|  |
| --- |
|  |

Duale Hochschule Baden-Württemberg Mosbach

Portfolio Webprogrammierung

|  |  |
| --- | --- |
| **Gruppenteilnehmer** | Damien Arriens (Matrikel-Nr: 4364633)  Daniel Boger (Matrikel-Nr: 6015969)  Fabian Qarqur (Matrikel-Nr: 3655976)  Julian Stipovic (Matrikel-Nr: 8994343) |
| **Kurs** | WI20B |
| **Dozent** | Dr. Michael Hartwig |
|  |  |

Inhaltsverzeichnis

[1 Vision des Projekts 1](#_Toc87200382)

[2 Implementierung 2](#_Toc87200383)

[2.1 Architekturdiagramm 2](#_Toc87200384)

[2.2 Use-Cases- Liste 3](#_Toc87200385)

[2.3 Use-Cases Diagramm 4](#_Toc87200386)

[2.4 Nicht-funktionale Anforderungen 5](#_Toc87200387)

[2.5 API-Design 7](#_Toc87200388)

[2.6 Besonderheiten bei der Implementierung 8](#_Toc87200389)

[3 Vorgehen 9](#_Toc87200390)

[3.1 Zeitlicher Ablauf 9](#_Toc87200391)

[3.2 Projektmanagement 9](#_Toc87200392)

[3.3 Aufgabenverteilung 9](#_Toc87200393)

[3.4 Schwierigkeiten bei der Implementierung 10](#_Toc87200394)

[3.4.1 Back-End 10](#_Toc87200395)

[3.4.2 Front-End 13](#_Toc87200396)

[3.5 Literatur/Quellen die verwendet wurden 14](#_Toc87200397)

[4 Wissenschaftliche Reflexion 15](#_Toc87200398)

[4.1 Damien Arriens 15](#_Toc87200399)

[4.2 Daniel Boger 16](#_Toc87200400)

[4.3 Fabian Qarqur 18](#_Toc87200401)

[4.4 Julian Stipovic 20](#_Toc87200402)

# Vision des Projekts

Ein allzeit bekanntes Problem: man möchte nur kurz in den Supermarkt, um eine Kleinigkeit zu kaufen und schon wieder sind die Artikel in anderen Regalen aufgestellt. Nach vergeblichem Suchen fragt man den Mitarbeiter und dieser antwortet, dass der Artikel bereits ausverkauft ist. Verärgert muss man in Google Maps nach der nächstliegenden Filiale suchen. Das Abenteuer beginnt erneut.

Product Navigator behebt das Problem. Die Software zeigt dem Benutzer ausgehend von seinem aktuellen Standort die nächstliegende Filiale, in der das Produkt zur Verfügung steht. Der Benutzer kann die Filiale auswählen und erhält den Standort auf Google Maps angezeigt. Bei der Auswahl des Produkts, welches benötigt wird, erhält der Nutzer in der Software den Standort. Hierbei markiert Product Navigator auf einem Filialplan das entsprechende Regal in welchem sich der Artikel befindet.

Außerdem stellt Product Navigator dem Nutzer eine Auswahl an Alternativen, falls das Produkt nicht in der entsprechenden Filiale zur Verfügung steht. Die Software zeigt gleiche Produkt in einer anderen Marke oder eine andere Filiale, in der das Produkt zur Verfügung steht. Außerdem werden die verschiedenen Angebote der entsprechenden Filiale angezeigt. Ein weiteres Feature ist die Integration von Alexa. Der Nutzer kann Alexa fragen, in welcher Filiale das Produkt zu Verfügung steht und erhält eine Antwort.

Der Nutzer kann somit bereits von zu Hause alle gewünschten Produkte in der Software markieren und sich somit das Suchen in der Filiale sparen. Product Navigator spart Zeit beim Einkaufen und lässt keinen Raum für Verzweiflung, wenn die Artikel wieder auf andere Regale verteilt werden.

