Software Engineering & Projektmanagement
SEPM Gruppenphase - Getting Started Guide

Version 4.0

(28. März 2019)

Inhaltsverzeichnis

1 Einführung **3**

2 Installation **4**
2.1 Technische Voraussetzungen 4
2.2 Java 4
2.3 Angular 7 4
2.4 Entwicklungsumgebung 4
2.5 SEPM Grundgerüst zur Gruppenphase 5
2.5.1 Backend 5
2.5.2 Frontend 5
2.5.3 Datenbank 6
2.5.4 Data Generator 6

3 Backend **7**
3.0.1 Endpoint Package 8
3.0.2 Swagger Annotations & SwaggerUI 9
3.0.3 Service Package 11
3.0.4 Entity Package 11
3.0.5 Repository Package 12

4 FAQ **13**
4.1 Was darf am Grundgerüst geändert werden? 13
4.2 Dürfen zusätzliche Bibliotheken verwendet werden? 13
4.3 Ich kenne mich mit Technologie-XY nicht aus, was nun? 13

1 Einführung

Das SEPM Gruppenphasengrundgerüst ermöglicht Ihnen die Entwicklung sofort mit einer vollständig lauffähigen Webapplikation zu beginnen, ohne sich mit dem Projektsetup beschäftigen zu müssen. Darüber hinaus wurden bereits einige Funktionen implementiert, um Ihnen das Zusammenspiel der einzelnen Komponenten zu demonstrieren. Der bereitgestellte Sourcecode muss verändert, erweitert und angepasst werden.

Dieser Guide soll Ihnen bei den ersten Schritten in der SEPM Gruppenphase helfen. Er beginnt mit einer Installationsanleitung und stellt Ihnen im Anschluss die einzelnen Projekte und Komponenten vor, sowie deren Anwendung.

Wir sind bemüht diesen Guide so gut wie möglich an die Bedürfnisse der Studierenden anzupassen. Dazu benötigen wir jedoch Ihre Hilfe. Wenn Sie Anregungen, Fehler oder Kritikpunkte gefunden haben, so zögern Sie bitte nicht uns diese mitzuteilen. Entweder über Ihren Tutor oder direkt per Mail an sepm@inso.tuwien.ac.at bzw. an sepm@qse.ifs.tuwien.ac.at

Viel Spaß und gutes Gelingen!

2 Installation

2.1 Technische Voraussetzungen

Die folgenden Unterkapitel erläutern das notwendige technische Setup, um das Grundgerüst der SEPM Gruppenphase verwenden zu können. Halten Sie sich an die vorgegebenen Versionsnummern, nur diese werden von uns supported.

2.2 Java

Für die SEPM Gruppenphase wird - analog zur SEPM Einzelphase - **OpenJDK 11**[1] verwendet. Ihre bisherige OpenJDK Installation kann somit weiterverwendet werden.
Beachten Sie jedoch, dass die *JAVA_HOME* Umgebungsvariable gesetzt ist (z.B. C:\Programme \Java\jdk-11_xx). Dies ist wichtig, da Maven die *JAVA_HOME*-Umgebungsvariable verwendet. Zusätzlich muss die *PATH*-Umgebungsvariable um den Ordner "JAVA_HOME"\bin erweitert werden (z.B. C:\Programme\Java\jdk-11_xx\bin).

2.3 Angular 7

Für die Entwicklung des Webfrontends kommt Angular 7 zum Einsatz. Installieren Sie dafür **Node.js 10**[2]. Node.js ist eine JavaScript Laufzeitumgebung. Mit Node.js wird automatisch **npm** installiert. Der Node Package Manager (npm) dient zur Installation diverser JavaScript Pakete, u.a. Angular 7 und Angular CLI, die für die Entwicklung des Webfrontends benötigt werden.

2.4 Entwicklungsumgebung

Im Rahmen der LVA sollen Sie eine moderne Entwicklungsumgebung benützen, wobei in SEPM IntelliJ von Jetbrains verwendet wird. Diese steht in einer freien Community Edition und einer kostenpflichtigen Ultimate Edition zur Verfügung. Für die SEPM Gruppenphase empfehlen wir die Ultimate Edition!

