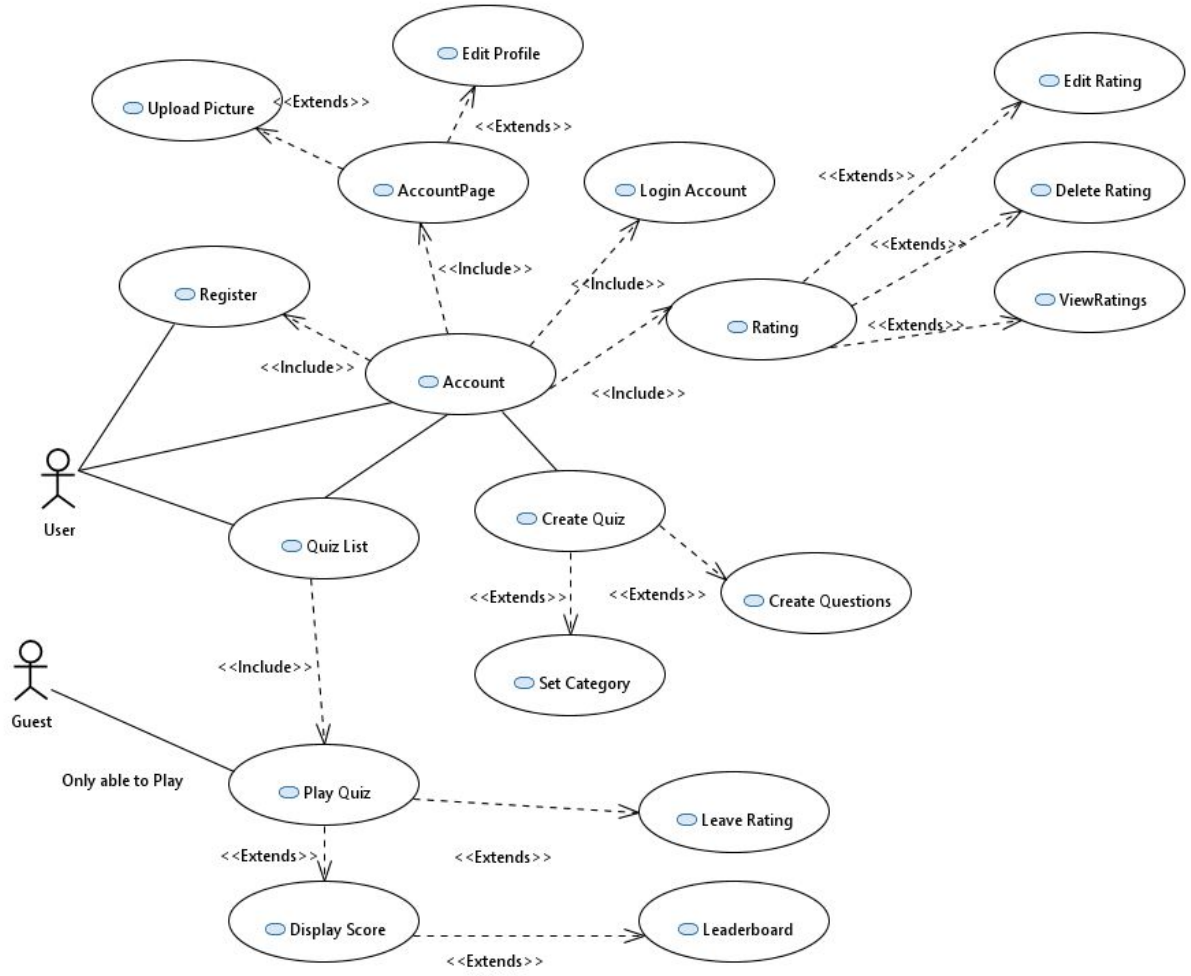
QuizApp - Group 9

Introduction

QuizApp is a web quiz app which takes inspiration from the famous kahoot! web page. In this app people can either play or create a quiz, write reviews on them and check whoever is on the leaderboard for each quiz.

