МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине: «Фронтенд-разработка»
по профилю: Разработка программных продуктов и проектирование информационных систем направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Мусатов И. А.
Группа: ИКБО-11-23
Срок представления к защите: 29.05.2025
Руководитель: _____

Тема Приложение для создания интерактивных учебных материалов

Исходные данные: используемые технологии: HTML5, CSS3, JS TS, React, нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18, ГОСТ 7.32-2017.

Перечень вопросов, подлежащих разработке, и обязательного графического материала:
1. Провести анализ предметной области разрабатываемого веб-сайта. 2. Сформировать требования к разрабатываемому веб-сайту. 3. Сформировать концепцию веб-сайта. 4. Обосновать выбор технологий разработки веб-сайта. 5. Разработать дизайн интерфейса веб-сайта 6. Реализовать фронтенд веб-сайта. 7. Разработка теста для проверки функционала веб-сайта 8. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: _____ / Болбаков Р. Г. /, «14» 03 2025 г.
Задание на КР выдал: _____ / Болбаков Р. Г. /, «13» 03 2025 г.
Задание на КР получил: _____ / Мусатов И. А. /, «13» марта 2025 г.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

LMS	– Learning Management System (то же самое, что и СДО)
MOOC	– Massive Open Online Courses (открытые для широкой публики онлайн-курсы)
UI	– User Interface (пользовательский интерфейс)
GUI	– Graphical User Interface (графический пользовательский интерфейс)
UX	– User eXperience (методика ориентации на пользовательский опыт)
UML	– Unified Modeling Language (унифицированный язык моделирования)
SPA	– Single Page Application (одностраничное приложение)
CBA	– Component-Based Architecture (компонентная архитектура приложения)
MVVM	– Model-View-ViewModel (архитектурный шаблон модель-представление)
ПО	– Программное Обеспечение
БД	– База данных
СДО	– Система Дистанционного Обучения

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

HTML	– Hyper Text Markup Language (язык гипертекстовой разметки документов в веб-браузере)
CSS	– Cascading Style Sheets (каскадные таблицы стилей веб-документа)
TS	– TypeScript (типизированный JavaScript – язык для написания скриптов для веб-документов)
React	– Библиотека JavaScript для создания реактивного фронтенда веб-сайта
Use case	– Вариант использования (пользовательский сценарий использования системы)
Peer-review	– Рецензирование научных работ специалистами из той же области
Роутинг	– Маршрутизация страниц сайта
Фронтенд сайта	– Клиентская часть интернет-ресурса (выполняется в браузере)
Бэкенд сайта	– Серверная часть интернет-ресурса (размещена на удаленном сервере)

ВВЕДЕНИЕ

Наш мир непрерывно и неизбежно меняется. Постоянно возникают и развиваются новые технологии и методы. В той или иной мере эти изменения касаются всех сфер человеческой жизни. Не исключением стало и образование.

Изобретение книгопечатания и типографики позволило упростить создание учебных материалов и сделало их доступными. Появление и широкое распространение Интернета сделало также возможным мгновенный обмен такими материалами. Теперь, в эпоху развитых технологий, перед учителями и преподавателями стоит ключевая задача: использование всех доступных средств для создания качественных и эффективных образовательных материалов.

Как показывают педагогические исследования [1], интерактивность учебных материалов значительно повышает интерес и активность обучающихся, положительно влияя на усвоение материала. Для создания подобных интерактивных материалов необходимы специфические средства. Стандартные методы вроде текстовых документов и видеозаписей не предоставляют всю широту образовательных возможностей для студента. Поэтому разработка и внедрение инструментов для создания интерактивных учебных материалов является *актуальной* проблемой.

В соответствии с этой проблемой, *целью* данной курсовой работы является разработка инструмента для создания интерактивных учебных материалов. А конкретно, веб-платформы, специализирующейся на возможности создания и распространения таких материалов. Прототипов подобной платформы все еще не существует, в этом мы видим *новизну* проекта.

Для достижения поставленной цели необходимо решить следующие *задачи*:

- подробный анализ предметной области дидактики и цифровой трансформации образования;
- анализ и оценка конкурентных решений;
- формирование требований к продукту;
- создание концепции продукта;
- проектирование дизайна пользовательского интерфейса;
- программная разработка клиентской части веб-сайта;
- тестирование программного продукта.

Ожидаемым результатом работы является готовый программный продукт (веб-сайт), предоставляющий пользователям возможности по созданию и распространению интерактивных учебных материалов.

1 ТЕОРЕТИЧЕСКИЙ АНАЛИЗ

1.1 Исследование предметной области

Сфера образования в наши дни активно адаптируется к инновационным технологиям. Так, в нашей стране для этого даже существует отдельный термин – «Цифровая трансформация образования» – и соответствующие ему проекты.

Цель такой трансформации – использование возможностей новых технологий для повышения эффективности образования, а также адаптации учебной программы к потребностям меняющегося общества. Польза и последствия таких проектов являются местом для дискуссии. Тем не менее, нас интересует фундаментальная часть вопроса. Для понимания интересующей нас организации интерактивных учебных материалов обратимся к дидактике.

Дидактика – это наука, изучающая проблемы обучения. Она исследует закономерности усвоения знаний, а также определяет объем и структуру содержания образования. В контексте нашей работы эта наука полезна тем, что определяет основные принципы оптимального закрепления знаний:

- системность;
- наглядность;
- стимулирование самостоятельного мышления;
- обратная связь.

Исследование [2] приводит практическое применение принципов дидактики. В основном, это технологии смешанного обучения, где обучающийся может изучать материал в том числе самостоятельно, без помощи учителя.

Такая возможность обеспечивается полнотой учебных материалов, которые содержат как структурированную и подробную теорию, так и практические задания с тестированием и подсказками. Особое значение имеет инфографика – представление сложных данных в виде изображений.

Формализуя дидактические заключения, можем обозначить список обязательных для интерактивных учебных материалов компонентов:

- заголовки и параграфы;
- вопросы с вариантами ответов;
- интерактивные схемы и модели;
- мультимедийные элементы (видео, аудио);
- обратная связь и подсказки.

Для полного анализа предметной области стоит также рассмотреть форму предоставления учебных ресурсов. Вообще говоря, существующие решения можно разделить на две группы: академические и «игровые».

Академические решения представляют собой веб-платформы со структурированием образовательного курса. Это могут быть, например, системы дистанционного обучения (активно используются в университетах) или массовые открытые онлайн-курсы (Coursera, Stepik, Практикум). Подобные решения являются довольно фундаментальными в том плане, что нацелены на обучение целым дисциплинам или областям знаний с полноценным отслеживанием прогресса и аттестациями.

В случае же «игровых» или, вернее, неформальных решений упор делается на мультимедийность материала. В качестве примеров можно привести образовательные платформы Учи.ру (мультимедийные задания с автопроверкой для школьников), Duolingo (интерактивное изучение языков), Kahoot! (викторины в реальном времени). Такие решения ориентированы на

практическое изучение и проверку знаний конкретных тем или уроков. Это делает их менее системными, зато более пригодными для проработки важных тем.

Пожалуй, наиболее правильным было бы использовать возможности и преимущества обеих групп. Чтобы можно было из интерактивных уроков и материалов выстраивать полноценные курсы.

1.2 Анализ конкурентных решений

Существующие решения по созданию и предоставлению учебных ресурсов имеют конкретную специфику. Зачастую это либо образовательные платформы с закрытой структурой материалов и четким образовательным планом, либо сайты, поддерживающие создание интерактивных компонентов и викторин. Решений, которые позволяют как создавать свои собственные материалы, так и удобно встраивать их в систему обучения, как таковых, нет. Поэтому, строго говоря, конкурентов в нашей специфике предметной области не имеется.

Тем не менее, рассмотрим лидеров программных решений в областях, тесно смежных с нашей. Будем анализировать как преимущества решений, так и недостатки, извлекая из этого пользу для собственного продукта.

В образовательной среде значимое место занимают так называемые LMS (Learning Management Systems), или же СДО (Системы Дистанционного Обучения). Они специализируются на структурировании учебных курсов в образовательных организациях, предоставляя порталы с возможностью загрузки файлов, организации тестов, поддержки чатов и прочего.

Одним из самых популярных LMS в мире является платформа Moodle. Она позволяет образовательным организациям выстраивать курсы любой сложности – с модулями, ветвлениями и сроками сдачи. Также уделяется внимание обеспечению оценки и аналитики: поддерживается система

журналов, баллов и прогресса обучения. Это, безусловно, является преимуществом для процесса образования.

Ядро платформы разрабатывалось еще в начале 2000х. Поэтому пользовательский интерфейс (см. рис. 1) и структура взаимодействия с элементами оставляют желать лучшего.

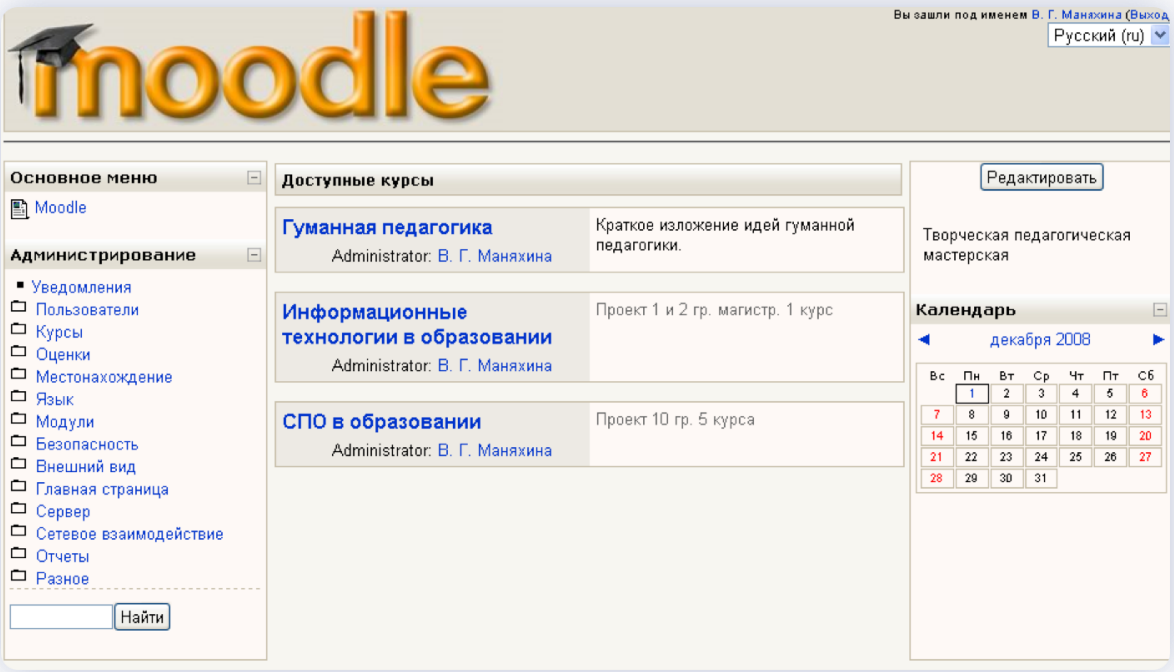

Рисунок 1 – Пользовательский интерфейс Moodle

Moodle «из коробки» имеет маленький выбор готовых интерактивных модулей. Для использования современного визуального подхода невозможно обойтись без сторонних плагинов.

Из основных недостатков системы можно выделить высокий уровень сложности для начинающих. Без изучения дополнительных инструкций преподавателю будет сложно разобраться с системой.

Перейдем к следующему типу образовательных ресурсов. В мировой практике они называются как MOOC (Massive Open Online Courses), в переводе – массовые открытые онлайн курсы. В отличие от LMS, MOOC не структурируют деятельность конкретной образовательной организации.

Зачастую это специализированные коммерческие или открытые курсы от университетов и компаний.

Примерами таких решений являются платформы Stepik и Яндекс.Практикум. Их предшественником и вдохновителем является платформа Coursera. Поэтому рассмотрим ее более подробно.

Coursera предоставляет возможность получения котирующихся дипломов и сертификатов, но вместе с тем большинство курсов являются платными.

Пользователь может строить свое обучение в удобном для себя формате и темпе. Поддерживается интерактивность в виде видеоматериалов, автопроверки и peer-review. Пользовательский интерфейс (см. рис. 2) довольно современен и понятен, что увеличивает удобство использования платформы.

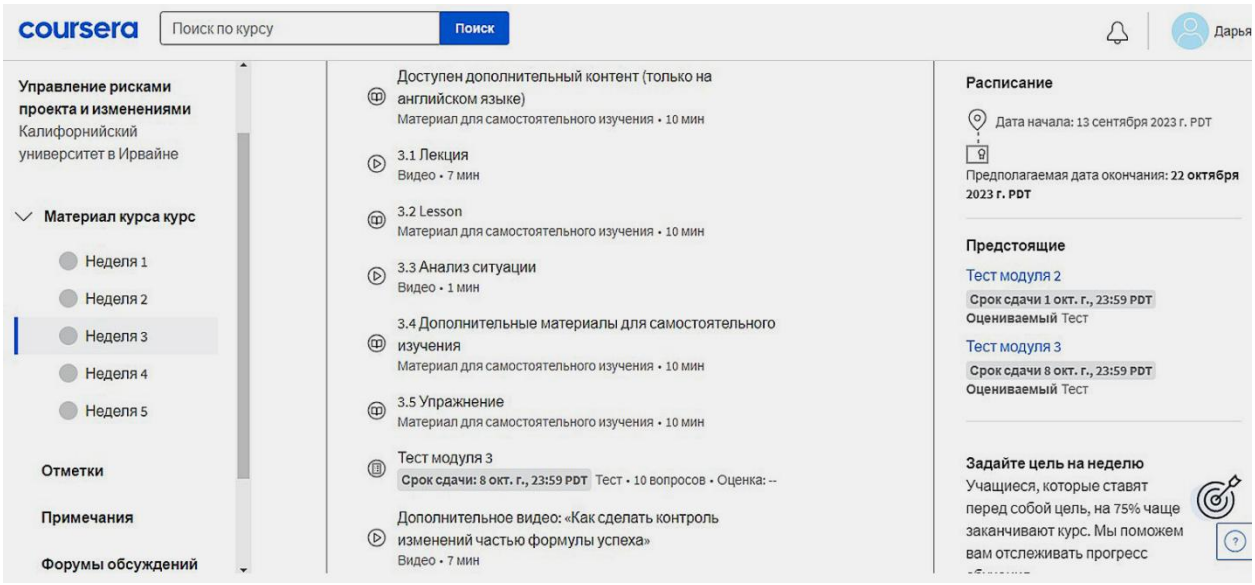

Рисунок 2 – Интерфейс образовательной среды в Coursera

Тем не менее, у платформы есть и значительные недостатки. Coursera имеет замкнутую экосистему. Создатели курсов не могу использовать платформу под свои задачи с привлечением внешней инфраструктуры. Также, что самое печальное в рамках понятия открытого образования, авторы

контента не могут самостоятельно публиковать курсы на платформе без заключения партнерства.

