Projektarbeit

Thema: Überbetrieblicher Kurs 223

Dokumentinformationen

Dateiname: uek_223_dokumentation-detken-aeberli.docx
Speicherdatum: 03.03.2022

Autoreninformationen

Autor: Anton Detken
E-Mail: anton@detken.ch
Tel: +41 77 474 12 33

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

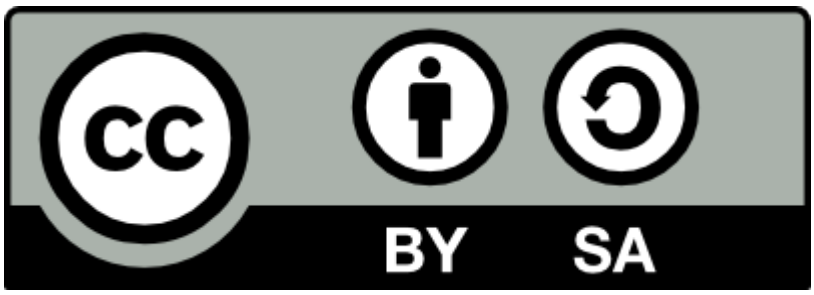
Änderungsgeschichte

Version	Datum	Autor	Details
1.0	23.02.2022	Detken	Dokument erstellt

Tabelle 1 Versionen

Lizenz

Creative Commons License



1 Einleitung

1.1 Ehrenwörtliche Erklärung

Ich bestätige:

- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.
- Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

1.2 Projekthintergrund

Dieses Projekt ist im Rahmen des überbetrieblichen Kurses M223 «Multi-User-Applikationen objektorientiert realisieren» entstanden.

1.3 Danksagung

Vielen Dank an **Luis Lüscher** für die Bereitstellung einer Dokumentations-Struktur an welcher diese Dokumentation inspiriert ist und für die Tipps bezüglich Dokumentation und Projektplanung.

Vielen Dank an **Luca Widmer** für die Einführung in die Themen des üKs.

1.4 Darstellung und Aufbau

Als Rechtschreibehilfe wurde die integrierte Überprüfungsfunktion von Microsoft Word verwendet. Ausserdem wurde die Dokumentation von verschiedenen Personen auf die Rechtschreibung überprüft.

1.4.1 Textformatierung

Es wird unter verschiedenen Textsorten unterschieden. Dafür wurde die Formatierung selbst definiert.

`Code`	Programmcode wird mit einem grauen Hintergrund und der Schriftart «Courier New» dargestellt.
`Inline-Code`	Programmcode, welcher in einer Zeile erwähnt wird, wird mit der Schriftart «Courier New» dargestellt.
Text	Texte, welche besonders zu beachten sind, werden **fett** hervorgehoben.
Text	Texte, welche besonders zu beachten sind, und Zitate werden *kursiv* dargestellt.
URL	Links werden unterstrichen.

1.4.2 Abbildung

Abbildungen sind nummeriert und mit einer Beschreibung versehen.

Abbildung 1 Beispiel

1.4.3 Tabelle

Tabellen sind nummeriert und mit einer Beschreibung versehen. Spalten- oder Zeilentitel sind **fett** hervorgehoben.

Titel 1	Titel 2
Lorem	Lorem
Ipsum	Ipsum
dolor sit amet	dolor sit amet

Tabelle 2 Beispiel

1.5 Sinn und Zweck

Dieses Dokument beschreibt das Projekt im üK.

1.6 Referenzdokumente

[1] ProjektAufgabe_2022_v4.pdf

1.7 Abkürzungen

Abkürzung	Beschreibung
DB	**D**aten**b**ank / **D**ata**b**ase
üK / ÜK	**ü**berbetrieblicher **K**urs

Tabelle 3 Abkürzungen

1.8 Modulidentifikation

Modulnummer 223

Titel Multi-User-Applikationen objektorientiert realisieren.

Kompetenz Multi-User-Applikation objektorientiert entwerfen, erforderliche Datenbankanpassungen vornehmen und Applikation implementieren, testen und dokumentieren.

1.8.1 Handlungsziele

1. Einschätzen, ob eine Datenbank die Anforderungen der Multi-User-Fähigkeit erfüllt und allfällige Anpassungen dokumentieren.
2. Applikationen entwerfen und mittels Transaktionen Multi-User-Fähigkeit sicherstellen.
3. User Interfaces, Datenbankanpassungen und Transaktionen implementieren.
4. Testspezifikation für funktionale und nicht-funktionale Aspekte der Multi-User-Fähigkeit definieren, Applikation testen und Tests protokollieren.
5. Transaktionen dokumentieren und dabei auf Wartbarkeit und Nachvollziehbarkeit achten.

1.8.2 Handlungsnotwendige Kenntnisse

1. Kennt Anforderungen an das Datenbankmanagement-System bezüglich Multi-User-Fähigkeit.
2. Kennt Aspekte bei der Datenmodellierung, welche die Multi-User-Fähigkeit ermöglichen.
3. Kennt die prinzipiellen Unterschiede zwischen Geschäftsobjektmodell und relationalem Datenmodell.
4. Kennt wichtige Architekturvarianten und -konzepte (Client/Server, Multi-Tier, Middleware, Framework, Klassenbibliothek).
5. Kennt die Umsetzung einer objekt-relationalen Abbildung eines Geschäftsobjektmodells und deren Spezifikation mittels UML (Klassendiagramm, Sequenzdiagramm).
6. Kennt spezifische Elemente für die Umsetzung von Multi-User-fähigen Benutzerschnittstellen (z.B. Profil, unterschiedliche Benutzersichten, Berechtigungskonzept, usw.).
7. Kennt Möglichkeiten ein mehrbenutzer-fähiges Rechtemanagement zu implementieren.
8. Kennt Möglichkeiten, um Transaktionen im DBMS sicherzustellen.
9. Kennt Möglichkeiten, um Transaktionen in der Applikation zu implementieren.
10. Kennt relevante Techniken für die Implementation einer Persistenzschicht.
11. Kennt relevante Aspekte, welche bei der Testspezifikation einer Multi-User-Applikation zu berücksichtigen sind.
12. Kennt ein Vorgehen, um nicht-funktionale Anforderungen zu testen.
13. Kennt Möglichkeiten zur Dokumentation von Transaktionen um DBMS und in der Applikation.

