Sprint 4

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)
Henry J. v. Rooyen (HvR)

18. Juli 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	HLF	SZ	PK
Klassendiagramme	SZ	LK, HLF	SZ	HLF, PK
Designpattern	LK	LK	LK	HLF, PK
Verhalten	LK	LK	LK	HLF, PK
Prototyp Frontend	PK	PK	ZSH, PK	-
Implementation und Tests				
HTTP-Request-Tests	LK	LK	-	-
Datenbank-Tests	TMS, MOP	TMS, MOP	-	-
Logic-Tests	HLF	HLF	-	HvR
Abnahmetest.T4	ZSH	ZSH	-	PK
Tracing - Komponenten im Code	LK	LK	-	-
Tracing - Beendeter Aufgaben	LK	LK	-	-
Laufender Prototyp	PK	PK		-
Abweichung von Sprintplanung	PK	PK	-	-

Task	Kurzbeschreibung	Name	Anteil%
R2.F1	Erstelle Anzeige für Rollen	PK	100%
R2.F2	Erstelle Funktionalität für Rollen	PK	100%
R2.F3	Erstellen vis. Rollen	LK	100%
R2.B1	Erstelle Backend-Funktionalität zum Ausgeben aller zu einer rechtebezogenen Rolle verlinkten visuellen Rollen, welche R2.D2 aufruft und die visuellen Rollen ausgibt.	HvR	100%
R2.B2	Erstelle Backend-Funktionalität zum Erstellen visueller Rollen, welche R2.D3 aufruft	HvR	100%
R2.D1	Erstelle Speicher für visuelle Rollen	MOP TMS	50% 50%
R2.D2	Erstelle Funktionalität zur Ausgabe visuelle Rollen	MOP TMS	50% 50%
R2.D3	Erstelle Funktionalität zum Erstellen visueller Rollen	MOP TMS	50% 50%
R3.F1	Visuelle Rolle umbenennen Anzeige	PK	%
R3.F2	Anpassen vis. Rollen	LK	100%
R3.B1	Erstelle Backend-Funktionalität zum Anpassen visueller Rollen, welche R3.D1 aufruft	HvR	100%
R3.D1	Erstelle Funktionalität zum Umbenennen visueller Rollen	MOP TMS	50% 50%
R4.F1	Anzeige für Löschen visueller Rollen	PK	100%
R4.F2	Löschen vis. Rollen	LK	100%
R4.B1	Erstelle Backend-Funktionalität zum Löschen visueller Rollen, welche R4.D1 aufruft	HvR	100%
R4.D1	Erstelle Funktionalität zum Löschen visueller Rollen	MOP TMS	50% 50%
R5.F1	Anzeige für Rollen Anzeige visueller Rollen	PK	100%
R5.F2	Anpassen Profil um vis. Rollen	LK	100%
R5.F3	Auslesen gewählter vis. Rollen	LK	100%
R5.B1	Ausgeben Profil mit vis. Rollen	LK	100%
R5.B2	Anpassen Profil um vis Rollen Backend	LK	100%
R5.D1	Erweitere Speicher für Nutzer um visuelle Rollen	MOP TMS	50% 50%
R5.D2	Erstelle Funktionalität zum Zuweisen visueller Rollen für Nutzer	MOP TMS	50% 50%
U7.F1	Erstelle Frontend-Funktionalität zum Anzeigen aller User-Storys	ZSH	100%
T14.F1	Erstelle Frontend-Funktionalität zum Anzeigen aller Tasks	ZSH	100 %
TB5.F1	Erweitere Anzeige für Task-Boards	ZSH	100 %
S1	Synchronisation	LK	100%
	Logic-Tests	HLF HvR	80% 20%
	DB-Tests	MOP TMS	50% 50%
	Product Owner	SZ	100%
	Anlegen und Verwalten von Tasks	SZ	100%
	Anlegen des Prozessdokumentes	SZ	100%
	Anlegen/Halten der Presentation	LK PK SZ HLF	30% 30% 30% 10%

Inhaltsverzeichnis

1 Backlog **2**
1.1 Änderungen am Backlog . . . 2
1.2 Planung des Sprints . . . 3
1.2.1 [R] Rolle . . . 3
1.2.2 [U] User-Story . . . 6
1.2.3 [T] Task . . . 6
1.2.4 [TB] Task-Board . . . 7
1.2.5 [S] System . . . 7
1.2.6 Ablaufplan zu User-Storys und Tasks . . . 8

2 Feinentwurf **9**
2.1 Komponentendiagramm . . . 9
2.2 Klassendiagramme und Design Pattern . . . 10
2.2.1 Klassendiagramm zum Backlog-Abschnitt Role . . . 10
2.2.2 Klassendiagramm zum Backlog-Abschnitt User-Story und Task . . . 11
2.2.3 Klassendiagramm zum Backlog-Abschnitt Task-Board . . . 11
2.2.4 Klassendiagramm zum Backlog-Abschnitt System . . . 12
2.3 Designpattern . . . 13
2.4 Neues/verändertes Verhalten . . . 14
2.5 Prototypen Frontend . . . 16

3 Implementation und Tests **16**
3.1 Tests . . . 16
3.1.1 HTTP-Request-Test . . . 16
3.1.2 Datenbank-Test . . . 17
3.1.3 Logic-Test . . . 17
3.2 Tracing - Komponenten im Code . . . 18
3.3 Tracing - Beendete Aufgaben . . . 19
3.4 Laufender Prototyp . . . 20

4 Abweichungen von Sprintplanung **21**

5 Technologien **22**

1 Backlog

1.1 Änderungen am Backlog

Element	von	zu
T18		Als Nutzer kann ich Tasks danach filtern, ob sie fertig sind, um nur Tasks, die fertig sind, zu finden.
T19		Als Nutzer kann ich Tasks danach filtern, ob sie unfertig sind, um nur Tasks, die unfertig sind, zu finden.
T20		Als Nutzer kann ich Tasks nach ihrer Aufwandsschätzung filtern, um nur Tasks, deren Aufwandsschätzungen in einem ausgewählten Intervall liegen, zu finden.
T21		Als Nutzer kann ich Tasks nach ihrer Bearbeitungszeit filtern, um nur Tasks, deren Bearbeitungszeiten in einem ausgewählten Intervall liegen, zu finden.

Tabelle 1: Erwartete Filterfunktionen zu den neuen Attributen von Tasks hinzugefügt.

Element	von	zu
R5	Als Nutzer kann ich mich auf meinem Profil einer visuellen Rolle zuordnen, damit diese auf meinem Profil sichtbar ist.	Als Nutzer kann ich mich den visuellen Rollen zu meiner rechtebezogenen Rolle zuordnen, damit diese auf meinem Profil sichtbar sind.

Tabelle 2: Ein Nutzer soll mehrere visuelle Rollen haben dürfen.

Element	von	zu
K3	Als Nutzer kann ich Meetings mit einem Namen, einem Zeitraum, einem Ort, einer Art und einer Auflistung der betroffenen Nutzer beziehungsweise Rollen erstellen, um Meetings zu planen.	Als Manager oder Product Owner kann ich Meetings mit einem Namen, einem Zeitraum, einem Ort, einer Art und einer Auflistung der betroffenen Nutzer beziehungsweise Rollen erstellen, um Meetings zu planen.
K5	Als Nutzer kann ich Sprints mit einem Namen, einem Zeitraum und einer Auflistung der betroffenen Nutzer beziehungsweise Rollen erstellen, um Sprints zu planen.	Als Product Owner kann ich Sprints mit einem Namen, einem Zeitraum und einer Auflistung der betroffenen Nutzer beziehungsweise Rollen erstellen, um Sprints zu planen.

Tabelle 3: Meetings und Sprints sollen nicht von beliebigen Nutzern erstellt werden können.

Element	von	zu
T15	Als Nutzer kann ich Tasks nach deren Abgabefrist filtern, um nur Tasks, deren Abgabefristen in einem ausgewählten Intervall liegen, zu finden.	Als Nutzer kann ich Tasks nach ihrer Abgabefrist filtern, um nur Tasks, deren Abgabefristen in einem ausgewählten Intervall liegen, zu finden.
TB5.F1	Erweitere TB2.F1, sodass zu jeder Task die Liste zugeordneter Nutzer eingesehen werden kann.	Erweitere Anzeige für Task-Boards, sodass zu jeder Task die Liste zugeordneter Nutzer eingesehen werden kann.

Tabelle 4: Die Änderungen sind Korrekturen oder sprachliche Anpassungen.

