Kompetenznachweis Modul 295 - Hinweise

Abgabe

- Abzugeben ist eine ZIP-Datei pro Teilnehmer. Dateiname: *Ihr-Nachname_Ihr-Vorname_Projektarbeit_Modul_295.zip*
- Der Kursleiter wird den Ablageort im Kurs bestimmen (Netzwerkfreigabe oder Moodle Upload)
  Achtung: Bis zur Abgabe sind Sie für die sichere Aufbewahrung ihrer Arbeit zuständig!
- Die ZIP-Datei muss beinhalten:
  - Projekt (Source Code) als ZIP (siehe nächtes Kapitel)
  - Projektbeschreibung (siehe Definition_Projektarbeit.pdf)
  - Dokument, welches die für den Start des Projekts nötigen Konfigurationseinstellungen enthält:
    - Datenbankverbindung (Name DB, User, Passwort)
    - Einstellung Keycloak (Realmname, Rollen)
    - Verwendete Netzwerk Ports falls nicht die Standardwerte verwendet werden.
    - Weitere Informationen, welche der Kursleiter benötigt, um das Projekt bei sich zu starten.

Setup Projekt

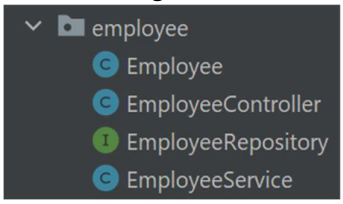
- Sie können entweder ein neues Spring Boot Projekt mit dem Spring Initializr (https://start.spring.io) erstellen oder das Demoprojekt als Basis verwenden. Wenn das Demoprojekt verwendet wird, müssen alle nicht für ihr Projekt relevanten Daten / Dateien entfernt werden.
- Als Basispackage ist zu verwenden: *ch.[name.vorname].[projektname]*
- Die Projektstruktur und die Codestruktur sollte dem Demoprojekt entsprechen. Selbstverständlich können Sie, wo nötig, Erweiterungen anbringen.
- Den Source Code können Sie in einem beliebigen GitHub Repository ablegen. Sie geben am letzten Tag das ganze Projekt (inklusive .git Verzeichnis) als ZIP Datei dem Kursleiter ab.
- In Keycloak erstellen Sie einen neuen Realm der ihrem Projektnamen entspricht.

Datenbankanbindung

- Als Datenbank ist PostgreSQL zu verwenden.
- Verwenden Sie, wie im Demoprojekt, JpaRepositories um auf die Datenbank zuzugreifen. Die Datenbankstruktur sollte beim ersten Start der Applikation automatisch erstellt werden (Standard im Demoprojekt). Dies kann über die entsprechende Einstellung in der Konfigurationsdatei application.yml gesteuert werden.

```yaml
spring:
  jpa:
    show-sql: true
    generate-ddl: true
    hibernate:
      ddl-auto: update
```

- Zwischen einem Repository und einem Controller erstellen Sie eine Service Klasse, welche den Datenbankzugriff und, falls nötig, weitere Logik beinhaltet.



- Ihre Entitäten konfigurieren Sie mit sinnvollen Annotationen, um die Validität der verwalteten Daten sicherzustellen.

Implementierung REST-Controller

- Unterteilen Sie ihren Code in sinnvolle REST-Controller Klassen. Achten Sie darauf, das Single-Responsibility-Prinzip zu berücksichtigen. Beispiel Demoprojekt: Die Klasse EmployeeController nimmt alle Anfragen zu Mitarbeitern entgegen. Die Klasse ist z.B. nicht für Fahrzeuge zuständig.
- Sie dürfen natürlich auch mehr als die geforderten 4 Controller implementieren, falls dies für Ihre Applikation nötig ist.
- Vergewissern Sie sich, dass die Controller Methoden über das Token abgesichert sind! Dies geht sehr schnell einmal vergessen. Verwenden Sie sinnvolle Benutzerrollen.
- Optimieren Sie die Serialisierung/Deserialisierung ihrer Entitäten mit sinnvollen Annotationen (z.B. @JsonIgnore).

Testing

- Die Swagger UI Komponente muss konfiguriert werden damit eine Authentifizierung mittels Bearer Token durchgeführt werden kann. Beachten Sie dazu die Klasse OpenApi30Config und die jeweiligen Controller Annotationen im Demoprojekt.
- Komplettieren Sie die Swagger Konfiguration durch den Einsatz sinnvoller Annotation, um Operationen zu gliedern und Rückgabewerte und Parameter transparent zu beschreiben. Verwenden Sie dazu Annotation wie z.B.
    - @Parameter
    - @Tag
    - @ApiResponse
    - Etc.
- Erstellen Sie einen JUnit Test für einen beliebigen REST-Controller (muss zumindest CRUD Methoden implementieren). Überprüfen Sie die CRUD-Methoden.
- Erstellen Sie einen JUnit Test für ein JpaRepository. Testen Sie auch hier alle CRUD-Operationen.