Sprint 2

Freyschmidt, Henry Lewis (HLF)
Hama, Zana Salih (ZSH)
Krasnovska, Paula (PK)
Krüger, Lucas (LK)
Prüger, Marvin Oliver (MOP)
Seep, Tom-Malte (TMS)
Zabel, Steven (SZ)

10. Mai 2024

	Verantwortung	Inhalt	Grafik	Korrektur
Backlog				
Änderungen am Backlog	SZ	SZ	-	-
User-Storys & Tasks	SZ	SZ	-	Alle
Aufwandsschätzung der Tasks	Alle	Alle	-	-
Ablaufplan zu den Tasks	SZ	SZ	SZ	-
Feinentwurf				
Komponentendiagramm	SZ	LK	SZ	-
Klassendiagramme	SZ	HLF, LK	SZ	HLF
Neues/verändertes Verhalten	LK	LK	LK	HLF
Implementation und Tests				
HTTP-Request-Tests	LK	LK	-	HLF
Logic-Tests	HLF	HLF	-	-
Datenbank-Tests	MOP	MOP	-	HLF
Abnahmetest.T2	ZSH	ZSH	-	PK
Laufender Prototyp	PK	PK, ZSH	PK, ZSH	-
Abweichung von Sprintplanung	PK	PK	-	-
Abweichung Datenbank Auswertung	TMS	TMS	-	-

Task-ID	Task-Beschreibung/Name	Teammitglied	Beitrag in %
A4.F1	Erstelle Anzeige zur Sicherheitscode-Anforderung	PK	100%
A4.B1	Funktionalität: Sicherheitscode-Erstellung	LK	100%
A4.B2	Funktionalität: Email-Erstellung	HLF	100%
A4.B3	Funktionalität: Email-Versendung	HLF	100%
A4.B4	Funktionalität: Behandlung einer Sicherheitscode-Anfrage	LK	100%
A5.F1	Erstelle Anzeige zur Passwort-Wiederherstellung	PK	100%
A5.B1	Funktionalität: Passwort-Wiederherstellung	LK HLF	75% 25%
A5.D1	Erstelle Funktionalität zur Passwort-Überspeicherung	TMS MOP	50% 50%
P1.F1	Erstelle Anzeige für Profile	PK	100%
P1.B1	Funktionalität: Profil-Anzeige	LK HLF	75% 25%
P1.D1	Erstelle Speicher für Profile	TMS MOP	50% 50%
P1.D2	Erstelle Funktionalität zur Profile-Ausgabe	TMS MOP	50% 50%
P2.F1	Erstelle Anzeige zur Profil-Anpassung	PK	100%
P2.B1	Funktionalität: Profil-Erstellen&-Bearbeiten	LK HLF	75% 25%
P2.D1	Erstelle Funktionalität zur Profile-Speicherung	TMS MOP	50% 50%
U3.B1	Funktionalität: User-Story-Erstellen	LK HLF	75% 25%
U4.B1	Funktionalität: User-Story-Bearbeiten	LK HLF	75% 25%
U6.F1	Erstelle Anzeige zum User-Story-Löschen	ZSH	100%
U6.B1	Funktionalität: User-Story-Erstellen	LK HLF	50% 50%
U6.D1	Erstelle Funktionalität zur User-Story-Löschung	TMS MOP	50% 50%
T1.F1	Erstelle Anzeige zum Tasks-Anzeigen, [...]	ZSH	100%
T1.B1	Funktionalität: Task-Anzeigen	LK HLF	75% 25%
T1.D1	Erstelle Funktionalität zur Task-Ausgabe zu T3.D1	TMS MOP	50% 50%
T3.F1	Erstelle Anzeige zur Task-Erstellung [...]	ZSH PK	90% 10%
T3.B1	Funktionalität: Task-Erstellen	LK HLF	50% 50%
T3.D1	Erstelle Speicher für Tasks mit User Story-Verknüpfung	TMS MOP	50% 50%
T3.D2	Erstelle Funktionalität zur Task-Speicherung	TMS MOP	50% 50%

Task-ID	Task-Beschreibung/Name	Teammitglied	Beitrag in %
T4.F1	Erstelle Anzeige zur Task-Anpassung, [...]	ZSH	90%
		PK	10%
T4.B1	Funktionalität: Task-Bearbeiten	LK	50%
		HLF	50%
T4.D1	Erstelle Funktionalität zur Task-Überspeicherung	TMS	50%
		MOP	50%
T5.F1	Erstelle Anzeige zum Task-Löschen	ZSH	100%
T5.B1	Funktionalität: Task-Löschen	LK	50%
		HLF	50%
T5.D1	Erstelle Funktionalität zur Task-Löschung	TMS	50%
		MOP	50%
S7.B1	Erstelle Funktionalität zur SessionID-Erstellung	LK	100%
S7.B2	Erweitere Login für SessionID Zuweisung	LK	100%
S7.B3	Nur User mit SessionID können System verwenden	LK	100%
S7.B4	Erstelle Funktionalität zur SessionID-PrÜfung		%
S7.D1	Erweitere den User um eine SessionID	TMS	50%
		MOP	50%
S7.D2	Erstelle Funktionalität zur SessionID-Speicherung	TMS	50%
		MOP	50%
S7.D3	Erstelle Funktionalität zur User- Rechteverwaltung	TMS	50%
		MOP	50%
S7.D4	Erstelle Funktionalität zur User-SessionID-Löschung	TMS	50%
		MOP	50%
	Refactoring Backend	HLF	50%
		LK	50%
	Refactoring Frontend	PK	100%
	Refactoring Datenbank	TMS	75%
		MOP	25%
	Logic-Tests	HLF	100%
	HTTP-Tests	LK	100%
	DB-Tests	MOP	50%
		TMS	50%
	Product Owner	SZ	100%
	Anlegen und Verwalten von Tasks	SZ	100%

Inhaltsverzeichnis

1 Backlog

1.1 Änderungen am Backlog

Element	von	zu
TL1	Als Nutzer kann ich mir alle Task-Listen anzeigen lassen, um eine Übersicht über diese zu bekommen.	
TL6	Als Nutzer kann ich Task-Listen nach der Mächtigkeit filtern, um nur Task-Listen, deren Mächtigkeiten in einem ausgewählten Intervall liegen, zu finden.	
TL4	Als Nutzer kann ich zum Projekt eine Task-Liste mit den Tasks des Projekts finden, um einen Überblick über die Tasks des Projekts zu bekommen.	
TL5	Als Nutzer kann ich zum Sprint eine Task-Liste mit den Tasks des Sprints finden, um einen Überblick über die Tasks des Sprints zu bekommen.	
TB1	Als Nutzer kann ich ein Task-Board mit der Task-Liste des Projekts und den Task-Listen jedes Sprints aufrufen, um das Projekt verwalten zu können.	
TL2	Als Nutzer kann ich eine Task-Liste ausklappen, um ihre Tasks einzublenden.	
TL3	Als Nutzer kann ich eine Task-Liste einklappen, um ihre Tasks auszublenden.	
TL8		Als Nutzer stehen mir zu jedem Task-Board die Task-Listen „freie Tasks“, „Tasks in Bearbeitung“, „Tasks unter Review“, „Tasks unter Test“ und „fertiggestellte Tasks“ zur Verfügung, um Tasks während der Bearbeitung einordnen zu können.
TB2	Als Nutzer kann ich ein Task-Board mit der Task-Liste eines Sprints und den zum Sprint spezifischen Task-Listen 'Tasks in Bearbeitung', 'Tasks unter Review', 'Tasks unter Test' und 'fertiggestellte Tasks' aufrufen, um den Sprint verwalten zu können.	Als Nutzer kann ich ein Task-Board mit allen seinen Task-Listen aufrufen, um es zu verwalten.
T16		Als Nutzer kann ich einem Task-Board Tasks zuordnen, um anzugeben, dass diese auf diesem Task-Board zu bearbeiten sind.

Tabelle 1: Task-Listen sollen im Tool nur auf den Task-Boards einsehbar (TL1 TL6, TL4, TL5) und immer ausgeklappt (TL2, TL3) sein. Da TL4 und TL5 entfernt wurden, wird auch TB1 entfernt. TL8 und die neue Beschreibung von TB2 ersetzen TL5 und die alte Beschreibung von TB2. Es muss im Projekt möglich sein, Tasks den Task-Boards zuzuordnen, um mit den Task-Boards arbeiten zu können (T16).

Element	von	zu
S7		Als Nutzer habe ich eine temporäre, für eine Stunde gültige, SessionID, um mich mit dieser gegenüber des Systems auszuweisen.

Tabelle 2: S7 soll genutzt werden, um einem Nutzer das mehrfache Aufrufen des Systems ohne häufigeres Einloggen zu gewährleisten, das System vor Nutzern, welche nicht eingeloggt sind, zu verbergen und innerhalb des Systems anhand der SessionID auf den Nutzer zu verweisen, beispielsweise bei der Abfrage nach seinen Rechten.

Element	von	zu
P2	Als Nutzer kann ich auf meinem Profil meinen Namen, eine persönliche Beschreibung, eine visuelle Rolle und eine projektbezogene Beschreibung angeben, um mich zu beschreiben.	Als Nutzer kann ich auf meinem Profil einen Namen, eine persönliche Beschreibung, eine visuelle Rolle und eine projektbezogene Beschreibung anpassen, um mich zu beschreiben.
P3	Als Nutzer kann ich meine Angaben auf meinem Profil ändern, um diese zu aktualisieren.	

Tabelle 3: P2 und P3 wurden zusammengeführt, da P2 lediglich eine Funktion von P3 beschrieben hatte.

Element	von	zu
U3.B1	Erstelle Funktionalität zur User-Story-Erstellung, welche, wenn der Nutzer die Rechte dazu hat, die Anzeige zur User-Story-Erstellung ausliest und die Funktionalität zur User-Story-Speicherung aufruft.	Erstelle Funktionalität zur User-Story-Erstellung, welche die Funktionalität zur Rechteverwaltung (S7.D3) aufruft und, falls der Nutzer die Rechte zum Erstellen von User-Storys hat, die Anzeige zur User-Story-Erstellung (U3.F1) ausliest und die Funktionalität zur User-Story-Speicherung (U3.D2) aufruft.
U4.B1	Erstelle Funktionalität zur User-Story-Anpassung, welche, wenn der Nutzer die Rechte dazu hat, die Anzeige zur User-Story-Anpassung ausliest und die Funktionalität zur User-Story-Überspeicherung aufruft.	Erstelle Funktionalität zur User-Story-Anpassung, welche die Funktionalität zur Rechteverwaltung (S7.D3) aufruft und, falls der Nutzer die Rechte zum Editieren von User-Storys hat, die Anzeige zur User-Story-Anpassung (U4.F1) ausliest und die Funktionalität zur User-Story-Überspeicherung (U4.D1) aufruft.
T1	Als Nutzer kann ich mir alle Tasks anzeigen lassen, um eine Übersicht über diese zu bekommen.	Als Nutzer kann ich mir alle Tasks anzeigen lassen, um eine Übersicht über die Tasks des Projekts zu bekommen.
T5	Als Nutzer kann ich eine Task löschen, um nicht mehr nötige Aufgabe zu entfernen.	Als Nutzer kann ich eine Task löschen, um eine nicht mehr nötige Aufgabe zu entfernen.

Tabelle 4: Bei den Tasks und User-Storys gab es keine inhaltliche Änderung, sondern sie wurden lediglich an die sonstigen Änderungen angepasst (U3.B1, U4.B1, T1) oder korrigiert (T5).

1.2 Planung des Sprints

Die Tasks sind eingeteilt in die Teams [F] Frontend, [B] Backend und [D] Daten-Ebene. Das Frontend umfasst die Funktionalität zur Präsentation, das Backend umfasst die Funktionalität zur Logik und die Daten-Ebene umfasst die Funktionalität zur Datenspeicherung, -ausgabe und -manipulation.