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, HvR
- Daten-Entwicklung: MOP, TMS

In diesem Sprint werden folgende inhaltliche Komplexe angegangen:

1. die Umsetzung visueller Rollen, sodass diese angepasst werden können und Nutzer sich ihnen zuordnen können,
2. die Einführung von Filter-Funktionen, sodass sich die Suche nach Tasks und User-Storys vereinfacht und
3. die Synchronisation von Daten, sodass Sichten auf das System aktuell gehalten werden.

1.2.1 [R] Rolle

⊗ **R2** Als Manager kann ich zu einer rechtebezogenen Rolle eine visuelle Rolle erstellen, damit Nutzer sich anhand dieser einordnen können.

- ⊗ **R2.F1** Erstelle Anzeige für Rollen, welche die Auswahl einer rechtebezogenen Rolle ermöglicht, zu dieser alle verlinkten visuellen Rollen anzeigen kann und die Eingabe einer visuellen Rolle ermöglicht.
  - Aufwandsschätzung: 1h durch PK
  - Bearbeitung: 2h durch PK
  - Review: 0,25 durch ZSH, PK
  - Test: 0,25 durch ZSH, PK
- ⊗ **R2.F2** Erstelle Frontend-Funktionalität zum Anzeigen von R2.F1, welche, wenn ein Nutzer mit gültiger Session sie nutzt, R2.B1 aufruft und R2.F1 anzeigt.
  - Aufwandsschätzung: 0,25h durch PK
  - Bearbeitung: 0,25h durch PK
  - Review: 0,25h durch ZSH, PK
  - Test: 0,25h durch ZSH, PK

- ⊗ **R2.F3** Erstelle Frontend-Funktionalität zum Erstellen visueller Rollen, welche, wenn ein Manager mit gültiger Session sie nutzt, R2.F1 ausliest und R2.B2 aufruft.
  - Aufwandsschätzung: 0,5h durch PK
  - Bearbeitung: 0,5h durch LK
  - Review: 0,25h durch ZSH
  - Test: 0,25h durch LK
- ⊗ **R2.B1** Erstelle Backend-Funktionalität zum Ausgeben aller zu einer rechtebezogenen Rolle verlinkten visuellen Rollen, welche R2.D2 aufruft und die visuellen Rollen ausgibt.
  - Aufwandsschätzung: 0,5h
  - Bearbeitung: 0,5h durch HvR
  - Test: 0,25h durch HvR
- ⊗ **R2.B2** Erstelle Backend-Funktionalität zum Erstellen visueller Rollen, welche R2.D3 aufruft.
  - Aufwandsschätzung: 0,5h
  - Bearbeitung: 0,5h durch HvR
  - Test: 0,25h durch HvR
- ⊗ **R2.D1** Erstelle Speicher für visuelle Rollen.
  - Aufwandsschätzung: 0,5h
  - Bearbeitung: 0,5h durch TMS, MOP
  - Test: 0,5h durch TMS, MOP
- ⊗ **R2.D2** Erstelle Daten-Funktionalität zum Ausgeben aller zu einer rechtebezogenen Rolle verlinkten visuellen Rollen.
  - Aufwandsschätzung: 0,75h
  - Bearbeitung: 0,5h durch TMS, MOP
  - Test: 0,5h durch TMS, MOP
- ⊗ **R2.D3** Erstelle Daten-Funktionalität zum Erstellen visueller Rollen, welche zu einer rechtebezogenen Rolle eine visuelle Rolle erstellt.
  - Aufwandsschätzung: 0,5h
  - Bearbeitung: 0,5h durch TMS, MOP
  - Test: 0,5h durch TMS, MOP

⊗ **R3** Als Manager kann ich eine visuelle Rolle umbenennen, um sie anzupassen.

- ⊗ **R3.F1** Erweitere R2.F1, sodass angezeigte visuelle Rollen umbenannt werden können.
  - Aufwandsschätzung: 0,5h
  - Bearbeitung: 0,5h durch PK
  - Review: 0,25h durch ZSH
  - Test: 0,25h durch ZSH, PK
- ⊗ **R3.F2** Erstelle Frontend-Funktionalität zum Anpassen visueller Rollen, welche, wenn ein Manager mit gültiger Session sie nutzt, R3.F1 ausliest und R3.B1 aufruft.
  - Aufwandsschätzung: 0,25h
  - Bearbeitung: 0,25h durch LK
  - Review: 0,25h durch PK
  - Test: 0,25h durch LK
- ⊗ **R3.B1** Erstelle Backend-Funktionalität zum Anpassen visueller Rollen, welche R3.D1 aufruft.

    - Aufwandsschätzung: 0,25h
    - Bearbeitung: 0,25h durch HvR
    - Test: 0,25h durch HvR
  - ⊗ **R3.D1** Erstelle Daten-Funktionalität zum Umbenennen visueller Rollen, welche den Namen einer bestehenden visuellen Rolle anpasst.
    - Aufwandsschätzung: 0,25h
    - Bearbeitung: 0,25h durch TMS, MOP
    - Test: 0,5h durch TMS, MOP

- ⊗ **R4** Als Manager kann ich eine visuelle Rolle löschen, damit Nutzer sich dieser nicht mehr zuordnen können.
  - ⊗ **R4.F1** Erweitere R2.F1, sodass angezeigte visuelle Rollen gelöscht werden können.
    - Aufwandsschätzung: 0,25h
    - Bearbeitung: 0,25h durch PK
    - Review: 0,25h durch PK, ZSH
    - Test: 0,25h durch PK, ZSH
  - ⊗ **R4.F2** Erstelle Frontend-Funktionalität zum Löschen visueller Rollen, welche, wenn ein Manager mit gültiger Session sie nutzt, R4.F1 ausliest und R4.B1 aufruft.
    - Aufwandsschätzung: 0,25h
    - Bearbeitung: 0,25h durch LK
    - Review: 0,25h durch PK
    - Test: 0,25h durch LK
  - ⊗ **R4.B1** Erstelle Backend-Funktionalität zum Löschen visueller Rollen, welche R4.D1 aufruft.
    - Aufwandsschätzung: 0,25h
    - Bearbeitung: 0,25h durch HvR
    - Test: 0,25h durch HvR
  - ⊗ **R4.D1** Erstelle Daten-Funktionalität zum Löschen visueller Rollen, welche eine visuelle Rolle löscht.
    - Aufwandsschätzung: 0,25h
    - Bearbeitung: 0,25h durch TMS, MOP
    - Test: 0,5h durch TMS, MOP

- ⊗ **R5** Als Nutzer kann ich mich den visuellen Rollen zu meiner rechtebezogenen Rolle zuordnen, damit diese auf meinem Profil sichtbar sind.
  - ⊗ **R5.F1** Erweitere Anzeige für Profile um visuelle Rollen und einer Auswahl für visuelle Rollen.
    - Aufwandsschätzung: 0,5h durch PK
    - Bearbeitung: 0,25h durch PK
    - Review: 0,25h durch ZSH
    - Test: 0,25h durch ZSH
  - ⊗ **R5.F2** Erweitere Frontend-Funktionalität zum Anzeigen eines Profils um die visuellen Rollen.
    - Aufwandsschätzung: 1h durch PK
    - Bearbeitung: 0,25h durch LK

    - Review: 0,25h durch PK, ZSH
    - Test: 0,25h durch LK
- ⊗ **R5.F3** Erweitere Frontend-Funktionalität zum Anpassen des Profils, sodass diese die ausgewählten visuellen Rollen von R5.F1 ausliest und R5.B2 aufruft.
    - Aufwandsschätzung: 1h durch PK
    - Bearbeitung: 0,25h durch LK
    - Review: 0,25h durch PK
    - Test: 0,25h durch LK
- ⊗ **R5.B1** Erweitere Backend-Funktionalität zur Ausgabe eines Profils, sodass diese die visuellen Rollen des Nutzers ausgibt, R2.D2 mit der rechtebezogenen Rolle des Nutzers aufruft und auch die davon erhaltenen visuellen Rollen ausgibt.
    - Aufwandsschätzung: 1h
    - Bearbeitung: 1h durch LK
    - Test: 0,25 durch LK
- ⊗ **R5.B2** Erweitere Backend-Funktionalität zum Anpassen des Profils, sodass diese R5.D2 aufrufen kann.
    - Aufwandsschätzung: 0,5h
    - Bearbeitung: 0,5h durch LK
    - Test: 0,25h durch LK
- ⊗ **R5.D1** Erweitere Speicher für Nutzer um visuelle Rollen.
    - Aufwandsschätzung: 0,25h
    - Bearbeitung: 0,25h durch TMS, MOP
    - Test: 0,5h durch TMS, MOP
- ⊗ **R5.D2** Erstelle Daten-Funktionalität zum Anpassen der visuellen Rollen eines Nutzers.
    - Aufwandsschätzung: 5h
    - Bearbeitung: 4h durch TMS, MOP
    - Test: 0,5h durch TMS, MOP

