ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»
(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)

УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ДОПУСКАЕТСЯ К ЗАЩИТЕ
Председатель ПЦК специальности 09.02.07 Информационные системы и программирование
_____А.И. Глускер
«_____»_____2023 г.

ДИПЛОМНЫЙ ПРОЕКТ

на тему: Разработка электронного образовательного ресурса для использования
в образовательной организации (на примере темы «Молекулярная физика. Термодинамика» дисциплины «Физика»)

Студента группы 090207-9о-19/1
Специальности 09.02.07 Информационные системы и программирование
Сарапулова Даниила Владиславовича

Студент _____ Д.В. Сарапулов
Руководитель _____ Л.В. Салахова
Консультанты:
Нормоконтроль _____ И.Г. Дзюба

Дата защиты «____» _____ 2023 г.
Оценка:_____
Председатель ГЭК _____ П.Р. Сафиканов

Москва
2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время наличие электронных образовательных ресурсов (ЭОР) является неотъемлемой частью обучения в различных образовательных учреждениях, а также в иных сферах деятельности. Электронные образовательные ресурсы входят в обиход образовательных учреждений ввиду их доступности с любого устройства, с любой точки планеты.

Электронные образовательные ресурсы легко заменяют книги и учебники, а также позволяют лучше донести информацию до пользователя из-за интерактивности ресурсов, информация легче усваивается пользователем в более непринужденной форме.

Сфера образования нуждается в такой же оцифровке, легкодоступности и мобильности как другие сферы жизни человека. Данная ситуация доказывается потребностью целевой аудитории, которая большую часть своего времени проводит в смартфонах и персональных компьютерах, в следствии которой уже существует множество различных электронных образовательных ресурсов, направленных, как и на общую тематику, так и на узкую (к примеру электронный образовательный ресурс в котором изучают только «Математику»).

Современная молодежь в большинстве своем мало осведомлена в области физики что создает ошибочное восприятие о данном направлении, и его деятельности, поэтому возникает малая вовлеченность в нужные и фундаментальные профессии.

В связи с этим актуальна разработка узконаправленного электронного образовательного ресурса, который поможет учебному заведению продемонстрировать и ознакомить обучающихся с дисциплиной «Физика», а также некоторыми ее направлениями.

Целю этой работы является полная разработка электронного образовательного ресурса для использования в образовательной организации (на примере темы «Молекулярная физика. Термодинамика» дисциплины «Физика»).

Были поставлены задачи**:**

– спроектировать и разработать базы данных;

– разработать серверный компонент rest api;

– внедрить функцию регистрации и авторизации пользователя;

– реализовать шифрование данных пользователя (пароль);

– реализовать выдачу данных по токену пользователя;

– разработать интерфейс веб-приложения (фронтенд spa);

– реализовать динамический фон;

– реализовать замедленное моделирование теплового движения молекул в зависимости от давления, температуры, объема;

– реализовать моделирование адиабатических, изохорных, изобарных, изотермических процессов;

– провести расчет экономической эффективности разработки.

Гипотезы выпускной квалификационной работы:

– предполагается, что целевая аудитория повысит уровень знаний и осведомленности в дисциплине «Физика»;

– предполагаются положительные отзывы о разработанном электронном образовательным ресурсом;

– предполагается, что электронный образовательный ресурс будет использоваться учебным заведением для обучения учеников и повышения осведомленности.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ предметной области

В соответствии с ГОСТ Р 52653-2006 Электронный образовательный ресурс должен включать образовательный контент, метаданные и программные компоненты.

Образовательный контент – это структурированный предметный контент, используемый в образовательном процессе, информационно-релевантный контент электронный образовательного ресурса. Программные компоненты обеспечивают представление элементов контента пользователю в определенных комбинациях, а также обеспечивают интерактивный способ работы с контентом.

Метаданные – структурированные данные, задача которых описывать характеристики электронного образовательного ресурса.

Физика – это раздел естествознания, экспериментальная наука в которой изучается материя, ее структура и движение, законы природы

Термодинамика – раздел физики, в котором изучаются тепловые явления. В отличие от других разделов исходит из общих закономерностей тепловых процессов и свойств макроскопических систем. В термодинамике рассматривают изолированные системы тел, которые находятся в термодинамическом равновесии, что указывает на то что в рассматриваемых системах прекратились все макроскопические процессы.

Молекулярная физика – раздел физики в котором изучаются свойства макроскопических тел и их агрегатные состояния, то молекулы в них взаимодействуют и двигаются.

В обоих разделах, которые тесно связанны друг с другом, часто для исследований или для объяснения тех или иных явлений применяют «идеальный газ».

Идеальный газ – математическая модель газа в теории, в которой не учитываются реальные размеры газовых частиц и показатели силы взаимодействующих газовых частиц.

В такой модели предполагается что кинетическая энергия частиц в разы больше чес энергия взаимодействия частиц газа.

В данных разделах популярна тема тепловых процессов, которые описываются следующими определениями:

– Адиабатические процессы – процесс при котором газ не производит теплообмен с окружающей его средой;

– Изохорный процесс – тоже самое что и адиабатический за исключением того что в этом процессе постоянен объем;

– Изометрический процесс – все тоже самое, но при постоянной температуре;

– Изобарный процесс – тоже самое, но уже давление не изменяется.

Архитектура REST API

API – Application Programming Interface. Это программный интерфейс, который позволяет связывать между собой различные приложения. На примере этой дипломной работы API будет связывать серверный компонент и базу данных, что бы отправляя запрос на серверный компонент можно было получить данные из базы данных.

REST – это архитектура API, которая работает с помощью протокола HTTP, и имеет возможность предоставлять данные с сервера.

Фреймворк – специальное «дополнение» к языку программирования, которое определяет структуру проекта, предоставляет некоторые инструменты, для разработки которых нет в «чистом» языке. Так же фреймворки могут объединять несколько функций в одну для упрощения и удобства разработки.

Почти для каждого языка программирования существуют фреймворки, ранее для данной работы был выбран язык программирования JavaScript.

Из-за того, что для этой работы был выбран JavaScript, для написания клиентской части и серверной, фреймворки для задач используются разные.

SPA (Single Page Application) – дословно переводя это одностраничное приложение, вся суть данной архитектуры заключается в том что веб–приложение работает на одной странице html, отрисовывая нужные компоненты.

Данная архитектура обладает рядом преимуществ в отличие от переадресации пользователя на отдельные html документы:

– быстродействие – не надо при каждом запросе загружать страницу, все уже заранее отрисованно;

– гибкость – зачастую применяется такой паттерн программирования, при котором существует главный родительский компонент, и на нем уже размещаются дочерние, к примеру блок с навигацией — это отдельный компонент, блок с контентом — это тоже отдельный компонент и т.д., эти компоненты легко настраивать по отдельности;

– модифицируемость – свойство программного обеспечения, характеризующая какие усилия необходимо приложить для улучшения и исправления программного обеспечения, в связи с найденными дефектами или подстраиванию под новые стандарты в сфере.

Требования к ресурсу - в соответствии с ГОСТ Р 52653-2006 Электронный образовательный ресурс должен включать образовательный контент, метаданные и программные компоненты.

Образовательный контент – это структурированный предметный контент, используемый в образовательном процессе, информационно-релевантный контент электронный образовательного ресурса. Программные компоненты обеспечивают представление элементов контента пользователю в определенных комбинациях, а также обеспечивают интерактивный способ работы с контентом.

Метаданные – структурированные данные, задача которых описывать характеристики электронного образовательного ресурса.

– ресурс должен быть интерактивным и модифицируемым.

Интерактивность – способность компонентов (в данном случае электронно-образовательного ресурса) активно реагировать на те или иные действия со стороны пользователя (к примеру, изображение при наведении на нее начинает двигаться или делать любые иные действия).

Модифицируемость – свойство программного обеспечения, характеризующая какие усилия необходимо приложить для улучшения и

исправления программного обеспечения, в связи с найденными дефектами или подстраиванию под новые стандарты в сфере.

– ресурс должен соответствовать современному представлению и правилам дизайна.

Следует придерживаться следующих правил в дизайне:

1) единство стиля на всех страницах;
2) удобство работы с навигацией сайта;
3) единство стиля на всех страницах;
4) удобство работы с навигацией сайта;
5) любой текстовый материал должен представляться пользователю максимально лаконично;
6) интерфейс должен быть юзерфрендли;
7) страницы не должны бать переполнены информацией если того не требуется;
8) любая аудио и медиа информация на сайте должна быть обоснована;

Юзерфрендли – современное понятие в сфере разработки обозначающее легкость и простоту использования интерфейса или функций пользователем.

– Ресурс должен быть понятен разработчикам с технической точки зрения.

В связи с тем, что рынок профессий в сфере разработки наполняется новыми кадрами, для любой компании свойственна «текучка» кадров и распределение большего количества человек для разработки и поддержания одного проекта. В связи с этим должен быть выбран более распространённый язык программирования для работы в команде.

1.2 Анализ и выбор инструментальных средств

1.2.1 Языки программирования

В настоящее время существует большое множество языков программирования каждый язык по-своему уникален, но не все они «гибкие» и универсальные. Поэтому стоит подбирать язык программирования, который легко освоить и максимально может закрыть потребности при разработке.

Очень часто такие языки являются самыми популярными в мире, список таких языков публикуется на сайте для разработчиков GitHub основанным в 2008 году, использующий и развивающий систему GIT каждый год публикуется статистика самых используемых языков, популярными языками программирования в сфере веб разработки являются (Рисунок 1, Рисунок 2):

– javascript;
– python;
– typescript;
– php;
– ruby

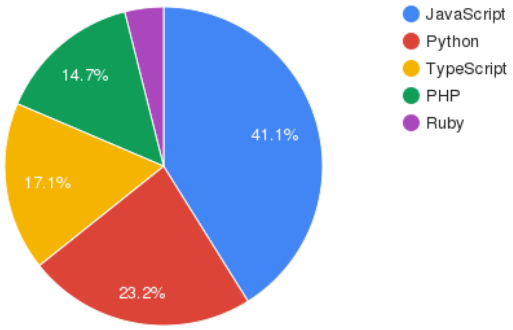

Рисунок 1 – Диаграмма часто используемых языков веб-программирования по данным GitHub

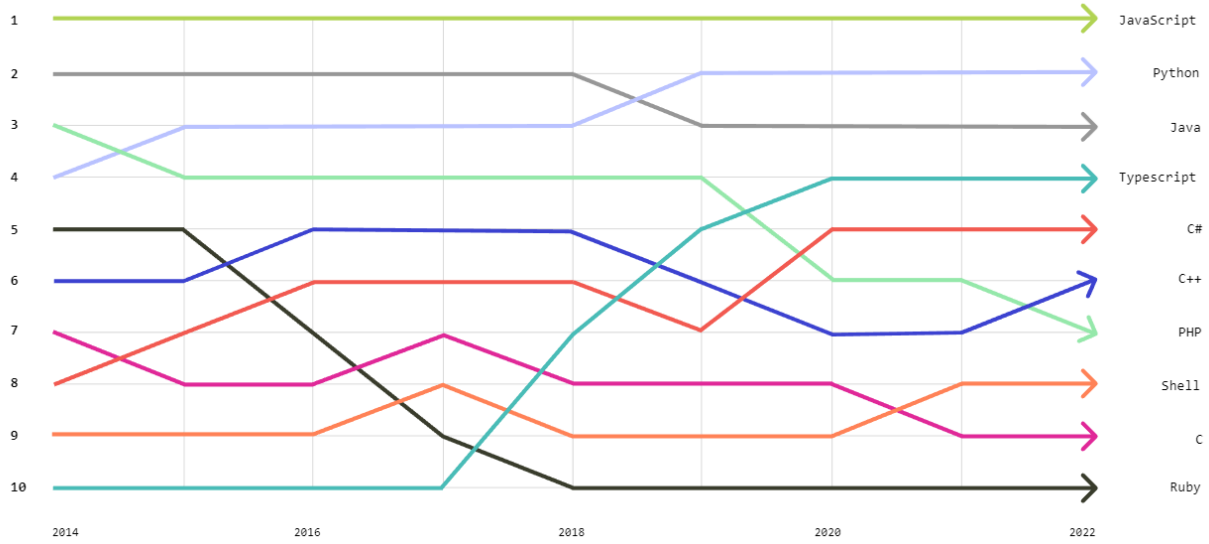

Рисунок 2 – График часто используемых языков в сервисе GitHub

1.2.2 Среды разработки

Среда разработки (IDE) выбирается самим разработчиком исходя из собственных нужд. Существуют универсальные среды разработки, которые поддерживают некоторое множество языков или все. Так же существуют среды разработки под определенные языки такие как PyCharm, который нацелен на работу только с Python. Для данной задачи лучше всего подходит Visual Studio и Visual Studio Code, так как они имеют поддержку огромного количества языков программирования, что позволяет пользоваться одним редактором для разных задач, и разных языков. В Visual Studio и Visual Studio Code так же есть встроенные подсказки синтаксиса, что упрощает изучение языка и разработку проекта. В данных продуктах есть интеграция с Git и GitHub

1.2.3 Система контроля версий

В разработке любых проектов разработчики используют систему контроля версий, такие как:

- Git;
- Mercurial;
- Subversion;
- Perforce.