1.8.3 Leistungsbeurteilungsvorgaben

Institution Noser Young

Übersicht Zweiteilige LBV; Erstes Element: Projekt / Zweites Element: Fachgespräch

Der momentane Teil ist hellblau markiert, bereits absolvierte Teile sind grün markiert, noch nicht absolvierte Teile sind weiss markiert.

Teil	**1**
Gewichtung	60%
Richtzeit (Empfehlung)	~40h
Element-Beschreibung	Projekt
Hilfsmittel	Keine Limitationen
Bewertung	
Praxisbezug	Bildungsplan Applikationsentwicklung (2014)

Tabelle 4 Leistungsbeurteilungsvorgaben (Teil 1)

Teil	**2**
Gewichtung	40%
Richtzeit (Empfehlung)	~15min
Element-Beschreibung	Fachgespräch
Hilfsmittel	Keine Hilfsmittel erlaubt
Bewertung	
Praxisbezug	Bildungsplan Applikationsentwicklung (2014)

Tabelle 5 Leistungsbeurteilungsvorgaben (Teil 2)

2 Umfeld und Ablauf

2.1 Projektumfeld

Blog-Website, auf der mehrere User Blog-Posts erstellen können. Die Posts sind zugänglich für die Öffentlichkeit. Diese Page wird verwaltet von Admins, die Kategorien erstellen und verwalten können. Diese Kategorien können dann von Usern zu Blog-Posts zugeteilt werden. User können Teil einer Gruppe werden.

2.2 Aufgabenstellung

2.2.1 Titel der Arbeit

Originaltext gemäss ProjektAufgabe_2022_v4.pdf

Projekt-Auftrag ÜK 223 2022

2.2.2 Ausgangslage

In einer Gruppenarbeit müssen verschiedene Kompetenzen und Themen bearbeitet werden. In unserem Fall sind in unserer Gruppe

- Detken, Anton
- Aeberli, Remo

Die erarbeiteten Resultate werden in einer Dokumentation abgelegt.

Originaltext gemäss ProjektAufgabe_2022_v4.pdf

Ziel ist es, ein Backend für eine Blog-Website zu erstellen, auf der mehrere User Blog-Posts erstellen können. Die Posts sind zugänglich für die Öffentlichkeit. Diese Page wird verwaltet von Admins, die Kategorien erstellen und verwalten können. Diese Kategorien können dann von Usern zu Blog-Posts zugeteilt werden. User können Teil einer Gruppe werden.

2.2.3 Detaillierte Aufgabenstellung

Originaltext gemäss ProjektAufgabe_2022_v4.pdf

Ziel dieses Projekt ist es sein ein Backend mithilfe von SpringBoot und PostgreSQL zu erstellen, die die unten aufgeführten Bedingungen erfüllt. Dabei wird auf Multiuser-fähigkeit und Sicherheit geachtet. Als Vorgabe erhalten sie ein Skelet, indem die Grundlagen eines Spring Projekts erstellt wurde. Dieses Projekt enthält Funktionalitäten, um bestehende User einzuloggen und ermöglicht das manuelle Erstellen von Usern, Rollen und Autoritäten.

2.2.4 Themenübersicht

- Spring Boot Stuktur
- User Rollen & Privilegien
- Security

- User Management

2.2.5 Mittel und Methoden

Für diese Arbeit wird IPERKA als Vorgehensweise verwendet. Die Inhalte werden via Recherchen und bereits vorhandenen Fachkenntnissen erarbeitet und so aufbereitet, dass diese gut anschaulich dokumentiert werden können.

2.2.6 Vorkenntnisse

Die Projektmitglieder verfügen über Fachkenntnisse in der Softwareentwicklung und kennen einige der zu behandelnden Themen bereits aus dem Basislehrjahr. Die zur erarbeitenden Themen sollte auf einer guten fachlichen Stufe erarbeitet werden.

2.2.7 Vorarbeiten

- Arbeit im Betrieb
- Erweiterter üK 307

2.2.8 Neue Lerninhalte

- Spring Boot

2.3 Projektantrag

Projekttitel	Projekt-Auftrag ÜK 223 2022	
Projektnummer	00001	
Projektart	Gruppenarbeit	
Projektleiter/in	Detken, Anton	
Projektmitglieder	Detken, Anton Aeberli, Remo	
Projektauftraggeber/in	Widmer, Luca	
Projektkunde(n)	Noser Young AG	
Projektdauer	**Geplanter Beginn**	23.02.2022 08:15 Uhr
	Geplantes Ende	03.03.2022 17:00 Uhr
Ausgangssituation / Problembeschreibung	Siehe *2. Umfeld und Ablauf*	
Projektgesamtziel	Spring-Boot-Backend	
Projektressourcen	**Ressource**	**Menge**
	Personal	1
	Laptops	1
	Arbeitsplätze	1
Projektbudget	Das Projektbudget beläuft sich auf CHF 0.	
Sonstige relevante Informationen	-	

Tabelle 6 Projektantrag

2.4 Arbeitsumfeld

In diesem Kapitel wird das Arbeitsumfeld beschrieben. Das beinhaltet den Arbeitsplatz, die Hard- und Software und die Organisation der Dokumentablage.

2.4.1 Arbeitsplatz

Zur Arbeit ist vorausgesetzt, dass eine stabile Internetverbindung und ein Stromanschluss vorhanden sind.

2.4.2 Hardware

Server: Ein Entwicklungsserver steht unter anton.bz zur Verfügung. Der Server läuft auf Linux Ubuntu 18.04 und ist mit 20GB RAM, 4 Xeon Cores und 100GB NVMe Speicher ausgerüstet. Folgende Umgebungen sind auf dem Server vorhanden: Apache Webserver, PHP, MySQL/MariaDB, WordPress.

Laptops: Jeder Mitarbeiter nutzt seinen privaten Windows oder MacOS Laptop.

2.4.3 Software

IntelliJ: IntelliJ IDEA oder darauf basierende Entwicklungsumgebungen werden zur Entwicklung von ausführbarem Programmcode (i.e: Java) verwendet.