1.2.2 [U] User-Story

- ⊗ **U7** Als Nutzer kann ich User-Storys nach der Priorität (Urgent > High > Normal > Low) filtern, um nur User-Storys ausgewählter Prioritäten anzuzeigen.
    - ⊗ **U7.F1** Erstelle Frontend-Funktionalität zum Anzeigen aller User-Storys, sodass es möglich ist, nur User-Storys mit bestimmten Prioritäten anzuzeigen.
        - Aufwandsschätzung: 1h durch ZSH
        - Bearbeitung: 1h durch ZSH
        - Review: 0,25h durch ZSH, PK
        - Test: 0,25h durch ZSH, PK

1.2.3 [T] Task

- ⊗ **T14** Als Nutzer kann ich Tasks nach der Priorität (Urgent > High > Normal > Low) filtern, um nur Tasks ausgewählter Prioritäten anzuzeigen.
    - ⊗ **T14.F1** Erstelle Frontend-Funktionalität zum Anzeigen aller Tasks, sodass es möglich ist, nur Tasks mit bestimmten Prioritäten anzuzeigen.
        - Aufwandsschätzung: 0.5h durch ZSH

- Bearbeitung: 0,25h durch ZSH
- Review: 0,25h durch ZSH, PK
- Test: 0,25h durch ZSH, PK

1.2.4 [TB] Task-Board

⊗ **TB5** Als Nutzer kann ich auf einem Task-Board einsehen, welcher Task welche Nutzer zugeordnet sind, um die Verteilung von Tasks zu erleichtern.

⊗ **TB5.F1** Erweitere Anzeige für Task-Boards, sodass zu jeder Task die Liste zugeordneter Nutzer eingesehen werden kann.

- Aufwandsschätzung: 2h durch PK,ZSH
- Bearbeitung: 1h durch PK
- Review: 0,25h durch PK, ZSH
- Test: 0,25h durch PK, ZSH

1.2.5 [S] System

⊗ **S1** Als Nutzer kann ich innerhalb von 2 Sekunden neue Angaben aller anderen Nutzer im System sehen, um immer auf dem aktuellsten Stand zu sein.

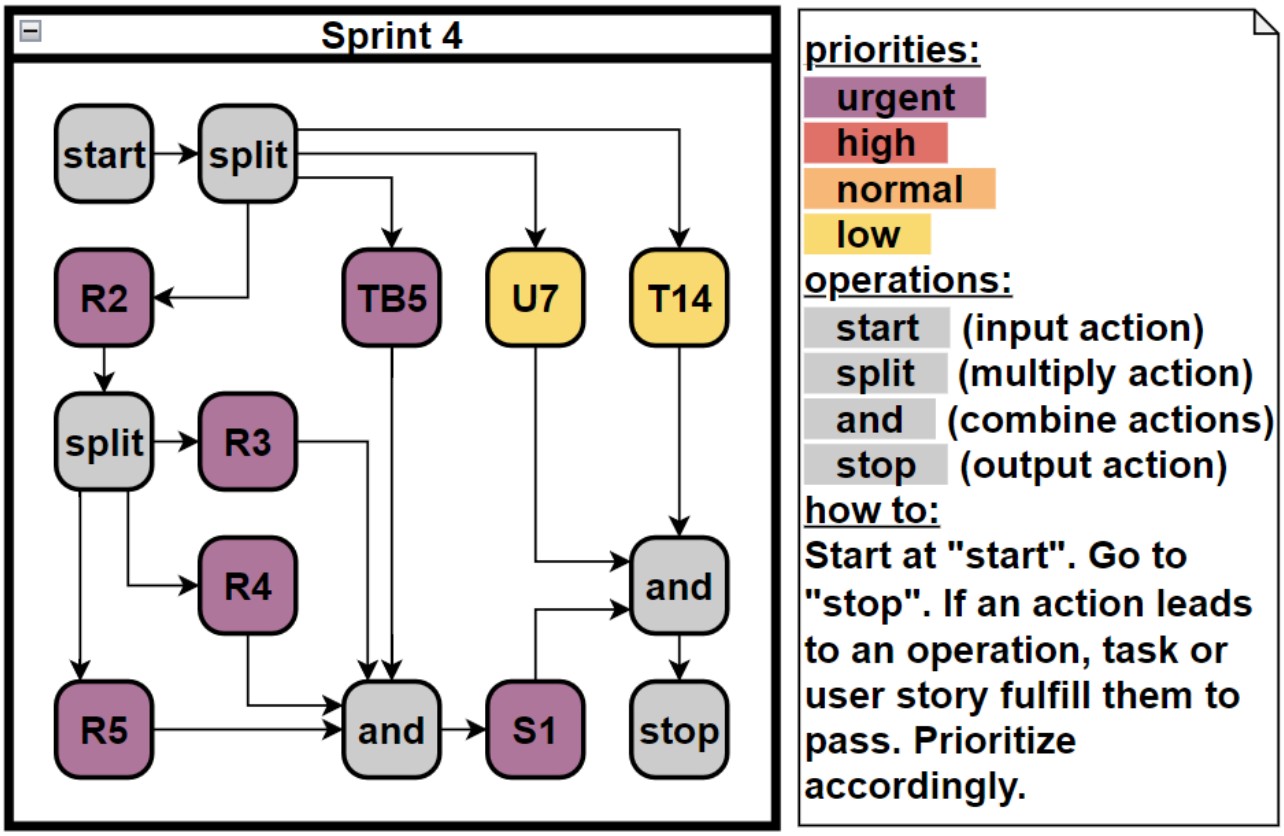
⊗ **S1** Erstelle Funktionalität zur Synchronisation, welche, wenn eine Änderung im System passiert (Erstellung, Anpassung und Löschung von User-Storys, Tasks, Task-Boards, rechtebezogenen Rollen, visuellen Rollen und Profilen), alle Sichten auf das System innerhalb von 2 Sekunden aktualisiert.

- Aufwandsschätzung: 20h
- Bearbeitung: 20h durch LK
- Test: 1h durch LK

Zusätzlicher Aufwand:

- Suchfunktion für User-Stories und Tasks + zusätzliche Filtermöglichkeiten für Tasks: 2h durch ZSH
- Login Page verändert durch PK: 1h
- Refactoring Frontend (HTML und JS) durch ZSH: 2h

