INDIVIDUELLE PRAKTISCHE ARBEIT – Multi-User Applikation

AUTHOR(EN)	Dominic Landolt
VERSION	1.0
STATUS	Final
QUELLE	Atos
DOKUMENTDATUM	08. Dezember 2021
ANZAHL DER SEITEN	60
OWNER	Atos

Inhaltsverzeichnis

Änderungshistorie

Version	Datum	Beschreibung	Autor(en)
0.1	29.11.2021	Erstellung initialer Dokumentenstruktur	Dominic Landolt
0.2	29.11.2021	Überarbeitung und Erweiterung der Dokumentstruktur und Erstellung des Zeitplans	Dominic Landolt
0.3	30.11.2021	Teil 1 und IPERKA Schritt «Informieren» abgeschlossen	Dominic Landolt
0.4	06.12.2021	IPERKA Schritte «Planen» und «Entscheiden» abgeschlossen	Dominic Landolt
1.0	08.12.2021	Abschluss der Arbeit	Dominic Landolt

TELL

1 Einleitung

Diese IPA beschäftigt sich über das Erstellen und Dokumentieren einer Applikation, welche Benutzer und deren Berechtigungen verwalten soll. Zusätzlich sollte diese Applikation relativ einfach von anderen Services und Applikationen zur zentralen Authentifizierung und Autorisierung verwendet werden können.

1.1 Atos

Atos AG ist ein weltweit führender Anbieter für digitale Transformation. Das Unternehmen beschäftigt 107’000 Mitarbeitern und hat einen Jahresumsatz von über 11 Milliarden Euro. Als europäischer Marktführer für Cybersecurity sowie Cloud und High Performance Computing bietet die Atos Gruppe maßgeschneiderte Lösungen für sämtliche Branchen in 71 Ländern an.

Das Ziel von Atos ist es, die Zukunft der Informationstechnologie mitzugestalten. Fachwissen und Services von Atos fördern Wissensentwicklung, Bildung sowie Forschung in einer multikulturellen Welt und tragen zu wissenschaftlicher und technologischer Exzellenz bei.

2 Aufgabenstellung

2.1 Ausgangslage

2.1.1 Vorgeschichte & Gründe für dieses Projekt

Es soll eine Webseite geben, die verschiedene Dinge darstellt und Services anbietet. Einige dieser Services sollen nicht für die allgemeine Öffentlichkeit verfügbar sein. Deshalb müssen die Webseite und diese Services mehrere User mit unterschiedlichen Rechten unterstützen. Um nicht für jeden dieser Services ein einzelnes Benutzer-System schreiben zu müssen, soll es ein zentrales System geben.

2.1.2 Umfeld

Die oben genannten Services existieren zu einem grossen Teil noch nicht. Bereits bestehend ist jedoch die Webseite, auch wenn diese noch keinen nützlichen Inhalt darstellt. Im Hintergrund läuft bereits ein Webserver auf welchen man mit der Domain «landolt.dev» zugreifen kann. Auch wurde über «Let's Encrypt» SSH aufgesetzt.

2.2 Detaillierte Aufgabenstellung

2.2.1 Rahmenbedingungen

- Objektorientiert
- Umfang gemäss Planung (ca. 18h Entwicklung, 15h Dokumentation)
- Zentrale Datenbank
- Mehrere Clients müssen gleichzeitig auf den gleichen Datenbestand zugreifen
- Zentrale Benutzer- und Rechte-Verwaltung

2.2.2 Schnittstellen

Das System ist in sich selbst abgeschlossen. So ist die einzige Schnittstelle, die API-Endpoints, die vom System für die anderen Services zur Verfügung gestellt werden. Die Datenbank, die auch zu diesem System gehört, kann später auch von anderen Services gebraucht werden. Diese werden aber keinen Zugriff auf das Benutzer Dokument/Tabelle haben.

2.2.3 Funktionalität

Es werden verschiedene Endpoints angeboten. Dieser werden für die folgenden Aktionen verwendet:

- SignUp (Neues Konto erstellen)
- LogIn (In Konto einloggen)
- CheckPermission (Überprüfen, ob eingeloggtes Konto Recht hat X zu machen / auf X zuzugreifen)
- Permission-Seite (KANN) (Seite, auf der die Berechtigungen anderer geändert werden können (Benötigt maximale Berechtigung))
- Konto-Seite (KANN) (Seite, auf der die eigenen Konto Informationen dargestellt werden und geändert werden können)

2.2.4 Erwartetes Resultat

Benutzer können Konten erstellen und sich in diese einloggen. Diese Konten haben Rechte. Services, die dieses System verwenden, können überprüfen, ob ein Konto die benötigten Rechte hat, um eine Aktion durchzuführen.

Kann: Diese Rechte können auf einer Permission-Seite von einem Konto mit ausreichenden Rechten vergeben und entzogen werden. Die eigenen Konto Informationen können auf einer Konto-Seite eingesehen und bearbeitet werden.

2.2.5 Tests

Aus Zeitgründen muss kein automatisches Test-System aufgesetzt werden (Unit-Test, etc.)

Es müssen manuelle Testfälle beschrieben sein, die jede der zuvor aufgelisteten Funktionalitäten überprüfen (SignUp, LogIn, CheckPermission, Kann: ChangePermission, ChangeUserInfo).

2.3 Mittel und Methoden

- IDE: Visual Studio Code
- Versionierungssystem: Git und GitHub Repository
- OS: Windows 10/11, Ubuntu Desktop
- Diagramme: diagrams.net (früher draw.io)
- Sprachen: Node.Js (TypeScript und JavaScript) (+ HTML und CSS)
- Datenbank: MongoDB

2.4 Vorkenntnisse

Zuvor Multi-User-Systeme aufgesetzt, jedoch ohne oder nur mit sehr primitiven Berechtigungen. Bekannt mit Visual Studio Code, Git & GitHub, diagrams.net und Node.Js.

2.5 Vorarbeiten

Node.Js Webserver mit SSL aufgesetzt (landolt.dev). Jedoch ohne Inhalt und Datenbank etc.

2.6 Neue Lerninhalte

- Berechtigungs-System planen
- MongoDB als Datenbank

2.7 Arbeiten im letzten Halbjahr

- Node.Js Webserver mit SSL aufsetzten. Arbeit an mehreren C# Projekten.

2.8 IPA Arbeitstage

29.11.2021 - 01.12.2021 & 06.12.2021 – 08.12.2021 (Total: ca. 33h)

3 Projektorganisation

3.1 Lehrbetrieb

Der Kandidat, Dominic Landolt, ist momentan in Ausbildung bei der Atos AG.

3.2 Personen

Person	Rolle	Kontakt
Dominic Landolt	Kandidat	dominic.landolt.external@atos.net
Remo Steinmann	Hauptexperte	remo.steinmann@siemens.net
Merjem Hamza	Nebenexperte	merjem.hamza@siemens.com
Liam Kürner	Nebenexperte	liam.kuerner@siemens.com
Sasa Nikolic	Verantwortliche Fachkraft	sasa.nikolic@atos.net

Tabelle 1 - Personen

3.3 Projektmanagementmethode

In meiner Abteilung arbeite ich mit der Projektmanagementmethode Scrum, welche sich sehr gut für die agile Prozessentwicklung eignet. Bei Scrum werden verschiedene Arbeitsschritte in Zyklen (Sprints) abgehandelt, was das Anpassen von Vorgaben zwischen den Zyklen und die transparente Kommunikation innerhalb des Entwicklerteams ermöglicht.

Scrum ist jedoch für eine IPA nicht gut geeignet, da der Mehraufwand für die Scrum-Artefakte (Daily, Retro, etc.) zu hoch wäre und diese für eine Einzel-Person keinen Sinn ergeben. Auch gibt es bei der IPA, anders als bei den meisten anderen Informatikprojekten, keine Änderungen oder Anpassungen am Auftrag während der Realisierung. So nützt die Agilität von Scrum nicht viel.

Ich habe mich deshalb für die Projektmanagementmethode IPERKA (Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten) entschieden, da ich damit schon viel Erfahrung in der Schule sammeln konnte. Diese klassische Projektmanagementmethode eignet sich für die IPA perfekt, da der Ablauf respektive die Struktur mit den Anforderungen an die Arbeit übereinstimmen.

3.4 Backup-Konzept

Bei der Umsetzung der Arbeit ist es essenziell, dass man stehts auf eine valide Version der Arbeit zurückgreifen kann. Wenn man zum Beispiel einen grossen Fehler macht und man die Applikation nicht wieder zum Laufen bekommt, ist es wichtig, dass man wieder zu einem früheren, funktionierenden Stand zurückkehren kann und man nicht von vorne beginnen muss.

Deshalb wird für den Programmcode und die Dokumentation der Arbeit als Versionierungssystem Git verwendet. Zusätzlich wird dieser lokale Stand bei grösseren Änderungen auf GitHub gepusht, um so über ein «off-site-backup» zu verfügen. So bleiben diese Daten auch erhalten, wenn der Entwicklungslaptop während der Arbeit abstürzen würde.

4 Risikoanalyse

Die Risikoanalysetabelle listet die diversen möglichen Risiken auf, die während der IPA auftreten können. Die Risiken sind nach Eintrittswahrscheinlichkeit und Ausmass bewertet. Die Eintrittswahrscheinlichkeit wurde mit der Skala: Gering, mittel und hoch bewertet. Das Ausmass des Risikos mit der Skala: Gering, mittel und schwer. Dazu werden zu jedem Risiko die entsprechenden Massnahmen zugewiesen und wenn möglich eine vorbeugende Massnahme definiert. Die ganze Analyse dient dazu, dass mögliche Risiken schon früh berücksichtig werden und, dass der Kandidat darauf vorbereitet ist. Wenn ein Risiko dann eintritt, kann so schnell die entsprechende Massnahme aktiviert werden, um einem eventuellen Kontrollverlust entgegenwirken zu können.

Risiko	Eintrittswahrscheinlichkeit	Ausmass	Massnahmen	Vorbeugende Massnahmen
Kandidat wird krank	Mittel	Schwer	Ausbildner, Berufsbildner und Experte informieren. Arztzeugnis besorgen. Planung anpassen.	
Lockdown	Mittel	Gering	IPA wird im Homeoffice durchgeführt.	Homeoffice einrichten.
Zeitmangel	Mittel	Schwer	Kann-Kriterien streichen. Mit dem Kunden Muss-Kriterien streichen.	Wichtige Kriterien priorisieren.

Tabelle 2 - Risikoanalyse

5 Feature Liste

Die Feature Liste listet die umzusetzenden Features und deren detaillierten Beschreibung auf. Dabei werden die Features nach ihrer Priorität, Kriterium und Dauer bewertet. Die Priorität wird nach der Skala: Gering, mittel und hoch bewertet. Das Kriterium beschreibt, ob das Feature ein «Muss» oder «Kann» Kriterium ist. Die Dauer entspricht der geschätzten Zeit zum Umsetzen des Features. Schlussendlich verweist die Referenz auf die Reflexion in welchen das Feature umgesetzt wurde oder gestrichen wurde ([Seitenzahl], [Abschnitt]). Die Liste erlaubt es einem einen schnellen Überblick über die geplanten Features zu gewinnen. Durchgestrichene Features wurden nicht umgesetzt.

