Министерство образования и науки РФ

Московский Авиационный институт
(Национальный исследовательский университет)

Институт №8
«Информационные технологии и прикладная математика»

Кафедра 813
«Компьютерная математика»

Курсовой проект по дисциплине «Базы данных»

Тема: «Веб-приложение для тестирования»

Студент: Васильев Дмитрий Олегович

Группа: М8О-310Б-18

Преподаватель: Романенков Александр Михайлович

Дата: 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 Контольная тестирующая система 5

1.3 Предметная область 5

1.4 Стэк технологий 5

1.5 Инструменты 6

2 Инфраструктура проекта **7**

2.1 Архитекрутра 7

2.1.1 Связь логических компонетов и применяемых технологих 8

2.2 Сущности 8

2.3 Сборка и запуск 9

2.3.1 Development 9

2.3.2 Production 9

2.4 Деплоинг 10

2.5 Организация работы с *Git* [5] 10

2.5.1 Git Workflow 10

2.5.2 Git hooks 10

3 Описание проекта **12**

3.1 Разработка дизайна 12

3.2 Авторизации через JSON Web Token (JWT) 12

3.2.1 Авторизация 12

3.2.2 Аутентификация 12

3.2.3 JSON Web Token (JWT) 12

3.2.4 Реализация 13

3.3 Схема базы данных 14

3.3.1 Связи 14

3.3.2 Хуки (Триггеры) 15

3.3.3 Процедуры (методы модели) 15

3.3.4 Каскадное удаление 16

4 Заключение **17**

4.1 Недостатки 17

A Визуализации структуры проекта **18**

B Код проекта **26**

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

Один клиент будет SPA веб-приложение. Другой десктоп на Electron. Оба общаются с сервером посредством REST API и HTTP протокола.

1.2.1 Тестирующая система

Данное приложение даёт возможность пользователю:

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

1.2.2 Контольная тестирующая система

Данное приложение нужно для прохождения тестов в очном режиме. Оно является урезанной версий предыдущего клиента.

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* [13] — *JavaScript* [8] библиотека для создания пользовательских интерфейсов.
- *Redux* [15] — контейнер состояния для *JavaScript* [8] приложения.
- *React Router* [14] — набор навигационных компонентов.
- *SCSS* [16] — 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.
- *PostgreSQL* [12] — объектно-реляционная база данных с открытым исходным кодом.
- *Sequelize* [17] — *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* [18] — проверяет качество кода на *SCSS* [16], *CSS* [1].
- Тестирующие фреймворки:
  - *Jest* [9] — среда тестирования *JavaScript* [8] с упором на простоту.

  Webpack [19] — сборщик статических модулей для современных *JavaScript* [8] приложений.

2 Инфраструктура проекта

2.1 Архитекрутра

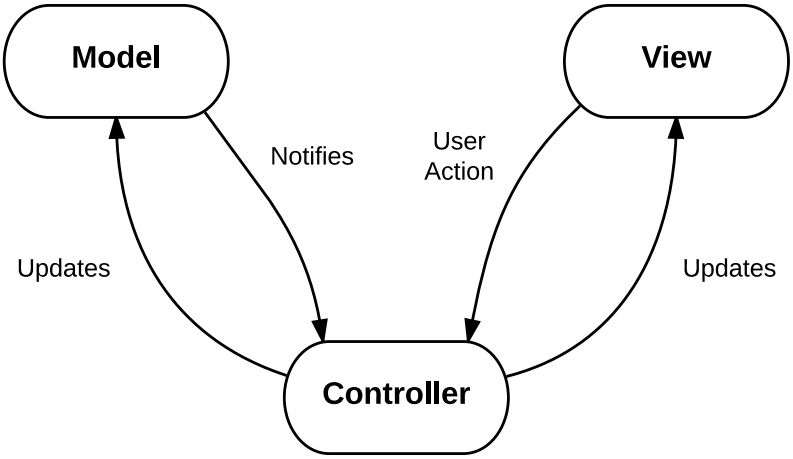

Рис. 1: Визуализация архитектуры MVP

За основу берётся архитектурный паттерн MVP. Он предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: Модель, Представление и Презентер – 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 презентера, 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.

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

2.1.1 Связь логических компонетов и применяемых технологих

- **Model** — *Sequelize* [17] для сервера и *Redux* [15] для клиента.
- **View** — *React* [13]
- **Presenter** — *Express* [4]

2.2 Сущности

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

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

Попытка	• 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 npm скриптом. Они все описываются в файле package.json. Также среди данных скриптов можно выделить две группы – Development и Production.

2.3.1 Development

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

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

2.3.2 Production

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

1. минификация статический файлов
2. оптимизация работы библиотек

3. сборка серверной части

2.4 Деплоинг

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

2.5 Организация работы с *Git* [5]

2.5.1 Git Workflow

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

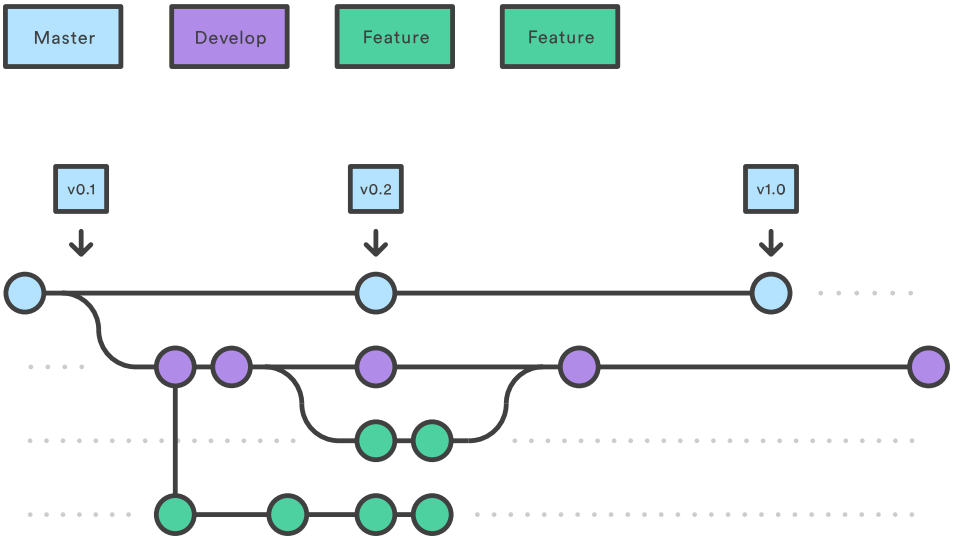

Рис. 2: Пример использования подхода Git Workflow

2.5.2 Git hooks

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

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

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

Листинг 1: Настройки для Git Hooks

3 Описание проекта

3.1 Разработка дизайна

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

3.2 Авторизации через JSON Web Token (JWT)

3.2.1 Авторизация

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

3.2.2 Аутентификация

Аутентификация — процедура проверки подлинности данный.

3.2.3 JSON Web Token (JWT)

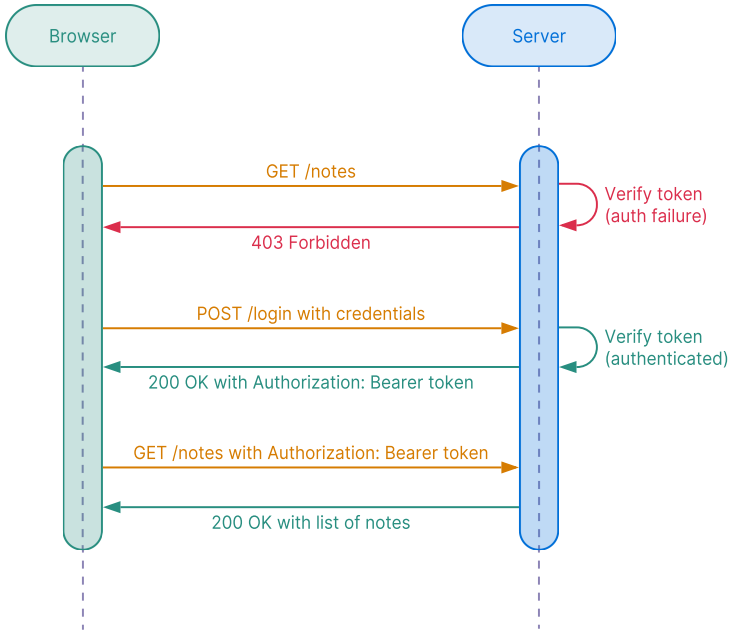

Рис. 3: Демонстрация работы JWT

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

Вот несколько сценариев, в которых полезен JWT:

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

JWT состоит из следующих частей:

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

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

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

3.2.4 Реализация

Авторизация проходит следущим образом:

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

2. Пользователь делает `POST /api/signin` запрос на аутентификацию. Если данные верные, то высылается JWT вместе с состоянием пользователя (логин, почта). Когда ответ с сервера получен, то JWT сохраняется в localStorage (долговременное хранилище), а состояние передается в глобальное *Redux* [15] хранилище.

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

Если перезагрузить веб-страницу, то *Redux* [15] хранилище обнуляется. Поэтому нам нужно сделать следующее: при запуске приложения проверять на валидность JWT, который лежит в localStorage. Это делается через `POST /api/init` запрос. Если токен валидный, то переавторизовываем пользователя. Иначе перенаправляем на главную страницу.

Далее этот токет будет использоваться для доступа к защищёнными ресурсам.

