Web-basierte Hochschul-App

Modulare Web-Architektur

Bachelorarbeit

an der Hochschule für angewandte Wissenschaften Hof
Fakultät Informatik
Studiengang Informatik

Vorgelegt bei

Prof. Dr. Jürgen Heym
Alfons-Goppel-Platz 1
95028 Hof

Vorgelegt von

Dennis Brysiuk
Richard-Wagner-Straße 64
95030 Hof

Noah Christopher Lehmann
Thüringer Str. 7
95028 Hof

Hof, 03.12.2019

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

AI	Artificial Intelligence
API	Application Programming Interface
App	Application
AWS	Amazon Web Service
CRUD	Create,Read,Update und Delete
DB	Database
DDoS	Distributed Denial of Service
DevOps	Development and IT Operations
DoS	Denial of Service
DO	Data Object
ER	Entity Relationship
FTP	File Transfer Protocol
GPS	Global Positioning System
HATEOAS	Hypermedia As The Engine Of Application State
HföD AIV	Hochschule für den öffentlichen Dienst - Allgemeine Innere Verwaltung
HTTPS	Hypertext Transfer Protocol (Secure)
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
ID	Identification Number
iOS	iPhone Operation System
IoT	Internet of Things

IT	Information Technology
JMS	Java Message Service
JPA	Java Persistence API
JSON	JavaScript Object Notation
KISS	Keep It Simple and Stupid
NoSQL	nicht relational
OASIS	Organization for the Advancement of Structured Information Standards
PC	Personal Computer
PDF	Portable Document Format
PHP	Hypertext Preprocessor
REST	Representational State Transfer
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SOA	Service-oriented Architecture
SQL	Structured Query Language
SSL	Secure Sockets Layer (auch Transport Layer Security (ehemals Secure Sockets Layer (auch TLSgenannt) (SSL)) (TLS)genannt)
SSO	Single Sign On
TCP	Transmission Control Protocol
TestDaF	Test Deutsch als Fremdsprache
TLS	Transport Layer Security (ehemals SSL)

TO	Transfer Object
UI	User Interface
UNIcert	UNIcert-Zertifikatssystem
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UX	User Experience
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language

1 Einleitung

Die Digitalisierung ist im zweiten Jahrzehnt des 21. Jahrhunderts kaum noch aufzuhalten. Was früher aufwändig manuell gesammelt und dargestellt wurde, hält heute Einzug in den Smartphones der Menschen. So wird zur Navigation keine Karte mehr benötigt, lediglich ein Global Positioning System (GPS)-fähiges Handy. Zum Einkaufen muss man sich keinen Merkzettel mehr schreiben, sondern dem Smart-Assistenten diktieren, was man braucht. Dieser speichert alles automatisiert auf dem Smartphone, griffbereit sobald man im Laden angekommen ist. Die Nachrichten liest man nicht mehr in der Zeitung, sondern im News-Feed. Das Wetter muss man nicht mehr in den Nachrichten aufschnappen, viel einfacher ist die minutengenaue Aktualisierung der Daten auf dem digitalen Begleiter.

Dieser Trend hat sich in den vergangenen Jahren auch in den Lehrinstituten durchgesetzt. Somit wurden im Zuge der Digitalisierung die Mechanismen neu überdacht und durch modernere Alternativen ersetzt. Was früher mit Kreide an die Tafeln geschrieben wurde, wird heute durch den Beamer an die Wand projiziert. Die Zettel, die einst vielfach gedruckt und verteilt werden mussten, werden durch Lernplattformen wie Moodle und Chamilo ersetzt. Dort können die Dokumente einfach hochgeladen und verteilt werden. Die Informationen, die man damals noch im Sekretariat erfragen musste, werden heute auf den jeweiligen Internet Seiten dargestellt. Und so steigen die Schulen und Hochschulen auch langsam auf die Digitalisierung im Bereich des Stundenplans um. Man hat sich an die Bedürfnisse der Studierenden angepasst. Im Laufe der letzten beiden Jahrzehnte wurden für weite Teile der Organisation der Institutionen digitale Helfer eingeführt. Diese reichen von Datenbanken zum Speichern von Zeitplänen und anderem bis zu Applications (Apps) zum erleichterten Anzeigen von wichtigen Informationen wie Stundenplan Daten.

1.1 Beweggründe

So wurde auch an der Hochschule Hof - im Zuge dieser Bewegung - die Bereitstellung der Stundenplaninformationen für die Studierenden weitgehend erleichtert. Bereits der Zugang des Stundenplans über die Website war ein großer Fortschritt, doch wurde auf Antrieb der Fakultät Informatik sogar die Entwicklung nativer Anwendungen für das Smartphone angestoßen. Diese ermöglichten es den Studierenden die nötigen Zeiten und Veranstaltungsorte ihrer Vorlesungen jederzeit über das Smartphone abzurufen. Des weiteren wurden die Funktionalitäten dieser Apps über die Jahre so erweitert, dass sie bis zuletzt zusätzliche Funktionen wie Speiseplan Informationen, Raumsuche und Kalendereinbindung anbieten konnten. Durch diese Entwicklungen konnten bis zum Jahr 2019 mehr als 90% der Studierenden der Hochschule Hof dazu bewegt werden, eine der angebotenen Smartphone Anwendungen zu nutzen. Lediglich 7% nutzen immer noch die Website zum Suchen ihrer Veranstaltungen[1].

Allerdings gibt es bei den Nutzungsstatistiken der hochschuleigenen Apps noch große Unterschiede zwischen den einzelnen Nutzergruppen, speziell den internationalen Studierenden und denen, die dauerhaft an der Hochschule eingeschrieben sind. So nutzen lediglich 75% der internationalen Studierenden eine der Anwendungen der Hochschule, der Rest bedient sich bei den Informationen bei der Webseite. Zudem berichten auch nur 30% der Internationalen Studierenden, dass sie ohne größeren Aufwand einen vollständigen Stundenplan einrichten können. Des weiteren wird angemerkt, dass viele Informationen in den Hochschul-Apps nur in deutscher Sprache aufgeführt werden, was es anderssprachigen Studierenden deutlich schwerer macht, die Inhalte korrekt zu interpretieren. Doch auch die deutschsprachigen Studierenden haben im Rahmen der Umfrage knapp fünfzig Verbesserungsvorschläge erbracht und auch klar gemacht, dass eine verbesserte Version der aktuellen Apps deutlich öfter genutzt werden würde.

Zusätzlich zum Funktionsumfang der bereits vorhandenen Anwendungen sind die Entwicklungsumstände dieser Apps im aktuellen Zustand zu bemängeln. Eines der Probleme liegt darin, dass die Hochschule für drei Betriebssysteme jeweils eine eigene App anbietet. So ergibt sich die dreifache Entwicklungsarbeit, denn es muss jeweils eine Anwendung für Android, iPhone Operation System (iOS) und Windows gepflegt werden. In einigen Bereichen der Entwicklung hat der Einsatz in der

[1] Siehe Kapitel 2.1

vergangenen Zeit stark nachgelassen, zumal die Hauptentwickler der stark genutzten Android App in den vergangenen Semestern die Hochschule verlassen haben.

1.2 Zielsetzung

Aus den genannten Problemen der Hochschul-Apps hat sich das Thema dieser Bachelorarbeit geformt. Diese wird sich um die Entwicklung einer Lösung für diese Schwierigkeiten kümmern.

Anhand von Umfragen, die an die Nutzer der Hochschul-Apps gerichtet sind, soll analysiert werden, welche Aspekte einer Überarbeitung bedürfen und welche aus Vorhandenem erfüllt werden können. Des weiteren werden die aktuellen Apps der Hochschule Hof kritisch betrachtet. Mithilfe des International Office und des Sprachenzentrums, sowie des Auftraggebers Prof. Dr. Heym werden dann die Anforderungen an eine neue Anwendung zur Darstellung von Stundenplan Daten, Terminänderungen, Mensa-Speiseplänen und möglichen weiteren Informationen gesammelt. Diese funktionalen Anforderungen werden dann auf ihre Machbarkeit geprüft und schlussendlich schrittweise in einen Prototyp für eine plattformübergreifende Hochschul-App übernommen. Die genauen Funktionen dieser App werden aufgeführt und erläutert, worauf die Vorgehensweise für die Implementierung dieser dargestellt wird. Der Fokus dabei liegt auf den angewandten Techniken und Frameworks, welche mit ihren Alternativen verglichen werden, wobei der Entscheidungsvorgang zu den jeweiligen Techniken erarbeitet wird. Daraufhin soll die Anwendung als Ganzes betrachtet werden, was eine genauere Erschließung der Architektur erfordert, auf die diese aufbaut. Abschließend soll dann betrachtet werden, welche Funktionen im Rahmen dieser Arbeit implementiert werden können und wie die App in Zukunft noch weiter entwickelt werden kann.

1.3 Zielgruppe

Im allgemeinen behandelt diese Arbeit die Entwicklungsphase und die Vorbereitung einer neuen und verbesserten, plattformunabhängigen Hochschul-App. In den ersten Kapiteln werden also die Rahmenbedingungen für die Entwicklung gesetzt. Es werden unter anderem die bereits vorhandenen Anwendungen der Hochschule analysiert und die Zufriedenheit der Nutzer dieser Apps genauer betrachtet. Danach werden anhand der gesammelten Ergebnisse, aber auch durch gegebene Prämissen,

die allgemeinen funktionalen Anforderungen aufgeführt und erläutert. Dieser Teil der Bachelorarbeit ist für eine breite Gruppe an Lesern interessant. Anhand der Umfrage zur Zufriedenheit können beispielsweise die Entwickler der vorhandenen Apps ihre Schlüsse ziehen und die eigenen Projekte verbessern, durch die Sammlung der Anforderungen hingegen wird das Projekt und dessen Umfang genau definiert, was besonders für den Auftraggeber und die beteiligten Entwickler von Bedeutung ist.

Der zweite Teil der Arbeit geht dann genauer auf die Umsetzung des Projektes ein, es wird erklärt, in welche Teile das Projekt aufgeteilt wird und warum diese Teilung vorgenommen wird. Anhand dieser Teilung wird dann das Rückgrat des Projektes, die Datenquelle entworfen. Mit diesem Teil können die Entwickler des Projektes genau einsehen, warum Entscheidungen zu den genutzten Techniken getroffen wurden und was das für das gesamte Projekt bedeutet. Für spätere Arbeiten an dem Projekt ist der zweite Teil der Arbeit auch ein guter Einstieg in die Thematik und bildet eine gute Grundlage zu dem benötigten Wissen über die genutzten Design Patterns.

Im letzten Teil der Arbeit wird dann eine konkrete Lösung des Problems erarbeitet. Dabei werden alle relevanten Architekturentscheidungen getroffen und erläutert, sowie die nötigen Schnittstellen definiert. Dieser Teil der Bachelorarbeit bietet die Grundlage für die Entwicklung des eigentlichen Prototypen. Entwickler in allen Stadien des Projektes können sich durch diese Kapitel einlesen und die Grundstruktur des Projektes verstehen. Genaueres zur Vorgehensweise und den benutzten Techniken findet man allerdings in der zu dieser Bachelorarbeit gehörigen Praxisarbeit, die im Kapitel 1.5 nochmals erwähnt wird.

1.4 Vorausgesetztes Wissen

Im allgemeinen ist diese Bachelorarbeit eine Arbeit aus dem Bereich der Informatik, dementsprechend ist ein solides Grundwissen im Bereich des Informationsmanagements, der Programmierung und der Datenspeicherung von Vorteil. Grundlegende Konzepte des Software Entwurfs sollten dem Leser ebenfalls bekannt sein. Des weiteren analysiert diese Arbeit mehrere Ideen aus dem Bereich der Webentwicklung, wobei dort eher allgemeine Konzepte vorgestellt werden. Genauere Umsetzungen werden in den darauf folgenden Kapiteln genauer beleuchtet und erläutert. Basie-

rend auf den Webtechnologien, die angeschnitten werden, sollten dem Leser die gängigsten Datenformate und Kommunikationsprotokolle aus dem Webbereich ein Begriff sein. Wo fundiertes Wissen voraus gesetzt wird, wird ebenfalls eine kurze Zusammenfassung des benötigten Wissens geliefert.

Trotz der technischen Aspekte dieser Arbeit kann ein großer Teil auch von Lesern verstanden werden, die kein Wissen aus der Informatik oder speziell aus den oben genannten Bereichen vorweisen können. Die ersten Kapitel befassen sich, wie bereits erwähnt, mit der Analyse der vorhandenen Anwendungen der Hochschule Hof, sowie mit der Zufriedenheit der Nutzer dieser Anwendungen. Daraufhin werden die funktionalen Anforderungen gesammelt - auch dazu wird kein fundiertes Wissen aus der Informatik benötigt. Der zweite Teil der Arbeit befasst sich dann mit der Analyse von verschiedenen Konzepten, wo ein fundiertes Grundwissen von Vorteil ist, aber nicht zwingend notwendig. Auch hier werden viele Erklärungen geliefert, um dem Leser eine gute Grundlage für das Verständnis der letzten Kapitel zu geben. In diesen werden dann genaue Konzepte und Lösungsansätze erarbeitet, die überwiegend technisch versiert sind. Spätestens dort werden, wie oben erwähnt, die fundierten Grundlagen im Bereich Informatik benötigt, um die erarbeiteten Lösungen vollständig nachvollziehen zu können.

1.5 Vorgeschlagene Nebenlektüre

Um ein bestmögliches Verständnis dieser Arbeit zu erlangen, ist es ratsam einige andere Quellen und Arbeiten zum Thema der Entwicklung der Hochschul Anwendung und zu anderen technischen Themen zu lesen.
Im Rahmen des Projektes der Neuentwicklung einer Hochschul-App werden mehrere Arbeiten verfasst, welche sowohl die Anforderungen analysieren, als auch die Vorgehensweise beschreiben. Eine dieser Arbeiten ist eine weitere Bachelorarbeit, welche sich mit der Anwenderverwaltung, Autorisierung und mit der grafischen Oberfläche des Prototypen befasst.[2] Dort werden einige Teile der Gesamtarbeit analysiert, welche zur Vollständigkeit dieser Arbeit und um den Lesefluss zu erhalten, in dieser Bachelorarbeit nur kurz beschrieben und referenziert werden.

Des weiteren werden zu beiden Bachelorarbeiten, sowohl zu dieser, als auch zur

[2] Glaser (2019a).

zweiten innerhalb dieses Projektes,[3] Praxisarbeiten angefertigt, welche umfangreiche Dokumentationen zum Aufbau und zur Umsetzung der Implementierung des Prototypen der Hochschul-App beinhalten. Dort werden alle Frameworks und technischen Komponenten näher erläutert. Des weiteren wird dort die Vorgehensweise bei der Implementierung beleuchtet, welche näheren Einblick auf die Entwicklung geben soll, sodass der Umfang und die umgesetzten - beziehungsweise die ausgelassenen - Funktionen und Komponenten des Prototypen gerechtfertigt werden können.[4]

Des weiteren finden sich zu dem Thema *Modulare Web-Architektur* einige gute Bücher, die den Einstieg deutlich vereinfachen und die im Rahmen dieser Arbeit auch referenziert werden. Dazu gehört beispielsweise der Titel *Grundlagen des modularen Softwareentwurfs*[5] von Herbert Dowalil, welcher ausführlich auf die Konzepte *Service-orientierte Architektur* und *Microservices* eingeht. Wer jedoch einen allgemeineren Blick auf die Konzepte der Web-Technologien erlangen möchte, die in dieser Arbeit analysiert werden, dem wird das Buch *Verteilte Systeme*[6] eine bessere Einsicht verschaffen. Weitere Bücher zu diesem und zu den in dieser Arbeit relevanten Themen sind dem Literaturverzeichnis zu entnehmen.

[3] Glaser (2019a).
[4] Brysiuk; Lehmann (2019b); Glaser (2019b).
[5] Dowalil (2018).
[6] Schill; Springer (2012).

2 Auswertung vorhandener Anwendungen

Um eine fundierte Grundlage für die Entwicklung einer allgemein akzeptierten Hochschul-App zu schaffen, muss erst analysiert werden, welche Alternativen die Hochschule Hof bereits anbietet und wie zufrieden die Nutzer dieser Anwendungen mit ihnen sind. Im folgenden wird demnach erst eine Zufriedenheitsumfrage untersucht, worauf die vorhandenen Anwendungen analysiert werden, bei denen dann gezielt auf die in der Umfrage gefundenen Mängelpunkte geachtet werden kann.

2.1 Allgemeine Auswertung der Zufriedenheit

Im Rahmen der Analyse der vorhandenen Anwendungen der Hochschule Hof wurde eine Befragung der Studierenden durchgeführt. Diese richtete sich an zwei verschiedene Nutzergruppen und wurde demnach ebenfalls in zwei unterschiedlichen Fassungen durchgeführt, welche später genauer beleuchtet werden. Das Ziel dieser Umfrage ist die Erfassung des allgemeinen Zufriedenheitsstands der Nutzergruppen mit dem aktuellen Angebot an Anwendungen der Hochschule Hof, welche den Studenten Stundenplaninformationen und möglicherweise noch weiteres liefern.

2.1.1 Zielgruppen

Die Befragung der Studierenden richtet sich an zwei Benutzergruppen der aktuellen Stundenplananwendungen der Hochschule Hof. Die deutlich größere der beiden Nutzergruppen, auf die auch das Hauptaugenmerk in der Entwicklungsphase der neuen Hochschul-App liegt, sind die deutschsprachigen Studierenden der Hochschule Hof, welche dauerhaft als Studenten eingetragen sind. Diese Teilnehmergruppe der Umfrage besteht aus 225 anonymen Studierenden. Diese Nutzergruppe stellt den Regelfall der Nutzung der App dar. Die Studierenden sind alle immatrikuliert, belegen genau einen Studiengang und sind in einer Fakultät eingeschrieben. Der Regelstudent hat somit in den Grundlagen- und Kernbereichen seines Studiums

einen festen Stundenplan, welcher sich nach dem Fachsemester dessen richtet. Dies macht die Entwicklung einer verbesserten Hochschul-App deutlich einfacher, da viele Faktoren vorgegeben sind und kaum Spezialfälle abgedeckt werden müssen. Mögliche Abweichungen sind hier lediglich Langzeitstudierende und Zwischensemester.

Die kleinere der beiden befragten Nutzergruppen bilden die internationalen Studierenden der Hochschule Hof, welche sehr wahrscheinlich eine andere Sprache als Deutsch als Muttersprache nutzen und nur zeitlich begrenzt an der Hochschule Hof eingeschrieben sind. Die Gruppe der Befragten umfasst 31 anonyme Studierende. Diese Studierenden kommen oft aus Hochschulen, deren Organisation sich von der der Hochschule Hof deutlich unterscheidet. Somit ist eine Aufteilung dieser Nutzer in die drei Fakultäten der Hochschule deutlich schwieriger. Auch die Auswahl der Vorlesungen aus den Studiengängen unterscheidet sich in der Regel vom beschriebenen Normalfall, somit belegen viele Studierende aus dem Ausland oder aus anderen Hochschulen oft Vorlesungen aus verschiedenen Fakultäten und unterschiedlichen Studiengängen und Fachsemestern. Dies macht die Erstellung von passenden Stundenplänen für die betroffenen Internationalen deutlich komplexer und bedarf deshalb eine genaue Betrachtung der Zufriedenheit dieser Studierenden mit dem aktuellen Angebot an Anwendungen der Hochschule Hof, um deren Bedürfnisse besser zu erkennen.

2.1.2 Ergebnisse

Aus den Befragungen ergab sich, dass 92% der regulären Studierenden eine der drei Smartphone Anwendungen der Hochschule nutzen. Bei den internationalen Teilnehmern lag diese Zahl lediglich bei etwa 74%. Die Nutzungshäufigkeit der Anwendungen ergab bei beiden Gruppen, dass mehr als drei viertel der Befragten die App mindestens zwei bis drei mal wöchentlich nutzen. Aus den Antworten der internationalen Studierenden der Hochschule Hof ergibt sich ebenfalls, dass nur etwa 61% alle studienrelevanten Informationen aus den Anwendungen der Hochschule beziehen können. Zudem sagen nur etwa 29%, dass es ihnen leicht fällt, einen vollständigen Stundenplan zu erstellen. Diese Erkenntnisse lassen darauf schließen, dass die deutschsprachigen Studierenden der Hochschule deutlich besser mit dem aktuellen Angebot der Hochschule umgehen können, während sich internationale Studierende eher schwer tun. Dies kann man darauf zurückführen, dass die Anwendungen alle in deutscher Sprache implementiert wurden. Zudem bauen die vorhan-

denen Anwendungen darauf auf, dass das Studiensemester und der Studiengang des Nutzers bekannt sind, damit der Stundenplan den Angaben entsprechend erstellt werden kann. Bei internationalen Studierenden ist dies aus dem oben genannten Problemen der betroffenen Befragten nicht möglich. Auch das International Office hat zu diesem Punkt angegeben, dass das Feedback der Internationalen Studierenden in dieser Hinsicht eher negativ ausfällt. Wer sich international präsentieren will, sollte seine Anwendungen nicht nur auf Deutsch anbieten[1].

Eine weiterentwickelte und den Nutzern angepasste App findet in beiden Gruppen allgemeinen Zuspruch. Hierbei sagen mehr als 95% der internationalen Studierenden, dass sie eine ihnen angepasste Anwendung häufiger nutzen würden. Bei den regulären Studierenden der Hochschule Hof liegt dieser Wert bei immerhin 76%. Dies kann darauf zurückzuführen sein, dass die Internationalen Studierenden eine zuverlässige Informationsquelle, die alle relevanten Daten enthält, deutlich bevorzugen würden. Diese Annahme kann durch die Aussagen des International Office bezüglich der Informationsführung und der Stundenplanzusammensetzung der internationalen Studierenden bekräftigt werden[2]. Des weiteren liegen Anhand der Befragung einige Aspekte der vorhandenen Anwendungen vor, welche nach Meinung der Befragten einer Verbesserung bedürfen. Ein Punkt, welcher häufiger angemerkt wurde, ist die uneinheitliche Darstellung der Daten zwischen den einzelnen Varianten der Hochschulanwendungen. Demnach liegen in der Android Version der Hochschul-App sowohl Stundenplan-, als auch Speiseplaninformationen vor. Das ist in der iOS-App nicht der Fall und wird von deren Nutzern auch deutlich kritisiert. Zudem wird auch der Zwang kritisiert, die nativen Apps über die offiziellen Quellen der Plattformhersteller zu beziehen. Demnach würden etwa 71% der internationalen Nutzer und 55% der regulären Studierenden eine plattformübergreifende Hochschulanwendung bevorzugen.

2.1.3 Fazit

Abschließend kann man Anhand der Umfrage erkennen, dass die Entwicklung einer neuen Lösung zur Informationsdarstellung der Stundenpläne und anderer Daten von den Nutzern der aktuellen Anwendungen unterstützt wird. Allen voraus wird eine Anwendung gefordert, die zuverlässig ist, Stundenplandaten mit anderen hilfreichen Informationen verbindet und die unabhängig von der genutzten Plattform

[1] Siehe Kapitel 3.2
[2] ebd.

ist. Aktuelle Anwendungen der Hochschule werden zwar momentan genutzt, die Zufriedenheit der Nutzer ist allerdings eher gering.

2.2 Anwendungen

Zur Entwicklung eines Prototypen für eine Anwendung, welche ein Nachfolger bereits vorhandener Produkte sein soll, gehört es auch, diese Produkte vorab zu analysieren und deren Stärken und Schwächen zu identifizieren. Dies wird in den folgenden Abschnitten genauer betrachtet. Um den einzelnen Anwendungen genauere Gewichtungen zuzuschreiben, ist es wichtig, die Nutzer vorab zu befragen, welche der Anwendungen sie bevorzugt benutzen.

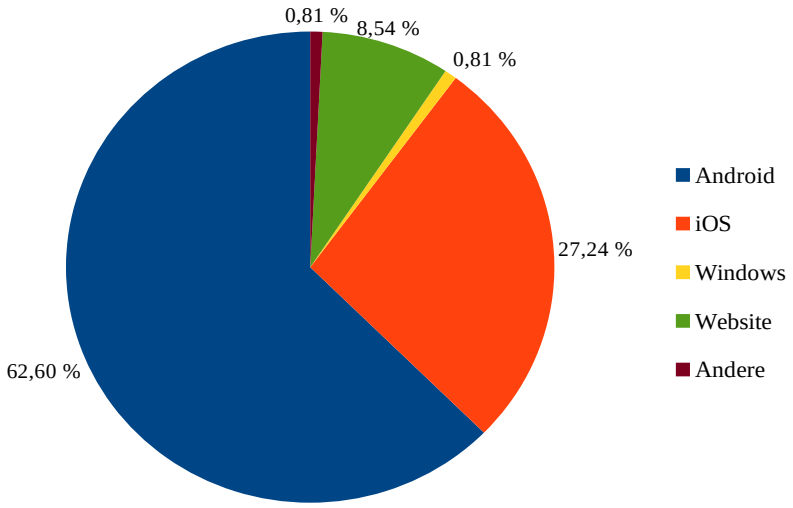


Abbildung 2.1: Umfragewerte zur Nutzung der Apps nach Plattform[3]

Wie man Anhand Abbildung 2.1 erkennt, wird die Android App der Hochschule Hof von 63% der Studierenden mit Abstand am meisten genutzt, gefolgt von der iOS-App mit etwa 29% aller Nutzer. Als letzte große Informationsquelle wird die Hochschul Website von etwa 6% aller befragten Studierenden genutzt. Die Windows App und andere Alternativen werden kaum genutzt.

[3] Brysiuk, Lehmann (2019)

2.2.1 Android

Die Android App der Hochschule Hof erfreut sich in den vergangenen Jahren großer Beliebtheit bei den Studierenden. Wie bereits gezeigt wurde, ist sie die meist genutzte App aller Alternativen aus dem Hochschul Angebot. Neben den Grundfunktionen eines Stundenplans, eines personalisierten Stundenplans und dem Anzeigen der Stundenplanänderungen besitzt die App im Vergleich zu den anderen nativen Anwendungen für iOS und Windows die meisten zusätzlichen Funktionen. Unter anderem wurde eine Webview des Primuss Portal eingebaut, welche sich nach einem einmaligen Anmelden auch die Anmeldedaten des Nutzers merkt, was mehrfaches Einloggen unnötig macht. Ebenfalls durch eine Webview integriert wurde eine Navigation- und Fahrplanfunktion. Durch diese kann man sich Fahrzeiten und Verbindungen aller öffentlichen Verkehrsmittel anzeigen lassen. Weitere nützliche Funktionen sind die Raumsuche, *Wo bin ich?* und die Chatfunktion für jede Vorlesung.

Die Android App der Hochschule Hof besitzt ebenfalls eine Schnittstelle, mit der sie Speiseplaninformationen aus dem Angebot des Studentenwerk Oberfrankens anzeigen kann. In dieser kann man seine Vorlieben speichern und sich nur die Speisen anzeigen lassen, welche für den Nutzer selbst relevant sind. Die Funktion ist besonders hervorzuheben, da nicht alle Anwendungen der Hochschule Hof das Menü des Studentenwerks bereitstellen.

2.2.2 iOS

Mit knapp 29% der befragten Nutzer liegt die iOS Anwendung der Hochschule Hof an zweiter Stelle im Nutzungsranking der Umfrageteilnehmer. Das korreliert auch mit den allgemeinen Marktanteilen der Smartphone Betriebssysteme, welche in Abbildung 2.2 dargestellt werden.[4] Auch die Grundlegenden Funktionen des Stundenplans sind wie bei der Android App vorhanden. Zusätzlich dazu hat der Nutzer die Möglichkeit, die offiziellen Termine der Hochschule Hof einzusehen, sowie eine eigene Aufgabenliste anzulegen und zu pflegen. Als kleinen Zusatz hat die iOS-App auch ein sogenanntes Widget. Das bedeutet, die Anwendung kann Daten auf dem Bildschirm des Smartphones anzeigen, ohne, dass die App geöffnet ist.

Zu bemängeln ist, dass die Nutzer dieser Anwendung keinen direkten Zugriff auf

[4] Vgl. Kantar (2019c).

den Speiseplan der Mensa der Hochschule für den öffentlichen Dienst - Allgemeine Innere Verwaltung (HföD AIV) haben. Stattdessen haben befragte Nutzer angegeben, dass sie den Speiseplan über die offizielle Website des Studentenwerk Oberfrankens beziehen, sich aber eine Einbindung in die App wünschen.[5]

2.2.3 Windows App

Die mit Abstand am wenigsten genutzte Anwendung der Hochschule Hof ist die Windows App. Nur ein Prozent der befragten Studierenden nutzen demnach diese App. Umgerechnet sind das lediglich 2 Nutzer. Trotz der geringen Nutzung bietet die Windows Anwendung ein sauberes Interface, über welches der Nutzer die üblichen Informationen wie die anstehenden Vorlesungen einsehen kann. Auch hier hat der Anwender die Möglichkeit, seinen Stundenplan zu personalisieren. Weitere nützliche Funktionen sucht man hingegen vergeblich. Im Gegensatz zur iOS-App bietet die Windows App dennoch die aktuellen Speiseplan Informationen an.

2.2.4 Website

Als Alternative zur Nutzung der nativen, plattformabhängigen und unabhängig entwickelten Anwendungen der Hochschule Hof soll hier nochmals die Website als Informationsquelle für Stundenplaninformationen und weitere nützliche Daten betrachtet werden. Die Nutzergruppe, die keine der Smartphone Apps nutzt macht immerhin fast 6% aus und ist damit deutlich größer als die der Windows User.

Stundenplan Informationen können nach einer einfachen Auswahl des Studiengangs und des Fachsemesters eingesehen werden. Weitere Verfeinerungen der Vorlesungen bietet die Art der Veranstaltung, welche die Trefferliste eingrenzt. Zusätzlich bietet die Seite die Möglichkeit an, den Stundenplan als Portable Document Format (PDF) herunterzuladen und zu drucken. Ähnlich werden auch die Stundenplanänderungen angezeigt. Nach Auswahl des Studiengangs und des Fachsemesters werden dem Nutzer alle anstehenden Änderungen angezeigt. Zusätzlich kann der Nutzer auch ein Datum angeben, um die Ergebnisse zu filtern.

Um den Mensa Speiseplan des Studentenwerk Oberfrankens mit einzubinden, wurde der Website ein Verweis auf die offizielle Website des Anbieters hinzugefügt.

[5] Brysiuk; Lehmann (2019a).

Dieser leitet allerdings nur auf die offizielle Website weiter und bietet keine weiteren Informationen.

2.2.5 Fazit

Im Allgemeinen kann man erkennen, dass die Nutzungsverteilung der Apps der Hochschule Hof mit den Marktanteilen der Smartphone Betriebssysteme in Deutschland korreliert. Dies ist besonders gut in Abbildung 2.2 zu erkennen.[6] Da die meisten der Studierenden ein Android betriebenes Handy besitzen, wird diese App auch am meisten genutzt. Genauso lässt sich die Nutzung der iOS-App erklären.

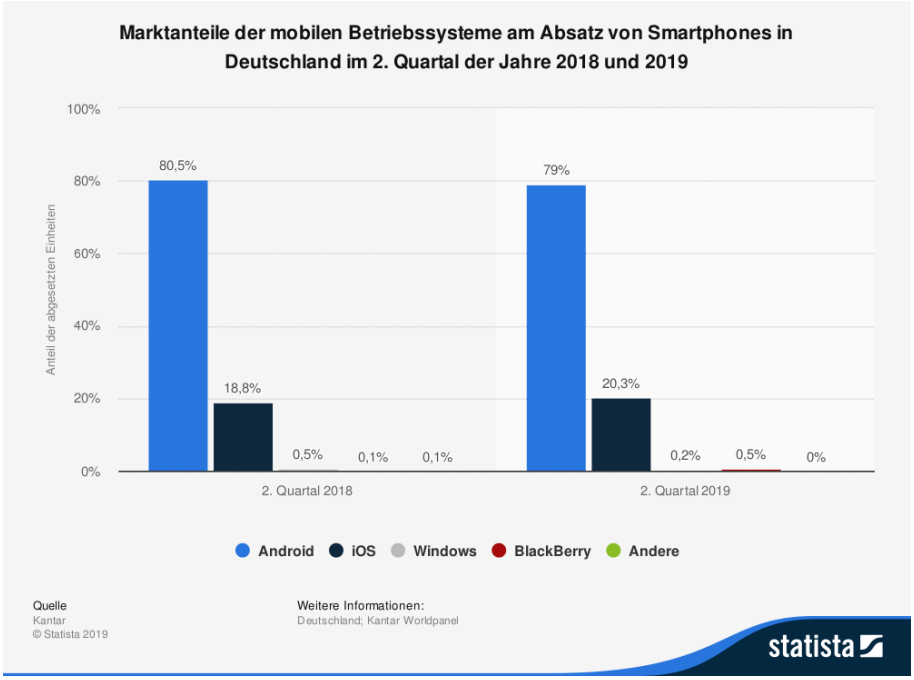


Abbildung 2.2: Marktanteile der mobilen Betriebssysteme am Absatz von Smartphones in Deutschland im 2. Quartal der Jahre 2018 und 2019[7]

Dennoch ist anzumerken, dass die Android Version die meisten zusätzlichen Features bietet, auch wenn sie laut Nutzerberichten nicht zuverlässig funktionieren und oft auch nicht als sinnvoll eingestuft werden.[8] Auch die Einbindung der Speiseplan-Informationen ist eine gern gesehene Funktion bei den Studierenden und wird demnach auch bei den Nutzern der iOS-App gefordert.

[6] Kantar (2019c).
[7] ebd.
[8] Vgl. Brysiuk; Lehmann (2019a).