Здесь стоит отметить для себя ключевой момент. Если цель продукта – продвижение открытого и всеобщего образования, он, в первую очередь, должен соответствовать принципам открытого ПО и быть доступен для всех желающих создавать и размещать материалы.

Последней категорией из списка систем предоставления обучающего контента являются контент-редакторы. Ярким и наиболее разносторонним примером является платформа H5P.

H5P позволяет создавать более 40 различных типов интерактивного контента (видео, викторины, drag-and-drop, диаграммы). Визуальный редактор обеспечивает простоту создания и настройки компонентов, что обеспечивает легкость освоения данного инструмента (см. рис. 3).

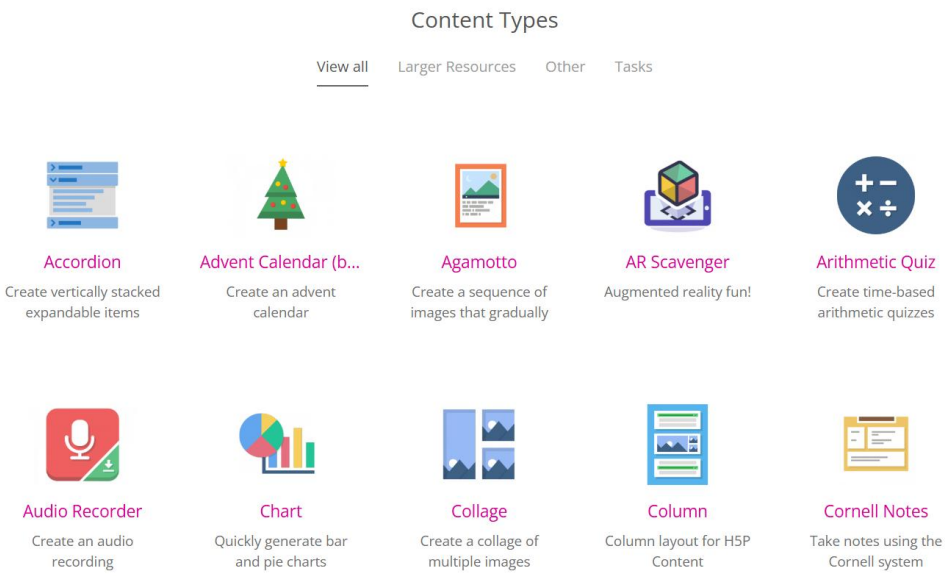

Рисунок 3 – Набор типов интерактивного контента в H5P

Основным преимуществом H5P является встраиваемость. Сформированные компоненты можно легко переносить и внедрять в другие платформы и проекты.

Однако простота и удобство системы оборачиваются другой проблемой: компоненты H5P имеют ограниченную логику поведения и интерфейс (см. рис. 4). Используя лишь мощности этого инструмента невозможно построить сложный сценарий обучения или хотя бы полноценный учебный материал.

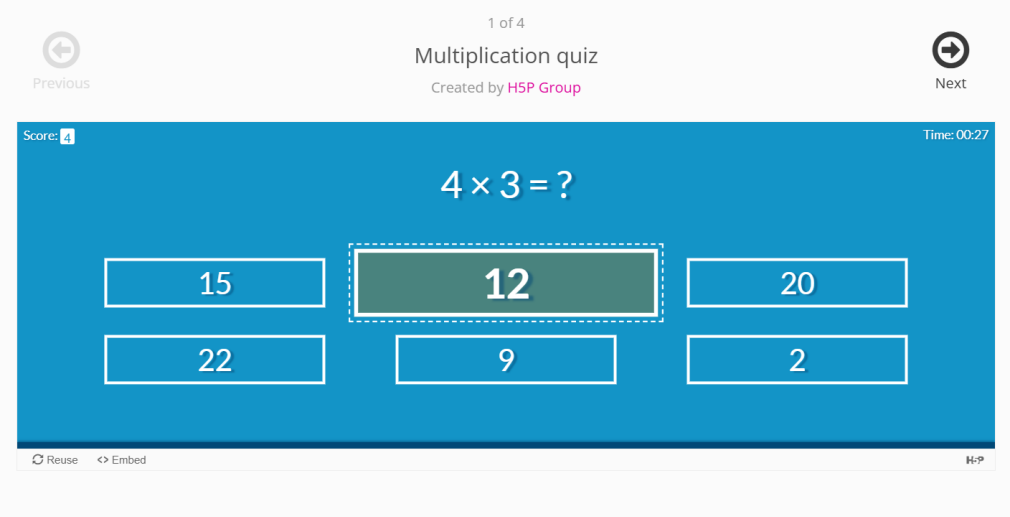

Рисунок 4 – Пользовательский интерфейс теста в H5P

Подводя итоги по контент-редакторам, можно отметить, что они не имеют четкой направленности на образовательный процесс. Это, скорее, интерактивный инструмент, который могут применять специалисты разных сфер.

В данном разделе также стоит отметить, что ни один из типов перечисленных ресурсов не предоставляет удобного формата для создания, оформления и распространения материалов. Они преподносят либо содержание без формы (стандартные текстовые документы или же видео, прикрепленные к курсу), либо форму без содержания (компоненты интерактивного контента).

Задача же нашего продукта – совмещение академического стиля изложения материала и технологий интерактивного контента. А поддержание оформления и распространения материалов – основа гибкости процесса образования.

1.3 Формирование требований к продукту

В соответствии с проведенным анализом конкурентов и специфики предметной области, а также с учетом инновационного подхода к созданию учебных материалов формируются следующие требования к программному продукту. Список функциональных требований и вариантов использования представлен в таблице 1.

Таблица 1 – Функциональные требования к продукту

Область требований	**Содержание требования**
Регистрация и авторизация	Возможность регистрации нового пользователя посредством ввода базовой информации (имя, email, пароль) в форму
	Поддержка авторизации через токены
	Возможность восстановления пароля
Личный кабинет (профиль)	Предоставление пользователю доступа к просмотру и редактированию его профиля
	Предоставление пользователю доступа к списку созданных им материалов
	Возможность удаления аккаунта
Создание и редактирование учебных материалов	Обеспечение GUI конструктора учебных материалов
	Поддержка drag&drop загрузки изображений
	Возможность предпросмотра учебного материала
Хранение и структуризация материалов	Обеспечение хранилища для набора учебных материалов
	Хранение материалов в структурной форме, содержащей метаинформацию и контент

Продолжение таблицы 1

Импорт и экспорт материалов	Поддержка собственного файлового расширения, обеспечивающего универсальность описания и представления учебных материалов
	Возможность экспорта материалов в собственном расширении
	Возможность импорта материалов в собственном расширении
Навигация по платформе	Поддержка роутинга страниц
	Отображение статуса авторизации
Адаптивность пользовательского интерфейса	Корректное отображение и поддержка функциональности на устройствах с различными форматами и размерами экранов

Помимо функциональных требований к программному продукту предъявляются также требования в смежных сферах (представлены в таблице 2).

Таблица 2 – Нефункциональные требования к продукту

Область требований	**Содержание требования**
Пользовательский интерфейс и дизайн	Неперегруженный, понятный дизайн в соответствии с принципами UI/UX
	Использование единой визуальной палитры
	Наличие логотипа продукта
Расширяемость	Возможность последующего расширения типов контента учебных материалов
	Открытость платформы и простота внедрения новой функциональности

Продолжение таблицы 2

Безопасность	Обеспечение защиты от XSS и CSRF уязвимостей
	Защищенное хранение пользовательских токенов
Браузерная совместимость	Корректная работа программного продукта в большинстве актуальных браузеров
Развертывание	Возможность дальнейшего развертывания платформы с использованием полноценной серверной части и базы данных

2 КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ

2.1 Концепция продукта

Разрабатываемый программный продукт представляет собой веб-платформу для создания, редактирования и распространения интерактивных учебных материалов.

Платформа предоставляет пользователю удобный графический интерфейс (GUI), позволяющий составлять образовательные материалы из структурированных блоков контента разного рода, в том числе и интерактивного. Предусмотрена возможность публикации и поиска материалов на базе платформы.

Особенностью проекта является возможность сохранения, а также экспорта и импорта интерактивных учебных материалов в уникальном формате (в виде отдельного файлового расширения), что обеспечивает переносимость и переиспользуемость созданного контента.

Проект ориентирован на гибкую поддержку педагогических сценариев и самостоятельное создание дидактически осмысленных материалов без необходимости знания программирования и документной разметки. Платформа реализует модульный подход, поддерживает предпросмотр, а также предлагает базовые средства авторизации и персонализации.

Целевой аудиторией программного продукта являются участники образовательного процесса, а именно две категории: преподаватели и обучающиеся. В роли преподавателей могут выступать школьные учителя, преподаватели специальных и высших учебных заведений, методисты (разработчики дидактических ресурсов), независимые авторы и репетиторы, аспиранты. В качестве обучающихся могут быть как школьники, студенты, аспиранты, так и любые желающие ознакомиться с различными интерактивными учебными материалами.

В случае расширения проекта и увеличения гибкости материалов, веб-платформа может также стать полезным подспорьем в организации работы технических специалистов: ученых, разработчиков, инженеров.

Для наглядного представления взаимодействий между пользователями и системой целесообразно использовать UML диаграмму Use Case (диаграмму вариантов использования). Она демонстрирует ключевые действия, которые могут совершать разные категории пользователей, и помогает уточнить функциональные границы программного продукта [3]. Диаграмма вариантов использования представлена на рис. 5.

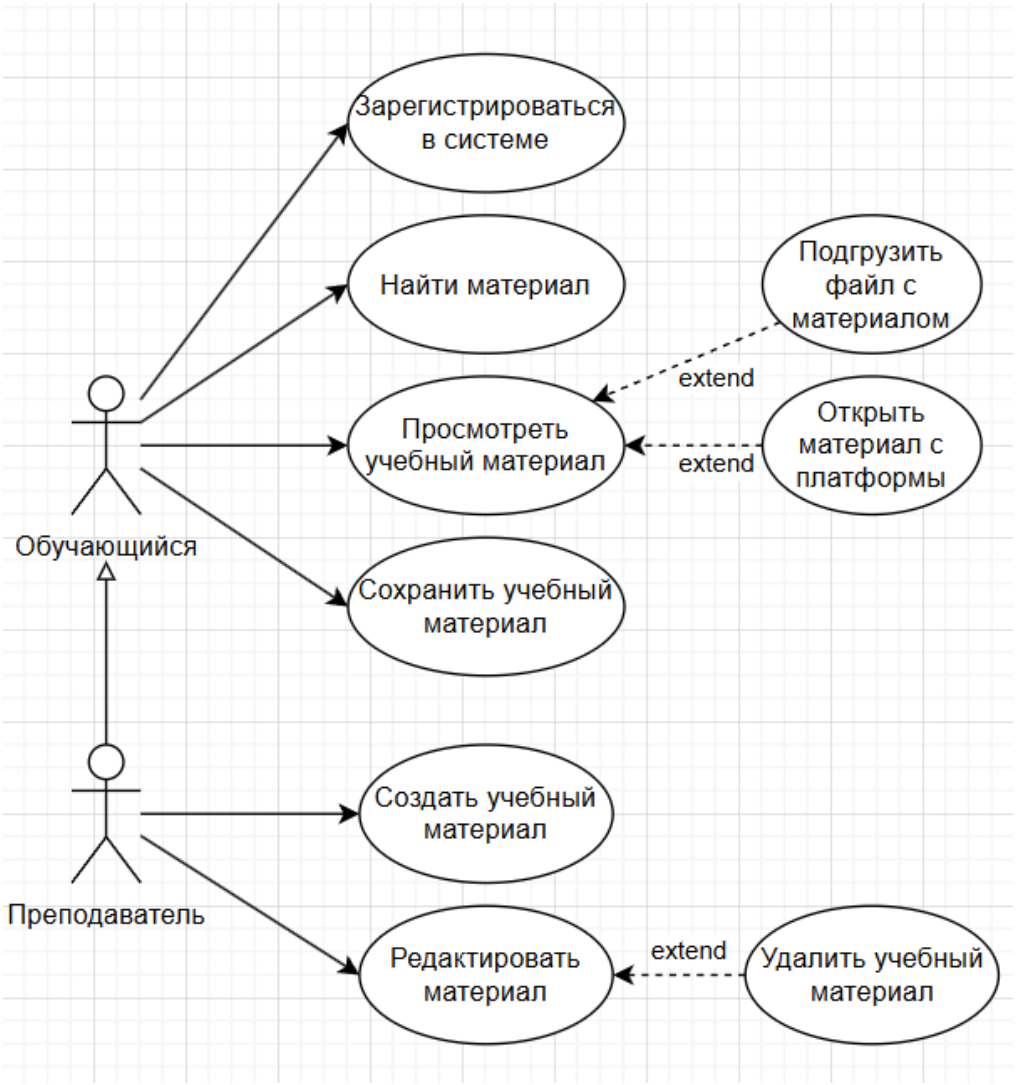

Рисунок 5 – Диаграмма вариантов использования веб-платформы

Рассмотрим подробнее схему сайта, а именно устройство его страниц. Веб-платформа содержит 9 страниц:

- страница регистрации – стандартная страница с подробной формой регистрации;
- страница входа – стандартная страница авторизации;
- страница профиля – личный кабинет пользователя системы с персональной информацией;
- главная страница – приветственная страница, знакомящая посетителя с функциональностью платформы;
- страница подгрузки материала – функциональная страница с возможностью просмотра подгруженного файла интерактивных материалов;
- страница поиска – раздел, обеспечивающий фильтрованный поиск опубликованных на платформе материалов;
- страница просмотра материала – страница с отрендеренным материалом, находящимся на платформе;
- страница редактирования материала – страница с GUI редактирования и создания интерактивных материалов;
- 404 not found – страница, обрабатывающая все неправильные и несуществующие переходы. Сообщает об отсутствии выбранного перехода.

Важным аспектом проектирования является описание переходов между страницами сайта. Формально это называется роутингом (маршрутизацией). Корректно организованная структура переходов помогает пользователю быстро и безошибочно следовать своему сценарию. Диаграмма переходов между страницами сайта представлена на рис. 6.

Рисунок 6 – Диаграмма состояний страниц сайта

В целях узнаваемости продукта и создания бренда было решено дать платформе название. С учетом специфики предметной области, а также дидактического подхода к созданию образовательных материалов было выбрано название «*Didacticum*» («*Дидактикум*»). Название подчеркивает академическую направленность и образовательную природу проекта. Латинизированная форма придает названию универсальный характер, легко воспринимаемый как в русском, так и международном контексте.