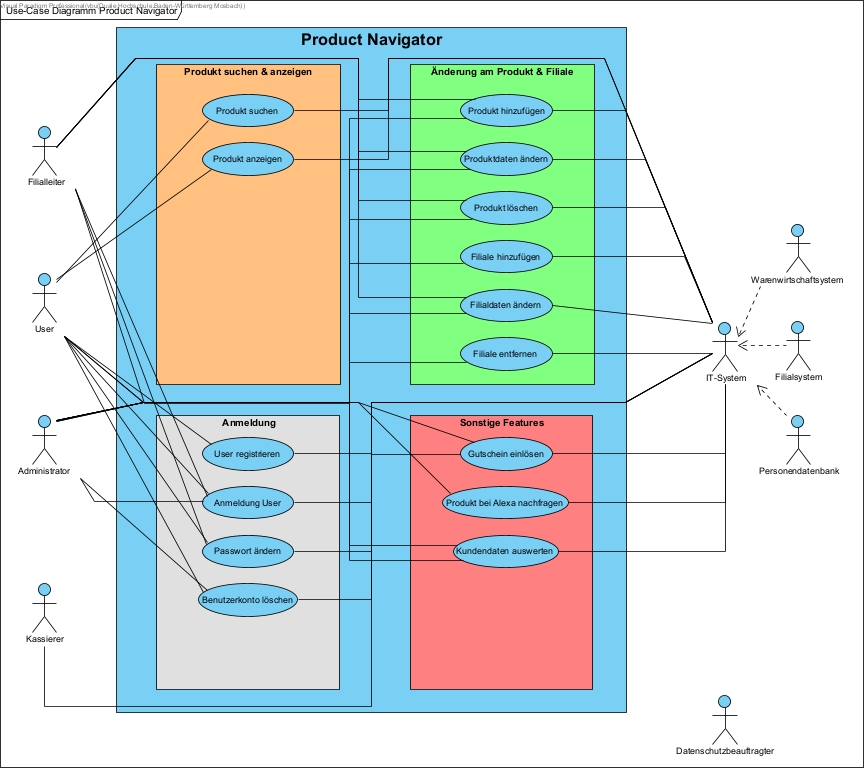
# Implementierung

## Architekturdiagramm

## Use-Cases- Liste

|  |  |  |
| --- | --- | --- |
| Use-Cases | Kurzbeschreibung | Umsetzung |
| Produkt suchen | Der Kunde sucht nach einem Produkt mithilfe der Suchleiste | umgesetzt |
| Produkt anzeigen | Der Kunde wählt ein Produkt aus und bekommt alle Daten vom Produkt angezeigt | umgesetzt |
| Produktdaten ändern | Es werden Änderungen an einem Produkt vorgenommen | umgesetzt |
| Produkt hinzufügen | Ein neues Produkt wird hinzugefügt | umgesetzt |
| Produkt löschen | Ein Produkt wird entfernt | umgesetzt |
| User registrieren | Der Kunde registriert sich bei Product Navigator | umgesetzt |
| Anmeldung User | Der Kunde meldet sich bei Product Navigator an | umgesetzt |
| Filialdaten ändern | Bei einer bestehenden Filiale wird eine Änderung vorgenommen |  |
| Filiale hinzufügen | Eine neu eröffnete Filiale wird hinzugefügt |  |
| Filiale entfernen | Eine nicht geöffnete Filiale entfernen |  |
| Gutschein einlösen | Ein Gutschein wird per Barcode-Scanner eingelöst | nicht umgesetzt |
| Produkt bei Alexa nachfragen | Ein Produkt wird zur Verfügbarkeit in der Filiale nachgefragt | nicht umgesetzt |
| Kundendaten auswerten | Es werden Kundendaten ausgewertet zur Optimierung der Filialstruktur | nicht umgesetzt |
| Passwort ändern | Benutzer ändert sein Passwort für die Anmeldung |  |
| Benutzerkonto löschen | Ein Benutzerkonto wird vom Benutzer aufgelöst |  |

## Use-Cases Diagramm



## Nicht-funktionale Anforderungen

|  |  |  |
| --- | --- | --- |
| Anforderung | Beschreibung | Messbarkeit |
| Funktionalität | | |
| Interoperabilität | Das System soll auf dem PC sowie dem mobilen Gerät anwendbar sein. |  |
| Benutzbarkeit | | |
| Verständlichkeit & Bedienbarkeit | Der Benutzer soll mit geringem Aufwand das System verstehen und anwenden können. |  |
| Zuverlässigkeit | | |
| Fehlertoleranz | Die Software hat trotz Fehlern ein spezifisches Leistungsniveau |  |
| Wiederherstellbarkeit | Das System kann bei einem Versagen sich neustarten und das Leistungsniveau wiederherstellen. |  |
| Verfügbarkeit | Die Verfügbarkeit des Systems muss bei 94 % liegen. | Wahrscheinlichkeit der Nichtverfügbarkeit |
| Performance | | |
| Zeitverhalten | Das System soll innerhalb von zwei Sekunden den Aufruf verarbeitet haben | Reaktionszeit auf Benutzereingaben |
| Genauigkeit | Die Software soll bei einer Suche das Ergebnis genau anzeigen und bei Ähnlichkeiten diese auch anzeigen. |  |
| Wartbarkeit |  |  |
| Analysierbarkeit | Mängel oder Ursachen von Versagen sollen diagnostiziert werden können |  |
| Modifizierbarkeit | Verbesserungen und Fehlerbeseitigungen können mit geringem Aufwand bearbeitet werden |  |
| Sonstige Anforderungen | | |
| Datenschutz | Die deutschen Datenschutzrichtlinien müssen erfüllt werden | Prüfung durch einen Datenschutzbeauftragten |

## API-Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Methode | URL | Erklärung | Data In | Data Out |
| POST | /api/user/login | Login | LoginData: {Name, E-Mail, Password} | {\_id, name, email, password, city, street, birthday} |
| POST | /api/user/register | Register | RegisterData: {Name, E-Mail, Password} | {\_id, name, email, password, city, street, birthday} |
| POST | /api/user/update | Update User Information | personalUpdate: {Name, E-Mail, Password, Birthday} | Bei Erfolg wird nichts zurückgegeben und bei Misserfolg der Fehler |
| POST | /api/user/update | Update Location Information | locationUpdate: {Name, City, Street} | Bei Erfolg wird nichts zurückgegeben und bei Misserfolg der Fehler |
| POST | /api/products/searchProduct | Produktsuche | String: {Name} | Array mit allen Produkten die gefunden werden |
| POST | /api/products/addProduct | Produkt hinzufügen | productData: {marketID, Name, Price, kcal, fat, carbohydrate, protein, salt, location, stock} | Bei Erfolg wird nichts zurückgegeben und bei Misserfolg der Fehler |
| POST | https://product-navigator-back-end.herokuapp.com/api/products/updateProduct | Produkt updaten | productData: {marketID, Name, Price, kcal, fat, carbohydrate, protein, salt, location, stock} | Bei Erfolg wird nichts zurückgegeben und bei Misserfolg der Fehler |

