УДК 004.428.4
DOI 10.17513/snt.39626

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АВТОМАТИЗАЦИИ ВЕДЕНИЯ БАЛЛЬНО-РЕЙТИНГОВОЙ СИСТЕМЫ ВУЗА

[1]Грязных И.Д., [1,2,3]Ананченко И.В., [1]Гайков А.В.

[1]ФГБОУ ВО «Санкт-Петербургский государственный технологический институт (технический университет)», Санкт-Петербург, e-mail: gryaznykh.ivan@gmail.com, av489@yandex.ru;

[2]ФГАОУ ВО «Национальный исследовательский университет ИТМО», Санкт-Петербург;

[3]ФГБОУ ВО БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова, Санкт-Петербург, e-mail: anantchenko@yandex.ru

Рассматривается процесс разработки программного обеспечения (ПО) для автоматизации балльно-рейтинговой системы (БРС) высшего учебного заведения. На момент подготовки публикации в открытом доступе отсутствует специализированное программное обеспечение для поддержки БРС учебных заведений. Использование для этих целей возможностей бесплатных систем электронного обучения (например, LMS Moodle) достаточно часто не является оптимальным решением, так как такие системы в первую очередь ориентированы на управление электронными курсами и их достаточно трудно настроить для поддержки всех функций БРС. Требования к ПО БРС могут различаться и зависеть от использующей БРС организации, но, безусловно, должны учитывать такие важные составляющие, как быстродействие ПО, надежность и гибкость. Рассмотрены ключевые этапы разработки ПО БРС, начиная от этапа проектирования и заканчивая созданием пользовательского интерфейса. Были проанализированы инструменты, необходимые для создания системы, такие как фреймворки, библиотеки и инструменты для работы с базами данных. По итогам анализа выбраны средства разработки: React для создания пользовательского интерфейса и фронтенд-части системы, Koajs для бэкенд-части системы и обработки запросов, MySql для управления данными и Prisma ORM для упрощения работы с базой данных. С учетом того, что современная реализация ПО БРС является клиент-серверной системой, а пользователи-клиенты могут подключаться с использованием различных мобильных и стационарных устройств, приводятся рекомендации по созданию эффективного и масштабируемого веб-приложения. Исходный код разработанного программного проекта БРС доступен на GitHub в свободном доступе и может быть использован желающими не только для развертывания и применения предлагаемой БРС, но и для разработки собственного ПО БРС на предложенной основе. Приводится информация о развертывании и тестировании ПО БРС на общедоступном интернет-хостинге.

Ключевые слова: балльно-рейтинговая система, проектирование базы данных, React, KoaJS, Prisma ORM

DEVELOPMENT OF SOFTWARE FOR AUTOMATING THE MANAGEMENT OF THE SCORE-RATING SYSTEM OF THE UNIVERSITY

[1]Gryaznykh I.D., [1,2,3]Ananchenko I.V., [1]Gaykov A.V.

[1]Saint Petersburg State Technological Institute (Technical University), Saint-Petersburg, e-mail: gryaznykh.ivan@gmail.com, av489@yandex.ru;

[2]Saint-Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint-Petersburg;

[3]Baltic State Technical University "VOENMEH" named after D.F. Ustinov, Saint-Petersburg, e-mail: anantchenko@yandex.ru

The process of software development for automation of the point-rating system (BRS) of a higher educational institution is considered. At the time of preparation of the publication, there is no specialized software in the public domain to support the BRS of educational institutions. Using the capabilities of free e-learning systems (for example, LMS Moodle) for these purposes is often not the optimal solution, since such systems are primarily focused on managing e-courses and it is difficult enough to configure them to support all the functions of the BRS. The requirements for BRS software may vary and depend on the organization using BRS, but, of course, they must take into account such important components as software performance, reliability and flexibility. The key stages of BRS software development are considered, starting from the design stage and ending with the creation of the user interface. The tools needed to create the system, such as frameworks, libraries and tools for working with databases, were analyzed. Based on the results of the analysis, the following development tools were selected: React for creating the user interface and the frontend part of the system, Koajs for the backend part of the system and query processing, MySQL for data management and Prisma ORM to simplify working with the database. Taking into account the fact that the modern implementation of the BRS software is a client-server system, and client users can connect using various mobile and stationary devices, recommendations are given for creating an efficient and scalable web application. The source code of the developed BRS software project is freely available on GitHub and can be used by those who wish not only to deploy and use the proposed BRS, but also to develop their own BRS software on the proposed basis. Information is provided on the deployment and testing of BRS software on a public Internet hosting.

