**Титульный лист**

**задание**

**аннотация**

**СОДЕРЖАНИЕ**

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 8](#_Toc165729947)

[ВВЕДЕНИЕ 9](#_Toc165729948)

[1 Анализ предметной области 11](#_Toc165729949)

[1.1 Анализ требований 11](#_Toc165729950)

[1.2 Обзор наиболее распространенных решений 13](#_Toc165729951)

[1.3 Вывод по разделу 1 16](#_Toc165729952)

[2 Проектирование 17](#_Toc165729953)

[2.1 Построение диаграммы использования 17](#_Toc165729954)

[2.2 Построение диаграммы бизнес-процессов с использованием методологии IDEF0 18](#_Toc165729955)

[2.3 Построение диаграммы деятельности 19](#_Toc165729956)

[2.4 Построение диаграммы последовательности 21](#_Toc165729957)

[2.5 Проектирование диаграммы базы данных 26](#_Toc165729958)

[2.6 Вывод по разделу 2 27](#_Toc165729959)

[3 Разработка серверной части 28](#_Toc165729960)

[3.1 Регистрация и авторизация 28](#_Toc165729961)

[3.2 Разработка функциональных возможностей преподавателя 31](#_Toc165729962)

[3.3 Разработка функциональных возможностей студента 35](#_Toc165729963)

[3.4 Тестирование и документирование серверной части системы 37](#_Toc165729964)

[3.5 Вывод по разделу 3 43](#_Toc165729965)

[4 Разработка клиентской части 44](#_Toc165729966)

[4.1 Разработка дизайна пользовательского интерфейса 44](#_Toc165729967)

[4.2 Разработка клиентской части административной панели 46](#_Toc165729968)

[4.3 Разработка клиентской части панели студента 51](#_Toc165729969)

[4.4 Тестирование клиентской части 56](#_Toc165729970)

[4.5 Вывод по разделу 3 56](#_Toc165729971)

[ЗАКЛЮЧЕНИЕ 57](#_Toc165729972)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59](#_Toc165729973)

# **ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

В данном отчете о практике применяют следующие термины с соответствующими определениями:

**Валидация –** это процесс проверки данных, введенных пользователем, на соответствие заданным критериям

**Клиент** – это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу

**Плагин** – независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей

**Стэк** – это набор инструментов, применяющийся при работе в проектах и включающий языки программирования, фреймворки, системы управления базами данных, компиляторы и так далее

**Фреймворк** – это набор инструментов, библиотек и рекомендаций, предназначенных для разработки программного обеспечения

**Хеширование** – преобразование входного массива данных произвольной длины в выходную битовую строку фиксированной длины

**ACID** – набор требований к транзакционной системе, обеспечивающий наиболее надёжную и предсказуемую её работу — атомарность, согласованность, изоляцию, устойчивость

**API (Application Programming Interface)** – это совокупность инструментов и функций в виде интерфейса для создания новых приложений, благодаря которому одна программа будет взаимодействовать с другой

**GitHub** – это веб-платформа для совместной разработки программного обеспечения, предоставляющая инструменты, которые помогают программистам работать над проектами вместе в режиме реального времени

**JSON Web Token** – это открытый стандарт для создания токенов доступа, основанный на формате JSON

**Id (идентификатор)** – уникальный признак объекта, позволяющий отличать его от других объектов, то есть идентифицировать

# **ВВЕДЕНИЕ**

Управление учебной деятельностью это большая работа, которая заключается не только в проверке решений и выставлении баллов, но и в организационных процессах, подготовке материалов и общем контроле обучения. На всю работу по организации и управлению учебной деятельностью уходит много времени, которое очень ценно в современных реалиях. Неправильная организация учебной деятельности вызывает множество проблем, связанных с коммуникацией, осведомленностью студентов, уровнем обучения и его своевременностью, что также приводит к еще большей нехватке времени из-за попыток решить эти проблемы. Для лучшего управления учебной деятельностью были разработаны системы, которые автоматизируют большинство организационных и учебных процессов, помогают сэкономить время, повысить уровень взаимного понимания, обучения и улучшить коммуникацию между преподавателем и студентами. Данные системы широко распространены как в университетах, так и в школах, однако они могут не подходить под определенные задачи, отчего спрос на разработку собственных систем до сих пор существует. Запрос на разработку такой системы был получен от преподавателя факультета инфокоммуникационных технологий по фронтенд и бэкенд разработке, Добрякова Давида Ильича. Функциональные требования, выдвинутые заказчиком, на данный момент не выполняются доступными в России системами управления обучением.

В настоящее время рассматривается вариант разработки общей системы управления учебной деятельностью с разделением доступа к функционалу в зависимости от роли: преподаватель и студент. Данная система позволит преподавателю вести отчетность по успеваемости студентов, подготавливать материалы, а также предоставлять доступ к данным материалам, в то время как студент сможет получать задания, следить за дедлайнами, записываться на варианты и решать контрольные.

Для реализации данной системы будет написан собственный бэкенд, реализующий необходимый функционал, который будет разграничен проверкой роли пользователя. Бэкенд планируется реализовывать на таких технологиях как: Express, фреймворк Node.js, и нереляционная база данных MongoDB. Фронтенд часть веб-приложения будет разработана на React.js с использованием необходимых хуков и библиотеки для работы с файлами. Данный стэк технологий называется “MERN” и расшифровывается как MongoDB, Express, React.js, Node.js [3]. Этот набор инструментов применим для fullstack-разработки, каждая часть данного набора подобрана для эффективной работы с JavaScript. Можно сказать, что MERN - основа для амбициозных идей, которые легко масштабируются благодаря веб-серверу на Node.js и экономно разрабатываются благодаря популярности и доступности языка JavaScript [1].

**Цель**: разработать систему управления учебной деятельностью для автоматизации рабочих процессов преподавателя.

**Задачи**:

1. обзор существующих решений,
2. проектирование,
3. разработка серверной части,
4. разработка клиентской части.

**Планируемый результат в соответствии с Индивидуальным заданием**: в ходе работы должна быть спроектирована система управления учебной деятельностью и разработаны ее клиентская и серверная части, отвечающие современным требованиям и требованиям преподавателя практики факультета инфокоммуникационных технологий, Добрякова Давид Ильича, именуемого в дальнейшем “Заказчик”. Данная система позволит автоматизировать рабочие процессы преподавателя, а также консолидировать и структурировать основной процесс обучения.

# **Анализ предметной области**

## **Анализ требований**

Перед началом работы Заказчиком были определены функциональные и нефункциональные требования к разрабатываемой системе. Как упоминалось ранее, весь функционал приложения разделен между двумя группами пользователей: администратор и студент. Функционал администратора (преподавателя) представляет из себя следующие возможности:

* авторизация по почте и паролю,
* создание, просмотр, удаление контрольных, лабораторных работ и дополнительных заданий,
* создание, просмотр, удаление вариантов контрольных и лабораторных работ,
* создание, удаление вопросов разных сложности и типа по темам контрольных работ с указанием правильного ответа,
* создание вариантов контрольных работ путем их генерации из заранее определенного количества вопросов той или иной сложности,
* указание штрафных баллов за просрочку дедлайна на 1 день по лабораторным работам,
* контроль доступа к системе зарегистрированным пользователям,
* просмотр статистики записей на лабораторные работы и дополнительные задания,
* проверка текстовых ответов студентов в контрольных работах,
* предоставление дополнительных материалов при создании дополнительных заданий и вариантов лабораторных работ,
* ограничение количества мест для записи на дополнительные задания и варианты лабораторных работ,
* непрогнозируемое распределение вариантов контрольных работ среди студентов,
* открытие доступа к лабораторным, контрольным работам и дополнительным заданиям для выбранных групп,
* поиск по фамилии и фильтрация пользователей по учебной группе.

Функционал студента для данной версии системы сокращен и предоставляет следующие возможности:

* регистрация с обязательными почтой, паролем, именем, фамилией, учебной группой, ссылками на профили в GitHub и Telegram,
* авторизация по почте и паролю,
* использование функционала систему осуществляется только после того, как администратор одобрит студенту доступ к нему в системе,
* решение контрольных работ с автоматической проверкой вопросов с вариантами ответа,
* просмотр предварительных баллов за решение контрольной работы,
* просмотр окончательных баллов за решение контрольной работы после проверки преподавателем,
* просмотр, запись, отмена записи на дополнительные задания и варианты лабораторных работ с ограниченным количеством мест,
* просмотр таблицы дедлайнов по лабораторным работам и дополнительным заданиям с прогнозируемым максимальным количеством баллов на сегодняшний день с учетом дней просрочки предоставления работы,
* просмотр личной информации, баллов за контрольные работы, работа с записями на лабораторные работы и дополнительные задания в личном кабинете,
* фильтрация вариантов лабораторных работ по предмету и названию,
* фильтрация дополнительных заданий и контрольных работ по предмету.

Также Заказчиком были предоставлены нефункциональные требования к системе:

* система должна обеспечивать защиту личных данных пользователей,
* в системе должно применяться шифрование данных для предотвращения несанкционированного доступа к информации,
* система должна корректно обрабатывать данные пользователя и предоставлять точную и корректную информацию обо всех работах и заданиях,
* система должна быть бесплатной,
* доступ к функционалу системы должен быть разграничен между разными группами пользователей не только на уровне сервера, но и клиента.

