Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
Л.Н. Храмова
подпись инициалы, фамилия
«14» 06 2024 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ «РАСПИСАНИЕ» ДЛЯ КГБ ПОУ ЛЕСОСИБИРСКИЙ ТЕХНОЛОГИЧЕСКИЙ ТЕХНИКУМ

Руководитель	14.06.2024	доцент, канд. пед. наук	А.В. Фирер
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	14.06.2024		А.А. Копейкин
	подпись, дата		инициалы, фамилия
Нормоконтролер	14.06.2024		А.В. Фирер
	подпись, дата		инициалы, фамилия

Лесосибирск 2024

РЕФЕРАТ

Выпускная квалификационная работа по теме «Проектирование и разработка веб-приложения «Расписание» для КГБПОУ Лесосибирский технологический техникум» содержит 68 страниц текстового документа, 17 иллюстраций, 1 таблицу, 40 использованных источников.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, PYTHON, FLASK, REACT, JAVA-SCRIPT, GIT, GITFLOW, ВЕБ-РАЗРАБОТКА, ВЕБ-ПРИЛОЖЕНИЕ.

Цель исследования – теоретически обосновать и разработать веб-приложение «Расписание» для управления расписанием занятий в КГБПОУ «Лесосибирский технологический техникум».

Объект исследования – процесс управления расписанием в образовательной организации.

Предмет исследования – процесс проектирования и разработки веб-приложения как инструмента для управления расписанием в «КГБПОУ Лесосибирский технологический техникум».

Для достижения поставленной цели были сформулированы следующие задачи:

– на основе анализа предметной области, выделить требования к веб-приложению «Расписание» и разработать модели системы;

– обосновать выбор архитектуры веб-приложения «Расписание» и инструментальных средств его разработки;

– разработать серверную и клиентскую части системы и интегрировать их с созданной базой данных, внедрить разработанное веб-приложение в образовательный процесс КГБПОУ «Лесосибирский технологический техникум».

В результате выполнения выпускной квалификационной работы разработано и внедрено в образовательный процесс КГБПОУ «Лесосибирский технологический техникум» веб-приложение «Расписание».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

С развитием информационных технологий в современном мире многие сферы жизни сталкиваются с необходимостью использования современных IT-решений, включая образование. Одной из таких областей является управление расписанием занятий в образовательных учреждениях. Ручное создание и поддержка расписания требуют значительных трудовых и временных затрат, что часто приводит к ошибкам. Чтобы обеспечить максимальную эффективность обучения в современных техникумах и колледжах с большим количеством студентов и преподавателей, необходимо эффективно организовать учебный процесс. Разработка автоматизированной информационной системы «Расписание» является одной из ключевых задач в этом контексте, так как она позволит автоматизировать процесс составления расписания занятий и учебных групп, а также обеспечит быстрый и удобный доступ к информации о расписании как для студентов, так и для преподавателей.

Цель работы – теоретически обосновать и разработать веб-приложение «Расписание» для управления расписанием занятий в КГБПОУ «Лесосибирский технологический техникум». Данная система должна облегчить процесс создания расписания и поддержания его в актуальном состоянии, снизить вероятность ошибок и обеспечить быстрый доступ к информации о расписании для всех заинтересованных лиц.

Цель обусловила постановку и решение следующих задач:

– на основе анализа предметной области, выделить требования к веб-приложению «Расписание» и разработать модели системы;

– обосновать выбор архитектуры веб-приложения «Расписание» и инструментальных средств его разработки;

– разработать серверную и клиентскую части системы и интегрировать их с созданной базой данных, внедрить разработанное веб-приложение в образовательный процесс КГБПОУ «Лесосибирский технологический техникум».

Методы исследования:

− теоретические: анализ учебной и научно-технической литературы по теме исследования, обобщение и сравнительный анализ, моделирование

− эмпирические: наблюдение, интервьюрирование.

Результаты исследования были представлены на следующих научных мероприятиях:

1. II Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 10–15 апреля 2023 г., участие).

2. VII Всероссийская научно-практической конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 1–2 ноября 2023 г., диплом I степени).

3. Конкурс проектов «Точка роста» в рамках VIII Всероссийского (IX Регионального) молодежного форума «Российское могущество прирастать будет Сибирью…» (г. Лесосибирск, ЛПИ – филиал СФУ, 14 декабря 2023 г., участие)

4. III Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г., участие).

5. Конкурс проектов «Точка роста» научно-исследовательской направленности молодежного научно-образовательного фестиваля «Ступени» в рамках III Всероссийского молодежного научного форума «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г., участие).

6. XX Международная научная конференция студентов, аспирантов и молодых ученых «Проспект Свободный – 2024» (г. Красноярск, СФУ, 15–20 апреля 2024 г., участие).

По результатам исследования принята к публикации статья: Копейкин А. А. / Архитектура веб-приложения для управления расписанием: ключевые аспекты. / А. А. Копейкин // Цифровые технологии в образовании, науке и технике: материалы XX Международная научная конференция студентов, аспирантов и молодых ученых «Проспект Свободный – 2024», г. Красноярск, / Сибирский федеральный университет. – Красноярск, 2024.

Структура работы – работа состоит из введения, двух глав, заключения, списка используемых источников, включающего 40 наименований, 8 приложений. Результаты работы представлены в 17 иллюстрациях и 1 таблице. Общий объем работы – 68 страниц.

1 Проектирование веб-приложения «Расписание»

1.1 Анализ предметной области

Перед началом проектирования веб-приложения необходимо проанализировать предметную область, в качестве которой выбран процесс автоматизации расписания в КГБПОУ «Лесосибирский технологический техникум».

Анализ предметной области является важным этапом в проектировании веб-приложения «Расписание». Этот этап включает в себя изучение особенностей образовательного процесса, структуры учебного заведения, текущих методов управления расписанием, а также выявление требований и ожиданий пользователей системы.

Автоматизация процесса создания и просмотра расписания позволит значительно повысить эффективность работы техникума, улучшить управление учебным процессом и удовлетворить потребности всех участников образовательного процесса.

1.1.1 Структура учебного заведения

КГБПОУ «Лесосибирский технологический техникум» (далее – техникум) представляет собой учебное учреждение среднего профессионального образования, включающее в рассматриваемом контексте следующие основные компоненты:

а) учебные группы:

1) группы студентов распределены по курсам и специальностям;
2) каждая группа может быть разделена на подгруппы для посещения лабораторных и практических занятий.

б) преподаватели:

1) преподавательский состав состоит из специалистов, ведущих различные дисциплины;

2) преподаватели могут вести занятия в нескольких группах и подгруппах.

в) аудитории и учебные корпуса:

1) учебные помещения включают лекционные залы, лаборатории, компьютерные классы и специализированные кабинеты;

2) аудитории имеют различные характеристики и оборудованы для проведения определенных видов занятий;

3) для каждого учебного корпуса определены свои аудитории.

1.1.2 Текущие методы управления расписанием

В настоящее время управление расписанием в техникуме осуществляется вручную. Этот процесс включает несколько этапов:

а) сбор данных:

- информация о занятиях, преподавателях, учебных группах и доступных аудиториях собирается и обрабатывается вручную;
- учебные планы и программы служат основой для составления расписания.

б) составление расписания:

- расписание составляется с учетом множества факторов, включая занятость преподавателей, доступность аудиторий и учебные планы;
- процесс планирования часто требует значительных временных и трудовых затрат.

в) публикация расписания:

- готовое расписание фиксируется в электронных таблицах, которые затем распечатываются и размещаются на информационных стендах;
- изменения в расписании вносятся вручную.

1.1.3 Проблемы и ограничения текущих методов

Анализ текущих методов управления расписанием выявил ряд проблем и ограничений, которые могут быть решены с помощью автоматизации:

а) высокая трудоемкость:

- ручное составление и обновление расписания требует значительных временных затрат со стороны администраторов;
- процесс подвержен ошибкам из-за человеческого фактора.

б) низкая оперативность:

- внесение изменений в расписание занимает много времени, что затрудняет оперативное реагирование на изменения и потребности.

в) отсутствие прозрачности:

- студенты и преподаватели могут испытывать трудности с доступом к актуальной информации о расписании;
- расписание может быть недоступно вне учебного заведения или в нерабочее время.

г) ограниченная гибкость:

- ручное планирование ограничивает возможности оптимизации и автоматической проверки расписания на наличие конфликтов.

1.1.4 Определение цели автоматизации и потенциальных пользователей

На основе анализа предметной области можно выделить несколько цель создания веб-приложения для автоматизации части процессов составления расписания и его редактированием: оптимизация процесса составления расписания и обеспечение оперативного доступа к информации. Потенциальными пользователями системы являются обучающиеся и преподаватели (неавторизованные пользователи), диспетчер и администратор системы (авторизованные пользователи)

1.2 Требования к веб-приложению «Расписание»

Проектирование веб-приложения «Расписание» начинается с детального анализа и четкого определения требований. На данном этапе на основе изучения существующих методов и процессов составления и редактирования учебного расписания в техникуме проводится определение функциональных и нефункциональных требований.

В своей книге Карл Вигерс [3, с. 9] дает следующее определение функциональным требованиям «Функциональные требования (functional requirements) определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес-требований.» Таким образом функциональные требования описывают основные функции, которые должна выполнять система «Расписание». Важными аспектами функциональности системы являются составление и редактирование расписания, управление пользователями и предоставление удобного интерфейса для обучающихся и преподавателей. Разрабатываемое веб-приложение должно выполнять следующие функции:

а) составление и редактирование расписания:

- ввод информации о занятиях, преподавателях, учебных группах и учебных предметах, аудиториях, времени проведения занятий;
- редактирование и обновление расписания в реальном времени.

б) управление пользователями:

- регистрация и аутентификация пользователей (администраторов, диспетчеров);
- разграничение прав доступа в зависимости от ролей пользователей.

в) возможность просмотра расписания на различных устройствах для пользователей:

- доступ к актуальному расписанию занятий в любое время и с любого устройства (компьютеры, мобильные телефоны, планшеты);
- отображение расписания в соответствии с выбранными фильтрами.

г) аутентификация и авторизация:

- реализация механизмов аутентификации для ограничения доступа к серверной части системы только авторизованным пользователям;
- использование хэш-функции, для защиты паролей пользователей, согласно рекомендациям стандарта ГОСТ Р 34.11—2012 [7].

Нефункциональные требования определяют общие характеристики системы, такие как производительность, безопасность и удобство использования. Анализ источников [4; 13] позволил выделить следующие требования:

а) производительность:

- максимальное приемлемое время отклика для веб-приложений считается 5 секунд;
- система должна поддерживать одновременную работу не менее 200 пользователей без снижения производительности.

б) надежность и отказоустойчивость:

- автоматическое резервное копирование данных должно выполняться ежедневно, а время на восстановление данных из резервной копии не должно превышать 1 часа. Такие меры обеспечивают защиту данных и их восстановление в случае сбоя;
- система должна корректно обрабатывать все ошибки и исключения, обеспечивая непрерывную работу и логирование всех инцидентов для дальнейшего анализа.

в) удобство использования

− интерфейс пользователя должен быть интуитивно понятным и доступным, с минимальным количеством кликов для выполнения основных операций;
− время на обучение пользователей (студентов, преподавателей, администрации) должно быть минимальным, не превышая 2 часов для каждого типа пользователя.

г) защита от атак:

− применение мер защиты от распространенных уязвимостей, таких как использование параметризированных запросов для предотвращения SQL-инъекций и XSS-атак.

1.3 Архитектура системы

Архитектура автоматизированной информационной системы «Расписание» определяет структурную организацию системы, взаимодействие её компонентов и основные технологические решения, которые будут использованы при разработке. Архитектура должна обеспечить гибкость, масштабируемость, надежность и безопасность системы.

Основой для системы «Расписание» является клиент-серверная архитектура, которая обеспечивает эффективное взаимодействие между компонентами через сеть. В своей книге А.А. Дубаков [11, с. 8] писал, что «Любой компьютер, сервер, подключённый к локальной или глобальной сети, называется хостом (host- хозяин, принимающий гостей). Клиент/серверная модель представляет собой архитектуру разработки приложений, предназначенную для отделения уровня представления данных от их обработки и хранения.»

Применение клиент-серверной архитектуры позволяет достичь следующих целей:

а) Разделение обязанностей. Клиентская часть отвечает за отображение расписания и взаимодействие с пользователем, тогда как серверная часть

обрабатывает запросы, выполняет бизнес-логику и управляет данными. Это разделение позволяет эффективно организовать работу приложения и обеспечить пользователей необходимым функционалом.

б) Масштабируемость и производительность. Разделение системы на клиентскую и серверную части позволяет каждой из них масштабироваться независимо, обеспечивая высокую производительность даже при большом количестве пользователей. Это также способствует отказоустойчивости системы.

в) Безопасность и защита данных. Серверная часть отвечает за защиту данных, включая аутентификацию пользователей, контроль доступа и шифрование данных. Это обеспечивает надежную защиту конфиденциальной информации и предотвращает несанкционированный доступ.

Принципы чистой архитектуры (Clean Architecture), применяются для создания модульных, гибких и легко поддерживаемых систем. Основные идеи чистой архитектуры в своей книге описал Р. Мартин [18, с. 203]:

− независимость от фреймворков: код приложения не привязан к конкретным технологиям, что обеспечивает гибкость и возможность легкого перехода на другие инструменты;

− простота тестирования: архитектура способствует легкости тестирования каждого компонента системы;

− независимость от пользовательского интерфейса и базы данных: позволяет изменять интерфейс и тип базы данных без влияния на бизнес-логику;

− независимость от внешних сервисов: делает приложение более гибким и устойчивым к изменениям во внешнем окружении.