Studierende und Lehrende können eine kostenfreie Educational License zu beantragen[3], die sie während ihres Studiums verwenden können.

[1] `https://jdk.java.net/11/`
[2] `https://nodejs.org/en/download/`
[3] `https://www.jetbrains.com/student/`

2.5 SEPM Grundgerüst zur Gruppenphase

Als ersten Schritt müssen Sie das SEPM Gruppenphasengrundgerüst aus TUWEL herunterladen. Entpacken Sie es in ein Verzeichnis Ihrer Wahl. Das Grundgerüst besteht aus zwei Projekten. Das Projekt *backend* beinhaltet das REST Backend, das Projekt *frontend* beinhaltet das Webfrontend in Form eines Angular Clients.
Beachten Sie, dass es sich dabei um zwei getrennte Projekte handelt (backend, frontend), die in IntelliJ auch als solche separat, in zwei unterschiedlichen Fenstern, geöffnet werden müssen.

2.5.1 Backend

Um nun das REST Backend zu starten, wechseln Sie in der Kommandozeile in das *backend* Verzeichnis. Führen Sie dort folgendes Kommando aus (siehe Listing 1).

Listing 1: Maven run Backend

```
1 ./mvnw spring-boot:run
```

Dadurch werden alle benötigten Bibliotheken heruntergeladen und das Backend gestartet. Der Prozess darf nicht abgebrochen und das Kommandozeilenfenster nicht geschlossen werden, da sonst der Server beendet wird.

Hinweis: Unter Windows kann sich das Kommando je nach verwendeter Windowsversion und Terminal leicht unterscheiden. Unter Umständen müssen Sie statt "./mvnw" nur "mvnw" oder auch "mvnw.cmd" schreiben

2.5.2 Frontend

Einmalig müssen Sie im frontend Projekt (in der Kommandozeile im *frontend* Verzeichnis) folgende Befehle ausführen (siehe Listing 2)

Listing 2: Installation JS Packages

```
1 npm install
2 npm install -g @angular/cli
```

Das Frontend kann mit folgendem Befehl (in der Kommandozeile im *frontend* Verzeichnis) gestartet werden (siehe Listing 3)

Listing 3: Start Frontend

```
ng serve
```

Nachdem Frontend und Backend erfolgreich gestartet wurden, können Sie unter `http://localhost:4200/` im Webbrowser das Frontend aufrufen. Mit "**admin**" als Benutzernamen und "**password**" als Passwort und dem Klick auf *Login*, können Sie sich als Administrator anmelden.
Für eine Anmeldung als normaler Benutzer, verwenden Sie "**user**" als Benutzernamen und "**password**" als Passwort.

2.5.3 Datenbank

Beim Starten des Backends wird automatisch eine Datenbank mitgestartet. Die Datenbank wird im Unterverzeichnis *./database/* erstellt und sollte nie in das Git-Repository eingecheckt werden.
Der Server ist so konfiguriert, dass versucht wird das Datenmodell in der Datenbank entsprechend dem in den Entities definierten Datenmodell zu aktualisieren. Da das nicht immer funktioniert, kann es sein, dass Sie manuell die Datenbank löschen müssen. Sie wird dann automatisch beim nächsten Start neu erstellt.
Mit der Datenbank startet auch ein Webserver, über den direkt auf die Datenbank zugegriffen werden kann. Über die URL `http://localhost:8080/h2-console` können Sie eine Verbindung zum integrierten Datenbankmanager aufbauen.
Geben Sie im JDBC-Connection-String den absoluten Pfad zu Ihrer Datenbank an:
jdbc:h2:file:<aboslutePathToProject>/database/backend
Als Username verwenden Sie *admin* und als Passwort *password*.

2.5.4 Data Generator

Das Backend bietet die Möglichkeit Testdaten zu generieren, sollten diese noch nicht vorhanden sein. Dies wird über eigene DataGenerator-Klassen erreicht. Starten Sie dazu Ihren Server mit dem entsprechenden Profil (siehe Listing 4).