- Product Owner: SZ
- Frontend-Entwicklung: PK, ZSH
- Backend-Entwicklung: HLF, LK
- Daten-Entwicklung: MOP, TMS

Der Sprint-Backlog ist auch in Trello verfügbar:
Link: https://trello.com/invite/b/9ToB2gde/ATTI98be4b5d08ce8e43d4d1298692e6cf47E03F05DD/sprint-2

In diesem Sprint werden folgende inhaltliche Komplexe angegangen:

1. die Umsetzung der User-Storys und Tasks, welche aus dem ersten Sprint noch zu beenden sind,
2. die Umsetzung der Passwort-Änderung,
3. die Einführung von Profilen, sodass ein Nutzer sein eigenes Profil ansehen und bearbeiten kann,
4. die Einführung von Tasks, sodass ein Nutzer Aufgaben definieren und einsehen kann und
5. die Umsetzung von SessionIDs, um Nutzer zu authentifizieren, sodass man sich einloggen muss, um das System zu nutzen.

1.2.1 [A] Account

⊗ **A4** Als potenzieller Nutzer kann ich mit meiner Mail einen Sicherheitscode anfordern, um diesen, falls ich mein Passwort vergessen habe, zur Authentifikation zu nutzen.

- ⊗ **A4.F1** Erstelle Anzeige zur Sicherheitscode-Anforderung, welche eine Mail abfragt.
  - Aufwandseinschätzung: 0,25h
  - Bearbeitung: 0,75h durch PK
  - Review: 0,25h durch ZSH
  - Test: 0,25h durch ZSH
- ⊗ **A4.B1** Erstelle Funktionalität zur Sicherheitscode-Erstellung, welche einen Sicherheitscode erstellt und zurückgibt.
  - Aufwandseinschätzung: 2h
  - Bearbeitung: 1h durch LK
  - Review: 0.25h durch HLF
  - Test: 0,25h durch HLF, LK
- ⊗ **A4.B2** Erstelle Funktionalität zur Mail-Erstellung, welche eine Mail erstellt.
  - Aufwandseinschätzung: 1h
  - Bearbeitung: 0,5h durch HLF
- ⊗ **A4.B3** Erstelle Funktionalität zur Mail-Versendung, welche eine Mail versendet.
  - Aufwandseinschätzung: 2h
  - Bearbeitung: 3h durch HLF

- ⊗ **A4.B4** Erstelle Funktionalität zur Sicherheitscode-Anforderung, welche die Anzeige zur Sicherheitscode-Anforderung (A4.F1) ausliest, die Funktionalität zur Sicherheitscode-Erstellung (A4.B1), die Funktionalität zur Mail-Erstellung (A4.B2), die Funktionalität zur Mail-Versendung (A4.B3) und die Funktionalität zur Passwort-Wiederherstellung (A5.B1) aufruft.
  - Aufwandseinschätzung: 2h
  - Bearbeitung: 2h durch LK
  - Review: 0,25h durch HLF
  - Test: 0,25h

- ⊗ **A5** Als potenzieller Nutzer kann ich mit meiner Mail und meinem aktuellsten Sicherheitscode mein Passwort zurücksetzen, um einen verlorenen oder unsicheren Zugang selbstständig wiederherzustellen oder sicherer zu machen.
  - ⊗ **A5.F1** Erstelle Anzeige zur Passwort-Wiederherstellung, welche einen Sicherheitscode und ein neues Passwort abfragt.
    - Aufwandseinschätzung: 0,25h
    - Bearbeitung: 0,5h durch PK
    - Review: 0,25h durch ZSH
    - Test: 0,25h durch ZSH
  - ⊗ **A5.B1** Erstelle Funktionalität zur Passwort-Wiederherstellung, welche einen Sicherheitscode und eine Mail entgegennimmt, bei Nutzung der Anzeige zur Passwort-Wiederherstellung (A5.F1), die Anzeige zur Passwort-Wiederherstellung (A5.F1) ausliest, den Sicherheitscode vergleicht und bei Erfolg die Funktionalität zur Passwort-Speicherung (A5.D2) aufruft.
    - Aufwandseinschätzung: 1h
    - Bearbeitung: 0,25h durch LK, HLF
    - Review: 0,25h durch HLF
    - Test: 0,25h
  - ⊗ **A5.D1** Erstelle Funktionalität zur Passwort-Speicherung, welche eine Mail und ein Passwort entgegennimmt und im Speicher für Accounts (A1.D1) zu einer Mail das Passwort überspeichert.
    - Aufwandseinschätzung: 0,25h
    - Bearbeitung: 0,25h durch MOP, TMS

1.2.2 [P] Profil

- ⊗ **P1** Als Nutzer habe ich ein Profil, um mich mit diesem im System zu zeigen.
  - ⊗ **P1.F1** Erstelle Anzeige für Profile, welche einen Namen, eine persönliche Beschreibung, eine visuelle Rolle und eine projektbezogene Beschreibung anzeigt.
    - Aufwandseinschätzung: 1h
    - Bearbeitung: 1,5h durch PK
    - Review: 0,25h durch ZSH
    - Test: 0,25h durch ZSH
  - ⊗ **P1.B1** Erstelle Funktionalität zum Profil-Anzeigen, welche die Funktionalität zur Profil-Ausgabe (P1.D2) aufruft und das erhaltene Tupel auf die Anzeige für Profile (P1.F1) abbildet.
    - Aufwandseinschätzung: 1h
    - Bearbeitung: 0,75h durch LK, HLF

    - Test: 0,5h durch LK
  - ⊗ **P1.D1** Erstelle Speicher für Profile, welcher zu einem Account einen Namen, eine persönliche Beschreibung, eine visuelle Rolle und eine projektbezogene Beschreibung speichert.
    - Aufwandseinschätzung: 0,5h
    - Bearbeitung: 0,25h durch TMS
  - ⊗ **P1.D2** Erstelle Funktionalität zur Profil-Ausgabe, welche aus dem Speicher für Profile (P1.D1) zu einem Nutzer den Namen, die persönliche Beschreibung, die visuelle Rolle und die projektbezogene Beschreibung ausgibt.
    - Aufwandseinschätzung: 1h
    - Bearbeitung: 1h durch MOP, TMS
    - Test: 0,5h durch MOP

- ⊗ **P2** Als Nutzer kann ich auf meinem Profil einen Namen, eine persönliche Beschreibung, eine visuelle Rolle und eine projektbezogene Beschreibung anpassen, um mich zu beschreiben.
  - ⊗ **P2.F1** Erstelle Anzeige zur Profil-Anpassung, welche es erlaubt, einen Namen, eine persönliche Beschreibung und eine projektbezogene Beschreibung einzugeben und eine visuelle Rolle auszuwählen.
    - Aufwandseinschätzung: 2 Stunden
    - Bearbeitung: 2h durch PK
    - Review: 0,25h durch ZSH
    - Test: 0,25h durch ZSH
  - ⊗ **P2.B1** Erstelle Funktionalität zur Profil-Anpassung, welche die Anzeige zur Profil-Anpassung (P2.F1) ausliest und die Funktionalität zur Profil-Speicherung (P2.D1) aufruft.
    - Aufwandseinschätzung: 0,5h
    - Bearbeitung: 0,75h durch LK, HLF
    - Test: 0,75h durch HLF
  - ⊗ **P2.D1** Erstelle Funktionalität zur Profil-Speicherung, welche im Speicher für Profile (P1.D1) zu einem Nutzer den Namen, die persönliche Beschreibung, die visuelle Rolle und die projektbezogene Beschreibung überschreibt.
    - Aufwandseinschätzung: 1h
    - Bearbeitung: 1h durch MOP, TMS
    - Test: 0,5h durch MOP

1.2.3 [U] User-Story

- ⊗ **U3** Als Product Owner kann ich eine User-Story erstellen, um Anforderungen dem Projekt hinzuzufügen.
  - ⊗ **U3.B1** Erstelle Funktionalität zur User-Story-Erstellung, welche die Funktionalität zur Rechteverwaltung (S7.D3) aufruft und, falls der Nutzer die Rechte zum Erstellen von User-Storys hat, die Anzeige zur User-Story-Erstellung (U3.F1) ausliest und die Funktionalität zur User-Story-Speicherung (U3.D2) aufruft.
    - Aufwandseinschätzung: 5h
    - Bearbeitung: 5h durch HLF, LK
    - Review: 1h durch ZSH, PK
    - Test: 1h durch HLF
- ⊗ **U4** Als Product Owner kann ich eine User-Story editieren, um sie an neue Umstände anzupassen.

- ⊗ **U4.B1** Erstelle Funktionalität zur User-Story-Anpassung, welche die Funktionalität zur Rechteverwaltung (S7.D3) aufruft und, falls der Nutzer die Rechte zum Editieren von User-Storys hat, die Anzeige zur User-Story-Anpassung (U4.F1) ausliest und die Funktionalität zur User-Story-Überspeicherung (U4.D1) aufruft.
  - ∗ Aufwandseinschätzung: 3h
  - ∗ Bearbeitung: 3h durch HLF
  - ∗ Review: 1h durch ZSH, PK
  - ∗ Test: 1h durch HLF

⊗ **U6** Als Product Owner kann ich eine User-Story löschen, um nicht mehr nötige Anforderungen zu entfernen.

- ⊗ **U6.F1** Erstelle Anzeige zum User-Story-Löschen.
  - ∗ Aufwandseinschätzung: 0,25h
  - ∗ Bearbeitung: 0,25h durch ZSH
- ⊗ **U6.B1** Erstelle Funktionalität zum User-Story-Löschen, welche beim Benutzen der Anzeige zum User-Story-Löschen (U6.F1) die Funktionalität zur Rechteverwaltung (S7.D3) aufruft und, falls der Nutzer die Rechte zum Löschen von User-Storys hat, zu jeder zur User-Story verlinkten Task die Funktionalität zur Task-Löschung (T5.D1) und für die User-Story die Funktionalität zur User-Story-Löschung (U6.D1) aufruft.
  - ∗ Aufwandseinschätzung: 1h
  - ∗ Bearbeitung: 1,25h durch LK
  - ∗ Test: 0,75h durch HLF
- ⊗ **U6.D1** Erstelle Funktionalität zur User-Story-Löschung, welche eine User-Story entgegen nimmt und diese aus dem Speicher für User-Storys (U3.D1) löscht.
  - ∗ Aufwandseinschätzung: 0,5h
  - ∗ Bearbeitung: 0,25h durch MOP, TMS
  - ∗ Test: 0,5h durch MOP, TMS

1.2.4 [T] Task

⊗ **T1** Als Nutzer kann ich mir alle Tasks anzeigen lassen, um eine Übersicht über diese zu bekommen.