1.2.6 Ablaufplan zu User-Storys und Tasks



2 Feinentwurf

2.1 Komponentendiagramm

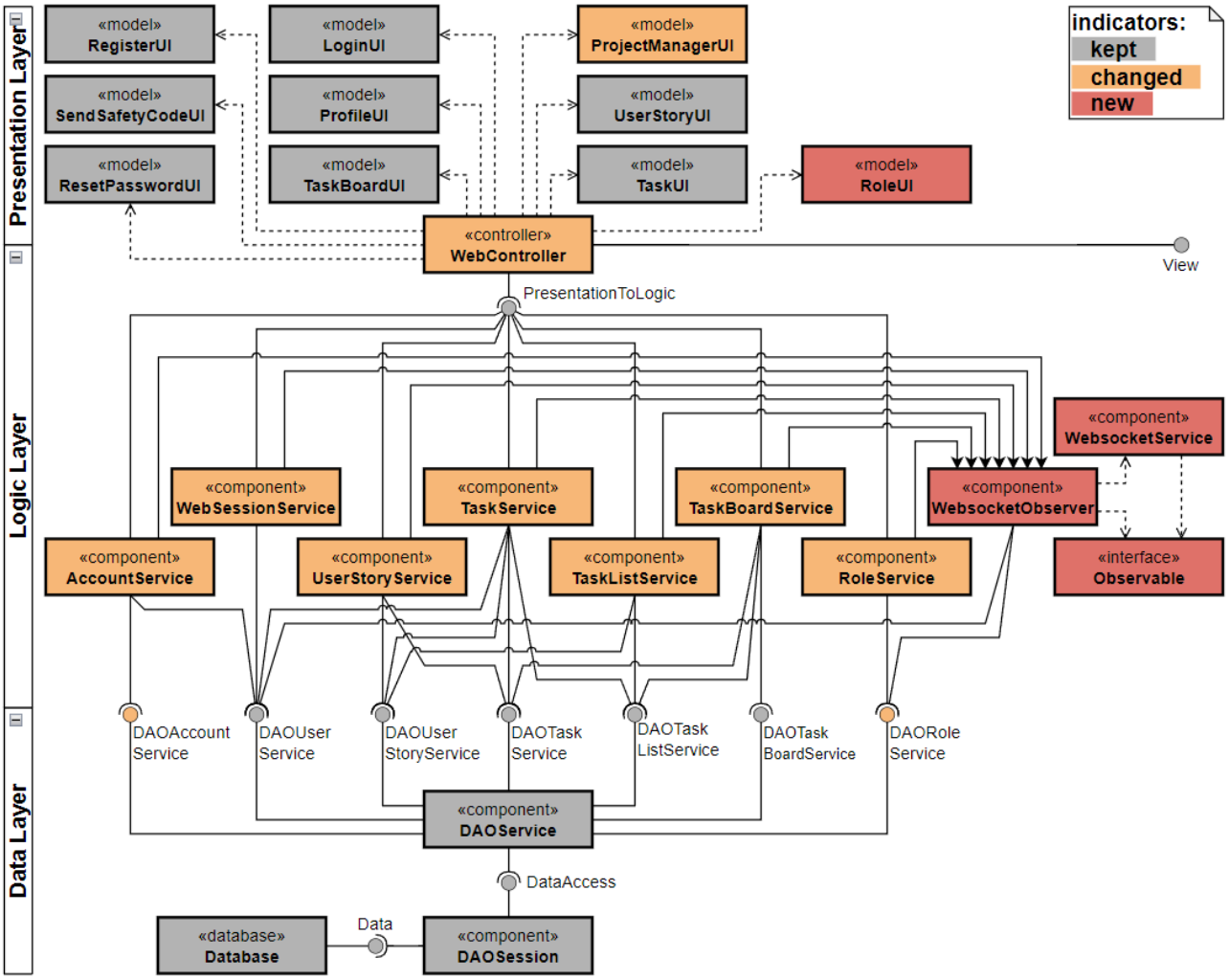


Abbildung 1: Komponentendiagramm Sprint 4

In diesem Sprint wurden visuelle Rollen und die Synchronisation eingebaut. Wegen der visuellen Rollen wurde in der Präsentationsschicht eine neue UI benötigt, die mit “RoleUI“ betitelt ist, und die ProjectManagerUI musste angepasst werden, um dort die visuellen Rollen anzeigen zu können. Weiterhin mussten im “Webcontroller“ Methoden hinzugefügt werden, um die Behandlung von visuellen Rollen zu ermöglichen. Zur Umsetzung der Synchronisation wurde in der Logikschicht ein Observer aufgebaut, bestehend aus den neuen Elementen “WebsocketObserver“, “WebsocketService“ und “Observable“. Des Weiteren ist für die Realisierung der Synchronisation eine Veränderung in allen Service-Klassen notwendig.

2.2 Klassendiagramme und Design Pattern

2.2.1 Klassendiagramm zum Backlog-Abschnitt Role

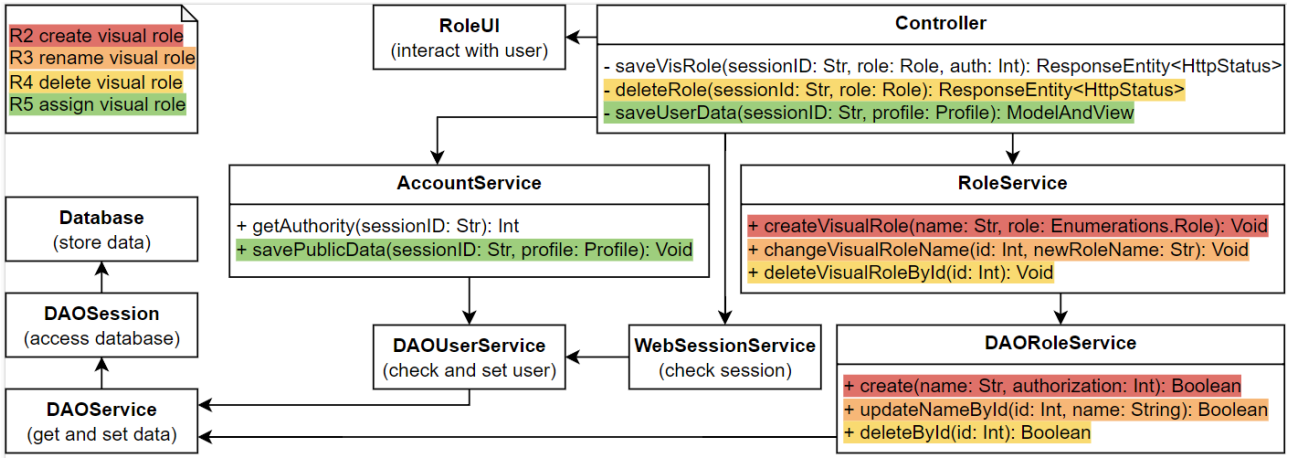


Abbildung 2: Klassendiagramm zum Abschnitt Role

In diesem Sprint befasst der Backlog-Abschnitt Role sich mit visuellen Rollen. Das umfasst die Erstellung (R2; rot in Abbildung 2), die Bearbeitung (R3; orange in Abbildung 2), das Löschen einer visuellen Rolle (R4; gelb in Abbildung 2) und das Zuteilen eines Nutzers zu einer visuellen Rolle (R5; grün in Abbildung 2).
Wenn ein Nutzer mit den Rechten eines Managers eine visuelle Rolle erstellt, wird ein Request an den “Controller“ geschickt. Bevor ein Request bearbeitet wird, wird durch den “WebSessionService“ geprüft, ob die SessionID des Nutzers noch gültig ist. Sobald die Gültigkeit bestätigt ist, wird der Request bearbeitet. Im Falle der Userstory R2 löst der Request im “Controller“ den Methodenaufruf “saveVisRole“ aus. Dadurch wird im “RoleService“ “createVisualRole“ aufgerufen, was wiederum im “DAORoleService“ die Methode “create“ aufruft. Diese verwendet den “DAOService“, um in der Datenbank einen Eintrag für die visuelle Rolle zu erstellen.
Um den Namen einer visuellen Rolle zu ändern, muss ein Nutzer mit den Rechten eines Managers den entsprechden Request schicken. Nach Überprüfung der SessionID des Nutzers wird im “Controller“ “saveVisRole“, wodurch im “RoleService“ der Aufruf der Methode “changeVisualRole“ ausgelöst wird. “changeVisualRole“ ruft die Methode “updateNameById“ vom “DAORoleService“ auf. Diese wird mithilfe des “DAOService“ den Eintrag in der Datenbank der entsprechenden visuellen Rolle bearbeiten.
Beim Löschen einer visuellen Rolle, wird ein Request von einem Client geschickt, dessen Nutzer die Rechte des Managers hat, der nach Überprüfung der SessionID im “Controller“ verwertet wird und die Methode “deleteRole“ aufgerufen. Diese ruft “deleteVisualRoleById“ vom “RoleService“ auf, die durch den Aufruf von “deleteById“ vom “DAORoleService“ und mit den Operationen des “DAOService“ den Eintrag der spezifischen visuellen Rolle aus der Datenbank entfernt.
Wenn sich ein Nutzer einer visuellen Rolle zuordnet, wird der entsprechende Request an den “Controller“ geschickt. Dieser ruft nach Überprüfung der SessionID “saveUserData“ im “Controller“ auf, wodurch “savePublicData“ vom “AccountService“ aufgerufen wird. Diese verwendet Methoden des “DAOUserService“, um mithilfe des “DAOService“ die Zugehörigkeit des Nutzers zu der visuellen Rolle in der Datenbank abzuspeichern.

