MODUL 223

Projekt von Jan G.

Inhaltsverzeichnis

Task- & Project-Manager

Projektbeschreibung:

Die Task- & Projekt-Managementapp dient zum Verwalten von Aufgaben, Notizen und Projekten (vergleichbar mit Trello). Auf dieser App kann man sich anmelden und eigene Aufgaben erstellen, bearbeiten, löschen und teilen.

User Stories:

User Story 1:

Als registrierter Benutzer möchte ich Aufgaben/Notizen erstellen und löschen können, weil ich die Projekte so im Team verwalten kann.

Akzeptanzkriterien:

- Der Benutzer muss eingeloggt sein.
- Eine Aufgabe kann über ein Formular mit Titel und Beschreibung erstellt werden.
- Nur der Ersteller oder ein Projektadministrator kann die Aufgabe löschen.
- Nach dem Löschen verschwindet die Aufgabe aus dem UI und der Datenbank.

User Story 2:

Als Projektmitglied möchte ich den Status einer Aufgabe (z. B. "offen", "in Bearbeitung", "erledigt") ändern können, weil alle Teammitglieder den aktuellen Fortschritt einsehen können sollen.

Akzeptanzkriterien:

- Nur eingeloggte Mitglieder des Projekts dürfen Aufgabenstatus ändern.
- Der Status kann per Dropdown oder Drag-and-Drop geändert werden.
- Die Änderung wird in der Datenbank gespeichert.

User Story 3:

Als eingeloggter Benutzer möchte ich Aufgaben bestimmten Projektmitgliedern zuweisen, weil wir so unsere Arbeit im Team besser koordinieren können.

Akzeptanzkriterien:

- Nur eingeloggte Projektmitglieder dürfen Aufgaben zuweisen.
- Die zugewiesenen Personen werden im Aufgabenfeld sichtbar angezeigt.
- Die Zuweisung ist in der Datenbank gespeichert.

