Alexander Ernst

Projektdokumentation Jetstream Skiservice API mongoDB

Inhaltsverzeichnis

[Ausgangslage 2](#_Toc125839424)

[Anforderungen 2](#_Toc125839425)

[Zeitplanung/PSP 3](#_Toc125839426)

[Vorgehensweise 4](#_Toc125839427)

[1. Informieren 4](#_Toc125839428)

[1.1. Ausgangslage/Anforderungen 4](#_Toc125839429)

[1.2. Informieren über Web API 4](#_Toc125839430)

[2. Planen 4](#_Toc125839431)

[2.1. Datenbank Aufbau 4](#_Toc125839432)

[2.2. Web API Aufbau 4](#_Toc125839433)

[2.3. Zeitplanung und PSP 4](#_Toc125839434)

[3. Entscheiden 5](#_Toc125839435)

[3.1. Für Versionierungsplattform/Tools entscheiden 5](#_Toc125839436)

[3.2. Für Datenbankschema entscheiden 5](#_Toc125839437)

[3.2.1. Skripte für das Migrieren von Daten 6](#_Toc125839438)

[3.2.2. Skripte für das Backup/Restore 6](#_Toc125839439)

[3.2.3. Berechtigungskonzept 6](#_Toc125839440)

[3.3. Für Web API Aufbau entscheiden 6](#_Toc125839441)

[3.4. Für Zusatzanforderungen entscheiden 7](#_Toc125839442)

[4. Realisieren 7](#_Toc125839443)

[4.1. Datenbank mit Schema und Index erstellen 7](#_Toc125839444)

[4.2. Modelle erstellen 7](#_Toc125839445)

[4.3. Controller/Services mit HTTP Methoden erstellen 7](#_Toc125839446)

[4.4. Zusatzfeatures erstellen 7](#_Toc125839447)

[4.5. Authentifikation erstellen 7](#_Toc125839448)

[4.6. Migrierung/Backup/Restore erstellen 7](#_Toc125839449)

[5. Kontrollieren 7](#_Toc125839450)

[5.1. Web API mit Postman und WPF testen 7](#_Toc125839451)

[6. Auswerten 7](#_Toc125839452)

[6.1. Reflexion/Fazit zu Projekt 7](#_Toc125839453)

[6.2. Dokumentation fertigstellen/Präsentation erstellen 7](#_Toc125839454)

# Ausgangslage

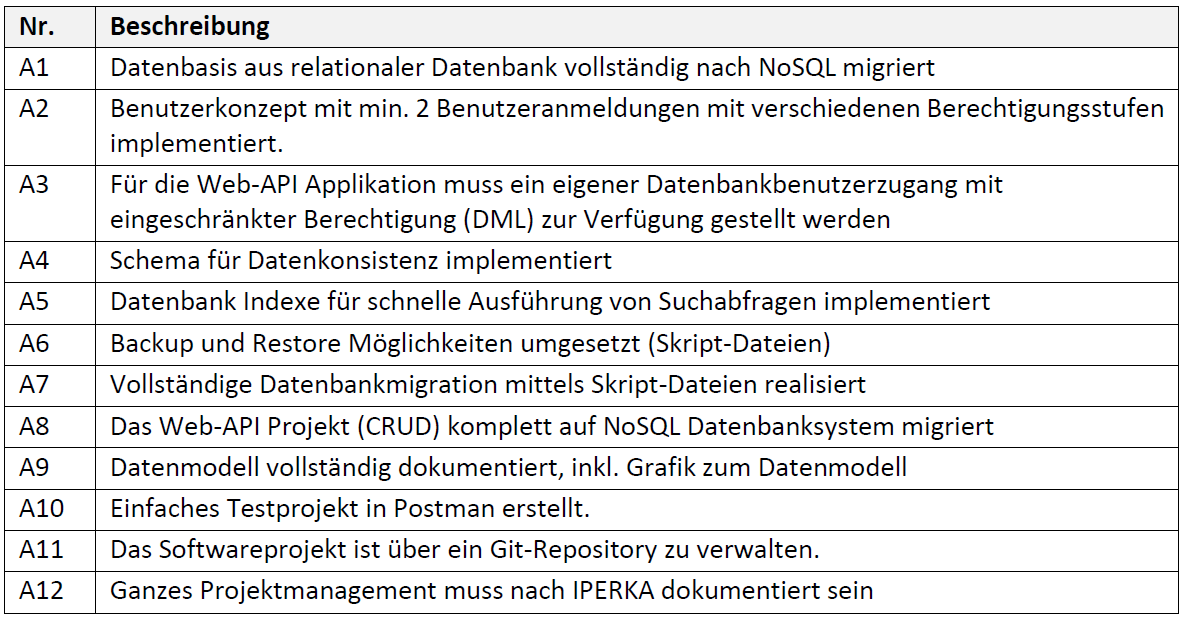
Die Firma Jetstream-Service führt als KMU in der Wintersaison Skiservicearbeiten durch und hat in den letzten Jahren grosse Investitionen in eine durchgängige digitale Auftragsanmeldung und Verwaltung, bestehend aus einer datenbankbasierender Web-Anmeldung und Auftragsverwaltung getätigt.

