Министерство образования и науки РФ

Московский Авиационный институт
(Национальный исследовательский университет)

Институт №8
«Информационные технологии и прикладная математика»

Кафедра 813
«Компьютерная математика»

Курсовой проект по дисциплине «Базы данных»

Тема: «Веб-приложение для тестирования»

Студент: Васильев Дмитрий Олегович

Группа: М8О-310Б-18

Преподаватель: Романенков Александр Михайлович

Дата: 13 октября 2020 г.

Оценка: _____

Подпись преподавателя: _____

Подпись студента: _____

Москва 2020

Содержание

1 Введение

1.1 Формальные требования

1. Необходимо выбрать предметную область для создания базы данных. Выбранная предметная область должна быть уникальной для всего потока, а не только в рамках учебной группы.

2. Необходимо описать таблицы и их назначение. Выполнить проектирование логической структуры базы данных. Описать схему базы данных. Все реальные таблицы должны иметь 3 нормальную форму или выше. База данных должна иметь минимум 5 таблиц.

3. Необходимо разработать два клиентских приложения для доступа к базе данных. Данные приложения должны быть написаны на двух разных языках программирования и иметь разный интерфейс (например, классическое оконное приложение и web-приложение). Выбор языков программирования произволен.

4. Необходимо организовать различные роли пользователей и права доступа к данным. Далее, необходимо реализовать возможность создания архивных копий и восстановления данных из клиентского приложения.

5. При разработке базы данных следует организовать логику обработки данных не на стороне клиента, а, например, на стороне сервера, базы данных, клиентские приложения служат только для представления данных и тривиальной обработки данных.

6. Ваша база данных должна иметь представления, триггеры и хранимые процедуры, причем все эти объекты должны быть осмысленны, а их использование оправдано.

7. При показе вашего проекта необходимо уметь демонстрировать таблицы, представления, триггеры и хранимые процедуры базы данных, внешние ключи, ограничения целостности и др. В клиентских приложениях уметь демонстрировать подключение к базе данных, основные режимы работы с данными (просмотр, редактирование, обновление ...)

8. Необходимо реализовать корректную обработку различного рода ошибок, которые могут возникать при работе с базой данных.

1.2 Клиентские приложения

Оба клиента будут SPA приложениями, которые общаются с сервером посредством REST API.

1.2.1 Тестирующая система

Данное приложение даёт возможность пользователю:

- создавать, редактировать, комбинировать, удалять тесты.
- рассылать приглашения на прохождения тестов.

1.2.2 Система управления

Данное приложение доступно только для администратора. Оно даёт ему следующие возможности:

- рассылать email–рассылку.
- банить тесты, пользователей.
- удалять тесты, пользователей.

1.3 Предметная область

Область применения данного приложения универсальна. Можно использовать тестирование на сотрудниках, школьниках, студентах и так далее.

1.4 Стэк технологий

- *JavaScript* [8] — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта *ECMAScript* [2].
- *React* [14] — *JavaScript* [8] библиотека для создания пользовательских интерфейсов.
- *Redux* [16] — контейнер состояния для *JavaScript* [8] приложения.
- *React Router* [15] — набор навигационных компонентов.
- *SCSS* [17] — препроцессор, который расширяет *CSS* [1].

- *CSS* [1] — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.
- *HTML* [7] — гипертекстовый язык разметки.
- *Node.js* [11] — среда выполнения JavaScript, созданная на основе движка Chrome V8 JavaScript.
- *Express* [4] — минимальный и гибкий *Node.js* [11] фреймворк для создания веб-приложений.
- *Python* [13] — язык программирования, который позволяет быстро работать и более эффективно интегрировать системы.
- *PostgreSQL* [12] — объектно-реляционная база данных с открытым исходным кодом.
- *Sequelize* [19] — *Node.js* [11] ORM на основе обещаний для Postgres *PostgreSQL* [12].

1.5 Инструменты

- *Git* [5] — система контроля версий.
- Postman — платформа совместной разработки API
- IDEs: — интегрированная среда разработки.
  - WebStorm
  - DataGrip
- Линтеры — программы, которые следят за качеством кода.
  - *ESLint* [3] — проверяет качество кода на *JavaScript* [8].
  - *Stylelint* [20] — проверяет качество кода на *SCSS* [17], *CSS* [1].
- Тестирующие фреймворки:
  - *Jest* [9] — среда тестирования *JavaScript* [8] с упором на простоту.
  - *Selenium with Python* [18] — предоставляют простой API для написания тестов с использованием Selenium WebDriver.

Webpack [21] — сборщик статических модулей для современных *JavaScript* [8] приложений.

2 Инфраструктура проекта

2.1 Архитекрутра

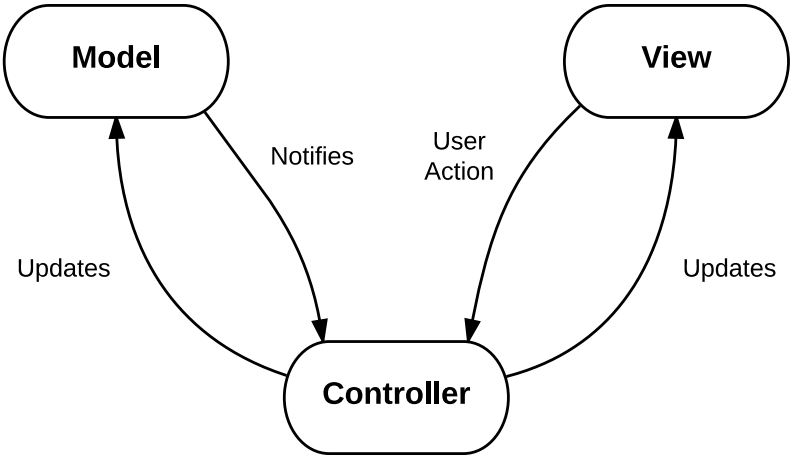

Рис. 1: Визуализация архитектуры MVC

За основу берётся архитектурный паттерн MVC. Он предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Контроллер – таким образом, что модификация каждого компонента может осуществляться независимо.

- **Модель** — предоставляет собой объектную модель некой предметной области, включает в себя данные и методы работы с этими данными, реагирует на запросы из контроллера, возвращая данные и/или изменяя своё состояние. При этом модель не содержит в себе информации о способах визуализации данных или форматах их представления, а также не взаимодействует с пользователем напрямую.

- **Представление** — отвечает за отображение информации. Одни и те же данные могут представляться различными способами и в различных форматах. Например, коллекцию объектов при помощи разных представлений можно представить на уровне пользовательского интерфейса как в табличном виде, так и списком.

- **Контроллер** — обеспечивает связь между пользователем и системой, использует модель и представление для реализации необходимой реакции на действия пользователя. Как правило, на уровне контроллера осуществляется фильтрация полученных данных и авторизация — проверяются права пользователя на выполнение действий или получение информации.

2.1.1 Связь логических компонетов и применяемых технологих

- **Model** — *Sequelize* [19] для сервера и *Redux* [16] для клиента.
- **View** — *React* [14]
- **Controller** — *Express* [4]

2.2 Сущности

Всегда перед проектированием проекта описывают сущности и их атрибуты. На основе данной информации будет стоится база данных. В моём случае они следующие:

Сущность	**Атрибуты**
Пользователь	• ID пользователя • Роли • Логин • Пароль • Email • Дата создания
Тест	• ID теста • Название • Описание • Теги • Контент • Ответы • Дата создания • Дата последнего изменения
Тег	• ID тега • Название

Попытка	• ID попытки • ID пользователя • ID теста • Результат • Ответы пользователя • Дата прохождения • Длительность прохождения попытки
Роль	• ID роли • Название

Таблица 1: Описание сущностей и атрибутов

2.3 Сборка и запуск

Все процессы отвечающие за сборку и запуск приложения я разделил на подзадачи. Каждая такая подзадача является npm скриптом. Они все описываются в файле package.json. Также среди данных скриптов можно выделить две группы – Development и Production.

2.3.1 Development

Скрипты из данной группы отвечают за то, чтобы приложение можно было запускать в режиме разработки, а именно:

1. сборка клинтской части не занимало слишком много времени
2. клинет пересобирался при изменении какого-либо файла
3. сервер перезапускался при изменения кода серверверной части
4. в браузере были доступны source map

2.3.2 Production

Скрипты из данной группы отвечают за то, чтобы приложение можно было запускать в режиме с максимальными оптимизациями, а именно:

1. минификация статический файлов
2. оптимизация работы библиотек
3. сборка серверной части

2.4 Деплоинг

Приложение разворачивается в системе *Heroku* [6]. Там же работает СУБД.

2.5 Организация работы с *Git* [5]

2.5.1 Git Workflow

Для организации работы с системой контроля версий в проекте используется подход Git Workflow. Он нужен для согласованного и продуктивного выполнения работы.

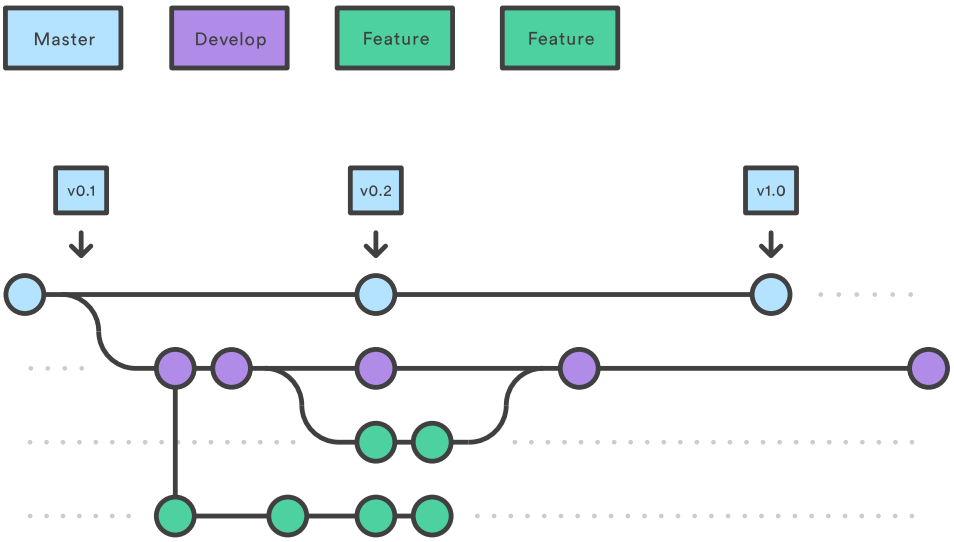

Рис. 2: Пример использования подхода Git Workflow

2.5.2 Git hooks

Чтобы в репозитории хранился код, который проходит проверки линтеров и тестовых фреймворков, нужно использовать Git Hooks. Они позволяют обработать события pre-commit, pre-push, post-commit и так далее.

Есть удобный пакет в npm – husky. Он позволяет определить в package.json обработку событий. В моём проекте нужно, чтобы на событие pre-commit выполняли проверки линтеры, а потом при успешном результате исполнялись unit-тесты. Также необходимо запускать selenium-тесты при событии pre-push.

```
{
    "hooks": {
        "pre-commit": "yarn es-lint && yarn style-lint && yarn test",
        "pre-push": "./venv/bin/pytest tests"
    }
}
```

Listing 1: Настройки для Git Hooks

3 Описание проекта

3.1 Разработка дизайна

Так как я не дизайнер, то мне нужно оперировать концептами и эскизами интерфейса. Поэтому вначале я сделал макет страниц и связь между ними.

3.2 Авторизации через JSON Web Token (JWT)

3.2.1 Авторизация

Авторизация — это процесс предоставления определённому лицу или группе лиц прав на выполнение определённых действий. Также сюда входит проверка данных, прав при попытке выполнения этих действий.

3.2.2 Аутентификация

Аутентификация — процедура проверки подлинности данный.

3.2.3 JSON Web Token (JWT)

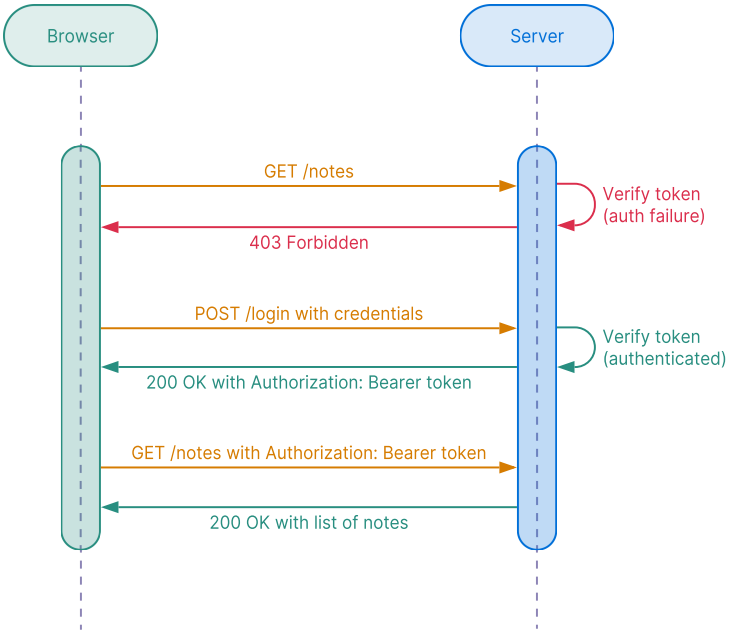

Рис. 3: Демонстрация работы JWT

JSON Web Token (JWT) — это открытый стандарт (RFC 7519), который определяет способ для безопасной передачи информации между сторонами с помощью JSON объектов. Эту информацию можно проверить, потому что она имеет цифровую подпись.

Вот несколько сценариев, в которых полезен JWT:

- **Авторизация** — это наиболее распространенный сценарий использования JWT. После того, как пользователь вошел в систему, каждый последующий запрос будет включать JWT, позволяя пользователю получать доступ к маршрутам, службам и ресурсам, разрешенным с помощью этого токена.
- **Обмен информацией** — JWT хороший способ безопасной передачи информации между сторонами. Поскольку JWT могут быть подписаны, например, с использованием пар открытого и закрытого ключей, вы можете быть уверены, что отправители являются теми, кем они себя называют. Кроме того, поскольку подпись рассчитывается с использованием **заголовка** и **полезных данных**, вы также можете убедиться, что содержимое не было изменено.

JWT состоит из следующих частей:

- **Заголовок** — содержит информацию о том, как должна вычисляться подпись. Обычно состоит из двух частей: типа токена, которым является JWT, и используемого алгоритма подписи, такого как HMAC SHA256 или RSA.
- **Полезные данные** — это данные, которые хранятся внутри JWT. Они также называют JWT-claims (заявки). Список доступных полей для JWT доступен на Wiki.
- **Подпись** — используется для проверки того, что сообщение не было изменено в процессе. В компактной форме JWT является сторой, которая состоит из трех частей, разделенных точками. Псевдокод вычисления подписи:

```
SECRET_KEY = 'some string';
unsignedToken = encodeBase64Url(header) + '.' + encodeBase64Url(payload)
signature = SHA256(unsignedToken, SECRET_KEY);

// собираем всё вместе
jwt = encodeBase64Url(header) + '.' + encodeBase64Url(payload) + '.' + encodeBase64Url(signature);
```

3.2.4 Реализация

Авторизация проходит следущим образом:

1. Пользователь делает `POST /api/signup` запрос на регистрацию. Если всё нормально, то в базе данных создаётся запись с данными пользователя.
2. Пользователь делает `POST /api/signin` запрос на аутентификацию. Если данные

верные, то высылается JWT вместе с состоянием пользователя (логин, почта). Когда ответ с сервера получен, то JWT сохраняется в localStorage (долговременное хранилище), а состояние передается в глобальное *Redux* [16] хранилище.

3. После того, как состояние глобального хранилища обновилось, приложение обновляет интерфейс.

Если перезагрузить веб-страницу, то *Redux* [16] хранилище обнуляется. Поэтому нам нужно сделать следующее: при запуске приложения проверять на валидность JWT, который лежит в localStorage. Это делается через `POST /api/init` запрос. Если токен валидный, то переавторизовываем пользователя. Иначе перенаправляем на главную страницу.

