Dokumentation M133

Grade Manager

von Lazar Petrovic und Davide Marcoli

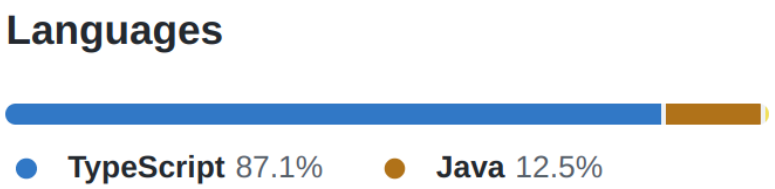
Planung

Technologien

Dank all unseren absolvierten überbetrieblichen Kursen haben wir eine breite Auswahl an Technologien die wir verwenden können für diese Modularbeit. Im Backend lernten wir Spring Boot und für das Frontend lernten wir React native kennen.

Da wir beide diese Technologien vor kurzem im üK vertieft und intensiv angewendet haben, entschieden wir uns diesen Weg zu gehen. Auch aus privaten Projekten oder aus unseren Firmen kennen wir andere Technologien, wie Angular, was auch eine Option war. Der Grund für React Native war, dass wir die Applikation einmal entwickeln dürfen und dann für alle Plattformen exportieren können.

Für den Data Layer entschieden wir uns auch für eine Technologie, welche wir in einem üK benutzt haben. Wir entschieden uns für MySQL, eine relationale Datenbank.



Hier als Beweis unsere GitHub Analyze der Startseite unseren Repos, dass wir uns schlussendlich auch an unseren Plan gehalten haben. Zu erkennen ist, dass knapp 90% TS Code ist, welcher aus dem Frontend kommt. Das Backend wird mit Java und nur 12.5% gemessen.

Use Cases

Registrierung

Use Case	Registrierung
Use Case ID	1
Beschreibung	Der Benutzer registriert sich
Precondition	-
Actor (Primary)	Benutzer
Actor (Secondary)	kein
Procedure: 1. Der Benutzer navigiert zur Registrierungsseite 2. Der Benutzer füllt die Felder Name, E-Mail und Passwort. 3. Optional: Der Benutzer lädt ein Profilbild hoch 4. WENN alle Felder valid sind a. Sende den Registrierungs Request	
Postcondition:	keine
Alternative flows: kein	

Login

Use Case	Login
Use Case ID	2
Beschreibung	Der Benutzer loggt sich ein
Precondition	-
Actor (Primary)	Benutzer
Actor (Secondary)	
Procedure: 1. Der Benutzer navigiert zur Loginseite 2. Der Benutzer füllt die Felder E-Mail und Passwort. 3. WENN alle Felder valid sind und Benutzername / Passwort Kombination existiert a. Logge den User ein	
Postcondition:	
Alternative flows: kein	

Note hinzufügen

Use Case	Note hinzufügen
Use Case ID	3
Beschreibung	Der Benutzer fügt eine neue französisch Note hinzu
Precondition	Keine Noten gespeichert
Actor (Primary)	User
Actor (Secondary)	
Procedure: 1. Der Benutzer navigiert zur Noten Verfassungs Seite 2. Der Benutzer füllt die Felder Note, Schule, Fach und Gewichtung. 3. WENN alle Felder valid sind a. Logge den User ein	
Postcondition:	
Alternative flows: kein	

Note löschen

Use Case	Note löschen
Use Case ID	4
Beschreibung	Der Benutzer löscht eine bestehende französisch Note
Precondition	Keine Noten gespeichert
Actor (Primary)	User
Actor (Secondary)	
Procedure: 1. Der Benutzer navigiert zur Noten Löschen Seite 2. Der Benutzer wählt die Note aus, die er löschen möchte 3. WENN alle Felder valid sind a. Löschen die Note	
Postcondition:	
Alternative flows: kein	

Note bearbeiten

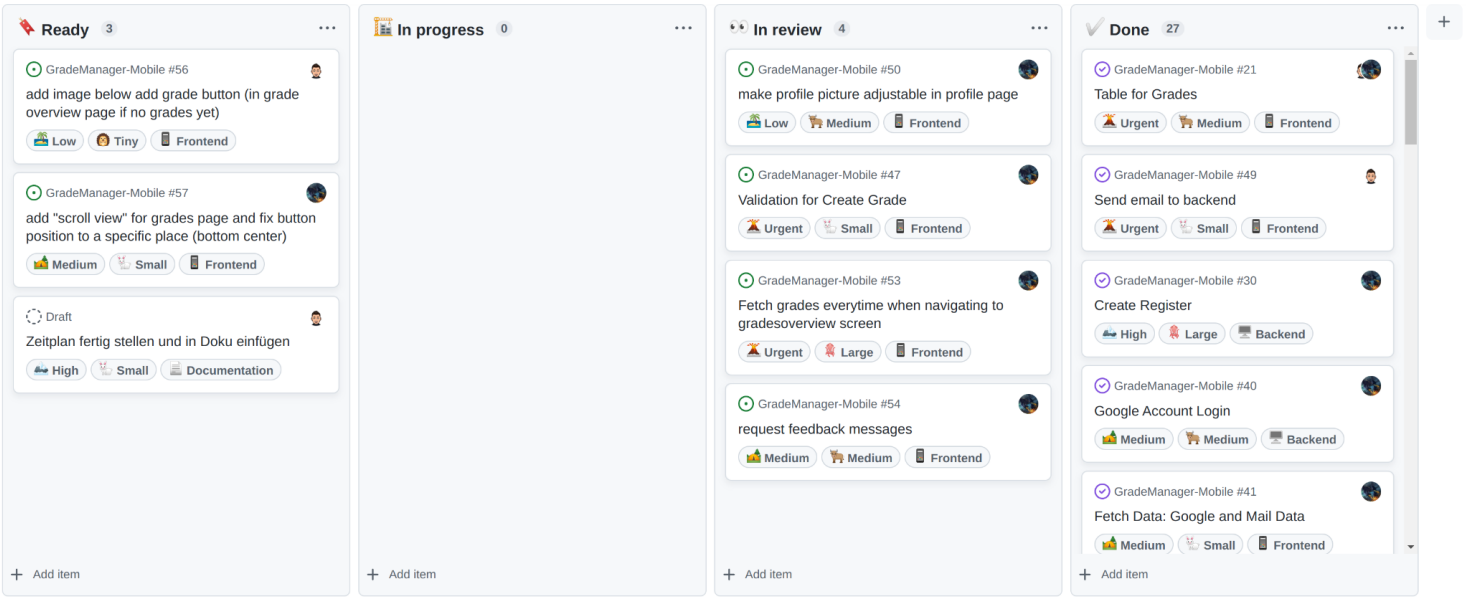
Use Case	Note bearbeiten
Use Case ID	5
Beschreibung	Der Benutzer bearbeitet eine bestehende französisch Note
Precondition	Keine Noten gespeichert
Actor (Primary)	User
Actor (Secondary)	
Procedure: 4. Der Benutzer navigiert zur Noten Bearbeitungsseite 5. Der Benutzer wählt die Note aus, die er bearbeiten möchte 6. Der Benutzer füllt die Felder Note, Schule, Fach und Gewichtung. 7. WENN alle Felder valid sind a. Update die Note	
Postcondition:	
Alternative flows: kein	

