Projektarbeit

Harun Silajdzija

Matrikelnummer 30265109

11.06.2025

—

Fachbereich
Anwendungsentwicklung

—

Fachhochschule Südwestfalen

SkillHub

Inhaltsverzeichnis

Projektaufgabe

Ziel dieses Projekts ist die Entwicklung einer webbasierten Unternehmensanwendung zur Erstellung, Planung, Buchung und Verwaltung von Kursen. Die Anwendung soll eine vollständige Benutzeroberfläche für CRUD-Operationen (Create, Read, Update, Delete) entsprechend der Benutzerrollen bieten und eine intuitive Handhabung ermöglichen. Wie in Abbildung 1 dargestellt, richtet sich die Anwendung an drei zentrale Nutzertypen – Administratoren, Trainer und Teilnehmer – deren unterschiedliche Berechtigungen und Aufgaben durch spezifische Funktionalitäten abgebildet werden.

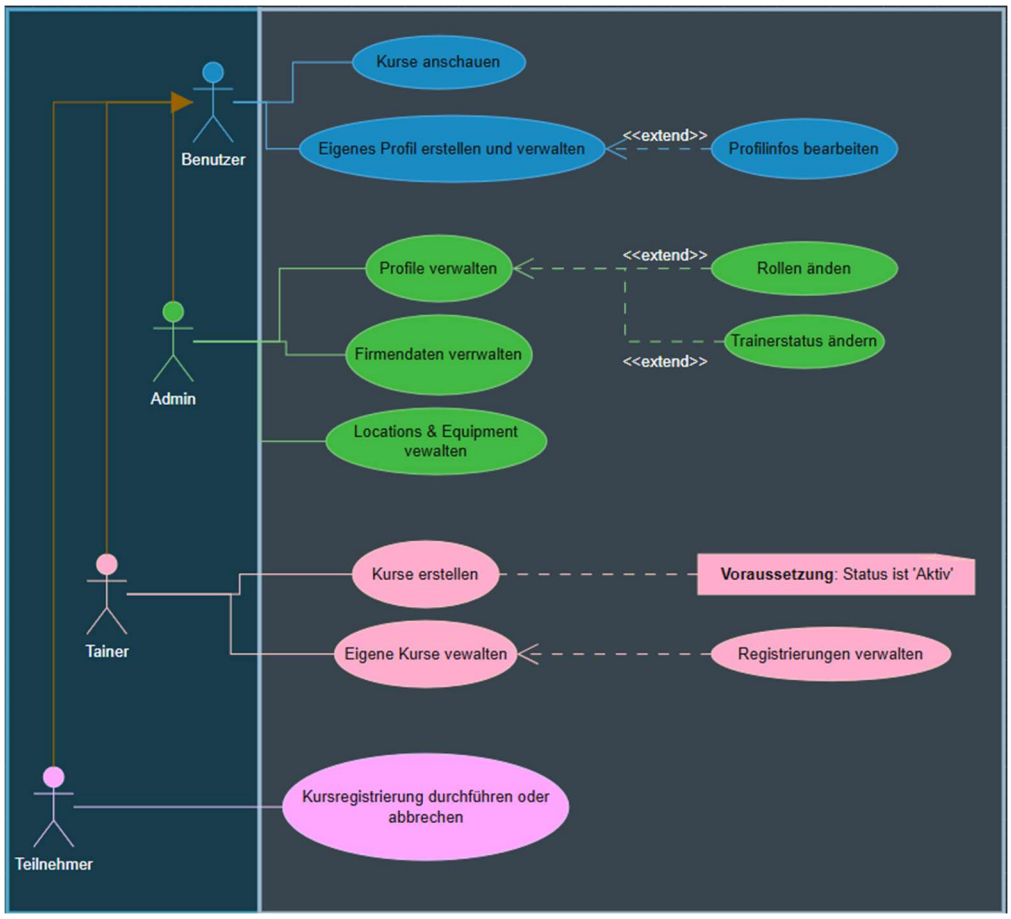

Abbildung 1: Anwendungsfalldiagramm

Allgemeine Benutzeranforderungen:

- Benutzer können eine Übersicht aller verfügbaren Kurse anzeigen. Dabei sollen Kursdetails wie Beschreibung, Zeit und Ort einsehbar sein.
- Benutzer können ein persönliches Profil anlegen und bearbeiten. Die Bearbeitungsmöglichkeit umfasst den Namen, Nachnamen, Mailadresse und Passwort.
- Auch kann jeder Benutzer seinen Account löschen.

Admin-Anforderungen

- Ein Administrator kann Benutzerprofile verwalten. Dazu gehören das Erstellen, Ansehen und Löschen von Profilen, sowie das Ändern von persönlichen Informationen, wie Name, Nachname und E-Mail.
- Es soll die Vergabe und der Entzug von Benutzerrollen ermöglicht werden, sowie die Änderung des Status für Trainer. Der Status bestimmt, ob ein Trainer aktuell Kurse erstellen darf.
- Des Weiteren pflegt der Admin die Daten zur Organisation oder Firma. Dies umfasst den Namen, die Anschrift, Kontaktdaten und die Öffnungszeiten.
- Als Admin verfügt der Benutzer die Berechtigung, Locations und Equipment zu erfassen, sowie den Status zu definieren, um so Veranstaltungsorte für die Erstellung von Kursen anzubieten.

Trainer-Anforderungen

- Trainer können neue Kurse erstellen, wenn sie einen aktiven Status haben.
- Sie können zu jeder Zeit ihre Kurse bearbeiten oder löschen und aktuelle Registrierungen verwalten.

Teilnehmer-Anforderungen

- Teilnehmer können Kursregistrierungen durchführen oder abbrechen.
- Eine Teilnahme muss allerdings erst vom Trainer eines Kurses bestätigt werden. Daher können Teilnehmer auch den Status ihrer Registrierung einsehen.

