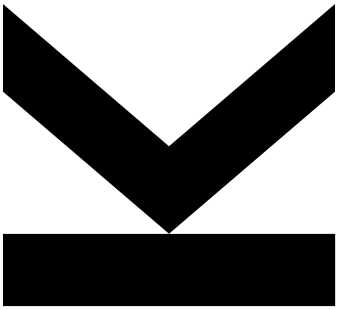
JOHANNES KEPLER
UNIVERSITÄT LINZ

Eingereicht von
Name des / der Studierenden

Angefertigt am
Institut für Wirtschaftsinformatik – Software Engineering

Monat Jahr
November 2023

PROJEKTBERICHT IDENTITY MANAGEMENT



KV Service Engineering – Gruppe D

JOHANNES KEPLER UNIVERSITÄT LINZ
Altenberger Straße 69
4040 Linz, Österreich
jku.at
DVR 0093696

Inhaltsverzeichnis

1. Funktionalität

Das System bietet eine umfassende Funktionalität zur Verwaltung von Übungsabgaben in einer Lehrveranstaltung (LVA). Die Client/Server-Kommunikation bildet die Grundlage der Implementierung, die zwei Benutzergruppen unterstützt: LVA-Leiter und Studierende.

Studierende können sich mit ihrer E-Mail-Adresse und einem Passwort registrieren. Nach erfolgreicher Authentifizierung können sie Abgaben. LVA-Leiter haben Zugriff auf eine Liste aller registrierten Studierenden und können die erstellten Abgaben einsehen und bewerten.

Die Anwendung stellt zwei Ansichten bereit: eine Listenansicht für eine schnelle Übersicht über die Abgaben und eine Detailansicht für tiefergehende Informationen. Die grundlegenden CRUD-Operationen (Erstellen, Lesen, Aktualisieren, Löschen) ermöglichen eine flexible Datenverwaltung.

Die serverseitige Datenhaltung erfolgt mit MariaDB, was eine zuverlässige Speicherung und Abfrage der Daten sicherstellt. Für die Authentifizierung wird Keycloak verwendet. Diese Plattform wurde aufgrund ihrer weiten Verbreitung und umfassenden Dokumentation ausgewählt. Sie bietet einfache Integrationsmöglichkeiten mit verschiedenen Technologien und gewährleistet eine sichere Benutzerverwaltung.

Die Realisierung dieses Systems konzentriert sich auf die Authentifizierung und Rollenverwaltung. LVA-Leiter haben erweiterte Rechte, einschließlich der Verwaltung von Studierenden, während Studierende ausschließlich auf ihre eigenen Abgaben zugreifen können. Die gewählte Technologie ermöglicht eine zukunftssichere Implementierung und bietet Raum für Erweiterungen.

2. Keycloak

Keycloak ist eine Open-Source-Lösung für Identitäts- und Zugriffsmanagement, die es erleichtert, Authentifizierung und Autorisierung in Anwendungen zu integrieren. Entwickler können sich auf die Kernfunktionen ihrer Anwendung konzentrieren, da Keycloak das Management von Benutzerdaten und Authentifizierungsprozessen übernimmt.

2.1. Grundlegende Funktionen

Keycloak bietet umfangreiche Features wie Benutzerföderation, Authentifizierung, Rollen- und Benutzerverwaltung sowie Autorisierungsmechanismen. Mit Single-Sign-On und Single-Sign-Out ermöglicht die Plattform eine nahtlose Benutzererfahrung, indem sich Nutzer einmalig anmelden und damit Zugang zu mehreren Anwendungen erhalten.
Darüber hinaus unterstützt Keycloak die Anmeldung über soziale Netzwerke sowie die Integration mit LDAP- und Active Directory-Diensten. Für die Verwaltung von Benutzern und Anwendungen stellt die Admin-Konsole eine zentrale Oberfläche bereit.

