USER STORIES

---

1. Account Management

1. **Account Anlegen:**
   Als neuer Nutzer möchte ich einen Account anlegen können, damit ich die Plattform nutzen und auf personalisierte Inhalte zugreifen kann.
2. **Einloggen:**
   Als registrierter Nutzer möchte ich mich mit meinen Zugangsdaten einloggen können, damit ich auf meine Daten und personalisierte Inhalte zugreifen kann.
3. **Passwort Zurücksetzen:**
   Als registrierter Nutzer möchte ich mein Passwort zurücksetzen können, falls ich es vergessen habe, damit ich wieder Zugriff auf meinen Account erhalte.

---

2. Backlog Management

1. **User Stories Anlegen:**
   Als Product Owner (PO) möchte ich neue User Stories im Backlog anlegen können, damit ich die Anforderungen des Projekts dokumentieren und verwalten kann.
2. **User Story Prioritäten Festlegen:**
   Als Product Owner (PO) möchte ich die Prioritäten der User Stories im Backlog festlegen können, damit das Team an den wichtigsten Aufgaben zuerst arbeitet.

---

3. Task Management

1. **Task Erstellen:**
   Als Teammitglied möchte ich Tasks erstellen können, damit ich die Arbeit in kleinere Einheiten aufteilen kann.
2. **Task mit User Story Verlinken:**
   Als Teammitglied möchte ich Tasks mit den entsprechenden User Stories verlinken können, damit die Zusammenhänge klar ersichtlich sind.
3. **Tasks Priorisieren:**
   Als Teammitglied möchte ich die Tasks priorisieren können, damit ich sicherstellen kann, dass die wichtigsten Aufgaben zuerst erledigt werden.
4. **Tasks als "Complete" oder "Incomplete" Markieren:**
   Als Teammitglied möchte ich Tasks als „complete“ oder „incomplete“ markieren können, damit der Fortschritt der Arbeit klar sichtbar ist.

---

4. Task/Story Listen

1. **Listen für Tasks und User Stories Erstellen:**
   Als Teammitglied möchte ich mehrere Listen für Tasks und User Stories erstellen

Was heißt "getrennt verwalten"? Sind sie komplett unabhängig voneinander oder geht es beispielsweise nur um den Status der Bearbeitung oder ähnliches?

können, damit ich zwischen Projekt Backlog und Sprint Backlog klar unterscheiden und diese getrennt verwalten kann.

2. **Status für Tasks Anlegen:**
   Als Teammitglied möchte ich verschiedene Status für Tasks anlegen können (z. B. in Bearbeitung, unter Review, fertiggestellt), damit der Fortschritt besser dokumentiert werden kann.
3. **Status für Tasks Ändern:**
   Als Teammitglied möchte ich den Status von Tasks ändern können, damit der Fortschritt der Arbeit aktuell gehalten werden kann.
4. **Tasks zwischen Listen Bewegen:**
   Als Teammitglied möchte ich Tasks zwischen den Listen (z. B. Projekt Backlog, Sprint Backlog oder Status-Listen) hin- und herbewegen können, damit ich Aufgaben flexibel organisieren und an den aktuellen Arbeitsablauf anpassen kann.
5. **Übersicht der Listen und Tasks:**
   Als Teammitglied möchte ich eine übersichtliche Darstellung der Listen und Tasks mit Filter- und Sortieroptionen sehen können, damit ich den Fortschritt des Projekts besser nachvollziehen kann.

---

5. Nutzer Profile

1. **Nutzer Profile Ansehen:**
   Als Teammitglied möchte ich Profile anderer Nutzer mit persönlichen Informationen (z. B. Name und Rolle) ansehen können, damit ich die Rollen und Verantwortlichkeiten meines Teams besser verstehe.
2. **Nutzer Profile Bearbeiten:**
   Als Nutzer möchte ich mein eigenes Profil bearbeiten können, damit ich meine Informationen aktuell und korrekt halten kann.
3. **Rollen Festlegen und Anpassen:**
   Als Administrator möchte ich die Rollen der Nutzer (z. B. Tester, PO, Entwickler) festlegen und anpassen können, damit die Verantwortlichkeiten innerhalb des Teams klar definiert sind.

---

6. Nutzer-Task Zuordnung

1. **Nutzer-Task Zuordnung Erstellen:**
   Als Teammitglied möchte ich Nutzer zu spezifischen Tasks zuordnen können, damit klar ersichtlich ist, wer für welche Aufgabe verantwortlich ist.
2. **Übersicht der Nutzer-Task Zuordnung:**
   Als Teammitglied möchte ich eine visuelle Übersicht der Nutzer-Task Zuordnungen (z. B. in einer Tabelle oder einem Dashboard) haben, damit ich schnell erkennen kann, wer für welche Aufgabe verantwortlich ist.

---

7. Pair Programming

1. **Mehrere Nutzer zu einer Task zuordnen:**
   Als Teammitglied möchte ich mehreren Nutzern eine einzelne Task zuordnen können, damit Pair-Programming oder Teamarbeit an einer Aufgabe geplant und umgesetzt werden kann.
2. **Pair-Programming Übersicht:**
   Als Teammitglied möchte ich eine Übersicht über alle Tasks haben, bei denen Pair-Programming geplant ist, damit ich sehen kann, welche Aufgaben in Zusammenarbeit erledigt werden.

---

8. Schätzungstracking

1. **Aufwandsschätzungen für Tasks hinzufügen:**
   Als Teammitglied möchte ich zu jeder Task eine Aufwandsschätzung hinzufügen können, damit der voraussichtliche Zeitbedarf klar dokumentiert ist.
2. **Tatsächliche Bearbeitungszeit einer Task dokumentieren:**
   Als Teammitglied möchte ich die tatsächliche Bearbeitungszeit einer Task dokumentieren können, damit ich nachvollziehen kann, wie lange die Lösung tatsächlich gedauert hat.
3. **Visualisierung von Schätzungsabweichungen:**
   Als Teammitglied möchte ich eine Visualisierung der Abweichungen zwischen geschätzter und tatsächlicher Bearbeitungszeit sehen, damit ich die Genauigkeit unserer Schätzungen analysieren und verbessern kann.

---

9. Suche und Filter

1. **Aufgaben nach Prioritäten Filtern:**
   Als Nutzer möchte ich Aufgaben nach Prioritäten filtern können, damit ich effektiver arbeiten kann.
2. **Aufgaben nach Status Filtern:**
   Als Nutzer möchte ich Aufgaben nach ihrem Status filtern können, damit ich sicherstellen kann, dass die Aufgaben in der richtigen Reihenfolge erledigt werden.
3. **Nutzer für Aufgaben Anzeigen:**
   Als Benutzer möchte ich die Person sehen können, die einer Aufgabe zugeordnet ist, damit ich mit der zuständigen Person Kontakt aufnehmen kann.

---

10. Benachrichtigungen

1. *Als Nutzer möchte ich vom System per E-Mail oder in der App benachrichtigt werden, wenn Aufgaben anstehen, damit ich rechtzeitig eingreifen kann, ohne Fristen zu verpassen.*

Sollen Besprechungen über die entwickelte Software durchgeführt werden können?

11. Agile Practices

nicht das "System"?

1. *Als Product Owner möchte ich, dass mein Team agile Praktiken wie Sprint-Planung, Retrospektiven und tägliche Besprechungen unterstützt, damit wir die Prozesse kontinuierlich verbessern und effektiver arbeiten können.*
2. *Als Product Owner möchte ich, dass mein Team tägliche Besprechungen durchführt, damit sichergestellt wird, dass das Team nach den agilen Arbeitsgrundsätzen arbeitet.*

Die zweite User Story hat nichts mit dem System zu tun, oder?

---

12. Synchronisation

1. *Als Benutzer möchte ich, dass meine Änderungen an einer Aufgabe für alle Teammitglieder in Echtzeit sichtbar sind, damit alle auf dem Laufenden sind und Missverständnisse vermieden werden.*

Was ist "Echtzeit"

---

13. Parallele Bearbeitung

1. *Als Benutzer möchte ich, dass meine Änderungen an einer Aufgabe für alle Teammitglieder in Gleichzeit sichtbar sind, damit alle auf dem Laufenden sind und Missverständnisse vermieden werden.*

---

14. Multi-Nutzer

Warum will denn der Administrator das?

1. *Als Administrator möchte ich, dass mehrere Benutzer gleichzeitig am System arbeiten können, damit parallele Aufgabenbearbeitung ermöglicht wird.*
2. *Als Administrator möchte ich, dass die Benutzer die Aufgaben flexibel organisieren können, damit die Arbeit im Team strukturiert und nachvollziehbar bleibt.*

---

15. Verfügbarkeit

1. *Als Nutzer möchte ich zu jeder Tageszeit auf das System zugreifen und meine Aufgaben bearbeiten können, damit ich flexibel arbeiten kann.*

✓

---

16. Updates

Aha! Also ist das "Echtzeit"

1. *Als Benutzer möchte ich, dass Änderungen in den Aufgabenlisten innerhalb von maximal 2 Sekunden synchronisiert und für alle Nutzer sichtbar werden, damit der Projektfortschritt immer aktuell bleibt.*

Würde ich nicht unbedingt als User Story formulieren, da der hier angesprochene "Entwickler" kein richtiger Nutzer des Systems ist, sondern eben der Entwickler.

17. Änderungen

1. **Listentypen Hinzufügen:**
   Als Entwickler möchte ich, dass das System nach seiner Freigabe leicht erweiterbar ist, sodass ich neue Listentypen hinzufügen kann, damit verschiedene Arten von Aufgaben und Prozessen flexibel abgebildet werden können.
2. **Filteroptionen Hinzufügen:**
   Als Entwickler möchte ich, dass das System nach seiner Freigabe leicht erweiterbar ist, sodass ich neue Filteroptionen hinzufügen kann, damit Nutzer relevante Aufgaben schneller finden und priorisieren können.
3. **Benachrichtigungstypen Hinzufügen:**
   Als Entwickler möchte ich, dass das System nach seiner Freigabe leicht erweiterbar ist, sodass ich neue Benachrichtigungstypen hinzufügen kann, damit das Team besser über wichtige Ereignisse und Fristen informiert wird.

