CHALMERS | UNIVERSITY OF GOTHENBURG
Department of Computer Science and Engineering
DAT076/DIT126
Group 6
Jacob Spilg, Isak Magnusson, Alexander Tepic, Adrian Håkansson, Oussama Anadani

Red drink: a WebApp in Java EE and React

Introduction

The app that has been developed during this project offers users a solution to find and share their recipes for both alcoholic and non-alcoholic beverages on a social platform.

Github: https://github.com/theman550/webapps

A list of use cases

- Finding beverages
    - Browse beverages that are sorted by popularity
    - Browse beverages that are sorted by time of creation
    - Get a random beverage
    - Search for beverages based on name
    - Search for beverages based on ingredients
    - Get user's own created drinks
    - Get user's own upvoted drinks
    - Find the percentage of alcohol in a drink
- User functionality
    - Register a user account
    - Store password hashed and salted
    - Upvote beverages
    - Downvote beverages
    - See a list of created beverages
    - See a list of upvoted beverages
    - Add recipes
    - Delete created recipes
    - Upload pictures of the beverages made from the recipes
    - Share a direct link to a specific beverage

User manual

The app should be relatively self explanatory. If the user simply wants to browse existing recipes, no account is required. To register, press "Log In" then register. If a registered user wants to add a recipe, there is an "Add drink" button at the top in the menu-bar. For more information about a drink, i.e. it's ingredients, description and alcohol-percentage, a details button can be found on each drink card. If the user created said drink, a "Delete" button will also show which the user can press to remove the drink from the website. The user can search using the "Search"-box and autocomplete suggestions will show, with the search query either being a drink or a specific ingredient, which shows up in parenthesis after the search-term. The user can sort the shown drinks by either "Most popular" or "Newest" using the dropdown-menu to the left of the "Search"-box. If the search ends up giving more than 20 drinks in response, the paginator can be used to move between pages. The user can also get a drink randomly by pressing the "I'm brave" button. On the user page, which can be accessed by pressing the icon in the upper-right corner, the user can see when they created their account, the recipes they've added, and their liked drinks.

Design

Frontend libraries:

- React
- PrimeReact
    - A large library of javascript components
- Formik
    - A comprehensive form-library for React. It allows for array inputs (used for ingredients) and presents a lot of convenient methods, such as helpers for validation and array modification.
- Yup
    - Form data validation, including error message generation. Compatible with Formik.
- React-native-web
    - A component library compatible with Formik. It was merely used for the input-fields, since PrimeFaces' input fields were not compatible with Formik.

