Jakub Gawrylkowicz (a01326002)

MS3 – User Interface – Status Report

1. Core design decisions

Since we are free to use any technology stack for to complete this assignment, after some research in some of the latest web technologies, I have decided to create the user interface with JavaScript, with a MEAN (MongoDB, Express, Angular, Node) stack to be precise.

Express and Node are together the core of my back-end and the RESTful API. They handle all incoming requests and pass the data to the front-end of the application. The front-end is a single page application with dynamic views based on the AngularJS. All the necessary code is retrieved at the first load of the page, which decreases the number of HTTP requests dramatically. For the time being, the database created in Mongo is a part of the backup service incase any of the other services are unavailable.

We, as a team, decided to use REST architecture to provide data to our services and connect with each other. Every one of our micro-services has a client and a server instance which handle the request and responses via JSON files over HTTP. All the necessary CRUD operations are handled via HTTP methods such as GET, POST, PUT and DELETE.

2. Service deployment

The service is deployed through the Express framework and is reachable for other teammates under `10.102.107.5:8080`. Even though the site will route to the home page with all the lists automatically, it should still be accessed from `10.102.107.5:8080/login` first. Otherwise, no user session will be created, which is necessary to be able to access any data from the server.

3. Testing and presentation

Like I mentioned before, I am testing my micro-service currently the Mongo database to test my micro-service. The specific preconfigured routes can be found in the file `app/routes.js`. As the other services mature, I am going to implement their routes into my service.

In the end, I think I going to keep my “local” database, because it may prove a great solution if other services are unavailable. For example, if the MS2 is offline I can generate an offline user session (this functionality exists by the way) and save the lists into my “local” mongo database, which then can be later synced to an existing account. This way I can provide a nearly full user experience without connecting to the MS2.

The lack of connection to the MS1 may prove to be a bigger problem. For the time being, I haven't found a suitable solution to this problem yet. I think I would have to probably save the credentials locally and somehow hash and check if user input matches them, which does not sound like a good idea, to be honest.

4. Current state

Functionality fully supported and tested	• CRUD of lists • CRUD of the list • Getting the right lists for user (every list contains the username) • Saving and Restoring the user's session (even if the browser's windows closes via window.LocalStorage) • Offline-Mode: Creating an offline account if MS1 and MS2 are unreachable and connecting to a local MongoDB. • Input validation
Functionality supported, but not tested	• Connection to MS2 over VPN has been established and HTTP routes has been set. • Login and Registration: The code for handling of both processes is already written, but the routes of MS1 has not been set yet as of 11th November. Offline user creation works via MongoDB.
Functionality not yet supported	• Authorisation and Login via MS1 because of not existing routes.

5. Installation

As of writing the status report the mongoDB has to be configured manually, since I had issues with setting up the database via JavaScript (unhandled rejection errors).

5.1 Required packages

My project requires following packages in order to function properly, which can be downloaded via any package manager of your choice:

- NPM 5.X or newer
- Node.js 8.X or newer
- MongoDB

5.2 Installation of required project dependencies via NPM

First we need to download all the required node modules and save them locally.

```
$ npm install
```

5.3 Setting up MongoDB

Then we have to create a database for the local entries. First we need to make sure that MongoDB service is up and running.

a) To start the service (read the documentation on how to start the service on your system https://docs.mongodb.com/manual/administration/install-community/)

```
$ services start mongod (Ubuntu)
$ systemctl start mongodb.service (Arch)
```

b) Then we have to enter the shell of mongo in order to create a database

```
$ mongo
```

c) Create a new database

```
$ use todosdb
```

d) Create the required collections

```
$ db.createCollection('lists')
$ db.createCollection('users')
```