Данные системы контроля версий записывают и сохраняют все изменения в файлах проекта, позволяют делать несколько различных ответвлений проекта, чтобы в них можно было вносить изменения без ущерба основной версии проекта, а после успешного обновления проекта на другом его ответвлении можно «слиять» вторичную и основную ветви проектов. Так же системы контроля версий позволяют вернуться к предыдущей версии проекта.

Самой популярной системой контроля версий является Git выпущенный в 2005 году, который в свою очередь имеет интеграцию с различными веб-сервисами для хостинга различных проектов и их совместной разработки. Такие веб-сервисы обеспечивают доступ к проектам с любого устройства, и из любой точки планеты.

Существует множество таких веб-сервисов, все они работают и выглядят по одному принципу, самым же популярным из них является GitHub.

Исходя из этого лучшим вариантом является система контроля версий Git, и веб-сервис GitHub.

1.2.4 Разработка и прототипирование дизайна

Для облегчения разработки дизайна были созданы специальные приложения и сервисы, с помощью которых веб-дизайнеры делают представление сайта по техническому заданию. В дальнейшем разработчики при создании используют эти представления для создания клиентской части сайта.

В данном направлении разработки так же существует много сервисов и приложений, например:

- Figma;
- Blocs Website Builder;
- InVision;
- Turbologo;
- Zeplin.

Для задач этой работы лучше всего подходит Figma.

1.2.5 Базы данных

Существует два типа баз данных: реляционные и нереляционные, их основное отличие в том, что в реляционных базах данных все данные хранятся в виде столбцов и строк. В нереляционных базах данных в свою очередь информация храниться в определенных типах, таких как: JSON, графы, ключи значений.

Для задач данной работы подходят реляционные базы данных, популярными представителями являются:

- MySQL;
- Oracle RDBMS;
- PostgreSQL;

– Microsoft SQL Server.

Из этих баз данных лучше всего подходит MySQL, из-за большой популярности, простоты освоения и использования, так же будет использоваться ответвление MariaDB.

1.2.6 Фреймворк для серверной части (BackEnd, RESTAPI)

Для разработки серверного компонента зачастую используют следующие фреймворки:

– Express;

– Spring Boot;

– Node.js;

– GraphQL.

Для разработки электронного образовательного ресурса больше всего подходит Node.js, эта система предназначена для асинхронной обработки, поэтому серверный компонент лучше работает с большим количеством пользователей. Этот фреймворк использует синтаксис JavaScript, упрощается его освоение и использования. Node.js имеет полную поддержку JSON формата файлов, что пригодиться в дальнейшей разработке.

1.2.7 Фрейморк для клиентской части (FrontEnd)

– React JS;

– Vue JS;

– Angular;

– Ember JS;

– Preact.

Лучшим выбором является Vue.js, потому что имеет большую скорость отрисовки чем другие, имеет реактивную модель, которая обновляет представление при изменении данных, и при установке имеет встроенные критически важные компоненты. Так же во vue используется отображение отдельных компонентов, а не целых страниц, что позволяет более точно и гибко настроить итоговую страницу, которая будет видна пользователям.

1.2.8 Интегрированная среда разработки

Интегрированная среда разработки – комплекс программных средств для разработки программного обеспечения. Такие среды работают по принципу «все в одном», то есть одна программа содержит в себе одно или несколько средств для развертывания баз данных, запуска локальных серверов. Так же такие среды могут содержать в себе встроенные редакторы кода.

- Laragon;
- Open Server Panel;
- WampServer;
- XAMPP;
- MAMP.

Для данной разработки электронного образовательного ресурса подходит OpenServerPanel – интегрированная среда разработки имеет многофункциональность, поддержку большого количества языков интерфейса, просмотр логов в реальном времени, и конечно же быстроту запуска.

1.2.9 Проектировщик ER-диаграмм

ER–диаграммы применяются для проектировки структур реляционных баз данных: как будут взаимодействовать таблицы между собой, какой компонент будет первичным ключом по которому можно однозначно определить каждую строку в таблице. Есть множество программ, при помощи которых можно создать ER–даиграммы.

- Er assistant;
- Software Ideas Modeler;
- Edraw.

Самой подходящей является Er assistant – она наиболее приспособлена для создания ER–даиграмм баз данных, имеет простой не нагруженный интерфейс, в котором легко разобраться.

1.2.10 Дополнительные инструменты

Canvas – элемент HTML который предназначен для создания растовых изображений используя скрипты языка JavaScript.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Проектирование программного изделия

2.1.1 Разработка названия электронного образовательного ресурса

Имя продукта одна из важнейших деталей, так как имя будет напрямую будет ассоциироваться с продуктом и с его производителем.

Так же если имя уникально и необычно, то оно лучше запоминается у потребителя.

Отталкиваясь от темы выпускной квалификационной работы, за основную идею имени был выбран оксида лития, это лучше всего ассоциирует электронный образовательный ресурс для использования в образовательной организации (на примере темы «Молекулярная физика. Термодинамика» дисциплины «Физика») с наукой. Для комфортного и современного произношения, название было переведено на английский язык: «Lithium oxide».

2.1.2 Разработка логотипа электронного образовательного ресурса

Логотип является неотъемлемой частью любой компании и продукта, логотип крепко ассоциируется с продуктом или компанией. Благодаря ему можно выделиться на рынке среди конкурентов. Логотип обеспечивает визуальный контакт с потребителем. Если логотип индивидуален и не похож на другие, то это показывает отношение компании к своему продукту, и показывает пользователю, что производитель/разработчик заботятся о своих клиентах. Логотип – это прямое продолжение имени продукта или компании.

В первую версию логотипа (Рисунок 3) легло имя оксида, атом физики, и символ пустого множества (Рисунок 4).

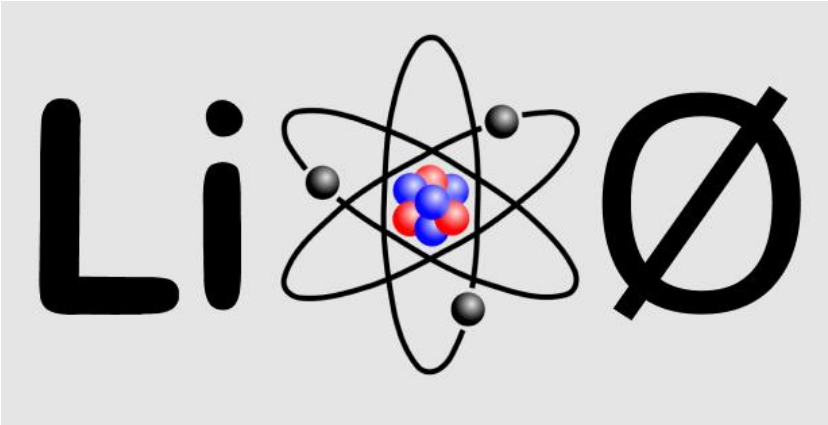

Рисунок 3 – Первая версия логотипа

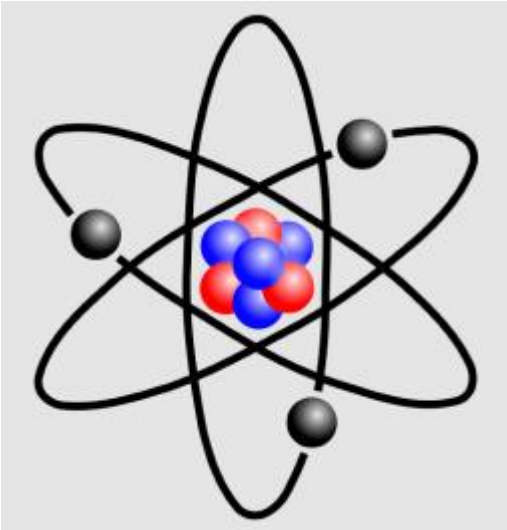

Рисунок 4 – Атом физики

После более подробного изучения современных логотипов электронных образовательных ресурсов, была создана новая версия логотипа, который был изменен и осовременен: были убраны острые углы, атом физики был сделан черно-белым, из-за чего весь логотип стал полностью черным (Рисунок 5).

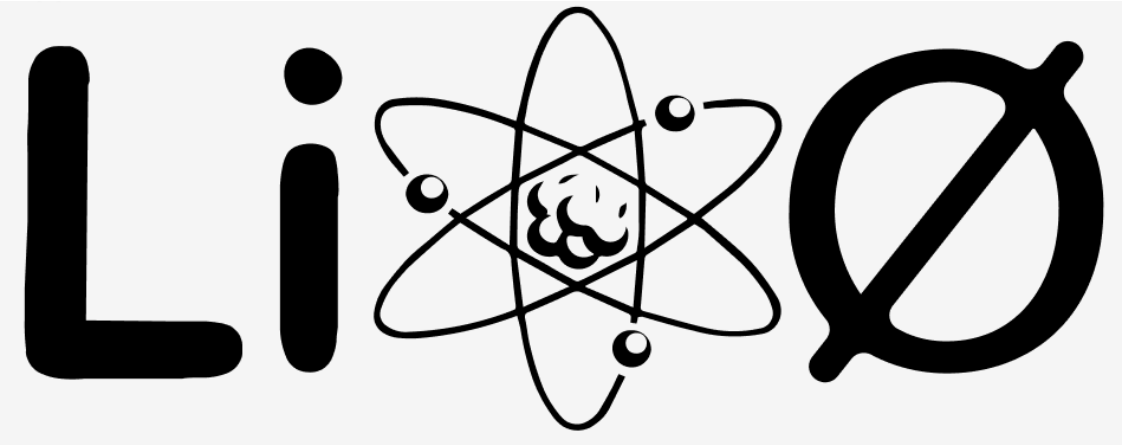

Рисунок 5 – Обновленная версия логотипа

2.1.3 Разработка дизайна электронного образовательного ресурса

Для разработки дизайна интерфейса будет использоваться ранее выбранная Figma. Работа над дизайном и прототипом были созданы следующие страницы электронного образовательного ресурса:

1) Страница, на которой реализована авторизация (Рисунок 6);

2) Главная страница, на которой можно выбрать доступные уроки (Рисунок 7);

3) Страница урока по теме «Термодинамика» (Рисунок 8);

4) Страница урока по теме «Молекулярная физика» (в рамках работы над прототипом, функционал страницы не разрабатывался).

Ограниченный по функционалу прототип сайта нужен для понимания дальнейшей разработки электронного образовательного ресурса.