Расширение собственного формата файлов было решено назвать ILM (.ilm) – сокращение от Interactive Learning Material. Расшифровка однозначно определяет содержимое такого файла.

Стоит также отметить, что автор данного продукта придерживается принципов открытой архитектуры и свободного программного обеспечения. Это означает открытость платформы к дальнейшему расширению, участию сообщества разработчиков и встраиванию в другие платформы.

2.2 Обоснование технических решений

В качестве архитектурной основы проекта выбрана модель SPA (single page application – одностраничное приложение). Данная архитектура позволяет загружать приложение единожды, а затем динамически обновлять содержимое без полной перезагрузки страницы. Такой подход обеспечивает более высокую отзывчивость интерфейса и быструю навигацию между страницами. В условиях сложной структуры учебных материалов и конструктора блоков минимизация задержек особенно важна.

В рамках данной работы разрабатывается клиентская часть платформы, поэтому на текущем этапе, до создания полноценного бэкенда, для локального хранения данных (например, опубликованных материалов) будем использовать встроенное в браузер хранилище IndexedDB. Это низкоуровневое API для хранения больших объемов структурированных данных, включая бинарные объекты. В отличие от localStorage, IndexedDB предоставляет асинхронный интерфейс и более гибкую модель работы с индексами [4]. IndexedDB – это временное функционально полноценное решение, которое при масштабировании проекта легко сможет быть заменено настоящей базой данных в серверной части.

В качестве механизмов хранения токенов аутентификации используется HTTP cookies. Это решение совместимо с серверной аутентификацией, а также позволяет сохранять данные между сессиями пользователя. К тому же cookies обеспечивают базовую защиту от CSRF атак, что соответствует требованиям к продукту.

Как уже было отмечено в концепции продукта, наша платформа поддерживает собственный формат файлов ILM. Выбор собственного формата обусловлен необходимостью обеспечения универсальной работы со структурой материалов, а также возможности их экспорта и импорта. В целях простоты сериализации и десериализации в JS-объекты, собственный формат

ILM будет основываться на формате JSON (JavaScript Object Notation – объектная нотация языка JavaScript). К тому же данный формат является одним из наиболее легковесных.

2.3 Обоснование выбора инструментария

Разработка клиентской части интернет-ресурса начинается с выбора базовой технологии (платформы). Хоть сфера фронтенд-разработки и не может похвастаться разнообразием допустимых технологий (работа браузеров заточена под JS), небольшая вариативность, все же, присутствует. Поэтому рассмотрим и сравним некоторые базовые технологии фронтенда (см. таблицу 3).

Таблица 3 – Сравнительный анализ базовых технологий фронтенда

Критерий сравнения	**JavaScript**	**TypeScript**	**WebAssembly**
Типизация	Динамическая	Статическая	Статическая
Браузерная совместимость	Полная	Полная	Полная
Производительность	Средняя	Средняя	Очень высокая (особенно в вычислениях и обработке данных)
Пригодность для написания UI	Пригоден	Пригоден	Не пригоден
Экосистема	Широкая	Расширяет JS	Ограниченная
Удобство масштабирования	Низкое	Высокое	Высокое для вычислений

Характеристики из таблицы помогают разделить области применения JS, TS и WebAssembly. Если первые два языка традиционно используются для

написания фронтенда с упором на пользовательский интерфейс, то WebAssembly нацелен на низкоуровневые задачи оптимизации и вычисления.

Ввиду личного предпочтения автора к языкам со статической типизацией, а также убежденностью в том, что без системы типов невозможно строить сложные и масштабируемые системы, в качестве базового языка разработки выбран TypeScript.

В индустрии фронтенда устоялись часто используемые шаблоны разработки. Для их упрощения и автоматизации разработаны специальные средства. Наиболее распространенными являются библиотека React, а также фрейморки Angular и Vue.js. Проведем сравнительную характеристику этих инструментов (см. таблицу 4).

Таблица 4 – Сравнительный анализ инструментов разработки

Критерий сравнения	**React**	**Angular**	**Vue.js**
Архитектурный подход	CBA	MVVM	MVVM
Скорость разработки	Высокая	Средняя	Очень высокая
Производительность	Высокая	Высокая	Высокая
Уровень контроля	Высокий	Ограниченный	Средний
Экосистема	Большая (библиотеки)	Средняя (корпоративная)	Средняя (плагины)
Традиционная область применения	Кастомные SPA	Корпоративные приложения	Быстрое создание UI и прототипов

Анализ различий инструментов показывает, что для уровня средних задач (как в нашем случае – 9 страниц) наиболее подходящим и гибким инструментом становится библиотека React. Активное сообщество обеспечивает широкий набор библиотек и программных средств для решения типовых задач разработки. А относительная простота и компонентный подход снижают сложность погружения в инструмент.

Для обеспечения полноценного процесса разработки осталось определить только наиболее подходящий сборщик модулей и ассетов. Он позволяет объединить код проекта, ресурсы, зависимости и конфигурации в единую оптимизированную структуру, подходящую для использования в браузере.

Наиболее известные инструменты – Webpack, Vite, Rollup. Кратко рассмотрим их.

Webpack – мощный модульный бандлер с широкой поддержкой плагинов и конфигураций. Поддерживает lazy-loading, tree-shaking и другие оптимизационные методы. Предоставляет возможности полной и самостоятельной настройки проекта, что делает его с одной стороны гибким, а с другой – сложным.

Vite – современный сборщик на основе ES-модулей. Характеризуется быстрой скоростью dev-сборки, а также удобством настройки. Тем не менее, предоставляет меньшую гибкость и вариативность, чем Webpack.

Rollup – модульный бандлер, фокусирующийся на библиотеках. Предоставляет минимальные сборки, но не предназначен для SPA-приложений. Требует дополнительных доработок.

Мы остановим свой выбор на Webpack. Хоть инструмент и сложен, но он прозрачно показывает настройку «с нуля» и позволяет лучше понять устройство проекта.

Таким образом, стек нашего инструментария представляет собой формулу TypeScript + React + Webpack. Для реализации специфической функциональности также будут использоваться и дополнительные библиотеки.

2.4 Проектирование дизайна интерфейса

Первоочередной задачей дизайна продукта является определение цветовой палитры. Набор и сочетание цветов – это не только декоративный элемент, но и средство когнитивной организации информации. Исследование [5] показывает, что цвет играет ключевую роль в оформлении дизайна, а также определяет характер восприятия контента, особенно образовательного.

В академической традиции синий – это основной и традиционный цвет, ассоциирующийся со знаниями и сосредоточенностью. Многие известные университеты используют его в оформлении своих сайтов: МГУ, МФТИ Оксфорд, университет Северной Каролины, Йельский университет.

Контрастные к синему цвета (как светло-серый и голубой) позволят обеспечить выделение функциональных элементов и повысят читаемость материала. В результате подбора сочетаемости цветов была выбрана следующая «академическая» палитра (см. рис. 7).

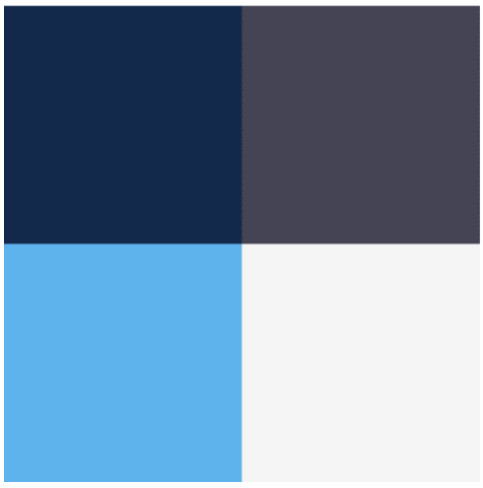

Рисунок 7 – Палитра дизайна веб-сайта

Перед тем как перейти к проектированию дизайна страниц веб-сайта, вернемся к формированию бренда нашей платформы. На этапе дизайна важным аспектом становится создание логотипа. Он позволяет выделять продукт среди других, а также обеспечивает его некоторую «философию».

Наша платформа называется «Didacticum» и нацелена на создание учебных материалов. Такое определение естественным образом приводит к ассоциациям с латинской буквой «D» и свитком – как одной из первых форм хранения знаний. Среди открытых источников были найдены прототипы данных объектов (см. рис. 8).

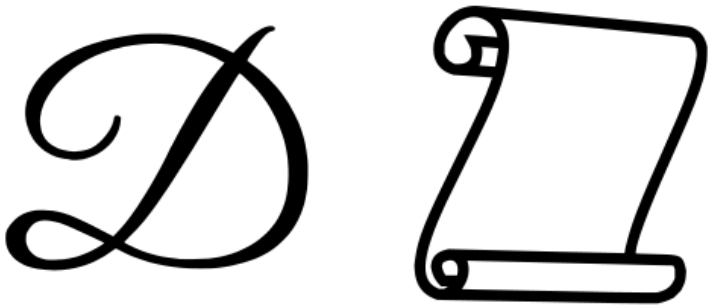

Рисунок 8 – Прототипы дизайна логотипа

Пристальный анализ и некоторый уровень абстракции позволяет выделить общие черты у объектов – плавные изгибы на концах. При попытке объединения этих объектов предпочтение стоит отдавать литере «D», т.к. именно она отсылает нас к названию. А свиток, скорее, контекстно дополняет ее.

Логотип – одна из важнейших частей бренда. Поэтому к его созданию стоит подойти с определенным профессионализмом. Автор уже имеет некоторый опыт работы с созданием векторной графики, поэтому было принято решение о самостоятельном формировании дизайна логотипа.

В качестве инструмента создания изображения был выбран векторный графический редактор Inkscape. Основным инструментом в процессе

проектирования стало «перо», позволившее делать плавные переходы и линии.

Дизайн логотипа (см. рис. 9) получился необычным. Завитки на краях даже создают образ ленты Мебиуса, что, конечно, довольно приятный бонус в контексте платформы учебных материалов. Дизайн сразу представлен в двух вариантах: стандартном и инверсивном.

Рисунок 9 – Дизайн логотипа

Перейдем теперь к проектированию дизайна веб-страниц платформы. Дизайн будем проектировать в удобной среде создания графического интерфейса Figma.

Начнем проектирование с описание размещения компонентов. Традиционно страницу делят на три компонента: «шапка» (header), основная часть (main) и «подвал» (footer). Так как наша платформа будет организована по принципу SPA, удобнее всего будет сделать «шапку» и «подвал» одинаковыми для всех страниц. Это обеспечит удобство доступа пользователей к ссылкам из «шапки» и справочной информации из «футера».

Первыми сделаем страницу входа и регистрации (рис. 10 и 11 соответственно). Разместим посередине страницы формы, которые будут запрашивать необходимую информацию. Использование замещенного текста

(placeholder) поможет пользователю понять формат вводимых данных. А яркие голубые кнопки и ссылки делают их заметными и намекают на их интерактивность.

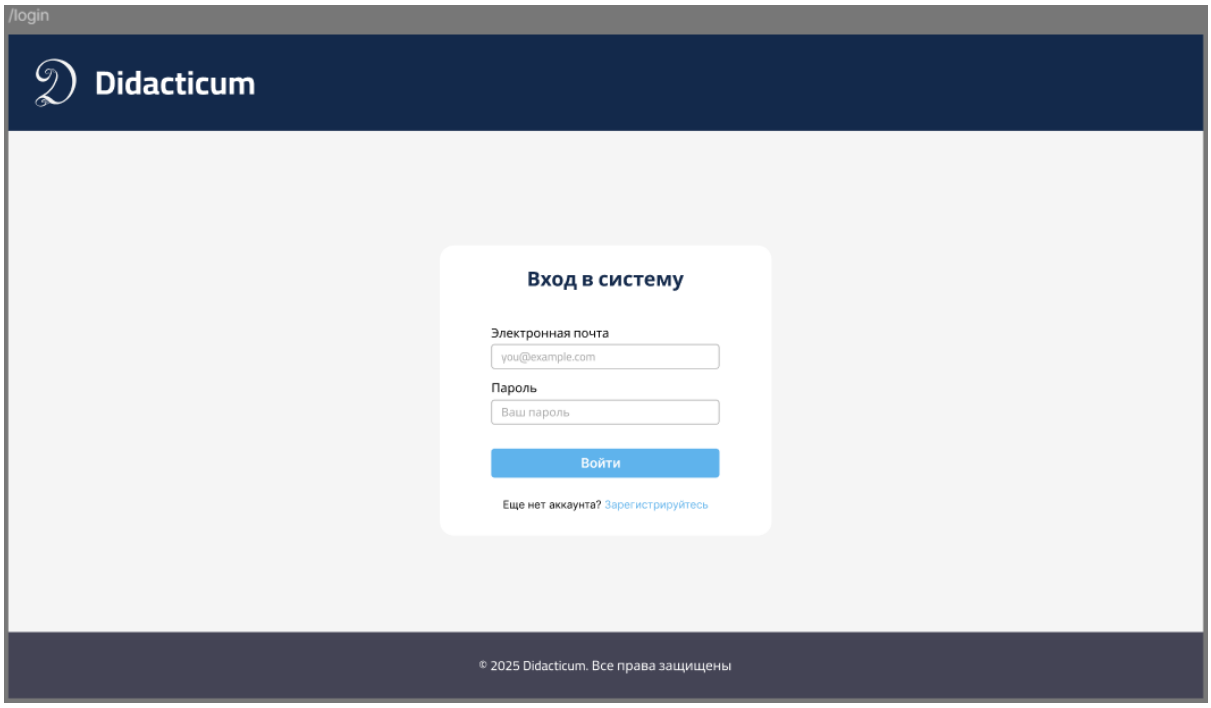

Рисунок 10 – Дизайн UI страницы входы

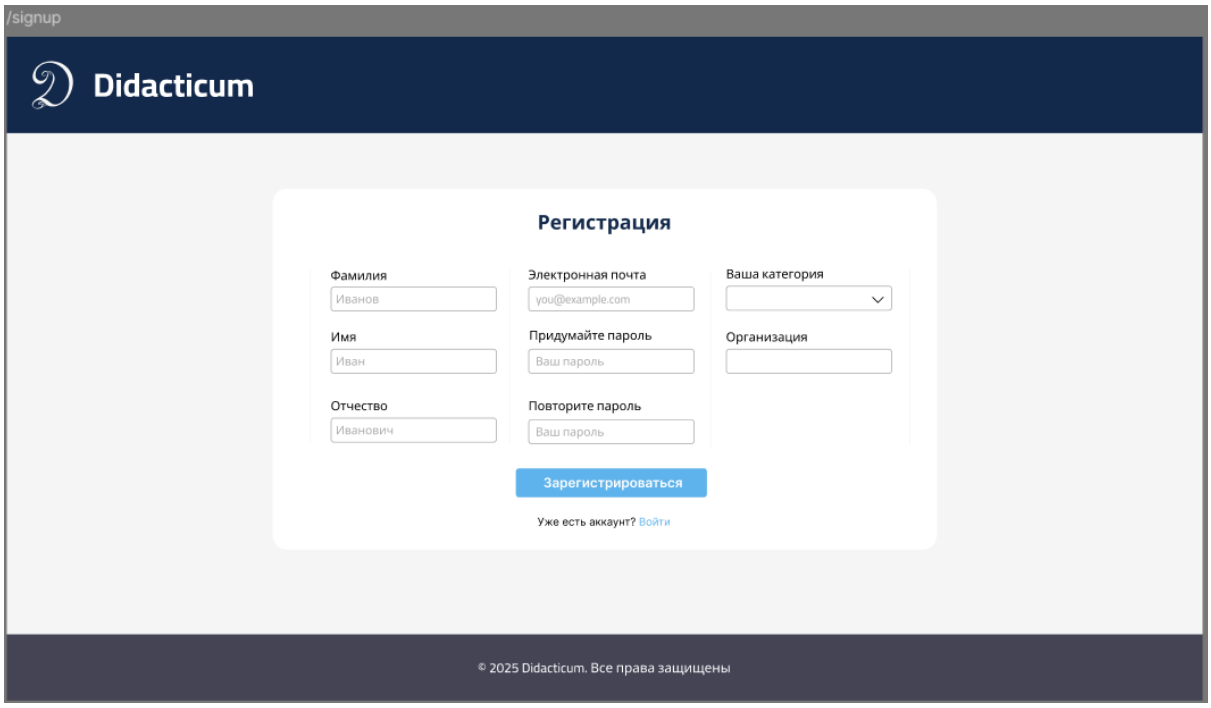

Рисунок 11 – Дизайн UI страницы регистрации

Основным компонентом в потоке опубликованных учебных материалов станет карточка учебных материалов. Она должна отражать название, автора и суть содержания учебного материала. Отобразим всю эту информацию при помощи игры цвета и деления карточки на заглавную часть, часть содержания и нижнюю часть (см. рис. 12).

