0.1 Dokumentengeschichte

Zeitraum	PL/Autor(en)	Änderungen
Sommersemester 1980	IHR NAME	text
Wintersemester 2017/18	Jan Schuster	Erstellung eines Konzeptes zur Nutzerverwaltung

Tabelle 1: Dokumentengeschichte

0.2 Aufgabe der Komponente

Diese Komponente soll eine Grundlage für die spätere Entwicklung eines Webservers liefern. Darin enthalten, soll sein, ein Prototypisches Verhalten für die Benutzerverwaltung und die Kommunikation über Rest Schnittstellen. Ein Nutzerprogramm soll dabei die Möglichkeit erhalten, sich auf den Server zu authentifizieren und entsprechend seiner Rechte, Daten lesen oder schreiben können. Die Kommunikation soll über REST stattfinden. Außerdem soll sie über Apache Tomcat Lauffähig sein.

Das Grundlegende Ziel ist es, diese Authentifizierung mit so wenig externen Komponenten wie möglich zu bauen.

0.3 Architektur

0.3.1 Überlick

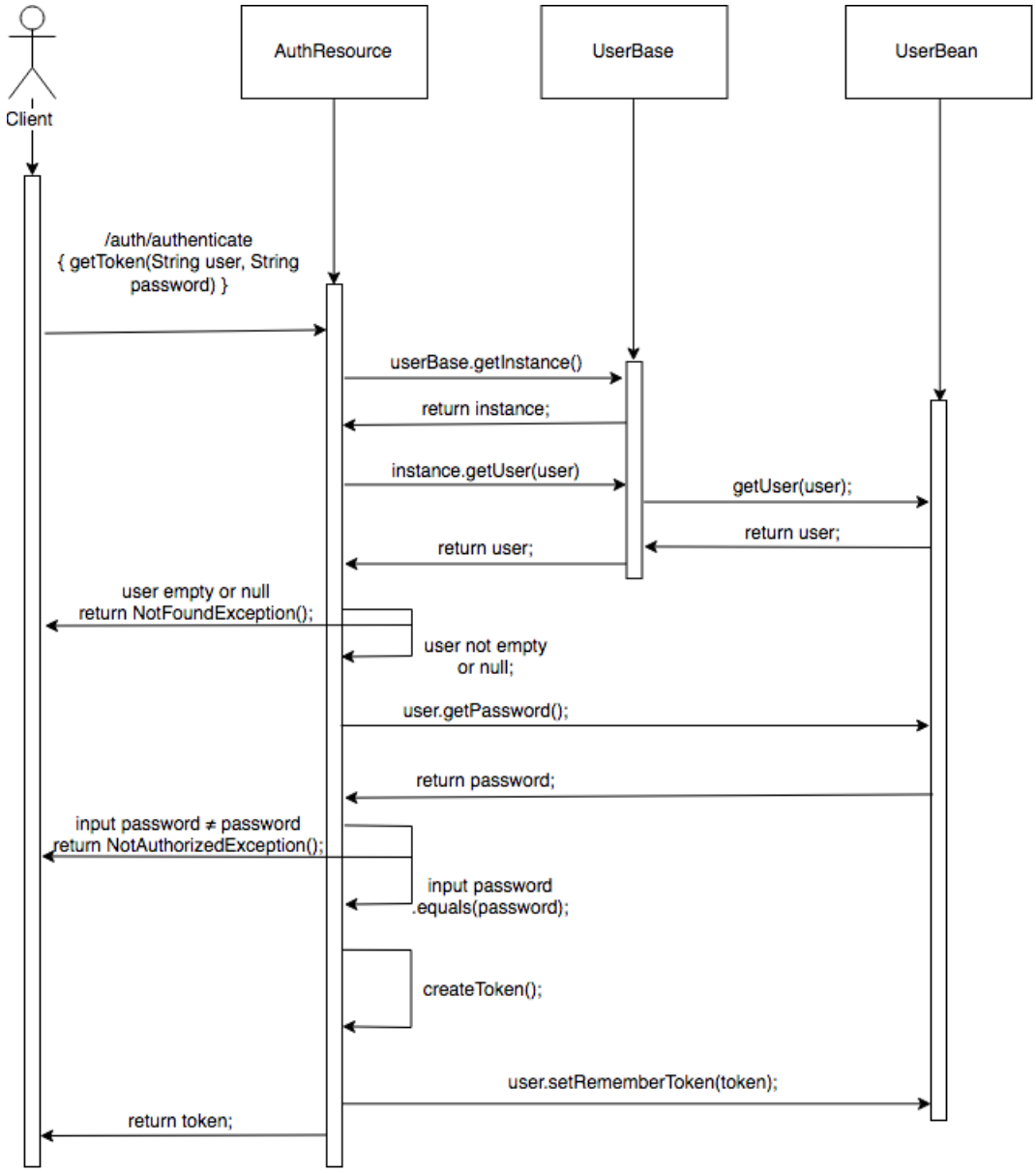
Die grundlegenden Nutzerdaten werden in einer UserBean vorgehalten. Das Singleton UserBase kümmert sich um die Verwaltung der Nutzer. Beim Initialisieren werden alle Nutzer in einer Collection gepackt und es wird ein Timer in die jeweilige Bean aus der UserBase mit rein gegeben. Wird ein Remembertoken in der UserBean gesetzt, wird ein Thread mit einen Timertask erzeugt. Beim erneuten setzen des Timers, wird der Thread auf den Ursprung zurück gesetzt. Jegliche Anfragen zu einen Benutzer findet ausschließlich über die Instanz des Singletons UserBase statt. Das bearbeiten oder verändern von Benutzern verantworten die Schnittstellen selbst.