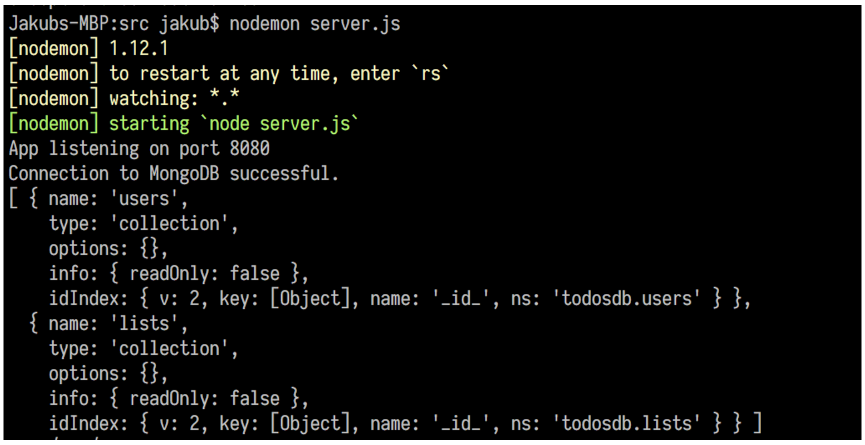
e) The now empty database can be reached via

```
mongodb://127.0.0.1:27017/todosdb
```

5.4 Starting the server

After everything has been installed correctly, it is now possible to start the server via:

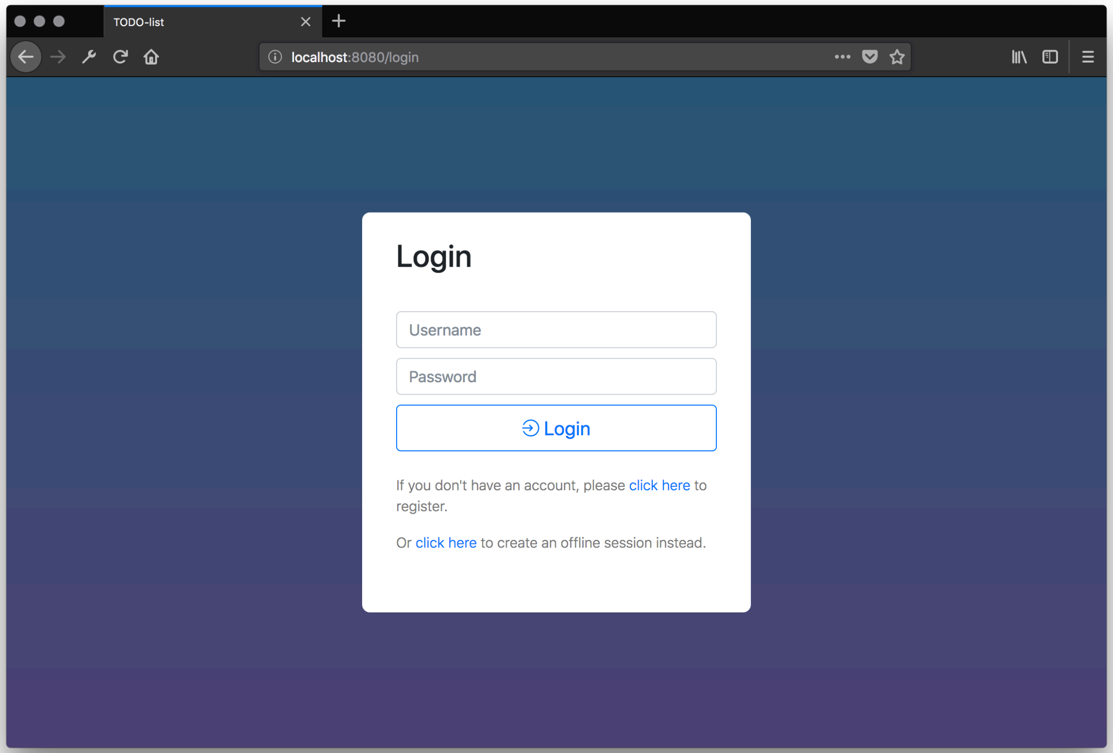
```
$ node server.js
```
or
```
$ nodemon server.js
```



```
Jakubs-MBP:src jakub$ nodemon server.js
[nodemon] 1.12.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`
App listening on port 8080
Connection to MongoDB successful.
[ { name: 'users',
    type: 'collection',
    options: {},
    info: { readOnly: false },
    idIndex: { v: 2, key: [Object], name: '_id_', ns: 'todosdb.users' } },
  { name: 'lists',
    type: 'collection',
    options: {},
    info: { readOnly: false },
    idIndex: { v: 2, key: [Object], name: '_id_', ns: 'todosdb.lists' } } ]
```

