

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: Василийев Дмитрий Олегович

Группа: М8О-310Б-18

Преподаватель: Романенков Александр Михайлович

Дата: 7 ноября 2020 г.

Оценка: _____

Подпись преподавателя: _____

Подпись студента: _____

Москва 2020

Содержание

1	Введение	4
1.1	Формальные требования	4
1.2	Клиентские приложения	5
1.2.1	Тестирующая система	5
1.2.2	Система управления	5
1.3	Предметная область	5
1.4	Стэк технологий	5
1.5	Инструменты	6
2	Инфраструктура проекта	8
2.1	Архитектура	8
2.1.1	Связь логических компонент и применяемых технологий	9
2.2	Сущности	9
2.3	Сборка и запуск	10
2.3.1	Development	10
2.3.2	Production	11
2.4	Деплоинг	11
2.5	Организация работы с <i>Git</i> [5]	11
2.5.1	Git Workflow	11
2.5.2	Git hooks	11
3	Описание проекта	13
3.1	Разработка дизайна	13
3.2	Авторизации через JSON Web Token (JWT)	13
3.2.1	Авторизация	13
3.2.2	Аутентификация	13
3.2.3	JSON Web Token (JWT)	13
3.2.4	Реализация	14
3.3	Схема базы данных	15
4	Заключение	16
4.1	Недостатки	16
A	Визуализации структуры проекта	17

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 Архитектура

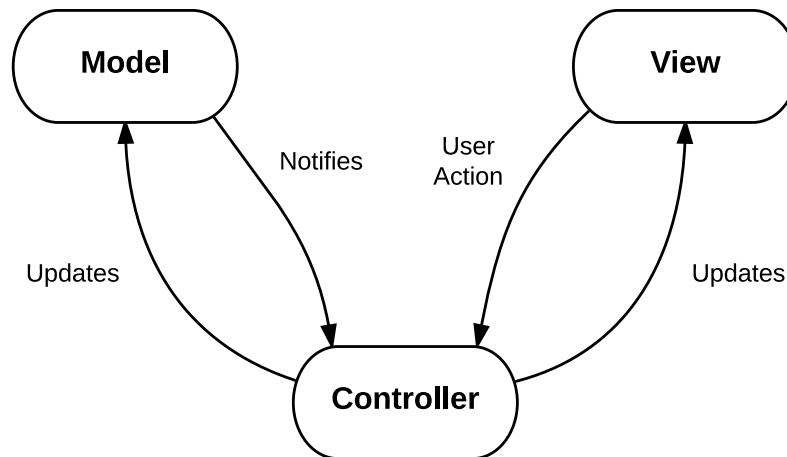


Рис. 1: Визуализация архитектуры MVC

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 компонентов и применяемых технологий

- **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 строиться база данных. В моём случае они следующие:

Сущность	Атрибуты
Пользователь	<ul style="list-style-type: none">• ID пользователя• Роли• Логин• Пароль• Email• Дата создания• Дата последнего изменения
Тест	<ul style="list-style-type: none">• ID теста• Название• Описание• Теги• Контент• Ответы• Дата создания• Дата последнего изменения

Тег	<ul style="list-style-type: none"> • ID тега • Название
Попытка	<ul style="list-style-type: none"> • ID попытки • ID пользователя • ID теста • Результат • Ответы пользователя • Дата прохождения

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 прм 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 клиентской части не занимало слишком много времени
2. клиент пересобирался при изменении какого-либо файла
3. сервер перезапускался при изменении 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 статических 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.

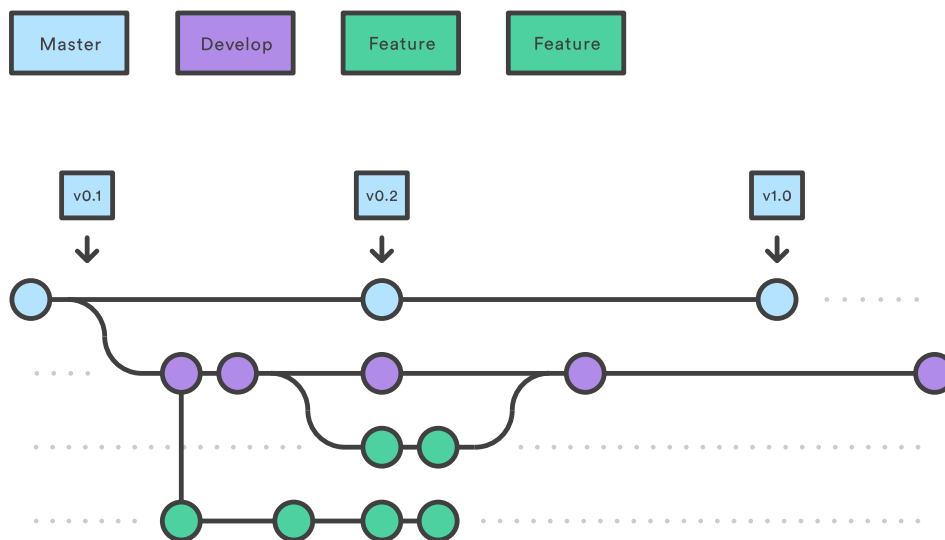


Рис. 2: Пример использования подхода Git Workflow

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.

```
1 {  
2   "hooks": {  
3     "pre-commit": "yarn es-lint && yarn style-lint && yarn test",  
4     "pre-push": "./venv/bin/pytest tests"  
5   }  
6 }
```

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 данных.

3.2.3 JSON Web Token (JWT)

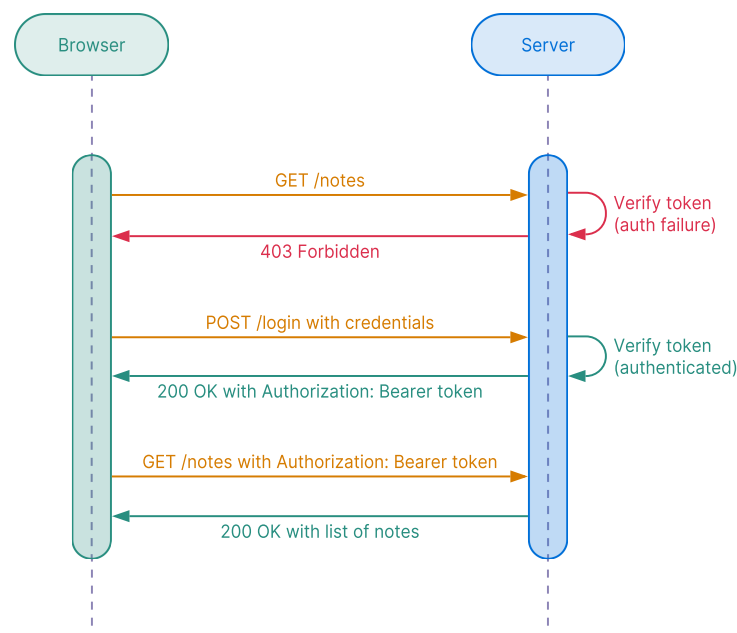


Рис. 3: Демонстрация работы JWT

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 строкой, которая состоит из трех частей, разделенных точками. Псевдокод вычисления подписи:

```
1 SECRET_KEY = 'some string';
2 unsignedToken = encodeBase64Url(header) + '.' + encodeBase64Url(payload)
3 signature = SHA256(unsignedToken, SECRET_KEY);
4
5 // собираем всё вместе
6 jwt = encodeBase64Url(header) + '.' + encodeBase64Url(payload) + '.' + encodeBase64Url(signature);
```

3.2.4 Реализация

Авторизация проходит следующим 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 токен будет использоваться для доступа к защищённым ресурсам.

3.3 Схема базы данных

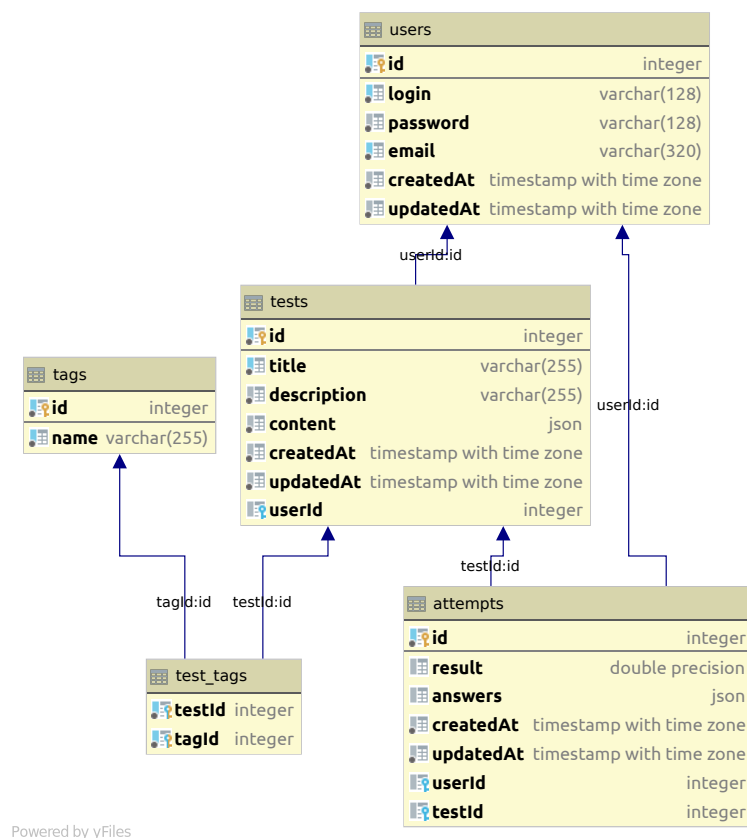


Рис. 4: Схема базы данных

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 Modifier (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 реляционными. Хранить JSON в таблице плохо, поэтому для этой задачи подходит *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
----- main.js
----- setupTests.js
----- .stylelintrc
----- config.js
----- index.js
----- webpack.config.js
----- .eslintrc.js
----- /client
----- contest.jsx
----- main.jsx
----- /containers
----- /TestEditorQuestionList
----- index.js
----- TestEditorQuestionList.jsx
----- /TestEditorTagList
```

```
----- TestEditorTagList.jsx
----- index.js
----- /Test
----- index.js
----- Test.jsx
----- /reducers
----- auth.js
----- testPassing.js
----- index.js
----- testEditor.js
----- /pages
----- /SignUp
----- style.scss
----- index.js
----- SignUp.jsx
----- /Test
----- style.scss
----- index.js
----- Test.jsx
----- /TestResult
----- style.scss
----- index.js
----- TestResult.jsx
----- /Login
----- style.scss
----- Login.jsx
----- index.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
----- /TestEditor
----- style.scss
----- index.js
----- TestEditor.jsx
----- /Profile
----- style.scss
----- Profile.jsx
----- index.js
----- /ProfileAttempts
----- style.scss
```

```

----- ProfileAttempts.jsx
----- index.js
----- /Home
----- style.scss
----- index.js
----- Home.jsx
----- /TestStatistic
----- TestStatistic.jsx
----- index.js
----- /components
----- /Header
----- style.scss
----- Header.jsx
----- index.js
----- /AnswerEditList
----- style.scss
----- index.js
----- AnswerEditList.jsx
----- /QuestionEditItem
----- QuestionEditItem.jsx
----- index.js
----- /AnswerEditItem
----- style.scss
----- AnswerEditItem.jsx
----- index.js
----- /TableStatistic
----- index.js
----- TableStatistic.jsx
----- /QuestionEditList
----- style.scss
----- index.js
----- QuestionEditList.jsx
----- /contest
----- App.jsx

```

```
----- /pages
----- /AllTests
----- style.scss
----- AllTests.jsx
----- index.js
----- /Home
----- style.scss
----- index.js
----- Home.jsx
----- /components
----- /Header
----- style.scss
----- Header.jsx
----- index.js
----- /components
----- /TagList
----- style.scss
----- index.js
----- TagList.jsx
----- /Tag
----- style.scss
----- index.js
----- Tag.jsx
----- /ResultStatus
----- style.scss
----- ResultStatus.jsx
----- index.js
----- /TestCard
----- style.scss
----- index.js
----- TestCard.jsx
----- /Question
----- style.scss
----- index.js
```

```
----- Question.jsx
----- /ListTestCards
----- style.scss
----- index.js
----- ListTestCards.jsx
----- /AnswerList
----- style.scss
----- index.js
----- AnswerList.jsx
----- /Answer
----- style.scss
----- Answer.jsx
----- index.js
----- /Footer
----- style.scss
----- Footer.jsx
----- index.js
----- /ErrorFormAlert
----- ErrorFormAlert.jsx
----- index.js
----- /HttpErrorInfo
----- style.scss
----- HttpErrorInfo.jsx
----- index.js
----- /actions
----- auth.js
----- testPassing.js
----- testEditor.js
----- /history
----- index.js
----- /services
----- testResult.js
----- attempt.js
----- auth.js
```

```
----- testPassing.js
----- editProfileSettings.js
----- editTest.js
----- /hoc
----- /NotIsLoggedInRoute
----- index.js
----- NotIsLoggedInRoute.jsx
----- /PrivateRoute
----- PrivateRoute.jsx
----- index.js
----- /routes
----- main.js
----- attempt.js
----- auth.js
----- profileModify.js
----- testEditor.js
----- /tests
----- smoke.test.js
----- /services
----- /controllers
----- attempt.js
----- auth.js
----- profileModify.js
----- testEditor.js
----- /helpers
----- FormListErrors.js
----- /templates
----- contest.handlebars
----- index.handlebars
----- /layouts
----- main.handlebars
----- /partials
----- favicon.handlebars
----- loader.handlebars
```



```
----- meta.handlebars
----- /middlewares
----- checkToken.js
----- errorHandler.js
----- /models
----- TestTag.js
----- index.js
----- /Tag
----- constraints.js
----- index.js
----- Tag.js
----- /Test
----- constraints.js
----- config.js
----- index.js
----- Test.js
----- /User
----- constraints.js
----- User.js
----- index.js
----- /Attempt
----- Attempt.js
----- index.js
--- /db
----- init_db.sql
```

В Код проекта

db-course-project-app/package.json

```
1 {
2   "name": "db-course-project-app",
3   "version": "1.0.0",
4   "description": "Web-application for course project by Database.",
5   "main": "build/index.js",
6   "author": "Dmitry Vasiliev",
7   "scripts": {
8     "test": "jest",
9     "start-dev": "nodemon --config \"/util/nodemon.json\"/",
10    "build": "rm -rf ./build/* ./src/public/contest.bundle.js ./src/public/main.bundle.js && babel src -d
11    ↪ build && webpack --config src/webpack.config.js --mode=\"production\"",
12    "start": "node -r dotenv/config build/index.js",
13    "es-lint": "eslint . -c src/.eslintrc.js --ext \"jsx,js\"",
14    "style-lint": "stylelint --ignore-pattern src/client/tests --config src/.stylelintrc src/client/*",
15    "watch": "webpack --config src/webpack.config.js --watch",
16    "desktop-dev": "electron ./src/main.js",
17    "desktop-start": "electron ./build/main.js"
18  },
19  "dependencies": {
20    "@fortawesome/fontawesome-svg-core": "^1.2.32",
21    "@fortawesome/free-brands-svg-icons": "^5.15.1",
22    "@fortawesome/free-regular-svg-icons": "^5.15.1",
23    "@fortawesome/free-solid-svg-icons": "^5.15.1",
24    "@fortawesome/react-fontawesome": "^0.1.11",
25    "bcrypt": "^5.0.0",
26    "bootstrap": "^4.5.0",
27    "compression": "^1.7.4",
28    "dotenv": "^8.2.0",
29    "express": "^4.17.1",
30    "express-handlebars": "^5.0.0",
31    "history": "^5.0.0",
32    "http-status-codes": "^1.4.0",
33    "jsonwebtoken": "^8.5.1",
34    "lodash": "^4.17.19",
35    "mdbreact": "^4.27.0",
36    "morgan": "^1.10.0",
37    "multer": "^1.4.2",
38    "pg": "^8.3.0",
39    "pg-hstore": "^2.3.3",
40    "prop-types": "^15.7.2",
41    "pug": "^3.0.0",
42    "react": "^16.13.1",
43    "react-bootstrap": "^1.2.2",
44    "react-dom": "^16.13.1",
45    "react-redux": "^7.2.1",
46    "react-router-bootstrap": "^0.25.0",
47    "react-router-dom": "^5.2.0",
48    "react-router-prop-types": "^1.0.5",
49    "redux": "^4.0.5",
50    "sequelize": "^6.3.3",
51    "serve-favicon": "^2.5.0",
52    "validate.js": "^0.13.1"
53  },
54  "devDependencies": {
55    "@babel/cli": "^7.11.6",
56    "@babel/core": "^7.11.6",
57    "@babel/node": "^7.10.5",
58    "@babel/plugin-transform-runtime": "^7.11.5",
59    "@babel/preset-env": "^7.11.5",
60    "@babel/preset-react": "^7.10.4",
61    "@types/compression": "^1.7.0",
62    "@types/dotenv": "^8.2.0",
63    "@types/express": "^4.17.7",
64    "@types/express-handlebars": "^3.1.0",
65    "@types/history": "^4.7.8",
66    "@types/jest": "^26.0.4",
67    "@types/jsonwebtoken": "^8.5.0",
68    "@types/lodash": "^4.14.161",
69    "@types/morgan": "^1.9.1",
70    "@types/multer": "^1.4.4",
```

```

70     "@types/node": "~14.0.22",
71     "@types/react": "~16.9.43",
72     "@types/react-dom": "~16.9.8",
73     "@types/react-redux": "~7.1.9",
74     "@types/react-router-bootstrap": "~0.24.5",
75     "@types/react-router-dom": "~5.1.5",
76     "@types/redux": "~3.6.0",
77     "@types/sequelize": "~4.28.9",
78     "@types/serve-favicon": "~2.5.0",
79     "babel-loader": "~8.1.0",
80     "css-loader": "~3.6.0",
81     "electron": "~10.1.5",
82     "eslint": "~7.9.0",
83     "eslint-plugin-jest": "~24.0.1",
84     "eslint-plugin-react": "~7.20.6",
85     "husky": "~4.2.5",
86     "jest": "~26.1.0",
87     "jest-fetch-mock": "~3.0.3",
88     "lorem-ipsu": "~2.0.3",
89     "nodemon": "~2.0.4",
90     "sass": "~1.26.10",
91     "sass-loader": "~9.0.2",
92     "style-loader": "~1.2.1",
93     "stylelint": "~13.6.1",
94     "stylelint-config-sass-guidelines": "~7.0.0",
95     "webpack": "~4.43.0",
96     "webpack-cli": "~3.3.12"
97   },
98   "husky": {
99     "hooks": {
100       "pre-commit": "yarn es-lint && yarn style-lint && yarn test",
101       "pre-push": "./venv/bin/pytest tests"
102     }
103   }
104 }

```

db-course-project-app/.babelrc

```

1 {
2   "presets": ["@babel/preset-env", "@babel/preset-react"],
3   "plugins": [
4     "@babel/plugin-transform-runtime"
5   ]
6 }

```

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

```

1 module.exports = {
2   setupFiles: ["./src/setupTests.js"],
3   testEnvironment: 'node',
4   testRegex: '(/src/tests/|/src/client/tests/).*\\. (test|spec)?\\. (js|jsx)$',
5   moduleFileExtensions: ['js', 'jsx', 'json', 'node']
6 };

```

db-course-project-app/requirements.txt

```

1 attrs==19.3.0
2 iniconfig==1.0.1
3 more-itertools==8.4.0
4 packaging==20.4
5 pluggy==0.13.1
6 py==1.9.0
7 pyparsing==2.4.7
8 pytest==6.0.1
9 selenium==3.141.0
10 six==1.15.0
11 toml==0.10.1
12 urllib3==1.25.10

```

db-course-project-app/README.md

```
1 # db-course-project-app
2 :book: Web-application for course project by Database
3
4 Link to description about project:
  ↳ [db-course-project-report](https://github.com/swimmwatch/db-course-project-report)
```

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

```
1 {
2   "watch": ["src"],
3   "ext": "js",
4   "ignore": ["src/public"],
5   "exec": "babel-node -r dotenv/config src/index.js"
6 }
```

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

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

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

```
1 const electron = require('electron');
2 // Module to control application life.
3 const app = electron.app;
4 // Module to create native browser window.
5 const BrowserWindow = electron.BrowserWindow;
6
7 // Keep a global reference of the window object, if you don't, the window will
8 // be closed automatically when the JavaScript object is garbage collected.
9 let mainWindow;
10
11 function createWindow() {
12   // Create the browser window.
13   mainWindow = new BrowserWindow({
14     width: 800,
15     height: 600
16   });
17
18   // and load the index.html of the app.
19   mainWindow.loadURL('http://localhost:3000');
20
21   // Open the DevTools.
22   // mainWindow.webContents.openDevTools();
23
24   // Emitted when the window is closed.
25   mainWindow.on('closed', function () {
26     // Dereference the window object, usually you would store windows
27     // in an array if your app supports multi windows, this is the time
28     // when you should delete the corresponding element.
29     mainWindow = null;
30   });
31 }
32
33 // This method will be called when Electron has finished
34 // initialization and is ready to create browser windows.
35 // Some APIs can only be used after this event occurs.
36 app.on('ready', createWindow);
37
38 // Quit when all windows are closed.
39 app.on('window-all-closed', function() {
40   // On OS X it is common for applications and their menu bar
41   // to stay active until the user quits explicitly with Cmd + Q
42   if (process.platform !== 'darwin') {
43     app.quit();
44   }
45 });
```

