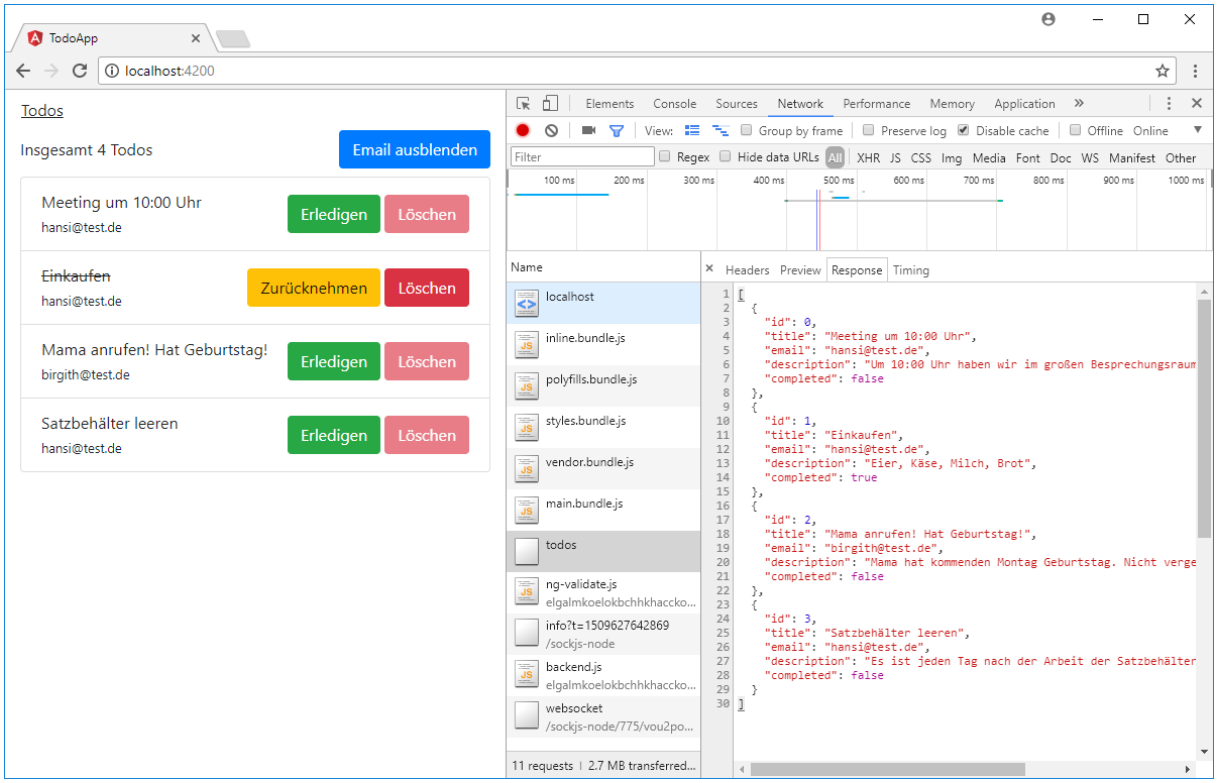
# 1. HttpClient [U-070]

> *Als Benutzer möchte ich die Liste der Todos auf einem Server hinterlegen und verwalten, um Änderungen an der Liste dauerhaft zu speichern und online darauf zugreifen zu können.*

## 1. Lernziele

- ✓ Sie nutzen das HttpClient-Modul um Anfragen an ein Back-End mit Datenbank zu implementieren.

## 2. Ergebnis




## 3. Benötigte Dateien

- `src/app/todo.service.ts`
- `src/app/app.module.ts`
- `src/app/app.component.html`
- `src/app/todo-list.component.ts`

- `src/app/todo-detail.component.ts`

## 4. Anleitung

### VORBEREITUNGEN FÜR UMSTELLUNG

---

Schritt 1: Installieren und Starten des JSON-Servers

1. Wenn noch nicht geschehen, installieren Sie den json-server mittels `npm` global auf Ihrem Rechner: `npm install -g json-server`
2. Starten Sie den JSON-Server, indem sie den Befehl `npm run db` ausführen. Dadurch wird unter der URL `http://localhost:3000/todos` eine REST-API bereitgestellt, welche die zuvor verwendeten `mockTodos` zurückliefert.

---

Schritt 2: `HttpClient` in Service-Klasse injizieren und die Konstante `BASE_URL` mit dem Wert `http://localhost:3000/todos` definieren

3. Nehmen Sie `HttpClientModule` aus `@angular/common/http` unter `imports` in `AppModule` auf
4. Injizieren Sie eine Variable `http` vom Typ `HttpClient` im Konstruktor der TodoService-Klasse
5. Erstellen Sie außerhalb dieser Klasse eine neue Konstante `BASE_URL` mit dem Wert `http://localhost:3000/todos`.

