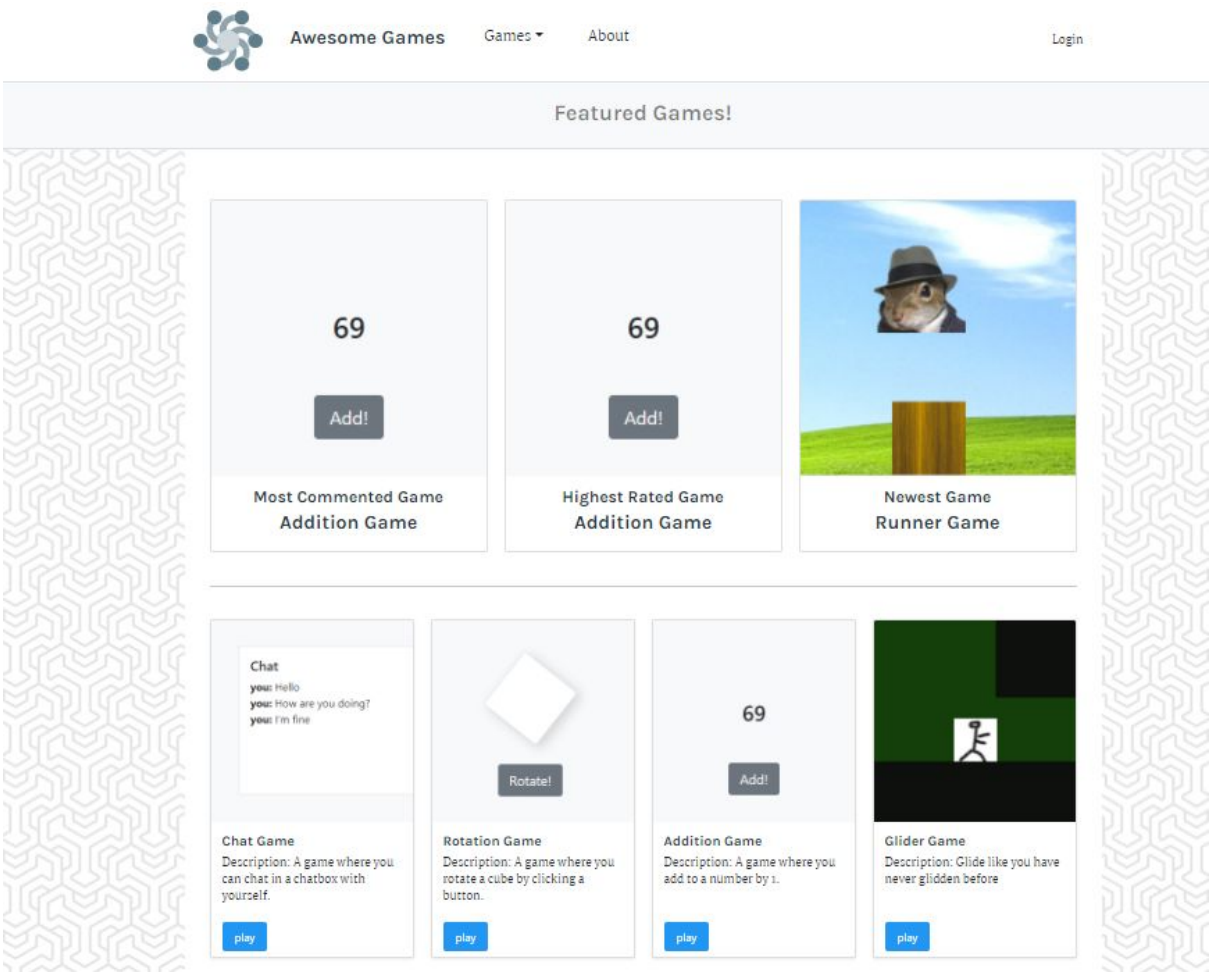
Awesome Games



DAT076 / DIT126 Web applications
Group 10
Tobias Mauritzon
Matteus Straznyk
David Andersson
Joachim Antfolk
William Jönsson
Simon Terling
Project Supervisor: Matthías Páll Gissurarson

1. Introduction

This web-app is a social game platform. You can play, rate, submit your high scores on games and chat about it in the comments. The site features "users" where all your high scores and comments will be stored. If you change your mind about a rating you can update or delete it. If you regret or posted a comment by mistake, you can always delete it and post it again. The source code can be found on Github.

2. Use cases

Implemented use cases:

1. Users can create an account
2. Users can login to an existing account
3. Users can rate games if logged in
4. Users can change their rating if logged in
5. User can remove rating if logged in
6. Users can leave comments on games if logged in
7. Users can remove their comment if logged in
8. Users can submit high scores
9. Admins can remove everyone else's comments
10. Everyone/ Users can play games
11. Everyone/ Users can navigate the site
12. Everyone/ Users can read about us
13. Everyone/ Users can sort comments
14. Everyone/ Users can read comments
15. Everyone/ Users can see top rated game
16. Everyone/ Users can see most commented game
17. Everyone/ Users can see newest game
18. Everyone/ Users can see average rating on a game

Next Steps if we would have had more time:

1. Users can edit comments if logged in
2. Users can add games
3. User have their own user page
4. Users can change their password/user name
5. Users can upload a profile image, which is shown on the site
6. More Games
7. Multiplayer Games

3. User manual

Navigating:

When you first load the page you can see our games displayed in cards on the main page, on the cards you can read about the game and you can click the card to navigate to the game page. On all pages in the top right corner you can press the login text to sign in or create a new account. You can navigate to the games either by the card on the main page or the drop down menu in the header which is available on every page. If you are on a game page and want to return to the main page you can press the text or logo in the top left corner.

