Prof. Dr. Sebastian von Klinski, klinski@beuth-hochschule.de Raum B230a

ÜBUNG 1 – MEILENSTEIN 1			
LV	**WEB-ENGINEERING II**		
THEMA	**REST-SERVER**	FÄLLIGKEIT	**SIEHE MOODLE**

ÜBUNG 1, MEILENSTEIN 1 - VORGABEN

Grundlage für die Umsetzung und Benotung dieses Meilensteins sind die „Allgemeinen Regelungen für Übungsaufgaben“ sowie die Aufgabenstellung für Übung 1, die Sie beide im Moodle finden.

In diesem Dokument werden sowohl die funktionalen als auch allgemeinen Anforderungen an den Meilenstein 1 von Übung 1 weiter detailliert. Bitte beachten Sie, dass alle nachfolgenden Vorgaben exakt eingehalten werden müssen, weil viele Aspekte über automatisierte Tests geprüft werden.

Für Abweichungen von den funktionalen Anforderungen gibt es bis zu 15 Abzugspunkte. Darüber hinaus werden Abzugspunkte berechnet, wenn die allgemeinen Anforderungen nicht eingehalten werden. Die Abzugspunkte orientieren sich an dem Aufwand der notwendig wäre, um das Problem zu beheben.

Falls manuelle Eingriffe notwendig sind, um die abgegebene Lösung zu testen, werden für den ersten Eingriff mindestens 3 Abzugspunkte berechnet. Für alle weiteren Eingriffe mindestens 2 Abzugspunkte. Ist der Korrekturaufwand sehr groß, können auch mehr Abzugspunkte berechnet werden.

Sollte es grundlegende Probleme mit der abgegebenen Lösung geben, müsste auch für diesen Meilenstein eine persönliche Abnahme durchgeführt werden.

FUNKTIONALE ANFORDERUNGEN

Setzen Sie den Public-User-Endpoint („/publicUsers“ siehe Aufgabenstellung für Übung 1) zum Verwalten der User-Daten um. Dieser Endpoint soll noch keine Authentifizierung oder Token-Handling umsetzen.

Die Anforderungen an diesen Meilenstein sind einfach gehalten, um den Einstieg in die Implementierung zu erleichtern. Der Endpoint soll alle CRUD-Methoden (Create, Read, Update, Delete) für User umsetzen und die Suche nach einem User per User-ID. Die Umsetzung soll den Vorgaben der Vorlesung „Best-Practices bei Umsetzung von REST-Services“ entsprechen. Im Moodle gibt es für diesen Meilenstein eine Liste von REST-Client-Tests. Diese sollten bei er Aufgabe auf jeden Fall funktionieren.

Die URL des Endpoints ist „/publicUsers“.

Das User-Objekt sollte entsprechend der Aufgabenstellung die folgenden Attribute umfassen:

- userID
- password
- firstName
- lastName
- isAdministrator

Wenn Sie planen, Zusatzleistungen umzusetzen, können Sie auch weitere Attribute (beispielsweise für die E-Mail oder ein Bild des Users) hinzufügen. Die oben angegebenen Attribute müssen jedoch in jedem Fall enthalten sein. Bitte achten Sie auf die exakte Schreibweise aller Attribute! Auch Groß- und Kleinschreibung ist relevant.

Bei den Routen dieses Endpoints, bei denen User-Daten zurückgeben werden, sollen stets alle Attribute zurückgegeben werden. Insbesondere ist auch der Hashwert des Passworts zurückzugeben. Das ist zur automatischen Überprüfung des Hashings erforderlich.

Hinweis 1: Bei einem korrekten REST-Service würde nie das Passwort von Usern (auch nicht der Hashwert) zurückgegeben werden. Das ist nur für die automatische Prüfung der von Ihnen abgegebenen Lösung im Meilenstein 1 notwendig. Diese Route soll jedoch auch bei allen weiteren Meilensteinen erhalten bleiben.

Hinweis 2: User sollten über die User-ID (z.B. „admin“) und nicht über den Unique-Identifier gesucht werden!

Hinweis 3: Beim Starten des Servers soll noch nicht automatisch ein Standard-Administrator angelegt werden. Das soll erst ab dem Meilenstein 2 umgesetzt werden.

ALLGEMEINE ANFORDERUNGEN

Bitte halten Sie neben den Vorgaben der Aufgabenstellung sowie den Allgemeinen Regelungen auch die folgenden Anforderungen exakt ein.

