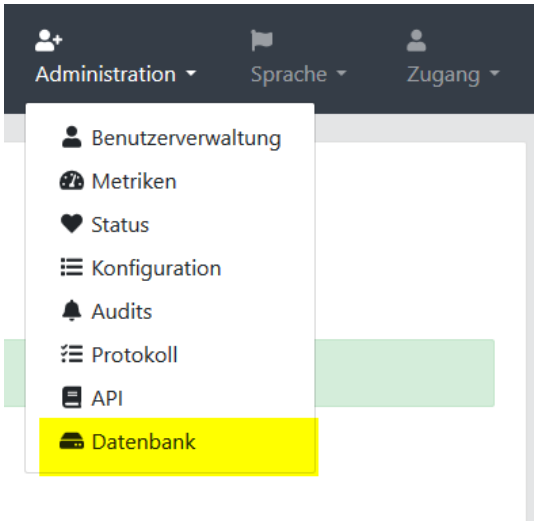


Abbildung 2.15: Datenbank über Admin-Bereich öffnen

In diesem Dialog werden die vorhandenen Tabellen angezeigt. Desweiteren sind SQL-Abfragen möglich. Dies zeigt die Abbildung A.7 im Anhang auf Seite G.

2.5 Architektur

Dieser Abschnitt beschreibt die Projektstrukturen der Multiple-Choice-App sowie der Muliple-Choice-Webanwendung.

2.5.1 Architektur der Multiple-Choice-App

Projektstruktur App

Für die Multiple-Choice-App wurde die Visual Studio Projektmappe 'MCAPP_UI' angelegt. Diese Projektmappe enthält vier Projekte:

- MCAPP_Project.Core - enthält sämtlichen plattformübergreifenden Quellcode. Dazu gehören die Klassen der Modell- sowie der View-Modell-Schicht.
- MCAPP_Project.Droid - ist dafür vorgesehen, um Views für Android-Plattformen zu enthalten.
- MCAPP_Project.iOS - enthält alle Views für iOS-Plattformen.
- MCAPP_Test - enthält Unittests für die Servicemethoden.

Paketstruktur App

Das Projekt MCAPP_Project.Core enthält folgende Pakete:

- Models - enthält Entitäten und die benötigten Fachklassen.
- Repositories - enthält die benötigten Repositories für den Datenzugriff.
- Services - enthält Serviceklassen, welche nach den Fachthemen 'Frage', 'Quiz' und 'Webservice' gruppiert sind. Die Services werden von den View-Modellen aufgerufen.
- ViewModels - enthält alle benötigten View-Modelle. Die View-Modelle werden an die Benutzeroberflächen gebunden.

Das Projekt MCAPP_Project.iOS enthält folgende Pakete:

- Views - enthält die Storyboards für die iOS Benutzeroberflächen

Das Projekt MCAPP_TEST enthält folgende Pakete:

- Tests - enthält die vorhandenen Unittest-Sammlungen. Diese sind nach Services gruppiert.

Unit-Tests

Mithilfe von Unit-Tests werden isolierte Einheiten des Quellcodes geprüft [1, s.S. 18]. Die Einheiten können ganze Klassen oder Methoden einer Klasse sein.

Bei Unit-Tests handelt es sich um Black-Box-Tests, welche Methoden mit bestimmten Parametern aufrufen und die Korrektheit dieser Methoden dadurch prüfen, ob ein erwartetes Ergebnis zurückgeliefert wird. Die genaue Implementierung innerhalb dieser Methode wird also nicht geprüft, sondern das Ergebnis.

Testgetriebene Entwicklung

Die Verwendung von Unit-Tests hat mehrere Vorteile. So sieht der Ansatz der sog. Test getriebenen Entwicklung [32] vor, dass vor Implementierung einer Methode die Unit-Tests geschrieben werden, damit alle notwendigen Anforderungen an eine Methode vor Umsetzung festgestellt werden. Nun wird, nach Ausführung der fehlgeschlagenen Tests, die Methode so implementiert, dass die Tests nach und nach erfolgreich durchlaufen.

Der Vorteil hierbei ist, dass die Entwicklerin nicht nur den Idealfall betrachtet, sondern auch die Sonderfälle. Desweiteren ist die Ausführung von Unittests wesentlich schneller möglich, als ein manueller Test in der Anwendung, da das Testszenario nur einmal aufgebaut werden muss und die Tests beliebig wiederholt werden können. Ein weiterer wesentlicher Vorteil ist, dass die

Unit-Tests nach jeder Änderung wiederholt werden können und somit sichergestellt wird, das Änderungen am Quellcode keine unerwünschten Seiteneffekte zur Folge haben.

Das Projekt MCAPP_Test enthält die Test-Klassen, welche die Unit-Tests ausführen. Jede dieser Klassen prüft die Methoden eines Services. Die Unittests wurden mit dem Framework NUnit[63] geschrieben. NUnit

Damit die Unit-Tests unabhängig von einem bestimmten Datenbankzustand funktionieren, wurde ein Dummy-Repositorie verwendet. Das Dummy-Repositorie, enthält einen bestimmten Datenbankzustand, welcher die zu prüfenden Fälle ermöglicht.

Für das Schreiben der Tests wurde die Bibliothek Moq[64] verwendet. Mit Moq werden bestimmte Aufrufe oder Aktionen simuliert. So werden beispielsweise Speicheraktionen simuliert, da für die Tests keine echte Speicheraktion ausgeführt werden darf.
Das nachfolgende Listing 2.1 zeigt ein Beispiel für eine Test-Klasse sowie einen Unittest.

Listing 2.1: Beispiel für Unit Test

```
[TestFixture]
   public class QuizServiceTests
   {
       QuizService quizService;

       [SetUp]
       public void SetUp()
       {
           IFragenRepository repoFragen = new DummyFragenRepository();
           IQuizRepository repoQuiz = new DummyQuizRepository();
           quizService = new QuizService(repoFragen, repoQuiz);
       }

       [Test]
       public async Task increaseQuizID()
       {
           Quiz q1 = await quizService.CreateQuiz();
           Quiz q2 = await quizService.CreateQuiz();
           Assert.Greater(q2.quizID, q1.quizID);
       }
```

Die Annotation [TestFixture] markiert diese Klasse als TextFixture, also als eine Klasse, welche Tests enthält. Die Annotation [SetUp] wird vor Ausführung eines Unittests aufgerufen und bereitet alles notwendige vor, um den Test ausführen zu können. Die Annotation [Test] markiert eine Methode als Test. Diese wird von einem Test-Runner aufgerufen.

Die Testmethode sollte einen Assert-Aufruf enthalten. Mit Assert wird eine Aussage auf ihre Richtigkeit geprüft. Im Listing 2.1 wird in der Testmethode 'increaseQuizID' geprüft, ob nach Erstellung eines Quiz-Objekts unterschiedliche IDs vergeben wurden. Die Assert-Methode vergleicht also das zweite Objekt mit dem ersten Objekt und prüft, ob die ID des zweiten Objekts größer ist. Sofern der Assert-Aufruf erfolgreich war, wird dieser Test als erfolgreich gemeldet. Assert

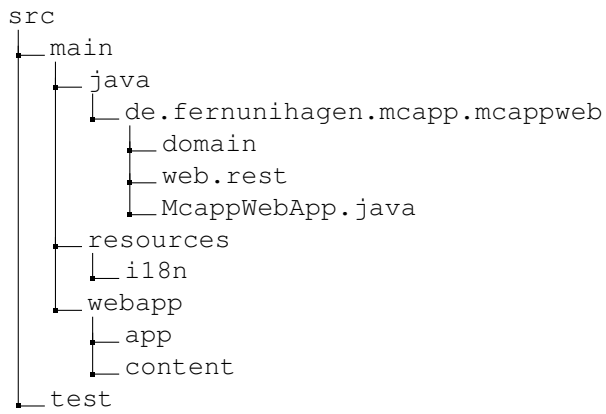
2.5.2 Architektur der Webanwendung

Die Webanwendung besteht aus dem Projektordner MCAPP_WEB. Dieses Projekt enthalt sowohl den Webclient, als auch die Fachobjekte und die Webservices. Projektstruktur Webanwendung

Der Ordner MCAPP_WEB enthält mehrere Ordner und Dateien. Um die wichtigsten zu nennen:

- pom.xml - die Konfigurationsdatei für den Maven-Build. Siehe Abschnitt 2.3.6 auf Seite 20.
- jhipster-jdl.jh - die Konfigurationsdatei für die Generierung der Entitäten. Diese Datei wurde beim Aufsetzen des Projektes importiert und muss nicht erneut importiert werden.
- src - enthält die eigentlichen Quelldateien.
- target - enthält die kompilierten Klassen sowie das Deploymentartefakt.

Der nachfolgende Dateibaum zeigt die wichtigsten Pfade innerhalb des Quellenordners: Paketstruktur



Der Ordner 'domain' enthält die Fachobjekte der Webanwendung. Dazu gehört bspw. die Klasse 'Frage' oder die Klasse 'Thema'.

Der Ordner 'web.rest' enthält die Controllerklassen der REST-Webservices. Dieser enthält u. A. eine Controllerklasse für das Laden und Verarbeiten der Themen. Diese Klasse heißt 'ThemaResource'.

Die Klasse 'McappWebApp' enthält die Main-Methode zum Start der Webanwendung.

Der Ordner 'resources' enthält u. A. die Texte für die Komponenten der Benutzeroberfläche. Die Texte sind in beiden ausgewählten Sprachen hinterlegt.

Der 'webapp'-Ordner enthält die Benutzeroberflächen. Es werden vorwiegend html-Seiten verwendet. Die html-Seiten enthalten jedoch Angular Direktiven, welche wiederum in Dateien beschrieben werden, welche mit '.ts' enden. Das 'ts' steht für TypeScript.

Der 'test'-Ordner enthält sowohl Unittest für den Javacode, als auch für die TypeScript-Komponenten.

Kapitel 3

Multiple-Choice-App

Dieses Kapitel beschreibt die Umsetzung der Multiple-Choice-App. Die nachfolgenden Abschnitte erläutern Ergebnisse und Lösungswege zu den einzelnen Anforderungen.

3.1 Auswertung spezifischer Fragen (Anforderung 1)

Damit die Anwenderin eine spezifische Frage bearbeiten kann, wurde zunächst die Benutzeroberfläche mithilfe eines Storyboards gestaltet.

3.1.1 Benutzeroberfläche

Ein Storyboard wird verwendet, um die Benutzeroberfläche für einen oder für mehrere Bildschirme zu definieren. Ein Bildschirm beschreibt eine Benutzeroberfläche, welche den kompletten Bildschirm eines Gerätes ausfüllt.

Storyboard

Bildschirm

Zu jedem Storyboard wird ein View-Controller hinzugefügt. Der View-Controller wird für die Verwaltung der Benutzeroberfläche benötigt. Mithilfe einer Toolbox werden weitere Elemente auf dem Storyboard platziert[1, s.S. 359].

View-Controller

Die Abbildung 3.1 zeigt das Storyboard 'QuestionTableView' für die Bearbeitung einer spezifischen Frage. Es handelt sich um ein Storyboard mit einem 'UITableView'- Controller. Der 'UITableView'-Controller wird verwendet, wenn eine Liste von Daten anzuzeigen ist. Hierzu bedient sich dieser Controller einer Datenquelle, welche die anzuzeigenden Daten liefert[1, s. S. 386].

UITableView

Die Abbildung 3.1 zeigt außerdem die Bestandteile einer spezifischen Frage. Bei den Tabellenzeilen handelt es sich um Prototypzeilen, welche durch den View-Controller beliebig oft eingefügt werden könnten.

Die erste Tabellenzeile enthält das Label 'ThemaText', welches das Thema dieser Frage anzeigt.

Die zweite Tabellenzeile enthält die laufende Nummer der Frage, sowie den Text der Fragestellung. Die laufende Nummer soll der Anwenderin zeigen, wieviele Fragen bereits bearbeitet wurden sowie wieviele Fragen noch zu bearbeiten sind.

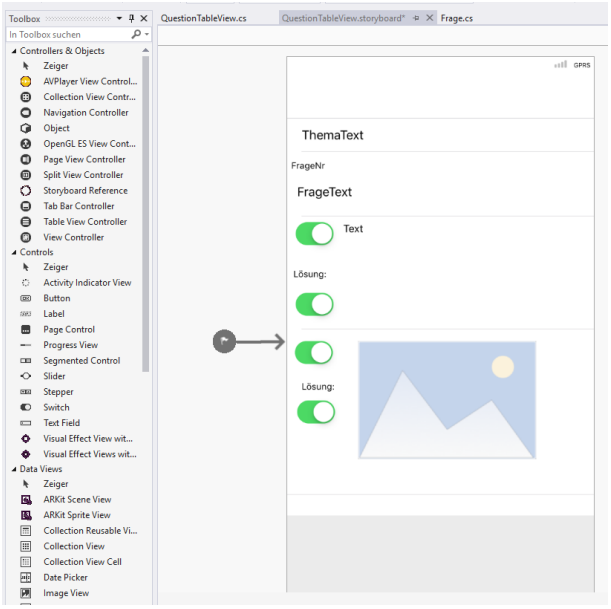


Abbildung 3.1: Storyboard der Frage-Tabelle

Die dritte Tabellenzeile zeigt eine Textantwort. Diese besteht aus einem Label, welche den Antworttext enthält. Desweiteren wird ein Switch-Element verwendet, welches zur An- oder Abwahl einer Antwort betätigt werden kann.

Bei der Auswertung der Fragen wird die richtige Antwort durch ein separates Switch-Element angezeigt. Dies wird im Kapitel 3.6 etwas näher erläutert.

Der obere Bereich des Bildschirms besteht aus einer Navigations-Bar. Diese Navigations-Bar zeigt den Namen der App, sowie Schaltflächen, um zwischen mehreren Bildschirmen zu navigieren. Navigation bar

Die Abbildung 3.2 zeigt den Bildschirm, auf dem die Anwenderin die spezifische Frage bearbeitet. Die Antwortmöglichkeiten werden untereinander angezeigt. Ein Scrollbalken an der Seite zeigt an, dass sich noch weitere Antwortmöglichkeiten im nicht-sichtbaren Bereich befinden. Es werden bis zu acht Antwortmöglichkeiten angezeigt.

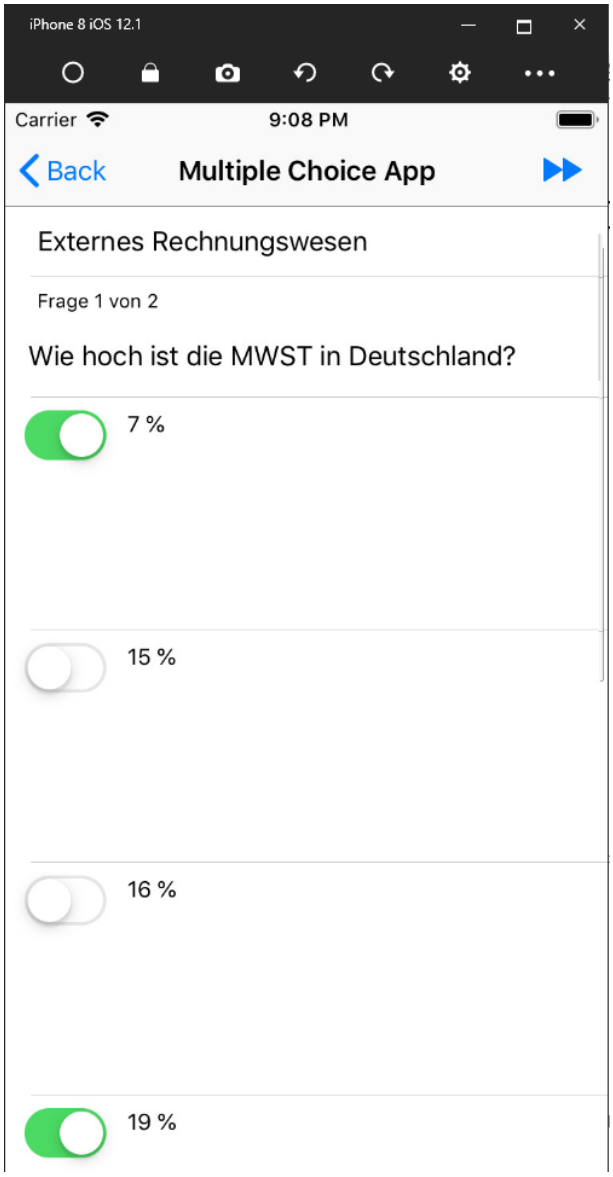


Abbildung 3.2: Spezifische Frage bearbeiten

Auto-Layout und Constraints

Eine besondere Herausforderung bei der Gestaltung der Benutzeroberflächen ist das Auto-Layout mithilfe von Constraints. Da der Bildschirm auf verschiedensten iOS-Geräten mit unterschiedlichen Größen und Auflösungen ähnlich aussehen soll, werden für die Positionierung der Oberflächenelemente keine absoluten Positionskoordinaten verwendet. Stattdessen wird der Abstand zum Bildschirmrand oder zu einem benachbarten Element definiert. Die Verwendung von Constraints ermöglicht ein ähnliches Erscheinungsbild der Benutzeroberfläche bei unterschiedlichen Bildschirmformaten.

Die Abbildung 3.3 zeigt die Constraints eines Switchelements.

Ein weiteres Problem ergibt sich bei der Verwendung der Labels, welche unterschiedliche Textlängen enthalten könnten. Sofern keine Contraints definiert sind, so könnte ggfls. nur ein Teil des Textes angezeigt werden, da sich entweder das Label nicht an die Länge des Textes anpassen kann oder die Zeilenhöhe nicht das komplette Label anzeigt. Aus diesem Grund wurden für die verwendeten Labels entsprechenden Constraints gesetzt, damit Texte komplett angezeigt werden.

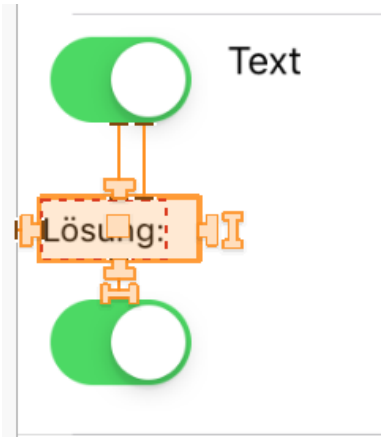


Abbildung 3.3: Constraints für Switch-Element

3.1.2 Fachklassen und Services

Für die Bearbeitung der Fragen wurden im Paket 'Models' des Projektes MCAPP_Project.Core drei Fachklassen benötigt.

Die Klasse 'Thema' enthält den anzuzeigenden Text.

Die Klasse 'Frage' enthält die Fragestellung in Textform sowie Verweise sowohl zum Thema, als auch zu den Antworten.

Die Klasse 'Antwort' enthält die Eigenschaft 'wahr', um festzustellen, ob es sich um eine wahre Aussage handelt. Desweiteren beschreibt die Eigenschaft 'Auswahl', ob die Anwenderin diese Antwort als 'richtig' markiert hat. Von der Klasse 'Antwort' werden die beiden Unterklassen 'Textantwort' sowie 'Bildantwort' abgeleitet. Die Klasse 'Bildantwort' enthält das anzuzeigende Bild als Byte-Datenfeld. Dies wird im Kapitel 3.7 näher erläutert.

Das Klassendiagramm in Abbildung 3.4 verdeutlicht den Aufbau und den Zusammenhang dieser Klassen.

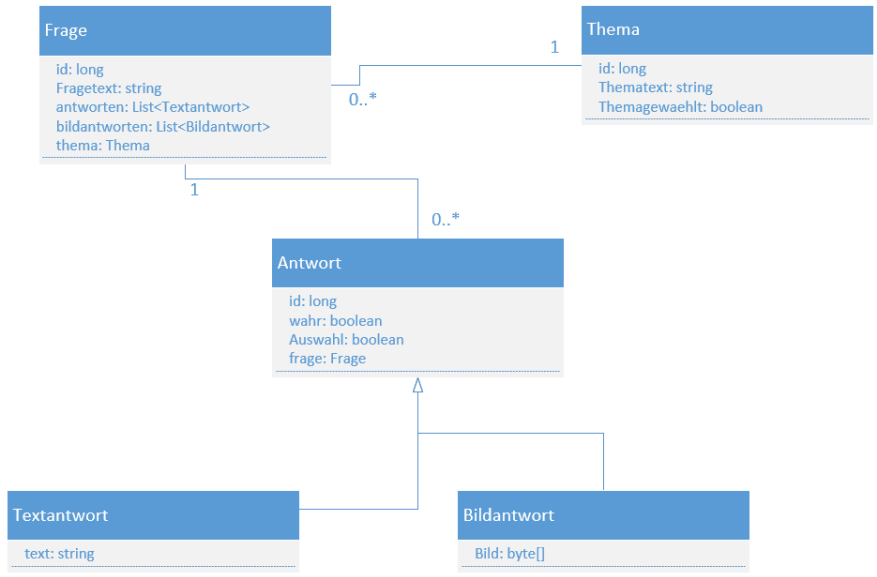


Abbildung 3.4: Klassendiagramm für Fragen

Die Fachobjekte werden nicht innerhalb der View-Modell-Schicht erstellt, sondern werden über einen Service aus 'Project.Core' geladen. Zu diesem Zweck enthält der Ordner 'Services' die Klasse 'FrageService' bzw. die entsprechende Schnittstelle 'IFrageService'.

Für den einfachsten Fall wurde die Servicemethode 'GetSampleFrage()' erstellt, welche eine anzuzeigende Beispielfrage erstellt. Das Listing 3.1 zeigt die Implementierung.

