СОДЕРЖАНИЕ

Введение 4

1 Технический проект 6

1.1 Анализ предметной области 6

1.2 Постановка задачи 8

1.3 Проектирование функциональности веб–приложения 12

1.4 Проектирование веб–приложения 15

1.5 Проектирование структуры базы данных 20

1.6 Выбор средств разработки 22

1.7 Требования к программному обеспечению и техническим средствам 30

2 Рабочий проект 32

2.1 Логическая структура базы данных 33

2.2 Функциональное взаимодействие модулей веб–приложения 33

2.3 Файловая структура веб–приложения 37

2.4 Входные и выходные данные 39

2.5 Работа с веб–приложением в качестве администратора 40

2.6 Работа с веб–приложением в качестве разработчика 42

2.7 Работа с веб–приложением в качестве пользователя 43

Заключение 45

Список использованных источников 47

Приложение А (справочное) Use Case диаграмма проектируемого веб–приложения 49

Приложение Б (справочное) Листинг модуля index.js 50

ВВЕДЕНИЕ

В связи с быстрым развитием технологий потребность в непрерывном обучении становится все более актуальной, а дистанционное изучение образовательных курсов становится все более эффективным способом получения знаний.

Основной проблемой является отсутствие универсальных, гибких и масштабируемых платформ, которые бы позволяли организовать дистанционное изучение образовательных курсов. Существующие решения зачастую громоздки, имеют ограниченный функционал и сложны в использовании.

Темой дипломного проекта является «Разработка веб–приложения для организации дистанционного изучения образовательных курсов».

Разрабатываемое веб–приложение призвано упростить работу разработчикам курса и слушателям, предоставляя удобную и функциональную платформу для организации дистанционного изучения образовательных курсов.

Актуальность данной темы обуславливается тем, что в современном мире образование и технологии тесно связаны, предлагая уникальные способы распространения материала для обучения. Веб–приложение позволяет расширить круг обучающихся на онлайн–курсах веб–приложения и обеспечить круглосуточный доступ к размещенной на нем информации [1].

Целью проекта является создание веб–приложения, обеспечивающее организацию дистанционного изучения образовательных курсов, позволяющие пользователям учиться в любое время, разработчикам управлять образовательными курсами, а администраторам управлять пользователями.

В ходе работы необходимо будет рассмотреть и решить следующие задачи:

* выполнить анализ существующих решений для организации дистанционного обучения;
* выполнить анализ требований и функций разрабатываемого веб–приложения;
* разработать структуру курсов и дизайн интерфейса;
* разработать концептуальную, логическую и физическую модели базы данных;
* реализовать физическую модель базы данных;
* выбрать технологии для создания веб–приложения;
* разработать код веб–приложения;
* провести тестирование и оптимизацию веб–приложения;
* подготовить руководство пользователя и документацию по проекту.

Объектом исследования являются веб–приложения для дистанционного обучения.

Предметом исследования являются методы и средства разработки веб–приложений.

Разрабатываемое веб–приложение для дистанционного обучения призвано повысить доступность образования.

# ТЕХНИЧЕСКИЙ ПРОЕКТ

**1.1 Анализ предметной области**

Реализация веб–приложения для работы с дистанционным изучением образовательных курсов является актуальным и востребованным направлением в области образования. Для более глубокого понимания этой предметной области и определения основных аспектов, связанных с разработкой подобного приложения, необходимо провести анализ существующих решений.

Существует множество технологий и инструментов, предназначенных для дистанционного изучения. Важными компонентами являются учебные платформы, веб–конференции, электронные учебники, системы управления курсами и многие другие. Они позволяют организовать процесс обучения, предоставляя пользователям доступ к курсам.

Анализ конкурентов поможет определить недостатки существующих решений и спроектировать веб–приложение таким образом, чтобы оно имело преимущества, например, простой интерфейс для создания интерактивных курсов, расширенные возможности управления учебным процессом.

При анализе предметной области необходимо рассмотреть существующие решения на рынке дистанционного обучения. Рассмотрим три популярные платформы, входящие в топ–16 по данным Тинькофф Журнал – Stepik, Moodle и ПостНаука [19].

Stepik – это образовательная платформа и конструктор онлайн–курсов и уроков. Она позволяет создавать интерактивные образовательные курсы с автоматической проверкой заданий и обратной связью.

Плюсы Stepik:

* интеграция с системами видеоконференций для вебинаров;
* бесплатное использование для создания курсов.

Минусы Stepik:

* бесплатная версия для создания курса предлагает маленький функционал;
* ограниченный функционал для управления учебным процессом;
* сложный интерфейс для создания курсов.

Moodle – это система управления обучением (LMS), предоставляющая возможности для организации онлайн–курсов и дистанционного образования.

Плюсы Moodle:

* открытый исходный код и бесплатное использование;
* расширенная функциональность для управления учебным процессом;
* гибкие настройки для создания различных типов курсов;
* поддержка стандартов обмена учебными материалами.

Минусы Moodle:

* необходимость самостоятельного хостинга и настройки;
* ограниченные возможности для создания интерактивных курсов.

ПостНаука – это сайт о современной фундаментальной науке и ученых, которые ее создают. Он предоставляет возможности для изучения научных идей и достижений через видеоматериалы и тексты.

Плюсы ПостНаука:

* большой объем материалов – за прошедшее время было опубликовано более 3500 материалов, из них почти 2000 – видео о достижениях фундаментальной науки и важных современных технологиях;
* экспертность авторов – все авторы ПостНауки – эксперты в своей научной дисциплине;
* фокус на фундаментальную науку – темы освещают в первую очередь фундаментальную науку, перенося на нее акцент с прикладных областей.

Минусы ПостНаука:

* бесплатная версия для создания курса предлагает маленький функционал;
* ограниченный функционал для управления учебным процессом.

Проанализировав существующие решения, можно сделать вывод, что образовательная платформа должна сочетать в себе следующие элементы:

* гибкость и функциональность – платформа должна предлагать спектр инструментов создания курсов;
* простота использования – интерфейс должен быть интуитивно понятным, чтобы преподаватели могли без труда создавать курсы, а пользователи – изучать материалы образовательных курсов;
* бесплатное использование – доступ к образованию должен быть открытым для всех, поэтому важно, чтобы платформа предлагала бесплатные возможности для создания и изучения курсов.

При разработке веб–приложения для дистанционного изучения могут возникнуть различные ограничения и проблемы, которые необходимо учесть:

* скорость интернет–соединения;
* совместимость браузеров;
* защита личных данных;
* авторизация;
* регистрация;
* управление контентом.
  1. Постановка задачи

Целевой аудиторией разрабатываемого веб–приложения являются:

* пользователи, ищущие возможность изучения курсов на дому или вне учебных заведений;
* администраторы веб–приложения;
* разработчики образовательных курсов.

Необходимо разработать веб–приложение, предназначенное для организации дистанционного изучения образовательных курсов. Веб–приложение будет создаваться с учетом современных требований в области дистанционного изучения и информационных технологий, а также потребностей пользователей образовательных программ.

Большинство пользователей используют мобильные устройства для доступа к веб–приложениям. Это подчеркивает важность реализации адаптивного веб–приложения.

Адаптивный веб–приложение – это интернет–ресурс с дизайном, который подходит для просмотра с мобильного, планшета и компьютера с разными разрешениями экранов. Адаптивность веб–приложения – это способность «подстраиваться» под различные технические условия (а именно, под размеры экрана пользовательского устройства). Адаптивное веб–приложение хорошо смотрится и на десктопном мониторе (обычный ПК), и на планшетном компьютере, и на экране смартфона.

При разработке важно учесть требования к безопасности данных, производительности и масштабируемости приложения.

Функциональные требования для пользователей:

* регистрация и создание личного кабинета;
* поиск и фильтрация курсов;
* просмотр доступных курсов и их описаний (без контента);
* доступ к учебным материалам (для авторизованных пользователей).

Функциональные требования для разработчика:

* создание и редактирование образовательных курсов, разделов, материалов, реализованных разработчиком (то есть, каждый курс должен каждый разработчик должен иметь свой личный кабинет с его разработанными курсами);
* загрузка различного формата контента.

Функциональные требования для администратора:

* управление всеми пользователями веб–приложения (CRUD–операции);
* отображение всех курсов и CRUD.

Нефункциональные требования:

* производительность – быстрое время отклика, способность обрабатывать большое количество одновременных запросов;
* безопасность – защита персональных данных пользователей, аутентификация и авторизация, шифрование данных;
* масштабируемость – возможность наращивания вычислительных мощностей и расширения функциональности при росте нагрузки;
* доступность – круглосуточная доступность веб–приложения с минимальными периодами простоя;
* кроссплатформенность – совместимость с различными операционными системами и веб–браузерами;
* юзабилити – интуитивно понятный и удобный пользовательский интерфейс.

Для обеспечения функционала необходимо решить следующие задачи:

* 1. разработать интерфейс и структуру веб–приложения:
* создать пользовательский интерфейс, который будет интуитивно понятен и удобен для использования пользователям, разработчикам и администраторам;
* определить структуру курсов и учебных материалов.
  1. разработать функциональность веб–приложения:
* реализовать возможность для пользователей зарегистрироваться и войти в систему;
* обеспечить разработчикам и администраторам возможность создавать и управлять образовательными курсами (CRUD–операции);
* предоставить каждому пользователю личный кабинет, а также обеспечить доступ к учебным материалам.
  1. создать базу данных:
* разработать структуру базы данных, включая таблицы для хранения информации о пользователях, ролях, курсах, учебных материалах;
* обеспечить безопасное и надежное хранение данных пользователей и курсов.
  1. разработать систему администрирования:
* создать административный интерфейс для управления курсами;
* реализовать механизмы авторизации и регистрации.
  1. разработать адаптивный дизайн веб–приложения:
* учесть различные размеры экранов устройств (смартфоны, планшеты, компьютеры) при разработке дизайна интерфейса;
* обеспечить корректное отображение и функционирование веб–приложения на всех типах устройств;
* протестировать веб–приложение на различных устройствах и разрешениях экрана для обеспечения его адаптивности.
  1. реализовать использование S3 Object Storage:
* разработать функциональность для загрузки файлов курса на S3 Object Storage;
* обеспечить безопасность и конфиденциальность данных при передаче файлов в облачное хранилище данных;
* реализовать механизмы для управления файлами в облаке, включая загрузку, обновление, удаление и доступ к файлам;
* провести тестирование функциональности S3 Object Storage для обеспечения его корректной работы и безопасности.
  1. разработать документацию:
* подготовить техническую документацию, описывающую структуру и функциональность приложения;
* создать рабочие проекты, включая диаграммы классов, вариантов использования и другие документы, необходимые для понимания архитектуры приложения.
  1. провести тестирование:
* оценить корректность работы приложения путем проверки его функциональности и безопасности;
* провести блочное тестирование, включая тестирование интерфейса.

