

Thema:

Musik-Editor z. B. Tool zum Erstellen von Beats oder Sounds

Dokumentation

im Studiengang Engineering Business Information Systems
des Fachbereiches 2: Informatik und Ingenieurwissenschaften

Themensteller/in: Prof. Dr. Armin Lehmann

vorgelegt von: Mert Uzeken 1142543
Aryan Bashir 1458978
Dinara Kurmanbek 1346718

Wintersemester 2022/23

Abgabetermin: 05.03.2023

1	Einleitung.....	1
1.1	Ziel und Aufbau des Projekts.....	1
1.2	Probleme und Herausforderungen	1
1.3	Tools und Frameworks	1
2	Anforderungen und Funktionsumfang.....	2
3	Projektplanung	3
4	Umsetzung	5

1 Einleitung

1.1 Ziel und Aufbau des Projekts

Ziel des Projekts ist der Aufbau einer Webanwendung in der Programmiersprache Java mit dem Open Source Framework Springboot. Die Umsetzung des Frontend wird durch HTML5 und JavaScript realisiert sowie CSS zur Gestaltung und Formatierung der Webpage realisiert.

Die Umsetzung des Frontend wird nach dem Model View Controller Architekturmuster aufgebaut und beinhaltet die Controller zwecks Mapping der Frontendanfragen (requests), sowie die Service zur Bearbeitung und die Modelle zum Speichern und Abbilden der Daten auf den Webserver nachdem REST-Prinzip.

1.2 Probleme und Herausforderungen

Die Umsetzung des Musik Editors erfordert eine genaue Planung des Funktionsprinzips mit klaren Anforderungen an den Funktionsumfang. Dazu wurden zuerst Anforderungen gesammelt und festgehalten. Der Umsetzung geht die Wöchentliche Projektplanung voraus und bestimmt die Aufgaben Verteilung sowie den Aufgabenzeitraum (??).

Während der Entwicklungsphase im Zeitraum zwischen dem 09.01.2023 – 05.03.2023 kam es zum Ausstieg einiger Gruppenmitglieder des Entwicklungsteams wodurch der Funktionsumfang des Projekts dementsprechend angepasst werden musste.

Durch das nun kleinere Team wurde beschlossen das der Funktionsumfang verkleinert wird, und der Fokus auf den Aufbau der Rest Schnittstelle nach dem MVC-Prinzip gesetzt wird.

1.3 Tools und Frameworks

IDE/Projekt: Visual Studio Code mit Maven als Package Manager.

Frontend: HTML5, JavaScript, CSS.

Backend: Java

- Springboot Framework
- Postman

2 Anforderungen und Funktionsumfang

Zu den Anforderungen an den Musik Editor gehören das Anlegen von Projekten mit einem Projektnamen sowie Autornamen, und die Erstellung von Tracks, welche dem jeweiligen Projekt zugeordnet sind.

Ein Projekt kann einen oder mehrere Tracks beinhalten, und es dem Nutzer erlauben diese zu benennen sowie, die Auswahl des Instruments und die Note (Ton) zu ermöglichen.

Jeder Track ermöglicht es dem Nutzer auszuwählen, wann ein Ton abgespielt wird oder ausgesetzt, wodurch das Zusammenspiel und Parallele abspielen von Tracks gewährleistet werden soll.

Projekte können gespeichert, und später durch den Namen wieder aufgerufen und geladen werden.

3 Projektplanung

1. Projektwoche:

Die erste Projektwoche begann durch das Kick-Off Meeting. Die Aufgaben lagen darin zu Recherchieren, um einen Überblick zu erhalten, welche Tools und Bibliotheken zur Umsetzung verfügbar sind. Des Weiteren wurden Naming Conventions also Programmierregelungen für das ganze Projekt festgehalten. Zum Schluss wurden Stärken und Schwächen der Gruppen Mitglieder im Hinblick auf die Programmiertechnische Umsetzung ermittelt um dies für das weitere Vorgehen zu berücksichtigen.

2. Projektwoche:

Beinhaltete das Erstellen einer Git Repository und Anlegen eines Spring Projekts mit VSCode und Maven als Package Manager mit anschließendem Push auf den Main Branch der Git Repository. Des Weiteren wurde für die Lokale Versionskontrolle für alle Mitglieder das GitHub Desktop Tool festgelegt, sodass Git befehle und das lokale Repository Management vereinfacht werden soll.

Die Aufgaben aus Woche 1 wurden in Woche zwei zum Abschluss gebracht wodurch die Recherche, welche unseres Projekts zugrunde liegt zu Ende gebracht wurde und das Regelwerk (Naming Conventions) vervollständigt.

3. Projektwoche:

Wurden Skizzen für das Frontend angefertigt um ein Abstraktes Modell für die Webpage (UI) bilden. Für das Backend wurden grobe Klassenskizzen in Form eines Klassendiagramm für die REST-Schnittstelle nach dem MVC Prinzip angefertigt.