```

46
47 app.on('activate', function() {
48   // On OS X it's common to re-create a window in the app when the
49   // dock icon is clicked and there are no other windows open.
50   if (mainWindow === null) {
51     createWindow();
52   }
53 });

```

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

```

1 import fetchMock from "jest-fetch-mock";
2
3 fetchMock.enableMocks();

```

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

```

1 {
2   "extends": "stylelint-config-sass-guidelines"
3 }

```

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

```

1 import * as path from "path";
2 import express from "express";
3 import expubs from "express-handlebars";
4 import compression from "compression";
5 import morgan from "morgan";
6 import serveFavicon from "serve-favicon";
7 import { Sequelize } from "sequelize";
8
9 import config from "./config";
10
11 import mainRouter from "./routes/main";
12 import authRouter from "./routes/auth";
13 import testEditorRouter from "./routes/testEditor";
14 import profileModify from "./routes/profileModify";
15 import attemptsRouter from "./routes/attempts";
16 import errorHandler from "./middlewares/errorHandler";
17
18 import models from "./models";
19
20 const app = express();
21
22 // set static path
23 app.use("/static", express.static("src/public"));
24
25 // set template engine
26 app.set('views', path.join(process.cwd(), '/src', '/templates'));
27 app.engine('handlebars', expubs());
28 app.set('view engine', 'handlebars');
29
30 // set response compression
31 app.use(compression());
32 // set logger
33 app.use(morgan("common"));
34 // serve json requests
35 app.use(express.json());
36 // serve form requests
37 app.use(express.urlencoded({ extended: true }));
38 // serve favicon
39 app.use(serveFavicon(path.join(process.cwd(), '/src', '/public', 'favicon.ico')))
40
41 const PORT = process.env.PORT || 3000;
42
43 const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
44
45 app.use("/api", authRouter);
46 app.use("/api/profile", profileModify);

```

```

47 app.use("/api/attempt", attemptsRouter);
48 app.use("/api/test", testEditorRouter);
49
50 app.use("*", mainRouter);
51
52 app.use(errorHandler);
53
54 app.listen(PORT, async () => {
55   try {
56     await sequelize.authenticate();
57     await sequelize.sync({ force: true });
58
59     for (let model of models) {
60       await model.sync();
61     }
62
63     console.log('Connection has been established successfully.');
```

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

```

1 const path = require('path');
2
3 module.exports = {
4   mode: 'development',
5   entry: {
6     main: './src/client/main.jsx',
7     contest: './src/client/contest.jsx'
8   },
9   devtool: 'source-map',
10  module: {
11    rules: [
12      {
13        test: /\.jsx?$/,
14        use: 'babel-loader',
15        exclude: /node_modules/,
16      },
17      {
18        test: /\.s[ac]ss$/i,
19        use: [
20          // Creates `style` nodes from JS strings
21          'style-loader',
22          // Translates CSS into CommonJS
23          'css-loader',
24          // Compiles Sass to CSS
25          'sass-loader',
26        ],
27      },
28    ],
29  },
30  resolve: {
31    extensions: [ '.jsx', '.js' ],
32  },
33  output: {
34    filename: '[name].bundle.js',
35    path: path.resolve(__dirname, 'public'),
36  },
37 };
```

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

```

1 module.exports = {
2   "env": {
3     "browser": true,
4     "es2020": true,
5     "node": true,
6     "jest": true

```

```

7   },
8   "extends": [
9     "eslint:recommended",
10    "plugin:react/recommended",
11    "plugin:jest/recommended"
12  ],
13  "parserOptions": {
14    "ecmaFeatures": {
15      "jsx": true
16    },
17    "ecmaVersion": 11,
18    "sourceType": "module"
19  },
20  "plugins": [
21    "react",
22    "jest"
23  ],
24  "rules": {
25  }
26 };

```

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

```

1  import * as React from "react";
2  import * as ReactDOM from "react-dom";
3  import { Provider } from "react-redux";
4  import { BrowserRouter as Router } from "react-router-dom";
5  import { toggleLoader } from "../helpers/loader";
6
7  import App from "../apps/contest/App";
8  import { store, initAuthStore } from "../store";
9  import history from "../history";
10
11 import "bootstrap/scss/bootstrap.scss";
12
13 initAuthStore(store).then(() => {
14   toggleLoader();
15
16   ReactDOM.render(
17     <Provider store={store}>
18       <Router history={history}>
19         <App />
20       </Router>
21     </Provider>,
22     document.getElementById("root")
23   );
24 });

```

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

```

1  import * as React from "react";
2  import * as ReactDOM from "react-dom";
3  import { Provider } from "react-redux";
4  import { BrowserRouter } from "react-router-dom";
5  import { toggleLoader } from "../helpers/loader";
6
7  import App from "../apps/main/App";
8  import { store, initAuthStore } from "../store";
9  import history from "../history";
10
11 import "bootstrap/scss/bootstrap.scss";
12
13 initAuthStore(store).then(() => {
14   toggleLoader();
15
16   ReactDOM.render(
17     <Provider store={store}>
18       <BrowserRouter history={history}>
19         <App />
20       </BrowserRouter>
21     </Provider>,
22     document.getElementById("root")

```

```
23     );
24   });
```

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

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

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

```
1 import {connect} from "react-redux";
2 import * as testEditorActions from "../../actions/testEditor";
3 import QuestionEditList from "../../client/apps/main/components/QuestionEditList";
4
5 function mapStateToProps(state) {
6   const { questions } = state.testEditor;
7
8   return { questions };
9 }
10
11 function mapDispatchToProps(dispatch) {
12   return {
13     deleteQuestion: i => dispatch(testEditorActions.deleteQuestion(i)),
14     updateQuestionTitle: (i, title) => dispatch(testEditorActions.updateQuestionTitle(i, title)),
15     changeQuestionType: (i, typeAnswer) => dispatch(testEditorActions.changeQuestionType(i,
16       ↪ typeAnswer)),
17     appendAnswer: i => dispatch(testEditorActions.appendAnswer(i))
18   };
19 }
20 const TestEditorQuestionList = connect(mapStateToProps, mapDispatchToProps)(QuestionEditList);
21
22 export { TestEditorQuestionList };
```

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

```
1 import {connect} from "react-redux";
2 import TagList from "../../client/components/TagList";
3 import * as testEditorActions from "../../actions/testEditor";
4
5 function mapStateToProps(state) {
6   const { tags } = state.testEditor.info;
7
8   return { tags };
9 }
10
11 function mapDispatchToProps(dispatch) {
12   return {
13     deleteTag: (i) => dispatch(testEditorActions.deleteTag(i))
14   };
15 }
16
17 const TestEditorTagList = connect(mapStateToProps, mapDispatchToProps)(TagList);
18
19 export { TestEditorTagList };
```

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

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

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

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

db-course-project-app/src/client/containers/Test/Test.jsx

```
1 import { connect } from "react-redux";
2 import * as testPassingActions from "../../actions/testPassing";
3 import Test from "../../pages/Test";
4
5 function mapStateToProps(state) {
6     const { questions } = state.testPassing;
7
8     return { questions };
9 }
10
11 function mapDispatchToProps(dispatch) {
12     return {
13         setInitData: initState =>
14             dispatch(testPassingActions.init(initState)),
15         updateAnswer: (questionId, answerId, typeAnswer, state) =>
16             dispatch(testPassingActions.updateAnswer(questionId, answerId, typeAnswer, state))
17     };
18 }
19
20 const connectedTest = connect(mapStateToProps, mapDispatchToProps)(Test);
21
22 export { connectedTest as Test };
```

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

```
1 import * as authActions from "../actions/auth";
2 import * as authService from "../services/auth";
3
4 let initState = { isLoggedIn: false, user: null }
5
6 export default (state = initState, action) => {
7     const { isLoggedIn, user } = action;
8
9     switch (action.type) {
10         case authActions.LOGIN_APPROVE:
11             return { isLoggedIn, user };
12         case authActions.LOGIN_FAILED:
13             return { isLoggedIn, user };
14         case authActions.LOGOUT:
15             authService.logout();
16
17             return { isLoggedIn, user };
18         default:
19             return state;
20     }
21 };
```

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

```
1 import * as testPassingActions from "../../actions/testPassing";
2 import {ANSWER_TYPE} from "../../models/Test/config";
3
4
5 const initState = {
6     questions: []
7 };
8
9 export default (state = initState, action) => {
10     let newState = JSON.parse(JSON.stringify(state));
11
12     switch (action.type) {
13         case testPassingActions.INIT: {
```

```

14         const { internalState } = action;
15
16         return { questions: internalState };
17     }
18     case testPassingActions.UPDATE_ANSWER: {
19         const {
20             questionId,
21             answerId,
22             typeAnswer,
23             state
24         } = action;
25         const { questions } = newState;
26         const { answers } = questions[questionId];
27
28         // reset states others answers
29         if (typeAnswer === ANSWER_TYPE.ONE) {
30             for (let currAnswer of answers) {
31                 currAnswer.isChecked = false;
32             }
33         }
34
35         answers[answerId].isChecked = state;
36
37         return newState;
38     }
39     default:
40         return state;
41 }
42 };

```

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

```

1 import { combineReducers } from "redux";
2
3 import auth from "../auth";
4 import testEditor from "../testEditor";
5 import testPassing from "../testPassing";
6
7 export default combineReducers({ auth, testEditor, testPassing });

```

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

```

1 import * as testEditorActions from "../actions/testEditor";
2 import { createQuestion, createAnswer } from "../helpers/question";
3 import { ANSWER_TYPE } from "../../models/Test/config";
4
5 let initState = {
6     info: {
7         title: '',
8         description: '',
9         tags: []
10    },
11    questions: [
12        createQuestion(),
13    ]
14 };
15
16 export default (state = initState, action) => {
17     let newState = JSON.parse(JSON.stringify(state));
18
19     switch (action.type) {
20         case testEditorActions.RESET:
21             return { ...initState };
22         case testEditorActions.UPDATE:
23             return action.content;
24         case testEditorActions.UPDATE_TITLE: {
25             newState.info.title = action.title;
26
27             return newState;
28         }
29         case testEditorActions.UPDATE_DESCRIPTION: {
30             newState.info.description = action.description;

```

```

31
32     return newState;
33 }
34 case testEditorActions.APPEND_TAG: {
35     const tagIsNotInList = !newState.info.tags.includes(action.tag);
36
37     if (action.tag.length && tagIsNotInList) {
38         newState.info.tags.push(action.tag);
39     }
40
41     return newState;
42 }
43 case testEditorActions.DELETE_TAG: {
44     const { id } = action;
45
46     newState.info.tags.splice(id, 1);
47
48     return newState;
49 }
50 case testEditorActions.APPEND_QUESTION: {
51     newState.questions.push(createQuestion());
52
53     return newState;
54 }
55 case testEditorActions.DELETE_QUESTION: {
56     const { id } = action;
57
58     if (newState.questions.length > 1) {
59         newState.questions.splice(id, 1);
60     }
61
62     return newState;
63 }
64 case testEditorActions.UPDATE_QUESTION_TITLE: {
65     const { id, title } = action;
66
67     newState.questions[id].title = title;
68
69     return newState;
70 }
71 case testEditorActions.UPDATE_QUESTION_TYPE: {
72     const { id, typeAnswer } = action;
73     const { answers } = newState.questions[id];
74
75     for (let currAnswer of answers) {
76         currAnswer.isRight = false;
77     }
78
79     newState.questions[id].typeAnswer = typeAnswer;
80
81     return newState;
82 }
83 case testEditorActions.APPEND_ANSWER: {
84     const { questionId } = action;
85
86     newState.questions[questionId].answers.push(createAnswer());
87
88     return newState;
89 }
90 case testEditorActions.DELETE_ANSWER: {
91     const { questionId, answerId } = action;
92     const { answers } = newState.questions[questionId];
93
94     if (answers.length > 2) {
95         answers.splice(answerId, 1);
96     }
97
98     newState.questions[questionId].answers = answers;
99
100     return newState;
101 }
102 case testEditorActions.UPDATE_ANSWER_TEXT: {
103     const { questionId, answerId, value } = action;
104     const { answers } = newState.questions[questionId];
105
106     answers[answerId].content = value;

```

```

107
108     return newState;
109 }
110 case testEditorActions.UPDATE_ANSWERS: {
111     const { questionId, answerId, isRight, typeAnswer } = action;
112     const { answers } = newState.questions[questionId];
113
114     if (typeAnswer === ANSWER_TYPE.ONE) {
115         for (let currAnswer of answers) {
116             currAnswer.isRight = false;
117         }
118     }
119
120     answers[answerId].isRight = isRight;
121
122     return newState;
123 }
124 default:
125     return state;
126 }
127 };

```

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

```

1 .main-signup-form {
2   &__title {
3     text-align: center;
4   }
5
6   &__label {
7     font-weight: bold;
8   }
9 }

```

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

```

1 import SignUp from "./SignUp";
2
3 export default SignUp;

```

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

```

1 import * as React from "react";
2 import ReactRouterPropTypes from "react-router-prop-types";
3 import { withRouter } from "react-router-dom";
4
5 import Form from "react-bootstrap/Form";
6 import Button from "react-bootstrap/Button";
7 import Container from "react-bootstrap/Container";
8 import Col from "react-bootstrap/Col";
9 import authService from "../../services/auth";
10
11 import ErrorFormAlert from "../../components/ErrorFormAlert";
12
13 import userConstraints from "../../models/User/constraints";
14
15 import "./style.scss";
16
17 const {
18   MIN_PASSWORD_LENGTH,
19   MAX_PASSWORD_LENGTH,
20   MIN_LOGIN_LENGTH,
21   MAX_LOGIN_LENGTH,
22   MAX_EMAIL_LENGTH
23 } = userConstraints;
24
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>
101                 <Form>
102                     <ErrorFormAlert listErrors={listErrors}
103                         show={listErrors.length !== 0}

```

```

104         onHide={this.hideErrorAlert} />
105     <Form.Group controlId="main-signup-form__email">
106         <Form.Control type="email"
107             placeholder="Enter email"
108             name="email"
109             maxLength={MAX_EMAIL_LENGTH}
110             required
111             onChange={this.handleInputChange} />
112     </Form.Group>
113     <Form.Group controlId="main-signup-form__login">
114         <Form.Control type="text"
115             placeholder="Enter login"
116             name="login"
117             minLength={MIN_LOGIN_LENGTH}
118             maxLength={MAX_LOGIN_LENGTH}
119             required
120             onChange={this.handleInputChange} />
121     </Form.Group>
122     <Form.Group controlId="main-signup-form__password">
123         <Form.Control type="password"
124             placeholder="Enter password"
125             name="password"
126             minLength={MIN_PASSWORD_LENGTH}
127             maxLength={MAX_PASSWORD_LENGTH}
128             required
129             onChange={this.handleInputChange} />
130     </Form.Group>
131     <Form.Group controlId="main-signup-form__repeat-password">
132         <Form.Control type="password"
133             placeholder="Enter password"
134             name="repeatPassword"
135             minLength={MIN_PASSWORD_LENGTH}
136             maxLength={MAX_PASSWORD_LENGTH}
137             required
138             onChange={this.handleInputChange} />
139     </Form.Group>
140     <Button variant="primary"
141         type="submit"
142         block
143         disabled={isLoading}
144         onClick={this.handleFormSubmit}>
145         { isLoading ? 'Loading...' : 'Submit' }
146     </Button>
147 </Form>
148 </Col>
149 </Container>
150 );
151 }
152 }
153
154 SignUp.propTypes = {
155     history: ReactRouterPropTypes.history
156 };
157
158 export default withRouter(SignUp);

```

db-course-project-app/src/client/pages/Test/style.scss

1

db-course-project-app/src/client/pages/Test/index.js

```

1 import Test from "../Test";
2
3 export default Test;

```

db-course-project-app/src/client/pages/Test/Test.jsx

```

1 import * as React from "react";

```

```

2 import PropTypes from "prop-types";
3 import ReactRouterPropTypes from "react-router-prop-types";
4 import { withRouter } from "react-router-dom";
5 import Container from "react-bootstrap/Container";
6 import Row from "react-bootstrap/Row";
7 import Col from "react-bootstrap/Col";
8 import Button from "react-bootstrap/Button";
9 import Question from "../../components/Question";
10 import * as testPassingService from "../../services/testPassing";
11 import {ANSWER_TYPE} from "../../models/Test/config";
12
13 class Test extends React.Component {
14   constructor(props) {
15     super(props);
16
17     // get test id from url params
18     const { location } = props;
19     let query = new URLSearchParams(location.search);
20
21     const testId = parseInt(query.get("id"));
22
23     this.state = { testId };
24
25     this.handleSubmit = this.handleSubmit.bind(this);
26   }
27
28   async componentDidMount() {
29     const { setInitData } = this.props;
30     const { testId } = this.state;
31
32     let initState = null;
33     try {
34       initState = await testPassingService.init(testId);
35     } catch (err) {
36       // TODO: handle errors
37
38       console.error(err);
39     }
40
41     setInitData(initState);
42   }
43
44   async handleSubmit() {
45     const { questions, history } = this.props;
46     const { testId } = this.state;
47
48     let attemptId = -1;
49     try {
50       attemptId = await testPassingService.submit(questions, testId);
51     } catch (err) {
52       console.error(err);
53     }
54
55     history.push(`/test/result?id=${attemptId}`);
56   }
57
58   render() {
59     const { questions, updateAnswer } = this.props;
60
61     return (
62       <Container className="p-3">
63         {
64           questions.map((question, i) => {
65             const { title, typeAnswer, answers } = question;
66
67             return (
68               <Row key={i}>
69                 <Col lg={{span: 6, offset: 3}}
70                   style={{paddingBottom: '10px'}}>
71                   <Question title={title}
72                     type={typeAnswer}
73                     id={i}
74                     answers={answers}
75                     onAnswerChange={updateAnswer}/>
76                 </Col>
77               </Row>

```

```

78         });
79     });
80     }
81     <Row className="justify-content-center">
82         <Button variant="success"
83             size="lg"
84             onClick={this.handleSubmit}>
85             Submit
86         </Button>
87     </Row>
88 </Container>
89 );
90 }
91 }
92
93 Test.propTypes = {
94     questions: PropTypes.arrayOf(
95         PropTypes.exact({
96             title: PropTypes.string,
97             typeAnswer: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]),
98             answers: PropTypes.arrayOf(
99                 PropTypes.exact({
100                     content: PropTypes.string,
101                     isChecked: PropTypes.bool
102                 })
103             )
104         })
105     ).isRequired,
106     setInitData: PropTypes.func.isRequired,
107     updateAnswer: PropTypes.func.isRequired,
108     location: ReactRouterPropTypes.location.isRequired,
109     history: ReactRouterPropTypes.location.isRequired
110 };
111
112 export default withRouter(Test);

```

db-course-project-app/src/client/pages/TestResult/style.scss

1

db-course-project-app/src/client/pages/TestResult/index.js

```

1 import TestResult from "../TestResult";
2
3 export default TestResult;

```

db-course-project-app/src/client/pages/TestResult/TestResult.jsx

```

1 import * as React from "react";
2 import { withRouter } from "react-router-dom";
3 import ReactRouterPropTypes from "react-router-prop-types";
4 import Container from "react-bootstrap/Container";
5 import Table from "react-bootstrap/Table";
6 import ResultStatus from "../../components/ResultStatus";
7 import * as testResultService from "../../services/testResult";
8
9 class TestResult extends React.Component {
10     constructor(props) {
11         super(props);
12
13         // get attempt id from url params
14         const { location } = props;
15         let query = new URLSearchParams(location.search);
16
17         const attemptId = parseInt(query.get("id"));
18
19         this.state = {
20             attemptId,
21             userAnswers: []

```



```

22     };
23   }
24
25   async componentDidMount() {
26     const { attemptId } = this.state;
27
28     let userAnswers = [];
29     try {
30       userAnswers = await testResultService.init(attemptId);
31     } catch (err) {
32       console.log(err);
33     }
34
35     this.setState({ userAnswers });
36   }
37
38   render() {
39     const { userAnswers } = this.state;
40
41     return (
42       <Container className="p-3">
43         <Table responsive="lg">
44           <thead>
45             <tr>
46               <th>Question ID</th>
47               <th>Result</th>
48             </tr>
49           </thead>
50           <tbody>
51             {
52               userAnswers.length ? (
53                 userAnswers.map((answer, i) => {
54                   const { isCorrect } = answer;
55
56                   return (
57                     <tr key={i}>
58                       <td>{i + 1}</td>
59                       <td><ResultStatus isCorrect={isCorrect} /></td>
60                     </tr>
61                   );
62                 })
63               ) : (
64                 <tr>
65                   <td>None</td>
66                 </tr>
67               )
68             }
69           </tbody>
70         </Table>
71       </Container>
72     );
73   }
74 }
75
76 TestResult.propTypes = {
77   location: ReactRouterPropTypes.location
78 };
79
80 export default withRouter(TestResult);

```

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

