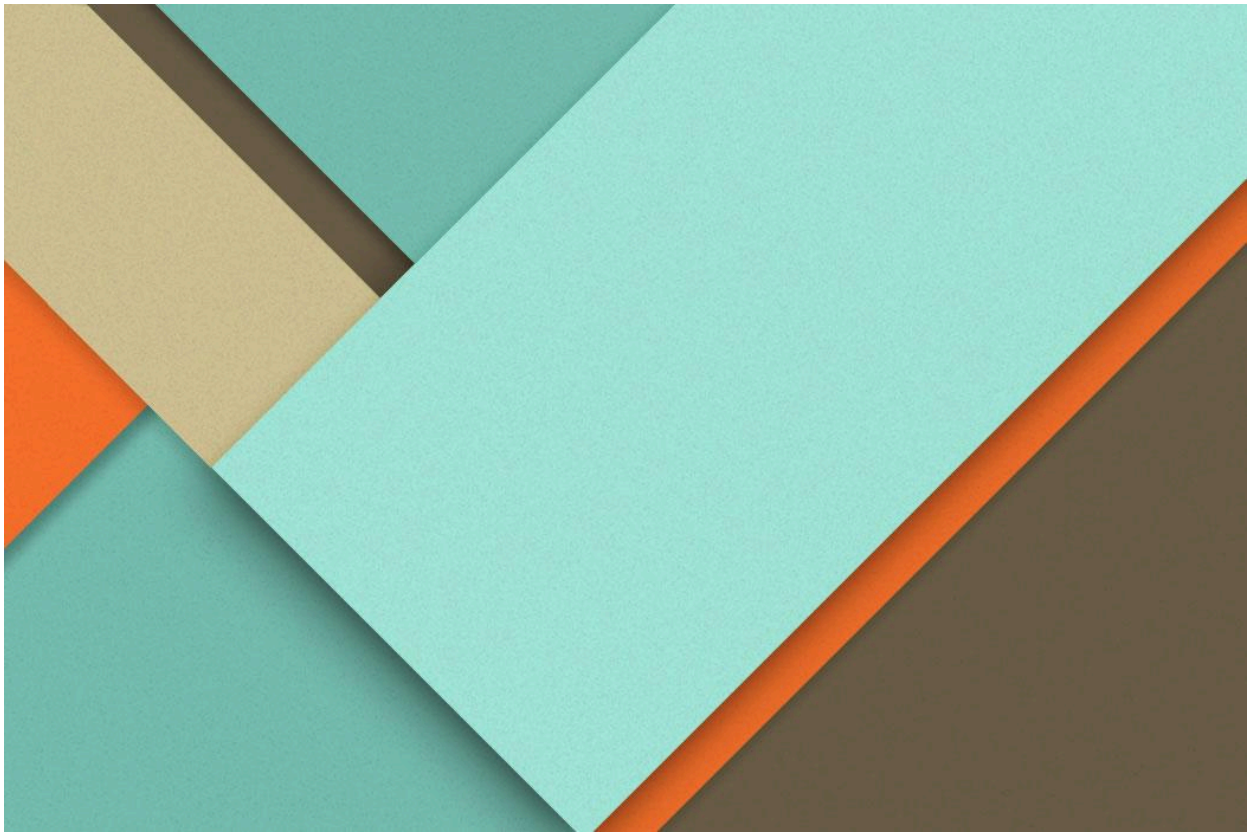


Dokumentation üK 223

21.02.2025

Loris Imbrogno, Marvin Kropf, Cédric Ackermann

Multiuser-Applikationen objektorientiert realisieren

Gruppe 1

Inhaltsverzeichnis

Unser Projekt

Funktionale Anforderungen

User Rollen und Privilegien:

• Bestehende Rollen und Autoritäten werden bearbeitet oder erweitert um untenstehende Anforderungen zu erfüllen und zu testen.

• Die persönlichen Informationen eines Users sind nur für Administratoren oder den User selbst zugänglich.

• Admins können ausserdem andere Benutzer bearbeiten, erstellen und löschen.

Frontend

• Im Minimum enthält die Applikation:

• Login-Page, die öffentlich zugänglich ist (bereits vorhanden)

• Eine öffentlich zugängliche Homepage (bereits vorhanden)

• eine Homepage für alle eingeloggten User

• Eine Admin Page (nur für Admins zugänglich).

