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: 15 сентября 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* [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* [10] — среда выполнения JavaScript, созданная на основе движка Chrome V8 JavaScript.
- *Express* [4] — минимальный и гибкий *Node.js* [10] фреймворк для создания веб-приложений.
- *Python* [12] — язык программирования, который позволяет быстро работать и более эффективно интегрировать системы.
- *PostgreSQL* [11] — объектно-реляционная база данных с открытым исходным кодом.
- *Sequelize* [18] — *Node.js* [10] ORM на основе обещаний для Postgres *PostgreSQL* [11].

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* [19] — проверяет качество кода на *SCSS* [16], *CSS* [1].
- Тестирующие фреймворки:
  - *Jest* [9] — среда тестирования *JavaScript* [8] с упором на простоту.
  - *Selenium with Python* [17] — предоставляют простой API для написания тестов с использованием Selenium WebDriver.

Webpack [20] — сборщик статических модулей для современных *JavaScript* [8] приложений.

2 Инфраструктура проекта

2.1 Архитекрутра

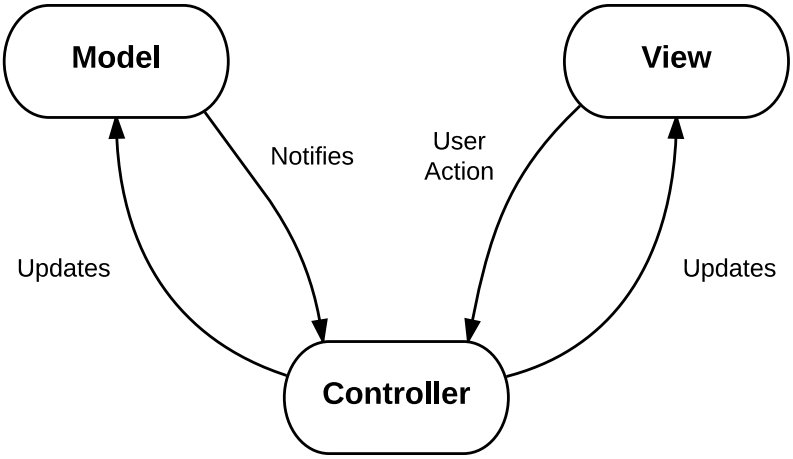

Рис. 1: Визуализация архитектуры MVC

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* [18] для сервера и *Redux* [15] для клиента.
- **View** — *React* [13]
- **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 • 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.

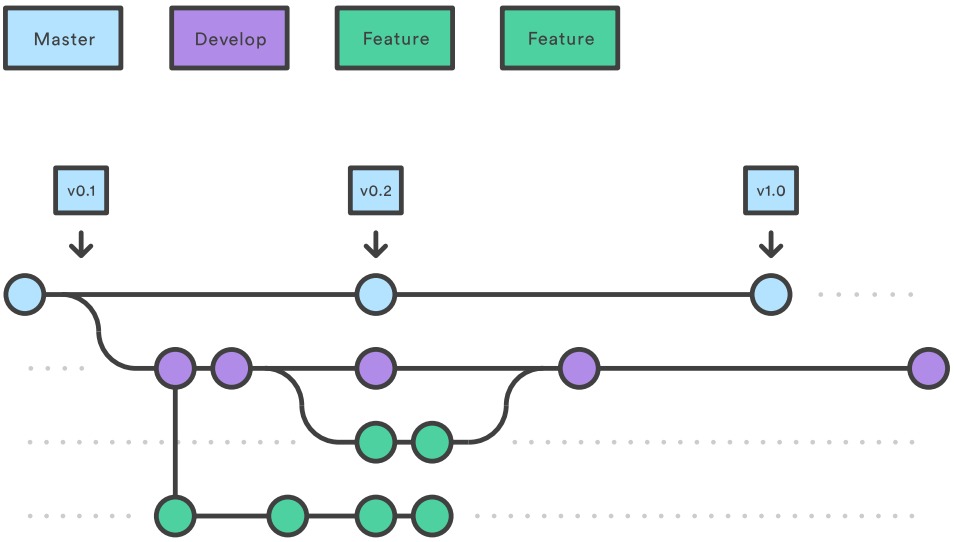

Рис. 2: Пример использования подхода Git Workflow

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 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.

3.3 Аутентификация

Аутентификация — процедура проверки подлинности данный.