```

1 .main-login-form {
2   &__forgot-password {
3     display: block;
4     margin: 10px 0;
5     text-align: center;
6   }
7
8   &__title {
9     text-align: center;
10  }
11
12  &__label {

```

```

13     font-weight: bold;
14 }
15 }

```

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

```

1  import * as React from "react";
2  import ReactRouterPropTypes from "react-router-prop-types";
3  import PropTypes from "prop-types";
4  import { connect } from "react-redux";
5
6  import Form from "react-bootstrap/Form";
7  import Button from "react-bootstrap/Button";
8  import Container from "react-bootstrap/Container";
9  import Col from "react-bootstrap/Col";
10 import ErrorFormAlert from "../../components/ErrorFormAlert";
11
12 import authService from "../../services/auth";
13 import * as authActions from "../../actions/auth";
14 import userConstraints from "../../models/User/constraints";
15
16 import "../style.scss";
17
18 const {
19     MIN_PASSWORD_LENGTH,
20     MAX_PASSWORD_LENGTH,
21     MIN_LOGIN_LENGTH,
22     MAX_LOGIN_LENGTH
23 } = userConstraints;
24
25 class Login extends React.Component {
26     constructor(props) {
27         super(props);
28
29         this.state = {
30             login: '',
31             password: '',
32             readMeChecked: false,
33
34             listErrors: [],
35
36             isLoading: false
37         }
38
39         this.handleChange = this.handleChange.bind(this);
40         this.handleCheckboxChange = this.handleCheckboxChange.bind(this);
41         this.handleSubmit = this.handleSubmit.bind(this);
42         this.hideErrorAlert = this.hideErrorAlert.bind(this);
43     }
44
45     handleChange(event) {
46         const { name, value } = event.target;
47
48         this.setState({ [name]: value });
49     }
50
51     handleCheckboxChange(event) {
52         const { name, checked } = event.target;
53
54         this.setState({ [name]: checked });
55     }
56
57     _generateFormData() {
58         const formData = new FormData();
59
60         formData.append('login', this.state.login);
61         formData.append('password', this.state.password);
62
63         return formData;
64     }
65
66     async handleSubmit(event) {
67         event.preventDefault();
68

```

```

69     const { history, dispatch } = this.props;
70     const formData = this._generateFormData();
71
72     this.toggleLoadingState();
73
74     try {
75         const { token, user } = await authService.signIn(formData);
76
77         localStorage.setItem('TOKEN', token);
78
79         dispatch(authActions.success(user));
80
81         history.push('/');
82     } catch ({ errors }) {
83         this.setState({ listErrors: errors });
84     }
85
86     this.toggleLoadingState();
87 }
88
89 hideErrorAlert() {
90     this.setState({
91         listErrors: []
92     });
93 }
94
95 toggleLoadingState() {
96     this.setState(prev => {
97         return {
98             isLoading: !prev.isLoading
99         }
100     });
101 }
102
103 render() {
104     const { listErrors, isLoading } = this.state;
105
106     return (
107         <Container className="p-3">
108             <Col lg={{offset: 3, span: 6}}>
109                 <h2 className="main-login-form__title">Login form</h2>
110                 <Form>
111                     <ErrorFormAlert listErrors={listErrors}
112                                     show={listErrors.length !== 0}
113                                     onHide={this.hideErrorAlert} />
114                     <Form.Group controlId="main-login-form__login">
115                         <Form.Control type="text"
116                                       placeholder="Enter login"
117                                       name="login"
118                                       minLength={MIN_LOGIN_LENGTH}
119                                       maxLength={MAX_LOGIN_LENGTH}
120                                       required
121                                       onChange={this.handleInputChange} />
122                     </Form.Group>
123                     <Form.Group controlId="main-login-form__password">
124                         <Form.Control type="password"
125                                       placeholder="Enter password"
126                                       name="password"
127                                       minLength={MIN_PASSWORD_LENGTH}
128                                       maxLength={MAX_PASSWORD_LENGTH}
129                                       required
130                                       onChange={this.handleInputChange}/>
131                     </Form.Group>
132                     <Button variant="primary"
133                           type="submit"
134                           block
135                           disabled={isLoading}
136                           onClick={this.handleFormSubmit}>
137                         { isLoading ? 'Loading...' : 'Submit' }
138                     </Button>
139                 </Form>
140             </Col>
141         </Container>
142     );
143 }
144 }

```

```

145
146 Login.propTypes = {
147   location: ReactRouterPropTypes.location,
148   history: ReactRouterPropTypes.history,
149   dispatch: PropTypes.func
150 };
151
152 const connectedLogin = connect()(Login);
153
154 export { connectedLogin as Login };

```

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

```

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

```

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

```

1 describe("Sum", () => {
2   it("addition", () => {
3     expect(6).toBe(6);
4   });
5 });

```

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

```

1 import authService from "../../services/auth";
2 import {OK} from "http-status-codes";
3
4 describe("client sign up", () => {
5   it("doesn't sign up when error list are", async () => {
6     const formErrorList = {
7       errors: [
8         { message: 'test' }
9       ]
10    };
11
12    fetch.mockReject(() => Promise.reject(formErrorList));
13
14    await expect(authService.signUp()).rejects.toEqual(formErrorList);
15  });
16
17   it("sign up when API responds with status OK", async () => {
18     fetch.mockResponse([
19       null,
20       { status: OK }
21     ]);
22
23     await expect(authService.signUp()).resolves.toEqual(undefined);
24   });
25 });
26
27 describe("client sign in", () => {
28   it("doesn't sign in when error list are", async () => {
29     const formErrorList = {
30       errors: [
31         { message: 'test' }
32       ]
33    };
34
35    fetch.mockReject(() => Promise.reject(formErrorList));
36
37    await expect(authService.signIn()).rejects.toEqual(formErrorList);
38  });
39
40   // it("sign in when API responds with status OK", async () => {
41   //   const respond = { test: 'test' };
42   //

```

```

43     //     fetch.mockResponse([
44     //         JSON.stringify(respond),
45     //         { status: OK }
46     //     ]);
47     //
48     //     await expect(authService.signIn()).resolves.toEqual(respond);
49     // });
50 });

```

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

```

1  import { createStore } from "redux";
2  import rootReducer from "../reducers";
3  import * as authActions from "../actions/auth";
4  import {getToken} from "../helpers/token";
5  import {appendAuth} from "../helpers/header";
6  import history from "../history";
7
8  const store = createStore(
9      rootReducer,
10     window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__()
11 );
12
13 export const initAuthStore = async (store) => {
14     const token = getToken();
15
16     if (!token) {
17         store.dispatch(authActions.failed());
18     } else {
19         let response = null;
20         try {
21             const headers = new Headers();
22
23             appendAuth(headers, token);
24
25             response = await fetch('/api/init', { headers, method: 'POST' });
26         } catch (err) {
27             store.dispatch(authActions.logout());
28         }
29
30         if (response.ok) {
31             const responseJson = await response.json();
32             const { user } = responseJson;
33
34             store.dispatch(authActions.success(user));
35         } else {
36             store.dispatch(authActions.logout());
37
38             history.push('/login');
39         }
40     }
41 };
42
43 export default store;

```

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

```

1  import store, { initAuthStore } from "./store";
2
3  export { store, initAuthStore };

```

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

```

1  /**
2   * Add to HTTP header 'Authorization' field
3   * @param {Headers} headers
4   * @param {string} token
5   * @return {Headers}
6   */

```

```

7 export function appendAuth(headers, token) {
8   headers.append('Authorization', token);
9
10  return headers;
11 }
12
13 /**
14  * Add to HTTP headers for working with JSON
15  * @param {Headers} headers
16  * @return {Headers}
17  */
18 export function appendJSON(headers) {
19   headers.append('Accept', 'application/json');
20   headers.append('Content-Type', 'application/json');
21
22   return headers;
23 }

```

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

```

1 import {ANSWER_TYPE} from "../../models/Test/config";
2
3 /**
4  * Create answer
5  * @param {string} content - Answer text
6  * @param {boolean} isRight - Result
7  * @return {{isRight: boolean, content: string}}
8  */
9 export const createAnswer = (content = '', isRight = false) => {
10   return {content, isRight};
11 };
12
13 /**
14  * Create question
15  * @param {string} title - Question title
16  * @param {string} typeAnswer - Answer type
17  * @param {Array<{content: string, isRight: boolean}>} answers
18  * @return {{answers: {isRight: boolean, content: string}[], title: string, typeAnswer: string}}
19  */
20 export const createQuestion = (title = '',
21                                typeAnswer = ANSWER_TYPE.ONE,
22                                answers = [createAnswer(), createAnswer()]) => {
23   return {title, typeAnswer, answers};
24 };

```

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

```

1 /**
2  * Toggle state of app loader
3  */
4 export function toggleLoader() {
5   const loader = document.querySelector('.loader');
6
7   loader.classList.toggle('loader_hide');
8 }

```

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

```

1 const TOKEN_ID = 'TOKEN';
2
3 /**
4  * Get JWT
5  * @return {string}
6  */
7 export function getToken() {
8   return localStorage.getItem(TOKEN_ID);
9 }
10
11 /**

```

```

12  * Remove JWT from localStorage
13  */
14  export function removeToken() {
15      localStorage.removeItem(TOKEN_ID);
16  }

```

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 import Test from "../../containers/Test";
13 import TestResult from "../../pages/TestResult";
14
15 import Footer from "../../components/Footer";
16 import Header from "./components/Header";
17 import Profile from "./pages/Profile";
18 import HttpErrorInfo from "../../components/HttpErrorInfo";
19 import {NOT_FOUND} from "http-status-codes";
20 import TestStatistic from "./pages/TestStatistic";
21
22 class App extends React.Component {
23     render() {
24         return (
25             <div>
26                 <Header />
27
28                 <Switch>
29                     <Route exact path="/" component={Home}/>
30                     <NotIsLoggedInRoute path="/login" component={Login}/>
31                     <NotIsLoggedInRoute path="/signup" component={SignUp}/>
32                     <PrivateRoute path="/profile" component={Profile}/>
33                     <PrivateRoute exact path="/test/edit" component={TestEditor} />
34                     <PrivateRoute exact path="/test/result" component={TestResult} />
35                     <PrivateRoute exact path="/test/pass" component={Test} />
36                     <PrivateRoute exact path="/test/statistic" component={TestStatistic} />
37                     <Route component={() => <HttpErrorInfo status={NOT_FOUND} />} />
38                 </Switch>
39
40                 <Container className="p-3">
41                     <Footer/>
42                 </Container>
43             </div>
44         );
45     }
46 }
47
48 export default App;

```

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

1

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

```

1  import * as React from "react";
2  import Container from "react-bootstrap/Container";
3  import Row from "react-bootstrap/Row";
4  import Col from "react-bootstrap/Col";
5  import ErrorFormAlert from "../../components/ErrorFormAlert";
6  import UpdatePasswordForm from "./components/UpdatePasswordForm";

```

```

7 import DeleteProfileForm from "../components/DeleteProfileForm";
8
9 import "../style.scss";
10
11 class ProfileSettings extends React.Component {
12     constructor(props) {
13         super(props);
14
15         this.state = {
16             listErrors: []
17         }
18
19         this.updateListErrors = this.updateListErrors.bind(this);
20         this.hideErrorAlert = this.hideErrorAlert.bind(this);
21     }
22
23     updateListErrors(errors) {
24         this.setState({ listErrors: errors })
25     }
26
27     hideErrorAlert() {
28         this.setState({ listErrors: [] });
29     }
30
31     render() {
32         const { listErrors } = this.state;
33
34         return (
35             <Container className="p-3">
36                 <Row>
37                     <ErrorFormAlert listErrors={listErrors}
38                                     show={listErrors.length !== 0}
39                                     onHide={this.hideErrorAlert} />
40                 </Row>
41                 <Row>
42                     <Col lg={6}>
43                         <UpdatePasswordForm onSubmitError={this.updateListErrors} />
44                     </Col>
45                 </Row>
46                 <Row>
47                     <Col lg={6}>
48                         <DeleteProfileForm onSubmitError={this.updateListErrors} />
49                     </Col>
50                 </Row>
51             </Container>
52         );
53     }
54 }
55
56 export default ProfileSettings;

```

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

```

1 import ProfileSettings from "../ProfileSettings";
2
3 export default ProfileSettings;

```

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

```

1 import * as React from "react";
2 import { withRouter } from "react-router-dom";
3 import { connect } from "react-redux";
4 import PropTypes from "prop-types";
5 import ReactRouterPropTypes from "react-router-prop-types";
6 import Button from "react-bootstrap/Button";
7 import Modal from "react-bootstrap/Modal";
8 import * as authActions from "../../../../../actions/auth";
9 import * as editProfileSettings from "../../../../../services/editProfileSettings";
10
11 class DeleteProfileForm extends React.Component {
12     constructor(props) {
13         super(props);

```



```

14
15     this.state = {
16       modalShow: false
17     }
18
19     this.handleSubmit = this.handleSubmit.bind(this);
20     this.toggleWarningModal = this.toggleWarningModal.bind(this);
21   }
22
23   toggleWarningModal() {
24     this.setState(prev => {
25       return {
26         modalShow: !prev.modalShow
27       };
28     });
29   }
30
31   async handleSubmit() {
32     const { history, dispatch, onSubmitError } = this.props;
33
34     try {
35       await editProfileSettings.remove();
36
37       dispatch(authActions.logout());
38
39       history.push('/');
40     } catch ({ errors }) {
41       onSubmitError(errors);
42     }
43   }
44
45   render() {
46     const { modalShow } = this.state;
47
48     return (
49       <>
50         <h5>Danger zone:</h5>
51         <Button variant="danger"
52           onClick={this.toggleWarningModal}>Delete profile</Button>
53
54         <Modal
55           size="md"
56           aria-labelledby="contained-modal-title-vcenter"
57           centered
58           onHide={this.toggleWarningModal}
59           show={modalShow}
60         >
61           <Modal.Header closeButton>
62             <Modal.Title id="contained-modal-title-vcenter">Delete profile</Modal.Title>
63           </Modal.Header>
64           <Modal.Body>
65             <p>Are you sure you want to delete profile?</p>
66           </Modal.Body>
67           <Modal.Footer>
68             <Button onClick={this.handleSubmit}>Yes</Button>
69           </Modal.Footer>
70         </Modal>
71       </>
72     );
73   }
74 }
75
76 DeleteProfileForm.propTypes = {
77   onSubmitError: PropTypes.func,
78   history: ReactRouterPropTypes.history,
79   dispatch: PropTypes.func
80 };
81
82 const connectedDeleteProfileForm = connect()(DeleteProfileForm);
83
84 export default withRouter(connectedDeleteProfileForm);

```

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

```

1 import DeleteProfileForm from "../DeleteProfileForm";
2
3 export default DeleteProfileForm;

```

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

```

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

```

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

```

1 import * as React from "react";
2 import { connect } from "react-redux";
3 import { withRouter } from "react-router-dom";
4 import ReactRouterPropTypes from "react-router-prop-types";
5 import PropTypes from "prop-types";
6 import Form from "react-bootstrap/Form";
7 import Button from "react-bootstrap/Button";
8 import * as authActions from "../../../../../actions/auth";
9 import * as editProfileSettings from "../../../../../services/editProfileSettings";
10
11 class UpdatePasswordForm extends React.Component {
12   constructor(props) {
13     super(props);
14
15     this.state = {
16       password: '',
17       newPassword: '',
18       repeatNewPassword: '',
19
20       isLoading: false
21     };
22
23     this.handleChange = this.handleChange.bind(this);
24     this.handleSubmit = this.handleSubmit.bind(this);
25     this._generateFormData = this._generateFormData.bind(this);
26     this.toggleLoadingState = this.toggleLoadingState.bind(this);
27   }
28
29   handleChange(event) {
30     const { name, value } = event.target;
31
32     this.setState({ [name]: value });
33   }
34
35   _generateFormData() {
36     const formData = new FormData();
37     formData.append('password', this.state.password);
38     formData.append('newPassword', this.state.newPassword);
39     formData.append('repeatNewPassword', this.state.repeatNewPassword);
40
41     return formData;
42   }
43
44   async handleSubmit(event) {
45     event.preventDefault();
46
47     const { dispatch, history, onSubmitError } = this.props;
48
49     this.toggleLoadingState();
50
51     try {
52       const formData = this._generateFormData();
53       await editProfileSettings.updatePassword(formData);
54
55       dispatch(authActions.logout());
56
57       history.push("/login");
58     } catch ({ errors }) {
59       this.toggleLoadingState();

```

```

60
61     onSubmitError(errors);
62   }
63 }
64
65 toggleLoadingState() {
66   this.setState(prev => {
67     return { isLoading: !prev.isLoading }
68   });
69 }
70
71 render() {
72   const { isLoading } = this.state;
73
74   return (
75     <
76       <h5>Update password:</h5>
77       <Form>
78         <Form.Group controlId="profile-settings-form__password">
79           <Form.Control type="password"
80             placeholder="Enter current password"
81             name="password"
82             onChange={this.handleInputChange} />
83         </Form.Group>
84         <Form.Group controlId="profile-settings-form__repassword">
85           <Form.Control type="password"
86             placeholder="Enter new password"
87             name="newPassword"
88             onChange={this.handleInputChange} />
89         </Form.Group>
90         <Form.Group controlId="profile-settings-form__repassword">
91           <Form.Control type="password"
92             placeholder="Enter new password again"
93             name="repeatNewPassword"
94             onChange={this.handleInputChange} />
95         </Form.Group>
96         <Button variant="primary"
97           disabled={isLoading}
98           onClick={this.handleFormSubmit}>{ isLoading ? 'Loading...' : 'Save' }</Button>
99       </Form>
100     </>
101   );
102 }
103 }
104
105 UpdatePasswordForm.propTypes = {
106   onSubmitError: PropTypes.func.isRequired,
107   dispatch: PropTypes.func,
108   history: ReactRouterPropTypes.history
109 };
110
111 const connectedUpdatePasswordForm = connect()(UpdatePasswordForm);
112 const connectedUpdatePasswordFormWithRouter = withRouter(connectedUpdatePasswordForm);
113
114 export { connectedUpdatePasswordFormWithRouter as UpdatePasswordForm };

```

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

1

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

```

1 import * as React from "react";
2 import Container from "react-bootstrap/Container";
3 import ListTestCards from "../../../components/ListTestCards";
4 import * as editTest from "../../../services/editTest";
5 import HttpErrorInfo from "../../../components/HttpErrorInfo";
6 import {NO_CONTENT} from "http-status-codes";
7 import Form from "react-bootstrap/Form";
8 import InputGroup from "react-bootstrap/InputGroup";
9 import FormControl from "react-bootstrap/FormControl";

```

```

10 import Button from "react-bootstrap/Button";
11 import Row from "react-bootstrap/Row";
12 import Col from "react-bootstrap/Col";
13 import TagList from "../../components/TagList";
14
15 import "../style.scss";
16
17 class ProfileTests extends React.Component {
18   constructor(props) {
19     super(props);
20
21     this.state = {
22       profileTests: [],
23       searchTitle: '',
24       searchTagValue: '',
25       searchTags: []
26     };
27
28     this.handleDeleteTestCard = this.handleDeleteTestCard.bind(this);
29     this.handleSearchTitleChange = this.handleSearchTitleChange.bind(this);
30     this.handleSearchTagValueChange = this.handleSearchTagValueChange.bind(this);
31     this.handleAddingTag = this.handleAddingTag.bind(this);
32     this.handleSearchTagDeleting = this.handleSearchTagDeleting.bind(this);
33     this.searchFilter = this.searchFilter.bind(this);
34   }
35
36   async handleDeleteTestCard(testId) {
37     await editTest.deleteTest(testId);
38
39     // delete test card from list
40     this.setState(prev => {
41       const { profileTests } = prev;
42
43       const delI = profileTests.map(test => test.testId)
44         .indexOf(testId);
45
46       return {
47         profileTests: [
48           ...profileTests.slice(0, delI),
49           ...profileTests.slice(delI + 1),
50         ]
51       };
52     });
53   }
54
55   handleSearchTagDeleting(tagId) {
56     this.setState(prev => {
57       const { searchTags } = prev;
58
59       return {
60         searchTags: [
61           ...searchTags.slice(0, tagId),
62           ...searchTags.slice(tagId + 1),
63         ]
64       };
65     });
66   }
67
68   handleSearchTitleChange({ target }) {
69     const { value } = target;
70
71     this.setState({ searchTitle: value });
72   }
73
74   handleSearchTagValueChange({ target }) {
75     const { value } = target;
76
77     this.setState({ searchTagValue: value })
78   }
79
80   handleAddingTag() {
81     const { searchTagValue, searchTags } = this.state;
82     const tagIsNotIncluded = !searchTags.includes(searchTagValue);
83
84     if (searchTagValue && tagIsNotIncluded) {
85       this.setState(prev => {