Es gibt zwei Klassen die REST Schnittstellen zur Verfügung stellen. Die erste ist die Klasse AuthResource und die zweite die UserResource. Folgende Use-Cases sind durch die beiden Klassen abgedeckt:

AuthResource:
- Methode zum Testen der regulären Rest Funktion
- Methode zum Authentifizieren von Nutzerdaten -¿eventuelle Rückgabe eines Tokens zur weiteren Verwendung

UserResource:
- Methode zur Ausgabe von Nutzerdaten anhand eines herein gegebenen Nutzers
- Methode zum Setzen eines Passwortes für einen Nutzer anhand eines Tokens und eines Passwortes
- Methode zum anzeigen der Rechte eines Tokens anhand des herein gegebenen Tokens

Beispiel eines Authorisierungsprozesses:



0.3.2 Schnittstellendefinitionen

Authentifizierung: AuthResource

Die erste ist eine Default Test Methode zum testen der allgemeinen Funktionsfähigkeit und gibt einen String mit "Default REST Methodßurück.
Methodenname: restTest()
Schnittstelle: /auth/test
Medien Typ: Text_Html, Text_Plain

Die zweite Methode ist für die Authentifizierung gedacht. Die Schnittstelle selbst bekommt einen Nutzer und ein Passwort als Eingabe herein. Als erstes wird überprüft ob auch ein Nutzer in die Methode gegeben wurde. Ist dies der Fall, wird sich eine Instanz der UserBase geholt. Danach wird überprüft ob ein Password herein gegeben wurde und ob dieses mit dem zur Verfügung gestellten Benutzer übereinstimmt.