Aufteilung der Aufgaben

GitHub Projects

Um uns den Prozess der Implementation unserer 3-Tier Applikation zeitgemäss und so spassig wie möglich gestalten können, entschieden wir uns grossen Wert in die Planung zu legen. In früheren Projekten benutzten wir auch schon GitHub Projects und sammelten enorm gute Erfahrungen dabei.

In unserem *GitHub Repo werden Issues durch GitHub Projects in einem Kanban Board graphisch angezeigt, sodass wir in der Entwicklerrolle eine direkte Übersicht haben über den Entwicklungsstand der Applikation.



Zu Beginn erstellten wir die notwendigsten Aufgaben als Issues in der Ready Spalte und klärten, auf wer mit welcher Task beginnt. Der grösste Pluspunkt für uns Entwickler ist, dass jeder wirklich selbstständig arbeiten kann und sich selbst eine Aufgabe zuweisen kann, auf diese er gerade Lust hat. Ist diese Aufgabe erledigt wird sie unter “In review” geschoben, sodass die andere Person aus der Gruppe sich die Implementation anschauen kann. Dadurch haben wir wiederum die Möglichkeit von uns gegenseitig zu lernen, was wir auch sehr gut ausnutzen konnten schlussendlich.

*GitHub Repo: https://github.com/davidemarcoli/GradeManager-Mobile

Architektur / Design

Frontend-Architektur

Wir arbeiten mit dem Framework React Native, dass uns eine plattformübergreifende App kompilieren kann. Wir, die Entwickler, müssen den Code nur einmal schreiben und es funktioniert auf allen Plattformen.

Models

Bei unserer Applikation sind die Klassen in einem Ordner namens “model”. Wir haben dort 2 Klassen, die Grade-Klasse und die User-Klasse. Diese haben alle Attribute, die unsere Applikation braucht.

Grade:

```
id: string
name: string
grade: number
subject: string
school: string
user: User
```

User:

```
id: string
email: string
password: string
name: string
profilePictureUrl: string
```

Services

In den Services werden Abläufe, die nicht direkt etwas mit dem UI zu tun haben, erledigt. Wir haben in unserem Service-Ordner 3 dieser Services erstellt:

Grade Service

In diesem Service werden alle Informationen, die mit den Noten zu tun haben, mit dem Backend verknüpft. Dieser Service hat Methoden, um Requests an unser Spring Boot Backend zu senden. Es wird mit der fetch() Methode von JavaScript gearbeitet. Das Objekt, dass mitgeschickt wird, muss jeweils noch zu einem JSON-String konvertiert werden, da man nur Strings als Informationen mitschicken kann.

Theme Storage Service

Der Storage Service ist für die lokale Speicherung der Theme-Auswahl zuständig. Er kann das im Moment ausgewählte Theme speichern, ersetzen und es wieder laden. Er arbeitet mit den asynchronen Funktionen einer Community-Implementation von AsyncStorage. Das heisst, die Applikation funktioniert weiterhin, ohne auf das Speichern zu warten.

User Service

Der User Service ist in 2 Teile aufgeteilt. Auf der einen Seite, enthält er Methoden, um die Registrierung und das Login durchzuführen. Andererseits gibt es aber auch Methoden hier drin, die für das Speichern des Logins zuständig sind, sodass sich der Benutzer nicht immer auf's Neue anmelden muss. Immer wenn der User auf die Login Seite kommt, wird in dieser Klasse überprüft, ob schon ein User im oben erwähnten LocalStorage gespeichert wird und wenn ja, wird versucht sich mit diesem Benutzer einzuloggen. Davon würde der Benutzer so gut wie nichts mitbekommen. Falls es bei diesem automatischen Login jedoch zu Fehlern kommt, wird der Benutzer auf die Login-Seite geleitet, wo er sich anmelden muss.

Atoms

Atoms bilden die Grundlage des Atom-Designs und sind die simpelsten Komponente. Beispiele von Atoms können ein einfaches Input Field oder ein Text sein.

Molecules

Molecules sind eine Ansammlung von Atoms. Sie kombinieren diese, um etwas grösseres zu bilden. Man könnt die oben erwähnten Atoms zu einem Molecule kombinieren, um ein Input Field mit einem Label zu bekommen. Die gleichen Atoms können in verschiedenen Molecules verwendet werden.

Organisms

Organismen bilden die grösste Art der alleinstehenden Komponente. Es sind Kollektionen von Molecules und Atoms.

Screens

Screens (oder auch Pages genannt) sind verschiedene Seiten der Applikation.

Backend-Architektur

Im Backend arbeiten wir mit dem Spring Framework, was das Erstellen von Backend-Applikationen mit Vereinfachungen, wie simpel zu konfigurierende REST-Endpoints oder automatische DDL-Generierung mit Hibernate vereinfacht.