Playing a game:

The chosen game is loaded along with the game page. Some games start directly while others require the user to press a start button. After the game has been loaded (and started) the user can play the game.

Rating & Commenting:

Once a user has logged-in you can comment and rate games on it’s game page. A user can leave as many comments as they want and they can sort the comment section in either ascending or descending order of time posted. You can rate a game as many times as you want but only the last rating is counted. As a user rates a game, it is counted towards the game’s average rating which is displayed over the user rating.

Highscore:

After a user has played a game (which supports highscore) they can submit their highscore on the game page. If the high score is in the top ten it is displayed on the game page.

Login:

On the top right on each page there is a login button, if pressed it takes the user to the login page. There the user can log in or choose to create a new account. If the user decides to make a new account they are redirected to the create user account page where they can fill in their information and register to the database.

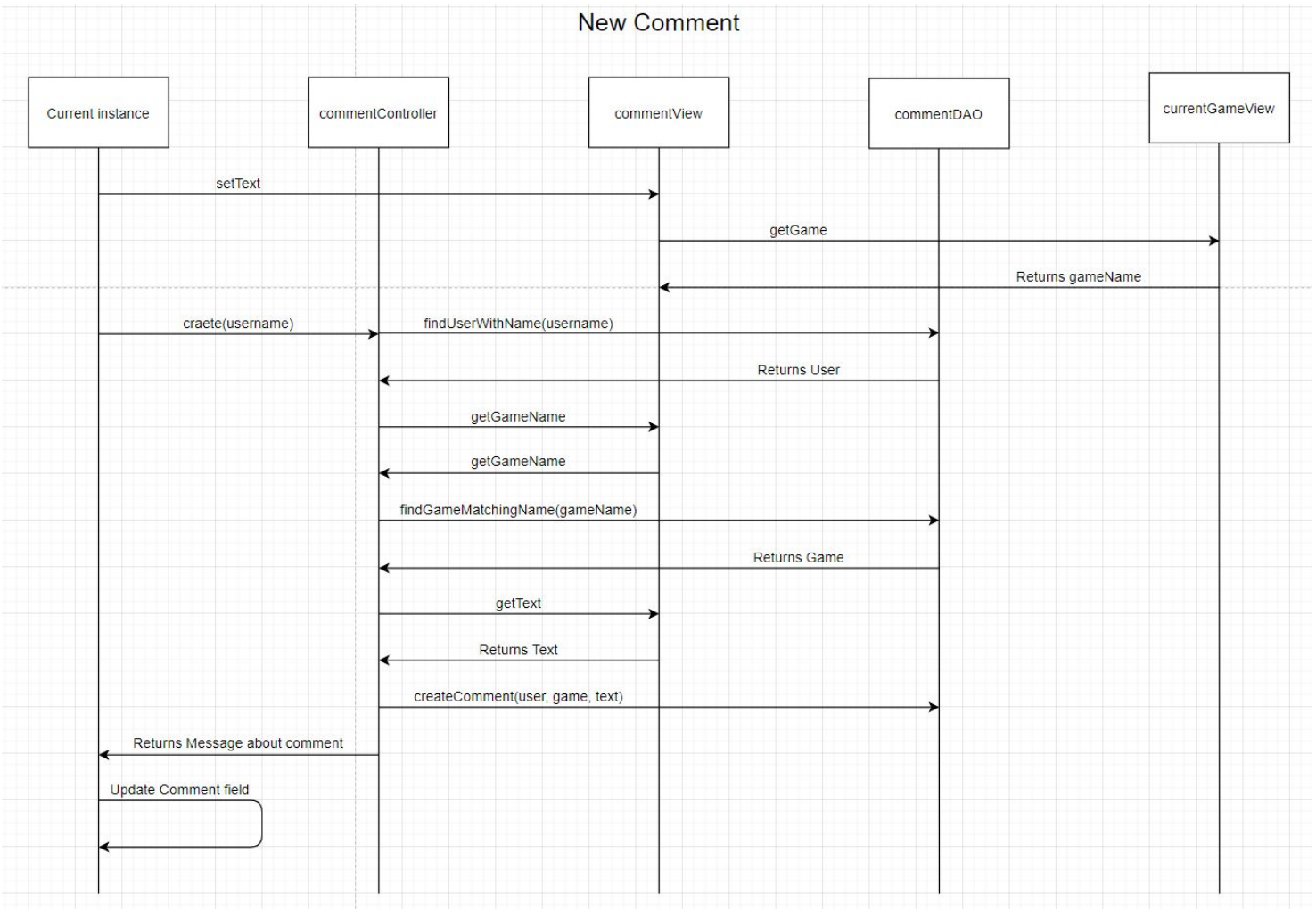
4. Design

Bootstrap - Used for big “picture” styling.
Primefaces - Used for its components.
JSF - Frontend to backend communication.
Arquillian - Testing library.
Javascript - Only used for our games.
Java DB & Derby - Storage of entities.
Git/github - The way we shared code and collaborated.
CSS - Used if bootstrap was not enough and we needed more fine styling.
Lombok - Library that creates getters, setters and help with storing data.
Maven - Library/package manager.
Jquery - The preferred way of searching for DOM elements.
Jacoco - Needed for the code coverage report.
Java - The backend language used for the project.
Mockito - Another testing library.