Nicht-funktionale Anforderungen

- Das Login-System für Benutzer umfasst eine Sicherheitsintegration mit Keycloak, basierend auf OpenID Connect als Identity Provider zur Authentifizierung und Autorisierung.
- Keycloaks Identity Brokering soll so konfiguriert sein, dass alternative Identity Provider genutzt werden können, um ein Profil anzumelden.
- Ein rollenbasiertes UI-Management soll die Benutzeroberfläche gemäß der Benutzerrolle einschränken und nur erlaubte Funktionen anzeigen.
- Formulare sollen Validierungsmechanismen haben, um die Benutzerfreundlichkeit zu erhöhen.
- Zusätzliche Validierungen sollen im Backend für eine sichere und stabile Anwendung sorgen.
- Die benutzeroberfläche basiert auf dem Angular Framework und soll ein responsives Design erfüllen.
- Entwicklung der Server-Anwendung basiert auf dem Spring Boot Framework. Es soll eine REST API für CRUD-Operationen zur Verfügung stellen.
- Sowohl UI als auch REST API werden rollenbasiert durch Mechanismen zur Authentifizierung geschützt.
- Eine Datenbank zur Persistierung von Daten, die für die Geschäftsprozesse notwendig sind, soll integriert werden. Die Benutzerverwaltung obliegt der Keycloak-Datenbank, welche von der Datenbank für die Server-Applikation getrennt sein muss.

Durch diese Anforderungen soll eine robuste und zuverlässige Lösung zur Kursplanung für Unternehmen bereitgestellt werden.

Fachkonzept

Das folgende Fachkonzeptmodell (siehe *Abbildung 2*) bildet die zentrale Datenstruktur der webbasierten Kursverwaltungsanwendung ab. Es stellt die relevanten Domänenobjekte sowie deren Beziehungen untereinander dar und dient als Grundlage für die Implementierung im Backend (Spring Boot). Die Objekte spiegeln die fachlichen Anforderungen aus dem Use-Case-Diagramm wider.

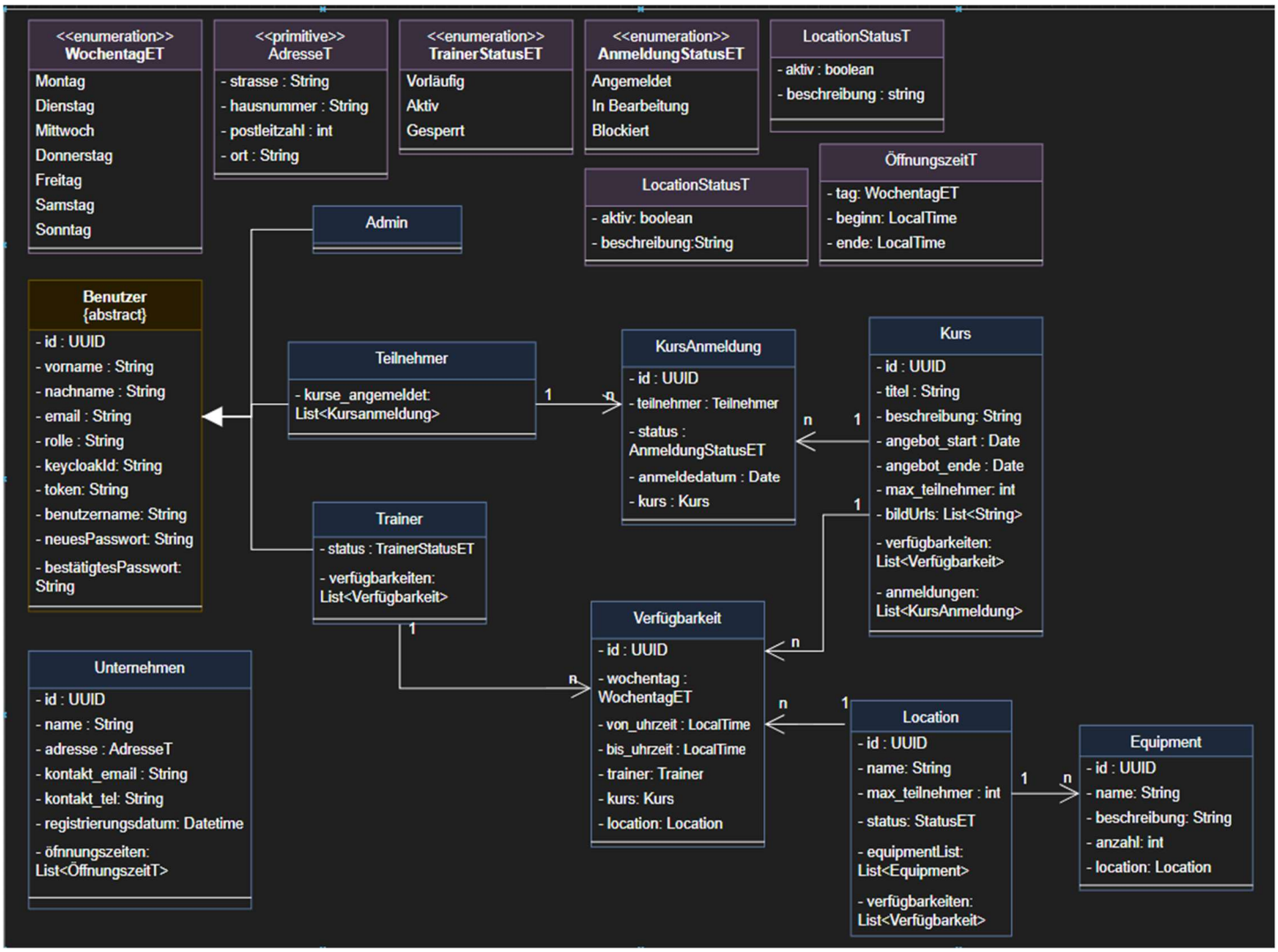

Abbildung 2: Fachkonzeptmodell

Haupt-Entitäten und Beziehungen

- Die abstrakte Klasse „Benutzer“ stellt die gemeinsame Basisklasse für alle Benutzerarten (Admin, Trainer, Teilnehmer) dar. Alle Attribute dieser Klasse sind als „@Transient“ annotiert, sodass sie nicht in der Datenbank persistiert werden, da sie bereits in Keycloak verfügbar sind. Lediglich die ID und Keycloak-ID werden zu Referenzzwecken persistiert und helfen Keylcoak-Daten zu einem bestimmten Benutzer abzurufen.
- Die Klasse Teilnehmer spezifiziert die Klasse Benutzer und stellt diejenigen Benutzer dar, die sich für Kurse anmelden können. Dabei kann ein Teilnehmer mehrere

Kursanmeldungen haben, während eine Kursanmeldung einem bestimmten Teilnehmer zuzuordnen ist. Eine Ausnahme besteht bei der redundanten Anmeldung für den gleichen Kurs. Dies soll nicht möglich sein, solange eine Kursanmeldung besteht.

