Tabea Schuster (5038307), Moumita Ahmad (5027729), Niels Wennesheimer (5033302)

Im Modul Datenbankbasierte Web-Anwendungen haben wir, Tabea Schuster (5038307), Moumita Ahmad (5027729) und Niels Wennesheimer (5033302), uns dazu entschieden, eine Webseite für die Bewohner einer Fantasie-Welt zu erstellen, auf der sie “vorbildliches Verhalten” festhalten und belohnen können. Dazu können die Nutzer Tugenden erfüllen, um ihren SocialScore zu erhöhen und untereinander Dienste anfragen.

Dieser Idee sind wir im Verlauf des Semesters auch treu geblieben. Lediglich der Punkt der zunächst angedachten Bewertungen ist am Ende rausgefallen, da er für uns eine niedrigere Priorität hatte und wir die Zeit lieber für die Optimierung und Erweiterung der anderen Features nutzen wollen.

Die Arbeitsaufteilung war variabel und nicht fest verteilt. So konnte sich jeder in allen Bereichen mit einbringen.

Zunächst haben wir uns alle zusammen darum gekümmert, dass Angular, node.js und MySQL bei jedem von uns läuft. Dann haben wir auch schon unser github-Repository aufgesetzt. Die ersten Komponenten haben wir noch zusammen erstellt, um mit Angular vertraut zu werden.

Dann hat die Aufteilung schon begonnen. Jeder hat sich Punkte von der ToDo-Liste genommen und diese umgesetzt. Angefangen hat Moumita zum Beispiel mit der Anmeldung und der Registrierung. Niels hat sich zunächst um die Datenbankabfragen der Dashboards und der dementsprechenden Darstellung gekümmert und Tabea ist dort mit eingestiegen und hat sich für den Beginn mit den POST-Methoden und dem Erstellen und Bearbeiten von Tugenden beschäftigt.

Danach wurden alle weiteren Funktionalitäten nach und nach implementiert und immer mal wieder Design-Verbesserungen vorgenommen. Durch unseren XD-Prototyp hatten wir eine gute Planung, was wir alles benötigen. Während der Umsetzung haben wir diese Planung ergänzt, falls nötig.

Die Arbeitsaufteilung war ziemlich gleichmäßig und es lässt sich schwer einschränken und definieren, wer genau was gemacht hat, da jeder überall mal etwas getan hat.

Zur Clientseitigen Umsetzung haben wir mit Angular gearbeitet. Veranlasst dazu haben uns natürlich grundsätzlich die Lehrveranstaltungen zu diesem Framework. Aber nachdem man sich mit Angular mehr und mehr auseinandergesetzt hat, hat man die Stärken immer mehr gespürt und zu nutzen gewusst. Durch den Aufbau und die Verknüpfung von/durch Komponenten war es möglich, die Anwendung in kleine, wiederverwendbare, gekapselte Elemente zu zerlegen. Dadurch war es leicht, den Überblick zu behalten und auch ohne Konflikte zu erzeugen, mit mehreren Personen gleichzeitig daran zu arbeiten.

Für die serverseitige Entwicklung verwendeten wir das Framework express.js auf Basis der Laufzeitumgebung node.js und MySQL zur Datenbankanbindung. Dadurch konnten wir nah an den Vorlesungen arbeiten und hatten wenige Probleme beim Einrichten und Verwenden des Servers und der Datenbank.

Auch wenn es kein Framework ist, möchten wir hier das Tool Postman erwähnen, das wir während der Entwicklung verwendet haben. Die Datenbankabfragen haben wir zunächst in der MySQL Workbench selbst getestet und dann die dazu erstellten Endpunkte unseres Servers mit Hilfe von Postman. Erst im Anschluss daran haben wir sie aus der Anwendung heraus getestet. Dadurch folgten wir einem strukturierten Vorgehen und war es leichter, die Fehler zu identifizieren und zu beheben.

Um ein einheitliches Design zu schaffen und Zeit für die Endwicklung des Backend zu gewinnen, haben wir uns entschieden das Open Source CSS-Framework Bulma (https://bulma.io) zu verwenden. Durch viele vorgegebene CSS-Klassen, die den jeweiligen Elementen nur zugeteilt werden müssen, war das Styling schnell anbringbar und es mussten

immer wieder nur Kleinigkeiten angepasst werden. Zudem haben wir mit Icons von Fontawesome gearbeitet.

Mit Hilfe von Socket.IO haben wir am Ende noch die Echtzeit-Kommunikation zwischen Web Server und zwischen den Clients mit Websockets implementiert. Dadurch ist es nun möglich wichtige Informationen direkt in Echtzeit zu aktualisieren, ohne dass der Nutzer die Seite erst neu laden muss. Zum Beispiel, wenn ein Tugendhafter eine neue Anfrage erhält, zu einem von ihm angebotenen Dienst. Ebenfalls sind die Antworten des Tugendhaften auf die Anfrage sofort beim Suchenden zu sehen und die entsprechende Komponente der Seite wird aktualisiert. Dabei werden jeweils auch die kleinen Notification-Zähler am Dashboard (in der Menüleiste) aktualisiert und ein kleines Notification-Banner erscheint rechts oben. Die dritte notwendige Echtzeit-Kommunikation findet bei der Ausschüttung der Bonusprogramme statt. Tugendhafte werden somit sofort benachrichtigt, wenn ein Ältester ein Bonusprogramm auslöst und es auf jenen zutrifft.
Die gerade beschriebenen Funktionen waren die, für die Echtzeitaktualisierung am wichtigen ist. Es gibt noch weitere Stellen, für die man Echtzeit-Aktualisierungen anwenden kann. Da wir die letzten auftretenden Probleme bei der Umsetzung des Websockets erst kurz vor Abschluss des Projekts lösen konnten, haben wir nicht alle weiteren Stellen umsetzen können. Exemplarisch haben wir es jedoch noch einmal für eine reine Informationsseite implementiert. Wenn ein Tugendhafter einen Dienst erstellt oder wiederherstellt, erscheint dieser auch direkt bei anderen Clients, die sich zu diesem Zeitpunkt auf der Seite “Dienste” befinden. Nach demselben Prinzip würden auch die Aktualisierung von Bonusprogrammen, Tugenden und anderen informativen Teilen und Komponenten erfolgen (bei Erstellen, Bearbeiten, Löschen). Da dies aber nicht weiter technisch interessant, sondern reine Fleißarbeit ist und wir schon umfangreich Sockets eingebaut haben, haben wir diese erst einmal ausgelassen.

