Blog Post

Jan Schefer / Luk Schrodt

20/06/2023

Modul 347

Inhaltsverzeichnis

Glossar 3
Projektauftrag 4
Einleitung 4
Zielpublikum unseres Projektes 4
Informationsbeschaffung 5
Auftrag 5
Ideen 5
Planung & Entscheidungen 5
Verwendete Technologien 6
Design 6
Kostenschätzung 7
AWS 7
Azure 7
Realisierung 8
Layout und Gestaltung der Webseitenoberfläche 8
Backend und REST API 8
Datenbank 8
Hosting 9
Veränderungen an den Technologien 9
Kontrollen 10
Planungskontroll-Tabelle 10
Testing 11
Funktionstests 11
Fazit 14
Gruppenfazit 14
Fazit Jan Schefer 15
Fazit Luk Schrodt 15

Glossar

Im Glossar werden Fachspezifische Begriffe erklärt.

Begriff	Erläuterung
Begriff	Erläuterung
Layout	Text- und Bildgestaltung einer Seite.
VM	Eine virtuelle Maschine (VM) ist ein Betriebssystem (OS) oder eine Anwendungsumgebung, die auf einer Software installiert ist, die eine dedizierte Hardware imitiert.
Monolith	Eine monolithische Architektur ist ein herkömmliches Modell eines Softwareprogramms, das als komplette Einheit erstellt wurde, die in sich geschlossen und unabhängig von anderen Anwendungen ist.
Microservice	Microservices sind ein Architekturmuster der Informationstechnik, bei dem komplexe Anwendungssoftware aus unabhängigen Prozessen generiert wird.
API	Eine API (Application Programming Interface) ist ein Satz von Befehlen, Funktionen, Protokollen und Objekten, die Programmierer verwenden können, um eine Software zu erstellen oder mit einem externen System zu interagieren.
VPN	Ein VPN (Virtual Private Network) ist ein Dienst, der eine sichere, verschlüsselte Online-Verbindung herstellt
UX	User-Experience-Design oder UX-Design befasst sich mit der Analyse, Kreation und Optimierung der Nutzererfahrung
UI	UI ist die Abkürzung für User Interface (Benutzerschnittstelle). Gemeint ist damit im Webdesign meist das visuelle Erscheinungsbild einer Website oder eine Online-Anwendung.
CI/CD	CI/CD (Continuous Integration/Continuous Delivery) sind bewährte DevOps-Methoden zur Automatisierung in der Anwendungsentwicklung. Die Hauptkonzepte von CI/CD sind Continuous Integration (kontinuierliche Integration), Continuous Delivery und Continuous Deployment (kontinuierliche Verteilung).
Hosting	Dienstleistung, die darin besteht, dem Nutzer bestimmte das Internet betref-fende Leistungen anzubieten

Katalogisierung

Projektauftrag

Projektbezeichnung	Erstellen von Blogposts
Kurzbeschreibung	Es werden vier Container erstellt. Jeweils einer für Datenbank und Frontend und zwei für das Backend.
Ziel	Wir wollen einen Katalog erstellen indem Blogposts erstellt, angepasst und verwaltet werden können.
Auftraggeber	Herr Thanam Pangri
Projektergebnis	Eine Website mit Login, für Blogposts.
Projektmanager	Jan Schefer, Luk Schrodt
Arbeitszuweisung	Dokumentation: Jan Schefer Backend: Luk Schrodt Frontend: Jan Schefer Planung: Jan Schefer/Luk Schrodt

Einleitung

Zielpublikum unseres Projektes

Dieses Projekt soll eine extrem vereinfachte Version von etwa Instagramm, Facebook oder Twitter darstellen. Somit wäre die Zielgruppe eine ähnliche und würde auch vermutlich auch diese Leute erreichen. Es ist logischerweise nicht auf einem Stand bei dem es dies würde, allerdings würde für ein solches Projekt auch die Zeit fehlen.

