Таким образом, можно сделать вывод о том, что сельскохозяйственные организации обладают специфическими чертами, которые делают их более экономически уязвимыми по сравнению с другими предприятиями. Так, для каждого предприятия должна быть разработана индивидуальная система индикаторов производственной безопасности, поскольку они помогают дать точную характеристику состояния безопасности организации. Также была разработана модель производственной безопасности сельскохозяйственной организации, которая позволяет эффективно обеспечивать безопасность предприятия.

Библиографический список

1. Хоружий, Л.И., Катков, Ю.Н., Полетаева, Д.М. Учетно-аналитическое обеспечение производственной безопасности организации АПК. / Л.И. Хоружий, Ю.Н. Катков, Д.М. Полетаева // Бухучет в сельском хозяйстве. – 2018. – №6. – С. 22–34.
2. Хоружий, Л.И., Катков, Ю.Н. Учетно-аналитическое обеспечение экономической безопасности хозяйствующего субъекта / Л.И. Хоружий, Ю.Н. Катков // Вестник профессиональных бухгалтеров. – 2016. – №4. – С. 39–44.

УДК 004.9

СОЗДАНИЕ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ НА ОСНОВЕ RESTFUL API АРХИТЕКТУРЫ

***Гречихин Андрей Геннадиевич**, студент 4 курса института экономики и управления АПК, РГАУ-МСХА им. К. А. Тимирязева*
***Гильматдинов Ильдар Ринатович**, студент 4 курса института экономики и управления АПК, РГАУ-МСХА им. К. А. Тимирязева*
***Ивонин Олег Алексеевич**, студент 4 курса института экономики и управления АПК, РГАУ-МСХА им. К. А. Тимирязева*

***Научный руководитель: Никаноров Михаил Сергеевич**, старший преподаватель кафедры прикладной информатики, института экономики и управления АПК, РГАУ-МСХА им. К. А. Тимирязева*

***Аннотация:** В данной работе было спроектировано и разработано клиент-серверное приложение с архитектурой Restfull и взаимодействием с клиентом по API. Затронуты проблемы, связанные с проектированием микросервисов и его взаимодействия с клиентом. В качестве клиента был выбран React JS и тема электронного журнала для учета успеваемости студентов.*

***Ключевые слова:** Проектирование, Алгоритмы, Микросервисы, React JS, Разработка ПО, Асинхронное программирование, API, Restful, Java, Spring.*

Одна из задач, которая часто встречается в статистике – группировка наблюдений по определенному признаку. Одним из способов решения данной задачи является визуальная группировка, где в качестве осей выступают значения признаков, а сами точки на плоскости наблюдениями, которые необходимо отнести к какой-либо группе.

Цель работы: Изучить принципы работы клиент-серверных приложений с Restful API архитектурой. Создать серверную часть приложения с подобной архитектурой. Создать клиентское веб-приложение с целью проверки работоспособности реализованной серверной части, выявить последующую возможность применения подобной серверной архитектуры приложения.

Серверная архитектура RESTful в современном мире клиент-серверных приложений является наиболее популярной по причине распределения нагрузок между несколькими серверами путем разбиения всего функционала на несколько микросервисов, с которыми может взаимодействовать клиентское приложения. Для реализации подобного серверного приложения был выбран язык программирования JAVA в связке с Фреймворком Spring [1].

Одним из архитектурных решений в проекте является REST. Для взаимодействия между сервером и клиентом используется RESTful API. С помощью HTTP или HTTPS запросов (POST и GET), клиент может получать какие-либо данные, либо же изменять, добавлять и удалять компоненты [2].

К таковым компонентам относятся: группы, дисциплины, оценки, студенты. В случае с изменением данных, сутью RESTful API является получение POST запроса извне, конвертирование модели в сущность базы данных и сохранение как таковой. Преимуществами подобной архитектуры является тот факт, что данный API возможно использовать в будущем при разработке приложений или интегрировании нового функционала в существующие приложения. Мы же, придерживаемся мнения, что созданный API является хорошим задатком для того, чтобы создать в будущем мобильное приложение с подобным функционалом. Схема работы такого сервера с RESTful API архитектурой отображена на рисунке 1.