```

```

86         return {
87             searchTags: [...prev.searchTags, searchTagValue]
88         };
89     });
90 }
91 }
92
93 async componentDidMount() {
94     const responseJson = await editTest.getOwnTests();
95
96     this.setState({ profileTests: responseJson });
97 }
98
99 searchFilter() {
100     const { profileTests, searchTags, searchTitle } = this.state;
101
102     return profileTests.filter(({ tags, title }) => {
103         const containAllTags = tags.some(tag => searchTags.includes(tag));
104         const lowerTitle = title.toLowerCase();
105         const lowerSearchTitle = searchTitle.toLowerCase();
106         const likeTitle = lowerTitle.indexOf(lowerSearchTitle) !== -1;
107
108         return (searchTags.length > 0 ? containAllTags : true) && (searchTitle ? likeTitle : true);
109     });
110 }
111
112 render() {
113     const { searchTags } = this.state;
114     const profileTests = this.searchFilter();
115
116     return (
117         <Container className="p-3">
118             <Row>
119                 <Col lg={12} style={{ padding: '0' }}>
120                     <Form>
121                         <Form.Group controlId="">
122                             <FormControl
123                                 aria-label="test title"
124                                 aria-describedby="basic-addon2"
125                                 required
126                                 placeholder="Enter test title"
127                                 onChange={this.handleSearchTitleChange}
128                             />
129                         </Form.Group>
130                         <Form.Group controlId="">
131                             <InputGroup className="mb-3">
132                                 <FormControl
133                                     aria-label="Recipient's username"
134                                     aria-describedby="basic-addon2"
135                                     required
136                                     placeholder="Enter tag name"
137                                     onChange={this.handleSearchTagValueChange}
138                                 />
139                                 <InputGroup.Append>
140                                     <Button variant="primary"
141                                         onClick={this.handleAddingTag}>
142                                         Add
143                                     </Button>
144                                 </InputGroup.Append>
145                             </InputGroup>
146                         </Form.Group>
147                         <TagList tags={searchTags}
148                             canDelete={true}
149                             deleteTag={this.handleSearchTagDeleting} />
150                     </Form>
151                 </Col>
152             </Row>
153             {
154                 profileTests.length ? (
155                     <ListTestCards tests={profileTests}
156                         onDeleteTestCard={this.handleDeleteTestCard}/>
157                 ) : (
158                     <HttpErrorInfo status={NO_CONTENT}
159                         reason="You don't have tests." />
160                 )
161             }
162         </Container>
163     );
164 }

```

```

162         </Container>
163     );
164 }
165 }
166
167 export default ProfileTests;

```

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

```

1 import ProfileTests from "../ProfileTests";
2
3 export default ProfileTests;

```

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

```

1 .test-editor {
2     &__submit-section {
3         text-align: center;
4     }
5
6     &__submit-section-row {
7         margin: 10px 0;
8     }
9
10    &__textarea {
11        resize: none;
12    }
13 }

```

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

```

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

```

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

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import ReactRouterPropTypes from "react-router-prop-types";
4 import { withRouter } from "react-router-dom";
5 import { connect } from "react-redux";
6 import * as testEditorActions from "../../../../../actions/testEditor";
7
8 import Container from "react-bootstrap/Container";
9 import Row from "react-bootstrap/Row";
10 import Col from "react-bootstrap/Col";
11 import Form from "react-bootstrap/Form";
12 import InputGroup from "react-bootstrap/InputGroup";
13 import FormControl from "react-bootstrap/FormControl";
14 import Button from "react-bootstrap/Button";
15 import TestEditorTagList from "../../../../containers/TestEditorTagList";
16 import TestEditorQuestionList from "../../../../containers/TestEditorQuestionList";
17 import ErrorFormAlert from "../../../../components/ErrorFormAlert";
18 import {ANSWER_TYPE} from "../../../../models/Test/config";
19 import * as editTest from "../../../../../services/editTest";
20
21 import "../style.scss";
22
23 class TestEditor extends React.Component {
24     constructor(props) {
25         super(props);
26
27         const { location } = props;
28         let query = new URLSearchParams(location.search);
29
30         const testId = parseInt(query.get("id"));

```

```

31
32     const isEditing = !!testId;
33
34     this.state = {
35         isEditing,
36         testId: testId ? testId : -1,
37         tagValue: '',
38         listErrors: [],
39         isLoading: false
40     };
41
42     this.handleTitleChange = this.handleTitleChange.bind(this);
43     this.handleDescriptionChange = this.handleDescriptionChange.bind(this);
44     this.handleAppendTag = this.handleAppendTag.bind(this);
45     this.handleTagInputChange = this.handleTagInputChange.bind(this);
46     this.handleAppendQuestion = this.handleAppendQuestion.bind(this);
47     this.hideErrorAlert = this.hideErrorAlert.bind(this);
48     this.handleFormSubmit = this.handleFormSubmit.bind(this);
49     this.toggleLoadingState = this.toggleLoadingState.bind(this);
50 }
51
52 async componentDidMount() {
53     const { dispatch } = this.props;
54     const { isEditing, testId } = this.state;
55
56     if (isEditing) {
57         let response = null;
58         try {
59             response = await editTest.getTestForEdit(testId);
60
61             dispatch(testEditorActions.update(response));
62         } catch (err) {
63             console.error(err);
64         }
65     } else {
66         dispatch(testEditorActions.reset());
67     }
68 }
69
70 toggleLoadingState() {
71     this.setState(prev => {
72         return {
73             isLoading: !prev.isLoading
74         }
75     });
76 }
77
78 handleTitleChange({ target: { value } }) {
79     const { dispatch } = this.props;
80
81     dispatch(testEditorActions.updateTitle(value));
82 }
83
84 handleDescriptionChange({ target: { value } }) {
85     const { dispatch } = this.props;
86
87     dispatch(testEditorActions.updateDescription(value));
88 }
89
90 handleTagInputChange({ target: { value } }) {
91     this.setState({ tagValue: value });
92 }
93
94 handleAppendTag() {
95     const { tagValue } = this.state;
96     const { dispatch } = this.props;
97
98     dispatch(testEditorActions.appendTag(tagValue));
99 }
100
101 handleAppendQuestion(event) {
102     event.preventDefault();
103
104     const { dispatch } = this.props;
105
106     dispatch(testEditorActions.appendQuestion());

```

```

107     }
108
109     hideErrorAlert() {
110         this.setState({ listErrors: [] });
111     }
112
113     async handleFormSubmit(event) {
114         event.preventDefault();
115
116         const { history, testEditor } = this.props;
117         const { isEditing, testId } = this.state;
118
119         this.toggleLoadingState();
120
121         try {
122             if (!isEditing) {
123                 await editTest.create(testEditor);
124             } else {
125                 await editTest.update(testEditor, testId);
126             }
127
128             history.push('/profile/tests');
129         } catch ({ errors }) {
130             this.toggleLoadingState();
131
132             this.setState({
133                 listErrors: errors
134             });
135         }
136     }
137
138     render() {
139         const { listErrors, isLoading } = this.state;
140         const { testEditor: { info } } = this.props;
141
142         return (
143             <Container className="p-3">
144                 <Row>
145                     <Col lg={{offset: 2, span: 8}}>
146                         <ErrorFormAlert listErrors={listErrors}
147                             show={listErrors.length !== 0}
148                             onHide={this.hideErrorAlert} />
149                     <Form>
150                         <Row>
151                             <Col lg={4}>
152                                 <Form.Label>Title:</Form.Label>
153                             </Col>
154                             <Col lg={8}>
155                                 <Form.Group controlId="">
156                                     <Form.Control required
157                                         value={info.title}
158                                         onChange={this.handleTitleChange} />
159                                 </Form.Group>
160                             </Col>
161                         </Row>
162                         <Row>
163                             <Col lg={4}>
164                                 <Form.Label>Description:</Form.Label>
165                             </Col>
166                             <Col lg={8}>
167                                 <Form.Group controlId="">
168                                     <Form.Control className="test-editor__textarea"
169                                         as="textarea"
170                                         rows={3}
171                                         value={info.description}
172                                         required
173                                         onChange={this.handleDescriptionChange} />
174                                 </Form.Group>
175                             </Col>
176                         </Row>
177                         <Row>
178                             <Col lg={4}>
179                                 <Form.Label>Tags:</Form.Label>
180                             </Col>
181                             <Col lg={8}>
182                                 <Form.Group controlId="">

```



```

183         <InputGroup className="mb-3">
184             <FormControl
185                 aria-label="Recipient's username"
186                 aria-describedby="basic-addon2"
187                 required
188                 onChange={this.handleTagInputChange}
189             />
190             <InputGroup.Append>
191                 <Button variant="primary"
192                     onClick={this.handleAppendTag}>
193                     Add
194                 </Button>
195             </InputGroup.Append>
196         </InputGroup>
197     </Form.Group>
198     <TestEditorTagList />
199 </Col>
200 </Row>
201
202 <hr/>
203
204 <TestEditorQuestionList />
205
206 <Row>
207     <Col lg={12}>
208         <div className="test-editor__submit-section">
209             <div className="test-editor__submit-section-row">
210                 <Button className="test-editor__submit-btn"
211                     type="primary"
212                     size="lg"
213                     onClick={this.handleAppendQuestion}>
214                     Add question
215                 </Button>
216             </div>
217             <div className="test-editor__submit-section-row">
218                 <Button className="test-editor__submit-btn"
219                     type="primary"
220                     size="lg"
221                     disabled={isLoading}
222                     onClick={this.handleFormSubmit}>
223                     {isLoading ? 'Loading...' : 'Publish'}
224                 </Button>
225             </div>
226         </div>
227     </Col>
228 </Row>
229 </Form>
230 </Col>
231 </Row>
232 </Container>
233 );
234 }
235 }
236
237 TestEditor.propTypes = {
238     dispatch: PropTypes.func,
239     history: ReactRouterPropTypes.history,
240     location: ReactRouterPropTypes.location,
241     testEditor: PropTypes.exact({
242         info: PropTypes.exact({
243             title: PropTypes.string,
244             description: PropTypes.string,
245             tags: PropTypes.arrayOf(PropTypes.string)
246         }),
247         questions: PropTypes.arrayOf(
248             PropTypes.exact({
249                 title: PropTypes.string,
250                 typeAnswer: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]),
251                 answers: PropTypes.arrayOf(
252                     PropTypes.exact({
253                         content: PropTypes.string,
254                         isRight: PropTypes.bool
255                     })
256                 )
257             })
258         )
259     })

```

```

259     }).isRequired
260   };
261
262   function mapStateToProps(state) {
263     const { testEditor } = state;
264
265     return { testEditor };
266   }
267
268   const connectedTestEditor = connect(mapStateToProps)(withRouter(TestEditor));
269
270   export { connectedTestEditor as TestEditor };

```

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

```

1 .profile {
2   &__username {
3     font-weight: bold;
4     text-align: center;
5   }
6
7   &__avatar {
8     border: 6px solid #f0f8ff;
9     border-radius: 100%;
10    display: block;
11    margin: 0 auto;
12    width: 70%;
13  }
14 }

```

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

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import { connect } from "react-redux";
4 import {NOT_FOUND} from "http-status-codes";
5
6 import { Switch, Route } from 'react-router-dom';
7 import { LinkContainer } from "react-router-bootstrap";
8 import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
9 import { faList, faUserCog, faPoll } from "@fortawesome/free-solid-svg-icons";
10 import Container from "react-bootstrap/Container";
11 import Col from "react-bootstrap/Col";
12 import Row from "react-bootstrap/Row";
13 import Nav from "react-bootstrap/Nav";
14 import ProfileSettings from "../ProfileSettings";
15 import ProfileAttempts from "../ProfileAttempts";
16 import HttpErrorInfo from "../../../../../components/HttpErrorInfo";
17 import ProfileTests from "../ProfileTests";
18
19 import "../style.scss";
20
21 const Profile = ({ user }) => {
22   return (
23     <Container className="p-3">
24       <Row>
25         <Col lg={3}>
26           
28           <p className="profile__username">{ user.login }</p>
29         </Col>
30         <Col lg={9}>
31           <Nav variant="tabs" defaultActiveKey="tests">
32             <Nav.Item>
33               <LinkContainer to="/profile/tests">
34                 <Nav.Link eventKey="tests">
35                   <FontAwesomeIcon icon={faList} /> Tests
36                 </Nav.Link>
37               </LinkContainer>
38             </Nav.Item>
39             <Nav.Item>
40               <LinkContainer to="/profile/attempts">

```

```

41         <Nav.Link eventKey="attempts">
42             <FontAwesomeIcon icon={faPoll} /> Attempts
43         </Nav.Link>
44     </LinkContainer>
45 </Nav.Item>
46 <Nav.Item>
47     <LinkContainer to="/profile/settings">
48         <Nav.Link eventKey="settings">
49             <FontAwesomeIcon icon={faUserCog} /> Settings
50         </Nav.Link>
51     </LinkContainer>
52 </Nav.Item>
53 </Nav>
54
55 <Switch>
56     <Route exact path="/profile" render={() => <ProfileTests />} />
57     <Route path="/profile/tests" render={() => <ProfileTests />} />
58     <Route path="/profile/attempts" component={ProfileAttempts} />
59     <Route path="/profile/settings" component={ProfileSettings} />
60     <Route component={() => <HttpErrorInfo status={NOT_FOUND} />} />
61 </Switch>
62 </Col>
63 </Row>
64 </Container>
65 );
66 }
67
68 Profile.propTypes = {
69     user: PropTypes.shape({
70         login: PropTypes.string
71     })
72 };
73
74 function mapStateToProps(state) {
75     const { user } = state.auth;
76
77     return { user };
78 }
79
80 const connectedProfile = connect(mapStateToProps)(Profile);
81
82 export { connectedProfile as Profile };

```

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

```

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

```

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

```

1

```

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

```

1 import * as React from "react";
2 import { Link } from "react-router-dom";
3 import Container from "react-bootstrap/Container";
4 import Table from "react-bootstrap/Table";
5 import * as attemptsService from "../../../services/attempts";
6 import {NO_CONTENT, OK} from "http-status-codes";
7
8 import "../style.scss";
9 import HttpErrorInfo from "../../../components/HttpErrorInfo";
10
11 class ProfileAttempts extends React.Component {
12     constructor(props) {
13         super(props);
14

```

```

15     this.state = {
16         attempts: [],
17         requestIsFailed: false,
18         failedResponse: {
19             status: OK,
20             message: ''
21         }
22     };
23 }
24
25 async componentDidMount() {
26     try {
27         const attempts = await attemptsService.getOwnAttempts();
28
29         this.setState({ attempts });
30     } catch (err) {
31         const { status, message } = err;
32
33         this.setState({
34             requestIsFailed: true,
35             failedResponse: {
36                 status,
37                 message
38             }
39         });
40     }
41 }
42
43 render() {
44     const { attempts, failedResponse, requestIsFailed } = this.state;
45
46     return (
47         <Container style={{ padding: '15px 0' }}>
48             {
49                 requestIsFailed ? (
50                     <HttpErrorInfo status={failedResponse.status}
51                         reason={failedResponse.message} />
52                 ) : (
53                     attempts.length ? (
54                         <Table responsive="lg">
55                             <thead>
56                                 <tr>
57                                     <th>Test</th>
58                                     <th>Result</th>
59                                     <th>Date</th>
60                                 </tr>
61                             </thead>
62                             <tbody>
63                                 {
64                                     attempts.map((attempt, i) => {
65                                         const { title, testId, result, date } = attempt;
66                                         const link = `/test/pass?id=${testId}`;
67
68                                         return (
69                                             <tr key={i}>
70                                                 <td>
71                                                     <Link to={link}>{title}</Link>
72                                                 </td>
73                                                 <td>{result * 100}%</td>
74                                                 <td>
75                                                     <time dateTime={date}>{date}</time>
76                                                 </td>
77                                             </tr>
78                                         );
79                                     })
80                                 }
81                             </tbody>
82                         </Table>
83                     ): (
84                         <HttpErrorInfo status={NO_CONTENT}
85                             reason={"You don't have any attempts."} />
86                     )
87                 )
88             }
89         </Container>
90     );

```

```

91     }
92 }
93
94 export default ProfileAttempts;

```

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

```

1 import ProfileAttempts from "../ProfileAttempts";
2
3 export default ProfileAttempts;

```

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

```

1

```

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

```

1 import Home from "../Home";
2
3 export default Home;

```

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

```

1 import * as React from "react";
2
3 import Jumbotron from "react-bootstrap/Jumbotron";
4 import Container from "react-bootstrap/Container";
5 import Row from "react-bootstrap/Row";
6 import Col from "react-bootstrap/Col";
7
8 import "../style.scss";
9
10 const Home = () => {
11     return (
12         <Container className="p-3">
13             <Jumbotron>
14                 <h1>Hello!</h1>
15                 <p>
16                     This is a simple hero unit, a simple jumbotron-style component for calling
17                     extra attention to featured content or information.
18                 </p>
19             </Jumbotron>
20             <Row>
21                 <Col lg={4}>
22                     <h2>Create</h2>
23                     <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab blanditiis delectus
24                     ↳ dolorem eius expedita nemo neque pariatur perferendis quasi! Accusantium ad atque
25                     ↳ corporis deleniti eius, eligendi qui rem sunt vitae!</p>
26                 </Col>
27                 <Col lg={4}>
28                     <h2>Share</h2>
29                     <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab blanditiis delectus
30                     ↳ dolorem eius expedita nemo neque pariatur perferendis quasi! Accusantium ad atque
31                     ↳ corporis deleniti eius, eligendi qui rem sunt vitae!</p>
32                 </Col>
33             </Row>
34         </Container>
35     );
36 }
37
38 export default Home;

```

...-project-app/src/client/apps/main/pages/TestStatistic/TestStatistic.jsx

```
1 import * as React from "react";
2 import Container from "react-bootstrap/Container";
3 import TableStatistic from "../../components/TableStatistic";
4 import { withRouter } from "react-router-dom";
5 import * as attemptService from "../../../services/attempt";
6 import ReactRouterPropTypes from "react-router-prop-types";
7
8 class TestStatistic extends React.Component {
9   constructor(props) {
10     super(props);
11
12     // get attempt id from url params
13     const { location } = props;
14     let query = new URLSearchParams(location.search);
15
16     const testId = parseInt(query.get("id"));
17
18     this.state = {
19       data: [],
20       testId
21     };
22   }
23
24   async componentDidMount() {
25     const { testId } = this.state;
26
27     try {
28       const data = await attemptService.getOwnTestAttempts(testId);
29
30       this.setState({ data });
31     } catch (err) {
32       console.error(err);
33     }
34   }
35
36   render() {
37     const { data } = this.state;
38
39     return (
40       <Container className="p-3">
41         <TableStatistic rows={data} />
42       </Container>
43     );
44   }
45 }
46
47 TestStatistic.propTypes = {
48   location: ReactRouterPropTypes.location
49 };
50
51 export default withRouter(TestStatistic);
```

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

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

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

```
1 //header {
2 //  @_user-dropdown-menu {
3 //    float: right;
4 //  }
5 //}
```

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import { withRouter } from "react-router-dom";
4 import ReactRouterPropTypes from "react-router-prop-types";
5 import { connect } from "react-redux";
6 import * as authActions from "../../../../../actions/auth";
7
8 import {
9   faClipboardList,
10   faUser,
11   faSignOutAlt,
12   faSignInAlt,
13   faIdCard,
14   faUserPlus,
15   faPlus
16 } from "@fortawesome/free-solid-svg-icons";
17 import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
18 import Navbar from "react-bootstrap/Navbar";
19 import Nav from "react-bootstrap/Nav";
20 import NavDropdown from "react-bootstrap/NavDropdown";
21 import Modal from "react-bootstrap/Modal";
22 import Button from "react-bootstrap/Button";
23 import { LinkContainer } from "react-router-bootstrap";
24
25 import "../style.scss";
26
27 const Header = ({ isLoggedIn, user, dispatch, history }) => {
28   const [modalShow, setModalShow] = React.useState(false);
29
30   const showModal = () => setModalShow(true);
31
32   const hideModal = () => setModalShow(false);
33
34   const onLogOut = () => {
35     dispatch(authActions.logOut());
36
37     setModalShow(false);
38
39     history.push("/");
40   };
41
42   return (
43     <Navbar bg="dark" variant="dark">
44       <LinkContainer to="/">
45         <Navbar.Brand>
46           <FontAwesomeIcon icon={faClipboardList} /> PassQuiz
47         </Navbar.Brand>
48       </LinkContainer>
49       <Nav className="mr-auto">
50         { !isLoggedIn ? (
51           <>
52             <LinkContainer to="/signup">
53               <Nav.Link> <FontAwesomeIcon icon={faUserPlus} /> Sign Up</Nav.Link>
54             </LinkContainer>
55             <LinkContainer to="/login">
56               <Nav.Link> <FontAwesomeIcon icon={faSignInAlt} /> Login</Nav.Link>
57             </LinkContainer>
58           </>
59         ) : (
60           <NavDropdown title={
61             <span>
62               <FontAwesomeIcon icon={faUser} /> {user.login}
63             </span>
64           } id="user-nav-dropdown">
65             <LinkContainer to="/profile">
66               <NavDropdown.Item>
67                 <FontAwesomeIcon icon={faIdCard} /> My profile
68               </NavDropdown.Item>
69             </LinkContainer>
70             <NavDropdown.Divider />
71             <LinkContainer to="#">
72               <NavDropdown.Item onClick={showModal}>
73                 <FontAwesomeIcon icon={faSignOutAlt} /> Logout
74               </NavDropdown.Item>

```