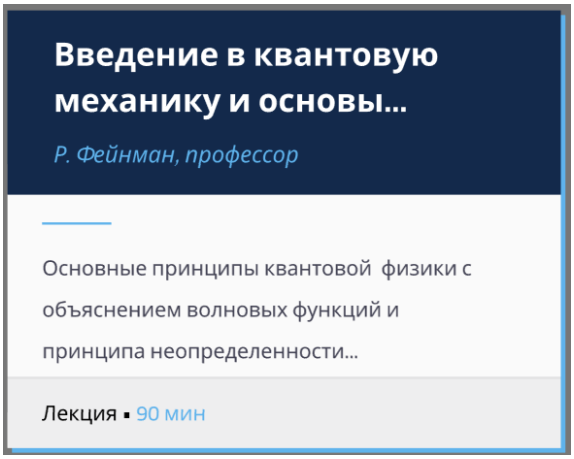

Рисунок 12 – Дизайн карточки учебного материала

Главная страница (см. рис. 13) будет встречать пользователя приветствием и кратким описанием возможностей платформы. Под приветствием располагается строка поиска, при активации которой пользователь попадет на страницу поиска.

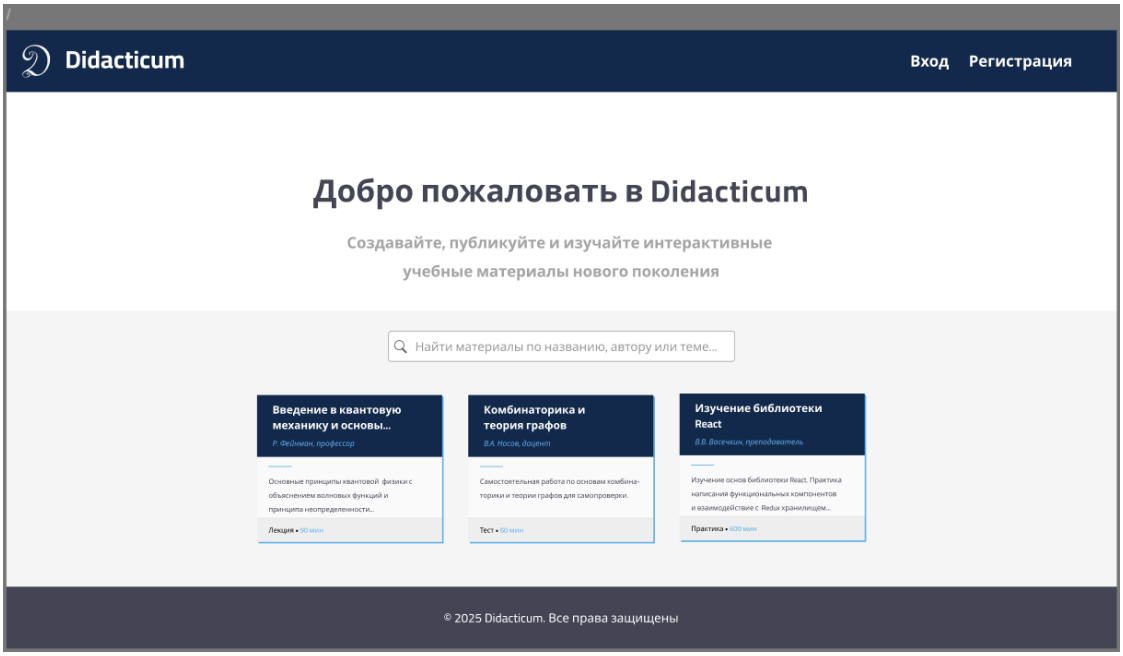

Рисунок 13 – Дизайн UI главной страницы

Личный кабинет (профиль) пользователя для разных категорий пользователей будет содержать разные компоненты. В случае с преподавателями есть две вкладки: «Мои материалы» с созданными данным автором материалами и «Избранное» – с сохраненными материалами. В случае обычных пользователей личный кабинет будет содержать только вкладку «избранное». Дизайн профиля представлен на рис. 14.

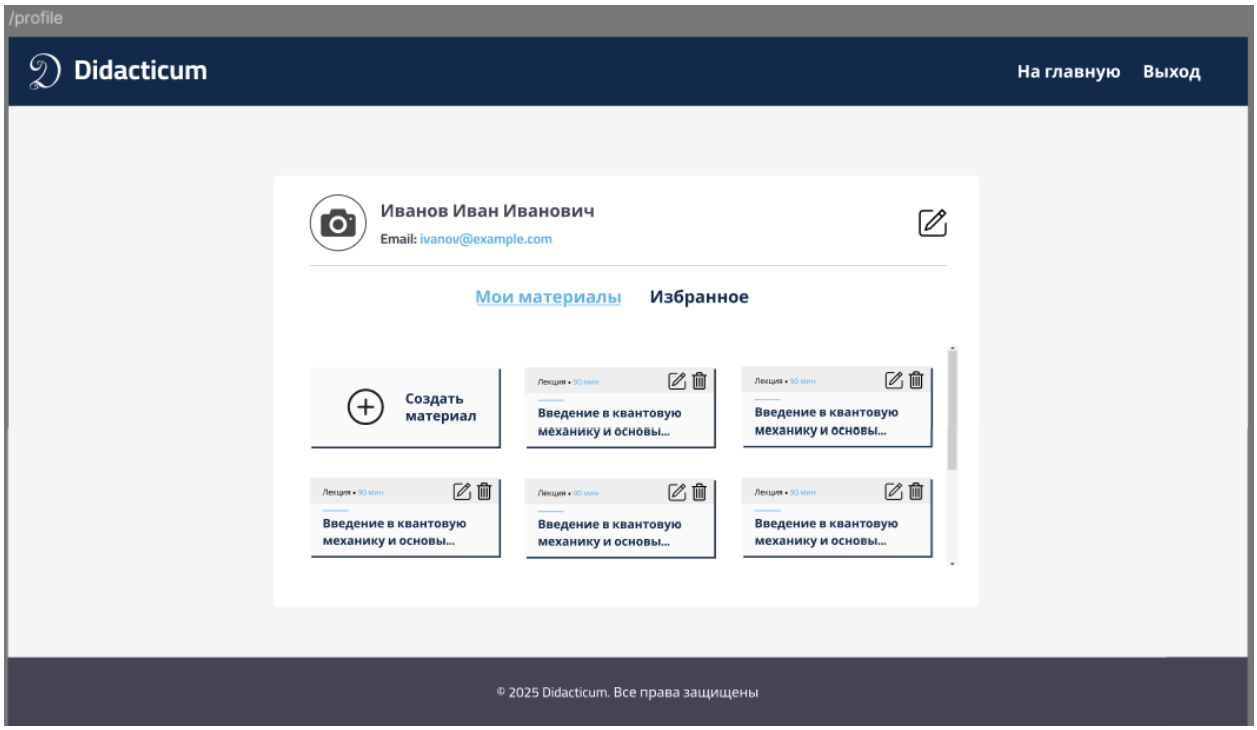

Рисунок 14 – Дизайн UI страницы профиля

Теперь опишем дизайн главной функциональной страницы сайта. Это страница редактирования учебных материалов. Центральную часть страницы занимает пространство для блоков содержания материала. Левый сайдбар содержит доступные инструменты, благодаря которым можно добавлять блоки контента. Правый сайдбар состоит из различных кнопок взаимодействия с материалом: загрузки, скачивания, сохранения и просмотра.

На рис. 15. Представлен дизайн страницы редактирования материалов.

Страница просмотра материала содержит отрендеренные в красивой форме блоки контента, а также минимальную функциональность по скачиванию материала и добавлению в избранное (см. рис. 16).

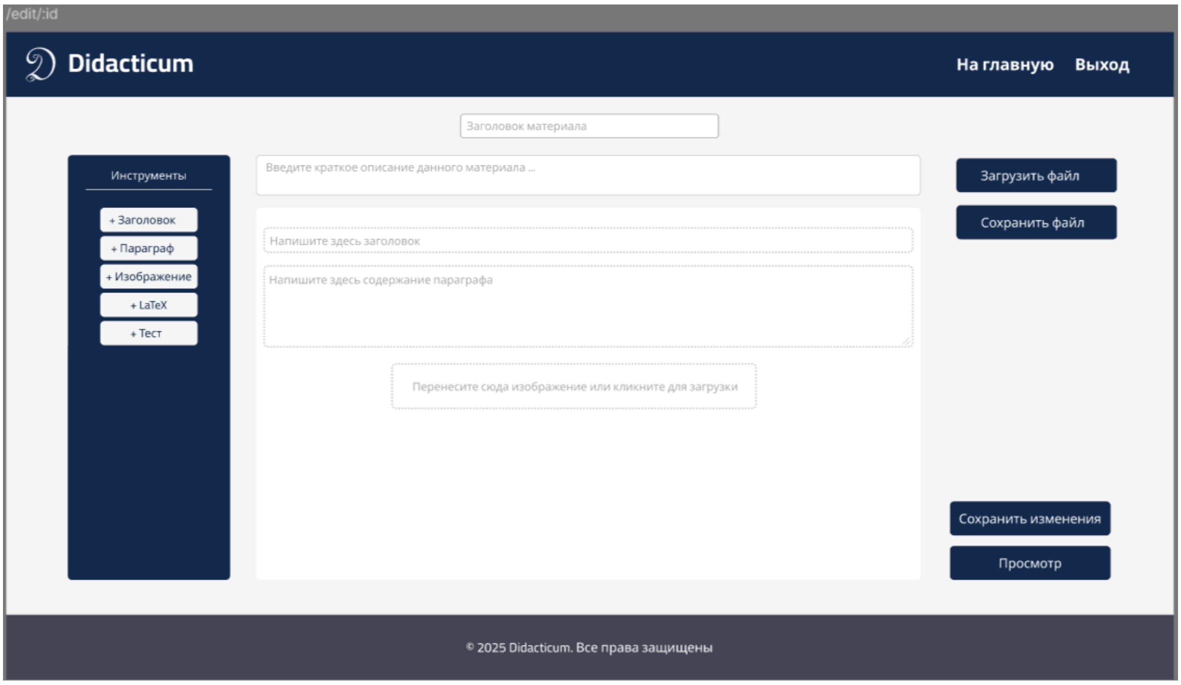

Рисунок 15 – Дизайн UI страницы редактирования материалов

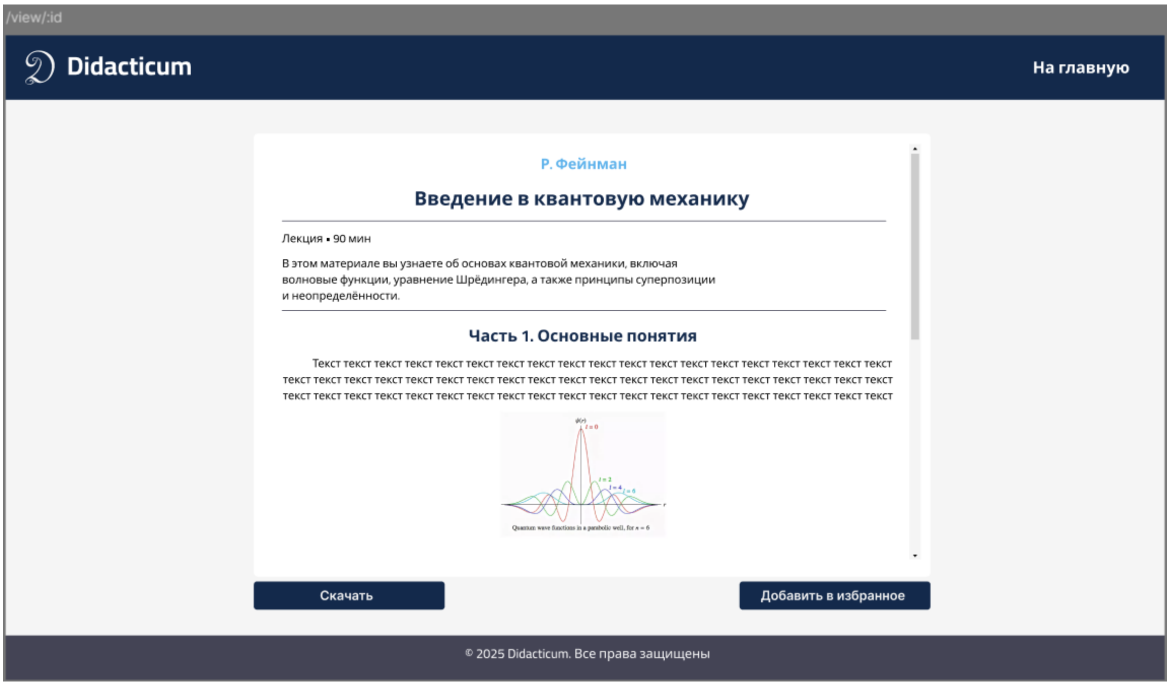

Рисунок 16 – Дизайн UI страницы просмотра материалов

Спроектированный в данном разделе дизайн уже определяет структуру приложения и описывает пользовательский интерфейс, что станет основой последующей программной разработки.