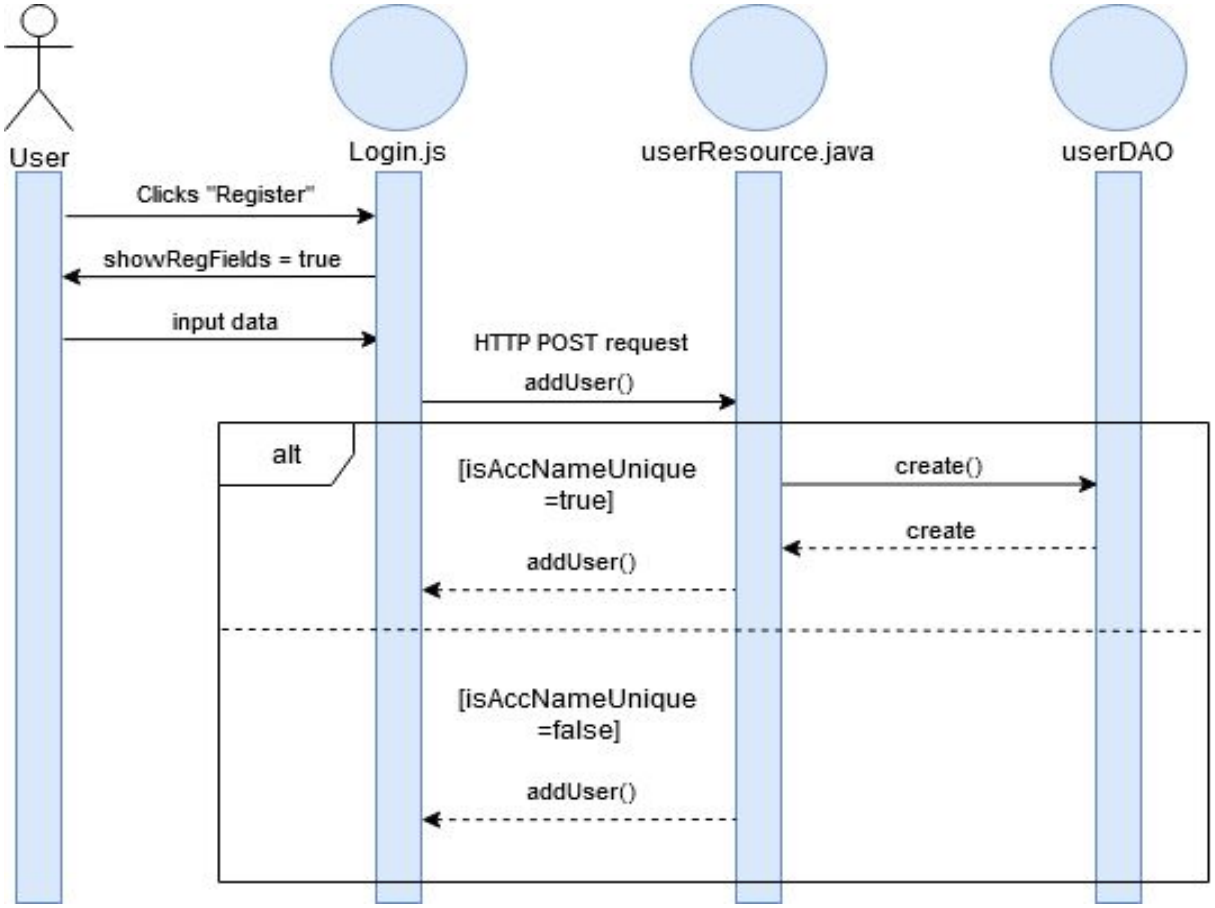
Backend libraries:

- Netbeans
    - The IDE we used to write and compile the backend code.
- Payara
    - Runs all server code.
- Arquillian
    - Server that executes all test files upon building the project.
- QueryDSL
    - An abstraction layer for database manipulation. Was used in the data-access objects to simplify reading, writing and updating database entries.
- Java JWT
    - A unique token based on the user's account name and a secret key generated by the server on startup, given to a client upon successful login. Used to authenticate the client in the backend without exposing their password.
- Maven
    - An automation tool that manages all project dependencies.
- Lombok
    - Provides helper methods for data access objects. Supplies setters, getters etc automatically.

