CHALMERS UNIVERSITY OF TECHNOLOGY

WEB-APPLICATIONS

DAT076

Documentation

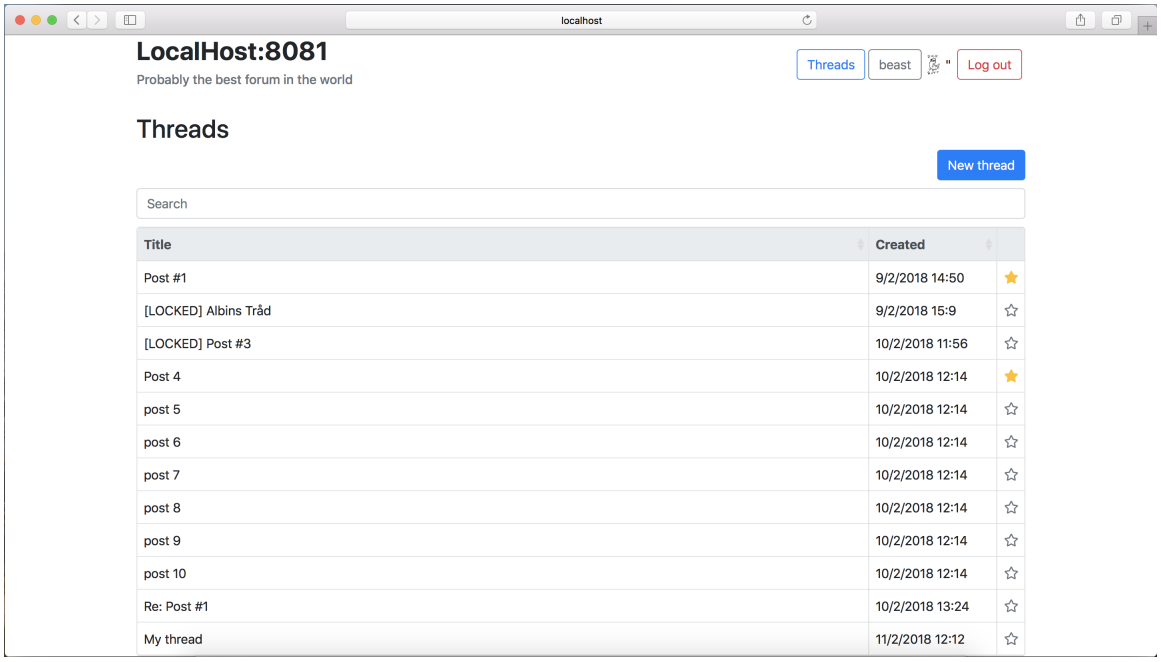
Authors
Albin BÅÅW
Eric SHAO
Pedram SHIRMOHAMMAD

Supervisor
Joachim VON HACHT

March, 2018

Group members (left to right): Pedram, Eric Albin

1 The website



The website/application that we choose to build is a forum to discuss whatever might be on the user's mind. How to use the forum:

1. Register an account
2. Login in with your credentials
3. Start interacting on the forum by starting new threads, commenting on your/others' threads and favorite posts you like.
4. Visit your user profile to change your profile image and view your favorite threads.

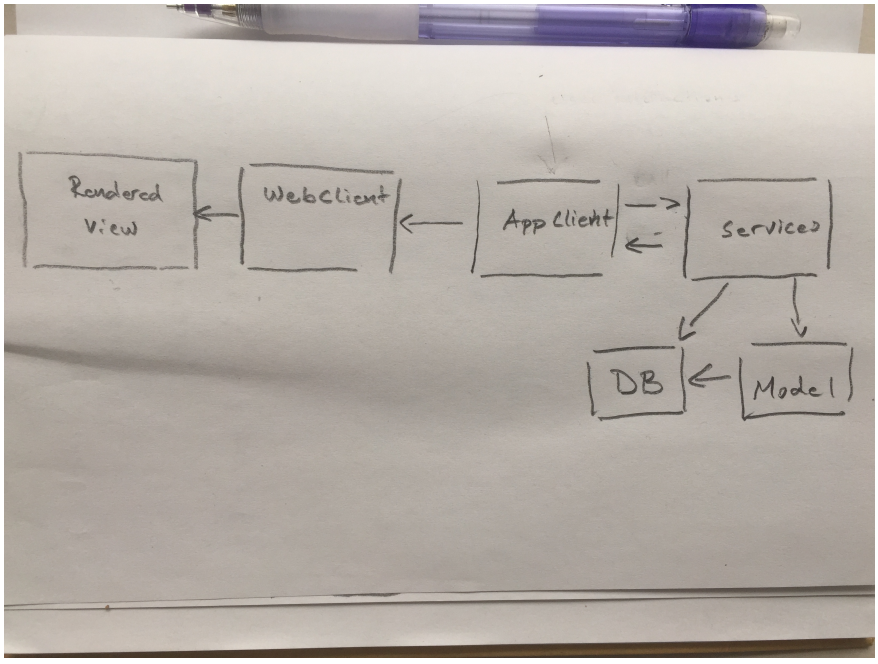
2 Use cases

The use cases which we chose to implement are the following:

1. Register a new user account
2. Login as a user
3. Login as an admin

4. Logout
5. Create new threads
6. Comment on threads
7. View users' threads
8. View specific user created threads
9. Search after threads
10. Sort threads based on title
11. Sort threads based on creation date
12. Favorite/star threads
13. Change your profile picture
14. View threads you've favorite:d
15. As an admin ban users
16. As an admin lock threads (stop users from commenting)
17. View other users' profiles.

3 Architecture



When developing the application we used a Model-View-Controller-pattern (MVC-pattern), where the application can be divided into three significant parts. The picture above describes a top-down view of the system. There is a component called webclient which based on data sent from the appclient renders the view which the user can see. The appclient-component acts as a router/controller, as it based on get-/post-requests sent to the server tells the systems which operations it should perform and then presents the result to the user either by asking webclient to render a view, or by sending back json-data. The model-part of the app consists of the service- and model-components combined. Where the model-component is used to easier interact with the database (called DB in the picture) when performing CRUD-operations. The service-component acts as an intermediator between the appclient- and the model-component (e.g. when creating a new user), but can also perform batch-operations on the database by itself (e.g. fetch all threads).

3.1 Backend

3.1.1 Appclient

The appclient (server.js) is the most central part of the application, as it handles all the post- and get-requests sent to the server. By interacting with the classes contained in the services-components it can tell the system in which order it should perform different operations in order to achieve the correct result.

3.1.2 Services

The services-components consists of different managers corresponding to their own models. For instance, there is a user manager which interacts with the user model in order to perform operations related to the user (e.g. update user info, get a specific user, etc.). A manager can also interact directly with the database in order to perform batch-operations, for instance, the post manager (used to handle threads) queries directly from the database in order to fetch multiple threads at once.

3.1.3 Models

The models-component consists of different classes used to represent rows of specific tables in the database. For instance, the user model represents a row in the user table. A model also contains functions in order to perform basic queries from the database.

3.1.4 Webclient

The webclient uses data embedded in a request in order to render the views the user can see. It also has other functions such as setting cookies after the user successfully logs in.

3.1.5 Authentication

In almost every request made to the server we need to authenticate the user. For this we use a node module called jsonwebtoken, which basically encodes user info which we later save as a cookie in the user's client. We then read that cookie every time the user makes a request to the server, and if the cookie is still valid we continue with the request and if otherwise we send them to the login-page.

3.2 Frontend

3.2.1 EJS

For HTML-templating we are using a framework called EJS, making it possible to write HTML-templates using javascript.

3.2.2 AJAX

We are only using AJAX when necessary, i.e. when posting forms and fetching data retroactively, e.g. when fetching all the users' names and profile pictures in thread-view. In order to more effectively use AJAX we have written a couple of help functions located in main.js.

3.3 Database

For the database we are using mysql, for effective database handling we are using a set of classes gathered in the folder DATSQL. The tables in the database looks as following:

3.3.1 users

id – INT, NOT NULL, AUTO_INCREMENT, PRIMARY KEY
name – UNIQUE
password – VARCHAR(511), NOT NULL
banned – BOOLEAN, NOT NULL
admin – BOOLEAN, NOT NULL
image

3.3.2 posts

id – INT, NOT NULL, AUTO_INCREMENT, PRIMARY KEY
user_id – INT, NOT NULL
parent_id – INT
repost_id – INT
title
text
date
locked – BOOLEAN, NOT NULL

(Note: `repost_id` is never used as we dropped the corresponding feature)

3.3.3 favorites

id – INT, NOT NULL, AUTO_INCREMENT, PRIMARY KEY
user_id – INT, NOT NULL
post_id – INT, NOT NULL

4 Frameworks

4.1 CSS

We choose to use Bootstrap for our css-framework as it helps us with layout and some basic design components such as good looking buttons.

4.2 Javascript

We are using a couple of different frameworks and plug-ins for javascript. We are using jQuery for utilities (e.g. ajax simplified). We are also using Bootstrap and bootstrap-table (standalone plug-in) for dynamic tables.