## Besonderheiten bei der Implementierung

Node.js

MongoDB

# Vorgehen

## Zeitlicher Ablauf

|  |  |  |
| --- | --- | --- |
| Phase | Zeitraum | Aufgaben |
| API | 30.11.2021 – 09.11.2021 |  |
|  |  |  |

## Projektmanagement

Im ersten Meeting wurden die Rahmenbedingungen festgelegt, wie das Projekt durchgeführt werden soll.

Die Kommunikation verläuft über Discord, bei der wir je Meeting die Aufgaben verteilen und die Aufgaben besprochen werden, welche erledigt wurden vom einzelnen Projektteilnehmer. Hierbei wird jedem einzelnen Feedback gegeben. Einmal wöchentlich wird ein Meeting vereinbart, damit jeder auf dem aktuellen Stand ist.

Dokumente/Dateien werden über GitHub gespeichert und jedem zur Verfügung gestellt. Dokumente werden vor einem Termin in GitHub gespeichert, damit andere Teilnehmer sich bereits Notizen machen können.

## Aufgabenverteilung

|  |  |  |
| --- | --- | --- |
| Aufgabe | Zuständigkeit | Deadline |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Schwierigkeiten bei der Implementierung

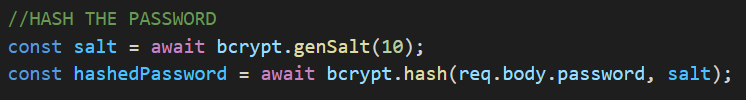
### Back-End

Vorab möchte ich anmerken, dass es allgemeine Schwierigkeiten gab sich wieder in das Programmieren einzuarbeiten. Die Funktionen mit GitHub und Visual Studio Code mussten sich erst angeeignet werden. Die Wahl der Programmiersprache für das Backend, war ebenfalls nicht einfach. Das Gewöhnen an JavaScript, im genaueren Node.js, hat das Ganze nicht wirklich vereinfacht. Dort mussten ebenfalls die grundlegenden Dinge gelernt werden über etliche Internetseiten und YouTube-Videos. Natürlich mussten am Anfang auch viele Pakete installiert werden, mit denen man besser und einfacher arbeiten kann. Darunter zählen z.B. express, mongoose usw. Wenn man diese Hürden überwunden hatte, kam es langsam zu den richtigen Problemen.

Die ersten grundlegenden Schwierigkeiten, die aufgetreten sind, gab es bei der Implementierung der Datenbank. Wir waren uns nicht sicher, welche Datenbank wir benutzen sollten. Außerdem wussten wir nicht, wie man eine Datenbank einrichtet oder mit dieser kommuniziert. Wir haben uns für die MongoDB entschieden. Dort gibt es eine Gratis-Version, welche bis zu 500 MB an Speicherplatz verfügt. Die Verbindung zur Datenbank konnte relativ einfach mit Hilfe der oben genannten Hilfsmittel hergestellt werden und die ersten Tests, etwas zu speichern, haben ebenfalls funktioniert. Die Schwierigkeiten hierbei lagen, den richtigen Command zu finde, da MongoDB Lokal auf dem PC anders angesprochen wird. Wir natürlich wollten die Datenbank online haben und nicht auf einem lokalen Rechner. Da wir eine Aufgabenteilung gemacht haben war die Kommunikation des Web-Servers und der API erstmal nicht so wichtig. Wir haben uns am Anfang jeder auf seinen Teil fokussiert. Was für mich am Anfang schwerer zu begreifen war, waren „router.post“ oder „router.get“.

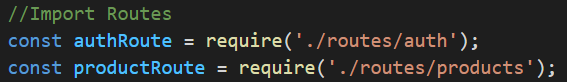
Für mich hat das große und ganze dabei gefehlt. Dadurch, dass wir eben Frontend und backend strikt getrennt programmiert hatten, wusste ich nicht wirklich wie das Frontend funktioniert und dementsprechend auch nicht wie das Frontend auf „‘/register‘“ zugreift. Diese Dinge haben sich dann später mit der Zeit gelegt, da ich mir den Frontend Code genauer angeschaut habe und konnte somit dann auch das backend besser verstehen.

Das Umschreiben auf JSON und das Schicken der Daten, war ebenfalls ein Schwerpunkt. Mit Daten, die im „body“ vom Frontend an das Backend geschickt wurden, konnte ich am Anfang nicht viel anfangen, wobei genau das der Hauptpunkt einer API ist.

Beim Registrieren war der Schwerpunkt, dass Haschen des Passwortes.

Hierfür haben wir das Package „bcrypt“ verwendet. Es war nicht klar, wie wir das Passwort verschlüsseln können oder ob wir das überhaupt brauchen. Wir haben es zur Sicherheit trotzdem implementiert, da es dazu relativ viele Erklärungen und Informationen im Internet gab und dieses Thema somit sehr wichtig gewirkt hat.