С полным макетом дизайна можно ознакомиться по ссылке: https://www.figma.com/community/file/1507878607651954459

3 РЕАЛИЗАЦИЯ ПРОДУКТА

3.1 Разработка программного продукта

3.1.1 Настройка базовых конфигураций

Разработка приложения начинается с настройки окружения и внедрения необходимых зависимостей. В нашем случае для начала необходимо инициализировать Node.js проект. Затем установить библиотеку React и вспомогательные библиотеки Redux и React-DOM. После этого настроить поддержку типов и компиляцию TypeScript. Ход первоначальной настройки проекта представлен на рис. 17.

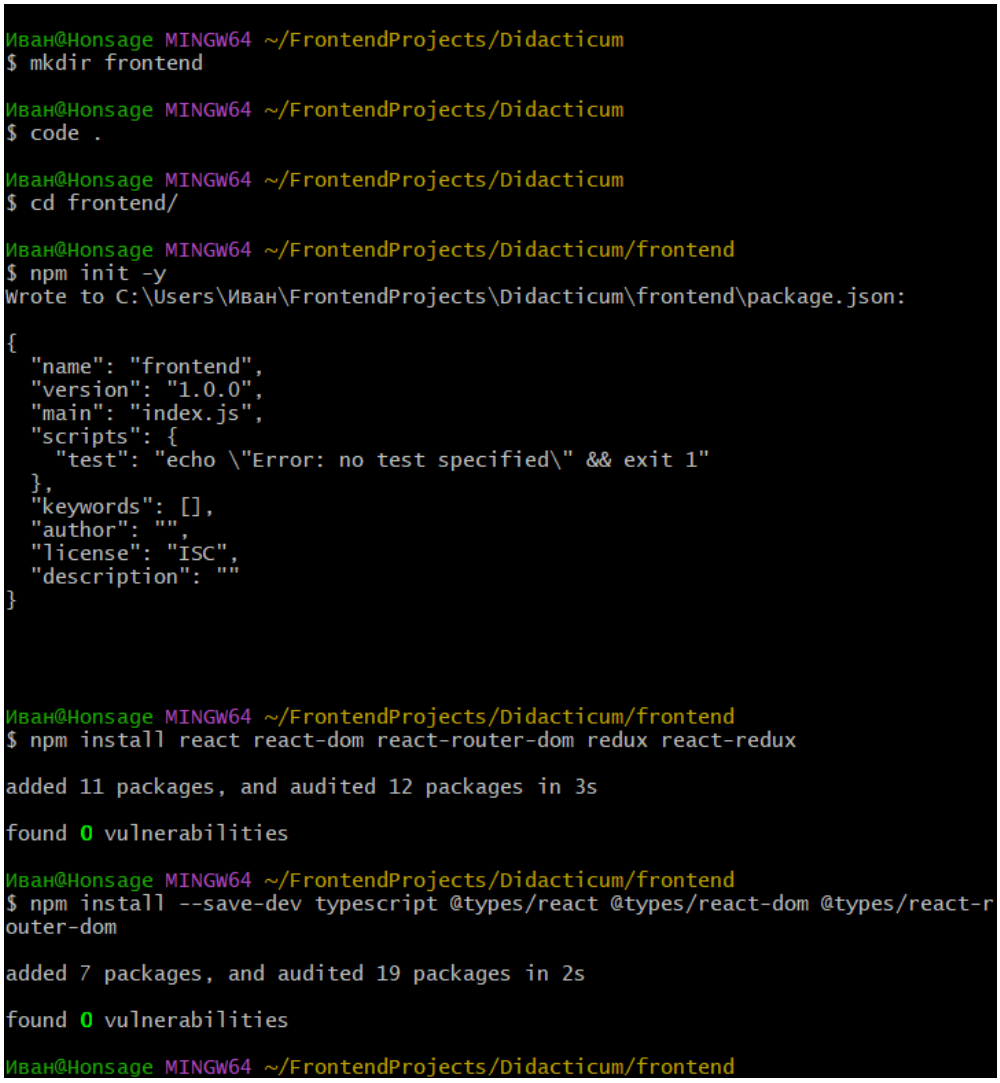

Рисунок 17 – Установка библиотеки React и настройка TS

После того, как инструменты разработки установлены, необходимо настроить сборщик проекта. В нашем случае устанавливаем Webpack и необходимые загрузчики к нему. Для стандартизации кода и более строгой проверки конструкций настраиваем также инструменты ESLint и Prettier. Использование этих инструментов позволит поддерживать чистоту кода, что снизит сложность его дальнейшего восприятия. Процесс настройки сборки представлен на рис. 18.

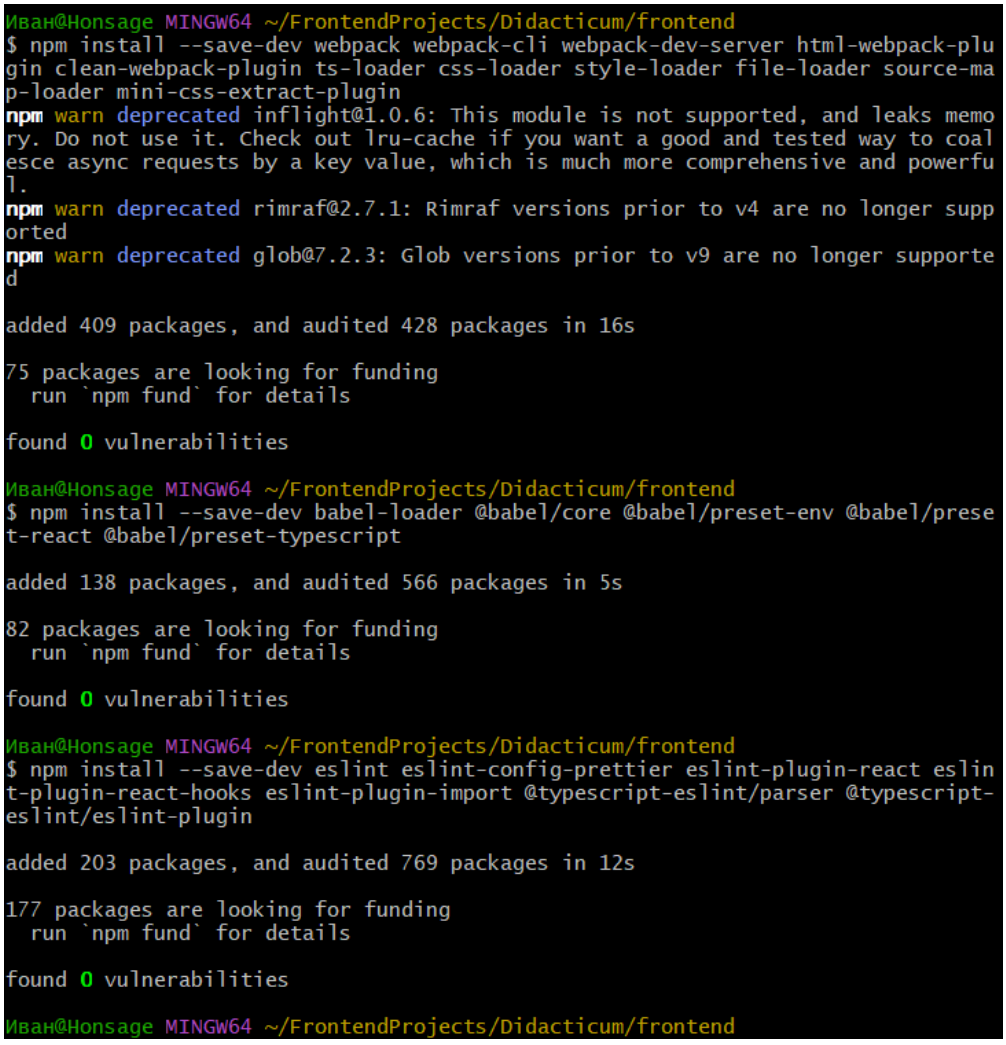

Рисунок 18 – Настройка сборщика Webpack и линтеров

В нашей платформе также будут использоваться компоненты, рендерящие Markdown и LaTeX элементы. Поэтому воспользуемся уже готовыми библиотеками React-Markdown и Katex. Установим их в наш проект (см. рис. 19).

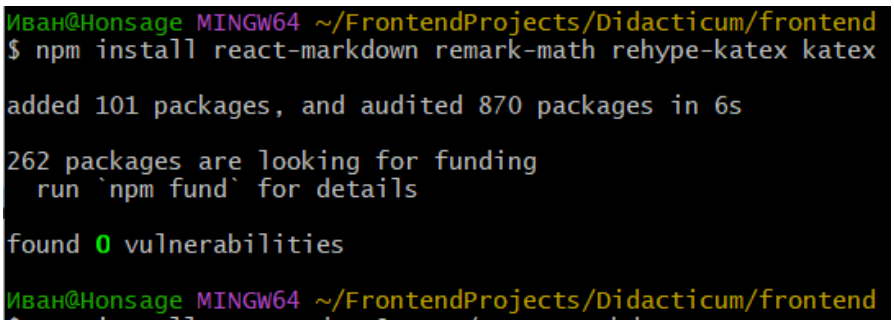

Рисунок 19 – Установка библиотек для работы с Markdown и LaTeX

Для удобной сборки и запуска проекта настроим npm скрипты в файле package.json (см. рис. 20).

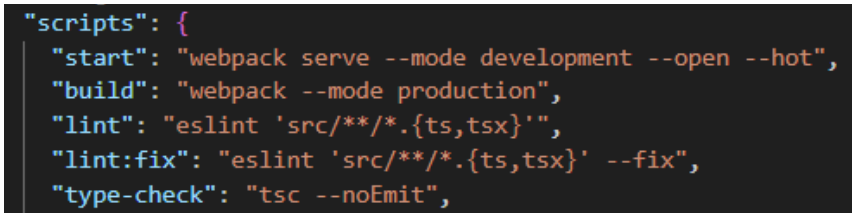

Рисунок 20 – Настройка npm скриптов

А webpack.config.js настроим так, чтобы при запуске проекта автоматически открывался браузер на указанном порту (см. рис. 21).

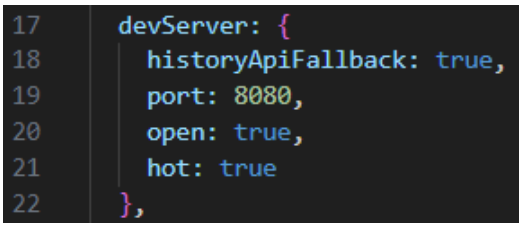

Рисунок 21 – Настройка сервера разработки

На этом основные настройки проекта завершены, и можно приступать непосредственно к написанию программного кода продукта.

3.1.2 Организация хранения данных

Перед тем, как приступать к настройке регистрации и авторизации пользователей, стоит уделить внимание вопросам хранения информации о них.

Ранее мы установили библиотеку Redux. Она позволяет организовывать удобное хранилище (redux store), доступное из всех компонентов системы. Redux оперирует такими понятиями, как «срезы» (slices). Это структурированный набор методов и правил взаимодействия с хранилищем, отвечающих за конкретную область хранилища. Организуем такой «срез» userSlice для хранения информации о пользователе (см. рис. 22).

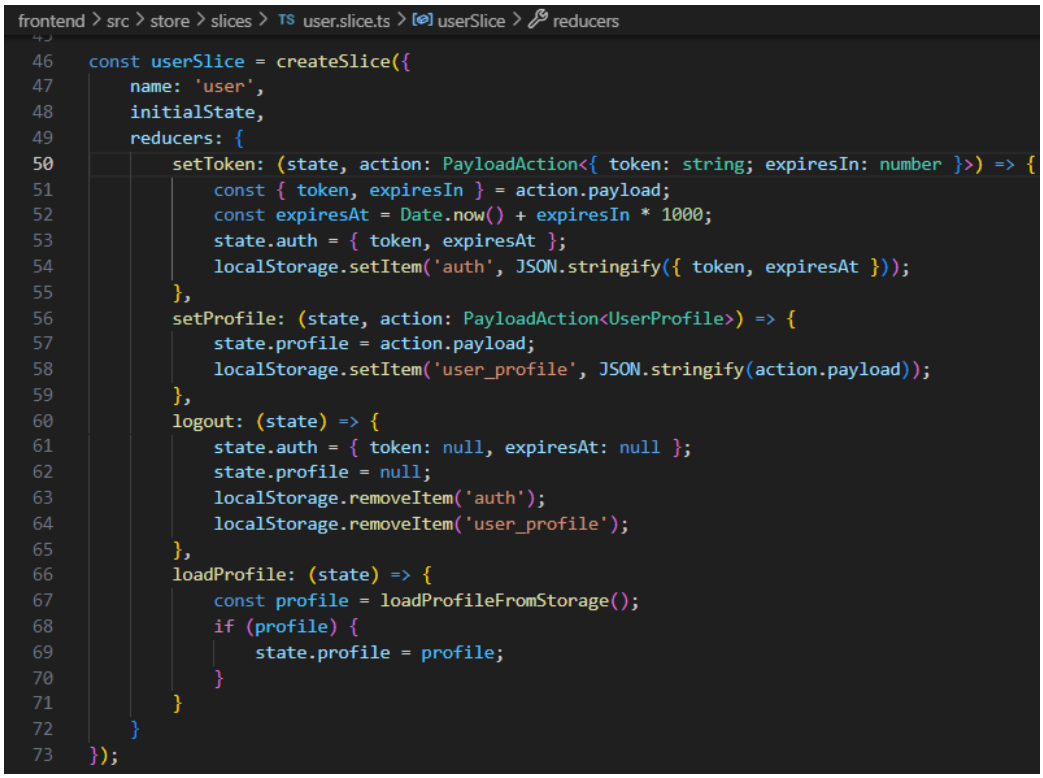

Рисунок 22 – Реализация среза userSlice

Теперь мы легко можем настроить само хранилище (см. рис. 23). Для этого достаточно просто сослаться на обработчика из «среза».