Далее этот токет будет использоваться для доступа к защищёнными ресурсам.

3.3 Схема базы данных

Схема базы данных

4 Заключение

Благодаря данному курсовому проекту, я поверхностно освоил разработку SPA приложений с помощью библиотеки *React* [14] и фреймворка *Express* [4].

4.1 Недостатки

Подводя итоги, мне бы хотелось перечислить вещи, на которые я буду обращать внимание при разработке следующих проектов:

- Использование CI/CD.
- Использование методологий в вёрстке. Например, Block Element Modificator (BEM). Её разработали внутри компании Яндекс. У них есть свой стек технологий под данную методологию, который облегчает разработку клиентской части.
- Использование NoSQL баз данных вместе с реалиционными. Хранить JSON в таблице плохо, поэтому для этой задачи подходить *MongoDB* [10].
- Разделение клиентского кода на чанки.
- Микросервисная архитектура.
- Использование TypeScript или Flow. Во время разработки я отказался от TypeScript из-за того, что надо очень много времени тратить на type-hinting.

A Визуализации структуры проекта

```
 /db-course-project-app
--- .gitignore
--- package.json
--- yarn-error.log
--- .babelrc
--- jest.config.js
--- requirements.txt
--- .eslintignore
--- yarn.lock
--- .env
--- README.md
--- LICENSE
--- /util
------ nodemon.json
--- /tests
------ test_smoke.py
--- /src
------ setupTests.js
------ .stylelintrc
------ config.js
------ index.js
------ webpack.config.js
------ .eslintrc.js
------ /client
--------- main.jsx
--------- admin.jsx
--------- /containers
------------ /TestEditorQuestionList
--------------- index.js
--------------- TestEditorQuestionList.jsx
------------ /TestEditorTagList
--------------- TestEditorTagList.jsx
```

```
--------------- index.js
--------- /reducers
------------ auth.js
------------ index.js
------------ testEditor.js
--------- /tests
------------ smoke.test.js
------------ /services
--------------- auth.test.js
--------- /store
------------ store.js
------------ index.js
--------- /helpers
------------ header.js
------------ question.js
------------ loader.js
------------ token.js
--------- /apps
------------ /main
--------------- App.jsx
--------------- /pages
------------------ /ProfileSettings
---------------------- style.scss
---------------------- ProfileSettings.jsx
---------------------- index.js
---------------------- /components
------------------------ /DeleteProfileForm
--------------------------- DeleteProfileForm.jsx
--------------------------- index.js
------------------------ /UpdatePasswordForm
--------------------------- index.js
--------------------------- UpdatePasswordForm.jsx
------------------ /ProfileTests
---------------------- style.scss
```

```
---------- ProfileTests.jsx
---------- index.js
---------- /SignUp
---------- style.scss
---------- index.js
---------- SignUp.jsx
---------- /TestEditor
---------- style.scss
---------- index.js
---------- TestEditor.jsx
---------- /Profile
---------- style.scss
---------- Profile.jsx
---------- index.js
---------- /ProfileAttempts
---------- style.scss
---------- ProfileAttempts.jsx
---------- index.js
---------- /Test
---------- /Home
---------- style.scss
---------- index.js
---------- Home.jsx
---------- /Login
---------- style.scss
---------- Login.jsx
---------- index.js
---------- /components
---------- /TagList
---------- style.scss
---------- index.js
---------- TagList.jsx
---------- /Tag
---------- style.scss
```

---------- index.js

---------- Tag.jsx

---------- /Header

---------- style.scss

---------- Header.jsx

---------- index.js

---------- /TestCard

---------- style.scss

---------- index.js

---------- TestCard.jsx

---------- /AnswerEditList

---------- style.scss

---------- config.js

---------- index.js

---------- AnswerEditList.jsx

---------- /ListTestCards

---------- style.scss

---------- index.js

---------- ListTestCards.jsx

---------- /QuestionEditItem

---------- QuestionEditItem.jsx

---------- index.js

---------- /AnswerEditItem

---------- style.scss

---------- AnswerEditItem.jsx

---------- index.js

---------- /Footer

---------- style.scss

---------- Footer.jsx

---------- index.js

---------- /ErrorFormAlert

---------- ErrorFormAlert.jsx

---------- index.js

---------- /HttpErrorInfo

```
---------------------- style.scss
---------------------- HttpErrorInfo.jsx
---------------------- index.js
------------------ /QuestionEditList
---------------------- style.scss
---------------------- index.js
---------------------- QuestionEditList.jsx
------------ /admin
--------------- Login.jsx
--------------- App.jsx
--------- /actions
------------ auth.js
------------ testEditor.js
--------- /services
------------ auth.js
------------ editProfileSettings.js
------------ editTest.js
--------- /hoc
------------ /NotIsLoggedInRoute
--------------- index.js
--------------- NotIsLoggedInRoute.jsx
------------ /PrivateRoute
--------------- PrivateRoute.jsx
--------------- index.js
------ /routes
--------- main.js
--------- auth.js
--------- profileModify.js
--------- testEditor.js
------ /tests
--------- smoke.test.js
--------- /services
------ /controllers
--------- auth.js
```

```
--------- profileModify.js
--------- testEditor.js
------ /helpers
--------- FormListErrors.js
------ /templates
--------- admin.handlebars
--------- index.handlebars
--------- /layouts
------------ main.handlebars
--------- /partials
------------ favicon.handlebars
------------ loader.handlebars
------------ meta.handlebars
------ /middlewares
--------- checkToken.js
--------- errorHandler.js
------ /models
--------- TestTag.js
--------- index.js
--------- UserRole.js
--------- Role.js
--------- /Tag
------------ constraints.js
------------ index.js
------------ Tag.js
--------- /Test
------------ constraints.js
------------ index.js
------------ Test.js
--------- /User
------------ constraints.js
------------ User.js
------------ index.js
------ /services
```

```
--- /db
------ init_db.sql
```

B Код проекта

db-course-project-app/package.json

```
{
  "name": "db-course-project-app",
  "version": "1.0.0",
  "description": "Web-application for course project by Database.",
  "main": "build/index.js",
  "author": "Dmitry Vasiliev",
  "scripts": {
    "test": "jest",
    "start-dev": "nodemon --config \"./util/nodemon.json\"/",
    "build": "rm -rf ./build/* && babel src -d build && webpack --config src/webpack.config.js
    ↪  --mode=\"production\"",
    "start": "node -r dotenv/config build/index.js",
    "es-lint": "eslint . -c src/.eslintrc.js --ext \"jsx,js\"",
    "style-lint": "stylelint --ignore-pattern src/client/tests --config src/.stylelintrc src/client/*",
    "watch": "webpack --config src/webpack.config.js --watch"
  },
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^1.2.32",
    "@fortawesome/free-brands-svg-icons": "^5.15.1",
    "@fortawesome/free-regular-svg-icons": "^5.15.1",
    "@fortawesome/free-solid-svg-icons": "^5.15.1",
    "@fortawesome/react-fontawesome": "^0.1.11",
    "bcrypt": "^5.0.0",
    "bootstrap": "^4.5.0",
    "compression": "^1.7.4",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-handlebars": "^5.0.0",
    "http-status-codes": "^1.4.0",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.19",
    "morgan": "^1.10.0",
    "multer": "^1.4.2",
    "pg": "^8.3.0",
    "pg-hstore": "^2.3.3",
    "prop-types": "^15.7.2",
    "pug": "^3.0.0",
    "react": "^16.13.1",
    "react-bootstrap": "^1.2.2",
    "react-dom": "^16.13.1",
    "react-redux": "^7.2.1",
    "react-router-bootstrap": "^0.25.0",
    "react-router-dom": "^5.2.0",
    "react-router-prop-types": "^1.0.5",
    "redux": "^4.0.5",
    "sequelize": "^6.3.3",
    "serve-favicon": "^2.5.0"
  },
  "devDependencies": {
    "@babel/cli": "^7.11.6",
    "@babel/core": "^7.11.6",
    "@babel/node": "^7.10.5",
    "@babel/plugin-transform-runtime": "^7.11.5",
    "@babel/preset-env": "^7.11.5",
    "@babel/preset-react": "^7.10.4",
    "@types/compression": "^1.7.0",
    "@types/dotenv": "^8.2.0",
    "@types/express": "^4.17.7",
    "@types/express-handlebars": "^3.1.0",
    "@types/jest": "^26.0.4",
    "@types/jsonwebtoken": "^8.5.0",
    "@types/lodash": "^4.14.161",
    "@types/morgan": "^1.9.1",
    "@types/multer": "^1.4.4",
    "@types/node": "^14.0.22",
    "@types/react": "^16.9.43",
    "@types/react-dom": "^16.9.8",
    "@types/react-redux": "^7.1.9",
    "@types/react-router-bootstrap": "^0.24.5",
    "@types/react-router-dom": "^5.1.5",
```

```
    "@types/redux": "^3.6.0",
    "@types/sequelize": "^4.28.9",
    "@types/serve-favicon": "^2.5.0",
    "babel-loader": "^8.1.0",
    "css-loader": "^3.6.0",
    "eslint": "^7.9.0",
    "eslint-plugin-jest": "^24.0.1",
    "eslint-plugin-react": "^7.20.6",
    "husky": "^4.2.5",
    "jest": "^26.1.0",
    "jest-fetch-mock": "^3.0.3",
    "lorem-ipsum": "^2.0.3",
    "nodemon": "^2.0.4",
    "sass": "^1.26.10",
    "sass-loader": "^9.0.2",
    "style-loader": "^1.2.1",
    "stylelint": "^13.6.1",
    "stylelint-config-sass-guidelines": "^7.0.0",
    "webpack": "^4.43.0",
    "webpack-cli": "^3.3.12"
  },
  "husky": {
    "hooks": {
      "pre-commit": "yarn es-lint && yarn style-lint && yarn test",
      "pre-push": "./venv/bin/pytest tests"
    }
  }
}
```

db-course-project-app/.babelrc

```
{
  "presets": ["@babel/preset-env", "@babel/preset-react"],
  "plugins": [
    "@babel/plugin-transform-runtime"
  ]
}
```

db-course-project-app/jest.config.js

```
module.exports = {
  setupFiles: ["./src/setupTests.js"],
  testEnvironment: 'node',
  testRegex: '(/src/tests/|/src/client/tests/).*\\.(test|spec)?\\.(js|jsx)$',
  moduleFileExtensions: ['js', 'jsx', 'json', 'node']
};
```

db-course-project-app/requirements.txt

```
attrs==19.3.0
iniconfig==1.0.1
more-itertools==8.4.0
packaging==20.4
pluggy==0.13.1
py==1.9.0
pyparsing==2.4.7
pytest==6.0.1
selenium==3.141.0
six==1.15.0
toml==0.10.1
urllib3==1.25.10
```

db-course-project-app/README.md

```
# db-course-project-app
:book: Web-application for course project by Database
```

```
Link to description about project:
↪  [db-course-project-report](https://github.com/swimmwatch/db-course-project-report)
```

db-course-project-app/util/nodemon.json

```
{
  "watch": ["src"],
  "ext": "js",
  "ignore": ["src/public"],
  "exec": "babel-node -r dotenv/config src/index.js"
}
```

db-course-project-app/tests/test_smoke.py

```
def test_add():
    assert 2 + 2 == 4
```

db-course-project-app/src/setupTests.js

```
import fetchMock from "jest-fetch-mock";

fetchMock.enableMocks();
```

db-course-project-app/src/.stylelintrc

```
{
  "extends": "stylelint-config-sass-guidelines"
}
```

db-course-project-app/src/index.js

```
import * as path from "path";
import express from "express";
import exphbs from "express-handlebars";
import compression from "compression";
import morgan from "morgan";
import serveFavicon from "serve-favicon";
import { Sequelize } from "sequelize";

import config from "./config";

import mainRouter from "./routes/main";
import authRouter from "./routes/auth";
import testEditorRouter from "./routes/testEditor";
import profileModify from "./routes/profileModify";
import errorHandler from "./middlewares/errorHandler";

import * as models from "./models";

const app = express();

// set static path
app.use("/static", express.static("src/public"));

// set template engine
app.set('views', path.join(process.cwd(), '/src', '/templates'));
app.engine('handlebars', exphbs());
app.set('view engine', 'handlebars');

// set response compression
app.use(compression());
// set logger
app.use(morgan("common"));
```

```
// serve json requests
app.use(express.json());
// serve form requests
app.use(express.urlencoded({ extended: true }));
// serve favicon
app.use(serveFavicon(path.join(process.cwd(), '/src', '/public', 'favicon.ico')))

const PORT = process.env.PORT || 3000;

const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);

// app.get("/admin", (req, res) => {
//     res.render("admin");
// });

app.use("/api", authRouter);
app.use("/api/profile", profileModify);
app.use("/api/test", testEditorRouter);

app.use("*", mainRouter);

app.use(errorHandler);

app.listen(PORT, async () => {
    try {
        await sequelize.authenticate();
        await sequelize.sync({ force: true });

        for (let modelName in models) {
            await models[modelName].sync();
        }

        console.log('Connection has been established successfully.');
    } catch (error) {
        console.error('Unable to connect to the database:', error);
    }

    console.log(`Server started on port: ${PORT}.`);
});
```

db-course-project-app/src/webpack.config.js

```
const path = require('path');

module.exports = {
    mode: 'development',
    entry: {
        main: './src/client/main.jsx',
        admin: './src/client/admin.jsx'
    },
    devtool: 'source-map',
    module: {
        rules: [
            {
                test: /\.jsx?$/,
                use: 'babel-loader',
                exclude: /node_modules/,
            },
            {
                test: /\.s[ac]ss$/i,
                use: [
                    // Creates `style` nodes from JS strings
                    'style-loader',
                    // Translates CSS into CommonJS
                    'css-loader',
                    // Compiles Sass to CSS
                    'sass-loader',
                ],
            },
        ],
    },
    resolve: {
        extensions: [ '.jsx', '.js' ],
    },
```

```
    output: {
        filename: '[name].bundle.js',
        path: path.resolve(__dirname, 'public'),
    },
};
```

db-course-project-app/src/.eslintrc.js

```
module.exports = {
    "env": {
        "browser": true,
        "es2020": true,
        "node": true,
        "jest": true
    },
    "extends": [
        "eslint:recommended",
        "plugin:react/recommended",
        "plugin:jest/recommended"
    ],
    "parserOptions": {
        "ecmaFeatures": {
            "jsx": true
        },
        "ecmaVersion": 11,
        "sourceType": "module"
    },
    "plugins": [
        "react",
        "jest"
    ],
    "rules": {
    }
};
```

db-course-project-app/src/client/main.jsx

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { Provider } from "react-redux";
import { BrowserRouter } from "react-router-dom";
import { toggleLoader } from "./helpers/loader";

import App from "./apps/main/App";
import { store, initAuthStore } from "./store";

import "bootstrap/scss/bootstrap.scss";

initAuthStore(store).then(() => {
    toggleLoader();

    ReactDOM.render(
        <Provider store={store}>
            <BrowserRouter>
                <App />
            </BrowserRouter>
        </Provider>,
        document.getElementById("root")
    );
});
```

db-course-project-app/src/client/admin.jsx

```
import * as React from "react";
import * as ReactDOM from "react-dom";
import { BrowserRouter } from "react-router-dom";

import "bootstrap/scss/bootstrap.scss";

```

```
import App from "./apps/admin/App";

ReactDOM.render(
    <BrowserRouter>
        <App />
    </BrowserRouter>,
    document.getElementById("root")
);
```

...ourse-project-app/src/client/containers/TestEditorQuestionList/index.js

```
import { TestEditorQuestionList } from "./TestEditorQuestionList";

export default TestEditorQuestionList;
```

...src/client/containers/TestEditorQuestionList/TestEditorQuestionList.jsx