3 Funktionale Anforderungen

Um eine solide Grundlage für die Entwicklung einer Anwendung zu haben, benötigt es ein Lastenheft, das vor dem Entwicklungsprozess definiert werden muss. Die oberste Priorität in einem solchen Lastenheft haben die Anforderungen des eigentlichen Auftraggebers. Zusätzlich wurden nach Wunsch des Auftraggebers noch weitere Anforderungen aus dem International Office und dem Sprachenzentrum gesammelt. Diese zwei Teile des Lastenhefts richten sich nach einer der Säulen des Leitbilds der Hochschule Hof, der Internationalisierung mit Fokus auf Indien. Im folgenden werden die gesammelten Anforderungen an die web-basierte Hochschul-App genauer betrachtet.

3.1 Auftraggeber

In den vergangenen Jahren wurden die Anwendungen der Hochschule Hof durch verschiedene Professoren und Mitarbeiter der Hochschule Hof beaufsichtigt und geleitet. Im vergangenen Sommersemester 2019 ist die Zuständigkeit der Android App an Prof. Dr. Heym übergegangen. Statt sich jedoch auf die plattformabhängige Version zu fixieren strebt dieser es stattdessen an, eine betriebssystemunabhängige Anwendung zu schaffen, die auf allen Endgeräten funktioniert und dennoch für mobile Endgeräte optimiert ist.

3.1.1 Grundlegende Anforderung

Der Fokus soll in der kommenden App darauf liegen, dass sie für alle Anwender gleich nutzbar ist, unabhängig davon, welches Endgerät genutzt wird, ob Personal Computer (PC), Smartphone oder Tablet. Auch das Betriebssystem soll dabei keine Rolle mehr spielen. Zudem ist ein großer Beweggrund für diese Arbeit die Neuentwicklung einer Anwendung, welche durch die gesammelten Erfahrungen der Apps der vergangenen Jahre an die Bedürfnisse der Nutzer angepasst wird.

3.1.2 Funktionale Anforderungen

Im folgenden werden die Anforderungen und deren Gewichtungen aufgeführt und näher erläutert.

Stundenplan

Wie in allen Anwendungen der Hochschule Hof ist die zentrale Funktion der App das Anzeigen des Stundenplans. Dieser Stundenplan soll sich allgemein nach dem Studiengang und dem Fachsemester des Anwenders richten. Zusätzlich soll es einen personalisierten Stundenplan geben, den sich der Nutzer eigenständig zusammenstellen kann. Anders als in den bereits vorhandenen Anwendungen soll der Fokus des Stundenplans auf der Personalisierung liegen.

Hierbei soll auf folgende Punkte geachtet werden:

- Fakultät des Nutzers
- Studiengang des Nutzers
- Fachsemester des Nutzers
- Einfache Einbindung nicht regulärer Vorlesungen
- Einfache Einbindung des erweiterten Vorlesungsspektrums der Hochschule (z.B. Sprachkurse)

Stundenplan Änderungen

Zusätzlich zu den allgemeinen Stundenplan Informationen und der personalisierten Ansicht sollen auch die Änderungen im Vorlesungsprogramm dargestellt werden. Hierbei ist zwischen langfristigen Änderungen und den einmaligen Änderungen zu unterscheiden. Langfristige Änderungen könnten zum Beispiel dauerhafte Raumänderungen, Wechsel des Dozenten oder die Absage einer Veranstaltung sein. Diese sollten automatisiert angepasst werden, wobei dem Nutzer einmalig mitgeteilt werden soll, dass sich diese Veranstaltung geändert hat. Kurzfristige Änderungen hingegen sind Änderungen, welche Einmalig eintreten. Darunter fallen zum Beispiel einmalige Raumänderungen, Ausfälle durch Verhinderung des Dozenten oder Verschiebungen von Vorlesungen. Diese sollten möglich klar in die Ansicht des Stundenplans fallen, sodass der Nutzer sofort erkennt, dass der aktuelle Zeitplan vom

Regelfall abweicht. Zusätzlich soll der Anwender benachrichtigt werden, wenn sich eine Änderung ergibt.

Speiseplan

Wie auch in den Anwendungen für Android und Windows soll die neue App die Speiseplan Informationen des Studentenwerk Oberfrankens anzeigen. Diese Informationen sollen nach den Wünschen des Anwenders gefiltert werden. Demnach kann sich der Anwender alle Preiskategorien anzeigen lassen oder eben nur die Kategorie, in die er selber auch fällt. Zudem soll standardmäßig das ganze Menü angezeigt werden. Stattdessen soll der Nutzer auch einige Teile des Menüs ausblenden können. So kann ein vegetarischer Nutzer zum Beispiel nur die vegetarischen Gerichte anfordern. Weiterhin sollen die Speisen mit allen zusätzlichen Informationen angezeigt werden, wie zum Beispiel den Inhaltsstoffen und Allergenen.

Anwenderverwaltung

Um die weiter oben erwähnten personalisierten Darstellungen der Stundenplan und Speiseplan Informationen anbieten zu können, ohne dass sie bei jeder Öffnung der Anwendung neu eingestellt werden müssen, ist es notwendig, dass eine Anwenderverwaltung implementiert wird. Diese kann dann die nötigen Einstellungen der Nutzer speichern. Diese Anwenderverwaltung soll jedoch so flexibel wie möglich bleiben. Im Fokus stehen nach wie vor die Studierenden der Hochschule Hof, die eine E-Mail-Adresse der Hochschule besitzen. Jedoch ist dies nicht immer der Fall. Die Nutzergruppe der Anwendung kann folgende Studierende enthalten:

- Reguläre Studierende
- Gaststudierende
- Sprachkurs Teilnehmer
- Internationale Studierende

Gaststudierende und Teilnehmer der Sprachkurse sind nicht immatrikuliert und haben demnach auch keine E-Mail-Adresse der Hochschule Hof. Dennoch belegen sie Kurse und sollen einen personalisierten Stundenplan erstellen können. Deshalb ist es wichtig, dass auch sie im Bereich der Anwenderverwaltung berücksichtigt werden.

Um nicht unnötig Daten zu sammeln und die Nutzer, die nicht mehr an der Hochschule studieren aus der Anwenderverwaltung zu entfernen ist es auch wichtig, dass Nutzerdaten automatisiert gelöscht werden, wenn sie nicht mehr benötigt werden oder der zugehörige Nutzer nicht mehr berechtigt ist, sich anzumelden. Diese Anforderung wird jedoch im Laufe dieser Arbeit nicht weiter betrachtet. Stattdessen wird auf die parallel dazu erstellte Bachelorarbeit zur Authentifizierung und Personalisierung verwiesen.[1]

Mehrsprachigkeit

Neben dem International Office stellt auch der Auftraggeber die Anforderung, die Daten der Anwendung in mehreren Sprachen bereitzustellen. Unter diesen Sprachen sollte demnach mindestens Deutsch und Englisch sein. Das Ziel dieser Anforderung ist es, die Internationalen Studierenden besser zu unterstützen.

Mobile First

Wie bereits erwähnt wurde, soll die Anwendung Plattform unabhängig sein. Demnach muss es egal sein, welches Endgerät der Anwender nutzt, um die App aufzurufen. Ebenfalls egal ist das Betriebssystem, das auf dem Endgerät installiert ist. Jedoch soll die Anwendung, auch wenn sie sich auf allen Endgeräten gleich verhalten muss, auf mobile Endgeräte angepasst sein. Das bedeutet, dass die Informationsdarstellung und die Navigation primär auf mobilen Endgeräten einfach zu bedienen sein muss. Weiterer Implementierungsaufwand kann dann nach Bedarf noch in die dynamische Anpassung an den Client investiert werden.

3.2 International Office

Die Hochschule Hof hat sich im Laufe der vergangenen Jahre immer mehr mit dem Thema Internationalisierung auseinandergesetzt. Aus diesem Grund wurde die Internationalisierung mit Fokus auf Indien im Frühjahr 2011 sogar als zentrale Säule in das Leitbilds der Hochschule Hof aufgenommen.[2]

Konkret heißt es im Leitbild der systemakkreditierten Hochschule Hof:[3]

[1] Glaser (2019a).
[2] Vgl. Broderdörp; Lender (2012).
[3] Vgl. Hochschule Hof - University of Applied Sciences (2019).

> Der Erfolg der Absolventen in nachhaltig wirtschaftenden und international agierenden Unternehmen bestimmt das Handeln aller Mitglieder des wissenschaftlichen Unternehmens Hochschule Hof. Die Studierenden werden in unserer weltoffenen Green Tech University exzellent betreut. Praxisorientierte, international ausgerichtete und der Ressourceneffizienz verpflichtete Aus- und Weiterbildung prägt unsere Arbeit. Die angewandte Forschung sichert die Aktualisierung des Wissens für die Lehre und entwickelt Lösungen zum Nutzen für die Wirtschaft.

Klar zu erkennen ist hier die mehrmalige Erwähnung der Internationalisierung der Hochschule Hof. Aus diesem Grund wird im Laufe der Neuentwicklung einer Hochschul-App auch mit dem International Office der Hochschule Hof zusammengearbeitet.

3.2.1 Problem

Wie bereits im Kapitel 2.1.1 beschrieben wurde, ist eine der Nutzergruppen der Hochschul-Anwendungen die der internationalen Studierenden der Hochschule Hof. Durch Sprachbarrieren und andere Kommunikationsschwierigkeiten, sowie durch fehlende oder nur schwer auffindbare auf englisch verfasste Informationen, fällt es diesen schwer das volle Informationsangebot der Hochschule Hof zu nutzen. Auch die Nutzung der hochschuleigenen Apps ist davon betroffen, denn viele Internationale Studierenden sind mit dem Funktionsumfang der angebotenen Anwendungen unzufrieden.

3.2.2 Funktionale Anforderungen

Den oben genannten Gründen zufolge werden folgende Anregungen und Anforderungen des International Office mit in die Analyse der verbesserten Hochschul-App aufgenommen und im Entwicklungsprozess berücksichtigt.

Spracherweiterung

Die wohl einfachste Anforderung, die sich aus den Gesprächen mit dem International Office ergeben, ist die, zukünftige Anwendungen nicht nur in deutscher Sprache anzubieten, sondern auch in anderen Sprachen, allen voran Englisch. Englisch wurde beispielsweise von etwa 23% der gesamten Befragten als weitere Sprache erwünscht, während sowohl Spanisch als auch Indisch jeweils von 20% der Befragten

angegeben wurde. Auch Französisch besitzt mit 14% einen hohen Zuspruch unter den befragten Studierenden.

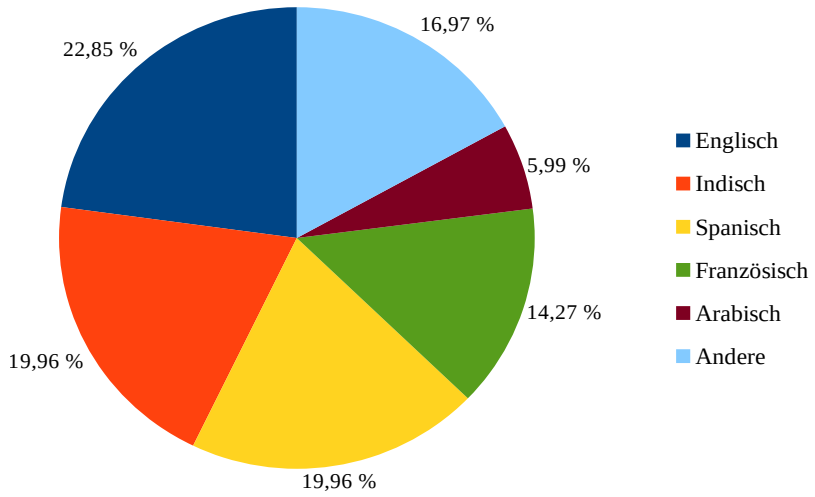


Abbildung 3.1: Umfragewerte zu gewünschten Spracherweiterungen[4]

Einfachere Stundenplanerstellung

Aus den Gesprächen mit dem International Office der Hochschule Hof hat sich ergeben, dass viele Internationale Studierende große Probleme damit haben, einen vollständigen Stundenplan über eine der Apps zu pflegen. Das liegt vor allem daran, dass viele der Internationals Vorlesungen aus verschiedenen Studiengängen und teilweise sogar aus verschiedenen Fakultäten belegen. Es ist zwar prinzipiell möglich, einen solchen Stundenplan in den Apps zu erstellen, allerdings ist dies nicht einmal vorgesehen und funktioniert so nur durch Zufall. Die internationalen Studierenden, die sich bei der Bedienung der Anwendungen ohnehin schon schwer tun, finden solche Schlupflöcher einfach nicht und befinden die Apps somit als unbrauchbar. Das International Office wünscht sich deshalb, das die personalisierte Stundenplanzusammenstellung in den Anwendungen oder in einer neuen Anwendung deutlich flexibler wird.

[4] Brysiuk, Lehmann (2019)

3.3 Sprachenzentrum

Die Hochschule Hof bietet neben ihren regulären Veranstaltungen, die durch die Fakultäten organisiert und gehalten werden, ebenfalls eine reiche Auswahl an Sprachkursen an. Diese werden vom Sprachenzentrum der Hochschule Hof veranstaltet. Die hohe Bedeutung von gut organisierten Sprachkursen an einer akademischen Institution ist unbestreitbar. Das Sprachenzentrum selbst formuliert es wie folgt:

> Sprachen sind das Tor zur Welt; unerlässlich in einer akademischen Karriere. Daher wird an der Hochschule Hof als international ausgerichteter Hochschule besonderer Wert auf eine profunde Sprachausbildung gelegt.[5]

Konkret werden aktuell an der Hochschule Hof allgemeine Sprachkurse in den Sprachen Englisch, Spanisch, Französisch, Chinesisch, Russisch, Türkisch sowie Deutsch angeboten. Diese werden im Rahmen von Pflicht- und Wahlveranstaltungen für Studierende, sowie als UNIcert-Zertifikatssystem (UNIcert)-Prüfungen und Test Deutsch als Fremdsprache (TestDaF) angeboten.

Jedoch bringt das Angebot von zusätzlichen Veranstaltungen, die nicht von den Fakultäten organisiert werden, einige Herausforderungen mit sich. Diese werden im folgenden aufgelistet, wobei mögliche Lösungen im Rahmen dieser Bachelorarbeit erörtert werden.

3.3.1 Problem

Das Ziel der Hochschule Hof ist es, den Studierenden einen einfachen und übersichtlichen Zugang zu den benötigten Informationen des Sprachenzentrums zu geben. Vor allem bei den internationalen Besuchern der Vorlesungen ist es wichtig, dass sie das Angebot der Sprachkurse leicht einsehen können, denn für sie sind deutsche Sprachkurse oft verpflichtend. Aber auch reguläre Studierende belegen oft Sprachkurse, entweder, um ihre Kenntnisse aus vorangegangenen Kursen zu vertiefen oder um sich neue Sprachen anzueignen. So geben etwa 89% der deutschsprachigen Studenten an, dass sie Informationen zum Sprachenzentrum gerne mit in eine allgemeine Hochschulanwendung eingebunden hätten. Auffällig ist hier auch die Statistik bei den nicht deutschsprachigen Studierenden. Hier sind es bereits 93%,

[5] Sprachenzentrum (2019).

die eine Erweiterung der aktuellen Anwendungen mit dem Angebot an Sprachkursen begrüßen würden. Hier hilft oft auch die Nutzung der Website nicht weiter. Somit sind zum Aufrufen allgemeiner Infos des Sprachzentrums bereits fünf Klicks von der Startseite aus gezählt nötig, für Änderungen im Kursprogramm sind es sogar schon sechs.[6]

Für die internationalen Studierenden ist hier die Informationssuche deutlich schwieriger, da diese Informationen im englischen Angebot des Sprachenzentrums auf der Hochschul Website nicht aufgeführt werden. Zudem ist die Informationsbeschaffung der Studenten in Bezug auf die Sprachkursinformationen im Vergleich zu den fachbezogenen Vorlesungen deutlich unterscheidbar. In der Android App ist das Sprachenzentrum beispielsweise als eigener Studiengang geführt und kann nur so in den Stundenplan aufgenommen werden. Dies kann bei vielen Studierenden zu Verwirrung führen. Andere Nutzer kennen diesen Spezialfall in der Implementierung der Anwendung nicht und können die Stunden ihrer Sprachkurse somit auch nicht in ihren persönlichen Stundenplan aufnehmen. Zudem sind Sprachkursinformationen, sowohl allgemeine Informationen als auch Stundenpläne zu dem Kursangebot, auf der Website der Hochschule separat geführt. Somit ist die Suche nach den Sprachkursen, die ein Studierender nutzen möchte, mit einem deutlichen Mehraufwand gegenüber der regulären Vorlesungssuche verbunden.

3.3.2 Funktionale Anforderungen

Aus den oben genannten Problemen der aktuellen Informationsdarstellung ergeben sich einige Anforderungen an eine neue Hochschul-App, welche nun weiter erörtert werden.

Einbinden von Sprachkursinformationen

Eine Anforderung des Sprachenzentrums ist es, die Informationen, die es anbietet, in eine Hochschul-App einzubinden. Der Fokus sollte darauf liegen, das Sprachangebot der Hochschule wie normale Vorlesungen zu behandeln, zu denen die Studierenden ebenfalls alle relevanten Informationen abgreifen können.

[6] Brysiuk; Lehmann (2019a).

Mehrsprachige Sprachkursinformationen

Des weiteren ist es wünschenswert, die Informationen des Sprachenzentrums nicht nur in deutscher Sprache, sondern auch in Fremdsprachen, vor allem in Englisch, einzubinden. Dabei sollte der Fokus nicht nur auf allgemeinen Informationen liegen, sondern auf spezielleren Daten, wie Details zu Stundenplanänderungen im Bereich der Sprachkurse.

Einheitliche Darstellung

Es ist wünschenswert, dass die Sprachkurse im selben Rahmen wie die regulären, verpflichtenden Vorlesungen der Studierenden dargestellt werden. Es sollte also möglich sein, die belegten Sprachkurse genauso leicht in einen personalisierten Stundenplan zu übernehmen, wie die regulär belegten Vorlesungen.

Vollständige Informationsdarstellung

Zu den Sprachkursen sollten nicht nur allgemeine Informationen wie Veranstaltungsort und Zeitpunkt stehen, stattdessen fordert das Sprachenzentrum die genauere Einbindung einiger Details. Eines dieser Details sind die Sprachniveaus der angebotenen Kurse. Werden diese angezeigt, ist es auch wichtig, dass ersichtlich ist, ob ein Kurs einen sogenannten Placementtest benötigt, der die Studierenden zur Teilnahme am Kurs berechtigt.

Vorgeschlagene Features

Das Sprachenzentrum hat des weiteren einige Vorschläge zur Verbesserung der Nutzbarkeit einer neuen Anwendung erbracht. Einer dieser Ideen ist die Möglichkeit, bei der Auswahl eines Sprachkurses in einen personalisierten Stundenplan direkt zu der Anmeldung dieses Kurses weitergeleitet zu werden. Zudem kann es hilfreich sein, angebotene Sprachkurse nach deren Niveau filtern zu können. Ebenso kann eine verbesserte Hochschul-App dem Nutzer auch die angebotenen Sprachkurse anzeigen, die in den freien Stunden seines personalisierten Stundenplans angeboten werden. So kann er die Freizeit zwischen seinen Pflichtveranstaltungen sinnvoll nutzen und die Sprachkurse werden optimal ausgelastet.

3.4 Pflichtenheft

Die in Kapitel 3 gesammelten Anforderungen können für das Projekt der neuen Hochschul-App als Lastenheft angesehen werden. Dieses wird benötigt, um klar definieren zu können, was der Auftraggeber und andere Interessenparteien von einem Projekt erwarten. Was ein Lastenheft jedoch nicht definiert ist der Rahmen und Umfang der Anforderungen, die auch im späteren Implementierungsprozess umgesetzt werden. Dafür muss vor der Implementierung noch ein Pflichtenheft definiert werden. Die darin definierten Pflichten sind dann genau die Anforderungen, die im Rahmen der zu dieser Bachelorarbeit parallel geführten Praxisarbeit umgesetzt werden.[7] Um das Lastenheft und das Pflichtenheft klar voneinander trennen zu können wird das Pflichtenheft in der eben genannten Praxisarbeit definiert. Eine Gegenüberstellung der geplanten und umgesetzten Anforderungen kann dann anhand der Referenznummern aus dem Lastenheft erfolgen. Das Lastenheft kann dem Anhang dieser Arbeit entnommen werden.

[7] Vgl. Brysiuk; Lehmann (2019b).

4 Architektur von Softwaresystemen

Die virtuelle Welt wächst immer schneller und schneller, die Digitalisierung ist mitten im Gange. Schlagzeilen über Industrie 4.0, Internet of Things (IoT), Cloud, Block Chain, Development and IT Operations (DevOps), BigData oder Artificial Intelligence (AI) tauchen immer öfter in den digitalen Nachrichten Welt auf. Viele Unternehmen setzten sich das Ziel, deren Strukturen zu modernisieren, so wie die Unternehmensprozesse zu optimieren. Diese sollen flexibler, schneller und effizienter ablaufen. Doch auch Privatpersonen rüsten stark nach, ein Leben ohne Smartphone ist heutzutage kaum noch vorstellbar. Navigation, Kommunikation, Unterhaltung, Shopping, Banking und vieles mehr damit sind möglich, ein Gerät für alle Fälle. Allein im zweiten Quartal des Jahres 2019 stehen über 4.5 Millionen Anwendungen für sogenannte Smartphones zur Verfügung und es werden täglich immer mehr.[1]

Der Wandel unter dem Schlagwort *Digitale Transformation* hat natürlich auch gravierende Auswirkungen auf die Systemarchitekturen. Denn die Bauweisen solcher Systeme sollen Zukunftsfähig sein, obwohl dies nicht so einfach ist, schon allein weil heute noch nicht bekannt ist, was morgen neu erscheinen wird. Die Gefahr besteht darin, dass viele der aktuellen Information Technology (IT)-Systeme in ihrer Existenz bedroht wären, wenn sie keine grundlegende Modularisierung vorweisen könnten. Genau deshalb muss das Ziel der modernen IT-Entwickler sein, deren Systemarchitekturen so zu gestalten, dass eine möglichst hohe Modularität und Flexibilität eine Grundlage deren wird.[2]

Die Modularisierung ermöglicht die Beherrschung der *digitalen Transformation*. Dadurch entstehen Flexibilitätsvorteile, die die ideale Reaktion auf das dynamische Wachstum der Digitalisierung ist. Somit handelt es sich beim Thema Modularisierung um eine der wichtigsten Konzepte der Softwarearchitektur. Die Modularisierung eines IT-Systems ist erfüllt, wenn das gesamte IT-System in einzelne

[1] Vgl. Kantar (2019a).
[2] Vgl. Dowalil (2018), S. 17 f.

Komponente zerlegt werden kann. Jede einzelne Komponente ist als abgegrenzter Teil des Systems definiert und übernimmt eine entsprechende Funktionalität des IT-Systems.
Die einzelnen Komponenten sollten möglichst unabhängig voneinander sein und zudem die einfache Ersetzbarkeit durch eine andere Komponente mit denselben Eigenschaften ermöglichen. Auch bei einem Ausfall einer der Komponenten wird das System weiterhin funktionieren, dann eben ohne die Funktionalität der ausgefallenen Komponente.
Die Zerlegung in einzelne Komponenten bieten folgende Vorteile:[3]

- Diese können ohne Seiteneffekte und Abstimmungsarbeit weiterentwickelt werden.
- Diese können unabhängig von Rest dokumentiert und verstanden werden.
- Diese können an ihren ein- und ausgehenden Schnittstellen isoliert getestet werden.
- Diese können leicht ausgetauscht werden.
- Diese sind flexibel an das Wachstum des IT-Systems anpassbar.
- Diese reduzieren die Komplexität des IT-Systems.

> „Sollte eine Software mit der Zeit irgendwelche Probleme bekommen, sei es beispielsweise mit der Stabilität oder mit der Wartbarkeit, so werden sich diese Probleme, so man auch immer auf Modularisierung geachtet hat, immer nur auf einen Teil des Systems beziehen und niemals auf die gesamte Software. Software ab einer gewissen Größenordnung, die keine erkennbaren Strukturen aufweist, kommt potenziell in existenzielle Bedrohung, sobald sie die ersten Probleme dieser Art aufweist.“ [4]

Ein ausgezeichnetes Negativbeispiel dafür sind die aktuellen Hochschul-Apps. Diese wurden für jedes System separat Entwickelt und das nur mit genau den Funktionalitäten, die zu dieser Zeit erforderlich waren. Somit hat jedes IT-System eine eigene Softwarearchitektur bekommen und die Funktionalitäten wurden für jedes

[3] Vgl. Dowalil (2018), S. 3 f.
[4] a.a.O., S. 3.

System auf eine andere Art und Weise implementiert. Jedoch sind durch die bereits erwähnte Digitalisierung über die Jahre immer neue Anforderungen an die IT-Systeme entstanden. Diese wurden bereits ausführlich in Kapitel 3 beschrieben. Durch die nicht modularisierte Gestaltung der Softwarearchitektur der bestehenden IT-Systemen ist die Entwicklung bei einem Punkt angekommen, bei der gemeinsame Erweiterungen oder Änderungen der Anwendungen für alle IT-Systeme entweder aufwändig oder sogar nicht möglich sind.

Somit handelt es sich bei der Erstellung einer Softwarearchitektur um eine der wichtigsten Disziplinen in der IT-Branche. Zu beachten ist jedoch, dass die Softwarearchitektur nicht nur aus dem Drang nach Modularität besteht, denn es müssen noch viele andere wichtigen Anforderungen und Prinzipien eingehalten werden, um eine zukunftsfähige Softwarearchitektur zu erstellen. Auf diese Anforderungen und Prinzipien wird in den nächsten Kapiteln eingegangen. Außerdem muss berücksichtigt werden, dass nur modulare Softwarearchitekturen in Frage kommen, denn das Ziel dieser Bachelorarbeit ist es, eine erweiterbare, betriebssystemunabhängige Hochschul-App zu entwickeln.

4.1 Grundlegendes

Die Realisierung eines jeden Projekts benötigt immer einen ersten Architekturentwurf. Egal welche Industriezweige es betrifft, ob der Bau einer Brücke geplant werden soll, ein Netzwerk umgesetzt werden muss oder eine Anwendung implementiert werden soll, am Anfang steht immer die Architektur. Ein früher Entwurf, der später noch detaillierter beschrieben werden kann, bietet anfangs eine gute Grundlage für vorzeitige Bewertungen und Entscheidungen zur Verteilung der Ressourcen. Außerdem lassen sich oft vorzeitig Probleme erkennen und schneller beheben.

Beim Entwurf einer Anwendung werden die Spezifikationen der einzelnen Funktionalitäten in Komponente zerlegt um die Komplexität weitgehend nach dem Prinzip *teile und herrsche* zu minimieren. Dabei wird der Aufbau der benötigten Teilsysteme, sowie die Festlegung der Verbindungen und Abhängigkeiten der Komponenten untereinander, beschrieben. Jedoch besteht eine Softwarearchitektur nicht nur aus der Anwendung selbst. Stattdessen müssen auch andere Schichten beachtet werden, um einerseits die nicht-funktionalen Anforderungen abzudecken und andererseits die Anpassung an die Umgebung, auf dem die Anwendung später implementiert

werden soll, zu gewährleisten. Zudem muss auch auf die nötige Middleware eingegangen werden, die benötigt wird, um mit externen Systemen kommunizieren zu können. Somit ist die Entscheidung des Software-Designs einer der wichtigsten Punkte bei Erstellung einer Anwendung.[5]

4.2 Nicht-funktionale Anforderungen

Die nicht-funktionale Anforderungen sind meistens die Anforderungen, die nicht explizit vom Auftraggeber erwähnt werden, sondern die, die sich eher automatisch aus den Vorgaben ergeben oder die man bei aktueller Software als selbstverständlich sieht. Die meisten solcher Anforderungen sind in Abbildung 4.1 zu sehen.

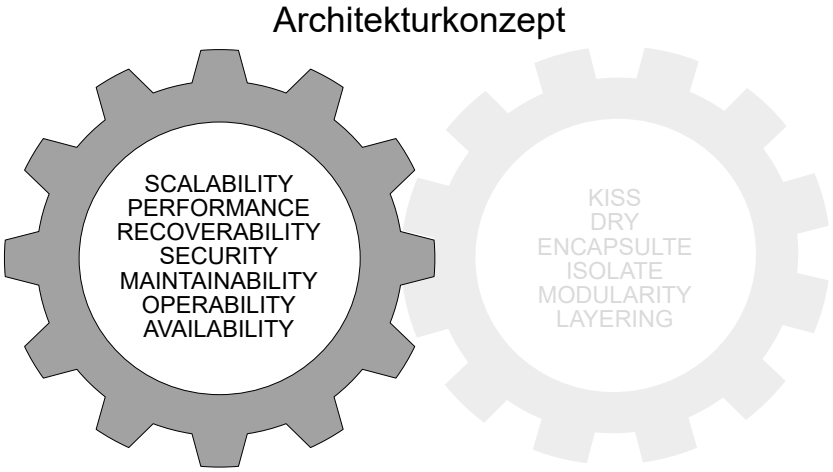


Abbildung 4.1: Allgemeine Anforderungen an eine Architektur[6]

Es ist bei einem Softwareentwurf immer wichtig, sich über alle nicht-funktionalen Anforderungen Gedanken zu machen, denn diese bilden den Grundbaustein für eine erfolgreiche und zukunftsfähige Anwendung. Dadurch können spätere Mängel leichter behoben werden, neue Anforderungen schneller umgesetzt werden und die Anwendung an das Wachstum der Hochschule angepasst werden.

[5] Vgl. Softwarearchitektur (2019); Vgl. Dowalil (2018), S. 1.
[6] Brysiuk, Lehmann (2019)

4.3 Prinzipien

Ein weiterer wichtiger Punkt ist die Beachtung von Prinzipien beim Softwareentwurf. Diese werden bei den nicht-funktionalen Anforderungen aus Abbildung 4.1 im Architekturkonzept ergänzt. Das Resultat in Abbildung 4.2 bietet eine Basis zur Vorgehensweise bei der Entwicklung eines Softwareentwurfs.

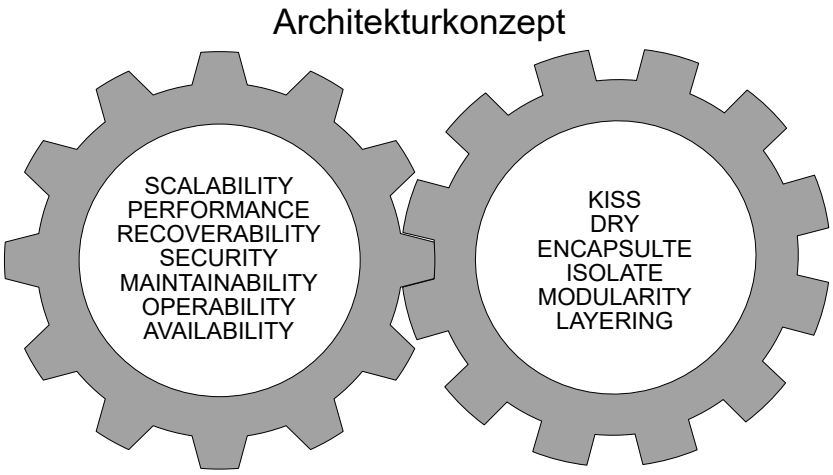


Abbildung 4.2: Allgemeine Prinzipien einer Architektur[7]

Die nicht-funktionalen Anforderungen sind im allgemeinen selbsterklärend, deshalb wird nicht weiter auf sie eingegangen. Jedoch bestehen die Prinzipien des Softwareentwurfs aus weniger selbsterklärenden Begriffen und Akronymen, weshalb diese im folgenden genauer erläutert werden.

KISS - Keep it Simple and Stupid

Bereits im 20. Jahrhundert hat der amerikanische Flugzeugkonstrukteur Clarence Leonard (Kelly) Johnson das Keep It Simple and Stupid (KISS)-Prinzip erstmals erwähnt und treffend beschrieben:

> „Reduce reports and other paperwork to a minimum.“[8]

Die Idee dahinter ist, dass die Gestaltung der Maßnahmen oder Entscheidungen so wenig wie möglich Komplexität beinhalten, jedoch sollen die Ergebnisse die erwarteten Anforderungen erfüllen und dazu flexibel genug sein, um Änderungen

[7] Brysiuk, Lehmann (2019)
[8] Rich (1995), S. 231.

vorzunehmen. Also soll die Umsetzung einfach, schnell, verständlich, logisch, begreifbar und passend an die Anforderungen und nicht auf die Möglichkeiten sein. Bei einer Softwarearchitektur darf bei der Einhaltung des KISS-Prinzips nicht auf die angemessene Modularisierung verzichtet werden.[9]
Häufig wird beim Konzeptionieren versucht von Anfang an auf alle möglichen Variationen der Anforderungen einzugehen. Dies umfasst auch die Anforderungen, die noch nicht gestellt wurden, von denen man allerdings ausgeht, as sie im Verlauf des Projektes noch vom Auftraggeber ergänzt werden. Das Ziel davon ist es, in der Zukunft den zusätzlichen Aufwand für die Implementierung zu sparen. Das Problem dabei ist, dass die Wahrscheinlichkeit ziemlich gering ist, dass solche zusätzliche Funktionalitäten tatsächlich noch benötigt werden. Das führt letztendlich zu einer komplexeren Anwendung, deren Flexibilität auf ein Minimum reduziert wird. Dies wird zusätzlich dadurch unterstützt, dass eine Anforderung sich während des Entwicklungsprozesses ändert, da der Auftraggeber oft nicht mehr zufrieden mit der ursprünglichen Lösung ist[10].