3.3 Схема базы данных

3.3.1 Связи

- (users) $1 : N$ (tests)
- (tests) $M : N$ (tags)
- (attempts) $1 : 1$ (tests)
- (attempts) $1 : 1$ (users)

Примечание:

- $1 : N$ – один ко многим
- $M : N$ – многие ко многим
- $1 : 1$ – один к одному

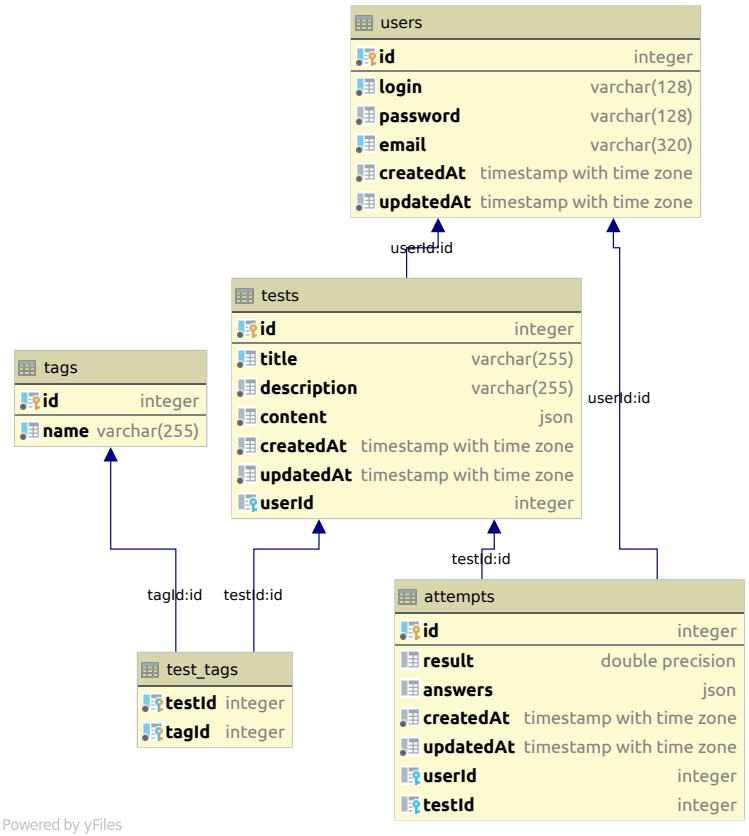

Рис. 4: Схема базы данных

3.3.2 Хуки (Триггеры)

Для некоторых таблиц заданы хуки, которые позволяют обработать события связанные с вставкой, обновлением, удалением и т.д.

- `User`:
  - `beforeCreate` – шифрует пароль перед вставкой в таблицу.
  - `beforeUpdate` – шифрует пароль перед обновлением.
- `Attempt`:
  - `afterUpdate` – удаляет все попытки у теста, если он был изменён.
- `TestTag`:
  - `afterDelete` – удаляет теги, которое не используются.

3.3.3 Процедуры (методы модели)

Для модели `User` есть две процедуры:

- `comparePasswords(password: string)` – инстанс метод, который стравнивает пароль текущего пользователя с `password`.
- `hashPassword(password: string)` – статический метод, который шифрует `password`.

3.3.4 Каскадное удаление

Каскадное удаление в мире реляционных баз данных позволяет удалять связанные данные из зависимой таблицы, при удалении данных из основной таблицы. У меня оно использую во всех таблицах.

4 Заключение

Благодаря данному курсовому проекту, я поверхностно освоил разработку SPA приложений с простой JWT авторизации. Также были получены навыки для работы с СУБД *PostgreSQL* [12].

4.1 Недостатки

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

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

A Визуализации структуры проекта

```
 /db-course-project-app
--- .gitignore
--- package.json
--- yarn-error.log
--- .babelrc
--- jest.config.js
--- requirements.txt
--- .eslintignore
--- yarn.lock
--- .env
--- README.md
--- LICENSE
--- /util
------ nodemon.json
--- /tests
------ test_smoke.py
--- /src
------ 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
```

B Код проекта

db-course-project-app/package.json

```
{
  "name": "db-course-project-app",
  "version": "1.0.0",
  "description": "Web-application for course project by Database.",
  "main": "build/index.js",
  "author": "Dmitry Vasiliev",
  "scripts": {
    "test": "jest",
    "start-dev": "nodemon --config \"./util/nodemon.json\"/",
    "build": "rm -rf ./build/* ./src/public/contest.bundle.js
    ↪  ./src/public/main.bundle.js && babel src -d build && webpack
    ↪  --config src/webpack.config.js --mode=\"production\"",
    "start": "node -r dotenv/config build/index.js",
    "es-lint": "eslint . -c src/.eslintrc.js --ext \"jsx,js\"",
    "style-lint": "stylelint --ignore-pattern src/client/tests --config
    ↪  src/.stylelintrc src/client/*",
    "watch": "webpack --config src/webpack.config.js --watch",
    "desktop-dev": "electron ./src/main.js",
    "desktop-start" : "electron ./build/main.js"
  },
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^1.2.32",
    "@fortawesome/free-brands-svg-icons": "^5.15.1",
    "@fortawesome/free-regular-svg-icons": "^5.15.1",
    "@fortawesome/free-solid-svg-icons": "^5.15.1",
    "@fortawesome/react-fontawesome": "^0.1.11",
    "bcrypt": "^5.0.0",
    "bootstrap": "^4.5.0",
    "compression": "^1.7.4",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-handlebars": "^5.0.0",
    "history": "^5.0.0",
    "http-status-codes": "^1.4.0",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.19",
    "mdbreact": "^4.27.0",
    "morgan": "^1.10.0",
    "multer": "^1.4.2",
    "pg": "^8.3.0",
    "pg-hstore": "^2.3.3",
    "prop-types": "^15.7.2",
    "pug": "^3.0.0",
    "react": "^16.13.1",
    "react-bootstrap": "^1.2.2",
    "react-dom": "^16.13.1",
```

```
    "react-redux": "^7.2.1",
    "react-router-bootstrap": "^0.25.0",
    "react-router-dom": "^5.2.0",
    "react-router-prop-types": "^1.0.5",
    "redux": "^4.0.5",
    "sequelize": "^6.3.3",
    "serve-favicon": "^2.5.0",
    "validate.js": "^0.13.1"
  },
  "devDependencies": {
    "@babel/cli": "^7.11.6",
    "@babel/core": "^7.11.6",
    "@babel/node": "^7.10.5",
    "@babel/plugin-transform-runtime": "^7.11.5",
    "@babel/preset-env": "^7.11.5",
    "@babel/preset-react": "^7.10.4",
    "@types/compression": "^1.7.0",
    "@types/dotenv": "^8.2.0",
    "@types/express": "^4.17.7",
    "@types/express-handlebars": "^3.1.0",
    "@types/history": "^4.7.8",
    "@types/jest": "^26.0.4",
    "@types/jsonwebtoken": "^8.5.0",
    "@types/lodash": "^4.14.161",
    "@types/morgan": "^1.9.1",
    "@types/multer": "^1.4.4",
    "@types/node": "^14.0.22",
    "@types/react": "^16.9.43",
    "@types/react-dom": "^16.9.8",
    "@types/react-redux": "^7.1.9",
    "@types/react-router-bootstrap": "^0.24.5",
    "@types/react-router-dom": "^5.1.5",
    "@types/redux": "^3.6.0",
    "@types/sequelize": "^4.28.9",
    "@types/serve-favicon": "^2.5.0",
    "babel-loader": "^8.1.0",
    "css-loader": "^3.6.0",
    "electron": "^10.1.5",
    "eslint": "^7.9.0",
    "eslint-plugin-jest": "^24.0.1",
    "eslint-plugin-react": "^7.20.6",
    "husky": "^4.2.5",
    "jest": "^26.1.0",
    "jest-fetch-mock": "^3.0.3",
    "lorem-ipsum": "^2.0.3",
    "nodemon": "^2.0.4",
    "sass": "^1.26.10",
    "sass-loader": "^9.0.2",
    "style-loader": "^1.2.1",
    "stylelint": "^13.6.1",
```

```
    "stylelint-config-sass-guidelines": "^7.0.0",
    "webpack": "^4.43.0",
    "webpack-cli": "^3.3.12"
  },
  "husky": {
    "hooks": {
      "pre-commit": "yarn es-lint && yarn style-lint && yarn test",
      "pre-push": "./venv/bin/pytest tests"
    }
  }
}
```

db-course-project-app/.babelrc

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

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

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

db-course-project-app/requirements.txt

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

db-course-project-app/README.md

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

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

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

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

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

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

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

```
const electron = require('electron');
// Module to control application life.
const app = electron.app;
// Module to create native browser window.
const BrowserWindow = electron.BrowserWindow;

// Keep a global reference of the window object, if you don't, the window
 ↪  will
// be closed automatically when the JavaScript object is garbage
 ↪  collected.
let mainWindow;

function createWindow() {
    // Create the browser window.
    mainWindow = new BrowserWindow({
        width: 800,
        height: 600
    });

    // and load the index.html of the app.
    mainWindow.loadURL('http://localhost:3000');

    // Open the DevTools.
    // mainWindow.webContents.openDevTools();

```

```
    // Emitted when the window is closed.
    mainWindow.on('closed', function () {
        // Dereference the window object, usually you would store windows
        // in an array if your app supports multi windows, this is the
        ↪  time
        // when you should delete the corresponding element.
        mainWindow = null;
    });
}

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', createWindow);

// Quit when all windows are closed.
app.on('window-all-closed', function() {
    // On OS X it is common for applications and their menu bar
    // to stay active until the user quits explicitly with Cmd + Q
    if (process.platform !== 'darwin') {
        app.quit();
    }
});

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

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

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

fetchMock.enableMocks();
```