3.3.1 JSON Web Token (JWT)

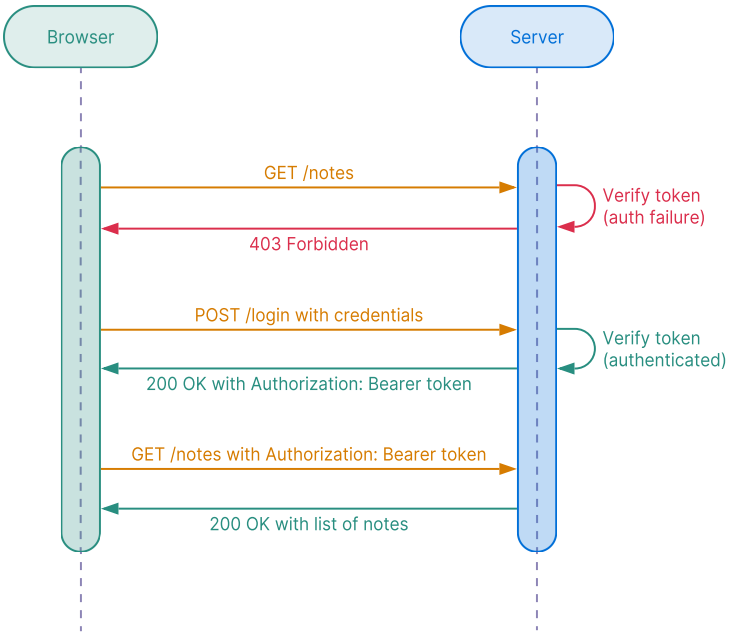

Рис. 3: Демонстрация работы JWT

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);
```

4 Заключение

Благодаря данной курсовому проекту, я поверхностно освоил SPA приложений с помощью *React* [13] и *Express* [4].

4.0.1 Недостатки

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

- CI/CD — объединяет разработку, развёртывание и команду, ускоряя процесс сборки, тестирования и развёртывания приложения.
- Неиспользование методологий в вёрстке

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
---------- Home.jsx
---------- SignUp.jsx
---------- /pages
```

```
------------------ /Login
---------------------- style.scss
---------------------- Login.jsx
---------------------- index.js
------------ /admin
--------------- Login.jsx
--------------- App.jsx
------ /routes
--------- auth.js
------ /tests
--------- smoke.test.js
------ /controllers
--------- auth.js
------ /templates
--------- admin.handlebars
--------- index.handlebars
--------- /layouts
------------ main.handlebars
--------- /partials
------------ favicon.handlebars
------------ meta.handlebars
------ /middlewares
--------- checkToken.js
--------- authErrorHandler.js
------ /models
--------- User.js
--------- UserRole.js
--------- Role.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/* && 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",
    "pg": "^8.3.0",
    "pg-hstore": "^2.3.3",
    "pug": "^3.0.0",
    "react": "^16.13.1",
    "react-bootstrap": "^1.2.2",
    "react-dom": "^16.13.1",
    "react-router-bootstrap": "^0.25.0",
    "react-router-dom": "^5.2.0",
    "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/node": "^14.0.22",
    "@types/react": "^16.9.43",
    "@types/react-dom": "^16.9.8",
    "@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",
    "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 = {
  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
```

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

```python
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 authErrorHandler from "./middlewares/authErrorHandler";

import authRouter from "./routes/auth";
import checkToken from "./middlewares/checkToken";

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("/", (req, res) => {
    res.render("index");
});

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(authErrorHandler);

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

```
61
62          console.log('Connection has been established successfully.');
63      } catch (error) {
64          console.error('Unable to connect to the database:', error);
65      }
66
67      console.log(`Server started on port: ${PORT}.`);
68  });
```

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

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

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

```
1  module.exports = {
2      "env": {
3          "browser": true,
4          "es2020": true,
5          "node": true,
6          "jest": true
7      },
8      "extends": [
9          "eslint:recommended",
10         "plugin:react/recommended",
11         "plugin:jest/recommended"
12     ],
13     "parserOptions": {
14         "ecmaFeatures": {
15             "jsx": true
16         },
17         "ecmaVersion": 11,
18         "sourceType": "module"
19     },
20     "plugins": [
21         "react",
```

```
        "jest"
    ],
    "rules": {
    }
};
```

db-course-project-app/src/client/main.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/main/App";

ReactDOM.render(
    <BrowserRouter>
        <App />
    </BrowserRouter>,
    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 "./pages/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

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

import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
import { LinkContainer } from 'react-router-bootstrap';

import Login from "./pages/Login";
import Home from "./Home"
import SignUp from "./SignUp";

class App extends React.Component {
    render() {
        const App = () => (
            <div>
                <Navbar bg="light">
                    <LinkContainer to="/">
```

```
                        <Navbar.Brand>Home</Navbar.Brand>
                    </LinkContainer>
                    <Nav>
                        <LinkContainer to="/signup">
                            <Nav.Link>Sign Up</Nav.Link>
                        </LinkContainer>
                    </Nav>
                </Navbar>
                <Switch>
                    <Route exact path='/' component={Home}/>
                    <Route path='/login' component={Login}/>
                    <Route path='/signup' component={SignUp}/>
                </Switch>
            </div>
        );

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

export default App;
```

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

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

import { LinkContainer } from "react-router-bootstrap";
import Button from "react-bootstrap/Button";
import Jumbotron from "react-bootstrap/Jumbotron";
import Container from "react-bootstrap/Container";