Folgende Formel zur Berechnung des Nutzen eines Mehrwerts $N_{mehrwert}$ kann dazu dienen, besser abschätzen zu können, ob ein gewünschter Mehrwert tatsächlich vorher berücksichtigt werden soll[11]:

$$N_{mehrwert} = (A_{nach} * P_{gebrauch}) - A_{vor} \tag{4.1}$$

wobei
A_{vor} = *Aufwand eines Features bei der Planung von Anfang an*
A_{nach} = *Aufwand eines Features bei späterer Aufnahme*
$P_{gebrauch}$ = *Wahrscheinlichkeit, dass ein Feature später benötigt wird*

DRY - Dont't Repeat Yourself

<Durch generische Programmierung können Code-Duplikate vermieden werden und wiederholende Algorithmen an mehreren Stellen wiederverwendet werden. Die

[9] Vgl. Dowalil (2018), S. 19.
[10] a.a.O., S. 19 f.
[11] a.a.O., S. 20.

Idee ist es also beim DRY-Prinzip, die Redundanzen von Programmstellen zu reduzieren , beziehungsweise sie komplett zu vermeiden. Hier ist aber Vorsicht geboten, denn die nicht-funktionalen Anforderungen wie Verfügbarkeit oder Wartbarkeit haben eine höhere Priorität. Man soll alle Prinzipien und Anforderungen auswerten und sogar priorisieren, um eine ausgewogenen Softwarearchitektur erstellen zu können.[12]

Encapsulation - Prinzip der Kapselung

Das Konzept des *Information Hidings* wurde bereits im Jahr 1972 von David Parnas vorgestellt:

> „The second decomposition was made using "information hiding" as a criterion. The modules no longer correspond to steps in the processing. The line storage module, for example, is used in almost every action by the system. Alphabetization may or may not correspond to a phase in the processing according to the method used. Similarly, circular shift might, in some circumstances, not make any table at all but calculate each character as demanded. Every module in the second decomposition is characterized by its knowledge of a design decision which it hides from all others. Its interface or definition was chosen to reveal as little as possible about its inner workings."[13]

Der Zweck besteht darin eine Art Geheimnisprinzip zu erstellen, um bestimmte Stellen zu schützen, ob für die Zugriffsberechtigungen oder für die Sicherung der Integrität der Daten. Es werden Komponenten verborgen und eine Schnittstelle für einen öffentlichen Zugriff definiert, somit hat eine Änderung oder der Austausch einer Komponente keine Auswirkung nach außen. Außerdem wird die Robustheit der Anwendung durch die erhöhte Begrenzung der Abhängigkeiten der Komponenten gestärkt.[14]

Lose Kopplung

l IT-Systeme bestehen meist aus mehreren Bausteinen, Komponenten und Subsystemen. Diese werden immer zwangsläufig Abhängigkeiten untereinander bilden, da

[12] Vgl. Dowalil (2018), S. 21.
[13] Parnas (2001), S. 1056.
[14] Vgl. Dowalil (2018), S. 21 f.

einfach das ganze IT-System nicht völlig isoliert werden kann. Die Idee der losen Kopplung ist es, die Verbindungen der einzelnen Bausteine untereinander zu verringern, sowohl auf der Hardware-, als auch auf der Softwareebene. Bei den Verbindungen sollen folgende Punkte berücksichtigt werden:[15]

- Laufzeitumgebungen
- Ausführungsort
- Verwendete Technologien
- Ausführungszeit
- Daten und Formate bei der Kommunikation

Separation Of Concerns

Auf das Prinzip *Seperation Of Concerns*, zu Deutsch etwa *Trennung der Angelegenheiten*, wurde im Verlauf den gesamten Kapitels immer wieder aufmerksam gemacht und auf dessen Wichtigkeit hingewiesen. *Separation of Concerns* ist das Prinzip der Modularität, welches bereits ausführlich am Anfang des Kapitels 4 beschrieben wurde.

Layering - Schichtenarchitektur

Wie bereits in Prinzip der Modularität beschrieben wurde, ist es wichtig beim Entwurf eines IT-Systems dieses in einzelne Komponente und Module zu zerlegen. Dabei bilden die Komponenten in ihrem Zusammenspiel auf einer Ebene verschiedene Schichten. Dieser Schichten sind nach Funktionalität getrennt. Beispielhaft ist das in der Abbildung 4.3 dargestellt. Das System besteht aus vier verschiedenen Schichten die wiederum einzelne Komponenten enthalten[16].

[15] Vgl. Dowalil (2018), S. 25 ff.
[16] Vgl. a.a.0., S. 31 ff.

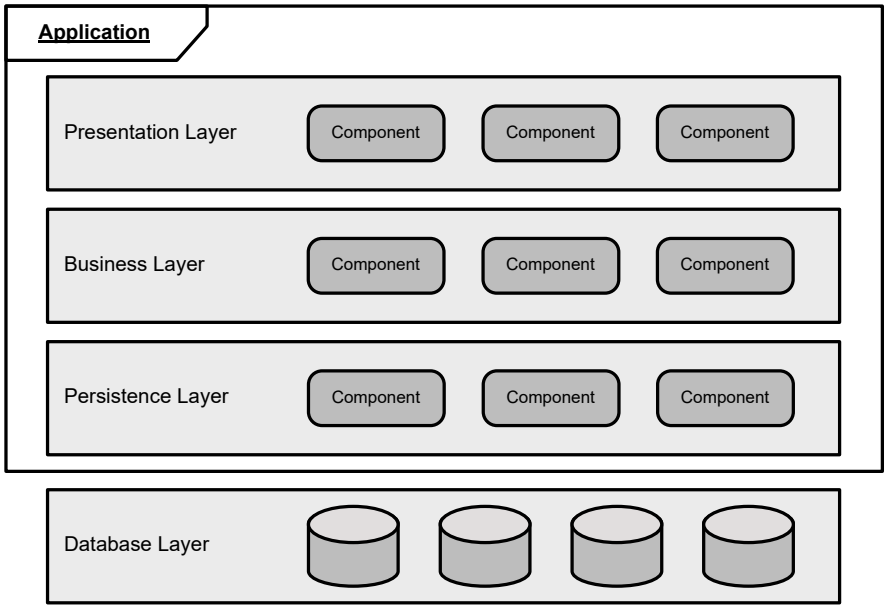


Abbildung 4.3: Schichtenarchitektur[17]

Für Entwickler werden große Projekte durch die Einhaltung des *Layerings* deutlich beherrschbarer. Ähnlich wie unser Gehirn solche Zusammenhalte verarbeitet wird das große Ganze in Schichten unterteilt, die sich in Funktionalitäten trennen. Die einzelnen Schichten werden dann wieder in Komponenten zerlegt, bis eine Komponente klein genug ist um sie detailliert zu verstehen. So muss man sich am Ende nur einen Überblick über das Gesamtbild verschaffen[18].

4.4 Fazit

Die Verbreitung der verteilten Programme, im Englischen *Distributed Apps*, hat sich in den vergangenen Jahren schnell weiterentwickelt. Diese Anwendungen laufen nicht auf einem einzelnen lokalen Rechner, sondern werden auf mehrere unabhängige Module verteilt, die wiederum auf mehreren unterschiedlichen Servern im Netzwerk verteilt sein können. Somit stellen verteilte Anwendungen eine Art verteiltes - beziehungsweise vernetztes - System dar.[19]

Ein Beispiel hierfür sind Cloud Computing Systeme, diese bauen auf der Softwarearchitektur von verteilten Systemen auf. In Abbildung 4.4 ist eine Statistik abge-

[17] Vgl. Dowalil(2018), S. 31 ff.
[18] Vgl. a.a.O., S. 36.
[19] Vgl. Schill; Springer (2012), S. 3 f.

bildet, in der das Wachstum für die Nutzung von Cloud Computing Systemen in deutschen Unternehmen in den Jahren 2011 bis 2018 zu sehen. Man erkennt, dass bereits im Jahr 2018 73% der befragten Unternehmen Cloud Computing Systeme benutzten. Weitere 19% überlegten oder planten bereits solche Systeme zu nutzen.[20]

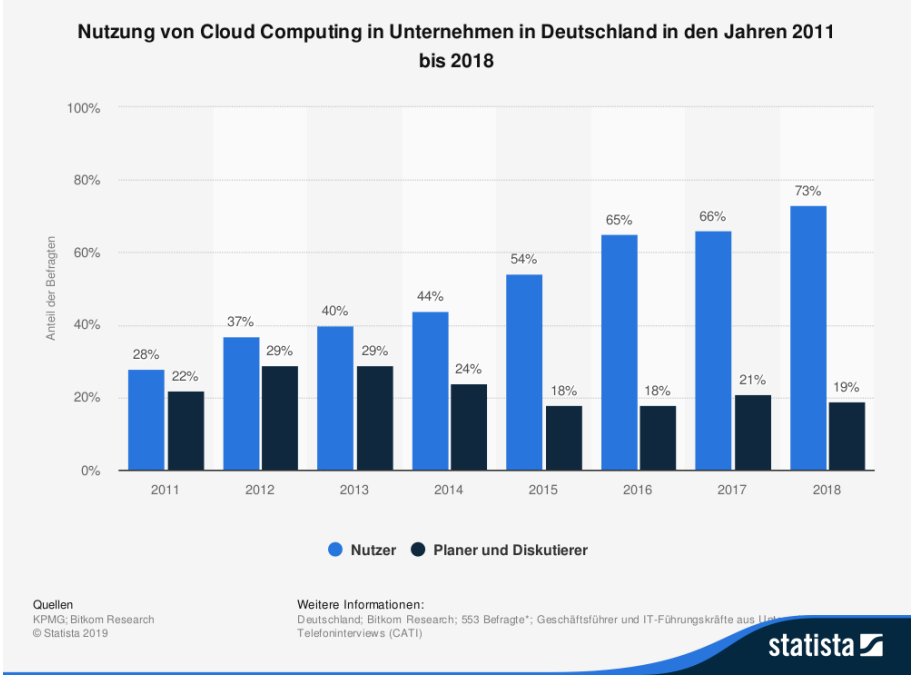


Abbildung 4.4: Nutzung von Cloud Computing[21]

Wie bereits erwähnt setzen sich verteilte Systeme aus mehreren unabhängigen Modulen zusammen, die auf unterschiedlichen Servern implementiert sind. Durch die Kopplung der Module entsteht ein vollständiges System, welches eine räumliche Trennung über weite geografische Distanzen ermöglicht. Dies ermöglicht einen parallelen und effizienten Zugriff, in unserem Fall zum Beispiel auf Stundenplan- und Mensadaten, denn die Abfragen würden von zwei unterschiedlichen Modulen auf verschiedenen Servern bearbeitet werden. Einen weiterer Vorteil bei dieser Herangehensweise ist der Lastausgleich. Durch eine Modularisierung der Funktionalitäten und der Verteilung dieser auf mehreren Servern, kann die Verarbeitungslast eines zentralen Servers auf mehrere Instanzen verteilt werden, um eine einseitige Auslastungen zu vermeiden. Außerdem wird ein hohes Maß an Fehlertoleranz, Ausfallsicherheit, Skalierbarkeit und Verfügbarkeit auf der Basis verteilter Systeme gewährleistet.[22]

[20] Kantar (2019b).
[21] ebd.
[22] Vgl. Schill; Springer (2012), S. 5 f.

Ausgehend von den bisherig beschriebenen Anforderungen und Prinzipien an die Softwarearchitektur, sowie der Wichtigkeit von Modularisierung und den Anforderungen aus Kapitel 3, bieten verteilte Systeme eine gute Voraussetzungen für die Architekturkonzepte der Hochschul-App. Dabei bieten diese in sich verschiedene Umsetzungen der Systemarchitektur, die teilweise aufeinander aufbauen[23]:

- Client-Server-Modell
- Objektorientiertes Modell
- Komponenten basiertes Modell
- Dienst orientiertes Modell

Verteilte Anwendungen basieren meist auf dem Dienst orientierten Modell, welches auch unter dem Begriff der *Service-orientierten* Architektur im Zusammenhang mit Webservices bekannt ist[24]. Somit sind Webservices ein guter Ansatz für die konkrete Implementierung der verteilten Anwendung und der Grundbaustein für die Hochschul-App.

[23] Vgl. Schill; Springer (2012), S. 13 f.
[24] ebd.

5 Web Services

Web Services ermöglichen es, verschiedene Programme miteinander in einem Netzwerk zu verbinden. Hierfür stellen sie eine einheitliche Schnittstelle für die Kommunikation untereinander zur Verfügung. Dabei spielt die Wahl der Plattform - ob Windows, Linux oder iOS - als auch die Wahl der Programmiersprachen wie Java, Hypertext Preprocessor (PHP) oder Python keine Rolle. Somit können einzelne Programme, Module oder Komponenten in unterschiedlichen Sprachen von unterschiedlichen Teams entworfen werden, so wie auf beliebigen Systemen installiert werden.

Eine der zentralen Funktionen eines Webservices ist die Synchronisation von Client und Server. Ein Webservice stellt dem Client Informationen, zum Beispiel über Datentypen, möglichen Operationen sowie die Parameter und Rückgabewerte und Hinweise darüber, wie diese Aufgerufen werden können, zur Verfügung. Für die Kommunikation ist es wichtig ein Transportprotokoll und das Format für die übertragenen Daten festzulegen. Als Transportprotokolle können beispielsweise Simple Mail Transfer Protocol (SMTP), Transmission Control Protocol (TCP), Java Message Service (JMS), HTTP oder Hypertext Transfer Protocol (Secure) (HTTPS) genutzt werden. Übliche Datenformate sind dabei Extensible Markup Language (XML) oder JavaScript Object Notation (JSON).

Allein mit dem Konzept der Webservices ist die Implementierung einer verteilten Anwendung jedoch nicht möglich. Denn Webservices stellen hauptsächlich Dienste und Funktionen über definierte Schnittstellen und Standards über das Netzwerk zur Verfügung, sie dienen also eher zur Kommunikation der Systeme untereinander und weniger zur Abarbeitung logischer Konzepte. Zudem bieten Webservices kein ausreichendes Maß an loser Kopplung. Wie bereits in Kapitel 4.4 erwähnt wurde, werden Webservices oft im Zusammenhang mit Service-oriented Architecture (SOA) gebracht. Dies liegt daran, dass die grundlegenden Konzepte von SOA

durch Webservices umgesetzt werden können.[1]

5.1 Serviceorientierte Architektur

Um zu verstehen, was SOA ausmacht und wie es verstanden werden muss empfiehlt es sich folgende Aussage zu lesen:

> „Service Oriented Architecture (SOA) is an architectural paradigm that has gained significant attention within the information technology (IT) and business communities.“[2]

Eine konkrete oder einheitliche Definition für SOA existiert derzeit nicht, deswegen wird SOA als ein Paradigma der Architektur bezeichnet. Außerdem ist SOA eben deswegen nicht wirklich eine Architektur, sondern eher ein Design/Konzept für verteilte Systeme. Da viele unter SOA etwas unterschiedliches verstehen und das Paradigma meistens verschieden implementiert wird, hat die Organization for the Advancement of Structured Information Standards (OASIS) ein SOA-Referenzmodell definiert um es zu vereinheitlichen.[3]

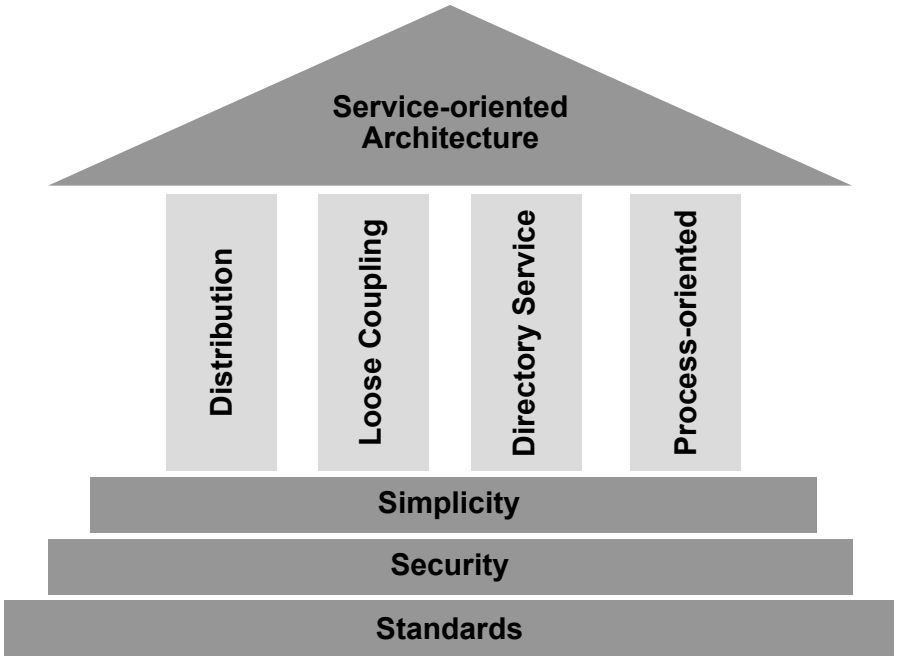


Abbildung 5.1: SOA Tempel[4]

[1] Vgl. Kiltz (2010), S. 23 ff.; Vgl. Melzer (2007), S. 8 ff.
[2] Reference Architecture Foundation for Service Oriented Architecture V. 1.0 (2012).
[3] Vgl. Melzer (2007), S. 11; Vgl. Josuttis (2008), S. 2 f.; Vgl. OASIS SOA Reference Model TC (2005).

> Autor Ingo Melzer beschreibt das Grundkonzept von SOA passend als einen Tempel, wie er in Abbildung 5.1 zusehen ist, über den er folgendes sagt:
>
> „Das Fundament wird von offenen Standards, Sicherheit und Zuverlässigkeit gebildet. Die verteilten Dienste, die lose Kopplung, die Plattformunabhängigkeit und die prozessorientierte Struktur sind die tragenden Säulen.“ [5]

Aus diesen grundlegenden Aussagen über SOA können einige Schlussfolgerungen abgeleitet werden. Durch das Paradigma SOA wird die funktionale Zerlegung eines Gesamtsystems in einzelne Komponenten, genannt Services, und die Verteilung dieser auf ein Netzwerk ermöglicht. Wobei hierfür die einzelnen Services Plattform-, sprachen- und Frameworkunabhängig von einander entwickelt und wiederverwendet werden können.[6]

5.1.1 Technische Konzepte

Um das Potential von SOA nutzen zu können, werden verschiedene technische Konzepte verwendet. Die wichtigsten technischen Konzepte von SOA sind folgende:[7]

- **Interoperabilität**

  Unter Interoperabilität ist die Sicherstellung der Kommunikation unterschiedlichen Systemen ohne großen Aufwand miteinander gemeint - mit den Fokus auf den Datenfluss.

- **Services**

  Services dürfen nicht mit Webservices verwechselt werden. Webservices sind ein technisches Konzept zur Umsetzung der Kommunikation in SOA, wobei Services in SOA funktionale Komponenten sind.[8]

[4] Melzer (2007)
[5] a.a.O., S. 11 f.
[6] Vgl. Dowalil (2018), S. 121 f.
[7] Vgl. Josuttis (2008), S. 11.
[8] Vgl. Dowalil (2018), S. 123 f.

- **Lose Kopplung**

  Wurde bereits in Unterkapitel 4.3 ausführlich behandelt.

5.1.2 Zusammenfassung

Die Service orientierte Architektur mit Unterstützung von Web Services bietet eine gute Voraussetzung für die Abdeckung der nicht-funktionalen Anforderungen sowie der Prinzipien für eine Softwarearchitektur aus Kapitel 7. An folgender Stelle wird genauer erläutert, welche nicht-funktionalen Anforderungen vorhanden sind und warum diese von SOA in Kombination mit Webservice-Ansätzen erfüllt werden:

- **Skalierbarkeit**

  Durch die Implementierung der Anwendung als verteiltes System kann diese auf einen beliebigen Server mit den benötigten Ressourcen ausgelagert werden. Sollten die Ressourcen irgendwann unvorhergesehen nicht mehr ausreichen, kann die Anwendung auf einen anderen Server ausgelagert werden, ohne eine Beeinflussung der Funktionalitäten der Anwendung zu erzwingen. Eine weitere Möglichkeit hierfür ist die Auslagerung der Anwendung in eine Cloud, diese Vorgehensweise ist auch als Infrastructure as a Service (IaaS) bekannt. Unter IaaS sind Cloud-Computing-Plattformen gemeint, die Speicher und Rechenleistung auf den Servern für die Benutzer zur Verfügung stellen. Einer der bekanntesten IaaS ist der Amazon Web Service (AWS). Durch die Auslagerung der Anwendung auf die AWS-Cloud wird das Management Werkzeug *Elastic Beanstalk* für den Benutzer zur Verfügung gestellt. Das Elastic Beanstalk übernimmt automatisch das Management über die Bereitstellung zusätzlicher Kapazitäten, Lastenverteilung sowie die Skalierung.[9]

  Eine weitere Möglichkeit dafür, die Skalierbarkeit zu erhöhen, bieten Docker Container. Die Anwendung kann ebenso wie auf mehrere Server auch in einen Docker Container verschoben werden. Die Container lassen sich leicht von einem System auf ein anderes übertragen, benötigen weniger Ressourcen als virtuelle Maschinen und sollten zusätzliche Instanzen der Anwendung benötigt werden, dann können einfach neue Container gestartet werden. Sobald

[9] Vgl. AWS Elastic Beanstalk (2019).

diese nicht mehr benötigt werden, können diese ebenso wieder gestoppt werden.[10] Somit steht der Skalierbarkeit der Anwendung nichts im Wege.

- **Performance**

  Generell gilt, dass die Ladezeiten einer Anwendung in der Regel nicht länger als Drei Sekunden dauern dürfen. Die Performance von SOA hängt zunächst von unterschiedlichen Faktoren ab. Manche können eher gut beeinflusst und optimiert werden, andere eher weniger. Die heutige Technologie ist ausreichend ausgereift für die Implementierung schneller und effizienter verteilter Anwendungen. Ein Beispiel dafür ist der weltbekannte Streaming-Anbieter Netflix, der bereits die erwähnten IaaS von AWS verwendet. Die Filme oder Serien lassen sich in Echtzeit in beliebiger Auflösung streamen.[11] Außerdem ermöglicht SOA bei einem unerwartet hohen Zugriff immer noch das Management der Lastenverteilung der einzelnen Komponenten. Somit wird die Performance durch höhere Auslastungen der Anwendung nicht beeinflusst.

- **Verfügbarkeit**

  Eine Anwendung soll im Idealfall permanent erreichbar sein. Ein hohes Maß an Verfügbarkeit kann durch SOA erbracht werden. Durch die Replikation der Anwendung auf mehreren Servern, als IaaS oder durch Auslagerung in den Docker-Container, kann bei einem Ausfall einer Instanz eine andere Instanz verwendet werden.

- **Sicherheit**

  Sicherheit ist einer der Wichtigsten Aspekten in einer Software, denn auch für die Anwendungen, die keine vertraulichen Informationen beinhalten, sollte immer einer Sicherheitskonzept entworfen werden. Bei einer Absicherung eines Webservices spielen folgende Aspekte eine wichtige Rolle:

  - Authentifizierung
  - Autorisierung
  - Integrität

[10] Vgl. Was sind Docker-Container? (2017).
[11] Vgl. Netflix-Fallstudie (2016).

Es ist nicht unmöglich eine sichere Service-orientierte Architektur zu konzipieren, dennoch muss dafür frühzeitig auf die Sicherheitsaspekte eingegangen werden. Zu diesen Themen existieren bereits eine Reihe bewährter Technologien, Techniken und Standards um Webservices sicherer zu gestalten.[12]

Um die Vertraulichkeit im Nachrichtenaustausch aufbauen zu können, können verschiedene Methoden des symmetrischen oder asymmetrischen Verschlüsselungsverfahren verwendet werden. Außerdem sind folgende Standards in Webservices bereits etabliert und erleichtern die Implementierung der Sicherheitsaspekte enorm:[13]

- WS-Security
- WS-Policy
- WS-Trust
- WS-SecureConversation
- WS-Privacy
- WS-Federation
- WS-Authorization

Wie die Authentifizierung, Autorisierung und Vertraulichkeit in einem Webservice gewährleistet werden kann, wird ausführlicher in der Bachelorarbeit zur Planung der Nutzerverwaltung und des Frontends der Stundenplan-App der Hochschule Hof behandelt.[14]

- **Wartbarkeit**

Die Wartbarkeit ist einer der Wichtigsten nicht-funktionalen Anforderungen, denn in der Praxis ist es oft der Fall, dass die Anforderungen an die Anwendungen sich mit der Zeit ändern oder neue Funktionalitäten hinzu kommen. Außerdem werden des öfteren erst nach einiger Zeit Fehler in der Anwendungen erkannt, die während der Entwicklung verborgen geblieben sind. Durch die modulare und flexible Gestaltung der Services in einer SOA sind diese leicht Austausch- oder Änderbar.

[12] Vgl. Melzer (2007), S. 188 f.
[13] Vgl. a.a.O., S. 190 f.; Vgl. a.a.O., S. 208 ff.
[14] Vgl. Glaser (2019a).

Im weiteren Verlauf wird nun erläutert, welche Prinzipien der Softwarearchitektur durch SOA in Verbindung mit Webservice-Ansätzen bereits von Beginn an gegeben sind.

- **Kapselung**

  Services stellen eine definierte Schnittstelle für den Zugriff bereit, jedoch sind die genauen Details der Implementierung für den Benutzer unsichtbar.[15]

- **Lose Kopplung**

  Die Services können auf verschiedenen Systemen in verschiedenen Programmiersprachen realisiert werden, dadurch wird die Kopplung zwischen den Hardware- und Softwareebenen verringert.

- **Modularisierung**

  Durch die Trennung der logischen Funktionalitäten in einzelne Komponenten und der Verteilung dieser auf verschiedene Services werden die Abhängigkeiten der Komponenten reduziert. Außerdem entsteht eine fachliche Trennung der Funktionalitäten.

- **Layering**

  Die Umsetzung von Service-orientierten Architekturen kann in mehreren Schichten realisiert werden, die bereits in Abbildung 4.3 zu sehen waren und im Kapitel 7.3 detaillierter beschrieben werden.

Somit bietet die Service-orientierte Architektur mit Webservices einen idealen Ansatz für die Umsetzung einer sicheren und zukunftsfähigen Hochschul-App. Jedoch kann die Realisierung von Webservices nochmals durch zwei Arten unterschieden werden, Simple Object Access Protocol (SOAP) und Representational State Transfer (REST).

[15] Vgl. Melzer (2007), S. 13.

5.2 Webservice Arten

In den weiteren Unterkapiteln werden die beiden Webservice Arten kurz erklärt und anhand einer Entscheidungsmatrix gegenübergestellt und bewertet. Zu unterscheiden ist, dass SOAP ein definiertes Netzwerkprotokoll ist, das auf XML basiert. REST ist hingegen ein *Webservice-Light-Technologie* Architekturstil, der sich nur auf das Kommunikationsprotokoll HTTP, respektive HTTPS, beschränkt und meisten mit dem JSON-Datenformat in Verbindung gebracht wird.[16]

5.2.1 Simple Object Access Protocol (SOAP)

> „[Simple Object Access Protocol] provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML.“[17]

SOAP wurde durch das World Wide Web Consortium (W3C) als industrieller Standard definiert, in dem es in drei wesentlichen Teile aufgeteilt ist:

- **SOAP Envelope**

  Der SOAP-Envelope definiert ein Framework, das beschreibt, was in einer Nachricht enthalten ist, wer damit umgehen sollte, und ob gewisse Parameter optional oder obligatorisch sind. Die SOAP Nachricht besteht aus einem Envelope, einem Header und einem Body und ist als XML-Dokument realisiert. Es wird dadurch versucht für alle Systeme den gleichen Sprachsatz zu definieren - für Anfragen und Antworten. Somit können Nachrichten mit beliebigem XML-Inhalt ausgetauscht werden.[18] Durch die Trennung des Bodies vom Header entsteht eine saubere Trennung zwischen den Anwendungsdaten und Webservice spezifischen Daten[19]

- **SOAP Encoding Regeln**

  Diese Regeln definieren einen Serialisierungsmechanismus, mit dem anwendungsdefinierte Datentypen zwischen den Services ausgetauscht werden können. Wobei hier anzumerken ist, dass XML eine sehr flexible Codierung von

[16] Vgl. Kiltz (2010), S. 23 ff.
[17] Simple Object Access Protocol (SOAP) 1.1 (2000).
[18] Vgl. Simple Object Access Protocol (SOAP) 1.1 (2000).
[19] Vgl. Kiltz (2010), S. 63 f.

Daten ermöglicht. Die unter dem W3C beschriebenen Kodierungsregeln können in Verbindung mit der Remote Procedure Call (RPC)-Darstellung verwendet werden.[20]

- **SOAP RPC-Darstellung**

  Eines der Entwurfsziele von SOAP besteht darin, RPC-Aufrufe unter Verwendungen der Erweiterbarkeit und Flexibilität von XML zu kapseln und auszutauschen[21]. Bei der Verwendung von SOAP durch RPCs sind folgende Transportprotokollbindungen möglich:[22]

  - TCP
  - HTTPS
  - SMTP
  - File Transfer Protocol (FTP)
  - JMS
  - Remote Method Invocation (RMI)

Ein weiterer industrieller Standard des W3C ist die Web Services Description Language (WSDL). Diese stellt ein Model und ein XML-Format zur Beschreibung der Webservices bereit. Zusammen mit SOAP entsteht eine Grundlage für Webservice Anwendungen[23]. Jedoch hat die Realität gezeigt, dass SOAP weder leicht in der Umsetzung ist, noch tatsächlichen Zugriff auf die Objekte bietet, es ist somit eher ein Kommunikationsprotokoll für Webservice Anwendungen. Durch die Bindung von unterschiedlichen Transportprotokollen, können unterschiedliche Arten von verteilten Anwendungen mit Hilfe von SOAP entwickelt werden. Das Problem bei SOAP besteht darin, dass eine SOAP Nachricht mit dem XML-Datenformat unnötig viel Overhead produziert. Für einen einfachen, beispielhaften Request an die Hochschul-App, wie `getAllLectures`,die keine Parameter und keinen Body enthalten, muss dennoch ein Body definiert werden. Bei einem kleinen Datendurchsatz stellt dies noch keine Probleme dar, jedoch wird es mit Steigerung der Daten problematisch, kompliziert und ineffizient.[24]

[20] Vgl. Simple Object Access Protocol (SOAP) 1.1 (2000).
[21] ebd.
[22] Vgl. Melzer (2007), S. 85 f.
[23] Vgl. a.a.O., S. 101 f.
[24] Vgl Kiltz (2010), S. 57 f.

Aus diesem Grund wurde im Jahr 2000 durch Roy Thomas Fielding ein vereinfachter Architekturstil namens REST eingeführt, der, wie bereits am Anfang des Kapitel erwähnt wurde, auf das JSON-Datenformat und das Kommunikationsprotokoll HTTPS basiert.[25]

5.2.2 Representational State Transfer (REST)

REST hat seinen Ursprung bereits vor den Webservices, denn es wurde damals versucht die jetzt unter REST bekannten Prinzipien als Architekturleitpfad für das Internet zu verwenden. Durch Roy Thomas Fielding wurde REST mit Webservices kombiniert und damit eine neue Revolution von der Prinzipien ausgelöst. Von nun an war REST auch unter dem Namen RESTful Webservice oder RESTful Web-API bekannt. Die Webseite *www.programmableweb.com* bietet eines der größten Archive für Web-APIs im Internet an. Im Juli 2019 hat die Webseite eine Statistik für den Wachstum des Web-API Archivs seit 2005 veröffentlicht.[26]

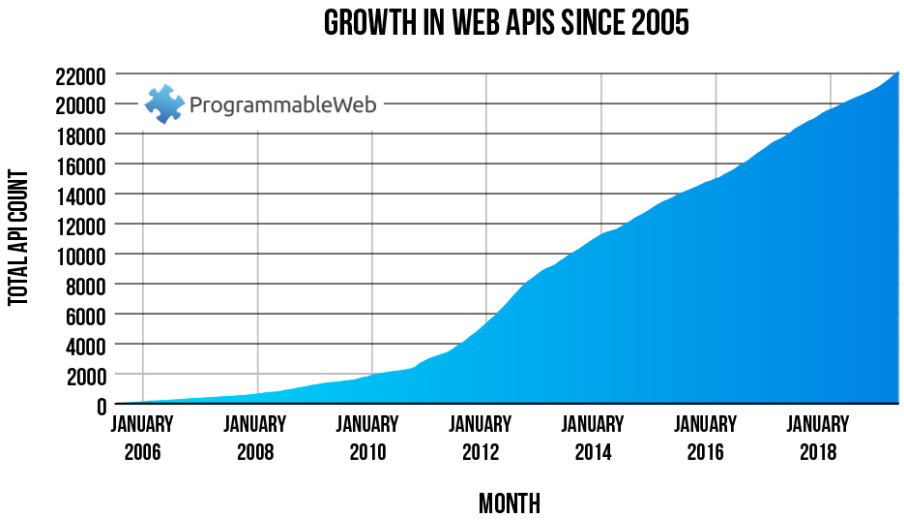


Abbildung 5.2: Wachstum der ProgrammableWeb API Archive von 2005[27]

Anhand der Statistik in Abbildung 5.2 ist deutlich zu erkennen, dass die Anzahl von Web-APIs seit 2006 kontinuierlich steigt. Das Potential von REST wurde schnell erkannt, obwohl REST immer noch - im Gegensatz zu SOAP - kein industrieller Standard, sondern eher eine Richtlinie ist, die die möglichen Nachteile von SOAP umgehen soll. Genauere Ansätze und die erwähnten Richtlinien zu REST werden noch im Kapitel 6 beschrieben.

[25] Vgl Kiltz (2010), S. 77 f.
[26] APIs show Faster Growth Rate in 2019 than Previous Years (2019).
[27] ebd.