Для успешной реализации проекта необходимо выбрать соответствующие технологии разработки, обеспечивающие безопасность, масштабируемость и высокую производительность веб–приложения.

1.3 Проектирование функциональности веб–приложения

Для описания функциональных требований к веб–приложению воспользуемся методологией UML, а именно диаграммой Use Case. Данная диаграмма была реализована с помощью инструмента PlantUML. Она (диаграмма) представляет собой графическое представление взаимодействия пользователей с системой. На ней отображаются все возможные варианты использования системы, а также взаимодействие системы с внешними акторами.

Диаграмма Use Case помогает лучше понять, как пользователи будут взаимодействовать с нашим веб–приложением, и какие функциональные требования к нему предъявляются (Приложение А).

Алгоритм авторизации любого пользователя (пользователь, разработчик, администратор) веб–приложения:

* пользователь вводит свои учетные данные (логин / email и пароль) на странице авторизации;
* веб–приложение отправляет запрос на сервер для проверки введенных данных;
* сервер проверяет наличие пользователя в базе данных и сравнивает введенный пароль с хешированным значением из базы данных;
* если данные верны, сервер генерирует JWT–токен (Json Web Token) и возвращает его клиенту;
* клиент сохраняет JWT–токен;
* при последующих запросах к защищенным ресурсам веб–приложения, JWT–токен передается в заголовке запроса;
* сервер проверяет валидность JWT–токена и предоставляет доступ к запрашиваемым ресурсам в соответствии с правами пользователя.

Алгоритм регистрации пользователя:

* пользователь заполняет форму регистрации на клиентской стороне, вводя необходимые данные (имя пользователя, логин, пароль);
* клиентское приложение отправляет данные из формы регистрации на сервер;
* сервер проверяет введенные данные на корректность и соответствие требованиям (правильный формат электронной почты, достаточная длина пароля);
* сервер проверяет уникальность введенного логин пользователя по сравнению с существующими записями в базе данных;
* если введенные данные корректны и уникальны, сервер хеширует пароль с использованием безопасного алгоритма хеширования bcrypt;
* сервер создает новую запись в базе данных пользователей, сохраняя введенные данные и захешированный пароль;
* сервер генерирует JWT–токен (Json Web Token) для нового пользователя, содержащий информацию о пользователе и срок действия токена;
* сервер отправляет сгенерированный JWT–токен клиентскому приложению;
* клиентское приложение сохраняет полученный JWT–токен;
* при последующих запросах к защищенным ресурсам веб–приложения, клиентское приложение передает JWT–токен в заголовке запроса;
* сервер проверяет валидность JWT–токена и, при успешной проверке, предоставляет доступ к запрашиваемым ресурсам в соответствии с правами пользователя.

Алгоритм добавления нового курса разработчиком:

* разработчик заполняет форму на веб–странице, вводя название курса, его описание, название раздела, название урока, описание урока и выбирает файл урока.
* при нажатии на кнопку «Добавить урок», в форму динамически добавляется новый блок для ввода информации об уроке (название, описание и выбор файла).
* при нажатии на кнопку «Добавить раздел», в форму динамически добавляется новый блок для ввода информации о разделе (название раздела, название урока, описание урока и выбор файла).
* после заполнения всех необходимых полей формы, разработчик нажимает на кнопку «Опубликовать».
* клиентский код выполняет следующие действия:

1. собирает введенные данные из формы;
2. для каждого урока формирует объект, который включает название курса, описание курса, название раздела, название урока, описание урока и файл урока;
3. отправляет объекты на сервер по адресу с помощью Fetch API и метода POST;
4. сервер обрабатывает полученные данные, сохраняет файлы уроков и возвращает URL–адреса загруженных файлов;
5. клиентский код формирует массив объектов, представляющих разделы курса, где каждый раздел содержит массив уроков с названием, описанием и URL–адресом загруженного файла;
6. отправляет данные о курсе (название, описание, массив разделов) на сервер с помощью Fetch API и метода POST;
7. если курс успешно добавлен, клиентский код очищает форму и перенаправляет пользователя на страницу с реализованными им курсами.

* на сервере должна быть реализована обработка полученных данных, сохранение файлов уроков и добавление информации о новом курсе в базу данных.

1.4 Проектирование структуры веб–приложения

Веб–приложение – это сайт или программное обеспечение, которое позволяет пользователям совершать с контентом различные действия (получать, производить, размещать, распространять, хранить) посредством созданного пользователем аккаунта или публичного сообщества [16].

По функциональности выделяют следующие типы сайтов [17]:

* информационные сайты;
* коммерческие сайты;
* образовательные сайты;
* сайты–портфолио;
* форумы;
* вики–сайты;
* сайты–знакомств;
* сайты–развлечений;
* персональные сайты.

Информационные сайты – на них размещаются материалы информационного характера, ориентированные на читателя. Цель таких сайтов – раскрыть определённую тематику, проблематику или вопрос, донести информацию до аудитории и держать читателей в курсе новостей конкретной тематики или тематик.

Коммерческие сайты – созданы с целью продвижения товаров или услуг. Они обычно содержат посадочные страницы услуг или карточки товаров, контактные данные, форму обратной связи и оформления заказа. Классическим примером коммерческого сайта является интернет–магазин, где все элементы созданы для стимулирования посетителей к покупке.

Образовательные сайты – предназначены для обучения и самообразования. Они предоставляют доступ к различным образовательным материалам, таким как учебники, научные журналы, видео–лекции, интерактивные упражнения и даже мобильные игры.

Сайты–портфолио – позволяют пользователям создавать и публиковать свои портфолио, показывая различные проекты, работы и достижения. Портфолио – это собрание выполненных работ, проектов и других документов, которые демонстрируют ваш профессионализм и опыт. Они используются для поиска потенциальных работодателей и заказчиков, а также для повышения авторитета в индустрии.

Форумы – предназначены для обсуждения определенных тем. Они представляют собой место в интернете, где люди общаются на важные для них темы, советуются, обмениваются опытом, помогают друг другу. Форумы облегчают общение посредством цепочек обсуждений.

Вики–сайты – платформа, данные которого пользователи могут самостоятельно изменять с помощью инструментов, предоставляемых самим сайтом. Форматирование текста и вставка различных объектов в текст производятся с использованием вики–разметки.

Сайты знакомств – помогают людям находить и общаться с другими людьми с похожими интересами. Они обычно предлагают различные функции, такие как профили пользователей, чаты, системы совместимости.

Сайты развлечений – предназначены для предоставления различных видов развлекательного контента. Они могут включать в себя новости о знаменитостях, фильмах, музыке, играх, а также предлагать онлайн–игры, комиксы, видео, подкасты.

Персональные веб–сайты – созданы отдельными лицами, и обычно представляют собой коллекцию веб–страниц, которые содержат информацию об этом человеке. Они могут включать в себя блоги, портфолио, резюме, личные проекты, фотографии, видео и другую информацию, которую человек хочет поделиться с общественностью.

Исходя из рассмотренных выше типов сайтов по функциональности, разрабатываться будет образовательный сайт.

Проектирование и разработка веб–приложения включают в себя следующие этапы:

* определение структурной схемы веб–приложения – расположения контента, разделов и элементов навигации;
* разработка веб–дизайна – создание графических элементов макета веб–приложения, стилей и элементов навигации;
* разработка модулей, базы данных и других элементов;
* разработка взаимодействия модулей и базы данных;
* тестирование и размещение веб–приложения в сети Интернет.

Веб–приложение будет организовано в виде многостраничной структуры. Многостраничная структура сайта – это веб–сайт, состоящий из нескольких страниц, которые связаны между собой. Каждая страница может содержать различный контент, такой как текст, изображения, видео, формы обратной связи и другие элементы. Многостраничные сайты часто используются для представления большого объема информации, например: веб–магазины, блоги, новостные порталы и корпоративные сайты. Каждая страница может иметь свой уникальный URL и навигацию между страницами обычно осуществляется с помощью гиперссылок или меню.

Необходимо спроектировать структуру базы данных для хранения информации о пользователях, курсах, разделах, материалах и других сущностях. Также важно определить роли пользователей (администратор, преподаватель, студент) и их привилегии доступа.

Следует уделить внимание проектированию API для взаимодействия между клиентской и серверной частями приложения, определив необходимые конечные точки и форматы передачи данных.

Для повышения безопасности необходимо предусмотреть использование протокола HTTPS, а также механизмы аутентификации и авторизации пользователей с использованием JSON Web Tokens (JWT) или других современных технологий.

Кроме того, важно спроектировать масштабируемую архитектуру приложения, чтобы обеспечить его стабильную работу при росте числа пользователей и курсов.

Логическая структура веб–приложения – это совокупность всех страниц на сайте, расположенных с учётом иерархии. Это взаимосвязь страниц, в которых прослеживается их принадлежность к разделам, категориям, подкатегориям и другим типам страниц (карточкам товара, тегам, фильтрам и прочее).

Структура веб–приложения должна быть построена таким образом, чтобы упростить пользователям переход от общих тем к конкретной информации, за которой они пришли. От этого зависит удобство пользователя, время, которое он проведет на вашем сайте, целевые действия, которые он совершит.

Важно, чтобы структура веб–приложения была простой и понятной, так как это влияет на то, как быстро посетитель найдет нужную ему информацию. Запутанная система навигации и чрезмерное структурирование контента могут привести к тому, что пользователь не сможет найти нужную информацию.

Также существуют определённые требования поисковых систем к структуре веб–приложения, которым нужно следовать для оптимизации веб–приложения под поисковые системы.