Listing 3.1: Beispielfrage mit Builder erstellen

```
public Frage GetSampleFrage()
{
    FragenBuilder builder = new FragenBuilder();
    return builder.createFrage(1, "Wie hoch ist die MWSt in Deutschland ?", 1)
        .WithAntwort("7 %", true)
        .WithAntwort("15 %", false)
        .WithAntwort("16 %", false)
        .WithAntwort("19 %", true)
        .WithAntwort("23 %", false)
        .WithAntwort("24 %", false)
        .WithAntwort("50 %", false)
        .WithAntwort("keine der Antworten ist richtig", false)
        .Build();
}
```

Für die Erstellung des Frage-Objektes wurde das Builder-Pattern[55] verwendet. Dieses Entwurfsmuster erleichtert die Erstellung der benötigten Objekte. Die Parameter zur Erzeugung des Objektes werden nicht über den Konstruktur, sondern über zusätzliche Methoden übergeben. Builder-Pattern

Die Erzeugung weiterer Unterobjekte, bspw. für Textantworten, wird also über den FragenBuilder ausgeführt.

Der verwendete FragenBuilder wird desweiteren auch für die Unit-Tests verwendet, da für die Ausführung der Tests ein Satz an Fragen benötigt wird. Dies wird im Abschnitt 3.4 etwas weiter ausgeführt.

3.1.3 Bindung Benutzeroberfläche an View-Modell und Modell

Im Abschnitt 2.4 wurde das MVVM-Muster näher erläutert. Desweiteren wurde als Framework MVVMCross gewählt, damit die Komponenten der Benutzeroberfläche an das View-Modell gebunden werden können. Das zuvor erstellte Storyboard 'QuestionTableView' verwendet den gleichnamigen View-Controller. Dieser View-Controller enthält die Methode 'ViewDidLoad'. Die Methode 'ViewDidLoad' wird stets ausgeführt, sobald der Bildschirm geladen wurde. Diese Methode eignet sich daher, um die Komponenten der Benutzeroberfläche an das View-Modell zu binden. Das Listing 3.2 zeigt den verwendeten Code.

Listing 3.2: Bindung View an View-Modell

```
var button = new UIBarButtonItem(UIBarButtonSystemItem.FastForward);
NavigationItem.SetRightBarButtonItem(button, false);

var source = new QuestionTableViewSource(TableView);
TableView.Source = source;

var set = this.CreateBindingSet<QuestionTableView, QuestionTableViewModel
    >();
set.Bind(source).To(vm => vm.Tables);
set.Bind(button).To(vm => vm.NextButtonCommand);
set.Apply();
```

Da der verwendete Viewcontroller eine Tabelle verwaltet, wird für das Anzeigen der Zeilen eine Datenquelle benötigt. Diese Quelle wird mithilfe der Klasse 'QuestionTableViewModel' über die Eigenschaft 'Tables' geliefert.

Für die Tabellenzeilen-Prototypen sind weitere Controllerklassen vorhanden. Dazu gehört die Klasse 'QuestionTableAntwortTextCell', welche der Anzeige der Text-Antworten dient. Diese Controllerklasse bindet, auf gleiche Weise, das Label sowie die Switch-Elemente an das View-Modell.

Das View-Modell 'QuestionTableViewModel' enthält eine Methode namens 'Prepare'. Diese Methode bereitet die Daten für die Tabelle vor. Hierzu werden die Daten aus der Modell-Schicht geladen und für die Bindung an die Tabellenzeilen bereitgestellt.

3.2 Auswertung generischer Fragen (Anforderung 2)

Generische Fragen unterscheiden sich nur unwesentlich von spezifischen Fragen. Charakteristisch für spezifische Fragen ist eine bestimmte Fragestellung, auf welche sich die möglichen Antworten beziehen.

generische Fragen

Bei generische Fragen geht es hingegen darum, den Wahrheitsgehalt verschiedener Aussagen zu prüfen. Die Aussagen beziehen sich aber nicht auf das spezifische Thema aus der Fragestellung. Für gewöhnlich gehören aber alle Aussagen zu einem bestimmten Themengebiet.

Technisch betrachtet unterscheiden sich beide Frageformen jedoch nicht. In beiden Frageformen zeigt der Bildschirm eine Fragestellung an. Die Anwenderin prüft nun den Wahrheitsgehalt der angezeigten Antworten und wählt nun die Aussagen aus, welche die Anwenderin für richtig hält.

Die Abbildung 3.5 zeigt den Bildschirm einer generischen Frage. Die Fragestellung lautet: 'Welche der folgenden Aussagen treffen zu'. Die Zeilen darunter zeigen die zu prüfenden Aussagen.

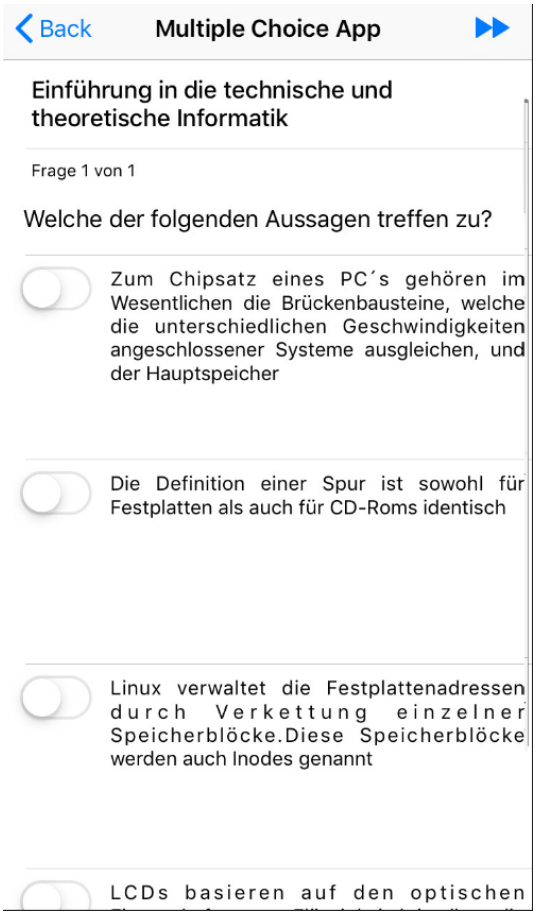


Abbildung 3.5: Generische Frage bearbeiten

3.3 Themengebiet wählen (Anforderung 3)

Damit die Anwenderin die Themengebiete wählen kann, wurde ein weiterer Bildschirm mithilfe eines Storyboards umgesetzt.

3.3.1 Benutzeroberfläche

Das Storyboard 'ThemenwahlView' verwendet den gleichnamigen 'UITableView'-Controller und enthält drei Prototyp-Zeilen. Die erste Zeile enthält ein Label, welches die Anwenderin dazu auffordert, ein oder mehrere Themengebiete zu wählen. Die zweite Zeile beschreibt eines der auswählbaren Themen. Die Auswahl eines Themas erfolgt über ein Switch-Element. Die dritte Zeile enthält eine Schaltfläche, welche den Multiple-Choice-Test startet.

Die Abbildung 3.6 zeigt den Aufbau des Storyboards.

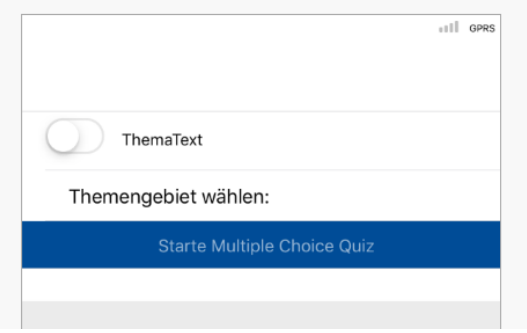


Abbildung 3.6: Storyboard für Themenwahl

Die Oberflächenelemente werden an die entsprechenden Eigenschaften aus dem View-Modell gebunden. Wenn die Anwenderin ein Thema auswählt, so wird dies an die Eigenschaft 'Thema-Gewaehlt' gekoppelt.

Das Listing 3.3 zeigt die Bindung der Themenzeile an das View-Modell.

Listing 3.3: Themenzeile an View-Modell binden

```
var set = this.CreateBindingSet<ThemenwahlCell, ThemaViewModel>();
set.Bind(ThemaText).To(vm => vm.ThemaText);
set.Bind(ThemaGewaehlt).To(vm => vm.ThemaGewaehlt);
set.Apply();
```

Die Schaltfläche zum Start des Multiple-Choice-Tests wird ebenfalls an das View-Modell gebunden. Der Button wird jedoch nicht an eine Eigenschaft des View-Modells gebunden, sondern an ein Kommando.

Kommando (MvxCommand)

Ein Kommando ist ein Objekt, welches die Fähigkeit zur Ausführung einer Aktion kapselt[1, s.S.62]. Die Fähigkeit beschreibt die Bedingungen, die es ermöglichen die Aktion auszuführen. Die Aktion ist, in diesem Fall, der Start eines Multiple-Choice-Tests. Die Bedingung lautet, dass die Anwenderin mindestens ein Thema ausgewählt hat. Dabei muss das ausgewählte Thema mindestens eine Frage enthalten. Solange die Anwenderin also kein entsprechendes Thema ausgewählt hat, kann die Aktion nicht ausgeführt werden und der Button ist somit inaktiv.

Die Listing 3.4 zeigt den relevanten Quellcode zur Erstellung des benötigten Kommandos.

Das Kommando-Objekt ist vom Typ 'MvxAsyncCommand' und wird mit zwei Parametern instanziert. Der erste Parameter heißt 'StartQuiz' und entspricht der eigentliche Aktion zum Starten eines Multiple-Choice-Tests. Der zweite Parameter heißt 'ThemaIstGewaehlt' und dient

der Prüfung, ob die Aktion ausgeführt werden kann. Die Methode 'ThemaIstGewaehlt' prüft, ob mindestens eines der Themen ausgewählt wurde.

Das Kommando-Objekt gehört zum View-Modell und wird mithilfe des BindingSets an den Button gebunden. Somit ist die Oberfläche in der Lage, einen Multiple-Choice-Test zu starten, ohne Einzelheiten aus dem View-Modell zu kennen.

Listing 3.4: Kommando zum Starten des Multiple-Choice-Test

```
 public ThemenwahlViewModel(IMvxNavigationService navigationService)
{
        ....
        this.StartQuizCommand = new MvxAsyncCommand(StartQuiz, ThemaIstGewaehlt);
        ...
}

public Boolean ThemaIstGewaehlt()
{
            Boolean istGewaehlt = false;

            foreach (Thema t in themenListe)
            {
                if (t.ThemaGewaehlt)
                {
                    return true;
                }
            }
            return false;
}

async Task StartQuiz()
{
   ...
}
```

Das Ergebnis zeigt die Abbildung 3.7. Zusätzlich zum Themen-Text wird die Anzahl der Fragen angezeigt, welche dem Thema zugeordnet sind.

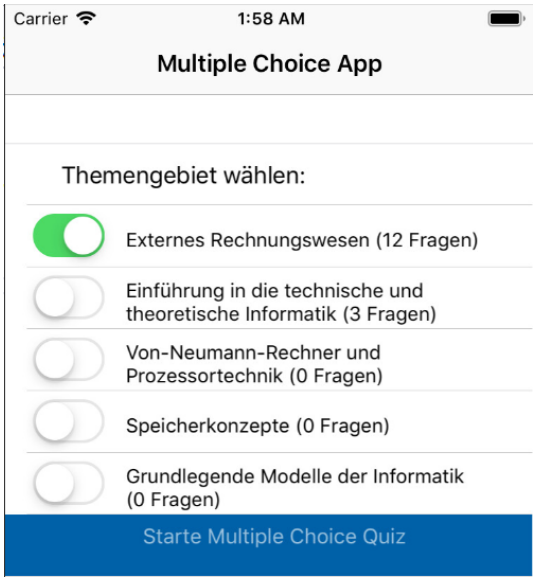


Abbildung 3.7: Bildschirm für Themenwahl

3.3.2 Fachklassen und Services

Für die Themenwahl werden die gleichen Klassen verwendet, welche in Abbildung 3.4 auf Seite 33 vorgestellt wurden. Zur Anzeige der benötigten Daten werden zwei Services benötigt. Der erste Service ermittelt die vorhandenen Themengebiete. Der zweite Service liefert zu den einzelnen Themen die zugeordneten Fragen.

Aus der Anforderung 14 geht hervor, dass ein Offlinebetrieb vorgesehen ist. Diese Anforderung wird im Abschnitt 3.9 auf Seite 55 näher erläutert. Mit der Umsetzung dieser Anforderung wurden die Daten nicht mehr zur Laufzeit der Anwendung erstellt, sondern aus der SQLite Datenbank des Endgerätes geladen.

Fachklasse auf Datenbank abbilden

Im Kaptel 2.4 auf Seite 22 wurde bereits erkläutert, dass für den Datenbankzugriff das ORM 'SQLite-Net-Pcl' verwendet wird. Das ORM ermöglicht den vereinfachten Zugriff auf die Datenbank unter Verwendung der Fachklassen.

Die Fachklasse 'Thema' wurde um zwei Annotationen erweitert, welche die Verwendung dieser Klasse für den Datenbankzugriff ermöglicht. Das nachfolgende Listing 3.5 zeigt den Quellcode der Fachklasse 'Thema'.

Listing 3.5: Fachklasse Thema mit Datenbankannotation

```
public class Thema
{

    [PrimaryKey, AutoIncrement]
    public long id
    {
        get; set;
    }
```

```
        public String ThemaText
        {
            get; set;
        }

        [Ignore]
        public Boolean ThemaGewaehlt
        {
            get; set;
        }
    }
```

Die ID wurde mit den Annotationen 'PrimaryKey' und 'AutoIncrement' beschriftet. Die Angabe des PrimaryKey ist wesentlich für die Datenbankverarbeitung, da durch diese Identifizierung die Aktualisierung eines bereits vorhandenen Datensatzes ermöglicht wird. PrimaryKey

Die Annotation 'AutoIncrement' sorgt dafür, dass bei neu erstellten Instanzen die nächsthöhere ID vergeben wird.

Die Eigenschaft 'ThemaText' enthält keine Annotation. Sobald ein Objekt vom Typ 'Thema' gespeichert wird, so wird dennoch in der Datenbank die Spalte mit dem Namen 'ThemaText' gefüllt.

Die Klasse enthält die boolesche Eigenschaft 'ThemaGewaehlt'. Diese Eigenschaft gehört nicht zum 'Thema'-Datensatz und soll folgerichtig auch nicht gespeichert werden. Aus diesem Grund wurde diese Eigenschaft mit der Annotation 'Ignore' beschriftet. Ignore

Für die Datenbankzugriffe wurden Repository-Klassen geschrieben und nach Fachdomänen getrennt. Die Zugriffe auf die Themen-Tabelle gehören zur Domäne 'Fragen'. Dementsprechend heißt das benötigte Repositorie 'FragenRepository'. In einem Repositorie wird eine Datenbankverbindung aufgebaut. Das Repositorie enthält Methoden zum Laden und Speichern der Fragen-Datensätze. Speichern und Laden Repositorie

Das nachfolgende Listing 3.6 zeigt die wichtigsten Datenbankaktionen.

Listing 3.6: Datenbankverbindung herstellen

```
public FragenRepository()
{
        var local = FileSystem.Current.LocalStorage.Path;
        var datafile = PortablePath.Combine(local, MCAPP_PROPERTIES.
            DATENBANK_IN_SMARTPHONE_NAME);
        connection = new SQLiteConnection(datafile);
        connection.CreateTable<Thema>();
}

public int SaveThema(Thema thema)
{
        return connection.InsertOrReplace(thema);
}

public List<Thema> GetAllThemen()
{
            return connection.Table<Thema>().ToList<Thema>();
}
```

Der Name der Datenbank wird über die Konfigurationsdatei 'MCAPP_PROPERTIES' de-

finiert. Mit dem 'connection'-Objekt wird die Datenbank, sowie die benötigte Tabelle, erstellt. Wenn die Datenbank oder die Tabelle bereits vorhanden ist, so werden die vorhandenen Daten verwendet. Das Speichern eines Thema-Objektes erfolgt mit der Methode 'InsertOrReplace'. Das Laden der kompletten Datenbanktabelle erfolgt mit der Methode 'Table'.

Es ist möglich, die SQLite Datenbank im Entwicklungssystem zu prüfen. Hierfür notwendig ist die Installation des Programms sqlite3[18]. sqlite3

Mit diesem Kommandozeilenprogramm können Abfragen an die Datenbank des Gerätes gestellt werden. Desweiteren ist es möglich, die angelegten Tabellenstruktur zu prüfen.

Die Abbildung 3.8 zeigt eine mögliche Verwendung dieses Programms.

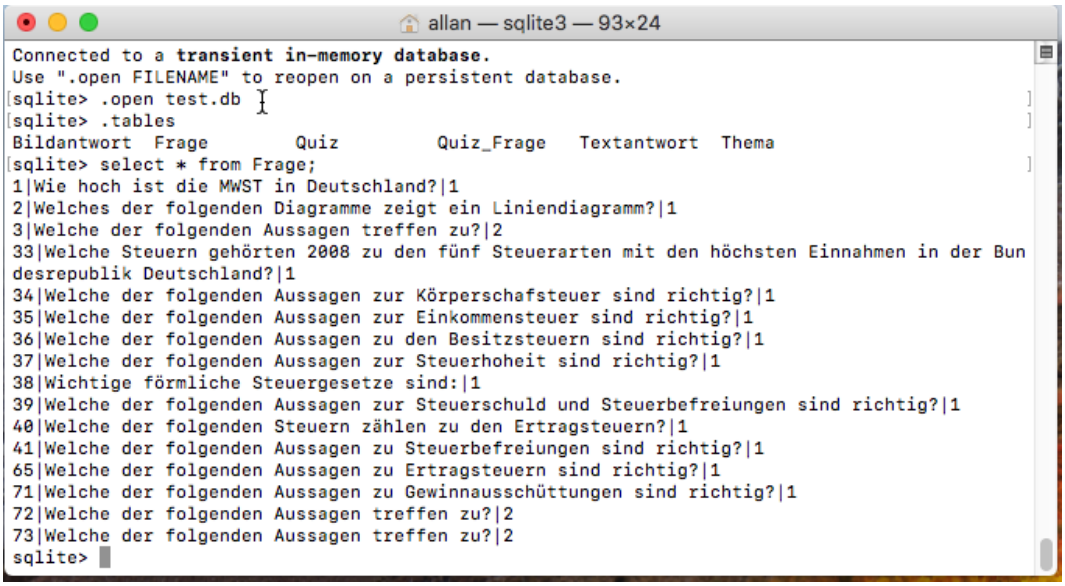


Abbildung 3.8: Datenbankabfrage in iOS

Die Verbindung zur Datenbankdatei wird mit dem Befehl '.open' hergestellt. Der Befehl '.tables' listet alle vorhandenen Datenbanktabelle auf. Mit dem Befehl 'select * from Frage;' werden alle vorhandenen Frage-Datensätze angezeigt. Der Befehl 'pragma table_info('frage'); zeigt die verwendeten Datenbankfelder an.

Das View-Modell greift nicht direkt auf das Repositorie zu, sondern nutzt Services, um die Liste der vorhandenen Themen zu laden. Dies hat u. A. den Vorteil, dass das Repositorie ausgetauscht werden kann, ohne das eine Änderung im View-modell notwendig wird. Servicemethoden

Das nachfolgende Listing 3.7 zeigt eine Methode zum Laden aller Themen. Der Service greift auf die Methoden des Repositories zu und liefert eine Liste aller Themen. Diese Themen werden im weiteren Verlauf auf dem Bildschirm angezeigt.

Listing 3.7: Servicemethode zum Laden aller Themen

```
public List<Thema> GetAllThemen()
```

```
{
    return repository.GetAllThemen();
}
```

Die Themenliste zeigt zu jedem Thema die Anzahl der zugeordneten Fragen an. Um dies zu ermöglichen, wurde die Klasse 'FragenRepository' erweitert. Das Repositorie verwendet die 'Frage'-Tabelle und liefert eine Liste von Fragen-Datensätzen, zu einem bestimmten Thema.

Fragen zum Thema ermitteln

Das nachfolgende Listing 3.8 zeigt die Servicemethode 'GetFragen'. Als Parameter wird die ID eines Themas übergeben.

Listing 3.8: Servicemethode zum Laden aller Fragen eines Themas

```
public List<Frage> GetFragen(long themaID)
{
    return repository.GetFragen(themaID);
}
```

Im View-Modell wird zu jedem Thema die Frageliste geladen. Beim Anzeigen des Thema-Textes wird aus dieser Liste die Anzahl der Fragen gezählt und angezeigt.

3.3.3 Navigation

Die Multiple-Choice-App besteht aus mehreren Bildschirmen. Diese Anforderung zeigt die Liste der Themen auf einem Bildschirm. Durch Betätigung einer Schaltfläche wird ein Multiple-Choice-Test gestartet. Dieser Multipe-Choice-Test wird auf einem anderen Bildschirm angezeigt. Der Wechsel zwischen zwei Bildschirmen wird als Navigation bezeichnet. Charakteristisch für eine Navigation innerhalb einer App ist u. A. die Möglichkeit zum vorherigen oder zum nächsten Bildschirm zu wechseln.

Im Zusammenhang mit dem MVVM-Entwurfsmuster beschreibt Bennett zwei Ansätze, um zwischen Bildschirmen zu navigieren[1, s.S. 76].

Der erste Ansatz wird View-first genannt. View-first bedeutet, dass die View die Navigation steuert. Sobald die Benutzeroberfläche aufgebaut wurde, erzeugt diese View das View-Modell der nächsten Benutzeroberfläche. Die View-Schicht steuert also die Navigation zu einer anderen View.