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

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

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

```
import * as path from "path";
import express from "express";
```

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

import config from "./config";

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

import models from "./models";

const app = express();

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

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

// set response compression
app.use(compression());
// set logger
app.use(morgan("common"));
// serve json requests
app.use(express.json());
// serve form requests
app.use(express.urlencoded({ extended: true }));
// serve favicon
app.use(serveFavicon(path.join(process.cwd(), '/src', '/public',
  ↪  'favicon.ico')))

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

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

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

app.use("*", mainRouter);
```

```

app.use(errorHandler);

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

        for (let model of models) {
            await model.sync();
        }

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

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

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

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

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

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

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

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

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

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

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

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

import "bootstrap/scss/bootstrap.scss";

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

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

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

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

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

import "bootstrap/scss/bootstrap.scss";

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

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

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

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

```

```
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,
           ↪  typeAnswer)),
16         appendAnswer: i => dispatch(testEditorActions.appendAnswer(i))
17     };
18 }
19
20 const TestEditorQuestionList = connect(mapStateToProps,
   ↪  mapDispatchToProps)(QuestionEditList);
21
22 export { TestEditorQuestionList };
```

...oject-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     };
```

```
}

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

export { TestEditorTagList };
```

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

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

export default TestEditorTagList;
```

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

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

export default Test;
```

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

```
import { connect } from "react-redux";
import * as testPassingActions from "../../actions/testPassing";
import Test from "../../pages/Test";

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

    return { questions };
}

function mapDispatchToProps(dispatch) {
    return {
        setInitData: initState =>
            dispatch(testPassingActions.init(initState)),
        updateAnswer: (questionId, answerId, typeAnswer, state) =>
            dispatch(testPassingActions.updateAnswer(questionId,
                answerId, typeAnswer, state))
    };
}

const connectedTest = connect(mapStateToProps, mapDispatchToProps)(Test);

export { connectedTest as Test };
```

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

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

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

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

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

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

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

```
import * as testPassingActions from "../actions/testPassing";
import {ANSWER_TYPE} from "../../models/Test/config";


const initState = {
    questions: []
};

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

    switch (action.type) {
        case testPassingActions.INIT: {
            const { internalState } = action;

            return { questions: internalState };
        }
        case testPassingActions.UPDATE_ANSWER: {
            const {
                questionId,
                answerId,
                typeAnswer,
                state
            } = action;
            const { questions } = newState;
```

```
            const { answers } = questions[questionId];

            // reset states others answers
            if (typeAnswer === ANSWER_TYPE.ONE) {
                for (let currAnswer of answers) {
                    currAnswer.isChecked = false;
                }
            }

            answers[answerId].isChecked = state;

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

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

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

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

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

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

```
import * as testEditorActions from "../actions/testEditor";
import {createQuestion, createAnswer} from "../helpers/question";
import {ANSWER_TYPE} from "../../models/Test/config";

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

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

    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
```

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

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

            newState.questions[id].typeAnswer = typeAnswer;

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

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

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

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

            newState.questions[questionId].answers = answers;

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

            answers[answerId].content = value;

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

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

```
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,
```

```
    MAX_PASSWORD_LENGTH,
    MIN_LOGIN_LENGTH,
    MAX_LOGIN_LENGTH,
    MAX_EMAIL_LENGTH
} = userConstraints;

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

        this.state = {
            email: '',
            login: '',
            password: '',
            repeatPassword: '',

            listErrors: [],

            isLoading: false
        };

        this.handleInputChange = this.handleInputChange.bind(this);
        this.handleFormSubmit = this.handleFormSubmit.bind(this);
        this.hideErrorAlert = this.hideErrorAlert.bind(this);
        this.toggleLoadingState = this.toggleLoadingState.bind(this);
    }

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

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

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

    _generateFormData() {
        const formData = new FormData();

        formData.append('login', this.state.login);
        formData.append('password', this.state.password);
        formData.append('email', this.state.email);
        formData.append('repeatPassword', this.state.repeatPassword);

        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                      }
```

```
                    <Row className="justify-content-center">
                        <Button variant="success"
                                size="lg"
                                onClick={this.handleSubmit}>
                            Submit
                        </Button>
                    </Row>
                </Container>
            );
        }
}

Test.propTypes = {
    questions: PropTypes.arrayOf(
        PropTypes.exact({
            title: PropTypes.string,
            typeAnswer: PropTypes.oneOf([ANSWER_TYPE.ONE,
            ↪  ANSWER_TYPE.MULTIPLE]),
            answers: PropTypes.arrayOf(
                PropTypes.exact({
                    content: PropTypes.string,
                    isChecked: PropTypes.bool
                })
            )
        })
    ).isRequired,
    setInitData: PropTypes.func.isRequired,
    updateAnswer: PropTypes.func.isRequired,
    location: ReactRouterPropTypes.location.isRequired,
    history: ReactRouterPropTypes.location.isRequired
};

export default withRouter(Test);
```

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

```

```

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

```
import TestResult from "./TestResult";

export default TestResult;
```

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

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

```
import ReactRouterPropTypes from "react-router-prop-types";
import Container from "react-bootstrap/Container";
import Table from "react-bootstrap/Table";
import ResultStatus from "../../components/ResultStatus";
import * as testResultService from "../../services/testResult";

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

        // get attempt id from url params
        const { location } = props;
        let query = new URLSearchParams(location.search);

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

        this.state = {
            attemptId,
            userAnswers: []
        };
    }

    async componentDidMount() {
        const { attemptId } = this.state;

        let userAnswers = [];
        try {
            userAnswers = await testResultService.init(attemptId);
        } catch (err) {
            console.log(err);
        }

        this.setState({ userAnswers });
    }

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

        return (
            <Container className="p-3">
                <Table responsive="lg">
                    <thead>
                        <tr>
                            <th>Question ID</th>
                            <th>Result</th>
                        </tr>
                    </thead>
                    <tbody>
                        {
                            userAnswers.length ? (
```

```
                                userAnswers.map((answer, i) => {
                                    const { isCorrect } = answer;

                                    return (
                                        <tr key={i}>
                                            <td>{i + 1}</td>
                                            <td><ResultStatus
                                            ↪  isCorrect={isCorrect}
                                            ↪  /></td>
                                        </tr>
                                    );
                                })
                            ) : (
                                <tr>
                                    <td>None</td>
                                </tr>
                            )
                        }
                    </tbody>
                </Table>
            </Container>
        );
    }
}

TestResult.propTypes = {
    location: ReactRouterPropTypes.location
};

export default withRouter(TestResult);
```

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

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

  &__title {
    text-align: center;
  }

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

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

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

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

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

import "./style.scss";

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

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

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

            listErrors: [],

            isLoading: false
        }

        this.handleInputChange = this.handleInputChange.bind(this);
        this.handleCheckboxChange = this.handleCheckboxChange.bind(this);
        this.handleFormSubmit = this.handleFormSubmit.bind(this);
        this.hideErrorAlert = this.hideErrorAlert.bind(this);
    }

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

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

```
    }

    handleCheckboxChange(event) {
        const { name, checked } = event.target;

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

    _generateFormData() {
        const formData = new FormData();

        formData.append('login', this.state.login);
        formData.append('password', this.state.password);

        return formData;
    }

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

        const { history, dispatch } = this.props;
        const formData = this._generateFormData();

        this.toggleLoadingState();

        try {
            const { token, user } = await authService.signIn(formData);

            localStorage.setItem('TOKEN', token);

            dispatch(authActions.success(user));

            history.push('/');
        } catch ({ errors }) {
            this.setState({ listErrors: errors });
        }

        this.toggleLoadingState();
    }

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

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

```
            }
        });
    }

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

        return (
            <Container className="p-3">
                <Col lg={{offset: 3, span: 6}}>
                    <h2 className="main-login-form__title">Login
                    ↪ form</h2>
                    <Form>
                        <ErrorFormAlert listErrors={listErrors}
                                        show={listErrors.length !== 0}
                                        onHide={this.hideErrorAlert} />
                        <Form.Group controlId="main-login-form__login">
                            <Form.Control type="text"
                                          placeholder="Enter login"
                                          name="login"
                                          minLength={MIN_LOGIN_LENGTH}
                                          maxLength={MAX_LOGIN_LENGTH}
                                          required

                                          ↪ onChange={this.handleInputChange}
                                          ↪ />
                        </Form.Group>
                        <Form.Group
                        ↪ controlId="main-login-form__password">
                            <Form.Control type="password"
                                          placeholder="Enter password"
                                          name="password"
                                          minLength={MIN_PASSWORD_LENGTH}
                                          maxLength={MAX_PASSWORD_LENGTH}
                                          required

                                          ↪ onChange={this.handleInputChange}/>
                        </Form.Group>
                        <Button variant="primary"
                                type="submit"
                                block
                                disabled={isLoading}
                                onClick={this.handleFormSubmit}>
                            { isLoading ? 'Loading...' : 'Submit' }
                        </Button>
                    </Form>
                </Col>
            </Container>
        );
    }
```

```
}

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

const connectedLogin = connect()(Login);

export { connectedLogin as Login };
```

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

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

export default Login;
```

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

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

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

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

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

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

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

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

```
            { status: OK }
        ]);

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

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

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

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

    // it("sign in when API responds with status OK", async () => {
    //     const respond = { test: 'test' };
    //
    //     fetch.mockResponse([
    //         JSON.stringify(respond),
    //         { status: OK }
    //     ]);
    //
    //     await expect(authService.signIn()).resolves.toEqual(respond);
    // });
});
```

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

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

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

export const initAuthStore = async (store) => {
```

```
    const token = getToken();

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

            appendAuth(headers, token);

            response = await fetch('/api/init', { headers, method: 'POST' });
        } catch (err) {
            store.dispatch(authActions.logOut());
        }

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

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

            history.push('/login');
        }
    }
};

export default store;
```