Anforderungen

1. **ANF-001:** Nutzerregistrierung und Authentifikation
   Erlaubt Nutzern Accounts anzulegen, sich einzuloggen und ihre Passwörter zurückzusetzen. Nicht atomar!
2. **ANF-002:** User Story- und Backlog-Management
   Erlaubt dem Team und speziell dem Product Owner (PO), neue User Stories im Backlog anzulegen und zu priorisieren.
3. **ANF-003:** Task Management
   Erlaubt dem Team, Tasks zu erstellen, zu priorisieren und mit den User Stories zu verlinken.
4. **ANF-004:** Task/User Story Listen
   Erlaubt dem Team, mehrere Listen für Tasks und User Stories zu verwalten und Tasks zwischen Listen zu verschieben.
5. **ANF-005:** Nutzer Profile
   Erlaubt Nutzern, Profile mit persönlichen Informationen anzusehen und zu bearbeiten.
6. **ANF-006:** Nutzer-Task Zuordnung
   Erlaubt dem System, Nutzer mit spezifischen Tasks zu verknüpfen.
7. **ANF-007:** Pair Programming
   Erlaubt, mehrere Nutzer zu einer Task zuzuordnen und Pair-Programming zu planen.
8. **ANF-008:** Schätzungstracking
   Erlaubt dem Team, Aufgaben mit Aufwandsschätzungen und tatsächlichen Bearbeitungszeiten zu dokumentieren.
9. **ANF-009:** Suche und Filter
   Bietet Such- und Filterfunktionen, um Aufgaben basierend auf Kriterien wie Priorität oder Status zu finden.
10. **ANF-010:** Benachrichtigungen
    Basiert auf den Schätzungen des Teams, um Nutzer über anstehende Aufgaben zu benachrichtigen.
11. **ANF-011:** Agile Practices
    Unterstützt agile Praktiken wie Sprint-Planung, Retrospektiven und tägliche Besprechungen.

12. **ANF-012:** Synchronisation
    Stellt sicher, dass Änderungen an Aufgaben in Echtzeit für alle Nutzer sichtbar sind.
13. **ANF-013:** Parallele Bearbeitung
    Erlaubt mehreren Nutzern, gleichzeitig an denselben Aufgaben zu arbeiten.
14. **ANF-014:** Multi-Nutzer
    Stellt sicher, dass mehrere Nutzer gleichzeitig am System arbeiten können.
15. **ANF-015:** Verfügbarkeit
    Stellt sicher, dass das System rund um die Uhr verfügbar ist.
16. **ANF-016:** Updates
    Sorgt dafür, dass Änderungen in den Aufgabenlisten innerhalb von maximal 2 Sekunden sichtbar werden.
17. **ANF-017:** Änderungen
    Erlaubt Entwicklern, das System nach der Freigabe leicht zu erweitern.

Anforderungen sind nicht ausreichend überarbeitet

User Stories (Mit allen IDs und Tracing)

Hier ist jetzt nur das Tracing neu, oder? Also die User Stories sind dieselben wie weiter oben?

Für ANF-001: Nutzerregistrierung und Authentifikation

1. **US-001:** Als neuer Nutzer möchte ich einen Account anlegen können, damit ich die Plattform nutzen und auf personalisierte Inhalte zugreifen kann.
   Traced to: ANF-001
2. **US-002:** Als registrierter Nutzer möchte ich mich mit meinen Zugangsdaten einloggen können, damit ich auf meine Daten und personalisierte Inhalte zugreifen kann.
   Traced to: ANF-001
3. **US-003:** Als registrierter Nutzer möchte ich mein Passwort zurücksetzen können, falls ich es vergessen habe, damit ich wieder Zugriff auf meinen Account erhalte.
   Traced to: ANF-001

Für ANF-002: User Story- und Backlog-Management

4. **US-004:** Als Product Owner möchte ich neue User Stories im Backlog anlegen können, damit ich die Anforderungen des Projekts dokumentieren und verwalten kann.
   Traced to: ANF-002
5. **US-005:** Als Product Owner möchte ich die Prioritäten der User Stories im Backlog festlegen können, damit das Team an den wichtigsten Aufgaben zuerst arbeitet.
   Traced to: ANF-002

Für ANF-003: Task Management

6. **US-006:** Als Teammitglied möchte ich Tasks erstellen können, damit ich die Arbeit in kleinere Einheiten aufteilen kann.
   Traced to: ANF-003
7. **US-007:** Als Teammitglied möchte ich Tasks mit den entsprechenden User Stories verlinken können, damit die Zusammenhänge klar ersichtlich sind.
   Traced to: ANF-003

8. **US-008:** Als Teammitglied möchte ich die Tasks priorisieren können, damit ich sicherstellen kann, dass die wichtigsten Aufgaben zuerst erledigt werden.
   Traced to: ANF-003
9. **US-009:** Als Teammitglied möchte ich Tasks als „complete“ oder „incomplete“ markieren können, damit der Fortschritt der Arbeit klar sichtbar ist.
   Traced to: ANF-003

Für ANF-004: Task/User Story Listen

10. **US-010:** Als Teammitglied möchte ich mehrere Listen für Tasks und User Stories erstellen können, damit ich zwischen Projekt Backlog und Sprint Backlog klar unterscheiden und diese getrennt verwalten kann.
    Traced to: ANF-004
11. **US-011:** Als Teammitglied möchte ich verschiedene Status für Tasks anlegen können, damit der Fortschritt besser dokumentiert werden kann.
    Traced to: ANF-004
12. **US-012:** Als Teammitglied möchte ich den Status von Tasks ändern können, damit der Fortschritt der Arbeit aktuell gehalten werden kann.
    Traced to: ANF-004
13. **US-013:** Als Teammitglied möchte ich Tasks zwischen den Listen hin- und herbewegen können, damit ich Aufgaben flexibel organisieren und an den aktuellen Arbeitsablauf anpassen kann.
    Traced to: ANF-004
14. **US-014:** Als Teammitglied möchte ich eine übersichtliche Darstellung der Listen und Tasks mit Filter- und Sortieroptionen sehen können, damit ich den Fortschritt des Projekts besser nachvollziehen kann.
    Traced to: ANF-004

Für ANF-005: Nutzer Profile

15. **US-015:** Als Teammitglied möchte ich Profile anderer Nutzer mit persönlichen Informationen (z. B. Name und Rolle) ansehen können, damit ich die Rollen und Verantwortlichkeiten meines Teams besser verstehe.
    Traced to: ANF-005
16. **US-016:** Als Nutzer möchte ich mein eigenes Profil bearbeiten können, damit ich meine Informationen aktuell und korrekt halten kann.
    Traced to: ANF-005
17. **US-017:** Als Administrator möchte ich die Rollen der Nutzer (z. B. Tester, PO, Entwickler) festlegen und anpassen können, damit die Verantwortlichkeiten innerhalb des Teams klar definiert sind.
    Traced to: ANF-005

Für ANF-006: Nutzer-Task Zuordnung

18. **US-018:** Als Teammitglied möchte ich Nutzer zu spezifischen Tasks zuordnen können, damit klar ersichtlich ist, wer für welche Aufgabe verantwortlich ist.
    Traced to: ANF-006
19. **US-019:** Als Teammitglied möchte ich eine visuelle Übersicht der Nutzer-Task Zuordnungen (z. B. in einer Tabelle oder einem Dashboard) haben, damit ich schnell erkennen kann, wer für welche Aufgabe verantwortlich ist.
    Traced to: ANF-006

Für ANF-007: Pair Programming

20. **US-020:** Als Teammitglied möchte ich mehreren Nutzern eine einzelne Task zuordnen können, damit Pair-Programming oder Teamarbeit an einer Aufgabe geplant und umgesetzt werden kann.
    Traced to: ANF-007
21. **US-021:** Als Teammitglied möchte ich eine Übersicht über alle Tasks haben, bei denen Pair-Programming geplant ist, damit ich sehen kann, welche Aufgaben in Zusammenarbeit erledigt werden.
    Traced to: ANF-007

Für ANF-008: Schätzungstracking

22. **US-022:** Als Teammitglied möchte ich zu jeder Task eine Aufwandsschätzung hinzufügen können, damit der voraussichtliche Zeitbedarf klar dokumentiert ist.
    Traced to: ANF-008
23. **US-023:** Als Teammitglied möchte ich die tatsächliche Bearbeitungszeit einer Task dokumentieren können, damit ich nachvollziehen kann, wie lange die Lösung tatsächlich gedauert hat.
    Traced to: ANF-008
24. **US-024:** Als Teammitglied möchte ich eine Visualisierung der Abweichungen zwischen geschätzter und tatsächlicher Bearbeitungszeit sehen, damit ich die Genauigkeit unserer Schätzungen analysieren und verbessern kann.
    Traced to: ANF-008

Für ANF-009: Suche und Filter

25. **US-025:** Als Nutzer möchte ich Aufgaben nach Prioritäten filtern können, damit ich effektiver arbeiten kann.
    Traced to: ANF-009
26. **US-026:** Als Nutzer möchte ich Aufgaben nach ihrem Status filtern können, damit ich sicherstellen kann, dass die Aufgaben in der richtigen Reihenfolge erledigt werden.
    Traced to: ANF-009

27. **US-027:** Als Benutzer möchte ich die Person sehen können, die einer Aufgabe zugeordnet ist, damit ich mit der zuständigen Person Kontakt aufnehmen kann.
    Traced to: ANF-009