• Mindestens eine Komponente, um gruppenspezifische Funktionalitäten im Frontend zu ermöglichen.

Security

• Jeder REST-Endpoint soll nur mit sinnvollen Autoritäten zugänglich sein. Dies wird mit automatisierten Tests überprüft.

• Es gibt Bereiche des Front-Ends die nur für eingeloggte Benutzer zugänglich sind.

• Es gibt Bereiche des Front-Ends die nur für Admins zugänglich sind.

• Der Authentifizierung-Mechanismus wird mit JSON-Web-Tokens implementiert. (bereits vorhanden)

Nicht Funktionale Anforderungen

Implementation

• Daten werden in einer PostgreSQL Datenbank persistiert, das OR-Mapping wird mit JPA realisiert.

• Ein Frontend mit React (Typescript) wird genutzt.

• Ein Backend Springboot (Java) wird genutzt.

• Der Sourcecode wird täglich in einem GIT-Repository commited.

Testing

• Generelle Funktionalität aller selbst implementierten Endpoints wird mit Cypress (zwingend), Postman und/oder JUnit getestet.

• Besonderer Wert wird auf das Testen von Zugriffsberechtigungen gelegt.

• Mindestens ein Use-Case wird ausführlicher mit Cypress getestet. Dies beinhaltet im Minimum:

o Der Endpoint wird mit mehreren Usern & Rollen getestet

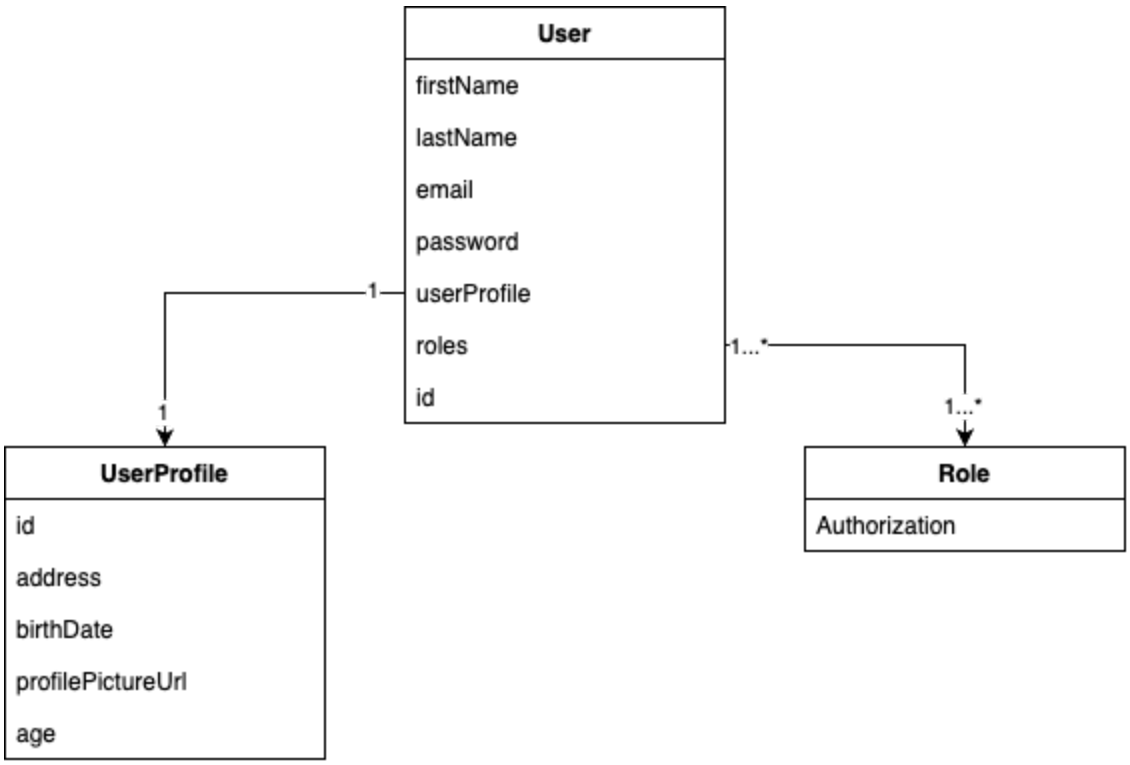
o Mindestens ein Erfolgsfall und ein Errorfall wird getestet.

o Für diese Fälle werden Use-Cases nach UML-Standard beschrieben.