```
import {connect} from "react-redux";
import * as testEditorActions from "../../actions/testEditor";
import QuestionEditList from "../../../client/apps/main/components/QuestionEditList";

function mapStateToProps(state) {
    const { questions } = state.testEditor;

    return { questions };
}

function mapDispatchToProps(dispatch) {
    return {
        deleteQuestion: i => dispatch(testEditorActions.deleteQuestion(i)),
        updateQuestionTitle: (i, title) => dispatch(testEditorActions.updateQuestionTitle(i, title)),
        changeQuestionType: (i, typeAnswer) => dispatch(testEditorActions.changeQuestionType(i,
        ↪  typeAnswer)),
        appendAnswer: i => dispatch(testEditorActions.appendAnswer(i))
    };
}

const TestEditorQuestionList = connect(mapStateToProps, mapDispatchToProps)(QuestionEditList);

export { TestEditorQuestionList };
```

...oject-app/src/client/containers/TestEditorTagList/TestEditorTagList.jsx

```
import {connect} from "react-redux";
import TagList from "../../../client/apps/main/components/TagList";
import * as testEditorActions from "../../actions/testEditor";

function mapStateToProps(state) {
    const { tags } = state.testEditor.info;

    return { tags };
}

function mapDispatchToProps(dispatch) {
    return {
        deleteTag: (i) => dispatch(testEditorActions.deleteTag(i))
    };
}

const TestEditorTagList = connect(mapStateToProps, mapDispatchToProps)(TagList);

export { TestEditorTagList };
```

db-course-project-app/src/client/containers/TestEditorTagList/index.js

```
import { TestEditorTagList } from "./TestEditorTagList";

```

```
export default TestEditorTagList;
```

db-course-project-app/src/client/reducers/auth.js

```
import * as authActions from "../actions/auth";
import * as authService from "../services/auth";

let initState = { isLoggedIn: false, user: null }

export default (state = initState, action) => {
    const { isLoggedIn, user } = action;

    switch (action.type) {
        case authActions.LOGIN_APPROVE:
            return { isLoggedIn, user };
        case authActions.LOGIN_FAILED:
            return { isLoggedIn, user };
        case authActions.LOGOUT:
            authService.logOut();

            return { isLoggedIn, user };
        default:
            return state;
    }
};
```

db-course-project-app/src/client/reducers/index.js

```
import { combineReducers } from "redux";

import auth from "./auth";
import testEditor from "./testEditor";

export default combineReducers({ auth, testEditor });
```

db-course-project-app/src/client/reducers/testEditor.js

```
import * as testEditorActions from "../actions/testEditor";
import {createQuestion, createAnswer} from "../helpers/question";
import {ANSWER_TYPE} from "../apps/main/components/AnswerEditList/config";

let initState = {
    info: {
        title: '',
        description: '',
        tags: []
    },
    questions: [
        createQuestion(),
    ]
};

export default (state = initState, action) => {
    let newState = JSON.parse(JSON.stringify(state));

    switch (action.type) {
        case testEditorActions.RESET:
            return { ...initState };
        case testEditorActions.UPDATE:
            return action.content;
        case testEditorActions.UPDATE_TITLE: {
            newState.info.title = action.title;

            return newState;
        }
        case testEditorActions.UPDATE_DESCRIPTION: {
            newState.info.description = action.description;

            return newState;
```

```
        }
        case testEditorActions.APPEND_TAG: {
            const tagIsNotInList = !newState.info.tags.includes(action.tag);

            if (action.tag.length && tagIsNotInList) {
                newState.info.tags.push(action.tag);
            }

            return newState;
        }
        case testEditorActions.DELETE_TAG: {
            const { id } = action;

            newState.info.tags.splice(id, 1);

            return newState;
        }
        case testEditorActions.APPEND_QUESTION: {
            newState.questions.push(createQuestion());

            return newState;
        }
        case testEditorActions.DELETE_QUESTION: {
            const { id } = action;

            if (newState.questions.length > 1) {
                newState.questions.splice(id, 1);
            }

            return newState;
        }
        case testEditorActions.UPDATE_QUESTION_TITLE: {
            const { id, title } = action;

            newState.questions[id].title = title;

            return newState;
        }
        case testEditorActions.UPDATE_QUESTION_TYPE: {
            const { id, typeAnswer } = action;
            const { answers } = newState.questions[id];

            for (let currAnswer of answers) {
                currAnswer.isRight = false;
            }

            newState.questions[id].typeAnswer = typeAnswer;

            return newState;
        }
        case testEditorActions.APPEND_ANSWER: {
            const { questionId } = action;

            newState.questions[questionId].answers.push(createAnswer());

            return newState;
        }
        case testEditorActions.DELETE_ANSWER: {
            const { questionId, answerId } = action;
            const { answers } = newState.questions[questionId];

            if (answers.length > 2) {
                answers.splice(answerId, 1);
            }

            newState.questions[questionId].answers = answers;

            return newState;
        }
        case testEditorActions.UPDATE_ANSWER_TEXT: {
            const { questionId, answerId, value } = action;
            const { answers } = newState.questions[questionId];

            answers[answerId].content = value;

            return newState;
```

```
        }
        case testEditorActions.UPDATE_ANSWERS: {
            const { questionId, answerId, isRight, typeAnswer } = action;
            const { answers } = newState.questions[questionId];

            if (typeAnswer === ANSWER_TYPE.ONE) {
                for (let currAnswer of answers) {
                    currAnswer.isRight = false;
                }
            }

            answers[answerId].isRight = isRight;

            return newState;
        }
        default:
            return state;
    }
};
```

db-course-project-app/src/client/tests/smoke.test.js

```
describe("Sum", () => {
    it("addiction", () => {
        expect(6).toBe(6);
    });
});
```

db-course-project-app/src/client/tests/services/auth.test.js

```
import authService from "../../services/auth";
import {OK} from "http-status-codes";

describe("client sign up", () => {
    it("doesn't sign up when error list are", async () => {
        const formErrorList = {
            errors: [
                { message: 'test'}
            ]
        };

        fetch.mockReject(() => Promise.reject(formErrorList));

        await expect(authService.signUp()).rejects.toEqual(formErrorList);
    });

    it("sign up when API responds with status OK", async () => {
        fetch.mockResponse([
            null,
            { status: OK }
        ]);

        await expect(authService.signUp()).resolves.toEqual(undefined);
    });
});

describe("client sign in", () => {
    it("doesn't sign in when error list are", async () => {
        const formErrorList = {
            errors: [
                { message: 'test'}
            ]
        };

        fetch.mockReject(() => Promise.reject(formErrorList));

        await expect(authService.signIn()).rejects.toEqual(formErrorList);
    });

    // it("sign in when API responds with status OK", async () => {
    //     const respond = { test: 'test' };
    //
```

```
    //     fetch.mockResponse([
    //         JSON.stringify(respond),
    //         { status: OK }
    //     ]);
    //
    //     await expect(authService.signIn()).resolves.toEqual(respond);
    // });
});
```

db-course-project-app/src/client/store/store.js

```
import { createStore } from "redux";
import rootReducer from "../reducers";
import * as authActions from "../actions/auth";
import {getToken} from "../helpers/token";
import {appendAuth} from "../helpers/header";

const store = createStore(
    rootReducer,
    window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__()
);

export const initAuthStore = async (store) => {
    const token = getToken();

    if (!token) {
        store.dispatch(authActions.failed());
    } else {
        let response = null;
        try {
            const headers = new Headers();

            appendAuth(headers, token);

            response = await fetch('/api/init', { headers, method: 'POST' });
        } catch (err) {
            console.log(err);
        }

        if (response.ok) {
            const responseJson = await response.json();
            const { user } = responseJson;

            store.dispatch(authActions.success(user));
        } else {
            store.dispatch(authActions.failed());
        }
    }
};

export default store;
```

db-course-project-app/src/client/store/index.js

```
import store, { initAuthStore } from "./store";

export { store, initAuthStore };
```

db-course-project-app/src/client/helpers/header.js

```
export function appendAuth(headers, token) {
    headers.append('Authorization', token);

    return headers;
}

export function appendJSON(headers) {
    headers.append('Accept', 'application/json');
    headers.append('Content-Type', 'application/json');
```

```
10
11      return headers;
12  }
```

db-course-project-app/src/client/helpers/question.js

```
1  import {ANSWER_TYPE} from "../apps/main/components/AnswerEditList/config";
2
3  export const createAnswer = (content = '', isRight = false) => {
4      return {content, isRight};
5  };
6
7  export const createQuestion = (title = '',
8                                 typeAnswer = ANSWER_TYPE.ONE,
9                                 answers = [createAnswer(), createAnswer()]) => {
10     return {title, typeAnswer, answers};
11 };
```

db-course-project-app/src/client/helpers/loader.js

```
1  export function toggleLoader() {
2      const loader = document.querySelector('.loader');
3
4      loader.classList.toggle('loader_hide');
5  }
```

db-course-project-app/src/client/helpers/token.js

```
1  const TOKEN_ID = 'TOKEN';
2
3  export function getToken() {
4      return localStorage.getItem(TOKEN_ID);
5  }
6
7  export function removeToken() {
8      localStorage.removeItem(TOKEN_ID);
9  }
```

db-course-project-app/src/client/apps/main/App.jsx

```
1  import * as React from 'react';
2  import { Switch, Route } from 'react-router-dom';
3  import PrivateRoute from "../../hoc/PrivateRoute";
4  import NotIsLoggedInRoute from "../../hoc/NotIsLoggedInRoute";
5
6  import Container from "react-bootstrap/Container";
7
8  import Login from "./pages/Login";
9  import Home from "./pages/Home"
10 import SignUp from "./pages/SignUp";
11 import TestEditor from "./pages/TestEditor";
12
13 import Footer from "./components/Footer";
14 import Header from "./components/Header";
15 import Profile from "./pages/Profile";
16 import HttpErrorInfo from "./components/HttpErrorInfo";
17 import {NOT_FOUND} from "http-status-codes";
18
19 class App extends React.Component {
20     render() {
21         return (
22             <div>
23                 <Header />
24
25                 <Switch>
26                     <Route exact path='/' component={Home}/>
27                     <NotIsLoggedInRoute path='/login' component={Login}/>
```

```
                    <NotIsLoggedInRoute path='/signup' component={SignUp}/>
                    <PrivateRoute path='/profile' component={Profile}/>
                    <PrivateRoute exact path='/test/edit' component={TestEditor} />
                    <Route component={() => <HttpErrorInfo status={NOT_FOUND} />} />
                </Switch>

                <Container className="p-3">
                    <Footer/>
                </Container>
            </div>
        );
    }
}

export default App;
```

...ourse-project-app/src/client/apps/main/pages/ProfileSettings/style.scss

```

```

...ject-app/src/client/apps/main/pages/ProfileSettings/ProfileSettings.jsx

```
import * as React from "react";
import Container from "react-bootstrap/Container";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import ErrorFormAlert from "../../components/ErrorFormAlert";
import UpdatePasswordForm from "./components/UpdatePasswordForm";
import DeleteProfileForm from "./components/DeleteProfileForm";

import "./style.scss";

class ProfileSettings extends React.Component {
    constructor(props) {
        super(props);

        this.state = {
            listErrors: []
        }

        this.updateListErrors = this.updateListErrors.bind(this);
        this.hideErrorAlert = this.hideErrorAlert.bind(this);
    }

    updateListErrors(errors) {
        this.setState({ listErrors: errors })
    }

    hideErrorAlert() {
        this.setState({ listErrors: [] });
    }

    render() {
        const { listErrors } = this.state;

        return (
            <Container className="p-3">
                <Row>
                    <ErrorFormAlert listErrors={listErrors}
                                    show={listErrors.length !== 0}
                                    onHide={this.hideErrorAlert} />
                </Row>
                <Row>
                    <Col lg={6}>
                        <UpdatePasswordForm onSubmitError={this.updateListErrors} />
                    </Col>
                </Row>
                <Row>
                    <Col lg={6}>
                        <DeleteProfileForm />
                    </Col>
```

```
                </Row>
            </Container>
        );
    }
}

export default ProfileSettings;
```

...-course-project-app/src/client/apps/main/pages/ProfileSettings/index.js

```
import ProfileSettings from "./ProfileSettings";

export default ProfileSettings;
```

...ages/ProfileSettings/components/DeleteProfileForm/DeleteProfileForm.jsx

```
import * as React from "react";
import { withRouter } from "react-router-dom";
import { connect } from "react-redux";
import PropTypes from "prop-types";
import ReactRouterPropTypes from "react-router-prop-types";
import Button from "react-bootstrap/Button";
import Modal from "react-bootstrap/Modal";
import * as authActions from "../../../../../../actions/auth";
import * as editProfileSettings from "../../../../../../services/editProfileSettings";

class DeleteProfileForm extends React.Component {
    constructor(props) {
        super(props);

        this.state = {
            modalShow: false
        }

        this.handleSubmit = this.handleSubmit.bind(this);
        this.toggleWarningModal = this.toggleWarningModal.bind(this);
    }

    toggleWarningModal() {
        this.setState(prev => {
            return {
                modalShow: !prev.modalShow
            };
        });
    }

    async handleSubmit() {
        const { history, dispatch } = this.props;

        try {
            await editProfileSettings.remove();

            dispatch(authActions.logOut());

            history.push('/');
        } catch (err) {
            console.log(err);
        }
    }

    render() {
        const { modalShow } = this.state;

        return (
            <>
                <h5>Danger zone:</h5>
                <Button variant="danger"
                        onClick={this.toggleWarningModal}>Delete profile</Button>

                <Modal
                    size="md"
                    aria-labelledby="contained-modal-title-vcenter"
```

```
                    centered
                    onHide={this.toggleWarningModal}
                    show={modalShow}
                >
                    <Modal.Header closeButton>
                        <Modal.Title id="contained-modal-title-vcenter">Delete profile</Modal.Title>
                    </Modal.Header>
                    <Modal.Body>
                        <p>Are you sure you want to delete profile?</p>
                    </Modal.Body>
                    <Modal.Footer>
                        <Button onClick={this.handleSubmit}>Yes</Button>
                    </Modal.Footer>
                </Modal>
            </>
        );
    }
}

DeleteProfileForm.propTypes = {
    onSubmitError: PropTypes.func,
    history: ReactRouterPropTypes.history,
    dispatch: PropTypes.func
};

const connectedDeleteProfileForm = connect()(DeleteProfileForm);

export default withRouter(connectedDeleteProfileForm);
```

...t/apps/main/pages/ProfileSettings/components/DeleteProfileForm/index.js

```
import DeleteProfileForm from "./DeleteProfileForm";

export default DeleteProfileForm;
```

.../apps/main/pages/ProfileSettings/components/UpdatePasswordForm/index.js

```
import { UpdatePasswordForm } from "./UpdatePasswordForm";

export default UpdatePasswordForm;
```

...es/ProfileSettings/components/UpdatePasswordForm/UpdatePasswordForm.jsx

```
import * as React from "react";
import { connect } from "react-redux";
import { withRouter } from "react-router-dom";
import ReactRouterPropTypes from "react-router-prop-types";
import PropTypes from "prop-types";
import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";
import * as authActions from "../../../../../../actions/auth";
import * as editProfileSettings from "../../../../../../services/editProfileSettings";

class UpdatePasswordForm extends React.Component {
    constructor(props) {
        super(props);

        this.state = {
            password: '',
            newPassword: '',
            repeatNewPassword: '',

            isLoading: false
        };

        this.handleInputChange = this.handleInputChange.bind(this);
        this.handleFormSubmit = this.handleFormSubmit.bind(this);
        this._generateFormData = this._generateFormData.bind(this);
        this.toggleLoadingState = this.toggleLoadingState.bind(this);
```

