ДИПЛОМ

Разработка платформы электронного обучения с построением индивидуальных образовательных курсов

Оглавление

ВВЕДЕНИЕ

Данная работа выполнена в рамках участия автора в проекте «Открытая адаптивная школа талантов и фундаментального превосходства в области точных наук на базе синтеза возможностей человеческого и искусственного интеллектов», реализуемого по Программе стратегического академического лидерства «Приоритет-2030». Участие в этом проекте позволило не только обеспечить практическую направленность платформы, но и получить институциональную поддержку для реализации и развития исследуемых подходов к персонализированному обучению.

Разработка, представленная в дипломе, легла в основу проекта, удостоенного Государственной премии Правительства Санкт-Петербурга в номинации «Развитие инновационной деятельности в образовательной организации». Несмотря на то, что автор формально не был включён в число лауреатов из-за статуса студента, именно он разработал архитектуру и реализовал ключевые компоненты платформы.

Результаты, полученные в процессе работы, были представлены в ряде публикаций с участием автора. Среди них — статьи, опубликованные в материалах научных конференций и специализированных изданиях:
– «Модуль интерактивного обучающего тестирования StudyWays», Современное образование: содержание, технологии, качество, 2022;
– «Программная среда системы интерактивного обучающего тестирования с автоматической генерацией вариантов заданий и реакций системы на ответы учащихся», Современное образование, 2022;
– «Концепция и опыт применения системы электронного адаптационного тестирования в преподавании физики», Информатизация образования, 2022;
– и другие.

Актуальность темы определяется растущей потребностью в индивидуализации образовательного процесса, особенно в условиях массового и дистанционного обучения. Электронные образовательные платформы играют в этом процессе ключевую роль, предоставляя не только структурированный доступ к учебным материалам и инструментам тестирования, но и возможности для адаптации содержания под уровень подготовки, интересы и динамику прогресса каждого учащегося.

Разработанная в рамках настоящего исследования платформа StudyWays представляет собой программное решение, ориентированное на реализацию персонализированного адаптивного обучения. За счёт модульной архитектуры, использования рекомендательных систем и специально разработанных инструментов, StudyWays позволяет формировать индивидуальные образовательные траектории, обеспечивая более глубокое и осознанное освоение материала.

Обьект исследования – организация электронного обучения с использованием технологий, направленных на персонализацию учебного процесса.

Предмет исследования – архитектурные и алгоритмические решения, направленные на выстраивание индивидуальных образовательных траекторий в электронных обучающих платформах.

Цель исследования – разработка программной реализации электронной образовательной платформы, обеспечивающей построение индивидуальных образовательных траекторий.

В рамках работы выделены следующие задачи исследования:

1. Проанализировать существующие образовательные платформы и используемые в них механизмы реализации индивидуального обучения.
2. Сформатировать требования к разрабатываемой платформе.
3. Спроектировать и реализовать программную реализацию образовательной платформы.
4. Осуществить апробацию и внедрение платформы в образовательный процесс одной или нескольких учебных организаций.
5. Сформировать рекомендации по дальнейшему развитию платформы.

Практическая значимость работы заключается в разработке и внедрение образовательной платформы, ориентированной на создание индивидуальных образовательных траекторий обучения.

Разработанная платформа апробирована и активно применяется для сопровождения обучения по направлению физика в таких образовательных учреждениях как:

- Санкт-Петербургский государственный электротехнический университет «ЛЭТИ».
- Университет ИТМО.
- Губернаторский физико-математический лицей №30.

Таким образом, результаты данного исследования уже доказали свою эффективность в образовательной практике и могут быть масштабированы на другие дисциплины и образовательные учреждения.

ГЛАВА 1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ

В настоящей главе рассматриваются теоретические и технологические основания проектирования платформы StudyWays. Последовательное изложение включает фиксацию ключевых терминов и понятий (раздел 1.1), анализ существующих решений в области цифрового образования и их ограничений (раздел 1.2), а также обоснование выбора архитектурных решений, включая серверный стек и схему взаимодействия через GraphQL API (раздел 1.3).

Далее будет рассмотрана целесообразность выделения отдельного внутреннего сервиса на базе Express и Prisma (раздел 1.4), проанализированы вопросы аутентификации и авторизации (раздел 1.5), а также сформулированы требования к клиентскому приложению, реализованному с использованием React (раздел 1.6).

1.1 Основные определения

Прежде чем перейти к анализу существующих образовательных платформ, необходимо уточнить основные определения, которые задают рамки обсуждения и обеспечивают терминологическую точность при формулировании требований к проектируемой системе.

Электронное обучение (e-learning) — форма образовательного процесса, основанная на применении цифровых технологий для доставки учебных материалов, организации взаимодействия между участниками и контроля результатов обучения.

Образовательная платформа — программный комплекс, предназначенный для электронного сопровождения обучения, включающий функции хранения и отображения учебных материалов, проведения тестирования, отслеживания успеваемости и взаимодействия с пользователями.

Индивидуальная образовательная траектория — персонализированный маршрут освоения учебного материала, формируемый с учётом уровня подготовки, темпа и интересов обучающегося.

Адаптивное обучение — подход, при котором содержание, способ и темп подачи материала изменяются в зависимости от характеристик и прогресса обучающегося.

Рекомендательная система — компонент, использующий алгоритмы анализа данных и машинного обучения для подбора релевантных материалов или предложений по следующему шагу в обучении.

Образовательный ресурс — структурированная единица учебного материала (видео, текст, изображение, тест и т. д.), используемая в составе одного или нескольких курсов.

Процедурная генерация курса — метод автоматического построения структуры учебного курса на основе запросов, предпочтений и предыдущих действий пользователя с применением алгоритмов искусственного интеллекта.

Уточнение данных понятий позволяет очертить предметную область исследования и зафиксировать критерии анализа образовательных решений. В следующем разделе (1.2) рассматриваются существующие платформы с точки зрения их способности поддерживать индивидуальные образовательные траектории.

1.2 Сравнительный анализ образовательных платформ

В настоящее время на рынке представлено множество платформ электронного обучения, отличающихся архитектурными решениями, функциональностью, уровнем персонализации и целевой аудиторией. В таблице ниже приведён сравнительный анализ наиболее известных решений по ключевым критериям.

Платформа	Адаптивность	Инд. траектории	Рекомендации	Интерактивность	Генерация курсов	Персонализация
Moodle	✗	✗	✗	ограничено	✗	низкий
Coursera / edX	✗ / частично	✗ / частично	частично	средняя	✗	средний
Khan Academy	✓	частично	частично	высокая	✗	средний
Duolingo	✓	✓	✓	высокая	✗	высокий
Stepik	частично	✗	частично	высокая	✗	средний
StudyWays	✓	✓	✓	высокая	✓	высокий

Проведённый анализ показывает, что большинство популярных платформ реализуют персонализацию лишь частично и не поддерживают автоматическое формирование индивидуальных курсов. В отличие от них, платформа StudyWays ориентирована на полное сопровождение индивидуальных образовательных траекторий, что достигается за счёт применения рекомендательных алгоритмов и модульного подхода к структурированию учебного контента.

К числу ключевых отличительных особенностей StudyWays относятся:

- модульная структура контента, предполагающая независимость ресурсов от курсов;
- автоматическая генерация межкурсовых связей на основе тематического совпадения материалов;
- гибкая серверная архитектура, сочетающая Django и Node.js, обеспечивающая масштабируемость;
- реализация интерактивного обучающего тестирования с адаптивной системой подсказок;
- механизм процедурной генерации AI-курсов, основанный на предпочтениях и истории пользователя;
- высокий уровень персонализации, позволяющий учитывать уровень подготовки и интересы обучающегося.

Таким образом, StudyWays демонстрирует более высокий уровень адаптивности и расширяемости по сравнению с традиционными решениями. Это достигается за счёт реализации двухслойной серверной архитектуры: публичного API на базе Django и GraphQL, а также внутреннего сервиса на Express и Prisma. Структура и взаимодействие этих компонентов подробно рассмотрены в разделе 1.3.

1.3 Архитектурные основы серверной части платформы: Django и GraphQL

Одной из ключевых характеристик платформы StudyWays является её открытая архитектура. Система проектировалась как решение, предназначенное не только для использования внутри отдельного учебного заведения, но и как публичный образовательный сервис, ориентированный на

широкий круг пользователей — студентов, преподавателей и разработчиков внешних цифровых образовательных решений.

Серверная часть платформы реализована с использованием веб-фреймворка Django, а взаимодействие с данными организовано посредством языка запросов GraphQL, реализованного через библиотеку Graphene-Django. Такой подход обеспечивает построение унифицированного и расширяемого API, позволяющего внешним системам интегрироваться с платформой и использовать её ресурсы, тестовые задания и метаинформацию в сторонних приложениях и сервисах.

Выбор GraphQL в качестве основного механизма доступа к данным был обусловлен рядом факторов:

- необходимостью предоставить публичное API, обеспечивающее выборочную загрузку только тех данных, которые требуются клиенту;
- стремлением обеспечить простоту интеграции с внешними компонентами, включая рекомендательные алгоритмы, аналитические системы и платформы дистанционного обучения (LMS);
- задачей сохранить контроль над структурой и логикой данных, предоставляя при этом гибкость в работе с ними со стороны клиентов.

Таким образом, архитектурное решение, основанное на связке Django и GraphQL, обеспечивает одновременно масштабируемость, интеграбельность и контролируемость взаимодействия с образовательной платформой.

Далее в главе будут рассмотрены:
- особенности архитектуры серверной части на Django;
- структура и преимущества GraphQL API;
- организация доступа к данным через Django ORM;
- реализация умного поиска по образовательным ресурсам на платформе.

Django

Django – это популярный веб-фреймворк на базе Python. Его ключевыми особенностями являются высокая скорость разработки, особенно для типичных задач, проработанная система безопасности, высокая масштабируемость и большое сообщество разработчиков.

В основе этого фреймворка лежат принцип DRY, который говорит о том, что код не должен дублироваться, а одни и те же концепции описываются в одном

месте, что позволяет избежать размазывания их по всему проекту. Также Django предоставляет готовые решения для интеграции с другими инструментами разработки, такими как разные видами баз данных, системы авторизации и шаблонизаторами. Одним из важных особенностей Django является то что в нем заложен модульный подход, части приложения полностью независимы и могут изменяться и дополняться не оказывая влияние на бизнес логику других модулей.

Стандартное приложение на Django состоит из следующих компонентов:

1. Модели. Модель является описанием данных в приложение и соответствует таблице в базе данных. Django содержит в себе модуль Django ORM, который сверяет состав полей в модели и таблицы в базе данных и в случае отклонения создает миграцию для того, чтобы привести таблицу в базе данных в соответствие с моделью.
2. Панель администрирования. Django изначально содержит в себе готовое решение для управления данными в приложение с минимальной настройкой. Данный модуль позволяет на раннем этапе создания веб-приложение не создавать свою систему администрирования, а воспользоваться готовым решением, содержащим в себе базовый функционал по созданию, редактированию, удалению и валидации данных.
3. Обработчики. Обработчики – это компонент, отвечающий за бизнес-логику приложения, он отвечает за обработку Http-запроса и возврат Http-ответа.
4. Шаблоны. Шаблоны – это специальный инструмент, который позволяет создавать HTML страницы и подставлять в них данные.
5. Маршрутизаторы. В этом слое определяется какому url будет соответствовать какой обработчик, иными словами, определяется правило по которому будет происходить направление запросов на определенный обработчики на основе того, какой путь содержится в запросе

Graphene-Django

Для удобства разработки и минимизации сложности серверного приложения было принято решение реализовать Graph QL api, для этого была использована библиотека Graphene-Django.

Структура приложения с использованием Graphene-Django будет отличаться и состоять из компонентов:

1. Модели. Модели также описывают данные приложения и используются для таблиц в базе данных, но теперь на основе них еще и создаются типы для Graph QL схемы.
2. Административная панель. Этот модуль остается без изменений.
3. Обработчики. Поскольку всю бизнес логику по получению и изменению данных в Graph QL определяет схема, то данный слой полностью отсутствует.
4. Шаблоны. Поскольку Graph QL запросы не подразумевают передачу HTML страниц, то данный слой отсутствует.
5. Маршрутизаторы. Graph QL использует единый url для всех запросов /graphql, поэтому данный слой также отсутствует.

Таким образом Graphene-Django позволяет значительно сократить число слоев и сложность серверного веб-приложения, позволяя создать полноценное GraphQL api на основе моделей и механизмов Django ORM.

Graph QL – это язык запросов, который является альтернативой более популярной концепции REST. Данный подход позволяет клиенту самому описывать схему данных, которые он хочет получить от сервера. Graph QL позволяет получать только необходимые поля и использовать всего один маршрут /graphql для всех запросов.

Основные концепции:

1. Схема. В схеме описываются типы всех сущностей и связи между ними.
2. Запросы. Запросы используются для получения данных клиентом. В них клиент передает схему того какие сущности и поля он ожидает получить от сервера. Сервер же в свою очередь производит запросы к базе данных в формате SQL и затем объединяет и структурирует данные согласно той схеме, которая была передана клиентом.
3. Мутации. Мутации позволяют клиенту изменять данные на сервере, производя создание, редактирование и удаление данных.
4. Подписки. Данный механизм используется для получения данных и их обновления в реальном времени.

Основные плюсы использования Graph QL:

1. Гибкость в запросах данных. Graph QL позволяет клиентам самим определять какие поля и связанные сущности будут получены с сервера, благодаря этому можно одним запросом получить все необходимые данные, а не совершать несколько подряд идущих запросов к разным сущностям.
2. Упрощение серверного приложения. Поскольку всю логику по получению данных из базы данных и преобразование в нужный клиенту формат производит движок Graphene-Django, то разработка серверного приложения упрощается до формирования моделей Django, после чего Graphene-Django создаст Graph QL схему на основе текущего состояния моделей, а Django ORM произведет синхронизацию со схемой базы данных.
3. Глубокая типизация и валидация. Поскольку в Graph QL схеме определены типы всех сущностей, то это позволяет автоматически получить строгую типизацию и валидацию данных на стороне клиента. Таким образом этот подход позволяет экономить время при разработки, необходимое на описание типов данных и валидации запросов от сервера.
4. Простата расширения api. При добавление новых полей к сущности нет необходимости проверять что эти поля не передается в ранее созданные запросы. Это делает развитие api более безопасным.
5. Единый url. Все запросы направляются на один адрес, что значительно упрощает маршрутизацию.

Основные ограничения, которые накладывает GraphQL:

1. Уязвимость к нагрузочным атакам. Поскольку GraphQL позволяет создавать запросы с любым уровнем вложенности, то это открывает возможность злоумышленникам создавать запросы, которые будут создавать огромную нагрузку на инфраструктуры.
2. Сложность сокрытия чувствительной информации. GraphQL требует очень детальной настройки доступов к данным, которую нужно производить для каждого узла схемы индивидуально, а в некоторых случаях и для каждого поля в этих узлах, это может потребовать очень значительного времени и сопровождается высоким уровнем рисков.
3. Сложность кэширования. Поскольку GraphQL использует один url для всех запросов и использует HTTP метод POST, то классические решения для кэширования невозможно применить.

4. Сложность логирования. Для анализа ошибок в логах необходимо вручную анализировать тело запроса, это требует значительно больше времени, чем анализ логов классических HTTP запросов.
5. Высокая нагрузка на сервер. Graphene-Django для обработки каждого узла в запросе клиента делает отдельный запрос в базу данных, это создает огромную нагрузку, тогда как при стандартном подходе те же данные можно было бы получить одним запросом.
6. Повышение уровня сложность клиентского приложения. GraphQL по сути позволяет на клиентской стороне работать с базой данных, но вместо синтаксиса SQL использовать синтаксис GraphQL, в свою очередь это приводит к тому, что вся бизнес-логика приложения сосредотачивается на клиентской стороне приводя к тому что frontend разработчик должен решать backend задачи в условиях когда вместо мощного языка SQL он использует сильно ограниченный язык GraphQL.