Screenshot 1: Successful start of the service

The site can be accessed locally via `localhost:8080/login` and it is now possible to enter the main application though “Create an offline mode”.



Login

Username

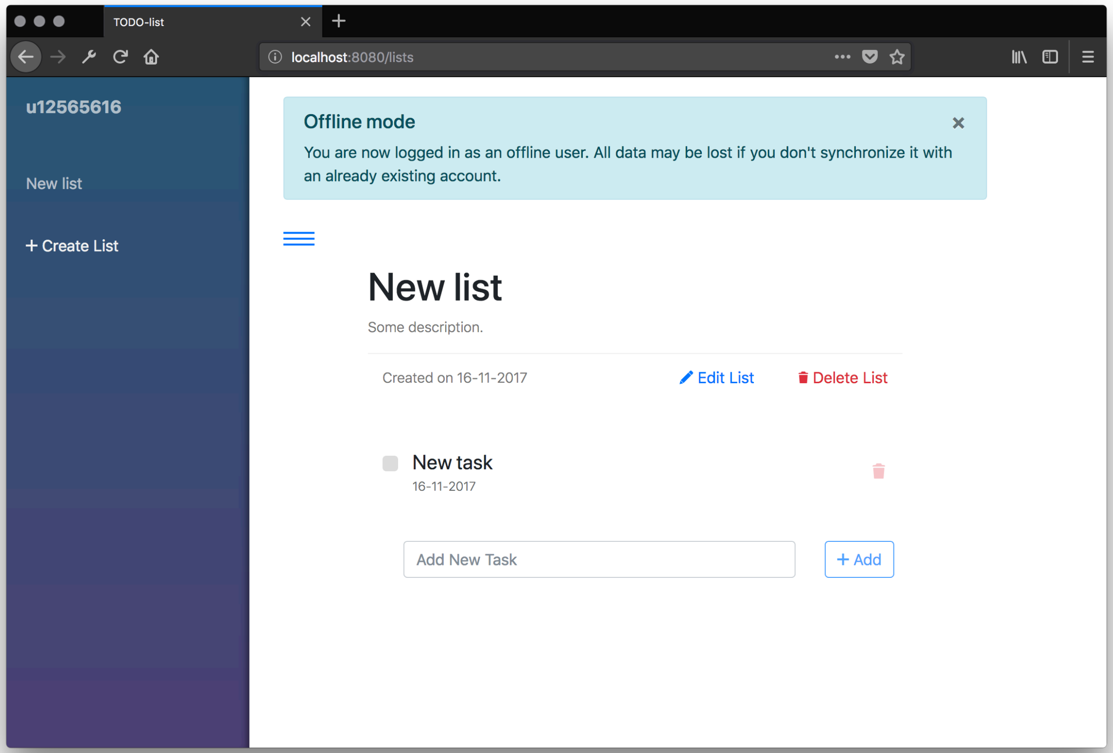
Password

Login

If you don't have an account, please click here to register.

Or click here to create an offline session instead.

Screenshot 2: Login page



Offline mode

You are now logged in as an offline user. All data may be lost if you don't synchronize it with an already existing account.

New list

Some description.

Created on 16-11-2017 Edit List Delete List

New task
16-11-2017

Add New Task + Add

Screenshot 3: Lists page