FH Erfurt

Angewandte Informatik

Module: Programmierung Java 1

Dozent: Professor Rhöse

Projektarbeit:

Dokumentation

Student_self_organization-Service

vorgelegt von:

Felix Zwicker - DevLead

Matr.-Nr.: 120048201

E-Mail: felix.zwicker@fh-erfurt.de

Friedemann Taubert

Matr.-Nr.: 120060157

E-Mail: friedemann.taubert@fh-erfurt.de

Maximilian Keller

Matr.-Nr.: 120063608

E-Mail: maximilian.keller@fh-erfurt.de

Jenny Wagner

Matr.-Nr.: 120061365

E-Mail: jenny.wagner@fh-erfurt.de

Tim Eisenberg

Matr.-Nr.: 120066175

E-Mail: tim.eisenberg@fh-erfurt.de

Abgabetermin: 25. März 2022

Inhaltsverzeichnis

1. Was macht der Service?

2. Service Struktur
   2.1. Forum
   2.2. FAQ
   2.3. To-do-Liste
   2.4. Info-Page

3. UML - Diagramme
   3.1. Forum
   3.2. FAQ
   3.3. Todo
   3.4. Info-Page

4. Top-Down-Sicht

1. Was macht der Service?

Der Service befasst sich mit der Studentischen Selbstverwaltung und beinhaltet ein Forum, FAQ, eine To-do-Liste und Infoseite.
In dem Forum ist es dem Studenten oder auch dem Dozenten möglich verschiedene Fragen zu erstellen, worauf andere Nutzer antworten können. Das FAQ fasst alle wichtigen und oft gestellten Fragen in einer Übersicht zusammen. Die To-do-Liste ist der organisatorische Rahmen für das Managen aller Aufgaben eines Studenten. Sie ist eine Aufzählung von mehreren Tätigkeiten, die der Nutzer selbst hinzufügen, bearbeiten und löschen kann.

2. Service Struktur

Unser Service ist unterteilt in das Forum, FAQ, die To-do-Liste und Infoseite.

2.1. Forum

Unser Forum ist unterteilt in 4 verschiedene Klassen:

- ***ForumRepository***

  Die Klasse Forum bildet hierbei den Kern des Forums an sich und alles andere ist zentral um ihn herum aufgebaut. Im Forum befinden sich Topics, welche in Form einer Array-Liste dargestellt sind. Weitere Array Listen gibt es für Fragen und Antworten. Wir haben uns für Array Listen entschieden, da für die vordefinierten Funktionen den Umgang bei der Implementierung des allgemeinen Codes und der Tests sehr einfach gestaltet haben.
  In der Klasse Forum befindet sich zuerst ein Konstruktor des Forums, danach findet man die triviale Methode addNewQuestions, welche keinen Test benötigt. Um Fragen zu entfernen haben wir die Methode removeQuestions implementiert, welche eine Frage löscht, sobald der eingegebene, zu löschende Name der Frage gefunden wurde. Hierbei haben wir uns als Rückgabewert für boolean entschieden, um den Fall, dass die Frage nicht existiert, mit abzudecken.
  Als Nächstes haben wir die Methode getByTopic implementiert, welche eine Array-Liste von Fragen zurückgibt, sollte die zu suchende Topic existieren. In diesem Fall lösen wir das Problem, falls die Topic nicht gefunden wird nicht mit einem boolean Wert, sondern geben eine leere Liste zurück, welche in jedem Fall beim Start der Methode erstellt wird. Im Zusammenspiel mit getByTopic ist die Methode isTopic entstanden, welche lediglich überprüft, ob die zu suchende Topic existiert. Hierbei findet wieder eine boolean Variable Verwendung.
  Im Anschluss haben wir getByQuestionTitle implementiert, welche nahezu analog zur Methode getByTopic aufgebaut ist. Hier wird lediglich nicht vorher überprüft, ob die zu suchende Frage existiert. Der Rückgabewert ist ebenfalls wieder eine Array-Liste. Die Methode getQuestionbyAuthor ist analog zur vorherigen getByQuestionTitle aufgebaut.
  Für die Methode addNewTopic kommt wieder boolean als Rückgabewert zum Einsatz, da wir hierbei überprüfen, ob bereits eine Topic mit diesem Namen existiert. Der Rückgabewert ist true, wenn noch keine Topic mit diesem Namen existiert. Ähnlich läuft removeTopic ab, hierbei ist der Rückgabewert jedoch true, wenn wir das Topic gefunden haben und false, wenn nicht. Das

Löschen und Hinzufügen der Topics wird über die bereits erwähnten vordefinierten Befehle einer Array-Liste umgesetzt.

