Softwaretechnik-Projekt

Prof. Dr. Jörg Hettel
Hochschule Kaiserslautern
Studiengang IT-Analyst
– Version: 6. Mai 2024–

Spezifikation zur Projektarbeit SoSe 2024

Inhaltsverzeichnis

---

1 Vorbemerkung

Ziel der Projektarbeit ist die Anwendung der in dem Modul besprochenen Technologien. Im Rahmen der Projektarbeit soll eine Single-Page-Anwendung geschrieben werden. Das Design der Bedienoberflächen steht hierbei nicht im Vordergrund.

1.1 Technologien

Folgende Technologien sollen eingesetzt werden:

- **Web:** HTML, CSS und JavaScript (keine Frameworks)
- **Server:** Spring-Boot (REST-Anwendung)
- **Datenbank:** Als Datenbank soll eine MySQL verwendet werden.
- **Docker**: Einzelne Anwendungskomponenten laufen in einem Docker-Container

2 Das Datenbank-Schema

Das UML- und Datenmodell können Sie den Abbildung 1 und 2 entsprechen. Die Datenbank-Scripte finden Sie auch zusätzlich auf der OLAT-Kursseite.

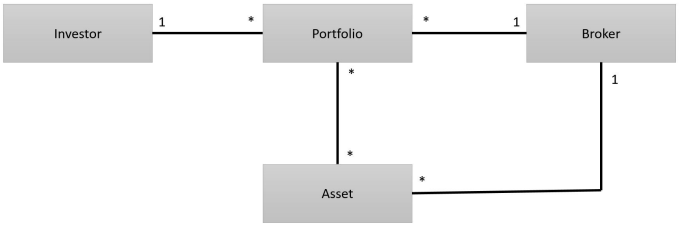

Abbildung 1: UML-Model.

```sql
CREATE TABLE investor (
    investor_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(20) NOT NULL UNIQUE,
    firstname VARCHAR(20) NOT NULL,
    lastname VARCHAR(20) NOT NULL,
    password_hash BINARY(20) NOT NULL,
    password_salt BINARY(32) NOT NULL
) CHARACTER SET utf8mb4;

CREATE TABLE broker (
    broker_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(20) NOT NULL UNIQUE,
    company VARCHAR(20) NOT NULL,
    password_hash BINARY(20) NOT NULL,
    password_salt BINARY(32) NOT NULL
) CHARACTER SET utf8mb4;


CREATE TABLE portfolio (
    portfolio_id INT AUTO_INCREMENT PRIMARY KEY,
    investor_id INT NOT NULL,
    broker_id INT NOT NULL,
    creation_date TIMESTAMP NOT NULL,
    FOREIGN KEY (investor_id) REFERENCES investor(investor_id),
    FOREIGN KEY (broker_id) REFERENCES broker(broker_id),
    CONSTRAINT investorRelation UNIQUE (investor_id,broker_id)
) CHARACTER SET utf8mb4;

CREATE TABLE asset (
    asset_id INT AUTO_INCREMENT PRIMARY KEY,
    broker_id INT NOT NULL,
    kind ENUM ('Share', 'Bond') NOT NULL,
    name VARCHAR(20) NOT NULL,
    FOREIGN KEY (broker_id) REFERENCES broker(broker_id)
) CHARACTER SET utf8mb4;

CREATE TABLE portfolio_assets (
    investor_id INT NOT NULL,
    asset_id INT NOT NULL,
    PRIMARY KEY (investor_id, asset_id),
    FOREIGN KEY (investor_id) REFERENCES investor(investor_id),
    FOREIGN KEY (asset_id) REFERENCES asset(asset_id)
) CHARACTER SET utf8mb4;
```

Abbildung 2: Datenbankschema.

3 Projektaufgabe

3.1 Kurzbeschreibung

Mit der zu entwickelnden Anwendung können Portfolios verwaltet werden. Ein Investor kann mehrere Portfolios besitzen. Ein Portfolio ist hierbei genau einem Broker zugeordnet. Ein Broker hat nun mehrere Assets im Angebot. Ein Investor kann sich nun aus dieser Auswahl Assets in sein Portfolio legen. Ein Porfolio enhält kann nur Assets enthalten, die von dem zugehörigen Broker angeboten werden.

3.2 Die Anwendung

Die Anwendung soll zwei Einstiegspunkte bereitstellen. Einen Zugang für die Investoren und einen Zugang für die Broker. Dies kann z.B. über zwei verschiedene Web-Seiten erfolgen. (Es gibt aber nur eine REST-Anwendung, die mehrere Controller besitzt.)

3.2.1 Funktionalität für Broker

- Broker können sich über Benutzernamen und ein korrektes Password an der Anwendung anmelden.
- Weiter ist es möglich, dass sich Broker auch neu an der Anwendung registrieren können. (Daten werden dann in der Datenbank abgelegt.)
- Brokern können sich auch wieder deregistrieren (löschen).
- Nach erfolgreicher Anmeldung bzw. Registrierung werden dem Broker seine Asseets angezeigt und die Broker können ihre Assets verwalten:
  - Assets neu anlegen
  - angelegt Assets löschen, falls sie nicht einem Portfolio zugeordnet sind. Ansonsten ist das Löschen nicht möglich und es wird eine entsprechende Fehlermeldung angezeigt. Alternativ kann die Löschmöglichkeit schon bei der Anzeige *disabled* werden.
- Broker können sich von der Anwendung auch wieder abmelden.
- Nur angemeldete Broker können die REST-Schnittstelle aufrufen.