Nr.	Feature	Beschreibung	Priorität	Kriterium	Dauer	Referenz
1	SignUp	Neues Konto erstellen	5	Muss	1.5h	(25, Z.Fassung)
2	LogIn	In Konto einloggen	4	Muss	1.5h	(25, Z.Fassung)
3	CheckPermission	Benutzer Rechte überprüfen	3	Muss	1.5h	(25, Z.Fassung)
~~4~~	~~Permission-Seite~~	~~Benutzer Rechte ändern~~	~~2~~	~~Kann~~	~~0.75h~~	(21, Probleme)
~~5~~	~~Konto-Seite~~	~~Benutzer Details anzeigen~~	~~1~~	~~Kann~~	~~0.75h~~	(21, Probleme)

Tabelle 3 - Feature Liste

6 Zeitplanung

In der Zeitplanung werden alle nötigen Aktivitäten in verschiedene Meilensteine aufgeteilt und mit Erfüllungskriterien erweitert. Auch wird ihnen das geplante Fertigstellungsdatum angehängt. Im Gantt-Diagramm kann dann eingelesen werden, wann an was gearbeitet wurde, wann es fertiggestellt wurde und, ob die Zeitplanung eingehalten werden konnte. Bei jeder Aktivität ist die Dokumentation und minimales testen der Aktivität inbegriffen. Jede Spalte im Zeitplan entspricht einer Periode von 2 Stunden. Es wird an jedem Tag 4 oder 6 Stunden gearbeitet. Die Soll Zeit wird mit Grün in den Soll-Reihen markiert. Die Ist-Zeit wird mit Gelb in den Ist-Reihen markiert. Meilensteine werden mit Rot dargestellt. In der ersten Spalte werden die verschiedenen Aktivitäten in acht Gruppen unterteilt. Diese Gruppen sind die sechs Schritte von IPERKA, ein Punkt ist für die Dokumentation und einer für Zusätzliches.

6.1 Meilensteine

Meilensteine sind wichtige Teilpunkte im Projektverlauf. Sie werden als Prüfpunkte verwendet und wirken sich positiv auf die Qualitätssicherung aus. Die erledigten Ergebnisse können so mit den erwarteten Vorgraben abgeglichen werden. Die folgenden Meilensteine können auch im Gantt-Diagramm wieder gefunden werden.

Nr.	Meilenstein	Erfüllungskriterien	Geplantes Datum
M1	Informieren	Der IPERKA-Schritt Informieren ist vollständig dokumentiert und abgeschlossen.	30.11.2021
M2	Planen	Der IPERKA-Schritt Planen ist vollständig dokumentiert und abgeschlossen.	01.12.2021
M3	Entscheiden	Der IPERKA-Schritt Entscheiden ist vollständig dokumentiert und abgeschlossen.	01.12.2021
M4	Realisieren	Der IPERKA-Schritt Realisieren ist vollständig dokumentiert und abgeschlossen.	06.12.2021
M5	Kontrollieren	Der IPERKA-Schritt Kontrollieren ist vollständig dokumentiert und abgeschlossen.	07.12.2021
M6	Auswerten	Der IPERKA-Schritt Auswerten ist vollständig dokumentiert und abgeschlossen.	07.12.2021
M7	Dokumentation	Die Dokumentation ist vollständig abgeschlossen.	08.12.2021

Tabelle 4 - Meilensteine

6.2 Gantt Diagramm

		Datum	29.11.2021			30.11.2021			01.12.2021			06.12.2021			07.12.2021			08.12.2021	
		Std.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
I	Kurzfassung erstellen	Soll																	
		Ist																	
	Auftrag analysieren	Soll						M1											
		Ist						M1											
P	Projektbeschreibung erstellen	Soll																	
		Ist																	
	Diagramme erstellen	Soll																	
		Ist																	
	Testkonzept erstellen	Soll									M2								
		Ist									M2								
E	Entscheidungen treffen	Soll									M3								
		Ist										M3							
R	Datenbank aufsetzen	Soll																	
		Ist																	
	Backend & Frontend User Applikation erstellen	Soll												M4					
		Ist																M4	
K	Tests durchführen & Test-Fazit schreiben	Soll														M5			
		Ist																M5	
A	Reflexion & Fazit schreiben	Soll															M6		
		Ist																	M6
D	Gantt-Diagramm erstellen	Soll																	
		Ist																	
	Dokumentstruktur anpassen und erweitern	Soll																	
		Ist																	
	Einleitung schreiben und Aufgabenstellung einfügen	Soll																	
		Ist																	
	Teil 1 fertigstellen (Ohne Arbeitsprotokolle)	Soll																	
		Ist																	
	Arbeitsprotokoll schreiben	Soll																	
		Ist																	
	Dokument korrigieren und finalisieren	Soll																M7	
		Ist																	M7
Z	Besuch Experte	Soll																	
		Ist																	

Legende: Soll | Ist | MX (X steht für die Nummer des Meilensteins)

Tabelle 5 - Gantt Diagramm

7 Zeitmanagement

7.1 Massnahmen zur Einhaltung der Zeitplanung

Die wichtigste Massnahme zur Einhaltung der Zeitplanung, die ich einsetzte, ist das Einplanen von Puffern. Dies erlaubt es einem bei Verzögerungen im Zeitplan zu bleiben ohne, dass die Qualität der Arbeit darunter leidet. Falls dann keine Verzögerungen auftreten, können kleine Dinge in der Arbeit verbessert werden.

7.2 Massnahmen bei Verzug

Sobald ein Meilenstein oder eine Tätigkeit nicht zur definierten Zeit erledigt werden konnte, muss eine Massnahme ergriffen werden, damit weitere wichtige Tätigkeiten nicht in Gefahr geraten. So können zum Beispiel kleinere, weniger wichtige Dinge weggelassen oder zumindest verschoben werden. Auch können Kann-Kriterien weggelassen werden.

Trotz diesen Massnahmen kann es noch vorkommen, dass wichtige Dinge nicht mehr erledigt werden können. In diesem Fall sollte mit dem Experten gesprochen werden, um Massnahmen für die Weiterarbeit zu definieren. So können dann zum Beispiel auch Muss-Kriterien weggelassen werden.

8 Arbeitsprotokolle

Das Arbeitsprotokoll dient zur Repräsentation des Tagesablaufs und der Fortschrittbelegung. An jedem Arbeitstag wird das Arbeitsprotokoll erweitert und gepflegt. Das Arbeitsprotokoll ist nach der Methode von Ivy Lee aufgebaut. Die Methode soll die Produktivität erhöhen, indem am Morgen die Arbeit in kleinen Stücken geplant und priorisiert wird. Danach werden die Punkte nach der festgelegten Reihenfolge abgearbeitet. Die Liste wird jeweils abends neu zusammengestellt. So können der Arbeitsverlauf und mögliche Verzögerungen eingesehen werden. Auch wird täglich eine Reflexion über den Tagesverlauf und den Fortschritt geschrieben. Es können auch Massnahmen für aufgetretene Erfolge und Misserfolge definiert werden.

8.1 Tag 1 – 29.11.2021

Geplante Arbeit	Dauer geplant	Dauer tatsächlich	Priorität	Erledigt
Zeitplan erstellen	3h	3h	3	X
Dokumentstruktur anpassen und erweitern	2h	2h	2	X
Einleitung erstellen und Aufgabenstellung einfügen	30 min	45min	1	(X)
Experten Gespräch	30 min	15 min	4	X

Tabelle 6 - Arbeitsprotokoll Tag 1

Zusammenfassung

Die IPA habe ich mit dem Erstellen des Zeitplans begonnen. Danach habe ich die Dokumentstruktur angepasst und erweitert. Ziemlich genau als ich damit fertig wurde hatte ich das erste Gespräch mit dem Experten. Am Schluss habe ich noch die Aufgabenstellung in die Dokumentation eingefügt und die vom Experten vorgeschlagenen Änderungen implementiert. Zur Einleitung bin ich heute nicht mehr gekommen.

Probleme

Das Einfügen der Aufgabenstellung hat länger gedauert, da ich Probleme mit dem Format hatte. So bin ich heute nicht mehr zur Einleitung der IPA gekommen.

Hilfestellung

Remo Steinmann (Experte) hat mir Fragen beantwortet und mit mir das Experten Gespräch durchgeführt.

Ausserplanmässige Arbeiten

Keine

Reflexion

Ich konnte gut in die IPA starten, habe aber das Einfügen der Aufgabenstellung ein wenig unterschätz. Deshalb bin ich heute nicht mehr zum Schreiben der Einleitung gekommen. Mit den anderen Arbeiten bin ich jedoch zufrieden und ich bin weiterhin zuversichtlich, dass ich die IPA gut und zeitgerecht erledigen kann.

Massnahmen

Da ich heute mit der Einleitung nicht fertig geworden bin, muss ich dies morgen nachholen.

Nächste Planung

Geplante Arbeit	Dauer geplant	Priorität
Einleitung schreiben nachholen	30 min	5
Teil 1 fertigstellen	3h	4
Kurzfassung schreiben	45 min	3
Auftrag analysieren	1h	2
Projektbeschreibung erstellen	45 min	1

Tabelle 7 - Arbeitsprotokoll Planung für Tag 2

8.1 Tag 2 – 30.11.2021

Geplante Arbeit	Dauer geplant	Dauer tatsächlich	Priorität	Erledigt
Einleitung schreiben nachholen	30 min	30 min	5	X
Teil 1 fertigstellen	3h	3h	4	X
Kurzfassung schreiben	45 min	45 min	3	X
Auftrag analysieren	1h	1h	2	X
Projektbeschreibung erstellen	45 min	30 min	1	X

Tabelle 8 - Arbeitsprotokoll Tag 2

Zusammenfassung

Ich konnte heute alle ursprünglich geplanten Aufträge erledigen. Mit der Projektbeschreibung bin ich sogar ein wenig schneller fertig geworden als geplant. Auch konnte ich das Schreiben der Einleitung nachholen.

Probleme

Ich bin heute auf keine grossen Probleme gestossen. Da ich heute jedoch eher müde war und deshalb ein wenig langsamer gearbeitet habe, bin ich mit der Qualität der heutigen Arbeit nicht ganz zufrieden. Ich konnte nämlich nicht ganz genug Zeit investieren, um diese sicherzustellen.

Hilfestellung

Remo Steinmann hat uns weitere Inputs mit verschiedenen Tipps für die IPA und IPA-Vorbereitung gegeben.

Ausserplanmässige Arbeiten

Ich musste heute die Einleitung der IPA nachschreiben. Diese hätte ich eigentlich gestern fertigstellen sollen.

