ToDo-Website

Dokumentation

von

Johanna Wolfstieg

Juli 2025

Matrikelnummer, Kurs 2937506, TINF24B5

Inhaltsverzeichnis

1 User Authentifizierung

1.1 Überblick

1.2 Einbindung in das Gesamtprojekt

Die User Authentifizierung ist notwendig für die personalisierte Nutzung der App und der Speicherung der Daten. Ohne die Authentifizierung wäre bspw. keine personalisierte Speicherung der Todos möglich und damit die Nutzung der App auch unsinnig. Zur Authentifizierung wurden in diesem Projekt folgende Funktionen implementiert:

- Login
- Registrierung
- Passwort vergessen
- Nutzerdaten ändern

Die Userdaten sind, während der User eingeloggt ist, im Frontend durch das Behaviour-Subject und Observable verfügbar. Damit wird der aktuelle Benutzerzustand verwaltet und kann von anderen Komponenten verwendet werden. Durch folgende Strukur im User-Service ist das möglich:

```
private userSubject: BehaviorSubject<User | null>;
public user: Observable<User | null>;
```

UserSubject ist ein privates BehaviorSubject, das entweder den aktuell eingeloggten Benutzer (User) oder null speichert, wenn kein Benutzer angemeldet ist. Der große Vorteil von BehaviorSubject besteht darin, dass es immer den letzten Zustand enthält und diesen automatisch an alle Abonnenten weitergibt, in diesem Fall dem Userservice.

Das ermöglicht es, den Benutzerstatus auch nach einem Seitenwechsel sofort verfügbar zu haben.

User ist ein öffentliches Observable, das auf userSubject basiert. Es dient als schreibgeschützte Schnittstelle für Komponenten und andere Services, die nur über Änderungen am Benutzerstatus informiert werden sollen, aber selbst keine Änderungen daran vornehmen dürfen. Dies stellt sicher, dass der Benutzerzustand kontrolliert verwaltet wird. Alle Komponenten, die userService.user abonniert haben, werden automatisch über diese Änderungen informiert, wie z.B. in app.component.ts:

```
constructor(private router: Router, private route: ActivatedRoute) {
  this.userService.user.subscribe(x => this.user = x);
}
```

Damit kann dann bspw. in der dazugehörigen html Datei logout oder login angezeigt werden, jenachdem ob ein Nutzer vorhanden ist oder nicht. Außerdem wird durch die Hilfsfunktion `LoggedActivate` sichergestellt, dass nur ein eingeloggter Nutzer auf bestimmte Routen zugreifen kann. Somit wird verhindert, dass ein nicht authentifierter Nutzer auf Todos oder Bereiche zugreift.

1.3 Zugrunde liegende Idee und Funktionsweise

1.3.1 Frontend

Alle Komponenten sind folgendermaßen aufgebaut:

- HTML: Formular mit Bootstrap

Im Formular können die Werte der verschiedenen Inputs mit FormControl von Angular behandelt werden. Durch Validators wird die Eingabe auf Kriterien geprüft, welche wichtig für eine korrekte Anmeldung sind. Dafür werden sowohl vorhandene Angular Validators genutzt, als auch eigene Validators, welche für E-Mail und Passwort Validation geschrieben wurden.

- .ts Datei mit handleSubmit()

Die zugehörige TypeScript-Datei enthält die Logik. Wird das Formular abgeschickt (`handleSubmit()`), prüft die Methode die eingegebenen Werte und sendet diese je nach Funktion über den `UserService` an das Backend. Bei Erfolg wird eine Bestätigung angezeigt und im Fehlerfall erscheint eine entsprechende Fehlermeldung.

Außerdem enthält die `User-Edit-Component` eine `delete()`-Methode, mit der der aktuell eingeloggte Benutzer gelöscht werden kann. Nach erfolgreichem Löschen wird der User zur Startseite navigiert.

- CSS

Die zugehörige `.css`-Datei sorgt für ein responsives Layout. Das `.form`-Element erhält einen gleichmäßigen Abstand nach innen (`padding`) und außen (`margin`), wobei das Padding nach links und rechts größer ist um das Formular mittig anzuordnen.

Die Klasse `.btn` definiert einen Abstand um Schaltflächen, um die Lesbarkeit und Übersichtlichkeit zu verbessern.

Mit `@media` wird das Layout auch für kleinere Bildschirme (maximale Breite 768 px und Höhe 440 px) angepasst.

1.3.2 Backend

Das Backend besteht aus dem `User-Router`, dem `User-Model`, welches mit Zod auch im Backend eine Validierung hat, und dem `User-Schema` für die Datenbank. Im `User-Router` werden die Express-Routen für die User-Aktionen definiert. In jeder Route befindet sich zudem ein Datenbank Abruf mit Drizzle ORM. Die Routen parsen die eingehenden Daten mit dem jeweiligen `Userschema`, wodurch nochmals eine sichere Speicherung der Daten gewährleistet wird. Die `UserId` wird im Backend als UUID erzeugt, damit diese eindeutig ist. Ist das Parsen erfolgreich, kann der Datenabruf via Drizzle erfolgen.

1.4 Entscheidungen

Beim Passwort-Reset wird das Passwort einfach geleert – dies sollte später durch einen Token-basierten Prozess ersetzt werden. Da die Passwort Zurücksetzung jedoch eher

ein Beispiel ist, es wird bislang nämlich auch keine Mail mit einem Link zur Eingabe des neuen Passworts versendet, ist das Passwort Zurücksetzen als Demo zu sehen und nicht als fertige Komponente für die Nutzung einer Website. In Bezug auf die Speicherung des Passworts ist die Sicherheit nicht gewährleistet, auch hier bräuchte es eine andere Vorgehensweise. Für den Rahmen dieses Projekts wurde sich jedoch aus Zeitgründen dagegen entschieden, klar ist jedoch dass die Website so nicht den Security-Standards entspricht.

2 Filter-/ Sortieroptionen

Für einen guten Überblick der Todos sind Filter-/ und Sortieroptionen wichtig. Folgendes wurde implementiert: Sortierung nach:

- Deadline
- Priorität
- Schwierigkeit
- Alphabetische

Filtern nach:

- Erledigt
- Unerledigt
- Zeitraum

Diese Optionen wurden mit einer wiederverwendbaren Dropdown Komponente implementiert. Es werden verschiedene Optionen sowie der aktuelle Zustand per Input entgegengenommen und per Output werden Nutzerinteraktionen zurückgegeben. So kann beispielsweise die Sortierung aufsteigend sowie absteigend gewählt werden. Für die Auswahl des Zeitraums ist ein Modal implementiert, das nur erscheint, wenn der User nach Zeitraum filtern möchte.

In der `Todos.Component` und dem `UserService` ist die Logik für das Sortieren und Filtern implementiert. Der Array von Todos, welcher aus dem Backend zurückgegeben wird, wird dafür verwendet.