Entitäten

Im Backend gibt es für jede Tabelle in der Datenbank eine Entität, die mit einer Java-Klasse dargestellt wird. Diese haben folgende wichtige Eigenschaften:

```
@Entity (name = "grades")
@Setter
@Getter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class Grade {
   @Id
   private String id;
   private String name;
   private double grade;
   private String subject;
   private String school;
   @ManyToOne
   @JoinColumn(name = "user_id")
   private User user;
}
```

Mit @Entity wird angegeben, dass für diese Klasse eine Datenbanktabelle erstellt werden sollte. Man kann ausserdem noch den Name dieser Tabelle angeben, falls man nicht möchte, dass er von Hibernate selbst generiert wird.
Primary Keys werden mit der @Id Annotation gekennzeichnet. Diese Column wird dann unique sein und als Identifier gebraucht.
Mit @ManyToOne und @JoinColumn kann man Foreign Keys erstellen. Dieser Foreign Key liest sich wie folgt: “One User kann Many Grades haben” und “Many Grades können zu One User gehören”.

Controller

REST-Endpoints können mittels Controllern erstellt werden. So kann man von aussen mittels Requests Daten von der Applikation bekommen oder diese hinzufügen und verändern. Sie sind die Schnittstelle zwischen der Spring Boot Applikation und des Internets.

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/grades")
public class GradeController {

    private final GradeService gradeService;

    @GetMapping("/user/{id}")
    public List<Grade> getGradesByUserID(@PathVariable("id") String userId) {
        return gradeService.getGradesByUserID(userId);
    }

    @PostMapping("/persistence/addgrade")
    public Grade saveGrade(@RequestBody Grade grade) throws LoginException {
        return gradeService.saveGrade(grade);
    }

    @DeleteMapping("/delete/{id}")
    public void deleteGradeByID(@PathVariable("id") String gradeId) {
        gradeService.deleteGradeByID(gradeId);
    }

    @PutMapping("/update/{id}")
    public Grade updateGradeByID(
            @PathVariable("id") String gradeId, @RequestBody Grade updatedGrade
    ) {
        return gradeService.updateGradeByID(gradeId, updatedGrade);
    }
}
```

Das ist ein Teils unseres Grade-Controllers. Auch hier wird viel mit Annotationen gearbeitet. Eine wichtige ist dabei @RestController: Diese sagt Spring, dass in dieser Klasse Methoden kommen, die REST-Endpoints darstellen können. Um diese aber zu erstellen, braucht es die Mapping Annotationen, die die Route zu der Methode anzeigen. Bei Request Mappings sind alle Request Types erlaubt. Falls man nur gewisse, wie zum Beispiel “GET” zulassen möchte, kann man die spezifischen Annotationen gebrauchen, die folgendermassen aufgebaut sind: ${Request-Type}-Mapping.

Services

In den Services werden die Daten, die durch die Endpoints zur Applikation gelangen, verarbeitet. Es werden Sachen ausgerechnet, bearbeitet oder neu erstellt. Es besteht meistens aus einem Interface und der dazugehörigen Implementation.

```
public interface GradeService {
   List<Grade> getAllGrades();

   Grade getGradeByID(String id);

   Grade saveGrade(Grade grade);

   List<Grade> getGradesByUserID(String userId);

   void deleteGradeByID(String gradeId);

   Grade updateGradeByID(String gradeId, Grade newGrade);
}
```

```
@Service
public class GradeServiceImpl implements GradeService {

   @Autowired
   private GradeRepository gradeRepository;

   @Override
   public Grade saveGrade(Grade grade) {
       return gradeRepository.save(grade);
   }