Bis auf die genannten Libraries, haben wir keinen direkten Fremd-Code übernommen. Natürlich haben wir uns Tutorials zu neuen Themen und etablierte Vorgehensweisen angesehen, um zu verstehen wie etwas funktioniert. Bei der Umsetzung haben wir aber stets auf die eigenständige, an unsere App und Anwendungszwecke angepasste Umsetzung geachtet.

Im Folgenden wollen wir noch einmal grob auf die Projektstruktur in der Angular-App eingehen. Im Ordner “app” werden alle Komponenten verwaltet.
Es gibt folgende Unterordner:

- models, in dem sich die Models der Daten-Klassen befinden,
- modules, in dem sich alle Komponenten befinden und
- services, in dem sich alle Services befinden.

Durch diese Struktur ist das Projekt trotz vieler Komponenten und Dateien übersichtlich und logisch sortiert.
Die Komponenten sind zunächst weiter nach Punkten aus der Menüleiste gegliedert. Dort werden die Komponenten in immer kleinere Gruppen zerlegt, wie im Dashboard in die Nutzergruppen (Tugendhafter, Suchender, Ältester) und anschließend auch in immer speziellere Funktionen.

Im Ordner Services befinden sich zum einen die Services, die die Schnittstelle zum Server bilden und die dortigen Daten-Endpunkte ansprechen (“data”-Services, z.B. buerger.service oder dienst.service) und zum anderen “utility”-Services, die für bestimmte zentrale Aufgaben zuständig sind. Ein Beispiel für letzteres ist der auth.service der Authentifizierungsdienste beinhaltet, wie das Registrieren und Anmelden, mit der Verwaltung der local Storage Variablen in der Session. Der do-update.service wird für abonnierte Aktualisierungen zwischen Komponenten innerhalb der App verwendet (mittels Observern). Methoden dieses Services werden wiederum vom websocket.service mit genutzt. Dieser ermöglichst die Kommunikation zwischen den Clients mithilfe von Websockets via den Server.

Um unsere Anwendung zu starten benötigen Sie ein Command Line interface und unser git-Repository (https://github.com/NiJoW/NiJoW.github.io.git, Repository ist auch als zip-Datei angefügt).

Zudem brauchen sie eine VPN-Verbindung zur Hochschule, da dort unser Datenbank-Server liegt. Wir haben den „cisco AnyConnect Secure Mobility Client“ verwendet. Steht diese Verbindung und sind die Projektdateien lokal vorhanden, kann es gestartet werden.

Dazu muss in der Konsole zum Ordner des Angular Projektes navigiert werden (bis xxx/fantasy-score-app). Dort müssen sie zunächst die folgenden Befehle eingeben:

1. npm install --> zum Installieren der Dependencies
2. ng build --> um den Auslieferungscode zu generieren
3. ng serve --> um die App zu starten

Es müsste nun die Ausgabe erfolgt sein, dass die Anwendung unter der URL http://localhost:4200/ zu erreichen ist.

Damit die Anwendung korrekt funktioniert, muss der Server noch gestartet werden. Öffnen sie ein weiteres Konsolen-Fenster, navigieren sie in die App (bis xxx/fantasy-score-app) und führen den folgenden Befehl aus:

1. node server.js

In diesem Fenster müsste nun die Ausgabe stehen, dass der Server auf dem Port :8080 läuft.

Im Browser (wir haben meistens Google Chrome oder Safari benutzt, es sollten aber auch genauso alle anderen funktionieren) geben sie als URL nun http://localhost:4200/ ein und die Anwendung läuft.

Damit sie aus jeder Nutzergruppe einen Benutzer haben, mit dem sie anfangen können, sich durch die Anwendung zu klicken, haben wir ihnen hier drei Bürger aufgelistet.
Zur Vereinfachung haben wir jedem Nutzer zunächst dasselbe Passwort gegeben (sie können es in der Anwendung jederzeit ändern).

Tugendhafter:	Suchender:	Ältester:
Benutzername: moumita	Benutzername: tabea	Benutzername: niels
Passwort: 12345	Passwort: 12345	Passwort: 12345

→ Wollen sie trotzdem einen neuen Ältesten registrieren, ist das hier der Code der benötigt wird: 43173573r! (einfach in das Feld „Code“ im Registrieren-Form des Ältesten eintragen)

→ Sie werden bei der Benutzung über viele andere Nutzernamen stolpern. Sie können sich auch jederzeit mit diesen Nutzern anmelden (Passwort ebenfalls „12345“)!

Die unterschiedlichen Rollen und Privilegien sollten weiterhin denen entsprechen, die wir in unserer Abgabe zu Beginn des Semesters definiert haben.