Multiuserfähigkeit

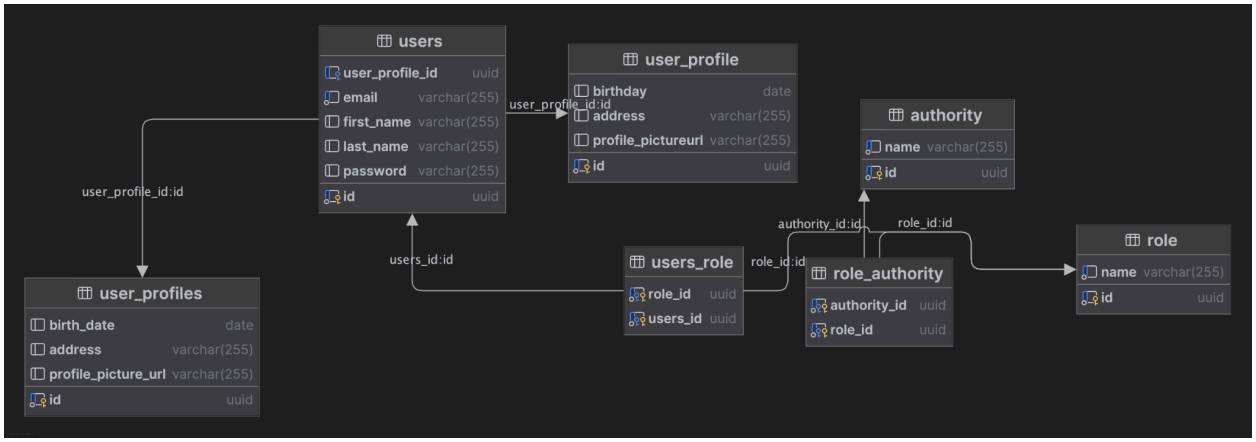
• Aspekte der Multiuserfähigkeit, wie z.B. Einhaltung der ACID-Prinzipien werden berücksichtigt

Domain Model



Das Diagramm zeigt ein Domain Model für ein Benutzersystem mit drei Hauptklassen: User, UserProfile und Role. Ein User verfügt über Attribute wie Vorname, Nachname, E-Mail, Passwort, Rollen und eine eindeutige ID. Es besteht eine 1:1-Beziehung zwischen User und UserProfile, wobei das Profil zusätzliche Informationen wie Adresse, Geburtsdatum, Profilbild-URL und Alter enthält. Zwischen User und Role besteht eine 1:n-Beziehung, die es ermöglicht, dass ein Benutzer mehrere Rollen mit entsprechenden Berechtigungen haben kann. Dieses Modell bildet die Grundlage für die Verwaltung von Benutzerdaten, Profilinformationen und rollenbasierten Berechtigungen.

ERD



Das ERD zeigt die Datenbankstruktur eines Benutzersystems. Die zentrale Tabelle users enthält Felder wie E-Mail, Vorname, Nachname, Passwort und eine Verknüpfung zu einem Benutzerprofil. Es besteht eine 1:1-Beziehung zwischen users und user_profiles, wobei das Profil zusätzliche Informationen wie Geburtsdatum, Adresse und Profilbild-URL speichert. Über die Zwischentabelle users_role wird eine n:m-Beziehung zwischen Benutzern und Rollen hergestellt. Die role-Tabelle definiert Rollen, die über die role_authority-Tabelle mit spezifischen Berechtigungen aus der authority-Tabelle verknüpft sind. Dieses Modell ermöglicht eine flexible Verwaltung von Benutzerprofilen, Rollen und Berechtigungen.

Testing Strategie

I. Cypress

Wir werden Cypress benutzen, um automatisierte Tests für unsere API zu schreiben und zu validieren. Es ermöglicht uns, API-Tests direkt in JavaScript zu schreiben und sicherzustellen, dass unsere Endpunkte wie erwartet funktionieren.