Visual Studio Code: Visual Studio Code (auch VS Code) wird zur Entwicklung von Web-Applikationen, welche auf HTML basieren, verwendet. Wird HTML in Verbindung mit Java oder PHP verwendet, wird auf IntelliJ oder PhpStorm ausgewichen.

Docker: Für virtuelle Umgebungen wird gegebenenfalls Docker verwendet, falls keine Umgebung auf dem Entwicklungsserver anton.bz vorhanden ist.

Microsoft Office: Word wird zur Dokumentation verwendet. PowerPoint wird zur Präsentation verwendet.

Adobe Acrobat: Zur Verwaltung von PDF-Dokumenten.

2.4.4 Dokumentablage

OneDrive: Dokumentationsdateien und andere Unterlagen werden in OneDrive in einem Projektordner abgelegt.

GitHub: Programmcode wird auf GitHub gespeichert.

MySQL: Daten werden in MySQL persistiert. Dieser MySQL-Server läuft auf dem Entwicklungsserver von Anton Detken, welcher unter anton.bz erreichbar ist.

2.5 Namenskonventionen

2.5.1 Definition Code-Case-Typen

camelCase

camelCase beginnt mit einem Kleinbuchstaben. Der erste Buchstabe jedes neuen nachfolgenden Wortes wird grossgeschrieben und mit dem vorherigen Wort zusammengesetzt.

Beispiel: `camel case var` wird zu `camelCaseVar`.

snake_case

Im snake_case ersetzt man alle Leerzeichen mit einem "_„. Alle Wörter sind kleingeschrieben.

Beispiel: `snake case var` wird zu `snake_case_var`.

kebab-case

Im kebab-case ersetzt man alle Leerzeichen mit einem “-“. Alle Wörter sind kleingeschrieben.

Beispiel: `kebab case var` wird zu `kebab-case-var`.

PascalCase

Im PascalCase werden alle Wörter ohne Leerzeichen zusammengehängt. Der erste Buchstabe jedes Wortes ist grossgeschrieben.

Beispiel: `pascal case var` wird zu `PascalCaseVar`.

UPPER_CASE_SNAKE_CASE

Dieser Case Typ funktioniert exakt gleich wie der snake_case, nur dass alle Buchstaben grossgeschrieben werden.

Beispiel: `upper case snake case var` wird zu `UPPER_CASE_SNAKE_CASE_VAR`.

2.5.2 Code-Case-Typen per Sprache

Case Typ	Java	JavaScript	Python	HTML/PHP
camelCase	Felder	Variablen, Funktionen		
snake_case			Variablen, Funktionen, Module	Variablen, Funktionen
UPPER_CASE_ SNAKE_CASE	Konstanten	Konstanten	Konstanten	Konstanten
kebab-case				HTML-Klassen
PascalCase	Klassen, Interfaces, Enums	Klassen, Typen	Klassen	PHP-Klassen

Tabelle 7 Code-Case-Typen per Sprache

2.5.3 Dateien

Bei der Namensgebung für Dateien wird snake_case mit kebab-case gemischt. Kohärente Wörter werden mit snake_case verbunden, während zwischen diesen kebab-case verwendet wird.

```
dokumentation-detken_anton-muster_max.pdf
```

Bei Dateien, bei welchen das Datum relevant ist, herrscht folgende Konvention:

```
YYYYMMDD-dokument_name.pdf
```

3 Projektmanagement

3.1 IPERKA

Für dieses Projekt wird nach dem Vorgehensmodell IPERKA vorgegangen und die Planung ist entsprechend dem Modell aufgebaut. Dies spiegelt sich auch in der Dokumentation wider.

IPERKA wurde bereits in einigen Projekten eingesetzt und hat sich für solche Arbeiten bewährt.

Bei IPERKA beschreibt jeder Buchstabe des Namens einen Projektabschnitt:

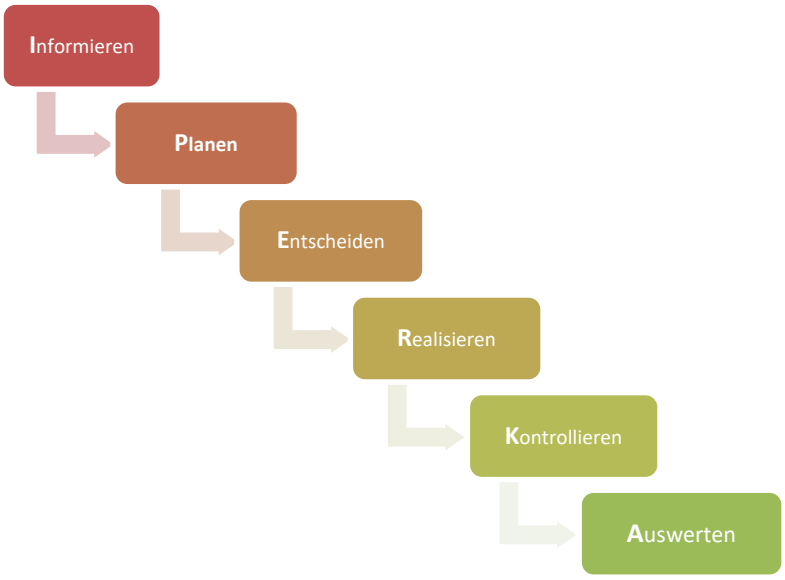


Abbildung 2 IPERKA (Quelle: Luis Lüscher)

3.1.1 Informieren

Beim Informieren werden die Informationen abgeholt, die für die Durchführung des Projekts benötigt werden. Damit wird ein klares Bild des Auftrages geschaffen und erste Fragen werden bereits geklärt.

Am Ende dieser Phase sind folgende Fragen beantwortet:

- Wie lautet der genaue Auftrag?
- Was für Bedingungen muss ich erfüllen?
- Was ist das Ziel des Projekts?
- Habe ich die notwendigen Mittel, um das Projekt durchzuführen?

3.1.2 Planen

Das ganze Projekt wird geplant. Hier wird ein genauer Zeitplan erstellt, in dem definiert ist, wer was wann macht und die benötigten Ressourcen werden definiert. In diesem Schritt wird klar, *wie* das Projekt durchgeführt wird.