   @Override
   public void deleteGradeByID(String gradeId) {
       gradeRepository.deleteById(gradeId);
   }
```

Hier ein kleiner Ausschnitt unseres Grade-Services. Wie man, werden dort Daten des Controllers entgegengenommen und es wird mit dem Repository kommuniziert, zu dem wir im nächsten Abschnitt kommen werden. Da man ein Interface hat, sieht es aufgeräumt aus und man verliert nie den Überblick, da man im Interface nachschauen kann, welche Methoden es in dieser Klasse sicher gibt.

Repository

Das Repository ist ein wichtiger Bestandteil der Datenverwaltung, denn mit ihm kann man auf einfache Art mit der Datenbank kommunizieren. Es ist die Schnittstelle zwischen Java-Code und SQL-Querys. Hier werden die Klassen mittels Hibernate in SQL-Entitäten konvertiert, die funktioniert auch in die entgegengesetzte Richtung.

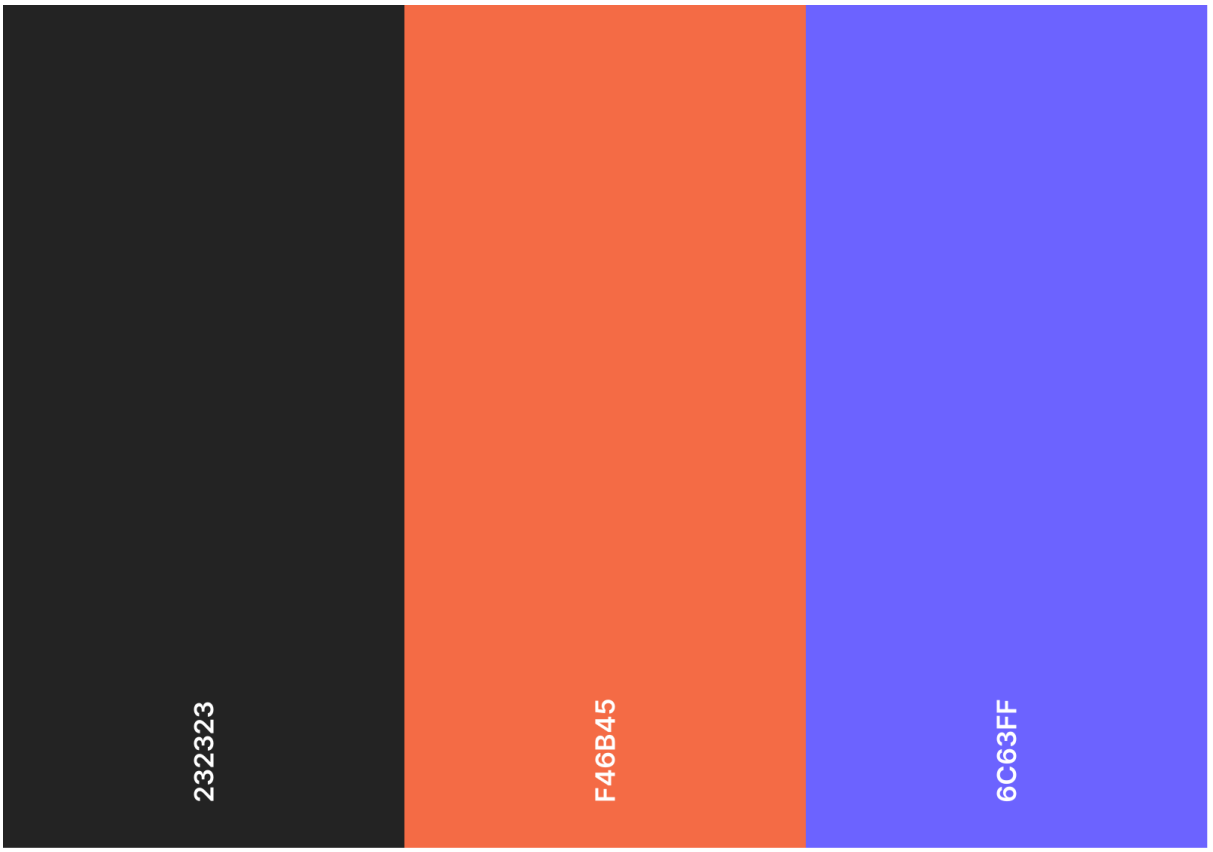
```
@Repository
public interface GradeRepository extends JpaRepository<Grade, String> {
   List<Grade> findByUser_Id(String id);
}
```

Man sieht, dass dieses Interface fast keinen Inhalt hat. Dies ist so, weil wir vom JpaRepository erben und somit schon viele vorgeschriebene Methoden haben. Viel gebrauchte Methoden, wie zum Beispiel findById() oder getAll() sind dort bereits vorhanden und müssen nicht jedes Mal von uns implementiert werden. Auch speziell ist, dass man keine Implementation dieses Interfaces anfertigen muss, da dies intern von Spring erledigt wird. Dies ist sehr praktisch und spart sehr viel Arbeit.

Design

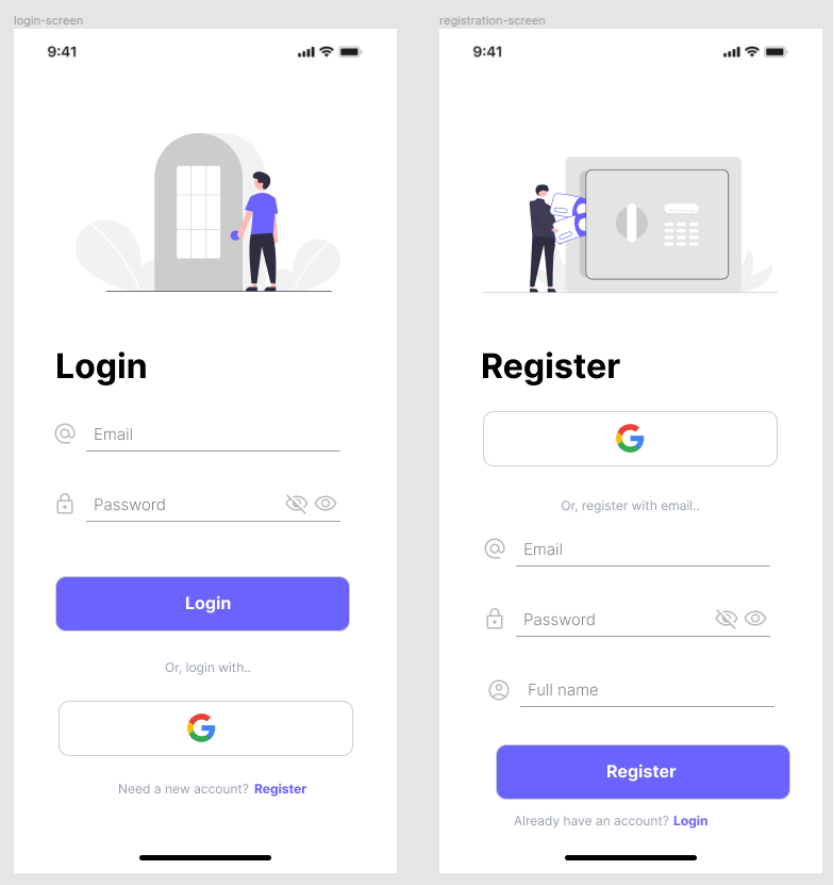
Wir haben die Designrichtlinien von Material Design 3 beachtet.
Lerne unter folgendem Link mehr dazu: https://m3.material.io/

Farbpalette



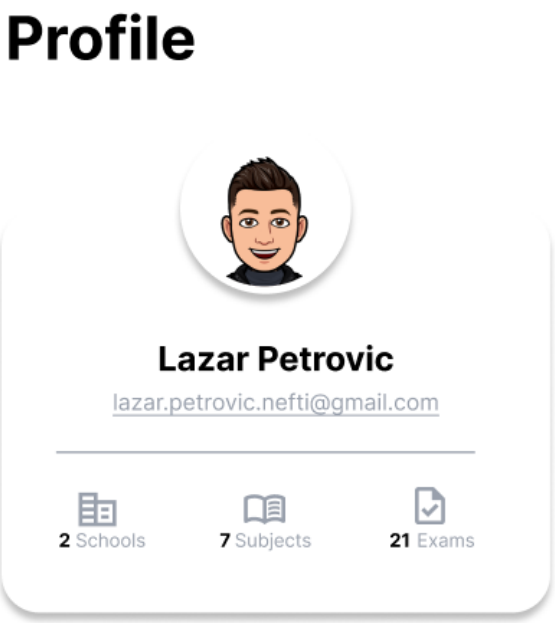
Mockup

Wir haben uns bevor wir mit dem Programmieren angefangen haben, viel Gedanken zum Design gemacht und unsere Überlegungen anhand von einem Mockup ausgedrückt.

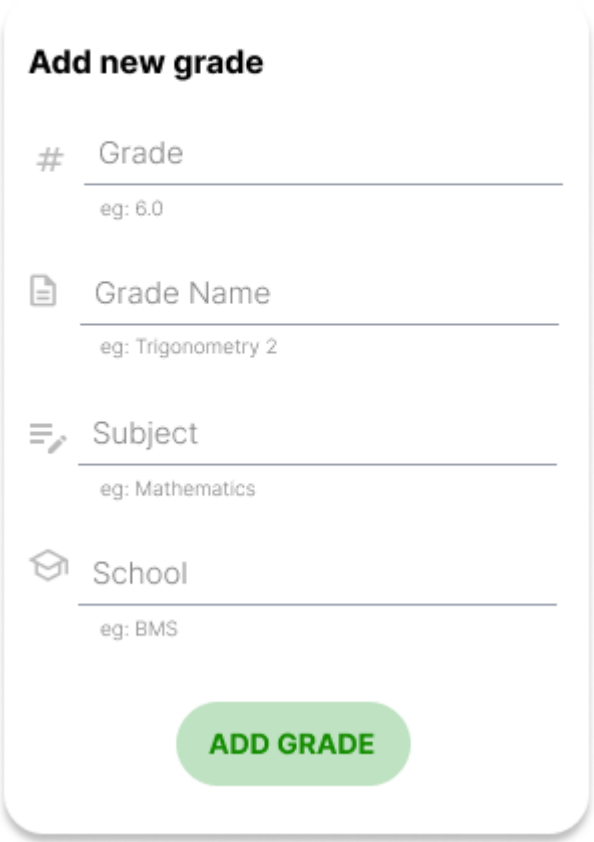


Das sind die Login- und die Register Seite. Es sind die Seiten, die der Benutzer als erstes sieht, wenn er unsere App benutzt. Das heisst, sie sind sehr wichtig, um den User einen ersten Eindruck unseres Designs zu vermitteln.