View-first

Der zweite Ansatz wird 'View-model-first' genannt. View-model-first bedeutet, dass das View-Modell die Navigation zum nächsten Bildschirm ausführt. Üblicherweise wird eine Navigation durch Betätigung einer Schaltfläche ausgelöst. Die Schaltfäche wird an ein Kommando aus dem View-Modell gebunden. Das aufgerufene Kommando löst die Navigation zum nächsten View-Modell aus. Das View-Modell steuert somit die Navigation zu einer anderen View.

View-model-first

Der 'view-model-first' Ansatz ist weit verbreitet, da die Methoden der Navigation thematisch zum View-Modell gehören. Dieser Ansatz erlaubt es, dass weniger plattformspezifischer Code in der View-Schicht vorhanden ist[1, s.S. 78]. Aus diesem Grund wurde für die Navigation der 'view-model-first'-Ansatz gewählt.

Für die Navigation, zwischen den Bildschirmen, stellt das Framework MvvmCross den Service 'IMvxNavigationService' zur Verfügung. Dieser Service wird in der Klasse 'ThemenwahlView-Model' über 'Dependency Injection' geladen und verwendet.

IMvxNavigation Service

Das folgende Listing zeigt die Navigation zur ersten Seite des Multiple-Choice-Tests. Das Objekt 'navigationService' ist vom Typ 'IMvxNavigationService' und löst über die Methode 'Navigation' den Wechsel zum View-Modell 'QuestionTableViewModel' aus.

Listing 3.9: Navigation zur Fragentabelle

```
 async Task StartQuiz()
{
...
 await navigationService.Navigate(typeof(QuestionTableViewModel), quiz);
...
}
```

Bei der Navigation zum nächsten Bildschirm wird die asyncrone Methode 'Navigate' verwendet. Asyncrone Methoden sollen verhindern, dass die Anwendung blockiert wird, während bspw. Daten geladen werden. Diese asyncrone Methode könnte also weitere Anweisungen enthalten, die ausgeführt werden, während die Navigation ausgeführt wird. async

Das Schlüsselwort 'await' zeigt an, dass der Kontrollfluss für die Navigation gestoppt wird. Sofern weitere Anweisungen folgen würden, welche das Ergebnis dieses asyncronen Aufrufs benötigen, so würden diese bis zum Ende der Verarbeitung warten. await

Da die Methode 'StartQuiz' eine asyncrone Methode aufruft, muss die Methode selbst ebenfalls als asyncron deklariert werden. Desweiteren müssen asyncrone Methoden den Datentyp 'Task' zurückliefern. Aus dem Datentyp 'Task' wird am Ende der asyncronen Verarbeitung das Ergebnis ausgelesen. Task

3.3.4 Unittests

Die Unittests zu dieser Anforderung befinden sich in der Klasse 'FragenServiceTests'. Das Listing 3.10 zeigt exemplarisch den Unittest 'fragenFuerEinThema'. Dieser Test prüft die Liste der Fragen zu einem Thema mit der ID 1. Der Unittest prüft, ob die Methode 'GetFragen(themaID)' aus dem Fragenservice tatsächlich nur die Fragen liefert, welche dem Thema '1' zugeordnet wurden.

Listing 3.10: Unittest - Fragen für ein Thema

```
[Test]
        public void fragenFuerEinThema()
        {
            List<Frage> fragen = service.GetFragen(1);
            Boolean ok = true;
            foreach(Frage f in fragen)
            {
                if (f.thema_id!=1)
                {
                    ok = false;
                }
            }

            Assert.IsTrue(ok);
        }
```

3.4 Fragen anfordern (Anforderung 4)

Aus den zuvor gewählten Themen werden die zugeordneten Fragen gezogen. Hierbei ist zu berücksichtigen, dass Fragen, die zuvor richtig beantwortet wurden, mit kleinerer Wahrscheinlichkeit gezogen werden. Desweiteren wurde die Navigation erweitert, damit ein Wechsel zwischen den verschiedenen Fragen möglich ist.

3.4.1 Fachklassen und Services

Fachklassen 'Quiz' und 'Quiz_Frage'

Für die Ausführung eines Multiple-Choice-Tests könnte ein einfaches List-Objekt verwendet werden. Aus den Anforderungen geht jedoch hervor, dass über die Bearbeitung der Fragen eine Auswertung zu erstellen ist. Desweiteren muss die Datenbank Informationen enthalten, wenn Fragen zuvor richtig beantwortet wurden.

Um diese Anforderungen zu erfüllen sind weitere Fachklassen notwendig, um die benötigten Informationen in die Datenbank zu speichern. Die Abbildung 3.9 zeigt die Fachklassen 'Quiz' und 'Quiz_Frage' sowie deren Zusammenhang zur eigentlichen Frage-Klasse.

Abbildung 3.9: Klassendiagramm für Quiz

Im Abschnitt 2.4 auf Seite 23 wurde bereits der Begriff 'Quiz' eingeführt. Ein Quiz entspricht der Ausführung eines Multiple-Choice-Tests. Die Benennung 'Quiz' wurde gewählt, um Missverständnissen beim Lesen des Quellcodes vorzubeugen, da Quelldateien, die den Namen 'Test' enthalten, häufig für Unittests verwendet werden.

Die Klasse 'Quiz' enthält den Zeitpunkt der Ausführung des Multiple-Choice-Tests. Desweiteren ist die boolesche Eigenschaft 'auswertung_gesendet' vorhanden. Diese Eigenschaft stellt

sicher, dass die Auswertung nur einmal gesendet werden wird. Die Klasse 'Quiz' enthält eine Liste von Fragen. Diese Liste wird für die Navigation benötigt, wird aber nicht in der gleichnamigen Datenbanktabelle gespeichert.

Das Speichern der gezogenen Fragen wird über die Klasse 'Quiz_Frage' durchgeführt. Die Klasse 'Quiz_Frage' enthält einen Verweis zu einem Frage-Objekt. Diese Klasse enthält außerdem den booleschen Wert 'richtig_beantwortet', welche nach Bearbeitung der Fragen gesetzt wird. Weiterhin enthält diese Klasse ein Datumsfeld, welches dazu dient, die Auswertung der beantworteten Fragen zu vereinfachen. Für die Auswertung der Nutzung wird jedoch das Datumsfeld der 'Quiz'-Klasse berücksichtigt.

Die Klasse 'ThemenwahlViewModel' enthält das Kommando 'StartQuiz', welches durch Betätigung der Schaltfläche ausgelöst wird. Zu diesem Zeitpunkt hat die Anwenderin bereits die gewünschten Themen ausgewählt. Desweiteren wurde die Anzahl der Fragen auf '10' festgesetzt. Damit ein 'Quiz'-Objekt erstellt werden kann, wird eine Liste mit bis zu zehn Fragen gezogen. Hierzu wurde der 'FrageService' wie folgt erweitert:

Listing 3.11: Servicemethode um Fragen für ausgewählte Themen zu ermitteln

```
Task<List<Frage>> GetFragen(List<Thema> gewaelteThemen, int anz);
```

Anzahl Fragen pro Thema

Bevor die eigentlichen Fragen gezogen werden können, muss festgelegt werden, wieviele Fragen zu jedem Thema zu ziehen sind.

Im einfachsten Fall wählt die Anwenderin ein Thema aus, welches mehr als zehn Fragen enthält. In diesem Fall ist keine weitere Berechnung notwendig und es werden zehn Fragen zu diesem Thema gezogen.

Wählt die Anwenderin hingegen drei Themen aus und jedes dieser Themen enthält mindestens vier Fragen, so ist festzulegen, wie die Fragen gezogen werden, damit insgesamt zehn Fragen vorhanden sind. Für die Anwenderin macht es i.d.R. Sinn, wenn zu jedem Thema eine möglichst gleiche Anzahl an Fragen gezogen werden würde. Da aber insgesamt zehn Fragen bearbeitet werden, muss zwangsläufig von einem der ausgewählten Themen eine vierte Frage gezogen werden.

Bei der Berechnung der Anzahl ist außerdem zu berücksichtigen, dass es Themen geben kann, die weniger Fragen enthalten, als im Durchschnitt möglich wären. In diesem Fall werden ebenfalls entsprechend mehr Fragen eines anderen Themas gezogen.

Zuletzt ist zu beachten, dass es sich bei der Anzahl von zehn Fragen um einen variablen Wert handelt und im Zusammehhang mit der Anforderung 12 (siehe Abschnitt 3.8 auf Seite 54) eine andere Anzahl möglich wäre.

Das nachfolgende Listing zeigt einen Auszug der Methode 'CalcAnzahlProThema' aus dem 'FrageService':

Listing 3.12: Anzahl der Fragen pro Thema berechnen

```
 public int[] CalcAnzahlProThema(List<Thema> gewaelteThemen, int anz)
{

            int anzahlAlleGewaehltenFragen = GetFragen(gewaelteThemen).Count;

            // Durchschnitt bilden
```

```
            int durchschnittFragenProThema = anz / gewaelteThemen.Count;

            // In diesem Feld halten wir fest, wieviele Fragen pro Thema
            int[] anzahlProThema = new int[gewaelteThemen.Count];

            // Hier halten wir fest, wenn zu einem Thema bereits alle Fragen gezogen
                werden
            bool[] themaHatZuwenigFragen = new bool[gewaelteThemen.Count];

            for (int i = 0; i < gewaelteThemen.Count; i++)
            {
                int count = GetFragen(gewaelteThemen[i].id).Count;

            ...
        }
}
```

Die Methode erhält, als Eingabeparameter, eine Liste der gewählten Themen, sowie die Anzahl der gewünschten Fragen. Die Methode liefert ein Datenfeld zurück, welches die Anzahl der Fragen pro Thema enthält. Die Methode bildet nun den Durchschnitt, aus der gewünschten Anzahl der Fragen sowie aus der Anzahl der Themen.

Im weiteren Verlauf wird nun zu jedem Thema geprüft, ob ausreichend Fragen vorhanden sind. Sofern die Anzahl der Fragen eines Themas nicht ausreicht, so wird die entsprechende Anzahl aus dem nächsten Thema gewählt. Sollten insgesamt weniger Fragen vorhanden sein, als angegeben, so wird zwangsläufig eine kleinere Anzahl an Fragen zurückgeliefert.

Zufallsfrage ziehen

Der 'FrageService' wurde um die Methode 'GetZufallsFragen' erweitert. Diese Methode erhält einen Parameter für das Thema, sowie einen Parameter für die Anzahl der zu ziehenden Fragen. Das nachfolgende Listing zeigt einen Auszug dieser Methode:

Listing 3.13: Zufallsfragen aus Thema ziehen

```
for (int i = 0; i < anz; i++)
{

                lock (syncLock)
                {
                    if (alleFragen.Count == 0 && nichtBeantworteteFragen.Count == 0)
                    {
                        break;
                    }

                    List<Frage> fragenListe = (nichtBeantworteteFragen.Count > 0) ?
                        nichtBeantworteteFragen : alleFragen;

                    int index = random.Next(0, fragenListe.Count);
                    zufallsFragen.Add(fragenListe[index]);
                    fragenListe.RemoveAt(index);
                }
}

            return zufallsFragen;
```

Damit das Verfahren durchgeführt werden kann, wird zunächst geprüft, ob Fragen vorhanden sind, die zuvor nicht korrekt beantwortet wurden. Dies setzt voraus, dass zuvor bereits ein Multiple-Choice-Test ausgewertet wurde. Dies wird näher im Abschnitt 3.6 auf Seite 49 erläutert.

nicht-beantwortete Frage

Alle Fragen, die noch nicht korrekt beantwortet wurden, werden in der Liste 'nichtBeantwor-

teteFragen' gesammelt. Zu den Fragen, die nicht korrekt beantwortet wurden gehören einerseits Fragen, die noch nie bearbeitet wurden. Andererseits werden auch Fragen berücksichtigt, die in der jüngsten Auswertung von der Anwenderin falsch beantwortet wurden. Wenn eine Frage bereits richtig beantwortet wurde, zu einem späteren Zeitpunkt aber erneut gezogen, aber falsch beantwortet wurde, so gilt diese Frage als 'nichtBeantwortet'.

Das Listing zeigt, dass in einem iterativen Prozess Fragen gezogen werden. Solange noch Fragen vorhanden sind, welche nicht beantwortet wurden, so wird mithilfe des Befehls 'random' aus dieser Liste eine zufällige Frage gezogen. Andernfalls wird aus der Liste der bereits beantworteten Fragen eine Frage gezogen. Jede bereits gezogene Frage wird aus der entsprechenden Liste entfernt. random

Die Methode muss syncronisiert ausgeführt werden, da andernfalls aufgrund der Nebenläufigkeit[57] der Prozessoren, die 'random'-Methode gleiche Zahlen zurückliefern würde. Der Befehl 'lock(syncLock)' stellt somit sicher, dass kein anderer Thread erzeugt wird, welcher zum gleichen Zeitpunkt eine Frage zieht. lock

Das erläuterte Verfahren stellt sicher, dass zu den ausgewählten Themen Zufallsfragen gezogen werden. Desweiteren erscheinen bereits richtig beantwortete Fragen mit kleinerer Wahrscheinlichkeit.

3.4.2 Navigation

Im Listing 3.4 auf Seite 37 wurde bereits die Navigation zur FragenTabelle erläutert. Diesem Befehl wird nun das 'quiz'-Objekt mit den zuvor gezogenen Fragen übergeben.

Der Bildschirm zur Bearbeitung einer Frage enthält in der Navigationsbar eine Schaltfläche für die Navigation zur nächsten Frage. Diese Schaltfläche wurde mithilfe von 'MVVMCross' an das Kommando 'NextButtonCommand' gebunden. Das nachfolgende Listing zeigt die Navigation zum nachfolgenden Bildschirm:

Listing 3.14: Navigation zur nächsten Frage

```
async Task NextButton()
        {
            int newpos = this.quiz.position+1;

            if (quiz.fragen.Count> (newpos))
            {
                this.quiz.position = this.quiz.position + 1;
                await navigationService.Navigate(typeof(QuestionTableViewModel), this.
                    quiz);
            } else
            {
                await navigationService.Navigate(typeof(AuswertungTabelleViewModel),
                    this.quiz);
            }
        }
```

Die Klasse 'Quiz' enthält die Eigenschaft 'position'. Diese Eigenschaft wird einerseits verwendet, damit das View-Modell die gewünsche Frage aus der Fragenliste angezeigt. Andererseits ist die Angabe auch notwendig, damit angezeigt werden kann, wieviele Fragen noch zu bearbeiten sind.

Sofern bereits alle Fragen bearbeitet wurden, so erfolgt die Navigation zur Auswertungstabelle.

3.4.3 Unittests

Die Klasse 'FrageServiceTest' enthält mehrere Unittests zu den erläuterten Funktionen.

Mit der Testmethode 'fragenFuerMehrereThemen' werden Fragen zu zwei Themen angefordert. Diese Prüfung stellt sicher, dass zu beiden Themen Fragen zurückgeliefert werden.

Mit der Testmethode 'FragenMitFesterAnzahl' wird geprüft, ob zu einem Thema zehn Fragen geliefert werden.

Die Methoden 'FragenNachThemenGleichverteilt' sowie 'VerteilungVonFragenAufThemen' stellen sicher, dass nach Auswahl mehrerer Themen eine weitestgehend gleiche Verteilung der Fragen auf die Themen vorhanden ist.

Die Methode 'zufallsFragenNotSame' stellt sicher, dass beim Ziehen der Fragen nicht mehrmals die gleichen Fragen gezogen werden.

Die Methoden 'zufallsFrageMitNichtBeantworteteFragen' sowie 'ZufallsFrageEnthaeltAuchBeantworteFragen' pruefen, ob die nicht-beantworteten Fragen mit höherer Wahrscheinlichkeit gezogen werden.

3.5 Datenschutz (Anforderung 5)

Für den Einsatz der Multiple-Choice-App wurde die Datenschutz Grundverordnung geprüft [33] und entsprechende Maßnahmen umgesetzt.

3.5.1 Schutz personenbezogener Daten nach DSGVO

Das Ziel dieser Untersuchung war sicherzustellen, ob der Zugriff und der Versand der Daten dem geltenden Recht entspricht. Die DSVGO 'enthält Vorschriften zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten und zum freien Verkehr solcher Daten' [33, s. (1)]. Desweiteren regelt diese Verordnung 'deren Recht auf Schutz personenbezogener Daten' [33, s. (2)].

In Art. 4 wird der Begriff 'personenbezogene Daten' wie folgt deklariert: 'alle Informationen, die sich auf eine identifizierte oder identifizierbare natürliche Person...beziehen; ....die direkt oder indirekt, insbesondere mittels Zuordnung zu einer Kennung wie einem Namen, zu einer Kennnummer, zu Standortdaten, zu einer Online-Kennung oder zu einem oder mehreren besonderen Merkmalen identifiziert werden kann, die Ausdruck der physischen, physiologischen, genetischen, psychischen, wirtschaftlichen, kulturellen oder sozialen Identität dieser natürlichen Person sind' [33, Art 4 DSGVO].

Die Verarbeitung 'personenbezogener Daten' sind in der DSGVO zum Schutz natürlicher Personen geregelt und die DSGVO ist gemäß Art. 3 anzuwenden, da Daten verarbeitet werden von betroffenen Personen, welche sich in der Union befinden[33, Art 3 DSGVO].

Aus den Anforderungen zur Multiple-Choice-App geht jedoch hervor, dass die Verarbeitung personenbezogener Daten nicht erwünscht ist und nicht umzusetzen ist. Dies bedeutet insbesondere, dass zur Nutzung der Multiple-Choice-App von der Anwenderin keine Daten erhoben werden, welche gemäß Art. 4 DSGVO als personenbezogen deklariert sind.

Desweiteren werden für die Auswertungen zur Nutzung der Multiple-Choice-App, an die Kursbetreuerin nur die Informationen gesendet, welche für die Auswertung relevant sind. Diese enthalten jedoch keine Informationen, die einen Rückschluss auf natürliche Personen ermöglichen könnten.

Für den Einsatz der Multiple-Choice-App, sowie für den Autorenzugang, werden somit keine personenbezogenen Daten verwendet. Aus diesem Grund sind keine weiteren Maßnahmen zum Schutz natürlicher Personen erforderlich.

3.5.2 Schutz des Contents

Der Zugriff auf den Autorenzugang ist nur der Kursbetreuerin gestattet. Zum Schutz dieses Bereiches, vor unauthorisiertem Zugriff, wird Spring Security eingesetzt.

Spring Security

Die Sicherheitseinstellungen der Webanwendung werden in der Klasse 'SecurityConfiguration' konfiguriert. Diese Klasse enthält die Annotation @EnableWebSecurity. Die Annotation @EnableWebSecurity aktiviert den Zugriffsschutz in Spring MVC Anwendungen[39, s.S.249]. Dies bedeutet insbesondere, dass der Zugriff auf den Autorenzugang, sowie der Zugriff auf die REST-Services, nur dann möglich ist, wenn zuvor ein Authentifizierungstoken geladen und an einer entsprechenden Stelle im Request-Header geschrieben wurde.

@EnableWebSecurity

Der Zugriffsschutz kann bspw. mit der Anwendung soapUI[58] geprüft werden. Die Abbildung A.8 im Anhang auf Seite H zeigt den Zugriff auf den REST-Service '/api/text-antworts'. Über diesen REST-Webservice werden die Antworten der Fragen geladen. Dieser Aufruf könnte exemplarisch in soapUI angelegt werden. Wird in dem Request-Header ein ungültigen Zugriffstoken hinterlegt, so liefert der Webservice die Meldung 'Vollständige Authentifikation wird benötigt, um auf diese Ressource zuzugreifen'.

Über die Ressource '/api/authenticate' kann hingegen ein gültiges Zugriffstoken angefordert werden. Dieses Token wird nur dann herausgegeben, wenn der Requestheader die richtige Username/Kennwort-Kombination für den Autorenzugang enthält. Wird nun das gültige Token in den ursprünglichen Request für die Authorisierung eingesetzt, so liefert der REST-Webservice die angefragten Informationen. Dies zeigt die Abbildung A.9 im Anhang auf Seite H.

Damit sollten die Bereiche des Autorenzugangs ausreichend geschützt sein.

3.5.3 Verschlüsselte Übertragung mit SSL

Der Autorenzugang unterstützt die verschlüsselte Übertragung aller Daten über das Secure-Socket-Layer-Protokoll (SSL).

Laut Dr. Stefan Wohlfeil erlaubt die Verwendung von SSL eine 'vertrauliche und authentische Kommunikation' [60, s.S. 135]. Desweiteren empfiehlt Wohlfeil die Verwendung von TLS 1.2, da dieser Standard neue Verschlüsselungs-Algorithmen wie SHA-256 oder AEG einführt. Weiterhin schrieb Wohlfeil: 'Der sinnvolle Einsatz von SSL erfordert ein Zertifikat'[60, s.S.190].

TLS 1.2

Die Konfiguration für die verschlüsselte Übertragung wird in der Dokumentation von JHipster beschrieben[50]. Der Anbieter empfiehlt die Nutzung von SSL im Produktivbetrieb zu aktivieren.

Für die Erstellung eines Zertifikates wird das Programm 'keytool' verwendet, welches mit der Java Laufzeitumgebung ausgeliefert wird. Der Befehl zum Erstellen einer Zertifikatsdatei lautet: keytool

Listing 3.15: Zertifikat erstellen

```
keytool -genkey -alias mcapp -storetype PKCS12 -keyalg RSA -keysize 2048 -keystore
    keystore.p12 -validity 3650
```

Bei der Erstellung des Zertifikates ist die Angabe eines Kennwortes erforderlich. Das Kennwort, sowie der Alias der Zertifikatsdatei, ist in der Datei 'application-prod.yml' einzutragen.