II. Postman Collection

Wir werden eine Postman Collection erstellen, um alle relevanten API-Requests zu dokumentieren und auszuführen. Diese Collection enthält Tests für verschiedene Anwendungsfälle:

- Erstellen eines Benutzerprofils (Post User Profile)
  - Ein korrekt formatierter Request erstellt erfolgreich ein Benutzerprofil.
  - Fehlformatierte Anfragen (z. B. falscher JSON-Body) werden korrekt abgelehnt.
  - Ein Request mit nicht erforderlichen Feldern sollte keine Probleme verursachen.
- Abrufen eines spezifischen Benutzerprofils (Get specific User Profile)
  - Berechtigte Benutzer können ihr eigenes Profil abrufen.
  - Unberechtigte Benutzer dürfen keine fremden Profile einsehen.
  - Admins haben Zugriff auf alle Benutzerprofile.
- Abrufen aller Benutzerprofile (Get all User Profiles)
  - Berechtigte Benutzer können auf alle Benutzerprofile zugreifen.
  - Unberechtigte Benutzer erhalten keinen Zugriff.
  - Das Abrufen mit Sortierungsparametern funktioniert korrekt.
  - Das Abrufen mit Limitierungsparametern liefert die erwartete Anzahl an Ergebnissen.
- Aktualisieren eines Benutzerprofils (Put User Profile)
  - Benutzer dürfen nur ihr eigenes Profil aktualisieren.
  - Admins können jedes Benutzerprofil bearbeiten.
  - Wenn das Benutzerprofil nicht existiert, sollte eine entsprechende Fehlermeldung erscheinen.
  - Fehlformatierte Anfragen oder zusätzliche nicht erforderliche Felder werden korrekt behandelt.
- Löschen eines Benutzerprofils (Delete User Profile)
  - Wenn ein Benutzer gelöscht wird, wird sein User Profile automatisch entfernt.

Erwartetes Verhalten

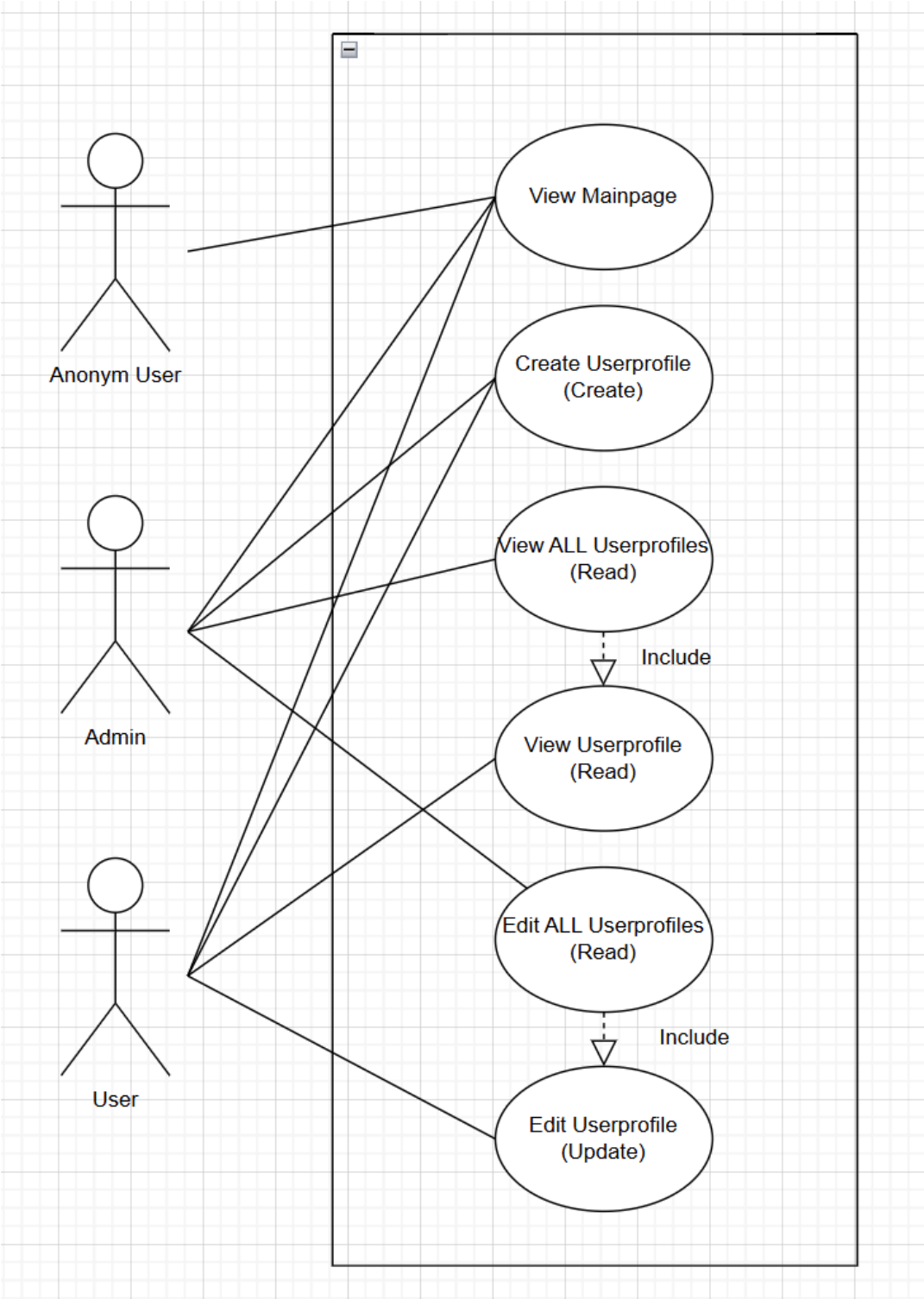
Die API sollte die erwarteten Daten zurückgeben und den zum Status passenden HTTP-Statuscode liefern.