Внутренняя структура веб–приложения – это система организации и взаимосвязи веб–страниц на сайте. Она включает в себя следующие аспекты:

* порядок формирования URL. Это порядок, согласно которому страницы ссылаются друг на друга и на главную;
* четкая ссылочная схема и карта веб–приложения. Это важно для поисковых систем, таких как Яндекс;
* уникальные адреса страниц. Это еще одно требование поисковых систем;
* типы внутренних структур веб–приложения. Включают линейную, линейную с ответвлениями, блочную и древовидную структуры.

Внутренняя структура веб–приложения – это упорядоченное размещение страниц веб–приложения относительно главной страницы, она играет важную роль в обеспечении удобства навигации для пользователей и оптимизации веб–приложения для поисковых систем. Внутренняя структура разрабатываемого веб–приложения представлена на рисунке 1.

Благодаря навигационной панели на главной странице пользователь сможет легко передвигаться по всем разделам веб–приложения.

Схема функционального взаимодействия модулей с внутренней структурой, разрабатываемого веб–приложения представлена на рисунке 2.

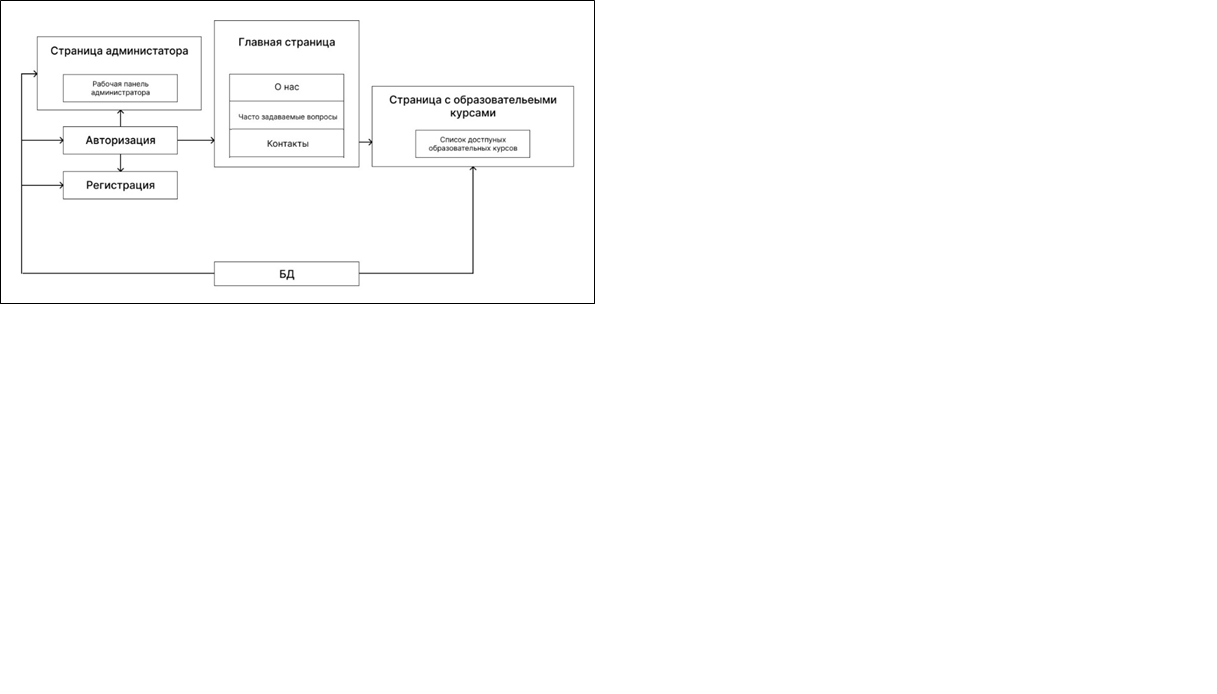
****

Рисунок 1 – Внутренняя структура веб–приложения

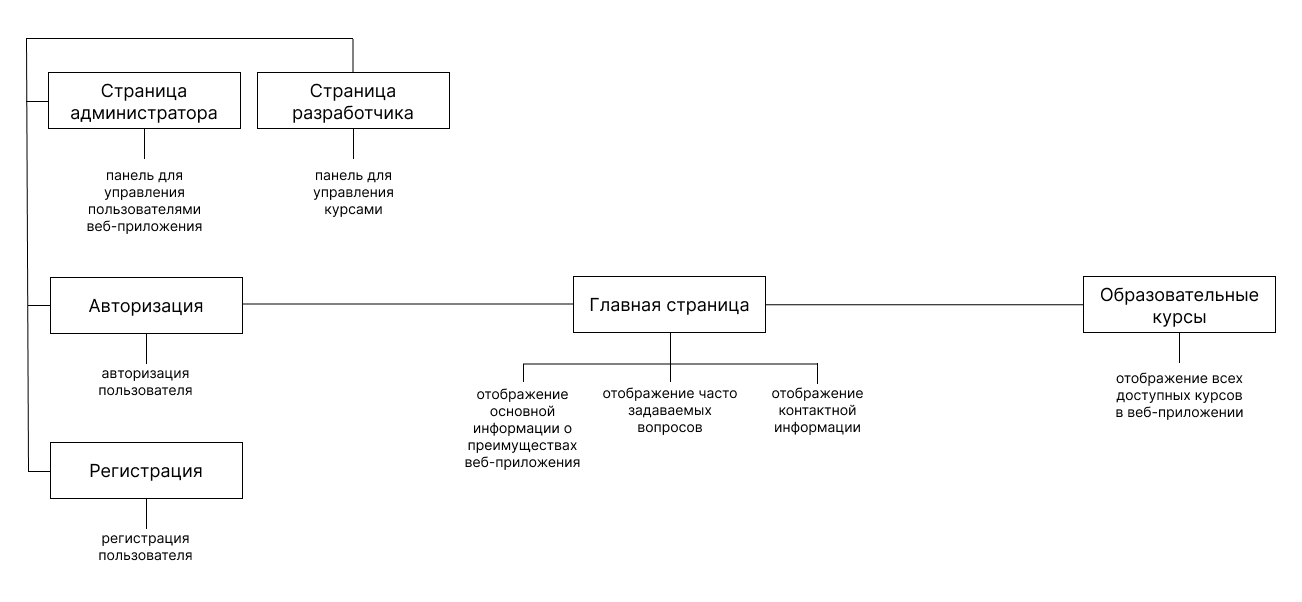


Рисунок 2 – Функциональные модули веб–приложения

1.5 Проектирование структуры базы данных

Для реализации полнофункционального веб–приложения необходимо использование базы данных [2].

В ходе исследования предметной области был выявлен набор коллекций базы данных (БД), которая будет состоять из следующих коллекций:

* пользователь;
* роль пользователя;
* образовательный курс;
* категория образовательного курса.

Была разработана концептуальная модуль базы данных, и логическая модель, которая отражает связи между коллекциями.

Концептуальная и логическая модели базы данных представлены на рисунке 4 и рисунке 5 соответственно.

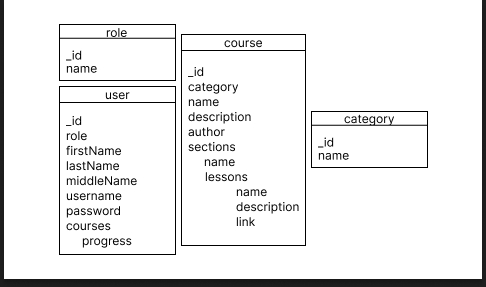


Рисунок 4 – Концептуальная модель базы данных

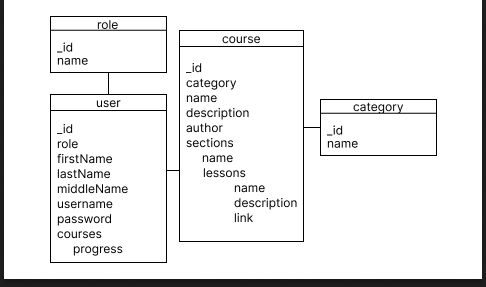


Рисунок 5 – Логическая модель базы данных

При взаимодействии с базой данных будет использоваться технология клиент–сервер. В основе этой концепции лежит идея о том, что помимо хранения файлов базы данных, центральный процессор должен выполнять основную часть обработки данных. В отличие от других технологий, технология клиент–сервер после обработки запроса выдает пользователю только ту информацию, которая ему необходима, а не весь набор данных.

Таким образом, предотвращается перегрузка сети серверами.

Кроме того, можно выделить следующие преимущества клиент–серверной системы:

* система предоставляет сильную централизованную защиту;
* файлы хранятся в централизованном месте, что облегчает управление данными;
* серверы могут совместно использовать доступное техническое и программное обеспечение;
* система обеспечивает простую управляемость при большом числе пользователей;
* система предлагает дружественный интерфейс для удобства работы пользователей;
* программные продукты легко модифицировать и настраивать;
* система обеспечивает высокую производительность;
* система может масштабироваться в соответствии с потребностями.

1.6 Выбор средств разработки

Для разработки веб–приложения, предназначенного для организации дистанционного изучения образовательных курсов, могут быть выбраны следующие средства разработки, учитывая их актуальность, распространенность и возможность эффективной реализации задачи:

HTML – это стандартный язык разметки, используемый для создания и структурирования веб–страниц. HTML предоставляет средства для определения структуры, содержания и элементов на веб–страницах, и определяет, как эти элементы будут отображаться в веб–браузере.

CSS – это язык стилей, который используется для определения внешнего вида и стиля веб–страниц и веб–приложений. CSS позволяет контролировать, как элементы HTML размечаются и отображаются на экране, включая такие аспекты, как цвет текста, размер шрифта, расположение элементов, отступы, границы, фоны и анимации [9].

Для создания веб–приложения наиболее подходящим языком программирования является JavaScript. JavaScript (часто сокращенно JS) – это высокоуровневый, интерпретируемый язык программирования, который применяется в веб–разработке для создания интерактивных веб–страниц и веб–приложений. Он является одним из ключевых компонентов веб–технологий и выполняется в браузерах, что позволяет создавать динамические и интерактивные элементы на веб–страницах.

В качестве IDE для разработки веб–приложения можно использовать Visual Studio Code.