Das Listing A.1 im Anhang auf Seite H zeigt eine mögliche Konfiguration, um SSL zu aktivieren.

Wird die Anwendung, nach Aktivierung von SSL, gestartet, so ist der Zugriff auf den Autorenzugang nur noch über das https-Protokoll möglich.

Die entsprechende Meldung im Serverprotokoll lautet dann: 'i.g.j.s.ssl.UndertowSSLConfiguration : Setting user cipher suite order to true'.

Zusätzlich muss das verwendete Zertifikat im http-Client der Multiple-Choice-App mitgesendet werden, damit die Multiple-Choice-App über das https-Protokoll mit dem Server kommunizieren kann. Dies wurde durch die Eigenschaft 'WITH_SSL' in der Klasse 'MCAPP_PROPERTIES' weitestgehend vorbereitet. SSL im Client aktivieren

3.6 Auswertung (Anforderung 7)

Nach Bearbeitung der Fragen erhält die Anwenderin die Ergebnisse des Multiple-Choice Tests. Einerseits wird, nach Themen gruppiert, die Anzahl der richtig und falsch beantworteten Fragen angezeigt. Andererseits kann die Anwenderin durch die bearbeiteten Fragen navigieren und ihre Antworten mit den Lösungen vergleichen.

3.6.1 Benutzeroberfläche

Für den Auswertungsbildschirm wird ein weiteres Storyboard mit einem 'UITableView'-Controller benötigt. Das Storyboard enthält zwei Prototypzeilen. Die erste Zeile zeigt das Thema der Auswertung sowie einen Auswertungstext. Die zweite Zeile enthält eine Schaltfläche, um einen neuen Multiple-Choice-Test zu starten. Die Abbildung 3.10 zeigt das Storyboard der Auswertungsseite.

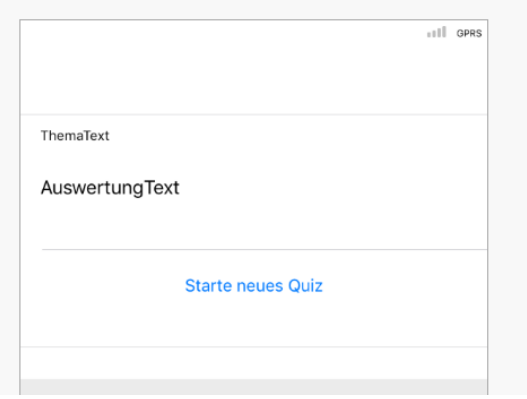


Abbildung 3.10: Storyboard für Auswertungsbildschirm

Zu dem Auswertungsbildschirm gehört die Klasse 'AuswertungThemaViewModel'. Beim Erzeugen dieses View-Modells wird das 'Quiz'-Objekt ausgewertet. Dies bedeutet, dass die eingegebenen Antworten mit den Lösungen verglichen werden und eine entsprechende Auswertungsmeldung erstellt wird. Die Abbildung 3.11 zeigt den Auswertungsbildschirm, welcher nach Bearbeitung der Fragen angezeigt wird.

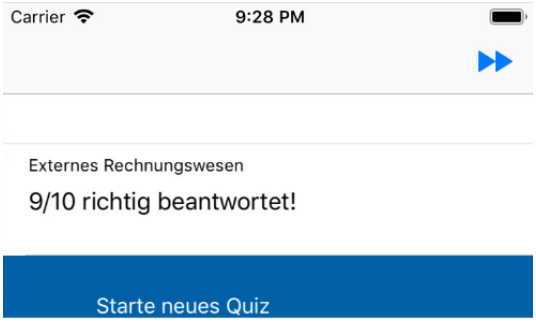


Abbildung 3.11: Ergebnis des Multiple-Choice-Test

Der Bildschirm zeigt, dass die Anwenderin keine Möglichkeit mehr hat, um zur Bearbeitung der Fragen zurückzunavigieren. Dies würde andernfalls das Ergebnis verfälschen.

Die Anwenderin erhält durch Betätigung der 'Weiter'-Schaltfläche die Möglichkeit, die Fragen erneut zu betrachten und die Eingaben mit den Lösungen zu vergleichen. Die Abbildung 3.12 zeigt die Lösung der ersten Frage. Die einzelnen Bildschirme enthalten ebenfalls Schaltflächen, damit eine Navigation zwischen den Fragen möglich ist. Auf dem letzten Bildschirm ist eine Schaltfläche vorhanden, um einen neuen Multiple-Choice-Test zu starten.

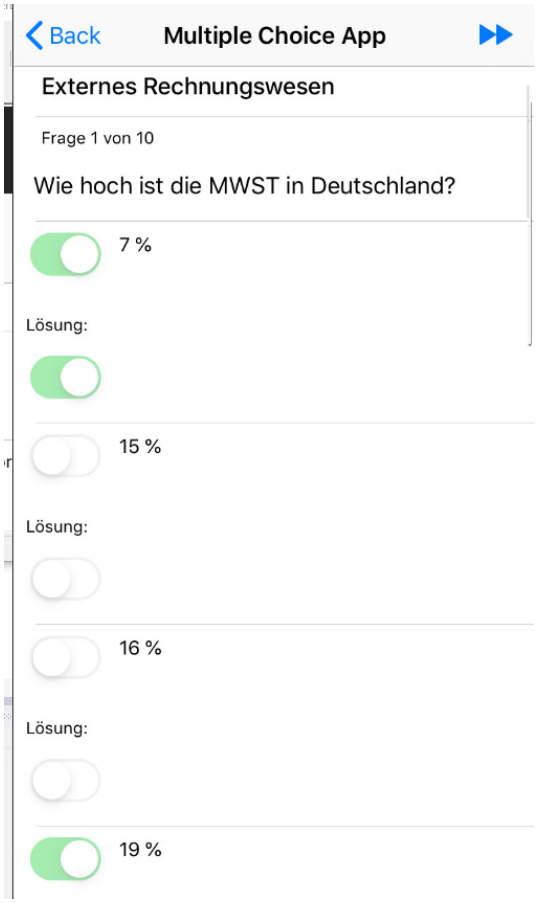


Abbildung 3.12: Lösung der ersten Frage

3.6.2 Servicemethoden

Die Ergebnisse des Multiple-Choice-Tests werden mit Methoden aus der Serviceklasse 'QuizService' erstellt. Das nachfolgende Listing zeigt die Auswertung der eingegebenen Antworten.

Listing 3.16: Auswertung einer Frage

```
 public bool FrageRichtigBeantwortet(Frage frage)
{
            Boolean fragerichtig = true;

            if (frage.antworten!=null && frage.antworten.Count!=0)
            {
                foreach (Textantwort a in frage.antworten)
                {
                    if (a.wahr && !a.Auswahl)
                    {
                        fragerichtig = false;
                    }
                    else if (!a.wahr && a.Auswahl)
                    {
                        fragerichtig = false;
                    }
                }
            }
          ...
}
```

Im Wesentlichen werden alle Antwortmöglichkeiten einer Frage geprüft. Sobald eine Abweichung zwischen der Eingabe der Anwenderin und der Lösung festgestellt wird, so wird die Frage als 'nicht richtig beantwortet' gekennzeichnet.

Die Klasse 'QuizService' enthält die Methode 'createAuswertungText'. Diese Methode zählt die Anzahl der richtigen Antworten und erstellt einen entsprechenden Auswertungstext.

3.7 Bilder als Antwortmöglichkeiten (Anforderung 10)

Die Multiple-Choice-App zeigt Fragen an, deren Antworten in Bildform hinterlegt sind.

3.7.1 Fachklassen und Datenbank

Das Klassendiagramm 3.4 auf Seite 33 zeigt den Zusammenhang zwischen der Klasse 'Antwort' und den beiden Unterklassen 'Textantwort' sowie 'Bildantwort'. Die Bildantwort enthält das anzuzeigende Bild als byte-Datenfeld. Die Vererbungsbeziehung erleichtert die Erstellung der Auswertung, da die gleichen Eigenschaften ausgewertet werden.

Das nachfolgende Listing zeigt die Fachklasse 'Bildantwort':

Listing 3.17: Fachklasse Bildantwort

```
public class Bildantwort : Antwort
{
        public byte[] Bild { get; set; }
}
```

Die Klasse enthält keine Annotationen für die Speicherung in die Datenbank, da diese bereits in der Oberklasse 'Antwort' enthalten sind. Die Fachklasse Bildantwort enthält lediglich das benötigte byte-Datenfeld. Da das verwendete ORM 'SQLite-Net-Pcl' mit dem Datentyp 'byte[]'

umgehen kann, wird die benötigte Bilddatei aus der Datenbank geladen. Die Datenbanktabelle 'Bildantwort' enthält die Bilddatei im Datenfeld 'Bild', welches in SQLite als Datentyp 'blob' definiert ist.

Der Quellcode der Multiple-Choice-App wurde an mehreren Stellen angepasst, damit sowohl die Text- als auch die Bildantworten ausgewertet werden. Somit sind alle Voraussetzungen erfüllt, um Bilder anzuzeigen.

3.7.2 Benutzeroberfläche

Das Storyboard 'QuestionTableView' wurde um eine Prototypzeile erweitert. Diese Prototypzeile enthält eine Komponente zum Anzeigen der Bilder vom Typ 'UIImageView'. Desweiteren sind in dieser Prototypzeile ebenfalls die beiden 'Switch'-Elemente für die Auswahl sowie für die Lösung vorhanden. Die Abbildung 3.13 zeigt das Storyboard für die Bildantworten.

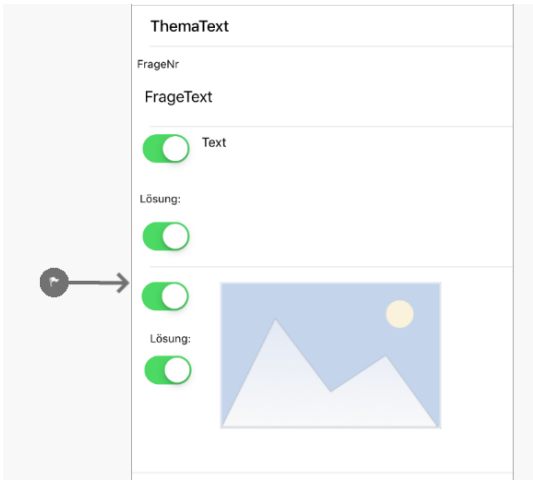


Abbildung 3.13: Storyboard für Bildantwort

Die erstellte Prototypzeile ist vom Typ 'QuestionTableBildantwortTextCell'. Da sämtliche Elemente der Benutzeroberfläche über MVVMCross an das View-Modell gebunden werden, ergibt sich nun eine Herausforderung: Die verwendete Oberflächen-Komponente, vom Typ 'UIImageView', benötigt die Bilddatei als Objekt des Typs 'UIImage'. UIImageView

Der Typ 'UIImage' gehört zu Xamarion.iOS und ist somit plattformspezifisch. Da das View-Modell jedoch plattformunabhängig funktionieren soll, kann das Bild im View-Modell nur als byte-Datenfeld zur Verfügung gestellt werden. Daraus folgt, dass die verwendeten 'UIImageView' nicht ohne weiteres an das byte-Datenfeld gebunden werden kann.

Dieses Problem wurde durch einen ValueConverter gelöst. Das nachfolgende Listing zeigt, wie die verwendete Komponente an das byte-Datenfeld gebunden wird. Dabei findet mithilfe des ValueConverters 'BytesToUIImage' die Umwandlung des byte-Datenfeldes in ein Objekt vom Typ 'UIImage' statt. ValueConverter

Listing 3.18: UIImageView an byte-Datenfeld über ValueConverter binden

```
set.Bind(AntwortImageView).For(v=>v.Image).To(vm => vm.AntwortBild).WithConversion("
    BytesToUIImage");
```

Die Abbildung 3.14 zeigt eine Frage mit Bildantworten.

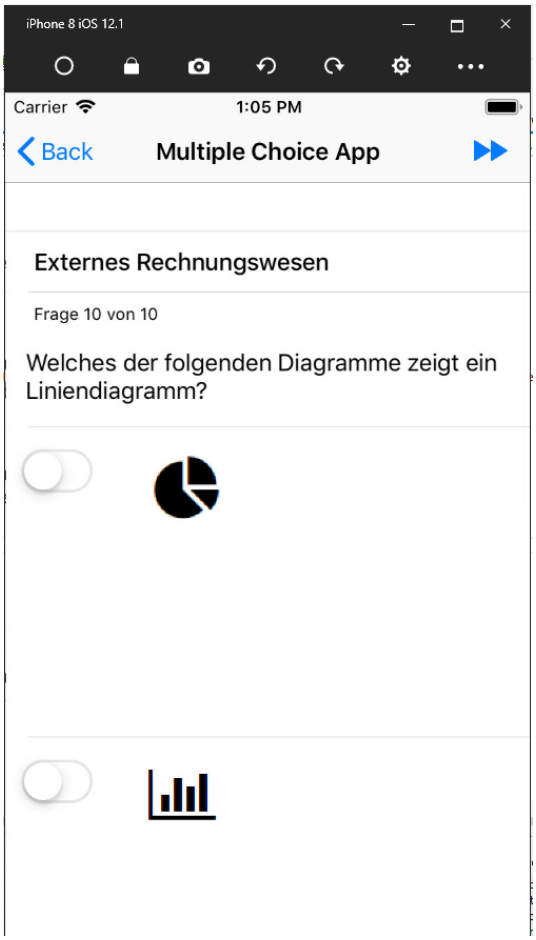


Abbildung 3.14: Frage mit Bildantworten

3.8 Anzahl Fragen festlegen (Anforderung 12)

Standardmäßig wird ein Multiple-Choice-Test mit zehn Fragen erstellt. Mithilfe einer 'Stepper'-Komponente ist es möglich, die Anzahl der Fragen beliebig zu erhöhen. Die Abbildung 3.15 zeigt die entsprechende Komponente.

Sofern die Anwenderin eine höhere Anzahl einstellt, als tatsächlich Fragen vorhanden sind, so werden alle Fragen aus den ausgewählten Themen bearbeitet.

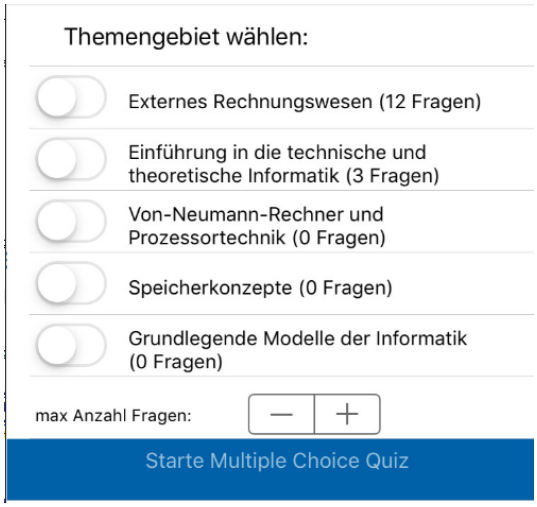


Abbildung 3.15: Anzahl Fragen anpassen

3.9 Offline-Implementierung (Anforderung 14)

Mit der Umsetzung dieser Anforderung wurde sichergestellt, dass die Multiple-Choice-App auch dann verwendet werden kann, wenn der Webserver nicht erreichbar ist. Die Multiple-Choice-App lädt die Daten bei der Ausführung eines Multiple-Choice-Tests nicht direkt vom Server, sondern aus der SQLite-Datenbank des Endgerätes. Dies ist möglich, da bei jedem Start der Multiple-Choice-App eine Synronisation der sqllite-Datenbank mit den Daten aus der Webanwendung ausgeführt wird.

3.9.1 Ladebildschirm

Beim Start der Multiple-Choice-App wird der Ladebildschirm angezeigt. Die Abbildung 3.16 zeigt diesen Ladebildschirm.

Der Ladevorgang sollte maximal fünf Sekunden dauern. In dieser Zeit werden die Themen und Fragen über den REST-Webservice geladen und in die Datenbank gespeichert. Sofern der Server nicht innerhalb von fünf Sekunden antwortet, so wird der Ladevorgang abgebrochen.

Die Wartezeit ist im Quellcode in der Klasse 'MCAPP_PROPERTIES' über die Variable 'TIMEOUT_IN_SECONDS_FOR_ALIVE_CHECK' einstellbar.

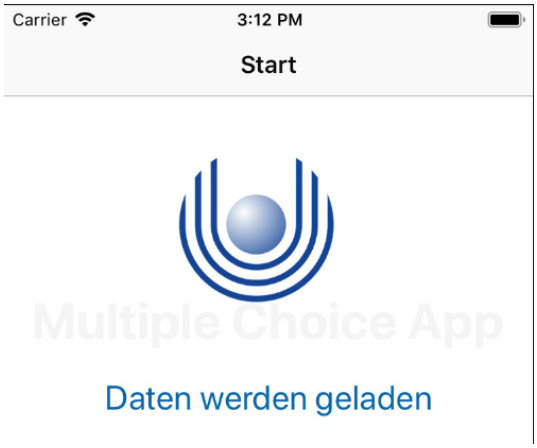


Abbildung 3.16: Ladebildschirm beim Start der Multiple-Choice-App

3.9.2 Demomodus

Damit die Anwenderin die Multiple-Choice-App verwenden kann, muss zunächst eine Datensyncronisation stattgefunden haben. Dies setzt jedoch voraus, dass der Webserver beim Start der App erreichbar ist und dass bereits Daten erfasst wurden.

Damit die App auch dann getestet werden kann, wenn keine Daten vorhanden sind, wurde ein Demomodus umgesetzt. Der Demomodus wird dann aktiv, wenn die Datenbank auf dem Gerät keine Themen enthält. Die Abbildung 3.17 zeigt die Thementabelle im Demomodus.

Wird das vorhandene Demo-Thema ausgewählt, so kann die Anwenderin eine Demofrage bearbeiten.

Sofern beim nächsten Start der Multiple-Choice-App die Datensyncronation durchgeführt werden konnte, so wird das Demo-Thema nicht mehr angezeigt. Desweiteren wird über das Demo-Thema keine Auswertung gespeichert.

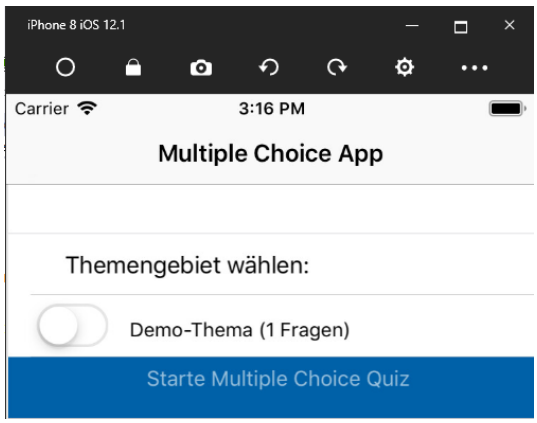


Abbildung 3.17: Demomodus

3.9.3 Themen und Fragen über REST-Service laden

Das Laden der Themen und Fragen erfolgt über die Klasse 'MCAPPWebService'. Diese Serviceklasse enthält die Methode 'isAlive'. Mit dieser Methode wird geprüft, ob der Server erreichbar ist.

Die Klasse 'MCAPPWebService' enthält außerdem die Methoden 'GetFragen' sowie 'GetThemen'. Zu dieser Serviceklasse gehört desweiteren die Klasse 'MCAPPWebRepositorie'. Dieses Repositorie greift mithilfe eines Http-Clients auf den Webserver zu, um die angefragten Daten über den REST-Webservice zu laden.

Bevor die benötigten Ressourcen geladen werden können, muss zuvor ein Authentifizierungstoken erstellt werden. Dieses Verfahren wurde bereits im Abschnitt 3.5.2 auf Seite 48 erläutert. Das folgende Listing zeigt den Quellcode, um das Autentifzierungstoken zu laden:

Listing 3.19: Authentifizierungstoken laden

```
 public async Task<Token> holeToken()
{
            HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Post, "api/
                authenticate");
            request.Content = new StringContent("{\"password\":\"" + MCAPP_PROPERTIES.
                SERVER_PASSWORD + "\",\"username\":\"" + MCAPP_PROPERTIES.SERVER_USER
                + "\",␣\"rememberMe\":true}",
                                    Encoding.UTF8,
                                    "application/json");

            try
            {
                var response = await httpClient.SendAsync(request);
                var json = await response.Content.ReadAsStringAsync().ConfigureAwait(
                    false);
                return JsonConvert.DeserializeObject<Token>(json);
            }
            catch(Exception e)
            {
                throw new MCAPPWebserviceException("Webservice␣fehler", e);
            }
        }
```

Im Wesentlichen wird ein POST-Request erstellt. Dieser Request enthält den Usernamen und das Kennwort für den Zugriff auf den Autorenzugang. Der Webservice sendet als Antwort einen Response im JSON-Format. Mithilfe eines Konverters wird das Token in das benötigte Format überführt.

Die Methode 'GetThemen' erzeugt nun einen GET-Request und fügt den Token in den Header für die Authentifzierung ein. Der Request wird mithilfe des Http-Clients an den Server gesendet. Die Antwort des Servers erfolgt im JSON-Format. Mithilfe eines Konverters wird die Liste der Themen-Objekte in das benötigte Format überführt. Das nachfolgende Listing zeigt den verwendeten Quellcode.