Endpoint

Verwenden Sie für die Umsetzung des REST-Severs http, noch kein HTTPS. Verwenden Sie den Standard-http-Port 80.

Starten des Servers

Die Anwendung muss bei mir auf dem Rechner laufen. Dementsprechend sollten Konfigurationseinstellungen und Umgebungsvariablen so ausgelegt werden, dass der REST-Server auf einem anderen Rechner ohne Anpassung läuft. Testen Sie das bitte vor der Abgabe.

MongoDB muss unter dem Standard-Port angebunden werden.

Abhängigkeiten prüfen

Über den Befehl „npm install“ werden alle Abhängigkeiten zu Drittanbietermodulen aufgelöst und die Module im Verzeichnis „node_modules“ abgelegt. Achten Sie darauf, dass alle erforderlichen Abhängigkeiten in der package.json eingetragen sind. Wenn Sie Module als global in Ihrer Entwicklungsumgebung eingetragen hatten, kann es sein, dass die Abhängigkeit nicht drinsteht! Testen Sie das vor Ihrer Abgabe. Falls nicht alle Abhängigkeiten korrekt aufgelöst werden, werden Abzugspunkte berechnet.

Start-Skript anlegen

Durch den Befehl „npm start“ kann die Anwendung gestartet werden. Dazu muss aber das Start-Skript in der package.json definiert werden. Definieren Sie in der package.json-Datei das Start-Skript. Der Server muss mit dem Befehl „npm start“ gestartet werden können. Das kann beispielsweise wie folgt gemacht werden:

```
"scripts": {
    "start": "node ./HttpServer.js"
  },
```

Wenn das Startskript nicht definiert wurde, werden dafür Abzugspunkte berechnet.

Bcrypt-Version

Für das Hashing der Passwörter sollen Sie Bcrypt verwenden. Es gibt unterschiedliche NPM-Module für Brypt. Die klassische Bycrypt-Version ist eine Implementierung in C, die plattformabhängig ist. Beim Wechsel zwischen Betriebssystemen gibt es immer wieder Probleme, die dann bei den Abnahmen erst aufgelöst werden müssen.

Es gibt eine Implementierung, die rein in JavaScript umgesetzt ist. Dieses Modul heißt „bcryptjs“. Bitte verwenden Sie dieses Modul, damit es beim automatisierten Test Ihrer Anwendung keine Probleme gibt. Die Dependency in der package.json sollte dann wie folgt aussehen. Verwenden Sie nicht das richtige Modul, werden Abzugspunkte berechnet.

```
"bcryptjs": "^2.4.3",
```

TESTS FÜR PRÜFUNG, OB IHR SERVER KORREKT FUNKTIONIERT

Im Moodle-Kurs finden Sie eine Datei mit REST-Client Tests, mit denen Sie in Visual Studio ihren REST-Server testen können. Nutzen Sie in jedem Fall diese Tests, um sicherzustellen, dass Sie den Endpoint korrekt umgesetzt haben.

Bei der automatisierten Prüfung Ihrer abgegebenen Lösung werden darüber hinaus noch weitere Prüfungen, insbesondere auf die Korrektheit der Antworten vorgenommen. Es wird auch das Fehler-Handling überprüft. Hierzu gehören beispielsweise die folgenden Tests:

- Was passiert, wenn ein zweiter User mit der gleichen User-ID angelegt wird? Hier sollte es eine entsprechende Fehlermeldung geben.
- Was passiert, wenn ein User gelöscht werden soll, den es nicht gibt?
- Was passiert, wenn ein User geändert werden soll, den es nicht gibt?
- Was passiert, wenn ein User angelegt werden soll, der keine User-ID hat? Das sollte nicht möglich sein.

HINWEISE ZU DEN REST-SERVER-ANTWORTEN

Beachten Sie auch die Vorgaben zu den Antworten aus der Aufgabenstellung für Übung 1. Insbesondere sollten Sie folgende Vorgaben für die Antworten einhalten:

- Wenn alle User abgerufen werden und noch kein User existiert, sollte ein leerer Array zurückgegeben werden.
- Der REST-Server sollte „Einfache Antworten“ zurückgeben (siehe Vorlesung „Best-Practices bei der Umsetzung von REST-Services“). Es sollten keine „Wrapped“-Responses oder einfache Strings zurückgegeben werden.

Viel Erfolg!