## **Обзор наиболее распространенных решений**

На данный момент не существует системы, которая подходила бы по всем указанным Заказчиком требованиям. Однако существует множество высокотехнологичных, хорошо продуманных, широко используемых систем, которые могли бы подойти, будь требования немного другими. Так, например, система управления обучением “Контур Класс” является наиболее подходящей, ведь имеет возможность добавления студентов в систему по ссылкам-приглашениям (Рисунок 1), предоставляет возможности по созданию контрольных (Рисунок 2), заданий, отслеживанию успеваемости (Рисунок 3). Однако данная система является платной, самостоятельная запись на задания отсутствует и не ограничена количеством мест, отсутствует возможность генерации вариантов тестов, контрольных, предоставление доступа к заданиям производится по ФИО студента, что является неудобным решением и занимает большое количество времени, ведь заказчику необходимо предоставление доступа по учебным группам. К тому же в данной системе можно предоставить доступ только ко всему курсу целиком, что не соответствует логике учебного процесса (Рисунок 4). К примеру, группы обычно получают лабораторные задания постепенно, в зависимости от практических занятий и лекций, также работа организована и с контрольными, поэтому нельзя предоставить доступ к курсу всем группами одновременно, ведь тогда у какой-либо группы может быть преимущество в виде большего количество времени на работу с лабораторными или раннее открытие доступа к контрольной [7].

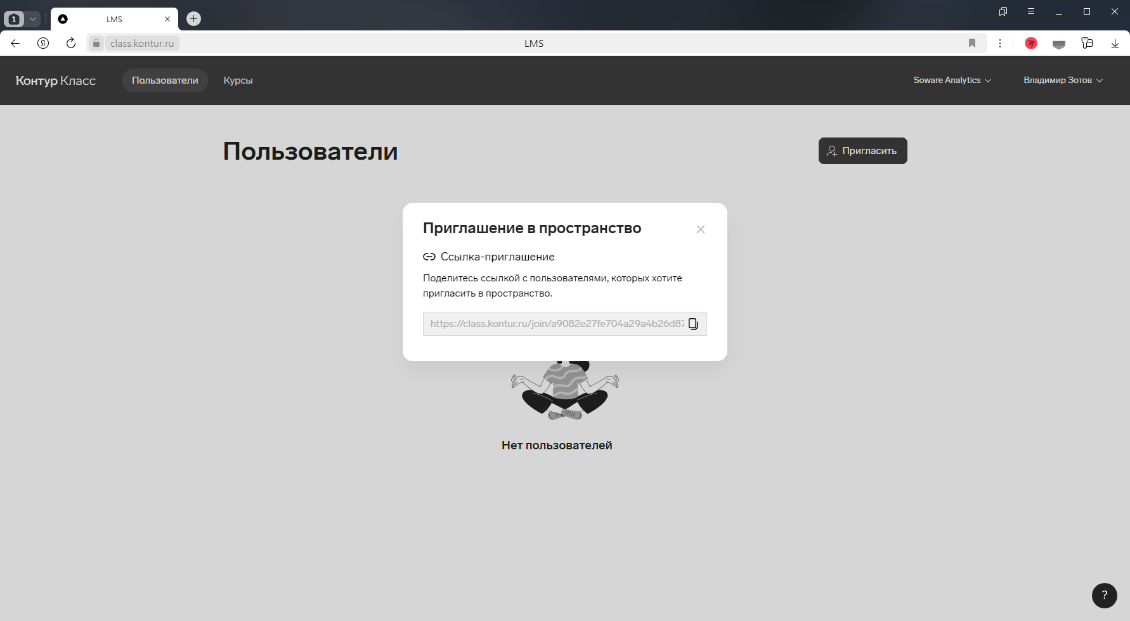


Рисунок 1 – Приглашение студентов по ссылке на платформе “Контур Класс”

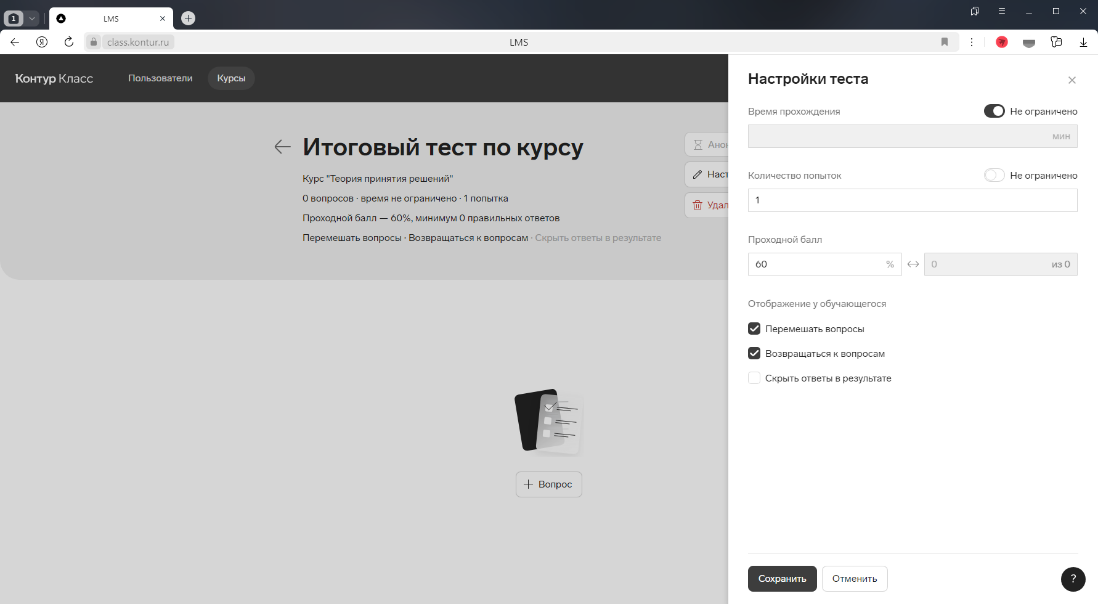


Рисунок 2 – Создание тестов на платформе “Контур Класс”

Изображение выглядит как текст, снимок экрана, Человеческое лицо, веб-страница

Автоматически созданное описание

Рисунок 3 – Отслеживание качества обучения на платформе “Контур Класс”

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4 – Предоставление доступа к курсам на платформе “Контур Класс”

Следующим аналогичным решением и одним из самых популярных на рынке среди компаний является Teachbase, стоимость использования которого равна 565 500 рублей в год для управления 300 пользователей и доступом ко всему необходимому функционалу. Данная система намного больше предыдущей, ведь предусматривает использование не только учебными заведениями, но и бизнесом, поэтому имеет функционал по развитию карьеры, мотивации, интеграции дополнительных курсов и блоков кода по сторонним API. Teachbase предоставляет практически весь необходимый для Заказчика функционал по контролю активности, отчетности, работе с контрольными. Однако, как и предыдущий рассмотренный аналог, данная система не предоставляет возможности самостоятельной записи на задания и не отслеживает ограничение по количеству мест. Отсутствует генерация вариантов тестов, контрольных, поэтому преподавателю придется составлять их самостоятельно. Проверка открытых вопросов с коротким ответом не подразумевает проверку преподавателем, а проверяется автоматически по совпадению с введенными преподавателем вариантами ответа, также не является оптимальным решением, поскольку на верность ответа в данном случае будут влиять порядок и форма слов [8].

Прочие системы управления обучением не подходят Заказчику по тем же причинам, которые уже были рассмотрены, а именно: несоответствие требованиям по записи на варианты, составлению и проверке контрольных, предоставлению доступа к контрольным и лабораторным. Данные системы больше нацелены на обучение, в то время как первоначальная цель Заказчика – управление учебной деятельностью. Поэтому целевым пользователям разрабатываемой системы не нужен функционал по изучению материалов, лекций, в то время как выбор вариантов лабораторных и самостоятельная запись на дополнительные задания по желанию являются приоритетными.

Для более детального анализа аналогов и понимания необходимости разработки системы необходимо проанализировать методы, которые используются Заказчиком на данный момент, и трудности, с которыми он сталкивается в процессе. В основном заказчик работает в электронных таблицах Google Sheets, формах Google и Github. С помощью таблиц Заказчик контролирует успеваемость студентов и дедлайны, с помощью форм - создает контрольные, а с помощью Github - выдает и проверяет задания. Однако эти системы не автоматизированы и потому не ускоряют работу, а лишь переводят ее в электронный формат, предоставляя постоянный онлайн доступ к результатам данной работы. Также негативно сказывается отсутствие консолидации всех трех систем, так как необходимая информация находится на разных платформах, не синхронизируется между собой, отчего часто приходится выполнять одну и ту же работу, самостоятельно актуализировать информацию и перепроверять полученную.

## **Вывод по разделу 1**

Таким образом, были сформулированы функциональные и нефункциональные требования к системе. При обзоре аналогов не было выявлено прямых аналогичных решений, но было найдено много косвенных. Однако данные системы не удовлетворяют всем потребностям и требованиям Заказчика, из чего можно сделать вывод о необходимости разработки персонализированной системы управления учебной деятельностью.

# **Проектирование**

## **Построение диаграммы использования**

Для более глубокого понимания функционала системы и его распределения между разными ролями пользователей необходимо построить диаграмму прецедентов (диаграмму использования). Данная диаграмма в UML отражает отношения между акторами и прецедентами и является составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. В основном это диаграмма, описывающая доступность функционала разрабатываемой программной системы в зависимости от принадлежности пользователя к группе.

Разрабатываемая система разделена на две группы пользователей: студент и администратор (преподаватель). Работа преподавателя в основном заключается в контроле и создании необходимых для обучения материалов и тестов, в то время как основная цель пользователя – следовать предоставленному процессу обучения, выполнять задания. Подробнее с функциональными возможностями каждой группы авторизованных и неавторизованных пользователей можно ознакомиться на диаграмме использования (Рисунок 5–7). На рисунке 5 можно заметить, что у администратора нет функционала для регистрации, поскольку, в целях предотвращения атак и обеспечения безопасности данных, созданием аккаунтов администраторов занимается разработчик системы.