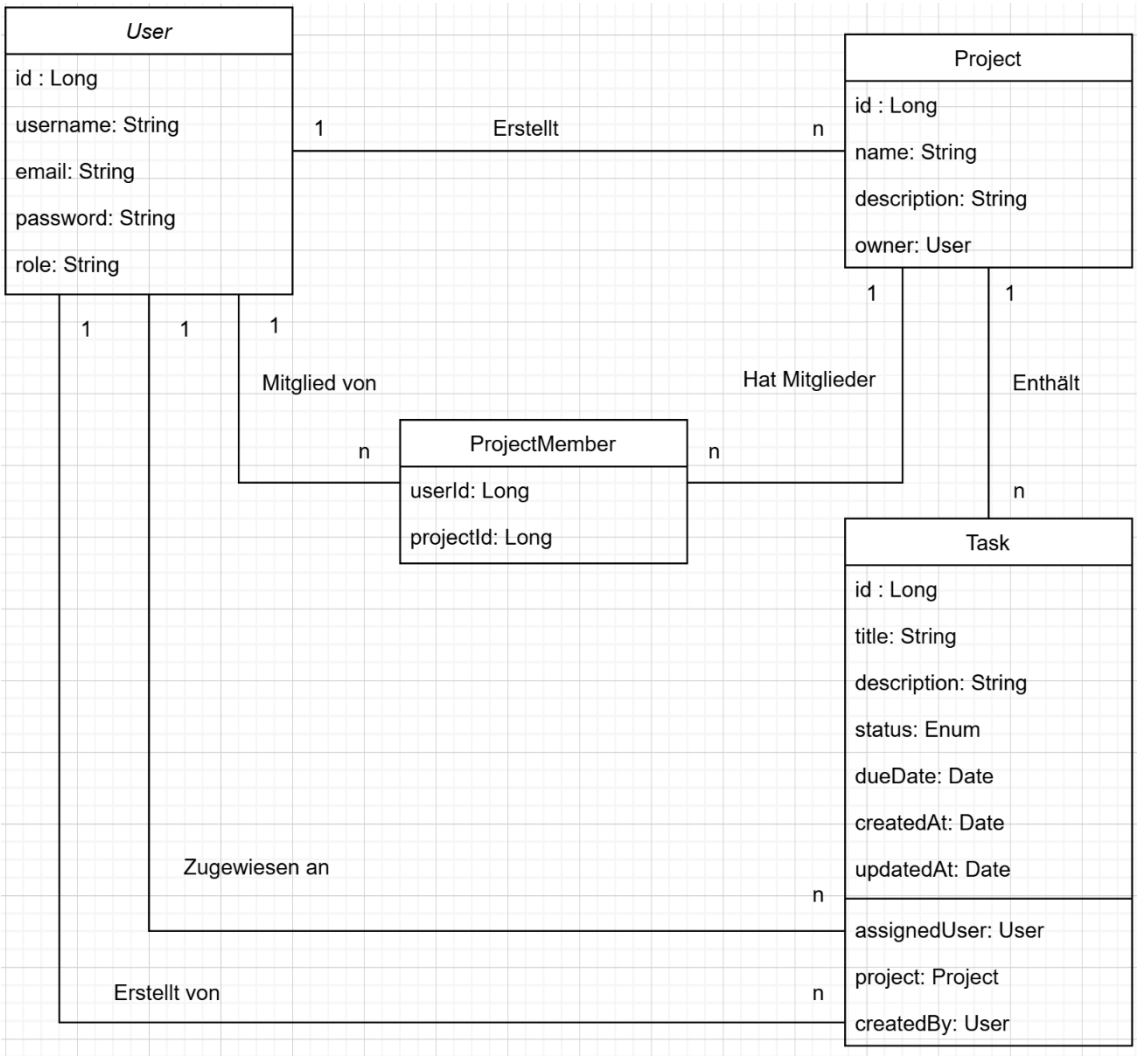
User Story 4:

Als angemeldeter Benutzer möchte ich Aufgaben/Notizen bearbeiten und anpassen können, weil ich so aktualisierte Aufgaben und Projekte verändern und verwalten kann.

Akzeptanzkriterien:

- Nur eingeloggte Projektmitglieder dürfen Aufgaben bearbeiten.
- Titel, Beschreibung und Deadline sind editierbar.
- Änderungen werden in der Datenbank gespeichert und direkt im UI aktualisiert.

Datenmodell



Arbeitsplan:

Arbeitsschritte	Erwarteter Zeitaufwand	Tatsächlicher Zeitaufwand
Projektbeschreibung	5min	5min
Git Repository erstellen	5min	5min
User Stories	20min	20min
Datenmodell erstellen	25min	30min
Arbeitsplan erstellen	20min	25min
Backend dokumentieren	35min	40min
Transaktionen dokumentiert	25min	20min
Frontend dokumentieren	30min	35min
Umsetzung Projekte (Backend)	200min	190min
Umsetzung Projekte Nutzerverwaltung (Backend)	100min	70min
Umsetzung Aufgaben (Backend)	200min	180min
Umsetzung Aufgabenzuordnung (Backend)	70min	60min
Umsetzung Aufgaben Nutzerverwaltung (Backend)	100min	60min
Umsetzung Projektseite (Frontend)	80min	---
Umsetzung Aufgabenansicht (Frontend)	80min	---
Umsetzung Design (Frontend)	70min	50min
Backend-Testprotokoll erstellen	35min	35min
Frontend-Testprotokoll erstellen	35min	35min
Sicherheitskonzept dokumentieren	35min	30min

Arbeitsschritte	Erwarteter Zeitaufwand	Tatsächlicher Zeitaufwand
Soll-Ist-Vergleich erstellen	25min	20min
Problemanalyse	30min	35min
Arbeitsjournal	65min	70min
Insgesamt:	**21h 30min**	**16h 55min**