- Eine weitere Spezialisierung der Klasse „Benutzer“ wird durch die Klasse „Trainer“ implementiert. Durch den Status des Trainers, können Administratoren die Berechtigung zur Erstellung von Kursen geben oder entziehen, ohne das Profil sperren bzw. entsperren zu müssen. Ein Trainer bietet einen Kurs durch das Ausdrücken von Verfügbarkeiten an, welche bei der Kurserstellung generiert werden. Ein Trainer kann dabei mehrere Verfügbarkeiten für den gleichen oder unterschiedliche Kurse haben. Eine Verfügbarkeit ist allerdings an einen festen Trainer gebunden.

Kurs-Management

- Ein Kurs wird durch Titel, Beschreibung, Start- und Enddatum, sowie einer maximalen Teilnehmerzahl definiert. Der Trainer kann bei der Erstellung von Kursen optional Bilder hochladen.
- Kurse können an unterschiedlichen Tagen verfügbar sein. Die Verfügbarkeit drückt ein Trainer durch die Auswahl eines Tages, einer Location und der Angabe eines Zeitfensters aus. Dabei ist eine Verfügbarkeit an einen Kurs zu einem Tag an einer Location gebunden. Die ausgewählte Location muss mindestens die Kapazität der maximalen Teilnehmerzahl des zu erstellenden Kurses haben.
- Kursanmeldungen drücken den Status der Anmeldung eines Teilnehmers aus. Der Trainer hat dadurch die Möglichkeit ein Feedback zu geben. Kursanmeldungen sind nicht für vergangene oder volle Kurse möglich.

Standortverwaltung

- Locations stellen Räumlichkeiten dar, welche das Unternehmen für Kurse und Veranstaltungen anbietet. Die aktuelle Logik setzt voraus, dass die Räumlichkeiten im Unternehmen liegen, da Öffnungszeiten an das Unternehmen gebunden sind. Eine Location kann vom Admin gesperrt oder freigegeben werden. Dies wirkt sich auf die Verfügbarkeit der Location bei einer Kurserstellung aus.
- Zusätzlich kann die Ausstattung einer Location erfasst werden. Dies erfolgt über die Klasse „Equipment“, welche den Namen, eine Beschreibung und die Anzahl erfasst. Ausstattung ist immer an eine feste Location gebunden, während eine Location verschiedene Ausstattung haben kann.
- Die Klasse „Unternehmen“ repräsentiert die Organisation, die Kurse anbietet. Hauptsächlich sollen dadurch die Adresse, Öffnungszeiten und Kontaktinformationen festgehalten werden.

Enumerationen / Typen

Zum Fachkonzept gehören darüber hinaus verschiedene Enumerationen und komplexe Typen, die zur Strukturierung und Typisierung der Entitätsattribute dienen. Diese erweitern die Funktionalität der Kernentitäten und sorgen für konsistente Datenmodelle. Eine Übersicht dieser Typen ist in Tabelle 1 dargestellt.

Name	Beschreibung
WochentagET	Montag – Sonntag
TrainerStatusET	Vorläufig, Aktiv, Gesperrt
AnmeldungStatusET	Angemeldet, in Bearbeitung, Blockiert
LocationStatusT	Aktiv (boolean), Beschreibung
AdresseT	Straße, Hausnummer, PLZ, Ort
ÖffnungszeitT	Tag (WochenTagET), Beginn, Ende (Uhrzeit)

Tabelle 1: Enumerationen und Typen

Systemarchitektur

Die Architektur der Anwendung (Abbildung 3) basiert auf drei zentralen Teilsystemen:

1. Keycloak als Identity Provider

Keycloak übernimmt die Authentifizierung und Autorisierung der Benutzer durch OpenID Connect. Es umfasst:

- **Realm**: Stellt die organisatorische Einheit für das Identitätsmanagement der Rechnungsanwendung dar.
- **Clients**: Definiert Zugriffsrichtlinien für das Frontend und Backend.
- **Benutzer & Rollen:** Verwaltung von Nutzern und deren Berechtigungen, um verschiedene Zugriffsebenen im System zu gewährleisten.
- **Identity Provider:** Ermöglicht durch Identity Brokering die Anbindung externer Authentifizierungsdienste (z. B. Google, GitHub oder andere OpenID-Connect-kompatible Provider). Benutzer können sich dadurch mit bestehenden Konten authentifizieren, ohne separate Zugangsdaten für die Anwendung erstellen zu müssen.

Durch diese Struktur stellt Keycloak sicher, dass nur autorisierte Nutzer Zugriff auf die Anwendung erhalten.

2. Spring Boot Backend mit REST-API und H2-Datenbank

Das Backend ist in Spring Boot entwickelt und stellt die zentrale Schnittstelle für Datenverwaltung und Geschäftslogik dar. Es verwendet:

- **Tomcat** als Applikationsserver für die Bereitstellung der REST-API.
- **H2 In-Memory-Datenbank**, die sich ideal für Demo- und Testzwecke eignet, da sie bei jedem Neustart der Anwendung den Ursprungszustand wiederherstellt.
- **Hibernate** als ORM (Object-Relational Mapping), das eine automatische Erstellung und Verwaltung der Datenbanktabellen aus den Entitätsklassen ermöglicht.
- **JPA** (Java Persistence API) für die Datenzugriffslogik mittels Repositories, die CRUD-Operationen über Java-Methoden ermöglichen

- **Entitätsklassen (Models):** Diese Java-Klassen repräsentieren Tabellen in der Datenbank. Sie enthalten Attribute als Spalten und sind mit Annotationen wie @Entity und @Table versehen.
- **Service-Klassen:** Sie kapseln die Geschäftslogik und ermöglichen den Zugriff auf die Repository-Schicht.
- **REST-Controller:** Stellen Endpunkte für HTTP-Anfragen bereit und ermöglicht die Kommunikation mit dem Frontend.
- **Websicherheitskonfiguration:** Integriert OAuth2 mit Keycloak und validiert eingehende Anfragen basierend auf den übermittelten Tokens.