Изображение выглядит как текст, снимок экрана, круг, Шрифт

Автоматически созданное описание

Рисунок 5 – Диаграмма использования неавторизованных групп пользователей

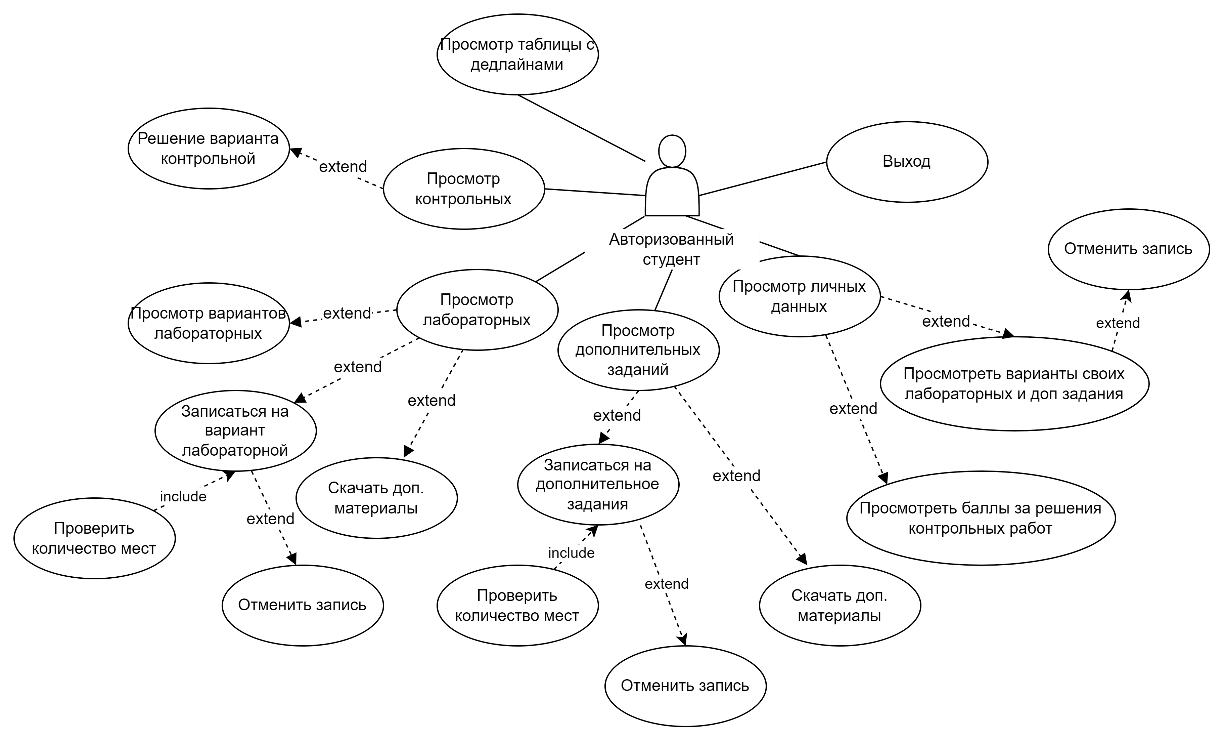


Рисунок 6 – Диаграмма использования авторизованного студента

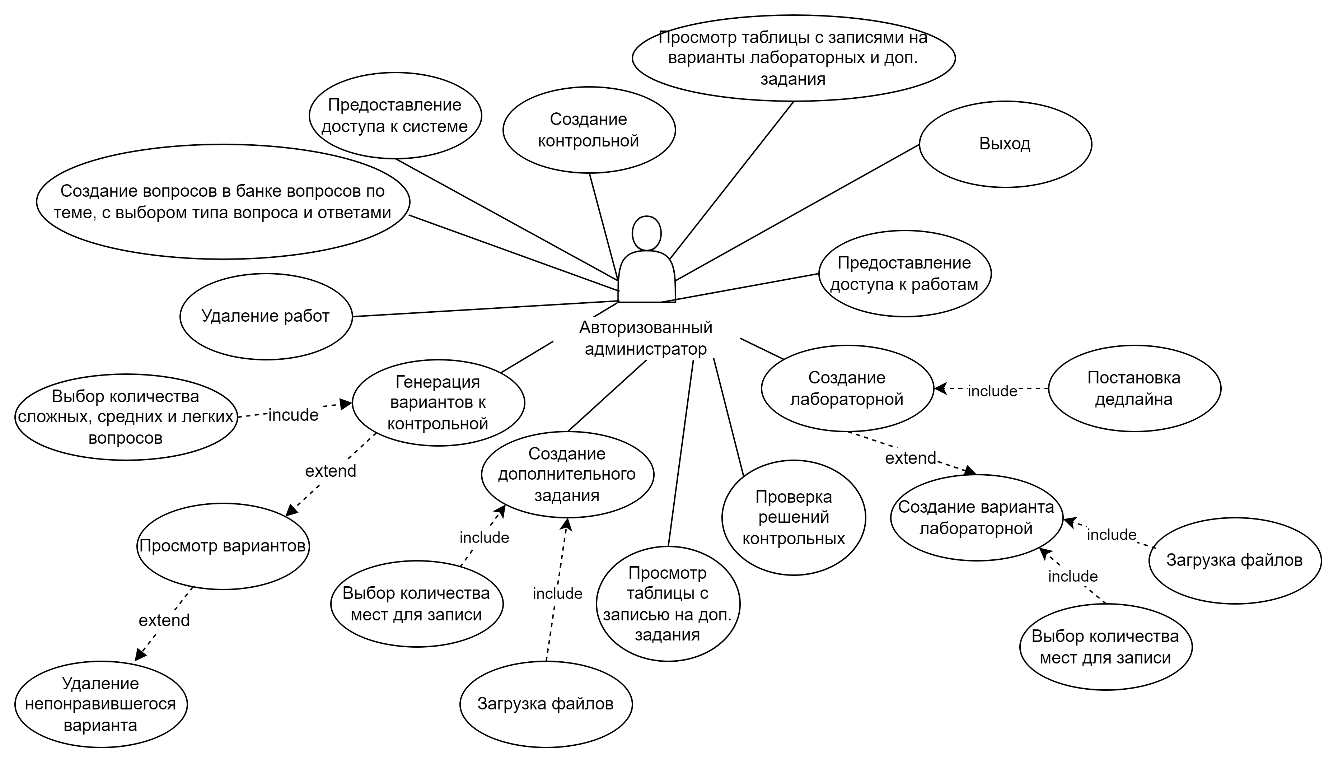


Рисунок 7 – Диаграмма использования авторизованного администратора

## **Построение диаграммы бизнес-процессов с использованием методологии IDEF0**

IDEF0 - метод функционального моделирования, а также графическая нотация, которая используется для описания и формализации бизнес-процессов [2]. Данный метод построения диаграмм бизнес-процессов позволяет не только понять функциональные возможности системы и их разделение по группам пользователей, но также обратить внимание на данные, которые необходимы для реализации той или иной функции. На данной диаграмме показан весь путь пользователя от регистрации и авторизации до просмотра отчетных таблиц администратором и записи на задание студентом. Результат построения данной диаграммы представлен на рисунке 8.

Изображение выглядит как текст, снимок экрана, диаграмма

Автоматически созданное описание

Рисунок 8 – Диаграмма бизнес-процессов с использованием методологии IDEF0

## **Построение диаграммы деятельности**

Для того, чтобы определить взаимосвязи разных групп пользователей и как функционал одной группы влияет на функциональные возможности другой, необходимо построить диаграмму деятельности. Диаграммы деятельности (Activity diagram) – это один из видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Это блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой [5]. Для построения данной диаграммы были выбраны основные бизнес-процессы системы: работа групп пользователей с контрольными, лабораторными и дополнительными заданиями, поскольку данные сущности и работа с ними являются основой всех бизнес-процессов системы. Результат построения диаграмм деятельности представлен на рисунках 9 – 11.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 9 – Диаграмма деятельности работы с контрольными

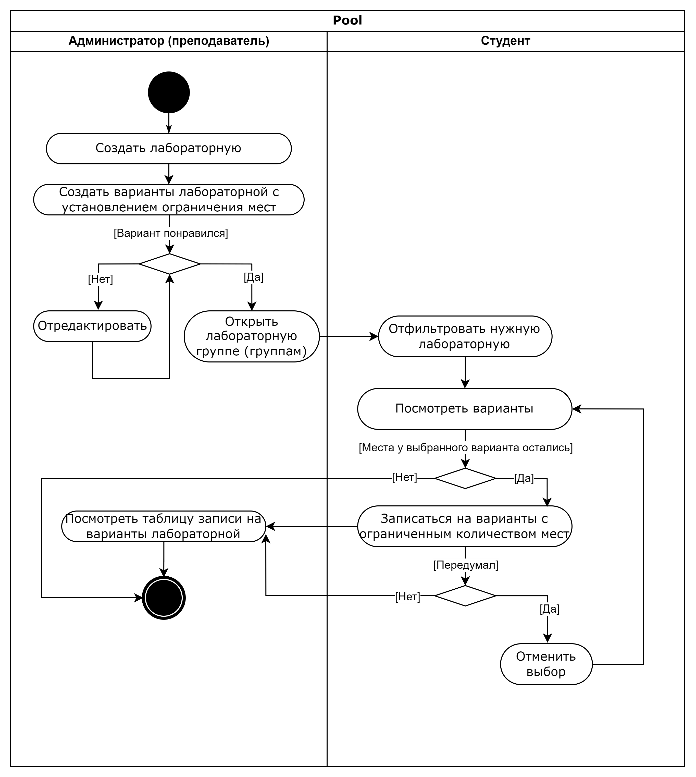


Рисунок 10 – Диаграмма деятельности работы с лабораторными

Изображение выглядит как текст, снимок экрана, программное обеспечение, дизайн

Автоматически созданное описание

Рисунок 11 – Диаграмма деятельности работы с дополнительными заданиями

## **Построение диаграммы последовательности**

Так как система состоит из клиентской и серверной частей, необходимо понять, как будут разделены функциональные требования между ними и как правильно организовать работу системы. Для этого лучше всего подойдет диаграмма последовательности. Диаграмма последовательности (sequence diagram) предназначена для представления взаимодействия между элементами модели программной системы в терминологии линий жизни и сообщений между ними [6].