Informationsbeschaffung

Auftrag

Für das Modulprojekt erhalten Sie im Unterricht 18 Lektionen Zeit für die Planung und Umsetzung. Sollte die Zeit nicht reichen, ist der Rest des Projekts in Ihrer Freizeit zu erledigen. Ziel des Projekts ist es, dass Sie am Ende eine "containerized app" geschaffen haben, die Sie optimalerweise im Alltag oder auch in anderen Schulprojekten weiterverwenden können.

Das Projekt darf lokal entwickelt werden. Es soll aber auch auf dem LernMAAS deployed werden (spätestens für die Abgabe - besser aber schon früher). Für den Zugriff auf LernMAAS benötigen Sie WireGuard und von der Lehrperson die Information, welche VM Ihnen gehört und wer welche IP-Adresse und welchen Key für den VPN-Zugang erhält. Dies wird durch die Lehrperson koordiniert, damit keine IP-Konflikte entstehen.

Ideen

Für die Ideensammlung sassen wir zusammen und überlegten uns einen praxisbezogen Auftrag, welcher in Zukunft weiter verwendet werden kann. Schnell einigten wir uns auf ein simples Frontend welches mit Daten aus dem Backend gefüllt wird. Nun mussten wir uns noch ein Thema finden, welches uns Daten liefert für ein Funktionierendes Programm. Wir entschieden uns für eine Art Blogpost-Webseite zu machen, da es sehr simpel war und offen zur weiteren Verwendung. Desertieren war es uns so möglich einfach Daten zu beschaffen und analysieren.

Planung & Entscheidungen

Im Abschnitt Planung werden folgende Themen behandelt:

- Zeiteinteilung
- Design und Aufbau

Verwendete Technologien

Wir haben Folgende Technologien verwendet, wobei später noch mehr dazu kam welche nicht in der Planung mit einberechnet war.

Anwendungsort	Geplante Technologien
Frontend	React, Type Script, CSS
Backend	Springboot, Gradle, Hibernate
Datenbank	PostgreSQL
Hosting	LernMAAS, Docker, Dockerhub, Kubernetes

In unserem Gantt-Diagramm wird aufgezeigt, bis wann ein Teil-Projekt fertiggestellt werden muss. Die Legende zeigt zudem wer für welchen Teil zuständig ist.

ID	Aufgabenname	Anfang	Abschluss	Dauer
1	Projekt defineiren	06.06.2023	06.06.2023	,2w
2	Frontend aufsetzen	06.06.2023	20.06.2023	2,2w
3	Backend aufsetzen	13.06.2023	04.07.2023	3,2w
4	Datenbank füllen	27.06.2023	27.06.2023	,2w
5	Dokumentation schrieben	13.06.2023	07.07.2023	3,8w
6	Testing	20.06.2023	04.07.2023	2,2w

Legende	Farbcode
Luk Schrodt	
Jan Schefer	
Jan Schefer/Luk Schrodt	

Design

Bei dieser Applikation haben wir uns für ein sehr einfaches UX-Design entschieden. Grund hierfür war das wir bereits bei Beginn dieses Projektes wussten das es eher ein knappes vorhaben wird eine fullstack- Applikation in dieser Zeit zu machen, welche auch noch gut aussieht. Wir wollten uns hauptsächlich auf die Funktionen konzentrieren.

Der Aufbau unserer Applikation war für uns auch recht schnell klar. Wir haben beide bereits an mehreren Applikationen gearbeitet, wobei wir im Front- und Backend gearbeitet haben. Wir haben dabei die besten Erfahrungen gemacht mit einem React Frontend und einem Springboot Gradel Backend.