- ***Answer***
  Die Klasse Answer stellt eine Antwort auf eine Forum Frage dar. Sie beinhaltet nur eine Message.
  getTitle ist eine Methode um den Titel der gespeicherten Nachricht auszugeben.

- ***Message***
  Die Klasse Message stellt eine Nachricht dar die gekennzeichnet wird durch Titel, Text und Autor und verfügt nur über normale Getter und Setter Methoden.

- ***Question***
  Die Klasse Question stellt das Gebilde einer Forum Frage dar. Als Eigenschaften besitzt sie eine Message, Topic und eine Array-Liste mit dazugehörigen Antworten.
  Die Methode addAnswer fügt eine Antwort zu der Forum Frage hinzu. removeAnswer löscht die Antwort falls sie existiert. getTitle gibt den Titel der Nachricht zurück. getAuthor gibt den Autor der Nachricht zurück. getTopic ist eine normale Getter Funktion.

2.2. FAQ

Das FAQ ist unterteilt in die Core-Elemente und dem Repository.
Zu den Core-Elementen gehören:

- ***Enums***
  Die Enumerations SortDirection und SortPriority sammeln die Sortiermöglichkeiten.

- ***EntryNotFoundException***
  In dieser Klasse ist eine simple Exception definiert, welche eine ErrorMessage übergeben bekommt. Diese Klasse kann dann von anderen Klassen aufgerufen werden, um eine Exception zu werden.

- ***Element***
  Die Klasse Element wird genutzt um die Inhalte des FAQ´s zu erstellen, indem man ihr die benötigten Daten übergibt. Dabei wird das Builder pattern benutzt um das erstellen und updaten zu vereinfachen. Besonders wenn in einem späteren Stand des Services verschiedene Attribute für ein FAQ-Element benötigt oder nicht benötigt werden.
  Zudem besitzt die Klasse einen Override toString, um das Arbeiten mit der Klasse zu vereinfachen.

- ***SortSettings***
  In SortSettings können die gewünschten Sortiereinstellungen erstellt werden. Dabei werden ihr die Enums SortDirection und SortPriority übergeben.

Das Repository stellt die noch nicht vorhandene Datenbank dar, in der die erstellten Elemente gesammelt und verwaltet werden.
Zu dem Repository gehören:

- ***SortFaq***
  In dieser Klasse werden durch die zuvor definierten SortSettings zwei Objekte einer Collection oder in unserem Fall einer Liste miteinander verglichen und ein entsprechender int-Wert zurückgegeben. Dieser wird in der FaqRepository Klasse genutzt.

- ***FaqRepository***
  Die Klasse FaqRepository beinhaltet den Kern des FAQ.
  Das FaqRepository wird über eine Liste “faqList” realisiert, welche die FAQ-Elemente sammelt.
  Die Funktion addElement fügt ein Element der Liste hinzu, dafür wird ein Element an die Funktion übergeben. Danach wird geschaut welche ID das zuletzt hinzugefügte Element besitzt und dementsprechende wird die ID des neu übergebenen Elements Eins höher gesetzt. Sollte dies das erste Element für die Liste sein, so wird die ID auf Eins gesetzt. Durch “add” wird das Element schließlich der Liste hinzugefügt.
  Die Funktion getElementById bekommt eine ElementId übergeben, nach dieser wird dann durch einen Stream die Liste durchsucht. Das gefundene Element, welche mit der gesuchten ID übereinstimmt, wird dann zurückgegeben. Sollte kein Element mit der gesuchten ID existieren, so wird die EntryNotFoundException geworfen.
  Die Funktion deleteElementById entfernt ein Element aus der Liste. Dafür wird ihr die ID des gesuchten Elements übergeben und durch die Funktion getElementById das gesuchte Element. Schließlich wird durch “remove” das Element entfernt.
  Die Funktion GetElementByAuthor durchsuchte die Liste nach Elemente mit dem selben Author und sammelte diese in einer neu erstellten Liste. Diese wird dann zurückgegeben. Sollte kein Element mit gesuchten Author existieren wird auch hier einen Exception durch EntryNotFoundException geworfen.
  Die Funktion sortList sortiert die Liste. Dafür müssen ihr erstellte SortSettings übergeben werden. Durch die SortFaq Klasse werden dann immer zwei Elemente der Liste miteinander verglichen und nach den SortSettings neu zusammengelegt.
  getFaqList gibt die Liste mit allen Elementen zurück.
  clearFaqList entfernt jeden Inhalt der Liste.
  printFaqList gibt die Liste in der Konsole aus, wobei die toString Funktion von der Klasse Element genutzt wird. Dies war besonders hilfreich, da man den Inhalt der List in String-Form sehen konnte und somit die einzelnen Funktionen leicht überprüfen konnte.