Visual Studio Code (VS Code) – это бесплатная интегрированная среда разработки (IDE), разработанная и поддерживаемая компанией Microsoft. Она предназначена для создания и отладки программного обеспечения, а также обладает обширной расширений, что делает ее мощным и гибким инструментом для разработчиков.

Для хранения данных приложения, включая информацию о пользователях, курсах, учебных материалах будет использоваться MongoDB.

MongoDB – это документоориентированная NoSQL база данных. Он разработан, чтобы быть простым, гибким и масштабируемым, с акцентом на производительность и доступность. MongoDB поставляется с открытым исходным кодом и предлагает как серверную, так и базу данных с документами.

MongoDB использует JSON–подобный документ в качестве основной единицы хранения, и эти документы хранятся в коллекции. Каждая коллекция может содержать нулевые или несколько документов, и каждый документ может иметь разные поля и типы данных.

PostgreSQL – это свободная объектно–реляционная система управления базами данных (СУБД). Она существует во множестве реализаций для UNIX–подобных и других операционных систем. PostgreSQL была создана на основе некоммерческой СУБД Postgres, разработанной в Калифорнийском университете в Беркли.

MySQL – это свободная реляционная система управления базами данных. Она была разработана и поддерживается корпорацией Oracle, которая получила права на торговую марку после поглощения Sun Microsystems. MySQL часто используется в малых и средних приложениях, таких как WAMP, AppServ, LAMP и портативных сборках серверов, например, Denwer, XAMPP и VertrigoServ.

MySQL предлагает различные типы таблиц, включая MyISAM для быстрого чтения и InnoDB для транзакций и ссылочной целостности. Также в MySQL есть поддержка транзакций, SSL, кэширования запросов, репликации, полнотекстовой индексации и поиска, а также встроенная библиотека базы данных и поддержка Юникода (UTF–8).

MS SQL (Microsoft SQL Server) – это реляционная система управления базами данных, разработанная корпорацией Microsoft. Она используется для работы с данными веб–приложений и поддерживает свой диалект SQL, называемый Transact–SQL.

Node.js – является средой выполнения JavaScript, которая позволяет разработчикам использовать JavaScript для создания серверного программного обеспечения. Node.js отличается от других подобных решений тем, что он использует асинхронный, многопоточный API, который позволяет обрабатывать данные без блокировки ресурсов. Это делает Node.js идеальным выбором для высокопроизводительных, масштабируемых приложений.

Node.js имеет большое сообщество и множество доступных модулей и пакетов, которые позволяют разработчикам легко создавать веб–приложения. Выбор указанных средств разработки обеспечит создание веб–приложения для организации дистанционного изучения образовательных курсов.

React – это библиотека JavaScript для создания пользовательских интерфейсов, разработанная Facebook. React использует концепцию компонентов, которые позволяют разделить код на повторно используемые части. Он также использует виртуальный DOM для оптимизации производительности. React широко используется для создания одностраничных приложений (SPA) и имеет большое сообщество разработчиков.

Angular – это полнофункциональный фреймворк JavaScript, разработанный Google. Он основан на TypeScript и предлагает всеобъемлющий набор инструментов и библиотек для создания веб–приложений. Angular имеет модульную структуру и поддерживает такие концепции, как зависимости, инъекции, директивы и многое другое. Он популярен для создания корпоративных веб–приложений.

Tailwind CSS – это утилитарная CSS–библиотека, которая предоставляет низкоуровневые классы для создания стилей. Вместо традиционного подхода с предопределенными компонентами, Tailwind CSS позволяет разработчикам создавать свои собственные компоненты с помощью комбинации маленьких классов. Это обеспечивает гибкость и контроль над стилями.

NestJS – это фреймворк для Node.js, который основан на архитектурных принципах Angular. Он использует TypeScript и опирается на концепции вроде модулей, контроллеров, провайдеров и других. NestJS упрощает создание масштабируемых серверных приложений, предоставляя четкую архитектуру и множество встроенных функций.

Vue.js 3 – это прогрессивный фреймворк JavaScript для создания пользовательских интерфейсов. Он использует модель реактивного программирования для эффективного обновления DOM при изменении данных. Vue.js 3 является последней версией фреймворка и предлагает улучшения производительности, совместимость с TypeScript и новые функции, такие как Composition API.

Vuex – это библиотека управления состоянием для Vue.js. Она следует архитектурному паттерну Flux и помогает разработчикам управлять состоянием приложения в централизованном хранилище. Vuex упрощает обработку изменений состояния и обеспечивает предсказуемый поток данных в приложении Vue.js.

NuxtJS 3 – это современный фреймворк для создания веб–приложений на Vue.js. Он предлагает множество улучшений и новых возможностей по сравнению с предыдущими версиями.

Nuxt UI – это библиотека компонентов Vue.js, разработанная специально для использования с Nuxt 3. Она предоставляет набор готовых компонентов пользовательского интерфейса, таких как кнопки, модальные окна, уведомления и многое другое. Nuxt UI тесно интегрирована с Nuxt 3 и следует его принципам и рекомендациям по разработке.

Для хранения всех материалов курсов рекомендуется использовать S3 Object Storage. Оно предоставляет мощное и гибкое облачное хранилище. S3 (Simple Storage Service) позволяет хранить и управлять большими объемами данных в виде объектов. Таких как файлы любых форматов: текстовые документы, изображения, видео– и аудиофайлы, презентации и другие учебные материалы.

S3 обеспечивает высокий уровень безопасности данных. Большинство облачных хранилищ (Yandex Cloud, Amazon Web Service, Google Cloud) предоставляют различные инструменты и механизмы для защиты данных, такие как управление доступом, шифрование данных в покое и в движении, мониторинг доступа, что позволяет обеспечить конфиденциальность и целостность учебных материалов.

Yandex Cloud – это публичная облачная платформа от транснациональной интернет–компании «Яндекс». Многие компоненты Yandex.Cloud основаны на внутренних продуктах компании, а также используют ПО с открытым исходным кодом. В состав Yandex.Cloud входят инфраструктурные сервисы, сервисы для управления данными, инструменты для разработки облачных приложений и модели машинного обучения.

AWS – это модуль для Node.js, который облегчает работу с облачным хранилищем Yandex Object Storage (S3 API). Модуль прост в установке и использовании, поддерживает авторизацию через AccessKeyId и SecretAccessKey, а также имеет возможность подключения к различным каталогам в Yandex Cloud. Yandex Object Storage предоставляет возможность хранить и управлять объектами в облаке, предоставляя аналогичный функционал, как у Amazon S3.

Файлы окружения (.env) – представляют собой текстовые файлы, содержащие конфигурационные переменные окружения для приложений. Они используются для хранения конфиденциальных данных, таких как: пароли, ключи API, учетные данные для баз данных и другие параметры, которые могут различаться в зависимости от окружения (например, разработка, тестирование, продуктивное окружение).

Express.js – это веб–фреймворк для Node.js, который помогает разработчикам создавать масштабируемые и эффективные веб–приложения. Он предлагает набор функций и инструментов, которые упрощают разработку и поддерживают лучшие практики для создания быстрых и надежных приложений.

Express.js был создан на основе Connect.js, другого популярного веб–фреймворка для Node.js. Он был разработан для того, чтобы помочь разработчикам быстро и легко создать веб–приложение, используя минимальный объем кода.

Express Validator – это библиотека для Node.js, которая помогает валидировать входящие запросы. Она обеспечивает простой и надежный способ проверки данных, передаваемых в API. Express Validator может использоваться для проверки структуры запроса, значений параметров, а также для выполнения проверок на основе бизнес–логики.

JSON Web Token (JWT) – это компактный, безопасный способ представления утверждений, которые можно передавать между двумя сторонами. Утверждения – это утверждения о сущности (обычно пользователе) и дополнительных данных. JWT обычно используется для авторизации и обмена информацией.

BCryptJS (Bcrypt JavaScript) – это библиотека для JavaScript, предназначенная для обеспечения безопасности паролей путем их хеширования. Она реализует алгоритм Bcrypt, который является одной из самых надежных и проверенных криптографических хеш–функций для паролей.

BCrypt был разработан как решение для проблемы хеширования паролей на основе обычного пароля, где злоумышленник может просто перебрать все возможные пароли, чтобы найти подходящий. Алгоритм Bcrypt вводит понятие «стоимости» (work factor), которое делает каждый перебор намного сложнее и дольше. Это означает, что чем больше «стоимость», тем больше времени потребуется злоумышленнику для взлома пароля путем перебора.

Mongoose – это открытый исходный код, основанный на JavaScript, объектно–реляционное сопоставление (ORM) для MongoDB. Он предоставляет функции, которые позволяют разработчикам использовать привычный синтаксис JavaScript и объектной модели для определения схем, а также для обработки запросов, обновлений, сохранений и удалений. Mongoose также обеспечивает встроенную поддержку для некоторых дополнительных функций, таких как валидация данных, отношения между документами, асинхронные операции и многое другое.

Mongoose имеет простой и понятный API с функциями, такими как Model.find(), которые позволяют легко выполнять запросы к базе данных. Он также предоставляет методы для обработки ошибок, проверки типов данных и других задач, связанных с обработкой данных.

Одним из главных преимуществ Mongoose является его простота и удобство использования. Благодаря своему простому синтаксису, разработчикам не требуется изучать сложные API или протоколы для работы с MongoDB. Mongoose также предлагает множество дополнительных функций, которые делают работу с MongoDB еще проще и удобнее.

Cross–Origin Resource Sharing (CORS) – это механизм, позволяющий веб–страницам, загруженным с одного домена, взаимодействовать с ресурсами, расположенными на других доменах. Это достигается путем добавления дополнительных HTTP–заголовков в ответы сервера, которые сообщают браузеру, что такое взаимодействие разрешено.

Основная цель CORS – предотвратить ограничения, вызванные политикой безопасности разных источников (same–origin policy), которая не позволяет скриптам на странице одного домена получить доступ к ресурсам на другом домене, чтобы предотвратить возможные атаки.

CORS поддерживается всеми современными браузерами и позволяет веб–приложениям, использующим XMLHttpRequest (XHR) для взаимодействия с серверами, отправлять HTTP–запросы на серверы, расположенные на других доменах (например, для загрузки изображений, скриптов или стилей).

