257.130 & 257.131

Communications Engineering

(Kompetenztraining)

SS 2020



Themenbereich

**Gruppe 9**

**Projektleitung:**

K11808821, Thomas Deutsch, [deutsch-thomas@gmx.at](mailto:deutsch-thomas@gmx.at)

**Team:**

K11818868, Stefan Bolda

K11818867, Stefan Hinterhölzl

Erfahrungsbericht - Projekttitel

# Eine kurze Geschichte des CE KT im Sommersemester 2020

In der ersten Präsenzeinheit hatten wir die Möglichkeit uns eigenständig in 3er Gruppen zu finden und eine selbstgewählte Projektidee zu verwirklichen. Da wir uns schon aus dem bisherigen Studienverlauf kannten, war die Wahl der Teamkollegen nicht schwer. Gemeinsam tüftelten wir diverse Ideen aus und entschieden und schlussendlich für shortly. Grund dafür war das gemeinsame Interesse an Web Development, welche in unserem Studiengang ansonsten nicht teil gewesen wäre, als auch die Möglichkeit diese Idee mit etwas zusätzlicher privater Entwicklungszeit auch marktreif zu gestalten, und somit eventuell erste Entrepreneur-Erfahrungen zu sammeln. Die größte Hürde hierbei, war jedoch, dass niemand von uns Erfahrungen mit Web Development hatte. Deshalb war für uns der Beginn Projektes mit sehr viel Eigenständigem recherchieren, lesen und lernen verbunden. Dabei ergab sich als Grundstruktur der Aufbau, welcher einem MVC Pattern gleicht. Für das Frontend gab es diverse Möglichkeiten, komplett ohne spezielles Framework arbeiten, oder doch mit Frameworks wie Angular, React, Ember, Vue et cetera. Schlussendlich haben wir uns für Angular entschieden, aus dem einfachen Grund, dass ein Projektbeteiligter dies für sein Sommerpraktikum benötigt, und sich dieser Kurs als gute Lernbasis empfahl. Für das Styling der Webpage haben wir uns aus persönlichem Geschmack für eine Kombination aus Materialize und Bootstrap entschieden.

Für das Backend ergab unsere Recherche den Einsatz von einer Datenbank in Verbindung mit einer API (application programming interface).

Ähnlich wie zum Frontend gab es auch hier unzählige Möglichkeiten diese Umzusetzen. Obwohl für kleinere Websites, wie es für dieses Projekt der Fall ist, das Internet so gut wie immer MongoDB als Datenbank empfiehlt, entschlossen wir uns dennoch aufgrund unserer Erfahrungen mit SQL für die SQLite Datenbank. Diese Datenbank ist sehr schnell aufgesetzt, und ist kostenfrei verwendbar für unser Projekt. Weiters könnte die Bestehende Logik nur mit minimalen Änderungen auf eine andere SQL Datenbank umgelegt werden, was das Projekt skalierbar macht. Die Wahl der API war in unserem Fall recht schnell geklärt, wir waren uns von Anfang an einig, dass es eine REST API werden soll mit JSON um asynchrone Kommunikation zu ermöglichen, dies ist für uns auf der Seite, welche das Weiterleiten übernimmt, wichtig. Da die REST API sehr gut dokumentiert ist, haben wir uns auch keine zu großen Sorgen gemacht, diese zu erstellen.

Die nächste, zu treffende, Entscheidung war es, die Entwicklungsumgebung zu wählen. Obwohl Eclipse nach wie vor unsere „Heimat“ und sicherer Hafen ist, waren wir uns einig, dass es nicht für Web Development optimiert ist. Schlussendlich entschieden wir uns für die Wahl von Visual Code, aufgrund der Leichtgewichtigkeit und einfachen Anbindung an GitHub. Die genaue Recherche sowie Vor und Nachteile aller von uns untersuchten Entwicklungsumgebungen, APIs, Frameworks können bei Interesse im Dokument „Theoretische\_Ausarbeitung“ eingesehen werden. Um als Team am Projekt arbeiten zu können, haben wir GitHub verwendet, somit konnten wir den Fortschritt gut verwalten und bei begangen Fehlern ohne große Mühen zurückrudern.

Da wir nun mal Studenten sind und kein Geld für Server Hosting bezahlen wollen, hat glücklicherweise Stefan Hinterhölzl bei sich zuhause einen alten Computer, welchen wir zu einem Server umfunktioniert haben. Durch remote Desktop hatten wir alle Zugriff und so konnte jeder von uns den Server, die API, oder das Frontend starten und konfigurieren. Mittels einem gratis DDNS Services war es möglich die Dynamische IP zu verfolgen, und auf permanenten Zugriff zu gewähren, ohne täglich die neue IP Mitteilen zu müssen. Somit haben GitHub und die Remote Desktop Anbindung eine schnelle und einfache Zusammenarbeit ermöglicht.

Da ab diesem Punkt die groben Einzelheiten geklärt waren, begann wir mit dem Erstellen der SQLite Datenbank und der REST API. Für die REST API wurde node.js verwendet um alle Packages des Projekts zu verwalten. Somit konnten Packages, welche das übersetzen der Daten in SQL übernahmen und JSON lesen/schreiben sehr schnell ermöglichten. Für die Erreichbarkeit der API wurde Nodemon eingesetzt, welche die API als Server erreichbar machte. Wie angenommen, war dieser Teil des Projekts sehr gut dokumentiert, so konnten wir diversen Anleitungen folgen und mit kleineren Änderungen die API für unser Projekt anpassen. Somit konnten wir mit Hilfe von Postman die ersten Funktionen testen, und erfolgreich implementieren. Schnell war es möglich, Einträge zu erstellen, zu löschen und zu erhalten.

