Documentation

Assignment 1

# Introduction

This application is a simple version of StackOverflow. It is implemented with java spring and I use the mariaDB which is a mySQL fork.

Key features: It allows users to ask questions, post answers, and vote on questions and answers.

- Only registered users can perform actions on the system, and passwords are encrypted for security.

Technology

My application is built using Java Spring and uses MariaDB as the database management system. I am using IntelliJ IDEA as my Integrated Development Environment (IDE).

# Use case diagrams

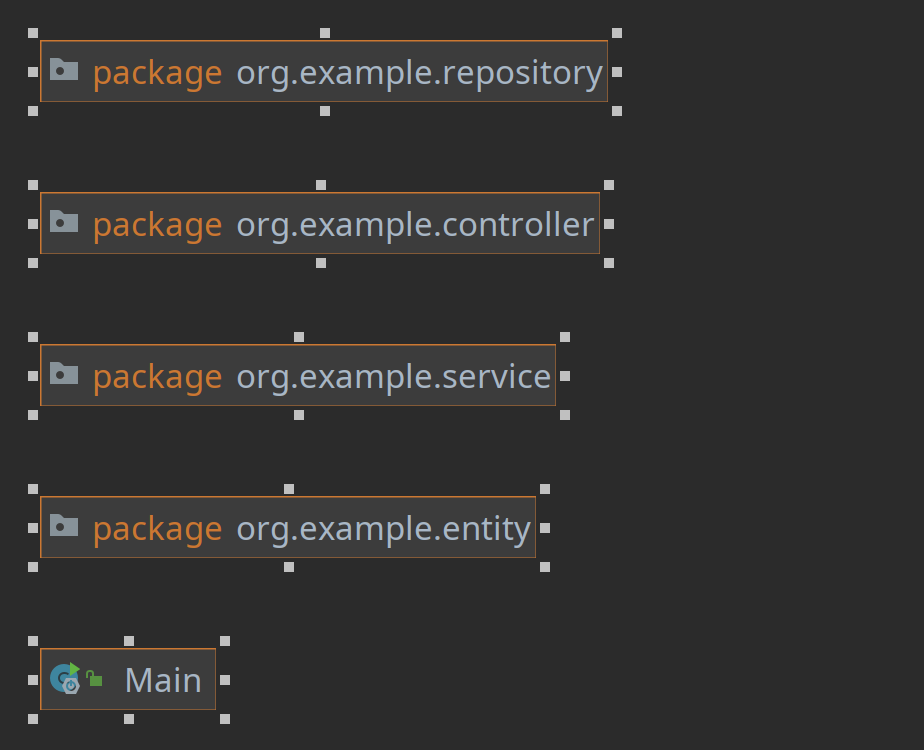
A use case diagram specifies the expected behavior of the application features and is usually used while developing the application while in its begging stages and design.

In this application so far there are the following use cases:

* Register: A user can create an account on the system.
* Login: A user can authenticate the system.
* Ask Question: A user can create a new question.
* Edit Question: A user can modify a question they have previously created.
* Delete Question: A user can delete a question they have previously created.
* Answer Question: A user can post an answer to a question.
* Edit Answer: A user can modify an answer they have previously posted.
* Delete Answer: A user can delete an answer they have previously posted.
* Vote: A user can vote on a question or answer.