User Stories



As a user I want to be able to do a quiz.
As a user I want to be able to search or filter a list of quizzes
As a user I want to be able to create my own quizzes.
As a user I want to see a leaderboard for quizzes I have finished.
As a user I want to see and edit my ratings.
As a user I want to add pictures to my quizzes.
As a user I want to be able to create an account.

As a user I want to be able to play without creating an account.
As a user I want to be able to choose a profile picture.
As a user I want to be able to rate the quiz I played.
As a user I want to see what rating a quiz has.

User Manual

This user manual assumes start from the home page.

Create Account
To create an account, press the “Create an account” link at the bottom right of the Login box. This will load the page where an account can be created. Fill in a unique username, an age, an email, and a password. The password has to be at least eight characters long with one upper case letter, one lowercase letter and one digit. To register the account press the button “Register” which will create the account and navigate to the start page where the account can be used to log in. To go back to the start page without creating the account, press the link “Home”.

Login as User
After having created an account, the user can log in via the log in form on the homepage with username and password. After logging in the webpage will navigate to the account page.

Play as guest
To play as a guest press the button “Play as guest” on the start page. This will load a list of quizzes. In guest mode quizzes can be played but the player will not end up in the leaderboard and the player can not leave a rating. A guest can not create its own quiz and when pressing “My account” in the navigation bar will load the start/login page, and not the account page.

Account Page
After logging in the account page will be shown. The account page displays account information. Here navigation buttons for various pages are presented. “Go to quiz list” leads to the page where quizzes can be played. “Account settings” leads to a page where account settings can be changed. “My ratings” leads to a page displaying the ratings made by the currently logged in user. “Logout” logs out the user and navigates to the start page. “Create New Quiz” leads to a page where the user can create a quiz.

Create quiz
To reach the page to create a quiz, go to the account page and press “Create new quiz”. To create a quiz, a few steps needs to be followed. Start by adding the first question by filling in the Question text, then the options. After filling in the options the correct answer needs to be filled in. The correct answer should be written as 1 for Option1, 2 for Option2 and so on. After this the question can be added to the quiz by pressing “Add question”. After this another question can be entered in the same manner. After all questions have been entered it is time to fill in the Quiz Title and to choose a fitting category for the quiz. Now press the “Create Quiz" button and the quiz will be created and playable from the list of quizzes.

Quiz list
In the list of quizzes the default quizzes and the quizzes created by users can be found. In the quiz list a certain quiz can be found by searching for the title in the title field, or by creator in the creator field. The quiz list can also be filtered by category or average score. To play a quiz, press the “quiz page” button for the desired quiz.

Play a quiz
A quiz can be played either as a guest or as a logged in user. After selecting a quiz in the list, the player is brought to a page displaying the quiz in more detail. On this page there is a button which the player can press to start the quiz. The player is then met with a question and four alternatives, as well as information about what question they are on, how many points they have and a timer ticking down. If the timer reaches zero the user will not get a chance to answer, and gets zero points. After pressing one of the answer buttons, the player will be notified about whether the answer was correct or incorrect. In order to move on to the next question the player presses the button “Next Question”. After answering the final question the player can now press “End Quiz” to end the quiz. The results are summarized and the player can (if logged in) see their result in the leaderboard. The player is also given the option to leave a rating by opening the rating dialogue with a button press. At this point the player can also go to their account or back to the quiz list, either by using the buttons or the quick links in the page header.