5.2.3 Bewertung

Die folgende Entscheidungsmatrix 5.1 dient dazu, die beiden Webservice Arten gegenüberzustellen und eine Entscheidung darüber zu treffen, welche der Webservice Arten besser für die Hochschul-App geeignet ist.

Kriterien	REST	SOAP
Unterstützung leichtgewichtiger Clients	1	0
Geringere Komplexität	1	0
Effizienz	1	0
Unterstützung unterschiedlichen Datenformate	1	1
Caching-Mechanismen	1	0
Industrieller Standardisierung	0	1
Unterschiedliche Kommunikationsprotokolle	0	1
Sicherheit	1	1
Ergebnis	**6**	**4**

Tabelle 5.1: Entscheidungsmatrix

Bedeutung der Bewertungszahlen:

- **1:** trifft zu
- **0:** trifft nicht zu

REST schneidet mit 2 Punkten besser als SOAP ab. Aus diesem Grund und vor allem, da REST für leichtgewichtige Clients wie Browser oder Smartphones entwickelt wurde und auch leichter für diese zu implementieren ist, da der ganze Overhead entfällt und es zahlreiche Caching-Mechanismen unterstützt werden, geht REST als Sieger für die Entwicklung der Hochschul-App hervor. Somit stellt REST eine interessantere Grundlage für das Design der Hochschul-App dar.

Um die Entscheidungsmatrix und die aufgeführten Punkte, die für REST sprechen, zu verstehen, wird im nächsten Kapitel näher auf die beschriebenen Merkmale und Richtlinien von REST eingegangen.

6 REST API Design

Wie bereits zuvor erwähnt wurde, ist REST API kein industriell definierter Standard, weswegen es zahlreiche Varianten der Implementierung und Umsetzung gibt. Das Ziel dieses Kapitels ist es, allgemeingültige Richtlinien und Entwurfsstandards für die Erstellung einer REST-API der Hochschul-App festzulegen. Diese Standards helfen beim genaueren Erarbeiten einer APIs, die intuitiv ist und von den Entwicklern leicht verstanden und wieder- oder weiterverwendet werden kann.

Eine REST-API kann in vier Bereiche aufgeteilt werden:

1. Ressourcen
2. Requests
3. Responses
4. Hypermedia

Diese Bereiche werden in den kommenden Kapitel ausführlicher behandelt.

6.1 Ressourcen Design

Das folgende Kapitel betrifft Standards und Richtlinien für die Modellierung und Benennung von Uniform Resource Identifier (URI) Ressourcen, sowie für die Handhabung der HTTP-Methoden.

6.1.1 Versionsverwaltung

Die Versionierung ist ein wichtiger Aspekt des API-Designs. Es kann jederzeit nachvollzogen werden, wann und was sich zu einer Version geändert hat - bei einem Problem kann das System auf eine vorherige Version wiederhergestellt werden. Somit soll jede API einer Versionsnummer zugeordnet werden. Die Version kann

aus einer Hauptversion und optional aus einer Nebenversion bestehen. Hierzu gibt es verschiedene Ansätze für die Versionsverwaltung. Es können bei verschiedenen Methoden, den Request-Parametern oder bei individuellen Headern die gewünschte Version angegeben werden. Zur Vereinfachung sollte das KISS-Prinzip aus Kapitel 4.3 angewendet werden und die Version im Pfad der URI mit dem Präfix *v* und der Hauptversionsnummer angegeben werden. Bei folgenden Änderungen sollte die Version erhöht werden:

- Pflichtfeld wird als optional abgeändert oder umgekehrt
- Pflichtfeld oder optionales Feld wird eingefügt
- Ressourcen werden eingefügt, abgeändert oder entfernt
- Datentypen werden hinzugefügt, geändert, entfernt oder eingeschränkt
- Fehlercodes werden ergänzt

6.1.2 URI Namenskonvention

API-Uniform Resource Locators (URLs) sollten folgende Kriterien enthalten:

- Adresse des Servers, auf dem die API gehostet wird
- Kennzeichnung, dass es sich um einen API-Endpunkt handelt
- Spezifizierung des API-Namens
- Spezifizierung der Version der API
- Spezifizierung der Ressource
- Jede Pfadebene der Ressourcen muss sinnvolle Informationen liefern

Durch Beachtung dieser Kriterien sollte eine API-URL folgendermaßen aussehen:

Listing 6.1: URL Namenskonvention

```
1 http[s]://<server>/api/<api-name>/<api-version>/<resource>
2 https://hof-university.de/api/mensa-service/v1/menu
```

6.1.3 Ressourcen Namenskonvention

Folgende Namenskonventionen sollten für die Ressourcen in der Hochschul-App-API eingehalten werden:

- Jede Ressource muss einen aussagekräftigen und selbsterklärenden Namen haben.
- Wenn eine Ressource über Subressourcen verfügt, so werden diese durch Schrägstriche getrennt und der Pfad zur Subressourcen sollte hierarchisch und selbsterklärend aufgebaut werden.

Dies kann beispielsweise folgendermaßen aussehen:

Listing 6.2: Subressourcen Aufbau

```
1 /menu
2 /menu/{day}
3 /menu/{day}/{category}
```

Die Ressource `/menu` soll

- alle verfügbaren Informationen zum *Menü* liefern, `/menu/today` nur für den heutigen Tag und `/menu/today/desserts` nur für den heutigen Tag und nur für die Desserts,
- keine Create,Read,Update und Delete (CRUD) Operationsnamen in Ressourcen verwenden,
- keine Mischung zwischen Groß- und Kleinschreibung enthalten,
- keine Zeichen, die eine URL-Codierung erfordern, verwenden,
- Bindestrich anstelle eines Leerzeichens oder einer Unterstreichung verwenden.

6.1.4 HTTP Methoden

Nach der Identifizierung der Ressourcen für eine API [1], muss die Aktion, die für diese Ressource ausgeführt werden soll, festgelegt werden. Dafür werden die HTTP-Methoden verwendet um die ausführende Aktion anzugeben. Die primären und am häufigsten verwendete HTTP-Methoden sind POST, GET, PUT und DELETE.

[1] Siehe Kapitel 8

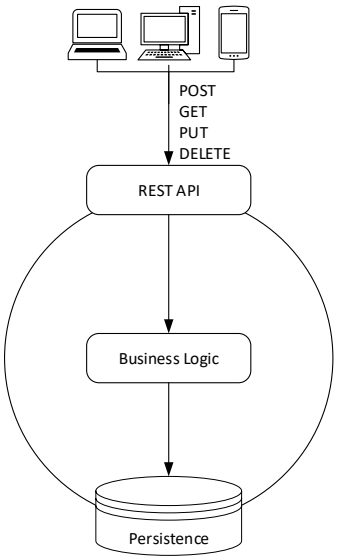


Abbildung 6.1: Einordung der HTTP-Methoden in die Gesamtarchitektur[2]

Diese Methode helfen bei der Ausführung der CRUD-Operationen für die Ressource. Die Richtlinien für die Verwendung der Methoden in der Hochschul-App-API werden in Tabelle 6.1 erläutert.[3]

Methoden	Definition
GET	Wird zum Abrufen oder Lesen der Informationen verwendet
POST	Wird zum Erstellen einer neuen Ressource verwendet
PUT	Wird zum Aktualisieren/Änderung einer ganzen Ressource verwendet
DELETE	Wird zum Löschen einer vorhandenen Ressource verwendet
PATCH	Wird zum Aktualisieren/Änderung eines teils einer Ressource verwendet
TRACE	Wird zur Verfolgung von den Request verwendet
OPTIONS	Anforderung einer Liste vorhandenen HTTP-Methoden für diese Ressource
HEAD	Ähnlich wie GET, jedoch wird nur der Response-Header zurückgegeben
CONNECT	Bereitstellung eines SSL-Tunnels von Proxyservern

Tabelle 6.1: Erläuterung der HTTP-Methoden

[2]Brysiuk, Lehmann (2019)
[3]Vgl. HTTP/Anfragemethoden (2018).

Zudem kann die Methode *TRACE* dazu dienen, die Requests auf Veränderungen während der Übertragung dieser zu überprüfen. *HEAD* bietet unter anderem die Möglichkeit, die gecachten Daten auf Ihre Gültigkeit zur prüfen, ohne die ganzen Daten komplett neu anzufordern.[4]

6.2 Request Struktur

Das folgende Kapitel behandelt die Nachtrichten-Struktur-Richtlinien für Inhalt, Format, Header und Query Parameter einer Nachricht.

6.2.1 Header

Die folgende Liste enthält HTTP-Header, die von der Hochschul-App-APIs verwendet werden sollen:

- Accept
- Authorization
- Content-Type
- X-HTTP-Method-Override

Das Payload-Format der Hochschul-App wird nur auf das Format JSON beschränkt, da JSON einige Vorteile gegenüber anderen Formaten bietet, was in Kapitel 5 bereits erwähnt wurde. Mit dem Header `accept` kann der Medientyp des erwarteten Antwortinhalts und mit dem `content-type` der Medientyp des Requests angegeben werden. Also wird der *Mediatype* als `application/json` im `content-type` und im `accept` Header definiert. Der Header `authorization` wird für die Benutzerauthentifizierung verwendet, der Header `x-api-key` hingegen für die Clientauthentifizierung.

6.2.2 Query Parameter

Query Parameter sollten im allgemeinen möglichst vermieden werden. Zuerst soll immer versucht werden die Parameter als Subressourcen in der URI hierarchisch aufzubauen. Dies erhöht die Lesbarkeit, Verständlichkeit, Flexibilität und reduziert die Komplexität der Ressourcen. Jedoch lässt es sich manchmal nicht vermeiden

[4] Vgl. HTTP/Anfragemethoden (2018).

Query Parameter zu verwenden, meistens ist dies bei der Filterung oder Sortierungen der Ressourcen der Fall. Werden Query Parameter verwendet, soll folgendes beachtet werden:

- Query Parameter sollen sich nur auf die spezifizierte Ressource beziehen.
- Komplexe oder aufwendige Query Parameter sollen im Request Body übergeben werden.

6.2.3 Aufteilung einer Ressource

Wenn eine Ressource eine große Menge von Ergebnissen zurückgeben kann, soll diese aus Performance gründen auf kleinere Teilmengen aufgeteilt werden. Folgende Richtlinien sollen bei Aufteilung einer Ressource beachtet werden:

- Die Größe der Teilmenge sollte auf einen Standardwert festgelegt werden.
- Der Standardwert kann durch den Client geändert werden.
- Das Ergebnis der Teilmenge sollten folgende Metadaten enthalten:
  - Link zur ersten Seite
  - Link zur vorherigen Seite, falls vorhanden
  - Link zur aktuellen Seite
  - Link zur nächsten Seite, falls vorhanden
  - Link zur letzten Seite
  - die Gesamtzahl der Teilmengen
- Die Seite der Teilmenge sollte mit dem Parameter *offset* gekennzeichnet werden.
- Der Parameter *offset* und der Standardwert können als Query Parameter übergeben werden.

Eine mögliche Umsetzung dieser Richtlinien könnte folgend aussehen:

Listing 6.3: Aufteilung der Ressource Vorlesung

```
{
  links: {
    first:
      "/stundenplan-service/v1/lectures?offset=0,limit=50"
    prev:
      "/stundenplan-service/v1/lectures?offset=1,limit=50"
    self:
      "/stundenplan-service/v1/lectures?offset=2,limit=50"
    next:
      "/stundenplan-service/v1/lectures?offset=3,limit=50"
    last:
      "/stundenplan-service/v1/lectures?offset=23,limit=50"
  }
}
```

6.3 Response Handling

Im folgenden werden die Antworten behandelt, welche auf einen HTTP-Request an den aufrufenden Client zurückgegeben werden.

6.3.1 Response Inhaltsformat

Das Format von Antwortinhalten ist bei der Hochschul-App auf JSON beschränkt, wie bereits im Kapitel 6.2 erwähnt wurde.

6.3.2 HTTP Statuscode

REST-APIs müssen Informationen zum Austausch mit den richtigen Statuscodes kommunizieren. Es gibt verscheiden Arten von Statuscodes, die Erfolgs- als auch Fehlerinformation mitteilen. Die Statuscodes können in vier Kategorien unterteilt werden:

- 2xx: Für Erfolgreiche Nachrichten
- 3xx: Für Umleitungen
- 4xx: Für Fehler Client-seitig
- 5xx: Für Fehler Server-seitig

In Tabelle 6.2 sind die wichtigsten Statuscodes für die Hochschul-App erläutert.

Code	Nachricht	Definition
200	OK	Die Anfrage war erfolgreich.
201	Created	Neue Ressource wurde erfolgreich angelegt.
202	Accepted	Die Anfrage wurde erfolgreich angenommen, die Informationen über die tatsächliche Ausführung der Anfrage können nicht sofort zurückgesendet werden.
400	Bad Request	Fehlerhafte Anfrage des Clients.
401	Unauthorized	Anfrage für diese Ressource benötigt Autorisierung.
403	Forbidden	Client hat keinen Zugriff auf die angeforderte Ressource.
404	Not Found	Angefragte URI wurde vom Server nicht gefunden.
405	Method Not Allowed	Die Anfrage auf die Ressource ist nicht zulässig.
409	Conflict	Die Anforderung konnte aufgrund eines Konflikts mit dem aktuellen Status der Ressource nicht verarbeitet werden.
414	Request-URI Too Large	Die Länge der angefragten URI ist länger als das zulässige Limit für den Server.
415	Unsupported Media Type	Angefordertes oder übergebenes Format wird vom Server nicht unterstützt.
429	Too Many Requests	Zu viele Anfragen wurden vom Client an die Ressource gestellt.
500	Internal Server Error	Anfrage kann aufgrund eines unerwarteten Fehlers auf dem Server nicht verarbeitet werden.
503	Service Unavailable	Die Anfrage an den Service kann vorübergehen nicht verarbeitet werden.

Tabelle 6.2: Erläuterung der HTTP-Statuscodes

Außerdem ist die Übermittlung von richtigen Statuscodes zusätzlich mit einer Fehlernachricht für die gute Nutzbarkeit der API von entscheidender Bedeutung. Durch gut definierte Statuscodes und Fehlernachrichten können Debugging und Fehlerbehebungen deutlich erleichtert werden, sowohl Client- als auch Server-seitig. Der Aufbau solcher Fehlernachrichten mit Statuscodes sieht folgendermaßen aus:

Listing 6.4: Aufbau Fehlernachricht

```
{
  status_code: {status},
  status_message: "status message",
  errors: [{
    error_code: {error code},
    error_message: "error message"
  }]
}
```

Eine API, die mehrere Aufrufe an andere APIs oder Systeme weiterleitet oder koordiniert, sollte einen einzelnen Statuscode zurückgeben, der das Ergebnis der kombinierten Ausführung von Aufrufen darstellt.

6.3.3 Asynchrone Callbacks

Es gibt zwei Standards für asynchrone Callbacks, entweder eine Callback-URL oder eine Push-Notification. Bei einer Callback-URL können die Clients eine URL der API zur Verfügung stellen, über welche die API dem Client über Ergebnisse oder Ereignisse berichten kann. Bei Push-Notifications soll der Client über die Ereignisse informiert werden, auch wenn dieser gerade nicht aktiv auf dem Service ist. Der Standard von Push-Notifications für asynchrone Callbacks wird in der Hochschul-App ebenfalls verwendet. Push-Notifications stellen eine gute Möglichkeit dar, die Studenten beispielsweise über Stundenplanänderungen in der Hochschul-App zu informieren. Weitere Informationen darüber sind im Kapitel 8.4 zu finden.

6.4 Hypermedia

Die Technologie Hypermedia As The Engine Of Application State (HATEOAS) ermöglicht es, eine API-Ressource leichter auffindbar zu machen und erleichtert die Verwendung der API-Ressourcen für den Benutzer. Dies wird dadurch ermöglicht, dass bei einem Request auf eine Ressource als Antwort Hyperlinks zu verwandten Ressourcen oder Endpunkten mit bereitgestellt werden. Für die Benutzer bedeutet das, dass keine Vorkenntnisse des Datenmodells oder Schemas erforderlich sind, um mit der API navigieren und interagieren zu können. Wenn im Client bei der Implementation der Anwendung HATEOAS berücksichtigt werden, werden die in der API-Schnittstelle abgeänderten Ressourcen automatisch in der Client-Anwendung

angepasst.

Für die Vorteile von HATEOAS in der Hochschul-App kann folgende Situation betrachtet werden. Ein Benutzer der Hochschul-App kann sich einen personalisierten Stundenplan anlegen. Dieser Stundenplan wird in einem User-Service abgespeichert, wobei nicht alle Informationen zu den Vorlesungen gespeichert werden, sondern nur deren Identification Numbers (IDs). Bei der Darstellung des personalisierten Stundenplans im Client muss dieser zuerst eine Anfrage an den User-Service stellen und mit der erhaltenen Antwort eine neue Anfrage mit den IDs aufbauen und an Stundenplan-Service senden, um die Informationen der Vorlesungen zur erhalten. Durch die Verwendung von HATEOAS stellt der Client eine Anfrage an den User-Service und in der Antwort ist ein Hyperlink enthalten, der bereits die Query für die benötigten IDs der Vorlesungen enthält.

Listing 6.5: HATEOAS Hyperlink

```
1 {
2   links: {
3     lectures: "/stundenplan-service/v1/lectures?ids=1,2,3"
4   }
5 }
```

Der Client muss nur den Hyperlink aufrufen, ohne eine aufwändige Query zu erstellen, um Vorlesungsinformationen zu erhalten. Sollte sich die Ressource `/lectures` ändern, so hat das keine Auswirkung auf die Funktionalität des Clients, da die neue Ressource wieder als Hyperlink zur Verfügung steht. Die erwähnte User-Service Ausarbeitung ist in der parallel zu dieser Arbeit entwickelten Bachelorarbeit ausführlich beschrieben.[5]

[5] Vgl. Glaser (2019a).

7 Architektur Hochschul-App

Eine Anwendung, die täglich von einem Großteil der Studierenden der Hochschule Hof genutzt wird, bedarf einer gründlichen Vorbereitung in Bezug auf deren Umsetzung und Aufbau. Um den Anforderungen der Erweiterbarkeit, Langlebigkeit und der Modularität gerecht zu werden, sollte die grundlegende Architektur der Anwendung gründlich analysiert und erarbeitet werden. Anhand der Erkenntnisse, die in den vorigen Kapiteln erarbeitet wurden, wird im folgenden der Aufbau der Architektur der Hochschul-App erläutert.

7.1 Serviceorientierte Architektur

In Kapitel 5.1 SOA wurden einige ausschlaggebende Argumente gebracht, die gezeigt haben, dass die Service-orientierte Architektur einen idealen Ansatz für die Umsetzung einer zukunftsfähigen Hochschul-App mitbringt. Jedoch bringen Service-orientierte Architekturen auch einige Nachteile mit sich. Anhand Abbildung 7.1 kann man erkennen, welche Probleme bei einer Umsetzung der Datenschnittstelle der Hochschul-App durch eine Service-orientierte Architektur auftreten können.

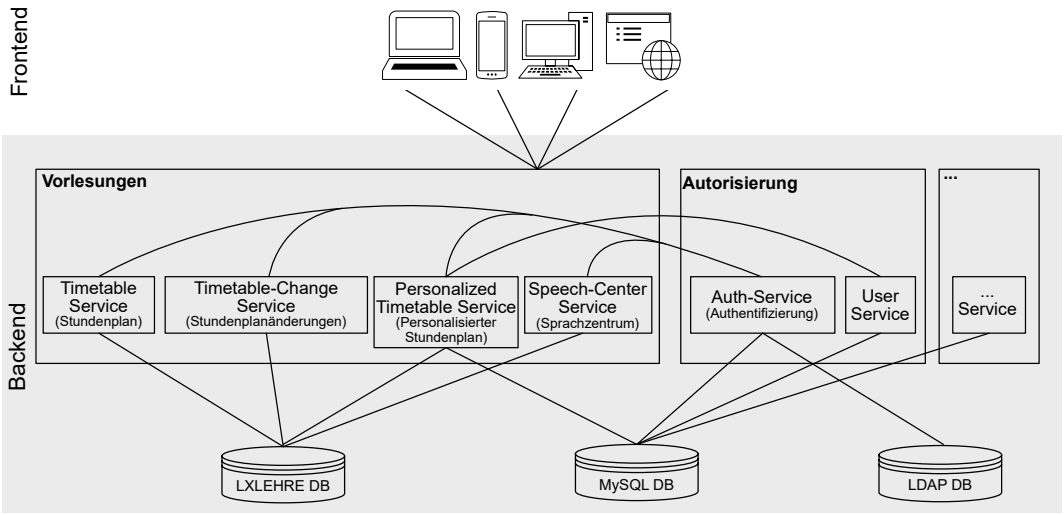


Abbildung 7.1: SOA Hochschul-App Architektur[1]

Zu erkennen ist einerseits, dass die Funktionalitäten der Anwendung in Services ausgelagert werden, die durch Module im Sourcecode getrennt sind und somit auch getrennt entwickelt werden können, jedoch nach dem Build-Prozess des Programmes in eine Komponente zusammengefasst werden. Die Entwicklung der Schnittstelle bleibt somit modular und flexibel, jedoch ist das Endprodukt nach jeder Änderung nach wie vor ein großer Monolith. Die Auslagerung und unabhängige Funktion der Services ist somit nicht gegeben, die Services können nicht nach deren Benutzungsstatistiken skaliert werden, lediglich die gesamte Anwendung könnte mehrmals parallel laufen, was aber weniger genutzte Teile der Anwendung ebenfalls hoch skalieren würde, was wiederum eine hohe Ressourcenverschwendung und hohe Kosten mit sich ziehen würde.

Andererseits ist zu erkennen, das die Service-orientierte Architektur ebenfalls eine Abhängigkeit der einzelnen Services mit sich bringt. Bei einem Request müssen die Services Informationen untereinander austauschen können. So würde ein Request an den *personalisierten Stundenplan* in der Service-orientierten Architektur die Folge haben, dass der Service zusätzlich Informationen vom *User Service* benötigen würde um den Request verarbeiten zu können. Das erschafft Abhängigkeiten, durch die einzelne Services möglicherweise nicht mehr funktionieren, wenn sie benötigte Informationen nicht zurückerhalten. Außerdem ist zu erkennen, dass die Datenbank unter den Services geteilt wird. Durch die Verwendung einer gleicher Datenbank von mehreren Services könnten sogenannte *Deadlocks* verursacht werden. Durch die bereits vorhandene Abhängigkeiten zwischen den Services wird die Ausfallwahrscheinlichkeit der Funktionalitäten durch diese *Deadlocks* weiter verstärkt. Die Skalierung der Anwendung wird das Problem nicht lösen können, sondern im Gegenteil sogar verschlechtern, denn es entstehen immer mehr Instanzen die sich die gleichen Ressourcen teilen müssen.

Um die Abhängigkeiten zu reduzieren und weitere vorhandene Probleme von SOA zu minimieren steht der Microservice Architektur Ansatz zur Verfügung.

7.2 Microservice Architektur

Die Microservice Architektur kann das Problem der Abhängigkeiten zwischen den Services bis zu einem gewissen Grat eliminieren. Im Beispiel des *personalisierten*

[1] Brysiuk, Lehmann (2019)

Stundenplans aus Kapitel 7.1 würden die benötigten Informationen aus dem *User Service* lediglich als Übergabeparameter bei der Anfrage an den Service übergeben werden. Um dies genauer zu verstehen, werden die Grundlagen der Microservice Architektur im folgenden kurz erläutert.

Microservice Konzept

Die Grundidee der Microservice Architektur kann durch das eben angebrachte Beispiel gut erklärt werden. In der SOA Architektur werden die Funktionalitäten schon einmal in einzelne Module, Services genannt, getrennt. Diese sind jedoch nur im Entwicklungsprozess getrennt, nicht im Produktionsbetrieb, denn nach dem Build-Prozess und dem Deployment auf dem Server bilden diese Services einen einzigen Monolithen, der, wie schon in Kapitel 7.1 erwähnt wurde, nur im ganzen skaliert und vervielfacht werden kann.

Microservices lösen dieses Problem, in dem sie jeden Service nicht nur als Modul im Sourcecode trennen, sondern auch im Produktionsbetrieb. Jeder Service wird somit als eigene Anwendung gepflegt und kann unabhängig von allen anderen Services laufen und Anfragen bearbeiten. Die Vorteile dieser Strategie liegen auf der Hand.

Zum einen können die einzelnen Services, in der Microservice Architektur auch Microservices genannt, einzeln skaliert werden. Somit können Teile der Anwendung, welche stark belastet werden, gut hoch skaliert werden, ohne auch die Teile der Anwendung zu skalieren, die kaum genutzt werden. Jeder Microservice kann als eigenes unabhängiges Modul auf unterschiedlichen Servern ausgelagert werden oder - wie bereits in Kapitel 5.1.2 erwähnt wurde - in Cloud-basierte-Plattformen verschoben werden oder als Docker-Container realisiert werden. Somit können die vorhandenen Ressourcen optimal genutzt werden. Zum anderen kann man die Ausfallsicherheit und die Sicherheit im allgemeinen deutlich erhöhen, wenn man nur die einzelnen Microservices redundant hält. Sollte ein Microservice ausfallen, so kann eine andere Instanz des Services einsetzen, sollte keine weitere Instanz verfügbar sein, so läuft der Rest der Anwendung trotzdem problemlos weiter, da die anderen Funktionalitäten in eigene Microservices ausgelagert werden, die komplett unabhängig voneinander laufen. Somit werden durch die Microservice Architektur einige wichtige nicht-funktionale Anforderungen wie die Skalierbarkeit, Performance und Verfügbarkeit deutlich erweitert.

Aber auch Denial of Service (DoS) und Distributed Denial of Service (DDoS) Angriffe können einfacher kompensiert werden. Sollte keine Sicherheitsmaßnahme getroffen werden, die solche Attacken frühzeitig erkennen, so würde bei einem Angriff auf einen einzelnen Microservice nur diese Funktionalität betroffen sein, die anderen Services, die unabhängig davon laufen, sind nicht gefährdet. Außerdem wird durch die erweiterte Modularisierung und Auflösung der Abhängigkeiten zwischen den Funktionalitäten das Testen, die Fehlerbehebung und die Erweiterungen der Funktionalitäten erleichtert. Die begründet sich einerseits, weil die Microservices leichtgewichtiger gestaltet sind, was die Erweiterbarkeit, Veränderbarkeit und Austauschbarkeit erleichtert, andererseits dadurch, dass durch die geringeren Abhängigkeiten die Komplexität der Funktionalitäten gemindert wird. Zusätzlich kann und sollte ebenfalls ein API-Gateway genutzt werden, welches die Anfragen auf die einzelnen Microservices kontrolliert und nochmals eine Sicherheitsbarriere darstellt, sowie die einzelnen Microservices zu einer ganzen Anwendung verbindet. Wo diese in der Anwendung einzubinden ist, zeigt Abbildung 7.2, genaueres zum API-Gateway wird noch in Kapitel 7.2 erläutert.

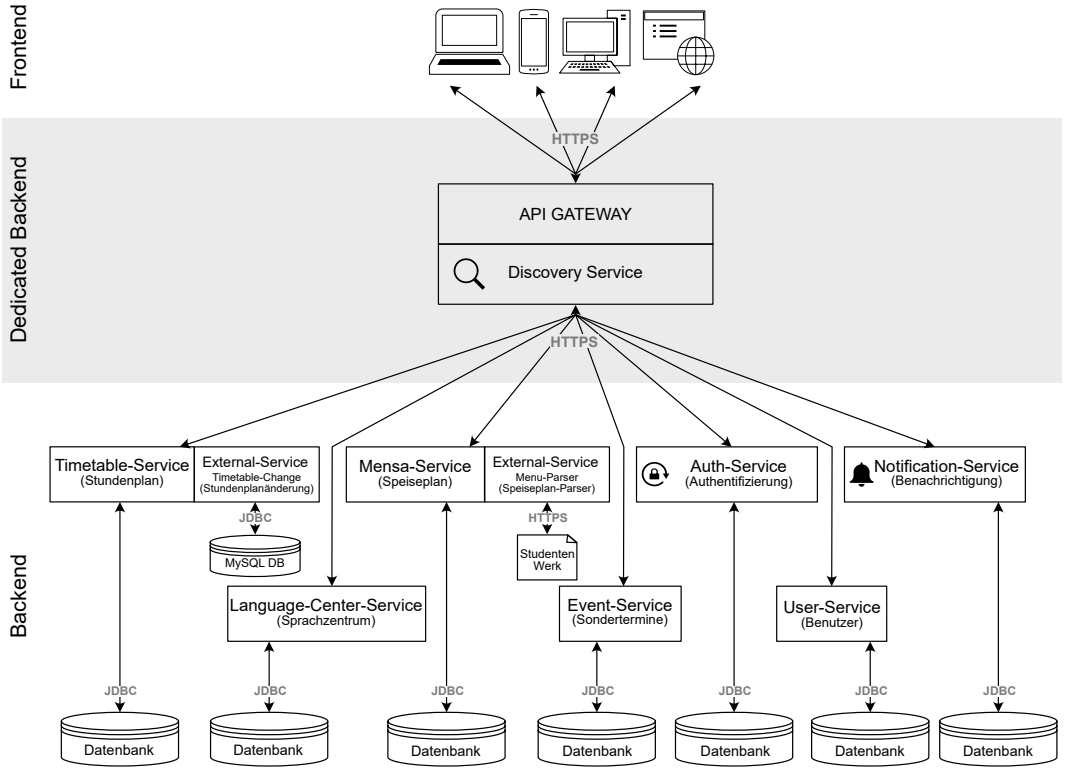


Abbildung 7.2: Microservices Hochschul-App Architektur[2]

Wie auf Abbildung 7.2 ebenfalls zu erkennen ist, wurden die einzelnen Anforderungen aus Kapitel 3 in kleine unabhängige Microservices mit jeweils einer eigenen Datenbank getrennt. Das erfordert zwar, dass man mehrere Systeme aufsetzen muss, was mehr Aufwand in der Entwicklung bedeutet, jedoch kann man die Microservices und Datenbanken an die Bedürfnisse der einzelnen Funktionen anpassen und dementsprechend optimieren. So muss zum Beispiel nicht jeder Service auf die gleiche Art von Datenbank zurückgreifen, sonder kann die Art nutzen, die seine Anforderungen am besten erfüllt. Dies kann beispielsweise eine nicht relational (NoSQL)-Datenbank statt einer klassischen relationalen Datenbank sein, was bei einer SOA-Architektur deutlich aufwändiger wäre, da dafür andere Schnittstellen in der Persistenz der Anwendung benötigt werden. Die Definition und die Spezifikation der einzelnen Microservices wird später noch in Kapitel 8 behandelt.

Discovery Service

Die Grundidee der Microservice Architektur zielt in großen Teilen auf die Skalierbarkeit und die gekapselte Zuständigkeit der einzelnen Services ab. Dieser Grundsatz beinhaltet im Wesentlichen zwei Teile. Einerseits möchte man bei der Kapselung der Zuständigkeiten der einzelnen Services erreichen, dass sowohl die Entwicklung, als auch der Produktionsbetrieb dieser Einzelteile unabhängig verlaufen können. Dabei spielen auch die physische Adresse und der Deployment-Server eine Rolle, denn einem Service, der gestartet wird, muss auch eine Adresse vergeben werden, mit er er angesprochen wird. Bei diesem Prozess spielt der *Discovery Service* eine maßgebliche Rolle. Er bietet einen Anlaufpunkt, bei dem sich alle anderen Services mit ihrer Adresse registrieren können. Wird dann ein Service benötigt, so muss man den Discovery Service nur nach der Bezeichnung des Services fragen, worauf dieser dann mit der Adresse des gesuchten Services antwortet.

Andererseits möchte man bei der Aufteilung der Services erreichen, dass die einzelnen Instanzen optimal ausgelastet werden. So kann es bei der neuen Hochschul-Anwendung zum Beispiel vorkommen, dass der Stundenplan-Service stark ausgelastet ist, da das die zentrale Aufgabe dieser App sein soll. Man kann bei Überlastungen dieses Services sehr einfach weitere Instanzen dessen auf anderen physikalischen Geräten deployen, um die Last gleichmäßig zu verteilen. Jedoch muss man nun einen Dienst einbinden, welcher die Adressen dieser Instanzen kennt und auch im Auge behält, welcher Dienst wie stark ausgelastet ist. Auch diese Aufga-

[2] Brysiuk, Lehmann (2019)

be übernimmt der Discovery Service. Da sich alle Instanzen der Services mit ihrer Bezeichnung bei dem Discovery Service registrieren kann dieser die verschiedenen Instanzen des gleichen Services gruppieren und die Anfragen gleichmäßig verteilen. Der Client ruft dafür nur die Bezeichnung des Services beim Discovery Service ab und bekommt darauf die Adresse des Services, der diese Aufgabe erledigt und der aktuell am wenigsten ausgelastet ist.

Um diesen Discovery Service sinnvoll nutzen zu können, benötigt die Anwendung ein API-Gateway, welches sich ebenfalls als Service beim Discovery Service registriert. Auch für dieses Gateway gilt, dass sich mehrere Instanzen registrieren können. Auch dieser Teil der Anwendung kann somit skaliert werden. Das API-Gateway ruft dann für einen Client die Services beim Discovery Service ab. Die genaue Funktion des Gateways werden im Kapitel 7.2 *API-Gateway* betrachtet.

API-Gateway

Als Verbindungsstück zwischen den Services und ihrer einzelnen Instanzen fungiert der im vorigen Kapitel 7.2 *Discovery Service* beschriebene Service. Der große Vorteil dessen ist, dass man lediglich die Adresse des Discovery Services benötigt, um dann mit allen anderen Microservices zu kommunizieren. Um einem externen Client jedoch die Möglichkeit zu geben, die Vorteile dessen nutzen zu können, benötigt es ein API-Gateway, bei dem der Nutzer die gewünschten Ressourcen anfragt und das sich dann um die Beschaffung dieser kümmert. Somit muss auch der Client nicht mehr jede Adresse der Services kennen, sondern nur die desAPI-Gateways. Dies führt einerseits zu sauberem Code, da eine echte Adresse nicht mehr fest im Programmcode eingegeben werden muss, andererseits auch zu einer Möglichkeit, den Zugriff auf die Funktionen des Discovery Services und alle darunter liegenden Microservices zu kontrollieren.