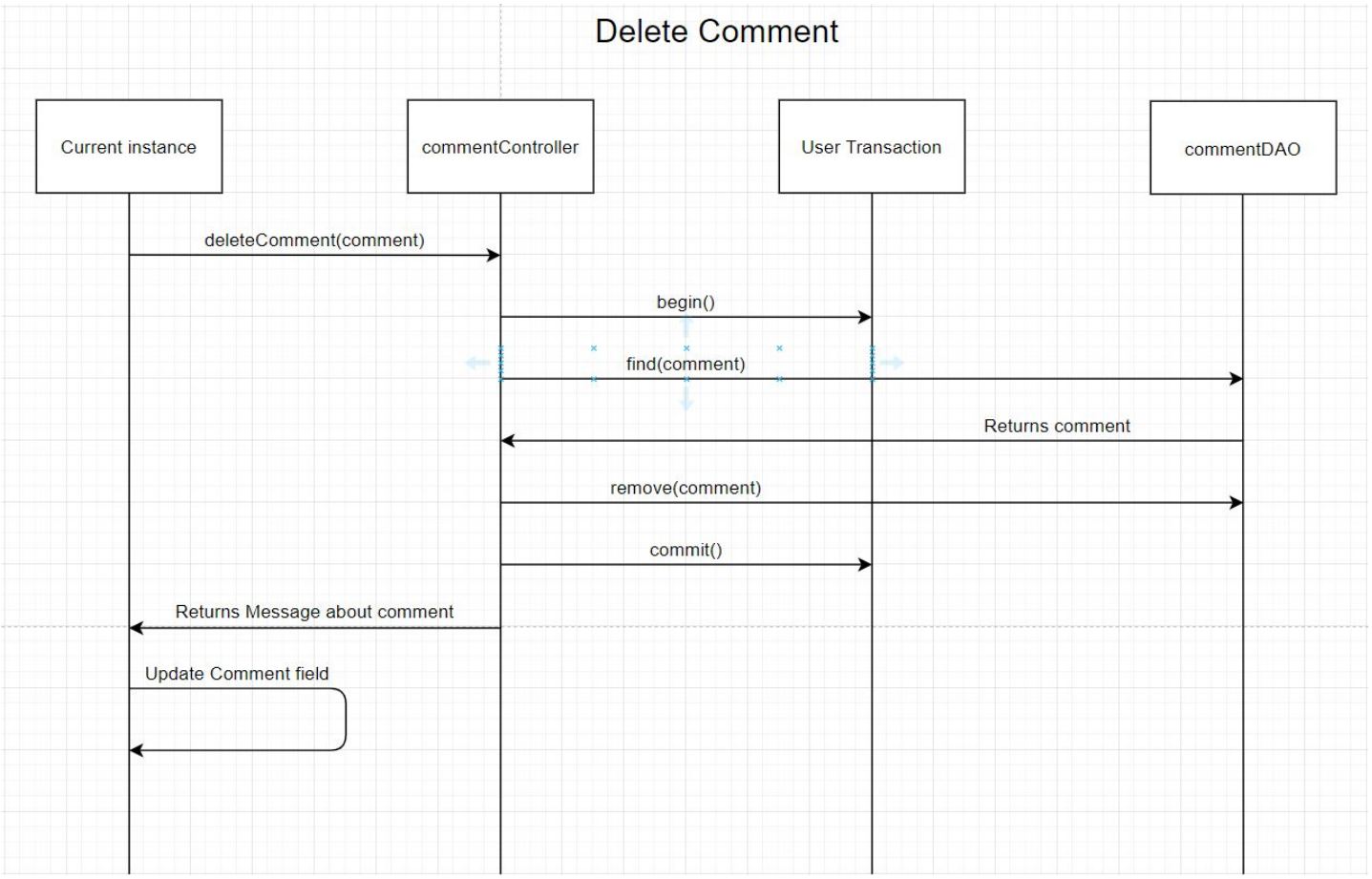
Bootstrap, Primefaces, CSS, Javascript and Jquery are the things used by the frontend.
Java, JSF and Lombok handle backend and communication.
Jacoco, Arquillian and Mockito are used for unit testing, other tests and code coverage.
Java DB & Derby are used for our database.
Git/github and Maven are used for handling libraries, dependencies, sharing code and making collaborating easier.

We followed the MVC pattern and the lab instructions in the creation of the project.