```
    }

    handleInputChange(event) {
        const { name, value } = event.target;

        this.setState({ [name]: value });
    }

    _generateFormData() {
        const formData = new FormData();
        formData.append('password', this.state.password);
        formData.append('newPassword', this.state.newPassword);
        formData.append('repeatNewPassword', this.state.repeatNewPassword);

        return formData;
    }

    async handleFormSubmit(event) {
        event.preventDefault();

        const { dispatch, history, onSubmitError } = this.props;

        this.toggleLoadingState();

        try {
            const formData = this._generateFormData();
            await editProfileSettings.updatePassword(formData);

            dispatch(authActions.logOut());

            history.push("/login");
        } catch ({ errors }) {
            this.toggleLoadingState();

            onSubmitError(errors);
        }
    }

    toggleLoadingState() {
        this.setState(prev => {
            return { isLoading: !prev.isLoading }
        });
    }

    render() {
        const { isLoading } = this.state;

        return (
            <>
                <h5>Update password:</h5>
                <Form>
                    <Form.Group controlId="profile-settings-form__password">
                        <Form.Control type="password"
                                      placeholder="Enter current password"
                                      name="password"
                                      onChange={this.handleInputChange} />
                    </Form.Group>
                    <Form.Group controlId="profile-settings-form__repassword">
                        <Form.Control type="password"
                                      placeholder="Enter new password"
                                      name="newPassword"
                                      onChange={this.handleInputChange} />
                    </Form.Group>
                    <Form.Group controlId="profile-settings-form__repassword">
                        <Form.Control type="password"
                                      placeholder="Enter new password again"
                                      name="repeatNewPassword"
                                      onChange={this.handleInputChange} />
                    </Form.Group>
                    <Button variant="primary"
                            type="submit"
                            disabled={isLoading}
                            onClick={this.handleFormSubmit}>{ isLoading ? 'Loading...' : 'Save' }</Button>
                </Form>
            </>
        );
```

```
    }
}

UpdatePasswordForm.propTypes = {
    onSubmitError: PropTypes.func.isRequired,
    dispatch: PropTypes.func,
    history: ReactRouterPropTypes.history
};

const connectedUpdatePasswordForm = connect()(UpdatePasswordForm);
const connectedUpdatePasswordFormWithRouter = withRouter(connectedUpdatePasswordForm);

export { connectedUpdatePasswordFormWithRouter as UpdatePasswordForm };
```

...b-course-project-app/src/client/apps/main/pages/ProfileTests/style.scss

```

```

...se-project-app/src/client/apps/main/pages/ProfileTests/ProfileTests.jsx

```
import * as React from "react";
import Container from "react-bootstrap/Container";
import ListTestCards from "../../components/ListTestCards";
import * as editTest from "../../../../services/editTest";

import "./style.scss";

class ProfileTests extends React.Component {
    constructor(props) {
        super(props);

        this.state = {
            profileTests: []
        };

        this.handleDeleteTestCard = this.handleDeleteTestCard.bind(this);
    }

    async handleDeleteTestCard(testId) {
        try {
            await editTest.deleteTest(testId);
        } catch (err) {
            console.error(err);
        }

        // delete test card from list
        this.setState(prev => {
            const { profileTests } = prev;

            const delI = profileTests.map(test => test.testId)
                                     .indexOf(testId);

            return {
                profileTests: [
                    ...profileTests.slice(0, delI),
                    ...profileTests.slice(delI + 1),
                ]
            }
        });
    }

    async componentDidMount() {
        let responseJson = null;
        try {
            responseJson = await editTest.getOwnTests();
        } catch (err) {
            console.error(err);
        }

        this.setState({
            profileTests: responseJson
```

```
        });
    }

    render() {
        const { profileTests } = this.state;

        return (
            <Container className="p-3">
                <ListTestCards tests={profileTests}
                               onDeleteTestCard={this.handleDeleteTestCard}/>
            </Container>
        );
    }
}

export default ProfileTests;
```

db-course-project-app/src/client/apps/main/pages/ProfileTests/index.js

```
import ProfileTests from "./ProfileTests";

export default ProfileTests;
```

db-course-project-app/src/client/apps/main/pages/SignUp/style.scss

```
.main-signup-form {
  &__title {
    text-align: center;
  }

  &__label {
    font-weight: bold;
  }
}
```

db-course-project-app/src/client/apps/main/pages/SignUp/index.js

```
import SignUp from "./SignUp";

export default SignUp;
```

db-course-project-app/src/client/apps/main/pages/SignUp/SignUp.jsx

```
import * as React from "react";
import ReactRouterPropTypes from "react-router-prop-types";
import { withRouter } from "react-router-dom";

import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";
import Container from "react-bootstrap/Container";
import Col from "react-bootstrap/Col";
import authService from "../../../../services/auth";

import ErrorFormAlert from "../../components/ErrorFormAlert";

import userConstraints from "../../../../../../models/User/constraints";

import "./style.scss";

const {
    MIN_PASSWORD_LENGTH,
    MAX_PASSWORD_LENGTH,
    MIN_LOGIN_LENGTH,
    MAX_LOGIN_LENGTH,
    MAX_EMAIL_LENGTH
} = userConstraints;

```

```
25 class SignUp extends React.Component {
26     constructor(props) {
27         super(props);
28 
29         this.state = {
30             email: '',
31             login: '',
32             password: '',
33             repeatPassword: '',
34 
35             listErrors: [],
36 
37             isLoading: false
38         };
39 
40         this.handleInputChange = this.handleInputChange.bind(this);
41         this.handleFormSubmit = this.handleFormSubmit.bind(this);
42         this.hideErrorAlert = this.hideErrorAlert.bind(this);
43         this.toggleLoadingState = this.toggleLoadingState.bind(this);
44     }
45 
46     handleInputChange(event) {
47         const { name, value } = event.target;
48 
49         this.setState({ [name]: value });
50     }
51 
52     toggleLoadingState() {
53         this.setState(prev => {
54             return {
55                 isLoading: !prev.isLoading
56             }
57         });
58     }
59 
60     _generateFormData() {
61         const formData = new FormData();
62 
63         formData.append('login', this.state.login);
64         formData.append('password', this.state.password);
65         formData.append('email', this.state.email);
66         formData.append('repeatPassword', this.state.repeatPassword);
67 
68         return formData;
69     }
70 
71     async handleFormSubmit(event) {
72         event.preventDefault();
73 
74         const { history } = this.props;
75         const formData = this._generateFormData();
76 
77         this.toggleLoadingState();
78 
79         try {
80             await authService.signUp(formData);
81 
82             history.push('/login');
83         } catch ({ errors }) {
84             this.setState({ listErrors: errors });
85         }
86 
87         this.toggleLoadingState();
88     }
89 
90     hideErrorAlert() {
91         this.setState({ listErrors: [] });
92     }
93 
94     render() {
95         const { listErrors, isLoading } = this.state;
96 
97         return (
98             <Container className="p-3">
99                 <Col lg={{ offset: 3, span: 6 }}>
100                     <h2 className="main-signup-form__title">Sign Up form</h2>
```

```
                    <Form>
                        <ErrorFormAlert listErrors={listErrors}
                                        show={listErrors.length !== 0}
                                        onHide={this.hideErrorAlert} />
                        <Form.Group controlId="main-signup-form__email">
                            <Form.Control type="email"
                                          placeholder="Enter email"
                                          name="email"
                                          maxLength={MAX_EMAIL_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Form.Group controlId="main-signup-form__login">
                            <Form.Control type="text"
                                          placeholder="Enter login"
                                          name="login"
                                          minLength={MIN_LOGIN_LENGTH}
                                          maxLength={MAX_LOGIN_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Form.Group controlId="main-signup-form__password">
                            <Form.Control type="password"
                                          placeholder="Enter password"
                                          name="password"
                                          minLength={MIN_PASSWORD_LENGTH}
                                          maxLength={MAX_PASSWORD_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Form.Group controlId="main-signup-form__repeat-password">
                            <Form.Control type="password"
                                          placeholder="Enter password"
                                          name="repeatPassword"
                                          minLength={MIN_PASSWORD_LENGTH}
                                          maxLength={MAX_PASSWORD_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Button variant="primary"
                                type="submit"
                                block
                                disabled={isLoading}
                                onClick={this.handleFormSubmit}>
                            { isLoading ? 'Loading...' : 'Submit' }
                        </Button>
                    </Form>
                </Col>
            </Container>
        );
    }
}

SignUp.propTypes = {
    history: ReactRouterPropTypes.history
};

export default withRouter(SignUp);
```

db-course-project-app/src/client/apps/main/pages/TestEditor/style.scss

```
.test-editor {
  &__submit-section {
    text-align: center;
  }

  &__submit-section-row {
    margin: 10px 0;
  }

  &__textarea {
    resize: none;
  }
}
```

db-course-project-app/src/client/apps/main/pages/TestEditor/index.js

```
import { TestEditor } from "./TestEditor";

export default TestEditor;
```

...course-project-app/src/client/apps/main/pages/TestEditor/TestEditor.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import ReactRouterPropTypes from "react-router-prop-types";
import { withRouter } from "react-router-dom";
import { connect } from "react-redux";
import * as testEditorActions from "../../../../actions/testEditor";

import Container from "react-bootstrap/Container";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import Form from "react-bootstrap/Form";
import InputGroup from "react-bootstrap/InputGroup";
import FormControl from "react-bootstrap/FormControl";
import Button from "react-bootstrap/Button";
import TestEditorTagList from "../../../../containers/TestEditorTagList";
import TestEditorQuestionList from "../../../../containers/TestEditorQuestionList";
import ErrorFormAlert from "../../components/ErrorFormAlert";
import {ANSWER_TYPE} from "../../components/AnswerEditList/config";
import * as editTest from "../../../../services/editTest";

import "./style.scss";

class TestEditor extends React.Component {
    constructor(props) {
        super(props);

        const { location } = props;
        let query = new URLSearchParams(location.search);

        const testId = parseInt(query.get("id"));

        const isEditing = !!testId;

        this.state = {
            isEditing,
            testId: testId ? testId : -1,
            tagValue: '',
            listErrors: [],
        };

        // TODO: add loading state

        this.handleTitleChange = this.handleTitleChange.bind(this);
        this.handleDescriptionChange = this.handleDescriptionChange.bind(this);
        this.handleAppendTag = this.handleAppendTag.bind(this);
        this.handleTagInputChange = this.handleTagInputChange.bind(this);
        this.handleAppendQuestion = this.handleAppendQuestion.bind(this);
        this.hideErrorAlert = this.hideErrorAlert.bind(this);
        this.handleFormSubmit = this.handleFormSubmit.bind(this);
    }

    async componentDidMount() {
        const { dispatch } = this.props;
        const { isEditing, testId } = this.state;

        if (isEditing) {
            let response = null;
            try {
                response = await editTest.getTestForEdit(testId);

                dispatch(testEditorActions.update(response));
            } catch (err) {
```

```
63                 console.error(err);
64             }
65         } else {
66             dispatch(testEditorActions.reset());
67         }
68     }
69
70     handleTitleChange({ target: { value } }) {
71         const { dispatch } = this.props;
72
73         dispatch(testEditorActions.updateTitle(value));
74     }
75
76     handleDescriptionChange({ target: { value } }) {
77         const { dispatch } = this.props;
78
79         dispatch(testEditorActions.updateDescription(value));
80     }
81
82     handleTagInputChange({ target: { value } }) {
83         this.setState({ tagValue: value });
84     }
85
86     handleAppendTag() {
87         const { tagValue } = this.state;
88         const { dispatch } = this.props;
89
90         dispatch(testEditorActions.appendTag(tagValue));
91     }
92
93     handleAppendQuestion(event) {
94         event.preventDefault();
95
96         const { dispatch } = this.props;
97
98         dispatch(testEditorActions.appendQuestion());
99     }
100
101    hideErrorAlert() {
102        this.setState({
103            listErrors: []
104        });
105    }
106
107    async handleFormSubmit(event) {
108        event.preventDefault();
109
110        const { history, testEditor } = this.props;
111        const { isEditing, testId } = this.state;
112
113        try {
114            if (!isEditing) {
115                await editTest.create(testEditor);
116            } else {
117                await editTest.update(testEditor, testId);
118            }
119
120            history.push('/profile/tests');
121        } catch ({ errors }) {
122            this.setState({
123                listErrors: errors
124            });
125        }
126    }
127
128    render() {
129        const { listErrors } = this.state;
130        const { testEditor: { info } } = this.props;
131
132        return (
133            <Container className="p-3">
134                <Row>
135                    <Col lg={{offset: 2, span: 8}}>
136                        <ErrorFormAlert listErrors={listErrors}
137                                        show={listErrors.length !== 0}
138                                        onHide={this.hideErrorAlert} />
```

```
                    <Form>
                        <Row>
                            <Col lg={4}>
                                <Form.Label>Title:</Form.Label>
                            </Col>
                            <Col lg={8}>
                                <Form.Group controlId="">
                                    <Form.Control required
                                                  value={info.title}
                                                  onChange={this.handleTitleChange} />
                                </Form.Group>
                            </Col>
                        </Row>
                        <Row>
                            <Col lg={4}>
                                <Form.Label>Description:</Form.Label>
                            </Col>
                            <Col lg={8}>
                                <Form.Group controlId="">
                                    <Form.Control className="test-editor__textarea"
                                                  as="textarea"
                                                  rows={3}
                                                  value={info.description}
                                                  required
                                                  onChange={this.handleDescriptionChange} />
                                </Form.Group>
                            </Col>
                        </Row>
                        <Row>
                            <Col lg={4}>
                                <Form.Label>Tags:</Form.Label>
                            </Col>
                            <Col lg={8}>
                                <Form.Group controlId="">
                                    <InputGroup className="mb-3">
                                        <FormControl
                                            aria-label="Recipient's username"
                                            aria-describedby="basic-addon2"
                                            required
                                            onChange={this.handleTagInputChange}
                                        />
                                        <InputGroup.Append>
                                            <Button variant="primary"
                                                    onClick={this.handleAppendTag}>
                                                Add
                                            </Button>
                                        </InputGroup.Append>
                                    </InputGroup>
                                </Form.Group>
                                <TestEditorTagList />
                            </Col>
                        </Row>

                        <hr/>

                        <TestEditorQuestionList />

                        <Row>
                            <Col lg={12}>
                                <div className="test-editor__submit-section">
                                    <div className="test-editor__submit-section-row">
                                        <Button className="test-editor__submit-btn"
                                                type="primary"
                                                size="lg"
                                                onClick={this.handleAppendQuestion}>
                                            Add question
                                        </Button>
                                    </div>
                                    <div className="test-editor__submit-section-row">
                                        <Button className="test-editor__submit-btn"
                                                type="primary"
                                                size="lg" onClick={this.handleFormSubmit}>
                                            Publish
                                        </Button>
                                    </div>
                                </div>
```

```
                                </Col>
                            </Row>
                        </Form>
                    </Col>
                </Row>
            </Container>
        );
    }
}

TestEditor.propTypes = {
    dispatch: PropTypes.func,
    history: ReactRouterPropTypes.history,
    location: ReactRouterPropTypes.location,
    testEditor: PropTypes.exact({
        info: PropTypes.exact({
            title: PropTypes.string,
            description: PropTypes.string,
            tags: PropTypes.arrayOf(PropTypes.string)
        }),
        questions: PropTypes.arrayOf(
            PropTypes.exact({
                title: PropTypes.string,
                typeAnswer: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]),
                answers: PropTypes.arrayOf(
                    PropTypes.exact({
                        content: PropTypes.string,
                        isRight: PropTypes.bool
                    })
                )
            })
        )
    }).isRequired
};

function mapStateToProps(state) {
    const { testEditor } = state;

    return { testEditor };
}

const connectedTestEditor = connect(mapStateToProps)(withRouter(TestEditor));

export { connectedTestEditor as TestEditor };
```

db-course-project-app/src/client/apps/main/pages/Profile/style.scss

```
.profile {
  &__username {
    font-weight: bold;
    text-align: center;
  }

  &__avatar {
    border: 6px solid #f0f8ff;
    border-radius: 100%;
    display: block;
    margin: 0 auto;
    width: 70%;
  }
}
```

db-course-project-app/src/client/apps/main/pages/Profile/Profile.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import {NOT_FOUND} from "http-status-codes";

import { Switch, Route } from 'react-router-dom';
import { LinkContainer } from "react-router-bootstrap";
import Container from "react-bootstrap/Container";
```