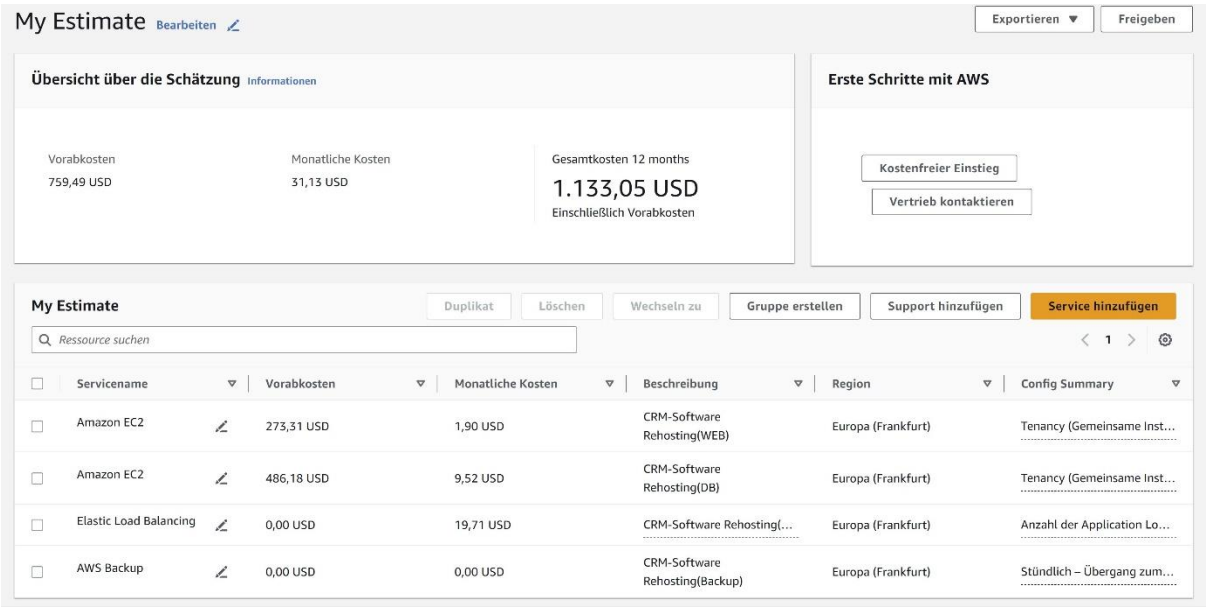
Wir habe ausserdem eine PostgreSQL Datenbank verwendet, welche wir durch Hibernate mit unserem Backend verbunden war. Wir haben so entschieden da wir ein üK dazu gehabt haben und es für uns das einfachste war.