Use-Case Description

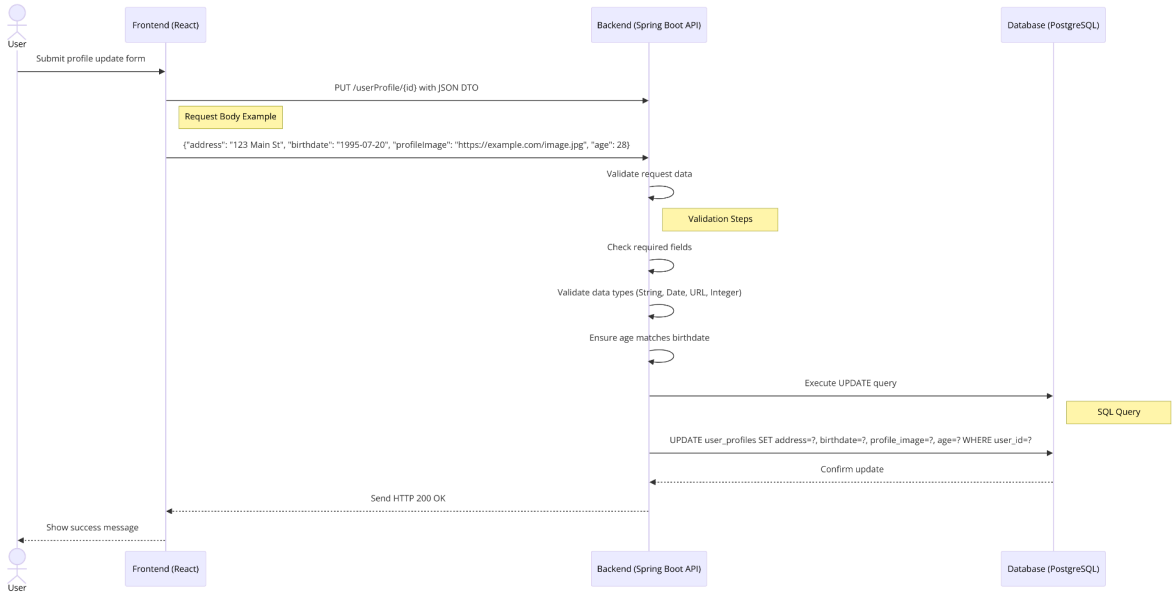
User	Admin
Beschreibung	Der Admin hat die Möglichkeit, die Details eines beliebigen Benutzerprofils im System zu bearbeiten. Dies kann notwendig sein, wenn Benutzerinformationen geändert oder aktualisiert werden müssen, z. B. bei einer Änderung der Kontaktdaten oder anderer persönlicher Informationen.
VorBedingungen	Der Admin muss erfolgreich im System angemeldet sein und über die erforderlichen Admin-Rechte verfügen, um Benutzerprofile zu bearbeiten.
	Das Benutzerprofil, das bearbeitet werden soll, muss bereits im System existieren.
NachBedingungen	Nach erfolgreicher Bearbeitung werden die Änderungen im Benutzerprofil gespeichert. Das System aktualisiert die entsprechenden Daten in der Datenbank, sodass die Änderungen in zukünftigen Abrufen des Profils berücksichtigt werden.
Normaler Ablauf	1. **Auswahl des Benutzerprofils:** Der Admin öffnet die Benutzerverwaltungsansicht im System, die eine Liste aller Benutzerprofile anzeigt. Alternativ kann der Admin die Suchfunktion nutzen, um ein bestimmtes Benutzerprofil zu finden.
	2. **Abrufen der Profildaten:** Sobald das gewünschte Benutzerprofil ausgewählt wurde, fragt das System die aktuellen Profildaten aus der Datenbank ab und stellt diese in einem bearbeitbaren Formular dar. Das Formular zeigt alle relevanten Felder an, die vom Admin geändert werden können.
	3. **Änderung der Profildaten:** Der Admin bearbeitet die angezeigten Daten nach Bedarf. Hier kann der Admin Felder wie den Benutzernamen, die E-Mail-Adresse oder andere persönliche Informationen ändern. Es ist wichtig, dass der Admin sicherstellt, dass alle Änderungen korrekt eingegeben werden, um die Konsistenz der Daten zu gewährleisten.
	4. **Speichern der Änderungen:** Nach der Bearbeitung klickt der Admin auf die Schaltfläche Speichern, um die Änderungen zu speichern. Das System überprüft nun die Eingabedaten.