Компоненты системы взаимодействуют посредством четко определённых интерфейсов и протоколов:

– клиент-серверное взаимодействие: веб-интерфейс (клиент) посылает запросы к серверу через HTTP/HTTPS. Сервер обрабатывает запросы, выполняет бизнес-логику и взаимодействует с базой данных;

– API и микросервисы: использование RESTful API для обмена данными между клиентом и сервером. Возможность разбиения системы на микросервисы для повышения гибкости и масштабируемости.

Проанализировав источники [6; 28; 33] было принято решение внедрения следующих мер обеспечивающих безопасность системы:

а) аутентификация и авторизация:

- использование протокола JWT (JSON Web Tokens) для аутентификации пользователей;
- разграничение прав доступа на основе ролей.

б) шифрование данных:

- использование SSL/TLS для шифрования данных при передаче;
- шифрование чувствительных данных в базе данных.

в) защита от атак:

- реализация мер для защиты от распространённых веб-атак, таких как SQL-инъекции, XSS (Cross-Site Scripting) и CSRF (Cross-Site Request Forgery).

Архитектура системы «Расписание» направлена на создание гибкой, масштабируемой и надежной системы, способной удовлетворить потребности всех категорий пользователей и обеспечить эффективное управление учебным процессом в КГБПОУ Лесосибирский технологический техникум.

1.4 Моделирование системы

Моделирование системы является важным этапом разработки программного обеспечения, так как позволяет наглядно изобразить структуру и поведение системы, до начала ее реализации, что существенно сокращает время на реализацию и позволяет четко определить задачи разработки. Одним из методов

моделирования является создание ER-моделей (Entity-Relationship Models). Данная модель была предложена Питером Ченом в 1976 году.

1.4.1 ER-Модель

Проектирование ER-Модели является универсальным и подходит как для проектирования модели базы данных, так и для любой системы. Этот этап определяет основные функции, сущность, атрибуты к ним и правила взаимодействия. В учебном пособии Д. А. Попова-Коварцева [27, с. 51] связи между сущностями определяются следующим образом: «Модель «сущность–связь» основывается на некоторой важной семантической информации о реальном мире и предназначена для логического представления данных. Она определяет значения данных в контексте их взаимосвязи с другими данными.»

В рамках исследования построим ER-модель для базы данных веб-приложения «Расписание» для техникума. Данное проектирование включает в себя определение сущностей, связей между ними и функциями системы.

Представленная на рисунке 1 модель, описывает взаимодействие сущностей между собой, что обеспечивает функционирование системы и работу основных функций, таких как составление, редактирование и просмотр расписания.

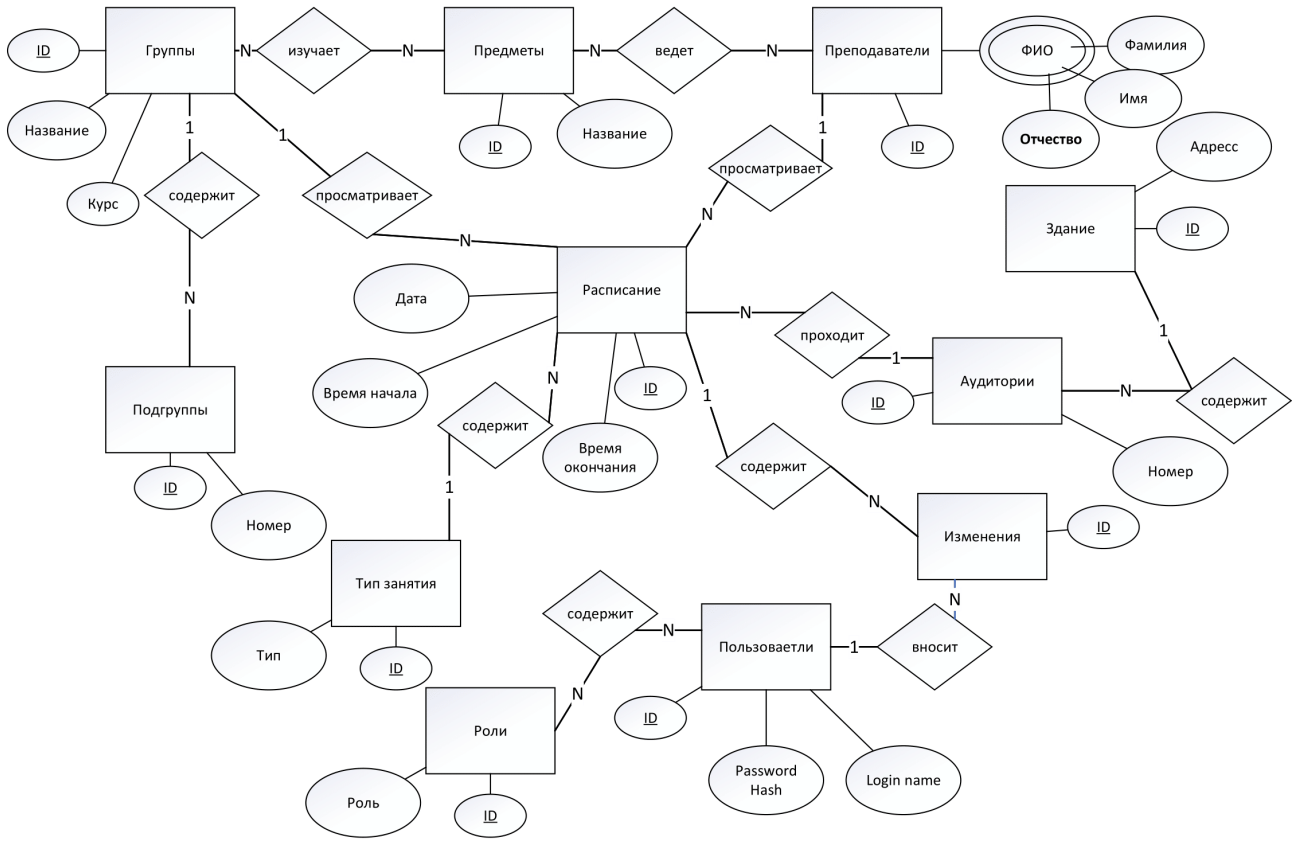

Рисунок 1 – ER-модель

В рамках проектирования определены следующие сущности и их взаимосвязи:

а) сущности:

1) преподаватель;

2) предметы;

3) группы;

4) подгруппы;

5) расписание;

6) тип занятия;

7) аудитории;

8) здание;

9) пользователи;

10) изменения;

11) роли.

б) Связи между сущностями:

1) преподаватель ведет несколько предметов;
2) группа студентов обучается нескольким предметам;
3) группа разделяются на подгруппы;
4) расписание содержит изменения, которые могу вносить несколько пользователей;
5) занятия проходят в разных аудиториях;
6) аудитории располагаются в разных зданиях;
7) занятия имеют определенный тип.

1.4.2 Модели взаимодействия с системой

Модели взаимодействия с системой представляют собой диаграммы взаимодействия, которые Мацяшек Л. А. [20, с. 223] описал следующим образом: «Диаграммы взаимодействий разделяются на два вида – диаграммы последовательностей и диаграммы коммуникации (до появления версии UML 2.0 они назывались диаграммами кооперации). Диаграмма последовательностей представляет собой двумерный граф. Роли (объекты) располагаются по горизонтали.» Эти модели используются для визуализации и анализа потоков данных, а также для понимания логики взаимодействия между различными элементами системы. В данном разделе представлены две основные модели взаимодействия: взаимодействие студента с системой и взаимодействие диспетчера расписания с системой.

Демонстрация процесса представлена на рисунке 2. Студент запрашивает и получает расписание через интерфейс пользователя (UI). Этот процесс содержит несколько ключевых этапов:

− инициирование запроса: студент с помощью UI отправляет запрос на просмотр расписания;

– обработка запроса на сервере: запрос передается на сервер, который выполняет соответствующий SQL-запрос для извлечения данных из базы данных;

– возвращение данных: данные расписания возвращаются от сервера через UI обратно студенту.

Данная модель позволяет понять, как данные переносятся от базы данных к конечному пользователю и какие шаги включены в этот процесс.

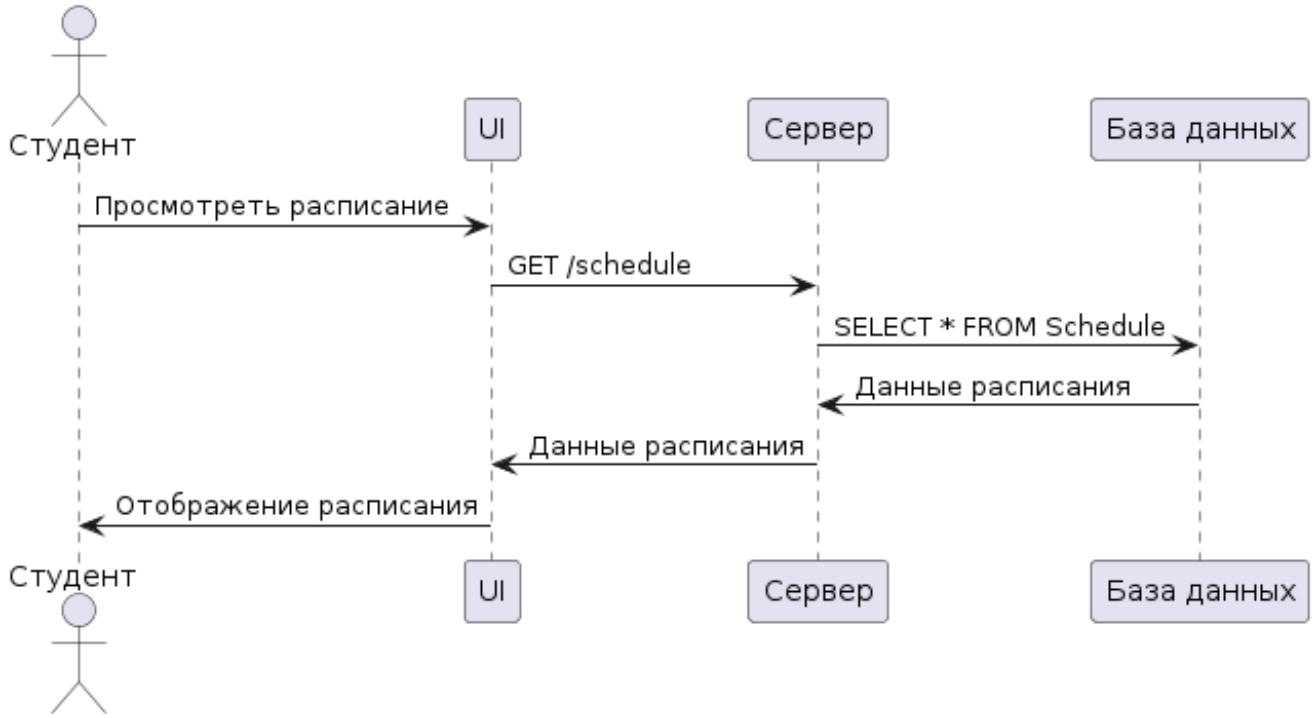

Рисунок 2 – Взаимодействие неавторизованного пользователей с системой

На рисунке 3 представлен процесс добавления нового расписания диспетчером. Процесс добавления включает в себя следующие шаги:

– инициирование добавления: диспетчер отправляет данные нового расписания через UI административной части;

– обработка данных на сервере: UI отправляет POST-запрос на сервер, который выполняет SQL-запрос для добавления данных в базу данных;

– получение результата: сервер возвращает результат операции на UI, информируя диспетчера об успехе или неудаче добавления расписания с помощью всплывающих уведомлений;

– уведомление пользователей: сервер отправляет уведомление о добавлении нового расписания в систему уведомлений, которая затем оповещает соответствующих пользователей.

Модель взаимодействия диспетчера с расписанием позволяет понять процесс внесения изменений в систему и их влияние на другие компоненты и пользователей системы.