Reflexion

Ich konnte heute alle Aufgaben erledigen, auch diese, die ich gestern leider nicht mehr erledigen konnte. Von der Zeitplanung her waren die für heute geplanten Aufgaben jedoch fast alle ein wenig zu knapp eingeplant. Ich hätte bei allen mehr Zeit brauchen können. Auch habe ich die vorherige Nacht sehr schlecht geschlafen und war deshalb heute ein wenig langsamer. Dies führt dazu, dass ich mit der Qualität der Texte, die ich heute verfasst habe, nicht ganz zufrieden bin. Deshalb werde ich diese Texte wahrscheinlich am letzten Tag im Schritt «Dokumentation korrigieren und finalisieren» nochmals ein wenig überarbeiten.

Massnahmen

Heute müssen keine Massnahmen getroffen werden.

Nächste Planung

Geplante Arbeit	Dauer geplant	Priorität
Diagramme erstellen	3h	3
Testkonzept erstellen	2h	2
Entscheidungen treffen	1h	1

Tabelle 9 - Arbeitsprotokoll Planung für Tag 3

8.1 Tag 3 – 01.12.2021

Geplante Arbeit	Dauer geplant	Dauer tatsächlich	Priorität	Erledigt
Diagramme erstellen	3h	3.5h	3	X
Testkonzept erstellen	2h	2.5h	2	X
Entscheidungen treffen	1h	0h	1	-

Tabelle 10 - Arbeitsprotokoll Tag 3

Zusammenfassung

Heute habe ich alle Diagramme, die ich benötigen werde, erstellt und habe das Testkonzept erstellt. So habe ich den Planungs-Schritt abgeschlossen.

Zum Entscheiden bin ich aber leider nicht mehr gekommen. Diesen werde ich nächste Woche nachholen müssen.

Probleme

Ich habe länger als geplant an den Diagrammen und am Testkonzept gebraucht und habe deshalb zu wenig Zeit für das Entscheiden gehabt.

Hilfestellung

Heute konnten wir Remo noch einige Fragen stellen. Dies habe ich aber nicht getan, da ich keine hatte.

Ausserplanmässige Arbeiten

Es gab heute keine ausserplanmässigen Arbeiten.

Reflexion

Beim Diagramme erstellen bin ich eher langsam vorangekommen. Ich habe schon sehr lange keine Diagramme mehr erstellt und musste mich deshalb bei jedem Diagramm zuerst wieder daran erinnern, was die Regeln dieser sind. Zusätzlich habe ich in der Planung nicht daran gedacht, dass ich einige Diagramme für beide möglichen Varianten erstellen muss. Beim Erstellen des Testkonzept hatte ich keine Probleme, es war einfach eine grössere Arbeit als ursprünglich geplant. Deshalb bin ich heute nicht mehr zum Entscheiden gekommen.

Massnahmen

Da ich mit dem IPERKA Schritt «Entscheiden» heute nicht fertig geworden bin, muss ich dies nächste Woche nachholen. Aus diesem Grund ist es eher unwahrscheinlich, dass ich noch genügend Zeit haben werde, um die Kann-Kriterien dieses Projektes zu erledigen. So wird meine Applikation wahrscheinlich kein Frontend haben.

Nächste Planung

Geplante Arbeit	Dauer geplant	Priorität
Entscheidungen treffen nachholen	1h	4
Datenbank-Server aufsetzen	2h	3
Backend Applikation	2.5h (von 4h)	2
Frontend Applikation (Kann-Kriterien)	0h (von 2h)	1

Tabelle 11 - Arbeitsprotokoll Planung für Tag 4

8.1 Tag 4 – 06.12.2021

Geplante Arbeit	Dauer geplant	Dauer tatsächlich	Priorität	Erledigt
Entscheidungen treffen nachholen	1h	1.5h	4	X
Datenbank-Server aufsetzen	2h	3h	3	X
Backend Applikation	2.5h (von 4h)	1.25h	2	-
Frontend Applikation (Kann-Kriterien)	0h (von 2h)	0h	1	-

Tabelle 12 - Arbeitsprotokoll Tag 4

Zusammenfassung

Heute habe ich den IPERKA-Schritt «Entscheiden» nachgeholt und habe den Datenbank Server aufgesetzt. Beide diese Aufgaben haben länger als geplant gebraucht. Auch habe ich mit dem Implementieren des Backends begonnen. Ich konnte jedoch noch keine Funktionalitäten vollständig abschliessen.

Probleme

Da ich für das Entscheiden und das Aufsetzen des Datenbank-Servers länger gebraucht habe, hatte ich weniger Zeit für das Implementieren des Backends. Auch kann ich jetzt mit Sicherheit sagen, dass ich keine Zeit mehr haben werde, um das Frontend, also die beiden Kann-Kriterien, umzusetzen.

Hilfestellung

Wir konnten erneut Remo fragen stellen. Ich habe gefragt, ob wir im Planungsschritt Diagramme von allen möglichen Varianten integrieren sollen. Diese Frage hat er mit ja beantwortet. Da ich dies bereits so gemacht habe, musste ich keine Änderungen vornehmen.

Ausserplanmässige Arbeiten

Ich musste heute das Dokumentieren des Kapitels «Entscheiden» nachholen. Sonst gab es keine ausserplanmässigen Arbeiten.

Reflexion

Beim Entscheiden zwischen den Varianten habe ich ein wenig länger gebraucht, da mir nicht genügend Kriterien eingefallen sind.

Den Datenbank-Server konnte ich schnell aufsetzen. Jedoch musste ich diesem noch Benutzer hinzufügen, die ausserhalb des Servers darauf zugreifen können, da ich den Datenbank-Server zuhause auf meinem Ubuntu-Server aufgesetzt habe. Dies hat um einiges länger gebraucht, da ich nicht wusste, wie man diese Benutzer und Einstellungen erstellen und ändern muss. Auch hatte ich noch ein kleines Problem mit der Firewall.

Massnahmen

Wie im Kapitel «Probleme» schon beschrieben, musste ich die Kann-Kriterien zum Frontend streichen, da ich zu wenig Zeit dafür haben werde. Auch werden ich Arbeit am Backend nachholen müssen, da ich hier ebenfalls hinter meiner Planung bin.

Nächste Planung

Geplante Arbeit	Dauer geplant	Priorität
Experten Gespräch	30 min	4
Backend Applikation	3h	3
Tests durchführen & Test-Fazit schreiben	1h	2
Reflexion & Fazit schreiben	1.25h	1

Tabelle 13 - Arbeitsprotokoll Planung für Tag 5

8.1 Tag 5 – 07.12.2021

Geplante Arbeit	Dauer geplant	Dauer tatsächlich	Priorität	Erledigt
Experten Gespräch	30 min	30 min	4	X
Backend Applikation	3h	5h	3	-
Tests durchführen & Test-Fazit schreiben	1h	0h	2	-
Reflexion & Fazit schreiben	1.25h	0h	1	-

Tabelle 14 - Arbeitsprotokoll Tag 5

Zusammenfassung

Heute konnte ich nur die Hälfte der geplanten Aufträge erledigen. Grund dafür war Node.Js. Ich konnte mich nämlich nicht mit der MongoDB Datenbank verbinden. Später hat sich herausgestellt, dass dies an meiner Node.Js Version gelegen hat. Ich habe also versucht Node.Js zu updaten. Dies hat mich fast 2 Stunden gebraucht, da ich Probleme mit Administratoren-Berechtigungen auf meinem Laptop hatte. Nach diesen 2 Stunden hatte ich Node.Js endlich geupdatet, konnte aber immer noch nicht arbeiten, da nun einige Libraries mit der neuen Node.Js Versionen hatte. Schlussendlich konnte ich nur ca. 30-60 Minuten wirklich programmieren, wieso ich noch keine Funktionalitäten abschliessen konnte.

Probleme

Wie in der Zusammenfassung genannt, hatte ich Probleme mit Node.Js und der Verbindung zur MongoDB Datenbank.

Hilfestellung

Heute hatte ich das Expertengespräch mit Remo. Ich habe ihm meinen derzeitigen Stand erklärt und ihm meine Dokumentation und mein Backend gezeigt. Er hat mir dazu einige Tipps gegeben. Auch hat er mir einige Fragen beantwortet.

Ausserplanmässige Arbeiten

Heuete gab es keine ausserplanmässigen Arbeiten.

Reflexion

Ich bin heute sehr schlecht vorangekommen und bin deswegen nicht ansatzweise so weit gekommen, wie ich es geplant habe. Ich hätte früher aufhören sollen am Backend zu arbeiten und hätte dokumentieren sollen, anstatt so viel Zeit daran zu verschwenden. Das Expertengespräch ist gut verlaufen.

Massnahmen

Nachholen aller nicht erledigten Aufgaben.

Nächste Planung

Geplante Arbeit	Dauer geplant	Priorität
Backend Grund-Funktionalitäten fertigstellen	1.5h	1
Tests durchführen & Test-Fazit schreiben	20 min	1
Reflexion & Fazit schreiben	20 mi	1
Dokument finalisieren	30 min	1

Tabelle 15 - Arbeitsprotokoll Planung für Tag 6

8.1 Tag 6 – 08.12.2021

Geplante Arbeit	Dauer geplant	Dauer tatsächlich	Priorität	Erledigt
Backend Grund-Funktionalitäten fertigstellen	1.5h	1.5h	1	X
Tests durchführen & Test-Fazit schreiben	20 min	20 min	1	X
Reflexion & Fazit schreiben	20 mi	30 min	1	X
Dokument finalisieren	30 min	10 min	1	(X)

Tabelle 16 - Arbeitsprotokoll Tag 6

Zusammenfassung

Heute habe ich noch 3 Grund-Funktionen vom Backend implementiert. Dies lauten: SignUp, LogIn und CheckPermission. Danach habe ich die Tests durchgeführt und den Kontrollieren Schritt abgeschlossen. Als nächstes habe ich den Schritt «Auswerten» abgeschlossen und habe danach noch das Dokument so gut wie möglich finalisiert. Leider hatte ich keine Zeit mehr das Abbildungsverzeichnis zu generieren und das Tabellenverzeichnis zu aktualisieren. Auch das Glossar ist sehr kurz geraten und enthält nicht alle wichtigen Begriffe.

Probleme

Heute hatte ich Zeitprobleme, da ich vieles auf heute verschoben habe. Wenn ich mich an die Planung gehalten hätte, hätte heute alles ohne Probleme fertiggestellt werden können.

Hilfestellung

Remo hat mir mitgeteilt, wie ich die IPA abgeben soll. Und hat mir erneut die Möglichkeit gegeben Fragen zu stellen.

Ausserplanmässige Arbeiten

Heute gab es keine ausserplanmässigen Arbeiten.

Reflexion

Ich habe heute sehr schnell und effizient gearbeitet, habe mir aber heute sehr viel vorgenommen, da ich viel auf heute verschoben habe. So bin ich mit dem Finalisieren des Dokuments nicht ganz fertig geworden. Alle anderen Aufgaben konnte ich erledigen.

TEIL 2

9 Kurzfassung

9.1 Ausgangslage

Die Aufgabenstellung beschreibt ein System, mit welchem Benutzer zusammen mit ihren Rechten und Daten zentral verwaltet werden können. Diese Benutzer sollen dann weiteren Systemen zur Verfügung stehen und in diesen die Authentifizierung und Autorisierung erleichtern. Zurzeit existiert jedoch noch keines dieser Systeme.