Keywords: point-rating system, database design, React, KoaJS, Prisma ORM

Использование балльно-рейтинговых систем является важным инструментом оценки эффективности деятельности в различных областях, в том числе в образовании. Балльно-рейтинговая система позволяет оценить качество работы учебных заведений, преподавателей и студентов, а также обеспечивает стимулирование участников образовательного процесса к улучшению своих результатов. Балльно-рейтинговая система может помочь в развитии конкурентной среды между учебными заведениями и способствовать улучшению качества образования в целом, так как создание прозрачной системы оценки результатов обучения ведет к повышению мотивации студентов и учебных заведений для достижения лучших результатов. Отмечаем, что разработка балльно-рейтинговой системы для университета требует тщательного анализа и выбора критериев, которые будут использоваться для оценки. Критерии должны быть объективными и учитывать различные аспекты деятельности университета, такие как академические достижения, инновационные исследования, социальная ответственность и др. Кроме того, при разработке балльно-рейтинговой системы для вуза необходимо учитывать индивидуальные особенности учебного заведения, такие как направления подготовки обучающихся, задачи, структура и т.д. [1]. Только в этом случае система будет действительно эффективной и позволит оценить качество деятельности университета в целом [2].

Цель разработки – создание балльно-рейтинговой системы для университета, обеспечивающей объективную оценку успеваемости студентов и эффективности учебного процесса. Указанная цель предполагает решение следующих задач:

1) разработать архитектуру программного комплекса (ПК), реализующего балльно-рейтинговую систему и ее функционал, учитывая потребности университета и стандарты образования;

2) создать систему авторизации и доступа к балльно-рейтинговой системе с учетом безопасности и конфиденциальности персональных данных;

3) реализовать механизмы сбора и анализа данных об успеваемости студентов;

4) обеспечить интерфейсы для управления ПК балльно-рейтинговой системы администраторами и использования его студентами и преподавателями;

5) обеспечить масштабируемость и поддержку балльно-рейтинговой системы на протяжении всего времени ее использования.

Выбор инструментов для разработки ПО БРС

Для разработки ПО БРС рассматривались современные языки программирования, фреймворки, библиотеки и инструменты для фронтенда, бэкенда и работы с базами данных [3]. Были выбраны:

1) ***React*** – для создания интерфейса пользователя и фронтенд-части системы;

2) ***Koajs*** – для создания бэкенд-части системы и обработки запросов;

3) ***Mysql*** – для хранения и управления данными системы;

4) ***Prisma ORM*** – для упрощения работы с базой данных и уменьшения количества необходимого кода.

Инструменты были выбраны исходя из того, что создаваемый программный код БРС должен обеспечивать надлежащую производительность, удобство использования, масштабируемость и надежность [4].

1. *Производительность.* React позволяет создавать быстрые и отзывчивые пользовательские интерфейсы благодаря использованию виртуального DOM и оптимизации рендеринга. Koajs обеспечивает быстрое и эффективное обслуживание запросов благодаря использованию асинхронных функций и генераторов. Mysql – реляционная база данных, имеет оптимизированный движок и обеспечивает быстрое выполнение запросов. Prisma ORM позволяет быстро и эффективно работать с базой данных благодаря использованию генерации кода и оптимизации запросов.

2. *Удобство использования.* React использует компонентный подход, делающий код более читаемым и поддерживаемым. Koajs использует цепочку обработчиков, что делает код более читаемым и удобным для отладки. Mysql имеет простой интерфейс для работы с БД и широкие возможности по настройке безопасности и защите от несанкционированного доступа. Prisma ORM генерирует необходимый код автоматически, что позволяет сократить количество необходимого кода и упрощает работу с базой данных.

3. *Масштабируемость.* React позволяет легко добавлять новые компоненты и управлять состоянием приложения. Koajs позволяет легко масштабировать API, обрабатывать запросы в режиме реального времени. Mysql позволяет легко добавлять новые таблицы и поля в базу данных и обрабатывать большой объем данных. Prisma ORM позволяет легко масштабировать работу с базой данных и генерировать необходимый код автоматически.

4. *Надежность*. React использует строгую типизацию, что позволяет выявлять ошибки на ранней стадии разработки. Koajs использует механизмы обработки ошибок, что позволяет быстро и эффективно реагировать на возникающие проблемы. Mysql имеет высокую степень надежности и защиты от сбоев и потери данных. Prisma ORM обеспечивает высокую степень надежности при работе с базой данных и упрощает процесс обновления и изменения данных.