Unser Ziel war es eine Single-Page-Application (SAP) zu erstellen, deshalb verwendeten wir Angular und schachtelten die Komponenten. Das erlenen von Angular hat einige Zeit und Tutorials benötigt, bis wir die Struktur und den Aufbau verstanden haben, und wussten wie ein korrektes Angular-Projekt aufgebaut wird. Da wir einen SAP Aufbau nutzen, war es uns somit möglich, sehr einfach einen Globalen Header zu setzen. Die dynamischen Inhalte wurden von Angular, entsprechend dem von uns vorgegebenen Routing reingeladen. Somit startet man als normaler Nutzer auf shortly.at, wo dann das UserpageComponent geladen und angezeigt wird. Fügt man etwas beliebiges der URL hinzu wie z.B. shortly.at/gl so wird der RedirectComponent geladen, welche das weiterleiten des Nutzers übernimmt. Die einzigen ausnahmen der Endungen sind „admin“ und „shortly“. Shortly läd die auch die UserpageComponent, und admin läd eine Komponente, welche über mehr API Kommunikationsfunktionen verfügt als die UserComponentPage. Da wir für diese Website keine Nutzer implementierten, ist dies zwar ein Sicherheitsrisiko, war uns aber für das Ausmaß dieses Kurses eine solide Lösung.

Die Website durchlief mehrere Styling, ursprünglich war es nur reines HTML, bis alle Funktionen implementiert waren, wir wollten aber auch CSS erlenen, somit haben wir die zwei Frameworks Bootstrap und Materialize verwendet. Die Implementierung dieser zwei Framework war mit dem vorgegebenen Installationsguide sehr schnell und einfach. Jedoch viel uns bei der Implementation auf, dass sich diese 2 teilweise überschneiden, und wir dann erst herausfinden mussten welches Styling dann von der Website verwendet wird. CSS war nicht sehr schwer, aber viel Herumspielerei. Wir wussten zwar was z.B. Padding und Border sind, jedoch ist es als Anfänger schwer zu verstehen wie man ein Flexitem stylen muss damit es z.B. dynamisch 80 des Bildschirms einnimmt, außer man befindet sich auf dem Handy, dann sollen es 100% Breite sein. Alles in allem war es viel probieren mit einem sehr steilen Lernkurve.

Zuletzt war es uns wichtig, die bisher mit dem Live-Server Plugin nur lokal erreichbare Website für alle frei zugänglich erreichbar zu machen. Hierfür war zu aller erst eine Domain notwendig. Diese haben wir nach schnellem vergleich mehrerer Anbieter bei OVH gekauft. Ursprünglich haben wir die Domain auf einen DDNS Service directed, welche die Dynamische IP Adresse von Stefan verfolgte und erreichbar machte. Später fanden wir heraus, dass OVH selbst diesen Service auch anbietet, und verzichteten auf die zwischenstelle, welche zusätzliche Ladezeiten verursachte.

Der nächste Schritt war es, einen Server für die Website zu erstellen. Ein Nodemon Server so wie wir ihn für die API verwendet haben, schien hier mit Angular nicht zu funktionieren. Wir testeten IIS von Windows, schafften es aber nicht, die Website zu erreichen… Vielleicht war es ein Deploy Fehler? Oder ein Konfigurationsfehler? Vielleicht auch einfach Angular? Wir haben es nicht rausgefunden… .

Dann testeten wir Firebase, dies sollte der Ideale Serverhosting Anbieter für Angular Projekte sein, da dieser Service von Google angeboten wird, welche auch die Entwickler von Angular sind. Prinzipiell wirkte der Service sehr gut und einfach, aber die gratis Version hatte für uns nicht funktioniert, und die vielversprechende bezahlte Version wollten wir nicht bezahlen, da wir ja einen „gratis“ Server haben. Final haben wir Browsersync verwendet. Diese Version hat zwar nicht die schnellsten Ladenzeiten, war aber für unser Projekt ideal. Es war zum einen sehr schnell erreichbar und installiert, und zum anderen konnte man mehrere Browserfenster, auch welche am Handy, synchronisieren, und so konnte man scrollen und Textfelder beschreiben und ähnliches, und sah es auf mehreren Bildschirmgrößen gleichzeitig. Dies war für das laufende CSS Styling sehr vorteilhaft.

Somit kann man abschließend sagen, dass wir in diesem Projekt sehr viel gelernt haben. Wir starteten mit keinen Kenntnissen im Bereich Web Development, und haben nun schlussendlich die Fähigkeiten erworben, eine REST API zu erstellen, eine Website mittels Angular aufzubauen, mit einer API zu kommunizieren, sowie zwischen Pages zu navigieren. Weiters erlernten wir, wie man eine Website hostet und gestaltet, sowie den Erwerb von Domains und deren Konfiguration zur gewünschten Zieladresse. Die freie Auswahl des Projekts hat uns in diesem Kurs sehr gefallen, da man endlich mal Zeit hat ein kleines Hobbyprojekt während der Uni zu realisieren. So eine Website zu erstellen stand sowieso auf unserer Sommerferien-Projekt-Liste, welche wir nun im Rahmen dieses Kurses umsetzten, konnten. Die durchgeführte Dokumentation und Planung dieses Projekt, wurde aber viel Detailliertes durchgeführt, als wenn wir es selbst privat gemacht hätten. Diese Detailliertheit hat sich aber auf unseren Fortschritt sehr positiv ausgewirkt, und wird vermutlich auch bei privaten Projekten in ähnlicher Form Anwendung finden.