Таким образом выбор Graphene-Django позволяет без глубоких знаний в области серверной разработки быстро создавать GraphQL api на основе обьектного представления базы данных Django, но создает множество рисков при работе с чувствительными данными.

Работа с базой данных

Для работы с базой данных Graphene-Django использует обьектное представление базы данных, то есть Django ORM. Django ORM – это механизм, который позволяет взаимодействовать с базой данных так, словно таблицы в базе данных - это обыкновенные обьекты на языке Python.

Модели в Django описывают таблицы в базе данных. Каждый атрибут модели соответствует определенному столбцу в таблице и позволяет описывать его название, тип и дополнительную информацию. Модели позволяют описывать связи между друг другом по принципам один к одному, один ко многим и многие ко многим, автоматически преобразовываясь во внешние ключи или таблицы связей.

Для того, чтобы синхронизовать схему базы данных и ее объектную модель используется механизм создания миграций. Этот механизм обнаруживает, различая между схемой базы данных и объектным представлением и на основе этой разности генерирует миграционный SQL файл, после успешного

применения которого схема базы данных становится идентичной обьектному представлению. Такой подход позволяет управлять версиями схемы базы данных на всем пути разработки серверного приложения и исключает возможность формирования SQL запроса механизмами Django ORM в котором будут запрошены поля или таблицы, которые отсутствуют в базе данных.

Django ORM позволяет без написания SQL скриптов производить с моделями базовые операции такие как создание, обновление, удаление и чтение, а также запрашивать данные из связанных таблиц.

Важным свойством Django ORM является то, что он позволяет работать абсолютно единообразно с разными СУБД, такими как PostgreSQL, MySQL, SQLite, причем переход на любую из поддерживаемых СУБД не потребует изменения бизнес-логики.

Django ORM автоматически производит экранирование всех передаваемых параметров, это позволяет значительно снизить риск SQL- инъекций.

Умный поиск информации

Умный поиск – это подход к поиску информации когда ответы выдаются не только на основе совпадения слов в запросе пользователя и имеющейся информации, но также учитываются падежи, синонимы и общий контекст запроса. Этот подход позволяет давать пользователю более точные результаты поиска и в конченом случае улучшить его пользовательский опыт.

Стандартный подход к поиску на основе SQL команд LIKE или contains не может быть использован поскольку обладает рядом минусов:

1) Сложность поиска внутри базы данных возрастает линейно в зависимости от обьема базы данных, что приведет к тому, что операция поиска для больших обьемов данных становится слишком дорогостоящей.
2) Поиск чувствителен к регистру слова, его падежу и форме, что значительно понижает качество поиска для таких языков, как русский.
3) Отсутствует система оценки релевантности, все совпадения слов равны по значениям.
4) В поиске учитываются союзы и служебные частицы, не несущие смысловой нагрузки.

Основные плюсы использования умного поиска:

1) Умный поиск способен учитывать регистр слов, падежи, времена, склонения и в некоторых случаях исправлять опечатки.
2) Для умного поиска существуют индексы для базы данных, которые позволяют значительно ускорить поиск информации.
3) Умный поиск позволяет учитывать релевантность информации и в зависимости от этого ранжировать результаты поиска.

Умный поиск на основе полнотекстовых индексов

Для реализации умного поиска без использования сторонних сервисов таких как Elasticsearch была использован механизм полнотекстовых индексов для базы данных Postgres и Django ORM.

Плюсы использования полнотекстовых индексов:

1) Высокая скорость поиска, в том числе для больших обьемов текста.
2) Поддержка лингвистических моделей для русского языка.
3) Возможность индексации данных и хранение этих индексов в формате tsvector.
4) Возможность использовать вместе с фильтрами и сортировками, реализуемыми через Django ORM.

Алгоритм работы полнотекстового поиска:

1) При помощи лингвистической модели для русского языка данные очищаются от стоп слов, знаков препинания, слова приводятся к нормальной форме.
2) Создаётся tsvector, представляющий из себя набор «слово + позиция».
3) Пользовательский запрос превращается в tsquery.
4) Осуществляется поиск на основе сравнения tsvector и tsquery
5) В случае если был создан GIN-индекс, он используется для того, чтобы не проводить сканирование всей таблицы.

Тем не менее часть операций — например, подготовка персонализированных AI-курсов — требует сложной бизнес-логики, которую нецелесообразно выносить в публичное GraphQL API. Для изоляции этих процессов во втором

подразделе показано, как внутренний сервис на Express и Prisma дополняет открытую архитектуру.

1.4 Реализация внутренней бизнес-логики с использованием Express.js и Prisma ORM

Платформа StudyWays проектируется как открытая образовательная экосистема, предоставляющая публичное API для получения образовательных ресурсов и сопровождения индивидуального обучения. Однако, по мере развития платформы стало очевидно, что ряд внутренних процессов требует реализации сложной, специфичной бизнес-логики, которая не должна быть частью открытого интерфейса взаимодействия с клиентом.

В целях разделения ответственности и архитектурной чистоты было принято решение выделить внутреннюю бизнес-логику в отдельное серверное приложение, реализованное с использованием платформы Node.js, языка TypeScript, фреймворка Express.js и ORM-библиотеки Prisma.

Такое решение позволило:
- обеспечить чёткое разграничение между публичным и внутренним API;
- использовать раздельную архитектуру для обработки пользовательских и системных запросов;
- вынести бизнес-логику за пределы GraphQL, что существенно упростило архитектуру клиентского кода;
- разработать и сопровождать серверную логику с учётом всех требований безопасности, масштабируемости и расширяемости.

Таким образом, второе серверное приложение является закрытым внутренним компонентом StudyWays, предназначенным исключительно для реализации внутренних процессов платформы, недоступных через публичное GraphQL API, и представляет собой отдельный уровень логической обработки данных, необходимый для полноценной работы образовательной среды.

Что такое Node js

Node js – это среда исполнения JavaScript кода вне браузера, построенная на движке V8 от компании Google. Благодаря Node js открывается возможность

использовать JavaScript не только для клиентских, но и для серверных приложения.

TypeScript – это язык построенный поверх JavaScript, добавляющий статическую типизацию, что значительно упрощает создание и сопровождение крупных проектов, а также ускоряет разработку за счет глубокой интеграции со средами разработки, позволяющей получать подсказки и проверку типов во время написания программы до выполнения компиляции.

Express js – это высокоуровневый фреймворк на Node js, позволяющий создавать серверные приложения на языках JavaScript и TypeScript. Ключевой особенностью Express js является то, что он не диктует структуру того, как должен быть организован проект, что позволяет очень быстро и при минимальных усилиях создать серверное приложение по своей структуре оптимальное для проекта среднего размера.

Работа с базой данных во втором серверном приложении

Для работы с базой данных во втором серверном приложение была использована библиотека Prisma ORM. Prisma – это ORM библиотека, разработанная специально для Node js экосистемы, в основе ее принципов лежат статическая типизация и глубокая интеграция с TypeScript, высокая производительность и удобство работы с базой данных.

Prisma js отличается от классических ORM библиотек наличием декларативного подхода к описанию схемы базы данных, автогенерацией типизированных методов для доступа к данным и двустороннюю синхронизацию между схемой и кодом.

Единая точка правды в проектах с использованием Prisma ORM является Prisma schema, это специальный файл, в котором на языке prisma описана структура базы данных. В этом файле определяются структуры таблиц, типы полей и связи между таблицами, а также настраиваются генераторы кода, например для TypeScript и подключение к базе данных.

Prisma поддерживает два подхода к синхронизации модели и базы данных.

1. Миграции. Это классический поход, используемый например в Django ORM, он заключается в том что после внесения изменений в prisma schema, производится запуск команды, которая анализирует произведенные изменения и на основе них создает SQL миграционный

файл, после чего данный файл выполняется по отношению к базе данных

2. Интроспекция. Данный подход позволяет произвести синхронизацию модели и базы данных в обратном направление и является важнейшим механизмом, позволяющим использовать Prisma ORM в проекте с двумя серверными приложениями. Принцип этого подхода заключается в том, что Prisma анализирует базу данных и на основе нее создает файл prisma schema. Этот подход позволяет значительно сэкономить время в условиях когда Prisma используется в проектах с уже существующей базой данных.

Prisma ORM позволяет легко создать серверное приложение для уже существующей базы данных без изменения ее схему, что позволяет создать второе серверное приложение рядом с уже существующим без нарушения работы изначального.

Таким образом, закрытый сервис обеспечивает гибкость и безопасность обработки внутренних процессов, сохраняя при этом чистоту публичного API. Следующей задачей становится защита всей системы с точки зрения аутентификации и авторизации, что рассмотрено в разделе 1.5.

1.5 Аунтификация и авторизация.

Для реализации аунтификации и авторизации был использован сервис Auth0. Auth0 – это облачная платформа, предоставляющая аунтификацию и авторизацию как сервис. Данный сервис предоставляет:

1. Настраиваемые формы регистрации и входа.
2. Подключение провайдеров авторизации, таких как Google и VK.
3. Работу с JWT токенами.
4. Управление правами доступа и пользователями.
5. Встроенный механизм подтверждения через почту.
6. Поддержку корпоративных провайдеров авторизации, таких как LDAP и SAML.
7. Поддержку двухфакторной аунтификации.
8. Использование токенов для API и клиентской стороны.
9. Возможность внедрять собственную логику.

Auth0 следует лучшим практикам по обеспечению безопасности:

1. Авторизационные токены хранятся в cookies с использованием флагов Secure и HttpOnly.
2. Работа происходит по Https.
3. Поддерживается ротация ключей.
4. Имеется настройка времени жизни токенов.
5. В сервис встроена защита от таких типов атак как CSRF, XSS, Brute-force и др.

Важной особенностью Auth0 является то, что данный сервис имеет готовый к использованию SDK для большинства популярных языков и платформ, что позволило использовать единый сервис авторизации и аунтификации для обоих серверных приложений и клиентского приложения.

Для интеграции Auth0 и Node js используется официальный SDK и middleware. Для интеграции с Django нет официального SDK, поэтому используется библиотека PyJWT.

Итак, единая связка Auth0 + JWT обеспечивает согласованную схему авторизации для обоих серверных приложений и клиентского SPA. Права доступа проверяются симметрично, а токен формата JWT передаётся в каждом запросе к GraphQL-API или Logic-API. С надёжным контуром безопасности можно перейти к устройству клиентской части, через которую пользователь взаимодействует со всей платформой.

1.6 Клиентское приложение.

Основой клиентского приложения является библиотека React, эта библиотека позволяет создавать динамические и отзывчивые frontend приложения. В этой библиотеки реализован компонентный подход и однонаправленный поток данных, а также за счет виртуального DOM дерева и пакетной системы обновления реального DOM дерева достигается высокая производительность.

Основный концепции React

1. Компонентный подход.
   React реализует идею компонентов, которые позволяют переиспользовать и изолировать блоки из которых строится пользовательский интерфейс. Внутри компонентов можно реализовать бизнес-логику и отображение интерфейса, вкладывать компоненты в

друг-друга, передавать компонентам входные параметры и реагировать на события.

2. Виртуальное DOM дерево.
   Операции с реальным DOM деревом браузера очень затратны по производительности, поэтому для того чтобы приложения были плавными и изменения на странице происходили быстро в React реализована концепция виртуального DOM дерева. Виртуальное DOM дерево – это облегченная копия реального DOM дерева, но хранящаяся в виде java script обьекта. При перерендере компонента, React создает новое виртуальное DOM дерево, сравнивает с тем, каким оно было до ренедера и если обнаруживается различие, то производится пакетное обновление реального DOM дерева, чтобы оно пришло в соответствие с обновленным виртуальным DOM деревом.
3. Однонаправленный поток данных.
   Компоненты в React могут принимать входящие параметры и реагировать на их изменение, это позволяет передавать данные от родительского компонента к дочерним, что реализует однонаправленный поток данных, в свою очередь дочерние компоненты могут передавать родительским события благодаря механизму передачи call-back функций в качестве параметра.
4. Хуки.
   Для хранения данных и реагирования на события в React используется механизм хуков. Хук – специальная функция, для работы с которой может влиять на жизненный цикл компонента. Хук useState позволяет хранить данные и при изменение их перерисовывать компнент, хук useEffect позволяет отслеживать изменения данных или вызовы перерисовки компонента, хуки useRef, useCallback, useMemo позволяют оптимизировать повторные отрисовки компонентов.
5. Подходы к стилизации.
   React позволяет разработчику самому выбирать один или несколько подходов к стилизации, начиная от написания CSS файлов, заканчивая CSS in JS, styled-components, emotion, Tailwind CSS и MUI.
6. Работа с глобальным состоянием.
   React сам по себе не имеет решения для работы с глобальным состоянием, но это реализуется благодаря таким библиотекам как Redux и MobX, реализующие концепцию глобальных обьектов, а также существует относительно новые подходы на основе менеджеров данных React Query и SWR.
7. Серверный рендеринг.

Для оптимизации SEO и быстрого открытия страниц очень важно, чтобы у технологии для клиентского приложения была возможность быть отрендеренной на серверной стороне, чтобы передать клиенту готовую HTML страницу, на которую в последствие будет добавлена интерактивность благодаря процессу гидрации, для React такой технологией является фреймворк Next js, позволяющей произвонить серверный рендеринг страниц и серверную генерацию статических страниц.

Библиотека компонентов для пользовательского интерфейса

Для реализации интерфейса на React была использована библиотека компонентов MUI. Material UI – это популярная open-source библиотека готовых компонентов для создания пользовательского интерфейса, основанная на Material Design – стандарте компонентов пользовательского интерфейса от компании Google. Данная библиотека поставляет компоненты для управления, такие как кнопки, чек боксы, слайдеры, текстовые поля, компоненты для позиционирования элементов, таблицы, списки, диалоговые окна, при этом все компоненты поддерживают глубокую кастомизацию. MUI реализует механизм Theme API, позволяя задавать глобальные настройки, переопределяя цвета, отступы, скругления. Для стилизации компонентов по месту использования в MUI разработан механизм sx параметров, позволяющим задавать значения учитывающие настройки глобальной темы. MUI поддерживает адаптивную верстку через систему breakpoints, позволяющую задавать стили отдельно для разных размеров экрана через sx параметр. MUI полностью типизирован через TypeScript, поэтому поддерживает проверку типов на этапе разработки и предоставляет подсказки в IDE. Таким образом MUI представляет собой надежную и гибкую библиотеку для создания пользовательского интерфейса, идеально подходящую для проектов среднего размера.

Библиотека для управления состоянием пользовательского интерфейса

React предоставляет решение для управлением состоянием приложения через хук useState, но при применение такого подхода в средних и больших приложениях возникает ряд проблем связанных с производительностью и резким возрастанием сложности кода за счет смешения кода пользовательского интерфейса и бизнес логики, поэтому для управления состоянием в средних и больших приложениях используются менеджеры

состояния. MobX – это одна из самых популярных open-source библиотек для управления состоянием приложения, позволяющая декларативно описывать бизнес логику. MobX позволяет создать наблюдаемые переменные при изменение которых интерфейс будет автоматически обновлен в тех местах, где используются эти переменные. В MobX существует механизм подписки на изменение переменной и вызова на основе этого определенных событий, это позволяет, например, реализовывать системы автоматического сохранения данных в редакторах. Таким образом MobX позволяет выносить всю бизнес-логику в отдельные обьекты, разделяя код относящийся к интерфейсу и логики приложения.