```
import Col from "react-bootstrap/Col";
import Row from "react-bootstrap/Row";
import Nav from "react-bootstrap/Nav";
import ProfileSettings from "../ProfileSettings";
import ProfileAttempts from "../ProfileAttempts";
import HttpErrorInfo from "../../components/HttpErrorInfo";
import ProfileTests from "../ProfileTests";

import "./style.scss";

const Profile = ({ user }) => {
    return (
        <Container className="p-3">
            <Row>
                <Col lg={3}>
                    <img className="profile__avatar"
                         src="http://ssl.gstatic.com/accounts/ui/avatar_2x.png" alt="avatar"/>
                    <p className="profile__username">{ user.login }</p>
                </Col>
                <Col lg={9}>
                    <Nav variant="tabs" defaultActiveKey="tests">
                        <Nav.Item>
                            <LinkContainer to="/profile/tests">
                                <Nav.Link eventKey="tests">My tests</Nav.Link>
                            </LinkContainer>
                        </Nav.Item>
                        <Nav.Item>
                            <LinkContainer to="/profile/attempts">
                                <Nav.Link eventKey="attempts">My attempts</Nav.Link>
                            </LinkContainer>
                        </Nav.Item>
                        <Nav.Item>
                            <LinkContainer to="/profile/settings">
                                <Nav.Link eventKey="settings">Settings</Nav.Link>
                            </LinkContainer>
                        </Nav.Item>
                    </Nav>

                    <Switch>
                        <Route exact path="/profile" render={() => <ProfileTests />} />
                        <Route path="/profile/tests" render={() => <ProfileTests />} />
                        <Route path="/profile/attempts" component={ProfileAttempts} />
                        <Route path="/profile/settings" component={ProfileSettings} />
                        <Route component={() => <HttpErrorInfo status={NOT_FOUND} />} />
                    </Switch>
                </Col>
            </Row>
        </Container>
    );
}

Profile.propTypes = {
    user: PropTypes.shape({
        login: PropTypes.string
    })
};

function mapStateToProps(state) {
    const { user } = state.auth;

    return { user };
}

const connectedProfile = connect(mapStateToProps)(Profile);

export { connectedProfile as Profile };
```

db-course-project-app/src/client/apps/main/pages/Profile/index.js

```
import { Profile } from "./Profile";

export default Profile;
```

...ourse-project-app/src/client/apps/main/pages/ProfileAttempts/style.scss

```

```

...ject-app/src/client/apps/main/pages/ProfileAttempts/ProfileAttempts.jsx

```jsx
import * as React from "react";

import "./style.scss";

const ProfileAttempts = () => {
    return (
        <h2>Attempts</h2>
    );
};

export default ProfileAttempts;
```

...-course-project-app/src/client/apps/main/pages/ProfileAttempts/index.js

```jsx
import ProfileAttempts from "./ProfileAttempts";

export default ProfileAttempts;
```

db-course-project-app/src/client/apps/main/pages/Home/style.scss

```

```

db-course-project-app/src/client/apps/main/pages/Home/index.js

```jsx
import Home from "./Home";

export default Home;
```

db-course-project-app/src/client/apps/main/pages/Home/Home.jsx

```jsx
import * as React from "react";

import Jumbotron from "react-bootstrap/Jumbotron";
import Container from "react-bootstrap/Container";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";

import "./style.scss";

const Home = () => {
    return (
        <Container className="p-3">
            <Jumbotron>
                <h1>Hello!</h1>
                <p>
                    This is a simple hero unit, a simple jumbotron-style component for calling
                    extra attention to featured content or information.
                </p>
            </Jumbotron>
            <Row>
                <Col lg={4}>
                    <h2>Create</h2>
                    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab blanditiis delectus
                    ↪  dolorem eius expedita nemo neque pariatur perferendis quasi! Accusantium ad atque
                    ↪  corporis deleniti eius, eligendi qui rem sunt vitae!</p>
                </Col>
                <Col lg={4}>
                    <h2>Share</h2>
```

```
                    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab blanditiis delectus
                    ↪  dolorem eius expedita nemo neque pariatur perferendis quasi! Accusantium ad atque
                    ↪  corporis deleniti eius, eligendi qui rem sunt vitae!</p>
                </Col>
                <Col lg={4}>
                    <h2>Statistic</h2>
                    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab blanditiis delectus
                    ↪  dolorem eius expedita nemo neque pariatur perferendis quasi! Accusantium ad atque
                    ↪  corporis deleniti eius, eligendi qui rem sunt vitae!</p>
                </Col>
            </Row>
        </Container>
    );
}

export default Home;
```

db-course-project-app/src/client/apps/main/pages/Login/style.scss

```
.main-login-form {
  &__forgot-password {
    display: block;
    margin: 10px 0;
    text-align: center;
  }

  &__title {
    text-align: center;
  }

  &__label {
    font-weight: bold;
  }
}
```

db-course-project-app/src/client/apps/main/pages/Login/Login.jsx

```
import * as React from "react";
import ReactRouterPropTypes from "react-router-prop-types";
import PropTypes from "prop-types";
import { connect } from "react-redux";

import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";
import Container from "react-bootstrap/Container";
import Col from "react-bootstrap/Col";
import LinkContainer from "react-router-bootstrap/lib/LinkContainer";
import ErrorFormAlert from "../../components/ErrorFormAlert";

import authService from "../../../../services/auth";
import * as authActions from "../../../../actions/auth";
import userConstraints from "../../../../../models/User/constraints";

import "./style.scss";

const {
    MIN_PASSWORD_LENGTH,
    MAX_PASSWORD_LENGTH,
    MIN_LOGIN_LENGTH,
    MAX_LOGIN_LENGTH
} = userConstraints;

class Login extends React.Component {
    constructor(props) {
        super(props);

        this.state = {
            login: '',
            password: '',
            readMeChecked: false,

            listErrors: [],
```

```
36
37              isLoading: false
38          }
39
40          this.handleInputChange = this.handleInputChange.bind(this);
41          this.handleCheckboxChange = this.handleCheckboxChange.bind(this);
42          this.handleFormSubmit = this.handleFormSubmit.bind(this);
43          this.hideErrorAlert = this.hideErrorAlert.bind(this);
44      }
45
46      handleInputChange(event) {
47          const { name, value } = event.target;
48
49          this.setState({ [name]: value });
50      }
51
52      handleCheckboxChange(event) {
53          const { name, checked } = event.target;
54
55          this.setState({ [name]: checked });
56      }
57
58      _generateFormData() {
59          const formData = new FormData();
60
61          formData.append('login', this.state.login);
62          formData.append('password', this.state.password);
63
64          return formData;
65      }
66
67      async handleFormSubmit(event) {
68          event.preventDefault();
69
70          const { history, dispatch } = this.props;
71          const formData = this._generateFormData();
72
73          this.toggleLoadingState();
74
75          try {
76              const { token, user } = await authService.signIn(formData);
77
78              localStorage.setItem('TOKEN', token);
79
80              dispatch(authActions.success(user));
81
82              history.push('/');
83          } catch ({ errors }) {
84              this.setState({ listErrors: errors });
85          }
86
87          this.toggleLoadingState();
88      }
89
90      hideErrorAlert() {
91          this.setState({
92              listErrors: []
93          });
94      }
95
96      toggleLoadingState() {
97          this.setState(prev => {
98              return {
99                  isLoading: !prev.isLoading
100             }
101         });
102     }
103
104     render() {
105         const { listErrors, isLoading } = this.state;
106
107         return (
108             <Container className="p-3">
109                 <Col lg={{offset: 3, span: 6}}>
110                     <h2 className="main-login-form__title">Login form</h2>
111                     <Form>
```

```
                        <ErrorFormAlert listErrors={listErrors}
                                        show={listErrors.length !== 0}
                                        onHide={this.hideErrorAlert} />
                        <Form.Group controlId="main-login-form__login">
                            <Form.Control type="text"
                                          placeholder="Enter login"
                                          name="login"
                                          minLength={MIN_LOGIN_LENGTH}
                                          maxLength={MAX_LOGIN_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Form.Group controlId="main-login-form__password">
                            <Form.Control type="password"
                                          placeholder="Enter password"
                                          name="password"
                                          minLength={MIN_PASSWORD_LENGTH}
                                          maxLength={MAX_PASSWORD_LENGTH}
                                          required
                                          onChange={this.handleInputChange}/>
                        </Form.Group>
                        <Form.Group controlId="main-login-form__checkbox">
                            <Form.Check type="checkbox"
                                        label="Remember me"
                                        name="readMeChecked"
                                        onChange={this.handleCheckboxChange} />
                        </Form.Group>
                        <Button variant="primary"
                                type="submit"
                                block
                                disabled={isLoading}
                                onClick={this.handleFormSubmit}>
                            { isLoading ? 'Loading...' : 'Submit' }
                        </Button>
                        <LinkContainer to="#">
                            <a className="main-login-form__forgot-password">Forgot password?</a>
                        </LinkContainer>
                    </Form>
                </Col>
            </Container>
        );
    }
}

Login.propTypes = {
    location: ReactRouterPropTypes.location,
    history: ReactRouterPropTypes.history,
    dispatch: PropTypes.func
};

const connectedLogin = connect()(Login);

export { connectedLogin as Login };
```

db-course-project-app/src/client/apps/main/pages/Login/index.js

```
import { Login } from "./Login";

export default Login;
```

...b-course-project-app/src/client/apps/main/components/TagList/style.scss

```
.tag-list {
  display: inline;
  margin: 10px 0;
  padding: 0;
}
```

db-course-project-app/src/client/apps/main/components/TagList/index.js

```
import TagList from "./TagList";

export default TagList;
```

...-course-project-app/src/client/apps/main/components/TagList/TagList.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import Tag from "../Tag";

import "./style.scss";

const TagList = ({ tags, deleteTag, canDelete = true }) => {
    return (
        <ul className="tag-list">
            {
                tags.map((content, i) => {
                    return (
                        <Tag content={content}
                             key={i}
                             canDelete={canDelete}
                             onDelete={() => deleteTag(i)} />
                    );
                })
            }
        </ul>
    );
};

TagList.propTypes = {
    tags: PropTypes.arrayOf(PropTypes.string).isRequired,
    canDelete: PropTypes.bool,
    deleteTag: PropTypes.func,
};

export default TagList;
```

db-course-project-app/src/client/apps/main/components/Tag/style.scss

```
$tag-color: #0366d6;
$tag-bg-color: #f1f8ff;

.tag {
  background-color: $tag-bg-color;
  border-radius: 10%;
  color: $tag-color;
  display: inline-block;
  list-style: none;
  margin: 5px;
  padding: 6px 8px;
  user-select: none;

  &__content {
    margin-right: 5px;
  }

  &__delete-btn {
    background: none;
    border: 0;
    border-radius: 100%;
    color: $tag-color;
    font-size: 0.85rem;
    padding: 5px 7px;

  }

  &__delete-btn:hover {
    background-color: #def;
  }

  &__delete-btn:focus {
```

```
    outline: none;
  }
}
```

db-course-project-app/src/client/apps/main/components/Tag/index.js

```
import Tag from "./Tag";

export default Tag;
```

db-course-project-app/src/client/apps/main/components/Tag/Tag.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faTimes } from '@fortawesome/free-solid-svg-icons';

import "./style.scss";

const Tag = ({ content, onDelete, canDelete }) => {
    return (
        <li className="tag">
            <span className="tag__content">{content}</span>
            {
                canDelete && (
                    <button className="tag__delete-btn"
                            onClick={event => {
                                event.preventDefault();
                                onDelete();
                            }}>
                        <FontAwesomeIcon icon={faTimes} />
                    </button>
                )
            }
        </li>
    );
};

Tag.propTypes = {
    content: PropTypes.string.isRequired,
    canDelete: PropTypes.bool.isRequired,
    onDelete: PropTypes.func,
};

export default Tag;
```

db-course-project-app/src/client/apps/main/components/Header/style.scss

```
//.header {
//  &__user-dropdown-menu {
//    float: right;
//  }
//}
```

db-course-project-app/src/client/apps/main/components/Header/Header.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import { withRouter } from "react-router-dom";
import ReactRouterPropTypes from "react-router-prop-types";
import { connect } from "react-redux";
import * as authActions from "../../../../actions/auth";

import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
import NavDropdown from "react-bootstrap/NavDropdown";
import Modal from "react-bootstrap/Modal";
```

```
12 import Button from "react-bootstrap/Button";
13
14 import { LinkContainer } from "react-router-bootstrap";
15
16 import "./style.scss";
17
18 const Header = ({ isLoggedIn, user, dispatch, history }) => {
19     const [modalShow, setModalShow] = React.useState(false);
20
21     const showModal = () => setModalShow(true);
22
23     const hideModal = () => setModalShow(false);
24
25     const onLogOut = () => {
26         dispatch(authActions.logOut());
27
28         setModalShow(false);
29
30         history.push("/");
31     };
32
33     return (
34         <Navbar bg="dark" variant="dark">
35             <LinkContainer to="/">
36                 <Navbar.Brand>PassQuiz</Navbar.Brand>
37             </LinkContainer>
38             <Nav className="mr-auto">
39                 { !isLoggedIn ? (
40                     <>
41                         <LinkContainer to="/signup">
42                             <Nav.Link>Sign Up</Nav.Link>
43                         </LinkContainer>
44                         <LinkContainer to="/login">
45                             <Nav.Link>Login</Nav.Link>
46                         </LinkContainer>
47                     </>
48                 ) : (
49                     <NavDropdown title={ user.login } id="user-nav-dropdown">
50                         <LinkContainer to="/profile">
51                             <NavDropdown.Item>My profile</NavDropdown.Item>
52                         </LinkContainer>
53                         <NavDropdown.Divider />
54                         <LinkContainer to="#">
55                             <NavDropdown.Item onClick={showModal}>Logout</NavDropdown.Item>
56                         </LinkContainer>
57                     </NavDropdown>
58                 ) }
59             </Nav>
60             {
61                 isLoggedIn && (
62                     <Nav>
63                         <LinkContainer to="/test/edit">
64                             <Button type="primary">Create test</Button>
65                         </LinkContainer>
66                     </Nav>
67                 )
68             }
69
70             <Modal
71                 size="md"
72                 aria-labelledby="contained-modal-title-vcenter"
73                 centered
74                 onHide={hideModal}
75                 show={modalShow}
76             >
77                 <Modal.Header closeButton>
78                     <Modal.Title id="contained-modal-title-vcenter">Log out</Modal.Title>
79                 </Modal.Header>
80                 <Modal.Body>
81                     <p>Are you sure you want to log-off?</p>
82                 </Modal.Body>
83                 <Modal.Footer>
84                     <Button onClick={onLogOut}>Yes</Button>
85                 </Modal.Footer>
86             </Modal>
87         </Navbar>
```

```
    );
}

Header.propTypes = {
    history: ReactRouterPropTypes.history,
    dispatch: PropTypes.func,
    isLoggedIn: PropTypes.bool,
    user: PropTypes.shape({
        login: PropTypes.string
    })
};

function mapStateToProps(state) {
    const { isLoggedIn, user } = state.auth;

    return { isLoggedIn, user };
}

const headerWithRouter = withRouter(Header);
const connectedHeaderWithRouter = connect(mapStateToProps)(headerWithRouter);

export { connectedHeaderWithRouter as Header };
```

db-course-project-app/src/client/apps/main/components/Header/index.js

```
import { Header } from "./Header";

export default Header;
```

...-course-project-app/src/client/apps/main/components/TestCard/style.scss

```
.test-card {
  margin: 0 0 15px;
  width: 100%;

  &__title {
    font-size: 1.1rem;
  }

  &__label-info {
    font-weight: bold;
    padding-right: 5px;
  }

  &__control {
    display: flex;
    justify-content: space-between;
  }
}
```

db-course-project-app/src/client/apps/main/components/TestCard/index.js

```
import TestCard from "./TestCard";

export default TestCard;
```