2.2.2 Klassendiagramm zum Backlog-Abschnitt User-Story und Task

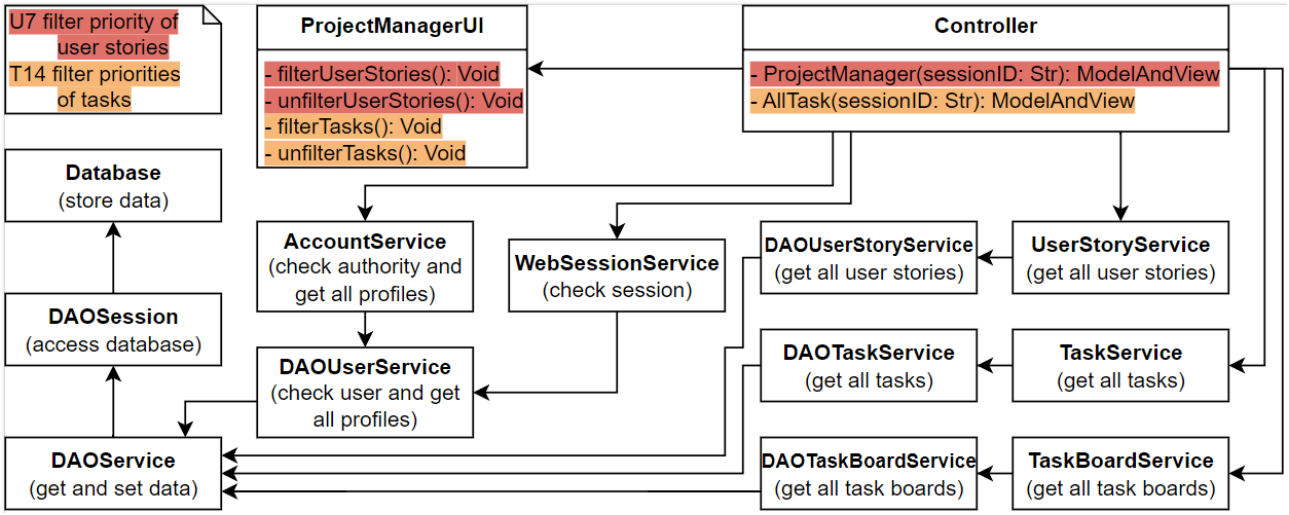


Abbildung 3: Klassendiagramm zum Abschnitt UserStory und Task

Basierend auf den Funktionalitäten der letzten Sprints ist in Abbildung 3 die Filter-Funktion der User-Storys und Tasks implementiert. Wie in Abbildung 3 farblich hervorgehoben, findet die Bearbeitung dieser Aufgaben primär im Frontend der Webanwendung - also beim Client selbst - statt. Hierbei werden über die jeweiligen "Controller"-Funktionen der 'ProjectManager" für Tasks oder User-Storys mit allen Einträgen an den Client übermittelt. Bei diesem werden dann basierend auf der Eingabe des Nutzers durch die entsprechenden Filter-Funktionen jene Eingaben ausgeblendet, welche das Filterkriterium nicht erfüllen. Werden die Filter wieder entfernt, so werden alle Einträge wieder eingeblendet.

2.2.3 Klassendiagramm zum Backlog-Abschnitt Task-Board

(in Sprint 3 behandelt)

2.2.4 Klassendiagramm zum Backlog-Abschnitt System

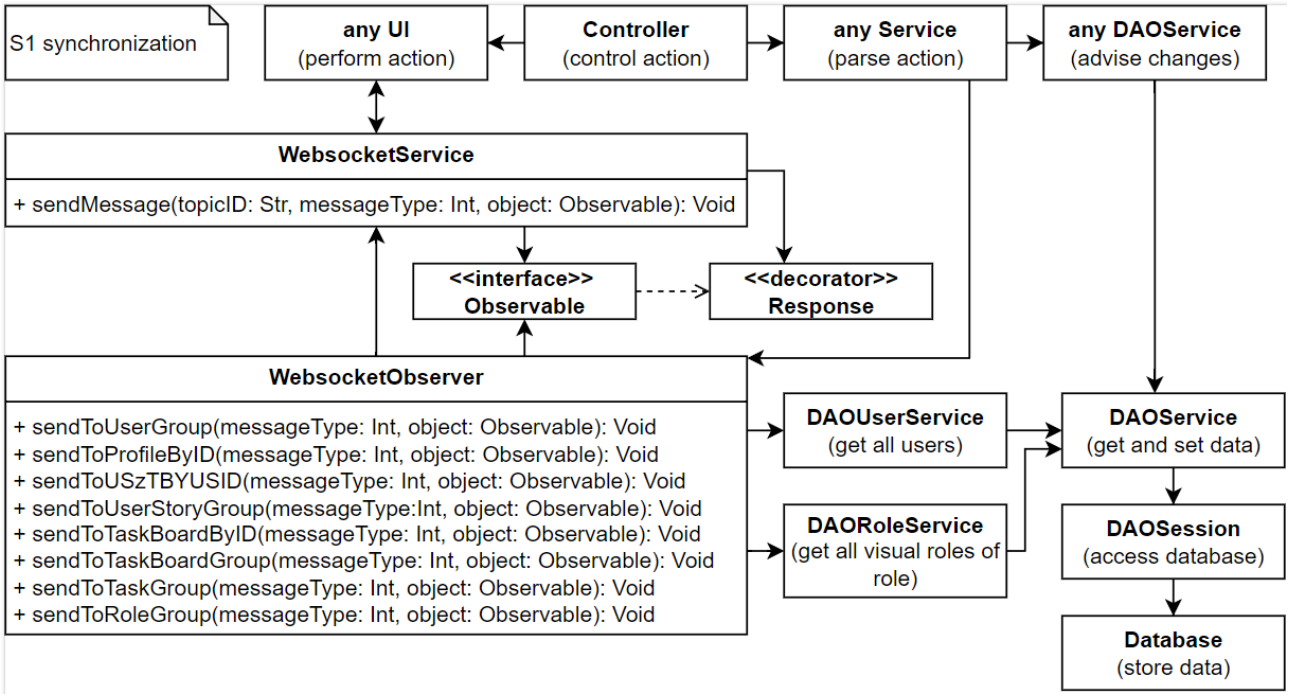


Abbildung 4: Klassendiagramm zum Abschnitt System

In Abbildung 4 ist die Implementation des in folgenden Abschnitt 2.3 beschriebenen Designpatterns "Observer" als Methode der Synchronisation zwischen Nutzern der Webanwendung beschrieben. Hierbei werden UIs, wie schon in früheren Sprints beschrieben, durch den Controller bereitgestellt. Haben diese erfolgreich beim Nutzer geladen, verbinden diese sich automatisch mit dem Websocket und subskribieren zu dem der UI entsprechenden Topic. Dadurch existiert eine bidirektionale Kommunikations-Kanal zwischen dem Client und dem Server, welcher zur Synchronisation der gezeigten Inhalte genutzt werden kann.
Wird nun durch einen Nutzer analog zu den beschriebenen Funktionalitäten der bisherigen Sprints über den Controller eine Änderung an den gespeicherten Daten hervorgerufen, wird über den entsprechenden Service die "sendToX" Funktion des "WebSocketObserver" aufgerufen - diese stellen die "updateToTopic"-Funktion des Observer-Patterns dar. Durch den Aufruf wird im "WebSocketOberserver" festgelegt, welche Topics benachrichtigt werden.
Basierend auf dieser Festlegung ruft der "WebSocketObserver" für die betroffenen Topics die "sendMessage"-Funktion des "WebSocketService" auf. Diese wandelt die Anfrage in ein "Responce"-Object um und übermittelt dieses dann mittels des Web Sockets an die entsprechenden Nutzer. Diese übermittelten Daten werden dann zur Synchronisation der UI verwendet.

2.3 Designpattern

Diesen Sprint wurde zum Erfüllen der User-Story S1 eine Abwandlung des Observer-Patterns implementiert, um alle aktiven Nutzer der Webanwendung schnellstmöglich über Änderungen der für sie relevanten Daten zu informieren.