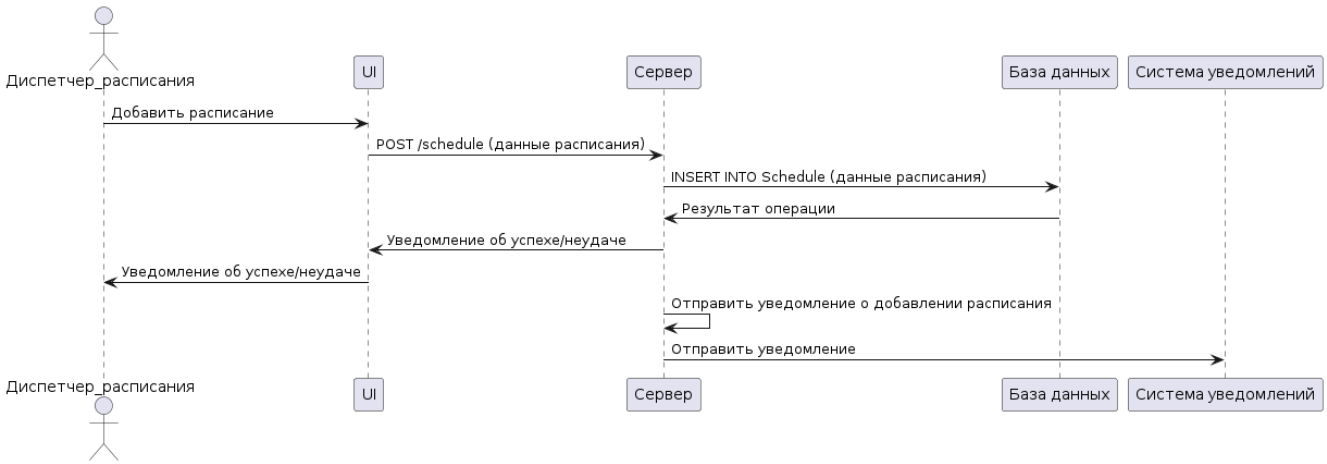

Рисунок 3 – Взаимодействие авторизованного пользователя с системой

Диаграммы последовательностей отображают взаимодействие с системой, что является важным инструментом для анализа и проектирования веб-приложения «Расписание». Они позволяют четко представить последовательность действий и обмена данными между различными участниками и компонентами системы, что способствует более эффективной разработке веб-приложения.

2 Разработка веб-приложения «Расписание»

2.1 Выбор инструментальных средств разработки

Выбор технологий и инструментов для разработки автоматизированной информационной системы «Расписание» играет ключевую роль для успешной реализации. Основными критериями выбора являются производительность, масштабируемость, удобство разработки и поддержка сообщества.

Рассмотрим выбранные инструментальные средства, представленные в таблице 1.

Таблица 1 – Инструментальные средства разработки

Инструментальные средства	**Версии**	**Преимущества**	**Недостатки**	**Назначение**
React JS [39]	18.2.0	Высокая производительность; Компонентный подход; Виртуальный DOM	Высокий порог вхождения	Создание пользовательских интерфейсов
Python [38]	3.11	Простота и читаемость кода; Большая стандартная библиотека; Поддержка многопоточности и асинхронности	Низкая производительность по сравнению с компилируемыми языками; Высокая потребность в памяти	Разработка веб-приложений, научных вычислений, скриптов
Flask [37]	2.2.3	Легковесность; Простой в изучении и использовании; Многочисленные расширения	Ограниченная функциональность; Подходит для небольших приложений	Разработка веб-приложения и API
MySQL [10]	8.0.33	Высокая производительность	Ограниченная поддержка современных типов данных	Управление реляционными базами данных

Окончание таблицы 1

Инструментальные средства	Версии	Преимущества	Недостатки	Назначение
Git [8]	2.41.0	Распределённая система контроля версий; Высокая производительность; Поддержка ветвления и слияния	Сложность в освоении для новичков; Возможны конфликты при слиянии	Контроль версий и совместная разработка
Gitflow [9]	-	Упрощение управления ветвлением; Чёткая структура разработки	Сложность для небольших проектов.	Управление процессом разработки с использованием Git
Visual Studio Code [40]	1.90.2	Лёгкость и быстрота; Поддержка множества расширений; Интеграция с система ми контроля версий	Большое потребление оперативной памяти	Редактирование и отладка кода

Для разработки клиентской части системы выбрана библиотека React JS, так как она обладает высокой производительностью, компонентным подходом, так же существует большая поддержка сообщества разработчиков и имеется множество готовых библиотек. React JS позволяет создавать интерактивные пользовательские интерфейсы и управлять состоянием приложения с использованием Redux или Context API.

Серверная часть системы использует язык программирования Python. Его основные преимущества включают простоту и удобочитаемость, обширную стандартную библиотеку и поддержку множества фреймворков и инструментов. В качестве веб-фреймворка был выбран Flask, который отличается легковесностью, гибкостью и возможностью быстрого старта разработки. Flask легко интегрируется с различными библиотеками и инструментами.

В качестве хранения данных используется реляционная система управления базами данных MySQL, обеспечивающая высокую производительность и

надежность. MySQL применяется для хранения информации о расписании, пользователях, учебных группах и аудиториях.

Для управления исходным кодом и совместной работы используется система контроля версий Git. Инструментарий Gitflow помогает управлять версиями и координировать работу в команде, обеспечивая удобное введение новых функций, исправление ошибок и релизов, что способствует структурированному и упорядоченному процессу разработки.

Использование данных технологий и инструментов обеспечивает создание эффективной, надежной и легко поддерживаемой системы, соответствующей современным тенденциям и подходам к разработке программного обеспечения.

2.2 Создание и ведение репозитория с помощью Git и Gitflow

В электронной документации Git [8] дается следующее определение системы контроля версий – «Система контроля версий – это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определённой версии». Ее важность объясняется тем, что для распределенных команд нужна система управления, чтобы отслеживать изменения, происходящие со временем. Это позволяет шаг за шагом видеть, какие файлы изменились и как. Особенно это полезно при анализе выполненной работы в рамках одной задачи: возможность вернуться назад к предыдущим версиям бывает неоценима.

2.2.1 Создание репозитория

Рассмотрим процесс создания репозитория.

1. Инициализация Git репозитория:

Для инциализации репозитория необходимо перейти в корневую директорию проекта «cd shedules-app/». Затем с помощью команды git init инициализируем новый Git репозиторий.

2. Создание репозитория на GitHub:

Для создания нового репозитория на GitHub необходимо связать локальный репозиторий с удалённым. Для этого необходимо использовать следующую команду: «git remote add origin https://github.com/yourusername/schedule-app.git». После связи локального репозитория с удаленным нужно проиндексировать все файлы проекта с помощью команды: «git add .».

Создание первого коммита инициализирует репозиторий и сохраняет текущее состояние проекта: «git commit -m "Initial commit"». Далее необходимо отправить изменения на удаленный репозитой командой: «git push -u origin master».

2.2.2 Введение Gitflow

Gitflow – это стратегия управления ветками в Git, которая помогает организовать разработку, упрощает процессы релизов и позволяет поддерживать стабильность основной ветки. В целом GitFlow состоит из двух постоянных веток и нескольких типов временных веток. Пример ветвления, следуя данной стратегии, представлен на рисунке 4.

Основные ветки в Gitflow:

- master — основная ветка, всегда содержит стабильный код, готовый для релиза;
- develop — это ветка для разработки. Потенциально она может быть нестабильная.

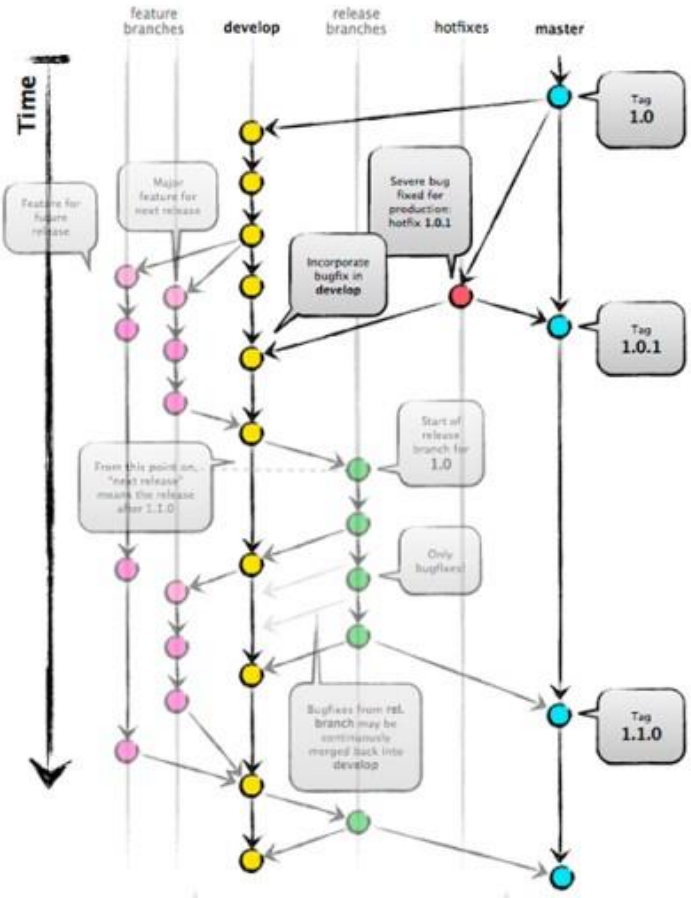

Рисунок 4 – Gitflow

Вспомогательные ветки:

– feature/* — ветки для разработки новой функциональности;

– release/* — ветки для подготовки выпуска новой версии проекта;

– hotfix/* — быстрое решение дефекта, который нашли уже реальные пользователи на реальном сервере.

Преимущества Gitflow:

− организация и структура: Gitflow чётко определяет роли веток и процесс работы, что способствует порядку в разработке;

− параллельная работа над функциями: разработчики могут работать над различными функциями независимо друг от друга, создавая отдельные feature-ветки;

− упрощение процесса релиза: отдельные release-ветки позволяют готовить релиз, внося последние исправления и проверки без риска нарушить стабильность ветки develop;

− быстрое исправление критических ошибок: Hotfix-ветки позволяют быстро исправлять критические баги и сразу деплоить исправления, не дожидаясь завершения работы над другими фичами;

− четкость и контроль версий: Gitflow облегчает управление версиями и контроль над выпуском новых релизов и исправлений.

Инициализация Gitflow в проекте осуществляется с помощью команды: «git flow init». Вывод команд инициализации репозитория представлен на рисунке 5.

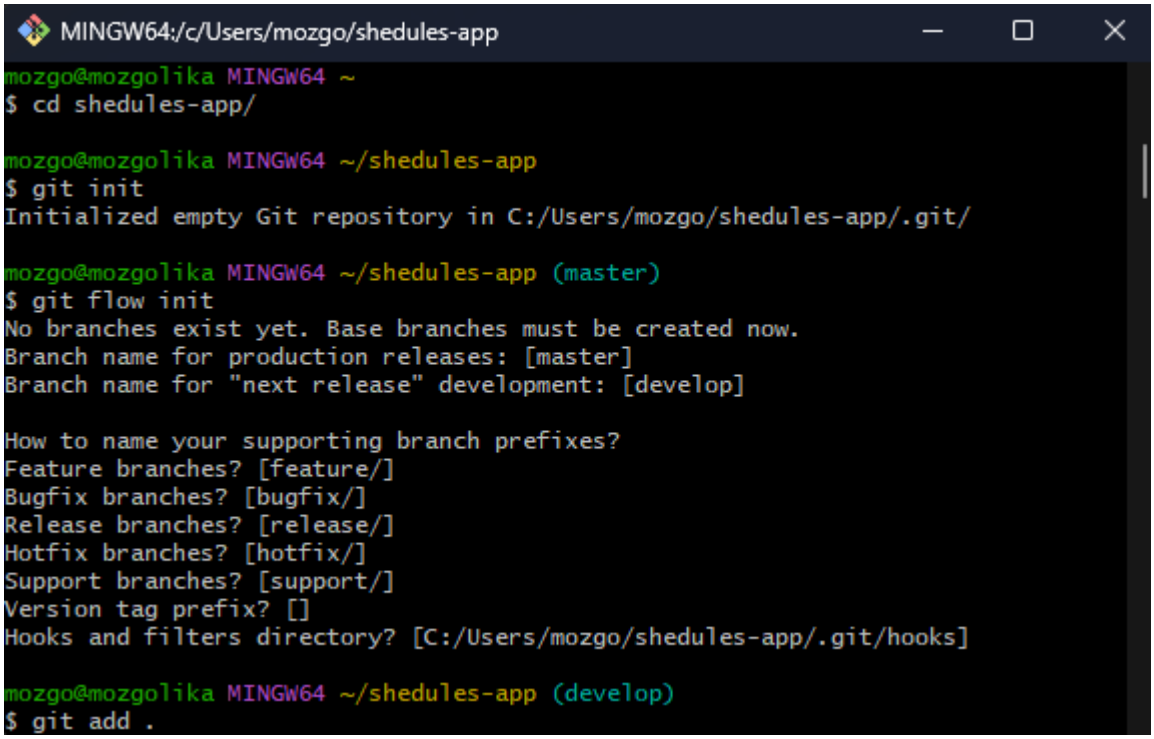

Рисунок 5 – Создание и инициализация репозитория

2.2.3 Основные команды Gitflow

Перечислим основные команды Gitflow, используемые в данном иследовании.

1. Создание новой feature-ветки.

Для разработки новой функциональности в проекте, Gitflow предлагает создание временной ветки feature. Это позволяет изолировать изменения, связанные с новой функцией, от основной ветки разработки.

Для создания новой функциональности необходимо начать новую временную ветку с помощью команды: «git flow feature start feature-name».

Данная команда создает новую ветку с именем feature-«название ветки», основанную на текущей ветке develop. После того как работа над новой функциональностью завершена и все необходимые изменения внесены, нужно завершить ветку feature и объединить ее с веткой develop. Для этого используется команда: «git flow feature finish feature-name»

После завершения работы над feature-веткой, изменения должны быть отправлены в удалённый репозиторий, чтобы синхронизировать работу с другими членами команды. Изменения отправляются с помощью команды: «git push origin develop».

2. Создание release-ветки.