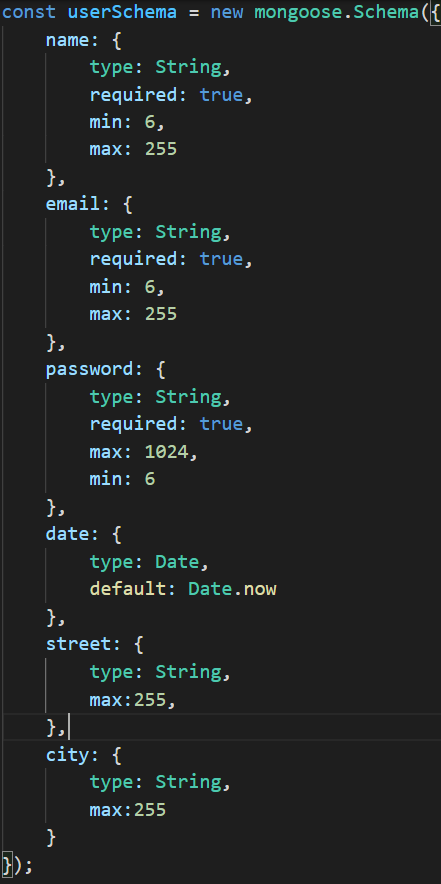
Das Importieren von „routes“ war am Anfang auch neues Wissen, welches man sich erstmal aneignen musste. Bisher haben wir in Java nicht mit „routes“ gearbeitet. Generell haben wir sehr wenig mit unterschiedlichen Dateien, die ineinander greifen gearbeitet. Wichtig für mich bei diesem Thema war, dass man die „routes“ von „get“ und „post“ nicht mit den „routes“, die auf eine andere Datei hinweisen, verwechselt. In der Index Datei wird dann auf die verschiedenen „routes“ verwiesen (siehe Abbildung).



Die Verwendung von Middlewares war ebenfalls sehr schwierig, da wir nicht wussten, ob wir für so eine kleine Webseite „middlewares“ benötigen. Für die Strukturierung und Organisation sind diese nützlich, jedoch bringen diese bei größeren und aufwendigeren Seiten wesentlich mehr. Wir haben sie dann doch implementiert, weil diese bei großen Webseiten ebenfalls benutzt werden.

Die Schemas waren schnell verständlich. Dort war es jedoch entscheidend, welche Einschränkungen und Anforderungen man an die jeweiligen „propertys“ des Schemas hat. Wir mussten uns abstimmen welchen „type“ man verwendet und wie viele Buchstaben z.B. ein String haben darf oder haben muss. Außerdem war wichtig, ob diese property „required“ ist oder nicht. Bei den Schemas war ebenfalls wichtig dann herauszufinden, wie man alle Dateien eines Schemas abfragen kann. Wenn man Produkte auf der Webseite sucht, muss die Liste der Produkte dem Frontend geschickt werden, welches dann das richtige Produkt heraussucht.

Wie unser Schema User aussieht, sehen Sie in der folgenden Abbildung.



Die property „date“ gibt den Tag an, an welchem der Account erstellt wurde. Da diese property auf Default: Date.now gesetzt ist.

Es ist wichtig das Schema richtig zu codieren, damit man keine fehlerhaften Dateien bekommen kann. Mit dem Schema setzt man die Regeln der erstellbaren Datei fest.

Darüber hinaus gab es mehrere kleine Probleme. Da JavaScript viele Klammern benötigt und tiefe Verschachtlungen möglich sind ist es vorgekommen, dass man wegen einer fehlenden Klammer länger suchen musste. Manche Fehlermeldungen musste man Googlen, weil man diese nicht verstanden hatte. Jedoch konnte man sich immer mit guter Recherche weiterhelfen.

### Front-End

Das Front-End stand bereits zum größten Teil, bevor das Back-End angegangen wurde. Das Grundgerüst aus vielen <div> Blöcken hat uns dabei geholfen, eine bessere Vorstellung unseres Projekts zu bekommen. Jedoch gab es auch bei der Implementierung des Front-Ends Probleme.

Aufgrund der Aufgabenverteilung war es nötig, einzelne Meetings zu vereinbaren, um das Coding überhaupt zu verstehen und somit den aktuellen Stand korrekt zu vermitteln.

Eines unser ersten „Probleme“ war einheitliches Design. Jeder hatte leicht unterschiedliche Ideen, wenn es um Farben, Layout und allgemein Design ging. Bereits kleine Unterschiede im Blauton in verschiedenen Bereichen fielen ins Auge. Eine einfache Lösung dazu war das Anlegen einer Farbpalette, mithilfe des var() statements. Somit konnten wir mehrere Farben in Variablen speichern, diese benennen und verwenden. Das erleichtert vor allem die Entscheidung. Jedes Mitglied sieht somit bei Eingeben des var() statements, welche Farben zur Verfügung stehen.

Die Variablen trugen ebenfalls dazu bei, ein einheitliches Bild zwischen Login- bzw. Homepage zu generieren.