3. Angular Frontend als Single Page Applikation (SPA)

Das Frontend ist als SPA mit Angular realisiert. Eine SPA bedeutet, dass die gesamte Anwendung auf einer einzigen HTML-Seite läuft und dynamisch Inhalte nachlädt, ohne vollständige Seitenneuladungen.

Struktur und Komponenten

- **Views**: Stellen die Grundstruktur der Benutzeroberfläche dar und bietet Interaktionsmöglichkeiten.
- **Generelle Komponenten**: Untergeordnete Komponenten zur Modularisierung, z. B. für Modal-Dialoge oder spezifische UI-Elemente.
- **Service-Klassen**:
  - Kapseln Geschäftslogik, um Komponenten zu entlasten.
  - Ermöglichen Datenaustausch zwischen Komponenten, z. B. durch das Verwalten von Bilder-Uploads.
  - API-Services bilden die Schnittstelle zum Backend.
- **Model-Klassen**:
  - Entsprechen den Entitätsklassen aus dem Backend.
  - Ermöglichen eine typisierte Verarbeitung und strukturierte Übermittlung von Daten an die REST-API.

Sicherheit und Authentifizierung

- **Authentifizierungs-Guard & Rollen-Direktive**: Prüfen, ob der Benutzer authentifiziert und autorisiert ist.
- **HTTP-Interceptor**: Fügt das von Keycloak bereitgestellte JWT-Token automatisch in jede API-Anfrage ein.
- **Keycloak-Service**:
  - Startet den Anmeldeprozess beim Laden der Anwendung.
  - Holt und speichert die Benutzerinformationen sowie das JWT-Token.

Durch diese Architektur wird eine sichere, skalierbare und modular aufgebaute Anwendung gewährleistet.

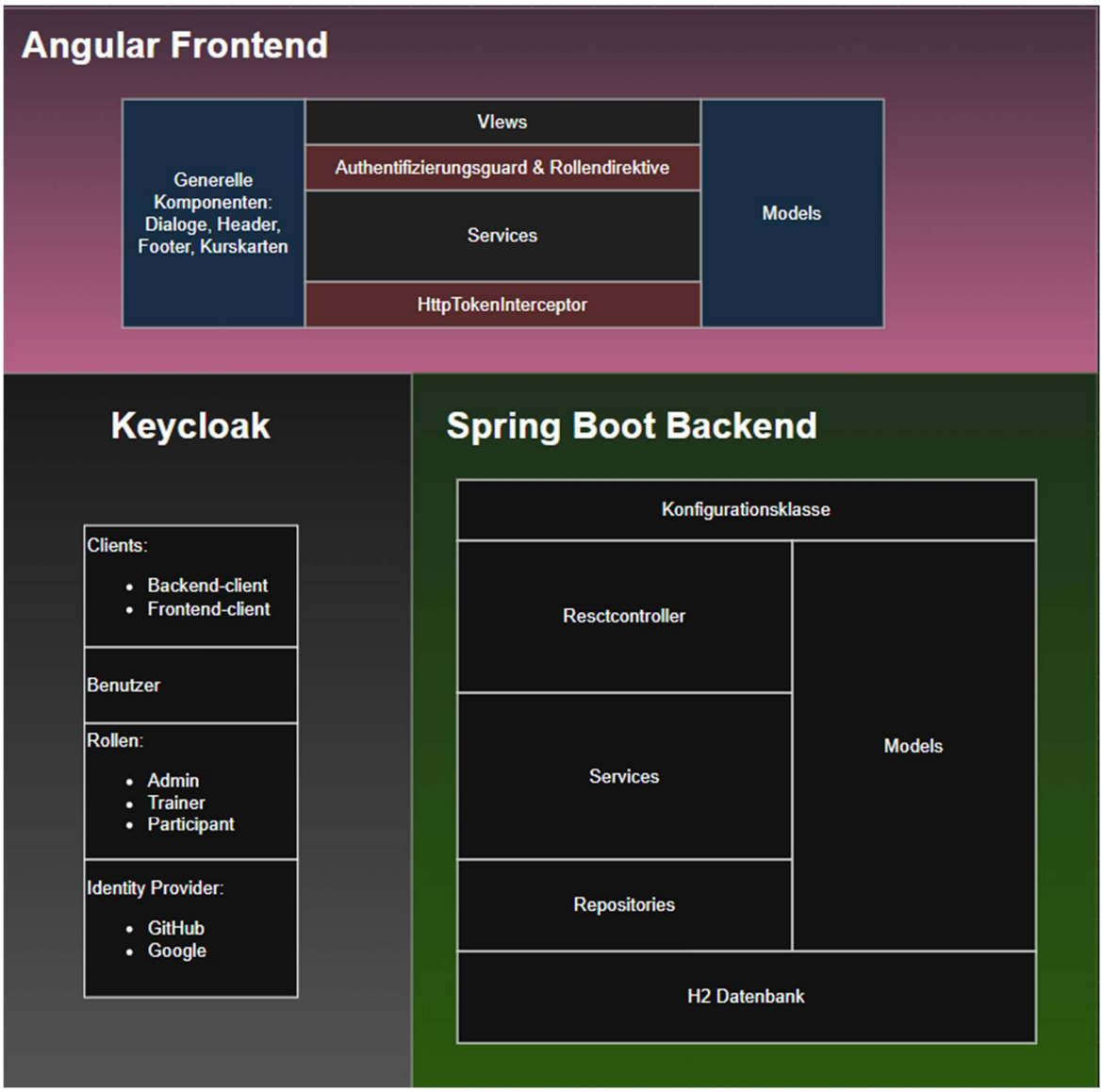

Abbildung 3: Systemarchitektur

Keycloak-Setup

Die Installation von Keycloak kann einfach über Docker erfolgen. Der folgende Befehl startet Keycloak in einem Container:

```
docker run -p 8080:8080 -e KC_BOOTSTRAP_ADMIN_USERNAME=admin -e KC_BOOTSTRAP_ADMIN_PASSWORD=admin quay.io/keycloak/keycloak:26.1.0 start-dev
```

Nach dem Start ist die Keycloak-Administrationskonsole unter „http://localhost:8080“ erreichbar. Dort kann sich der Administrator mit den angegebenen Zugangsdaten anmelden.