Favicon (сокращение от «favorite icon») – это небольшая графическая иконка, которая обычно отображается в адресной строке браузера рядом с URL–адресом сайта или в закладках (вкладках). Основное предназначение фавикона – обеспечивать визуальное узнаваемое представление для пользователей сайта.

Обычно фавиконы имеют размер 16 на 16 пикселей, хотя также могут быть использованы и более крупные размеры. Они могут быть созданы в различных форматах, таких как ICO (Windows Icon), PNG, GIF или JPEG. Фавиконы помогают пользователям легко идентифицировать сайт среди множества открытых вкладок или закладок браузера, а также добавляют визуальный элемент брендинга.

Open Graph – это протокол метаданных, разработанный компанией Facebook, который позволяет веб–страницам предоставлять информацию для социальных сетей, когда пользователи делятся контентом. Основная цель Open Graph – обеспечить богатое и информативное отображение контента при его распространении в социальных сетях.

Выбор был остановлен на языке программирования JavaScript, с использованием Node.js, интегрированной среде разработки Visual Studio Code (далее – VS Code), и СУБД MongoDB.

Для контроля версий исходного кода и совместной работы над проектом будет использоваться система контроля версий Git.

Для создания фавикона (favicon) сайта была использована онлайн–платформа favicon.by [10].

Выбор указанных средств разработки обеспечит создание веб–приложения для организации дистанционного изучения образовательных курсов.

1.7 Требования к программному обеспечению и техническим средствам

Для обеспечения корректной и эффективной работы, разрабатываемого веб–приложения для организации дистанционного изучения образовательных курсов необходимо учесть следующие минимальные требования к программному обеспечению и техническим средствам как на стороне разработчиков, так и на стороне пользователей:

Требования к программному обеспечению (ПО) для пользователей:

* операционная система: Windows 10 / 11, macOS, Linux или мобильная операционная система Android, iOS;
* современный веб–браузер: Яндекс. Браузер, Google Chrome, Microsoft Edge, Safari или другой поддерживаемый браузер;
* стабильное интернет–соединение с достаточной скоростью для работы с веб–приложением.

Требования к программному обеспечению для разработки веб–приложения:

* операционная система – Windows 10 / 11, macOS или Linux;
* браузер – Яндекс. Браузер (при этом рекомендуется использовать три, или более трех браузеров для проверки работоспособности веб–приложения);
* графический редактор – Figma;
* интерфейс командной строки – Git Bash;
* среда разработки – Visual Studio Code;
* среда выполнения – Node.js;
* платформа вычислений – Java;
* инструмент для работы с базой данных – MongoDB Compass.

Требования к аппаратному обеспечению для разработки веб–приложения:

* процессор: современный многоядерный процессор с тактовой частотой не менее 2 ГГц;
* оперативная память: не менее 8 ГБ ОЗУ;
* жесткий диск: не менее 500 ГБ свободного пространства на SSD или HDD;
* видеокарта: интегрированная или дискретная видеокарта с не менее 2 ГБ видеопамяти.

Требования к серверному оборудованию для развертывания веб–приложения:

* процессор – не менее 2 ядер;
* оперативная память – не менее 4 гигабайт (ГБ);
* жесткий диск – не менее 50 гигабайт (ГБ) свободного пространства;
* операционная система – Linux или Windows Server.

Также для правильной работы скриптов программного продукта необходимо активное интернет–соединение и разрешение использования его в браузерах.

Учитывая эти минимальные требования, разработанное веб–приложение будет функционировать стабильно и обеспечивать удовлетворительное качество обучения и тестирования для пользователей, которые соответствуют указанным техническим характеристикам своих компьютеров и операционных систем.

# 2 РАБОЧИЙ ПРОЕКТ

**2.1 Логическая структура базы данных**

На основе полученной в техническом проекте концептуальной модели базы данных была получена физическая модель базы данных (таблицы 1–4).

Физическая модель данных описывает то, как данные хранятся в компьютере, представляя информацию о структуре, их упорядоченности и существующих путях доступа. Также описываются типы, идентификаторы и разрядность полей.

Для проектирования и реализации базы данных была использована нереляционная база данных MongoDB.

Таблица 1 – Физическая модель коллекции «Role»

|  |  |  |
| --- | --- | --- |
| Имя | Подпись поля | Тип данных |
| \_id | Идентификатор | ObjectId |
| name | Роль пользователя | String |

Таблица 2 – Физическая модель коллекции «User»

|  |  |  |
| --- | --- | --- |
| Имя | Подпись поля | Тип данных |
| \_id | Идентификатор | ObjectId |
| role | Роль пользователя | String |
| firstName | Фамилия пользователя | String |
| lastName | Имя пользователя | String |
| middleName | Отчество пользователя | String |
| username | Логин пользователя | String |
| password | Пароль | String |
| courses | Курсы | Array |

Таблица 3 – Физическая модель коллекции «Course»

|  |  |  |
| --- | --- | --- |
| Имя | Подпись поля | Тип данных |
| \_id | Идентификатор | ObjectId |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| category | Категория | String |
| name | Название курса | String |
| description | Описание курса | String |
| author | Автор курса | String |
| sections | Раздел курса | Array of Objects |
| sections.section\_name | Название раздела | String |
| sections.lessons | Уроки раздела | Array of Objects |
| sections.lessons.lesson\_name | Название урока | String |
| sections.lessons.lesson\_description | Описание урока | String |
| sections.lessons.course\_link | Ссылка на материал урока | String |

Таблица 4 – Физическая модель коллекции «Category»

|  |  |  |
| --- | --- | --- |
| Имя | Подпись поля | Тип данных |
| \_id | Идентификатор | ObjectId |
| name | Категория | String |

2.2 Функциональное взаимодействие модулей веб–приложения

На основе технического проекта было создано веб–приложение, предоставляющее образовательные курсы. Веб–приложение представляет собой многостраничный сайт, который содержит всю необходимую функциональность.

При разработке веб–приложения использовались следующе системные модули языка программирования JavaScript:

* DOM (Document Object Model) – представляет структуру HTML–документа в виде дерева объектов, где каждый элемент HTML, атрибут, текстовый контент и другие элементы документа представлены как объекты. DOM обеспечивает интерфейс, который позволяет программам на языке JavaScript изменять структуру, стиль и содержимое веб–страницы динамически;
* Fetch API – предоставляет простой и мощный интерфейс для отправки и получения HTTP–запросов. Он заменяет устаревший XMLHttpRequest и предоставляет более современный способ взаимодействия с сервером;
* Express.js – это минималистичный и гибкий веб–фреймворк для Node.js, который обеспечивает удобное создание веб–приложений и API;
* Amazon Web Services (AWS) – предоставляет набор инструментов, который позволяет разработчикам взаимодействовать с облачным сервисами AWS. В нашем случае, мы будем использовать данный инструмент для работы с Yandex Object Storage;
* Mongoose – это библиотека для удобной работы с базой данных MongoDB в Node.js.

Также были использованы следующие атрибуты модулей:

Из модуля DOM:

* id – уникальный идентификатор элемента;
* class – список классов элемента;
* src – адрес ресурса, такого как изображение;
* href – адрес гиперссылки;
* method – HTTP–метод запроса.

Из модуля Fetch API:

* fetch() – метод Fetch API для отправки HTTP–запросов.

Из модуля Express.js:

* app.get() – определяет обработчик для GET–запроса по указанному пути;
* app.post() – определяет обработчик для POST–запроса по указанному пути;
* app.use() – подключение для обработчика запросов.

Из модуля Mongoose:

* mongoose.connect() – подключение к кластеру MongoDB.

Для подключения модулей Express.js и Mongoose используется конструкции const и require.

Данная конструкция использовалась при реализации веб–приложения.

Для реализации авторизации пользователя:

* с помощью командной строки и команды «npm i название\_пакета» были установлены следующие менеджеры пакетов: Passport.js, JWT, bcrypt;
* Passport.js – для обработки запросов на вход в систему. Оно (промежуточное ПО) проверяет учетные данные пользователя (логин / электронная почта и пароль) и создает сессию пользователя;
* JWT (JSON Web Tokens) – формат токена для безопасной передачи информации между клиентом и сервером. После успешного входа в систему, сервер создает JWT который содержит уникальный идентификатор пользователя и другую информацию, и отправляет его клиенту. Клиент затем использует этот токен для подтверждения своей личности при выполнении последующих запросов к серверу;
* bcrypt – для хеширования паролей. Вместо хранения паролей в открытом виде, необходимо хранить хешированные версии, что повышает безопасность. Когда пользователь вводит пароль для входа в систему, bcrypt используется для сравнения хешированной версии введенного пароля с хешированной версией пароля, хранящегося в базе данных. Если хеши совпадают, пароль считается верным, и пользователь входит в систему.

Для реализации регистрации пользователя были использованы:

* Passport.js – данный пакет был использован при регистрации нового пользователя для проверки, что логин / электронная почта пользователя уникальны;
* bcrypt – после того, как Passport.js проверил учетные данные пользователя, было использовано bcrypt для хеширования пароля пользователя перед его сохранением в базе данных. Это помогает защитить пароль пользователя;
* JWT – после успешной регистрации и авторизации пользователя, сервер создает JWT и отправляет его клиенту.

Для реализации загрузки и хранения файлов:

* с помощью командной строки и команды «npm i easy–yandex–s3» был установлен пакет easy Yandex.S3;
* Yandex Cloud, а именно Object Storage – для облачного хранения файлов. Данная платформа предоставляет надежное хранение данных любого формата и объема;
* easy Yandex.S3 – для выполнения различных операций с файлами в облачном хранилище Object Storage. Она (библиотека) предоставляет: загрузку файлов, загрузку массивов файлов, скачивание файлов, отображение файлов.

Для кроссплатформенности веб–приложения был создан файл reset.css. Данный файл служит примером CSS сброса, который используется для единообразия стилей HTML–элементов на разных браузерах, удаляя стили по умолчанию, предоставляемые браузерами.

Большинство браузеров, по умолчанию, имеют разные стили. Это может привести к тому, что одно и то же веб–приложение будет выглядеть по–разному в разных браузерах.

Именно поэтому разработчики часто используют CSS «сбросы» (CSS resets), чтобы убедиться, что все элементы имеют единообразные стили во всех браузерах.

Для круглосуточной доступности веб–приложение было развернуто на хостинге.