Для подготовки к новому релизу, создается временная release-ветка. Данная ветка предназначена для подготовки к выпуску новой версии, включающей финальные исправления, тестирование и документацию. Для создания новой release-ветки используется команда: «git flow release start release-version». После завершения всех подготовительных работ для релиза, нужно завершить release-ветку, объединяя изменения как в ветку master, так и в ветку develop: «git flow release finish release-version».

Release-ветка обеспечивает плавный переход от стадии разработки к стадии релиза, позволяя финализировать изменения и подготовить проект к выпуску.

3. Создание hotfix-ветки.

Иногда в проекте возникают критические ошибки, требующие немедленного исправления. Для таких случаев Gitflow предлагает создание временной hotfix-ветки, что позволяет быстро вносить исправления без воздействия на текущую

разработку. Для создания новой hotfix-ветки используется команда: «git flow hotfix start hotfix-name».

Эта команда создаст ветку с именем hotfix-name, основанную на текущей ветке master. После внесения всех необходимых исправлений, нужно завершить hotfix-ветку, объединяя изменения как в ветку master, так и в ветку develop: «git flow hotfix finish hotfix-name».

Hotfix-ветка позволяет оперативно и эффективно устранять критические ошибки, обеспечивая стабильность и надежность проекта.

2.3 Разработка клиентской части

В данном разделе подробно описан процесс создания клиентской части веб-приложения для расписания с использованием библиотеки React. Опишем этапы создания проекта, а также разработки пользовательского интерфейса.

2.3.1 Создание проекта на React

React – это библиотека JavaScript с открытым кодом для создания внешних пользовательских интерфейсов. В отличие от других библиотек JavaScript, предоставляющих полноценную платформу приложений, React ориентируется исключительно на создание представлений приложений через инкапсулированные единицы (называются компонентами), которые сохраняют состояние и генерируют элементы пользовательского интерфейса.

Для работы с React необходимо установить Node.js и npm (Node Package Manager). Node.js позволяет запускать JavaScript на сервере разработки, а npm управляет пакетами и зависимостями проекта. Node.js и npm устанавливаются с официального сайта следуя инструкции по установке.

Для того чтобы настроить проект необходимо использовать Create React App. Это удобный инструмент, который позволяет быстро создать и настроить новый проект на React без необходимости конфигурировать вручную сборщики, транспилеры и другие инструменты. Для установки необходимо выполнить следующую команду «npx create-react-app schedule-app», перейти в каталог проекта и с помощью команды «npm start» запустить проект. После выполнения данных команд будет создан проект и запущен сервер разработки, а приложение будет доступно по адресу http://localhost:3000/. После открытия этот адрес в браузере, увидим начальную страницу React.

После создания проекта с помощью Create React App, структура каталогов будет следующей:

а) node_modules/ — каталог, содержащий установленные npm-пакеты;

б) public/ — статические файлы, которые будут служить базой для приложения;

− index.html — основной HTML-файл приложения;

в) src/ — исходные файлы проекта;

− App.js — основной компонент приложения;

− index.js — точка входа в приложение.

2.3.2 Разработка пользовательского интерфейса панели Администратора

Для функционирования системы должны быть заполнены и актуализированы все справочники, заполнение каждого справочника и связей между ними осуществляется с помощью следующих форм:

− добавить предмет;

− добавить преподавателя;

− добавить группу;

− добавить предмет преподавателю.

Форма, представленная на рисунке 6, предназначена для добавления новых предметов в справочник. Эта форма не только упрощает процесс ввода данных, но и включает механизм проверки на наличие конфликтов и дубликатов предметов, уже существующих в справочнике. Таким образом, система обеспечивает целостность и актуальность данных.

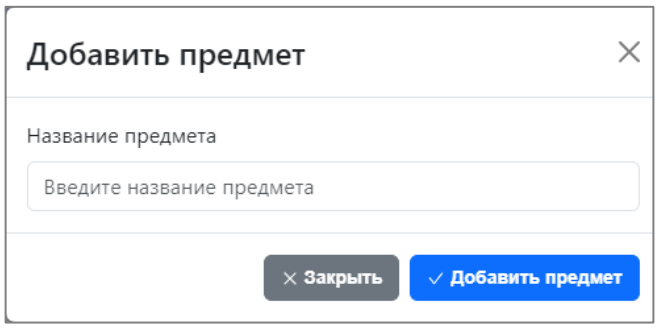

Рисунок 6 – Форма «Добавление предмета»

На рисунке 7 представлена форма, которая используется для ввода информации о новых преподавателях. Представленная форма помогает администратору легко и быстро добавлять новых сотрудников, обеспечивая при этом проверку корректности введенных данных и исключая возможность дублирования информации.

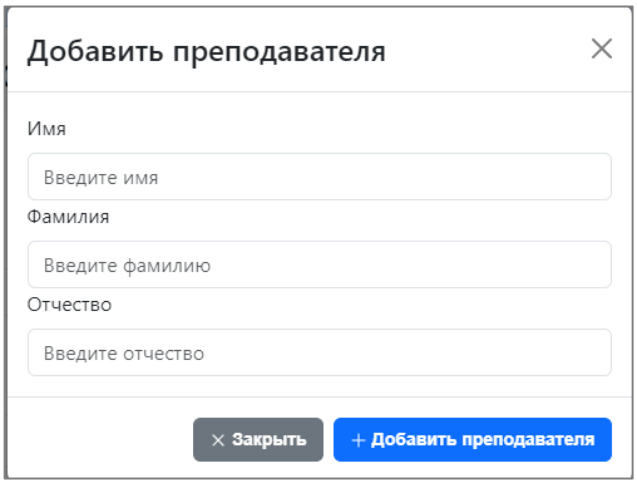

Рисунок 7 – Форма «Добавление нового преподавателя»

На рисунке 8 представлена форма для добавления новых групп студентов. Она позволяет администраторам вводить данные о группах, автоматически проверяя их на наличие ошибок и дублирующих записей. Это помогает поддерживать актуальность информации в системе и облегчает дальнейшее планирование расписания.

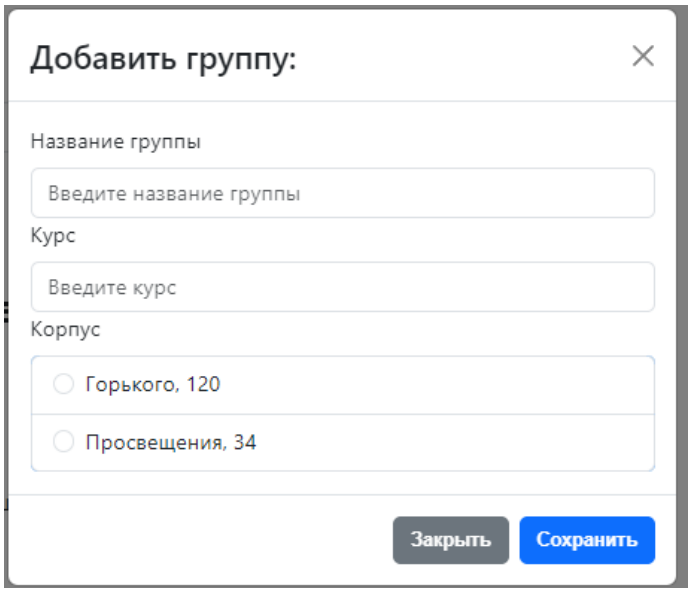

Рисунок 8 – Форма «Добавление группы»

Форма, представленная на рисунке 9 позволяет назначать предметы конкретным преподавателям. Этот процесс важен для установления связей между преподавателями и предметами, что в дальнейшем облегчит создание расписания и управление учебным процессом.

Для работы с расписанием основной страницей является «Расписание занятий», она представлена на рисунке 10. На данной странице расположены параметры, которые нужно учитывать, чтобы начать заполнять расписание:

– выбор корпуса;

– выбор группы;

– выбор даты, на которую заполняется расписание.

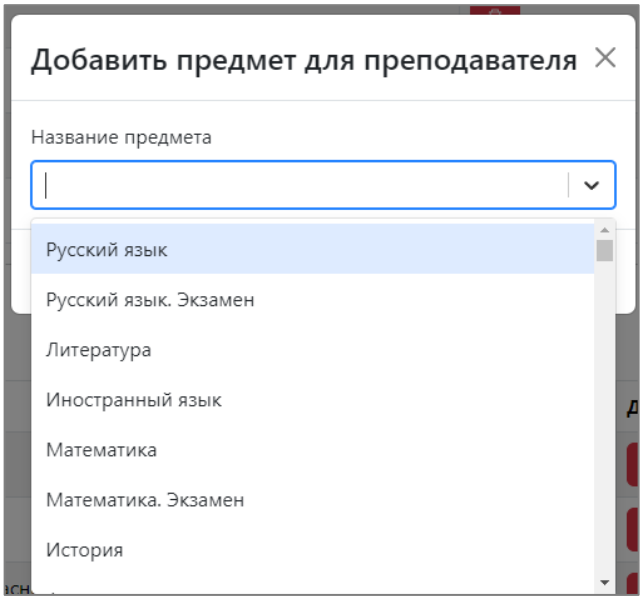

Рисунок 9 – Форма «Добавление предмета преподавателю»

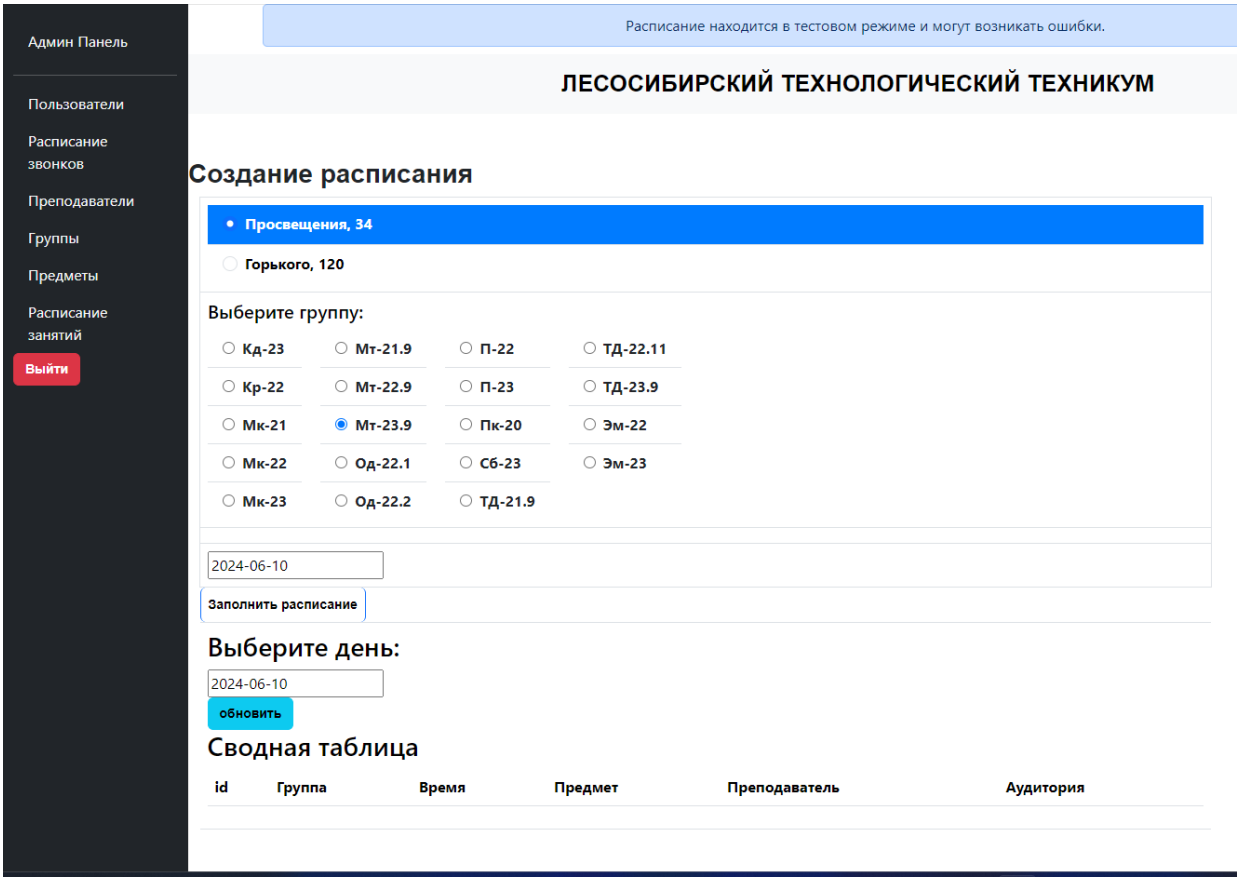

Рисунок 10 – Основная страница для работы с расписанием

Форма заполнения расписания, представленная на рисунке 11, является ключевым элементом для работы с расписанием. Она позволяет администратору назначать занятия для конкретной группы на один учебный день. Этот подход был выбран на основе обратной связи от диспетчеров расписания и результатов промежуточного тестирования системы. В данной форме также предусмотрена возможность разделения на подгруппы, что позволяет проводить разные занятия для подгрупп одновременно. Кроме того, была добавлена функция копирования занятий, что значительно ускоряет процесс заполнения расписания при наличии повторяющихся занятий.