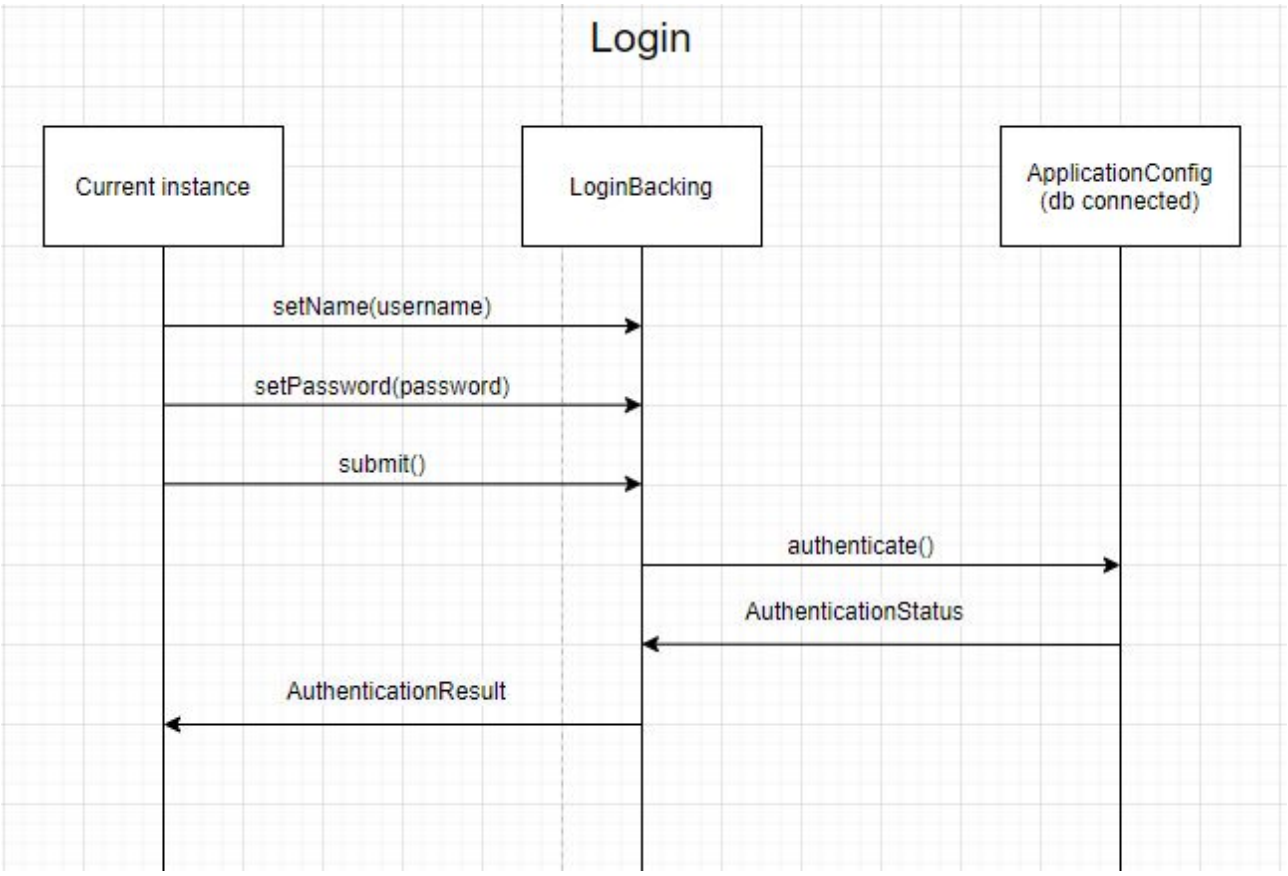
5. Application flow



Creating a new comment or rating is done in the same way but with different components.