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

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

export { store, initAuthStore };
```

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

```
/**
 * Add to HTTP header 'Authorization' field
 * @param {Headers} headers
 * @param {string} token
 * @return {Headers}
 */
export function appendAuth(headers, token) {
    headers.append('Authorization', token);

```

```
    return headers;
}

/**
 * Add to HTTP headers for working with JSON
 * @param {Headers} headers
 * @return {Headers}
 */
export function appendJSON(headers) {
    headers.append('Accept', 'application/json');
    headers.append('Content-Type', 'application/json');

    return headers;
}
```

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

```
import {ANSWER_TYPE} from "../../models/Test/config";

/**
 * Create answer
 * @param {string} content - Answer text
 * @param {boolean} isRight - Result
 * @return {{isRight: boolean, content: string}}
 */
export const createAnswer = (content = '', isRight = false) => {
    return {content, isRight};
};

/**
 * Create question
 * @param {string} title - Question title
 * @param {string} typeAnswer - Answer type
 * @param {Array<{content: string, isRight: boolean}>} answers
 * @return {{answers: {isRight: boolean, content: string}[], title:
 ↪  string, typeAnswer: string}}
 */
export const createQuestion = (title = '',
                               typeAnswer = ANSWER_TYPE.ONE,
                               answers = [createAnswer(),
                                ↪  createAnswer()]) => {
    return {title, typeAnswer, answers};
};
```

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

```
/**
 * Toggle state of app loader
 */
```

```
export function toggleLoader() {
    const loader = document.querySelector('.loader');

    loader.classList.toggle('loader_hide');
}
```

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

```
const TOKEN_ID = 'TOKEN';

/**
 * Get JWT
 * @return {string}
 */
export function getToken() {
    return localStorage.getItem(TOKEN_ID);
}

/**
 * Remove JWT from localStorage
 */
export function removeToken() {
    localStorage.removeItem(TOKEN_ID);
}
```

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

```
import * as React from 'react';
import { Switch, Route } from 'react-router-dom';
import PrivateRoute from "../../hoc/PrivateRoute";
import NotIsLoggedInRoute from "../../hoc/NotIsLoggedInRoute";

import Container from "react-bootstrap/Container";

import Login from "../../pages/Login";
import Home from "./pages/Home"
import SignUp from "../../pages/SignUp";
import TestEditor from "./pages/TestEditor";
import Test from "../../containers/Test";
import TestResult from "../../pages/TestResult";

import Footer from "../../components/Footer";
import Header from "./components/Header";
import Profile from "./pages/Profile";
import HttpErrorInfo from "../../components/HttpErrorInfo";
import {NOT_FOUND} from "http-status-codes";
import TestStatistic from "./pages/TestStatistic";

class App extends React.Component {
```

```
    render() {
        return (
            <div>
                <Header />

                <Switch>
                    <Route exact path='/' component={Home}/>
                    <NotIsLoggedInRoute path='/login' component={Login}/>
                    <NotIsLoggedInRoute path='/signup'
                    ↪  component={SignUp}/>
                    <PrivateRoute path='/profile' component={Profile}/>
                    <PrivateRoute exact path='/test/edit'
                    ↪  component={TestEditor} />
                    <PrivateRoute exact path='/test/result'
                    ↪  component={TestResult} />
                    <PrivateRoute exact path='/test/pass'
                    ↪  component={Test} />
                    <PrivateRoute exact path='/test/statistic'
                    ↪  component={TestStatistic} />
                    <Route component={() => <HttpErrorInfo
                    ↪  status={NOT_FOUND} />} />
                </Switch>

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

export default App;
```

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

```

```

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

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

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

```
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

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

export default ProfileSettings;
```

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

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

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

        this.state = {
            modalShow: false
        }

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

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

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

        try {
            await editProfileSettings.remove();

            dispatch(authActions.logOut());

            history.push('/');
        } catch ({ errors }) {
```

```
            onSubmitError(errors);
        }
    }

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

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

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

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

const connectedDeleteProfileForm = connect()(DeleteProfileForm);

export default withRouter(connectedDeleteProfileForm);
```

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

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

export default DeleteProfileForm;
```

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

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

export default UpdatePasswordForm;
```

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

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

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

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

            isLoading: false
        };

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

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

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

    _generateFormData() {
```

```
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 handleFormSubmit(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

```
import * as React from "react";
import Container from "react-bootstrap/Container";
import ListTestCards from "../../../../components/ListTestCards";
import * as editTest from "../../../../services/editTest";
import HttpErrorInfo from "../../../../components/HttpErrorInfo";
import {NO_CONTENT} from "http-status-codes";
import Form from "react-bootstrap/Form";
import InputGroup from "react-bootstrap/InputGroup";
import FormControl from "react-bootstrap/FormControl";
import Button from "react-bootstrap/Button";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
import TagList from "../../../../components/TagList";

import "./style.scss";

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

        this.state = {
            profileTests: [],
            searchTitle: '',
            searchTagValue: '',
            searchTags: []
        };

        this.handleDeleteTestCard = this.handleDeleteTestCard.bind(this);
        this.handleSearchTitleChange =
        ↪  this.handleSearchTitleChange.bind(this);
        this.handleSearchTagValueChange =
        ↪  this.handleSearchTagValueChange.bind(this);
        this.handleAddingTag = this.handleAddingTag.bind(this);
        this.handleSearchTagDeleting =
        ↪  this.handleSearchTagDeleting.bind(this);
        this.searchFilter = this.searchFilter.bind(this);
    }

    async handleDeleteTestCard(testId) {
        await editTest.deleteTest(testId);

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

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

            return {
```

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

    handleSearchTagDeleting(tagId) {
        this.setState(prev => {
            const { searchTags } = prev;

            return {
                searchTags: [
                    ...searchTags.slice(0, tagId),
                    ...searchTags.slice(tagId + 1),
                ]
            };
        });
    }

    handleSearchTitleChange({ target }) {
        const { value } = target;

        this.setState({ searchTitle: value });
    }

    handleSearchTagValueChange({ target }) {
        const { value } = target;

        this.setState({ searchTagValue: value })
    }

    handleAddingTag() {
        const { searchTagValue, searchTags } = this.state;
        const tagIsNotIncluded = !searchTags.includes(searchTagValue);

        if (searchTagValue && tagIsNotIncluded) {
            this.setState(prev => {
                return {
                    searchTags: [...prev.searchTags, searchTagValue]
                };
            });
        }
    }

    async componentDidMount() {
        const responseJson = await editTest.getOwnTests();

        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.handleSearchTagValueChan
138                                     />
139                                     <InputGroup.Append>
140                                         <Button variant="primary"
```

```
                                                    ↪  onClick={this.handleAddingTag}>
                                                Add
                                            </Button>
                                        </InputGroup.Append>
                                    </InputGroup>
                                </Form.Group>
                                <TagList tags={searchTags}
                                         canDelete={true}

                                         ↪  deleteTag={this.handleSearchTagDeleting}
                                         ↪  />
                            </Form>
                        </Col>
                    </Row>
                    {
                        profileTests.length ? (
                            <ListTestCards tests={profileTests}

                                           ↪  onDeleteTestCard={this.handleDeleteTestC
                        ) : (
                            <HttpErrorInfo status={NO_CONTENT}
                                           reason="You don't have tests." />
                        )
                    }
                </Container>
            );
        }
}

export default ProfileTests;
```

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

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

export default ProfileTests;
```

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

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

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

```

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

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

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

export default TestEditor;
```

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

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

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

import "./style.scss";

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

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

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

        const isEditing = !!testId;

        this.state = {
            isEditing,
```

```
            testId: testId ? testId : -1,
            tagValue: '',
            listErrors: [],
            isLoading: false
        };

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

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

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

                dispatch(testEditorActions.update(response));
            } catch (err) {
                console.error(err);
            }
        } else {
            dispatch(testEditorActions.reset());
        }
    }

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

    handleTitleChange({ target: { value } }) {
        const { dispatch } = this.props;

        dispatch(testEditorActions.updateTitle(value));
    }

    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.handleTitl
                                                      ↪  />
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.handleDesc
                                                      ↪  />
174                                     </Form.Group>
175                                 </Col>
176                             </Row>
177                             <Row>
178                                 <Col lg={4}>
```

```
                    <Form.Label>Tags:</Form.Label>
                </Col>
                <Col lg={8}>
                    <Form.Group controlId="">
                        <InputGroup className="mb-3">
                            <FormControl
                                aria-label="Recipient's
                                ↪ username"

                                ↪ aria-describedby="basic-addon2"
                                required

                                ↪ onChange={this.handleTagInputCh
                            />
                            <InputGroup.Append>
                                <Button variant="primary"

                                    ↪ onClick={this.handleApp
                                    Add
                                </Button>
                            </InputGroup.Append>
                        </InputGroup>
                    </Form.Group>
                    <TestEditorTagList />
                </Col>
            </Row>

            <hr/>

            <TestEditorQuestionList />

            <Row>
                <Col lg={12}>
                    <div
                    ↪ className="test-editor__submit-section">
                        <div
                        ↪ className="test-editor__submit-section-
                            <Button
                            ↪ className="test-editor__submit-btn"
                                type="primary"
                                size="lg"