9.2 Umsetzung

Die Umsetzung umfasst unter anderem ein MongoDB Datenbank-Server, der für das Speichern der zuvor genannten Informationen zuständig ist. Zusätzlich gibt es ein Backend, den Hauptteil des Systems, welches diese Informationen verwaltet und validiert. Ursprünglich wäre auch ein Frontend geplant gewesen. Für dieses ist nun aber keine Zeit mehr geblieben.

9.3 Ergebnis

Die Applikation konnte teilweise erfolgreich umgesetzt werden. Es musste auf einige Features verzichtet werden, da wegen mehreren Problemen keine Zeit mehr für diese übrig blieb. Diese Features sind, das Frontend, und das Einsehen und Bearbeiten von Benutzerinformationen und Berechtigungen.

10 Informieren

Das Informieren umfasst die Auftragsanalyse, die Präzisierung der Aufgabe und die Abklärungen mit dem Kunden oder dem Auftraggeber. Es ist der erste Schritt in der gewählten Projektmethode IPERKA. In dieser Phase können offene Fragen beantwortet und potenzielle zukünftige Fehler vermieden werden. Dieser Schritt gilt als Grundlage für das nächste Kapitel «Planen».

10.1 Projektbeschreibung

Es soll eine Applikation zur Verwaltung von Benutzern geben. In dieser sollen allen Benutzern verschiedene Rechte vergeben werden können. Auch sollen von jedem Benutzer einige Daten gespeichert werden. Die Mindestanforderungen schreiben keine graphische Benutzeroberfläche vor. Wenn genug Zeit bleibt, wird eine Webseite mit zwei Seiten erstellt, auf der die Benutzerinformationen und Rechte eingesehen und bearbeitet werden können.

10.2 Präzisierung der Aufgabenstellung

Die Benutzer sollen über einen SignUp-Endpoint erstellt werden können und sollen sich über einen SignIn-Endpoint einloggen können. Zusätzlich soll es ein CheckPermission-Endpoint geben, mit welchem überprüft werden kann, ob ein Benutzer Rechte hat eine bestimmte Aktion auszuführen.

Wenn genug Zeit bleibt, soll es zwei Seiten auf einer Webseite geben, mit der die Benutzerinformationen und Rechte angezeigt und bearbeitet werden können. Die Rechte sollen nur bearbeitet werden können, wenn man genug Rechte hat die Aktion «Rechte bearbeiten» auszuführen. Für diese Seiten werden zusätzlich weitere Endpoints benötigt.

10.3 Abklärungen mit dem Kunden

Es musste nichts mit dem Kunden abgeklärt werden.

11 Planen

Im zweiten Schritt der IPERKA-Methode werden nun die möglichen Lösungsvarianten und das Vorgehen ausgearbeitet. Zudem werden Qualitäts- / sowie Funktions-Kriterien für die «Kontrollieren»-Phase erstellt.

11.1 Projektbeschreibung

Das Projekt umfasst das Erstellen einer Backend-Applikation zur Verwaltung von Benutzern. Diese Benutzer haben verschiedene Rechte und Benutzerinformationen.

Dabei sollen die verschiedenen Rechte eines Benutzers getestet und abgefragt werden können. So müssen andere Systeme kein eigenes Benutzersystem implementieren, sondern können diesen Service für ihre Authentifizierung und Autorisierung verwenden.

11.2 Technologie

Als Programmiersprachen werden JavaScript und TypeScript eingesetzt. Diese werden über Node.Js ausgeführt. Für den Datenbankserver wird MongoDB verwendet.

Für die Webseite, wenn genügend Zeit bleibt diese zu erstellen, werden HTML, CSS und JavaScript gebraucht.

11.3 Systembeschreibung

Das System umfasst das Erstellen und Verwalten von Benutzern und ihren Rechten. Diese Informationen werden in einer Datenbank gespeichert und vom Backend des Systems verarbeitet und der Webseite respektive anderen Services übergeben und als Endpoints zur Verfügung gestellt.

Es soll eine Dynamische Liste von Aktionen geben, für die die Benutzer Rechte haben können.

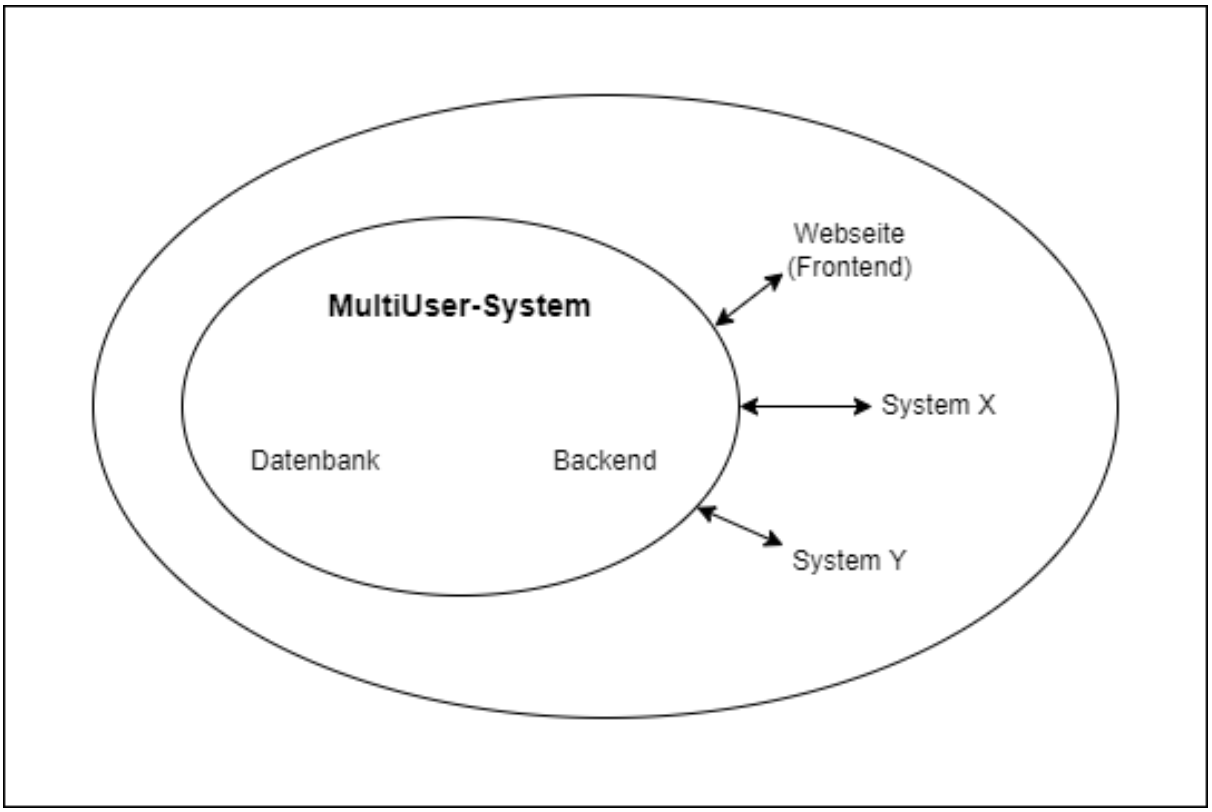
Es soll auch eine Art von Rollen geben. Dafür stehen zwei Möglichkeiten zur Verfügung:

1. Ein Benutzer hat nicht direkt Rechte, sondern gehört einer Rolle an, die verschiedene Rechte hat.
2. Ein Benutzer hat direkt Rechte. Es gibt eine Liste von Rollen, die ebenfalls verschiedene Berechtigungen haben. Wenn einem Benutzer eine Rolle hinzugefügt wird, werden alle Berechtigungen des Benutzers auf die Werte der Rolle gesetzt. Wenn die Berechtigungen des Benutzers dann manuell geändert werden, ist nicht mehr nachzuweisen, welcher Rolle ein Benutzer angehört hat.

Im Abschnitt «Entscheiden» wird zwischen diesen beiden Varianten entschieden.

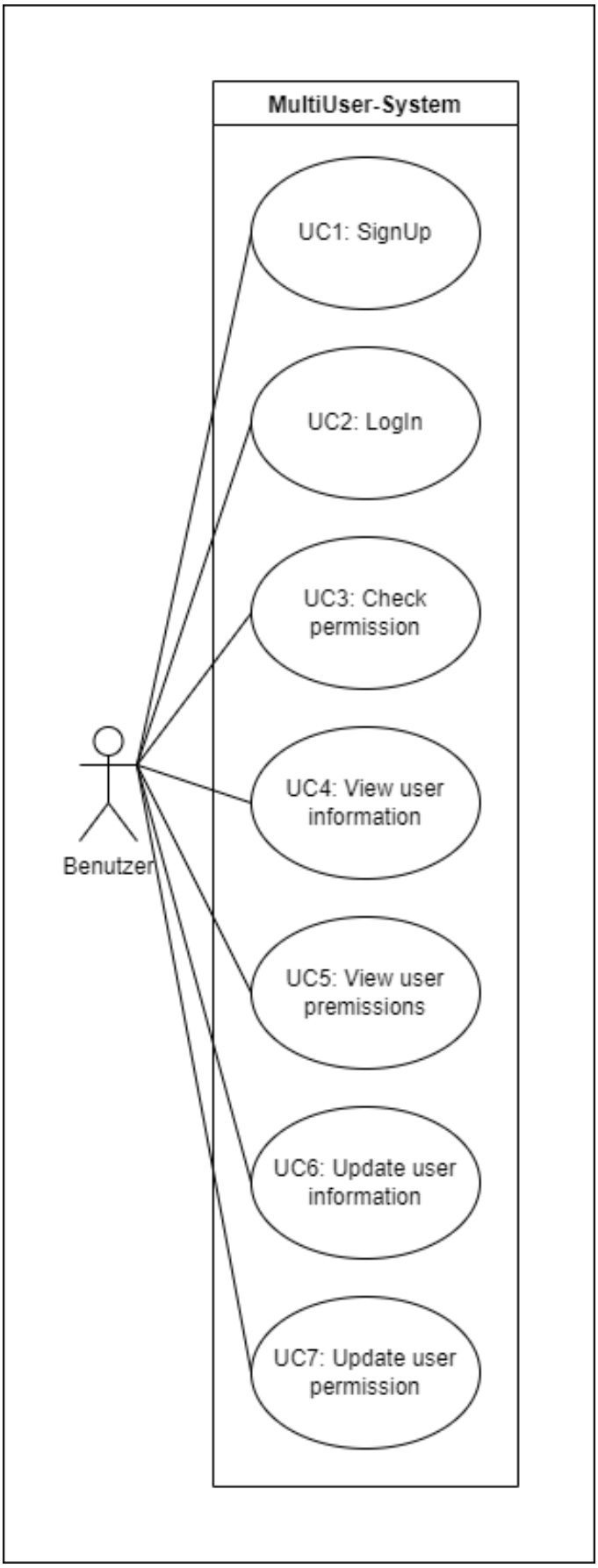
Die Webseite wird sehr einfach gehalten werden. Deshalb wird auch kein Mockup erstellt. Für jedes Feld, das bearbeitet werden kann, wird ein Input-Feld verwendet, für die anderen wird ein deaktiviertes Input-Feld oder einfach Text verwendet.