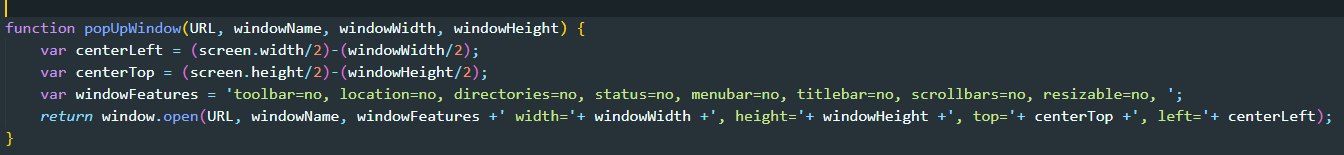
Die Implementierung des Hintergrunds war zunächst auch ein Problem, da das Bild immer verzogen aussah. Zuerst wurde mit verschiedenen Auflösungen experimentiert, was jedoch nicht geholfen hat. Nach kurzer Recherche im Internet, entdeckten wir das Schlüsselwort background-size. Mithilfe dieses Statements konnten wir die Größe des Bilds auf die gesamte Seite strecken (mithilfe von „cover“) und somit ein einheitliches, passendes Bild erzeugen.



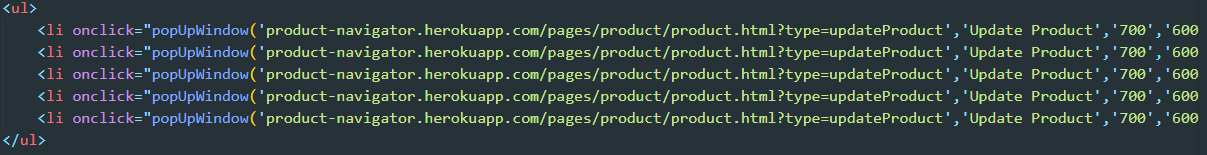
Die Implementierung von Google Maps stellte uns vor eine Herausforderung. Das Einbinden an sich war kein Problem, da Google einen hilfreichen Leitfaden für die Einbindung von Google Maps erstellt haben. Der JavaScript Code wurde von Google erläutert und konnte somit ohne große Probleme an unsere Website angepasst werden. Das eigentliche Problem war die schlussendliche Funktion von Google Maps. Ständig bekamen wir den Fehler „Google Maps kann auf dieser Website nicht richtig geladen werden“. Uns war zunächst nicht klar, warum es nicht ging, da wir uns sogar um einen API Key gekümmert hatten, ohne den die Einbindung gar nicht erst möglich gewesen wäre. Nach kurzer Recherche wurde uns klar, dass das Projekt auf der Google Cloud mit einem Rechnungskonto verknüpft werden muss. Google bietet glücklicherweise eine kostenlose Testversion an, mit der das Verknüpfen des Rechnungskontos ohne Kosten abbildbar ist.

Bei der Implementierung der Produktliste wollte wir zunächst einfache <div> Blöcke nutzen. Schlussendlich haben wir uns jedoch für eine unordered list (<ul>) entschieden. Im Falle eines Ausbaus der Website, ist das Hinzufügen weiterer Seiten und weiterer Einträge über <li> deutlich praktischer und einfacher. Somit konnten wir auch das Design der Liste einfach mit CSS anpassen, indem wir das <li> Element ansprechen.

Um die Produktanzeige übersichtlicher zu gestalten, haben wir uns anstelle einer separaten Seite für ein Popup entschieden. Somit würde jedes Produkt sein eigenes, kleines und übersichtliches Fenster mit allen relevanten Informationen bekommen. Die Implementierung ließ sich einfach über eine JavaScript Funktion abbilden, die mithilfe von „onclick“ ausgelöst wird. Die Funktion öffnet eine neue URL und passt automatisch die Größe an.

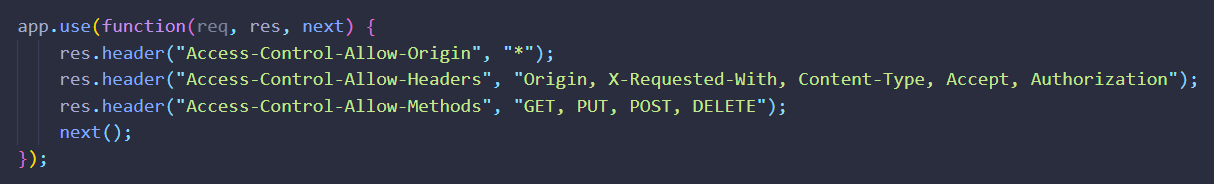


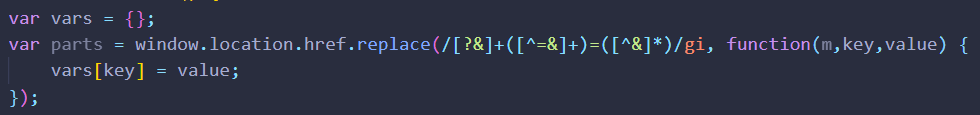
Auf der HTML Seite wird einfach die gewünschte URL und die gewünschte Größe als Argument eingetragen.