```

75         </LinkContainer>
76     </NavDropdown>
77     ) }
78 </Nav>
79 {
80     isLoggedIn && (
81         <Nav>
82             <LinkContainer to="/test/edit">
83                 <Button variant="primary">
84                     Create test <FontAwesomeIcon icon={faPlus} />
85                 </Button>
86             </LinkContainer>
87         </Nav>
88     )
89 }
90
91 <Modal
92     size="md"
93     aria-labelledby="contained-modal-title-vcenter"
94     centered
95     onHide={hideModal}
96     show={modalShow}
97 >
98     <Modal.Header closeButton>
99         <Modal.Title id="contained-modal-title-vcenter">Log out</Modal.Title>
100     </Modal.Header>
101     <Modal.Body>
102         <p>Are you sure you want to log-off?</p>
103     </Modal.Body>
104     <Modal.Footer>
105         <Button onClick={onLogout}>Yes</Button>
106     </Modal.Footer>
107 </Modal>
108 </Navbar>
109 );
110 }
111
112 Header.propTypes = {
113     history: ReactRouterPropTypes.history,
114     dispatch: PropTypes.func,
115     isLoggedIn: PropTypes.bool,
116     user: PropTypes.shape({
117         login: PropTypes.string
118     })
119 };
120
121 function mapStateToProps(state) {
122     const { isLoggedIn, user } = state.auth;
123
124     return { isLoggedIn, user };
125 }
126
127 const headerWithRouter = withRouter(Header);
128 const connectedHeaderWithRouter = connect(mapStateToProps)(headerWithRouter);
129
130 export { connectedHeaderWithRouter as Header };

```

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

```

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

```

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

```

1 .answer-edit-list {
2     border: 0;
3 }

```

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

```

1 import { AnswerEditList } from "../AnswerEditList";
2
3 export { AnswerEditList };

```

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

```

1 import * as React from "react";
2 import { connect } from "react-redux";
3 import PropTypes from "prop-types";
4 import Form from "react-bootstrap/Form";
5 import Button from "react-bootstrap/Button";
6 import AnswerEditItem from "../AnswerEditItem";
7 import { ANSWER_TYPE } from "../../../models/Test/config";
8 import * as testEditorActions from "../../../actions/testEditor";
9
10 class AnswerEditList extends React.Component {
11   constructor(props) {
12     super(props);
13   }
14
15   render() {
16     const {
17       name,
18       type,
19       answers,
20       onAppendAnswer,
21       deleteAnswer,
22       updateAnswerText,
23       updateAnswers
24     } = this.props;
25
26     return (
27       <>
28         <p>Choose right answer:</p>
29         <Form.Group controlId="">
30           {
31             answers.map((el, i) => {
32               const { content, isRight } = el;
33
34               return (
35                 <AnswerEditItem key={i}
36                   name={name}
37                   type={type}
38                   isRight={isRight}
39                   content={content}
40                   onChangeAnswer={(isRight, typeAnswer) =>
41                     ↪ updateAnswers(name, i, isRight, typeAnswer)}
42                   onDeleteAnswer={() => deleteAnswer(name, i)}
43                   onChangeAnswerText={value => updateAnswerText(name, i,
44                     ↪ value)} />
45               );
46             })
47           }
48         </Form.Group>
49         <Button variant="primary" onClick={onAppendAnswer}>
50           Add answer
51         </Button>
52       </>
53     );
54   }
55
56   static propTypes = {
57     name: PropTypes.oneOfType([ PropTypes.string, PropTypes.number ]).isRequired,
58     type: PropTypes.oneOf([ ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE ]).isRequired,
59     answers: PropTypes.arrayOf(
60       PropTypes.exact({
61         content: PropTypes.string,
62         isRight: PropTypes.bool
63       }).isRequired,
64     ),
65     onAppendAnswer: PropTypes.func.isRequired,
66     deleteAnswer: PropTypes.func.isRequired,

```

```

66     updateAnswerText: PropTypes.func.isRequired,
67     updateAnswers: PropTypes.func
68   };
69
70   function mapStateToProps() {
71     return {};
72   }
73
74   function mapDispatchToProps(dispatch) {
75     return {
76       deleteAnswer: (questionId, answerId) => dispatch(testEditorActions.deleteAnswer(questionId,
77         ↪ answerId)),
78       updateAnswerText: (questionId, answerId, value) =>
79         ↪ dispatch(testEditorActions.updateAnswerText(questionId, answerId, value)),
80       updateAnswers: (questionId, answerId, isRight, typeAnswer) =>
81         ↪ dispatch(testEditorActions.updateAnswers(questionId, answerId, isRight, typeAnswer))
82     };
83   }
84
85   const connectedAnswerEditList = connect(mapStateToProps, mapDispatchToProps)(AnswerEditList);
86
87   export { connectedAnswerEditList as AnswerEditList };

```

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

```

1  import * as React from "react";
2  import PropTypes from "prop-types";
3  import Row from "react-bootstrap/Row";
4  import Col from "react-bootstrap/Col";
5  import Form from "react-bootstrap/Form";
6  import ToggleButtonGroup from "react-bootstrap/ToggleButtonGroup";
7  import ToggleButton from "react-bootstrap/ToggleButton";
8  import { ANSWER_TYPE } from "../../../../../models/Test/config";
9  import { AnswerEditList } from "../AnswerEditList";
10 import Button from "react-bootstrap/Button";
11
12 class QuestionEditItem extends React.Component {
13   constructor(props) {
14     super(props);
15
16     this.handleToggleChange = this.handleToggleChange.bind(this);
17   }
18
19   handleToggleChange({ target: { value } }) {
20     const { onChangeAnswerType } = this.props;
21
22     onChangeAnswerType(value);
23   }
24
25   render() {
26     const {
27       name,
28       typeAnswer,
29       answers,
30       onDelete,
31       onUpdateTitle,
32       onAnswerListUpdate,
33       title
34     } = this.props;
35
36     return (
37       <>
38         <Row>
39           <Col lg={4}>
40             <Form.Label>Question:</Form.Label>
41           </Col>
42           <Col lg={8}>
43             <Form.Group controlId="">
44               <Form.Control className="test-editor__textarea"
45                 as="textarea"
46                 rows={3}
47                 value={title}
48                 onChange={event => {
49                   const titleVal = event.target.value;

```

```

50
51         onUpdateTitle(titleVal);
52     }} />
53     </Form.Group>
54 </Col>
55 </Row>
56 <Row>
57     <Col lg={4}>
58         <Form.Label>Question type:</Form.Label>
59     </Col>
60     <Col lg={8}>
61         <Form.Group>
62             <ToggleButtonGroup type="radio"
63                 name={`_${name}_toggle_answer_type`}
64                 defaultValue={ANSWER_TYPE.ONE}>
65                 <ToggleButton value={ANSWER_TYPE.ONE}
66                     onChange={this.handleToggleChange}>One answer</ToggleButton>
67                 <ToggleButton value={ANSWER_TYPE.MULTIPLE}
68                     onChange={this.handleToggleChange}>Multiple
69                     ↵ answers</ToggleButton>
70             </ToggleButtonGroup>
71         </Form.Group>
72     </Col>
73 </Row>
74 <Row>
75     <Col lg={4}>
76         <Form.Label>Answers:</Form.Label>
77     </Col>
78     <Col lg={8}>
79         <AnswerEditList name={name}
80             type={typeAnswer}
81             answers={answers}
82             onAppendAnswer={() => onAnswerListUpdate(name)} />
83     </Col>
84 </Row>
85 <Row>
86     <Col lg={12}>
87         <Button className="float-right"
88             variant="danger"
89             onClick={onDelete}>
90             Delete question
91         </Button>
92     </Col>
93 </Row>
94 <hr/>
95 </>
96 );
97 }
98 }
99
100 QuestionEditItem.propTypes = {
101     name: PropTypes.oneOfType([ PropTypes.string, PropTypes.number ]).isRequired,
102     typeAnswer: PropTypes.oneOf([ ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE ]).isRequired,
103     title: PropTypes.string,
104     answers: PropTypes.arrayOf(
105         PropTypes.exact({
106             content: PropTypes.string,
107             isRight: PropTypes.bool
108         })
109     ).isRequired,
110     onDelete: PropTypes.func,
111     onUpdateTitle: PropTypes.func,
112     onChangeAnswerType: PropTypes.func,
113     onAnswerListUpdate: PropTypes.func
114 };
115
116 export default QuestionEditItem;

```

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

```

1 import QuestionEditItem from "../QuestionEditItem";
2
3 export default QuestionEditItem;

```

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

```
1 .answer-edit-item {
2   margin: 10px 0;
3 }
```

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

```
1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import InputGroup from "react-bootstrap/InputGroup";
4 import FormControl from "react-bootstrap/FormControl";
5 import Button from "react-bootstrap/Button";
6 import {ANSWER_TYPE} from "../../../../../models/Test/config";
7 import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
8 import { faTrash } from '@fortawesome/free-solid-svg-icons';
9
10 import "./style.scss";
11
12 class AnswerEditItem extends React.Component {
13   constructor(props) {
14     super(props);
15
16     const { content } = props;
17
18     this.state = {
19       answerValue: content
20     }
21
22     this.handleChangeAnswer = this.handleChangeAnswer.bind(this);
23   }
24
25   handleChangeAnswer(event) {
26     const { onChangeAnswer } = this.props;
27     const { checked, type } = event.target;
28
29     onChangeAnswer(checked, type);
30   }
31
32   render() {
33     const {
34       name,
35       type,
36       content,
37       isRight,
38       onDeleteAnswer,
39       onChangeAnswerText
40     } = this.props;
41
42     return (
43       <InputGroup className="answer-edit-item">
44         <InputGroup.Prepend>
45           {
46             (() => {
47               switch (type) {
48                 case ANSWER_TYPE.ONE:
49                   return (
50                     <InputGroup.Radio name={name}
51                       checked={isRight}
52                       onChange={this.handleChangeAnswer}
53                       aria-label="Radio button for following text"
54                       ↵ input" />
55                   );
56                 case ANSWER_TYPE.MULTIPLE:
57                   return (
58                     <InputGroup.Checkbox name={name}
59                       checked={isRight}
60                       onChange={this.handleChangeAnswer}
61                       aria-label="Radio button for following text"
62                       ↵ input" />
63                   );
64                 default:
65                   return null;
66               }
67             })()
68           }
69         </InputGroup.Prepend>
70         <FormControl type={type} value={content} />
71       </InputGroup>
72     );
73   }
74 }
```

```

61         );
62     }
63     })()
64 }
65 </InputGroup.Prepend>
66 <FormControl aria-label="Text input with radio button"
67     value={content}
68     onChange={(event => {
69         const { value } = event.target;
70
71         onChangeAnswerText(value);
72     }} />
73 <InputGroup.Append>
74     <Button variant="danger"
75         onClick={onDeleteAnswer}>
76         <FontAwesomeIcon icon={faTrash} />
77     </Button>
78 </InputGroup.Append>
79 </InputGroup>
80 );
81 }
82 }
83
84 AnswerEditItem.propTypes = {
85     name: PropTypes.oneOfType([ PropTypes.string, PropTypes.number ]).isRequired,
86     type: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]).isRequired,
87     content: PropTypes.string.isRequired,
88     isRight: PropTypes.bool.isRequired,
89     onDeleteAnswer: PropTypes.func.isRequired,
90     onChangeAnswerText: PropTypes.func.isRequired,
91     onChangeAnswer: PropTypes.func.isRequired
92 };
93
94 export default AnswerEditItem;

```

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

```

1 import AnswerEditItem from "../AnswerEditItem";
2
3 export default AnswerEditItem;

```

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

```

1 import TableStatistic from "../TableStatistic";
2
3 export default TableStatistic;

```

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

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import Container from "react-bootstrap/Container";
4 import { MDBDataTable } from 'mdbreact';
5
6
7 const TableStatistic = ({rows}) => {
8     const data = {
9         columns: [
10             {
11                 label: 'Attempt ID',
12                 field: 'attemptId',
13             },
14             {
15                 label: 'User',
16                 field: 'login',
17             },
18             {
19                 label: 'Result',
20                 field: 'result',

```

```

21         },
22         {
23             label: 'Date',
24             field: 'date',
25         }
26     ],
27     rows: rows
28 };
29
30     return (
31         <Container className="p-3">
32             <MDBDataTable
33                 striped
34                 bordered
35                 small
36                 sortable
37                 data={data}
38             />
39         </Container>
40     );
41 };
42
43 TableStatistic.propTypes = {
44     rows: PropTypes.arrayOf(
45         PropTypes.exact({
46             attemptId: PropTypes.number,
47             login: PropTypes.string,
48             result: PropTypes.number,
49             date: PropTypes.string
50         })
51     ).isRequired
52 };
53
54 export default TableStatistic;

```

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

1

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

```

1 import QuestionEditList from "../QuestionEditList";
2
3 export default QuestionEditList;

```

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

```

1 import * as React from "react";
2 import {ANSWER_TYPE} from "../../../../../models/Test/config";
3 import PropTypes from "prop-types";
4
5 import QuestionEditItem from "../QuestionEditItem";
6
7 class QuestionEditList extends React.Component {
8     constructor(props) {
9         super(props);
10    }
11
12    render() {
13        const {
14            questions,
15            deleteQuestion,
16            updateQuestionTitle,
17            changeQuestionType,
18            appendAnswer
19        } = this.props;
20
21        return (
22            <>

```

```

23         {
24             questions.map((el, i) => {
25                 const {
26                     typeAnswer,
27                     answers,
28                     title
29                 } = el;
30
31                 return (
32                     <QuestionEditItem name={i}
33                                     key={i}
34                                     typeAnswer={typeAnswer}
35                                     answers={answers}
36                                     title={title}
37                                     onDelete={() => deleteQuestion(i)}
38                                     onUpdateTitle={titleVal => updateQuestionTitle(i, titleVal)}
39                                     onChangeAnswerType={typeAnswer => changeQuestionType(i,
40                                                             ↵ typeAnswer)}
41                                     onAnswerListUpdate={id => appendAnswer(id)} />
42                 );
43             })
44         </>
45     );
46 }
47 }
48
49 QuestionEditList.propTypes = {
50     questions: PropTypes.arrayOf(
51         PropTypes.exact({
52             title: PropTypes.string,
53             typeAnswer: PropTypes.oneOf([ ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE ]),
54             answers: PropTypes.arrayOf(
55                 PropTypes.exact({
56                     content: PropTypes.string,
57                     isRight: PropTypes.bool
58                 })
59             )
60         }).isRequired
61     ),
62     deleteQuestion: PropTypes.func,
63     updateQuestionTitle: PropTypes.func,
64     changeQuestionType: PropTypes.func,
65     appendAnswer: PropTypes.func
66 };
67
68 export default QuestionEditList;

```

db-course-project-app/src/client/apps/contest/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 Home from './pages/Home';
9 import Login from "../../pages/Login";
10 import SignUp from "../../pages/SignUp";
11 import Test from "../../containers/Test";
12 import TestResult from "../../pages/TestResult";
13
14 import Footer from "../../components/Footer";
15 import Header from "./components/Header";
16 import AllTests from "./pages/AllTests";
17 import HttpErrorInfo from "../../components/HttpErrorInfo";
18 import {NOT_FOUND} from "http-status-codes";
19
20 class App extends React.Component {
21     render() {
22         return (
23             <div>
24                 <Header />

```

```

25
26         <Switch>
27             <Route exact path="/" component={Home} />
28             <PrivateRoute path="/tests" component={AllTests} />
29             <PrivateRoute exact path="/test/pass" component={Test} />
30             <NotIsLoggedInRoute path="/login" component={Login}/>
31             <NotIsLoggedInRoute path="/signup" component={SignUp} />
32             <PrivateRoute exact path="/test/result" component={TestResult} />
33             <Route component={() => <HttpErrorInfo status={NOT_FOUND} />} />
34         </Switch>
35
36         <Container className="p-3">
37             <Footer/>
38         </Container>
39     </div>
40 );
41 }
42 }
43
44 export default App;

```

db-course-project-app/src/client/apps/contest/pages/AllTests/style.scss

1

...-course-project-app/src/client/apps/contest/pages/AllTests/AllTests.jsx

```

1 import * as React from "react";
2 import Container from "react-bootstrap/Container";
3 import ListTestCards from "../../components/ListTestCards";
4 import * as editTest from "../../services/editTest";
5 import HttpErrorInfo from "../../components/HttpErrorInfo";
6 import {NO_CONTENT} from "http-status-codes";
7 import Form from "react-bootstrap/Form";
8 import InputGroup from "react-bootstrap/InputGroup";
9 import FormControl from "react-bootstrap/FormControl";
10 import Button from "react-bootstrap/Button";
11 import Row from "react-bootstrap/Row";
12 import Col from "react-bootstrap/Col";
13 import TagList from "../../components/TagList";
14
15 import "../style.scss";
16
17 class AllTests extends React.Component {
18     constructor(props) {
19         super(props);
20
21         this.state = {
22             tests: [],
23             searchTitle: '',
24             searchTagValue: '',
25             searchTags: []
26         };
27
28         this.handleSearchTitleChange = this.handleSearchTitleChange.bind(this);
29         this.handleSearchTagValueChange = this.handleSearchTagValueChange.bind(this);
30         this.handleAddingTag = this.handleAddingTag.bind(this);
31         this.handleSearchTagDeleting = this.handleSearchTagDeleting.bind(this);
32         this.searchFilter = this.searchFilter.bind(this);
33     }
34
35     handleSearchTagDeleting(tagId) {
36         this.setState(prev => {
37             const { searchTags } = prev;
38
39             return {
40                 searchTags: [
41                     ...searchTags.slice(0, tagId),
42                     ...searchTags.slice(tagId + 1),
43                 ]
44             };
45         });
46     }
47 }

```



```

45     });
46 }
47
48 handleSearchTitleChange({ target }) {
49     const { value } = target;
50
51     this.setState({ searchTitle: value });
52 }
53
54 handleSearchTagValueChange({ target }) {
55     const { value } = target;
56
57     this.setState({ searchTagValue: value })
58 }
59
60 handleAddingTag() {
61     const { searchTagValue, searchTags } = this.state;
62     const tagIsNotIncluded = !searchTags.includes(searchTagValue);
63
64     if (searchTagValue && tagIsNotIncluded) {
65         this.setState(prev => {
66             return {
67                 searchTags: [...prev.searchTags, searchTagValue]
68             };
69         });
70     }
71 }
72
73 async componentDidMount() {
74     const responseJson = await editTest.getAllTests();
75
76     this.setState({ tests: responseJson });
77 }
78
79 searchFilter() {
80     const { tests, searchTags, searchTitle } = this.state;
81
82     return tests.filter(({ tags, title }) => {
83         const containAllTags = tags.some(tag => searchTags.includes(tag));
84         const lowerTitle = title.toLowerCase();
85         const lowerSearchTitle = searchTitle.toLowerCase();
86         const likeTitle = lowerTitle.indexOf(lowerSearchTitle) !== -1;
87
88         return (searchTags.length > 0 ? containAllTags : true) &&
89             (searchTitle ? likeTitle : true);
90     });
91 }
92
93 render() {
94     const { searchTags } = this.state;
95     const tests = this.searchFilter();
96
97     return (
98         <Container className="p-3">
99             <Row>
100                 <Col lg={12} style={{ padding: '0' }}>
101                     <Form>
102                         <Form.Group controlId="">
103                             <FormControl
104                                 aria-label="test title"
105                                 aria-describedby="basic-addon2"
106                                 required
107                                 placeholder="Enter test title"
108                                 onChange={this.handleSearchTitleChange}
109                             />
110                         </Form.Group>
111                         <Form.Group controlId="">
112                             <InputGroup className="mb-3">
113                                 <FormControl
114                                     aria-label="Recipient's username"
115                                     aria-describedby="basic-addon2"
116                                     required
117                                     placeholder="Enter tag name"
118                                     onChange={this.handleSearchTagValueChange}
119                                 />
120                                 <InputGroup.Append>

```