Для взаимодействия с Restful сервером было решено реализовать клиентское веб-приложение на языке программирования JavaScript с использованием Node.JS и React JS. Данная связка позволяет использовать JavaScript на сервере, повысить масштабируемость и читаемость реализуемого кода приложения. Основным же плюсом данного решения является разрешение использования JS для работы с данными и файлами (в противном случае данные операции блокирует встроенная в браузеры политика CORS (политика, работающая с взаимодействием между разными доменами, разрешая, или запрещая подобны обмен)) [3].

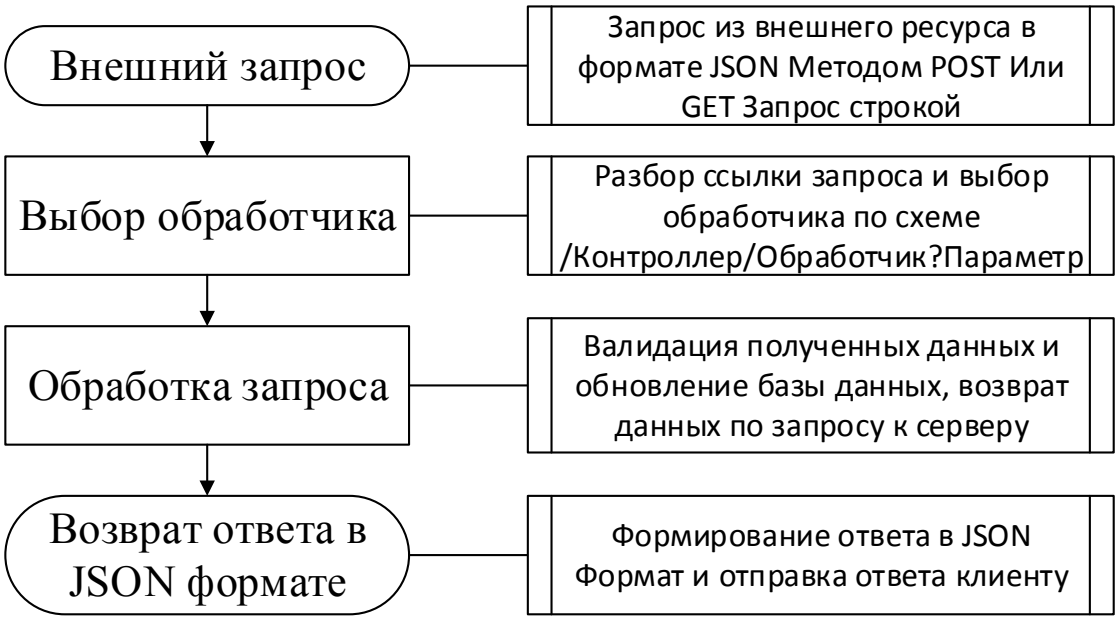

Рис. 2. **Схема работы RESTful API сервера**

Так как React JS является асинхронным, то есть часть программного кода может запуститься и дать «обещание» основному коду, что он вернет результат по окончании выполнения своей части, возникает проблема, что изменения данных состояний необходимо контролировать. Асинхронность наиболее часто используется как раз таки при использовании внешних API, которые взаимодействуют со сторонним сервером. Таким образом, клиентское приложение обрисовывает интерфейс сразу же после получения «обещаний» от кода, который взаимодействует с API, а уже после получения «обещанных» данных обновляет только ту часть приложения, к которым относятся полученные данные. Упрощенная схема работы подобного клиентского приложения изображена на рисунке 2. [4].