3.2.2 Funktionalität für Investoren

- Investoren können sich über Benutzernamen und ein korrektes Password an der Anwendung anmelden.
- Weiter ist es möglich, dass sich Investoren auch neu an der Anwendung registrieren können. (Daten werden dann in der Datenbank abgelegt.)
- Investoren können sich auch wieder deregistrieren (löschen)
- Nach erfolgreicher Anmeldung bzw. Registrierung werden dem Investor seine Portfolios angezeigt.

- Der Investor kann existierende Portfolios verkaufen, was der Einfachheit wegen durch ein Löschen realisiert werden soll.
- Der Investor kann neue Portfolios anlegen, wobei jedem Portfolio ein (registrierter) Broker zugeordnet werden muss. Hierzu könnten z.B. alle Broker als Liste zur Auswahl angeboten werden. Ist der Broker festgelegt, können Assets des Brokers in das Portfolio übernommen werden. (Ein Portfolio kann nur Assets des ihm zugeordneten Brokers enthalten.)
- Investoren können sich von der Anwendung auch wieder abmelden.
- Nur angemeldete Investoren können die REST-Schnittstelle aufrufen.

3.2.3 Architekturüberblick

Die HTML-Seite und CSS- und JavaScript-Dateien werden auf einem Web-Server gehostet, die eigentliche Serveranwendung stellt eine REST-Schnittstelle zur Verfügung.

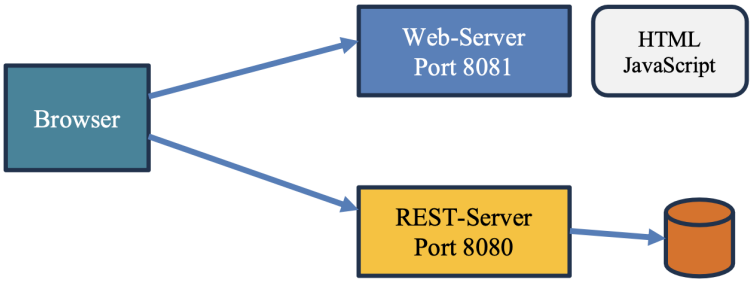

Abbildung 3: Verteilungsübersicht.

Es gibt keine Vorgaben für das Design der Web-Seite. Die einzige Anforderung ist, dass es sich um Single-Page-Anwendungen handelt, wobei hier jeweils eine für Broker und eine für Investoren implementiert werden kann. (Sie können auch beide Funktionalitäten in eine Single-Page-Anwendung packen.)
Es gibt keine ≪Doppelrollen≫, d.h. falls eine Person sowohl Broker als Investor ist, muss sie sich einmal als Broker und einmal als Investor registrieren.
Die lauffähige Anwendung besteht am Ende aus drei Docker-Container: Web-Server, REST-Server und DB-Server, die über ein Docker-Compose gestartet bzw. gestoppt werden.

3.3 Bewertungskriterium

Folgende Kriterien werden fü die Bewertung herangezogen:

- Funktionsumfang und Korrektheit der Umsetzung.
- Klassendesign und Codestrukturierung. Der Klassenentwurf sollte gängige Designprinzipien berücksichtigen.
- Korrekte Implementierung des ≪Zugriffsschutzes≫der REST-Schnittstelle.
- Verwendung von JPA und korrektes Mapping der Beziehungen

- Trennung von Layout und HTML und sinnvoller Einsatz von CSS. Die HTML-Seite sollte möglichst keinen JavaScript-Code enthalten.
- Verständlicher Code (insbesondere bei JavaScript und Java). Beachten Sie die Regeln von *Clean Code*.
- Paketierung der Anwendung in Docker-Container
- Präsentation. Am Ende muss jeder Teilnehmer seine fertig gestellte Anwendung vorstellen und gegebenenfalls Fragen dazu beantworten. (Termine werden auf der OLAT-Seite veröffentlicht.)

Abzugeben ist der komplette Projektcode und alle entstandenen Artefakte.

3.4 Erklärung zur Ausarbeitung

Weiter ist eine Erklärung abzugeben, die folgenden Wortlaut hat:

> Hiermit erkläre ich, *Vorname Nachname (Matrikel)*, dass ich die von mir abgegebenen Artefakte im Modul Software-Technik Projekt selbstständig und ohne fremde Hilfe angefertigt habe und keine anderen als in der Abhandlung oder im Source angegebenen Hilfen benutzt habe; dass ich die Übernahme von Source-Code, wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der angegebenen Artefakte gekennzeichnet habe. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.
>
> _____
>
> Unterschrift

Die Abgabe der unterschriebenen Erklärung kann in Form einer Scann-Kopie erfolgen und muss ebenfalls über OLAT abgegeben werden.

3.5 Die Abgabe und Prüfung

Verpacken Sie Ihre Anwendung, einschließlich Erklärung in eine Zip-Datei. Die Erklärung bitte ins Root-Verzeichnis legen. Stellen Sie sicher, dass die Zip-Datei keine generierten Sourcen (target-Verzeichnis) enthält. (Ansonsten gibt es Punktabzug!) Benutzen Sie für die Zip-Datei folgende Namenskonvention:

```
SWTP_<<Nachname>>_<<Vorname>>.zip
```

Wichtig: Achten Sie darauf, dass alle Sourcen enthalten sind.

Die Abgabe erfolgt über OLAT. Sie finden einen entsprechenden Abgabeordner auf der OLAT-Seite des Moduls.