Wir wollten den Benutzer eine gute Übersicht über seine Aktionen in der App geben und haben darum diesen Profile-Screen designt. Auf diesem sieht der User wichtige Statistiken auf einen Blick, ohne sich in verschiedenen Untermenüs zu verirren.



Grades



Auf dieser Seite kann der Benutzer eine neue Note erstellen. Er hat vier Felder, die er füllen muss, um das Formular abzuschicken.

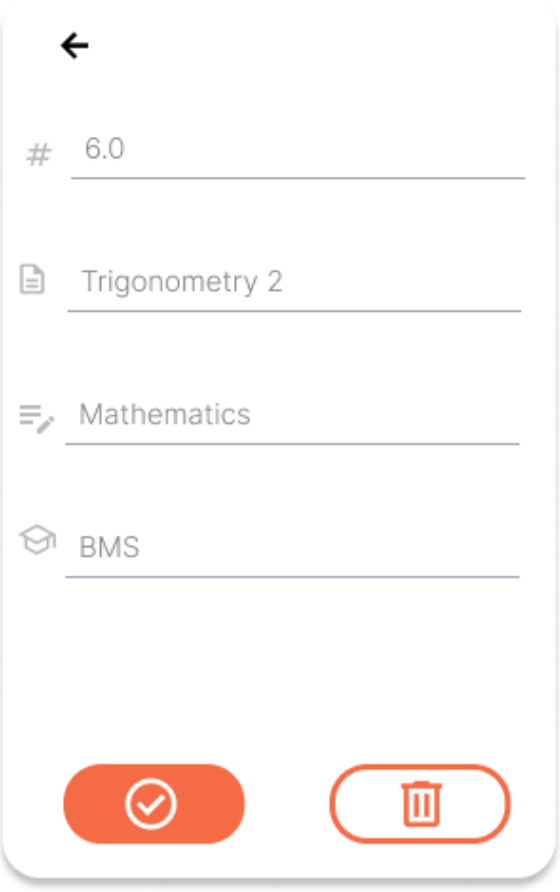
Grade:
Eine Note zwischen 1 und 6. Sie kann auch Kommastellen enthalten.

Grade Name:
Das Thema der Prüfung. Zum Beispiel Trigonometrie 2.

Subject:
Das Fach, in dem die Prüfung stattgefunden hat. Zum Beispiel Mathematik.

School:
Die Schule, in der diese Note geschrieben wurde.

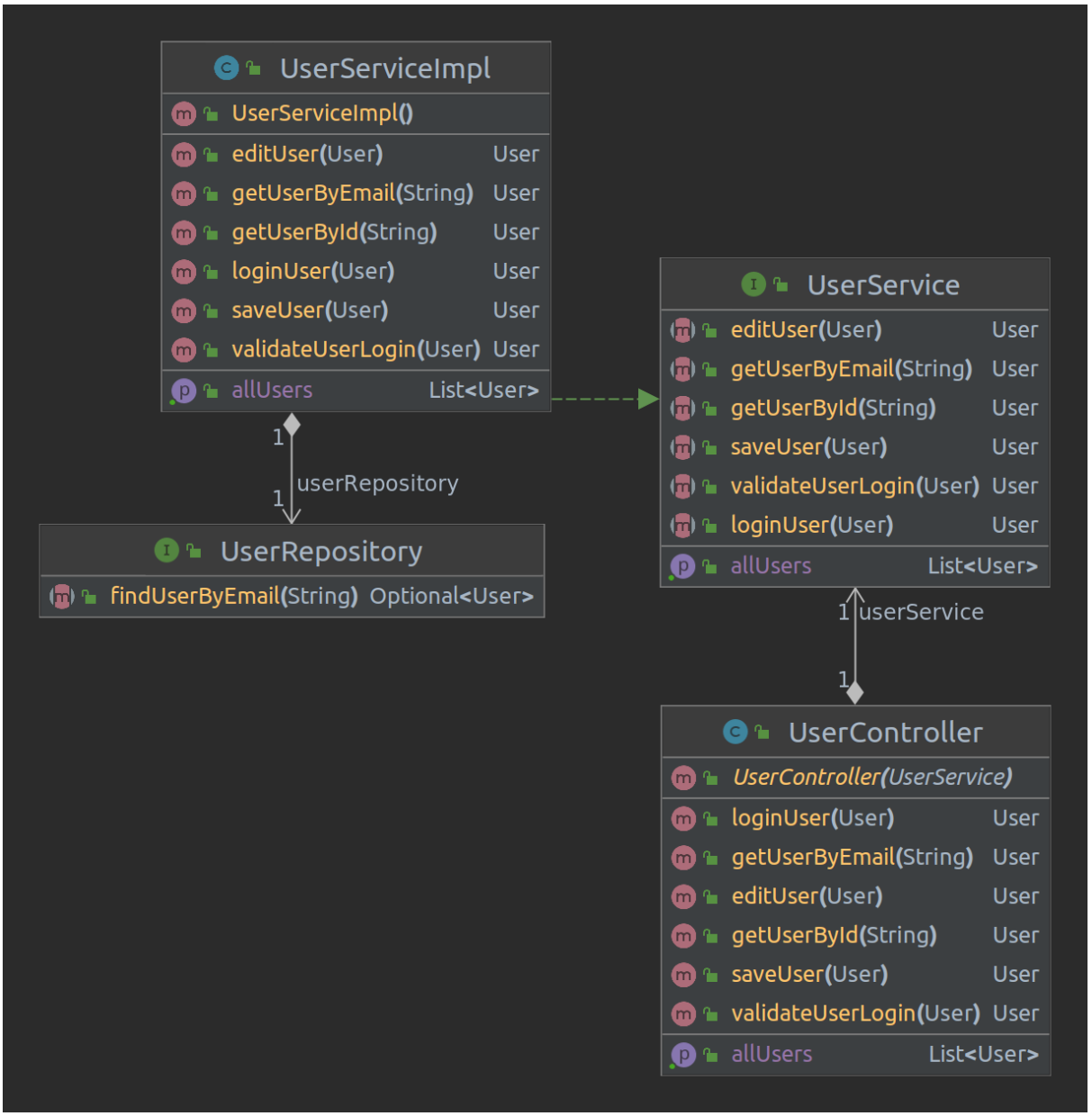
Ausserdem sollte der Benutzer erfasste Noten auch bearbeiten können. Für dies ist dieses Modal zuständig. Die Daten der zu bearbeitenden Note werden automatisch ausgefüllt und der Benutzer kann dann entweder gewisse Felder beliebig anpassen oder mit dem Delete-Button diese Note löschen, falls er sie nicht mehr braucht.



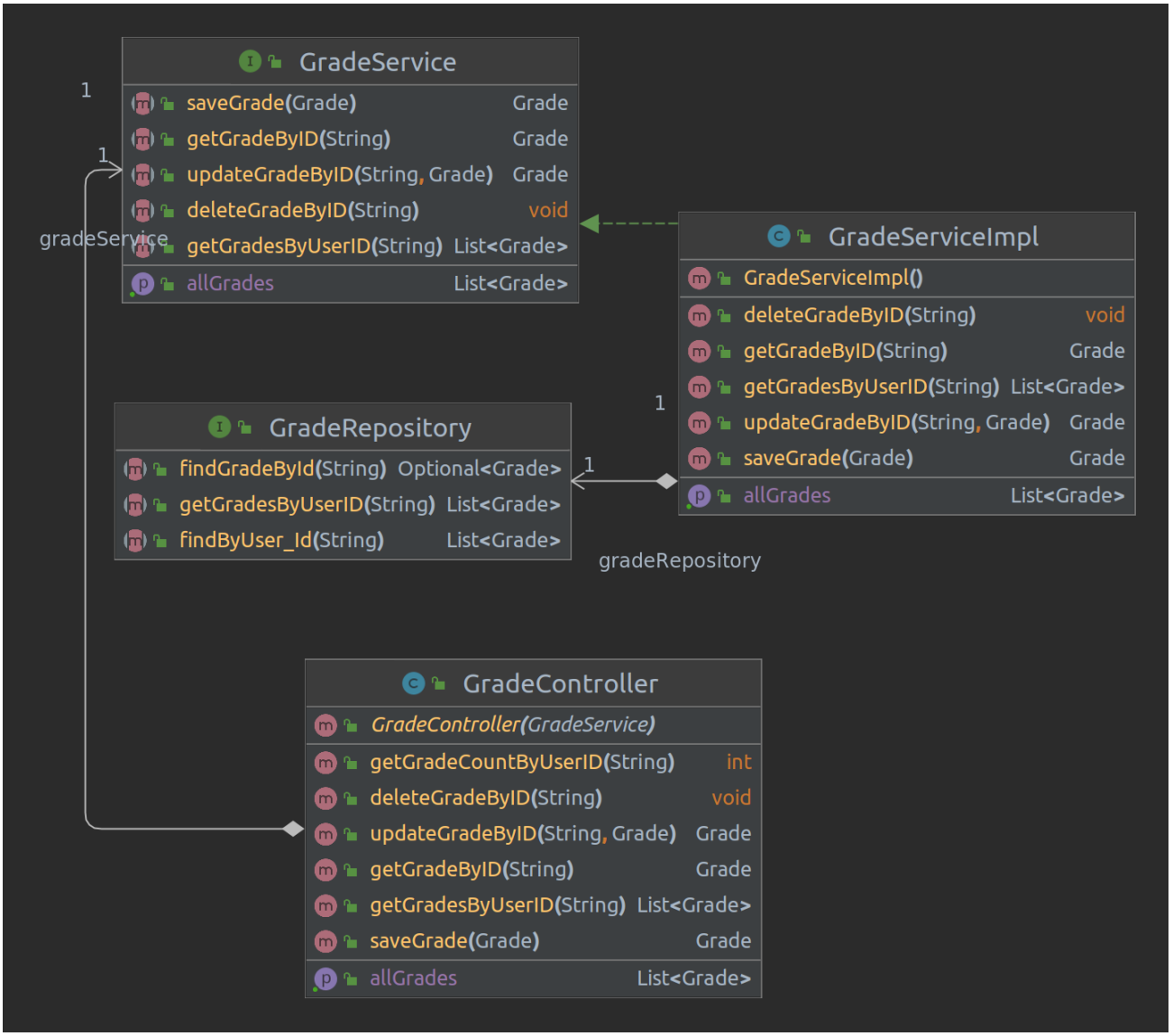
UML

In den zwei Bildern unten sehen wir den typischen Aufbau einer Spring Boot Applikation. Es besteht aus drei Komponenten: Repository, Controller und Service. Im Controller sind die Endpoints unserer API. Dank dem Repository können wir mithilfe von JPA auf unsere Datenbank zugreifen. Im Service implementieren wir die eigentliche Logik die ausgeführt werden soll, wenn ein bestimmter endpoint aufgerufen wird.

User

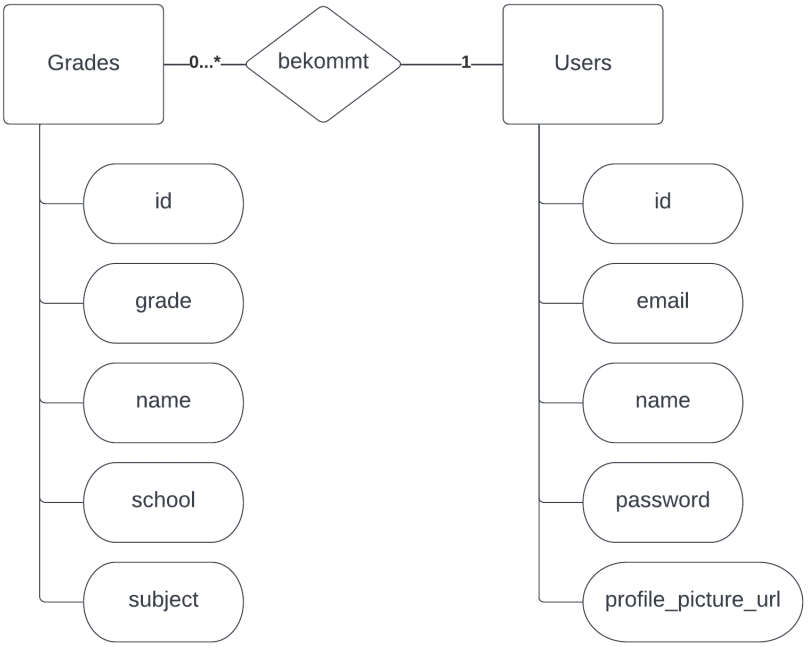


Grade

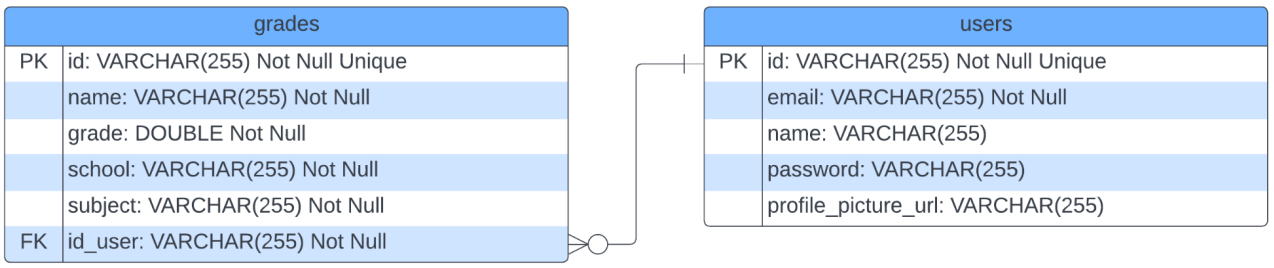

GradeService
saveGrade(Grade) Grade
getGradeByID(String) Grade
updateGradeByID(String, Grade) Grade
deleteGradeByID(String) void
getGradesByUserID(String) List<Grade>
allGrades List<Grade>
GradeServiceImpl
GradeServiceImpl()
deleteGradeByID(String) void
getGradeByID(String) Grade
getGradesByUserID(String) List<Grade>
updateGradeByID(String, Grade) Grade