Процесс разработки

Наиболее очевидная задача, которую необходимо выполнить при создании вебресурса – построение структуры базы данных. Структурирование данных является одним из важных аспектов проектирования БД, имеет важное значение для обеспечения эффективного хранения и обработки информации. Одним из ключевых инструментов для достижения этой цели являются нормальные формы, которые определяют, каким образом данные должны быть структурированы и организованы в базе данных. Нормальные формы являются формальными правилами, которые определяют, какие типы зависимостей должны существовать между атрибутами таблицы, чтобы база данных была эффективной, надежной и безопасной. Созданная структура, соответствующая первым 3 нормальным формам, представлена на рис. 1.

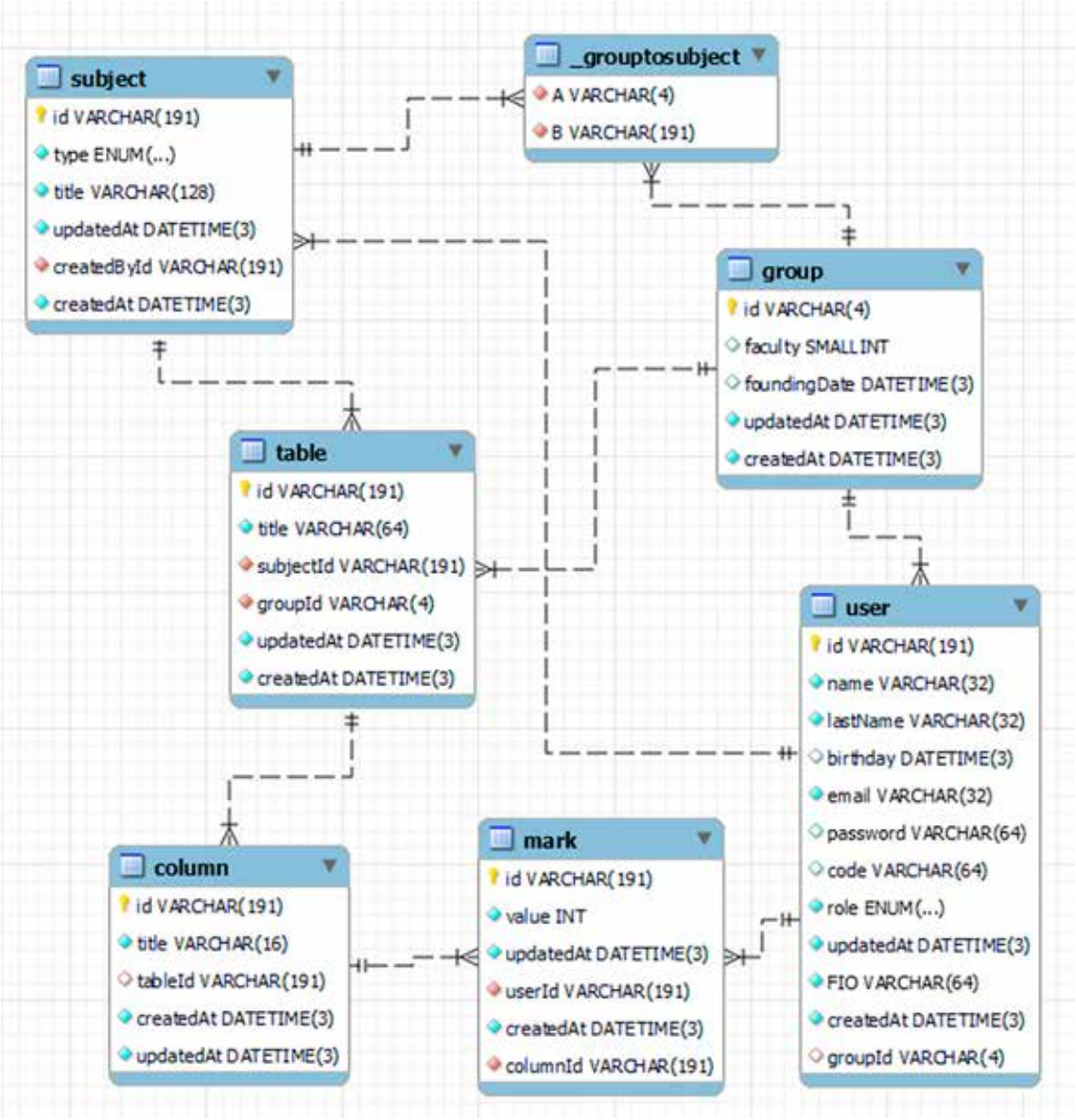

Рис. 1. ER диаграмма базы данных

Следующий этап – построение структуры и разработка серверной логики для эффективной и масштабируемой работы. Оптимальная, грамотно разработанная структура проекта облегчает сопровождение, упрощает сопровождения и добавление новых функций. Также упрощает отладку и тестирование проекта в целом, так как все файлы и функции имеют четко определенные места в структуре проекта. Желаемой масштабируемости и эффективности удалось добиться путем использования контроллеров и роутинговой структуры. Контроллеры представляют собой модули, отвечающие за обработку запросов, обращения к базе данных и формирование ответов. Роутинговая структура определяет, какие запросы должны быть переданы на обработку соответствующим контроллерам. Вместе они обеспечивают правильную организацию кода, что позволяет создавать сложные серверные приложения, которые могут легко масштабироваться и поддерживаться в долгосрочной перспективе. Были написаны промежуточные обработчики (middleware), которые позволили создать разделение прав на запросы, определяя, какие запросы могут быть обработаны только авторизованными пользователями, а какие являются общедоступными. Пример обработчика, фильтрующего запросы от пользователей, не имеющих соответствующей роли:

```
import jwt from ‘jsonwebtoken’;
import { Context, Next } from ‘koa’;
import { UserRoles } from ‘../types/requestTypes’;
export default (roles: (keyof typeof UserRoles)[]) => {
    return async (ctx: Context, next: Next) => {
        try {if (ctx.request.user === undefined) {
                return ctx.throw(401, “Token was not provided”);
            }
            if (roles.includes(ctx.request.user.role) === false) {
                return ctx.throw(403, “Forbidden”);           }
   return next();        } catch(e: any) {ctx.throw(e.status, e.message); }}}
```

Для использования обработчика остается указывать при роутинге, какие роли допускаются для обработки запросов сервером. Пример нескольких роутов с ограничением доступа:

```
const router = new Router();
// users page routes
router.put(“/createUser”, roleFilter([UserRoles.ADMIN]), adminController.createUser);