...ourse-project-app/src/client/apps/main/components/TestCard/TestCard.jsx

```
import * as React from "react";
import PropType from "prop-types";
import { useHistory } from "react-router-dom";
import Card from "react-bootstrap/Card";
import Button from "react-bootstrap/Button";
import Dropdown from "react-bootstrap/Dropdown";
import TagList from "../TagList";

import "./style.scss";
```

```

const TestCard = ({ title, description, author, tags, onDeleteTestCard, testId }) => {
    const history = useHistory();

    return (
        <Card className="test-card">
            <Card.Body>
                <Card.Title className="test-card__title">{title}</Card.Title>

                <div className="test-card__author">
                    <span className="test-card__label-info">Author:</span> {author}
                </div>

                <div className="test-card__tags">
                    <span className="test-card__label-info">Tags:</span>
                    {
                        tags.length ? (
                            <TagList tags={tags} canDelete={false} />
                        ) : (
                            "None"
                        )
                    }
                </div>

                <Card.Text className="test-card__description">{description}</Card.Text>

                <div className="test-card__control">
                    <Button className="test-card__pass-btn"
                            variant="primary">Pass test</Button>

                    <Dropdown className="test-card__dropdown-menu">
                        <Dropdown.Toggle variant="primary">Menu</Dropdown.Toggle>
                        <Dropdown.Menu>
                            <Dropdown.Item as="button" onClick={() => {
                                history.push(`/test/edit?id=${testId}`);
                            }}>Edit</Dropdown.Item>
                            <Dropdown.Item as="button"
                                           onClick={() => {
                                onDeleteTestCard(testId);
                            }}>
                                Delete
                            </Dropdown.Item>
                            <Dropdown.Item as="button">Share</Dropdown.Item>
                        </Dropdown.Menu>
                    </Dropdown>
                </div>
            </Card.Body>
        </Card>
    );
};

TestCard.propTypes = {
    title: PropType.string.isRequired,
    description: PropType.string.isRequired,
    author: PropType.string.isRequired,
    tags: PropType.arrayOf(PropType.string).isRequired,
    testId: PropType.number.isRequired,
    onDeleteTestCard: PropType.func
};

export default TestCard;
```

...e-project-app/src/client/apps/main/components/AnswerEditList/style.scss

```
.answer-edit-list {
  border: 0;
}
```

...rse-project-app/src/client/apps/main/components/AnswerEditList/index.js

```
import { AnswerEditList} from "./AnswerEditList";

```

```
export { AnswerEditList };
```

...t-app/src/client/apps/main/components/AnswerEditList/AnswerEditList.jsx

```
import * as React from "react";
import { connect } from "react-redux";
import PropTypes from "prop-types";
import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";
import AnswerEditItem from "../AnswerEditItem";
import {ANSWER_TYPE} from "./config";
import * as testEditorActions from "../../../../actions/testEditor";

class AnswerEditList extends React.Component {
    constructor(props) {
        super(props);
    }

    render() {
        const {
            name,
            type,
            answers,
            onAppendAnswer,
            deleteAnswer,
            updateAnswerText,
            updateAnswers
        } = this.props;

        return (
            <>
                <p>Choose right answer:</p>
                <Form.Group controlId="">
                    {
                        answers.map((el, i) => {
                            const { content, isRight } = el;

                            return (
                                <AnswerEditItem key={i}
                                                name={name}
                                                type={type}
                                                isRight={isRight}
                                                content={content}
                                                onChangeAnswer={(isRight, typeAnswer) =>
                                                ↪  updateAnswers(name, i, isRight, typeAnswer)}
                                                onDeleteAnswer={() => deleteAnswer(name, i)}
                                                onChangeAnswerText={value => updateAnswerText(name, i,
                                                ↪  value)} />
                            );
                        })
                    }
                </Form.Group>
                <Button variant="primary" onClick={onAppendAnswer}>
                    Add answer
                </Button>
            </>
        );
    }
}

AnswerEditList.propTypes = {
    name: PropTypes.oneOfType([ PropTypes.string, PropTypes.number ]).isRequired,
    type: PropTypes.oneOf([ ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE ]).isRequired,
    answers: PropTypes.arrayOf(
        PropTypes.exact({
            content: PropTypes.string,
            isRight: PropTypes.bool
        })
    ).isRequired,
    onAppendAnswer: PropTypes.func.isRequired,
    deleteAnswer: PropTypes.func.isRequired,
    updateAnswerText: PropTypes.func.isRequired,
    updateAnswers: PropTypes.func
};
```

```
69
70 function mapStateToProps() {
71     return {};
72 }
73
74 function mapDispatchToProps(dispatch) {
75     return {
76         deleteAnswer: (questionId, answerId) => dispatch(testEditorActions.deleteAnswer(questionId,
        ↪  answerId)),
77         updateAnswerText: (questionId, answerId, value) =>
        ↪  dispatch(testEditorActions.updateAnswerText(questionId, answerId, value)),
78         updateAnswers: (questionId, answerId, isRight, typeAnswer) =>
        ↪  dispatch(testEditorActions.updateAnswers(questionId, answerId, isRight, typeAnswer))
79     };
80 }
81
82 const connectedAnswerEditList = connect(mapStateToProps, mapDispatchToProps)(AnswerEditList);
83
84 export { connectedAnswerEditList as AnswerEditList };
```

...se-project-app/src/client/apps/main/components/ListTestCards/style.scss

```
1 .list-tests {
2   &__col {
3     padding: 0;
4   }
5 }
```

...urse-project-app/src/client/apps/main/components/ListTestCards/index.js

```
1 import ListTestCards from "./ListTestCards";
2
3 export default ListTestCards;
```

...ect-app/src/client/apps/main/components/ListTestCards/ListTestCards.jsx

```
1  import * as React from "react";
2  import PropTypes from "prop-types";
3
4  import Row from "react-bootstrap/Row";
5  import TestCard from "../TestCard";
6
7  import "./style.scss";
8
9  const ListTestCards = ({ tests, onDeleteTestCard }) => {
10     return (
11         <Row>
12             {
13                 tests.map(test => {
14                     const {
15                         title,
16                         description,
17                         author,
18                         tags,
19                         testId
20                     } = test;
21
22                     return (
23                         <TestCard title={title}
24                                   description={description}
25                                   author={author}
26                                   tags={tags}
27                                   key={testId}
28                                   testId={testId}
29                                   onDeleteTestCard={onDeleteTestCard} />
30                     );
31                 })
32             }
33         </Row>
```

```
    );
};

ListTestCards.propTypes = {
    tests: PropTypes.array,
    onDeleteTestCard: PropTypes.func
};

export default ListTestCards;
```

...p/src/client/apps/main/components/QuestionEditItem/QuestionEditItem.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import Form from "react-bootstrap/Form";
import ToggleButtonGroup from "react-bootstrap/ToggleButtonGroup";
import ToggleButton from "react-bootstrap/ToggleButton";
import { ANSWER_TYPE } from "../AnswerEditList/config";
import { AnswerEditList } from "../AnswerEditList";
import Button from "react-bootstrap/Button";

class QuestionEditItem extends React.Component {
    constructor(props) {
        super(props);

        this.handleToggleChange = this.handleToggleChange.bind(this);
    }

    handleToggleChange({ target: { value } }) {
        const { onChangeAnswerType } = this.props;

        onChangeAnswerType(value);
    }

    render() {
        const {
            name,
            typeAnswer,
            answers,
            onDelete,
            onUpdateTitle,
            onAnswerListUpdate,
            title
        } = this.props;

        return (
            <>
                <Row>
                    <Col lg={4}>
                        <Form.Label>Question:</Form.Label>
                    </Col>
                    <Col lg={8}>
                        <Form.Group controlId="">
                            <Form.Control className="test-editor__textarea"
                                          as="textarea"
                                          rows={3}
                                          value={title}
                                          onChange={event => {
                                              const titleVal = event.target.value;

                                              onUpdateTitle(titleVal);
                                          }} />
                        </Form.Group>
                    </Col>
                </Row>
                <Row>
                    <Col lg={4}>
                        <Form.Label>Question type:</Form.Label>
                    </Col>
                    <Col lg={8}>
                        <Form.Group>
                            <ToggleButtonGroup type="radio"
```

```
                                        name={`${name}_toggle_answer_type`}
                                        defaultValue={ANSWER_TYPE.ONE}>
                                <ToggleButton value={ANSWER_TYPE.ONE}
                                        onChange={this.handleToggleChange}>One answer</ToggleButton>
                                <ToggleButton value={ANSWER_TYPE.MULTIPLE}
                                        onChange={this.handleToggleChange}>Multiple
                                        ↪  answers</ToggleButton>
                            </ToggleButtonGroup>
                        </Form.Group>
                    </Col>
                </Row>
                <Row>
                    <Col lg={4}>
                        <Form.Label>Answers:</Form.Label>
                    </Col>
                    <Col lg={8}>
                        <AnswerEditList name={name}
                                        type={typeAnswer}
                                        answers={answers}
                                        onAppendAnswer={() => onAnswerListUpdate(name)} />
                    </Col>
                </Row>
                <Row>
                    <Col lg={12}>
                        <Button className="float-right"
                                variant="danger"
                                onClick={onDelete}>
                            Delete question
                        </Button>
                    </Col>
                </Row>

                <hr/>
            </>
        );
    }
}

QuestionEditItem.propTypes = {
    name: PropTypes.oneOfType([ PropTypes.string, PropTypes.number ]).isRequired,
    typeAnswer: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]).isRequired,
    title: PropTypes.string,
    answers: PropTypes.arrayOf(
        PropTypes.exact({
            content: PropTypes.string,
            isRight: PropTypes.bool
        })
    ).isRequired,
    onDelete: PropTypes.func,
    onUpdateTitle: PropTypes.func,
    onChangeAnswerType: PropTypes.func,
    onAnswerListUpdate: PropTypes.func
};

export default QuestionEditItem;
```

...e-project-app/src/client/apps/main/components/QuestionEditItem/index.js

```
import QuestionEditItem from "./QuestionEditItem";

export default QuestionEditItem;
```

...e-project-app/src/client/apps/main/components/AnswerEditItem/style.scss

```
.answer-edit-item {
  margin: 10px 0;
}
```

...t-app/src/client/apps/main/components/AnswerEditItem/AnswerEditItem.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import InputGroup from "react-bootstrap/InputGroup";
import FormControl from "react-bootstrap/FormControl";
import Button from "react-bootstrap/Button";
import { ANSWER_TYPE } from "../AnswerEditList/config";
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faTrash } from '@fortawesome/free-solid-svg-icons';

import "./style.scss";

class AnswerEditItem extends React.Component {
    constructor(props) {
        super(props);

        const { content } = props;

        this.state = {
            answerValue: content
        }

        this.handleChangeAnswer = this.handleChangeAnswer.bind(this);
    }

    handleChangeAnswer(event) {
        const { onChangeAnswer } = this.props;
        const { checked, type } = event.target;

        onChangeAnswer(checked, type);
    }

    render() {
        const {
            name,
            type,
            content,
            isRight,
            onDeleteAnswer,
            onChangeAnswerText
        } = this.props;

        return (
            <InputGroup className="answer-edit-item">
                <InputGroup.Prepend>
                    {
                        (() => {
                            switch (type) {
                                case ANSWER_TYPE.ONE:
                                    return (
                                        <InputGroup.Radio name={name}
                                                          checked={isRight}
                                                          onChange={this.handleChangeAnswer}
                                                          aria-label="Radio button for following text
                                                          ↪  input" />
                                    );
                                case ANSWER_TYPE.MULTIPLE:
                                    return (
                                        <InputGroup.Checkbox name={name}
                                                             checked={isRight}
                                                             onChange={this.handleChangeAnswer}
                                                             aria-label="Radio button for following text
                                                             ↪  input" />
                                    );
                            }
                        })()
                    }
                </InputGroup.Prepend>
                <FormControl aria-label="Text input with radio button"
                             value={content}
                             onChange={(event => {
                                 const { value } = event.target;

                                 onChangeAnswerText(value);
                             })} />
                <InputGroup.Append>
```

```
                    <Button variant="danger"
                            onClick={onDeleteAnswer}>
                        <FontAwesomeIcon icon={faTrash} />
                    </Button>
                </InputGroup.Append>
            </InputGroup>
        );
    }
}

AnswerEditItem.propTypes = {
    name: PropTypes.oneOfType([ PropTypes.string, PropTypes.number ]).isRequired,
    type: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]).isRequired,
    content: PropTypes.string.isRequired,
    isRight: PropTypes.bool.isRequired,
    onDeleteAnswer: PropTypes.func.isRequired,
    onChangeAnswerText: PropTypes.func.isRequired,
    onChangeAnswer: PropTypes.func.isRequired
};

export default AnswerEditItem;
```

...rse-project-app/src/client/apps/main/components/AnswerEditItem/index.js

```
import AnswerEditItem from "./AnswerEditItem";

export default AnswerEditItem;
```

db-course-project-app/src/client/apps/main/components/Footer/style.scss

```
.footer {
  &__copyright {
    font-weight: bold;
  }
}
```

db-course-project-app/src/client/apps/main/components/Footer/Footer.jsx

```
import * as React from "react";

import "./style.scss"

const Footer = () => {
    const currDate = new Date();

    return (
        <footer className="footer">
            <hr/>
            <p className="footer__copyright">&#169; Copyright {currDate.getFullYear()}</p>
        </footer>
    );
};

export default Footer;
```

db-course-project-app/src/client/apps/main/components/Footer/index.js

```
import Footer from './Footer';

export default Footer;
```

...t-app/src/client/apps/main/components/ErrorFormAlert/ErrorFormAlert.jsx

```
import * as React from "react";
```

```
import PropTypes from "prop-types";

import Alert from "react-bootstrap/Alert";

const ErrorFormAlert = ({ listErrors, show, onHide }) => {
    return (
        <Alert variant="danger" show={show} onClose={onHide} dismissible>
            <Alert.Heading>You got an error!</Alert.Heading>
            <ul>
                {
                    listErrors.map(({ message }, i) => <li key={i}>{message}</li>)
                }
            </ul>
        </Alert>
    );
}

ErrorFormAlert.propTypes = {
    listErrors: PropTypes.array,
    show: PropTypes.bool,
    onHide: PropTypes.func
};

export default ErrorFormAlert;
```

...rse-project-app/src/client/apps/main/components/ErrorFormAlert/index.js

```
import ErrorFormAlert from "./ErrorFormAlert";

export default ErrorFormAlert;
```

...se-project-app/src/client/apps/main/components/HttpErrorInfo/style.scss

```
.http-error-info {
  align-items: center;
  display: flex;
  justify-content: center;
  min-height: 360px;

  &__box {
    margin: auto;
    text-align: center;
  }
}
```

...ect-app/src/client/apps/main/components/HttpErrorInfo/HttpErrorInfo.jsx

```
import * as React from "react";
import PropTypes from "prop-types";
import { getStatusText } from "http-status-codes";

import "./style.scss";

const HttpErrorInfo = ({ status, reason }) => {
    const reasonText = reason ? reason : getStatusText(status);

    return (
        <div className="http-error-info">
            <div className="http-error-info__box">
                <h3 className="http-error-info__status">{status}</h3>
                <p className="http-error-info__reason">{reasonText}</p>
            </div>
        </div>
    );
};

HttpErrorInfo.propTypes = {
    status: PropTypes.number,
    reason: PropTypes.string
```

```
}

export default HttpErrorInfo;
```

...urse-project-app/src/client/apps/main/components/HttpErrorInfo/index.js

```
import HttpErrorInfo from "./HttpErrorInfo";

export default HttpErrorInfo;
```

...project-app/src/client/apps/main/components/QuestionEditList/style.scss

```

```

...e-project-app/src/client/apps/main/components/QuestionEditList/index.js

```
import QuestionEditList from "./QuestionEditList";

export default QuestionEditList;
```