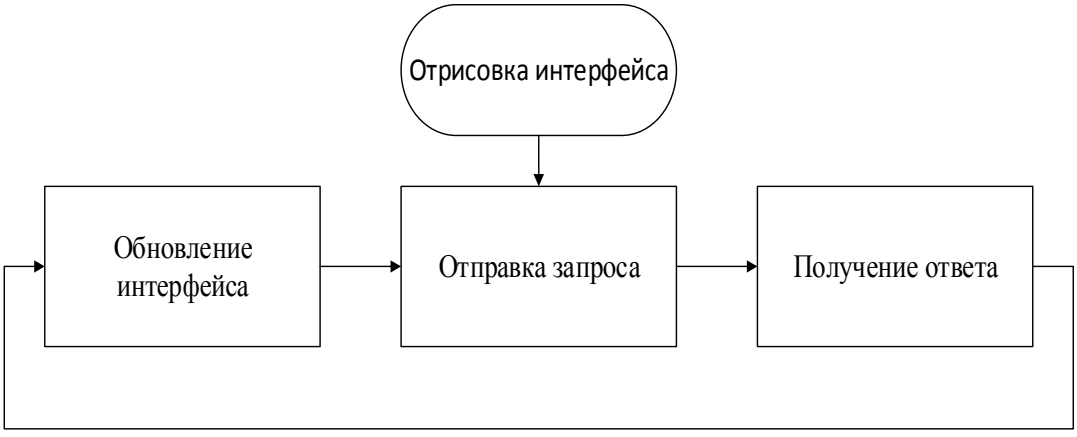

Рис. 3. **Схема работы клиентского веб-приложения**

В результате полученных теоретических данных было реализовано два приложения. Одно с серверной части, а другое с клиентской, которые взаимодействуют между собой через API интерфейсы, которые позволяют реализовывать клиентскую часть на любой из доступных платформ. Они позволяют отправлять и принимать POST и GET запросы.

В процессе реализации были выявлены проблемы, с которыми сталкивается разработчик при реализации асинхронных запросов к серверу.
Полученный результат можно масштабировать как с клиентской части, например, собирать статистические данные и отражать их, так и добавлять новые данные на уже реализованном сервере. Другим же способом расширения функционала клиента является обращение к другим серверам, которые работают с другими данными, таким образом реализовывая микросервисный подход к разработке клиентских веб-приложений.

Библиографический список

1. Лонг Д Java в облаке. Spring Boot, Spring Cloud, Cloud Foundry [учебное пособие] / Лонг Д . – Издательский Дом ПИТЕР, 2019 – 624 с.
2. Арно Лоре Проектирование веб-API [учебное пособие] / Беликов Д. А. – ДМК-Пресс, 2020 г – 440 с.
3. Руководство по React JS [электронный ресурс]: URL: https://ru.reactjs.org/docs/getting-started.html.
4. Руководство по работе с REST API в React JS [электронный ресурс]: URL: https://ru.reactjs.org/docs/faq-ajax.html.

УДК 332.12

ПРОБЛЕМА ЭКОНОМИЧЕСКОЙ ДИФФЕРЕНЦИАЦИИ РЕГИОНОВ РОССИЙСКОЙ ФЕДЕРАЦИИ

Зубова Мария Андреевна, *студентка 1 курса, института экономики и управления АПК, РГАУ-МСХА им. К. А. Тимирязева*

Научный руководитель: Арзамасцева Наталья Вениаминовна, *доцент кафедры политической экономии, института экономики и управления АПК, РГАУ-МСХА им. К. А. Тимирязева*

Аннотация*: Целью статьи является оценка ключевых причин диспропорции экономического развития регионов Российской Федерации. В ходе исследования рассмотрены важнейшие аспекты теории пространственного неравенства и проанализированы основы конкурентных преимуществ.*

Ключевые слова*: диспропорция, регион, экономическое развитие, теория пространственного неравенства, причины дифференциации регионов, конкурентные преимущества.*

Социально-экономическое неравенство существовало всегда и естественно для всех государств. Однако решение вопроса территориальной дифференциации страны – одна из насущных задач внутренней политики любого государства, поскольку данное явление угрожает социальной стабильности и благосостоянию страны в целом. В настоящий момент отсутствует прогресс в преодолении социально-экономической дис-