Für die Anwednung wird ein neuer Realm mit dem Namen „skillhub“ erstellt. Die Standardeinstellungen können hierbei übernommen werden.

Im nächsten Schritt werden zwei Clients eingerichtet:

Frontend-Client – „skillhub-frontend-client“

Dieser Client wird als öffentlicher Client (Public Client) konfiguriert, da der Nutzer direkt mit ihm interagiert. Damit entfällt die Notwendigkeit einer Client-Authentifizierung, weshalb das Feld "Client authentication" deaktiviert sein muss.
Das Frontend benötigt spezielle Sicherheitsmechanismen, da der Client auf Nutzerseite ausgeführt wird. Dies wird durch eine JWT-Token-basierte Authentifizierung realisiert:

1. Der Nutzer meldet sich über das Frontend an.
2. Nach erfolgreicher Authentifizierung erhält der Client ein JWT-Token von Keycloak.
3. Dieses Token wird bei jeder API-Anfrage an das Backend gesendet, welches die Gültigkeit überprüft.

Die **UUID** des Frontend-Clients muss in der “application.properties” des Backends eingetragen werden, damit das Backend die Client-Rollen dieses Clients anfragen kann. Die UUID wird in der URL des Frontend-Clients angezeigt.

Wichtige Konfigurationsoptionen für den Frontend-Client:

- **Standard Flow** aktivieren: Wird für die OpenID Connect-Authentifizierung benötigt, damit der Benutzer sich über den Browser anmelden kann.
- **Direct Access Grants** aktivieren: Ermöglicht Token-Anfragen mit Benutzername und Passwort direkt über das Frontend. Dies ist nützlich für API-Zugriffe außerhalb des Browsers.
- **Implicit Flow** aktivieren: Erlaubt es dem Client, Tokens direkt aus der URL zu extrahieren. Dies ist vorteilhaft für SPAs (Single Page Applications).

Konfiguration der URLs:

Da das Frontend während der Entwicklung sowohl direkt aus der IDE (localhost:4200) als auch über das Backend (localhost:3000) bereitgestellt werden kann, müssen folgende Einträge gemacht werden:

- **Root URL & Admin URL:** http://localhost:3000
- **Valid Redirect URIs:**
  - http://localhost:3000/* (Backend-gehostetes Frontend)
  - http://localhost:4200/* (Direktes Frontend in der Entwicklungsumgebung)
- **Valid Post Logout Redirect URIs:** Gleich wie die Redirect URIs

Abmeldeeinstellungen:

- **Backchannel Logout Session Required aktivieren**: Diese Option stellt sicher, dass Benutzer auf allen verbundenen Clients (z. B. bei mehreren Tabs oder Geräten) abgemeldet werden.

Rollen & Berechtigungen für den Frontend-Client:

Um Zugriffskontrollen zu ermöglichen, werden für den Client folgende Rollen erstellt:

- **admin**: Berechtigt den Nutzer, Benutzerprofile und Standortdaten zu verwalten.
- **trainer**: Berechtigt den Nutzer, Kurse zu erstellen und eigene Kurse zu verwalten.

- **participant**: Berechtigt den Nutzer, sich für Kurse an- und abzumelden.

Backend-Client – „skillhub -backend-client“

Das Backend benötigt eine sichere und nicht öffentliche Kommunikation mit Keycloak, weshalb dieser Client als vertraulicher Client (Confidential Client) konfiguriert wird.

Wichtige Konfigurationsoptionen für den Backend-Client:

- **Client Authentication aktivieren**: Da das Backend nicht direkt mit Nutzern interagiert, kann es sich sicher über ein **Client-Secret** authentifizieren.
- **Secret generieren**: Nach der Erstellung des Clients wird unter dem Reiter **"Credentials"** ein Secret generiert.
- **Eintragen des Secrets in die application.properties**

Zugriffseinstellungen für das Backend:

- **Root URL & Admin URL:** http://localhost:3000
- **Valid Redirect URIs:** http://localhost:3000/*
- **Valid Post Logout Redirect URIs:** Gleich wie die Redirect URIs

Abmeldeeinstellungen:

- **Backchannel Logout Session Required aktivieren**: Wie beim Frontend sorgt dies für eine vollständige Abmeldung auf allen verbundenen Clients.
- **Front Channel Logout aktivieren**: Da das Backend nicht über einen Browser arbeitet, aber API-Clients eine Logout-Nachricht erhalten sollen, kann diese Option aktiviert werden.

Identity Brokering mit Keycloak

Identity Brokering kommt zum Einsatz, um die Anmeldung über externe Identitätsanbieter wie Google und GitHub zu ermöglichen. Dadurch können sich Benutzer mit bestehenden Konten authentifizieren, ohne separate Zugangsdaten für die Kursverwaltungsanwendung erstellen zu müssen.

Der Prozess des Identity Brokering in Keycloak erfolgt nach einer definierten Abfolge (vgl. https://www.keycloak.org/docs/latest/server_admin/index.html#_identity_broker):

1. **Der Benutzer ruft die Anwendung auf.**
2. **Die Anwendung verweist den Benutzer zur Authentifizierung an Keycloak.**
3. **Keycloak fungiert als Identity Broker** und zeigt eine Auswahl verknüpfter externer Identity Provider (z. B. Google, GitHub) an.
4. **Der Benutzer wählt einen Anbieter** aus der Liste aus.
5. **Keycloak leitet den Benutzer zum gewählten Identity Provider weiter.**
6. **Der Benutzer gibt seine Zugangsdaten** auf der Seite des gewählten Providers ein.
7. **Der Identity Provider authentifiziert den Benutzer** und sendet eine Antwort an Keycloak.
8. **Keycloak verarbeitet die Rückmeldung** und erstellt ggf. ein Benutzerkonto oder ordnet es einem bestehenden Benutzer zu.
9. **Keycloak gibt die Authentifizierungsdaten an die Anwendung weiter**, welche bei erfolgreicher Anmeldung Zugriff gewährt.