Design

Frontend-Libraries

Java Server Faces version 2.3

Primefaces version 10.0.0

Omnifaces version 3.10.1

Bootsfaces version 1.5.0

Argon2Password version 2.5

Backend-Libraries

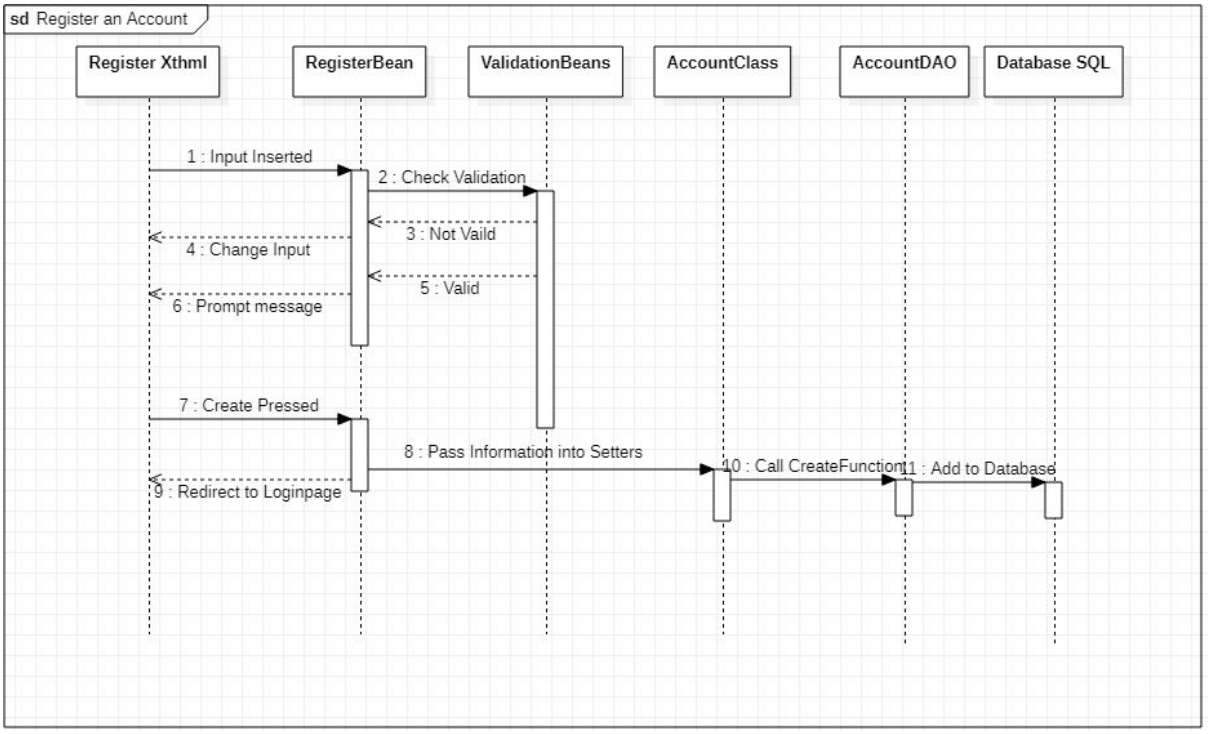
QueryDSL version 2.5.0

Lombok version 1.18.12

Junit version 4.13.1

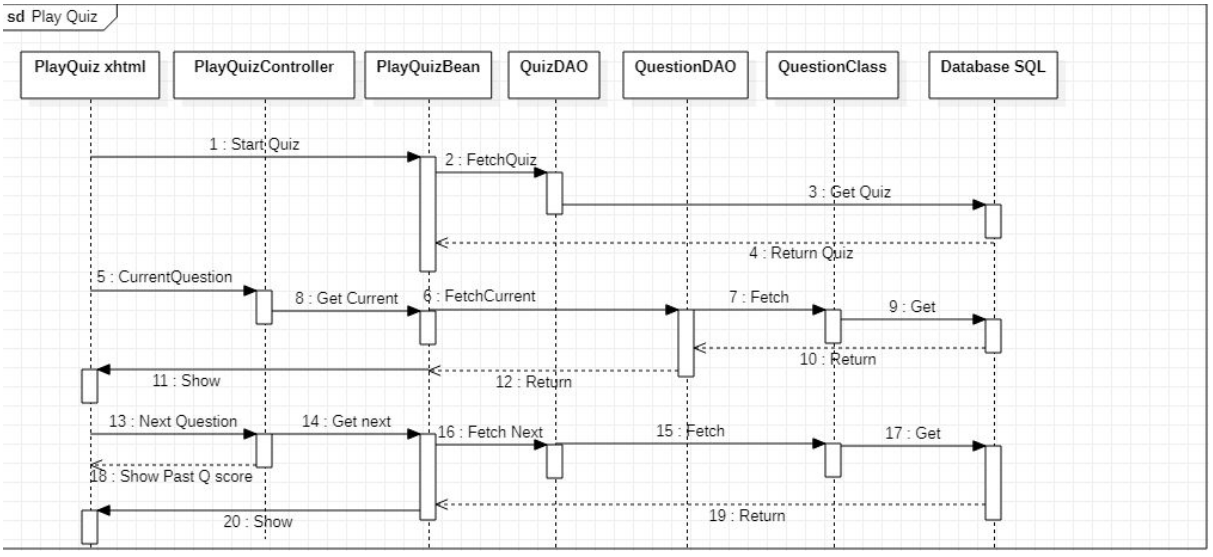
Application Flow

Sequence Diagram - Register an Account



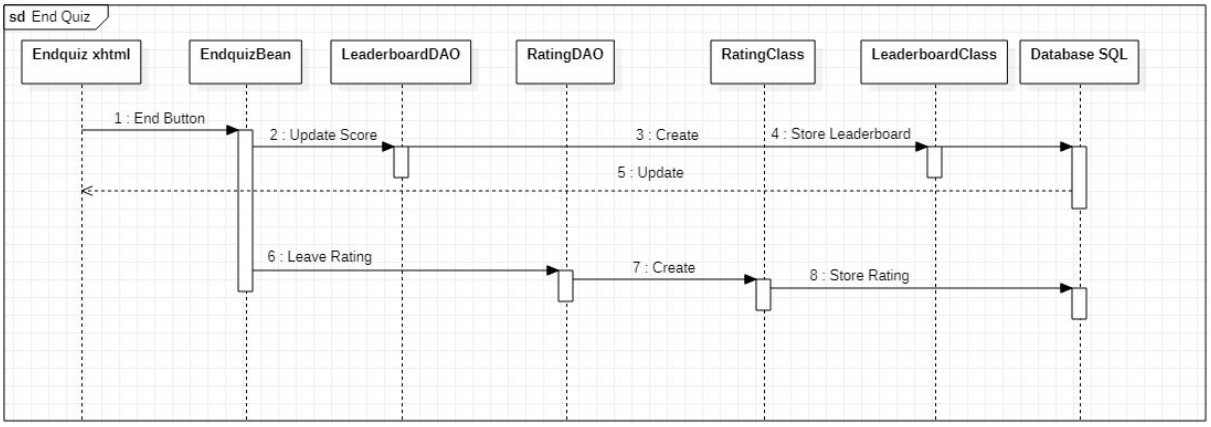
When registering an account from its web page, input is inserted from the register web page. Sent to the Registerbean which checks with the Validationsbeans if any of the input as email and username exists in the SQL database, if so sends back a message prompting the user to change its input to ensure no duplicates in the database. If the input is valid the create button will return valid which creates the account sending it through a function found in AccountDAO class and the persist the account within the SQL database. A redirect message from the Registerbean function sends the user to the login page.