- ⊗ **T1.F1** Erstelle Anzeige zum Tasks-Anzeigen, welche zum Anzeigen aller Tasks dient.
  - ∗ Aufwandseinschätzung: 1h
  - ∗ Bearbeitung: 1h durch ZSH
- ⊗ **T1.B1** Erstelle Funktionalität zum Tasks-Anzeigen, welche beim Aufruf der Anzeige zum Tasks-Anzeigen (T1.F1) die Funktionalität zum Tasks-Ausgeben (T1.D1) aufruft und die erhaltenen Tasks auf die Anzeige zum Tasks-Anzeigen (T1.F1) abbildet.
  - ∗ Aufwandseinschätzung: 0,25h
  - ∗ Bearbeitung: 0,25h durch LK
  - ∗ Test: 0,25h durch LK
- ⊗ **T1.D1** Erstelle Funktionalität zum Tasks-Ausgeben, welche alle Tasks im Speicher für Tasks (T3.D1) ausgibt.
  - ∗ Aufwandseinschätzung: 0,75h
  - ∗ Bearbeitung: 1h durch MOP, TMS
  - ∗ Test: 0,5h durch MOP, TMS

- ⊗ **T3** Als Nutzer kann ich eine zu einer User-Story verlinkte Task erstellen, um diese User-Story zu verfeinern.
  - ⊗ **T3.F1** Erstelle Anzeige zur Task-Erstellung, welche die Eingabe eines Titels und einer Beschreibung ermöglicht.
    - Aufwandseinschätzung: 2h
    - Bearbeitung: 2h durch ZSH
  - ⊗ **T3.B1** Erstelle Funktionalität zur Task-Erstellung, welche die Anzeige zur Task-Erstellung (T3.F1) ausliest und die Funktionalität zur Task-Speicherung (T3.D2) aufruft.
    - Aufwandseinschätzung: 0.25h
    - Bearbeitung: 0,5h durch LK, HLF
    - Test: 0,75h durch HLF
  - ⊗ **T3.D1** Erstelle Speicher für Tasks, welcher zu einer User-Story Tasks speichert.
    - Aufwandseinschätzung: 0,25h
    - Bearbeitung: 0,25h durch TMS
  - ⊗ **T3.D2** Erstelle Funktionalität zur Task-Speicherung, welche eine Task zu einer User-Story entgegennimmt und diese in den Speicher für Tasks (T3.D1) schreibt.
    - Aufwandseinschätzung: 0,75h
    - Bearbeitung: 0,5h durch MOP, TMS
    - Test: 0,25h durch MOP, TMS
- ⊗ **T4** Als Nutzer kann ich eine Task editieren, um sie an neue Umstände anzupassen.
  - ⊗ **T4.F1** Erstelle Anzeige zur Task-Anpassung, welche die Eingabe eines Titels und einer Beschreibung ermöglicht.
    - Aufwandseinschätzung: 0,25h
    - Bearbeitung: 0,25h durch ZSH
  - ⊗ **T4.B1** Erstelle Funktionalität zur Task-Anpassung, welche die Anzeige zur Task-Anpassung (T4.F1) ausliest und die Funktionalität zur Task-Überspeicherung (T4.D1) aufruft.
    - Aufwandseinschätzung: 0,25h
    - Bearbeitung: 0,5h durch LK, HLF
    - Test: 1h durch HLF
  - ⊗ **T4.D1** Erstelle Funktionalität zur Task-Überspeicherung, welche eine Task entgegen nimmt und diese Task im Speicher für Tasks (T3.D1) überschreibt.
    - Aufwandseinschätzung: 1h
    - Bearbeitung: 1h durch MOP, TMS
    - Test: 0,25h durch MOP, TMS
- ⊗ **T5** Als Nutzer kann ich eine Task löschen, um eine nicht mehr nötige Aufgabe zu entfernen.
  - ⊗ **T5.F1** Erstelle Anzeige zum Task-Löschen.
    - Aufwandseinschätzung: 0,25h
    - Bearbeitung: 0,25h durch ZSH
  - ⊗ **T5.B1** Erstelle Funktionalität zum Task-Löschen, welche beim Benutzen der Anzeige zum Task-Löschen (T5.F1) die Task ermittelt und die Funktionalität zur Task-Löschung (T5.D1) aufruft.
    - Aufwandseinschätzung: 0,25h

- Bearbeitung: 0,25h durch LK
- Test: 0,5h durch HLF

⊗ **T5.D1** Erstelle Funktionalität zur Task-Löschung, welche eine Task entgegen nimmt und diese aus dem Speicher für Tasks (T3.D1) löscht.

- Aufwandseinschätzung: 0,5h
- Bearbeitung: 0,5h durch MOP, TMS
- Test: 0,5h durch MOP, TMS

1.2.5 [S] System

⊗ **S7** Als Nutzer habe ich eine temporäre, für eine Stunde gültige, SessionID, um mich mit dieser gegenüber des Systems auszuweisen.

⊗ **S7.B1** Erstelle Funktionalität zur SessionID-Erstellung, welche zu einem Nutzer eine SessionID bestehend aus einer Zufallszahl, einem Timestamp und der ID des Nutzers erstellt.

- Aufwandseinschätzung: 1h
- Bearbeitung: 1h durch LK
- Test: 0,25h

⊗ **S7.B2** Erweitere Funktionalität zum Login (A3.B1), sodass vor einem erfolgreichen Login erst die Funktionalität zur SessionID-Erstellung (S7.B1) und die Funktionalität zur SessionID-Speicherung (S7.D2) aufgerufen werden.

- Aufwandseinschätzung: 1h
- Bearbeitung: 0,5h durch LK
- Test: 0,25h durch LK

⊗ **S7.B3** Erweitere Funktionalität, welche User Interfaces verwaltet, sodass Nutzer ohne gültige SessionID nur die User Interfaces zum Login (A3.F1), zur Account-Registrierung (A2.F1), zur Sicherheitscode-Anforderung (A4.F1) und zur Passwort-Wiederherstellung (A5.F1) einsehen und nutzen können.

- Aufwandseinschätzung: 2h
- Bearbeitung: 2h durch LK
- Test: 1h durch LK

⊗ **S7.B4** Erstelle Funktionalität zur SessionID-Prüfung, welche zu einer SessionID prüft, ob diese noch gültig ist und, falls sie nicht mehr gültig ist, die Funktionalität zur SessionID-Löschung (S7.D3) aufruft.

- Aufwandseinschätzung: 1h
- Bearbeitung: 1h durch LK
- Test: 0,25h durch LK

⊗ **S7.D1** Erweitere Speicher für Accounts (A1.D1) um eine SessionID.

- Aufwandseinschätzung: 0,25h
- Bearbeitung: 0,25h durch TMS

⊗ **S7.D2** Erstelle Funktionalität zur SessionID-Speicherung, welche eine SessionID entgegennimmt und in den Speicher für Accounts (A1.D1) schreibt.

- Aufwandseinschätzung: 0,5h
- Bearbeitung: 0,25h durch MOP, TMS

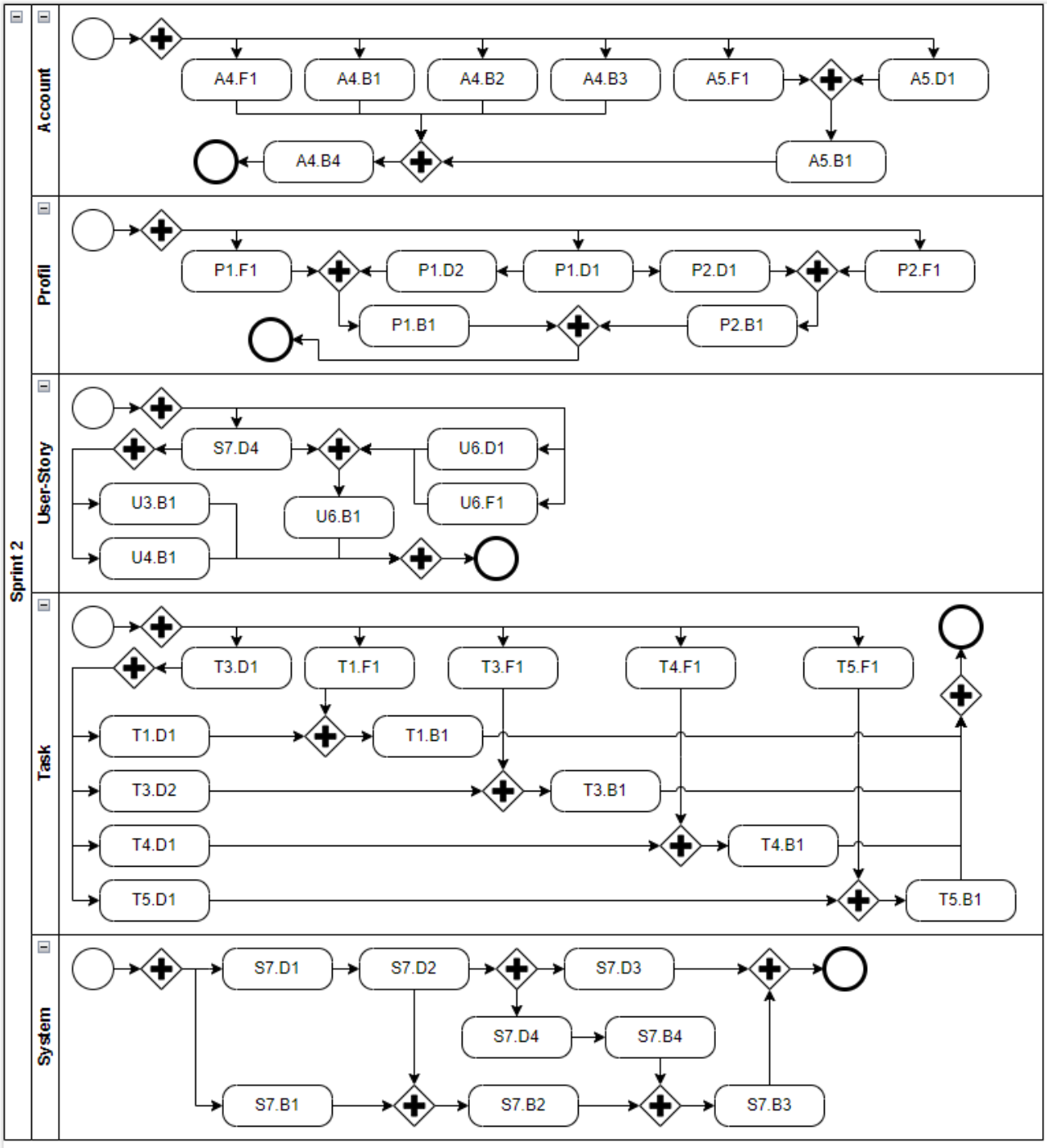
⊗ **S7.D3** Erstelle Funktionalität zur Rechteverwaltung, welche eine SessionID annimmt und die Rechte des Nutzers zurückgibt.

    - Aufwandseinschätzung: 0,5h
    - Bearbeitung: 0,25h durch MOP, TMS
- ⊗ **S7.D4** Erstelle Funktionalität zur SessionID-Löschung, welche eine SessionID aus dem Speicher für Accounts (A1.D1) löscht.
    - Aufwandseinschätzung: 0,5h
    - Bearbeitung: 0,25h durch MOP, TMS

Zusätzlicher Aufwand:

- Refactoring Backend: 9h durch HLF, LK
- Refactoring Frontend: 1h durch PK
- veränderte Darstellung der User-Stories im Project-Manager: 2h durch ZSH
- Refactoring Datenbank: 14h durch TMS, MOP
- Planung der Backend-Tests: 0.5h durch HLF, LK
- Schreiben der HTTP-Request-Test: 4.5h durch LK
- Schreiben der Logic-Tests: 2.5h durch HLF
- Schreiben der Datenbank-Tests: 4h durch MOP
- Schreiben des Abnahmetests.T2: 2h durch ZSH