2.2. Rollen und Berechtigungen

In der aktuellen Implementierung werden drei Rollen definiert:

- **LVAAdmin**: Verantwortlich für die Verwaltung der gesamten Anwendung, einschließlich Benutzer- und Rollenmanagement.
- **LVALeiter**: Hat Zugriff auf alle registrierten Studierenden und deren Übungsabgaben. Bewertet die Abgaben.
- **Studierender**: Kann Übungsabgaben erstellen.

2.3. Authentifizierung und Client-Einrichtung

Für die Kommunikation zwischen der Anwendung und Keycloak wurde der Client **Spring-auth** eingerichtet. Dieser Client ermöglicht es der Anwendung, auf die Authentifizierungs- und Autorisierungsdienste von Keycloak zuzugreifen.
Der Keycloak-Server läuft in einem Docker-Container auf Port 8080. Details zur Konfiguration sind in der **Docker-Compose**-Datei zu finden.

2.4. Ablauf der Authentifizierung

1. Ein Benutzer greift auf eine geschützte Ressource der Anwendung zu.
2. Die Anwendung leitet ihn zur Anmeldeseite von Keycloak weiter.
3. Der Benutzer gibt seine Zugangsdaten ein, und Keycloak übernimmt die Authentifizierung.
4. Nach erfolgreicher Anmeldung erhält die Anwendung zwei Token:
    - **Identitätstoken**: Enthält Benutzerinformationen wie Name und E-Mail-Adresse.
    - **Zugriffstoken**: Beinhaltet die Rollen und Zugriffsrechte des Benutzers.

3. MariaDB

MariaDB ist ein leistungsstarkes und vielseitiges Open-Source-Datenbankmanagementsystem, das von den ursprünglichen Entwicklern von MySQL ins Leben gerufen wurde. Es ist in C und C++ geschrieben und unterstützt eine Vielzahl von Betriebssystemen, darunter Windows, Linux und macOS. Aufgrund seiner hohen Geschwindigkeit, Zuverlässigkeit und einfachen Handhabung hat sich MariaDB als hervorragende Lösung für zahlreiche Anwendungsfälle etabliert, von kleinen Projekten bis hin zu groß angelegten Webanwendungen und E-Commerce-Plattformen.

3.1. Warum MariaDB?

Für das Projekt, welches sich auf die Verwaltung von Lehrveranstaltungsabgaben konzentriert, wurden MariaDBs ausgewählt, weil bereits erhebliches Vorwissen im Team im Umgang mit diesem DBMS vorhanden war..

3.2. Integration mit Docker-Compose

MariaDB wurde im Rahmen des Projekts mittels Docker-Compose eingerichtet, um eine einfache Bereitstellung und Verwaltung der Datenbank zu ermöglichen.

4. Spring Boot

Spring Boot, eine erweiterte Plattform basierend auf dem Spring Framework, bietet eine einfache Möglichkeit, Java-Anwendungen zu entwickeln, insbesondere Webanwendungen. Spring Boot ermöglicht es durch seine Annotationen Projekte effizient umzusetzen, indem es viele Konfigurationsaspekte automatisiert und vorkonfigurierte Funktionen bereitstellt. Durch die Bereitstellung von Embedded-Servern und automatischem Konfigurationsmanagement lassen sich Anwendungen schnell und unkompliziert starten.

4.1. Nutzung von Spring Boot im Projekt

Im Rahmen unseres Projekts wird Spring Boot verwendet, um eine Web-Anwendung zu entwickeln. In Kombination mit Maven als Build-Tool und für das Dependency-Management sorgt Spring Boot für eine strukturierte und leicht wartbare Entwicklungsumgebung.

4.2. Visualisierung der Daten mit Thymeleaf

Für die Frontend-Darstellung wurde Thymeleaf als Template-Engine integriert. Thymeleaf ermöglicht es, serverseitig generierte HTML-Seiten dynamisch zu erstellen und Daten aus der Datenbank effizient zu visualisieren. Die Integration in Spring Boot erfolgt nahtlos, wodurch eine benutzerfreundliche Oberfläche für die Listen- und Detailansichten der Übungsabgaben bereitgestellt wird.