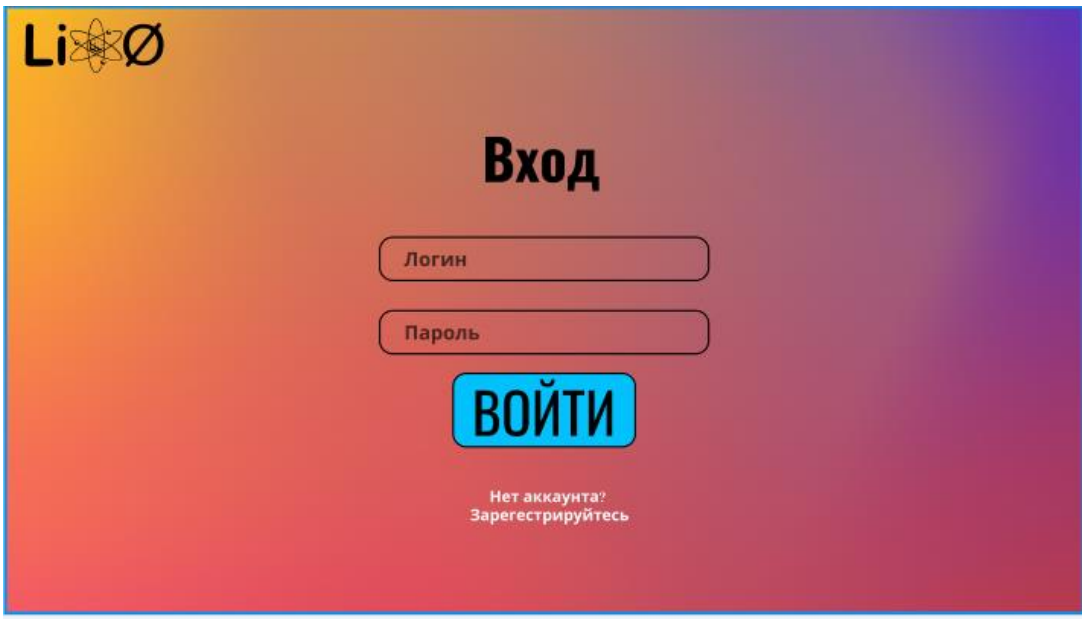

Рисунок 6 – Страница авторизации

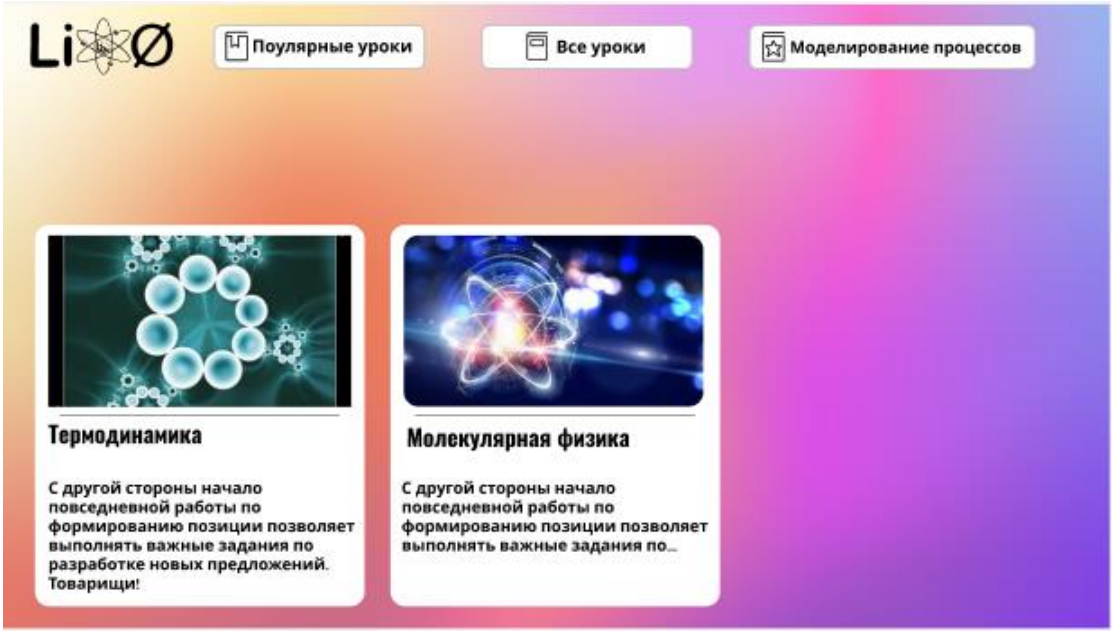

Рисунок 7 – Страница доступных уроков

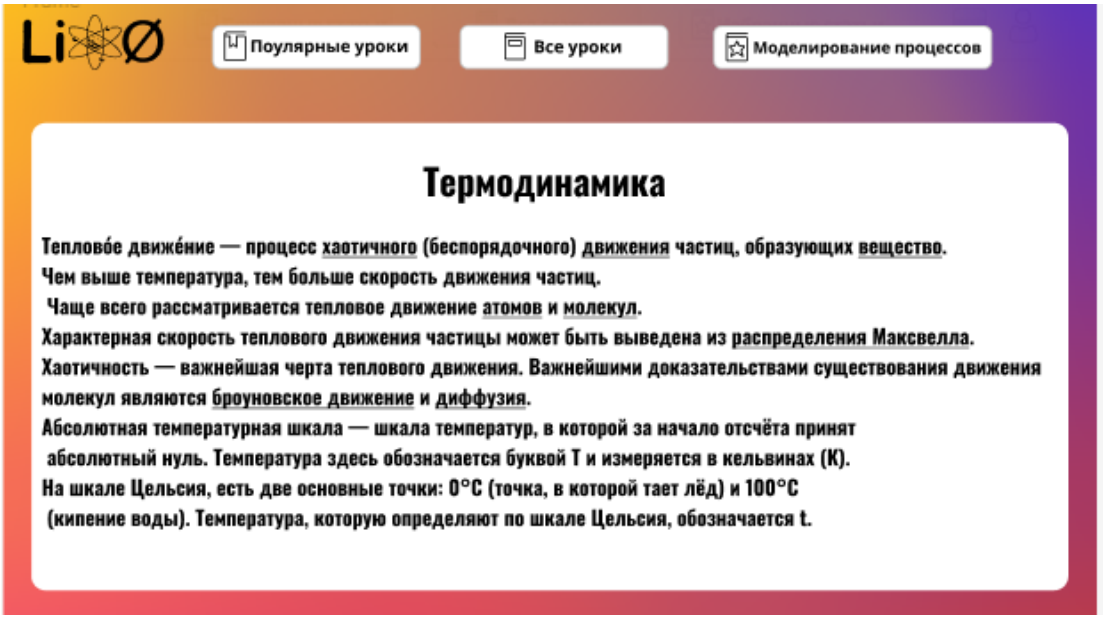

Рисунок 8 – Страница урока по теме «Термодинамика»

Дизайн был вдохновлен несколькими ресурсами:

– https://duolingo.com (Рисунок 9);

– https://stepik.org (Рисунок 10);

– https://razoom.mgutm.ru (Рисунок 11).

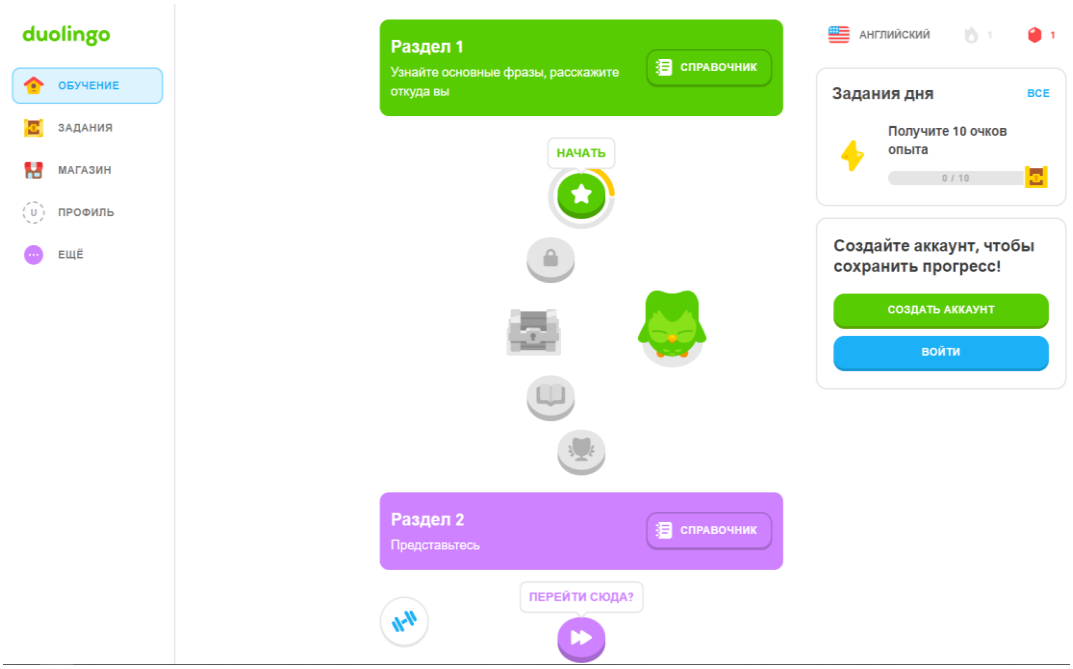

Рисунок 9 – Электронный образовательный ресурс «duolingo»

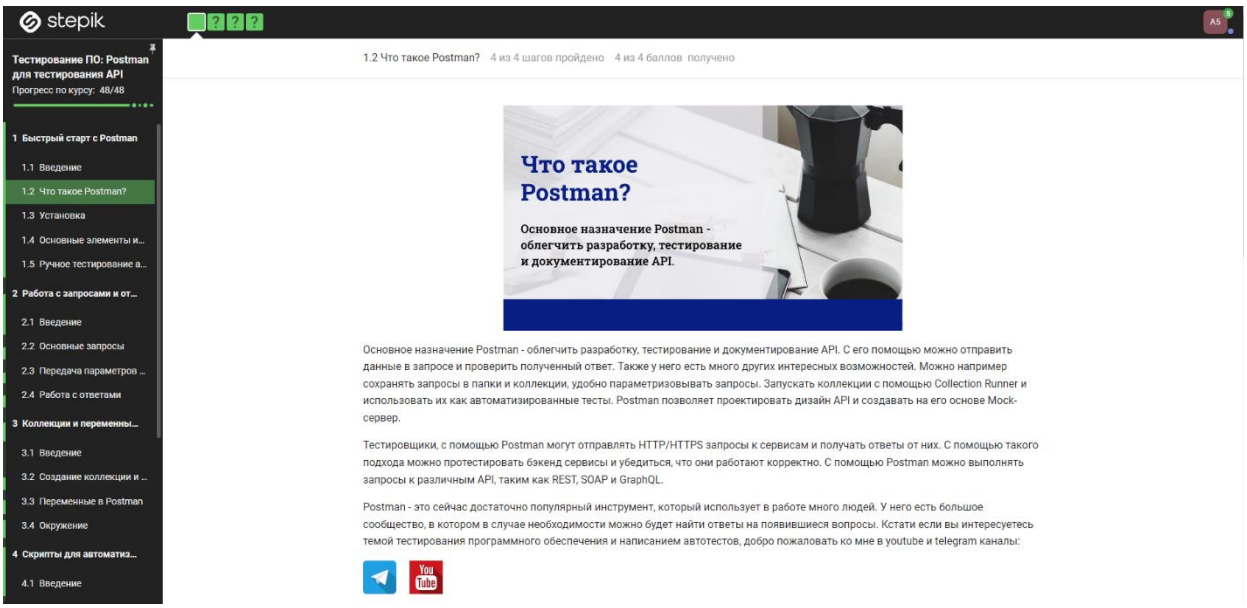

Рисунок 10 – Электронный образовательный ресурс «stepik»

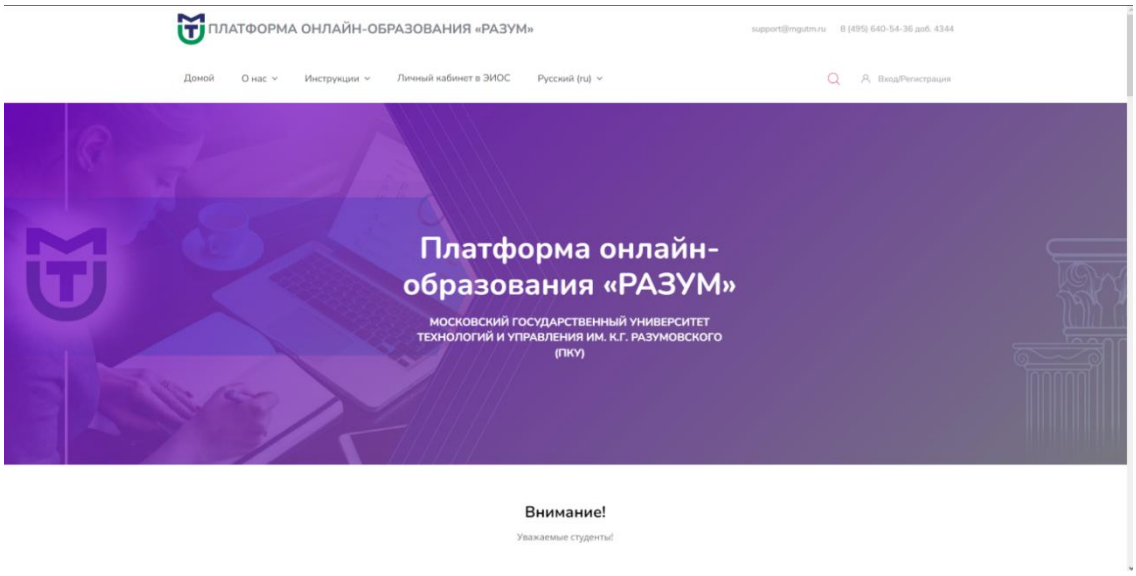

Рисунок 11 – Платформа онлайн-образования «РАЗУМ»

2.1.4 Проектировка базы данных

Для создания структуры будущей базы данных была сделана ER–диаграмма (Рисунок 12).

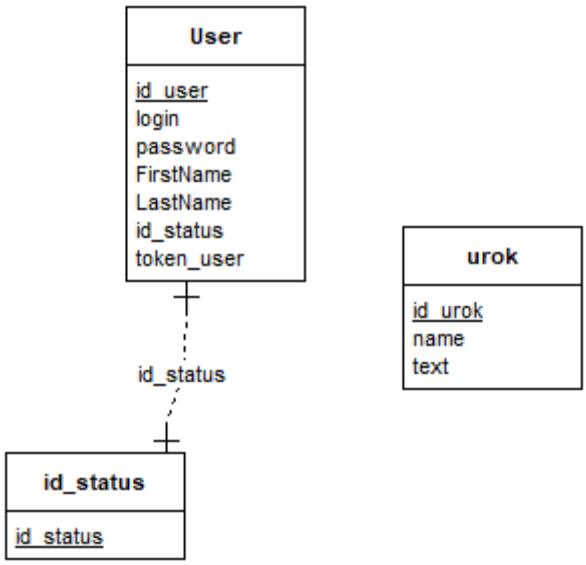

Рисунок 12 – ER–диаграмма базы данных

Так как в разрабатываемом электронном образовательном ресурсе все уроки доступны для всех зарегистрированных пользователей, таблицы «User» и «Urok» никак не связаны. У каждого пользователя есть статус: пользователя или администратора, который может создавать новые уроки.

2.2 Разработка программного обеспечения

2.2.1 Разработка базы данных

Для создания базы данных будет использоваться компонент OpenServerPanel под названием PhpMyAdmin. После запуска OpenServerPanel и открытия PhpMyAdmin, система предлагает нам авторизоваться, по умолчанию стоит логин «root», а пароля нет. В перспективе разработки проекта — это небезопасно, поэтому в главном меню нужно сменить логин и пароль для доступа к базе данных (Рисунок 13).