Как уже упоминалось, основным функционалом данного веб-приложения является работа с контрольными, лабораторными и дополнительными заданиями. Была построена общая диаграмма по созданию контрольных, лабораторных и дополнительных заданий, так как эти функции отличаются между собой только вводимыми данными при создании (Рисунок 12). На рисунке можно увидеть, что клиентская часть не обращается на серверную с запросом на получение массива объектов, а контролирует данный процесс на бэкенде, отправляя данный запрос единожды при авторизации в веб-приложении. Это сделано для того, чтобы не перегружать работу сервера, а также сделать использование веб-приложения быстрее.

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

Рисунок 12 – Диаграмма последовательности по созданию контрольных, лабораторных, дополнительных заданий

Одним из основных требований Заказчика по работе с контрольными является генерация вариантов контрольных с помощью выбора заданного количества уже составленных вопросов различной сложности случайным образом. Чтобы наглядно показать, как будет устроена работа с данным требованием, была построена диаграмма последовательности (Рисунок 13). На данной диаграмме видно, что преподаватель отправляет данные по желаемому количеству вопросов той или иной сложности. Все вопросы поделены на три уровня: легкие, средние, сложные. Затем эти предпочтения отправляются в качестве запроса на сервер, где происходит случайный выбор вопросов из списка имеющихся с учетом их сложности и количества. В последствие составленный вариант отправляется клиенту и добавляется в массив. Так же, как и на предыдущей диаграмме, можно заметить, что клиент не отправляет дополнительный запрос на сервер для получения списка всех вариантов.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 13 – Диаграмма последовательности для генерации варианта контрольной

Работа с контрольными также включает в себя еще два важных требования: решение контрольной и ее проверка. Для наглядного представления данной функциональности были построены две диаграммы последовательности (Рисунок 14 и 15). Стоит упомянуть, что для понижения уровня списывания студентам достается случайный вариант контрольной, что отображено на диаграмме. Также на диаграммах можно заметить, что система обладает двумя уровнями проверок: автоматическая и проверка преподавателем. Первая занимается проверкой закрытых вопросов, с помощью совпадения выбранного варианта ответа с правильным, указанным преподавателем при создании. Вторая же, в свою очередь, заключается в проверке открытых вопросов. Как упоминалось при обзоре аналогов, это важный аспект проверки любых тестов, ведь прописать все возможные правильные ответы на открытые вопросы занимает много времени и сил, гораздо больше, чем единовременная проверка работы, поскольку сложно предусмотреть все возможные ответы студентов.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 14 – Диаграмма последовательности по решению контрольной

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание

Рисунок 15 – Диаграмма последовательности по проверке контрольных преподавателем

Несмотря на то, что работа с вариантами лабораторных также является важной частью общей функциональности, было принято решение не строить диаграмму последовательности по причине того, что создание варианта лабораторной по своей структуре очень похоже на создание лабораторной и отличается только вводимыми преподавателем данными. Однако важно показать диаграмму последовательности по авторизации пользователей в системе, чтобы продемонстрировать, каким образом проверяется роль пользователя и на что это влияет (Рисунок 16). Как видно на диаграмме, роль пользователя записана в объекте пользователя, данная роль хешируется вместе с токеном и отправляется на клиентскую часть, где также проверяется и, в зависимости от роли, перенаправляет пользователя на определенные роуты в системе. Теперь при любых запросах на сервер клиент отправляет не только свой захешированный токен, но и роль, которая также проверяется для доступа к обработке запроса.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 16 – Диаграмма последовательности для авторизации

## **Проектирование диаграммы базы данных**

Перед началом этапа разработки системы необходимо спроектировать базу данных. Это поможет понять, из каких частей, моделей должна состоять система, какие данные необходимы и как они будут использоваться. Как говорилось раннее, была выбрана документо-ориентированная система управления базами данных MongoDB. Данная система управления считается нереляционной, то есть в ней не используется табличная схема строк и столбцов. В таких базах данных применяется модель хранения, оптимизированная под конкретные требования типа хранимых данных [4]. Данный выбор был сделан по нескольким причинам:

1. поддержка JSON-документов, что является большим плюсом при использовании языка программирования JavaScript,
2. высокая скорость работы СУБД за счет возможности индексации,
3. простота администрирования.

Как и у любой СУБД, в MongoDB есть свои минусы. Одним из основных является не полное соответствие ACID. Несмотря на это, количество преимуществ в контексте рассматриваемого проекта превалирует [9].

Для проектирования базы данных было решено создать следующие коллекции:

* Users для хранения информации о пользователях,
* Controls для хранения информации о контрольных работах,
* ControlMarks для хранения информации о решениях контрольных работ,
* Questions для хранения вопросов для генерации контрольных работ,
* Tasks для хранения информации о лабораторных и дополнительных заданиях.

Более подробно с наполнением каждой коллекции можно ознакомиться на рисунке 17.

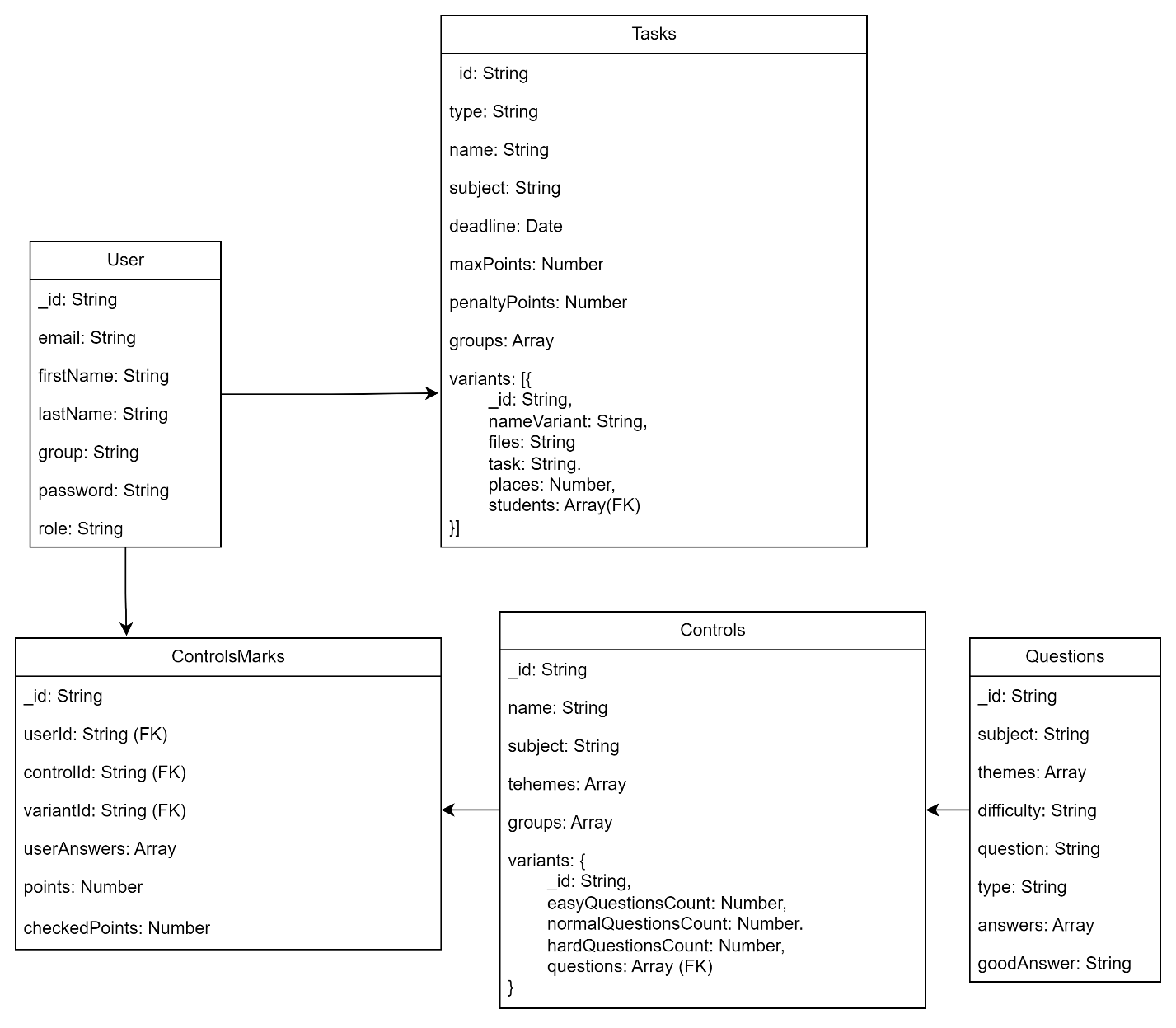


Рисунок 17 – Диаграмма базы данных

## **Вывод по разделу 2**

В результате работы над вторым разделом были построены диаграммы, которые демонстрируют все имеющиеся бизнес-процессы и их соответствие функциональным требованиям. Данные диаграммы позволяют понять весь функционал разрабатываемого веб-приложения, реализацию данного функционала внутри системы, его разделение на группы пользователей, взаимосвязь. Помимо этого, были построены диаграммы, определяющие составные части системы и необходимые для разработки данные.