4.3. Verwendung von Jakarta als ORM-Layer

Zur Verwaltung der Datenbankinteraktionen wurde Jakarta als ORM (Object-Relational Mapping) Layer implementiert. Dieser Layer erleichtert die Abbildung von Java-Objekten auf Datenbanktabellen und ermöglicht eine saubere Trennung von Geschäftslogik und Datenbankzugriff. Mit der Hilfe von Jakarta können CRUD-Operationen (Create, Read, Update, Delete) einfach und effizient umgesetzt werden.

4.4. Zusammenspiel der Technologien

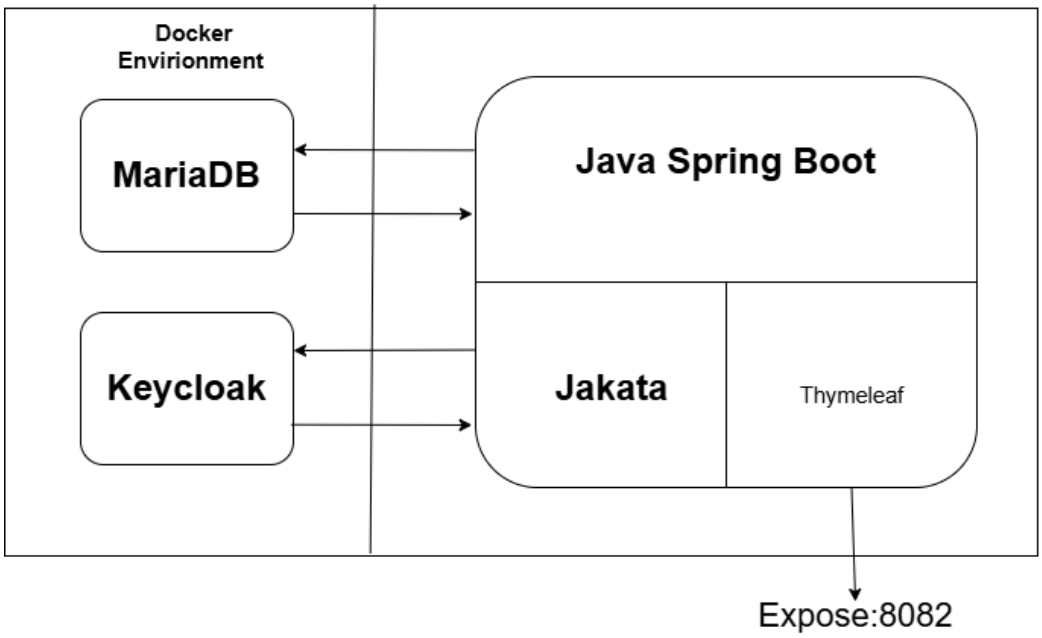
In der aktuellen Implementierung handelt es sich um eine monolithische Anwendung, in der sowohl die Datenverarbeitung als auch die Visualisierung der Daten integriert sind. Das Backend im klassischen Sinne entfällt, da alle Verabreitungsschritte und Datenverarbeitungsprozesse direkt innerhalb der Anwendung ablaufen. Die Daten werden lediglich zur dauerhaften Speicherung an die Datenbank weitergeleitet.

5. Projekt

5.1. Architektur

Das Diagramm stellt eine Monolith-Architektur dar, die in einem Docker-Umfeld betrieben wird.

- **MariaDB:** Eine relationale Datenbank, die für die Speicherung der persistenten Daten zuständig ist. Sie empfängt Daten von der Hauptanwendung und stellt sie bei Bedarf bereit. MariaDB läuft in einem Docker-Container.
- **Keycloak**: Keycloak wird für die Authentifizierung und Autorisierung verwendet. Es ermöglicht die sichere Verwaltung von Benutzern und deren Zugriff auf die Anwendung. Keycloak ist über Docker konfiguriert und integriert sich direkt in die Spring Boot-Anwendung.
- **Java Spring Boot**: Dies ist der Kern der Anwendung. Es beinhaltet sowohl die Datenverarbeitung als auch die Visualisierung und kombiniert sie in einer monolithischen Struktur.
    - **Jakarta**: Dieser Teil dient als ORM (Object-Relational Mapping)-Schicht, die die Interaktion mit der MariaDB-Datenbank abwickelt. Hier werden die Datenmodelle definiert und verarbeitet.
    - **Thymeleaf**: Eine Template-Engine, die für die dynamische Generierung von HTML-Seiten zuständig ist. Sie sorgt für die Darstellung der Daten, die von der Jakarta-Schicht verarbeitet wurden.