Am Ende dieser Phase sind folgende Fragen beantwortet:

- *Wie* wird das Projekt realisiert?
- *Was* für Ressourcen werden benötigt?
- *Was* wird *wann* erledigt?

3.1.3 Entscheiden

Im nächsten Schritt wird entschieden, welche Tools und Produkte verwendet werden sollen, um das Projekt umzusetzen. Dafür werden passende Kriterien definiert. In Frage kommende Möglichkeiten werden verglichen.

Am Ende dieser Phase sind folgende Fragen beantwortet:

- Mit *welcher* Lösung setze ich das Projekt um?
- Ist diese Lösung *sinnvoll*?
- Hat die Entscheidung eine ausschlaggebende *Begründung*?

3.1.4 Realisieren

In diesem Schritt wird das Projekt effektiv umgesetzt. Die geplanten Arbeiten zur Umsetzung werden möglichst nach Plan ausgeführt.

3.1.5 Kontrollieren

Die gesamte Arbeit wird nochmals kontrolliert. Es wird geprüft, ob das Realisierte den Anforderungen und der Planung entspricht. Um dies möglichst effektiv zu prüfen, wird ein Testprotokoll erstellt und ausgefüllt und die Arbeit wird somit auf Fehler geprüft.

Am Ende dieser Phase sind folgende Fragen beantwortet:

- Entspricht das Produkt den gestellten Anforderungen?
- Ist das Produkt vollständig getestet und fehlerlos?
- Sind alle Ziele erreicht worden?

Wurden nicht alle Anforderungen erfüllt, gibt es einen Sprung zum Realisierungsschritt, bis das Produkt final ist.

3.1.6 Auswerten

Es wird auf das ganze Projekt nochmals zurückgeschaut und es werden Erkenntnisse festgehalten. Es wird ausgearbeitet, was in zukünftigen Projekten besser gemacht werden kann. Es handelt sich also um eine Reflektion.

Am Ende dieser Phase sind folgende Fragen beantwortet:

- Was lief gut? Was lief schlecht?
- Was kann verbessert werden?
- Wie ist die Zufriedenheit mit dem Produkt?
- Was sollte nächstes Mal unbedingt anders gemacht werden?

3.2 Gantt

Die Zeitplanung dieses Projektes läuft nach dem Gantt-Modell. Im Schritt «Planen» (siehe: IPERKA) wird ein solches Gantt-Diagramm erstellt. Gantt ist im Projekt Management eine weitverbreitete Methode, um Aktivitäten zeitlich darzustellen. Auf der linken Seite ist eine Liste mit allen Aktivitäten, während rechts eine Leiste den Zeitraum dieser Aktivität angibt.

3.2.1 Beispiel

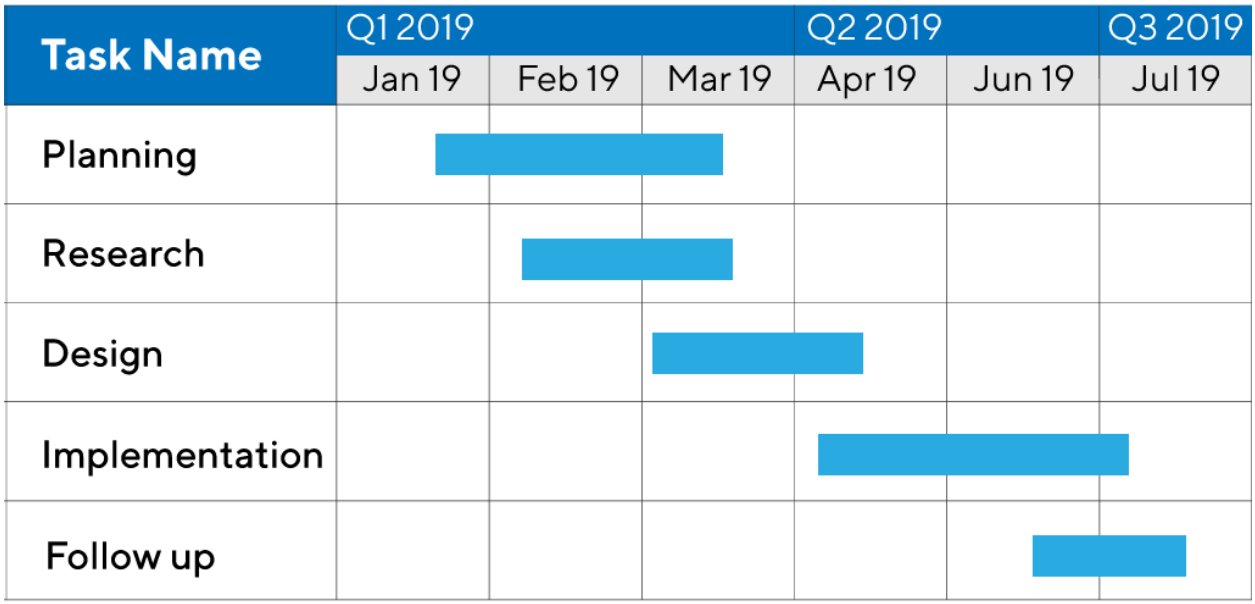


Abbildung 3 Gantt Beispiel

3.2.2 Tools

Es gibt eine Menge an Tools, um ein solches Diagramm zu erstellen. Die Beliebtesten sind Excel Templates und Click-Up.

3.3 Projektaufbauorganisation

Das Projekt ist folgendermassen organisiert:

- **Luca Widmer** ist der Auftraggeber.
- **Anton Detken** ist der Projektleiter, Software-Engineer. Die wichtigste Entscheidung muss er beim Abschnitt «Entscheiden» fällen. Dieser Schritt ist im IPERKA-Modell essenziell. Ausserdem arbeitet Anton Detken in der Umsetzung bei den Punkten Software-Engineering und Design.
- **Remo Aeberli** ist Teil des Projektteams und arbeitet als Software-Engineer mit. Seine Hauptaufgaben liegen in der Umsetzung beim Punkt Software-Engineering und bei der Kontrolle, sowie Auswertung. Zudem unterstützt er die Projektleitung bei verschiedenen Aufgaben.

3.3.1 Definition Projektleiter

Originaltext von projektmanagement-definitionen.de.