Listing 4: Maven run backend server with demo data

```
./mvnw spring-boot:run -Dspring-boot.run.profiles=generateData
```

3 Backend

In diesem Kapitel werden das Backend und dessen Packages genauer beschrieben, sowie deren Struktur und Verwendung erläutert. Abbildung 1 zeigt alle Packages des Servers und deren Abhängigkeiten.
Das Spring-Framework bietet mehrere Varianten zur Konfiguration einer Applikation an, wobei in SEPM die Konfiguration über Annotationen und Konfigurationsklassen erfolgt.

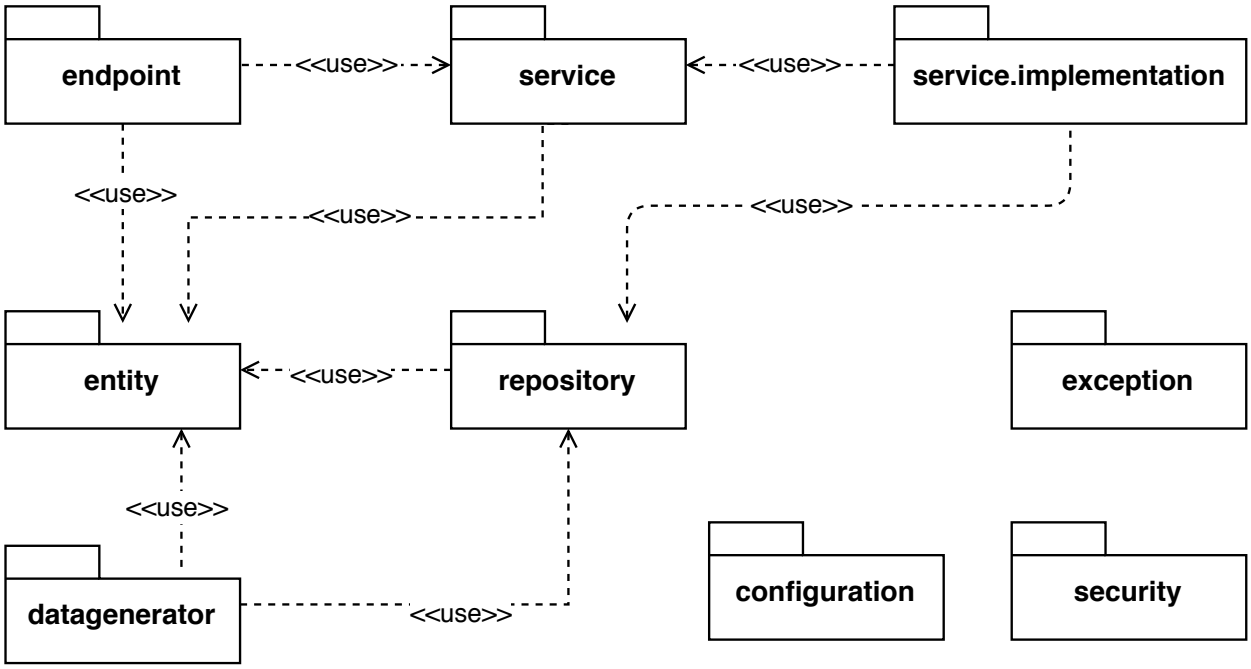

Abbildung 1: Backend Packages

Folgende Konfigurationsdateien (im *YAML*-Format) sind für das Backend relevant:

application.yml
In der *application.yml* werden alle Standardkonfigurationen eingetragen. Das beinhaltet zum Beispiel Einstellungen zum Logging, aber auch Informationen zur Datenbankverbindung oder zur Anwendungssicherheit.
Spring kann weitere Konfigurationsdateien je nach angegebenem Profil laden. Für ein *test*-Profil würde die Konfigurationsdatei dann beispielsweise *application-test.yml* heißen.

@Configuration-Klassen
Zusätzlich zur Konfiguration mit *YAML*-Files werden sogenannte Configuration-Klassen verwendet. Es handelt sich dabei um Java-Klassen, die mit der *@Configuration*-Annotation versehen sind. Spring erkennt diese Klassen automatisch und führt Sie beim Starten der Anwendung aus.
Hier wird beispielsweise die JSON Darstellung der Objekte, aber auch die Applikationssi-

cherheit und die Swagger-Dokumentation konfiguriert.
In der Security-Configuration können beispielsweise neue User eingefügt werden.