Для контроля версий исходного кода и совместной работы над проектом были использованы следующие инструменты:

* система контроля версий Git;
* сервис онлайн–хостинга репозиториев – GitHub;
* интерфейс командной строки – Git Bash.

Для обеспечения безопасности данных веб–приложения было реализовано:

* шифрование данных (bcrypt);
* управление доступом – у каждого пользователя есть только те привилегии, которые ему действительно необходимы для работы с веб–приложением;
* авторизация;
* меры безопасности веб–приложения от хостинга: защита от DDoS–атак, резервное копирование данных, управление сервером.

Для создания логотипа веб–приложения с помощью Figma:

* был создан личный кабинет для работы в графическом редакторе;
* создан проект;
* использованы инструменты необходимые для создания логотипа;
* после завершения работы проект был экспортирован в формате .svg.

2.3 Файловая структура веб–приложения

Файловая структура веб–приложения состоит из файлов:

* HTML;
* CSS;
* JavaScript.

Так как для реализации проекта была использована среда выполнения Node.js, которая позволяет запускать JavaScript код вне браузера, а на сервере, то в директории node\_modules располагаются модули веб–приложения.

Файловая структура веб–приложения состоит из:

* «images» – содержит изображения, необходимые для правильного отображения дизайна веб–приложения такие как логотип и фавикон веб–приложения;
* «middleware» – содержит в себе файл authMiddleware.js, который выполняет аутентификацию пользователя по токену;
* «models» – содержит в себе файлы с моделями коллекций базы данных необходимые для работы с СУБД MongoDB;
* «node\_modules» – директория, содержащая все зависимости проекта, установленные с использованием менеджера проектов npm (Node Package Manager);
* «styles» – содержит файлы стилей для HTML–шаблонов и шрифты;
* «scripts» – содержит файлы сценариев на языке программирования JavaScript;
* package.json – файл, содержащий метаданные о проекте и его зависимости. В нем указываются основные параметры проекта, такие как название, версия, описание, точка входа (entry point) и многие другие;
* package–lock.json – этот файл создаётся автоматически при установке зависимостей с помощью команды «npm install». Он содержит точные версии каждого установленного пакета, а также информацию о их зависимостях.

Файловая структура корневой директории веб–приложения представлена на рисунке 6.

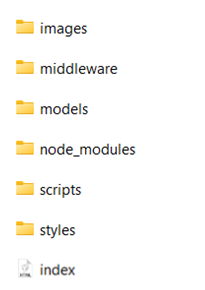


Рисунок 6 – Файловая структура корневого веб–приложения

2.4 Входные и выходные данные

При разработке веб–приложения для организации дистанционного изучения образовательных курсов были определены следующие входные и выходные данные.

Входные данные:

* для регистрации: имя, логин, пароль;
* для образовательных курсов: название, автор, описание, название раздела, название урока, учебный материал.
* для учебных материалов: текстовые файлы, видео, презентации.

Выходные данные:

* после успешной регистрации или авторизации в систему пользователь получает доступ к личному кабинету.

Веб–приложение будет обрабатывать данные в формате JSON (JavaScript Object Notation). Этот формат широко используется для обмена данными между клиентом и сервером в современных веб–приложениях благодаря своей компактности и легкости чтения и написания для человека и машины.

2.5 Работа с веб–приложением в качестве администратора

Веб–приложение было успешно размещено на хостинге с использованием домена, что позволяет интернет–пользователям посещать сайт в любое время. Хостинг – это услуга для хранения сайта. Домен необходим для хранения и отображения сайта в открытом доступе [3]. Сайт доступен по домену: cq04208.tw1.ru.

Для того чтобы начать работу с веб–приложением, необходимо перейти по адресу http://cq04208.tw1.ru/. Пользователь перейдёт на главную страницу веб–приложения, которая представлена на рисунке 7.

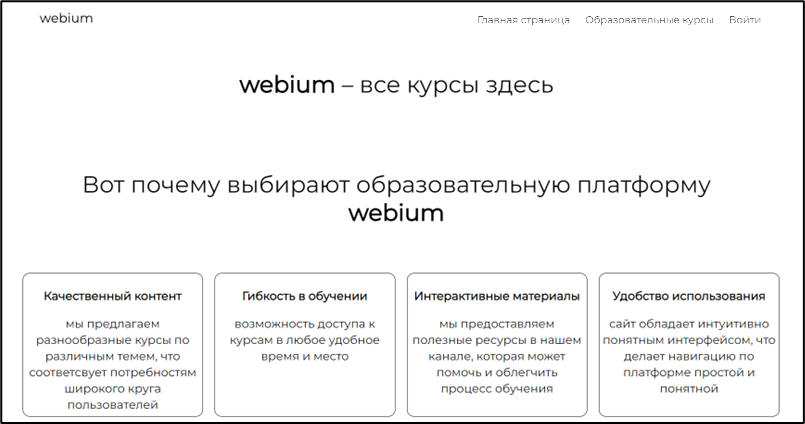


Рисунок 7 – Главная страница веб–приложения

При нажатии на кнопку «Войти» (расположена в навигационной панели главной страницы веб–приложения) пользователь перейдет на страницу авторизации. Данная страница представлена на рисунке 8.

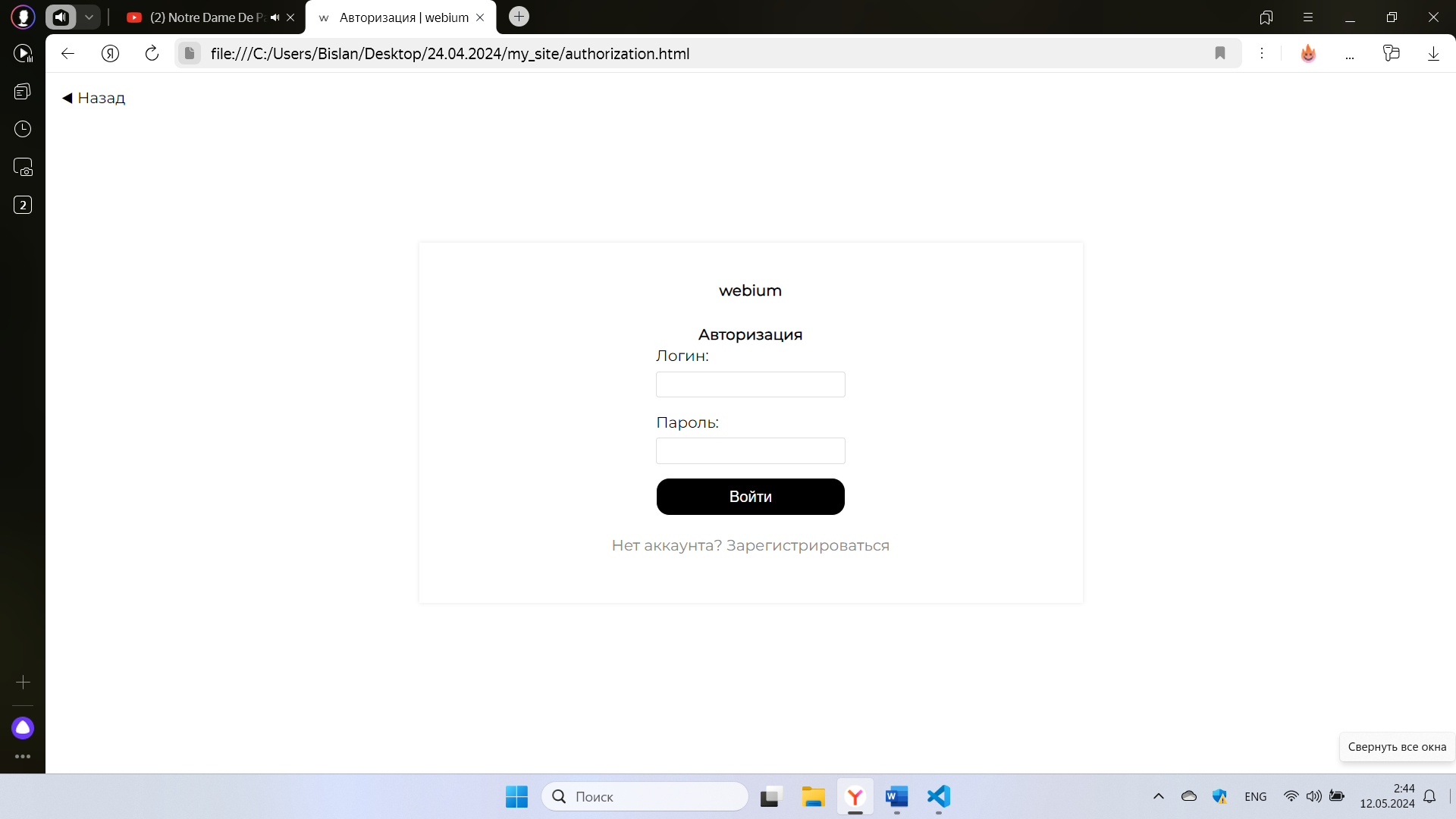


Рисунок 8 – Страница «Авторизация»

После успешной авторизации будет выполнен переход на административную панель для управления образовательными курсами и пользователями веб–приложения. Страница с административной панелью представлена на рисунке 9.

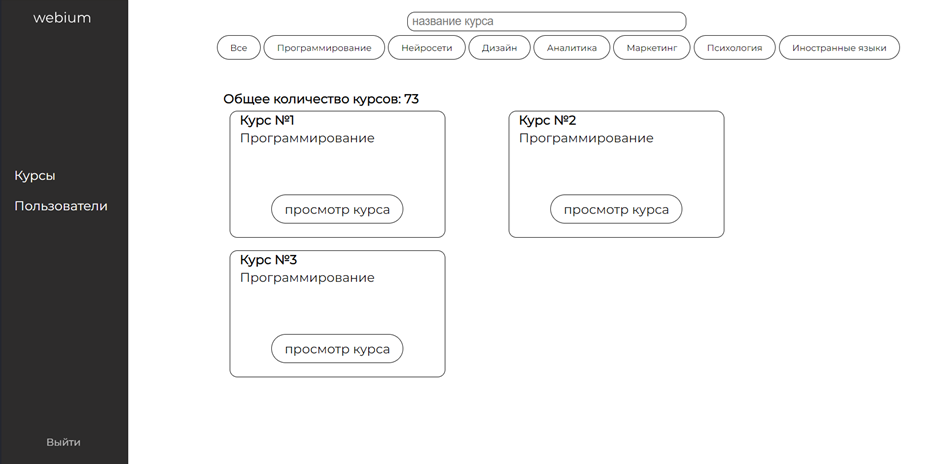


Рисунок 9 – Административная панель

2.6 Работа с веб–приложением в качестве разработчика

При вводе учетных данных разработчика будет выполнен вход в личный кабинет, где расположена рабочая панель. Личный кабинет разработчика представлен на рисунке 10.