Existiert eine Gruppe oder wird ein Projekt in Angriff genommen, so nehmen die beteiligten Parteien verschiedene Rollen innerhalb des Prozesses ein. Oft sind diese Rollen im Vorhinein festgelegt, oft kristallisieren sie sich erst in der team-building-Phase heraus.

Beide Varianten haben ihre Vor- und Nachteile. So führt die Vakanz des Projektleiterpostens oftmals dazu, dass ungeahnte Ressourcen ans Licht kommen – andererseits sind von vornherein klar geregelte Zuständigkeiten potentiell von Vorteil für den Gesamtprozess. Über diesen Aspekt zu entscheiden, gehört zu den Aufgaben des Projektmanagements.

Der Projektleiter oder Projektmanager ist zuständig für die operative Planung und Steuerung eines Prozesses. Er trägt die Verantwortung für das Erreichen bestimmter Ziele. Dabei kann es sich um Sachleistungen, Termine und Kosten, i.d.R. alle drei Varianten handeln. Der Projektleiter ist es, der Ziele bzw. benötigte Ressourcen festlegt.

Ein grober faux-pas des Projektmanagements ist es, vor versammelter Mannschaft einen Projektleiter zu ernennen, der nicht recht weiß, wie ihm geschieht. Mitarbeiter, die man im Gesamtprozess in einer dermaßen verantwortungsvollen Rolle sehen will, sollten deswegen im Vorfeld zu einem persönlichen Gespräch geladen werden.

3.3.2 Definition Software Engineer

Originaltext von mein-studium-karriere.ch.

Ein Softwareentwickler – oft auch als Software Developer bezeichnet – kümmert sich um Software: Er konzeptioniert sie, realisiert die Umsetzung und wartet sie schlussendlich auch. Es gibt unterschiedliche Softwarestacks, sogenannte Lösungsstapel, die in der Regel eine langjährige Ausbildung und Erfahrung benötigen. Softwarestacks wie Apple, Android, aber auch .NET Core, XAMPP und viele andere sind möglich.

Softwareentwickler arbeiten beispielsweise im sogenannten Frontend- oder Backend-Bereich, aber auch in vielen anderen Bereichen. Einfach ausgedrückt: Der Softwareentwickler entwickelt Software, also ein Programm, eine App, eine Applikation oder irgendetwas, was auf einem Computer oder ähnlichem Gerät läuft und mit dem Menschen etwas Unterhaltsames oder etwas Sinnvolles anstellen können.

3.4 Pflichtenheft

«Das Pflichtenheft ist der von Auftragnehmer erstellte Projektplan, mit dem er das Lastenheft des Auftraggebers erfüllen möchte.»

~ Dr. Georg Angermeier

3.4.1 Pflichten Projektleiter

Stelle besetzt durch: Anton Detken

Unterstützt durch: Remo Aeberli

Folgende Pflichten innerhalb des Projekts:

- Teamorganisation
- Projektdokumentation
- Übersicht im Team
- Verlauf des Projektes bestimmen
- Aufgabenstellung lesen & verstehen
- Review & Überarbeitung der Dokumentation
- Abgabe der Produkte
- Verwaltung der GitHub Issues

3.4.2 Pflichten Software Engineer

Stelle besetzt durch: Anton Detken, Remo Aeberli

Folgende Pflichten innerhalb des Projekts:

- Aufgabenstellung lesen & verstehen
- Abarbeiten der Aufgaben gemäss Aufgabenaufteilung
- Projektstatus an Projektleiter melden
- Unterstützung der Projektleitung in verschiedenen Tätigkeiten
- Schreiben einer Reflexion
- Review der Dokumentation

3.4.3 Aufgaben Anton Detken

Nach GitHub-Issues.

3.4.4 Aufgaben Remo Aeberli

Nach GitHub-Issues.

3.5 SWOT

SWOT steht für

- **Strengths** (Stärken)
- **Weaknesses** (Schwächen)
- **Opportunities** (Chancen)
- **Threats** (Gefahren)

Die SWOT-Analyse ist ein Werkzeug des strategischen Managements, wird aber auch für die Qualitätsentwicklung von Programmen und Projekten eingesetzt. Mit dieser zugleich einfachen und flexiblen Methode können sowohl Stärken und Schwächen innerhalb des Projektes als auch

externe Chancen und Gefahren betrachtet werden. Aus dieser Kombination kann eine Strategie für die weitere Ausrichtung des Projektes abgeleitet werden.

3.5.1 Vorteile

- Schnelle Auseinandersetzung mit positiven und negativen Aspekten einer Situation.
- Projizierung dieser Situation in die Zukunft.

3.5.2 Nachteile

- Oberflächliche Ergebnisse bei fehlender Ernsthaftigkeit oder Infragestellen des Nutzens der Analyse möglich.

3.5.3 Bereiche

In diesem Abschnitt werden die einzelnen Bereiche untersucht. Es folgen passende Fragen für jeden Bereich, auf die eine Analyse bezogen werden kann.

Strengths (Stärken)

- Was zeichnet das Unternehmen aus?
- Was sind oder waren seine *grössten Erfolge*?
- Im direkten Vergleich: Was kann das Unternehmen *besser* als seine Konkurrenten?

Weaknesses (Schwächen)

- Worin ist das Unternehmen nicht gut?
- Was fehlt im Unternehmen?
- Im direkten Vergleich: Was können die Konkurrenten besser?

Opportunities (Chancen)

- Welche *positiven* Trends zeichnen sich ab?
- Welche gesellschaftlichen, wirtschaftlichen, technologischen, oder politischen Entwicklungen könnten dem Unternehmen *in die Karten spielen*?
- Welche sonstigen Rahmenbedingungen sind positiv oder ändern sich in eine positive Richtung?

Threats (Bedrohungen)

- Welche *negativen* Trends zeichnen sich ab?
- Welche gesellschaftlichen, wirtschaftlichen, technologischen, oder politischen Entwicklungen könnten dem Unternehmen direkt oder indirekt *schaden*?
- Welche sonstigen Rahmenbedingungen sind negativ oder ändern sich in eine negative Richtung?