Listing 3.20: Themen werden vom Server geladen

```
 public async Task<List<Thema>> GetThemen()
{
                if (token == null)
                {
                    this.token = await holeToken();
                }
```

```
                httpClient.DefaultRequestHeaders.Authorization = new
                    AuthenticationHeaderValue("Bearer", token.ID_TOKEN);
                HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, "
                    api/themas");
                var response = httpClient.SendAsync(request).Result;
                var json = response.Content.ReadAsStringAsync().Result;

                List<Thema> themen = JsonConvert.DeserializeObject<List<Thema>>(json);

                return themen;
}
```

3.9.4 Syncronisation der SQLite Datenbank

Der Ladebildschirm wird über die Klasse 'StartViewModel' verwaltet. Diese View-Modell-Klasse enthält mehrere Syncronisationsmethoden.

Zunächst wird anhand der Methode 'isAlive' geprüft, ob der Webserver erreichbar ist. Anschließend werden nacheinander die Themen und Fragen geladen und mit dem Datenstand in der SQLite-Datenbank verglichen.

Wenn neue Datensätze vorhanden sind, so werden diese in die Datenbank gespeichert. Vorhandene Datensätze werden aktualisiert.

Sofern die SQLite-Datenbank Themen oder Fragen enthält, die nicht geliefert wurden, so werden diese entfernt.

Mit diesen Methoden sollte sichergestellt sein, dass nach dem Abgleich auf dem Endgerät der gleiche Datenstand des Servers vorhanden ist. Dieser Datenstand steht somit auch im Offline-Modus zur Verfügung.

Kapitel 4

Multiple-Choice-App Webanwendung

Dieses Kapitel beschreibt den Autorenzugang. Die Kursbetreuerin nutzt den Autorenzugang, um Fragen und Themen für die Nutzung der Multiple-Choice-App anzulegen. Desweiteren kann die Kursbeteuerin in die Auswertungen einsehen. Die einzelnen Anforderungen werden in den nachfolgenden Abschnitten näher erläutert.

4.1 Zugriff auf Autorenzugang über ein Benutzerkonto (Anforderung 13)

Im Abschnitt 2.3.5 auf Seite 19 wurde bereits erläutert, dass das Grundgerüst der Webanwendung mithilfe des Frameworks JHipster erstellt wurde. Zu diesem Grundgerüst gehört eine Benutzerverwaltung, die dem Schutz vor unberechtigtem Zugriffen dient.

Der Autorenzugang enthält zwei Anwendungsprofile. Für die Entwicklung des Autorenzugang wurde ausschließlich das Entwicklungsprofil verwendet. Das Entwicklungsprofil verwendet für die Datenspeicherung eine h2-Datenbank. Desweiteren sind bereits zwei Benutzerkonten vorhanden.

Anwendungsprofil

Das erste Benutzerkonto heißt 'user' und erhält Zugriff auf alle Entitäten und Auswertungen. Das zweite Benutzerkonto heißt 'admin' und hat darüberhinaus die Berechtigung, um in die Datenbank einzusehen. Mit dem Benutzerkonto 'admin' ist es weiterhin möglich, das Serverprotokoll zu lesen sowie weitere Wartungsfunktionen zu nutzen.

Administrator

Das zweite Anwendungsprofil ist für den Produktivbetrieb gedacht. Für den Produktivbetrieb ist eine mysql-Datenbank vorgesehen. Die beiden Benutzerkonten 'user' und 'admin' sind nicht für den Produktivbetrieb gedacht. Die benötigten Benutzerkonten sind stattdessen explizit anzulegen.

mySQL

Damit Benutzerkonten angelegt werden können, muss in der entsprechenden Konfigurationsdatei ein eMailkonto hinterlegt werden. Die entsprechenden Konfigurationsdateien heißen 'application-dev.yml' sowie 'application-prod.yml'.

eMailkonto hinterlegen

Die Abbildung 4.2 zeigt den Startbildschirm des Autorenzugangs im Entwicklungssystem.

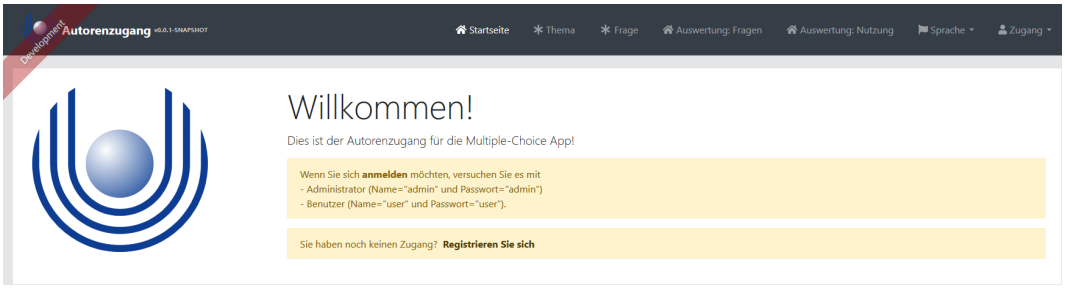


Abbildung 4.1: Startbildschirm des Autorenzugangs

Der Startbildschirm wurde für den Autorenzugang angepasst und zeigt das Logo der FernUniversität. Die Kursbetreuerin erhält den Hinweis, dass für den Zugriff ein Benutzerkonto benötigt wird. Sobald die Kursbetreuerin eine der Schaltflächen betätigt, so erscheint ein Anmeldebildschirm. Dieser Anmeldebildschirm wird in der Abbildung 4.2 gezeigt.

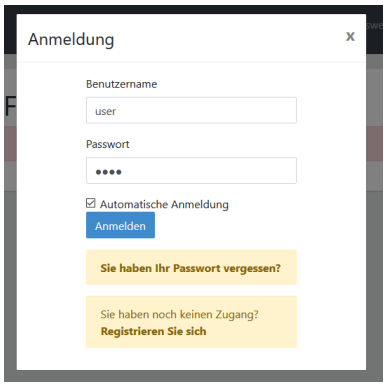


Abbildung 4.2: Anmeldung an Autorenzugang

Nach Eingabe einer gültigen Benutzername/Passwort-Kombination ist der Zugriff auf den Autorenzugang vollumfänglich möglich.

4.2 Themen anlegen und bearbeiten (Anforderung 15)

Die Themen gehören zu den Entitäten des Autorenzugangs. Für alle Entitäten wurde unter Verwenden des JHipster-Frameworks das Grundgerüst der Benutzeroberflächen erstellt. Zu diesem Grundgerüst gehört eine Tabelle für die Anzeige der vorhandenen Einträge sowie Dialoge zum Anlegen und Bearbeiten der Einträge. Um die Anforderungen des Autorenzugangs vollständig zu erfüllen, wurden die Benutzeroberflächen an mehreren Stellen angepasst.

4.2.1 Übersicht und Bearbeitung

Die Navigationsleiste im oberen Bereich der Webanwendung enthält verschiedene Schaltflächen. Durch Betätigung der Schaltfläche 'Thema' werden die Themengebiete für die Fragen angelegt oder bearbeitet. Die Abbildung 4.3 zeigt die Übersichtstabelle der Themen.

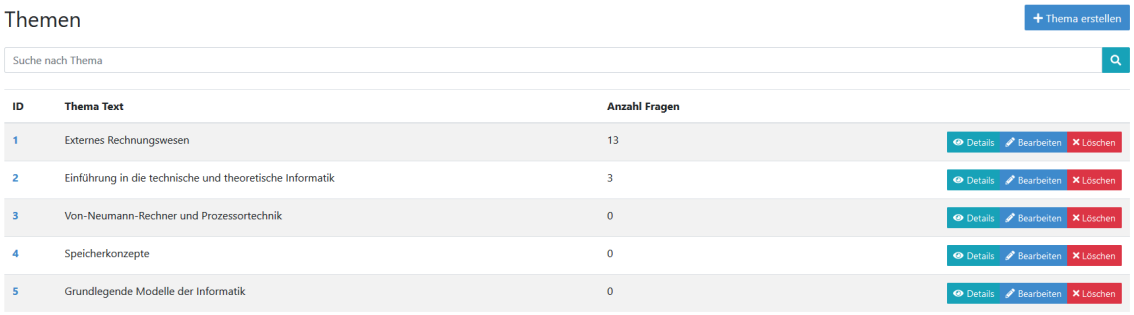


Abbildung 4.3: Themenübersicht

Durch Betätigung der Schaltfläche '+ Thema erstellen' wird ein neues Thema angelegt. Die Abbildung 4.4 zeigt die Eingabe des Thementextes.

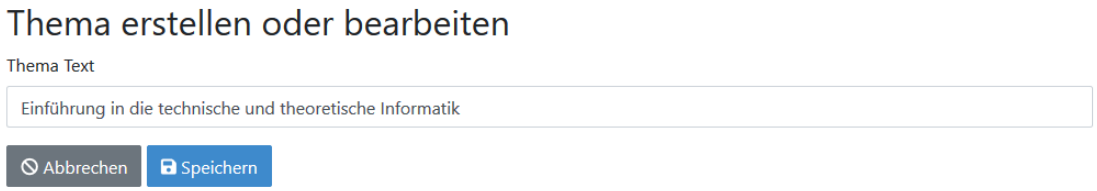


Abbildung 4.4: Thema bearbeiten

Bereits vorhandene Einträge werden über die Schaltflächen, innerhalb der Übersichtstabelle, bearbeitet oder gelöscht. Die Themeneinträge können desweiteren über Links an anderer Stelle des Autorenzugangs geöffnet und bearbeitet werden.

4.2.2 Anzahl der Fragen eines Themas anzeigen

Die Abbildung 4.3 zeigt in der Thementabelle die Anzahl der bereits zugeordneten Fragen. Für diese Angabe wurde der Quellcode an mehreren Stellen angepasst.

Die Webseite 'thema.component.html' wurde um eine zusätzliche Spalte erweitert. Um in dieser Spalte die Anzahl der vorhandenen Fragen anzuzeigen, muss zuvor eine zusätzliche Abfrage ausgeführt werden.

Es wäre hier möglich gewesen, einen weiteren REST-Service bereitzustellen, welcher die Fragen zu einem bestimmten Thema zählt. Das Listing A.2 im Anhang auf Seite I zeigt eine mögliche Methode. Diese Methode benötigt die ID eines Themas und setzt diese in einen Suchstring ein. Die vorhandene Implementierung generiert aus diesem Suchstring eine Datenbankabfrage und liefert die Fragen des angegebenen Themengebietes.

Eine einfachere Möglichkeit zeigt sich bei Betrachtung der Fachklasse 'Thema'. Die Klasse 'Thema' enthält eine Beziehung zu den Frage-Objekten, welche in dem Set 'frageIDS' abgebildet ist. Dieses Set könnte verwendet werden, um die Fragen des Themas zu zählen. Das Listing 4.1 zeigt das verwendete Set, innerhalb der Klasse 'Thema'.

Listing 4.1: Einstellung des FetchTypes auf EAGER am Set 'frageIDS'

```
@OneToMany(mappedBy = "thema", fetch = FetchType.EAGER)
@Cache(usage = CacheConcurrencyStrategy.NONE)
private Set<Frage> frageIDS = new HashSet<>();
```

Bei der Verwendung dieses Sets stellte sich heraus, dass dieses leer ist. Das liegt daran, dass der REST-Service die Ergebnisliste aus der Datenbank lädt und sofort in ein JSON-Objekt konvertiert. Die Ergebnisliste ist leer, da der FetchType standardmäßig auf LAZY eingestellt war.

FetchType

Mit der Einstellung LAZY soll erreicht werden, dass kleinere Datenbankabfragen abgesetzt werden und fehlende Daten bei Bedarf nachgeladen werden. Dieser Bedarf wird jedoch nicht festgestellt, wenn eine Konvertierung durchgeführt wird. Um dieses Problem zu lösen, wurde der FetchType[59] auf EAGER eingestellt. Mit EAGER wird bei der Datenbankabfrage die Beziehung zur FragenTabelle hergestellt und folglich ein komplettes JSON-Objekt mit Themen und Fragen zurückgeliefert.

LAZY

EAGER

Ein weiteres Problem ist die Standardeinstellung der Cache-Strategie. Der zuvor aktive Cache führte zu dem Problem, dass nach dem Anlegen neuer Fragen die Anzahl nicht erhöht wurde. Da der Cache dafür sorgt, dass gleiche Abfrage nicht erneut ausgeführt werden, wird ggfls. die alte Anzahl zurückgeliefert. Aus diesem Grund wurde die Cache-Stategy auf NONE gestellt, sodass nach dem Anlegen neuer Fragen auch eine aktuelle Anzahl in der Thementabelle angezeigt wird.

Cache

Die Anzahl der Fragen ergibt sich nun aus der Länge des Sets 'frageIDS'. Diese Länge wurde in einer neuen Spalte der Webseite 'thema.component.html' eingefügt.

Es ist nun zu beachten, dass es auch Themen geben könnten, denen noch keine Fragen zugeordnet wurden. Damit es bei Ausführung der Angular Direktiven zu keinem Fehlverhalten kommt, wird nun eine Auswahlanweisung mithilfe einer Angular Expression durchgeführt.

Angular Expression

Das Listing 4.2 zeigt, dass vor dem Zählen des Sets 'frageIDS' geprüft wird, ob dieses initialisert wurde. Sofern die Initialisierung nicht stattgefunden hat, so wird die Zahl 0 zurückgeliefert.

Listing 4.2: Anzahl Fragen anzeigen

```
<td>{{ thema.frageIDS != null ? thema.frageIDS.length : 0 }}</td>
```

4.3 Fragen anlegen und bearbeiten (Anforderung 6)

Die Fragen und die dazugehörigen Antworteinträge gehören zu den Entitäten des Autorenzugangs. Auch für diese Entitäten wurde das Grundgerüst der Benutzeroberflächen erzeugt. Das Framework erstellt jedoch für jede Entität eine eigene Übersichtstabelle, sowie einen eigenen Erstellungsdialog. Da jedoch die Antwortmöglichkeiten immer zu einer Frage gehören, wurde die Benutzeroberflächen sinngemäß zusammengeführt.

4.3.1 Übersicht und Suchfunktion

Die Abbildung 4.5 zeigt die Übersichtstabelle der Fragen-Einträge an.

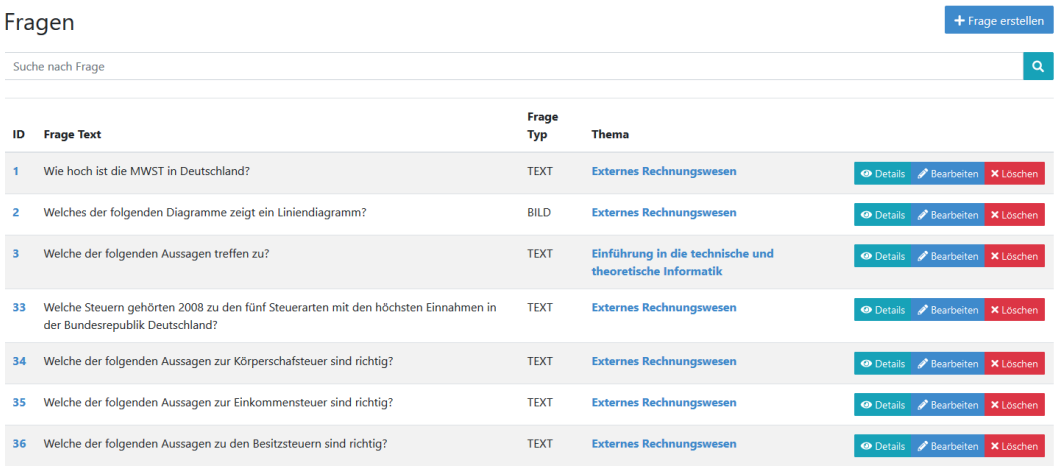


Abbildung 4.5: Übersichtstabelle für Fragen

Die Übersichtstabelle funktioniert im Wesentlichen wie bereits im vorherigen Abschnitt, bei den Themen, erläutert.

Die Tabelle zeigt zusätzlich an, ob es sich um eine Text- oder Bildfrage handelt. Desweiteren wird angezeigt, welchem Thema die Fragen zugeordnet wurden.

Zu der Übersichtsseite gehört auch eine Suchfunktion. Wenn die Kursbetreuerin einen Suchbegriff eingibt, so wird der eingegebene Begriff mit allen angezeigten Feldern verglichen. Die Kursbetreuerin kann also die Menge der angezeigten Fragen, durch ein Wort aus dem Fragetext, einschränken. Die Suchfunktion kann aber auch auf das Thema angewendet werden. Suchfunktion

4.3.2 Bearbeitung einer Frage

Zu jeder Frage gehört ein Fragetext. Die Eingabe des Textes wurde auf 255 Zeichen beschränkt. Eine Frage kann entweder vom Typ 'Text' oder vom Typ 'Bild' sein. Der Fragetyp wird aus einer Auswahlliste gewählt.

Mit der Auswahl des Fragetyps ist es möglich, entweder Text- oder Bildantworten zu erfassen. Eine kombinierte Eingabe von Text- und Bildantworten ist aktuell nicht vorgesehen und wird daher nicht unterstützt. Für jede Frage können bis zu acht Antwortmöglichkeiten erfasst werden. Sofern bereits acht Antwortmöglichkeiten vorhanden sind, so wird die Schaltfläche zur Erstellung einer weiteren Antwortmöglichkeit inaktiv.

Die Abbildung 4.6 zeigt den Dialog zur Bearbeitung einer Frage. Vorhandene Textantworten werden in einer darunterliegenden Tabelle angezeigt.

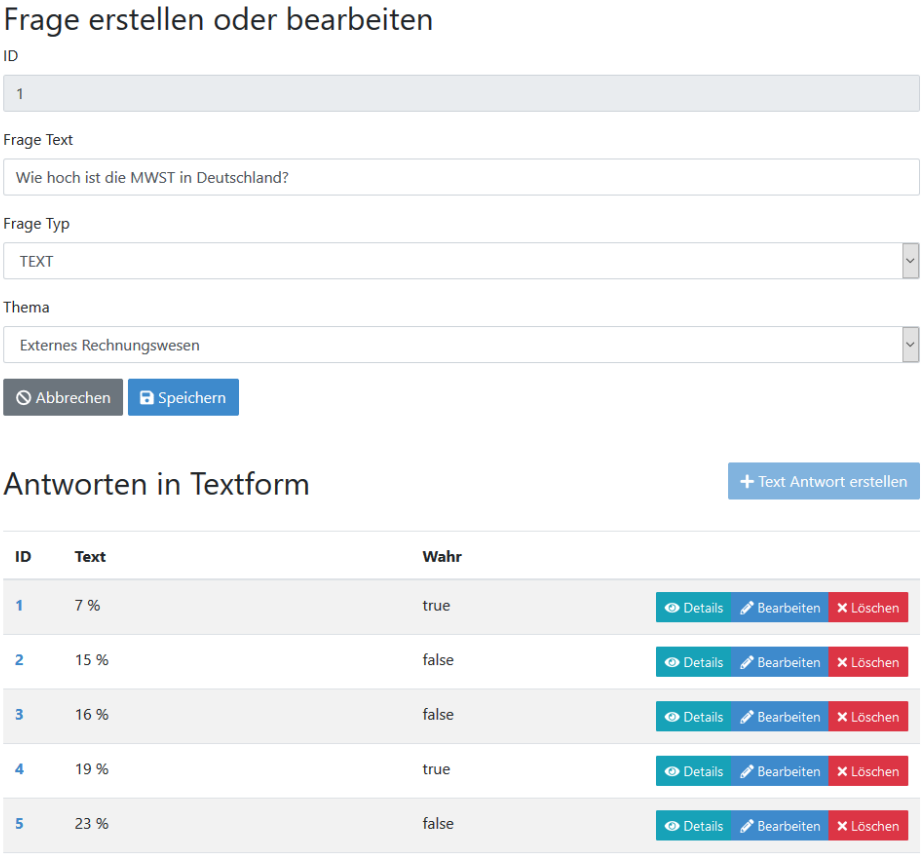


Abbildung 4.6: Frage und Antwortmöglichkeiten anzeigen

Die Antwortmöglichkeiten werden in einem eigenen Dialog erfasst. Dieser Dialog wird über die Schaltfläche '+ Text Antwort erstellen' geöffnet.

Für eine Textwort ist die Eingabe eines Antworttextes erforderlich. Mithilfe einer Checkbox wird eine Antwortmöglichkeit als 'wahr' markiert. Die Abbildung 4.7 zeigt diesen Dialog.

Abbildung 4.7: Antwortmöglichkeit bearbeiten

4.4 Bilder für Antworten hinterlegen (Anforderung 11)

Wenn zu einer Frage die Antwortmöglichkeiten in Bildform zu erfassen sind, so wählt die Kursbetreuerin den FrageTyp 'Bild'.

Der Dialog zur Bearbeitung einer Frage zeigt daraufhin eine Schaltfläche zur Erfassung von Bildantworten an. Desweiteren werden bereits vorhandene Bildantworten in einer Tabelle darunter angezeigt.

Die Abbildung 4.9 zeigt eine Frage mit vorhandenen Bildantworten.