Aufgrund guter Auftragslage hat sich die Geschäftsführung für eine Diversifizierung mit Neu-eröffnungen an verschiedenen Standorten entschieden.

Die bis anhin eingesetzte relationale Datenbank genügt den damit verbundenen Ansprüchen an Datenverteilung und Skalierung nicht mehr. Um einerseits den neuen Anforderungen gerecht zu werden sowie anderseits Lizenzkosten einzusparen, soll im Backend der Anwendung die Datenbank auf ein NoSQL Datenbanksystem migriert werden.

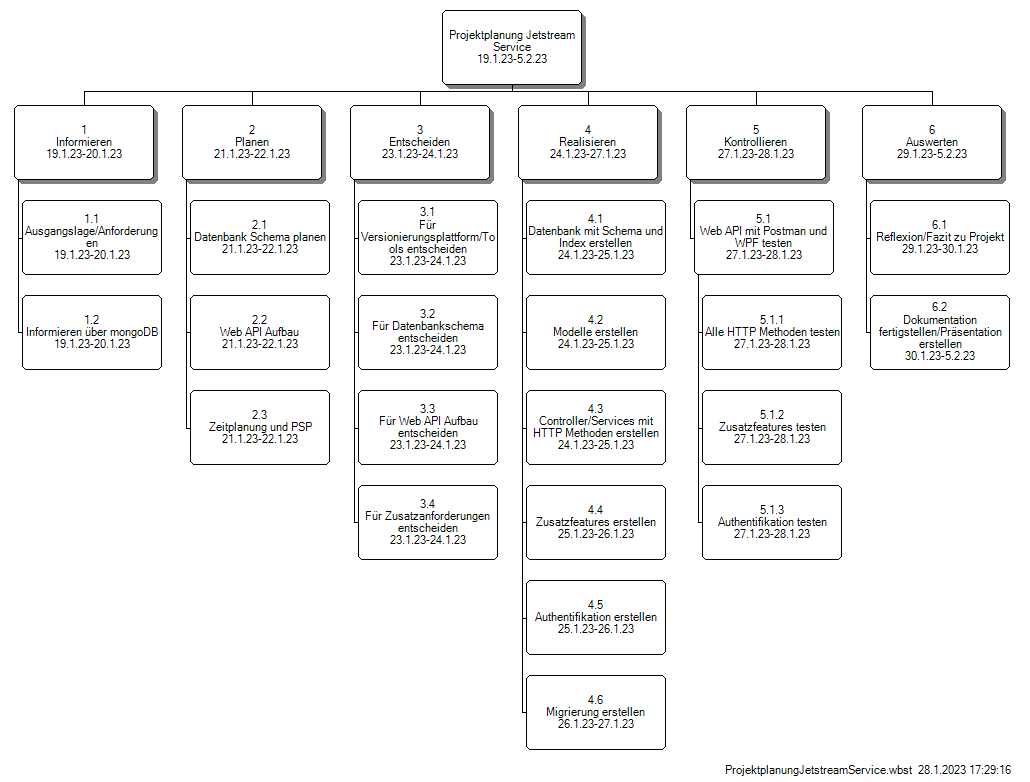
# Anforderungen

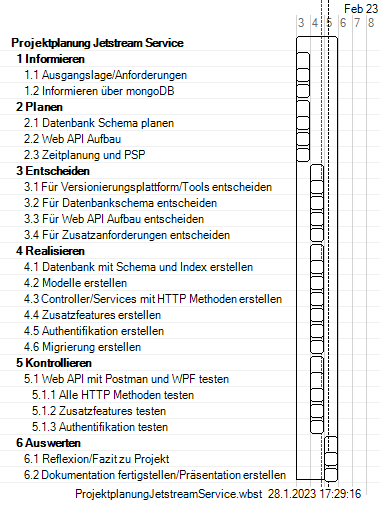
* Das Auftragsmanagement muss folgende Funktionen zur Verfügung stellen:
* Login mit Benutzername und Passwort
* Anstehende Serviceaufträge anzeigen (Liste)
* Bestehende Serviceaufträge mutieren. Dazu stehen folgende Stati zu Verfügung: Offen, InArbeit und abgeschlossen
* Aufträge löschen (ggf. bei Stornierung)



# Zeitplanung/PSP

Mit WBSTool erstellt





# Vorgehensweise

## Informieren

### Ausgangslage/Anforderungen

Als ersten Arbeitsschritt habe ich mich über die Anforderungen und die Ausgangslage dieses Projektes informiert. Hierzu habe ich die zur Verfügung gestellten Unterlagen verwendet.

Durch diese Recherche konnte ich die Anforderungen und das Grundkonzept der Web-APIs verstehen, und wusste ungefähr, was zu tun war.

### Informieren über Web API

Als Nächstes habe ich mich über Technologien, die ich für dieses Projekt brauchen werde, informiert. Zu diesen gehört einerseits die MongoDB Datenbank aber auch Faktoren wie die C# MongoDB Driver.

Diese Recherche habe ich einerseits mit dem Unterrichtstoff durchgeführt. Andererseits habe ich aber auch mich im Internet weiter informiert.

## Planen

### Datenbank Aufbau

Hier habe ich den Aufbau/das Schema der Datenbank geplant. Ich habe mir einerseits überlegt ein Schema zu erstellen welches Imbedded JSON Dokumente enthält andererseits aber auch ein simpleres Schema ohne Embedding oder Referenzen.

Zusätzlich habe ich hier auch das Berechtigungskonzept und Validierungsmöglichkeiten durch Pflichtfelder, Enums und Minium/Maximum Werte geplant.

In diesem Schritt musste ich auch Backup und Restore Möglichkeiten planen.

### Web API Aufbau

In diesem Schritt habe ich geplant, wie ich, dass Web API aufbauen will, hierzu gehören die DTO Klassen, welche Methoden ich implementiere, und welche Zusatzfeatures ich noch hinzufüge.

Vieles dieser Anforderungen waren mir auch schon vorgegeben da wir schon ein API mit einer Relationalen Datenbank haben und ich dort schon die Entscheidungen zu Authentifikation/Aufbau getroffen habe.

### Zeitplanung und PSP

Als Letztes habe ich eine Zeitplanung/ein PSP erstellt, da ich jetzt eine grobe Planung meines Projektes habe.