Schritt 3: Implementieren einer Methode `getAll()` im Service, die die Methode `get` des `HttpClient` verwendet, um die Liste von Todos zu laden. Der Rückgabetyp dieser Funktion wird ein `Promise<TodoItem[]>` sein.

6. Ersetzen Sie die Rückgabe von `todos` durch einen Aufruf der Methode `get` von `http`. Geben Sie dabei `TodoItem[]` als Rückgabedatentyp von `get` an.
7. Übergeben Sie `BASE_URL` als Parameter an `get`.
8. Wandeln Sie die Rückgabe mit der Methode `firstValueFrom()` in ein `Promise`-Objekt um und geben Sie dieses als Rückgabewert der `getAll`-Methode zurück. Vervollständigen Sie den Rückgabetyp der Funktionssignatur für `getAll()`.

Schritt 4: Verwendung von `getAll()` in der `TodoListComponent` und Anpassen der Funktionalität an den neuen Rückgabetypen `Promise<TodoItem[]>`

9. Die Methode `getAll` des TodoService liefert nun anstatt eines Arrays einen Promise zurück. D.h. die Zuweisung muss nun asynchron erfolgen. Weisen Sie daher den Rückgabewert von `getAll()` nicht einer Variablen zu, sondern hängen nach dem Aufruf von `getAll()` einen Aufruf zu `.then()` an.
10. Übergeben Sie dem `then()`-Operator des Promise eine Funktion, welche die Liste der Todos übergeben bekommt und sie der `todoList` zuweist:

```
promise.then((erhaltenerWert) => doSomething());
```

Starten Sie die Anwendung und prüfen Sie ob die Lsite korrekt angezeigt wird.

Schritt 5: Aufgrund des asynchronen Aufrufs, um Daten zu erhalten, bei Deklaration von `todoList` ein leeres Array zuweisen

11. Da diese Art der Initialisierung asynchron geschieht, müssen Sie `todoList` bei Deklaration ein leeres Array zuweisen, um Null-Pointer-Exceptions zu vermeiden.

Schritt 6: `Todo` in der Methode `update` der Klasse `TodoService` an den Server schicken

12. Verwenden Sie die Methode `put` von `http` in der Methode `update` der `TodoService`-Klasse.
13. Konstruieren Sie aus `BASE_URL` und dem Attribut `id` des Parameters `todo` einen neuen String welchen Sie der `put`-Methode als ersten Parameter (`url`) übergeben.
    Beispiel: Das Todo mit der Id 1 → `http://localhost:3000/todos/1`
14. Geben die als zweiten Parameter das `todo` an. Hierbei handelt es sich um den Body des Requests.
15. Wandeln Sie die Rückgabe mit der Methode `firstValueFrom()` in ein Promise-Objekt um und geben Sie dieses als Rückgabewert der Methode zurück.
16. Passen Sie den Methodenaufruf in der `TodoListComponent` an.

Schritt 7: Umbau der Methode `delete` des `TodoService`, um auf dem Server einen Eintrag zu löschen.

17. Verwenden Sie in der `delete` Methode der `TodoService`-Klasse die `delete`-Methode des `HttpClient` um die Anfrage an die Datenbank/den Server zu schicken.
18. Konstruieren Sie aus `BASE_URL` und dem Attribut `id` des Parameters `todo` einen neuen String für die URL, welchen Sie der `delete`-Methode als Parameter übergeben.

19. Wandeln Sie die Rückgabe mit der Methode `firstValueFrom()` in ein Promise-Objekt um und geben Sie dieses als Rückgabewert der Methode zurück.

Beachten Sie, dass aktuell noch keine Funktionalität implementiert wurde, um Todos auf dem Server anzulegen!

---

Schritt 8: Logik für `ngOnInit` der `TodoListComponent` in eine Methode `fetchTodos` auslagern

20. Erstellen Sie in der Klasse `TodoListComponent` eine neue, öffentliche Methode `fetchTodos()` welche keine Parameter annimmt und keine Rückgabewerte hat.
21. Verschieben Sie die Logik von `ngOnInit` nach `fetchTodos`.
22. Rufen Sie `fetchTodos` in `ngOnInit` auf.

---

Schritt 9: Aktualisieren der Liste von Todos nach dem Löschen eines Todos

23. In der `onDeleteClick()`-Methode der `TodoListComponent` fügen Sie nach dem Aufruf des `delete`-Operators ein `.then()` an.
24. Übergeben Sie dieser `then()`-Methode eine Funktion, welche keine Parameter annimmt und die neue Funktion `fetchTodos()` aufruft.

## 5. Kontrollfragen

- Sie kennen die Methoden des HttpClients, welche die CRUD-Operationen realisieren?
- Sie kennen den Aufbau einer http-Service-Methode?
- Wie wird der erwartete Rückgabetyp der Http-Anfragen angegeben?
- Wie erfolgt der Import der `firstValueFrom()`-Methode

## 6. Weiterführende Materialien

- Angular-Dokumentation über die HttpClient-Klasse `https://tinyurl.com/gs-angular-httpclient-class`