router.post(“/changeUserSettings/:id”, roleFilter([UserRoles.ADMIN]), adminController.
changeUserSettings);
router.delete(“/removeUser/:id”, roleFilter([UserRoles.ADMIN]), adminController.re-
moveUser);
// group page routes
router.get(“/group/search”, roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]), group-
Controller.getGroupsBySearch);
router.get(“/group/getUserGroup”, groupController.getUserGroup);
router.put(“/group/create”, roleFilter([UserRoles.ADMIN]), groupController.creat-
eGroup)
router.get(“/group/:id”, groupController.getGroupById);
router.post(“/group/:id/addStudentToGroup”, roleFilter([UserRoles.HEADMAN, UserRoles.
ADMIN, UserRoles.HEADMAN]), groupController.addStudentToGroup);
router.post(“/group/:id/removeStudentFromGroup”, roleFilter([UserRoles.HEADMAN, User-
Roles.ADMIN, UserRoles.HEADMAN]), groupController.removeStudentFromGroup);
router.post(“/group/:id/change”, roleFilter([UserRoles.ADMIN]), groupController.
changeGroupSettings);
router.delete(“/group/:id/remove”, roleFilter([UserRoles.ADMIN]), groupController.
removeGroup);
// subject page routes
router.get(“/subject/search”, subjectController.getSubjectBySearch)
router.put(“/subject/create”, roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]), sub-
jectController.createSubject)
router.get(“/subject/:id”, subjectController.getSubjectById)
router.post(“/subject/:id/addGroupToSubject”, roleFilter([UserRoles.ADMIN, UserRoles.
```

```
TEACHER]), subjectController.addGroupToSubject)
router.post("/subject/:id/removeGroupFromSubject", roleFilter([UserRoles.ADMIN, User-
Roles.TEACHER]), subjectController.removeGroupFromSubject)
router.post("/subject/:id/change", roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]),
subjectController.changeSubjectSettings)
router.delete("/subject/:id/remove", roleFilter([UserRoles.ADMIN, UserRoles.TEACH-
ER]), subjectController.removeSubject)
// table page routes
router.put("/table/create", roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]), table-
Controller.createTable)
router.get("/table/getNames", roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]),
tableController.getTableNames)
router.get("/table/getStudentNames", tableController.getStudentTableNames)
router.get("/table/:id", tableController.getTableById)
router.put("/table/:id/setMark", roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]),
tableController.setMark)
router.put("/table/:id/addColumn", roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]),
tableController.addColumn)
router.post("/table/:id/changeColumn", roleFilter([UserRoles.ADMIN, UserRoles.TEACH-
ER]), tableController.changeColumnName)
router.delete("/table/:id/remove", roleFilter([UserRoles.ADMIN, UserRoles.TEACHER]),
tableController.removeTable)
router.delete("/table/:id/removeColumn", roleFilter([UserRoles.ADMIN, UserRoles.
TEACHER]), tableController.removeColumn) export default router.routes();
```