2.3. Todo

- ***Todo***

  Die Klasse Todo beinhaltet Funktionen zum erstellen einer LinkedList (“Todo”), zum hinzufügen eines Eintrags in die Liste(“addTask”), zum suchen eines Eintrags(“getTaskById”), sowie zum löschen eines Eintrages (“deleteTaskById”) oder aller Einträge (“clearTodoList”) und zum anzeigen der Todo-Liste(“printTodoList”). Ebenfalls beinhaltet die die Klasse Todo die Methode check und uncheck zum setzen der Variable isChecked zu true oder false. Um ein Task der Liste hinzuzufügen wird die addTask Methode aufgerufen und ein Task übergeben. Wenn die Liste bereits Einträge hat wird die Variable lastTaskInList, lastTaskID, sowie eine ID für den nun hinzugefügten Eintrag, gesetzt. Falls noch kein Eintrag existiert wird die Id immer auf 1 gesetzt.
  DeleteTaskById wird eine taskId übergeben und die Methode löscht den Eintrag, der über die Id gefunden wurde. Zum finden des Eintrags wird in der Methode getTaskById verwendet.
  Diese Methode wirft eine Exception, falls kein Eintrag für die Id gefunden wurde.
  PrintTodoList gibt die Liste in der Konsole aus.
- ***Task***

  Definiert einen Eintrag mit den Variablen title, date, isChecked, taskId und priority. Es wird das Builder-pattern verwendet um den Umgang mit der Klasse und zukünftige Änderungen zu erleichtern.
- ***Enum-Priority***

  Der Enum Priority definiert die drei verschiedene Zustände: HIGHLY_IMPORTANT, IMPORTANT und INSIGNIFICANT. Diese können jedem Task zugeordnet werden.

2.4. InfoPage

Die InfoPage ist recht simpel aufgebaut. Sie besitzt einen Header, Content und eine Liste mit Person, die Ansprechpartner darstellt. Für Personen wird die Schnittstelle zum Person-Client genutzt, welcher uns Personen zur Verfügung stellt.

- ***InfoPage***

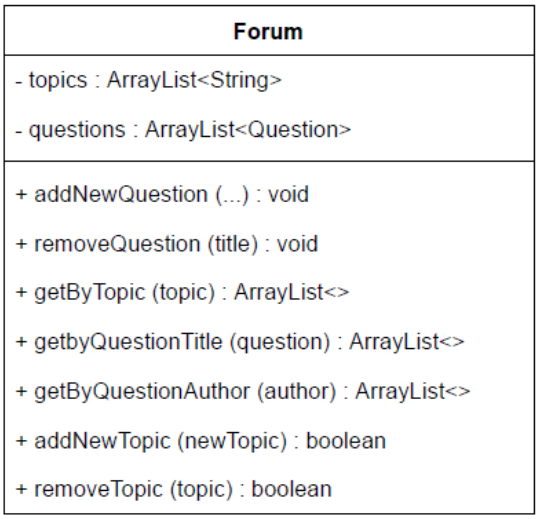
  Die Klasse wird genutzt um eine Informationsseite zu erstellen, indem man die gewünschten Daten übergibt.

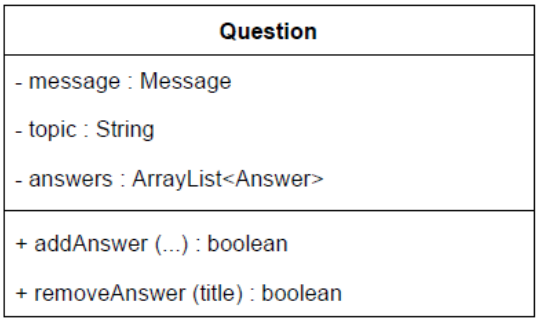
- **InfoPageRepository**

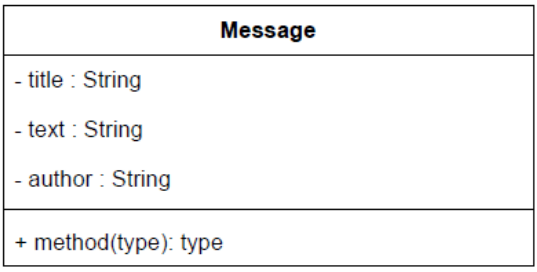
  Die Klasse wird genutzt um die Informationsseite zu verwalten.
  Die Funktion addPersonFromInfoPageById fügt der in der InfoPage erstellten Liste eine neue Person hinzu.
  Die Funktion removePersonFromInfoPageById durchsucht die Personenliste mittels Stream und entfernt die gefunden Person.
  updateHeader setzt den Header auf den übergebenen Wert.
  updateConten setzt den Content auf den übergebenen Wert.
  getInfoPage gibt die erstellte InfoPage zurück und showInfoPage gibt die InfoPage in der Konsole aus.