                                ↪ onClick={this.handleAppendQ
                                Add question
                            </Button>
                        </div>
                        <div
                        ↪ className="test-editor__submit-section-
                            <Button
                            ↪ className="test-editor__submit-btn"
```

```
                                                    type="primary"
                                                    size="lg"
                                                    disabled={isLoading}
                                                    onClick={this.handleFormSub
                                                {isLoading ? 'Loading...'
                                                : 'Publish'}
                                            </Button>
                                        </div>
                                    </div>
                                </Col>
                            </Row>
                        </Form>
                    </Col>
                </Row>
            </Container>
        );
    }
}

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

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

    return { testEditor };
```

```
}

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

export { connectedTestEditor as TestEditor };
```

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

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

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

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

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

import { Switch, Route } from 'react-router-dom';
import { LinkContainer } from "react-router-bootstrap";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faList, faUserCog, faPoll } from
  ↪  "@fortawesome/free-solid-svg-icons";
import Container from "react-bootstrap/Container";
import Col from "react-bootstrap/Col";
import Row from "react-bootstrap/Row";
import Nav from "react-bootstrap/Nav";
import ProfileSettings from "../ProfileSettings";
import ProfileAttempts from "../ProfileAttempts";
import HttpErrorInfo from "../../../../components/HttpErrorInfo";
import ProfileTests from "../ProfileTests";

import "./style.scss";

const Profile = ({ user }) => {
    return (
```

```
23          <Container className="p-3">
24              <Row>
25                  <Col lg={3}>
26                      <img className="profile__avatar"
27                           ↪ src="http://ssl.gstatic.com/accounts/ui/avatar_2x.png"
                             ↪ alt="avatar"/>
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>
```

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

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

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

    return { user };
}

const connectedProfile = connect(mapStateToProps)(Profile);

export { connectedProfile as Profile };
```

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

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

export default Profile;
```

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

```

```

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

```
import * as React from "react";
import { Link } from "react-router-dom";
import Container from "react-bootstrap/Container";
import Table from "react-bootstrap/Table";
import * as attemptsService from "../../../../services/attempt";
import {NO_CONTENT, OK} from "http-status-codes";

import "./style.scss";
import HttpErrorInfo from "../../../../components/HttpErrorInfo";

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

        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) => {
```

```
                                        const { title, testId, result, date } = attempt;
                                        const link = `/test/pass?id=${testId}`;

                                        return (
                                            <tr key={i}>
                                                <td>
                                                    <Link to={link}>{title}</Link>
                                                </td>
                                                <td>{result * 100}%</td>
                                                <td>
                                                    <time dateTime={date}>{date}</tim
                                                </td>
                                            </tr>
                                        );
                                    })
                                }
                                </tbody>
                            </Table>
                        ) : (
                            <HttpErrorInfo status={NO_CONTENT}
                                           reason={"You don't have any attempts."} />
                        )
                    )
                }
            </Container>
        );
    }
}

export default ProfileAttempts;
```

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

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

export default ProfileAttempts;
```

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

```

```

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

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

export default Home;
```

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

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

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

import "./style.scss";

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

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

export default Home;
```

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

```
import * as React from "react";
import Container from "react-bootstrap/Container";
import TableStatistic from "../../components/TableStatistic";
import { withRouter } from "react-router-dom";
import * as attemptService from "../../../../services/attempt";
import ReactRouterPropTypes from "react-router-prop-types";

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

        // get attempt id from url params
        const { location } = props;
        let query = new URLSearchParams(location.search);

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

        this.state = {
            data: [],
            testId
        };
    }

    async componentDidMount() {
        const { testId } = this.state;

        try {
            const data = await attemptService.getOwnTestAttempts(testId);

            this.setState({ data });
        } catch (err) {
            console.error(err);
        }
    }
```

```
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  //}
```

db-course-project-app/src/client/apps/main/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      faClipboardList,
10     faUser,
11     faSignOutAlt,
12     faSignInAlt,
13     faIdCard,
14     faUserPlus,
```

```
    faPlus
} from "@fortawesome/free-solid-svg-icons";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
import NavDropdown from "react-bootstrap/NavDropdown";
import Modal from "react-bootstrap/Modal";
import Button from "react-bootstrap/Button";
import { LinkContainer } from "react-router-bootstrap";

import "./style.scss";

const Header = ({ isLoggedIn, user, dispatch, history }) => {
    const [modalShow, setModalShow] = React.useState(false);

    const showModal = () => setModalShow(true);

    const hideModal = () => setModalShow(false);

    const onLogOut = () => {
        dispatch(authActions.logOut());

        setModalShow(false);

        history.push("/");
    };

    return (
        <Navbar bg="dark" variant="dark">
            <LinkContainer to="/">
                <Navbar.Brand>
                    <FontAwesomeIcon icon={faClipboardList} /> PassQuiz
                </Navbar.Brand>
            </LinkContainer>
            <Nav className="mr-auto">
                { !isLoggedIn ? (
                    <>
                        <LinkContainer to="/signup">
                            <Nav.Link> <FontAwesomeIcon icon={faUserPlus}
                            ↪  /> Sign Up</Nav.Link>
                        </LinkContainer>
                        <LinkContainer to="/login">
                            <Nav.Link> <FontAwesomeIcon
                            ↪  icon={faSignInAlt} /> Login</Nav.Link>
                        </LinkContainer>
                    </>
                ) : (
                    <NavDropdown title={
                        <span>
```

```
                        <FontAwesomeIcon icon={faUser} /> {user.login}
                    </span>
                } id="user-nav-dropdown">
                    <LinkContainer to="/profile">
                        <NavDropdown.Item>
                            <FontAwesomeIcon icon={faIdCard} /> My profile
                        </NavDropdown.Item>
                    </LinkContainer>
                    <NavDropdown.Divider />
                    <LinkContainer to="#">
                        <NavDropdown.Item onClick={showModal}>
                            <FontAwesomeIcon icon={faSignOutAlt} /> Logout
                        </NavDropdown.Item>
                    </LinkContainer>
                </NavDropdown>
            ) }
        </Nav>
        {
            isLoggedIn && (
                <Nav>
                    <LinkContainer to="/test/edit">
                        <Button variant="primary">
                            Create test <FontAwesomeIcon icon={faPlus} />
                        </Button>
                    </LinkContainer>
                </Nav>
            )
        }

        <Modal
            size="md"
            aria-labelledby="contained-modal-title-vcenter"
            centered
            onHide={hideModal}
            show={modalShow}
        >
            <Modal.Header closeButton>
                <Modal.Title id="contained-modal-title-vcenter">Log out</Modal.Title>
            </Modal.Header>
            <Modal.Body>
                <p>Are you sure you want to log-off?</p>
            </Modal.Body>
            <Modal.Footer>
                <Button onClick={onLogOut}>Yes</Button>
            </Modal.Footer>
```

```
            </Modal>
        </Navbar>
    );
}

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

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

    return { isLoggedIn, user };
}

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

export { connectedHeaderWithRouter as Header };
```

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

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

export default Header;
```

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

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

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

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

export { AnswerEditList };
```

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

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

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

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

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

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

                            return (
                                <AnswerEditItem key={i}
                                                name={name}
                                                type={type}
                                                isRight={isRight}
                                                content={content}
                                                onChangeAnswer={(isRight,
                                                ↪  typeAnswer) =>
                                                ↪  updateAnswers(name,
                                                ↪  i, isRight,
                                                ↪  typeAnswer)}
                                                onDeleteAnswer={() =>
                                                ↪  deleteAnswer(name,
                                                ↪  i)}
                                                onChangeAnswerText={value
                                                ↪  =>
                                                ↪  updateAnswerText(name,
                                                ↪  i, value)} />
```

```
                                );
                            })
                        }
                    </Form.Group>
                    <Button variant="primary" onClick={onAppendAnswer}>
                        Add answer
                    </Button>
                </>
            );
        }
}

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

function mapStateToProps() {
    return {};
}

function mapDispatchToProps(dispatch) {
    return {
        deleteAnswer: (questionId, answerId) => dispatch(testEditorActions.deleteAnswer(questionId, answerId)),
        updateAnswerText: (questionId, answerId, value) => dispatch(testEditorActions.updateAnswerText(questionId, answerId, value)),
        updateAnswers: (questionId, answerId, isRight, typeAnswer) => dispatch(testEditorActions.updateAnswers(questionId, answerId, isRight, typeAnswer))
    };
}

const connectedAnswerEditList = connect(mapStateToProps, mapDispatchToProps)(AnswerEditList);

```

```
84 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"
```

```
                                    as="textarea"
                                    rows={3}
                                    value={title}
                                    onChange={event => {
                                        const titleVal =
                                        ↪ event.target.value;

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

                                ↪ name={`${name}_toggle_answer_typ

                                ↪ defaultValue={ANSWER_TYPE.ONE}>
                                <ToggleButton value={ANSWER_TYPE.ONE}

                                ↪ onChange={this.handleToggleChange
                                ↪ answer</ToggleButton>
                                <ToggleButton
                                ↪ value={ANSWER_TYPE.MULTIPLE}

                                ↪ onChange={this.handleToggleChange
                                ↪ answers</ToggleButton>
                            </ToggleButtonGroup>
                        </Form.Group>
                    </Col>
                </Row>
                <Row>
                    <Col lg={4}>
                        <Form.Label>Answers:</Form.Label>
                    </Col>
                    <Col lg={8}>
                        <AnswerEditList name={name}
                                        type={typeAnswer}
                                        answers={answers}
                                        onAppendAnswer={() =>
                                        ↪ onAnswerListUpdate(name)} />
                    </Col>
                </Row>
                <Row>
                    <Col lg={12}>
```