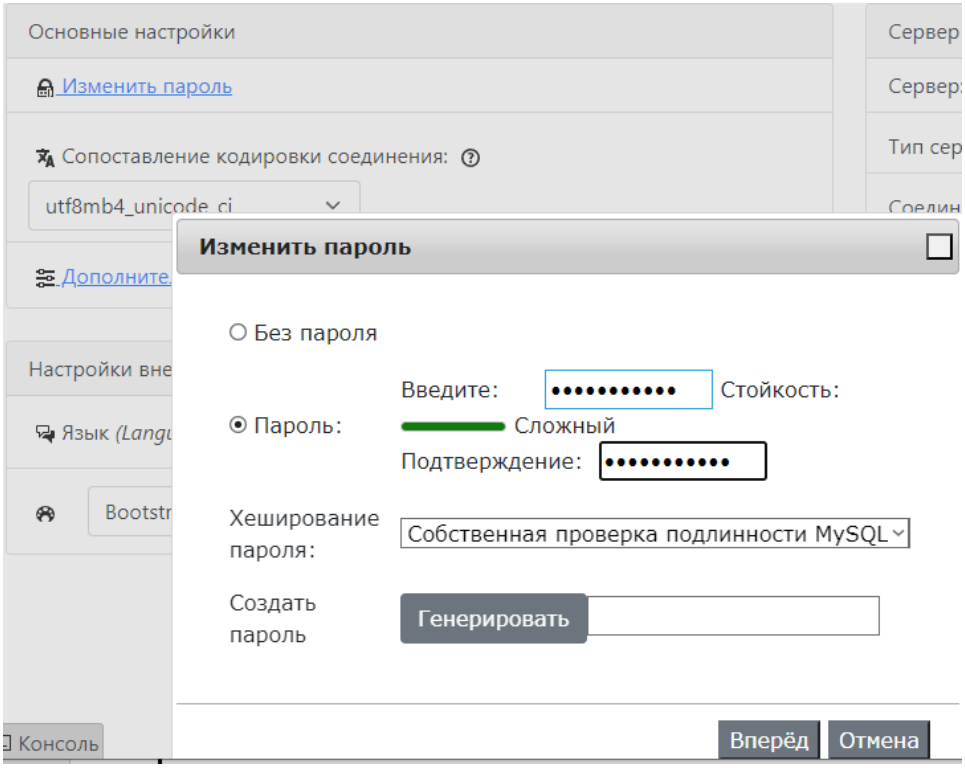

Рисунок 13 – Смена пароля для доступа к базе данных

После этого можно приступать к созданию самой базы данных по ER–диаграмме, сделанной ранее (Рисунок 12).

Были созданы таблицы: «user» (Рисунок 14), «urok» (Рисунок 15), «id_status» (Рисунок 16).

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	**id_user**	int(128)			Нет	*Нет*		AUTO_INCREMENT
☐	2	**login**	char(64)	*utf8mb4_unicode_ci*		Нет	*Нет*		
☐	3	**password**	varchar(96)	*utf8mb4_unicode_ci*		Нет	*Нет*		
☐	4	**FirstName**	varchar(24)	*utf8mb4_unicode_ci*		Нет	*Нет*		
☐	5	**LastName**	varchar(24)	*utf8mb4_unicode_ci*		Нет	*Нет*		
☐	6	**id_status**	int(1)			Нет	*Нет*		
☐	7	**token_user**	varchar(64)	*utf8mb4_unicode_ci*		Нет	*Нет*		

Рисунок 14 – Структура таблицы «user»

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	**id_urok**	int(128)			Нет	*Нет*		AUTO_INCREMENT
☐	2	**name**	varchar(64)	*utf8mb4_unicode_ci*		Нет	*Нет*		
☐	3	**text**	varchar(256)	*utf8mb4_unicode_ci*		Нет	*Нет*		

Рисунок 15 – Структура таблицы «urok»

	#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно
☐	1	**id_status**	int(1)			Нет	*Нет*		AUTO_INCREMENT
☐	2	**status**	char(11)	*utf8mb4_unicode_ci*		Нет	*Нет*		

Рисунок 16 – Структура таблицы «id_status»

В таблице «user» (Рисунок 14) присутствует строка «login», ей присвоен уникальный ключ, который не дает создать повторяющиеся логины пользователей. В базе данных создано разделение прав пользователей, данная функция реализована в таблице «id_user» (Рисунок 16), вручную добавлены две строки, которые отвечают за права пользователей (Рисунок 17).

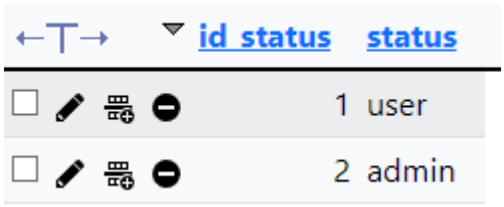

Рисунок 17 – Разграничение прав в базе данных

По умолчанию всем пользователям присваивается значение статуса 1, означает – обычный пользователь и не имеет прав на редактирование контента.

Функции администратора пока не реализованы в проекте, и все пользователи являются простыми читателями, но в будущем планируется добавление функционала администратора.

2.2.1 Разработка электронного образовательного ресурса

После создания базы данных идет разработка серверного компонента с архитектурой REST API, для реализации потребуется два файла, в одном будут прописаны URL для обращения к серверу, а также будет производиться парсинг данных (преобразование данных в формат JSON). Второй же файл будет содержать сами методы получения данных из базы данных в которых будут указаны SQL запросы.

Разработка серверного компонента была начата с объявления констант подключения к базе данных (Рисунок 18).

```
const connection = mysql.createConnection({
    host: "127.0.0.1",
    user: "root",
    database: "eor",
    password: "1lc9go4xZ3_"
});


const pool = mysql.createPool({
  connectionLimit: 100,
  host: "127.0.0.1",
  user: "root",
  database: "eor",
  password: "1lc9go4xZ3_"
});
```

Рисунок 18 – Константы подключения к базе данных

Такое решение позволяет избежать нагромождение кода, и в случае чего, данные константы можно спрятать в части DOM’а проекта недоступную обычному пользователю в целях безопасности.

Так же была создана константа «pool» она практически идентична с константой подключения, однако имеет ограничение на подключение, сделано это специально, что бы пользователь мог запрашивать определенное количество раз одни и те же данные.

Далее создается метод, который будет отправлять SQL запрос на создание новой записи в таблицу «user» в базу данных (Рисунок 19), так же что бы этот метод корректно работал нужно во втором файле, в котором парсятся данные, создать. Для дополнительной безопасности данных, пароли хэшируются (шифруются).

```
async createUser(id_user, FirstName, LastName, login, password) {
    try {
        return new Promise((resolve, reject) => {
        connection.connect((err) => {
        connection.query(
          'INSERT INTO user SET ?',
          {
              id_user: id_user,
              FirstName: FirstName,
              LastName: LastName,
              login: login,
              password: hashPassword(password)

          },
          (err, results, fields) => {
              if (err) {
                  reject(err);
              } else {
                  resolve(results.insertId);
              }
              connection.end((err) => {
                  if (err) {
                      console.log(err);
                  } else {
                      console.log('Database closed');
                  }
              });
          });
    })
    });
    } catch(err) {
        throw (err);
    }
  }
```

Рисунок 19 – Метод регистрации нового пользователя

```
else if (req.url === "/registration" && req.method === "POST") {
let user_data = await getReqData(req);
const user = JSON.parse(user_data);
let newUser = await new Urok().createUser(user.id_user, user.FirstName, user.LastName, user.login, user.password);
    res.writeHead(200, {
        "Content-Type": "application/json",
        'Access-Control-Allow-Origin': '*',
        'Access-Control-Allow-Methods': '*',
        'Access-Control-Allow-Headers': '*'
    });
res.end(JSON.stringify({ id: newUser }));
}
```

Рисунок 20 – Путь для обращения к серверу для регистрации

После надо создать метод авторизации пользователя, который при успешной авторизации будет генерировать токен пользователя и записывать его в базу данных (Рисунок 21, Рисунок 22).

```
else if (req.url === "/login" && req.method === "POST") {
    let user_data = await getReqData(req);
    const user = JSON.parse(user_data);
    let LoginUser = await new Urok().loginUser(user.login, user.password);
    res.writeHead(200, {
        "Content-Type": "application/json",
        'Access-Control-Allow-Origin': '*',
        'Access-Control-Allow-Methods': '*',
        'Access-Control-Allow-Headers': '*'
    });
    res.end(JSON.stringify({ token_user: LoginUser }));
}
```

Рисунок 21 – Метод авторизации пользователя

```
async loginUser(Login, Password) {
  return new Promise((resolve, reject) => {
    connection.query(
      'SELECT id_user FROM user WHERE login = ? and password = ?',
      [Login, hashPassword(Password)],
      (err, results, fields) => {
        if (err) {
          reject(err);
        } else if (results.length === 0) {
          const error = new Error('Invalid credentials');
          error.status = 401;
          return (error);
        } else {
          const generate_token = (length) => {
            const a = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890".split("");
            const b = [];
            for (let i = 0; i < length; i += 1) {
              let j = Math.floor(Math.random() * (a.length - 1));
              b.push(a[j]);
            }
            return b.join('');
          };
          const token_user = generate_token(32);
          const id_user = results[0].id_user;
          console.log(token_user);
          connection.query(
            'UPDATE user SET token_user = ? WHERE id_user = ?',
            [token_user, id_user],
            (err, results, fields) => {
              if (err) {
                reject(err);
              } else {
                resolve(token_user);
              }
            }
          );
        }
      }
    );
  }).catch(error => {
    console.error('An error occurred:', error);
    if (error.status) {
      return Promise.reject(error);
    } else {
      return Promise.reject(new Error('Internal server error'));
    }
  });
}
```

Рисунок 22 – Путь для обращения к серверу при авторизации

После того как пользователь авторизовался, ему выдался токен, по которому уже можно получить все записи из таблицы «urok». Наличие токена будет проверятся на стороне сервера (Рисунок 24).

```
async getUroks(token_user) {
  return new Promise((resolve, reject) => {
    if (!token_user) {
      reject(new Error('Unauthorized'));
      return;
    }
    pool.getConnection((err, connection) => {
      if (err) {
        console.error('Database connection error:', err);
        reject(err);
        return;
      }
      connection.query('SELECT * FROM urok', (err, results, fields) => {
        connection.release();

        if (err) {
          console.error('Database query error:', err);
          reject(err);
          return;
        }

        console.log('Database query executed successfully');
        resolve(results);
      });
    });
  });
}
```

Рисунок 23 – Метод получения уроков по токену

Так же понадобиться выводить по одной записи из базы данных для просмотра страницы с каждым уроком (Рисунок 25).

```
if (req.url.match(/\/lessons\/([0-9]+)/) && req.method === "GET") {
    try {
        const id = req.url.split("/")[2];
        const urok = (await new Urok().getUroks("a")).find(iter => iter.id_urok == id);
        res.writeHead(200, {
            "Content-Type": "application/json",
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': '*',
            'Access-Control-Allow-Headers': '*'
        });
        res.end(JSON.stringify(urok));
    } catch (error) {
        res.writeHead(404, {
            "Content-Type": "application/json",
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': '*',
            'Access-Control-Allow-Headers': '*'
        });
        res.end(JSON.stringify({ message: error }));
    }
}
```

Рисунок 24 – Получение одного урока из базы данных

После создания серверного компонента можно приступать к созданию клиентской части приложения, которая будет содержать в себе следующие страницы:

– регистрация;

– авторизация;

– главная страница;

– страница на которой моделируется движение молекул;

– страница для каждого урока которая будет сама создаваться для каждого урока.

При использовании vue.js можно создавать spa приложения, для этого нужно сначала создать и настроить роутер приложения (Рисунок 26).

```
<template>
  <div>
    <SignIn @enter="$event => step = 'Mode-Prescription'" v-if="step === 'Mode-SignIn'" />
    <SignUp @login="$event => step = 'Mode-SignIn'" v-if="step === 'Mode-SignUp'" />
    <Prescription @model="$event => step= 'Mode-Model'" v-if="step === 'Mode-Prescription'" @view-lesson="showLesson" />
    <Lesson @main="$event => step = 'Mode-Prescription'" v-if="step == 'Mode-View-Lesson'" :lesson="lesson" />
    <!-- <Model  v-if="step === 'Mode-Model'" :model="model" /> -->
    <Model @main="$event => step = 'Mode-Prescription'" v-if="step === 'Mode-Model'" />
    <!-- <Model @model="step = 'Mode-Model'" v-if="step === 'Mode-SignUp'" /> -->
  </div>
</template>

<script>
import SignUp from "./signup.vue";
import SignIn from "./signin.vue";
import Prescription from "./prescription.vue";
import Lesson from './lesson.vue';
import Model from './model.vue';


export default {
  components: { SignIn, SignUp, Prescription, Lesson, Model },
  data() {
    return {
      step: "Mode-SignUp",
      lesson: 0
    };
  },
  beforeMount() {
      const parts = location.pathname.split("/");
      if (parts[1] == "lesson") {
        this.lesson = parts[2];
        this.step = "Mode-View-Lesson";
      } else {
        this.step = "Mode-SignUp";
      }
  },
  methods: {
    handleEnter() {
      this.step = "Mode-Prescription";
    }
  },
    showLesson(id) {
    this.lesson = id;
    this.step = "Mode-View-Lesson";
  },
  goToModel(){
    this.step = "Mode-Model";
  }


};
</script>
```

Рисунок 25 – Роутер приложения

Функция роутера понятна из его названия – роутер направляет пользователя по компонентам, грубо говоря роутер — это холст на котором уже отображаются компоненты, роутер сам по себе не отображается, но на себе отображает компоненты (Рисунок 26).