1.2.6 Ablaufplan zu den Tasks



2 Feinentwurf

2.1 Komponentendiagramm

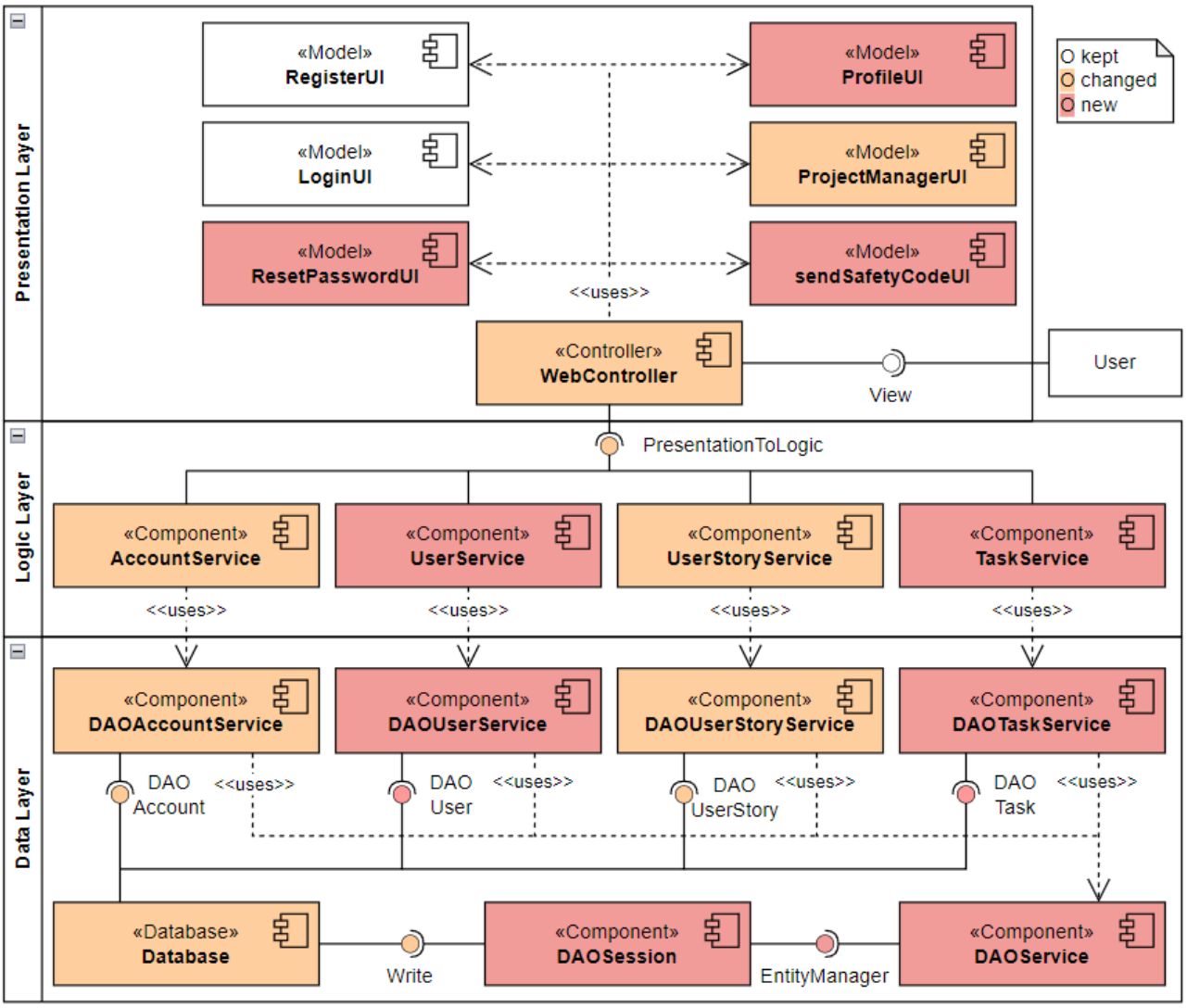


Abbildung 1: Komponentendiagramm des geplanten Software-Projekts

Beginnend mit diesem Sprint repräsentiert das Komponentendiagramm (Abbildung 1) nur noch den aktuellen Scope des Projektes. Somit werden noch geplante Komponenten nicht aufgegriffen, wenn sie nicht in dem aktuellen Sprint behandelt oder geändert werden.
In diesem Sprint wurde das Projekt mit den Funktionalitäten für Profilerstellung, -bearbeitung, -anzeige und -speicherung erweitert, um zwischen den Nutzern der Webanwendung differenzieren zu können. Darüber hinaus ist zukünftig geplant, die Profile anderer Nutzer einsehen zu können, um mehr über seine Kollegen zu erfahren und diese als Projektmanager zu verwalten. Diese Funktionalität ist repräsentiert und umgesetzt durch die Komponenten “ProfileUI“, “UserService“ und “DAOUserService“ (im Komponentendiagramm rot).
Weiterhin wurde der derzeitige Prototyp des Projektes um die Funktionalität der Taskerstellung, -management, -anzeige und -speicherung, sowie dem Zuweisen zu einer User-Story ergänzt. Diese werden durch die Komponenten “ProjectManagerUI“, “TaskService“, “DAOTaskService“ repräsentiert und umgesetzt (im Komponentendiagramm rot). Somit wird sowohl das Erfassen des Zusammen-

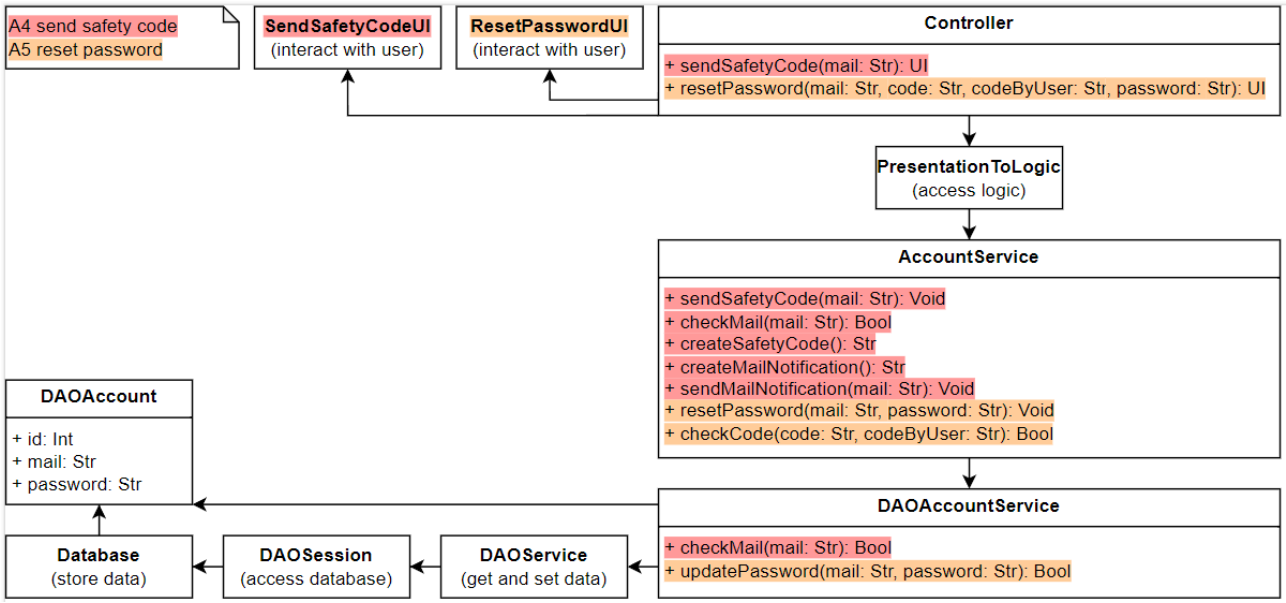
hangs zwischen Tasks und User-Storys als auch das Erhalten eines kompletten Überblicks ermöglicht. Letztlich haben die bereits existenten Funktionalitäten ein substantielles Refactoring erfahren, um das weitere Arbeiten mit diesen zu erleichtern, oder mit neuen Funktionen zur Unterstützung der neuen Komponenten erweitert (im Komponentendiagramm in gelb)

2.2 Klassendiagramme und Design Pattern

In folgenden Abschnitt werden die Klassendiagramme der jeweiligen Aufgabenkomplexe kurz beschrieben und mit eventuellem Design Pattern in Verbindung gesetzt. Aufgrund der Lesbarkeit des folgenden Textes werden folgende Design Pattern schon im Voraus beschrieben, da diese in jeder Funktionalität (abgesehen vom Abschnitt System) vertreten sind.
Design Pattern "Adapter" wird durch die Services der Backend-Ebene umgesetzt, um die Datenübertragung von Frontend zur Datenbank und umgekehrt zu realisieren. Wir haben uns für diese Umsetzung entschieden, um Exceptions schon vor der Verarbeitung in der Datenbank zu catchen und behandeln, sowie die Struktur der Datenbank zu verbergen und eventuelle Konflikte in Datentypen und Speicherung zu behandeln.
Design Pattern "Facade" in Kombination mit dem Design Pattern "Singleton" wurde durch die Datei "PresentationToLogic.java" umgesetzt, um einmal die Lesbarkeit des Codes zu gewährleisten und andererseits das Projekt für zukünftige Erweiterungen offenzuhalten. Das "Singleton"-Pattern hat dazu noch die Funktion zu verhindern, dass durch mehrfache Instanziierung dieser Datei globale Variablen überschrieben werden und so Konflikte auftreten.

2.2.1 Klassendiagramm zum Backlog-Abschnitt Account

Abbildung 2: Klassendiagramm zum Abschnitt Account

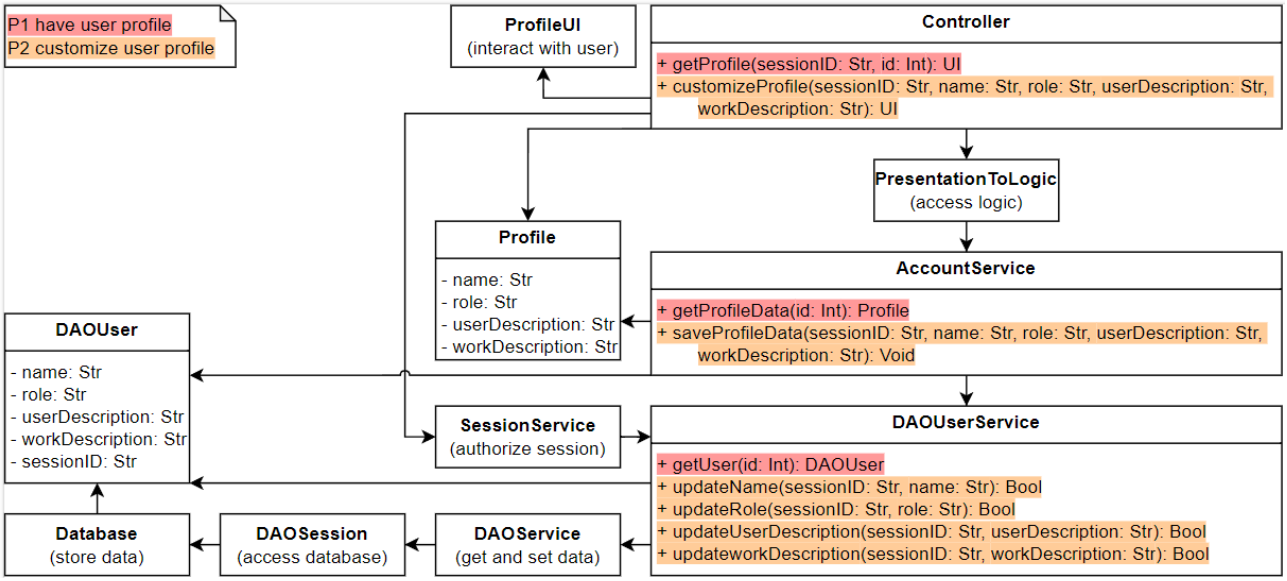


Zur Realisierung der User-Storys des Backlog-Abschnitts Account (rot: A4 und gelb: A5) muss ein Sicherheitscode generiert und das bestehende Passwort in der Datenbank überschrieben werden. Für das Senden des Sicherheitscode (A4) wird durch den dazugehörigen Request die Funktion "sendSafetyCode" aufgerufen, welche erstmals über die Funktion "checkMail" die Existenz des angefragten Accounts sicherstellt. Nach diesem Check wird mittels der Funktion "createSafetyCode" der besagte Sicherheitscode generiert, um dann über die "sendSafetyCode" Funktion das Senden des Codes per Mail zu starten.
Zwar wurde es aufgrund von technischen Problemen nicht umgesetzt, aber planmäßig wäre nun über die "createMailNotification" ein Mail-Object an die gegebene E-Mail erstellt und mit dem Sicherheitscode belegt. Dieses Mail-Object wäre schlussendlich über "sendMailNotification" an den Nutzer gesendet worden.
Für das Zurücksetzen des Passworts (A5) wird im Controller die Funktion "resetPasswort" mit der

E-Mail des Nutzers, dem zugehörigen Sicherheitscode und dem neuen Passwort aus dem Frontend aufgerufen. Diese verifiziert mittels “checkMail“ und “checkCode“ die Authentizität des Codes sowie der E-Mail-Adresse und ruft dann die Datenbank-Anfrage “updatePassword“ auf, um das neue Passwort zu speichern.