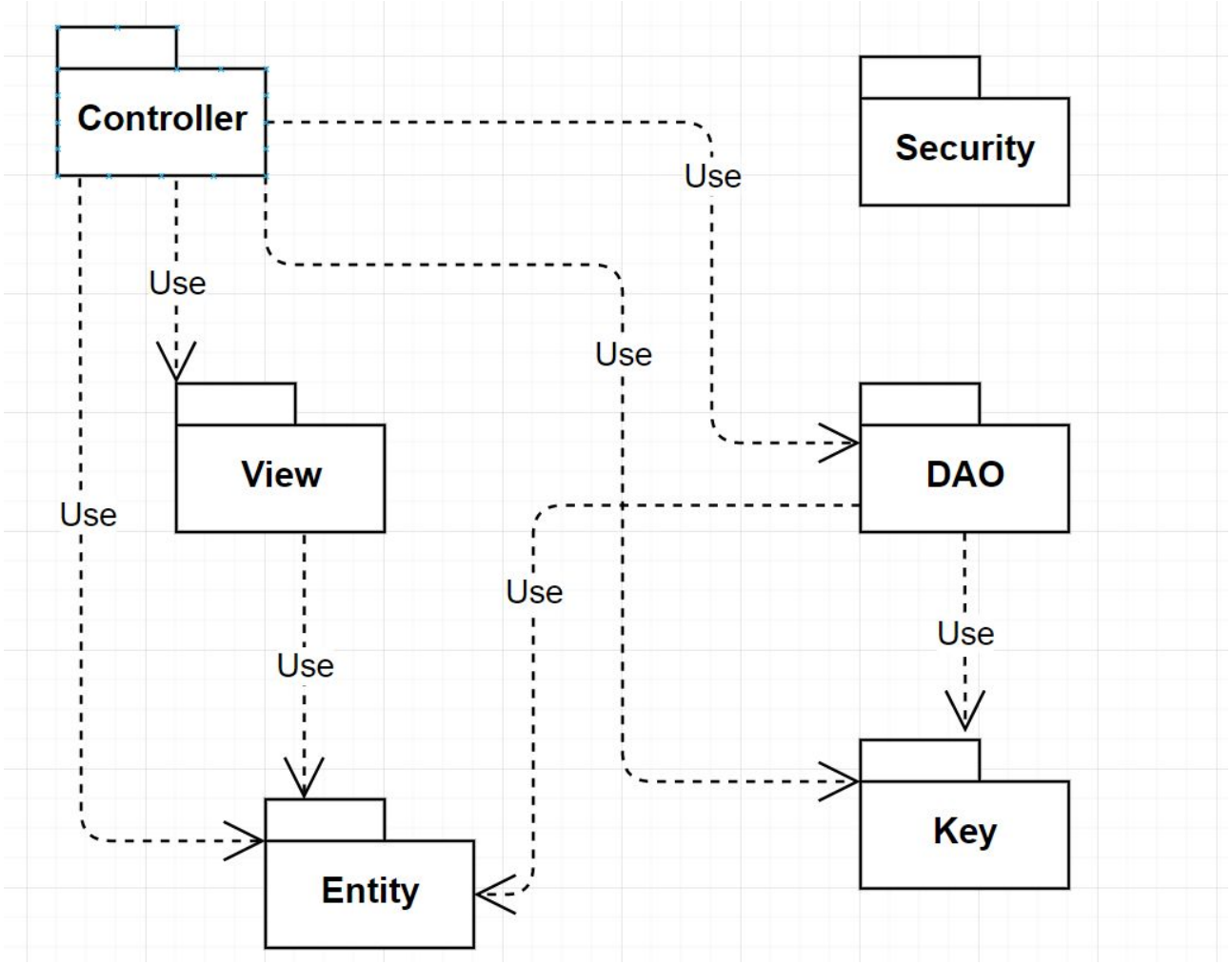


Deleting a comment or rating was made a little bit harder because we had to reattach the detached entity we wanted to remove from the db. To reattach an entity we injected UserTransaction to the controller, then we had to find the entity again and remove it from the db.

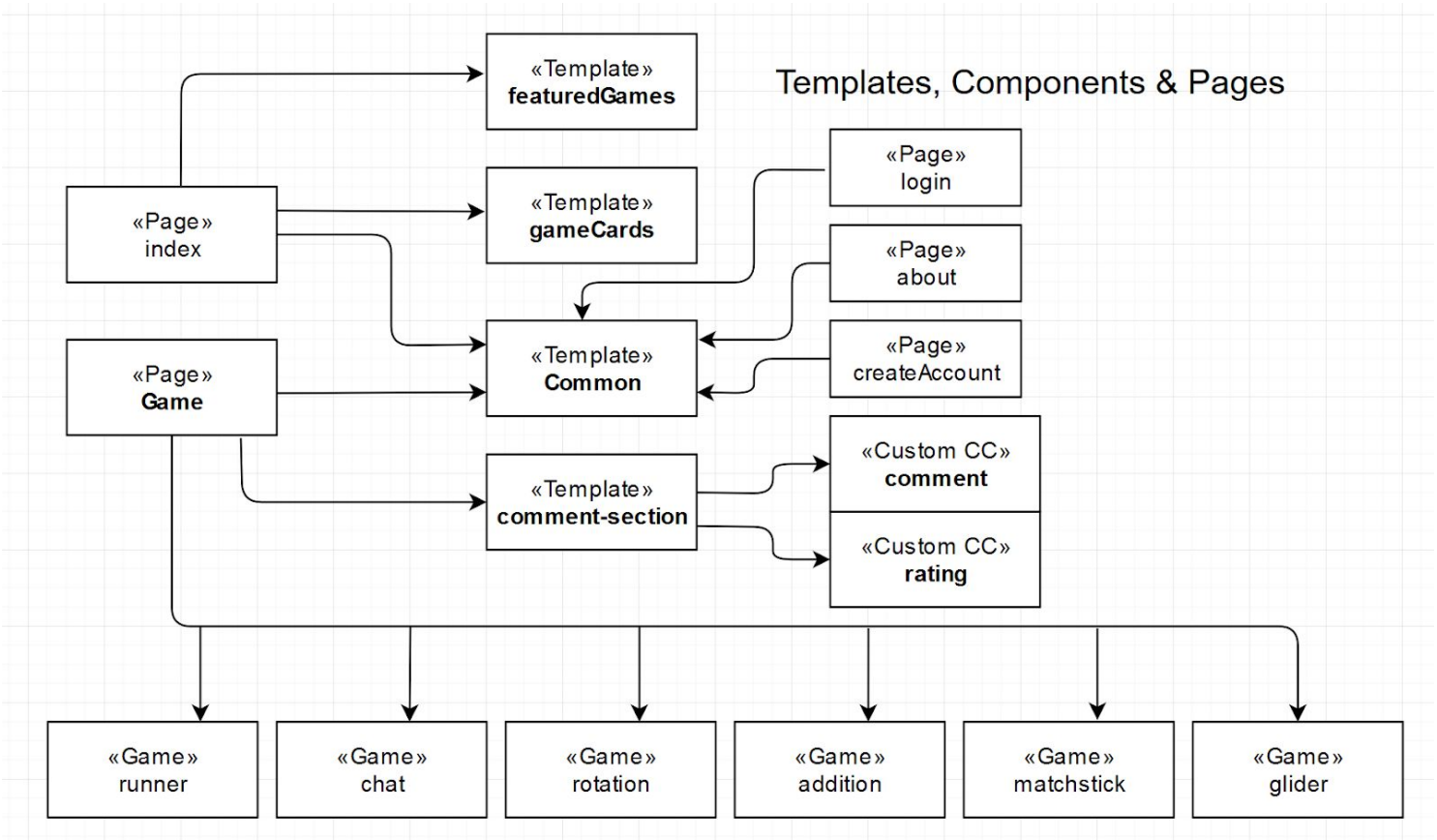


When attempting to login the user will first fill out their username and password. They are then set in loginBacking when the user submits the form. The username and password are then authenticated against applicationConfig which is used to verify the account information with the data in the database. Depending if the user exists or not applicationConfig will return a message which is then interpreted in loginbacking. LoginBacking will then take the appropriate action like displaying a growl with login failed or redirect the user if the login is successful.

