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: 3 октября 2020 г.

Оценка: _____

Подпись преподавателя: _____

Подпись студента: _____

Москва 2020

Содержание

1 Введение

1.1 Формальные требования

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

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

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

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

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

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

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

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

1.2 Клиентские приложения

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

1.2.1 Тестирующая система

Данное приложение даёт возможность пользователю:

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

1.2.2 Система управления

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

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

1.3 Предметная область

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

1.4 Стэк технологий

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

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

1.5 Инструменты

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

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

2 Инфраструктура проекта

2.1 Архитекрутра

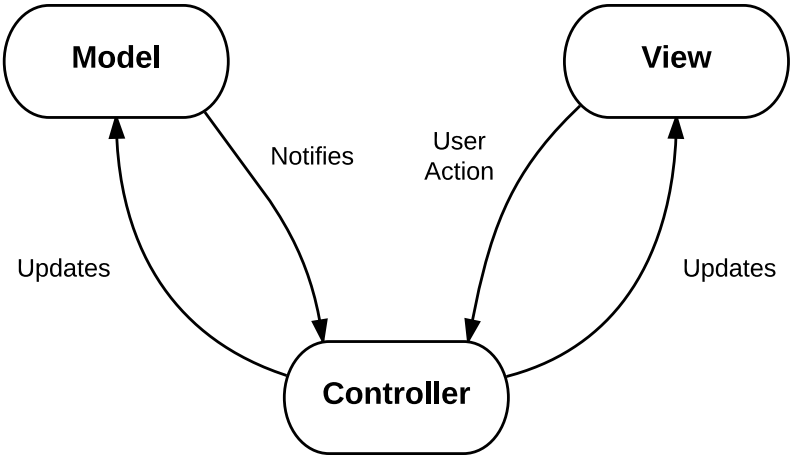

Рис. 1: Визуализация архитектуры MVC

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

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

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

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

2.1.1 Связь логических компонетов и применяемых технологих

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

2.2 Сущности

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

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

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

Таблица 1: Описание сущностей и атрибутов

2.3 Сборка и запуск

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

2.3.1 Development

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

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

2.3.2 Production

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

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

2.4 Деплоинг

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

2.5 Организация работы с *Git* [5]

2.5.1 Git Workflow

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

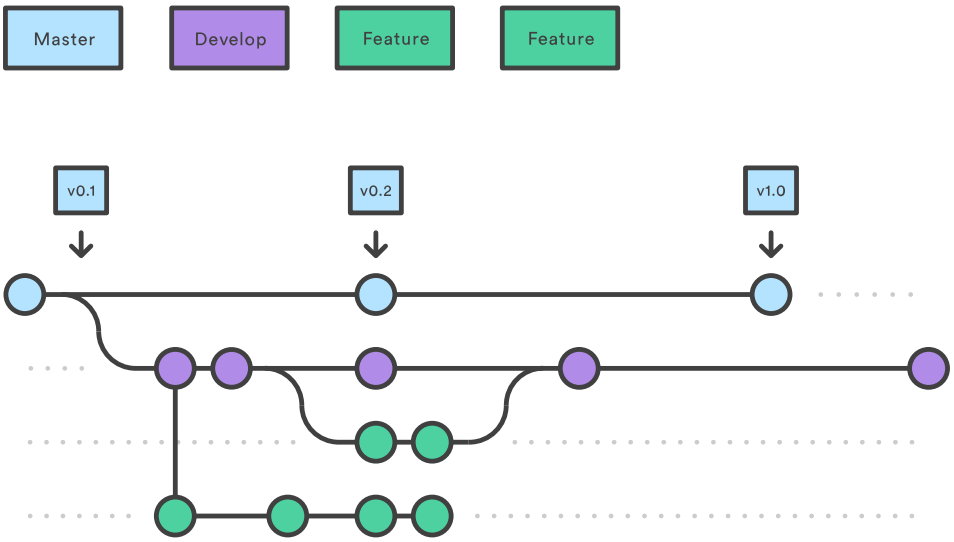

Рис. 2: Пример использования подхода Git Workflow

2.5.2 Git hooks

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

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

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

Listing 1: Настройки для Git Hooks

3 Описание проекта

3.1 Разработка дизайна

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

3.2 Авторизации через JSON Web Token (JWT)

3.2.1 Авторизация

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

3.2.2 Аутентификация

Аутентификация — процедура проверки подлинности данный.