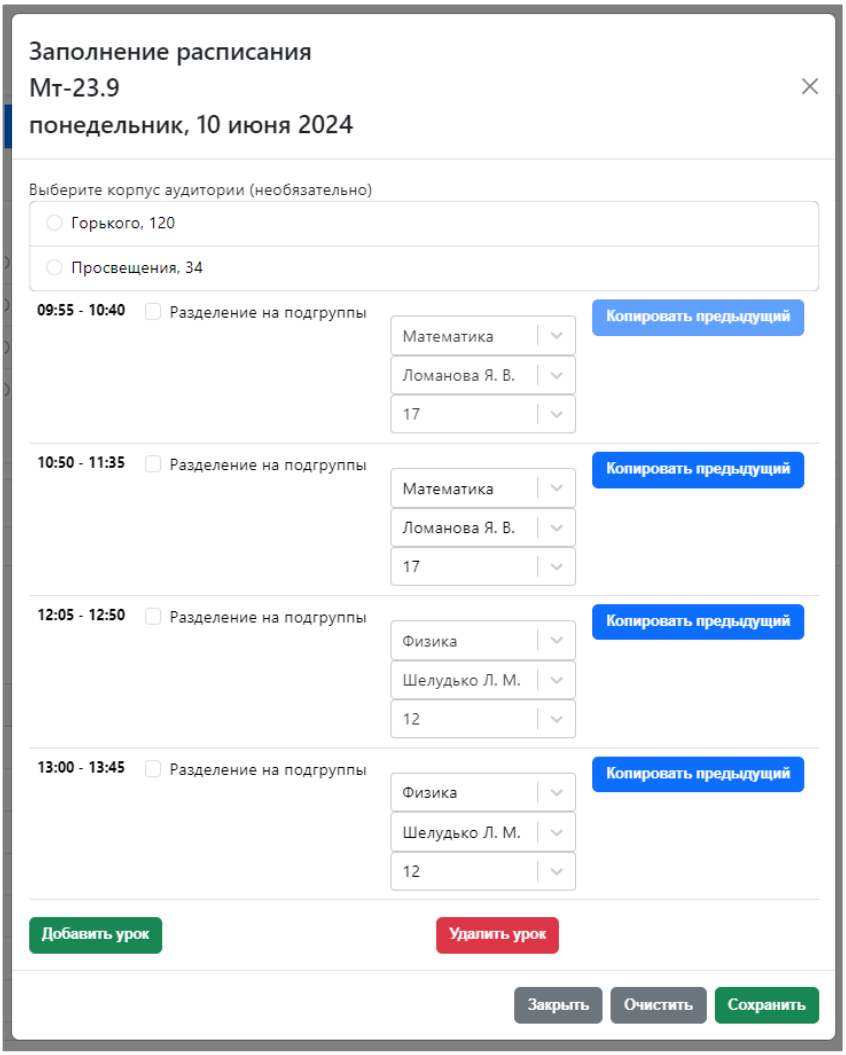

Рисунок 11 – Форма «Заполнение расписания»

На страницу «Расписание занятий» была добавлена сводная таблица расписания, представленная на рисунке 12. Эта таблица служит для контроля и проверки заполненного расписания диспетчером. Благодаря этому инструменту диспетчер может легко отслеживать актуальность и корректность введенных данных, своевременно вносить необходимые изменения и избегать возможных конфликтов в расписании.

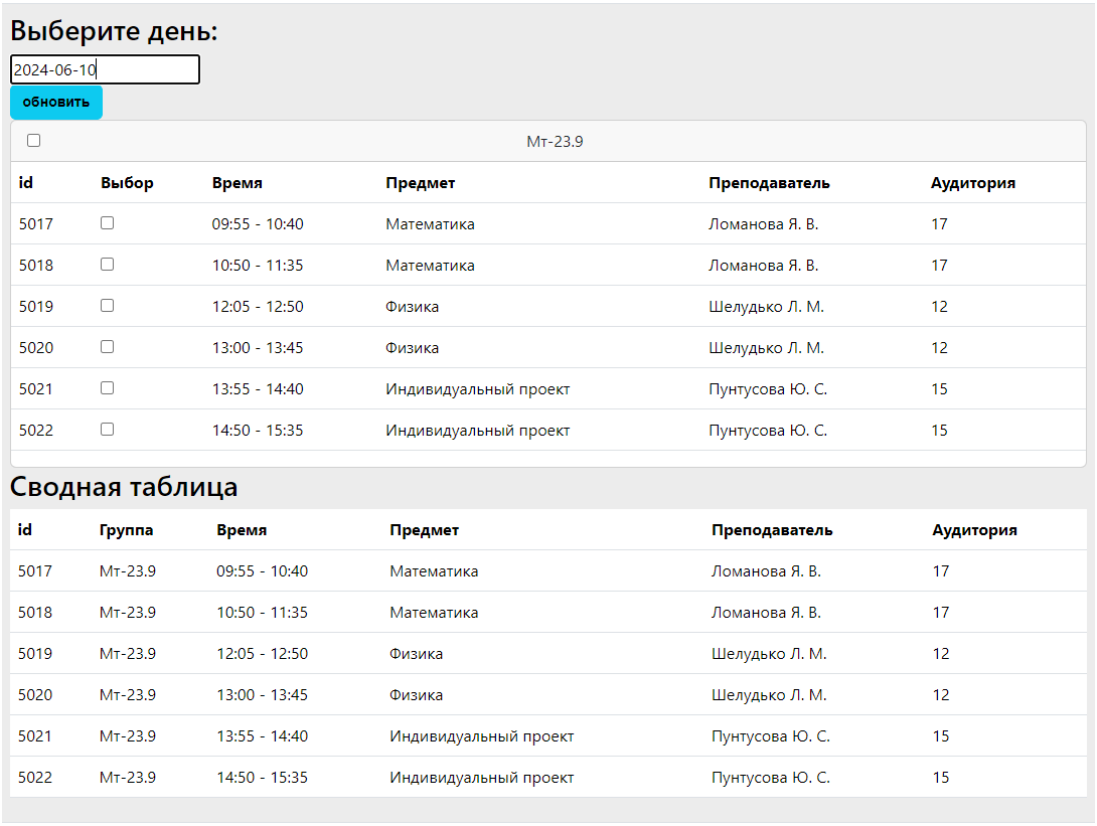

Рисунок 12 – Сводная таблица расписания на один день

2.3.3 Разработка пользовательского интерфейса общедоступных страниц

Для получения доступа к расписанию неавторизованных пользователей были разработаны следующие страницы:

− выбор группы;

− расписание для группы;

− расписание преподавателя.

Выбор группы, представленный на рисунке 13, предлагает пользователю последовательно выбрать корпус, курс и соответствующую группу. Такой подход уменьшает область поиска нужной группы и позволяет быстро перейти к расписанию выбранной группы.

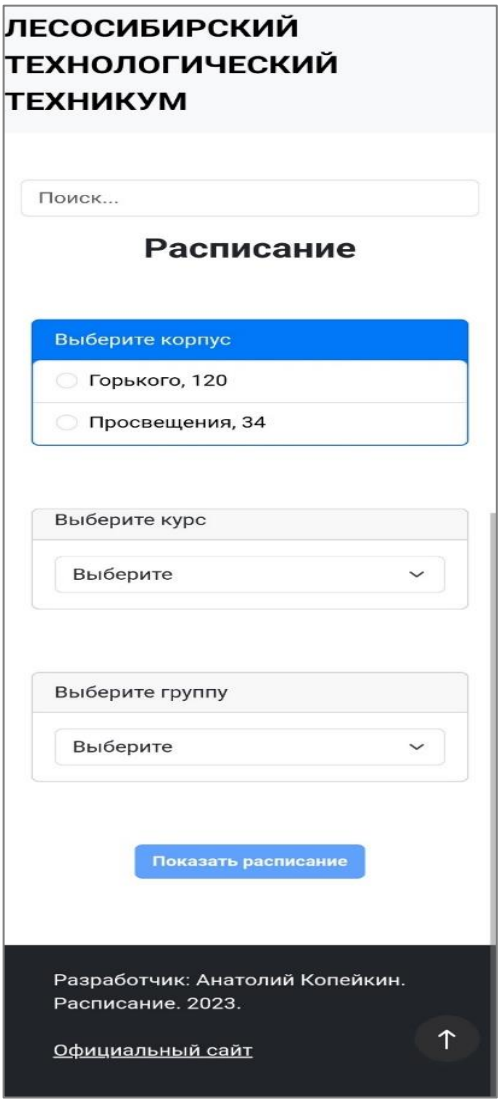

Рисунок 13 – Выбор группы

Расписание для группы, представленное на рисунке 14 в виде карточек, позволяет пользователю легко ориентироваться в расписании. Карточки с

предметами являются интерактивными кнопками, при нажатии на которые открывается расписание преподавателя, отображенное на рисунке 15. Помимо этого, принято решение отображать актуальное расписание на ближайшие два дня. Это позволяет гибко вносить изменения до официального опубликования расписания.

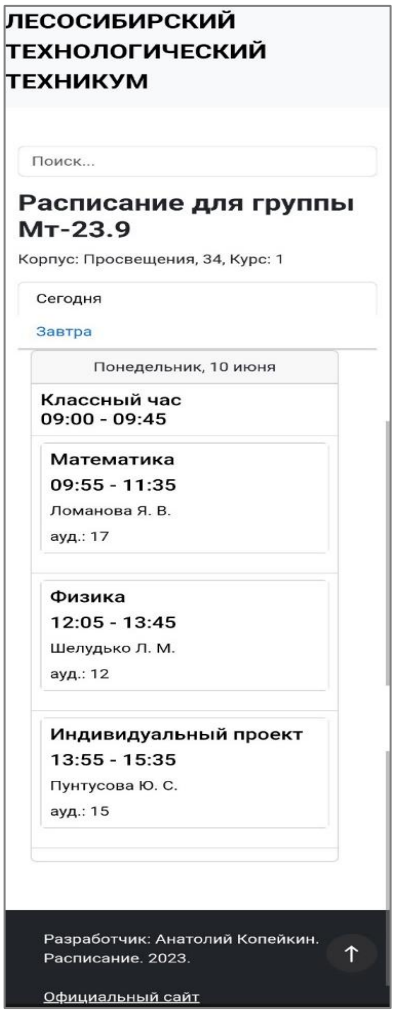

Рисунок 14 – Расписание для группы

Расписание преподавателя, представленное на рисунке 15, отличается от расписания групп. Оно показывает расписание на текущую неделю с указанием следующих параметров для каждого занятия:

− предмет;

– время проведения;

– группа;

– адрес, аудитория.

Данные параметры являются ключевыми для каждого занятия в расписании.

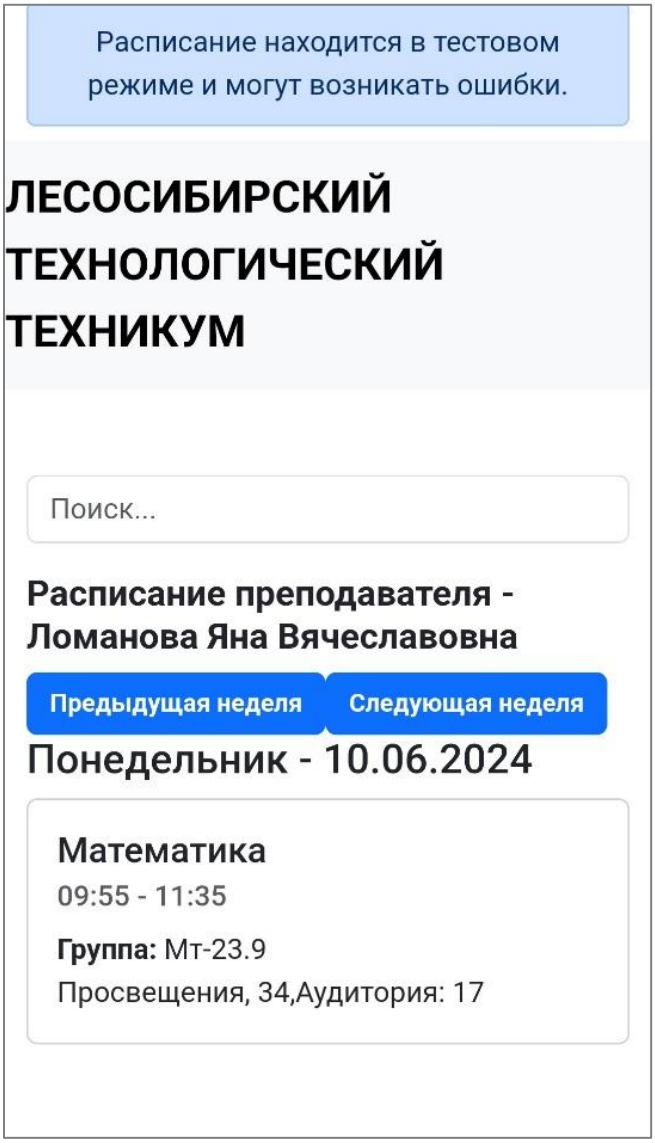

Рисунок 15 – Расписание преподавателя

На рисунке 16 представлен пример поиска группы. Модуль поиска позволяет пользователю быстро находить нужную группу, по ее названию, что значительно ускоряет процесс перехода к нужному расписанию.