Sequence Diagram - Play Quiz



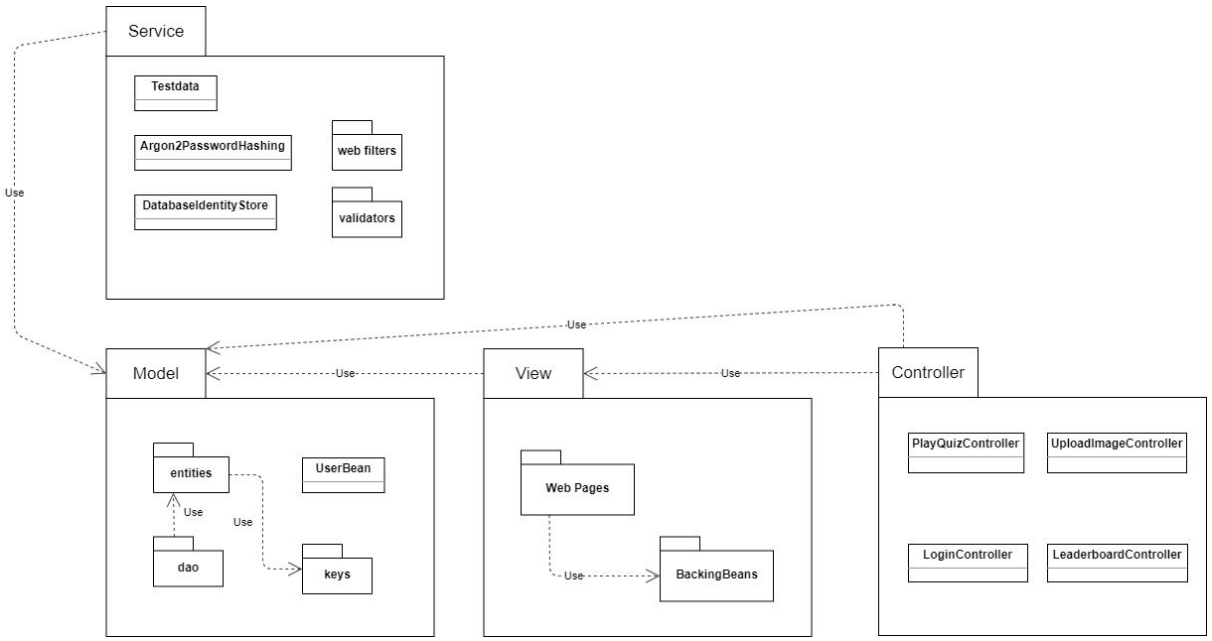
Play Quiz starts from its web page by calling its Playquizbean that fetch the quiz based on id sent as a parameter. This gets the quiz from the database and returns the quiz. Fetching the current question is sent to the bean that uses a function from QuestionDAO class that gets the right question from the database, returning the current question based on the quiz id. PlayQuizBean shows the current question of the quiz back on the web page. The PlayQuizController class calls on functions from Playquizbean when the button for “next question” is pressed. Which then increments a variable that holds controls over the current index of the list of questions in the quiz. Then the next question in the quiz is fetched from the database and displayed on the web page.

Sequence Diagram -End Quiz



Endquiz web page is instantiated by the end quiz button from Playquiz web page and gets quiz score and quiz id sent to it from the parameter sent by the button. This updates the leaderboard at the Endquiz page. On this page also users can leave a rating if they have an account and are logged in.