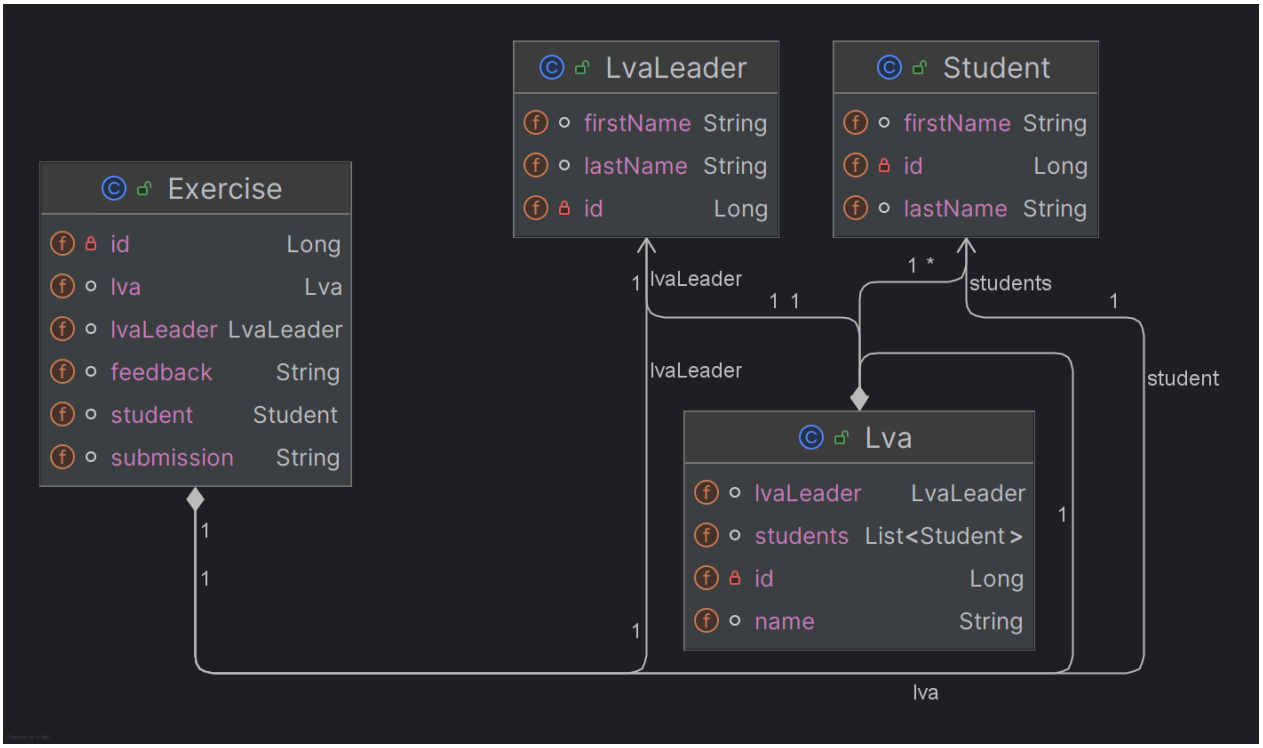
5.2. Klassendiagramm

Das UML-Diagramm stellt die Klassenstruktur und die Beziehungen in einem Lehrveranstaltungsverwaltungssystem dar. Im Zentrum des Systems steht die Klasse Lva (Lehrveranstaltung), die die grundlegenden Informationen zur Veranstaltung speichert. Jede Lva wird von genau einem LvaLeader (Lehrveranstaltungsleiter) verwaltet, der durch Attribute wie firstName, lastName und eine eindeutige id charakterisiert ist.