Einrichtung von Identity Brokering
Die Konfiguration externer Identity Provider erfolgt über die **Keycloak Admin-Konsole**. In der im Projekt verwendeten Version ist die Einstellung unter **„Configure“ → „Identity Providers“** im linken Seitenmenü zu finden.
Unter der Kategorie **„Social“** werden verschiedene vordefinierte Anbieter wie Google, GitHub, Facebook usw. angezeigt. Bei Auswahl eines Providers öffnet sich ein Konfigurationsformular, in dem insbesondere folgende Informationen erforderlich sind:

- **Client ID**
- **Client Secret**
- **Redirect URI** (automatisch von Keycloak generiert und muss beim jeweiligen Provider eingetragen werden)

Diese Angaben erhält man direkt beim jeweiligen Anbieter nach Registrierung der Anwendung.

Integration von Google
Für die Integration von Google als Identity Provider wurde ein bewährtes Tutorial verwendet:

- https://medium.com/@stefannovak96/signing-in-with-google-with-keycloak-bf5166e93d1e

Dort wird beschrieben, wie man eine Google OAuth 2.0-Clientkonfiguration erstellt, die benötigten Zugangsdaten erhält und in der Keycloak-Konsole hinterlegt.

Integration von GitHub
Für die Einbindung von GitHub als externer Identity Provider ist folgendes Vorgehen erforderlich:

1. Besuche die GitHub Developer-Seite: https://github.com/settings/developers
2. Erstelle eine neue OAuth App:
    - **Application name**: frei wählbar
    - **Homepage URL**: die URL der eigenen Anwendung
    - **Authorization callback URL**: die Redirect URI, die von Keycloak im Konfigurationsformular für GitHub angezeigt wird
3. Nach dem Speichern erhält man:
    - **eine Client ID**
    - **die Möglichkeit, ein Client Secret zu generieren**
4. Diese Zugangsdaten werden anschließend in der Keycloak-Konsole unter „Identity Providers“ → „GitHub“ eingetragen.
5. Nach dem Speichern ist GitHub erfolgreich als Login-Option in der Anwendung verfügbar.

Geschäftsprozesse

Im Rahmen dieser Dokumentation werden ausschließlich die beiden Geschäftsprozesse **„Administrierung“** und **„Kursverwaltung“** modelliert und im Detail beschrieben. Die Auswahl dieser Prozesse erfolgte bewusst, da sie aufgrund ihrer Komplexität und ihres strategischen Stellenwerts für den Betrieb und die Nutzung der SkillHub-Plattform von zentraler Bedeutung sind. Alle weiteren, weniger komplexen Prozesse sind bereits durch die beschriebenen Anforderungen und das Fachkonzept ausreichend abgedeckt und werden daher an dieser Stelle nicht weiter modelliert.