3.2.3 JSON Web Token (JWT)

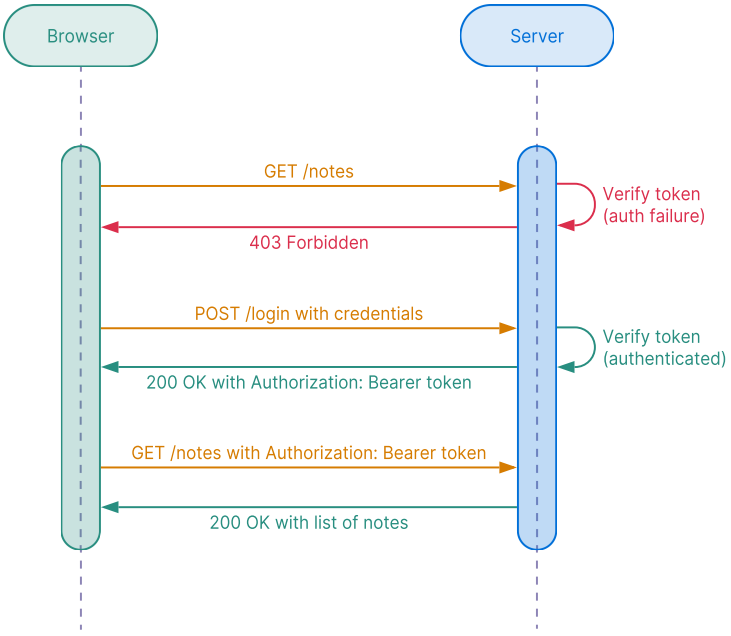

Рис. 3: Демонстрация работы JWT

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

Вот несколько сценариев, в которых полезен JWT:

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

JWT состоит из следующих частей:

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

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

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

3.2.4 Реализация

Авторизация проходит следущим образом:

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

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

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

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

Далее этот токет будет использоваться для доступа к защищёнными ресурсам.

3.3 Схема базы данных

Схема базы данных

4 Заключение

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

4.1 Недостатки

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

- Использование CI/CD.
- Использование методологий в вёрстке. Например, Block Element Modificator (BEM). Её разработали внутри компании Яндекс. У них есть свой стек технологий под данную методологию, который облегчает разработку клиентской части.
- Использование NoSQL баз данных вместе с реалиционными. Хранить JSON в таблице плохо, поэтому для этой задачи подходить *MongoDB* [10].
- Разделение клиентского кода на чанки.
- Микросервисная архитектура.

A Визуализации структуры проекта

```
 /db-course-project-app
--- .gitignore
--- package.json
--- .babelrc
--- jest.config.js
--- requirements.txt
--- .eslintignore
--- yarn.lock
--- .env
--- README.md
--- LICENSE
--- /util
------ nodemon.json
--- /tests
------ test_smoke.py
--- /src
------ .stylelintrc
------ config.js
------ index.js
------ webpack.config.js
------ .eslintrc.js
------ /client
--------- main.jsx
--------- admin.jsx
--------- /tests
---------- smoke.test.js
--------- /apps
---------- /main
---------- App.jsx
---------- /pages
---------- /ProfileSettings
---------- style.scss
```

---------- ProfileSettings.jsx
---------- index.js
---------- /SignUp
---------- style.scss
---------- index.js
---------- SignUp.jsx
---------- /TestEditor
---------- /Profile
---------- style.scss
---------- Profile.jsx
---------- index.js
---------- /ProfileAttempts
---------- style.scss
---------- ProfileAttempts.jsx
---------- index.js
---------- /Test
---------- /Home
---------- style.scss
---------- index.js
---------- Home.jsx
---------- /Login
---------- style.scss
---------- Login.jsx
---------- index.js
---------- /components
---------- /Header
---------- style.scss
---------- Header.jsx
---------- index.js
---------- /TestCard
---------- style.scss
---------- index.js
---------- TestCard.jsx
---------- /ListTestCards

```
---------------------- index.js
---------------------- ListTestCards.jsx
------------------ /Footer
---------------------- style.scss
---------------------- Footer.jsx
---------------------- index.js
--------------- /services
------------ /admin
--------------- Login.jsx
--------------- App.jsx
------ /routes
--------- main.js
--------- auth.js
------ /tests
--------- smoke.test.js
------ /controllers
--------- auth.js
------ /helpers
--------- FormListErrors.js
------ /templates
--------- admin.handlebars
--------- index.handlebars
--------- /layouts
------------ main.handlebars
--------- /partials
------------ favicon.handlebars
------------ meta.handlebars
------ /middlewares
--------- checkToken.js
------ /models
--------- UserRole.js
--------- Role.js
--------- /User
------------ constraints.js
```

```
------------ User.js
------------ index.js
------ /services
--- /db
------ init_db.sql
```