```

121         <Button variant="primary"
122             onClick={this.handleAddingTag}>
123             Add
124         </Button>
125     </InputGroup.Append>
126 </InputGroup>
127 </Form.Group>
128 <TagList tags={searchTags}
129     canDelete={true}
130     deleteTag={this.handleSearchTagDeleting} />
131 </Form>
132 </Col>
133 </Row>
134 {
135     tests.length ? (
136         <ListTestCards tests={tests}
137             editMenu={false} />
138     ) : (
139         <HttpErrorInfo status={NO_CONTENT}
140             reason="You don't have tests." />
141     )
142 }
143 </Container>
144 );
145 }
146 }
147
148 export default AllTests;

```

db-course-project-app/src/client/apps/contest/pages/AllTests/index.js

```

1 import AllTests from "../AllTests";
2
3 export default AllTests;

```

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

```

1

```

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

```

1 import Home from "../Home";
2
3 export default Home;

```

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

```

1 import * as React from "react";
2
3 import Jumbotron from "react-bootstrap/Jumbotron";
4 import Container from "react-bootstrap/Container";
5 import Row from "react-bootstrap/Row";
6 import Col from "react-bootstrap/Col";
7
8 import "../style.scss";
9
10 const Home = () => {
11     return (
12         <Container className="p-3">
13             <Jumbotron>
14                 <h1>Hello!</h1>
15                 <p>
16                     This is a simple hero unit, a simple jumbotron-style component for calling
17                     extra attention to featured content or information.
18                 </p>
19             </Jumbotron>
20             <Row>

```

```

21         <Col lg={4}>
22             <h2>Create</h2>
23             <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>
24         </Col>
25         <Col lg={4}>
26             <h2>Share</h2>
27             <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>
28         </Col>
29         <Col lg={4}>
30             <h2>Statistic</h2>
31             <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>
32         </Col>
33     </Row>
34 </Container>
35 );
36 }
37
38 export default Home;

```

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

```

1 //header {
2 //  @_user-dropdown-menu {
3 //    float: right;
4 //  }
5 //}

```

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

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import { withRouter } from "react-router-dom";
4 import ReactRouterPropTypes from "react-router-prop-types";
5 import { connect } from "react-redux";
6 import * as authActions from "../../../../../actions/auth";
7
8 import {
9   faUser,
10  faSignOutAlt,
11  faSignInAlt,
12  faUserPlus,
13  faClipboardList,
14 } from "@fortawesome/free-solid-svg-icons";
15 import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
16 import Navbar from "react-bootstrap/Navbar";
17 import Nav from "react-bootstrap/Nav";
18 import NavDropdown from "react-bootstrap/NavDropdown";
19 import Modal from "react-bootstrap/Modal";
20 import Button from "react-bootstrap/Button";
21 import { LinkContainer } from "react-router-bootstrap";
22
23 import "./style.scss";
24
25 const Header = ({ isLoggedIn, user, dispatch, history }) => {
26   const [modalShow, setModalShow] = React.useState(false);
27
28   const showModal = () => setModalShow(true);
29
30   const hideModal = () => setModalShow(false);
31
32   const onLogOut = () => {
33     dispatch(authActions.logOut());
34
35     setModalShow(false);
36
37     history.push("/");

```

```

38     };
39
40     return (
41         <Navbar bg="dark" variant="dark">
42             <LinkContainer to="/">
43                 <Navbar.Brand>
44                     <FontAwesomeIcon icon={faClipboardList} /> PassQuiz Contest
45                 </Navbar.Brand>
46             </LinkContainer>
47             <Nav className="mr-auto">
48                 { !isLoggedIn ? (
49                     <>
50                         <LinkContainer to="/signup">
51                             <Nav.Link> <FontAwesomeIcon icon={faUserPlus} /> Sign Up</Nav.Link>
52                         </LinkContainer>
53                         <LinkContainer to="/login">
54                             <Nav.Link> <FontAwesomeIcon icon={faSignInAlt} /> Login</Nav.Link>
55                         </LinkContainer>
56                     </>
57                 ) : (
58                     <NavDropdown title={
59                         <span>
60                             <FontAwesomeIcon icon={faUser} /> {user.login}
61                         </span>
62                     } id="user-nav-dropdown">
63                         <LinkContainer to="#">
64                             <NavDropdown.Item onClick={showModal}>
65                                 <FontAwesomeIcon icon={faSignOutAlt} /> Logout
66                             </NavDropdown.Item>
67                         </LinkContainer>
68                     </NavDropdown>
69                 ) }
70             </Nav>
71             {
72                 isLoggedIn && (
73                     <Nav>
74                         <LinkContainer to="/tests">
75                             <Button variant="primary">Tests</Button>
76                         </LinkContainer>
77                     </Nav>
78                 )
79             }
80
81             <Modal
82                 size="md"
83                 aria-labelledby="contained-modal-title-vcenter"
84                 centered
85                 onHide={hideModal}
86                 show={modalShow}
87             >
88                 <Modal.Header closeButton>
89                     <Modal.Title id="contained-modal-title-vcenter">Log out</Modal.Title>
90                 </Modal.Header>
91                 <Modal.Body>
92                     <p>Are you sure you want to log-off?</p>
93                 </Modal.Body>
94                 <Modal.Footer>
95                     <Button onClick={onLogout}>Yes</Button>
96                 </Modal.Footer>
97             </Modal>
98         </Navbar>
99     );
100 }
101
102 Header.propTypes = {
103     history: ReactRouterPropTypes.history,
104     dispatch: PropTypes.func,
105     isLoggedIn: PropTypes.bool,
106     user: PropTypes.shape({
107         login: PropTypes.string
108     })
109 };
110
111 function mapStateToProps(state) {
112     const { isLoggedIn, user } = state.auth;
113

```

```

114     return { isLoggedIn, user };
115 }
116
117 const headerWithRouter = withRouter(Header);
118 const connectedHeaderWithRouter = connect(mapStateToProps)(headerWithRouter);
119
120 export { connectedHeaderWithRouter as Header };

```

...b-course-project-app/src/client/apps/contest/components/Header/index.js

```

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

```

db-course-project-app/src/client/components/TagList/style.scss

```

1 .tag-list {
2   display: inline;
3   margin: 10px 0;
4   padding: 0;
5 }

```

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

```

1 import TagList from "../TagList";
2
3 export default TagList;

```

db-course-project-app/src/client/components/TagList/TagList.jsx

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import Tag from "../Tag";
4
5 import "../style.scss";
6
7 const TagList = ({ tags, deleteTag, canDelete = true }) => {
8   return (
9     <ul className="tag-list">
10       {
11         tags.map((content, i) => {
12           return (
13             <Tag content={content}
14               key={i}
15               canDelete={canDelete}
16               onDelete={() => deleteTag(i)} />
17           );
18         })
19       }
20     </ul>
21   );
22 };
23
24 TagList.propTypes = {
25   tags: PropTypes.arrayOf(PropTypes.string).isRequired,
26   canDelete: PropTypes.bool,
27   deleteTag: PropTypes.func,
28 };
29
30 export default TagList;

```

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

```

1 $tag-color: #0366d6;

```

```

2 $tag-bg-color: #f1f8ff;
3
4 .tag {
5   background-color: $tag-bg-color;
6   border-radius: 10%;
7   color: $tag-color;
8   display: inline-block;
9   list-style: none;
10  margin: 5px;
11  padding: 6px 8px;
12  user-select: none;
13
14  &__content {
15    margin-right: 5px;
16  }
17
18  &__delete-btn {
19    background: none;
20    border: 0;
21    border-radius: 100%;
22    color: $tag-color;
23    font-size: 0.85rem;
24    padding: 5px 7px;
25  }
26
27  &__delete-btn:hover {
28    background-color: #def;
29  }
30
31  &__delete-btn:focus {
32    outline: none;
33  }
34 }
35 }

```

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

```

1 import Tag from "./Tag";
2
3 export default Tag;

```

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

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
4 import { faTimes } from '@fortawesome/free-solid-svg-icons';
5
6 import "./style.scss";
7
8 const Tag = ({ content, onDelete, canDelete }) => {
9   return (
10     <li className="tag">
11       <span className="tag__content">{content}</span>
12       {
13         canDelete && (
14           <button className="tag__delete-btn"
15             onClick={event => {
16               event.preventDefault();
17               onDelete();
18             }}>
19             <FontAwesomeIcon icon={faTimes} />
20           </button>
21         )
22       }
23     </li>
24   );
25 };
26
27 Tag.propTypes = {
28   content: PropTypes.string.isRequired,
29   canDelete: PropTypes.bool.isRequired,

```

```

30     onDelete: PropTypes.func,
31   };
32
33   export default Tag;

```

db-course-project-app/src/client/components/ResultStatus/style.scss

```

1 .result-status {
2   font-size: 1.2rem;
3
4   &__correct {
5     color: #77d418;
6   }
7
8   &__incorrect {
9     color: #d92727;
10  }
11 }

```

...-course-project-app/src/client/components/ResultStatus/ResultStatus.jsx

```

1 import * as React from "react";
2 import { faCheckCircle, faTimesCircle } from "@fortawesome/free-regular-svg-icons";
3 import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
4 import PropTypes from "prop-types";
5
6 import "./style.scss";
7
8 const ResultStatus = ({ isCorrect }) => {
9   return (
10     <span className={`result-status ${isCorrect ? 'result-status__correct' :
11       'result-status__incorrect'}`}>
12       <FontAwesomeIcon icon={isCorrect ? faCheckCircle : faTimesCircle} />
13     </span>
14   );
15 };
16
17 ResultStatus.propTypes = {
18   isCorrect: PropTypes.bool.isRequired
19 };
20 export default ResultStatus;

```

db-course-project-app/src/client/components/ResultStatus/index.js

```

1 import ResultStatus from "./ResultStatus";
2
3 export default ResultStatus;

```

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

```

1 .test-card {
2   margin: 0 0 15px;
3   width: 100%;
4
5   &__title {
6     font-size: 1.1rem;
7   }
8
9   &__label-info {
10    font-weight: bold;
11    padding-right: 5px;
12  }
13
14  &__control {
15    display: flex;
16    justify-content: space-between;

```

```

17   }
18 }

```

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

```

1 import TestCard from "../TestCard";
2
3 export default TestCard;

```

db-course-project-app/src/client/components/TestCard/TestCard.jsx

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
4 import {
5   faUser,
6   faTags,
7   faTrash,
8   faEdit,
9   faShareSquare,
10  faPollH
11 } from "@fortawesome/free-solid-svg-icons";
12 import Card from "react-bootstrap/Card";
13 import Button from "react-bootstrap/Button";
14 import { LinkContainer } from "react-router-bootstrap";
15 import Dropdown from "react-bootstrap/Dropdown";
16 import TagList from "../TagList";
17
18 import "../style.scss";
19
20 const TestCard = ({
21   title,
22   description,
23   author, tags,
24   onDeleteTestCard,
25   testId,
26   editMenu=true
27 }) => {
28   return (
29     <Card className="test-card">
30       <Card.Body>
31         <Card.Title className="test-card__title">{title}</Card.Title>
32
33         <div className="test-card__author">
34           <span className="test-card__label-info">
35             <FontAwesomeIcon icon={faUser} /> Author:
36           </span>
37           {author}
38         </div>
39
40         <div className="test-card__tags">
41           <span className="test-card__label-info">
42             <FontAwesomeIcon icon={faTags} /> Tags:
43           </span>
44           {
45             tags.length ? (
46               <TagList tags={tags} canDelete={false} />
47             ) : (
48               "None"
49             )
50           }
51         </div>
52
53         <Card.Text className="test-card__description">{description}</Card.Text>
54         <div className="test-card__control">
55           <LinkContainer to={` /test/pass?id=${testId}`}>
56             <Button className="test-card__pass-btn"
57               variant="primary">Pass test</Button>
58           </LinkContainer>
59
60           {
61             editMenu && (

```



```

62         <Dropdown className="test-card__dropdown-menu">
63             <Dropdown.Toggle variant="primary">Menu</Dropdown.Toggle>
64             <Dropdown.Menu>
65                 <LinkContainer to={` /test/edit?id=${testId}`}>
66                     <Dropdown.Item as="button">
67                         <FontAwesomeIcon icon={faEdit} /> Edit
68                     </Dropdown.Item>
69                 </LinkContainer>
70                 <Dropdown.Item as="button"
71                     onClick={() => {
72                         onDeleteTestCard(testId);
73                     }}>
74                     <FontAwesomeIcon icon={faTrash} /> Delete
75                 </Dropdown.Item>
76                 <Dropdown.Item as="button" onClick={async () => {
77                     const link = `${location.origin}/test/pass?id=${testId}`;
78
79                     try {
80                         await navigator.clipboard.writeText(link);
81
82                         alert("Link has been copied!");
83                     } catch (err) {
84                         alert('Could not copy link. ');
85                     }
86                 }}>
87                     <FontAwesomeIcon icon={faShareSquare} /> Share
88                 </Dropdown.Item>
89                 <LinkContainer to={` /test/statistic?id=${testId}`}>
90                     <Dropdown.Item as="button">
91                         <FontAwesomeIcon icon={faPollH} /> Show results
92                     </Dropdown.Item>
93                 </LinkContainer>
94             </Dropdown.Menu>
95         </Dropdown>
96     )
97 }
98 </div>
99 </Card.Body>
100 </Card>
101 );
102 };
103
104 TestCard.propTypes = {
105     title: PropTypes.string.isRequired,
106     description: PropTypes.string.isRequired,
107     author: PropTypes.string.isRequired,
108     tags: PropTypes.arrayOf(PropTypes.string).isRequired,
109     testId: PropTypes.number.isRequired,
110     onDeleteTestCard: PropTypes.func,
111     editMenu: PropTypes.bool
112 };
113
114 export default TestCard;

```

db-course-project-app/src/client/components/Question/style.scss

1

db-course-project-app/src/client/components/Question/index.js

```

1 import Question from "../Question";
2
3 export default Question;

```

db-course-project-app/src/client/components/Question/Question.jsx

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import Card from "react-bootstrap/Card";

```

```

4 import AnswerList from "../AnswerList";
5 import Answer from "../Answer";
6 import {ANSWER_TYPE} from "../../../models/Test/config";
7
8 import "../style.scss";
9
10 const Question = ({ title, answers, type, id, onAnswerChange }) => {
11   return (
12     <Card style={{ width: '100%' }}>
13       <Card.Header>
14         <Card.Title>Question {id + 1}</Card.Title>
15         <Card.Body>
16           <Card.Text>{title}</Card.Text>
17         </Card.Body>
18       </Card.Header>
19
20       <AnswerList>
21         {
22           answers.map((answer, i) => {
23             const name = `${id}_answer`;
24             const { content, isChecked } = answer;
25
26             return (
27               <Answer type={type}
28                 content={content}
29                 name={name}
30                 isChecked={isChecked}
31                 onAnswerChange={checked => onAnswerChange(id, i, type, checked)}
32                 key={i} />
33             );
34           })
35         }
36       </AnswerList>
37     </Card>
38   );
39 };
40
41 Question.propTypes = {
42   title: PropTypes.string.isRequired,
43   type: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]).isRequired,
44   answers: PropTypes.arrayOf(
45     PropTypes.exact({
46       content: PropTypes.string,
47       isChecked: PropTypes.bool
48     })
49   ).isRequired,
50   id: PropTypes.number.isRequired,
51   onAnswerChange: PropTypes.func.isRequired
52 };
53
54 export default Question;

```

db-course-project-app/src/client/components/ListTestCards/style.scss

```

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

```

db-course-project-app/src/client/components/ListTestCards/index.js

```

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

```

...ourse-project-app/src/client/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, editMenu=true, 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               editMenu={editMenu}
30               onDeleteTestCard={onDeleteTestCard} />
31           );
32         })
33       }
34     </Row>
35   );
36 };
37
38 ListTestCards.propTypes = {
39   tests: PropTypes.array.isRequired,
40   onDeleteTestCard: PropTypes.func,
41   editMenu: PropTypes.bool
42 };
43
44 export default ListTestCards;

```

db-course-project-app/src/client/components/AnswerList/style.scss

1

db-course-project-app/src/client/components/AnswerList/index.js

```

1 import AnswerList from "../AnswerList";
2
3 export default AnswerList;

```

db-course-project-app/src/client/components/AnswerList/AnswerList.jsx

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import ListGroup from "react-bootstrap/ListGroup";
4
5 import "./style.scss";
6
7 const AnswerList = ({ children }) => {
8   return (
9     <ListGroup className="list-group-flush">
10       {children}
11     </ListGroup>
12   );
13 };

```

```

14
15 AnswerList.propTypes = {
16   children: PropTypes.arrayOf(PropTypes.element).isRequired
17 };
18
19 export default AnswerList;

```

db-course-project-app/src/client/components/Answer/style.scss

```

1

```

db-course-project-app/src/client/components/Answer/Answer.jsx

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import ListGroupItem from "react-bootstrap/ListGroupItem";
4 import Form from "react-bootstrap/Form";
5 import {ANSWER_TYPE} from "../../models/Test/config";
6
7 import "./style.scss";
8
9 const Answer = ({ type, name, content, isChecked, onAnswerChange }) => {
10   return (
11     <ListGroupItem>
12       <Form.Check
13         type={type}
14         name={name}
15         label={content}
16         checked={isChecked}
17         onChange={event => {
18           const { checked } = event.target;
19
20           onAnswerChange(checked);
21         }}
22       />
23     </ListGroupItem>
24   );
25 };
26
27 Answer.propTypes = {
28   type: PropTypes.oneOf([ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE]).isRequired,
29   name: PropTypes.string.isRequired,
30   content: PropTypes.string.isRequired,
31   isChecked: PropTypes.bool.isRequired,
32   onAnswerChange: PropTypes.func.isRequired
33 };
34
35 export default Answer;

```

db-course-project-app/src/client/components/Answer/index.js

```

1 import Answer from "../Answer";
2
3 export default Answer;

```

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

```

1 .footer {
2   &__copyright {
3     font-weight: bold;
4   }
5 }

```

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

```

1 import * as React from "react";
2
3 import "../style.scss"
4
5 const Footer = () => {
6   const currDate = new Date();
7
8   return (
9     <footer className="footer">
10       <hr/>
11       <p className="footer__copyright">©169; Copyright {currDate.getFullYear()}</p>
12     </footer>
13   );
14 };
15
16 export default Footer;

```

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

```

1 import Footer from '../Footer';
2
3 export default Footer;

```

...rse-project-app/src/client/components/ErrorMessage/ErrorMessage.jsx

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3
4 import Alert from "react-bootstrap/Alert";
5
6 const ErrorMessage = ({ listErrors, show, onHide }) => {
7   return (
8     <Alert variant="danger" show={show} onClose={onHide} dismissible>
9       <Alert.Heading>You got an error!</Alert.Heading>
10       <ul>
11         {
12           listErrors.map(({ message }, i) => <li key={i}>{message}</li>)
13         }
14       </ul>
15     </Alert>
16   );
17 }
18
19 ErrorMessage.propTypes = {
20   listErrors: PropTypes.array,
21   show: PropTypes.bool,
22   onHide: PropTypes.func
23 };
24
25 export default ErrorMessage;

```

db-course-project-app/src/client/components/ErrorMessage/index.js

```

1 import ErrorMessage from "../ErrorMessage";
2
3 export default ErrorMessage;

```

db-course-project-app/src/client/components/HttpErrorInfo/style.scss

```

1 .http-error-info {
2   align-items: center;
3   display: flex;
4   justify-content: center;
5   min-height: 360px;
6
7   &__box {

```

```

8     margin: auto;
9     text-align: center;
10 }
11 }

```

...ourse-project-app/src/client/components/HttpErrorInfo/HttpErrorInfo.jsx

```

1 import * as React from "react";
2 import PropTypes from "prop-types";
3 import { getStatusText } from "http-status-codes";
4
5 import "./style.scss";
6
7 const HttpErrorInfo = ({ status, reason }) => {
8     const reasonText = reason ? reason : getStatusText(status);
9
10    return (
11        <div className="http-error-info">
12            <div className="http-error-info__box">
13                <h3 className="http-error-info__status">{status}</h3>
14                <p className="http-error-info__reason">{reasonText}</p>
15            </div>
16        </div>
17    );
18 };
19
20 HttpErrorInfo.propTypes = {
21     status: PropTypes.number.isRequired,
22     reason: PropTypes.string
23 }
24
25 export default HttpErrorInfo;

```

db-course-project-app/src/client/components/HttpErrorInfo/index.js

```

1 import HttpErrorInfo from "../HttpErrorInfo";
2
3 export default HttpErrorInfo;

```

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

```

1 export const LOGIN_APPROVE = 'LOGIN_APPROVE';
2 export const LOGIN_FAILED = 'LOGIN_FAILED';
3 export const LOGOUT = 'LOGOUT';
4
5 export const success = (user) => {
6     return { type: LOGIN_APPROVE, user, isLoggedIn: true };
7 };
8
9 export const failed = () => {
10    return { type: LOGIN_FAILED, user: null, isLoggedIn: false };
11 };
12
13 export const logOut = () => {
14    return { type: LOGOUT, user: null, isLoggedIn: false };
15 };