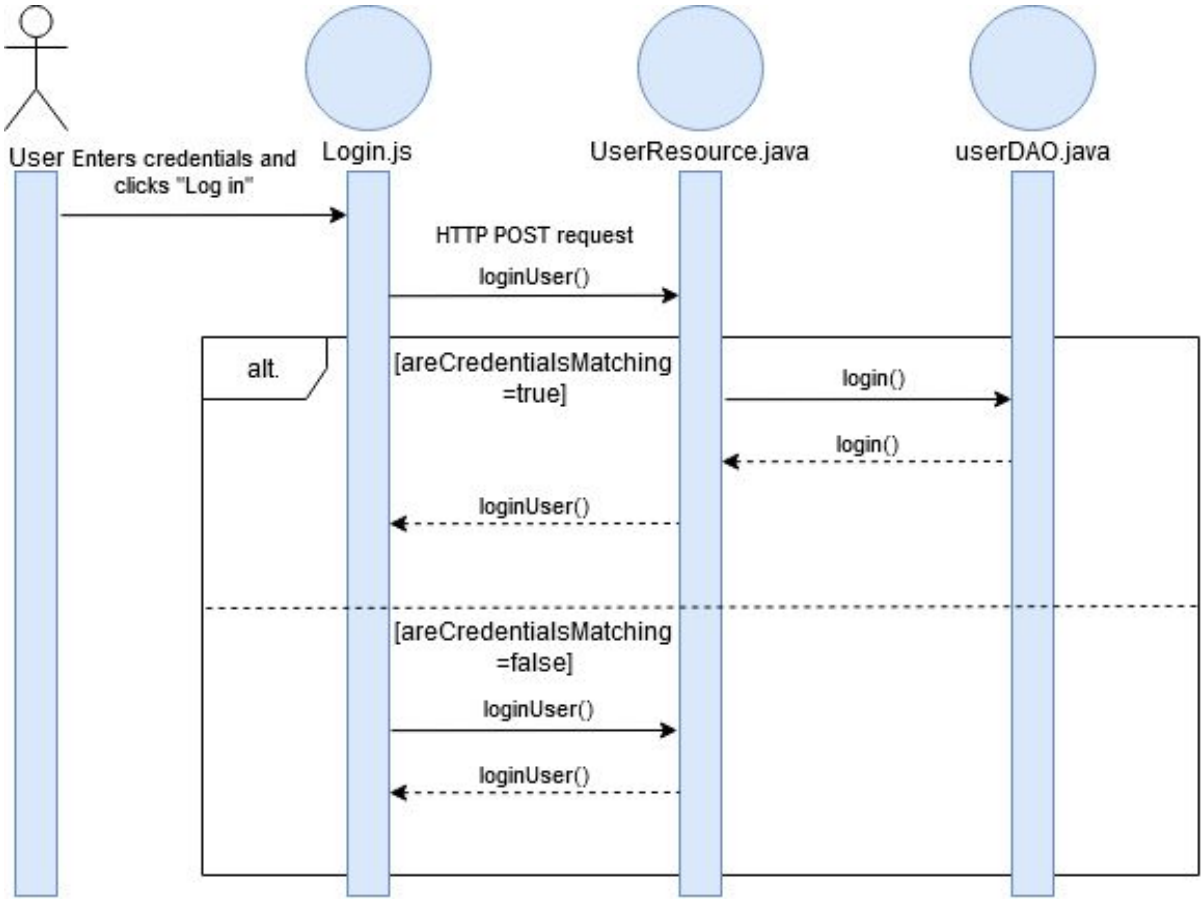
Application flow

Registering and logging in

- Create a user account
    - A fetch with the requested display name, account name and password as payload is sent using the login.js view. The userResource.java servlet will compute the request and add the user, given that the user name was not already taken.