# Package diagram

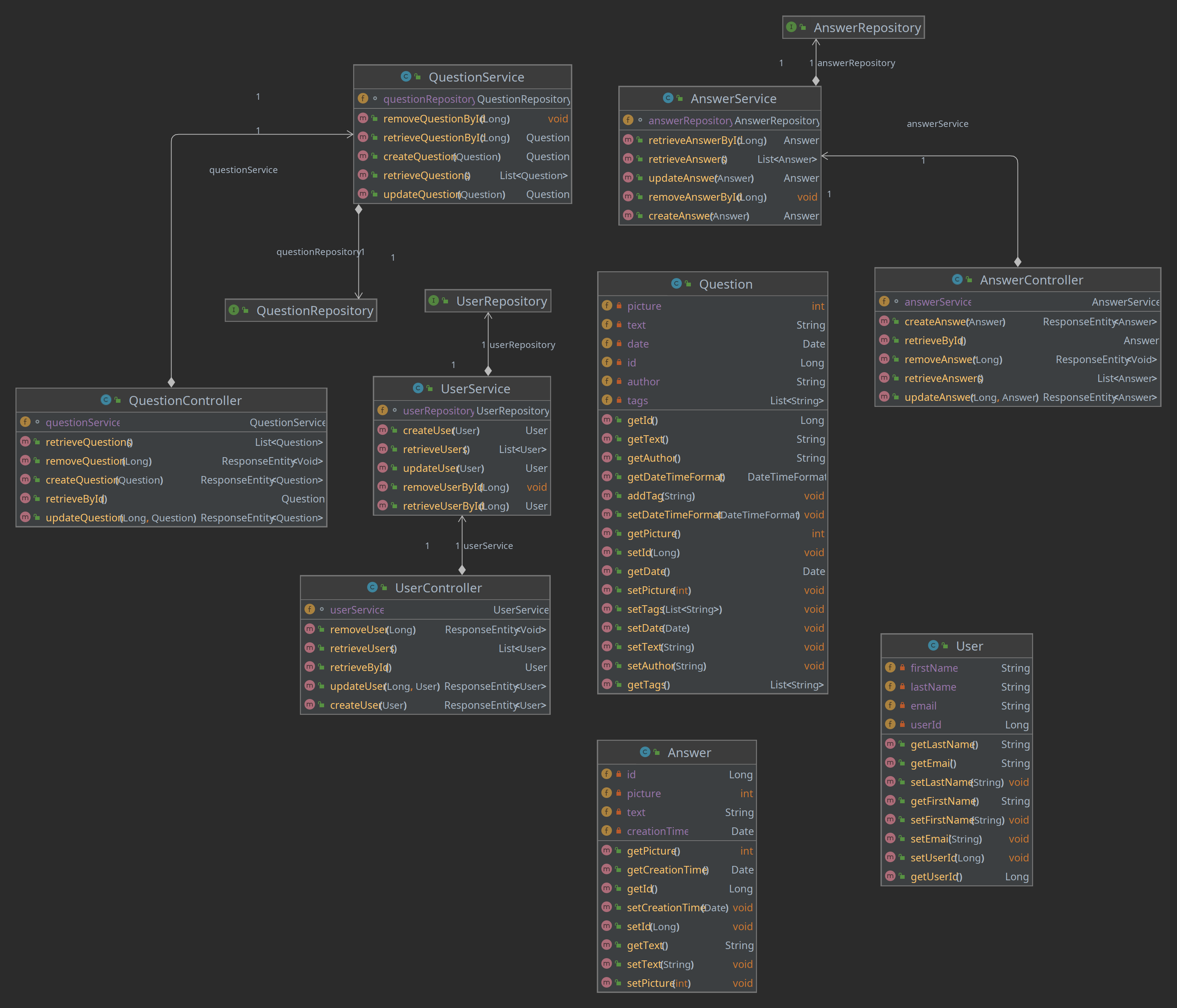
Here are all the packages present in the layered architecture.



# Class diagram

* User: Represents a user of the system.
* Question: Represents a question posted by a user.
* Answer: Represents an answer posted by a user.

Each class has attributes such as name, email, password, text, creation date, picture, tags, vote count, and relationships with other classes such as one-to-many between User and Question, one-to-many between Question and Answer, and many-to-one between Answer



Architecture

This Java Spring application follows a layered architecture pattern, which is a widely used design pattern in software development. The layered architecture pattern divides the application into distinct layers, with each layer performing specific tasks and communicating with the adjacent layers.

My layers are:

1. Entity Layer
2. Repository Layer
3. Service Layer
4. Controller Layer

These are the layers and their responsibilities:

**Entity Layer:** The entity layer is responsible for defining the domain models of the application. In this case, the User, Question, and Answer entities are defined in this layer. These entities represent the core data structures of my application and encapsulate the business logic that operates on the data.

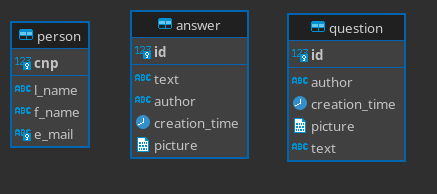
**Repository Layer:** The repository layer provides an abstraction over the data persistence layer, enabling my application to interact with the database. The repositories encapsulate the database operations such as Create, Read, Update, and Delete (CRUD) operations for the entities defined in the entity layer.

**Service Layer:** The service layer is responsible for implementing the business logic of my application. This layer serves as an intermediary between the repository layer and the controller layer. The service layer coordinates the data access and manipulations across multiple entities, implements the business rules, and ensures data integrity. The services in this layer provide high-level operations that can be called by the controller layer.

**Controller Layer:** The controller layer is the entry point for my application's API endpoints. It receives HTTP requests from the client and returns HTTP responses. The controller layer maps the incoming request to the appropriate service layer method and returns the results back to the client in the form of a response.

* Overall, the layered architecture pattern provides a clear separation of concerns in my application, making it easier to maintain, test and extend. The clear separation between the different layers of the application allows for greater modularity, making it easier to swap out or modify individual components without affecting the rest of the system.

Database diagram



Endpoints