saveGrade(Grade) Grade
allGrades List<Grade>
GradeRepository
findGradeById(String) Optional<Grade>
getGradesByUserID(String) List<Grade>
findByUser_Id(String) List<Grade>
GradeController
GradeController(GradeService)
getGradeCountByUserID(String) int
deleteGradeByID(String) void
updateGradeByID(String, Grade) Grade
getGradeByID(String) Grade
getGradesByUserID(String) List<Grade>
saveGrade(Grade) Grade
allGrades List<Grade>
1
1
gradeService
1
1
gradeRepository

Database

ERM



ERD



Testing

Umgebung

- Development
    - Ubuntu 22.04
    - 17 Zoll Bildschirm
    - Intellij Idea (Version 2022.1.2) and Visual Studio Code (Version 1.67.1)
    - Android Emulator: Pixel 4 API 32
    - Java Openjdk 18
- Testing
    - Pixel 4 API 32 (virtual)
    - Expo Go (Version 2.24.4)

Test cases

Testfall-Nr.		1		
Testfall-Bezeichnung		Navigieren zwischen verschiedenen Seiten		
User-Story-Nr.		1		
Zu testende Funktionalität		Navigation		
Datum der Testdurchführung		28. Juni 2022		
Tester		Davide Marcoli		
Voraussetzungen		App gestartet und eingeloggt		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	Auf den “Grades” Button in der Navbar klicken	Navigation zur “Grades Overview” Seite	Navigation zur “Grades Overview” Seite	OK

Testfall-Nr.		2		
Testfall-Bezeichnung		Auflistung der Noten		
User-Story-Nr.		2		
Zu testende Funktionalität		Auflistung Noten		
Datum der Testdurchführung		28. Juni 2022		
Tester		Davide Marcoli		
Voraussetzungen		App gestartet und auf der “Grades Overview”		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	keine (siehe Voraussetzungen)	Auf dem Screen sind alle Grades zu sehen.	Auf dem Screen sind alle Grades zu sehen.	OK

Testfall-Nr.		3		
Testfall-Bezeichnung		Neue Note erstellen		
User-Story-Nr.		3		
Zu testende Funktionalität		Neue Note		
Datum der Testdurchführung		28. Juni 2022		
Tester		Davide Marcoli		
Voraussetzungen		App gestartet, eingeloggt und auf der “Grades Overview” Seite		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	Auf den “New Grade” Button klicken	Navigation zur Seite “Add Grade”	Navigation zur Seite “Add Grade”	OK
