Jitter

*Изготвили:*

*Драгомир Любомиров Пройчев, ФН 81251*

*Георги Мартинов Божинов, ФН 81333*

1. Функционалност.
   1. 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 |

* 1. 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

1. Използвани билиотеки и модули

Всички описани библиотеки са достъпни за изтегляне през npm.

* 1. 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
  1. 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-Saga – redux middleware за асинхронни операции
* Reselect – библиотека за кеширане на данни
* Socketio-client – клиентска имплементация на Socketio

1. Нетривиални аспекти на архитектурата
   1. 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.

* 1. 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 chanels, които осигуряват потока на данни от сървъра директно към съответния Redux store.

1. 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).

1. 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

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

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

* Разширяване на функционалността на приложението – например имплементиране на user messaging
* 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

1. 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/