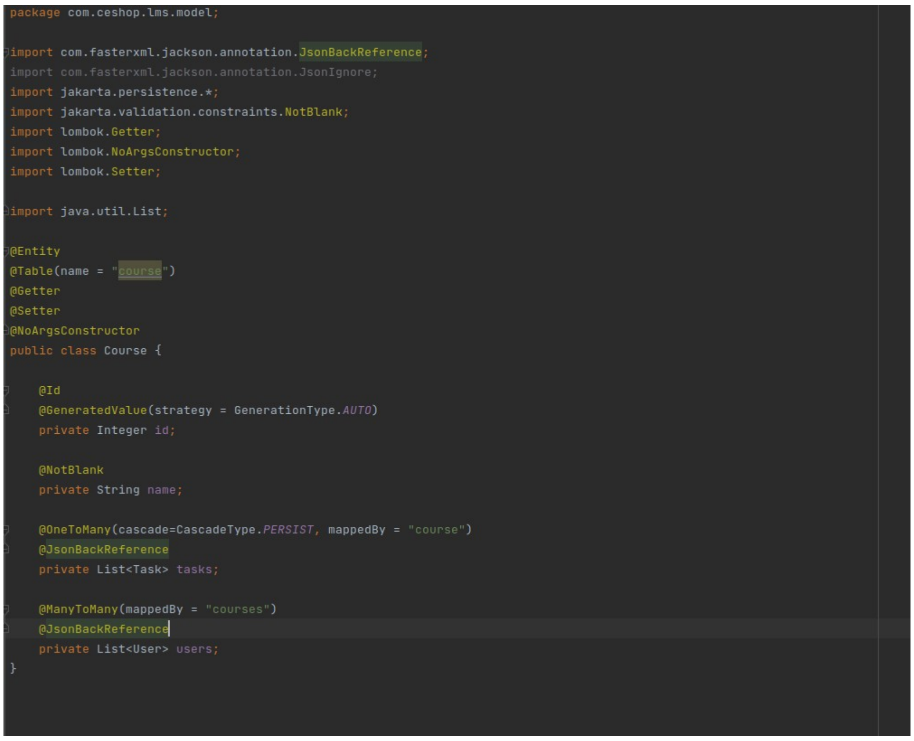
Project 1

Each course has questions, and Users answer the questions. So we have one Many To One Relationship between Question and Course tables, and a Many To Many Relationships between Question and User tables. A question is completed when 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 course will show the progress during the course. Each course has questions, and Users answer the questions.

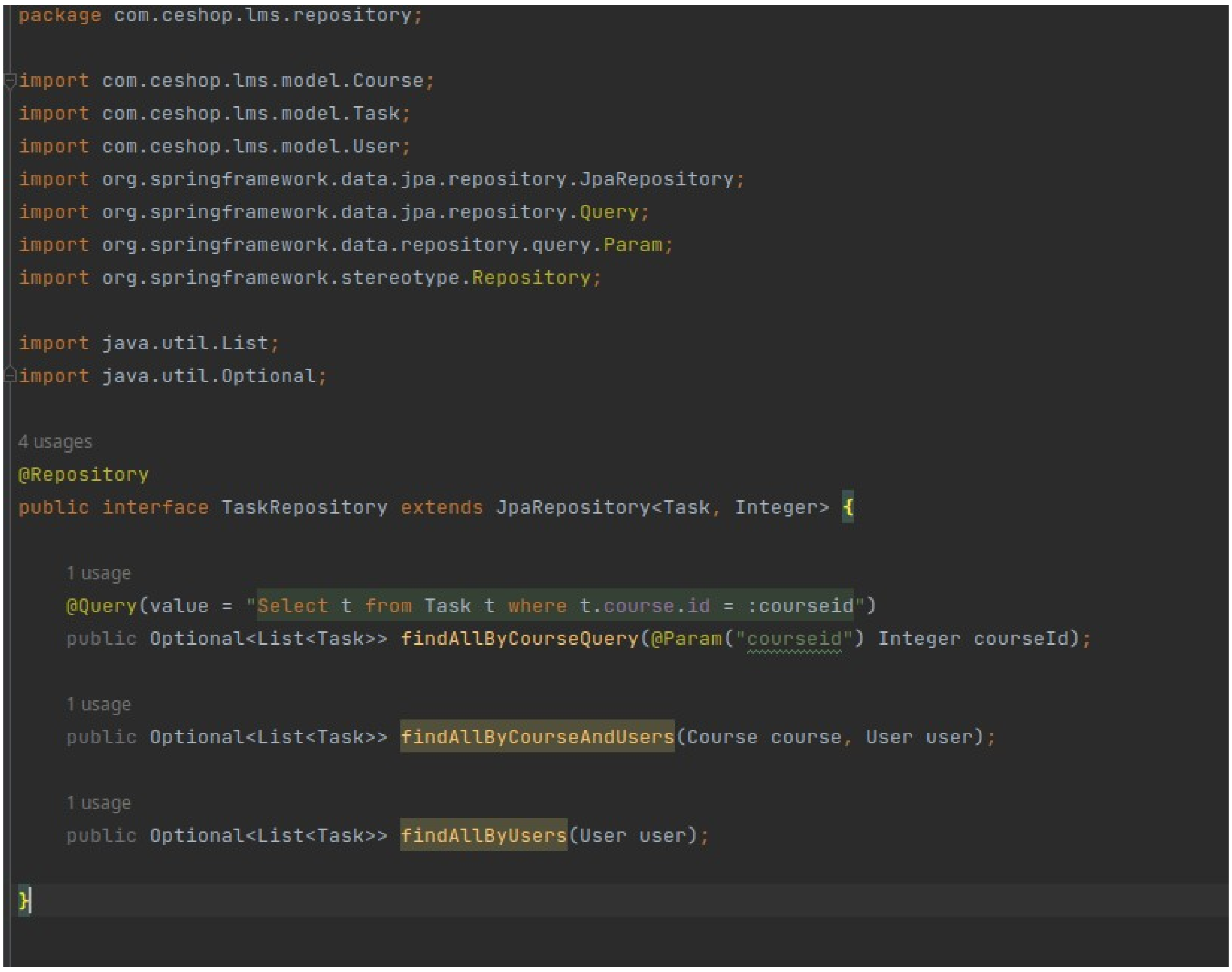
For this example, without more detailed information, I choose to use Spring Boot on the back end, angular with typescript and bootstrap on the front end, and H2 database to store data. I would look for another database in a production environment, like PostgreSQL or MySql.

To access the database, I use JPA and ORM. Here is the entity that maps Course:To access the database, I use JPA and ORM. Here is the entity that maps the 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 database connection is managed by Spring JPA, and the DAO layer uses JPA Repository that provides functions to query the database. Queries will be built on runtime by Spring JPA, based on method signature, and also can be customized like in the example bellow:



```
package com.ceshop.lms.repository;

import com.ceshop.lms.model.Course;
import com.ceshop.lms.model.Task;
import com.ceshop.lms.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;

4 usages
@Repository
public interface TaskRepository extends JpaRepository<Task, Integer> {

    1 usage
    @Query(value = "Select t from Task t where t.course.id = :courseid")
    public Optional<List<Task>> findAllByCourseQuery(@Param("courseid") Integer courseId);

    1 usage
    public Optional<List<Task>> findAllByCourseAndUsers(Course course, User user);

    1 usage
    public Optional<List<Task>> findAllByUsers(User user);

}
```

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.