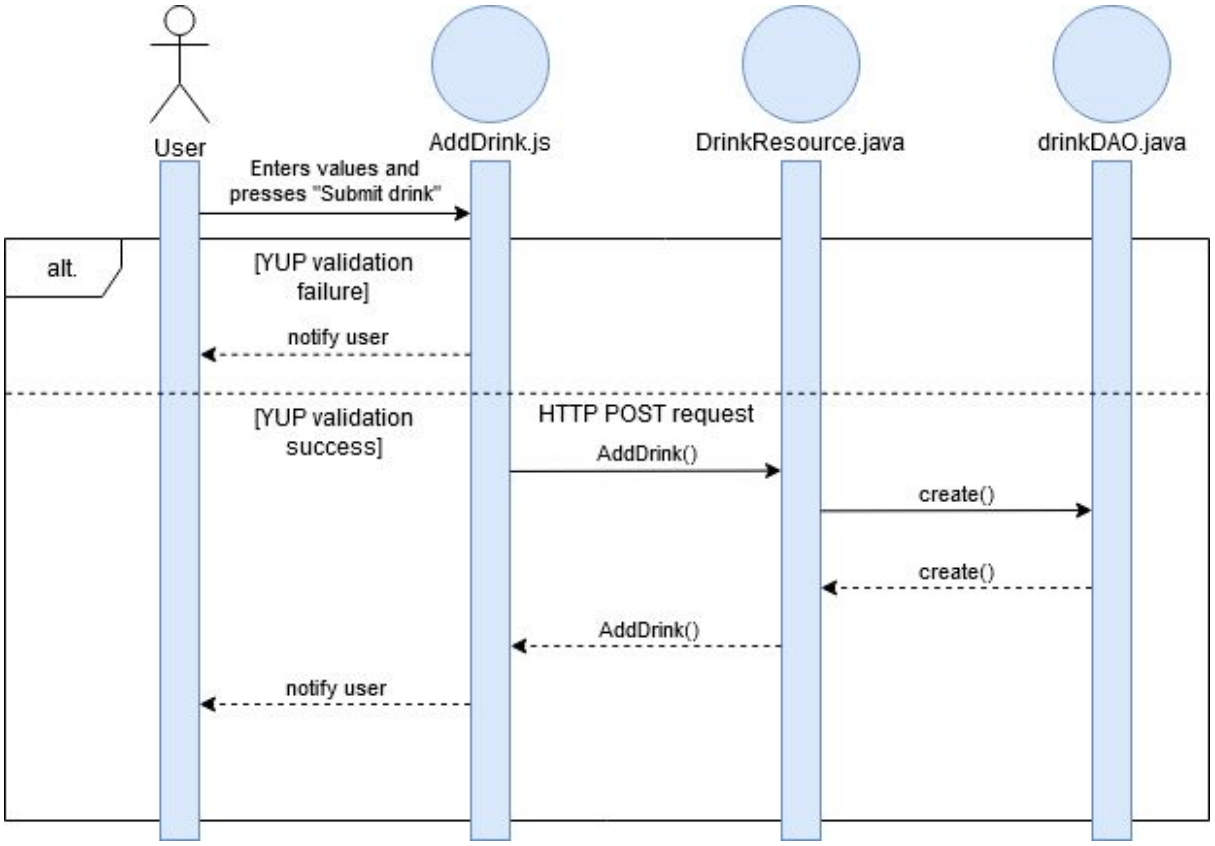


- Log in
    - A fetch is again sent to the userResource servlet. The response from the server, given that the user name and password matches, is an authentication token, which is stored by the client using localStorage.