# **Разработка серверной части**

## **Регистрация и авторизация**

В первую очередь необходимо реализовать функционал регистрации и авторизации, поскольку функциональность веб-приложения зависит от роли пользователя в системе, которую контролируют данные функции. Функционал регистрации на клиентской части доступен только для группы студентов в целях безопасности системы. Как видно по диаграмме базы данных, у пользователя должна быть роль, в зависимости от которой будет разделен функционал. При регистрации студенту автоматически проставляется роль “user”, которую невозможно изменить благодаря наличию валидации при получении запроса от клиента, написанной с использованием библиотеки celebrate (Рисунок 18). Регистрация администратора, в свою очередь, осуществляется схожим образом, но роль пользователя проставляется со значением “admin”.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 18 – Валидация регистрации

При регистрации пароль пользователя хешируется. Это сделано для того, чтобы даже у администратора системы не было доступа к данной информации. Хеширование осуществляется с помощью библиотеки bcrypt. В данном случае число 10 обозначает стоимость для алгоритма. Стоимость определяет, сколько раз будет выполнено данное хеширование (Рисунок 19).

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 19 – Функция регистрации с хешированием пароля

Для реализации функции авторизации необходимо реализовать поиск пользователя в системе по почте и захешированному паролю. Для этого был написан статический метод модели (Рисунок 20).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 20 – Статический метод поиска пользователя по почте

Функция авторизации состоит также из отправки JSON Web Token. Данный токен необходим на клиенте для проверки авторизации пользователя, а также для получения id пользователя и его роли. С помощью секретного ключа генерируется подпись JWT на основе содержимого токена, что обеспечивает целостность данных (Рисунок 21).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 21 – Функция авторизации пользователя

Для безопасности системы были написаны две проверки: наличие авторизации и роли администратора соответственно. Первая осуществляется перед всеми функциями, кроме регистрации и авторизации (Рисунок 22). В то время как вторая - только перед функционалом администратора (Рисунок 23).

Изображение выглядит как текст, электроника, компьютер, снимок экрана

Автоматически созданное описание

Рисунок 22 – Проверка наличия авторизации

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рисунок 23 – Проверка роли пользователя

## **Разработка функциональных возможностей преподавателя**

В первую очередь необходимо разобрать наиболее важные бизнеc-процессы, связанные с контрольными работами и данными, влияющими на них, поскольку этих процессов большинство. Функция создания контрольных работ необходима для всей последующей работы с контрольными. Реализация данного функционала представлена на рисунке 24.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рисунок 24 – Функция создания контрольной работ

Важным функциональным требованием является генерация варианта контрольной работы с использованием вопросов из модели “Questions”. При генерации варианта важно получить от пользователя данные о количестве сложных, средних и легких вопросов. Функция находит вопросы, соответствующие заданной тематике контрольной работы, а затем отбирает определенное количество вопросов разного уровня сложности и собирает их в один вариант. Реализация этой функции представлена на рисунках 25 и 26.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 25 – Функция выбора вопросов для генерации варианта

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 26 – Функция создания варианта контрольной работы

Следующим по важности функциональным требованием является возможность предоставления группам доступа к контрольным работам и его удаление. Для реализации данного требования были написаны функции: openControl и deleteAccess. OpenControl принимает от клиентской части массив групп и добавляет их в поле “groups” модели контрольных, в то время как функция deleteAccess принимает одну группу и удаляет ее из соответствующего поля модели. Реализация функции openControl представлена на рисунке 27.

Изображение выглядит как текст, снимок экрана, дисплей, компьютер

Автоматически созданное описание

Рисунок 27 – Функция предоставления группам доступа к контрольной работе

Следующим требованием является проверка решенных контрольных. Данная функция получает id записи о решении пользователем контрольной и количество баллов, а затем перезаписывает данное значение (Рисунок 28).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 28 – Функция проверки решения контрольной

Функциональные требования по работе с лабораторными и дополнительными заданиями схожи с уже представленными функциями для работы с контрольными работами. Они также дают возможность создавать, удалять и предоставлять доступ. Единственные отличия в отсутствии возможности генерации вариантов. Для создания варианта лабораторной работы используется модель “Tasks”. Функция принимает данные о количестве мест для записи на вариант, названии, самом задании, файле, если его добавили (Рисунок 29).

Изображение выглядит как текст, снимок экрана, дисплей, компьютер

Автоматически созданное описание

Рисунок 29 – Функция создания варианта лабораторной работы

Для обработки полученного файла, его проверки на валидность и загрузки на сервер используются библиотеки multer и moment. Первая отвечает за загрузку файла в хранилище, а вторая – за именование файла в системе. Реализация данной функции представлена на рисунке 30.

Изображение выглядит как текст, снимок экрана, дисплей, компьютер

Автоматически созданное описание

Рисунок 30 – Функция загрузки файлов на сервер

## **Разработка функциональных возможностей студента**

Основными процессами студента при работе с разрабатываемым веб-приложением являются: получение контрольных и лабораторных работ, дополнительных заданий, решение контрольной работы, а также запись на варианты лабораторных работ и дополнительные задания.

Функции по получению работ и дополнительных заданий имеют схожую структуру. Для этого функция принимает номер группы студента и отправляет массив доступных для этой группы работ и заданий. В качестве примера реализации выбрана функция getUserControls, предоставляющая контрольные работы (Рисунок 31).

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рисунок 31 – Функция получения доступных контрольных работ

Важно отметить, что при решении контрольной работы студент получает случайный вариант из массива всех вариантов. Это требование разработано для того, чтобы повысить уровень уникальности работ. Данная функция принимает информацию о пользователе, варианте контрольной работы, ответах пользователя и предварительных баллах (Рисунок 32). Предварительные баллы получаются посредством автоматической проверки соответствия выбранных пользователем ответов на закрытые вопросы правильным.

Изображение выглядит как текст, снимок экрана, мультимедиа, дисплей

Автоматически созданное описание

Рисунок 32 – Функция решения контрольной работы

В отличии от контрольных работ, варианты лабораторных работ и дополнительные задания пользователь выбирает сам и записывается на них. Функция makeAppointmnetTask берет id студента и id выбранного задания, затем записывает id пользователя в массив студентов, проверяя при этом, что места для записи еще есть, и что пользователь еще не записан на это задание (Рисунок 33).

Изображение выглядит как текст, снимок экрана, компьютер, дисплей

Автоматически созданное описание

Рисунок 33 – Функция записи на задание

## **Тестирование и документирование серверной части системы**

Для тестирования и документирования серверной части системы было выбрано приложение “Postman”. Данное приложение поддерживает все популярные типы архитектуры, позволяет работать с данными разного формата, имеет облачное хранилище, наглядно показывает результаты тестирования, запоминает токены доступа, что ускоряет работу над тестированием. Документирование с помощью Postman очень удобно по ряду причин: возможность комментирования текста, параметров, заголовков, тела запроса и его результат, автоматическое составление документации из заранее созданных коллекции и запросов с сохраненными результатами, возможность объединять запросы, делая структуру документа удобнее и понятнее, предоставление доступа к документации по ссылке с возможностью скрытия необходимой информации, а также удобный и понятный интерфейс [10].

В результате разработки серверной части для группы пользователей “администратор” были созданы следующие запросы метода GET:

* /controls – получение списка контрольных работ,
* /tasks – получение списка дополнительных заданий,
* /tasks/:filename – скачивание файла дополнительного задания,
* /labs – получение списка лабораторных работ,
* /labs/variant/:filename – скачивание файла лабораторной работы,
* /questions – получение списка вопросов,
* /controlmarks – получение списка решений контрольных работ
* /access – получение списка пользователей.

Список запросов метода DELETE для администратора:

* + /controls/:controlId – удаление доступа к контрольной работе для определенной группы из тела запроса,
  + /controls/delete/:controlId – удаление контрольной работы,
  + /controls/:controlId/:controlVariant – удаление варианта контрольной работы,
  + /tasks/:taskId – удаление доступа к дополнительному заданию для определенной группы из тела запроса,
  + /tasks/delete/:taskId – удаление дополнительного задания,
  + /labs/delete/:labId – удаление лабораторной работы,
  + /labs/:labId – удаление доступа к лабораторной работе для определенной группы из тела запроса,
  + /labs/:labId/:labVariant – удаление варианта лабораторной работы,
  + /questions/:questionId – удаление вопроса,
  + /access/:userId – удаление доступа к системе.

Список запросов метода PATCH для администратора:

* + - /controls/:controlId – предоставление доступа к контрольной работе для определенных групп из тела запроса,
    - /labs/:labId – предоставление доступа к лабораторной работе для определенных групп из тела запроса,
    - /questions/:questionId – обновление вопроса с информацией о сложности вопроса, его типе, вариантах ответа или правильном ответе из тела запроса,
    - /tasks/:taskId – предоставление доступа к дополнительному заданию для определенных групп из тела запроса.

Список запросов метода POST для администратора:

* + - * /controls – создание контрольной работы,
      * /controls/variant/:controlId – генерация варианта контрольной работы с помощью передачи количества простых, средних и сложных вопросов из тела запроса,
      * /labs – создание лабораторной работы,
      * /labs/variant/:labId – создание варианта лабораторной работы с возможным добавлением дополнительных материалов,
      * /tasks – создание дополнительного задания с возможным добавлением дополнительных материалов,
      * /questions – создание вопроса.