Kostenschätzung

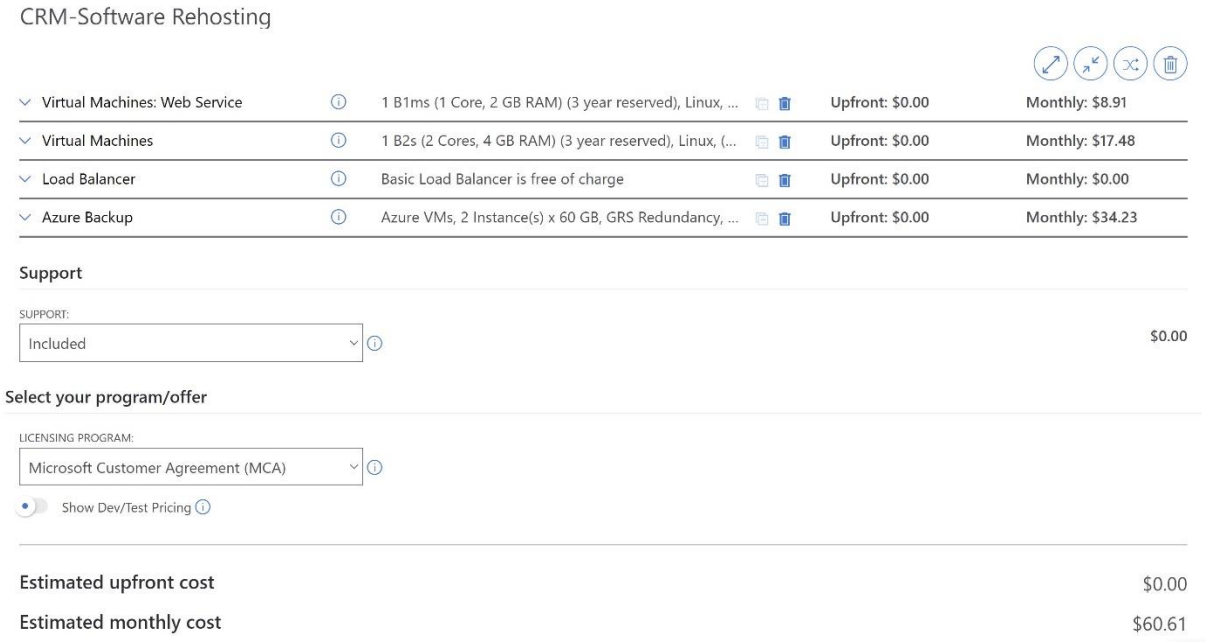
Die Erhaltungskosten für diese Applikation würden vermutlich sehr gering ausfallen, wenn man sie Lokal selbst hosten würde. Hierbei muss allerdings erwähnt werden das die Leistung schlechter wäre als bei einem Cloud Anbieter. Zudem wurden auch keine Mitarbeiter Kosten verrechnet.

Wir haben diverse Cloud Anbieter verglichen und haben folgende Preise erhalten:

AWS



Azure



In unserem Fall würde es sich vermutlich mehr lohnen die Applikation auf Azure zu hosten. Wichtig hierbei ist das wir vom Stand 07.07.2023 ausgehen, und es noch keine Microservices gibt und somit ein Gateway nicht mit einberechnet wurde.

Realisierung

Layout und Gestaltung der Webseitenoberfläche

Der erste Schritt in Richtung Frontend war die Weboberfläche. Wir haben hierbei MUI verwendet, da es ohne weitere umstände ein Simples Design erstellt ohne CSS. So haben wir uns den Teil des Stylings sparen können, da wir wie in der Planung erwähnt eh ein simpleres Layout wollten. Wichtig hierbei war jedoch das wir die UX-Richtlinien einzuhalten, um eine Intuitive Navigation zu erstellen und keine Dark Patterns zu erhalten. Leider kam es auf Luks Laptop zu unerwarteten Problemen mit den Code, wodurch er sämtlichen code nur noch unabhängig vom Projekt testen konnte.

Backend und REST API

Bei der Datenbank haben wir Anfangs alles so aufgebaut wie bei einer Monolith Struktur. Dies haben wir getan um so sicher eine funktionierende Abgabe zu haben und um später diese aufteilen zu können für die Microservices. Leider hat dies mit der Security nicht ganz wie geplant funktioniert, aufgrund des JWT, weshalb wir daraufhin uns entschieden ein Grossteil der Services auf dem ersten Backendcontainer zu lassen und diesen zu sichern. Wir haben dafür ein zweiten Backendcontainer gemacht der nur Zitate generiert wenn man den End Point anspricht. Somit konnten wir dennoch eine Art von Microservices implementieren, ohne die Security zu gefährden.

Datenbank

Für die Datenbank haben wir wie geplant vorgehen können. Wir haben eine PostgreSQL Datenbank erstellt und diese an unser Backend angebunden. Wir haben allerding kein SQL geschrieben sondern es schreiben lassen von Hibernate. Somit konnten wir das aufwendige schreiben der Datenbank eingrenzen und verringern.