### Verbindung zwischen Frontend und Backend

Anfangs war es für uns unklar wie die Verbindungen aussehen werden zwischen dem Frontend und dem Backend, da wir uns damit erst später beschäftigt haben als die Grundgerüste von Frontend und Backend schon standen. Nach langem Probieren haben wir uns für die von JavaScript standardisierte Art von Requests entschieden, dem „Fetch“ Befehl. Diesen haben wir zum Posten von JSON Objekten zum Backend verwendet, um die Kommunikation zu starten. Jedoch hatten wir in der Anwendung dabei viele Probleme bezüglich Typen von Requests, somit mussten wir im Header den Content-Type auf „application/json“ setzten. Damit konnte das Backend die Daten dann auch als JSON Objekt lesen. Die anfängliche Kommunikation hat damit auch schon funktioniert. Jedoch konnten wir die Daten, die das Backend dann an das Frontend zurücksendet, nicht lesen, also das Response konnte nicht als JSON dargestellt werden. Zwar reichte es Langezeit aus, dass man mit dem Befehl „response.ok“ abprüfen konnte ob der POST-Request funktioniert hat oder nicht, jedoch mussten wir später auch Daten von dem Backend lesen, insbesondere Daten, die aus der Datenbank kommen. Nach langem Suchen nach einer Lösung sind wir auf die Lösung gestoßen, dass wir das JSON Objekt, also das Response erst mit „response.json()“ returnen müssen. Dies wandelt das Response Objekt in ein JSON Objekt um, da es vorher noch kein JSON Objekt war. Danach konnte man mit diesem, welches wir „data“ genannt haben die Daten abrufen und auf diese zugreifen. Dies beschleunigte das weitere Arbeiten um einiges. Im Folgenden wird so ein Fetch-Befehl dargestellt.

Im weiteren Verlauf sind wir immer wieder auf das Problem des Cors-Headers gestoßen, indem das Response sicherstellen möchte, dass es auch nur an autorisierte Quellen zurücksendet und nicht den Zugriff von anderen Quellen erlaubt. Dieses Problem ist leider nicht einfach auszuschalten und nach stundenlangem Googlen sind wir auf eine Lösung gefunden, die in unserem Backend Server von Anfang an alle Quellen erlaubt, da es für unser Projekt nicht entscheidend war von woher diese Anfragen kommen, da unser Projekt keinem Realitätsprojekt entspricht. Somit wird im Server eine Middleware gestzt, die mit dem Header „Access-Control-Allow-Origin: „\*““, alle Origins erlaubt, also Anfragen von überall. Dies vereinfachte unsere Kommunikation und es musste nicht immer wieder überprüft werden, ob man alle Quellen mit https hat und konnte auch mit Fremdprogrammen wie zum Beispiel „Postman“, einem http-Request Programm unsere Programme testen.

Des Weiteren wussten wir nicht wie wir die Daten, die uns beim Login von der Datenbank gesendet wurden auf der Home-Seite darstellen lassen, da wir keine Lösung gefunden haben, diese zu übergeben. Dabei haben wir es mit Aufrufen und Importieren von Funktionen aus der jeweiligen anderen Klasse versucht und nichts hat funktioniert, bis wir schließlich auf die Lösung gestoßen sind, dass wir hinter den Link, den wir aufrufen, um die neue Seite aufzurufen mit einem „?“ die Variablen deklarieren können. Zum Beispiel übergibt der Link „…?name=Klaus“ die Variable „name“ mit dem Wert „Klaus“ an die Seite weiter. Diese URL haben wir danach decoded und die einzelnen Variablen in einem Array abgespeichert, wie man im Folgenden sehen kann.

Damit greifen wir auf Benutzerdaten zu, die wir auf der Login-Seite vom Backend gesendet bekommen und zeigen sie auf der Benutzeransicht. Jedoch ist hierbei das Problem, dass auch das Passwort erkenntlich in der URL steckt und wir nicht wussten, wie wir dies übergeben. Dafür haben wir bis jetzt noch keine Lösung gefunden und haben es erstmal dabei belassen, da es für diese Zwecke ausreichend ist.

### Deployment

Beim Deployment auf Heroku sind für mich persönlich die größten Probleme aufgetreten, da man viele Sachen einstellen und ändern musste. Bevor wir es deployed haben, lief unser System lokal mit einem lokalen Node.js Server und mit dem Live Server Add-On von Visual Studio Code. Zuallererst haben wir das Front-End deployed mit dem PHP Buildpack, welches super lief, danach haben wir das Backend hinzugefügt, jedoch kam es zu lauter Fehlern, da Heroku die „node\_modules“ auch aus GitHub geladen hat, welches zu Import Fehlern führte. Dieses Problem haben wir gelöst, indem wir die „node\_modules“ aus GitHub ausschließen und somit Heroku selbst die Module herunterladen lassen aus der Datei „package. json“, wo alle Dependecies aufgelistet sind. Nachdem auch das Backend mit dem Node.js Buildpack deployed war, funktionierte das Front-End nicht mehr. Nach langem suchen nach dem Fehler haben wir herausgefunden, dass das PHP Buildpack automatisch ein Dyno für uns erstellt, hat um das Frontend zu deployen. Jedoch ist dieses verschwunden, als wir das Backend deployed haben, da wir dort eine „Procfile“ definiert haben, die unseren Server mit dem Command „web: node start“ startet. Procfile ist dabei eine Datei, die von Heroku erkannt wird und automatisch als Dyno ausgelöst wird. Bis uns dies aufgefallen ist, haben wir weitere Sachen probiert, wie das Frontend mit Node.js zu starten, jedoch ohne Erfolg. Da man mit dem kostenlosen Account von Heroku nur ein Dyno pro Applikation haben darf konnten wir auch nicht den Befehl von PHP manuell einfügen. Somit sind wir schließlich zu dem Entschluss gekommen, dass wir zwei Heroku Applikationen brauchen, eine für das Frontend und eine für das Backend. Somit brauchten wir auch zwei GitHub Repositories, die jeweils getrennt Frontend-Code und Backend-Code enthalten. Nach ein paar Versuchen und weiteren Cors-Problemen, wie in 3.4.3 erläutert, hat es dann schließlich funktioniert und wir haben zwei Heroku Applikationen, die miteinander kommunizieren.