B Код проекта

db-course-project-app/package.json

```
{
  "name": "db-course-project-app",
  "version": "1.0.0",
  "description": "Web-application for course project by Database.",
  "main": "build/index.js",
  "author": "Dmitry Vasiliev",
  "scripts": {
    "test": "jest",
    "start-dev": "nodemon --config \"./util/nodemon.json\"/",
    "build": "rm -rf ./build/* && babel src -d build && webpack --config src/webpack.config.js
    ↪  --mode=\"production\"",
    "start": "node -r dotenv/config build/index.js",
    "es-lint": "eslint . -c src/.eslintrc.js --ext \"jsx,js\"",
    "style-lint": "stylelint --ignore-pattern src/client/tests --config src/.stylelintrc src/client/*",
    "watch": "webpack --config src/webpack.config.js --watch"
  },
  "dependencies": {
    "bcrypt": "^5.0.0",
    "bootstrap": "^4.5.0",
    "compression": "^1.7.4",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-handlebars": "^5.0.0",
    "http-status-codes": "^1.4.0",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.19",
    "morgan": "^1.10.0",
    "multer": "^1.4.2",
    "pg": "^8.3.0",
    "pg-hstore": "^2.3.3",
    "prop-types": "^15.7.2",
    "pug": "^3.0.0",
    "react": "^16.13.1",
    "react-bootstrap": "^1.2.2",
    "react-dom": "^16.13.1",
    "react-redux": "^7.2.1",
    "react-router-bootstrap": "^0.25.0",
    "react-router-dom": "^5.2.0",
    "react-router-prop-types": "^1.0.5",
    "redux": "^4.0.5",
    "sequelize": "^6.3.3",
    "serve-favicon": "^2.5.0"
  },
  "devDependencies": {
    "@babel/cli": "^7.11.6",
    "@babel/core": "^7.11.6",
    "@babel/node": "^7.10.5",
    "@babel/plugin-transform-runtime": "^7.11.5",
    "@babel/preset-env": "^7.11.5",
    "@babel/preset-react": "^7.10.4",
    "@types/compression": "^1.7.0",
    "@types/dotenv": "^8.2.0",
    "@types/express": "^4.17.7",
    "@types/express-handlebars": "^3.1.0",
    "@types/jest": "^26.0.4",
    "@types/jsonwebtoken": "^8.5.0",
    "@types/lodash": "^4.14.161",
    "@types/morgan": "^1.9.1",
    "@types/multer": "^1.4.4",
    "@types/node": "^14.0.22",
    "@types/react": "^16.9.43",
    "@types/react-dom": "^16.9.8",
    "@types/react-redux": "^7.1.9",
    "@types/react-router-bootstrap": "^0.24.5",
    "@types/react-router-dom": "^5.1.5",
    "@types/redux": "^3.6.0",
    "@types/sequelize": "^4.28.9",
    "@types/serve-favicon": "^2.5.0",
    "babel-loader": "^8.1.0",
    "css-loader": "^3.6.0",
```

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

db-course-project-app/.babelrc

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

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

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

db-course-project-app/requirements.txt

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

db-course-project-app/README.md

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

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

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

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

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

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

db-course-project-app/src/.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 checkToken from "./middlewares/checkToken";
import profileModify from "./routes/profileModify";

const app = express();

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

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

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

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

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

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

app.get("/test_token", checkToken, (req, res) => {
    res.send("hello! you can read this secure resource.");
});
```

```

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

app.use("*", mainRouter);