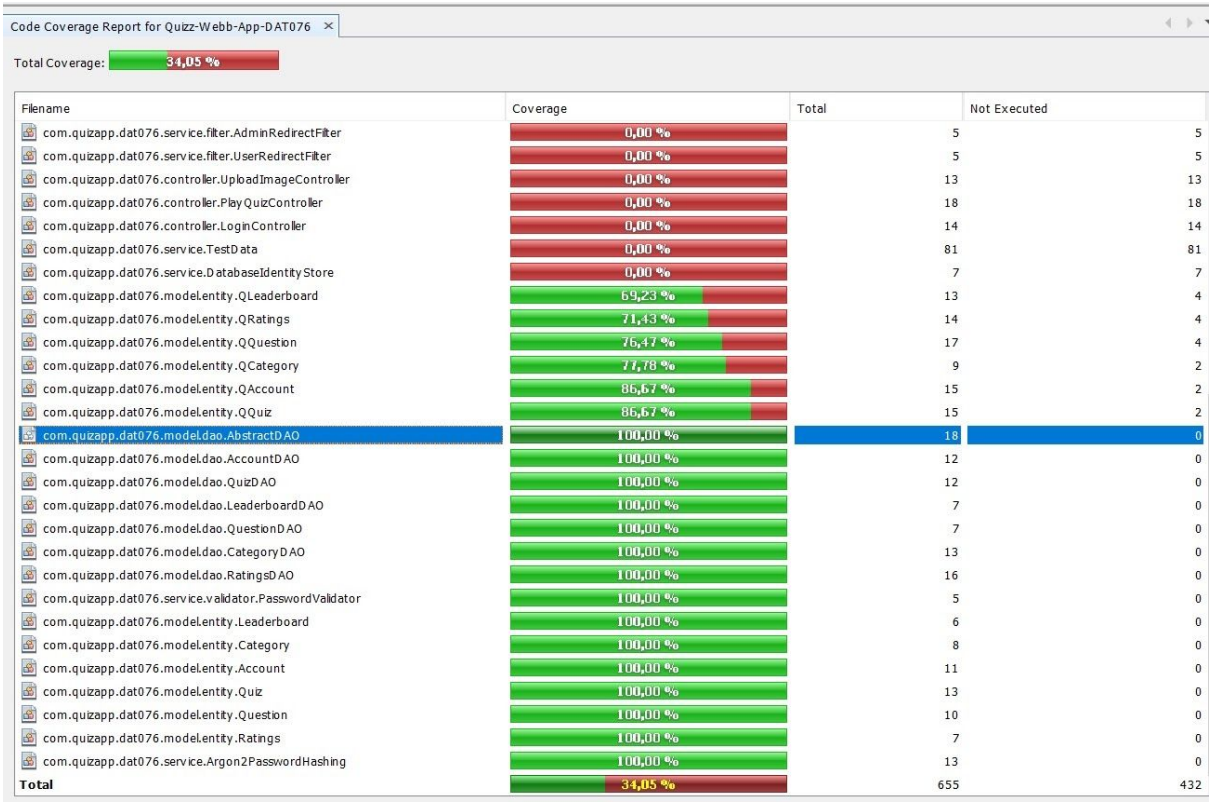
Package Diagram



The project largely follows the MVC-model. Entities, their keys and daos are included in the Model. Web pages and their viewscoped beans are part of the View, while more logically complex user interaction are placed in the Controller layer. A number of additional supporting classes are placed in the Service package, since they provide functionality that is not directly tied to any of the three MVC-packages.

Code Coverage Report

Due to difficulty with Arquillian Drone, there are no frontend tests on the application which results in a lower code coverage. All of the DAOs and entities however, have 100% coverage. Part of the reason for the low % is also because of the TestData class that has a lot of non executed data (see picture). We thought that testing that class in particular seemed unnecessary since it's just adding and creating different quizzes/accounts/categories etc which is already tested in the entity/entitydao tests.