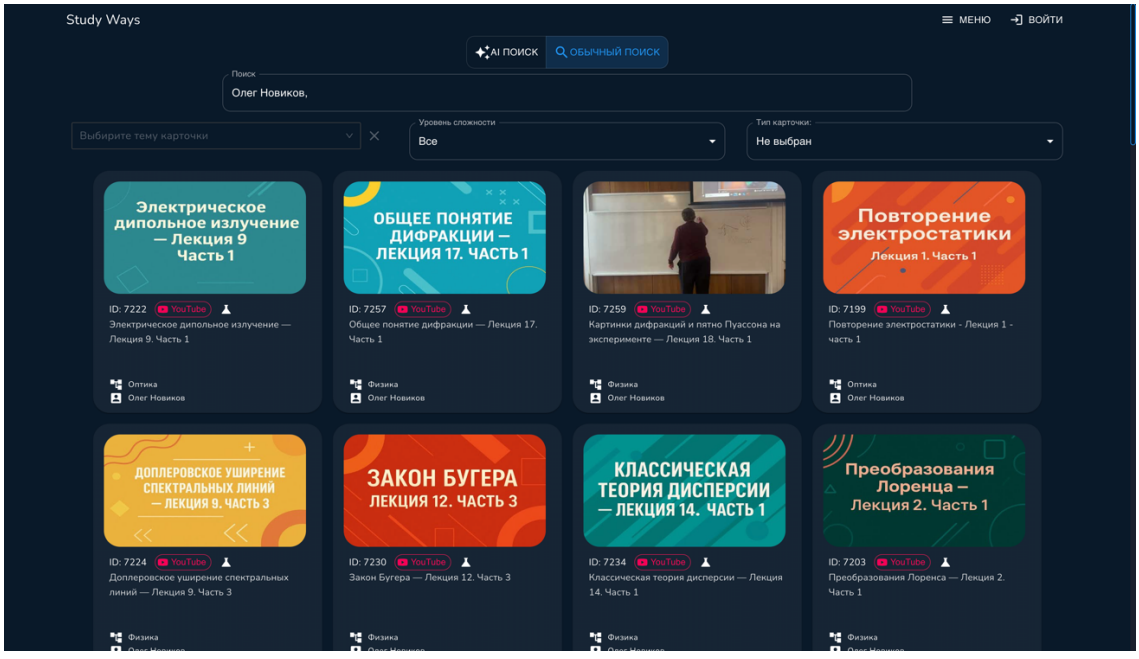
ГЛАВА 2. РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА

В данной главе подробно рассматриваются модули, составляющие программную платформу StudyWays. Каждый модуль отвечает за выполнение отдельной функции в рамках образовательной системы, обеспечивая взаимодействие пользователей с контентом, прохождение курсов и тестов, а также сбор и обработку информации, необходимой для построения индивидуальных образовательных траекторий.

Реализация платформы опирается на модульный принцип: каждый компонент выполняет свою задачу, но при этом тесно интегрирован с остальными модулями, что обеспечивает гибкость, масштабируемость и персонализацию образовательного процесса. В описании каждого модуля уделено внимание следующим аспектам:

- целям и назначению модуля;
- пользовательскому интерфейсу;
- архитектурным и техническим решениям;
- механизмам персонализации обучения;
- взаимодействию с другими модулями платформы.

2.1 Модуль выбор образовательных ресурсов



1. Назначение и цели модуля

Модуль обеспечивает пользователю доступ к образовательным ресурсам, размещённым на платформе, с возможностью гибкого поиска и фильтрации. Основной целью является облегчение навигации по контенту и предоставление пользователю персонализированных рекомендаций, соответствующих его интересам и уровню подготовки.

2. Описание интерфейса и компонентов

Интерфейс модуля включает:

- поле ввода текста для поиска по заголовкам и описаниям ресурсов;
- переключатель между двумя режимами поиска: на основе искусственного интеллекта и полнотекстового поиска;
- фильтры по теме, уровню сложности и типу контента (видео, изображение, ссылка);
- карточки найденных ресурсов с отображением:
  - изображения;
  - ID и названия ресурса;
  - типа контента;
  - уровня сложности;
  - темы и автора.

Для выбора темы используется компонент с древовидным отображением, позволяющий включать вложенные категории.

3. Техническая реализация

Модуль реализован на клиентской части с использованием React и библиотеки MUI. Два режима поиска поддерживаются следующим образом:

- Поиск на основе ИИ реализован через внешнюю рекомендательную систему Recombee. Система анализирует поведение пользователя и предсказывает наиболее релевантные ресурсы.
- Поиск по полнотекстовым индексам работает через PostgreSQL и механизм tsvector, интегрированный в Django ORM. Такой подход обеспечивает быстрое и точное извлечение информации по ключевым словам с учётом морфологии русского языка.

Межмодульная интеграция позволяет использовать этот компонент также в редакторе ресурсов при их создании и редактировании.

4. Механизмы персонализации

Модуль интегрирован с рекомендательной системой, которая:

- предлагает релевантные ресурсы ещё до ввода текста;
- учитывает историю взаимодействия пользователя с платформой;
- использует поведение других пользователей с похожими интересами для улучшения рекомендаций;
- обучается в реальном времени на предпочтениях конкретного пользователя.

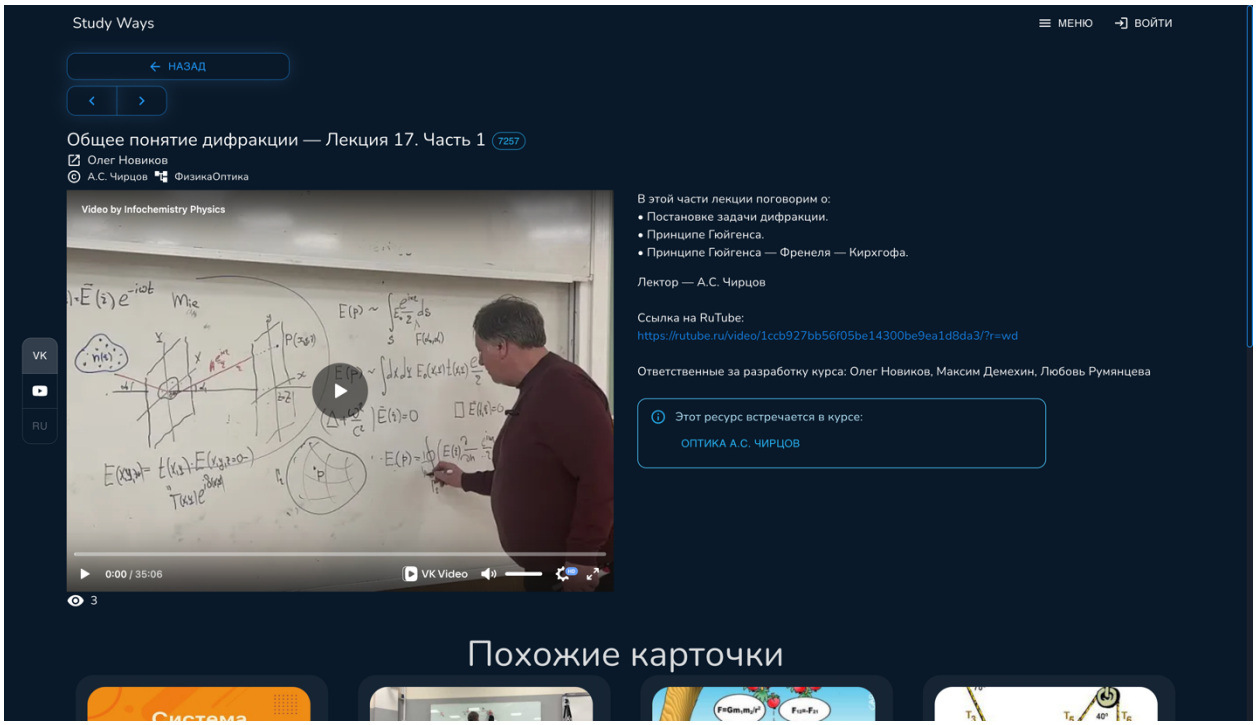
При использовании полнотекстового поиска применяется фильтрация по уровню и тематике, что также способствует индивидуализации выдачи.

5. Взаимодействие с другими модулями

Модуль тесно связан с:

- модулем отображения ресурса (2.2) — по нажатию на карточку осуществляется переход к полному содержимому ресурса;
- редактором ресурсов (2.7) — используется в интерфейсе создания и редактирования контента;
- модулем курсов (2.3) — обеспечивает поиск и добавление ресурсов в курсы.

2.2 Модуль отображения ресурса



1. Назначение и цели модуля

Модуль предназначен для отображения полного содержимого образовательного ресурса. Он предоставляет пользователю доступ к медиа-материалам, информации об авторе, а также к связанным курсам и рекомендациям. Целью является не только демонстрация контента, но и обеспечение его контекстного включения в учебный процесс за счёт связей с курсами и тестами.

2. Описание интерфейса и компонентов

Интерфейс страницы отображения ресурса включает следующие элементы:

- название и ID ресурса;
- имя автора и тема;
- изображение или видео с возможностью переключения между хостингами (YouTube, VK и др.);
- текстовое описание;
- блок с перечнем курсов, в которых содержится данный ресурс, с возможностью перехода к нужному фрагменту;
- блок «Похожие ресурсы» — автоматически сгенерированный на основе рекомендаций.

Также реализована система логирования пользовательской активности: фиксируется время просмотра ресурса и последующий переход, что используется для анализа интересов пользователя.

3. Техническая реализация

Модуль реализован на React с использованием MUI для визуальных компонентов. Видео поддерживаются через встроенные плееры с поддержкой нескольких источников. Похожие ресурсы формируются через Recombee API в реальном времени и загружаются асинхронно.

Информация о переходах и длительности взаимодействия с ресурсом отправляется на внутренний сервер (Express + Prisma), где агрегируется для последующего использования в обучении рекомендательной системы.

4. Механизмы персонализации

Для повышения качества рекомендаций модуль:

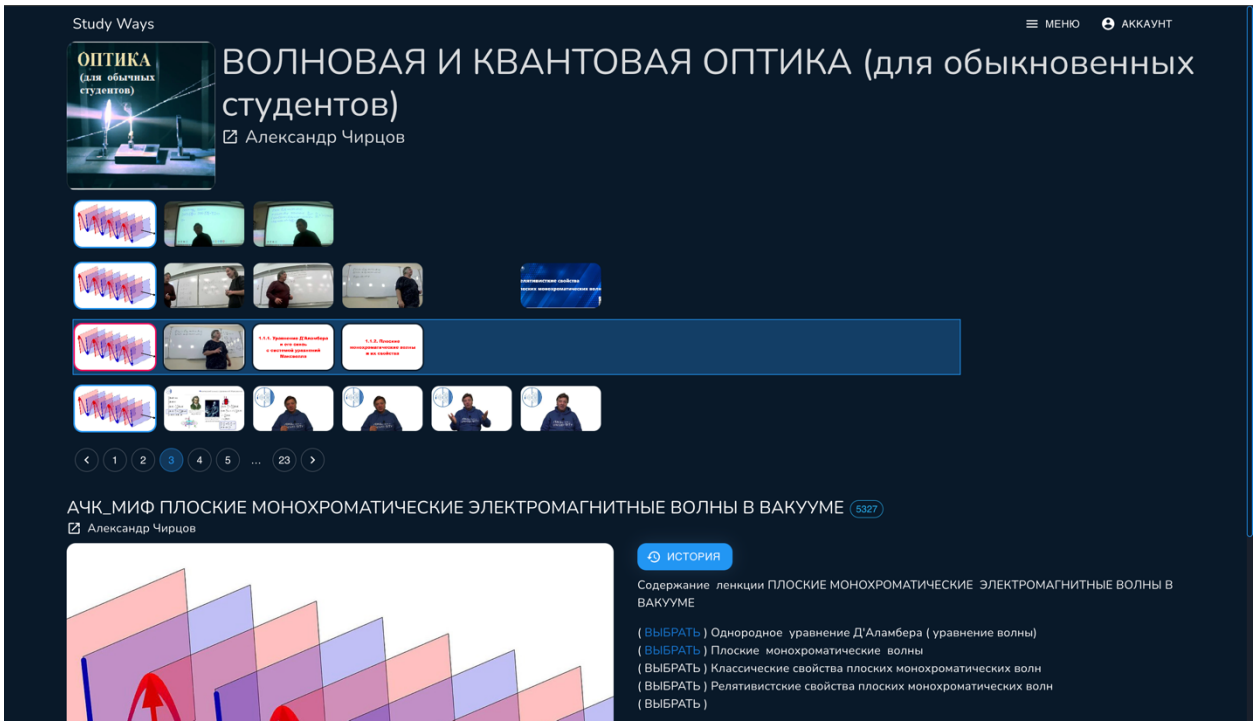
- отслеживает поведение пользователя (время просмотра, клики, переходы);
- использует эти данные для корректировки рекомендаций в будущем;
- отображает похожие ресурсы, релевантные интересам пользователя;
- поддерживает персонализированные переходы к связанным курсам и тестам, формируя индивидуальные маршруты обучения.

5. Взаимодействие с другими модулями

Модуль активно взаимодействует с:

- модулем курсов (2.3) — используется при отображении курса для показа конкретных ресурсов;
- модулем тестов (2.5) — обеспечивает переход к тестам, ассоциированным с данным ресурсом;
- модулем выбора ресурсов (2.1) — связан через карточки, ведущие на страницу текущего модуля;
- редактором ресурсов (2.7) — для пользователей с правами преподавателя или администратора доступна возможность перейти к редактированию ресурса.

2.3 Модуль курсов



1. Назначение и цели модуля

Модуль курсов предоставляет пользователю возможность проходить структурированные образовательные курсы, собранные из отдельных ресурсов. Основная цель — предоставить удобную навигацию по многоуровневым курсам, а также обеспечить возможность адаптации материала под уровень подготовки обучающегося.

2. Описание интерфейса и компонентов

Интерфейс страницы курса включает:

- название и изображение курса;
- информацию об авторе;
- визуальное представление структуры курса в виде двумерной сетки, где каждый элемент соответствует образовательному ресурсу;
- выделенную «основную линию» курса — ключевой маршрут изучения материала, отмеченный синим цветом;
- компонент для переключения между страницами курса;
- блок отображения выбранного ресурса.

Ресурсы в курсе организованы по принципу многоуровневости, что позволяет пользователю видеть как основную последовательность, так и альтернативные пути изучения.

3. Техническая реализация

Каждый курс хранится в базе данных в формате JSON. Такая структура позволяет:

- представлять курс как набор страниц, каждая из которых содержит двумерную сетку ячеек;
- в каждой ячейке указывать один или несколько ресурсов;
- при необходимости — вставлять ссылки на конкретные позиции в других курсах, обеспечивая межкурсовую навигацию и переиспользование материалов.

На клиентской стороне курс визуализируется с помощью React и библиотеки MUI. Для отрисовки используется двумерная графическая сетка, элементы которой динамически подгружаются на основе содержимого JSON.

4. Механизмы персонализации

Модуль предоставляет обучающемуся выбор уровня сложности материалов в рамках одного и того же курса. Кроме того:

- пользователь может самостоятельно выбирать траекторию прохождения: идти по основной линии или исследовать альтернативные ресурсы;
- рекомендательная система предлагает другие курсы, в которых используется изучаемый материал;
- платформа запоминает прогресс прохождения и может адаптировать рекомендации в зависимости от успешности освоения предыдущих блоков.