## Literatur/Quellen die verwendet wurden

# Wissenschaftliche Reflexion

## Damien Arriens

## Daniel Boger

Das Projekt war für mich eine spannende Aufgabe, da man das Erlernte vom letzten und jetzigen Semester tatsächlich anwenden musste. Außerdem bekam man eine ungefähre Vorstellung wie es im Berufsleben ungefähr aussehen könnte, wenn man in einem Projekt zugeteilt ist. Man musste sich, während dem Projekt gemeinsam im Team organisieren, Themen recherchieren, selbständig arbeiten, wodurch man sich selbst verbessern und Erfahrung sammeln konnte.

Am Anfang der Projektphase war mir erst vieles unklar, wie genau wir die Aufgabe bewältigen sollten. Beim ersten Treffen mit der Gruppe konnten aber einige Unklarheiten geklärt werden. Ich finde, dass wir bereits von Anfang an in der Gruppe uns gut koordiniert haben und jeder motiviert war, dass Projekt erfolgreich zu erledigen.

Während der Projektphase habe ich mich hauptsächlich um die Anforderungen und die Modellierung von der Webseite gekümmert, da ich ungefähr die Vorstellungen hatte welche Anforderungen der Kunde haben könnte, dass er den Product Navigator benutzt. Grad die Modellierung, beispielsweise vom Use-Case Diagramm oder dem Architekturdiagramm fiel mir leicht.

Bei der Programmierung von Product Navigator war meine Schwäche das Backend. Ich habe dort nicht ganz verstanden, wie ich zum Beispiel die Datenbank mit dem System verbinde, dass die Daten von der Datenbank abgerufen oder Daten geändert werden können. Lediglich habe ich verstanden, dass man die Methoden „GET“ und „POST“ Methoden benötigt, damit man etwas abruft bzw. ändert, aber die genaue Einsetzung war mir unklar.

Im Bereich vom Frontend hatte ich mehr Verständnis. Hierbei wusste ich wie das HTML aufgebaut werden muss sowie ich mit CSS die Webseite designen kann. Nur das Thema JavaScript war ich mir noch etwas unsicher, auch dahingehend geschuldet, dass die Vorlesung spät vorgetragen wurde, in der das Thema detailliert behandelt wurde. Außerdem ist die Informationsrecherche im Frontend mir leichter gefallen als im Backend.

Die Gruppenarbeit war meiner Meinung nach sehr gut, da jeder was gemacht hat und mir nicht das Gefühl rüber kam das jemand gar nichts macht. Außerdem war die Aufgabenverteilung gerecht, nur die Verteilung zwischen der Modellierung und der Programmierung könnte mehr gemischt sein, dass auch jemand eine Aufgabe bekommt, wo man eine Schwäche dafür hat, aber so etwas Neues lernt.

Die Kommunikation war in Ordnung in der Gruppe, da wir uns mindestens einmal in der Woche getroffen haben und Probleme auch schnell geklärt werden konnten, aber ich mir mehr Treffen im Nachhinein gewünscht hätte.

Für die Studenten nächstes Jahr empfehle ich, dass sie sich in der Gruppe oft zusammensitzen und gegenseitig Feedback geben über ihre Aufgaben. Auch die Planung und Modellierung sollte frühzeitig bestehen, damit mit der Programmierung der Software frühzeitig angefangen werden kann. Außerdem ist eine klare Aufgabenteilung sinnvoll, da jeder so weiß was er tun sollte.

## Fabian Qarqur

Ich würde sagen das Portfolio und Programmieraufgabe war bis jetzt einer der schwersten Projekte, die ich hatte. Zum einen wurde man mehr oder weniger ins kalte Wasser geworfen und hat sich vieles selbst beibringen müssen. Das heißt allerdings nicht, dass dies unbedingt eine schlechte Sache ist. Man hat sehr viel gelernt. Beispielsweise wurde das eigenverantwortliche Arbeiten sehr gefördert, die intensive und genaue Recherche über gewisse Themen oder Gruppenkommunikation und Gruppenlenkung.

Das ganze Projekt musste man erstmal verstehen, bevor man sich einen Plan machen konnte. Es musste mindestens einer einen vollen Durchblick haben, damit dieser das Team steuern kann. Ebenfalls war sehr wichtig, dass die Aufgabenverteilung gestimmt hat. Diese war essenziell wichtig, um so viele parallele Projekte zu bewältigen. Ich würde sagen wir haben sehr strukturiert und organisiert gearbeitet zu mindestens dafür, dass das unser erstes großes Projekt in diesem Ausmaß war.

Ich sehe definitiv meine stärken im Backend und der API. Das war meine Aufgabe in diesem Projekt. Am Anfang war ich sehr überfordert, da ich nichts verstanden hatte und nicht wirklich wusste, wie die Schnittstellen bei so einer Webseite aussehen sollen. Dies wurde jedoch mit der Zeit immer besser und man konnte sich immer mehr auf das Projekt einlassen und hat somit auch immer mehr von der Arbeit der anderen verstanden. Meine Stärke ist somit ebenfalls die Datenbank. Diese habe ich aufgesetzt und die Daten eingepflegt. Was mir geholfen hat, ist die Applikation „Postman“ mit dieser kann man wunderbar alle „get“ und „post“ Funktionen ausprobieren und gegebenenfalls abändern.