4. Projektwoche:

Wurden die Diagramme besprochen und abschließend vervollständigt, sodass die Implementierungsphase beginnen konnte. Des Weiteren wurden offene Fragen zu den Aufgaben in der Gruppe diskutiert. Am Ende der 4ten Projektwoche stieg ein Gruppenmitglied aus, welches für die Frontend Entwicklung zuständig war, wodurch die Anpassung und Verteilung der Aufgaben auf die anderen Gruppenmitglieder erfolgen musste.

5. Projektwoche:

Fertigstellung der Landingpage sowie die Weiterentwicklung des Projekt Controllers im Backend (In & Output Mapping). Außerdem wurde die Implementation an der Methode zur Erweiterung zum Lesen / Einbinden von (-/Eigenen) Sounds begonnen.

6. Projektwoche:

Mit dem Abschluss der 6ten Projektwoche ist die Fertigstellung des Controller Mappings sowie Frontend (Visualizing) für Töne abgeschlossen. Das Einbinden von Sounds und lesen ist fertig. (Noch allgemein gehalten, muss noch angepasst werden)

7. Projektwoche:

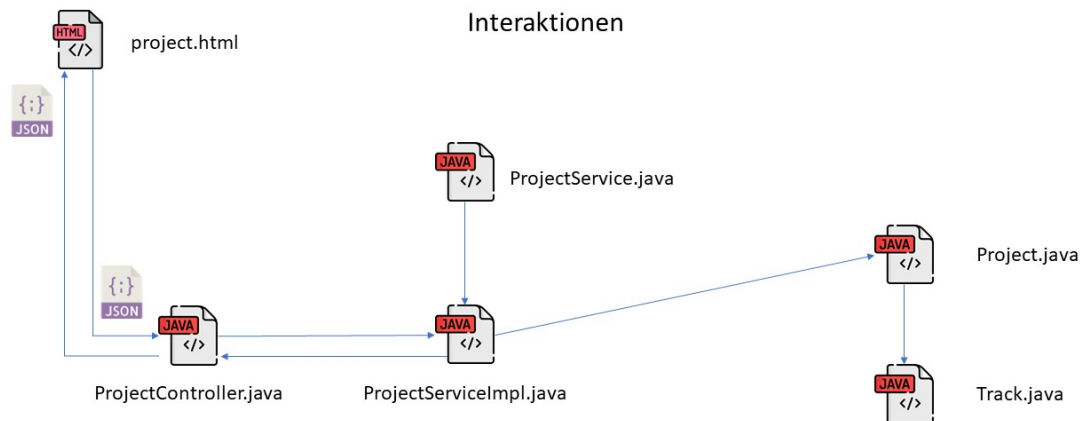
In der 7. Projektwoche ist leider einiges schiefgelaufen. Im Team hat sich eine Spaltung bemerkbar gemacht. Insbesondere das Frontend-Team hat kaum etwas zur Entwicklung beigetragen, obwohl es bei den Teammeetings anders kommuniziert wurde. Es wurde behauptet, dass das Frontend Team seine Aufgaben erledigen würde. Das Backend-Team hat aufgrund dieser Aussagen geglaubt, dass beim All-in-One-Test alles reibungslos laufen würde. Davor wurde lediglich nur über die Teammeetings kommuniziert und es wurden JSON-Dateien vom Backend hinterlegt. Leider hat sich dann ein weiteres Teammitglied (Jean Gabriel Hannia) entschieden die Gruppe ohne Angaben und Rückmeldung zu verlassen und es mussten einige unvorhergesehene Probleme gelöst werden. Somit wurden die Rollen neu verteilt und die Anforderungen der neuen Teamgröße angepasst.

8. Projektwoche:

Des Weiteren entnahm Dinara Kurmanbek einen Teil ihres Aufgabenbereichs aus dem Internet, der nicht den Anforderungen für unsere Projektabgabe entsprach. Das von ihr heruntergeladene Frontend war nicht mit unserem Backend kompatibel und hatte somit keine Schnittstelle. Wir waren uns nach Rücksprache alle einig, dass wir unsere Leistungen eigenständig erbringen möchten. Um diese Herausforderungen zu lösen, haben wir beschlossen, das Projekt im Front Backend neu aufzusetzen und entsprechend anzupassen. Mert und Aryan übernahmen die Leitung des Projekts, um einen erfolgreichen Abschluss zu gewährleisten. Trotzdem ergaben sich bis zum Ende noch Probleme. Wir haben versucht nach bester Möglichkeit die neuen Anforderungen umzusetzen.

4 Umsetzung

Interaktionen:



Controller:

ProjectController.java:

Ist die Zentrale Schnittstelle für das Handling der GET/POST Requests aus dem Frontend zuständig. Empfängt von Frontend JSON Daten und reicht sie weiter durch den Service um sie auf unser Modell mappen.