2.2.2 Klassendiagramm zum Backlog-Abschnitt Profil

Abbildung 3: Klassendiagramm zum Abschnitt Profil



In Abbildung 3 sind die für die Profilanzeige (P1) relevanten Methoden rot und für die Profilerstellung und -bearbeitung (P2) gelb markiert. Für P1 wird nach passendem Request des Clients im “Controller“ die Methode “getProfile“ aufgerufen, die das Profil zurückgibt, dass zu der übergebenen SessionID gehört. Nachdem die SessionID auf Gültigkeit geprüft wurde, wird im “AccountService“ die gleichnamige Methode aufgerufen, welche die Datenbank-Methoden des “DAOUserService“ aufruft. Da in der Datenbank das Profil als ein eigenes Objekt, sondern als weitere Attribute des Nutzers gespeichert werden, wird mit “getUser“ der Nutzer und somit auch die benötigten Informationen ausgelesen. Das “getProfile“ des “AccountService“ extrahiert die benötigten Daten aus dem Nutzer und übergibt sie als Profil an den “Controller“.
Für P2 wird durch einen Request des Clients die Methode “customizeProfile“ vom “Controller“ aufgerufen. Die Methode bekommt die SessionID des Nutzers und alle profil-spezifische Informationen wie Name, Rolle und Beschreibung übergeben, welche nach Sicherstellung der Gültigkeit der SessionID an “saveProfileData“ des “AccountServices“ weitergegeben werden. Diese ruft die Funktionen “updateName“, “updateRole“, “updateUserDescription“, “updateworkDescription“ des “DAOUserService”, welche die Attribute des Nutzers, dem die übergebene SessionID zugeordnet ist, überschreiben.

2.2.3 Klassendiagramm zum Backlog-Abschnitt User-Story

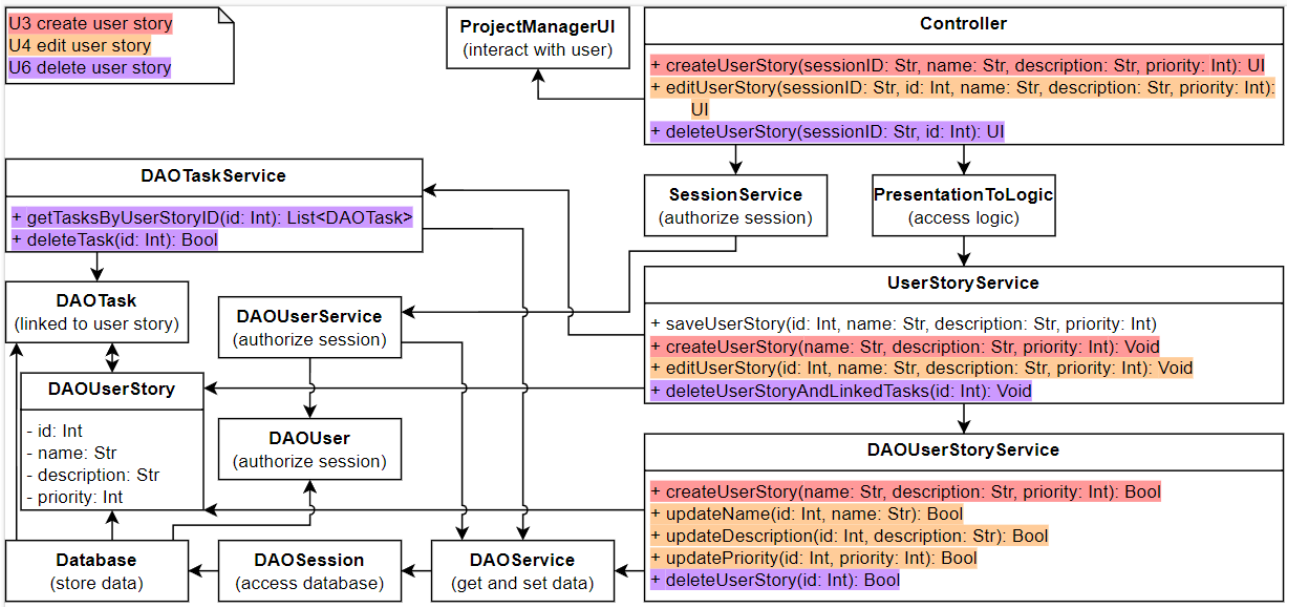


Abbildung 4: Klassendiagramm zum Abschnitt User-Story

Zur Umsetzung der Erstellung (U3), Veränderung (U4) und Löschen (U6) einer User-Story (U3: rot, U4: gelb, U6: lila in Abbildung 4) werden SessionID des Nutzers und User-Story-spezifische Daten wie ID, Name, Beschreibung und Priorität benötigt. Die SessionID ist notwendig, da geprüft werden muss, ob der Nutzer die entsprechenden Rechte besitzt und ob die SessionID gültig ist, beziehungsweise ob das Zeit-Limit überschritten wurde. Dies findet zu Beginn jeder in Abbildung 4 erwähnten Methode des “Controller“ statt. Die Informationen werden vom Request des Clients im “Controller“ empfangen und abhängig vom Request wird “createUserStory“, “editUserStory“ oder “deleteUserStory“ mittels gegebener Daten aufgerufen. Durch “createUserStory“ wird im “UserStoryService“ die Methode “saveUserStory“ aufgerufen. Diese wird sowohl fürs Erstellen als auch fürs Verändern einer User-Story verwendet. Zur Differenzierung der Aufrufe, wird zur Erstellung eine “-1“ als User-Story-ID übergeben. Ein Methodenaufruf mit “-1“ als User-Story-ID provoziert einen Aufruf der Methode “createUserStory“ im “DAOUserStoryService“, die anhand der Argumente eine neue User-Story in der Datenbank erstellt.
Für die Veränderung von User-Storys wird im “Controller“ die Methode “editUserStory“ aufgerufen, die ihre Argumente an die Funktion “saveUserStory“ des “UserStoryService“ weitergibt. Da die User-Story-ID zu einer User-Story gehört und somit nicht “-1“ beträgt, wird im “DAOUserStoryService“ die Methoden “updateName“, “updateDescription“, “updatePriority“ aufgerufen, falls die entsprechenden Argumente ”nicht-nullWerte sind. Diese Funktionen überspeichern die gleichnamigen Attribute der User-Story, dessen ID übergeben wurde.
Beim Aufruf von “deleteUserStory“ im “Controller“ wird nach Sicherstellen der benötigten Rechte des Nutzers im “UserStoryService“ “deleteuserStoryAndLinkedTasks“ aufgerufen. Durch diesen Aufruf wird die User-Story-ID an die “deleteUserStory“-Methode des “DAOUserStoryService“ übergeben, die den Eintrag zur passenden User-Story-ID löscht und alle Tasks, die an diese User-Story-ID verknüpft sind, werden gelöscht (das Löschen einer Task wird im Abschnitt 2.2.4 näher beschrieben).

2.2.4 Klassendiagramm zum Backlog-Abschnitt Task

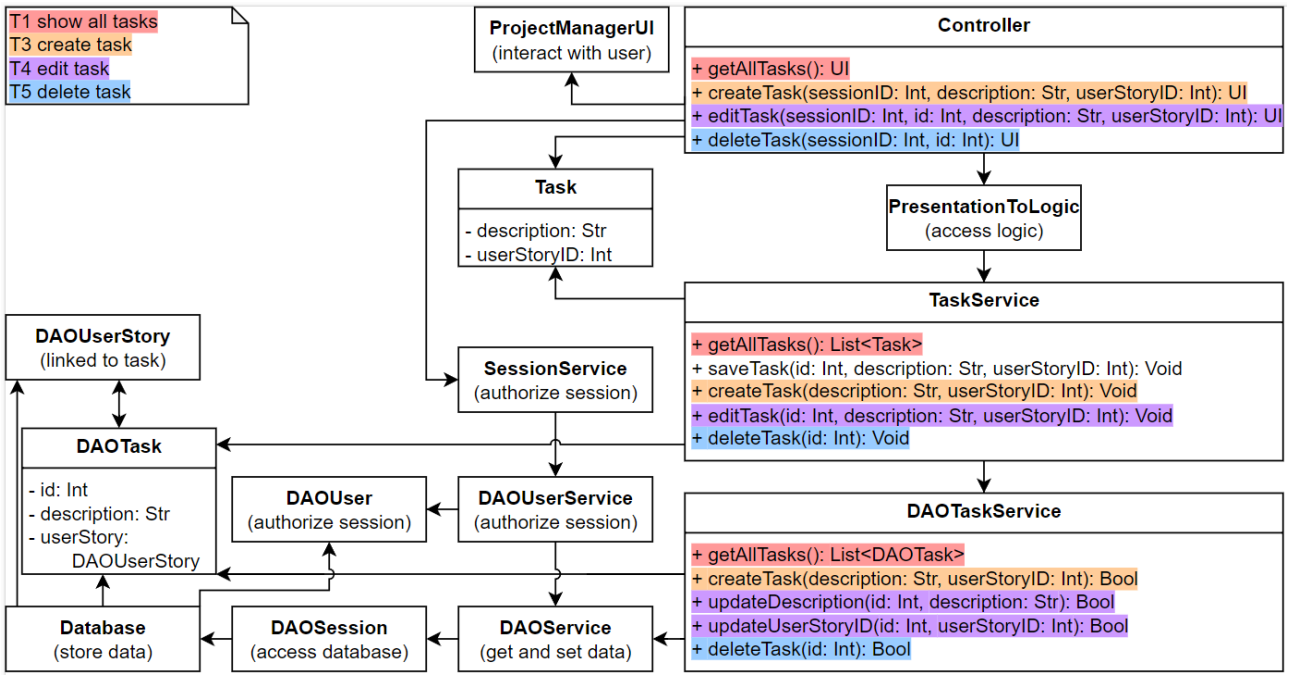


Abbildung 5: Klassendiagramm zum Abschnitt Task

Für die Realisierung User-Storys T1, T3, T4 und T5 müssen Tasks in der Datenbank gespeichert und überschrieben werden können. In Abbildung 5 sind sind die Methoden farblich gekennzeichnet, um ihre Notwendigkeit für die konkrete User-Story darzustellen (T1: rot, T3: gelb, T4: lila, T5: blau). Bevor eine Methode des “TaskService“ vom “Controller“ aufgerufen wird, muss geprüft werde, ob die SessionID des Nutzers gültig ist, beziehungsweise ob das Zeit-Limit überschritten wurde. Um alle Tasks aus dem Nutzer anzeigen zu können, wird durch der passende Request des Clients im “Controller“ die Methode “getAllTasks“ aufgerufen. Diese führt zum Aufruf der gleichnamigen Methode im “TaskService“, welche den wiederum gleichnamigen Aufruf im “DAOTaskService“ hervorbringt. Dadurch werden aus der Datenbank alle Tasks ausgelesen und in eine Liste eingefügt, welche im Aufruf des “TaskService“ in die gewünschte Struktur formatiert werden.
Beim Erstellen einer Task, wird durch den Request eines Clients die Methode “createTask“ vom “Controller“ aufgerufen, welche im “TaskService“ die Funktion “saveTask“ aufruft. Diese Methode wird sowohl für das Erstellen, als auch für das Veränderne einer Task verwendet. Wie bei der User-Story im Abschnitt 2.2.3 wird durch das Argument User-Story-ID unterschieden, da beim Erstellen eine “-1“ und beim Verändern eine existente User-Story-ID übergeben wird. Somit wird im “DAOTaskService“ die Methode “createTask“ aufgerufen, die in der Datenbank die übergebenen Parameter als neue Task abspeichert.
Für das Bearbeiten einer Task, wird nach dem passenden Request des Clients im “Controller“ die “editTask“-Methode aufgerufen, die der Methode “saveTask“ im “TaskService“ dessen Argumente übergibt. Der Aufruf von “saveTask“ wird durch die existente Task-ID als Bearbeitung einer Task erkannt und ruft somit die Methoden “updateDescription’ünd“updateUserStoryID“ vom “DAOTaskService“, welche die Beschreibung und die verknüpfte User-Story der Task in der Datenbank überschreiben.
Um eine Task zu löschen wird initial einen passenden Request des Clients benötigt, der den Aufruf der Methode “deleteTask“ vom “Controller“ provoziert. Dadurch wird im “TaskService“ die gleichnamige Methode aufgerufen, die wiederum im “DAOTaskService“ die gleichnamige Funktion aufruft. Dieser Aufruf bewirkt in der Datenbank das Löschen der Task, die zu der übergebenen Task-ID gehört.