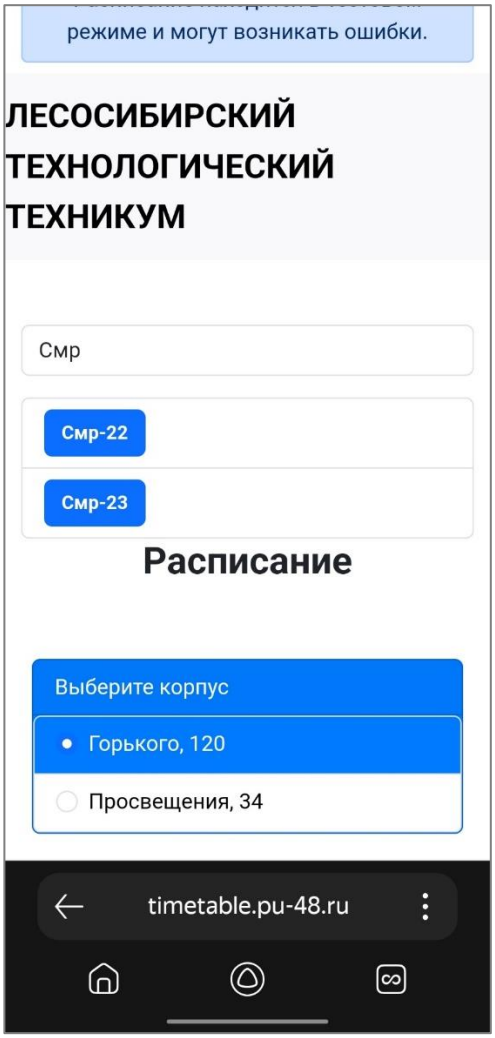

Рисунок 16 – Поиск группы

На рисунке 17 представлен пример поиска преподавателя. Модуль поиска позволяет пользователю быстро находить нужного преподавателя по имени или фамилии, что облегчает доступ к расписанию преподавателя.

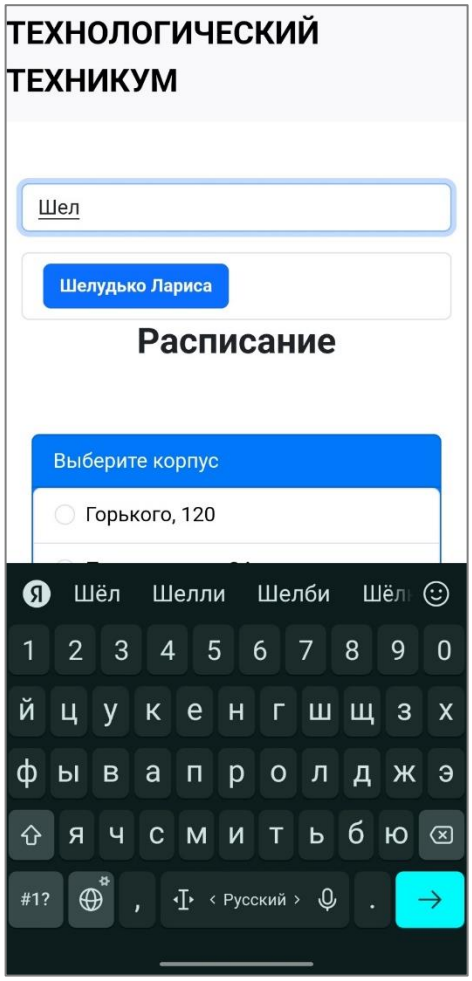

Рисунок 17 – Поиск преподавателя

Таким образом в процессе разработки пользовательского интерфейса общедоступных страниц были созданы несколько ключевых элементов, обеспечивающих удобный и быстрый доступ к информации о расписании.

Были разработаны три основных страницы:

– страница выбора группы, которая позволяет пользователю последовательно выбрать корпус, курс и группу;

– страница расписания для группы, представленная в виде карточек. Этот формат облегчает визуальное восприятие и навигацию по расписанию. Карточки являются интерактивными и позволяют быстро переходить к расписанию преподавателя;

– страница расписания преподавателя, где представлено расписание на текущую неделю с подробной информацией о каждом занятии. Это позволяет преподавателям эффективно планировать своё время и готовиться к занятиям.

На каждую страницу был интегрирован модуль поиска, позволяющий быстро находить нужные группы и преподавателей. Примеры реализации поиска по группам и преподавателям демонстрируют, как этот функционал помогает пользователям быстро получать доступ к необходимой информации.

Внедрение такого пользовательского интерфейса обеспечивает эффективную навигацию в веб-приложении для студентов и преподавателей. Разработанные решения способствуют более эффективному управлению учебным процессом, оперативному внесению изменений и улучшению взаимодействия между всеми участниками образовательного процесса.

2.4 Разработка серверной части

2.4.1 Создание проекта на Flask

Flask – компактный фреймворк для быстрой разработки веб-приложений. Он предоставляет минимально необходимую функциональность и не навязывает строгие правила относительно структуры и архитектуры приложения, как это делает Django. Flask универсален: он подходит как для создания сложных приложений и API, так и для разработки небольших проектов. Его главный плюс – гибкость и простота.

Основные преимущества Flask:

– минималистичность. Flask отличается небольшим размером, в нем есть все самое необходимое и нет ничего лишнего;

– гибкость. Фреймворк не диктует определенных правил, позволяя разработчику сохранить полный контроль над структурой приложения;

– простота в использовании. Flask имеет несколько встроенных функций, которые позволяют сразу начать создание полноценного веб-приложения, даже при отсутствии опыта в веб-разработке на языке программирования Python. Flask содержит встроенный сервер, поддержка сессий, обработка форм и шаблонизатор;

– интеграция с дополнительными библиотеками. Flask легко интегрируется с многочисленными библиотеками, расширяющими его функциональность. Это позволяет создать гибкий и масштабируемый проект для любой сферы;

– простота тестирования. Flask имеет встроенный тестовый клиент, который максимально упрощает процесс тестирования и отладки.

Устанавливать Flask лучше всего в виртуальное окружение – это позволяет избежать появления ошибок, связанных с конфликтами версий различных библиотек и модулей.

2.4.2 Разработка API

Определение API было раскрыто в книге Арно Лоре [1, с 31] «API – это только интерфейс, предоставляемый неким программным обеспечением. Это абстракция базовой реализации (базовый код – это то, что на самом деле происходит внутри программного продукта при использовании API).» API часто используется для обозначения всего программного продукта, включая API и его реализацию.

Для создания API приложения на Flask необходимо определить конечные точки (endpoints). Конечная точка (endpoint) – это URL, по которому клиентское приложение может обратиться к серверу для выполнения определенной операции. В каждой конечной точке описывается, какой HTTP-метод используется (GET, POST, PUT, DELETE и т. д.), и какая информация отправляется или запрашивается. Конечные точки позволяют структурировать и организовать взаимодействие с данными и функциональностью приложения. В веб-приложении «Расписание» определены следующие конечные точки:

– /api/getTeacher – получение информации о преподавателях;

– /api/teacher – операции с данными преподавателей;

– /api/building – операции с данными о зданиях;

– /api/getCorpusOptions – получение информации о корпусах;

– /api/getCourseOptions – получение информации о курсах;

– /api/classroom – операции с данными аудиторий;

– /api/getClassroom – получение информации о конкретной аудитории;

– /api/group – операции с данными групп;

– /api/subgroup – операции с данными подгрупп;

– /api/getGroupOptions – получение информации о группах;

– /api/getSubject – получение информации о предметах;

– /api/lesson_type – операции с типами занятий;

– /api/schedule – операции с расписанием;

– /api/getTodaySchedule – получение расписания на сегодня;

– /api/subject – операции с данными о предметах;

– /api/teachers_subjects – операции с данными о предметах преподавателей;

– /api/Intervals – получение интервалов времени;

– /api/users – операции с данными пользователей;

– /api/day_of_week – операции с данными о днях недели;

– /api/group_subjects – операции с данными о предметах групп.

Все описанные конечные точки являются основой для взаимодействия клиентской части веб-приложения с серверной частью, позволяя эффективно управлять и обрабатывать данные. Каждая конечная точка определяет соответствующий URL и HTTP-метод, используемый для доступа к функционалу API.

2.4.3 Интеграция с базой данных

Интеграция с базой данных в веб-приложении «Расписание» включает создание и использование нескольких таблиц для хранения данных. Каждая таблица представляет собой структурированное хранилище информации, необходимой для функционирования системы. Основные таблицы, используемые в разработанном приложении:

1) Buildings: хранит информацию о зданиях, включая их название и адрес.

2) ChangeNotifications: связывает уведомления об изменениях с соответствующими изменениями в системе.

3) Classrooms: содержит данные об аудиториях, включая номера аудиторий и связи с соответствующими зданиями.

4) days_of_week: представляет дни недели для использования в расписании.

5) FieldMappings: используется для сопоставления полей на разных языках, что может быть полезно для интернационализации системы.

6) Group_Subjects: связывает учебные группы с учебными предметами, которые они изучают.

7) Groups_table: хранит информацию о учебных группах, включая их название, курс и связь с соответствующим зданием.

8) lesson_type: содержит различные типы занятий, такие как лекции, семинары и т. д.

9) Notifications: сохраняет уведомления для пользователей, например, о изменениях в расписании.

10) role и roles_users: используются для управления ролями пользователей в системе.

11) Schedule и ScheduleChanges: хранят информацию о расписании занятий и его изменениях.

12) SubGroup: содержит информацию о подгруппах учебных групп.

13) Subjects: содержит данные об учебных предметах.

14) Teachers и Teachers_Subjects: связывают преподавателей с учебными предметами, которые они ведут.

15) time_intervals: хранит интервалы времени, используемые в расписании занятий.

Для каждой из таблиц были созданы классы на языке программирования Python с соответствующими полями. Данные классы позволяют работать с данными непосредственно в приложении и призваны структурировать работу с информацией и СУБД MySQL.

Разберем класс Schedule на основе языка программирования Python с использованием SQLAlchemy для взаимодействия с созданной базой данных. Давайте рассмотрим основные аспекты этого класса.

Класс Schedule является моделью данных, которая отражает структуру таблицы Schedule в базе данных и содержит следующие поля, которые соответствуют полям в базе данных:

– id, который является первичным ключом и уникальным идентификатором каждой записи расписания;

– date представляющее собой дату;

– start_time и end_time, определяющие время начала и окончания занятия;

– lesson_type_id, teachers_subjects_id, classroom_id, group_id, subgroup_id, которые являются внешними ключами, связывающими запись в таблице Schedule с соответствующими данными в других таблицах, таких как тип занятия, преподаватели и предметы, аудитории, группы и подгруппы.

В конструкторе __init__ класса Schedule проводится инициализация этих полей для создания новой записи в расписании или обновления существующей. Также был определен метод serialize, который преобразует объект Schedule в словарь Python, чтобы упростить передачу этих данных через API и их последующую обработку.

Такой подход позволяет управлять данными расписания в учебном заведении, обеспечивая быстрый доступ к актуальной информации и минимизацию ошибок при составлении расписания. А также позволяет работать с информацией в удобном для пользователей виде.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы по теме «Проектирование и разработка веб-приложения «Расписание» для КГБПОУ Лесосибирский технологический техникум» была достигнута основная цель исследования – проектирование и разработка веб-приложения для управления расписанием занятий. Данная система была спроектирована и реализована с использованием современных технологий и инструментов, таких как Python, Flask, React и JavaScript.

Были выполнены выполнены все поставленные задачи:

− проанализирована структура учебного заведения и текущие методы управления расписанием; определены ключевые потребности пользователей, что позволило сформулировать цели автоматизации и требования к веб-приложению; разработаны модели системы;

− разработана архитектура веб-приложения с учетом принципов чистой архитектуры, обеспечивающей масштабируемость и отказоустойчивость; определены методы взаимодействия компонентов и меры обеспечения безопасности; определены инструментальные средства разработки;

− создана серверная часть (на основе Flask) и клиентская часть (с помощью React разработан пользовательский интерфейс как для панели администратора, так и для общедоступных страниц), реализована обработка HTTP запросов для взаимодействия клиентской и серверной частей веб-приложения, обеспечена интеграция с базой данных MySQL с использованием SQLAlchemy для управления данными расписания;

− данная система внедрена в образовательный процесс КГБПОУ «Лесосибирский технологический техникум».

Все поставленные задачи были выполнены в полном объеме. Проектирование и разработка системы выполнены с учетом современных требований к веб-приложениям, что позволило создать эффективный инструмент для управления

расписанием в учебном заведении. Внедренная система облегчает процесс создания и поддержания расписания, снижает вероятность ошибок и обеспечивает быстрый доступ к актуальной информации для всех заинтересованных лиц.

Разработанная система «Расписание» для КГПОУ Лесосибирский технологический техникум является значительным вкладом в автоматизацию и управление учебным процессом, обеспечивая современный и удобный способ организации расписания занятий.

Результаты исследования были представлены на следующих научных мероприятиях:

1. II Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 10–15 апреля 2023 г., участие).

2. VII Всероссийская научно-практической конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла» (г. Лесосибирск, ЛПИ – филиал СФУ, 1–2 ноября 2023 г., диплом I степени).

3. Конкурс проектов «Точка роста» в рамках VIII Всероссийского (IX Регионального) молодежного форума «Российское могущество прирастать будет Сибирью…» (г. Лесосибирск, ЛПИ – филиал СФУ, 14 декабря 2023 г., участие)

4. III Всероссийский молодежный научный форум «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г., участие).

5. Конкурс проектов «Точка роста» научно-исследовательской направленности молодежного научно-образовательного фестиваля «Ступени» в рамках III Всероссийского молодежного научного форума «Современное педагогическое образование: теоретический и прикладной аспекты» (г. Лесосибирск, ЛПИ – филиал СФУ, 9 апреля 2024 г., участие).

