Studienleistung 3

Alexander Bazo und Jakob Fehle

ToDo-App mit Server-Datenbank

In dieser Aufgaben entwickeln Sie die Client-Server-Interaktion eines Web-basierten Notizblocks. In diesem Notizblock können Einträge erstellt, aktualisiert und als Erledigt markiert werden. Ihre Anwendung kommuniziert im Hintergrund mit einer Datenbank und ermöglicht so das Auslesen und persistente Verändern und Abspeichern von Daten.

Durch die Betätigung des "+"-Elements im UI der Webseite kann der Liste des Notizblockes ein neues Element hinzugefügt werden. Das Erstellen eines Elements soll an die Datenbank des Webservers weitergeleitet und dort abgespeichert werden. Ebenso sollen auch Änderungen am Element (Text oder auch das Markieren als Erledigt) mit der Datenbank kommuniziert werden. Falls der Datenbank bereits Einträge hinzugefügt wurden, sollen diese beim erneuten Laden des Notizblockes von der Datenbank bezogen und angezeigt werden, um eine aktuelle Ansicht aller Einträge zu repräsentieren.

Vorgaben

Im Starterpaket finden Sie ein angefangenes Projekt, in welchem ein vollständiges HTML-Dokument und eine bereits verknüpfte CSS-Datei vorhanden sind, die vom Webserver an den Client ausgeliefert werden. Ebenfalls ausgeliefert werden mehrere JavaScript-Dateien, die zusammen mit der HTML-Datei aufgerufen werden. Die Dateien `Task.js`, `TaskList.js` und `TaskView.js` im Ordner `app/resources/js/task` übernehmen die Client-seitige Darstellung und Interaktion des Notizblocks. Die Klassen `Task` und `TaskList` verwalten hierzu die interne Objekt-Struktur des Notizblocks und die Klasse `TaskView` realisiert das User Interface der Anwendung, also Darstellung und Interaktion der Listen-Elemente. Zudem wird Ihnen über die Datei `app/resources/js/http/FetchHelper.js` die Klasse FetchHelper bereitgestellt, mit welcher Sie über die vorgefertigte Funktion `makeRequest` Anfragen an Ihren Webserver stellen können. An keiner dieser Dateien müssen Sie eigenständig Veränderungen vornehmen. Sie können Ihre Arbeiten in der Datei `app/resources/js/app.js` beginnen, welche die zentrale JavaScript-Datei Ihrer Client-Webseite darstellt.

Für den Webserver existiert bereits eine SQL3-Datenbank (`data/todo.db`) sowie die dazugehörige JavaScript-Datei `Database.js` im Stammverzeichnis `lib`, welche die grundsätzliche Anbindung an die Datenbank übernimmt. In der JavaScript-Datei `index.js` wird die grundlegende Funktionalität (Bereitstellung von HTTP-Routen und der statischen Webseite) des Webservers realisiert. Sowohl die Datei `Database.js` als auch `index.js` sollen um Funktionialität ergänzt werden.

Alle benötigten Dateien sind kommentiert und zu erledigende Aufgaben sind im Quellcode markiert.

Starten der Anwendung

Um die Anwendung korrekt auszuführen, wird ein lokaler Webserver benötigt. Die LiveShare-Funktion von *Visual Studio Code* reicht hierfür nicht mehr aus. Statt dessen starten wir den Server direkt über die *Node.js*-Umgebung. Installieren Sie *Node.js* über die entsprechende Installationsdatei für Ihr Betriebssystem, die Sie hier herunterladen können. Öffnen Sie dann den Projektordner in *Visual Studio Code* und Starten Sie das integrierte Terminal. Führen Sie dort den Befehl **npm install** aus, um das Projekt vorzubereiten. Dannach können Sie über die Eingabe des Befehls **npm start** den Server starten und den Client im Browser über die Adresse **http://localhost:8080/** aufrufen. Wenn Sie im integrierten Terminal die Tastenkombination **STRG** + **C** drücken, wird der Server beendet.

Verpflichtende Anforderungen

Folgende Anforderungen müssen Sie für alle drei Interaktionsmöglichkeiten zwischen dem Client-User Interface und der Webserver-Datenbank umsetzen (Laden aller gespeicherten Daten, Erstellen eines neuen Eintrags, Aktualisieren eines bestehenden Eintrags):

- Implementieren Sie die bereits bestehenden Funktionen `getAllTasks`, `addTask` und `updateTask` in der Datei `Database.js` um die geforderten Funktionialitäten (GET, PUT und UPDATE) für die Datenbank in Form von SQL-Queries zu realisieren.
- Verarbeiten Sie im Webserver eingehende HTTP-Anfragen in den Funktionen `onTasksRequested`, `onTaskAdded` und `onTaskUpdated` der Datei `index.js`, indem Sie die die jeweilige Anfrage verarbeiten, an die Datenbank weiterleiten und abschließend eine Antwort zurücksenden.
- Entwickeln Sie passende HTTP-Anfragen, um die Funktionalitäten des Notizblocks an den Webserver weiterzuleiten. Erweitern Sie hierzu die Funktionen `loadDatabaseFromServer` (Lade alle Notizbuch-Einträge von der Datenbank und stelle diese im UI dar), `onNewTaskButtonClicked` (Erstelle neuen Eintrag und speichere diesen in der Datenbank) und `onTaskChangedByUser` (Aktualisiere veränderten Eintrag in der Datenbank) der Datei `app.js` um verschiedene Fetch-Anfragen, deren Funktionalität über die Funktion `makeRequest` der Klasse `FetchHelper` bereitgestellt wird.
- Verarbeiten Sie die Antwort des Webservers auf Ihre HTTP-Anfragen wo nötig (z.B. Erstellen einer Darstellung der bereits existierenden Einträge des Notizblocks).

Optionale Anforderungen

- Implementieren Sie die Möglichkeit Einträge aus dem Notizblock zu löschen. Sie können sich hierfür an bereits bestehender Programmstruktur anderer Funktionalitäten orientieren.

Achten Sie auf bisher gelernte Prinzipien zur Code-Qualität und kommentieren Sie Ihre Lösung ausreichend.

Abgabekriterien:

Laden Sie Ihre Lösung bis spätestens 23.07.2021 (23:59 Uhr) als zip-komprimierten Ordner auf GRIPS hoch. Benennen Sie die einzelnen Dateien pro Aufgabe sinnvoll und verwenden Sie geeignete Formate:

- Aufgabe 1: Das gesamte Projekt (HTML, JS, CSS)

Der Name der Datei ergibt sich aus dem Präfix „SL_WT_SS21“, der Nr. der Studienleistung, ihrem Vor- und Nachnamen jeweils getrennt durch _ .

Beispiel: **SL_WT_SS21_3_Max_Mustermann.zip**