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

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

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

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

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

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

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

```
module.exports = {
    "env": {
        "browser": true,
        "es2020": true,
        "node": true,
        "jest": true
    },
    "extends": [
        "eslint:recommended",
        "plugin:react/recommended",
        "plugin:jest/recommended"
```

```
    ],
    "parserOptions": {
        "ecmaFeatures": {
            "jsx": true
        },
        "ecmaVersion": 11,
        "sourceType": "module"
    },
    "plugins": [
        "react",
        "jest"
    ],
    "rules": {
    }
};
```

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

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

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

import "bootstrap/scss/bootstrap.scss";

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

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

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

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

import "bootstrap/scss/bootstrap.scss";

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

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

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

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

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

```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 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";

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}/>
                    <Route component={() => <HttpErrorInfo status={NOT_FOUND} />} />
                </Switch>

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

export default App;
```

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

```scss

```

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

```jsx
import * as React from "react";
import Container from "react-bootstrap/Container";
import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";

import "./style.scss";

const ProfileSettings = () => {
    return (
        <Container className="p-3">
            <Form>
                <Form.Group controlId="main-signup-form__email">
                    <Form.Label className="main-signup-form__label">Email address:</Form.Label>
                    <Form.Control type="email"
                                  placeholder="Enter email"
                                  name="email" />
                </Form.Group>
                <Form.Group controlId="main-signup-form__login">
                    <Form.Label className="main-signup-form__label">Login:</Form.Label>
                    <Form.Control type="text"
                                  placeholder="Enter login"
                                  name="login" />
```

```
                </Form.Group>
                <Form.Group controlId="main-signup-form__password">
                    <Form.Label className="main-signup-form__label">Password:</Form.Label>
                    <Form.Control type="password"
                                  placeholder="Enter password"
                                  name="password" />
                </Form.Group>
                <Form.Group controlId="main-signup-form__repassword">
                    <Form.Label className="main-signup-form__label">Repeat password:</Form.Label>
                    <Form.Control type="password"
                                  placeholder="Enter password"
                                  name="repassword" />
                </Form.Group>
                <Button variant="primary"
                        type="submit"
                        block>Submit</Button>
            </Form>
        </Container>
    );
};

export default ProfileSettings;
```

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

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

export default ProfileSettings;
```

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

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

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

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

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

export default SignUp;
```

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

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

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

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

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

import "./style.scss";

const {
    MIN_PASSWORD_LENGTH,
```

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

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;
    }

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

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

        this.toggleLoadingState();

        try {
            await authService.signUp(formData);

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

        this.toggleLoadingState();
    }

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

    render() {
```

```
        const { listErrors, isLoading } = this.state;

        return (
            <Container className="p-3">
                <Col lg={{ offset: 3, span: 6 }}>
                    <h2 className="main-signup-form__title">Sign Up form</h2>
                    <Form>
                        <ErrorFormAlert listErrors={listErrors}
                                        show={listErrors.length !== 0}
                                        onHide={this.hideErrorAlert} />
                        <Form.Group controlId="main-signup-form__email">
                            <Form.Label className="main-signup-form__label">Email address:</Form.Label>
                            <Form.Control type="email"
                                          placeholder="Enter email"
                                          name="email"
                                          maxLength={MAX_EMAIL_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Form.Group controlId="main-signup-form__login">
                            <Form.Label className="main-signup-form__label">Login:</Form.Label>
                            <Form.Control type="text"
                                          placeholder="Enter login"
                                          name="login"
                                          minLength={MIN_LOGIN_LENGTH}
                                          maxLength={MAX_LOGIN_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Form.Group controlId="main-signup-form__password">
                            <Form.Label className="main-signup-form__label">Password:</Form.Label>
                            <Form.Control type="password"
                                          placeholder="Enter password"
                                          name="password"
                                          minLength={MIN_PASSWORD_LENGTH}
                                          maxLength={MAX_PASSWORD_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Form.Group controlId="main-signup-form__repeat-password">
                            <Form.Label className="main-signup-form__label">Repeat password:</Form.Label>
                            <Form.Control type="password"
                                          placeholder="Enter password"
                                          name="repeatPassword"
                                          minLength={MIN_PASSWORD_LENGTH}
                                          maxLength={MAX_PASSWORD_LENGTH}
                                          required
                                          onChange={this.handleInputChange} />
                        </Form.Group>
                        <Button variant="primary"
                                type="submit"
                                block
                                disabled={isLoading}
                                onClick={this.handleFormSubmit}>
                            { isLoading ? 'Loading...' : 'Submit' }
                        </Button>
                    </Form>
                </Col>
            </Container>
        );
    }
}

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

export default withRouter(SignUp);
```