6. XX Международная научная конференция студентов, аспирантов и молодых ученых «Проспект Свободный – 2024» (г. Красноярск, СФУ, 15–20 апреля 2024 г., участие).

По результатам исследования принята к публикации статья: Копейкин А. А. / Архитектура веб-приложения для управления расписанием: ключевые аспекты. / А. А. Копейкин // Цифровые технологии в образовании, науке и технике: материалы XX Международная научная конференция студентов, аспирантов и молодых ученых «Проспект Свободный – 2024», г. Красноярск, / Сибирский федеральный университет. – Красноярск, 2024.

В дальнейшем разработанная система может быть улучшена по средствам добавления методов обработки, анализа расписания и автоматического формирования расписания на основе уже имеющихся данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Арно Лоре Проектирование веб-API / Пер. с англ. Д. А. Беликова. – Москва : ДМК Пресс, 2020 – 440 с. – ISBN 978-5-97060-861-6

2. Берг, Д. Б. Модели жизненного цикла: учебное пособие / Д. Б. Берг, Е. А. Ульянова, П. В. Добряк. – Оренбург : Уральский университет, 2014. – 74 с. – ISBN 978-5-7996-1311-2.

3. Вигерс, К. Разработка требований к программному обеспечению. / К. Вигерс, Д. Битти. – 3-е изд., дополненное – Санкт-Петербург : БХВ-Петербург, 2014. – 736 с. – ISBN 978-5-9775-3348-5

4. ГОСТ 27.002-2015. Надежность в технике. Основные понятия. Термины и определения = Dependability in technics. Terms and definitions : межгосударственный стандарт : Издание официальное : принят Межгосударственным советом по стандартизации, метрологии и сертификации (протокол от 28 декабря 2015 г. N 83-П) : взамен ГОСТ 27.002-89 : дата введения 2017-03-01 / разработан Обществом с ограниченной ответственностью «Институт надежности машин и технологий» (ООО «ИНМиТ») – Москва : Стандартинформ, 2016. – IV, 29с.

5. ГОСТ 34.601-90 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания: : национальный стандарт Российской Федерации : Издание официальное : утвержден и введен в действие Постановлением Государственного комитета СССР по управлению качеством продукции и стандартам от 29.12.90 N 3469: введен взамен ГОСТ 24.601-86, ГОСТ 24.602-86 : дата введения 1992-01-01 / РАЗРАБОТАН И ВНЕСЕН Государственным комитетом СССР по управлению качеством продукции и стандартам – Москва : Стандартинформ, 2016. – IV, 19с.

6. ГОСТ 56939. Защита информации. Разработка безопасного программного обеспечения. Общие требования = Information protection. Secure software development. General requirements : национальный стандарт Российской Федерации : Издание официальное : утвержден и введен в действие приказом Федерального

агентства по техническому регулированию и метрологии от 1 июня 2016 г. №458-ст : введен впервые : дата введения 2017-06-01 / разработан Закрытым акционерным обществом «Научно-производственное объединение «Эшелон» (ЗАО «НПО «Эшелон»). – Москва : Стандартинформ, 2016. – IV, 19с.

7. ГОСТ Р 34.11-2012. Информационная технология. Криптографическая защита информации. Функция хэширования = Information technology. Cryptographic data security. Hash-function : национальный стандарт Российской Федерации : издание официальное : утвержден и введен в действие приказом Федерального агентства по техническому регулированию и метрологии от 7 августа 2012 г. № 216-ст : введен впервые : дата введения 2013-01-01 / разработан Центром защиты информации и специальной связи ФСБ России с участием Открытого акционерного общества «Информационные технологии и коммуникационные системы»(ОАО «ИнфоТеКС»). – Москва : Стандартинформ, 2013. – IV, 19с.

8. Документация GIT // GIT : официальный сайт. – 2024. – URL: https://git-scm.com/book/ru/v2 (дата обращения: 22.04.2024)

9. Документация GitHub Flow // GitHub Flow : официальный сайт. – 2024. –URL https://githubflow.github.io/ (дата обращения: 11.05.2024)

10. Документация MySQL Workbench // MySQL Workbench : официальный сайт. – 2024. –URL: https://dev.mysql.com/doc/workbench/en/ (дата обращения: 10.04.2024).

11. Дубаков А. А. Сетевое программирование: учебное пособие / А. А. Дубаков – Санкт-Петербург : НИУ ИТМО, 2013 – 248 с.

12. Исаев, Г. Н. Проектирование информационных систем. Учебное пособие / Г. Н. Исаев. – Москва : Омега-Л, 2015. – 424 с. – ISBN 978-5-370-02508-2.

13. Калиберда, Е. А., Федотова, И. В. Анализ требований к программным продуктам с выбором метода тестирования на примере web-ориентированных приложений / Е. А. Калиберда, И. В Федотова // ОНВ. 2012. №2 (110). URL: https://cyberleninka.ru/article/n/analiz-trebovaniy-k-programmnym-produktam-s-

vyborom-metoda-testirovaniya-na-primere-web-orientirovannyh-prilozheniy (дата обращения: 20.05.2024)..

14. Кент, Б. Экстремальное программирование. Разработка через тестирование / Б. Кент. – Санкт-Петербург : Питер, 2022. – 224 с. – ISBN 978-5-4461-1439-9.

15. Кириченко, А. В. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков / А. В. Кириченко, А. П. Никольский, Е. В. Дубовик. – Санкт-Петербург : Питер, 2021. – 432 с. – ISBN 978-5-94387-271-6.

16. Ломов, А. Ю. HTML, CSS, скрипты: практика создания сайтов: самоучитель / А. Ю. Ломов. – Санкт-Петербург : БХВ-Петербург, 2007. – 399 с.– ISBN 5-94157-698-6.

17.Майерс, Г. Искусство тестирования программ/ М. Майерс, Т. Баджетт., К. Сандлер. – Москва : Диалектика, 2012 – 272 с. – ISBN 978-5-8459-1796-6.

18. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. / Р. Мартин Пер. с англ. – Санкт-Петербург : Питер, 2021. 352 с. – ISBN 978-5-4461-0772-8.

19. Мартин Р. Чистый код. Создание, анализ и рефакторинг. / Р. Мартин Пер. с англ. – Санкт-Петербург : Питер, 2013. 464 с. - ISBN 978-5-496-00487-9.

20. Мацяшек Л. А. Анализ и проектирование информационных систем с помощью UML 2.0, 3-е изд. – Москва : Вильямс, 2008. — 816 с. –ISBN 978-5-8459-1430-9.

21. Мюллер, Р.Д. Проектирование баз данных и UML / Р. Д. Мюллер. – Москва : ЛОРИ, 2002. – 420 с. - ISBN 5-85582-168-4.

22. Назаров, С. В. Архитектура и проектирование программных систем / С. В. Назаров. – Москва : ИНФРА-М, 2013. – 413 с. – ISBN 978-5-16-011753-9.

23. Никитин, И. В. Сравнение подходов монолитной архитектуры и микросервисной архитектуры при реализации серверной части веб-приложения / И. В. Никитин, Т. Ю. Гриценко // Дневник науки. – 2020.– №3. – с. 3.

24. Общие сведения о React // Microsoft : официальный сайт – 2024. – URL: https://learn.microsoft.com/ru-ru/windows/dev-environment/javascript/react-overview

25. Остроух, А. В. Проектирование информационных систем / Н. Е. Суркова, А. В. Остроух. – Санкт-Петербург : Издательство «Лань», 2021. – 164 с. – ISBN 978-5-8114-8377-8.

26. Прохоренок, Н. А. Bootstrap и CSS-препроцессор Sass. Самое необходимое / Н. А. Прохоренок. – Санкт-Петербург : БХВ, 2021. – 496 с. – ISBN 978-5-9775-6769-5.

27.Попова-Коварцева, Д. А. Основы проектирования баз данных : учеб. пособие / Д. А. Попова-Коварцева, Е. В. Сопченко. – Самара : Самарский университет, 2019. – 112 с.: ил. ISBN 978-5-7883-1450-1

28. Р 50.1.111-2016. Информационная технология. Криптографическая защита информации. Парольная защита ключевой информации = Information technology. Cryptographic data security Password-based protection of key information : рекомендации по стандартизации : издание официальное : утвержден и введен в действие Приказом Федерального агентства по техническому регулированию и метрологии от 23 ноября 2016 г. № 1752-ст : введен впервые : Дата введения 2017-06-01 / разработан подкомитетом 1 Технического комитета по стандартизации ТК 26 «Криптографическая защита информации». – Москва : Стандартинформ, 2016. – IV, 7с.

29. Современный учебник JavaScript. // JavaScript : официальный сайт. – 2024. – URL: https://learn.javascript.ru/(дата обращения: 28.05.2024).

30. Стружкин, Н. П. Базы данных: проектирование: Учебник для академического бакалавриата / Н. П. Стружкин, В. В. Годин. – Москва : Юрайт, 2024. – 477 с. – ISBN 978-5-534-00229-4.

31. Тамре, Л. Введение в тестирование программного обеспечения. Руководство / Л. Тамре. – Москва : /Вильямс, 2003. – 359 с. – ISBN 5-8459-0394-7 .

32. Федеральные законы и акты Правительства Федеральный Закон: «О персональных данных»: [Принят Государственной Думой 8 июля 2006 года: Одобрен Советом Федерации 14 июля 2006 года] (дата обращения: 28.05.2024).

33. Федеральные законы и акты Правительства Федеральный Закон: «Об информации, информационных технологиях и о защите информации»: [Принят Государственной Думой 8 июля 2006 года: Одобрен Советом Федерации 11 июля 2006 года]

34. Фрейн, Б. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Б. Фрейн. – Санкт-Петербург : Питер, 2018. – 304 с.– ISBN 978-5-496-00185-4.

35. Чаллавала, Ш. MySQL8 для больших данных / Ш. Чавалла, Д. Лакхатария, Ч. Мехта. – Москва : «ДМК Пресс», 2017. – 228 с. – ISBN 978-5-97060-653-7. 80

36. Чистов, Д. В. Проектирование информационных систем: учебник и практикум для СПО / Д. В. Чистов – Москва: Юрайт, 2019. – 258 с. – ISBN 978-5-534-03173-7.

37. Flask // документация : официальный сайт. – 2024. – URL: https://flask.palletsprojects.com/en/3.0.x/ (дата обращения: 28.05.2024).

38.Python // документация : официальный сайт. – 2024. – URL: https://www.python.org/ (дата обращения: 28.05.2024).

39.React // документация : официальный сайт. – 2024. – URL: https://react.dev/ (дата обращения: 28.05.2024).

40.Visual Studio Code // документация : официальный сайт. – 2024. – URL: https://code.visualstudio.com/docs (дата обращения: 28.05.2024).

ПРИЛОЖЕНИЕ А

Листинг точки входа клиентской части веб-приложения

```
import React, { useEffect, useState } from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import { Navigate } from 'react-router-dom';
import 'bootstrap/dist/css/bootstrap.min.css';

import {
  apiUrl,
  privateRoutes,
  adminRoutes,
  publicRoutes,
} from './config/config';

import CustomNavbar from './components/Navbar';
import WarningSchedule from './components/warning';
import PageContainer from './components/PageContainer';
import Footer from './components/Footer';
import GroupSelectionForm from './components/GroupSelectionForm';
import TeacherSchedule from './components/TeacherSchedule';
import PrivateRoute from './PrivateRoute';

import Shedule from './components/Shedule';

const App = () => {
  const [tokenChecked, setTokenChecked] = useState(false);
  const [isAuthenticated, setIsAuthenticated] = useState(false);

  useEffect(() => {
    const checkToken = async () => {
      try {
        const response = await fetch(`${apiUrl}/check_token`, {
          method: 'POST',
          headers: {
            Authorization: `Bearer ${localStorage.getItem('access_token')}`,
          },
        });

        if (response.ok) {
          setIsAuthenticated(true);
        }
```

```
      setTokenChecked(true);
    } catch (error) {
      console.error('Error checking token:', error);
      setTokenChecked(true);
    }
  };

  checkToken();
}, []);

return (
  <Router>
    <div>
      <WarningSchedule />
      <CustomNavbar />
      <PageContainer>
        <Routes>
          {publicRoutes.map((route) => (
            <Route key={route.path} path={route.path} element={route.element} />
          ))}
          <Route path="/" element={<GroupSelectionForm />} />
          <Route path="/teacher/:te_id" element={<TeacherSchedule />} />
          {isAuthenticated && (
            <Route path="/admin" element={<Navigate to="/admin/dashboard" />} />
          )}
          <Route
            path="/admin"
            element={<PrivateRoute isAuthenticated={isAuthenticated} />}
          >
            {adminRoutes.map((route) => (
              <Route key={route.path} path={route.path} element={route.element} />
            ))}
          </Route>

          <Route path="/shedule/:groupId" element={<Shedule />} />
        </Routes>
      </PageContainer>
      <Footer />
    </div>
    {!tokenChecked && <div>Loading...</div>}
  </Router>
```

```
  );
};

export default App;
```