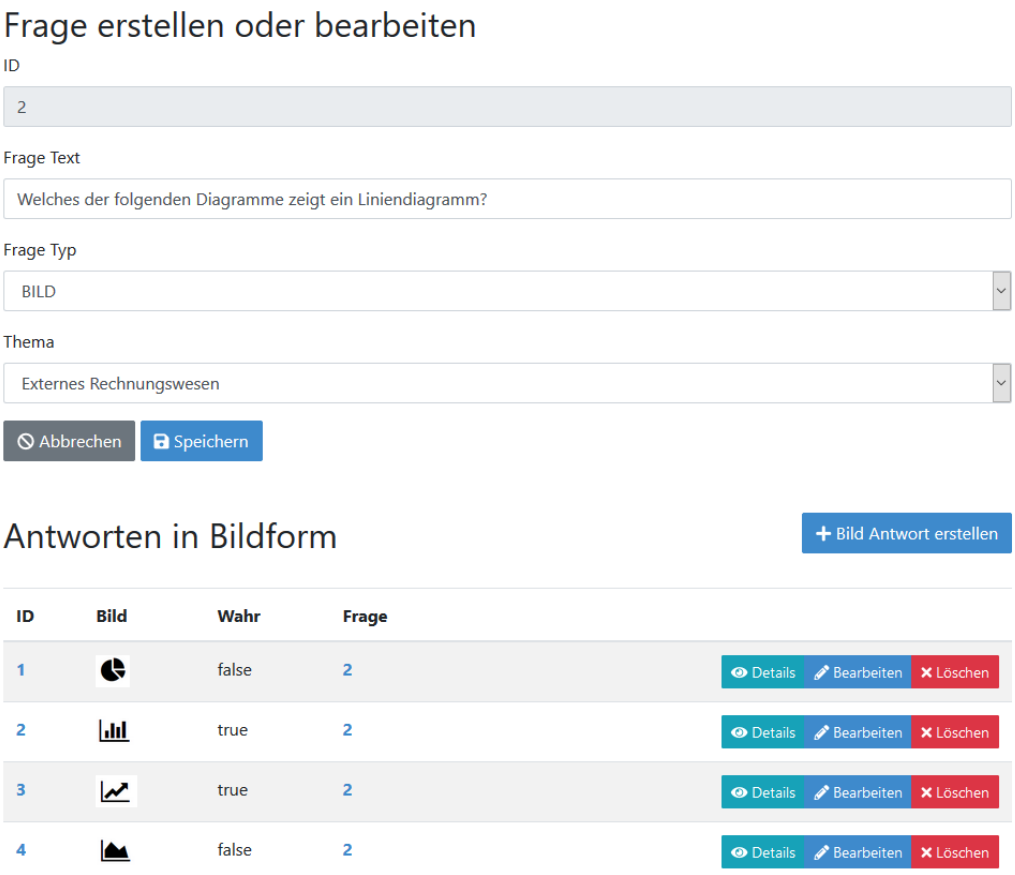


Abbildung 4.8: Frage mit Bildern als Antwortmöglichkeit

Für die Erfassung von Antwortmöglichkeiten in Bildform ist ein separater Erfassungsdialog vorhanden.

Bei der Erfassung einer Antwort in Bildform wählt die Kursbetreuerin eine Bilddatei aus dem Dateisystem aus. Nach Betätigung der Schaltfläche 'Speichern' wird die Bilddatei als byte-Datenfeld in die Datenbank übertragen.

Die Abbildung 4.9 zeigt diesen Erfassungsdialog.

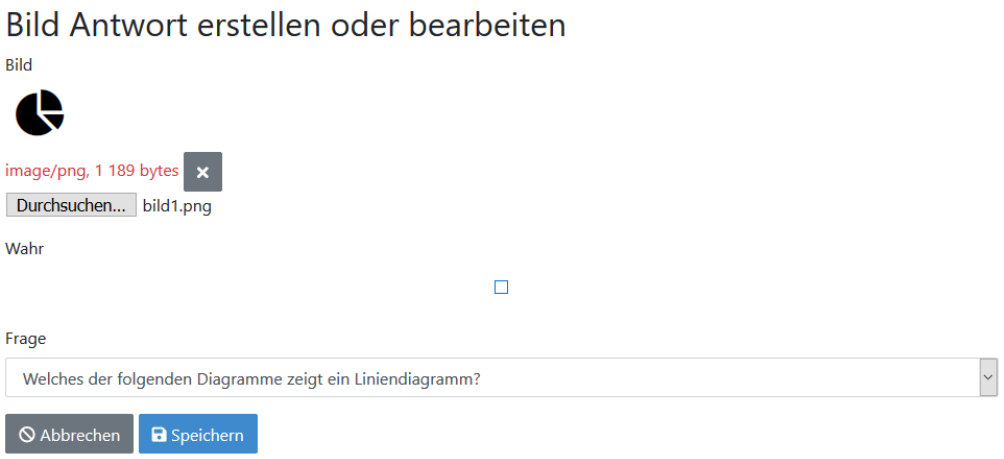


Abbildung 4.9: Bildantwort erstellen

4.5 Auswertung zur Benutzung der Multiple-Choice-App (Anforderung 8)

Der Autorenzugang wurde um eine Auswertungsfunktion erweitert. Die Kursbetreuerin wertet aus, wie häufig die Multiple-Choice-App in den letzten sieben Tagen genutzt wurde. Weiterhin zeigt die Auswertung an, wie häufig die Multiple-Choice-App in den letzten dreissig Tagen genutzt wurde.

Diese Auswertungen setzen voraus, dass bereits Daten zur Nutzung der Multiple-Choice-App gesendet wurden.

4.5.1 Auswertungsinformationen senden

Die Multiple-Choice-App führt mit jedem Programmstart eine Syncronisation der Datenbank aus. Da zu diesem Zeitpunkt eine Verbindung zum Server vorhanden ist, eignet sich dieser Zeitpunkt hervorragend für den Versand der Auswertungsinformationen.

In dem Klassendiagramm aus Abbildung 3.9 auf Seite 43 wurde die Fachklasse 'Quiz' erklärt. Für jede Ausführung eines Multiple-Choice-Tests wird ein Quiz-Objekt erstellt und in die SQLite-Datenbank gespeichert. Ein 'Quiz'-Datensatz enthält das Datum der Ausführung.

Weitere Informationen sind für die Auswertung der Nutzung nicht erforderlich.

Die Klasse 'StartViewModell' im Quellcode der Multiple-Choice-App wurde um die Methode 'synchronisiereAuswertung' erweitert. Diese Methode ermittelt eine Liste aller noch nicht gesendeten 'Quiz'-Instanzen. Die 'Quiz'-Instanzen werden über ein Post-Request an den Server gesendet und von dort in die Datenbank des Servers gespeichert.

Das nachfolgende Listing zeigt den verwendeten Quellcode für den Datenversand. Der Quellcode zeigt, dass für die Auswertung lediglich das Ausführungsdatum übergeben wird.

Listing 4.3: Versand des Ausführungsdatums

```
...
httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer
    ", token.ID_TOKEN);
HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Post, "api/quizzes");

String datum = quiz.datum.ToString("yyyy-MM-dd");
request.Content = new StringContent("{id:␣null,␣datum:␣" + datum + ",␣quizFrageIDS:␣
    null}",
                Encoding.UTF8,
                "application/json"
);
var response = await httpClient.SendAsync(request);
...
```

4.5.2 Auswertung in Autorenzugang anzeigen

Für die Auswertung wurde ein neues Angular-Modul geschrieben. Steyer schrieb zum Thema Angular Module: 'Um eine Anwendung zu strukturieren, fasst Angular die einzelnen Anwendungsbestandteile, z. B. Komponenten, zu Modulen zusammen.' [52, s.S. 57].

Module in Angular

Das Framework JHipster hat für die bisher verwendeten Entitäten bereits Angular-Module angelegt. Die Anforderungen zu den Auswertungen weichen jedoch von dem erstellten Grundgerüst der Anwendung ab, sodass zusätzliche Module erforderlich sind.

Das erforderliche Modul wird über die Datei 'nutzung.module.ts' definiert. Zu diesem Modul gehört die Komponente 'nutzung.component.ts'. Diese Komponente enthält den Quellcode sowie den Verweis zum Template 'nutzung.component.html'. Das Template wird verwendet, um die eigentliche Auswertung anzuzeigen. Der Quellcode dieser Komponente enthält die Berechnung der Auswertung. Da es sich um eine Angular-Komponente handelt, wurde der Quellcode in der Skriptsprache TypeScript geschrieben.

Bei dieser Berechnung wird, ausgehend vom heutigen Datum, sieben bzw. dreissig Tage zurückgerechnet. Als nächstes werden die vorhandenen 'Quiz'-Datensätze betrachtet. Jeder 'Quiz'-Datensatz, welcher im Betrachtungszeitraum liegt, wird gezählt.

Als Ergebnis liegt die gewünschte Nutzungsauswertung vor. Das nachfolgende Listing zeigt den verwendeten Quellcode für die Auswertung.

Listing 4.4: Auswertung für Nutzung berechnen

```
...
let sevenDays = new Date(Date.now());
let thirtydays = new Date(Date.now());
```

```
sevenDays.setDate(sevenDays.getDate() - 7);
thirtydays.setDate(thirtydays.getDate() - 30);
let seven = 0;
let thirty = 0;

this.quizzes.forEach(function(value) {
        if (value.datum.toDate() >= sevenDays) {
                seven++;
           }
           if (value.datum.toDate() >= thirtydays) {
                   thirty++;
           }
});
...
```

Die Kursbetreuerin erhält nach Betätigung der Schaltfläche 'Auswertung: Nutzung' die angeforderte Auswertung. Die Abbildung 4.10 zeigt eine entsprechende Auswertung.

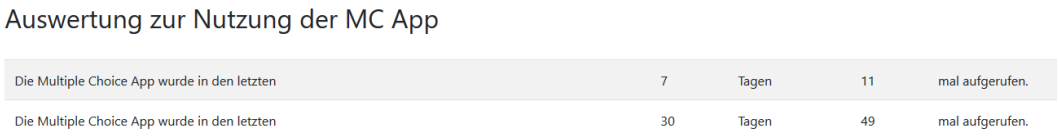


Abbildung 4.10: Auswertung über Nutzung der Mulitiple-Choice-App

4.6 Auswertung der Multiple-Choice-Fragen (Anforderung 9)

Der Autorenzugang enthält eine weitere Auswertungsfunktion. Die Kursbetreuerin wertet die Ergebnisse der Multiple-Choice-Fragen aus. In dieser Auswertung geht es darum, wie häufig eine Frage bearbeitet wurde. Desweiteren stellt die Kursbetreuerin fest, wie häufig die Frage richtig oder falsch beantwortet worden ist.

4.6.1 Auswertungsinformationen senden

In dem Klassendiagramm aus Abbildung 3.9 auf Seite 43 wurde die Fachklasse 'Quiz_Frage' erklärt. Für jede bearbeitete Fragen eines Multiple-Choice Tests wird auf dem Gerät ein' Quiz_Frage'-Objekt erstellt und in die SQLite-Datenbank gespeichert. Ein 'Quiz_Frage'-Datensatz enthält die Angabe 'richtig_beantwortet' sowie die ID einer Frage. Weitere Informationen sind für die Auswertung der Multiple-Choice Fragen nicht erforderlich.

Die Methode 'synchronisiereAuswertung' wurde erweitert und übersendet die auf dem Gerät vorhandenen 'Quiz_Frage'-Datensätze über einen Post-Request an den Server. Der REST-Service verarbeitet diesen Request und speichert diese Datensätze in die Datenbank des Servers.

Die Auswertungsinformationen werden für jede gespeicherte 'Quiz'-Instanz nur einmal an den Server übertragen. Damit ist sichergestellt, dass keine Ausführung eines Multiple-Choice-Tests mehrfach in der Auswertung erscheint.

4.6.2 Auswertung in Autorenzugang anzeigen

Für die Auswertung wurde ein weiteres Angular-Modul geschrieben. Das erforderliche Modul wird über die Datei 'auswertung-frage.module.ts' definiert. Zu diesem Modul gehört die Komponente 'auswertung-frage.component.ts'. Diese Komponente enthält den Quellcode sowie den Verweis zum Template 'auswertung-frage.component.html'. Der Quellcode dieser Komponente enthält die Berechnung der Auswertung.

Bei dieser Berechnung werden alle vorhandenen 'Quiz_Frage'-Datensätze nach der FrageID gruppiert. Für jede Frage wird nun gezählt, wie oft die Frage richtig oder falsch beantwortet wurde. Desweiteren wird die Gesamtzahl mitgezählt.

Das nachfolgende Listing zeigt den verwendeten Quellcode. Als Ergebnis wird eine Liste von Auswertungssätzen geliefert. Dabei entspricht jeder Eintrag genau einer vorhandenen Frage.

Listing 4.5: Auswertung der Fragen berechnen

```
...
let map = new Map<number, IQuizFrage>();

this.quizFrages.forEach(function(value) {
        if (!map.get(value.frage.id)) {
                value.anzRichtig = 0;
                value.anzFalsch = 0;
                value.anzGesamt = 0;
                map.set(value.frage.id, value);
        }
        let quizfrage = map.get(value.frage.id);
        quizfrage.anzGesamt++;
        if (value.richtig) {
                quizfrage.anzRichtig++;
        } else {
                quizfrage.anzFalsch++;
        }
});

this.quizFrages = Array.from(map.values());
```

Die Kursbetreuerin erhält, nach Betätigung der Schaltfläche 'Auswertung: Fragen', die angeforderte Auswertung.

Die Abbildung 4.11 zeigt eine entsprechende Auswertung.

Auswertung der Fragen

ID	Frage	Anz. Antworten	richtig beantwortet	falsch beantwortet
1	Wie hoch ist die MWST in Deutschland?	4	1	3
2	Welches der folgenden Diagramme zeigt ein Liniendiagramm?	2	1	1
3	Welche der folgenden Aussagen treffen zu?	2	0	2
41	Welche der folgenden Aussagen zu Steuerbefreiungen sind richtig?	2	0	2
39	Welche der folgenden Aussagen zur Steuerschuld und Steuerbefreiungen sind richtig?	1	1	0
71	Welche der folgenden Aussagen zu Gewinnausschüttungen sind richtig?	2	0	2
38	Wichtige förmliche Steuergesetze sind:	2	0	2
37	Welche der folgenden Aussagen zur Steuerhoheit sind richtig?	2	0	2
33	Welche Steuern gehörten 2008 zu den fünf Steuerarten mit den höchsten Einnahmen in der Bundesrepublik Deutschland?	2	1	1
36	Welche der folgenden Aussagen zu den Besitzsteuern sind richtig?	1	1	0
35	Welche der folgenden Aussagen zur Einkommensteuer sind richtig?	2	0	2

Abbildung 4.11: Auswertung der Mulitiple-Choice-Fragen

Kapitel 5

Fazit

Im Rahmen dieser Bachelor-Thesis wurde eine iOS-App für die Bearbeitung von Multiple-Choice Fragen entwickelt. Diese App wurde auf einer Entwicklungsplattform umgesetzt, welche die Weiterentwicklung dieser App auch für andere Plattformen ermöglicht.

Die Multiple-Choice-App lädt die Inhalte von einem Webserver. Die Inhalte werden mit jedem Start der App syncronisiert. Die Multiple-Choice-App funktioniert somit auch im Offlinebetrieb.

Für die Inhalte der Multiple-Choice-App steht ein webbasierter Autorenzugang zur Verfügung. Dieser Autorenzugang ist vor unauthorisierten Zugriffen geschützt. Der Autorenzugang ermöglicht die Erfassung und Bearbeitung von Fragen und Themen. Darüberhinaus enthält der Autorenzugang mehrere Auswertungsfunktionen.

Diese Bachelor-Thesis hat gezeigt, wie und mit welchen Mitteln eine Multiple-Choice-App entwickelt werden könnte. Sämtliche Anforderungen aus der Problembeschreibung wurden in der vorgegebenen Zeit umgesetzt.

Die Multiple-Choice-App und der Autorenzugang wurden mit modernen Programmiertechniken umgesetzt und ist für alle künftigen Anforderungen mit wenig Aufwand erweiterbar. Mit diesem Ergebnis sollte die Einführung einer Multiple-Choice-App für Studierende und Kursbetreuerinnen schon bald möglich sein.

Literaturverzeichnis

[1] Jim Bennett *Xamarin in Action*
Manning Publications Co, 2018

[2] Michael Kofler *Swift 2 : das umfassende Handbuch*
Rheinwerk Verlag, 1. Auflage 2015

[3] Thomas Theis *Einstieg in C# mit Visual Studio 2017*
Rheinwerk Verlag, 5. aktualisierte Auflage 2017

[4] Microsoft Docs *Xamarin.iOS*
https://docs.microsoft.com/de-de/xamarin/ios/

[5] Visual Studio-Tools für Xamarin *https://visualstudio.microsoft.com/de/xamarin*
[Aufruf 10.10.2018]

[6] Erstellen und Bewerten von Multiple-Choice-Aufgaben *Universität Hannover*
https://www.uni-hannover.de/fileadmin/luh/content/elearning/practicalguides2
/didaktik/elsa_handreichung_zum_erstellen_und_bewerten_von_mc-
fragen_2013.pdf)

[7] Richtlinien zur Erstellung von einfachen Multiple-Choice-Aufgaben nach Gronlund *Bernhard Jacobs, Universität Saarbrücken*
http://www.phil.uni-sb.de/FR/Medienzentrum/verweise/psych/aufgaben/mcguideline.html
-
Aufruf über https://web.archive.org mit Datum 23.02.2010

[8] Wissenschaftliches Arbeiten-Grundfragen, Orientierung, Werkzeuge *Univ-Prof. Dr. Stefan Strecker, Dr. Kristina Rosenthal*

[9] Developing and Validating Multiple-choice Test Items *Thomas M. Haladyna, 2015*

[10] Design and Implementation of a Multiple-Choice E-voting Scheme on Mobile System using Novel t-out-of-n Oblivious Signature* *SHIN-YAN CHIOU AND JIUN-MING CHEN*
http://jise.iis.sinica.edu.tw/JISESearch/pages/View/PaperView.jsf?keyId=160_2110

[11] Erstellung einer Webapplikation zur Online-Beantwortung von Multiple-Choice-Klausuren *Benjamin Höschele, 2003*

[12] Problem Multiple-Choice-Tests: Vom Ankreuzen und Raten bei Uniprüfungen *Miguel de la Riva*
https://derstandard.at/2000045417547/Vom-Ankreuzerln-und-Raten-bei-Unipruefungen
[Aufruf 03.07.2018]

[13] Balsamiq Mockups
https://balsamiq.com/products/
[Aufruf 06.07.2018]

[14] MacInCloud
http://www.macincloud.com/pricing/academic

[15] Github Student Developer Pack
https://education.github.com/pack

[16] Prüfungsordnung *https://www.fernuni-hagen.de/wirtschaftswissenschaft/studium/download/ordnungen/ba_winf_po.pdf*
[Aufruf 07.08.2018]

[17] *https://www.nuget.org/packages/sqlite-net-pcl/*
[Aufruf 26.08.2018]

[18] *https://www.sqlite.org/index.html* SQLite Home Page
[Aufruf 04.12.2018]

[19] Objektorientierte Systemanalyse *Dr. Andreas Bortfeldt*
[Sommersemester 2016]

[20] Betriebliche Informationssysteme *Prof. Dr. Lars Mönch*
[Wintersemester 2017/18]

[21] Apple iOS: a brief history *https://www.telegraph.co.uk/technology/apple/11068420/Apple-iOS-a-brief-history.html*
[Aufruf 09.10.2018]

[22] *https://www.apple.com/de/ios/ios-12/*
[Aufruf 09.10.2018]

[23] *https://web.archive.org/web/20130928164416/https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/TheiOSEnvironment/TheiOSEnvironment.html*
[Aufruf 09.10.2018]

[24] Thomas Sillmann *Swift 3 im Detail*
Carl Hanser Verlag, 2017

[25] Swift. Eine leistungsstarke offene Sprache, mit der jeder großartige Apps erstellen kann *https://www.apple.com/de/swift/*
[Aufruf 09.10.2018]

[26] Swift Has Reached 1.0 *https://developer.apple.com/swift/blog/?id=14*
[Aufruf 09.10.2018]

[27] Xcode 10 *https://developer.apple.com/xcode/*
[Aufruf 09.10.2018]

[28] Getting Started with Xamarin.Forms *https://docs.microsoft.com/de-de/xamarin/xamarin-forms/get-started/index*
[Aufruf 10.10.2018]

[29] Übersicht Cordova *https://cordova.apache.org/docs/de/latest/guide/overview/index.html* [Aufruf 10.10.2018]

[30] Visual Studio-Dokumentation *https://docs.microsoft.com/de-de/visualstudio/* [Aufruf 11.10.2018]

[31] Welcome to the MvvmCross documentation! *https://www.mvvmcross.com/documentation/* [Aufruf 11.10.2018]

[32] Kent Beck *Test Driven Development: By Example* Addison-Wesley Professional; Auflage: 01 (8. November 2002)

[33] Datenschutz-Grundverordnung DSVGO *https://dsgvo-gesetz.de/* [Aufruf 13.10.2018]

[34] How to: Specify the Web Server for Web Projects in Visual Studio *https://msdn.microsoft.com/en-us/library/ms178108.aspx* [Aufruf 04.11.2018]

[35] RESTful Webservices (1): Was ist das überhaupt? von Martin Helmich *https://www.mittwald.de/blog/webentwicklung-design/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt* [Aufruf 04.11.2018]

[36] James Martin *Managing the Data Base Environment* Pearson Education, Limited, 1983

[37] Hypertext Transfer Protocol – HTTP/1.1 *https://tools.ietf.org/html/rfc2616#section-6.1.1* [Aufruf 17.11.2018]

[38] Spring Boot *http://spring.io/projects/spring-boot* [Aufruf 17.11.2018]

[39] Craig Walls *Spring in action* Manning Publications Co

[40] Java EE - the Most Lightweight Enterprise Framework? *https://blogs.oracle.com/java/lightweight-enterprise-framework* [Aufruf 20.11.2018]

[41] Welcome to Apache Maven *https://maven.apache.org/* [Aufruf 21.11.2018]

[42] What is maven? *https://maven.apache.org/what-is-maven.html* [Aufruf 24.11.2018]

[43] Gradle Build Tool *https://gradle.org/* [Aufruf 21.11.2018]

[44] Installing Spring Boot *https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html* [Aufruf 21.11.2018]

