MOTIUS

WE R&D.

JAVASCRIPT TEST PROJECT

Datum	9. Februar 2018
Autor	Matthias Lehner

MOTIUS GMBH

Lichtenbergstraße 8
85748 Garching b München
Deutschland
http://www.motius.de
mail@motius.de

GESCHÄFTSFÜHRER

Zièd Bahrouni, Daniel Weiß
+49 (0) 89 / 21 55 16 16
Amtsgericht MUC: HRB 205305
UST-ID: DE 28 90 82 626
Steuernummer: 143-164-10880

BANKVERBINDUNG

Stadtsparkasse München
KN 1002 8653 58
BLZ 701 500 00
IBAN DE68 7015 0000 1002 8653 58
BIC SSKMDEMMXXX

Goal

Design and implement a JS application similar to Instagram. In the following the app will be called “Motigram”. Motigram should allow a user to log in, view a list of posts consisting of a photo with a caption and the username of the creator. Additionally, if a user clicks on a username, a profile-like screen should be shown. Last but not least a user should also be able to create a new post. The entire application should look and feel in accordance to Googles Material Design Language.

You can build the application with React + Redux but also Angular or any other awesome framework that you can think of!

Functional Requirements

1. The application should offer the ability to login in. To do so, implement a login component and use the login endpoint of the backend at
   http://127.0.0.1:8080/auth/login
   to authenticate. A post request to this endpoint with the below specified username and password returns, if successful, a JSON object containing a authentication token and the user that logged in. Both will be useful later on. The authentication token has to be set as a header called “Authorization” on every request to an endpoint that contains “/api” in their path.

   ```
   {
     username: “motee”,
     password: “motiusisawesome”
   }
   ```

2. Once logged in, Motigram should show a list of posts consisting of a photo, an optional caption and the creator’s username in a Material Design card (https://material.io/guidelines/components/cards.html).
   The card should also contain a button that allows for liking or disliking a post and show the name of the user that created said post. A list of posts can be retrieved from the endpoint at http://127.0.0.1:8080/api/posts/.

3. Above the list of posts a toolbar
   (https://material.io/guidelines/components/toolbars.html)
   should be displayed. When scrolling the posts list, the toolbar should always stay on top. In the toolbar, one should find the name of the currently logged in user as well as an upload button that allows for creating a new post and one that lets the user log out.

4. If a user clicks on the username either in the toolbar or in one of the posts they should be shown a sort of profile component of the respective user. On this component, a profile picture, the username and the date the user joined should be visible. This can be either done using a popup or on an entirely new page, be creative!

5. If the user clicks on the upload button from the toolbar, a dialog should open that allows you to upload an image and have a text field to insert a suitable caption for the

image. On pressing a finish button, the post should be uploaded and the list of posts refreshed. To upload an image an extra endpoint at http://127.0.0.1:8080/api/files/ has to be used. To upload an image, this endpoint takes a multipart-form-data encoded file assigned to the key "data". If you run into a "request entity too large" error, make sure the "Content-Type" header is removed and does not say "application/json". The ID of the uploaded file, which is, with other information, returned on upload has then to be stored with the actual post object. The payload for uploading a new post therefore looks something like the following:

```
{
  photo: <ID of the photo you previously uploaded>,
  caption:  „This is a new post!",
  likes: 0
}
```

6. **OPTIONAL BONUS:** Implement the ability to not only like or dislike a post, but also leave a comment. All comments that so far have been issued for a post should be visible below it. This can be done using the http://127.0.0.1:8080/api/posts/<ID>/comments/ endpoint. The payload for uploading a new comment has to be structured like the following:

```
{
  text:  „This is a new post!"
}
```

Non-Functional Requirements

1. The application should use Googles Material Design. This should be done using the React Material library from http://www.material-ui.com/#/. More information on Material Design can be found at https://material.io.

2. Even though this is just a small project, try to keep your code modular and well-documented. (This is your test project after all!)

3. Use ES6 style JavaScript with JSX and use Babel (or similar) for compiling to ES5.

4. Include your package.json file with all your dependencies and extra build steps. If you use other tools like webpack or gulp, also attach all other files required for us to run your application.

Tasks

1. Install Node.js, MongoDB and your frontend framework.
   Instructions to install Node.js can be found at:
   https://nodejs.org/en/download/
   Instructions to install MongoDB can be found at:
   https://www.mongodb.com/de
2. Setup a GitHub or Bitbucket project.
3. Setup the backend server.
   To do so, go to the folder where the server's files reside
   in the terminal emulator of your choice and type „npm install".
   If the installation finishes successfully, the server can be started
   using "npm start".
4. Create your new JS application.
5. Implement the listed functional requirements and keep the non-functional ones in mind.
6. Document the build and deployment process in a README if required.
7. Send a link to your repository to career@motius.de.