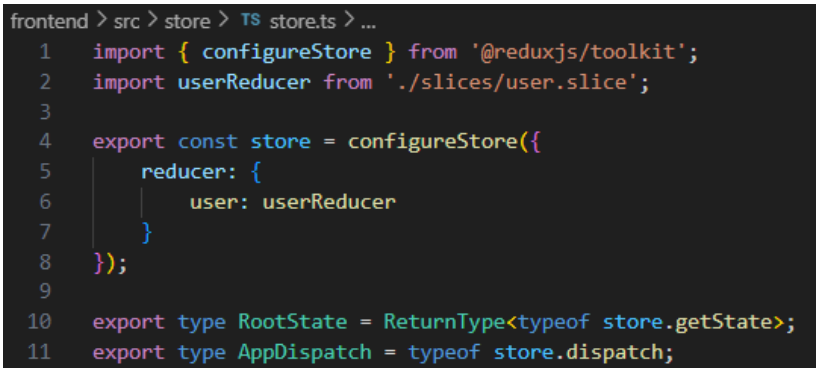

Рисунок 23 – Настройка Redux store

3.1.3 Настройка регистрации и авторизации

Чтобы обработка состояния авторизации пользователя не стала настоящей морокой, вынесем все методы по проверке состояния, организации регистрации, осуществления входа и прочие в отдельный класс AuthService (см. рис. 24).

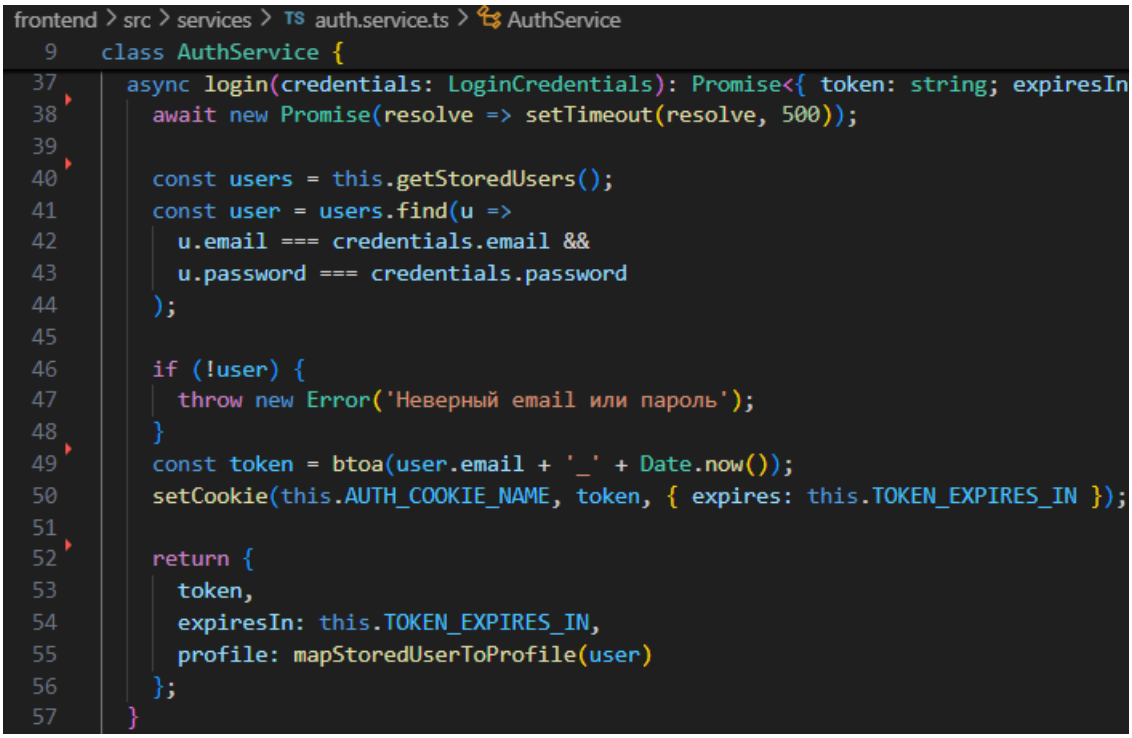

Рисунок 24 – Реализация класса для обработки регистрации и авторизации

Теперь мы легко можем использовать функциональность регистрации и входа пользователя в других компонентах.

Авторизация осуществляется за счет генерации токена входа в систему. Также поддерживается периодическая генерация refresh-токена для обеспечения возможности сохранения сессии даже при закрытии браузера. Токены хранятся в cookies.

3.1.4 Настройка роутинга

Теперь, когда у нас настроена авторизация пользователей, можно говорить о таком понятии как «защищенный роут». Это страница, доступ к которой есть только у авторизованного пользователя.

В соответствии со структурой нашей платформы определим, какие маршруты являются защищенными, а какие – общедоступными. Весь роутинг определим в функциональном компоненте App – собирающем все страницы сайта в объединенное приложение (см. рис. 25).

```
export default function App() {
  return (
    <Routes>
      <Route path="/" element={<HomePage />} />
      <Route path="/search" element={<SearchPage />} />
      <Route path="/login" element={<LoginPage />} />
      <Route path="/signup" element={<SignupPage />} />
      <Route path="/viewer" element={<ViewerPage />} />
      <Route path="/viewer/:id" element={<ViewerPage />} />
      <Route
        path="/profile"
        element={
          <ProtectedRoute>
            <ProfilePage />
          </ProtectedRoute>
        }
      />
      <Route
        path="/edit/:id"
        element={
          <ProtectedRoute>
            <EditPage />
          </ProtectedRoute>
        }
      />
      <Route path="*" element={<NotFound />} />
    </Routes>
  );
}
```

Рисунок 25 – Настройка роутинга сайта

3.1.5 Разработка переиспользуемых компонентов

Перейдем к реализации компонентов. Так как их в проекте очень много, а структура их схожа, то разберем разработку компонентов на примере компонента формы входа LoginForm.

В данном проекте компоненты реализуются в функциональном стиле. Это значит, что каждый компонент представляет собой функцию, получающую в качестве аргументов внешние данные («пропсы») и возвращающую JSX-разметку.

Динамичность компонентов обеспечивается хуками (hooks) – специальными функциями высшего порядка, позволяющими хранить состояние различных элементов, выполнять фоновые задачи или ссылаться на объекты в разметке.

React-проект традиционно для разметки использует не html-файлы, а JSX разметку, т.е. указание разметки компонента прямо в его функциональном коде. Это обеспечивает наглядность и удобство разработки. И разметка, и ее обработка находятся в одном месте, что позволяет концентрироваться на конкретной задаче. Реализация компонента LoginForm с применением JSX разметки представлена на рис. 26.

```
15  const LoginForm: React.FC<LoginFormProps> = ({
16    email,
17    password,
18    error,
19    isLoading = false,
20    onEmailChange,
21    onPasswordChange,
22    onSubmit
23  }) => {
24    return (
25      <form className={styles.form} onSubmit={onSubmit} data-testid="login-form">
26        <h1 className={styles.title}>Вход в систему</h1>
27
28        {error && <div className={styles.error} data-testid="error-message">{error}</div>}
29
30        <div className={styles.inputGroup}>
31          <label htmlFor="email">Электронная почта</label>
32          <input
33            type="email"
34            id="email"
35            name="email"
36            value={email}
37            onChange={(e) => onEmailChange(e.target.value)}
38            placeholder="you@example.com"
39            required
40            disabled={isLoading}
41            data-testid="email-input"
42          />
43        </div>
```

Рисунок 26 – Реализация функционального компонента LoginForm

Для обеспечения стилей отображения используются файлы с CSS стилями. В нашем проекте применяется модульный подход. Он позволяет создавать стили для каждого компонента по отдельности (см. рис. 27).

frontend > src > components > login-form > # login-form.module.css >

```
.form {
  background: white;
  padding: 2.5rem;
  border-radius: 12px;
  box-shadow: 0 4px 6px rgba(19, 41, 75, 0.1);
  width: 100%;
  max-width: 480px;
}

.title {
  color: #13294B;
  font-size: 2rem;
  font-weight: 600;
  margin-bottom: 2rem;
  text-align: center;
}
```

Рисунок 27 – CSS- разметка компонента LoginForm

И сам компонент, и его стили хранятся вместе в отдельной папке. Такая структура и обеспечивает независимость и переиспользуемость компонентов. На рис. 29 показан список с папами компонентов нашего проекта.

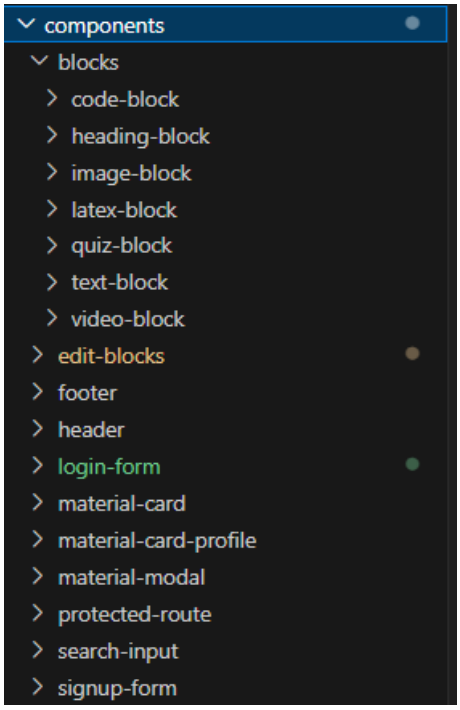

Рисунок 28 – Список директорий с компонентами проекта

После того как все унитарные компоненты готовы, можно переходить к их использованию в разметке страниц. Для страниц все так же используются функциональные компоненты, возвращающие JSX разметку. В качестве примера приведем страницу LoginPage, использующую компоненты Header, LoginForm и Footer (см. рис. 28).

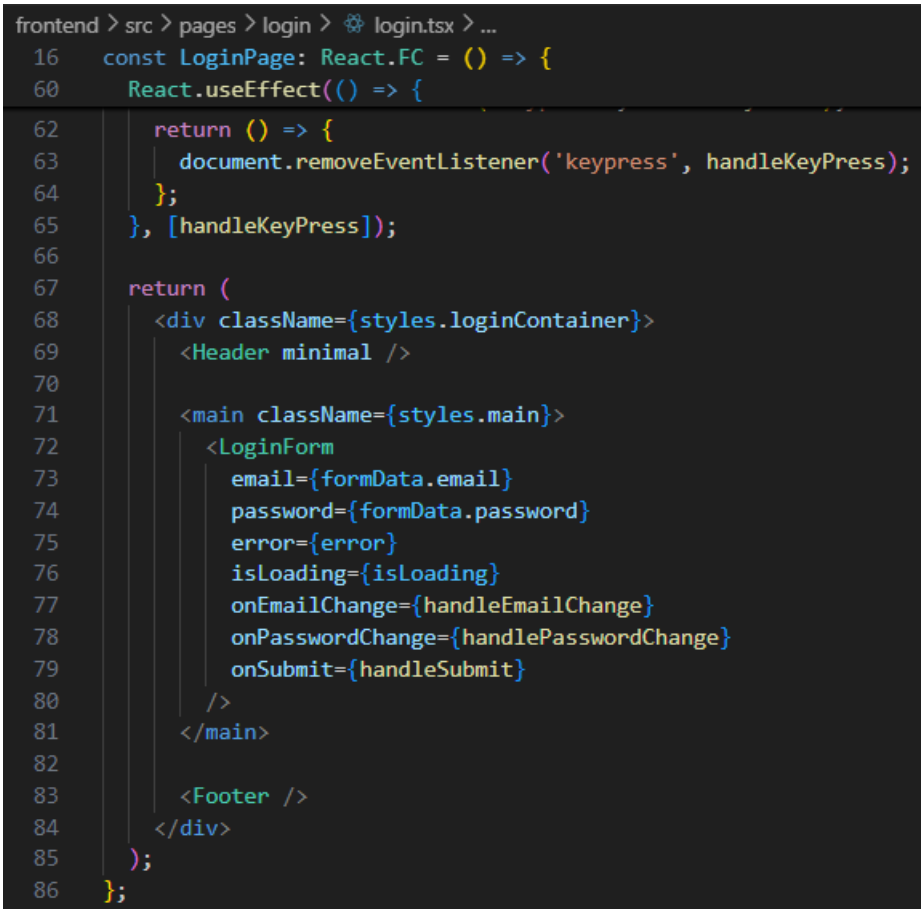

Рисунок 29 – Реализация страницы входа

3.1.6 Настройка IndexedDB

В обосновании принимаемых технических решений мы делали ремарку насчет временного хранения учебных материалов. Пока у нашего проекта нет бэкенда, будем использовать браузерную базу данных IndexedDB.

Для работы с ней и обеспечения принципа DRY (don’t repeat yourself – не повторяйся) создадим отдельный класс MaterialsService. В нем будем инициализировать базу, осуществлять добавление, удаление и

индексированный поиск в ней. Реализация многофункционального класса представлена на рис. 30.

frontend > src > services > storage > TS materials.service.ts > ...

```
8   class MaterialsService {
9       private db: IDBDatabase | null = null;
10
11      async init(): Promise<void> {
12          if (this.db) return;
13
14          return new Promise((resolve, reject) => {
15              const request = indexedDB.open(DB_NAME, DB_VERSION);
16
17              request.onerror = () => {
18                  reject(new Error('Failed to open database'));
19              };
20
21              request.onsuccess = () => {
22                  this.db = request.result;
23                  resolve();
24              };
25
26              request.onupgradeneeded = (event) => {
27                  const db = (event.target as IDBOpenDBRequest).result;
28
29                  if (db.objectStoreNames.contains(MATERIALS_STORE)) {
30                      db.deleteObjectStore(MATERIALS_STORE);
31                  }
32
33                  const store = db.createObjectStore(MATERIALS_STORE, { keyPath: 'metadata.id' });
34
35                  store.createIndex('title', 'metadata.title', { unique: false });
36                  store.createIndex('type', 'metadata.type', { unique: false });
37                  store.createIndex('authorId', 'metadata.author.id', { unique: false });
38                  store.createIndex('createdAt', 'metadata.createdAt', { unique: false });
39
40                  store.transaction.oncomplete = () => {
41                      const materialStore = db.transaction(MATERIALS_STORE, 'readwrite')
42                          .objectStore(MATERIALS_STORE);
43
44                      LEARNING_MATERIALS.forEach(material => {
45                          materialStore.add(material);
46                      });
47                  };
48              };
49          });
50      }
```

Рисунок 30 – Реализация класса для работы с IndexedDB

3.1.7 Создание конструктора учебных материалов

Конструктор учебных материалов представляет собой удобный GUI инструмент, позволяющий преподавателю создавать учебные материалы при помощи добавления блоков контента.

Блоки контента представляют собой стандартные функциональные компоненты. Для управления ими организован отдельный компонент

BlockManager. Он оборачивает каждый из блоков в перемещаемый контекст, тем самым обеспечивая возможность изменения порядка следования блоков.

Рисунок 31 – Реализация компонента BlockManager

За выдачу конкретной реализации контента отвечает утилита block-factory (см. рис. 32).