Для эффективной работы БРС на клиентской стороне веб-приложения необходим фронтенд, обеспечивающий взаимодействие пользователя с серверной частью приложения. Фронтенд состоит из HTML, CSS и JavaScript-кода и отвечает за отображение данных и обработку действий пользователя, а также за отправку запросов на сервер и обработку полученных от него ответов. В нашем случае для взаимодействия с сервером в клиентской части проекта используем библиотеку React и фреймворк Redux Toolkit (RTK). RTK предоставляет удобные инструменты для написания асинхронных запросов к серверу, таких как thunk-функции и срезы (slices) состояния. Комбинация React и RTK позволяет создавать удобный и масштабируемый интерфейс для взаимодействия с сервером, удобный пользовательский интерфейс и обеспечить более эффективную работу с сервером. Полный исходный программный код, разработанной БРС, размещен в открытом доступе на платформе GitHub [5]. Ниже приведен код сервиса, обеспечивающего коммуникацию с сервером для получения и редактирования данных о дисциплинах:

```
import { appApi } from "../store/reducers/appApi";
import { ISubjectAddGroupRequest, ISubjectAddGroupResponse, ISubjectCreateRequest,
ISubjectCreateResponse, ISubjectGetRequest, ISubjectGetResponse, ISubjectGetSearchRe-
quest, ISubjectGetSearchResponse, ISubjectRemoveRequest, ISubjectRemoveResponse,
ISubjectRemoveGroupRequest, ISubjectRemoveGroupResponse, ISubjectSettingsChangeRe-
quest, ISubjectSettingsChangeResponse } from "../types/api";
export const subjectService = appApi.injectEndpoints({
    endpoints: bulider => ({
        getSubject: bulider.query<ISubjectGetResponse, ISubjectGetRequest>({
            query: ({ id }) => `subject/${ id }`,
            providesTags: ['Subject']        }),
       getSubjectSearch: bulider.query<ISubjectGetSearchResponse,ISubjectGetSearchRequest>({
            query: (credentials) => ({url: "subject/search",
               method:"GET", params:credentials}), providesTags: ['Subjects']        }),
       addGroupToSubject: bulider.mutation<ISubjectAddGroupResponse, ISubjectAddGroupRequest>({
            query: ({ id, groupId }) => ({url: `subject/${ id }/addGroupToSubject`,
                method: "POST", body: {id: groupId                }            }),
            invalidatesTags: ['Subjects', 'Subject', 'Groups']        }),
        removeGroupFromSubject: bulider.mutation<ISubjectRemoveGroupResponse, ISub-
jectRemoveGroupRequest>({
```

```
            query: ({ id, groupId }) => ({
                url: `subject/${ id }/removeGroupFromSubject`,
                method: “POST”,
                body: { id: groupId  }          }),
            invalidatesTags: [‘Subject’]        }),
        createSubject: bulider.mutation<ISubjectCreateResponse, ISubjectCreateRe-
quest>({
            query: (credentials) => ({url: “subject/create”,
                method: “PUT”,
                body: credentials
            }),            invalidatesTags: [‘Subjects’]        }),
        changeSubjectSettings: bulider.mutation<ISubjectSettingsChangeResponse, ISub-
jectSettingsChangeRequest>({
            query: ({ initialId, ...credentials }) => ({
                url: `subject/${ initialId }/change`,
                method: “POST”,
                body: credentials            }),
            invalidatesTags: [‘Subject’]         }),
        removeSubject: bulider.mutation<ISubjectRemoveResponse, ISubjectRemoveRe-
quest>({
            query: ({ id }) => ({url: `subject/${ id }/remove`,
                method: «DELETE»}),
            invalidatesTags: [‘Subjects’]})})})
```