ПРИЛОЖЕНИЕ Б

Листинг всех маршрутов веб-приложения

```
// Import your components
import LoginForm from '../admin/dashboard/LoginForm';
import UserList from '../admin/dashboard/UserList';
import RegistrationForm from '../admin/dashboard/RegistrationForm';
import AdminDashboard from '../admin/dashboard/AdminDashboard';
import Teacher from '../admin/dashboard/teacher';
import Groups from '../admin/dashboard/groups';
import DayOfWeekForm from '../admin/components/DayOfWeekForm';
import TimeIntervalForm from '../admin/components/TimeIntervalForm';
import Bulding from '../admin/dashboard/building';
import LessonType from '../admin/dashboard/lessonType';
import Subjects from '../admin/dashboard/subjects';
import ScheduleCreationPage from '../admin/components/ScheduleCreationPage';
import Shedule from '../components/Shedule';
import GroupSelectionForm from '../components/GroupSelectionForm';
import TeacherSchedule from '../components/TeacherSchedule';
import GroupSubject from '../admin/dashboard/GroupSub';

// Define your routes
export const publicRoutes = [
  { path: '/login', element: <LoginForm /> },
  { path: '/', element: <GroupSelectionForm /> },
  { path: '/teacher/:te_id', element: <TeacherSchedule /> },
  { path: '/shedule/:groupId', element: <Shedule /> },
];
export const adminRoutes = [
    { path: 'users', element: <UserList /> },
    { path: 'register', element: <RegistrationForm /> },
    { path: '', element: <AdminDashboard /> },
    { path: 'teacher', element: <Teacher /> },
   // { path: 'groups', element: <Groups /> },
    { path: 'groups', element: <GroupSubject /> },
    { path: 'day-of-week', element: <DayOfWeekForm /> },
    { path: 'time-intervals', element: <TimeIntervalForm /> },
    { path: 'buildings', element: <Bulding /> },
    { path: 'lesson-types', element: <LessonType /> },
    { path: 'subjects', element: <Subjects /> },
    { path: 'schedule_table', element: <ScheduleCreationPage /> },
  ];
```

```
export const apiUrl = 'http://timetable.pu-48.ru/api';
export const apiSoket = 'http://timetable.pu-48.ru:5000/';
export const url = 'http://timetable.pu-48.ru/';
export const adminUrl = 'http://timetable.pu-48.ru/admin';
```

ПРИЛОЖЕНИЕ В

Листинг страницы расписания

```
// Schedule.js
import { useLocation } from 'react-router-dom';
import React, { useEffect, useState } from 'react';
import queryString from 'query-string';
import { useParams } from 'react-router-dom';
import { ListGroup, Container, Row, Col } from 'react-bootstrap';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom'; // Import
Routes instead of Switch
import ScheduleTabs from '../components/ScheduleTabs';
import TodaySchedule from '../components/TodaySchedule';
import TomorrowSchedule from '../components/TomorrowSchedule';
import WeeklySchedule from '../components/WeeklySchedule'; // Добавляем
компонент WeeklySchedule

import '../styles/CenteredContainer.css';
import { apiUrl } from '../config/config';
import SearchComponent from './SearchComponent';
 function Shedule() {
  const [queryParams, setQueryParams] = useState({});
  const [activeTab, setActiveTab] = useState('today');
  const { groupId } = useParams();
  console.log(groupId);
  const [groupInfo, setGroupInfo] = useState(null);

  useEffect(() => {
    const fetchData = async () => {
      try {
        const groupData = await fetch(`${apiUrl}/group/${groupId}`).then((response) =>
response.json());
        const buildingData = await
fetch(`${apiUrl}/building/${groupData.building_id}`).then((response) =>
response.json());

        setGroupInfo({
          buildingId: groupData.building_id,
          building_name: buildingData.building_name,
          courseId: groupData.course,
          courseName: groupData.course, // Replace with actual course name retrieval logic
          group_name: groupData.group_name,
```

```
      });
    } catch (error) {
      console.error('Error:', error.message);
      // Handle the error in a way that fits your application logic
    }
  };

  fetchData();
}, [groupId]);

if (!groupInfo) {
  // You might want to render a loading state here
  return <div>Loading...</div>;
}

const corpusId = groupInfo.buildingId;
const corpus = groupInfo.building_name;
const courseId = groupInfo.courseId;
const course = groupInfo.courseName;
const group = groupInfo.group_name;
const handleTabChange = (tab) => {
  setActiveTab(tab);
};
return (
  <Container>
    <SearchComponent></SearchComponent>
    <Row className="justify-content-center">
      <Col xs={12} sm={10} md={8} lg={6}>
        <div>
          <div>
            <h1>Расписание для группы {group}</h1>
            <p>
              Корпус: {corpus}, Курс: {course}
            </p>
            <ScheduleTabs
              activeTab={activeTab}
              onTabChange={handleTabChange}
              selectedCorpusId={corpusId} // Передаем selectedCorpusId в ScheduleTabs
            />
            <ListGroup>
              {activeTab === 'today' ? (
                <div className="container">
```

```
                <TodaySchedule groupId={groupId} />
              </div>
            ) : activeTab === 'tomorrow' && (
              <div className="container">
                <TomorrowSchedule groupId={groupId} />
              </div>
            )}
            {corpusId === '2' && (
              <div className="container">
                <div>
                  <p></p>
                  <h3>Расписание на неделю</h3>
                  <p></p>
                </div>
                <WeeklySchedule groupId={groupId} />
              </div>
            )}
          </ListGroup>
        </div>
      </div>
    </Col>
  </Row>
 </Container>
 );
};

export default Shedule;
```

ПРИЛОЖЕНИЕ Г

Листинг CRUD операций для работы с базой данных

```
from sqlalchemy import insert
from app import db
class QueryComponent:
    def apply(self, query):
        pass
class JoinComponent(QueryComponent):
    def __init__(self, table, condition, select_from=True):
        self.table = table
        self.condition = condition
        self.select_from = select_from

    def apply(self, query):

        return query.join(self.table, self.condition)

class SelectComponent(QueryComponent):
    def __init__(self, *columns):
        self.columns = columns

    def apply(self, query):
        return query.with_entities(*self.columns)

class FilterComponent(QueryComponent):
    def __init__(self, *conditions):
        self.conditions = conditions

    def apply(self, query):
        for condition in self.conditions:
            query = query.filter(condition)
        return query

class InsertComponent(QueryComponent):
    def __init__(self, model, data):
        self.model = model
        self.data = data

    def apply(self, query):
        ins = insert(self.model).values(**self.data)
        query.session.execute(ins)
```

```
        query.session.commit()
        return query

class UpdateComponent(QueryComponent):
    def __init__(self, model, item_id, data):
        self.model = model
        self.item_id = item_id
        self.data = data

    def apply(self, query):
        query.session.query(self.model).filter(self.model.id ==
self.item_id).update(self.data)
        query.session.commit()
        return query

class DeleteComponent(QueryComponent):
    def __init__(self, model, item_id):
        self.model = model
        self.item_id = item_id

    def apply(self, query):
        query.session.query(self.model).filter(self.model.id == self.item_id).delete()
        query.session.commit()
        return query
class OrderByComponent(QueryComponent):
    def __init__(self, *columns):
        self.columns = columns

    def apply(self, query):
        return query.order_by(*self.columns)

class GroupByComponent(QueryComponent):
    def __init__(self, *columns):
        self.columns = columns

    def apply(self, query):
        return query.group_by(*self.columns)
```

ПРИЛОЖЕНИЕ Д

Листинг конструктора запросов

```
from app import db
from .components import QueryComponent
class QueryBuilder:
    def __init__(self, model):
        self.model = model
        self.query = db.session.query(model)
        self.components = []

    def add_component(self, component):
        self.components.append(component)

    def execute(self):
        for component in self.components:
            self.query = component.apply(self.query)
        return self.query.all()

    def select_from(self, table):
        self.query = self.query.select_from(table)
```

ПРИЛОЖЕНИЕ Е

Листинг точки запуска серверной части веб-приложения

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
from flask_jwt_extended import JWTManager, jwt_required, create_access_token
from flask import Flask, request, jsonify
from routes import teacher_bp, building_vw_bp, building_bp, classroom_bp,
classroom_vw_bp, group_bp, lesson_type_bp
from routes import subject_bp, teachers_subjects_bp, schedule_bp, course_vw_bp,
group_vw_bp, schedule_vw_bp,teacher_vw_bp
from routes import time_interval_vw_bp, Users_bp, day_of_week_bp,
group_subjects_bp, subgroup_bp,group_subj_vw_bp
from flask_cors import CORS
from flask_socketio import SocketIO

def create_flask_app():
    application = Flask(__name__)
    CORS(application)
    socketio = SocketIO(application, cors_allowed_origins ='*')

    application.config['SQLALCHEMY_DATABASE_URI'] =
'mysql://login:pass@localhost/ Schedule?charset=utf8mb4'
    application.config['JWT_SECRET_KEY'] = 'your-secret-key'
    jwt = JWTManager(application)
    db = SQLAlchemy(application)
    # application.register_blueprint(admin, url_prefix='/admin')
    application.register_blueprint(teacher_vw_bp, url_prefix='/api')
    application.register_blueprint(teacher_bp, url_prefix='/api')
    application.register_blueprint(building_bp, url_prefix='/api')
    application.register_blueprint(building_vw_bp, url_prefix='/api')
    application.register_blueprint(course_vw_bp, url_prefix='/api')
    application.register_blueprint(classroom_bp, url_prefix='/api')
    application.register_blueprint(classroom_vw_bp, url_prefix='/api')
    application.register_blueprint(group_bp, url_prefix='/api')
    application.register_blueprint(subgroup_bp, url_prefix='/api')
    application.register_blueprint(group_vw_bp, url_prefix='/api')
    application.register_blueprint(group_subj_vw_bp, url_prefix='/api')
    application.register_blueprint(lesson_type_bp, url_prefix='/api')
    application.register_blueprint(schedule_bp, url_prefix='/api')
    application.register_blueprint(schedule_vw_bp, url_prefix='/api')
```

```
application.register_blueprint(subject_bp, url_prefix='/api')
application.register_blueprint(teachers_subjects_bp, url_prefix='/api')
application.register_blueprint(time_interval_vw_bp, url_prefix='/api')
application.register_blueprint(Users_bp, url_prefix='/api')
application.register_blueprint(day_of_week_bp, url_prefix='/api')
application.register_blueprint(group_subjects_bp, url_prefix='/api')
@application.route("/api/login", methods=['POST'])
def login():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')

    user = db.User.query.filter_by(username=username).first()

    if user and check_password_hash(user.password_hash, password):
        access_token = create_access_token(identity=username)
        return jsonify(access_token=access_token), 200
    else:
        return jsonify(message='Неверные учетные данные'), 401
```

ПРИЛОЖЕНИЕ Ж

Листинг маршрута регистрации пользователей

```
from flask import request, jsonify
from app import db, Users,application
from werkzeug.security import generate_password_hash
@application.route('/api/register', methods=['POST'])
def register():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')
    existing_user = Users.query.filter_by(username=username).first()
    if existing_user:
        return jsonify(message='Пользователь уже существует'), 409
    # Создаем нового пользователя и хешируем пароль
    new_user = Users(username=username)
    new_user.set_password(password)
    # Добавляем пользователя в сессию и сохраняем изменения в базе данных
    db.session.add(new_user)
    db.session.commit()
    return jsonify(message='Пользователь зарегистрирован'), 201
```

ПРИЛОЖЕНИЕ И

Листинг класса Schedule

```
from app import db
from sqlalchemy import ForeignKey
from datetime import time

class Schedule(db.Model):
    __tablename__ = "Schedule"

    id = db.Column(db.Integer, primary_key=True)
    date = db.Column(db.Date, nullable=False)
    day_of_week = db.Column(db.String(10), nullable=False)
    start_time = db.Column(db.Time, nullable=False)
    end_time = db.Column(db.Time, nullable=False)
    lesson_type_id = db.Column(db.Integer, ForeignKey('lesson_type.id'))
                    teachers_subjects_id = db.Column(db.Integer,
ForeignKey('Teachers_Subjects.teachers_subjects_id'))
    classroom_id = db.Column(db.Integer, ForeignKey('Classrooms.id'))
    group_id = db.Column(db.Integer, ForeignKey('Groups_table.id'))
    subgroup_id = db.Column(db.Integer, ForeignKey('SubGroup.id'))
    num_lesson = db.Column(db.Integer, nullable=True)

    def __init__(self, id, date, day_of_week,start_time, end_time, lesson_type_id,
teachers_subjects_id, classroom_id, group_id,subgroup_id=None,num_lesson=None):
        self.id = id
        self.date = date
        self.day_of_week = day_of_week
        self.start_time = start_time
        self.end_time = end_time
```

```
        self.lesson_type_id = lesson_type_id
        self.teachers_subjects_id = teachers_subjects_id
        self.classroom_id = classroom_id
        self.group_id = group_id
        self.subgroup_id = subgroup_id
        self.num_lesson = num_lesson

    def serialize(self):
        serialized_data = {
            'id': self.id,
            'date': self.date,
            'day_of_week': self.day_of_week,
            'start_time': self.start_time.strftime('%H:%M'),
            'end_time': self.end_time.strftime('%H:%M'),
            'lesson_type_id': self.lesson_type_id,
            'teachers_subjects_id': self.teachers_subjects_id,
            'classroom_id': self.classroom_id,
            'group_id': self.group_id,
            'subgroup_id': self.subgroup_id,
            'num_lesson': self.num_lesson
        }

        return serialized_data
```