11.4 Systemgrenzen



Das Systemgrenzen-Diagramm zeigt links das MultiUser-System, welches hier umgesetzt wird. Dieses besteht aus einer Datenbank und dem Backend. Rechts sieht man die Systeme, die das MultiUser-System verwenden, aber nicht zum System gehören. Wie in der Aufgabenstellung bereits erwähnt, existieren diese Systeme bis jetzt noch nicht. Zusätzlich kann man Rechts die Webseite sehen, die in diesem Projekt als Kann-Kriterium implementiert wird.

11.5 Use Case



Das Use Case Diagramm zeigt die Interaktionen des Benutzers mit der Applikation. Da das Frontend ein Kann-Kriterium ist, gelten die Use Cases als erledigt, sobald die Informationen und Aktionen über REST-Endpoints abgefragt beziehungsweise ausgeführt werden können.

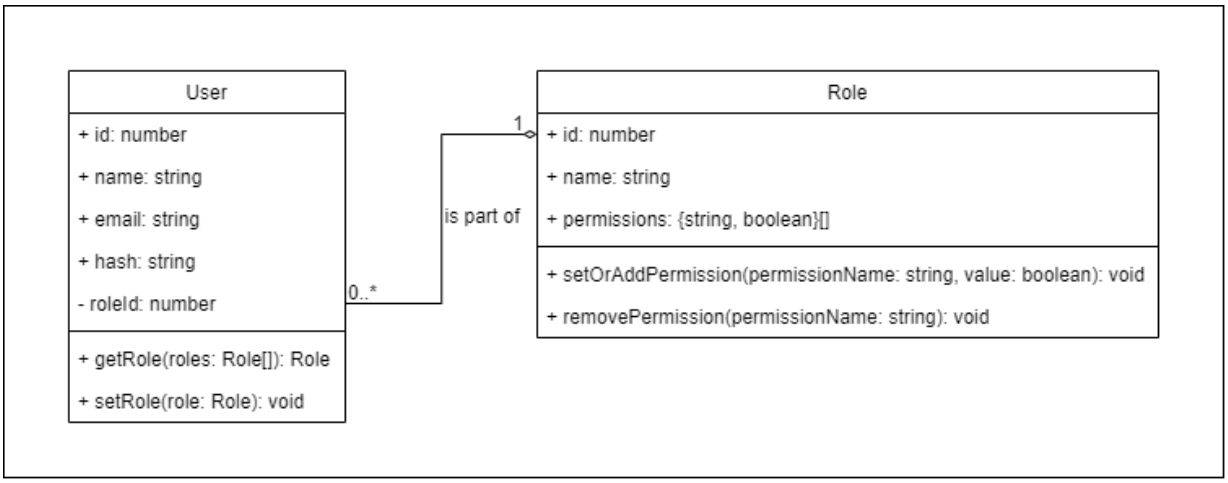
Der Benutzer kann folgende Dinge:

- UC1: SignUp:
  Der Benutzer kann sich mithilfe eines Benutzernamens, einer E-Mail und eines Passworts registrieren. Das Passwort muss dazu zur Bestätigung zwei Mal eingegeben werden. Dabei wird ein neuer Benutzer mit Standard-Berechtigungen erstellt. Auch wird der Benutzer danach automatisch nach UC2 eingeloggt.
- UC2: LogIn:
  Der Benutzer gibt seine E-Mail oder seinen Benutzernamen und sein Passwort an und wird bei korrekten Informationen eingeloggt. Als Antwort erhält er einen Session-Token.
- UC3: Check permission:
  Ein eingeloggter Benutzer gibt den Namen einer Berechtigung ein und erhält als Antwort, ob er die Rechte dafür hat.
- UC4: View user information:
  Ein eingeloggter Benutzer kann seine eigenen Benutzerinformationen abfragen und hat je nach Berechtigung auch auf die Benutzerinformationen von anderen Benutzern Zugriff.

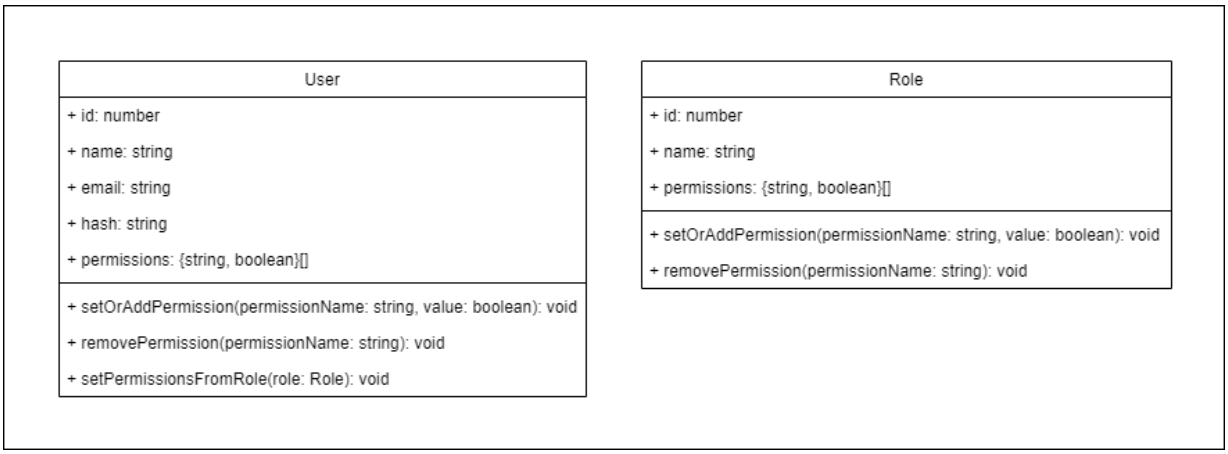
- UC5: View user permissions:
  Ein eingeloggter Benutzer kann seine eigenen Berechtigungen abfragen und hat je nach Berechtigung auch auf die Berechtigungen von anderen Benutzern Zugriff.
- UC6: Update user information:
  Ein eingeloggter Benutzer kann seine eigenen Benutzerinformationen aktualisieren und hat je nach Berechtigung auch die Möglichkeit die Benutzerinformationen von anderen Benutzern zu ändern.
- UC7: Update user permission:
  Ein eingeloggter Benutzer kann seine eigenen Berechtigungen aktualisieren und hat je nach Berechtigung auch die Möglichkeit die Berechtigungen von anderen Benutzern zu ändern.

11.6 Klassendiagramm

11.6.1 Variante 1



11.6.2 Variante 2



11.7 Datenbank Struktur

Datenbank: auth

11.7.1 Variante 1

Collection: users

- id [number]
- name [string]
- email [string]
- hash [string]
- roleId [number]

Collection: roles

- id [number]
- name [string]
- permissions [Collection]
    - <permissionName1> [boolean]
    - <permissionName2> [boolean]

11.7.2 Variante 2

Collection: users

- id [number]
- name [string]
- email [string]
- hash [string]
- permissions [Collection]
    - <permissionName1> [boolean]
    - <permissionName2> [boolean]

Collection: roles

- id [number]
- name [string]
- permissions [Collection]
    - <permissionName1> [boolean]
    - <permissionName2> [boolean]

12 Entscheiden

In der «Entscheiden»-Phase werden die in der Planungsphase erarbeiteten Lösungsvarianten analysiert, verglichen und eine davon ausgewählt. Dafür wird eine Entscheidungsmatrix angewendet. Für diese werden verschiedene Faktoren, mit welchen die Lösungsvarianten verglichen werden können, benötigt.

12.1 Auswahlkriterien

Als erstes Auswahlkriterium wurde die Konfigurationsmöglichkeit gewählt. Damit ist gemeint, wie genau einzelne Berechtigungen vergeben werden können, und ob diese von vorgegebenen Rollen abweichen können.

Das zweite Kriterium ist «Best Practice». Diese beschreibt den best practice mit einer MongoDB Datenbank. Bei Dokumentbasierten Datenbanken, wie MongoDB, wird nämlich davon abgeraten Fremd-Schlüssel (Foreign Keys) zu verwenden.

Das dritte Auswahlkriterium ist Redundanz / Speicherminimierung. Bei Variante 2 werden für jeden Benutzer alle Berechtigungen einzeln gespeichert. Dies benötigt viel mehr Speicher als bei Variante 1, wo mehrere Benutzer auf die gleiche Rolle verweisen können. Hier speichert diese dann die Berechtigungen.

Die Möglichkeit die Rolle eines Benutzers auszulesen ist das vierte Kriterium und die benötigte Entwicklungszeit das letzte.

12.2 Entscheidungsmatrix

Varianten		**Variante 1: Benutzer hat Rolle.**		**Variante 2: Benutzer hat direkt Berechtigungen**	
Kriterium	**Gewichtung**	**Erfüllt**	**Bewertung**	**Erfüllt**	**Bewertung**
Konfigurationsmöglichkeit	30	Gering (0)	0	Hoch (1)	30
Best Practice	25	Nein (0)	0	Ja (1)	25
Redundanz / Speicherminimierung	20	Ja (1)	20	Nein (1)	0
Rolle auslesbar	15	Ja (1)	15	Nein (0)	0
Entwicklungszeit	10	Mittel (0.5)	5	Gering (1)	10
Total	**100**	**40**		**65**	
Platzierung		**2**		**1**	

12.3 Gewählte Parameter

Die Gewichtung wurde an die Wichtigkeit des Kriteriums angepasst. Da der Entwicklungszeit Unterschied zwischen Variante 1 und Variante 2 relativ klein ist, wurde dem Kriterium Entwicklungszeit eine tiefe Gewichtung gegeben. In der «Erfüllt»-Spalte steht in Klammern der Faktor, mit dem die Gewichtung multipliziert wird, um die Bewertung zu erhalten.

13 Realisieren

Das Realisieren und somit der vierte Schritt der IPERKA-Methode, ist die zeitintensivste Phase in den meisten Projekten. Hier werden die zuvor erfüllten Schritte von IPERKA aus der Theorie in die Praxis umgewandelt. Dabei ist es wichtig den Arbeitsablauf einzuhalten und diesen nicht ohne zwingende Gründe zu ändern. In diesem Schritt ist es äusserst wichtig genügend Zeit für das Arbeitsprotokoll einzuplanen, sodass schnell auf Misserfolge reagiert werden kann und die daraus entstehenden Massnahmen geplant und umgesetzt werden können.