Die Funktion des API-Gateways und der Zusammenarbeit des Discovery Services mit diesem werden in Abbildung 7.3 gezeigt.

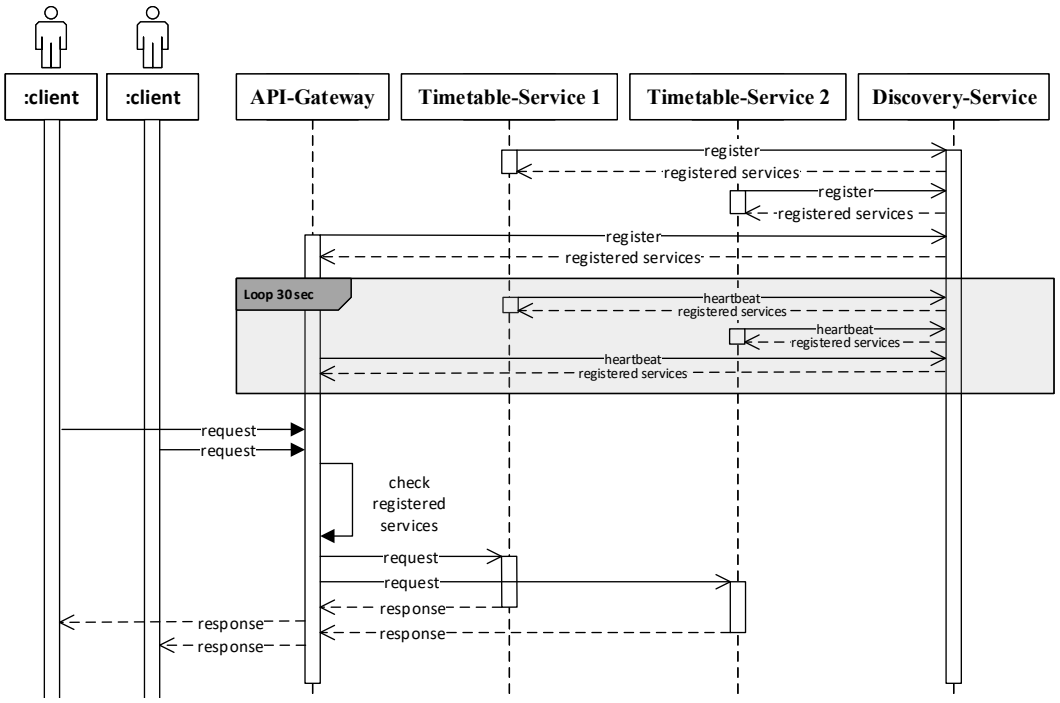


Abbildung 7.3: Sequenzdiagramm zwischen Discovery, API-GATEWAY und Services[3]

Wie zu erkennen ist, registrieren sich alle Services, das API-Gateway eingeschlossen, beim Discovery Service, welcher diesen dann eine Liste der aktuell registrierten Services und deren Adressen zurückgibt. Diese Liste wird in jedem Service gespeichert. Nach der erfolgreichen Registrierung melden sich die Services dann alle 30 Sekunden wieder beim Discovery Service, der dann wieder eine aktualisiert Liste zurückgibt. Dieses wiederholte Melden und Aktualisieren nennt man den *Heartbeat* der registrierten Services. Sollte dieser Heartbeat einmal ausfallen, so entfernt der Discovery Service den Service, der sich nicht mehr meldet, automatisch nach einem definiertem Threshold an Fehlversuchen.

Sobald das API-Gateway sich beim Discovery Service registriert hat, können die Clients erfolgreiche Anfragen an dieses schicken. Je nach Anfrage überprüft das Gateway dann, welcher passende Service bei ihm in der Liste, die er vom Discovery Service erhalten hat, enthalten ist und spricht diesen dann direkt mit der hinterlegten Adresse an. Die Antwort bekommt dieser darauf direkt vom angesprochenen Service, die Kommunikation läuft somit direkt zwischen den betroffenen Services, was die Performanz deutlich steigert. Lediglich die Registrierung und der Austausch der Adressen läuft parallel dazu im Hintergrund über den Discovery Service.

[3] Brysiuk, Lehmann (2019)

7.3 Schichtenarchitektur

Wie bereit in Kapitel 4.3 beschrieben wurde, soll die Architektur in Schichten aufgeteilt sein, um die Zuständigkeiten der einzelnen Funktionalitäten des Softwaresystems zu trennen und modular zu halten. Im Falle der Web-basierten Hochschul-App trifft das auf die Architekturen der einzelnen Microservices zu. Diese werden, wie in Abbildung 7.4 gezeigt, in 4 Schichten aufgeteilt, von denen eine nochmals physisch ausgelagert werden kann.

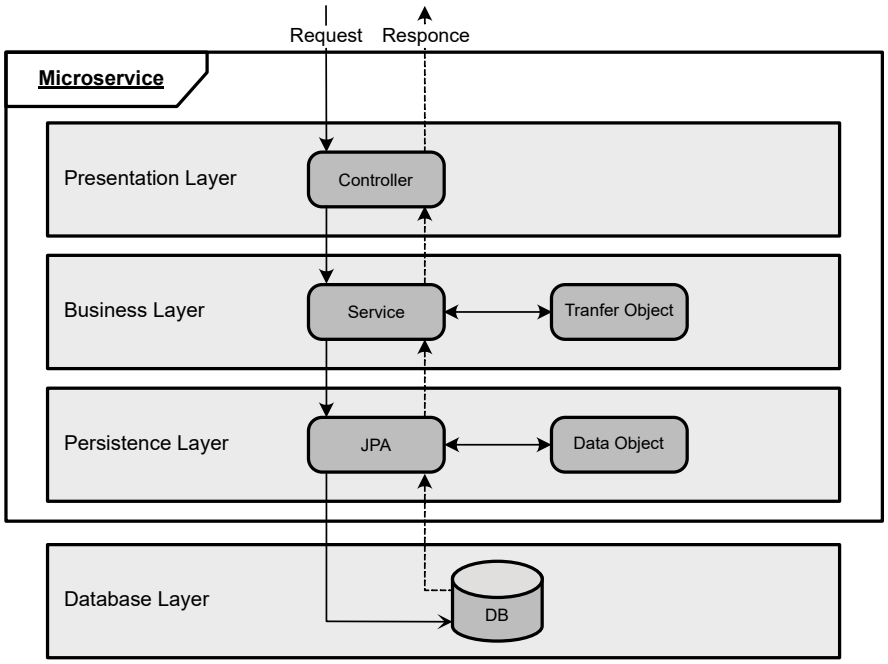


Abbildung 7.4: Microservice in Schichten Aufgeteilt[4]

Konkret übernehmen die Schichten folgende Aufgaben:

- **Presentation Layer**
  Wie der Name der Schicht schon sagt, ist diese Schicht für die äußerliche Darstellung der Anwendung zuständig. Im Falle einer REST-Schnittstelle sind das die Endpunkte, über die ein Client Daten abfragen kann. Diese Schicht ist nur für die Schnittstelle zuständig, sie definiert also die URLs für die Endpunkte und sichert die Anwendung nach außen ab. Zudem leitet sie die Anfragen an die darunterliegende Schicht, die Business Schicht, weiter.

- **Business Layer**
  Die Business Schicht übernimmt die eigentliche Aufgabe, für die die Anwen-

[4]Brysiuk, Lehmann (2019)

dung geschrieben wurde. Sie enthält also die Kernlogik des Systems. Alle Algorithmen und Datenverarbeitungen werden in dieser Schicht durchgeführt. Benötigt eine dieser Vorgänge Daten, so werden diese aus der darunter liegenden Schicht, der Persistenz Schicht, abgefragt.

- **Persistence Layer**
  Die Persistenz Schicht hat im Grunde nur zwei Aufgaben, Daten auszulesen und Daten zu speichern. Um die Datenquellen vor der Business Schicht zu verstecken und den Zugriff für alle Quellen zu vereinheitlichen bietet diese Schicht eine einheitliche Schnittstelle für die Daten aus allen Datenquellen. Wenn die Business Schicht die gelesenen Daten ändern möchte oder neue Daten einspeisen will, so kann sie das auch über die Persistenz Schicht tun. In diesem Fall gilt das gleiche Prinzip, wie beim Auslesen. Die Schnittstelle wird für alle Datenquellen angeglichen. So muss beim Austausch der Datenquellen nur die Persistenz Schicht, nie eine der darüber liegenden Schichten angepasst werden.

- **Database Layer**
  Diese Schicht ist im Rahmen der Schichtenarchitektur nicht zwingend notwendig, allerdings wird sie im Großteil der Services der Hochschul-App angewendet. Sie bildet die Grundlage der Datenschnittstelle der Services, die Datenbank. Die Datenbank ist kein Teil des eigentlichen Sourcecodes der Anwendung und kann somit separat zur Anwendung gepflegt werden. Jedoch kann die Anwendung ohne die Datenbank keine Daten bereitstellen, somit ist die Datenbank Schicht für die Hochschul-App unverzichtbar.

Ablauf einer Anfrage

Eine Anfrage eines Clients durchläuft im allgemeinen immer den selben Ablauf. Die Anfrage selbst wird an einen Endpunkt gerichtet, der durch eine URL definiert wurde. Die Präsentations Schicht greift diese Anfrage auf und untersucht, ob der Client, der die Anfrage gestellt hat, die nötigen Berechtigungen für den Zugriff hat.

Trifft das zu, so wird die Anfrage an die Business Schicht weitergeleitet, welche sie dann verarbeitet. Benötigt die Business Schicht zum Bearbeiten der Anfrage weitere Daten, so fragt sie diese in der Persistenz Schicht ab. Die Persistenz Schicht kann dann unterscheiden, um welche Art von Daten es sich handelt und wo diese zu finden sind. Sind die Daten aus der Datenbank auszulesen, so werden diese in der

Datenbank Schicht abgefragt. Es kann sich aber auch um Daten handeln, welche von einem externen Anbieter geholt werden müssen. In diesem Fall schickt die Persistenz Schicht eine Anfrage an die Schnittstelle des externen Anbieters und liefert dann die Daten an die Business Schicht zurück. Die Daten werden zwischen der Persistenz Schicht und der Business Schicht immer in sogenannten *Data Objects*, manchmal auch *Business Entities* genannt, gekapselt. In der Programmiersprache Java werden diese Wrapper Objekte typischerweise mit dem Namen der Datenart, gefolgt von den Buchstaben *Data Object (DO)* benannt.

Die Business Schicht verarbeitet die Daten dann dementsprechend. Die veränderten Daten können dort auch wieder an die Persistenz Schicht weitergegeben werden, wo sie dann wieder in der zugehörigen Datenquelle abgespeichert werden. Im Falle der Hochschul-App werden die Daten wieder in der Datenbank persistiert. Die Daten, die die Präsentations Schicht zur Beantwortung der Anfrage des Client benötigt, werden dann wieder von der Business Schicht an die Präsentations Schicht zurückgegeben. Diese Daten werden in *Transfer Objects* weitergegeben. Hier werden die Wrapper Klassen in Java typischerweise mit dem Namen gefolgt von den Buchstaben *Transfer Object (TO)* benannt.

Die Antwort der Business Schicht an die Präsentations Schicht hängt immer von der Anfrage ab. Es können sowohl echte Daten, als auch nur Erfolgs-/ Misserfolgsbenachrichtigungen zurückgeliefert werden. Sollten Daten zurückgeliefert werden, so immer in Form von *Transfer Objects*. Fehlermeldungen werden in in Java oft als Exceptions zurückgeliefert, in anderen Programmiersprachen mit ähnlichen Konstrukten. Erfolge werden meist durch Boolesche Werte oder durch das zurückkehren ohne Fehlermeldung signalisiert. Die empfangenen Rückgabewerte werden dann von der Präsentations Schicht an den Client zurückgegeben.

Die Unterscheidung von *Transfer Objects* und *Data Objects* ist deshalb nötig, da in der Business Schicht oft Daten aus der Persistenz Schicht benötigt werden, die der Client entweder nicht einsehen darf, oder die er nicht verarbeiten kann. Deshalb werden die Daten, die über die Schnittstelle der Präsentations Schicht an den Client gehen, immer so bearbeitet, dass sie in der richtigen Form sind und keine sensiblen Daten mehr enthalten.

Aufbau der anwendungsinternen Schichten

In Abbildung 7.4 werden die groben Schichten der kompletten Software Architektur der Services und ihre Aufgaben betrachtet. Abbildung 7.5 hingegen betrachtet den Aufbau der drei oberen Schichten, die im Sourcecode der Anwendung liegen. Die Datenbank Schicht wird hierbei nicht weiter betrachtet.

Package Struktur

Ähnlich, wie die Schichten der Anwendung, wird auch der Programmcode unterteilt. Wie bereits in Kapitel 7.3 beschrieben wurde, sind die oberen drei Schichten auch im Sourcecode der Hochschul-App zu finden. Die Package Struktur dieser richtet sich nach den drei Schichten, so gibt es ein Presentation Package, ein Business Package und ein Persistence Package.

Das Controller Package ist, wie zu erahnen ist, für die Endpunkte der Anwendung zuständig. In Microservice Anwendungen nennt man die zugehörigen Klassen oder Handler oft Controller, da es REST-Schnittstellen sind, auch REST-Controller. Demnach richtet sich auch die Namenskonvention. Den Anfang des Namens macht der Name der Schnittstelle, in Abbildung 7.5 beispielhaft *Test* genannt, darauf folgend wird das Wort *Controller* angehängt. Für jeden Teil der Schnittstelle kann man einen eigenen Controller definieren, so wird der Code auch übersichtlicher.

Das Service Package enthält alle Algorithmen zur Datenverarbeitung. Auch hier ist die Namenskonvention ähnlich wie im Controller Package aufgebaut, jedoch nennt man die zugehörigen Klassen oder Handler hier *Services*. Der Namen beginnt auch hier mit dem Namen des Services und endet mit *Service*. Zusätzlich zu den Services enthält dieses Package auch die in Kapitel 7.3 erwähnten *Transfer Objects*. Diese werden im untergeordneten Package *TransferObject* definiert. Die zugehörige Namenskonvention wurde bereits erläutert.

Das letzte Package, das sich nach der allgemeinen Schichtenarchitektur richtet, ist das Repository Package. Hier werden die Daten aus den verschiedenen Datenquellen gesammelt. Die Namenskonvention ist hier unterschiedlich, je nachdem, um welche Art von Datenquelle es sich handelt. Bei nativen Structured Query Language (SQL)-Statements werden die zuständigen Klassen oder Handler oft *Database-Accessor* oder *[Name]Accessor* genannt. Bei externen Schnittstellen wird oft der Name gefolgt von der Abkürzung *API* genutzt. Bei der Nutzung der Java Persis-

tence API (JPA) wir der Name der Datenquelle gefolgt von der Abkürzung *JPA* verwendet. Im Repository Package werden, wie in der Persistenz Schicht definiert wurde, auch die *Data Objects* in einem untergeordneten Package gelagert. Auch für diese wurde die Namenskonvention bereits dargestellt.

Um die Aufgaben des Services von anderen Hilfsbibliotheken oder Prozeduren zu trennen benötigt die Package Struktur der Hochschul-App und von Microservices im Allgemeinen noch weitere Packages für externes. Darunter gehören zum Beispiel externe Services, die man nicht ganz von der Aufgabe des Microservices trennen kann, die aber auch nicht direkt dazugehören. Ein Beispiel dafür ist der Zugriff auf eine externe Schnittstelle eines anderen Anbieters. Der Zugriff an sich wird dabei von einer externen Bibliothek durchgeführt. Ein weiteres Package ist das *Util* Package. Dieses beinhaltet Hilfsbibliotheken und Klassen. Ein Beispiel hierfür sind Datenparser, welche gelesene Daten in ein einheitliches Format bringen.

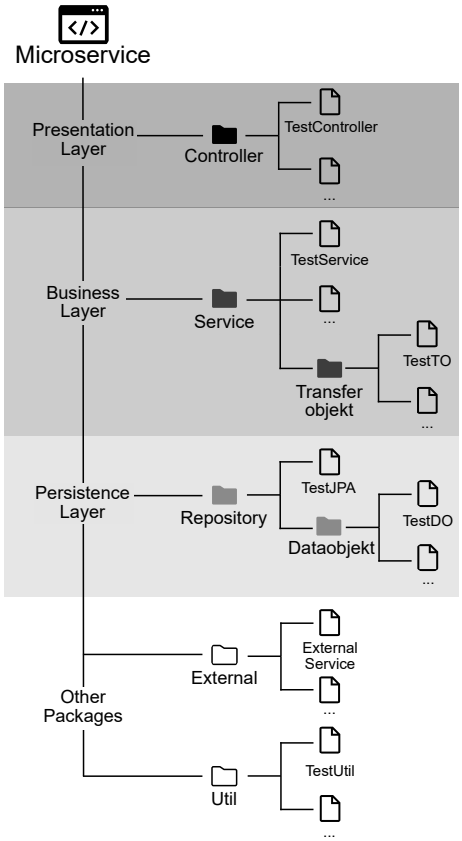


Abbildung 7.5: Microservice Schichtenarchitektur[5]

[5] Brysiuk, Lehmann (2019)

8 Services der Hochschul-App

Wie im vorhergehenden Kapitel 7 bereits beschrieben wurde, basiert eine Microservice Architektur darauf, dass man einzelne Funktionalitäten der Anwendung in einzelne Services aufteilt. Die allgemeinen Richtlinien bei der Erstellung einer Web Service Architektur wurden ebenfalls schon im Kapitel 5 zum Thema Web Services beschrieben.

Im Allgemeinen werden die Services der Datenschnittstelle der neuen Hochschul App anhand der funktionalen Anforderungen und der architekturbedingten Vorgaben aufgeteilt. Die Services für den Stundenplan, die Stundenplanänderungen, den Mensaplan, die Benachrichtigungen und der Authentifikation bilden einzelne Services, die sich nach der entsprechenden funktionalen Anforderung richten. Die Namensgebung beschreibt hier in dem Fall schon eindeutig, welche Aufgaben jeder einzelne Service erfüllt. Die eindeutige Namensgebung trifft in diesem Fall auch auf den Anwenderservice zu, welcher sich jedoch weniger nach den funktionalen Anforderungen richtet, sondern vielmehr an den technischen Vorgaben der Microservice Architektur. In diesem Fall wird eine Funktion der Anwendung so ausgelagert, dass keine der anderen Services von diesem einen Service abhängig sind. Dieser Service ermöglicht es dennoch, die Funktion der Anwenderverwaltung zu kapseln und somit redundanten Programmcode weitgehend zu vermeiden. Die genauen Anforderungen an die einzelnen Services, deren Funktionsweise und Aufbau, sowie die grundlegende Struktur der dahinter liegenden Daten werden im folgenden genauer erarbeitet.

8.1 Stundenplan (Timetable-Service)

Die wichtigste Funktion der Hochschul-App soll es sein, den Stundenplan leichter für mobile Endgeräte darzustellen. Wie bereits in den vorhandenen Anwendungen der Hochschule Hof baut auf dieser Funktion die komplette Anwendung auf. Demnach ist es wichtig, dass die Erarbeitung dieser zentralen funktionalen Anforderung

unter höchster Sorgfalt geschieht. Diese beinhaltet nicht nur die Bereitstellung des allgemeinen Stundenplans zum Verlesungsangebot der Hochschule, sondern auch die Möglichkeit, personalisierte Anfragen an die Datenquelle zu stellen, die in einer grafischen Oberfläche ebenfalls in einen personalisierten Stundenplan umgewandelt werden können.

Anforderung an Schnittstelle

Die Daten zum Vorlesungsangebot der Hochschule Hof werden separat zur Anwendung in einer Datenbank gehalten und gepflegt. Zur Nutzung dieser und zum Arbeiten mit den Daten werden diese allerdings in modifizierter Form in einer eigenen Datenbank abgelegt. Die Schnittstelle des Services muss dem Client alle relevanten Daten zu den einzelnen Vorlesungen geben. Diese Vorlesungen kann der Client einzeln oder in verschiedenen Listen abfragen.

Jede einzelne Vorlesung muss mindestens folgende Informationen liefern:

- Eine eindeutige Kennung der Vorlesung (ID)
- Der Name der Vorlesung
- Die Art der Vorlesung
- Dozent der Veranstaltung
- Start und Ende der Vorlesung (speziell für Block-/ Sonderveranstaltungen)
- Vorlesungszeiten
- Die Studiengänge, in denen die Vorlesung angeboten wird
- Fachsemester der Vorlesung
- Vorlesungsraum

Die Daten lassen sich in allgemeine Informationen zur Veranstaltung, zeitliche Einschränkungen und in Informationen zum Rahmen der Vorlesung aufteilen. Beim Abfragen der Daten muss der Client die Möglichkeit haben, eine Liste von Veranstaltungen anhand von verschiedenen Suchkriterien zu erhalten. Möchte ein möglicher Nutzer demnach seinen aktuellen Stundenplan abfragen, so muss der Client eine Anfrage stellen können, die den Studiengang und das Fachsemester enthält.

Die Antwort muss dann das komplette Vorlesungsangebot für diesen Nutzer enthalten. Ähnlich verhält es sich bei Suchen nach speziellen Vorlesungen. Es muss möglich sein, eine Liste an Vorlesungen zu erhalten, die gewisse Suchkriterien erfüllen. Möchte der Client demnach eine Liste aller Vorlesungen eines Dozenten haben, oder alle Veranstaltungen zu einem gewissen Zeitpunkt abfragen, so muss er das tun, indem er diese Suchkriterien in der Anfrage angibt.

Anders als beim personalisierten Stundenplan gibt der Client bei einem allgemeinen Stundenplan eine Anfrage ab, bei der er nicht weiß, welche und wie viele Ergebnisse er zurückerhält. Möchte der Client allerdings die in einem personalisierten Stundenplan abgelegten Vorlesungen abfragen, so muss er eine Anfrage stellen, in der alle eindeutigen Kennzeichnungen der Vorlesungen enthalten sind. Solang eine Kennung auch existiert, muss die Schnittstelle die zugehörigen Daten auch zurückgeben.

Spezifikation der Ressourcen

In den GET-Methoden der Subresource /room/{room} können im Body die Filter für Uhrzeit, Datum oder Wochentag übergeben werden. Für die restlichen Subresourcen, außer /{id} und /{name}, können Filter für Dozent, Uhrzeit, Vorlesungssprache, Art der Vorlesung und Datum oder Wochentag übergeben werden. Die Filter Parameter sind optional, ohne eine Übergabe der Filter werden standardmäßig alle Vorlesungen zurückgegeben.

- **GET/POST/DELETE:**
  /lectures

  Die Methode GET liefert alle Vorlesungen unter der Berücksichtigung von gesetzten Filtern. POST erlaubt erstellen von einer oder mehreren Vorlesungen und DELETE löscht alle Vorlesungen.

- **GET/PATCH/DELETE:**
  /lectures/{id}

  Die Methode GET liefert eine Vorlesung für die übergebende ID. Mit PATCH kann die Vorlesung geändert beziehungsweise korrigiert werden und DELETE löscht diese Vorlesung.

- **GET:**
  /lectures/{lecture}

  Liefert eine Vorlesung unter der Berücksichtigung von übergebenen Vorlesungsbezeichnung.

- **GET/POST/DELETE:**
  /lectures/faculty

  Die Methode GET liefert eine Liste vorhandener Fakultäten. POST erlaubt die Fakultätsliste zu erweitern und DELETE löscht die komplette Fakultätsliste.

- **GET/PATCH/DELETE:**
  /lectures/faculty/{faculty}

  Die Methode GET liefert alle Vorlesungen die zur der übergebende Fakultät gehören. Mit PATCH kann die Fakultät geändert beziehungsweise korrigiert werden und DELETE löscht diese Fakultät aus der Fakultätsliste.

- **GET/POST/DELETE:**
  /lectures/faculty/{faculty}/program

  Die Methode GET liefert eine Liste vorhandener Studiengänge. POST erlaubt die Studiengangliste zu erweitern und DELETE löscht die komplette Studiengangliste.

- **GET/PATCH/DELETE:**
  /lectures/faculty/{faculty}/program/{program}

  Die Methode GET liefert alle Vorlesungen die zur gesetzten Fakultät und Studiengang gehören. Mit PATCH kann der Studiengang geändert beziehungsweise korrigiert werden und DELETE löscht diesen Studiengang aus der Studiengangliste.

- **GET/POST/DELETE:**
  /lectures/faculty/{faculty}/program/{program}/semester

  Die Methode GET liefert eine Liste vorhandener Semester. POST erlaubt die Semesterliste zu erweitern und DELETE löscht die komplette Semesterliste.

- **GET/PATCH/DELETE:**
  /lectures/faculty/{faculty}/program/{course}/program/{semester}

  Die Methode GET liefert alle Vorlesung die zur der gesetzten Fakultät, Studiengang und Semester gehören. Mit PATCH kann der Semester geändert beziehungsweise korrigiert werden und DELETE löscht diesen Semester aus der Semesterliste.

- **GET:**
  /lectures/room

  Liefert eine Liste vorhandener Vorlesungsräume.

- **GET:**
  /lectures/room/{room}

  Liefert alle Vorlesungen die zur gesetzten Raum gehören unter der Berücksichtigung gesetzter Filterkriterien.

- **GET:**
  /lectures/filters

  Liefert alle vorhandenen Filterkriterien wie Dozenten, Vorlesungssprache, Vorlesungsart, Uhrzeit, Datum und Wochentag.

- **GET/POST/DELETE:**
  /lectures/filters/instructor

  Die Methode GET liefert eine Liste aller Dozenten. Mit POST können ein oder mehrere neue Dozenten eingefügt werden und DELETE löscht die komplette Dozentenliste.

- **GET/PATCH/DELETE:**
  /lectures/filters/instructor/{id}

  Die Methode GET liefert Informationen zur der übergebenen Dozenten ID. Mit PATCH kann die Information zum dem Dozent geändert beziehungsweise korrigiert werden und DELETE löscht diesen Dozenten.

- **GET:**
  /lectures/filters/time

  Liefert die Formatierung für den Filter Uhrzeit.

- **GET/POST/DELETE:**
  /lectures/filters/language

  Die Methode GET liefert eine Liste verfügbarer Vorlesungssprachen. Mit POST können ein oder mehrere Vorlesungssprachen eingefügt werden und DELETE löscht die komplette Vorlesungssprachenliste.

- **GET/PATCH/DELETE:**
  /lectures/filters/language/{id}

  Die Methode GET liefert Informationen zur der übergebenen Vorlesungssprachen ID. Mit PATCH kann die Vorlesungssprache geändert beziehungsweise korrigiert werden und DELETE löscht diese Vorlesungssprache.

- **GET:**
  /lectures/filters/type

  Liefert eine Liste vorhandener Vorlesungstypen.

- **GET:**
  /lectures/filters/date

  Liefert die Formatierung für den Filter Datum.

- **GET:**
  /lectures/filters/day

  Liefert eine Liste vorhandener Wochentagen und den Format für den Filter.

Datenbank Konzept

Wie bereits erwähnt liegt für die Stundenplan Daten und die zugehörigen Informationen zu Änderungen und Fremdschlüsseltabellen bereits eine Datenbank zur Verfügung. Diese läuft aktuell auf dem Lehre Server der Hochschule Hof.

Dort ist eine MySQL Installation vorzufinden, auf der die Datenbank *huapp_time_table* liegt. Jedoch gibt es bei der original Datenbank einige Designspezifikationen, welche das Arbeiten mit dieser für die Stundenplan Funktion schwierig macht. Das ausschlaggebende Problem, welches zur Nutzung einer eigenen Datenbank geführt hat, ist die Neugenerierung der Primärschlüssel, wenn ein neuer Eintrag geschrieben wurde. Dies macht die Abfrage durch außerhalb gespeicherte IDs unsicher, da diese ID durch die Neuvergebung des Primärschlüssels eventuell nicht vergeben oder einem anderen Eintrag vergeben wurde. Deshalb liegt der Fokus der selbst entwickelten Datenbank darauf, die Einträge der Vorlesungen eindeutig zu halten, auch wenn sich eine Vorlesung dauerhaft ändert. Die Herausforderung dabei ist, dass man erkennen muss, wenn sich ein Eintrag geändert hat.

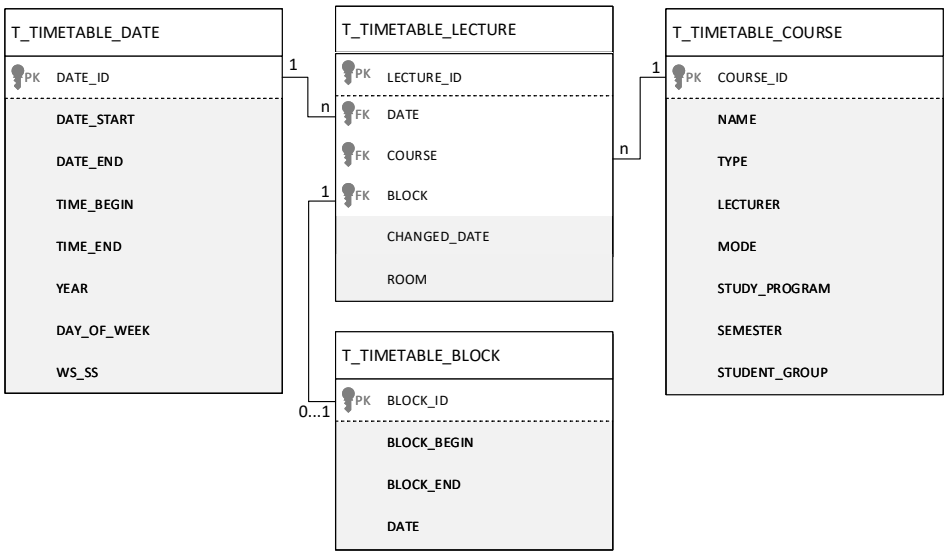


Abbildung 8.1: Entity Relation Diagramm der Stundenplan Datenbank[1]

8.2 Planänderungen (Timetable-Change-Service)

Zusätzlich zu der zentralen Aufgabe des allgemeinen und personalisierten Stundenplans muss die Anwendung auch die Änderungen am Stundenplan für den Nutzer zur Verfügung stellen. Dies muss auf verschiedenen Wegen möglich sein. Die Änderungen müssen einerseits frei aufrufbar sein, das heißt, nach einer Anfrage an die Schnittstelle zurückgegeben werden, als neue Änderung aber auch von außen in die Datenquelle eingespeist werden. Um die Benachrichtigung der Clients der Schnittstelle ist dann ein anderer Service zuständig, der seine Informationen jedoch vom Stundenplanänderung-Service bezieht.

Eine normale Anfrage an die vorhandenen Änderungen kann nochmals in zwei Arten unterteilt werden. Einerseits muss die Schnittstelle alle Daten zurückgeben können, die sie enthält, andererseits soll eine Anfrage auch gefilterte Ergebnisse zurückerhalten können. So kann ein registrierter Nutzer beispielsweise alle Änderungen abfragen, die seine Vorlesungen betreffen oder die sein Fachsemester in seinem Studiengang betreffen.

Die Daten, die die Anwendung auf eine solche Anfrage zurückgibt, muss mindestens folgende Informationen enthalten:

- Kennzeichnung der Vorlesung (Kann bei der Zuordnung zu einer echten Vorlesung genutzt werden)
- Art der Änderung (Ausfall, Verschiebung, etc.)
- Zeitpunkt der Änderung (Datum, Uhrzeit)

Anhand dieser Informationen ist nicht nur die allgemeine Vorlesung identifizierbar, sondern auch der Termin, an dem die Änderung eintrifft. Dies ist besonders bei Verschiebungen oder Ausfällen von einzelnen Veranstaltungen wichtig.

Anforderung an Schnittstelle

Der Service und die Schnittstelle für die Stundenplanänderungen ist im Vergleich zum Stundenplan Service relativ leichtgewichtig. Hier wird lediglich eine Schnittstelle benötigt, welche die Änderungen bereitstellt. Diese sollten entweder alle ausgegeben oder anhand von Kriterien wie Datum, Studiengang oder ähnlichem gefil-

[1] Brysiuk, Lehmann (2019)

tert werden. Zudem soll es möglich sein, neue Änderungen über die REST-Schnittstelle hinzuzufügen, da im Hintergrund des Services eine eigene Datenbank stehen soll. Zudem muss der Service den Benachrichtigungsservice über neue Änderungen informieren, sobald er diese erkennt.

Spezifikation der Ressourcen

In der Ressource `/changes` und allen Subressourcen außer `/{id}` und `/filters` kann im Body der Filter für das Datum übergeben werden.

- **GET/POST/DELETE:**
  /changes

  Die Methode GET liefert alle Änderungen unter der Berücksichtigung vom gesetzten Datum. POST erlaubt erstellen von eine oder mehreren Änderungen und DELETE löscht alle Änderungen.

- **GET/PATCH/DELETE:**
  /changes/{id}

  Die Methode GET liefert eine Änderung für die übergebene ID. Mit PATCH kann die Vorlesung Änderung geändert beziehungsweise korrigiert werden und DELETE löscht diese Änderung

- **GET/POST/DELETE:**
  /changes/program

  Die Methode GET liefert eine Liste vorhandener Studiengänge. POST erlaubt die Studiengangliste zu erweitern und DELETE löscht die komplette Studiengangliste.