2	Der Benutzer gibt seine Note, den Notennamen, das Thema und die Schule ein. Der Benutzer klickt auf den “Add Grade” Button	Note wird erstellt.	Note wird erstellt	OK

Testfall-Nr.		4		
Testfall-Bezeichnung		Note bearbeiten		
User-Story-Nr.		4		
Zu testende Funktionalität		Note bearbeiten		
Datum der Testdurchführung		28. Juni 2022		
Tester		Lazar Petrovic		
Voraussetzungen		App gestartet, eingeloggt und auf der “Grades Overview” Seite eine existierende Note anklicken		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	Auf eine existierende Note klicken.	Edit Popup mit vorausgefüllten Input Feldern mit Daten der ausgewählten Note	Popup geht auf und alle Input Felder werden richtig gefüllt.	OK
2	Der Benutzer gibt ändert den Wert der Note von 6 zu 4 und klickt auf den Knopf mit dem Haken als Bestätigung.	Note wird aktualisiert.	Note wird aktualisiert.	OK

Testfall-Nr.		5		
Testfall-Bezeichnung		Note löschen		
User-Story-Nr.		5		
Zu testende Funktionalität		Note löschen		
Datum der Testdurchführung		28. Juni 2022		
Tester		Lazar Petrovic		
Voraussetzungen		App gestartet, eingeloggt und auf der “Grades Overview” Seite eine existierende Note anklicken		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	Auf eine existierende Note klicken.	Edit Popup mit vorausgefüllten Input Feldern mit Daten der ausgewählten Note	Popup geht auf und alle Input Felder werden richtig gefüllt.	OK
2	Der Benutzer klickt auf den “Delete” Button mit dem Mülleimer Icon.	Note wird gelöscht.	Note wird gelöscht.	OK

Testfall-Nr.		6		
Testfall-Bezeichnung		Sich in der App registrieren mit Google		
User-Story-Nr.		6		
Zu testende Funktionalität		Google Registration		
Datum der Testdurchführung		28. Juni 2022		
Tester		Lazar Petrovic		
Voraussetzungen		App gestartet und noch nicht eingeloggt		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	In der Login Page unten auf “register” klicken.	Zur Register Page weitergeleitet.	Zur Register Page weitergeleitet.	OK