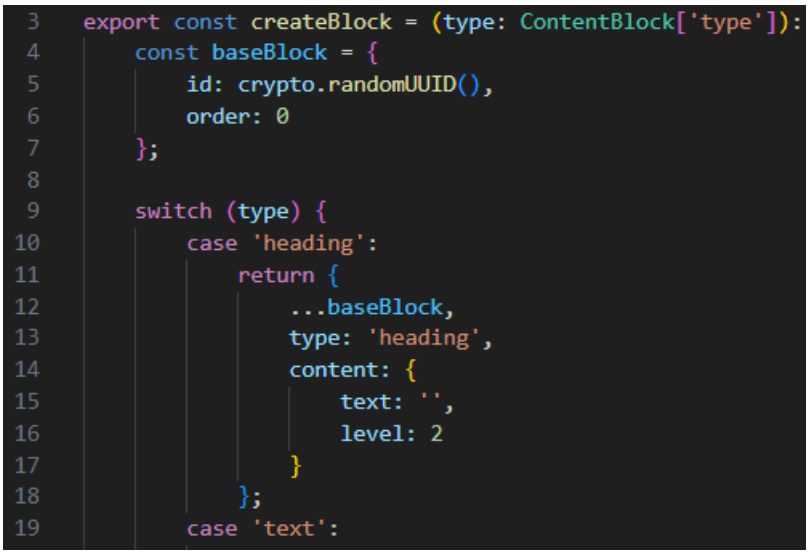

Рисунок 32 – Фабрика блоков контента

Разработанная функциональность позволяет пользователю просто нажимать на кнопки добавления блока (см. рис. 33), после чего заполнять блоки текстом, ссылками или своими изображениями.

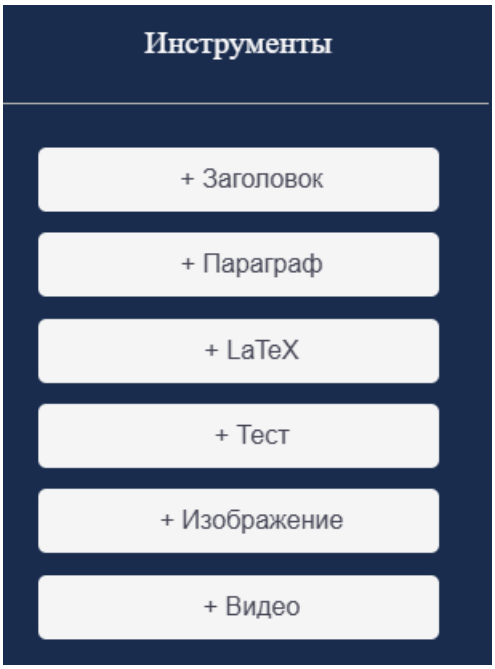

Рисунок 33 – Набор инструментов добавления блоков

Каждый из блоков имеет справа панель, позволяющую удобно перемещать блоки вверх-вниз и удалять (см. рис. 34).

Рисунок 34 – Пример редактирования интерактивного блока тестирования

3.1.8 Спецификация файлового расширения

Специфицируем наш формат ILM. Определим его как JS-объект, имеющий два обязательных поля первого порядка: metadata и blocks. Metadata описывает метаданные материала (автора, название, тип, дата создания и т.п.), а blocks содержит массив блоков, описанных каждый по своей структуре (см. рис. 35).

Рисунок 35 – Пример хранения материала в формате ILM

Для осуществления сохранения и подгрузки материалов такой структуры определим утилиты сериализации и десериализации (см. рис. 36).

frontend > src > utils > ilm.utils.ts > deserializeFromILM

```typescript
import { Material, ContentBlock } from '../types/material.types';

export const serializeToILM = (material: Material): string => {
    return JSON.stringify(material, null, 2);
};


export const deserializeFromILM = (content: string): Material => {
    try {
        const material = JSON.parse(content);

        if (!material.metadata || !material.metadata.id || !material.metadata.title) {
            throw new Error('Некорректная структура файла ILM: отсутствуют обязательные поля метаданных');
        }

        if (material.blocks) {
            if (!Array.isArray(material.blocks)) {
                throw new Error('Некорректная структура файла ILM: blocks должен быть массивом');
            }

            material.blocks.forEach((block: ContentBlock, index: number) => {
                if (!block.id || !block.type || !block.content) {
                    throw new Error(`Некорректная структура блока ${index}: отсутствуют обязательные поля`);
                }
            });
        }

        return material;
    } catch (error) {
        if (error instanceof Error) {
            throw new Error(`Ошибка при чтении файла ILM: ${error.message}`);
        }
        throw new Error('Ошибка при чтении файла ILM');
    }
};
```

Рисунок 36 – Реализация утилиты по конвертации между ILM и программным представлением

3.2 Тестирование

3.2.1 Модульное тестирование

Модульное тестирование представляет собой тестирование отдельных компонентов в изоляции. Такое тестирование позволяет выявить неправильную реализацию логики работы компонента.

Для модульного тестирования будем использовать инструмент Jest. Для тестирования достаточно рядом с определением компонента создать файл такого же названия, но с интерполированным «.test.».

Организуем тестирование модулей нашего проекта. Сначала протестируем корректность работы нашего хранилища. Для этого рассмотрим возвращаемые значения методов из «срезов». Реализация тестирования представлена на рис. 37.

```
describe('Слайс пользователя', () => {
    let testStore: ReturnType<typeof configureStore>;

    beforeEach(() => {
        mockLocalStorage.clear();
        testStore = configureStore({
            reducer: {
                user: userReducer
            }
        });
    });

    describe('Начальное состояние', () => {
        it('должно иметь пустые значения для auth и profile', () => {
            const state = testStore.getState() as RootState;
            expect(state.user.auth).toEqual({ token: null, expiresAt: null });
            expect(state.user.profile).toBeNull();
        });

        it('не должно загружать просроченный auth из localStorage', () => {
            const mockAuth = {
                token: 'test-token',
                expiresAt: Date.now() - 3600000
            };
            mockLocalStorage.setItem('auth', JSON.stringify(mockAuth));

            testStore = configureStore({
                reducer: {
                    user: userReducer
                }
            });
```

Рисунок 37 – Пример тестирования хранилища и слайсов

Для отдельных компонентов и страниц проекта также напишем модульные тесты, которые будут проверять соответствие ожидаемых возвращаемых значений действительным. На рисунке 38 представлен модульный тест уже рассмотренного компонента LoginForm. В частности, проверяется корректность реакции полей формы на ввод данных.

Запустим инструмент Jest в консоли и посмотрим на отчет о выполнении тестов (см. рис. 39). Сообщение отчета говорит нам об успешном выполнении всех тестов.

Успех модульного тестирования позволяет нам сделать вывод о корректности реализации логики компонентов.

```
describe('Тестирование компонента LoginForm', () => {
    beforeEach(() => {
        jest.clearAllMocks();
    });

    it('обработка изменения email', () => {
        const props = { ...mockProps };
        props.onEmailChange('test@example.com');
        expect(props.onEmailChange).toHaveBeenCalledWith('test@example.com')
    });

    it('обработка изменения пароля', () => {
        const props = { ...mockProps };
        props.onPasswordChange('testpass123');
        expect(props.onPasswordChange).toHaveBeenCalledWith('testpass123');
    });

    it('обработка отправки формы', () => {
        const props = { ...mockProps };
        props.onSubmit();
        expect(props.onSubmit).toHaveBeenCalled();
    });

```

Рисунок 38 – Пример тестирования компонента формы входа

```
      at Object.require (src/store/store.test.ts:1:1)

 PASS  src/components/login-form/login-form.test.tsx

Test Suites: 7 passed, 7 total
Tests:       47 passed, 47 total
Snapshots:   0 total
Time:        1.156 s
Ran all test suites.
```

Рисунок 39 – Отчет об успешном выполнении модульных тестов

3.2.2 Интеграционное тестирование

Интеграционное тестирование применяют для тестирования системы как целостного программного продукта.

В частности, мы будем осуществлять e2e тестирование, которое проверяет функциональность «от начала и до конца», или, говоря более понятно, «в боевых условиях».

Ограничимся «боевым крещением» только самых главных сегментов системы – регистрации и авторизации пользователя и создания учебных материалов. Остальное тестирование осуществим вручную.

Для осуществления e2e тестирования будем использовать инструмент Cypress. Он предоставляет удобный графический интерфейс для просмотра хода выполнения тестов. Описание тестов похоже на модульное, но уже использует не отдельные компоненты, а реальные методы, как набор данных с клавиатуры, клик мыши, переход по роуту и другое. На рис. 40 представлена часть интеграционного теста авторизации.

```
81    it('Успешный вход', () => {
82        cy.visit('/login');
83
84        cy.get('[data-testid="email-input"]')
85            .should('be.visible')
86            .type(testUser.email);
87
88        cy.get('[data-testid="password-input"]')
89            .should('be.visible')
90            .type(testUser.password);
91
92        cy.get('[data-testid="login-button"]')
93            .should('be.visible')
94            .should('not.be.disabled')
95            .click();
96
97        cy.url().should('not.include', '/login');
98
99        cy.get('[data-testid="user-name"]').should('contain', 'Admин');
100       cy.get('[data-testid="profile-link"]').should('exist');
101       cy.get('[data-testid="logout-button"]').should('exist');
102   });
```

Рисунок 40 – Пример тестирования сценария успешного входа в систему

Запуск сценариев e2e тестирования в области регистрации и авторизации завершился с успехом (см. рис. 41).

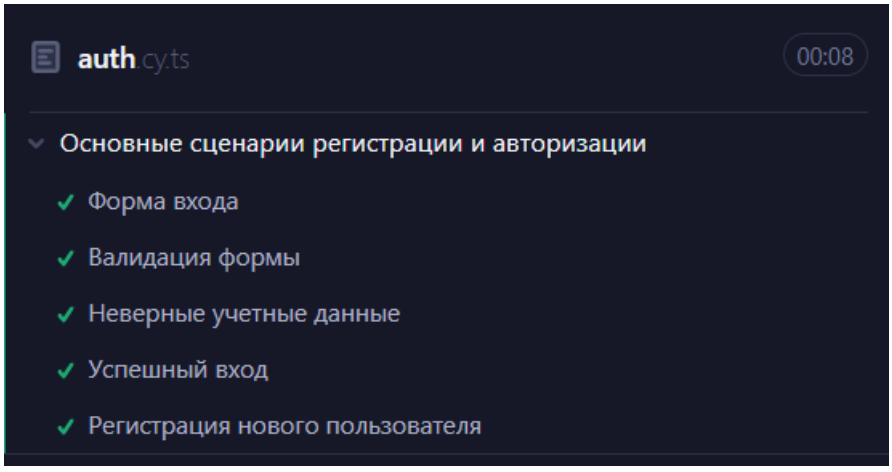

Рисунок 41 – Отчет об успешном выполнении тестирования авторизации

Проведем аналогичное тестирование и для конструктора учебных материалов (см. рис. 42).

```
31    it('Создание нового материала с разными блоками', () => {
32        cy.get('[data-testid="material-title"]').should('exist');
33        cy.get('[data-testid="material-description"]').should('exist');
34        cy.get('[data-testid="blocks-sidebar"]').should('exist');
35        cy.get('[data-testid="preview-sidebar"]').should('exist');
36
37        const title = 'Тестовый материал';
38        const description = 'Описание тестового материала';
39
40        cy.get('[data-testid="material-title"]').type(title);
41        cy.get('[data-testid="material-description"]').type(description);
42
43        cy.get('[data-testid="add-heading-block"]').click();
44        cy.get('[data-testid="heading-input"]').last().type('Тестовый заголовок');
45
46        cy.get('[data-testid="add-paragraph-block"]').click();
47        cy.get('[data-testid="paragraph-input"]').last().type('Тестовый параграф с текстом');
48
49        cy.get('[data-testid="add-latex-block"]').click();
50        cy.get('[data-testid="latex-input"]').last().type('x^2 + y^2 = z^2');
51
52        cy.get('[data-testid="content-block"]').should('have.length', 3);
53        cy.get('[data-testid="content-block"]').eq(0).find('[data-testid="heading-input"]').sh
54        cy.get('[data-testid="content-block"]').eq(1).find('[data-testid="paragraph-input"]').
55        cy.get('[data-testid="content-block"]').eq(2).find('[data-testid="latex-input"]').shou
56
57        cy.get('[data-testid="save-material"]').click();
58
59        cy.url().should('include', '/profile');
60    });
61
62    it('Перемещение блоков', () => {
```

Рисунок 42 – Пример тестирования сценария создания материала из блоков

Результат выполнения тестов в области конструктора материалов также успешен (см. рис. 43).

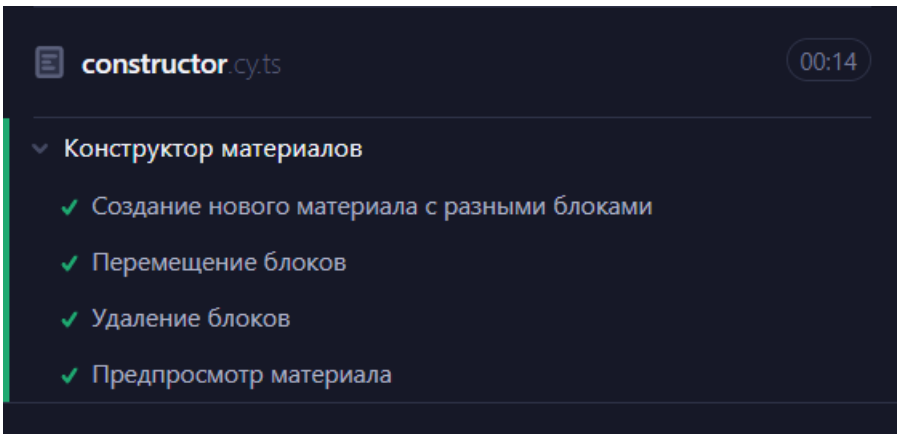

Рисунок 43 – Отчет об успешном выполнении тестирования конструктора

3.2.3 Ручное тестирование

Написание автоматических тестов – очень трудоемкий и времязатратный процесс. Порой бывает, что некоторые функции легче протестировать вручную. Проведем остаточное тестирование элементов платформы.

Рассмотрим работу модальных окон. При нажатии на карточку учебного материала открывается окно (см. рис. 44) с предложением вариантов взаимодействия с материалом.

