Jitter

*Изготвили:*

*Драгомир Любомиров Пройчев, ФН 81251*

*Георги Мартинов Божинов, ФН 81333*

Съдържание

[1. Функционалност. 3](#_Toc517637803)

[1.1. Функционални изисквания 3](#_Toc517637804)

[1.2. Нефункционални изисквания 4](#_Toc517637805)

[2. Използвани библиотеки и модули 4](#_Toc517637806)

[2.1. Сървърна част 4](#_Toc517637807)

[2.2. Клиентка част 5](#_Toc517637808)

[3. Нетривиални аспекти на архитектурата 6](#_Toc517637809)

[3.1. Сървърна част 6](#_Toc517637810)

[3.2. Клиентска част 6](#_Toc517637811)

[4. Описание на REST API архитектурата 6](#_Toc517637812)

[5. Описание на начина на инсталиране и конфигуриране на разработената система. 7](#_Toc517637813)

[6. Заключение - срещнати/преодолени трудности и перспективи за бъдещо развитие на системата. 7](#_Toc517637814)

[7. Използвани материали 7](#_Toc517637815)

# Функционалност.

## Функционални изисквания

|  |  |  |
| --- | --- | --- |
| **Who** | **What** | **Why** |
| As a guest | I want to be able to register an account | So that I can access all of the site’s features |
| As a guest | I want to be able to log in into my existing account | So that I can continue using my profile |
| As an authenticated user | I want to be able to share personal information and edit my profile | So that others can find out more about me |
| As an authenticated user | I want to be able to create posts with content and optional images | So that I can share my experiences and thoughts |
| As an authenticated user | I want to be able to edit and delete my existing posts | So that I can fix any mistakes I’ve made |
| As an authenticated user | I want to be able to comment on existing posts | So that I can start a discussion or contribute to an existing one |
| As an authenticated user | I want to be able to edit and delete my comments | So that I can fix any mistakes I’ve made |
| As an authenticated user | I want to be able to like or dislike other posts | So that I can clearly state my opinion about them |
| As an authenticated user | I want to be able to search for other users | So that I can check out their profile and find out more about them |
| As an authenticated user | I want to be able to send friend requests to other users | So that we can follow each other’s activity and keep a connection |
| As an authenticated user | I want to be able to accept or decline friend requests | So that I can decide who I want to connect with |
| As an authenticated user | I want to be able to remove existing friends | So that I can cut my ties with people I no longer desire to connect with |
| As an authenticated user | I want to be able to see which of my friends are currently online | So that I know who I can instantly connect with |
| As an authenticated user | I want to be able to see the recent posts of all my friends | So that I can keep up with what is happening in their lives |

## Нефункционални изисквания

Сайтът трябва да:

* Бъде достъпен през всяка част на денонощието
* Да дава бърз отговор при заявка към сървърната част
* Да работи бързо и да предоставя добър User Experience на клиентската част
* Да пази личните данни на потребителите и да не ги издава без тяхното съгласие.

# Използвани библиотеки и модули

Всички описани библиотеки са достъпни за изтегляне през npm.

## Сървърна част

* Aws-sdk - за съхраняване на снимки на cloud
* Bcryptjs - за хеширане на потребителски пароли
* Body-parser – за форматиране на настъпващи към сървъра данни
* Cors – за извършване на Cross Origin Requests
* Express – гръбнакът на сървърната част
* Express-boom – за форматиране на friendly error responses
* Faker – генериране на dummy данни при тестване
* JsonWebToken – генериране на токени за user authentication
* Lodash – helper javascript methods
* MongoDb – nosql database
* Mongoose – ODM над MongoDb
* Mongoose-id-validator – валидатор за mongoose който проверява дали при създаване на връзка към друг документ документът съществува
* Mongoose-unique-validator – дава friendly exception message при unique mongodb exception
* Socketio – библиотека за използване на websockets при комуникация клиент-сървър
* Socketio-auth – библиотека за аутентикация на потребители, свързани със сървъра чрез websockets
* Uuid – библиотека за генериране на уникални идентификатори
* Validator – библиотека, която предоставя допълнителни методи за валидация

## Клиентка част

* Axios – библиотека за http заявки
* Babel – библиотека за транспилиране на JS
* Immutable – библиотека за поддържане на Immutable колекции
* Material-UI – библиотека, която предоставя множество UI компоненти
* Normalizr – за нормализиране на данни
* React – гръбнакът на клиентската част
* Redux – state management библиотека
* Redux-Form – имплементация на two way form binding чрез Redux
* Redux-form-material-ui – интеграция на някои компоненти в Material UI с Redux-Form
* Redux-Saga – redux middleware за асинхронни операции
* Reselect – библиотека за кеширане на данни
* Socketio-client – клиентска имплементация на Socketio

# Нетривиални аспекти на архитектурата

## Сървърна част

Сървърът представлява стандартен REST API и спазва основните Restful принципи. За абстракция над връзката с базата се използват Mongoose модели. В допълнение на http routes се използват и websockets за връзка в реално време с клиента.

## Клиентска част

Клиентът е SPA написан на React, заедно с Redux за state management. За да се осигури immutability на предоставения от Redux store се използва библиотеката Immutable.js. Данните във store-a се нормализират, за да се улесни работата с тях. За да се увеличи производителността на приложението се кешират данни, които компонентите извличат от Redux store-a чрез Reselect. Redux-Saga се използва за справяне с асинхронни заявки към сървърната част. За справяне със данните предоставени чрез websockets се използват event channels, които осигуряват потока на данни от сървъра директно към съответния Redux store.

# Описание на REST API архитектурата

Спазват се всички RESTful принципи. За достъп до колекция от ресурси (profiles, post, likes, comments) се използват стандартните Http методи (GET, PUT, POST, DELETE) в комбинация със уникалния идентификатор на достъпвания ресурс. Използват се и пътища за достъп до синглетони (например профилът на логнат user, и съответният like/dislike на user-a за даден пост).

# Описание на начина на инсталиране и конфигуриране на разработената система.

За инсталиране на необходимите библиотеки и модули потребителят трябва да навигира до папките server и client в проекта и да изпълни командата **npm install.** За стартиране на проекта, потребителят трябва първо да пусне инстанция на mongodb на машината си, след което да навигира до папките client и server и да изпълни командата **npm start.** След това сайтът е достъпен на следния URL: http//localhost:3000

# Заключение - срещнати/преодолени трудности и перспективи за бъдещо развитие на системата.

При писането на приложението основните трудности бяха свързани с поддържането на висока продуктивност и писането на качествен код. В бъдеще перспективите за развитие на системата са следните:

* Разширяване на функционалността на приложението – например имплементиране на user messaging и потребителски страници
* Подобряване на производителността – оптимизиране на клиентската част чрез максмимално намаляване на пререндерирането на компоненти, имплементиране на Infinite Scroll/ Paging за намаляване на потока на данни от сървъра към клиента

# Използвани материали

* React docs - <https://reactjs.org/docs/hello-world.html>
* Redux docs - <https://redux.js.org/>
* Immutable docs - <https://facebook.github.io/immutable-js/docs/>
* Express docs – https://expressjs.com/en/api.html
* Socketio docs - https://socket.io/docs/