2.2.5 Klassendiagramm zum Backlog-Abschnitt System

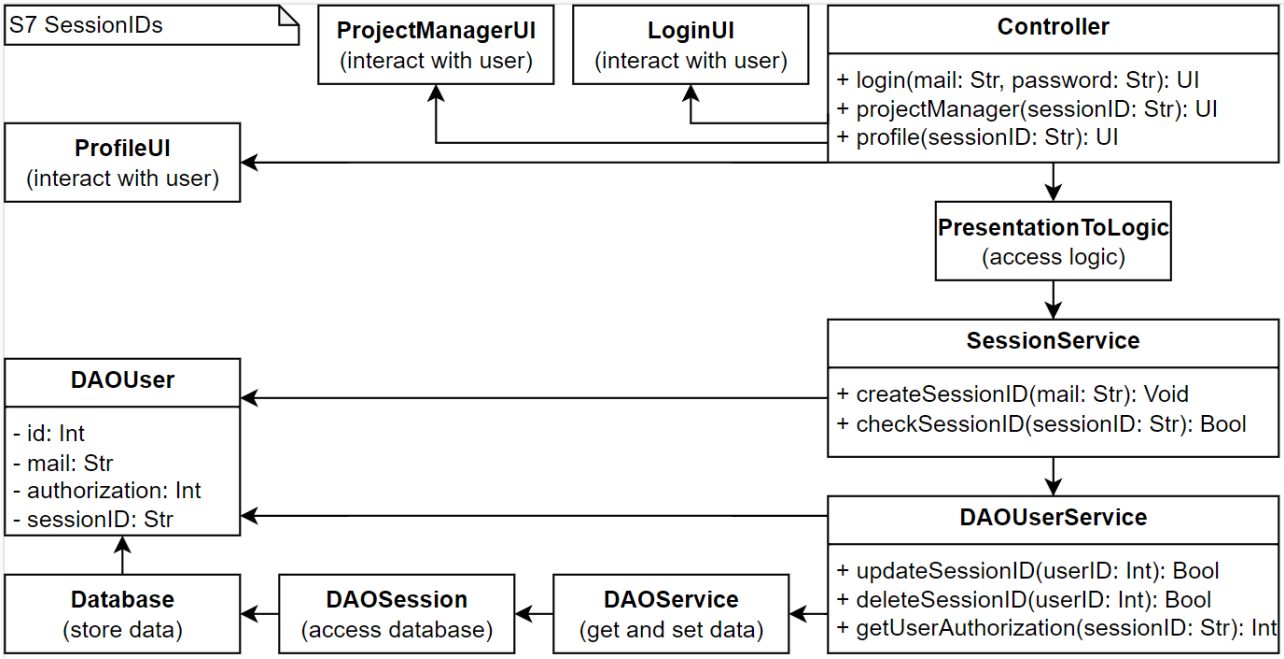


Abbildung 6: Klassendiagramm zum Abschnitt System

Der Prozess der Verwaltung der SessionID eines Users ist ein zweiteiliger Prozess, der den Nutzer gegenüber des Systems verifiziert. Der erste Teil ist das Verbinden mit dem System. Dies geschieht durch die mit dem gleichnamigen Request aufgerufene Funktion “login“ des Controllers, diese ruft wie in Abbildung 7 die Login-Funktionalität auf und erstellt dem Nutzer daraufhin mittels “createSessionID“ eine eindeutige SessionID zur Verifikation gegenüber des Systems und speichert diese zusammen mit der aktuellen Systemzeit in der Datenbank.
Nachdem der Nutzer Zugriff auf das System hat, wird jeder weitere Request an den Controller, wie “projectManager“ oder “profile“, mit der SessionID verifiziert. Dies geschieht bei jedem Aufruf dieser Funktionen mittels ”checkSessionID”. “checkSessionID“ prüft dabei die Existenz der SessionID in der Datenbank und ihre Gültigkeit mittels der im Text zu Abbildung 7 geschriebenen Time To Live (TTL). Erst nachdem dieser Check positiv beendet wurde, werden die jeweiligen Anfragen vom System weiter verarbeitet.

2.3 Neues/verändertes Verhalten

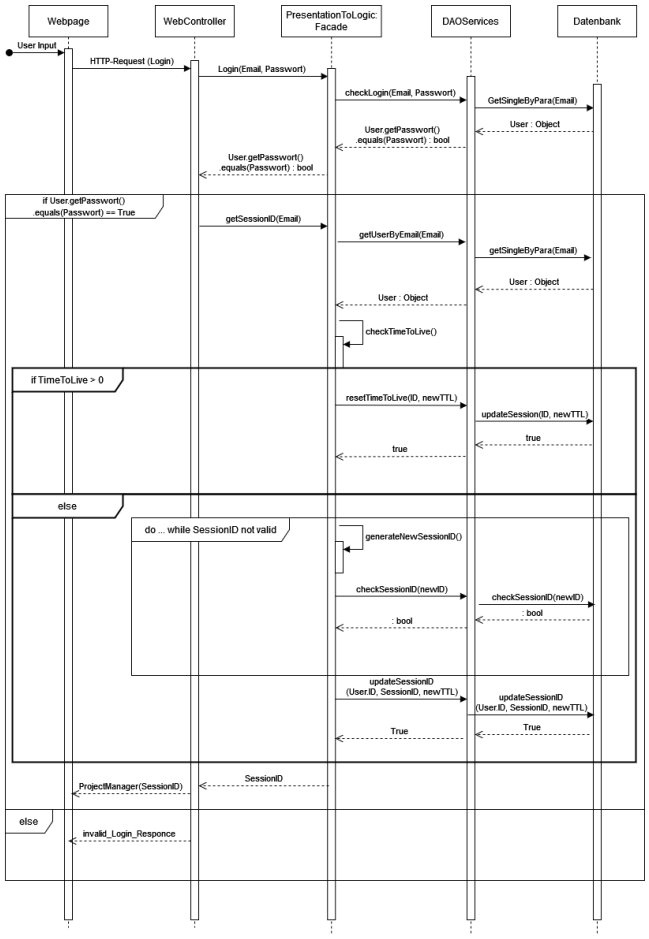
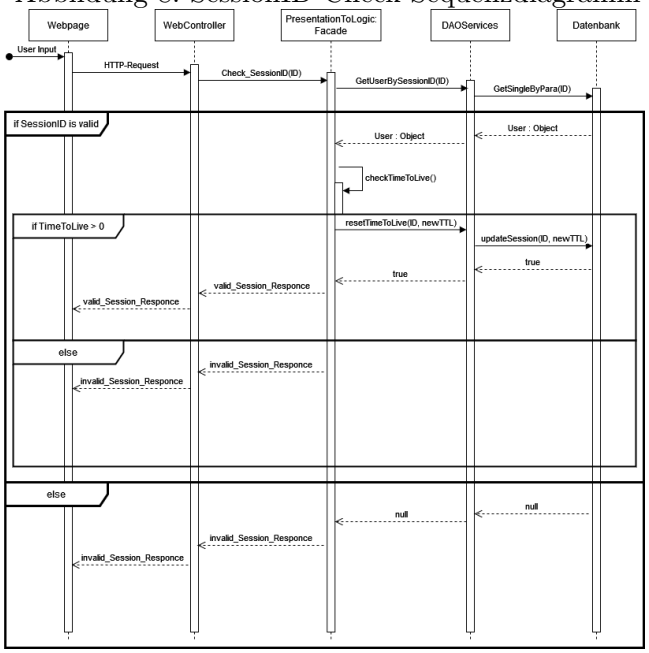


Abbildung 7: Login-Prozess Sequenzdiagramm

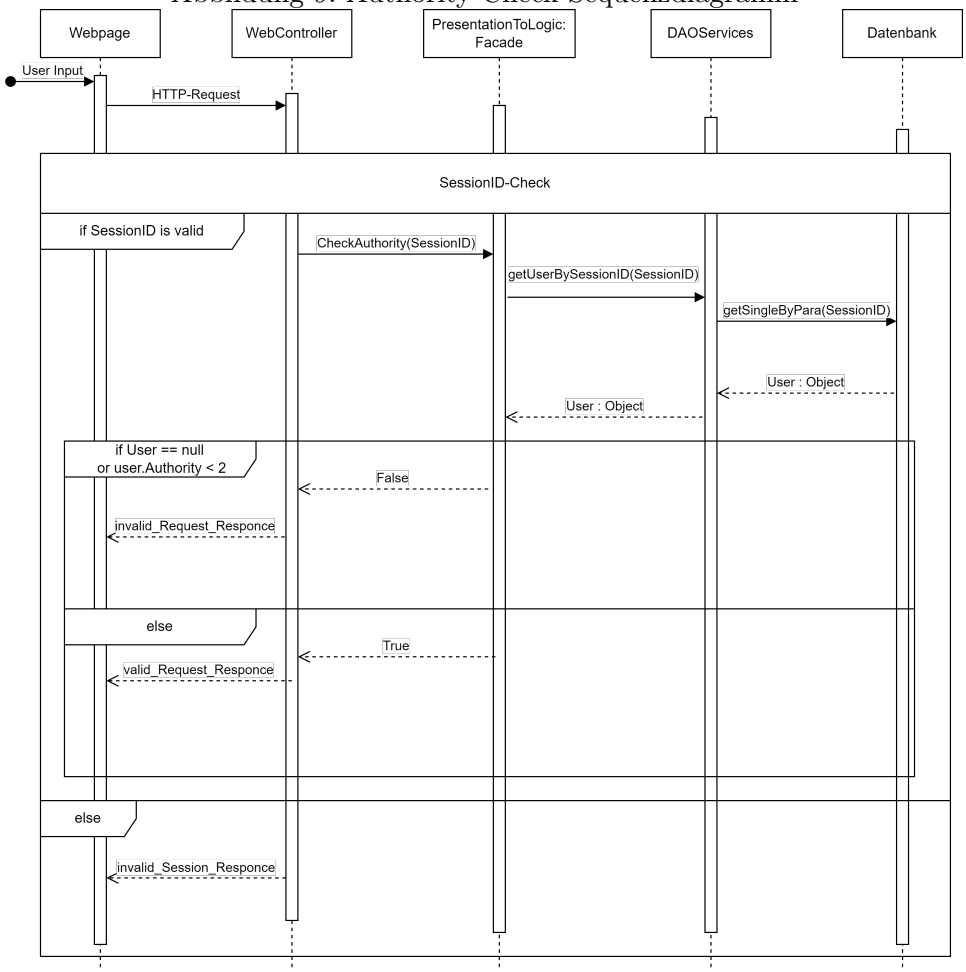
Zur Realisierung der Userstory S7 wurde das Sequenzdiagramm des Login-Prozesses um die Funktionalität der SessionID-Prüfung und -Erstellung erweitert. Somit wird nach der erfolgreichen Prüfung der Login-Informationen ermittelt, ob der Nutzer eine gültige SessionID besitzt. Eine SessionID ist gültig, wenn ihre Time To Live (TTL) größer als 0 ist. Die TTL ist die Differenz zwischen dem in der Datenbank gespeicherten SessionDate (Zeitpunkt der Erstellung oder letzten Prüfung der SessionID) und der aktuellen Systemzeit des Servers.
Sollte der Nutzer keine gültige SessionID besitzen, so wird eine neue generiert, welche nicht in der Datenbank existiert. Bei einer gültigen oder neuen SessionID wird das SessionDate auf die aktuelle Systemzeit gesetzt und der Nutzer zum Projekt Manager weitergeleitet. Sollte bei einem dieser Prozesse ein Error auftauchen oder der Nutzer nicht existieren, beziehungsweise keine gültigen Login-Informationen angegeben hat, so wird ein “invalid_Login_Responce“ übermittelt.