Backend-Planung

REST-API Endpoints & HTTP-Methoden:

Projektverwaltung:

Endpoint	Methode	Beschreibung	Rolle
/projects	POST	Neues Projekt erstellen	User
/projects	GET	Alle eigenen Projekte abrufen	User
/projects/{id}	GET	Details eines Projekts	User
/projects/{id}	PUT	Projekt aktualisieren	Admin
/projects/{id}	DELETE	Projekt löschen	Admin
/projects/{id}/members	POST	Nutzer zu einem Projekt hinzufügen	Admin
/projects/{id}/members/{userId}	DELETE	Nutzer entfernen	Admin

Aufgabenverwaltung:

Endpoint	Methode	Beschreibung	Rolle
/projects/{projectId}/tasks	GET	Aufgaben des Projekts abrufen	User (wenn im Projekt)
/projects/{projectId}/tasks	POST	Neue Aufgabe erstellen	User (wenn im Projekt)
/tasks/{taskId}	GET	Einzelne Aufgabe anzeigen	User (wenn im Projekt)
/tasks/{taskId}	PUT	Aufgabe bearbeiten	User (wenn im Projekt)
/tasks/{taskId}	DELETE	Aufgabe löschen	User (Ersteller der Aufgabe) / Admin (wenn im Projekt)
/tasks/{taskId}/assign	POST	Aufgabe zuweisen	User (Ersteller der Aufgabe) / Admin (wenn im Projekt)
/tasks/{taskId}/status	PATCH	Status ändern	User (Ersteller der Aufgabe) / Admin (wenn im Projekt)

Rollen und Benutzern:

USER – normaler Nutzer

ADMIN – kann alle Projekte bearbeiten

Transaktionen:

Durch das Setzen von @Transactional auf Service-Methoden wird sichergestellt, dass Datenbankoperationen in einer konsistenten Transaktion ausgeführt werden.

Leseoperationen sind mit (readOnly = true) annotiert, um Performance-Optimierungen durch den JPA Provider zu ermöglichen.

Schreiboperationen (Erstellen, Aktualisieren, Löschen) laufen in einer Standard-Transaktion, die bei einem Fehler komplett zurückgerollt wird.

Controller enthalten keine Transaktionslogik, um die Trennung von Geschäftslogik und Präsentationslogik sicherzustellen.

Die Endpunkte müssen Transaktionssicher sein, damit nicht jeder die Projekte und Aufgaben nach belieben bearbeiten kann.

Backend-Testprotokoll

Testname	Testbeschreibung	Eingabe / Setup	Erwartetes Ergebnis	Tatsächliches Ergebnis	Status
Project Service-Test	Testet, ob ein Projekt korrekt erstellt wird	ProjectCreateDTO mit gültigem Namen, Beschreibung, OwnerId=1	HTTP 200, Projekt mit ID und Daten	wie erwartet	Passed
Task Controller-Test	POST /api/tasks mit gültigem TaskCreateDTO	JSON mit title, description, projectId, assignedUserId, createdByUserId = 1	HTTP 200, Task mit ID im Response	HTTP 400 (fehlt Feld/Mock?)	Failed

Frontend-Testprotokoll

Testname	Beschreibung	Eingaben	Erwartetes Verhalten	Tatsächliches Verhalten	Status
Projekt anlegen	Formular zum Anlegen eines Projekts ausfüllen	Name: "Projekt X", Beschreibung: "Test", Owner: Benutzer 1	Projekt wird angelegt, Anzeige der Projektdetails	Erfolgreich	Passed
Task anlegen im Projekt	Task-Formular im Projekt ausfüllen	Titel: "Task 1", Beschreibung: "Details", AssignedUser: 1, CreatedBy: 1	Task erscheint in Liste, korrekte Zuordnung zum Projekt	Fehler: Fehlermeldung beim Speichern	Failed
Nutzer Login	Login mit gültigen Credentials	Benutzername & Passwort korrekt	Login erfolgreich, Weiterleitung zum Dashboard	Erfolgreich	Passed
Nutzer Login (ungültig)	Login mit falschem Passwort	Falsches Passwort	Fehlermeldung "Ungültige Anmeldedaten"	Erfolgreich	Passed

Sicherheitskonzept

Authentifizierung:

Die Anwendung nutzt JWT (JSON Web Token) für die Authentifizierung.