db-course-project-app/src/client/apps/main/pages/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 { LoremIpsum } from "lorem-ipsum";
import { random } from "lodash";
import * as React from "react";
import PropTypes from "prop-types";
import { connect } from "react-redux";
import {NOT_FOUND} from "http-status-codes";

import { Switch, Route } from 'react-router-dom';
import { LinkContainer } from "react-router-bootstrap";
import Container from "react-bootstrap/Container";
import Col from "react-bootstrap/Col";
import Row from "react-bootstrap/Row";
import Nav from "react-bootstrap/Nav";
import ListTestCards from "../../components/ListTestCards";
import ProfileSettings from "../ProfileSettings";
import ProfileAttempts from "../ProfileAttempts";

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

const lorem = new LoremIpsum({
    sentencesPerParagraph: {
        max: 8,
        min: 4
    },
    wordsPerSentence: {
        max: 16,
        min: 4
    }
});


// TODO: Delete mock
const tests = [];

for (let i = 0; i < 7; i++) {
    tests.push({
        title: lorem.generateWords(random(1, 10)),
        description: lorem.generateWords(random(10, 50)),
        author: "Dmitry"
    });
}

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

```
                            <LinkContainer to="/profile/attempts">
                                <Nav.Link eventKey="attempts">My attempts</Nav.Link>
                            </LinkContainer>
                        </Nav.Item>
                        <Nav.Item>
                            <LinkContainer to="/profile/settings">
                                <Nav.Link eventKey="settings">Settings</Nav.Link>
                            </LinkContainer>
                        </Nav.Item>
                    </Nav>

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

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

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

    return { user };
}

const connectedProfile = connect(mapStateToProps)(Profile);

export { connectedProfile as Profile };
```

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

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

export default Profile;
```

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

```

```

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

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

import "./style.scss";

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

export default ProfileAttempts;
```

db-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;
```

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

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

```
  }

  &__title {
    text-align: center;
  }

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

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

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

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

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

import "./style.scss";

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

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

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

            listErrors: [],

            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);
```

```
62          formData.append('password', this.state.password);
63
64          return formData;
65      }
66
67      async handleFormSubmit(event) {
68          event.preventDefault();
69
70          const { history, dispatch } = this.props;
71          const formData = this._generateFormData();
72
73          this.toggleLoadingState();
74
75          try {
76              const { token, user } = await authService.signIn(formData);
77
78              localStorage.setItem('TOKEN', token);
79
80              dispatch(authActions.success(user));
81
82              history.push('/');
83          } catch ({ errors }) {
84              this.setState({ listErrors: errors });
85          }
86
87          this.toggleLoadingState();
88      }
89
90      hideErrorAlert() {
91          this.setState({
92              listErrors: []
93          });
94      }
95
96      toggleLoadingState() {
97          this.setState(prev => {
98              return {
99                  isLoading: !prev.isLoading
100             }
101         });
102     }
103
104     render() {
105         const { listErrors, isLoading } = this.state;
106
107         return (
108             <Container className="p-3">
109                 <Col lg={{offset: 3, span: 6}}>
110                     <h2 className="main-login-form__title">Login form</h2>
111                     <Form>
112                         <ErrorFormAlert listErrors={listErrors}
113                                         show={listErrors.length !== 0}
114                                         onHide={this.hideErrorAlert} />
115                         <Form.Group controlId="main-login-form__login">
116                             <Form.Label className="main-login-form__label">Login:</Form.Label>
117                             <Form.Control type="text"
118                                           placeholder="Enter login"
119                                           name="login"
120                                           minLength={MIN_LOGIN_LENGTH}
121                                           maxLength={MAX_LOGIN_LENGTH}
122                                           required
123                                           onChange={this.handleInputChange} />
124                         </Form.Group>
125                         <Form.Group controlId="main-login-form__password">
126                             <Form.Label className="main-login-form__label">Password:</Form.Label>
127                             <Form.Control type="password"
128                                           placeholder="Enter password"
129                                           name="password"
130                                           minLength={MIN_PASSWORD_LENGTH}
131                                           maxLength={MAX_PASSWORD_LENGTH}
132                                           required
133                                           onChange={this.handleInputChange}/>
134                         </Form.Group>
135                         <Form.Group controlId="main-login-form__checkbox">
136                             <Form.Check type="checkbox"
137                                         label="Remember me"
```