Hosting

Swagger: http://10.4.31.18:8080/swagger-ui/index.html

Frontend: http://10.4.31.18:8000

Login frontend

Email: admin@example.com

Password: 1234

Für das Hosting wurde uns eine VM auf LernMAAS zur Verfügung gestellt. Das Initialisieren für das Hosten verlief sehr holprig. Wir haben uns bereits nach dem Erstellen der Monolith-Form dazu entschieden, unsere Applikation zu hosten. Dies schien zu funktionieren und wir konzentrierten uns auf das refactoring auf die Microservice-Architektur. In der Zwischenzeit fiel uns auf, dass unser CI/CD nicht so funktionierte wie wir uns das wünschten und stellten fest, dass das Hosting nie wirklich funktioniert hat. Im Anschluss mussten wir uns vom Kubernetes verabschieden, da es Komplikationen in den config-files gab. Nach diversen erfolglosen Versuchen erzielten wir endlich einen Erfolg. Mittels puTTY konnten wir unsere Container auf der VM installieren und unsere Applikation hosten. Unter den oben aufgeführten links ist das Frontend, sowie unsere Swagger Dokumentation zu finden.

Veränderungen an den Technologien

Es gab während der Realisierung noch ein paar unerwartete Änderungen an den verwendeten Technologien. Die folgende Tabelle enthält die schlussendlich verwendeten Technologien.

Anwendungsort	Geplante Technologien
Frontend	React, Type Script, MUI
Backend	Springboot, Gradle, Hibernate
Datenbank	PostgreSQL, DBeaver
Hosting	LernMAAS, Docker, Dockerhub, PuTTY

Kontrollen

Planungskontroll-Tabelle

In dieser Tabelle wird wurde nachverfolgt, ob die im Planungsabschnitt gesetzten Termine zeitgerecht erfüllt wurden. Bei Abweichungen wurde in der Spalte Kommentar eine Erklärung dazu abgegeben.

Auftrag	Verantwortlich	Frist	Abgeschlossen	Kommentar
Planung	Jan Schefer / Luk Schrodt	06.06.23	06.06.23	-
Provisorisches backend	Luk Schrodt	13.06.23	20.06.23	Provisorisch fertig. Wird laufend leicht angepasst
Provisorisches Frontend	Jan Schefer	13.06.23	13.06.23	Provisorisch fertig. Wird laufend leicht angepasst.
Dokumentation	Jan Schefer	11.07.23	11.07.23	Wurde stets aktuell gehalten und aktualisiert
Login erstellen	Jan Schefer	13.06.23	13.06.23	
DockerFile erstellen	Luk Schrodt	20.06.23	20.06.23	Wurde gleichzeitig mit der Registrierung erstellt.
Monolith fertig	Jan Schefer / Luk Schrodt	20.06.23	27.06.23	Durch Komplikationen (z.B CORS) wurde das Ziel um eine Woche verfehlt.
Microservice Implementieren	Jan Schefer / Luk Schrodt	27.06.23	10.07.23	Wurde aus der Bewertung entfernt. Wir haben es noch eingebaut, da wir kurz Zeit hatten
Kubernetes implementieren	Jan Schefer / Luk Schrodt	11.07.23	-	Wurde aus der Bewertung entfernt. Konnte nicht im vorgegebenen Zeitrahmen erreicht werden

Testing

Auswahl

Wir haben uns beim Testen unserer Applikation für Cypress entschieden. Zudem haben wir durch Exploration-Testing die Funktionalität unserer Applikation sichergestellt und allfällige Fehler aufgedeckt.

Vorgehen

Während der Entwicklungsphase haben wir durch Cypress Tests die Funktionalität des Codes gesichert. Bei allfälligen Änderungen und Erweiterungen des Codes konnten wir durch das Ausführen der Tests so sicherstellen können, dass der bereits implementierte Code nicht von den Anpassungen beeinflusst wurde. Gegen Ende des Projektes haben wir zudem den Entschluss getroffen, die Applikation durch Exploration Tests nochmals zu testen. Der Grundgedanke dahinter war, dass wir mit diesen Tests neue Fehler aufdecken, die von den Cypress-Tests nicht abgedeckt wurden.