Folgende Mapping sind enthalten:

```
@GetMapping("/project")
public String loadProject(Model model)
```

Liefert die Töne aus dem Backend basierend auf der Tonbelegung im Frontend.

```
@GetMapping("/test")
public String getSoundfiles(Model model)
```

Testmethode zum Abrufen von Soundfiles aus dem Backend.

```
@PostMapping("/project")
public ResponseEntity<String> createProject(@RequestBody Map<String,
Object> jsonMap)
```

Liefert die Töne aus dem Frontend basierend auf der JSONan das Backend.

Service:

ProjectService.java:

```
public List<Project> localStore = new ArrayList<>();
List<Project> showAllProject();
Project getLocalProjByName(String string);
public void saveLocal(Project project);
```

Ist eine Interface Klasse, welche den Aufbau der ProjektServiceImpl.class beschreibt.

Anmerkung: Es sind weitere ungenutzte Methoden für Datenbanken vorhanden, welche aufgrund der Umstände leer blieben.

```
(void checkProject(Project project);
(void downloadProject(Project project); ...
```

ProjectServiceImpl.java:

Die Service Klasse welche Operationen an den Modellen Project & Track Klassen durchführt, um Daten in Form vom Klasseninstanzen aufzunehmen.

Folgende Methoden sind enthalten:

```
public void saveLocal(Project project){
    localStore.add(project);
}
```

Speichert die Empfangene JSON als Project Objekt in einer Lokalen Liste `List<Project>localStore` vom Typ Project ab.

```
public Project getLocalProjByName(String name){

    for (Project project : localStore){
        if(project.getProjectName().equals(name)){
            return project;
        }
    }
    return null;
}
```

Lädt aus der `localStore` Liste das jeweilige Projekt raus mit demselben Namen und liefert sie dem Controller als Typ `Project` zurück.

```
public String[][] openProject(Project project) {

    int n = project.getTracks().get(0).getTrackList().size();
```



```

        int m= project.getTracks().size();
        String[][] arrayBox = new String[n][m];

        for (int i=0; i<n; i++){
            for (int j=0; i<m; i++){
                String a="";
                if (project.getTracks().get(i).getTrackList().get(j) ==
1){
                    a = "sound"+"/"+ project.getTracks().get(i).getInstrument-
Name()+ "/" + project.getTracks().get(i).getNote() + ".mp3";
                }
                else {
                    a= "";
                }
                arrayBox[i][j] = a;
            }
        }
        System.out.println((String)arrayBox [0][0]);
        return arrayBox;
    }

```

Ruft das jeweilige Objekt Projekt auf und hinterlegt mehrere Pfade in einer Matrix für sodass die Matrix im Controller wieder an das Frontend übergeben werden kann. Dabei werden

Modelle:

Project.java:

Ist die Abbildung des Projekts im Frontend und hält alle notwendigen Daten Felder bereit, um bei Speicherung einer JSON die Datentypen abzubilden. Die Belegung der Variablen wird standardisiert durch Getter/Setter Methoden durch einen Konstruktor durchgeführt. Daten des Modells:

```

private static long id;
private String projectName;
private long authorId;
private String authorName;
private List<Track> tracks = new ArrayList<Track>();

```

Wichtig ist hier die `List<Project>tracks` welche mehrere Tracks vom Typ `Track` hält. Diese wird im Konstruktor beim Erstellen des Objects befüllt durch das Mappen der Attribute und Iterativem einlesen durch eine for schleife. Somit erstellt der Konstruktor der `Project.java` im Grunde genommen automatisch basierend auf der Liste an Tracks, die ihm übergeben wurden `Track` Objekte, welche er dann einliest.

Track.java:

Ist ein Modell welches alle Track Attribute aus dem Frontend, im Backend abbildet.

```
private static int id = 0;  
private boolean active;  
private String trackName;  
private String instrumentName;  
private String note;  
private String path;
```

Dieses Modell dient dazu im Objekt **Project**, **Track** Objekte abbilden zu können, um diese in einer Liste abzuspeichern. Die Belegung der Variablen wird standardisiert durch Getter/Setter Methoden durchgeführt.

Eidesstaatliche Erklärung

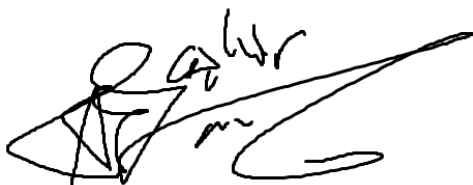
Hiermit versichern wir, dass wir die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der erlaubten Mittel angefertigt haben, des Weiteren hat die Arbeit in gleicher Form noch keiner anderen Prüfbehörde vorgelegen.

Frankfurt am Main, den 05.03.2023

Mert Uzeken

A handwritten signature in black ink, appearing to be 'Mert U.' with a stylized flourish.

Aryan Bashir

A handwritten signature in black ink, appearing to be 'Aryan Bashir' with a large, sweeping flourish.

Dinara Kurmanbek

A handwritten signature in black ink, appearing to be 'Dinara' with a stylized flourish.