- **GET/PATCH/DELETE:**
  /changes/program/{program}

  Die Methode GET liefert alle Änderungen die zur gesetzten Studiengang gehören. Mit PATCH kann der Studiengang geändert beziehungsweise korrigiert werden und DELETE löscht diesen Studiengang aus der Studiengangliste.

- **GET/POST/DELETE:**
  /changes/program/{program}/semester

  Die Methode GET liefert eine Liste vorhandener Semester. POST erlaubt die Semesterliste zu erweitern und DELETE löscht die komplette Semesterliste.

- **GET/PATCH/DELETE:**
  /changes/program/{program}/semester/{semester}

  Die Methode GET liefert alle Änderungen die zu gesetzten Studiengang und Semester gehören. Mit PATCH kann der Semester geändert beziehungsweise korrigiert werden und DELETE löscht diesen Semester aus der Liste.

- **GET:**
  /changes/filters

  Liefert alle vorhandenen Filterkriterien, in dem Fall nur das Datum.

- **GET:**
  /changes/filters/date

  Liefert die Formatierung für den Filter Datum.

Datenbank Konzept

Die Stundenplanänderungen liegen, wie die Stundenplaninformationen, ebenfalls bereits auf dem Lehre Server der Hochschule Hof in der Datenbank *huapp_time_table*. Dort existiert eine Tabelle, in der die Veranstaltungen, die sich ändern, mit den alten Daten und den neuen Daten abgelegt werden. Jedoch werden diese Änderungen nur mit den Namen der dazugehörigen Vorlesung und den zeitlichen Daten der ursprünglichen Veranstaltung hinterlegt. Es wurde kein Fremdschlüssel genutzt, um die Vorlesung, die betroffen ist, mit der Änderung zu verknüpfen. Somit sind die Daten für das menschliche Auge durchaus lesbar, für Maschinen aber nur schwer zu verarbeiten.

Deshalb wird auch hierfür eine eigene Datenbank genutzt, welche die eingehenden Einträge mit den vorhandenen Vorlesungen abgleicht und mit der passenden

Vorlesungs-ID referenziert. Somit kann jede Änderung auch einer einzigen Veranstaltung zugeordnet werden. Dies erfordert jedoch einen externen Service, welcher die Tabellen der originalen Datenbank miteinander Vergleicht und daraus die Einträge für die neue Datenbank generiert.

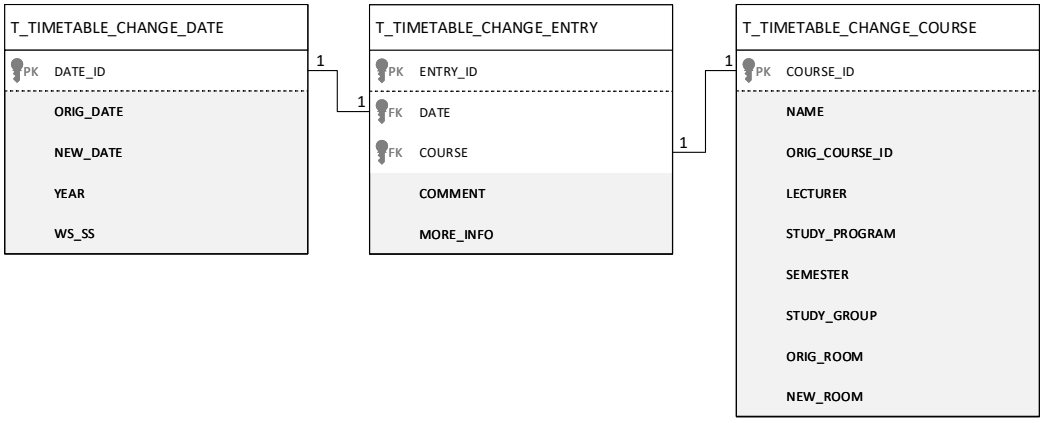


Abbildung 8.2: Entity Relation Diagramm der Datenbank zu den Planänderungen[2]

8.3 Speiseplan (Mensa-Service)

Bei der Wahl der Ernährung spielen soziale, politische, ökonomische, psychologische und kulturelle Dimensionen eine Rolle. Somit macht die Ernährung mehr als nur den Körper mit Nährstoffen zu versorgen. Das Ziel ist es, den Studierenden die Möglichkeit zur Verfügung stellen, ihren Speiseplan individuell nach ihren Bedürfnissen anzupassen. Dies wird durch den Mensa-Service realisiert. Der Service stellt dem Client alle Informationen über Gerichte zur Verfügung und erlaubt auch diese gezielt zu sortieren und zu personalisieren.

Anforderung an Schnittstelle

Eine Schnittstelle für die Speiseplan Daten steht derzeit nicht zur Verfügung. Die Informationen für den Speiseplan, die für die Hochschul-App benötigt werden, werden auf der Webseite des Studentenwerk-Oberfranken im Navigationsmenü Speiseplan für die Standorte Hof und Münchberg zur Verfügung gestellt. Der Ausschnitt der Webseite für den Speiseplan aus Abbildung 8.3 lässt sich in 7 Bereiche kategorisieren:

[2]Brysiuk, Lehmann (2019)

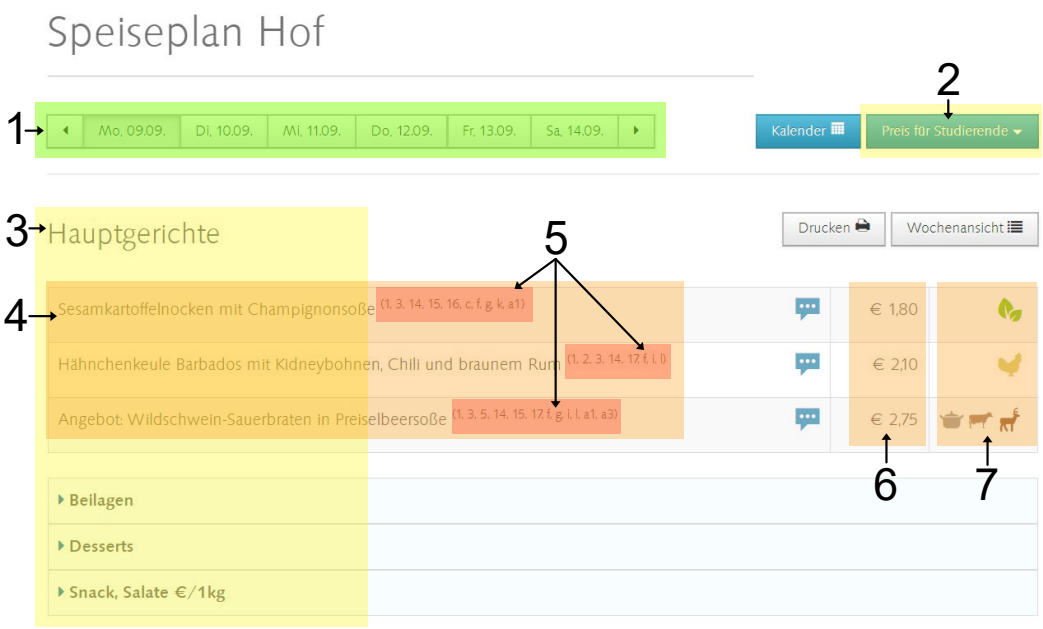


Abbildung 8.3: Speiseplan Informationsblöcke[3]

1. Datum für Speiseplan
2. Preistyp für alle Gerichte
3. Kategorie des Gerichts
4. Bezeichnung des Gerichts
5. Zusatzstoffe/ Inhaltsstoffe/ Allergien des Gerichts
6. Preis des Gerichts
7. Kennzeichnung des Gerichts

Wobei sich diese sieben Blöcke in drei weitere übergeordnete Bereiche einordnen lassen. Der erste bezieht sich auf die zeitliche Darstellung des Speiseplans. Der Speiseplan muss für ein beliebiges Datum oder einen Zeitraum dargestellt werden können. Der zweite bezieht sich auf die Darstellung und den Umfang des Speiseplans, bei dem der Preistyp und die Kategorie der Gerichte eine Rolle spielt. Also muss der Speiseplan zusätzlich zu den zeitlichen Einstellungen die Gerichte nach Preistyp und Kategorie sortiert werden können. Der dritte Block bezieht sich auf ein spezifisches Gericht, bei dem Bezeichnung, Zusatzstoffe/ Inhaltsstoffe/ Allergene,

[3]Brysiuk, Lehmann (2019)

Kennzeichnung und der Preis des Gerichts eine Rolle spielen. Durch die Abbildung 8.4 wird dargestellt, nach welchen Faktoren der Speiseplan sortiert und personalisiert werden kann. Es muss aber beachtet werden, dass das Backend nur für Kategorisierung und Filterung der einzelnen Gerichten zuständig ist und das Frontend für die Implementierung der Suche.

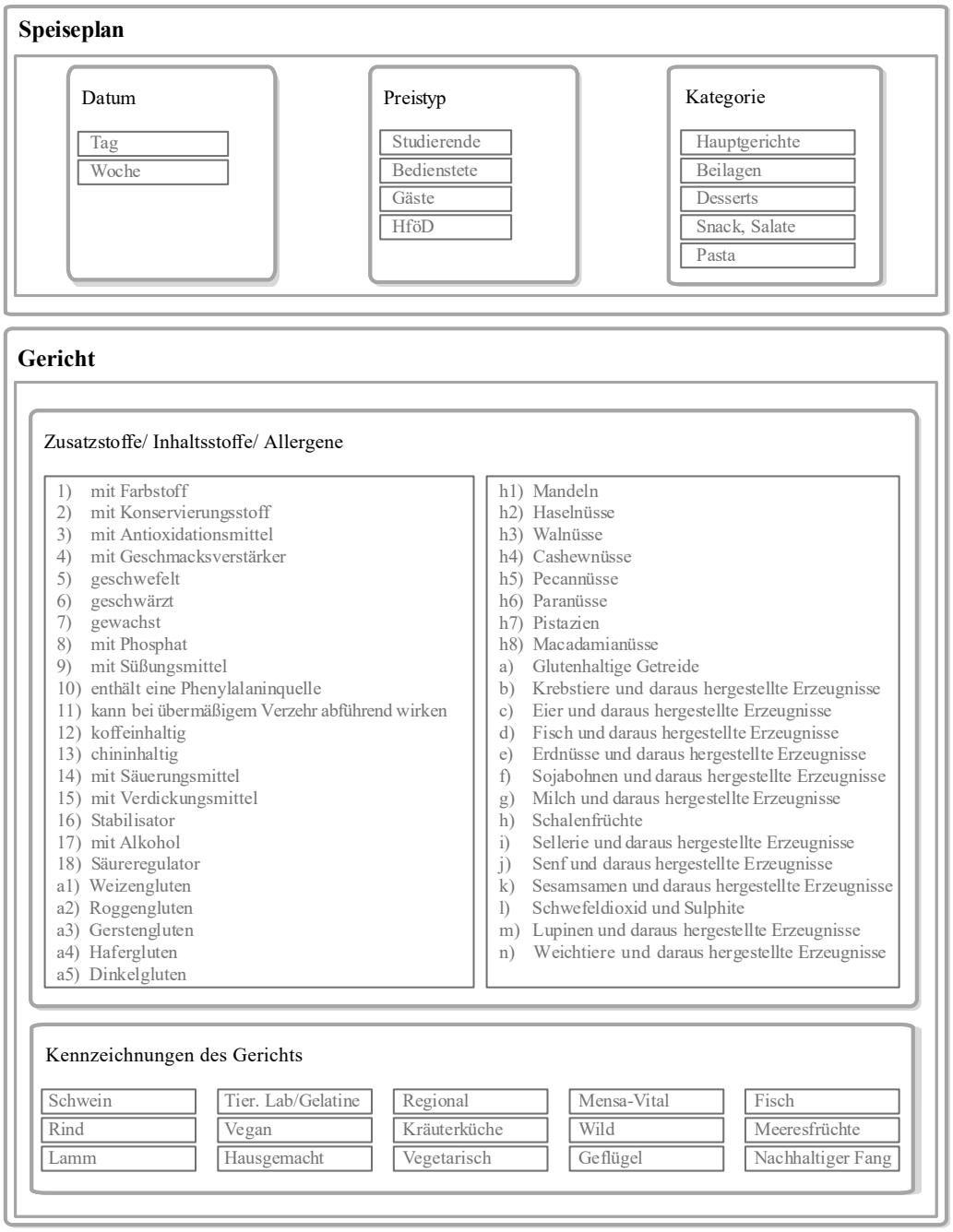


Abbildung 8.4: Bereiche der Speiseplaninformationen[4]

Solange keine Schnittstelle für den Speiseplan zur Verfügung steht, wird ein externer Service erstellt, der die Speiseplan Daten von der Webseite des Stundenwerk-Oberfranken ausliest und in eine Datenbank abspeichert.

Spezifikation der Ressourcen

Bezogen auf die technischen Anforderungen aus dem Kapitel 8.3 und der Kategorisierung des Speiseplans, werden hier die notwendigen Ressourcen für den Mensa-Service spezifiziert.

Alle GET-Methoden - außer die Subressourcen /filters und /{id} - können im Body die Filter für Preis, Kategorie, Kennzeichnung und Zusatzstoffe/Inhaltsstoffe/Allergie der Gerichte entgegen nehmen. Es müssen nicht alle Filter gesetzt werden. Ohne Übergabe der Filter werden standardmäßig alle Gerichte zurückgegeben. Wird der Filter Preistyp gesetzt, so werden nur die Gerichte mit diesem Preistyp zurückgegeben, sonst werden alle Preistypen zurückgegeben. In Kategorie, Zusatzstoffe oder Kennzeichnung des Gerichts können ein oder mehrere Filtertypen festgelegt werden. Wobei die übergebenden Filtertypen für Kategorie und Kennzeichnungen in den Gerichten enthalten sind und die Zusatzstoffe/Inhaltsstoffe/Allergene nicht.

- **GET/POST/DELETE:**
  /menu

  Die Methode GET liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern. POST erlaubt erstellen von ein oder mehreren Gerichten und DELETE löscht alle Gerichte.

- **GET/PATCH/DELETE:**
  /menu/{id}

  Die Methode GET liefert ein Gericht für die übergebene ID. Mit PATCH kann das Gericht geändert beziehungsweise korrigiert werden und DELETE löscht dieses Gericht.

[4] Brysiuk, Lehmann (2019)

- **GET:**
  /menu/filters

  Liefert alle Kategorien, Legenden, Zusatzstoffe und Preistypen die vorhanden sind.

- **GET/POST/DELETE:**
  /menu/filters/categories

  Die Methode GET liefert alle möglichen Kategorien. Mit POST können ein oder mehrere neue Kategorien eingefügt werden und DELETE löscht alle verfügbaren Kategorien.

- **GET/PATCH/DELETE:**
  /menu/filters/categories/{id}

  Die Methode GET liefert Informationen zur der übergebene Kategorien ID. Mit PATCH kann die Kategorie geändert beziehungsweise korrigiert werden und DELETE löscht diese Kategorie.

- **GET/POST/DELETE:**
  /menu/filters/labels

  Die Methode GET liefert alle möglichen Kennzeichnungen zu den Gerichten. Mit POST können ein oder mehrere neue Kennzeichnungen eingefügt werden und DELETE löscht alle verfügbaren Kennzeichnungen.

- **GET/PATCH/DELETE:**
  /menu/filters/labels/{id}

  Die Methode GET liefert Informationen zur der übergebene Kennzeichnung ID. Mit PATCH kann die Kennzeichnung geändert beziehungsweise korrigiert werden und DELETE löscht diese Kennzeichnung.

- **GET/POST/DELETE:**
  /menu/filters/excludes

  Die Methode GET liefert alle verfügbaren Zusatzstoffe/Inhaltsstoffe/Allergie. Mit POST können ein oder mehrere neue Zusatzstoffe/Inhaltsstoffe/Allergie eingefügt werden und DELETE löscht alle verfügbaren Zusatzstoffe/Inhaltsstoffe/Allergie.

- **GET/PATCH/DELETE:**
  /menu/filters/excludes/{id}

  Die Methode GET liefert Informationen zur der übergebenen Zusatzstoffe/Inhaltsstoffe/Allergie ID. Mit PATCH kann der Zusatzstoff/Inhaltsstoff/Allergie geändert beziehungsweise korrigiert werden und DELETE löscht diesen Zusatzstoff/Inhaltsstoff/Allergie.

- **GET:**
  /menu/filters/prices

  Die Methode GET liefert alle verfügbaren Preistypen.

- **GET:**
  /menu/week

  Liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern für die aktuelle Woche.

- **GET:**
  /menu/week/{week}

  Liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern für die übergebene Woche.

- **GET:**
  /menu/week/{week}/{day}

  Liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern für den übergebenen Wochentag in der übergebenen Woche.

- **GET:**
  /menu/week/{week}/{day}/{until-day}

  Liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern für die Zeitspanne innerhalb der übergebenen Wochentagen (inklusiv) in der übergebenen Woche.

- **GET:**
  /menu/date

  Liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern für den heutigen Tag.

- **GET:**
  /menu/date/{date}

  Liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern für den übergebenden Datum.

- **GET:**
  /menu/date/{date}/{until-date}

  Liefert alle Gerichte unter der Berücksichtigung von gesetzten Filtern für die Zeitspanne zwischen übergebenen Datumsangaben (inklusiv).

Datenbank Konzept

Um eine performante und anpassbare Datenquelle für die Daten des Speiseplans des Studentenwerk Oberfrankens zu haben, wird auch hierfür eine Datenbank erstellt. Da sich die Daten kaum ändern entlastet dies auch die Schnittstelle, von der letztendlich die Daten ausgelesen werden. Für die Funktionen des Mensa Services werden lediglich die Gerichte an sich benötigt. Diese Gerichte brauchen folgende Informationen:

- **ID**
  Einen klaren Identifier für ein Gericht, um für dieses Gericht weitere Informationen abfragen zu können

- **Datum**
  Das Datum, an dem das Gericht angeboten wird
- **Kategorie**
  Die Art von Gericht, zu dem der Eintrag zugeteilt werden kann
- **Beschreibung**
  Eine Kurzbeschreibung für das Gericht
- **Zugeordnete Kennzeichnungen**
  Kennzeichnungen, die das Gericht beschreiben
- **Preis für alle Preistypen**
  Verschiedene Preise für verschiedene Gästegruppen
- **Enthaltene Zusatzstoffe/Inhaltsstoffe/Allergene**
  Hinweise zum Inhalt und für Allergiker

Das resultierende Entity Relationship (ER)-Diagramm in Abbildung 8.5 zeigt die einfache Struktur der Datenbank für den Mensa Service. Für die Zusatzstoffe und die Kennzeichner der Gerichte werden sogenannte *n zu n* Beziehungen benötigt, für alles andere reichen Fremdschlüssel zur Zuordnung der weiteren Informationen. Die Tabelle *T_MENU_PRIZE* ähnelt zudem einer Dimensionstabelle eines Data Warehouses, statt die Preise anhand eines Standardwertes berechnen zu müssen, werden alle möglichen Kombinationen abgelegt und können dementsprechend ausgelesen werden. Es bietet sich ebenfalls an, einen *Non-Clustered*-Index auf die Fremdschlüssel zur Tabelle *T_MENU_CATEGORY_TYPE* in der Spalte *CATEGORY* der Tabelle *T_MENU_DISH* anzulegen. Des weiteren ist ein weiterer *Non-Clustered*-Index auf der Spalte *DATE* der selben Tabelle sinnvoll. Diese beiden Werte sind die, nach denen am wahrscheinlichsten sortiert und gefiltert wird. Im späteren Betrieb der Datenbank könnte man auch noch testen, ob sich einer der beiden Indizes als *Clustered*-Index eignen würde, da eine dauerhafte Sortierung der Einträge der Tabelle *T_MENU_DISH* nach der *ID* nicht zwingend hilfreich ist.

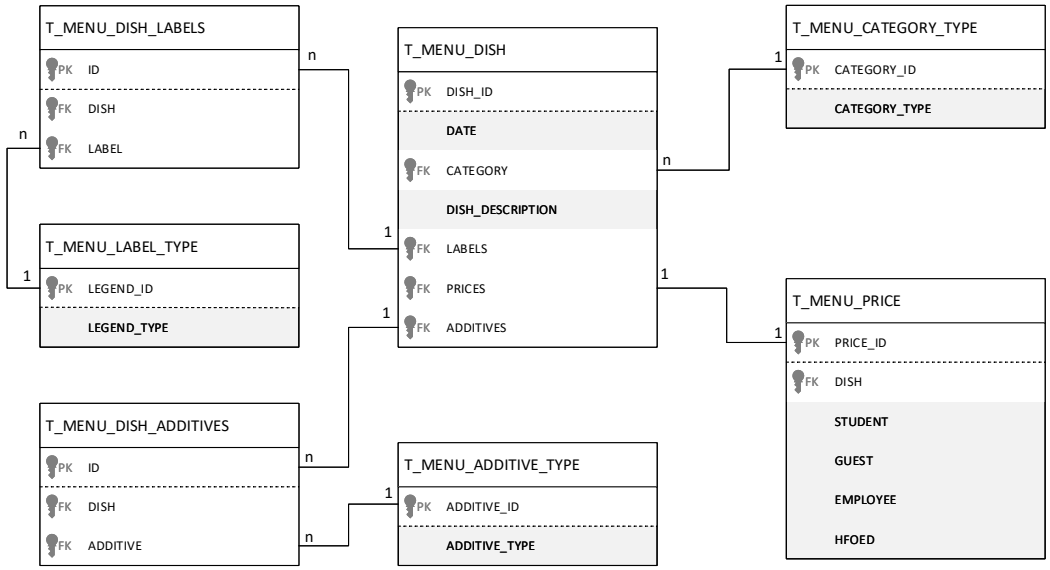


Abbildung 8.5: Entity Relation Digramm des Mensa-Speiseplans[5]

Weiterhin zu bemerken ist, dass die Datenbank auf Dauer nur die Gerichte der aktuellen und der nächsten Woche, also im Zeitraum von zwei Wochen, speichern wird. Ein automatisierter Scheduler Job im Mensa Service wird die alten Einträge automatisiert löschen und neue Einträge einspeisen. Für das Löschen der alten Einträge bietet es sich eventuell auch an, einen Datenbank Job zu schreiben, das kann in der Entwicklungsphase getestet werden. Jedoch richtet sich diese Datenbank stark nach dem KISS Prinzip, weshalb auch auf komplexere Strukturen verzichtet wurde. Das lässt sich dadurch begründen, dass nur Einträge für den Zeitraum von zwei Wochen gespeichert werden und keine historischen Daten gespeichert werden. Außerdem ist auch eine Änderung in den Formaten der eigentlichen Datenquelle nicht vorhersehbar. Die eigene Tabelle für die Spalte *CATEGORY* der Haupttabelle *T_MENU_DISH* beinhaltet auch eine Beschreibung der einzelnen Einträge, weshalb ein einfacher *Enumeration Type* für diesen Wert keinen Sinn ergibt.

8.4 Benachrichtigungen (Notification-Service)

Fast in jeder Anwendung sind Push-Notifications eingebunden. Am häufigsten kennt man es von Facebook oder dem Whats-App Messenger, dass die Nachrichten in Echtzeit übertragen werden und direkt auf den Smartphone als Benachrichtigung angezeigt werden. Die Funktionsweise ist einfach, man erhält Benachrichtigungen

[5] Brysiuk, Lehmann (2019)

von einer Programm ohne diese öffnen zu müssen. Dies ermöglichtes, den Benutzer über wichtige Ereignisse oder Neuigkeiten zur informieren ohne die Anwendung zu starten oder diese neu zu laden. Solche Benachrichtigung können für die Studierenden von Vorteil sein. Sollte sich eine Vorlesung ändern oder ausfallen, werden die an der Vorlesung teilnehmenden Studierenden informiert. Aber auch andere Services können von dieser Technik profitieren, wie beispielsweise der Mensa-Service oder der Sprachzentrum-Service. Das Ziel ist es, ein Notification-Service zu erstellen, der eine einheitliche Schnittstelle für alle anderen Services zur Verfügung stellt, um die betroffenen Benutzer über eine Änderung jeglicher Art zu informieren.

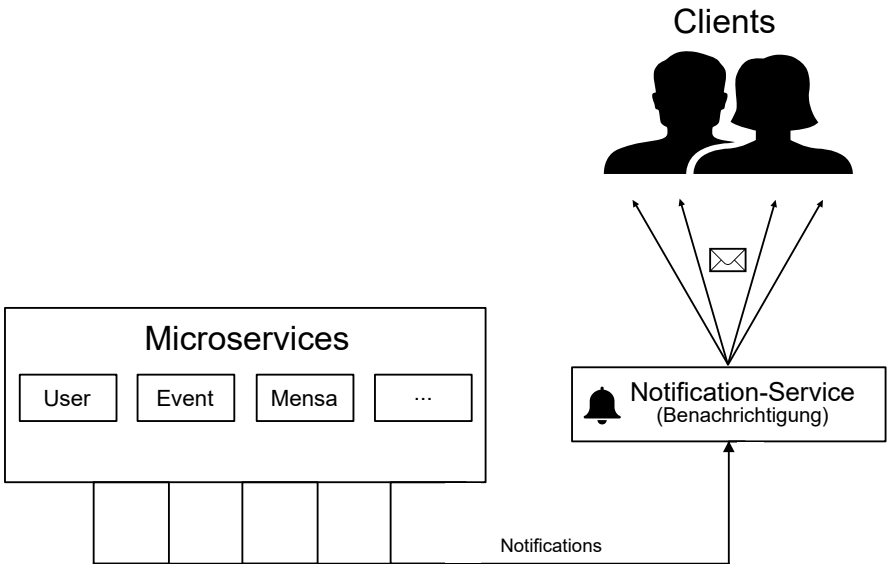


Abbildung 8.6: Benachrichtigungsservice[6]

Anforderung an Schnittstelle

Die Funktionalität eines Services, der den Notification-Service in Anspruch nimmt, darf nicht vom Ergebnis des Services beeinflusst werden. Sobald ein Service eine Benachrichtigung an den Notification-Service übermittelt, ist es dem Service nicht mehr wichtig wie, wann und ob die Nachricht zugestellt wird. Damit wird die Abhängigkeit zwischen unterschiedlichen Funktionalitäten reduziert und es wird ein Maß an loser Kopplung garantiert.

[6]Brysiuk, Lehmann (2019)

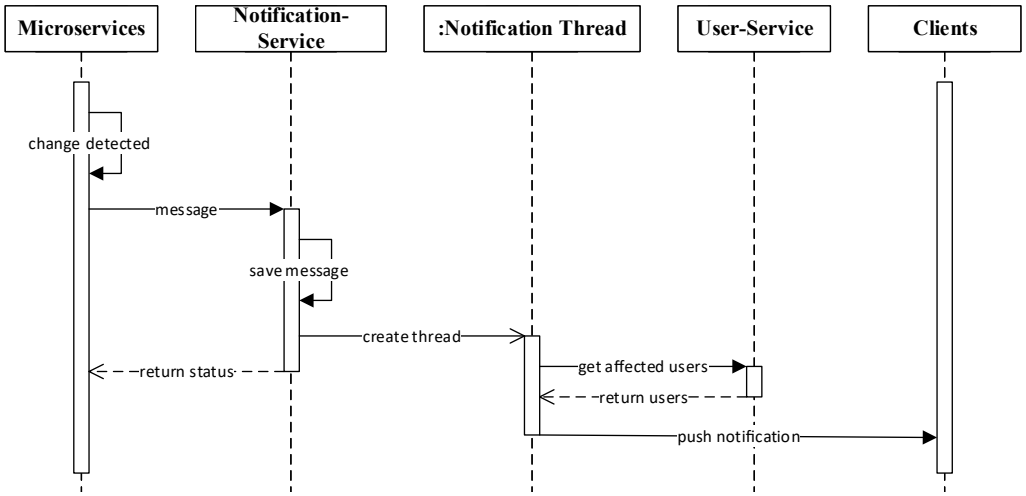


Abbildung 8.7: Sequenzdiagramm zum Ablauf einer Benachrichtigung[7]

In Abbildung 8.7 ist ein Sequenz Diagramm zu sehen das einen groben Ablauf darstellt, wie eine Benachrichtigung von einem Microservice bis zum Benutzer kommuniziert wird. Der Microservice sendet die Nachricht an den Notification-Service und endet der Prozess für den Microservice. Der *return status* wird vom Microservice ignoriert, sowohl bei einer erfolgreichen Antwort als auch bei einer nicht erfolgreichen Antwort. Der *return status* wurde im Sequenz Diagramm nur aufgenommen, weil eine REST-API immer eine Antwort zurück senden muss. Ab jetzt ist nur der Notification-Service zuständig für eine erfolgreiche Übermittlung der Nachricht an die Clients. Um sicher zu gehen, dass die Nachricht nicht auf dem Weg verloren geht, bei ersten Versuch nicht erfolgreich an den Client übermittelt werden kann oder beim Absturz des Services verloren geht, wird die Nachricht zuerst in der Datenbank abgelegt. Der Notification-Service ist selbständig verantwortlich dafür, die jeweiligen Benutzer für die Nachricht zu finden und zu informieren.

Spezifikation der Ressourcen

Der Notification-Service benötigt im Grunde zwei Ressourcen, die eine zur Registrierung eines Clients, die andere für die Übergabe einer Nachricht. Um den Service leichter mit den Hochschul-App Services nutzen zu können, wurde der Service um die Subressourcen */lectures* und */mensa* erweitert. Außerdem wurde die Benachrichtigung einzelner Benutzer berücksichtigt. Die Übergabe der Informationen an den Notification-Service erfolgt im Body Bereich.

[7] Brysiuk, Lehmann (2019)

- **POST:**
  /register

  Für die Registrierung des Clients, der über Neuigkeiten informiert werden soll.

- **POST:**
  /message

  Benachrichtigung an alle Benutzer des registrierten Clients.

- **POST:**
  /message/{userId}

  Benachrichtigung an übergebenen Benutzer.

- **POST:**
  /message/lectures

  Benachrichtigung an Benutzer, die für die Vorlesung hinterlegt sind.

- **POST:**
  /message/mensa

  Benachrichtigung an Benutzer, die für ein Gericht hinterlegt sind.

Datenbank Konzept

Wie bereits vorher erwähnt wurde, werden die Nachrichten temporär in der Datenbank abgespeichert und solange behalten, bis die Benachrichtigung tatsächlich die Benutzer erreicht hat. Durch die Spezifikation der Ressourcen ergeben sich folgende Tabellen.

1. Abspeicherung der Registrierten Clients
2. Benachrichtigung und zugehörige Informationen

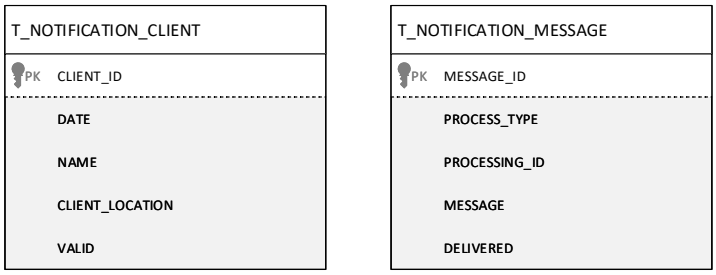


Abbildung 8.8: Entity Relation Diagramm der Benachrichtigungs Datenbank[8]

8.5 Sicherheit (Auth-Service)

Um alle Services abzusichern, wird ein Sicherheitskonzept benötigt. Zum einen müssen die Clients autorisiert werden und anderen die Benutzer authentifiziert werden. Um einen gesicherten Zugriff auf die Microservices zu gewährleisten, ohne dass die Services vom Authentifizierung-Service abhängig sind, wird ein Token an die Microservices übersendet, über welchen die Zugriffserlaubnis von jeden Service unabhängig geprüft werden kann, ohne den Authentifizierung Service aufzurufen.

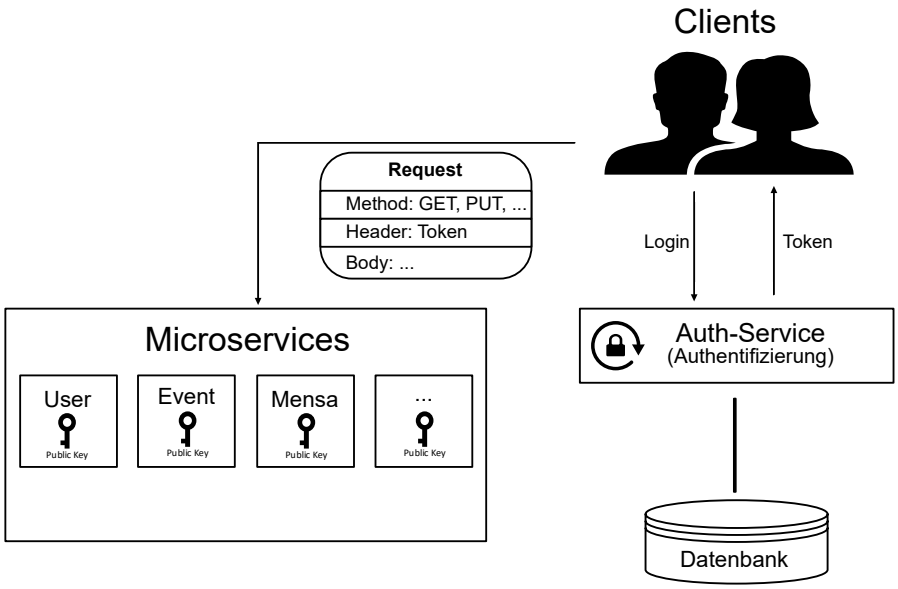


Abbildung 8.9: Authentifizierungsservice[9]

Für die Zugriffssteuerung wird das Konzept von Rollen und Rechten verwendet.

[8] Brysiuk, Lehmann (2019)
[9] ebd.