13.1 Umsetzung

Als erstes wurde der MongoDB Datenbank Server aufgesetzt und auf geschützten öffentlichen Zugriff konfiguriert. Dies wurde über Remote-Desktop auf meinem Ubuntu-Server gemacht. Danach wurde mit dem Backend begonnen. Dafür wurden zuerst alle benötigten Libraries installiert und danach den Zugriff auf die MongoDB Datenbank getestet. Dabei sind Inkompatibilitäten mit der Node.Js Version aufgetreten, wieso Node.Js geupdatet werden musste. Daraufhin wurden die verschiedenen Klassen nach dem Klassendiagramm in der Planung unter Variante 2 implementiert. Da zu diesem Zeitpunkt nur noch wenig Zeit für das Projekt übrigblieb, wurden danach nur noch die ersten 3 geplanten Features implementiert. Diese sind:

- SignUp
- LogIn
- CheckPermission

13.2 Installationsanweisung

- «Backend» Ordner auf Github (https://github.com/DominicLandolt/M223_ProbeIPA_MultiUser) herunterladen.
- Node.Js Version 16.X herunterladen.
- «sample.env»-Datei in «.env» umbenennen und fehlende Werte darin ersetzten. (Im «Backend» Ordner
- Konsole öffnen und in «Backend» Ordner navigieren.
- Befehl «npm i» ausführen. Dies installiert alle benötigten Libraries.
- Befehl «npm run start» ausführen. Dies kompiliert den TypeScript code und führt danach das Programm aus. Falls dieser Schritt nicht funktioniert muss TypeScript möglicherweise noch manuell heruntergeladen werden. Dies kann mit dem Befehl «npm install -g typescript» erreicht werden.

13.3 Benutzeranleitung

Da die Applikation kein Frontend hat, kann sie nur mit REST-Request getestet werden. Die Applikation hört auf dem Port, der zuvor in der «.env»-Datei eingegeben wurde. Es gibt drei Endpoints:

- signup
- login
- checkpermission

Alle davon hören nur auf GET-Requests. Ich empfehle ein Programm namens «Postman», um diese Requests zu senden.

Hier sind einige gültige Request, die an das Backend gesendet werden können. Wenn Parameter wegelassen werden oder verfälscht werden, gibt das Programm ein Fehler aus. <SESSION-TOKEN> sollte mit einem zurzeit gültigen Session-Token ersetzt werden.

- `localhost:4646/signup?UserName=Test123456&Email=Test123456@Test.com&Password=123456&RepeatPassword=123456`
- `localhost:4646/login?UserName=Test123456&Password=123456`
- `localhost:4646/login?Email=Test123456@Test.com&Password=123456`
- `localhost:4646/checkpermission?SessionToken=<SESSION-TOKEN>&Permission=testPermission`
- `localhost:4646/checkpermission?SessionToken=<SESSION-TOKEN>&Permission=TEST`

14 Kontrollieren

Der wichtigste Schritt der IPERKA-Methode ist das Kontrollieren. Eine Arbeit kann nicht abgegeben werden, ohne dass vorher nochmals alles kontrolliert wurde. Hierbei werden vor allem die Vorgaben, welche sich aus der Planungsphase ergeben haben, mit dem tatsächlichen Ergebnis verglichen und die Resultate festgehalten. Neben der Kontrolle des eigentlichen Produktes ist es auch wichtig das Arbeitsverhalten zu überprüfen und allenfalls Abweichungen aus den Plänen aufzuzeigen.

14.1 Testkonzept

Die Applikation wird aus Zeitgründen nur manuell getestet. Um Unit-Tests aufzusetzen hätte ich zu lange gebraucht. Die Testfälle werden stückweise während der Realisierung durchgeführt, um eine Lauffähige Version zu gewährleisten. Es wird jedoch nur die letzte Durchführung der Tests dokumentiert.

Hierbei wird eine Testgruppe abgearbeitet und die einzelnen Tests manuell durchgeführt. So kann schnell ein Überblick über die funktionierenden Features gewährleistet werden.

14.2 Testumgebung

Die Tests werden auf meinem Arbeitslaptop durchgeführt.

- Lenovo P70 (I7-6820HQ & 16 GB RAM)

Dieser verwendet zurzeit das Betriebssystem Windows 10 Pro mit der Version 19042 (20H2).

Die Tests sollten aber auf allen Systemen gleich funktionieren, da die Applikation auf einem externen Ubuntu Server gehostet wird.

14.3 Testart

Die Testart sieht folgendermassen aus:

- Voraussetzung für den Testfall
- Test-Schritte
- Erwartetes Ergebnis

Beim Evaluieren, ob die Tests erfolgreich waren, werden Mängelklassen verwendet. Diese lauten 0: Mangelfrei, 1: Belangloser Mangel, 2: Leichter Mangel, 3: Schwerer Mangel und 4: Kritischer Mangel.

Dabei ist das Programm oder Feature bei der Mängelklasse 4 unbrauchbar und bei der Mängelklasse 0 gibt es nichts zu bemängeln.

14.4 Testfälle

14.4.1 Testübersicht

Use Case	Test Name	
UC1	SignUp	
	T-01	Valide Daten werden eingegeben
	T-02	Invalide Daten werden eingegeben
UC2	LogIn	
	T-03	Valide Daten werden eingegeben
	T-04	Invalide Daten werden eingegeben
UC3	Check permission	
	T-05	Gewollter Berechtigungs-Name wird eingegeben
UC4	View user information	
	T-06	Benutzerinformationen werden mit benötigter Berechtigung angefragt
	T-07	Benutzerinformationen werden ohne benötigte Berechtigung angefragt
UC5	View user permissions	
	T-08	Berechtigungen werden mit benötigter Berechtigung angefragt
	T-09	Berechtigungen werden ohne benötigte Berechtigung angefragt
UC6	Update user information	
	T-10	Valide Daten werden eingegeben und Berechtigungen sind vorhanden
	T-11	Invalide Daten werden eingegeben und Berechtigungen sind vorhanden
UC7	Update user permission	
	T-12	Valide Daten werden eingegeben und Berechtigungen sind vorhanden
	T-13	Valide Daten werden eingegeben und Berechtigungen fehlen

14.4.2 Tests

Alle Testfälle wurden von Dominic Landolt am 08.12.2021 durchgeführt.

Test-ID	T-01 (SignUp)
Test-Name	Valide Daten werden eingegeben
Test-Beschreibung	Es wird der SignUp-Endpoint mit korrekten Daten getestet.
Vorbedingung	Benutzer «Test» existiert nicht bereits.
Nr.	**Test-Schritte**
1	Request an SignUp-Endpoint senden. (UserName = “Test”, Email = “test@test.com”, Password = “Test1234”, RepeatPassword = “Test1234”)
Erwartetes Ergebnis	Applikation antwortet mit einem Session-Token und Benutzer wird erfolgreich erstellt.
Mängelklasse	0

Test-ID	T-02 (SignUp)
Test-Name	Invalide Daten werden eingegeben
Test-Beschreibung	Es wird der SignUp-Endpoint mit falschen oder nicht gültigen Daten getestet.
Vorbedingung	Benutzer «Test» existiert nicht bereits.
Nr.	**Test-Schritte**
1	Request an SignUp-Endpoint senden. (UserName = “Test”, Email = “KeineEmail”, Password = “Test1234”, RepeatPassword = “AnderesPasswort”)
Erwartetes Ergebnis	Es wird kein Benutzer erstellt, die Fehlermeldung zeigt, dass Email oder RepeatPassword Falsch ist.
Mängelklasse	0

Test-ID	T-03 (LogIn)
Test-Name	Valide Daten werden eingegeben
Test-Beschreibung	Es wird der LogIn-Endpoint mit korrekten Daten getestet.
Vorbedingung	Benutzer von T-01 korrekt erstellt.
Nr.	**Test-Schritte**
1	Request an LogIn-Endpoint senden. (UserName = “Test” oder Email = “test@test.com” und Password = “Test1234”)
Erwartetes Ergebnis	Applikation antwortet mit einem Session-Token.
Mängelklasse	0

Test-ID	T-04 (LogIn)
Test-Name	Invalide Daten werden eingegeben
Test-Beschreibung	Es wird der LogIn-Endpoint mit falschen Daten getestet.
Vorbedingung	Benutzer von T-01 korrekt erstellt. Benutzer «ABC» existiert nicht.
Nr.	**Test-Schritte**
1	Request an LogIn-Endpoint senden. (UserName = “ABC” oder Email = “ABC@test.com” und Password = “Test1234”)
Erwartetes Ergebnis	Applikation antwortet mit: “UserName/Email or Password is incorrect”
Mängelklasse	0

Test-ID	T-05 (Check permission)
Test-Name	Gewollter Berechtigungs-Name wird eingegeben
Test-Beschreibung	Es wird geprüft, ob der Benutzer mit dem beigelegten Session-Token die eingegebene Berechtigung hat oder nicht.
Vorbedingung	Valider Session-Token vorhanden (Erfolgreich eingeloggt).
Nr.	**Test-Schritte**
1	Request an CheckPermission-Endpoint senden. (SessionToken = <ValiderSessionToken> und Permission = “BeispielBerechtigung”)
Erwartetes Ergebnis	Die Applikation antwortet mit «True», wenn der Benutzer die Berechtigung hat oder mit «False», wenn er diese nicht hat.
Mängelklasse	0

Test-ID	T-06 (View user information)
Test-Name	Benutzerinformationen werden mit benötigter Berechtigung angefragt
Test-Beschreibung	Ein Benutzer versucht seine eigenen Benutzerdaten anzuzeigen. Dafür benötigt man keine spezielle Berechtigung.
Vorbedingung	Session-Token und id von einem gleichen Benutzer sind bekannt.
Nr.	**Test-Schritte**
1	Request an ViewUserInformation-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdDesSelbenBenutzers>)
Erwartetes Ergebnis	Die Applikation antwortet mit allen wichtigen Informationen des Benutzers (id, UserName, Email).
Mängelklasse	4

Test-ID	T-07 (View user information)
Test-Name	Benutzerinformationen werden ohne benötigte Berechtigung angefragt
Test-Beschreibung	Ein Benutzer versucht die Benutzerdaten eines anderen Benutzers anzuzeigen und hat nicht die nötigen Berechtigungen dazu.
Vorbedingung	Session-Token und id von unterschiedlichen Benutzern sind bekannt. Der User mit dem Session-Token hat nicht die nötigen Berechtigungen.
Nr.	**Test-Schritte**
1	Request an ViewUserInformation-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdEinesAnderenBenutzers>)
Erwartetes Ergebnis	Die Applikation antwortet mit einem Fehler. Der Benutzer hat zu wenige Berechtigungen, um diese Aktion auszuführen.
Mängelklasse	4

Test-ID	T-08 (View user permissions)
Test-Name	Berechtigungen werden mit benötigter Berechtigung angefragt
Test-Beschreibung	Ein Benutzer versucht seine eigenen Berechtigungen anzuzeigen. Dafür benötigt man keine spezielle Berechtigung.
Vorbedingung	Session-Token und id von einem gleichen Benutzer sind bekannt.
Nr.	**Test-Schritte**
1	Request an ViewUserPermissions-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdDesSelbenBenutzers>)
Erwartetes Ergebnis	Die Applikation antwortet mit allen Berechtigungen des Benutzers.
Mängelklasse	4