[45] Java Persistence API *https://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html* [Aufruf 21.11.2018]

[46] H2 Database Engine *http://www.h2database.com/html/main.html*
[Aufruf 21.11.2018]

[47] What is JHipster? *https://www.jhipster.tech/*
[Aufruf 21.11.2018]

[48] JDL-Studio *https://start.jhipster.tech/jdl-studio/*
[Aufruf 26.11.2018]

[49] JHipster Domain Language (JDL) *https://www.jhipster.tech/jdl/*
[Aufruf 26.11.2018]

[50] Using JHipster in production *https://www.jhipster.tech/production/*
[Aufruf 10.12.2018]

[51] *https://angular.io/* Angular
[Aufruf 21.11.2018]

[52] Manfred Steyer & Daniel Schwab *Angular - Das Praxisbuch zu Grundlagen und Best Practices*
2. Auflage 2017, dpunkt.verlang GmbH

[53] Microservice-Architekturen nicht nur für agile Projekte *https://www.informatik-aktuell.de/entwicklung/methoden/microservice-architekturen-nicht-nur-fuer-agile-projekte.html*
[Aufruf 24.11.2018]

[54] The Java IDE for Professional Developers by JetBrains *https://www.jetbrains.com/idea/*
[Aufruf 24.11.2018]

[55] Builder design pattern *http://designpattern.co.il/Builder.html*
[Aufruf 01.12.2018]

[56] async (C# Reference) *https://docs.microsoft.com/de-de/dotnet/csharp/language-reference/keywords/async*
[Aufruf 05.12.2018]

[57] Parallel Computing *https://msdn.microsoft.com/de-de/vstudio/bb964701*
[Aufruf 06.12.2018]

[58] SoapUI: The World's Most Popular Automated API Testing Tool *https://www.soapui.org/*
[Aufruf 06.12.2018]

[59] FetchType (JavaEE 6) *https://docs.oracle.com/javaee/6/api/javax/persistence/FetchType.html*
[Aufruf 08.12.2018]

[60] Dr. Stefan Wohlfeil *Sicherheit im Internet: Kurseinheit 1: Sicherheit in der Informationstechnik*

[61] Microsoft Docs *https://docs.microsoft.com/de-de/xamarin/ios*
[Aufgerufen am 03.07.2018]

[62] Undertow *http://undertow.io/*
[Aufgerufen am11.12.2018]

[63] What Is NUnit? *https://nunit.org/*
[Aufgerufen am11.12.2018]

[64] Moq: an enjoyable mocking library *https://www.nuget.org/packages/Moq*
[Aufgerufen am11.12.2018]

Anhang A

Weitere Abbildungen

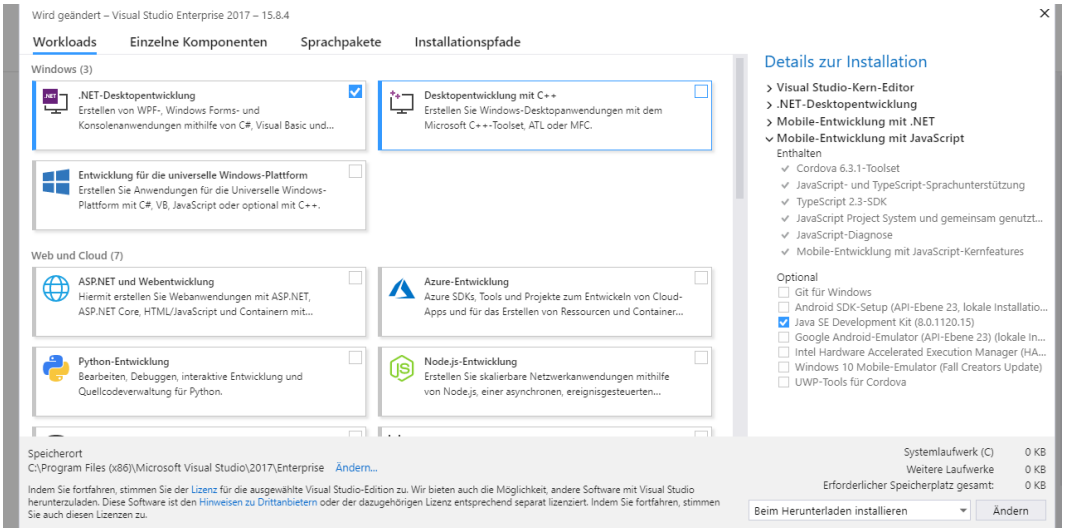


Abbildung A.1: Visual Studio Installation[30]

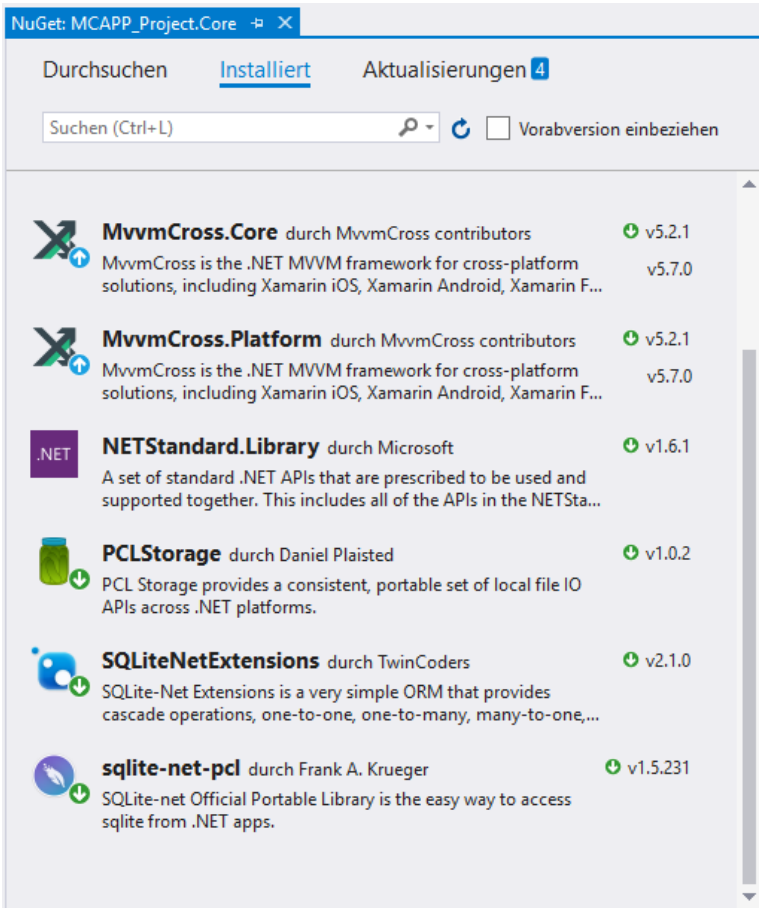


Abbildung A.2: Visual Studio Plugins

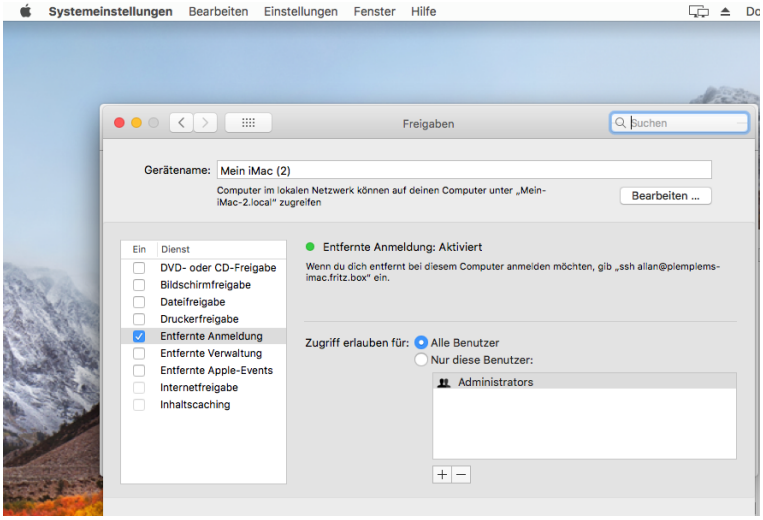


Abbildung A.3: Verbindung mit Mac

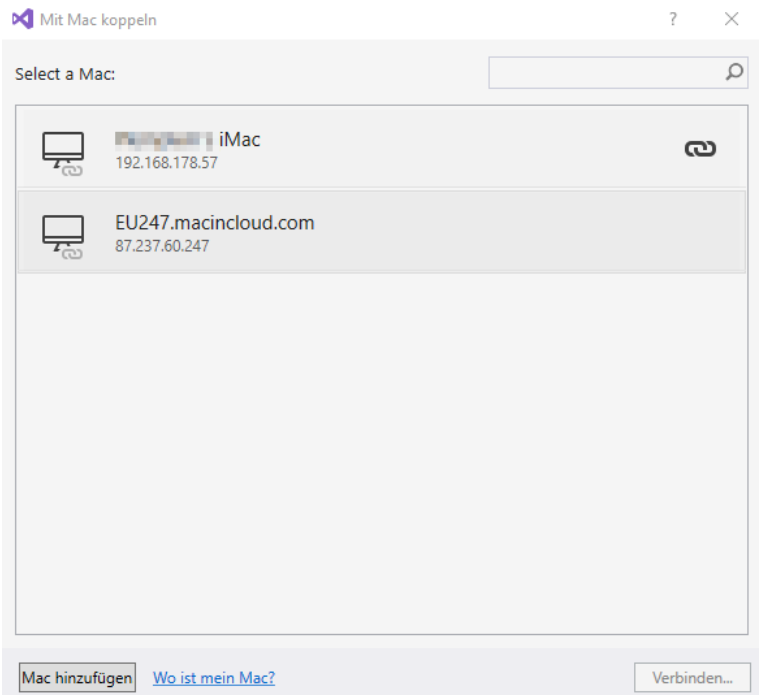


Abbildung A.4: Koppeln mit Mac

Abbildung A.5: JHipster

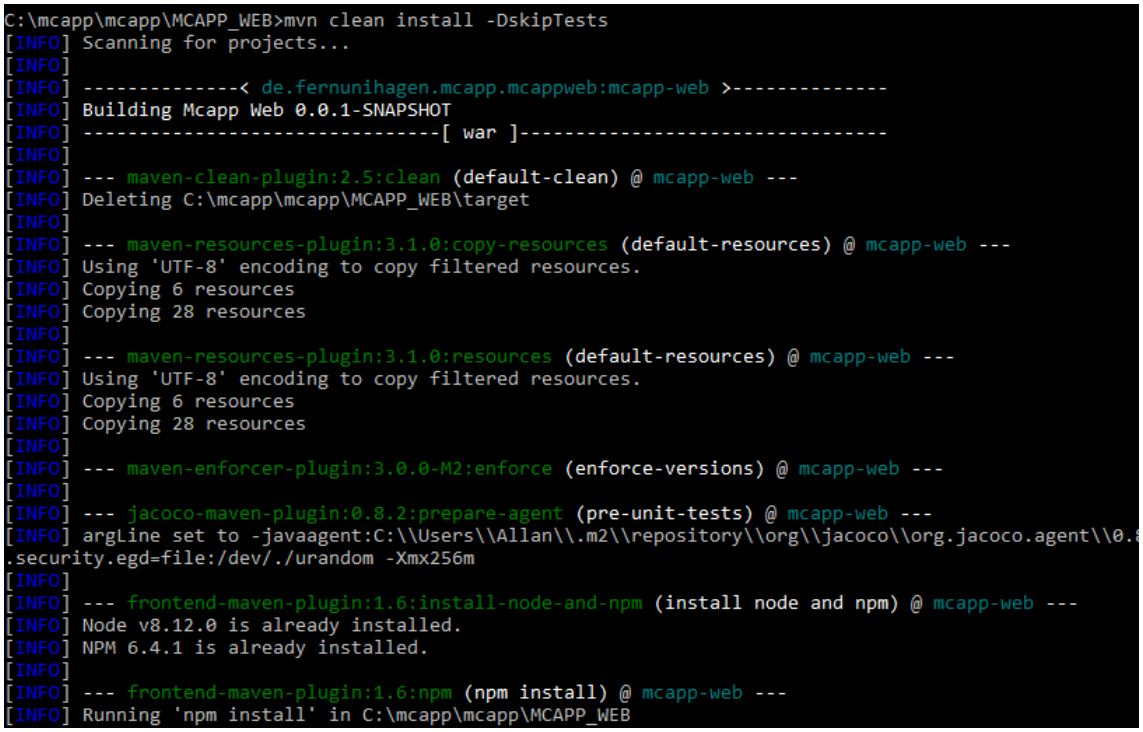


Abbildung A.6: Maven Build

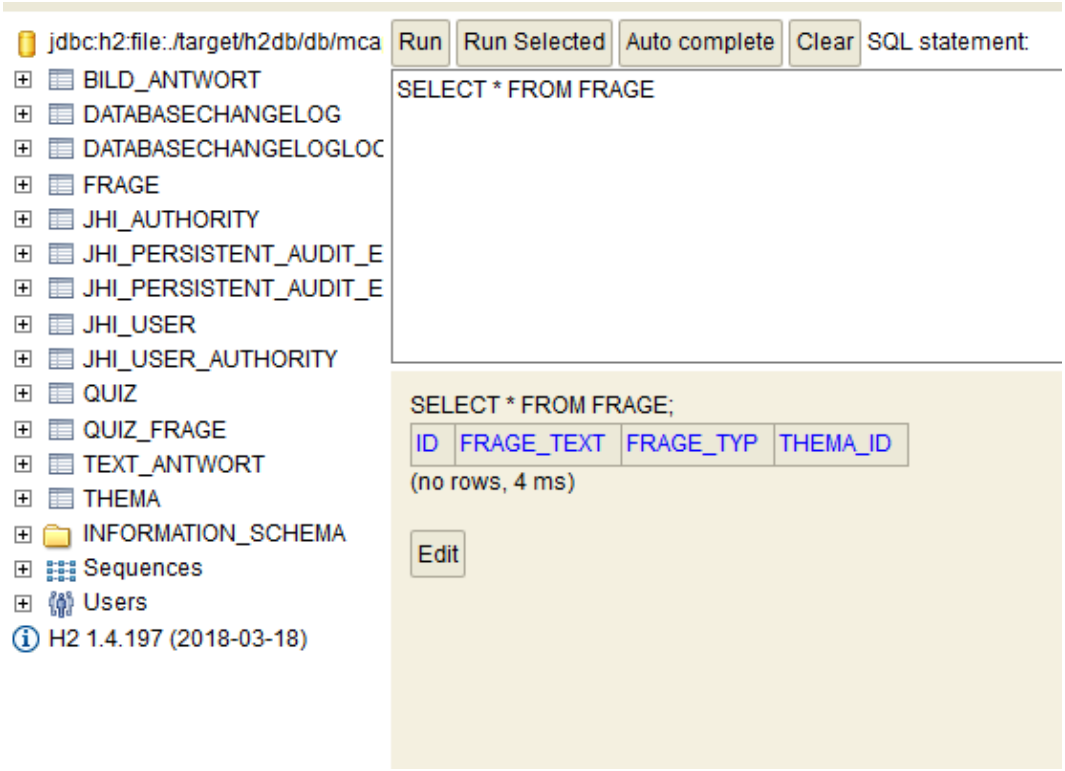


Abbildung A.7: Datenbank abfragen

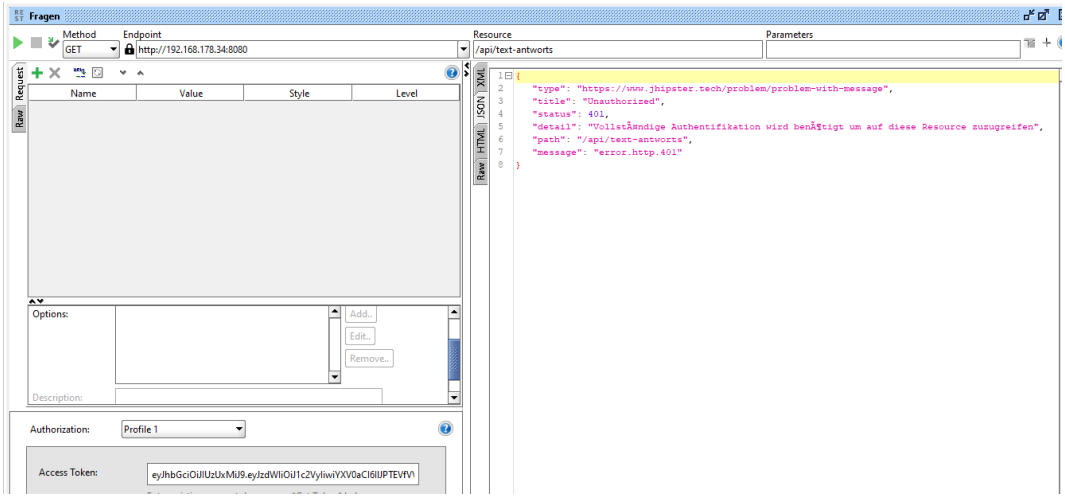


Abbildung A.8: Unauthorisierter Zugriff auf REST-Service

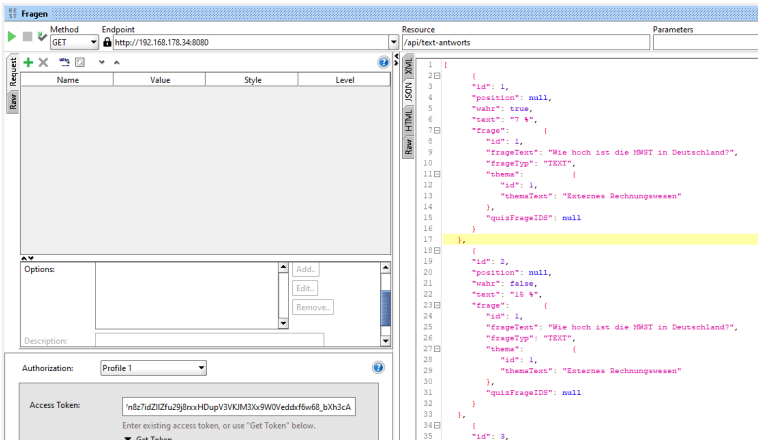


Abbildung A.9: Autorisierter Zugriff auf REST-Service

Listing A.1: SSL für Autorenzugang aktivieren

```
ssl:
key-store: keystore.p12
key-store-password: mcapptest
keyStoreType: PKCS12
keyAlias: mcapp
ciphers: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,
    TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ,
    TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 ,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 ,
    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA256,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,
    TLS_DHE_RSA_WITH_AES_256_CBC_SHA256,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,
    TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_256_GCM_SHA384,
    TLS_RSA_WITH_AES_128_CBC_SHA256,TLS_RSA_WITH_AES_256_CBC_SHA256,
    TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,
    TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA,TLS_RSA_WITH_CAMELLIA_256_CBC_SHA,
    TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA,TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
```

```
enabled-protocols: TLSv1.2
```

Listing A.2: Fragen für Thema ermitteln

```
@GetMapping("/fragesByThema/{themaid}")
@Timed
public List<Frage> GetAllFragesByThema(@PathVariable Long themaid) {
    String query = "THEMA␣=␣" + themaid;
    return StreamSupport
        .stream(frageSearchRepository.search(queryStringQuery(query)).spliterator
            (), false)
        .collect(Collectors.toList());
...
}
```

Anhang B

Entwurf der Benutzeroberflächen (Exposé)

Aus den Anforderungen wurden Ideen für Benutzeroberflächen abgeleitet und mit der Anwendung 'Balsamic Mockups 3 for Desktop' erstellt. Mit Balsamic Mockups ist es möglich, bereits in einem frühen Stadium eines Softwareentwicklungsprojektes Benutzeroberflächen zu konzipieren. [13]

B.0.1 Benutzeroberflächen für App

Die Abbildung B.1 zeigt unter (a) den Startbildschirm der MC-App. Die Anwenderin wählt eines oder mehrere Themen und betätigt den Button 'Start MC-Test' um einen Multiple-Choice-Test zu starten. Auf dem Startbildschirm kann ggfls. auch die Anzahl der Fragen eingestellt werden. Unter (b) wird die erste Frage des MC-Tests angezeigt. Der Bildschirm zeigt das Thema dieser Frage, die Anzahl '1 von X', die eigentliche Fragestellung sowie die Antwortmöglichkeiten. Es können bis zu acht Antworten angezeigt und ausgewählt werden. Über eine Schaltfläche wird zur nächsten Frage navigiert.

(a) Start (b) Spezifische Frage

Abbildung B.1: App Entwurf

(a) Bildfrage (b) generische Frage

Abbildung B.2: App Entwurf

Die Abbildung B.2 zeigt unter a) eine Frage mit Bildern als Antwortmöglichkeit. Desweiteren wird gezeigt, dass ein Zurücknavigieren zur vorherigen Frage möglich ist. Unter b) wird eine generische Frage mit sehr viel Text angezeigt. Damit die Anwenderin alle Antworten lesen kann, wird ein Scrollbalken eingeblendet. Desweiteren handelt es sich um die letzte Frage des Multiple-Choice-Tests, weshalb der Test durch Betätigung des Buttons 'Zur Auswertung' abgeschlossen werden kann.

(a) Ergebnis MC-Test (b) Antwort spezifische Frage

Abbildung B.3: App Entwurf

Die Abbildung B.3 zeigt unter (a) das Ergebnis des Multiple-Choice-Test. Dieses wird, nach Themengebiet gruppiert, angezeigt. Die Anwenderin kann nun durch Betätigung des Buttons 'Neuer Test' zum Startbildschirm zurückkehren, oder durch Betätigung des Buttons 'Lösungen' seine ausgewählten Antworten mit den Lösungen vergleichen.