3.5.4 Strategie

Mit der Analyse der vier Bereiche ist hat man nun zwar einen guten Überblick über die aktuelle Situation sowie anstehende Herausforderungen, aber wenn man jetzt aufhört, verpasst man einen wichtigen abschliessenden Analyseschritt. Das **eigentliche Ziel** einer SWOT Analyse ist es nämlich nicht, diese Faktoren einfach zu sammeln, sondern – darauf aufbauend – **strategische Massnahmen** zu identifizieren.

Um dies zu erreichen, müssen die Wechselwirkungen der vier Bereiche analysiert werden. Aus den unterschiedlichen Kombination entstehen wiederum vier Kategorien an strategischen Massnahmen.

SO-Strategie – Strengths & Opportunities

Welche *Stärken* können genutzt werden, um von den *Chancen* zu profitieren?

Strategien, welche aus diesem Bereich abgeleitet werden, fallen in die Kategorie «*Führungsperson ausbauen*» und sind relativ einfach durchzuführen.

WO-Strategie – Weaknesses & Opportunities

Welche *Schwächen* hindern uns daran, von den *Chancen* zu profitieren?

Strategien, welche aus diesem Bereich abgeleitet werden, fallen in die Kategorie «*Zum Wettbewerb aufholen*».

ST-Strategie – Strengths & Threats

Welche *Stärken* können genutzt werden, um von den *Bedrohungen* zu reduzieren?

Strategien, welche aus diesem Bereich abgeleitet werden, fallen in die Kategorie «*Absichern*».

WT-Strategie – Weaknesses & Threats

Welche *Schwächen* hindern uns daran, die *Bedrohungen* zu reduzieren?

Strategien, welche aus diesem Bereich abgeleitet werden, fallen in die Kategorie «*Vermeiden*».

3.5.5 SWOT Analyse

SWOT-Analyse Im Rahmen von: Projektarbeit Mxxx «[Modul-Titel]» Durchgeführt durch: Anton Detken, [Person], [Person] Datum: 19. Februar 2022		Projektanalyse	
		Stärken (Strengths) S1: Hohe Motivation der MA S2: Gute Sozialkompetenz S3: Erfahrungen in Bereich S4: Kurze Entscheidungswege S5: Hohes Selbstbewusstsein S6: Gute Dokumentation, da viel Erfahrung von Luis	**Schwächen** (Weaknesses) W1: Dokumentation wird nur durch eine Person geführt W2: Abhängig von jedem Gruppenmitglied, da sehr straffer Zeitplan.
Umweltanalyse	**Chancen** (Opportunities) O1: Bessere Lösung als andere Teams erarbeiten O2: Umfangreiche Dokumentation O3: Zeitplan wird gelockert.	Aus welchen Stärken ergeben sich neue Chancen? SO1: Erarbeitung von tollen Produkten, da alle hohe Motivation haben. (Motivation MA) SO2: Umfangreiche Dokumentation, da hohe Motivation sowie erfahrene Projektmitglieder.	Schwächen eliminieren, um neue Chancen zu nutzen WO1: Arbeit gut aufteilen, sodass Last entsprechend verteilt ist. WO2: Kommunikation gut aufrechterhalten, sodass Engpässe ausgeschlossen werden können

	Risiken (Threats) T1: Umfang der Arbeit könnte zu gross sein. T2: Einfluss anderer Teams auf unser Projekt	Welche Stärken minimieren Risiken? ST1: Sozialkompetenz ist sehr gut, Kollaboration somit kein Problem. ST2: Kurze Entscheidungswege, dadurch schnelle Entscheidungen. (Weniger Diskussionen)	Strategien, damit Schwächen nicht zu Risiken werden? WT1: Dokumentation sollte vor der Abgabe durch alle Projektmitglieder angeschaut werden.

Tabelle 8 Beispiel einer SWOT-Analyse (Layout: Luis Lüscher)

3.6 Risikoanalyse

3.6.1 Erklärung

Die Risikoanalyse ist eine Diagnose, um mögliche Probleme zu erkennen, einzudämmen und zu eliminieren. Gründe für solch eine Analyse sind die Prävention für eventuell auftauchende Probleme, die vorrausschauende Planung des Projektes und die Garantie eines reibungslosen Ablaufs.

3.6.2 Vorgehensweise

1. **Ziele nach SMART beschreiben**
   → siehe Abschnitt zum Thema «SMART»
   Die Punkte M, R, T müssen bei der Analyse vorhanden sein.
2. **Risikobereich identifizieren**
   In diesem Schritt werden Risiken gesucht, wobei alle Projektdimensionen beachtet werden müssen (Qualität, Ressourcen, …). Nicht die Symptome, sondern die *Ursache* dieser Risiken müssen unbedingt genannt werden.
3. **Symptome identifizieren**
   Anhand der Symptomen kann erkannt werden, ob das Problem bereits eingetreten ist oder in naher oder ferner Zukunft einzutreten droht.
4. **Risiken gemäss Risikomatrix einstufen**
   Jedem Risiko wird nach den Kriterien «*Wahrscheinlichkeit des Eintreffens*» und «*Tragweite*» bewertet.
5. ***Vorbeugende* Massnahmen umsetzen**
   Gegenmassnahmen zur Prävention oder Eingrenzung des Problems werden mittels Risikoanalysetabelle umgesetzt.
6. ***Alternative* Massnahmen planen**
   Bei besonders kritischen Risikobereichen werden bereits in der Planungsphase alternative Massnahmen vorgesehen, falls die vorbeugenden Massnahmen aus Schritt 5 nicht greifen.