...p/src/client/apps/main/components/QuestionEditList/QuestionEditList.jsx

```
import * as React from "react";
import {ANSWER_TYPE} from "../AnswerEditList/config";
import PropTypes from "prop-types";

import QuestionEditItem from "../QuestionEditItem";

class QuestionEditList extends React.Component {
    constructor(props) {
        super(props);
    }

    render() {
        const {
            questions,
            deleteQuestion,
            updateQuestionTitle,
            changeQuestionType,
            appendAnswer
        } = this.props;

        return (
            <>
                {
                    questions.map((el, i) => {
                        const {
                            typeAnswer,
                            answers,
                            title
                        } = el;

                        return (
                            <QuestionEditItem name={i}
                                              key={i}
                                              typeAnswer={typeAnswer}
                                              answers={answers}
                                              title={title}
                                              onDelete={() => deleteQuestion(i)}
                                              onUpdateTitle={titleVal => updateQuestionTitle(i, titleVal)}
                                              onChangeAnswerType={typeAnswer => changeQuestionType(i,
                                              ↪  typeAnswer)}
                                              onAnswerListUpdate={id => appendAnswer(id)} />
                        );
                    })
                }
            </>
```

```
        );
    }
}

QuestionEditList.propTypes = {
    questions: PropTypes.arrayOf(
        PropTypes.exact({
            title: PropTypes.string,
            typeAnswer: PropTypes.oneOf([ ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE ]),
            answers: PropTypes.arrayOf(
                PropTypes.exact({
                    content: PropTypes.string,
                    isRight: PropTypes.bool
                })
            )
        }).isRequired
    ),
    deleteQuestion: PropTypes.func,
    updateQuestionTitle: PropTypes.func,
    changeQuestionType: PropTypes.func,
    appendAnswer: PropTypes.func
};

export default QuestionEditList;
```

db-course-project-app/src/client/apps/admin/Login.jsx

```
import * as React from "react";

import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";

const Login = () => {
    return (
        <Form>
            <Form.Group controlId="formBasicEmail">
                <Form.Label>Email address</Form.Label>
                <Form.Control type="email" placeholder="Enter email" />
            </Form.Group>
            <Form.Group controlId="formBasicPassword">
                <Form.Label>Password</Form.Label>
                <Form.Control type="password" placeholder="Password" />
            </Form.Group>
            <Form.Group controlId="formBasicCheckbox">
                <Form.Check type="checkbox" label="Check me out" />
            </Form.Group>
            <Button variant="primary" type="submit">Submit</Button>
        </Form>
    );
};

export default Login;
```

db-course-project-app/src/client/apps/admin/App.jsx

```
import * as React from 'react';
import { Switch, Route } from 'react-router-dom';

import Container from 'react-bootstrap/Container';

import Login from "./Login";

class App extends React.Component {
    render() {
        const App = () => (
            <div>
                <Container className="p-3">
                    <Switch>
                        <Route path='/admin/login' component={Login}/>
                    </Switch>
                </Container>
            </div>
```

```
        );

        return (
            <Switch>
                <App/>
            </Switch>
        );
    }
}

export default App;
```

db-course-project-app/src/client/actions/auth.js

```
export const LOGIN_APPROVE = 'LOGIN_APPROVE';
export const LOGIN_FAILED = 'LOGIN_FAILED';
export const LOGOUT = 'LOGOUT';

export const success = (user) => {
    return { type: LOGIN_APPROVE, user, isLoggedIn: true };
};

export const failed = () => {
    return { type: LOGIN_FAILED, user: null, isLoggedIn: false };
};

export const logOut = () => {
    return { type: LOGOUT, user: null, isLoggedIn: false };
};
```

db-course-project-app/src/client/actions/testEditor.js

```
export const RESET = "RESET";
export const UPDATE = "UPDATE";
export const UPDATE_TITLE = "UPDATE_TITLE";
export const UPDATE_DESCRIPTION = "UPDATE_DESCRIPTION";
export const APPEND_TAG = "APPEND_TAG";
export const DELETE_TAG = "DELETE_TAG";
export const APPEND_QUESTION = "APPEND_QUESTION";
export const DELETE_QUESTION = "DELETE_QUESTION";
export const UPDATE_QUESTION_TITLE = "UPDATE_QUESTION_TITLE";
export const UPDATE_QUESTION_TYPE = "UPDATE_QUESTION_TYPE";
export const APPEND_ANSWER = "APPEND_ANSWER";
export const DELETE_ANSWER = "DELETE_ANSWER";
export const UPDATE_ANSWER_TEXT = "UPDATE_ANSWER_TEXT";
export const UPDATE_ANSWERS = "UPDATE_ANSWERS";

export const reset = () => {
    return { type: RESET };
};

export const update = content => {
    return {
        type: UPDATE,
        content
    };
};

export const updateTitle = title => {
    return {
        type: UPDATE_TITLE,
        title
    };
};

export const updateDescription = description => {
    return {
        type: UPDATE_DESCRIPTION,
        description
    };
};

```

```
export const appendTag = tag => {
    return {
        type: APPEND_TAG,
        tag
    };
};

export const deleteTag = id => {
    return {
        type: DELETE_TAG,
        id
    };
};

export const appendQuestion = () => {
    return { type: APPEND_QUESTION };
};

export const deleteQuestion = id => {
    return {
        type: DELETE_QUESTION,
        id
    };
};

export const updateQuestionTitle = (id, title) => {
    return {
        type: UPDATE_QUESTION_TITLE,
        id,
        title
    };
};

export const changeQuestionType = (id, typeAnswer) => {
    return {
        type: UPDATE_QUESTION_TYPE,
        id,
        typeAnswer
    };
};

export const appendAnswer = questionId => {
    return { type: APPEND_ANSWER, questionId }
};

export const deleteAnswer = (questionId, answerId) => {
    return {
        type: DELETE_ANSWER,
        questionId,
        answerId
    };
};

export const updateAnswerText = (questionId, answerId, value) => {
    return {
        type: UPDATE_ANSWER_TEXT,
        questionId,
        answerId,
        value
    };
};

export const updateAnswers = (questionId, answerId, isRight, typeAnswer) => {
    return {
        type: UPDATE_ANSWERS,
        questionId,
        answerId,
        isRight,
        typeAnswer
    };
};
```

db-course-project-app/src/client/services/auth.js

```
import {removeToken} from "../helpers/token";

async function signUp(formData) {
    const response = await fetch("/api/signup", {
        method: "POST",
        body: formData
    });

    if (response.ok) {
        return Promise.resolve();
    } else {
        throw await response.json();
    }
}

async function signIn(formData) {
    const response = await fetch("/api/signin", {
        method: "POST",
        body: formData
    });

    const responseJson = await response.json();

    if (response.ok) {
        return Promise.resolve(responseJson);
    } else {
        throw responseJson;
    }
}

export function logOut() {
    removeToken();
}

export default {
    signUp,
    signIn,
    logOut
};
```

db-course-project-app/src/client/services/editProfileSettings.js

```
import {BAD_REQUEST, FORBIDDEN} from "http-status-codes";
import {appendAuth} from "../helpers/header";
import {getToken} from "../helpers/token";

const remove = async () => {
    const token = getToken();
    const headers = new Headers();

    appendAuth(headers, token);

    const response = await fetch("/api/profile/remove", {
        headers,
        method: 'POST'
    });

    if (response.ok) {
        return Promise.resolve();
    } else {
        switch (response.status) {
            case FORBIDDEN:
                return Promise.reject();
            default:
                break;
        }
    }
};

const updatePassword = async (formData) => {
    const token = getToken();
    const headers = new Headers();

```

```
    appendAuth(token);

    const response = await fetch("/api/profile/update-password", {
        method: 'POST',
        headers,
        body: formData
    });

    if (response.ok) {
        return Promise.resolve(null);
    } else {
        const responseJson = await response.json();

        switch (response.status) {
            case BAD_REQUEST:
                return Promise.reject(responseJson);
            default:
                break;
        }
    }
};

export { remove, updatePassword };
```

db-course-project-app/src/client/services/editTest.js

```
import {appendAuth, appendJSON} from "../helpers/header";
import {getToken} from "../helpers/token";

export const deleteTest = async testId => {
    const token = getToken();
    const headers = new Headers();

    appendAuth(headers, token);
    appendJSON(headers);

    const response = await fetch(`/api/test/delete`, {
        method: 'DELETE',
        headers,
        body: JSON.stringify({ testId })
    });

    if (response.ok) {
        return Promise.resolve();
    } else {
        // TODO: handle if something went wrong
        return Promise.reject();
    }
};

export const getOwnTests = async () => {
    const token = getToken();
    const headers = new Headers();

    appendAuth(headers, token);

    const response  = await fetch('/api/test/profile', {
        method: 'GET',
        headers
    });

    if (response.ok) {
        const responseJson = await response.json();

        return Promise.resolve(responseJson);
    } else {
        // TODO: handle if something wrong
        return Promise.reject();
    }
};

export const create = async (testData) => {
    const token = getToken();
    const headers = new Headers();
```

```
49
50      appendAuth(headers, token);
51      appendJSON(headers);
52
53      const response = await fetch('/api/test/create', {
54          method: 'POST',
55          body: JSON.stringify(testData),
56          headers,
57      });
58
59      if (response.ok) {
60          return Promise.resolve();
61      } else {
62          const responseJson = await response.json();
63
64          return Promise.reject(responseJson);
65      }
66  };
67
68  export const getTestForEdit = async testId => {
69      const token = getToken();
70      const headers = new Headers();
71
72      appendAuth(headers, token);
73      appendJSON(headers);
74
75      const response = await fetch('/api/test/update', {
76          method: 'POST',
77          body: JSON.stringify({ testId }),
78          headers,
79      });
80
81      const responseJson = await response.json();
82      if (response.ok) {
83          return Promise.resolve(responseJson);
84      } else {
85          return Promise.reject();
86      }
87  };
```

db-course-project-app/src/client/hoc/NotIsLoggedInRoute/index.js

```
1  import NotIsLoggedInRoute from "./NotIsLoggedInRoute";
2
3  export default NotIsLoggedInRoute;
```

...se-project-app/src/client/hoc/NotIsLoggedInRoute/NotIsLoggedInRoute.jsx

```
1  import * as React from "react";
2  import PropType from "prop-types";
3  import { connect } from "react-redux";
4  import { Route, Redirect } from "react-router-dom";
5
6  const NotIsLoggedInRoute = ({ component: Component, isLoggedIn, ...rest }) => {
7      return (
8          <Route { ...rest } render={() => (
9              !isLoggedIn ? (
10                 <Component />
11             ) : (
12                 <Redirect to="/" />
13             )
14         )} />
15     );
16 };
17
18 NotIsLoggedInRoute.propTypes = {
19     isLoggedIn: PropType.bool,
20     component: PropType.elementType,
21 };
22
23 function mapStateToProps(state) {
24     const { isLoggedIn } = state.auth;
```

```
25
26      return { isLoggedIn };
27  }
28
29  export default connect(mapStateToProps)(NotIsLoggedInRoute);
```

db-course-project-app/src/client/hoc/PrivateRoute/PrivateRoute.jsx

```
1   import * as React from "react";
2   import PropTypes from "prop-types";
3   import { connect } from "react-redux";
4   import { Route, Redirect } from "react-router-dom";
5
6   const PrivateRoute = ({ component: Component, isLoggedIn, ...rest }) => {
7       return (
8           <Route { ...rest } render={() => (
9               !isLoggedIn ? (
10                  <Redirect to="/login" />
11              ) : (
12                  <Component />
13              )
14          )}
15          />
16      );
17  };
18
19  PrivateRoute.propTypes = {
20      component: PropTypes.elementType,
21      isLoggedIn: PropTypes.bool
22  };
23
24  function mapStateToProps(state) {
25      const { isLoggedIn } = state.auth;
26
27      return { isLoggedIn };
28  }
29
30  const connectedPrivateRoute = connect(mapStateToProps)(PrivateRoute);
31
32  export { connectedPrivateRoute as PrivateRoute };
```

db-course-project-app/src/client/hoc/PrivateRoute/index.js

```
1   import { PrivateRoute } from "./PrivateRoute";
2
3   export default PrivateRoute;
```

db-course-project-app/src/routes/main.js

```
1   import { Router } from "express";
2
3   const router = new Router();
4
5   router.get("*", (req, res) => {
6       res.render("index");
7   });
8
9   export default router;
```

db-course-project-app/src/routes/auth.js

```
1   import multer from "multer";
2   import { Router } from "express";
3   import checkToken from "../middlewares/checkToken";
4   import * as authController from "../controllers/auth";
5
6   const upload = multer();
```

```

const router = new Router();

router.post('/signup', upload.none(), authController.signUp);
router.post('/signin', upload.none(), authController.signIn);
router.post('/init', checkToken, authController.initAuth);

export default router;
```

db-course-project-app/src/routes/profileModify.js

```
import Router from "express";
import multer from "multer";
import checkToken from "../middlewares/checkToken";
import * as profileModify from "../controllers/profileModify";

const upload = multer();
const router = new Router();

router.post("/update-password", upload.none(), checkToken, profileModify.updatePassword);
router.post("/remove", checkToken, profileModify.remove);

export default router;
```

db-course-project-app/src/routes/testEditor.js

```
import { Router } from "express";
import * as testEditor from "../controllers/testEditor";
import checkToken from "../middlewares/checkToken";

const router = new Router();

router.post("/create", checkToken, testEditor.create);
router.put("/update", checkToken, testEditor.update);
router.post("/update", checkToken, testEditor.getTestForEdit);
router.get("/profile", checkToken, testEditor.getOwnTests);
router.delete("/delete", checkToken, testEditor.deleteTest);

export default router;
```

db-course-project-app/src/tests/smoke.test.js

```
describe("Sum", () => {
    it("addiction", () => {
        expect(6).toBe(6);
    });
});
```

db-course-project-app/src/controllers/auth.js

```
import {
    INTERNAL_SERVER_ERROR,
    BAD_REQUEST,
    OK
} from "http-status-codes";
import * as jwt from "jsonwebtoken";
import FormListErrors from "../helpers/FormListErrors";
import User from "../models/User";

export const signUp = async (req, res, next) => {
    const {
        login,
        repeatPassword,
        password,
        email,
    } = req.body;
    const formListErrors = new FormListErrors();
```

```
18
19      if (repeatPassword !== password) {
20          formListErrors.add("passwords doesn't equal.");
21
22          next({
23              status: BAD_REQUEST,
24              errors: formListErrors.data.errors
25          });
26      }
27
28      try {
29          await User.create({ email, login, password });
30      } catch ({ errors }) {
31          formListErrors.addFromModelErrors(errors);
32
33          next({
34              status: BAD_REQUEST,
35              errors: formListErrors.data.errors
36          });
37      }
38
39      res.sendStatus(OK);
40  };
41
42  export const signIn = async (req, res, next) => {
43      const {
44          login,
45          password,
46      } = req.body;
47      const formListErrors = new FormListErrors();
48
49      let user;
50      try {
51          user = await User.findOne({ where: { login } });
52      } catch (error) {
53          formListErrors.addDefault();
54
55          next({
56              status: INTERNAL_SERVER_ERROR,
57              errors: formListErrors.data.errors
58          });
59      }
60
61      if (!user) {
62          formListErrors.add("user with such name not found.");
63
64          next({
65              status: BAD_REQUEST,
66              errors: formListErrors.data.errors
67          });
68      } else {
69          const isRightPassword = await user.comparePasswords(password);
70
71          if (isRightPassword) {
72              const token = jwt.sign({ sub: user.id }, process.env.JWT_SECRET);
73
74              res.json({
75                  ...user.initState(),
76                  token
77              });
78          } else {
79              formListErrors.add("password is invalid");
80
81              next({
82                  status: BAD_REQUEST,
83                  errors: formListErrors.data.errors
84              });
85          }
86      }
87  };
88
89  export const initAuth = async (req, res) => {
90      const user = await User.findByPk(req.userId);
91
92      res.json({ ...user.initState() });
93  };
```

db-course-project-app/src/controllers/profileModify.js

```
import User from "../models/User";
import {BAD_REQUEST, OK} from "http-status-codes";
import FormListErrors from "../helpers/FormListErrors";

export const updatePassword = async (req, res, next) => {
    const formListErrors = new FormListErrors();
    const { userId } = req;
    const { password, newPassword, repeatNewPassword } = req.body;

    const user = await User.findByPk(userId);

    const isRightPassword = await user.comparePasswords(password);

    if (isRightPassword) {
        if (repeatNewPassword !== newPassword) {
            formListErrors.add("passwords doesn't equal.");

            next({
                status: BAD_REQUEST,
                errors: formListErrors.data.errors
            });
        }

        try {
            await user.update({ password: newPassword });
        } catch ({ errors }) {
            formListErrors.addFromModelErrors(errors);

            next({
                status: BAD_REQUEST,
                errors: formListErrors.data.errors
            });
        }
    } else {
        formListErrors.add('current password is invalid.');

        next({
            status: BAD_REQUEST,
            errors: formListErrors.data.errors
        });
    }

    res.sendStatus(OK);
};

export const remove = async (req, res) => {
    const { userId } = req;

    const user = await User.findByPk(userId);

    await user.destroy();

    res.sendStatus(OK);
};
```