Начальным компонентом, который будет отображаться является компонент регистрации (Рисунок 27).

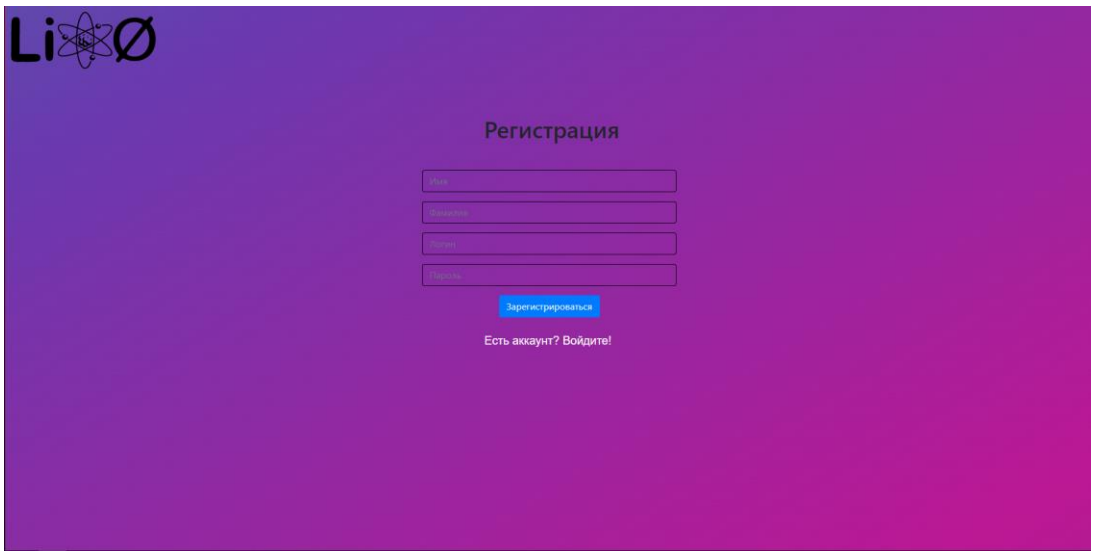

Рисунок 26 – Компонент регистрации

В компоненте передаются данные на сервер JSON файлом по средствам вызова метода «SignUp» в форме регистрации при нажатии кнопки зарегистрироваться (Рисунок 28) который в себе содержит метод JavaScript «fetch ()» (Рисунок 29).

```
<div class="mb-3">
    <input v-model="FirstName" type="FirstName" class="form-control" id="FirstName" placeholder="Имя" aria-describedby="FirstName">
</div>

<div class="mb-3">
    <input v-model="LastName" type="LastName" class="form-control" id="LastName" placeholder="Фамилия" aria-describedby="LastNmae">
</div>

<div class="mb-3">
    <input v-model="login" type="login" class="form-control" id="login" placeholder="Логин" aria-describedby="login">
</div>

<div class="mb-3">
    <input v-model="password" type="password" class="form-control" placeholder="Пароль" id="password">
</div>

<button @click="SignUp" type="button" class="btn btn-primary" style="position: relative; left: 50%;transform: translate(-50%, 0);">Зарегистрироваться</button>
```

Рисунок 27 – Форма регистрации

```
export default {
    data() {
        return {
            FirstName: '',
            LastName: '',
            login: '',
            password: ''
        }
    },


    emits: ['login'],
    methods: {
        SignUp() {
            const res = fetch('http://localhost:5000/registration', {
                method: 'Post',
                headers: {
                    'Content-Type': 'application/json'
                },


                body: JSON.stringify({
                    id_user: this.id_user,
                    FirstName: this.FirstName,
                    LastName: this.LastName,
                    login: this.login,
                    password: this.password

                })
            })
            res.then().catch()
            console.log(res)

        }
    }
}
```

Рисунок 28 – Передача данных при помощи метода «SignUp»

Так же в компоненте присутствует переадресация на компонент авторизации (Рисунок 30) в случае если у пользователя уже есть аккаунт.

Компонент авторизации работает по такому же принципу, как и регистрация: передается JSON файл при помощи метода «SignIn» который содержит в себе JavaScript метод «fetch» (Рисунок 31).

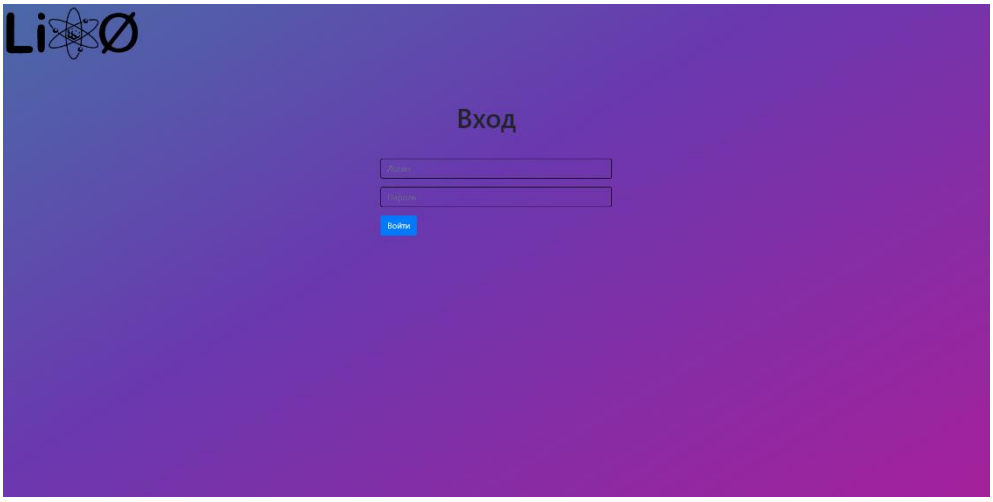

Рисунок 29 – Компонент авторизации

```
export default {
  data() {
    return {
      login: "",
      password: "",
      error: ""
    };
  },
  methods: {
    signIn() {
      const currentObject = this;
      fetch("http://localhost:5000/login", {
        method: "POST",
        headers: {
          "Content-Type": "application/json"
        },
        body: JSON.stringify({
          login: this.login,
          password: this.password
        })
      })
        .then(res => res.json())
        .then(function(res) {
          if (res.error) {
            currentObject.error = res.error;
          } else {
            localStorage.setItem("token", res.token_user);
            currentObject.$emit("enter");
          }
        })
        .catch(function(error) {
          console.error("An error occurred:", error);
          currentObject.error = "Ошибка авторизации";
        });
    }
  }
};
```

Рисунок 30 – Передача данных при помощи метода «SignIn»

В случае успешной авторизации пользователя перенаправляет на компонент главной страницы, на котором показаны все уроки, так же данный компонент обладает реактивностью что позволяет моментально отображать новые записи на странице, без обновления пользователем самой страницы (Рисунок 32).

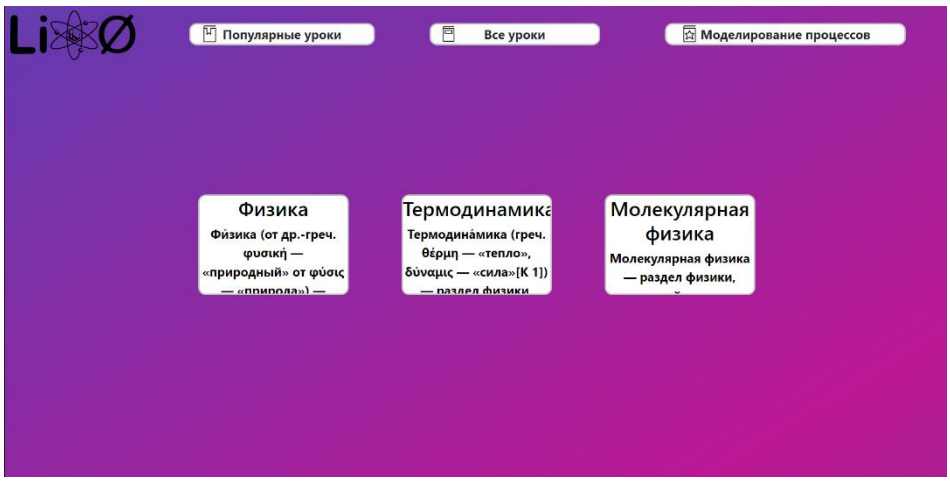

Рисунок 31 – Компонент главной страницы

Так же каждой записи присваивается свой URL адрес исходя из их id (Рисунок 33), при переходе по этой ссылке вместо стартового компонента отображается компонент с содержимым одного урока (Рисунок 34).

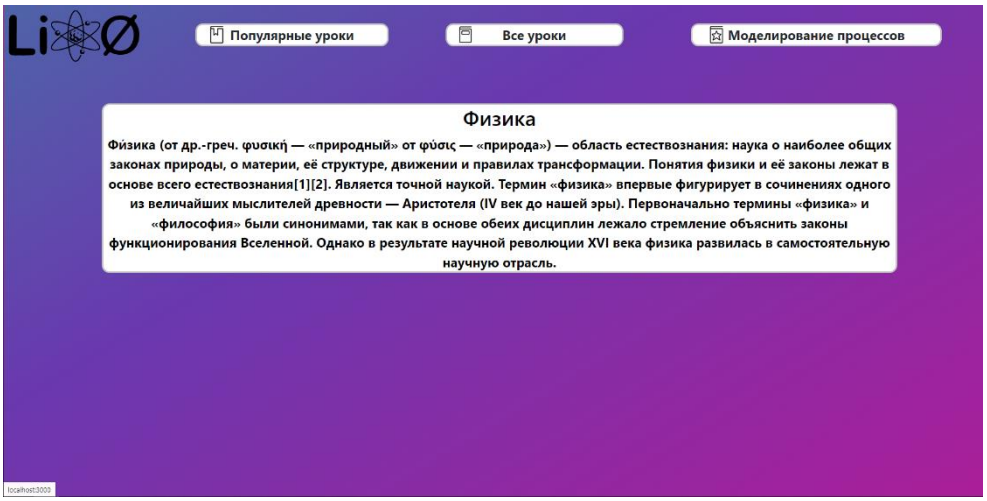

Рисунок 32 – Компонент с содержимым страницы

Так же было реализовано моделирование теплового движения молекул в зависимости от давления, температуры, объема (Рисунок 36). Перейти к нему можно при нажатии на вкладку «Моделирование процессов».

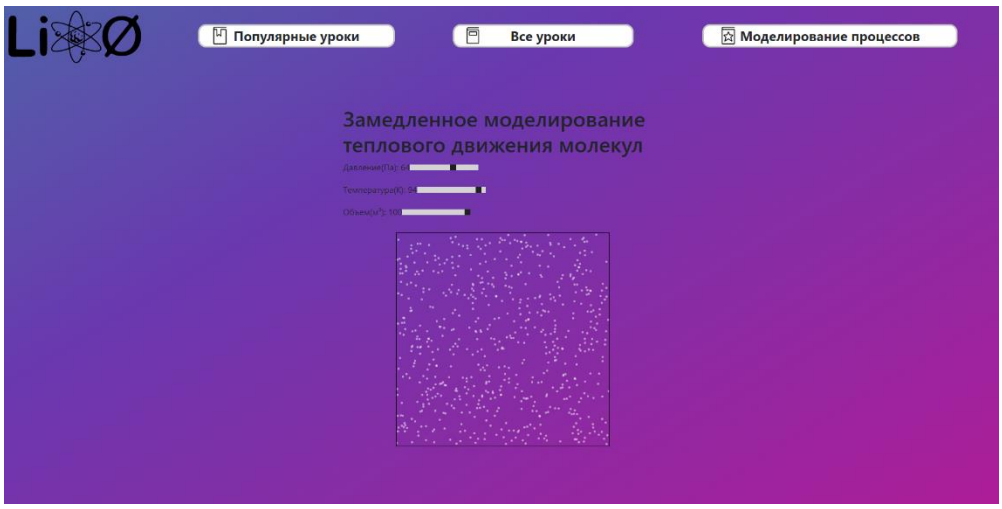

Рисунок 33 – Компонент с моделированием процесса

```
methods: {
  initializeParticles() {
    this.particles = [];

    const numParticles = Math.floor((this.pressure / 100) * 1000);
    const particleRadius = 2;

    for (let i = 0; i < numParticles; i++) {
      const x =
        Math.random() * (this.$refs.canvas.width - particleRadius * 2) +
        particleRadius;
      const y =
        Math.random() * (this.$refs.canvas.height - particleRadius * 2) +
        particleRadius;
      const dx = (Math.random() - 0.5) * (this.temperature / 25);
      const dy = (Math.random() - 0.5) * (this.temperature / 25);
      const color = "rgba(255, 255, 255, 0.5)";

      this.particles.push({ x, y, dx, dy, radius: particleRadius, color });
    }
  },

  startAnimation() {
    this.animate();
  },
  stopAnimation() {
    cancelAnimationFrame(this.animationFrameId);
  },
  animate() {
    this.canvasContext.clearRect(
      0,
      0,
      this.$refs.canvas.width,
      this.$refs.canvas.height
    );

    this.particles.forEach((particle) => {
      // Обновление позиции частицы
      particle.x += particle.dx;
      particle.y += particle.dy;

      // Обработка столкновений с границами холста
      if (
        particle.x + particle.radius > this.$refs.canvas.width ||
        particle.x - particle.radius < 0
      ) {
        particle.dx = -particle.dx;
      }
      if (
        particle.y + particle.radius > this.$refs.canvas.height ||
        particle.y - particle.radius < 0
      ) {
        particle.dy = -particle.dy;
      }
```

Рисунок 34 – Скрипт для моделирования

После завершения проекта, его репозиторий размещается на GitHub используя систему контроля версий Git. (Рисунок 36).