Für ANF-010: Benachrichtigungen

28. **US-028:** Als Nutzer möchte ich vom System per E-Mail oder in der App benachrichtigt werden, wenn Aufgaben anstehen, damit ich rechtzeitig eingreifen kann, ohne Fristen zu verpassen.
    Traced to: ANF-010

Für ANF-011: Agile Practices

29. **US-029:** Als Product Owner möchte ich, dass mein Team agile Praktiken wie Sprint-Planung, Retrospektiven und tägliche Besprechungen unterstützt, damit wir die Prozesse kontinuierlich verbessern und effektiver arbeiten können.
    Traced to: ANF-011
30. **US-030:** Als Product Owner möchte ich, dass mein Team tägliche Besprechungen durchführt, damit sichergestellt wird, dass das Team nach den agilen Arbeitsgrundsätzen arbeitet.
    Traced to: ANF-011

Für ANF-012: Synchronisation

31. **US-031:** Als Benutzer möchte ich, dass meine Änderungen an einer Aufgabe für alle Teammitglieder in Echtzeit sichtbar sind, damit alle auf dem Laufenden sind und Missverständnisse vermieden werden.
    Traced to: ANF-012

Für ANF-013: Parallele Bearbeitung

32. **US-032:** Als Benutzer möchte ich, dass meine Änderungen an einer Aufgabe für alle Teammitglieder gleichzeitig sichtbar sind, damit alle auf dem Laufenden sind und Missverständnisse vermieden werden.
    Traced to: ANF-013

Für ANF-014: Multi-Nutzer

33. **US-033:** Als Administrator möchte ich, dass mehrere Benutzer gleichzeitig am System arbeiten können, damit parallele Aufgabenbearbeitung ermöglicht wird.
    Traced to: ANF-014

34. **US-034:** Als Administrator möchte ich, dass die Benutzer die Aufgaben flexibel organisieren können, damit die Arbeit im Team strukturiert und nachvollziehbar bleibt.
    Traced to: ANF-014

Für ANF-015: Verfügbarkeit

35. **US-035:** Als Nutzer möchte ich zu jeder Tageszeit auf das System zugreifen und meine Aufgaben bearbeiten können, damit ich flexibel arbeiten kann.
    Traced to: ANF-015

Für ANF-016: Updates

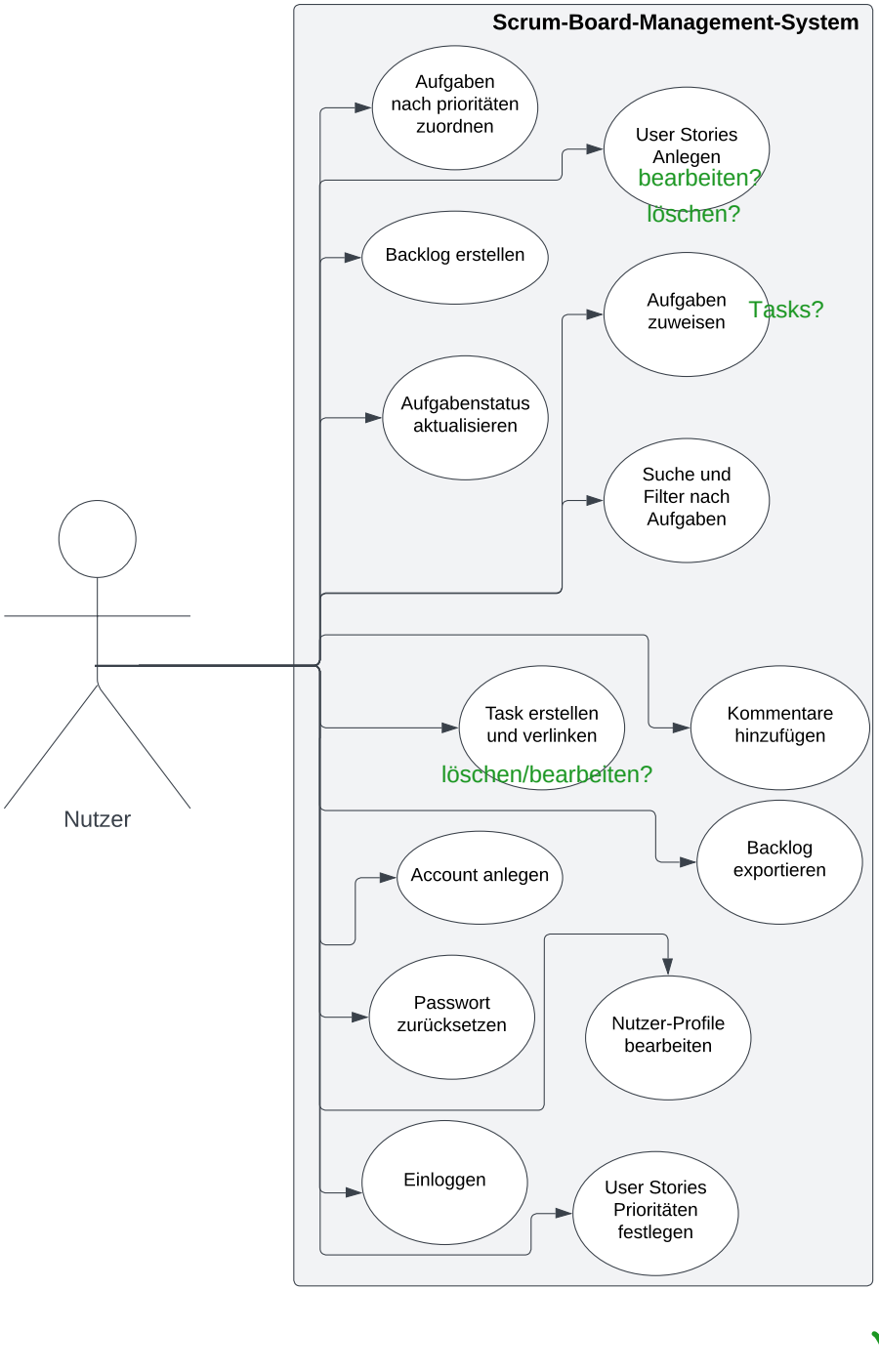
36. **US-036:** Als Benutzer möchte ich, dass Änderungen in den Aufgabenlisten innerhalb von maximal 2 Sekunden synchronisiert und für alle Nutzer sichtbar werden, damit der Projektfortschritt immer aktuell bleibt.
    Traced to: ANF-016

Für ANF-017: Änderungen

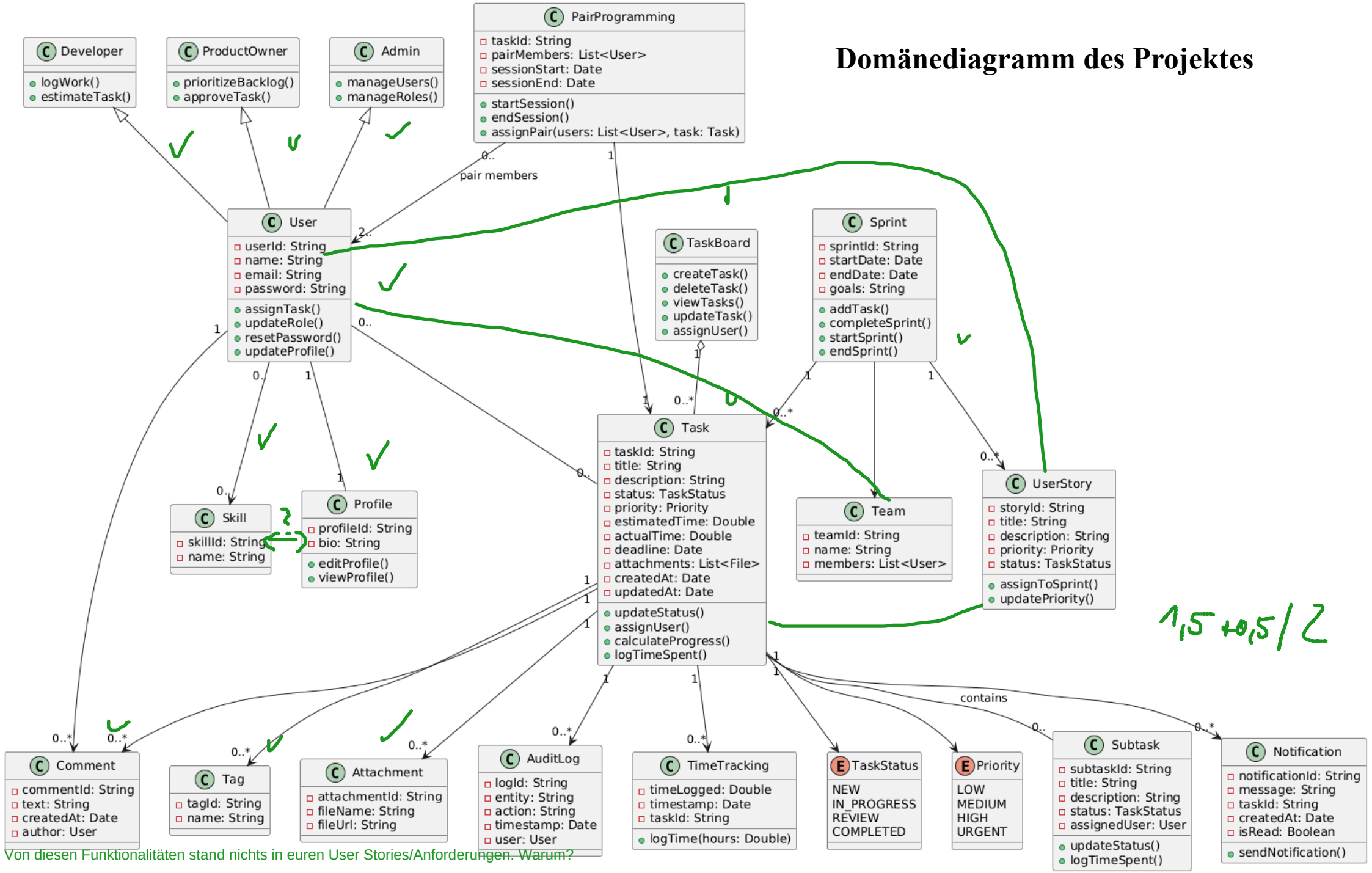
37. **US-037:** Als Entwickler möchte ich, dass das System nach seiner Freigabe leicht erweiterbar ist, sodass ich neue Listentypen hinzufügen kann, damit verschiedene Arten von Aufgaben und Prozessen flexibel abgebildet werden können.
    Traced to: ANF-017
38. **US-038:** Als Entwickler möchte ich, dass das System nach seiner Freigabe leicht erweiterbar ist, sodass ich neue Filteroptionen hinzufügen kann, damit Nutzer relevante Aufgaben schneller finden und priorisieren können.
    Traced to: ANF-017
39. **US-039:** Als Entwickler möchte ich, dass das System nach seiner Freigabe leicht erweiterbar ist, sodass ich neue Benachrichtigungstypen hinzufügen kann, damit das Team besser über wichtige Ereignisse und Fristen informiert wird.
    Traced to: ANF-017