```
                        <Button className="float-right"
                                variant="danger"
                                onClick={onDelete}>
                            Delete question
                        </Button>
                    </Col>
                </Row>

                <hr/>
            </>
        );
    }
}

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

export default QuestionEditItem;
```

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

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

export default QuestionEditItem;
```

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

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

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

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

import "./style.scss";

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

        const { content } = props;

        this.state = {
            answerValue: content
        }

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

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

        onChangeAnswer(checked, type);
    }

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

        return (
            <InputGroup className="answer-edit-item">
                <InputGroup.Prepend>
                    {
                        (() => {
                            switch (type) {
                                case ANSWER_TYPE.ONE:
                                    return (
```

```
                                        <InputGroup.Radio name={name}
                                                          checked={isRight}
                                                          onChange={this.handle
                                                          aria-label="Radio
                                                          button for
                                                          following
                                                          text input"
                                                          />
                                    );
                                case ANSWER_TYPE.MULTIPLE:
                                    return (
                                        <InputGroup.Checkbox name={name}
                                                             checked={isRight}
                                                             onChange={this.han
                                                             aria-label="Radio
                                                             button
                                                             for
                                                             following
                                                             text
                                                             input"
                                                             />
                                    );
                            }
                        })()
                    }
                </InputGroup.Prepend>
                <FormControl aria-label="Text input with radio button"
                             value={content}
                             onChange={(event => {
                                 const { value } = event.target;

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

```

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

export default AnswerEditItem;
```

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

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

export default AnswerEditItem;
```

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

```
import TableStatistic from "./TableStatistic";

export default TableStatistic;
```

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

```
import * as React from "react";
import PropTypes from "prop-types";
import Container from "react-bootstrap/Container";
import { MDBDataTable } from 'mdbreact';


const TableStatistic = ({rows}) => {
    const data = {
        columns: [
            {
                label: 'Attempt ID',
                field: 'attemptId',
            },
            {
                label: 'User',
                field: 'login',
            },
            {
                label: 'Result',
                field: 'result',
```

```
            },
            {
                label: 'Date',
                field: 'date',
            }
        ],
        rows: rows
    };

    return (
        <Container className="p-3">
            <MDBDataTable
                striped
                bordered
                small
                sortable
                data={data}
            />
        </Container>
    );
};

TableStatistic.propTypes = {
    rows: PropTypes.arrayOf(
        PropTypes.exact({
            attemptId: PropTypes.number,
            login: PropTypes.string,
            result: PropTypes.number,
            date: PropTypes.string
        })
    ).isRequired
};

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

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

export default QuestionEditList;
```

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

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

```
import {ANSWER_TYPE} from "../../../../../models/Test/config";
import PropTypes from "prop-types";

import QuestionEditItem from "../QuestionEditItem";

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

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

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

                        return (
                            <QuestionEditItem name={i}
                                              key={i}
                                              typeAnswer={typeAnswer}
                                              answers={answers}
                                              title={title}
                                              onDelete={() =>
                                              ↪  deleteQuestion(i)}
                                              onUpdateTitle={titleVal =>
                                              ↪  updateQuestionTitle(i,
                                              ↪  titleVal)}

                                              ↪  onChangeAnswerType={typeAnswer
                                              ↪  =>
                                              ↪  changeQuestionType(i,
                                              ↪  typeAnswer)}
                                              onAnswerListUpdate={id =>
                                              ↪  appendAnswer(id)} />
                        );
                    })
                }
```

```
            </>
        );
    }
}

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

export default QuestionEditList;
```

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

```
import * as React from 'react';
import { Switch, Route } from 'react-router-dom';
import PrivateRoute from "../../hoc/PrivateRoute";
import NotIsLoggedInRoute from "../../hoc/NotIsLoggedInRoute";

import Container from "react-bootstrap/Container";

import Home from './pages/Home';
import Login from "../../pages/Login";
import SignUp from "../../pages/SignUp";
import Test from "../../containers/Test";
import TestResult from "../../pages/TestResult";

import Footer from "../../components/Footer";
import Header from "./components/Header";
import AllTests from "./pages/AllTests";
import HttpErrorInfo from "../../components/HttpErrorInfo";
import {NOT_FOUND} from "http-status-codes";

class App extends React.Component {
```

```
    render() {
        return (
            <div>
                <Header />

                <Switch>
                    <Route exact path='/' component={Home} />
                    <PrivateRoute path='/tests' component={AllTests} />
                    <PrivateRoute exact path='/test/pass'
                    ↪  component={Test} />
                    <NotIsLoggedInRoute path='/login' component={Login}/>
                    <NotIsLoggedInRoute path='/signup' component={SignUp}
                    ↪  />
                    <PrivateRoute exact path='/test/result'
                    ↪  component={TestResult} />
                    <Route component={() => <HttpErrorInfo
                    ↪  status={NOT_FOUND} />} />
                </Switch>

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

export default App;
```

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

```

```

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

```
import * as React from "react";
import Container from "react-bootstrap/Container";
import ListTestCards from "../../../../components/ListTestCards";
import * as editTest from "../../../../services/editTest";
import HttpErrorInfo from "../../../../components/HttpErrorInfo";
import {NO_CONTENT} from "http-status-codes";
import Form from "react-bootstrap/Form";
import InputGroup from "react-bootstrap/InputGroup";
import FormControl from "react-bootstrap/FormControl";
import Button from "react-bootstrap/Button";
import Row from "react-bootstrap/Row";
import Col from "react-bootstrap/Col";
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
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.handleSearchTagValueChan
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

```
import AllTests from "./AllTests";

export default AllTests;
```

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

```

```

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

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

export default Home;
```

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

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

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

import "./style.scss";

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

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

export default Home;
```

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

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

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

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

import {
    faUser,
    faSignOutAlt,
    faSignInAlt,
    faUserPlus,
    faClipboardList,
} from "@fortawesome/free-solid-svg-icons";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
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}
```

```
                                onDelete={() => deleteTag(i)} />
                        );
                    })
                }
            </ul>
    );
};

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

export default TagList;
```

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

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

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

  &__content {
    margin-right: 5px;
  }

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

  }

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

```

```
  &__delete-btn:focus {
    outline: none;
  }
}
```

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

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

export default Tag;
```

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

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

import "./style.scss";

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

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

export default Tag;
```

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

```
.result-status {
  font-size: 1.2rem;

  &__correct {
    color: #77d418;
  }

  &__incorrect {
    color: #d92727;
  }
}
```

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

```
import * as React from "react";
import { faCheckCircle, faTimesCircle } from
↪  "@fortawesome/free-regular-svg-icons";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import PropTypes from "prop-types";

import "./style.scss";

const ResultStatus = ({ isCorrect }) => {
    return (
        <span className={`result-status ${isCorrect ?
        ↪  'result-status__correct' : 'result-status__incorrect'}`}>
            <FontAwesomeIcon icon={isCorrect ? faCheckCircle :
            ↪  faTimesCircle} />
        </span>
    );
};

ResultStatus.propTypes = {
    isCorrect: PropTypes.bool.isRequired
};

export default ResultStatus;
```

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

```
import ResultStatus from "./ResultStatus";

export default ResultStatus;
```

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

```
.test-card {
```

```
  margin: 0 0 15px;
  width: 100%;

  &__title {
    font-size: 1.1rem;
  }

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

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

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

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

export default TestCard;
```

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

```
import * as React from "react";
import PropType from "prop-types";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import {
    faUser,
    faTags,
    faTrash,
    faEdit,
    faShareSquare,
    faPollH
} from "@fortawesome/free-solid-svg-icons";
import Card from "react-bootstrap/Card";
import Button from "react-bootstrap/Button";
import { LinkContainer } from "react-router-bootstrap";
import Dropdown from "react-bootstrap/Dropdown";
import TagList from "../TagList";

import "./style.scss";

const TestCard = ({
      title,
      description,
      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 =
                                        ↪ `${location.origin}/test/pass?id=${test
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: PropType.string.isRequired,
106     description: PropType.string.isRequired,
```

```
    author: PropType.string.isRequired,
    tags: PropType.arrayOf(PropType.string).isRequired,
    testId: PropType.number.isRequired,
    onDeleteTestCard: PropType.func,
    editMenu: PropType.bool
};

export default TestCard;
```

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

```

```

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

```
import Question from "./Question";

export default Question;
```

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

```
import * as React from "react";
import PropTypes from "prop-types";
import Card from "react-bootstrap/Card";
import AnswerList from "../AnswerList";
import Answer from "../Answer";
import {ANSWER_TYPE} from "../../../models/Test/config";

import "./style.scss";

const Question = ({ title, answers, type, id, onAnswerChange }) => {
    return (
        <Card style={{ width: '100%' }}>
            <Card.Header>
                <Card.Title>Question {id + 1}</Card.Title>
                <Card.Body>
                    <Card.Text>{title}</Card.Text>
                </Card.Body>
            </Card.Header>

            <AnswerList>
                {
                    answers.map((answer, i) => {
                        const name = `${id}_answer`;
                        const { content, isChecked } = answer;

                        return (
                            <Answer type={type}
```

```
                                        content={content}
                                        name={name}
                                        isChecked={isChecked}
                                        onAnswerChange={checked =>
                                        ↪  onAnswerChange(id, i, type,
                                        ↪  checked)}
                                        key={i} />
                                );
                            })
                        }
                    </AnswerList>
                </Card>
            );
};

Question.propTypes = {
    title: PropTypes.string.isRequired,
    type: PropTypes.oneOf([ANSWER_TYPE.ONE,
    ↪  ANSWER_TYPE.MULTIPLE]).isRequired,
    answers: PropTypes.arrayOf(
        PropTypes.exact({
            content: PropTypes.string,
            isChecked: PropTypes.bool
        })
    ).isRequired,
    id: PropTypes.number.isRequired,
    onAnswerChange: PropTypes.func.isRequired
};

export default Question;
```