db-course-project-app/src/controllers/testEditor.js

```
import Test from "../models/Test";
import User from "../models/User";
import Tag from "../models/Tag";
import TestTag from "../models/TestTag";
import {BAD_REQUEST, OK} from "http-status-codes";

export const update = async (req, res) => {
    res.send();
};

export const create = async (req, res, next) => {
```

```
12      const { userId } = req;
13      const { info } = req.body;
14      const content = req.body.questions;
15      const { title, description, tags } = info;
16
17      try {
18          // create new tags
19          const createdTags = [];
20          for (let tagName of tags) {
21              const newTag = await Tag.create({ name: tagName });
22              createdTags.push(newTag);
23          }
24
25          // create new test
26          const newTest = await Test.create({
27              title,
28              description,
29              content,
30              userId,
31          }, {
32              include: Tag
33          });
34
35          // create link from tags to tests
36          for (let currTag of createdTags) {
37              await TestTag.create({
38                  testId: newTest.id,
39                  tagId: currTag.id
40              });
41          }
42      } catch (err) {
43          console.log(err);
44
45          next({
46              status: BAD_REQUEST,
47              errors: err.errors
48          });
49      }
50
51      res.sendStatus(OK);
52  };
53
54  export const getOwnTests = async (req, res) => {
55      const { userId } = req;
56
57      const tests = await Test.findAll({
58          where: { userId },
59          include: [User, Tag]
60      });
61
62      const response = [];
63      for (let test of tests) {
64          const { title, description, id } = test;
65          const { login } = test.user;
66          const tags = await test.getTags();
67
68          response.push({
69              title,
70              description,
71              tags: tags.map((tag => tag.name)),
72              author: login,
73              testId: id
74          });
75      }
76
77      res.json(response);
78  };
79
80  export const deleteTest = async (req, res) => {
81      const { testId } = req.body;
82      const { userId } = req;
83
84      const test = await Test.findByPk(testId, {
85          include: User
86      });
87
```

```
    if (!test) {
        // TODO: handle if test not found
    }

    if (test.userId === userId) {
        await Test.destroy({
            where: {
                id: testId
            }
        });

        res.sendStatus(OK);
    } else {
        // TODO: handle if testId != test.id
    }
};

export const getTestForEdit = async (req, res) => {
    const { testId } = req.body;
    const { userId } = req;

    const test = await Test.findByPk(testId, {
        include: [User, Tag]
    });

    if (!test) {
        // TODO: handle if test not found
    }

    if (test.userId === userId) {
        const tags = await test.getTags();
        const { title, description, content } = test;

        res.json({
            info: {
                title,
                description,
                tags: tags.map(tag => tag.name)
            },
            questions: content
        });
    } else {
        // TODO: handle if testId != test.id
    }
};
```

db-course-project-app/src/helpers/FormListErrors.js

```
// TODO: add documentation

export default class FormListErrors {
    constructor() {
        this.data = { errors: [] };
    }

    addFromModelErrors(errors) {
        this.data.errors.push(
            ...errors.map(err => {
                let { message } = err;

                return { message };
            })
        );
    }

    addDefault() {
        this.data.errors.push({
            message: "Oops, something went wrong."
        });
    }

    add(message) {
        this.data.errors.push({ message });
    }
```

```
27
28      isNotEmpty() {
29          return this.data.errors.length;
30      }
31  }
```

db-course-project-app/src/templates/admin.handlebars

```
1  <main id="root">
2      <!-- React entry point -->
3  </main>
4
5  <script src="/static/admin.bundle.js"></script>
```

db-course-project-app/src/templates/index.handlebars

```
1  <main id="root">
2      <!-- React entry point -->
3  </main>
4
5  <script src="/static/main.bundle.js"></script>
```

db-course-project-app/src/templates/layouts/main.handlebars

```
1  <html lang="en">
2  <head>
3      {{> meta }}
4      {{> favicon }}
5      <title>passquiz</title>
6  </head>
7  <body>
8      {{> loader }}
9      {{{ body }}}
10 </body>
11 </html>
```

db-course-project-app/src/templates/partials/favicon.handlebars

```
1  <link rel="apple-touch-icon" sizes="180x180" href="/static/apple-touch-icon.png">
2  <link rel="icon" type="image/png" sizes="32x32" href="/static/favicon-32x32.png">
3  <link rel="icon" type="image/png" sizes="192x192" href="/static/android-chrome-192x192.png">
4  <link rel="icon" type="image/png" sizes="16x16" href="/static/favicon-16x16.png">
5  <link rel="manifest" href="/static/site.webmanifest">
6  <meta name="apple-mobile-web-app-title" content="lms.labchecker.ru">
7  <meta name="application-name" content="lms.labchecker.ru">
8  <meta name="msapplication-TileColor" content="#00aba9">
9  <meta name="theme-color" content="#ffffff">
```

db-course-project-app/src/templates/partials/loader.handlebars

```
1  <style>
2      .loader {
3          position: fixed;
4          display: flex;
5          width: 100%;
6          height: 100%;
7      }
8
9      .loader_hide {
10         display: none;
11     }
12
13     .loader__dots {
14         margin: auto;
```

```
    }

    .loader__dot {
        width: 10px;
        height: 10px;
        display: inline-block;
        border-radius: 100%;
        background-color: #000;
        transition: opacity .4s;
        animation: .6s linear 0s infinite alternate fade-dot;
    }

    .loader__dot:nth-child(1) {
        animation-delay: .2s;
    }

    .loader__dot:nth-child(2) {
        animation-delay: .4s;
    }

    .loader__dot:nth-child(3) {
        animation-delay: .6s;
    }

    @keyframes fade-dot {
        from {
            opacity: 100%;
        }

        to {
            opacity: 0;
        }
    }
</style>

<div class="loader">
    <div class="loader__dots">
        <div class="loader__dot"></div>
        <div class="loader__dot"></div>
        <div class="loader__dot"></div>
    </div>
</div>
```

db-course-project-app/src/templates/partials/meta.handlebars

```
<meta charset="UTF-8">
<meta name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
      ↪  minimum-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
```

db-course-project-app/src/middlewares/checkToken.js

```
import * as jwt from "jsonwebtoken";

export default async (req, res, next) => {
    const token = req.headers["authorization"];

    let tokenObj = null;
    try {
        tokenObj = await jwt.verify(token, process.env.JWT_SECRET);
    } catch (err) {
        console.error(err);
    }

    req.userId = tokenObj.sub;

    next();
};
```

db-course-project-app/src/middlewares/errorHandler.js

```
import { INTERNAL_SERVER_ERROR } from "http-status-codes";

// eslint-disable-next-line no-unused-vars
export default function(err, req, res, next) {
    let { status, errors } = err;

    if (!status) {
        status = INTERNAL_SERVER_ERROR;
    }

    res.status(status).json({ errors });
}
```

db-course-project-app/src/models/TestTag.js

```
import { Sequelize } from "sequelize";

import config from "../config";

import Test from "./Test";
import Tag from "./Tag";

const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);

// setup many to many
const TestTag = sequelize.define("test_tag", {});

Test.belongsToMany(Tag, { through: TestTag });
Tag.belongsToMany(Test, { through: TestTag });

export default TestTag;
```

db-course-project-app/src/models/index.js

```
import Test from "./Test";
import User from "./User";
import Tag from "./Tag";
import Role from "./Role";
import UserRole from "./UserRole";
import TestTag from "./TestTag";

export {
    Test,
    User,
    Tag,
    Role,
    UserRole,
    TestTag,
};
```

db-course-project-app/src/models/UserRole.js

```
import { Sequelize } from "sequelize";

import config from "../config";

import User from "./User";
import Role from "./Role";

const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);

// setup many to many
const UserRole = sequelize.define("user_role", {});

User.belongsToMany(Role, { through: UserRole });
Role.belongsToMany(User, { through: UserRole });
```

```

export default UserRole;
```

db-course-project-app/src/models/Role.js

```
import { Sequelize } from "sequelize";

import config from "../config";

const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);

const Role = sequelize.define("role", {
    id: {
        type: Sequelize.INTEGER.UNSIGNED,
        autoIncrement: true,
        primaryKey: true
    },
    role_name: {
        type: Sequelize.STRING(64),
        allowNull: false,
        unique: true
    }
});

export default Role;
```

db-course-project-app/src/models/Tag/constraints.js

```
const MIN_TAG_NAME_LENGTH = 1;
const MAX_TAG_NAME_LENGTH = 16;

export {
    MIN_TAG_NAME_LENGTH,
    MAX_TAG_NAME_LENGTH
}
```

db-course-project-app/src/models/Tag/index.js

```
import Tag from "./Tag";

export default Tag;
```

db-course-project-app/src/models/Tag/Tag.js

```
import * as DataTypes from "sequelize";
import { Sequelize } from "sequelize";
import config from "../../config";

import * as tagConstraints from "./constraints";

const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);

const Tag = sequelize.define('tag', {
    id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true
    },
    name: {
        type: DataTypes.STRING,
        unique: true,
        allowNull: false,
        validate: {
            len: {
                msg: `tag must has length between ${tagConstraints.MIN_TAG_NAME_LENGTH} and
                ↪  ${tagConstraints.MAX_TAG_NAME_LENGTH}.`,
                args: [
```

```
                    tagConstraints.MIN_TAG_NAME_LENGTH,
                    tagConstraints.MAX_TAG_NAME_LENGTH
                ]
            }
        }
    }
});

export default Tag;
```

db-course-project-app/src/models/Test/constraints.js

```
const MIN_TITLE_LENGTH = 1;
const MAX_TITLE_LENGTH = 128;
const MIN_DESCRIPTION_LENGTH = 1;
const MAX_DESCRIPTION_LENGTH = 256;

export {
    MIN_TITLE_LENGTH,
    MAX_TITLE_LENGTH,
    MIN_DESCRIPTION_LENGTH,
    MAX_DESCRIPTION_LENGTH
};
```

db-course-project-app/src/models/Test/index.js

```
import Test from "./Test";

export default Test;
```

db-course-project-app/src/models/Test/Test.js

```
import * as DataTypes from "sequelize";
import { Sequelize } from "sequelize";
import config from "../../config";

import {
    MIN_TITLE_LENGTH,
    MAX_TITLE_LENGTH,
    MIN_DESCRIPTION_LENGTH,
    MAX_DESCRIPTION_LENGTH
} from "./constraints";

const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);

const Test = sequelize.define("test", {
    id: {
        type: Sequelize.INTEGER,
        autoIncrement: true,
        primaryKey: true
    },
    title: {
        type: Sequelize.STRING,
        allowNull: false,
        unique: true,
        validate: {
            len: {
                msg: `title must has length between ${MIN_TITLE_LENGTH} and ${MAX_TITLE_LENGTH}.`,
                args: [
                    MIN_TITLE_LENGTH,
                    MAX_TITLE_LENGTH
                ]
            }
        }
    },
    description: {
        type: Sequelize.STRING,
        allowNull: false,
        validate: {
```

```
            len: {
                msg: `description must has length between ${MIN_DESCRIPTION_LENGTH} and
                ↪  ${MAX_DESCRIPTION_LENGTH}.`,
                args: [
                    MIN_DESCRIPTION_LENGTH,
                    MAX_DESCRIPTION_LENGTH
                ]
            }
        }
    },
    content: {
        type: DataTypes.JSON,
        allowNull: false,
        // TODO: validation
    }
});

export default Test;
```

db-course-project-app/src/models/User/constraints.js

```
export default {
    // Password
    MIN_PASSWORD_LENGTH: 10,
    MAX_PASSWORD_LENGTH: 128,

    // Login
    MIN_LOGIN_LENGTH: 6,
    MAX_LOGIN_LENGTH: 128,

    // Email
    MAX_EMAIL_LENGTH: 320
};
```

db-course-project-app/src/models/User/User.js

```
import { Sequelize } from "sequelize";
import * as bcrypt from "bcrypt";

import config from "../../config";
import userConstraints from "./constraints";
import Test from "../Test";

const {
    MIN_PASSWORD_LENGTH,
    MAX_PASSWORD_LENGTH,
    MIN_LOGIN_LENGTH,
    MAX_LOGIN_LENGTH,
    MAX_EMAIL_LENGTH
} = userConstraints;

const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);

const User = sequelize.define("user", {
        id: {
            type: Sequelize.INTEGER,
            autoIncrement: true,
            primaryKey: true,
        },
        login: {
            type: Sequelize.STRING(MAX_LOGIN_LENGTH),
            allowNull: false,
            unique: true,
            validate: {
                len: {
                    msg: `login must has length between ${MIN_LOGIN_LENGTH} and ${MAX_LOGIN_LENGTH}.`,
                    args: [
                        MIN_LOGIN_LENGTH,
                        MAX_LOGIN_LENGTH
                    ]
                }
            }
```

```
        },
        password: {
            type: Sequelize.STRING(MAX_PASSWORD_LENGTH),
            allowNull: false,
            validate: {
                len: {
                    msg: `password must has length between ${MIN_PASSWORD_LENGTH} and
                    ↪  ${MAX_PASSWORD_LENGTH}.`,
                    args: [
                        MIN_PASSWORD_LENGTH,
                        MAX_PASSWORD_LENGTH
                    ]
                }
            }
        },
        email: {
            type: Sequelize.STRING(MAX_EMAIL_LENGTH),
            unique: true,
            allowNull: false,
            validate: {
                isEmail: {
                    msg: 'invalid email address.'
                }
            }
        },
    }
);

User.hashPassword = async (value) => {
    const salt = await bcrypt.genSalt(10);

    return await bcrypt.hash(value, salt);
};

User.prototype.comparePasswords = async function(password) {
    return await bcrypt.compare(password, this.password);
};

User.prototype.initState = function() {
    const { login } = this;

    return {
        user: { login }
    }
};

User.beforeCreate(async user => {
    user.password = await User.hashPassword(user.password);
});

User.beforeUpdate(async user => {
    user.password = await User.hashPassword(user.password);
});

User.hasMany(Test);
Test.belongsTo(User);

export default User;
```

db-course-project-app/src/models/User/index.js

```
import User from "./User";

export default User;
```

db-course-project-app/db/init_db.sql

```
CREATE TABLE IF NOT EXISTS users (
   user_id SERIAL NOT NULL,
   login VARCHAR(255) NOT NULL UNIQUE,
   password VARCHAR(255) NOT NULL,
   email VARCHAR(255) NOT NULL,
```

```
6     PRIMARY KEY (user_id)
7 );
```

Список использованных источников

[1] *CSS*. URL: https://ru.wikipedia.org/?oldid=107701928.

[2] *ECMAScript*. URL: https://ru.wikipedia.org/?oldid=108101383.

[3] *ESLint*. URL: https://eslint.org.

[4] *Express*. URL: https://expressjs.com.

[5] *Git*. URL: https://git-scm.com.

[6] *Heroku*. URL: https://heroku.com.

[7] *HTML*. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

[8] *JavaScript*. URL: https://ru.wikipedia.org/?oldid=107293496.

[9] *Jest*. URL: https://jestjs.io.

[10] *MongoDB*. URL: https://www.mongodb.com/.

[11] *Node.js*. URL: https://nodejs.org.

[12] *PostgreSQL*. URL: https://www.postgresql.org.

[13] *Python*. URL: https://www.python.org.

[14] *React*. URL: https://reactjs.org.

[15] *React Router*. URL: https://reactrouter.com.

[16] *Redux*. URL: https://redux.js.org.

[17] *SCSS*. URL: https://sass-lang.com.

[18] *Selenium with Python*. URL: https://selenium-python.readthedocs.io.

[19] *Sequelize*. URL: https://sequelize.org.

[20] *Stylelint*. URL: https://stylelint.io.

[21] *Webpack*. URL: https://webpack.js.org.