Таким образом, модуль способствует формированию персонализированной образовательной траектории.

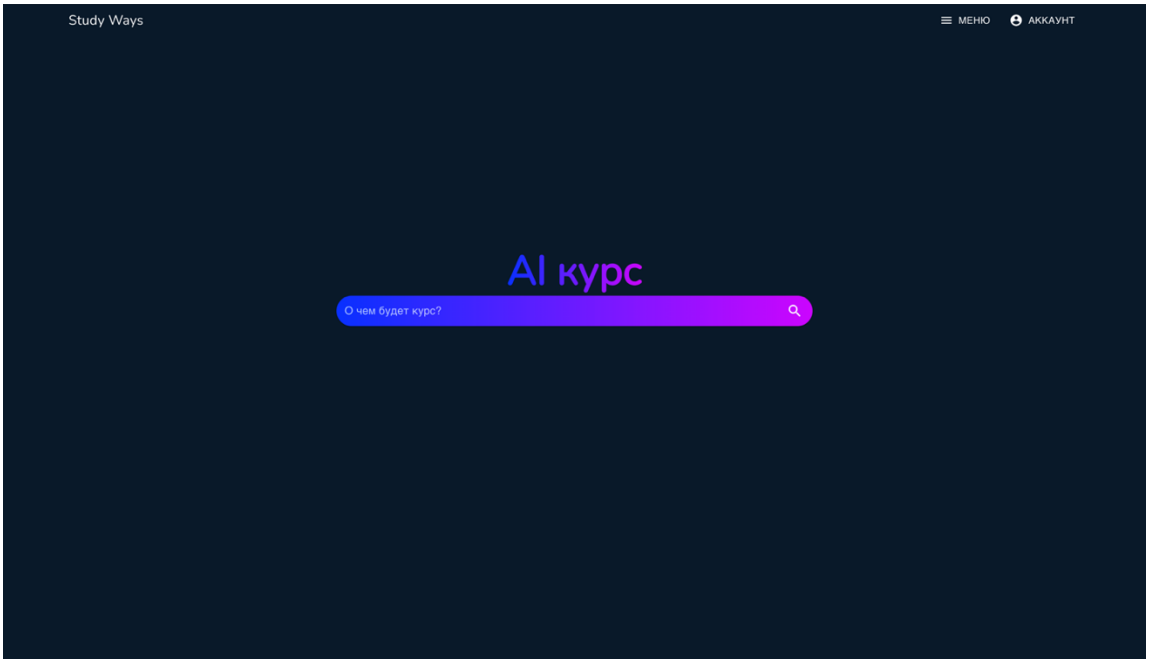
5. Взаимодействие с другими модулями

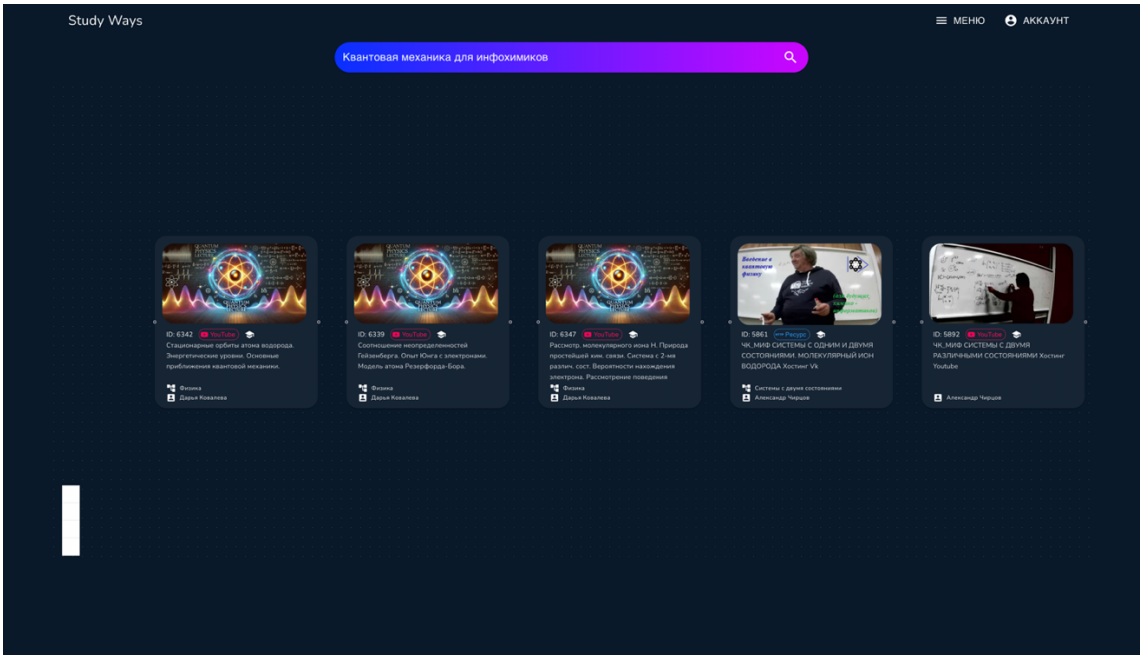
Модуль курсов связан с:

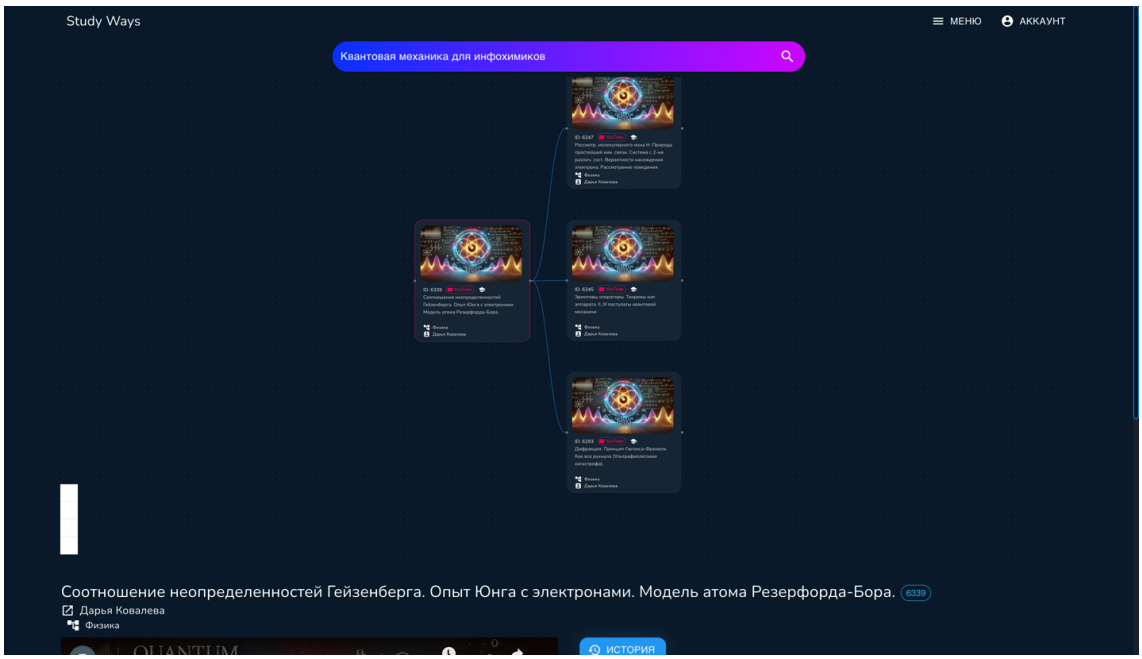
- модулем отображения ресурса (2.2) — ресурсы отображаются в отдельном блоке или по клику открываются полностью;

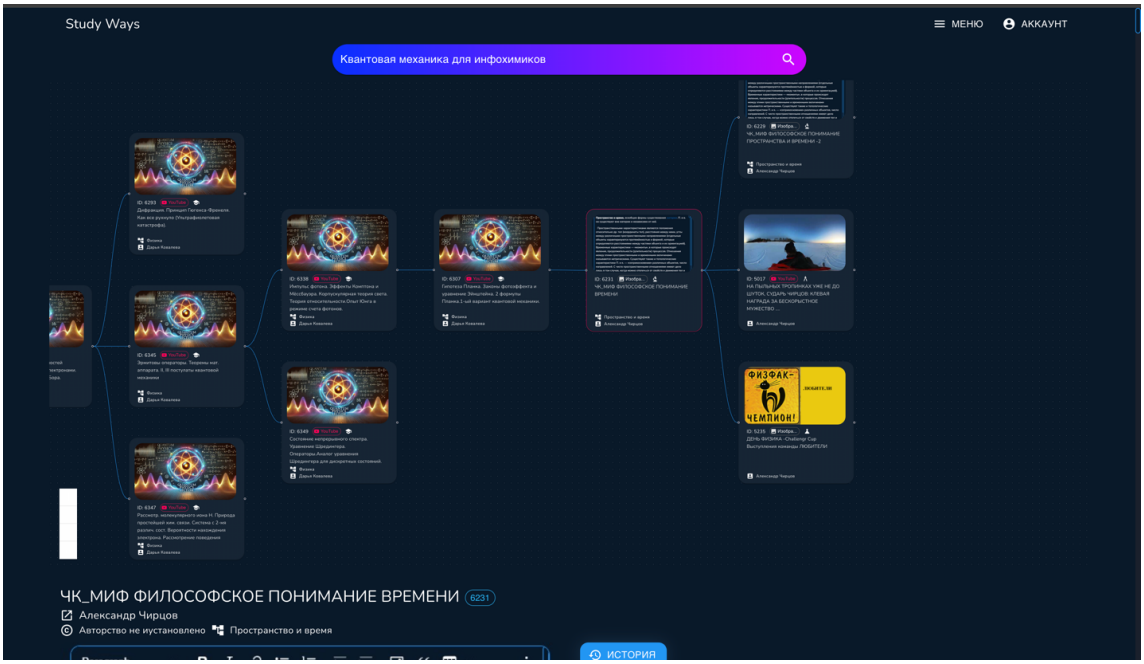
- редактором курсов (2.8) — используется преподавателями и администраторами для создания и изменения структуры курсов;
- модулем тестов (2.5) — при наличии тестов, привязанных к ресурсам, пользователь может переходить к ним напрямую;
- модулем AI-курсов (2.4) — предоставляет альтернативную форму прохождения материала на основе процедурной генерации.

2.4 Модуль курсов на основе искусственного интеллекта.









1. Назначение и цели модуля

Данный модуль предоставляет пользователю возможность построения персонального образовательного курса в интерактивном режиме. Система формирует курс на основе интересов, предыдущих взаимодействий и предпочтений пользователя, используя рекомендательные алгоритмы и механизмы процедурной генерации. Цель модуля — предоставить обучающемуся свободу выбора маршрута, сохраняя при этом образовательную целостность.

2. Описание интерфейса и компонентов

Интерфейс включает:

- строку поиска, в которой пользователь вводит тему интересующего его курса;
- индикатор загрузки — отображается в процессе генерации курса;
- двумерную плоскость с узлами-ресурсами (граф курса);
- пять стартовых ресурсов, сгенерированных системой на основе запроса и предпочтений пользователя;
- визуализацию выбранного ресурса в нижней части экрана;
- интерактивный механизм выбора следующего шага: при клике на ресурс система подбирает три наиболее релевантных следующих ресурса и отображает их как дочерние узлы;
- возможность возврата к предыдущим узлам, перехода по альтернативным ветвям или повторного изучения материалов.

Навигация по курсу реализуется в виде графа, в котором пользователь сам формирует последовательность шагов, исследуя тему в глубину или в ширину.

3. Техническая реализация

Модуль реализован на React, визуализация графа построена с использованием canvas/WebGL-библиотеки для производительного отображения узлов и связей. Компоненты React управляют состоянием текущего узла и истории переходов.

Построение курса выполняется с использованием внешней рекомендательной системы Recombee, которая:

- анализирует введённую тему;
- учитывает историю пользователя;
- применяет поведенческие и контентные фильтры;
- возвращает релевантные ресурсы с учётом уровня подготовки и предпочтений.

Все переходы, выборы и время взаимодействия с ресурсами логируются и могут использоваться как для повторной персонализации, так и для аналитики.

4. Механизмы персонализации

AI-курсы являются ключевым элементом персонализации в платформе и реализуют её в следующих аспектах:

- поисковая адаптация: стартовые ресурсы подбираются индивидуально;
- рекомендательная генерация: на каждом шаге система предлагает только те ресурсы, которые соответствуют поведению и интересам пользователя;
- учёт глубины погружения: пользователь может остановиться на базовом уровне или пойти дальше, строя ветви курса;
- обратная связь: анализ поведения пользователя (время просмотра, клики, возвраты) влияет на дальнейшие предложения.

Таким образом, пользователь самостоятельно формирует курс «на лету», взаимодействуя с системой, как с интеллектуальным наставником.

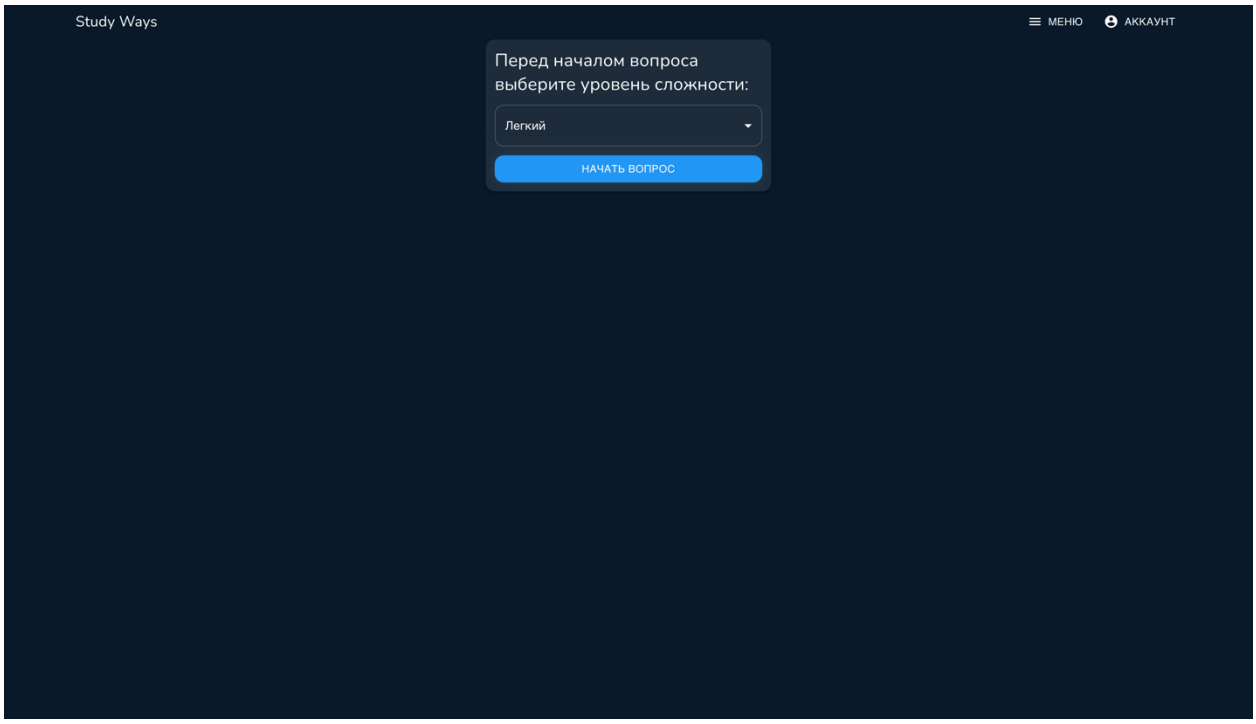
5. Взаимодействие с другими модулями

AI-модуль интегрирован с:

- модулем отображения ресурса (2.2) — для отображения выбранного материала;
- модулем тестов (2.5) — ресурсы могут содержать привязанные тесты;
- модулем просмотра результатов (2.6) — анализ успешности может влиять на рекомендации;
- поисковой системой (2.1) — строка поиска темы курса интегрирована с AI-поиском.