Administrierung

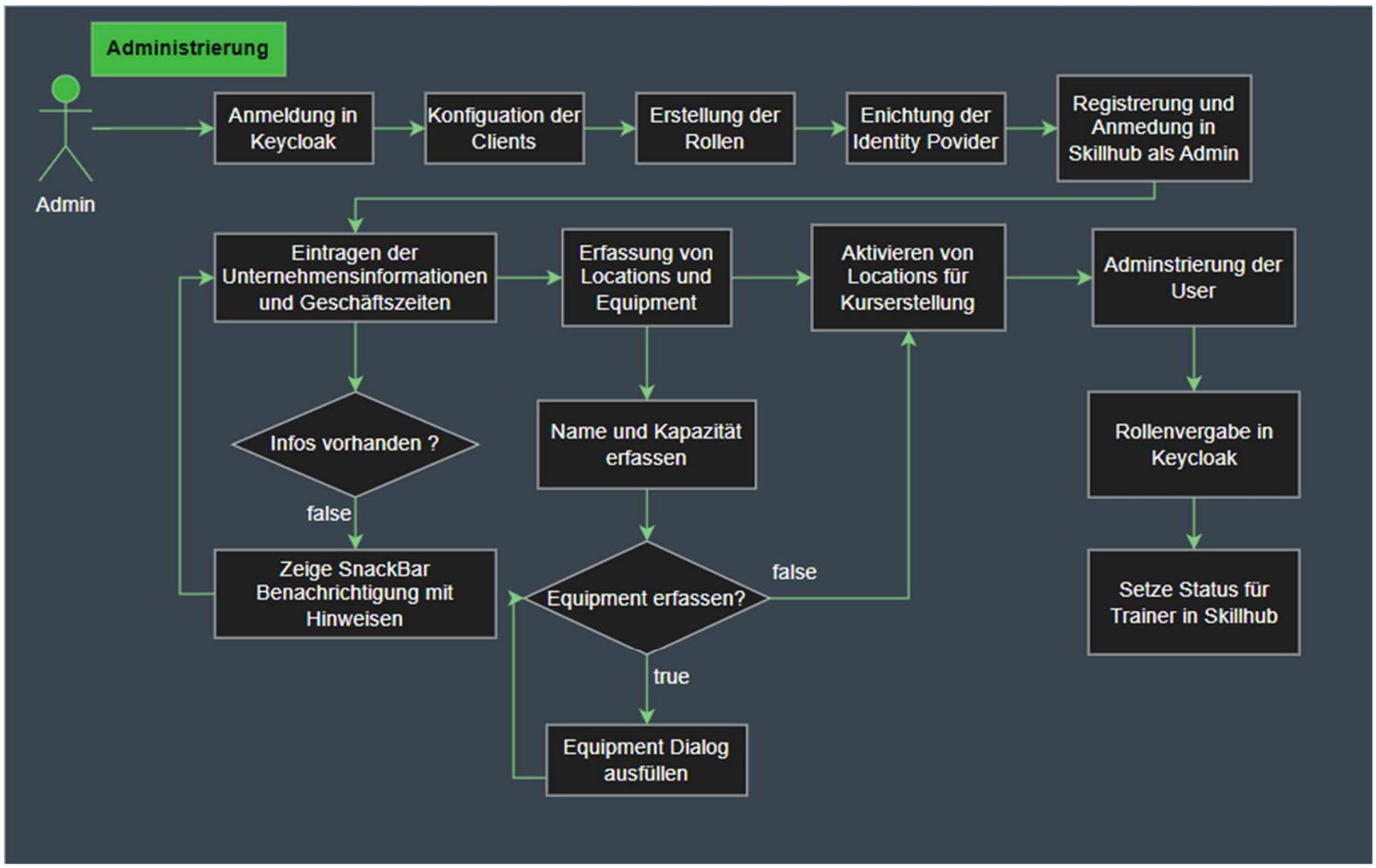

Abbildung 4: Prozessablauf für Administrierung

Der dargestellte Geschäftsprozess beschreibt die administrativen Tätigkeiten eines Admins bei der Einrichtung und Verwaltung der SkillHub-Plattform unter Einbindung von Keycloak als Identity Provider.

1. **Anmeldung und Grundkonfiguration**
   Der Prozess beginnt mit der Anmeldung des Admins in Keycloak. Anschließend erfolgt die Konfiguration der Clients, die Erstellung der Rollen sowie die Einrichtung des Identity Providers.
2. **Registrierung in SkillHub**
   Nach Abschluss der Keycloak-Konfiguration registriert sich der Admin in SkillHub und meldet sich dort an.

3. **Unternehmensdaten und Geschäftszeiten**
   Der Admin trägt die Unternehmensinformationen und Geschäftszeiten ein. Falls diese Informationen nicht vollständig sind, wird eine SnackBar-Benachrichtigung mit Hinweisen angezeigt.
4. **Erfassung von Locations und Equipment**
   Der Admin erfasst die verschiedenen Standorte (Locations) und deren Kapazitäten. Optional kann für jede Location auch das vorhandene Equipment erfasst werden. Ist dies gewünscht, wird ein entsprechender Dialog zur Eingabe angezeigt.
5. **Aktivierung von Locations**
   Nach der Erfassung werden die Locations für die Kurserstellung aktiviert.
6. **User- und Rollenkonfiguration**
   Der Admin verwaltet die Benutzer in SkillHub. Die Rollenzuweisung erfolgt in Keycloak. Für Trainer wird zusätzlich der entsprechende Status in SkillHub gesetzt.

Kursverwaltung

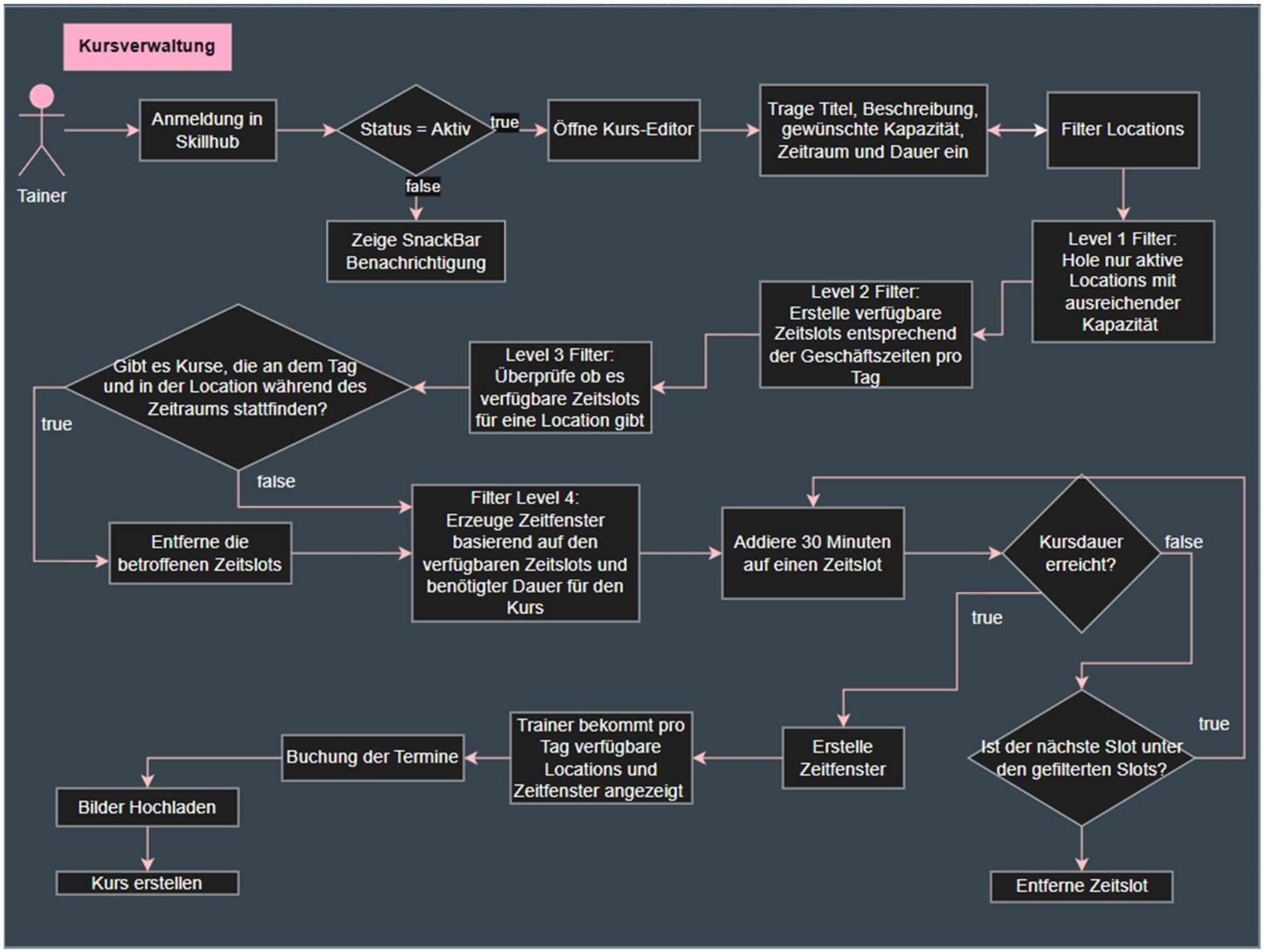

Abbildung 5: Prozessablauf für Kursverwaltung

Der dargestellte Prozess beschreibt die Schritte, die ein Trainer in SkillHub zur Verwaltung und Erstellung von Kursen durchführt.