2	Auf den Google button klicken	Ein browser Fenster öffnet sich, wo sich der User mit seinem Google Account einloggen kann	Ein browser Fenster öffnet sich, wo sich der User mit seinem Google Account einloggen kann	OK
3	Sich mit den korrekten Google Daten einloggen.	Automatisch zurück zur App navigiert und zur Exams page geleitet.	Automatisch zurück zur App navigiert und zur Exams page geleitet.	OK

Testfall-Nr.		7		
Testfall-Bezeichnung		Sich in die App einloggen mit Google		
User-Story-Nr.		7		
Zu testende Funktionalität		Google Login		
Datum der Testdurchführung		28. Juni 2022		
Tester		Lazar Petrovic		
Voraussetzungen		App gestartet und noch nicht eingeloggt		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	In der Login Page auf den Google Button klicken.	Ein browser Fenster öffnet sich, wo sich der User mit seinem Google Account einloggen kann	Ein browser Fenster öffnet sich, wo sich der User mit seinem Google Account einloggen kann	OK
2	Sich mit den korrekten Google Daten einloggen.	Automatisch zurück zur App navigiert und zur Exams page geleitet.	Automatisch zurück zur App navigiert und zur Exams page geleitet.	OK

Testfall-Nr.		8		
Testfall-Bezeichnung		Sich aus der App ausloggen		
User-Story-Nr.		8		
Zu testende Funktionalität		Logout		
Datum der Testdurchführung		28. Juni 2022		
Tester		Lazar Petrovic		
Voraussetzungen		App gestartet und sich eingeloggt		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	Zur Profile Page navigieren und auf Logout klicken	Zur Login page navigiert und ausgeloggt.	Zur Login page navigiert und ausgeloggt.	OK

Testfall-Nr.		9		
Testfall-Bezeichnung		Von Light zu Dark Mode wechseln		
User-Story-Nr.		9		
Zu testende Funktionalität		App Background Theme ändern		
Datum der Testdurchführung		28. Juni 2022		
Tester		Lazar Petrovic		
Voraussetzungen		App gestartet und sich eingeloggt		
Testschritte				
Nr.	Aktion	Erwartetes Ergebnis	Eingetroffenes Ergebnis	Status
1	Zur Profile Page navigieren und auf das Settings Icon klicken	Zur Settings Page navigiert.	Zur Settings Page navigiert.	OK
2	Auf den Toggle Button “Toggle Dark Mode” klicken.	Wechselt zum anderen Theme.	Wechselt zum anderen Theme.	OK