## Entscheiden

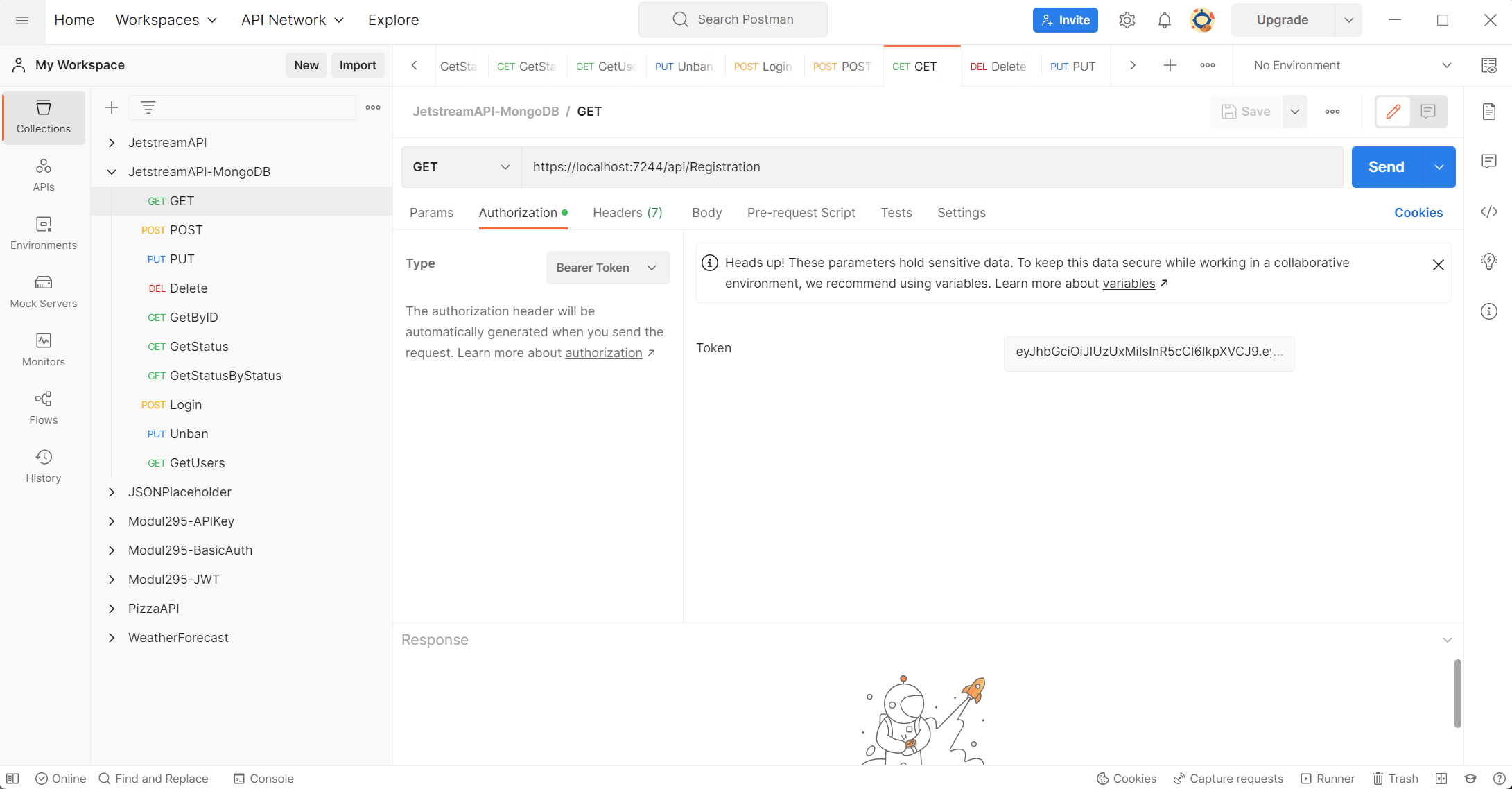
### Für Versionierungsplattform/Tools entscheiden

Hier musste ich mich entscheiden, welche Versionierungsplattform/andere Tools ich verwende. Für das Versionieren des Codes habe ich GitHub verwendet, und für das Schreiben des Codes habe ich den IDE Visual Studio verwendet.

Als Testing Tool für die API habe ich Postman gebraucht, da dies eine grosse Funktionalität bietet und für einfaches exportieren von Tests erlaubt. Zusätzlich habe ich auch die WPF Applikation verwendet da dies ist was die Kunden der Applikation verwenden werden.

Zusätzlich habe ich mich auch noch entschieden einen Logger in das Projekt einzubauen, der Fehler in einer Logdatei loggt.

GitHub: <https://github.com/alexanderternst/JetstreamSkiserviceAPI>



### Für Datenbankschema entscheiden

In diesem Schritt musste ich mich für den Aufbau/das Schema der Datenbank entscheiden.

Hier habe ich mich entschieden zwei Collections zu erstellen eine für Registrationen und eine für Benutzer. In diesen Collections habe ich keine Embeddede Dokumente oder Referenzen auf andere Dokumente.