3.0.1 Endpoint Package

Das Package *at.ac.tuwien.sepm.groupphase.backend.endpoint* enthält die serverseitigen REST Schnittstellen, die vom Angular Frontend aufgerufen werden. Jeder Request an das Backend führt zu einem Aufruf einer der Methoden, die mit der *@RequestMapping*-Annotation versehen sind (siehe Listing 5).

Listing 5: Beispiel Message Request Mapping

```
1 @RestController
2 @RequestMapping(value = "/messages")
3 @Api(value = "messages")
4 public class MessageEndpoint {
5
6    // ...
7
8    @RequestMapping(method = RequestMethod.GET)
9    @ApiOperation(value = "Get␣list␣of␣simple␣message␣entries")
10   public List<SimpleMessageDTO> findAll() {
11      return messageMapper.messageToSimpleMessageDTO(messageService.
            findAll());
12   }
13
14   // ...
15
16 }
```

Mit der *@RequestMapping*-Annotation wird also das Mapping zwischen URL und Funktion definiert. Durch den *method*-Parameter können die gültigen HTTP Request Arten festgelegt werden (GET, POST, PUT, DELETE). Weiterhin können mit *@RequestParam* und *@PathVariable* Funktionsparameter definiert werden. Auch andere Arten der Parameterübergabe sind möglich. Weitere Informationen dazu finden Sie in der Spring Dokumentation[4].
Jede Controller-Klasse muss mit *@RestController* annotiert werden, da Spring diese Klasse nur so beim Starten des Servers instanziiert.
Abhängigkeiten, die von der Klasse benötigt werden, werden im Konstruktor übergeben. Wenn Spring die Klasse instanziiert, werden die angegebenen Objekte automatisch überge-

[4] http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-developing-web-applications.html

ben, sofern diese bereits in Spring vorhanden sind, dh bereits vorher von Spring instanziiert wurden. Dieses Verfahren wird als "constructor injection" bezeichnet.
Im Codebeispiel (siehe Listing 6) wird beispielsweise eine Instanz des *MessageService* und eine Instanz des *MessageMapper* im Konstruktor angefordert. Da diese beiden Klassen Spring bereits bekannt sind, werden Sie beim Instanziieren von *MessageEndpoint* automatisch übergeben.

Listing 6: Konstruktor Parameter

```
1 @RestController
2 @RequestMapping(value = "/messages")
3 @Api(value = "messages")
4 public class MessageEndpoint {
5
6    private final MessageService messageService;
7    private final MessageMapper messageMapper;
8
9    public MessageEndpoint(MessageService messageService, MessageMapper
         messageMapper) {
10       this.messageService = messageService;
11       this.messageMapper = messageMapper;
12   }
13
14   // ...
15
16 }
```

3.0.2 Swagger Annotations & SwaggerUI

Im Package *at.ac.tuwien.sepm.groupphase.backend.endpoint* werden zudem auch Annotationen des Swagger-Frameworks[5] verwendet, mit deren Hilfe automatisiert eine interaktive Dokumentation der REST-Schnittstelle erstellt wird.
Beispielsweise werden die Annotationen *@Api(...)* und *@ApiOperation(...)* verwendet.

Das Swagger-Projekt bietet mit SwaggerUI auch ein User-Interface für die Anzeige der Dokumentation, sowie zum Testen der REST-Schnittstellen. Sobald der Server gestartet ist, erreichen Sie unter der URL `http://localhost:8080/swagger-ui.html` das SwaggerUI.
Achten Sie darauf, dass Sie sich gegenüber der REST-Schnittstelle authentifizieren müssen, da diese sonst für die meisten Anfragen eine "401 - Unauthorized" Fehlermeldung liefert. Kli-

[5] `https://github.com/swagger-api/swagger-core/wiki/Annotations`

cken Sie dazu in der SwaggerUI zuerst auf den POST-Request des authentication-Endpoints und schicken Sie Benutzername und Passwort an den Server. Als Antwort erhalten Sie zwei Tokens, den *currentToken* und den *futureToken*. Klicken Sie danach auf den Button "Authorize" und fügen Sie hier den currentToken mit einem vorangestellten "Bearer" ein. Danach sind Sie, je nach Rolle für die weiteren Anfragen authorisiert. Der ausgestellte Token ist in der Standardkonfiguration 600 Sekunden lang gültig.