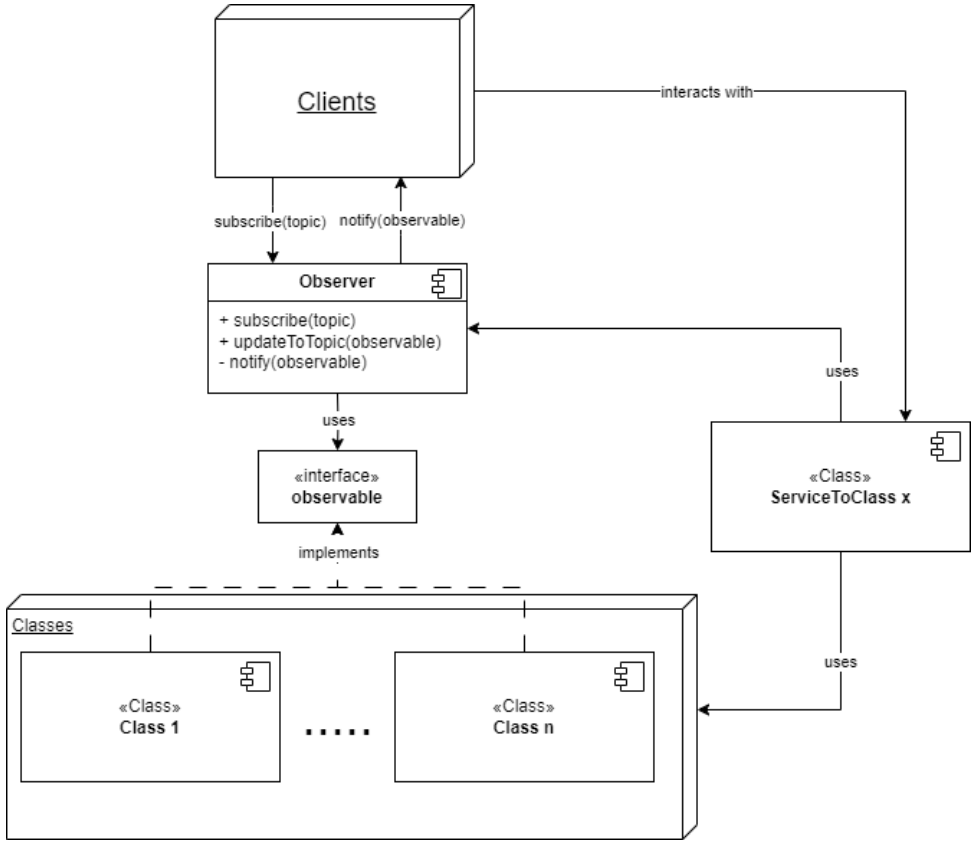


Abbildung 5: Observer-Pattern

Sei ein Client unserer Webanwendung zu einem Topic subskribiert. Interagiert nun ein Nutzer mit der Webanwendung und löst eine Änderung der Daten in der Datenbank aus, so nimmt dies der Observer, durch ein durch eine “ServiceToClass X“ aufgerufenes ”update”, wahr und benachrichtigt alle Nutzer, die zu betroffenen Topics subskribiert sind. Eine Änderung hat dabei Relevanz für ein Topic, wenn die zugehörige Seite der Webanwendung die veränderten Daten repräsentiert. Beim Subskribieren zu einem anderen Topic wird der Client aus der Subskriptionsliste des zu vorigen Topics entfernt.

Weiterhin haben wir mit der Klasse ”Responce” das Decorator-Pattern verwendet, um bei der Synchronisation zusätzliche Daten übermitteln zu können, ohne eine Vielzahl von neuen Klassen schreiben zu müssen. In unserem Fall erweitert der Decorator jedes Object der ”Observable” Klasse mit einem ”messageTyp”, welcher die Veränderung des Objectes im System definiert - Object ist neu, gelöscht, verändert.

2.4 Neues/verändertes Verhalten

In Abbildung 7 ist der Synchronisations-Prozess zwischen Clients beschrieben. Besucht ein Nutzer eine Seite der Webanwendung, so subskribiert er automatisch zu dem entsprechenden Topic über die WebSocketConfig. Geschieht dies erfolgreich, so kann eine Synchronisation stattfinden.
Sobald ein Nutzer eine Änderung hervorruft, werden diese analog zum Sequenzdiagramm der CRUD-Funktionen (siehe frühere Sprints) an die Datenbank weitergeleitet. Ob eine Synchronisation stattfindet, hängt von der Responce der Datenbank auf die Anfrage ab. Im “False“-Fall - ein Konflikt bei der Speicherung der Daten - wird die entsprechende Exception an den Anfragenden-Client übermittelt und dieser bearbeitet daraufhin die Exception. Eine Synchronisation ist dann nicht erforderlich.
Im “True“-Fall werden Daten in der Datenbank verändert, weshalb eine Synchronisation notwendig ist. Es wird dem “WebSocketObserver“ mittels “sendToGroup(...)“ gemeldet, dass eine Veränderung stattfand, die für eine Menge von Topics relevant ist. Abhängig von der Objekt-Klasse wird im “WebSocketObserver“ mittels “evaluate(...)“ eine Nachricht - “messageTyp“ -, die Information zum Update enthält, erstellt. Nun wird “sendMessage(...)“ aufgerufen, was zur Informierung aller betroffenen Clients über die Veränderung führt. Dazu wird vorerst im “WebSocketService“ die Erstellung eines Responce veranlasst. Das Responce-Objekt wird aus dem “messageTyp“ und dem Objekt generiert und zu einem JSON-String konvertiert, was durch “new Responce(...)“ und “Responce.toJSON“ der Responce-Klasse umgesetzt wird. Der JSON-String wird im “WebSocketService“ in eine “ResponceEntity“ mittels der “WebSocketConfig“ verpackt. Danach wird die “ResponceEntity“ durch “sendToTopic(...)“ an alle betroffenen Clients verschickt. Diese reagieren entsprechend auf die Mitteilungen, was zu einer Synchronisation der Clients bezüglich der für sie sichtbaren Daten führt.