Funktionstests

Testfall-Nr	1
Testfall-Bezeichnung	Erstellung von User
Anforderungs-Nr.	1
Testumgebung	Containerized Frontend, Backend und Datenbank mit Cypress
Zu testende Funktionalität	- Einloggen als Admin - Erstellen eines neuen Users - Bearbeiten eines Users - Löschen eines Users
Datum der Testdurchführung	08.07.2023
Tester	Jan Schefer, Luk Schrodt

Testschritte:

Nr.	Aktion	Erwartetes Ergebnis	Effektives Ergebnis	Erfüllt	Kommentar
1	Einloggen mit dem Admin-login	Eingeloggter User mit Admin-rechten	Eingeloggter User mit Admin-rechten		-
2	Auf die User-create Seite wechseln	User Formik mit placeholder	User Formik mit placeholder		-
3	Werte aus dem JSON File in das Formik-formular	User mit den vorgegebenen Werten	User mit den vorgegebenen Werten		-

	einfügen und speichern				
4	Erstellten User bearbeiten	Der neue User wird angeklickt und die Daten bearbeitbar	User wurde nicht gefunden		Name der Referenz auf dem User war falsch gesetzt
5	Erstellten User bearbeiten	Der neue User wird angeklickt und die Daten bearbeitbar	Der neue User wird angeklickt und die Daten bearbeitbar		User Referenz wurde angepasst und gefunden
6	Bearbeiteter User löschen	User ist nicht mehr in der DB vorhanden	User wurde nicht gefunden		Referenz wurde nicht gesetzt
7	Bearbeiteter User löschen	User ist nicht mehr in der DB vorhanden	Der Knopf zum Löschen konnte nicht erreicht werden		Es muss ein Scrollen eingebaut werden
8	Bearbeiteter User löschen	User ist nicht mehr in der DB vorhanden	User ist nicht mehr in der DB vorhanden		-

Testfall-Nr	1
Testfall-Bezeichnung	Erstellung eines Blog-Post
Anforderungs-Nr.	1
Testumgebung	Containerized Frontend, Backend und Datenbank mit Cypress
Zu testende Funktionalität	- Einloggen als Admin - Erstellen eines neuen Posts - Bearbeiten eines Posts - Löschen eines Posts
Datum der Testdurchführung	08.07.2023
Tester	Jan Schefer, Luk Schrodt

Testschritte:

Nr.	Aktion	Erwartetes Ergebnis	Effektives Ergebnis	Erfüllt	Kommentar
1	Einloggen mit dem Admin-login	Eingeloggter User mit Admin-rechten	Eingeloggter User mit Admin-rechten		-
2	Auf die Create-Blog-Post Seite wechseln	Blog-Post Formik mit placeholder	Blog-Post Formik mit placeholder		-
3	Werte aus dem JSON File in das Formik-formular einfügen und speichern	Neuer Blog-Post mit Title, Text erstellt	Neuer Blog-Post mit Title, Text erstellt		-
4	Erstellten Post bearbeiten	Inhalt des Posts ist angepasst	Inhalt des Posts ist angepasst		-
5	Bearbeiteter Post löschen	Post ist nicht mehr in der DB vorhanden	Der Knopf zum Löschen konnte nicht erreicht werden		Es muss ein Scrollen eingebaut werden
6	Bearbeiteter Post löschen	Post ist nicht mehr in der DB vorhanden	Post ist nicht mehr in der DB vorhanden		-