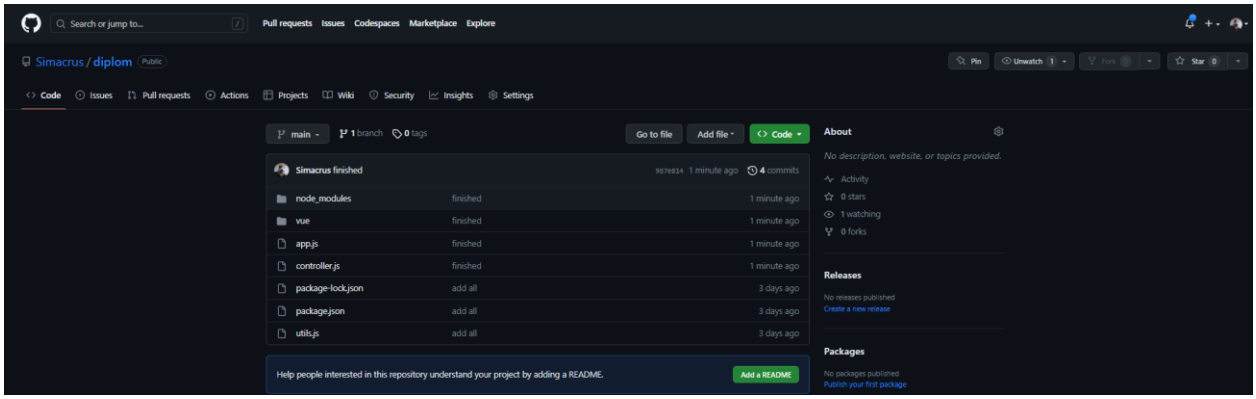

Рисунок 35 – Размещенный проект на платформе GitHub

2.3 Расчет экономической эффективности разработки сервиса

При разработке велся учет расходов, для расчета экономической эффективности разработки сервиса.

Экономическая эффективность (англ. Economic efficiency) — это величина, определяемая соотношением полученных результатов деятельности человека, производства продукции (товаров или услуг) и затрат труда и средств на производство.

На основании расчетов экономической эффективности будут определены трудозатраты по реализации проекта и полная его стоимость.

Определение затрат производится способом подсчета плановой себестоимости. В первую очередь следует рассчитать все плановые затраты, которые связанны с разработкой и не зависимы от источников финансирования. Расчет был основан на почасовой ставке HTML-верстальщиков на фрилансе. Часовая тарифная ставка – фиксированная оплата труда работника в единицу времени за выполнение нормы установленной сложности (квалификации). Это только вознаграждение за выполненную работу и не включает в себя дополнительные выплаты, компенсации, пособия и социальные выплаты.

Фрилансер – свободный специалист, который трудится на себя. Он самостоятельно ищет заказчиков, а также устанавливает себе график. Трудиться могут как с одним, так и сразу с несколькими заказчиками.

Средняя ставка дизайнера Figma 200 р/час.

Средняя ставка Vue,js разработчика составляет 200р/час.

Средняя ставка backend разработчика на Node.js составляет 166р/час

Стоит отметить, что разработка сервиса осуществлялась на протяжении двух недель.

Таблица 1 – Расчет стоимости создания сервиса

Виды работ	Затраты времени, час.	Стоимость работ, руб.
Разработка дизайна	20	4000
Разработка серверной части и базы данных	30	5000
Разработка клиентской части на Vue.js	20	4000
Итого	70	13000

В результате проделанных расчетов можно сделать выводы:

– итоговое количество часов, которое было затрачено на разработку равно 70 часам,

– заработная плата составляет 13000 рублей.

ЗАКЛЮЧЕНИЕ

Цель первой части дипломного проекта заключалась в исследовании предметной области с целью определения специфики разрабатываемого сервиса. Производилось изучение понятий физики.

Далее был произведен анализ и выбор инструментальных средств, используемых при разработке сервиса.

Для реализации сайта были выбраны следующие инструментальные средства:

– JavaScript;

– Visual Studio Code;

– Git;

– Figma;

– MySQL;

– NodeJS;

– VueJS;

– Open Server Panel;

– Er assistant.

В практической части проекта был описан процесс проектировки и разработки программного продукта, включающая следующие действия:

– разработка названия;

– разработка логотипа;

– проектировка и создание базы данных;

– подключение базы данных;

– создание серверной части;

– создание клиента и наполнение контентом.

В продолжении практической части проекта проанализирована сумма всех затрат на разработку электронного образовательного ресурса, произведено прогнозирование показаний экономической деятельности в результате

использования сервиса и определён срок окупаемости полученных затрат.

Итоговая версия электронного образовательного ресурса предоставляет возможность регистрации и авторизации пользователя, просмотр всех уроков, возможность просмотра каждого урока по отдельности, возможность просмотра и взаимодействия с моделью частиц.

В заключении можно отметить, что данный электронный образовательный ресурс исходя из экономических расчетов эффективности является не только экономически выгодным, но и удобным для использования в образовательной организации.

СПИСОК ЛИТЕРАТУРЫ

1. Понятие сервиса – https://habr.com/ru/articles/466455/ (дата обращения 24.05.2023).
2. Как создать свою социальную сеть – https://sibdev.pro/blog/articles/kak-sozdat-svoyu-socialnuyu-set (дата обращения 26.05.2023).
3. Работы Джеймса Барнса – https://www.gutenberg.org/ebooks/author/38095 (дата обращения 27.05.2023).
4. Киберсоциализация человека за авторством Плешакова В.А. – http://homocyberus.ru/sites/default/files/homocyberus.pdf (дата обращения 27.05.2023).
5. Основы HTML – Изучение веб-разработки – URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/HTML_basics (дата обращения 29.05.2023).
6. How to Learn HTML – Справочник – URL: https://medium.com/codex/how-to-learn-html-in-2023-a-comprehensive-guide-54813d5ca496 (дата обращения 29.05.2023).
7. Основы CSS – Изучение веб-разработки – URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/CSS_basics (дата обращения 29.05.2023).
8. CSS a Handy Guide – Справочник – URL: https://medium.com/free-code-camp/the-css-handbook-a-handy-guide-to-css-for-developers-b56695917d11 (дата обращения 29.05.2023).
9. JavaScript – Глоссарий – URL: https://developer.mozilla.org/ru/docs/Glossary/JavaScript (дата обращения 29.05.2023).
10. JavaScript – Справочник – URL: https://medium.com/free-code-camp/the-complete-javascript-handbook-f26b2c71719c (дата обращения 01.06.2023).

11. JavaScript – Современный учебник JavaScript – URL: https://learn.javascript.ru/ (дата обращения 01.06.2023).

12. NodeJS – Справочник Node.js – URL: https://nodejsdev.ru/ (дата обращения 01.06.2023).

13. Что такое NodeJS: для чего он нужен, преимущества – Блог – URL: https://loftschool.com/blog/posts/node-java-script (02.06.2023).

14. Как тестирование помогает улучшить код – Руководство – URL: https://procodings.ru/dev-ru/kak-testirovanie-pomogaet-nam-uluchshit-nash-kod/ (дата обращения 02.06.2023).

15. VueJS – Глоссарий – URL: https://vuepress.vuejs.org/miscellaneous/glossary.html (дата обращения 02.06.2023).

16. VueJS – Справочник – URL: https://medium.com/free-code-camp/the-vue-handbook-a-thorough-introduction-to-vue-js-1e86835d8446 (дата обращения 03.06.2023).

17. jQuery – Глоссарий – URL: https://learnjavascript.co.uk/jq/reference/jqglossary.html (дата обращения 03.06.2023).

18. jQuery – Википедия – URL: https://ru.wikipedia.org/wiki/JQuery (дата обращения 03.06.2023).

19. Laragon – Руководство – URL: https://rajkhaan.medium.com/laragon-the-best-alternative-of-the-xampp-server-b0444720b8b6 (дата обращения 03.06.2023).

20. Laragon – Документация – URL: https://laragon.org/docs/ (дата обращения 03.06.2023).

ПРИЛОЖЕНИЕ А

Серверный компонент (controller.js)

```
const mysql = require("mysql");
const { hashPassword } = require("./utils");
const connection = mysql.createConnection({
    host: "127.0.0.1",
    user: "root",
    database: "eor",
    password: "1lc9go4xZ3_"
});
const pool = mysql.createPool({
  connectionLimit: 100,
  host: "127.0.0.1",
  user: "root",
  database: "eor",
  password: "1lc9go4xZ3_"
});
class Controller {
  async getUroks(token_user) {
    return new Promise((resolve, reject) => {
      if (!token_user) {
        reject(new Error('Unauthorized'));
        return;
      }
      pool.getConnection((err, connection) => {
        if (err) {
          console.error('Database connection error:', err);
          reject(err);
          return;
        }
        connection.query('SELECT * FROM urok', (err, results, fields) => {
          connection.release();

          if (err) {
            console.error('Database query error:', err);
            reject(err);
            return;
          }

          console.log('Database query executed successfully');
          resolve(results);
        });
      });
    });
  }
```

```
async createUser(id_user, FirstName, LastName, login, password) {
  return new Promise((resolve, reject) => {
      connection.connect((err) => {
          if (err) {
              reject(err);
          } else {
              console.log('Database connected');
              connection.query(
                  'INSERT INTO user SET ?',
                  {
                      id_user: id_user,
                      FirstName: FirstName,
                      LastName: LastName,
                      login: login,
                      password: hashPassword(password)

                  },
                  (err, results, fields) => {
                      if (err) {
                          reject(err);
                      } else {
                          resolve(results.insertId);
                      }
                      connection.end((err) => {
                          if (err) {
                              console.log(err);
                          } else {
                              console.log('Database closed');
                          }
                      });
                  });
          }
      })
  });
}
  async loginUser(Login, Password) {
    return new Promise((resolve, reject) => {
      connection.query(
        'SELECT id_user FROM user WHERE login = ? and password = ?',
        [Login, hashPassword(Password)],
        (err, results, fields) => {
          if (err) {
            reject(err);
          } else if (results.length === 0) {
            const error = new Error('Invalid credentials');
            error.status = 401;
            return (error);
          } else {
            const generate_token = (length) => {
              const a =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890".split("");
```

```
            const b = [];
            for (let i = 0; i < length; i += 1) {
              let j = Math.floor(Math.random() * (a.length - 1));
              b.push(a[j]);
            }
            return b.join('');
          };
          const token_user = generate_token(32);
          const id_user = results[0].id_user;
          console.log(token_user);
          connection.query(
            'UPDATE user SET token_user = ? WHERE id_user = ?',
            [token_user, id_user],
            (err, results, fields) => {
              if (err) {
                reject(err);
              } else {
                resolve(token_user);
              }
            }
          );
        }
      }
    );
  }).catch(error => {
    console.error('An error occurred:', error);
    if (error.status) {
      return Promise.reject(error);
    } else {
      return Promise.reject(new Error('Internal server error'));
    }
  });
 }
};
module.exports = Controller;
```

ПРИЛОЖЕНИЕ Б

Серверный компонент (app.js)

```
const http = require("http");
const Urok = require("./controller");
const { getReqData } = require("./utils");
const PORT = process.env.PORT || 5000;
const server = http.createServer(async (req, res) => {

    if (req.method === 'OPTIONS') {
        res.writeHead(
            200,
            {
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': '*',
                'Access-Control-Allow-Headers': '*'
            }
        )
        res.end();
    }else
    if (req.url === "/lessons" && req.method === "GET") {
      const body = await getReqData(req);
      //const token = JSON.parse(body);
      const token = req.headers["authorization"];
      const lessons = await new Urok().getUroks(token);
      res.writeHead(200, {
        "Content-Type": "application/json",
        "Access-Control-Allow-Origin": "*",
        "Access-Control-Allow-Methods": "*",
        "Access-Control-Allow-Headers": "*"
      });
      res.end(JSON.stringify(lessons));
    }
    if (req.url.match(/\/lessons\/([0-9]+)/) && req.method === "GET") {
        try {
            const id = req.url.split("/")[2];
            const urok = (await new Urok().getUroks("a")).find(iter =>
iter.id_urok == id);
            res.writeHead(200, {
                "Content-Type": "application/json",
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': '*',
                'Access-Control-Allow-Headers': '*'
            });
            res.end(JSON.stringify(urok));
        } catch (error) {
            res.writeHead(404, {
                "Content-Type": "application/json",
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Methods': '*',
```

```
                'Access-Control-Allow-Headers': '*'
            });
            res.end(JSON.stringify({ message: error }));
        }
    }
    else if (req.url === "/registration" && req.method === "POST") {
    let user_data = await getReqData(req);
    const user = JSON.parse(user_data);
    let newUser = await new Urok().createUser(user.id_user, user.FirstName,
user.LastName, user.login, user.password);
        res.writeHead(200, {
            "Content-Type": "application/json",
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': '*',
            'Access-Control-Allow-Headers': '*'
        });
    res.end(JSON.stringify({ id: newUser }));
    }
    else if (req.url === "/login" && req.method === "POST") {
        let user_data = await getReqData(req);
        const user = JSON.parse(user_data);
        let LoginUser = await new Urok().loginUser(user.login, user.password);
        res.writeHead(200, {
            "Content-Type": "application/json",
            'Access-Control-Allow-Origin': '*',
            'Access-Control-Allow-Methods': '*',
            'Access-Control-Allow-Headers': '*'
        });
        res.end(JSON.stringify({ token_user: LoginUser }));
    }
});
server.listen(PORT, () => {
    console.log(`Server started on port: ${PORT}`);
});
```