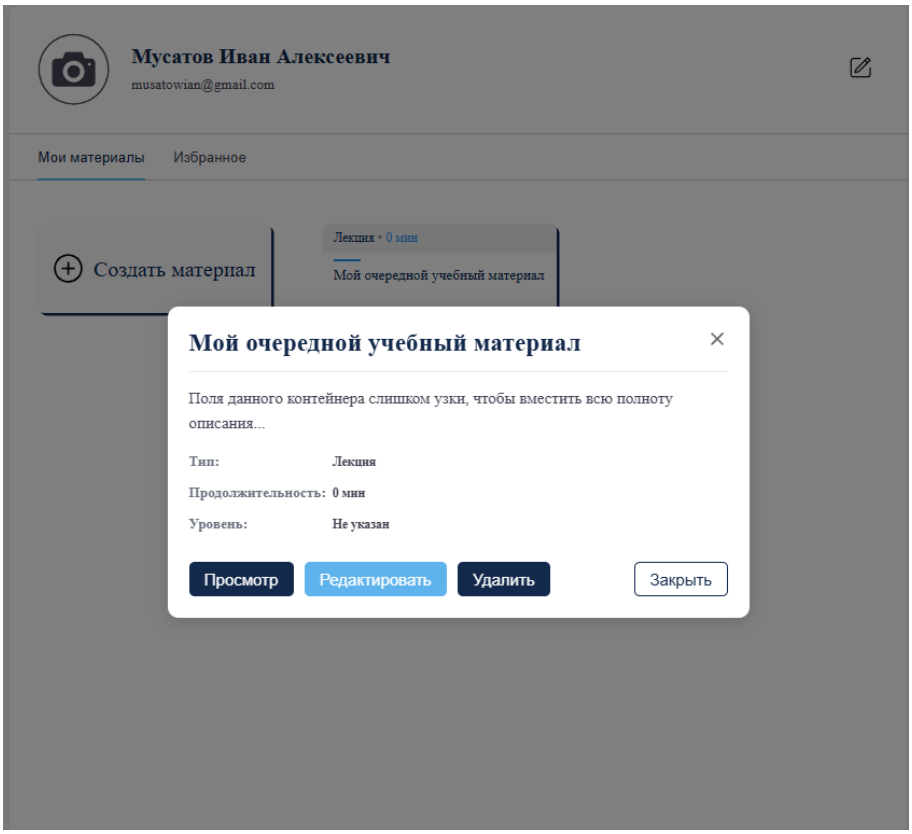

Рисунок 44 – Тестирование открытия модального окна карточки

Если нажать на кнопку редактирования, то платформа перенесет нас в конструктор материалов (см. рис. 45).

Отдельно проверим корректность загрузки и скачивания материалов в ILM формате (см. рис. 46).

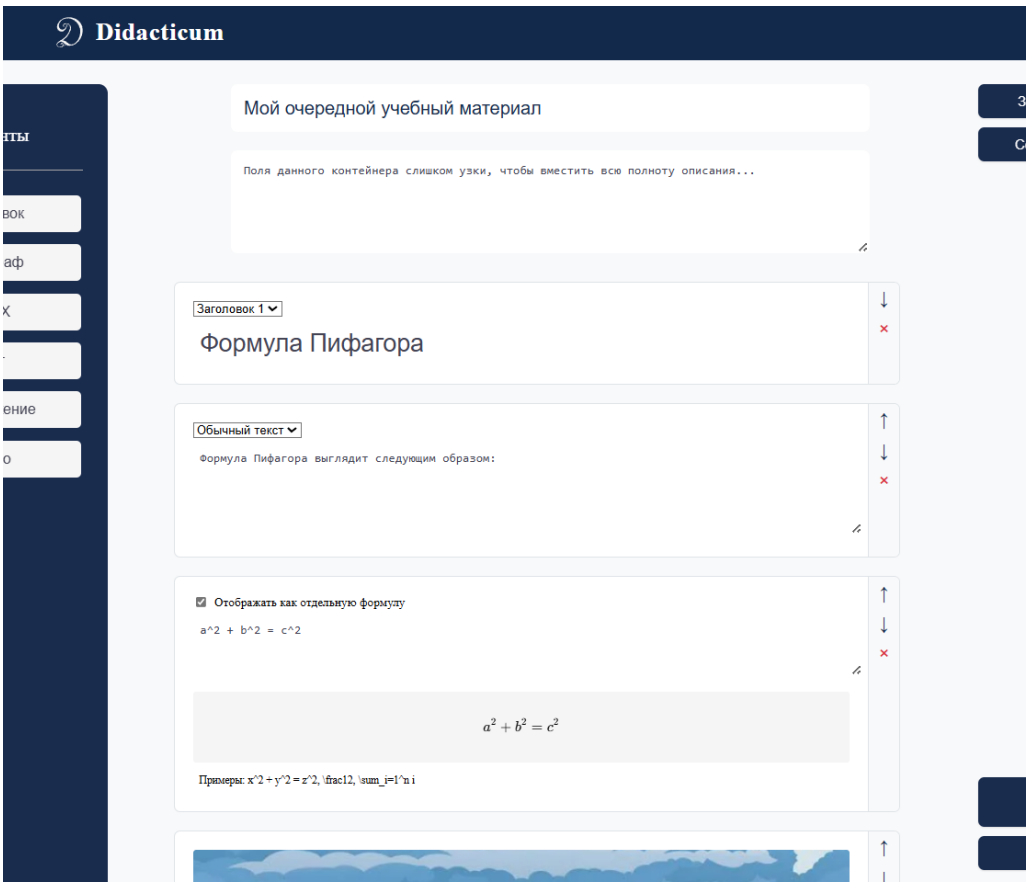

Рисунок 45 – Тестирование перехода к редактированию материала

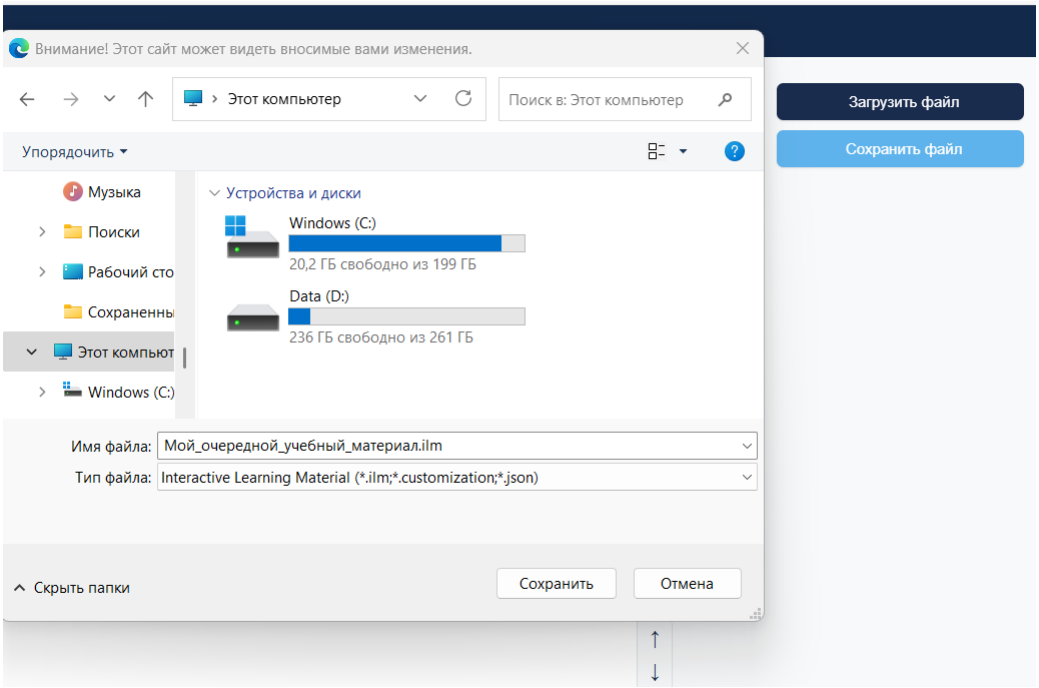

Рисунок 46 – Тестирование экспорта материала в формат ILM

Напоследок убедимся, что рендер материалов происходит корректно (см. рис. 47).

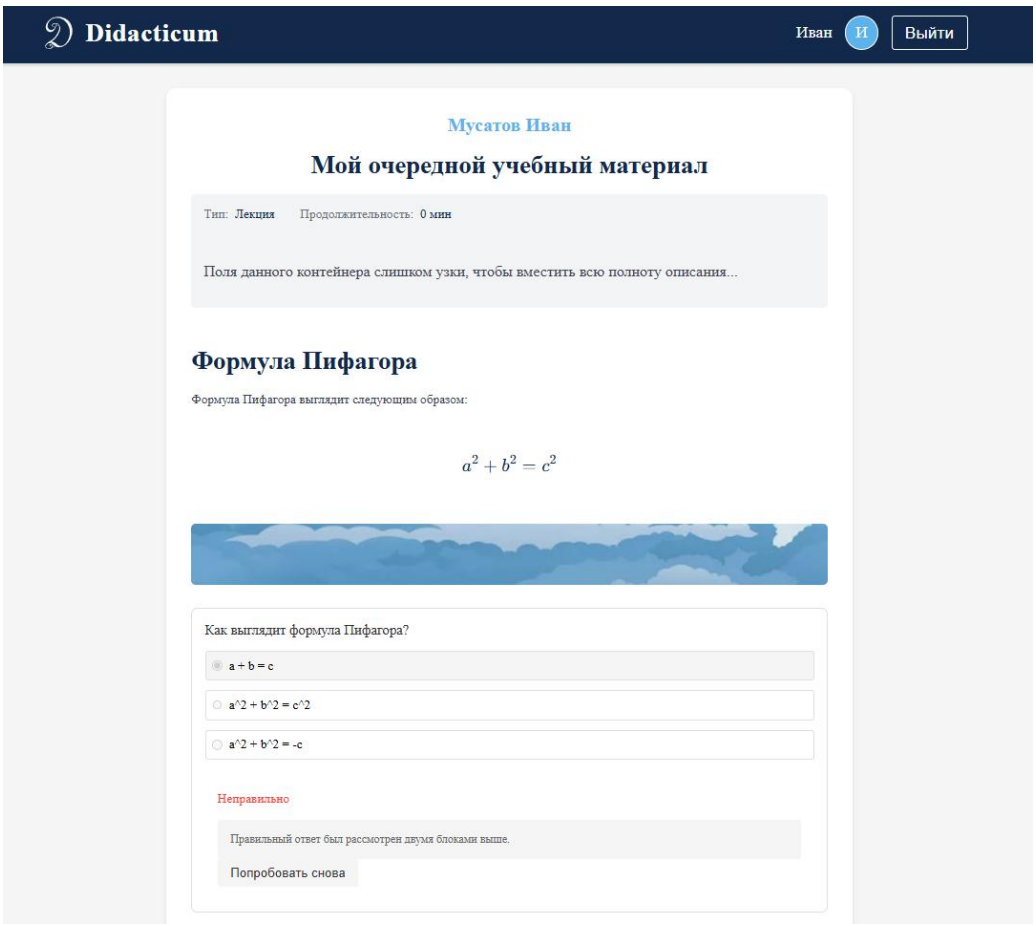

Рисунок 47 – Проверка корректности просмотра учебных материалов

Ручное тестирование позволило нам рассмотреть пользовательские сценарии поведения и убедиться в корректности работы программного продукта.

ЗАКЛЮЧЕНИЕ

В результате полноценного процесса проектирования и разработки была создана клиентская часть веб-платформы, предоставляющая возможности создания, редактирования и распространения интерактивных учебных материалов. Полученный программный продукт полностью соответствует поставленной цели, что свидетельствует об успешности реализации проекта.

Работа над продуктом началась с обширного теоретического анализа. Исследование предметной области позволило погрузиться в задачи современной дидактики и специфику цифровых образовательных продуктов. Анализ конкурентных платформ позволил выявить преимущества и недостатки существующих решений, что помогло учесть лучшие практики разработки в нашей предметной области. На основе анализа были сформированы как функциональные, так и нефункциональные требования к продукту, позволившие понять характеристику того, как приложение должно быть разработано.

Следующим этапом работы стало концептуальное моделирование. Сформирована концепция продукта, определяющая его структуру, описание и целевую аудиторию. Для формализации логики приложения были построены диаграммы пользовательских сценариев и схемы сайта. Особое внимание было уделено созданию бренда платформы. На основе проведения сравнительных анализов были обоснованы принимаемые технические решения и выбор инструментария разработки. Наконец, был спроектирован дизайн пользовательского интерфейса в виде полноценного макета в Figma.

Заключительным этапом работы над проектом стала реализация программного продукта. В процессе разработки были воплощены все ключевые технические решения и функции: авторизация, роутинг, хранилище данных и полноценный конструктор материалов. Архитектура фронтенда организована по принципам компонентного подхода, что обеспечивает ее

гибкость и масштабируемость. Для проверки работоспособности системы было проведено полноценное тестирование трех видов: модульное, интеграционное (e2e) и ручное. Каждый из этапов тестирования подтвердил стабильность и корректность работы программного продукта.

Таким образом, можно сделать вывод об успешности выполнения данной курсовой работы. Разработанный продукт имеет широкие возможности для масштабирования и развития, что может послужить основой для создания полноценной образовательной платформы, предоставляющей инновационные и универсальные методы построения интерактивных учебных материалов.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Камышева, Е.Ю. Педагогические условия реализации интерактивности в цифровой среде вуза // Вестник Шадринского государственного педагогического университета – 2022. – №1(53). – С. 30-36. – URL: https://vestnikshspu.ru/journal/article/view/884/661 (дата обращения: 22.05.2025).

2. Жистина, Л. Ф. Интерактивные методы цифровой дидактики / Л. Ф. Жистина // Академия профессионального образования. – 2020. – № 3(94). – С. 73-79. – URL: https://elibrary.ru/item.asp?id=42695710 (дата обращения: 22.05.2025).

3. Фаулер М. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. 3-е изд. / Пер. с англ. — СПб.: Символ-Плюс, 2004. — 192 с. – URL: https://books.4nmv.ru/books/uml_osnovy_3-e_izdanie_fail_pdf_595320.pdf?utm_source=chatgpt.com (дата обращения: 23.05.2025).

4. IndexedDB – Интерфейсы веб API: [Электронный ресурс]. URL: https://developer.mozilla.org/ru/docs/Web/API/IndexedDB_API (дата обращения: 24.05.2025).

5. Калугина, Ю.В. Роль цвета в веб-дизайне / Ю. В. Калугина, А. А. Кондакова, А. С. Михайлов, С. В. Стрельникова // Решетневские чтения. – 2018. – Т. 2. – С. 560-562. – URL: https://elibrary.ru/download/elibrary_36741883_31929757.pdf (дата обращения: 24.05.2025).

6. Home | htmlbook.ru: [Электронный ресурс]. URL: https://htmlbook.ru/ (дата обращения: 25.05.2025).

7. Руководство по React: [Электронный ресурс]. URL: https://metanit.com/web/react/ (дата обращения: 29.05.2025).

8. Katex – The fastest math typesetting library for the web: [Электронный ресурс]. URL: https://katex.org/docs/api (дата обращения: 27.05.2025).

9. Introduction to Cypress | Cypress Documentation: [Электронный ресурс]. URL: https://docs.cypress.io/app/core-concepts/introduction-to-cypress (дата обращения: 29.05.2025).