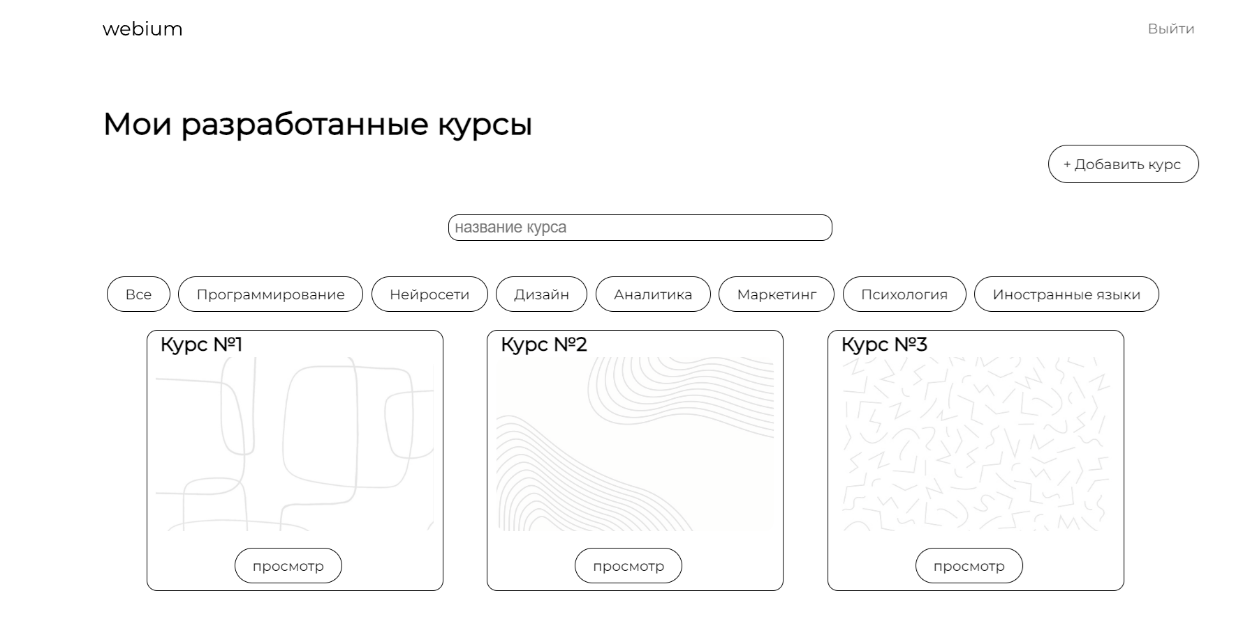
****

Рисунок 10 – Личный кабинет разработчика

Рабочая панель отображает разработанные курсы, а также предоставляет разработчику возможность добавления нового курса. Для этого в рабочей панели имеется кнопка «+ Добавить курс». При нажатии на неё, разработчик может ввести всю необходимую информацию о новом курсе и сохранить его в системе. Страница для добавления нового курса представлена на рисунке 11.

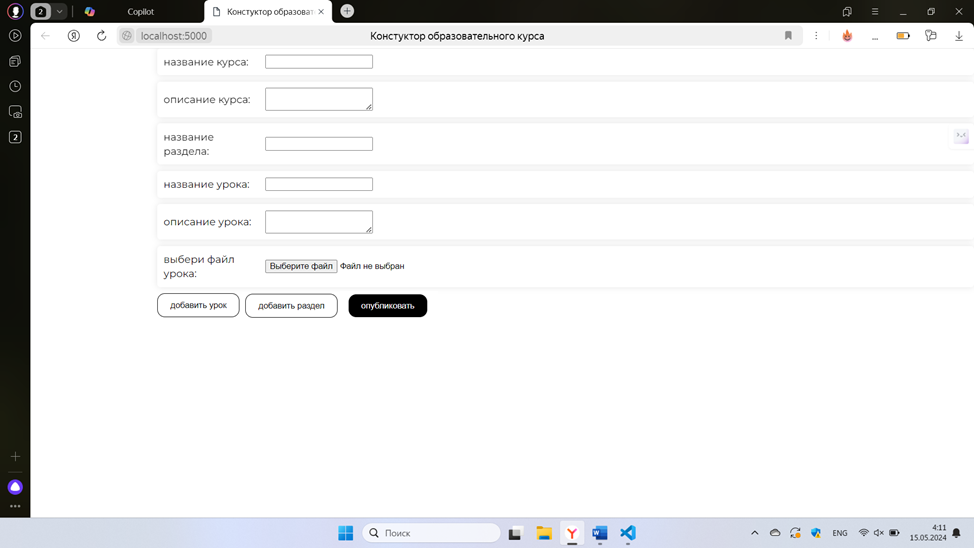


Рисунок 11 – Добавление нового курса

2.7 Работа с веб–приложением в качестве пользователя

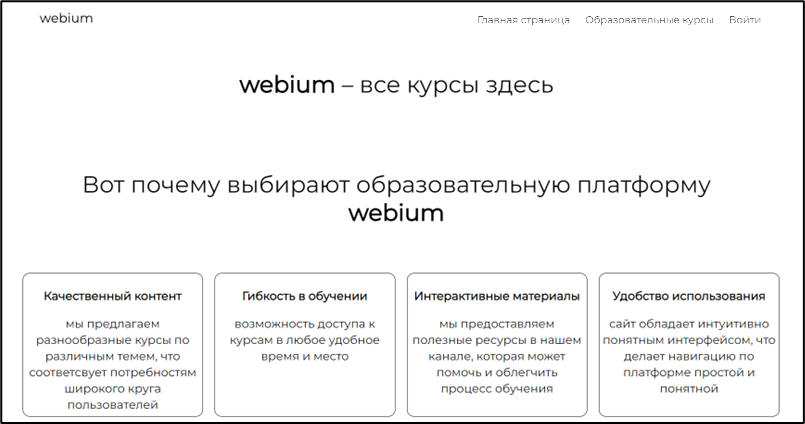


Рисунок 12 – Главная страница веб–приложения

При нажатии на пункт «Образовательные курсы» навигационной панели (главная страница веб–приложения) будет выполнен переход на страницу с доступными образовательными курсами. Данная страница представлена на рисунке 13.

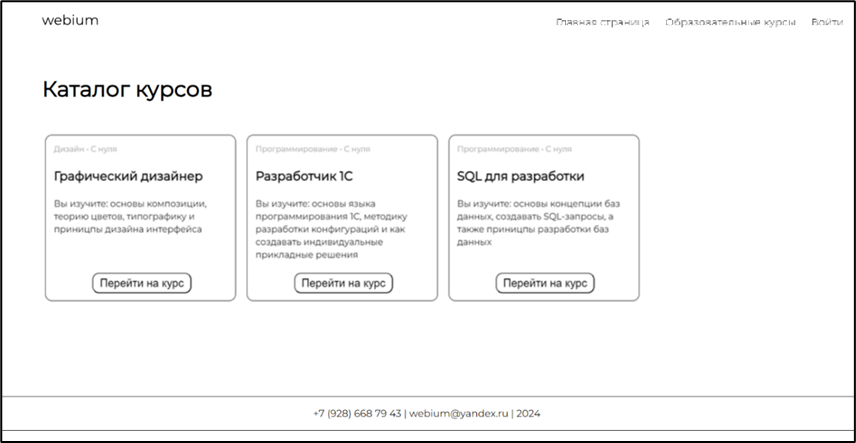


Рисунок 13 – Страница «Образовательные курсы»

Если пользователь не имеет учетной записи – он может легко создать её. Для этого необходимо нажать «Зарегистрироваться» на странице авторизации. Страница для создания учетной записи (регистрация нового пользователя) представлена на рисунке 14.