Für dies habe ich mich bewusst entschieden da dieses API keine Embedded/Referenzierte Dokumente braucht, und dies das ganze API viel komplexer machen würde. Dies würde zu einer unnötigen Komplexität führen die das API gar nicht braucht.

Ich habe mich auch dafür entschieden eine Schemavalidierung hier habe ich eine kleine Übersicht über die Validierungen, die eingesetzt wurden:

**Registration Collection:**

* Name, Priorität, Status, Dienstleistung sind Pflichtfelder
* Eigenschaften müssen sinnvollen Datentyp haben
* Status, Priorität, Dienstleistung dürfen nur Bestimmte Werte haben:
  + Status: "Offen", "InArbeit", "abgeschlossen"
  + Priorität: "Express", "Standard", "Tief"
  + Dienstleistung: "Kleiner-Service", "Grosser-Service", "Rennski-Service", "Bindung-montieren-und-einstellen", "Fell-zuschneiden", "Heisswachsen"

**User Collection:**

* Username, Password, Counter, Rolle sind Pflichtfelder
* Eigenschaften müssen sinnvollen Datentyp haben
* Counter hat einen Minium Wert von 0 und einen Maximalen Wert von 3
* Eigenschaft Rolle darf nur die Werte, «level1», «level2» oder «level 3 haben.