Test-ID	T-09 (View user permissions)
Test-Name	Berechtigungen werden ohne benötigte Berechtigung angefragt
Test-Beschreibung	Ein Benutzer versucht die Berechtigungen eines anderen Benutzers anzuzeigen und hat nicht die nötigen Berechtigungen dazu.
Vorbedingung	Session-Token und id von unterschiedlichen Benutzern sind bekannt. Der User mit dem Session-Token hat nicht die nötigen Berechtigungen.
Nr.	**Test-Schritte**
1	Request an ViewUserPermissions-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdEinesAnderenBenutzers>)
Erwartetes Ergebnis	Die Applikation antwortet mit einem Fehler. Der Benutzer hat zu wenige Berechtigungen, um diese Aktion auszuführen.
Mängelklasse	4

Test-ID	T-10 (Update user information)
Test-Name	Valide Daten werden eingegeben und Berechtigungen sind vorhanden
Test-Beschreibung	Es werden eigene Benutzerinformationen geändert.
Vorbedingung	Session-Token und id von einem gleichen Benutzer sind bekannt.
Nr.	**Test-Schritte**
1	Request an UpdateUserInformation-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdDesSelbenBenutzers>, UserName = «NeuerBenutzername», Password = «NeuesPasswort»)
Erwartetes Ergebnis	Die Benutzerinformationen werden nach Anfrage aktualisiert.
Mängelklasse	4

Test-ID	T-11 (Update user information)
Test-Name	Invalide Daten werden eingegeben und Berechtigungen sind vorhanden
Test-Beschreibung	Es werden eigene Benutzerinformationen geändert. Die eingegebenen Daten sind nicht gültig.
Vorbedingung	Session-Token und id von einem gleichen Benutzer sind bekannt.
Nr.	**Test-Schritte**
1	Request an UpdateUserInformation-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdDesSelbenBenutzers>, Email = «NichtGüligeEmail», Password = «NeuesPasswort»)
Erwartetes Ergebnis	Die gültigen Benutzerinformationen werden nach Anfrage aktualisiert. Ungültige Felder, wie hier die Email, werden nicht geändert und es wird ein Fehler ausgegeben.
Mängelklasse	4

Test-ID	T-12 (Update user permission)
Test-Name	Valide Daten werden eingegeben und Berechtigungen sind vorhanden
Test-Beschreibung	Es wird eine eigene Berechtigung geändert. Der Benutzer hat die Berechtigungen dafür.
Vorbedingung	Session-Token und id von einem gleichen Benutzer sind bekannt. Dieser Benutzer hat die Rechte seine Berechtigungen zu bearbeiten.
Nr.	**Test-Schritte**
1	Request an UpdateUserPermission-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdDesSelbenBenutzers>, permissionName, permissionValue = true/false)
Erwartetes Ergebnis	Die Berechtigung wird nach Anfrage aktualisiert
Mängelklasse	4

Test-ID	T-13 (Update user permission)
Test-Name	Valide Daten werden eingegeben und Berechtigungen fehlen
Test-Beschreibung	Es wird eine eigene Berechtigung geändert. Der Benutzer hat nicht die benötigten Berechtigungen dafür.
Vorbedingung	Session-Token und id von einem gleichen Benutzer sind bekannt. Dieser Benutzer hat nicht die Rechte seine Berechtigungen zu bearbeiten.
Nr.	**Test-Schritte**
1	Request an UpdateUserPermission-Endpoint senden. (SessionToken = <ValiderSessionToken> und id = <IdDesSelbenBenutzers>, permissionName, permissionValue = true/false)
Erwartetes Ergebnis	Es wird nichts aktualisiert und die Applikation antwortet mit einem Fehler. Der Benutzer hat zu wenige Berechtigungen, um diese Aktion auszuführen.
Mängelklasse	4

14.5 Testfazit

Es waren 5 der 13 manuellen Tests erfolgreich. Bei diesen Tests gibt es nichts zu bemängeln. Die anderen 8 Tests sind jedoch mit Mängelklasse 4 fehlgeschlagen. Der Grund dafür ist, dass keine Zeit mehr blieb die Funktionalitäten hinter diesen Test zu implementieren. So ist es logisch, dass diese Tests fehlschlagen.

14.6 Auswertung der Meilensteine

Nr.	Meilenstein	Erfüllungskriterien	Geplantes Datum	Tatsächliches Datum
M1	Informieren	Der IPERKA-Schritt Informieren ist vollständig dokumentiert und abgeschlossen.	30.11.2021	30.11.2021
M2	Planen	Der IPERKA-Schritt Planen ist vollständig dokumentiert und abgeschlossen.	01.12.2021	01.12.2021
M3	Entscheiden	Der IPERKA-Schritt Entscheiden ist vollständig dokumentiert und abgeschlossen.	01.12.2021	06.12.2021
M4	Realisieren	Der IPERKA-Schritt Realisieren ist vollständig dokumentiert und abgeschlossen.	06.12.2021	08.12.2021
M5	Kontrollieren	Der IPERKA-Schritt Kontrollieren ist vollständig dokumentiert und abgeschlossen.	07.12.2021	08.12.2021
M6	Auswerten	Der IPERKA-Schritt Auswerten ist vollständig dokumentiert und abgeschlossen.	07.12.2021	08.12.2021
M7	Dokumentation	Die Dokumentation ist vollständig abgeschlossen.	08.12.2021	08.12.2021

Wie man an der Tabelle sehen kann, konnten nur die ersten beiden Meilensteine am geplanten Datum erledigt werden. Danach wurden alle mit Verspätung erledigt, bis auf den «Dokumentation»-Meilenstein, der ja sowieso am Ende des Projekts war.

15 Auswerten

In der letzten IPERKA Phase wird nochmals über die ganze Arbeit reflektiert. Beim Auswerten geht es um eine Selbsteinschätzung der erledigten Tätigkeiten. Es wird notiert was aus der Arbeit gelernt wurde, was verbessert werden kann und wo oder wie das Gelernte in der Zukunft angewendet werden kann.

15.1 Reflexion

Ich finde, dass ich während der Arbeit gut gearbeitet habe, trotzdem bin ich mit dem Projekt aber nicht ansatzweise fertig geworden. Dafür gibt es eigentlich nur zwei Gründe. Diese sind 1. Die Planung und 2. das nicht an die Planung halten. Zum ersten Punkt. Eigentlich habe ich bei meiner Planung schon relativ viel Pufferzeit eingebaut. Trotzdem habe ich viele Aufgaben unterschätzt, wieso ich auch mit dieser Pufferzeit nicht fertig geworden bin. Das Realisieren habe ich bei meiner Planung als letzten eingeplant. Links Davon alles, dass vorher eingeplant werden muss und rechts davon alles was danach getan werden muss. So ist mir nur Total ein Tag für das Realisieren geblieben. Das war von Anfang an eher knapp und bei diesem Schritt wurde definitiv keine Pufferzeit eingeplant. Als ich dann auch noch das «Entscheiden» auf den Tag des Realisierens verschieben musste war klar, dass ich entweder umplanen musste oder weniger umsetzen kann. Ich habe mich für das weniger umsetzen entschieden. Kommen wir zum zweiten Punkt und dem Problem, dass ich mich dann nicht an diese Entscheidung gehalten habe und deswegen auch mit den folgenden Schritten in den Stress gekommen bin. Ich bin mit meiner Arbeit also nicht ganz zufrieden. Hätte ich mich besser über die Node.Js MongoDB Kompatibilität informiert und weniger im Realisieren eingeplant, wäre ich sicher mit dem ganzen Projekt fertig geworden.

In Zukunft möchte ich mich also besser über die Möglichkeit und Kompatibilitäten von den verschiedenen Bausteinen eines Projekts informieren und anhand von diesen besser planen. Auch muss ich lernen zu akzeptieren ein sehr unfertiges Produkt abzugeben, wenn ich es gut begründen kann, anstatt meine Planung über den Haufen zu werfen und am Produkt weiterzuarbeiten.

15.2 Fazit

Das Projekt war für mich eine interessante Arbeit. Während der Arbeit konnte ich mich verschiedenen anspruchsvollen Aufgaben stellen. Eine davon war das Aufsetzen von MongoDB. Die an sich gut ging, einfach komplexer war als ursprünglich gedacht. In Verzug gekommen bin ich nicht, weil ich schlecht gearbeitet habe, sondern weil ich nicht erwartet habe, dass meine Node Version nicht mit MongoDB kompatibel ist und, dass ich keine richtigen Berechtigungen habe, um diese zu aktualisieren. Glücklicherweise konnte ich dank meines Vorwissens die ersten drei Funktionalitäten dann relativ schnell implementieren und so wieder ein wenig Zeit gut machen. Trotzdem hätte ich mich am Schluss mehr auf die Dokumentation fokussieren sollen.

16 Verzeichnis

Anbei werden alle Verwendeten Ressourcen, Abbildungen und Tabellen aufgelistet.

16.1 Quellenverzeichnis

- https://docs.mongodb.com/
- https://stackoverflow.com/

16.2 Tabellen

16.3 Glossar

Im Glossar werden die wichtigsten Fremdwörter und technischen Ausdrücke erklärt.

Library	Bibliothek, Code der einem Projekt hinzugefügt werden kann.
MongoDB	Datenbank-Anbieter & Server.

16.4 Danksagung

Ich möchte meinem Experten, Remo Steinmann, danken, dass er mir immer wieder Tipps und andere Vorschläge gegeben hat und sich Zeit für mich genommen hat.