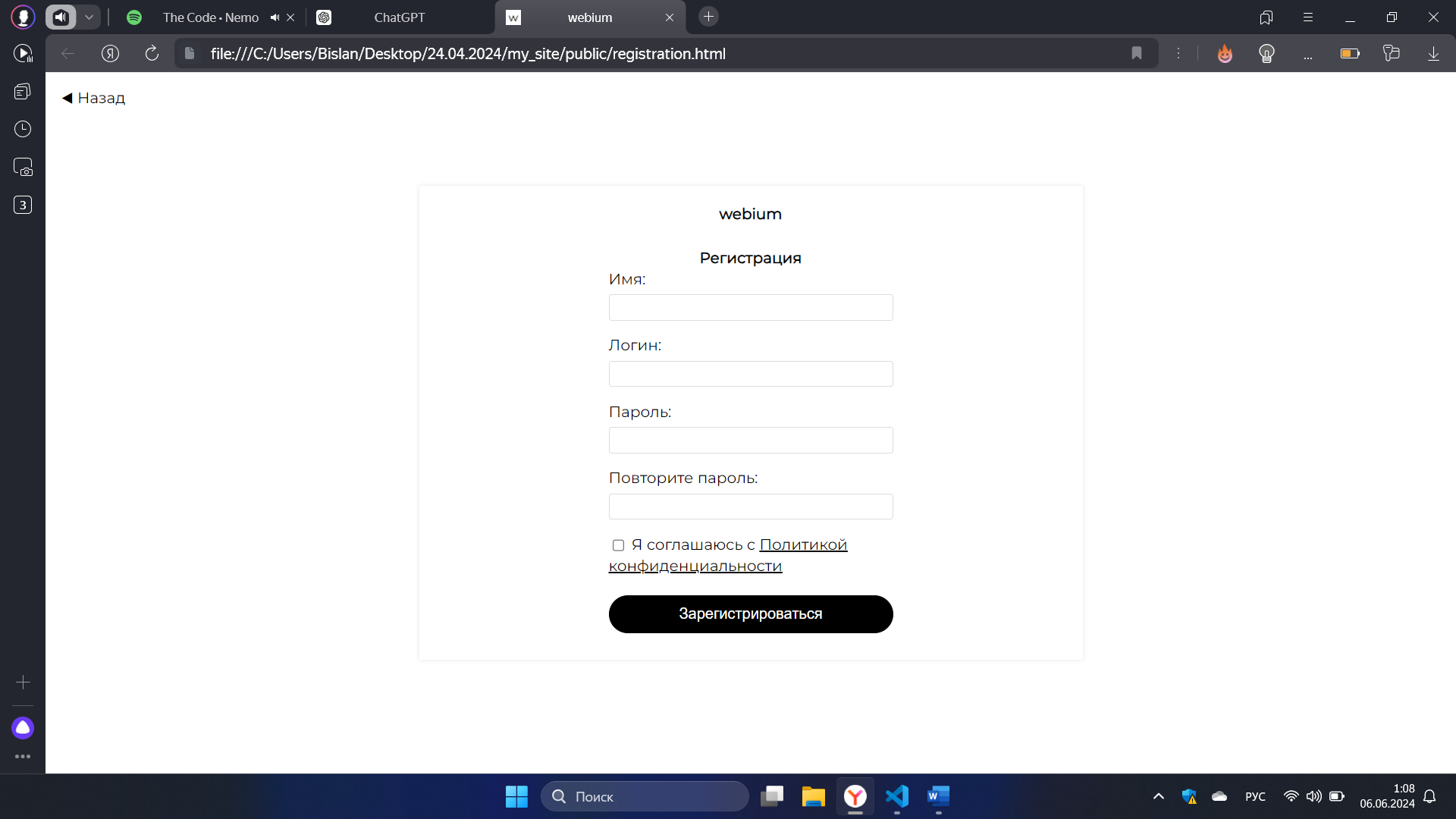


Рисунок 14 – Страница «Регистрация»

Страница «Регистрация» имеет все необходимые поля для создания новой учетной записи пользователя:

* имя пользователя;
* логин;
* пароль (с повторением введенного пароля);
* флажок (чекбокс) с политикой конфиденциальности, который показывает, что пользователь согласен с условиями обработки его персональных данных.

Если введенные данные совпадают с данными уже существующего пользователя, на странице появится сообщение: «Пользователь с такими данными уже зарегистрирован».

ЗАКЛЮЧЕНИЕ

Разработанное в рамках проекта веб–приложение для организации дистанционного изучения образовательных курсов позволяет создать курс, предоставляющий доступ к учебным материалам разного типа.

Разработанное веб–приложение имеет интуитивно понятный и адаптивный интерфейс, широкий набор инструментов для создания и управления образовательным контентом, гибкая система ролей и привилегий для различных категорий пользователей, а также масштабируемая архитектура.

В ходе работы были выполнены следующие задачи:

* выполнен анализ существующих решений для организации дистанционного изучения;
* проанализированы требования и функции веб–приложения;
* разработан дизайн интерфейса и структура курса;
* рассмотрены основные методы разработки веб–приложений;
* выбраны технологии для создания приложения;
* разработаны концептуальная, логическая и физическая модели базы данных;
* разработана база данных;
* разработан код веб–приложения;
* проведены тестирование и оптимизация веб–приложения;
* подготовлено руководство пользователя и документация по проекту.

Разработанное веб–приложение для дистанционного изучения образовательных курсов предоставляет современное и удобное средство обучения, которое помогает улучшить доступность образования. Реализованный проект может быть новым инструментом в области образования, который будет полезен широкому кругу пользователей, стремящихся к непрерывному самообразованию.

Таким образом, поставленная цель проекта – создание веб–приложения с доступом к изучение образовательных курсов и их материалам, позволяющее пользователям учиться в любое время, разработчикам управлять образовательными курсами, а администраторам управлять пользователями, была успешно достигнута.

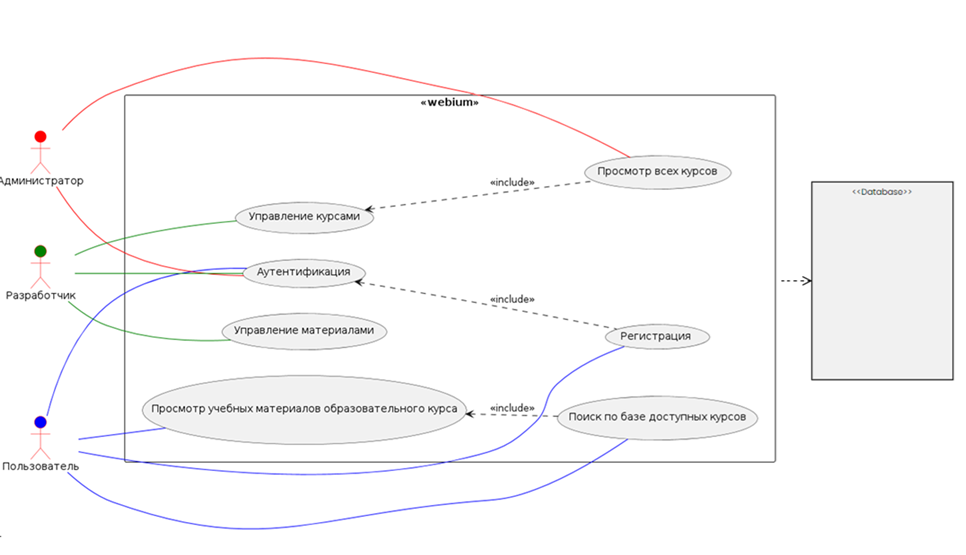
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Акбаров С. Т. Веб–приложения в эпоху современных требований: разработка и применение / С. Т. Акбаров; автор публикации Innovative Development in Educational Activities – Тблиси, 2023. – 30–37 с.
2. Браун Э. Веб–разработка с применением Node и Express: полноценные данные и интерфейсы / Э. Браун; автор. – Санкт–Петербург: Питер, 2023. – 336 с.
3. Гинзбург И. Б. Дистанционное обучение через Интернет с помощью специализированных веб–приложений / И. Б. Гинзбург, А. А. Ермаков, С. Н. Падалко – Казань: Научно–технический вестник Поволжья, 2021. – 19 с.
4. Гоголадзе Г. Д. Архитектура веб–приложений: современные концепции и методы / Г. Д. Гоголадзе; автор публикации Труды СПИИРАН, 2021. – 34–47 с.
5. Дакетт Д. React: Быстрый старт / Д. Дакетт; автор. – Москва: Питер, 2023. – 352 с.
6. Зайцева Т. В. Разработка и внедрение веб–приложения для дистанционного обучения / Т. В. Зайцева, А. Н. Петров; авторы публикации Современные информационные технологии и ИТ–образование, 2020. – 146–151 с.
7. Лепеха Э., Кобзев В. Г. Элементы веб–приложения для дистанционного изучения детей школьного возраста / Э. Лепеха, В. Г. Кобзев – Харьков: ХНУРЕ, 2020.
8. Маркин А. Постреляционные базы данных. MongoDB: учебное пособие / А. Маркин; автор. – Красноярск: Ipr Media, 2020. – 336 c.
9. Прохоренок Н. JavaScript и Node.js для веб–разрабочтиков: учебное пособие / Н. Прохоренок; автор. – Санкт–Петербург: Bhv–Cпб, 2022. – 768 с.
10. Рамирес П. Масштабирование Node.js приложений: практическое руководство / П. Рамирес; автор. – Санкт–Петербург: Питер, 2022. – 304 с.
11. Робин Р. Разработка современных веб–приложений: руководство для профессиональных веб–разработчиков / Р. Робин; автор. – Санкт–Петербург: БХВ–Петербург, 2022. – 688 с.
12. Савчиц И. О. Web–приложение для дистанционного изучения студентов по дисциплине «Организация и функционирование ЭВМ и ПУ» / И. О. Савчиц; автор публикации, 2019.
13. Использование CSS–анимации – CSS: каскадные таблицы стилей / MDN. URL: https://developer.mozilla.org/ru/docs/Web/CSS/CSS\_animations/ Using\_CSS\_animations (дата обращения: 21.05.2024).
14. Фавикон: что это такое, как сделать онлайн / Skillbox Media. URL: https://skillbox.ru/media/marketing/chto–takoe–favikon–i–kak–ego–sozdat/ (дата обращения: 11.05.2024).
15. <footer> – HTML / MDN. URL: https://developer.mozilla.org/ru /docs/Web/HTML/Element/footer (дата обращения: 02.05.2024).
16. Что такое онлайн–платформы и почему они делятся данными? / drfl. URL: https://drfl.kz/ru/online–platforms/ (дата обращения: 27.05.2024).
17. Какие бывают сайты? Виды сайтов по контенту, дизайну и функциональности / DENISOV. URL: https://sdvv.ru/articles/testovyy–razdel/kakie–byvayut–sayty–vidy–saytov–po–kontentu–dizaynu–i–funktsionalnosti/ (дата обращения: 14.05.2024).
18. Бесплатные курсы на русском языке: 16 онлайн–платформ / Тинькофф Журнал. URL: https://journal.tinkoff.ru/list/study–for–free/ (дата обращения: 02.05.2024).

**ПРИЛОЖЕНИЕ А**

**(справочное)**

**Use Case диаграмма проектируемого веб–приложения**

****

**ПРИЛОЖЕНИЕ Б**

**(справочное)**

**Листинг модуля index.js**

const express = require('express')

const cors = require('cors')

const mongoose = require('mongoose')

const authRouter = require('./authRouter')

const PORT = process.env.PORT || 5000

const app = express()

app.use(cors())

app.use(express.json())

app.use(express.urlencoded({ extended: true }));

app.use('/auth', authRouter)

const start =async () => {

try {

await mongoose.connect('mongodb+srv://a63282512:pBckr1CBnOuw6wcN@cluster0.cjpv3xq.mongodb.net/?retryWrites=true&w=majority')

app.listen(PORT, () => console.log(`server started on port ${PORT}`))

}

catch(e) {

console.error('Ошибка при запуске сервера:', e);

}

}

start()