ПРИЛОЖЕНИЕ В

Серверный компонент (utils.js)

```
const crypto = require(`crypto`);
function getReqData(req) {
    return new Promise((resolve, reject) => {
        try {
            let body = "";
            req.on("data", (chunk) => {
                body += chunk.toString();
            });
            req.on("end", () => {
                resolve(body);
            });
        } catch (error) {
            reject(error);
        }
    });
}
function hashPassword(password) {
    return crypto.createHash(`sha256`)
        .update(password)
        .digest(`hex`)
}
module.exports = { getReqData, hashPassword };
```

ПРИЛОЖЕНИЕ Г

Роутер клиентской части

```
<template>
 <div>
  <SignIn @enter="step = 'Mode-Prescription'" v-if="step === 'Mode-SignIn'" />
  <SignUp @login="step = 'Mode-SignIn'" v-if="step === 'Mode-SignUp'" />
  <Prescription @model="step= 'Mode-Model'" v-if="step === 'Mode-Prescription'"
@view-lesson="showLesson" />
  <Lesson @main="step = 'Mode-Prescription'" v-if="step == 'Mode-View-Lesson'"
:lesson="lesson" />
  <Model @main="step = 'Mode-Prescription'" v-if="step === 'Mode-Model'" />
  </div>
</template>
<script>
import SignUp from "./signup.vue";
import SignIn from "./signin.vue";
import Prescription from "./prescription.vue";
import Lesson from './lesson.vue';
import Model from './model.vue';
export default {
  components: { SignIn, SignUp, Prescription, Lesson, Model },
  data() {
    return {
      step: "Mode-SignUp",
      lesson: 0
    }; },
  beforeMount() {
        const parts = location.pathname.split("/");
        if (parts[1] == "lesson") {
            this.lesson = parts[2];
            this.step = "Mode-View-Lesson";
        } else {
            this.step = "Mode-SignUp";} },
  methods: {
    handleEnter() {
      this.step = "Mode-Prescription";
    }
  },
    showLesson(id) {
    this.lesson = id;
    this.step = "Mode-View-Lesson";
  },
  goToModel(){
    this.step = "Mode-Model";
  }
};
</script>
<style src="./styles.css" scoped></style>
```

ПРИЛОЖЕНИЕ Д

Компонент регистрации

```
<template>
<div class="gradient">
    <div class="header">
        <img style="position:absolute;left:10px;top:10px;"
src="https://psv4.vkuseraudio.net/s/v1/d/ZOjISoBZgYhf9zOcQaGBsq38YMZwUAFBu2z2BQ14
7CgUP2y5ynVO_Bdt19mSUYq1aqmPEsH2f-x5vchoWw0FU4yIPNyOXu9PUv8-
UsQdz8dFf0R3OWS3eg/logo.png">
    </div>
    <div class="re">
        <h1>Регистрация</h1>
    </div>
<form class="form">
    <div class="col-sm-3 mx-auto mt-5">
        <div class="mb-3">
            <input v-model="FirstName" type="FirstName" class="form-control"
id="FirstName" placeholder="Имя" aria-describedby="FirstName">
        </div>

        <div class="mb-3">
            <input v-model="LastName" type="LastName" class="form-control"
id="LastName" placeholder="Фамилия" aria-describedby="LastNmae">
        </div>

        <div class="mb-3">
            <input v-model="login" type="login" class="form-control" id="login"
placeholder="Логин" aria-describedby="login">
        </div>

        <div class="mb-3">
            <input v-model="password" type="password" class="form-control"
placeholder="Пароль" id="password">
        </div>

        <button @click="SignUp" type="button" class="btn btn-primary"
style="position: relative; left: 50%;transform: translate(-50%,
0);">Зарегистрироваться</button>
        <p @click="$emit('login')" class="about">Есть аккаунт? Войдите!</p>

    </div>
</form>
</div>
</template>

<script>
    export default {
        data() {
```

```
            return {
                FirstName: '',
                LastName: '',
                login: '',
                password: '' } },
        emits: ['login'],
        methods: {
            SignUp() {
                const res = fetch('http://localhost:5000/registration', {
                    method: 'Post',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({
                        id_user: this.id_user,
                        FirstName: this.FirstName,
                        LastName: this.LastName,
                        login: this.login,
                        password: this.password

                    })
                })
                res.then().catch()
                console.log(res)

            }
        }
    }
</script>
<style src="./styles.css" scoped></style>
```

ПРИЛОЖЕНИЕ Е

Компонент авторизации

```
<template>
  <div class="gradient">
    <div class="header">
      <img style="position:absolute;left:10px;top:10px;"
src="https://psv4.vkuseraudio.net/s/v1/d/ZOjISoBZgYhf9zOcQaGBsq38YMZwUAFBu2z2BQ14
7CgUP2y5ynVO_Bdt19mSUYq1aqmPEsH2f-x5vchoWw0FU4yIPNyOXu9PUv8-
UsQdz8dFf0R3OWS3eg/logo.png">
    </div>
    <div>
      <h1 class="in">Вход</h1>
    </div>
    <form class="formin">
      <div class="col-sm-3 mx-auto mt-5">
        <div class="mb-3">
          <input v-model="login" type="login" class="form-control" id="login"
placeholder="Логин" aria-describedby="login">
        </div>
        <div class="mb-3">
          <input v-model="password" type="password" class="form-control"
placeholder="Пароль" id="password">
        </div>
        <button @click="signIn" type="button" class="btn btn-
primary">Войти</button>
        <div v-if="error" class="alert alert-danger mt-3">
          {{ error }}
        </div>
      </div>
    </form>
  </div>
</template>
<script>
export default {
  data() {
    return {
      login: "",
      password: "",
      error: ""
    };
  },
  methods: {
    signIn() {
      const currentObject = this;
      fetch("http://localhost:5000/login", {
        method: "POST",
        headers: {
          "Content-Type": "application/json"
```

```
        },
        body: JSON.stringify({
          login: this.login,
          password: this.password
        })
      })
        .then(res => res.json())
        .then(function(res) {
          if (res.error) {
            currentObject.error = res.error;
          } else {
            localStorage.setItem("token", res.token_user);
            currentObject.$emit("enter");
          }
        })
        .catch(function(error) {
          console.error("An error occurred:", error);
          currentObject.error = "Ошибка авторизации";
        });
    }
  }
};
</script>
<style src="./styles.css" scoped></style>
```

ПРИЛОЖЕНИЕ Ж

Компонент главной страницы

```
<template>
  <div class="gradient">
    <div class="header">
      <img style="position:absolute;left:10px;top:10px;"
src="https://psv4.vkuseraudio.net/s/v1/d/ZOjISoBZgYhf9zOcQaGBsq38YMZwUAFBu2z2BQ14
7CgUP2y5ynVO_Bdt19mSUYq1aqmPEsH2f-x5vchoWw0FU4yIPNyOXu9PUv8-
UsQdz8dFf0R3OWS3eg/logo.png">
      <div class="pop">
        <img class="popp" src="https://sun9-37.userapi.com/impg/CV6JO9-
Lz7Ne74SVWBiwpiyCFVU4NbEJjTx9cA/_C7pjA1SvWY.jpg?size=47x51&quality=96&sign=f3a712
8fd66eb2deabc7b3ddb1886a39&type=album">
        <div class="popf">Популярные уроки</div>
      </div>
      <div class="all">
        <img class="allp" src="https://sun9-
64.userapi.com/impg/4l42qxR4_YYC9znDYQBUbnhpenTQfpTeyMX6aA/rOgEu_e589c.jpg?size=4
7x51&quality=96&sign=b422c1d50d78e85aee7a10b1dd4965ea&type=album">
        <div class="allf">Все уроки</div>
      </div>
      <div class="fav">
        <img class="favp" src="https://sun9-
80.userapi.com/impg/N7Ive0haac7j7Zp9S6c1FlLMf008IKlc4w8eWA/2oBp_cxHPkQ.jpg?size=4
7x51&quality=96&sign=00931bda687dab269c362c45421b83b8&type=album">
        <p class="favf" @click="$emit('model')">Моделирование процессов</p>
      </div>
    </div>
    <div class="container">
      <a v-for="urok in uroks" :key="urok.id" class="block" :href="'/lesson/' +
urok.id_urok">
        <h1>{{ urok.name }}</h1>
        <a>{{ urok.text }}</a>
        </a>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      uroks: []
    };
  },
  created() {
    const token = localStorage.getItem('token');
    if (!token) {
```

```
      this.$router.push('/signin');
    } else {
      this.getUroks(token);
      setInterval(() => {
        this.getUroks(token);
      }, 5000);
    }
  },
  emits: ['go', 'main'],
  methods: {
    getUroks(token) {
      fetch('http://localhost:5000/lessons', {
        headers: {
          "Authorization": token
        }
      })
        .then(response => response.json())
        .then(data => {
          this.uroks = data.map(urok => ({
            ...urok,
            id: urok._id
          }));
        })
        .catch(error => {
          console.error('Error fetching uroks:', error);
        });
    },
    logout() {
      const token = localStorage.getItem('token');
      fetch(`http://localhost:5000/logout?token_user=${token}`, {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json'
        }
      })
        .then(response => response.text())
        .then(res => {
          alert(res);
          localStorage.removeItem('token');
          this.$router.push('/signin');
        })
        .catch(error => {
          console.error('Error logging out:', error);
        });
    },
    viewLesson(id) {
    this.$emit("view-lesson", id);
    },
  } };
</script>
<style src="./styles.css" scoped></style>
```

ПРИЛОЖЕНИЕ 3

Компонент страницы урока

```
<template>
<div class="gradient">
    <div class="header">
      <a href="/">
      <img style="position:absolute;left:10px;top:10px;"
src="https://psv4.vkuseraudio.net/s/v1/d/ZOjISoBZgYhf9zOcQaGBsq38YMZwUAFBu2z2BQ14
7CgUP2y5ynVO_Bdt19mSUYq1aqmPEsH2f-x5vchoWw0FU4yIPNyOXu9PUv8-
UsQdz8dFf0R3OWS3eg/logo.png">
    </a>
      <div class="pop">
        <img class="popp" src="https://sun9-37.userapi.com/impg/CV6JO9-
Lz7Ne74SVWBiwpiyCFVU4NbEJjTx9cA/_C7pjA1SvWY.jpg?size=47x51&quality=96&sign=f3a712
8fd66eb2deabc7b3ddb1886a39&type=album">
        <div class="popf">Популярные уроки</div>
      </div>
      <div class="all">
        <img class="allp" src="https://sun9-
64.userapi.com/impg/4l42qxR4_YYC9znDYQBUbnhpenTQfpTeyMX6aA/rOgEu_e589c.jpg?size=4
7x51&quality=96&sign=b422c1d50d78e85aee7a10b1dd4965ea&type=album">
        <div class="allf">Все уроки</div>
      </div>
      <div class="fav">
        <img class="favp" src="https://sun9-
80.userapi.com/impg/N7Ive0haac7j7Zp9S6c1FlLMf008IKlc4w8eWA/2oBp_cxHPkQ.jpg?size=4
7x51&quality=96&sign=00931bda687dab269c362c45421b83b8&type=album">
        <div class="favf" href="/">Моделирование процессов</div>
      </div>
    </div>
    <div class="lesson">
      <div v-if="lessonData" class="lesson_content">
        <h1>{{ lessonData.name }}</h1>
        <a>{{ lessonData.text }}</a>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  props: {
    lesson: { type: String }
  },
  data() {
    return {
      lessonData: null
    };
```

```
  },
  mounted() {
    this.loadLesson();
  },
  methods: {
    loadLesson() {
      fetch(`http://localhost:5000/lessons/${this.lesson}`, {
        headers: {
          Authorization: "a"
        }
      })
        .then(response => response.json())
        .then(data => {
          console.log(data);
          this.lessonData = data;
        })
        .catch(error => {
          console.error("Error fetching uroks:", error);
        });
    }
  }
}
</script>
<style src="./styles.css" scoped></style>
```