```
                                        name="readMeChecked"
                                        onChange={this.handleCheckboxChange} />
                            </Form.Group>
                            <Button variant="primary"
                                    type="submit"
                                    block
                                    disabled={isLoading}
                                    onClick={this.handleFormSubmit}>
                                { isLoading ? 'Loading...' : 'Submit' }
                            </Button>
                            <LinkContainer to="#">
                                <a className="main-login-form__forgot-password">Forgot password?</a>
                            </LinkContainer>
                        </Form>
                    </Col>
                </Container>
            );
        }
}

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

const connectedLogin = connect()(Login);

export { connectedLogin as Login };
```

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

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

export default Login;
```

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

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

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

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

import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
import NavDropdown from "react-bootstrap/NavDropdown";
import Modal from "react-bootstrap/Modal";
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 onLogOut = () => {
```

```
        dispatch(authActions.logOut());

        setModalShow(false);

        history.push("/");
    };

    return (
        <Navbar bg="dark" variant="dark">
            <LinkContainer to="/">
                <Navbar.Brand>PassQuiz</Navbar.Brand>
            </LinkContainer>
            <Nav>
                { !isLoggedIn ? (
                    <>
                        <LinkContainer to="/signup">
                            <Nav.Link>Sign Up</Nav.Link>
                        </LinkContainer>
                        <LinkContainer to="/login">
                            <Nav.Link>Login</Nav.Link>
                        </LinkContainer>
                    </>
                ) : (
                    <NavDropdown title={ user.login } id="user-nav-dropdown">
                        <LinkContainer to="/profile">
                            <NavDropdown.Item>My profile</NavDropdown.Item>
                        </LinkContainer>
                        <NavDropdown.Divider />
                        <LinkContainer to="#">
                            <NavDropdown.Item onClick={showModal}>Logout</NavDropdown.Item>
                        </LinkContainer>
                    </NavDropdown>
                ) }
            </Nav>

            <Modal
                size="lg"
                aria-labelledby="contained-modal-title-vcenter"
                centered
                onHide={onLogOut}
                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;
```

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

```
.test-card {
  margin: 15px 0;
  position: relative;

  &__description {
    max-height: 170px;
  }

  &__title {
    font-size: 1.1rem;
    max-height: 60px;
  }

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

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

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

export default TestCard;
```

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

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

import "./style.scss";

const TestCard = ({ title, description, author }) => {
    return (
        <Card className="test-card">
            <Card.Body>
                <Card.Title className="test-card__title">{title}</Card.Title>

                <p className="test-card__author">
                    Author: <LinkContainer to="#">
                                <a className="test-card__author-name">{author}</a>
                            </LinkContainer>
                </p>

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

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

                    <Dropdown className="test-card__dropdown-menu">
                        <Dropdown.Toggle variant="primary">Menu</Dropdown.Toggle>
                        <Dropdown.Menu>
                            <Dropdown.Item as="button">Edit</Dropdown.Item>
                            <Dropdown.Item as="button">Delete</Dropdown.Item>
                            <Dropdown.Item as="button">Share</Dropdown.Item>
                        </Dropdown.Menu>
```

```
                    </Dropdown>
                </div>
            </Card.Body>
        </Card>
    );
};

TestCard.propTypes = {
    title: PropType.string,
    description: PropType.string,
    author: PropType.string
};

export default TestCard;
```

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

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

export default ListTestCards;
```

db-course-project-app/src/client/apps/main/components/ListTestCards/ListTestCards.j

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

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

import "./style.scss";

const ListTestCards = ({ tests }) => {
    return (
        <Container className="p-3">
            <Row>
                {
                    tests.map((el, i) => {
                        const { title, description, author } = el;

                        return (
                            <Col lg={12} key={i} className="list-tests__col">
                                <TestCard title={title} description={description} author={author} />
                            </Col>
                        );
                    })
                }
            </Row>
        </Container>
    );
};

ListTestCards.propTypes = {
    tests: PropTypes.array
};

export default ListTestCards;
```