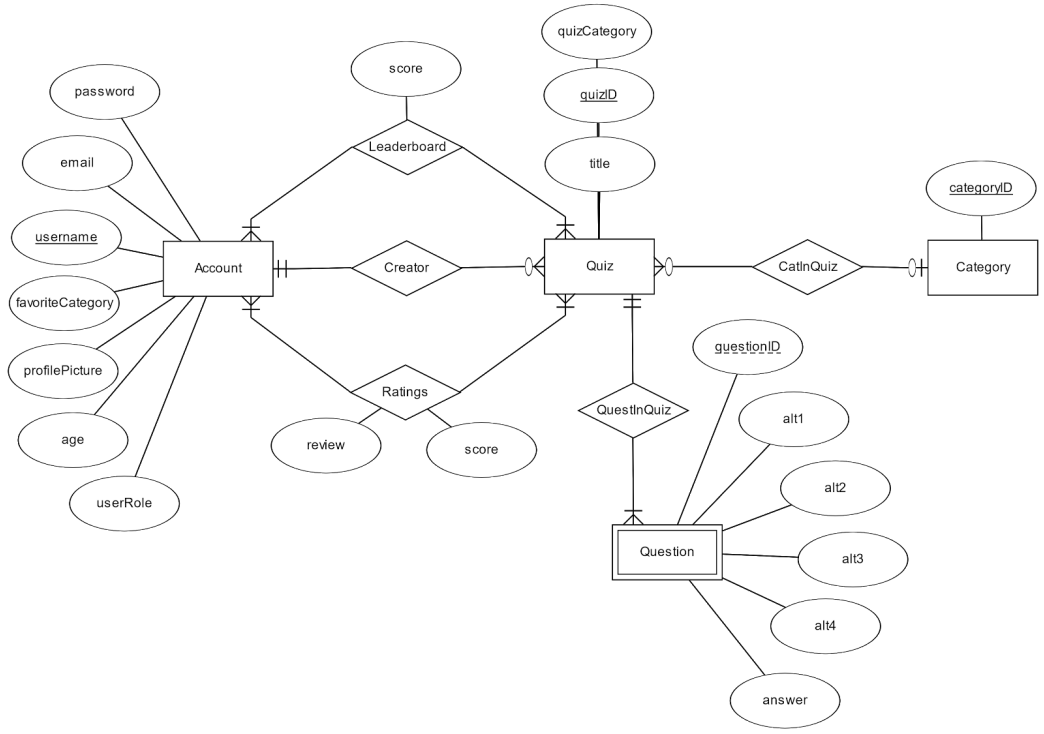


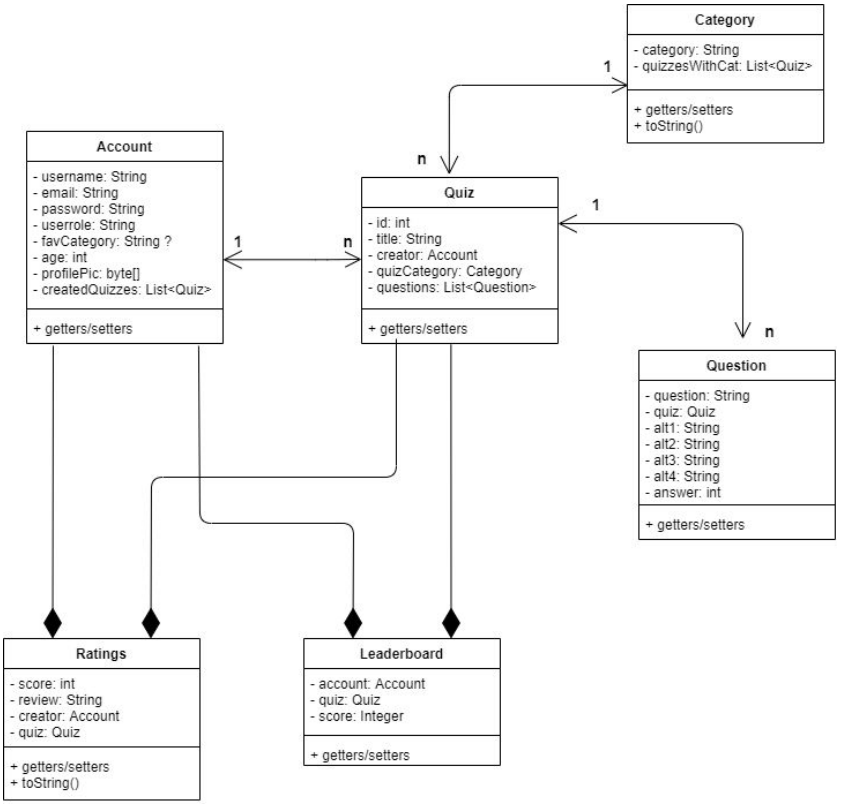
Code Coverage Report for Quizz-Webb-App-DAT076

Total Coverage: 34,05 %

Filename	Coverage	Total	Not Executed
com.quizapp.dat076.service.filter.AdminRedirectFilter	0,00 %	5	5
com.quizapp.dat076.service.filter.UserRedirectFilter	0,00 %	5	5
com.quizapp.dat076.controller.UploadImageController	0,00 %	13	13
com.quizapp.dat076.controller.PlayQuizController	0,00 %	18	18
com.quizapp.dat076.controller.LoginController	0,00 %	14	14
com.quizapp.dat076.service.TestData	0,00 %	81	81
com.quizapp.dat076.service.DatabaseIdentityStore	0,00 %	7	7
com.quizapp.dat076.model.entity.QLeaderboard	69,23 %	13	4
com.quizapp.dat076.model.entity.QRatings	71,43 %	14	4
com.quizapp.dat076.model.entity.QQuestion	76,47 %	17	4
com.quizapp.dat076.model.entity.QCategory	77,78 %	9	2
com.quizapp.dat076.model.entity.QAccount	86,67 %	15	2
com.quizapp.dat076.model.entity.QQuiz	86,67 %	15	2
com.quizapp.dat076.model.dao.AbstractDAO	100,00 %	18	0
com.quizapp.dat076.model.dao.AccountDAO	100,00 %	12	0
com.quizapp.dat076.model.dao.QuizDAO	100,00 %	12	0
com.quizapp.dat076.model.dao.LeaderboardDAO	100,00 %	7	0
com.quizapp.dat076.model.dao.QuestionDAO	100,00 %	7	0
com.quizapp.dat076.model.dao.CategoryDAO	100,00 %	13	0
com.quizapp.dat076.model.dao.RatingsDAO	100,00 %	16	0
com.quizapp.dat076.service.validator.PasswordValidator	100,00 %	5	0
com.quizapp.dat076.model.entity.Leaderboard	100,00 %	6	0
com.quizapp.dat076.model.entity.Category	100,00 %	8	0
com.quizapp.dat076.model.entity.Account	100,00 %	11	0
com.quizapp.dat076.model.entity.Quiz	100,00 %	13	0
com.quizapp.dat076.model.entity.Question	100,00 %	10	0
com.quizapp.dat076.model.entity.Ratings	100,00 %	7	0
com.quizapp.dat076.service.Argon2PasswordHashing	100,00 %	13	0
Total	34,05 %	655	432

