Rest-API absichern

Um eine API abzusichern muss vor dem Zugriff auf die Daten eine `Authentifizierung` sowie eine `Autorisierung` stattfinden.

Als Projektgrundlage kannst du ein bereits bestehendes Projekt verwenden, falls du ein neues anlegen möchtest/musst beachte bitte folgendes:

Erstelle auf https://start.spring.io/ ein Spring-Projekt mit den folgenden Dependencies:

- H2 Database
- Spring Security
- Spring Data JPA
- Lombok
- Rest Repositories
- ... wahlweise weitere Libraries, Datenbanktreiber etc.

Authentifizierung

Damit ist gemeint, dass sich ein Benutzer einloggen muss, um die Identität festzustellen. Dazu reicht eine `Basic Authentication`. Um Username+Passwort mitzusenden muss man den Request folgendermaßen aufbauen:

```
### get ressource
GET http://localhost:8080/<secret-ressource>
Authorization: Basic <username> <password>
```

Die übermittelten Daten kann man dann mit einem `Authentication`-Objekt abfangen beim Endpunkt. Allerdings sind die Daten darin NICHT verifiziert. Den Schritt muss man noch selber übernehmen.

```
@GetMapping("/<secret-ressource>")
Iterator<Object> getSecretRessource(Authentication auth){
    if(isAuthValid(auth)){ # todo implement a authentication check
        return repo.findAll();
    }
}
```

Speichern der Benutzerdaten

In Spring gibt es bereits das Interface `UserDetails`. Dieses speichert den Benutzernamen, das Passwort und enthält eine Liste von `GrantedAuthoritiy`. Damit kann man Berechtigungen festhalten. Es werden auch noch einige weitere Details gespeichert, die sind aber für die aktuelle Aufgabenstellung nicht wichtig.

Die Interfaces direkt kann man nicht speichern, jedoch kann man in je einer Klasse die Interfaces implementieren und mit einer `@Entity`-Annotation dazu wird es speicherbar.

Authorization

Um festzustellen ob ein bereits eingeloggter Benutzer auch tatsächlich berectigt ist bestimmte Ressorucen aufzurufen kann man mittels einer Access Control List pro Objekt und Benutzer berechtigungen vergeben.

> https://en.wikipedia.org/wiki/Access-control_list

Speichern von Berechtigungen

Bei Access-Control Lists kann für jedes Objekt seperat gespeichert werden wer welche Zugriffsberechtigung darauf hat. Bei der Klasse die das speichert, würde es sich anbieten dass diese `GrantedAuthority` implementiert. In dieser GrantedAuthority sollten folgende Daten enthalten sein:

- Id des Benutzers der Zugriff hat
- Id des Objects auf welches sich die Berechtigung bezieht
- Vollständiger Name des Objekts (z.B at.spengermed.patient)
- Die Berechtigung
- ... wahlweise weitere Daten, falls Hilfreich

Zu testende Situationen

Die folgenden Aufgaben sollen über das Aufrufen von Endpoints geschehen (via mockMvc):

- ~~Ein Benutzer wird neu angelegt (20P)~~
- ~~Ein Administrator teilt dem neuen Benutzer Berechtigungen zu (20P)~~
- Anlegen eines neuen FHIR-Objekts (z.B Patient, oder eine kleinere FHIR Ressource) (20P)
- Freigeben des neuen FHIR-Objekts von einem Benutzer zu einem anderen (20P)
- Aufrufen eines neuen FHIR-Objekts mit Berechtigungen (10P)
- Versuch ein Objekt mit fehlenden Berechtigungen aufzurufen (10P)

Checkliste vor der Abgabe

- Sind alle Änderungen comitted?
- Sind alle Änderungen gepusht und hat jeder den gleichen Stand?
- Stürzt das Programm ab wenn man es startet oder die neuen Endpunkte (mit Fehlerhaften Daten) aufruft?
- Werden alle Tests erfolgreich ausgeführt in der Konsole mit den Befehlen `gradle test` oder `mvn test`?
- Hat jeder ungefähr gleich viel comitted?

Anleitungen

> Beispielrepository + Anleitung

> Aktuelles Spring Security Tutorial