Ausführliche Behandlung diesen Services ist in der hierzu parallel angefertigten Bachelorarbeit zu sehen.[10]

8.6 Anwenderverwaltung (User-Service)

Der User-Service dient dazu, Benutzer zu verwalten und das mitsamt ihrer Accountinformationen und Einstellungen. Die ausführliche Behandlung dieses Services ist in der hierzu parallel angefertigten Bachelorarbeit zu sehen[11].

8.7 Weitere Dienste

Die Services Sprachzentrum und Termine sind zwar in der Architektur der Hochschul-App berücksichtigt worden, jedoch überschreiten diese den Umfang dieser Bachelorarbeit. Durch die modularisierte und flexible Architektur können diese Services jederzeit separat entwickelt werden und an die aus dieser Arbeit entstandene Hochschul-App verknüpft werden.

[10] Vgl. Glaser (2019a).
[11] ebd.

9 Ausblick und Fazit

Im Zuge der Digitalisierung werden die Anforderungen an moderne Apps immer höher. Zum einen muss die Anwendung benutzerfreundlich gestaltet sein, zum anderen immer den neuesten Entwicklungen folgen, die es im Bereich der Smartphones gibt. So muss man sich im Laufe der Entwicklung eines neuen Produktes oft entscheiden, ob man eine leichtgewichtige Anwendung schafft, die nur die Kernfunktionen übernimmt, für die sie ursprünglich gedacht wurde, oder ob die Entwicklung breiter ausfällt und man eine Anwendung schafft, die dem Nutzer mehr abnimmt, als nur die Kernfunktionen. Dabei muss am Anfang der Erarbeitung des Konzeptes immer der Grundsatz stehen, das Programm so aufzubauen, dass man im Entwicklungsprozess alle Möglichkeiten offen hat und genau das ist es, was das große Ziel dieser Arbeit ist. Eine Anwendung zu entwerfen, die anfangs die wichtigsten Grundlagen bereitstellt, die aber im Laufe der Zeit immer weiter entwickelt und erweitert werden kann.

Der Fokus auf modulare und erweiterbare Anwendungen steht aktuell im Mittelpunkt der modernen Softwareentwicklung, deshalb sollte auch das das Ziel eines Projektes sein, das in einer Hochschule von Studierenden bearbeitet wird, die später mit bestem und modernstem Wissen in die Arbeitswelt einsteigen sollen. In den vorhergehenden Kapiteln dieser Arbeit wird mehrmals der Grundsatz der Erweiterbarkeit bei einer Microservice Architektur erwähnt und erläutert. Genau nach diesem Grundsatz wird die Hochschul-App auch aufgebaut sein, das heißt, dass es in Zukunft die Möglichkeit geben wird, die Funktionen der Anwendung zu erweitern. Dieses Kapitel soll nun kurz beleuchten, welche weiteren Features noch möglich und sinnvoll sind.

9.1 Ausblick

Einige der funktionalen Anforderungen aus den Kapiteln 3.1, 3.2 und 3.3, wurden in dieser Arbeit nicht weiter betrachtet. Wie bereits erwähnt, stellt die gewählte Ar-

chitektur des Projektes die Möglichkeit dar, weitere Funktionen einzubinden, ohne etwas am bestehenden Programmcode ändern zu müssen. Die in dieser Bachelorarbeit nicht betrachteten möglichen Funktionen werden im folgenden aufgelistet.

Sprachumfang

Im Rahmen der Internationalisierungsbemühungen der Hochschule Hof liegt es im Interesse des International Office, eine Hochschul-App anbieten zu können, die in mehreren Sprachen erhältlich ist. Einige mögliche Vorschläge und favorisierte Sprachen der aktuellen Studierenden der Hochschule Hof wurden in Kapitel 3.2.2 bereits aufgelistet. Zusätzlich zu den bereits im Funktionsumfang dieser Bachelorarbeit und der zugehörigen Praxisarbeit [1] enthaltenen Sprachen Deutsch und Englisch kann die Anwendung ebenfalls um weitere Sprachen erweitert werden. Dazu könnte ein Konzept erstellt werden, das eine kontinuierliche Einbindung neuer Sprachen unterstützt. So könnten internationale Studierende schnell eigene Sprachen einpflegen.

Verlinkung wichtiger Informationen

Ein Wunsch, der aus den Umfragen in Kapitel 2.1.2 hervorgeht, ist die Verlinkung wichtiger Informationen zu den Vorlesungen, die im Stundenplan aus der Schnittstelle ausgelesen werden. Dazu gehören beispielsweise eine Referenz auf das entsprechende Modulhandbuch der Vorlesung oder auch die in Kapitel 3.3.2 erwähnten Anmeldungen zu den Sprachkursen. Diese Informationen sind in den vorhandenen Datenbanken der Hochschule nicht vorhanden, somit muss erst analysiert werden, wie man die passenden Informationen zu den Vorlesungen zuteilen kann.

Sprachkursinformationen

Anders als nur die Verlinkung der Anmeldung zu den Sprachkursen verhält es sich mit den allgemeinen Informationen zu den Sprachkursen. Ein Sprachkurs an sich enthält weit mehr Informationen, als es normale Vorlesungen tun. Ein Sprachkurs hat vorausgesetzte Kenntnisse oder vorhergehende Sprachkurse, die erst abgelegt werden müssen, um teilnehmen zu dürfen. Zudem bieten viele Sprachkurse ebenfalls eine Zertifizierung an, welche am Ende des Kurses abgelegt werden kann. Zu den bereits genannten Punkten kommen noch weitere Informationen, die man über

[1] Siehe Kapitel 1.5

einen Sprachkurs wissen möchte. Diese anzuzeigen benötigt jedoch weitere Analysen, denn die Informationen werden in einer anderen Datenquelle gepflegt, als die regulären Vorlesungsinformationen. Weitere Anmerkungen zu diesem Problem wurden bereits im Kapitel 3.3.1 aufgeführt.

Kursvorschläge

Engagierte Studierende wünschen sich oft, sich auch außerhalb ihrer Pflichtveranstaltungen weiterbilden zu können. Im allgemeinen wurde in der Umfrage, die in Kapitel 2.1 genauer betrachtet wurde, oft bemerkt, dass man sich wünscht, in freien Zeiten andere relevante Vorlesungen vorgeschlagen zu bekommen. Darunter fallen für die meisten Studierenden auch Sprachkurse. Die Einbindung einer solchen Funktion bedarf weiteren Entwicklungsaufwand, denn es wird ein Algorithmus benötigt, der mögliche Vorlesungen findet. Zudem wäre es ratsam, nicht jede Art von Vorlesung vorzuschlagen, sonder nur die, die tatsächlich relevant für den Nutzer ist. Dies bedarf einer Logik oder auch Intelligenz, die über den Rahmen dieser Arbeit hinaus geht.

Sondertermine

Die Hochschule Hof bietet ein breites Spektrum an Veranstaltungen an, das weit über die regulären Vorlesungen der Studierenden hinausgeht. Oft werden Lesungen oder Seminare angeboten, die im eigentlichen Stundenplan nicht angezeigt werden. Außerdem halten oft auch externe Personen Vorträge über aktuelle Themen, auch diese werden lediglich auf der Website der Hochschule angezeigt oder per E-Mail bekannt geben. Ein mögliches Feature der Hochschul-Anwendung könnte es demnach sein, auch aktuelle Sondertermine mit anzuzeigen und dem Nutzer auch vorzuschlagen, falls es relevant für das Themengebiet seines Studiums ist.

Primuss-Einbindung

Ein wichtiger Baustein im Alltag der Studierenden ist das Primuss-Portal der Hochschule Hof. Das Primuss-Portal ist ein Portal, bei dem die Studierenden alle organisatorischen Aufgaben für ihr Studium erledigen können. Dazu gehören beispielsweise Anträge und Formulare aus dem Studienbüro, das Ausdrucken von Studienbestätigungen, die Einrichtung der Bankverbindung für den Studienbeitrag und die Einsicht der vorläufigen Noten aus abgelegten Prüfungen. Eine Einbindung dieses

Portals in einer Hochschul-App würde den Nutzern der Anwendung viel Zeit dabei sparen, das Portal selbst aufzurufen und sich bei jedem Mal neu anmelden zu müssen.

Kalendersynchronisation

Wie bereits erwähnt wurde, hat die Hochschule Hof neben den eigentlichen Vorlesungen auch einige Sondertermine und auch andere Termine, die den Studienalltag betreffen, wie zum Beispiel den freien beweglichen Tag oder feste Termine, beispielsweise Semesterbeginn und Semesterende, sowie Rückmeldefristen und Prüfungszeiträume. Diese Termine, alternativ auch in Zusammenhang mit den eigentlichen Vorlesungen, können über eine Kalendersynchronisation in den Endgeräten und den privaten Konten der Nutzer automatisiert gespeichert werden. Dies bedarf einer Einbindung und Speicherung der Termine in einer Datenquelle und einen weiteren Service, über den die Daten abgefragt sowie synchronisiert werden können.

Die Liste der Vorschläge ist nur ein Auszug aus den Möglichkeiten, die zur Erweiterung der Hochschul-App umgesetzt werden könnten. Man erkennt bereits, dass dieses Projekt ein großes Potential für die Zukunft bietet, bei der die Hochschule Hof eine Anwendung vorzeigen kann, die den modernsten Ansprüchen und Techniken gerecht wird und die deren Studierenden eine einfache und mächtige Datenquelle für alles liefert, was sie für den Studienalltag benötigen.

9.2 Weitere Bachelorarbeitsthemen

Wie aus Kapitel 9.1 hervorgeht, bietet das Resultat dieser Bachelorarbeit und des daraus entstehenden Projektes ein großes Potential an Erweiterungen. Anhand der Umfrage, die in Kapitel 2.1 genauer betrachtet wurde, ist das Interesse der Studierenden an einer Hochschul-App, die einen großen Funktionsumfang vorweisen kann, hoch. Eine stetige Verbesserung der Anwendung erhält das Interesse der Studierenden an der Hochschul-App und erhält diese ebenfalls stets auf dem neuesten Stand der Entwicklungen, was die Langlebigkeit ebenfalls weit erhöht. Direkt am Anschluss dieser Arbeit lassen sich zwei neue Bachelorarbeitsthemen definieren, die die Qualität der App auf lange Sicht steigern sollen.

Erweiterung der web-basierten Hochschul-App

Die bereits aufgelisteten Erweiterungen und möglichen weiteren Funktionen der Hochschul-App sind lediglich einige Vorschläge, die die Nützlichkeit der Anwendung stark verbessern würden. Jedoch verbleibt es nicht nur bei den genannten Erweiterungen, eine Analyse der möglichen weiteren Verbesserungen und die anschließende Entwicklung und Einbindung dieser würden die Hochschul-App auf Dauer deutlich interessanter und attraktiver für die Studierenden der Hochschule Hof machen. Da die Grundfunktionen, für die die Anwendung eigentlich gedacht ist, bereits im Anschluss an diese Bachelorarbeit entwickelt werden, kann sich eine weitere Bachelorarbeit nur damit befassen, die Funktionen zu erweitern. Dabei sind die weiteren Funktionen auch nicht nur auf das oberflächliche Verhalten der App beschränkt, sondern können sich auch auf andere Aspekte wie die Sicherheit, der Anwenderverwaltung und dem Single Sign On (SSO) der Hochschule Hof beziehen.

Verbesserung der Nutzerfreundlichkeit der Hochschul-App

Im Rahmen dieser Bachelorarbeit und der parallel dazu bearbeiteten Bachelorarbeit [2] zum Thema Sicherheit, Nutzerverwaltung und der grafischen Oberfläche werden lediglich die Grundlagen für eine solide, langlebige und aktuelle Anwendung geschaffen. Die grundlegenden Funktionen werden implementiert und es wird eine erste Version der grafischen Benutzeroberfläche geschaffen. Jedoch liegt der Fokus bei der Entwicklung im Anfangsstadium der neuen web-basierten Hochschul-App auf der Funktionalität und der Modularität. Es liegt also nahe, eine weitere Arbeit anzufertigen, die sich mit der Nutzerfreundlichkeit der Anwendung auseinander setzt. Dabei können Aspekte wie die Nutzbarkeit und die optische Wirkung der Oberfläche betrachtet werden, jedoch auch die Nutzung der Daten Schnittstelle, welche in Kapitel 8 definiert wurde, verbessert werden. Sinnvoll ist es hierbei, Nutzungsstatistiken zu sammeln, Befragungen der Studierenden durchzuführen und auf allgemeine Aspekte der User Interface (UI) und User Experience (UX) Entwicklung zu achten. Anhand solcher Verbesserungen kann eine Anwendung erschaffen werden, mit der die Studierenden nicht nur ihre Informationen einsehen können, sondern an der sie sich auch erfreuen können und die sie gern nutzen. Zudem wird somit eine Anwendung entwickelt, die als Aushängeschild für eine qualitativ hochwertige Entwicklung der Hochschule Hof dient.

[2] Siehe Kapitel 1.5

9.3 Fazit

Im Einklang mit der Digitalisierung, die auch im Alltag der Hochschulen immer präsenter wird, wurde in dieser Bachelorarbeit eine web-basierte Hochschul-App entwickelt und erarbeitet, welche auch in weiterer Zukunft den Standards der Nutzer entsprechen wird. Dies wurde vor allem durch den modularen Aufbau und der Nutzung einiger Rahmenbedingungen, die die Langlebigkeit der App deutlich unterstützen, erreicht.

Die Analyse der allgemeinen Zufriedenheit der Nutzer mit dem vorhandenen Anwendungsangebot der Hochschule Hof hat ergeben, dass die Akzeptanz der Nutzer für den aktuellen Anwendungen im allgemeinen hoch ist. Der Aufwand, der in die Entwicklung dieser Anwendungen geflossen ist, ist dadurch gerechtfertigt, dass über 90% der Studierenden der Hochschule Hof eben einer dieser nativen, plattformgebundenen Anwendungen nutzen[3]. Jedoch ging aus der selben Umfrage auch heraus, dass der Wunsch nach einer plattformübergreifenden App genauso hoch ist, wie der Wunsch nach Erweiterungen und einer stabilen Anwendung, auf die sich die Studierenden beim Aufrufen wichtiger Informationen verlassen können.

Ebenfalls hervor ging aus dieser Umfrage, dass die Internationalen Studierenden sich schwer tun, die Apps vollständig zu nutzen. Darauf aufbauend gingen weitere funktionale Anforderungen aus den Bereichen des International Office und des Sprachenzentrums in die Liste der gewünschten Features der web-basierten Hochschul-App ein. Zu diesen wurden ebenfalls die funktionalen Anforderungen des Auftraggebers mit aufgezählt. Durch die klare Linie des Auftraggebers bei der Sammlung der Anforderungen konnte eine klare Struktur für die Anwendung geschaffen werden. Anders als bei den funktionalen Anforderungen sind die Vorgaben des Auftraggebers in Sachen Umsetzung sehr offen, was eine perfekte Rahmenbedingung für die Analyse und die Entwicklung einer grundlegenden Architektur schaffte.

Anhand des gegebenen Freiraums bei der Entwicklung konnte ein Konzept entwickelt werden, das den Anforderungen der Langlebigkeit und der Erweiterbarkeit in allen Ansprüchen gerecht wird. Mit der modularen Webservice Architektur wurde ein Konzept gefunden, das die funktionalen Anforderungen sowohl im Sourcecode, als auch im Deploymentprozess, klar trennt. Dies ermöglicht ein einfaches Aus-

[3] Siehe Kapitel 2.1.2

tauschen der Module, auch Webservices genannt, falls dies nötig wird. Ebenfalls konnte so eine optimale Auslastung der Ressourcen gesichert werden, denn die einzelnen Webservices sind separat skalierbar und können auf verschiedenen logischen Adressen laufen.

Im Rahmen der zweiten zu diesem Thema angefertigten Bachelorarbeit konnte ebenfalls klar gemacht werden, wie einfach es ist, im Laufe der Jahre die optische Darstellung der Anwendung auszutauschen, da diese genau so modular aufgebaut ist, wie der Rest der Anwendung. Zudem können mehrere Anwendungen entwickelt werden, die nur auf die Daten der Hochschul-App zugreifen. Dies wurde durch die Definition der REST-Schnittstelle geschafft. Für die zukünftige, kontinuierliche und konsistente Weiterentwicklung der Anwendung, im speziellen der Schnittstelle, wurde ebenfalls ein Regelwerk geschaffen, welches das Design der Datenschnittstellen klar definiert. Anhand der bereits entwickelten Endpunkte dieser Schnittstelle können spätere Entwickler der App klar erkennen, wie diese zu erweitern ist.

Wie aus Kapitel 9.1 hervorgeht, wurde hier eine Grundlage einer Anwendung geschaffen, die in Zukunft deutlich weiter wachsen kann. Genau das sollte auch der Fokus der zuständigen Entwickler und des Auftraggebers sein, eine App zu schaffen, die den Studierenden der Hochschule Hof den Alltag deutlich erleichtert. Konzentriert man sich dabei in Zukunft auf die in dieser Arbeit angebrachten Punkte und hält sich dabei an die beschriebenen Richtlinien, so steht einem erfolgreichen Projekt und der Erweiterung des Prototypen der web-basierten Hochschul-App zu einer vollständigen und erfolgreichen Anwendung nichts im Wege.

Anhang

Lastenheft

Ref.	Anforderung	Quelle
1	Plattform	Auftraggeber
1.1	Betriebssystemunabhängigkeit	Auftraggeber
1.2	Geräteunabhängigkeit	Auftraggeber
1.3	Mobile First	Auftraggeber
2	Stundenplan	Auftraggeber
2.1	Abrufen des allgemeinen Stundenplans	Auftraggeber
2.2	Personalisierung nach Fakultät	Auftraggeber
2.3	Personalisierung nach Studiengang	Auftraggeber
2.4	Personalisierung nach Fachsemester	Auftraggeber
2.5	Einbindung nicht regulärer Vorlesungen	Auftraggeber
2.6	Einbindung des erweiterten Vorlesungsspektrums	Auftraggeber
3	Stundenplan Änderungen	Auftraggeber
3.1	Unterscheidung einmalige/ langfristige Änderungen	Auftraggeber
3.2	Automatische Anpassung langfristiger Änderung in Stundenplan	Auftraggeber
3.3	Benachrichtigung der Nutzer bei Änderungen	Auftraggeber
3.4	Anzeige einmaliger Änderungen in Stundenplan	Auftraggeber
4	Speiseplan	Auftraggeber
4.1	Anzeige des Speiseplans (Studentenwerk Oberfranken)	Auftraggeber
4.2	Filterung der Anzeige von Speiseplan Informationen	Auftraggeber
4.3	Anzeige zusätzlicher Informationen pro Gericht (z.B. Allergene)	Auftraggeber

Tabelle IX.1: Lastenheft

Ref.	Anforderung	Quelle
5	Anwenderverwaltung	Auftraggeber
5.1	Speicherung Nutzer spezifischer Informationen	Auftraggeber
5.2	Anmeldung notwendig bei Speicherung der Daten	Auftraggeber
5.3	Aufteilung der Nutzer in verschiedene Gruppen	Auftraggeber
5.4	Anmeldung durch Hochschul-E-Mail-Adresse	Auftraggeber
5.5	Anmeldung auch für Studierende ohne FH-E-Mail	Auftraggeber
5.6	Automatisiertes Löschen alter Daten	Auftraggeber
6	Mehrsprachigkeit	Auftraggeber/ International-Office
6.1	App in deutscher Sprache	Auftraggeber
6.2	App in englischer Sprache	Auftraggeber/ International-Office
6.3	Einfache Erweiterung der App um weitere Sprache	International-Office
7	Einfache personalisierte Stundenplanerstellung	Auftraggeber/ International-Office
7.1	Fakultäten unabhängige Stundenplanerstellung	Auftraggeber/ International-Office
7.2	Studiengang unabhängige Stundenplanerstellung	Auftraggeber/ International-Office
7.3	Einfache Einbindung externer Kurse	Auftraggeber/ Sprachzentrum
8	Einbindung des Sprachenzentrums	Sprachzentrum
8.1	Einbinden von Sprachkursinformationen	Sprachzentrum
8.2	Mehrsprachige Sprachkursinformationen	Sprachzentrum
8.3	Einheitliche Darstellung der Sprachkursinformationen	Sprachzentrum
8.4	Vollständige Informationsdarstellung	Sprachzentrum

Tabelle IX.2: Lastenheft

Umfrage

Deutschsprachige Umfrage

Umfrage zur Erneuerung der Hochschul Stundenplan App

1. Über welche Art und Weise informieren Sie sich über Ihren Stundenplan, Stundenplanänderungen und den Speiseplan? *

   Anzahl Teilnehmer: 217

   137 (63.1%): Android App

   63 (29.0%): iOS App (iPhone und Co.)

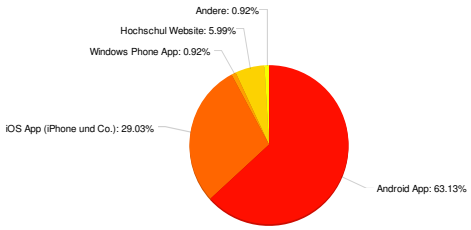
   2 (0.9%): Windows Phone App

   13 (6.0%): Hochschul Website

   2 (0.9%): Andere

   Antwort(en) aus dem Zusatzfeld:

   - VWA Portal
   - Speiseplan über Safary, funktioniert in der iOS App nicht



2. Wenn Sie eine der 3 Apps (Android, iOS, Windows Phone) nutzen, wie häufig nutzen Sie diese? *

   Anzahl Teilnehmer: 217

   65 (30.0%): täglich

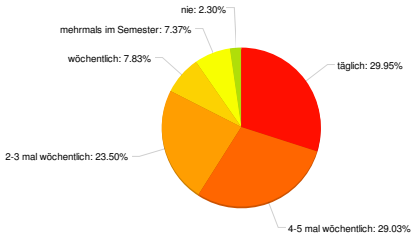
   63 (29.0%): 4-5 mal wöchentlich

   51 (23.5%): 2-3 mal wöchentlich

   17 (7.8%): wöchentlich

   16 (7.4%): mehrmals im Semester

   5 (2.3%): nie

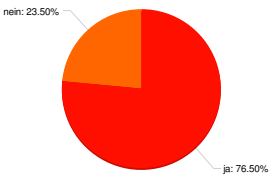


3. Würden Sie die App häufiger und intensiver nutzen, wenn sie verbessert werden würde? *

Anzahl Teilnehmer: 217

166 (76.5%): ja

51 (23.5%): nein



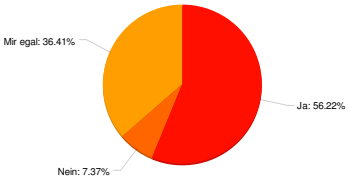
4. Würden Sie eine plattformunabhängige App bevorzugen? *

Anzahl Teilnehmer: 217

122 (56.2%): Ja

16 (7.4%): Nein

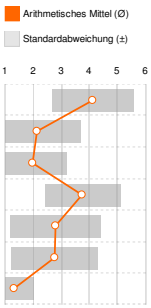
79 (36.4%): Mir egal



5. Wie sinnvoll finden Sie folgende Funktionen der bereits vorhandenen Anwendungen/ Apps der Hochschule Hof? *

Anzahl Teilnehmer: 206

	Unentbehrlich! (1)		Sinnvoll (2)		Vertretbar (3)		Verzichtbar (4)		Hier total fehl am Platz! (5)		Habe ich noch nie gesehen... (6)			
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Ø	±
Chat Option für jede Vor…	2x	0,97	33x	16,02	33x	16,02	65x	31,55	16x	7,77	57x	27,67	4,12	1,44
Raumsuche	88x	42,72	79x	38,35	11x	5,34	4x	1,94	1x	0,49	23x	11,17	2,13	1,54
Primuss Portal Einbindung	79x	38,35	92x	44,66	14x	6,80	10x	4,85	1x	0,49	10x	4,85	1,99	1,21
Wo bin ich? (Karte mit ak…	7x	3,40	33x	16,02	44x	21,36	79x	38,35	9x	4,37	34x	16,50	3,74	1,34
Fahrplan (Bus und Bahn)	35x	17,07	87x	42,44	31x	15,12	19x	9,27	3x	1,46	30x	14,63	2,80	1,60
Kalender Synchronisation	38x	18,54	79x	38,54	37x	18,05	22x	10,73	3x	1,46	26x	12,68	2,76	1,55
Push Notification bei Stu…	155x	75,61	44x	21,46	2x	0,98	2x	0,98	-	-	2x	0,98	1,31	0,70



6. Haben Sie allgemeine Verbesserungsvorschläge zu den Apps der Hochschule Hof?

Anzahl Teilnehmer: 48

Alle 13 vorangegangenen Antworten anzeigen

- Für IOS Speiseplan mit in Hochschulapp integrieren
- Eine Übersicht über Events an der Hochschule
- Ne hab keinen Vorschlag, aber auch kein Easteregg
- push notification reichen nicht aus, die stunde die ausfällt soll auf der listview mit einer dezent-sichtbare farbe hervorgehoben werden, damit man visuell sehen das da was etwas ist.
- Die Push Notification für Stundenplanänderungen funktioniert bei mir seit diesem Semester nicht mehr(Android). Ansonsten super
- Die App für‘s iPhone hat leider keinen Speiseplan.

  Bei dauerhaften Änderungen im Stundenplan muss man in der App für‘s iPhone das Fach erst abwählen, speichern und wieder anwählen. Das ist mühsam (falls man überhaupt mitbekommt, dass es eine Änderung gab)
- Lieber Noah, ich mag dich.
- Es wäre sehr hilfreich die Android-App endlich als reine APK-Datei zur Verfügung zu stellen. Der Google-Zwang ist für die APP einfach nicht nötig. Noch besser wäre es wenn die APP im freien App-Repository von F-Droid verfügbar wäre (wie das auch viele Hochschulen schon gemacht haben). Wenn der Quellcode um jeden Preis geheim bleiben muss (warum auch immer), kann immer noch mit minimalen Aufwand ein eigenes Repository für F-Droid angelegt werden. Zum Beispiel völlig kostenlos über ein Github-Projekt (siehe F-Droid Repo-Maker).
- App: Raumsuche sollte funktionieren; wäre angenehm, wenn alle Plattformen ein einheitliches Erscheinungsbild aufweisen würden Webseite: auf Handy: wäre toll, wenn man sich nicht erneut anmelden müsste, wenn man von WLAN zu mobilem Internet wechselt Webseite: generell: Übersichtlichere Gestaltung wäre hilfreich, damit man nicht für fast jede Anfrage die Suchfunktion benutzen muss
- Ausfälle füher bekannt geben
- Bessere Einbindung des primuss mit vollem Funktionsumfang
- Stabilere App
- Ein Zugang zum Hochschul-Mail-Account wäre super! Das fehlt mir total an der App.
- Hochschul email mit einbinden
- Teilen-Funktion der Stundenpläne als Bild
- Raumsuche und Chat funktionieren nicht (Android)
- Bugfixing (Bei Tippen auf einzelne Vorlesungen schließt die App sofort)
  Höhere Informationsdichte zu Vorlesungen (Anzeige von Blockveranstaltungen mit entsprechendem Datum)
- Android-App stürzt bei Chatfunktion immer ab,
  Raumsuche funktioniert leider nicht (bringt nur Fehlermeldung)
- Abrufen von Noten oder zumindest Push-Benachrichtigung bei Notenbekanntgabe

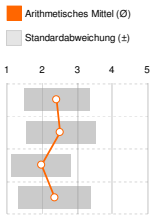
- Kalenderfunktion!
  Vor allem sodass die Initiativen der Hochschule ihre Termine veröffentlichen können und man alles auf einen Blick hat
- Es wäre schön, wenn man sich zu 100 % auf die Stundenplanänderungen der App verlassen könnte, ohne permanent die Website überprüfen zu müssen.
- Es wäre gut, wenn der Speiseplan auch in der iOS App sichtbar wäre
- Stundenplan nicht als Liste, sondern tabellarisch, so dass man auch freie Stunden optisch besser erkennen kann.
- Für Veranstaltungen, die nur für einen Tag angesetzt sind, fehlt oft das Datum in der Stundenplan App.
- Essensplan auch für
  iOS
- Die App sollte sich nicht selbst beenden, wenn man auf eine Vorlesung drückt/klickt
- - oft sind die Informationen in der Stundenplan-App nicht aktuell, d.h. es gab eine Raumänderung, die aber in der App nicht ersichtlich ist
- Bei der App für iOS funktioniert leider die Push-Benachrichtigung häufig nicht, weshalb ich eher auf die Hochschule-Website zugreife
- Unbedingt ein Mensa-Speiseplan auch bei apple Geräten! Nett wäre auch ein Speiseplan der zeigt welche der Mensa-Gerichte es in der cafete gibt.
- Einbindung der Hochschulemail
- Die meisten Features sind in der App für iOS nicht vorhanden. Daran sollte gearbeitet werden
- Für iOS App:
  - Speiseplan und Stundenplan in einer App zusammenfügen.

  Bei Android ist es ja auch möglich.
- Andere Hochschulveranstaltungen (zB die von Der Hochschule-Website (Veranstaltungskalender) oder der Initiativen) sollten abrufbar und dem eigenen Kalender hinzufügbar sein. Veranstaltungs-E-Mails finde ich nicht hilfreich, da zu viele E-Mails kommen. Ich schaue auf die Termine-Liste auf der Website. Aber die Termine in der App zu haben, wäre viel praktischer.
- Bugs beseitigen, Nutzlose Funktionen entfernen, Zuverlässigkeit erhöhen.
- Speiseplan für die ios App implementieren.
  Stundenplanänderungen sowohl bei permanenten Änderungen (Raum/Zeit) als auch bei einmaligen Änderungen und bei eingefügten Einmalveranstaltungen.

7. Wie sinnvoll fänden Sie folgende Features? *

Anzahl Teilnehmer: 204

	Lange überfällig! (1)		Hilfreich (2)		Keine schlechte Idee (3)		Bin ich nicht überzeugt... (4)		Total fehl am Platz (5)		Keine Ahnung, was gemeint ist... (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Einbindung der Modulhan...	30x	14,71	86x	42,16	54x	26,47	24x	11,76	2x	0,98	8x	2,40	0,93
Speiseplan Personalisier...	34x	16,67	63x	30,88	73x	35,78	27x	13,24	4x	1,96	3x	2,52	0,99
Vorlesungssuche (nach D...	66x	32,35	87x	42,65	40x	19,61	8x	3,92	1x	0,49	2x	1,97	0,85
Stundenplan exportieren...	46x	22,55	72x	35,29	55x	26,96	26x	12,75	4x	1,96	1x	2,36	1,03



8. Gibt es noch Features/Funktionen, die noch nicht genannt wurden, die Sie aber unbedingt in einer Hochschul App haben möchten?

Anzahl Teilnehmer: 29

- Wo wir als Studenten Rabatte bekommen (Restaurants, Bars, Freizeitaktivitäten, ...)!
- Termine der Hochschule und der Initativen. z. B. ein Kalender in der App, in der alle Verantstaltungen drinstehen (eventuell mit Syncronisation zu eigenen Kalender). Coming up Events, Kartenverkauf für Boom-Party, Diskussionsrunde der SUI, etc.
- Notizen zu einer Vorlesung anlegen, Prüfungsplan, Raumbelegung (wann ist der Raum frei und wann besetzt?), APO und SPO zum ausgewählten Studiengang, Abstimmung über die Vorlesungen im nächsten Semester
- Habe ich oben schon genannt.
  Die E-Mails der Hochschuladresse verwalten und beantworten.
- Aktuelle news rund um die Hochschule, eventuelle E-Mail Option um schnell einem Dozenten schreiben zu können? Eventuelle Moodle Integration?
- Suche/ Angebot für Mitfahrgelegenheiten oder das Teilen von z.B Bayernticket.
- Besonders für englischsprechende/internationale Studenten ist es oft schwer die Gerichte nachzuvollziehen.
  Vermutlich etwas schwierig lösbar, aber eine englische Übersetzung wäre oftmals hilfreich.
- Es wäre eine Exportfunktion der Nutzereinstellungen sinnvoll, sollte man sich vorher evtl. mühsam den Stundenplan zusammengestellt haben, der bspw. dann aus irgendwelchen Gründen auf Standardeinstellungen zurückgesetzt werden würde. Gleiches würde bspw. für den plattformübergreifenden bzw. -wechselnden Einsatz sinnvoll sein.
- Individueller Stundenplan machbar, wie in der Windowsapp mit verschiedenen Farben - zB Mathe rot und Bio grün
- Ne fancy PWA!
- Vielleicht auch ein Anbindung zur Hochschule Webseite und zu den SPO...
  Verknüpfung zur Hochschul-Mailadresse... (Lange überfällig!)
  Schwarzes Brett der Fakultäten mit anbinden (das findet man schon auf der Webseite immer so schwer bzgl. Prüfungseinsichten)
- CampusCard Abrechnung
- Buchen von Bib-Arbeitsräumen
- Studentenausweis über App per Kreditkarte o.ä. aufladen
- Speiseplan-Info für das Mittagsangebot in der Cafete wäre noch gut
- Nicht das ganze Modulhandbuch, aber Informationen zu Prüfungen wie Prüfungsart (StA, SchP90 etc) und erlaubte Hilfsmittel
- Man könnte eventuell das E-Mail Fach in die Hochschul-App einbinden (Android) Bisher braucht man dazu eine andere App
- Liste der Dozenten mit Kontaktinformationen (E-Mail, Telefonnummer) oder auch Telefonnummern für IT-Service, falls man mal Probleme mit Drucker oder sonstigem hat
- Prüfungstermine und -räume einbinden.
  Push benachrichtigen für Prüfungsanmeldung, Rückmeldung.
- Guthabenanzeige in der App
  (wieviel Geld noch drauf ist)
  Aufladen der Karte über die App mit PayPal nicht mit Bargeld
- - es gibt Blocktermine, die bleiben das ganze Semester über immer im Stundenplan drin... ich würde mir wünschen, dass so etwas rausfliegt, sobald der Termin rum ist... mit diesen Terminen ist es immer total unübersichtlich und der Stundenplan sieht immer so voll an.... außerdem ist nicht gleich ersichtlich, welche Termine noch anstehen (wie Vorlesungen am Samstag)
  - ich würde mir wünschen, dass man in der App auch seine Gruppen wählen kann... viele Vorlesungen im Master werden in Gruppen (Gruppe I und II) unterteilt und in der App stehen immer die Zeiten für beide Gruppen... das ist auch immer total unübersichtlich
- Hochschulmails aufrufen
- ein kleines persönliches notizfeld zu den jeweiligen stunden (ähnl. hausaufgabenheft in der schule) bzw allg. eine notizblockfunktion wäre evtl sinnvoll

  ansonsten finde ich die app bereits gut und übersichtlich
- Einbindung von Moodle ähnlich wie bei Primuss
- Raumsuche!! Ähnlich wie auf der Hochschuleseite differenzieren, nach Tag / Raum / Software
- Verknüpfung mit dem Studentenausweis, um aktuelles Guthaben einzusehen.
  Zugang, ähnlich wie beim Primuss Portal auch, für Onlinedienste der Hochschulbibliothek einrichten
- Eventuell Links zu wichtigen Hochschulseiten.
- evtl. Prüfungstermine abrufen
- Übersicht, welche Räume sind zur Zeit/in den nächsten paar Minuten/Stunden frei.