- Nach erfolgreichem Login wird ein JWT erzeugt (JwtUtils) und im Authorization-Header mitgesendet.
- Tokens sind HMAC-signiert und enthalten Ablaufdaten.

Autorisierung:

Ein rollenbasiertes Zugriffskontrollsystem (RBAC) regelt den Zugriff:

- Öffentliche Endpunkte (/api/auth/**, /public) sind ohne Login erreichbar.
- Geschützte Endpunkte (/projects/**, /tasks/**) erfordern Authentifizierung.
- Benutzerrollen (z. B. ROLE_USER) werden aus der Datenbank geladen (UserDetailsServiceImpl).

Sicherheitstechniken:

- BCrypt zur Passwortverschlüsselung
- Stateless Session Management (SessionCreationPolicy.STATELESS)
- CORS-Schutz: Nur Anfragen von http://localhost:5173 erlaubt
- CSRF-Schutz deaktiviert, da keine Sessions verwendet werden

Fehler- und Zugriffsschutz:

- AuthTokenFilter prüft JWT bei jedem Request.
- AuthEntryPointJwt gibt bei fehlender/ungültiger Authentifizierung 401 Unauthorized zurück.
- Logging aller sicherheitsrelevanten Fehler (z. B. Tokenfehler, unautorisierter Zugriff)

Auswertung

Soll-Ist Vergleich:

Funktionalität	Soll	Ist	Kommentar
Benutzer-Login & Authentifizierung	Muss möglich sein	Vollständig implementiert	Backend stabil
Aufgaben erstellen und löschen	Nur berechtigte Nutzer, UI + DB	Backend implementiert, UI unvollständig	Frontend fehlt noch
Aufgabenstatus ändern	Nur Projektmitglieder, UI + DB	Backend umgesetzt, UI noch nicht fertig	
Aufgaben zuweisen	Berechtigte Projektmitglieder	Backend vollständig, UI noch in Arbeit	

Problemanalyse:

Zeitliche Schwierigkeiten:
Aufgrund vieler Fehler im Backend hatte ich keine Zeit mehr, um das Frontend fertig zu stellen. Dadurch konnte ich keine Frontendtests durchführen.

Testbarkeit:
Ohne fertiggestelltes Frontend konnten manche Use Cases nur eingeschränkt getestet werden, was den Gesamtfortschritt bremste. Mit Backend Tests konnten jedoch Fehler eliminiert werden.

Arbeitsjournal

Block 1:

Wir haben einen Input zu dem Modul erhalten und die User Stories erstellt.

Block 2:

Wir haben angefangen an unserem Projekt zu arbeiten und haben weitere Instruktionen erhalten.

Block 3:

Vorlage für die Dokumentation wurde erstellt.

Block 4:

Authentifizierung wurde mit JWT-Token umgesetzt.

Block 5:

Fertigstellung der Models und Repositories im Projekt.

Block 6:

Die Controller wurden umgesetzt.

Block 7:

Ich haben nun an meinem eigenen Projekt gearbeitet und erste Models, DTO's und Controller hinzugefügt.

Block 8:

Ich habe weiter am Projekt gearbeitet und die Login Funktion, sowie die Models, DTO's und Controller angepasst. Das Projekt habe ich zu Hause fertiggestellt.

Block 9:

Im Block 9 präsentieren wir unsere fertigen Projekte und beantworten Fachfragen dazu.