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

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

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

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

import "./style.scss"

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

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

export default Footer;
```

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

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

export default Footer;
```

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

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

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

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

export default Login;
```

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

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

import Container from 'react-bootstrap/Container';

import Login from "./Login";

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

```
                    </Switch>
                </Container>
            </div>
        );

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

export default App;
```

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

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

const router = new Router();

router.get("*", (req, res) => {
    res.render("index");
});

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/tests/smoke.test.js

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

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

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

export const signUp = async (req, res) => {
    const {
        login,
        repeatPassword,
```

```
        password,
        email,
    } = req.body;
    const formListErrors = new FormListErrors();

    try {
        await User.create(
            { email, login, password },
            { repeatPassword });
    } catch (ex) {
        formListErrors.addFromModelErrors(ex.errors);

        res.status(BAD_REQUEST).json(formListErrors.data);
    }

    res.sendStatus(OK);
};

export const signIn = async (req, res) => {
    const {
        login,
        password,
    } = req.body;
    const formListErrors = new FormListErrors();

    let user;
    try {
        user = await User.findOne({ where: { login } });
    } catch (error) {
        formListErrors.addDefault();

        res.status(INTERNAL_SERVER_ERROR).json(formListErrors.data);
    }

    if (!user) {
        formListErrors.add("user with such name not found.");

        res.status(BAD_REQUEST).json(formListErrors.data);
    } 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");

            res.status(BAD_REQUEST).json(formListErrors.data);
        }
    }
};

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

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

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

```
// TODO: add documentation

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

    addFromModelErrors(errors) {
        this.data.errors.push(
            ...errors.map(err => {
```

```
                let { message } = err;

                return { message };
            })
        );
    }

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

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

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

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

<script src="/static/admin.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/meta.handlebars

```
<meta charset="UTF-8">
<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
3  export default async (req, res, next) => {
4      const token = req.headers["authorization"];
5
6      let tokenObj = null;
7      try {
8          tokenObj = await jwt.verify(token, process.env.JWT_SECRET);
9      } catch (err) {
10         console.error(err);
11     }
12
13     req.user_id = tokenObj.sub;
14
15     next();
16 };
```

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

```
1  import { Sequelize } from "sequelize";
2
3  import config from "../config";
4
5  import User from "./User";
6  import Role from "./Role";
7
8  const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
9
10 // setup many to many
11 const UserRole = sequelize.define("user_role", {});
12
13 User.belongsToMany(Role, { through: UserRole });
14 Role.belongsToMany(User, { through: UserRole });
15
16 export default UserRole;
```

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

```
1  import { Sequelize } from "sequelize";
2
3  import config from "../config";
4
5  const sequelize = new Sequelize(process.env.DATABASE_URL, config.db.options);
6
7  const Role = sequelize.define("role", {
8      id: {
9          type: Sequelize.INTEGER.UNSIGNED,
10         autoIncrement: true,
11         primaryKey: true
12     },
13     role_name: {
14         type: Sequelize.STRING(64),
15         allowNull: false,
16         unique: true
17     }
18 });
19
20 export default Role;
```

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,
    ValidationError,
    ValidationErrorItem
} from "sequelize";
import * as bcrypt from "bcrypt";

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

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

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

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

```
            validate: {
                isEmail: {
                    msg: 'invalid email address.'
                }
            }
        },
    }
);

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

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

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

User.prototype.initState = function() {
    return {
        user: {
            login: this.login
        }
    }
}

User.beforeCreate(async (user, options) => {
    const { repeatPassword } = options;

    if (repeatPassword !== user.password) {
        const validErr = new ValidationError();
        const validItemErr = new ValidationErrorItem("passwords doesn't equal.")

        validErr.errors.push(validItemErr);

        throw validErr;
    }

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

export default User;
```

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

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

export default User;
```

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

```
CREATE TABLE IF NOT EXISTS users (
   user_id SERIAL NOT NULL,
   login VARCHAR(255) NOT NULL UNIQUE,
   password VARCHAR(255) NOT NULL,
   email VARCHAR(255) NOT NULL,
   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] *Python*. URL: https://www.python.org.

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

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

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

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

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

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

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

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