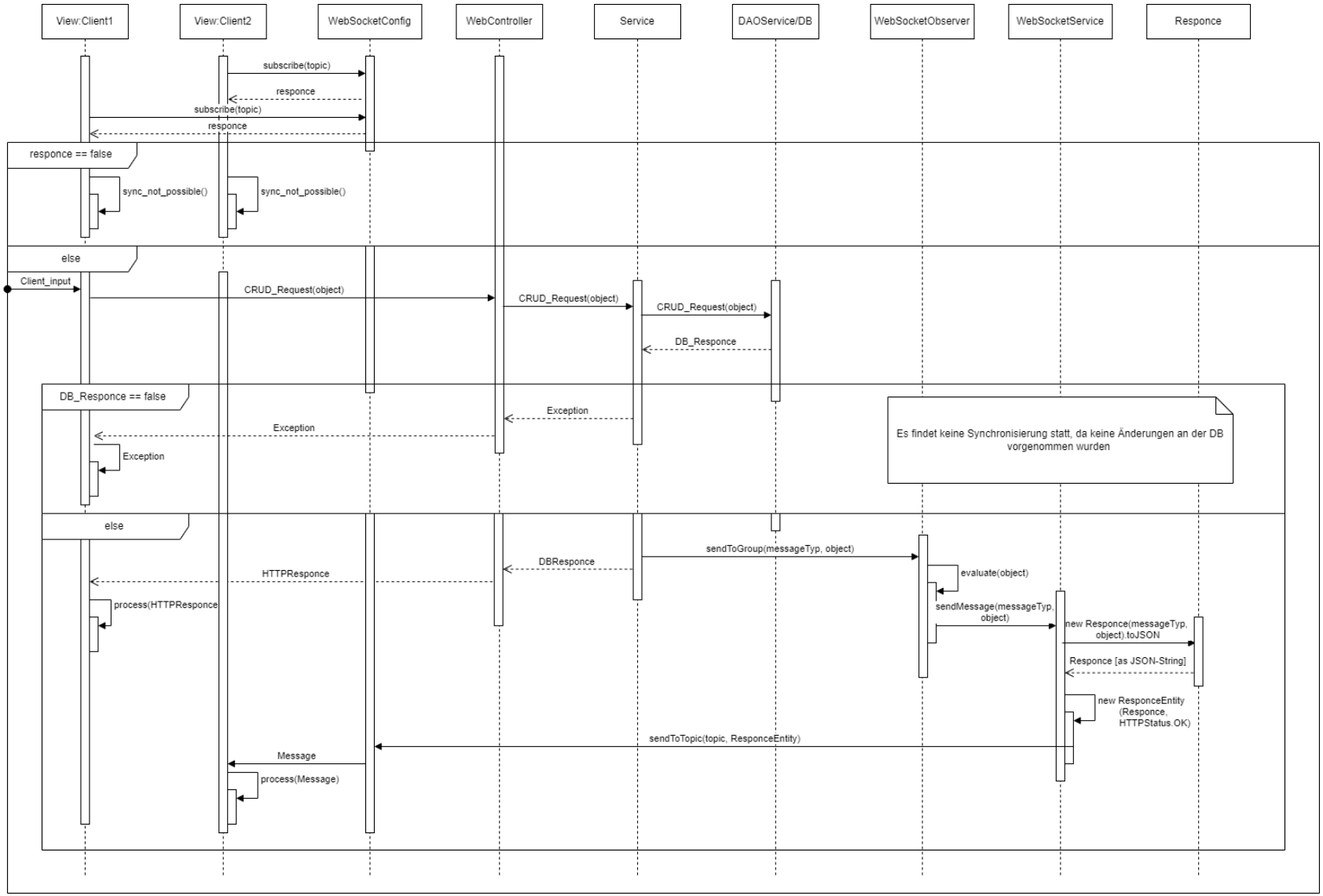


Abbildung 6: Sequenzdiagramm Synchronisation

2.5 Prototypen Frontend

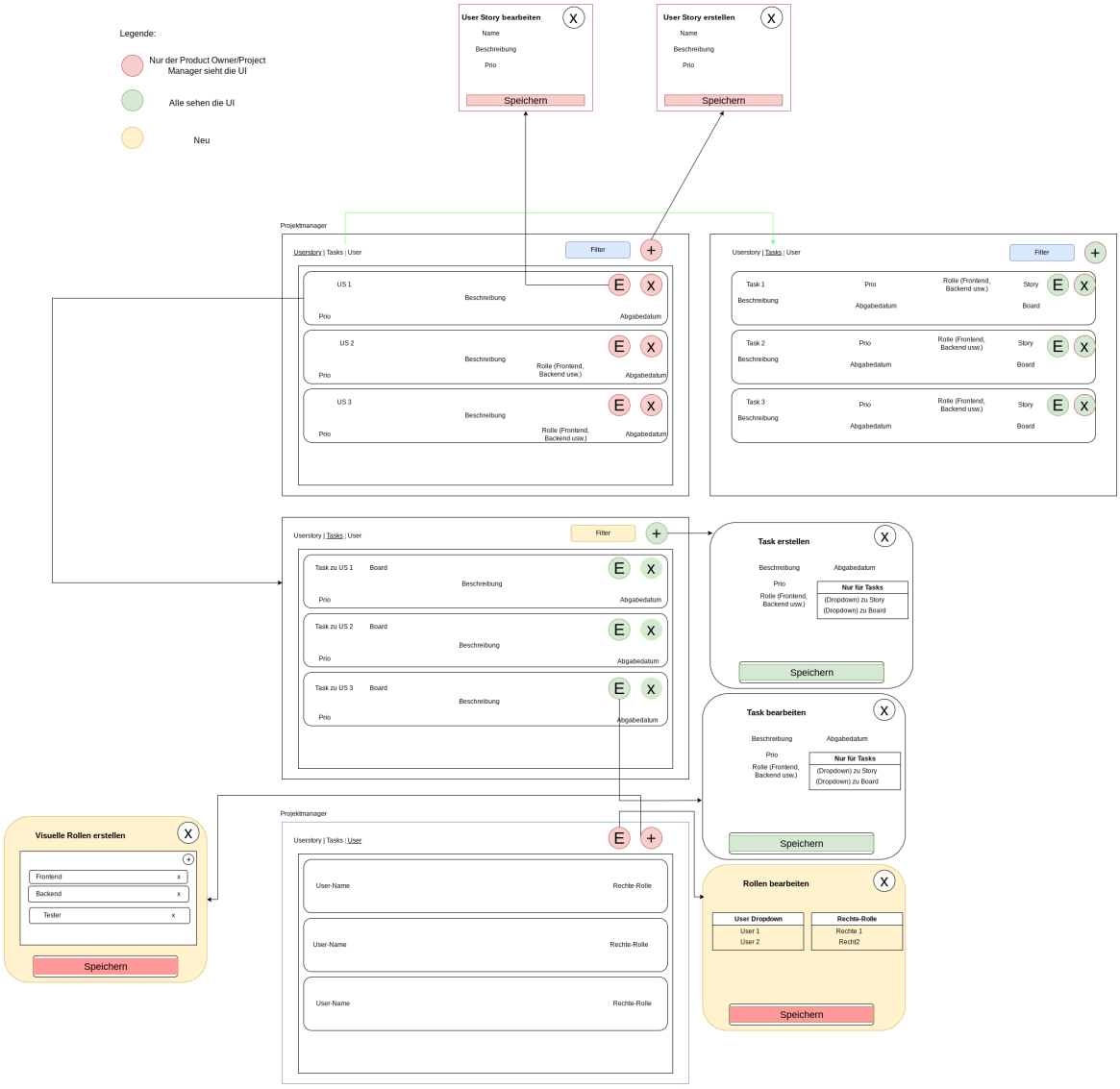


Abbildung 7: Prototyp Projektmanager

Im 4. Sprint sind die einzigen neuen Frontend-Änderungen die Filter/Suche sowie das Interface für die visuellen Rollen, wobei diese gelb gekennzeichnet wurden. Hierbei kann die Filter/Suchfunktion von jedem Nutzer verwendet werden. Die Schaltflächen für die visuellen Rollen werden nur Managern angezeigt, da nur sie visuelle Rollen anlegen dürfen, weswegen diese mit rot markiert wurden. Dementsprechend dürfen auch nur sie eine Rolle erstellen/verändern, sodass die "Speichern" Schaltflächen hier auch mit rot markiert wurden.

3 Implementation und Tests

3.1 Tests

3.1.1 HTTP-Request-Test

Diesen Sprint gab es keine neuen oder veränderten HTTP-Request-Tests.

3.1.2 Datenbank-Test

Diesen Sprint gab es keine neuen oder veränderten Datenbank-Tests.

3.1.3 Logic-Test

In diesem Sprint sind keine neuen Logiktests hinzugekommen, da die Tests für die Funktionalitäten des jetzigen Sprints schon im zu vorigen Sprint geschrieben wurden. Jedoch wurden die Tests in Test-Klassen umgebaut, haben einen neuen, strukturierten Aufbau bekommen und wurden mit Mockito umgeschrieben, weshalb die Tests wie neue Tests zu verstehen sind. Dies führt zu einer starken Verbesserung der Lesbarkeit der Tests. Aufgrund der Aufteilung der Tests in Klassen wurden die Test-IDs angepasst. Alle Test wurden erfolgreich abgeschlossen.

TestID	Herkunft	Erfolg	Author
Logic.Profile1	P1.B1	27.06.2024	Henry J. v. Rooyen
Logic.Profile2	P2.B1	27.06.2024	Henry J. v. Rooyen
Logic.Profile3	P4.B1	27.06.2024	Henry J. v. Rooyen
Logic.TaskBoard1	Grobentwurf	28.06.2024	Henry L. Freyschmidt
Logic.TaskBoard2	T16.B3	05.07.2024	Henry L. Freyschmidt
Logic.TaskBoard3	T16.B2	28.06.2024	Henry L. Freyschmidt
Logic.Task1	T3.B1	27.06.2024	Henry J. v. Rooyen
Logic.Task2	T4.B1	07.07.2024	Henry L. Freyschmidt
Logic.Task3	T5.B1	27.06.2024	Henry J. v. Rooyen
Logic.Task4	T8	05.07.2024	Henry L. Freyschmidt
Logic.Task5	T7.B4	07.07.2024	Henry L. Freyschmidt
Logic.Task6	T7.B2	07.07.2024	Henry L. Freyschmidt
Logic.Task7	T10.B3	07.07.2024	Henry L. Freyschmidt
Logic.Task8	T10.B2	07.07.2024	Henry L. Freyschmidt
Logic.Task9	T9.B3	07.07.2024	Henry L. Freyschmidt
Logic.Task10	T9.B2	07.07.2024	Henry L. Freyschmidt
Logic.Task11	T11.B2	07.07.2024	Henry L. Freyschmidt
Logic.UserStory1	U3.B1	25.06.2024	Henry L. Freyschmidt
Logic.UserStory2	U4.B1	25.06.2024	Henry L. Freyschmidt
Logic.UserStory3	U6.B1	25.06.2024	Henry L. Freyschmidt
Logic.User1	A2.B1	22.06.2024	Henry J. v. Rooyen
Logic.User2	A3.B1	25.06.2024	Henry L. Freyschmidt
Logic.User3	Grobentwurf	25.06.2024	Henry L. Freyschmidt
Logic.User4	R1.B2	25.06.2024	Henry L. Freyschmidt
Logic.VisualRole1	R2.B2	05.07.2024	Henry L. Freyschmidt
Logic.VisualRole2	R2.B1	05.07.2024	Henry L. Freyschmidt
Logic.VisualRole3	R3.B1	05.07.2024	Henry L. Freyschmidt
Logic.VisualRole4	R4.B1	05.07.2024	Henry L. Freyschmidt
Logic.VisualRole5	R2.B1	05.07.2024	Henry L. Freyschmidt