Eine Lva kann mehrere Students (Studierende) enthalten, die durch ihre persönlichen Daten wie firstName, lastName und eine eindeutige id identifiziert werden. Die Beziehung zwischen Lva und Student ist eine N-zu-M-Beziehung, was bedeutet, dass eine Lehrveranstaltung mehrere Studierende haben kann, ein Student benfalls zu mehreren Lehrveranstaltung zugehörig ist.

Zusätzlich enthält jede Lva eine oder mehrere Exercises (Übungsaufgaben). Jede Exercise gehört genau zu einer Lva und ist einem spezifischen Student zugeordnet, der die Übung eingereicht hat. Die Klasse Exercise enthält Attribute wie eine eindeutige id, die Übungsabgabe (submission) sowie ein feedback, das vom LvaLeader gegeben wird.

Die Beziehung zwischen Exercise und LvaLeader ist eine N-zu-1-Beziehung, da jede Übung genau von einem Lehrveranstaltungsleiter bewertet wird. Die Verknüpfung von Exercise mit Lva und Student stellt sicher, dass jede Übung eindeutig einer Lehrveranstaltung und einem Studierenden zugeordnet ist.



5.3. Projektstart

Um das Projekt erfolgreich auszuführen, müssen sowohl die Docker-Container als auch die Spring Boot-Anwendung gestartet werden. Das Projekt umfasst eine Docker-Umgebung, die die MariaDB-Datenbank und den Keycloak-Server bereitstellt, sowie eine Spring Boot-Anwendung, welche über IntelliJ gestartet werden muss und auf Port 8082 läuft.

Zunächst wird die Docker-Umgebung gestartet. Navigieren Sie dazu in das Verzeichnis, das die docker-compose.yml-Datei enthält. Mit dem Befehl docker-compose up -d werden die Docker-Container im Hintergrund gestartet. Diese Container umfassen die MariaDB-Datenbank, die für die persistente Speicherung der Daten zuständig ist, und Keycloak, das die Authentifizierungs- und Autorisierungsdienste bereitstellt. Nach dem Start der Container kann mit docker ps überprüft werden, ob diese erfolgreich ausgeführt werden.

Sobald die Docker-Dienste laufen, wird die Spring Boot-Anwendung in IntelliJ IDEA gestartet. Öffnen Sie dazu das Projekt in IntelliJ und stellen Sie sicher, dass die Anwendung auf Port 8082 konfiguriert ist. Diese Einstellung finden Sie in der application.properties, wo server.port=8082 angegeben sein sollte. Führen Sie anschließend die Hauptklasse der Spring Boot-Anwendung aus. Nach erfolgreichem Start ist die Anwendung über die URL http://localhost:8082 erreichbar.

Während die Anwendung läuft, können die bereitgestellten Dienste getestet werden. Über die Benutzeroberfläche der Anwendung lassen sich die verschiedenen Funktionen, wie die Verwaltung von Lehrveranstaltungen und Übungsabgaben, ausprobieren. Gleichzeitig ist Keycloak unter http://localhost:8080 zugänglich, um Benutzer und Rollen zu verwalten.

6. Anhang

6.1. Docker-Compose

```yaml
version: '3'

services:
  LVAService:
    image: mariadb
    container_name: LVAServiceDB
    ports:
      - 3306:3306
    environment:
      MYSQL_ROOT_PASSWORD: root
    volumes:
      - ./mariadb-data:/var/lib/mysql
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql  # Mounting init SQL file

  keycloak:
    image: keycloak/keycloak
    container_name: LVAServiceKeycloak
    ports:
      - 8080:8080
    environment:
      KEYCLOAK_ADMIN: admin
      KEYCLOAK_ADMIN_PASSWORD: admin
    volumes:
      - ./realm.json:/opt/keycloak/data/import/realm.json
    command:
      - start-dev
      - --import-realm
    #entrypoint: '/opt/keycloak/bin/kc.sh start-dev --import-realm''
```