Dies Validierungen habe ich einerseits eingesetzt damit Daten nicht im falschen Format eingegeben werde und damit Pflichtfelder nicht leergelassen werden und nichtexistierende Status nicht eingetragen werden.

### Skripte für das Migrieren von Daten

Zusätzlich habe ich mich auch entschieden Skripte für das Migrieren von Daten von SQL auf MongoDB in Form eines SQL und MongoDB Skriptes zu erstellen.

### Skripte für das Backup/Restore

Um Backup und Restore Möglichkeiten durchzusetzen habe ich zwei PowerShell Skripte erstellt, einen für das Backup und einen für die Restore Möglichkeit.

Damit diese Skripte funktionieren muss der bin Folder der MongoDB Tools den Environment Variables hinzugefügt werden da wir hier die Tools mongodump und mongorestore verwenden.

Diese beiden Skripte fragen den Benutzer nach Angaben wie Datenbank und Dateipfad und führen dann Backup/Restore durch.

### Berechtigungskonzept

Bei dem Berechtigungskonzept habe ich mich entschieden zwei Datenbankbenutzer zu erstellen, einer dieser Nutzer hat lese und schreib rechte, einer der Nutzer hat nur leserechte.

Dies habe ich so getan, da man vielleicht auch mal dem Kunden den Datenbankzugriff will geben, der Kunde soll aber die Datenbank nicht modifizieren können.

Um diese Benutzer zu erstellen habe ich auch einen MongoDB Skript, das Login durch den Nutzer kann man entweder nach der Verbindung oder während der Verbindung herstellen, mehr Informationen hierzu finden Sie auch in der Skript Datei.

Zusätzlich habe ich auch eine Authentifikation durch ein JWT Token erstellt welches man durch ein Login mit Benutzer und Passwort abrufen kann. Mit diesem JWT-Token hat man Zugang auf alle CRUD Befehle, ohne dieses kann man nur Registrationen erstellen.

Auf wieso ich das JWT Token gewählt habe werde ich hier nicht mehr eingehen, dies finden Sie in der Dokumentation zum anderen API.

Der Login durch das JWT-Token existiert da es vielleicht auch Mitarbeiter bei Jetstream Service gibt die nur Aufträge erstellen können, und diese nicht modifizieren/lesen können.

### Für Web API Aufbau entscheiden

Wie schon erwähnt hatte ich mich in dem letzten Projekt bereits für Projektaufbau und Authentifikation entschieden. Hier ist nochmals eine Übersicht der wichtigsten Punkte.

* Projekt mit Modellen, Controllern und Services umgesetzt für Wiederverwendbarkeit und Lesbarkeit
* Dependency Injection bei Services verwendet für Wiederverwendbarkeit
* JWT-Token für Authentifikation da dies das Sicherste und das geeignete für einen Benutzerlogin ist.
* 3 Controller (mit zugehörigen Services) für Status, Registrationen, Benutzer

### Für Zusatzanforderungen entscheiden

Hier habe ich mich entschieden, welche Zusätze ich durchsetzen wollte.

Ich habe mich hier, für das Bannen von Benutzern nach 3 falschen versuchen (auch entsprechende Entbannung Funktion), und den Kommentar bei Registrationen entschieden. Zusätzlich habe ich auch noch eingebaut, dass man über Status auf Registrationen zugreifen kann.

## Realisieren

### Datenbank mit Schema und Index erstellen

### Modelle erstellen

### Controller/Services mit HTTP Methoden erstellen

### Zusatzfeatures erstellen

### Authentifikation erstellen

### Migrierung/Backup/Restore erstellen

## Kontrollieren

### Web API mit Postman und WPF testen

## Auswerten

### Reflexion/Fazit zu Projekt

#### Was habe ich nicht geschafft/Verbesserungen

### Dokumentation fertigstellen/Präsentation erstellen