3.0.3 Service Package

Im Package *at.ac.tuwien.sepm.groupphase.backend.service* werden die Interfaces für die Service Klassen definiert und das Subpackage *implementation* enthält dementsprechend die Implementierungen dieser Interfaces (siehe Listing 7).

Listing 7: Spring Annotationen im MessageService

```
1 @Service
2 public class SimpleMessageService implements MessageService {
3
4    private final MessageRepository messageRepository;
5
6    public SimpleMessageService(MessageRepository messageRepository) {
7       this.messageRepository = messageRepository;
8    }
9
10   @Override
11   public List<Message> findAll() {
12      return messageRepository.findAll();
13   }
14
15   // ...
16
17 }
```

Jede Service Implementierung muss mit *@Service* annotiert sein. Andernfalls wird die Klasse nicht beim Starten des Servers instanziiert und das Injecten im Controller würde nicht funktionieren.
Auch im Service werden die Abhängigkeiten über den Konstruktor übergeben. Hierdurch wird die DAO, die für den Datenbankzugriff benötigt wird, übergeben. Dies funktioniert wiederum nur, wenn die DAO Implementierung mit *@Repository* annotiert ist.

3.0.4 Entity Package

Dieses Package enthält die Entities. Die Klassen, die für den Beispiel Anwendungsfall relevant sind, sind bereits enthalten.
Da in unterschiedlichen Ansichten jeweils nicht alle Daten benötigt werden, werden die Entity Objekte mit Mappern in die entsprechenden REST-Objekte (dto) umgewandelt. Dazu wird die Bibliothek MapStruct[6] verwendet. Mit ihrer Hilfe können sehr einfach automatisiert auch komplexere Mapper erstellt werden.

[6] `http://mapstruct.org`

3.0.5 Repository Package

Das Package *at.ac.tuwien.sepm.groupphase.backend.repository* enthält bisher das Interface *MessageRepository*, das das Interface *JpaRepository* erweitert. *JpaRepository* ist eine vom Spring Framework bereitgestellte Klasse, die CRUD-Funktionalitäten[7] zur Verfügung stellt. Weiters müssen die DAOs mit *@Repository* annotiert werden. Um die DAOs mit einfachen Abfragen zu erweitern, kann die JPA Criteria API verwendet werden. Kompliziertere Abfragen lassen sich mittels JPQL und der *@Query* Annotation definieren. Weitere Informationen zu beiden Varianten finden Sie in der JPA Repository-Dokumantation[8].

[7] `http://de.wikipedia.org/wiki/CRUD`
[8] `http://docs.spring.io/spring-data/jpa/docs/current/reference/html/`

4 FAQ

4.1 Was darf am Grundgerüst geändert werden?

Sie können bzw. müssen den Code nach Belieben ändern und an Ihre Bedürfnisse anpassen.

Das Grundgerüst soll Sie dabei unterstützen um möglichst rasch mit der Umsetzung der eigentlichen Funktionalität starten zu können.

Weiters handelt es sich bei der bestehenden Applikation um einen sogenannten *Durchstich*, d.h. die Applikation enthält einen kompletten UseCase, der einen Zugriff vom User Interface durch alle Schichten bis zur Datenbank zeigt.

4.2 Dürfen zusätzliche Bibliotheken verwendet werden?

Sollten Sie zusätzlich Bibliotheken benötigen, binden Sie diese über Maven ein. Achten Sie hierbei auf die jeweiligen Lizenzbestimmungen.

4.3 Ich kenne mich mit Technologie-XY nicht aus, was nun?

An erster Stelle steht hier die eigene Recherche. Lesen Sie die Dokumentation der betreffenden Technologie und suchen Sie im Internet (z.B. auf https://stackoverflow.com/) nach Lösungen für Ihr Problem. Sollten Sie danach nicht weiterkommen, wird Ihr Tutor Ihnen beratend zur Seite stehen.