ПРИЛОЖЕНИЕ И

Компонент отдельной страницы урока

```
<template>
<div class="gradient">
    <div class="header">
      <a href="/">
      <img style="position:absolute;left:10px;top:10px;"
src="https://psv4.vkuseraudio.net/s/v1/d/ZOjISoBZgYhf9zOcQaGBsq38YMZwUAFBu2z2BQ14
7CgUP2y5ynVO_Bdt19mSUYq1aqmPEsH2f-x5vchoWw0FU4yIPNyOXu9PUv8-
UsQdz8dFf0R3OWS3eg/logo.png">
    </a>
      <div class="pop">
        <img class="popp" src="https://sun9-37.userapi.com/impg/CV6JO9-
Lz7Ne74SVWBiwpiyCFVU4NbEJjTx9cA/_C7pjA1SvWY.jpg?size=47x51&quality=96&sign=f3a712
8fd66eb2deabc7b3ddb1886a39&type=album">
        <div class="popf">Популярные уроки</div>
      </div>
      <div class="all">
        <img class="allp" src="https://sun9-
64.userapi.com/impg/4l42qxR4_YYC9znDYQBUbnhpenTQfpTeyMX6aA/rOgEu_e589c.jpg?size=4
7x51&quality=96&sign=b422c1d50d78e85aee7a10b1dd4965ea&type=album">
        <div class="allf">Все уроки</div>
      </div>
      <div class="fav">
        <img class="favp" src="https://sun9-
80.userapi.com/impg/N7Ive0haac7j7Zp9S6c1FlLMf008IKlc4w8eWA/2oBp_cxHPkQ.jpg?size=4
7x51&quality=96&sign=00931bda687dab269c362c45421b83b8&type=album">
        <div class="favf" href="/">Моделирование процессов</div>
      </div>
    </div>
    <div class="lesson">
      <div v-if="lessonData" class="lesson_content">
        <h1>{{ lessonData.name }}</h1>
        <a>{{ lessonData.text }}</a>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  props: {
    lesson: { type: String }
  },
  data() {
    return {
      lessonData: null
    };
```

```
  },
  mounted() {
    this.loadLesson();
  },
  methods: {
    loadLesson() {
      fetch(`http://localhost:5000/lessons/${this.lesson}`, {
        headers: {
          Authorization: "a"
        }
      })
        .then(response => response.json())
        .then(data => {
          console.log(data);
          this.lessonData = data;
        })
        .catch(error => {
          console.error("Error fetching uroks:", error);
        });
    }
  }
}
</script>
<style src="./styles.css" scoped></style>
```

ПРИЛОЖЕНИЕ К

Компонент страницы на которой происходит моделирование

```
<template>
  <div class="gradient">
    <div class="header">
      <a @click="$emit('main')" class="main">
        <img style="position:absolute;left:10px;top:10px;"
src="https://psv4.vkuseraudio.net/s/v1/d/ZOjISoBZgYhf9zOcQaGBsq38YMZwUAFBu2z2BQ14
7CgUP2y5ynVO_Bdt19mSUYq1aqmPEsH2f-x5vchoWw0FU4yIPNyOXu9PUv8-
UsQdz8dFf0R3OWS3eg/logo.png">
      </a>
      <div class="pop">
        <img class="popp" src="https://sun9-37.userapi.com/impg/CV6JO9-
Lz7Ne74SVWBiwpiyCFVU4NbEJjTx9cA/_C7pjA1SvWY.jpg?size=47x51&quality=96&sign=f3a712
8fd66eb2deabc7b3ddb1886a39&type=album">
        <div class="popf">Популярные уроки</div>
      </div>
      <div class="all">
        <img class="allp" src="https://sun9-
64.userapi.com/impg/4l42qxR4_YYC9znDYQBUbnhpenTQfpTeyMX6aA/rOgEu_e589c.jpg?size=4
7x51&quality=96&sign=b422c1d50d78e85aee7a10b1dd4965ea&type=album">
        <div class="allf" @click="$emit('main')">Все уроки</div>
      </div>
      <div class="fav">
        <img class="favp" src="https://sun9-
80.userapi.com/impg/N7Ive0haac7j7Zp9S6c1FlLMf008IKlc4w8eWA/2oBp_cxHPkQ.jpg?size=4
7x51&quality=96&sign=00931bda687dab269c362c45421b83b8&type=album">
        <div class="favf">Моделирование процессов</div>
      </div>
    </div>
    <div class="teploo">
      <div class="teplo">
        <h1>Замедленное моделирование теплового движения молекул</h1>

        <div class="sliders">
          <div class="slider">
            <label>Давление(Па): {{ pressure }}</label>
            <input type="range" v-model="pressure" min="0" max="100" />
          </div>

          <div class="slider">
            <label>Температура(К): {{ temperature }}</label>
            <input type="range" v-model="temperature" min="0" max="100" />
          </div>

          <div class="slider">
            <label>Объем(м³): {{ volume }}</label>
            <input type="range" v-model="volume" min="0" max="100" />
```

```
          </div>
        </div>
        <div class="simulation">
          <canvas ref="canvas"></canvas>
        </div>
      </div>
    </div>
  </div>
</template>
<script>
export default {
  data() {
    return {
      pressure: 50,
      temperature: 50,
      volume: 50,
      particles: [],
      canvasContext: null,
      animationFrameId: null,
    };
  },
  mounted() {
    this.canvasContext = this.$refs.canvas.getContext("2d");
    this.updateCanvasSize();
    this.startAnimation();
  },
  watch: {
    pressure() {
      this.initializeParticles();
    },
    temperature() {
      this.initializeParticles();
    },
    volume() {
      this.updateCanvasSize();
      this.initializeParticles();
    },
  },
  methods: {
    initializeParticles() {
      this.particles = [];

      const numParticles = Math.floor((this.pressure / 100) * 1000);
      const particleRadius = 2;

      for (let i = 0; i < numParticles; i++) {
        const x =
          Math.random() * (this.$refs.canvas.width - particleRadius * 2) +
          particleRadius;
        const y =
          Math.random() * (this.$refs.canvas.height - particleRadius * 2) +
```

```
        particleRadius;
      const dx = (Math.random() - 0.5) * (this.temperature / 25);
      const dy = (Math.random() - 0.5) * (this.temperature / 25);
      const color = "rgba(255, 255, 255, 0.5)";

      this.particles.push({ x, y, dx, dy, radius: particleRadius, color });
    }
  },
  startAnimation() {
    this.animate();
  },
  stopAnimation() {
    cancelAnimationFrame(this.animationFrameId);
  },
  animate() {
    this.canvasContext.clearRect(
      0,
      0,
      this.$refs.canvas.width,
      this.$refs.canvas.height
    );

    this.particles.forEach((particle) => {
      particle.x += particle.dx;
      particle.y += particle.dy;
      if (
        particle.x + particle.radius > this.$refs.canvas.width ||
        particle.x - particle.radius < 0
      ) {
        particle.dx = -particle.dx;
      }
      if (
        particle.y + particle.radius > this.$refs.canvas.height ||
        particle.y - particle.radius < 0
      ) {
        particle.dy = -particle.dy;
      }
      this.canvasContext.beginPath();
      this.canvasContext.arc(
        particle.x,
        particle.y,
        particle.radius,
        0,
        Math.PI * 2,
        false
      );
      this.canvasContext.fillStyle = particle.color;
      this.canvasContext.fill();
      this.canvasContext.closePath();
    });
```

```
      this.animationFrameId = requestAnimationFrame(this.animate);
    },
    updateCanvasSize() {
      const canvas = this.$refs.canvas;
      canvas.width = (this.volume / 100) * 400;
      canvas.height = (this.volume / 100) * 400;
    },
  },
};
</script>

  <style src="./styles.css" scoped></style>
```

ПРИЛОЖЕНИЕ Л

Main.js

```
import { createApp } from 'vue'
import App from './App.vue';
createApp(App).mount('#app')
```

ПРИЛОЖЕНИЕ М

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css"
integrity="sha384-
zCbKRCUGaJDkqS1kPbPd7TveP5iyJE0EjAuZQTgFLD2ylzuqKfdKlfG/eSrtxUkn"
crossorigin="anonymous">
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>LiØ</title>
</head>
  <body>
    <div id="app"></div>
    <script type="module" src="/src/main.js"></script>
  </body>
</html>
```

ПРИЛОЖЕНИЕ Н

Стили сайта

```
.form {
    position:absolute;
  top:25%;
  bottom:0;
  left:0;
  right:0;
  margin:auto;
}
    .about {
        text-decoration: none;
        color: #ffffff;
        padding: 20px;
        font: 20px/40px Roboto, sans-serif;
        position: relative;
        transition: all .2s ease;
        left:50%;
        transform: translate(-30%, 0);
    }
        .about:after {
            content: '';
            position: absolute;
            border: 10px solid darkorange;
            border-top-color: transparent;
            border-left-color: transparent;
            border-right-color: transparent;
            left: calc(50% - 5px);
            transition: all .2s ease;
            bottom: -5px;
            opacity: 0;
        }
        .about:hover {
            color: #57dfeb;
            text-shadow: 0 0 1px darkorange;
        }

        body {
  margin: 0;
  padding: 0;
}
.gradient {
    /* position: sticky; */
    height: 100vh;
    width: 100%;
    background: linear-gradient(149deg, rgba(24,187,156,1) 0%, rgba(106,57,175,1)
42%, rgba(187,24,148,1) 72%, rgba(115,53,134,1) 100%);
    animation: gradient 10s infinite linear;
```

```
    background-size: 400%;
}
@keyframes gradient {
    0% {
        background-position: 80% 0%;
    }
    50% {
        background-position: 20% 100%;
    }
    100% {
        background-position: 80% 0%;
    }
}
    .form-control {
        background: transparent;
        border-color: rgb(0, 0, 0);


    }
    .about {
        cursor: pointer;
    }
    .table {
        position: absolute;
        top: 12%;
        margin: auto;
    }
    .logout {
        position: absolute;
        top: 20%;
        margin: auto;
    }
    .pop{
        box-sizing: border-box;
        position: absolute;
        width: 370px;
        height: 50px;
        left: 367px;
        top: 35px;
        transition: 0.3s;
    }
    .popp {
        position: absolute;
        width: 30px;
        left: 7%;
        top: 10%;
     }
    .popf {
        background: #FFFFFF;
        border: 3px solid #C4C4C4;
        border-radius: 15px;
```

```
    text-align: center;
    font-size: 25px;
    font: Judson;
    font-weight: 700;
    cursor: pointer;
}
.pop:hover {
    transform: scale(1.1);
    transition: 0.3s;
}
.all{
    box-sizing: border-box;
    position: absolute;
    width: 340px;
    height: 50px;
    left: 367px;
    top: 35px;
    transition: 0.3s;
    left: 45%;
}
.allp {
    position: absolute;
    width: 30px;
    left: 7%;
    top: 10%;
 }
.allf {
    background: #FFFFFF;
    border: 3px solid #C4C4C4;
    border-radius: 15px;
    text-align: center;
    font-size: 25px;
    font: Judson;
    font-weight: 700;
    cursor: pointer;
}
.all:hover {
    transform: scale(1.1);
    transition: 0.3s;
}
.fav{
    box-sizing: border-box;
    position: absolute;
    width: 480px;
    height: 50px;
    left: 367px;
    top: 35px;
    transition: 0.3s;
    left: 70%;

}
```

```
.favp {
    position: absolute;
    width: 30px;
    left: 7%;
    top: 10%;
 }
.favf {
    background: #FFFFFF;
    border: 3px solid #C4C4C4;
    border-radius: 15px;
    text-align: center;
    font-size: 25px;
    font: Judson;
    font-weight: 700;
    cursor: pointer;
}
.fav:hover {
    transform: scale(1.1);
    transition: 0.3s;
}
.logout {
    top:50%;
}
.container {
    display: flex;
    flex-wrap: wrap;        justify-content: space-between;
    align-items: center;
    padding-top:20%;
}
  .block {
    width: 300px;
    height: 200px;
    background: #FFFFFF;
    border: 3px solid #C4C4C4;
    border-radius: 15px;
    text-align: center;
    font-size: 25px;
    font: Judson;
    font-weight: 700;
    cursor: pointer;
    transition: 0.3s;
    -webkit-line-clamp: 2;
    overflow: hidden;
    text-overflow: ellipsis;
    color:rgb(0, 0, 0);
    text-decoration: none;
  }
  .block:hover {
    transform: scale(1.1);
    transition: 0.3s;
}
```

```css
.lesson{
    display: flex;
    padding-top:10%;
    padding-left:10%;
}
.lesson_content{
    width: 90%;
    height: 90%;
    background: #FFFFFF;
    border: 3px solid #C4C4C4;
    border-radius: 15px;
    text-align: center;
    font-size: 25px;
    font: Judson;
    font-weight: 700;
    cursor: pointer;
    word-wrap: break-word;
    color:rgb(0, 0, 0);
    text-decoration: none;
}
.teplo {
    max-width: 600px;
    margin: 0 auto;
    }
    .sliders {
    margin-bottom: 20px;
    }
    .slider {
    margin-bottom: 10px;
    }
    .simulation {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 400px;
    }
    canvas {
    border: 1px solid #000;
    }
    .teploo {
        padding-top: 10%;

    }
    .main {
        cursor: pointer;
    }
    .in{
        font: Judson;
        padding-left: 46%;
        padding-top: 10%;
        font-size: 50px;
```

```
}
.re {
    font: Judson;
    padding-left: 44%;
    padding-top: 10%;
    font-size: 50px;
}
```