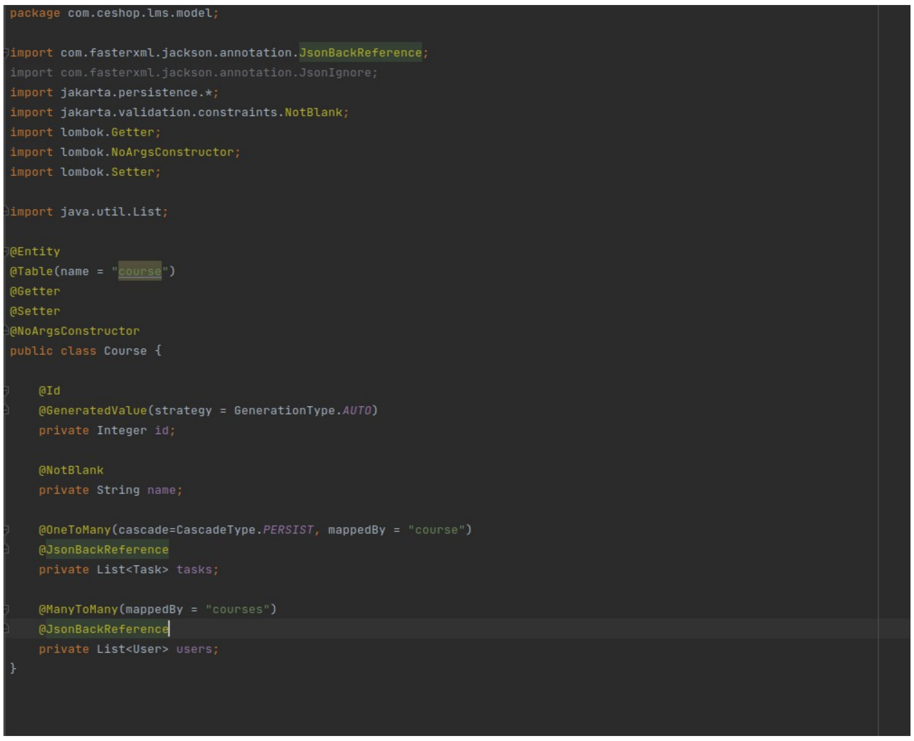
Project 1

Each course has questions, and Users answer the questions. So we have one Many To One Relationship between Question and Course tables, also we have a Many To Many Relationships between Question and User tables. With this approach, a question is completed when is saved to a specific user. To calculate the percentage of how much a course is completed, we use a simple formula that counts questions of a course and also counts the questions completed by the user for that specific course. The product of answered questions multiplied by 100 and divided by the total amount of questions for that specific course will show the progress during the course.Each course has questions, and Users answer the questions.

For this example, without more detailed information, I choose to use Spring Boot on the back end, angular with typescript and bootstrap on the front end, and H2 database to store data. I would look further for another database in a production environment, like PostgreSQL or MySql.For this example, without more detailed information, I choose to use Spring Boot on the back end, angular with typescript and bootstrap on the front end, and H2 database to store data. I would look for another database in a production environment, like PostgreSQL or MySql.

To access the database, I choose to use JPA and ORM. Here is an example of an entity that maps Course:To access the database, I choose to use JPA and ORM. Here is an example of an entity that maps Course:



```java
package com.ceshop.lms.model;

import com.fasterxml.jackson.annotation.JsonBackReference;
import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.*;
import jakarta.validation.constraints.NotBlank;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.List;

@Entity
@Table(name = "course")
@Getter
@Setter
@NoArgsConstructor
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Integer id;

    @NotBlank
    private String name;

    @OneToMany(cascade=CascadeType.PERSIST, mappedBy = "course")
    @JsonBackReference
    private List<Task> tasks;

    @ManyToMany(mappedBy = "courses")
    @JsonBackReference
    private List<User> users;
}
```

On the configuration provided in the code attached, tables and databases are created on the fly by Spring JPA. The connection between the database and the application is managed by Spring JPA. The DAO layer is implemented by JPA Repository. On the configuration provided in the code attached, tables and databases are created on the fly by Spring JPA. The connection between the Database and the Application is managed by Spring JPA. The DAO layer is implemented by JPA Repository.

The integration between the front-end and back-end would be built with microservices. The security layer would be built with Spring Security, JWT. In the POC provided, you can dig for further details.The integration between the front-end and back-end would be built with microservices. The security layer would be built with Spring Security, JWT. In the POC provided, you can dig into more details.