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

```
.list-tests {
  &__col {
    padding: 0;
  }
}
```

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

```
import ListTestCards from "./ListTestCards";

export default ListTestCards;
```

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

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

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

import Row from "react-bootstrap/Row";
import TestCard from "../TestCard";

import "./style.scss";

const ListTestCards = ({ tests, editMenu=true, onDeleteTestCard }) => {
    return (
        <Row>
            {
                tests.map(test => {
                    const {
                        title,
                        description,
                        author,
                        tags,
                        testId
                    } = test;

                    return (
                        <TestCard title={title}
                                  description={description}
                                  author={author}
                                  tags={tags}
                                  key={testId}
                                  testId={testId}
                                  editMenu={editMenu}
                                  onDeleteTestCard={onDeleteTestCard} />
                    );
                })
            }
        </Row>
    );
};

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

export default ListTestCards;
```

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

```

```

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

```
import AnswerList from "./AnswerList";

export default AnswerList;
```

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

```
import * as React from "react";
import PropTypes from "prop-types";
import ListGroup from "react-bootstrap/ListGroup";

import "./style.scss";

const AnswerList = ({ children }) => {
    return (
        <ListGroup className="list-group-flush">
            {children}
        </ListGroup>
    );
};

AnswerList.propTypes = {
    children: PropTypes.arrayOf(PropTypes.element).isRequired
};

export default AnswerList;
```

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

```

```

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

```
import * as React from "react";
import PropTypes from "prop-types";
import ListGroupItem from "react-bootstrap/ListGroupItem";
import Form from "react-bootstrap/Form";
import {ANSWER_TYPE} from "../../../models/Test/config";

import "./style.scss";

const Answer = ({ type, name, content, isChecked, onAnswerChange }) => {
    return (
        <ListGroupItem>
            <Form.Check
                type={type}
                name={name}
                label={content}
```

```
                checked={isChecked}
                onChange={event => {
                    const { checked } = event.target;

                    onAnswerChange(checked);
                }}
            />
        </ListGroupItem>
    );
};

Answer.propTypes = {
    type: PropTypes.oneOf([ANSWER_TYPE.ONE,
    ↪  ANSWER_TYPE.MULTIPLE]).isRequired,
    name: PropTypes.string.isRequired,
    content: PropTypes.string.isRequired,
    isChecked: PropTypes.bool.isRequired,
    onAnswerChange: PropTypes.func.isRequired
};

export default Answer;
```

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

```
import Answer from "./Answer";

export default Answer;
```

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

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

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

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

import "./style.scss"

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

    return (
        <footer className="footer">
            <hr/>
```

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

export default Footer;
```

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

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

export default Footer;
```

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

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

import Alert from "react-bootstrap/Alert";

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

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

export default ErrorFormAlert;
```

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

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

export default ErrorFormAlert;
```

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

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

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

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

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

import "./style.scss";

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

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

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

export default HttpErrorInfo;
```

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

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

```

```
export default HttpErrorInfo;
```

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

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

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

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

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

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

```
export const INIT = "INIT";
export const UPDATE_ANSWER = "UPDATE_ANSWER";

/**
 * Action that init
 * @param {Object[]} internalState - Store of questions
 * @param {string} internalState[].title - Question title
 * @param {string} internalState[].typeAnswer - Question type
 * @param {Object[]} internalState[].answers - Array of answers
 * @param {string} internalState[].answers[].content - Answer content
 * @param {boolean} internalState[].answers[].isChecked - Answer state
 * @return {{internalState: Object[], type: string}}
 */
export const init = internalState => {
    return {
        type: INIT,
        internalState
    };
};

/**
 * Action that updates answer in question
 * @param {number} questionId
 * @param {number} answerId
 * @param {string} typeAnswer
 * @param {boolean} state
```

```
 * @return {{answerId: number, questionId: number, state: boolean, type:
 ↪  string}}
 */
export const updateAnswer = (questionId,
                             answerId,
                             typeAnswer,
                             state) => {
    return {
        type: UPDATE_ANSWER,
        questionId,
        answerId,
        typeAnswer,
        state
    };
};
```

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

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

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

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

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

```
};

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

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

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

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

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

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

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

```

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

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

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

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

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

```
import { createBrowserHistory } from "history";

export default createBrowserHistory();
```

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

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

/**
 * Get attempt results
 * @param {number} attemptId - Attempt ID
 * @return {Promise<unknown>}
 */
```

```
export const init = async attemptId => {
    const token = getToken();
    const headers = new Headers();

    appendAuth(headers, token);

    // create url
    const url = `/api/test/result?id=${attemptId}`;

    const response = await fetch(url, {
        method: 'GET',
        headers,
    });

    let responseJson = null;
    try {
        responseJson = await response.json();
    } catch (err) {
        // handle case if json is invalid
        throw [{
            status: INTERNAL_SERVER_ERROR,
            message: 'something went wrong'
        }];
    }

    if (response.ok) {
        const { userAnswers } = responseJson;

        return Promise.resolve(userAnswers);
    } else {
        throw responseJson;
    }
};
```

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

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

/**
 * Get own attempts
 * @return {Promise<Object[]>}
 */
export const getOwnAttempts = async () => {
    const token = getToken();
    const headers = new Headers();

    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) {
```

```
        // handle case if json is invalid
        throw [{
            status: INTERNAL_SERVER_ERROR,
            message: 'something went wrong'
        }];
    }

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

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

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

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

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

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

    const responseJson = await response.json();

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

export function logOut() {
    removeToken();
}
```

```
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 {
```

```
                    status: NOT_FOUND,
                    message: 'test not found'
                };
            }
            default: {
                // TODO: handle default case

                console.log('something went wrong');
            }
        }
    }
};


/**
 * Send answers for checking and return attempt ID if status code is
 ↪  successful
 * @param {Object[]} questions - Question list
 * @param {number} testId - Test ID
 * @return {Promise<number|Object[]>}
 */
export const submit = async (questions, testId) => {
    const token = getToken();
    const headers = new Headers();

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

    // create url
    const url = `/api/test/check`;

    const response = await fetch(url, {
        method: 'POST',
        headers,
        body: JSON.stringify({ questions, testId })
    });

    let responseJson = null;
    try {
        responseJson = await response.json();
    } catch (err) {
        // handle case if json is invalid
        throw [{
            status: INTERNAL_SERVER_ERROR,
            message: 'something went wrong'
        }];
    }

    if (response.ok) {
        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) {
```

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

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

export { remove, updatePassword };
```

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

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

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

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

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

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

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

    appendAuth(headers, token);

    const response  = await fetch('/api/test/profile', {
        method: 'GET',
```

```
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();
```

```
    } else {
        const responseJson = await response.json();

        return Promise.reject(responseJson);
    }
};

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

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

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

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

/**
 * Get all tests
 * @return {Promise<Object[]>}
 */
export const getAllTests = async () => {
    const token = getToken();
    const headers = new Headers();

    appendAuth(headers, token);

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

    const responseJson = await response.json();

    return Promise.resolve(responseJson);
};
```

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

```
import NotIsLoggedInRoute from "./NotIsLoggedInRoute";

export default NotIsLoggedInRoute;
```

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

```
import * as React from "react";
import PropType from "prop-types";
import { connect } from "react-redux";
import { Route, Redirect } from "react-router-dom";

const NotIsLoggedInRoute = ({ component: Component, isLoggedIn, ...rest
  ↪  }) => {
    return (
        <Route { ...rest } render={() => (
            !isLoggedIn ? (
                <Component />
            ) : (
                <Redirect to="/" />
            )
        )} />
    );
};

NotIsLoggedInRoute.propTypes = {
    isLoggedIn: PropType.bool,
    component: PropType.elementType,
};

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

    return { isLoggedIn };
}

export default connect(mapStateToProps)(NotIsLoggedInRoute);
```

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

```
import * as React from "react";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import { Route, Redirect } from "react-router-dom";

const PrivateRoute = ({ component: Component, isLoggedIn, ...rest }) => {
    return (
        <Route { ...rest } render={() => (
            !isLoggedIn ? (
```

```
                <Redirect to="/login" />
            ) : (
                <Component />
            )
        )}
        />
    );
};

PrivateRoute.propTypes = {
    component: PropTypes.elementType,
    isLoggedIn: PropTypes.bool
};

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

    return { isLoggedIn };
}

const connectedPrivateRoute = connect(mapStateToProps)(PrivateRoute);

export { connectedPrivateRoute as PrivateRoute };
```

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

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

export default PrivateRoute;
```

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

```
import { Router } from "express";

const router = new Router();

router.get("*", (req, res) => {
    const isElectronApp = req.header('user-agent').includes('Electron');

    res.render(isElectronApp ? "contest" : "index");
});

export default router;
```

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

```
import { Router } from "express";
import * as attemptsController from "../controllers/attempt";
```

```
import checkToken from "../middlewares/checkToken";

const router = new Router();

router.get("/profile", checkToken, attemptsController.getOwnAttempts);
router.get("/test", checkToken, attemptsController.getOwnTestAttempts);

export default router;
```

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

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

const upload = multer();

const router = new Router();

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

export default router;
```

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

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

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

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

export default router;
```

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

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