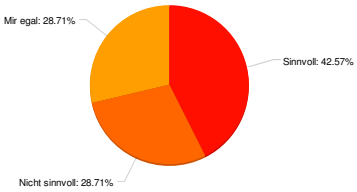
9. Wie sinnvoll fänden Sie eine Kalender Ansicht statt einer typischen Stundenplan Ansicht? *

Anzahl Teilnehmer: 202

86 (42.6%): Sinnvoll

58 (28.7%): Nicht sinnvoll

58 (28.7%): Mir egal

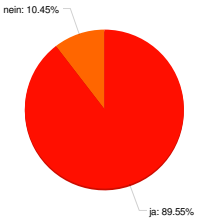


10. Hätten Sie die Sprachkurse gerne mit in den persönlichen Stundenplan integriert?

Anzahl Teilnehmer: 201

180 (89.6%): ja

21 (10.4%): nein



11. Fänden Sie es sinnvoll, aktuelle Veranstaltungen in eine kalendarische Ansicht der Hochschul App einzufügen? *

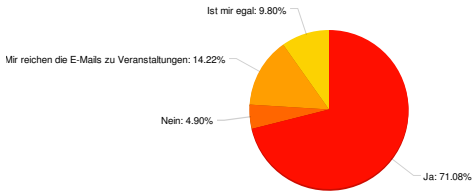
Anzahl Teilnehmer: 204

145 (71.1%): Ja

10 (4.9%): Nein

29 (14.2%): Mir reichen die E-Mails zu Veranstaltungen

20 (9.8%): Ist mir egal

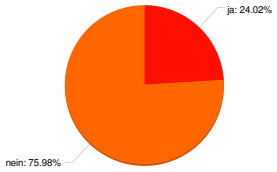


12. Finden Sie es sinnvoll, die Hochschul App in anderen Sprachen als nur Deutsch und Englisch anzubieten? *

Anzahl Teilnehmer: 204

49 (24.0%): ja

155 (76.0%): nein



13. Bitte geben Sie an, welche Sprachen sie als sinnvoll erachten.

Anzahl Teilnehmer: 34

- Bayrisch
- Z.B. Arabisch 😊
- Französisch
- Deutsch, Englisch
- Französisch
- Englisch
- englisch, französisch, spanisch, schwedisch
- Indisch
- Englisch
- Deutsch, Englisch
- Spanisch, Französisch
- Französisch
- Klingonisch, Englisch und wahrscheinlich Indisch :D
- alle sprachen die in der fh angeboten werden
- Indisch, Spanisch
- Spanisch, Englisch, Französisch
- Hindi
  Spanish
- Spanisch, Russisch
- englisch
- Englisch
- Spanisch, Türkisch, Hindi
- Indisch, Spanisch, Französisch
- Englisch
- englisch
- Indisch
- Deutsch, Englisch, Indisch, Russisch, Japanisch, Chinesisch
- indisch
- Englisch
- Englisch
- Indisch, Arabisch, Türkisch, Japanisch, Spanisch, Italienisch
- Spanisch, Französisch, Indisch
- Spanisch
- Indisch
- Spanisch vll.

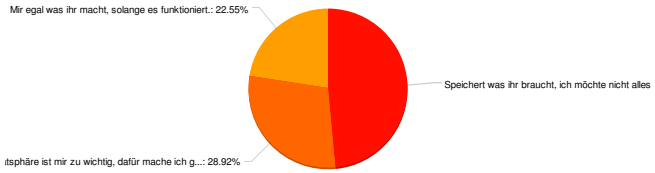
14. Bevorzugen Sie es, ein Account anzulegen, damit persönliche Einstellungen (Speiseplan, personalisierter Stundenplan, etc.) auf allen Geräten angezeigt werden können oder möchten Sie lieber keine Daten abspeichern und dafür die App an jedem Gerät neu einstellen? *

Anzahl Teilnehmer: 204

99 (48.5%): Speichert was ihr braucht, ich möchte nicht alles doppelt einstellen!

59 (28.9%): Privatsphäre ist mir zu wichtig, dafür mache ich gerne Abstriche.

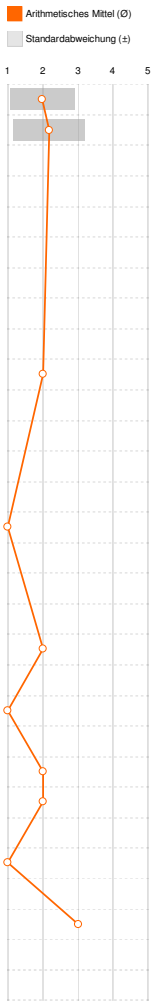
46 (22.5%): Mir egal was ihr macht, solange es funktioniert.



15. Falls Sie Technik interessiert sind und die entsprechenden Techniken kennen, für wie sinnvoll halten Sie diese?

Anzahl Teilnehmer: 158

	Super Idee!! (1)		Klingt spannend (2)		So muss es sein (3)		Davon würde ich abraten (4)		Nur über meine Leiche! (5)		Ich verstehe nur Bahnhof... (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Die Daten der App werde...	21x	13,46	32x	20,51	9x	5,77	4x	2,56	1x	0,64	89x	1,99	0,91
PWA (Progressive WebAp...	21x	13,46	34x	21,79	11x	7,05	10x	6,41	1x	0,64	79x	2,17	1,02
-	-	-	-	-	-	-	-	-	-	-	2x	-	-
.	-	-	-	-	-	-	-	-	-	-	3x	-	-
...	-	-	-	-	-	-	-	-	-	-	3x	-	-
/	-	-	-	-	-	-	-	-	-	-	1x	-	-
??	-	-	-	-	-	-	-	-	-	-	1x	-	-
a	-	-	-	-	-	-	-	-	-	-	2x	-	-
Ausversehen ausgewählt...	-	-	-	-	-	-	-	-	-	-	1x	-	-
H	-	-	1x	100,00	-	-	-	-	-	-	-	2,00	0,00
Hallo	-	-	-	-	-	-	-	-	-	-	2x	-	-
Hä	-	-	-	-	-	-	-	-	-	-	1x	-	-
Hört auf mir mit solchen...	-	-	-	-	-	-	-	-	-	-	1x	-	-
Hoffentlich klappt das wi...	-	-	-	-	-	-	-	-	-	-	1x	-	-
Kein AMP bitte, danke :)	1x	100,00	-	-	-	-	-	-	-	-	-	1,00	0,00
keine	-	-	-	-	-	-	-	-	-	-	5x	-	-
Keine Angabe	-	-	-	-	-	-	-	-	-	-	1x	-	-
l	-	-	-	-	-	-	-	-	-	-	1x	-	-
Nein	-	-	1x	14,29	-	-	-	-	-	-	6x	2,00	0,00
No	-	-	-	-	-	-	-	-	-	-	1x	-	-
Noah for President!	1x	100,00	-	-	-	-	-	-	-	-	-	1,00	0,00
Nope	-	-	-	-	-	-	-	-	-	-	1x	-	-
Onsen-UI ist generisch un...	-	-	1x	100,00	-	-	-	-	-	-	-	2,00	0,00
PWA+AMP	-	-	1x	100,00	-	-	-	-	-	-	-	2,00	0,00
sorry	-	-	-	-	-	-	-	-	-	-	1x	-	-
ups	1x	100,00	-	-	-	-	-	-	-	-	-	1,00	0,00
vorschläge	-	-	-	-	-	-	-	-	-	-	1x	-	-
Wehe ihr implementiert ...	-	-	-	-	1x	100,00	-	-	-	-	-	3,00	0,00
X	-	-	-	-	-	-	-	-	-	-	5x	-	-
yxy	-	-	-	-	-	-	-	-	-	-	1x	-	-



Internationale Umfrage

Survey for renewal of the Timetable App of Hof University

1. How do you look for information regarding the timetable, changes in your lesson schedule and the menu in the cantine? *

   Anzahl Teilnehmer: 29

   17 (58.6%): Android App

   4 (13.8%): iOS App (iPhone, iPad...)

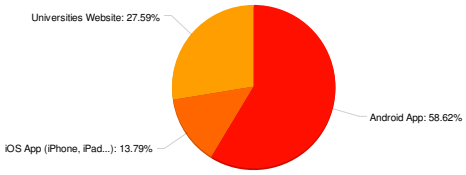
   - (0.0%): Windows Phone App

   8 (27.6%): Universities Website

   - (0.0%): Information from International Office

   - (0.0%): other (please specify)

   - (0.0%): Andere



2. If you use any of the Apps, how often do you use it? *

   Anzahl Teilnehmer: 29

   9 (31.0%): daily

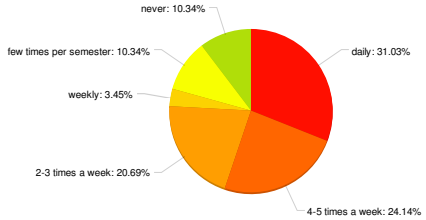
   7 (24.1%): 4-5 times a week

   6 (20.7%): 2-3 times a week

   1 (3.4%): weekly

   3 (10.3%): few times per semester

   3 (10.3%): never



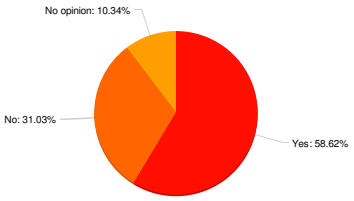
3. Do you feel that all neccessary information for your studies can be found on the english version of the universities website or the Apps? *

Anzahl Teilnehmer: 29

17 (58.6%): Yes

9 (31.0%): No

3 (10.3%): No opinion



4. How easy is it for you as an international to maintain a structured and complete time table for all your lessons? *

Anzahl Teilnehmer: 29

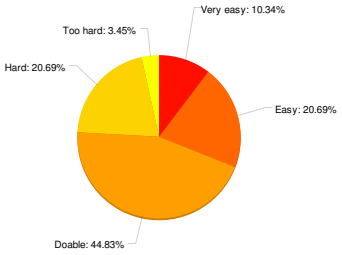
3 (10.3%): Very easy

6 (20.7%): Easy

13 (44.8%): Doable

6 (20.7%): Hard

1 (3.4%): Too hard

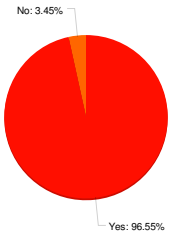


5. If the university had an Application which would fullfil all your needs better than the existing ones do, would you use it more frequently? *

Anzahl Teilnehmer: 29

28 (96.6%): Yes

1 (3.4%): No



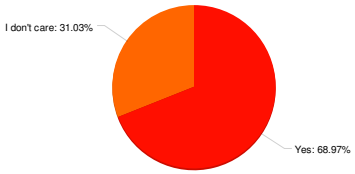
6. Would you prefer a platform independent application? *

Anzahl Teilnehmer: 29

20 (69.0%): Yes

- (0.0%): No

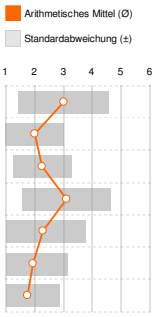
9 (31.0%): I don't care



7. How useful do you think the following features of the existing apps are? *

Anzahl Teilnehmer: 27

	Best idea ever! (1)		Important (2)		Ok (3)		Not important (4)		Totally unneccessary (5)		I never noticed that before... (6)			
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Ø	±
Chat feature for lessons	6x	22,22	4x	14,81	8x	29,63	5x	18,52	1x	3,70	3x	11,11	3,00	1,57
room search	7x	25,93	16x	59,26	3x	11,11	-	-	-	-	1x	3,70	2,00	1,00
Primuss Portal integratio..	5x	18,52	13x	48,15	8x	29,63	-	-	-	-	1x	3,70	2,26	1,02
Where am I? (Location o...	5x	18,52	4x	14,81	8x	29,63	6x	22,22	1x	3,70	3x	11,11	3,11	1,53
Bus / Train Schedules	10x	37,04	8x	29,63	4x	14,81	3x	11,11	-	-	2x	7,41	2,30	1,46
Calendar synchronisation	11x	40,74	12x	44,44	2x	7,41	-	-	1x	3,70	1x	3,70	1,93	1,21
Push notifications on les...	14x	51,85	9x	33,33	3x	11,11	-	-	-	-	1x	3,70	1,74	1,10



8. Do you have general improvement proposals for the existing apps?

Anzahl Teilnehmer: 4

- -No crashes on opening Chat(Android)
  -Dark mode(!)
- May be we can display our lecture timing in a particular day as outlook shows us if we have meeting coming with notification.
- All dishes under the menu are listed in German, maybe there could be an English version of the menu for international students who aren't affluent with the native language.
- The only one improvement:

  On the hof university website:

  A page that displays on a 2 weeks calendar (for the even and odd week) the schedule of the courses I select on a dropdown menu (with an option to select only modules that are taught in english). So I can easily see if I have overlapping slots and see if I have free slots to work freely on what I have.
  When I click on a course's slot on the calendar, it displays the information of the course (name, room, teacher, description, nb of ECTS, type of exam).
  And then, I can export this generated calendar for the whole semester on Outlook or Gmail, with only the name, the room and the teacher for each slot.

  OR

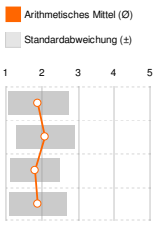
  Create a shared calendar for each course that all the students can download + add on their Outlook or Gmail account. So, if there is an update, the owner of the shared calendar update it, and all the students receive the modification on their smartphone.

  If something like this already exists, it's a big issue because I never heard about it before.

9. How useful do you think the possible new features would be? *

Anzahl Teilnehmer: 27

	I've been waiting for this! (1)		Helpful (2)		Not bad (3)		Useless (4)		Totally useless (5)		I don't understand the feature (0)		
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	Ø	±
Including module descrip...	10x	37,04	11x	40,74	5x	18,52	1x	3,70	-	-	-	1,89	0,85
Personalized Menu (only...	6x	22,22	13x	48,15	6x	22,22	1x	3,70	-	-	1x	2,08	0,80
Lesson search (by lectur...	9x	33,33	13x	48,15	4x	14,81	-	-	-	-	1x	1,81	0,69
Export/ print time table ...	10x	37,04	10x	37,04	7x	25,93	-	-	-	-	-	1,89	0,80



10. Are there any feature suggestions that you have for an university wide application?

Anzahl Teilnehmer: 2

- -Option to look up Lecturers and their information (i.e. email, room, fax, phone)
  -More detailed and updated map of campus (+GPS?)
  -Open source or feature submissions
- - I read the description only one time when I select my courses (and the description is a long text for modules I took, so not useful for a mobile device).
  - Export to PDF or Excel. For the pdf, I don't understand in which usecase it could be useful, For the Excel export, could be useful but for a limited range of usecases.

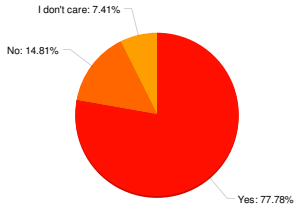
11. Would you prefer a calendar instead of a classic time table? *

Anzahl Teilnehmer: 27

21 (77.8%): Yes

4 (14.8%): No

2 (7.4%): I don't care



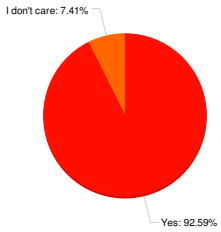
12. Would you like to have the language courses included into your regular time table? *

Anzahl Teilnehmer: 27

25 (92.6%): Yes

- (0.0%): No

2 (7.4%): I don't care



13. Would you like to have pending events included into a calendar view of the app? *

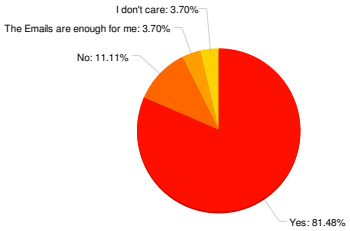
Anzahl Teilnehmer: 27

22 (81.5%): Yes

3 (11.1%): No

1 (3.7%): The Emails are enough for me

1 (3.7%): I don't care

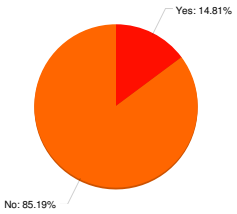


14. Would you offer an university wide application in any other languages than english and german? *

Anzahl Teilnehmer: 27

4 (14.8%): Yes

23 (85.2%): No



15. Please specify the languages you woud like to have included.

Anzahl Teilnehmer: 6

- hindi
- Arabic, Spanish
- If you make this application, according to the questions asked, it would be great
- French
- English, German
- English, German

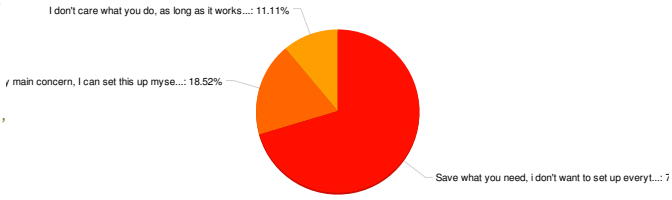
16. Would you rather prefer to register an Account so that personal settings (personalized time table or menu) can be synchronized over all your devices or would you rather not save your data and set up all your devices yourself? *

Anzahl Teilnehmer: 27

19 (70.4%): Save what you need, i don't want to set up everything all the time

5 (18.5%): Privacy is my main concern, I can set this up myself

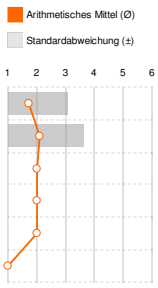
3 (11.1%): I don't care what you do, as long as it works...



17. If you are into IT and know the referrenced techniques, how useful would you rate them?

Anzahl Teilnehmer: 18

	Great idea! (1)		Kinda useful (2)		That's how it's supposed to be (3)		I wouldn't do it this way... (4)		No way!! (5)		I don't know what you are talking about... (6)			
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Ø	±
The data will be provide...	12x	66,67	3x	16,67	1x	5,56	1x	5,56	-	-	1x	5,56	1,72	1,36
PWA (Progressive WebAp...	9x	50,00	4x	22,22	1x	5,56	3x	16,67	-	-	1x	5,56	2,11	1,49
-	-	-	1x	100,00	-	-	-	-	-	-	-	-	2,00	0,00
AMP for the lightweight, ...	-	-	1x	100,00	-	-	-	-	-	-	-	-	2,00	0,00
AMP Pages	-	-	1x	100,00	-	-	-	-	-	-	-	-	2,00	0,00
None	1x	100,00	-	-	-	-	-	-	-	-	-	-	1,00	0,00



Autoren Referenz

Diese wissenschaftliche Arbeit ist eine Zusammenarbeit der beiden Autoren Dennis Brysiuk und Noah Lehmann. Um daher einen besseren Überblick über die Zuordnung der Inhalte zu haben wird im folgenden eine Tabelle dargestellt, die die Kapitel ihren jeweiligen Verfassern zuordnet.

Kapitel Nr.	Kapitel Bezeichnung	Autor
1	**Einleitung**	
1.1	Beweggründe	Noah Lehmann
1.2	Zielsetzung	Noah Lehmann
1.3	Zielgruppe	Noah Lehmann
1.4	Vorausgesetztes Wissen	Noah Lehmann
1.5	Vorgeschlagene Nebenlektüre	Noah Lehmann
2	**Auswertung vorhandener Anwendungen**	
2.1	Allgemeine Auswertung der Zufriedenheit	Noah Lehmann
2.1.1	Zielgruppen	Noah Lehmann
2.1.2	Ergebnisse	Noah Lehmann
2.1.3	Fazit	Noah Lehmann
2.2	Anwendungen	Noah Lehmann
2.2.1	Android	Noah Lehmann
2.2.2	iOS	Noah Lehmann
2.2.3	Windows App	Noah Lehmann
2.2.4	Website	Noah Lehmann
2.2.5	Fazit	Noah Lehmann
3	**Funktionale Anforderungen**	
3.1	Auftraggeber	Noah Lehmann
3.1.1	Grundlegenden Anforderungen	Noah Lehmann
3.1.2	Funktionale Anforderungen	Noah Lehmann
3.2	International Office	Noah Lehmann
3.2.1	Problem	Noah Lehmann
3.2.2	Funktionale Anforderungen	Noah Lehmann

Tabelle IX.3: Autoren Referenz

Kapitel Nr.	Kapitel Bezeichnung	Autor
3.3	Sprachenzentrum	Noah Lehmann
3.3.1	Problem	Noah Lehmann
3.3.2	Funktionale Anforderungen	Noah Lehmann
3.4	Pflichtenheft	Noah Lehmann
4	**Architektur von Softwaresystemen**	
4.1	Grundlegendes	Dennis Brysiuk
4.2	Nicht-funktionale Anforderungen	Dennis Brysiuk
4.3	Prinzipien	Dennis Brysiuk
4.3	Fazit	Dennis Brysiuk
5	**Web Services**	
5.1	Serviceorientierte Architektur	Dennis Brysiuk
5.1.1	Technische Konzepte	Dennis Brysiuk
5.1.2	Zusammenfassung	Dennis Brysiuk
5.2	Webservice Arten	Dennis Brysiuk
5.2.1	Simple Object Access Protocol (SOAP)	Dennis Brysiuk
5.2.2	Representational State Transfer (REST)	Dennis Brysiuk
5.2.3	Bewertung	Dennis Brysiuk
6	**REST API Design**	
6.1	Ressourcen Design	Dennis Brysiuk
6.1.1	Versionsverwaltung	Dennis Brysiuk
6.1.2	URI Namenskonvention	Dennis Brysiuk
6.1.3	Ressourcen Namenskonvention	Dennis Brysiuk
6.1.4	HTTP Methoden	Dennis Brysiuk
6.2	Request Struktur	Dennis Brysiuk
6.2.1	Header	Dennis Brysiuk
6.2.2	Query Parameter	Dennis Brysiuk
6.2.3	Aufteilung einer Ressource	Dennis Brysiuk
6.3	Response Handling	Dennis Brysiuk
6.3.1	Response Inhaltsformat	Dennis Brysiuk
6.3.2	HTTP Statuscode	Dennis Brysiuk
6.3.3	Asynchrone Callbacks	Dennis Brysiuk
6.4	Hypermedia	Dennis Brysiuk

Tabelle IX.4: Autoren Referenz

Kapitel Nr.	Kapitel Bezeichnung	Autor
7	**Architektur Hochschul-App**	
7.1	Serviceorientierte Architektur	Dennis Brysiuk
7.2	Microservice Architektur	Noah Lehmann
7.3	Schichtenarchitektur	Noah Lehmann
8	**Services der Hochschul-App**	
8.1	Stundenplan (Timetable-Service)	Noah Lehmann
8.2	Planänderungen (Timetable-Change-Service)	Noah Lehmann
8.3	Speiseplan (Mensa-Service)	Dennis Brysiuk
8.4	Benachrichtigungen (Notification Service)	Dennis Brysiuk
8.5	Sicherheit (Auth-Service)	Dennis Brysiuk
8.6	Anwenderverwaltung (User-Service)	Dennis Brysiuk
8.7	Weitere Dienste	Noah Lehmann
9	**Ausblick und Fazit**	
9.1	Ausblick	Noah Lehmann
9.2	Weitere Bachelorarbeitsthemen	Noah Lehmann
9.3	Fazit	Noah Lehmann

Tabelle IX.5: Autoren Referenz

Literaturverzeichnis

Broderdörp, K.; P. D. F. Lender u. a. (1. Feb. 2012): “campuls [02][2012]”. In: Das Infomagazin der Hochschule Hof. URL: https://www.hof-university.de/fileadmin/user_upload/hochschulkommunikation/campuls_2012_02_E-Mail.pdf

Brysiuk, D.; N. Lehmann, Hrsg. (2019a): Umfrage zur Nutzung der Hochschul-Apps. Hrsg. von D. Brysiuk; N. Lehmann

– **(2019b)**: “Web-basierte Hochschul-App (Praxisarbeit). Modulare Web-Architektur”. Hochschule für angewandte Wissenschaften Hof

Dowalil, H. (2018): Grundlagen des modularen Softwareentwurfs. Der Bau langlebiger Mikro- und Makro-Architekturen wie Microservvices und SOA 2.0. München: Carl Hanser Verlag

Glaser, A. (2019b): “Web-basierte Hochschul-App (Praxisarbeit). Authentifizierung und Personalisierung”. Hochschule für angewandte Wissenschaften Hof

– **(2019a)**: “Web-basierte Hochschul-App. Authentifizierung und Personalisierung”. Hochschule für angewandte Wissenschaften Hof

Josuttis, N. (2008): SOA in der Praxis. Heidelberg: dpunkt.verlag GmbH

Kiltz, F. (2010): Java Webservices. Frechen: Mitp Verlag

Melzer, I. (2007): Service-orientierte Architekturen mit Web Services. 2. Aufl. Heidelberg: Elsevier Spektrum Akademischer Verlag

Parnas, D. L. (2001): On the Criteria to Be Used in Decomposing Systems into Modules. Pioneers and Their Contributions to Software Engineering: Conference on Software Pioneers, Bonn, June 28/29, 2001, Original Historic Contributions. Hrsg. von M. Broy; E. Denert. Berlin, Heidelberg: Springer Verlag

Rich, B. R. (1995): Clarence Leonard (Kelly) Johnson. A Biographical Memoir. Washington D.C.: National Acedemies Press

Schill, A.; T. Springer (2012): Verteilte Systeme. Grundlagen und Basistechnologien. 2. Aufl. Heidelberg: Springer Verlag

Internetquellen

APIs show Faster Growth Rate in 2019 than Previous Years (2019): APIs show Faster Growth Rate in 2019 than Previous Years. URL: https://www.programmableweb.com/news/apis-show-faster-growth-rate-2019-previous-years/research/2019/07/17 (besucht am 19. 08. 2019)

AWS Elastic Beanstalk (2019): AWS Elastic Beanstalk. URL: https://docs.aws.amazon.com/de_de/elasticbeanstalk/latest/dg/Welcome.html (besucht am 02. 09. 2019)

HTTP/Anfragemethoden (2018): HTTP/Anfragemethoden. URL: https://wiki.selfhtml.org/wiki/HTTP/Anfragemethoden (besucht am 06. 10. 2019)

Hochschule Hof - University of Applied Sciences (2019): Hochschule Hof - University of Applied Sciences. Leitbild der systemakkreditierten Hochschule Hof. Hochschule für angewandte Wissenschaften Hof. URL: https://www.hof-university.de (besucht am 13. 08. 2019)

Kantar (2019a): Anzahl der verfügbaren Apps in den Top App-Stores im 2. Quartal 2019. Hrsg. von Statista. Statista GmbH. URL: https://de.statista.com/statistik/daten/studie/208599/umfrage/anzahl-der-apps-in-den-top-app-stores/ (besucht am 26. 08. 2019)

– **(2019b)**: Bitkom, Nutzung von Cloud Computing in Unternehmen in Deutschland in den Jahren 2011 bis 2018. Hrsg. von Statista. Statista GmbH. URL: https://de.statista.com/statistik/daten/studie/177484/umfrage/einsatz-von-cloud-computing-in-deutschen-unternehmen-2011 (besucht am 14. 08. 2019)

– **(2019c)**: Marktanteile der mobilen Betriebssysteme am Absatz von Smartphones in Deutschland im 2. Quartal der Jahre 2018 und 2019. Statista GmbH. URL: https://de.statista.com/statistik/daten/studie/177484/umfrage/einsatz-von-cloud-computing-in-deutschen-unternehmen-2011 (besucht am 15. 10. 2019)

Netflix-Fallstudie (2016): Netflix-Fallstudie. URL: https://aws.amazon.com/de/solutions/case-studies/netflix/ (besucht am 02. 09. 2019)

OASIS SOA Reference Model TC (2005): OASIS SOA Reference Model TC. URL: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm (besucht am 15. 08. 2019)

Reference Architecture Foundation for Service Oriented Architecture V. 1.0 (2012): Reference Architecture Foundation for Service Oriented Architecture V. 1.0. Committee Specification 01. URL: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.html#_Toc343761540 (besucht am 15. 08. 2019)

Simple Object Access Protocol (SOAP) 1.1 (2000): Simple Object Access Protocol (SOAP) 1.1. URL: %https://www.w3.org/TR/2000/NOTE-SOAP-20000508/ (besucht am 19. 07. 2020)

Softwarearchitektur (2019): Softwarearchitektur. URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/Members/sinz (besucht am 28. 08. 2019)

Sprachenzentrum (2019): Sprachenzentrum. Hochschule für angewandte Wissenschaften Hof Sprachenzentrum. URL: https://www.hof-university.de/studierende/info-service/sprachenzentrum/ (besucht am 02. 09. 2019)

Was sind Docker-Container? (2017): Was sind Docker-Container? URL: https://www.dev-insider.de/was-sind-docker-container-a-597762/ (besucht am 02. 09. 2019)

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich meinen Beitrag zur vorliegenden Gruppenarbeit (Kapitel 4, 5, 6, 7.1, 8.3, 8.4, 8.5, 8.6) selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; das gleiche gilt für die von den auf dem Titelblatt der Arbeit genannten Autoren gemeinsam verfassten Teile (Kapitel 1, 2, 3, 7.2, 7.3, 8.1, 8.2, 8.7, 9). Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde nach meiner besten Kenntnis bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Hof, den 03.12.2019

Abstract

Die vorliegende Bachelorarbeit zielt darauf ab, die Analyse für eine zukunftsfähige, beständige und erweiterbare Hochschulapp bereitzustellen. Um eine solide Grundlage dafür zu schaffen, werden in dieser Arbeit ebenfalls bestehende Anwendungen zur Informationsbeschaffung der Hochschule Hof analysiert und die Nutzergruppen zu deren Präferenzen befragt. Aus den gewonnenen Ergebnissen werden dann die Anforderungen gesammelt und anhand derer die finalen Entwürfe der Anwendung gefertigt und gerechtfertigt.
Der erste Teil der Arbeit befasst sich lediglich mit der Analyse der vorhandenen plattformgebundenen Anwendungen der Hochschule Hof, unter anderem der Android App, der iOS App aber auch mit der klassischen Website der Hochschule Hof. Es werden die Funktionen gesammelt und analysiert. Danach werden die Nutzer der Apps, die Studierenden der Hochschule Hof, zu den bestehenden Anwendungen befragt. Anhand der gewonnenen Erkenntnisse können die gewünschten Funktionen dann gesammelt und bewertet werden.
Mit den Informationen aus dem ersten Teil der Arbeit werden dann die Anforderungen an die Arbeit gesammelt. Dabei wird nicht nur der Auftraggeber mit einbezogen, sondern auch andere Interessengruppen, genauer das International Office und das Sprachenzentrum der Hochschule Hof. Zu den wichtigsten Anforderungen kommt die Plattformunabhängigkeit der Anwendung und die Internationalisierung in Form von Mehrsprachigkeit im Einklang mit dem Leitbild der Hochschule Hof. Zu den Anforderungen wird außerdem ein referenzierbares Lastenheft angefertigt.
Um die nicht funktionalen Anforderungen wie Erweiterbarkeit und Modularität gerecht zu werden, wird im Mittelteil der Arbeit der Fokus auf einige Prinzipien und Designphilosophien der Softwarearchitektur gelegt. Hier werden Ideen wie das KISS-Prinzip, Serviceorientierte Architektur und REST genauer betrachtet, teils auch mit Alternativen verglichen, um die passende Lösung für einen Prototypen einer neuen Hochschul-App zu finden. Durch die Identifizierung RESTful Webservices als Architekturgrundlage wird im Anschluss an die Sammlung der Paradigmen ebenfalls ein Regelwerk für das Arbeiten mit REST festgelegt.
Abschließend wird konkret an der Lösung der Hochschul-App gearbeitet. Hierfür wird erst die grundlegende Architektur angerissen und genauer erläutert. Lösungen wie Discovery Services und API-Gateways werden dort genauer erläutert. Abschließend wird dann der konkrete Aufbau der Microservice Architektur erklärt. Hierbei werden nicht nur die Ressourcen definiert, es wird auch auf die darunterliegende Logik und den Datenbankkonzepten eingegangen.
Zusammenfassend erarbeitet diese Bachelorarbeit den konkreten Prototypen der Hochschul-App in allen Phasen der Analyse. Durch die Ideen aus den vorhandenen Anwendungen und dem Feedback der Nutzer konnten die Anforderungen gesammelt werden. Aus diesen Anforderungen konnten dann die nötigen Architekturprinzipien identifiziert werden und daraus resultierend die passende Architektur gefunden werden. Aufbauend auf der Architektur konnten dann die Feinheiten der App ausgearbeitet werden.