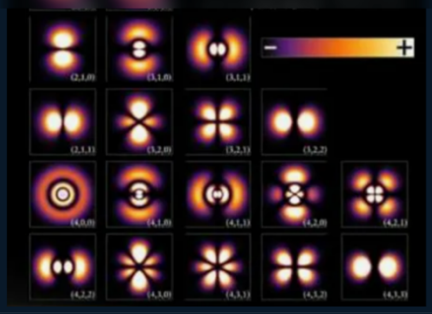
Результатом работы AI-модуля становится индивидуальный путь обучения, который может быть повторно воспроизведён, экспортирован или сохранён для продолжения.

2.5 Модуль тестов.



Study Ways

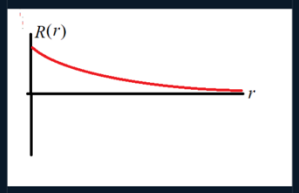
МЕНЮ АККАУНТ



Вопрос

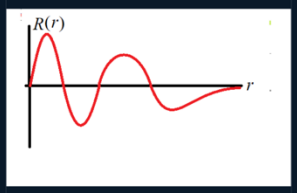
Из предлагаемых вам вариантов выберите правильные графики радиальных частей волновых функций стационарных состояний электрона в атоме водорода. Учтите, что для осложнения пользованием услугами gpt-чатом графики рисовались от руки без соблюдения относительных величин максимумов и расстояний между нулями волновых функций.

ПРОВЕРИТЬ СДАТЬСЯ



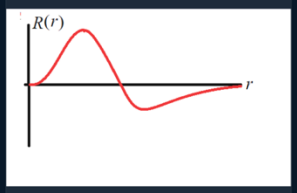
Радиальная часть волновой функции 1s - состояния электрона в атоме водорода

Пометки для себя (не учитываются при проверке)



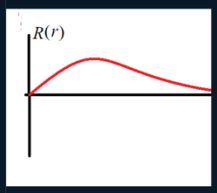
Радиальная часть волновой функции 2p - состояния электрона в атоме водорода

Пометки для себя (не учитываются при проверке)



Радиальная часть волновой функции 1s - состояния электрона в атоме водорода

Пометки для себя (не учитываются при проверке)

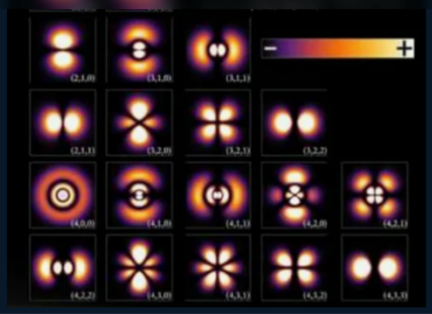


Радиальная часть волновой функ состояния электрона в атоме водо

Пометки для себя (не учитываются при прове

Study Ways

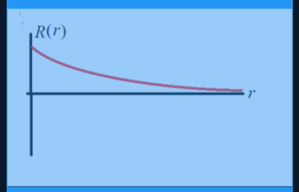
МЕНЮ АККАУНТ



Вопрос

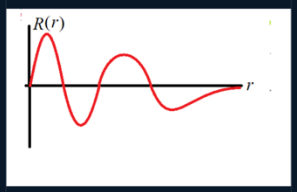
Из предлагаемых вам вариантов выберите правильные графики радиальных частей волновых функций стационарных состояний электрона в атоме водорода. Учтите, что для осложнения пользованием услугами gpt-чатом графики рисовались от руки без соблюдения относительных величин максимумов и расстояний между нулями волновых функций.

ПРОВЕРИТЬ СДАТЬСЯ



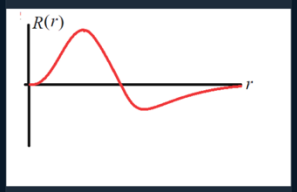
Радиальная часть волновой функции 1s - состояния электрона в атоме водорода

Пометки для себя (не учитываются при проверке)



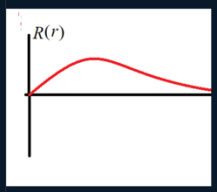
Радиальная часть волновой функции 2p - состояния электрона в атоме водорода

Пометки для себя (не учитываются при проверке)



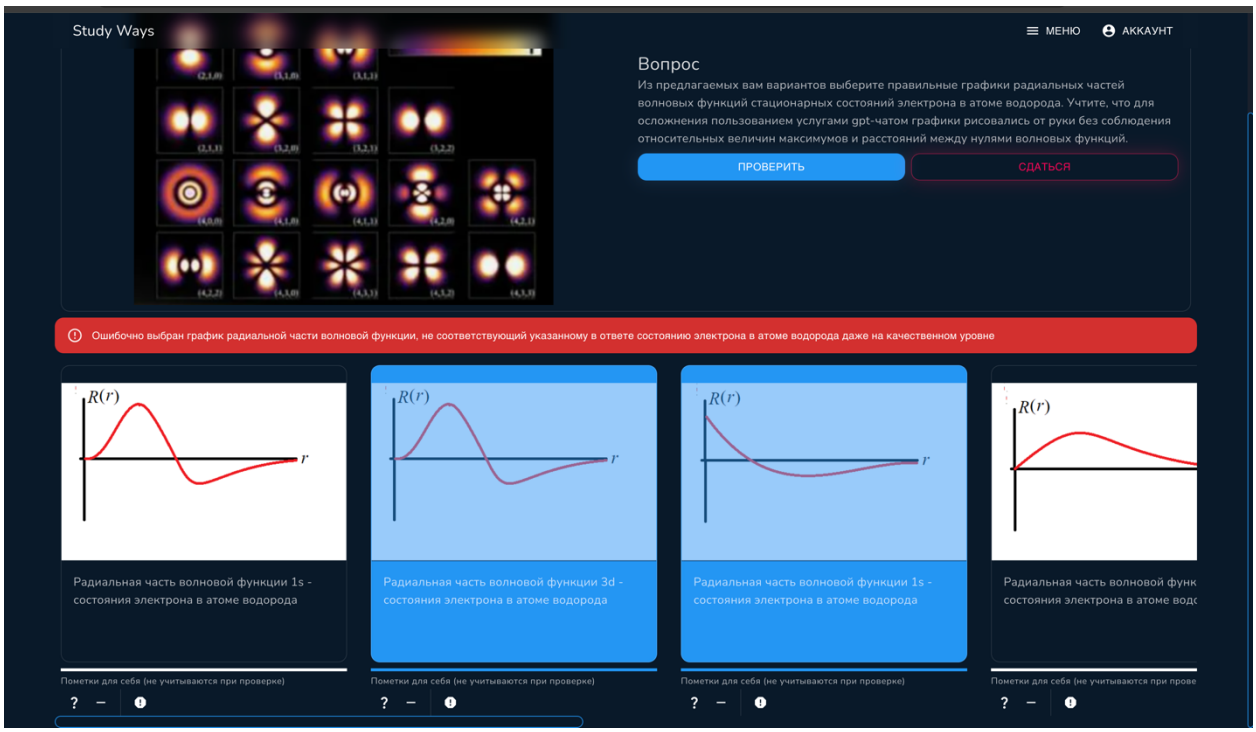
Радиальная часть волновой функции 1s - состояния электрона в атоме водорода

Пометки для себя (не учитываются при проверке)



Радиальная часть волновой функ состояния электрона в атоме водо

Пометки для себя (не учитываются при прове



1. Назначение и цели модуля

Модуль тестов предназначен для проведения интерактивной проверки знаний обучающихся. Тесты реализованы в форме диалога между системой и пользователем, в котором обучающийся может совершать ошибки, получать подсказки и пошагово приходить к правильному решению. Основная цель — не только контролировать усвоение материала, но и обучать через процесс постепенного нахождения правильного ответа.

2. Описание интерфейса и компонентов

Интерфейс модуля включает два уровня:

- страницу выбора теста — отображает карточки с ID и началом текста вопроса;
- интерфейс прохождения теста, включающий:
  - текст вопроса и связанное изображение (если есть);
  - список ответов (до 10), каждый с текстом и изображением;
  - кнопки для пользовательских пометок: «сомневаюсь», «вопрос ошибочен», «неверный вопрос»;
  - диалоговую подсказку, появляющуюся при ошибке;
  - кнопки подтверждения ответа и перехода к следующему этапу.

Ответы выводятся в случайном порядке и только те, у которых разрешён режим тренировки.

3. Техническая реализация

Модуль реализован на React, логика взаимодействия с тестами управляется через GraphQL API. Система подсказок и баллов реализована следующим образом:

- Уровни сложности ответов:
  - Лёгкий
  - Средний
  - Сложный
- Подсказки:
  - Отображается только одна подсказка — для самой грубой ошибки в попытке, то есть ошибки, связанной с ответом наименьшего уровня сложности.
- Система баллов:
  - За правильный ответ:
    - сложный: +15 баллов
    - нормальный: +10 баллов
    - лёгкий: +5 баллов

- За ошибку:
  - сложный: –5 баллов
  - нормальный: –10 баллов
  - лёгкий: –15 баллов

- Баллы за попытку считаются как сумма набранных и потерянных очков.
- Общий балл за тест рассчитывается как взвешенное среднее:
  - сумма отношений набранных баллов к максимальному,
  - с понижающим коэффициентом на каждую последующую попытку (вес уменьшается от попытки к попытке).

Тест считается завершённым, когда пользователь дал правильный ответ (с учётом всех обязательных вариантов) или сдался.

4. Механизмы персонализации

Модуль тестов является важнейшим компонентом адаптивного обучения. Персонализация реализуется через:

- выбор уровня сложности вопроса;
- индивидуальные подсказки, адаптированные под ошибки пользователя;
- влияние истории ошибок на дальнейшие рекомендации;
- гибкую систему баллов, стимулирующую внимательное чтение и рассуждение.

Также пользователь может видеть графики попыток (в модуле 2.6), что способствует саморефлексии и повторному прохождению сложных вопросов.

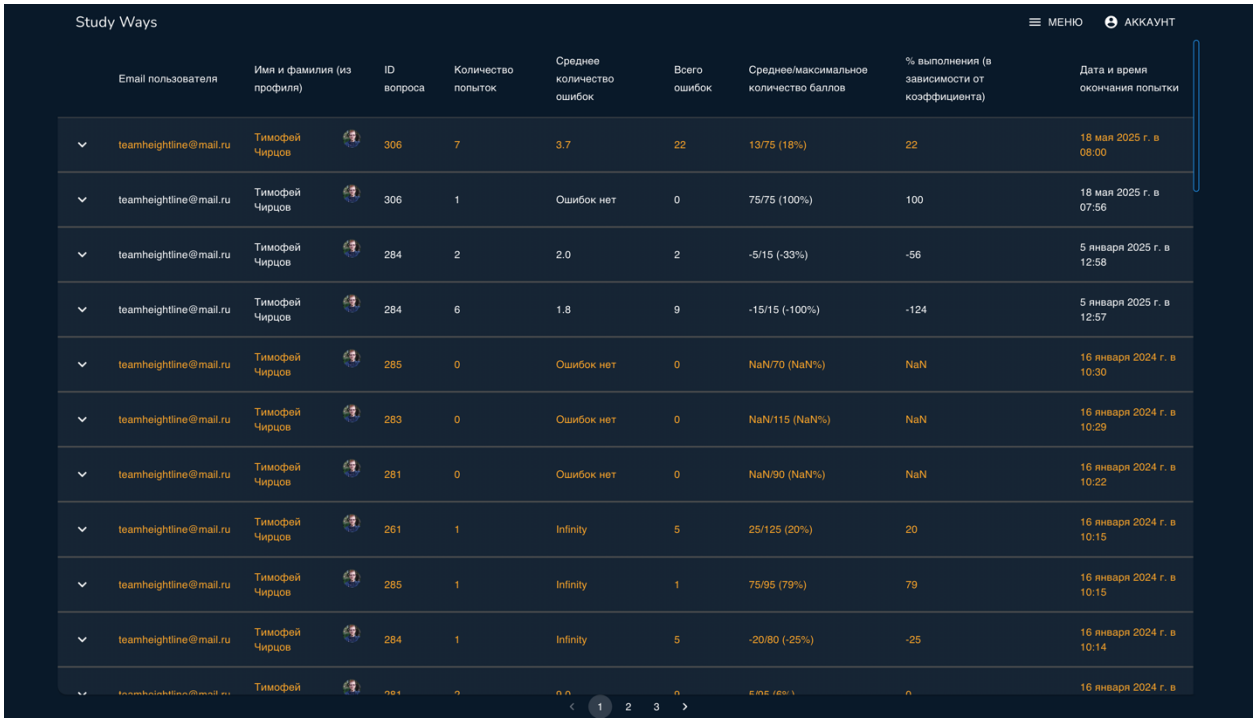
5. Взаимодействие с другими модулями

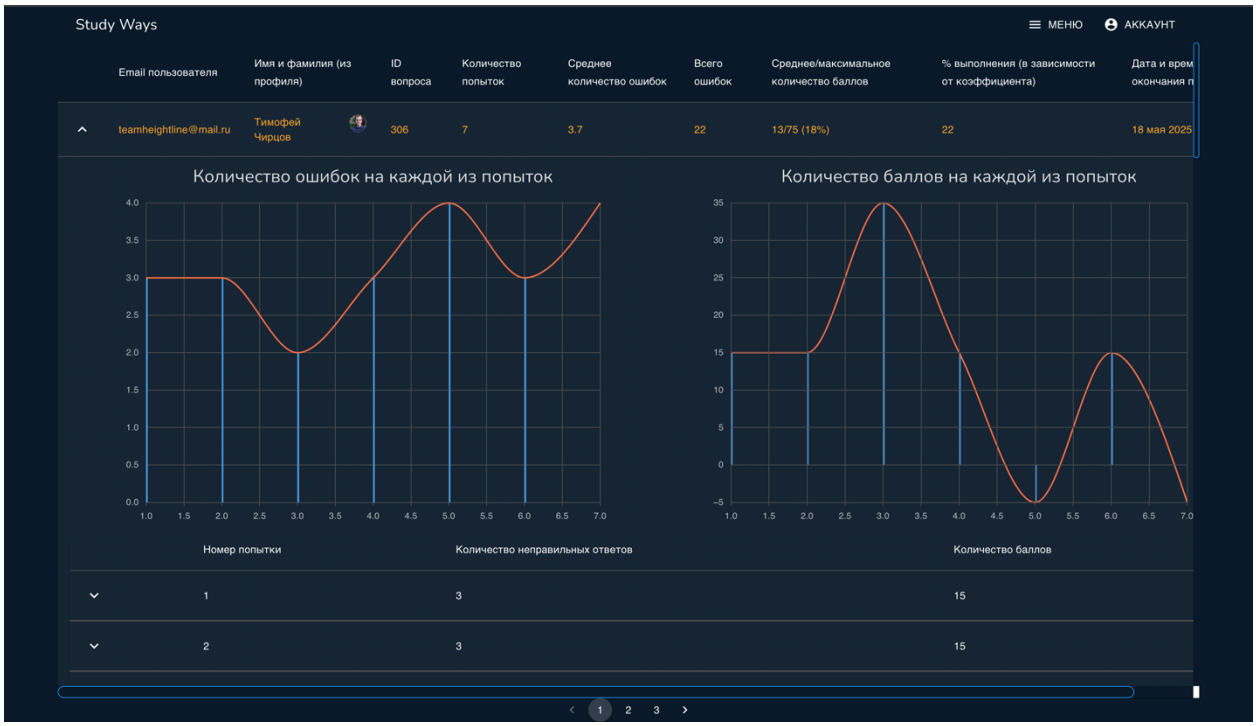
Модуль тестов связан с:

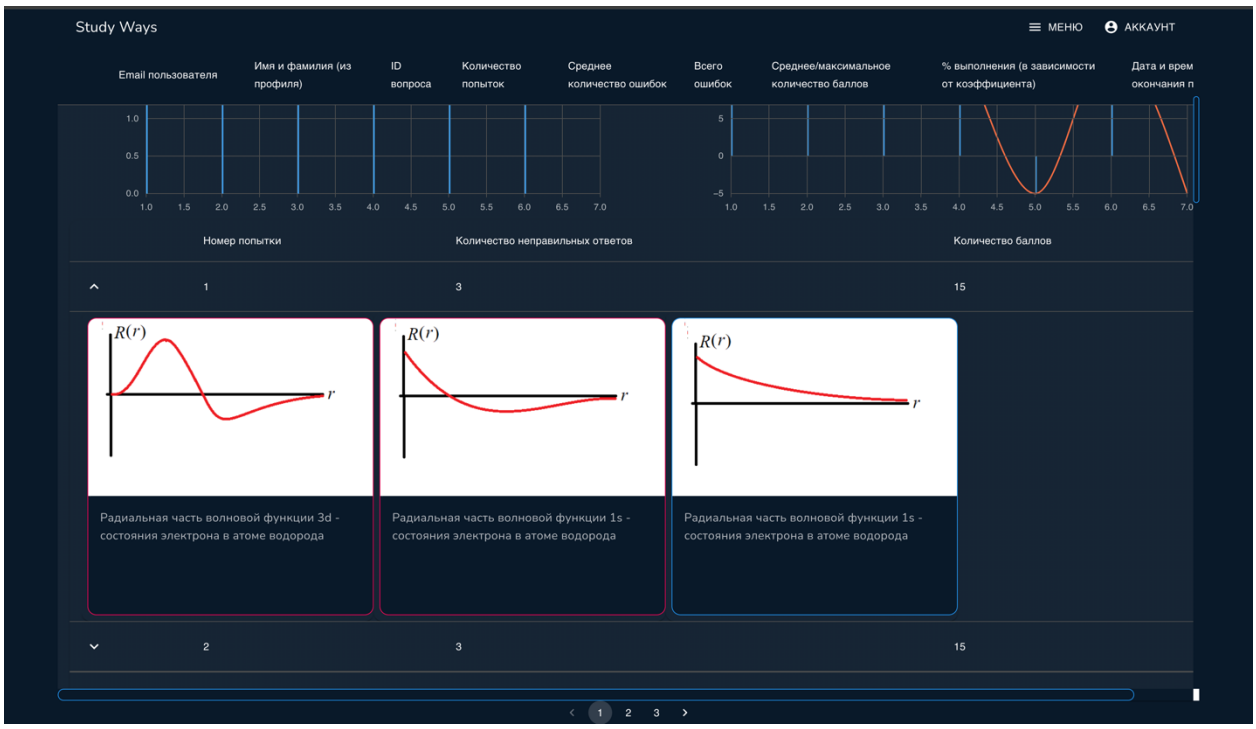
- модулем отображения ресурса (2.2) — тесты могут быть привязаны до или после ресурса;
- редактором вопросов (2.9) — источник содержания и логики тестов;
- модулем просмотра результатов (2.6) — хранит статистику по попыткам;

- модулем общей статистики (2.10) — агрегирует информацию по всем пользователям;
- AI-модулем (2.4) — тесты могут встраиваться в персональные траектории.

2.6 Модуль просмотра результатов тестов.







1. Назначение и цели модуля

Модуль предназначен для анализа результатов тестирования пользователей. Он предоставляет подробную информацию о попытках прохождения вопросов, позволяет отслеживать динамику обучения и выявлять сложные темы. Для преподавателей и администраторов предусмотрены расширенные инструменты просмотра ошибок и статистики, что делает модуль важным элементом системы адаптивного обучения.

2. Описание интерфейса и компонентов

Интерфейс модуля включает:

- таблицу результатов с постраничным выводом по 50 записей;
- фильтры:
  - по пользователю;
  - по вопросу;
  - по времени;
  - по режиму (подготовка / тестирование);
- индикаторы: ID вопроса, текст вопроса (по наведению), число попыток, среднее и общее число ошибок, средний балл, процент прохождения, дата и время последней попытки.

При клике на строку с результатами раскрывается график:

- числа ошибок на каждой попытке;
- баллов, набранных на каждой попытке.

Для пользователей с правами преподавателя или администратора появляется расширенный режим: просмотр ошибок по каждому варианту ответа на каждой попытке.

3. Техническая реализация

Модуль реализован как клиентский компонент на React, использующий библиотеку визуализации графиков Chart.js. Данные запрашиваются через GraphQL API с серверной стороны Django, где агрегируются результаты прохождения тестов.

Фильтры реализованы с использованием динамической подгрузки данных, что позволяет работать с большими объёмами информации без перегрузки интерфейса.

Раскрытие строк с результатами реализовано через компонент-аккордеон, внутри которого отображаются графики и детальная информация.

4. Механизмы персонализации

Модуль отображает не только «сырые» результаты, но и вычисляет персонализированные метрики:

- процент прохождения с учётом убывающего веса повторных попыток;
- отношение набранного балла к максимально возможному;
- среднее количество ошибок и успешных попыток;
- выявление типичных ошибок на уровне конкретных ответов.

Для каждого пользователя можно отследить прогресс во времени и выделить зоны, требующие дополнительного внимания. Эти данные могут использоваться рекомендательной системой для подбора последующих ресурсов или тестов.

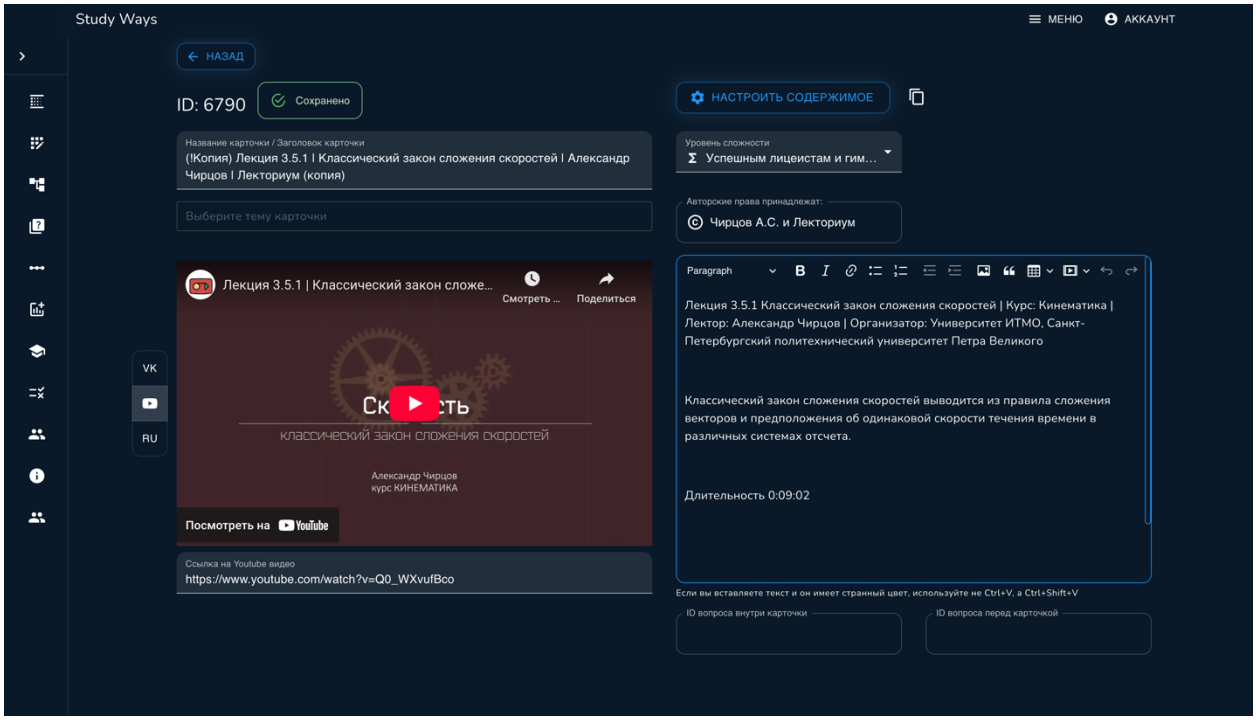
5. Взаимодействие с другими модулями

Модуль связан с:

- модулем тестов (2.5) — отображает накопленные данные по попыткам;
- модулем общей статистики (2.10) — агрегирует данные по всем пользователям;

- редактором вопросов (2.9) — при анализе ошибок преподаватель может перейти к редактированию конкретного вопроса.

2.7 Редактор ресурсов.



1. Назначение и цели модуля

Редактор ресурсов предоставляет пользователю с соответствующим уровнем доступа (преподавателю, администратору или студенту с правами автора) возможность создавать и редактировать образовательные ресурсы. Основная цель — обеспечить гибкий инструмент для подготовки и настройки учебного контента, который будет использоваться в курсах и тестах.

2. Описание интерфейса и компонентов

Интерфейс редактора включает следующие элементы:

- ID ресурса и индикатор статуса сохранения (автосохранение);
- поле для ввода названия ресурса;
- древовидный выбор темы, в которой будет размещён ресурс;
- выпадающий список для выбора уровня сложности;
- поле для указания авторских прав;
- зона вставки ссылки на видео (YouTube, VK, RuTube);

- зона загрузки изображения (для ресурсов с типом "изображение" или "внешняя ссылка");
- текстовый редактор описания, поддерживающий разметку, таблицы, ссылки, форматирование и откат изменений;
- поля для указания ID тестов до и после ресурса (пред- и пост-тестирование);
- кнопка создания копии ресурса;
- кнопка открытия расширенных настроек ресурса.

Редактор поддерживает гибкое редактирование и проверку данных, а также предоставляет превью контента, включая вставленные медиафайлы.

3. Техническая реализация

Редактор реализован на React с использованием библиотеки MUI и CKEditor 5 в качестве rich-text редактора. CKEditor обеспечивает поддержку форматирования, вставки таблиц, ссылок, списков и других элементов разметки, необходимых для подготовки учебного материала.

Все изменения в редакторе сохраняются автоматически, если пользователь не вводит данные в течение 5 секунд, редактор инициирует запрос на сохранение. Это обеспечивает баланс между отзывчивостью интерфейса и минимизацией числа запросов к серверу.

Сохранение происходит немедленно и без стадии публикации: после успешного сохранения изменения становятся доступны во всех модулях платформы. Чтобы избежать отображения устаревшей информации, при обновлении ресурса принудительно сбрасывается кэш на стороне бэкенда. Для ускорения доступа к данным используется глобальный in-memory кэш для запросов ресурсов, который также обновляется при изменении содержимого.

Тематика и уровень сложности сохраняются как метаинформация, используемая в других модулях (в частности, в рекомендательной системе и курсовом редакторе).

4. Механизмы персонализации

Редактор косвенно влияет на персонализацию за счёт:

- назначения тем и уровней сложности, используемых рекомендательной системой;

- возможности привязки к тестам (до и после ресурса), что формирует персонализированный маршрут прохождения;
- включения ресурсов в AI-курсы и многоуровневые программы с учётом структуры и метаданных, заданных в редакторе.

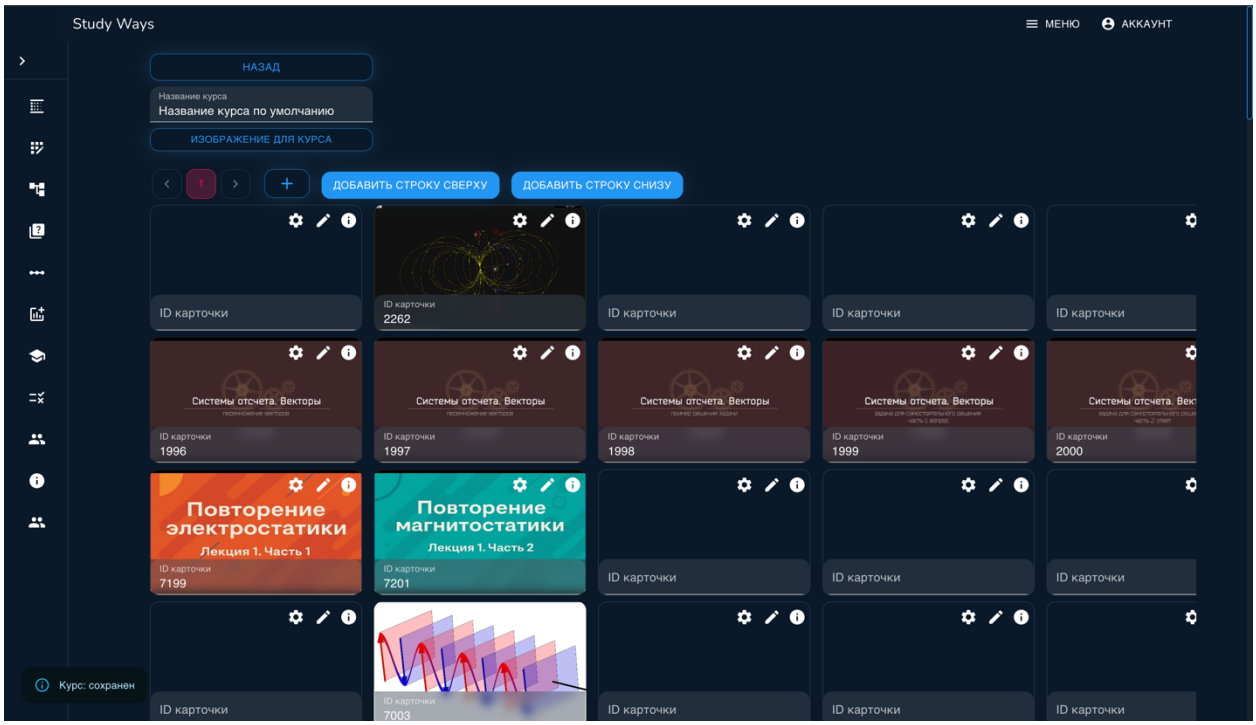
Кроме того, модуль может адаптироваться под роль пользователя: преподавателям доступны все функции, студентам — только базовый режим создания и редактирования.