Tracing in Ordnung

Use Case Diagramm



Domänediagramm des Projektes



1,5 +0,5 / 2

Von diesen Funktionalitäten stand nichts in euren User Stories/Anforderungen. Warum?

Begründung des Architekturmusters und der Hauptkomponenten

1. Architekturmuster

Gewähltes Architekturmuster:

Für dieses System wurde das *Layered Architecture*-Muster ausgewählt, da es klare Vorteile in Bezug auf Modularität, Flexibilität und Testbarkeit bietet.

- Modularität:

Durch die Trennung in verschiedene Schichten (Presentation, Business Logic und Data Access) können einzelne Teile des Systems unabhängig voneinander entwickelt und gewartet werden.

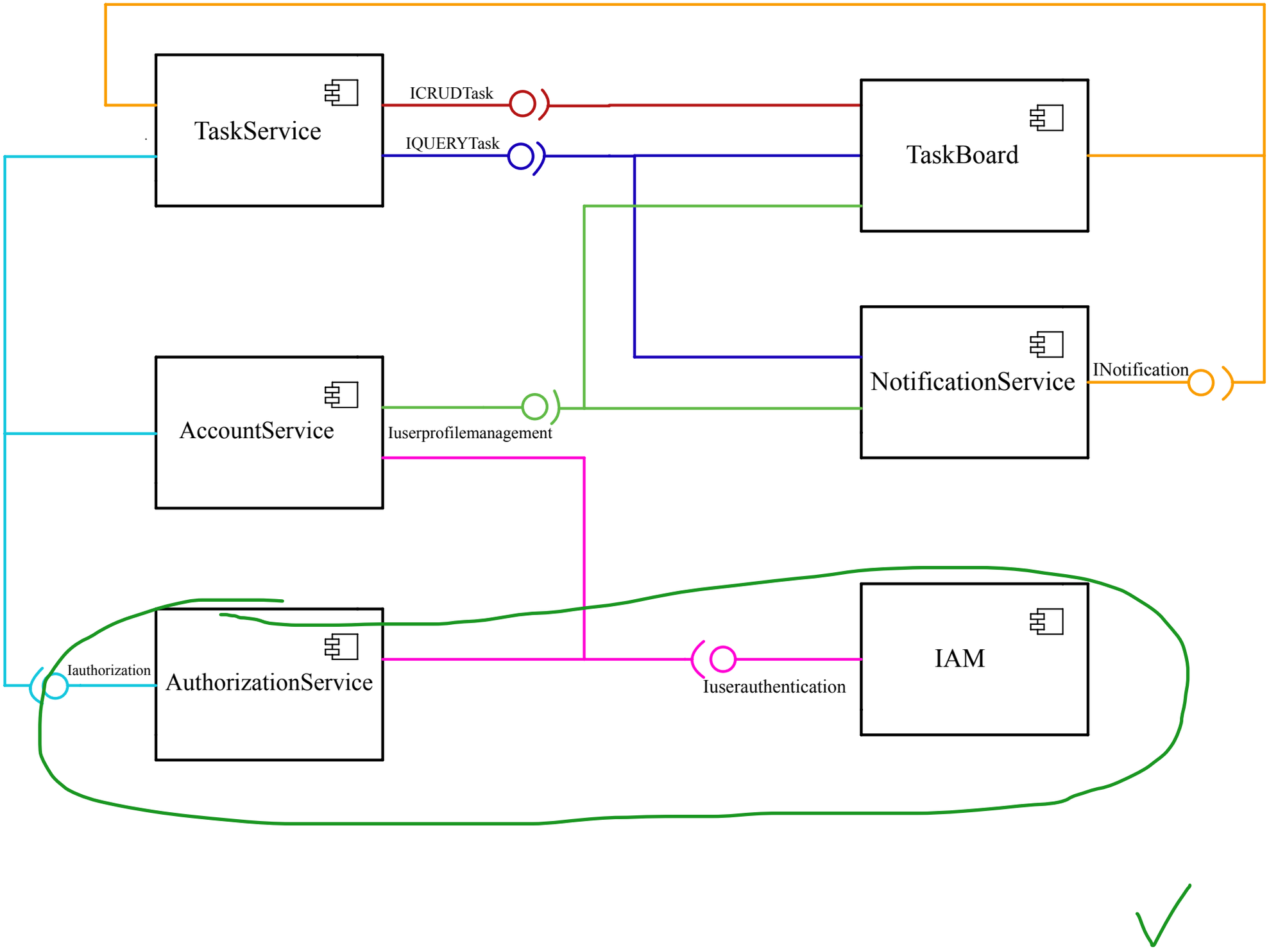
- Flexibilität:

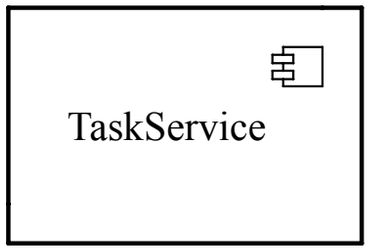
Änderungen in einer Schicht (z. B. Benutzeroberfläche) wirken sich nicht direkt auf andere Schichten (z. B. Datenzugriff) aus. Dies ermöglicht eine einfache Anpassung an zukünftige Anforderungen.

- Testbarkeit:

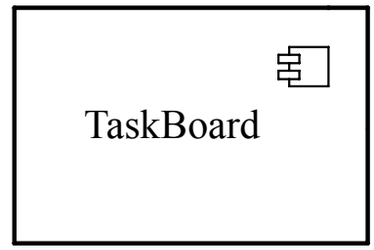
Jede Schicht kann isoliert getestet werden, was die Qualitätssicherung verbessert und Fehler schneller identifizieren lässt.

Softwaretechnik WS 2024-25 / Projekt-Team1-Übung2 / Komponentendiagramm



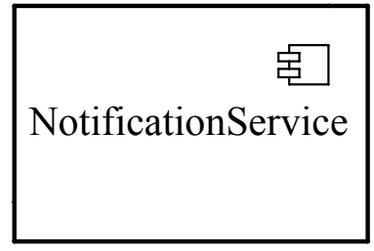


-Verantwortlich für die Tasksmanagement



-Visualisierung von Tasks

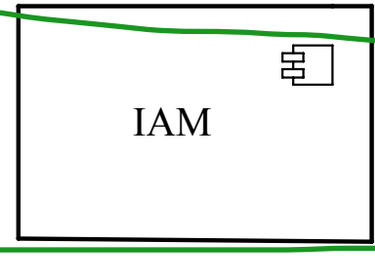
-Verwaltung von User Accounts



-Zuständig für das Versenden von Benachrichtugen

Warum trennt ihr das hier so auf?