```

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

```

1 export const INIT = "INIT";
2 export const UPDATE_ANSWER = "UPDATE_ANSWER";
3
4 /**
5  * Action that init
6  * @param {Object[]} internalState - Store of questions
7  * @param {string} internalState[].title - Question title
8  * @param {string} internalState[].typeAnswer - Question type

```

```

9  * @param {Object[]} internalState[].answers - Array of answers
10 * @param {string} internalState[].answers[].content - Answer content
11 * @param {boolean} internalState[].answers[].isChecked - Answer state
12 * @return {{internalState: Object[], type: string}}
13 */
14 export const init = internalState => {
15     return {
16         type: INIT,
17         internalState
18     };
19 };
20
21 /**
22  * Action that updates answer in question
23  * @param {number} questionId
24  * @param {number} answerId
25  * @param {string} typeAnswer
26  * @param {boolean} state
27  * @return {{answerId: number, questionId: number, state: boolean, type: string}}
28  */
29 export const updateAnswer = (questionId,
30                             answerId,
31                             typeAnswer,
32                             state) => {
33     return {
34         type: UPDATE_ANSWER,
35         questionId,
36         answerId,
37         typeAnswer,
38         state
39     };
40 };

```

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

```

1  export const RESET = "RESET";
2  export const UPDATE = "UPDATE";
3  export const UPDATE_TITLE = "UPDATE_TITLE";
4  export const UPDATE_DESCRIPTION = "UPDATE_DESCRIPTION";
5  export const APPEND_TAG = "APPEND_TAG";
6  export const DELETE_TAG = "DELETE_TAG";
7  export const APPEND_QUESTION = "APPEND_QUESTION";
8  export const DELETE_QUESTION = "DELETE_QUESTION";
9  export const UPDATE_QUESTION_TITLE = "UPDATE_QUESTION_TITLE";
10 export const UPDATE_QUESTION_TYPE = "UPDATE_QUESTION_TYPE";
11 export const APPEND_ANSWER = "APPEND_ANSWER";
12 export const DELETE_ANSWER = "DELETE_ANSWER";
13 export const UPDATE_ANSWER_TEXT = "UPDATE_ANSWER_TEXT";
14 export const UPDATE_ANSWERS = "UPDATE_ANSWERS";
15
16 export const reset = () => {
17     return { type: RESET };
18 };
19
20 export const update = content => {
21     return {
22         type: UPDATE,
23         content
24     };
25 };
26
27 export const updateTitle = title => {
28     return {
29         type: UPDATE_TITLE,
30         title
31     };
32 };
33
34 export const updateDescription = description => {
35     return {
36         type: UPDATE_DESCRIPTION,
37         description
38     };
39 };

```

```

40
41 export const appendTag = tag => {
42   return {
43     type: APPEND_TAG,
44     tag
45   };
46 };
47
48 export const deleteTag = id => {
49   return {
50     type: DELETE_TAG,
51     id
52   };
53 };
54
55 export const appendQuestion = () => {
56   return { type: APPEND_QUESTION };
57 };
58
59 export const deleteQuestion = id => {
60   return {
61     type: DELETE_QUESTION,
62     id
63   };
64 };
65
66 export const updateQuestionTitle = (id, title) => {
67   return {
68     type: UPDATE_QUESTION_TITLE,
69     id,
70     title
71   };
72 };
73
74 export const changeQuestionType = (id, typeAnswer) => {
75   return {
76     type: UPDATE_QUESTION_TYPE,
77     id,
78     typeAnswer
79   };
80 };
81
82 export const appendAnswer = questionId => {
83   return { type: APPEND_ANSWER, questionId }
84 };
85
86 export const deleteAnswer = (questionId, answerId) => {
87   return {
88     type: DELETE_ANSWER,
89     questionId,
90     answerId
91   };
92 };
93
94 export const updateAnswerText = (questionId, answerId, value) => {
95   return {
96     type: UPDATE_ANSWER_TEXT,
97     questionId,
98     answerId,
99     value
100   };
101 };
102
103 export const updateAnswers = (questionId, answerId, isRight, typeAnswer) => {
104   return {
105     type: UPDATE_ANSWERS,
106     questionId,
107     answerId,
108     isRight,
109     typeAnswer
110   };
111 };

```

```
1 import { createBrowserHistory } from "history";
2
3 export default createBrowserHistory();
```

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

```
1 import {getToken} from "../helpers/token";
2 import {appendAuth} from "../helpers/header";
3 import {INTERNAL_SERVER_ERROR} from "http-status-codes";
4
5 /**
6  * Get attempt results
7  * @param {number} attemptId - Attempt ID
8  * @return {Promise<unknown>}
9  */
10 export const init = async attemptId => {
11   const token = getToken();
12   const headers = new Headers();
13
14   appendAuth(headers, token);
15
16   // create url
17   const url = `/api/test/result?id=${attemptId}`;
18
19   const response = await fetch(url, {
20     method: 'GET',
21     headers,
22   });
23
24   let responseJson = null;
25   try {
26     responseJson = await response.json();
27   } catch (err) {
28     // handle case if json is invalid
29     throw [{
30       status: INTERNAL_SERVER_ERROR,
31       message: 'something went wrong'
32     }];
33   }
34
35   if (response.ok) {
36     const { userAnswers } = responseJson;
37
38     return Promise.resolve(userAnswers);
39   } else {
40     throw responseJson;
41   }
42 };
```

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

```
1 import {getToken} from "../helpers/token";
2 import {appendAuth} from "../helpers/header";
3 import {INTERNAL_SERVER_ERROR} from "http-status-codes";
4
5 /**
6  * Get own attempts
7  * @return {Promise<Object[]>}
8  */
9 export const getOwnAttempts = async () => {
10   const token = getToken();
11   const headers = new Headers();
12
13   appendAuth(headers, token);
14
15   // create url
16   const url = `/api/attempt/profile`;
17
18   const response = await fetch(url, {
19     method: 'GET',
20     headers
```

```

21     });
22
23     let responseJson = null;
24     try {
25         responseJson = await response.json();
26     } catch (err) {
27         // handle case if json is invalid
28         throw [{
29             status: INTERNAL_SERVER_ERROR,
30             message: 'something went wrong'
31         }];
32     }
33
34     if (response.ok) {
35         return Promise.resolve(responseJson);
36     } else {
37         throw responseJson;
38     }
39 };
40
41 /**
42  * Get own attempts
43  * @param {number} testId - Test ID
44  * @return {Promise<Object[]>}
45  */
46 export const getOwnTestAttempts = async (testId) => {
47     const token = getToken();
48     const headers = new Headers();
49
50     appendAuth(headers, token);
51
52     // create url
53     const url = `/api/attempt/test?id=${testId}`;
54
55     const response = await fetch(url, {
56         method: 'GET',
57         headers
58     });
59
60     let responseJson = null;
61     try {
62         responseJson = await response.json();
63     } catch (err) {
64         // handle case if json is invalid
65         throw [{
66             status: INTERNAL_SERVER_ERROR,
67             message: 'something went wrong'
68         }];
69     }
70
71     if (response.ok) {
72         return Promise.resolve(responseJson);
73     } else {
74         throw responseJson;
75     }
76 };

```

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

```

1 import {removeToken} from "../helpers/token";
2
3 async function signUp(formData) {
4     const response = await fetch("/api/signup", {
5         method: "POST",
6         body: formData
7     });
8
9     if (response.ok) {
10         return Promise.resolve();
11     } else {
12         throw await response.json();
13     }
14 }
15

```

```

16 async function signIn(formData) {
17     const response = await fetch("/api/signin", {
18         method: "POST",
19         body: formData
20     });
21
22     const responseJson = await response.json();
23
24     if (response.ok) {
25         return Promise.resolve(responseJson);
26     } else {
27         throw responseJson;
28     }
29 }
30
31 export function logOut() {
32     removeToken();
33 }
34
35 export default {
36     signUp,
37     signIn,
38     logOut
39 };

```

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

```

1 import {getToken} from "../helpers/token";
2 import {appendAuth, appendJSON} from "../helpers/header";
3 import {INTERNAL_SERVER_ERROR, NOT_FOUND} from "http-status-codes";
4
5 /**
6  * Get test for passing
7  * @param {number} testId - Test ID
8  * @return {Promise<unknown>}
9  */
10 export const init = async testId => {
11     const token = getToken();
12     const headers = new Headers();
13
14     appendAuth(headers, token);
15
16     // create url
17     const url = `/api/test/pass?id=${testId}`;
18
19     const response = await fetch(url, {
20         method: 'GET',
21         headers
22     });
23
24     let responseJson = null;
25     try {
26         responseJson = await response.json();
27     } catch (err) {
28         // handle case if json is invalid
29         throw {
30             status: INTERNAL_SERVER_ERROR,
31             message: 'something went wrong'
32         };
33     }
34
35     if (response.ok) {
36         return Promise.resolve(responseJson);
37     } else {
38         switch (response.status) {
39             case NOT_FOUND: {
40                 throw {
41                     status: NOT_FOUND,
42                     message: 'test not found'
43                 };
44             }
45             default: {
46                 // TODO: handle default case
47

```

```

48         console.log('something went wrong');
49     }
50 }
51 }
52 };
53
54
55 /**
56  * Send answers for checking and return attempt ID if status code is successful
57  * @param {Object[]} questions - Question list
58  * @param {number} testId - Test ID
59  * @return {Promise<number|Object[]>}
60  */
61 export const submit = async (questions, testId) => {
62     const token = getToken();
63     const headers = new Headers();
64
65     appendAuth(headers, token);
66     appendJSON(headers);
67
68     // create url
69     const url = `/api/test/check`;
70
71     const response = await fetch(url, {
72         method: 'POST',
73         headers,
74         body: JSON.stringify({ questions, testId })
75     });
76
77     let responseJson = null;
78     try {
79         responseJson = await response.json();
80     } catch (err) {
81         // handle case if json is invalid
82         throw [{
83             status: INTERNAL_SERVER_ERROR,
84             message: 'something went wrong'
85         }];
86     }
87
88     if (response.ok) {
89         const { attemptId } = responseJson;
90
91         return Promise.resolve(attemptId);
92     } else {
93         throw responseJson;
94     }
95 };

```

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

```

1 import {BAD_REQUEST, FORBIDDEN} from "http-status-codes";
2 import {appendAuth} from "../helpers/header";
3 import {getToken} from "../helpers/token";
4
5 const remove = async () => {
6     const token = getToken();
7     const headers = new Headers();
8
9     appendAuth(headers, token);
10
11     const response = await fetch("/api/profile/remove", {
12         headers,
13         method: 'POST'
14     });
15
16     if (response.ok) {
17         return Promise.resolve();
18     } else {
19         switch (response.status) {
20             case FORBIDDEN:
21                 return Promise.reject();
22             default:
23                 break;

```

```

24     }
25   }
26 };
27
28 const updatePassword = async (formData) => {
29   const token = getToken();
30   const headers = new Headers();
31
32   appendAuth(headers, token);
33
34   const response = await fetch("/api/profile/update-password", {
35     method: 'POST',
36     headers,
37     body: formData
38   });
39
40   if (response.ok) {
41     return Promise.resolve();
42   } else {
43     const responseJson = await response.json();
44
45     switch (response.status) {
46       case BAD_REQUEST:
47         return Promise.reject(responseJson);
48       default:
49         break;
50     }
51   }
52 };
53
54 export { remove, updatePassword };

```

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

```

1 import {appendAuth, appendJSON} from "../helpers/header";
2 import {getToken} from "../helpers/token";
3
4 export const deleteTest = async testId => {
5   const token = getToken();
6   const headers = new Headers();
7
8   appendAuth(headers, token);
9   appendJSON(headers);
10
11   const response = await fetch(`/api/test/delete`, {
12     method: 'DELETE',
13     headers,
14     body: JSON.stringify({ testId })
15   });
16
17   if (response.ok) {
18     return Promise.resolve();
19   } else {
20     // TODO: handle if something went wrong
21     return Promise.reject();
22   }
23 };
24
25 export const getOwnTests = async () => {
26   const token = getToken();
27   const headers = new Headers();
28
29   appendAuth(headers, token);
30
31   const response = await fetch('/api/test/profile', {
32     method: 'GET',
33     headers
34   });
35
36   if (response.ok) {
37     const responseJson = await response.json();
38
39     return Promise.resolve(responseJson);
40   } else {

```

```

41     // TODO: handle if something wrong
42     return Promise.reject();
43 }
44 };
45
46 export const create = async (testData) => {
47     const token = getToken();
48     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 update = async (testData, 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: 'PUT',
77         body: JSON.stringify({ ...testData, testId }),
78         headers,
79     });
80
81     if (response.ok) {
82         return Promise.resolve();
83     } else {
84         const responseJson = await response.json();
85
86         return Promise.reject(responseJson);
87     }
88 };
89
90 export const getTestForEdit = async testId => {
91     const token = getToken();
92     const headers = new Headers();
93
94     appendAuth(headers, token);
95     appendJSON(headers);
96
97     const response = await fetch('/api/test/update', {
98         method: 'POST',
99         body: JSON.stringify({ testId }),
100        headers,
101    });
102
103    const responseJson = await response.json();
104    if (response.ok) {
105        return Promise.resolve(responseJson);
106    } else {
107        return Promise.reject();
108    }
109 };
110
111 /**
112  * Get all tests
113  * @return {Promise<Object[]>}
114  */
115 export const getAllTests = async () => {
116     const token = getToken();

```

```

117     const headers = new Headers();
118
119     appendAuth(headers, token);
120
121     const response = await fetch('/api/test/all', {
122         method: 'GET',
123         headers
124     });
125
126     const responseJson = await response.json();
127
128     return Promise.resolve(responseJson);
129 };

```

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 PropTypes 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: PropTypes.bool,
20     component: PropTypes.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     const isElectronApp = req.header('user-agent').includes('Electron');
7
8     res.render(isElectronApp ? "contest" : "index");
9 });
10
11 export default router;

```

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

```

1 import { Router } from "express";
2 import * as attemptsController from "../controllers/attempt";
3 import checkToken from "../middlewares/checkToken";
4
5 const router = new Router();
6
7 router.get("/profile", checkToken, attemptsController.getOwnAttempts);
8 router.get("/test", checkToken, attemptsController.getOwnTestAttempts);
9
10 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();
7
8 const router = new Router();
9
10 router.post('/signup', upload.none(), authController.signUp);
11 router.post('/signin', upload.none(), authController.signIn);
12 router.post('/init', checkToken, authController.initAuth);
13
14 export default router;

```

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

```
1 import Router from "express";
2 import multer from "multer";
3 import checkToken from "../middlewares/checkToken";
4 import * as profileModify from "../controllers/profileModify";
5
6 const upload = multer();
7 const router = new Router();
8
9 router.post("/update-password", upload.none(), checkToken, profileModify.updatePassword);
10 router.post("/remove", checkToken, profileModify.remove);
11
12 export default router;
```

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

```
1 import { Router } from "express";
2 import * as testEditor from "../controllers/testEditor";
3 import checkToken from "../middlewares/checkToken";
4
5 const router = new Router();
6
7 router.post("/create", checkToken, testEditor.create);
8 router.put("/update", checkToken, testEditor.update);
9 router.post("/update", checkToken, testEditor.getTestForEdit);
10 router.get("/pass", checkToken, testEditor.getTestForPassing);
11 router.get("/result", checkToken, testEditor.getAttemptResults);
12 router.post("/check", checkToken, testEditor.check);
13 router.get("/profile", checkToken, testEditor.getOwnTests);
14 router.get("/all", checkToken, testEditor.getAllTests);
15 router.delete("/delete", checkToken, testEditor.deleteTest);
16
17 export default router;
```

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

```
1 describe("Sum", () => {
2   it("addition", () => {
3     expect(6).toBe(6);
4   });
5 });
```

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

```
1 import Attempt from "../models/Attempt";
2 import Test from "../models/Test";
3 import User from "../models/User";
4
5 export const getOwnAttempts = async (req, res) => {
6   const { userId } = req;
7
8   const ownAttempts = await Attempt.findAll({
9     where: { userId },
10    include: [Test]
11  });
12
13  // collect response
14  const response = [];
15  for (let attempt of ownAttempts) {
16    const { createdAt, result } = attempt;
17    const { title, id: testId } = attempt.test;
18
19    response.push({
20      title,
21      result,
```

```

22         testId,
23         date: createdAt,
24     });
25 }
26
27 res.json(response);
28 };
29
30 export const getOwnTestAttempts = async (req, res) => {
31     let { id } = req.query;
32     const testId = parseInt(id);
33
34     const ownAttempts = await Attempt.findAll({
35         where: { testId },
36         include: [User]
37     });
38
39     // collect response
40     const response = [];
41     for (let attempt of ownAttempts) {
42         const { createdAt, result, id } = attempt;
43         const { login } = attempt.user;
44
45         response.push({
46             login,
47             result,
48             attemptId: id,
49             date: createdAt,
50         });
51     }
52
53     res.json(response);
54 };

```

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

```

1 import {
2     INTERNAL_SERVER_ERROR,
3     BAD_REQUEST,
4     OK, FORBIDDEN
5 } from "http-status-codes";
6 import * as jwt from "jsonwebtoken";
7 import FormListErrors from "../helpers/FormListErrors";
8 import User from "../models/User";
9
10 export const signUp = async (req, res, next) => {
11     const {
12         login,
13         repeatPassword,
14         password,
15         email,
16     } = req.body;
17     const formListErrors = new FormListErrors();
18
19     if (repeatPassword !== password) {
20         formListErrors.add("passwords doesn't equal.");
21     }
22
23     next({
24         status: BAD_REQUEST,
25         errors: formListErrors.data.errors
26     });
27
28     try {
29         await User.create({ email, login, password });
30     } catch ({ errors }) {
31         formListErrors.addFromModelErrors(errors);
32     }
33
34     next({
35         status: BAD_REQUEST,
36         errors: formListErrors.data.errors
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, next) => {
90   const user = await User.findByPk(req.userId);
91
92   if (!user) {
93     next({
94       status: FORBIDDEN,
95       errors: [
96         { message: 'user with such id not found' }
97       ]
98     });
99   }
100
101   res.json({ ...user.initState() });
102 };

```

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

```

1 import User from "../models/User";
2 import {BAD_REQUEST, OK} from "http-status-codes";
3 import FormListErrors from "../helpers/FormListErrors";
4
5 export const updatePassword = async (req, res, next) => {
6   const formListErrors = new FormListErrors();
7   const { userId } = req;

```

```

8     const { password, newPassword, repeatNewPassword } = req.body;
9
10    const user = await User.findByPk(userId);
11
12    const isRightPassword = await user.comparePasswords(password);
13
14    if (isRightPassword) {
15        if (repeatNewPassword !== newPassword) {
16            formListErrors.add("passwords doesn't equal.");
17
18            next({
19                status: BAD_REQUEST,
20                errors: formListErrors.data.errors
21            });
22        }
23
24        try {
25            await user.update({ password: newPassword });
26        } catch ({ errors }) {
27            formListErrors.addFromModelErrors(errors);
28
29            next({
30                status: BAD_REQUEST,
31                errors: formListErrors.data.errors
32            });
33        }
34    } else {
35        formListErrors.add('current password is invalid. ');
36
37        next({
38            status: BAD_REQUEST,
39            errors: formListErrors.data.errors
40        });
41    }
42
43    res.sendStatus(OK);
44 };
45
46 export const remove = async (req, res) => {
47     const { userId } = req;
48
49     const user = await User.findByPk(userId);
50
51     await user.destroy();
52
53     res.sendStatus(OK);
54 };

```

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

```

1  import {
2      BAD_REQUEST,
3      FORBIDDEN,
4      INTERNAL_SERVER_ERROR,
5      NOT_FOUND,
6      OK
7  } from "http-status-codes";
8  import Test from "../models/Test";
9  import User from "../models/User";
10 import Tag from "../models/Tag";
11 import TestTag from "../models/TestTag";
12 import Attempt from "../models/Attempt";
13 import FormListErrors from "../helpers/FormListErrors";
14 import { zipWith, zip } from "lodash";
15
16 export const update = async (req, res, next) => {
17     const { userId } = req;
18     const { info, testId } = req.body;
19     const content = req.body.questions;
20     const { title, description, tags } = info;
21     const formListErrors = new FormListErrors();
22
23
24     const currTest = await Test.findByPk(testId, {

```