6. Package diagram

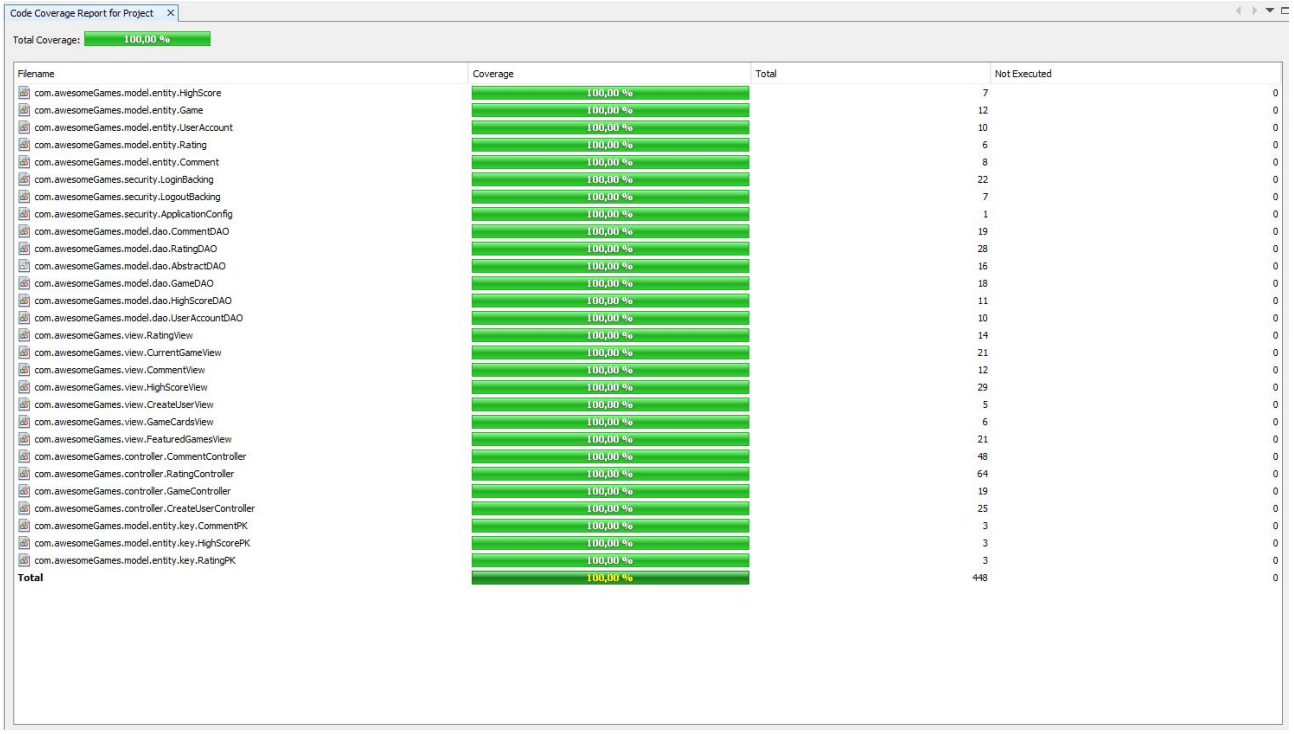


This is a high level package diagram of our project.



This UML shows which pages use which templates, components and games.