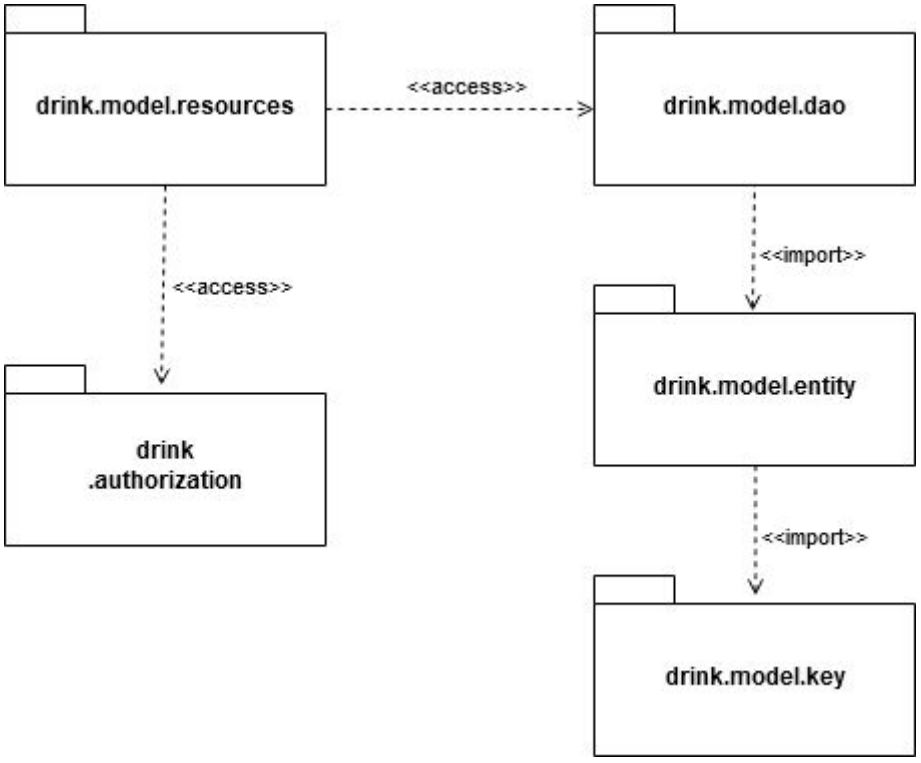


Adding a recipe (after having logged in)

- The addDrink.js view contains everything used to create a new drink. The user inputs data into all fields and presses submit, which prompts Yup to validate all fields.
- Should there be anything wrong with the inputs, a message is printed to the user and nothing is sent to the server.
- Once all data complies with the input requirements, a fetch request is sent to the drinkResource servlet. If a picture URL was provided, it is used, otherwise the site logo is used instead
- The drink resource consumes the JSON data and automatically converts it to a drink-entity. The ingredients are looped through, connecting them to the drink, and the user that submitted the drink is set as creator. Should the recipe be accepted by the server, a message success pops up in the user interface and the form is cleared.

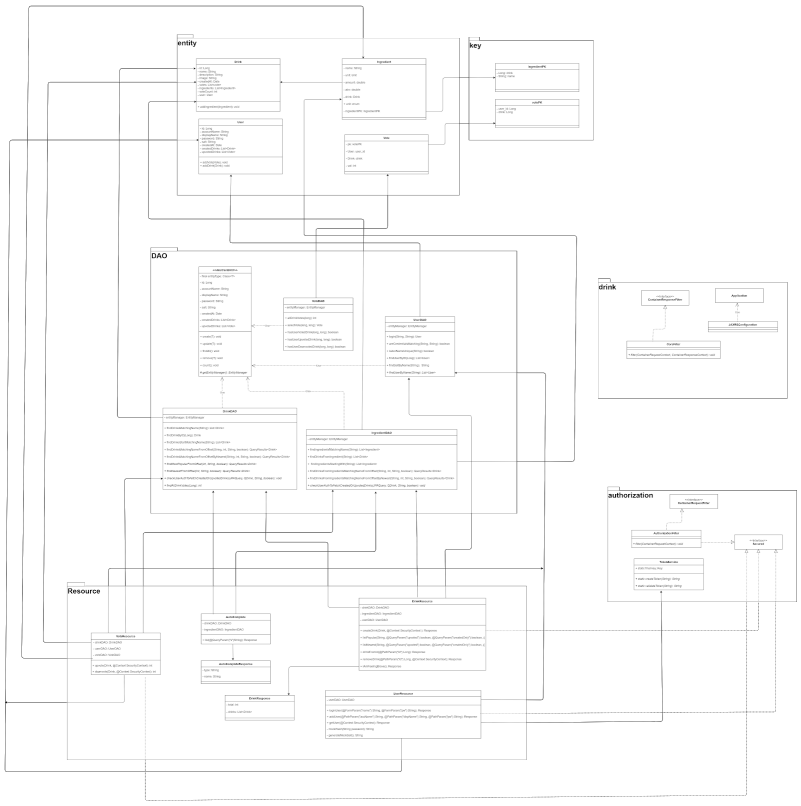


Package Diagram



The drink.authorization package is used for creating JWT's, which is necessary for things such as login, deletion of drinks and other account specific actions. The rest should be self-explanatory.