```

25     include: [Tag]
26   });
27
28   if (!currTest) {
29     formListErrors.addDefault();
30
31     next({
32       status: INTERNAL_SERVER_ERROR,
33       errors: formListErrors.data.errors
34     });
35   }
36
37   if (userId !== currTest.userId) {
38     formListErrors.addDefault();
39
40     next({
41       status: FORBIDDEN,
42       errors: formListErrors.data.errors
43     });
44   }
45
46   try {
47     // update current test
48     await currTest.update({
49       title,
50       description,
51       content,
52     });
53
54   } catch (err) {
55     formListErrors.addFromModelErrors(err.errors);
56
57     next({
58       status: BAD_REQUEST,
59       errors: formListErrors.data.errors
60     });
61   }
62
63   // create new tags
64   const currTestTags = [];
65   try {
66     for (let tagName of tags) {
67       const [newTag] = await Tag.findOrCreate({
68         where: { name: tagName }
69       });
70
71       currTestTags.push(newTag);
72     }
73   } catch (err) {
74     formListErrors.addFromModelErrors(err.errors);
75
76     next({
77       status: BAD_REQUEST,
78       errors: formListErrors.data.errors
79     });
80   }
81
82   const ownTestTags = await TestTag.findAll({
83     where: {
84       testId
85     },
86     include: [Tag]
87   });
88
89   // find and delete tags that don't need anymore
90   const notExistTagIds = ownTestTags.filter(tag => !tags.includes(tag.name))
91     .map(tag => tag.tagId);
92
93   await TestTag.destroy({
94     where: {
95       testId: currTest.id,
96       tagId: notExistTagIds
97     },
98     force: true
99   });
100

```

```

101 // update link from tags to tests
102 for (let currTag of currTestTags) {
103   await TestTag.findOrCreate({
104     where: {
105       testId: currTest.id,
106       tagId: currTag.id
107     }
108   });
109 }
110
111 res.sendStatus(OK);
112 };
113
114 export const create = async (req, res, next) => {
115   const { userId } = req;
116   const { info } = req.body;
117   const content = req.body.questions;
118   const { title, description, tags } = info;
119   const formListErrors = new FormListErrors();
120
121   // create new test
122   let newTest = null;
123   try {
124     newTest = await Test.create({
125       title,
126       description,
127       content,
128       userId,
129     }, { include: [Tag] });
130   } catch (err) {
131     formListErrors.addFromModelErrors(err.errors);
132
133     next({
134       status: BAD_REQUEST,
135       errors: formListErrors.data.errors
136     });
137
138     return;
139   }
140
141   // create or find tags
142   const currTestTags = [];
143   try {
144     for (let currTagName of tags) {
145       const [newTag] = await Tag.findOrCreate({
146         where: {
147           name: currTagName
148         }
149       });
150
151       currTestTags.push(newTag);
152     }
153   } catch (err) {
154     formListErrors.addFromModelErrors(err.errors);
155
156     next({
157       status: BAD_REQUEST,
158       errors: formListErrors.data.errors
159     });
160   }
161
162   // create link from tags to tests
163   for (let currTag of currTestTags) {
164     await TestTag.findOrCreate({
165       where: {
166         testId: newTest.id,
167         tagId: currTag.id
168       }
169     });
170   }
171
172   res.sendStatus(OK);
173 };
174
175 export const getOwnTests = async (req, res) => {
176   const { userId } = req;

```

```

177
178   const tests = await Test.findAll({
179     where: { userId },
180     include: [User, Tag]
181   });
182
183   const response = [];
184   for (let test of tests) {
185     const { title, description, id } = test;
186     const { login } = test.user;
187     const tags = await test.getTags();
188
189     response.push({
190       title,
191       description,
192       tags: tags.map((tag => tag.name)),
193       author: login,
194       testId: id
195     });
196   }
197
198   res.json(response);
199 };
200
201 export const getAllTests = async (req, res) => {
202   const tests = await Test.findAll({
203     include: [User, Tag]
204   });
205
206   const response = [];
207   for (let test of tests) {
208     const { title, description, id } = test;
209     const { login } = test.user;
210     const tags = await test.getTags();
211
212     response.push({
213       title,
214       description,
215       tags: tags.map((tag => tag.name)),
216       author: login,
217       testId: id
218     });
219   }
220
221   res.json(response);
222 };
223
224 export const deleteTest = async (req, res, next) => {
225   const { testId } = req.body;
226   const { userId } = req;
227   const formListErrors = new FormListErrors();
228
229   const test = await Test.findByPk(testId, {
230     include: User
231   });
232
233   if (!test) {
234     formListErrors.addDefault();
235
236     next({
237       status: BAD_REQUEST,
238       errors: formListErrors.data.errors
239     });
240   }
241
242   if (test.userId === userId) {
243     await Test.destroy({
244       where: {
245         id: testId
246       }
247     });
248
249     res.sendStatus(OK);
250   } else {
251     formListErrors.addDefault();
252

```

```

253     next({
254       status: FORBIDDEN,
255       errors: formListErrors.data.errors
256     });
257   }
258 };
259
260 export const getTestForEdit = async (req, res, next) => {
261   const { testId } = req.body;
262   const { userId } = req;
263   const formListErrors = new FormListErrors();
264
265   const test = await Test.findByPk(testId, {
266     include: [User, Tag]
267   });
268
269   if (!test) {
270     formListErrors.add('test not found');
271
272     next({
273       status: NOT_FOUND,
274       errors: formListErrors.data.errors
275     });
276   }
277
278   if (test.userId === userId) {
279     const tags = await test.getTags();
280     const { title, description, content } = test;
281
282     res.json({
283       info: {
284         title,
285         description,
286         tags: tags.map(tag => tag.name)
287       },
288       questions: content
289     });
290   } else {
291     formListErrors.addDefault();
292
293     next({
294       status: FORBIDDEN,
295       errors: formListErrors.data.errors
296     });
297   }
298 };
299
300 export const getTestForPassing = async (req, res, next) => {
301   let { id } = req.query;
302   const testId = parseInt(id);
303   // const { userId } = req;
304   const formListErrors = new FormListErrors();
305
306   const test = await Test.findByPk(testId, { include: [User] });
307
308   if (!test) {
309     formListErrors.add('test not found');
310
311     next({
312       status: NOT_FOUND,
313       errors: formListErrors.data.errors
314     });
315   }
316
317   // exclude from answers 'isRight' and add 'isChecked' properties
318   const questions = test.content.map(question => {
319     const { answers } = question;
320
321     const modifiedAnswers = answers.map(answer => {
322       return {
323         content: answer.content,
324         isChecked: false
325       };
326     });
327
328     return { ...question, answers: modifiedAnswers };

```



```

329     });
330
331     res.json(questions);
332   };
333
334   export const check = async (req, res, next) => {
335     const { questions: userQuestions, testId } = req.body;
336     const { userId } = req;
337     const formListErrors = new FormListErrors();
338
339     const test = await Test.findByPk(testId, { include: [User] });
340
341     if (!test) {
342       formListErrors.add('test not found');
343
344       next({
345         status: NOT_FOUND,
346         errors: formListErrors.data.errors
347       });
348     }
349
350     // count and check answers
351     let amountRightQuestions = 0;
352     const amountQuestions = test.content.length;
353     const userAnswersStates = [];
354     zipWith(test.content, userQuestions, (testQuestion, userQuestion) => {
355       const { answers: testAnswers } = testQuestion;
356       const { answers: userAnswers } = userQuestion;
357
358       const isRight = zip(testAnswers, userAnswers).every(
359         ([testAnswer, userAnswer]) => testAnswer.isRight === userAnswer.isChecked
360       );
361
362       if (isRight) {
363         amountRightQuestions++;
364       }
365
366       userAnswersStates.push({ isCorrect: isRight });
367     });
368
369     const result = amountRightQuestions / amountQuestions;
370
371     let newAttempt = null;
372     try {
373       newAttempt = await Attempt.create({
374         result,
375         userId,
376         testId,
377         answers: userAnswersStates
378       }, {
379         include: [User, Test]
380       });
381     } catch (err) {
382       // TODO: handle case if data is invalid
383
384       console.error(err);
385     }
386
387     res.json({ attemptId: newAttempt.id });
388   };
389
390   export const getAttemptResults = async (req, res, next) => {
391     let { id } = req.query;
392     const attemptId = parseInt(id);
393     const formListErrors = new FormListErrors();
394
395     const attempt = await Attempt.findByPk(attemptId);
396
397     if (!attempt) {
398       formListErrors.add('attempt not found');
399
400       next({
401         status: NOT_FOUND,
402         errors: formListErrors.data.errors
403       });
404     }

```

```
405
406   res.json({ userAnswers: attempt.answers });
407 };
```

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

```
1  /**
2   * Class representing a list of errors.
3   */
4  export default class FormListErrors {
5    /**
6     * Create error list
7     */
8    constructor() {
9      this.data = { errors: [] };
10   }
11
12   /**
13    * Add errors from ORM model
14    * @param {Array<ValidationErrorItem>} errors
15    */
16   addFromModelErrors(errors) {
17     this.data.errors.push(
18       ...errors.map(err => {
19         let { message } = err;
20
21         return { message };
22       })
23     );
24   }
25
26   /**
27    * Add default error message
28    */
29   addDefault() {
30     this.add("Oops, something went wrong.");
31   }
32
33   /**
34    * Add custom error message to list
35    * @param {string} message - Error message
36    */
37   add(message) {
38     this.data.errors.push({ message });
39   }
40
41   /**
42    * Get error list length
43    * @return {number}
44    */
45   isEmpty() {
46     return this.data.errors.length;
47   }
48 }
```

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

```
1 <main id="root">
2   <!-- React entry point -->
3 </main>
4
5 <script src="/static/contest.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;
15     }
16
17     .loader__dot {
18         width: 10px;
19         height: 10px;
20         display: inline-block;
21         border-radius: 100%;
22         background-color: #000;
23         transition: opacity .4s;
24         animation: .6s linear 0s infinite alternate fade-dot;
25     }
26
27     .loader__dot:nth-child(1) {
28         animation-delay: .2s;
29     }
30
31     .loader__dot:nth-child(2) {
32         animation-delay: .4s;
33     }
34
35     .loader__dot:nth-child(3) {
36         animation-delay: .6s;
37     }
38
39     @keyframes fade-dot {
```

```

40     from {
41         opacity: 100%;
42     }
43
44     to {
45         opacity: 0;
46     }
47 }
48 </style>
49
50 <div class="loader">
51     <div class="loader__dots">
52         <div class="loader__dot"></div>
53         <div class="loader__dot"></div>
54         <div class="loader__dot"></div>
55     </div>
56 </div>

```

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

```

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

```

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

```

1 import * as jwt from "jsonwebtoken";
2 import {FORBIDDEN} from "http-status-codes";
3
4 export default async (req, res, next) => {
5     const token = req.headers["authorization"];
6
7     let tokenObj = null;
8     try {
9         tokenObj = await jwt.verify(token, process.env.JWT_SECRET);
10    } catch (err) {
11        next({
12            status: FORBIDDEN,
13            errors: [{
14                message: "something went wrong"
15            }]
16        });
17    }
18
19    req.userId = tokenObj.sub;
20
21    next();
22 };

```

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

```

1 import { INTERNAL_SERVER_ERROR } from "http-status-codes";
2
3 // eslint-disable-next-line no-unused-vars
4 export default function(err, req, res, next) {
5     let { status, errors } = err;
6
7     if (!status) {
8         status = INTERNAL_SERVER_ERROR;
9     }
10
11    res.status(status).json({ errors });
12 }

```

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

```

1 import { Sequelize } from "sequelize";
2
3 import config from "../config";
4
5 import Test from "./Test";
6 import Tag from "./Tag";
7
8 const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
9
10 // setup many to many
11 const TestTag = sequelize.define("test_tag", {}, {
12   timestamps: false
13 });
14
15 Test.belongsToMany(Tag, { through: TestTag });
16 Tag.belongsToMany(Test, { through: TestTag });
17
18 Test.hasMany(TestTag);
19 TestTag.belongsTo(Test);
20 Tag.hasMany(TestTag);
21 TestTag.belongsTo(Tag);
22
23 export default TestTag;

```

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

```

1 import Test from "./Test";
2 import User from "./User";
3 import Tag from "./Tag";
4 import TestTag from "./TestTag";
5 import Attempt from "./Attempt";
6
7 const models = [
8   Test,
9   User,
10  Tag,
11  TestTag,
12  Attempt
13 ];
14
15 export default models;

```

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

```

1 const MIN_TAG_NAME_LENGTH = 1;
2 const MAX_TAG_NAME_LENGTH = 16;
3
4 export {
5   MIN_TAG_NAME_LENGTH,
6   MAX_TAG_NAME_LENGTH
7 }

```

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

```

1 import Tag from "./Tag";
2
3 export default Tag;

```

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

```

1 import * as DataTypes from "sequelize";
2 import { Sequelize } from "sequelize";
3 import config from "../../config";
4
5 import * as tagConstraints from "../constraints";
6

```

```

7 const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
8
9 const Tag = sequelize.define('tag', {
10   id: {
11     type: DataTypes.INTEGER,
12     primaryKey: true,
13     autoIncrement: true
14   },
15   name: {
16     type: DataTypes.STRING,
17     unique: true,
18     allowNull: false,
19     validate: {
20       len: {
21         msg: `tag must has length between ${tagConstraints.MIN_TAG_NAME_LENGTH} and
22         ↳ ${tagConstraints.MAX_TAG_NAME_LENGTH}.`,
23         args: [
24           tagConstraints.MIN_TAG_NAME_LENGTH,
25           tagConstraints.MAX_TAG_NAME_LENGTH
26         ]
27       }
28     }
29   }, {
30     timestamps: false
31   });
32
33 export default Tag;

```

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

```

1 const MIN_TITLE_LENGTH = 1;
2 const MAX_TITLE_LENGTH = 128;
3 const MIN_DESCRIPTION_LENGTH = 1;
4 const MAX_DESCRIPTION_LENGTH = 256;
5
6 export {
7   MIN_TITLE_LENGTH,
8   MAX_TITLE_LENGTH,
9   MIN_DESCRIPTION_LENGTH,
10  MAX_DESCRIPTION_LENGTH
11 };

```

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

```

1 import Test from "../Test";
2
3 export default Test;

```

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

```

1 import * as DataTypes from "sequelize";
2 import { Sequelize } from "sequelize";
3 import config from "../../config";
4 import validate from "validate.js";
5 import {ANSWER_TYPE} from "../../config";
6
7 import {
8   MIN_TITLE_LENGTH,
9   MAX_TITLE_LENGTH,
10  MIN_DESCRIPTION_LENGTH,
11  MAX_DESCRIPTION_LENGTH
12 } from "../constraints";
13
14 const testScheme = {
15   title: {
16     type: "string",
17     presence: true,
18     length: {

```

```

19         minimum: 1,
20         tooShort: "of question needs to be not empty"
21     },
22 },
23 typeAnswer: {
24     type: "string",
25     presence: true,
26     inclusion: [ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE],
27     length: { minimum: 1 }
28 },
29 answers: {
30     type: "array",
31     presence: true,
32     length: {
33         tooShort: "needs to have minimum 2",
34         minimum: 2
35     }
36 }
37 };
38
39 const answerScheme = {
40     content: {
41         type: "string",
42         length: { minimum: 1, tooShort: "of answer needs to have not empty length" },
43         presence: true
44     },
45     isRight: {
46         type: "boolean",
47         presence: true
48     }
49 };
50
51 const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
52
53 const Test = sequelize.define("test", {
54     id: {
55         type: Sequelize.INTEGER,
56         autoIncrement: true,
57         primaryKey: true
58     },
59     title: {
60         type: Sequelize.STRING,
61         allowNull: false,
62         unique: true,
63         validate: {
64             len: {
65                 msg: `title must has length between ${MIN_TITLE_LENGTH} and ${MAX_TITLE_LENGTH}.`,
66                 args: [
67                     MIN_TITLE_LENGTH,
68                     MAX_TITLE_LENGTH
69                 ]
70             }
71         }
72     },
73     description: {
74         type: Sequelize.STRING,
75         allowNull: false,
76         validate: {
77             len: {
78                 msg: `description must has length between ${MIN_DESCRIPTION_LENGTH} and
79 ↪ ${MAX_DESCRIPTION_LENGTH}.`,
80                 args: [
81                     MIN_DESCRIPTION_LENGTH,
82                     MAX_DESCRIPTION_LENGTH
83                 ]
84             }
85         }
86     },
87     content: {
88         type: DataTypes.JSON,
89         allowNull: false,
90         validate: {
91             isValidContent: function(value) {
92                 for (let currQuestion of value) {
93                     const resultCheck = validate(currQuestion, testScheme);

```

```

94         if (resultCheck !== undefined) {
95             const firstErrorMsg = Object.values(resultCheck)[0][0];
96
97             throw new Error(firstErrorMsg);
98         }
99     }
100 },
101 isValidAnswers: function (value) {
102     for (let currQuestion of value) {
103         for (let currAnswer of currQuestion.answers) {
104             const resultCheck = validate(currAnswer, answerScheme);
105
106             if (resultCheck !== undefined) {
107                 const firstErrorMsg = Object.values(resultCheck)[0][0];
108
109                 throw new Error(firstErrorMsg);
110             }
111         }
112     }
113 }
114 }
115 }
116 });
117
118 export default Test;

```

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

```

1 export default {
2     // Password
3     MIN_PASSWORD_LENGTH: 10,
4     MAX_PASSWORD_LENGTH: 128,
5
6     // Login
7     MIN_LOGIN_LENGTH: 6,
8     MAX_LOGIN_LENGTH: 128,
9
10    // Email
11    MAX_EMAIL_LENGTH: 320
12 };

```

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

```

1 import { Sequelize } from "sequelize";
2 import * as bcrypt from "bcrypt";
3
4 import config from "../config";
5 import userConstraints from "../constraints";
6 import Test from "../Test";
7
8 const {
9     MIN_PASSWORD_LENGTH,
10    MAX_PASSWORD_LENGTH,
11    MIN_LOGIN_LENGTH,
12    MAX_LOGIN_LENGTH,
13    MAX_EMAIL_LENGTH
14 } = userConstraints;
15
16 const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
17
18 const User = sequelize.define("user", {
19     id: {
20         type: Sequelize.INTEGER,
21         autoIncrement: true,
22         primaryKey: true,
23     },
24     login: {
25         type: Sequelize.STRING(MAX_LOGIN_LENGTH),
26         allowNull: false,
27         unique: true,
28         validate: {
29             len: {

```



```

30         msg: `login must has length between ${MIN_LOGIN_LENGTH} and ${MAX_LOGIN_LENGTH}.`,
31         args: [
32             MIN_LOGIN_LENGTH,
33             MAX_LOGIN_LENGTH
34         ]
35     }
36 }
37 },
38 password: {
39     type: Sequelize.STRING(MAX_PASSWORD_LENGTH),
40     allowNull: false,
41     validate: {
42         len: {
43             msg: `password must has length between ${MIN_PASSWORD_LENGTH} and
↵  ${MAX_PASSWORD_LENGTH}.`,
44             args: [
45                 MIN_PASSWORD_LENGTH,
46                 MAX_PASSWORD_LENGTH
47             ]
48         }
49     }
50 },
51 email: {
52     type: Sequelize.STRING(MAX_EMAIL_LENGTH),
53     unique: true,
54     allowNull: false,
55     validate: {
56         isEmail: {
57             msg: 'invalid email address.'
58         }
59     }
60 },
61 }
62 );
63
64 User.hashPassword = async (value) => {
65     const salt = await bcrypt.genSalt(10);
66
67     return await bcrypt.hash(value, salt);
68 };
69
70 User.prototype.comparePasswords = async function(password) {
71     return await bcrypt.compare(password, this.password);
72 };
73
74 User.prototype.initState = function() {
75     const { login } = this;
76
77     return {
78         user: { login }
79     }
80 };
81
82 User.beforeCreate(async user => {
83     user.password = await User.hashPassword(user.password);
84 });
85
86 User.beforeUpdate(async user => {
87     user.password = await User.hashPassword(user.password);
88 });
89
90 User.hasMany(Test, { onDelete: 'cascade' });
91 Test.belongsTo(User);
92
93 export default User;

```

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

```

1 import User from "../User";
2
3 export default User;

```

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

```
1 import * as DataTypes from "sequelize";
2 import { Sequelize } from "sequelize";
3 import config from "../../config";
4 import User from "../User";
5 import Test from "../Test";
6
7 const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
8
9 const Attempt = sequelize.define("attempt", {
10   id: {
11     type: DataTypes.INTEGER,
12     primaryKey: true,
13     autoIncrement: true
14   },
15   result: {
16     type: DataTypes.FLOAT,
17     validate: {
18       min: 0.0,
19       max: 1.0
20     }
21   },
22   answers: {
23     type: DataTypes.JSON,
24   }
25 });
26
27 // create associations
28 User.hasMany(Attempt, { onDelete: 'cascade' });
29 Attempt.belongsTo(Test, { onDelete: 'cascade' });
30 Attempt.belongsTo(User, { onDelete: 'cascade' });
31
32 Test.afterUpdate(async test => {
33   // handle case if test is updated
34
35   await Attempt.destroy({
36     where: {
37       testId: test.id
38     }
39   });
40 });
41
42 export default Attempt;
```

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

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

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

```
1 CREATE TABLE IF NOT EXISTS users (
2   user_id SERIAL NOT NULL,
3   login VARCHAR(255) NOT NULL UNIQUE,
4   password VARCHAR(255) NOT NULL,
5   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>.