Для демонстрации результатов тестирования разберем основной функционал административной панели. Разберем процессы создания, удаления, предоставление доступа к контрольным, лабораторным работам и дополнительным заданиям на примере с лабораторными работами. Для создания лабораторной работы необходимы следующие данные: название, предмет, дедлайн, количество баллов, штрафные баллы. При успешном запросе должен быть отправлен созданные объект лабораторной работы (Рисунок 34). Также система должна отправлять информацию об ошибках запроса с указанием сообщения и кода. Так при отсутствии обязательных данных система отправил ошибку валидации (Рисунок 35). Если в системе уже есть лабораторная работа с таким названием, то должна быть отправлена ошибка 409 с сообщением о конфликте.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 34 – Результат тестирования запроса по созданию лабораторной работы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 35 – Ошибка запроса по созданию лабораторной работы при отсутствии обязательных данных

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 36 – Ошибка создания лабораторной работы с существующим названием

Для создания варианта лабораторной работы также были использованы классы ошибок и сообщения для них. Само создание варианта представлено на рисунке 37. Можно заметить, что в данном запроса используется не JSON формат тела запроса, а form-data, что связано с наличием возможности загрузки файлов.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 37 – Результат тестирования запроса по созданию варианта лабораторной работы

Для предоставления доступа к лабораторной работе передается идентификатор лабораторной работы через параметры и массив групп через тело запроса формата JSON (Рисунок 38). Если в массив передается группа, у которой уже есть доступ, то отправляется сообщение об ошибке (Рисунок 39).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 38 – Результат тестирования запроса по предоставлению доступа к лабораторной работе

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 39 – Ошибка предоставления доступа к лабораторной работе группе, у которой уже есть доступ

В результате разработки серверной части системы для панели студента были созданы следующие запросы метода GET:

* + - * + /users/variant/:controlId – получение случайного варианта контрольной работы,
        + /users – получение информации о себе,
        + /users/controls – получение доступных контрольных работ,
        + /users/labs – получение доступных лабораторных работ,
        + /users/tasks – получение доступных дополнительных заданий,
        + /users/variant/:filename – скачивание прикрепленных файлов,
        + /users/controlmarks – получение результатов решения контрольных работ.

Список запросов метода DELETE:

/users/labs/:labId – отмена записи на вариант лабораторной работы,

/users/tasks/:taskId – отмена записи на вариант дополнительного задания.

Список запросов метода PATCH:

/users/labs/:labId – запись на вариант лабораторной работы,

/users/tasks /:taskId – запись на вариант дополнительного задания.

Список запросов метода POST:

/users/controlmarks/:controlId – решение контрольной работы.

Функционал записи на работы похож между собой, поэтому разберем данную функцию на примере записи на дополнительные задания. Запрос должен выдавать ошибку, если пользователь уже записан на выбранное дополнительное задание (Рисунок 40). При записи на лабораторные работы проверяется, что пользователь еще не записан ни на один вариант выбранной лабораторной работы. Процесс записи происходит с помощью Bearer Token без использования тела запроса.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 40 – Ошибка записи на дополнительное задание с уже имеющейся записью

## **Вывод по разделу 3**

В результате работы над разделом были созданы все необходимые функции. Веб-приложение предоставляет возможность регистрации, авторизации, создания, удаления, открытия, генерации контрольных и лабораторных работ, дополнительных заданий со стороны преподавателя, а также возможность записи, решения и получения этих работ и заданий со стороны студента. Все запросы были протестированы на успешный и неуспешный результат работы, а также задокументированы с помощью приложения Postman

# **Разработка клиентской части**

## **Разработка дизайна пользовательского интерфейса**

Для разработки дизайн макета был выбран графический редактор – Figma. Данный сервис является одним из самых распространенных решений во многих компаниях по ряду причин:

* наличие онлайн-доступа к сервису, что предоставляет возможность постоянного доступа ко всей информации без потери данных и необходимости скачивания приложения на устройство,
* возможность командной работы с разграничением доступа, что, например, позволяет разработчикам ориентироваться на дизайн-макет без необходимости постоянной пересылки файлов и случайного видоизменения макета,
* наличие прототипирования, позволяющее протестировать взаимодействие пользователя с системой как можно раньше, что благоприятно сказывается на уменьшении сроков разработки и своевременной корректировки построения логики разрабатываемой системы,
* наличие большого количества плагинов, предоставляющих необходимые изображения, иконки, а также ускоряющих работу [11].

Для предотвращений расхождений в дизане наиболее используемые элементы были перенесены в ui-kit и трансформированы в компоненты. Ui-kit – это полный набор элементов и компонентов, необходимый для сборки большого однородного продукта. Данный набор включает в себя компоненты фильтрации, карточек, кнопок, поиска и вопросы с вариантами ответа. Более подробно с набором компонентов можно ознакомиться на рисунке 41.

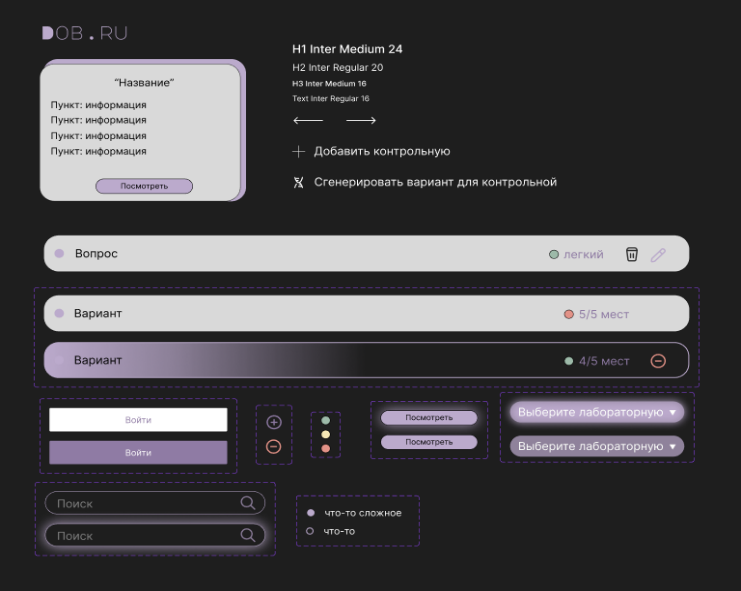


Рисунок 41 – Ui-kit дизайн-макета разрабатываемой системы

Вся система была поделена на несколько страниц, с учетом функциональных требований разных групп пользователей. Так, административная панель включает в себя страницы лабораторных и контрольных работ, дополнительных заданий, записей, проверки работ, банка вопросов и пользователей. Панель студента, в свою очередь, состоит из страниц лабораторных и контрольных работ, дополнительных заданий, личного кабинета и дедлайнов. Подробнее с дизайном каждой страницы можно ознакомиться в ПРИЛОЖЕНИИ А (Дописать) Для пользователей, чей доступ еще не подтвержден администратором, была создана приветственная страница, предупреждающая об отсутствии доступа (Рисунок 42). Также была разработана типовая страница для неверных маршрутов (Рисунок 43)



Рисунок 42 – Приветственная страница пользователя с неподтвержденным доступом

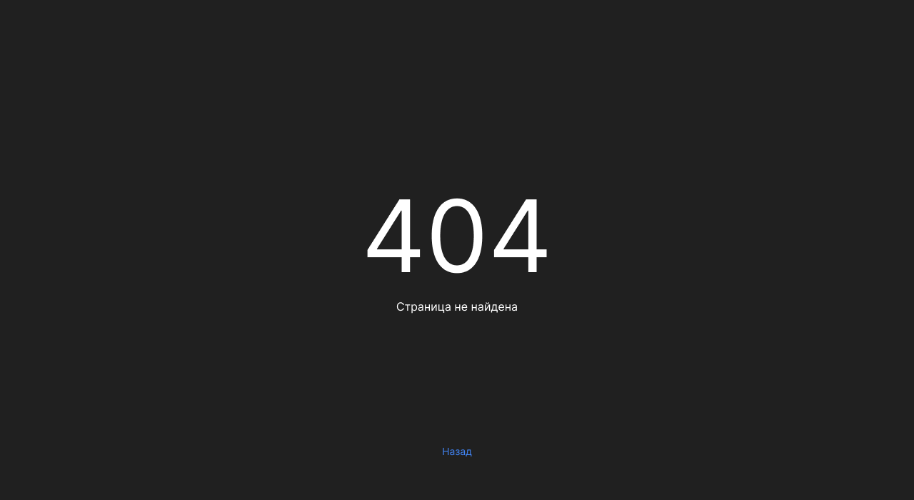


Рисунок 43 – Страница для неверных маршрутов

## **Разработка клиентской части административной панели**

Разработка административной панели началась с переноса дизайн макета на React с помощью JSX и CSS. В отрисовке использовался компонентный подход, благодаря которому страницы делятся на элементы и блоки, которые впоследствии можно переиспользовать. Для реализации карточек вопросов был создан отдельный компонент, так как его стили сильно отличаются от стилей карточек контрольных, лабораторных работ, дополнительных заданий и студентов (Рисунок 44 и 45).

Изображение выглядит как текст, снимок экрана, компьютер, программное обеспечение

Автоматически созданное описание

Рисунок 44 – Компонент карточки контрольных, лабораторных работ, дополнительных заданий, пользователей и решений

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 45 – Компонент карточки вопроса

Также были созданы следующие папки с функциональными компонентами и соответствующими стилями:

* Card,
* Controls,
* Labs,
* Filter,
* Tasks,
* Popup,
* QuestionCard,
* TableAppointments,
* TableStudents,
* Variant,
* Header,
* Footer,
* Enter.