Model Diagram(UML)



Link to the UML Diagram :
https://app.diagrams.net/?page-id=c3zgZ82mhYx07dWPn7HQ&scale=auto#G1aop4foqQHigz8ioHfo2xhbA6Y8_DMc12

Responsibilities

While we had some components where each individual put extra focus, all members of the group worked in all parts of the software during the course of the project. Here we list what each individual did the majority of work for:

Alexander: Majority of tests (drinkDAOTest, ingredientDAOTest, userDAOTest), DrinkResource, DrinkDAO, IngredientDAO, deletion of drinks (front- and backend), DrinkListItem, AutocompleteResponse

Jacob: AddDrink in frontend. Added the initial tests and initial database queries. Made the first fetch requests and set up a mock database for frontend testing. Stored user data in

frontend and made the site aware of whether you were logged in (not show login when logged in etc) and added logout.

Isak: Register user, parts of login, hash and salt, voting, testing voteDAO

Oussama: Login page(frontend) and a small part in the backend. Parts of details' dialog front- and backend, timestamp, small parts of createdDrinks and upvotedDrinks in profile page. Small parts here and there too. The whole UML Diagram model.

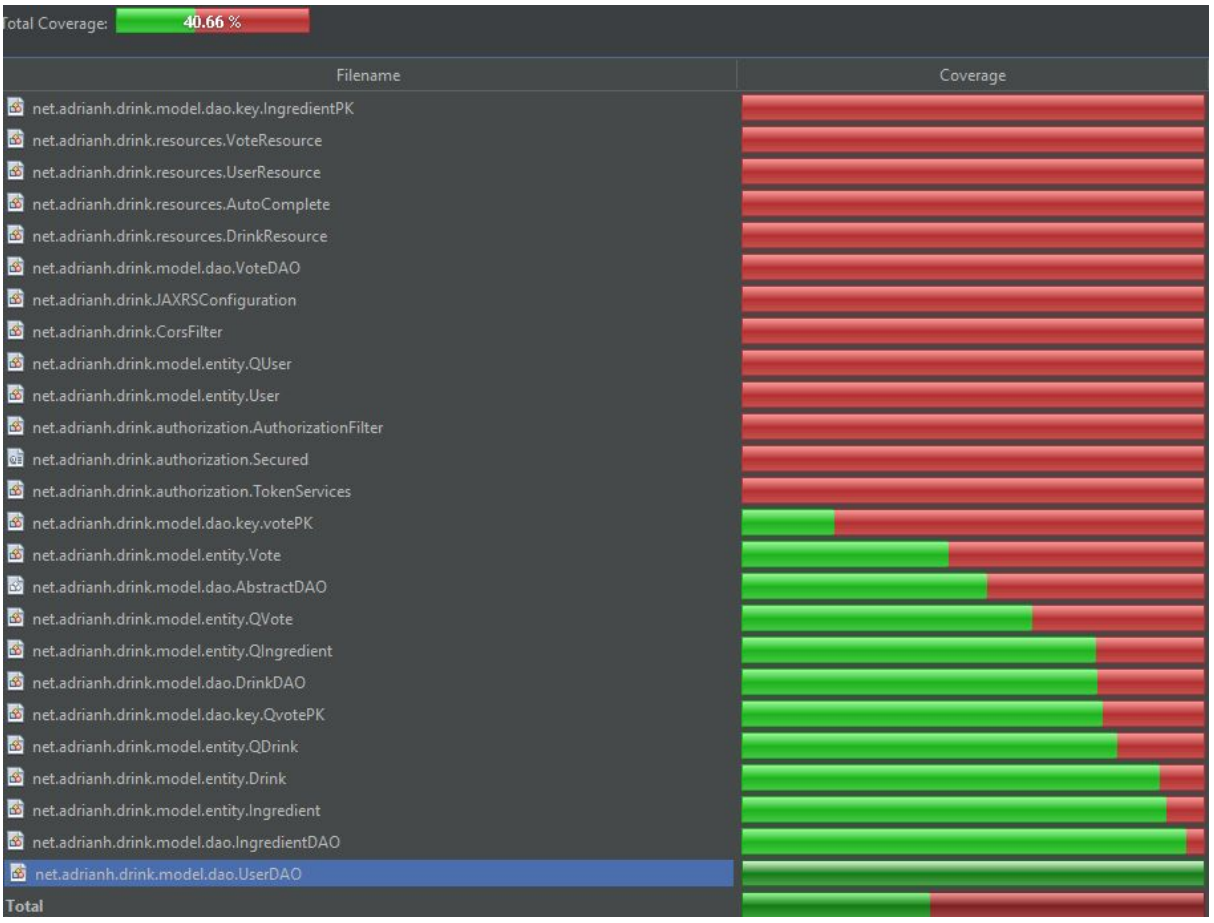
Adrian: Homepage in the frontend as well as the card, details and list components. Also responsible for authorization with JWT.