7. Code coverage report

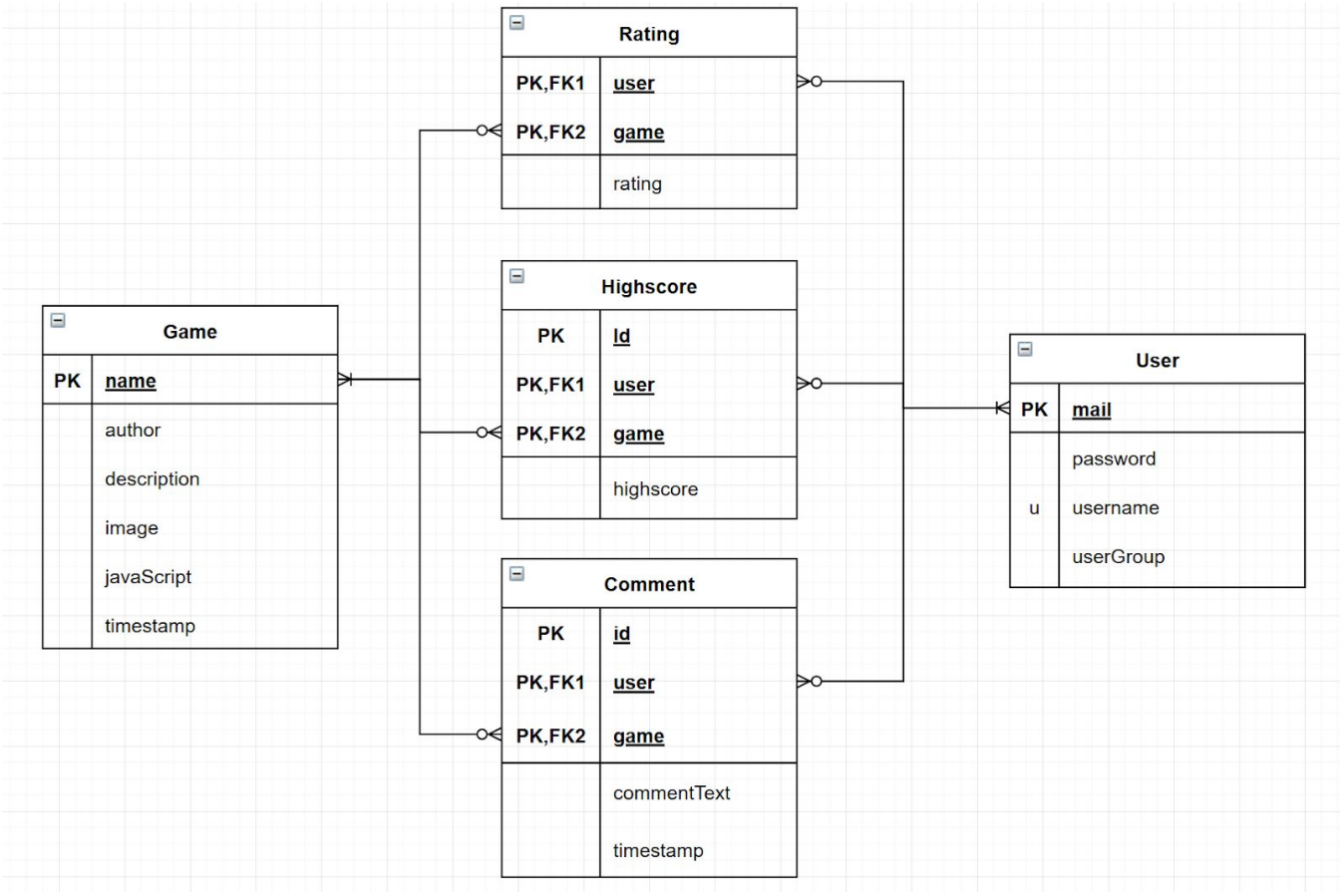

Code Coverage Report for Project

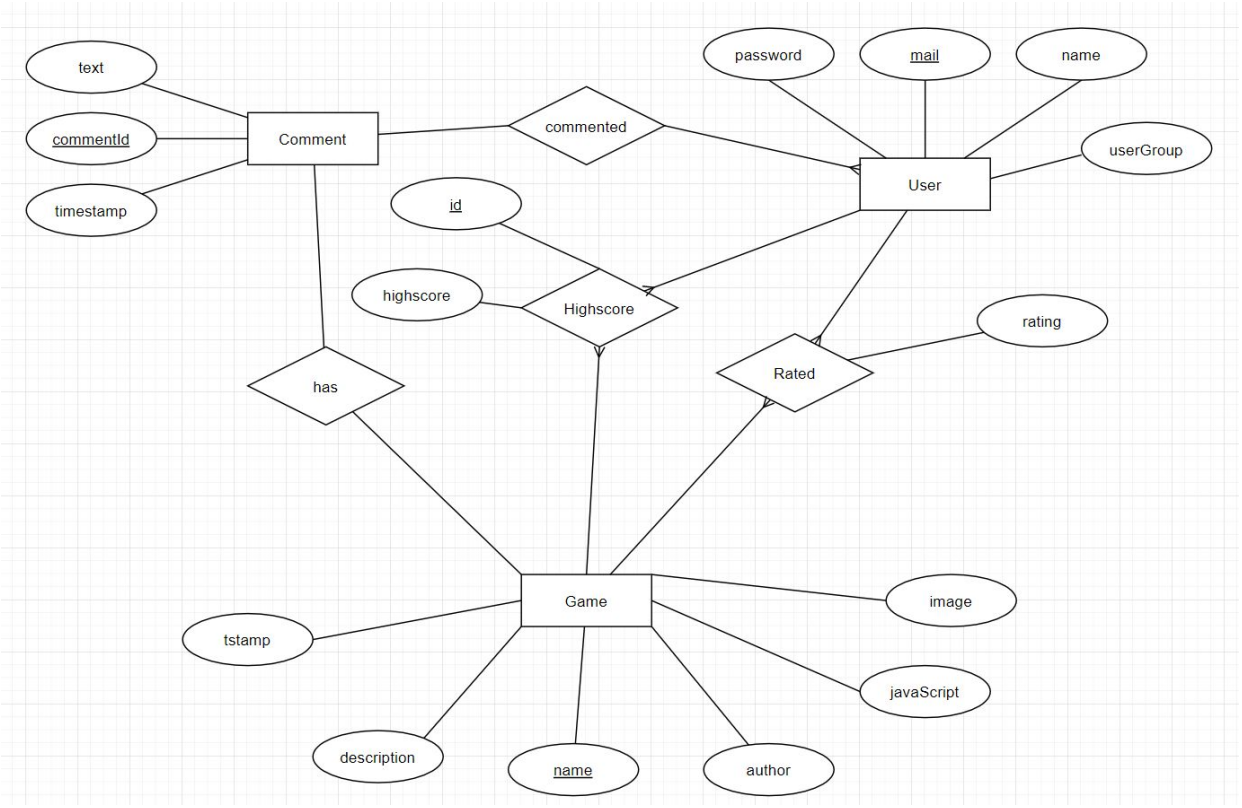
Total Coverage: 100,00 %

Filename	Coverage	Total	Not Executed
com.awesomeGames.model.entity.HighScore	100,00 %	7	0
com.awesomeGames.model.entity.Game	100,00 %	12	0
com.awesomeGames.model.entity.UserAccount	100,00 %	10	0
com.awesomeGames.model.entity.Rating	100,00 %	6	0
com.awesomeGames.model.entity.Comment	100,00 %	8	0
com.awesomeGames.security.LoginBacking	100,00 %	22	0
com.awesomeGames.security.LogoutBacking	100,00 %	7	0
com.awesomeGames.security.ApplicationConfig	100,00 %	1	0
com.awesomeGames.model.dao.CommentDAO	100,00 %	19	0
com.awesomeGames.model.dao.RatingDAO	100,00 %	28	0
com.awesomeGames.model.dao.AbstractDAO	100,00 %	16	0
com.awesomeGames.model.dao.GameDAO	100,00 %	18	0
com.awesomeGames.model.dao.HighScoreDAO	100,00 %	11	0
com.awesomeGames.model.dao.UserAccountDAO	100,00 %	10	0
com.awesomeGames.view.RatingView	100,00 %	14	0
com.awesomeGames.view.CurrentGameView	100,00 %	21	0
com.awesomeGames.view.CommentView	100,00 %	12	0
com.awesomeGames.view.HighScoreView	100,00 %	29	0
com.awesomeGames.view.CreateUserView	100,00 %	5	0
com.awesomeGames.view.GameCardsView	100,00 %	6	0
com.awesomeGames.view.FeaturedGamesView	100,00 %	21	0
com.awesomeGames.controller.CommentController	100,00 %	48	0
com.awesomeGames.controller.RatingController	100,00 %	64	0
com.awesomeGames.controller.GameController	100,00 %	19	0
com.awesomeGames.controller.CreateUserController	100,00 %	25	0
com.awesomeGames.model.entity.key.CommentPK	100,00 %	3	0
com.awesomeGames.model.entity.key.HighScorePK	100,00 %	3	0
com.awesomeGames.model.entity.key.RatingPK	100,00 %	3	0
Total	100,00 %	448	0

We managed 100% coverage for the backend. We had no time to do a formal integration test but we know it can be done with arquillian graphene and drone. We have a manual integration test protocol for testing the frontend.

8. Model diagram





Our Database description:

- Users can only have one rating per game.
- Users can create many comments per game.
- Users can set many highscores per game.
- One Game has many or no comments, from one or many different users.
- One Game has many or no highscores, from one or many different users.
- One Game can have multiple ratings, but only one per user.

9. Responsibilities

Tobias:

Frontend(Components(rating, comment_field), Templates(common, comment-section, rating), index, login, small fixes like dynamic resizing and correct navigation**)**
Games(Glider Game JS, xhtml and pictures**)**