ANHANG

17 Sourcecode

17.1 Main.ts

```
import jwt from 'jsonwebtoken';
import dotenv from 'dotenv';
import mongodb from "mongodb";
import bcrypt from 'bcryptjs';
import express from 'express';
import { Role } from './Role.js';
import { User } from './User.js';

dotenv.config();

const DB_URL = `mongodb://${process.env.DB_USER}:${process.env.DB_PASS-
WORD}@${process.env.DB_HOST}:${process.env.DB_PORT}?authMechanism=${pro-
cess.env.DB_AUTHMECHANISM}&authSource=${process.env.DB_DATABASE}`;
const DB_CLIENT = new mongodb.MongoClient(DB_URL)
const DB_CONNECTION = DB_CLIENT.connect()

var app = express();
let port: number = parseInt(process.env.PORT);

app.get("/signup", (req, res, next) => {
    let userName = <string>req?.query?.UserName;
    let userEmail = <string>req?.query?.Email;
    let userPassword = <string>req?.query?.Password;
    let userRepeatPassword = <string>req?.query?.RepeatPassword;

    if(userName == null || userName == ""){
        res.status(422).json({ message: "Error: Field \"UserName\" is re-
quired." });
    }
    else if(userEmail == null || userEmail == "" || !validateEmail(userE-
mail)){
        res.status(422).json({ message: "Error: Field \"Email\" is required
and must be valid." });
    }
    else if(userPassword == null || userPassword.length < 6){
        res.status(422).json({ message: "Error: Field \"Password\" must be at
least 6 characters long." });
    }
    else if(userPassword != userRepeatPassword){
        res.status(422).json({ message: "Error: Field \"RepeatPassword\" must
match Field \"Password\"." });
    }
    else{
        DB_CONNECTION.then(() => {
```

```
            const authDB = DB_CLIENT.db("auth");
            const usersColl = authDB.collection('users')

            usersColl.count( { name: escape(userName) }, (error: any, count:
number) => {
                if(error){
                    res.status(500).json({ message: "Error: Could not create
new user. Please try again later." });
                }
                else if(count != 0){
                    res.status(422).json({ message: "Error: A user with this
name already exists. Please choose a different name." });
                }
                else{
                    usersColl.count( { email: escape(userEmail) }, (error:
any, count: number) => {
                        if(error){
                            res.status(500).json({ message: "Error: Could not
create new user. Please try again later." });
                        }
                        else if(count != 0){
                            res.status(422).json({ message: "Error: A user
with this Email already exists. Please use a different email." });
                        }
                        else{
                            bcrypt.hash(userPassword, parseInt(pro-
cess.env.SALT), (err: any, hash: any) => {
                                if(err){
                                    res.status(500).json({ message: "Error:
Could not create new user. Please try again later." });
                                }
                                else{
                                    DB_CONNECTION.then(() => {
                                        const authDB = DB_CLIENT.db("auth");
                                        const rolesColl = authDB.collec-
tion('roles')
                                        rolesColl.findOne({name: "Default"},
(err: any, result: any) => {
                                            if(err){
                                                res.status(500).json({ mes-
sage: "Error: Could not create new user. Please try again later." });
                                            }
                                            else if(result == undefined || re-
sult == null || result == ""){
                                                res.status(500).json({ mes-
sage: "Error: Could not create new user. Please try again later." });
                                            }
                                            else{
```

```
                                                var role = new Role(result);
                                                if(role.id == null ||
role.name == null || role.permissions == null){
                                                    res.status(500).json({
message: "Error: Could not create new user. Please try again later." });
                                                }
                                                else{
                                                    var user = new User(es-
cape(userName), escape(userEmail), hash, role.permissions)
                                                    DB_CONNECTION.then(() => {
                                                        const valueToInsert =
{ name: user.name, email: user.email, hash: user.hash, permissions: user.per-
missions }
                                                        const authDB = DB_CLI-
ENT.db("auth");
                                                        const usersColl =
authDB.collection('users')
                                                        usersColl.inser-
tOne(valueToInsert, (err: any, result: any) => {
                                                            if(err){
                                                                res.sta-
tus(500).json({ message: "Error: Could not create new user. Please try again
later." });
                                                            }
                                                            else{
                                                                let token =
jwt.sign({ user }, process.env.JWTSECRET, { algorithm: 'HS256', expiresIn:
'1h' });
                                                                res.sta-
tus(200).json({ message: "Successfully created account and logged in.", token
});
                                                            }
                                                        })
                                                    });
                                                }
                                            }
                                        });
                                    }
                                );
                            }
                        );
                    }
                );
            });
        });
    }
});
```

```
app.get("/login", (req, res, next) => {
    let userName = <string>req?.query?.UserName;
    let userEmail = <string>req?.query?.Email;
    let userPassword = <string>req?.query?.Password;

    if((userName == null || userName == "") && (userEmail == null || userEmail
== "")){
        res.status(422).json({ message: "Error: At least one of fields
\"UserName\" and \"Email\" is required." });
    }
    else if(userPassword == null || userPassword == ""){
        res.status(422).json({ message: "Error: Field \"Password\" is re-
quired." });
    }
    else if(userName != null && userName != ""){
        //Log in with username
        DB_CONNECTION.then(() => {
            const authDB = DB_CLIENT.db("auth");
            const usersColl = authDB.collection('users')

            usersColl.findOne( { name: escape(userName) }, (error: any, re-
sult: any) => {
                if(error){
                    res.status(500).json({ message: "Error: Could not log in.
Please try again later." });
                }
                else if(result == null || result == undefined || result == ""
|| result._id == null || result._id == undefined || result._id == "" || re-
sult.name == null || result.name == undefined || result.name == "" || re-
sult.email == null || result.email == undefined || result.email == "" || re-
sult.hash == null || result.hash == undefined || result.hash == "" || re-
sult.permissions == null || result.permissions == undefined){
                    res.status(422).json({ message: "Error: Incorrect username
or password." });
                }
                else{
                    var user = new User(result.name, result.email, re-
sult.hash, result.permissions);
                    user.id = result._id;
                    bcrypt.compare(userPassword, user.hash, function(err: any,
bcResult: boolean){
                        if(err){
                            res.status(500).json({ message: "Error: Could not
log in. Please try again later." });
                        }
                        else{
                            if(bcResult){
```

```
                                let token = jwt.sign({ user }, pro-
cess.env.JWTSECRET, { algorithm: 'HS256', expiresIn: '1h' });
                                res.status(200).json({ message: "Successfully
logged in.", token });
                            }
                            else{
                                res.status(422).json({ message: "Error: Incor-
rect username or password." });
                            }
                        }
                    });
                }
            });
        });
    }
    else{
        if(userEmail == null || userEmail == "" || !validateEmail(userEmail)){
            res.status(422).json({ message: "Error: Field \"Email\" is not
valid." });
        }
        else{
            //Log in with email
            DB_CONNECTION.then(() => {
                const authDB = DB_CLIENT.db("auth");
                const usersColl = authDB.collection('users')

                usersColl.findOne( { email: escape(userEmail) }, (error: any,
result: any) => {
                    if(error){
                        res.status(500).json({ message: "Error: Could not log
in. Please try again later." });
                    }
                    else if(result == null || result == undefined || result ==
"" || result._id == null || result._id == undefined || result._id == "" || re-
sult.name == null || result.name == undefined || result.name == "" || re-
sult.email == null || result.email == undefined || result.email == "" || re-
sult.hash == null || result.hash == undefined || result.hash == "" || re-
sult.permissions == null || result.permissions == undefined){
                        res.status(422).json({ message: "Error: Incorrect
email or password." });
                    }
                    else{
                        var user = new User(result.name, result.email, re-
sult.hash, result.permissions);
                        user.id = result._id;
                        bcrypt.compare(userPassword, user.hash, function(err:
any, bcResult: boolean){
                            if(err){
```

```
                                res.status(500).json({ message: "Error: Could
not log in. Please try again later." });
                            }
                            else{
                                if(bcResult){
                                    let token = jwt.sign({ user }, pro-
cess.env.JWTSECRET, { algorithm: 'HS256', expiresIn: '1h' });
                                    res.status(200).json({ message: "Success-
fully logged in.", token });
                                }
                                else{
                                    res.status(422).json({ message: "Error:
Incorrect email or password." });
                                }
                            }
                        });
                    }
                });
            });
        }
    }
});

app.get("/checkpermission", (req, res, next) => {
    let token = <string>req?.query?.SessionToken;
    let permission = <string>req?.query?.Permission;

    if(permission == null || permission == ""){
        res.status(422).json({ message: "Error: Field \"Permission\" is re-
quired." });
    }
    else{
        let jwtValid = isTokenValid(token);
        if(!jwtValid.success){
            res.status(422).json({ message: "Error: Field \"SessionToken\" is
invalid. The current session might have expired." });
        }
        else{
            DB_CONNECTION.then(() => {
                const authDB = DB_CLIENT.db("auth");
                const usersColl = authDB.collection('users')
                usersColl.findOne( { name: jwtValid.user.name }, (error: any,
result: any) => {
                    if(error){
                        res.status(500).json({ message: "Error: Could not
check permission. Please try again later." });
                    }
```

```
                    else if(result == null || result == undefined || result ==
"" || result._id == null || result._id == undefined || result._id == "" || re-
sult.name == null || result.name == undefined || result.name == "" || re-
sult.email == null || result.email == undefined || result.email == "" || re-
sult.hash == null || result.hash == undefined || result.hash == "" || re-
sult.permissions == null || result.permissions == undefined){
                        res.status(500).json({ message: "Error: Could not
check permission. Please try again later." });
                    }
                    else{
                        var user = new User(result.name, result.email, re-
sult.hash, result.permissions);
                        user.id = result._id;

                        var hasPermission = false;
                        for (let p of user.permissions) {
                            if(p[permission] != null && p[permission]){
                                hasPermission = true;
                            }
                        }

                        res.status(200).json({ message: "Permission status of
permission \"" + permission + "\" is: " + hasPermission , hasPermission });
                    }
                });
            });
        }
    }
});

function escape(message: string){
    if(message == null && message == undefined){
        return "";
    }
    message = message.toString().replace(/</g, "&lt;").replace(/>/g,
"&gt;").replace(/"/g, "&#34;").replace(/'/g, "&#39;").replace(/`/g,
"&#96;").replace(/\(/g, "&#40;").replace(/\)/g, "&#41;").replace(/\//g,
"&#47;").replace(/\\/g, "&#92;").replace(/\[/g, "&#91;").replace(/\]/g,
"&#93;").replace(/\{/g, "&#123;").replace(/\}/g, "&#125;").replace(/\|/g,
"&#124;").replace(/\~/g, "&#126;");
    return message.trim();
}

function isTokenValid(token: string): any{
    var message = null;
    jwt.verify(token, process.env.JWTSECRET, (err: any, payload: any) => {
        if(err){
            message = { success: false };
```

```
        }else{
            message = { success: true, user: payload.user };
        }
    });
    return message;
}

function validateEmail(email: string)
{
    return /^(\w+([\.-]?\w+)*)@(\w+([\.-]?\w+)*(\.\w{2,4})+)$/.test(email);
}

app.listen(port, function(){
    console.log(`Server listening at *:${port}`);
});

process.on('SIGINT', signal => {
    console.log(`Process has been manually interrupted`);
    process.exit(0);
});

process.on('exit', exitCode => {
    console.log(`Process exited with code ${exitCode}`);
});
```

17.2 User.ts

```
import { Role } from './Role.js';

class User{
    public id: any
    public name: string
    public email: string
    public hash: (string | null)
    public permissions: []

    constructor(name: string, email: string, hash: (string | null), permis-
sions: []){
        this.name = name;
        this.email = email;
        this.hash = hash;
        this.permissions = permissions;
    }

    setPermissionFromRole(role: Role): void{
        this.permissions = role.permissions;
    }
    setOrAddPermission(permissionName: string, value: boolean): void{
        //TODO
    }
    removePermission(permissionName: string): void{
        //TODO
    }
}
export { User }
```

17.3 Role.ts

```
class Role{
    public id: any
    public name: string
    public permissions: []

    constructor(roleJson: any){
        if(roleJson != undefined && roleJson != null && roleJson._id != unde-
fined && roleJson._id != null && roleJson.name != undefined && roleJson.name
!= null && roleJson.permissions != undefined && roleJson.permissions != null){
            this.id = roleJson._id;
            this.name = roleJson.name;
            this.permissions = roleJson.permissions;
        }
    }

    setOrAddPermission(permissionName: string, value: boolean): void{
        //TODO
    }
    removePermission(permissionName: string): void{
        //TODO
    }
}
export { Role }
```

ANHANG 2

18 Dokumente

Es gibt keine Dokumente in diesem Projekt.