Model diagram





Responsibilities

Albin: Created the initial java entity classes based on the ER-diagram (Account, Quiz, Question, Category). Main responsibility was quiz-related functionality. Set up the QuizDAO and the QuizDAO-tests. Worked on the functionality and visuals for displaying a list of available quizzes, and being able to sort/filter the list. Added a page for displaying a quiz in more detail and from where the player can start the quiz. Worked on the functionality and visuals for playing a quiz, i.e. being able to answer questions, indication for whether the answer was correct or incorrect, a time limit and so on. Also added a page header as a template for all pages.

Rebecka:
Created the many to many entity Leaderboard and the DAO and DAO test for it. Created the leaderboard as an xhtml and later made it into a component. Also created the backingbean for leaderboard. Made the css for the leaderboard which is in quiz.css. Also implemented the component into quiz.xhtml and endquiz.xhtml. Created the functionality on the end quiz button where the current player with its score is added to the leaderboard. Created the category DAO and DAO test. Moved the css from template to a .css file which proved more difficult than imagined. Created a file for test data.

Emma:
Has been implementing the basic logic behind the register page and creating quizzes, including the Account DAO and beans for the .xhtml-files. Created the validations for the username and email fields. Was also responsible for improving the code coverage report and improving tests for the model, making sure all DAOs and entities were 100% covered. Has also been working on the overall styling of the app, moving all styling from the .xhtml-files to .css-files and added some basic styling to several pages.

Anton B:
Implementing logic behind the ratings entity and functionality connected with it. This involved extending the ratingsDAO class with corresponding beans classes for functionality. Implemented functions allowed for adding/displaying/manipulation of this entity in different pages like quizlist, my ratings show quiz and end quiz pages. Also built the logic and functionality behind the end quiz page allowing different buttons being disable based on the user having an account or being a guest. Helped with the basic layout of the playquiz page while Albin created all the functionality behind it and with its corresponding classes and the rest of the styling.

Anton Ekman
My first major role in any part of the project was the implementation of the homepage. The homepage's most crucial part was the login form that would handle the whole login functionality. I implemented a simple login that would check the email and password against the database. Later I would upgrade this login twice. Firstly I implement a database identity store that utilizes security context to validate user credentials. Secondly I implemented the password hashing algorithm Argon2 for even more security and to importantly not save passwords in the database as plaintext. Beyond the login functionality I also implemented an

account page for each user. This led me to create a user bean (UserBean.java) that would be the sessionscoped source throughout the application to get all the information about the user when needed as well as to check whether someone was logged in or just a guest. To accountpage were to display information about the user so I add 3 more attributes to a user: Age, favorite quiz category and a profile picture. In the account page and also a setting page for the account you could then change your age or your favorite category as well as upload any picture(with restrictions in mind) from your computer as your profile picture. I also contributed to the create quiz functionality that let's a user create their own quiz. Implemented the DAO for the Question entity. Obviously I had my hands in a lot of other things but these parts in particular I did major/all of the work for.