```

```
const router = new Router();

router.post("/create", checkToken, testEditor.create);
router.put("/update", checkToken, testEditor.update);
router.post("/update", checkToken, testEditor.getTestForEdit);
router.get("/pass", checkToken, testEditor.getTestForPassing);
router.get("/result", checkToken, testEditor.getAttemptResults);
router.post("/check", checkToken, testEditor.check);
router.get("/profile", checkToken, testEditor.getOwnTests);
router.get("/all", checkToken, testEditor.getAllTests);
router.delete("/delete", checkToken, testEditor.deleteTest);

export default router;
```

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

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

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

```
import Attempt from "../models/Attempt";
import Test from "../models/Test";
import User from "../models/User";

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

    const ownAttempts = await Attempt.findAll({
        where: { userId },
        include: [Test]
    });

    // collect response
    const response = [];
    for (let attempt of ownAttempts) {
        const { createdAt, result } = attempt;
        const { title, id: testId } = attempt.test;

        response.push({
            title,
            result,
            testId,
            date: createdAt,
        });
    }
```

```
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          next({
23              status: BAD_REQUEST,
24              errors: formListErrors.data.errors
25          });
26      }
27
28      try {
29          await User.create({ email, login, password });
30      } catch ({ errors }) {
31          formListErrors.addFromModelErrors(errors);
32
33          next({
34              status: BAD_REQUEST,
35              errors: formListErrors.data.errors
36          });
37      }
38
39      res.sendStatus(OK);
40  };
41
42  export const signIn = async (req, res, next) => {
43      const {
44          login,
45          password,
46      } = req.body;
47      const formListErrors = new FormListErrors();
48
49      let user;
50      try {
51          user = await User.findOne({ where: { login } });
52      } catch (error) {
53          formListErrors.addDefault();
54
55          next({
56              status: INTERNAL_SERVER_ERROR,
57              errors: formListErrors.data.errors
58          });
59      }
60
61      if (!user) {
62          formListErrors.add("user with such name not found.");
63
64          next({
65              status: BAD_REQUEST,
66              errors: formListErrors.data.errors
67          });
```

```
    } else {
        const isRightPassword = await user.comparePasswords(password);

        if (isRightPassword) {
            const token = jwt.sign({ sub: user.id },
            ↪  process.env.JWT_SECRET);

            res.json({
                ...user.initState(),
                token
            });
        } else {
            formListErrors.add("password is invalid");

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

export const initAuth = async (req, res, next) => {
    const user = await User.findByPk(req.userId);

    if (!user) {
        next({
            status: FORBIDDEN,
            errors: [
                { message: 'user with such id not found' }
            ]
        });
    }

    res.json({ ...user.initState() });
};
```

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

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

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

    const user = await User.findByPk(userId);
```

```

    const isRightPassword = await user.comparePasswords(password);

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

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

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

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

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

    res.sendStatus(OK);
};

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

    const user = await User.findByPk(userId);

    await user.destroy();

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

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

```
import {
    BAD_REQUEST,
```

```
    FORBIDDEN,
    INTERNAL_SERVER_ERROR,
    NOT_FOUND,
    OK
} from "http-status-codes";
import Test from "../models/Test";
import User from "../models/User";
import Tag from "../models/Tag";
import TestTag from "../models/TestTag";
import Attempt from "../models/Attempt";
import FormListErrors from "../helpers/FormListErrors";
import { zipWith, zip } from "lodash";

export const update = async (req, res, next) => {
    const { userId } = req;
    const { info, testId } = req.body;
    const content = req.body.questions;
    const { title, description, tags } = info;
    const formListErrors = new FormListErrors();


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

    if (!currTest) {
        formListErrors.addDefault();

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

    if (userId !== currTest.userId) {
        formListErrors.addDefault();

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

    try {
        // update current test
        await currTest.update({
            title,
            description,
            content,
        });
```

```
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 });
```

```
    }

    /**
     * Get error list length
     * @return {number}
     */
    isNotEmpty() {
        return this.data.errors.length;
    }
}
```

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

```
<main id="root">
    <!-- React entry point -->
</main>

<script src="/static/contest.bundle.js"></script>
```

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

```
<main id="root">
    <!-- React entry point -->
</main>

<script src="/static/main.bundle.js"></script>
```

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

```
<html lang="en">
<head>
    {{> meta }}
    {{> favicon }}
    <title>passquiz</title>
</head>
<body>
    {{> loader }}
    {{{ body }}}
</body>
</html>
```

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

```
<link rel="apple-touch-icon" sizes="180x180"
↪  href="/static/apple-touch-icon.png">
<link rel="icon" type="image/png" sizes="32x32"
↪  href="/static/favicon-32x32.png">
```

```
<link rel="icon" type="image/png" sizes="192x192"
  ↪  href="/static/android-chrome-192x192.png">
<link rel="icon" type="image/png" sizes="16x16"
  ↪  href="/static/favicon-16x16.png">
<link rel="manifest" href="/static/site.webmanifest">
<meta name="apple-mobile-web-app-title" content="lms.labchecker.ru">
<meta name="application-name" content="lms.labchecker.ru">
<meta name="msapplication-TileColor" content="#00aba9">
<meta name="theme-color" content="#ffffff">
```

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

```
<style>
    .loader {
        position: fixed;
        display: flex;
        width: 100%;
        height: 100%;
    }

    .loader_hide {
        display: none;
    }

    .loader__dots {
        margin: auto;
    }

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

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

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

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

```
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, minimum-scale=1.0">
4  <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, hooks: true });
16 Tag.belongsToMany(Test, { through: TestTag, hooks: true });
17
18 Test.hasMany(TestTag);
19 TestTag.belongsTo(Test);
20 Tag.hasMany(TestTag);
21 TestTag.belongsTo(Tag);
22
23 Test.afterDestroy(async () => {
24     const allTags = await Tag.findAll();
25
26     // delete tags that not exist
27     for (let tag of allTags) {
```

```
        const testTag = await TestTag.find({
            where: {
                tagId: tag.id
            }
        });

        if (!testTag) {
            await tag.destroy();
        }
    }
});

export default TestTag;
```

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

```
import Test from "./Test";
import User from "./User";
import Tag from "./Tag";
import TestTag from "./TestTag";
import Attempt from "./Attempt";

const models = [
    Test,
    User,
    Tag,
    TestTag,
    Attempt
];

export default models;
```

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

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

export {
    MIN_TAG_NAME_LENGTH,
    MAX_TAG_NAME_LENGTH
}
```

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

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

export default Tag;
```

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

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

import * as tagConstraints from "./constraints";

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

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

export default Tag;
```

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

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

export {
    MIN_TITLE_LENGTH,
    MAX_TITLE_LENGTH,
```

```
    MIN_DESCRIPTION_LENGTH,
    MAX_DESCRIPTION_LENGTH
};
```

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

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

export default Test;
```

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

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

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

const testScheme = {
    title: {
        type: "string",
        presence: true,
        length: {
            minimum: 1,
            tooShort: "of question needs to be not empty"
        }
    },
    typeAnswer: {
        type: "string",
        presence: true,
        inclusion: [ANSWER_TYPE.ONE, ANSWER_TYPE.MULTIPLE],
        length: { minimum: 1 }
    },
    answers: {
        type: "array",
        presence: true,
        length: {
            tooShort: "needs to have minimum 2",
            minimum: 2
        }
    }
};
```

```
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
                   ↪  ${MAX_DESCRIPTION_LENGTH}.`,
79                 args: [
80                     MIN_DESCRIPTION_LENGTH,
81                     MAX_DESCRIPTION_LENGTH
82                 ]
```

```
            }
        }
    },
    content: {
        type: DataTypes.JSON,
        allowNull: false,
        validate: {
            isValidContent: function(value) {
                for (let currQuestion of value) {
                    const resultCheck = validate(currQuestion,
                    ↪ testScheme);

                    if (resultCheck !== undefined) {
                        const firstErrorMsg =
                        ↪ Object.values(resultCheck)[0][0];

                        throw new Error(firstErrorMsg);
                    }
                }
            },
            isValidAnswers: function (value) {
                for (let currQuestion of value) {
                    for (let currAnswer of currQuestion.answers) {
                        const resultCheck = validate(currAnswer,
                        ↪ answerScheme);

                        if (resultCheck !== undefined) {
                            const firstErrorMsg =
                            ↪ Object.values(resultCheck)[0][0];

                            throw new Error(firstErrorMsg);
                        }
                    }
                }
            }
        }
    }
});

export default Test;
```

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

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

    // Login
```

```
    MIN_LOGIN_LENGTH: 6,
    MAX_LOGIN_LENGTH: 128,

    // Email
    MAX_EMAIL_LENGTH: 320
};
```

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

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

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

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

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

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

```
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 => {
```

```
    user.password = await User.hashPassword(user.password);
});

User.hasMany(Test, { onDelete: 'cascade' });
Test.belongsTo(User);

export default User;
```

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

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

export default User;
```

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

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

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

const Attempt = sequelize.define("attempt", {
    id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true
    },
    result: {
        type: DataTypes.FLOAT,
        validate: {
            min: 0.0,
            max: 1.0
        }
    },
    answers: {
        type: DataTypes.JSON,
    }
});

// create associations
User.hasMany(Attempt, { onDelete: 'cascade' });
Attempt.belongsTo(Test, { onDelete: 'cascade' });
Attempt.belongsTo(User, { onDelete: 'cascade' });

Test.afterUpdate(async test => {
```

```
    // handle case if test is updated

    await Attempt.destroy({
        where: {
            testId: test.id
        }
    });
});

export default Attempt;
```

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

```
import Attempt from "./Attempt";

export default Attempt;
```

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

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

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

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

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

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

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

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

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