# SEW-5 5AI 2019/20 PLF 16.03.2020

#### **Todo-Liste**

Gewünscht ist (schon wieder) eine Webanwendung zum Verwalten einer Todo-Liste. Auf dem Browser läuft eine Client-App, die über ein REST-API auf eine serverseitige Datenbank zugreift.

#### Server-Vorleistungen

- Aufgaben werden auf dem Server durch die Fachklasse Task repräsentiert. Jede Task hat bereits folgende Eigenschaften:
  - o Long id: Primärschlüssel
  - o long version: Datensatz-Version für "Optimistic Locking"
  - o String kurztext: beschreibt die Aufgabe, mind. 5 bis max. 255 Zeichen
  - o int prio: Priorität im Bereich 1 bis 3, darf nicht null sein
  - o Date faellig: Zeitpunkt, zu dem die Aufgabe erledigt sein soll, darf nicht null sein
  - o boolean erledigt: gibt an, ob die Aufgabe tatsächlich erledigt ist, darf nicht null sein
  - o boolean ueberfaellig: genau dann true, wenn eine Aufgabe noch nicht erledigt ist, der Zeitpunkt faellig aber schon in der Vergangenheit liegt
  - o String beilage: Inhalt einer beliebigen Datei als Data-URL, darf null sein
  - o boolean mitBeilage: genau dann true, wenn es eine beilage gibt
- Die beilage von Tasks wird über das REST-API ausnahmsweise <u>nicht</u> ausgeliefert. Es gibt jedoch eine Projektion TaskNurBeilage: diese liefert auf dem REST-API <u>nur</u> die beilage von Tasks.
- Benutzer/innen werden auf dem Server durch die Fachklasse **User** repräsentiert. Jeder **User** hat bereits folgende Eigenschaften:
  - o Long id: Primärschlüssel
  - o long version: Datensatz-Version für "Optimistic Locking"
  - o String username: eindeutiger Benutzername, darf nicht leer sein
  - o String password: BCrypt-Hash des Passworts, darf nicht null sein
  - o String avatar: Benutzer-Image als Data-URL, darf null sein
- Testdaten für die vollständigen Fachklassen (siehe unten) befinden sich in src/main/resources/data.sql.

#### Server-Anforderungen

- (1 Punkt) Erweitere Task um die Eigenschaft User verantwortlich. Dort soll automatisch diejenige Benutzerin gespeichert werden, die diese Task erzeugt hat.
- (1 Punkt) Kein User darf Tasks mit gleichen kurztexten haben, unterschiedliche User aber dürfen sehr wohl Tasks mit gleichen kurztexten haben. Erweitere die Task-Fachklasse um eine entsprechende Einschränkung.
- (1 Punkt) Beim Löschen eines Users sollen automatisch auch alle Tasks gelöscht werden, für die er/sie verantwortlich ist.
- (2 Punkte) Sorge dafür, dass User sich mit name und password beim Server anmelden können.
- (1 Punkt) Definiere die Projektion **UserKompakt**: diese liefert nur den **username** und den **avatar** einer Benutzerin; auf dem REST-API soll die Projektion **kompakt** heißen.
- (3 Punkte) Erstelle Repositories für User und Task und beachte folgende Anforderungen:
  - o Auf User soll man über das REST-API <u>überhaupt nicht</u> zugreifen können.
  - o User (z.B. wenn sie in Tasks vorkommen), sollen immer als UserKompakt exzerpiert werden.
  - o Tasks soll man im REST-API unter http://localhost:8080/api/aufgaben finden.
- (2 Punkte) Für Tasks sollen auf dem REST-API folgende Einschränkungen gelten:
  - o Man darf nur eigene Tasks bearbeiten können.
  - o Man darf nur <u>eigene</u> Tasks löschen können.

• (1 Punkt) Auf

http://localhost:8080/api/aufgaben/search/findByVerantwortlichUsernameContainsIgnoreCase \( \) OrderByErledigtAscFaelligAsc?suchbegriff=... sollen alle Tasks seitenweise geliefert werden, deren Verantwortliche in ihren Namen den angegebenen suchbegriff enthalten.

### Client-Vorleistungen

• Ein Anwendungsserver auf <a href="http://localhost:8181">http://localhost:8181</a>, der so implementiert ist wie oben beschrieben. Verwende diesen Server für deine Client-Lösung.