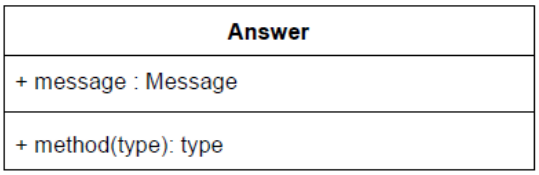
3. UML - Diagramme

3.1. Forum

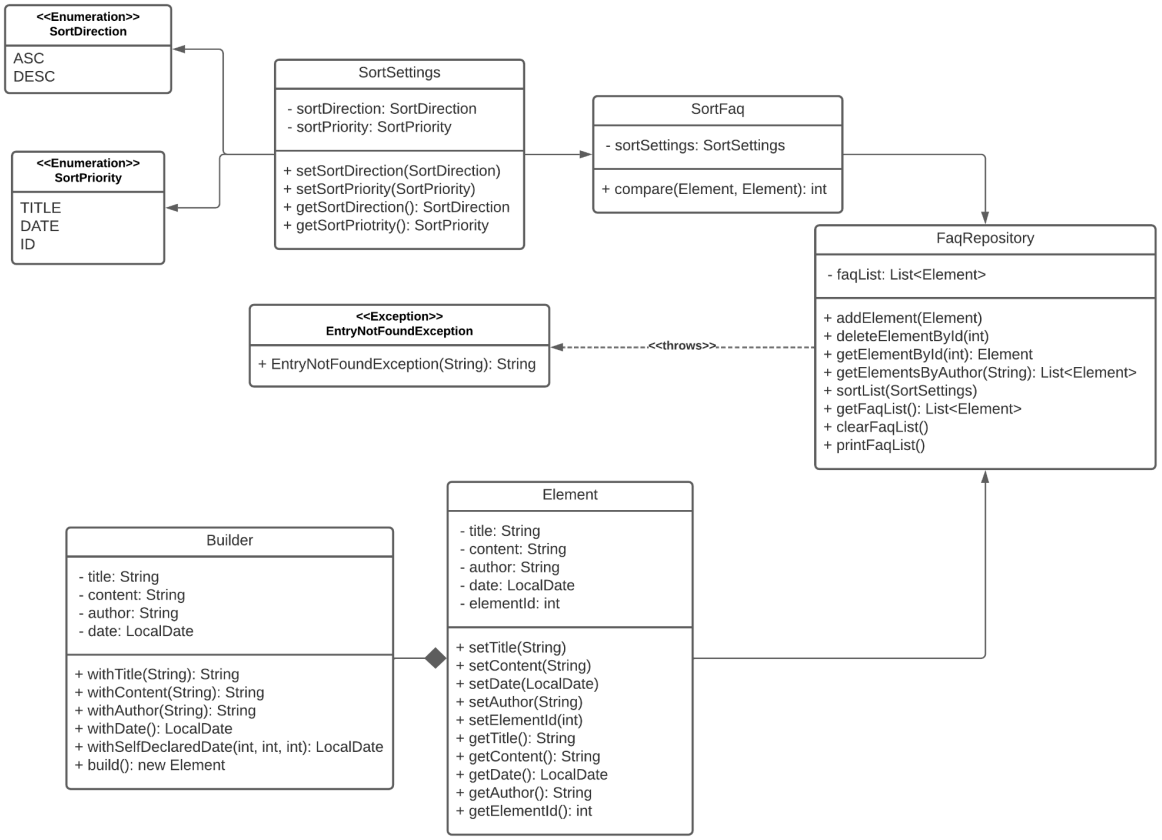




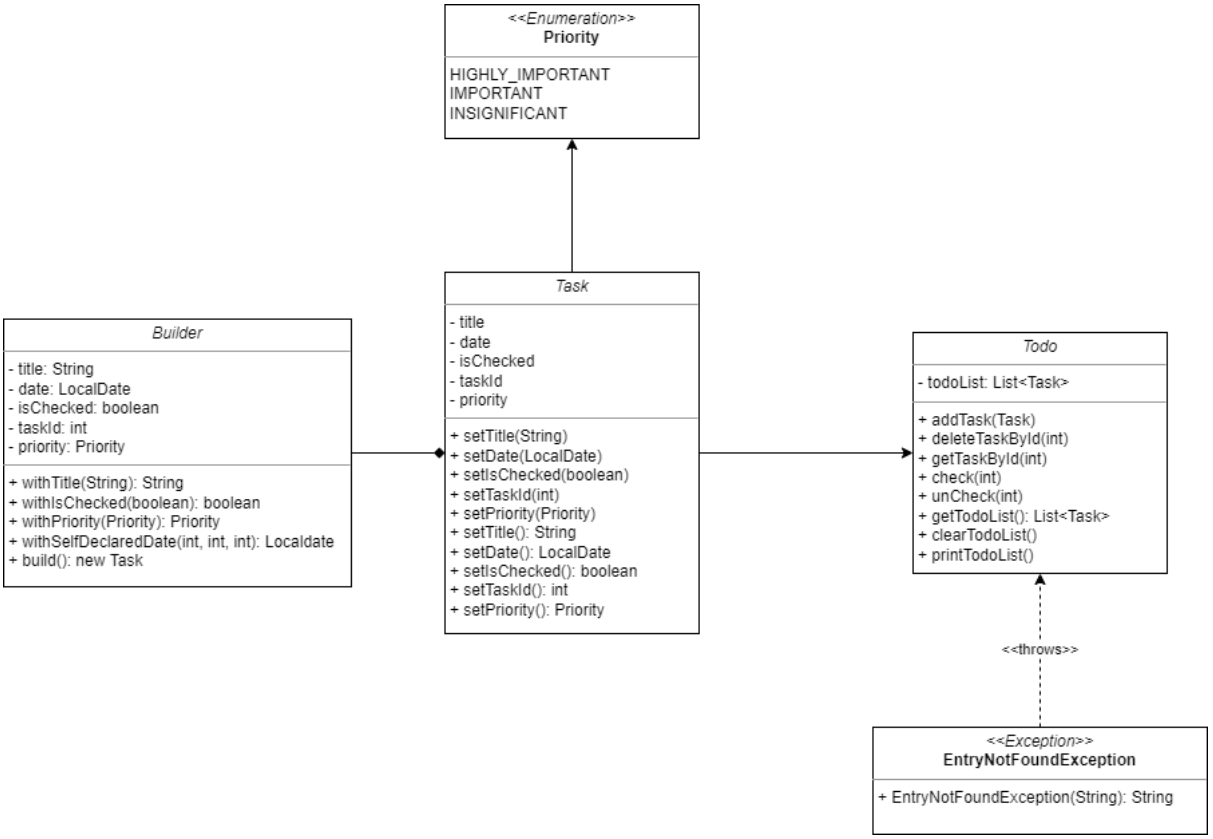




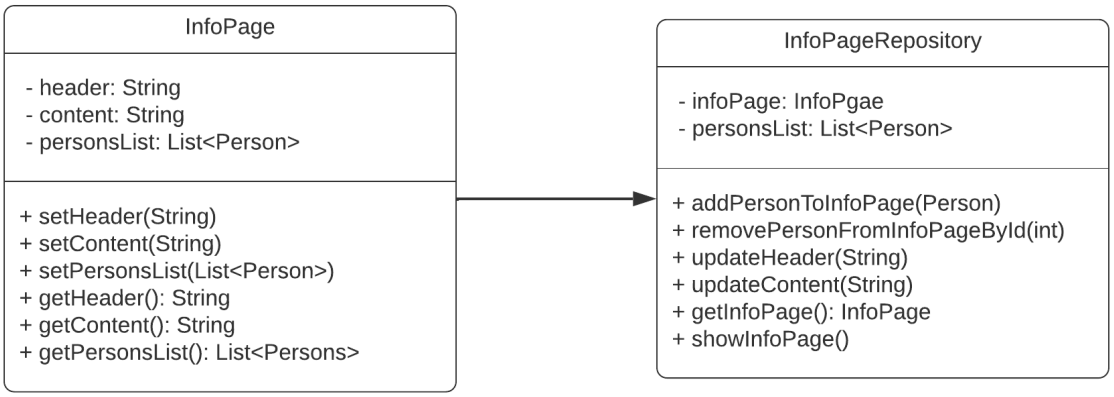
3.2. FAQ



3.3. Todo



3.4. Info-Page



4. Top-Down-Sicht

Der Service ist unterteilt in drei Layer:
Presentation-Layer: Dieser Layer beinhaltet die Interfaces für FAQ und Forum und kann von anderen Services genutzt werden.
Business-Layer: In diesem Layer befinden sich die Core-Elemente unseres Service.
Repository-Layer: Hier werden die gespeicherten Daten verwaltet.