Backend(Persistence, DB init, CommentController & View, RatingController & View, UserController & View (first draft), Rating, CommentDAO, RatingDAO, UserAccountDAO**)**
Testing(DAO(Rating, Comment**))**
Design(Mostly removed but you can see glider game**)**
Documentation(Report first draft and ongoing, UML, ER-diagram, Javadoc, Presentation**)**.

William:

Frontend(CreateAccount, Login, minor modifications to other pages to make login/logout/createAccount pop-ups**)**,
Backend(CommentController, CreateUserController, RatingController, LoginBacking, CreateUserView**)**,
Testing(CommentControllerTest, HighScoreViewTest**)**,
Documentation(JavaDoc, Manual testing protocol, Report**)**

Matteus:

Frontend(HomePage, Navigation, Featured Games, GameCards, GamePage, Comment section, Rating section, Description, Author, AboutPage, CSS, **Games(**Chat Game JS & Visuals, Runner Game JS & Visuals, Addition Game JS & Visuals, Rotation Game JS & Visuals**))**,
Backend(Comment Controller & View, GameCards View, FeaturedGames View, All Model Entities, **DAO(**Comment, Game, Highscore, Rating, UserAccount**))**,
Testing(DAO(Comment, Game, Highscore, Rating, UserAccount**)**, Featured Games, Game Cards**)**,
Design(Logo, Color, Font, Page Layout, Images**)**
Documentation(JavaDoc, ER diagram, UML, Repo Readme, Report**)**

Joachim:

Frontend(common, matchstick**)**
Games(Matchstick Game**)**
Backend(CreateUserController, GameController**)**
Testing(Contextmocker, ApplicationConfigTest, LoginBackingTest, LogoutBackingTest, CreateUserViewTest, CreateUserController, GameControllerTest**)**
Design(All images for Matchstick Game**)**
Documentation(Manual testing protocol, ER diagram, JavaDoc, Report**)**

David:

Frontend(Login, Common, gamePage, Modified the different games to work with iframes)
Backend(All entities, All compound keys, CurrentGameView, RatingView, GameController, CommentController, RatingController)
DAO(CommentDAO, GameDAO,HighScoreDAO,RatingDAO,UserAccountDAO)
Testing(CommentControllerTest, RatingControllerTest, CommentTest, GameTest, HighScoreTest, UserAccountTest, CommentPKTest, HighScorePKTest, RatingPKTest, CommentViewTest, CurrentGameViewTest, FeaturedGamesViewTest, RatingViewTest)
Documentation(JavaDoc, UML, Report)

Simon:

Frontend(component**(**Highscore_box), Game Description, comment-section, About**)**
Backend(HighscoreDAO, GameEntity, HighScoreView**)**
Games(Flappy**)**
Testing(GameDAOTest**)**
Design(Images for Flappy game**)**
Documentation(ER-diagram, Report, JavaDoc**)**