In einem Terminal kannst du ihn mit run-server todo-liste starten und mit Strg-C beenden. Bei jedem Neustart wird die Datenbank aus den Testdaten neu erstellt.

- Die Dateien task-seite.html und task-editor.html sind als Ausgangspunkte für deine Komponenten gedacht.
- Die Factories für Task für Task-Objekte und Seite für Seiten, die vom REST-API geliefert werden.
- Die Komponenten <auth> für die An- und Abmeldung, <seiten-nav> für die Navigation zwischen den Seiten, <file-content-chooser> zum Hochladen und <dataurl-downloader> zum Herunterladen eines Data-URLs.
- Die Services RestService für die Kommunikation mit dem REST-API und AuthService für die An- und Abmeldung.

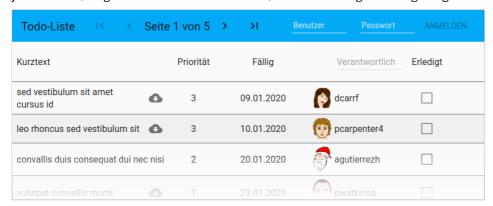
### Client-Anforderungen

Beim Öffnen von index.html soll eine Liste der Aufgaben erscheinen.

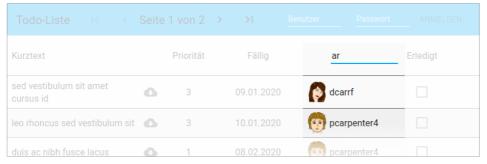
• (2 Punkte) Erstelle zunächst eine Komponente <task>, die ein Task-Objekt so darstellt:



- (1 Punkt) Der \_\_\_\_-Button soll nur erscheinen, wenn es eine Beilage gibt (mitBeilage). Mit diesem Button soll man die Beilage herunterladen können. Verwende dafür die Komponente <dataurl-downloader>.
- (2 Punkte) Erstelle einen Router-State mit einer Komponente <task-seite> mit einer Liste der Aufgaben. Auf jeder Seite (ausgenommen evtl. der letzten) sollen elf Aufgaben angezeigt werden:



• (1 Punkt) Es werden nur diejenigen Aufgaben angezeigt, deren Verantwortliche den Suchbegriff enthalten, den man in der Überschriftszeile eingegeben hat. Die Aktualisierung der Liste erfolgt automatisch bei der Eingabe mit einer Verzögerung vom 300ms. Verwende die Query-Methode des Task-Repositories!

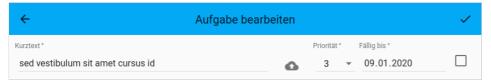


• (1 Punkt) Im Toolbar soll man zwischen den Seiten der Aufgabe-Liste navigieren können.

• (1 Punkt) Im Toolbar soll man sich mit Benutzername und Passwort anmelden können. Nach der Anmeldung erscheint ein Abmelden-Button mit dem Namen des Benutzers. Verwende dafür die Komponente <auth>.



• (2 Punkte) Ist man angemeldet, so erscheint ein / -Button bei allen Aufgaben, für die man selbst verantwortlich ist. Damit gelangt man in eine Ansicht zum Bearbeiten dieser Aufgabe:



Mit dem \_\_\_\_\_Button soll man eine Beilage (z.B. ein Dokument, beliebige Dateitypen sind zulässig) zur Aufgabe hinzufügen können.

- (1 Punkt) Das Speichern mit volume soll nur möglich sein, wenn die Eingaben alle Einschränkungen der Serverseite (siehe oben: "Server-Vorleistungen") erfüllen. bricht den Vorgang ab, ohne die Änderungen zu speichern. In jedem Fall gelangt man danach wieder zur Aufgaben-Liste.
- (1 Punkt) Der + -Button in der Überschriftszeile ist nur sichtbar, wenn man angemeldet ist. Er führt zur Bearbeiten-Ansicht wie oben gezeigt, wobei die Eingabefelder aber leer sind. Nach dem Ausfüllen wird beim Speichern mit / eine neue Aufgabe erzeugt.

### **Bewertung**

Punkte	Note
0 bis <12	5
12 bis 15	4
>15 bis 18	3
>18 bis 21	2
>21	1

## Viel Erfolg!