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>
                <p>
                    <LinkContainer to="/login">
                        <Button variant="primary">Sign In</Button>
                    </LinkContainer>
                </p>
            </Jumbotron>
        </Container>
    );
}

export default Home;
```

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

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

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

export default class SignUp extends React.Component {
    render() {
        return (
            <Container className="p-3">
                <Form>
                    <Form.Group controlId="formBasicEmail">
                        <Form.Label>Email address</Form.Label>
                        <Form.Control type="email" placeholder="Enter email" name="email" />
```

```
                    </Form.Group>
                    <Form.Group controlId="formBasicLogin">
                        <Form.Label>Login</Form.Label>
                        <Form.Control type="text" placeholder="Enter login" name="login" />
                    </Form.Group>
                    <Form.Group controlId="formBasicPassword">
                        <Form.Label>Password</Form.Label>
                        <Form.Control type="password" placeholder="Password" name="password" />
                    </Form.Group>
                    <Button variant="primary" type="submit">Submit</Button>
                </Form>
            </Container>
        );
    }
}
```

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;
  }
}
```

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

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

import Form from "react-bootstrap/Form";
import Button from "react-bootstrap/Button";
import Container from "react-bootstrap/Container";
import LinkContainer from "react-router-bootstrap/lib/LinkContainer";

import "./style.scss";

const Login = () => {
    return (
        <Container className="p-3 col-lg-4 offset-lg-4">
            <h2 className="main-login-form__title">Login form</h2>
            <Form>
                <Form.Group controlId="main-login-form">
                    <Form.Label>Login:</Form.Label>
                    <Form.Control type="text"
                                  placeholder="Enter login"
                                  name="login"/>
                </Form.Group>
                <Form.Group controlId="main-form-password">
                    <Form.Label>Password:</Form.Label>
                    <Form.Control type="password"
                                  placeholder="Enter password"
                                  name="password"/>
                </Form.Group>
                <Form.Group controlId="main-form-checkbox">
                    <Form.Check type="checkbox" label="Remember me"/>
                </Form.Group>
                <Button variant="primary"
                        type="submit"
                        block>Submit</Button>
                <LinkContainer to="#">
                    <a className="main-login-form__forgot-password">Forgot password?</a>
                </LinkContainer>
            </Form>
        </Container>
    );
};

```

```
export default 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/admin/Login.jsx

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

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

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

export default Login;
```

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

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

import Container from 'react-bootstrap/Container';

import Login from "./Login";

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

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

export default App;
```

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

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

const router = new Router();

router.post('/signup', authController.signup);
router.post('/signin', authController.signin);

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 {BAD_REQUEST, FORBIDDEN} from "http-status-codes";
import * as jwt from "jsonwebtoken";

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

export const signup = async (req, res, next) => {
    const { login, ...credentials } = req.body;

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

        res.status(FORBIDDEN);
    }

    if (user !== null) {
        next({
            status: BAD_REQUEST,
            message: "User with such name already exists"
        });
    } else {
        const password = await User.hashPassword(credentials.password);

        await User.create({ ...credentials, password, login });

        res.json("success");
    }
};

export const signin = async (req, res, next) => {
    const { login, ...credentials } = req.body;

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

        res.status(FORBIDDEN);
    }

    if (!user) {
        next({
            status: BAD_REQUEST,
            message: "User with such name not found"
        });
    } else {
```

```
        const isRightPassword = await user.comparePasswords(credentials.password);

        if (isRightPassword) {
            const token = jwt.sign({
                userId: user.id,
                role: ['user'] }, process.env.JWT_SECRET);

            res.send(token);
        } else {
            next({
                status: BAD_REQUEST,
                message: "Login or password is invalid"
            });
        }
    }
};
```

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="ru">
<head>
    {{> meta }}
    {{> favicon }}
    <title>passquiz</title>
</head>
<body>
    {{{ 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"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
      ↪  minimum-scale=1.0">
```

```
<meta http-equiv="X-UA-Compatible" content="ie=edge">
```

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

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

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

    const tokenObj = await jwt.verify(token, process.env.JWT_SECRET);

    console.log(tokenObj);

    next();
};
```

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

```
export default (err, req, res) => {
    const { status } = err;

    res.status(status).json(err);
}
```

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

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

import config from "../config";

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(128),
            allowNull: false,
            unique: true
        },
        password: {
            type: Sequelize.STRING(128),
            allowNull: false,
        },
        email: {
            type: Sequelize.STRING(128),
            allowNull: false,
        },
    }
);

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);
};

export default User;
```

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

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

import config from "../config";

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

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

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

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

export default UserRole;
```

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

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

import config from "../config";

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

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

export default Role;
```

db-course-project-app/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] *Node.js*. URL: https://nodejs.org.

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

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

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

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

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