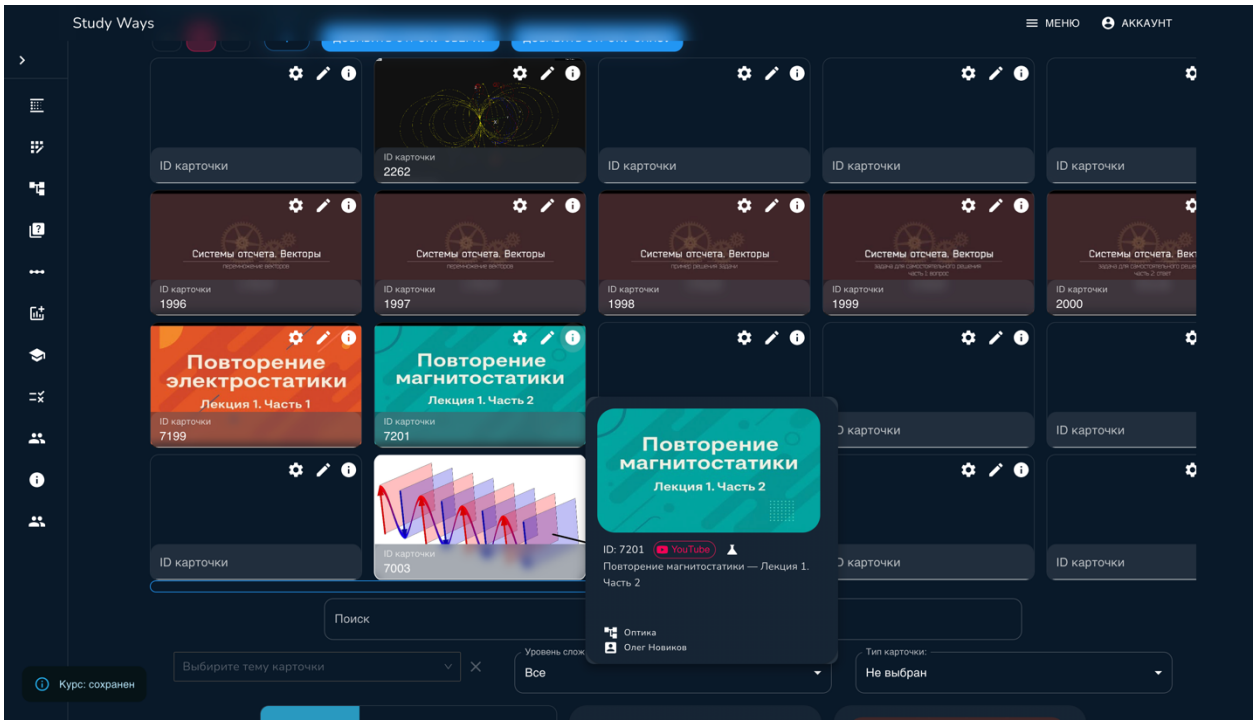
5. Взаимодействие с другими модулями

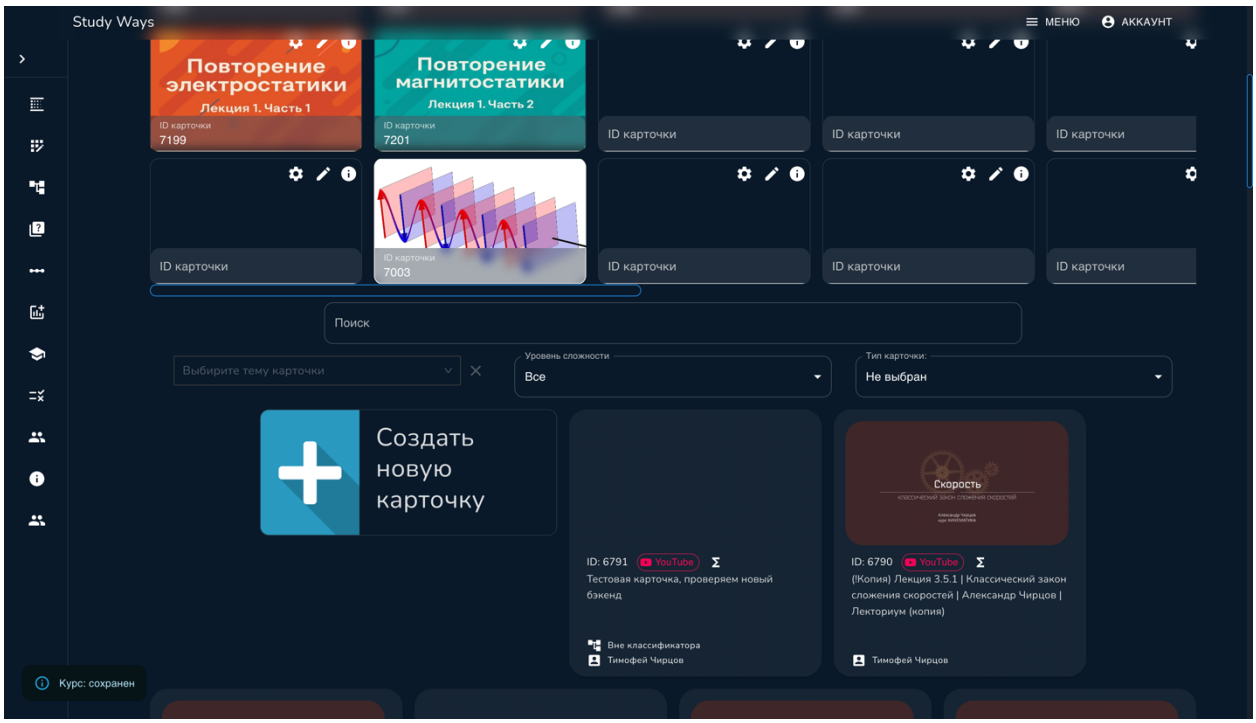
Редактор связан с:

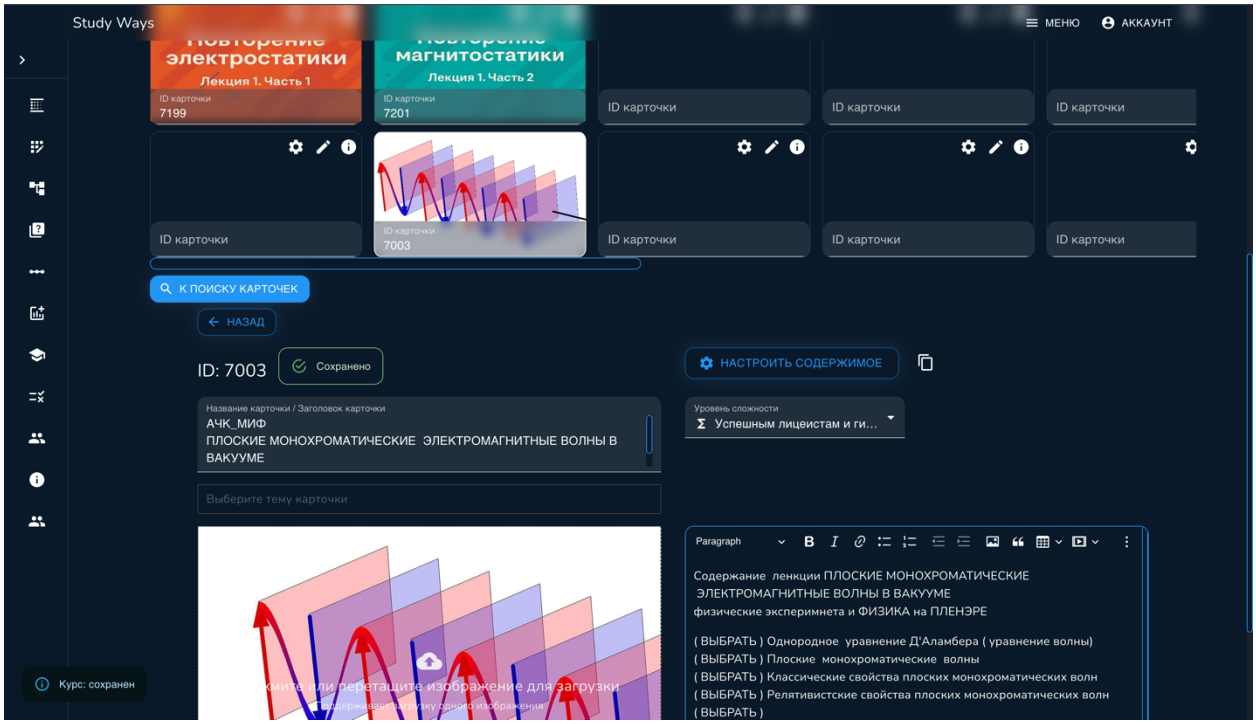
- модулем выбора ресурсов (2.1) — позволяет вставлять ресурсы через поиск;
- модулем отображения ресурса (2.2) — созданный ресурс можно просматривать как конечный пользователь;
- модулем курсов (2.3) — редактируемые ресурсы могут включаться в курс;
- редактором курсов (2.8) — при клике на ресурс в курсе открывается этот редактор;
- модулем тестов (2.5) — ресурсы могут быть связаны с вопросами до и после.

2.8 Редактор курсов.









1. Назначение и цели модуля

Редактор курсов предоставляет преподавателям, администраторам и авторам возможность создавать и модифицировать структуру образовательных курсов. Основная цель — реализовать гибкий и интуитивный инструмент для конструирования многоуровневых курсов, включающих ресурсы, ссылки и переходы между курсами. Редактор позволяет настраивать траектории обучения, адаптированные под разные уровни подготовки и интересы обучающихся.

2. Описание интерфейса и компонентов

Интерфейс редактора включает:

- поле для ввода названия курса;
- кнопку для загрузки изображения курса;
- переключатель страниц курса;
- кнопки для создания новой страницы, добавления строк сверху/снизу;
- двумерную таблицу ячеек, каждая из которых может содержать:
  - один или несколько ресурсов;
  - ссылки на позиции в других курсах;
- визуализацию ресурсов в ячейках (изображение, название, тип);
- редактор снизу страницы, который автоматически открывается при нажатии на иконку редактирования ресурса внутри ячейки;
- всплывающие карточки с информацией при наведении на элементы.

Редактор обеспечивает визуальное представление структуры курса и интуитивное взаимодействие с элементами.

3. Техническая реализация

Курс сохраняется в формате JSON, что обеспечивает:

- представление курса как набора страниц, каждая из которых состоит из таблицы ячеек;
- возможность указания нескольких ресурсов в одной ячейке;
- поддержку ссылок на конкретные позиции в других курсах, реализуя переиспользование материалов и построение сложных межкурсовых связей.

Редактор построен на React с использованием библиотеки MUI. Состояние редактируемого курса синхронизируется с сервером через GraphQL API. При редактировании ресурса из ячейки открывается встроенный компонент редактора ресурсов (2.7).

Изображения ресурсов подгружаются из кэша, обеспечивая визуальную идентификацию без задержек.

4. Механизмы персонализации

Редактор курсов позволяет формировать персонализированные маршруты обучения за счёт:

- структурирования контента по темам и уровням сложности, что учитывается рекомендательной системой;
- возможности включения ссылок на конкретные позиции в других курсах, что даёт гибкость в построении межкурсовых маршрутов;
- поддержки вставки нескольких ресурсов в одну ячейку — в особых случаях, когда на данном уровне сложности требуется расширить охват темы. Такая практика допускается, но не является рекомендуемой: основным подходом остаётся использование одного ресурса на ячейку, что способствует модульности, переиспользуемости и упрощает восприятие курса.

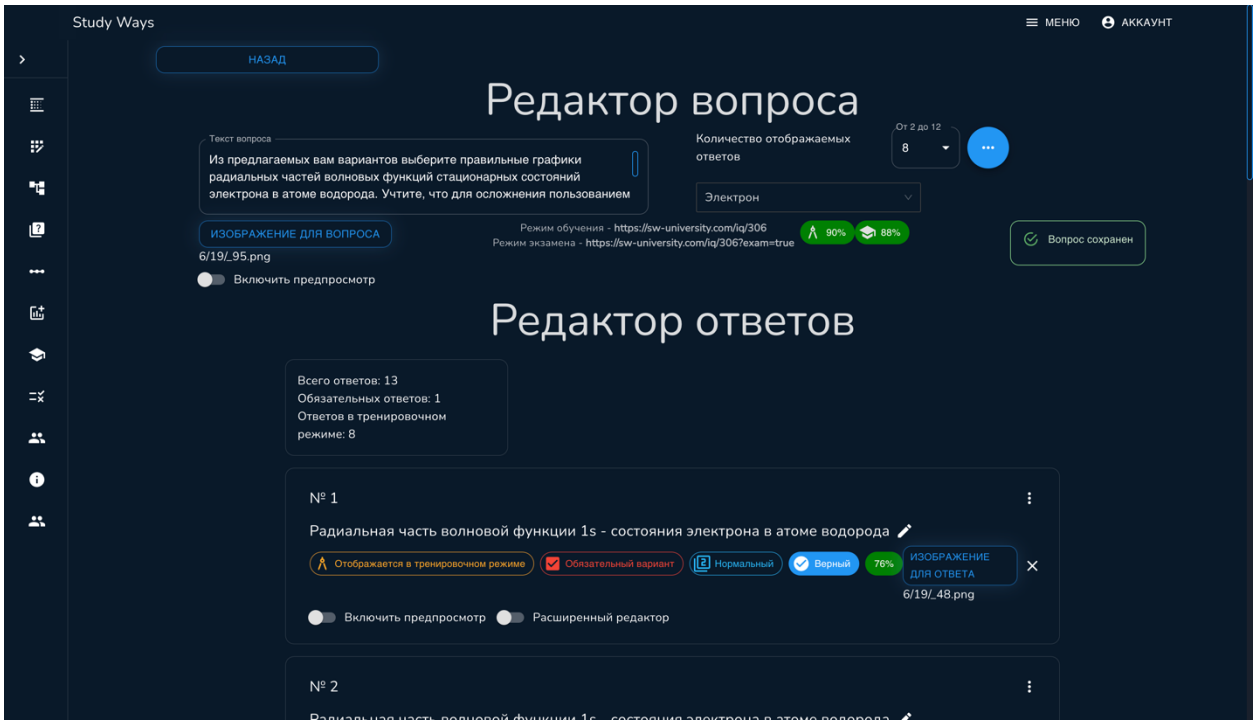
Таким образом, редактор поддерживает баланс между структурной строгостью и необходимой гибкостью, позволяя преподавателю адаптировать подачу материала под реальные потребности обучающихся.