3.2 Tracing - Komponenten im Code

Ausgehend von: .../Project/src/main/

Komponente	Implementation
User Interfaces	
RegisterUI	/resources/templates/register.html
LoginUI	/resources/templates/index.html
SendSafetyCodeUI	/resources/templates/passwortForgot.html
ResetPasswortUI	/resources/templates/neuesPasswort.html
ProfileUI	/resources/templates/profil.html
ProjectManagerUI	/resources/templates/projectManager.html
UserStoryUI	/resources/templates/projectManager-TasksZuUserstory.html
TaskUI	/resources/templates/projectManager-Tasks.html
RoleUI/NutzerUI	/resources/templates/projectManager-Nutzer.html
TaskBoardUI	/resources/templates/taskBoard.html
Controller	
WebController	/java/com/team3/project/Controller/WebController.java
Fassade	
PresentationToLogic	/java/com/team3/project/Facede/PresentationToLogic.java
Services	
AccountService	/java/com/team3/project/service/AccountService.java
RoleService	/java/com/team3/project/service/RoleService.java
TaskBoardService	/java/com/team3/project/service/TaskBoardService.java
TaskListService	/java/com/team3/project/service/TaskListService.java
TaskService	/java/com/team3/project/service/TaskService.java
UserStoryService	/java/com/team3/project/service/UserStoryService.java
WebSessionService	/java/com/team3/project/service/WebSessionService.java
DAOServices	
DAOAccountService	/java/com/team3/project/DAOService/DAOAccountService.java
DAORoleService	/java/com/team3/project/DAOService/DAORoleService.java
DAOService	/java/com/team3/project/DAOService/DAOService.java
DAOTaskBoardService	/java/com/team3/project/DAOService/DAOTaskBoardService.java
DAOTaskListService	/java/com/team3/project/DAOService/DAOTaskListService.java
DAOTaskService	/java/com/team3/project/DAOService/DAOTaskService.java
DAOUserService	/java/com/team3/project/DAOService/DAOUserService.java
DAOUserStoryService	/java/com/team3/project/DAOService/DAOUserStoryService.java
WebSocket	
WebSocketConfig	/java/com/team3/project/config/WebSocketConfig.java
WebSocketService	/java/com/team3/project/Websocket/WebsocketService.java
WebSocketObserver	/java/com/team3/project/Websocket/WebsocketObserver.java
Database	
Database	/resources/DB

Tabelle 5: Tracing - Implementation

3.3 Tracing - Beendete Aufgaben

Die folgenden Aufgaben-Bezeichnungen beziehen sich auf die Bezeichner der Projektaufgabe.pdf und der Status ist eine subjektive Einschätzung unsererseits ohne Abspreche mit einem externen Produkt Owner oder Shareholder.

Aufgabe	Status	zugehörige Komponenten
01. Nutzerregistrierung	Erledigt	LoginUI RegisterUI SendSaftCodeUI ResetPasswortUI AccountService DAOUser-Table
02. User Story- & Backlog-Management	Erledigt	ProjectManagerUI UserStoryUI UserStoryService RoleService DAOUserStory-Table
03. Task Management	Erledigt	ProjectManagerUI TaskUI TaskService DAOTask-Table
04. Task/User Story Listen	Erledigt	ProjectManagerUI TaskBoardUI TaskService TaskBoardService UserStoryService DAOUserStory-Table DAOTask-Table DAOTaskBoard-Table
05. Nutzer Profile	Erledigt	ProfileUI ProjectManagerUI AccountService DAOUser-Table
06. Nutzer-Task Zuordnung	Erledigt	ProjectManagerUI RoleUI AccountService TaskService DAOTaskXUser-Table
07. Pair-Programming Planung	Erledigt	ProjectManagerUI RoleUI AccountService TaskService DAOTaskXUser-Table

Tabelle 6: Tracing - Aufgaben

08. Schätzungstracker	Erledigt	TaskBoardUI TaskService DAOTask-Table
09. Suche und Filter	Erledigt	ProjectManagerUI TaskService UserStoryService DAOTask-Table DAOUserStory-Table
10. Benachrichtigungen	anhaltend	TaskService DAOTask-Table
11. Agile Practices	in Planung	
12. Synchronisation	Erledigt	TaskService UserStoryService UserService WebsocketService WebsocketController WebsocketObserver
13. Parallele Bearbeitung	Erledigt	TaskBoardUI TaskService UserStoryService UserService WebsocketService WebsocketController WebsocketObserver
14. Multi-Nutzer	Erledigt	WebSessionService DAOUser-Table
15. Verfügbarkeit	-	
16. Updates	Erledigt	WebsocketService WebsocketController WebsocketObserver
17. Änderungen	-	

Tabelle 7: Tracing - Aufgaben

3.4 Laufender Prototyp

Im Prototypen wurden alle geplanten Tasks im Projekt-Manager umgesetzt. Zum einen existiert nun eine Filter- und Suchfunktion. Durch die Filterfunktion hat man nun die Möglichkeit, Tasks sowie User Stories nach Prioritäten (low, normal, high, urgent) und nach dem Status, ob eine Task erledigt wurde oder nicht, zu filtern. Dabei kann man jeweils nur eine einzige Eigenschaft gleichzeitig auswählen. Diese Funktion findet man über den Task- und User-Story Kärtchen und erfüllt somit die geplanten Tasks T14 und U7.

Ebenfalls existiert neben der Filterfunktion eine Suchfunktion, mit welcher man Tasks und User-Stories nach ihrer Beschreibung suchen kann. Dies war nicht direkt im vierten Sprint geplant, wurde aber aufgrund von Vollständigkeit ebenfalls implementiert.

Eine weitere neue Funktion im Projekt-Manager sind die neuen visuellen Rollen. Drückt der Manager oben rechts auf die Schaltfläche "Rollen bearbeiten", öffnet sich ein Pop-up, in welchem man die Rollen erstellen, bearbeiten und löschen kann (R3, R4). Aufgrund dessen, dass jede Rolle an bestimmte Rechte gebunden ist, existiert beim Erstellen ein Dropdown, mit welchem man zur Rolle ein

Recht zuweisen kann. (R2)

Existieren Rollen, so hat man ebenfalls die Möglichkeit, die oben links die Rollen nach den Rechten zu filtern.
Diese Rollen können nun für Nutzer verwendet werden. Jeder Nutzer kann hierbei beim Bearbeiten seines Profils mehrere Rollen auswählen, solange diese zu seinen Rechten gehören. Die Rollen werden dann anschließend auf dem eigenen Profil angezeigt. Dies erfüllt die Anforderung R5. Als Nächstes wurde eine neue Funktion im Task-Board hinzugefügt, nämlich die Anzeige der zugewiesenen Nutzer zu einer Task. Diese wurde im letzten Sprint angefangen, aber nicht beendet. Die Nutzer lassen sich über das Erstellen/Bearbeiten einer Task zuweisen über das Dropdown-Menü. Wurden Nutzer zugewiesen, so kann man dies durch Doppelklick auf eine Task im Task-Board sehen. Hier werden die Nutzer als farbige Kreise angezeigt, wobei man durch Doppelklick auf einen Kreis auf das Profil dieses Nutzers gelangt. Dies erfüllt die Task TB5.

Zuallerletzt wurde die Synchronisation zwischen mehreren Nutzern implementiert. Würde man mit mehreren Nutzern die Seite gleichzeitig nutzen und beispielsweise eine Task ändern oder im Task-Board verschieben, so wird diese Änderung bei allen anderen Nutzern angezeigt, sodass die Task S1 erfüllt wurde.

4 Abweichungen von Sprintplanung

In diesem Sprint gab es im Vergleich zu den vorigen nicht allzu viele Abweichungen. Aufgrund dessen, dass unser Projekt die meisten Anforderungen der gestellten Projektaufgabe erfüllt hatte, haben wir in diesem Sprint nur noch wenige Tasks geplant, welche auch alle erfüllt wurden. Die restlichen geplanten Tasks, welche nicht in diesem Sprint vorgesehen wurden, werden dementsprechend nicht mehr implementiert, da diese nicht mehr nötig sind.
Eine kleine Abweichung sind die Rollen im Task-Board, welche ursprünglich geplant wurden, aber in Sprint 4 verworfen wurden. Aus diesem Grund existiert beim Doppelklick auf eine Task im Task-Board ein Feld für "Rollen", obwohl diese Option beim Erstellen einer Task nicht auswählbar ist.

Eine weitere Abweichung wäre die Suche innerhalb der User-Stories und Tasks. Diese waren für diesen Sprint nicht geplant, wurden aber wegen Vollständigkeit implementiert, um eine weitere Anforderung der Projektaufgabe zu erfüllen. Ebenfalls kann man die Rollen nach Rechten filtern, obwohl dies auch nicht in den Tasks geplant wurde.

5 Technologien

In diesem Sprint kam Mockito als genutzte Technologie hinzu, diese wird jedoch nur in den Logic-Tests verwandt, somit bleibt alles andere unverändert.
Es wird weiterhin folgendes genutzt:

- Im Backend
  - Java
  - Spring
  - Spring Boot
  - (Mockito)
- Im Frontend
  - HTML
  - CSS
  - JavaScript
- In der Datenbank-Schicht
  - SQLite
  - Hibernate