

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:
 - **Long id**: Primärschlüssel
 - **long version**: Datensatz-Version für „Optimistic Locking“
 - **String kurztext**: beschreibt die Aufgabe, mind. 5 bis max. 255 Zeichen
 - **int prio**: Priorität im Bereich 1 bis 3, darf nicht **null** sein
 - **Date faellig**: Zeitpunkt, zu dem die Aufgabe erledigt sein soll, darf nicht **null** sein
 - **boolean erledigt**: gibt an, ob die Aufgabe tatsächlich erledigt ist, darf nicht **null** sein
 - **boolean ueberfaellig**: genau dann **true**, wenn eine Aufgabe noch nicht **erledigt** ist, der Zeitpunkt **faellig** aber schon in der Vergangenheit liegt
 - **String beilage**: Inhalt einer **beliebigen** Datei als Data-URL, darf **null** sein
 - **boolean mitBeilage**: genau dann **true**, wenn es eine **beilage** gibt
- Die **beilage** von **Tasks** wird über das REST-API ausnahmsweise **nicht** ausgeliefert. Es gibt jedoch eine Projektion **TaskNurBeilage**: diese liefert auf dem REST-API **nur** die **beilage** von **Tasks**.
- Benutzer/innen werden auf dem Server durch die Fachklasse **User** repräsentiert. Jeder **User** hat bereits folgende Eigenschaften:
 - **Long id**: Primärschlüssel
 - **long version**: Datensatz-Version für „Optimistic Locking“
 - **String username**: eindeutiger Benutzername, darf nicht leer sein
 - **String password**: BCrypt-Hash des Passworts, darf nicht **null** sein
 - **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:
 - Auf **User** soll man über das REST-API **überhaupt nicht** zugreifen können.
 - **User** (z.B. wenn sie in **Tasks** vorkommen), sollen immer als **UserKompakt** exzerpiert werden.
 - **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:
 - Man darf nur **eigene** **Tasks** bearbeiten können.
 - Man darf nur **eigene** **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 `http://localhost:8181`, 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:

faucibus accumsan odio curabitur convallis		2	30.06.2020		user	<input checked="" type="checkbox"/>	
--	---	---	------------	---	------	-------------------------------------	---

- (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:

Todo-Liste ◀ < Seite 1 von 5 > ▶ Benutzer Passwort ANMELDEN							
Kurztext		Priorität	Fällig		Verantwortlich	Erledigt	
sed vestibulum sit amet cursus id		3	09.01.2020		dcarrf	<input type="checkbox"/>	
leo rhoncus sed vestibulum sit		3	10.01.2020		pcarpenter4	<input type="checkbox"/>	
convallis dui consequat dui nec nisi		2	20.01.2020		agutierrezh	<input type="checkbox"/>	
volutpat convallis morbi		1	23.01.2020		pwatkinsa	<input type="checkbox"/>	

- (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!

Todo-Liste ◀ < Seite 1 von 2 > ▶ Benutzer Passwort ANMELDEN							
Kurztext		Priorität	Fällig		Verantwortlich	Erledigt	
sed vestibulum sit amet cursus id		3	09.01.2020		dcarrf	<input type="checkbox"/>	
leo rhoncus sed vestibulum sit		3	10.01.2020		pcarpenter4	<input type="checkbox"/>	
duis ac nibh fusce lacus		1	08.02.2020		pcarpenter4	<input type="checkbox"/>	

- (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 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 Überschriftenzeile 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!