Ist dies der Fall, wird ein String aus dem Nutzer, dem Password und einer Random UUID erstellt. Aus diesem String wird mit Hilfe des Base 64 Encoder ein ein Token erzeugt. Daraufhin wird eine Methode im Nutzer selbst getriggert, die den Token als RememberToken speichert und ein Timer aktiviert, der die gültige Zeitdauer des Tokens bestimmt. Zum Schluss wird der Token selbst als String zurückgegeben.
Methodenname: getToken(String user, String password)
Schnittstelle: /auth/authenticate
Medien Typ: Text_Html, Text_Plain

Nutzer: UserResource

Die erste Methode ist als Rückgabe der angefragten Nutzerdaten als JSON gedacht. Der Methode wird nur ein Nutzer übergeben. Daraufhin wird überprüft, ob der Nutzer im System vorhanden ist. Ist dies der Fall, werden die Daten mithilfe der MediaType in eine JSON umgewandelt.
Methodenname: getUser(String user)
Schnittstelle: /user/getUser
Medien Typ: Application_JSON

Die Zweite Methode ist für das Setzen eines Passwortes im System zugehörig zu einem bestimmten Nutzer gedacht. Dazu werden in die Methode, der Identifizierungstoken und das neue Password herein gegeben. Als erstes wird überprüft ob der Token im System vorhanden ist. Ist dies der Fall, wird sich die Instanz des Nutzers zugehörig zum Token geholt. Danach wird das Passwort des Nutzers neu gesetzt sowie auch die Funktion setRememberToken mit dem Token aufgerufen. Das Triggert erneut den Timer und setzt ihn auf die Ursprungszeit zurück. Als Rückgabe wird der veränderte Nutzer als JSON ausgegeben.
Methodenname: setPassword(String token, String password)
Schnittstelle: /user/setPassword

Medien Typ: Application_JSON

Die dritte Methode ist dafür da, sich den Nutzer und dessen Rechte anhand des bereitgestellten Tokens anzusehen. Als erstes wird der Token überprüft, ob er im System vorhanden ist. Ist dies der Fall, werden die Nutzerdaten, inklusive Rechte, als JSON zurückgegeben.
Methodenname: myRights(String token)
Schnittstelle: /user/myRights
Medien Typ: Application_JSON

0.3.3 genutzte Komponenten

Glassfish Jersey

Dient als Webserver Applikation im Tomcat. Kann durch beliebigen anderen, im Tomcat Container laufenden Webserver ausgetauscht werden. (Bsp.: Jetty, Spring Boot) Daher wird hier nicht weiter auf die Versionierung eingegangen. Der Applikationsserver muss aber das Package javax.ws.rs zur Verfügung stellen.

Base64Encoder

Wird zur Erzeugung der Token genutzt. Ist im Package java.util.Base64 enthalten.

Java UUID

Wird für die Generierung der Random UUID genutzt. Ist im Package java.util.UUID enthalten.

0.4 Nutzung

0.4.1 Code

0.4.2 Deployment / Runtime

Aus dem Quellcode lässt sich mithilfe von dem Befehl mvn packageëine WAR Datei erstellen. Dies ist möglich da Maven als Paket Manager eingesetzt wurde. Anschließend lässt sich die WAR auf jeden laufenden Tomcat Server einbinden.

Beschreibung wie die Komponenten aus dem Quellcode erzeugt werden kann, wie sie installiert wird und wie man sie startet.

0.5 Qualitätssicherung

Innerhalb der Abfragen wird überprüft, ob Eingaben vorhanden sind und ob diese mit den schon vorhandenen Daten übereinstimmen. Ist dies nicht der Fall, werden entsprechende Fehlermeldungen geworfen. Für die Spätere Entwicklung des Authentifizierungsservers werden aber Service und Unit Tests empfohlen.

0.5.1 Test

Finden ausschließlich per Hand mit dem Browser statt.

0.6 Vorschläge / Ausblick

Hier wurde Prototypisch aufgezeigt wie eine Authentifizierung stattfinden kann. Sobald die Implementierung mit der Datenbank und der Clients Stattfindet, wird empfohlen ein entsprechendes BCrypt Format zu integrieren. (z.B. JBCrypt). Dieses dient zur Verhashung des Passworts und verhindert das ein Passwort direkt aus der Datenbank ausgelesen werden kann oder in Klartext übertragen wird.