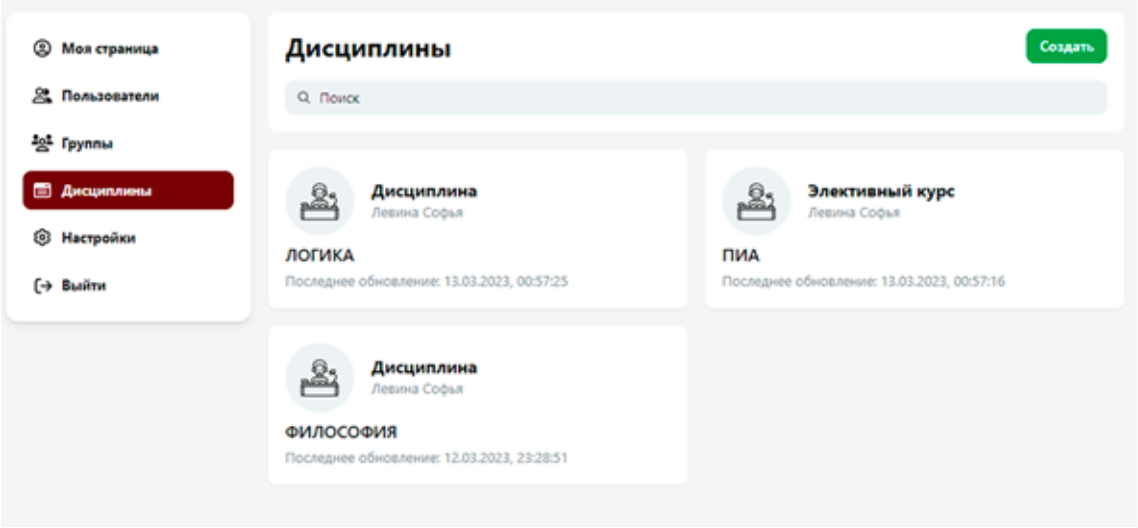

Рис. 2. Интерфейс страницы «Дисциплины»

Взаимодействие с пользователем через удобный и интуитивно понятный интерфейс – один из ключевых элементов создания успешного веб-приложения [6]. Разработка приложения включает в себя не только правильную организацию и представление данных, но также и эстетически приятный и легко понятный пользовательский интерфейс. Фрагмент разработанного интерфейса представлен на рис. 2.

Заключение

Для создания эффективного и масштабируемого веб-приложения БРС необходимо проектировать его с учетом построения правильной структуры базы данных, заканчивая созданием удобного пользовательского интерфейса. Важно учитывать принципы организации кода, использовать подходящие технологии для каждого этапа разработки. Коммуникация с базой данных должна происходить через надежный ORM-фреймворк, например такой, как Prisma, а структура проекта сервера должна быть оптимизирована для обеспечения удобства сопровождения и масштабирования. Правильно организованная роутинговая структура и промежуточные обработчики по-

могут создать разделение прав на запросы и обеспечить безопасность приложения. И, наконец, использование современных технологий, таких как React для клиентской части, позволит создать удобный пользовательский интерфейс и обеспечить более эффективную работу с сервером. Полный исходный программный код, разработанной БРС, размещен в открытом доступе на платформе GitHub.

Список литературы

1. Ананченко И.В. Облачные технологии в высшем образовании // Современные наукоемкие технологии. 2015. № 5. С. 48-52.

2. Юсупов Р.М., Мусаев А.А. Информационное зеркало университета // Известия Санкт-Петербургского государственного технологического института (технического университета). 2019. № 48(74). С. 22-35.

3. Коряковцева О.А. Преимущества и проблемы применения балльно-рейтинговой системы в вузе // Гуманитарные науки (г. Ялта). 2021. № 1(53). С. 62-69.

4. Свидетельство о государственной регистрации программы для ЭВМ № 2021613631 Российская Федерация. Автоматизированная информационная система «Балльно-рейтинговая аттестация» (АИС «БРА»): № 2021612449: заявл. 25.02.2021: опубл. 11.03.2021 / Д.Н. Петров, А.Н. Луцко; заявитель Федеральное государственное бюджетное образовательное учреждение высшего образования «Санкт-Петербургский государственный технологический институт.

5. brs-app // GitHub [Электронный ресурс]. URL: https:/ / github. com/Gryaznykh-Ivan/brs-app (дата обращения: 30.05.2023).

6. Русаков С.В., Русакова О.Л., Смольяков М.Д. Анализ эффективности учебной деятельности студентов в рамках балльно-рейтинговой системы // Новые информационные технологии в образовании и науке. 2022. № 5. С. 74-79. DOI: 10.17853/2587-6910-2022-05-74-79.