Meine Schwächen sind dementsprechend das Frontend. Ich verstehe, wie dieses Aufgebaut ist, jedoch habe ich in diesem Thema keine tiefergehenden Kompetenzen. Erstrecht wenn es zum Thema JavaScript kommt. Dort geht es meiner Meinung nach um kompliziertere Dinge geht als bei CSS und HTML. Außerdem würde ich sagen, dass meine Schwächen ebenfalls in Packages liegen, die man nicht sehr oft benutzt hat. Dinge wie „bcrypt“ oder „@hapi/joi“ habe ich zwar grundlegend verstanden, jedoch habe ich diese Funktion nur ein bis zwei Mal im Projekt verwendet und dementsprechend auch nicht weiter hineingearbeitet.

Ich würde im nächsten Projekt festlegen, dass man sich regelmäßig zu den gewissen Uhrzeiten und Tagen trifft. Getroffen haben wir uns relativ selten und auch nicht mit allen Gruppenmitgliedern. Die Kommunikation würde dadurch noch besser werden und man kann somit eventuell gewisse Fehler, die durch mangelhafte Kommunikation aufgetreten sind, vorbeugen. Ich würde ebenfalls mindestens zwei Gruppenmitglieder auf API und Datenbank ansetzten und zuteilen, damit man sich dort besser über Ideen austauschen kann.

Den Studenten nächstes Jahr würde ich Empfehlen, dass sie früh genüg anfangen und sich einen ordentlichen Plan zurechtlegen. Außerdem sollen sie sich nicht zu viele Funktionen für ihre Webseite ausdenken, da diese zum größten Teil nicht umgesetzt werden können.

## Julian Stipovic

Dieses Projekt hat uns alle gleichermaßen eingebunden und uns alle strapaziert, herausgefordert und geprägt. Jeder im Team Product Navigator hat seine eigenen Stärken und Schwächen – und genau darin bestand die Herausforderung. Wer übernimmt welche Aufgabe? Wer schafft es bis wann? Wer ist zuständig für das Front-End? Wer kümmert sich um das Back-End?

Das alles waren Fragen, die wir zunächst klären mussten. In meinen Augen war das eine ausgezeichnete Vorbereitung auf das „wahre Leben“ im Job nach dem Studium. In einem Projektteam ist nicht jeder Programmierer, Software-Engineer oder Projektmanager. Ein Projektteam ist immer bunt gemischt, um das Projekt möglichst nah an die gesetzte Vision zu bringen. Somit werden alle Facetten und Seiten beleuchtet. Ich persönlich bin seit Anbeginn des Studiums kein großer Fan des Programmierens. Zusammen mit meinem Projektkollege haben wir uns zunächst mit der Fallstudie anstatt Webprogrammierung befasst. Doch jeder muss auch mal in den sauren Apfel beißen und somit habe ich mir vorgenommen, wo es auch nur geht, beim Programmieren zu helfen. Einer unser Hauptentwickler hat mich in das Skelett des Front-Ends eingeführt und mir gezeigt, wo die Baustellen sind und wie ich das eine oder andere umsetzen könnte. Dabei wurden mir mehrere Hilfestellungen gegeben und wir hatten immer die Möglichkeit, uns flexibel über Discord zu treffen und mithilfe von Bildschirmübertragung gemeinsam zu coden. Das ist meiner Ansicht nach der Kern eines Projekts – gemeinsam das Ziel erreichen.

Im Laufe des Projekts wurde ich immer vertrauter mit dem Front-End und habe dort unterstützt, wo es nötig, entlastend und hilfreich war. Ich habe das Web-Design als ausbaufähiges Gebiet für mich entdeckt, welches mir wirklich Spaß bereitet hat. Das Aufbauen einer kohärenten Farbpalette und eines sinnvollen Designs hat mich herausgefordert und mich weitergebracht.

Dementsprechend war meine größte Schwäche während des Projekts das Back-End. Da wir NodeJS benutzt haben, fiel es mir schwer, den Code richtig zu verstehen, da ich kaum Kompetenzen und Fachwissen in diesem Bereich habe.

In zukünftigen Projekten ist möglicherweise ein früheres Planen aller Aktivitäten, vielleicht sogar unter Anwendung von agilem Projektmanagement, von Vorteil. Mehrere geregelte Meetings in großer Runde wären sicherlich auch hilfreich. Ebenfalls ist es sinnvoll, so früh wie möglich zu beginnen, um frühstmöglich ein realistisches Bild der Website zu bekommen. Manchmal ist es besser, weniger Funktionen zu haben und diese richtig einzubinden als sich zu viele Funktionen vorzunehmen, welche im Endeffekt nicht umgesetzt werden.

Abschließend lässt sich sagen, dass das Projekt trotz aller Schwierigkeiten, gelungen ist und uns allen einen Mehrwert gebracht hat. Wir haben uns der Herausforderung gestellt und jeder musste über seinen eigenen Schatten springen, um sich möglichst effizient einzubringen.