3.6.3 Risikoanalysetabelle

Nr.	Risiko	Symptome	Wahrscheinlichkeit	Tragweite	Gegenmassnahmen
Nr. 1	Abgabetermin kann nicht eingehalten werden	- Termin werden gemäss - Zu viele Meinungen sind zu berücksichtigen	Mittel	Hoch	- Kunden auf kritische Termine hinweisen. - Zu Entscheidungen verpflichten - Meinungen nur berücksichtigen anhand Empfehlung Projektleiter (Sofort einleiten)
Nr. 2	Budget wird nicht eingehalten	- Kosten höher als Budget - Kunde hat Bedenken bei den Kosten	Mittel	Mittel	- Genügend kostengünstigere Alternativen vorbereiten - Erarbeitetes Resultat gut verkaufen, sodass Kunde keine Bedenken hat. (Sofort einleiten)
Nr. 3	Kunde kann nicht bezahlen	- Rechnungen werden nicht bezahlt	Niedrig	Hoch	- Finanzen vor Projekt abklären - Anzahlung verlangen
Nr. 4	Dokumentation geht verloren	- Dokumentation ist nicht mehr auffindbar - Dokumentation ist veraltetet	Niedrig	Mittel	- Backup erstellen (Sofort einleiten) - Regelmässig in Teams Chat hochladen.
Nr. 5	Zu wenig Quellen für Informationen	- Es sind nicht genügend Informationen in einer Quelle	Niedrig	Niedrig	-Genügende Quellen vorbereiten - Quellen mit Kunde besprechen

Tabelle 9 Beispiel einer Risikoanalyse (Quelle: Luis Lüscher)

3.6.4 Risikomatrix

		Tragweite		
		Niedrig	*Mittel*	*Hoch*
Eintrittswahrscheinlichkeit	*Niedrig*			
	Mittel			
	Hoch			

Tabelle 10 Risikomatrix

4 Informieren

Ziel ist es, während der Zeit des Kurses, ein Backend für eine Blog-Website zu erstellen. Genauere Anforderungen sind in der Datei «ProjektAufgabe_2022_v4.pdf» beschrieben.

4.1 Themen

- Java
- Spring Boot
    - Security
    - Validation
    - Datenbanken
- http-Anfragen

4.2 Formulierung des Auftrages

Originaltext aus «ProjektAufgabe_2022_v4.pdf».

Ziel dieses Projekt ist es sein ein Backend mithilfe von SpringBoot und PostgreSQL zu erstellen, die die unten aufgeführten Bedingungen erfüllt. Dabei wird auf Multiuser-fähigkeit und Sicherheit geachtet. Als Vorgabe erhalten sie ein Skelet, indem die Grundlagen eines Spring Projekts erstellt wurde. Dieses Projekt enthält Funktionalitäten, um bestehende User einzuloggen und ermöglicht das manuelle Erstellen von Usern, Rollen und Autoritäten.

5 Planen

5.1 Benötigte Infrastruktur

- Laptop
- Internetzugang
- Software
    - IntelliJ
    - Postman
    - Docker
        - Postgres-Image
    - DBeaver

5.2 Testkonzept

Tests sind unverzichtbar. Für die Tests wurde eine Vorlage für ein Testkonzept mit Erklärungen erstellt. Da in diesem Projekt **Postman** verwendet wird, kommt dieses Testkonzept nicht zum Einsatz.

Testfall-Nummer			Nummer		
Bezeichnung			Kurzer Titel des Tests		
Beschreibung / Zu testende Funktionalität			Kurzbeschrieb		
Klassifikation			TP / FP / TN / FN – *siehe Klassifikation*		
Use-Case-Nummer			Use-Cases der getestet wird		
Datum			Datum der Testdurchführung (*YYYY-MM-DD*)		
Tester					
Voraussetzung			Umgebung, welche vorausgesetzt wird		
Testumgebung					
	Software		Beschreibung der Softwareumgebung, in welcher getestet wird (i.e: Datenbank «Buchhaltung» oder Chrome 98.0.4758.102)		
	Betriebssystem		i.e: MacOS 12.0.1		
	Hardware		Falls relevant: Bildschirm, Tastatur, Maus, …		
Testschritte					
	Aktion	**Erwartetes Ergebnis**	**Effektives Ergebnis**	**Erfüllt**	**Kommentar**
1					
2					

Tabelle 11 Testkonzept

5.2.1 Erklärung Klassifikation

Es gibt verschiedene Testarten. Man unterscheidet zwischen True Positive, False Positive, True Negative, False Negative.

	True	**False**
Positive	Es wird erwartet, dass etwas *funktioniert* und es *funktioniert* schlussendlich auch.	Es wird erwartet, dass etwas *nicht* funktioniert, aber es funktioniert schlussendlich *doch*.
Negative	Es wird erwartet, dass etwas *nicht* funktioniert und es funktioniert schlussendlich auch *nicht*.	Es wird erwartet, dass etwas *funktioniert*, aber es funktioniert *doch nicht*.

Tabelle 12 Test-Klassifikationen

5.2.2 Template

Testfall-Nummer					
Bezeichnung					
Beschreibung / Zu testende Funktionalität					
Klassifikation					
Use-Case-Nummer					
Datum					
Tester					
Voraussetzung					
Testumgebung					
	Software				
	Betriebssystem				
	Hardware				
Testschritte					
	Aktion	**Erwartetes Ergebnis**	**Effektives Ergebnis**	**Erfüllt**	**Kommentar**
1					-
2					-
3					-

Tabelle 13 Testfall-Template

5.3 Use-Case-Tabelle

5.3.1 Template

Use-Case-Nummer	
Titel	
Beschreibung	
Akteure	

Pre-Condition	
Ablauf	
Post-Condition	
Alternativer Ablauf	

Tabelle 14 Use-Case-Template

6 Entscheiden

In diesem Abschnitt wird auf die direkten Entscheidungen des Projektes eingegangen. Use-Cases und UML-Diagramme folgen hier. Use-Case-Tabellen sind abgeänderte Test-Templates.