Abbildung 8: SessionID Check Sequenzdiagramm



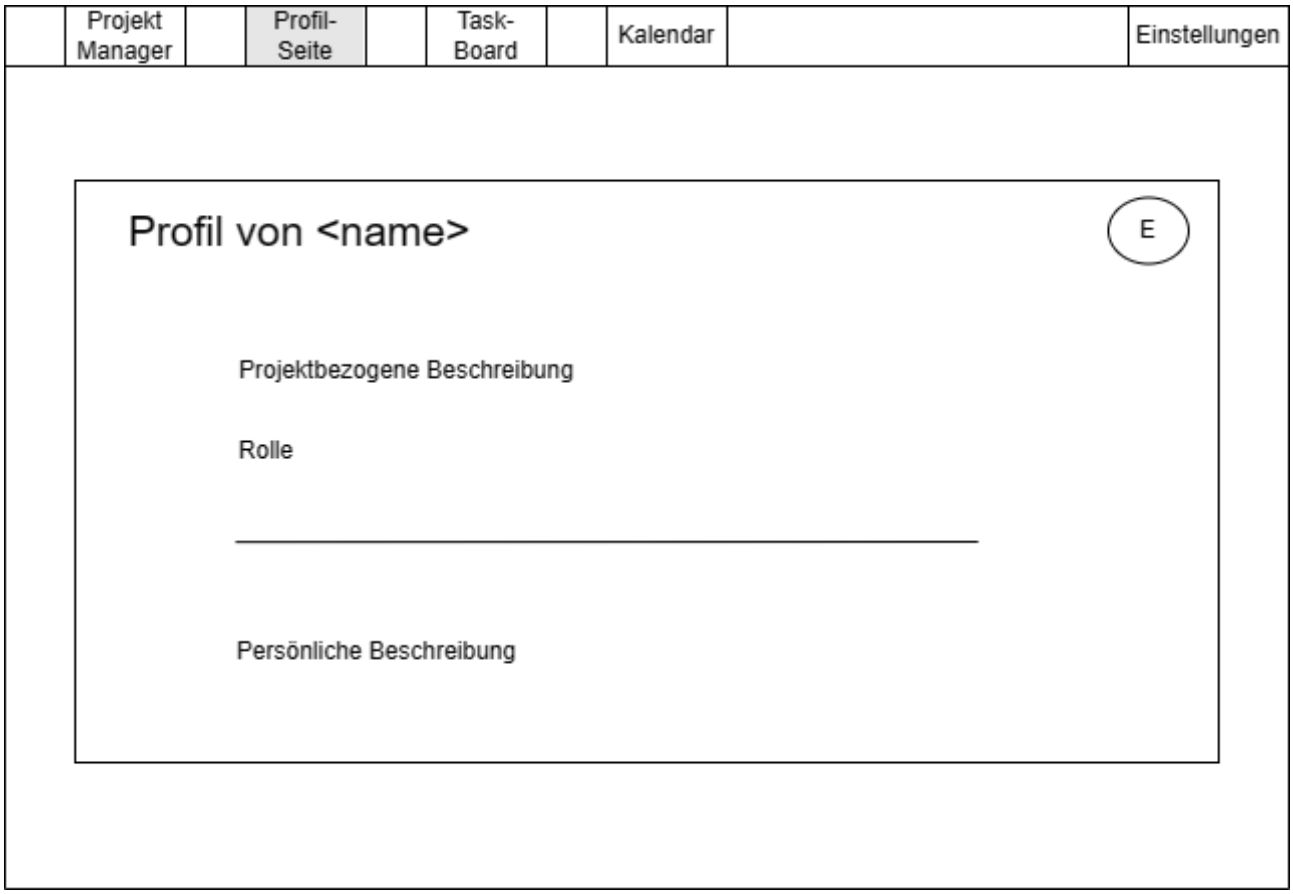
Neben der Überarbeitung des Login-Prozesses wird das Verhalten von allen anderen Funktionen des Servers, abgesehen von Registrieren, Passwort vergessen und neues Passwort anfordern, durch das Validieren der SessionID vor Funktionsaufruf erweitert. Hierbei wird die Gültigkeit der übermittelten SessionID über ihre Existenz in der Datenbank und TTL bestimmt. Bei gültiger SessionID wird der gefragte Funktionsaufruf gestartet und als in dem Sequenzdiagramm 8 als “valid_Session_Responce“ Repräsentierter Responce zurückgegeben.
Bei einer ungültigen SessionID wird ein “invalid_Session_Responce“ übermittelt, welcher in der Regel zu einem Weiterleiten an die Login-Page führt.

Abbildung 9: Authority Check Sequenzdiagramm

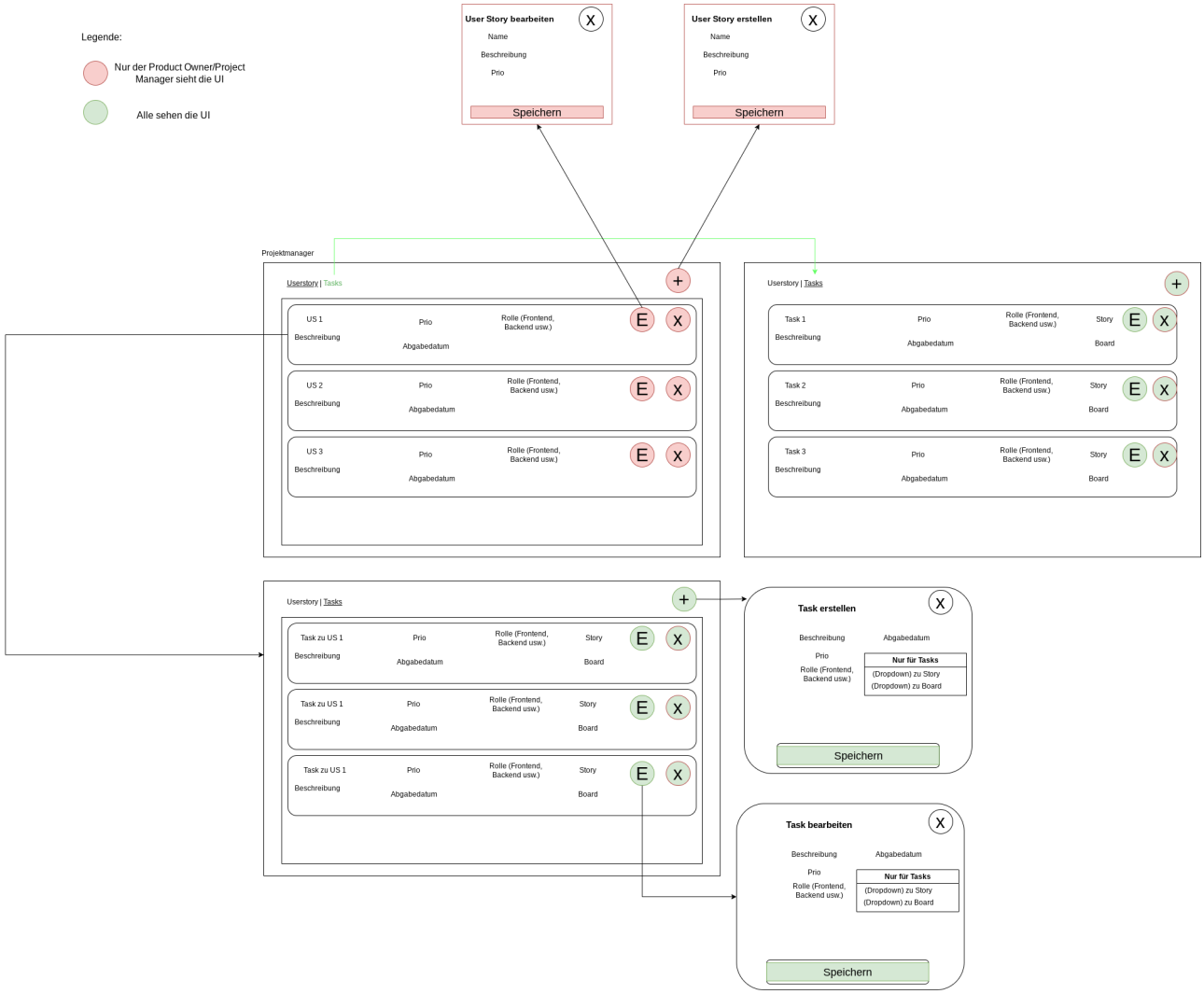


Funktionen, für welche Nutzer ein bestimmtes Autoritäts-Level benötigen, werden nach dem SessionID-Check (siehe Abbildung 8) mit einem Authority-Check erweitert. Somit wird nach einem positiven SessionID-Check mittels der SessionID das Autoritäts-Level des Nutzers ermittelt. Sollten in diesem Prozess Errors auftauchen oder ein Check nicht erfolgreich sein, wird ein “invalid_Request_Responce“ beziehungsweise “invalid_Session_Responce“ übermittelt, welches die Funktion abbricht, ohne die gefragte Funktion auszuführen. Bei erfolgreichem Check repräsentiert “valid_Request_Responce“ in Sequenzdiagramm 9 den Responce der darauffolgend ausgeführten gefragten Funktion.

2.4 Prototyp für Projekt-Manager und Profil



Für die Umsetzung des geplanten Profils wurde der obige Prototyp erstellt. Dieser umfasst den Namen des Nutzers, eine projektbezogene Beschreibung, eine für ihn zugewiesene visuelle Rolle sowie eine persönliche Beschreibung. Diese Merkmale lassen sich beliebig von dem Benutzer selbst editieren, indem oben rechts auf die Bearbeitungsschaltfläche gedrückt wird. Dies öffnet ein Bearbeitungsmenü, indem diese Eigenschaften beliebig anpassbar sind. Lediglich die Rolle ist nicht von dem Benutzer selbst bearbeitbar, da dies vom Projektleiter zugewiesen wird. Von dem Profil aus ist es möglich, im Reiter den "Project Manager"Knopf zu betätigen, um auf die nächste Seite zu gelangen.



Der neue Project Manager ist nun in einer Tabellenform. Durch Klicken auf Userstory oder Tasks (T1.F1) im äußeren Container gelangt man zu den verschiedenen Anzeigen, in denen alle Userstories und oder alle Tasks angezeigt werden in Kärtchenform. Dies wurde aus diesem Grund so umgesetzt, da es einfach einen festen Rahmen für die Objekte gibt. Zuvor (im Sprint 1) waren beispielsweise die Userstories recht “lose“ auf der Seite verteilt.

Drückt man innerhalb der Anzeige auf Userstory, so werden alle bisher angelegten Userstories angezeigt. Ist keine vorhanden, so lässt sich über den Button zur Userstory-Erstellung eine erstellen. Die dabei einzugebenden Parameter sind Name, Priorität, Beschreibung, als auch ein Abgabedatum. Das beim Klicken des Buttons aufkommende Pop-up-Fenster ist dabei vom ersten Sprint recycelt worden, bloß nun als zentriertes Pop-up und kein Seitenfenster. Innerhalb des Kärtchens wird außerdem die Möglichkeit geboten, durch einen Editier-Button die Userstory anzupassen (ebenfalls vom ersten Sprint recyceltes Fenster) und oder zu löschen (U6.F1).

Wechselt man durch Klicken auf Tasks zur Task-Ansicht, so werden alle angelegten Tasks angezeigt. Hier existieren alle eben genannten Optionen wie für Userstory. Außerdem existieren zwei neue Parameter. Ein Verweis zur Story, also zu welcher Userstory die Tasks gruppiert ist, als auch zum Board, also die Position der Task innerhalb des Task-Boards. Diese lassen sich im Bearbeitungsfenster der Tasks ebenfalls anpassen. Diese neuen Parameter sind essenziell, um die Tasks einordnen zu können.

Möchte man allerdings nicht alle Tasks gesondert, sondern nur Tasks zu einer spezifischen Userstory ansehen, so klickt man auf eine Userstory innerhalb der Userstory-Ansicht und ist nun in einer Subsection der Userstory, bei dem alle Tasks zu einer Userstory angezeigt werden. Einziger optionaler Unterschied zur Ansicht der Tasks ist, dass sich Tasks zur Userstory erstellen lassen (T3.F1, T4.F1, T5.F1). Diese werden dann auch ebenfalls in der Task-Ansicht angezeigt. Dies haben wir aufgrund dessen so umgesetzt, um es einfacher zu machen, bestimmte Tasks, die von Interesse sind für den jeweiligen Nutzer, anzusehen.