-Authorisierung und Ressourcenzugriffskontrolle



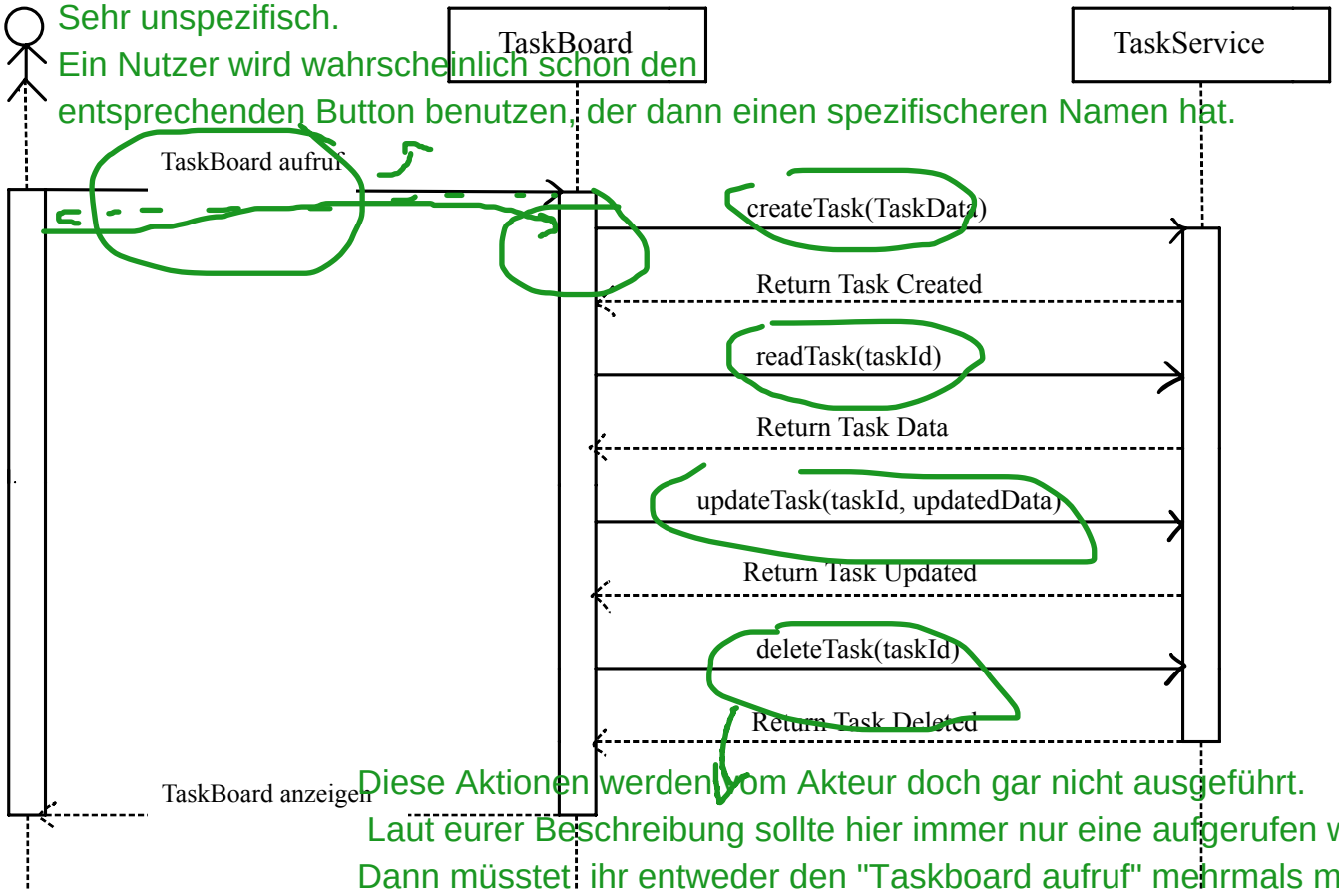
-Identity and an User Access Management

Suche und Filter?

3/3

ICRUDTask

Entwurf 2.2: Interfaces zwischen diesen Komponenten als Sequenzdiagramm



Beschreibung des Sequenzdiagramms für das ICRUDTask-Interface

Das von der TaskService-Komponente bereitgestellte ICRUDTask-Interface enthält die Methoden **createTask(taskData), readTask(taskId), updateTask(taskId, updatedData)** und **deleteTask(taskId)**. Dieses Interface wird von der TaskBoard-Komponente verwendet.

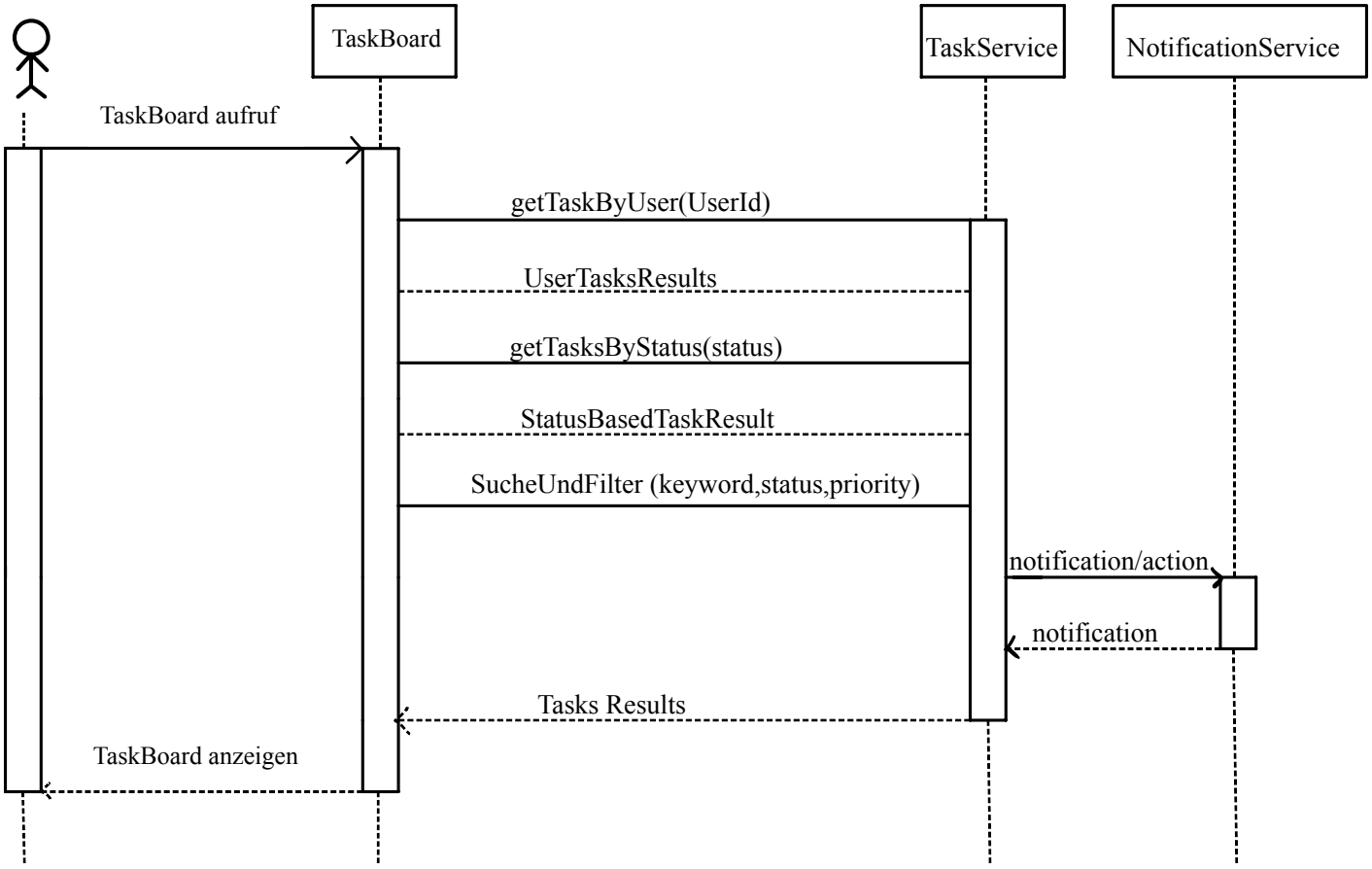
Wenn ein Nutzer über das TaskBoard eine CRUD-Operation an einer Aufgabe durchführen möchte, sendet er eine Botschaft namens Taskboardaufruf an das TaskBoard. Das TaskBoard leitet diese Botschaft je nach Art der Operation an die von der TaskService-Komponente bereitgestellten Methoden weiter.

Als Antwort auf die Botschaften gibt die TaskService-Komponente folgende Rückmeldungen zurück:

- Created Task (für die Methode createTask),
- Task Data (für die Methode readTask),
- Updated Task (für die Methode updateTask),
- Deleted Task (für die Methode deleteTask).

Diese Rückmeldungen ermöglichen es dem TaskBoard, dem Nutzer eine abschließende Rückmeldung in Form von Taskboard anzeigen zu geben.

IQUERYTask



Gleiches Problem wie beim ersten Diagramm

Beschreibung des Sequenzdiagramms für das IQueryTasks-Interface

Das von der TaskService-Komponente bereitgestellte IQueryTasks-Interface enthält die Methoden:

- getTasksByUser(userId),
- getTasksByStatus(status),
- sucheUndFilter(keyword, status, priority).

Dieses Interface wird von der TaskBoard-Komponente und der NotificationService-Komponente verwendet.Wenn ein Nutzer über das TaskBoard eine der oben genannten Operationen durchführen möchte, sendet er eine Botschaft namens Taskboardaufruf an das TaskBoard. Das TaskBoard leitet diese Botschaft je nach Art der Operation an die von der TaskService-Komponente bereitgestellten Methoden weiter:

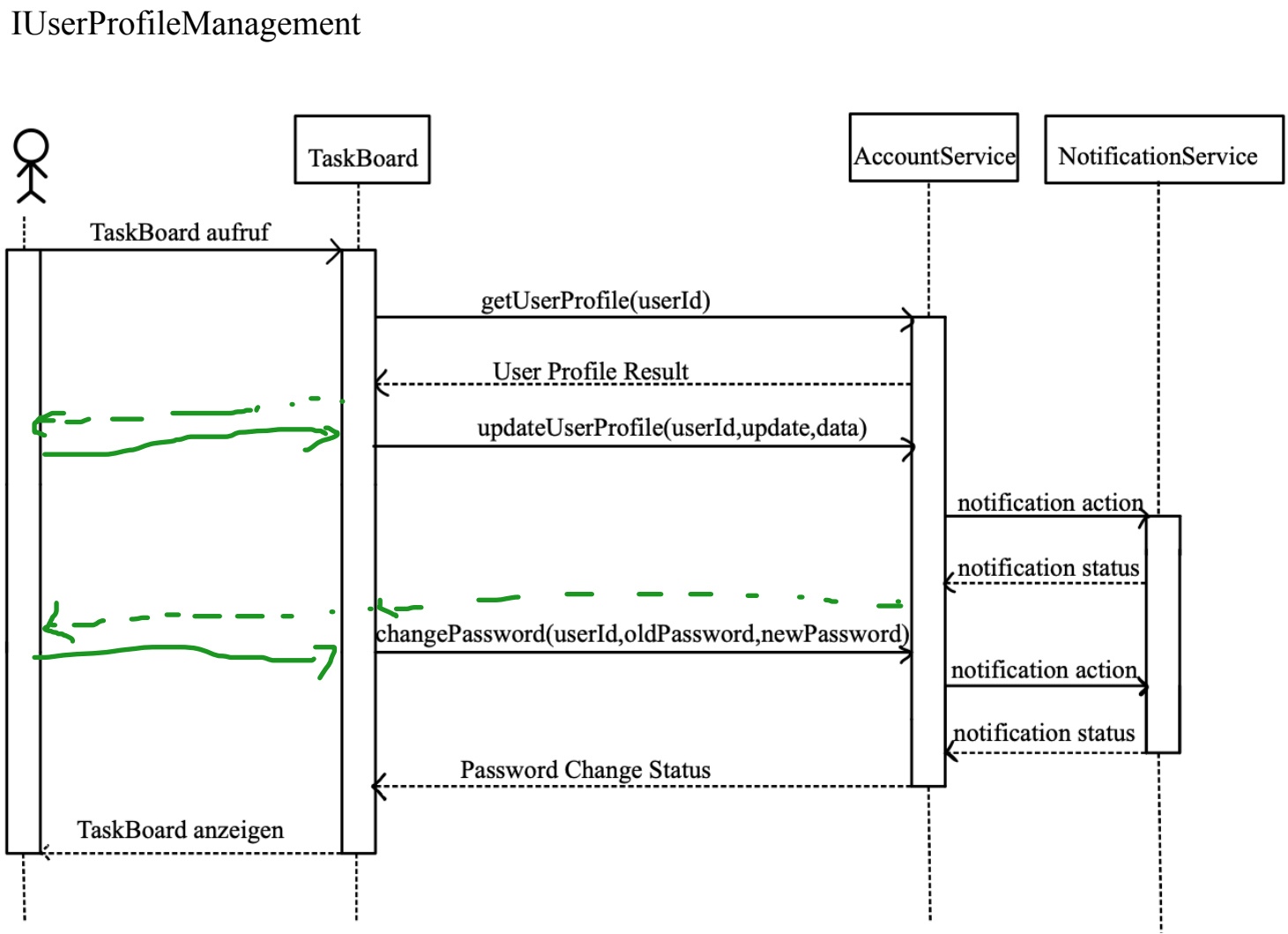
1. **getTasksByUser(userId):**
   - Diese Methode dient dazu, alle Aufgaben zu ermitteln, die einem bestimmten Nutzer zugeordnet sind.
   - Das TaskBoard sendet die Botschaft an die TaskService-Komponente, die daraufhin die User Tasks Result zurückgibt.
2. **getTasksByStatus(status):**
   - Methode ermöglicht es, Aufgaben nach ihrem Status zu filtern (z. B. „in Bearbeitung“, „abgeschlossen“).
   - Das TaskBoard sendet die Botschaft an die TaskService-Komponente, die daraufhin die Status Based Tasks Result zurückgibt.
3. **sucheUndFilter(keyword, status, priority):**
   - Diese Methode kombiniert eine Suche nach Schlüsselwörtern mit Filtern für Status und Priorität.
   - Aufgrund der Komplexität dieser Operation wird die Botschaft von der TaskService-Komponente an die NotificationService-Komponente weitergeleitet, um Benachrichtigungen zu generieren.
   - Die NotificationService-Komponente gibt eine Rückmeldung namens Notification Status zurück.

Erklärung der Einbindung der NotificationService-Komponente

Das verstehe ich nicht.

Die NotificationService-Komponente wird nur bei der Methode sucheUndFilter verwendet, da diese Methode potenziell komplexe oder kritische Such- und Filteroperationen beinhaltet, die zusätzliche Benachrichtigungen für die Nutzer erfordern könnten (z. B. bei besonders dringenden Aufgaben). Die Methoden getTasksByUser und getTasksByStatus hingegen generieren ausschließlich Datenabfragen, bei denen Benachrichtigungen nicht notwendig sind.

Das TaskBoard erhält alle Rückmeldungen (z. B. User Tasks Result, Status Based Tasks Result, Notification Status) von den zuständigen Komponenten. Auf Basis dieser Rückmeldungen ist das TaskBoard in der Lage, dem Nutzer eine abschließende Rückmeldung in Form von Taskboard anzeigen zu geben.



Beschreibung des Sequenzdiagramms für das IUserProfileManagement-Interface

Das von der AccountService-Komponente bereitgestellte IUserProfileManagement-Interface enthält die folgenden Methoden:

- getUserProfile(userId)
- updateUserProfile(userId, updatedData)
- changePassword(userId, oldPassword, newPassword)

Dieses Interface wird von der TaskBoard-Komponente und der NotificationService-Komponente verwendet.

Ablauf im Sequenzdiagramm
Wenn ein Nutzer über das TaskBoard eine der oben genannten Operationen durchführen möchte, sendet er eine Botschaft namens Taskboardaufruf an das TaskBoard. Das TaskBoard leitet diese Botschaft je nach Art der Operation an die von der AccountService-Komponente bereitgestellten Methoden weiter:

1. **getUserProfile(userId):**

   - Diese Methode wird verwendet, um die Profildaten eines bestimmten Nutzers abzurufen.
   - Das TaskBoard sendet die Botschaft an die AccountService-Komponente, die daraufhin die User Profile Result zurückgibt.
   - Warum keine Einbindung von NotificationService?: Diese Methode ruft lediglich bestehende Daten ab und führt keine Änderungen oder kritischen Operationen aus. Benachrichtigungen sind daher nicht notwendig.

2. **updateUserProfile(userId, updatedData):**

   - Diese Methode dient dazu, die Profildaten eines Nutzers zu aktualisieren.
   - Das TaskBoard leitet die Botschaft an die AccountService-Komponente weiter. Die AccountService-Komponente führt die Aktualisierung durch und sendet die Rückmeldungen:
   - Updated User Profile Result (Bestätigung der erfolgreichen Aktualisierung).
   - Notification Action: Zusätzlich wird die Botschaft an die NotificationService-Komponente gesendet, um eine Benachrichtigung über die Aktualisierung zu generieren. Die NotificationService-Komponente gibt daraufhin den Notification Status zurück.

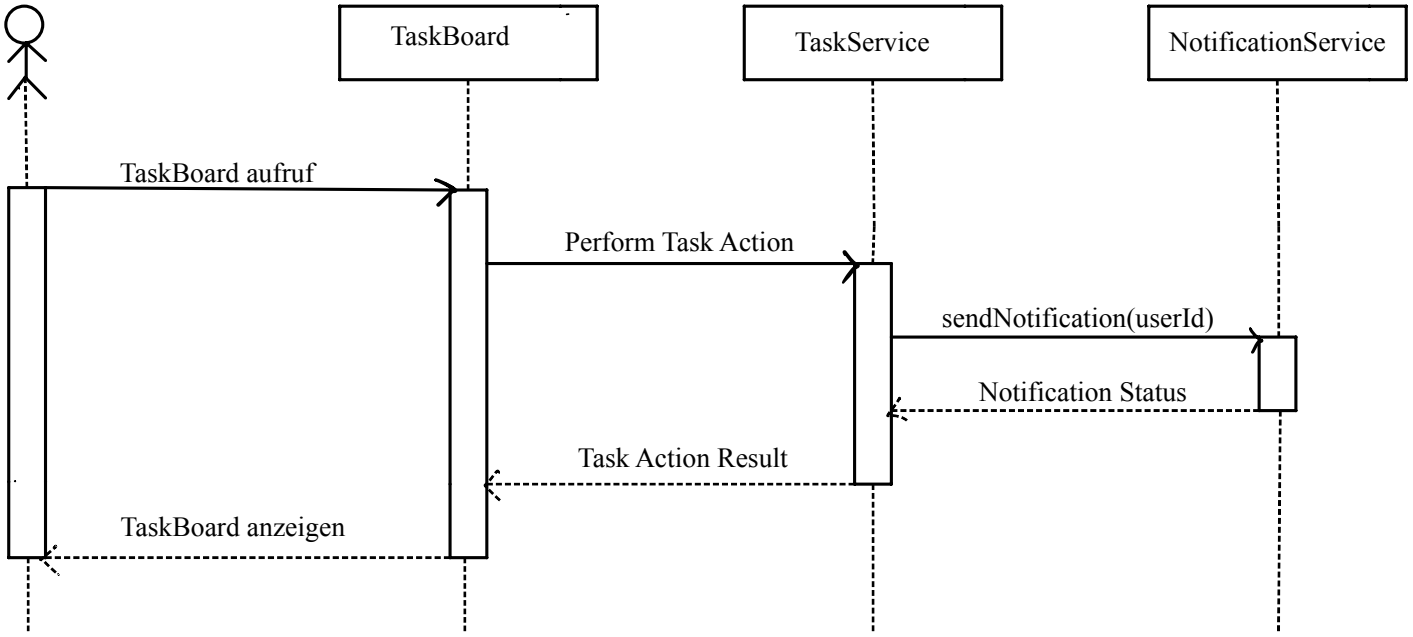
3. **changePassword(userId, oldPassword, newPassword):**

   - Diese Methode ermöglicht es einem Nutzer, sein Passwort zu ändern.
   - Das TaskBoard sendet die Botschaft an die AccountService-Komponente. Nach erfolgreicher Durchführung der Passwortänderung sendet die AccountService-Komponente die Rückmeldungen:
   - Password Change Status (Bestätigung der Passwortänderung).
   - Notification Action: Auch hier wird die Botschaft an die NotificationService-Komponente weitergeleitet, um den Nutzer über die Änderung zu informieren. Die NotificationService-Komponente gibt den Notification Status zurück.

Erklärung der Einbindung der NotificationService-Komponente
Die NotificationService-Komponente wird in den Methoden updateUserProfile und changePassword eingebunden, da diese Operationen Änderungen an sensiblen Daten eines Nutzers beinhalten. Diese Änderungen könnten für den Nutzer von hoher Bedeutung sein und erfordern daher eine Benachrichtigung. Die Methode getUserProfile hingegen ruft lediglich vorhandene Daten ab, was keine Benachrichtigung erforderlich macht.