6.1 UML-Diagramme

6.1.1 Class-Diagram

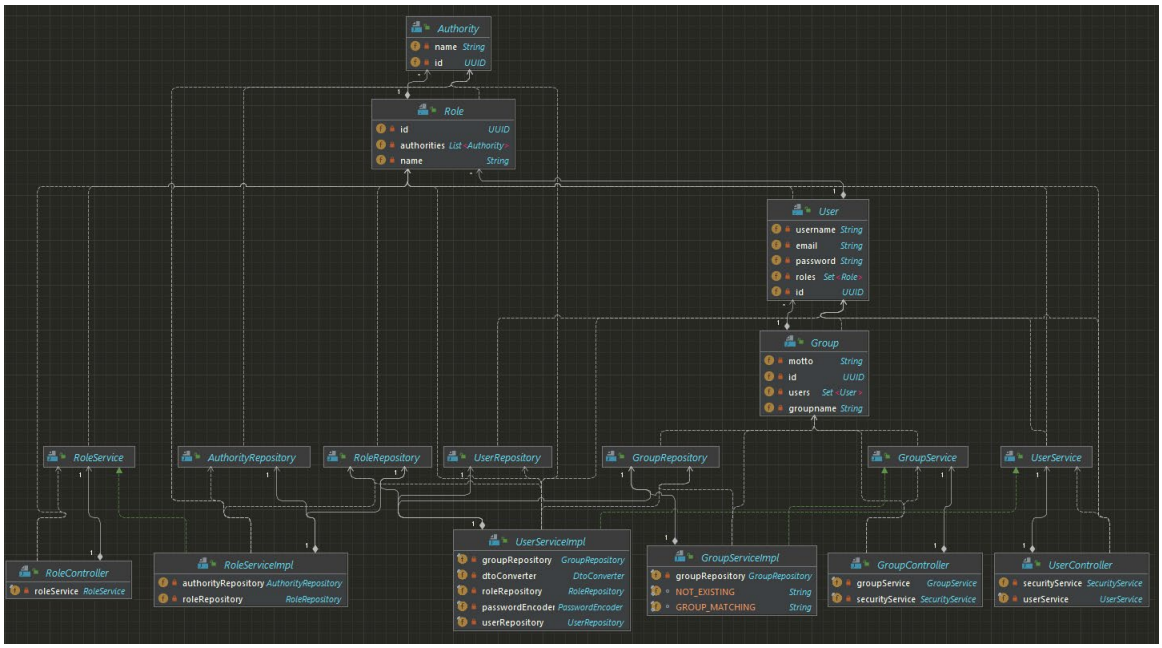


Abbildung 4 UML-Class-Diagram

6.2 Use-Cases

6.2.1 Detaillierter Test-Case

Use-Case-Nummer	**1**
Titel	Post Group
Beschreibung	Bei diesem Endpoint kann man eine Gruppe in form eines mitgegebenem JSON abspeichern
Akteure	Administrator
Pre-Condition	Laufendes Spring-Boot-Backend
Ablauf	1. JSON mit attribute der Gruppe wird definiert, in unserem Fall einen groupname ein motto und ein leerer array mit dem namen users 2. Es gibt drei Verschiedene Antwortmöglichkeiten 3. Wir definieren den namen mit POSTMAN_TEST. Sollte die Gruppe mit dem namen schon existieren gibt es einen HTTP Status 409 zurück. Sollte dies passieren wird ein test case geskippt und nur der testcase ausgeführt, der bestätigt das wir einen conflict (409) hatten.

	4. Wir definieren wieder den namen mit POSTMAN_TEST. Sollte die Gruppe noch nicht existieren, wird ein HTTP Status 201 zurückgegeben und wieder wird 1 test case geskippt.
Post-Condition	Nutzer erhält eine Antwort mit einem http-Status und einem JSON-body.
Alternativer Ablauf	-

Tabelle 15 Use-Case 1

6.3 Entwicklung

Zur Entwicklung werden in diesem Projekt eine Auswahl an Tools verwendet, wobei jedes Tool seinen eigenen Nutzen hat.

6.3.1 GitHub

Nicht nur praktische Versionskontrolle, sondern auch Verwaltung von Aufgaben bietet GitHub an. Für jede Kategorie im Programm wird ein Branch erstellt. Bei Fertigstellung einer Funktionalität oder beim Fix eines Bugs wird über eine *Pull Request* die Änderung in den develop-Branch geladen. Folgende Branch-Struktur herrscht in diesem Projekt:

main	Aktuellste Release-Version
develop	Entwicklerversion mit den neusten Features
feature/{Feature-Name}	Jedes neue Feature bekommt seinen eigenen Entwicklungsbranch. Beispielsweise wird alle User-spezifische Logik im «feature/*user*»-branch entwickelt.

Tabelle 16 Git-Branches

Ein weiterer Nutzen des Systems bietet *GitHub Issues*, mit der Verwaltung kleinerer und grösserer Aufgaben. Üblicherweise wird ein *Issue* mit dem Merge eines Branches geschlossen.

7 Realisieren

Das Projekt wurde im Zeitraum des überbetrieblichen Kurses während der Arbeitszeit realisiert. Ausserhalb der Zeiten was es uns nicht erlaubt, an dem Projekt zu arbeiten. Es kamen täglich neue Theorieinputs dazu. Dieses Wissen aus den Inputs wurde jeweils direkt implementiert.

8 Kontrollieren

Das finalisierte Produkt entspricht unseren Erwartungen und den Anforderungen der Leistungsbeurteilungsvorgabe. Es wurde alles im GitHub-Repository abgegeben. Die Anforderungen finden sich weiter oben in diesem Dokument und detailliert im Dokument «ProjektAufgabe_2022_v4.pdf»

8.1 Checkliste

Funktionale Anforderungen

- ✓ User, Rollen & Privilegien
- ✓ Security
- ✓ User Management
- ✓ Groups

Nicht funktionale Anforderungen

- ✓ Implementation
- ✓ Testing
- ✓ Multiuserfähigkeit
- ✓ Dokumentation
    - Domänenmodell fehlt

9 Auswerten

9.1 Verbesserungsmöglichkeiten

Teilweise gab es Probleme bei den Coding-Conventions. Jeder hat seine eigenen Gewohnheiten, weshalb wir uns anfangs ursprünglich auf gewisse Konventionen einigten. Diese wurden jedoch nicht immer eingehalten.

9.2 Reflexion

Wir sind beide mit unserem Endprodukt sehr zufrieden. Bei der Entwicklung lernten wir sehr viel über Spring Boot, was auch im Alltag angewandt werden kann. Generell sind wir in einem akzeptablen Tempo vorrangekommen. Die Kommunikation lief auch. Zum Schluss gerieten wir noch in einen leichten Zeitstress, da die Postman-Tests noch nicht fertiggestellt waren.

9.3 Zukunftsaussichten

Das Projekt wird noch in der Erweiterung des überbetrieblichen Kurses weitergeführt. Da die Implementierung sehr sauber geschehen ist, kann die Applikation ohne grosse Probleme erweitert werden.