Как говорилось раннее, проверка доступа должна осуществляться не только на сервере, но и на клиенте. Так после авторизации пользователя в системе проверяется его роль, после чего пользователь перенаправляется на соответствующий маршрут с помощью хука React.useNavigate. Также при монтировании компонента происходит вторая проверка на роль пользователя, чтобы отследить изменения localstorage и предотвратить неавторизованное использование системы. Административная панель состоит из следующих маршрутов:

* + /sign-in – страница авторизации,
  + /admin – страница по работе с контрольными работами,
  + /admin/tasks – страница по работе с дополнительными заданиями,
  + /admin/labs – страница по работе с лабораторными работами,
  + /admin/users – страница по предоставлению доступа пользователям,
  + /admin/questions – страница по работе с банком вопросов,
  + /admin/appointments – страница с таблицей записей,
  + /admin/controls/variant/:variantId – страница просмотра сгенерированного варианта контрольной работы,
  + /admin/labs/variant/:variantId – страница просмотра созданного варианта лабораторной работы,
  + /admin/tasks/:taskId – страница просмотра созданного дополнительного задания,
  + /admin/check – страница с решениями контрольных работ студентов,
  + /admin/check/:checkId – страница проверки контрольной работы студента.

Для перехода между маршрутами было создано меню, которое переиспользуется в зависимости от роли пользователя. Так для администратора меню выглядит следующим образом (Рисунок 46).

Изображение выглядит как текст, программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 46 – Компонент меню администратора

Помимо переиспользуемых компонентов по мере разработки были выявлены переиспользуемые функции, одна из которых направлена на форматирование даты, приходящей с сервера. Даты с сервера приходят в формате “YYYY-MM-DD”, что не удобно для запоминания и прочтения, поэтому такие даты надо переводить в формат “DD month”. Функция форматирования даты представлена на рисунке 47.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 47 – Реализация функции форматирования даты

Для реализации основного функционала административной панели и взаимодействия клиента с сервером был создан файл “api.js”, в котором прописаны запросы к серверу. В основном запросы были написаны с помощью метода fetch. Так, например, на рисунке 48 представлена реализация запроса на сервер для создания лабораторных работ, где this.getAuth() – метод для получения токена авторизации, а this*.*checkResponse – метод для обработки ответа. Стоит отметить, что после создания лабораторной работы не вызывается дополнительный запрос на получение списка лабораторных работ для того, чтобы не перегружать сервер. Вместо этого создана переменная состояния с помощью хука React.useState, которая хранит массив лабораторных работ. Так, если запрос на создание лабораторной работы завершился успешно, то лабораторная добавляется в массив (Рисунок 49). Таким же образом была построена вся работа веб-приложения.

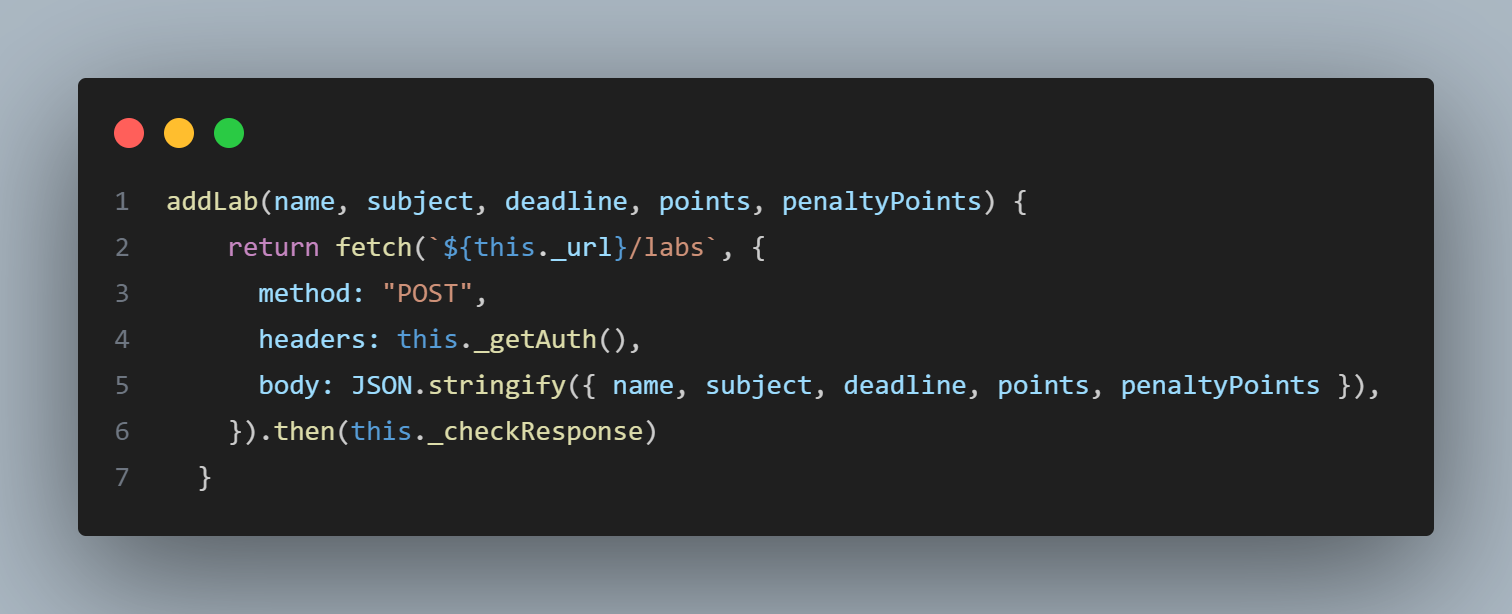


Рисунок 48 – Реализация запроса на сервер для создания лабораторной работы

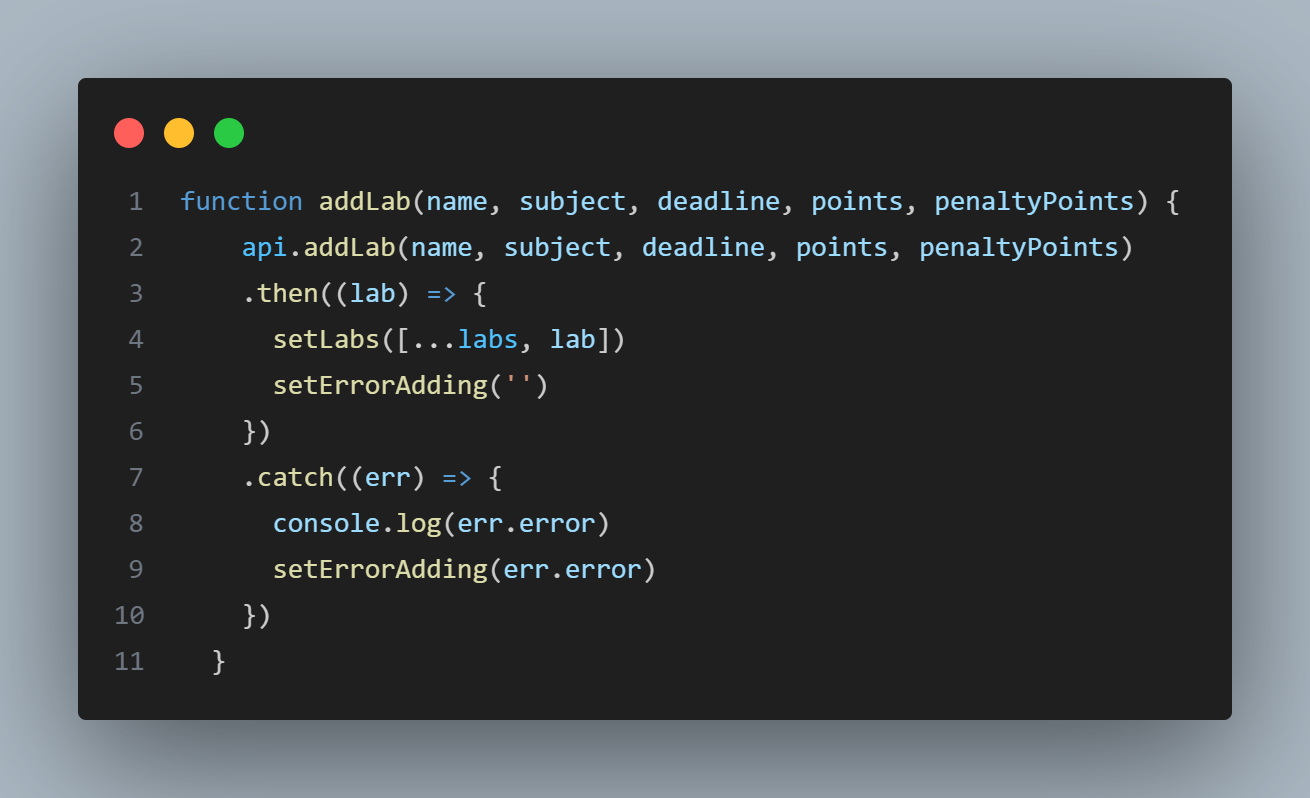


Рисунок 49 – Функция создания лабораторной работы

Помимо метода fetch для некоторых запросов использовалась библиотека axios. Данная библиотека используется для всех запросов, которые включают в себя передачу файлов, потому что axios предоставляет удобные средства по работе с этим типом данных, например FormData. Реализация загрузки файла при создании варианта лабораторной работы представлена на рисунке 50.



Рисунок 50 – Реализация запроса по созданию варианта лабораторной работы с помощью библиотеки axios

Функционал фильтрации было принято реализовывать на клиенте, не отправляя запрос на сервер, как это делается на многих сервисах. У данного решения есть несколько причин. Несмотря на возможные проблемы с безопасностью, реализация фильтрации на клиенте обеспечивает мгновенный результат работы, что делает интерфейс более отзывчивым, а также данное решение снижает и нагрузку на сервер. Для предотвращения возможных XSS атак была сделана валидация данных фильтрации и поиска, а также предусмотрено удаление потенциально опасных символов. Пример реализации функции фильтрации пользователей по группе и поиск по совпадению вводимых данных с фамилиями пользователей представлен на рисунке 51.