Unter (b) wird das Ergebnis der erste Frage angezeigt. Dabei zeigt die erste Spalte die richtige Antwort an, während die zweite Spalte die eingegebene Antwort anzeigt.

(a) Ergebnis mit Bildern (b) Ergebnis generische Frage

Abbildung B.4: App Entwurf

Die Abbildung B.4 zeigt unter (a) die Lösung einer Frage mit Bildern. Desweiteren wird unter (b) eine generische Frage mit viel Text angezeigt. Die eingeblendeten Scrollbalken weisen darauf hin, dass der Text vollständig gelesen werden kann. Da dieser Bildschirm die letzte Frage des MC-Tests zeigt, ist hier entweder das Zurücknavigieren möglich oder die Anwenderin kann einen neuen Test starten.

B.0.2 Benutzeroberflächen für Autorenzugang

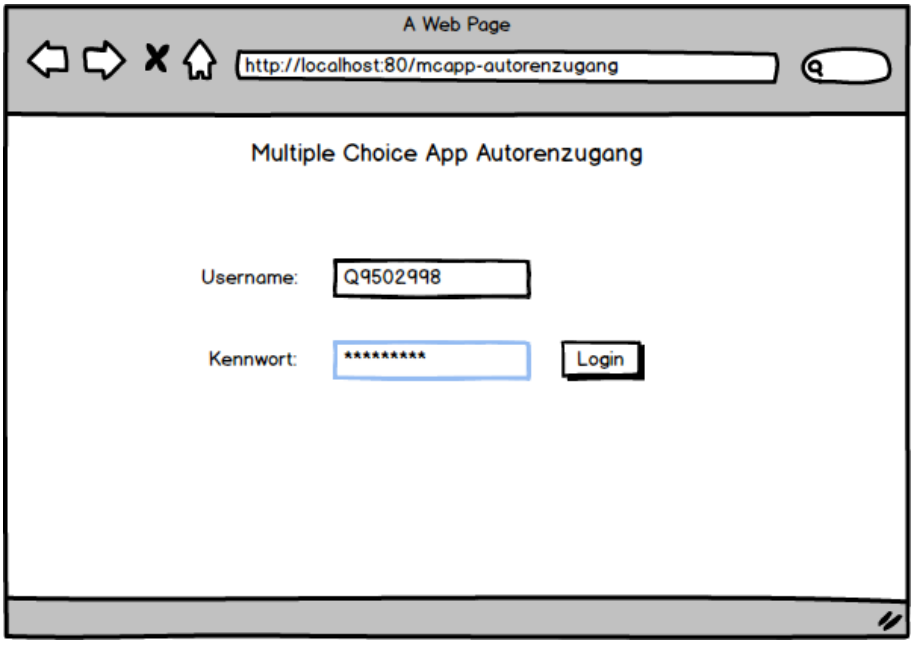


Abbildung B.5: Autorenzugang

Die Abbildung B.5 zeigt den Einstiegspunkt zum Autorenzugang, welcher über den Browser aufgerufen wird. Um den Autorenzugang vor unberechtigtem Zugriff zu schützen, muss der korrekte Benutzername mit Kennwort eingegeben werden.

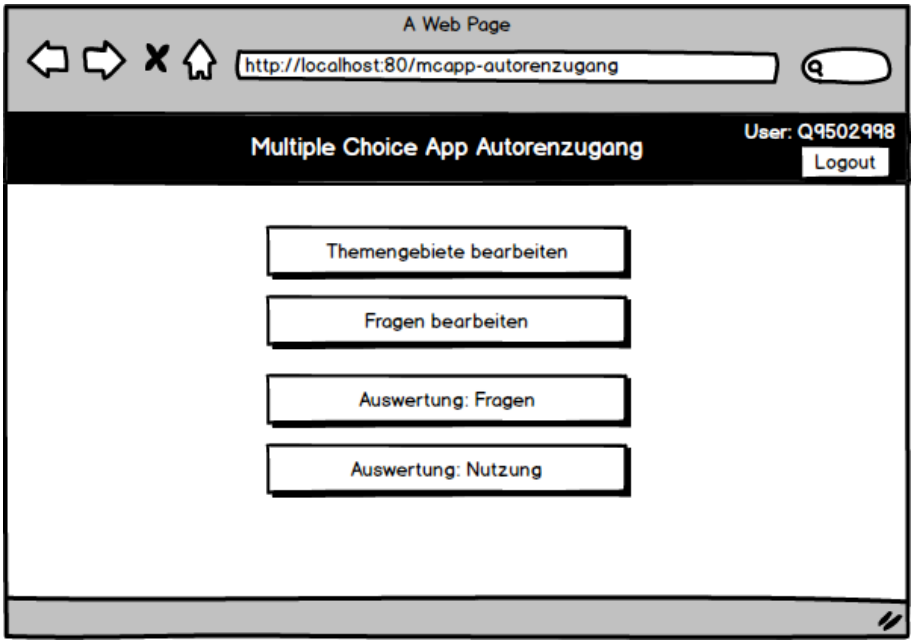


Abbildung B.6: Autorenzugang Übersicht

Die Abbildung B.6 zeigt das Menü des Autorenzugangs. Die einzelnen Bereiche des Autorenzugangs werden durch Betätigung der Schaltflächen aufgerufen.

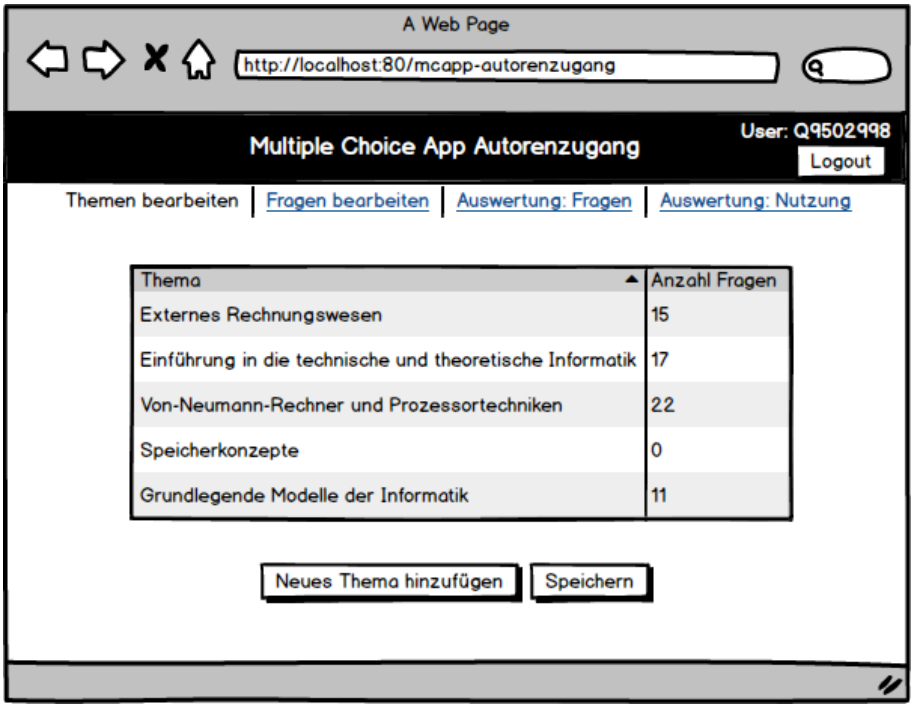


Abbildung B.7: Autorenzugang Thema

Die Abbildung B.7 zeigt die bereits vorhandenen Themen tabellarisch an. In der zweiten Spalte wird angezeigt, wieviele Fragen dem Thema zugeordnet wurden. Durch Betätigung der Schaltfläche 'Neues Thema hinzufügen' wird die Tabelle um eine neue Zeile erweitert, welche die Eingabe eines neuen Themas ermöglicht. Änderungen werden nach Betätigung über die vorhandene Schaltfläche gespeichert.

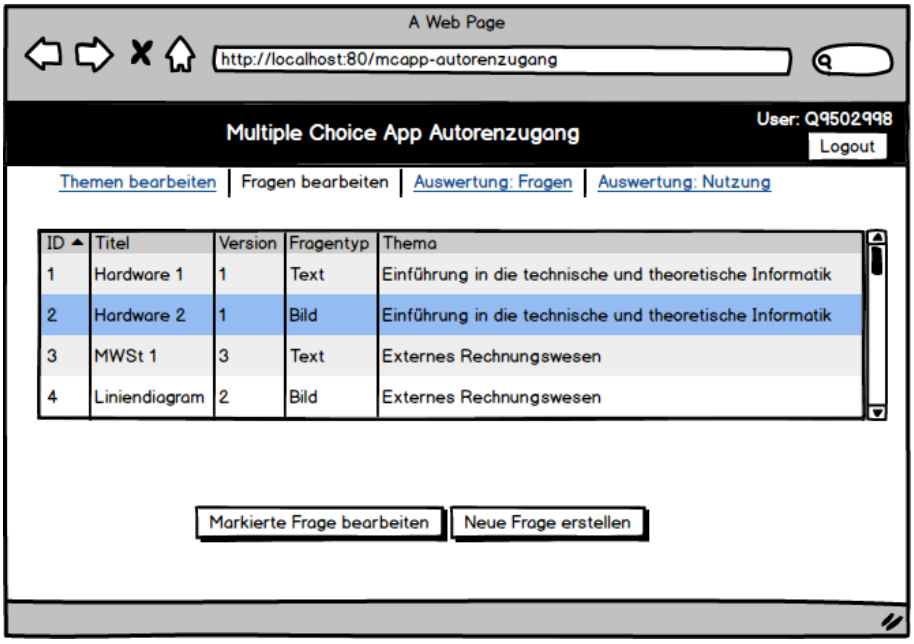


Abbildung B.8: Fragenuebersicht

Die Abbildung B.8 zeigt eine tabellarische Übersicht der bereits vorhandenen Fragen. Die Fragen haben eine eindeutige Nummer sowie einen Titel. Weiterhin zeigt die Tabelle den Fragetyp (Text oder Bild) sowie das Themengebiet. Mithilfe der Schaltflächen wird eine vorhandene Frage bearbeitet oder eine neue Frage erstellt.

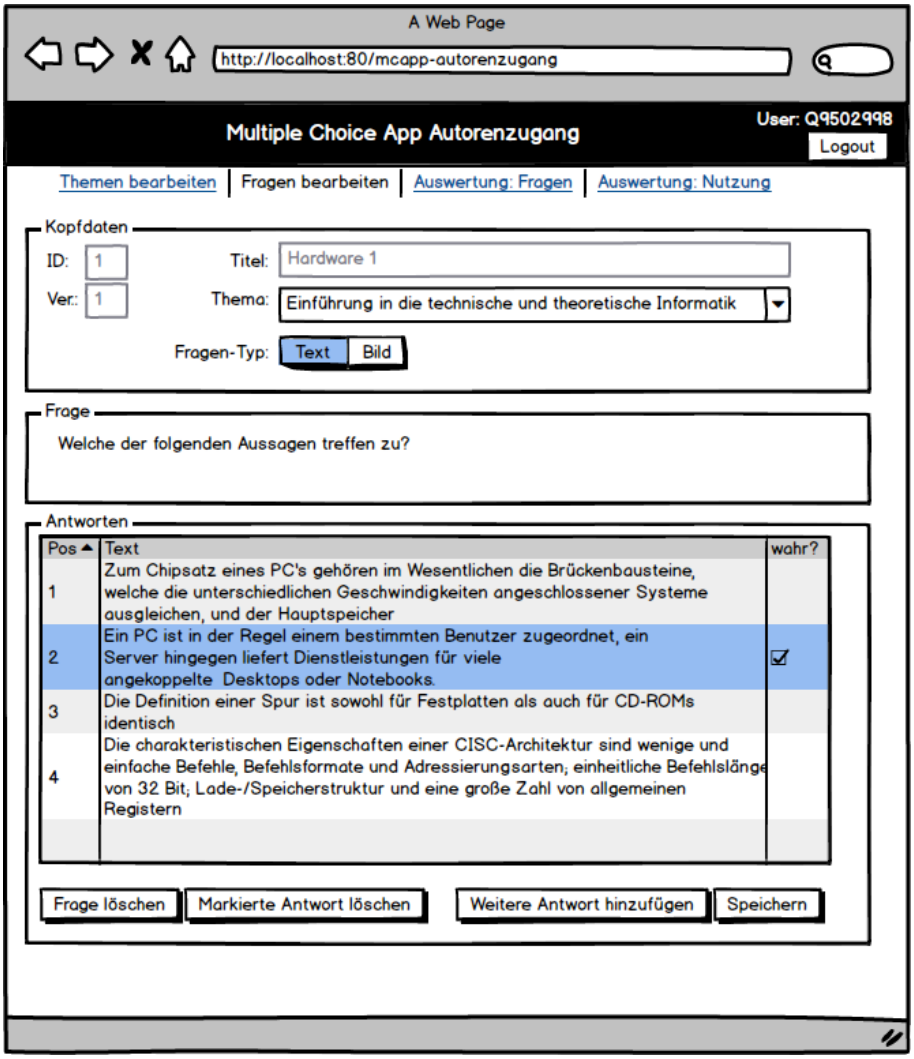


Abbildung B.9: Fragen bearbeiten

Die Abbildung B.9 zeigt den Bildschirm zur Bearbeitung einer Frage mit Texten als Antwortmöglichkeit. Im Bereich Kopfdaten dienen ID und Version als Anzeigefeld und werden vom System verwaltet. Der Titel dient dem Anwender zur Erkennung der Frage, bspw. in der zuvor beschriebenen Übersichtstabelle. Das Thema wird aus einer Auswahlliste gewählt. Desweiteren wird der Frage-Typ gewählt, da von dieser Auswahl die möglichen Antwortoptionen abhängen.

Der Bereich 'Frage' dient zur Eingabe der eigentlichen Fragestellung. Im Bereich 'Antworten' werden nun die möglichen Antworten eingegeben. Dabei dient die letzten Spalte zur Markierung wahrer Antworten. Die Schaltflächen im unteren Bereich ermöglichen das Löschen bestimmter Antwortmöglichkeiten oder das Löschen der kompletten Frage. Desweiteren können weitere Antworten hinzugefügt werden, sowie sämtliche Änderungen gespeichert werden.

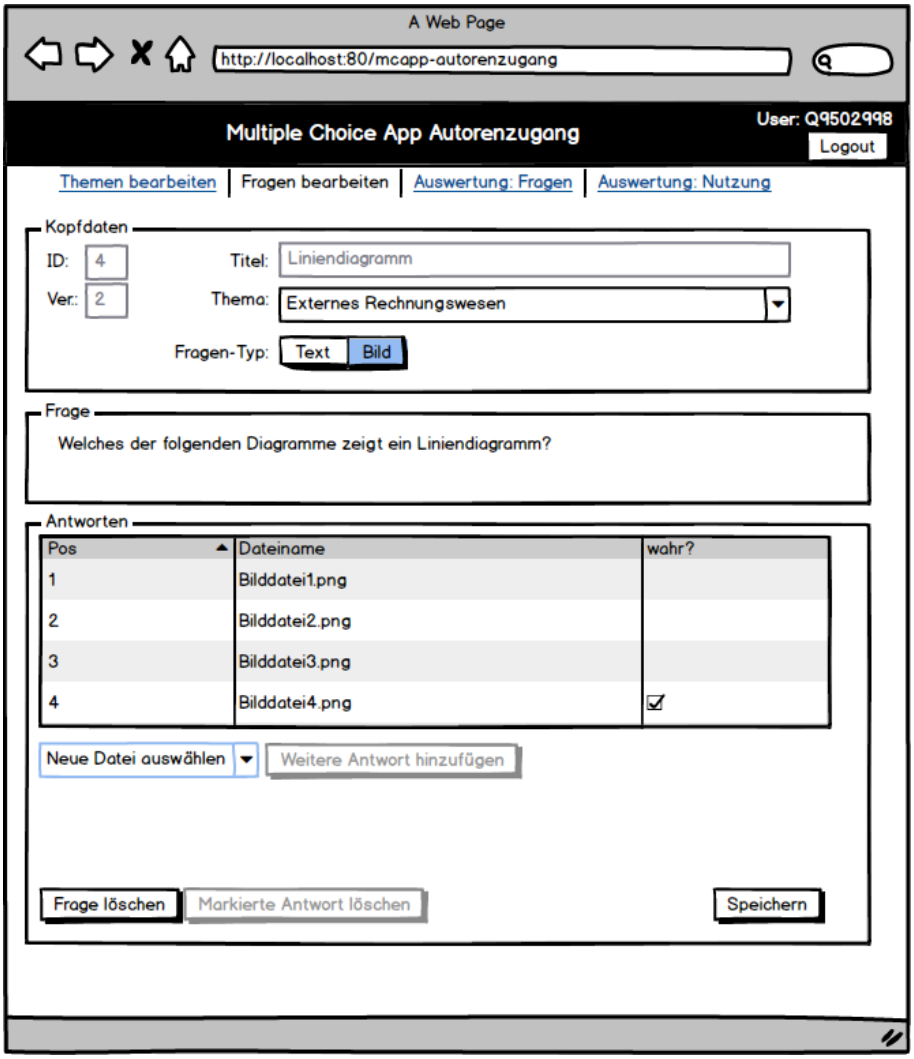


Abbildung B.10: Bildfrage bearbeiten

Die Abbildung B.10 zeigt die Bearbeitung einer Frage mit Bildern als Antwortmöglichkeit. Diese werden über eine Dateiauswahl-Komponente zur Liste der möglichen Antworten hinzugefügt.

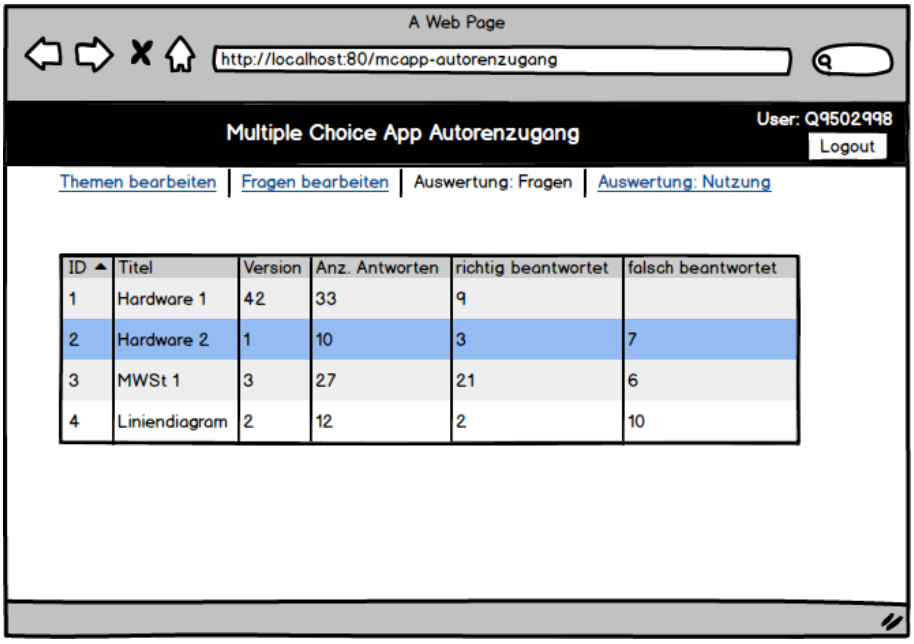


Abbildung B.11: Auswertung Fragen

Die Abbildung B.11 zeigt in tabellarischer Form, wie oft eine Frage in einem MC-Test bearbeitet wurde, sowie die Ergebnisse. Eine Zuordnung der Ergebnisse auf bestimmte Anwenderinnen ist nicht möglich.

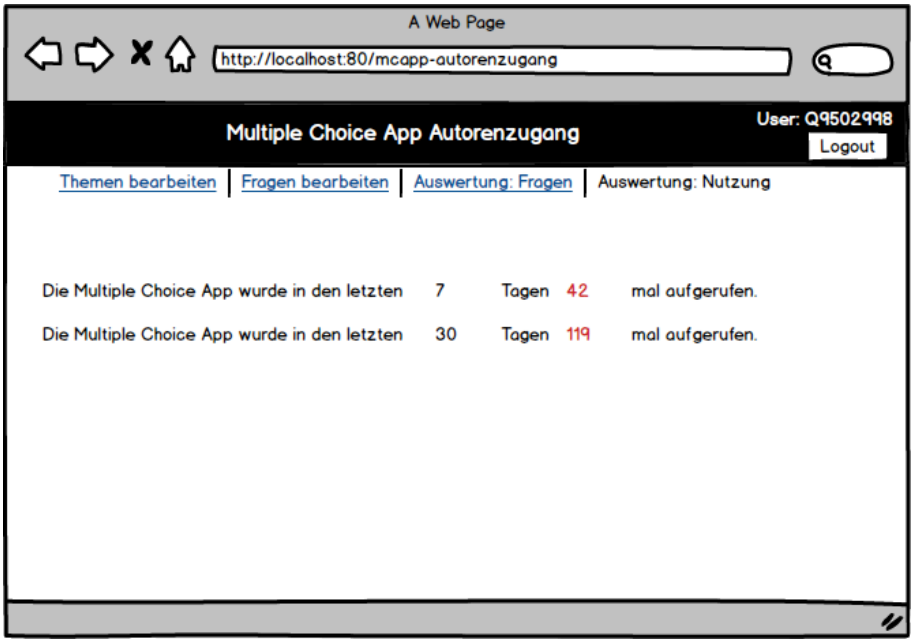


Abbildung B.12: Auswertung Nutzung

Die Abbildung B.12 zeigt an, wie oft die App in den letzten 7/30 Tagen aufgerufen wurde.