	5. **Validierung der Änderungen:** Das System führt eine Validierung der eingegebenen Daten durch, um sicherzustellen, dass alle Felder korrekt ausgefüllt sind und keine ungültigen Werte eingegeben wurden (z. B. eine ungültige E-Mail-Adresse oder ein zu kurzes Passwort).
	6. **Speichern der validierten Daten:** Wenn die Daten korrekt sind, speichert das System die Änderungen in der Datenbank und bestätigt dem Admin, dass das Benutzerprofil erfolgreich aktualisiert wurde.
Alternativer Ablauf	1. **Fehlende oder ungültige Daten:** Wenn die Validierung der eingegebenen Daten fehlschlägt (z. B. wenn ein Pflichtfeld leer bleibt oder eine falsche Formatierung erkannt wird), zeigt das System eine Fehlermeldung an. Der Admin wird aufgefordert, die fehlerhaften Eingaben zu korrigieren, bevor das Profil gespeichert werden kann. In diesem Fall muss der Admin die Änderungen noch einmal überprüfen und das Formular erneut absenden.
	2. **Manuelle Fehlerkorrektur:** Falls der Admin versehentlich falsche Informationen eingegeben hat (z. B. beim Ändern eines Namens oder einer E-Mail-Adresse), kann der Admin jederzeit zurückkehren und die Änderungen manuell anpassen.
Ausnahmen	1. **Datenbankverbindungsprobleme:** Wenn während des Speichervorgangs ein Problem mit der Verbindung zur Datenbank auftritt, zeigt das System eine Fehlermeldung an und schlägt vor, die Aktion später zu wiederholen. Der Admin wird darauf hingewiesen, dass die Änderungen nicht gespeichert werden konnten und es notwendig sein könnte, die Verbindung zu überprüfen.
	2. **Zugriffsprobleme:** Sollte der Admin aufgrund eines Berechtigungsproblems oder fehlender Rechte nicht in der Lage sein, das Profil zu bearbeiten, wird eine entsprechende Fehlermeldung angezeigt. Der Admin muss sicherstellen, dass er über die notwendigen Rechte verfügt, um die Bearbeitung des Benutzerprofils durchzuführen.

Use-Case Diagram



Sequence Diagram



Das Sequenzdiagramm zeigt den Ablauf eines Profil-Update-Prozesses. Der Benutzer sendet über das Frontend (React) ein Formular zur Aktualisierung des Profils. Anschließend wird eine PUT-Anfrage mit den Profildaten im JSON-Format an das Backend (Spring Boot API) gesendet. Dort werden die empfangenen Daten validiert, einschließlich der Prüfung auf erforderliche Felder, Datentypen (String, Datum, URL, Integer) und die Übereinstimmung von Alter und Geburtsdatum. Nach erfolgreicher Validierung wird eine UPDATE-Abfrage an die PostgreSQL-Datenbank ausgeführt, um die Profildaten zu aktualisieren. Sobald die Datenbank die Aktualisierung bestätigt, sendet das Backend eine HTTP 200 OK-Antwort an das Frontend, das daraufhin eine Erfolgsmeldung an den Benutzer anzeigt.