Das TaskBoard empfängt alle Rückmeldungen (z. B. User Profile Result, Updated User Profile Result, Password Change Status, Notification Status) von den zuständigen Komponenten. Auf Basis dieser Rückmeldungen ist das TaskBoard in der Lage, dem Nutzer eine abschließende Rückmeldung in Form von Taskboard anzeigen zu geben.

INotification



Beschreibung des Sequenzdiagramms für das INotification-Interface

Das von der NotificationService-Komponente bereitgestellte INotification-Interface enthält die Methode:

- sendNotification(userId, message)

Dieses Interface wird sowohl von der TaskBoard-Komponente als auch von der TaskService-Komponente verwendet.

Ablauf im Sequenzdiagramm

Wenn ein Nutzer über das TaskBoard eine Operation durchführen möchte, die eine Benachrichtigung erfordert, wird der folgende Ablauf dargestellt:

1. Der Nutzer sendet eine Botschaft namens Taskboardaufruf an das TaskBoard.
    - Diese Botschaft signalisiert, dass eine Aktion auf der Ebene des TaskBoards durchgeführt werden soll.
2. Das TaskBoard leitet die Botschaft in Form von perform Task Action an die TaskService-Komponente weiter.
    - Die TaskService-Komponente ist für die Verarbeitung der spezifischen Aufgabenlogik verantwortlich.
3. Innerhalb der TaskService-Komponente wird die Methode sendNotification(userId, message) des INotification-Interfaces aufgerufen.
    - Diese Botschaft wird an die NotificationService-Komponente gesendet, um eine Benachrichtigung zu generieren.
    - sendNotification(userId, message): Diese Methode dient dazu, eine Benachrichtigung mit einer bestimmten Nachricht an den definierten Nutzer zu senden.

4. Die NotificationService-Komponente verarbeitet die Anfrage und gibt eine Rückmeldung namens Notification Status an die TaskService-Komponente zurück.

- Diese Rückmeldung bestätigt den Status der Benachrichtigung (z. B. erfolgreich gesendet, Fehler, etc.).

5. Basierend auf der Rückmeldung von der NotificationService-Komponente gibt die TaskService-Komponente eine weitere Rückmeldung namens Task Action Result an das TaskBoard zurück.

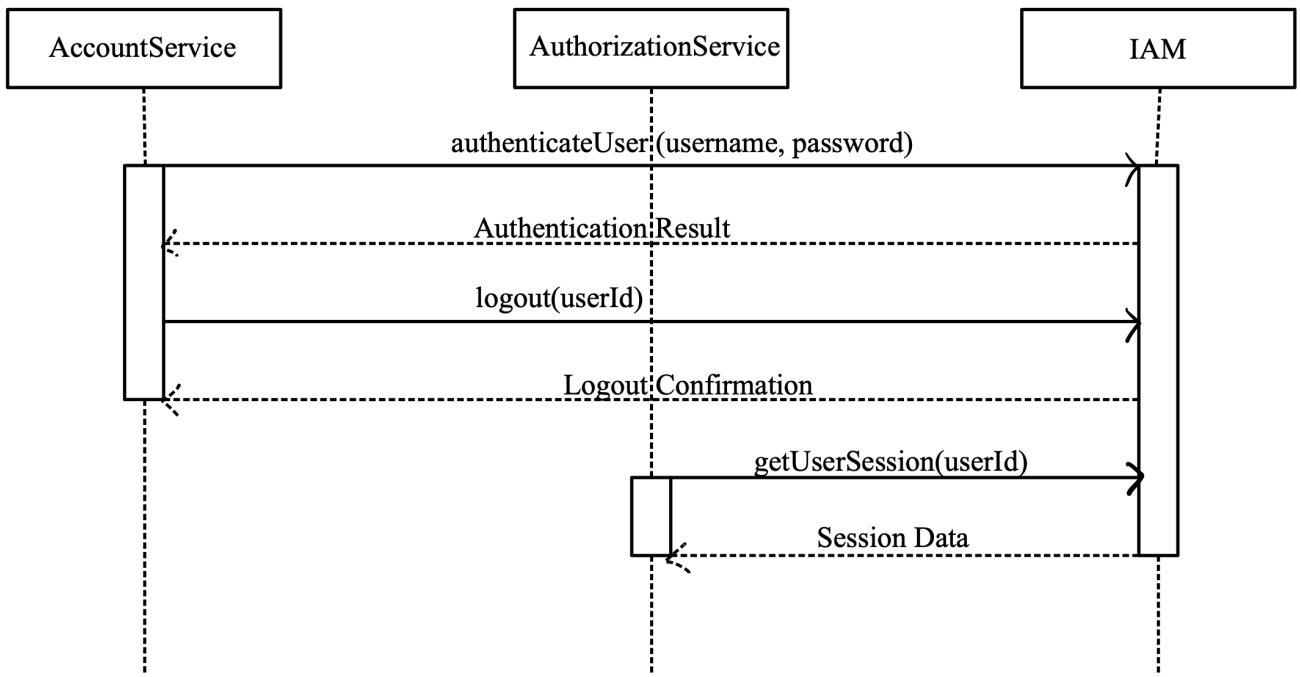
6. Das TaskBoard verwendet diese Rückmeldung, um dem Nutzer eine finale Rückmeldung in Form von Taskboard anzeigen zu geben.

Erklärung der Einbindung der NotificationService-Komponente

Die NotificationService-Komponente wird eingebunden, wenn eine Benachrichtigung erforderlich ist. Dies ist besonders bei Operationen relevant, die die Nutzer direkt betreffen (z. B. Statusänderungen, neue Aufgaben, oder wichtige Benachrichtigungen). Sie wird verwendet, um sicherzustellen, dass relevante Nachrichten die Nutzer rechtzeitig erreichen.

Das TaskBoard empfängt alle Rückmeldungen (z. B. Notification Status, Task Action Result) von den zuständigen Komponenten. Auf Basis dieser Rückmeldungen ist das TaskBoard in der Lage, dem Nutzer eine abschließende Rückmeldung in Form von Taskboard anzeigen zu geben.

IUserAuthentication



Beschreibung des Sequenzdiagramms für das IUserAuthentication-Interface

Das von der IAM-Komponente bereitgestellte IUserAuthentication-Interface enthält die folgenden Methoden:

- authenticateUser(username, password)
- logout(userId)
- getUserSession(userId)

Dieses Interface wird von der AccountService-Komponente und der AuthorizationService-Komponente verwendet.

Ablauf im Sequenzdiagramm

Das Sequenzdiagramm zeigt die Interaktionen zwischen den Komponenten AccountService, AuthorizationService und IAM im Rahmen des IUserAuthentication-Interfaces. Ein Nutzer wird hier nicht benötigt, da die Operationen ausschließlich zwischen diesen drei Komponenten stattfinden.

1. authenticateUser(username, password):

- Die AccountService-Komponente sendet die Botschaft authenticateUser(username, password) an die IAM-Komponente.
- Zweck der Botschaft: Diese Methode dient der Überprüfung der Anmeldedaten eines Nutzers (z. B. bei der Anmeldung). Die IAM-Komponente führt die Authentifizierung durch und gibt eine Rückmeldung namens Authentication Result an die AccountService-Komponente zurück.
- Diese Rückmeldung enthält Informationen darüber, ob die Authentifizierung erfolgreich war oder nicht.

2. logout(userId):

- Die AccountService-Komponente sendet die Botschaft logout(userId) an die IAM-Komponente.
- Zweck der Botschaft: Diese Methode beendet die Sitzung eines angemeldeten Nutzers, indem die zugehörigen Sitzungsdaten gelöscht oder invalidiert werden.
- Die IAM-Komponente gibt daraufhin eine Rückmeldung namens Logout Confirmation an die AccountService-Komponente zurück, die die erfolgreiche Abmeldung bestätigt.

3. getUserSession(userId):

- Die AuthorizationService-Komponente sendet die Botschaft getUserSession(userId) an die IAM-Komponente.
- Zweck der Botschaft: Diese Methode ruft die Sitzungsinformationen eines bestimmten Nutzers ab, um dessen Berechtigungen oder Status zu überprüfen.
- Die IAM-Komponente gibt eine Rückmeldung namens Session Data an die AuthorizationService-Komponente zurück, die die angeforderten Sitzungsinformationen enthält.

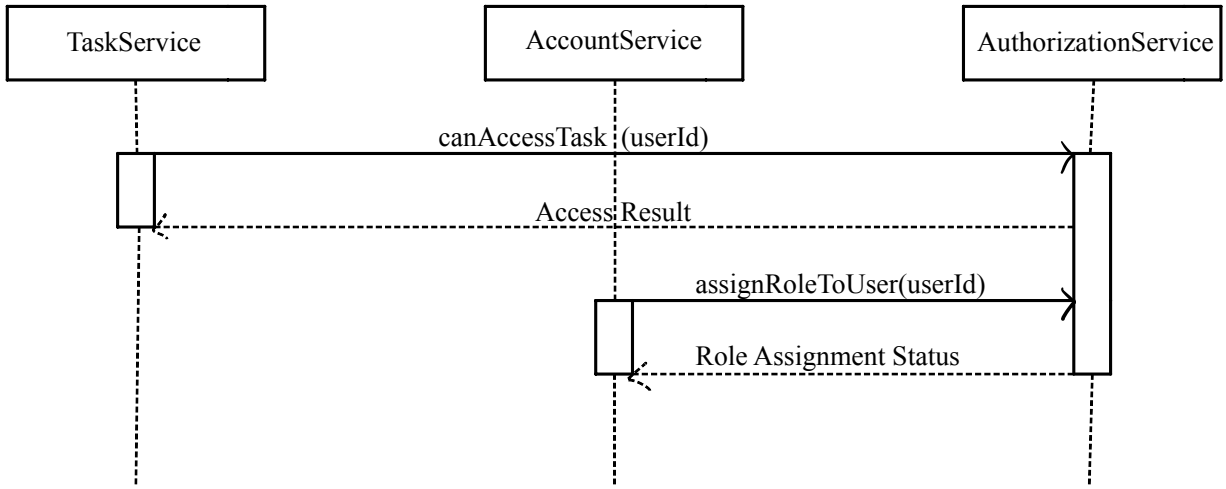
Warum ist kein Nutzer im Sequenzdiagramm notwendig?