Fazit

Gruppenfazit

Die Arbeit in der Gruppe verlief gut. Es war nicht das erste Projekt, welches wir zusammen bewältigt haben. Wir kannten bereits die Stärken und Schwächen des anderen und wussten, welche Vorlieben der jeweils andere in einer Projektarbeit hat. Durch diese Kenntnisse und die Vorgabe mittels eines Zeitplans erstellten wir ein Gant-Diagramm, um uns Übersicht zu verschaffen. Das Diagramm hat zu Beginn gut funktioniert, jedoch kahmen wir an den Punkt, dass wir unser Programm zu einem Kombinierten und anschliessend versuchten, noch Microservices einzubauen. Es schien nichts mehr zu funktionieren und wir verloren die Hoffnung. Wir waren kurz davor das Projekt neu aufzusetzen und bei 0 zu starten. Durch Hilfe von Mitschüler und Mentoren, fanden wir wieder die Spur und versuchten uns am Schluss sogar an Kubernetes und nochmals Microservice. Kubernetes hat funktioniert und wir hatten wieder Hoffnung. Anschliessend bemerkten wir, dass unser Programm nie wirklich gehostet war und immer die lokalen Container benutzte. Dieses Problem konnten wir zwar lösen, jedoch nur durch das Entfernen von Kubernetes. Dafür konnten wir am Schluss kurz und knapp mit einem Zitat beweisen, dass wir Microservice-Architektur verwendet haben. Auch wenn dies nicht im klassischen Sinne ist ungesetzt wurde. Uns ist bewusst, dass die Funktionen intelligenter verteilt werden sollen und jeder Container z.B. einen Service repräsentieren kann. Aus Zeitgründen, entschieden wir uns jedoch dafür die Basis zu verstehen und zu beweisen, dass die Umsetzung möglich ist.

Fazit Jan Schefer

Ich bin froh, dass das Projekt vorbei ist. Mir hat es zu beginn ziemlich zugesagt und ich war sehr motiviert. Schnell jedoch stellte ich fest, dass der Aufwand viel grösser ist als gedacht. Ich musste oft am Abend für das Modul arbeiten oder am Wochenende einen Tag investieren. Das Projekt ist schlussendlich nur mit der Hilfe von Mitschülern realisierbar geworden. Ich verlor zu beginn viel Zeit mit dem Frontend, sodass ich am Schluss viel Stress mit Microservice hatte. Das Timing der Abgabe war zudem für mich (und andere in der BMS) sehr ungünstig. Mir ist bewusst, dass man den Zeitpunkt der Abgabe nicht gross anpassen kann, dass das Modul nach Plan beendet wird, jedoch entstand bei einigen von uns ein riesen Stress, weil wir fast zeitgleich noch für zwei Abschlussprüfungen lernen mussten und dies sehr viel Zeit konsumierte. Es ist jedoch zu sagen, dass nicht alles schlecht lief. Die Teamarbeit mit Luk funktionierte jedoch gut. Wir hatten viel Kontakt und konnten die Arbeit gut aufteilen.

Fazit Luk Schrodt

Das Projekt war anfangs sehr interessant und spannend. Ich habe mit grosser Motivation daran gearbeitet und mich dafür engagiert ein tolles Produkt zu kreieren. Leider hielt diese Begeisterung nicht lange an, da wir kurz nach dem Start auf die ersten gravierenden Probleme. Ich konnte Beispielsweise das Frontend nicht starten. Dies hat uns einiges an Zeit gekostet und dazu keinen Fortschritt gebracht. Ich war somit teils auch etwas an die Dokumentation gebunden oder musste auf Jans Laptop arbeiten. Insbesondere für das Backend musste ich dies öfter. Auch gab es als der Grundteil der Projektes funktionierte sehr viele Probleme mit neuen Technologien wie Kubernetes, oder den Microservices. Es war schwierig so viel wissen in dieser kurzen Zeit zu sammeln was mich einige Stunden, nicht nur in der Schule, gekostet hat. Ich denke das wir besser eine Fachperson hätten kontaktieren sollen. Aber schlussendlich bin ich froh das ich alles geschafft habe, auch wenn nur mit etwas Stress.