Code Coverage

40.66%
(Only entity and dao classes are tested)

Total Coverage: 40.66 %

Filename	Coverage	Total	Not Executed
net.adrianh.drink.model.dao.key.IngredientPK		3	3
net.adrianh.drink.resources.VoteResource		33	33
net.adrianh.drink.resources.UserResource		28	28
net.adrianh.drink.resources.AutoComplete		9	9
net.adrianh.drink.resources.DrinkResource		113	113
net.adrianh.drink.model.dao.VoteDAO		33	33
net.adrianh.drink.JAXRSConfiguration		1	1
net.adrianh.drink.CorsFilter		6	6
net.adrianh.drink.model.entity.QUser		15	15
net.adrianh.drink.model.entity.User		21	21
net.adrianh.drink.authorization.AuthorizationFilter		19	19
net.adrianh.drink.authorization.Secured		0	0
net.adrianh.drink.authorization.TokenServices		6	6
net.adrianh.drink.model.dao.key.votePK		5	4
net.adrianh.drink.model.entity.Vote		9	5
net.adrianh.drink.model.dao.AbstractDAO		17	8
net.adrianh.drink.model.entity.QVote		16	6
net.adrianh.drink.model.entity.QIngredient		17	4
net.adrianh.drink.model.dao.DrinkDAO		77	18
net.adrianh.drink.model.dao.key.QvotePK		9	2
net.adrianh.drink.model.entity.QDrink		21	4
net.adrianh.drink.model.entity.Drink		20	2
net.adrianh.drink.model.entity.Ingredient		12	1
net.adrianh.drink.model.dao.IngredientDAO		49	2
net.adrianh.drink.model.dao.UserDAO		39	0
Total		578	343


Total Coverage: 40.66 %
Filename
Coverage
net.adrianh.drink.model.dao.key.IngredientPK
net.adrianh.drink.resources.VoteResource
net.adrianh.drink.resources.UserResource
net.adrianh.drink.resources.AutoComplete
net.adrianh.drink.resources.DrinkResource
net.adrianh.drink.model.dao.VoteDAO
net.adrianh.drink.JAXRSConfiguration
net.adrianh.drink.CorsFilter
net.adrianh.drink.model.entity.QUser
net.adrianh.drink.model.entity.User
net.adrianh.drink.authorization.AuthorizationFilter
net.adrianh.drink.authorization.Secured
net.adrianh.drink.authorization.TokenServices
net.adrianh.drink.model.dao.key.votePK
net.adrianh.drink.model.entity.Vote
net.adrianh.drink.model.dao.AbstractDAO
net.adrianh.drink.model.entity.QVote
net.adrianh.drink.model.entity.QIngredient
net.adrianh.drink.model.dao.DrinkDAO
net.adrianh.drink.model.dao.key.QvotePK
net.adrianh.drink.model.entity.QDrink
net.adrianh.drink.model.entity.Drink
net.adrianh.drink.model.entity.Ingredient
net.adrianh.drink.model.dao.IngredientDAO
net.adrianh.drink.model.dao.UserDAO
Total