3 Implementation und Tests

3.1 Tests

Aufgrund der Menge an geschriebenen Tests und durch die klare Abtrennung der zu testenden Bereiche, wurde sich entschieden, die Tests nicht nur in automatische und manuelle Tests aufzuteilen, sondern die Tests selbst nochmals zu gruppieren. Somit erhalten wir einen gewissen Grad an Übersichtlichkeit sowie die Möglichkeit ganze Teile des Programms zu testen, ohne alle betroffenen Tests einzeln oder gleich alle Tests durchlaufen lassen zu müssen.
Darüber hinaus wurde sich für diesen Sprint entschieden, die Ergebnisse der Tests automatisch in ein entsprechendes Textdokument zu schreiben, um zukünftige Dokumentierung einfacher zu gestalten.

3.1.1 HTTP-Request-Tests

Die sogenannten HTTP-Request-Tests umfassen alle Tests, die sich mit der Kommunikation zwischen dem Frontend, beziehungsweise der Webanwendung, und der Logik unserer Software beschäftigt. Wie der Name impliziert wird hier die Umsetzung der HTTP Post- und Get-Requests sowie die zugehörigen Responces durch den Server getestet, um Format- und Parameterfehler, wie fehlende Argumente, schon vor der Integrierung in das Frontend festzustellen.

Neu Implementierte Tests:

TestArt.TestID	Erstellungszeit	Herkunft	Author
HTTP.T6	12.04.2024	Grobentwurf	Lucas Krüger
HTTP.T7	17.04.2024	Grobentwurf	Lucas Krüger
HTTP.T8	17.04.2024	Grobentwurf	Lucas Krüger
HTTP.T9	17.04.2024	Grobentwurf	Lucas Krüger
HTTP.T10	17.04.2024	Grobentwurf	Lucas Krüger
HTTP.T11	17.04.2024	Grobentwurf	Lucas Krüger
HTTP.T12	20.04.2024	Grobentwurf	Lucas Krüger
HTTP.T13	20.04.2024	Grobentwurf	Lucas Krüger
HTTP.T14	20.04.2024	Grobentwurf	Lucas Krüger
HTTP.T15	20.04.2024	Grobentwurf	Lucas Krüger
HTTP.T16	20.04.2024	Grobentwurf	Lucas Krüger
HTTP.T17	20.04.2024	Grobentwurf	Lucas Krüger

Testlog für HTTP-Request-Tests unter:
Project/src/test/java/com/team3/project/logs/log_HttpRequestTest.txt

3.1.2 Logic-Tests

Die Logic-Tests prüfen die Methoden, die der Webcontroller für die Verarbeitung der Requests des Nutzers benötigt. Es wird geprüft, dass die Methoden sowohl die gewünschte Funktionalität umsetzt, als auch keine ungewollten Seiteneffekte hervorruft, damit die Datenbank im erwarteten Zustand bleibt beziehungsweise keine Inkonsistenz auftritt.
Neu Implementierte Tests:

TestArt.TestID	Erstellungszeit	Herkunft	Author
Logic.T3	13.04.2024	Task P1.B1	Henry Lewis Freyschmidt
Logic.T4	13.04.2024	Task P2.B1	Henry Lewis Freyschmidt
Logic.T5	15.04.2024	Task U6.B1	Henry Lewis Freyschmidt
Logic.T6	18.04.2024	Task T3.B1	Henry Lewis Freyschmidt
Logic.T7	19.04.2024	Task T4.B1	Henry Lewis Freyschmidt
Logic.T8	21.04.2024	Task T5.B1	Henry Lewis Freyschmidt

Testlog für Logic-Tests unter:
Project/src/test/java/com/team3/project/logs/log_LogicTest.txt

3.1.3 Datenbank-Tests

Die Datenbank-Tests prüfen die Funktionalitäten fürs laden und schreiben von Daten aus der Datenbank.
Neu Implementierte Tests:

TestArt.TestID	Erstellungszeit	Herkunft	Author
DB.U.T1	18.04.2024	Grobentwurf	Marvin Prüger
DB.U.T2	18.04.2024	Grobentwurf	Marvin Prüger
DB.U.T3	18.04.2024	Grobentwurf	Marvin Prüger
DB.U.T4	18.04.2024	Grobentwurf	Marvin Prüger
DB.U.T5	18.04.2024	Grobentwurf	Marvin Prüger
DB.U.T6	18.04.2024	Grobentwurf	Marvin Prüger
DB.T.T1	18.04.2024	Grobentwurf	Marvin Prüger
DB.T.T2	18.04.2024	Grobentwurf	Marvin Prüger

Testlog für Datenbank-Tests unter:
Project/src/test/java/com/team3/project/logs/log_DatabaseTest.txt

3.2 Laufender Prototyp

Im laufenden Prototypen wurde zunächst die Login-Seite erweitert. Möchte man nun sein Passwort zurücksetzen, so erscheinen zwei neue Fenster für diese Funktion. In dem ersten gibt man seine E-mail an, im zweiten den Sicherheitscode. Somit wird User-Story A4 teilweise im visuellen Sinne erfüllt. Es wurde zwar bereits eine Funktion für die Generierung eines Sicherheitscodes erstellt, jedoch wird keine eigentliche E-mail zur Authentifizierung des Benutzers gesendet. Zur Visualisierung das Zurücksetzen des eigenen Passworts kann man allerdings trotzdem seine Daten eingeben, und eine kleine Eingabebestätigung in Form eines kleinen Pop-Ups erhalten. Dies erfüllt die geplante User Story A4 und A5 visuell. Ist der Anmeldevorgang erfolgreich, so wird dem Benutzer wie im zu vorigen Sprint der Project Manager angezeigt.

Im Project Manager angelangt sieht man die Anzeige für die erstellten Userstories und Tasks. Durch den innerhalb der Kärtchen verankerten “x“-Button lässt sich eine Userstory schließen. Dies erfüllt Anforderung U6.F1. Wenn man jeweils auf die verschiedenen Anzeigen “Userstory“ oder “Task“ klickt, so bekommt man entweder alle Userstories oder Tasks angezeigt. Dies erfüllt Anforderung T1.F1. Will man innerhalb der Task-Anzeige eine neue Task erstellen, so kann man entweder in der Ansicht aller Tasks oder bei den Tasks zu den Userstories dies vornehmen. Die im Erstellungs- und Bearbeitungsmenu aufgeführten Parameter sind Beschreibung, Priorität, Titel, Abgabedatum und die zuweisende

User-Story. Dies erfüllt Task T3.F1 und T4.F1. Selbe Löschung von Tasks wie bei Userstories kann man innerhalb der Kärtchen über den Button vornehmen. Dies erfüllt Task T5.F1. Zuallerletzt ist es möglich, bei der Task-Bearbeitung die User-Story anzupassen, sodass jede Task nun zu einer User-Story gehört. Führt man einen Doppelklick auf eine User-Story aus, sieht man dann die zugehörigen Tasks, wenn welche vorhanden sind. Dies erfüllt die Verlinkung für User-Story T3.

Von dem Project Manager aus ist es dem Benutzer möglich, auf die Profil-Seite zu wechseln, indem oben im Reiter die „"Profil“ Schaltfläche gedrückt wird. Hier wird für den Benutzer nun sein eigenes Profil angezeigt, welches von ihm selbst bearbeitet werden kann. Die Anzeige erfüllt die User-Story P1. Für die Bearbeitung kann die Bearbeitungsschaltfläche betätigt werden, sodass ein neues Bearbeitungsfenster erscheint. In dem Bearbeitungsfenster sieht man Eingabezeilen für den Namen und den zwei Beschreibungen. Dies erfüllt die User-Story P2.

3.3 Abweichungen von Sprintplanung

Es wurden in diesem Sprint die geplanten User Stories zum größten Teil umgesetzt. Zum einen existieren nun die Fenster für die Zurücksetzung des Passworts und der anschließenden Eingabe des Sicherheitscodes. Zwar existieren diese Fenster visuell, jedoch wurden noch keine geplanten Funktionalitäten dazu implementiert. Dies lässt sich damit begründen, dass der Umfang des Arbeitens und Versendens von E-Mails unterschätzt wurde, sodass der Ansatz verworfen wird. Nun versuchen wir, die Passwort-Zurücksetzung trotzdem zu implementieren, jedoch ohne die Versendung einer E-Mail. Aufgrund der Zeit wird dies allerdings im nächsten Sprint angesetzt. Vorerst existiert der Ansatz erstmal rein visuell mit einer Benachrichtigung, dass eine E-Mail abgesendet wurde.
Aufgrund dessen, dass das Task-Board noch nicht implementiert wurde, existieren ebenfalls noch nicht die Möglichkeiten, eine User-Story zu einem Task-Board zuzuweisen. Dies wird ebenfalls im nächsten Sprint vorgesehen, da die Dropdown-Menüs auch noch nicht im Frontend erstellt wurden.
Zuallerletzt existiert eine Abweichung im Frontend des Projekt-Managers. Es existiert bereits eine Drag-and-Drop-Funktion zwischen des User-Stories, welche den Positionswechsel zwischen User-Stories ermöglicht. Diese Funktionalität war allerdings erst für das Task-Board geplant, sodass wir nun eine zusätzliche Funktionalität haben.

3.3.1 Datenbank-Auswertung

Es wurde ein Refactoring des Codes vorgenommen, bei der der Fokus auf die Vereinheitlichung und auf die Reduzierung von Redundanzen lag. Diese Optimierung beanspruchte leider mehr Zeit als initial angesetzt, da durch neue Erkenntnissen aus etlichen Recherchen die vorangestellten Optimierungen verbessert werden konnte und manche obsolet wurden. Ein kleines Refactoring war vorgesehen, doch sprengte dieses den Rahmen der angesetzten Zeit für diesen Sprints. So kam man leicht in Verzug, wodurch andere Teams auf die Commits des Dankenbank-Teams warten mussten. Durch die Optimierung des Refactoring können zukünfitge Aufgaben schneller umgesetzt werden, was sich während des Sprints schon des Öfteren ausgezahlt machte.
Weiterhin erfolgte auch die Umstellung von Hibernate-Spezifischer Nutzung zu JPA konform, welches die Möglichkeit bietet Hibernate Alternativen leicht einzubinden, doch ist der eigentliche Nutzen dieses Features fragwürdig. Hibernate selber bietet die Möglichkeit andere Datenbanken zu nutzen und das Ersetzen von Hibernate durch ein anderes ORM war, weder in den Anforderungen vorgesehen, noch ist es für die Zukunft geplant. Die eigentlichen Anpassungen hielten sich durch die Optimierung des Refactorings jedoch in Grenzen, sodass diese Änderung in rasanter Geschwindigkeit umgesetzt werden konnte.
Die Datenbank wurde minimal angepasst, um die derzeitigen Anforderungen umzusetzen und unkomplizierte Arbeit mit den jeweiligen Tabellen zu gewährleisten. So wurden unter anderen "NOT NULL"-Contraints entfernt, um die Tasks schon gänzlich zu implementieren, ohne jedoch alle Funktionalitäten der TaskList zu brauchen.

Die Integrierung der eigentlichen Aufgaben des zweiten Sprintes wurden erfolgreich abgeschlossen. Durch die Komplexität der Verhältnisse einzelner Tabellen zueinander in der Datenbank sind weitere Recherchen nötig. Es wurde über die eigentlichen Anforderungen des Sprints hinausgearbeitet, wobei die Fertigstellung der eigentlichen Aufgaben des Sprints vernachlässigt wurden. Dies ist einerseits der Komplexität, sowie den notwendigen Tests geschuldet, andererseits kam es dazu durch eine Misskommunikation, bei der Aufgaben aus dem Sprint aufgrund des Zeitmanagements gestrichen bzw. in den nächsten Sprint verschoben wurden.
Abschließend lässt sich sagen, dass alle Aufgaben des zweiten Sprintes umgesetzt wurden, sowie eine gute Grundlage für die Abarbeitung von neuen Aufgaben in zukünftigen Sprints geschaffen wurde, somit kann aus Datenbank-Team-Sicht der Sprint als erfolgreich bewertet werden.