Okay, aber ein Nutzer muss sich doch an irgendeinem Punkt anmelden?

Ein Nutzer wird in diesem Sequenzdiagramm nicht dargestellt, da die Operationen, die das IUserAuthentication-Interface betreffen, ausschließlich zwischen den technischen Komponenten (AccountService, AuthorizationService und IAM) stattfinden.
Das TaskBoard oder andere Nutzerinteraktionen sind nicht involviert, da das Interface auf Authentifizierungs- und Sitzungsoperationen auf technischer Ebene beschränkt ist. Diese Prozesse sind für die direkte Kommunikation zwischen den Komponenten erforderlich, ohne dass ein Nutzer direkt eingreifen muss.

Das Sequenzdiagramm zeigt klar die Interaktionen zwischen den drei Komponenten und die entsprechenden Rückmeldungen, die von der IAM-Komponente bereitgestellt werden. Auf diese Weise wird die Funktionalität des IUserAuthentication-Interfaces präzise und nachvollziehbar dargestellt.

IAuthorization



Beschreibung des Sequenzdiagramms für das IAuthorization-Interface

Das von der AuthorizationService-Komponente bereitgestellte IAuthorization-Interface enthält die folgenden Methoden:

- canAccessTask(userId)
- assignRoleToUser(userId)

Dieses Interface wird von der TaskService-Komponente und der AccountService-Komponente verwendet.

Ablauf im Sequenzdiagramm

Das Sequenzdiagramm zeigt die Interaktionen zwischen den Komponenten TaskService, AccountService und AuthorizationService im Rahmen des IAuthorization-Interfaces. Ein Nutzer wird hier nicht dargestellt, da die Operationen ausschließlich zwischen diesen drei Komponenten stattfinden.

1. canAccessTask(userId):

- Die TaskService-Komponente sendet die Botschaft canAccessTask(userId) an die AuthorizationService-Komponente.
- Zweck der Botschaft: Diese Methode überprüft, ob ein bestimmter Nutzer Zugriff auf eine bestimmte Aufgabe hat. Dies ist relevant, um sicherzustellen, dass nur autorisierte Nutzer auf bestimmte Aufgaben zugreifen können.
- Die AuthorizationService-Komponente verarbeitet die Anfrage und gibt eine Rückmeldung namens Access Result an die TaskService-Komponente zurück.
- Diese Rückmeldung enthält Informationen darüber, ob der Zugriff erlaubt oder verweigert wird.

2. assignRoleToUser(userId):

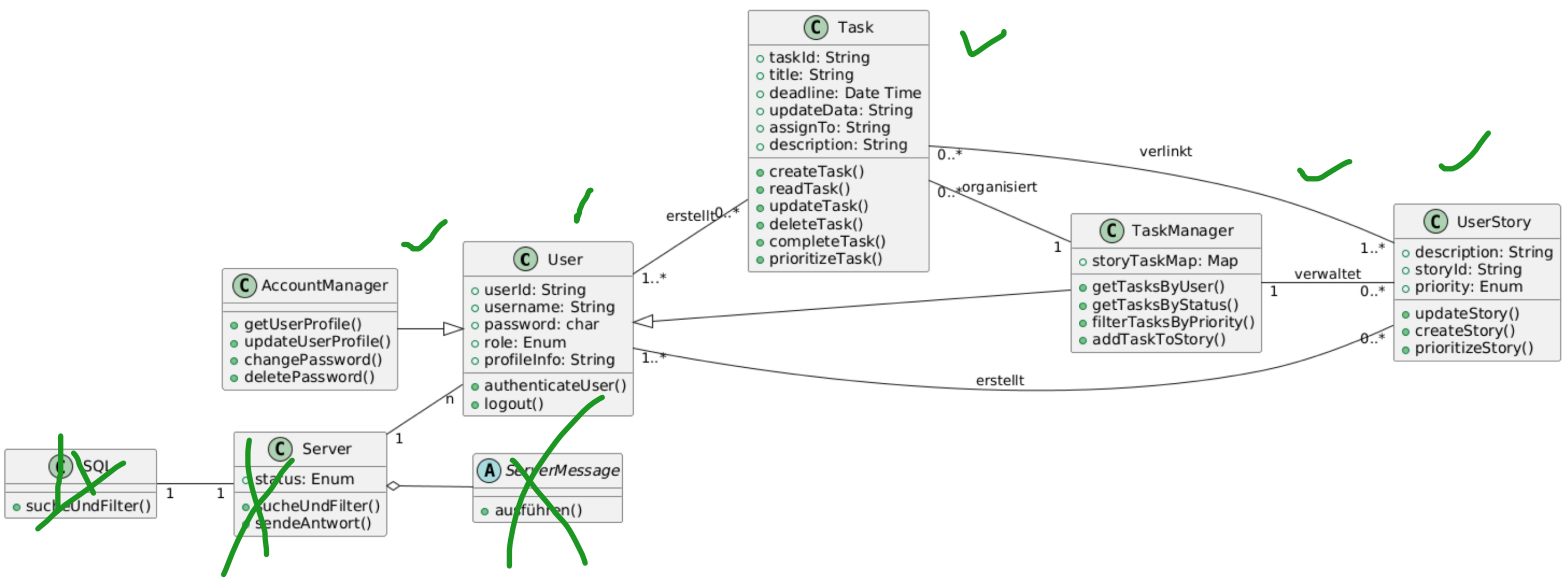
- Die AccountService-Komponente sendet die Botschaft assignRoleToUser(userId) an die AuthorizationService-Komponente.
- Zweck der Botschaft: Diese Methode weist einem bestimmten Nutzer eine Rolle zu, die seine Berechtigungen innerhalb des Systems definiert.
- Die AuthorizationService-Komponente verarbeitet die Anfrage und gibt eine Rückmeldung namens Role Assignment Status an die AccountService-Komponente zurück.
- Diese Rückmeldung bestätigt, ob die Rollenvergabe erfolgreich war oder ob ein Fehler aufgetreten ist.

Warum ist kein Nutzer im Sequenzdiagramm notwendig? aber ein Nutzer initiiert die Interaktion doch?

Ein Nutzer wird in diesem Sequenzdiagramm nicht dargestellt, da die Operationen, die das IAuthorization-Interface betreffen, ausschließlich zwischen den technischen Komponenten (TaskService, AccountService und AuthorizationService) stattfinden.
Das TaskBoard oder andere Nutzerinteraktionen sind nicht involviert, da dieses Interface auf Berechtigungsprüfungen und Rollenvergaben auf technischer Ebene beschränkt ist. Diese Prozesse erfordern keine direkte Beteiligung eines Nutzers, da die Aufrufe systemintern erfolgen.

1/2

Entwurf 2.3 : Datenstrukturen



Warum sind hier viel weniger Klassen als im Domänenmodell?
Die müssten sich doch hier größtenteils auch wiederfinden? Bspw. Kommentare/Tags

1/2

Welche Datenstrukturen werden gespeichert und welche werden an den Interfaces genutzt?

Projektabgabedokument

Entwurf 3.1: Technologien

Technologien, Datenbanken, Bibliotheken und Frameworks

Soweit absehbar, sind dies die Technologien, Datenbanken, Bibliotheken und Frameworks, die im Laufe der Projektentwicklung implementiert oder ausprobiert werden sollen sind die folgenden:

1 Programmiersprache: Java mit Spring Boot

Java wurde als Programmiersprache gewählt, da die objektorientierte Struktur die Softwareentwicklung erleichtert und die Sprache den Teammitgliedern bekannt ist. Um den Projektstart und das Management zu optimieren, wird das Open-Source-Framework **Spring Boot** eingesetzt, das eine Vielzahl an Bibliotheken mitbringt und damit die Entwicklung deutlich vereinfacht.

2 Datenbank: MySQL

Für die Datenbank wurde **MySQL** ausgewählt. MySQL ist ein weit verbreitetes und zuverlässiges relationales Datenbankmanagementsystem, das durch seine hohe Performance und Skalierbarkeit überzeugt. Es ermöglicht eine einfache Integration mit Java-Anwendungen und unterstützt die effiziente Speicherung und Abfrage von Daten.

3 Agile Methode: Scrum

Als agile Methode wurde **Scrum** gewählt, da es durch seine flexible und iterative Struktur teamorientiertes Arbeiten mit regelmäßigen Entwicklungszyklen ermöglicht, sodass wir den Entwicklungsprozess als Team problemlos einsehen und bearbeiten können.

4 Kommunikationstool: Discord

Die Kommunikation zwischen den Entwicklern erfolgt über **Discord**, da es eine unkomplizierte App ist, mit der alle Teammitglieder bereits vertraut sind. Discord bietet nützliche Funktionen wie Bildschirmfreigabe und den relativ einfachen Austausch von Dateien.

5 Versionskontrolle: Git mit GitLab

Git zusammen mit **GitLab** wird als Versionskontrolle für eine effektive Verwaltung von Daten und Code verwendet. Der Grund dafür ist, dass Teammitglieder sich gut mit dem Paket auskennen und es bereits für verschiedene Aufgaben genutzt haben.

V

1/1

8/10