1. **Anmeldung und Statusprüfung**
   Der Trainer meldet sich in SkillHub an. Es wird geprüft, ob der Status des Trainers „Aktiv“ ist. Ist dies nicht der Fall, erhält der Trainer eine SnackBar-Benachrichtigung.
2. **Kursdaten eingeben**
   Bei aktivem Status öffnet sich der Kurs-Editor. Der Trainer trägt Titel, Beschreibung, gewünschte Kapazität, Zeitraum und Dauer des Kurses ein.
3. **Location-Filterung**
   Zunächst werden nur aktive Locations mit ausreichender Kapazität (Level 1 Filter) berücksichtigt. Anschließend werden die Locations weiter gefiltert (Filter Locations).
4. **Zeitslot-Erstellung und -Filterung**
   - **Level 2 Filter:** Es werden verfügbare Zeitslots entsprechend den Geschäftszeiten pro Tag erstellt.
   - **Level 3 Filter:** Es wird geprüft, ob es verfügbare Zeitslots für eine Location gibt.
   - **Level 4 Filter:** Basierend auf den verfügbaren Zeitslots und der benötigten Kursdauer werden Zeitfenster erzeugt.
5. **Kollisionserkennung**
   Es wird geprüft, ob an dem gewünschten Tag und in der gewünschten Location bereits Kurse stattfinden. Falls ja, werden die betroffenen Zeitslots entfernt.
6. **Zeitslot-Iteration**
   Für die Erstellung eines passenden Zeitfensters wird iterativ jeweils 30 Minuten auf einen Zeitslot addiert, bis die gewünschte Kursdauer erreicht ist.
   - Ist die Kursdauer erreicht, wird geprüft, ob der nächste Slot unter den gefilterten Slots ist. Falls ja, wird der Zeitslot entfernt.
   - Andernfalls wird das Zeitfenster erstellt.
7. **Buchung und Kursanlage**
   Dem Trainer werden pro Tag die verfügbaren Locations und Zeitfenster angezeigt. Nach Auswahl erfolgt die Buchung der Termine. Optional können Bilder hochgeladen werden. Abschließend wird der Kurs erstellt.

Integration und Deployment

Grundlegende Voraussetzungen

- Java 21 und Maven 3.9.6
- Node 18.19.1

Integration

Die Integration erfordert das zuvor beschriebene Keycloak-Setup.

Frontend-Konfiguration

Im Frontend müssen die Hostnamen für die Backend-Applikation und den Keycloak-Server in den Dateien „environment.ts“ bzw. „environment.development.ts“ eingetragen werden. Falls der Name des Realms und Clients gemäß dem Keycloak-Setup gewählt wurde, sind hier keine weiteren Änderungen erforderlich. Ebenso ist keine Anpassung notwendig, wenn Keycloak und Backend über „localhost“ erreichbar sind, da diese Standardkonfiguration bereits bei der Auslieferung

eingetragen ist. Sollte die Backend-Applikation auf einem anderen Port laufen, muss dieser in der Backend-URL angepasst werden.

Backend-Konfiguration

Im Backend befindet sich die Konfiguration in der Datei „application.properties“. Die Applikation läuft standardmäßig auf Port 3000. Falls der Port geändert wird oder die Applikation nicht lokal gehostet wird, muss die Änderung sowohl in der Frontend-Konfigurationsdatei als auch in den Keycloak-Client-Einstellungen berücksichtigt werden (siehe Keycloak-Setup). Der Parameter zur Integration mit Keycloak „issuer-uri“ muss angepasst werden, falls Keycloak nicht über „localhost“ erreichbar ist oder ein anderer Name für den Realm verwendet wird. Ebenso müssen die Einstellungen „provider“ und „client-id“ gegebenenfalls geändert werden. Das Secret für den Backend-Client muss eingetragen werden, da es individuell generiert wird und nicht mit ausgeliefert wird.

Deployment und Ausführung

Nach Abschluss aller Konfigurationen kann die Anwendung über die parent-pom.xml im Hauptverzeichnis kompiliert werden. Dazu wird folgender Befehl ausgeführt: „mvn package“.

Die parent-pom.xml bindet die pom.xml des Frontends und Backends ein. Dabei werden:

- Die Node-Module installiert.
- Das Frontend gebaut und die kompilierten Dateien in das Verzeichnis static des Backends kopiert.
- Das Backend kompiliert und zusammen mit den Frontend-Dateien in ein JAR-Paket verpackt.

Auf Windows kann die Anwendung über die Start-Batchdatei im Hauptverzeichnis gestartet werden, beispielsweise über die Eingabeaufforderung.

Alternativ können beide Teilsysteme getrennt in der Entwicklungsumgebung (z. B. Visual Studio Code) gestartet werden:

- Frontend: Nach Installation der Node-Module (npm install) mit „ng serve“ starten.
- Backend: Als Java-Anwendung ausführen. In Visual Studio Code wird hierfür die Erweiterung „Spring Boot Dashboard" empfohlen.

Tests und Code-Qualität

Das Backend enthält umfangreiche Unit-Tests, die beim Kompilieren automatisch ausgeführt werden. Diese Tests stellen sicher, dass alle Komponenten reibungslos funktionieren und Änderungen frühzeitig erkannt werden. Dabei werden sowohl korrekte als auch fehlerhafte Fälle getestet.

Vielen Dank.