Изображение выглядит как текст, снимок экрана, мультимедиа, Шрифт

Автоматически созданное описание

Рисунок 51 – Реализация функции фильтрации и поиска пользователей

## **Разработка клиентской части панели студента**

Так как многие компоненты из административной панели переиспользуются в панели студента, то необходимо было перенести дизайн только уникальных компонентов, таких как:

* AppointmentCard – карточка для записи на варианты контрольных, лабораторных работ и дополнительные задания,
* Profile – страница личного кабинета пользователя,
* UserControl – компонент решения контрольной работы,
* Register – компонент регистрации.

В процессе реализации панели студента были созданы следующие маршруты:

* + - /sign-up – страница регистрации,
    - /user – страница записей на варианты лабораторных работ,
    - /user/tasks – страница записей на дополнительный задания,
    - /user/controls – страница для просмотра доступных контрольных работ,
    - /user/controls/:controlId – страница решения варианта контрольной работы,
    - /user/deadline – страница с таблицами дедлайнов по лабораторным работам и дополнительным заданиям с указанием максимального возможного количества баллов на сегодня с учетом всех штрафов,
    - /user/profile – страница личного кабинета.

Главной особенностью функционала студента является наличие регистрации. Для компонента регистрации использовалась библиотека react-hook-form. Данная библиотека упрощает работу с формами, проверяет вводимые данные с регулярными выражениями и самостоятельно создает объекты ошибок в случае несовпадения с установленными условиями. Так как регистрация содержит большое количество полей ввода, то использование данной библиотеки оказалось необходимым. В первую очередь необходимо было создать регулярные выражения. Необходимо, чтобы номер группы начинался с заглавной буквы английского алфавита, ведь неправильное указание группы сильно влияет на работу студента в системе. Также были созданы регулярные выражения для почты и ссылок на профили telegram и github:

* + - * /^[A-Z].\*/ – для номера группы,
      * /^https?:\/\/(t\.me|telegram\.me)\/[a-zA-Z0-9\_]+$/ – для ссылки на профиль telegram,
      * /^https?:\/\/github\.com\/[a-zA-Z0-9\_-]+$/ – для ссылки на профиль github,
      * /^[^\s@][+@[^\s@]+\.[^\s@]+$/](mailto:+@[%5e\s@%5d+\.%5b%5e\s@%5d+$/) – для почты.

Пример поля ввода для ссылки на профиль telegram представлен на рисунке 52. Если ошибок формы нет, то кнопка регистрации становится активной. После чего вызывается функция регистрации, которая отправляет запрос на сервер, и при успешном ответе сохраняет информацию о текущем пользователе и перенаправляет его на страницу авторизации (Рисунок 53).

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 52 – Реализация функции регистрации

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 53 – Реализация функции регистрации пользователя

Функционал студента заключается в основном в записи на работы и решении контрольных работ. Для записи на варианты работ предусматривается три стиля карточки: карточки с наличием пустых мест, карточка с отсутствием пустых мест, выбранная карточка. Весь компонент карточки представлен на рисунке 54.

Изображение выглядит как текст, снимок экрана, компьютер, дисплей

Автоматически созданное описание

Рисунок 54 – Компонент карточки записи на варианты лабораторных работ и дополнительные задания

Сама запись происходит по нажатию на кнопку, после чего вызывается соответствующая функция (Рисунок 55). Принцип облегчения работы сервер просматривается и здесь, ведь для обновления информации о записях на вариант лабораторной не вызывается еще один запрос для получения списка обновленных лабораторных работ. Вместо этого используется хук React.useState, с помощью которого создается массив лабораторных работ, и при записи на вариант лабораторной находится ее индекс в массиве и обновляется информацией из результата запроса. Для взаимодействия с сервером был создан файл “userApi.js”. На рисунке 56 можно увидеть POST запрос с помощью встроенного метода fetch.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

Рисунок 55 – Реализация функции записи на вариант лабораторной работы

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 56 – Реализация запроса на сервер для записи на вариант лабораторной работы

Процесс решения контрольной работы начинается с получения случайного варианта доступной контрольной работы. Для этого на сервер отправляется запрос на получения варианта выбранной контрольной работы (Рисунок 57). После чего студент перенаправляется на маршрут /user/controls/:controlId, где controlId – идентификатор варианта контрольной работы. Вместе с перенаправлением передается компонент варианта через location.state. В компоненте “UserControl” отрисовываются вопросы с вариантами ответа и открытые вопросы с полями для ввода текста. С кодом компонента можно ознакомиться на рисунке 58. При изменении полей ввода и выбора варианта ответа вызываются функции, изменяющие состояние переменной userAnswers. При сохранении ответов происходит отправка ответов на сервер, где вопросы с вариантами ответов проверяются автоматически по совпадению с отмеченными преподавателем правильными ответами. При успешном запросе пользователь перенаправляется на страницу контрольных работ, открывается всплывающее окно об успешном прохождении теста и показываются предварительные баллы (Рисунок 59).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 57 – Реализация запроса получения случайного варианта контрольной работы

Изображение выглядит как текст, снимок экрана, дисплей, компьютер

Автоматически созданное описание

Рисунок 58 – Компонент “UserControl”

Изображение выглядит как текст, снимок экрана, компьютер, программное обеспечение

Автоматически созданное описание

Рисунок 59 – Реализации функции сохранения решения контрольной работы и автоматической проверки ответов

## **Тестирование клиентской части**

Тестирование с помощью PerfectPixel, LightHouse, БЭМ валидатор, проверка кроссбраузерности и адаптива

## **Вывод по разделу 3**

Вывод по разделу 3

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения выпускной квалификационной работы были составлены и проанализированы функциональные и нефункциональные требования к разработке. На основе данных требований был сделан обзор существующих система управления учебной деятельностью. В результате анализа не было найдено прямых аналогов, подходящих под требования и желания Заказчика, но были выявлены косвенные, у которых есть ряд преимуществ, однако они не могут обеспечить Заказчику необходимые возможности системы по управлению учебной деятельностью. На основании выполненного анализа были сделаны выводы о необходимости разработки персональной системы управления учебной деятельностью.

На этапе проектирования были построены диаграммы, которые демонстрируют существующие бизнес-процессы и функциональные требования, распределенные по группам пользователей, взаимодействие требований и пользователей друг с другом, а также диаграммы, показывающие внутреннее строение системы, взаимодействие частей системы между собой в тех или иных функциях и строение базы данных разрабатываемого веб-приложения.

Разработка серверной части системы осуществлялась на express, фреймворке Node.js, с использованием NoSQL СУБД MongoDB. Была реализована требуемая Заказчиком функциональность: регистрация, авторизация, работа с дополнительными заданиями, контрольными и лабораторными работами с учетом разделения данных функций относительно групп пользователей. Все запросы были протестированы и задокументированы посредством приложения “Postman”.

Разработка клиентской части системы началась с создания дизайн-макета. На этом этапе использовался графический редактор Figma. Сама разработка осуществлялась на React, библиотеки JavaScript, с использованием библиотек react-router, react-router-dom, react-hook-form и соответствующих хуков. Для взаимодействия клиента с сервером в основном использовался встроенный метод fetch, для запросов с использованием файлов применялась библиотека axios и ее встроенное средство для передачи данных formData. В результате был создан и отрисован дизайн-макет системы управления учебной деятельностью, реализован весь необходимый функционал с разделением доступа между разными группами пользователей и налажено взаимодействие клиента с серверов. Клиентская часть была протестирована посредством Дописать

Таким образом была разработана серверная и клиентская части системы управления учебной деятельностью по заказу преподавателя практики факультета инфокоммуникационных технологий, Добрякова Давида Ильич. Система соответствует всем функциональным и нефункциональным требованиям Заказчика по надежности, безопасности и общей функциональности. Система автоматизирует рабочий процесс преподавателя, не выдает критических ошибок. В дальнейшем планируется расширять систему, предоставляя возможности по загрузке решений лабораторных работ и дополнительных заданий, а также возможности по отслеживанию баллов по всем работам.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. MongoDB, Express.js, React.js, Node.js: лучший технологический стек? [Электронный ресурс] // it.world – URL: https://www.it-world.ru/tech/technology/193941.html (дата обращения 08.02.2024).
2. IDEF0: что такое и как используется [Электронный ресурс] // Группа компаний Четыре Д – URL: https://software.by/info/blogs/1c/idef0-what-is-and-how-to-use/ (дата обращения 14.02.2024).
3. Аутсорсинг или заказная разработка на стеке MERN [Электронный ресурс] // Work Solutions – URL: https://worksolutions.ru/useful/autsorsing-ili-zakaznaya-razrabotka-na-steke-MERN/ (дата обращения 08.02.2024).
4. В чем особенности MongoDB и когда эта база данных вам подходит: руководство для новичков [Электронный ресурс] // Журнал VK Cloud “Завтра облачно” – URL: https://cloud.vk.com/blog/osobennosti-mongodb-kogda-baza-dannyh-vam-podhodit (дата обращения 20.02.2024).
5. Диаграмма деятельности uml [Электронный ресурс] // StudFiles – URL: https://studfile.net/preview/9431262/page:11/ (дата обращения 05.03.2024).
6. Диаграмма последовательности (sequence diagram) [Электронный ресурс] // ФСИС\_семинар-7\_Диаграмма-последовательности – URL: https://clck.ru/39qyyP (дата обращения 25.02.2024).
7. Контур Класс [Электронный ресурс] // Контур Класс – URL: https://kontur.ru/class (дата обращения 10.02.2024).
8. Менеджерам обучения [Электронный ресурс] // Teachbase – URL: https://help.teachbase.ru/ (дата обращения 10.02.2024).
9. Брэдшоу Ш., Брэзил Й., Ходоров К. Mongo DB. Полное руководство - М.: ДМК Пресс, 2020. - 540 с.
10. <https://learning.postman.com/docs/introduction/overview/>
11. <https://slashdesigner.ru/figma-guide> книга по фигме