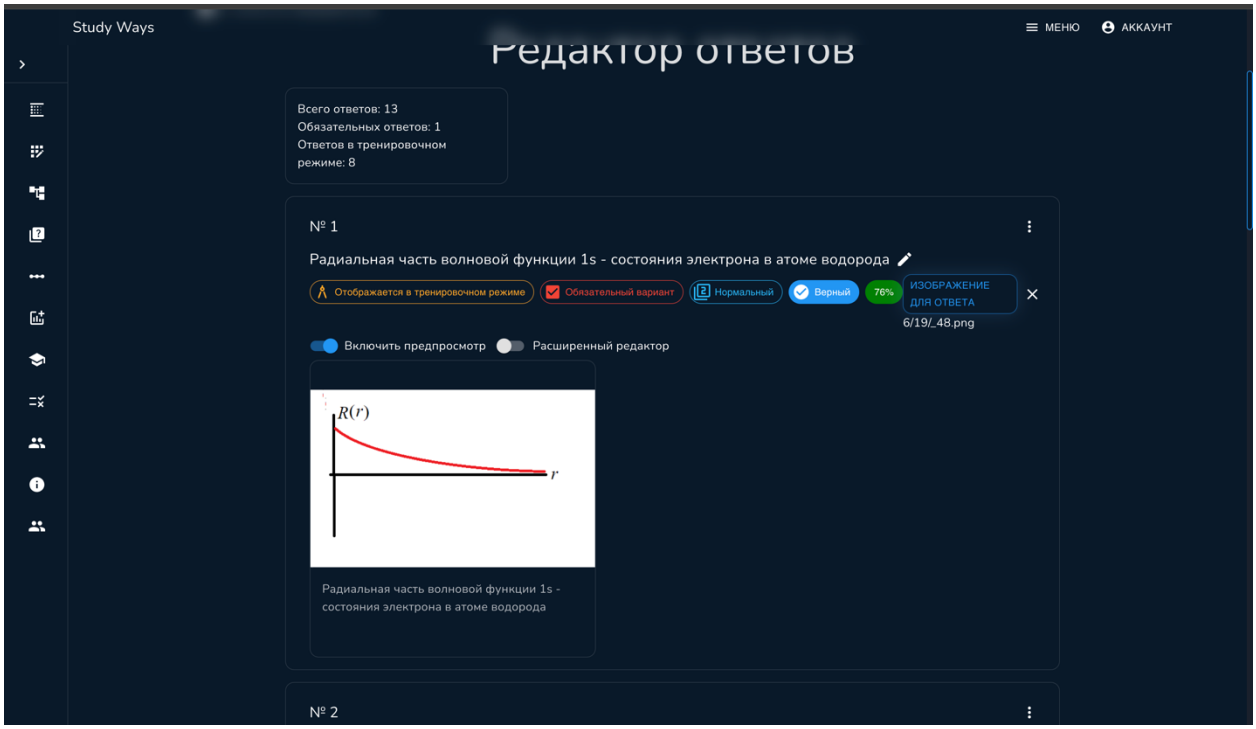
5. Взаимодействие с другими модулями

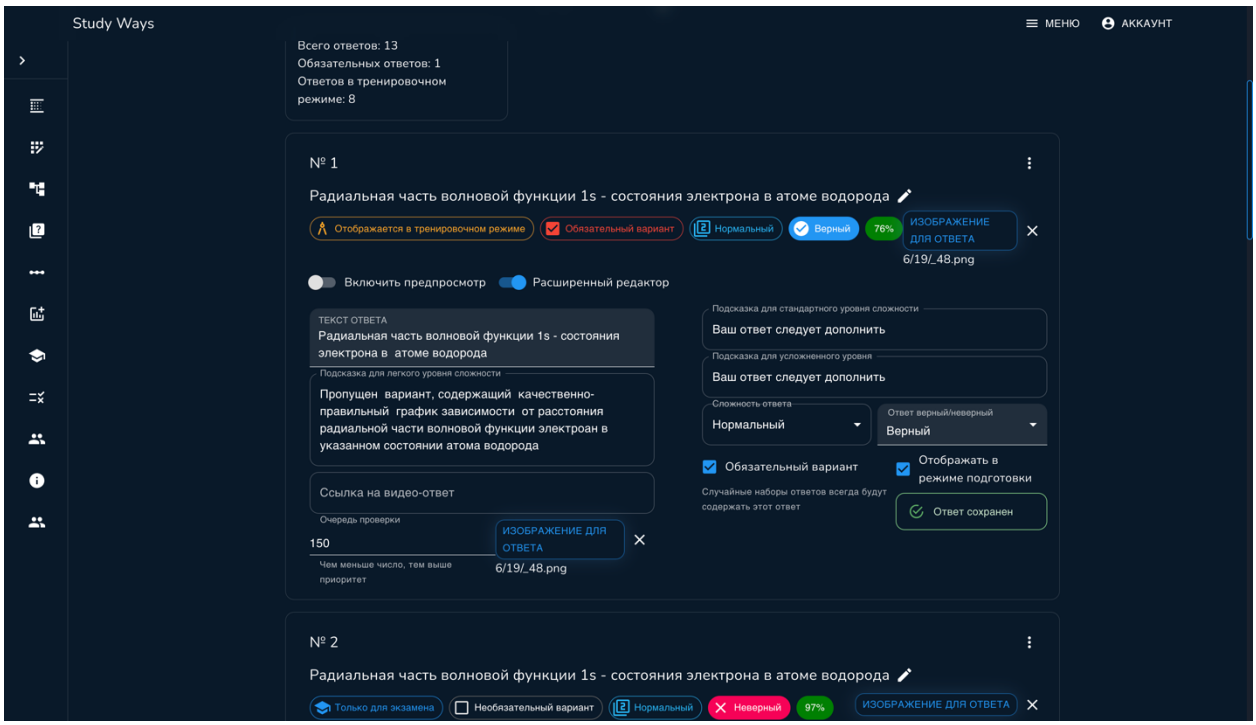
Редактор курсов интегрирован с:

- редактором ресурсов (2.7) — для редактирования ресурсов, указанных в ячейках;
- модулем отображения курса (2.3) — визуализирует курс для пользователя по структуре, заданной в редакторе;
- модулем отображения ресурса (2.2) — позволяет переходить к просмотру материалов;
- AI-модулем (2.4) — возможно использование ресурсов курсов в AI-траекториях;
- поисковым компонентом (2.1) — для быстрого добавления ресурсов.

2.9 Редактор вопросов.







1. Назначение и цели модуля

Редактор вопросов предназначен для создания, настройки и корректировки тестовых заданий, используемых в платформе. Он предоставляет преподавателям и администраторам удобный интерфейс для задания вопросов, вариантов ответов, уровней сложности и подсказок, обеспечивая тем самым высокую точность контроля знаний и возможность адаптивной проверки усвоения материала.

2. Описание интерфейса и компонентов

Интерфейс редактора включает:

- текст вопроса;
- изображение, прикреплённое к вопросу;
- компонент выбора темы (в виде дерева);
- чекбоксы режимов использования (подготовка, экзамен, копирование);
- ссылку для предварительного просмотра вопроса в интерфейсе прохождения;
- индикатор сохранения;
- статистику по прохождению вопроса в разных режимах (число попыток, успехи/ошибки).

Блок редактирования ответов включает:

- текст ответа;
- изображение (опционально);
- переключатели:
  - «Верный / Неверный»;
  - «Показывать в тренировочном режиме»;
  - «Обязательный к показу»;
  - «Уровень сложности»;
  - «Показать в предпросмотре»;
  - «Расширенный редактор» (открывает дополнительные поля);
- статистику ошибок по конкретному ответу;
- кнопки удаления и копирования.

В расширенном редакторе дополнительно доступны:

- выпадающие списки и чекбоксы вместо простых переключателей;
- поля для ввода подсказок на разных уровнях сложности, которые используются в интерактивной системе подсказок при тестировании.

3. Техническая реализация

Редактор реализован на React, использует компоненты MUI и динамически подгружаемые формы. Данные о вопросах и ответах сохраняются через GraphQL API. Все изменения валидируются на клиенте и сервере.

Редактор поддерживает автоматическое сохранение по таймеру без необходимости ручного подтверждения.

Для загрузки изображений используется общее файловое хранилище, ссылки сохраняются в модели ответа. Вопросы и ответы поддерживают глубокую типизацию и могут быть синхронизированы с системой статистики прохождений.

4. Механизмы персонализации

Редактор позволяет задавать уровни сложности для каждого ответа и устанавливать подсказки, что становится основой адаптивного тестирования:

- пользователи могут выбирать уровень помощи;

- система динамически подбирает подсказки в зависимости от ошибки и сложности;
- анализ ошибок позволяет рекомендовать повторное прохождение ресурса или альтернативные объяснения.

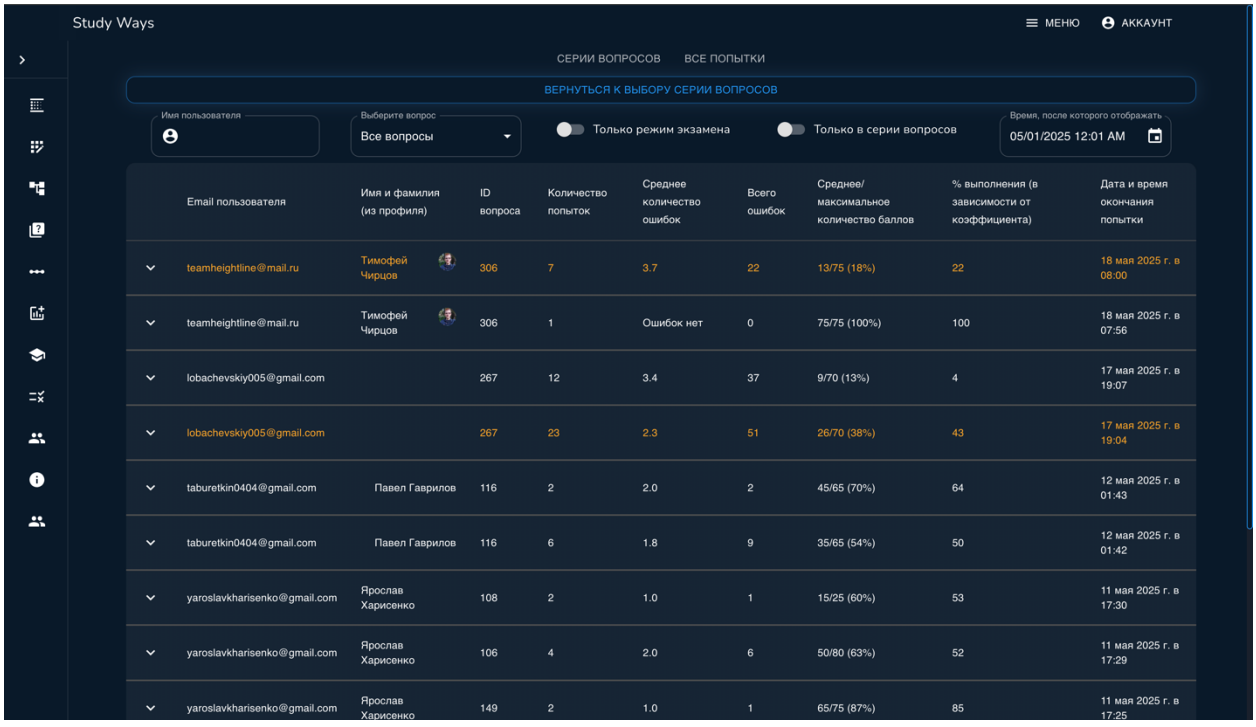
Таким образом, через правильную настройку вопросов преподаватель может существенно повлиять на персонализированный путь обучения.

5. Взаимодействие с другими модулями

Редактор связан с:

- модулем тестов (2.5) — создаёт и настраивает вопросы, используемые в тестировании;
- модулем просмотра результатов (2.6) — статистика ошибок по вопросам синхронизирована и используется для анализа;
- модулем общей статистики (2.10) — агрегирует данные по вопросам для преподавателей;
- редактором ресурсов (2.7) — при необходимости вопросы могут быть связаны с ресурсами (до и после ресурса).

2.10 Модуль общей статистики прохождения тестов.



1. Назначение и цели модуля

Модуль предназначен для анализа статистики прохождения тестов в масштабах всей платформы. Он позволяет преподавателям и администраторам отслеживать динамику успеваемости, выявлять наиболее проблемные вопросы, темы и пользователей, а также принимать обоснованные решения о корректировке курсов или тестов. Этот инструмент особенно полезен при масштабной работе с группами обучающихся или при оценке эффективности конкретных элементов образовательной программы.

2. Описание интерфейса и компонентов

Интерфейс модуля включает:

- переключатель между режимом отображения всех данных и данными по конкретному набору вопросов;
- фильтры:
  - по пользователю;
  - по конкретному вопросу;
  - по времени (интервал от и до);
  - по режиму прохождения (только тестирование);
- таблицу статистики (постраничная, по 50 записей), содержащую:
  - имя пользователя;
  - ID и текст вопроса (по наведению);
  - количество попыток;
  - среднее и общее количество ошибок;
  - итоговый балл;
  - отношение балла к максимуму;
  - вычисленный процент прохождения;
  - дата и время завершения.

При выборе записи преподаватель может раскрыть подробную статистику попыток (через связанный компонент из модуля 2.6), включая графики и ошибочные ответы.

3. Техническая реализация

Модуль реализован на React, взаимодействие с сервером осуществляется через GraphQL API. Для производительности используется серверная агрегация данных на стороне Django, а также постраничная подгрузка и кэширование результатов на время сессии.

Фильтрация и агрегация показателей выполняются по следующим метрикам:

- число попыток;
- среднее и общее количество ошибок;
- итоговый балл с учётом поэтапного понижения веса повторных попыток;
- процент успешности, основанный на отношении реального результата к максимально возможному баллу.

Данные вычисляются с учётом системы подсказок и весов, заданных в настройках вопросов (см. 2.5 и 2.6).

4. Механизмы персонализации

Модуль предоставляет преподавателям возможность:

- выявлять студентов, испытывающих трудности с определёнными вопросами или темами;
- анализировать тенденции по группам (например, какие уровни сложности вызывают наибольшее число ошибок);
- принимать решения о персонализированной доработке курсов и ресурсоёмких тем;
- выявлять неэффективные или перегруженные вопросы на основе статистики многократных попыток и частых ошибок.

Эти данные могут быть использованы как вручную, так и в качестве входных параметров для рекомендательной системы, формирующей персональные траектории повторения и углубления материала.

5. Взаимодействие с другими модулями

Модуль тесно интегрирован с:

- модулем просмотра результатов тестов (2.6) — при выборе записи открывается подробная статистика;

- редактором вопросов (2.9) — позволяет анализировать эффективность отдельных вопросов и возвращаться к их настройке;
- модулем тестов (2.5) — обеспечивает поток данных по прохождению и связывает вопросы с пользователями;
- рекомендательной системой (неявно) — статистика может служить входом для рекомендаций.

2.11 Вывод

Во второй главе была рассмотрена реализация программной платформы StudyWays с фокусом на модульной архитектуре и механизмах персонализации обучения. Каждый из представленных модулей выполняет чётко определённую функцию, в совокупности формируя гибкую и расширяемую образовательную среду.

В процессе описания были выделены ключевые особенности платформы:

- использование современных технологий на клиентской и серверной стороне (React, Django, Express, GraphQL, Prisma);
- чёткое разграничение открытого и внутреннего API для обеспечения безопасности и масштабируемости;
- продуманная организация пользовательского интерфейса, обеспечивающая интуитивное взаимодействие с контентом;
- поддержка редактирования и многократного переиспользования контента благодаря независимости ресурсов от курсов;
- интеграция с рекомендательной системой, позволяющая формировать персональные траектории обучения в режиме реального времени.

Особое внимание уделено реализации адаптивных и обучающих тестов, а также возможности построения AI-курсов — индивидуальных маршрутов изучения темы, формируемых динамически на основе поведения пользователя.

Таким образом, архитектура StudyWays реализует принципы современного электронного обучения: персонализацию, модульность, интерактивность и масштабируемость. Функциональные компоненты платформы взаимодействуют между собой как единое целое, формируя техническую основу для построения индивидуальных образовательных траекторий.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была разработана и реализована современная образовательная платформа StudyWays, ориентированная на сопровождение индивидуализированного электронного обучения. Основной целью проекта стало создание технологического решения, способного обеспечивать построение персонализированных образовательных траекторий, адаптированных к уровню подготовки, интересам и учебной активности каждого конкретного обучающегося.

В процессе работы были:

- проанализированы существующие образовательные платформы и подходы к адаптивному обучению;
- обоснован выбор архитектурных решений и технологий;
- реализованы серверные модули: открытое API на Django + GraphQL и внутренний сервис бизнес-логики на Express.js + Prisma;
- разработано клиентское приложение на React с применением современных подходов к управлению состоянием, стилизации и построению интерфейса;
- внедрены механизмы интеллектуального поиска, рекомендательные алгоритмы и процедурная генерация курсов;
- проведена апробация платформы в ряде учебных заведений, включая СПбГЭТУ «ЛЭТИ», Университет ИТМО и ФМЛ №30.